Vidyalankar

S.Y. Diploma : Sem. IV [CO/CD/CM/CW/IF]

Microprocessor and Programming

Prelim Question Paper Solution

1. (a) (i) (1) TRAP :

- This is input signal.
- This is nonmaskable interrupt and has the highest priority.
- It is vectored interrupt.

(2) READY :

- This is input signal.
- This signal is used to delay the microprocessor read and write cycles until a slow-responding peripheral is ready to send or accept data.
- When this signal goes low, the microprocessor waits for an integral number of clock cycles until it goes high.

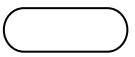
1. (a) (ii) Segment Registers :

- 8086 has four segment registers CS, DS, ES, SS which are 16 bit in length.
- They hold the base address of the segment.
- CS register hold the base address of code segment.
- DS register hold the base address of data segment.
- SS register is used to hold the base address of stack segment.
- ES register hold the base address of extra segment which is used as another data segment.

1. (a) (iii) The instruction of 8086 can be divided into following groups.

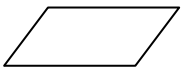
- Data transfer instructions
- Arithmetic instructions.
- Logic instructions
- Shift instructions
- Rotate instructions
- Flag control instructions
- Compare instructions
- Control flow and the jump Instructions
- Loop Handling Instructions.
- String Handling instructions.
- Special Instructions.

1. (a) (iv) Following are the symbols are used in a flowchart.



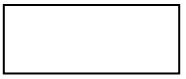
Start/End

This symbol is used to indicate Start or End of the program.



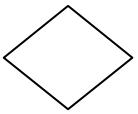
Input/Output

The above symbol is used to get the data and display the data i.e. for taking the input and display the output.



Processing

The above symbol is used when any arithmetic operation performed in program.



Decision box

The above symbol is used to check the condition.

1. (a) (v) Special Purpose Registers :

This group consist of two registers :

PC (Program Counter) :

- This is 16 bit register responsible for sequential execution of program.
- It always holds the address of next instruction to be fetched from memory for execution.
- When instruction pointed by it, is fetched and goes for execution, program counter is automatically incremented by 1 to point to the next instruction.
- In case of nonsequential execution, its contents are directly replaced by branch target address.

SP (Stack pointer) :

- This is 16 bit register.
- When program control goes to the called subroutine or interrupt subroutine, the returning address is stored in special memory called stack memory. The stack memory is LIFO (Last In First Out) memory and its current top is pointed by stack pointer.
- Sometimes the contents of registers are also saved on stack to make them free so that they can be used in subroutine and when program control returns back to main calling program, the original values of registers are reloaded from stack.
- When information is stored on stack, SP is incremented and when information is removed from stack, SP is decremented.

1. (a) (vi) Control Flags :

Trap Flag (TF) : If the flag is set, 8086 goes into the single-step mode of operation. When in the single-step mode, it executes an instruction and then jumps to a special service routine that may determining the effect of executing the instruction. This can be used for debugging.

Interrupt Flag (IF) : If this flag is set, the interrupt signal on INTR pin will be processed otherwise it will be ignored.

Direction Flag (DF) : When this flag is set, the processing string will take from higher address to lower address. If reset, processing of string will be from lower address to higher address.

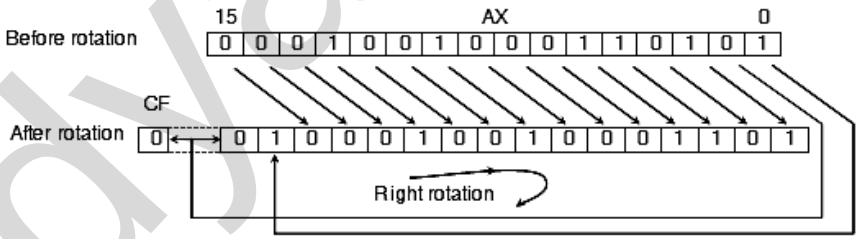
1. (a) (vii)Comparisons between Procedures and Macro :

	Procedure	Macro
1)	Procedure is an assembly language structure to implement the underlying mechanism of subroutines.	Macro does not implement any underlying mechanism but it substitutes runtime the block of instruction.
2)	The procedure is a subroutine in machine language and therefore makes use of stack. Before ending procedure, RET instruction is a must.	The macros do not make use of stack. Being runtime substitutions they do not require RET instruction.
3)	Procedures can be intrasegment (near) or intersegment (far).	Macros being runtime substitutions must be within a same code segment.
4)	In procedures, the control is transferred from main program to the subroutine. The body of procedure is executed and then the control is returned to main program.	In macro, the body of macro is substituted in the main program itself and is execute without any control transfer in the main program.

1. (a) (viii) ROR D, count instruction rotates the content of destination to the right by number of bit positions equal to count (either 1 or equal to content of CL). Each bit coming out from rightmost bit goes into the leftmost bit position. The last bit coming out from rightmost bit is also copied in CF.

e.g. (CL) = 02), (AX) = 1235_H

if ROR AX, CL is executed, operation will take place as follows.



1. (b) (i) Function of following assembly language programming tools.

(1) Editor: Editor is a program which helps you to construct assembly language program. Using an editor is creates a source program (.as m). eg. Turbo Editor.

(2) Assembler: Assembler is a program is used to convert assembly language instruction into machine language or binary instruction. Using assembler it generates (.obj) object file of source program. example : TASM Turbo Assembler

(3) Linker: Linker is a program which combines more than one separate modules into one large program. So it makes (.exe) executable file. example : TLINK Turbo Liner

(4) Debugger: The process of locating and correcting errors in a program called as debugging. Debugger is a program used to correct the errors in single step mode.
example : TD Turbo Debugger

1. (b) (ii) (1) **END – End program :** The END directive is put after the last statement of a program to tell the assembler that this is the end of program module. If program is made up of many procedures, then the name of main procedure is written after END directive. If ABC is main procedure then END ABC will be last statement of program. Any statement after this will be ignored by assembler.

