Poornima . V . Doijad
Microprocessor
CSE

Unit : 2

The Microprocessor & its Architecture

Introduction :-

In 1978, Intel released its first 16-bit microprocessor, the 8086. The 8086 can address (1MB = 2^{20} bytes) of memory, as it has a 20 bit address Bus.

The width of the data bus in the 8086 is 16 bits.

Another feature in the 8086 is the presence of a small six-byte instruction queue in which the instructions fetched from the memory are placed before they are executed. It is a 40pin, Dual Inline packaged IC

*) Architecture of 8086 :-

The functional blocks diagram of 8086 is subdivided into two units :-

i) An execution unit :-

which includes the ALU, eight 16-bit register, 16-bit flag register, and a control unit

ii) A Bus interface unit :-

It consist of four 16-bit Segment registers [cs, DS, SS & ES], a 16-bit instruction pointer (IP), a six byte instruction queue, & a bus control logic.

1) The Bus interface Unit (BIU)

Functions :-

1) It fetch the instruction (or) data from memory
2) It write the data to memory
3) It write the data to ports (I/O ports)
4) It reads data from ports

Bus interface unit mainly contains the
→ 4 Segment register
→ Instruction pointer
→ Instruction Queue

Three functional unit which is used to perform the above functions.

Instruction pointer → Address of the next instruction to be executed

It is a 16-bit register. It holds the offset of the next instruction in the code segment

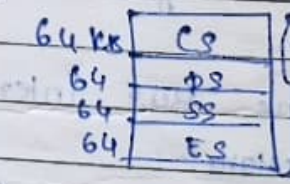
IP is incremented after every instruction byte is fetched.

Address of the next instruction is calculated as CS X 10H + IP

Segment Register :-

The memory space of 8086 is 1MB. The Segment register holds the 1MB of 8086 and segment into 4 blocks, Each block is specified by register with maximum size of 64 KB.

The four Segment registers are
→ Data Segment register (DS)
→ Stack Segment (SS)
→ Extra Segment (ES)
→ Code Segment (CS)



Code Segment holds the base address for the code segment. All programs are stored in the code segment

Data Segment holds the base address for the data segment

Stack Segment (SS) holds the base address for the Stacks Segment

ES [Extra Segment] holds the base address for the extra segment.

Instruction Queue :-

It is a 6-bit Queue. The Bus interface unit perform its operation in parallel with execution unit.

This Queue is used in 8086 in order to perform pipelining. At the time of decoding and execution, the BIU fetches the sequential upcoming instructions and stores it in the queue. The queue exhibits FIFO behavior.

2) Execution Unit :-

The Execution unit (EU) performs the decoding and execution of the instructions that are being fetched from the desired memory locations.

Functions :-

1) It is used to tell where to fetch the instruction and data from to the bus interface unit
2) To decode the instructions
3) To Execute the instructions
4) It contains the Control unit to perform Various internal operations.

Functional Parts :-

1) General purpose Registers
2) Pointer and index Registers
3) Arithmetic Logic Unit
4) Flag registers
5) Timing & control unit.

1) General purpose registers :-

The Execution unit consist of 16-bit general purpose registers — AX, BX, CX, DX.

Among these registers AX, BX, CX, DX can be further divided into two 8 bit registers

AH AL, BH BL, CH CL, DH DL → 8-bit

AX BX CX DX → 16 bit.

AX :- AL is used as a accumulator. It is used in the multiply, divide, and input & output operation.

BX :- It is used to refer data in the memory using the look-up table technique.

CX :- CX is used to hold the count value. The count value indicates the number of times the same code has to be executed, no of times the data item has to be shifted / rotated.

DX :- It is used to hold a part of the result during a multiplication operation and a part of the dividend before the division operation.

2) Pointer and Indexed register :-

The 8086 microprocessor has two, pointer register and two indexed register.

Pointer

1) Stack pointer
2) Base pointer

Index register

1) Source index
2) Destination index

Stack pointer :- (SP)

Stack pointer is used to hold the offset

address of the data stored at the top of the stack segment. The stack pointer register is used along with the stack segment (SS) to decide the address at which the data is to be pushed (or) popped.

