SERIES TEST 1

PART A - (Answer all questions. Each question carries 3 marks)

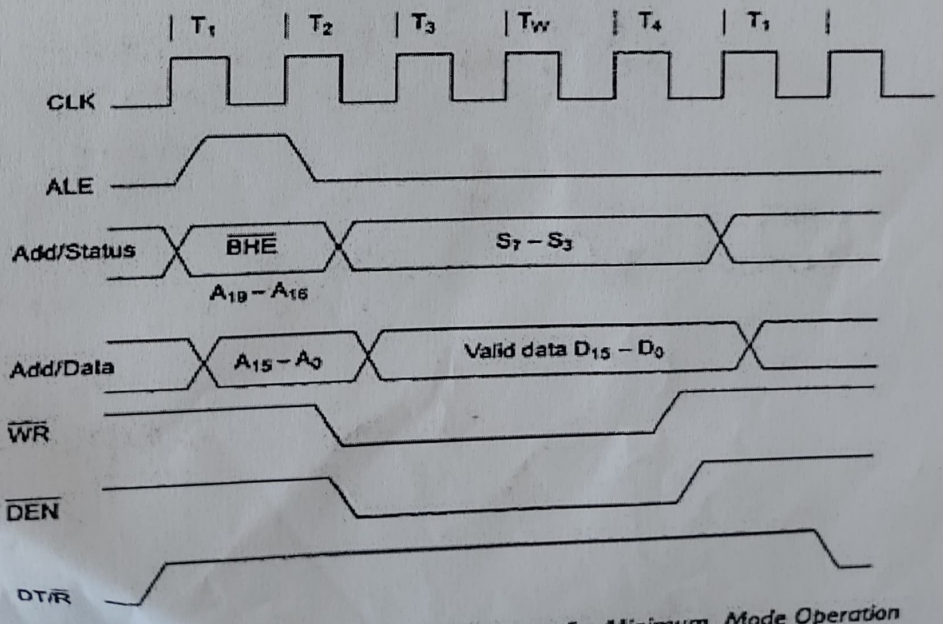
1. **What are the different information conveyed by the Queue status signals QS0 and QS1 of 8086 in maximum mode?**

   - These lines give information about the status of the code-prefetch queue. 1/2
   - These are active during the CLK cycle after while the queue operation is performed. 1/2
   - The 8086 architecture has 6-byte instruction prefetch queue

QS_1	QS_0	Status
0	0	No operation
0	1	1st byte of opcode from queue
1	0	Empty queue
1	1	Subsequent byte from queue

2

2. **Draw the timing diagram for the 8086 minimum mode memory write**



Write Cycle Timing Diagram for Minimum Mode Operation

name ½ x2
syntax
usage ½
(2)

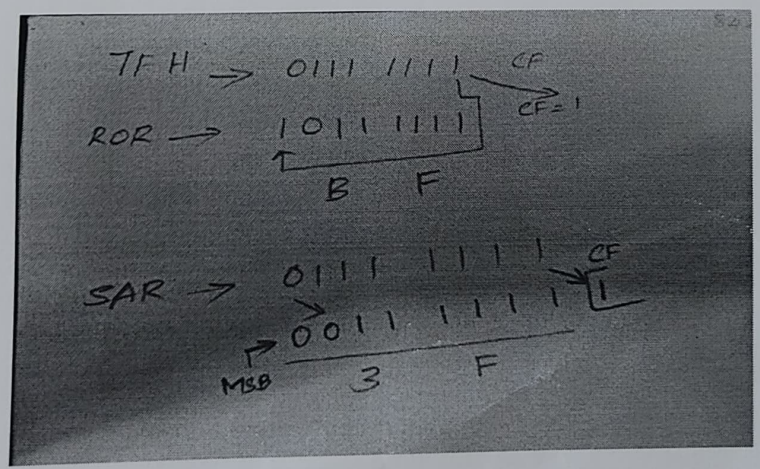
3. **What are assembler directives? List any two assembler directives and its usage.**

- An assembler is a program used to convert an assembly language program (ALP) into the equivalent machine code modules which may further be converted to executable codes. ½
- For completing all the tasks, an assembler needs to know • Required storage for a particular constant or a variable ½
    - Logical names of the segments
    - Types of the different routines and modules
    - End of file and so on
    - Assembler Directives are some predefined alphabetical strings that give these types of hints to the assembler
- They help the assembler to correctly understand the programs to prepare the codes.

