11

The 8086 Microprocessor

1. Draw the pin diagram of 8086.

Ans. There would be two pin diagrams—one for MIN mode and the other for MAX mode of 8086, shown in Figs. 11.1 and 11.2 respectively. The pins that differ with each other in the two modes are from pin-24 to pin-31 (total 8 pins).

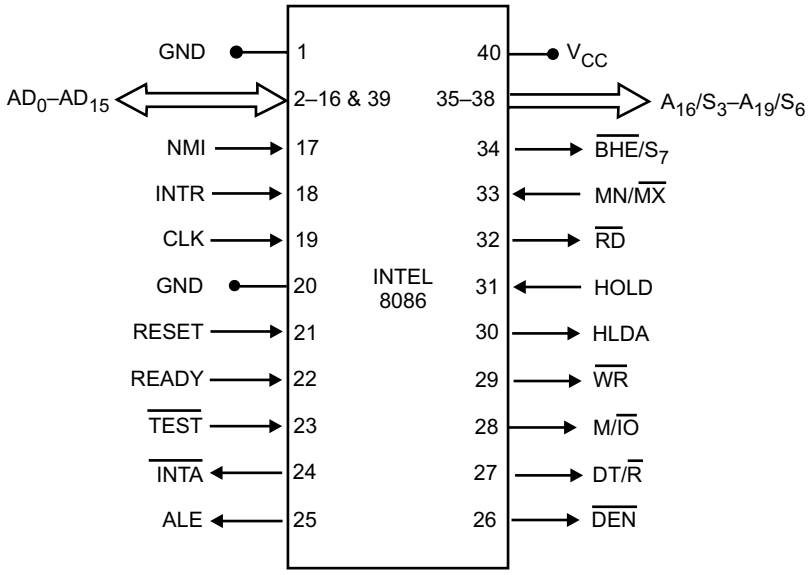


Fig. 11.1: Signals of intel 8086 for minimum mode of operation

2. What is the technology used in 8086 μP?

Ans. It is manufactured using high performance metal-oxide semiconductor (HMOS) technology. It has approximately 29,000 transistors and housed in a 40-pin DIP package.

3. Mention and explain the modes in which 8086 can operate.

Ans. 8086 μP can operate in two modes—MIN mode and MAX mode.

When MN/$\overline{MX}$ pin is high, it operates in MIN mode and when low, 8086 operates in MAX mode.

For a small system in which only one 8086 microprocessor is employed as a CPU, the system operates in MIN mode (Uniprocessor). While if more than one 8086 operate in a system then it is said to operate in MAX mode (Multiprocessor).

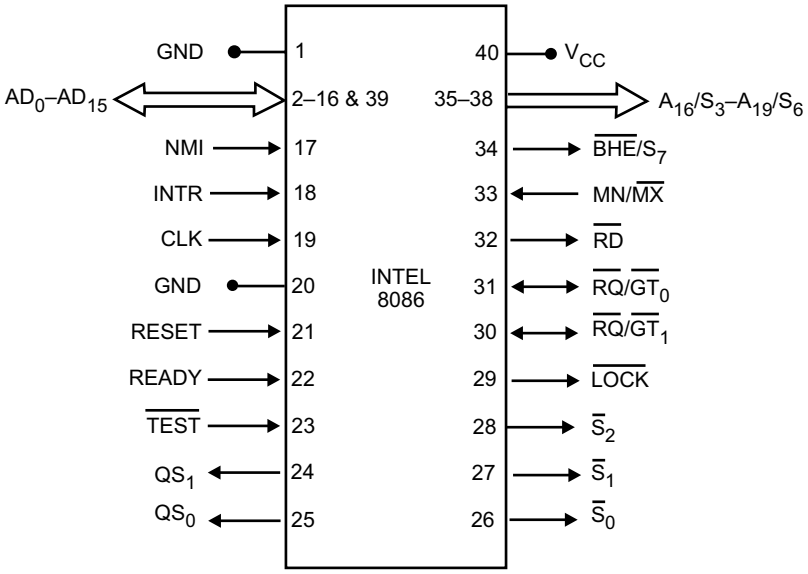


Fig. 11.2: Signals of intel 8086 for maximum mode of operation

The bus controller IC (8288) generates the control signals in case of MAX mode, while in MIN mode CPU issues the control signals required by memory and I/O devices.

4. Distinguish between the lower sixteen address lines from the upper four.

Ans. Both the lower sixteen address lines $(AD_0 \quad AD_{15})$ and the upper four address lines $(A_{16}/S_3 \quad A_{19}/S_6)$ are multiplexed.

During T_1, the lower sixteen lines carry address $(A_0 \quad A_{15})$, while during T_2, T_3 and T_4 they carry data.

Similarly during T_1, the upper four lines carry address (A_{16}– A_{19}), while during T_2, T_3 and T_4, they carry status signals.

5. In how many modes the minimum-mode signal can be divided?

Ans. In the MIN mode, the signals can be divided into the following basic groups: address/data bus, status, control, interrupt and DMA.

6. Tabulate the common signals, Minimum mode signals and Maximum mode signals. Also mention their functions and types.

Ans. Table 11.1 shows the common signals, Minimum mode signals and the Maximum mode signals, along with the functions of each and their types.

Table 11.1 : (a) Signals common to both minimum and maximum mode, (b) Unique minimum-mode signals, (c) Unique maximum-mode signals for 8086.

Common signals		
Name	**Function**	**Type**
AD15-AD0	Address/data bus	Bidirectional, 3–state
A19/S6-A16/S3	Address/status	Output, 3-state
MN/$\overline{MX}$	Minimum/maximum Mode control	Input
$\overline{RD}$	Read control	Output, 3-state
$\overline{TEST}$	Wait on test control	Input
READY	Wait state control	Input
RESET	System reset	Input
NMI	Nonmasakable Interrupt request	Input
INTR	Interrupt request	Input
CLK	System clock	Input
V_{cc}	+5V	Input
GND	Ground	

(a)

Minimum mode signals (MN/$\overline{MX}$ =V_{cc})		
Name	**Function**	**Type**
HOLD	Hold request	Input
HLDA	Hold acknowledge	Output
$\overline{WR}$	Write control	Output, 3-state
M/$\overline{IO}$	IO/memory control	Output, 3-state
DT/$\overline{R}$	Data transmit/receive	Output, 3-state
$\overline{DEN}$	Data enable	Output, 3-state
ALE	Address latch enable	Output
$\overline{INTA}$	Interrupt acknowledge	Output

(b)

Maximum mode signals (MN/$\overline{MX}$ =GND)		
Name	**Function**	**Type**
$\overline{RQ}$/$\overline{GT1,0}$	Request/grant bus access control	Bidirectional
$\overline{LOCK}$	Bus priority lock control	Output, 3-state
$\overline{S2}$ – $\overline{S0}$	Bus cycle status	Output, 3-state
QS1, QS0	Instruction queue status	Output

(c)

7. Mention the different varieties of 8086 and their corresponding speeds.

Ans. The following shows the different varieties of 8086 available and their corresponding speeds.

Types	Speeds
8086	5 MHz
8086–1	10 MHz
8086–2	8 MHz

8. Mention (a) the address capability of 8086 and (b) how many I/O lines can be accessed by 8086.

Ans. 8086 addresses via its A_0–A_{19} address lines. Hence it can address 2^{20} = 1MB memory.

Address lines A_0 to A_{15} are used for accessing I/O's. Thus, 8086 can access 2^{16} = 64 KB of I/O's.

9. What is meant by microarchitecture of 8086?

Ans. The individual building blocks of 8086 that, as a whole, implement the software and hardware architecture of 8086. Because of incorporation of additional features being necessitated by higher performance, the microarchitecture of 8086 or for that matter any microprocessor family, evolves over time.

10. Draw and discuss the architecture of 8086. Mention the jobs performed by BIU and EU.

Ans. The architecture of 8086 is shown below in Fig. 11.3. It has got two separate functional units—Bus Interface Unit (BIU) and Execution Unit (EU).

8086 architecture employs parallel processing—i.e., both the units (BIU and EU) work at the same time. This is *Unlike* 8085 in which *Sequential* fetch and execute operations take place. Thus in case of 8086, efficient use of system bus takes place and higher performance (because of reduced instruction time) is ensured.

- BIU has segment registers, instruction pointer, address generation and bus control logic block, instruction queue while the EU has general purpose registers, ALU, control unit, instruction register, flag (or status) register.