Base Pointer :- (BP):

It is also used to hold the address of the data to be read from (or) written into the stack segment.

Source index (SI)

It is used to hold the address of the source data in the data segment, while executing string instructions.

Destination index (DI) :-

It is used to hold the address of the destination data while executing string instructions.

3) Arithmetic & Logic unit :-

The ALU is 16 bit and it performs arithmetic & logical operations.

It performs 8 bit and 16-bit data.

Addition, substraction, multiplication & division.

4) Flag register :-

The 8086 has 16-bit flag registers. In this 16-bit flag, the 9 flags are used and remaining 7 flags are unused.

The 9 flags in the flag register classified into status flag & control flags.

There are 6 status flags and 3 control flags.

The status flags are carry flag (CF), parity flag (PF), Auxillary carry flag (AF), Zero flag (ZF), Sign flag (SF) & ~~zero~~ flag (OF). (OF) overflow flag.

The control flags are direction flag (DF), Interrupt flag (IF), Trap flag (TF). It controls the operation of the CPU.

1 = CY
0 = CNY

CF :- Carry flag holds the carry after an 8-bit (or) 16-bit addition (or) the borrow after an 8-bit (or) 16-bit substraction.

E=1, O=0

PF :- Parity flag → odd parity (0) & even parity (1)

AF :- Auxillary carry It also holds the carry after addition (or) substraction

A^0 A^0
0=1, 1=0

ZF :- Zero flag indicates the result of arithmetic (or) Logic operation.

P=1, N=0

SF :- Sign flag → Holds the arithmetic sign of the result after an arithmetic (or) logic instruction is executed.

Correcting

TF :- Trap flag used to debug a program.

DF :- Direction flag If D=0 the registers are automatically incremented if it is D=1 the registers are decremented. → SI, DI registers.

IF :- Interrupt flag If IF=0 the INTR pin is disabled if IF=1, the INTR pin is enabled.

OF :- Overflow flag → For 8-bit data the value is in between -128 = 80H & +127 = 7FH, For 16-bit -32,768 = 8000H & +32,767 = 7FFFH. If the result cross the limit the overflow of data occur.