4. **Assume AL register is having the value 7FH. What will be the content of AL after the following instructions are executed**

a) ROR AL,01 b) SAR AL,01

ROR – Used to rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF]



Rotate (1)
Convert & AL content is (½)

shift (1)
— (½)

SAR – Used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB

5. **Describe any three control transfer instructions in 8086.**

Unconditional branch instruction

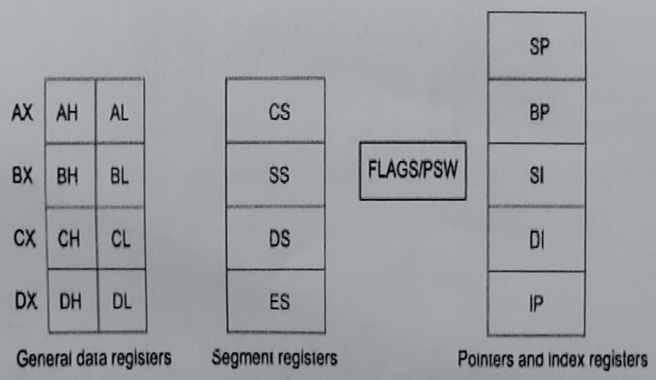
- **CALL** – Used to call a procedure and save their return address to the stack.
- **RET** – Used to return from the procedure to the main program.
- **JMP** – Used to jump to the provided address to proceed to the next instruction.

Conditional branch instruction

- **JA/JNBE** – Used to jump if CF=0 or ZF=0
- **JAE/JNB** – Used to jump if CF=0
- **JBE/JNA** – Used to jump if CF=1 or ZF=1
- **JC** – Used to jump if carry flag CF = 1
- **JE/JZ** – Used to jump if flag ZF = 1
- **JG/JNLE** – Used to jump if ZF=0 or any one of SF and OF is 1.
- **JGE/JNL** – Used to jump if greater than/equal/not less than instruction satisfies.
- **JL/JNGE** – Used to jump if less than/not greater than/equal instruction satisfies.
- **JLE/JNG** – Used to jump if less than/equal/if not greater than instruction satisfies.
- **JNC** – Used to jump if no carry flag (CF = 0)
- **JNE/JNZ** – Used to jump if not equal/zero flag ZF = 0
- **JNO** – Used to jump if no overflow flag OF = 0
- **JNP/JPO** – Used to jump if not parity/parity odd PF = 0
- **JNS** – Used to jump if not sign SF = 0
- **JO** – Used to jump if overflow flag OF = 1
- **JP/JPE** – Used to jump if parity/parity even PF = 1
- **JS** – Used to jump if sign flag SF = 1

PART B

6. **Explain register set of 8086** (7 marks)

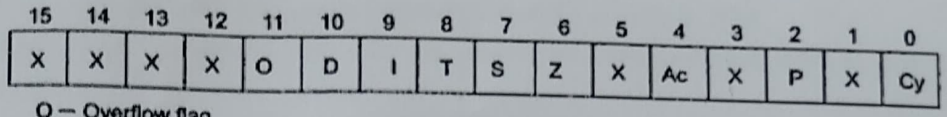


(1)

exp 1 (1)

1 (1)

1 (1)



O – Overflow flag
D – Direction flag
I – Interrupt flag
T – Trap flag
S – Sign flag
Z – Zero flag
Ac – Auxiliary carry flag
P – Parity flag
Cy – Carry flag
X – Not used

(2)
(1)