The main jobs performed by BIU are:

- BIU is the 8086's interface to the outside world, i.e., all *External* bus operations are done by BIU.
- It does the job of instruction fetching, reading/writing of data/operands for memory and also the inputting/outputting of data for peripheral devices.
- It does the job of filling the instruction queue.
- Does the job of address generation.

The main jobs performed by the execution unit are:

- Decoding/execution of instructions.
- It accepts instructions from the output end of instruction queue (residing in BIU) and data from the general purpose registers or memory.
- It generates operand addresses when necessary, hands them over to BIU requesting it (BIU) to perform read or write cycle to memory or I/O devices.
- EU tests the status of flags in the control register and updates them when executing instructions.
- EU waits for instructions from the instruction queue, when it is empty.
- EU has no connection to the system buses.

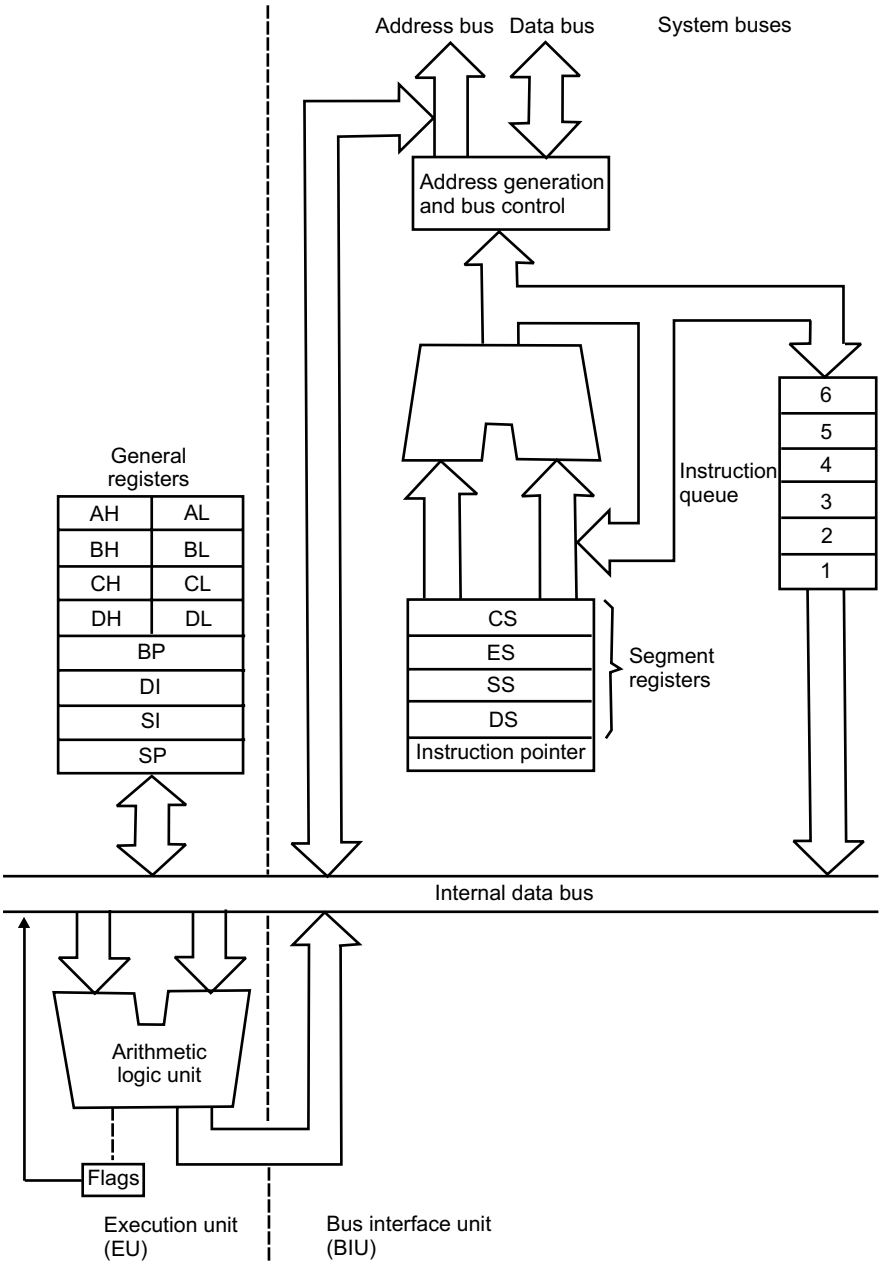


Fig. 11.3: CPU model for the 8086 microprocessor. A separate execution unit (EU) and bus interface unit (BIU) are provided.

11. Explain the operations of instructions queue residing in BIU.

Ans. The instruction queue is 6-bytes in length, operates on FIFO basis, and receives the instruction codes from memory. BIU fetches the instructions meant for the queue ahead of time from memory. In case of JUMP and CALL instructions, the queue is dumped and newly formed from the new address.

Because of the instruction queue, there is an overlap between the instruction execution and instruction fetching. This feature of fetching the next instruction when the current instruction is being executed, is called *Pipelining*.

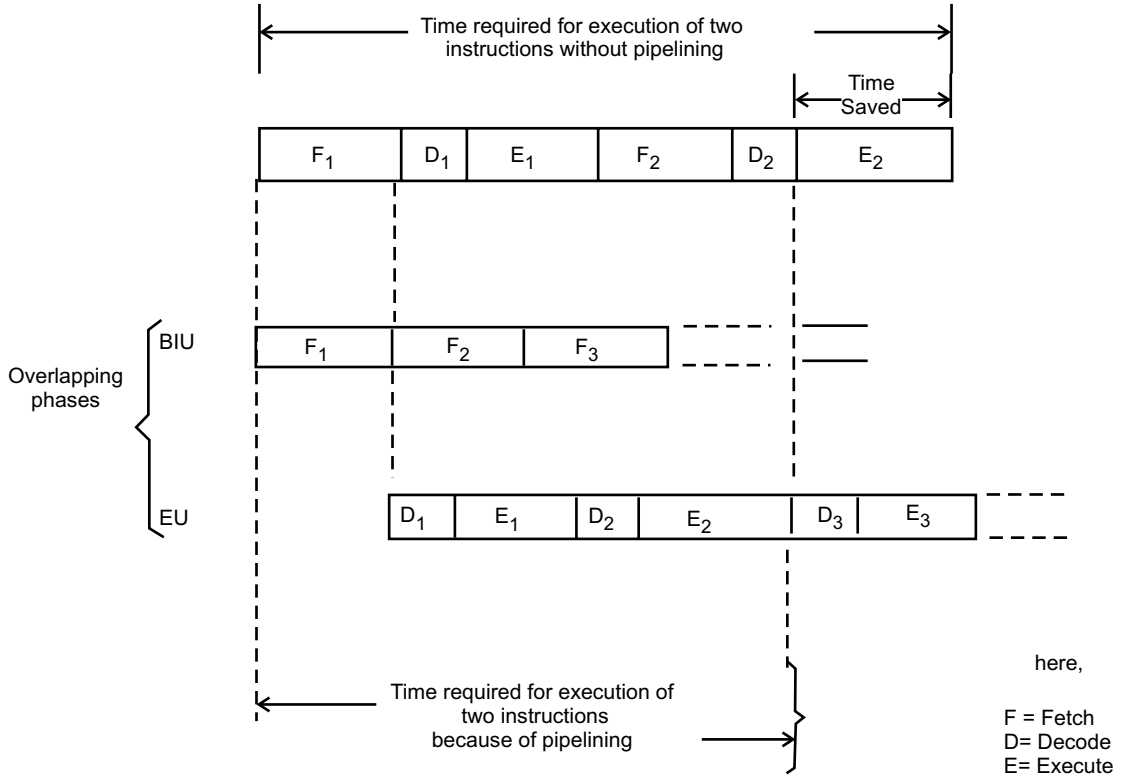


Fig.11.4: Pipelining procedure saves time

Fig. 11.4, which is self-explanatory, shows that there is definitely a time saved in case of overlapping phases (as in the case of 8086) compared to sequential phases (as in the case of 8085).

Initially, the queue is empty and CS : IP is loaded with the required address (from which the execution is to be started). Microprocessor 8086 starts operation by fetching 1 (or 2) byte(s) of instruction code(s) if CS : IP address is odd (even).

The 1st byte is always an opcode, which when decoded, one byte in the queue becomes empty and the queue is updated. The filling in operation of the queue is not started until two bytes of the instruction queue is empty. The instruction execution cycle is *never* broken for fetch operation.

