C H A P T E R VI

RESUME

Before discussion of achievements of the attempt, it would be appropriate to explain the style in which listing of the software is presented. The subdivision is in terms of modules/submodules is defined in the flow chart 4.1 and 5.1. The listing, brought out utilizing wordstar facility, is titled with the module, submodule, level of the system operation as defined in the flow charts, and the corresponding flow charts. Each module describes essentials of the action executed within the module; the Anatomy. Further the usage of registers, reference memory location and subroutines called are indicated before the actual assembly language instruction part of the module.

At present the modules have been subjected to various test procedures, as described below.

1) Integrity of register usage :- Index and segment and pointer registers are crucial. Pair SI and DS is used to point interpreter code blocks while SS, BP, DI are used to point data items. Default segments pointed by SS and DS is segment 0. The modules over-riding the allocation scheme e.g. module : Resolve DIT, Level_2, F.C. 4.1, reset the pointers to the default values before transferring the control to another module. Further register usage flow charts, for inter and intra module operations were prepared to confirm integrity.

Table 4.12 : Identifier Fields of Variables And Numeric Constants

Field Bits :	a7	a6	a5	a4	a3	a2	a1	a0

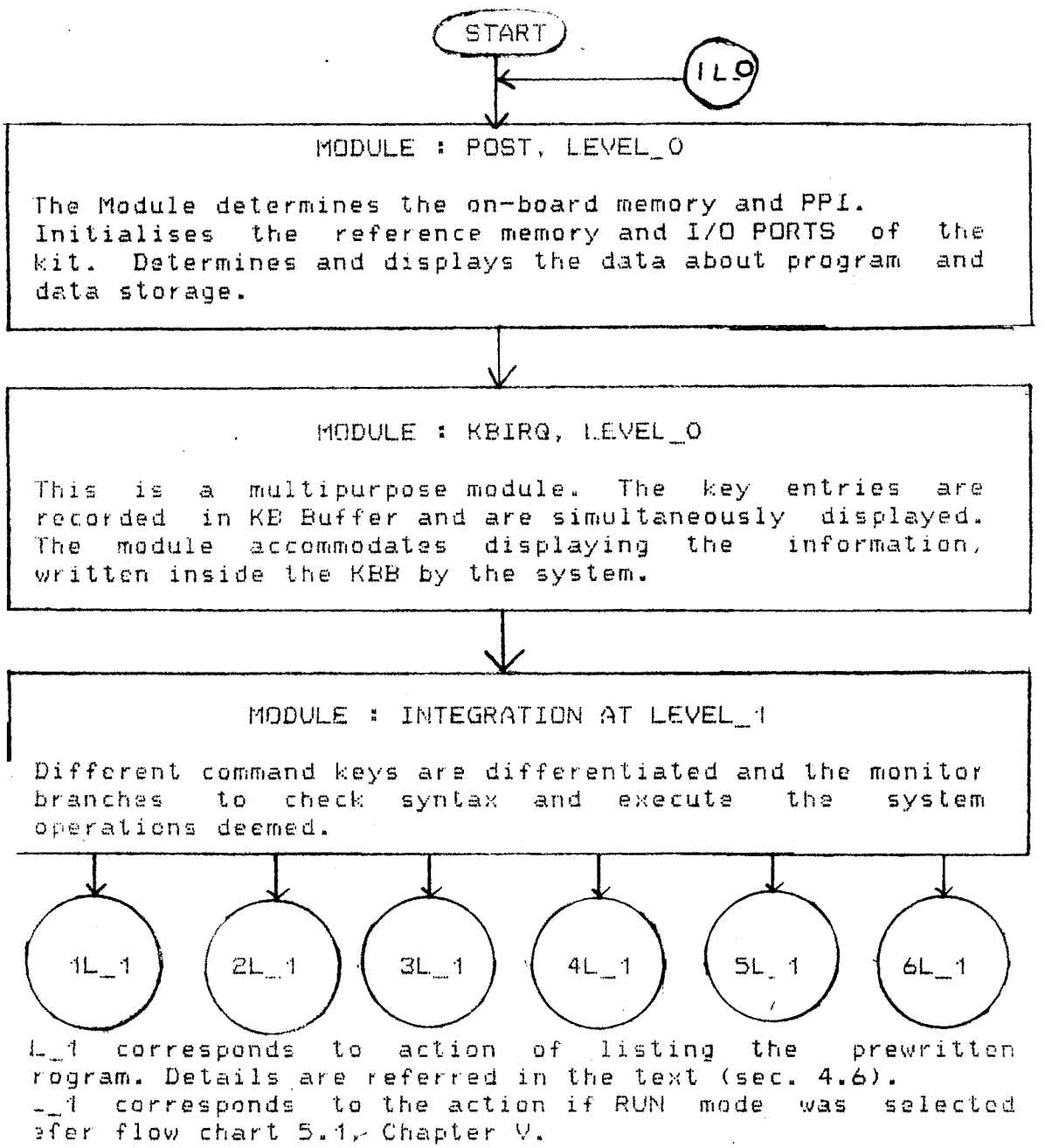
a1	a0	Specify Argument
0	0	Real Variable
0	1	Integer Variable
1	0	Real Numeric Constant
1	1	Integer Numeric Constant

a4	a3	a2	Specify Data Type
0	0	0	Default Type
0	0	1	Individual Type
0	1	0	16R or 8I Normal Type
0	1	1	24R or 16I Normal Type
1	0	0	16R or 8I Extended Type
1	0	1	24R or 16I Extended Type
1	1	0	16R or 8I Numeric Constants
1	1	1	24R or 16I Numeric Constants

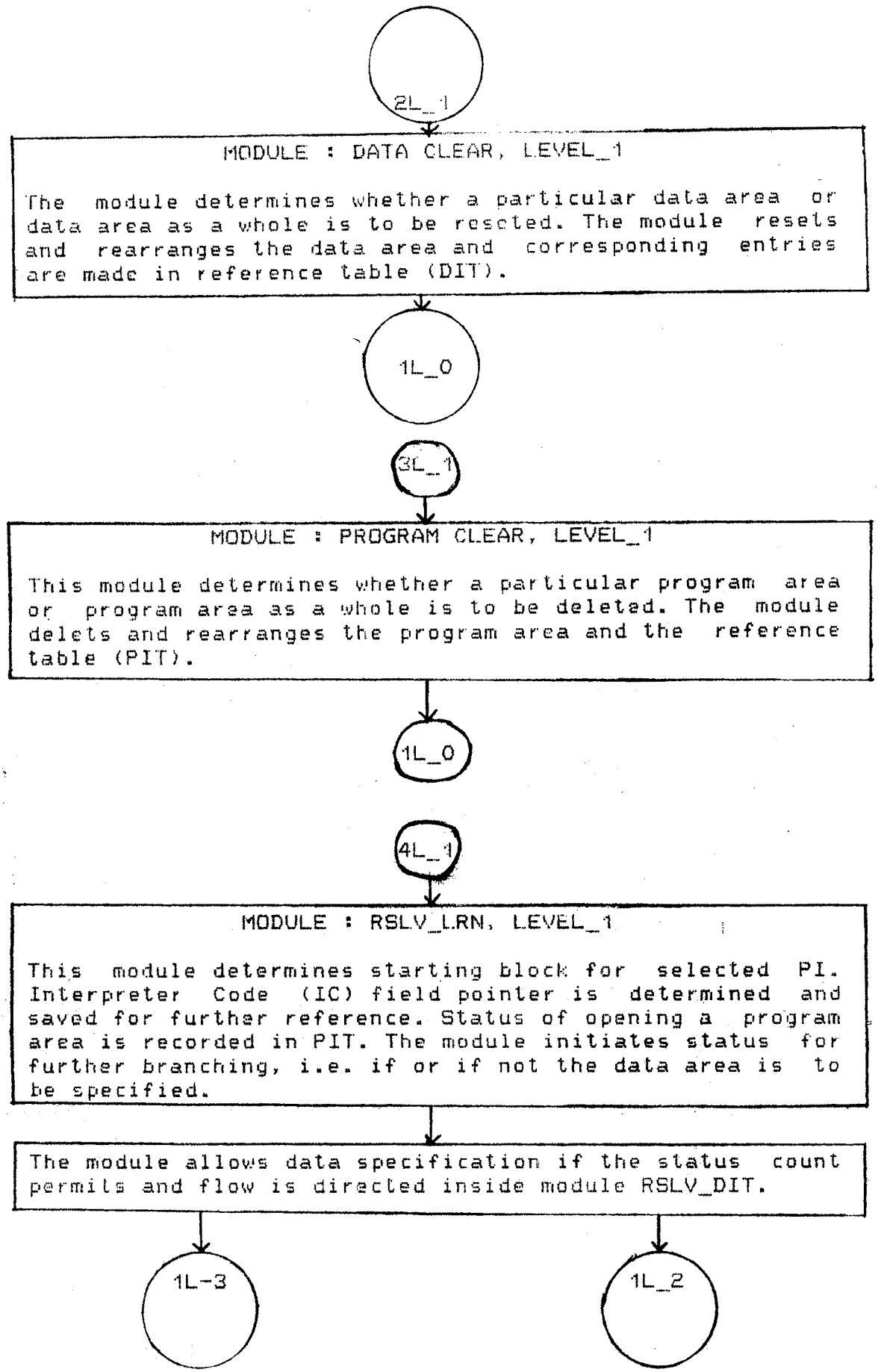
a7	a6	a5	Specify Index Type
0	0	0	Constant Valued Index
1	0	0	Without Index
0	1	1	Variable Default Type Index
1	0	1	Variable Normal Type Index
1	1	1	Variable Extended Type Index

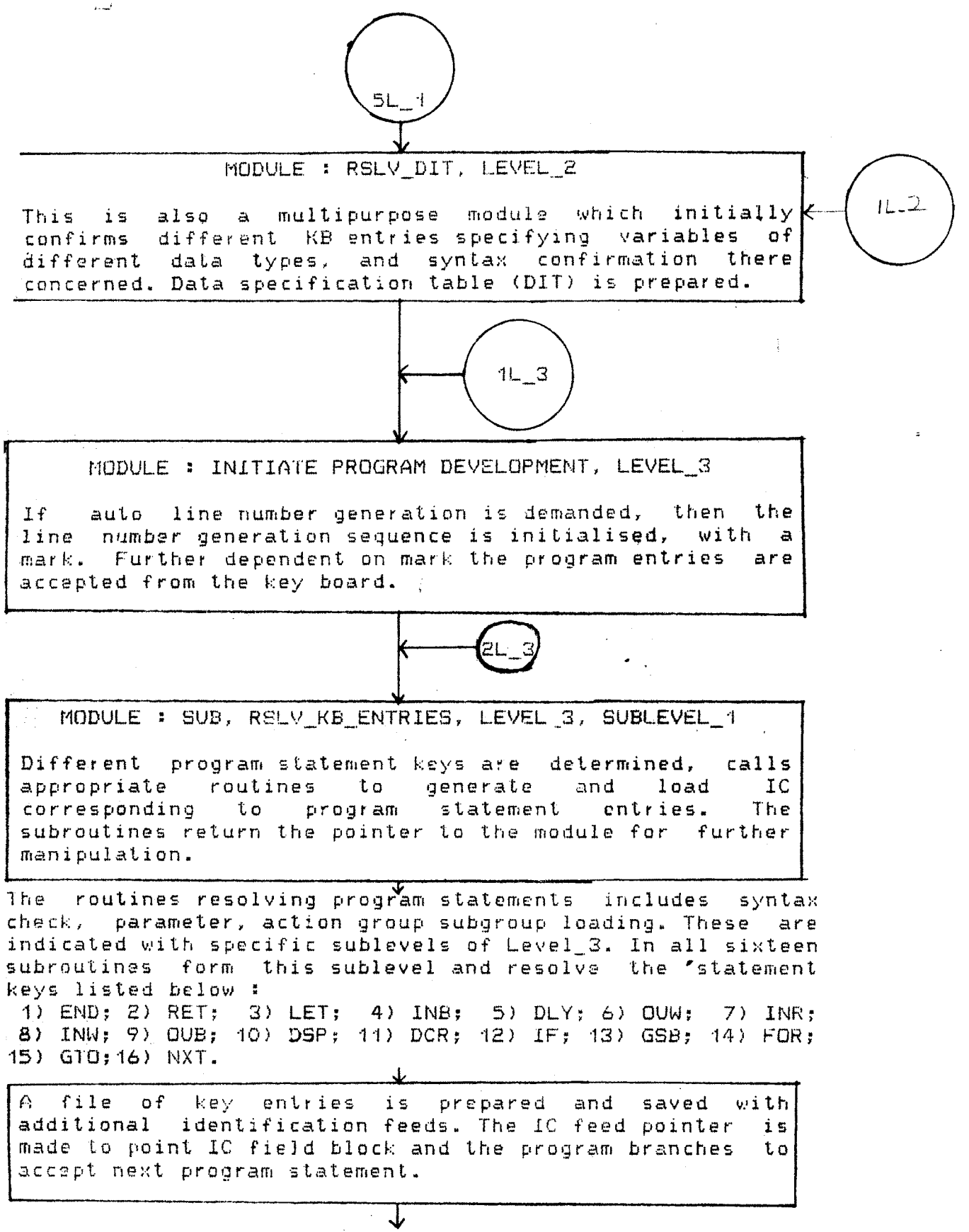
F L O W C H A R T 4.1

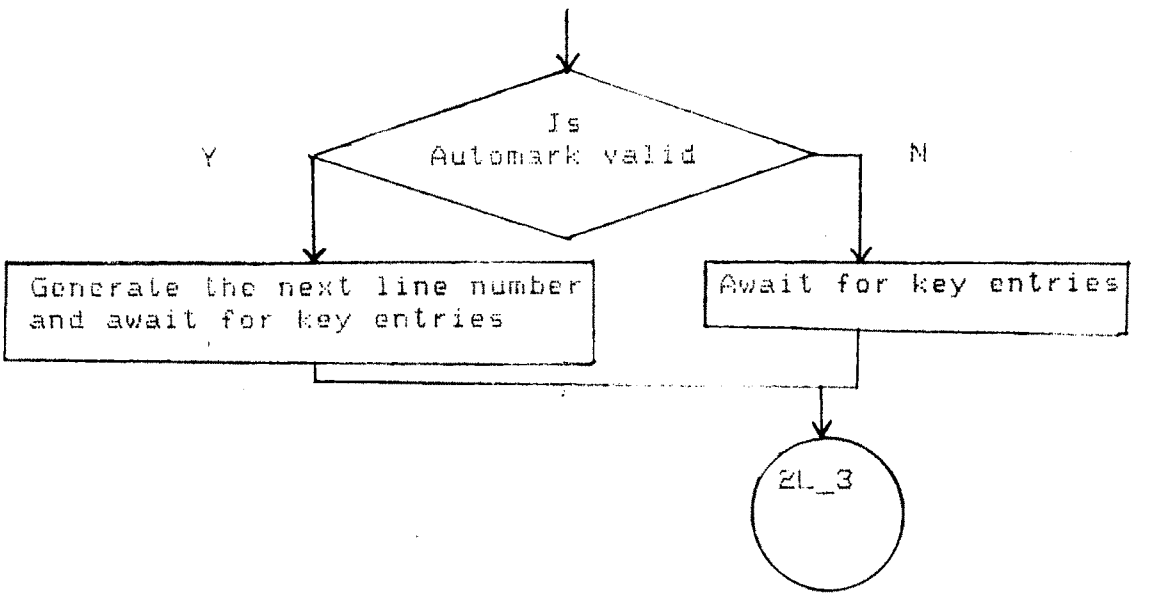
; Modules are bordered by rectangles. If the module a
: multifold structure, then the outgoing paths are indicated
: by down shoots, though not a standard convention.



L_1 corresponds to action of listing the prewritten rogram. Details are referred in the text (sec. 4.6).
_1 corresponds to the action if RUN mode was selected efer flow chart 5.1, Chapter V.


2L_1
MODULE : DATA CLEAR, LEVEL_1
The module determines whether a particular data area or data area as a whole is to be resetd. The module resets and rearranges the data area and corresponding entries are made in reference table (DIT).
1L_0
3L_1
MODULE : PROGRAM CLEAR, LEVEL_1
This module determines whether a particular program area or program area as a whole is to be deleted. The module delets and rearranges the program area and the reference table (PIT).
1L_0
4L_1
MODULE : RSLV_LRN, LEVEL_1
This module determines starting block for selected PI. Interpreter Code (IC) field pointer is determined and saved for further reference. Status of opening a program area is recorded in PIT. The module initiates status for further branching, i.e. if or if not the data area is to be specified.
The module allows data specification if the status count permits and flow is directed inside module RSLV_DIT.
1L-3
1L_2


5L_1
MODULE : RSLV_DIT, LEVEL_2
This is also a multipurpose module which initially confirms different KB entries specifying variables of different data types, and syntax confirmation there concerned. Data specification table (DIT) is prepared.
1L_2
1L_3
MODULE : INITIATE PROGRAM DEVELOPMENT, LEVEL_3
If auto line number generation is demanded, then the line number generation sequence is initialised, with a mark. Further dependent on mark the program entries are accepted from the key board.
2L_3
MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3, SUBLEVEL_1
Different program statement keys are determined, calls appropriate routines to generate and load IC corresponding to program statement entries. The subroutines return the pointer to the module for further manipulation.
The routines resolving program statements includes syntax check, parameter, action group subgroup loading. These are indicated with specific sublevels of Level_3. In all sixteen subroutines form this sublevel and resolve the 'statement keys listed below :
1) END; 2) RET; 3) LET; 4) INB; 5) DLY; 6) OUW; 7) INR; 8) INW; 9) OUB; 10) DSP; 11) DCR; 12) IF; 13) GSB; 14) FOR; 15) GTO; 16) NXT.
A file of key entries is prepared and saved with additional identification feeds. The IC feed pointer is made to point IC field block and the program branches to accept next program statement.


Is
Automark valid
Y
N
Generate the next line number
and await for key entries
Await for key entries
21_3

discussion about attach byte is relevant to the RUN mode and discussion is reserved for Chapter V. The next two bytes are the line number in packed BCD form. Line number 0000 is reserved for system to load program and data index applications, therefore maximum 9999 program statements could be loaded corresponding to an index. A provision for generating line number internally is also accomodated. The corresponding key entries syntax is 'AUTO' 'SEL' 'ENTER' or Auto Sel Line Number 'ENTER'. If the line number (LNO) is specified with the auto statement then it is used as present LNO and is displayed after unpacking the entries. The next LNO is generated by adding 0002 to the present LNO and is recorded and used as the present LNO for generating the next LNO. A fixed increment by 2 is only allowed as not more than 1 sentence could be added between the consecutive lines of an interpreted program. Next field in the interpreter code is action group and subgroup. Action groups are determined from key board entry while the subgroup from the actual entries within a sentence. Types of parameters to be loaded in the IC are action specific and are shown in Table 4.11. Before entering these codes all possible syntax errors like incorrect nesting of FOR - NXT structures, invalid use of variables etc. ..., are reported as error ERROR_1. (No error handling routines are developed presently). Entries except in case of LET group are self explanatory. In case of LET each variable entry is resolved and the variable identifiers are useful at RUN time are generated using DIT, Table 4.12. Further if numeric constants (NC) are

encountered these are converted into internal forms of representations and additionally identifiers of NC are also generated. As discussed in sec. 4.5 subgroups and conversion operators specifying conversion, before operating '=' operator, are also determined. These processes and processes of syntax check occurs simultaneously. The identifier fields, variable names or numeric entries are stored in Interpreter Code. While scanning the expression for the a/m actions the expression is also converted into the POST fixed notational form[44,16].

Primarily an 8 bytes block is considered to be interpreter codes corresponding to a sentence. If more than one 8 byte block is required for sentence entry, a delimeter FF is used as 9^{th} byte, to indicate continuation of the same sentence. If IC entries of a sentence end before 8 byte block boundry, zeros are inserted in the remaining part of the current block. System returns to Level_0 upon a reset.

At present as a concluding remark the structure developed for IC and operational structure is undergoing thorough estimates of speed, code optimizations and redesign etc. The software is tested at the level of individual modules only. As a step ahead the system is proposed to be emulated and tested on a PC/XT.

Table 4.1 : INPUT & OUTPUT STATEMENTS.

Statement No.	Syntax	Action [Code]		Action
		Group	Subgroup	
1.	'INB' 'KB'	A0	00	(AL) = (DATA PORT KBDC)
2.	'INB' 'VAR*'	A0	03	(VAR) = NC FROM KB
3.	'INB''ADD_H','ADD_L'**	A0	04	(AL) = (ADD_H, ADD_L)
4.	'INB''ADD_H','ADD_L','QR'#	A0	05	(QR) = (ADD_H, ADD_L)
5.	'INB' 'CH Q'##	A0	06	(AL) = (CH ADD)
6.	'INB' 'CH Q'##,'QR'	A0	07	(QR) = (CH ADD)
7.	'INW''ADD_H','ADD_L'	A8	00	(AX) = (ADD_H, ADD_L)
8.	'INW''ADD_H','ADD_L','QR'	A8	01	(QR) = (ADD_H, ADD_L)
9.	'OUB''ADD_H','ADD_L'	98	00	(ADD_H,ADD_L) = (AL)
10.	'OUB''ADD_H','ADD_L','QR'	98	01	(ADD_H,ADD_L) = (QR)
11.	'OUW''ADD_H','ADD_L'	B8	00	(ADD_H,ADD_L) = (AX)
12.	'OUW''ADD_H','ADD_L','QR'	B8	01	(ADD_H,ADD_L) = (QR)

* : VAR represents a variable reference (integer or real) for which address will be generated initially during RUN time.

** : 'ADD_H','ADD_L'is entered as 4 Hex entries representing address of a I/O PORT.

\# : QR represent integer type variable reference for which address will be generated internally during RUN time.

: CH Q, Q represents index of the channel to be selected. Address of the channel is generated during LRN time.

Table 4.2 : Comparison Operators

Sr.No.	Operation	Symbol
1)	Equal To	=
2)	Not Equal To	≠
3)	Greater Than	>
4)	Greater Than Or Equal To	>=
5)	Less Than	<
6)	Less Than Or Equal To	<=

Table 4.3 : Conditional Statements.

i) IF - THEN :

The actions are decisions based on comparison of magnitudes. In the action field of the Table magnitudes to be compared are given. If condition is satisfied jump is executed.

M1 : Magnitude ; on LHS
M2 : Magnitude ; on RHS

(Nos.continued from Table 4.1)

Statement No.	Syntax	Action [Code]		Action
		Group	Subgroup	
13.	'IF''ADD_H' 'ADD$_L$' 'OPRC'# 'DATA_8' 'THEN' 'LNO'	DO	00	M1 = (ADD_H ADD$_L$) M2 = (DATA_8)
14.	'IF''ADD_H' 'ADD$_L$' 'OPRC' 'VAR''THEN''LNO'	DO	01	M1 = (ADD_H ADD$_L$) M2 = (VAR)
15.	'IF' 'CH Q' 'OPRC''DATA_8' 'THEN''LNO'	DO	02	M1 = (CH Q) M2 = DATA_8
16.	'IF' 'CH Q' 'OPRC' 'VAR' 'THEN' 'LNO'	DO	03	M1 = (CH Q) M2 = (VAR)
17.	'IF' 'KB' 'OPRC''DATA_8' 'THEN''' 'LNO'	DO	04	M1 = (KBDC) M2 = DATA_8
18.	'IF' 'QR' 'OPRC' 'DATA_8' 'THEN''LNO'	DO	05	M1 = (QR) M2 = DATA_8
19.	'IF' 'VAR'$_L$ 'OPRC' 'VAR'$_H$ 'THEN''LNO'	DO	06	M1 =(VAR$_L$) M2 =(VAR$_H$)

: OPRC is a conditional operator as shown in Table 4.2.

ii) FOR_NXT : The statement within the structure are iterated.

Statement No.	Syntax	Action [Code]		Action
		Group	Subgroup	
20.	'FOR' 'Q' 'n' 'TO''m' (n < m)	E8	00	(Q) = Initial value n. (Q) = Final value m.
21.	'NXT' 'Q'	F8	00	(Q) = (Q)+1 if n <m and iterate the sentences enclosed.

iii) GTO :

Statement No.	Syntax	Action [Code]		Action
		Group	Subgroup	
22.	'GTO' 'LNO'*	F0	00	Back jump
		F0	01	Forth jump

* : LNO, 4 BCD entries representing the line number where the flow should branch, if Branch LNO > present LNO or if Branch LNO < present LNO.

iv) GSB - RET :

Statement No.	Syntax	Action [Code]		Action
		Group	Subgroup	
23.	'GSB' 'LNO'	D8	00	A program enclosed between LNO and RET is called.
24.	'RET'	88	00	The program flow resumes next LNO after 'GSB' sentence.

Table 4.4 : Allowed Data Types.

s	represent sign	
0	positive	
1	negative	
8 bit	INTEGER	8I
16 bit	INTEGER	16I
16 bit	REAL	16R
24 bit	REAL	24R

Data type	Representation	Range of Magnitude
8I	s 7 bit natural binary	+127 to -127
16I	s 15 bit natural binary	
16R	s 5 bit exponent 10 bit fraction (1<fract.<=0.5)	Max. $\pm$ 2999 X 10^{1} Min. $\pm$ 7999 X 10^{-9}
24R	s 6 bit exponent 17 bit fraction	Max. $\pm$ 199999 X 10^{4} Min. $\pm$ 199999 X 10^{-15}

Table 4.5 : The Data Specification.

Statement No.	Syntax	Action
1.	'DAT''SEL''Q'	Selects a particular data index.
2.	'EXT' 'LST'$_1$*	The variable listed are treated as array variables of dimension 256.
3.	'NOR' 'LST'$_1$	The variable listed are treated as array variables of dimension 16.
4.	'24R' 'LST'$_2$#	The variable listed as array variable of 24 REAL data type.
5.	'16R' 'LST'$_2$	The variable listed as array variable of 16 REAL data type.

* : LST_1 represents list of variables in ascending order (the entries of R and I type are to be the consecutive characters).

\# : LST_2 represents list of variables in ascending order.

Table 4.6 : Example of Data Specification.

'DAT' 'SEL' '1'		Selects data area index 1.
'EXT'	'F,G,H,I,X,Y,Z'	
'NOR'	'C,D,U,V,W'	
'24R'	'B,D,H,I'	
'16I'	'S,T,W,Y,Z'	

After resolving the a/m sentences the variable allocation would be as below. :

Default Integer	Q,R
16 Integer	S
8 Integer Normal Array	U,V
16 Integer Normal Array	W
8 Integer Extended Array	X
16 Integer Extended Array	Y,Z
Default Real	A
24 Real	B
16 Real Normal Array	C
24 Real Normal Array	D
16 Real Extended Array	F,G
24 Real Extended Array	H,I

Table 4.7 : Allowed Modes Expressions.

LHS	RHS	Mode	Conversion Operator
24R	24R or 16R	Normal	44H
	16I or 8I	Mixed	45H
16R	16R	Normal	00H
	16I or 8I	Mixed	05H
16I	16R	Mixed	50H
	16I or 8I	Normal	55H
8I	8I	Normal	11H

Table 4.8 : The LET Statement.

Statement No.	Syntax	Action [Code]		Action
		Group	Subgroup	
25.	'LET''VAR''='CNV''Q'	CO	00	VAR --> Destination operand. AL or AX --> Source operand.
26.	'LET''VAR_L''=' 'CNV''Q','VAR_R	CO	01	VAR_L --> Desination operand. VAR_R --> Source operand
27.	'LET''VAR_L''=' EXPRESSION* 24R	CO	02	
	'LET''VAR_L''=' EXPRESSION* 16R	CO	03	
	'LET''VAR_L''=' EXPRESSION* 16I	CO	04	
	'LET''VAR_L''=' EXPRESSION* 8I	CO	05	

* : NC Numeric Constant
OPR Arithmatic Operator (-, +, *, /)
24R Allowed Variable Types ...
24R and 16R; 16R; 16I or 8I; 8I.

Table 4.9 : The DELAY Statement.

Statement No.	Syntax	Action [Code]		Action
		Group	Subgroup	
28.	'DLY''MAGNITUDE'	AO	xx	

Magnitude comprises of 4 BCD entries with decimal point. The decimal point specifies the least count to be 0.1 sec., 0.01 sec., 0.001 sec. Subgroup (xx) is also dictated by decimal point.

Table 4.1C : DIT

BASE_ADD_DIT +	00	Default 16I
	01	Count 16I
		:
		Elements
		:
		Mark 8NI
		Count 8NI
		:
		Elements
		:
		Count 16NI
		:
		Elements
		:
		Mark 8EI
		Count 8EI
		:
		Elements
		:
		Count 16EI
		:
		Elements
		:
		Default 24R
		Count 24R
		:
		Elements
		:
		Mark 16NR
		Count 16NR
		:
		Elements
		:
		Count 24NR
		:
		Elements
		:
		Mark 16ER
		Count 16ER
		:
		Elements
		:
		Count 24ER
		:
		Elements

Table 4.11 : IC Code Blocks.

```
   0000          ATTACH BYTE
   0001 :        LNO
   0002 :
   0003          ACTION GROUP + SUBGROUP
   0004 :        OPERAND PARAMETERS
        :        OR 00H
        :
   0007 :
:  0009          OR FFH IF THE OPERAND SPECIFICATION CONTINUES IN
:                THE NEXT 8 BYTE BLOCK.
:
:
:  000F
-- 0010
```

The operand specifications are as indicated below :

Statement No.	1	2	4
	ADD_L KBDC ADD_H KBDC	FIELD OF VAR VAR INDEX (if any)	ADD_L ADD_H FIELD OF VAR VAR INDEX (if any)

2) Reference Memory Usage :- Parameter references and the usage is confirmed in the similar way as of registers. The parameter memory being in segment 0, needs management of DS, and is achieved through defining specific macros.

3) Testing the modules through MASM.COM :- The modules, where devicing test data items is a bit straight forward, are tested for flow of logic (especially Level_3). Though this procedure hides the actual pointer management required on the kit, but it appears to be a key unlocking the testing sequences for the interlinked modules, either using memory management by the DOS or with fixed memory allocations. Testing of interlinked modules is in progress at present. Further redesigning the modules involving block movements of data such that these could be tested using fixed memory reference is also in progress. This part of testing is found too time consuming as defining appropriate data sets itself is an involved endeavour.

Now, to sum up the achievements and to sketch the further developments in the field we make an attempt to evaluate the outcome of the endeavour. The system introduces a concept where the experimental procedures in the scientific laboratories could be delegated to the kit prototypes. The kit prototypes are programmable ones, through a language structure similar to microsoft basic, and designed to fit directly the environment of the cryophysics laboratories. Newness of the system does not lie at the level of being programmable but the hardware and software

monitor is so tailored that the tasks defined in the cryophysics laboratories could be very easily programmed through higher level language, without having to know the machine language and associated hand assembling. Within cryophysics laboratories the experimental procedures are too time consuming and operators are needed to keep on monitoring various transduced voltages and keep on switching the control instruments. This demands the number of pins to be installed. Additionally to achieve temperature stabilization and to introduce known heat quantum into the sample the laboratory needs more than one current source to be maintained. Multichannel capabilities offered by the prototype not only reduces the count of volt meter or current source circuit assemblies needed, but the conditional branching and facilities to call user developed assembly language routines, also relieves scientists from burdenous task of monitoring various events simultaneously. Further as the data acquisition is also programmable the system results advantageous in few more respects as (1) determination of closely inter-related data sets in lesser time spans as compared to manually operated benches is possible, (2) improved accuracy of the measurement owing to the fact that larger number of data items could be acquired per/set and linear portions of the data set could be used to calculate the property being measured could be achieved. (3) Scientist could be relieved from using conversion table calculators and to manupilate the data quantities in terms of the properties being measured (4) If system controllable hardware is deviced,

maintaining levels of liquid N_2 or He in cryostats, controlling, atleast annunciating, Helium recovery system manifold or any other task of continuous monitoring could be delegated to the prototype.

The kit being defined to work in comparison with a PC/XT (PC in general) facilities of utilities offered on PC could be tapped without having to key-in the data to the PC. A few of these are given below.

1) Curve fitting the data obtained, through higher level language routines.

2) Getting documentation of data sets obtained; be the graphs or the data tables, etc.

Along with the hardware defination of the kit, circuit elements to have a IEEE796[35] (multi bus interface through PC is also defined. The processor being used in kit and PC (8086 or 8088) being equivalent at the software level, software development of user defined task could be executed within the environment of PC. Further the monitor program, the interpreter could be redesigned within the same environment if needed. The object code files thus generated could be transfered to dual port memory of the appropriate multimaster kit, if the laboratory has more than one setups measuring different properties.

Though not attempted at present a elaborate utility software, making PC to become overall executive of the distributed experimental set ups in the laboratory is under

process of development in the laboratory. To conclude we find the attempt encouraging and have opened a field for further research and development where the laboratory functioning will be fully executed by a computer, remaining user friendly.

TITLE : MODULE : POST; LEVEL_0, FLOW CHART(F.C.) 4.1

ANATOMY:

PART_1 : THE PROGRAM READS DIP CONFIGURATION SWITCH FOR DETERMINATION OF TOTAL NUMBER OF AVAILABLE 256 BYTES BLOCKS OF RAM AVAILABLE ON THE BOARD. FURTHER THE NUMBER OF 8255 (PPI) IS DETERMINED FOR INITIALISATION PURPOSE. FURTHER 8254 (TIMER_1), 8254 (TIMER_2) AND 8259 (PIC) ARE ALSO INTITIALISED.

REGISTER USAGE :

DX : SEGMENT ADDRESSED, PORT ADDRESSES OF VARIOUS DEVICES.
AL : COMMAND BYTES TO VARIOUS DEVICES.
BX : MEMORY BLOCKS INDICATING AVAILABLE MEMORY OF SYSTEM.
BL : NUMBER OF PPI.

REFERENCE MEMORY LOCATIONS :

TOTAL_NO_PPI, AV_MEM_BLOCKS

PART-2 : AS THE RAM IS BATTERY BACKED FOR CONFIRMATION OF PROGRAM INDEX TABLE (PIT) IS DONE. IF A MISMATCH OCCURS THEN COMPLETE PROGRAM AND DATA AREA IS RESET. PRESENT_PI, PRESENT_DI AND AV_MEM BLOCKS ARE DISPLAYED.

REGISTER USAGE :

AL : PRESENT_DI, PRESENT_PI
CX : COUNTER, TALLY OF MEMORY BLOCKS, CRUNCHING DISPLAY NIBBLES
DX & BX : TALLY OF MEMORY BLOCKS, TRANSMIT PARAMETER FOR KBIRQ

REFERENCE MEMORY LOCATION :

PRESENT_PI, PRESENT_DI

SUBROUTINE :

DEL_ALL, L_1, F.C. 4.1

PART_1 :

```
MOV   DX, 0000H
MOV   DS, DX
MOV   ES, DX
MOV   SS, DX
MOV   DX, DIP_PORT
IN    AL, DX
MOV   AH, AL
```

```
                        AND     AL, 03H
                        JNZ     LOAD_AREA_1
                        JMP     ERROR_1
LOAD_AREA_1 :           MOV     BX, 0080 - 0011H
;AVAILABLE BLOCKS OF MEMORY IS SUBSTRACTED BY 11H, WHICH IS  THE
;AREA USED BY MONITOR FOR ITS REFERENCE.
                        TEST    AL, 01H
                        JZ      LOAD_AREA_2
                        ADD     BX, 0080H
LOAD_AREA_2 :           TEST    AL, 02H
                        JZ      LOAD_AREA_3
                        ADD     BX, 0100H
LOAD_AREA_3 :           MOV     AV_MEM_BLOCKS, BX
                        MOV     AL, AH
                        SHR     AL                  , 2 TIMES
                        AND     AL, 03H
                        JNZ     NOS_PPI
                        JMP     ERROR_1
NOS_PPI :               MOV     BL, AL
                        INR     BL
                        MOV     TOTAL_NO_PPI
                        MOV     DX, CNTROL_WORD_PPI_1
                        MOV     AL, 10011001B
; PPI_1 IS INITIALIZED WITH PORTS ABC AS INPUT PORTS, IN MODE_0
                        OVT     DX, AL
; COMMAND WORD IS SENT.
                        DCR     BL
                        MOV     DX, CNTROL_WORD_PPI_2
                        MOV     AL, 10011001B
;PPI_2 IS INITIALIZED WITH PORTS ABC AS OUTPUT PORTS, IN MODE_0
                        OVT     DX, AL
                        DCR     BL
                        JZ      SET_TIMER
                        MOV     DX, CNTROL_WORD_PPI_3
                        MOV     AL, 10111101B
;PORT_A  IS  INITIALIZED  FOR INPUT & PORT_B  IS  INITIALIZED  FOR
;OUTPUT IN MODE_1 OF PPI_3.
                        OUT     DX, AL
                        DCR     BL
                        JZ      SET_TIMER
                        MOV     DX, CNTROL_WORD_PPI_4
                        MOV     AL, 10011001B
;PORT A & C INITIALIZED FOR INPUT IN MOVE_1 AND PORT_B INITIALIZED
;FOR OUTPUT IN MODE_0 OF PPI_4.
                        OUT     DX, AL
SET_TIMER :             MOV     DX, CNTROL_WORD_TIMER_1
                        MOV     AL, 01100111B
;COUNTER_1  SELECTED TO READ/WRITE MOST SIGNIFICANT BYTE  ONLY  IN
```

```
;MODE_3 AND BCD COUNTING.
                        OUT     DX, AL
                        MOV     DX, COUNTER_1_CNTROL_REG
                        MOV     AL, 10H
                        OUT     DX, AL
                        MOV     DX, CNTROL_WORD_TIMER_1
                        MOV     AL, 10010111B
;COUNTER_2  INITIALIZED  IN MODE_3, READ/WRITE  LEAST  SIGNIFICANT
;BYTE ONLY AND BCD COUNTING
                        OUT     DX, AL
                        MOV     DX, COUNTER_2_CNTROL_REG
                        MOV     AL, 80H
                        OUT     DX, AL
; PROGRAMMING 8254_2 FOR PRODUCING BEEP IS LEFT AT THE  DISCRETION
;OF  MONITOR   BEEP FREQUENCY IS CONTROLLED BY COUNTER_1  AND  BEEP
;DURATION   IS  CONTROLLED  BY  COUNTER_2.  INT_6  FROM   COUNTER_2
;TERMINATES THE BEEP.
SET_INTERRUPT :         MOV     DX, CNTROL_WORD_PIC      ; ICW_1
                        MOV     AL, 00011111B
;ICW_1  SPECIFIES  LEVEL  TRIGGERED MODE, A PIC AND  ICW_4  TO  BE
;LOADED AND WITH CALL ADDRESS INTERVAL OF 4
                        OUT     DX, AL
                        MOV     AL, 00100000B            ; ICW_2
;32ND INTERUPPT INITIALIZED & SELECTED.
                        OUT     DX, AL
                        MOV     AL, 00001101B            ; ICW_4
;BUFFERED MODE/MASTER AND NORMAL EOI
                        OUT     DX, AL
                        MOV     AL 11111111B             ; OCW_1
;MASK ALL INTERRUPTS.
                        OUT     DX, AL
                        MOV     AL, 00100000B            ; OCW_2
;NON_SPECIFIC EOI
                        OUT     DX, AL
                        MOV     DX, CONTROL_PORT_KBDC
                        MOV     AL, 00110100B
;DIVIDE CLOCK INPUT. 20 THE CLOCK INPUTED TO THE CLK PIN OF 8279
;WILL BE A PREDIVIDED SYSTEM CLOCK.
                        OUT     DX, AL
                        MOV     AL, 00001000B
;ENCODED SCAN KB, 2 KEY LOCK OUT MODE.
                        MOV     DX, AL
;RESET OF REFERENCE MEMORY LOCATION
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     SI, BASE_ADD_REF_MEM_LOC
                        MOV     (SI), 000AH              ; MUL_WORD_0
                        ADD     SI, 0002H
```

```
                        MOV     (SI), 0064H             ; MUL_WORD_1
                        ADD     SI, 0002H
                        MOV     (SI), 03E8H             ; MUL_WORD-2
                        ADD     SI, 0002H
                        MOV     (SI), 2710H             ; MUL_WORD_3
                        ADD     SI, 0002H
                        MOV     (SI), 3CD4H             ; MUL_WORD_4
                        ADD     SI, 0002H
                        MOV     (SI), 3D09H             ; MUL_WORD_5
                        ADD     SI, 0002H
                        MOV     (SI), 5F5EH             ; MUL_WORD_6
                        ADD     SI, 0002H
                        MOV     (SI), EF6BH             ; MUL_WORD_7
                        MOV     CX, 0032H
LOAD_ZERO:              INR     SI
                        MOV     (SI), 00H
                        LOOP    LOAD_ZERO
```

PART_2 :

```
;PROGRAM CONFIRMS THE SPACE USED FOR PROGRAM STORAGE
                        MOV     BX, 0000H
                        MOV     SI, BASE_ADD_PIT
                        MOV     BP, 0000H
                        MOV     DI, BASE_ADD_KBB
                        MOV     DX, (SI)
;TOTAL NUMBER OF BLOCKS USED BY ALL PROGRAM INDICES (PI)
                        MOV     CX, 000AH
                        ADD     SI, 0020H
NXT_1 :                 ADD     SI, 0010H
                        ADD     BX, (SI)
;BLOCK NUMBERS FOR EACH PI
                        LOOP    NXT_1
                        CMP     BX, DX
                        JZ      NXT_3
                        AND     SI, FF00H
                        MOV     CX, AV_MEM_BLOCKS
                        CALL    DEL_ALL
;A PROGRAM WHICH RESETS ALL THE PROGRAM AND DATA AREA IS CALLED.
NXT_3:                  AND     SI, FF00H
                        ADD     SI, 0002H
                        MOV     BX, 0000H
                        MOV     CL, (SI)
                        ADD     SI, CX
                        MOV     AL, (SI)
                        MOV     PRESENT_PI, AL
                        SUB     SI, CX
```

```
                        MOV     (BP+DI), 'LRN'
                        INR     BH
                        INR     DI
                        MOV     (BP+DI), AL
                        INR     BH
                        INR     DI
                        MOV     CL, AL
                        ADD     CL, 03H
                        SHL     CL              , 4 TIMES
                        ADD     SI, CX
                        MOV     AL, (SI)
; GET THE DATA AREA USED FOR PRESENT PI.
                        MOV     PRESENT_DI, AL
                        MOV     (BP+DI), 'DAT'
                        INR     DI
                        INR     BH
                        MOV     (BP+DI), AL
                        INR     BH
                        INR     DI
                        MOV     SI, BASE_ADD_DIT
; TO CALCULATE TOTAL PROGRAM AND DATA AREA.
                        MOV     CX, (SI)
                        SHR     CX
                        ADD     DX, CX
                        ADD     DX, 0011H
                        MOV     CX, AV_MEM_BLOCKS
                        SUB     CX, DX
                        MOV     DX, CX
                        AND     CH, F0H
                        MOV     (BP+DI), CH
                        INR     BH
                        INR     DI
                        MOV     CH, DH
                        AND     CH, 0FH
                        MOV     (BP+DI), CH
                        INR     BH
                        INR     DI
                        MOV     CL, DL
                        AND     CL, F0H
                        MOV     (BP+DI), CL
                        INR     DI
                        INR     BH
                        MOV     CL, DL
                        AND     CL, 0FH
                        MOV     (BP+DI), CL
; AVAILABLE  BLOCKS  OF MEMORY ARE  TRANSMITTED  TO  THE  DISPLAY
; ALONGWITH DEFAULT LRN AND DATA INDICES.
```

```
MACRO                   LED_INDICATOR_ON
                        MOV    DX, LED_INDICATOR_PORT
                        MOV    AL, STATUS_WORD_ON
                        OUT    DX, AL
                        ENDM
MACRO                   UNMASK KBIRQ
                        MOV    DI, BASE_ADD_KBB
                        MOV    DX, CONTROL_WORD_PIC        ; (OCW_1)
                        MOV    AL, FEH
                        OUT    DX, AL
                        ENDM
                        HLT
                        CMP    AL, 'ENTER'
                        JZ     CONT_0
                        JMP    ERROR_1
CONT_0 :                NOP
MACRO                   LED_INDICATOR_OFF
                        MOV    DX, LED_INDICATOR_PORT
                        MOV    AL, STATUS_WORD_OFF
                        OUT    DX, AL
                        ENDM
MACRO                   BLANK_ALL
                        MOV    DX, CONTROL_WORD_KBDC       ; (8279)
                        MOV    AL, 10100011B
; COMMAND WORD INDICATING ALL DISPLAY TO BE BLANKED.
                        OUT    DX, AL
                        ENDM
                        UNMASK KBIRQ
                        MOV    AL, (BP+DI)
                        INT    VECTOR_KBIRQ
```

TITLE : MODULE : KBIRQ, LEVEL_0, F.C. 4.1

ANATOMY :

THIS IS A MULTIPURPOSE ROUTINE MEANT TO RECEIVE ENTRIES FROM KEYBOARD (KB) IN KEYBOARD BUFFER (KBB), LOAD BYTES FOR DISPLAY INTO DISPLAY BUFFER, TRANSMIT THE BYTES FROM DISPLAY TO KBDC SUCH THAT DISPLAY OCCURS IN MODIFIED LEFT ENTRY TTY MODE. IF THE NUMBER OF BYTES TO BE DISPLAYED IS GREATER THAN 16 THEN LAST 16 BYTES, INCLUDING CURSOR, ARE TRANSMITTED TO KBDC. FORWARD AND BACKWARD CURSOR MOVEMENT IS ALSO ACCOMODATED IN PROGRAM THEN SAME ROUTINE IS USED FOR DISPLAYING THE USER DEFINED DISPLAY ENTRIES OR ACQUIRING NUMERIC DATA AT THE RUN_TIME OR POST.

REGISTER USAGE :

DX : ADDRESSES OF PORTS.
BH : NUMBER OF BYTES IN KBB.
BL : NUMBER OF BYTES DISPLAYED.

REFERENCE_MEMORY_LOCATIONS :

DISPLAY_VALUE_TABLE

```
BACK :          MOV   DX, CONTROL_PORT KBDC
                MOV   AL, 50H
                OUT   DX, AL
                IN    AL, DX
                TEST  AL, 20H
                JZ    NO_OVERRUN
                MOV   DX, LED_INDICATOR_PORT
                MOV   AL, OVERRUN_STATUS_BYTE
                OUT   DX, AL
NO_OVERRUN :    AND   AL, 0FH
                MOV   CH, 00H
                MOV   CL, AL
                CMP   CL, 00H
                JNZ   NXT_0
                CMP   BH, BL
                JZ    BACK
                JMP   DISP
NXT_0 :         MOV   DX, DATA_PORT_KBDC
REPET :         IN    AL, DX
                CMP   AL, '-->'
                JE    NXT_1
                CMP   AL, '<--'
                JE    NXT_2
```

```
                    CMP     AH, 00H
                    JZ      NXT_3
                    CMP     AH, '↑'                 ; UP_CURSOR
                    JE      NXT_4
                    CMP     AH, '↓'                 ; DOWN_CURSOR
                    JE      NXT_4
NXT_3 :             CMP     AL, '↑'                 ; UP_CURSOR
                    JB      NXT_5
                    CMP     AL, '↓'                 ; DOWN_CURSOR
                    JA      NXT_5
                    JMP     ERROR_1
NXT_5 :             MOV     (BP+DI), AL
                    INR     DI
                    INR     BL
                    LOOP    REPET
                    CMP     AL, 'ENTER'
                    JZ      NXT_51
                    JMP     DISP
NXT_51 :            IRET
NXT_1 :             MOV     (BP+DI) 'BL'        ; LEFT BLANK = 'BL'
                    INR     DI
                    INR     BL
                    JMP     DISP_1
NXT_2 :             DCR     DI
                    CMP     BL, 00H
                    JZ      NXT_6
                    DCR     BL
                    DCR     DI
                    JMP     DISP_1
NXT_6 :             MOV     AL, (BP+DI)
                    CMP     AL, 00H
                    JNZ     NXT_61
                    JMP     BACK
NXT_61              CMP     AL, 3CH
                    JB      NXT_7
                    DCR     DI
                    MOV     (SI), 'BL'
                    SUB     SI, 0003H
                    ADD     (SI), '.'
                    MOV     CL, 04H
                    SUB     BH, 03H
                    MOV     DX, CONTROL_PORT_KBDC
                    CMP     BH, 0DH
                    JAE     NXT_8
                    MOV     AL, BH
                    ADD     AL, 70H
                    OUT     DX, AL
                    MOV     DX, DATA_PORT_KBDC
```

```
REPET_1 :       MOV   AL, (SI)
                OUT   DX, AL
                INR   SI
                LOOP  REPET_1
                SUB   SI, 0004H
                STI
                HLT
NXT_8 :         MOV   AL, 7DH
                OUT   DX, AL
                MOV   DX, DATA_PORT_KBDC
REPET_2 :       MOV   AL, (SI)
                OUT   DX, AL
                INR   SI
                LOOP  REPET_2
                SUB   SI, 0004H
                JMP   BACK
NXT_7 :         DCR   DI
                MOV   (SI), 'BL'
                DCR   SI
                ADD   (SI), '.'
                DCR   BH
                CMP   BH, 0FH
                JAE   NXT_9
                MOV   AL, BH
                ADD   AL, 70H
                OUT   DX, AL
                MOV   DX, DATA_PORT_KBDC
                MOV   AL, (SI)
                OUT   DX, AL
                INR   SI
                MOV   AL, (SI)
                OUT   DX, AL
                SUB   SI, 0002H
                JMP   BACK
NXT_9 :         MOV   AL, 7EH
                OUT   DX, AL
                MOV   DX, DATA_PORT_KBDC
                MOV   AL, (SI)
                OUT   DX, AL
                INR   SI
                MOV   AL, (SI)
                OUT   DX, AL
                SUB   SI, 0002H
                JMP   BACK
DISP_1 :        CMP   CL, 00H
                JZ    DISP
                MOV   DX, CONTROL_PORT_KBDC
                MOV   AL, C2H
```

```
                    OUT     DX, AL
DISP :              MOV     SI, BASE_ADD_DISP_BUFFER
                    MOV     CH, 00H
                    MOV     CL, BH
                    ADD     SI, CX
                    MOV     CL, DL
                    SUB     DI, CX
                    MOV     DX, DISPLY_VALUE_TABLE
                    MOV     BL, BH
REPET_3 :           MOV     AL, (BP+DI)
                    CMP     AL, 3CH
                    JAE     LOAD_3BYTES
                    XCHG    DX, BX
                    XLAT
                    XCHG    DX, BX
                    MOV     (SI), AL
                    INR     BH
                    INR     SI
                    INR     DI
                    JMP     NXT_62
LOAD_3BYTES :       XCHG,   DX, BX
                    XLAT
                    MOV     (SI), AL
                    INR     SI
                    INR     DH
                    INR     AL
                    XLAT
                    MOV     (SI), AL
                    INR     SI
                    INR     DH
                    INR     AL
                    XLAT
                    INR     SI
                    INR     DH
                    INR     DI
                    XCHG    DX, BX
NXT_62 :            LOOP    REPET_3
                    INR     SI
                    MOV     (SI), '.'
                    CMP     BH, 10H
                    JAE     NXT_10
                    MOV     CL, BH
                    SUB     CL, BL
                    SUB     SI, CX
                    MOV     DX, CONTROL_PORT_KBDC
                    MOV     AL, BL
                    ADD     AL, 70H
                    OUT     DX, AL
```

```
                        MOV     DX, DATA_PORT_KBDC
                        INR     CL
NXT_11 :                MOV     AL, (SI)
                        OUT     DX, AL
                        INR     SI
                        LOOP    NXT_11
                        JMP     BACK
NXT_10 :                MOV     CL, BH
                        SUB     SI, CX
                        SUB     CL, OFH
                        ADD     SI, CX
                        MOV     DX, CONTROL_PORT_KBDC
                        MOV     AL, 70H
                        OUT     DX, AL
                        MOV     DX, DATA_PORT_KBDC
                        MOV     CL, 10H
NXT_12 :                MOV     AL, SI
                        OUT     DX, AL
                        INR     SI
                        LOOP    NXT_12
                        JMP     BACK
```

TITLE : MODULE : INTEGRATION AT LEVEL_1, F.C. 4.1

ANATOMY :

PROGRAM DIFFERENTIATES COMMAND KEYS, CONFIRMS SYNTAX AND CALLS RESPECTIVE SUB PROGRAMS. THESE ARE LST, DEL, VAC, LRN, DAT, RUN.

REGISTER USAGE :

AL : COMMAND KEY CODE, PARAMETERS LIKE PI, DI.
DL : DATA AREA INDEX (DI).
CX : POINTER SEGMENT, COUNTER.

REFERENCE MEMORY LOCATIONS :

PRESENT_DI, PRESENT_PI, ICP_LOC_SI, ICP_LOC_DS, MARK_CURRENT_OPEN.

