COMPUTER ARCHITECTURE 1

RAJA'A M. MOHAMMED

SECOND STAGE

1. Introduction to microprocessors and microcomputers

1.1 *Introduction to microcomputers*

Computer is a machine that can be used to **solve problems** for people by carrying out the task by following the instructions given to it. A sequence of instructions describing how to perform a certain task or job is called a **program**.

Microcomputer system just as any computer system, include two principal components hardware and software.

The architecture of a computer is the general layout of its major component, the principal features of these components and how they are connected together explain bellow.

These components are:

a- Central processing unit (CPU)

CPU is a microprocessor and is often referred to as the MicroProcessor Unit (MPU), its purpose is to decode the instructions and use them to control the activity within the system, it's also performs all arithmetic and logical computations.

b- Timing circuitry (clock)

Is needed to synchronize the activity within the microprocessor and the bus control logic.

c- Memory (main memory)

Is used to store both the data and instruction, that are currently being used, its normally broken into several modules containing several thousand location, each location has owe memory address, this location may contain data or instruction, and the CPU dose its work by successively inputting or fetching instruction from memory and carrying out the tasks dictated by them.

d- Input /Output subsystem

This may consists of a variety of devices for communicating with the external world and for storing large quantities of information.

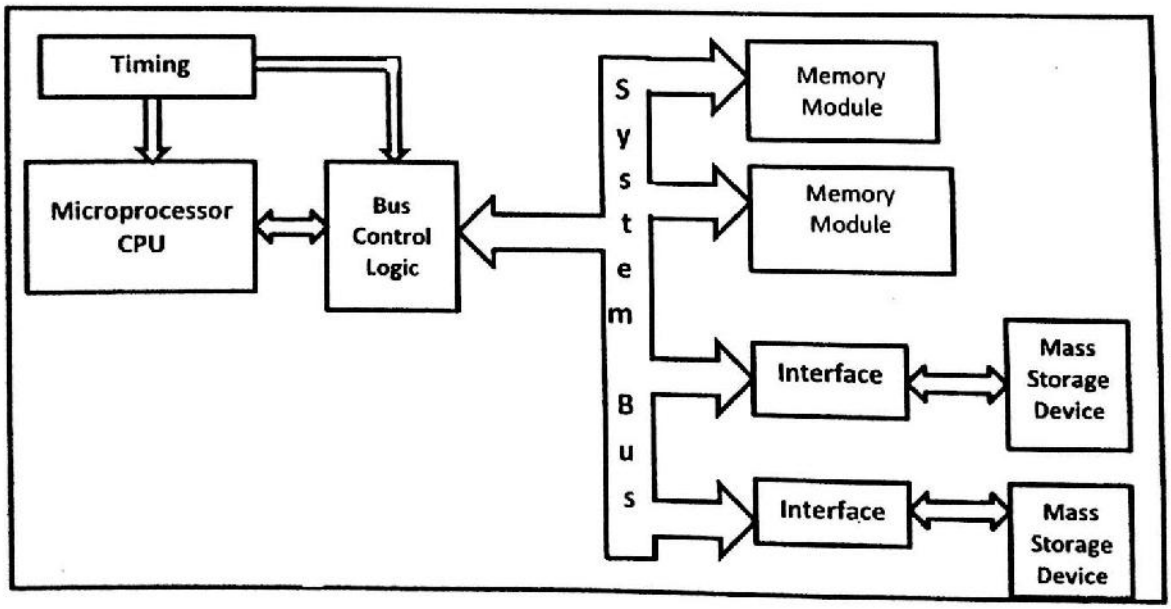


Fig (1):Architecture of a Typical Microprocessor

e- Bus control logic

Is the interface between the CPU and the others devices likes memory, I/O devices.

f- System bus

Is a set of conductors that connect the CPU to its memory and I/O devices, the bus conductors are separated in to three groups:.

1- The data line for transmitting the information (bi-directional)
2- The address line ,which indicate where the information is to come from or is to be stored (unidirectional)
3- The control lines, which regulate the activity on the bus

1.2 *Introduction to microprocessors*

Microprocessor is also called Central Processing Unit (CPU) it is used to process data.

The CPU includes:

1- Control unit and possibly a control memory.
Is used for decoding and carrying out the instructions
2- A set of working registers
For helping with the addressing and computational tasks
3- Arithmetic \logic unit (ALU)
For executing the arithmetic and logical operations
4- Bus control section (unit)
For helping communication with the bus

flags are used to indicate the current state of the CPU and the important characteristics of the result for the previous instruction.

SP: most microcomputers utilize something called a stack. This stack normally consists of a portion of the main memory; it is used to store some important information while subroutine or service routines are being executed. The stack is LIFO.

