SNS COLLEGE OF ENGINEERING

(Autonomous)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

19EC401 – MICROPROCESSORS AND MICROCONTROLLERS

8086 Architecture

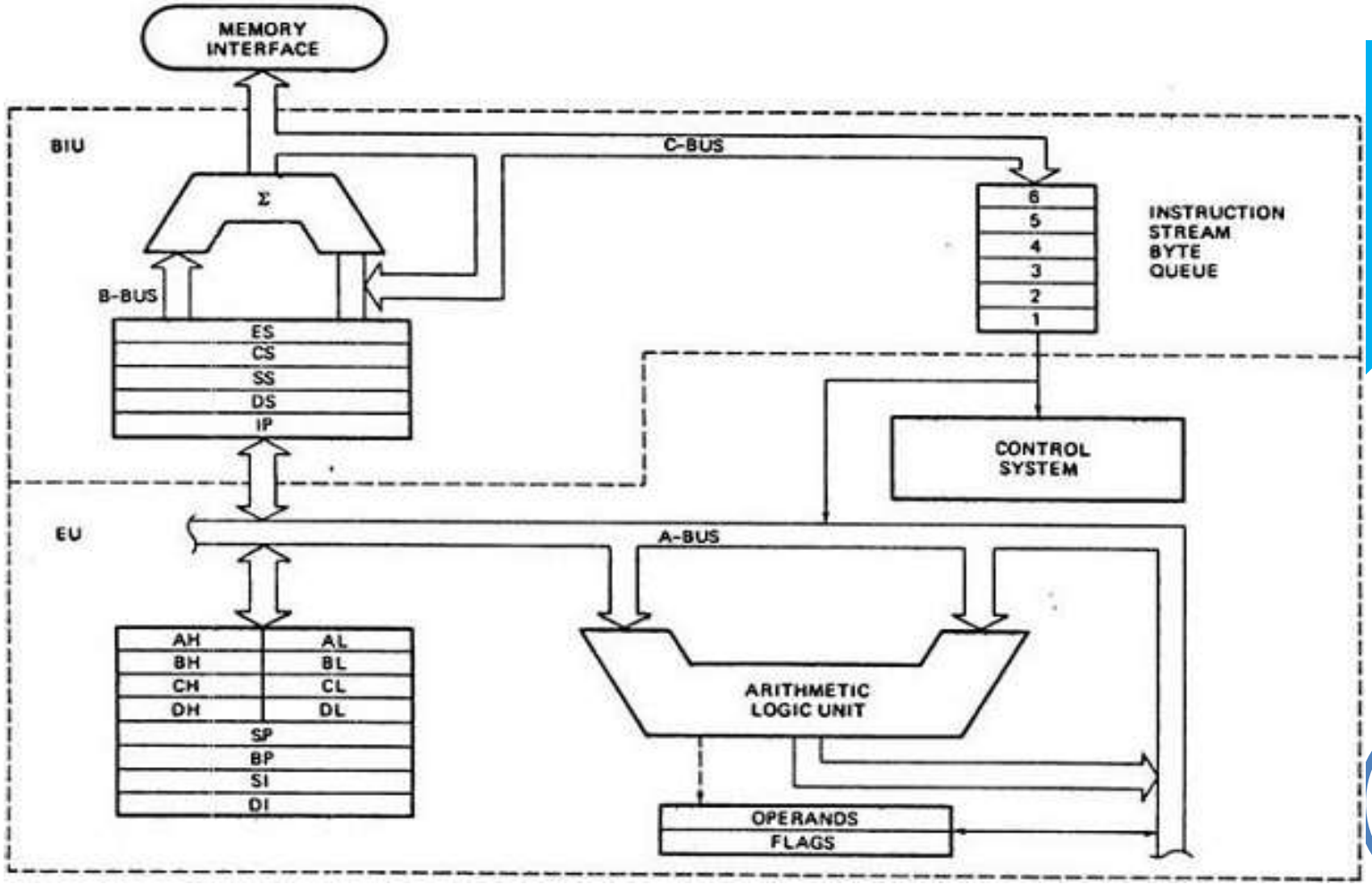
Features of 8086

- It consists of two main section
- Bus Interface Unit (BIU)
- Execution unit (EU)
- Two units are independent of each other
- EU contains the ALU, flags, and general purpose registers
- All registers are 16-bits wide
- The BIU controls the address, data and control buses