After decoding of the 1st byte, the decoder circuit gets to know whether the instruction is of single or double opcode byte.

For a single opcode byte, the next bytes are treated as data bytes depending upon the decoded instruction length, otherwise the next byte is treated as the second byte of the instruction opcode.

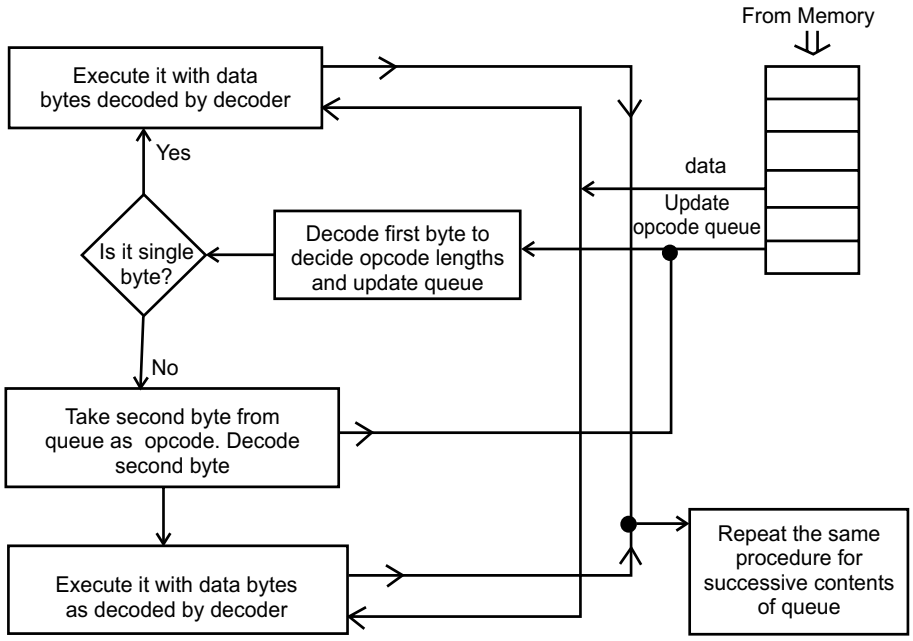


Fig.11.5 : The queue operation

For a 2-byte instruction code, the decoding process takes place taking both the bytes into consideration which then decides on the decoded instruction length and the number of subsequent bytes which will be treated as instruction data. Updation of the queue takes place once a byte is read from the queue.

The queue operation is shown in Fig. 11.5 in block schematic form.

12. Mention the conditions for which EU enters into WAIT mode.

Ans. There are three conditions that cause the EU to enter into WAIT state. These are:

- When an instruction requires the access to a memory location not in the queue.
- When a JUMP instruction is executed. In this case the current queue contents are aborted and the EU waits until the instructions at the jump address is fetched from memory.
- During execution of instruction which are very slow to execute. The instruction AAM (ASCII adjust for multiplication) requires 83 clock cycles for execution. For such a case, the BIU is made to wait till EU pulls one or two bytes from the queue before resuming the fetch cycle.

13. Mention the kind of operations possible with 8086.

Ans. It can perform bit, byte, word and block operations. Also multiplication and division operations can be performed by 8086.

14. Mention the total number of registers of 8086 and show the manner in which they are grouped.

Ans. There are in all fourteen numbers of 16-bit registers. The different groups are made as hereunder:

- Data group, pointers and index group, status and control flag group and segment group.
- The data group consists of AX (accumulator), BX (base), CX (count) and DX (data).
- Pointer and Index group consist of SP (Stack pointer), BP (Base pointer), SI (Source Index), DI (Destination index) and IP (Instruction pointer).
- Segment group consists of ES (Extra Segment), CS (Code Segment), DS (Data Segment) and SS (Stack Segment).
- Control flag group consists of a single 16-bit flag register.

Fig. 11.6 shows the registers placed in the different groups to form a programming model.

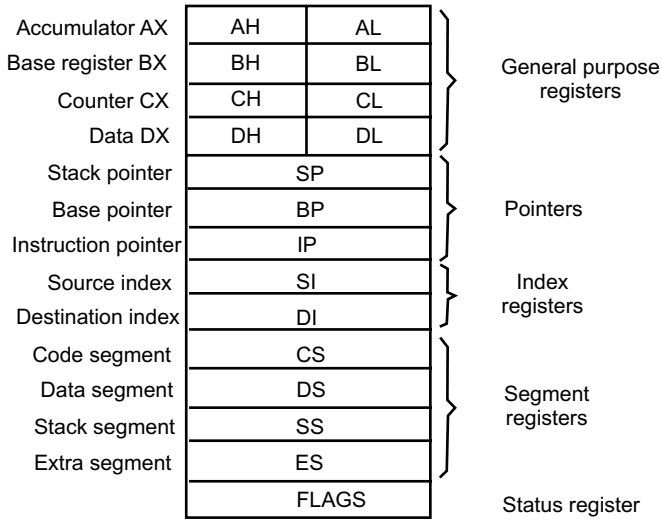
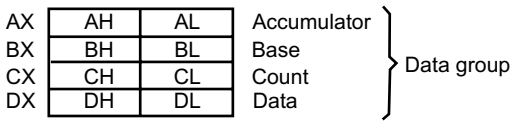


Fig.11.6: Schematic diagram of intel 8086 registers

15. Describe, in detail, the general purpose of data registers.

Ans. Fig. 11.7 shows the four data registers along with their dedicated functions also.



Register	Operations
AX	Word multiply, Word divide, Word I/O
AL	Byte multiply, byte divide, byte I/O, translate, decimal arithmetic
AH	Byte multiply, byte divide
BX	Translate
CX	String operations, Loops
CL	Variable shift and rotate
DX	Word multiply, Word divide, indirect I/O

Fig. 11.7: Data group registers and their functions

All the four registers can be used bytewise or wordwise. The alphabets X, H and L respectively refer to word, higher byte or lower byte respectively of any register.

All the four registers can be used as the source or destination of an operand during an arithmetic operation such as ADD or logical operation such as AND, although particular registers are earmarked for specific operations. Register C is used as a count register in string operations and as such is called a 'count' register. Register C is also used for multibit shift or rotate instructions.

Register D is used to hold the address of I/O port while register A is used for all I/O operations that require data to be inputted or outputted.

16. Describe the status register of 8086.

Ans. It is a 16-bit register, also called flag register or Program Status Word (PSW). Seven bits remain unused while the rest nine are used to indicate the conditions of flags. The status flags of the register are shown below in Fig. 11.8.

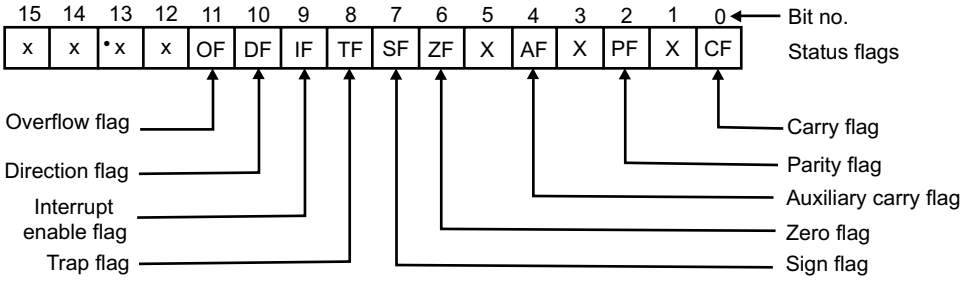


Fig.11.8: Status flags of intel 8086

Out of nine flags, six are condition flags and three are control flags. The control flags are TF (Trap), IF (Interrupt) and DF (Direction) flags, which can be set/reset by the programmer, while the condition flags [OF (Overflow), SF (Sign), ZF (Zero), AF (Auxiliary Carry), PF (Parity) and CF (Carry)] are set/reset depending on the results of some arithmetic or logical operations during program execution.

CF is set if there is a carry out of the MSB position resulting from an addition operation or if a borrow is needed out of the MSB position during subtraction.

PF is set if the lower 8-bits of the result of an operation contains an even number of 1's. AF is set if there is a carry out of bit 3 resulting from an addition operation or a borrow required from bit 4 into bit 3 during subtraction operation.

ZF is set if the result of an arithmetic or logical operation is zero.

SF is set if the MSB of the result of an operation is 1. SF is used with unsigned numbers.

OF is used only for signed arithmetic operation and is set if the result is too large to be fitted in the number of bits available to accommodate it.