SUBROUTINES :

RSLV_LRN, VAC_SEL, VAC_ALL, DEL_SEL, DEL_ALL.

```
                    MOV   AL, (BP+DI)
                    MOV   (BP+DI), 00H
                    CMP   AL, 'LST'
                    JB    NXT_10
                    JMP   CHK_VAC
NXT_10 :            CMP   AL, 'DAT'
                    JB    CHK_LRN
                    JNZ   NXT_11
                    JMP   CHK_DATA
NXT-11 :            CMP   AL, 'RUN'
                    JNZ   NXT_12
                    JMP   CHK_RUN
NXT_12 :            JMP   ERROR_1
CHK_LRN :           CMP   AL, 'LRN'
                    JZ    NXT_13
                    JMP   RSLV_DEF_LRN
NXT_13 :            INR   DI
                    CMP   (BP+DI), 'SEL'
                    JZ    NXT_14
                    JMP   ERROR_1
NXT_14 :            INR   DI
                    MOV   AL, (BP+DI)
                    CMP   AL, 09H
;PI BETWEE 0 TO 9 ARE ONLY ALLOWED.
                    JBE   NXT_15
```

```
                    JMP     ERROR_1
NXT_15 :            INR     DI
                    CMP     (BP+DI), 'ENTER'
                    JZ      NXT_16
                    JMP     ERROR_1
NXT_16 :            CALL    RSLV_LRN                ; L_1, F.C. 4.1
                    LED_INDICATOR_ON
                    UNMASK_KBIRQ
                    HLT
                    CMP     AL, 'ENTER'
                    JZ      CONT_1
                    JMP     ERROR_1
CONT_1 :            LED_INDICATOR_OFF
MACRO               MASK_KBIRQ
                    MOV     DI, BASE_ADD_KBB
                    MOV     DX, CONTROL_PORT_PIC    ; (OCW_1)
                    MOV     AL, FFH
                    OUT     DX, AL
                    ENDM
                    BLANK_ALL
                    MOV     AL, (BP+DI)
                    MOV     (BP+DI), 00H
                    CMP     AL, 'DAT'
                    JZ      NXT_17
                    JMP     LEVEL_3
NXT_17 :            INR     DI
                    CMP     (BP+DI), 'SEL'
                    JZ      NXT_18
                    JMP     ERROR_1
NXT_18 :            INR     DI
                    MOV     DL, (BP+DI)
                    CMP     DL, 04H
                    JBE     NXT_19
                    JMP     ERROR_1
NXT_19 :            CMP     DL, 00H
                    JNZ     NXT_20
                    JMP     LEVEL_3
NXT_20 :            INR     DI
                    CMP     (BP+DI), 'ENTER'
                    JZ      NXT_21
                    JMP     ERROR_1
NXT_21 :            MOV     AL, PRESENT_PI
                    CMP     DL, PRESENT_DI
                    JZ      NXT_22
                    CMP     MARK_CURRENT_OPEN, FFH
                    JNZ     NXT_22
                    JMP     ERROR_1
NXT_22 :            MOV     SI, BASE_ADD_PIT
```

```
                    MOV    CH, 00H
                    MOV    CL, AL
                    SHL    CL                          , 4 TIMES
                    ADD    SI, CX
                    ADD    SI, 0002H
                    CMP    (SI), 01H
                    JZ     NXT_23
                    JMP    ERROR_1
NXT_23 :            INR    SI
                    MOV    (SI), DL
                    MOV    PRESENT_DI, DL
                    MOV    SI, ICP_LOC_SI
                    MOV    CX, ICP_LOC_DS
                    MOV    DS, CX
                    ADD    SI, 0004H
                    MOV    (SI), DL
                    MOV    CX, 0000H
                    MOV    DS, CX
                    LED_INDICATOR_ON
                    UNMASK_KBIRQ
                    HLT
                    JMP    LEVEL_2
CHK-DATA :          CMP    AL, 'DAT'
                    JZ     NXT_24
                    JMP    ERROR_1
NXT_24 :            INR    DI
                    CMP    (BP+DI), 'SEL'
                    JE     NXT_25
                    JMP    ERROR_1
NXT_25 :            INR    DI
                    MOV    DL, (BP+DI)
                    CMP    DL, 04H
                    JBE    NXT_26
                    JMP    ERROR_1
NXT_26 :            CMP    DL, 00H
                    JNZ    NXT_27
                    JMP    LEVEL_3
NXT_27 :            INR    DI
                    CMP    (BP+DI), 'ENTER'
                    JZ     NXT_28
                    JMP    ERROR_1
NXT_28 :            MOV    AL, PRESENT_PI
                    MOV    SI, BASE_ADD_PIT
                    MOV    CH, 00H
                    MOV    CL, AL
                    ADD    CL, 03H
                    SHL    CL                          , 4 TIMES
                    ADD    SI, CX
```

```
                ADD     SI, 0003H
                CMP     (SI), 00H
                JZ      NXT_29
                JMP     ERROR_1
NXT_29 :        DCR     SI
                MOV     (SI), DL
                MOV     PRESENT_DI, DL
                MOV     SI, ICP_LOC_SI
                MOV     CX, ICP_LOC_DS
                MOV     DS, CX
                ADD     SI, 0004H
                MOV     (SI), DL
                MOV     CX, 0000H
                MOV     DS, CX
                LED_INDICATOR_ON
                UNMASK_KBIRQ
                HLT
                JMP     LEVEL_2
CHK_VAC :       CMP     AL, 'VAC'
                JNZ     NXT_2A
                JMP     RSLV_VAC
NXT_2A :        CMP     AL, 'LST'
                JNZ     NXT_2B
                JMP     RSLV_LST
NXT_2B :        CMP     AL, 'DEL'
                JZ      RSLV_DEL
                JMP     ERROR_1
RSLV_DEL :      INR     DI
                CMP     (BP+DI), 'SEL'
                JZ      NXT_2C
                CMP     (BP+DI), 'ENTER'
                JZ      NXT_2C
                CMP     (BP+DI), 'ENTER'
                JZ      NXT_2D
                JMP     ERROR_1
NXT_2D :        CALL    DEL_ALL                 ; L_1, F.C. 4.1
                JMP     LEVEL_0
NXT_2C :        INR     DI
                MOV     AL, (BP+DI)
                CMP     AL, 09H
                JBE     NXT_2E
                JMP     ERROR_1
NXT_2E :        INR     DI
                CMP     (BP+DI), 'ENTER'
                JZ      NXT_2F
                JMP     ERROR_1
NXT_2F :        CALL    DEL_SEL                 ; L_1, F.C. 4.1
                JMP     LEVEL_0
```

```
RSLV_VAC :          INR   DI
                    CMP   (BP+DI), 'SEL'
                    JZ    NXT_30
                    CMP   (BP+DI), 'ENTER'
                    JZ    NXT_31
                    JMP   ERROR_1
NXT_31 :            CALL  VAC_ALL                ; L_1, F.C. 4.1
NXT_30 :            INR   DI
                    MOV   AL, (BP+DI)
                    CMP   AL, 04H
                    JBE   NXT_32
                    JMP   ERROR_1
NXT_32 :            CMP   AL, 00H
                    JZ    NXT_33
                    JMP   ERROR_1
NXT_33 :            INR   DI
                    CMP   (BP+DI), 'ENTER'
                    JZ    NXT_34
                    JMP   ERROR_1
NXT_34 :            CALL  VAC_SEL                ; L_1, F.C. 4.1
                    JMP   LEVEL_0
RSLV_LST : REFER SECTION 3.   , 4.
RSLV_RUN :          INR   DI
                    CMP   (BP+DI), 'SEL'
                    JZ    NXT_35
                    JMP   ERROR_1
NXT_35 :            INR   DI
                    MOV   AL, (BP+DI)
                    CMP   AL, 09H
                    JBE   NXT_36
                    JMP   ERROR_1
NXT_36 :            INR   DI
                    CMP   (BP+DI), 'ENTER'
                    JZ    NXT_37
                    JMP   ERROR_1
NXT_37 :            JMP   RUN_LEVEL_1
RSLV_DEF_LRN :      MOV   AL, PRESENT_PI
                    MOV   SI, BASE_ADD_PIT
                    MOV   BX, (SI)
                    MOV   CL, AL
                    ADD   CL, 03H
                    SHL   CL                     , 4 TIMES
                    ADD   SI, CX
                    ADD   SI, 0003H
                    MOV   DH, (SI)
                    CMP   DH, 03H
                    JB    NXT_33
                    JMP   ERROR_3
```

```
NXT_38 :            MOV   CL, DH
                    MOV   STATUS_PRESENT_PI, DH
                    INR   DH
                    MOV   (SI), DH
                    INR   SI
                    MOV   DL, (SI)
                    MOV   PRESENT_DI, DL
                    SHL   CL
                    SHL   CL
                    ADD   SI, CX
                    INR   SI
                    ADD   BX, 0011H
                    MOV   (SI), BX
                    ADD   SI, 0002H
                    MOV   (SI), BX
                    AND   SI, FFF0H
                    MOV   CX, (SI)
                    INR   CX
                    MOV   (SI), CX
;NUMBER OF BLOCKS USED BY PRESENT PI IS INCREMENTED AND LOADED  IN
;PIT.
                    AND   SI, FF00H
                    MOV   CX, (SI)
                    INR   CX
                    MOV   (SI), CX
;TOTAL  NUMBER OF BLOCKS USED BY ALL PI IS INCREMENTED AND  LOADED
;IN PIT.
                    ADD   SI, 0002H
                    INR   (SI)
;COUNT OF NUMBER OF PI LOADED IS INCREMENTED.
                    MOV   CH, 00H
                    MOV   CL, (SI)
                    ADD   SI, CX
                    MOV   (SI), AL
;PRESENT PI IS LOADED IN PIT.
                    CMP   BX, 00FFH
                    JBE   CHK_NXT_3
                    MOV   DX, 1000H
                    MOV   DS, BX
                    SUB   BX, 0100H
CHK_NXT_3 :         SHL   BX                          , 8 TIMES
                    MOV   SI, BX
                    MOV   CX, DS
                    MOV   BX, 0000H
                    MOV   DS, BX
                    MOV   ICP_LOC_DS, CX
                    MOV   ICP_LOC_SI, SI
                    MOV   DS, CX
```

```
ADD     SI, 0004H
MOV     (SI), AL
INR     SI
MOV     (SI), DL
MOV     DS, DX
JMP     LEVEL_3
```

TITLE : MODULE : DATA CLEAR, LEVEL_1, F.C. 4.1

PART_1 : VAC_SEL

ANATOMY :

PROGRAM CLEARS THE PARTICULAR DATA AREA INDEX. REFERENCE IS MADE TO DIT AND THE BLOCK TRANSFERS ARE SO ACHIEVED THAT THE DESIRED DATA AREA IS DELETED WHILE THE DATA STORAGE IS CONFIRMED IN THE BOTTOM OF STORAGE ORDERING. ACCORDINGLY DIT IS ALSO ALTERED.

REGISTER USAGE :

AL, AH : PRESENT DI TO BE CLEARED.
BX, DX : DETERMINATION OF THE STARTING SS, DI FOR THE STRING TRANSFER, COUNT OF STRING TRANSFER, SEGMENT MANAGEMENT.
CX : COUNT OF STRING TRANSFER.

REFERENCE MEMORY LOCATIONS :

AV_MEM_BLOCKS, COUNT_ALTER, SRC_COUNT_3, DST_COUNT_4, END_BLK_SRC, END_BLK_DST

PART_2 : VAC_ALL

ANATOMY :

PROGRAM CLEARS ALL DATA AREA EXCEPT FOR DEFAULT AREA. THE DATA AREA OPENED PREVIOUSLY ARE CLEARED AND THE CLEARED MEMORY AREA COULD BE HENCE FORTH USED FOR PROGRAM OR DATA STORAGE.

```
VAC_SEL :           MOV   SI, BASE_ADD_DIT
                    ADD   SI, 0002H
                    MOV   AH, AL
                    MOV   DL, 04H
                    CMP   AH, DL
                    JZ    DATA_SEL_4
                    MOV   CX, 0000H
                    DCR   DL
                    DCR   AH
                    JZ    DATA_SEL_0
NXT_1 :             ADD   CX, (SI)
;NUMBER OF BLOCKS FOR EACH DATA AREA INDEX.
                    ADD   SI, 0002H
                    DCR   DL
```

```
                        DCR     AH
                        JNZ     NXT_1
DATA_SEL_0 :            MOV     BX, AV_MEM_BLOCKS
                        SHL     BX
                        SUB     BX, 0003H
                        SUB     BX, CX
                        MOV     END_BLK_DST, BX
                        MOV     CX, (SI)
                        CMP     CX, 0000H
                        JZ      CONT_0
                        JMP     LEVEL_0
CONT_0 :                MOV     DST_COUNT_4, CX
                        SUB     BX, CX
                        MOV     END_BLK_SRC, BX
                        MOV     CX, 0000H
NXT_2 :                 ADD     SI, 0002H
                        ADD     CX, (SI)
                        DCR     DL
                        JNZ     NXT_2
                        MOV     SRC_COUNT_3, CX
                        CMP     CX, DST_COUNT_4
                        JB      SRC_SMTHN_DST
                        MOV     BX, END_BLK_DST
                        SUB     BX, 01FFH
                        SHL     BX                      , 3 TIMES
                        MOV     ES, BX
                        MOV     BX, END_BLK_SRC
                        SUB     BX, 01FFH
                        SHL     BX                      , 3 TIMES
                        MOV     DS, BX
                        SHL     CX                      , 7 TIMES
                        STD
                        MOV     DI, FFFFH
                        MOV     SI, FFFFH
                REP     MOVSB
                        MOV     ES, CX
                        MOV     DS, CX
                        MOV     CX, SRC_COUNT_3
                        MOV     COUNT_ALTER, CX
                        JMP     ALTER_COUNT
SRC_SMTHN_DST :         MOV     BX, DST_COUNT_4
                        MOV     COUNT_ALTER, BX
                        SUB     BX, CX
                        MOV     DST_COUNT_4, BX
                        MOV     BX, END_BLK_DST
                        SUB     BX, 01FFH
                        SHL     BX                      , 3 TIMES
                        MOV     ES, BX
```

```
                    MOV   BX, END_BLK_SRC
                    SUB   BX, 01FFH
                    SHL   BX                    , 3 TIMES
                    MOV   DS, BX
                    SHL   CX                    , 7 TIMES
                    STD
                    MOV   DI, FFFFH
                    MOV   SI, FFFFH
               REP  MOVSB
                    MOV   CX, DST_COUNT_4
                    SHL   CX                    , 7 TIMES
                    MOV   SS, ES
NXT_3 :             MOV   (BP+DI), 00H
                    DCR   DI
                    LOOP  NXT_3
                    JMP   ALTER_COUNT
DATA_SEL_4 :        DCR   DL
                    MOV   CX, 0000H
NXT_4 :             ADD   SI, 0002H
                    ADD   CX, (SI)
                    DCR   DL
                    MOV   BX, AV_MEM_BLOCKS
                    SHL   BX
                    SUB   BX, 0003H
                    MOV   END_BLK_DST, BX
                    ADD   SI, 0002H
                    MOV   CX, (SI)
                    MOV   COUNT_ALTER, CX
                    SHL   CX                    , 7 TIMES
                    SUB   BX, 01FFH
                    MOV   SI, FFFFH
                    SHL   BX                    , 7 TIMES
NXT_5 :             MOV   (BX+SI), 00H
                    DCR   SI
                    LOOP  NXT_5
ALTER_COUNT :       MOV   SI, BASE_ADD_DIT
                    MOV   CX, 0000H
                    MOV   DS, CX
                    MOV   ES, CX
                    MOV   SS, CX
                    MOV   CX, COUNT_ALTER
                    SUB   (SI), CX
NXT_6 :             ADD   SI, 0002H
                    DCR   AL
                    JNZ   NXT_6
                    MOV   (SI), 00H
                    JMP   LEVEL_0
```

```
PART_2 :


VAC_ALL :           MOV   SI, BASE_ADD_DIT
                    MOV   CX, (SI)
                    MOV   BX, 1000H
                    MOV   DS, BX
                    MOV   SI, FEFFH
                    MOV   DX, 0000H
                    CMP   CX, 01FEH
                    JB    CHK_NXT_1
                    SUB   CX, 01FEH
                    MOV   DX, CX
                    MOV   CX, 01FEH
CHK_NXT_1 :         SHL   CX                    , 7 TIMES
NXT_1 :             MOV   (SI), 00H
                    DCR   SI
                    LOOP  NXT_1
                    MOV   DS, CX
                    CMP   DX, 0000H
                    JZ    RSLV_TAB
                    MOV   SI, FFFFH
                    MOV   CX, DX
                    SHL   CX                    , 7 TIMES
NXT_2 :             MOV   (SI), 00H
                    DCR   SI
                    LOOP  NXT_2
RSLV_TAB :          MOV   SI, BASE_ADD_DIT
                    MOV   CX, 000FH
NXT_3 :             MOV   (SI), 00H
                    INR   SI
                    LOOP  NXT_3
                    JMP   LEVEL_0
```

TITLE : MODULE : PROGRAM CLEAR, LEVEL_1 F.C. 4.1

PART_1 : DEL_SEL

ANATOMY :

THE PIT IS REFERRED TO POINT TO THE STARTING AND ENDING BLOCKS OF THE SPECIFIED PROGRAM INDEX WITH MAXIMUM STATUS COUNT. FURTHER STARTING AND ENDING BLOCKS OF THE PI AREAS BELOW GIVEN SPECIFIED ABOVE INDEX IS ALSO DETERMINED. THE STRING TRANSFER AND RECORRECTION IN PIT IS EXECUTED. THE PROCESS CONTINUES (LOOPS BACK) TILL ALL THE PROGRAM AREA OF SPECIFIED COUNT ARE DELETED.

REGISTER USAGE :

AL : PRESENT PROGRAM INDEX TO BE CLEARED.
AH : PREVIOUS PROGRAM INDEX, STATUS OF PREVIOUS PI.
BX, DX : DETERMINATION OF THE STARTING SS, DI FOR THE STRING TRANSFER, COUNT OF STRING TRANSFER, SEGMENT MANAGEMENT.
CX : COUNT OF STRING TRANSFER.

REFERENCE MEMORY LOCATIONS :

ST_BLK_SRC, ST_BLK_DST, SRC_COUNT_1, DST_COUNT_2, END_BLK_SRC, END_BLK_DST.

PART_2 : DEL_ALL

ANATOMY :

CLEARS ALL PROGRAM AREA.

```
DEL_SEL :       MOV   SI, BASE_ADD_PIT
                MOV   CX, (SI)
                ADD   SI, 0011H
                MOV   END_BLK_SRC, CX
                INR   SI
                MOV   BH, 00H
                MOV   BL, (SI)
                MOV   CH, 00H
                MOV   CL, AL
                ADD   CL, 00H
                SHL   CL                    , 4 TIMES
                ADD   SI, CX
```

```
                    ADD     SI, 0002H
                    MOV     DL, 00H
                    MOV     DH, (SI)
START_AGAIN :       MOV     SI, BASE_ADD_PIT
                    ADD     SI, BX
                    ADD     SI, 0002H
                    MOV     AH, (SI)
                    CMP     AH, AL
                    JZ      FOUND_PREVIOUS
                    MOV     CL, AH
                    ADD     CL, 00H
                    SHL     CL                      , 4 TIMES
                    MOV     SI, BASE_ADD_PIT
                    ADD     SI, CX
                    ADD     SI, 0002H
                    MOV     AH, (SI)
                    MOV     CL, AH
                    SUB     CL, DL
                    SHL     CL                      , 2 TIMES
                    ADD     SI, CX
                    MOV     CX, (SI)
                    MOV     ST_BLK_SRC, CX
                    DCR     BL
                    JMP     START_AGAIN
FOUND_PREVIOUS :    MOV     SI, BASE_ADD_PIT
                    MOV     CL, AL
                    ADD     CL, 03H
                    SHL     CL                      , 4 TIMES
                    ADD     SI, CX
                    MOV     CL, DH
                    SHL     CL                      , 2 TIMES
                    ADD     SI, CX
                    MOV     CX, (SI)
                    MOV     ST_BLK_DST, CX
                    ADD     SI, 0002H
                    MOV     CX, (SI)
                    MOV     END_BLK_DST, CX
                    INR     CX
                    MOV     SRC_COUNT_1, CX
                    MOV     CX, ST_BLK_DST
                    CMP     CX, ST_BLK_SRC
                    JZ      CONT_1
                    JMP     LOAD_STRING
CONT_1 :            MOV     CX, END_BLK_SRC
                    CMP     CX, END_BLK_DST
                    JZ      CONT_2
                    JMP     ERROR_3
CONT_2 :            CMP     CX, 00FFH
```

```
                        JA      BLK_GTHAN_FF
                        SUB     CX, ST_BLK_DST
                        SHL     CX                      , 8 TIMES
                        MOV     SI, CX
NXT_1 :                 MOV     (SI), 00H
                        INR     SI
                        LOOP    NXT_1
;PROGRAM AREA LOADED WITH 00.
                        JMP     RSLV_TAB
BLK_GTHAN_FF :          MOV     CX, SRC_COUNT_1
                        CMP     CX, 0100H
                        JA      BLK_IN_NXT_SEG
                        MOV     CX, ST_BLK_DST
                        SHL     CX                      , 4 TIMES
                        MOV     DS, ES
                        MOV     CX, SRC_COUNT_1
                        SHL     CX                      , 8 TIMES
                        MOV     SI, 0000H
NXT_2 :                 MOV     (SI), 00H
                        INR     SI
                        LOOP    NXT_2
                        JMP     RSLV_TAB
BLK_IN_NXT_SEG :        SUB     CX, 0100H
                        ADD     DST_COUNT_2, CX
                        MOV     CX, ST_BLK_DST
                        SHL     CX                      , 4 TIMES
                        MOV     DS, CX
                        MOV     SI, 0000H
                        MOV     CX, FFFFH
NXT_3 :                 MOV     (SI), 00H
                        INR     SI
                        LOOP    NXT_3
                        MOV     CX, DS
                        ADD     CX, 1000H
                        MOV     DS, CX
                        MOV     CX, DST_COUNT_2
                        SHL     CX                      , 8 TIMES
                        MOV     SI, 0000H
NXT_4 :                 MOV     (SI), 00H
                        INR     SI
                        LOOP    NXT_4
RSLV_TAB :              MOV     SI, BASE_ADD_PIT
                        MOV     CL, AL
                        ADD     CL, 03H
                        SHL     CL                      , 4 TIMES
                        ADD     SI, CX
                        MOV     CX, (SI)
                        SUB     CX, SRC_COUNT_1
```

```
                    MOV   (SI), CX
                    ADD   SI, 0002H
                    MOV   CL, DH
                    DCR   CL
                    MOV   (SI), CL
                    INR   CL
                    SHL   CL                        , 2 TIMES
                    ADD   SI, CX
                    SUB   SI, 0002H
                    MOV   CX, 0004H
NXT_5 :             MOV   (SI), 00H
                    INR   SI
                    LOOP  NXT_5
                    MOV   SI, BASE_ADD_PIT
                    MOV   CX, (SI)
                    SUB   CX, SRC_COUNT_1
                    MOV   (SI), CX
                    ADD   SI, 0002H
                    MOV   CL, BL
                    DCR   CL
                    MOV   (SI), CL
                    INR   CL
                    ADD   SI, CX
                    MOV   (SI), 00H
                    INR   DL
                    DCR   DL
                    JZ    CONT_3
                    JMP   START_AGAIN
CONT_3 :            JMP   LEVEL_0
LOAD_STRING :       MOV   CX, END_BLK_DST
                    INR   CX
                    MOV   DST_COUNT_2, CX
                    MOV   CX, END_BLK_SRC
                    SUB   CX, ST_BLK_SRC
                    INR   CX
                    MOV   SRC_COUNT_1, CX
                    MOV   CX, END_BLK_SRC
                    CMP   CX, 00FFH
                    JA    BLK_ABOV_FF
                    PUSH  DX
                    MOV   DX, 0000H
                    MOV   CX, DST_COUNT_2
                    CMP   CX, SRC_COUNT_1
                    JBE   CONT_4
                    SUB   CX, SRC_CONT_1
                    MOV   DX, CX
CONT_4 :            MOV   CX, SRC_COUNT_1
                    MOV   DI, ST_BLK_DST
```

```
                        SHL     DI                      , 8 TIMES
                        MOV     SI, ST_BLK_SRC
                        SHL     SI                      , 8 TIMES
                        SHL     CX                      , 8 TIMES
                    REP MOVSB
                        CMP     DX, 0000H
                        JNZ     CONT_5
                        JMP     RSLV_TAB2
CONT_5 :                INR     DI
NXT_6 :                 MOV     (DI), 00H
                        INR     DI
                        LOOP    NXT_6
                        JMP     RSLV_TAB2
BLK_ABOV_FF :           MOV     CX, DST_COUNT_2
                        CMP     CX, 0100H
                        JA      BLK_IN_NXT_SEG2
                        CMP     CX, SRC_COUNT_1
                        JA      DST_GTHAN_SRC
                        MOV     CX, ST_BLK_DST
                        SHL     CX                      , 4 TIMES
                        MOV     ES, CX
                        MOV     DI, 0000H
                        MOV     SI, 0000H
                        MOV     CX, SRC_COUNT_1
                        PUSH    DX
                        MOV     DX, ST_BLK_SRC
                        SHL     DX                      , 4 TIMES
                        MOV     DS, DX
                        POP     DX
                        CMP     CX, 0100H
                        JBE     COUNT_IN_FST_SEG
                        MOV     CX, FFFFH
                    REP MOVSB
                        MOV     CX, DX
                        ADD     CX, 0000H
                        MOV     DS, CX
                        PUSH    DS
                        MOV     SI, 0000H
                        MOV     CX, SRC_COUNT_1
                        SUB     CX, 0100H
                        SHL     CX                      , 8 TIMES
                        POP     DS
                    REP MOVSB
                        JMP     RSLV_TAB2
COUNT_IN_FST_SEG :      SHL     CX                      , 8 TIMES
                    REP MOVSB
                        MOV     DS, CX
                        MOV     ES, CX
```

```
                        JMP     RSLV_TAB2
DST_GTHAN_SRC :         MOV     SI, 0000H
                        MOV     CX, ST_BLK_DST
                        SHL     CX                      , 4 TIMES
                        MOV     ES, CX
                        MOV     DI, 0000H
                        MOV     CX, SRC_COUNT_1
                        PUSH    DX
                        MOV     DX, ST_BLK_SRC
                        SHL     DX                      , 4 TIMES
                        MOV     DS, DX
                        POP     DX
                        SHL     CX                      , 8 TIMES
                REP     MOVSB
                        PUSH    DS
                        MOV     DX, CX
                        MOV     CX, DST_COUNT_2
                        SUB     CX, SRC_COUNT_1
                        SHL     CX                      , 8 TIMES
                        POP     DS
                        INR     DI
NXT_7 :                 MOV     (DI), 00H
                        INR     DI
                        LOOP    NXT_7
                        JMP     RSLV_TAB2
BLK_IN_NXT_SEG2 :       MOV     DI, 0000H
                        MOV     CX, ST_BLK_DST
                        SHL     CX                      , 4 TIMES
                        MOV     ES, CX
                        MOV     CX, SRC_COUNT_1
                        SHL     CX                      , 8 TIMES
                        MOV     SI, ST_BLK_SRC
                        SUB     SI, 0100H
                        SHL     SI                      , 8 TIMES
                        PUSH    DX
                        MOV     DX, 1000H
                        MOV     DS, DX
                REP     MOVSB
                        MOV     CX, FFFFH
                        SUB     CX, DI
                        INR     CX
                        MOV     SS, ES
                        MOV     BP, 0000H
NXT_8 :                 MOV     (BP+DI), 00H
                        INR     DI
                        LOOP    NXT_8
                        MOV     DX, 0000H
                        MOV     DS, DX
```

```
                    MOV   ES, DX
                    MOV   CX, DST_COUNT_2
                    SUB   CX, 0100H
                    MOV   DX, SS
                    ADD   DX, 1000H
                    MOV   SS, DX
                    SHL   CX                    , 8 TIMES
                    MOV   DI, 0000H
NXT_9 :             MOV   (BP+DI), 00H
                    INR   DI
                    LOOP  NXT_9
                    MOV   SS, CX
                    POP   DX
RSLV_TAB2 :         MOV   CX, SRC_COUNT_1
                    CMP   CX, DST_COUNT_2
                    JNS   CONT_6
                    MOV   CX, DST_COUNT_2
CONT_6 :            MOV   SRC_COUNT_1
                    MOV   SI, BASE_ADD_PIT
                    SUB   (SI), CX
                    ADD   SI, 0002H
                    MOV   BH, 00H
REPEAT :            MOV   BL, (SI)
                    MOV   CX, 0000H
                    MOV   CL, BL
                    ADD   SI, CX
                    MOV   AH, (SI)
                    MOV   (SI), BH
                    CMP   AL, AH
                    JZ    FOUND_PI_EQUAL
                    MOV   BH, AH
                    MOV   SI, BASE_ADD_PIT
                    MOV   CL, AH
                    ADD   CL, 03H
                    SHL   CL                    , 4 TIMES
                    ADD   SI, CX
                    ADD   SI, 0002H
                    MOV   CL, (SI)
                    SUB   CL, DL
                    SHL   CL                    , 2 TIMES
                    ADD   SI, CX
                    MOV   CX, (SI)
                    SUB   CX, DST_COUNT_2
                    MOV   (SI), CX
                    SUB   SI, 0002H
                    MOV   CX, (SI)
                    SUB   CX, DST_COUNT_2
                    MOV   (SI), CX
```

```
                        MOV     SI, BASE_ADD_PIT
                        ADD     SI, 0002H
                        DCR     (SI)
                        JMP     REPEAT
FOUND_PI_EQUAL :        MOV     SI, BASE_ADD_PIT
                        MOV     CL, AL
                        ADD     CL, 03H
                        SHL     CL                      , 4 TIMES
                        ADD     SI, CX
                        MOV     CX, (SI)
                        SUB     CX, DST_COUNT_2
                        MOV     (SI), CX
                        ADD     SI, 0002H
                        DCR     (SI)
                        MOV     CL, DH
                        SHL     CL                      , 2 TIMES
                        ADD     SI, CX
                        MOV     (SI), 00H
                        SUB     SI, 0002H
                        INR     DL
                        DCR     DH
                        JZ      CONT_7
                        JMP     START_AGAIN
CONT_7 :                AND     SI, FFF0H
                        ADD     SI, 0003H
                        MOV     (SI), 00H
                        JMP     LEVEL_0

PART_2 :

DEL_ALL :               MOV     DX, 0000H
                        MOV     SI, BASE_ADD_PIT
                        MOV     CX, (SI)
                        ADD     CX, 0010H
                        MOV     END_BLK_SRC, CX
                        CMP     CX, 00FFH
                        JBE     BLK_IN_SEG_0
                        SUB     CX, 00FFH
                        MOV     DX, CX
                        MOV     CX, 00FFH
BLK_IN_SEG_0 :          SUB     CX, 0010H
                        SHL     CX                      , 8 TIMES
                        MOV     SI, 1100H
NXT_1 :                 MOV     (SI), 00H
                        INR     SI
                        LOOP    NXT_1
```

```
                    CMP     DX, 0000H
                    JZ      RSLV_TAB
                    MOV     CX, DX
                    MOV     DX, 1000H
                    MOV     DS, DX
                    MOV     SI, 0000H
                    SHL     CX                      , 8 TIMES
NXT_2 :             MOV     (SI), 00H
                    INR     SI
                    LOOP    NXT_2
RSLV_TAB :          MOV     DS, CX
                    MOV     SI, BASE_ADD_PIT
                    MOV     CX, 00FFH
NXT_3 :             MOV     (SI), 00H
                    INR     SI
                    LOOP    NXT_3
                    JMP     LEVEL_0
```

TITLE : MODULE : RSLV_LRN, LEVEL_1, F.C. 4.1

ANATOMY :

REFERS TO PIT AND DETERMINES STARTING BLOCK TO BE OFFERED FOR SELECTED PI. THE FIRST INTERPRETER CODE POINTER FIELD IS LOADED WITH THE SUPPLIED PI AND DI USING PIT ITSELF. THE INTERPRETER CODE POINTERS FIELD ARE SAVED TO BE REFERRED LATER.

REGISTER USAGE :

AL : PRESENT PI FROM KB ENTRY.
AH : PREVIOUS PI FROM PIT.
CX : USED FOR POINTING LOCATIONS FROM PIT, MANAGING BLOCK INDICES OF PI.
BX, DX : MANAGING BLOCK INDICES OF PI.
SI : BASE_ADD_PIT.

REFERENCE MEMORY LOCATIONS :

PRESENT_PI, PRESENT_DI, ICP_LOC_DS, ICP_LOC_SI, PRESENT_STATUS_PI, MARK_CURRENT_OPEN.

```
RSLV_LRN :          MOV   PRESENT_PI, AL
                    MOV   SI, BASE_ADD_PIT
                    ADD   SI, 0002H
                    MOV   CH, 00H
                    MOV   CL, (SI)
;TOTAL NUMBER OF PROGRAM INDEX.
                    ADD   SI, CX
;POINT LAST WORKING PROGRAM INDEX.
                    MOV   AH, (SI)
                    CMP   AL, AH
                    JNE   CHK_NXT_1
                    MOV   SI, BASE_ADD_PIT
                    MOV   CL, AH
                    ADD   CL, 03H
                    SHL   CX                    , 4 TIMES
                    ADD   SI, CX
                    ADD   SI, 0003H
;POINT STATUS OF PRESENT PROGRAM INDEX WHICH IS EQUAL TO PREVIOUS
;PROGRAM INDEX.
                    MOV   CL, (SI)
                    CMP   CL, 00H
                    JNZ   NXT_0
                    INR   CL
                    JMP   NXT_1
```

```
NXT_0 :                 CMP     CL, 01H
                        JNZ     NXT_1
                        MOV     MARK_CURRENT_OPEN, FFH
NXT_1 :                 CMP     CL, 03H
                        JB      NXT_2
                        JMP     ERROR_3
NXT_2 :                 DCR     SI
                        MOV     DL, (SI)
                        MOV     PRESENT_DI, DL
                        SHL     CL                          , 2 TIMES
                        ADD     SI, CX
                        MOV     CX, (SI)
;DETERMINE ENDING BLOCK OF PRESENT PROGRAM INDEX.
                        CMP     CX, 00FFH
                        JBE     CHK_NXT_2
                        MOV     DX, 1000H
                        MOV     DS, DX
                        SUB     CX, 0100H
CHK_NXT_2 :             MOV     BX, 0000H
                        SHL     CX                          , 8 TIMES
                        MOV     SI, CX
                        INR     SI
;POINT TO FIRST LINE NUMBER IN THE ENDING BLOCK.
CHK_CMP :               MOV     CL, (SI+BX)
                        CMP     CL, 00H
                        JE      LOAD_ICP
                        ADD     BX, 0008H
                        CMP     BX, 00F8H
                        JB      NXT_3
                        JMP     ERROR_3
;RETRIAL FROM LRN.
NXT_3 :                 JMP     CHK_CMP
LOAD_ICP :              ADD     SI, BX
                        MOV     CX, DS
                        DCR     SI
;SI POINTS TO ATTACH BYTE.
                        MOV     BX, 0000H
                        MOV     DS, BX
                        MOV     ICP_LOC_DS, CX
                        MOV     ICP_LOC_SI, SI
                        RET
CHK_NXT_1 :             MOV     SI, BASE_ADD_PIT
                        MOV     BX, (SI)
                        MOV     CL, AL
                        ADD     CL, 03H
                        SHL     CL                          , 4 TIMES
                        ADD     SI, CX
                        ADD     SI, 0003H
```

```
                        MOV    DH, (SI)
;STATUS  COUNT  OF PRESENT PROGRAM INDEX IS NOT  THE  MAXIMUM  ONE
;ALLOWED.
                        CMP    DH, 03H
                        JB     NXT_4
                        JMP    ERROR_3
NXT_4 :                 MOV    CL, DH
                        INR    DH
                        MOV    STATUS_PRESENT_PI, DH
                        MOV    (SI), DH
                        DCR    SI
                        MOV    DL, (SI)
                        MOV    PRESENT_DI, DL
                        SHL    CL
                        SHL    CL
                        ADD    SI, CX
                        ADD    BX, 0011H
;ADD STARTING BLOCK OFFSET TO TOTAL NUMBER OF BLOCKS.
                        MOV    (SI), BX
                        ADD    SI, 0002H
                        MOV    (SI), BX
                        ADD    SI, FFF0H
                        MOV    CX, (SI)
                        INR    CX
                        MOV    (SI), CX
;INCREMENT AREA USED BY PRESENT PROGRAM INDEX.
                        AND    SI, FF00H
                        MOV    CX, (SI)
                        INR    CX
                        MOV    (SI), CX
;INCREMENT TOTAL AREA USED.
                        ADD    SI, 0002H
                        INR    (SI)
;INCREMENT COUNT OF PROGRAM INDEX TABLE (PIT).
                        MOV    CL, (SI)
                        ADD    SI, CX
                        MOV    (SI), AL
;LOAD PRESENT PROGRAM INDEX IN PIT.
                        CMP    BX, 00FFH
                        JBE    CHK_NXT_3
                        MOV    DX, 1000H
                        MOV    DS, DX
                        SUB    BX, 0100H
CHK_NXT_3 :             SHL    BX                        , 8 TIMES
                        MOV    SI, BX
                        MOV    CX, DS
                        MOV    BX, 0000H
                        MOV    DS, BX
```


```
MOV    ICP_LOC_DS, CX
MOV    ICP_LOC_SI, SI
MOV    DS, CX
ADD    SI, 0004H
MOV    (SI), AL
INR    SI
MOV    (SI), DL
MOV    DS, BX
RET
```

TITLE : MODULE : RSLV_DIT, LEVEL_2, F.C. 4.1

ANATOMY :

PART_1 : THE KB ENTRIES CORRESPONDING TO EXTENDED ARRAY, NORMAL ARRAY, 24R ARRAY, 16I ARRAY ARE LOADED IN MEMORY. SUBSEQUENTLY THE ASCENDING ORDERS OF ENTRIES IS CONFIRMED.

REGISTER USAGE :

AL & AH : RESOLVING ASCENDING ORDER OF ARRAY ENTRIES.
BL : TOTAL COUNT OF ARRAYS, TOTAL COUNT OF REAL ARRAYS.
CL, CH : TOTAL COUNT OF INDIVIDUAL ARRAYS.
SI : OFFSETS OF (ER) EXTENDED REAL ARRAY, (NR) NORMAL REAL ARRAY, (EI) EXTENDED INTEGER ARRAY, (NI) NORMAL INTEGER ARRAY, (24R) 24_REAL ARRAY, (16I) 16_INTEGER ARRAY.
DI : BASE_ADD_KBB, BASE_ADD_DIT.

REFERENCE MEMORY LOCATIONS :

OFFSET_EI, OFFSET_NI, OFFSET_24R, OFFSET_16I, OFFSET_REAL, OFFSET_ER, OFFSET_NR.

PART_2 : SORTING AND COMBINATION OF THESE ARRAYS IS PERFORMED AND DIT IS PREPARED. SEE REFERENCE 4.

REGISTER USAGE :

SI : BASE_ADD_DIT.
CX : MANAGES DIT POINTER.
AX : NUMBER OF BLOCKS USED BY DATA AREA.
DX : ESTIMATE OF STORAGE LENGTH OF DATA TYPE.

REFERENCE MEMORY LOCATIONS :

PRESENT_DATA_SEG.

PART_3 : THE REQUIRED SPACE FOR DATA AREA IS CREATED.

REGISTER USAGE :

AL, AH : PARAMETERS SUCH AS DATA AREA INDEX. (DI).
BX, CX, DX : MANAGE COUNT AND POINTERS FOR THE REQUIRED STRING TRANSFER (SOURCE, DESTINATION, COUNT).

REFERENCE MEMORY LOCATIONS :

AV_MEM_BLOCKS, END_BLK_DST, ST_BLK_DI4_PREV,
END_BLK_SRC, ST_BLK_DI4_NEW

```
RSLV_DIT :          MOV   BX, (BP+DI)          ; TOTAL COUNT
                    DCR   BL
                    MOV   DI, 0000H
                    MOV   AH, '.'
                    MOV   AL, (BP+DI)
                    MOV   (BP+DI), 00H
                    CMP   AL, 'EXT_ARRY'
                    JZ    AGAIN
                    JMP   CHK_NORM
AGAIN :             INR   DI
                    MOV   AL, (BP+DI)
                    CMP   AL, AH
                    JA    NXT_1
                    JMP   ERROR_1
NXT_1 :             CMP   AL, 'Z'
                    JBE   NXT_2
                    JMP   ERROR_1
NXT_2 :             MOV   AH, AL
                    INR   DI
                    MOV   AL, (BP+DI)
                    CMP   AL, 'ENTER'
                    JZ    RSLV_ENTER
                    CMP   AL, ','
                    JZ    NXT_3
                    JMP   ERROR_1
NXT_3 :             SUB   BL, 02H
                    JNZ   AGAIN
RSLV_ENTER :        SUB   DI, BASE_ADD_KBB
                    MOV   BL, (BP+DI)
                    DCR   BL
                    SHR   BL                   ; TOTAL COUNT
                    MOV   DI, BASE_ADD_KBB
                    INR   DI
                    MOV   CL, 00H
                    MOV   CH, 00H
                    MOV   BP, BASE_ADD_24R
                    MOV   SI, OFFSET_ER
                    MOV   BP, CX
AGAIN_1 :           INR   SI
                    MOV   AL, (BP+DI)
                    CMP   AL, 'Q'
                    JAE   LOAD_EI
                    INR   CL
```

```
                        MOV     (SI), AL
                        ADD     DI, 0002H
                        DCR     BL
                        JMP     AGAIN_1
                        AND     SI, FFF0H
;GET ORIGINAL VALUE OF SI.
                        MOV     (SI), CL
                        CMP     BL, CL
                        JNZ     LOAD_EI
                        ADD     SI, 0010H
;THE  ARRAYS  ARE  PLACED ON AN  CONSECUTIVE  SIXTEEN  BYTE  PAGES
;INITIALLY.
                        MOV     (SI), 00H
LOAD_EI :               INR     SI
AGAIN_2 :               MOV     AL, (BP+DI)
                        MOV     (SI), AL
                        INR     CH
                        INR     SI
                        INR     DI, 0002H
                        DCR     BL
                        JNZ     AGAIN_2
                        AND     SI, FFF0H
                        MOV     (SI), CH
                        LED_INDICATOR_ON
                        UNMASK_KBIRQ
                        HLT
                        CMP     AL, 'ENTER'
                        JZ      NXT_4
                        JMP     ERROR_1
NXT_4 :                 LED_INDICATOR_OFF
                        BLANK_ALL
                        MASK_KBIRQ
                        MOV     CX, DI
                        DCR     CL
                        MOV     DI, 0000H
                        MOV     AL, (BP+DI)
                        MOV     (BP+DI), 00H
CHK_NORM :              CMP     AL, 'NORM_ARRY'
                        JNZ     CHK_24R
                        MOV     AH, '.'
AGAIN_3 :               INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, AH
                        JA      NXT_5
                        JMP     ERROR_1
NXT_5 :                 CMP     AL, 'Z'
                        JBE     NXT_6
                        JMP     ERROR_1
```

```
                        MOV     AH, AL
                        INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, 'ENTER'
                        JZ      RSLV_ENTERN
                        CMP     AL, ','
                        JZ      NXT_6
                        JMP     ERROR_1
                        SUB     CL, 02H
                        JNZ     AGAIN_3
RSLV_ENTERN :           MOV     CH, 00H
                        MOV     CL, 00H
                        MOV     BX, DI
                        SUB     BX, BASE_ADD_KBB
                        DCR     BX
                        SHR     BL
                        MOV     DI, BASE_ADD_KBB
                        MOV     BP, BASE_ADD_NORM
                        MOV     SI, OFFSET_NR
                        MOV     BP, CX
                        INR     DI
AGAIN_4 :               MOV     AL, (BP+DI)
                        CMP     AL, 'Q'
                        JAE     NXT_7
                        INR     SI
                        MOV     (SI), AL
                        INR     CL
                        ADD     DI, 0002H
                        DCR     BL
                        JMP     AGAIN_4
NXT_7 :                 AND     SI, FFF0H
                        MOV     (SI), CL
                        ADD     SI, 0010H
                        CMP     BL, CL
                        JNZ     LOAD_NI
                        MOV     (SI), 00H
LOAD_NI :               INR     SI
AGAIN_5 :               MOV     AL, (BP+DI)
                        MOV     (SI), AL
                        INR     CH
                        INR     SI
                        ADD     DI, 0002H
                        DCR     BL
                        JNZ     AGAIN_5
                        MOV     (SI), CH
                        LED_INDICATOR_ON
                        UNMASK_KBIRQ
                        HLT
```

```
                        CMP     AL, 'ENTER'
                        JZ      NXT_8
                        JMP     ERROR_1
                        LED_INDICATOR_OFF
                        MASK_KBIRQ
                        BLANK_ALL
NXT_8 :                 MOV     CX, DI
                        SUB     CX, BASE_ADD_KBB
                        DCR     CX
                        MOV     AL, (BP+DI)
                        MOV     (BP+DI), 00H
CHK_24R :               CMP     AL, '24R'
                        JNZ     CHK_16I
                        MOV     AH, '.'
                        ADD     SI, 0010H
                        MOV     CL, 00H
AGAIN_6 :               INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, AH
                        JBE     ERROR_1
                        CMP     AL, 'Q'
                        JB      NXT_9
                        JMP     ERROR_1
NXT_9 :                 MOV     AH, AL
                        INR     SI
                        INR     CL
;COUNT OF 24 ARRAY.
                        MOV     (SI), AL
                        INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, 'ENTER'
                        JZ      RSLV_24R
                        CMP     AL, ','
                        JZ      NXT_0A
                        JMP     ERROR_1
NXT_0A :                SUB     BL, 00H
                        JNZ     AGAIN_6
RSLV_24R :              AND     SI, FFF0H
                        MOV     (SI), CL
                        LED_INDICATOR_ON
                        UNMASK_KBIRQ
                        HLT
                        CMP     AL, 'ENTER'
                        JZ      NXT_0B
                        JMP     ERROR_1
                        LED_INDICATOR_OFF
                        MASK_KBIRQ
                        BLANK_ALL
```

```
NXT_0B :            MOV   CX, DI
                    SUB   BX, BASE_ADD_KBB
                    DCR   CX
                    MOV   AL, (BP+DI)
                    MOV   (BP+DI), 00H
CHK_16I :           CMP   AL, '16I'
                    JZ    NXT_0C
                    JMP   ERROR_1
NXT_0C :            MOV   AH, '.'
                    ADD   SI, 0010H
                    MOV   CH, 00H
AGAIN_7 :           INR   DI
                    MOV   AL, (BP+DI)
                    CMP   AL, AH
                    JA    NXT_0D
                    JMP   ERROR_1
NXT_0D :            CMP   AL, 'Q'
                    JAE   NXT_0E
                    JMP   ERROR_1
NXT_0E :            MOV   AH, AL
                    INR   SI
                    INR   CH
                    MOV   (SI), AL
                    INR   DI
                    MOV   AL, (BP+DI)
                    CMP   AL, 'ENTER'
                    JZ    RSLV_ENTER_16I
                    CMP   AL, ','
                    JZ    NXT_0F
                    JMP   ERROR_1
NXT_0F :            SUB   BL, 02H
                    JNZ   AGAIN_7
RSLV_ENTER_16I :    AND   SI, FFF0H
                    MOV   (SI), CH
                    CMP   CH, 00H
                    JZ    NXT_10
                    JMP   CHK_NI
NXT_10 :            SUB   SI, 0020H
                    MOV   CL, (SI)
                    CMP   CL, 00H
                    JZ    NXT_12
                    MOV   DI, BASE_ADD_DIT
                    ADD   DI, 0003H
                    INR   SI
                    MOV   AL, (SI)
                    DCR   AL
                    MOV   (BP+DI), AL
;DEF MARK LOADED AS FIRST N8I ELEMENT-1.
```

```
                        INR    DI
                        MOV    (BP+DI), 00H
                        INR    AL
                        INR    DI
                        MOV    (BP+DI), AL
                        INR    DI
                        MOV    (BP+DI), CL
                        INR    DI
                        MOV    (BP+DI), AL
                        DCR    CL
AGAIN_8 :               INR    SI
                        INR    DI
                        MOV    AL, (SI)
                        MOV    (BP+DI), AL
                        LOOP   AGAIN_8
                        INR    DI
                        MOV    (BP+DI), 00H
                        SUB    SI, 0020H
                        MOV    CL, (SI)
                        CMP    CL, 00H
                        JNZ    NXT_13
                        INR    DI
;PROCEEDING WHERE EI ELEMENT ARE NOT PRESENT.
                        MOV    (BP+DI), 2BH
;'Z'+1, MARK OF E_8I ELEMENT.
                        INR    DI
                        MOV    (BP+DI), 00H
                        INR    DI
                        MOV    (BP+DI), 00H
                        JMP    RSLV_REAL
NXT_12 :                SUB    SI, 0020H
                        MOV    CL, (SI)
                        CMP    CL, 00H
                        JNZ    NXT_14
                        MOV    DI, BASE_ADD_DIT
                        ADD    DI, 0003H
                        MOV    (BP+DI), 2BH
                        INR    DI
                        MOV    (BP+DI), 00H
                        INR    DI
                        MOV    (BP+DI), 'P'
                        INR    DI
                        MOV    (BP+DI), 00H
                        INR    DI
                        MOV    (BP+DI), 00H
                        INR    DI
                        MOV    (BP+DI), 'P'
                        INR    DI
```

```
                        MOV     (BP+DI), 00H
                        INR     DI
                        MOV     (BP+DI), 00H
                        JMP     RSLV_REAL
NXT_14 :                MOV     DI, BASE_ADD_DIT
                        ADD     DI, 0003H
                        INR     SI
                        MOV     AL, (SI)
                        DCR     AL
                        MOV     (BP+DI), AL
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        MOV     (BP+DI), 'P'
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        INR     AL
                        MOV     (BP+DI), AL
                        INR     DI
                        MOV     (BP+DI), CL
NXT_13 :                INR     SI
                        MOV     AL, (SI)
                        INR     DI
                        MOV     (BP+DI), AL
                        LOOP    NXT_13
                        INR     DI
                        MOV     (BP+DI), 00H
                        JMP     RSLV_REAL
CHK_NI :                SUB     SI, 0020H
                        MOV     CL, (SI)
                        CMP     CL, 00H
                        JZ      CL, 00H
                        JZ      NXT_11
                        JMP     CHK_EI
NXT_11 :                SUB     SI, 0020H
                        MOV     CL, (SI)                ; COUNT EI
                        CMP     CL, 00H
                        JNZ     NXT_15
                        ADD     SI, 0040H
                        MOV     DI, BASE_ADD_DIT
                        MOV     DI, 0003H
                        INR     SI
                        MOV     AL, (SI)
;FIRST 16I ELEMENT.
                        DCR     AL
```

```
                        MOV     (BP+DI), AL
;DEFAULT MARK LOADED.
                        INR     DI
                        MOV     (BP+DI), CH
                        DCR     SI
                        MOV     CL, CH
                        MOV     CH, 00H
AGAIN_9 :               INR     DI
                        INR     SI
                        MOV     AL, (SI)
                        MOV     (BP+DI), AL
                        LOOP    AGAIN_9
                        INR     DI
                        MOV     (BP+DI), 2BH
                        MOV     CX, 0005H
AGAIN_10 :              INR     DI
                        MOV     (BP+DI), 00H
                        LOOP    AGAIN_10
                        JMP     RSLV_REAL
NXT_15 :                INR     SI
                        MOV     AL, (SI)
;FIRST ELEMENT OF EI IN AL.
                        ADD     SI, 0040H
                        MOV     CL, 00H
                        MOV     DI, BASE_ADD_DIT
                        ADD     DI, 0003H
AGAIN_11 :              CMP     AL, (SI)
                        JAE     LOAD_CNT
                        INR     CL
                        INR     SI
                        JMP     AGAIN_11
LOAD_CNT :              ADD     SI, FFF0H
                        CMP     CL, 00H
                        JNZ     CHK_NXT_1
                        SUB     SI, 0040H
                        INR     SI
                        MOV     AL, (SI)
                        DCR     AL
                        MOV     (BP+DI), AL
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        MOV     (BP+DI), 'P'
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        MOV     (BP+DI), 00H
                        ADD     SI, 0040H
```

```
                        MOV     AL, (SI)
                        JMP     NXT_16
CHK_NXT_1 :             INR     SI
                        MOV     AL, (SI)
                        DCR     AL
                        MOV     (BP+DI), AL
                        MOV     AH, CL
                        INR     DI
                        MOV     (BP+DI), CL
                        DCR     SI
AGAIN_12 :              INR     SI
                        INR     DI
                        MOV     AL, (SI)
                        MOV     (BP+DI), AL
                        LOOP    AGAIN_12
                        AND     SI, FFF0H
                        MOV     AL, AH
                        MOV     AH, 00H
                        MOV     CL, AL
                        MOV     CH, (SI)
                        ADD     SI, AX
                        SUB     CH, CL
                        MOV     (SI), CH
                        MOV     OFFSET_16I, SI
                        INR     DI
                        MOV     (BP+DI), 'P'
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     SI
                        MOV     AL, (SI)
NXT_16 :                INR     DI
                        MOV     SI, OFFSET_EI
                        INR     SI
                        MOV     AH, (SI)
                        MOV     (BP+DI), AH
                        INR     DI
                        PUSH    DI
                        MOV     CL, 00H
AGAIN_13 :              CMP     AL, (SI)
                        JAE     LOAD_CNT_2
                        INR     DI
                        MOV     (BP+DI), AL
                        INR     SI
                        INR     CL
                        JMP     AGAIN_13
```

```
LOAD_CNT_2 :        AND   SI, FFF0H
                    CMP   CL, 00H
                    JNZ   CHK_NXT_2
                    POP   DI
                    MOV   (BP+DI), 00H
                    JMP   NXT_17
CHK_NXT_2 :         MOV   BX, DI
                    POP   DI
                    MOV   (BP+DI), CL
                    MOV   DI, BX
                    MOV   AH, CL
NXT_17 :            MOV   CH, (SI)
                    SUB   CH, CL
                    PUSH  SI
                    SUB   CL, AH
                    PUSH  SI
                    MOV   SI, OFFSET_16I
                    MOV   CL, (SI)
                    CMP   CH  CL
                    JZ    CONT_1
                    JMP   ERROR_3
CONT_1 :            POP   SI
                    MOV   AL, AH
                    MOV   AH, 00H
                    ADD   SI, AX
                    MOV   (SI), CL
                    DCR   SI
                    MOV   CH, 00H
                    INR   CL
AGAIN_14 :          INR   SI
                    INR   DI
                    MOV   AL, (SI)
                    MOV   (BP+DI), AL
                    LOOP  AGAIN_14
                    JMP   RSLV_REAL
CHK_EI :            MOV   BX, 0000H
                    INR   SI
                    MOV   AL, (SI)
                    ADD   SI, 0020H
                    MOV   DI, BASE_ADD_DIT
                    ADD   DI, 0003H
AGAIN_15 :          CMP   AL, (SI)
                    JAE   LOAD_CNT_3
                    INR   BL
;COUNT INDIVIDUAL_16I (INTEGER).
                    INR   SI
                    JMP   AGAIN_15
```

```
LOAD_CNT_3 :          AND   SI, FFF0H
                      CMP   BL, 00H
                      JNZ   CHK_NXT_3
                      SUB   SI, 0020H
;SI POINTS TO  NORMAL INTEGER ARRAY (NI) .
                      INR   SI
                      MOV   AL, (SI)
;FIRST ELEMENT OF NI.
                      DCR   AL
                      MOV   (BP+DI), AL
;LOAD DEFAULT_MARK.
                      INR   DI
                      MOV   (BP+DI), 00H
                      ADD   SI, 0020H
                      MOV   AL, (SI)
                      MOV   DX, CX
                      JMP   CHK_NXT_4
CHK_NXT_3 :           INR   SI
                      MOV   AL, (SI)
                      DCR   AL
                      MOV   (BP+DI), AL
                      INR   DI
                      MOV   (BP+DI), BL
                      MOV   DX, CX
                      MOV   CX, DX
                      DCR   SI
AGAIN_16 :            INR   SI
                      INR   DI
                      MOV   AL, (SI)
                      MOV   (BP+DI), AL
                      LOOP  AGAIN_16
                      AND   SI, FFF0H
                      MOV   AX, BX
                      ADD   SI, AX
                      SUB   DH, BL
                      MOV   (SI), DH
                      MOV   OFFSET_16I, SI
                      INR   SI
                      MOV   AL, (SI)
CHK_NXT_4 :           MOV   SI, OFFSET_NI
                      INR   SI
                      MOV   BX, 00H
AGAIN_17 :            CMP   AL, (SI)
                      JAE   LOAD_CNT_4
                      INR   SI
                      INR   BL
                      JMP   AGAIN_17
```

```
LOAD_CNT_4 :        AND   SI, FFFOH
                    INR   DI
                    INR   SI
                    MOV   AL, (SI)
                    MOV   (BP+DI), AL
                    INR   DI
                    MOV   (BP+DI), BL
                    DCR   SI
                    CMP   BL, OOH
                    JZ    CHK_NXT_5
                    MOV   CX, BX
AGAIN_18 :          INR   DI
                    INR   SI
                    MOV   AL, (SI)
                    MOV   (BP+DI), AL
                    LOOP  AGAIN_18
                    AND   SI, FFFOH
                    MOV   AX, BX
                    ADD   SI, AX
                    SUB   DL, BL
                    MOV   (SI), DL
                    MOV   OFFSET_NI, SI
CHK_NXT_5 :         MOV   CL, (SI)
                    CMP   CL, OOH
                    JZ    CONT_2
                    PUSH  DI
                    MOV   DI, OFFSET_16I          ;DI AT 16I
                    MOV   BX, OOOOH
AGAIN_19 :          INR   SI
                    INR   DI
                    MOV   AL, (SI)
                    CMP   AL, (BP+DI)
                    JNZ   LOAD_CNT_NI
                    INR   BL
                    JMP   AGAIN_19
LOAD_CNT_NI :       CMP   BL, CL
                    JZ    CONT_3
                    JMP   ERROR_1
CONT_3 :            POP   DI
                    MOV   SI, OFFSET_NI
                    MOV   CH, OOH
                    INR   DI
                    MOV   (BP+DI), CL
AGAIN_20 :          INR   DI
                    INR   SI
                    MOV   AL, (SI)
                    MOV   (BP+DI), AL
                    LOOP  AGAIN_20
```

```
CONT_2 :        MOV   SI, OFFSET_16I
                MOV   CH, (SI)
                SUB   CH, BL
                CMP   CH, 00H
                JNZ   CHK_NXT_6
                MOV   SI, OFFSET_EI
                MOV   CL, (SI)
                CMP   CL, 00H
                JNZ   CHK_NXT_7
                INR   DI
                MOV   (BP+DI), 2BH
                INR   DI
                MOV   (BP+DI), 00H
                INR   DI
                MOV   (BP+DI), 00H
                JMP   RSLV_REAL
CHK_NXT_7 :     INR   SI
                MOV   AL, (SI)
                INR   DI
                MOV   (BP+DI), AL
                DCR   SI
                INR   DI
                MOV   (BP+DI), CL
AGAIN_21 :      INR   SI
                INR   DI
                MOV   AL, (SI)
                MOV   (BP+DI), AL
                LOOP  AGAIN_21
                INR   DI
                MOV   (BP+DI), 00H
                JMP   RSLV_REAL
CHK_NXT_6 :     ADD   SI, BX
                MOV   (SI), CH
                MOV   OFFSET_16I, SI
                INR   SI
                MOV   AL, (SI)
                MOV   SI, OFFSET_EI
                MOV   BX, 0000H
AGAIN_22 :      INR   SI
                CMP   AL, (SI)
                JAE   LOAD_CNT_5
                INR   BL
                JMP   AGAIN_22
LOAD_CONT_5 :   CMP   BL, 00H
                JZ    LOAD_16EI_ONLY
                AND   SI, FFF0H
                INR   DI
                MOV   AH, (SI)
```

```
                        INR     SI
                        MOV     AL, (SI)
                        MOV     (BP+DI), AL
                        INR     DI
                        MOV     CX, BX
                        DCR     SI
AGAIN_23 :              INR     SI
                        INR     DI
                        MOV     AL, (SI)
                        MOV     (BP+DI), AL
                        LOOP    AGAIN_23
                        SUB     AH, BL
                        MOV     AL, AH
                        MOV     AH, 00H
                        ADD     SI, AX
                        MOV     OFFSET_EI, SI
LOAD_16EI_ONLY :        MOV     AH, AL
                        MOV     SI, OFFSET_16I
                        MOV     AL, (SI)
                        CMP     AL, AH
                        JZ      NXT_18
                        JMP     ERROR_3
NXT_18 :                INR     DI
                        MOV     (BP+DI), AL
                        MOV     CL. AL
AGAIN_24 :              INR     DI
                        INR     SI
                        MOV     AL, (SI)
                        MOV     (BP+DI), AL
                        LOOP    AGAIN_24
RSLV_REAL :             MOVE    AX, 0000H
                        MOV     BX, 0000H
                        MOV     OFFSET_REAL, DI
                        MOV     DI, BASE_ADD_DIT
                        ADD     DI, 0002H
                        PUSH    DI
                        ADD     DI, 0003H
                        MOV     AL, (BP+DI)
                        ADD     BL, AL
                        ADD     DI, AX
                        MOV     CX, 0002H
AGAIN_25 :              ADD     DI, 0002H
                        MOV     AL, (BP+DI)
                        ADD     BL, AL
                        ADD     DI, AX
                        INR     DI
                        MOV     AL, (BP+DI)
                        ADD     AL, BL
```

```
                        ADD     DI, BX
                        LOOP    AGAIN_25
                        POP     DI
                        ADD     BL, 09H
                        MOV     (BP+DI), BL
                        ADD     DI, BX
;LOADING OF REAL ARRAY BEGINS HERE.
                        MOV     SI, OFFSET_24R
                        MOV     CH, (SI)
                        CMP     CH, 00H
                        JZ      NXT_19
                        JMP     CHK_NR
NXT_19 :                SUB     SI, 0020H
                        MOV     CL, (SI)
                        CMP     CL, 00H
                        JZ      NXT_2R
                        MOV     DI, OFFSET_REAL
                        INR     SI
                        MOV     AL, (SI)
                        DCR     AL
                        MOV     (BP+DI), AL
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     AL
                        INR     DI
                        MOV     (BP+DI), AL
                        INR     DI
                        MOV     (BP+DI), CL
                        INR     DI
                        MOV     (BP+DI), AL
                        DCR     CL
BG_8 :                  INR     SI
                        INR     DI
                        MOV     AL, (SI)
                        MOV     (BP+DI), AL
                        LOOP    BG_8
                        INR     DI
                        MOV     (BP+DI), 00H
                        SUB     SI, 0020H
                        MOV     CL, (SI)
                        CMP     CL, 00H
                        JNZ     NXT_3R
                        INR     DI
                        MOV     (BP+DI), 2BH
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        MOV     (BP+DI), 00H
```

```
                        JMP     LENGTH_DIT
NXT_2R :                SUB     DI, 0020H
                        MOV     CL, (SI)
                        CMP     CL, 00H
                        JNZ     NXT_4R
                        MOV     DI, OFFSET_REAL
                        MOV     (BP+DI), 2BH
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        MOV     (BP+DI), 10H            ;16RN, 'A'-1
                        INR     DI
                        MOV     (BP+DI), 00H
                        JMP     LENGTH_DIT
NXT_4R :                MOV     DI, OFFSET_REAL
                        INR     SI
                        MOV     AL, (SI)
                        DCR     AL
                        MOV     (BP+DI), AL
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        MOV     (BP+DI), 'Z'
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        INR     AL
                        MOV     (BP+DI), AL
                        INR     DI
                        MOV     (BP+DI), AL
                        INR     DI
                        MOV     (BP+DI), CL
NXT_3R :                INR     SI
                        MOV     SL, (SI)
                        INR     DI
;ELEMENT OF 16_R FROM EXTENDED ARRAY LOADED.
                        LOOP    NXT_3R
                        INR     DI
                        MOV     (BP+DI), 00H
                        JMP     LENGTH_DIT
CHK_NR :                SUB     SI, 0020H
                        MOV     CL, (SI)
                        CMP     CL, 00H
                        JZ      NXT_1R
                        JMP     CHK_ER
```

```
NXT_1R :        SUB   SI, 20H
                MOV   CL, (SI)
                CMP   CL, 00H
                JNZ   NXT_5R
                ADD   SI, 0040H
                MOV   (BP+DI), OFFSET_REAL
                INR   SI
                MOV   AL, (SI)
                DCR   AL
                MOV   (BP+DI), AL
                INR   DI
                MOV   (BP+DI), CH
                DCR   SI
                MOV   CL, CH
                MOV   CH, 00H
BG_9 :          INR   DI
                INR   SI
                MOV   AL, (SI)
                MOV   (BP+DI), AL
                LOOP  BG_9
                INR   DI
                MOV   (BP+DI), 2BH
                MOV   CX, 0005H
BG_10 :         INR   DI
                MOV   (BP+DI), 00H
                LOOP  BG_10
                JMP   LENGTH_DIT
NXT_5R :        INR   SI
                MOV   AL, (SI)
                ADD   SI, 0040H
                MOV   CL, 00H
                MOV   DI, OFFSET_REAL
BG_11 :         CMP   AL, (SI)
                JAE   LOAD_CNT_IR
                INR   CL
                INR   SI
                JMP   BG_11
LOAD_CNT_IR :   AND   SI, FFF0H
                CMP   CL, 00H
                JNZ   CHK_NXT_1R
                SUB   SI, 0040H
                INR   SI
                MOV   AL, (SI)
;FIRST ER ELEMENT.
                DCR   AL
                MOV   (BP+DI), AL
                INR   DI
                MOV   (BP+DI), 00H
```

```
                        INR     DI
                        MOV     (BP+DI), 'Z'
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        MOV     (BP+DI), 00H
                        ADD     SI, 0040H
                        MOV     AL, (SI)
                        JMP     NXT_20
CHK_NXT_1R :            INR     SI
                        MOV     AL, (SI)
                        DCR     AL
                        MOV     (BP+DI), AL
                        MOV     AH, CL
                        INR     DI
                        MOV     (BP+DI), CL
                        DCR     SI
BG_12:                  INR     SI
                        INR     DI
                        MOV     AL, (SI)
                        MOV     (BP+DI), AL
                        LOOP    BG_12
                        AND     SI, FFF0H
                        MOV     AL, AH
                        MOV     AH, 00H
                        MOV     CL, AL
                        MOV     CH, (SI)
                        ADD     SI, AX
                        SUB     CH, CL
                        MOV     (SI), CH
                        MOV     OFFSET_24R, SI
                        INR     DI
                        MOV     (BP+DI), 'Z'
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     DI
                        MOV     (BP+DI), 00H
                        INR     SI
                        MOV     AL, (SI)
NXT_20 :                INR     DI
                        MOV     SI, OFFSET_ER
                        INR     SI
                        MOV     AL, (SI)
                        MOV     (BP+DI), AL
                        INR     DI
                        PUSH    DI
                        MOV     CL, 00H
```

```
BG_13 :          CMP   AL, (SI)
                 JAE   LOAD_CNT_2R
                 INR   DI
                 MOV   (BP+DI), AL
                 INR   DI
                 INR   CL
                 JMP   BG_13
LOAD_CNT_2R :    AND   SI, FFF0H
                 CMP   CL, 00H
                 JNZ   CHK_NXT_2R
                 POP   DI
                 MOV   (BP+DI), 00H
                 JMP   NXT_21
CHK_NXT_2R :     MOV   BX, DI
                 POP   DI
                 MOV   (BP+DI), CL
                 MOV   DI, BX
                 MOV   AH, CL
NXT_21 :         MOV   CH, (SI)
                 SUB   CH, CL
                 PUSH  SI
                 MOV   SI, OFFSET_24R
                 MOV   CL, (SI)
                 CMP   CH, CL
                 JZ    NXT_22
                 JMP   ERROR_3
NXT_22 :         POP   SI
                 MOV   AL, AH
                 MOV   AH, 00H
                 ADD   SI, AX
                 MOV   (SI), CL
                 DCR   SI
                 MOV   CH, 00H
                 INR   CL
BG_14 :          INR   SI
                 INR   DI
                 MOV   AL, (SI)
                 MOV   (BP+DI), AL
                 LOOP  BG_14
                 JMP   LENGTH_DIT
CHK_ER :         MOV   BX, 00H
                 INR   SI
                 MOV   AL, (SI)
                 ADD   SI, 0020H
                 MOV   DI, OFFSET_REAL
BG_15 :          CMP   AL, (SI)
                 JAE   LOAD_CNT_3R
                 INR   BL
```

```
                    INR   SI
                    JMP   BG_15
LOAD_CNT_3R :       AND   SI, FFF0H
                    CMP   BL, 00H
                    JNZ   CHK_NXT_3R
                    SUB   SI, 0020H
                    INR   SI
                    MOV   AL, (SI)
                    DCR   AL
                    MOV   (BP+DI), AL
                    INR   DI
                    MOV   (BP+DI), 00H
                    ADD   SI, 0020H
                    MOV   AL, (SI)
                    MOV   DX, CX
                    INR   BL
                    JMP   CHK_NXT_4R
CHK_NXT_3R :        INR   SI
                    MOV   AL, (SI)
                    DCR   AL
                    MOV   (BP+DI), AL
                    INR   DI
                    MOV   (BP+DI), BL
                    MOV   DX, CX
                    MOV   CX, BX
                    DCR   SI
BG_16 :             INR   SI
                    INR   DI
                    MOV   AL, (SI)
                    MOV   (BP+DI), AL
                    LOOP  BG_16
                    AND   SI, FFF0H
                    MOV   AX, DX
                    AND   SI, AX
                    SUB   DH, BL
                    MOV   (SI), DH
                    MOV   OFFSET_24R, SI
                    INR   SI
                    MOV   AL, (SI)
CHK_NXT_4R :        MOV   SI, OFFSET_NR
                    INR   SI
                    MOV   BX, 0000H
BG_17 :             CMP   AL, (SI)
                    JAE   LOAD_CNT_4R
                    INR   SI
                    INR   BL
                    JMP   BG_17
```

```
LOAD_CNT_4R :       AND   SI, FFFOH
                    INR   DI
                    INR   SI
                    MOV   AL, (SI)
                    MOV   (BP+DI), AL
                    INR   DI
                    MOV   (BP+DI), BL
                    DCR   SI
                    CMP   BL, OOH
                    JZ    CHK_NXT_5R
                    MOV   CX, BX
BG_18 :             INR   DI
                    INR   SI
                    MOV   AL, (SI)
                    MOV   (BP+DI), AL
                    LOOP  BG_18
                    AND   SI, FFFOH
                    MOV   AX, BX
                    ADD   SI, AX
                    SUB   DL, BL
                    MOV   (SI), DI
                    MOV   OFFSET_NR, SI
CHK_NXT_5R :        MOV   CL, (SI)
                    CMP   CL, OOH
                    JZ    CONT_2R
                    PUSH  DI
                    MOV   DI, OFFSET_24R
                    MOV   BX, OOOOH
BG_19 :             INR   SI
                    INR   DI
                    MOV   AL, (SI)
                    CMP   AL, (BP+DI)
                    JNZ   LOAD_CNT_NR
                    INR   BL
                    JMP   BG_19
LOAD_CNT_NR :       CMP   BL, CL
                    JZ    NXT-25
                    JMP   ERROR_1
NXT-25 :            POP   DI
                    MOV   SI, OFFSET_NR
                    MOV   CH, OOH
                    INR   DI
                    MOV   (BP+DI), CL
BG_20 :             INR   DI
                    INR   SI
                    MOV   AL, (SI)
                    MOV   (BP+DI), AL
                    LOOP  BG_20
```

```
CONT_2R :       MOV   SI, OFFSET_24R
                MOV   CH, (SI)
                SUB   CH, BL
                CMP   CH, 00H
                JNZ   CHK_NXT_6R
                MOV   SI, OFFSET_ER
                MOV   CL, (SI)
                CMP   CL, 00H
                JNZ   CHK_NXT_7R
                INR   DI
                MOV   (BP+DI), 2BH
                INR   DI
                MOV   (BP+DI), 00H
                INR   DI
                MOV   (BP+DI), 00H
                JMP   LENGTH_DIT
CHK_NXT_7R :    INR   SI
                MOV   AL, (SI)
                INR   DI
                MOV   (BP+DI), AL
                DCR   SI
                INR   DI
                MOV   (BP+DI), CL
BG_21 :         INR   SI
                INR   DI
                MOV   AL, (SI)
                MOV   (BP+DI), AL
                LOOP  BG_21
                INR   DI
                MOV   (BP+DI), 00H
                JMP   LENGTH_DIT
CHK_NXT_6R :    ADD   SI, BX
                MOV   (SI), CH
                MOV   OFFSET_24R, SI
                INR   SI
                MOV   AL, (SI)
                MOV   SI, OFFSET_ER
                MOV   BX, 0000H
BG_22 :         INR   SI
                CMP   AL, (SI)
                JAE   LOAD_CNT_5R
                INR   BL
                JMP   BG_22
LOAD_CNT_5R :   CMP   BL, 00H
                JZ    LOAD_24ER_ONLY
                AND   SI, FFF0H
                INR   DI
                MOV   AH, (SI)
```

```
                        INR     SI
                        MOV     AL, (SI)
                        MOV     (BP+DI), AL
                        INR     DI
                        MOV     CX, BX
                        DCR     SI
BG_23 :                 INR     SI
                        INR     DI
                        MOV     AL, (SI)
                        MOV     (BP+DI), AL
                        LOOP    BG_23
                        SUB     AH, BL
                        MOV     AL, AH
                        MOV     AH, 00H
                        ADD     SI, AX
                        MOV     OFFSET_ER, SI
LOAD_24ER_ONLY :        MOV     AH, AL
                        MOV     SI, OFFSET_24R
                        MOV     AL, (SI)
                        CMP     AL, AH
                        JZ      CONT_5R
                        JMP     ERROR_3
CONT_5R :               INR     DI
                        MOV     (BP+DI), AL
                        MOV     CL, AL
BG_24 :                 INR     DI
                        INR     SI
                        MOV     AL, (SI)
                        MOV     (BP+DI), AL
                        LOOP    BG_24

PART_2 :

LENGTH_DIT :            MOV     SI, PRESENT_DATA_SEG
                        MOV     AX, 0000H
                        MOV     DX, 0000H
                        MOV     CX, 0000H
                        ADD     SI, 0040H
                        MOV     CL, (SI)
                        MOV     DL, CL
                        SHL     CL
                        ADD     AX, CX
                        ADD     SI, DX
                        ADD     SI, 0002H
                        MOV     CL, (SI)
                        MOV     DL, CL
```

```
SHL    CL                          , 4 TIMES
ADD    AX, CX
ADD    SI, DX
INR    SI
MOV    CL, (SI)
MOV    DL, CL
SHL    CX                          , 5 TIMES
ADD    AX, CX
ADD    SI, DX
ADD    SI, 0002H
MOV    CH, 00H
MOV    CL, (SI)
MOV    DL, CL
SHL    CL                          , 8 TIMES
ADD    AX, CX
ADD    SI, DX
INR    SI
MOV    CH, 00H
MOV    CL, (SI)
MOV    DL, CL
SHL    CX                          , 9 TIMES
ADD    AX, CX
ADD    SI, DX
ADD    SI, 0002H
MOV    CH, 00H
MOV    CL, (SI)
MOV    DL, CL
SHL    CL
ADD    CL, DL
ADD    AX, CX
ADD    SI, DX
ADD    SI, 0002H
MOV    CL, (SI)
MOV    DL, CL
SHL    CX                          , 5 TIMES
ADD    AX, CX
ADD    SI, DX
INR    SI
MOV    CH, 00H
MOV    CL, (SI)
MOV    DL, CL
SHL    CL
ADD    CL, DL
SHL    CX                          , 4 TIMES
ADD    AX, CX
ADD    SI, DX
ADD    SI, 0002H
MOV    CH, 00H
```

```
                MOV     CL, (SI)
                MOV     DL, CL
                SHL     CX                          , 9 TIMES
                ADD     AX, CX
                ADD     SI, DX
                INR     SI
                MOV     CH, 00H
                MOV     CL, (SI)
                MOV     DL, CL
                SHL     CL
                ADD     CL, DL
                SHL     CX                          , 8 TIMES
                ADD     AX, CX
                SHR     AX                          , 7 TIMES
                INR     AX
                MOV     SI
                MOV     (SI), BASE_ADD_DIT
                MOV     (SI), AX
```