Architecture of 8086

The 8086 CPU is divided into two independent functional units:

1. Bus Interface Unit (BIU)
2. Execution Unit (EU)


MEMORY INTERFACE
BIU
C-BUS
Σ
B-BUS
6
5
4
3
2
1
INSTRUCTION STREAM BYTE QUEUE
ES
CS
SS
DS
IP
CONTROL SYSTEM
EU
A-BUS
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
SI
DI
ARITHMETIC LOGIC UNIT
OPERANDS
FLAGS

BIU Operation

- The function of BIU is to:
  - Fetch the instruction or data from memory.
  - Write the data to memory.
  - Write the data to the port.
  - Read data from the port.

BIU consists of the following

•Queue

•Segment Register

•Instruction pointer and address summation

Queue

•FIFO registers arranged like pipe called queue

•The BIU continuously fetch operations from the memory while the processor is executing current instruction

•BIU units stores the fetched bytes in the queue and the EU will read these bytes from the queue

Segment register

•1 MB memory is divided into segment

•Microprocessor 8086 can at a time access four segments

•Each segment is independent and up to 64K bytes long

•8086 consists of 4 segment registers

Code Segment Register

•Stores base address of 64 KB segment and microprocessor instructions / programs

•The instruction pointer is the by default registers used by the microprocessor to access the instructions from the CS

Stack Segment Register

•Contains the off set address of 64 KB segment

•Used as stack memory, operates on LIFO

•Performs Stack operations (PUSH and POP)

Data Segment Register

•Holds the logical address of 64 KB long data segment

•By default registers of this segment are

•AX

•BX

•CX

•DX

•Indexed register (SI and DI)

Extra Segment Register

•Contains the starting address of 64 KB segment

•This segment is the destination for the string data pointed by DI register (location pointed by DI register is stored here)

•Can not initialize ES register but can be changed using POP instruction

Instruction Pointer and Address Summation

•It contains the offset / logical address of the next byte to be read from the code segment

•Generation of 20 bit physical address

Execution Unit

•Consist of four 16-bit general purpose register AX, BX, CX, DX Which is divided among two parts (higher and lower parts 8 bits each)

•Table

•AX register used as 16 bit accumulator in 16-bit operations whereas AL is used for 8 bit operation

•BX used as memory pointer in DS. Used for based, based indexed or register indirect addressing mode

CX (count register)

•used as counter in string manipulation. Default counter in loop instruction

•Data register DX

•consists of 2 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX. When combined, DL register contains the low-order byte of the word, and DH contains the high-order byte

EU (Execution Unit)

- The functions of execution unit are:
- □ To tell BIU where to fetch the instructions or data from.
- □ To decode the instructions.
- □ To execute the instructions.
- The EU contains the control circuitry to perform various internal operations. A decoder in EU
- decodes the instruction fetched memory to generate different internal or external control signals
- required to perform the operation. EU has 16-bit ALU, which can perform arithmetic and logical
- operations on 8-bit as well as 16-bit.

EU (Execution Unit)

- THE EU CONTAINS THE FOLLOWING 8-BIT REGISTERS:

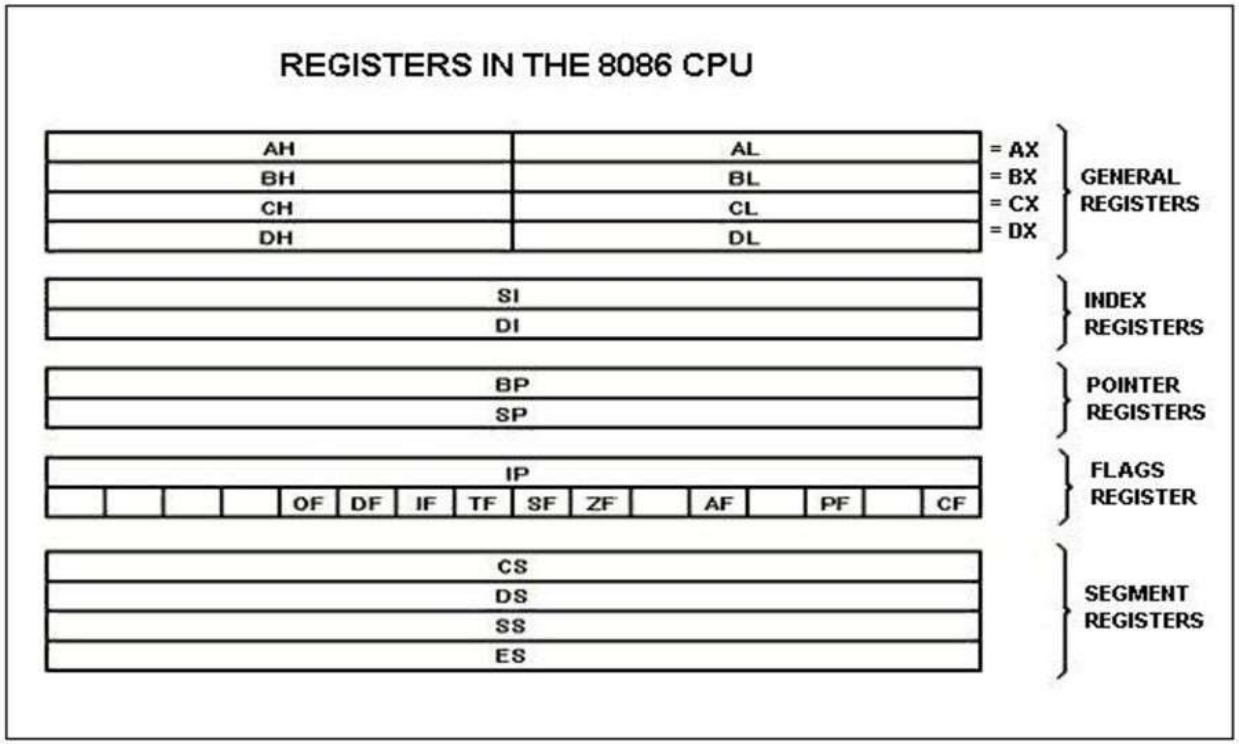
- AH & AL (AX-16 BIT)
- BH & BL (BX-16 BIT)
- CH & CL (CX-16 BIT)
- DH & DL (DX-16 BIT)

EU (Execution Unit)

- IT ALSO INCLUDES THE FOLLOWING 16-BIT REGISTERS:

- STACK POINTER (SP)
- BASE POINTER (BP)
- SOURCE INDEX (SI)
- DESTINATION INDEX (DI)

Register of 8086



GENERAL PURPOSE REGISTERS

- 8086 CPU has 8 general purpose registers

•**AX** - the accumulator register
(divided into **AH / AL**):

•Generates shortest machine code
•Arithmetic, logic and data transfer
•One number must be in AL or AX
•Multiplication & Division
•Input & Output

•**BX** - the base address register (divided into **BH / BL**).

GENERAL PURPOSE REGISTERS

- **CX** - the count register
  (divided into **CH / CL**):
- Iterative code segments using the LOOP instruction
- Repetitive operations on strings with the REP command
- Count (in CL) of bits to shift and rotate

- **DX** - the data register (divided into **DH / DL**):

- DX:AX concatenated into 32-bit register for some MUL and DIV operations

- Specifying ports in some IN and OUT operations

GENERAL PURPOSE REGISTERS

- **SI** - source index register:
- Can be used for pointer addressing of data
- Used as source in some string processing instructions
- Offset address relative to DS

- **DI** - destination index register:

- Can be used for pointer addressing of data
- Used as destination in some string processing instructions
- Offset address relative to ES

GENERAL PURPOSE REGISTERS

- **BP** - base pointer:
- Primarily used to access parameters passed via the stack
- Offset address relative to SS

- **SP** - stack pointer:
- Always points to top item on the stack
- Offset address relative to SS
- Always points to word (byte at even address)
- An empty stack will had SP = FFFEh

Segment Register

- **CS** (Code segment) - points at the segment containing the current program.
- **DS** (Data Segment) - generally points at segment where variables are defined.
- **ES** (Extra Segment)- extra segment register, it's up to a coder to define its usage.
- **SS** (Stack Segment) - points at the segment containing the stack.

- Although it is possible to store any data in the segment registers, this is never a good idea. The segment registers have a very special purpose - pointing at accessible blocks of memory.