The functions of the flags along with their bit positions are shown in Fig. 11.9 below.

Bit position	Name	Function
0	CF	Carry flag: Set on high-order bit carry or borrow; cleared otherwise
2	PF	Parity flag: Set if low-order 8-bit of result contain an even number of 1-bit; cleared otherwise
4	AF	Set on carry from or borrow to the low-order 4-bits of AL; cleared otherwise

6	ZF	Zero flag: Set if result is zero; cleared otherwise
7	SF	Sign Flag: Set equal to high-order bit of result (0 is positive, 1 if negative)
8	TF	Signal step flag: Once set, a single-step interrupt occurs after the next instruction executes; TF is cleared by the single-step interrupt
9	IF	Interrupt-enable flag: When set, maskable interrupts will cause the CPU to transfer control to an interrupt vector specified location.
10	DF	Direction flag: Causes string instructions to auto decrement the appropriate index register when set; clearing DF causes auto increment.
11	OF	Overflow flag: Set if the signed result cannot be expressed within the number of bits in the destination operand; cleared otherwise.

Fig.11.9: 8086 flags: DF, IF and TF can be set or reset to control the operations of the processor. The remaining flags are status indicators.

17. Discuss the three control flags of 8086.

Ans. The three control flags of 8086 are TF, IF and DF. These three flags are programmable, i.e., can be set/reset by the programmer so as to control the operation of the processor.

When TF (trap flag) is set (=1), the processor operates in single stepping mode—i.e., pausing after each instruction is executed. This mode is very useful during program development or program debugging.

When an interrupt is recognised, TF flag is cleared. When the CPU returns to the main program from ISS (interrupt service subroutine), by execution of IRET in the last line of ISS, TF flag is restored to its value that it had before interruption.

TF cannot be directly set or reset. So indirectly it is done by pushing the flag register on the stack, changing TF as desired and then popping the flag register from the stack.

When IF (interrupt flag) is set, the maskable interrupt INTR is enabled otherwise disabled (i.e., when IF = 0).

IF can be set by executing STI instruction and cleared by CLI instruction. Like TF flag, when an interrupt is recognised, IF flag is cleared, so that INTR is disabled. In the last line of ISS when IRET is encountered, IF is restored to its original value.

When 8086 is reset, IF is cleared, i.e., resetted.

DF (direction flag) is used in string (also known as block move) operations. It can be set by STD instruction and cleared by CLD. If DF is set to 1 and MOVS instruction is executed, the contents of the index registers DI and SI are automatically decremented to access the string from the highest memory location down to the lowest memory location.

18. Discuss the Pointers and Index group of registers.

Ans. The pointer registers are SP and BP while the index registers are SI and DI.

All the four are 16-bit registers and are used to store offset addresses of memory locations relative to segment registers. They act as memory pointers. As an example, MOV AH, [SI] implies, "Move the byte whose address is contained in SI into AH". If now, SI = 2000 H, then execution of above instruction will put the value FF H in register AH, shown in Fig. 11.10, [SI+1 : SI] = ABFF H, where obviously SI+1 points to memory location 2001 H and [SI+1] = AB H.

SI and DI are also used as general purpose registers. Again in certain string (block move) instructions, SI and DI are used as source and destination index registers

respectively. For such cases, contents of SI are added to contents of DS register to get the actual source address of data, while the contents of DI are added to the contents of ES to get the actual destination address of data.

SP and BP stand for stack pointer and base pointer with SP containing the offset address or the stack top address. The actual stack address is computed by adding the contents of SP and SS.

Data area(s) may exist in stack. To access such data area in stack segment, BP register is used which contains the offset address. BP register is also used as a general purpose register.

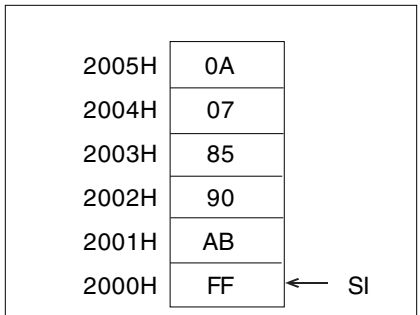


Fig. 11.10: SI is pointing to memory locations 2000 H.

Instruction pointer (IP) is also included in the index and pointers group. IP points to the offset instead of the actual address of the next instruction to be fetched (from the current code segment) in BIU. IP resides in BIU but cannot be programmed by the programmer.

19. Describe in brief the four segment registers.

Ans. The four segment registers are CS, DS, ES and SS—standing for code segment register, data segment register, extra segment register and stack segment register respectively. When a particular memory is being read or written into, the corresponding memory address is determined by the content of one of these four segment registers in conjunction with their offset addresses.

The contents of these registers can be changed so that the program may jump from one active code segment to another one.

The use of these segment registers will be more apparent in memory segmentation schemes.

20. Discuss A_{16}/S_3—A_{19}/S_6 Signals of 8086.

Ans. These are time multiplexed signals. During T_1, they represent $A_{19} - A_{16}$ address lines. During I/O operations, these lines remain low. During T_2–T_4, they carry status signals.

S_4 and S_3 (during T_2 to T_4) identify the segment register employed for 20-bit physical address generation.

Status signal S_5 (during T_2 to T_4) represents interrupt enable status. This is updated at the beginning of each clock cycle.

Status signal S_6 remains low during T_2 to T_4.

21. Discuss $\overline{BHE}/S_7$ signal.

Ans. During T_1, this becomes bus high enable signal and remains low while during T_2 to T_4 it acts as a status signal S_7 and remains high during this time.

During T_1, when $\overline{BHE}$ signal is active, i.e., remains low, it is used as a chip select signal on the higher byte of data bus—i.e., D_{15}–D_8.

Table 11.2 shows $\overline{BHE}$ and A_0 signals determine one of the three possible references to memory.

Table 11.2: Status of $\overline{BHE}$ and A_0 identify memory references

$\overline{BHE}$	A_0	Word/byte access
0	0	Both banks active, 16-bit word transfer on $AD_{15} - AD_0$
0	1	Only high bank active, upper byte from/to odd address on $AD_{15} - AD_8$
1	0	Only low bank active, lower byte from/to even address AD_7–AD_0
1	1	No bank active

22. Discuss the Reset pin of 8086.

Ans. Reset is an active high input signal and must be active for at least 4 CLK cycles to be accepted by 8086. This signal is internally synchronised and execution starts only after Reset returns to low value.

For proper initialisation, Reset pulse must *not* be applied before 50μS of 'power on' of the circuit. During Reset state, all three buses are tristated and ALE and HLDA are driven low.

During resetting, all internal register contents are set to 0000 H, but CS is set to F000 H and IP to FFF0 H. Thus execution starts from physical address FFFF0 H. Thus EPROM in 8086 is interfaced so as to have the physical memory location forms FFFF0 H to FFFFF H, i.e., at the end of the map.

23. Discuss the two pins (a) DT/$\overline{R}$ and (b) $\overline{DEN}$.

Ans. (a) DT/$\overline{R}$ is an output pin which decides the directions of data flow through the transreceivers (bidirectional buffers).

When the processor sends out data, this signal is 1 while when it receives data, the signal status is 0.

(b) $\overline{DEN}$ stands for data enable. It is an active low signal and indicates the availability of data over the address/data lines. This signal enables the transreceivers to separate data from the multiplexed address/data signal. It is active from the middle of T_2 until the middle of T_4.

Both DT/$\overline{R}$ and $\overline{DEN}$ are tristated during 'hold acknowledge'.

24. Elaborate the functions of the pins $\overline{S_2}$, $\overline{S_1}$ and $\overline{S_0}$.

Ans. These three are output status signals in the MAX mode, indicating the type of operation carried out by the processor.

The signals become active during T_4 of the previous cycle and remain active during T_1 and T_2 of the current cycle. They return to the passive state during T_3 of the current bus cycle so that they may again become active for the next bus cycle during T_4. Table 11.3 shows the different bus cycles of 8086 for different combinations of these three signals.

Table 11.3: Bus status codes

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	CPU cycles
0	0	0	Interrupt acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	HALT
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive

25. Explain the $\overline{\text{LOCK}}$ signal.

Ans. It is an active low output signal and is activated by LOCK prefix instruction and remains active until the completion of the next instruction. It floats to tri-state during hold acknowledge when $\overline{\text{LOCK}}$ signal is low, all interrupts get masked and HOLD request is not granted. $\overline{\text{LOCK}}$ signal is used by the processor to prevent other devices from accessing the system control bus. This symbol is used when CPU is executing some critical instructions and through this signal other devices are informed that they should not issue HOLD signal to 8086.

