C H A P T E R V

DESCRIPTION OF SYSTEM IN RUN MODE

System support at the RUN time could be logically subdivided amongst three stages.

.. (1) Sorting the interpreter code (IC) blocks corresponding to a program index and then arranging IC in the ascending order of line number (LNO). At this stage various tables for the references of itself or at the time of program execution are prepared.

(2) As described in previous section a program index could be slipted up within up to three IC units (sec. 4.6, 5), starting and ending pointers of a unit with status count 1 are determined and recorded. Program execution is initiated for this unit keeping track of ending pointer of set recorded.

(3) Succesively the IC unit of the remaining status count are executed in a way as similar to the stage 2. These stages are elaborated in the present discussion.

5.1. Arranging Program In Ascending Order.

As described in LRN mode each time a program area is opened, a appropriate status byte is recorded in PIT. Structure of PIT is shown in fig. 5.1. The count of status, maximum three, specifies number of IC units (sec. 4.6) forms the complete program. From PIT monitor confirms that the program is ended explicitly (Table 5.1) and calculates segment and offsets of IC unit

corresponding status count 10000001. Further it determines the ending segment and offset of the unit. Now monitor opens up three reference areas.

1) Line Number Table (fig. 5.2), indicating line number of first program statement in a block of 256 bytes.

2) Offset Table, indicating the segment and offset of a sentence not found in the ascending order (Fig. 5.3.).

3) Buffer Area where the off ascending order statements will be stored. A memory location is initialized to point to starting address of buffer and another location labelled 'ATTACH_BYTE' is resetted. These locations are updated while sequencing the program. Now monitor starts confirming ascending order of the sentences and makes entries in LNO table after a block of 256 bytes is checked. If a sentence is found off order then using the LNO table appropriate block where the sentence is to be inserted is determined and pointed. If it is a over written statement, then the previous statement is deleted and if space permits new sentence will be added in that position. If space does not permit to insert the sentence or if the sentence itself is an insertion then the attach byte field of next sentence (Table 4.11) is loaded with the appropriate attach byte and pointing to the next buffer address the sentence is loaded into the buffer area.

This procedure puts a few constraints on editing a program viz.

1) Only one sentence could be inserted between any two.

2) Total number of inserted sentence should be less than 254 (FEH) is the limiting byte as FF is used as a delimeter indicating extension of the sentence in the next 8 byte block (Table 4.11).

3) Total length of inserted codes should be less than 2 K bytes.

These constraints are well observable for a professional programmer.

Additionally trade-off of an alternative structure are also estimated where in the interpreter code structure (Table 4.11) instead of a attach byte, a word informing segment and offset pointer fields of interposing sentences are loaded, which adds one more byte to the IC structure. As 8 byte blocks are used to store IC of a sentence, estimates of IC code lengths for 6 sample control programs using both the alternatives work carried out. It has been observed that the structure with attach byte is space-economical. Therefore, though alternative structure does not impose any constraint on number of sentences to be inserted, paying a biased vote to space economy. Further analysis of a situation where alternative structure is used are reserved to be carried out at the stage of detailed trade-off comparisons.

The process of arranging a program in ascending order continues till last address of set for the last status count of the program selected.

5.2. Reference Table For Pointing Datum.

Generation of data index table (DIT) has been described (sec. 4.6), making use of this table monitor executes block alter routine (sec. 4.3) and calculate segment and base address of each type of data elements described in Table 4.4. These are stored in a table referred to as data reference table (DRT). Additionally this table loads title (key code) of 0^{th} ordered variable for each data type. This makes runtime routine convenient to point to variables, where the task left is merely a program of deciding what type of variable is the present one. Further the monitor repoints to the PIT, (Table 5.1) and calculates standing and ending offset of status count 1, remembers the status count of execution in program, and starts executing the sentences. While executing a program the pointer confirms attach byte to be zero and resolves group, subgroup of the sentence and execute the sentence. The program to execute various sentences is nested call structure, similar as in the LRN mode, flow chart 4.1. If attach byte is non-zero then the appropriate sentence in buffer area is pointed and executed, after saving the present pointer. After execution of each sentence end point of the unit is tested and if not the end point next sentence is executed. After reaching the end point the monitor executes the remaining IC units. The program flow resumes Level_0 (flow chart 4.1) after execution of end sentence in the program.