OR

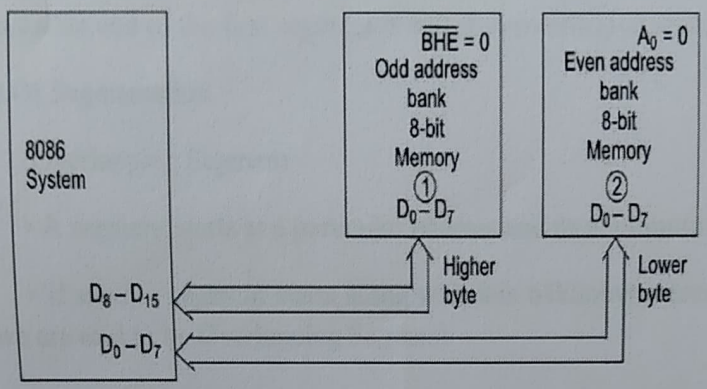
7. **Write an 8086 assembly language program to find the sum of a series of 8 bit numbers. The series contains 50 numbers.** **(7 marks)**

```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT                    ; Data segment starts
NUMLIST DB 52H, 23H,-           ; List of byte numbers
COUNT EQU 100D                  ; Number of bytes to be added
RESULT DW 01H DUP(?)            ; One word is reserved for result
DATA ENDS                       ; Data segment ends
CODE SEGMENT                        ; Code segment starts at relative
ORG 200H                            ; address 0200h in code segment
START:      MOV AX, DATA            ; Initialize data segment
            MOV DS, AX
            MOV CX, COUNT           ; Number of bytes to be added in CX
            XOR AX, AX              ; Clear AX and CF
            XOR BX, BX              ; Clear BH for converting the byte to
                                      word
            MOV SI,OFFSET NUMLIST   ; Point  to  the first number in the
                                      list
AGAIN:      MOV BL, [SI]            ; Take the first number in BL,BH is zero
            ADD AX, BX              ; Add AX with BX
            INC SI                  ; Increment pointer to the byte list
            DEC CX                  ; Decrement counter
            JNZ AGAIN               ; If all numbers are added,point to re-
                                      sult
            MOV DI, OFFSET RESULT   ; destination and store it
            MOV [DI], AX
            MOV AH, 4CH             ; Return to DOS
            INT 21H
            CODE ENDS
END         START
```

8. **a) With a neat diagram describe the physical memory organization of 8086. (6 marks)**

- The 1MB memory is physically organized as odd bank and even bank, each of 512kbytes, addressed in parallel by the processor. 1/2
- Byte data with even address is transferred on D7- D0 and byte data with odd address is transferred on D15-D8 . 1/2
- The processor provides two enable signals **BHE and A0** for selecting of either even or odd or both the banks. 1/2

- D15-D0/ A15-A0 are the common signal line in 8086 as AD15-AD0
- Memory chips are only 1 Byte size i.e., they can store only one byte in one memory location.
- To store 16 bit data, two successive memory locations are used
- The lower byte of 16 bit data can be stored in first memory location while the second byte is stored in next location.
- In a 16 bit read or write operation both of these bytes will be read or written in a single machine cycle.
- Thus the complete memory map of 8086 system is divided into even and odd address memory banks



BHE'	A0	Function
0	0	Whole word
0	1	Upper byte/ odd address
1	0	Lower byte/even address
1	1	none

If the address bus contains $FFFFO_H$

If $\overline{BHE} = 0$, then it reads the 16 bits data from memory location $FFFFO_H$ and $FFFF1_H$.

If $\overline{BHE} = 1$, it reads the 8 bits data from memory location $FFFFO_H$.

Fig shows the physical memory organisation of 8086.

b) Specify the significance of segmentation, how it is implemented in 8086 and its advantages. (8 marks)

The total memory size is divided into segments of various sizes. A segment is just an area in memory. The 8086 can access memory with address ranging from 00000 H to FFFFFH. Each Segment register stores the base address (starting address) of the corresponding segment.

• Segment registers only store the upper 16 bits. 20-bit address is obtained from 16- bit registers. • The 20-bit address of a byte is called its Physical Address.

