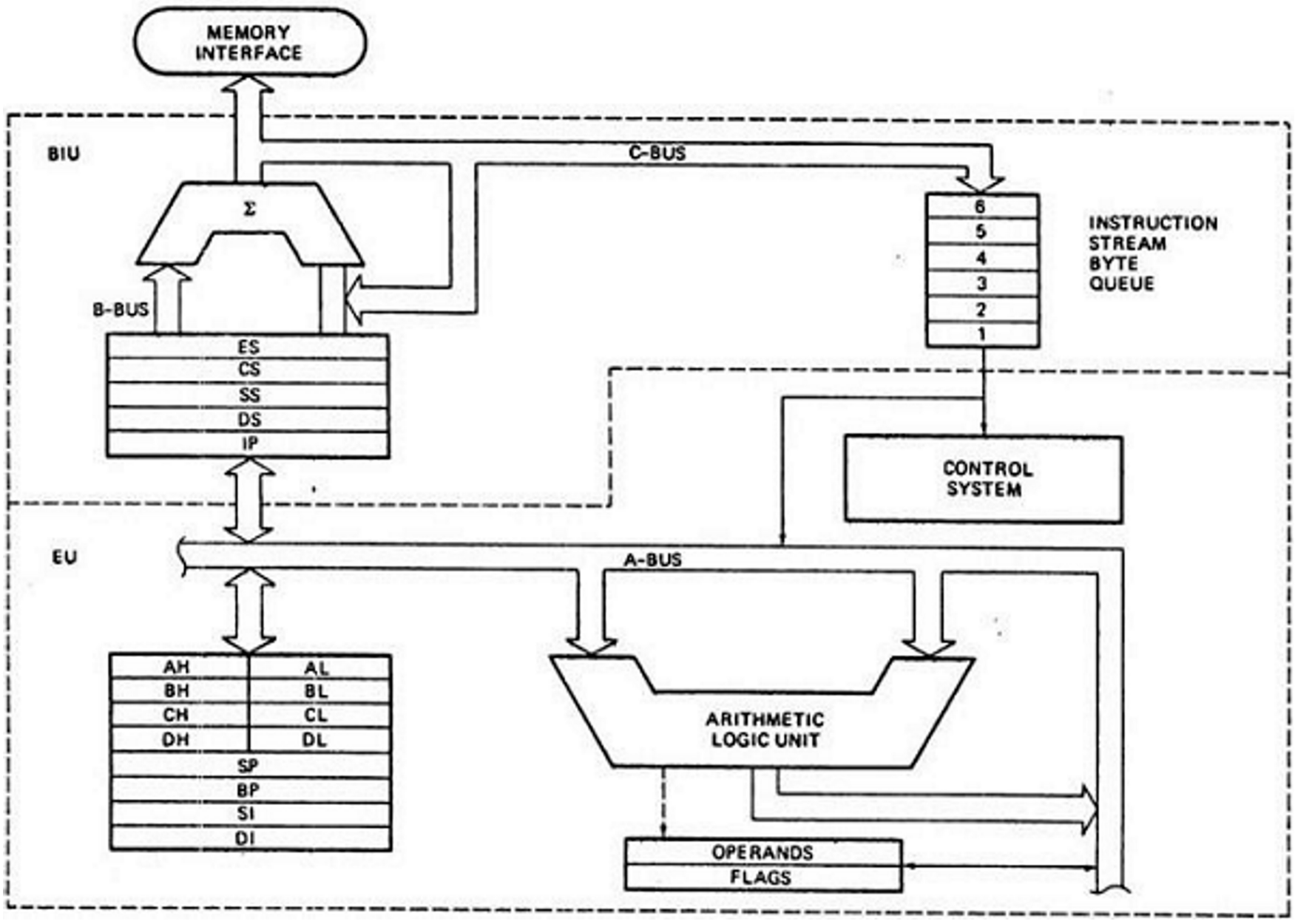
8086 architecture

20 August 2020 01:06



EU (Execution Unit)

Execution unit gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using the instruction decoder & ALU. EU has no direct connection with system buses as shown in the above figure, it performs operations over data through BIU.

