Unit 3

Registers

Data Registers

- AX = Accumulator Register
- BX = Base Register
- DX = Data Register
- CX = Count Register

Index Registers

- SI = Source Index
- DI = Destination Index

Segment Registers

- DS = Data Segment
- SS = Stack Segment
- ES = Extra Segment
- CS = Code Segment

Pointer Registers

- IP = Instruction Pointer
- BP = Base Pointer
- SP = Stack Pointer

Segment Register:

CS (Code Segment) :

Code segment (CS) is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions.

Stack segment (SS)

is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction.

Data segment (DS)

is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. DS register can be changed directly using POP and LDS instructions.

Extra segment (ES)

is a 16-bit register containing address of 64KB segment, usually with program data. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions. ES register can be changed directly using POP and LES instructions. It is possible to change default segments used by general and index registers by prefixing instructions with a CS, SS, DS or ES prefix.

IP (Instruction Pointer) :

To access instructions the 8086 uses the registers CS and IP. The CS register contains the segment number of the next instruction and the IP contains the offset. IP is updated each time an instruction is executed so that it will point to the next instruction. Unlike other registers the IP can’t be directly manipulated by an instruction, that is, an instruction may not contain IP as its operand.

General Registers :

All general registers of the 8086 microprocessor can be used for arithmetic and logic operations. The general registers are:

AX (Accumulator):

This is accumulator register. It gets used in arithmetic, logic and data transfer instructions. In manipulation and division , one of the numbers involved must be in AX or AL.

BX (Base Register):

This is base register. BX register is an address register. It usually contain a data pointer used for based, based indexed or register indirect addressing.

CX (Count register):

This is Count register. This serves as a loop counter. Program loop constructions are facilitated by it. Count register can also be used as a counter in string manipulation and shift/rotate instruction.

DX (Data Register):

This is data register. Data register can be used as a port number in I/O operations. It is also used in multiplication and division.

SP (Stack Pointer):

This is stack pointer register pointing to program stack. It is used in conjunction with SS for accessing the stack segment.

BP (Base Pointer):

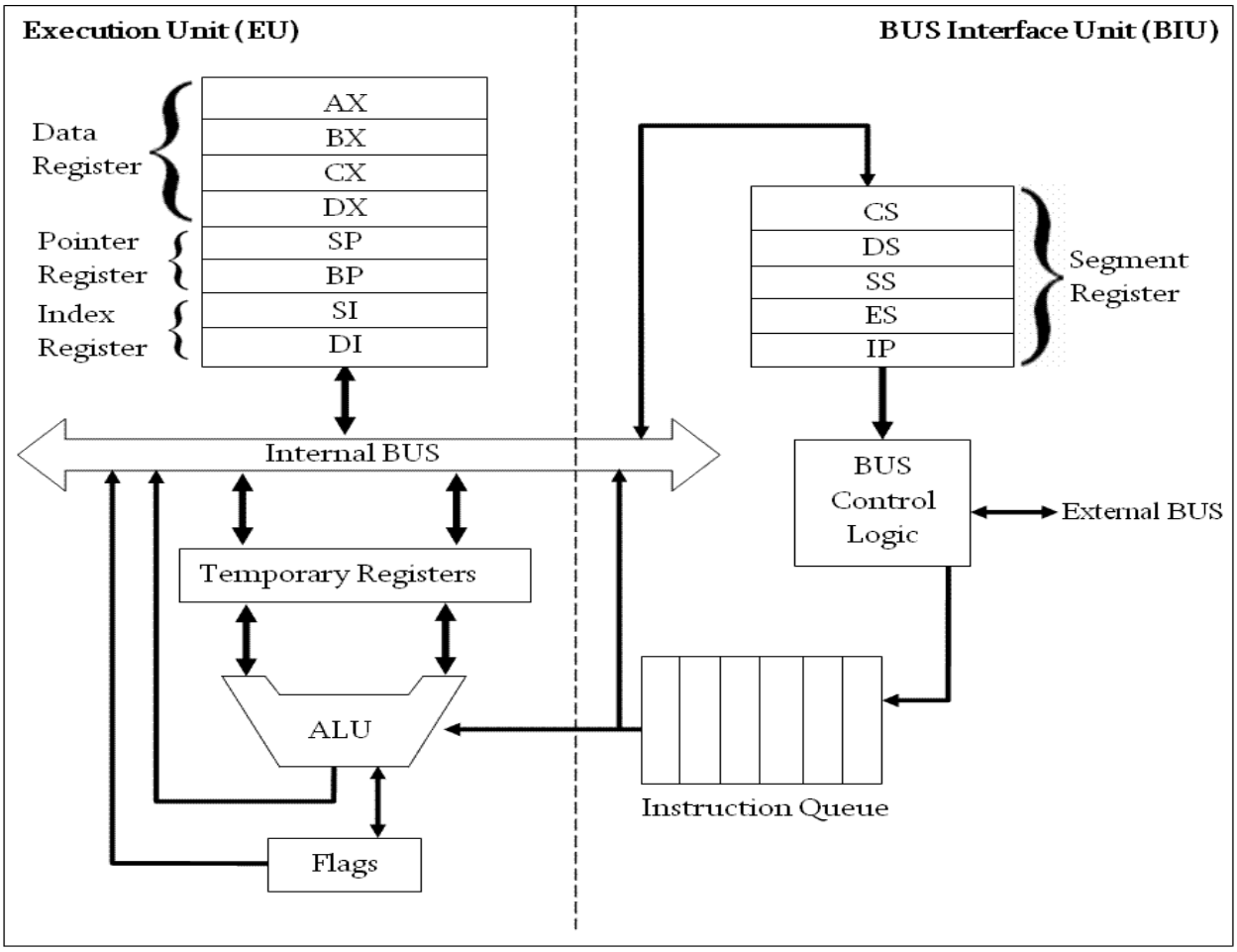
This is base pointer register pointing to data in stack segment. Unlike SP, we can use BP to access data in the other segments.