- Logical address is in the form of: Base Address : Offset
- Offset is the displacement of the memory location from the starting location of the segment.
- Eg: The value of Data Segment Register (DS) is 2222 H.
- The BIU appends 0H to the LSBs of the address.
- BIU uses the following formula: Effective Address = Starting Address of Segment + Offset. All offsets are limited to 16-bits. The maximum size possible for segment is 216 = 65,535 bytes (64 KB). The area of memory from the start of the second segment to the possible end of the first segment is called as overlapped segment.

Types Of Segmentation

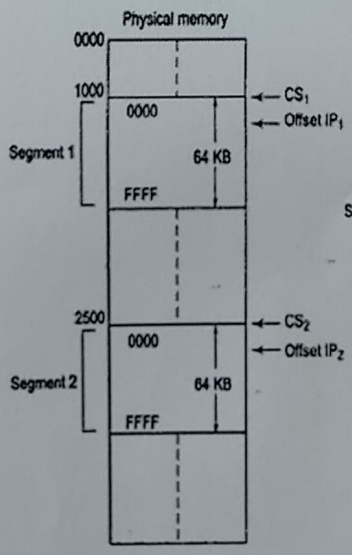
1. Overlapping Segment

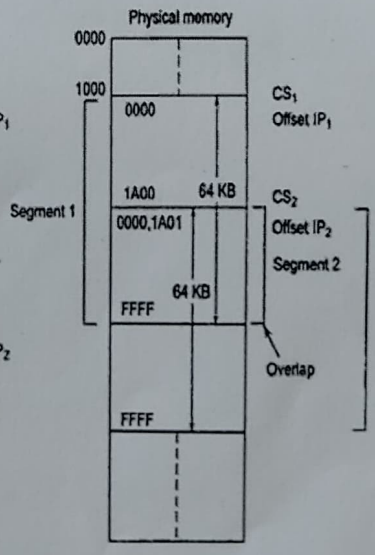
- A segment starts at a particular address and its maximum size can go up to 64kilobytes.
- if another segment starts along with this 64kilobytes location of the first segment, then the two are said to be Overlapping Segment

2. Non-Overlapping Segment

- A segment starts at a particular address and its maximum size can go up to 64kilobytes.

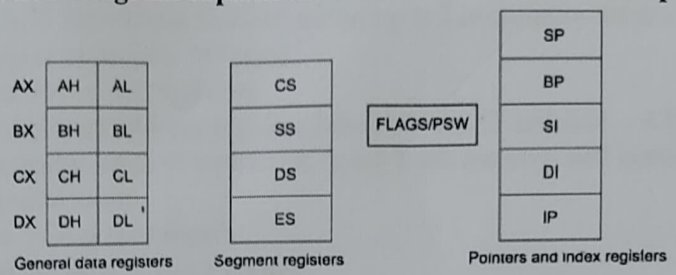
If another segment starts after this 64kilobytes location of the first segment, then the two segments are said to be Non-Overlapped Segment. The main advantages of the segmented memory scheme are as follows: a) Allows the memory capacity to be 1 MB although the actual addresses to be handled are of 16-bit size. b) Allows the placing of code data and stack portions of the same program in different parts (segments) of memory, for data and code protection. c) Permits a program and/ or its data to be put into different areas of memory each time program is executed, i.e., relocation may be done.