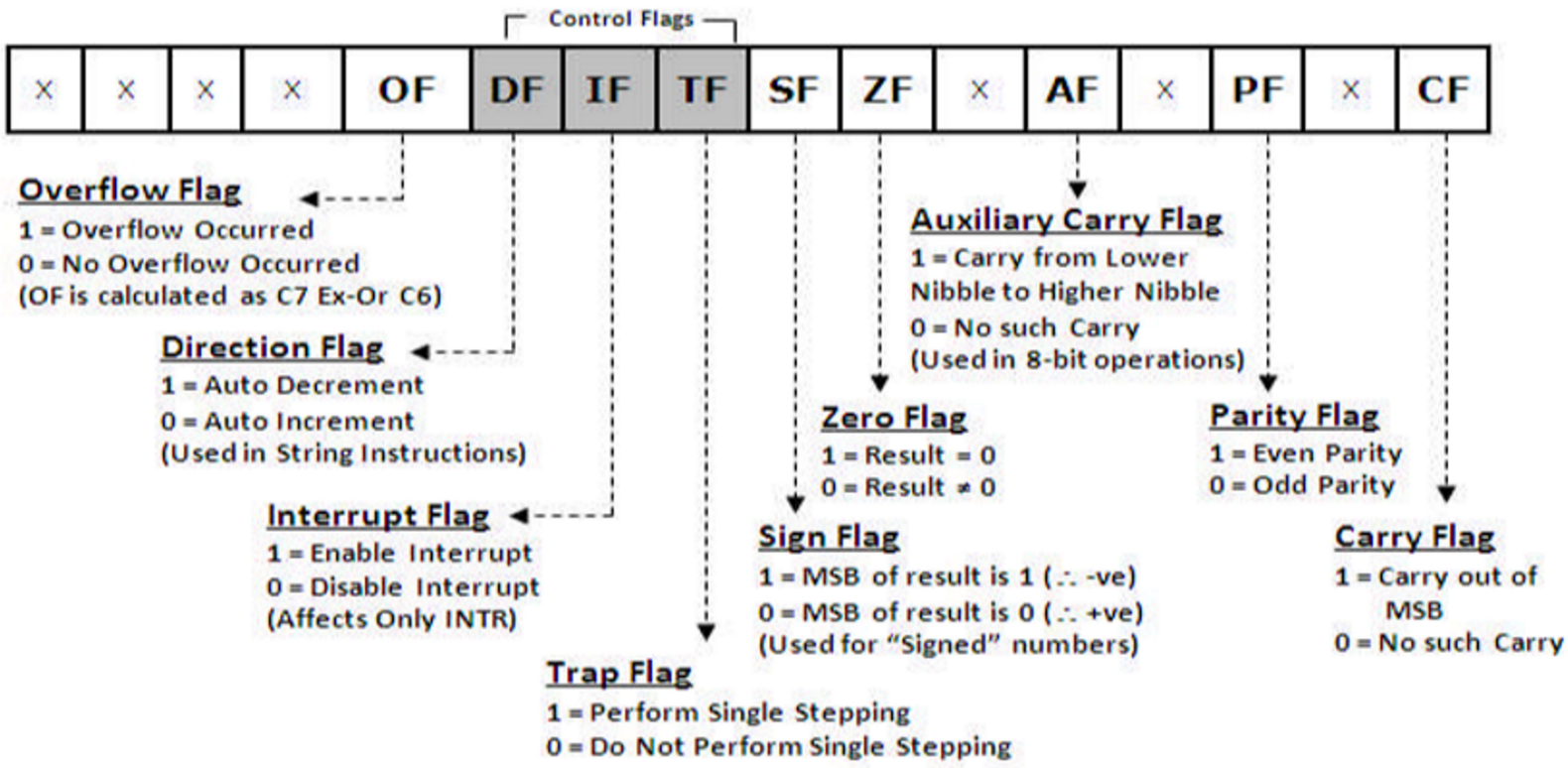
Let us now discuss the functional parts of 8086 microprocessors.