SI (Source Index): This is source index register which is used to point to memory locations in the data segment addressed by DS. By incrementing the contents of SI one can easily access consecutive memory locations.

DI (Destination Index):

This is destination index register performs the same function as SI. There is a class of instructions called string operations, that use DI to access the memory locations addressed by ES.

8086 Architecture



8086 microprocessor has two units; Execution Unit (EU) and Bus Interface Unit (BIU). They are dependent and get worked by each other. Below is a short description of these two units.

Execution Unit (EU):

Execution unit receives program instruction codes and data from the BIU, executes them and stores the results in the general registers. It can also store the data in a memory location or send them to an I/O device by passing the data back to the BIU. This unit, EU, has no connection with the system Buses. It receives and outputs all its data through BIU.

ALU (Arithmetic and Logic Unit) : The EU unit contains a circuit board called the Arithmetic

and Logic Unit. The ALU can perform arithmetic, such as, +,-,×,/ and logic such as OR, AND, NOT operations.
Registers : A register is like a memory location where the exception is that these are denoted by name rather than numbers. It has 4 data registers, AX, BX, CX, DX and 2 pointer registers SP, BP and 2 index registers SI, DI and 1 temporary register and 1 status register FLAGS .

AX, BX, CX and DX registers has 2 8-bit registers to access the high and low byte data registers. The high byte of AX is called AH and the low byte is AL. Similarly, the high and low bytes of BX, CX, DX are BH and BL, CH and Cl, DH and DL respectively. All the data, pointer, index and status registers are of 16 bits. Else these, the temporary register holds the operands for the ALU and the individual bits of the FLAGS register reflect the result of a computation.

Bus Interface Unit:

As the EU has no connection with the system Busses, this job is done by BIU. BIU and EU are connected with an internal bus. BIU connects EU with the memory or I/O circuits. It is responsible for transmitting data, addresses and control signal on the busses.
Registers : BIU has 4 segment busses, CS, DS, SS, ES. These all 4 segment registers holds the addresses of instructions and data in memory. These values are used by the processor to access memory locations. It also contain 1 pointer register IP. IP contains the address of the next instruction to executed by the EU.

Instruction Queue : BIU also contain an instruction queue. When the EU executes instructions, the BIU gets up to 6 bytes of the next instruction and stores them in the instruction queue and this process is called instruction prefetch. This is a process to speed up the processor. Also when the EU needs to be connected with memory or peripherals, BIU suspends instruction prefetch and performs the needed operations.