1.3 Types of Microprocessors

Microprocessors generally is categorized in terms of the maximum number of binary bits in the data they process – that I, their word length. Over time, five standard data widths have evolved for microprocessors: 4-bit, 8-bit, 16-bit, 32-bit, 64-bit. There are so many manufacturers of Microprocessors, but only two companies have been produces popular microprocessors: *Intel* and *Motorola*. Table 1 lists some of types that belong to these companies (families) of microprocessors

Type	Data bus width	Memory size
Intel family		
8085	8	64K
8086	16	1M
80286	16	16M
80386EX , 80386DX	16 , 32	64M , 4G
80486DX4	32	4G + 16K cache
Pentium	64	4G + 16K cache
PentiumIII , Pentium4	64	64G+32K L1 cache +256 L2 cache
Motorola family:		
6800	8	64K
68060	64	4G + 16K cache

Table (0): types of processors

It is important to note that 80286, 80386, 80486, and Pentium-Pentium4 microprocessors are upward compatible with the 8086 Architecture. This mean that 8086/8088 code will run on the 80286, 80386, 80486, and Pentium Processors, but the reverse in not true if any of the new instructions are in use. Beside to the ***general-purpose*** microprocessors,

these families involve another type called ***special-purpose*** microprocessors that used in embedded control applications. This type of embedded microprocessors is called *microcontroller*. The 8080, 8051, 8048, 80186, 80C186XL are some examples of microcontroller.

2- Intel 8085 CPU

is a 8 bit processor with 6 general purpose registers(defined by the letters B,C,D,E,L,H) which are 8-bit each and are associated in pairs.

One 8-bit accumulator (denoted by A), 16-bit stack pointer, 16-bit program counter and a PSW with five flags.

- ✓ The main features of 8085 μp are:
- ✓ It is a 8 bit microprocessor.
- ✓ It is manufactured with N-MOS technology.
- ✓ It has 16-bit address bus and hence can address up to 2^{16} = 65536 bytes (64KB) memory locations through A_0-A_{15}
- ✓ The first 8 lines of address bus and 8 lines of data bus are multiplexed $AD_0 - AD_7$.
- ✓ Data bus is a group of 8 lines $D_0 - D_7$.
- ✓ It supports external interrupt request.
- ✓ A 16 bit program counter (PC)
- ✓ A 16 bit stack pointer (SP)
- ✓ Six 8-bit general purpose register arranged in pairs: BC, DE, HL.
- ✓ It requires a signal +5V power supply and operates at 3.2 MHZ single phase clock.
- ✓ It is enclosed with 40 pins DIP (Dual in line package).

2.1 Registers

The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H, and L .They can be combined as register pairs- BC, DE, and HL - to perform some 16-bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions.

Individual	B, C, D, E, H, L
Combination	B&C D&E H&L

General purpose registers

Accumulator

The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations.

The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flags

The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero(Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags; their bit Positions in the flag register. The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions.

S	Z		AC		P		CY

Flag register

Carry flag - it indicates whether there is carry or not after an arithmetic and logical operation.

Zero flag - it indicates whether the result of an arithmetic or logical operation is zero or nonzero.

Sign flag - it indicates whether the result is positive or negative

Parity flag- it indicates whether the result contains odd number of **1s** or even number of **1s**.

Auxiliary carry -that is carry from the 3rd bit to 4th .

2.2 Programming Model of the Intel 8085 Microprocessor

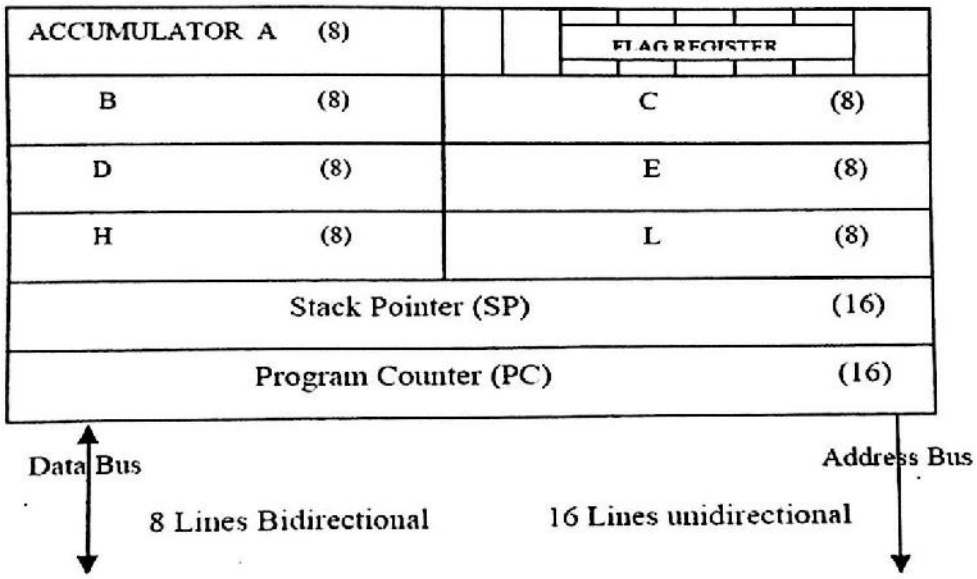


Fig (3): Intel 8085 Microprocessor

Addressing Modes of 8085

To perform any operation, we have to give the corresponding instructions to the microprocessor. In each instruction, programmer has to specify 3 things:

- Operation to be performed.