ALU

It handles all arithmetic and logical operations, like +, −, ×, /, OR, AND, NOT operations.

Flag Register

It is a 16-bit register that behaves like a flip-flop, i.e. it changes its status according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups − Conditional Flags and Control Flags.



Conditional Flags

It represents the result of the last arithmetic or logical instruction executed. Following is the list of conditional flags −

Nibble = 4 bit

- **Carry flag** − This flag indicates an overflow condition for arithmetic operations.
- **Auxiliary flag** − When an operation is performed at ALU, it results in a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), then this flag is set, i.e. carry given by D3 bit to D4 is AF flag. The processor uses this flag to perform binary to BCD conversion.
- **Parity flag** − This flag is used to indicate the parity of the result, i.e. when the lower order 8-bits of the result contains even number of 1's, then the Parity Flag is set. For odd number of 1's, the Parity Flag is reset.
- **Zero flag** − This flag is set to 1 when the result of arithmetic or logical operation is zero else it is set to 0.
- **Sign flag** − This flag holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.
- **Overflow flag** − This flag represents the result when the system capacity is exceeded.

Control Flags

Control flags controls the operations of the execution unit. Following is the list of control flags −

- **Trap flag** − It is used for single step control and allows the user to execute one instruction at a time for debugging. If it is set, then the program can be run in a single step mode.
- **Interrupt flag** − It is an interrupt enable/disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enabled condition and set to 0 for interrupt disabled condition.
- **Direction flag** − It is used in string operation. As the name suggests when it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-a-versa.

General purpose register

There are 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually to store 8-bit data and can be used in pairs to store 16bit data. The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX, CX, and DX respectively.

- AX register − It is also known as accumulator register. It is used to store operands for arithmetic operations.
- BX register − It is used as a base register. It is used to store the starting base address of the memory area within the data segment.
- CX register − It is referred to as counter. It is used in loop instruction to store the loop counter.
- DX register − This register is used to hold I/O port address for I/O instruction.

Stack pointer register

It is a 16-bit register, which holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack.