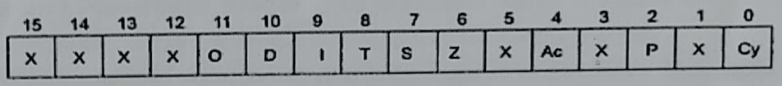




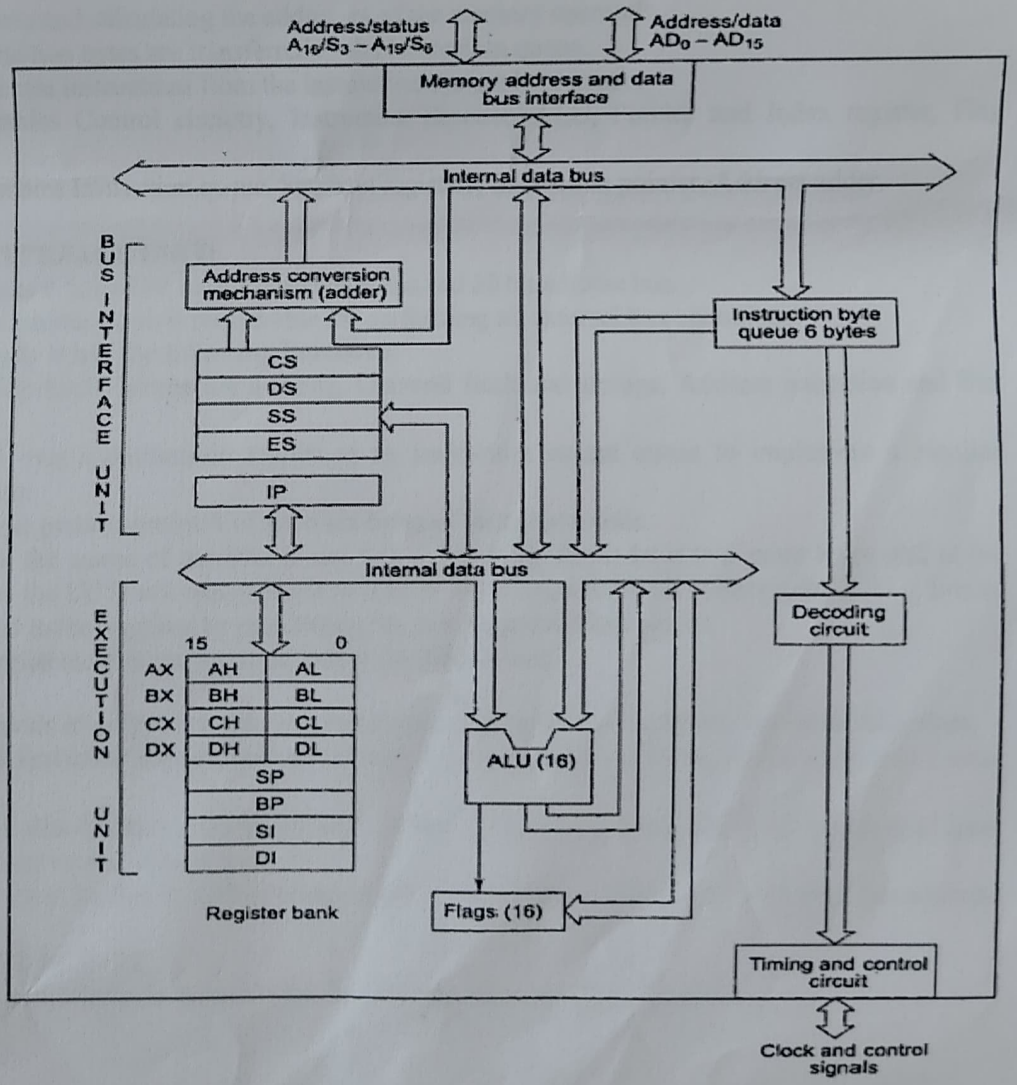
OR

9. With neat diagram explain the architecture of 8086 microprocessor. (14 marks)





O — Overflow flag
D — Direction flag
I — Interrupt flag
T — Trap flag
S — Sign flag
Z — Zero flag
Ac — Auxiliary carry flag
P — Parity flag
Cy — Carry flag
X — Not used



(4)

•It is a 16-bit µp. ½
•8086 has a 20 bit address bus can access up to 220 memory locations (1 MB). ½
•It can support up to 64K I/O ports.
•It provides 14, 16 -bit registers.
•It has multiplexed address and data bus AD0- AD15 and A16 – A19. ½
•It can prefetches upto 6 instruction bytes from memory and queues them in order to speed up instruction execution. ½
•It requires +5V power supply.
•A 40 pin dual in line package

Minimum and Maximum Modes: The minimum mode is selected by applying logic 1 to the MN / MX input pin. This is a single microprocessor configuration.
•The maximum mode is selected by applying logic 0 to the MN / MX input pin. This is a multi-microprocessors configuration.