(2) **DW – Define Word:** The DW directive is used to tell the assembler to define a variable of type word or to reverse storage locations of type word in memory.
e.g. STORAGE DW 1234–H, 3456H, 5678H
STORAGE DW 100 DUP (0)
STORAGE DW 100 DUP (?)
STORAGE is the same given to memory location.

(3) **EQU – Equate :** It is used to give a name to some value or symbol. Each time the assembler find the given name in the program, it will replace the name with the value or symbol you equated with that time.
e.g. ABC EQU 100
DECIMAL – ADJUST EQU DAA

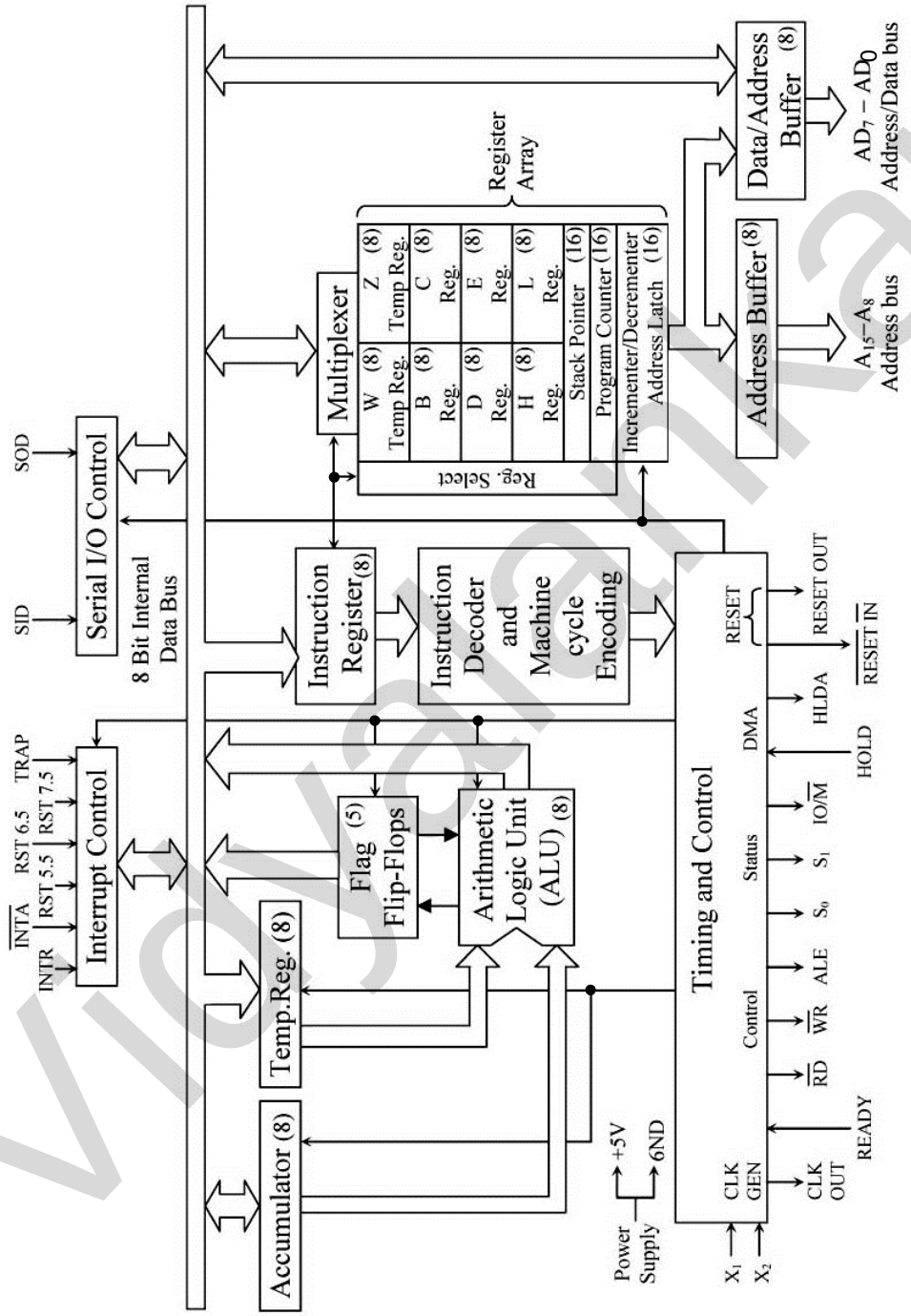
(4) **ORG – Originate :**
- As assembler assembles a section of data declaration or instruction statements, it uses location counter to keep track of how many bytes it is from the start of a segment at any time.
- The location counter is automatically set to 0000 when assembler starts reading a segment.

1. (b) (iii) Differentiate between NEAR and FAR CALLs

	NEAR CALL	FAR CALL
1)	It is also called as Intra-segment call.	It is also called as Inter-segment call.
2)	It replace old TP with new IP.	It replaces old CS: IP pairs with new CS:IP pairs
3)	Less stack memory locations are required.	More stack memory locations are required.
4)	In a program, if a destination location dies within a current segment called as NEAR CALL	In a program, if a destination location lies other than current segment or different segment called as FAR CALL.

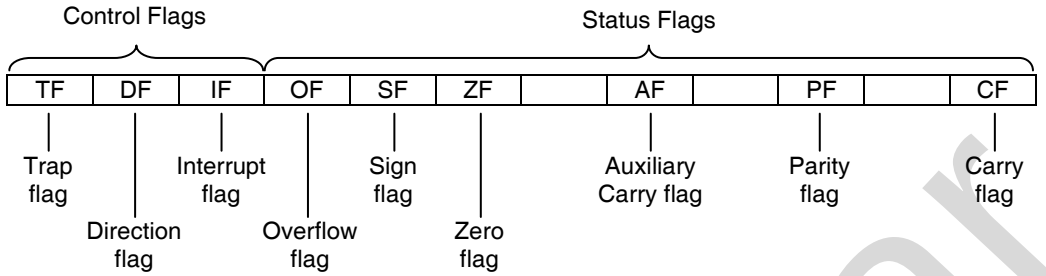
2. (a) Registers (Register organization) :

The registers present in 8085 can be classified in following groups –



Architecture of 8085

2. (b) Flag Register



8086 flags can be divided into two groups : Status flags and Control flags.