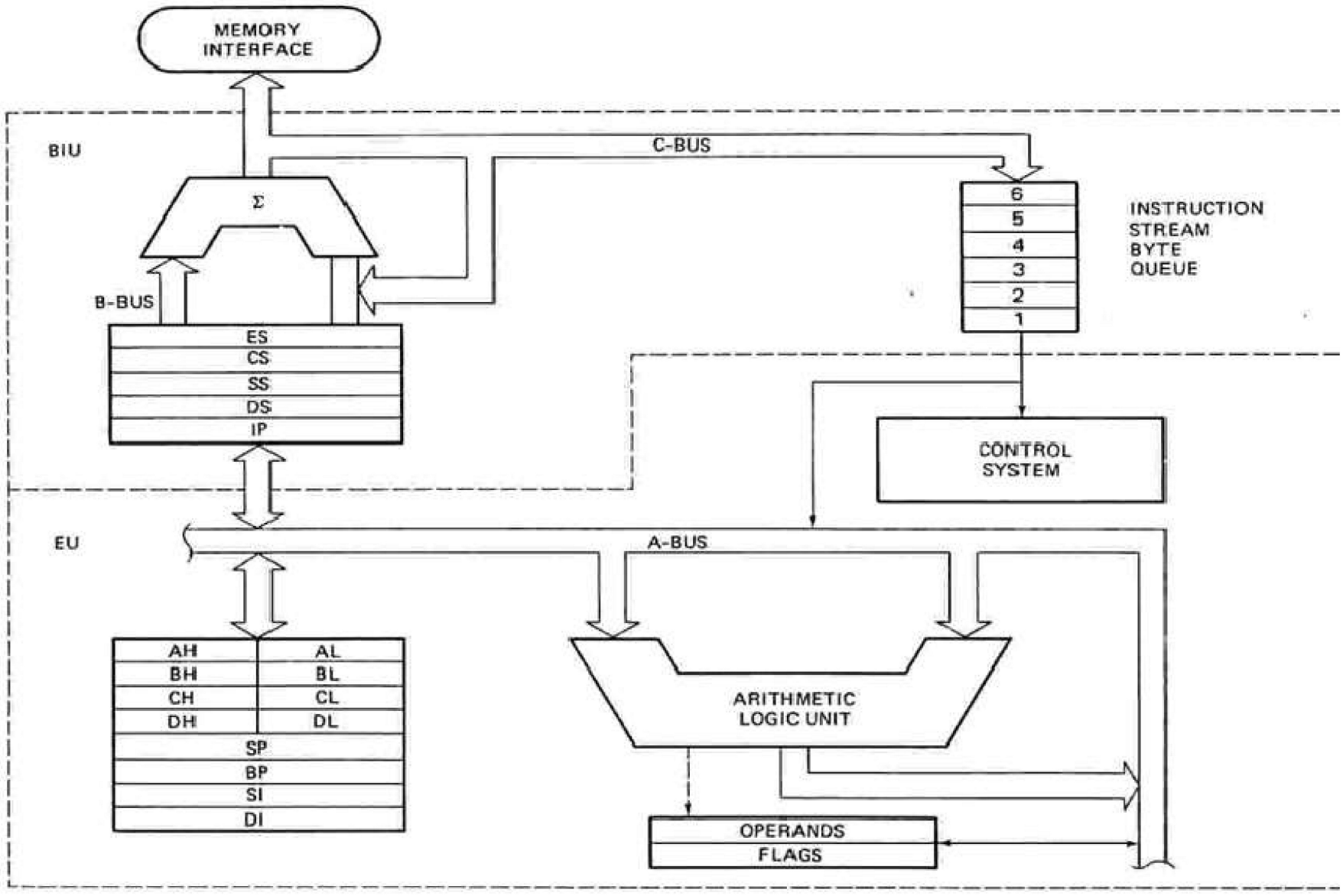
BIU (Bus Interface Unit)

BIU takes care of all data and addresses transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direct connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus.

It has the following functional parts −

- **Instruction queue** − BIU contains the instruction queue. BIU gets upto 6 bytes of next instructions and stores them in the instruction queue. When EU executes instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased execution speed.
- Fetching the next instruction while the current instruction executes is called pipelining.
- **Segment register** − BIU has 4 segment buses, i.e. CS, DS, SS& ES. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations. It also contains 1 pointer register IP, which holds the address of the next instruction to executed by the EU.
  - CS − It stands for Code Segment. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.
  - DS − It stands for Data Segment. It consists of data used by the program andis accessed in the data segment by an offset address or the content of other register that holds the offset address.
  - SS − It stands for Stack Segment. It handles memory to store data and addresses during execution.
  - ES − It stands for Extra Segment. ES is additional data segment, which is used by the string to hold the extra destination data.
- Instruction pointer − It is a 16-bit register used to hold the address of the next instruction to be executed.

Architecture of 8086 microprocessor



ARCHITECTURE OF 8086

- It is divided into 2 parts
  1) Bus interface unit (BIU)
  2) Execution unit (EU)
- **Bus Interface unit :=**
- It acts as interface between system bus and the execution unit.
- Fetches instruction from memory.
- Reads data from I/O ports and memories.
- Writes data to ports and memories
- Supports pipelining.
- BIU handles transfer of data on all the buses for the execution unit.

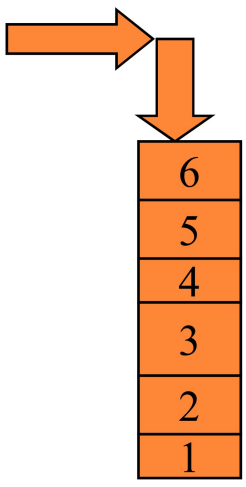
Blocks of BIU:=

1) Queue=
- To speed up the program execution BIU fetches as many as 6 instruction bytes from memory .
- These pre-fetched instruction byte for execution unit in FIFO group of registers called a QUEUE.