PART_3 :

```
BLK_ALTR :      MOV     AL, PRESENT_DI
                MOV     CH, 00H
                MOV     SI, PRESENT_DATA_SEG
                MOV     BX, (SI)
                MOV     SI, BASE_ADD_DIT
                MOV     CL, AL
                SHL     CL
                ADD     SI, CX
                CMP     BX, (SI)
                JA      CHK_NXT_1
                JE      OUT
                JB      CHK_NXT_2
CHK_NXT_1 :     SUB     BX, (SI)
                MOV     SI, BASE_ADD_DIT
                MOV     DX, (SI)
                ADD     DX, BX
                SHR     DX
                MOV     SI, BASE_ADD_PIT
                ADD     DX, (SI)
                CMP     DX, AV_MEM_BLOCKS
                JAE     CONT_1
                JMP     ERROR_3
CONT_1 :        MOV     SI, BASE_ADD_DIT
                ADD     SI, 0008H
                MOV     DL, 04H
```

```
                    CMP     AL, DL
                    JZ      NXT_1
                    MOV     CX, 0000H
NXT_0 :             ADD     CX, (SI)
                    DCR     DL
                    CMP     DL, AL
                    JZ      NXT_2
                    SUB     SI, 0002H
                    JMP     NXT_0
NXT_2 :             MOV     SI, BASE_ADD_DIT
                    MOV     DX, 03FEH
                    SUB     DX, (SI)
                    MOV     ST_BLK_DI4_PREV, DX
                    SUB     DX, BX
                    MOV     ST_BLK_DI4_NEW, DX
                    SHL     DX                      , 3 TIMES
                    MOV     ES, DX
                    MOV     DI, 0000H
                    MOV     DX, ST_BLK_DI4_PREV
                    SHL     DX                      , 3 TIMES
                    MOV     DS, DX
                    MOV     SI, 0000H
                    SHL     CX                      , 7 TIMES
                REP MOVSB
                    MOV     CX, 0000H
                    MOV     DS, CX
                    MOV     ES, CX
NXT_1 :             MOV     SI, BASE_ADD_DIT
                    ADD     (SI), BX
                    MOV     CH, 00H
                    MOV     CL, AL
                    SHL     CL
                    ADD     SI, CX
                    ADD     (SI), BX
                    JMP     RSLV_PROGRAM_ENTRY
CHK_NXT_2 :         SUB     BX, (SI)
                    NEG     BX
                    MOV     SI, BASE_ADD_DIT
                    MOV     DL, 04H
                    CMP     AL, DL
                    JZ      NXT_4
                    MOV     CX, 0000H
NXT_3 :             ADD     CX, (SI)
                    DCR     DL
                    CMP     AL, DL
                    JZ      NXT_2
                    SUB     SI, 0002H
                    JMP     NXT_3
```

```
                        MOV     SI, BASE_ADD_DIT
                        MOV     DX, 03F0H
                        SUB     DX, (SI)
                        ADD     DX, CX
                        DCR     DX
                        MOV     END_BLK_SRC, DX
                        SUB     DX, BX
                        MOV     END_BLK_DST, DX
                        SUB     DX, 01FFH
                        SHL     DX                          , 3 TIMES
                        MOV     ES, DX
                        MOV     DI, FFFFH
                        MOV     DX, END_BLK_SRC
                        SUB     DX, 01FFH
                        SHL     DX                          , 3 TIMES
                        MOV     DS, DX
                        MOV     SI, FFFFH
                        STD
                    REP MOVSB
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     ES, CX
NXT_4 :                 MOV     SI, BASE_ADD_DIT
                        SUB     (SI), BX
                        MOV     CH, 00H
                        MOV     CL, AL
                        SHL     CL
                        ADD     SI, CX
                        SUB     (SI), BX
                        JMP     RSLV_PROGRAM_ENTRY
;MONITOR CONTINUES TO GET THE NEXT PROGRAM ENTRY MODULE : LEVEL_3,
;F.C. 4.1
```

TITLE : MODULE : INITIATE PROGRAM DEVELOPMENT LEVEL_3, F.C. 4.1

ANATOMY :

THIS MODULE HAS TWO ENTRY POINTS (1) AFTER NEW SPECIFICATIONS FOR DATA AREA WERE ENTERED (MODULE : RSLV_DIT, LEVEL_2) OR (2) IF DEFAULT LEARN AND DATA INDICES WERE ACCEPTED.

THE ENTRIES ARE CHECKED FOR WHETHER THE AUTO LINE NUMBER GENERATION IS DEMANDED OR THE LINE NUMBERS ARE ENTERED BY USER IN THE INITIAL STAGE ONLY. LATER THE PROGRAM RESOLVES KEY ENTRIES INTO INTERPRETER CODES THROUGH A NESTED CALL STRUCTURE.

REGISTER USAGE :

AX, DX : CRUNCHING LINE NUMBERS.

REFERENCE MEMORY LOCATIONS :

AUTO_MARK, NXT_LINE_NO, ICP_LOC_DS, ICP_LOC_SI.

SUBROUTINES :

RSLV_KB_ENTRIES, LOAD_RAM_FILE.

```
MONIOR CONTINUES AFTER BLK_ALTER IF DATA SELECT IS EXECUTED.
                        LED_INDICATOR_ON
                        UNMASK_KBIRQ
;THIS PROGRAM LOADS BP, DI AT KBB AND LOADS BX, CX AT 0000H.
                        HLT
                        CMP    AL, 'ENTER'
                        JZ     CONT_1
                        JMP    ERROR_1
CONT_1 :                LED_INDICATOR_OFF
                        BLANK_ALL
                        MASK_KBIRQ
;THIS  MACRO  SAVES DI AND LOADS DI AT 0000 THAT IS  POINTING  THE
;BASE  OF KEY BOARD BUFFER (KBB). LEVEL_3 REFERS TO A LOCATION  IN
;MONITOR WHERE THE PROGRAM FLOW CONTINUES AFTER LRN  AND NO_DAT OR
;AFTER NO_LRN (NOLRN, NO_DAT) WHERE FLOW CONTINUES AT LEVEL_3.


LEVEL_3 :               CMP    (BP+DI), 'AUTO'
;AUTO KEY DIFFERENTIATES MODE SELECTION KEYS FROM COMMAND KEYS.
                        JB     NXT_01
                        JE     CONT_3
                        JMP    ERROR_1
NXT_01 :                JMP    CONT_2
CONT_3 :                MOV    AUTO_MARK, FFH
                        MOV    NEXT_LINE_NO, 0002H
```

```
                    INR     DI
                    MOV     AL, (BP+DI)
                    CMP     AL, 09H
                    JE      NXT_1
                    CMP     AL, 'ENTER'
                    JZ      NXT_2
                    JMP     ERROR_1
NXT_1 :             SHL     AL                          , 4 TIMES
                    MOV     DH, AL
                    INR     DI
                    MOV     (BP+DI), AL
                    CMP     AL, 09H
                    JBE     NXT_3
                    JMP     ERROR_1
NXT_3 :             ADD     DH, AL
                    INR     DI
                    MOV     AX, (BP+DI)
                    CMP     AH, 09H
                    JZ      NXT_4
                    JMP     ERROR_1
NXT_4 :             SHL     AH                          , 4 TIMES
                    MOV     DL, AH
                    CMP     AL, 09H
                    JZ      NXT_5
                    JMP     ERROR_1
NXT_5 :             ADD     DL, AL
                    INR     DI
                    CMP     (BP+DI), 'ENTER'
                    JZ      NXT_2
                    JMP     ERROR_1
NXT_2 :             MOV     DX, NEXT_LINE_NO
                    MOV     BX, 0000H
                    DCR     DI
                    MOV     AL, DH
                    MOV     AH, DH
                    AND     AL, 0FH
                    AND     AH, F0H
                    MOV     CH, 04H
                    SHR     AH, CL
                    INR     BH
                    MOV     (BP+DI), AH
                    INR     DI
                    INR     BH
                    MOV     (BP+DI), AL
                    MOV     AH, DL
                    MOV     AL, DL
                    AND     AH, F0H
                    AND     AL, 0FH
```

```
                        MOV     CL, 04H
                        SHR     AH, CL
                        INR     DI
                        INR     BH
                        MOV     (BP+DI), AH
                        INR     BH
                        INR     DI
                        MOV     (BP+DI), AL
                        MOV     AX, DX
                        ADD     AX, 0002H
                        DAA
                        JNZ     CONT_5
                        JMP     ERROR_1
CONT_5 :                MOV     NEXT_LINE_NO, AX
                        LED_INDICATOR_ON
MACRO                   UNMASK KBIRQ_1
                        MOV     DX, CONTROL_PORT_KBDC
                        MOV     AL, FEH
                        OUT     DX, AL
                        ENDM
                        INT     VECTOR_KBIRQ
                        HLT
                        LED_INDICATOR_OFF
                        BLANK_ALL
                        MASK    KBIRQ
;PACKING OF LINE NUMBERS IS PERFORMED.
CONT_2 :                MOV     CX, ICP_LOC_DS
                        MOV     DS, CX
                        MOV     SI, ICP_LOC_SI
                        ADD     SI, 0008H
CONT_9 :                INR     SI
                        MOV     AL, (BP+DI)
                        INR     DI
                        MOV     AH, (BP+DI)
                        SHL     AL                      , 4 TIMES
                        ADD     AL, AH
                        MOV     (SI), AL
                        INR     SI
                        INR     DI
                        MOV     AL, (BP+DI)
                        INR     DI
                        MOV     AH, (BP+DI)
                        SHL     AL                      , 4 TIMES
                        ADD     AL, AH
                        MOV     (SI), AL
                        CALL    RSLV_KB_ENTRIES
```

```
;INTERPRETER CODES CORRESPONDING TO THE SENTENCE ARE LOADED.
;AFTER RETURN FROM THE PROGRAM RSLV_KB_ENTRIES ICP IS ALSO LOADED
;AND NOW THE REMAINING BYTES OF 8 BYTE BLOCK ARE LOADED 00H.
LOAD_AGAIN :        INR    SI
                    TEST   SI, 0007H
                    JZ     NXT_6
                    MOV    (SI), 00H
                    JMP    LOAD_AGAIN
NXT_6 :             CALL   LOAD_RAM_FILE
                    MOV    DI
                    MOV    BP, BASE_ADD_KBB
                    CMP    AUTO_MARK, FFH
                    JNZ    CONT_6
                    MOV    DX, NEXT_LINE_NO
                    MOV    BX, 0000H
                    MOV    CH, 00H
                    MOV    AH, DH
                    MOV    AL, DH
                    AND    AH, F0H
                    AND    AL, 0FH
                    SHR    AH, CL
                    INR    BH
                    MOV    (BP+DI), AH
                    INR    BH
                    INR    DI
                    MOV    (BP+DI), AL
                    MOV    AL, DL
                    MOV    AH, DL
                    AND    AL, 0FH
                    AND    AH, F0H
                    MOV    CL, 04H
                    SHR    AH, AL
                    INR    BH
                    INR    DI
                    MOV    (BP+DI), AH
                    INR    BH
                    INR    DI
                    MOV    (BP+DI), AL
                    MOV    AX, DX
                    ADD    AX, 0002H
                    DAA
                    JNZ    CONT_7
                    JMP    ERROR_1
CONT_7 :            MOV    NEXT_LINE_NO, AX
CONT_6 :            LED_INDICATOR_ON
                    UNMASK_KBIRQ_1
                    HLT
                    CMP    AL, 'ENTER'
```

```
                        JZ      CONT_8
                        JMP     ERROR_1
CONT_8 :                JMP     CONT_9
;SYSTEM  RETURNS TO NEXT TO POST THAT IS LEVEL_1 AFTER A RESET  IS
;FED IN.
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3, SUBLEVEL_1, CL_3(1)], F.C. 4.1

ANATOMY :

PROGRAM DETECTS THE COMMAND KEY ENTERED AND JUMPS TO RESPECTIVE LOCATIONS FOR FURTHER EXECUTION.

REGISTER USAGE :

AL : COMMAND KEY ENTRY.

```
RSLV_KB_ENTRIES :       INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, 'END'
                        JNZ     CONT_1
                        JMP     LABEL_END
CONT_1 :                SHL     AL
;ACTION GROUP = LEFT SHIFTED KEY CODE.
                        CMP     AL, C0H
                        JA      CONT_2
                        JB      CONT_3
                        JMP     LABEL_LET
CONT_3 :                CMP     AL, A0H
                        JA      CONT_4
                        JB      CONT_5
                        JMP     LABEL_INB
CONT_5 :                CMP     AL, 90H
                        JA      CONT_6
                        JB      CONT_7
                        JMP     LABEL_DLY
CONT_7 :                CMP     AL, 88H
                        JNZ     CONT_8
                        JMP     LABEL_RET
CONT_6 :                CMP     AL, 98H
                        JNZ     CONT_8
                        JMP     LABEL_OUB
CONT_4 :                CMP     AL, B0H
                        JB      CONT_9
                        JA      CONT_A
                        JMP     LABEL_INR
CONT_9 :                CMP     AL, A8H
                        JNZ     CONT_8
                        JMP     LABEL_INW
CONT_A :                CMP     AL, B8H
                        JNZ     CONT_8
                        JMP     LABEL_OUW
```

```
CONT_8 :            JMP    ERROR_1
CONT_2 :            CMP    AL, E0H
                    JB     CONT_B
                    JA     CONT_C
                    JMP    LABEL_DSP
CONT_B :            CMP    AL, D0H
                    JB     CONT_D
                    JA     CONT_E
                    JMP    LABEL_IF
CONT_D :            CMP    AL, C8H
                    JNZ    CONT_8
                    JMP    LABEL_DCR
CONT_E :            CMP    AL, D8H
                    JNZ    CONT_8
                    JMP    LABEL_GSB
CONT_C :            CMP    AL, F0H
                    JB     CONT_F
                    JA     CONT_10
                    JMP    LABEL_GTO1
CONT_F :            CMP    AL, E8H
                    JNZ    CONT_8
                    JMP    LABEL_FOR
CONT_10 :           CMP    AL, F8H
                    JNZ    CONT_8
                    JMP    LABEL_NXT
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_1 : LOAD_ICP_END

ANATOMY :

THIS PROGRAM LOADS ONLY THE GROUP BYTE IN INTERPRETER CODE (IC) FIELD AND IN STATUS OF PI THE PHYSICAL END OF THE PROGRAM IS MARKED EXPLICITLY.

REGISTER USAGE :

AL : SUB_GROUP.
CX : MANAGING SEGMENT, POINTING STATUS OF PI.
CL : MANAGING STATUS.

REFERENCE MEMORY LOCATIONS :

PRESENT PI.

```
LABEL_END :     INR   SI
                MOV   (SI), AL
                INR   DI
                CMP   (BP+DI), 'ENTER'
                JZ    NXT_1
                JMP   ERROR_1
NXT_1 :         PUSH  DS
                PUSH  SI
                MOV   CX, 0000H
                MOV   DS, CX
                MOV   SI, BASE_ADD_PIT
                MOV   CL, PRESENT_PI
                ADD   CL, 03H
                SHL   CL                          , 4 TIMES
                ADD   CX, 0003H
                ADD   SI, CX
                MOV   CL, (SI)
                TEST  CL, 80H
                JZ    NXT_2
                JMP   ERROR_1
NXT_2 :         ADD   CL, 80H
                MOV   (SI), CL
                POP   SI
                POP   DS
                RET
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_2 : LOAD_ICP_RET

ANATOMY :

THIS COMMAND KEY IS USED TO TERMINATE SUBROUTINE. EXECUTION OF THE CODE RETURNS THE PROGRAM FLOW TO THE NEXT SENTENCE FROM WHERE SUBROUTINE IS CALLED (GSB). ONLY ACTION GROUP IS LOADED IN THE IC FIELD.

```
LABEL_RET :        INR    SI
                   MOV    (SI), AL
                   CMP    (BP+DI), 'ENTER'
                   JZ     NXT_1
                   JMP    ERROR_1
NXT_1 :            RET
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_3 : LOAD_ICP_LET

ANATOMY :

PROGRAM FILLS THE IC FIELD DEPENDING UPON THE VARIOUS SUBGROUPS.
IF A VARIABLE IS ENCOUNTERED THE FIELD IS LOADED INTIALLY THEN THE VARIABLE FOLLOWED BY THE INDEX. THIS ACTION IS EXECUTED BY CALLING A PROGRAM VAR_RSLV_0, L_3 (1,1).
IF A NUMERIC CONSTANT IS FOUND SIMILARLY AS FOR VARIABLE IDENTIFIER IS LOADED THEN THE FORMATED NUMERIC CONSTANT IS LOADED IN IC FIELD. THIS ACTION IS EXECUTED BY CALLING A PROGRAM RSLV_CNST, L_3(1,1).

REGISTER USAGE :

AL : KB ENTRIES.
CL : FIELD OF VARIABLE FORMED IN SUBROUTINE VAR_RSLV_0
CH : FIELD OF VARIABLE ON LHS USED FOR DETERMINATION OF SUBGROUP AND TO FORM CONVERSION OPERATOR.
BP : BASE_ADD_CNVRT_TAB.

REFERENCE MEMORY LOCATIONS :

MARK_1, MARK_2, COUNTER_1

SUBROUTINES :

VAR_RSLV_0, RSLV_CNST.

```
LABEL_LET :          INR   SI
                     PUSH  DS
                     MOV   CX, 0000H
                     MOV   DS, CX
                     MOV   MARK_1, SI
;SAVE  CONTENTS OF SI IN A MEMORY LOCATION TO LOAD SUBGROUP  AFTER
;CONFIRMATION.
                     POP   DS
                     INR   DI
                     MOV   AL, (BP+DI)
                     CMP   AL, 'A'
                     JAE   CONT_00
                     JMP   ERROR_1
CONT_00 :            CMP   AL, 'Z'
                     JBE   CONT_01
```

```
                        JMP     ERROR_1
CONT_01 :               CALL    VAR_RSLV_0
;SUBROUTINE  WHICH CHECKS TYPE OF VARIABLES, ASSOCIATES FIELD  AND
;LOADS  IC  FIELD  WITH  IDENTIFIER, VARIABLE AND INDEX, L_3(1,1).
                        INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, '='
                        JZ      CONT_02
                        JMP     ERROR_1
CONT_02 :               PUSH    AX
                        INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, 'CNVRT'
                        JNZ     CONT_03
                        JMP     RSLV_CNVRT
CONT_03 :               PUSH    DS
                        PUSH    CX
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     MARK_2, 00H
;A MEMORY LOCATION WHERE MARK OF UNARY SIGN IS STORED.
                        POP     CX
                        POP     DS
                        MOV     CH, CL
;SAVE FIELD OF VARIABLE ON LHS FOR FURTHER CHECK AND FORMATION  OF
;CONVERSION OPERATOR AND SUBGROUP.
                        CMP     AL, '-'
                        JNZ     NXT_CHAIN
                        MOV     MARK_2, FFH
                        INR     DI
                        MOV     AL, (BP+DI)
NXT_CHAIN :             CMP     AL, '.'
                        JA      CHK_VAR
                        CALL    RSLV_CNST
;SUBROUTINE WHICH CHECKS TYPES OF NUMERIC CONSTANTS, CONVERTS  AND
;LOADS IC FIELD WITH IDENTIFIER AND FORMATED NUMERIC CONSTANT,
;L_3 (1,1).
                        JMP     GEN_ACTION_GROUP
CHK_VAR :               CMP     AL, 'Z'
                        JBE     CONT_04
                        JMP     ERROR_1
CONT_04 :               CALL    VAR_RSLV_0
GEN_ACTION_GROUP :      TEST    CH, 07H
                        JNZ     CONT_05
                        JMP     CHK_16R
CONT_05 :               TEST    CH, 01H
                        JZ      CHK_24R
                        TEST    CH, 04H
```

```
                        JZ      CHK_8I
                        TEST    CL, 01H
                        JZ      NXT_2
                        PUSH    SI
                        PUSH    CX
                        PUSH    DS
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     SI, MARK_1
;LOAD SI POINTING GROUP AND SUBGROUP FIELD FROM WHERE SAVED.
                        POP     DS
                        MOV     (SI), C4H
                        POP     CX
                        POP     SI
                        JMP     FORM_CNV_OPR
NXT_2 :                 TEST    CL, 04H
                        JZ      CONT_4
                        JMP     ERROR_1
CONT_4 :                PUSH    SI
                        PUSH    CX
                        PUSH    DS
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     SI, MARK_1
                        POP     DS
                        MOV     (SI), C3H
                        POP     CX
                        POP     SI
                        JMP     FORM_CNV_OPR
CHK_8I :                TEST    CL, 01H
                        JNZ     CONT_5
                        JMP     ERROR_1
CONT_5 :                TEST    CL, 04H
                        JZ      CONT_6
                        JMP     ERROR_1
CONT_6 :                PUSH    SI
                        PUSH    CX
                        PUSH    DS
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     SI, MARK_1
                        POP     DS
                        MOV     (SI), C5H
                        POP     CX
                        POP     SI
                        JMP     FORM_CNV_OPR
CHK_24I :               TEST    CL, 01H
                        JNZ     NXT_3
```

```
                    PUSH   SI
                    PUSH   CX
                    PUSH   DS
                    MOV    CX, 0000H
                    MOV    DS, CX
                    MOV    SI, MARK_1
                    POP    DS
                    MOV    (SI), C2H
                    POP    CX
                    POP    SI
                    JMP    FORM_CNV_OPR
NXT_3 :             PUSH   SI
                    PUSH   CX
                    PUSH   DS
                    MOV    CX, 0000H
                    MOV    DS, CX
                    MOV    SI, MARK_1
                    POP    DS
                    MOV    (SI), C4H
                    POP    CX
                    POP    SI
                    JMP    FORM_CNV_OPR
CHK_16R :           TEST   CL, 01H
                    JZ     NXT_4
                    PUSH   SI
                    PUSH   CX
                    PUSH   DS
                    MOV    CX, 0000H
                    MOV    DS, CX
                    MOV    SI, MARK_1
                    POP    DS
                    MOV    (SI), C4H
                    POP    CX
                    POP    SI
                    JMP    FORM_CNV_OPR
NXT_4 :             TEST   CL, 04H
                    JZ     CONT_07
                    JMP    ERROR_1
CONT_7 :            PUSH   SI
                    PUSH   CX
                    PUSH   DS
                    MOV    CX, 0000H
                    MOV    DS, CX
                    MOV    SI, MARK_1
                    POP    DS
                    MOV    (SI), C3H
                    POP    CS
                    POP    SI
```

```
FORM_CNV_OPR :          AND    CH, 07H
                        MOV    CL, CH
                        SHL    CL                          , 4 TIMES
                        ADD    CH, CL
                        PUSH   CX
;FIELD  OF RECENTLY FOUND VARIABLE/NUMERIC CONSTANT IS  STORED  IN
;REGISTER CH.
                        PUSH   DS
                        MOV    CX, 0000H
                        MOV    DS, CX
                        MOV    AL, MARK_2
                        POP    DS
                        INR    SI
                        INCREMENT_SI
                        MOV    (SI), AL
                        MOV    AL, (BP+DI)
                        CMP    AL, '-'
                        JAE    CONT_8
                        JMP    ERROR_1
                        CMP    AL, '/'
                        JBE    NXT_5
                        CMP    AL, 'ENTER'
                        JZ     CONT_9
                        JMP    ERROR_1
CONT_9 :                JMP    RSLV_ENTER
NXT_5 :                 PUSH   AX
;STORE OPERATOR ON STACK.
                        MOV    COUNTER_1, 01H
;COUNTER_1,  A  MEMORY LOCATION WHICH KEEPS RECORD  OF  NUMBER  OF
;ENTRIES MADE ON STACK.
                        INR    DI
                        MOV    AL, (BP+DI)
                        CMP    AL, '.'
                        JA     CHK_VAR_1
                        CALL   RSLV_CNST
                        JMP    CHK_VALIDITY
CHK_VAR_1 :             CMP    AL, 'Z'
                        JBE    CONT_08
                        JMP    ERROR_1
CONT_08 :               CALL   VAR_RSLV_0
CHK_VALIDITY :          TEST   CH, 04H
;CHECK VALIDITY OF THE ENTRIES IN CONFIRMATION TO THE SUBGROUPS,
;SEE SECTION 4.    .
                        JA     NXT_7
                        JB     NXT_8
                        TEST   CL, 01H
                        JZ     CONT_09
                        JMP    ERROR_1
```

```
CONT_09 :               JMP    CHK_NXT_OPR
NXT_7 :                 TEST   CL, 01H
                        JNZ    CONT_0A
                        JMP    ERROR_1
CONT_0A :               JMP    CHK_NXT_OPR
NXT_8 :                 TEST   CH, 01H
                        JZ     NXT_9
                        TEST   CL, 01H
                        JNZ    CONT_0B
                        JMP    ERROR_1
CONT_0B :               TEST   CL, 04H
                        JZ     CONT_0C
                        JMP    ERROR_1
CONT_0C :               JMP    CHK_NXR_OPR
NXT_9 :                 TEST   CL_01H
                        JZ     CONT_0D
                        JMP    ERROR_1
CONT_0D :               TEST   CL_04H
                        JZ     CHK_NXT_OPR
                        JMP    ERROR_1
CHK_NXT_OPR :           INR    DI
                        MOV    AL, (BP+DI)
                        CMP    AL, '-'
                        JAE    CONT_05
                        JMP    ERROR_1
CONT_0F :               CMP    AL, '/'
                        JBE    CONT_10
                        CMP    AL, 'ENTER'
                        JNZ    CONT_10
                        JMP    RSLV_ENTER
CONT_10 :               MOV    CL, TOP_OF_STACK
                        CMP    AL, CL
;PRESENT OPERATOR COMPARED WITH THE OPERATOR PUSHED ON THE STACK.
                        JBE    NXT_B
                        PUSH   AX
                        INR    COUNTER_1
                        JMP    NXT_CHAIN
;GO BACK TO CHECK NEXT VARIABLE/NUMERIC CONSTANT PRESENT.
NXT_B :                 INR    SI
MACRO                   INCREMENT_SI
                        TEST   SI, 0007H            ; TEST FOR 9th SI
                        JZ     AGAIN_1
                        JMP    COMPLETE
AGAIN_1 :               MOV    (SI), FFH
;PUT FF IN 9th SI TO INDICATE THE 8 BYTE BLOCK IS COMPLETE AND
;NEXT 8 BYTE BLOCK IS BEING USED.
                        TEST   SI, 00FFH
                        JNZ    AGAIN_2
```

```
                    JMP     COMPLETE
AGAIN_2 :           PUSH    SI
                    PUSH    DS
                    PUSH    CX
                    MOV     CX, 0000H
                    MOV     DS, CX
                    MOV     SI, BASE_ADD_PIT
                    MOV     CX, (SI)
                    INR     CX
                    MOV     (SI), CX
                    SHL     CX
                    ADD     CX, BASE_ADD_DIT
                    SHR     CX
                    JNC     AGAIN_3
                    INR     CX
AGAIN_3 :           CMP     CX, AV_MEM_BLOCKS
                    JBE     AGAIN_4
                    JMP     ERROR_3
AGAIN_4 :           MOV     CH, 00H
                    MOV     CL, PRESENT_PI
                    ADD     CL, 03H
                    SHL     CL                          , 4TIMES
                    ADD     SI, CX
                    MOV     CX, (SI)
                    INR     CX
                    MOV     (SI), CX
                    MOV     CH, 00H
                    MOV     CL, PRESENT_STATUS_PI
                    SHL     CL
                    SHL     CL
                    ADD     SI, CX
                    ADD     SI, 0002H
                    MOV     CX, (SI)
                    INR     CX
                    MOV     (SI), CX
COMPLETE :          NOP
                    ENDM
                    MOV     (SI), AL
LOAD_AGAIN :        POP     AX
                    DCR     COUNTER_1
                    INR     SI
                    INCREMENT_SI
                    MOV     (SI), AL
                    CMP     COUNTER_1, 00H
                    JNZ     LOAD_AGAIN
                    JMP     NXT_CHAIN
RSLV_ENTER :        RET
```

```
RSLV_CNVRT :        INR    SI
                    PUSH   DS
                    MOV    CX, 0000H
                    MOV    DS, CX
                    MOV    MARK_1, SI
                    POP    DS
                    INR    DI
                    MOV    AL, (BP+DI)
                    CMP    AL, '.'
                    JB     CONT_11
                    JMP    ERROR_1
CONT_11 :           INR    DI
                    MOV    AL, (BP+DI)
                    CMP    AL, ','
                    JZ     CHK_CNVRT_VAR
                    CMP    AL, 'ENTER'
                    JZ     CONT_12
                    JMP    ERROR_1
CONT_12 :           PUSH   SI
                    PUSH   DS
                    MOV    CX, 0000H
                    MOV    DS, CX
                    MOV    SI, MARK_1
                    POP    DS
                    MOV    (SI), B8H
                    POP    SI
CHK_NXT_1 :         DCR    DI
                    MOV    AL, (BP+DI)
                    PUSH   BP
                    PUSH   DI
                    MOV    BP, BASE_ADD_CNVRT_TAB
                    SHL    AL
                    MOV    AH, 00H
                    MOV    DI, AX
                    MOV    AX, (BP+DI)
                    INR    SI
                    INCREMENT_SI
                    MOV    (SI), AH
                    INR    SI
                    INCREMENT_SI
                    MOV    (SI), AL
                    POP    DI
                    POP    BP
                    INR    SI
                    INCREMENT_SI
                    MOV    (SI), 00H
                    INR    SI
                    INCREMENT_SI
```

```
                        MOV     (SI), 30H
                        RET
CHK_CNVRT_VAR :         INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, '.'
                        JA      CONT_13
                        JMP     ERROR_1
CONT_13 :               CALL    VAR_RSLV_0
                        INR     DI
                        CMP     (BP+DI), 'ENTER'
                        JZ      CONT_14
                        PUSH    SI
                        PUSH    DS
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     SI, MARK_1
                        POP     DS
                        MOV     (SI), B9H
                        POP     SI
CONT_14 :               SUB     DI, 00002H
                        CMP     (BP+DI)
                        JZ      CONT_15
                        DCR     DI
CONT_15 :               JMP     CHK_NXT_1
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_4 : LOAD_ICP_INB

ANATOMY :

THE IC CODE FIELD TO INPUT A BYTE FROM A PORT OF WHICH THE ADDRESS IS SPECIFIED IN THE KEY BOARD ENTRIES, IS GENERATED. THE DESTINATION OF THE INPUTED BYTES IS ALSO SPECIFIED IN IC FIELD. A DISTINGUSHED SUBGROUP OF THE MODULE DEVELOPS INTERPRETER CODE TO ACQUISE AND LOAD NUMERIC CONSTANT FROM KB, AFTER CONVERSION INTO THE INTERNAL FORMS OF REPRESENTATION.

REGISTER USAGE :

AL : KEY CODE ENTRIES.
AH,AL : PACKING THE ADDRESS NIBBLES.
CH : GROUP+SUBGROUP FIELD OF IC.
DX : CALCULATING ADC CHANNEL ADDRESS.
CX : MANAGING SEGMENT.

REFERENCE MEMORY LOCATIONS :

MARK_1.

SUBROUTINES :

VAR_RSLV_0, L_3 (1,1).

```
LABEL_INB :     MOV   AH, 00H
                MOV   CH, AL
                INR   DI
                MOV   AL, (BP+DI)
                CMP   AL, '.'
                JB    CHK_ADD
                JA    CONT_1
                JMP   ERROR_1
CONT_1 :        CMP   AL, 'Z'
                JA    CONT_2
                JMP   CHK_VAR
CONT_2 :        CMP   AL, 'CH'
                JNZ   CONT_3
                JMP   CHK_CH
CONT_3 :        CMP   AL, 'KB'
                JZ    CHK_KB
                JMP   ERROR_1
CHK_KB :        INR   DI
                CMP   (BP+DI), 'ENTER'
```

```
                    JZ      CONT_4
                    JMP     ERROR_1
CONT_4 :            INR     SI
                    MOV     (SI), CH
                    INR     SI
                    MOV     (SI), KBDC_ADD_L
                    INR     SI
                    MOV     (SI), KBDC_ADD_H
                    RET
CHK_ADD :           ADD     CH, 04H
                    INR     SI
OUB_ENTRY :         MOV     (SI), CH
                    MOV     CX, 0000H
                    MOV     DS, CX
                    MOV     MARK_1, SI
                    POP     DS
                    MOV     CL, 02H
                    DCR     DI
LOAD_ADD :          INR     DI
                    MOV     AL, (BP+DI)
                    CMP     AL, '.'
                    JB      CONT_5
                    JMP     ERROR_1
CONT_5 :            SHL     AL                          , 4 TIMES
                    INR     DI
                    MOV     AH, (BP+DI)
                    CMP     AH, '.'
                    JB      CONT_6
                    JMP     ERROR_1
CONT_6 :            ADD     AL, AH
                    INR     SI
                    MOV     (SI), AL
                    LOOP    LOAD_ADD
MACRO               PORT_BYTE_SYNTAX_CHK
                    SUB     SI, 0002H
                    MOV     CX, (SI)
                    CMP     CX, DATA_PORT_A_PPI_1
                    JAE     NXT_1
                    JMP     ERROR_1
NXT_1 :             CMP     CX, DATA_PORT_C_PPI_2
                    JBE     NXT_2
                    JMP     ERROR_1
NXT_2 :             CMP     TOTAL_NO_PPI, 03H
                    JA      NXT_3
                    JMP     ERROR_1
NXT_3 :             CMP     CX, DATA_PORT_A_PPI_4
                    JAE     NXT_4
                    JMP     ERROR_1
```

```
NXT_4 :         CMP   CX, DATA_PORT_C_PPI_4
                JBE   NXT_5
                JMP   ERROR_1
NXT_6 :         ADD   SI, 0002H
                ENDM
                INR   DI
                MOV   AL, (BP+DI)
                CMP   AL, ','
                JZ    CONT_7
                CMP   AL, 'ENTER'
                JZ    CONT_8
                JMP   ERROR_1
CONT_8 :        RET
CONT_7 :        INR   DI
                MOV   AL, (BP+DI)
                CMP   AL, 'P'
                JA    CONT_9
                JMP   ERROR_1
CONT_9 :        CMP   AL, 'Z'
                JBE   CONT_10
                JMP   ERROR_1
CONT_10 :       CALL  VAR_RSLV_0
                TEST  CL, 04H
                JZ    CONT_11
                JMP   ERROR_1
CONT_11 :       INR   DI
                CMP   (BP+DI), 'ENTER'
                JZ    CONT_12
                JMP   ERROR_1
CONT_12 :       PUSH  DS
                PUSH  SI
                MOV   CL, 00H
                MOV   SI, MARK_1
                INR   (SI)
                POP   SI
                POP   DS
                RET
CHK_CH :        ADD   CH, 06H
                INR   SI
                MOV   (SI), CH
                PUSH  DS
                MOV   CX, 0000H
                MOV   DS, CX
                MOV   MARK_1, SI
                POP   DS
                INR   DI
                MOV   AL, (BP+DI)
                CMP   AL, 09H
```

```
                    JBE     CONT_13
                    JMP     ERROR_1
CONT_13 :           MOV     DX, BASE_ADD_ADC
                    SHL     AL
                    ADD     DX, AX
                    INR     SI
                    INCREMENT_SI
                    MOV     (SI), DX
                    INR     DI
                    MOV     AL, (BP+DI)
                    CMP     AL, ','
                    JZ      CONT_14
                    CMP     AL, 'ENTER'
                    JZ      CONT_15
                    JMP     ERROR_1
CONT_15 :           RET
CONT_14 :           INR     DI
                    MOV     AL, (BP+DI)
                    CMP     AL, 'Q'
                    JAE     CONT_16
                    JMP     ERROR_1
CONT_16 :           CMP     AL, 'Z'
                    JBE     CONT_17
                    JMP     ERROR_1
CONT_17 :           CALL    VAR_RSLV_0
                    CMP     CL, 04H
                    JZ      CONT_18
                    JMP     ERROR_1
CONT_18 :           INR     DI
                    CMP     (BP+DI), 'ENTER'
                    JZ      CONT_19
                    JMP     ERROR_1
CONT_19 :           PUSH    DS
                    PUSH    SI
                    MOV     CX, 0000H
                    MOV     DS, CX
                    MOV     SI, MARK_1
                    INR     (SI)
                    POP     SI
                    POP     DS
                    RET
CHK_VAR :           ADD     CH, 03H
                    INR     SI
                    MOV     (SI), CH
                    INR     DI
                    MOV     AL, (BP+DI)
                    CMP     AL, '.'
                    JA      CONT_20
```

```
                    JMP     ERROR_1
CONT_20 :           CMP     AL, 'Z'
                    JBE     CONT_21
                    JMP     ERROR_1
CONT_21 :           CALL    VAR_RSLV_0
                    CMP     (BP+DI), 'ENTER'
                    JZ      CONT_22
                    JMP     ERROR_1
CONT_22 :           RET
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_5 : LOAD_ICP_DLY

ANATOMY :

SYNTAX OF MAGNITUDE ENTRY OF DELAY AND THE SUBGROUP; INDICATING THE CLOCK RATE TO BE USED TO INCORPORATE THE DELAY ARE GENERATED AND LOADED IN IC FIELD.