Status Flags :

Carry Flag (CF) **:** CF flag is set if there is a carry-out or borrow-in for the most significant bit of the result during the execution of an instruction. Otherwise, CF is reset.

Parity Flag (PF) **:** If the result of instruction execution contain even number of 1's, this flag is set, otherwise reset.

Auxiliary Carry Flag (AF) **:** If carry is generated from lower nibble to upper nibble or borrow is taken by lower nibble from upper nibble, this flag is set. This flag is used internally by processor in BCD operations.

Zero Flag (ZF) **:** This flag is set if result produced by instruction execution is zero, otherwise rest.

Sign-Flag (SF) **:** The MSB of result is copied in SF. Thus SF is set if result is negative and reset if result is positive.

Overflow Flag (OF) **:** When OF is set, it indicates that the signed result is out of range. If the result is not out of range, OF remains reset.

2. (c)

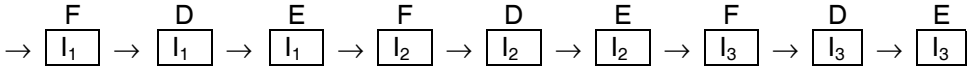
Queue is a 6 byte FIFO (FIRST IN FIRST OUT) RAM used to implement pipelining.

Bus Interface Unit fetches next instruction from code segment and placed it into queue.

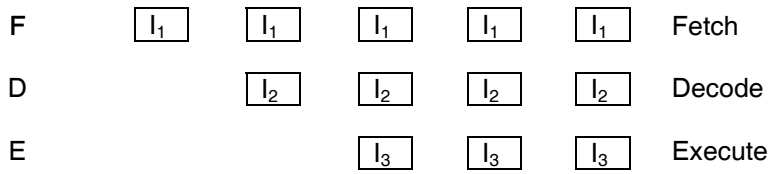
Fetching the next instruction while execution of current instruction is called pipelining.

At least two byte of queue should be empty status of a queue observed by QS_0 and QS_1 signals.

In 8085 (non-Pipelined Processor) nine clock cycles are required for execution of three instruction i.e. fetch, decode and execute of each instruction.



In 8086 (Pipelined Processor) only five clock cycles are required to execute three instruction. Because by using pipeling three instructions are executed in parallel.



In fetching Stage, it reads opcode from program memory.
In decoding Stage, it checks what operation performed by processor and it generated Control Word.

In executing, stage it does not execution of instruction.
To speed up the Processor. (80486 and Pentium)

- Prefetch instruction (PF)
- Decode instruction (D_1) Generate control word
- Decode instruction (D_2) Generate memory addresses.
- Execute Instruction (EX)
- Write Back result (WB)

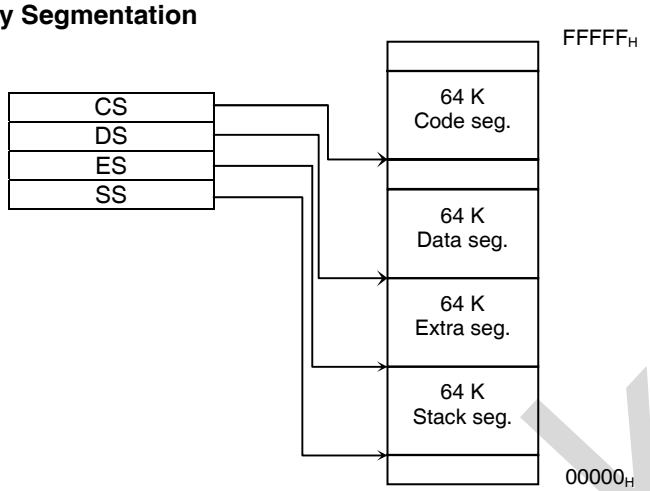
2. (d) Limitations (Drawbacks) of 8 bit microprocessors (8085) :

- The addressing capacity is only 64KB which is not sufficient for modern day applications.
- The 8 bit data can be processed at a time. To process data more than 8 bit, multiple set of instructions are required.
- It has non-pipelined architecture. Therefore the total execution time for an instruction will be addition of fetch time and execute time. This reduces the execution speed.
- There are only 7 general purpose registers. Hence the memory is accessed frequently which requires time.
- The instruction set is limited. No instructions for frequently done operations like multiplication and division.

2. (e)

```
        MOV AX, 05H
        MOV CX, AX
BACK :  DEC CX
          MUL CX
            loop : BACK
          MOV [D000], AX
        HLT
```

2. (f) Memory Segmentation



- 8086 has 20 address lines while all registers in 8086 are 16 bit in length.
- To get the 20 bit physical address, content of segment registers and offset is used.
- The content of segment register gives segment base address.
- The physical address is calculated by appending 4 zero's in binary to the right of segment base address and then adding offset to it.
- The offset is of 16 bit. Hence size of each segment is 64 KB.
- The four segments code segment, data segment, extra segment and data segment will be active at a time.
- Code segment is used to store program code. Data segment is used to store data required for program execution. Extra segment is additional data segment if data segment is not sufficient. Stack segment is used to implement stack.

3. (a) (i) XCHG

Mnemonic	Meaning	Format	Operation	Flags affected
XCHG	Exchange	XCHG,S	(D) $\leftrightarrow$ (S)	None

Execution : When this instruction is executed contents of destination (D) and source are exchanged.