Concept of pipelining:=

-fetching the next instruction while recent instruction executes is known as pipelining.

-



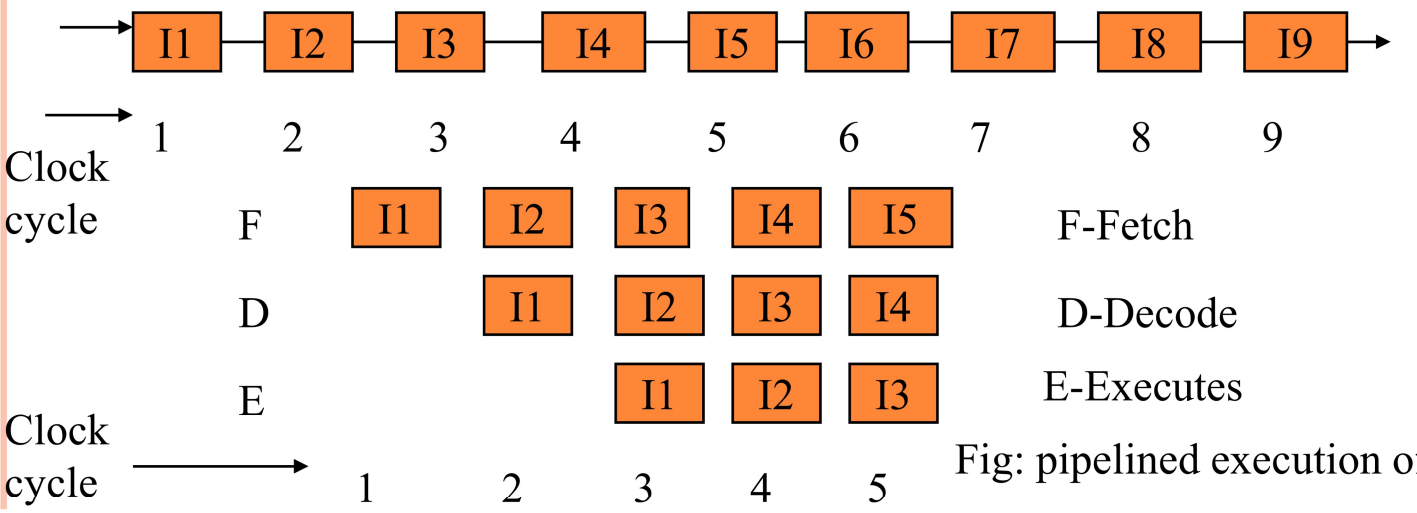


Fig: pipelined execution of 3 instruction

1. On non pipelined processor 9 clock cycle are required for individual fetch , decode & execute for 3 instruction.
2. On pipelined processor fetch, decode & execute operation are performed in parallel.
3. -only 5 cycle are required to execute 3 instruction.
4. - 1 instruction requires -> 3 cycle to completes.
5. Additional instruction complete at rate of one per cycle.
6. During clock cycle 5 I3 instruction executing , I4 is decoding , I5 instruction fetched.
7. If 1000 instruction it requires 3000 clock cycle on non pipelined processor.----> require 1002 clock cycle on pipelined processor.
8. In 8086 performs fetch , decode , & executes instruction in parallel.

2) <u>Segment Register :=</u>

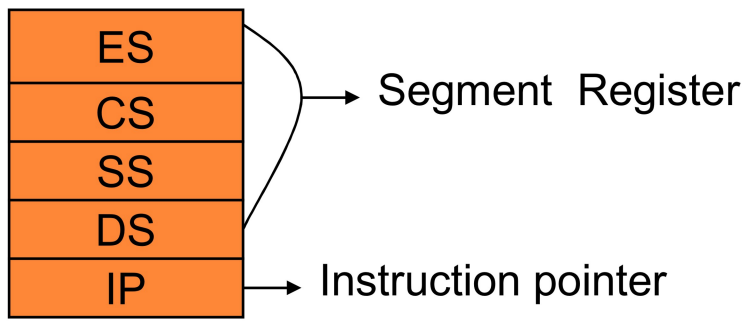
-The BIU contain 4 16 bit segment registers

- it hold upper 16 bit of starting address of 4 memory segment that 8086 is working with particular time .