REGISTER USAGE :

AL : KB ENTRIES SUCH AS DELAY PARAMETER NIBBLES.
BL,BH & AL,AH : PACKING BCD DELAY PARAMETER NIBBLES.
CL : SUBGROUP.
AX : PACKED DELAY PARAMETERS.

```
LABEL_DLY :     MOV   CX, 0003H
                MOV   BH, 00H
                INR   DI
                CMP   (BP+DI), 09H
                JBE   CONT_1
                JMP   ERROR_1
CONT_1 :        INR   DI
                MOV   AL, (BP+DI)
                CMP   AL, '.'
                JNZ   CONT_2
                INR   BH
                JMP   CONT_3
CONT_2 :        CMP   AL, 09H
                JBE   CONT_3
                JMP   ERROR_1
CONT_3 :        LOOP  CONT_1
                CMP   BH, 01H
                JZ    CONT_4
                JMP   ERROR_1
CONT_4 :        INR   DI
                CMP   (BP+DI), 09H
                JBE   CONT_5
                JMP   ERROR_1
CONT_5 :        INR   DI
                CMP   (BP+DI), 'ENTER'
                JZ    CONT_6
                JMP   ERROR_1
CONT_6 :        DCR   DI
                MOV   AL, (BP+DI)
                DCR   DI
```

```
                    MOV   BL, (BP+DI)
                    CMP   BL, '.'
                    JNZ   CONT_7
                    MOV   CL, SUB_GROUP_2
                    JMP   CONT_8
CONT_7 :            SHL   BL                          , 4 TIMES
                    ADD   AL, BL
                    DCR   DI
                    MOV   BL, (BP+DI)
                    CMP   BL, '.'
                    JNZ   CONT_9
                    MOV   CL, SUB_GROUP_1
                    JMP   CONT_10
CONT_9 :            MOV   AH, BL
                    MOV   CL, SUB_GROUP_0
                    SUB   DI, 0002H
                    MOV   BL, (BP+DI)
                    SHL   BL                    , 4 TIMES
                    ADD   AH, BL
                    JMP   LOAD_ICP
CONT_8 :            DCR   DI
                    MOV   BL, (BP+DI)
                    SHL   BL                    , 4 TIMES
                    ADD   AL, BL
                    DCR   DI
                    MOV   BL, (BP+DI)
                    MOV   AH, BL
                    DCR   DI
                    MOV   BL, (BP+DI)
                    SHL   BL                    , 4 TIMES
                    ADD   AH, BL
                    JMP   LOAD_ICP
CONT_10 :           DCR   DI
                    MOV   BL, (BP+DI)
                    DCR   DI
                    MOV   BL, (BP+DI)
                    SHL   BL                    , 4 TIMES
                    ADD   AH, BL
LOAD_ICP :          INR   SI
                    MOV   (SI), CL
                    INR   SI
                    MOV   (SI), AX
                    INR   DI
                    CMP   (BP+DI), 'ENTER'
                    JZ    CONT_11
                    JMP   ERROR_1
CONT_11 :           RET
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_6 : LOAD_ICP_OUW

ANATOMY :

EXCEPT FOR GROUP AND SUBGROUP ALLOCATION THE PART_8 AND PART_6 ARE DUPLICATES. THEREFORE THE PROGRAM JUMPS TO DUPLICATION IN PART_8 (LABEL_INW) AT OUW_ENTRY.

```
LABEL_OUW :        MOV    CH, AL
                   INR    SI
                   MOV    (SI), CH
                   JMP    OUW_ENTRY
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_7 : LOAD_ICP_INR.

ANATOMY :

THE VARIABLE ENTRY IS CONFIRMED TO BE AN INTEGER ENTRY APPROPRIATE ACTION GROUP AND VARIABLE FIELDS ARE LOADED IN IC FIELD.

REGISTER USAGE :

AL : KEY ENTRIES.

SUBROUTINES :

VAR_RSLV_0, L_3 (1,1).

```
LABEL_INR :     INR   SI
                MOV   (SI), BOH
DCR_ENTRY :     INR   DI
                MOV   AL, (BP+DI)
                CMP   AL, 'P'
                JA    CONT_1
                JMP   ERROR_1
CONT_1 :        CMP   AL, 'Z'
                JBE   CONT_2
                JMP   ERROR_1
CONT_2 :        CALL  VAR_RSLV_0
                TEST  CL, 01H
                JNZ   CHK_NXT
                JMP   ERROR_1
CHK_NXT :       INR   DI
                CMP   (BP+DI), 'ENTER'
                JZ    CONT_3
                JMP   ERROR_1
CONT_3 :        RET
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_8 : LOAD_ICP_INW

ANATOMY :

THIS PROGRAM IS MEANT TO LOAD IC FIELD, SOURCE AND DESTINATION PARAMETERS OF A WORD TO BE INPUTED. SYNTAX OF ALLOWED PORT ADDRESSES IS ALSO CONFIRMED.

REGISTER USAGE :

AL : KEY CODE ENTRIES.
AL & AH : PACKING ADDRESS NIBBLES.
CH : GROUP+SUBGROUP FIELDS OF IC.
CX : MANAGING SEGMENT.

REFERENCE MEMORY LOCATIONS :

MARK_1.

SUBROUTINES :

VAR_RSLV_0, L_3 (1,1).

```
LABEL_INW :     MOV   CH, AL
                INR   SI
                MOV   (SI), CH
                MOV   CX, 0000H
                PUSH  DS
                MOV   DS, CX
                MOV   MARK_1, SI
                POP   DS
                MOV   CL, 02H
LOAD_ADD :      INR   DI
                MOV   AL, (BP+DI)
                CMP   AL, '.'
                JBE   CONT_1
                JMP   ERROR_1
CONT_1 :        SHL   AL                    , 4 TIMES
                INR   DI
                MOV   AH, (BP+DI)
                CMP   AH, '.'
                JB    CONT_2
                JMP   ERROR_1
CONT_2 :        ADD   AL, AH
                INR   SI
                MOV   (SI), AL
                LOOP  LOAD_ADD
```

```
                        SUB     SI, 0002H
                        SHR     CX
                        JNC     CONT_3
                        JMP     ERROR_1
CONT_3 :                SHL     CX
                        CMP     CX, PORT_A_16_BIT
                        JBE     CONT_4
                        JMP     ERROR_1
CONT_4 :                CMP     CX, PORT_C_16_BIT
                        JBE     CONT_5
                        JMP     ERROR_1
CONT_5 :                ADD     SI, 0002H
                        INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, ','
                        JZ      CONT_6
                        CMP     AL, 'ENTER'
                        JZ      CONT_7
                        JMP     ERROR_1
CONT_7 :                RET
CONT_6 :                INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, 'P'
                        JA      CONT_8
                        JMP     ERROR_1
CONT_8 :                CMP     AL, 'Z'
                        JBE     CONT_9
                        JMP     ERROR_1
CONT_9 :                CALL    VAR_RSLV_0
                        JNZ     CONT_10
                        JMP     ERROR_1
CONT_10 :               PUSH    SI
                        PUSH    DS
                        MOV     CL, 00H
                        MOV     SI, MARK_1
                        POP     DS
                        INR     (SI)
                        POP     SI
                        RET
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_9 : LOAD_ICP_OUB.

ANATOMY :

THE PROGRAM STRUCTURE IS SIMILAR TO PART_8 AND PART_6 EXCEPT FOR SYNTAX CHECKS. THE FLOW JUMPS BACK TO ENTRY POINT IN PART_4 AT OUB ENTRY.

```
LABEL_OUB :     MOV    CH, AL
                INR    SI
                MOV    (SI), CH
                JMP    OUB_ENTRY
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_10 : LOAD_ICP_DSP.

ANATOMY :

THIS GROUP HAS TWO SUBGROUPS, (1) WHERE THE BACK CONVERTED NUMERIC ENTRIES AND MESSAGE IS TO BE DISPLAYED OR (2) ONLY MESSAGES ARE TO BE DISPLAYED.
MESSAGES &/NIL VARIABLE FIELDS ARE LOADED ON TO THE INTERPRETER CODE BLOCKS.

REGISTER USAGE :

AL : KB ENTRIES, PARAMETERS SUCH AS CHARACTERS OR VARIABLES.
CL : COUNTER.

```
LABEL_DSP :         INR   DI
                    CMP   (BP+DI), ','
                    JZ    CONT_1
                    JMP   DSP_GROUP_2
CONT_1 :            INR   SI
                    MOV   (SI), AL                ; SUBGROUP_1
                    PUSH  DS
                    MOV   CX, 0000H
                    MOV   DS, CX
                    MOV   MARK_1, SI
                    POP   DS
                    MOV   CX, 0000H
CHK_LOAD :          MOV   AL, (BP+DI)
                    CMP   AL, '.'
                    JA    CONT_2
                    JMP   ERROR_1
CONT_2 :            CMP   AL, 'Z'
                    JBE   CONT_3
                    JMP   ERROR_1
CONT_3 :            INR   DI
                    CMP   (BP+DI), ','
                    JZ    NXT_1
                    INR   SI
                    INCREMENT_SI
                    MOV   (SI), AL
                    INR   CL
                    INR   DI
                    JMP   CHK_LOAD
NXT_1 :             CMP   (BP+DI), 'ENTER'
```

```
                        JZ      CONT_4
                        JMP     ERROR_1
CONT_4 :                INR     SI
                        INCREMENT_SI
                        MOV     (SI), AL
                        INR     CL
                        CMP     CL, 08H
;ONLY 8 CHARACTERS ARE ALLOWED.
                        JBE     CONT_5
                        JMP     ERROR_1
CONT_5 :                PUSH    SI
                        PUSH    CX
                        PUSH    DS
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     SI, MARK_1
                        POP     DS
                        MOV     (SI), CL
;COUNT OF CHARACTERS LOADED.
                        POP     CX
                        POP     SI
                        RET
DSP_GROUP_2 :           INR     SI
                        INR     AL
                        MOV     (SI), AL                    SUBGROUP_2
                        INR     SI
                        PUSH    DS
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     MARK_1, SI
                        POP     DS
                        INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, '.'
                        JA      CONT_6
                        JMP     ERROR_1
CONT_6 :                CMP     AL, 'Z'
                        JBE     CONT_7
                        JMP     ERROR_1
CONT_7 :                INR     DI
                        MOV     AH, (BP+DI)
                        CMP     AH, '.'
                        JZ      CONT_8
                        CMP     AH, 'Z'
                        JBE     CONT_9
CONT_8 :                JMP     ERROR_1
CONT_9 :                INR     DI
                        CMP     (BP+DI), ','
```

```
                        JZ      CONT_10
                        JMP     ERROR_1
CONT_10 :               SUB     DI, 0002H
                        CALL    VAR_RSLV_0
                        MOV     CX, 0000H
                        INR     DI
DSP_CHK_LOAD_2 :        MOV     AL, (BP+DI)
                        CMP     AL, '.'
                        JA      CONT_11
                        JMP     ERROR_1
CONT_11 :               CMP     AL, 'Z'
                        JBE     CONT_12
                        JMP     ERROR_1
CONT_12 :               INR     DI
                        CMP     (BP+DI), ','
                        JNZ     NXT_2
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), AL
                        INR     CL
                        INR     DI
                        JMP     DSP_CHK_LOAD_2
NXT_2 :                 CMP     (BP+DI), 'ENTER'
                        JZ      CONT_13
                        JMP     ERROR_1
CONT_13 :               INR     CL
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), AL
                        CMP     CL, 06H
                        JBE     CONT_14
                        JMP     ERROR_1
CONT_14 :               PUSH    SI
                        PUSH    CX
                        PUSH    DS
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     SI, MARK_1
                        POP     DS
                        POP     CX
                        MOV     (SI), CL
                        POP     SI
                        INR     DI
                        CMP     (BP+DI), 'ENTER'
                        JZ      CONT_15
                        JMP     ERROR_1
CONT_15 :               RET
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_11 : LOAD_ICP_DCR.

ANATOMY :

THE ACTION GROUP AND SUBGROUP FOR DCR IS LOADED AND FLOW JUMPS BACK TO PART_7 AT DCR ENTRY.

```
LOAD_DCR :          INR    SI
                    MOV    (SI), AL
                    JMP    DCR_ENTRY
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_12 : LOAD_ICP_IF.

ANATOMY :

SYNTAX CHECK AS WELL AS SUBGROUP DETERMINATION CONTINUES WITHIN PROGRAM FRAGMENTS. THE APPARENT PARAMETERS FOLLOW THE CONVERSION OPERATOR IN IC FIELD. THEN ... LINE NUMBER ... PORTION OF IC FIELD IS LOADED WITH GROUP EQUIVALENT TO IC FIELD OF GTO ..LINE NO..

REGISTER USAGE :

AL : KB ENTRIES
AL, AH : PACKING ADDRESS NIBBLES.
CH : GROUP + SUBGROUP FIELD OF IC.
CX : MANAGING SEGMENT.

REFERENCE MEMORY LOCATIONS :

MARK_1.

SUBROUTINE :

VAR_RSLV_0, RSLV_CNST, L_3 (1,1).

```
LABEL_IF :      MOV   AH, 00H
                MOV   CH, AL
                INR   DI
                MOV   AL, (BP+DI)
                CMP   AL, '.'
                JB    CHK_ADD
                JA    CONT_1
                JMP   ERROR_1
CONT_1 :        CMP   AL, 'Z'
                JA    CONT_2
                JMP   CHK_VAR
CONT_2 :        CMP   AL, 'CH'
                JNZ   CONT_3
                JMP   CHK_CH
CONT_3 :        CMP   AL, 'KB'
                JZ    CHK_KB
                JMP   ERROR_1
CHK_KB :        INR   SI
                ADD   CH, 04H
                MOV   (SI), CH
                INR   SI
```

```
                        MOV     (SI), KBDC_L
                        INR     SI
                        MOV     (SI), KBDC_H
                        INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, '='
                        JAE     CONT_4
                        JMP     ERROR_1
CONT_4 :                CMP     AL, '>='
                        JBE     CONT_5
                        JMP     ERROR_1
CONT_5 :                PUSH    AX
;SAVE THE OPERATOR ON STACK.
                        INR     DI
                        MOV     AH, (BP+DI)
                        CMP     AH, '.'
                        JB      CONT_6
                        JMP     ERROR_1
CONT_6 :                INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, '.'
                        JB      CONT_7
                        JMP     ERROR_1
CONT_7 :                SHL     AH                          , 4 TIMES
                        ADD     AL, AH
                        INR     SI
                        MOV     (SI), AL
                        POP     AX
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), AL
                        JMP     LOAD_L_NO
CHK_ADD :               INR     SI
                        MOV     (SI), CH
                        MOV     CX, 0000H
                        PUSH    DS
                        MOV     DS. CX
                        MOV     MARK_1, SI
                        POP     DS
                        MOV     CL, 02H
                        DCR     DI
LOAD_ADD :              INR     DI
                        MOV     AL, (BP+DI)
                        CMP     AL, '.'
                        JB      CONT_8
                        JMP     ERROR_1
CONT_8 :                SHL     AL                          , 4 TIMES
                        INR     DI
```

```
                    MOV     AL, (BP+DI)
                    CMP     AH, '.'
                    JB      CONT_9
                    JMP     ERROR_1
CONT_9 :            ADD     AL, AH
                    INR     SI
                    MOV     (SI), AL
                    LOOP    LOAD_ADD
                    PORT_BYTE_SYNTAX_CHK
CHN_ENTRY :         INR     DI
                    MOV     AL, (BP+DI)
                    CMP     AL, '='
                    JAE     CONT_10
                    JMP     ERROR_1
CONT_10 :           CMP     AL, '>='
                    JBE     CONT_11
                    JMP     ERROR_1
CONT_11 :           PUSH    AX
                    INR     DI
                    MOV     AL, (BP+DI)
                    CMP     AL, '.'
                    JB      CHK_CONST
                    JA      CHK_VAR_1
                    JMP     ERROR_1
CHK_CONST :         INR     DI
                    MOV     AH, (BP+DI)
                    CMP     AH, '.'
                    JB      CONT_12
                    JMP     ERROR_1
CONT_12 :           SHL     AL
                    ADD     AL, AH
                    INR     SI
                    INCREMENT_SI
                    MOV     (SI), AL
                    POP     AX
                    INR     SI
                    INCREMENT_SI
                    MOV     (SI), AL
                    JMP     LOAD_L_NO
CHK_VAR_1 :         CMP     AL, 'Q'
                    JAE     CONT_13
                    JMP     ERROR_1
CONT_13 :           CMP     AL, 'Z'
                    JBE     CONT_14
                    JMP     ERROR_1
CONT_14 :           CALL    VAR_RSLV_0
                    TEST    CL, 04H
                    POP     AX
```

```
                    INR    SI
                    INCREMENT_SI
                    MOV    (SI), AL
                    JZ     CONT_15
                    JMP    ERROR_1
CONT_15 :           PUSH   SI
                    PUSH   DS
                    MOV    CX, 0000H
                    MOV    DS, CX
                    MOV    SI, MARK_1
                    POP    DS
                    INR    (SI)
                    POP    SI
                    JMP    LOAD_L_NO
CHK_CH :            ADD    CH, 02H
                    INR    SI
                    MOV    (SI), CH
                    PUSH   DS
                    MOV    CX, 0000H
                    MOV    DS. CX
                    MOV    MARK_1, SI
                    POP    DS
                    INR    DI
                    MOV    AL, (BP+DI)
                    CMP    AL, 09H
                    JBE    CONT_16
                    JMP    ERROR_1
CONT_16 :           MOV    DX, BASE_ADD_ADC
                    SHL    AL
                    ADD    DX, AX
                    INR    SI
                    MOV    (SI), DX
                    INR    SI
                    JMP    CHN_ENTRY
CHK_VAR :           ADD    CH, 05H
                    INR    SI
                    MOV    (SI), CH
                    MOV    CX, 0000H
                    PUSH   DS
                    MOV    DS, CX
                    MOV    MARK_1, SI
                    POP    DS
                    CALL   VAR_RSLV_0
                    MOV    CH, CL
                    INR    DI
                    MOV    AL, (BP+DI)
                    CMP    AL, '='
                    JAE    CONT_17
```

```
                    JMP     ERROR_1
CONT_17 :           CMP     AL, '>='
                    JB      CONT_18
                    JMP     ERROR_1
CONT_18 :           PUSH    AX
                    INR     DI
                    MOV     AL, (BP+DI)
                    CMP     AL, '.'
                    JB      CHK_CONST_1
                    JA      CHK_VAR_2
                    JMP     ERROR_1
CHK_CONST_1 :       CALL    RSLV_CNST
                    AND     CX, 0707H
                    SUB     CL, 02H
                    CMP     CH, CL
                    JZ      CONT_1A
                    JMP     ERROR_1
CONT_1A :           POP     AX
                    INR     SI
                    INCREMENT_SI
                    MOV     (SI), AL
                    JMP     LOAD_L_NO
CHK_VAR_2 :         CMP     AL, 'Z'
                    JBE     CONT_1B
                    JMP     ERROR_1
CONT_1B :           CALL    VAR_RSLV_0
                    XOR     CH, CL
                    TEST    CH, 07H
                    JZ      CONT_1C
                    JMP     ERROR_1
CONT_1C :           POP     AX
                    INR     SI
                    INCREMENT_SI
                    MOV     (SI), AL
                    PUSH    SI
                    PUSH    DS
                    MOV     CX, 0000H
                    MOV     DX, 0000H
                    MOV     SI, MARK_1
                    POP     DS
                    INR     (SI)
                    POP     SI
LOAD_L_NO :         INR     SI
                    INCREMENT_SI
                    MOV     (SI), GTO_ACT_GROUP
                    PUSH    DS
                    MOV     DS, CX
                    MOV     MARK_1, SI
```

```
                    POP    DS
                    INR    DI
                    CMP    (BP+DI), 'THEN'
                    JZ     GTO_ENTRY
                    JMP    ERROR_1
GTO_ENTRY :         INR    DI
                    MOV    AL, (BP+DI)
                    CMP    AL, 09H
                    JBE    CONT_1D
                    JMP    ERROR_1
CONT_1D :           INR    DI
                    MOV    AH, (BP+DI)
                    CMP    AH, 09H
                    JBE    CONT_1E
                    JMP    ERROR_1
CONT_1E :           SHL    AL                          , 4 TIMES
                    ADD    AL, AH
                    INR    SI
                    INCREMENT_SI
                    MOV    (SI), AL
                    MOV    DH, AL
                    INR    DI
                    MOV    AL, (BP+DI)
                    CMP    AL, 09H
                    JBE    CONT_1F
                    JMP    ERROR_1
CONT_1F :           INR    DI
                    MOV    AH, (BP+DI)
                    CMP    AH, 09H
                    JBE    CONT_20
                    JMP    ERROR_1
CONT_20 :           SHL    AL                          , 4 TIMES
                    ADD    AL, AH
                    INR    SI
                    INCREMENT_SI
                    MOV    (SI), AH
                    MOV    DL, AH
                    PUSH   SI
                    AND    SI, FFF8H
                    CMP    (SI), FFH
                    JNZ    CONT_21
                    SUB    SI, 0008H
CONT_21 :           INR    SI
                    CMP    DX, (SI)
                    JB     RSLV_ENTER
                    JA     CONT_22
                    JMP    ERROR_1
```

```
CONT_22 :               PUSH   DS
                        MOV    CX, 0000H
                        MOV    DS, CX
                        MOV    SI, MARK_1
                        POP    DS
                        INR    (SI)
RSLV_ENTER :            POP    SI
                        INR    SI
                        CMP    (BP+DI), 'ENTER'
                        JZ     CONT_23
                        JMP    ERROR_1
CONT_23 :               RET
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_13 : LOAD_ICP_GSB.

ANATOMY :

ACTION GROUP AND LINE NUMBER CONTRIBUTE TO THE IC FIELD. SYNTAX CHECK ALLOWS ONLY THOSE LINE NUMBER GREATER THAN THE CURRENT ONE.

REGISTER USAGE :

AL, AH : PACKING LINE NUMBER NIBBLES.
CL : COUNTER.
CX : MANAGING SEGMENT.

```
LABEL_GSB :     INR   SI
                MOV   (SI), AL
                MOV   CH, 00H
                MOV   CL, 02H
LOAD_LNO :      INR   DI
                MOV   AL, (BP+DI)
                CMP   AL, 09H
                JBE   CONT_1
                JMP   ERROR_1
CONT_1 :        SHL   AL                    , 4 TIMES
                INR   DI
                MOV   AH, (BP+DI)
                CMP   AH, 09H
                JBE   CONT_2
                JMP   ERROR_1
CONT_2 :        ADD   AL, AH
                INR   SI
                MOV   (SI), AL
                JMP   LOAD_LNO
                SUB   SI, 0002H
                MOV   DX, (SI)
                SUB   SI, 0003H
                CMP   DX, (SI)
                JA    CONT_3
                JMP   ERROR_1
CONT_3 :        INR   DI
                CMP   (BP+DI), 'ENTER'
                JZ    CONT_4
                JMP   ERROR_1
CONT_4          RET
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_14 : LOAD_ICP_FOR.
&
PART_16 : LOAD_ICP_NXT.

ANATOMY :

ALONG WITH LOADING THE PARAMETERS OF ITERATIONS IN IC FIELD THE SYNTAX OF ALLOWED NESTING IS CONFIRMED.

REGISTER USAGE :

AL : KB ENTRIES.
CL : VARIABLE IDENTIFIER.
CX : MANAGING SEGMENT.

REFERENCE MEMORY LOCATION :

MARK_1.

PART_14 :

```
LABEL_FOR  :        MOV   CH, AL
                    INR   SI
                    MOV   (SI), CH
                    INR   DI
                    MOV   AL, (BP+DI)
;ONLY  8I(INTEGER)  DATA  TYPE VARIABLES ALLOWED  WITH  INDEX   OO
;WHICH IS IMPLICIT.
                    CMP   AL, 'P'
                    JA    CONT_1
                    JMP   ERROR_1
CONT_1 :            CMP   AL, 'Z'
                    JBE   CONT_2
                    JMP   ERROR_1
CONT_2 :            CALL  VAR_RSLV_0
                    TEST  CL, 04H
                    JZ    CONT_3
                    JMP   ERROR_1
CONT_3 :            XOR   CL, 01000000B
                    TEST  CL, E0H
                    JZ    CONT_4
                    JMP   ERROR_1
CONT_4 :            MOV   AH, AL
                    PUSH  AX
```

```
                    INR     DI
                    MOV     AH, (BP+DI)
                    CMP     AH, '.'
                    JB      CONT_5
                    JMP     ERROR_1
CONT_5 :            INR     DI
                    MOV     AL, (BP+DI)
                    CMP     AL, '.'
                    JB      CONT_6
                    JMP     ERROR_1
CONT_6 :            CMP     AH, 00H
                    JZ      CHK_NXT
                    SHL     AH                          , 4 TIMES
CHK_NXT :           ADD     AH, AL
                    MOV     CL, AH
                    INR     SI
                    MOV     (SI), AH
                    INR     DI
                    CMP     (BP+DI), ','
                    JZ      CONT_7
                    JMP     ERROR_1
CONT_7 :            INR     DI
                    CMP     AL, (BP+DI)
                    CMP     AL, '.'
                    JB      CONT_8
                    JMP     ERROR_1
CONT_8 :            INR     DI
                    MOV     AH, (BP+DI)
                    CMP     AH, '.'
                    JB      CONT_9
                    JMP     ERROR_1
CONT_9 :            SHL     AL                          , 4 TIMES
                    ADD     AL, AH
                    CMP     AL, CL
                    JA      CONT_A
                    JMP     ERROR_1
CONT_A :            INR     SI
                    MOV     (SI), AL
                    INR     DI
                    CMP     (BP+DI), 'ENTER'
                    JZ      CONT_B
                    JMP     ERROR_1
CONT_B :            RET
```

PART_16 :

```
LABEL_NXT :         MOV   CH, AL
                    MOV   (SI), CH
                    INR   DI
                    MOV   AL, (BP+DI)
                    POP   CX
                    CMP   AL, CL
                    JZ    CONT_1
                    JMP   ERROR_1
CONT_1 :            INR   SI
                    MOV   (SI), CH
                    INR   SI
                    MOV   (SI), AL
                    INR   DI
                    CMP   (BP+DI), 'ENTER'
                    JZ    CONT_2
                    JMP   ERROR_1
CONT_2 :            CMP   SP, BASE_ADD_STACK
                    JBE   CONT_3
                    JMP   ERROR_1
CONT_3 :            RET
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

PART_15 : LOAD_ICP_GTO.

ANATOMY :

FORWARD AND BACKWARD JUMPS FORM TO DISTINCT SUBGROUPS OF THE ACTION. THE IC FIELD IS LOADED WITH ACTION GROUP, SUBGROUP AND THE LINE NUMBER AFTER JUMPING TO PART_12 AT GTO_ENTRY.

```
LABEL_GTO :     MOV   CH, AL
                INR   SI
                MOV   (SI), CH
                MOV   CX, 0000H
                PUSH  DS
                MOV   DS, CX
                MOV   MARK_1, SI
                POP   DS
                JMP   GTO_ENTRY
```

TITLE : MODULE : SUB, RSLV_KB_ENTRIES, LEVEL_3(1)

SUB_PART_1 : VAR_RSLV_0, LEVEL_3 (1,1), F.C. 4.1.

ANATOMY :

RESOLVES THE TYPE OF VARIABLE AND ASSOCIATES IDENTIFIER. IDENTIFIER FORMED, VARIABLE AND INDEX ARE LOADED IN INTERPRETER CODE POINTER FIELD. THE NUMBERS ARE CONVERTED INTO THE INTERNAL FORMS OF REPRESENTATION AND ARE LOADED IN IC FIELD LEAVING ASSOCIATED IDENTIFIER FOR FURTHER SYNTAX CONFIRMATION. THE IC FIELD POINTER POINTS TO A LOCATION BEFORE NEXT IC FIELD BYTE TO BE LOADED. WHILE THE KB POINTER ALSO FOLLOWS THE SAME STRATEGY.

REGISTER USAGE :

AL, AH : TO RESOLVE KB ENTRIES.
CL : TO FORM IDENTIFIER.
BX : BASE ADDRESS OF INTERPRETER CODE POINTER.
BP, DI : POINTS TO KBB OR DIT.
BL, BH : MANAGE DIT POINTER.

SUBROUTINE :

VAR_RSLV_1, VAR_RSLV_2 L_3 (1,1,1).

```
VAR_RSLV_0 :    INR   DI
                MOV   AL, (BP+DI)
                CMP   AL, 'ENTER'
                JAE   CHK_NXT_1
                CMP   AL, '.'
                JA    CHK_NXT_2
                JB    CHK_NXT_0
                JMP   ERROR_1
CHK_NXT_0 :     CALL  VAR_RSLV_1
                INR   SI
                INCREMENT_SI
                MOV   (SI), CL
                INR   SI
                INCREMENT_SI
                MOV   (SI), AL
                INR   DI
                MOV   AL, (BP+DI)
                SHL   AH                    , 4 TIMES
                ADD   AL, AH
                TEST  CL, 1CH
```

```
;IDENTIFIER CHECKED FOR DEFAULT TYPE VARIABLE.
                        JNZ     CHK_NORM_EXT
                        CMP     AL, 05H
;ONLY 0 TO 5 THAT IS 6 INDICES ARE ALLOWED WITH DEFAULT VARIABLE.
                        JBE     LOAD_INDEX
                        JMP     ERROR_1
CHK_NORM_EXT :          TEST    CL, 10H
                        JNZ     LOAD_INDEX
                        TEST    CL, 08H
                        JNZ     NXT_1
                        JMP     ERROR_1
NXT_1 :                 CMP     AL, 10H
;NORMAL TYPE VARIABLE ALLOWS ONLY 16 INDICES FROM 0 TO F.
                        JB      LOAD_INDEX
                        JMP     ERROR_1
LOAD_INDEX :            INR     SI
                        INCREMENT_SI
                        MOV     (SI), AL
                        RET
CHK_NXT_1 :             MOV       CL, 40H
;40H IS ATTACHED TO IDENTIFIER TO INDICATE THAT VARIABLE IS  WITH
;NO INDEX.
                        CALL    VAR_RSLV_1
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), CL
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), AL
                        RET
CHK_NXT_2 :             CALL    VAR_RSLV_2
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), CL
                        DCR     DI
                        MOV     AL, (BP+DI)
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), AL
                        INR     DI
                        MOV     AL, (BP+DI)
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), AL
                        RET
```

SUB_PART_1, 1 : VAR_RSLV_1 LEVEL_3 (1,1,1)

```
;THE  SUBROUTINE  DETERMINES THE ATTACHMENT  IDENTIFIER  FOR  THE
;VARIABLE, AS INDICATED IN TABLE       , SECTION       AND USES
;DATA AREA.
VAR_RSLV_1 :        DCR   DI
                    MOV   AL, (BP+DI)
                    PUSH  BP
                    PUSH  DI
;SAVE KBB POINTERS.
                    MOV   DI, 0000H
                    MOV   BP, BASE_ADD_DIT
;POINTER LOADED WITH BASE OF DATA AREA INDEX TABLE AND DI  POINTS
;TO FIRST LOCATION IN DIT.
                    CMP   AL, 'P'
                    JBE   RSLV_REAL
                    ADD   CL, 01H
                    ADD   DI, 0003H
                    CMP   AL, (BP+DI)
                    JA    NXT_V1
                    ADD   CL, 00H
                    JMP   BACK
NXT_V1 :            INR   DI
                    MOV   BH, 00H
                    MOV   BL, (BP+DI)
                    ADD   DI, BX
                    CMP   AL, (BP+DI)
                    JA    NXT_V2
                    ADD   CL, 04H
                    JMP   BACK
NXT_V2 :            INR   DI
                    CMP   AL, (BP+DI)
                    JAE   NXT_V21
                    JMP   ERROR_1
NXT_V21 :           INR   DI
                    MOV   BL, (BP+DI)
                    ADD   DI, BX
                    CMP   AL, (BP+DI)
                    JA    NXT_V3
                    ADD   CL, 08H
                    JMP   BACK
NXT_V3 :            INR   DI
                    MOV   BL, (BP+DI)
                    ADD   DI, BX
                    CMP   AL, (BP+DI)
                    JA    NXT_V4
                    ADD   CL, 0CH
                    JMP   BACK
```

```
NXT_V4 :        INR   DI
                CMP   AL, (BP+DI)
                JAE   NXT_V41
                JMP   ERROR_1
NXT_V41 :       INR   DI
                MOV   BL, (BP+DI)
                ADD   DI, BX
                CMP   AL, (BP+DI)
                JA    NXT_V5
                ADD   CL, 10H
                JMP   BACK
NXT_V5 :        INR   DI
                MOV   BL, (BP+DI)
                ADD   DI, BX
                CMP   AL, (BP+DI)
                JBE   NXT_V51
                JMP   ERROR_1
NXT_V51 :       ADD   CL, 14H
                JMP   BACK
RSLV_REAL :     ADD   CL, 00H
                ADD   DI, 0002H
                MOV   BL, (BP+DI)
                ADD   DI, BX
                CMP   AL, (BP+DI)
                JA    NXT_V6
                ADD   CL, 00H
NXT_V6 :        INR   DI
                MOV   BL, (BP+DI)
                ADD   DI, BX
                CMP   AL, (BP+DI)
                JA    NXT_V7
                ADD   CL, 04H
                JMP   BACK
NXT_V7 :        INR   DI
                CMP   AL, (BP+DI)
                JAE   NXT_V71
                JMP   ERROR_1
NXT_V71 :       INR   DI
                MOV   BL, (BP+DI)
                ADD   DI, BX
                CMP   AL, (BP+DI)
                JA    NXT_V8
                ADD   CL, 08H
                JMP   BACK
NXT_V8 :        INR   DI
                MOV   BL, (BP+DI)
                ADD   DI, BX
                CMP   AL, (BP+DI)
```

```
                    JA      NXT_V9
                    ADD     CL, 0CH
                    JMP     BACK
NXT_V9 :            INR     DI
                    CMP     AL, (BP+DI)
                    JAE     NXT_V91
                    JMP     ERROR_1
NXT_V91 :           INR     DI
                    MOV     BL, (BP+DI)
                    ADD     DI, BX
                    CMP     AL, (BP+DI)
                    JA      NXT_VA
                    ADD     CL, 10H
                    JMP     BACK
NXT_VA :            INR     DI
                    MOV     BL, (BP+DI)
                    ADD     DI, BX
                    CMP     AL, (BP+DI)
                    JBE     NXT_VA1
                    JMP     ERROR_1
NXT_VA1 :           ADD     CL, 14H
BACK :              POP     DI
                    POP     BP
                    RET
```

SUB_PART_1, 1 : VAR_RSLV_2 LEVEL_3 (1,1,1)

```
;THE SUBROUTINE DETERMINES THE ATTACHMENT IDENTIFIER FOR THE
;VARIABLE WITH A VARIABLE AS INDEX.

VAR_RSLV_2 :     CALL  VAR_RSLV_1
                 INR   DI
                 MOV   AL, (BP+DI)
                 CMP   AL, 'Q'
                 JAE   NXT_R1
                 JMP   ERROR_1
NXT_R1 :         CMP   AL, 'Z'
                 JBE   NXT_R11
                 JMP   ERROR_1
NXT_R11 :        PUSH  BP
                 PUSH  DI
                 MOV   DI, 0000H
                 MOV   BP, BASE_ADD_DIT
                 ADD   DI, 0003H
                 CMP   AL, (BP+DI)
                 JA    NXT_R2
                 ADD   CL, 60H
                 JMP   BACK_1
NXT_R2 :         INR   DI
                 MOV   BL, (BP+DI)
                 ADD   DI, BX
                 INR   DI
                 CMP   AL, (BP+DI)
                 JBE   NXT_R21
                 JMP   ERROR_1
NXT_R21 :        INR   DI
                 MOV   BL, (BP+DI)
                 ADD   DI, BX
                 CMP   AL, (BP+DI)
                 JA    NXT_R3
                 ADD   CL, A0H
                 JMP   BACK_1
NXT_R3 :         INR   DI
                 MOV   BL, (BP+DI)
                 ADD   DI, BX
                 INR   DI
                 CMP   AL, (BP+DI)
                 JAE   NXT_R31
                 JMP   ERROR_1
NXT_R31 :        INR   DI
                 MOV   BL, (BP+DI)
                 ADD   DI, BX
```

```
                        CMP     AL, (BP+DI)
                        JBE     NXT_R32
                        JMP     ERROR_1
NXT_R32 :               ADD     CL, E0H
BACK_1 :                POP     DI
                        POP     BP
                        RET
```

SUB_PART_2 : RSLV_CNST LEVEL_3 (1,1).

ANATOMY :

THE NUMERIC VALUES ARE BROUGHT FROM KBB AND ARE RESOLVED FOR DIFFERENT DATA TYPES REPRESENTED, SUCH AS 16 INTEGER, 8 INTEGER, 16 REAL AND 24 REAL. THE NUMBERS ARE CONVERTED INTO THE INTERNAL FORMS OF REPRESENTATION AND ARE LOADED IN IC FIELD LEAVING ASSOCIATED IDENTIFIER FOR FURTHER SYNTAX CONFIGURATION.

PART_1 : CHK_8I :

8 INTEGER NUMERIC ENTRIES ARE DIFFERENTIATED AT THE LEVEL OF TOTAL NUMBER OF ENTRIES IN KBB.

REGISTER USAGE :

AL, AH : FORMING NUMERIC BYTE.
CL : IDENTIFIER.

PART_2 : CHK_16I :

16 INTEGER NUMERIC ENTRIES ARE DIFFERENTIATED AT THE LEVEL OF TOTAL NUMBER OF ENTRIES IN KBB.

REGISTER_USAGE :

AL, AH, BL : FORMING INTEGER WORD.
CL : IDENTIFIER.

PART_3 : CHK_24R :

THE BCD ENTRIES ARE CONVERTED INTO BINARY FORM. THE 2^5 EXPONENT IS DETERMINED. THE NUMBER THUS GENERATED OCCUPIES REGISTERS DL AND AX. THIS PROCESS NEEDS EXHAUSTIVE USE OF REGISTERS AND MEMORY LOCATIONS WHILE EXECUTING 16 BIT OR 8 BIT MULTIPLICATION OR DIVISION.

PART_4 : CHK_16R :

PROCEDURE SIMILAR TO PART_3.

REFERENCE MEMORY LOCATIONS :