- Address of source of data.
- Address of destination of result.

The term addressing mode refers to the way in which the operand of the instruction is specified.

Intel 8085 uses the following addressing modes :

1. Direct Addressing Mode
2. Register Addressing Mode
3. Register Indirect Addressing Mode
4. Immediate Addressing Mode
5. Implicit Addressing Mode

1. Direct Addressing Mode

In this mode, the address of the operand is given in the instruction itself.

LDA 2500 H Load the contents of memory location 2500 H in accumulator.

2. Register Addressing Mode

In this mode, the operand is in general purpose register.

MOV A, B Move the contents of register B to A.

3. Register Indirect Addressing Mode

In this mode, the address of operand is specified by a register pair.

MOV A, M Move data from memory location specified by H-L pair to accumulator.

4. Immediate Addressing Mode

In this mode, the operand is specified within the instruction itself.

MVI A, 05 H 1/ , Move 05 H in accumulator.

5. Implicit Addressing Mode

If address of source of data as well as address of destination of result is fixed then there is no need to give any operand along with the instruction.

CMA Complement accumulator.

Instruction Set Classification

An instruction is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions, called the instruction set, determines what functions the microprocessor can perform. These instructions can be classified into the following five functional categories: data transfer (copy) operations, arithmetic operations, logical operations, branching operations, and machine-control operations.

3- The Intel 8086 Microprocessor

This microprocessor is a high performance 16 bits CPU is available in three clock rates 5,8 and 10 MHZ, in 40 pin .

8086 is 16 –bits data bus, it can address 1million bytes of memory, 20bits address.

3.1 8086 Architecture

The internal architecture 8086 microprocessor is as shown in the fig () .The 8086 CPU is divided into two independent functional parts, the Bus interface unit (BIU) and execution unit (EU).

The BIU contains the segment registers, the instruction pointer, the address generation adder, bus control logic, and an instruction queue. The execution unit contains the Data and Address registers, the Arithmetic and Logic Unit, the Control Unit and flags.

The BIU is responsible for performing all external bus operations, such as instruction fetching, reading and writing of data operands for memory, and inputting or outputting data for input/output peripherals. These information transfers take place over the system bus. The BIU is not only responsible for performing bus operations; it also performs other functions related to instruction and data obtained. For instance, it is responsible for instruction queuing and address generation. The BIU uses a mechanism known as an instruction queue to implement a pipelined architecture.