26. Explain the $\overline{\text{TEST}}$ signal.

Ans. It is an active low input signal. Normally the BUSY pin (output) of 8087 NDP is connected to the $\overline{\text{TEST}}$ input pin of 8086. When maths co-processor 8087 is busy executing some instructions, it pulls its BUSY signal high. Thus the $\overline{\text{TEST}}$ signal of 8086 is consequently high, and it (8086) is made to WAIT until the BUSY signal goes low. When 8087 completes its instruction executions, BUSY signal becomes low. Thus the $\overline{\text{TEST}}$ input of 8086 becomes low also and then only 8086 goes in for execution of its program.

27. Show how demultiplexing of address/data bus is done and also show the availability of address/data during read/write cycles.

Ans. The demultiplexing of lower 2-bytes of address/data bus (AD_0–AD_{15}) is done by 8282/8283 octal latch with 8282 providing non-inverting outputs while 8283 gives out inverted outputs. The chip outputs are also buffered so that more drive is available at their outputs.

A D latch is central to the demultiplexing operations of these latches. During T_1 when ALE is high, the latch is transparent and the output of latch is 'A' (address) only. At the end of T_1, ALE has a high to low transition which latches the address available at the D input of the latch, so that address is continued to be available from the Q output of the latch (i.e., whole of T_1 to T_4 states).

It is to be noted that memory and I/O devices do not access the data bus until the beginning of T_2, thus the 'data' is a 'don't care' till the end of T_1. This is shown in Fig. 11.11 and the timing diagram shows the availability of data for read and write cycles.

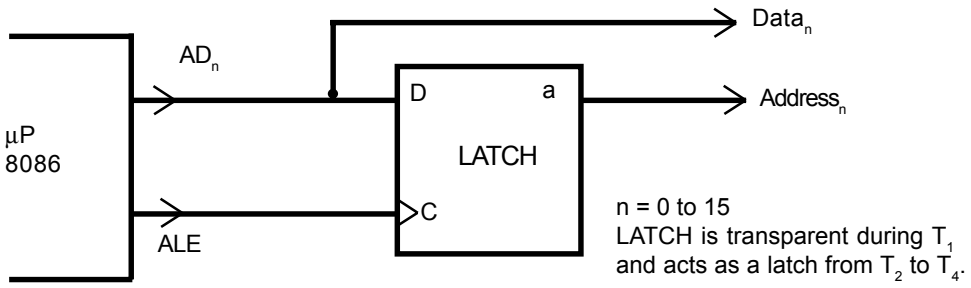


Fig.11.11: Demultiplexing the 8086 address/data bus

28. Discuss the Instruction Pointer (IP) of 8086.

Ans. Functionally, IP plays the part of Program Counter (PC) in 8085. But the difference is that IP holds the offset of the next word of the instruction code instead of the actual address (as in PC).

IP along with CS (code segment) register content provide the 20-bit physical (or real) address needed to access the memory. Thus CS:IP denotes the value of the memory address of the next code (to be fetched from memory).

Content of IP gets incremented by 2 because each time a word of code is fetched from memory.

29. Indicate the data types that can be handled by 8086 µP.

Ans. The types of data formats that can be handled by 8086 fall under the following categories:

- Unsigned or signed integer numbers—both byte-wide and word-wide.
- BCD numbers—both in packed or unpacked form.
- ASCII coded data. ASCII numbers are stored one number per byte.

30. Compare 8086 and 8088 microprocessors.

Ans. The Comparison between the two is tabulated below in Table 11.4.

Table 11.4: Comparison of 8086 and 8088

8086	8088
1. 2-byte data width, obtained by demultiplexing AD_0–AD_{15}.	1. 1-byte data width, obtained by demultiplexing AD_0–AD_7.
2. In MIN mode, pin-28 is assigned the signal M/$\overline{IO}$.	2. In MIN mode, pin-28 is assigned the signal IO/$\overline{M}$.
3. A 6-byte instruction queue.	3. A 4-byte instruction queue.
4. To access higher byte, $\overline{BHE}$ signal is used.	4. No such signal required, since data width is 1-byte only.
5. BIU dissimilar, but EU similar to 8088. Program instructions identical to 8088.	5. BIU dissimilar, but EU similar to 8086. Program instructions identical to 8086.
6. Program fetching from memory done only when 2-bytes are empty in queue.	6. Program fetching from memory done as soon as a byte is free in queue.
7. Pin-34 is $\overline{BHE}$/S_7. During T_1, $\overline{BHE}$ is used to enable data on D_8–D_{15}. During T_2–T_4, status of this pin is 0. In MAX mode, 8087 monitors this pin to identify the CPU—8086 or 8088? Accordingly it sets its queue length to 6 or 4 respectively.	7. Pin-34 is $\overline{SS_0}$. It acts as $\overline{S_0}$ in the MIN mode. In MAX mode $\overline{SS_0}$ = 1 always.

31. Comment on the instruction size of 8086.

Ans. It varies from 1 to 6 bytes.

32. Discuss the instruction format of 8086.

Ans. The instruction format of 8086 is shown in Fig. 11.12. It is extendable up to 6-bytes. The first byte contains D and W—Direction Register Bit and Data Size Bit respectively. Both D and W are 1-bit in nature.

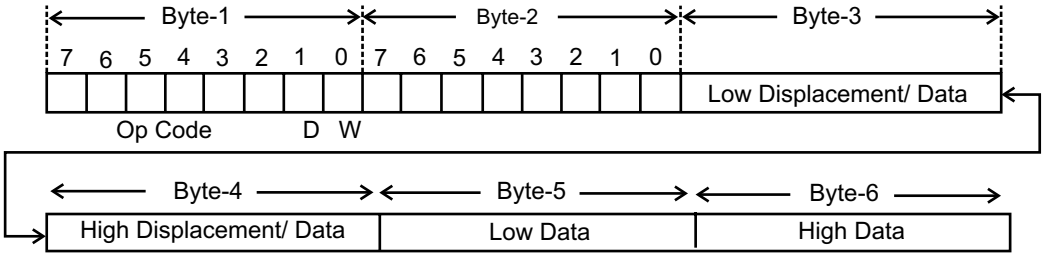


Fig. 11.12: 8086 Instruction format

- If D = 1, then register operand existing in byte-2 is the destination operand, otherwise (i.e., if D = 0) it is a source operand.
- W indicates whether the operation is an 8-bit or a 16-bit data. If W = 0 then it is an 8-bit operation, else (i.e., W = 1) it is 16-bit one.
- The 2nd byte (byte-2) indicates whether one of the operands is in memory or both are in registers. This byte contains three fields:

Field	Abbreviation	Length (no. of bits)
Mode field	MOD	2
Register field	REG	3
Register/Memory field	r/m	3

33. Discuss the MOD and r/m and REG fields.

Ans. MOD field is a 2-bit field. It addresses memory in the following manner.

MOD field values

0 0 ⟶ Memory addressing without displacement
0 1 ⟶ Memory addressing with 8-bit displacement
1 0 ⟶ Memory addressing with 16-bit displacement
1 1 ⟶ Register addressing with
W = 0 for 8-bit data
and W = 1 for 16-bit data

The r/m field which is a 3-bit field, along with MOD field defines the 2nd operand. If MOD = 11, then it is a register to register mode. Again if MOD = 00,01 or 10 then it is a memory mode. Table 11.5 shows how effective address of memory operand gets selected for MOD = 00,01 and 10 values.

Table 11.5: For effective address calculation, values of MOD and r/m

r/m	MOD 00	MOD 01	MOD 10	MOD 11	
				W = 0	W = 1
000	[BX] + [SI]	[BX]+[SI]+D8	[BX]+[SI]+D16	AL	AX
001	[BX] + [DI]	[BX]+[DI]+D8	[BX]+[DI]+D16	CL	CX
010	[BP] + [SI]	[BP]+[SI]+D8	[BP]+[SI]+D16	DL	DX
011	[BP] + [DI]	[BP]+[DI]+D8	[BP]+[DI]+D16	BL	BX
100	[SI]	[SI]+D8	[SI]+D16	AH	SP
101	[DI]	[DI]+D8	[DI]+D16	CH	BP
110	Direct Addressing	[BP]+D8	[BP]+D16	DH	SI
111	[BX]	[BX]+D8	[BX]+D16	BH	DI

Again, for MOD values 00,01 and 10, the default segment registers selected are shown in Table 11.6.