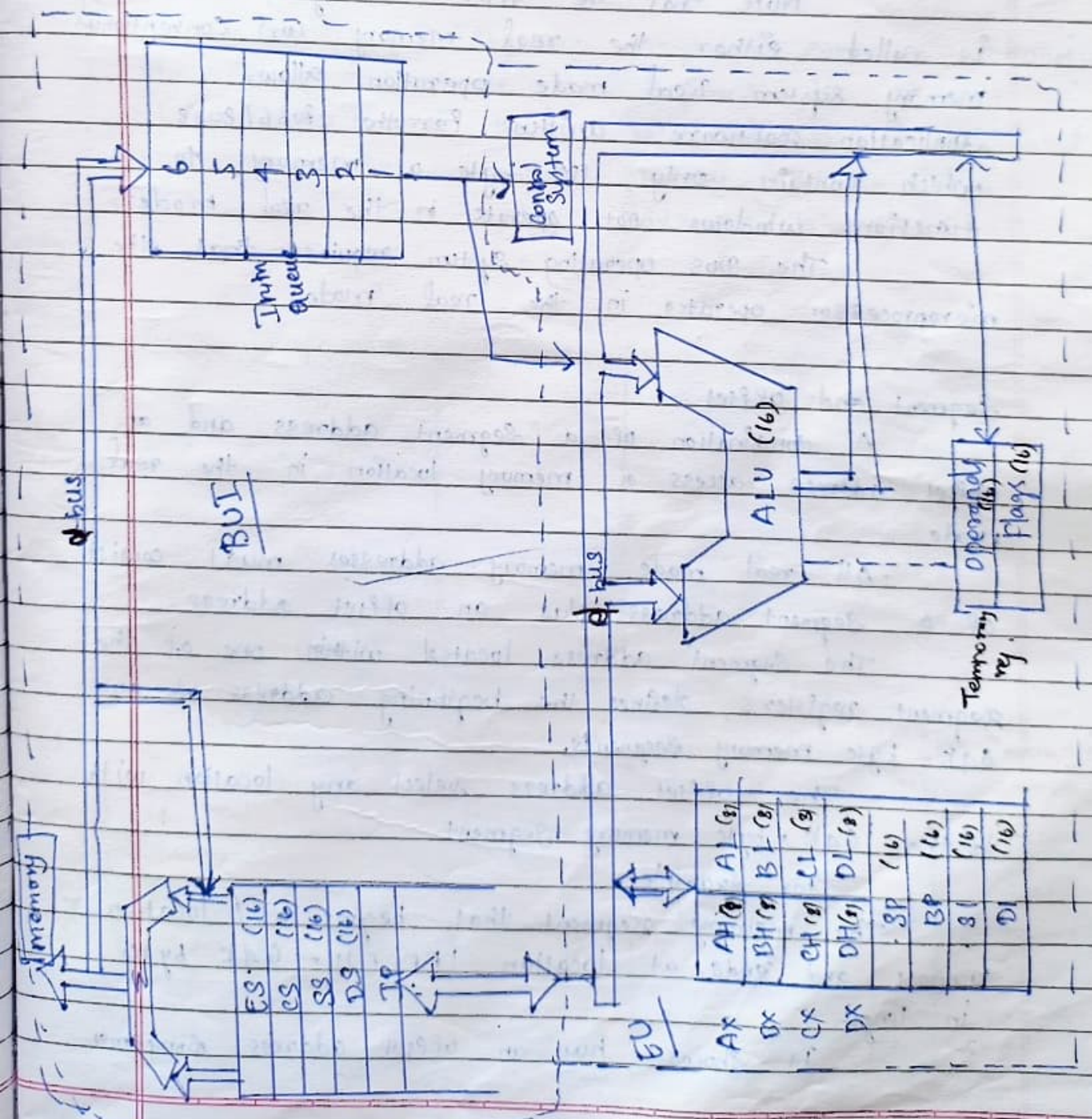
CS	IP
SS	SP, BP
DS	SI
ES	DI

D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
-	-	-	-	OF	DF	IF	TF	SF	ZF	-	AF	-	PF	-	CF

Flag register of 8086.

5) Timing & control unit :-

The control unit of execution unit directs all internal operation and also responsible for generation of signals



Real Mode Addressing:-

The 80286 and above operate in either the real or protected mode.

only the 8086 & 8088 operate exclusively in the real mode.

Real mode operation allows the microprocessor to address only the first 1M byte of memory

Note that the first 1M byte of memory is called either the real memory (or) conventional memory system. Real mode operation allows application software written for the 8086/8088, which contain only 1M byte of memory, to function. Windows not operate in the real mode

The Dos operating system requires that the microprocessor operates in the real mode.

Segment and offset :-

A combination of a segment address and an offset address access a memory location in the real mode.

All real mode memory addresses must consist of a segment address plus an offset address.

The segment address located within one of the segment registers, defines the beginning address of any 64k-byte memory segments.

The offset address select any location with in the 64k-byte memory segment

for example;