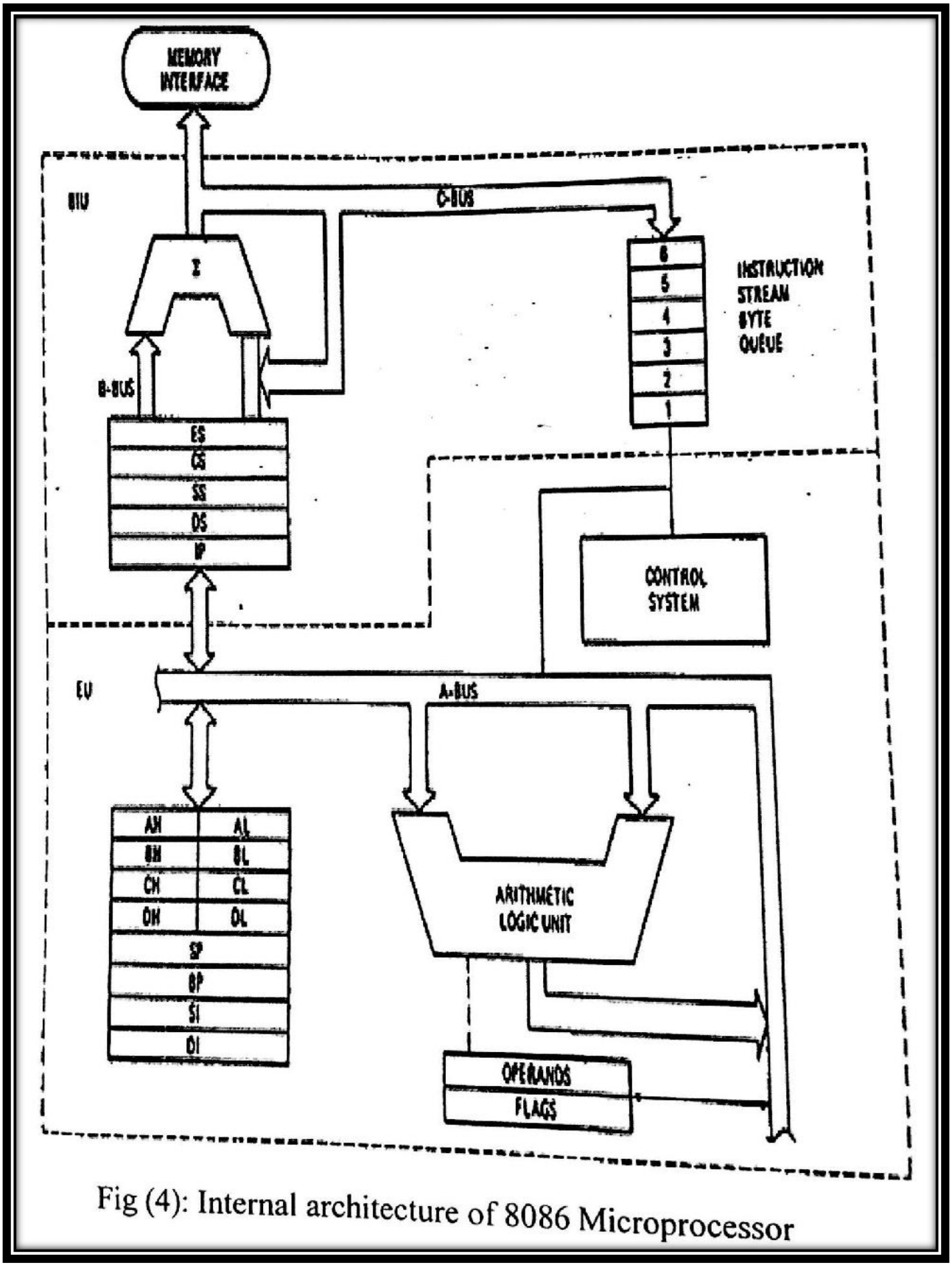


Fig (4): Internal architecture of 8086 Microprocessor

The EU: The execution unit is responsible for decoding and executing instructions. The EU accesses instructions from the output end of the instruction queue and data from the general-purpose registers or memory. During execution of the instruction, the EU may test the status and control flags, and updates these flags based on the results of executing the instruction.

The Queue: The BIU fetches up to 6 instruction bytes for the following instructions. The BIU stores these pre-fetched bytes in first-in-first-out register set called a queue. When the EU is ready for its next instruction it simply reads the instruction byte(s) for the instruction from the queue in the BIU. This is much faster than sending out an address to the system memory and waiting for memory to send back the next instruction byte or bytes. Except in the case of JMP and CALL instructions, where the queue must be dumped and then reloaded starting from a new address, this pre-fetch and queue scheme greatly speeds up processing. Fetching the next instruction while the current instruction executes is called pipelining

Word Read

Each of 1 MB memory address of 8086 represents a byte wide location.16-bit words will be stored in two consecutive memory locations. If first byte of the data is stored at an even address, 8086 can read the entire word in one operation.

For example if the 16 bit data is stored at even address 00520H is 9634H

MOV BX, [00520H]

8086 reads the first byte and stores the data in BL and reads the 2nd byte and stores the data in BH

BL= (00520H) i.e. BL=34H

BH= (00521H) BH=96H

If the first byte of the data is stored at an odd address, 8086 needs two operations to read the 16 bit data.

For example if the 16 bit data is stored at even address 00521H is 3897H

MOV BX, [00521H]

In first operation, 8086 reads the 16 bit data from the 00520H location and stores the data of 00521H location in register BL and discards the data of 00520H location In 2nd operation, 8086 reads the 16 bit data from the 00522H location and stores the data of 00522H location in register BH and discards the data of 00523H location.

BL= (00521H) i.e. BL=97H

BH= (00522H) BH=38H

Byte Read:

MOV BH, [Addr]

For Even Address:

Ex: MOV BH, [00520H]

8086 reads the first byte from 00520 location and stores the data in BH and reads the 2nd byte from the 00521H location and ignores it

BH =[00520H]

For Odd Address

MOV BH, [Addr]

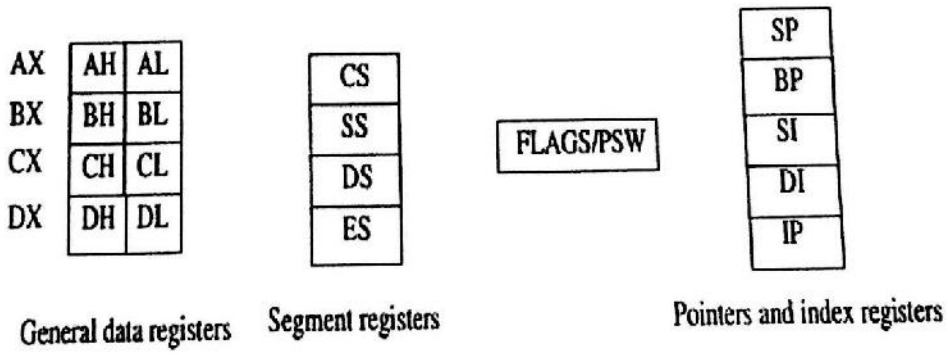
Ex: MOV BH, [00521H]