ALU (Arithmetic & Logic Unit):

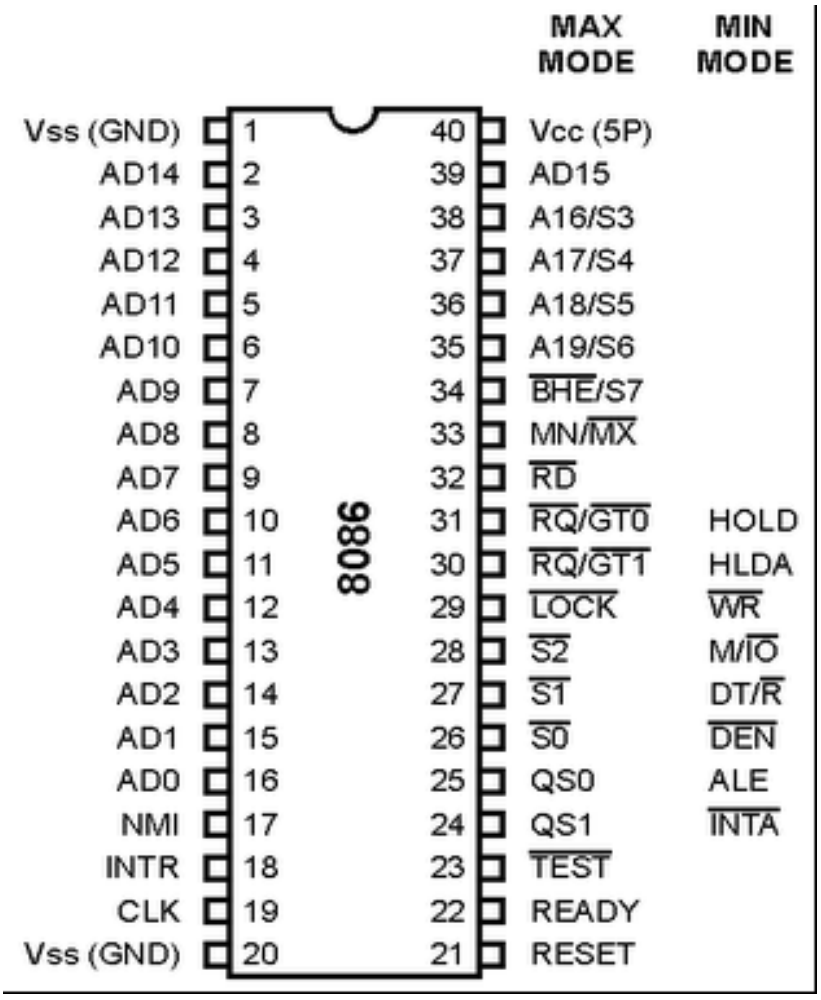
This unit can perform various arithmetic and logical operation, if required, based on the instruction to be executed. It can perform arithmetical operations, such as add, subtract, increment, decrement, convert byte/word and compare etc and logical operations, such as AND, OR, exclusive OR, shift/rotate and test etc.

Purpose of using Instruction Queue:

BIU contains an instruction queue. When the EU executes instructions, the BIU gets up to 6 bytes of the next instruction and stores them in the instruction queue and this process is called instruction prefetch. This is a process to speed up the processor. A subtle advantage of instruction queue is that, as next several instructions are usually in the queue, the BIU can access

memory at a somewhat "leisurely" pace. This means that slow-memory parts can be used without affecting overall system performance.

Pin Diagram and Pin description of 8086



The following pin function descriptions are for the microprocessor 8086 in either minimum or maximum mode.

AD0 - AD15 (I/O): Address Data Bus

These lines constitute the time multiplexed memory/IO address during the first clock cycle (T1) and data during T2, T3 and T4 clock cycles. A0 is analogous to BHE for the lower byte of the data bus, pins D0-D7. A0 bit is Low during T1 state when a byte is to be transferred on the lower portion of the bus in memory or I/O operations. 8-bit oriented devices tied to the lower half would normally use A0 to condition chip select functions. These lines are active high and float to tri-state during interrupt acknowledge and local bus "Hold acknowledge".

A19/S6, A18/S5, A17/S4, A16/S3 (0): Address/Status

During T1 state these lines are the four most significant address lines for memory operations. During I/O operations these lines are low. During memory and I/O operations, status information is available on these lines during T2, T3, and T4 states.S5: The status of the interrupt enable flag bit is updated at the beginning of each cycle. The status of the flag is indicated through this bus.

S6:

When Low, it indicates that 8086 is in control of the bus. During a "Hold acknowledge" clock period, the 8086 tri-states the S6 pin and thus allows another bus master to take control of the status bus.