Allowed operands are

Destination	Source
Register	Register
Memory	Register
Register	Memory

e.g. XCHG AX, DX
After execution
Original content of AX $\rightarrow$ (DX)
Original content of DX $\rightarrow$ (AX)

(ii) CMP (Compare Instruction)

Mnemonic	Meaning	Format	Operation	Flags affected
CMP	Compare	CMP D, S	(D) – (S) is used in setting or resetting the flags	CF, AF, OF, PF, SF, ZF.

Allowed operands

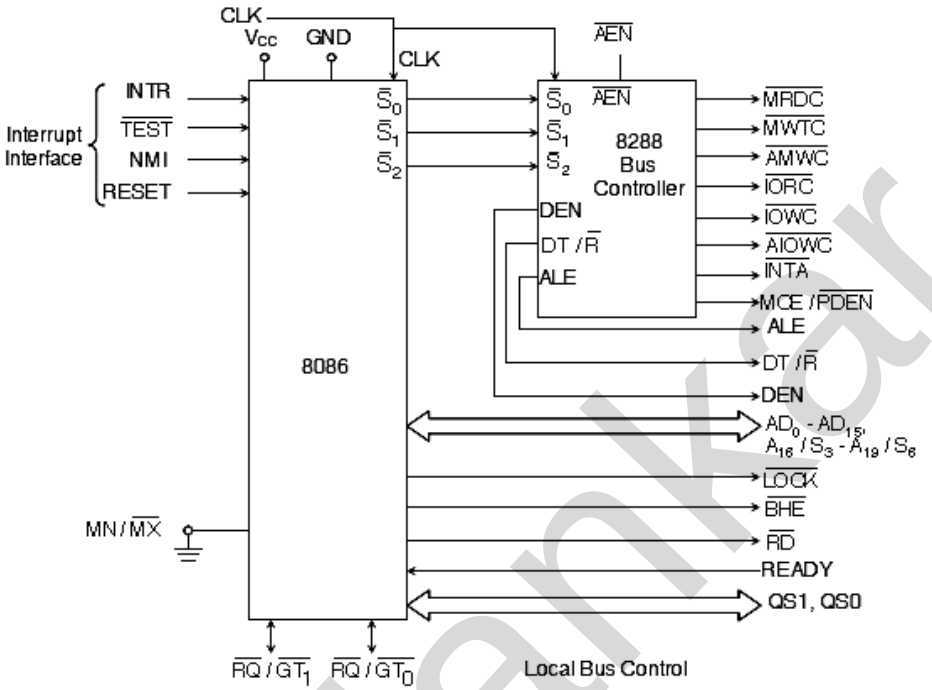
Destination	Source
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate

Execution : When CMP D, S instruction is executed, the content of source are subtracted from the content of source, but result of subtraction is not stored anywhere. Instead flags are affected as per the result of this subtraction. CF and ZF can be used to decide on comparison.

Condition	CF	ZF
D > S	0	0
D < S	1	0
D = S	0	1

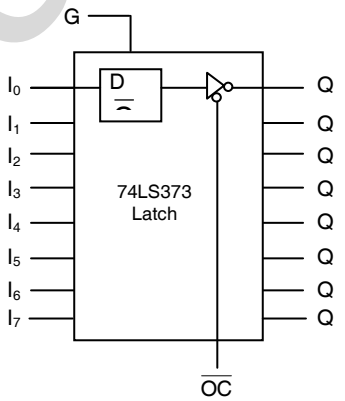
3. (b) Maximum Mode

- When $MN/\overline{MX}$ is connected to ground, 8086 will work in maximum mode.
- 8086 is used in maximum mode for medium to large systems. It is generally multiprocessor system.
- More than one 8086 can be connected. Also co-processor like I/O processor or numeric coprocessor can be connected.
- All signals necessary for interfacing memory or I/O device are not generated by 8086 in maximum mode.
- Status signals $\overline{S}_0 - \overline{S}_2$ are used to generate the signals necessary for interfacing using bus controller 8288.



Status signals			8288 Signals Active
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	
0	0	0	$\overline{INTA}$
0	0	1	$\overline{IORC}$
0	1	0	$\overline{IOWC}$, $\overline{AIOWC}$
0	1	1	none
1	0	0	$\overline{MRDC}$
1	0	1	$\overline{MRDC}$
1	1	0	$\overline{MWTC}$, $\overline{AMWC}$
1	1	1	none

3. (c) Octal Latch



The 74LS373 Latch contains 8 D-Type latches. When enable G is HIGH Q output follows the D input.
When enable G is low Q output will be latched at what data is send.
The D flip flop controlled by G input and invertor is controlled by $\overline{OC}$ pin.
This latch is used to address and data lines from multiplexed $AD_0 - AD_{15}$.

3. (d) Conditional Jump

Mnemonic	Meaning	Format	Operation	Flags affected
J_{CC}	Conditional jump	J_{CC} operand	If the specified condition $_{CC}$ is true the jump to the address specified by the operand is initiated; otherwise the next instruction is executed.	None

Allowed operands

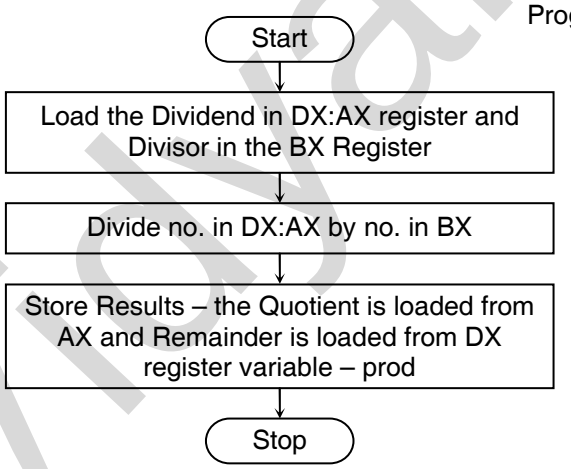
Operands
Short–label
Near–label
Far–label
Memptr 16
Regptr 16
Memptr 32

- The branching takes place if condition code (cc) is satisfied otherwise next sequential instruction will be executed.
- The meaning of operands is same as that for unconditional branching.
  The types of conditional branch instructions are