Internal Architecture of 8086
•8086 has two blocks BIU and EU.
•The BIU performs all bus operations such as instruction fetching, reading and writing operands for memory and calculating the addresses of the memory operands.
The instruction bytes are transferred to the instruction queue.
•EU executes instructions from the instruction system byte queue.
•EU contains Control circuitry, Instruction decoder, ALU, Pointer and Index register, Flag register.
•BIU contains Instruction queue, Segment registers, Instruction pointer, Address adder.

BUS INTERFACE UNIT:
•It provides a full 16 bit bidirectional data bus and 20 bit address bus.
•The bus interface unit is responsible for performing all external bus operations.
Specifically it has the following functions:
•Instruction fetch, Instruction queuing, Operand fetch and storage, Address relocation and Bus control.
The BIU uses a mechanism known as an instruction stream queue to implement a pipeline architecture.
•This queue permits prefetch of up to six bytes of instruction code. 4
Whenever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by prefetching the next sequential instruction.
•These prefetching instructions are held in its FIFO queue.

•The intervals of no bus activity, which may occur between bus cycles are known as **Idle state**.
•The BIU first completes the instruction fetch bus cycle before initiating the operand read / write cycle.
•The BIU also contains a dedicated adder which is used to generate the 20bit physical address that is output on the address bus.
This address is formed by adding an appended 16 bit segment address and a 16 bit offset address.

EXECUTION UNIT
The Execution unit is responsible for decoding and executing all instructions. 4

•The EU extracts instructions from the top of the queue in the BIU, decodes them, generates operands if necessary, passes them to the BIU and requests it to perform the read or write cycles to memory or I/O and perform the operation specified by the instruction on the operands.
•During the execution of the instruction, the EU tests the status and control flags and updates them based on the results of executing the instruction.
•If the queue is empty, the EU waits for the next instruction byte to be fetched and shifted to top of the queue.

10. Find the physical address of the memory locations referred by the following instructions, when [AX]-1000H, [BX]-2000H, [SI]-3000H, [DI]-4000H, [BP]-5000H, [SP]-6000H, [CS]-0000H, [DS]-1000H, [SS]-2000H, [IP]-7000H.

i. **MOV AX, [5000]** (1) Base address – 1000H (1)

ii. **MOV AX, [BX]** (1)

iii. **MOV AX, 5000[BX]** (2) Specify all add (offset, etc) (1) calcul (1)

iv. **MOV AX,[BX][SI]** (2)

v. **MOV AX, 5000[BX][SI]** (2)

(9 marks)

(i) Direct addressing mode

MOV AX, [5000H]

DS:OFFSET ⇔ 1000H: 5000H
10H* DS ⇔ 10000
Offset ⇔ + 5000

15000H - Effective address

(ii) Register indirect

MOV AX, [BX]

DS:BX ⇔ 1000H:2000H
10H*DS ⇔ 10000
[BX] ⇔ + 2000

12000H - Effective address

(iii) Register relative

MOV AX, 5000 [BX]

DS: [5000 + BX]
10H*DS ⇔ 10000
Offset ⇔ + 5000
[BX] ⇔ + 2000

17000H - Effective address

(iv) Based indexed

```
MOV AX, [BX] [SI]
```

DS:[BX + SI]
10H*DS ⇔ 10000
[BX] ⇔ + 2000
[SI] ⇔ + 3000
15000H - Effective address

(v) Relative based indexed

```
MOV AX, 5000 [BX] [SI]
```

DS: [BX + SI + 5000]
10H*DS ⇔ 10000
[BX] ⇔ + 2000
[SI] ⇔ + 3000
Offset ⇔ + 5000
1A000 - effective address

b) Explain various data transfer instructions in 8086? **(5 marks)**

Data Transfer Instruction :

Instruction to transfer a word (½) 1(½)

- MOV – Used to copy the byte or word from the provided source to the provided destination.
- PUSH – Used to put a word at the top of the stack.
- POP – Used to get a word from the top of the stack to the provided location.
- XCHG – Used to exchange the data from two locations.
- XLAT – Used to translate a byte in AL using a table in the memory

Instructions for input and output port transfer (1)

- IN – Used to read a byte or word from the provided port to the accumulator.
- OUT – Used to send out a byte or word from the accumulator to the provided port.