S3 & S4:

Lines are decoded as follows:

A17/S4	A16/S3	Function
0	0	Extra segment access
0	1	Stack segment access
1	0	Code segment access
1	1	Data segment access

After the first clock cycle of an instruction execution, the A17/S4 and A16/S3 pins specify which segment register generates the segment portion of the 8086 address., This feature also provides a degree of protection by preventing write operations to one segment from erroneously overlapping into another segment and destroying information in that segment.

BHE /S7 (O): Bus High Enable/Status

During T1 state theBHE should be used to enable data onto the most significant half of the data bus, pins D15 - D8. Eight-bit oriented devices tied to the upper half of the bus would normally use BHE to control chip select functions. BHE is Low during T1 state of read, write and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. The S7 status information is available during T2, T3 and T4 states. The signal is active Low and floats to 3-state during "hold" state. This pin is Low during T1 state for the first interrupt acknowledge cycle.

RD (O): READ

The Read strobe indicates that the processor is performing a memory or I/O read cycle. This signal is active low during T2 and T3 states and the Tw states of any read cycle. This signal floats to tri-state in "hold acknowledge cycle".

TEST (I)TEST pin is examined by the "WAIT" instruction. If the TEST pin is Low, execution continues. Otherwise the processor waits in an "idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.

INTR (I): Interrupt Request

It is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector look up table located in system memory. It can be internally masked by software resetting the interrupt enable bit INTR is internally synchronized. This signal is active HIGH.

NMI (I): Non-Muskable Interrupt

An edge triggered input, causes a type-2 interrupt. A subroutine is vectored to via the interrupt vector look up table located in system memory. NMI is not maskable internally by software. A transition from a LOW to HIGH on this pin initiates the interrupt at the end of the current instruction. This input is internally synchronized.

RESET (I)

Reset causes the processor to immediately terminate its present activity. To be recognised, the signal must be active high for at least four clock cycles, except after power-on which requires a 50 Micro Sec. pulse. It causes the 8086 to initialize registers DS, SS, ES, IP and flags to all zeros. It also initializes CS to FFFF H. Upon removal of the RESET signal from the RESET pin, the 8086 will fetch its next instruction from the 20 bit physical address FFFF0H. The reset signal to 8086 can be generated by the 8284. (Clock generation chip). To guarantee reset from power-up, the reset input must remain below 1.5 volts for 50 Micro sec. after Vcc has reached the minimum supply voltage of 4.5V.

READY (I)

Ready is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The READY signal from memory or I/O is synchronized by the 8284 clock generator to form READY. This signal is active HIGH. The 8086 READY input is not synchronized. Correct operation is not guaranteed if the setup and hold times are not met.

CLK (I): Clock

Clock provides the basic timing for the processor and bus controller. It is asymmetric with 33% duty cycle to provide optimized internal timing. Minimum frequency of 2 MHz is required, since the design of 8086 processors incorporates dynamic cells. The maximum clock frequencies of the 8086-4, 8086 and 8086-2 are4MHz, 5MHz and 8MHz respectively.
Since the 8086 does not have on-chip clock generation circuitry, and 8284 clock generator chip must be connected to the 8086 clock pin. The crystal connected to 8284 must have a frequency 3 times the 8086 internal frequency. The 8284 clock generation chip is used to generate READY, RESET and CLK.

MN/MX (I): Maximum / Minimum

This pin indicates what mode the processor is to operate in. In minimum mode, the 8086 itself generates all bus control signals. In maximum mode the three status signals are to be decoded to generate all the bus control signals.
Minimum Mode Pins The following 8 pins function descriptions are for the 8086 in minimum mode; MN/ MX = 1. The corresponding 8 pins function descriptions for maximum mode is explained later.

M/IO (O): Status line

This pin is used to distinguish a memory access or an I/O accesses. When this pin is Low, it accesses I/O and when high it access memory. M / IO becomes valid in the T4 state preceding a bus cycle and remains valid until the final T4 of the cycle. M/IO floats to 3 - state OFF during local bus "hold acknowledge".

WR (O): Write

Indicates that the processor is performing a write memory or write IO cycle, depending on the state of the M /IOsignal. WR is active for T2, T3 and Tw of any write cycle. It is active LOW, and floats to 3-state OFF during local bus "hold acknowledge ".

INTA (O): Interrupt Acknowledge

It is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T2, T3, and T4 of each interrupt acknowledge cycle.

ALE (O): Address Latch Enable

ALE is provided by the processor to latch the address into the 8282/8283 address latch. It is an active high pulse during T1 of any bus cycle. ALE signal is never floated.