Mnemonic	Meaning	Condition
JA	Jump if above	CF = 0 and ZF = 0
JAE	Jump if above or equal	CF = 0
JB	Jump if below	CF = 1
JBE	Jump if below or equal	CF = 1 or ZF = 1
JC	Jump if carry	CF = 1
JCXZ	Jump if CX register is zero	(CF or ZF) = 0
JE	Jump if equal	ZF = 1
JG	Jump if greater	ZF = 0 and SF = OF
JGE	Jump if greater or equal	SF = OF
JL	Jump if less	(SF XOR OF) = 1
JLE	Jump if less or equal	((SF XOR OF) or ZF) = 1
JNA	Jump if not above	CF = 1 or ZF = 1
JNAE	Jump if not above nor equal	CF = 0

Mnemonic	Meaning	Condition
JNB	Jump if not below	CF = 0
JNBE	Jump if not below nor equal	CF = 0 and ZF = 0
JNC	Jump if not carry	CF = 0
JNE	Jump if not equal	ZF = 0
JNG	Jump if not greater	((SF XOR OF) or ZF) = 1
JNGE	Jump if not greater nor equal	(SF XOR OF) = 1
JNL	Jump if not less	SF = OF
JNLE	Jump if not less nor equal	ZF = 0 and SF = OF
JNO	Jump if not overflow	OF = 0
JNP	Jump if not parity	PF = 0
JNS	Jump if not sign	SF = 0
JNZ	Jump if not zero	ZF = 0
JO	Jump if overflow	OF = 1
JP	Jump if parity	PF = 1
JPE	Jump if parity even	PF = 1
JPO	Jump if parity odd	PF = 0
JS	Jump if sign	SF = 1
JZ	Jump if zero	ZF = 1

3. (e) Flowchart



Program :

```
.model small
.data
    num1 dd 000000A5h
    num2 dw 0005h
   Quotient dw ?
    Remainder dw ?
.code
.startup

main:mov ax, @data
    mov ds, ax
    lea si, num1
    mov ax, [si]
    mov dx, [si+2]
    mov bx, num2
    div bx
    mov Quotient, ax
    mov Remainder, dx
.exit
end
```

3. (f) Based-Indexed Addressing Mode

When all components namely base, index and displacement of effective address calculation are present, addressing.

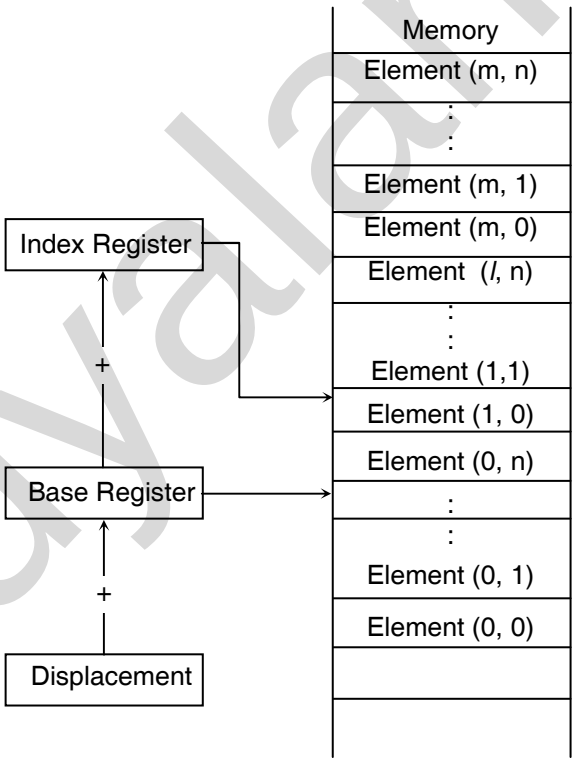
$$PA = SBA : EA$$

$$= SBA : Base + Index + Displacement$$

$$= \begin{Bmatrix} CS \\ DS \\ ES \\ SS \end{Bmatrix} : \begin{Bmatrix} BX \\ BP \end{Bmatrix} + \begin{Bmatrix} 8 \text{ bit displacement} \\ 16 \text{ bit displacement} \end{Bmatrix}$$

e.g. MOV AX, [BX] [SI] + 0300H. The offset is the addition of content of BX, SI and displacement 0300H. The data from memory location at this offset in current data segment is copied in register AX.

This addressing mode is used to access multidimensional array.



By changing content of base register, we can select one of may single dimensional array and by changing the content of index register we can access different locations from selected one dimensional array.

4. (a) (i) MOV CL, S4H – Immediate Addressing Mode.
(ii) MOV BX, [4172H] – Direct Addressing Mode
(iii) MOV DS, AX – Register addressing Mode.
(iv) MOV AX, {SI + AX + 04] – Relative Based Indexed Addressing Mode.

4. (b)
- For this AX entire register is first initialized to 0000h and then we go on comparing 8-bit numbers in AL register.
- Every time if the next number is larger, it is replaced in AL register; making always the biggest number in AL; up to any point of comparison.

Program : (The Flow chart is on the next page)

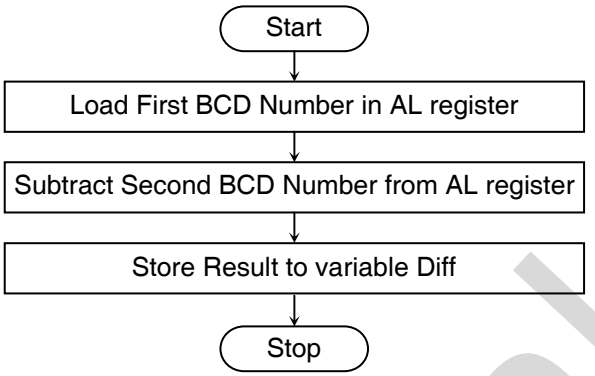
```
.model small
.data
      nums dw 4444h, 5555h, 2222h, 7777h, 1111h, 3333h, 8888h, 6666h
     count dw 0008h
   largest dw ?

.code
.startup
start:  mov ax, @data
        mov ds, ax
        mov ax, 0000h
        mov cx, count
        mov bx, offset nums
again:  cmp ax, [bx]
        jnc skip
        mov ax, [bx]
skip:   inc bx
        inc bx
        loop again
        mov largest, ax
.exit
end
```