MARK_2, MUL_BYTE_1, MUL_WORD_0, MUL_WORD_1, MUL_WORD_2, MUL_WORD_3, MUL_WORD_4, MUL_WORD_5, MUL_WORD_6, MUL_WORD_7, MUL_WORD_11, MUL_WORD_12, MUL_WORD_13.

```
RSLV_CNST :         ADD   DI, 0002H
                    CMP   (BP+DI), 'ENTER'
                    JB    NXT_1
                    JMP   CHK_8I
NXT_1 :             ADD   DI, 0002H
                    CMP   (BP+DI), 'ENTER'
                    JB    NXT_2
                    JMP   CHK_16I
NXT_2 :             INR   DI
                    CMP   (BP+DI), 'E'
                    JB    NXT_3
                    JMP   CHK_16R
                    ADD   DI, 0002H
                    CMP   (BP+DI), 'ENTER'
                    JAE   NXT_11
                    JMP   ERROR_1
NXT_11 :            JMP   CHK_24

PART_1 :

CHK_8I :            CMP   (BP+DI), '='
                    JB    NXT_12
                    JMP   ERROR_1
NXT_12 :            DCR   DI
                    MOV   AL, (BP+DI)
                    DCR   DI
                    MOV   AH, (BP+DI)
                    CMP   AH, 07H
                    JBE   NXT_13
                    JMP   ERROR_1
NXT_13 :            SHL   AH                    , 4 TIMES
                    ADD   AL, AH
                    MOV   CL, 9AH
                    INR   SI
                    INCREMENT_SI
                    MOV   (SI), CL
                    INR   SI
                    INCREMENT_SI
                    MOV   (SI), AL
                    RET
```

```
PART_2 :

CHK_16I :       CMP     (BP+DI), '='
                JB      NXT_14
                JMP     ERROR_1
NXT_14 :        DCR     DI
                MOV     AL, (BP+DI)
                DCR     DI
                MOV     BL, (BP+DI)
                SHL     BL                          , 4 TIMES
                ADD     AL, BL
                DCR     DI
                MOV     AH, (BP+DI)
                DCR     DI
                MOV     BL, (BP+DI)
                CMP     BL, 07H
                JBE     NXT_15
                JMP     ERROR_1
NXT_15 :        SHL     BL                          , 4 TIMES
                ADD     AH, BL
                MOV     CL, 9FH
                INR     SI
                INCREMENT_SI
                MOV     (SI), CL
                INR     SI
                INCREMENT_SI
                MOV     (SI), AH
                INR     SI
                INCREMENT_SI
                MOV     (SI), AL
                RET

PART_3 :

CHK_24R :       MOV     DX, 0007H
                MOV     BH, 00H
                MOV     CX, DX
;TO CHECK WHETHER ALL KB ENTRIES ARE NOT EQUAL TO 0 AND ONLY  ONE
;DOT IS ENTERED.
CHK_AGAIN :     DCR     DI
                MOV     AL, (BP+DI)
                CMP     AL, '.'
                JB      NXT_1
                JE      NXT_16
                JMP     CONT_1
```

```
NXT_16 :                INR    BH
NXT_1 :                 CMP    AL, 0AH
                        JB     NXT_17
                        JMP    ERROR_1
NXT_17 :                CMP    AL, 00H
                        JNZ    NXT_2
CONT_1 :                INR    DH
NXT_2 :                 LOOP   CHK_AGAIN
                        CMP    BH, 01H
                        JZ     NXT_18
                        JMP    ERROR_1
NXT_18 :                CMP    DH, 06H
                        JNZ    NXT_19
                        JMP    ERROR_1
NXT_19 :                PUSH   DS
                        MOV    DS, CX
                        MOV    DH, 00H
                        ADD    DI, DX
                        MOV    BL, 00H
                        CMP    (BP+DI), 'E'
                        JNZ    CHK_NXT_1
                        INR    DI
                        CMP    (BP+DI), '-'
                        JNZ    CHK_NXT_2
                        MOV    MARK_2, FFH
;A  MEMORY  LOCATION WHICH INDICATES UNARY SIGN  ASSOCIATED  WITH
;EXPONENT.
                        INR    DI
CHK_NXT_2 :             CMP    (BP+DI), '.'
                        JB     NXT_1A
                        JMP    ERROR_1
NXT_1A :                MOV    AL, (BP+DI)
                        INR    DI
                        CMP    (BP+DI), '.'
                        JB     NXT_1B
                        JMP    ERROR_1
NXT_1B :                MOV    AH, (BP+DI)
                        MUL    AL, MUL_BYTE_1
                        ADD    AL, AH
                        MOV    BL, AL
;EXPONENT OF 24R DATA TYPE FORMED IN REGISTER BL.
                        CMP    MARK_2, FFH
                        JNZ    CONT_2
                        NEG    BL                ;NEGATIVE EXPONENT
CONT_2 :                INR    DI
                        CMP    (BP+DI), 'ENTER'
                        JAE    CONT_3
                        JMP    ERROR_1
```

```
CONT_3 :            SUB   DI, 0002H
                    CMP   (BP+DI), 'E'
                    JZ    CONT_4
                    DCR   DI
CHK_NXT_1 :         CMP   (BP+DI), 'ENTER'
                    JAE   CONT_4
                    JMP   ERROR_1
CONT_4 :            DCR   DI
                    MOV   AX, 0000H
                    MOV   DX, 0000H
                    MOV   CX, 0000H
                    MOV   BH, 00H
                    MOV   AL, (BP+DI)
                    CMP   AL, 0AH
                    JB    CHK_NXT_4
                    DCR   DI
                    MOV   AL, (BP+DI)
CHK_NXT_4 :         ADD   CL, AL
                    DCR   DI
                    CMP   (BP+DI), '.'
                    JB    CHK_NXT_5
                    DCR   BL
                    DCR   DI
CHK_NXT_5 :         MOV   AL, (BP+DI)
                    CMP   AL, 00H
                    JZ    CHK_NXT_6
                    MUL   AL, MUL_BYTE_1
                    ADD   CL, AL
CHK_NXT_6 :         DCR   DI
                    CMP   (BP+DI). '.'
                    JB    NXT_7
                    SUB   BL, 02H
                    DCR   DI
                    MOV   AL, (BP+DI)
CHK_NXT_7 :         CMP   AL, 00H
                    JZ    CHK_NXT_8
                    MUL   AX, MUL_WORD_1
                    ADD   CX, AX
                    MOV   AX, 0000H
CHK_NXT_8 :         DCR   DI
                    CMP   (BP+DI), '.'
                    JB    CHK_NXT_9
                    SUB   BL, 03H
                    DCR   DI
CHK_NXT_9 :         MOV   AL, (BP+DI)
                    CMP   AL, 00H
                    JZ    CHK_NXT_A
                    MUL   AX, MUL_WORD_2
```

```
                        ADD     CX, AX
                        SUB     AX, AX
CHK_NXT_A :             DCR     DI
                        CMP     (BP+DI), '.'
                        JB      CHK_NXT_B
                        SUB     BL, 04H
                        DCR     DI
CHK_NXT_B :             MOV     AL, (BP+DI)
                        DCR     DI
                        CMP     (BP+DI), '.'
                        JB      CHK_NXT_C
                        SUB     BL, 05H
                        DCR     DI
CHK_NXT_C :             MOV     AL, (BP+DI)
                        CMP     AL, 00H
                        JZ      CHK_NXT_D
                        MUL     AL, MUL_BYTE_1
                        ADD     AL, AH
CHK_NXT_D :             MUL     AX, MUL_WORD_3
                        CLC
                        ADD     AX, CX
                        MOV     CX, 0000H
                        ADC     CL, DL
                        MOV     DL, 00H
                        DCR     DI
                        CMP     (BP+DI), '.'
                        JNZ     FORM_24R
                        SUB     BL, 06H
FORM_24R :              CMP     BL, 80H
                        JB      POS_EXP_24R
                        JMP     NEG_EXP_24R
POS_EXP_24R :           CMP     BL, 04H
                        JA      POS_EXP_5
                        JB      POS_EXP_3
                        MUL     AX, MUL_WORD_3
                        MOV     MUL_WORD_11, AX
                        MOV     MUL_WORD_12, DX
                        MOV     AX, CX
                        MUL     AX, MUL_WORD_3
                        ADD     AX, MUL_WORD_12
                        MOV     DX, AX
                        MOV     AX, MUL_WORD_11
                        JMP     FORM_EXP
POS_EXP_3 :             CMP     BL, 03H
                        JNZ     POS_EXP_2
                        MUL     AX, MUL_WORD_2
                        MOV     MUL_WORD_11, AX
                        MOV     MUL_WORD_12, DX
```


```
                        MOV     AX, CX
                        MUL     AX, MUL_WORD_2
                        ADD     AX, MUL_WORD_12
                        MOV     DX, AX
                        MOV     AX, MUL_WORD_11
                        JMP     FORM_EXP
POS_EXP_2 :             CMP     BL, 02H
                        JB      POS_EXP_1
                        MUL     AX, MUL_WORD_1
                        MOV     MUL_WORD_11, AX
                        MOV     MUL_WORD_12, DX
                        MOV     AX, CX
                        MUL     AX, MUL_WORD_2
                        ADD     AX, MUL_WORD_12
                        MOV     DX, AX
                        MOV     AX, MUL_WORD_11
                        JMP     FORM_EXP
POS_EXP_1 :             MUL     AX, MUL_WORD_0
                        MOV     MUL_WORD_11, AX
                        MOV     MUL_WORD_12, DX
                        MOV     AX, CX
                        MUL     AX, MUL_WORD_0
                        ADD     AX, MUL_WORD_12
                        MOV     DX, AX
                        MOV     AX, MUL_WORD_11
                        JMP     FORM_EXP
POS_EXP_5 :             CMP     CL, 00H
                        JZ      NXT_20
                        JMP     ERROR_1
NXT_20 :                CMP     BL, 05H
                        JNZ     POS_EXP_6
                        CMP     AX, 4E1FH
                        JBE     NXT_21
                        JMP     ERROR_1
NXT_21 :                MUL     AX, MUL_WORD_4
                        MOV     CL, 03H
DO_AGAIN :              SHL     AX
                        ROL     DX
                        LOOP    DO_AGAIN
                        JMP     FORM_EXP
POS_EXP_6 :             CMP     BL, 06H
                        JNZ     POS_EXP_7
                        CMP     AX, 07CFH
                        JBE     NXT_30
                        JMP     ERROR_1
NXT_30 :                MUL     AX, MUL_WORD_5
                        MOV     CL, 04H
DO_AGAIN_1 :            SHL     AX
```

```
                        ROL     DX
                        LOOP    DO_AGAIN_1
                        JMP     FORM_EXP
POS_EXP_7 :             CMP     BL, 07H
                        JNZ     POS_EXP_8
                        CMP     AX, 00C7H
                        JAE     NXT_23
                        JMP     ERROR_1
NXT_23 :                MUL     AX, MUL_WORD_0
                        DCR     BL
                        JMP     NXT_30
POS_EXP_8 :             CMP     BL, 08H
                        JNZ     POS_EXP_9
                        CMP     AX, 0013H
                        JBE     NXT_25
                        JMP     ERROR_1
NXT_25 :                MUL     AX, MUL_WORD_6
                        MOV     CL, 04H
DO_AGAIN_2 :            SHL     AX
                        ROL     DX
                        LOOP    DO_AGAIN_2
                        JMP     FORM_EXP
POS_EXP_9 :             CMP     BL_09H
                        JZ      NXT_26
                        JMP     ERROR_1
                        CMP     AX, 0001H
                        JZ      NXT_27
                        JMP     ERROR_1
NXT_27 :                MUL     AX, MUL_WORD_7
                        MOV     CL, 0EH
DO_AGAIN_3 :            SHL     AX
                        ROL     DX
                        LOOP    DO_AGAIN_3
FORM_EXP :              CMP     DX, 0000H
                        JZ      NXT_1
                        MOV     CL, 10H
                        MOV     BL, 21H
                        CLC
FIND_EXP :              DCR     CL
                        DCR     BL
                        SHL     DX
                        JNC     FIND_EXP
                        SHR     AX, CL
                        ADD     AX, DX
CONT_5 :                MOV     DL, 01H
                        ADD     BL, 20H
                        SHL     BL
                        CMP     MARK_2, FFH
```

```
                        JNZ     NXT_2
                        ADD     BL, 80H
NXT_2 :                 ADD     DL, BL
                        MOV     CL, 9EH
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), CL
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), DL
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), AH
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), AL
                        RET
NXT_1 :                 MOV     BL, 11H
                        DCR     BL
DO_AGAIN_4 :            SHL     AX
                        JNC     DO_AGAIN_4
                        JMP     CONT_5
NEG_EXP_24R :           CMP     BL, FFH
                        JNZ     N_E_2
                        CALL    FORM_COUNT
                        MOV     BH, F4H
                        MOV     CX, A000H
N_E_10 :                CMP     DH, 4FH
                        JE      N_E_11
                        JA      N_E_12
                        SHL     AX
                        ROL     DX
                        INR     BL
                        JMP     N_E_10
N_E_11 :                CMP     DL, FFH
                        JAE     N_E_13
                        SHL     AX
                        ROL     DX
                        INR     BL
                        JMP     N_E_10
N_E_13 :                CMP     AH, B0
                        JB      N_E_10
                        CALL    PROC_Y
                        JMP     LOAD_ICP
N_E_12 :                CALL    PROC_X
                        JMP     LOAD_ICP
N_E_2 :                 CMP     BL, FEH
                        JNZ     N_E_3
```

```
                    CALL    FORM_COUNT
                    MOV     BH, F8H
                    MOV     CX, 6400H
                    SHR     DX
                    ROR     AX
                    DCR     BL
N_E_20 :            CMP     DH, 31H
                    JE      N_E_21
                    JA      N_E_22
                    SHL     AX
                    ROL     DX
                    INR     BL
                    JMP     N_E_20
N_E_21 :            CMP     DL, FFH
                    JAE     N_E_23
                    SHL     AX
                    ROL     DX
                    INR     BL
                    JMP     N_E_20
N_E_23 :            CMP     AH, CEH
                    JB      N_E_20
                    CALL    PROC_Y
                    JMP     LOAD_ICP
N_E_22 :            CALL    PROC_X
                    JMP     LOAD_ICP
N_E_3 :             CMP     BL, F0H
                    JNZ     N_E_4
                    CALL    FORM_COUNT
                    MOV     BH, 04H
                    MOV     CX, 3E80H
                    SHR     DX
                    ROR     AX
                    SHR     DX
                    ROR     AX
                    SUB     BL, 02H
N_E_30 :            CMP     DH, 13H
                    JE      N_E_31
                    JA      N_E_32
                    SHL     AX
                    ROL     DX
                    INR     BL
                    JMP     N_E_30
N_E_31 :            CMP     DL, FFH
                    JAE     N_E_33
                    SHL     AX
                    ROL     DX
                    INR     BL
                    JMP     N_E_30
```

```
N_E_33 :        CMP     AH, F8H
                JB      N_E_30
                CALL    PROC_Y
                JMP     LOAD_ICP
N_E_32 :        CALL    PROC_X
                JMP     LOAD_ICP
N_E_4 :         CMP     BL, FCH
                JNZ     N_E_5
                CALL    FORM_COUNT
                MOV     BH, 00H
                MOV     CX, 2710H
                SHR     DX
                ROR     AX
                SHR     DX
                ROR     AX
                SUB     BL, 02H
N_E_40 :        CMP     DH, 13H
                JE      N_E_41
                JA      N_E_42
                SHL     AX
                ROL     DX
                INR     BL
                JMP     N_E_40
N_E_41 :        CMP     DL, 87H
                JAE     N_E_43
                SHL     AX
                ROL     DX
                INR     BL
                JMP     N_E_40
N_E_43 :        CMP     AH, EDH
                JB      N_E_40
                CALL    PROC_Y
                JMP     LOAD_ICP
N_E_42 :        CALL    PROC_X
                JMP     LOAD_ICP
N_E_5 :         CMP     BL, FBH
                JNZ     N_E_6
                CALL    FORM_COUNT
                MOV     BH, 03H
                MOV     CX, 30D4H
                SHR     DX
                ROR     AX
                SHR     DX
                ROR     AX
                SUB     BL, 02H
N_E_50 :        CMP     DH, 18H
                JE      N_E_51
                JA      N_E_52
```

```
                    SHL     AX
                    ROL     DX
                    INR     BL
                    JMP     N_E_50
N_E_51 :            CMP     DL, 69H
                    JAE     N_E_53
                    SHL     AX
                    ROL     DX
                    INR     BL
                    JMP     N_E_50
N_E_53 :            CMP     AH, E7H
                    JB      N_E_50
                    CALL    PROC_Y
                    JMP     LOAD_ICP
N_E_52 :            CALL    PROC_X
                    JMP     LOAD_ICP
N_E_6 :             CMP     BL, FAH
                    JNZ     N_E_7
                    CALL    FORM_COUNT
                    MOV     BH, 06H
                    MOV     CX, 3D09H
                    SHR     DX
                    ROR     AX
                    SHR     DX
                    ROR     AX
                    SUB     BL, 02H
N_E_60 :            CMP     DH, 1EH
                    JE      N_E_61
                    JA      N_E_62
                    SHL     AX
                    ROL     DX
                    INR     BL
                    JMP     N_E_60
N_E_61 :            CMP     DL, 80H
                    JAE     N_E_63
                    SHL     AX
                    ROL     DX
                    INR     BL
                    JMP     N_E_60
N_E_63 :            CMP     AH, 61H
                    JB      N_E_60
                    CALL    PROC_Y
                    JMP     LOAD_ICP
N_E_62 :            CALL    PROC_X
                    JMP     LOAD_ICP
N_E_7 :             CMP     BL, 09H
                    JNZ     N_E_8
                    INR     BL
```

```
                JMP     N_E_6
N_E_8 :         CMP     BL, F8H
                JNZ     N_E_9
                CALL    FORM_COUNT
                MOV     BH, 09H
                MOV     CX, 5F5EH
                SHR     DX
                ROR     AX
                DCR     BL
N_E_80 :        CMP     DH, 2FH
                JE      N_E_81
                JA      N_E_82
                SHL     AX
                ROL     DX
                INR     BL
                JMP     N_E_80
N_E_81 :        CMP     DL, AEH
                JAE     N_E_83
                SHL     AX
                ROL     DX
                INR     BL
                JMP     N_E_80
N_E_83 :        CMP     AH, 00H
                JB      N_E_80
                CALL    PROC_Y
                JMP     LOAD_ICP
N_E_82 :        CALL    PROC_X
                JMP     LOAD_ICP
N_E_9 :         CMP     BL, F7H
                JNZ     N_E_A
                CALL    FORM_COUNT
                MOV     BH, 14H
                MOV     CX, EE6BH
N_E_90 :        CMP     DH, 77H
                JE      N_E_91
                JA      N_E_92
                SHL     AX
                ROL     DX
                INR     BL
                JMP     N_E_90
N_E_91 :        CMP     DL, 35H
                JAE     N_E_93
                SHL     AX
                ROL     DX
                INR     BL
                JMP     N_E_90
N_E_93 :        CMP     AH, 08H
                JB      N_E_90
```

```
                        CALL    PROC_Y
                        JMP     LOAD_ICP
N_E_92 :                CALL    PROC_X
                        JMP     LOAD_ICP
N_E_A :                 CMP     BL, F6H
                        JNZ     N_E_B
                        CMP     DL, 00H
                        JNZ     NXT_1
                        CMP     AX, 0002H
                        JAE     NXT_1
                        JMP     ERROR_1
NXT_1 :                 MOV     CX, 9503H
                        CALL    FORM_COUNT
                        MOV     BH, 18H
N_E_A0 :                CMP     DH, 4AH
                        JE      N_E_A1
                        JA      N_E_A2
                        SHL     AX
                        ROL     DX
                        INR     BL
                        JMP     N_E_A0
N_E_A1 :                CMP     DL, 81H
                        JAE     N_E_A3
                        SHL     AX
                        ROL     DX
                        INR     BL
                        JMP     N_E_A0
N_E_A3 :                CMP     AH, 35H
                        JB      N_E_A0
                        CALL    PROC_Y
                        JMP     LOAD_ICP
N_E_A2 :                CALL    PROC_X
                        JMP     LOAD_ICP
N_E_B :                 CMP     BL, F5H
                        JNZ     N_E_C
                        CMP     DL, 00H
                        JNZ     NXT_2
                        CMP     AX, 0014H
                        JA      NXT_2
                        JMP     ERROR_1
NXT_2 :                 MOV     CX, 2E91H
                        CALL    FORM_COUNT
                        MOV     BH, E9H
                        SHR     DX
                        ROR     AX
                        SHR     DX
                        ROR     AX
                        SUB     BL, 02H
```

```
N_E_B0 :        CMP     DH, 17H
                JE      N_E_B1
                JA      N_E_B2
                SHL     AX
                ROL     DX
                INR     BL
                JMP     N_E_B0
N_E_B1 :        CMP     DL, 48H
                JAE     N_E_B3
                SHL     AX
                ROL     DX
                INR     BL
                JMP     N_E_B0
N_E_B3 :        CMP     AH, 6EH
                JB      N_E_B0
                CALL    PROC_Y
                JMP     LOAD_ICP
N_E_B2 :        CALL    PROC_X
                JMP     LOAD_ICP
N_E_C :         CMP     BL, F4H
                JNZ     N_E_D
                CMP     DL, 00H
                JNZ     NXT_3
                CMP     AX, 00C8H
                JA      NXT_3
                JMP     ERROR_1
NXT_3 :         MOV     CX, 3A55H
                CALL    FORM_COUNT
                MOV     BH, 26H
                SHR     DX
                ROR     AX
                SHR     DX
                ROR     AX
                SUB     BL, 02H
N_E_C0 :        CMP     DH, 1CH
                JE      N_E_C1
                JA      N_E_C2
                SHL     AX
                ROL     DX
                INR     BL
                JMP     N_E_C0
N_E_C1 :        CMP     DL, 1AH
                JAE     N_E_C3
                SHL     AX
                ROL     DX
                INR     BL
                JMP     N_E_C0
N_E_C3 :        CMP     AH, 62H
```

```
                    JB      N_E_C0
                    CALL    PROC_Y
                    JMP     LOAD_ICP
N_E_C2 :            CALL    PROC_X
                    JMP     LOAD_ICP
N_E_D :             CMP     BL, F3H
                    JNZ     N_E_E
                    CMP     DL, 00H
                    JNZ     NXT_4
                    CMP     AX, 07E0H
                    JAE     NXT_4
                    JMP     ERROR_1
NXT_4 :             MOV     CX, 9185H
                    CALL    FORM_COUNT
                    MOV     BH, 28H
N_E_D0 :            CMP     DH, 48H
                    JE      N_E_D1
                    JA      N_E_D2
                    SHL     AX
                    ROL     DX
                    INR     BL
                    JMP     N_E_D0
N_E_D1 :            CMP     DL, B2H
                    JAE     N_E_D3
                    SHL     AX
                    ROL     DX
                    INR     BL
                    JMP     N_E_D0
N_E_D3 :            CMP     AH, 37H
                    JB      N_E_D0
                    CALL    PROC_Y
                    JMP     LOAD_ICP
N_E_D2 :            CALL    PROC_X
                    JMP     LOAD_ICP
N_E_E :             CMP     BL, F2H
                    JNZ     N_E_F
                    CMP     DL, 00H
                    JNZ     NXT_5
                    CMP     AX, 4E20H
                    JAE     NXT_5
                    JMP     ERROR_1
NXT_5 :             MOV     CX, B5E6H
                    CALL    FORM_COUNT
                    MOV     BH, 31H
N_E_E0 :            CMP     DH, 5AH
                    JE      N_E_E1
                    JA      N_E_E2
                    SHL     AX
```

```
                        ROL     DX
                        INR     BL
                        JMP     N_E_E0
N_E_E1 :                CMP     DL, F2H
                        JAE     N_E_E3
                        SHL     AX
                        ROL     DX
                        INR     BL
                        JMP     N_E_E0
N_E_E3 :                CMP     AH, ADH
                        JB      N_E_E0
                        CALL    PROC_Y
                        JMP     LOAD_ICP
N_E_E2 :                CALL    PROC_X
                        JMP     LOAD_ICP
N_E_F :                 CMP     BL, F1H
                        JZ      NXT_F
                        JMP     ERROR_1
NXT_F :                 CMP     DL, 03H
                        JAE     NXT_6
                        JMP     ERROR_1
NXT_6 :                 MOV     CX, 3D38H
                        CALL    FORM_COUNT
                        MOV     BH, 36H
N_E_F0 :                CMP     DH, 1CH
                        JE      N_E_F1
                        JA      N_E_F2
                        SHL     AX
                        ROL     DX
                        INR     BL
                        JMP     N_E_F0
N_E_F1 :                CMP     DL, 6BH
                        JAE     N_E_F3
                        SHL     AX
                        ROL     DX
                        INR     BL
                        JMP     N_E_F0
N_E_F3 :                CMP     AH, E3H
                        JB      N_E_F0
                        CALL    PROC_Y
                        JMP     LOAD_ICP
N_E_F2 :                CALL    PROC_X
                        JMP     LOAD_ICP
FORM_COUNT :            MOV     CH, 00H
                        CMP     AL, 00H
                        JAZ     NXT_11
                        CMP     AH, 00H
                        JNZ     NXT_12
```

```
                    MOV     BL, 16H
AGAIN_1 :           INR     BL
                    SHL     AL
                    JC      NXT_13
                   JMP     AGAIN_1
NXT_13 :            ROR     AL
                    MOV     DH, AL
                    SHR     DX
                    MOV     AX, 00H
                    RET
NXT_12 :            MOV     BL, 0FH
AGAIN_2 :           INR     BL
                    SHL     AX
                    JC      NXT_21
                    JMP     AGAIN_2
NXT_21 :            ROR     AX
                    MOV     DX, AX
                    SHR     DX
                    MOV     AX, 0000H
                    RET
NXT_11 :            MOV     BL, 0AH
                    MOV     CL, 04H
                    SHL     DL, CL
AGAIN_3 :           INR     BL
                    SHL     DL
                    JC      NXT_14
                    JMP     AGAIN_3
NXT_14 :            ROR     DL
                    SHR     DL
                    MOV     BH, 00H
                    MOV     CL, 11H
                    SUB     CL, BL
AGAIN_4 :           SHR     AX
                    ROR     BH
                    LOOP    AGAIN_4
                    ADD     DX, AX
                    MOV     AH, BH
                    MOV     AL, 00H
                    RET
PROC_X :            DIV     DX, CX
                    SHR     CX
                    SHL     AX
                    CMP     DX, CX
                    JB      NXT_XX2
                    INR     AX
                    SUB     DX, CX
NXT_XX2 :           SHR     CX
                    CMP     DX, CX
```

```
                    JB      NXT_XX4
                    MOV     DL, 00H
                    SHL     AX
                    ROL     DX
                    INR     AX
NXT_XX4 :           MOV     DH, 31H
                    SUB     DH, BL
                    ADD     DH, BH
                    SHL     DH
                    ADD     DL, DH
                    CMP     MARK_2, FFH
                    JNZ     NXT_XX5
                    ADD     DL, 80H
NXT_XX5 :           RET
PROC_Y :            DIV     DX, CX
                    SHR     CX
                    CMP     DX, CX
                    JB      NXT_YY1
                    MOV     DX, 0000H
                    SHL     AX
                    ROL     DX
                    INR     AX
                    JMP     NXT_YY2
NXT_YY1 :           MOV     DX, 0000H
                    SHL     AX
                    ROL     DX
NXT_YY2 :           MOV     DH, 31H
                    SUB     DH, BL
                    ADD     DH, BH
                    SHL     DH
                    ADD     DL, DH
                    CMP     MARK_2, FFH
                    JNZ     NXT_YY3
                    ADD     DL, 80H
NXT_YY3 :           RET
LOAD_ICP :          INR     SI
                    INCREMENT_SI
                    MOV     (SI), 9CH           ; IDENTIFIER 24R
                    INR     SI
                    INCREMENT_SI
                    MOV     (SI), DL
                    INR     SI
                    INCREMENT_SI
                    MOV     (SI), AH
                    INR     SI
                    INCREMENT_SI
                    MOV     (SI), AL
                    RET
```

```
PART_4 :

CHK_16R :       MOV   DX, 0005H
                MOV   BH, 00H
                MOV   CX, DX
CHK_AGAIN :     DCR   DI
                MOV   AL, (BP+DI)
                CMP   AL, '.'
                JB    NXT_1
                JE    NXT_0
                JMP   ERROR_1
NXT_0 :         INR   BH
NXT_1 :         CMP   AL, 0AH
                JB    CONT_1
                JMP   ERROR_1
CONT_1 :        CMP   AL, 00H
                JNZ   NXT_2
                INR   DH
NXT_2 :         LOOP  CHK_AGAIN
                CMP   BH, 01H
                JZ    CONT_2
                JMP   ERROR_1
CONT_2 :        CMP   DH, 04H
                JNZ   CONT_3
                JMP   ERROR_1
CONT_3 :        MOV   DH, 00H
                ADD   DI, DX
                MOV   BL, 00H
                CMP   (BP+DI), 'E'
                JNE   CHK_NXT_1
                INR   DI
                CMP   (BP+DI), '-'
                JNZ   CHK_NXT_2
                MOV   MARK_2, FFH
                INR   DI
CHK_NXT_2 :     MOV   AH, 00H
                CMP   (BP+DI), 00H
                JBE   CONT_4
                JMP   ERROR_1
CONT_4 :        INR   DI
                CMP   (BP+DI), 0AH
                JB    CONT_5
                JMP   ERROR_1
CONT_5 :        MOV   AL, (BP+DI)
                MOV   BL, AL
                CMP   MARK_2, FFH
                JNE   CHK_NXT_1
```

```
                        NEG     BL
                        INR     DI
                        CMP     (BP+DI), 'ENTER'
                        JAE     CONT_6
                        JMP     ERROR_1
CONT_6 :                SUB     DI, 0002H
                        CMP     (BP+DI), 'E'
                        JE      CHK_NXT_1
                        DCR     DI
CHK_NXT_1 :             DCR     DI
                        MOV     AX, 0000H
                        MOV     CX, 0000H
                        MOV     DX, 0000H
                        MOV     BH, 00H
                        CMP     (BP+DI), '.'
                        JE      CHK_NXT_4
                        MOV     AL, (BP+DI)
                        ADD     CL, AL
CHK_NXT_4 :             DCR     DI
                        CMP     (BP+DI), '.'
                        JB      CHK_NXT_5
                        DCR     BL
                        DCR     DI
CHK_NXT_5 :             MOV     AL, (BP+DI)
                        CMP     AL, 00H
                        JZ      CHK_NXT_6
                        MUL     AL, MUL_BYTE_1
                        ADD     CL, AL
CHK_NXT_6 :             DCR     DI
                        CMP     (BP+DI), '.'
                        JB      CHK_NXT_7
                        SUB     BL, 02H
                        DCR     DI
CHK_NXT_7 :             MOV     AL, (BP+DI)
                        CMP     AL, 00H
                        JZ      CHK_NXT_8
                        MUL     AX, MUL_WORD_1
                        ADD     CX, AX
                        MOV     AX, 0000H
CHK_NXT_8 :             DCR     DI
                        CMP     (BP+DI), '.'
                        JB      CHK_NXT_9
                        SUB     BL, 03H
                        DCR     DI
CHK_NXT_9 :             MOV     AL, (BP+DI)
                        CMP     AL, 00H
                        JZ      CHK_NXT_A
                        MUL     AX, MUL_WORD_2
```

```
                        ADD     CX, AX
                        SUB     AX, AX
CHK_NXT_A :             DCR     DI
                        CMP     (BP+DI), '.'
                        JB      FORM_16R
                        SUB     BL, 04H
FORM_16R :              MOV     AX, CX
                        CMP     BL, 80H
                        JB      NEG_EXP_16R
                        CMP     BL, 04H
                        JB      POS_EXP_3
                        JE      CONT_7
                        JMP     ERROR_1
CONT_7 :                CMP     AX, 0003H
                        JBE     CONT_8
                        JMP     ERROR_1
CONT_8 :                MUL     AX, MUL_WORD_3
                        JMP     FORM_EXP
POS_EXP_3 :             CMP     BL, 03H
                        JNZ     POS_EXP_2
                        CMP     AX, 001DH
                        JBE     CONT_9
                        JMP     ERROR_1
CONT_9 :                MUL     AX, MUL_WORD_2
                        JMP     FORM_EXP
POS_EXP_2 :             CMP     BL, 02H
                        JNZ     POS_EXP_1
                        CMP     AX, 012BH
                        JBE     CONT_A
                        JMP     ERROR_1
CONT_A :                MUL     AX, MUL_WORD_1
                        JMP     FORM_EXP
POS_EXP_1 :             CMP     BL, 01H
                        JNZ     POS_EXP_0
                        CMP     AX, 0BB7H
                        JBE     CONT_B
                        JMP     ERROR_1
CONT_B :                MUL     AX, MUL_WORD_0
                        JMP     FORM_EXP
POS_EXP_0 :             CMP     BL, 00H
                        JZ      CONT_C
                        JMP     ERROR_1
CONT_C :                CMP     AX, 270FH
                        JBE     FORM_EXP
                        JMP     ERROR_1
FORM_EXP :              CMP     AH, 00H
                        JNZ     NXT_E1
                        MOV     AH, 19H
```

```
                        MOV     CL, 00H
                        CLC
AGAIN_1 :               SHL     AL
                        JC      NXT_E2
                        INR     CL
                        JMP     AGAIN_1
NXT_E2 :                ROL     AL
                        SUB     AH, CL
                        SHL     AX
                        SHL     AX
                        CMP     MARK_2, FFH
                        JNZ     NXT_E3
                        ADD     AH, 80H
                        JMP     NXT_E3
NXT_E1 :                MOV     BH, 11H
                        MOV     CX, 0000H
AGAIN_2 :               SHL     AX
                        JC      NXT_E11
                        INR     CL
                        JMP     AGAIN_2
NXT_E11 :               ROR     AX
                        SUB     BH, CL
                        MOV     CL, 06H
                        SHR     AX, CX
                        SHL     BH
                        SHL     BH
                        ADD     AH, BH
                        MOV     BH, 00H
                        CMP     MARK_2, FFH
                        JNZ     NXT_E3
                        ADD     AH, 80H
NXT_E3 :                INR     SI
                        INCREMENT_SI
                        MOV     CL, 98H
                        MOV     (SI), CL            ;IDENTIFIER 16R
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), AH
                        INR     SI
                        INCREMENT_SI
                        MOV     (SI), AL
                        RET
NEG_EXP_16R :           CMP     BL, FFH
                        JNE     N_E_2
                        CALL    FORM_COUNT
                        MOV     BH, F4H
                        MOV     CX, A000H
                        SHL     AX
```

```
                    ROL     DX
                    INR     BL
                    CMP     DX, 013FH
                    JA      N_E_11
                    JB      N_E_12
                    CMP     AX, 3000H
                    JAE     N_E_11
N_E_12 :            SHL     AX
                    ROL     DX
N_E_11 :            DIV     DX, CX
                    TEST    AH, FCH
                    JNZ     CONT_1
                    JMP     FORM_EXP_1
CONT_1 :            SHR     AX
                    JMP     FORM_EXP_1
N_E_2 :             CMP     BL, FEH
                    JNE     N_E_3
                    CALL    FORM_COUNT
                    MOV     BH, F8H
                    MOV     CX, 6400H
                    CMP     DL, C7H
                    JA      N_E_21
                    JB      N_E_22
                    CMP     AX, CED0H
                    JAE     N_E_21
N_E_22 :            SHL     AX
                    ROL     DX
N_E_21 :            DIV     DX, CX
                    TEST    AH, FCH
                    JNZ     CONT_2
                    JMP     FORM_EXP_1
CONT_2 :            SHR     AX
                    JMP     FORM_EXP_1
N_E_3 :             CMP     BL, FDH
                    JNE     N_E_4
                    CALL    FORM_COUNT
                    MOV     BH, FCH
                    MOV     CX, 3E80H
                    SHR     DX
                    ROR     AX
                    DCR     BL
                    CMP     DL, 6FH
                    JA      N_E_31
                    JB      N_E_32
                    CMP     AX, E0C0H
                    JAE     N_E_31
N_E_32 :            SHL     AX
                    ROL     DX
```

```
N_E_31 :        DIV     DX, CX
                TEST    AH, FCH
                JNZ     CONT_3
                JMP     FORM_EXP_1
CONT_3 :        SHR     AX
                JMP     FORM_EXP_1
N_E_4 :         CMP     BL, FCH
                JNE     N_E_5
                CALL    FORM_COUNT
                MOV     BH, 00H
                MOV     CX, 2710H
                SHR     DX
                ROR     AX
                DCR     BL
                CMP     DL, 4EH
                JA      N_E_41
                JB      N_E_42
                CMP     AX, 4C71H
                JAE     N_E_41
N_E_42 :        SHL     AX
                ROL     DX
N_E_41 :        DIV     DX, CX
                TEST    AH, FCH
                JNZ     CONT_4
                JMP     FORM_EXP_1
CONT_4 :        SHR     AX
                JMP     FORM_EXP_1
N_E_5 :         CMP     BL, FBH
                JNE     N_E_6
                CALL    FORM_COUNT
                MOV     BH, 03H
                MOV     CX, 30D4H
                SHR     DX
                ROR     AX
                DCR     BL
                CMP     DL, 61H
                JA      N_E_51
                JB      N_E_52
                CMP     AX, A7A6H
                JAE     N_E_51
N_E_52 :        SHL     AX
                ROL     DX
N_E_51 :        DIV     DX, CX
                TEST    AH, FCH
                JNZ     CONT_5
                JMP     FORM_EXP_1
CONT_5 :        SHR     AX
                JMP     FORM_EXP_1
```

```
N_E_6 :         CMP   BL, FAH
                JNE   N_E_7
                CMP   AL, 08H
                JBE   CONT_0
                JMP   ERROR_1
CONT_0 :        CALL  FORM_COUNT
                MOV   BH, 06H
                MOV   CX, 3D09H
                SHR   DX
                ROR   AX
                DCR   BL
                CMP   DL, 7AH
                JA    N_E_61
                JB    N_E_62
                CMP   AX, FCFBH
                JAE   N_E_61
N_E_62 :        SHL   AX
                ROL   DX
N_E_61 :        DIV   DX, CX
                TEST  AH, FCH
                JNZ   CONT_6
                JMP   FORM_EXP_1
CONT_6 :        SHR   AX
                JMP   FORM_EXP_1
N_E_7 :         CMP   BL, F9H
                JNZ   N_E_8
                INR   BL
                CMP   AX, 0050H
                JAE   CONT_7
                JMP   ERROR_1
CONT_7 :        JMP   N_E_6
N_E_8 :         CMP   BL, F8H
                JNE   N_E_9
                CMP   AX, 031FH
                JBE   CONT_01
                JMP   ERROR_1
CONT_01 :       CALL  FORM_COUNT
                MOV   BH, 09H
                MOV   CX, 5F5EH
                CMP   DL, BEH
                JA    N_E_81
                JB    N_E_82
                CMP   AX, F400H
                JAE   N_E_81
N_E_82 :        SHL   AX
                ROL   DX
N_E_81 :        DIV   DX, CX
                TEST  AH, FCH
```

```
                    JNZ     CONT_8
                    JMP     FORM_EXP_1
CONT_8 :            SHR     AX
                    JMP     FORM_EXP_1
N_E_9 :             CMP     BL, F7H
                    JZ      CONT_02
                    JMP     ERROR_1
CONT_02 :           CMP     AX, 1F3FH
                    JAE     CONT_9
                    JMP     ERROR_1
CONT_9 :            CALL    FORM_COUNT
                    MOV     BH, 14H
                    MOV     CX, EE6BH
                    SHL     AX
                    ROL     DX
                    INR     BL
                    CMP     DX, 01DCH
                    JA      N_E_91
                    JB      N_E_92
                    CMP     AX, FCCAH
                    JAE     N_E_91
N_E_92 :            SHL     AX
                    ROL     DX
N_E_91 :            DIV     DX, CX
                    TEST    AH, FCH
                    JNZ     CONT_91
                    JMP     FORM_EXP_1
CONT_91 :           SHR     AX
                    JMP     FORM_EXP_1
FORM_COUNT :        MOV     DX, 0000H
                    CMP     AH, 00H
                    JA      NXT_F1
                    MOV     BL, 0FH
                    MOV     CL, 00H
                    INR     CL
                    SHL     AL
                    JNC     AGAIN_2
                    ROR     AL
                    MOV     DL, AL
                    ADD     BL, CL
                    MOV     AL, 00H
                    RET
NXT_F1 :            MOV     BL, 07H
                    MOV     CL, 00H
AGAIN_2 :           INR     CL
                    SHL     AX
                    JNC     AGAIN_2
                    ROR     AX
```

```
                    ADD     BL, CL
                    MOV     DL, AH
                    MOV     AH, AL
                    MOV     AC, 00H
                    RET
FORM_EXP_1 :        MOV     DH, 1AH
                    SUB     DH, BL
                    ADD     DH, BH
                    SHL     DH
                    SHL     DH
                    ADD     AH, DH
                    CMP     MARK_2, FFH
                    JNZ     NXT_F2
                    ADD     AH, 80H
NXT_F2 :            INR     SI
                    INCREMENT_SI
                    MOV     CL, 98H
                    MOV     (SI), CL            ;IDENTIFIER 16R
                    INR     SI
                    INCREMENT_SI
                    MOV     (SI), AH
                    INR     SI
                    INCREMENT_SI
                    MOV     (SI), AL
                    RET
```

TITLE : MODULE : INITIALISE RUN ACTION LEVEL_2 F.C. 5.1

PART_1 :

ANATOMY :

ARRANGING THE PROGRAM IN ASCENDING ORDER, REFER SECTION 5. THE OPTION SELECTED FOR SEQUENCING THE PROGRAM (ADDITIONAL EDITING FACILITY) REARRANGES THE PROGRAM STORAGE IN THE ASCENDING ORDER. IF THE SENTENCE IS OVER WRITTEN THE LATEST ONE IS CONSIDERED VALID. IF THE SENTENCE IS AN INSERTION IT IS EXCLUSIVELY LOADED IN A BUFFER AREA INDICATING ITS APPEARANCE IN THE IC FIELD OF LINE NUMBER NEXT TO THE SENTENCE INSERTED. SIMILAR PROCEDURE IS ADOPTED IF OVER WRITTEN STATEMENT IS FOUND OF LENGTH GREATER THAN THE PREVIOUS ONE. THIS PROGRAM GENERATES LINE NUMBER TABLE AND SI TABLE FOR REFERENCE AT THE RUN TIME.

REFERENCE MEMORY LOCATIONS :

PMTR_POINT_PRES_STATUS, END_BLK_PI, MARK_EXEC_PI, NXT_ST_BLK_PI, NXT_END_BLK_PI, ST_BLK_PI, LN_NO_TAB, PRESENT_PI, STATUS_FOR_DET_PI, COUNT, REF_BLK, ATTACH_BYTE, SI_TAB, ST_ADD_BUFFER, STATUS_FOR_DET_BLK, PRESENT_STATUS_PI, ENDING_SI, STARTING_SI, NXT_ENDING_SI, MARK_SI, MARK_DS, PMTR_STATUS_DET, PREV_END_BLK_PI.

REGISTER USAGE :

CX : POINTING TO VARIOUS LOCATIONS IN PIT, POINTING TO THE BLOCK WHERE SENTENCE IS TO BE OVER WRITTEN OR INSERTED, MANAGING SEGMENT, COUNTER.

BX, DX : MANAGING SEGMENT AND STRING TRANSFER.

```
RUN_LEVEL_1 :     MOV   LN_NO_TAB, BASE_ADD_LN_NO_TAB
                  MOV   SI_TAB, BASE_ADD_SI_TAB
                  MOV   ATTACH_BYTE, 00H
                  MOV   ST_ADD_BUFFER, BASE_ADD_RAM_BUFFER
                  MOV   COUNT, 00H
                  MOV   PREV_END_BLK_PI, 0000H
                  MOV   SI  BASE_ADD_PIT
                  MOV   CH, 00H
                  MOV   CL, AL
                  ADD   CL, 03H
                  SHL   CL                  , 4 TIMES
```

```
                        ADD     SI, CX
                        ADD     SI, 0002H
                        MOV     DL, (SI)
                        MOV     PRESENT_DI, DL
                        INR     SI
                        MOV     CL, (SI)
                        TEST    CL, 80H
                        JNZ     CONT_0
                        JMP     ERROR_1
CONT_0 :                SUB     CL, 80H
                        MOV     DL, CL
                        INR     CL
                        MOV     PMTR_POINT_PRES_STATUS, CL
CONT_21 :               INR     CNT
                        SUB     CL, DL
                        SHL     CL                      , 2 TIMES
                        SUB     SI, 0003H
                        ADD     SI, CX
                        MOV     ST_BLK_PI, CX
                        ADD     SI, 0002H
                        MOV     BX, (SI)
                        SUB     BX, CX
                        CMP     BX, 00FFH
                        JBE     CONT_1
                        ADD     CX, 00FFH
                        MOV     END_BLK_PI, CX
                        MOV     MARK_EXEC_PI, 0FH
;MARK  LOADED TO INDICATE THAT THE PROGRAM HAS CONTINUED IN  NEXT
;SEGMENT.
                        INR     CX
                        MOV     NXT_SI_BLK_PI, CX
                        SUB     CX, 0100H
                        ADD     BX, CX
                        MOV     NXT_END_BLK_PI, BX
                        JMP     CONT_2
CONT_1 :                ADD     BX, CX
                        MOV     END_BLK_PI, BX
                        MOV     CX, ST_BLK_PI
                        MOV     BX, CX
                        SUB     CX, PREV_END_BLK_PI
                        SHL     CL
                        MOV     SI, LN_NO_TAB
CONT_0 :                MOV     (SI), FFH
;FF  LOADED  TO INDICATE THE PRESENT PI DOES NOT  RESIDE  IN  THE
;CORRESPONDING BLOCK.
                        INR     SI
                        LOOP    CONT_0
                        MOV     (SI), 00H
```

```
                        INR     SI
                        MOV     (SI), 00H
                        INR     SI
                        MOV     LN_NO_TAB, SI
                        MOV     CX, END_BLK_PI
                        MOV     PREV_END_BLK_PI, CX
                        SHL     BX                      , 4 TIMES
                        MOV     DS, BX
                        MOV     SI, 0001H
                        CMP     (SI), 00H
                        JZ      CHK_NXT_SNTCE_0
                        JMP     ERROR_1
CHK_NXT_SNTCE_0 :       ADD     SI, 0008H
CONT_3 :                MOV     AX, (SI)
                        DCR     SI
CHK_NXT_SNTCE_1 :       ADD     SI, 0008H
                        CMP     (SI), FFH
                        JNZ     CONT_61
                        TEST    SI, 00FFH
                        JNZ     CHK_NXT_SNTCE_1
                        PUSH    DS
                        PUSH    SI
                        MOV     CX, 0000H
                        MOV     DS, CX
                        INR     AX
                        MOV     SI, LN_NO_TAB
                        MOV     (SI), AX
;SINCE  SAME  SENTENCE CONTINUED IN NEXT BLOCK,  LINE  NUMBER  IS
;INCREMENTED AND PUT IN LINE NUMBER TABLE.
                        DCR     AX
                        ADD     SI, 0002H
                        MOV     LN_NO_TAB, SI
                        MOV     CX, ST_BLK_PI
                        INR     CX
                        MOV     ST_BLK_PI, CX
                        CMP     CX, END_BLK_PI
                        JA      CONT_NXT_SEG
                        POP     SI
                        POP     DS
                        JMP     CHK_NXT_SNTCE
CONT_NXT_SEG :          CMP     MARK_EXEC_PI, OFH
                        JZ      CONT_4
                        JMP     ERROR_3
CONT_4 :                MOV     CX, NXT_ST_BLK_PI
                        MOV     ST_BLK_PI, CX
                        MOV     CX, NXT_END_BLK_PI
                        MOV     END_BLK_PI, CX
                        MOV     MARK_EXEC_PI, 00H
```

```
                        POP     SI
                        POP     DS
                        MOV     CX, DX
                        ADD     CX, 1000H
                        MOV     DS, CX
                        JMP     CHK_NXT_SNTCE_1
CONT_61 :               TEST    SI, 00FFH
                        JNZ     CONT_6
                        PUSH    DS
                        PUSH    SI
                        INR     AX
                        MOV     SI, LN_NO_TAB
                        MOV     (SI), AX
                        DCR     AX
                        ADD     SI, 0002H
                        MOV     LN_NO_TAB, SI
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     CX, ST_BLK_PI
                        INR     CX
                        MOV     ST_BLK_PI, CX
                        JA      CONT_7
                        POP     SI
                        POP     DS
                        JMP     CONT_6
CONT_7 :                CMP     MARK_EXEC_P1, OFH
                        JZ      CONT_8
                        JMP     CONT_9
CONT_8 :                MOV     CX, NXT_ST_BLK_PI
                        MOV     ST_BLK_PI, CX
                        MOV     CX, NXT_END_BLK_PI
                        MOV     END_BLK_PI, CX
                        MOV     MARK_EXEC_PI, 00H
                        POP     SI
                        POP     DS
                        MOV     CX, DS
                        ADD     CX, 1000H
                        MOV     DS, CX
CONT_6 :                INR     SI
                        CMP     AX, (SI)
                        JAE     CONT_10
                        JMP     CONT_3
CONT_10 :               MOV     AX, (SI)
                        ADD     SI, 0002H
                        CMP     (SI), FFH
                        JNZ     CONT_11
                        JMP     NULL_STATEMENT
```

```
CONT_11 :           MOV   CH, 00H
                    MOV   CL, 01H
                    ADD   SI, 0005H

NXT_1 :             CMP   (SI), FFH
                    JNZ   NXT_2
                    TEST  SI, 00FFH
                    JNZ   NXT_11
                    PUSH  SI
                    PUSH  DS
                    MOV   DX, 0000H
                    MOV   DS, DX
                    MOV   SI, LN_NO_TAB
                    INR   AX
                    MOV   (SI), AX
                    DCR   AX
                    ADD   SI, 0002H
                    MOV   LN_NO_TAB, SI
                    POP   SI
                    POP   DS
NXT_11 :            INR   CL
                    ADD   SI, 0008H
                    JMP   NXT_1
NXT_2 :             MOV   BL, CL
;LENGTH  OF  THE SENTENCE IS LOADED IN ATTACH BYTE WHICH  IS  THE
;FIRST LOCATION OF ICP.
                    SHL   CL                  , 3 TIMES
                    SUB   SI, CL
                    MOV   (SI), BL
                    PUSH  DS
                    PUSH  SI
                    MOV   CX, 0000H
                    MOV   DS, CX
                    MOV   SI, LN_NO_TAB
NXT_3 :             CMP   AX, (SI)
                    JAE   NXT_4
                    INR   CX
                    SUB   SI, 0002H
                    JMP   NXT_3
NXT_4 :             MOV   SI, BASE_ADD_PIT
                    MOV   DX, CX
                    MOV   CL, PRESENT_PI
                    MOV   CH, 00H
                    ADD   CL, 03H
                    SHL   CL                  , 4 TIMES
                    ADD   SI, CX
                    MOV   CL, STATUS_FOR_DET_BLK
                    SUB   CL, PRESENT_STATUS
```

```
                    SHL   CL                    , 2 TIMES
                    ADD   SI, CX
                    MOV   CX, ST_BLK_PI
                    INR   CX
                    MOV   REF_BLK, CX
                    SUB   CX, (SI)
                    MOV   BH, COUNT
NXT_6 :             CMP   DX, CX
                    JBE   NXT_5
                    DCR   BH
                    JNZ   NXT_61
                    JMP   ERROR_1
NXT_61 :            SUB   DX, CX
                    SUB   SI, 0002H
                    MOV   CX, (SI)
                    INR   CX
                    MOV   REF_BLK, CX
                    SUB   SI, 0002H
                    SUB   CX, (SI)
                    JMP   NXT_6
NXT_5 :             MOV   CX, REF_BLK
                    SUB   CX, DX
                    MOV   SI, 0000H
                    SHL   CX                    , 4 TIMES
                    MOV   DS, CX
CONT_12 :           CMP   (SI), FFH
                    JNZ   CONT_12
                    ADD   SI, 0008H
                    JMP   CONT_12
CONT_12 :           INR   SI
                    CMP   AX, (SI)
                    JB    NXT_7
                    JE    NXT_8
                    ADD   SI, 0007H
                    JMP   NXT_12
NXT_8 :             MOV   CH, 00H
                    MOV   CL, 01H
                    DCR   SI
NXT_82 :            ADD   SI, 0008H
                    CMP   (SI), FFH
                    JNZ   NXT_81
                    INR   CL
                    JMP   NXT_82
NXT_81 :            MOV   BL, CL
                    SHL   CL                    , 8 TIMES
                    SUB   SI, CX
                    MOV   CH, 00H
                    MOV   CL, BH
```

```
NXT_81 :        CMP     BL, CL
                JA      NXT_83
                JB      NXT_84
                MOV     ES, DS
                MOV     DI, SI
                ADD     DI, 0003H
                POP     SI
                POP     DS
                MOV     (SI), 00H
                ADD     SI, 0003H
            REP MOVSB
                MOV     (SI), FFH
                INR     SI
                INR     DI
                SHL     CL                      , 3 TIMES
                SUB     CL, 04H
            REP MOVSB
                INR     SI
                MOV     CX, 0000H
                MOV     ES, CX
                JMP     CONT_61
NXT_84 :        MOV     ES, DS
                MOV     DI, SI
                ADD     DI, 0003H
                POP     SI
                POP     DS
                MOV     (SI), 00H
                ADD     SI, 0003H
            REP MOVSB
                MOV     (SI), FFH
                INR     SI
                INR     DI
                MOV     CH, BL
                MOV     BL, CL
                MOV     CL, CH
                MOV     CH, 00H
                SUB     BL, CL
                SHL     CL                      , 3 TIMES
                SUB     CL, 04H
            REP MOVSB
                MOV     CL, BL
                MOV     SS, ES
                MOV     BP, 0000H
                INR     SI
                INR     DI
NXT_841 :       MOV     (BP+DI), 00H
                INR     SI
                INR     DI
```

```
                LOOP    NXT_841
                MOV     CX, 0000H
                MOV     SS, CX
                MOV     ES, CX
                JMP     CONT_61
NXT_83 :        ADD     SI, 0003H
                MOV     (SI), FFH
                SHL     CL                      , 3 TIMES
                SUB     SI, 0003H
                ADD     SI, CX
                MOV     CL, ATTACH_BYTE
                CMP     CL, FEH
                JNZ     MONT_1
                JMP     ERROR_3
MONT_1 :        INR     CL
                MOV     ATTACH_BYTE, CL
                MOV     (SI), CL
                MOV     DI, ST_ADD_BUFFER
                MOV     CX, 0800H
                SUB     CX, DI
                SHR     CX                      , 3 TIMES
                CMP     CL, BL
                JAE     CONT_13
                JMP     ERROR_3
CONT_13 :       MOV     CH, 00H
                MOV     CL, BL
                SHL     CL                      , 3 TIMES
                SUB     CL, 04H
                MOV     BX, CX
                POP     SI
                POP     DS
                PUSH    AX
                PUSH    DI
                MOV     DI, SI_TAB
                MOV     DX, DS
                SHL     DX                      , 3 TIMES
                MOV     AX, SI
                SHR     AX
                ADD     DX, AX
                MOV     (BP+DI), DX
                ADD     DI, 0002H
                PUSH    DS
                MOV     AX, 0000H
                MOV     DS, AX
                MOV     SI_TAB, DI
                POP     DS
                POP     DI
                POP     AX
```

```
                    MOV     CL, 04H
                REP MOVSB
                    DCR     SI
                    MOV     (SI), FFH
                    INR     SI
                    MOV     CX, BX
                REP MOVSB
                    INR     DI
                    MOV     ST_ADD_BUFFER, DI
                    INR     SI
                    JMP     CONT_61
NXT_7 :             DCR     SI
                    CMP     (SI), 00H
                    JZ      CONT_14
                    JMP     ERROR_3
CONT_14 :           MOV     CL, ATTACH_BYTE
                    CMP     CL, FEH
                    JNZ     CONT_15
                    JMP     ERROR_3
CONT_15 :           INR     CL
                    MOV     ATTACH_BYTE, CL
                    MOV     (SI), CL
                    MOV     DI, ST_ADD_BUFFER
                    MOV     CX, 0800H
                    SUB     CX, DI
                    SHR     CX                  , 3 TIMES
                    CMP     CL, BL
                    JAE     CONT_13
                    JMP     ERROR_3
NULL_STATEMENT :    PUSH    DS
                    PUSH    SI
                    MOV     CX, 0000H
                    MOV     DS, CX
                    MOV     SI, BASE_ADD_LN_NO_TAB
NXT_11 :            CMP     AX, (SI)
                    JAE     NXT_10
                    INR     CX
                    SUB     SI, 0002H
                    JMP     NXT_11
NXT_10 :            MOV     DX, CX
                    MOV     CH, 00H
                    MOV     CL, PRESENT_PI
                    ADD     CL, 03H
                    SHL     CL                  , 4 TIMES
                    ADD     SI, CX
                    MOV     CL, STATUS_FOR_DET_BLK
                    SUB     CL, PRESENT_STATUS_PI
                    SHL     CL
```

```
                        SHL     CL
                        ADD     SI, CX
                        MOV     CX, EXEC_BLK_PI
                        INR     CX
                        MOV     REF_BLK, CX
                        SUB     CX, (SI)
                        MOV     BH, COUNT
NXT_13 :                CMP     DX, CX
                        JBE     NXT_12
                        DCR     BH
                        JZ      NXT_14
                        JMP     ERROR_3
NXT_14 :                SUB     DX, CX
                        SUB     SI, 0002H
                        MOV     CX, (SI)
                        INR     CX
                        MOV     REF_BLK, CX
                        SUB     SI, 0002H
                        SUB     CX, (SI)
                        JMP     NXT_13
NXT_12 :                MOV     CX, REF_BLK
                        SUB     CX, DX
                        SHL     CX                      , 4 TIMES
                        MOV     DX, CX
                        MOV     SI, 0000H
CONT_17 :               CMP     (SI), FFH
                        JNZ     CONT_18
                        ADD     SI, 0008H
                        JMP     CONT_17
CONT_18 :               INR     SI
                        CMP     AX, (SI)
                        JA      NXT_15
                        JB      NXT_16
                        ADD     SI, 0002H
                        MOV     (SI), FFH
                        JMP     NXT_16
NXT_15 :                ADD     SI, 0007H
                        JMP     CONT_17
NXT_16 :                POP     SI
                        POP     DS
                        ADD     SI, 0005H
NXT_17 :                CMP     (SI), FFH
                        JZ      NXT_18
                        JMP     CONT_61
NXT_18 :                TEST    SI, 00FFH
                        JNZ     NXT_19
                        PUSH    DI
                        PUSH    SI
```

```
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     SI, LN_NO_TAB
                        INR     AX
                        MOV     (SI), AX
                        DCR     AX
                        ADD     SI, 0002H
                        MOV     LN_NO_TAB, SI
                        POP     SI
                        POP     DS
NXT_19 :                ADD     SI, 0008H
                        JMP     NXT_17
CONT_9 :                MOV     CL, PRESENT_STATUS_PI
                        DCR     CL
                        JZ      CONT_20
                        MOV     PRESENT_STATUS_PI, CL
                        JMP     CONT_21
CONT_20 :               MOV     CX, 0000H
                        MOV     DX, CX
```

TITLE : MODULE : INITIALISE RUN ACTION LEVEL_2 F.C. 5.1

PART_2 :

ANATOMY :

CALCULATES BASE ADDRESSES OF EACH DATA TYPES.

```
                        MOV   DL, PRESENT_DI
                        CMP   DL, 00H
                        JNZ   PREP_DRT
                        JMP   NO_DATA_REF
; MODULE : INITIALISE RUN ACTION : PART_3, LEVEL_2  F.C. 5.1
PREP_DRT :              MOV   SI, BASE_ADD_DIT
                        MOV   CL, DL
                        MOV   CH, 00H
                        MOV   AX, 0000H
NXT_20 :                ADD   SI, 0002H
                        ADD   AX, (SI)
                        LOOP  NXT_20
                        MOV   BX, 03FEH
                        SUB   BX, AX
                        SHL   BX                    , 3 TIMES
                        MOV   PRESENT_DATA_SEG, BX
                        MOV   SI, BASE_ADD_DIT
                        MOV   CX, 00H
NXT_21 :                DCR   DL
                        CMP   DL, 00H
                        JZ    NXT_22
                        ADD   CL, 30H
                        JMP   NXT_21
NXT_22 :                ADD   SI, CX
                        MOV   BP, BASE_ADD_DATA_REF_TAB
                        MOV   DI, 0000H
                        ADD   SI, 0002H
                        MOV   (BP+DI), 00H
                        INR   DI
                        MOV   (BP+DI), 00H
                        INR   DI
                        MOV   DH, 00H
                        MOV   CH, 00H
                        MOV   AX, 0000H
                        MOV   DL, (SI)
                        INR   DL
                        MOV   (BP+DI), DL
                        INR   SI
```

```
MOV   DL, (SI)
MOV   CL, DL
SHL   DL
ADD   AX, DL
INR   DI
MOV   (BP+DI), AX
ADD   SI, CX
INR   SI
MOV   DL, (SI)
ADD   DI, 0002H
MOV   (BP+DI), DL
INR   SI
MOV   DL, (SI)
MOV   CL, DL
SHL   DX                    , 4 TIMES
ADD   AX, DX
INR   DI
MOV   (BP+DI), AX
ADD   SI, CX
INR   SI
MOV   DL, (SI)
ADD   DI, 0002H
MOV   (BP+DI), DL
INR   SI
MOV   DL, (SI)
MOV   CL, DL
SHL   DX                    , 5 TIMES
ADD   AX, DX
MOV   DH, 00H
INR   DI
MOV   (BP+DI), AX
ADD   SI, CX
INR   SI
MOV   DL, (SI)
ADD   DI, 0002H
MOV   (BP+DI), DL
INR   SI
MOV   DL, (SI)
MOV   CL, DL
SHL   DX                    , 8 TIMES
ADD   AX, DX
INR   DI
ADD   SI, CX
INR   SI
MOV   DL, (SI)
ADD   DI, 0002H
MOV   (BP+DI), DL
INR   SI
```

```
MOV   DL, (SI)
MOV   CL, DL
SHL   DX                        , 9 TIMES
ADD   AX, DX
INR   DI
MOV   (BP+DI), AX
ADD   SI, CX
INR   SI
MOV   DL, (SI)
ADD   DI, 0002H
INR   DL
INR   SI
MOV   DL, (SI)
MOV   CL, DL
SHL   DL
ADD   DL, CL
ADD   AX, DX
INR   DI
MOV   (BP+DI), AX
ADD   SI, CX
INR   SI
MOV   DL, (SI)
ADD   DI, 0002H
INR   DL
INR   SI
MOV   DL, (SI)
MOV   CL, DL
SHL   DX                        , 5 TIMES
ADD   AX, DX
INR   DI
MOV   (BP+DI), AX
MOV   DH, 00H
ADD   SI, CX
INR   SI
MOV   DL, (SI)
ADD   DI, 0002H
INR   DL
INR   SI
MOV   DL, (SI)
MOV   CL, DL
SHL   DL
ADD   DL, CL
SHL   DX                        , 4 TIMES
ADD   AX, DX
INR   DI
MOV   (BP+DI), AX
MOV   DH, 00H
ADD   SI, CX
```

```
INR   SI
MOV   DL, (SI)
ADD   DI, 0002H
INR   DL
INR   SI
MOV   DL, (SI)
MOV   CL, DL
SHL   DL
ADD   DL, CL
SHL   DX                          , 9 TIMES
ADD   AX, DX
INR   DI
MOV   (BP+DI), AX
MOV   DH, 00H
ADD   SI, CX
MOV   DL, (SI)
INR   DL
ADD   DI, 0002H
MOV   (BP+DI), DL
```

TITLE : MODULE : INITIALISE RUN ACTION LEVEL_2 F.C. 5.1

PART_3 : INITIATING RUN ACTION.

ANATOMY :

THE STARTING AND ENDING SI OF THE FIRST STATUS COUNT OF SPECIFIED PI IS CALCULATED, PROGRAM EXECUTION CONTINUES IN LEVEL_3 RUN. AFTER THE PROGRAM AREA CORRESPONDING TO PRESENT STATUS IS COMPLETED THE ROUTINE PROVIDES SEGMENT STARTING AND ENDING POINTERS OF NEXT STATUS COUNT.

```
NO_DATA_REF :       MOV   CL, PRESENT_PI
                    MOV   SI, BASE_ADD_PIT
                    ADD   CL, 03H
                    SHL   CL                    , 4 TIMES
                    ADD   SI, 0003H
                    MOV   DL, (SI)
                    MOV   PRESENT_STATUS_PI, DL
CONT_23 :           MOV   CL, PMTR_STATUS_DET
                    SUB   CL, PRESENT_STATUS_PI
                    SUB   SI, 0003H
                    SHL   CL
                    SHL   CL
                    ADD   SI, CX
                    MOV   CX, (SI)
                    CMP   CX, 0100H
                    JAE   CONT_22
                    SHL   CX                    , 8 TIMES
                    MOV   STARTING_SI, CX
                    ADD   SI, 0002H
                    MOV   CX, (SI)
                    CMP   CX, 0100H
                    JAE   CONT_24
                    INR   CX
                    SHL   CX                    , 8 TIMES
                    MOV   ENDING_SI, CX
                    JMP   FIRST_SENTENCE
;REFER MODULE : INTEGRITY AT RUN, LEVEL_3.
CONT_24 :           MOV   ENDING_SI, 0000H
                    MOV   MARK_SI, FFH
                    SUB   CX, 00FFH
                    SHL   CX                    , 8 TIMES
                    MOV   NXT_ENDING_SI, CX
                    JMP   FIRST_SENTENCE
```

```
CONT_22 :               MOV    MARK_DS, 01H
;TO INDICATE SECOND SEGMENT IS BEING USED.
                        SUB    CX, 0100H
                        SHL    CX                    , 8 TIMES
                        MOV    STARTING_SI, CX
                        ADD    SI, 0002H
                        MOV    CX, (SI)
                        SUB    CX, 00FFH
                        SHL    CX                    , 8 TIMES
                        MOV    ENDING_SI, CX
                        MOV    SP, BASE_STACK
                        MOV    BP, 0000H
```

TITLE : MODULE : INTEGRATION AT RUN, LEVEL_3, F.C. 5.1

ANATOMY :

THE POINTERS ARE LOADED AND ATTACH BYTE IS CHECKED FOR. THE FLOW CONTINUES IN RAM BUFFER IF A ATTACH BYTE IS LOCATED AND RETURNS AFTER EXECUTION OF THE SENTENCE. IF NO ATTACH BYTE, THEN THE SENTENCE IS EXECUTED THROUGH A NESTED CALL STRUCTURE. THE POINTER IS CONFIRMED WITHIN THE SPECIFIED AREA AND NEXT SENTENCE IS POINTED IF CHANGE IN STATUS COUNT WAS TO BE RECONCILED. THE PROGRAM FLOW LOOPS BACK FOR EXECUTION OF NEXT SENTENCE AND EXIT OUT OF LEVEL_3 OCCURS UPON ENCOUNTER OF END SENTENCE.

REGISTER USAGE :

CX : STATUS MANAGEMENT.
CL : PRESENT STATIS PI.

REFERENCE MEMORY LOCATIONS :

ENDING_SI, STARTING_SI, PRESENT_STATUS_PI, MARK_DS.

SUBROUTINE :

RSLV_RUN_GROUP_EXEC MODULE_SUB, LEVEL_3 (1), F.C.5.1

```
FIRST_SENTENCE :    MOV   CX, 0000H
                    CMP   MARK_DS, 01H
                    JNZ   CONT_25
                    MOV   CX, 1000H
CONT_25 :           MOV   DS, CX
                    MOV   SI, STARTING_SI
EXEC_8_BYTE_BLK :   CMP   (SI), 00H
                    JZ    CONT_26
                    MOV   CL, (SI)
                    MOV   CH, 00H
                    MOV   DX, 0000H
                    PUSH  SI
                    PUSH  DS
                    MOV   SI, BASE_ADD_RAM_BUFFER
                    MOV   BX, 0000H
                    MOV   DS, BX
CONT_30 :           DCR   CL
                    JZ    CONT_29
                    MOV   BL, (SI)
                    SHL   BL                    , 3 TIMES
                    ADD   SI, BX
```

```
                        JMP     CONT_30
CONT_29 :               ADD     SI, 03H
                        CALL    RSLV_RUN_GROUP_EXEC
                        POP     DS
                        POP     SI
                        INR     SI
CONT_26 :               ADD     SI, 0002H
                        CALL    RSLV_RUN_GROUP_EXEC
POINT_NXT_SENTENCE :    INR     SI
                        TEST    SI, 0007H
                        JNZ     POINT_NXT_SENTENCE
                        CMP     SI, ENDING_SI
                        JNZ     EXEC_8_BYTE_BLK
                        CMP     MARK_SI, FFH
                        JNZ     CONT_27
                        MOV     CX, NXT_ENDING_SI
                        MOV     ENDING_SI, CX
                        MOV     CX, 1000H
                        MOV     DS, CX
                        MOV     STARTING_SI, 0000H
                        JMP     EXEC_8_BYTE_BLK
CONT_27 :               MOV     CL, PRESENT_STATUS_PI
                        DCR     CL
                        JNZ     CONT_28
                        JMP     ERROR_3
CONT_28 :               MOV     PRESENT_STATUS_PI, CL
                        JMP     CONT_23
;REFER, MODULE : INITIALISE RUN ACTION, LEVEL_2, F.C. 5.1.
;PART_3.
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1), F.C. 5.1.

ANATOMY :

PROGRAM DETECTS THE COMMAND KEY ENTER AND JUMPS TO RESPECTIVE LOCATIONS FOR FURTHER EXECUTION.

REGISTER USAGE :

AL : ACTION+SUBGROUP.
BH : SUBGROUP.

```
RSLV_RUN_GROUP_EXEC : MOV    AL, (SI)
                      CMP    AL, 'END'
                      JNZ    NXT_01
                      JMP    LABLE_END_RUN
NXT_01 :              CMP    AL, 09H
                      JA     CONT_1
                      JMP    LABEL_0Q_RUN
CONT_1 :              CMP    AL, FFH
                      JNZ    CONT_2
                      JMP    LABEL_FF_RUN
CONT_2 :              MOV    BH, AL
                      AND    AL, F8H
;MASK ACTION GROUP.
                      SUB    BH, AL
                      CMP    AL, C0H
;REGISTER AL CONTENTS INDICATE ACTION GROUP
                      JA     CONT_2
                      JB     CONT_3
                      JMP    LABEL_LET_RUN
CONT_3 :              CMP    AL, A0H
                      JA     CONT_4
                      JB     CONT_5
                      JMP    LABEL_INB_RUN
CONT_5 :              CMP    AL, 90H
                      JA     CONT_6
                      JB     CONT_7
                      JMP    LABEL_DLY_RUN
CONT_7 :              CMP    AL, 88H
                      JNZ    CONT_8
                      JMP    LABEL_RET_RUN
CONT_6 :              CMP    AL, 98H
                      JNZ    CONT_8
                      JMP    LABEL_OUB_RUN
CONT_4 :              CMP    AL, B0H
                      JB     CONT_9
                      JA     CONT_A
```

```
                        JMP     LABEL_INR_RUN
CONT_9 :                CMP     AL, A8H
                        JNZ     CONT_8
                        JMP     LABEL_INW_RUN
CONT_A :                CMP     AL, B8H
                        JNZ     CONT_8
                        JMP     LABEL_OUW_RUN
CONT_8 :                JMP     ERROR_1
CONT_2 :                CMP     AL, E0H
                        JB      CONT_B
                        JA      CONT_C
                        JMP     LABEL_DSP_RUN
CONT_B :                CMP     AL, D0H
                        JB      CONT_D
                        JA      CONT_E
                        JMP     LABEL_IF_RUN
CONT_D :                CMP     AL, C8H
                        JNZ     CONT_8
                        JMP     LABEL_DCR_RUN
CONT_E :                CMP     AL, D8H
                        JNZ     CONT_8
                        JMP     LABEL_GSB_RUN
CONT_C :                CMP     AL, F0H
                        JB      CONT_F
                        JA      CONT_10
                        JMP     LABEL_GTO_RUN
CONT_F :                CMP     AL, E8H
                        JNZ     CONT_8
                        JMP     LABEL_FOR_RUN
CONT_10 :               CMP     AL, F8H
                        JNZ     CONT_8
                        JMP     LABEL_NXT_RUN
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1), F.C. 5.1.

PART_1 : RUN_ICP_END

ANATOMY :

THE SENTENCES FROM BUFFER RAM ARE BACK LOADED INTO THE DESTINATION DICTATED BY THE SI TABLE. ALL THE ATTACH BYTES ARE ZEROED. FLOW TERMINATES THE EXECUTION LEAVING THE SYSTEM AT LEVEL 0.

REGISTER USAGE :

BX : GENERATING NEXT RAM BUFFER BASE.
DX : SEGMENT MANAGEMENT.
CH : ACTION GROUP + SUBGROUP OF ATTACHED SENTENCE.

+

REGISTER USAGES AS IN PART_1 OF INITIALISATION RUN LEVEL_2, F.C. 5.1.