8086 reads the first byte from 00520H location and ignores it and reads the 2nd byte from the 00521 location and stores the data in BH

BH = [00521H]

3.2 Register organization of 8086:

8086 has a powerful set of registers containing general purpose and special purpose registers. All the registers of 8086 are 16-bit registers. The general purpose registers, can be used either 8-bit registers or 16-bit registers. The general purpose registers are either used for holding the data, variables and intermediate results temporarily. The special purpose registers are used as segment registers, pointers, index registers or as offset storage registers for particular addressing modes. We will categorize the register set into four groups as follows:



Fig(5): Register Organization Of 8086

1- Segment registers:

During each memory cycle ,the processoe assert 20-bits of address on the address bus, this result in arange of addresses from 0 to fffffh, all registers that contain addresses are 16-bits , the direct address mode specifies a

16-bits number ,so the absoulute (final or **physical**) address must be calculated in somway.

The 8086 memory is divided into 16 segments, each of 64 KB, so the 16bits address is used as an offset or internal segment address, also these 16-bits sometimes reffered to as the **logical** address , the logical address is added to the contents of one of four segment registers.

Each segment register contain 16-bits, the contents of segment are shifted left 4 bits position ,the result is a 20-bits number which is the **physical** memory addesss

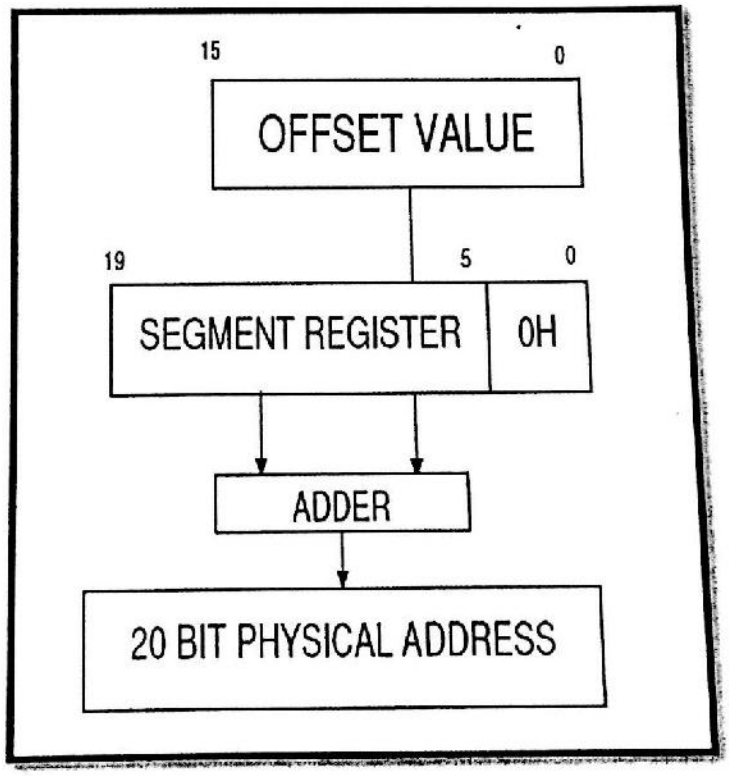


Fig (6):Physical address formation

Where the **CS** is used to extent the address of the IP (also called PC)

DS is used to extent the address when making reference to data in memory

SS is used to extent the address of the SP
ES is used especially with string operations

2- Data register

The registers AX, BX, CX, and DX are the general 16-bit registers.

- **AX Register:** Accumulator register consists of two 8-bit registers AL and AH, which can be combined together and used as a 16- bit register AX. AL in this case contains the low-order byte of the word, and AH contains the high-order byte. Accumulator can be used for I/O operations, rotate and string manipulation.
- **BX Register:** This register is mainly used as a base register. It holds the starting base location of a memory region within a data segment. It is used as offset storage for forming physical address in case of certain addressing mode.
- **CX Register:** It is used as default counter or count register in case of string and loop instructions.
- **DX Register:** provides I/O addresses for I/O instructions when required.

3- Pointers and index registers.