4. (c) Program :

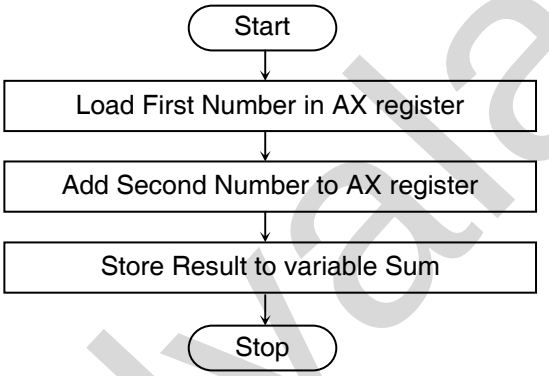
```
.model small
.data
   num1 db 81h
   num2 db 57h
   diff db ?
.code
.startup
main:mov ax, @data
   mov ds, ax
   mov al, num1
   sub al, num2
   DAS     ;Decimal Adjust for
            Subtraction, the
            result in AL register
```

```
    mov diff, al
.exit
end
```

Flowchart:



4. (d) Flowchart :



Program :

```
.model small
.data
    num1 dw 6666h
    num2 dw 4444h
    sum dw ?
.code
.startup
main:mov ax, @data
    mov ds, ax
    mov ax, num1
    add ax, num2
    mov sum, ax
.exit
end
```

4. (e) PUSH and POP Instructions

- As the number of registers available to the programmer are limited, they may be used in main program and subroutines. In some cases the value present in them need to be saved before transferring program control to subordinate so that when program control is transferred back to calling program after execution of subordinate, original values can be restored. This function is performed by PUSH and POP instruction and stack memory is used for storing the values of registers.

Mnemonic	Meaning	Format	Operation	Flags affected
PUSH	Push word onto stack	PUSH S	((SP)) ← (S)	None
POP	Pop word off stack	POP D	(D) ← ((SP)) (SP) ← (SP) + 2	None

Allowed operands

Operands (S or D)
Register
Seg – reg (CS illegal)
Memory

- When PUSH S instruction is executed, content of S (16 bit) are stored on top of the stack and stack pointer is decremented by 2

e.g. PUSH AX

((SP) – 1) ← (AH)
((SP) – 2) ← (AL)
(SP) ← (SP) – 2

- When POP D instruction is executed, content of two locations at the top of the stack are loaded into D and stack pointer is incremented by 2

(AL) ← ((SP))
(AH) ← ((SP) + 1)
(SP) ← (SP) + 2

} When POP AX instruction is executed

PUSH and POPF Instructions

Mnemonic	Meaning	Operation	Flags affected
PUSHF	Push flags onto stack	((SP)) ← (Flags) (SP) ← (SP) – 2	None
POPF	Pop flags from stack	(Flags) ← ((SP)) (SP) ← (SP) + 2	OF, DF, IF, TF, SF, ZF, AF, PF, CF

- PUSHF instruction is used to store the content of flag register on top of the stack.
- POPF instruction is used to load flag register from the top of the stack.

4. (f) Passing Parameters through Pointers

- Program stores the parameters to the memory and passes pointers (address of) to the subroutine/ procedure.
- Subroutine/Procedures accesses memory using the pointers and uses the parameters from there.

Multiplication Example :

```
• MODEL SMALL
• DATA
  NUM1 DB 42H
  NUM2 DB 23H
  RESULT DW ?
• CODE
  MOV AX, @ DATA
  MOV DS, AX
    ⋮
  MOV BV, OFFSET RESULT
  MOV SI, OFFSET NUM1
  CALL MULTI
    ⋮
```

```
MULTI :  PROC NEAR
         MOV AL, [SI]
         MOV CL, [SI + 01H]
         MUL CL
         MOV [BX], AX
MULTI :  ENDP
         END
```

5. (a) Total sum of series

Dosseg

- Model small
- stack 100h
- data

```
List db 12, 34, 56, 78, 98, 01, 13, 78, 18, 36
Total dw?
```

- Code

```
main proc
MOV AX, data
MOV DS, AX
MOV AX, 0000H
MOV CX, 0AH; Counter
MOV BL, 00H; to Count array
MOV S1, offset–List

Back : ADD AL, [S1]
IC label
Back 1 :   INC S1
           loop = Back
           MOV Total, AX
           MOV Total + 2, BL
           MOV AH, UCH
           INT21H
Label : INC BL
JMP Back1
Main endP
End Main
```

5. (b)

```
        MOV   BL, 00H
        MOV   CL, 05H
Loop1:  Add   BL, 02H
        Dec   CL
        JNZ   Loop1
```

Loop1 will be executed 5 times in the program. The content of BL is 0AH.

5. (c) Title reverse given string

Dosseg

- model small
- stack 100h
- data

string 1db, assembly language program, $ length dw $ – string 1 – 1

- Code

```
Main proc
MOV AX, 0 data
MOV DS, AX
MOV S1, offset string 1
MOV CX, length
AOP S1, CX
BACK : MOV DL, [s1]
       MOV AH, 02H
       INT 21H
       DEC S1
       loop Back
       MOV AH, 4CH
       INT 21H
       Main endP
       End Main
```

5. (d) (i) Multiply AL register contents by 4 using shift instruction.

```
MOV   AL, num1
MOV   CL, 02H
SHL   AL, CL
```

(ii) Move 1234H into DS register.

```
MOV   ax,  1234H
MOV   DS, ax
```

5. (e) String Handling Instructions:

- A series of data words or bytes that reside in consecutive memory locations is called string.
- There are five basic instructions related to string in instruction related to string in instruction set of 8086.