REFERENCE MEMORY LOCATIONS :

BASE_ADD_RAM_BUFFER, ATTACH_BYTE.

```
LABEL_END_RUN :     MOV   DI, 0000H
                    MOV   BP, BASE_SI_TAB
                    MOV   SI, BASE_RAM_BUFFER
                    MOV   CL, ATTACH_BYTE
CONT_0 :            MOV   BH, 00H
                    MOV   BL, (SI)
                    SHL   BX                        , 3 TIMES
                    ADD   BX, SI
                    MOV   BASE_ADD_RAM_BUFFER, BX
                    MOV   DX, 0000H
                    MOV   AX, (BP+DI)
                    ADD   SI, 0003H
                    MOV   CH, (SI)
                    SHL   AX
                    JNC   CONT_1
                    MOV   DX, 1000H
CONT 1 :            MOV   DS, DX
                    MOV   SI, AX
                    ADD   SI, 0003H
                    MOV   (SI), CH
                    MOV   DX, 0000H
                    MOV   DS, DX
```

```
                MOV   SI, BASE_ADD_RAM_BUFFER
                ADD   DI, 0002H
                DCR   CL
                JNZ   CONT_0
                MOV   CX, 0000H
                MOV   DS, CX
                MOV   CL, PRESENT_PI
                MOV   SI, BASE_ADD_PIT
                ADD   CL, 03H
                SHL   CL                        , 4 TIMES
                ADD   SI, 0003H
                MOV   DL, (SI)
                MOV   PRESENT_STATUS_PI, DL
CONT_2 :        MOV   CL, PMTR_STATUS_DET
                SUB   CL, PRESENT_STATUS_PI
                SUB   SI, 0002H
                SHL   CL
                SHL   CL
                ADD   SI, CX
                MOV   CX, (SI)
                CMP   CX, 0100H
                JAE   CONT_3
                SHL   CX                        , 8 TIMES
                MOV   STARTING_SI, CX
                ADD   SI, 0002H
                MOV   CX, (SI)
                CMP   CX, 0100H
                JAE   CONT_4
                INR   CX
                SHL   CX                        , 8 TIMES
                MOV   ENDING_SI, CX
                JMP   FIRST_SENTENCE_1
CONT_4 :        MOV   ENDING_SI, 0000H
                MOV   MARK_SI, FFH
                SUB   CX, 00FFH
                SHL   CX                        , 8 TIMES
                MOV   NXT_ENDING_SI, CX
                JMP   FIRST_SENTENCE_1
CONT_3 :        MOV   MARK_DS, 01H
                SUB   CX, 0100H
                SHL   CX                        , 8 TIMES
                MOV   STARTING_SI, CX
                ADD   SI, 0002H
                MOV   CX, (SI)
                SUB   CX, 00FFH
                SHL   CX                        , 8 TIMES
                MOV   ENDING_SI, CX
```

```
FIRST_SENTENCE_1 :      CMP     MARK_DS, 01H
                        JNZ     CONT_5
                        MOV     CX, 0000H
                        MOV     CH, 10H
CONT_5 :                MOV     DX, CX
                        MOV     SI, STARTING_SI
AGAIN_1 :               ADD     SI, 0008H
                        CMP     (SI), FFH
                        JZ      CONT_6
                        MOV     (SI), 00H
CONT_6 :                CMP     SI, ENDING_SI
                        JNZ     AGAIN_1
                        MOV     CL, PRESENT_STATUS_PI
                        DCR     CL
                        JNZ     CONT_7
                        JMP     LEVEL_0
CONT_7 :                MOV     PRESENT_STATUS_PI, CL
                        JMP     CONT_2
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1), F.C. 5.1.

PART_2 : RUN_ICP_RET
AND
PART_13 : RUN_ICP_GSB

ANATOMY :

THE POINTERS TO POINT A LOCATION PRIOR TO THE NEXT LINE ARE SAVED ON THE STACK (REFER MODULE : INTEGRATION AT RUN, LEVEL_3, F.C. 5.1.) THE SAVED POINTER RECONCILES WITH THE INTEGRATION MODULE WHICH TAKES CARE OF POINTING TO THE FIRST BYTE OF THE NEXT SENTENCE AFTER INCREMENTING SI. THE PROGRAM FLOW JUMPS TO LABEL_GTO_RUN TO EXECUTE THE FORWARD JUMP IN CASE OF SUBGROUP_GSB. IF THE SUBGROUP IS RET THEN THE POINTERS ARE POPPED BACK AND FLOW RETURNS TO THE INTEGRATION ROUTINE.

PART_2 :

```
LABEL_RET_RUN :     POP   DS
                    POP   SI
                    RET
```

PART_13 :

```
LABEL_GSB_RUN :     MOV   CX, SI
                    ADD   CX, 0005H
                    JNC   CONT_1
                    MOV   CX, FFFFH
                    PUSH  CX
                    MOV   CX, 1000H
                    PUSH  CX
                    JMP   CONT_2
CONT_1 :            DCR   CX
                    PUSH  CX
                    PUSH  DS
CONT_2 :            MOV   BH, 01H
                    JMP   LABEL_GTO_RUN
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1), F.C. 5.1.

PART_3 : RUN_ICP_LET

ANATOMY :

THE ROUTINE FRAGMENTS FOR TWO MAJOR SUBGROUPS VIZ.
(1) TO EXECUTE CALLS TO TARGET SUB PROGRAMS,
(2) TO EXECUTE THE EXPRESSION.
WHILE EXECUTING THE EXPRESSION THE OFFSET OF VARIABLE ON LHS IS COMPUTED AND SAVED ON THE STACK, FURTHER THE MARK INDICATING THE SEGMENT IS ALSO SAVED. THE SUBGROUPS INDICATING TYPE OF VARIABLE ON RHS MAKE DISTINCT BRANCHES.

BRANCH_1 : RHS_24R.
THE ENTRIES ON RHS COULD BE 16R OR 24R DATA TYPES. FURTHER THESE COULD BE VARIABLE REFERENCES OR NUMERIC CONSTANTS. THE APPROPRIATE VALUES ARE LOADED IN CX AND AX ARE SAVED ON THE STACK. FURTHER, SIMULTANEOUSLY OPERATORS ARE ALSO SCANNED AND TO EXECUTE OPERATIONS ON THE OPERANDS ON STACK, THE SPECIFIC SUB_PROGRAM IS CALLED. THE SUB_PROGRAM MANAGE SIGN AND EXPONENT OF THE OPERAND, TO EXECUTE ADDITION, SUBTRACTION, MULTIPLICATION, DIVISION OF THE 17 BIT OPERANDS WITHOUT VIOLATING THE MAXIMUM OCCURACY CONSTRAINT. THE ROUTINES ARE A BIT INVOLVED AND COMPLICATED.
TO ACHIEVE TRADE_OFF OF FASTNESS, THESE OCCUPY A BIT MORE SPACE. THE CONVERSION OPERATORS GETTING EXECUTED BEFORE LOADING, THE NUMBER IN THE SPECIFIED ADDRESS, IS ALSO INCLUDED AS ONE OF THE POSSIBLE OPERATORS.

BRANCH_2, _3, _4 : EXCEPT FOR REGISTER USAGE THE STRATEGY OF BRANCH_1 APPLIES TO THESE BRANCHES ALSO.
THE LENGTH OF ROUTINE DECREASES NOT BECAUSE OF OPERATORS MULTIPLICITY BUT BECAUSE OF LESSER OPERAND WIDTHS.

```
LABEL_LET_RUN :     INR   SI
                    CALL  DET_VAR_ADD          ;LEVEL_3 (1,1)
                    MOV   CX, 0000H
                    PUSH  AX
                    TEST  BL, 1CH
                    JNZ   NXT_01
                    MOV   CX, 00FFH
```

```
NXT_01 :                PUSH  CX
;REGISTER  CX  CONTENTS (00FFH), LOADED ON STACK  INDICATES  THAT
;ADDRESS DETERMINED IS FROM DEFAULT DATA AREA.
                        CMP   BH, 02H
                        JE    RSLV_24R_TYPE
                        JB    NXT_02
                        JMP   RSLV_16R_16I_8I_TYPE
NXT_02 :                JMP   RSLV_CONVERT
RSLV_24R_TYPE :         INR   SI
MACRO                   INCREMENT_SI_1
                        CMP   (SI), FFH
                        JNZ   CONT_0
                        INR   SI
CONT_0 :                NOP
                        ENDM
                        CALL  DET_VAR_ADD
                        TEST  BL, 20H
                        JNZ   VAR_FOUND
                        TEST  BL, 80H
                        JZ    VAR_FOUND
                        TEST  BL, 04H
                        JNZ   LOAD_24R_NC
                        INR   SI
                        INCREMENT_SI_1
                        MOV   AX, (SI)
MACRO                   CNVRT_16R_TO_24R
                        MOV   CX, 0000H
                        TEST  AX, 8000H
                        JZ    CONT_1
                        ADD   CL, 80H
CONT_1 :                MOV   BH, AH
                        AND   BH, 7CH
                        ADD   BH, 64H
                        SHR   BH
                        ADD   CL, 01H
                        SHL   AX                        , 7 TIMES
                        ENDM
                        PUSH  AX
                        PUSH  CX
                        JMP   START_NXT
LOAD_24R_NC :           MOV   CH, 00H
                        INR   SI
                        INCREMENT_SI_1
                        MOV   CL, (SI)
                        INR   SI
                        INCREMENT_SI_1
                        MOV   AH, (SI)
                        INR   SI
```

```
                        INCREMENT_SI_1
                        MOV     AL, (SI)
                        PUSH    AX
                        PUSH    CX
                        JMP     START_NXT
VAR_FOUND :             TEST    BL, 1CH
                        JZ      VAR_FROM_DEF
                        MOV     DX, PRESENT_DATA_SEG
                        MOV     SS, DX
                        MOV     BP, 0000H
                        MOV     DI, AX
                        TEST    CL, 04H
                        JZ      LOAD_16R_VAR
                        MOV     CH, 00H
                        MOV     CL, (BP+DI)
                        INR     DI
                        MOV     AX, (BP+DI)
LOAD_STACK :            MOV     DX, 0000H
                        MOV     SS, DX
                        PUSH    AX
                        PUSH    CX
                        JMP     START_NXT
LOAD_16R_VAR :          MOV     AX, (BP+DI)
                        CNVRT_16R_TO_24R
                        JMP     LOAD_STACK
VAR_FROM_DEF :          MOV     DX, 1FF0H
;ADDRESS OF LOCATION WHERE DEFAULT DATA AREA IS STORED.
                        MOV     SH, DX
                        JMP     LOAD_16R_VAR
START_NXT :             INR     SI
                        INCREMENT_SI_1
                        MOV     BH, (SI)
                        CMP     BH, '-'
                        JNZ     NON_UNARY
                        POP     CX
                        ADD     CL, 80H
;LOAD  MOST SIGNIFICANT BIT OF EXPONENT BYTE BY ONE  TO  INDICATE
;THE NUMER IS NEGATIVE.
                        PUSH    CX
RSLV_AGAIN_1 :          INR     SI
                        INCREMENT_SI_1
NON_UNARY :             CALL    DET_VAR_ADD
                        TEST    BL, 20H
                        JNZ     VAR_FOUND_1
                        TEST    BL, 80H
                        JZ      VAR_FOUND_1
                        TEST    BL, 04H
                        JNZ     LOAD_24R_NC_1
```

```
                        INR     SI
                        INCREMENT_SI_1
                        INR     SI
                        INCREMENT_SI_1
                        MOV     AX, (SI)
                        CNVRT_16R_TO_24R
                        PUSH    AX
                        PUSH    CX
                        JMP     START_NXT_1
LOAD_24R_NC_1 :         MOV     CH, 00H
                        INR     SI
                        INCREMENT_SI_1
                        MOV     CL, (SI)
                        INR     SI
                        INCREMENT_SI_1
                        MOV     AH, (SI)
                        INR     SI
                        INCREMENT_SI_1
                        MOV     AL, (SI)
                        PUSH    AX
                        PUSH    CX
                        JMP     START_NXT
START_NXT_1 :           MOV     DX, PRESENT_DATA_SEG
                        TEST    BL, 1CH
                        JNZ     VAR_NOT_FROM_DEF_1
                        MOV     DX, 1FF0H
VAR_NOT_FROM_DEF_1 :    MOV     SS, DX
                        MOV     BP, 0000H
                        MOV     DI, AX
                        TEST    CL, 04H
                        JZ      LOAD_16R_VAR_1
                        MOV     CH, 00H
                        MOV     CL, (BP+DI)
                        INR     DI
                        MOV     AX, (BP+DI)
LOAD_STACK_1 :          MOV     DX, 0000H
                        MOV     SS, DX
                        PUSH    AX
                        PUSH    CX
                        JMP     START_NXT_1
LOAD_16R_VAR_1 :        MOV     AX, (BP+DI)
                        CNVRT_16R_TO_24R
                        JMP     LOAD_STACK_1
START_NXT_1 :           INR     SI
                        INCREMENT_SI_1
                        MOV     BH, (SI)
                        CMP     BH, '-'
                        JA      NXT_03
```

```
                        JMP     NON_UNARY
NXT_03 :                CMP     BH, '/'
                        JBE     RSLV_OPR_1
                        CMP     BH, 80H
                        JZ      NXT_04
                        JMP     NON_UNARY
NXT_04 :                JMP     START_NXT_1
;CHECK  FOR  OPERATOR,  IF  FOUND  GO  TO  RESOLVE  AND  PERFORM
;OPERATION. IF NOT FOUND, CHECK FOR ANOTHER OPERATOR.
RSLV_OPR_1 :            CMP     BH, '-'
                        JMP     CHK_NXT_OPR_10
                        CALL    RSLV_N_24R
                        JMP     RSLV_AGAIN_1
CHK_NXT_OPR_10 :        CMP     BH, '+'
                        JMP     CHK_NXT_OPR_11
                        CALL    RSLV_P_24R
                        JMP     RSLV_AGAIN_1
CHK_NXT_OPR_11 :        CMP     BH, '*'
                        JMP     CHK_NXT_OPR_12
                        CALL    RSLV_M_24R
                        JMP     RSLV_AGAIN_1
CHK_NXT_OPR_12 :        CMP     BH, '/'
                        JMP     CHK_NXT_OPR_13
                        CALL    RSLV_D_24R
                        JMP     RSLV_AGAIN_1
CHK_NXT_OPR_13 :        POP     CX
                        POP     AX
                        MOV     DX, 1FF0H
                        POP     BX
                        CMP     BL, FFH
;CHECK WHETHER DATA IS TO BE KEPT IN THE DEFAULT DATA AREA.
                        JZ      DEF_AREA_1
                        MOV     DX, PRESENT_DATA_SEG
DEF_AREA_1 :            MOV     SS, DX
                        POP     DI
                        MOV     (BP+DI), CL
                        INR     DI
                        MOV     (BP+DI), AX
                        RET
RSLV_16R_16I_8I_TYPE:   CMP     BH, 04H
                        JBE     NXT_05
                        JMP     RSLV_8I_TYPE
NXT_05 :                JNZ     RSLV_16R_TYPE
                        JMP     RSLV_16I_TYPE
RSLV_16I_TYPE :         INR     SI
                        INCREMENT_SI_1
                        CALL    DET_VAR_ADD
                        TEST    BL, 20H
```

```
                        JNZ     VAR_FOUND_2
                        TEST    BL, 80H
                        JZ      VAR_FOUND_2
                        CMP     BL, 9AH
                        JZ      NXT_06
                        JMP     ERROR_3
NXT_06 :                INR     SI
                        INCREMENT_SI_1
                        MOV     AH, (SI)
                        INR     SI
                        INCREMENT_SI_1
                        MOV     AL, (SI)
                        PUSH    AX
                        JMP     START_NXT_2
VAR_FOUND_2 :           MOV     DX, PRESENT_DATA_SEG
                        TEST    BL, 1CH
                        JNZ     VAR_NOT_FROM_DEF_2
                        MOV     DX, 1FF0H
VAR_NOT_FROM_DEF_2 :    MOV     SS, DX
                        MOV     DI, AX
                        MOV     AX, (BP+DI)
                        MOV     DX, 0000H
                        MOV     SS, DX
                        PUSH    AX
START_NXT_2 :           INR     SI
                        INCREMENT_SI_1
                        MOV     BH, (SI)
                        CMP     BH, '-'
                        JNZ     NON_UNARY_1
                        POP     CX
                        ADD     CX, 80H
                        PUSH    CX
RSLV_AGAIN_2 :          INR     SI
                        INCREMENT_SI_1
NON_UNARY_1 :           CALL    DET_VAR_ADD
                        TEST    BL, 20H
                        JNZ     VAR_FOUND_3
                        TEST    BL, 80H
                        JZ      VAR_FOUND_3
                        CMP     BL, 0AH
                        JZ      NXT_07
                        JMP     ERROR_3
NXT_07 :                INR     SI
                        INCREMENT_SI_1
                        MOV     AH, (SI)
                        INR     SI
                        INCREMENT_SI_1
                        MOV     AL, (SI)
```

```
                        PUSH   AX
                        JMP    START_NXT_3
VAR_FOUND_3 :           MOV    DX, PRESENT_DATA_SEG
                        TEST   BL, 1CH
                        JNZ    VAR_NOT_FROM_DEF_3
                        MOV    DX, 1FF0H
VAR_NOT_FROM_DEF_3 :    MOV    SS, DX
                        MOV    DI, AX
                        MOV    AX, (BP+DI)
                        MOV    DX, 0000H
                        MOV    SS, DX
                        PUSH   AX
START_NXT_3 :           INR    SI
                        INCREMENT_SI_1
                        MOV    BH, (SI)
                        CMP    BH, '-'
                        JAE    NXT_08
                        JMP    NON_UNARY
NXT_08 :                CMP    BH, '/'
                        JBE    RSLV_OPR_2
                        CMP    BH, 82H
                        JNZ    VAR_16R_FOUND
MACRO                   CNVRT_16I_TO_16R
                        POP    AX
                        MOV    DL, AH
                        AND    DL, 80H
                        AND    AH, 7FH
                        CMP    AX, 0000H
                        JZ     CONT_3
                        MOV    DH, 10H
CONT_4 :                SHR    AX
                        JC     CONT_5
                        DCR    DH
                        JMP    CONT_4
CONT_5 :                ADD    DH, 10H
                        SHL    DH
                        SHL    DH
                        ADD    DL, DH
                        ROR    AX
                        MOV    CX, 0006H
                        SHR    AX, CX
                        ADD    AH, DL
CONT_3 :                PUSH   AX
                        ENDM
NXT_09 :                JMP    START_NXT_3
VAR_16R_FOUND :         CMP    BH, 81H
                        JZ     NXT_09
                        JMP    RSLV_AGAIN_2
```

```
RSLV_OPR_2 :            CMP     BH, '-'
                        JMP     CHK_NXT_OPR_20
                        CALL    RSLV_N_16R
                        JMP     RSLV_AGAIN_2
CHK_NXT_OPR_20 :        CMP     BH, '+'
                        JMP     CHK_NXT_OPR_21
                        CALL    RSLV_P_16R
                        JMP     RSLV_AGAIN_2
CHK_NXT_OPR_21 :        CMP     BH, '*'
                        JMP     CHK_NXT_OPR_22
                        CALL    RSLV_M_16R
                        JMP     RSLV_AGAIN_2
CHK_NXT_OPR_22 :        CMP     BH, '/'
                        JMP     CHK_NXT_OPR_23
                        CALL    RSLV_D_16R
                        JMP     RSLV_AGAIN_2
CHK_NXT_OPR_23 :        POP     CX
                        POP     AX
                        MOV     DX, 1FF0H
                        POP     BX
                        CMP     BL, FFH
;CHECK WHETHER DATA IS TO BE KEPT IN THE DEFAULT DATA AREA.
                        JZ      DEF_AREA_2
                        MOV     DX, PRESENT_DATA_SEG
DEF_AREA_2 :            MOV     SS, DX
                        POP     DI
                        MOV     (BP+DI), AX
                        RET
RSLV_16I_TYPE :         INR     SI
                        INCREMENT_SI_1
                        CALL    DET_VAR_ADD
                        TEST    BL, 20H
                        JNZ     VAR_FOUND_2
                        TEST    BL, 80H
                        JZ      VAR_FOUND_2
                        CMP     BL, 04H
                        JNZ     LOAD_16I_NC
                        CMP     BL, 9BH
                        JZ      NXT_06
                        JMP     ERROR_3
NXT_06 :                INR     SI
                        INCREMENT_SI_1
                        MOV     AL, (SI)
MACRO                   CNVRT_8I_TO_16I
                        POP     AX
                        MOV     DL, AL
                        AND     DL, 80H
                        MOV     AH, DL
```

```
                        PUSH   AX
                        ENDM
                        PUSH   AX
                        JMP    START_NXT_4
LOAD_16I_NC :           INR    SI
                        INCREMENT_SI_1
                        MOV    AH, (SI)
                        INR    SI
                        INCREMENT_SI_1
                        MOV    AL, (SI)
                        PUSH   AX
                        JMP    START_NXT_4
VAR_FOUND_4 :           MOV    DX, PRESENT_DATA_SEG
                        TEST   BL, 1CH
                        JZ     VAR_NOT_FROM_DEF_4
                        MOV    DX, 1FF0H
VAR_NOT_FROM_DEF_4 :    MOV    SS, DX
                        MOV    DI, AX
                        MOV    AX, (BP+DI)
                        MOV    DX, 0000H
                        MOV    SS, DX
                        PUSH   AX
START_NXT_4 :           INR    SI
                        INCREMENT_SI_1
                        MOV    BH, (SI)
                        CMP    BH, '-'
                        JNZ    NON_UNARY_3
                        POP    CX
                        ADD    CH, 80H
                        PUSH   CX
RSLV_AGAIN_3 :          INR    SI
                        INCREMENT_SI_1
NON_UNARY_3 :           CALL   DET_VAR_ADD
                        TEST   BL, 20H
                        JNZ    VAR_FOUND_5
                        TEST   BL, 80H
                        JZ     VAR_FOUND_5
                        CMP    BL, 04H
                        JNZ    LOAD_16I_NC_1
                        CMP    BL, 9BH
                        JZ     NXT_07
                        JMP    ERROR_3
NXT_07 :                INR    SI
                        INCREMENT_SI_1
                        MOV    AL, (SI)
                        CNVRT_8I_TO_16I
                        PUSH   AX
                        JMP    START_NXT_5
```

```
LOAD_16I_NC_1 :     INR    SI
                    INCREMENT_SI_1
                    MOV    AH, (SI)
                    INR    SI
                    INCREMENT_SI_1
                    MOV    AL, (SI)
                    PUSH   AX
                    JMP    START_NXT_5
VAR_FOUND_5 :       MOV    DX, PRESENT_DATA_SEG
                    TEST   BL, 1CH
                    JZ     VAR_NOT_FROM_DEF_5
                    MOV    DX, 1FFOH
VAR_NOT_FROM_DEF_5 : MOV   SS, DX
                    MOV    DI, AX
                    TEST   BL, 04H
                    JNZ    LOAD_16I_VAR_1
                    MOV    AL, (BP+DI)
                    CNVRT_8I_TO_16I
                    PUSH   AX
                    JMP    START_NXT_5
LOAD_16I_VAR_1 :    MOV    AX, (BP+DI)
                    PUSH   AX
START_NXT_5 :       INR    SI
                    INCREMENT_SI_1
                    MOV    BH, (SI)
                    CMP    BH, '-'
                    JAE    NXT_OD
                    JMP    NON_UNARY_3
NXT_OD :            CMP    BH, '/'
                    JBE    RSLV_OPR_3
                    CMP    BH, 83H
                    JNZ    NXT_OE
                    JMP    START_NXT_5
NXT_OE :            CMP    BH, 84H
                    JNZ    NXT_OF
MACRO               CNVRT_16R_TO_16I
                    POP    AX
                    MOV    CH, AH
                    AND    CH, 80H
                    MOV    CL, AH
                    AND    CL, 74H
                    AND    AH, 03H
                    CMP    CL, 1AH
                    JB     CONT_7
                    SUB    CL, 1AH
CONT_6 :            CMP    CL, DX
                    JZ     CONT_8
                    SHL    AX
```

```
                        DCR     DL
                        JMP     CONT_6
CONT_7 :                CMP     CL, 10H
                        JB      CONT_8
                        MOV     DL, 1AH
                        SUB     DL, CL
CONT_9 :                SHR     AX
                        DCR     DL
                        JNZ     CONT_9
CONT_8 :                ADD     AH, CH
                        PUSH    AX
                        ENDM
                        JMP     START_NXT_5
NXT_OF :                CMP     BH, 85H
                        JZ      NXT_10
                        JMP     RSLV_AGAIN_3
MACRO                   CNVRT_16R_TO_24R
                        POP     AX
                        MOV     DL, DH
                        AND     DL, 80H
                        AND     AH, 7FH
                        CMP     AX, 0000H
                        JZ      CONT_10
                        MOV     DH, 10H
CONT_11 :               SHR     AX
                        JC      CONT_12
                        DCR     DH
                        JMP     CONT_11
CONT_12 :               ADD     CL, 81H
                        ADD     DH, 32H
                        SHL     DH
                        ADD     DL, DH
                        ADD     CL, DL
                        MOV     CH, 00H
                        PUSH    AX
                        PUSH    CX
                        ENDM
                        JMP     RSLV_AGAIN_3
RSLV_OPR_3 :            CMP     BH, '-'
                        JMP     CHK_NXT_OPR_30
                        CALL    RSLV_N_16I
                        JMP     RSLV_AGAIN_3
CHK_NXT_OPR_30 :        CMP     BH, '+'
                        JMP     CHK_NXT_OPR_31
                        CALL    RSLV_P_16I
                        JMP     RSLV_AGAIN_3
CHK_NXT_OPR_31 :        CMP     BH, '*'
                        JMP     CHK_NXT_OPR_32
```

```
                        CALL    RSLV_M_16I
                        JMP     RSLV_AGAIN_3
CHK_NXT_OPR_32 :        CMP     BH, '/'
                        JNZ     CHK_NXT_OPR_34
                        CALL    RSLV_D_16I
                        JMP     RSLV_AGAIN_3
CHK_NXT_OPR_34 :        CMP     BL, 07H
                        JZ      LOAD_24R
                        POP     AX
                        MOV     DX, 1FF0H
                        POP     BX
                        CMP     BL, FFH
;CHECK WHETHER DATA IS TO BE KEPT IN THE DEFAULT DATA AREA.
                        JZ      NXT_12
                        MOV     DX, PRESENT_DATA_SEG
NXT_12 :                MOV     SS, DX
                        POP     DI
                        MOV     (BP+DI), AX
                        RET
LOAD_24R :              POP     CX
                        POP     AX
                        MOV     DX, 0000H
                        POP     DI
                        MOV     SS, DX
                        MOV     (BP+DI), CL
                        INR     DI
                        MOV     (BP+DI), AX
                        RET
RSLV_8I_TYPE :          INR     SI
                        INCREMENT_SI_1
                        CALL    DET_VAR_ADD
                        TEST    BL, 20H
                        JNZ     VAR_FOUND_6
                        TEST    BL, 80H
                        JZ      VAR_FOUND_6
                        CMP     BL, 9CH
                        JZ      NXT_06
                        JMP     ERROR_3
NXT_06 :                INR     SI
                        INCREMENT_SI_1
                        MOV     AL, (SI)
                        PUSH    AX
                        JMP     START_NXT_6
VAR_FOUND_6 :           MOV     DX, PRESENT_DATA_SEG
                        TEST    BL, 1CH
                        JZ      VAR_NOT_FROM_DEF_5
                        MOV     DX, 1FF0H
```

```
VAR_NOT_FROM_DEF_5 :    MOV     SS, DX
                        MOV     DI, AX
                        MOV     AX, (BP+DI)
                        MOV     DX, 0000H
                        MOV     SS, DX
                        PUSH    AX
START_NXT_6 :           INR     SI
                        INCREMENT_SI_1
                        MOV     BH, (SI)
                        CMP     BH, '-'
                        JNZ     NON_UNARY_4
                        POP     CX
                        ADD     CL, 80H
                        PUSH    CX
RSLV_AGAIN_4 :          INR     SI
                        INCREMENT_SI_1
NON_UNARY_4 :           CALL    DET_VAR_ADD
                        TEST    BL, 20H
                        JNZ     VAR_FOUND_7
                        TEST    BL, 80H
                        JZ      VAR_FOUND_7
                        CMP     BL, 9CH
                        JZ      NXT_12
                        JMP     ERROR_3
NXT_12 :                INR     SI
                        INCREMENT_SI_1
                        MOV     AL, (SI)
                        MOV     AH, 00H
                        PUSH    AX
                        JMP     START_NXT_7
VAR_FOUND_7 :           MOV     DX, PRESENT_DATA_SEG
                        TEST    BL, 1CH
                        JZ      VAR_NOT_FROM_DEF_6
                        MOV     DX, 1FF0H
VAR_NOT_FROM_DEF_6 :    MOV     SS, DX
                        MOV     DI, AX
                        MOV     AL, (BP+DI)
                        MOV     AH, 00H
                        MOV     DX, 0000H
                        MOV     SS, DX
                        PUSH    AX
START_NXT_7 :           INR     SI
                        INCREMENT_SI_1
                        MOV     BH, (SI)
                        CMP     BH, '-'
                        JAE     NXT_13
                        JMP     NON_UNARY_4
NXT_13 :                CMP     BH, '/'
```

```
                        JBE     RSLV_OPR_4
                        CMP     BH, 86H
                        JZ      START_NXT_7
                        JMP     NON_UNARY_4
RSLV_OPR_4 :            CMP     BH, '-'
                        JMP     CHK_NXT_OPR_40
                        CALL    RSLV_N_8I
                        JMP     RSLV_AGAIN_4
CHK_NXT_OPR_40 :        CMP     BH, '+'
                        JMP     CHK_NXT_OPR_41
                        CALL    RSLV_P_8I
                        JMP     RSLV_AGAIN_4
CHK_NXT_OPR_41 :        CMP     BH, '*'
                        JMP     CHK_NXT_OPR_42
                        CALL    RSLV_M_8I
                        JMP     RSLV_AGAIN_4
CHK_NXT_OPR_42 :        CMP     BH, '/'
                        JNZ     CHK_NXT_OPR_44
                        CALL    RSLV_D_8I
                        JMP     RSLV_AGAIN_4
CHK_NXT_OPR_44 :        POP     AX
                        MOV     DX, 1FF0H
                        POP     BX
                        CMP     BL, FFH
;CHECK WHETHER DATA IS TO BE KEPT IN THE DEFAULT DATA AREA.
                        JZ      NXT_14
                        MOV     DX, PRESENT_DATA_SEG
NXT_14 :                MOV     SS, DX
                        POP     DI
                        MOV     (BP+DI), AX
                        RET
RSLV_P_24I :            POP     CX
                        POP     AX
                        MOV     BP, SP
                        DCR     BP
                        MOV     CH, (BP)
                        MOV     DX, AX
                        AND     DX, 7F7FH
                        CMP     DH, DL
                        JAE     NXT_20
                        JMP     EXP_2_GTHAN_1
NXT_20 :                JE      NXT_21
                        JMP     EXP_1_GTHAN_2
NXT_21 :                MOV     DX, CX
                        AND     DX, 8080H
                        XOR     DH, DL
                        JNZ     SIGN_UNEQUAL_1
SIGN_EQUAL_1 :          CLC
```

```
                        SUB     SP, 0002H
                        POP     BX
                        ADD     AX, BX
                        ROR     AX
                        MOV     DL, CL
                        AND     DL, 7FH
                        CMP     DL, 7FH
                        JB      NXT_22
                        JMP     OVER_FLOW
;THE ERROR MESSAGE INDICATING OCCURANCE OF OVER FLOW.
NXT_22 :                ADD     CL, 02H
                        MOV     CH, 00H
                        PUSH    AX
                        PUSH    CX
                        RET
SIGN_UNEQUAL_1 :        STC
                        SUB     SP, 0002H
                        POP     BX
                        SUB     AX, BX
                        JNB     NC_1_POS
                        NEG     AX
                        ADD     CL, 80H
NC_1_POS :              CMP     AX, 0000H
                        JNZ     NXT_22
                        JMP     UNDER_FLOW
;THE ERROR MESSAGE INDICATING OCCURANCE OF UNDER FLOW.
NXT_22 :                MOV     CH, CL
                        AND     CH, 7EH
                        SHR     CH
                        INR     CH
AGAIN_SHIFT_1 :         SHL     AX
                        JC      STOP_SHIFT_1
                        DCR     CH
                        JNZ     NXT_23
                        JMP     UNDER_FLOW
NXT_23 :                JMP     AGAIN_SHIFT_1
STOP_SHIFT_1 :          AND     CL, 81H
                        SHL     CH
                        ADD     CH, CL
                        MOV     CL, CH
                        MOV     CH, 00H
                        PUSH    AX
                        PUSH    CX
                        RET
EXP_2_GTHAN_1 :         SHR     DH
                        SHR     DL
                        AND     CL, FEH
CHK_EXP_AGAIN_1 :       INR     DL
```

```
                        CMP     DL, 3FH
                        JBE     NXT_24
                        JMP     OVER_FLOW
NXT_24 :                CMP     DH, BL
                        JZ      EXP_BECOME_EQUAL_1
                        SHR     AX
                        JMP     CHK_EXP_AGAIN_1
EXP_BECOME_EQUAL_1 :    SHL     DL
                        ADD     CL, DL
                        MOV     CH, 00H
                        JMP     DO_ADDITION_1
EXP_1_GTHAN_2 :         MOV     CH, 00H
                        POP     BX
                        POP     DX
                        PUSH    AX
                        PUSH    CX
                        MOV     AX, DX
                        MOV     CX, BX
                        MOV     CH, (BP)
                        MOV     DX, CX
                        AND     DX, 7F7FH
                        JMP     EXP_2_GTHAN_1
DO_ADDITION_1 :         MOV     CH, (BP)
                        MOV     DX, CX
                        AND     DX, 8080H
                        XOR     DX, DL
                        JNZ     LOAD_STACK_1
DO_AGAIN_1 :            CLC
                        SUB     SP, 0002H
                        POP     BX
                        ADD     AX, BX
                        JNC     LOAD_STACK_2
                        SHR     AX
                        MOV     DL, CL
                        AND     DL, 7FH
                        CMP     DL, 7FH
                        JBE     NXT_25
                        JMP     OVER_FLOW
LOAD_STACK_2 :          MOV     CH, 00H
                        PUSH    AX
                        PUSH    CX
                        RET
LOAD_STACK_1 :          SUB     SP, 0002H
                        POP     BX
                        SUB     AX, BX
                        MOV     CH, 00H
                        PUSH    AX
                        PUSH    CX
```

```
                        RET
RSLV_N_24_R :           POP     CX
                        POP     AX
                        MOV     BP, SP
                        DCR     BP
                        MOV     CH, (BP)
                        MOV     DX, CX
                        AND     DX, 7F7FH
                        CMP     DH, DL
                        JA      EXP_2_GTHAN_1_1
                        JE      EXP_EQUAL_2
                        JMP     EXP_1_GTHAN_2_1
EXP_EQUAL_2 :           MOV     DX, CX
                        AND     DX, 8080H
                        XOR     DH, DL
                        JNZ     NXT_26
                        JMP     SIGN_UNEQUAL_1
NXT_26 :                JMP     SIGN_EQUAL_1
EXP_2_GTHAN_1_1 :       SHR     DH
                        SHR     DL
                        AND     CL, FEH
                        INR     DL
CHK_AGAIN_EXP_2 :       CMP     DL, 3FH
                        JBE     NXT_27
                        JMP     OVER_FLOW
NXT_27 :                CMP     DH, DL
                        JZ      EXP_EQUAL_3
                        SHR     AX
                        INR     DL
                        JMP     CHK_AGAIN_EXPT_2
EXP_EQUAL_3 :           SHL     DL
                        ADD     CL, DL
                        MOV     CH, 00H
                        JMP     DO_ADDITION_2
EXP_1_GTHAN_2_1 :       MOV     CH, 00H
                        POP     BX
                        POP     DX
                        PUSH    AX
                        PUSH    CX
                        MOV     AX, DX
                        MOV     CX, BX
                        MOV     CH, (BP)
                        MOV     DX, CX
                        AND     DX, 7F7FH
                        JMP     EXP_2_GTHAN_1_1
DO_ADDITION_2 :         MOV     [illegible]
                        MOV     DX, CX
                        AND     DX, 8080H
```

```
                        XOR     DH, DL
                        JZ      NXT_27
                        JMP     DO_AGAIN_1
NXT_27 :                JMP     LOAD_STACK_1
RSLV_P_16R :            POP     BX
                        POP     AX
                        MOV     DL, AH
                        MOV     DH, BH
                        MOV     CX, DX
                        AND     DX, 7C7CH
                        CMP     DH, DL
                        JBE     NXT_1
                        JMP     EXP_2_GTHAN_1
NXT_1 :                 JE      EXP_EQUAL
                        JMP     EXP_1_GTHAN_2
EXP_EQUAL :             MOV     DX, CX
                        ADD     DX, 8080H
                        XOR     DH, DL
                        JNZ     SIGN_UNEQUAL
SIGN_EQUAL :            AND     AH, 03H
                        AND     BH, 03H
                        ADD     AX, BX
                        TEST    AH, 04H
                        JZ      NXT_2
                        MOV     DL, CL
                        AND     DL, 7CH
                        JNZ     NXT_3
                        JMP     OVER_FLOW
NXT_3 :                 SHR     AX
                        SUB     CL, 04H
NXT_2 :                 ADD     AH, BL
                        DCR     SP
                        RET
SIGN_UNEQUAL :          STC
                        AND     AH, 03H
                        AND     BH, 03H
                        SUB     AX, BX
                        JNB     NC_1_GTHAN_NC_2
                        NEG     AX
                        ADD     CL, 80H
NC_1_GTHAN_NC_2 :       CMP     AX, 0000H
                        JNZ     NXT_4
                        JMP     UNDER_FLOW
NXT_4 :                 MOV     CH, CL
                        AND     CH, 7CH
                        SHR     CH
                        SHR     CH
                        INR     CH
```

```
AGAIN_SHIFT             SHL     AX
                        TEST    AH, 40H
                        JZ      NC_ADJUSTED
                        DCR     CH
                        JNZ     NXT_5
                        JMP     UNDER_FLOW
                        JMP     AGAIN_SHIFT
NC_ADJUSTED :           SHL     CH
                        SHL     CH
                        ADD     AH, CH
                        DCR     SP
                        PUSH    AX
                        RET
EXP_2_GTHAN_1 :         AND     DX, 7F7FH
                        SHR     DH
                        SHR     DH
                        SHR     DL
                        SHR     DL
                        AND     AL, 03H
                        INR     DL
AGAIN_SHIFT_1 :         CMP     DL, 1FH
                        JBE     NXT_6
                        JMP     OVER_FLOW
NXT_6 :                 CMP     DH, DL
                        JZ      EXP_MATCH
                        SHR     AX
                        INR     DL
                        JMP     AGAIN_SHIFT_1
EXP_MATCH :             SHL     DL
                        SHL     DL
                        ADD     AH, DL
                        JMP     CHK_SIGN
EXP_1_GTHAN_2 :         XCHG    AX, BX
                        XCHG    DH, DL
                        JMP     EXP_2_GTHAN_1
CHK_SIGN :              AND     DX, 8080H
                        XOR     DH, DL
                        JNZ     UNEQUAL_SIGN
EQUAL_SIGN :            AND     AH, 03H
                        AND     BH, 03H
                        ADD     AX, BX
                        CMP     AH, 04H
                        JZ      EXP_NO_CHANGE
                        MOV     DL, CL
                        AND     DL, 7CH
                        JNZ     NXT_7
                        JMP     OVER_FLOW
NXT_7 :                 SHR     AX
```

```
                        SUB     DL, 04H
EXP_NO_CHANGE :         ADD     AH, DL
                        DCR     SP
                        PUSH    AX
                        RET
UNEQUAL_SIGN :          AND     AH, 03H
                        AND     BH, 03H
                        SUB     AX, BX
                        ADD     AH, DH
                        DCR     SP
                        PUSH    AX
                        RET
RSLV_N_16R :            POP     BX
                        POP     AX
                        MOV     DL, AH
                        MOV     DH, AL
                        MOV     CX, DX
                        AND     DX, 7C7CH
                        CMP     DH, DL
                        JA      EXP_2_GTHAN_1_1
                        JB      EXP_1_GTHAN_2_1
                        AND     DX, 8080H
                        XOR     DH, DL
                        JNZ     NXT_8
                        JMP     SIGN_EQUAL
 NXT_8 :                JMP     SIGN_UNEQUAL
EXP_2_GTHAN_1_1 :       AND     DX, 7C7CH
                        SHR     DH
                        SHR     DH
                        SHR     DL
                        SHR     DL
                        AND     AH, 03
                        INR     DL
AGAIN_COMPARE :         CMP     DL, 1FH
                        JBE     NXT_9
                        JMP     OVER_FLOW
NXT_9 :                 CMP     DH, DL
                        JZ      EXP_MATCH_1
                        SHR     AX
                        INR     DL
                        JMP     AGAIN_COMPARE
EXP_MATCH_1 :           SHL     DX
                        SHL     DX
                        ADD     AH, AL
                        JMP     CHK_SIGN_2
EXP_1_GTHAN_2_1 :       XCHG    AX, BX
                        XCHG    DH, DL
                        JMP     EXP_2_GTHAN_1_1
```

```
CHK_SIGN_2 :          AND    DX, 8080H
                      XOR    DX, DL
                      JZ     NXT_A
                      JMP    UNEQUAL_SIGN
NXT_A :               JMP    EQUAL_SIGN
RSLV_M_24R :          POP    CX
                      POP    AX
                      MOV    BP, SP
                      DCR    SP
                      MOV    CH, (BP)
                      MOV    DX, CX
                      AND    DX, 8080H
                      XOR    DH, DL
                      JNZ    SIGN_UNEQUAL
                      MOV    DX, CX
                      TEST   DL, 80H
                      JZ     EXP_BOTH_NEG
                      MOV    DX, CX
                      AND    DX, 7E7EH
                      AND    DH, 3EH
                      ADD    DH, DL
                      CMP    DH, 50H
                      JBE    NXT_1
                      JMP    OVER_FLOW
                      MOV    CH, DH
                      JMP    MUL_24R
EXP_BOTH_NEG :        MOV    DX, CX
                      AND    DX, 7E7EH
                      ADD    DH, E0H
                      NEG    DH
                      CMP    DH, DL
                      JB     NXT_2
                      JMP    OVER_FLOW
NXT_2 :               SUB    DL, DH
                      CMP    DL, 0CH
                      JA     NXT_3
                      JMP    ERROR_3
NXT_3 :               MOV    CH, DL
                      JMP    MUL_24R
SIGN_UNEQUAL :        TEST   DH, 80H
                      JNZ    NC_2_NEG
                      MOV    DX, CX
                      AND    DX, 7E7EH
                      ADD    DL, E0H
                      NEG    DL
                      CMP    DL, DH
                      JB     NXT_4
                      JMP    OVER_FLOW
```

```
NXT_4 :          SUB   DH, DL
                 CMP   DH, 0CH
                 JA    NXT_5
                 JMP   ERROR_3
NXT_5 :          MOV   CH, DH
                 JMP   MUL_24R
NC_2_NEG :       MOV   DX, CX
                 AND   DX, 7E7EH
                 ADD   DH, A0H
                 NEG   DH
                 SUB   DL, DH
                 CMP   DL, 0CH
                 JA    NXT_6
                 JMP   ERROR_3
NXT_6 :          MOV   CH, DL
MUL_24R :        SUB   SP, 0002H
                 POP   BX
                 PUSH  AX
                 MUL   AX, DX
                 POP   AX
                 ADD   AX, BX
                 JC    ADJUST_NC_1
                 ADD   DX, AX
                 JC    ADJUST_NC_2
                 JMP   LOAD_STACK_2
ADJUST_NC_1 :    ADD   DX, AX
                 JC    LOAD_STACK_1
ADJUST_NC_2 :    SHR   DX
                 MOV   CL, CH
                 SUB   CL, 11H
                 MOV   CH, 00H
                 PUSH  DX
                 PUSH  CX
                 RET
LOAD_STACK_1 :   SHR   DX
                 ADD   DH, 80H
                 MOV   CL, CH
                 MOV   CH, 00H
                 SUB   CL, 11H
                 PUSH  DX
                 PUSH  CX
                 RET
LOAD_STACK_2 :   MOV   CL, CH
                 MOV   CH, 00H
                 SUB   CL, 10H
                 PUSH  DX
                 PUSH  CX
                 RET
```

```
RSLV_M_16R :        POP    BX
                    POP    AX
                    MOV    DL, AH
                    MOV    DH, BH
                    MOV    CX, DX
                    AND    DX, 8080H
                    XOR    DH, DL
                    JNZ    UNEQUAL_SIGN
                    TEST   DL, 80H
                    JNZ    NC_BOTH_NEG
                    MOV    DX, CX
                    AND    DX, 7C7CH
                    AND    DH, 3CH
                    ADD    DH, DL
                    CMP    DH, 40H
                    JBE    NXT_1
                    JMP    ERROR_3
NXT_1 :             MOV    CL, DH
                    JMP    MUL_16R
NC_BOTH_NEG :       MOV    DX, AX
                    AND    DX, 7C7CH
                    ADD    DH, F0H
                    NEG    DH
                    CMP    DH, DL
                    JB     NXT_2
                    JMP    ERROR_3
NXT_2 :             SUB    DL, DH
                    CMP    DL, 06H
                    JA     NXT_3
                    JMP    ERROR_3
NXT_3 :             MOV    CL, DL
                    JMP    MUL_16R
UNEQUAL_SIGN :      TEST   DH, 80H
                    JNZ    NC_1_POS_2_NEG
                    MOV    DX, CX
                    AND    DX, 7C7CH
                    ADD    DL, F0H
                    NEG    DL
                    CMP    DL, DH
                    JB     NXT_4
                    JMP    ERROR_3
                    SUB    DH, DL
                    CMP    DH, 06H
                    JA     NXT_5
                    JMP    ERROR_3
NXT_5 :             MOV    CL, DH
                    JMP    MUL_16R
NC_1_POS_2_NEG :    MOV    DX, CX
```

```
                        AND     DX, 7C7CH
                        ADD     DH, F0H
                        NEG     DH
                        CMP     BL, DH
                        JAE     NXT_6
                        JMP     ERROR_3
NXT_6 ;                 MOV     CL, DL
MUL_16R :               AND     AH, 03H
                        AND     BH, 03H
                        MUL     AX, BX
                        TEST    DL, 01H
                        JNZ     ADJUST_EXP_3
                        TEST    DL, 02H
                        JNZ     ADJUST_EXP_2
                        TEST    DL, 04H
                        JNZ     ADJUST_EXP_1
                        MOV     DH, CL
                        SUB     BH, 0AH
                        MOV     CX, 000AH
ADJUST_NC_FORMAT :      SHR     DL
                        ROR     AX
                        LOOP    ADJUST_NC_FORMAT
                        SHL     BH
                        SHL     BH
                        ADD     AH, BH
                        PUSH    AX
                        RET
ADJUST_EXP_1 :          MOV     BH, CL
                        SUB     BH, 09H
                        MOV     CX, 0009H
                        JMP     ADJUST_NC_FORMAT
ADJUST_EXP_2 :          MOV     BH, CL
                        SUB     BH, 08H
                        MOV     CX, 0008H
                        JMP     ADJUST_NC_FORMAT
ADJUST_EXP_3 :          MOV     BH, CL
                        SUB     BH, 07H
                        MOV     CX, 0001H
                        JMP     ADJUST_FORMAT
RSLV_D_16R :            POP     BX
                        POP     AX
                        MOV     DL, AH
                        MOV     DH, BH
                        MOV     CX, DX
                        AND     DX, 8080H
                        XOR     DH, DL
                        JNZ     UNEQUAL_SIGN
                        TEST    DH, 80H
```

```
                        JNZ     BOTH_NEG_SIGN
                        MOV     DX, CX
                        AND     DX, 7C7CH
                        AND     DH, 3CH
                        SUB     DL, DH
                        CMP     DL, 06H
                        JA      NXT_1
                        JMP     ERROR_3
NXT_1 :                 MOV     CL, DL
                        JMP     DIVIDE_16R
BOTH_NEG_SIGN :         AND     DX, 7C7CH
                        ADD     DH, F0H
                        NEG     DH
                        ADD     DL, DH
                        CMP     DL, 06H
                        JA      NXT_2
                        JMP     ERROR_3
NXT_2 :                 MOV     CL, DL
                        JMP     DIVIDE_16_R
UNEQUAL_SIGN :          TEST    DH, 80H
                        JNZ     NC_1_POS_2_NEG
                        MOV     DX, AX
                        AND     DX, 7C7CH
                        AND     DH, 3CH
                        SUB     DL, DH
                        CMP     DL, DH
                        JA      NXT_3
                        JMP     ERROR_3
NXT_3 :                 SUB     DL, DH
                        CMP     DL, 06H
                        JA      NXT_4
                        JMP     ERROR_3
NXT_4 :                 MOV     CL, DL
                        JMP     DIVIDE_16R
NC_1_POS_2_NEG :        MOV     DX, CX
                        AND     DX, 7C7CH
                        ADD     DH, F0H
                        NEG     DH
                        ADD     DL, DH
DIVIDE_16R :            MOV     DH, CL
                        MOV     CX, 000AH
AGAIN_SHIFT :           SHR     AX
                        ROR     DL
                        LOOP    AGAIN_SHIFT
                        MOV     CL, DH
                        DIV     DX, BX
;DIVIDEND  IS WITH LOGICAL 1 AT TWENTIETH BIT AND DIVISER  IS  OF
;10 BITS ONLY.
```

```
                        CMP     AH, 08H
                        JNZ     ADJUST_EXP_1
                        CMP     AH, 04H
                        JNZ     ADJUST_EXP_2
                        SUB     CL, 0AH
                        ADD     AH, CL
                        PUSH    AX
                        RET
ADJUST_EXP_1 :          SUB     CL, 08H
                        ADD     AH, AL
                        PUSH    AX
                        RET
ADJUST_EXP_2 :          SUB     CL, 09H
                        ADD     AH, AL
                        PUSH    AX
                        RET
RSLV_D_24R :            POP     CX
                        POP     AX
                        MOV     BP, SP
                        DCR     BP
                        MOV     CH, (BP)
                        MOV     DX, CX
                        AND     DX, 8080H
                        XOR     DH, DL
                        JNZ     UNEQUAL_SIGN
                        TEST    DL, 80H
                        JNZ     BOTH_NEG_SIGN
                        MOV     DX, CX
                        AND     DX, 7E7EH
                        AND     DH, 3EH
                        SUB     DL, DH
                        CMP     DL, 0CH
                        JA      NXT_1
                        JMP     ERROR_3
NXT_1 :                 MOV     CL, DL
                        JMP     DIVIDE_24R
BOTH_NEG_SIGN :         MOV     DX, CX
                        AND     DX, 7E7EH
                        ADD     DH, E0H
                        NEG     DH
                        ADD     DH, DL
                        CMP     DH, 0CH
                        JA      NXT_2
                        JMP     ERROR_3
NXT_2 :                 MOV     CL, DH
                        JMP     DIVIDE_24R
UNEQUAL_SIGN :          TEST    DH, 80H
                        JNZ     NC_1_POS_2_NEG
```

```
                        MOV     DX, CX
                        AND     DX, 7E7EH
                        AND     DH, 3EH
                        CMP     DL, DH
                        JA      NXT_3
                        JMP     ERROR_3
NXT_3 :                 SUB     DL, DH
                        CMP     DL, 0CH
                        JA      NXT_4
                        JMP     ERROR_3
NXT_4 :                 MOV     CL, DL
                        JMP     DIVIDE_24R
NC_1_POS_2_NEG :        MOV     DX, CX
                        ADD     DH, E0H
                        NEG     DH
                        ADD     DH, DL
                        CMP     DH, 4BH
                        JB      NXT_5
                        JMP     ERROR_3
NXT_5 :                 MOV     CL, DH
DIVIDE_24R :            DCR     SP, 0002H
                        POP     BX
                        SHR     BX
                        ADD     BH, 80H
                        MOV     DX, 0000H
                        SHR     AX
                        ROR     CH
                        SHR     AX
                        ROR     CH
                        MOV     DH, 40H
                        ADD     DX, AX
                        MOV     AH, CH
                        MOV     CH, 00H
                        PUSH    DS
                        PUSH    CX
                        MOV     CX, 0000H
                        MOV     DS, CX
                        POP     CX
                        MOV     LOW_WORD_1, 0000H
                        SHR     BX
                        MOV     UP_WORD_1, BX
                        JNC     LOAD_WORD_2
                        MOV     LOW_WORD_1, 8000H
LOAD_WORD_2 :           ROL     BX
                        DCR     BX
                        MOV     UP_WORD_2, BX
                        INR     BX
                        NEG     BX
```

```
                        MOV     LOW_WORD_2, BX
                        MOV     BX, UP_WORD_2
                        SHR     BX
                        MOV     UP_WORD_2, BX
                        MOV     BX, LOW_WORD_2
                        ROR     BX
                        MOV     LOW_WORD_2, BX
                        CMP     DX, UP_WORD_1
                        JA      DO_DIVISION_2
                        JB      DO_DIVISION_1
                        CMP     AX, LOW_WORD_2
                        JAE     DO_DIVISION_2
DO_DIVISION_1 :         CMP     DX, UP_WORD_2
                        JA      DO_DIVISION_3
                        JB      ADJUST_NC
                        CMP     AX, LOW_WORD_2
                        JAE     DO_DIVISION_3
ADJUST_NC :             CLC
                        SHL     AX
                        ROL     DX
                        JMP     DO_DIVISION_2
DO_DIVISION_3 :         DIV     DX, BX
                        SHL     AX
                        SHR     BX
                        CMP     DX, BX
                        JBE     NXT_5
                        INR     AX
NXT_5 :                 SHR     DX
                        SHR     BX
                        SHL     AX
                        ADD     CL, 01H
                        CMP     DX, BX
                        JBE     LOAD_STACK
                        INR     AX
                        JMP     LOAD_STACK
DO_DIVISION_2 :         DIV     DX, BX
                        SHR     BX
                        MOV     CX, 0000H
                        SHL     AX
                        ADD     CL, 01H
                        CMP     DX, BX
                        JBE     LOAD_STACK
                        INR     AX
                        SUB     CL, 0DH
                        POP     DS
                        PUSH    AX
                        PUSH    CX
                        RET
```

```
RSLV_P_16I :        POP   BX
                    POP   AX
                    MOV   DL, AH
                    MOV   DH, BH
                    MOV   CX, DX
                    AND   DX, 8080H
                    XOR   DH, DL
                    JNZ   UNEQUAL_SIGN
                    MOV   DX, CX
                    TEST  DL, 80H
                    JNZ   BOTH_NEG_SIGN
                    AND   AH, 7FH
                    AND   BH, 7FH
                    ADD   AX, BX
                    JNO   NXT_1
                    JMP   ERROR_3
NXT_1 :             PUSH  AX
                    RET
BOTH_NEG_SIGN :     AND   AH, 7FH
                    AND   BH, 7FH
                    NEG   AX
                    NEG   BX
                    ADD   AX, BX
                    TEST  AX, 8000H
                    JZ    NXT_2
                    JMP   LOAD_STACK
NXT_2 :             AND   AX, 7FFFH
                    NEG   AX
                    ADD   AX, 8000H
                    JMP   LOAD_STACK
UNEQUAL_SIGN :      MOV   DX, CX
                    AND   DX, 8080H
                    AND   AH, 7FH
                    AND   BH, 7FH
                    TEST  DL, 80H
                    JZ    NC_1_POS_2_NEG
NC_1_NEG_2_POS :    NEG   AX
                    ADD   AX, BX
                    TEST  AH, 80H
                    JZ    LOAD_STACK
                    AND   AH, 7FH
                    NEG   AX
                    ADD   AH, 80H
                    JMP   LOAD_STACK
NC_1_POS_2_NEG :    XCHG  AX, BX
                    JMP   NC_1_NEG_2_POS
LOAD_STACK :        PUSH  AX
                    RET
```

```
RSLV_N_16I :        POP   BX
                    POP   AX
                    MOV   DL, AH
                    MOV   DH, BH
                    MOV   CX, DX
                    AND   AH, 7FH
                    AND   BH, 7FH
                    XOR   DH, DL
                    JNZ   SIGN_UNEQUAL
                    MOV   DX, CX
                    TEST  DL, 80H
                    JNZ   BOTH_SIGN_NEG
                    NEG   BX
                    ADD   AX, BX
                    TEST  AH, 80H
                    JZ    LOAD_STACK_1
                    AND   AH, 7FH
                    NEG   AX
                    ADD   AH, 80H
                    JMP   LOAD_STACK_1
BOTH_SIGN_NEG :     NEG   AX
                    ADD   BX
                    JNO   LOAD_STACK_1
                    JMP   ERROR_3
SIGN_UNEQUAL :      MOV   DX, CX
                    TEST  DH, 80H
                    JZ    NC_1_NEG_2_POS
                    ADD   AX, BX
                    JNO   LOAD_STACK_1
                    JMP   ERROR_3
NC_1_NEG_2_POS :    NEG   AX
                    NEG   BX
                    ADD   AX, BX
                    TEST  AH, 80H
                    JZ    LOAD_STACK_1
                    AND   AH, 7FH
                    NEG   AX
                    ADD   AH, 80H
LOAD_STACK_1 :      PUSH  AX
                    RET
RSLV_M_16I :        POP   BX
                    POP   AX
                    MOV   CL, AH
                    MOV   CH, BH
                    AND   AH, 7FH
                    AND   BH, 7FH
                    MUL   AX, BX
                    TEST  AH, 80H
```

```
RSLV_N_16I :        POP   BX
                    POP   AX
                    MOV   DL, AH
                    MOV   DH, BH
                    MOV   CX, DX
                    AND   AH, 7FH
                    AND   BH, 7FH
                    XOR   DH, DL
                    JNZ   SIGN_UNEQUAL
                    MOV   DX, CX
                    TEST  DL, 80H
                    JNZ   BOTH_SIGN_NEG
                    NEG   BX
                    ADD   AX, BX
                    TEST  AH, 80H
                    JZ    LOAD_STACK_1
                    AND   AH, 7FH
                    NEG   AX
                    ADD   AH, 80H
                    JMP   LOAD_STACK_1
BOTH_SIGN_NEG :     NEG   AX
                    ADD   BX
                    JNO   LOAD_STACK_1
                    JMP   ERROR_3
SIGN_UNEQUAL :      MOV   DX, CX
                    TEST  DH, 80H
                    JZ    NC_1_NEG_2_POS
                    ADD   AX, BX
                    JNO   LOAD_STACK_1
                    JMP   ERROR_3
NC_1_NEG_2_POS :    NEG   AX
                    NEG   BX
                    ADD   AX, BX
                    TEST  AH, 80H
                    JZ    LOAD_STACK_1
                    AND   AH, 7FH
                    NEG   AX
                    ADD   AH, 80H
LOAD_STACK_1 :      PUSH  AX
                    RET
RSLV_M_16I :        POP   BX
                    POP   AX
                    MOV   CL, AH
                    MOV   CH, BH
                    AND   AH, 7FH
                    AND   BH, 7FH
                    MUL   AX, BX
                    TEST  AH, 80H
```

```
                        JZ      NXT_1
                        JMP     ERROR_3
NXT_1 :                 XOR     CL, CH
                        ADD     AH, CL
                        PUSH    AX
                        RET
RSLV_D_16I :            POP     BX
                        POP     AX
                        MOV     CL, AH
                        MOV     CH, BH
                        AND     AH, 7FH
                        AND     BH, 7FH
                        AND     CX, 8080H
                        CMP     AX, DX
                        JB      DIVS_GTHAN_DVND
                        MOV     DX, 00H
                        DIV     DX, BX
                        XOR     CL, CH
                        ADD     AH, CL
                        JMP     LOAD_STACK_2
DIVS_GTHAN_DVND :       MOV     AX, 0000H
;HERE THE DIVISOR IS GREATER THAN THE DIVIDEND, HENCE THE  RESULT
;IS LOADED AS ZERO.
LOAD_STACK_2 :          PUSH    AX
                        RET
RSLV_P_8I :             POP     BX
                        POP     AX
                        MOV     DL, AL
                        MOV     DH, BL
                        AND     DX, 8080H
                        MOV     CX, DX
                        AND     AL, 7FH
                        AND     BL, 7FH
                        XOR     DL, DH
                        JNZ     UNEQUAL_SIGN
                        MOV     DX, CX
                        TEST    DL, 80H
                        JNZ     BOTH_NEG_SIGN
                        AND     AL, BL
                        JNO     LOAD_STACK_4
                        JMP     ERROR_3
BOTH_NEG_SIGN :         NEG     AL
                        NEG     BL
                        ADD     AL, BL
                        TEST    AL, 80H
                        JZ      LOAD_STACK_4
                        AND     AL, 7FH
                        NEG     AL
```

```
                        ADD     AL, 80H
                        JMP     LOAD_STACK_4
UNEQUAL_SIGN :          MOV     DX, CX
                        TEST    DL, 80H
                        JZ      NC_1_POS_2_NEG
NC_1_NEG_2_POS :        NEG     AL
                        ADD     AL, BL
                        TEST    AL, 80H
                        JZ      LOAD_STACK_4
                        AND     AL, 7FH
                        AND     BL, 7FH
                        NEG     AL
                        ADD     AL, 80H
                        JMP     LOAD_STACK_4
NC_1_POS_2_NEG :        XCHG    AL, BL
                        JMP     NC_1_NEG_2_POS
LOAD_STACK_4 :          PUSH    AX
                        RET
RSLV_N_8I :             POP     BX
                        POP     AX
                        MOV     DL, AL
                        MOV     DH, BL
                        AND     DX, 8080H
                        MOV     CX, DX
                        AND     AL, 7FH
                        AND     BL, 7FH
                        XOR     DH, DL
                        JNZ     UNEQUAL_SIGN_1
                        MOV     DX, CX
                        TEST    DL, 80H
                        JZ      BOTH_SIGN_NEG
                        NEG     BL
                        ADD     AL, BL
                        TEST    AL, 80H
                        JZ      LOAD_STACK_5
                        AND     AL, 7FH
                        NEG     AX
                        ADD     AL, 80H
                        JMP     LOAD_STACK_5
BOTH_SIGN_NEG :         NEG     AL
                        ADD     AL, BL
                        JNO     LOAD_STACK_5
                        JMP     ERROR_3
UNEQUAL_SIGN_1 :        MOV     DX, CX
                        TEST    DH, 80H
                        JZ      NC_2_POS_1_NEG
                        ADD     AL, BL
                        JNO     LOAD_STACK_5
```

```
                        JMP     ERROR_3
NC_2_POS_1_NEG :        NEG     AL
                        NEG     BL
                        ADD     AL, BL
                        TEST    AL, 80H
                        JZ      LOAD_STACK_5
                        AND     AL, 7FH
                        NEG     AL
                        ADD     AL, 80H
LOAD_STACK_5 :          PUSH    AX
                        RET
RSLV_M_8I :             POP     BX
                        POP     AX
                        MOV     CL, AL
                        MOV     CH, BL
                        AND     CX, 8080H
                        AND     AL, 7FH
                        AND     BL, 7FH
                        MUL     AL, BL
                        TEST    AL, 80H
                        JZ      FORM_SIGN
                        JMP     ERROR_3
FORM_SIGN :             XOR     CL, CH
                        ADD     AL, CL
                        MOV     AH, 00H
                        PUSH    AX
                        RET
RSLV_D_8I :             POP     BX
                        POP     AX
                        MOV     CL, AL
                        MOV     CH, BL
                        AND     CX, 8080H
                        AND     AL, 7FH
                        AND     BL, 7FH
                        CMP     AL, BL
                        JB      DIVS_GTHAN_DVND_1
                        DIV     AX, BL
                        XOR     CL, CH
                        ADD     AL, CL
                        JMP     LOAD_STACK_3
DIVS_GTHAN_DVND_1 :     PUSH    AX
                        RET
RSLV_CONVERT_RUN :      TEST    BH, 01H
                        JZ      NXT_1
                        INR     SI
                        CALL    DET_VAR_ADD
                        MOV     DI, AX
                        MANAGE_SEG_0
```

```
                        MOV     CX, PRESENT_DATA_SEG
                        POP     DS
                        TEST    BL, 1CH
                        JNZ     CONT_1
                        MOV     CX, 1FF0H
CONT_1 :                MOV     SS, CX
                        TEST    BL, 01H
                        JNZ     LOAD_INT_VAR
                        TEST    BL, 04H
                        JZ      LOAD_16R
                        MOV     CL, (BP+DI)
                        INR     DI
LOAD_16R :              MOV     AX, (BP+DI)
                        JMP     CONT_2
LOAD_IN_VAR :           MOV     AL, (BP+DI)
                        TEST    BL, 04H
                        JZ      CONT_2
                        INR     DI
                        MOV     AH, (BP+DI)
CONT_2 :                INR     SI
                        CALL    (SI)
                        RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_4 : RUN_ICP_INB

ANATOMY :

THE FLOW IS FRAGMENTED JUDICIALLY FOR VARIOUS SUBGROUPS SUCH THAT CODES FORM THE COMMON PART OF ACTION NEED NOT BE REPEATED. A SEPARATE FRAGMENT LOADS '?' IN THE DISPLAY FIELD, ACQUISES NUMERIC ENTRIES AND LOADS THE INTERNAL FORM OF THE NUMBER IN THE DATA AREA. THE RUN ROUTINE ALLOWS REINITIALISATION OF PORT_C_PPI_4 OR PPI_2 IN THE INPUT MODE AND RESETING BACK TO ORIGINAL STATE (MODULE : POST, LEVEL_0). THESE TWO ACTIONS ARE INCLUDED AS MACRO. THE PORT ADDRESS BEING INDEPENDENT OF CONTENTS OF SEGMENT REGISTER RELIEVES US FROM ADDITIONAL SEGMENT MANAGEMENT EFFORT.

REGISTER USAGE :

AL : INPUTED DATA, TEMPORARY OPERANDS.
BH : SUBGROUP, INPUT DATA (TEMPORARY STORAGE).
CX : SEGMENT MANAGEMENT.
DX : ADDRESS OF THE PORT TO BE ACCESSED.

REFERENCE MEMORY LOCATIONS :

PRESENT_DATA_SEG, MARK_2.

SUBROUTINES :

DET_VAR_ADD [SUB_PART_1, LEVEL_3(1,1)].