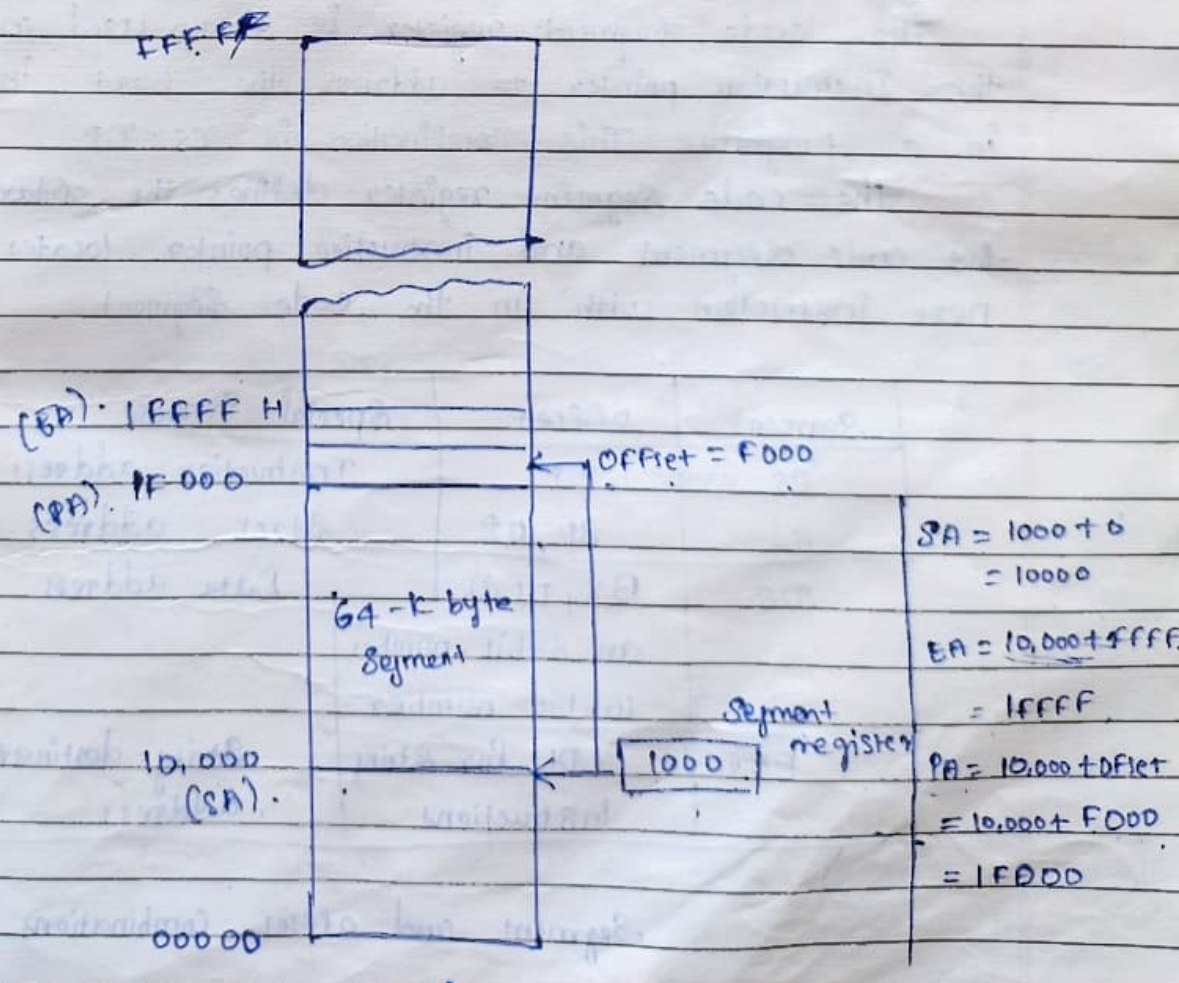
The memory segment that begins at location 10,000H and ends at location 1FFFFH - 64k bytes in length.

It shows how an offset address sometime

called as displacement, F000H selects location 1F000H in the memory system. Note that the offset (or) displacement is the distance above the start of the segment

Because a real mode segment of memory is 64 K in length, once the beginning address is known, the ending address is found by adding FFFFH.

For ex:- A segment register contains, 3000H the first address of the segment is 30,000H [3000H+0] and the last address is calculated by using the formula starting address plus FFFF H. The ending address is 3FFFF H



Example of real mode Segment addresses

Segment Register	Starting Address	Ending Address
2000 H	20000H	2FFFFH
2001H	20010H	3000FH
2100 H	21000H	30FFFH
AB00H	AB000H	BAFFFH
1234 H	12340H	2233FH

12340 FFFF

Default Segment and Offset Registers :-

The microprocessor has a set of rules that apply to segments whenever memory is addressed, these rules apply in the real and protected mode, define the segment register & offset register combination for example :

20010 FFFFF

The code segment register is always used with the instruction pointer to address the next instruction in a program. This combination is CS : IP

The code Segment register defines the start of the code segment and instruction pointer locates the next instruction with in the code segment.

Segment	offset	Special Purpose
CS	IP	Instruction address
SS	SP, BP	Stack address
DS	BX, DI, SI an 8-bit number, 16-bit number	Data address
ES	DI for string Instructions	String destination address

Segment and offset Combinations

We can specify the data through register or a memory address

Another default combination is the stack. Stack data are referred through the stack segment at the memory location addressed by either the stack pointer (or) base pointer (SS/SP).