Mnemonic	Meaning	Format	Operation	Flags affected
MOVS	Move string	MOVSB / MOVSW	((ES)0 + (DI)) ← ((DS) 0 + (SI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	None
CMPS	Compare String	CMPSB / CMPSW	Set flags as per ((DS)0 + (SI)) – ((ES)0 + (DI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
SCAS	Scan String	SCASB / SCASW	Set flags as per (AL or AX) – ((ES)0 + (DI)) (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
LODS	Load String	LODSB / LODSW	(AL or AX) ← ((DS) 0 + (SI)) (SI) ← (SI) ± 1 or 2	None
STOS	Store String	STOSB / STOSW	((ES)0 + (DI)) ← (AL or AX) (DI) ← (DI) ± 1 or 2	None

- In general source is memory location pointed by DS and SI while destination is memory location by pointed by ES and DI.
- 'B' in instruction specifies byte operation and 'W' specifies word.
- The source and destination pointer are automatically updated when such instruction is executed. They are updated by 1 if operand is of byte size and by 2 if operand is of word size.
- The operations on string can be done in incrementing order of address or decrementing order of address. If direction flag is set, then source and destination pointer will be decremented by either 1 or 2. If direction flag is reset, then source and destination pointer will be incremented by their 1 or 2.

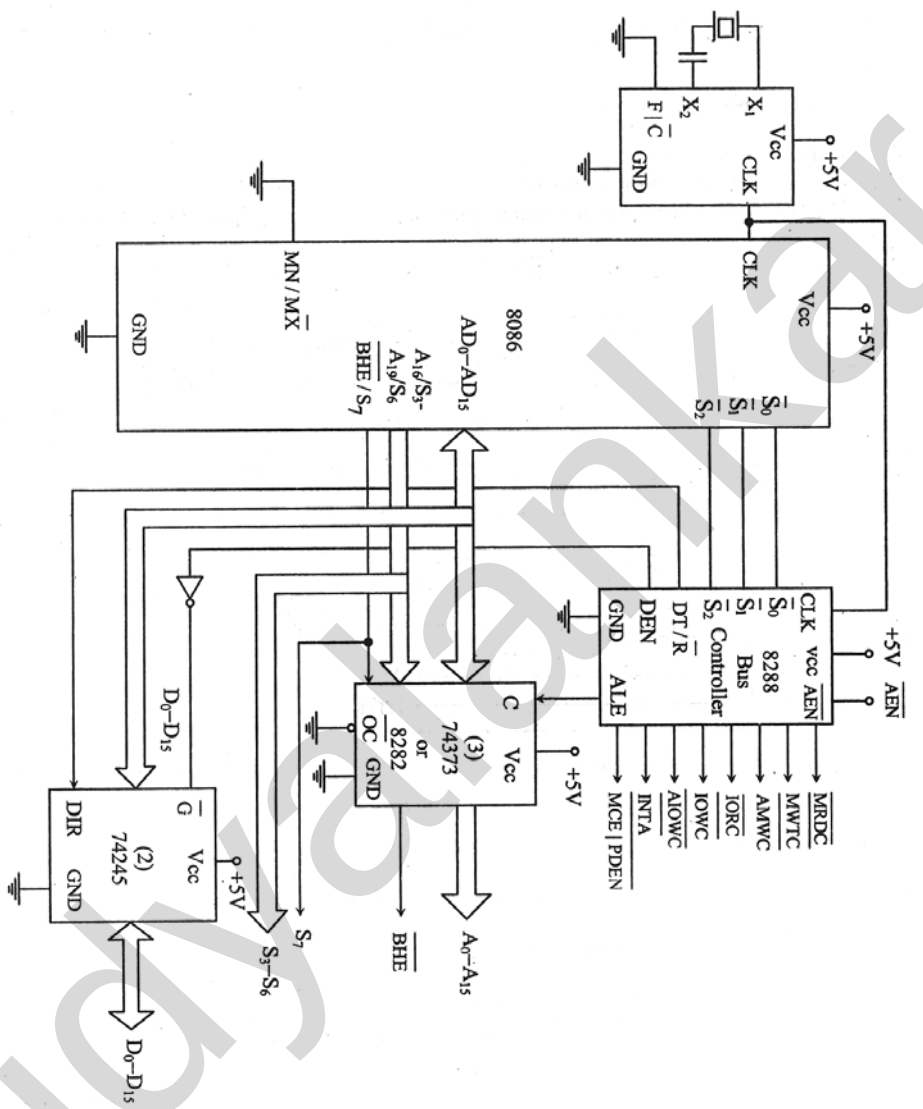
5. (f) (i) Linker

- Linker is a program used to join several object files into one large object file
- When writing large programs, it is usually much more efficient to divide the large program into smaller modules. Each module can be individually written, tested, and debugged. Then, when all the modules work, their object modules can be linked together to form a large, functioning program.
- Also the object modules for useful programs like square root program can be kept in library file and linked into other program as needed.
- The linker produces a link file which contain the binary codes for all the combined modules.
- The linker also produces a link map file which contains the address information about the linked files. The linker assigns only relative address to the program starting from zero. This form of program is said to be relocatable because it can be put anywhere in memory to be run.
- The linkers which come with the TASM or MASM assemblers produce link with. EXE extension.

(ii) Debugger

- If the program does not require any external hardware, debugger is used to load the .EXE file in main memory and run it.
- It automatically assigns physical starting address to the segments.
- Debugging is also possible with this program. By executing instructions stepwise, you can check the content of registers and memory locations and check your steps in problem solving logically.