5.3. Remark On Execution Of Interpreter Code.

1) INPUT, OUTPUT sentences :

A group of sentence to input a byte to the system has wider range of subgroups. The IC field directly or indirectly indicates source and destination of the subject.

IN AQ [QR] are syntax is meant to enter a numeric constant (NC) from KB. Here monitor unmasks the KBIRQ and awaits for entries from KB. Upon numeric entries monitor executes KBIRQ service routine and upon detecting a 'ENTER' key the flow returns to monitor. Now key entries in the key board buffer are converted into the internal forms of representation, sematics is confirmed and NC is loaded into the address of corresponding variable, indicated within IC block as a destination pointer. The remaining instructions of IN group are a bit straight forward. Here source address is loaded into the register DX. If the subgroup points source to be ADC then a OUT instruction is executed to select the analog input channel and system halts for ADC IRQ (sec. 3.2). Otherwise a IN AL, DX instruction is executed to receive the byte in register AL. If subgroup demands transfer of the acquised byte to memory then it is executed after evaluating address of destination pointer, using the DRT (sec. 5.2).

INW, OUT, OUW instructions are executed on the similar lines.

5.4. Control Statements.

1) GTO LNO :- After resolving the jump to be forward or backward from subgroup index of the block of the source sentence is evaluated. Using the index, an appropriate word in the LNO table is pointed. The table is scanned for LNOs, forward or backwards, to determine index of the destination block, where the program execution has to continue. This program structure is a bit complicated and reference to PIT are made to recalculated the ending segment and offset values of destination IC unit, if the jump makes program flow to resume within another IC unit than that of the source. Further destination block index is used to determine the pointer of the destination sentence, the LNO of interest.

2) FOR - NXT structure :- As the allowed nesting is confirmed at the LRN time itself at the RUN time syntax checking overhead does not exists. If FOR Q = n to m is encountered at the RUN time then the routine calculates address of the parameter Q to be valid. Further the parameter is initialized at n and a count providing the required number of iterations is calculated. Program pointer is moved to the next sentence and program pointer, data pointer and the counts are saved (PUSHed) on the stack. Upon encounter of NXT action group count is reloaded (POPed) and whether the program flow is to exit out of the structure or not is determined. If flow is to remain within the structure then data pointer and program pointer are poped,

counter is decremented and iteration parameters Q is incremented. Further the pointer and the count is resaved on the stack. If program flow is to exit out of the structure then stack pointer is readjusted accordingly.

3) IF condition THEN LNO :- This sentence requires magnitude to be compared to be either inputted from a specified port or to be loaded from memory using the variable name in the IC field. The procedure of inputting a byte is similar and is described previously. Magnitude on RHS is loaded inside the processor registers while the magnitude on LHS is saved on stack. Comparison routines for each type of data items are called and the result of comparison is transmitted to the main routine through registers. IF condition is fulfilled then program jumps to GTO LNO routine additionally, otherwise returns.

4) GOSUB LNO - RET :- While executing this structure IC field pointers for the next line are saved on the stack and jump to GTO LNO routine is executed. Prior to the jump register BH is loaded with the subgroup of forward jump upon RET the IC field pointer are reloaded from stack.

As an additional feature, if alternate structure of arranging program in the ascending order had implemented, then GOSUB structure operating within the IC field of another program area would have been feasible (sec. 5.1).

5.5. Execution of Arithmatic Equations.

As indicated in sec. 4.6 (6) the IC field of LET statement contains entries of variables, numeric constant and operators in the post fixed rotational form. The hierarchy scheme used at present form conversion of infix to postfix forms of expression associates explicit values to /, *, -, + operators, through /, * and -, + grouping is allowable. Within further developments of the languages where '(', ')' will be allowed in the expression, hierarchy at the level of group of operators is proposed to be implemented.

The equation (expression conversion operator (Table 4.8) = treated as an operator itself) is executed using the stack. A virtue of 8086 offering a way to access stack as normal data/memory area (using SP and BP), executing arithmatic operations on data elements in the stack becomes most convenient. Separate routines executing arithmatic operations on each type of data item, viz. 24 Real, 16 Real, 16 Integer, 8 Integer are developed and tested.