The pointers contain within the particular segments. The pointers IP, BP, SP usually contain offsets within the code, data and stack segments respectively

- **Stack Pointer** (SP) is a 16-bit register pointing to program stack in stack segment.
- **Base Pointer** (BP) is a 16-bit register pointing to data in stack segment.
- **Source Index** (SI) is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data addresses in string manipulation instructions.
- **Destination Index** (DI) is a 16-bit register. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions

4- Control register

this grouping refers to IP and the flags .the IP points to the next instruction to be feched, the flags are the condition code ,the flags are shown as 16-bits:

- ✓ SF - Sign Flag - The result is negative
- ✓ ZF - Zero Flag - The result is zero
- ✓ AF - Auxiliary Carry Flag - A BCD carry occurred
- ✓ PF - Parity Flag - Indicates the oddness or evenness of the number of bits
- ✓ CF - Carry Flag - An unsigned carry occurred
- ✓ OF - Overflow Flag - A signed overflow/carry occurred
- ✓ DF - Direction Flag - Controls the direction of repeated string operations
- ✓ IF - Interrupt Flag - Enables or disables interrupts

- ✓ TF - Trace Flag - Controls the debug single step interrupt

4-Addressing modes

The addressing modes is how the data or operand can appear in instruction, there are several data addressing modes:

1. Register operand addressing
   With the register addressing mode, the operand to be accessed is specified as residing in an internal register.
   Ex. MOV AX, BX
2. Immediate operand addressing mode
   The operand is part of the instruction instead of the contents of a register
   Ex. MOV AL, 15H
3. To reference an operand in memory , must be calculate the physical address of the operand and then initiate a read or write operation of this storage location, This mode includes five types:
   - ✓ Direct addressing: the value of the effective address is encoded directly in the instruction

     Ex. MOV CX, [1234H]

- ✓ Register indirect addressing: this mode is similar to the direct addressing but the offset is specified in a base register (BX), base pointer (BP) or an index register (SI or DI) within the 8086

Ex. MOV AX, [SI]

- ✓ Based addressing: this mode, the effective address is obtained by adding a direct or indirect displacement to the contents of either base register BX of Base pointer register BP

Ex. MOV [BX]+1234H, AL

- ✓ Indexed addressing: this mode, work in similar manner to that of the based addressing mode but the effective address is obtained by adding the displacement to the value in an index register (SI or DI).

Ex. MOV AL, [SI]+1234H

- ✓ Based-Indexed addressing: this mode combines the based addressing mode and indexed addressing mode. Fig 8belowshows the memory and registers before and after the execution of instruction

Ex. MOV AH, [BX][SI]+1234H

5- 8086 instruction set

1. Data transfer instructions
2. Arithmetic instructions
3. Logic instructions
4. Shift and Rotate instructions
5. Jump Instructions
6. Subroutines instructions
7. PUSH and POP instruction

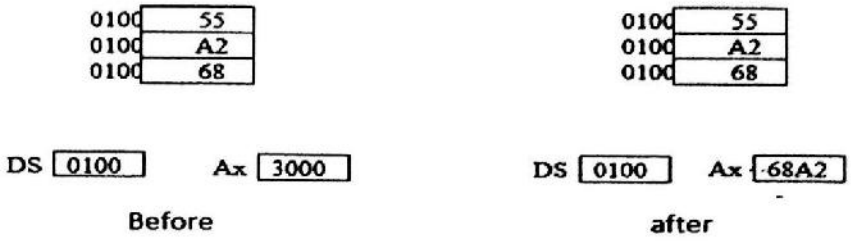
1. Data transfer instructions
   1) MOV
   2) XCHG

Instruction	Meaning	Format	Operation	Flags Effect
MOV	Move	MOV D,S	S→D	None
XCHG	Exchange	XCHG D,S	D↔S	None

Table (1): Data transfer instruction

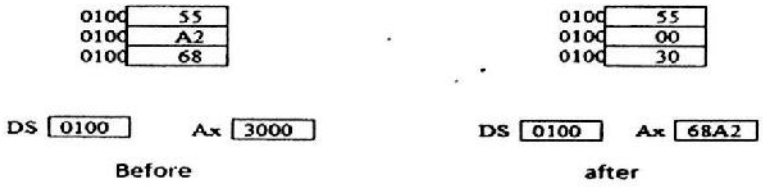
Example 1: For the figure below. What is the result of executing the following instruction?

MOV Ax , [0001]



Example 2: For the figure below. What is the result of executing the following instruction?

