Assignment No: 1

Page No.:
Date:
YOUVA

Q.1 With key Features Compare 8085 up & 8086 UP

	8085	8086
1.	8-bit microprocessor	1. 16-bit microprocessor.
2.	Non-pipeline micro-processor.	2. Pipelined microprocessor. Hence, perform high and fetch time is eliminated.
3.	Provide on-chip clock generation	3. Requires external clock generator.
4.	Memory is not segmented.	4. Memory is segmented, Hence can be used for multiprograming
5.	Can access 64 k memory locations & 256 IO ports	5. Can access 1 MB Memory location & 64k IO ports.
6.	Provides 5 interupts. But interupts are not Flexible.	6. Provides 256 interupts. they are Flexible.
7.	Does not provide on-chip Memory managment Unit	7. Provide on chip Memory Mgt unit
8.	Does not provide multiply & divide operations.	8. Provides multiply & divide operations.

Q. 2

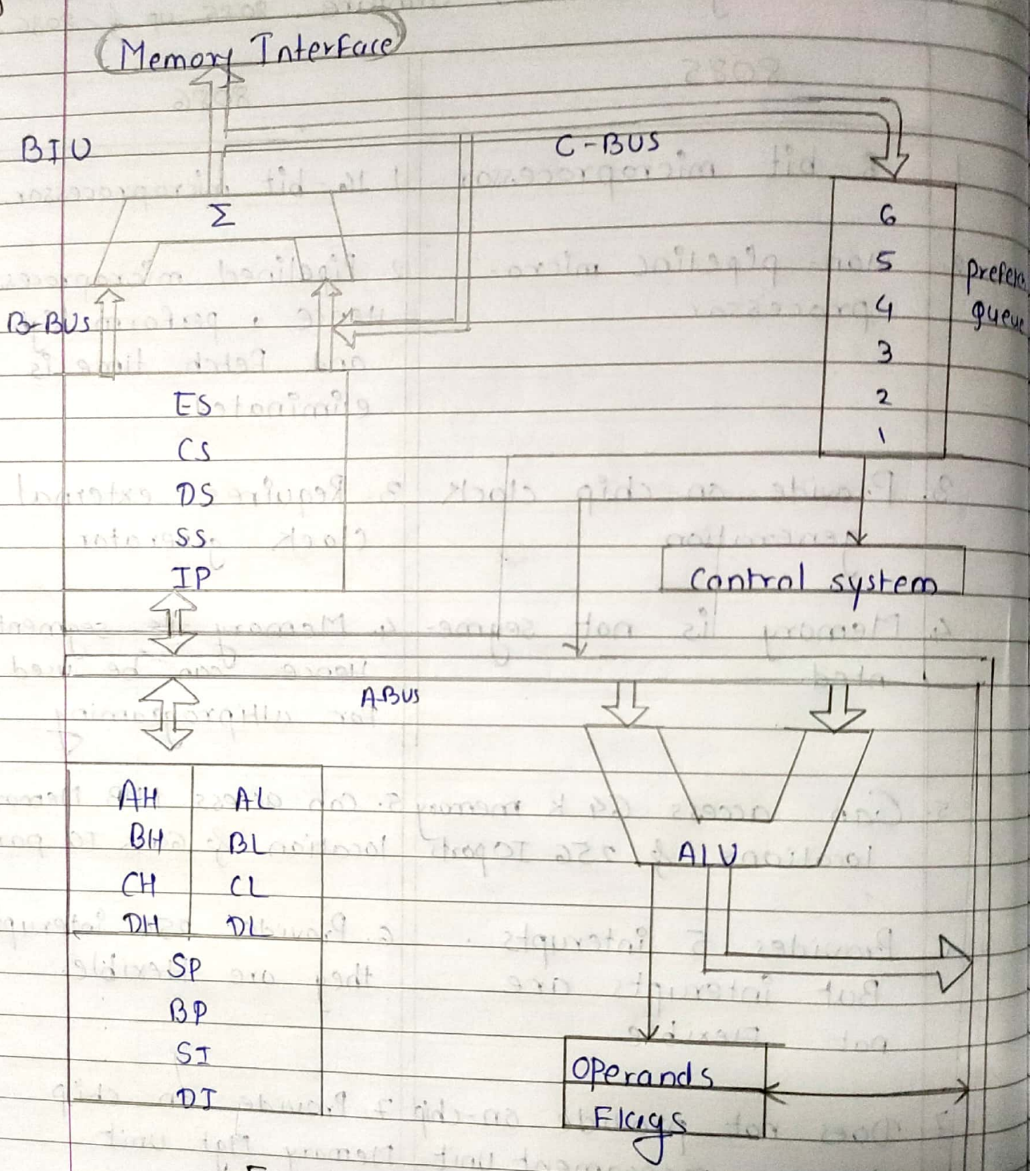

Fig :- 8086 block diagram

BIU : [Bus interface unit]