Table 11.6: Segment register for various memory addressing modes

r/m	MOD 00	MOD 01	MOD 10	Segment Register used
000	[BX] + [SI]	[BX]+[SI]+DS	[BX]+[SI]+D16	DS
001	[BX] + [DI]	[BX]+[DI]+DS	[BX]+[DI]+D16	DS
010	[BP] + [SI]	[BP]+[SI]+DS	[BP]+[SI]+D16	SS
011	[BP] + [DI]	[BP]+[DI]+DS	[BP]+[DI]+D16	SS
100	[SI]	[SI]+DS	[SI]+D16	DS
101	[DI]	[DI]+DS	[DI]+D16	DS
110	D16 Direct Addressing [DS]	[BP]+DS Stack pointer register [SS]	[BP]+D16 Stack segment register [SS]	DS or SS as in MOD column
111	[BX]	[BX]+DS	[BX]+D16	DS

The REG field is a 3-bit field and indicates the register for the first operand which can be source/destination operand, depending on D = 0/1.

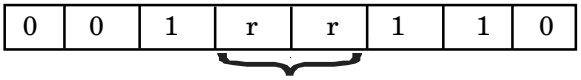
How REG field along with the status of W(0 or 1) select the different registers, is shown in Table 11.7.

Table 11.7: Definition of registers with 'W'

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

34. Discuss the instruction format for segment override prefix.

Ans. Default segment selection can be overriden by the override prefix byte, as shown in below.



Depending on the 2-bit rr values, the segments selected are shown in Table 11.8.

Table 11.8 : Segment selection by override prefix technique

rr values	Segments selected
00	ES
01	CS
10	SS
11	DS

The override prefix byte follows the opcode byte of the instruction, whenever used.

35. Is direct memory to memory data transfer possible in 8086?

Ans. No, 8086 does not have provision for direct memory to memory data transfer.

For this to be implemented, AX is used as an intermediate stage of data. The source byte (from the memory) is moved into AX register with one instruction. The second instruction moves the content of AX into destination location (into another memory location). As example,

```
MOV AH, [SI]
MOV [DI], AH
```

Here, the first instruction moves the content of memory location, whose offset address remains in SI, into AH. The second instruction ensures that the content of AH is moved into another memory location whose offset address is in DI.

36. Can the data segment (DS) register be loaded directly by its address?

Ans. No, it cannot be done directly. Instead, AX is loaded with the initial address of the DS register and then it is transferred to DS register, as shown below:

```
MOV AX, DS ADDR:   AX is loaded with initial address of DS register
MOV DS, AX:        DS is loaded with AX, i.e., ultimately with DS ADDR
```

37. Show, in tabular form, the default and alternate segment registers for different types of memory references.

Ans. Table 11.9 shows the default and alternate register segments which can be used for different types of memory references.

Table 11.9: Default and alternate register assignments

Type of memory reference	Default segment	Alternative segment	Offset (Logical address)
Instruction fetch	CS	None	IP
Stack operation	SS	None	SP, BP
General data	DS	CS, ES, SS	Effective address
String source	DS	CS, ES, SS	SI
String destination	ES	None	DI
BX used as pointer	DS	CS, ES, SS	Effective Address
BP used as pointer	SS	CS, ES, DS	Effective Address

APPENDIX I : A Stock Control Example
Program Flow Chart

APPENDIX II : Reserved Symbols

APPENDIX III : Special Punched Card Conventions

APPENDIX IV : Example of a Flow Chart produced by a Computer

1. INTRODUCTION

1.1 General

Today flow charts are in wide use for the diagrammatic representation of processes. They consist of symbols with appropriate text, and connecting lines. Such a representation can be followed more easily than a narrative description, chiefly because of its two-dimensional structure. It also simplifies checking for completeness and logical consistency.

This Standard for Flow Charts is not intended as a primer in flow charting, but is designed to ensure the general intelligibility of the diagrammatic part of flow charts. Note that the diagrammatic part has no meaning by itself until supplemented by the insertion of appropriate text into the symbols, and that this text will not be standardized. The standard is not rigid in that some freedom of adaption is permitted under certain circumstances.

The proposed graphical forms of the symbols were chosen bearing in mind the following objectives :

The number of basic graphical forms should be small.

The symbols should be easily drawn freehand, by means of template or by any automatic process (see Appendix IV).

Those symbols which may contain a varying amount of text should be easily adaptable in size.

1.2 Types of Flow Charts

In connection with data processing there arises a need for two basic types of flow charts. They are named respectively

Program Flow Chart and Data Flow Chart.

The full name is used only, when it is necessary to specify the type explicitly.

A program flow chart describes the flow of control within any computer program, i.e., the order in which the various program steps are to be executed. Therefore, it mainly consists of

(a) flow lines connecting successive program steps,
(b) operational symbols for the actual processing steps,
(c) flow control symbols defining the path to be followed under various conditions.

In so far as the plugboard wiring of a calculator or tabulator can be represented by a flow-chart, this standard is satisfactory. Other operations of plugboard control are not amenable to standardization, since they are too closely involved with the design of particular machines.

A data flow chart shows the flow of data through a processing system. Therefore, it mainly consists of

(a) flow lines indicating transfer of data or transport of data media;

(b) data symbols, namely symbols for data media and storage media;

(c) operational symbols.

Two levels of data flow-charting for punched card equipment may be distinguished :

(a) The presentation of the basic logical operations to be carried out on the data in order to solve the problem.

(b) The presentation of the processing functions required to implement this solution, in which the symbols stand for the work done in each machine operation, whether simple or complex.

In flow charts of level (a), it may be desirable to show the intended assignment of one or more logical operations to processing function which will appear as one symbol at level (b). Such a function may be carried out by one machine or by a group operated closely together. (See Appendix III).

A simple example (see Appendix I) illustrates to some extent, the basic differences between the two types of flow chart. This example makes use of symbols which will be defined later; therefore, additional remarks have been inserted into the charts to make them self-explanatory.

There may be several flow charts of both types to one problem, varying in the degree of detail as well as in their particular aim. Sometimes, also, flow charts are used which contain symbols belonging to both types of flow chart, and connecting lines, representing either control flow or data flow. In this case, care should be taken to distinguish clearly between them. Nevertheless, the ensuing Standard will discuss the two types of flow chart separately.

1.3 Conventions

The following convention applies to the flow lines in both program and data flow-chart :

(a) The direction of flow is mainly

left to right
top to bottom.

If arrow heads are missing, these directions are assumed.

(b) Arrows indicating the flow should be used whenever increased clarity will result.

(c) If a symbol contains more than one line of text, then they are to be read from top to bottom, irrespective of the direction of the flow lines.

(d) A junction is indicated by meeting of two incoming and one outgoing flow lines.

(e) For the sake of clarity there should be no mixture of incoming and outgoing lines at one edge of a symbol.

(f) Flow lines may cross; in this case they have no logical interrelation.

(g) While the standard does not make exact specifications about height to width ratios, it does require the user not to vary these to such an extent that the symbol is not immediately recognizable. For this reason it is also suggested that within a single flow chart the height to width ratios are held constant.

1.4 Presentation of the Standard

The following pages are divided into two vertically. The right hand side is reserved exclusively for definitive symbols of the Standards. Diagrams appearing on the left hand side are eigher illustrations of the use of the symbols or permitted extensions.

1.5 Maintenance of the Standard

It is foreseen that the Standard may require maintenance from time to time.

Special circumstances may arise which are not covered by the Standard. Should users find themselves in such a situation, they are requested to communicate with ECMA. In the meantime other symbols may be used provided that they cannot be confused with symbols already contained in the Standard (see Appendix II). They must be clearly defined, and their definition must be stated with the flow chart in which the y are used.

2. SYMBOLS FOR PROGRAM FLOW CHARTS

2.1 Operational Symbols

2.1.1 General Operational Symbol

The General Operational Symbol is used for any operation which creates, alters, transfers or erases data, or any other operation for which no specific symbol has been defined in the Standard. It can also replace any of the symbols for program flow chart.

REMARK :

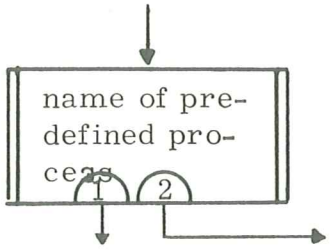
(1) The term "data" is not restricted to I/O data, it also includes instructions, indicators, etc.