*) Addressing Modes — the data to be operated by an instruction

*) Data Addressing Mode:- is defined as the way of specifying

The term effective address (EA) represent the offset address of the data within a segment, which is obtained by different methods, depending upon the addressing mode that is used in the instruction.

Let us assume that the various registers in the 8086 have the following values stored in them.

For eg:-

Register	CS	DS	SS	ES	BX	BP	SI	DI
Stored values	1000H	8000H	4000H	6000H	2000H	1000H	1000H	3000H

There are different data memory addressing modes are as follows

1) Direct addressing
2) Base addressing
3) Base relative addressing
4) Index addressing
5) Index relative addressing
6) Base plus index addressing
7) Base relative plus index addressing

1) Direct Addressing:-

In this mode, the 16-bit offset address of the data within the segment is directly given in the instruction.

Example :-

a) Mov AL, [1000 H]

In this instruction, the effective address is 1000 H. Since the destination is an 8-bit register.

DS X10H + EA

= 3000H X10H + 1000 H

= 30000H + 1000 H

= 31000 H

b) Mov BX, [2000H]

It is a 16-bit register since BX is 16-bit

DS X10H + EA

= 3000 X10 + [2000 H]

= 30000 + 2000 H

= 32000H

[Note :- Since a coord contain two bytes, the bytes present at the memory addresses 32000H and 32001H are moved to BL and BH respectively.]

2) Base Addressing :-

In this mode, EA is the content of the BX (or) BP register. When Bx is present data is taken from the data segment and when BP is present, data is taken from stack segment

Example :-

a) Mov CL, [BX]

Memory address [EA] = DS X10 + (BX)

= 32000H.

b) Mov DX, [BP]

EA = SS X10+[BP]

= 4000 X10 + 1000

= 40,000 + 1000

= 41000H.

3) Base Relative addressing :-

In this mode, EA is obtained by adding the content of the base register with an 8-bit (or) 16-bit displacement. The displacement is a signed number with negative values represented in 2's complement form.

The 16-bit displacement value ranges from -32768 to +32767

The 8-bit values is -128 to 127.

Example :-

a) Mov AX, [BX+5]

Memory address = DS×10H + [BX+5]

= 30000H + 2000H + 5

= 32005H

The word from the memory address 32005H is read & stored in AL.

b) Mov CH, [BX-100H]

EA = DS×10H + (BX) - 100H

= 30000H + 2000H - 100H

= 31F00H.

4) Index Addressing :-

In this mode EA is the content of SI and DI register, The data is taken from the data segment.

a) Mov BL (SI)

Memory address = DS×10H + SI

= 3000×10H + 1000H

= 31000H.

A byte (or) word is read from 31000H and stored in BL.

b) Mov CX, [DI]

Memory address = DS x 10H + (DI)
= 30000H + 3000H
= 33000H

A word from memory address 33000H is read and stored in CX.

5) Index relative addressing :-

This mode is same as the base relative addressing. except that instead of BP or BX register, the SI (or) DI register is used.

a) Mov BX, [SI - 100H].

Memory address = DS x 10H + (SI) - 100H
= 30000H + 1000H - 100H
= 30F00H.

b) Mov CL, [DI + 10H].

Memo EA = (DI) + 10H

Memory address = DS x 10H + (DI) + 10H
= 30000H + 3000H + 10H
= 33010H.

6) Base plus index addressing :-

In this mode, EA is obtained by adding the content of base register and index register

Eg :-

a) Mov AX, [BX + SI]

EA = [BX + SI]

Memory Address = DS x 10H + (BX) + (SI)
= 30000H + 2000H + 1000H
= 33000H

7) Base relative plus index addressing :-

In this mode EA, is obtained by adding the content of a base register, index, and a displacement.

Eg:-

a) MOV CX, [BX + SI + 50H]

EA = [BX) + (SI) + (50H)

Memory Address = DS X10H + (BX) + (SI) + 50H

= 30000H + 2000H + 1000H + 50H

= 33050H.

A word from the memory address 33050H is read & stored in CX.