DT/ R (O): DATA Transmit/Receive

In minimum mode, 8286/8287 transceiver is used for the data bus. DT/ R is used to control the direction of data flow through the transceiver. This signal floats to tri-state off during local bus "hold acknowledge".

DEN (O): Data Enable

It is provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver. DEN is active LOW during each memory and IO access. It will be low beginning with T2 until the middle of T4, while for a write cycle, it is active from the beginning of T2 until the middle of T4. It floats to tri-state off during local bus "hold acknowledge".

HOLD & HLDA (I/O): Hold and Hold Acknowledge

Hold indicates that another master is requesting a local bus "HOLD". To be acknowledged, HOLD must be active HIGH. The processor receiving the "HOLD " request will issue HLDA (HIGH) as an acknowledgement in the middle of the T1-clock cycle. Simultaneous with the issue of HLDA, the processor will float the local bus and control lines. After "HOLD" is detected as being Low, the processor will lower the HLDA and when the processor needs to run another cycle, it will again drive the local bus and control lines.

Maximum Mode The following pins function descriptions are for the 8086/8088 systems in maximum mode (i.e.. MN/MX = 0). Only the pins which are unique to maximum mode are described below.

S2, S1, S0 (O): Status Pins

These pins are active during T4, T1 and T2 states and is returned to passive state (1,1,1 during T3 or Tw (when ready is inactive). These are used by the 8288 bus controller to generate all

memory and I/O operation) access control signals. Any change by S2, S1, S0 during T4 is used to indicate the beginning of a bus cycle. These status lines are encoded as shown in table 3.

S2	S1	S0	Characteristics
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 access1 0 1 Read memory
1	1	0	Write memory
1	1	1	Passive State

Table 3

QS0, QS1 (O): Queue – Status

Queue Status is valid during the clock cycle after which the queue operation is performed. QS0, QS1 provide status to allow external tracking of the internal 8086 instruction queue. The condition of queue status is shown in table 4.

Queue status allows external devices like In-circuit Emulators or special instruction set extension co-processors to track the CPU instruction execution. Since instructions are executed from the 8086 internal queue, the queue status is presented each CPU clock cycle and is not related to the bus cycle activity. This mechanism allows (1) A processor to detect execution of a ESCAPE instruction which directs the co- processor to perform a specific task and (2) An in-circuit Emulator to trap execution of a specific memory location.

QS1	QS1	Characteristics
0	0	No operation
0	1	First byte of opcode from queue

1	0	Empty the queue
1	1	Subsequent byte from queue

Table 4

LOCK (O)

It indicates to another system bus master, not to gain control of the system bus while LOCK is active Low. The LOCK signal is activated by the "LOCK" prefix instruction and remains active until the completion of the instruction. This signal is active Low and floats to tri-state OFF during 'hold acknowledge'. Example:

LOCK XCHG reg., Memory ; Register is any register and memory GT0

; is the address of the semaphore.

RQ/GT0 and RQ/GT1 (I/O): Request/Grant

These pins are used by other processors in a multi processor organization. Local bus masters of other processors force the processor to release the local bus at the end of the processors current bus cycle. Each pin is bi-directional and has an internal pull up resistors. Hence they may be left un-connected.

Pipelining in 8086

1. The process of fetching the next instruction when the present instruction is being executed is called as pipelining.
2. Pipelining has become possible due to the use of queue.
3. BIU (Bus Interfacing Unit) fills in the queue until the entire queue is full.
4. BIU restarts filling in the queue when at least two locations of queue are vacant.