① BIU acts as an interface betⁿ the system bus and ~~exter~~ execution unit.
② It sends out address to I/O ports and memories.
③ BIU fetches the instruction. From Memory
④ It read data From I/O ports & memories.
5) It writes data to port & mories.
6) BIU supports pipelining
7) BIU consist of Prefect queue, segment register, instruction painter, 20-bit adder.

I) Instruction Prefetch Queue :-

1) BIU fetches as many as 6 instⁿ bytes ahead of time From memory.
2) These pre-fetched instruction bytes are held in FIFO group of register, refferd as instruction queue.
3) These bytes are then given to EU.
4) This pre-fetching operation of BIU is in parallel with execution operation of EU which improves speed of execution.

II) Segment Register

1. BIU contains 4 16-bit segment register.
2. - Code segment Register (CS)
3. - Data segment Register (DS)
   - Stack segment Register (SS)
   - Extra segment Register (ES)

3.) CS register holds 16-bit of starting address of code segment. (A memory segment where the program is stored).

4) DS register holds 16-bit of starting address of Data segment (A memory segment where Data is stored).

5) SS register holds 16 bit address of stack segment (A memory where data/~~segment~~ address is stored).

6) ES register holds 16 bit starting address of Extra segment. (A memory segment used to store Additional data segment.)

III) Instruction Pointer :-

1) It is 16-bit register.
2) It holds 16-bit address of the next code byte to be executed within code segment.
3) The value in Instruction pointer is refered as offset. It is added to the content of CS register to obtain physical address of next inst^n word to be executed.

Date:

2. EU (Execution Unit) :-

1) EU provides addresses to BIU from where instruction / Data is to be Fetched.
2) It reads inst^n from pre-fetch queue, then decodes & executes the inst^n.
3) It contains control system, instruction decoder, ALU, Flag register, general purpose Register.

I) Control System :-

1) EU contains control circuit which direct all internal operations.
2) It controls all the operations & Flow of data within microprocessor.

II) Instruction Decoder :-

1) It decodes the instruction Fetch from memory & generates Various control signals to complete the operations.

III) ALU :- 1) EU of 8086 has a 16-bit ALU which carries out Arithmatic & logical operations on 8 bit as well as 16-bit data.
2) It performs operations like Addition, Subtraction, AND, OR, XOR, Increment, decrement, rotate etc.

4) Flag Register

i) EU of 8086 has 16-bit flag register. It contains 9 active flags.

i) Carry Flag :- • This flag is set to 1, when the carry is generated from MSB's of operands in addition.
• In subtraction this flag functions as borrow.
• If there is No carry / borrow the ALU reset this flag to 0.

ii) Parity Flag :-
• 8086 provides even carry flag
• If result contains even no. of 1's then ALU set the parity flag to 1.
• If result contains odd no. of 1's then ALU reset this flag to 0.

iii) Auxilliary Carry Flag :-
• If an operation performed by ALU generates carry / borrow from D3 bit then AC flag is set to 1 otherwise it is reset to 0.

IV) Zero Flag :- If the result of operation is zero the ALU set this Flag to 1 otherwise reset zero flag.

V) Sign Flag :- • This Flag is used in signed arithmatic operations.
• It is ignored in unsigned arithmatic & logical operation.
• If the result of operation is negative then Sign flag is set to 1 otherwise it is reset to 0.

VI) OverFlow Flag :- • In signed arithmatic operation MSB is reserved for sign.
• In such case if result is too long to fit in 7 bit or 15 bit overflow flag is set otherwise it is reset.

VII) Trap Flag :- • By setting this Flag 8086 is forced into single step mode.
• This allows user to execute one inst^n of a program at a time.

VIII) Interupt Flag :- • IF is set, 8086 recognise external interupts on INTR
• If it is reset, 8086 does not recognise external interupts

IX) Direction Flag :- • It is used in string operations
• If it is set, string pointer is auto decrement.
• If it is reset, string pointer is auto incremented.