Instructions to transfer the address (1)

- LEA – Used to load the address of operand into the provided register.
- LDS – Used to load DS register and other provided register from the memory
- LES – Used to load ES register and other provided register from the memory.

Instructions to transfer flag registers (1 ½)

- LAHF – Used to load AH with the low byte of the flag register.
- SAHF – Used to store AH register to low byte of the flag register.
- PUSHF – Used to copy the flag register at the top of the stack.
- POPF – Used to copy a word at the top of the stack to the flag register.

11. a) Explain the addressing modes supported by 8086 with one example for each.(9 marks)

- Addressing mode indicates a way of locating data or operands 1/2
- It describes the types of operands and the way they are accessed for execution of instruction 1/2
- According to the flow of instruction execution, instructions may be classified as
  - Sequential control flow executions
  - Control transfer instructions

1. Sequential control flow executions
   - After execution of one instruction, control is transferred to the next instruction immediately after it
   - Egs: arithmetic, logical, data transfer, processor control instructions
2. Control transfer instructions
   - Transfer control to some predefined address or address specified in the instruction • Egs: JUMP, CALL, RET

Addressing modes for sequential control flow instructions are:

1. Immediate Addressing mode
2. Direct Addressing mode
3. Register Addressing mode
4. Register Indirect Addressing mode
5. Indexed Addressing mode
6. Register Relative Addressing mode
7. Based indexed Addressing mode
8. Relative based indexed Addressing mode

Any 8
each name 1/2
ex 1/2

1. IMMEDIATE ADDRESSING MODE
   - Immediate data is a part of the instruction

   eg: MOV AX, 0005H 0005H is the immediate data can be 8 or 16 bit
2. DIRECT ADDRESSING MODE
   - the 16 bit memory address(offset) or I/O address is directly specified in the instruction
   - Eg: MOV BX, [5000H]
   - ¬Data resides in a memory location 5000 away from the data segment
   - ¬square brackets around the 5000H denote the contents of the memory location

¬Effective address is calculated as left shift 4 times the content of DS + 5000 (offset)

¬ EA = 10H * [DS] + 5000H

¬ Where shifting a number 4 times is equal to multiplying it by 10H

3. REGISTER ADDRESSING MODE:

- The instruction will specify the name of the register which holds the data to be operated by the instruction.
- All registers except IP may be used in this mode
- Examples:
  - **MOV AX, BX**
    - The content of 16-bit register BX is moved to another 16-bit register AX
  - **ADC AL, CL**
    - Operation Add with Carry is performed with the Contents of 8-bit register AL and 8-bit register CL

4. REGISTER INDIRECT ADDRESSING MODE

- Here the address of memory location which contains data or operand is determined in an indirect way using offset register
- Offset address is in any of the following registers: BX, SI or DI registers.
- Examples
  - **MOV AX, [BX]**
    - Suppose the register BX contains 2000H, then the contents of 2000H (offset address) + 10H * [DS] is moved to AX
  - **ADD CX, [BX]**
    - Suppose BX register contains 4000H, then the content of 4000H + 10H * [DS] is added with content of CX and the result is stored in CX

5. INDEXED ADDRESSING MODE

- In this addressing mode, the operands offset address is stored in one of the index registers .
- DS and ES are the default segments for index registers SI and DI respectively.
- This is the special case of the previous register indirect addressing mode.
- Example
  - **MOV AX, [SI]**
    - Effective address is 10H *[DS] + [SI]. Content from this address is moved to AX
  - **ADD CX, [DI]**
    - Effective address is 10H *[DS] + [DI]. Content from this address is added with the content of CX and the result is moved to CX