2.1.2 Predefined Process Symbol

The Predefined Process Symbol is used when a named section of program is considered as a single operation for the purpose of this flow chart, e.g. a subroutine.

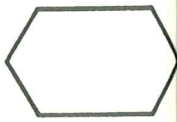
REMARK :

(1) In cases where the predefined process has more than one entry or exit, the symbol may show these and should include the reference to the used entry or exit.
Example :



2.1.3 Input/Output Symbol

The Input/Output Symbol is used where it is desired to stress I/O operations.



2.1.4 Preparation Symbol

The Preparation Symbol is used when it is desired to accentuate that an operation partially or completely determines the selection of a particular exit at given Branch Symbols (see 2.2.1). The Preparation Symbol is commonly used in the following ways :

Prepare a Decision

The creation or alternation of an indicator, or a quantity, which appears either in a Branch Symbol, or in another Preparation Symbol;

Set a Switch

Setting one or more switches by selecting one exit in each of them.

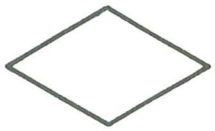
Initialize a routine

For example setting account, clearing registers, etc.

2.2 Flow Control Symbols

The Flow Control Symbols are used to represent the points where flow lines diverge or converge.

2.2.1 Branch Symbol



The Branch Symbol has one entry line and more than one exit line. In passing through this symbol, one and only one exit will be used.

The Branch Symbol is commonly used in the following two ways :

Decision

The symbol contains a description of the test on which the selection of an exit is based. The various possible results of this test may be shown against the corresponding exits.

Switch

The symbol contains the name of the switch. The possible settings may be shown against the exits. When reaching this symbol, one exit is already set (see 2.1.4).

2.2.2 Flow Line Symbol

This symbol connects successive program steps. (See par. 1.3 convention a)

Crossing of flow lines
(see par. 1.3 convention f)

Junction of flow lines
(see par. 1.3 convention d)

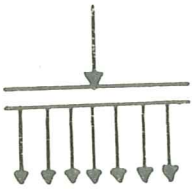
Examples :

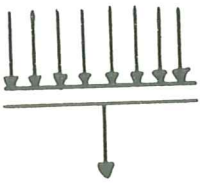
Simple junction

Multiple junction

2.2.3 Parallel Mode Symbol

There may be occasions, in a program flow chart, when two or more paths of control operate simultaneously, and it is necessary to define their relation. In this case the Parallel Mode Symbol is used. After it has been reached through all the entry lines, all the exit lines are used in parallel and/or arbitrary sequence. There are two special cases :





2.3 Auxiliary Symbols

2.3.1 Connector Symbol

The Connector Symbol represents an exit to, or an entry from another part of the flow chart. Related exit(s) and entry must have the same identifier.

2.3.2 Terminal Symbol

The Terminal Symbol is used for the beginning of a flow line, e.g. the start of a program or the entrance to a subroutine. It is also used for the end of a flow line, e.g. the end of a program, the exit from a subroutine, the return to a control program or interrupt.

2.3.3 Comment Symbol

The Comment Symbol is designed to contain additional information which it is desired to include at this point of the chart.

REMARK :

(1) This symbol may be attached to flow lines and to any other symbol. The information may be either text or reference to text elsewhere.

3. SYMBOLS FOR DATA FLOW CHARTS

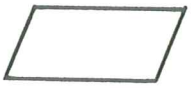
3.1 Data Symbols

The main task for data symbols is to represent the existence of certain data; as a by-product, they may also give some information about the medium on which the data is held. The process of writing on, or reading from a medium

is not represented explicitly, since the flow lines connecting the symbols imply this.

When the data is implicitly indicated by the preceding and following operational symbols, the symbol for the data element may be omitted.

3.1.1 General Data Symbol

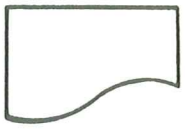


3.1.2 Specific Data Symbols

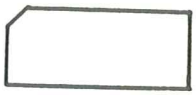
a) Source document

The general data symbol (3.1.1) is considered to suffice for this purpose.

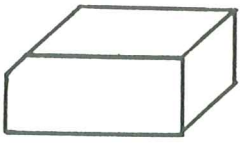
b) Printed Document Symbol



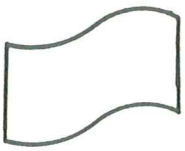
c) Punched Card Symbol
(see also Appendix III)



d) Punched Card Deck Symbol
(see also Appendix III)

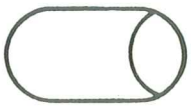


e) Punched Tape Symbol

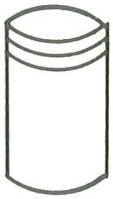


f) Magnetic Tape Symbol

g) Magnetic Drum Symbol

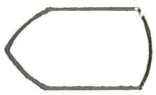


h) Magnetic Disk Symbol



i) Core Symbol

j) Display Symbol



3.2 Operational Symbol

The main purpose of an Operational Symbol is to indicate the operation performed on the preceding data in order to obtain new data; as a by-product, it may also give some information about the hardware unit which is used for this operation. When the operation is sufficiently indicated by the preceding and following Data Symbol, the Operational Symbol may be omitted (e.g., card-to-tape conversion).

3.2.1 General Operational Symbol

The General Operational Symbol may be used for any operation on data.

3.2.2 Specific Operational Symbols

(a) Merging

DEFINITION :

The formation of an ordered set of items from two or more ordered sets sequenced according to a common key.
(Shown here with two entry lines only)

(b) Extracting

DEFINITION :

The selection from a single set of items of one or more subsets, each of which meets some criterion. If the single set is sequenced, the subsets will be sequenced accordingly.
(Shown here with two exit lines only)

(c) Collating

DEFINITION :

Merging with extracting.
(Shown here with two entry lines and two exit lines only)

(d) Sorting / Sequencing

DEFINITION :

To arrange a set of items in sequence according to a certain key. (The entry line and the exit line are not part of the symbol.)

(e) Manual Intervention Symbol

The Manual Intervention Symbol is used for the introduction of information at the time of processing by manual action, e.g., by operating a keyboard or a console, or equivalent action.

3.2.3 Manual Operation Symbol

This Manual Operation Symbol represents any offline process geared to the speed of a human being.

3.3 Data Transfer Symbols

3.3.1 Flow Line Symbol

The Flow Line Symbol indicates transfer of data or transport of data media. (See par. 1.3, convention a).

REMARKS :

(1) If it should be necessary to stress transportation of data medium, this may be represented as follows :

(2) When the process described by a flow line takes place beyond the time scale of the flow chart, the line may be shown dotted :

(3) Crossing of flow lines
(see par. 1.3, convention f)

Junction of flow lines
(see par. 1.3, convention d)

Examples :

Simple junction

Multiple junction

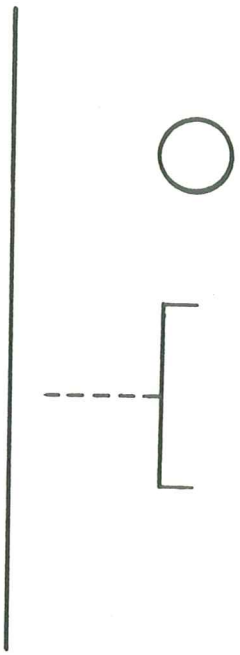
3.3.2 Communication Link Symbol

The Communication Link Symbol is used when the information is transmitted by a telecommunication link. (See par. 1.3, convention a).

3.4 Auxiliary Symbols

3.4.1 Connector Symbol

The Connector Symbol represents an exit to, or an entry from another part of the flow chart. Related exit(s) and entry must have the same identifier.