XCHG Ax, [0001]



Operand1	Operand2	Length
Register	Register	2 bytes
Register	Memory	2bytes(indirect) 4bytes(direct)
Register	Immediate	2-3 bytes
Memory	Immediate	3-4bytes(indirect)5-6bytes(direct)

Table (2): length of data transfer instruction

2. Arithmetic instruction

Ins.	Meaning	Format	Operation	Flags Effect
ADD	Addition	ADD D,S	S+D→D	
ADC	Addition with carry	ADC D,S	S+D+CF→D	
INC	Increment by 1	INC D	D+1→D	
AAA	ASCII adjust for addition	AAA		
DAA	Decimal adjust for addition	DAA		
SUB	Subtraction	SUB D,S	D-S→D	
SBB	Subtraction with borrow	SBB D,S	D-S-CF→D	
DEC	Decrement by 1	DEC D	D-1→D	
AAS	ASCII adjust for subtraction			
DAS	Decimal adjust for subtraction			
CMP	Compare	CMP D,S	S-D→D	AF

DIV	divide (unsigned)	DIV D	1)Q(Ax/S8)→AL R(Ax/S8→AH 2) Q(Dx.Ax)/S16→Ax R(Dx.Ax)/S16→Dx	CF OF PF SF ZF
MUL	multiply (unsigned)	MUL D	AL .S8→AX Ax.S16→AX,DX ..	
IDIV	Integer divide (signed)	IDIV D	1)Q(Ax/S8)→AL R(Ax/S8→AH 2) Q(Dx.Ax)/S16→Ax R(Dx.Ax)/S16→Dx	
IMLT	Integer multiply (signed)	IMLT D	AL .S8→AX Ax.S16→AX,DX	

Table (3) : Arithmetic instruction

Example: what is the result of executing the following instruction sequence?

```
ADD AL , BL
AAA
```

Assume that AL contains 32H (the ASCII code for number 2), BL contain 34H (the ASCII code for number 4) , and AH has been cleared.

Solution:

Al	32

BL	34

Before

Al	66

BL	34

after ADD

AL	06

BL	34

after AAA

Example 11: what is the result of executing the following instruction sequence?

```
ADD AL , BL
DAA
```

Assume that AL contains 29H (the BCD code for decimal number 29), BL contain 13H (the BCD code for decimal number 13), and AH has been cleared.

Solution:

Al	29

Bl	13

Before

Al	3C

Bl	13

after ADD

Al	42

BL	13

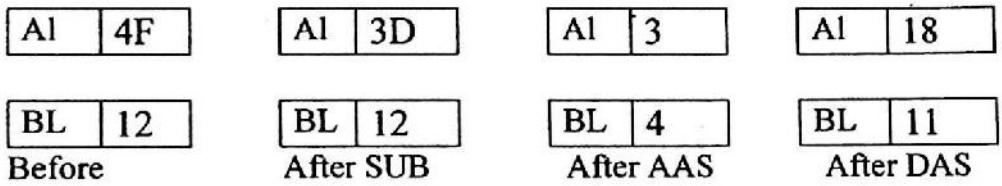
after DAA

Example: what is the result of executing the following instruction sequence?

```
SUB AL , BL
AAS
DAS
```

Assume that AL contains 4FH (the ASCII code for number 7, the BCD code for decimal number 29), BL contain 12H (the ASCII code for number 4, the BCD code for decimal number 11) , and AH has been cleared.

Solution:



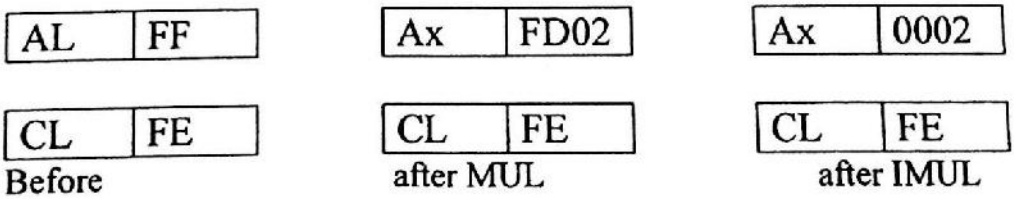
Example 13: what is the result of executing the following instruction?

```
MUL CL
```

What is the result of executing the following instruction?

```
IMUL CL
```

Assume that AL contains FFH (the 2'complement of the number 1), CL contain FEH (the 2'complement of the number 2).



Example: Assume that each instruction starts from these values:

AX= 1250H, BL= 90H

Solution

1) DIV AX/CL= positive/positive = 1250/90= Q= 20, R=50
2) IDIV AX/CL= positive/negative = positive/2' negative =

1250/2'(90) =1250/70 = Q=29 , R=60

But 29H (positive) →2'(29H)= D7H →

AX	1250

BL	90

Before

Ax	5020

CL	90

after DIV

Ax	60D7

CL	FE

after IDIV

Operand1	Operand2	Length
Register	Register	2 bytes
Register	Memory	2bytes(indirect) 4bytes(direct)
Register	Immediate	3-4 bytes
Memory	Immediate	3-4bytes(indirect)5-6bytes(direct)

Table (4): length of data arithmetic instruction

3) Logical group

Ins.	Meaning	Format	Operation	Flags Effect
AND	Logical AND	AND D,S	(S).(D) =D	CF OF PF SF ZF
OR	Logical OR	OR D,S	(S)+(D) =D	
XOR	Logical exclusive OR	XOR D,S	(S)⊕(D) =D	
NOT	Logical NOT	NOT D	(D)= D	