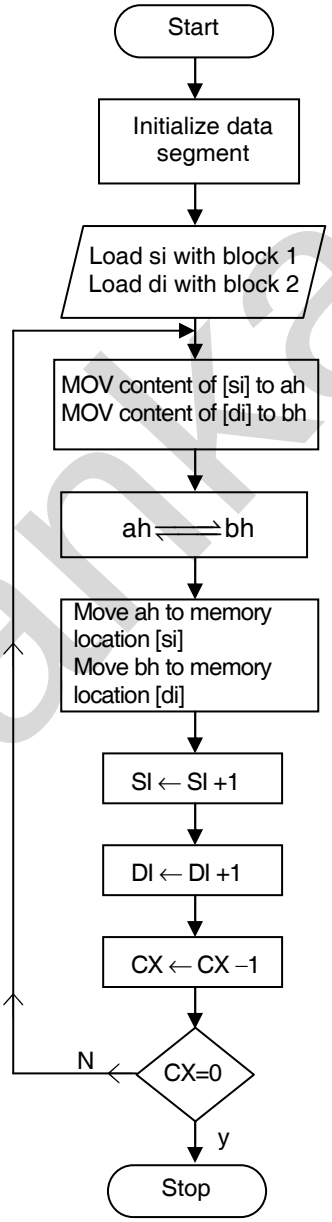
6. (a) Maximum mode of 8086 micro processor.



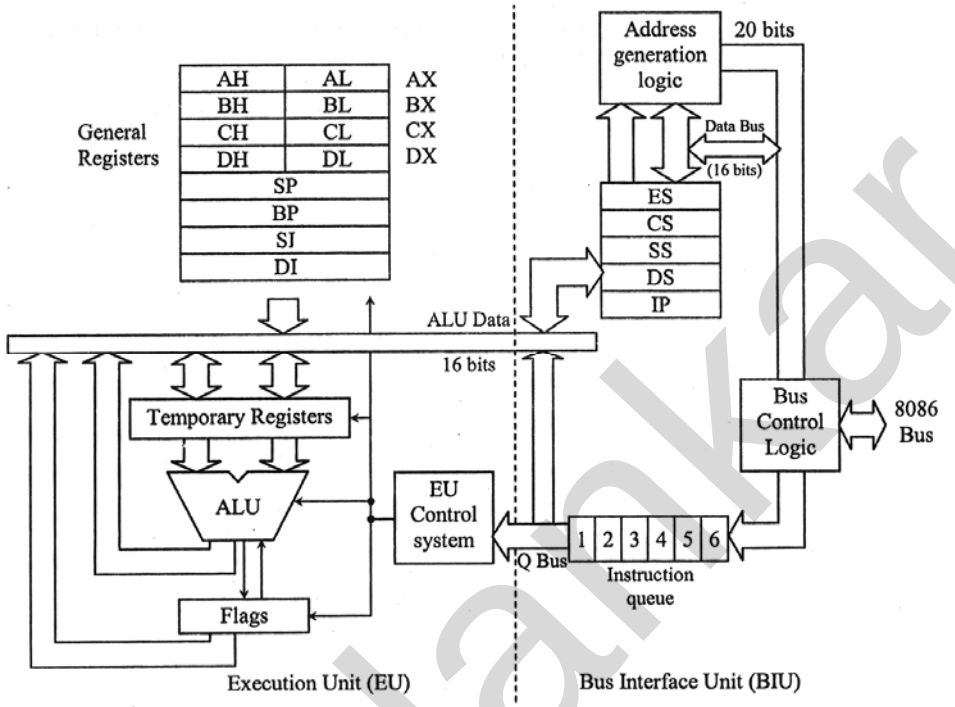
6. (b)

```
.model small
. data
 block1  db  01H, 02H, 03H, 04H, 05H
 block2  db  11H,12H, 13H, 14H, 15H
.code
 MOV    ax,  @data
 MOV    els, ax
 MOV    cx,  0005H
      Lea      si,   block1
      Lea      di,   block2
up:   MOV      ah,  [si]
      MOV      bh,  [di]
      xchg     ah,  bh
      MOV      [si], ah
      MOV      [di], bh
      inc      si
      inc      di
      dec      cx
      Jnz      ap
      int 3h
      Ends
      End.
```

Flow chart :



6. (c) Architecture of 8086 Microprocessor



- Architecture of 8086 is divided into two units namely bus interface unit (BIU) and execution unit (EU). They work independently.

(i) Bus Interface Unit (BIU) :

- Bus interface unit is responsible for interface of 8086 with external world.
- It fetches the instructions and data from external interface and stores the result in external interface whenever required.
- To access any location from memory, 20 bit physical address is required. But all registers in 8086 are 16 bit in length. Hence conversion of 16 bit address into 20 bit address is required. This process is done by bus interface unit.
- Whenever execution unit does not want data from memory for execution of its current instruction and bus interface unit is free, it prefetches the instructions from memory ahead of time and stores it in the queue. For this operation to happen, at least space of two bytes must be free in the queue. This operation is called prefetching. The instruction queue of 8086 is of 6 bytes and works on FIFO (First In First Out) principle. Its output end is accessed by execution unit.
- If execution unit does not require any data and queue is full, bus interface unit does not have to perform any function. This state is called idle state.

(ii) Execution Unit (EU) :

- The output end of instruction queue is accessed by execution unit.
- The instruction is decoded and the control signals are general in proper sequence for internal blocks to carry out the execution of the instruction.
- ALU (Arithmetic Logic Unit) is responsible for carrying out arithmetic operations like addition, subtraction, multiplication, division etc. and logical operations like AND, OR, NOT, EXOR, shift, rotate etc. Multiplication and division operations are implemented with barrel shifter.
- The result produced by operations in ALU is reflected in flag register. Flags show the status of result. Some flags are used to control some of the operations of processor.
- General purpose registers AX, BX, CX, DX, SP, BP, SI, DI are used for storing data as well as intermediate results. All these registers are 16 bit registers. Out of them AX, BX, CX, DX can be used as 8 bit registers as AL, AH, BL, BH, CL, CH, DL, DH.
- Registers CS, DS, ES, SS are used to hold segment base address.
- Register IP is used to hold the offset of next instruction to be fetched from current code segment.

❑ ❑ ❑ ❑ ❑