6. REGISTER RELATIVE ADDRESSING MODE

- In register relative, data is available at an effective address formed by adding an 8-bit or 16-bit displacement with the content of any one of the registers BX, BP, SI and DI in default DS or ES segment

- Example
  - **MOV AX, 50H [BX]**
    - 50 H is immediate 8-bit displacement
    - EA = 10H*[DS] + [BX] + 50H
    - Content of EA is moved to AX

7. BASED INDEXED ADDRESSING MODE:

- In this addressing mode, the effective address is formed by adding the contents of base register (BX or BP) to the content of index register (SI or DI).
- The default segment registers may be ES or DS
- Example:
  - MOV AX, [BX] [SI]
    - EA = 10H * DS + [BX] +[SI]
    - Content of EA is moved to AX
  - MOV [BX] [DI], AX
    - EA = 10H * DS + [BX] +[DI]
    - Content of AX is moved to the EA calculated

8. RELATIVE BASED INDEXED ADDRESSING MODE:

- Effective Address is calculated by adding
  - 8 or 16-bit displacement
  - the sum of the contents of
    - any one of the base registers (BX or BP) and
    - any one of the index registers (SI or DI) in default segment

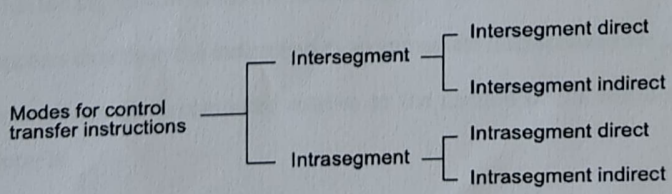
- Example
  - MOV AX, 50H [BX] [SI]
    - 50 H is immediate 8-bit displacement
    - EA = 10H*[DS] + [BX] + [SI] + 50H

For Control Transfer / Branch Instructions :

- Addressing mode depends on whether the destination location is within the same segment or in a different segment.

- Basically 2 addressing modes for Control Transfer Instructions
- **Intersegment mode**
  - If the location to which the control is to be transferred lies in a different segment, then it is intersegment mode
- **Intrasegment mode**
  - If the destination location lies in the same segment, then it is intrasegment mode

- **Addressing Modes for Control Transfer / Branch Instructions**



- **Intrasegment Indirect Addressing Mode**
- In this mode the displacement to which the control is to be transferred is in the same segment in which the control transfer instruction lies
- but it is passed to the instruction indirectly
- Here the branch address is found as the content of a register or a memory location .
- Example **JMP [AX]**
  - AX contains the branch address to which the control is to be transferred

Intersegment Direct Addressing Mode

- In this mode the address to which the control is to be transferred is in a different segment
- This addressing mode provides a means of branching from one code segment to another code segment.
- Here the **CS and IP** of the destination address are specified directly in the instruction.
- Example: **JMP 2000H: 3000H**
  - Jump to effective address 3000H in segment 2000H

Intersegment Indirect Addressing Mode

- Here, the address to which the control is to be transferred lies in a different segment and
- it is passed to the instruction indirectly
- Ie., Content of memory block containing four bytes **IP(LSB) ,IP(MSB), CS(LSB) and CS(MSB)** are passed sequentially
- The starting address of the memory block may be referred using any of the addressing mode except immediate mode .
- Example **JMP [5000H];**
  - Memory location 5000 points to four bytes IP(LSB) ,IP(MSB), CS(LSB) and CS(MSB) and jumps to the location specified

- **Intrasegment Direct Addressing Mode**
- In this mode, the address to which the control is to be transferred lies within the segment in which the control transfer instruction lies and
- It appears directly in the instruction as an immediate displacement value .
- The displacement is computed relative to the content of the instruction pointer IP.
- Eg: JMP SHORT LABEL;
  - Here, SHORT LABEL represents a signed displacement.

b) **Find the physical address of the destination operands referred in the following instructions, if DS=0223H, DI=0CCCH and SI=1234H**

a) MOV [DI], AL **b) MOV [SI][56H], BL** **(5 marks)**