3.4.2 Comment Symbol

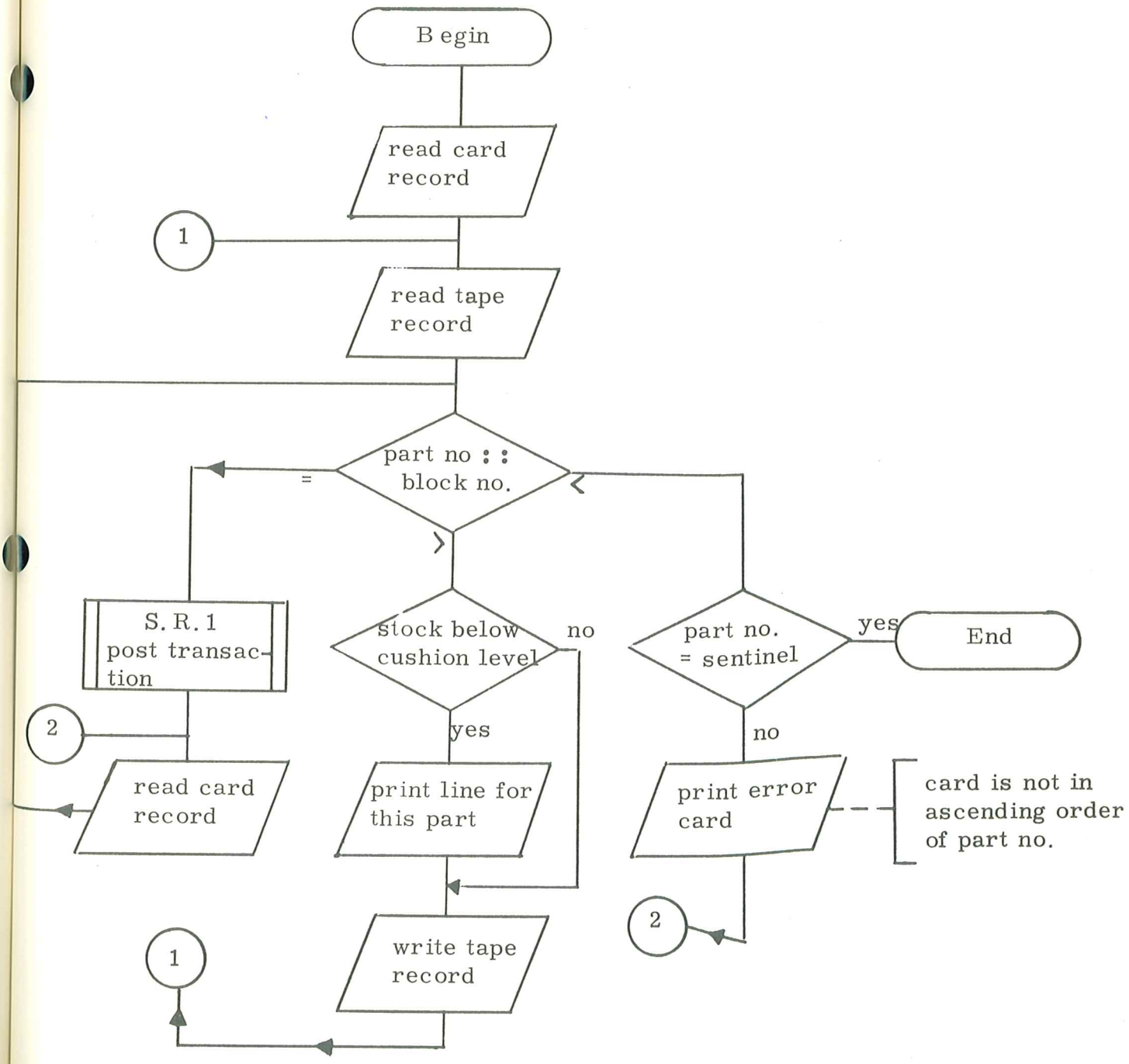
The Comment Symbol is designed to contain additional information which it is desired to include at this point of the chart.

REMARK :

(1) This symbol may be attached to flow lines and to any other symbol. The information may be either text or reference to text elsewhere.

Program Flow Chart

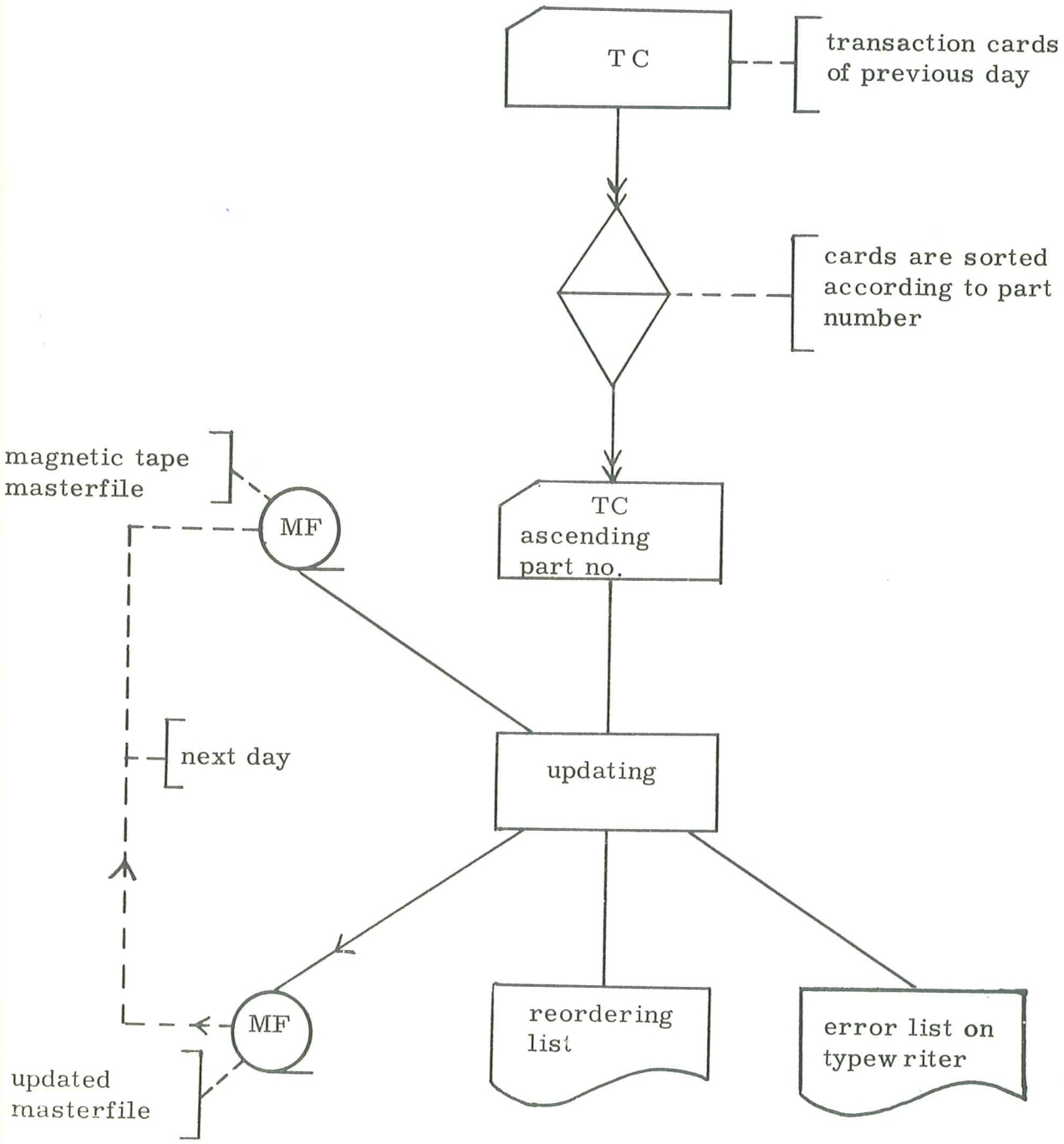
This program flow chart might represent the updating process shown in the data flow chart on preceding page.



APPENDIX I

A STOCK CONTROL EXAMPLE

Data Flow Chart



APPENDIX II

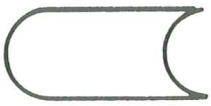
RESERVED SYMBOLS

If it is wished to extend the Standard to cater for new concepts or individual requirements, then care must be taken not to use any of the following symbols with any other meaning than that indicated in the ISO Recommendation :

File of cards :

Offline Storage :

Online Storage :

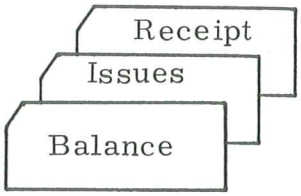


Auxiliary Operation :

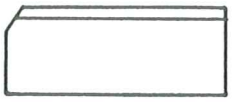
APPENDIX III

SPECIAL PUNCHED CARD CONVENTIONS

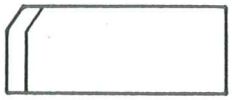
1. The sequencing of card types may be shown by partially superimposing two or more Punched Card Symbols :



2. Interpreted card can be shown as follows :



3. Reproduced card can be shown as follows :



4. The Punched Card Deck Symbol may be sub-divided to express the passage of two or more complete files in a given order through the following operation :