8) Program Memory Addressing Mode:-

The program memory addressing mode used with the JMP and CALL instruction. and consist of three distinct forms

1) Direct
2) Relative
3) Indirect

1) Direct :-

Direct program memory addressing stores both the segment and offset address where the control has to be transferred with the opcode.

Eg:-

The instruction is equivalent to JMP 32000H. when it is executed, the 16-bit offset value 2000H is loaded in the IP register and the 16 bit Segment value 3000H is loaded in CS.

When the microprocessor calculates the memory address from where it has to fetch the instruction using the relation CS X10H + IP. the address 32000H is obtained using CS and IP values.

EAH	00H	20H	00H	30H
(opcode)	(lower order byte)	(IP-Higher order byte)	(CS - lower)	(CS - Higher)

CS x 10H + IP

= 32000H.

This type of jump is known as inter-segment jump. using which the microprocessor can jump to any memory locations within the memory system (i.e) within 1MB. It is also known as FAR jump

Eg:-

a) JMP FAR PTR CONTINUE

b) JMP FAR PTR SIMULATE

The assembler directives FAR PTR is sometime used to indicate the inter-segment jump instruction.

In above example continue & simulate are the labels of memory locations that are present in the code segment

2) Relative Addressing :-

The term relative means relative to the instruction pointer (IP). Relative JMP & CALL instructions contain either an 8-bit (or) 16-bit signed displacement, which is added to the instruction pointer.

Based on the new value of IP, the address is calculated by using the formula

CS x 10H + IP

The 8-bit (or) 16-bit signed displacement allows a forward (or) reverse memory reference, depending on the sign of the displacement.

If the displacement is positive, the PC is incremented by the displacement value. If displacement is negative, the PC is decremented

A one-byte displacement is used in the short jump, call instructions. A two-byte displacement is used in the near jump & call instructions.

An 8-bit displacement has a jump range between -128 to 127 & 16-bit displacement has a jump range from -32768, +32767

The Opcode for the relative short jump & near jump instructions are EBH & E9H

Eg :-

```
JMP START OVER
JMP NEAR PTR FIND.
```

OVER & FIND are the labels of memory locations that are present in the code segment.

The assembler directives SHORT & NEAR PTR is used to indicate the short jump & near jump.

B) Indirect Addressing :-

The indirect jump (or) call instructions use a 16-bit register (AX, BX, CX, DX, SP, BP, SI or DI) a relative register (BP), (BX), (DI) or (SI) or a relative register with displacement.

The opcode of the indirect jump instruction is FFH. It can be either inter-segment (or) intra segment indirect jump.

If a 16-bit register holds the jump address in an indirect JMP instruction, the operation is near jump. If the CX register contains 2000H and the jump CX instruction present in a code segment is executed. the microprocessor jumps to the offset address 2000H in the current code segment to take the next instruction for execution (this is done by loading the IP with the content of CX without changing the content of CS)

f) Stack Memory Addressing mode:-

The Stack is used to hold the data temporarily during program execution and also used to store the return address for procedures

The stack memory is a last-in first out memory (LIFO). Data are placed into the stack using push and taken out using the POP instruction

The CALL instruction is used to hold the return address for procedures. and RET instruction is used to remove the return address from the stack.

The stack segment is maintained by two register stack pointer (SP) and stack segment (SS).

Data is pushed or popped from the stack as words (16-bit data). Since bytes (8-bit data) cannot be used with the PUSH & POP instructions.

When a word of data is pushed into the stack, the higher order eight bits of the word are placed in the memory location specified by SP-1 (i.e the address SSX10H+SP-1) and the lower order 8-bits of the word are placed in the memory location specified by (SP-2) (i.e) the address (SSX10H+SP-2)

Eg:-

PUSH AX - Push the content of AX into stack
PUSH DS - Push the content of DS into Stack
PUSH DX - Push " " " DX into Stack.

PUSHF instruction is used to push the flag register content into the stack.

POP BX - Pop the content of BX from the stack
POP ES - Pop " " " ES " " "
POP [BP] - Pop " " " BP " " "

The POPF instruction is used to pop a word stored in the stack moved into the flag register.

Push Bx.

Bx = 1234
L H

SP = 2000 H.

SP-1 = H

1999 = 34

SP-2 = L.

1998 = 12.