```
LABEL_INB_RUN :     TEST   BH, 04H
                    JZ     CONT_1
                    INR    SI
                    MOV    DX, (SI)
                    INR    SI
MACRO               PORT_INITIALISATION_CONFIRM
                    PUSH   DS
                    MOV    CX, 0000H
                    MOV    DS, CX
                    CMP    DX, PORT_C_PPI_2
                    JA     AGAIN_1
                    CMP    BX, PORT_A_PPI_2
                    JB     AGAIN_2
                    MOV    CX, DX
                    MOV    DX, CONTROL_PORT_PPI_2
```

```
                        MOV     AL, MASK
;PORT_A_B_C ARE MADE INPUT PORTS IN MODE_0.
                        OUT     DX, CX
                        MOV     MARK_2, OFH
;THE MARK INDICATES INITIALISATION OF PPI_2.
                        JMP     AGAIN_2
AGAIN_1 :               CMP     DX, PORT_B_PPI_4
                        JNZ     AGAIN_2
                        MOV     CX, DX
                        MOV     DX, CONTROL_PORT_PPI_4
                        MOV     AL, MASK
;PORT_A_B_C ARE MADE INPUT PORTS IN MODE_0.
                        OUT     DX, AL
                        MOV     MARK_2, FOH
;THE MARK INDICATES INITIALISATION OF PPI_4.
                        MOV     DX, CX
AGAIN_2 :               POP     DS
                        ENDM
                        TEST    BH, 02H
                        JZ      CONT_2
                        OUT     DX, AL
                        HLT
                        JMP     CONT_3
CONT_2 :                IN      AL, DX
CONT_3 :                TEST    BH, 01H
                        JZ      CONT_4
                        MOV     BH, AL
                        CALL    DET_VAR_ADD
                        MOV     DI, AX
                        MANAGE_SEG_0
                        MOV     CX, PRESENT_DATA_SEG
                        POP     DS
                        TEST    BL, 1CH
                        JNZ     CONT_5
                        MOV     CX, 1FFOH
CONT_5 :                MOV     SS, CX
                        MOV     AL, BH
                        MOV     (BP+DI), AL
                        MOV     CX, 0000H
                        MOV     SS, CX
MACRO                   PORT_REINITIALISATION
                        PUSH    DS
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     BH, MARK_2
                        CMP     BH, 00H
                        JZ      AGAIN_4
                        CMP     MARK_2, OFH
```

```
                        JNZ     AGAIN_5
                        MOV     DX, CONTROL_PORT_PPI_2
                        MOV     AL, MASK
;PORT_A_B_C, OUTPUT, MODE_0.
                        OUT     DX, AL
                        MOV     MARK_2, 00H
                        JMP     AGAIN_4
AGAIN_5 :               CMP     BH, F0H
                        JZ      AGAIN_6
                        JMP     ERROR_3
AGAIN_6 :               MOV     DX, CONTROL_PORT_PPI_4
                        MOV     AL, MASK
;PORT_A_B, INPUT, PORT_C, OUTPUT, MODE_0.
                        OUT     DX, AL
                        MOV     MARK_2, 00H
AGAIN_4 :               POP     DS
                        ENDM
CONT_4 :                RET
CONT_1 :                TEST    BH, 03H
                        JNZ     CONT_6
                        MOV     CX, DX
                        MOV     DX, CONTROL_PORT_KBDC
                        MOV     AL, DX
                        MOV     AH, AL
                        MOV     AL, 00H
                        TEST    AH, MASK
                        JZ      CONT_7
                        MOV     DX, CX
                        IN      AL, DX
CONT_7 :                RET
CONT_6 :                MOV     DI, BASE_ADD_KBB
                        MOV     BL, 00H
                        MOV     BH, 01H
                        MOV     (BP+DI), '?'
                        INT     VECTOR_KBIRQ
                        UNMASK_KBIRQ
                        LED_INDICATOR_ON
                        HLT
                        CMP     AL, 'ENTER'
                        JZ      CONT_8
                        JMP     ERROR_1
CONT_8 :                MASK    KBIRQ
                        LED_INDICATOR_OFF
                        BLANK_ALL
                        CALL    RSLV_CNST
                        MOV     CH, CL
                        MOV     BH, CL
                        INR     SI
```

```
                        SUB     CH, 02H
                        MOV     CL, (SI)
                        AND     CX, 0707H
                        CMP     CH, CL
                        JZ      CONT_9
                        JMP     ERROR_1
CONT_9 :                PUSH    AX
                        PUSH    DX
                        PUSH    BX
                        CALL    DET_VAR_ADD
                        MOV     DI, AX
                        MANAGE_SEG_0
                        MOV     CX, PRESENT_DATA_SEG
                        POP     DS
                        TEST    BL, 1CH
                        JNZ     CONT_A
                        MOV     CX, 1FF0H
CONT_A :                MOV     SS, CX
                        POP     BX
                        POP     DX
                        POP     AX
                        TEST    BH, 01H
                        JNZ     LOAD_INT
                        TEST    BH, 04H
                        JZ      LOAD_16R
                        MOV     (BP+DI), DL
                        INR     DI
LOAD_16R :              MOV     (BP+DI), AX
CONT_B :                MOV     CX, 00H
                        MOV     SS, CX
                        RET
LOAD_INT :              TEST    BH, 04H
                        JNZ     LOAD_16R
                        MOV     (BP+DI), AL
                        JMP     CONT_B
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_5 : RUN_ICP_DLY

ANATOMY :

THE ROUTINE USES IC PARAMETERS TO INITIALISE COUNTING IN TIMER_1. REFER MODULE : POST, LEVEL_0, F.C. 4.1.

REGISTER USAGE :

DX : PORT AND CONTROL REGISTER ADDRESS OF TIMER_1.
BH : SUBGROUP
AL : COMMAND WORDS FOR TIMER_1

```
                        UNMASK KBIRQ
LABEL_DLY_RUN :         TEST   BH, 00H
                        JNZ    CHECK_GROUP_2
                        MOV    DX, CONTROL_PORT_TIMER_1
                        MOV    AL, CONTROL_WORD_FOR_COUNTER_2
;MODE_3 AND LSB_FIRST, MSB_NEXT, BCD_COUNT.
                        OUT    DX, AL
                        MOV    DX, COUNTER_2_DATA_PORT_TIMER_1
;8000D = 1E40H, IS LOADED.
                        MOV    AL, 40H
                        OUT    DX, AL
                        MOV    AL, 1EH
                        OUT    DX, AL
                        MOV    DX, CONTROL_PORT_TIMER_1
                        MOV    AL, CONTROL_WORD_FOR_COUNTER_0
;MODE_3 AND LSB_FIRST, MSB_NEXT, BCD_COUNT.
                        OUT    DX, AL
                        INR    SI
                        MOV    AX, (SI)
                        OUT    DX, AL
                        MOV    AL, AH
                        OUT    DX, AL
                        JMP    DLY_END
CHK_GROUP_2 :           TEST   BH, 01H
                        JZ     CHK_GROUP_3
                        MOV    DX, CONTROL_PORT_TIMER_1
                        MOV    AL, CONTROL_WORD_FOR_COUNTER_2
;MODE_3 AND LSB_FIRST, MSB_NEXT, BCD_COUNT.
                        OUT    DX, AL
                        MOV    DX, COUNTER_2_DATA_PORT
                        MOV    AL, 20H
                        OUT    DX, AL
```

```
                        MOV     AL, 03H
                        OUT     DX, AL
                        MOV     DX, CONTROL_PORT_TIMER_1
                        MOV     AL, CONTROL_WORD_FOR_COUNTER_0
;MODE_3 AND LSB_FIRST, MSB_NEXT, BCD_COUNT.
                        OUT     DX, AL
                        MOV     DX, COUNTER_0_DATA_PORT
                        INR     SI
                        MOV     AX, (SI)
                        OUT     DX, AL
                        MOV     AL, AH
                        OUT     DX, AL
                        JMP     DLY_END
CHK_GROUP_3 :           MOV     DX, CONTROL_PORT_TIMER_1
                        MOV     AL, CONTROL_WORD_FOR_COUNTER_0
;MODE_3 AND LSB_FIRST, MSB_NEXT, BCD_COUNT.
                        OUT     DX, AL
                        MOV     DX, COUNTER_0_DATA_PORT
                        INR     SI
                        MOV     AX, (SI)
                        OUT     DX, AL
                        MOV     AL, AH
                        OUT     DX, AL
DLY_END :               HLT
;UPON INTERRUPT FROM COUNTER_0 OF TIMER_1.
                        MASK    KBIRQ
                        RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_6 : RUN_ICP_OUB

ANATOMY :

THE PROGRAM STRUCTURE IS SIMILAR TO SUBGROUP_04 AND _05 OF PART_4. THE EXPLANATION OF PART_4 IS VALID IN THIS CASE.

```
LABEL_OUB_RUN :     INR    SI
                    MOV    DX, (SI)
                    INR    SI
                    TEST   BH, 01H
                    JZ     CONT_1
                    PUSH   DX
                    CALL   DET_VAR_ADD
                    MOV    DI, AX
                    MANAGE_SEG_0
                    TEST   BL, 1CH
                    JNZ    CONT_2
                    MOV    CX, 1FF0H
COUNT_2 :           MOV    SS, CX
                    MOV    AL, (BP+DI)
                    MOV    CX, 0000H
                    MOV    SS, CX
                    POP    DX
CONT_1 :            MANAGE_SEG_0
                    MOV    BH, AL
                    MOV    CX, CONTROL_PORT_PPI_1
                    CMP    DX, PORT_C_PPI_1
                    JA     AGAIN_7
                    XCHG   CX, DX
                    MOV    AL, MASK
;PORT_A_B_C ARE MADE OUTPUT_PORTS IN MODE_0.
                    OUT    DX, AL
                    XCHG   DX, CX
                    MOV    MARK_2, 0FH
;INDICATES INITIALISATION OF PPI_1 ALTER.
                    JMP    AGAIN_8
AGAIN_7 :           CMP    DX, PORT_C_PPI_4
                    JZ     AGAIN_8
                    MOV    CX, DX
                    MOV    DX, CONTROL_PORT_PPI_4
                    MOV    AL, MASK
;PORT_A_B_C ARE MADE OUTPUT_PORTS IN MODE_0.
```

```
                        OUT     DX, AL
                        MOV     MARK_2, F0H
;INDICATES INITIALISATION OF PPI_4 ALTER.
                        XCHG    DX, CX
AGAIN_8 :               POP     DS
                        MOV     AL, BH
                        OUT     DX, AL
                        XCHG    DX, CX
                        PUSH    DS
                        MOV     CX, 0000H
                        MOV     DS, CX
                        MOV     BH, MARK_2
                        CMP     BH, 00H
                        JZ      AGAIN_9
                        CMP     BH, 0FH
                        JNZ     AGAIN_A
                        MOV     AL, MASK
;PORT_A_B_C ARE MADE INPUT_PORTS IN MODE_0.
                        OUT     DX, AL
                        JMP     AGAIN_9
AGAIN_A :               CMP     BH, F0H
                        JZ      AGAIN_D
                        JMP     ERROR_3
AGAIN_B :               MOV     AL, MASK
;PORT_A_C ARE MADE INPUT_PORTS, B_PORT_OUTPUT IN MODE_0.
                        OUT     DX, AL
AGAIN_9 :               POP     DS
                        RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_7 : RUN_ICP_INR

ANATOMY :

THE MODULE POINTS THE SPECIFIED VARIABLE AND CONFIRM NO OVER FLOW OCCURS. AFTER INCREMENTATION (ALLOWED RANGE OF NUMBERS, 0 TO + MAX.). THE MAGNITUDE OF SPECIFIED VARIABLE IS INCREMENTED.

REGISTER USAGE :

AX : ARGUMENT TO BE INCREMENTED.

REFERENCE MEMORY LOCATIONS :

PRESENT_DATA_SEG

SUBROUTINES :

DET_VAR_ADD

```
LABEL_INR_RUN :     INR    SI
                    CALL   DET_VAR_ADD
                    MOV    DI, AX
                    MANAGE_SEG_0
                    MOV    CX, PRESENT_DATA_SEG
                    POP    DS
                    TEST   BL, 1CH
                    JNZ    CONT_1
                    MOV    CX, 1FF0H
CONT_1 :            MOV    SS, CX
                    CMP    BL, 04H
                    JZ     CONT_2
                    MOV    AX, (BP+DI)
                    CMP    AX, 7FFFH
;CHECK WHETHER 16_I VARIABLE CONTENTS IS FULL.
                    JB     CONT_3
                    JMP    ERROR_3
CONT_3 :            INR    AX
                    MOV    (BP+DI), AX
                    JMP    CONT_4
CONT_2 :            CMP    (BP+DI), 7FH
                    JB     CONT_5
                    JMP    ERROR_3
CONT_5 :            INR    (BP+DI)
CONT_4 :            MOV    CX, 0000H
                    RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_8 : RUN_ICP_INW

ANATOMY :

TAKING CARE OF REINITIALISATION, AND RESETTING IT BACK THE WORD FROM THE SPECIFIED PORT IS ACCESSED. IF SPECIFIED IN IC FIELD, THE WORD IS TRANSFERRED TO DATA AREA.

REGISTER USAGE :

DX : PORT ADDRESS
BH : SUBGROUP
AX : INPUTED DATA

REFERENCE MEMORY LOCATIONS :

PRESENT_DATA_SEG, ROUTINES, DET_VAR_ADD

```
LABEL_INW_RUN :         INR    SI
                        MOV    DX, (SI)
                        INR    SI
                        MOV    CX, DX
                        MOV    DX, CONTROL_PORT_PPI_2
                        MOV    AL, MASK
;PORT_A_B_C IN INPUT MODE_0.
                        OUT    DX, AL
                        IN     AX, DX
                        CMP    BH, 01H
                        JNZ    CONT_1
                        PUSH   AX
                        CALL   DET_VAR_ADD
                        MANAGE_SEG_0
                        MOV    CX, PRESENT_DATA_SEG
                        MOV    DI, AX
                        TEST   BL, 1CH
                        JNZ    CONT_2
                        MOV    CX, 1FF0H
CONT_2 :                MOV    SS, CX
                        POP    AX
                        MOV    (BP+DI), AX
                        MOV    CX, 0000H
                        MOV    SS, CX
CONT_1 :                MOV    DX, CONTROL_PORT_PPI_2
                        MOV    AL, MASK
;PORT_A_B_C IN OUTPUT MODE_0.
                        RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_9 : RUN_ICP_OUW

ANATOMY :

PERFORMING INITIALISATION MANAGEMENT AS IN PART_4 AND PART_8 A WORD IS OUTPUTED TO SPECIFIED PORT.

REGISTER USAGE :

DX : PORT ADDRESSES.
CX : TEMPORARY PORT ADDRESSES.
AX : INPUT DATA PARAMETERS AND TEMPORARY OPERAND.

REFERENCE MEMORY LOCATIONS :

PRESENT_DATA_SEG

SUBROUTINE :

DET_VAR_ADD

```
LABEL_OUW_RUN :     INR   SI
                    MOV   DX, (SI)
                    INR   SI
                    TEST  BH, 01H
                    JZ    CONT_1
                    PUSH  DX
                    CALL  DET_VAR_ADD
                    MOV   DI, AX
                    MOV   CX, PRESENT_DATA_SEG
                    TEST  BL, 1CH
                    JNZ   CONT_3
                    MOV   CX, 1FF0H
CONT_3 :            MOV   SS, CX
                    MOV   AX, (BP+DI)
                    MOV   CX, 0000H
                    MOV   SS, CX
                    POP   DX
CONT_1 :            MOV   CX, DX
                    MOV   DX, CONTROL_PORT_PPI_1
                    MOV   AL, MASK
;PORT_A_B_C OUTPUT_PORTS IN MODE_0.
                    OUT   DX, AL
                    XCHG  DX, CX
                    OUT   DX, AX
                    XCHG  DX, CX
                    MOV   AL, MASK
```

```
;PORT_A_B_C INPUT_PORTS IN MODE_0.
                        OUT   DX, AL
                        RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_10 : RUN_ICP_DSP

ANATOMY :

IF THE SUBGROUP INDICATES A DISPLAY OF NUMERIC VALUES AND MESSAGES THEN THE NUMERIC VALUES FROM SPECIFIED ADDRESS AND DATA TYPE ARE BACK CONVERTED USING SUBROUTINES BACK CONVERT. THIS ROUTINE LOADS THE KBB WITH THE DESIRED NIBBLES. IF THE MESSAGE IS TO BE SUPPLIMENTED THE ROUTINE LOADS THE MESSAGE FROM ICP INTO THE KBB AND MANAGES BX ACCORDINGLY (REGISTER BH SPECIFIES THE NUMBER OF BYTES TO BE DISPLAYED). REFER MODULE : KBIRQ, LEVEL_1, F.C. 4.1.

```
LABEL_DSP_RUN :        CMP    BH, 01H
                       JNZ    CHK_VAR_CHR
                       MOV    BX, 0000H
                       MOV    CH, 00H
                       MOV    DI, BASE_ADD_KBB
                       INR    SI
                       MOV    CL, (SI)
;COUNT OR NUMBER OF CHARACTERS TO BE DISPLAYED.
CONT_2 :               ADD    BH, CL
CONT_1 :               INR    SI
                       MOV    AL, (SI)
                       MOV    (BP+DI), AL
                       INR    DI
                       LOOP   CONT_1
                       UNMASK_KBIRQ
                       INT    VECTOR_KBIRQ
                       HLT
                       MASK_KBIRQ
                       RET
                       INR    SI
                       MOV    CH, 00H
                       MOV    CL, (SI)
                       PUSH   CX
                       INR    SI
                       CALL   DET_VAR_ADD
                       TEST   BL, 01H
                       JNZ    CHK_INT_VAR
                       TEST   BL, 04H
                       JZ     VAR_16R
```

```
                        CALL    BACK_CONVERT_24R
                        JMP     DSP_END
VAR_16R :               CALL    BACK_CONVERT_16R
                        JMP     DSP_END
CHK_INT_VAR :           TEST    BL, 04H
                        JZ      VAR_8I
                        CALL    BACK_CONVERT_16I
                        JMP     DSP_END
VAR_8I :                CALL    BACK_CONVERT_8I
DSP_END :               POP     CX
                        JMP     CONT_2
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_11 : RUN_ICP_DCR.

ANATOMY : THE MODULE POINTS THE SPECIFIED VARIABLE AND CONFIRM NO UNDER FLOW OCCURS. AFTER DECREMENTATION (ALLOWED RANGE OF NUMBERS, 0 TO + MAX.). THE MAGNITUDE OF SPECIFIED VARIABLE IS DECREMENTED.

REGISTER USAGE :
AX : ARGUMENT TO BE DECREMENTED.

REFERENCE MEMORY LOCATIONS :
PRESENT_DATA_SEG.

SUBROUTINES :
DET_VAR_ADD

```
LABEL_DCR_RUN :     INR    SI
                    CALL   DET_VAR_ADD
                    MOV    DI, AX
                    MANAGE_SEG_0
                    MOV    CX, PRESENT_DATA_SEG
                    POP    DS
                    TEST   BL, 1CH
                    JNZ    CONT_1
                    MOV    CX, 1FF0H
CONT_1 :            MOV    SS, CX
                    CMP    BL, 04H
                    JZ     CONT_2
                    MOV    AX, (BP+DI)
                    TEST   AH, 80H
                    JZ     CONT_3
                    JMP    ERROR_3
CONT_3 :            CMP    AX, 0000H
                    JNZ    CONT_6
                    JMP    ERROR_3
CONT_6 :            DCR    AX
                    MOV    (BP+DI), AX
                    JMP    CONT_4
CONT_2 :            TEST   (BP+DI), 80H
                    JZ     CONT_5
                    JMP    ERROR_3
CONT_5 :            CMP    (BP+DI), 00H
                    JNZ    CONT_7
```

```
                        JMP     ERROR_3
CONT_7 :                DCR     (BP+DI)
CONT_4 :                MOV     CX, 0000H
                        MOV     SS, CX
                        RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_12 : RUN_ICP_IF

ANATOMY :

OPERANDS ARE COMPARED WITH CONDITIONAL OPERATORS, IF CONDITION PROVED FLOW JUMPS TO BRANCH LINE NUMBER.

```
LABEL_IF_RUN :      CMP    BH, 05H
                    JB     NXT_1
                    JMP    RSLV_VAR
NXT_1 :             INR    SI
                    MOV    DX, (SI)
                    INR    SI
                    TEST   BH, 04H
                    JNZ    NXT_2
                    JMP    RSLV_KB
NXT_2 :             PORT_INITIALISATION_CONFIRM
                    TEST   BH, 01H
                    JNZ    RSLV_VAR_ADD
                    TEST   BL, 02H
                    JZ     CONT_1
                    OUT    DX, AL
                    HLT
                    JMP    CONT_2
CONT_1 :            IN     AL, DX
CONT_2 :            INR    SI
                    MOV    BL, (SI)
                    INR    SI
                    MOV    CH. (SI)
                    JMP    COMPARISION_8I
RSLV_VAR_ADD :      TEST   BL, 02H
                    JZ     CONT_3
                    OUT    DX, AL
                    HLT
                    JMP    CONT_4
CONT_3 :            IN     AL, DX
CONT_4 :            MOV    BH, AL
                    CALL   DET_VAR_ADD
                    MOV    DI, AX
                    MANAGE_SEG_O
                    MOV    CX, PRESENT_DATA_SEG
                    TEST   BL, 1CH
                    JNZ    CONT_5
                    MOV    CX, 1FFOH
```

```
                    MANAGE_SEG_O
                    MOV     CX, PRESENT_DATA_SEG
                    POP     DS
                    MOV     SS, CX
                    MOV     CL, (BP+DI)
                    INR     DI
                    MOV     BX, (BP+DI)
                    INR     SI
                    MOV     CH, (SI)
                    JMP     COMPARISION_24R
CONT_9 :            MOV     AX, (BP+DI)
                    MOV     DX, OOOOH
                    MOV     SS, DX
                    PUSH    AX
                    TEST    BH, O1H
                    JNZ     NXT_4
                    JMP     RSLV_NC_16R
NXT_4 :             INR     SI
                    CALL    DET_VAR_ADD
                    MOV     DI, AX
                    MANAGE_SEG_O
                    MOV     CX, PRESENT_DATA_SEG
                    POP     DS
                    TEST    BL, 1CH
                    JNZ     CONT_10
                    MOV     CX, 1FFOH
CONT_10 :           MOV     SS, CX
                    MOV     BX, (BP+DI)
                    MOV     DX, OOOOH
                    MOV     SS, DX
                    INR     SI
                    MOV     CH, (SI)
                    POP     AX
                    JMP     COMPARISION_16R
CONT_8 :            TEST    BL, O4H
                    JZ      CONT_11
                    MOV     AX, (BP+DI)
                    MOV     DX, OOOOH
                    MOV     SS, DX
                    PUSH    AX
                    TEST    BH, O1H
                    JNZ     NXT_5
                    JMP     RSLV_NC_16I
NXT_5 :             INR     SI
                    CALL    DET_VAR_ADD
                    MOV     DI, AX
                    MANAGE_SEG_O
                    MOV     CX, PRESENT_DATA_SEG
```

```
CONT_5 :        MOV     SS, CX
                MOV     BL, (BP+DI)
                MOV     CX, 0000H
                MOV     SS, CX
                INR     SI
                MOV     CH, (SI)
                PORT_REINITIALISE
                MOV     AL, BH
                JMP     COMPARISION_8I
RSLV_KB :       MOV     CX, DX
                MOV     DX, CONTROL_PORT_KBDC
                IN      AL, DX
                TEST    AL, MASK
;MASK OF KBIRQ.
                JZ      CONT_6
                MOV     DX, CX
                IN      AL, DX
CONT_6 :        INR     SI
                MOV     BL, (SI)
                INR     SI
                MOV     CH, (SI)
                JMP     COMPARISION_8I
RSLV_VAR :      INR     SI
                CALL    DET_VAR_ADD
                MOV     DI, AX
                MANAGE_SEG_0
                MOV     CX, PRESENT_DATA_SEG
                TEST    BL, 1CH
                JNZ     CONT_7
                MOV     CX, 1FF0H
CONT_7 :        MOV     SS, CX
                MOV     CH, 00H
                TEST    BL, 01H
                JNZ     CONT_8
                TEST    BL, 04H
                JZ      CONT_9
                MOV     CL, (BP+DI)
                INR     DI
                MOV     AX, (BP+DI)
                MOV     DX, 0000H
                PUSH    AX
                PUSH    CX
                TEST    BH, 01H
                JNZ     NXT_3
                JMP     RSLV_NC_24R
NXT_3 :         INR     SI
                CALL    DET_VAR_ADD
                MOV     DI, AX
```

```
                    POP     DS
                    MOV     SS, CX
                    MOV     BX, (BP+DI)
                    MOV     DX, 0000H
                    MOV     SS, DX
                    INR     SI
                    MOV     CH, (SI)
                    POP     AX
                    JMP     COMPARISION_16I
CONT_11 :           TEST    BH, 01H
                    JNZ     NXT_6
                    JMP     RSLV_NC_8I
NXT_6 :             MOV     DX, 0000H
                    MOV     SS, DX
                    MOV     BH, AL
                    INR     SI
                    CALL    DET_VAR_ADD
                    MOV     DI, AX
                    MANAGE_SEG_0
                    MOV     CX, PRESENT_DATA_SEG
                    POP     DS
                    TEST    BL, 1CH
                    JNZ     CONT_12
                    MOV     CX, 1FF0H
CONT_12 :           MOV     SS, CX
                    MOV     BL, (BP+DI)
                    MOV     AL, BH
                    MOV     DX, 0000H
                    MOV     SS, DX
                    INR     SI
                    MOV     CH, (SI)
                    JMP     COMPARISION_8I
RSLV_NC_24R :       ADD     SI, 0002H
                    MOV     DL, (SI)
                    INR     SI
                    MOV     BX, (BP+DI)
                    ADD     SI, 0002H
                    MOV     CH, (BP+DI)
                    JMP     COMPARISION_24R
RSLV_NC_16R :       ADD     SI, 0002H
                    MOV     BX, (SI)
                    ADD     SI, 0002H
                    MOV     CH, (SI)
                    POP     AX
                    JMP     COMPARISION_16R
RSLV_NC_16I :       ADD     SI, 0002H
                    MOV     BX, (SI)
                    ADD     SI, 0002H
```

```
                        MOV     CH, (SI)
                        POP     AX
                        JMP     COMPARISION_16I
RSLV_NC_8I :            ADD     SI, 0002H
                        MOV     BL, (SI)
                        INR     SI
                        MOV     CH, (SI)
                        JMP     COMPARISION_8I
COMPARISION_8I :        MOV     AH, BL
                        MOV     BH, AL
                        AND     AX, 1010H
                        CMP     CH, '='
                        JNZ     NXT_OPERATOR_1
                        CMP     BL, BH
                        JZ      NXT_1
                        RET
NXT_1 :                 MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_1 :        CMP     CH, '≠'
                        JNZ     NXT_OPERATOR_2
                        CMP     BL, BH
                        JNZ     NXT_21
                        RET
NXT_21 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_2 :        CMP     CH, '<'
                        JNZ     NXT_OPERATOR_3
                        XOR     AL, AH
                        JNZ     NXT_3
                        CMP     BL, BH
                        JB      NXT_31
                        RET
NXT_3 :                 TEST    AH, 80H
                        JZ      NXT_31
                        RET
NXT_31                  MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_3 :        CMP     CH, '<='
                        JNZ     NXT_OPERATOR_4
                        XOR     AL, AH
                        JNZ     NXT_4
                        CMP     BL, BH
                        JB      NXT_41
                        RET
NXT_4 :                 TEST    AH, 10H
                        JZ      NXT_41
                        RET
NXT_41 :                MOV     CL, FFH
```

```
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_4 :        CMP     CH, '>'
                        JNZ     NXT_OPERATOR_5
                        XOR     AL, AH
                        JNZ     NXT_5
                        CMP     BL, BH
                        JB      NXT_51
                        RET
NXT_5 :                 TEST    AH, 10H
                        JZ      NXT_51
                        RET
NXT_51 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_5 :        CMP     CH, '>='
                        JZ      NXT_60
                        JMP     ERROR_3
NXT_60 :                XOR     AL, AH
                        JNZ     NXT_61
                        CMP     BL, BH
                        JAE     NXT_62
                        RET
NXT_61 :                TEST    AH, 10H
                        JNZ     NXT_62
                        RET
NXT_62 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
                        RET
COMPARISION_16I :       CMP     CH, '='
                        JNZ     NXT_OPERATOR_1
                        CMP     AX, BX
                        JZ      NXT_1
                        RET
NXT_1 :                 MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_1 :        CMP     CH, '≠'
                        JNZ     NXT_OPERATOR_2
                        CMP     AX, BX
                        JNZ     NXT_21
                        RET
NXT_21 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_2 :        CMP     CH, '<'
                        JNZ     NXT_OPERATOR_3
                        TEST    AH, 10H
                        JZ      NXT_3
                        TEST    BH, 10H
                        JZ      NXT_31
                        CMP     AX, DX
```

```
                        JB      NXT_31
NXT_32 :                RET
NXT_3 :                 TEST    BH, 10H
                        JNZ     NXT_32
                        CMP     AX, BX
                        JB      NXT_31
                        RET
NXT_31 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_3 :        CMP     CH, '<='
                        JNZ     NXT_OPERATOR_4
                        TEST    AH, 10H
                        JZ      NXT_4
                        TEST    BH, 10H
                        JZ      NXT_41
                        CMP     AX, BX
                        JBE     NXT_41
                        RET
NXT_4 :                 TEST    BH, 10H
                        JNZ     NXT_42
                        CMP     AX, BX
                        JBE     NXT_42
                        RET
NXT_42 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_4 :        CMP     CH, '>'
                        JNZ     NXT_OPERATOR_5
                        TEST    AH, 10H
                        JNZ     NXT_5
                        CMP     BH, 10H
                        JNZ     NXT_51
                        CMP     AX, BX
                        JA      NXT_51
                        RET
NXT_5 :                 TEST    BH, 10H
                        JZ      NXT_51
                        RET
NXT_51 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_5 :        CMP     CH, '>='
                        JZ      NXT_60
                        JMP     ERROR_3
NXT_60 :                TEST    AH, 10H
                        JNZ     NXT_61
                        CMP     BH, 10H
                        JNZ     NXT_62
                        CMP     AX, BX
                        JAE     NXT_62
```

```
                        RET
NXT_61 :                TEST    BH, 10H
                        JZ      NXT_62
                        CMP     AX, BX
                        JAE     NXT_62
                        RET
NXT_62 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
                        RET
COMPARISION_16R :       CMP     CH, '='
                        JNZ     NXT_OPERATOR_1
                        CMP     AX, BX
                        JZ      NXT_1
                        RET
NXT_1 :                 MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_1 :        CMP     CH, '≠'
                        JNZ     NXT_OPERATOR_2
                        CMP     AX, BX
                        JNZ     NXT_21
                        RET
NXT_21 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_2 :        MOV     BP, SP
                        SUB     BP, 0002H
                        CMP     CH, '<'
                        JNZ     NXT_OPERATOR_3
                        TEST    AH, 10H
                        JZ      NXT_3
                        TEST    BH, 10H
                        JZ      NXT_32
                        MOV     AL, BH
                        AND     AX, 7C7CH
                        CMP     AH, AL
                        JB      NXT_32
                        JE      NXT_31
                        RET
NXT_31 :                MOV     AX, (BP)
                        AND     AX, 02FFH
                        AND     BX, 02FFH
                        CMP     AX, BX
                        JB      NXT_32
                        RET
NXT_3 :                 TEST    BH, 10H
                        JZ      NXT_34
                        RET
NXT_34 :                MOV     AL, BH
                        AND     AX, 7C7CH
```

```
                        CMP     AH, AL
                        JB      NXT_32
                        JE      NXT_35
                        RET
NXT_35 :                MOV     AX, (BP)
                        AND     AX, 02FFH
                        AND     DX, 02FFH
                        CMP     AX, BX
                        JB      NXT_32
                        RET
NXT_32 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_3 :        CMP     CH, '<='
                        JNZ     NXT_OPERATOR_4
                        TEST    AH, 10H
                        JZ      NXT_4
                        TEST    BH, 10H
                        JZ      NXT_42
                        MOV     AL, BH
                        AND     AX, 7C7CH
                        CMP     AH, AL
                        JB      NXT_42
                        JE      NXT_41
                        RET
NXT_41 :                MOV     AX, (BP)
                        AND     AX, 02FFH
                        MOV     DX, BX
                        AND     DX, 02FFH
                        CMP     AX, DX
                        JA      NXT_42
                        RET
NXT_4 :                 TEST    BH, 10H
                        JZ      NXT_44
                        RET
NXT_44 :                MOV     AL, BH
                        AND     AX, 7C7CH
                        CMP     AH, AL
                        JB      NXT_42
                        JE      NXT_45
                        RET
NXT_45 :                MOV     AX, (BP)
                        AND     AX, 02FFH
                        MOV     DX, BX
                        AND     DX, 02FFH
                        CMP     AX, DX
                        JB      NXT_42
                        RET
NXT_42 :                MOV     CL, FFH
```

```
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_4 :        CMP     CH, '>'
                        JNZ     NXT_OPERATOR_5
                        TEST    AH, 10H
                        JZ      NXT_5
                        TEST    BH, 10H
                        JNZ     NXT_51
                        RET
NXT_51 :                MOV     AL, BH
                        AND     AX, 7C7CH
                        CMP     AH, AL
                        JA      NXT_52
                        JE      NXT_53
                        RET
NXT_53 :                MOV     AX, (BP)
                        AND     AX, 02FFH
                        MOV     DX, BX
                        AND     DX, 02FFH
                        CMP     AX, DX
                        JB      NXT_52
                        RET
NXT_5 :                 TEST    BH, 10H
                        JNZ     NXT_52
                        MOV     AL, BH
                        AND     AX, 7C7CH
                        CMP     AH, AL
                        JA      NXT_52
                        JE      NXT_54
                        RET
NXT_54 :                MOV     AX, (BP)
                        AND     AX, 02FFH
                        MOV     DX, AX
                        AND     DX, 02FFH
                        CMP     AX, DX
                        JA      NXT_52
                        RET
NXT_52 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_5 :        CMP     CH, '>='
                        JZ      NXT_6
                        RET
NXT_6 :                 TEST    AH, 10H
                        JZ      NXT_60
                        TEST    BH, 10H
                        JNZ     NXT_61
                        RET
NXT_61 :                MOV     AL, BH
                        AND     AX, 7C7CH
```

```
                        CMP     AH, AL
                        JA      NXT_62
                        JE      NXT_63
                        RET
NXT_63 :                MOV     AX, (BP)
                        AND     AX, 02FFH
                        MOV     DX, BX
                        AND     DX, 02FFH
                        CMP     AX, DX
                        JBE     NXT_62
                        RET
NXT_60 :                TEST    BH, 10H
                        JNZ     NXT_62
                        MOV     AL, BH
                        AND     AX, 7C7CH
                        CMP     AH, AL
                        JA      NXT_62
                        JE      NXT_64
                        RET
NXT_64 :                MOV     AX, (BP)
                        AND     AX, 02FFH
                        MOV     DX, AX
                        AND     DX, 02FFH
                        CMP     AX, DX
                        JAE     NXT_62
                        RET
NXT_62 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
COMPARISION_24R :       MOV     BP, SP
                        MOV     DL, (BP)
                        DCR     BP
                        MOV     AX, (BP)
                        CMP     CH, '='
                        JNZ     NXT_OPERATOR_1
                        CMP     CL, DL
                        JNZ     NXT_11
                        CMP     AX, BX
                        JZ      NXT_12
NXT_11 :                RET
NXT_12 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_1 :        CMP     CH, '≠'
                        JNZ     NXT_OPERATOR_2
                        CMP     DL, CL
                        JZ      NXT_21
                        CMP     AX, DX
                        JNZ     NXT_22
NXT_21 :                RET
```

```
NXT_22 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
NXT_OPERATOR_2 :        CMP     CH, '<'
                        JNZ     NXT_OPERATOR_3
                        TEST    DL, 10H
                        JZ      NXT_30
                        TEST    CL, 10H
                        JZ      NXT_32
                        CMP     DL, CL
                        JB      NXT_32
                        JE      NXT_31
                        RET
NXT_31 :                CMP     AX, BX
                        JB      NXT_32
                        RET
NXT_30 :                CMP     CL, 10H
                        JZ      NXT_33
                        RET
NXT_33 :                CMP     CL, DL
                        JB      NXT_32
                        JE      NXT_34
                        RET
NXT_34 :                CMP     AX, BX
                        JB      NXT_32
                        RET
NXT_32 :                MOV     CL, FFH
                        JMP     LABEL_GTO_RUN
                        RET
NXT_OPERATOR_3 :        CMP     CH, '<='
                        JNZ     NXT_OPERATOR_4
                        TEST    DL, 10H
                        JZ      NXT_40
                        TEST    CL, 10H
                        JZ      NXT_42
                        CMP     DL, CL
                        JB      NXT_42
                        JE      NXT_41
                        RET
NXT_41 :                CMP     AX, BX
                        JBE     NXT_42
                        RET
NXT_40 :                CMP     CL, 10H
                        JZ      NXT_43
                        RET
NXT_43 :                CMP     CL, DL
                        JB      NXT_42
                        JE      NXT_44
                        RET
```

```
NXT_44 :              CMP   AX, BX
                      JBE   NXT_42
                      RET
NXT_42 :              MOV   CL, FFH
                      JMP   LABEL_GTO_RUN
                      RET
NXT_OPERATOR_4 :      CMP   CH, '>'
                      JNZ   NXT_OPERATOR_5
                      TEST  DL, 10H
                      JZ    NXT_50
                      TEST  CL, 10H
                      JZ    NXT_51
                      RET
NXT_51 :              CMP   DL, CL
                      JA    NXT_52
                      JE    NXT_53
                      RET
NXT_53 :              CMP   AX, BX
                      JA    NXT_52
                      RET
NXT_50 :              CMP   CL, 10H
                      JNZ   NXT_52
                      CMP   DL, CL
                      JA    NXT_52
                      JE    NXT_54
                      RET
NXT_54 :              CMP   AX, BX
                      JA    NXT_52
                      RET
NXT_52 :              MOV   CL, FFH
                      JMP   LABEL_GTO_RUN
                      RET
NXT_OPERATOR_5 :      CMP   CH, '>='
                      JZ    NXT_6
                      JMP   ERROR_1
NXT_6 :               TEST  DL, 10H
                      JZ    NXT_60
                      TEST  CL, 10H
                      JZ    NXT_61
                      RET
NXT_61 :              CMP   DL, CL
                      JA    NXT_62
                      JE    NXT_63
                      RET
NXT_63 :              CMP   AX, BX
                      JAE   NXT_62
                      RET
```

```
NXT_60 :        CMP     CL, 10H
                JNZ     NXT_62
                CMP     DL, CL
                JA      NXT_62
                JE      NXT_64
                RET
NXT_64 :        CMP     AX, BX
                JA      NXT_62
                RET
NXT_62 :        MOV     CL, FFH
                JMP     LABEL_GTO_RUN
                RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_14 : RUN_ICP_FOR

ANATOMY :

THE COUNT OF ITERATIONS IS GENERATED THE PARAMETER OF LOOP IS INITIALISE THE NEXT SENTENCE IN THE SEQUENCE IS POINTED AND PROGRAM POINTER, DATA POINTER, COUNT ARE SAVED ON THE STACK.

REGISTER USAGE :

CX : MANAGING SEGMENT, COUNT OF ITERATIONS

REFERENCE MEMORY LOCATIONS :

PRESENT_DATA_SEG

SUBROUTINES :

DET_VAR_ADD

```
LABEL_FOR_RUN :     INR    SI
                    CALL   DET_VAR_ADD
                    INR    SI
                    MOV    DH, (SI)
                    INR    SI
                    MOV    DL, (SI)
                    MOV    DI, AX
                    MANAGE_SEG_O
                    MOV    CX, PRESENT_DATA_AREA_SEG
                    POP    DS
                    TEST   BL, 1CH
                    JNZ    CONT_1
                    MOV    CX, 1FFOH
CONT_1 :            MOV    SS, CX
                    MOV    (BP+DI), DH
                    SUB    DL, DH
                    MOV    CL, DL
                    INR    CL
                    MOV    CH, OOH
AGAIN :             INR    SI
                    CMP    SI, 0007H
                    JZ     NXT_1
                    JMP    AGAIN
                    DCR    SI
NXT_1 :             PUSH   SI
                    PUSH   DS
```

```
PUSH   DI
PUSH   SS
PUSH   CX
RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_15 : RUN_ICP_GTO

ANATOMY :

THE BLOCK WHERE THE JUMP IS TO BE MADE IS DETERMINED USING THE LINE NUMBER TABLE GENERATED AT INITIALISATION STAGE. REFER MODULE INITIALISATION AT RUN, LEVL_2 F.C.5.1. THE LINE NUMBER TABLE LOADS FF AS LINE NUMBER IF A PARTICULAR BLOCK CORRESPONDING TO ANOTHER PI AREA i.e. CHANGE IN STATUS COUNT. WHILE DETERMINING THE DESTINATION BLOCK CHANGE IN STATUS COUNT IS ALSO RECORDED, AND IF A CHANGE IN STATUS COUNT IS LOCATED ENDING SI MAGNITUDES ARE ALTERED ACCORDINGLY.

REGISTER USAGE :

DX : SEGMENT MANAGEMENT.
BL : STATUS PARAMETER.
CL : PI PARAMETER.

REFERENCE MEMORY LOCATIONS :

PRESENT_PI, PRESENT_STATUS_PI, ENDING_SI, NXT_ENDING_SI, MARK_SI, BASE_ADD_LN_NO_TAB

```
LABEL_GTO_RUN :     INR   SI
                    INCREMENT_SI
                    MOV   AH, (SI)
                    INR   SI
                    INCREMENT_SI
                    MOV   AL, (SI)
                    MOV   CX, SI
                    SHR   CX                    , 8 TIMES
                    MOV   DX, DS
                    CMP   DX, 0000H
                    JZ    CONT_1
                    ADD   CX, 0100H
                    MOV   DX, 0000H
                    MOV   DS, DX
                    MOV   DX, CX
                    SHL   CX
                    MOV   DI, CX
                    MOV   BP, BASE_ADD_LN_NO_TAB
                    MOV   [illegible]
                    MOV   BL, PRESENT_STATUS_PI
```

```
                        CMP     BH, 00H
                        INZ     FORTH_JUMP
                        JMP     BACK_JMP
FORTH_JUMP :            ADD     DI, 0002H
                        INR     CX
                        CMP     (BP+DI), FFH
                        JNZ     NXT_01
SET_COOUNT_P :          INR     BL
                        ADD     DI, 0002H
                        DCR     CX
                        CMP     (BP+DI), FFH
                        JNZ     NXT_01
                        JMP     SET_COUNT_P
NXT_01 :                CMP     AX, (BP+DI)
                        JA      FORTH_JMP
                        JB      FOUND_BLK
                        ADD     DX, CX
                        PUSH    DX
REPEAT_1 :              CMP     DX, 0100H
                        JB      NXT_02
                        MOV     CX, 1000H
                        MOV     DS, CX
                        SUB     DX, 0100H
NXT_02 :                SHL     DX                      , 8 TIMES
                        MOV     SS, DX
                        JMP     DET_END_SI
FOUND_BLK :             DCR     CX
                        ADD     DX, CX
                        PUSH    DX
REPEAT_2 :              CMP     DX, 0100H
                        JB      NXT_03
                        MOV     CX, 1000H
                        MOV     DS, CX
                        SUB     DX, 0100H
NXT_03 :                SHL     DX                      , 8 TIMES
                        MOV     SI, DX
                        INR     SI
CHK_NXT_LN_NO :         ADD     SI, 0008H
                        CMP     AX, (SI)
                        JB      CHK_NXT_LN_NO
                        JMP     DET_END_SI
BACK_JMP :              CMP     AX, (BP+DI)
                        JA      NXT_05
                        JE      FOUND_LN_NO_EQUAL
                        SUB     DI, 0002H
                        INR     CX
                        CMP     (BP+DI), FFH
                        JNZ     BACK_JMP
```

```
SET_COUNT_M :          SUB   DI, 0002H
                       INR   CX
                       CMP   (BP+DI), FFH
                       JNZ   BACK_JMP
                       JMP   SET_COUNT_M
FOUND_LN_NO_EQUAL :    SUB   DX, AX
                       PUSH  DX
                       JMP   REPEAT_1
NXT_05 :               SUB   DX, CX
                       PUSH  DX
                       JMP   REPEAT_2
                       MANAGE_SEG_0
DET_END_SI :           CMP   BL, PRESENT_STATUS_PI
                       JNZ   CONT_02
                       DCR   SI
                       POP   DX
                       POP   DS
                       RET
CONT_02 :              MOV   PRESENT_STATUS_PI, BL
                       MOV   CH, 00H
                       PUSH  SI
                       MOV   SI, BASE_ADD_PIT
                       MOV   CL, PRESENT_PI
                       ADD   CL, 03H
                       SHL   CL                    , 4 TIMES
                       ADD   SI, CX
                       MOV   CL, STATUS_DET_PI
                       SUB   CL, BL
                       SHL   CL                    , 2 TIMES
                       ADD   SI, CX
                       ADD   SI, 0002H
                       MOV   CX, (SI)
                       POP   DX
                       CMP   DX, 0100H
                       JAE   CONT_03
                       CMP   CX, 0100H
                       JB    CONT_04
                       MOV   ENDING_SI, 0000H
                       MOV   MARK_SI, FFH
                       SUB   DX, 00FFH
                       SHL   DX                    , 8 TIMES
                       MOV   NXT_ENDING_SI
CONT_05 :              POP   DS
                       DCR   SI
                       RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_16 : RUN_ICP_NXT

ANATOMY :

THE DIRECTION OF FLOW IS DETERMINED ON THE BASIS OF COUNT. IF THE PROGRAM FLOW WAS TO CONTINUE WITHIN THE LOOP, POINTERS AND COUNTS ARE REMANAGED. IF THE FLOW WAS CONTINUATION TO THE NEXT SENTENCE THAN STACK POINTER IS MANAGED.

```
LABEL_NXT_RUN :     POP   CX
                    DCR   CX
                    JZ    NXT_1
                    POP   SS
                    POP   DI
                    INR   (BP+DI)
                    POP   DS
                    POP   SI
                    PUSH  SI
                    PUSH  DS
                    PUSH  DI
                    PUSH  SS
                    PUSH  CX
                    RET
NXT_1 :             SUB   SP, 0AH
                    MOV   DX, 0000H
                    MOV   SS, DX
                    RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_17 : RUN_ICP_OQ

ANATOMY :

THE POINTER BEING ON THE BOUNDRY OF 256 BYTES OR NOT DECTATES DIFFERENT ACTIONS TO BE EXECUTED. IF ON 256 BYTE BOUNDRY THE ENTRIES OF PI AND DI ARE CONFIRMED, IN IC FIELD IF NOT ON THE BOUNDRY, THE POINTER IS INCREMENTED TO POINT A LOCATION JUST BEFORE THE BOUNDRY. REFER SECTION 5.