- Es hold upper 16 bits of starting address of extra segment

- Cs hold upper 16 bits of starting address of code segment

- Ss & Ds hold upper 16 bit of starting address of stack segment & data segment.



Each segment contains 64 kbytes of memory.

 Code segment register (CS): is used for addressing memory location in the code segment of the memory,where the executable program is stored.

 Data segment register (DS): points to the data segment of the memory where the data is stored.

 Extra Segment Register (ES) : also refers to a segment in the memory which is another data segment in the memory.

 Stack Segment Register (SS): is used for addressing stack segment of the memory. The stack segment is that segment of memory which is used to store stack data.

While addressing any location in the memory bank, the **physical address is calculated from two parts:**

Physical address= segment address 0 + offset address

 The first is segment address, the segment registers contain 16-bit segment base addresses, related to different
segment.

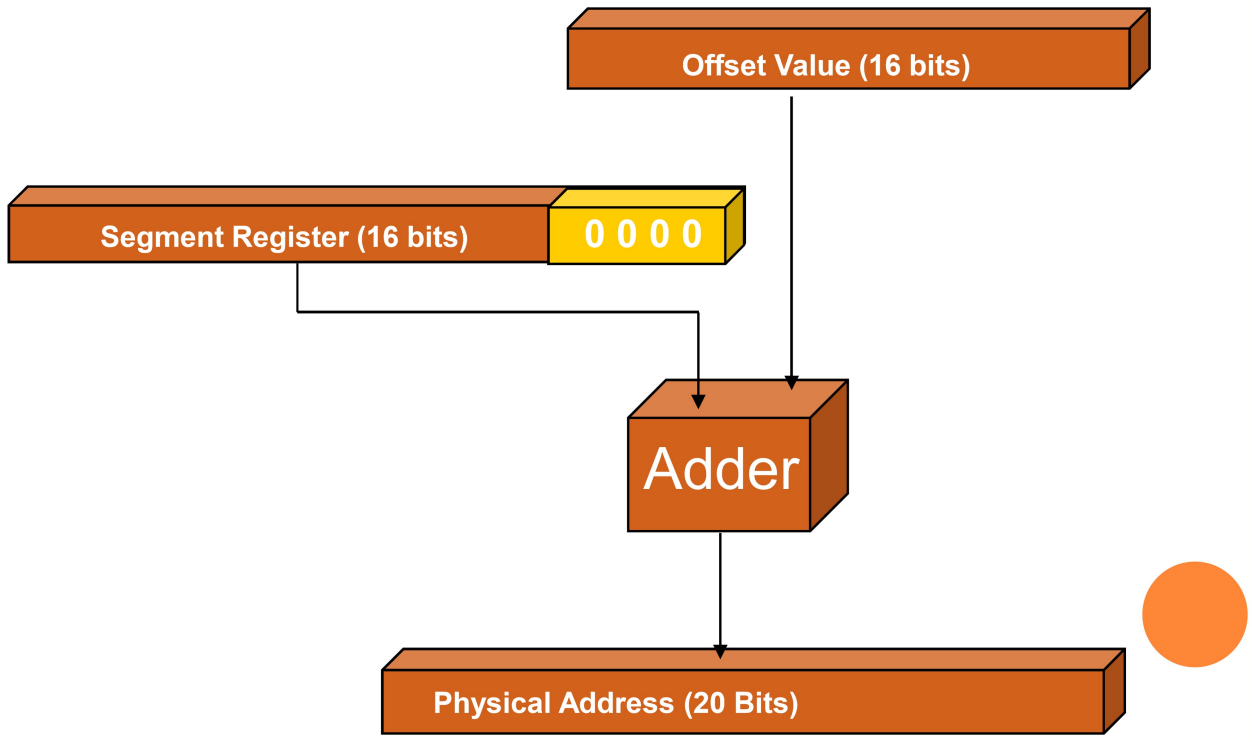
 The second part is the offset value in that segment.It is present in the pointer and index registers..