The subprogram executing the LET group determines subgroup of operation. If the subgroup indicates target program call, then program flow jumps to execute a inter segment indirect call instruction using the parameter of the IC field and through nested returns the flow extends to execute the next sentence in the sequence (flow chart 5.1).

If the subgroup indicates arithmatic operations, then subgroup specifying type of operands on RHS quadruplicates the routine flow. If the operand specification of RHS is 24 REal or 16 Integer type then micro converting 16 Real to 24 Real or 8 Integer to 16 Integer are additionally incorporated in the flow, as these types are self accomodable (sec.4.4). The IC is scanned and offset and segment of the variable on LHS are loaded on to the stack. While scanning IC field for entries on RHS of expression, if a variable reference or a NC is encountered then the corresponding values are loaded on to the stack. If an operator is encountered then proper subprogram executing the operation and returning result to stack, after deletion of operands on the stack, is called. The process continues till a = operator is encounterd. This operator loads the operand on the stack using the segment and offset of variable on LHS stored on the stack itself.

5.6. Incorporating The Desired Delays.

It has been specified while discussing hardware configuration (sec. 3.2). that the counter 1 and 2 of 8254_1 are loaded with appropriate count and mode words to generate a square wave of 1 KHz out of OUT 2, while OUT 2 is fed back as CLK 0. Now, monitor supports three subgroups for delay where BCD count down with CLK rates of 1 KHz, 100 Hz or 10 Hz is to be implemented. If subgroup specifies clock rate of 1 KHz then counter 0 of 82541 is simply loaded with count in IC field in

mode 0 and BCD counting. For clock rates of 100 Hz and 10 Hz counter 2 and counter 0 are loaded with appropriate count and mode words. INT output of counter 0 terminates the delay.

5.7. Null Sentences And Exit From RUN.

A null sentence indicates that the monitor should point to next sentence, implying no action sequence to be executed. The corresponding action groups are 00h and FFH. The action group 00 indicates an option

1) the action group could be 00 if the selected program index area is 00h. Further this entry occurs only in the beginning where the program area is opened.

2) Action group 00 is encounter anywhere but not in a IC field in the beginning of a 256 byte block.

If action group is 00 and of type (1) then correctness of program index and data area are confirmed. If action group of type (2) is encountered then the IC field counter is incremented to point to the next 256 byte boundry, Level_2 (flow chart 5.1).

If FFH appears as an action group then the flow continues to point to the next sentence the interpreter code structure for these sentences is given in fig. 5.3. If action group indicating physical end of program is encountered then RAM BUFFER POINTER and SI TAB POINTER are reinitialized. Further using these two tables the sentences from BUFFER area are reloaded into the places from where these sentences were brought in BUFFER area.

Finally, all the attach bytes within the IC field of the current program area are resetted and program flow jumps to Level_0.

As a concluding remark the structure developed for executing interpreter code is undergoing or getting subjected a thorough evaluation of speed and code optimization, redesign etc. Further, validity of the structure in the environments of implementation of the kit is to be tested.

Table 5.1 : PIT.

GENERAL INFORMATION		
BASE_ADD_PIT +	00	Total areas (Blocks of 256 used)
	01	used by all the program.
	02	Count indicating total number of IC units.
	03	Program indices sequentially
	¦	corresponding to IC units.
	¦	
	¦	
SPECIFIC INFORMATION		
INDEX 0		
	00	Total area used
	01	
	02	Data select index
	03	Status count
	04	Starting block
	05	for status 1
	06	Ending block
	07	for status 1
	08	Starting block
	09	for status 2
	0A	Ending block
	0B	for status 2
	0C	Starting block
	0D	for status 3
	0E	Ending block
	0F	for status 3

Table 5.2 : LNO Table.