- Segment registers work together with general purpose register to access any memory value.
- For example if we would like to access memory at the physical address **12345h**(hexadecimal),
- we could set the **DS = 1230h** and **SI = 0045h**. This way we can access much more memory than with a single register, which is limited to 16 bit values.
  The CPU makes a calculation of the physical address by multiplying the segment register by 10h and adding the general purpose register to it (1230h * 10h + 45h = 12345h):

  The address formed with 2 registers is called an **effective address**.
  By default **BX, SI** and **DI** registers work with **DS** segment register;
  BP and **SP** work with **SS** segment register.
  Other general purpose registers cannot form an effective address.
  Also, although **BX** can form an effective address, **BH** and **BL** cannot.
-

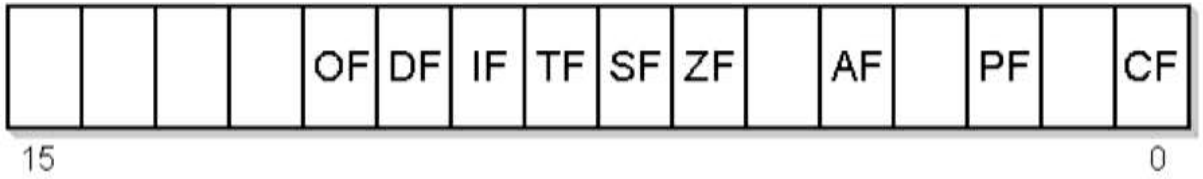
Special Purpose Register

- **IP** - the instruction pointer:
- Always points to next instruction to be executed
- Offset address relative to CS
- **IP** register always works together with **CS** segment register and it points to currently executing instruction.

Flag Register

- **Flags Register** - determines the current state of the processor. They are modified automatically by CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program. Generally you cannot access these registers directly.

Flag register in EU is of 16-bit



8086 has 9 flags and they are divided into two categories:

1. Conditional Flags

2. Control Flags

- **Conditional Flags**

- Conditional flags represent result of last arithmetic or logical instruction executed. Conditional flags are as follows:
- **Carry Flag (CF):** This flag indicates an overflow condition for unsigned integer arithmetic.
  It is also used in multiple-precision arithmetic.

- **Auxiliary Flag (AF):** 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.

- **Parity Flag (PF):** 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.

- **Zero Flag (ZF):** It is set; if the result of arithmetic or logical operation is zero else it is reset.

- **Sign Flag (SF):** In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.

- **Overflow Flag (OF):** It occurs when signed numbers are added or subtracted. An OF indicates that the result has exceeded the capacity of machine

- **Control Flags**
- Control flags are set or reset deliberately to control the operations of the execution unit.

Control flags are as follows:

- **Trap Flag (TP):**

a. It is used for single step control.

b. It allows user to execute one instruction of a program at a time for debugging.

c. When trap flag is set, program can be run in single step mode.

- **Interrupt Flag (IF):**

a. It is an interrupt enable/disable flag.

b. If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled.

c. It can be set by executing instruction sit and can be cleared by executing CLI instruction.