<u>3) Instruction Pointer Register (IP) :=</u>

-It stores the memory location of next instruction to be executed. The pointer register *IP contains offset address of the code segment.*

Memory Address Generation

- The BIU has a dedicated adder for determining physical memory addresses

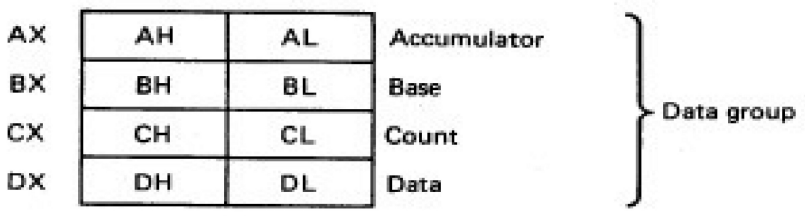


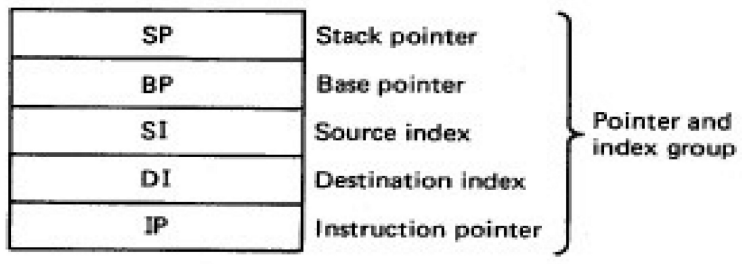
FUNCTIONAL BLOCK OF EU

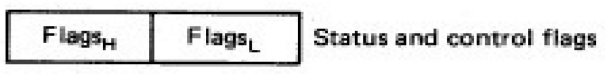
1) Control circuit ,instruction decoder , ALU

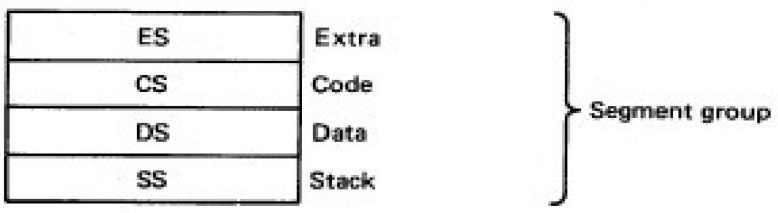
- The execution unit contain control circuit that direct all internal operation.
- The execution unit has 16 bit ALU block that perform arithmetic & logical operation like +, - , *,AND , OR ,NOT
- The instruction decoder in EU that translates the instruction fetch from memory into a series of action that EU carried out.

8086/88 internal registers 16 bits (2 bytes each)









AX, BX, CX and DX are two bytes wide and each byte can be accessed separately

These registers are used as memory *pointers*.

Flags will be discussed later

Segment registers are used as base address for a segment in the 1 M byte of memory

-The execution unit has 8 general purpose register labeled as AH,AL,BH,BL,CH,CL,DH,DL

- These register can be used for temporary storage of 8 bit data & 16 bit data
- AH-AL pair is referred as AX , BH- BL pair is referred as BX , CH-CL pair is referred as CX , DH-DL pair is referred as DX.

1) Stack pointer register:

- It contain a 16 bit offset added to stack segment register indicates address of memory location when word was most recently stored.

2) Base pointer register:

- The base pointer register can be used instead of stack pointer register to access memory location within the stack
- The 20 bit address can be obtained by shifting the contents of the stack segment by 4 bits & adding the contents of BP to it

3) Source index register:

-The source index register is used to load 16 bit offset of a data word in data segment .

-The physical address of the data word can be obtained by shifting the contents of DS register by 4 bits & adding into SI

4) Destination index register

-It is used to hold the 16 bit offset of a data word in extra segment while executing string instruction

-The 20 bit physical address is calculated from DS & DI