BASE_ADD_LNO +	00	FF	LNO	First program block
	01	FF	LNO	
	02			Second program block
	03			
	⋮			
	⋮			

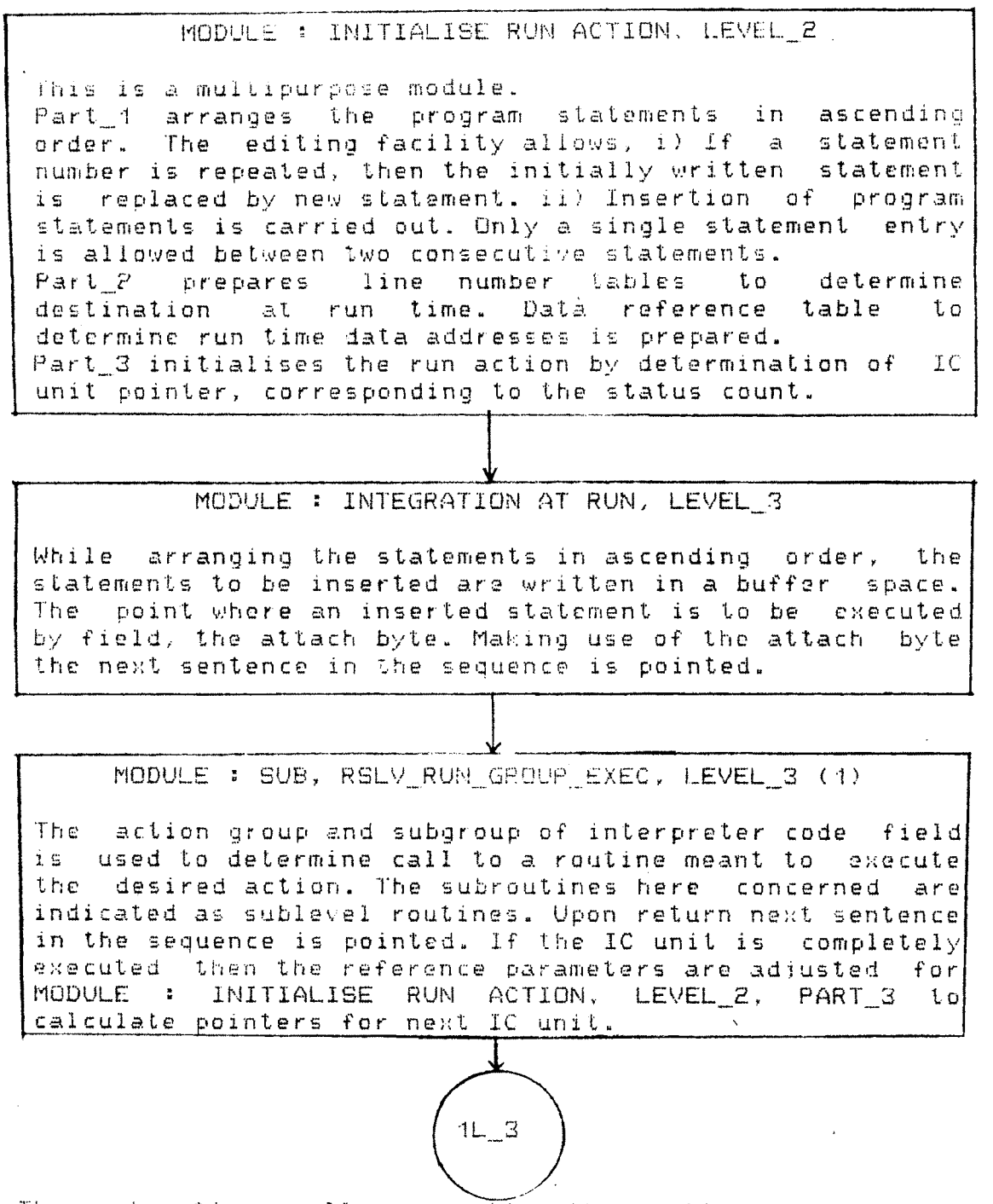
FF indicates block does not corresponds to program index selected.

Table 5.3 : OFFSET Table.

MSB of the word representing OFFSET specifies segment, whether 0^{th} or 1^{st} remaining 15 bit represent OFFSET.

BASE_ADD_OFFSET +	00	$OFFSET_L$	Corresponding to first program line transferred to RAM buffer
	01	$OFFSET_H$	

F L O W C H A R T 5.1



;The subroutine calls, executing the actions, accommodate ;execution of the IC fields loaded by the interpreter for its ;reference. The action groups resolved are listed below :
; 1) END; 2) OQ; 3) FF; 4) RET; 5) LET; 6) INB; 7) DLY
; 8) OUW; 9) INR; 10) INW; 11) OUB; 12) DSP;13) DCR; 14) IF
;15) GSB;16) FOR; 17) GTO; 18) NXT.