```
LABEL_OQ_RUN :      SUB    SI, 0003H
                    TEST   SI, 00FFH
                    JZ     CONT_1
                    ADD    SI, 0003H
                    CMP    (SI), 00H
                    JZ     CONT_2
                    JMP    ERROR_3
CONT_2 :            ADD    SI, 00FFH
                    AND    SI, 00FFH
                    DCR    SI
                    RET
CONT_1 :            ADD    SI, 0003H
                    MANAGE_SEG_0
                    CMP    (SI), PRESENT_PI
                    JZ     CONT_3
                    JMP    ERROR_3
CONT_3 :            INR    SI
                    CMP    (SI), PRESENT_DI
                    JZ     CONT_4
                    JMP    ERROR_3
CONT_4 :            RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1) F.C. 5.1

PART_18 : RUN_ICP_FF

ANATOMY :

THE ACTION GROUP FF SPECIFIES IGNORING THE SENTENCE.

```
LABEL_FF_RUN :      ADD    SI, 0004H
                    RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1,1)
F.C. 5.1

SUB_PART_1 : DET_VAR_ADD

ANATOMY :

IDENTIFIER OF THE VARIABLE IS LOADED IN REGISTER BL FOR FURTHER REFERENCES. USING BL ADDRESS OF SET OF SPECIFIED VARIABLE IS LOADED IN REGISTER AX. FURTHER WHEN THE VARIABLE HAS THE INDEX AS A VARIABLE OR A NUMERIC CONSTANT OR NO INDEX ARE SEPARATELY RECONCILED WITH. THE ROUTINE REFERS TO DATA REFERENCE TABLE, MODULE INITIALISATION_RUN, PART_2, LEVEL_2, FLOW CHART 5.1.

REGISTER USAGE :

AX : ADDRESS OF VARIABLE.
BL : FIELD OF VARIABLE.
CL : INDEX OF THE VARIABLE.
DL : TEMPORARY STORAGE.
DX : DEFAULT AREA ADDRESS.

REFERENCE MEMORY LOCATIONS :

PRESENT_DATA_SEG

```
DET_VAR_ADD :       MOV    BL, (SI)
                    TEST   BL, 20H
                    JNZ    NXT_0
                    JMP    NON_VAR_IND_NC
NXT_0 :             INR    SI
                    INCREMENT_SI_1
                    INR    SI
                    INCREMENT_SI_1
                    MOV    CL, (SI)
                    MOV    CH, 00H
                    TEST   BL, 80H
                    JNZ    VAR_NE_8I
                    SUB    CL, 'Q'
                    SHL    CL
                    MOV    DL, CL
                    SHL    CL
                    ADD    CL, DL
                    MOV    DI, CX
                    MOV    DX, 1FF0H
                    MOV    SS, DX
```

```
                        MOV     BP, 0000H
                        MOV     CL, (BP+DI)
                        MOV     DX, 0000H
                        MOV     SS, DX
                        DCR     SI
MACRO                   DECREMENT_SI_1
                        TEST    SI, 0007H
                        JNZ     CONT_0
                        DCR     SI
CONT_0 :                NOP
                        ENDM
                        JMP     DET_ADD
VAR_NE_8I :             MOV     BP, BASE_ADD_DRT
                        MOV     DI, 0000H
                        TEST    BL, 40H
                        JNZ     VAR_E_8I
                        ADD     DI, 0003H
;NORMAL_8I TYPE OF VARIABLE.
                        MOV     AX, (BP+DI)
                        ADD     DI, 0002H
                        SUB     CL, (BP+DI)
                        SHL     CL                      , 4 TIMES
                        JMP     LOAD_INDEX
VAR_E_8I :              ADD     DI, 0009H
                        MOV     AX, (BP+DI)
;EXTENDED_8I TYPE VARIABLE.
                        ADD     DI, 0002H
                        SUB     CL, (BP+DI)
                        SHL     CX                      , 8 TIMES
LOAD_INDEX :            ADD     AX, CX
                        MOV     DI, AX
                        MOV     BP, 0000H
                        MANAGE_SEG_0
                        MOV     CX, PRESENT_DATA_SEG
                        POP     DS
                        MOV     DS, CX
                        MOV     CL, (BP+DI)
                        MOV     DX, 0000H
                        MOV     SS, CX
                        DCR     DI
                        DECREMENT_SI_1
                        JMP     DET_ADD
NON_VAR_IND_NC :        TEST    BL, 80H
                        JZ      NXT_1
                        JMP     LOAD_NC
NXT_1 :                 MOV     CX, 0000H
                        TEST    BL, E0H
                        JNZ     DET_ADD
```

```
                    INR    SI
                    INCREMENT_SI_1
                    INR    SI
                    INCREMENT_SI_1
                    MOV    CL, (SI)
                    DCR    SI
                    DECREMENT_SI
DET_ADD :           TEST   BL, 1CH
                    JZ     NXT_2
                    JMP    NON_DEF_VAR
NXT_2 :             MOV    AX, 0000H
                    TEST   BL, 01H
                    JZ     DEF_REAL
                    ADD    AX, CX
                    MOV    CL, (SI)
                    SUB    CL, `0'
                    SHL    CL
                    MOV    DL, CL
                    SHL    CL
                    ADD    CL, DL
                    ADD    AX, DX
MACRO               CHK_NON_INDEX
                    TEST   BL, 20H
                    JNZ    CONT_1
                    TEST   BL, 40H
                    JNZ    CONT_2
CONT_1 :            INR    SI
                    INCREMENT_SI_1
CONT_2 :            NOP
                    ENDM
                    RET
NON_DEF_VAR :       MOV    DX, 0000H
                    MOV    SS, DX
                    MOV    BP, BASE_ADD_DRT
                    MOV    DI, 0000H
                    TEST   BL, 01H
                    JZ     NXT_3
                    JMP    REAL_VAR_ADD
NXT_3 :             MOV    DL, BL
                    ADD    DL, 1FH
                    CMP    DL, 05H
                    JNZ    NXT_4
                    MOV    CH, 00H
                    MOV    CL, (SI)
                    MOV    AX, (BP+DI)
                    ADD    DI, 0002H
                    SUB    CL, (BP+DI)
                    SHL    CL
```

```
                    ADD     AX, CX
                    CHK_NON_INDEX
                    RET
NXT_4 :             CMP     DL, 09H
                    JNZ     NXT_5
                    ADD     DI, 0003H
                    MOV     AX, (BP+DI)
                    MOV     CL, (SI)
                    ADD     DI, 0002H
                    SUB     CL, (BP+DI)
                    SHL     CL                      , 4 TIMES
                    ADD     AX, CX
                    CHK_NON_INDEX
                    RET
NXT_5 :             CMP     DL, 0DH
                    JNZ     NXT_6
                    ADD     DI, 0006H
                    MOV     AX, (BP+DI)
                    MOV     CH, 00H
                    SHL     CL
                    ADD     AX, CX
                    MOV     CL, (SI)
                    ADD     DI, 0002H
                    SUB     CL, (BP+DI)
                    SHL     CX                      , 5 TIMES
                    ADD     AX, CX
                    CHK_NON_INDEX
                    RET
NXT_6 :             CMP     DL, 11H
                    JNZ     NXT_7
                    ADD     DI, 0009H
                    MOV     AX, (BP+DI)
                    MOV     CH, 00H
                    ADD     AX, CX
                    MOV     CL, (SI)
                    ADD     DI, 0002H
                    SUB     CL, (BP+DI)
                    SHL     CX                      , 8 TIMES
                    ADD     AX, CX
                    CHK_NON_INDEX
                    RET
NXT_7               CMP     DL, 15H
                    JZ      NXT_8
                    JMP     ERROR_3
NXT_8 :             ADD     DI, 000CH
                    MOV     AX, (BP+DI)
                    MOV     CH, 00H
                    SHL     CL                      , 2 TIMES
```

```
                        ADD     AX, CX
                        MOV     CL, (SI)
                        ADD     DI, 0002H
                        SUB     CL, (BP+DI)
                        SHL     CX                      , 9 TIMES
                        ADD     AX, CX
                        CHK_NON_INDEX
                        RET
REAL_VAR_ADD :          MOV     DL, BL
                        AND     DL, 1FH
                        CMP     DL, 04H
                        JNZ     CONT_5
                        ADD     DI, 000FH
                        MOV     AX, (BP+DI)
                        MOV     CL, (SI)
                        MOV     CH, 0CH
                        ADD     DI, 0002H
                        SUB     CL, (BP+DI)
                        MOV     DL, CL
                        SHL     CL
                        ADD     AX, CX
                        CHK_NON_INDEX
                        RET
CONT_5 :                CMP     DL, 08H
                        JNZ     CONT_6
                        ADD     DI, 0012H
                        MOV     AX, (BP+DI)
                        MOV     CH, 00H
                        SHL     CL
                        ADD     AX, CX
                        MOV     CL, (SI)
                        ADD     DI, 0002H
                        SUB     CL, (BP+DI)
                        SHL     CX                      , 5 TIMES
                        ADD     AX, CX
                        CHK_NON_INDEX
                        RET
CONT_6 :                CMP     DL, 0CH
                        JNZ     CONT_7
                        ADD     DI, 0015H
                        MOV     AX, (BP+DI)
                        MOV     CH, 00H
                        MOV     DL, CL
                        SHL     CL
                        ADD     CL, DL
                        ADD     AX, CX
                        MOV     CL, (SI)
                        ADD     DI, 0002H
```

```
                    SUB     CL, (BP+DI)
                    MOV     DL, CL
                    SHL     CL
                    ADD     CL, DL
                    SHL     CX                      , 4 TIMES
                    ADD     AX, CX.
                    CHK_NON_INDEX
                    RET
CONT_7 :            MOV     DL, BL
                    AND     DL, 1FH
                    CMP     DL, 10H
                    JNZ     CONT_8
                    ADD     DI, 0018H
                    MOV     AX, (BP+DI)
                    MOV     CH, 00H
                    SHL     CL
                    ADD     AX, CX
                    MOV     CL, (SI)
                    ADD     DI, 0002H
                    SUB     CL, (BP+DI)
                    SHL     CX                      , 9 TIMES
                    ADD     AX, CX
                    CHK_NON_INDEX
                    RET
CONT_8 :            CMP     DL, 10H
                    JZ      CONT_9
                    JMP     ERROR_3
CONT_9 :            ADD     DI, 001BH
                    MOV     AX, (BP+DI)
                    MOV     DL, CL
                    ADD     CL, DL
                    MOV     CH, 00H
                    ADD     AX, CX
                    MOV     CL, (SI)
                    ADD     DI, 0002H
                    SUB     CL, (BP+DI)
                    MOV     DL, CL
                    SHL     CL
                    ADD     CL, DL
                    SHL     CX                      , 8 TIMES
                    ADD     AX, CX
                    CHK_NON_INDEX
                    RET
```

TITLE : MODULE : SUB : RSLV_RUN_GROUP_EXEC, LEVEL_3(1,1)
F.C. 5.1

SUB_PART_2 : BACK_CONVERSION

ANATOMY :

SUB_PART_2_1 : BACK_CONVERT_24R :

THE NUMBER IS CONVERTED FROM FLOATING POINT REPRESENTATION TO NATURAL BINARY FORM. THE NATURAL BINARY NUMBER THUS FORMED IS CONVERTED TO REPRESENT THE AMGNITUDE OF THE DESIRED ACCURACY AND A CORRESPONDING EXPONENT OF 10.

FURTHER THE CORRECTED MAGNITUDE IS CONVERTED INTO THE BCD FORM. THE ROUTINE INVOLVES MULTIPLE JUMPS, THEREFORE THE LENGTH APPEARS A' BIT MORE, THOUGH THE ACTUAL EXECUTION ENCOUNTERED FOR A CONVERSION WILL NEVER FLOW THROUGH THE LENGTH.

REGISTER USAGE :

AX : ADDRESS OF VARIABLE, CRUNCHING NUMBERS.
BH : SIGN OF THE NUMBER.
BL : FIELD OF VARIABLE, SIGN OF EXPONENT AND EXPONENT OF TEN.
DX : TEMPORARY STORAGE.
AL : EXPONENT OF 2.

REFERENCE MEMORY LOCATIONS :

MSWFM, LSWFM, PRESENT_DATA_SEG.

```
BACK_CONVERT_24R :    MOV   DI, AX
                      MANAGE_SEG_0
                      MOV   CX, PRESENT_DATA_SEG
                      POP   DS
                      TEST  BL, 1CH
                      JNZ   CONT_0
                      MOV   CX, 1FF0H
CONT_0 :              MOV   SS, CX
                      MOV   AL, (BP+DI)
                      MOV   BH, 2EH
                      TEST  AL, 80H
                      JNZ   CONT_1
```

```
                        INR     BH
CONT_1 :                AND     AL, 7EH
                        SHR     AL
                        CMP     AL, 20H
                        JB      RSLV_NEG_CHAR
                        SUB     AL, 20H
                        CMP     AL, 11H
                        JAE     CONT_2
                        JMP     NEG_EXP_OF_TEN
CONT_2 :                SUB     AL, 11H
                        CMP     AL, 00H
                        JNZ     NXT_11
                        JMP     RSLV_PE_0
NXT_11 :                CMP     AL, 01H
                        JNZ     NXT_12
                        JMP     RSLV_PE_1
NXT_12 :                CMP     AL, 02H
                        JNZ     NXT_13
                        JMP     RSLV_PE_2
NXT_13 :                CMP     AL, 03H
                        JNZ     NXT_14
                        JMP     RSLV_PE_3
NXT_14 :                CMP     AL, 04H
                        JNZ     NXT_15
                        JMP     RSLV_PE_4
NXT_15 :                CMP     AL, 05H
                        JNZ     NXT_16
                        JMP     RSLV_PE_5
NXT_16 :                CMP     AL, 06H
                        JNZ     NXT_17
                        JMP     RSLV_PE_6
NXT_17 :                CMP     AL, 07H
                        JNZ     NXT_18
                        JMP     RSLV_PE_7
NXT_18 :                CMP     AL, 08H
                        JNZ     NXT_19
                        JMP     RSLV_PE_8
NXT_19 :                CMP     AL, 09H
                        JNZ     NXT_1A
                        JMP     RSLV_PE_9
NXT_1A :                CMP     AL, 0AH
                        JNZ     NXT_1B
                        JMP     RSLV_PE_A
NXT_1B :                CMP     AL, 0BH
                        JNZ     NXT_1C
                        JMP     RSLV_PE_B
NXT_1C :                CMP     AL, 0CH
                        JNZ     NXT_1D
```

```
                        JMP     RSLV_PE_C
NXT_1D :                CMP     AL, 0DH
                        JNZ     NXT_1E
                        JMP     RSLV_PE_D
NXT_1E :                CMP     AL, 0EH
                        JNZ     NXT_1F
                        JMP     ERROR_3
NEG_EXP_OF_TEN :        CMP     AL, 10H
                        JNZ     NXT_21
                        JMP     RSLV_NE_1
NXT_21 :                CMP     AL, 0FH
                        JNZ     NXT_22
                        JMP     RSLV_NE_2
NXT_22 :                CMP     AL, 0EH
                        JNZ     NXT_23
                        JMP     RSLV_NE_3
NXT_23 :                CMP     AL, 0DH
                        JNZ     NXT_24
                        JMP     RSLV_NE_4
NXT_24 :                CMP     AL, 0CH
                        JNZ     NXT_25
                        JMP     RSLV_NE_5
NXT_25 :                CMP     AL, 0BH
                        JNZ     NXT_26
                        JMP     RSLV_NE_6
NXT_26 :                CMP     AL, 0AH
                        JNZ     NXT_27
                        JMP     RSLV_NE_7
NXT_27 :                CMP     AL, 09H
                        JNZ     NXT_28
                        JMP     RSLV_NE_8
NXT_28 :                CMP     AL, 08H
                        JNZ     NXT_29
                        JMP     RSLV_NE_9
NXT_29 :                CMP     AL, 07H
                        JNZ     NXT_2A
                        JMP     RSLV_NE_10
NXT_2A :                CMP     AL, 06H
                        JNZ     NXT_2B
                        JMP     RSLV_NE_11
NXT_2B :                CMP     AL, 05H
                        JNZ     NXT_2C
                        JMP     RSLV_NE_12
NXT_2C :                CMP     AL, 04H
                        JNZ     NXT_2D
                        JMP     RSLV_NE_13
NXT_2D :                CMP     AL, 03H
                        JNZ     NXT_2E
```

```
                        JMP     RSLV_NE_14
NXT_2E :                CMP     AL, 02H
                        JNZ     NXT_2F
                        JMP     RSLV_NE_15
NXT_2F :                CMP     AL, 01H
                        JNZ     NXT_30
                        JMP     RSLV_NE_16
NXT_30 :                JMP     RSLV_NE_17
RSLV_NEG_CHAR :         SUB     AL, 20H
                        SUB     AL, 11H
                        CMP     AL, 00H
                        JNZ     NXT_31
                        JMP     RSLV_NE_49
NXT_31 :                CMP     AL, 01H
                        JNZ     NXT_32
                        JMP     RSLV_NE_48
NXT_32 :                CMP     AL, 02H
                        JNZ     NXT_33
                        JMP     RSLV_NE_47
NXT_33 :                CMP     AL, 03H
                        JNZ     NXT_34
                        JMP     RSLV_NE_46
NXT_34 :                CMP     AL, 04H
                        JNZ     NXT_35
                        JMP     RSLV_NE_45
NXT_35 :                CMP     AL, 05H
                        JNZ     NXT_36
                        JMP     RSLV_NE_44
NXT_36 :                CMP     AL, 06H
                        JNZ     NXT_37
                        JMP     RSLV_NE_43
NXT_37 :                CMP     AL, 07H
                        JNZ     NXT_38
                        JMP     RSLV_NE_42
NXT_38 :                CMP     AL, 08H
                        JNZ     NXT_39
                        JMP     RSLV_NE_41
NXT_39 :                CMP     AL, 09H
                        JNZ     NXT_3A
                        JMP     RSLV_NE_40
NXT_3A :                CMP     AL, 0AH
                        JNZ     NXT_3B
                        JMP     RSLV_NE_39
NXT_3B :                CMP     AL, 0BH
                        JNZ     NXT_3C
                        JMP     RSLV_NE_38
NXT_3C :                CMP     AL, 0CH
                        JNZ     NXT_3D
```


```
                        JMP     RSLV_NE_37
NXT_3D :                CMP     AL, 0DH
                        JNZ     NXT_3E
                        JMP     RSLV_NE_36
NXT_3E :                CMP     AL, 0EH
                        JNZ     NXT_3F
                        JMP     RSLV_NE_35
NXT_3F :                CMP     AL, 0FH
                        JNZ     NXT_40
                        JMP     RSLV_NE_34
NXT_40 :                CMP     AL, 10H
                        JNZ     NXT_41
                        JMP     RSLV_NE_33
NXT_41 :                CMP     AL, 11H
                        JNZ     NXT_42
                        JMP     RSLV_NE_32
NXT_42 :                CMP     AL, 12H
                        JNZ     NXT_43
                        JMP     RSLV_NE_31
NXT_43 :                CMP     AL, 13H
                        JNZ     NXT_44
                        JMP     RSLV_NE_30
NXT_44 :                CMP     AL, 14H
                        JNZ     NXT_45
                        JMP     RSLV_NE_29
NXT_45 :                CMP     AL, 15H
                        JNZ     NXT_46
                        JMP     RSLV_NE_28
NXT_46 :                CMP     AL, 16H
                        JNZ     NXT_47
                        JMP     RSLV_NE_27
NXT_47 :                CMP     AL, 17H
                        JNZ     NXT_48
                        JMP     RSLV_NE_26
NXT_48 :                CMP     AL, 18H
                        JNZ     NXT_49
                        JMP     RSLV_NE_25
NXT_49 :                CMP     AL, 19H
                        JNZ     NXT_4A
                        JMP     RSLV_NE_24
NXT_4A :                CMP     AL, 1AH
                        JNZ     NXT_4B
                        JMP     RSLV_NE_23
NXT_4B :                CMP     AL, 1BH
                        JNZ     NXT_4C
                        JMP     RSLV_NE_22
NXT_4C :                CMP     AL, 1CH
                        JNZ     NXT_4D
```

```
                        JMP     RSLV_NE_21
NXT_4D :                CMP     AL, 1DH
                        JNZ     NXT_4E
                        JMP     RSLV_NE_20
NXT_4E :                CMP     AL, 1EH
                        JNZ     NXT_4F
                        JMP     RSLV_NE_19
NXT_4F :                CMP     AL, 1FH
                        JNZ     NXT_50
                        JMP     RSLV_NE_18
NXT_50 :                JMP     ERROR_3
RSLV_PE_0 :             MOV     BL, 00H
                        INR     DI
                        MOV     AX, (BP+DI)
                        MOV     DX, 0001H
                        JMP     LOAD_KBB
RSLV_PE_1 :             MOV     BL, 00H
                        MOV     CL, 02H
                        CALL    MULT_INT
                        JMP     LOAD_KBB
RSLV_PE_2 :             MOV     BL, 00H
                        MOV     CL, 04H
                        CALL    MULT_INT
                        JMP     LOAD_KBB
RSLV_PE_3 :             MOV     BL, 00H
                        MOV     CL, 08H
                        INR     DI
                        MOV     AX, (BP+DI)
                        CMP     AX, E848H
                        JAE     NXT_31
                        DCR     DI
                        CALL    MULT_INT
                        JMP     NXT_32
NXT_31 :                MOV     CX, CCCCH
                        DCR     DI
                        CALL    MULT_FRACT
                        INR     BL
NXT_32 :                JMP     LOAD_KBB
RSLV_PE_4 :             MOV     BL, 01H
                        MOV     CX, 9999H
                        CALL    MULT_FRACT
                        MOV     CL, 01H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
```

```
RSLV_PE_5 :     MOV   BL, 01H
                MOV   CX, 3333H
                CALL  MULT_FRACT
                MOV   CL, 03H
                CALL  MULT_INT
                CLC
                ADD   AX, LSWFM
                ADC   DX, MSWFM
                JMP   LOAD_KBB
RSLV_PE_6 :     MOV   BL, 01H
                MOV   CX, 6666H
                CALL  MULT_FRACT
                MOV   CL, 06H
                CALL  MULT_INT
                CLC
                ADD   AX, LSWFM
                ADC   DX, MSWFM
                JMP   LOAD_KBB
RSLV_PE_7 :     MOV   BL, 02H
                MOV   CX, 47ADH
                CALL  MULT_FRACT
                MOV   CL, 06H
                CALL  MULT_INT
                CLC
                ADD   AX, LSWFM
                ADC   DX, MSWFM
                JMP   LOAD_KBB
RSLV_PE_8 :     MOV   BL, 02H
                MOV   CX, 8F5CH
                CALL  MULT_FRACT
                MOV   CL, 02H
                CALL  MULT_INT
                CLC
                ADD   AX, LSWFM
                ADC   DX, MSWFM
                JMP   LOAD_KBB
RSLV_PE_9 :     MOV   BL, 02H
                MOV   CX, 1DB8H
                CALL  MULT_FRACT
                MOV   CL, 05H
                CALL  MULT_INT
                CLC
                ADD   AX, LSWFM
                ADC   DX, MSWFM
                JMP   LOAD_KBB
RSLV_PE_A :     MOV   BL, 03H
                MOV   CX, 0626H
                CALL  MULT_FRACT
```

```
                        MOV     CL, 01H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_PE_B :             MOV     BL, 03H
                        MOV     CX, 0C48H
                        CALL    MULT_FRACT
                        MOV     CL, 02H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_PE_C :             MOV     BL, 03H
                        MOV     CX, 1890H
                        CALL    MULT_FRACT
                        MOV     CL, 04H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_PE_D :             MOV     BL, 03H
                        MOV     CX, 3136H
                        INR     DI
                        MOV     AX, (BP+DI)
                        CMP     AX, DCD6H
                        JAE     NXT_D1
                        DCR     DI
                        CALL    MULT_FRACT
                        MOV     CL, 08H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     NXT_D2
NXT_D1 :                MOV     CX, D1C7H
                        INR     BL
                        CALL    MULT_FRACT
NXT_D2 :                JMP     LOAD_KBB
RSLV_PE_E :             MOV     BL; 04H
                        MOV     CX, A36DH
                        CALL    MULT_FRACT
                        MOV     CL, 01H
                        CALL    MULT_INT
                        CLC
```

```
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_1 :             MOV     BL, 81H
                        MOV     CL, 05H
                        CALL    MULT_INT
                        JMP     LOAD_KBB
RSLV_NE_2 :             MOV     BL, 81H
                        MOV     CX, 8000H
                        CALL    MULT_FRACT
                        MOV     CL, 02H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_3 :             MOV     BL, 81H
                        MOV     CX, 4000H
                        CALL    MULT_FRACT
                        MOV     CL, 01H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_4 :             MOV     BL, 82H
                        MOV     CX, 4000H
                        CALL    MULT_FRACT
                        MOV     CL, 06H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_5 :             MOV     BL, 82H
                        MOV     CX, 2000H
                        CALL    MULT_FRACT
                        MOV     CL, 03H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_6 :             MOV     BL, 82H
                        MOV     CX, 9000H
                        CALL    MULT_FRACT
                        MOV     CL, 01H
                        CALL    MULT_INT
```

```
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_7 :             MOV     BL, 83H
                        MOV     CX, D000H
                        CALL    MULT_FRACT
                        MOV     CL, 07H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_8 :             MOV     BL, 83H
                        MOV     CX, E70AH
                        CALL    MULT_FRACT
                        MOV     CL, 03H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_9 :             MOV     BL, 83H
                        MOV     CX, F3FEH
                        CALL    MULT_FRACT
                        MOV     CL, 01H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_10 :            MOV     BL, 83H
                        MOV     CX, F9FBH
                        CALL    MULT_FRACT
                        JMP     LOAD_KBB
RSLV_NE_11 :            MOV     BL, 84H
                        MOV     CX, E1FFH
                        CALL    MULT_FRACT
                        MOV     CL, 04H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_12 :            MOV     BL, 84H
                        MOV     CX, 70FFH
                        CALL    MULT_FRACT
                        MOV     CL, 02H
```

```
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_13              MOV     BL, 84H
                        MOV     CX, 387FH
                        CALL    MULT_FRACT
                        MOV     CL, 01H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_14 :            MOV     BL, 85H
                        MOV     CX, 1A7EH
                        CALL    MULT_FRACT
                        MOV     CL, 06H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_15 :            MOV     BL, 85H
                        MOV     CX, 0D3FH
                        CALL    MULT_FRACT
                        MOV     CL, 03H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_16 :            MOV     BL, 85H
                        MOV     CX, 869AH
                        CALL    MULT_FRACT
                        MOV     CL, 01H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_17 :            MOV     BL, 86H
                        MOV     CX, A119H
                        CALL    MULT_FRACT
                        MOV     CL, 07H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
```

```
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_18 :            MOV     BL, 86H
                        MOV     CX, D0D9H
                        CALL    MULT_FRACT
                        MOV     CL, 03H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_19 :            MOV     BL, 86H
                        MOV     CX, E844H
                        CALL    MULT_FRACT
                        MOV     CL, 01H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_20 :            MOV     BL, 86H
                        MOV     CX, F425H
                        CALL    MULT_FRACT
                        JMP     LOAD_KBB
RSLV_NE_21 :            MOV     BL, 87H
                        MOV     CX, C4D0H
                        CALL    MULT_FRACT
                        MOV     CL, 04H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_22 :            MOV     BL, 87H
                        MOV     CX, 8254H
                        CALL    MULT_FRACT
                        MOV     CL, 02H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_23 :            MOV     BL, 87H
                        MOV     CX, 3126H
                        CALL    MULT_FRACT
                        MOV     CL, 01H
                        CALL    MULT_INT
                        CLC
```

```
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_24 :            MOV     BL, 88H
                        MOV     CX, 5FC2H
                        CALL    MULT_FRACT
                        MOV     CL, 05H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_25 :            MOV     BL, 88H
                        MOV     CX, FAE1H
                        CALL    MULT_FRACT
                        MOV     CL, 02H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_26 :            MOV     BL, 88H
                        MOV     CX, 7D77H
                        CALL    MULT_FRACT
                        MOV     CL, 01H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_27 :            MOV     BL, 89H
                        MOV     CX, 7333H
                        CALL    MULT_FRACT
                        MOV     CL, 07H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_28 :            MOV     BL, 89H
                        MOV     CX, B9ADH
                        CALL    MULT_FRACT
                        MOV     CL, 03H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
```

```
RSLV_NE_29 :        MOV   BL, 89H
                    MOV   CX, DCD3H
                    CALL  MULT_FRACT
                    MOV   CL, 01H
                    CALL  MULT_INT
                    CLC
                    ADD   [illegible]
                    ADC   DX, MSWFM
                    JMP   LOAD_KBB
RSLV_NE_30 :        MOV   BL, 89H
                    MOV   CX, EE69H
                    CALL  MULT_FRACT
                    JMP   LOAD_KBB
RSLV_NE_31 :        MOV   BL, 90H
                    MOV   CX, A816H
                    CALL  MULT_FRACT
                    MOV   CL, 04H
                    CALL  MULT_INT
                    CLC
                    ADD   AX, LSWFM
                    ADC   DX, MSWFM
                    JMP   LOAD_KBB
RSLV_NE_32 :        MOV   BL, 90H
                    MOV   CX, 540BH
                    CALL  MULT_FRACT
                    MOV   CL, 02H
                    CALL  MULT_INT
                    CLC
                    ADD   AX, LSWFM
                    ADC   DX, MSWFM
                    JMP   LOAD_KBB
RSLV_NE_33 :        MOV   BL, 90H
                    MOV   CX, 2A02H
                    CALL  MULT_FRACT
                    MOV   CL, 01H
                    CALL  MULT_INT
                    CLC
                    ADD   AX, LSWFM
                    ADC   DX, MSWFM
                    JMP   LOAD_KBB
RSLV_NE_34 :        MOV   BL, 91H
                    MOV   CX, D1EBH
                    CALL  MULT_FRACT
                    MOV   CL, 05H
                    CALL  MULT_INT
                    CLC
                    ADD   AX, LSWFM
                    ADC   DX, MSWFM
```

```
                JMP     LOAD_KBB
RSLV_NE_35 :    MOV     BL, 91H
                MOV     CX, E8F5H
                CALL    MULT_FRACT
                MOV     CL, 02H
                CALL    MULT_INT
                CLC
                ADD     AX, LSWFM
                ADC     DX, MSWFM
                JMP     LOAD_KBB
RSLV_NE_36 :    MOV     BL, 91H
                MOV     CX, 747AH
                CALL    MULT_FRACT
                MOV     CL, 01H
                CALL    MULT_INT
                CLC
                ADD     AX, LSWFM
                ADC     DX, MSWFM
                JMP     LOAD_KBB
RSLV_NE_37 :    MOV     BL, 92H
                MOV     CX, 46A1H
                CALL    MULT_FRACT
                MOV     CL, 07H
                CALL    MULT_INT
                CLC
                ADD     AX, LSWFM
                ADC     DX, MSWFM
                JMP     LOAD_KBB
RSLV_NE_38 :    MOV     BL, 92H
                MOV     CX, A34DH
                CALL    MULT_FRACT
                MOV     CL, 03H
                CALL    MULT_INT
                CLC
                ADD     AX, LSWFM
                ADC     DX, MSWFM
                JMP     LOAD_KBB
RSLV_NE_39 :    MOV     BL, 92H
                MOV     CX, D1A3H
                CALL    MULT_FRACT
                MOV     CL, 01H
                CALL    MULT_INT
                CLC
                ADD     AX, LSWFM
                ADC     DX, MSWFM
                JMP     LOAD_KBB
RSLV_NE_40 :    MOV     BL, 92H
                MOV     CX, E8CEH
```

```
                        CALL    MULT_FRACT
                        JMP     LOAD_KBB
RSLV_NE_41 :            MOV     BL, 93H
                        MOV     CX, 8C08H
                        CALL    MULT_FRACT
                        MOV     CL, 01H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_42 :            MOV     BL, 93H
                        MOV     CX, 4611H
                        CALL    MULT_FRACT
                        MOV     CL, 02H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_43 :            MOV     BL, 93H
                        MOV     CX, 5E35H
                        CALL    MULT_FRACT
                        MOV     CL, 01H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_44 :            MOV     BL, 94H
                        MOV     CX, AF2EH
                        CALL    MULT_FRACT
                        MOV     CL, 05H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
RSLV_NE_45 :            MOV     BL, 94H
                        MOV     CX, D793H
                        CALL    MULT_FRACT
                        MOV     CL, 02H
                        CALL    MULT_INT
                        CLC
                        ADD     AX, LSWFM
                        ADC     DX, MSWFM
                        JMP     LOAD_KBB
```

```
RSLV_NE_46 :    MOV   BL, 94H
                MOV   CX, 6D0FH
                CALL  MULT_FRACT
                MOV   CL, 01H
                CALL  MULT_INT
                CLC
                ADD   AX, LSWFM
                ADC   DX, MSWFM
                JMP   LOAD_KBB
RSLV_NE_47 :    MOV   BL, 95H
                MOV   CX, 2A57H
                CALL  MULT_FRACT
                MOV   CL, 07H
                CALL  MULT_INT
                CLC
                ADD   AX, LSWFM
                ADC   DX, MSWFM
                JMP   LOAD_KBB
RSLV_NE_48 :    MOV   BL, 95H
                MOV   CX, 8CCCH
                CALL  MULT_FRACT
                MOV   CL, 03H
                CALL  MULT_INT
                CLC
                ADD   AX, LSWFM
                ADC   DX, MSWFM
                JMP   LOAD_KBB
RSLV_NE_49 :    MOV   BL, 95H
                MOV   CX, C6A7H
                CALL  MULT_FRACT
                MOV   CL, 01H
                CALL  MULT_INT
                CLC
                ADD   AX, LSWFM
                ADC   DX, MSWFM
                JMP   LOAD_KBB
LOAD_KBB :      MOV   BP, 0000H
                MOV   DI, BASE_ADD_KBB
                MOV   (BP+DI), BH
                MOV   CX, 2710H
                DIV   DX, CX
                MOV   CL, 0AH
                DIV   AX, CL
                INR   DI
                MOV   (BP+DI), AL
                INR   DI
                MOV   (BP+DI), AH
                MOV   AX, DX
```

```
                        MOV     CX, 03E8H
                        MOV     DX, 0000H
                        DIV     DX, CX
                        INR     DI
                        MOV     (BP+DI), AL
                        MOV     AX, DX
                        MOV     CL, 64H
                        DIV     AX, CL
                        INR     DI
                        MOV     (BP+DI), AL
                        MOV     AL, AH
                        MOV     AH, 00H
                        MOV     CL, 0AH
                        DIV     AX, CL
                        INR     DI
                        MOV     (BP+DI), AL
                        INR     DI
                        MOV     (BP+DI), AH
                        MOV     BH, '+'
                        TEST    BL, 80H
                        JZ      NXT_1
                        MOV     BH, '-'
NXT_1 :                 INR     DI
                        MOV     (BP+DI), BH
                        MOV     AL, BL
                        AND     AL, 70H
                        SHR     AL                      , 4 TIMES
                        INR     DI
                        AND     BL, 0FH
                        INR     DI
                        MOV     (BP+DI), AL
                        RET
MULT_FRACT :            MOV     DX, 0000H
                        MOV     AX, (BP+DI)
                        MUL     AX, CX
                        MOV     AX, DX
                        ADD     AX, CX
                        JNC     NXT_2
                        ADD     DL, 01H
NXT_2 :                 MOV     MSWFM, DX
                        MOV     LSWFM, AX
;DX AND AX CONTENTS ARE SAVED FOR FURTHER CALCULATIONS.
                        DCR     DI
                        RET
MULT_INT :              INR     SI
                        MOV     AX, (BP+DI)
                        MOV     DX, 0000H
                        CMP     CL, 01H
```

```
                    JA      MULT_2
                    RET
MULT_2 :            CMP     CL, 02H
                    JA      MULT_3
                    SHL     AX
                    ROL     DX
                    RET
MULT_3 :            CMP     CL, 03H
                    JA      MULT_4
                    SHL     AX
                    ROL     DX
                    ADD     AX, (BP+DI)
                    JNC     NXT_3
                    INR     DX
NXT_3 :             INR     DX
                    RET
MULT_4 :            CMP     CL, 04H
                    JA      MULT_5
                    SHL     AX
                    ROL     DX
                    RET
MULT_5 :            CMP     CL, 05H
                    JA      MULT_6
                    SHL     AX
                    ROL     DX
                    SHL     AX
                    ROL     DX
                    ADD     AX, (BP+DI)
                    JNC     NXT_4
                    INR     DX
NXT_4 :             INR     DX
                    RET
MULT_6 :            CMP     CL, 06H
                    JA      MULT_7
                    SHL     AX
                    ROL     DX
                    ADD     AX, (BP+DI)
                    JNC     NXT_5
                    INR     DX
NXT_5 :             INR     DX
                    SHL     AX
                    ROL     DX
                    RET
MULT_7 :            CMP     CL, 07H
                    JA      MULT_8
                    SHL     AX
                    ROL     DX
                    ADD     AX, (BP+DI)
```

```
                        JNC     NXT_6
                        INR     DX
NXT_6 :                 INR     DX
                        SHL     AX
                        ROL     DX
                        ADD     AX, (BP+DI)
                        JNC     NXT_7
                        INR     DX
NXT_7 :                 INR     DX
                        RET
MULT_8 :                CMP     CL, 08H
                        JE      NXT_8
                        JMP     ERROR_3
NXT_8 :                 SHL     AX
                        ROL     DX
                        SHL     AX
                        ROL     DX
                        SHL     AX
                        ROL     DX
                        RET
```

SUB_PART_22 : BACK_CONVERT_16R :

ANATOMY :

PROCEDURE SIMILAR TO BACK_CONVERT_24R.

REGISTER USAGE :

AX : ADDRESS OF VARIABLE, CRUNCHING NUMBERS.
BH : SIGN OF THE NUMBER.
BL : FIELD OF VARIABLE, SIGN OF EXPONENT AND EXPONENT OF TEN.
DX : TEMPORARY STORAGE.
AL : EXPONENT OF 2.

REFERENCE MEMORY LOCATIONS :

WORDFM.

```
BACK_CONVERT_16R :      MOV     DI, AX
                        MANAGE_SEG_0
                        MOV     CX, PRESENT_DATA_SEG
                        POP     DS
                        TEST    BL, 1CH
                        JNZ     CONT_0
                        MOV     CX, 1FF0H
```

```
CONT_0 :            MOV   SS, CX
                    MOV   AX, (BP+DI)
                    MOV   BH, '-'
                    TEST  AH; 80H
                    JNZ   CONT_1
                    MOV   BH, '+'
CONT_1 :            MOV   CL, AH
                    AND   CL, 7CH
                    SHR   CL                    , 2 TIMES
                    AND   AH, 03H
                    CMP   CL, 10H
                    JAE   CONT_3
                    JMP   NEG_EXP_2
CONT_3 :            SUB   CL, 10H
                    CMP   CL, 10H
                    JB    NEG_EXP_OF_10
                    SUB   CL, 0AH
                    CMP   CL, 05H
                    JNZ   NXT_10
                    JMP   RSLV_P_5
NXT_10 :            CMP   CL, 04H
                    JNZ   NXT_11
                    JMP   RSLV_P_4
NXT_11 :            CMP   CL, 03H
                    JNZ   NXT_12
                    JMP   RSLV_P_3
NXT_12 :            CMP   CL, 02H
                    JNZ   NXT_13
                    JMP   RSLV_P_2
NXT_13 :            CMP   CL, 01H
                    JNZ   NXT_14
                    JMP   RSLV_P_1
NXT_14 :            CMP   CL, 00H
                    JNZ   NXT_15
                    JMP   RSLV_P_0
NXT_15 :            JMP   ERROR_3
NEG_EXP_OF_10 :     CMP   CL, 09H
                    JNZ   NXT_16
                    JMP   RSLV_NE_1
NXT_16 :            CMP   CL, 08H
                    JNZ   NXT_17
                    JMP   RSLV_NE_2
NXT_17 :            CMP   CL, 07H
                    JNZ   NXT_18
                    JMP   RSLV_NE_3
NXT_18 :            CMP   CL, 06H
                    JNZ   NXT_19
                    JMP   RSLV_NE_4
```

```
NXT_19 :        CMP     CL, 05H
                JNZ     NXT_1A
                JMP     RSLV_NE_5
NXT_1A :        CMP     CL, 04H
                JNZ     NXT_1B
                JMP     RSLV_NE_6
NXT_1B :        CMP     CL, 03H
                JNZ     NXT_1C
                JMP     RSLV_NE_7
NXT_1C :        CMP     CL, 02H
                JNZ     NXT_1D
                JMP     RSLV_NE_8
NXT_1D :        CMP     CL, 01H
                JNZ     NXT_1E
                JMP     RSLV_NE_9
NXT_1E :        JMP     RSLV_NE_10
NEG_EXP_2 :     SUB     CL, 10H
                CMP     CL, 15H
                JNZ     NXT_20
                JMP     RSLV_NE_11
NXT_20 :        CMP     CL, 14H
                JNZ     NXT_21
                JMP     RSLV_NE_12
NXT_21 :        CMP     CL, 13H
                JNZ     NXT_22
                JMP     RSLV_NE_13
NXT_22 :        CMP     CL, 12H
                JNZ     NXT_23
                JMP     RSLV_NE_14
NXT_23 :        CMP     CL, 11H
                JNZ     NXT_24
                JMP     RSLV_NE_15
NXT_24 :        CMP     CL, 10H
                JNZ     NXT_25
                JMP     RSLV_NE_16
NXT_25 :        CMP     CL, 0FH
                JNZ     NXT_26
                JMP     RSLV_NE_17
NXT_26 :        CMP     CL, 0EH
                JNZ     NXT_27
                JMP     RSLV_NE_18
NXT_27 :        CMP     CL, 0DH
                JNZ     NXT_28
                JMP     RSLV_NE_19
NXT_28 :        CMP     CL, 0CH
                JNZ     NXT_29
                JMP     RSLV_NE_20
```

```
NXT_29 :        CMP     CL, 0BH
                JNZ     NXT_2A
                JMP     RSLV_NE_21
NXT_2A :        CMP     CL, 0AH
                JNZ     NXT_2B
                JMP     RSLV_NE_22
NXT_2B :        CMP     CL, 09H
                JNZ     NXT_2C
                JMP     RSLV_NE_23
NXT_2C :        CMP     CL, 08H
                JNZ     NXT_2D
                JMP     RSLV_NE_24
NXT_2D :        CMP     CL, 07H
                JNZ     NXT_2E
                JMP     RSLV_NE_25
NXT_2E :        CMP     CL, 06H
                JNZ     NXT_2F
                JMP     RSLV_NE_26
NXT_2F :        JMP     ERROR_3
RSLV_PE_0 :     MOV     BL, 00H
                JMP     LOAD_KBB_1
RSLV_PE_1 :     MOV     BL, 00H
                MOV     CL, 02H
                CALL    MULT_INT_1
                JMP     LOAD_KBB_1
RSLV_PE_2 :     MOV     BL, 00H
                MOV     CL, 04H
                CALL    MULT_INT_1
                JMP     LOAD_KBB_1
RSLV_PE_3 :     MOV     BL, 00H
                MOV     CL, 08H
                CALL    MULT_INT_1
                JMP     LOAD_KBB_1
RSLV_PE_4 :     MOV     BL, 01H
                MOV     CX, 9999H
                CALL    MULT_FRACT_1
                MOV     CL, 01H
                CALL    MULT_INT_1
                ADD     AX, WORDFM
                JMP     LOAD_KBB_1
RSLV_PE_5 :     MOV     BL, 01H
                MOV     CX, 3333H
                CALL    MULT_FRACT_1
                MOV     CL, 03H
                CALL    MULT_INT_1
                ADD     AX, WORDFM
                JMP     LOAD_KBB_1
```

```
RSLV_NE_1 :     MOV   BL, 81H
                MOV   CL, 05H
                CALL  MULT_INT_1
                JMP   LOAD_KBB_1
RSLV_NE_2 :     MOV   BL, 81H
                MOV   CX, 8000H
                CALL  MULT_FRACT_1
                MOV   CL, 02H
                CALL  MULT_INT_1
                ADD   AX, WORDFM
                JMP   LOAD_KBB_1
RSLV_NE_3 :     MOV   BL, 81H
                MOV   CX, 4000H
                CALL  MULT_FRACT_1
                MOV   CL, 01H
                CALL  MULT_INT_1
                ADD   AX, WORDFM
                JMP   LOAD_KBB_1
RSLV_NE_4 :     MOV   BL, 82H
                MOV   CX, 8000H
                CALL  MULT_FRACT_1
                MOV   CL, 06H
                CALL  MULT_INT_1
                ADD   AX, WORDFM
                JMP   LOAD_KBB_1
RSLV_NE_5 :     MOV   BL, 82H
                MOV   CX, 2000H
                CALL  MULT_FRACT_1
                MOV   CL, 03H
                CALL  MULT_INT_1
                ADD   AX, WORDFM
                JMP   LOAD_KBB_1
RSLV_NE_6 :     MOV   BL, 82H
                MOV   CX, 9000H
                CALL  MULT_FRACT_1
                MOV   CL, 01H
                CALL  MULT_INT_1
                ADD   AX, WORDFM
                JMP   LOAD_KBB_1
RSLV_NE_7 :     MOV   BL, 83H
                MOV   CX, D000H
                CALL  MULT_FRACT_1
                MOV   CL, 07H
                CALL  MULT_INT_1
                ADD   AX, WORDFM
                JMP   LOAD_KBB_1
RSLV_NE_8 :     MOV   BL, 83H
                MOV   CX, E70AH
```

```
                        CALL    MULT_FRACT_1
                        MOV     CL, 03H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_9 :             MOV     BL, 83H
                        MOV     CX, F3FEH
                        CALL    MULT_FRACT_1
                        MOV     CL, 01H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_10 :            MOV     BL, 83H
                        MOV     CX, F9FBH
                        CALL    MULT_FRACT_1
                        JMP     LOAD_KBB_1
RSLV_NE_11 :            MOV     BL, 84H
                        MOV     CX, E1FFH
                        CALL    MULT_FRACT_1
                        MOV     CL, 04H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_12 :            MOV     BL, 84H
                        MOV     CX, 70FFH
                        CALL    MULT_FRACT_1
                        MOV     CL, 02H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_13 :            MOV     BL, 84H
                        MOV     CX, B87FH
                        CALL    MULT_FRACT_1
                        MOV     CL, 01H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_14 :            MOV     BL, 85H
                        MOV     CX, 1A7EH
                        CALL    MULT_FRACT_1
                        MOV     CL, 06H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_15 :            MOV     BL, 85H
                        MOV     CX, 0D3FH
                        CALL    MULT_FRACT_1
                        MOV     CL, 03H
```

```
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_16 :            MOV     BL, 85H
                        MOV     CX, 869AH
                        CALL    MULT_FRACT_1
                        MOV     CL, 01H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_17 :            MOV     BL, 86H
                        MOV     CX, A119H
                        CALL    MULT_FRACT_1
                        MOV     CL, 07H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_18 :            MOV     BL, 86H
                        MOV     CX, D089H
                        CALL    MULT_FRACT_1
                        MOV     CL, 03H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_19 :            MOV     BL, 86H
                        MOV     CX, E844H
                        CALL    MULT_FRACT_1
                        MOV     CL, 01H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_20 :            MOV     BL, 86H
                        MOV     CX, F425H
                        CALL    MULT_FRACT_1
                        JMP     LOAD_KBB_1
RSLV_NE_21 :            MOV     BL, 87H
                        MOV     CX, C4D0H
                        CALL    MULT_FRACT_1
                        MOV     CL, 04H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_22 :            MOV     BL, 87H
                        MOV     CX, 8254H
                        CALL    MULT_FRACT_1
                        MOV     CL, 02H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
```

```
                        JMP     LOAD_KBB_1
RSLV_NE_23 :            MOV     BL, 87H
                        MOV     CX, 3126H
                        CALL    MULT_FRACT_1
                        MOV     CL, 01H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_24 :            MOV     BL, 88H
                        MOV     CX, F5C2H
                        CALL    MULT_FRACT_1
                        MOV     CL, 05H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_25 :            MOV     BL, 88H
                        MOV     CX, FAE1H
                        CALL    MULT_FRACT_1
                        MOV     CL, 02H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
                        JMP     LOAD_KBB_1
RSLV_NE_26 :            MOV     BL, 88H
                        MOV     CX, 7D77H
                        CALL    MULT_FRACT_1
                        MOV     CL, 01H
                        CALL    MULT_INT_1
                        ADD     AX, WORDFM
LOAD_KBB_1 :            MOV     BP, 0000H
                        MOV     DI, BASE_ADD_KBB
                        MOV     (BP+DI), BH
                        MOV     CX, 03E8H
                        MOV     DX, 0000H
                        DIV     DX, CX
                        INR     DI
                        MOV     (BP+DI), AL
                        MOV     AX, DX
                        MOV     CL, 64H
                        DIV     AX, CL
                        INR     DI
                        MOV     (BP+DI), AL
                        MOV     AL, AH
                        MOV     AH, 00H
                        MOV     CL, 0AH
                        DIV     AX, CL
                        INR     DI
                        MOV     (BP+DI), AL
                        INR     DI
```

```
                        MOV     (BP+DI), AH
                        MOV     BH, '+'
                        TEST    BL, 80H
                        JZ      NXT_1
                        MOV     BH, '-'
NXT_1 :                 INR     DI
                        MOV     (BP+DI), BH
                        MOV     AL, BL
                        AND     AL, 70H
                        SHR     AL                      , 4 TIMES
                        INR     DI
                        AND     BL, 0FH
                        INR     DI
                        MOV     (BP+DI), AL
                        RET
MULT_FRACT_1 :          MUL     AX, CX
                        MOV     WORDFM, DX
                        DCR     DI
                        RET
MULT_INT_1 :            INR     DI
                        MOV     AX, (BP+DI)
                        AND     AH, 03H
                        CMP     AL, 01H
                        JA      MULT_12
                        RET
MULT_12 :               CMP     CL, 02H
                        JA      MULT_13
                        SHL     AX
                        RET
MULT_13 :               CMP     CL, 03H
                        JA      MULT_14
                        MOV     DX, AX
                        SHL     AX
                        ADD     AX, DX
                        RET
MULT_14 :               CMP     CL, 04H
                        JA      MULT_15
                        SHL     AX
                        SHL     AX
                        RET
MULT_15 :               CMP     CL, 05H
                        JA      MULT_16
                        MOV     DX, AX
                        SHL     AX
                        SHL     AX
                        ADD     AX, DX
                        RET
```

```
MULT_16 :       CMP     CL, 06H
                JA      MULT_17
                MOV     DX, AX
                SHL     AX
                ADD     AX, DX
                SHL     AX
                RET
MULT_17 :       CMP     CL, 07H
                JA      MULT_18
                MOV     DX, AX
                SHL     AX
                ADD     AX, DX
                SHL     AX
                ADD     AX, DX
                RET
MULT_18 :       CMP     CL, 08H
                JZ      NXT_8
                JMP     ERROR_3
NXT_8 :         SHL     AX                      , 3 TIMES
                RET
```

SUB_PART_2_3 : BACK_CONVERT_16I
&
SUB_PART_2_4 : BACK_CONVERT_8I

ANATOMY :

THE ROUTINE SPLITS THE NUMBER INTO THE CORRESPONDING NIBBLES AND LOADS THE NUMBER WITH A MARK OF ITS SIGN.

REGISTER USAGE :

AX : ADDRESS OF THE VARIABLE, NUMBER TO BE CONVERTED.
BX, AH : TEMPORARY STORAGE.
BH : SIGN OF THE NUMBER.
BL : IDENTIFIER OF VARIABLE.

```
BACK_CONVERT_16I :    MOV   DI, AX
                      MANAGE_SEG_0
                      MOV   CX, PRESENT_DATA_SEG
                      POP   DS
                      TEST  BL, 1CH
                      JNZ   CONT_0
                      MOV   CX, 1FF0H
CONT_0 :              MOV   SS, CX
                      MOV   AX, (BP+DI)
                      MOV   BH, '-'
                      TEST  AH, 80H
                      JNZ   CONT_1
                      INR   BH
                      MOV   DI, BASE_ADD_KBB
                      MOV   BP, 0000H
CONT_1 :              AND   AH, 7FH
                      MOV   BL, AH
                      AND   BL, F0H
                      SHR   BL                    , 4 TIMES
                      MOV   (BP+DI), BH
                      INR   DI
                      MOV   (BP+DI), BL
                      AND   AH, 0FH
                      INR   DI
                      MOV   (BP+DI), AH
                      MOV   AH, AL
                      AND   AH, F0H
                      SHR   AH                    , 4 TIMES
                      INR   DI
                      MOV   (BP+DI), AH
                      AND   AL, 0FH
                      INR   DI
```

```
                        MOV     (BP+DI), AL
                        RET


BACK_CONVERT_8I :       MOV     DI, AX
                        MANAGE_SEG_0
                        MOV     CX, PRESENT_DATA_SEG
                        POP     DS
                        TEST    BL, 1CH
                        JNZ     CONT_0
                        MOV     CX, 1FF0H
CONT_0 :                MOV     SS, CX
                        MOV     AL, (BP+DI)
                        MOV     BH, '-'
                        TEST    AH, 80H
                        JNZ     CONT_1
                        INR     BH
CONT_1 :                MOV     BP, 0000H
                        MOV     DI, BASE_ADD_KBB
                        MOV     BL, AL
                        AND     BL, 70H
                        SHR     BL                      , 4 TIMES
                        MOV     (BP+DI), BL
                        AND     AL, 0FH
                        INR     DI
                        MOV     (BP+DI), AL
                        RET
```