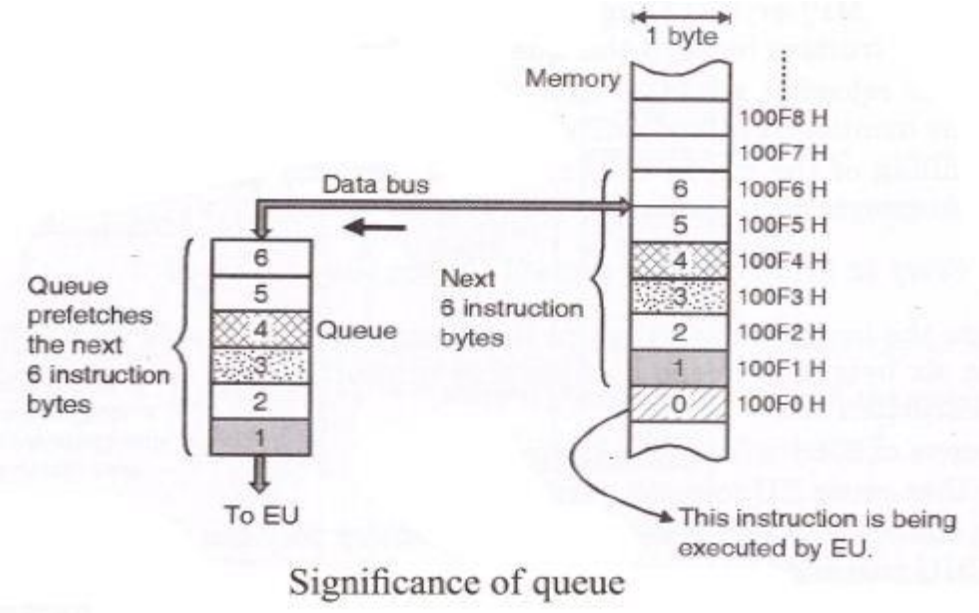
5. **Advantages of pipelining:**

- The execution unit always reads the next instruction byte from the queue in BIU. This is faster than sending out an address to the memory and waiting for the next instruction byte to come.
- In short pipelining eliminates the waiting time of EU and speeds up the processing. -The 8086 BIU will not initiate a fetch unless and until there are two empty bytes in its queue. 8086 BIU normally obtains two instruction bytes per fetch.

The Instruction Queue:

1. The execution unit (EU) is supposed to decode or execute an instruction.
2. Decoding does not require the use of buses.
3. When EU is busy in decoding and executing an instruction, the BIU fetches up to six instruction bytes for the next instructions.
4. These bytes are called as the pre-fetched bytes and they are stored in a first in first out (FIFO) register set, which is called as a queue.

Significance of Queue:

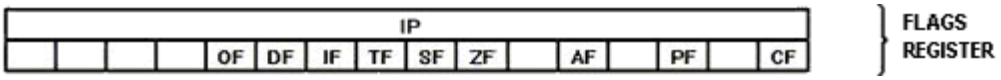


Significance of queue

1. As shown in the above figure, while the EU is busy in decoding the instruction corresponding to memory location 100F0, the BIU fetches the next six instruction bytes from locations 100F1 to 100F6 numbered as 1 to 6.
2. These instruction bytes are stored in the 6 byte queue on the first in first out (FIFO) basis.
3. When EU completes the execution of the existing instruction and becomes ready for the next instruction, it simply reads the instruction bytes in the sequence 1, 2…. from the Queue.
4. Thus the Queue will always hold the instruction bytes of the next instructions to be executed by the EU.

8086 Flag Registers

The 8086 microprocessor has a 16 bit register for flag register. In this register 9 bits are active for flags. This register has 9 flags which are divided into two parts that are as follows



Conditional Flags

Conditional flags represent result of last arithmetic or logical instruction executed. Conditional flags are as follows:

1. **CF (Carry Flag)**

   This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic.

2. **AF (Auxiliary Flag)**

   If an operation performed in ALU generates a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), the AF flag is set i.e. carry given by D3 bit to D4 is AF flag. This is not a general-purpose flag; it is used internally by the processor to perform Binary to BCD conversion.

3. **PF (Parity Flag)**

   This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1's, the Parity Flag is set and for odd number of 1's, the Parity Flag is reset.

4. **ZF (Zero Flag)**

   It is set; if the result of arithmetic or logical operation is zero else it is reset.

5. **SF (Sign Flag)**

   In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.

6. **OF (Overflow Flag)**

   This stands for over flow flag. It occurs when signed numbers are added or subtracted. An OF indicates that the result has exceeded the capacity of machine. It becomes set if the sign result cannot express within the number of bites.

Control Flags

Control flags are set or reset deliberately to control the operations of the execution unit. Control flags are as follows:

1. **TF (Trap Flag):**

   It is used for single step control. It allows user to execute one instruction of a program at a time for debugging. When trap flag is set, program can be run in single step mode.

2. **IF (Interrupt Flag):**

   It is an interrupt enable/disable flag. This stands for interrupt flag. This flag is used to enable or disable the interrupt in a program. If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled. It can be set by executing instruction sit and can be cleared by executing CLI instruction.

3. **DF (Direction Flag):**

   This flag stands for direction flag and is used for the direction of strings. If it is set, string bytes are accessed from higher memory address to lower memory address. When it is reset, the string bytes are accessed from lower memory address to higher memory address