5] General Purpose Register :-

1) EU of 8086 has eight 8-bit general purpose register.
AH, AL, BH, BL, CH, CL, DH, DL
2) These registers can be used for storage of 8-bit data.
3) These register can be used in pair for 16 bit storage
AH-AL = AX BH-BL = BX
CH-CL = CX DH-DL = DX
4) Some of these register are also used for special task, for ex CX is used as counter.
5) It also contains SP, BP, SI, DI
6) SP is stack pointer which contain 16-bit offset within stack segment.
7) BP is Base pointer which is used to hold 16-bit offset from particular segment.
8) SI is Source index register.
9) DI is destination index register

Q.3 With the example, explain all addressing mode of INTEL - 86 µP.

⇒

1. Immidiate Addressing Mode :-

IF data is present in the inst itself, it is called as Immidiate addressing mode.

e.g :- ADD AL, 01H
MOV CX, 1048H

2) Direct Addressing Mode :-

If the data is present at the 16 bits address which is also called as offset address & the segment address being indicated by the inst^n is called as direct Addressing mode.

For. e.g :-

MOV AX, [5000H] Suppose DS = 2345H

P.A = DS x 10 + [5000H]

= 2345 + 10 + 5000

= 28450

3) Register Addressing Mode :-

In this mode, operands are specified during use of registers also register may be use as source operand, destination operand or both.

Ex :- MOV BX, CX

ADD CL, AL

4) Register Indirect Addressing Mode :-

If the data is present in the address equal to the value of data in register mentioned in the inst^n then it is called as register indirect Addressing Mode.

Ex :-

MOV AX, [BX]

5) Register Relating Addressing Mode :-

If the data is located at an address equal to the sum of value & contaihts of register then it is called as register relative Addressing mode

Ex :- MOV CL, [BX + 04H]

E.A = DS * 10 + [BX + 04H]

6] Index Addressing Mode :-

In this addressing mode, the operand address is calculated as segment / Base register + an index register.

For. e.g :-

MOV AX, [SI]
MOV DX, [DI]

E.A = DS * 10 + [SI]
DS * 10 + [DI]

7] Base Indexed Addressing Mode :-

If the data is located at an address is equals to the sum of value in a register & an indexed register like SI, DI, then it is called as Base indexed AM

For. e.g :-

MOV AX, [BX][SI]

E.A = DS * 10 + [BX] + [SI]

8) Relative Based Indexed Addressing Mode.
This is the combination of Index Addressing & Base Indexd addressing mode.
e.g :-

MOV AX, 40H [BX][SI]

EA = DS * 10 + 40H + [BX] + [SI]

Q. 4 What are different two types of procedure what is CALL & RET inst^n for FAR & NEAR Procedure.

⟹

1) Procedure :- (i) A procedure is group of instruction that usually performs one task. It is a reusable section of a software which is stored in memory once but can be used as often as necessary.

(e) There are 2 types of Procedure.
i) Near Procedure ii) Far Procedure.

i) Near Procedure :- If, A procedure is written in the same code segment which is calling that procedure called as Near Procedure.

ii) FAR Procedure :- A procedure is known as FAR procedure if is written in the different code segment that the calling segment.

II) CALL Instruction :- ① The CALL inst^n is used to transfer execution to a procedure ② It perform two operation; when it executes, First it stores the address of instruction after the CALL instruction on the stack. ③ Second it changes the content of IP in case of Near CALL & changes the content of IP & CS in case of FAR call.

④ There are two types of CALLS
- Near CALL
- FAR CALL

* Operation for Near CALL :- When 8086 executes a near CALL instruction, it decrements the stack pointer by 2 & copies the IP register contents on to the stack. It copies address of first inst^n of called Procedure.

SP ← SP-2
IP → Store onto stack
IP ← Starting address of a procedure.

* Operation of FAR CALL :- when 8086 executes FAR CALL, it decrements the stack pointer by 2 & copies the contents of CS to the stack. It decrements the stack pointer by 2 again & copies the content of IP register to the stack.

SP ← SP-2
CS contents → stored on stack
SP ← SP-2
IP contents → stored on stack
CS ← Base Address of segment having Proced.
IP ← address of first inst^n in procedure.

III) RET Instruction :- 1) The RET inst^n will return execution from a procedure to the next instruction after CALL the main Program.
2) At the end of every procedure RET inst^n must be executed.
3)
Operation for Near Procedure :-

1) For Near procedure, the return is done by replacing the IP register with a address popped off from stack & then SP will be incremented by 2.
IP ← Address from top of stack
SP ← SP + 2

2) Operation for FAR Procedure :-
IP register is replaced by address popped off from top of stack, then SP will be incremented by 2.

Ip ← Address from top of stack
SP ← SP + 2
CS ← Address from top of stack
SP ← Sp + 2