Table (5): Logical group

AND

✓ used to clear certain bits in the operand (masking)

Example : Clear the high nibble of BL register

AND BL, 0FH (xxxxxxxx**AND** 0000 1111 = 0000 xxxx)

Example: Clear bit 5 of DH register

AND DH, DFH (xxxxxxxx**AND**1101 1111 = xx0xxxxx)

OR

✓ Used to set certain bits

Example: Set the lower three bits of BL register

OR BL, 07H (xxxxxxxx**OR** 0000 0111 = xxxx x111)

Example: Set bit 7 of AX register

OR AH, 80H (xxxxxxxx**OR**1000 0000 = 1xxxxxxx)

XOR

✓ Used to invert certain bits (toggling bits)

✓ Used to clear a register by XORed it with itself

Example: Invert bit 2 of DL register

XOR BL, 04H (xxxxxxxx**OR** 0000 0100 = xxxx x***x***x□xx)

Example: ClearDX register

XOR DX, DX (DX will be 0000H)

Operand1	Operand2	Length
Register	Register	2 bytes
Register	Memory	2bytes(indirect) 4bytes(direct)
Register	Immediate	3-4 bytes
Memory	Immediate	3-4bytes(indirect)5-6bytes(direct)

Table (6): length of logical instruction

4. Shift and Rotate Instructions

• The four shift instructions of the 8086 can perform two basic types of shift operations: the logical shift, the arithmetic shift

The source can specified in two ways

Shift by One bit — Value of 1 :

Shift by the value of CL register — Value of CL register :

Ins.	Meaning	Format	Operation	Flag Effe
SAL/ SHL	Shift arithmetic left/shift logical left	SAL/SHL D,Count	Shift the (D)left by the number of bit position equal to count and fill the vacated bits position on the right with zeros	CF OF PF SF ZF AF
SHR	Shift logical right	SHR D,Count	Shift the (D) right by the number of bit position equal to count and fill the vacated bits position on the left with zeros	
SAR	Shift arithmetic right	SAR D,Count	Shift the (D)left by the number of bit position equal to count and fill the vacated bits position on the right with the original most significant bit	
ROL	Rotate left	ROL D,count	Rotate the (D) left by the number of positions equal to count. Each bit shifted out from the leftmost bit goes back into the rightmost bit position	CF OF
ROR	Rotate right	ROR D,count	Rotate the (D) right by the number of positions equal to count. Each bit shifted out from the rightmost bit goes back into the leftmost bit position	
RCL	Rotate Left through Carry	RCL D,count	Same ROL except carry is attached to (D) for rotation	
RCR	Rotate right through carry	RCR D,count	Same as ROR except carry is attached to (D) for rotation	

Table (7): Shift and Rotate Instructions

5. Jump Instructions

There are two types of jump, **unconditional** and **conditional**. In unconditional jump, as the instruction is executed, the jump always takes place to change the execution sequence.

a) **Unconditional Jump**

Ins.	Meaning	Format	Operation	Flags Effect
JMP	unconditional jump	JMP operand	Jump is initiated to th address specified by the operand	none

Table (8): Unconditional Jump

Examples:

JMP 1234H; IP will take the value 1234H

JMP CX; IP will take the value in CX

JMP [CX]; IP will take the value in memory location pointed to by CX

JMP X

- ✓ Short (2bytes): the value of X is between -128 <= X <=127

  Ex: JMP 10

  JMP -30

- ✓ Near: the operation of jump is taken place with in the same segment (64kb).

  Ex: JMP 200
  JMP 2000
  JMP 1500

- ✓ Far: the operation of jump is taken place in other segment

  Ex: JMP 0014H:3456H

JMP 1990H:4531H

b) Conditional Jump

instruction	Condition	Length
JNC	C=0	2 bytes
JC	C=1	
JS	S=1	
JNS	S=0	
JE,JZ	Z=1	
JNE,JNZ	Z=0	
JO	O=1	
JNO	O=0	
JP,JPE	P=1	
JNP,JPO	P=0	
JA (above)	S=0,Z=0	
JAE	S=0	
JB (below)	S=1,Z=0	
JBE	S=1	

Table (9): Conditional Jump

6. Subroutines and subroutine-handling instructions

- ✓ A subroutine is a special segment of program that can be called for execution form any point in program.
- ✓ There two basic instructions for subroutine : **CALL** and **RET**
- ✓ **CALL** instruction is used to call the subroutine.
- ✓ **RET** instruction must be included at the end of the subroutine to initiate the return sequence to the main program environment.

Examples: CALL 1234h
CALL CX
CALL [CX]

7. PUSH and POP instruction