- **Direction Flag (DF):**

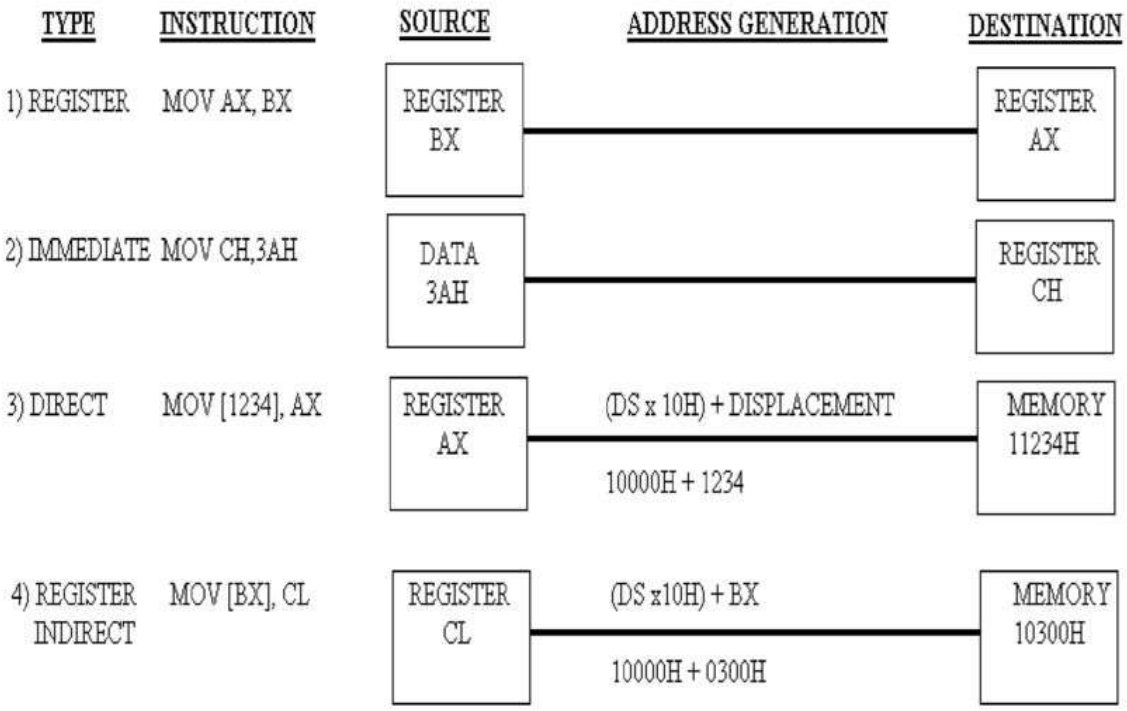
a. It is used in string operation.

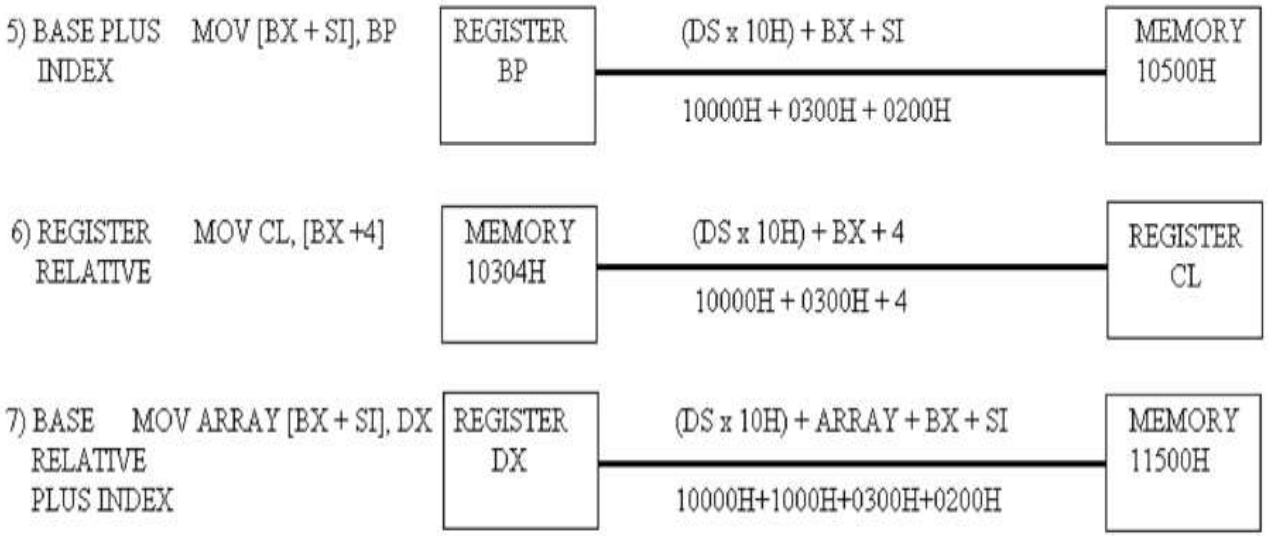
b. If it is set, string bytes are accessed from higher memory address to lower memory address.

c. When it is reset, the string bytes are accessed from lower memory address to higher memory address.

Addressing Modes of 8086

8086 ADDRESS MODES





ASSUME: BX = 0300H; SI = 0200H; ARRAY = 1000H; DS = 1000H.

1.The intel 8086 microprocessor is --------a processor
2. In 8086 microprocessor , the address bus is -------bit wide
3. The BIU prefetches the instruction from memory and store them in--------

THANK YOU