Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q. 1 A) Attempt any Six of the following **12M**

a) What are W and Z registers of 8085? **2M**

Ans: (W register-1M , Z registers – 1M)

The temporary registers W and Z are intended for internal use of the processor and it cannot be used by the programmer. 8085 uses it internally during execution of instructions like XCHG, to hold the data or address temporarily.

b) List the 16 Bit register pairs of 8085?(Any 2 pair , 1 Mark each) **2M**

Ans:

The valid 16 bit register pair of 8085 are

B	C
D	E
H	L

c) Give one Example each of register addressing mode and direct addressing mode? **2M**

Ans: (Register Addressing mode – 1M , direct addressing mode – 1M)

(Note: Marks should be given for any valid example)

Register Addressing Mode: MOV AL,BL
MOV CH,DL etc

Direct Addressing Mode: MOV AL,[1200h]
MOV [5000h],CL etc

d) Find errors in MOV AL, CX AND MOV DS, ES Instruction? 2M

Ans: (MOV AL, CX- 1M MOV DS, ES- 1M)

MOV AL, CX: In this Instruction the content of CX cannot be copied to AL register since AL is 8 bit register and CX is 16 Bit register. The source and destination both should be of same type that is type Byte or Word.

MOV DS, ES: CS, DS, SS, ES are segment registers and transfer from a segment register to another segment register is not allowed in 8086. However, you can use a general purpose register (ax) to retrieve and set these registers.

e) What is the use of DF flag bit in 8086? 2M

Ans:

This bit is specially used by string instructions.

If DF=1, the string instruction will automatically decrement the pointer. If DF=0, the string instruction will automatically increment the pointer.

f) Give the initialization checklist for 8086 assembly language programming? 2M

Ans:

- In program there are many variables, constants and various part of the system such as segment registers, flags, stack, programmable ports etc which must be initialize properly.
- The best way to approach the initialization task is to make the checklist of the entire variables, constants, and all the registers flags and programmable ports in the program
- At this point you will come to know which part on the checklist will have to be initialized

g) Explain ENDP directive with Example? 2M

Ans: (explanation 1M)

ENDP directive is used along with the name of the procedure to indicate the end of a procedure to the assembler

Example: (1M)

SQUARE_NUM PROC; It start the procedure

Some steps to find the square root of a number

SQUARE_NUM ENDP ; Here it is the End for the procedure

h) What are the advantages of Segmentation? (any two advantages 1 Mark each)

Ans:

Some of the advantages of memory segmentation in the 8086 are as follows:

- The Address bus of 8086 is 20 bit. With the help of memory segmentation a user is able to access 20 bit address with 16 bit registers only.
- By memory segmentation the various portions of a program can be of more than 64kb.

- The data and the user's code can be stored separately allowing for more flexibility.
- Also due to segmentation, the logical address range is from 0000H to FFFFH the code can be loaded at any location in the memory.

Q. 1B) Attempt any Two of the following? **8M**

a) What are the major steps in developing ALP? (Each step – 1 Mark, Total - 4M)

Ans:

- Editor: The first step in the process is to develop assembly program text. Assemblers typically provide assembly text editors to help program development. At assemble time, the input text is passed to the assembler, which parses the text, emitting a program listing and possibly error reports.
- Assembler: If there are no errors, the assembler produces a binary machine language module. The module must contain information about where the program or module is to be loaded in memory, and if it contains the starting address of the program, this start symbol must be made known.
- Linker: If the module has to be linked with other modules, then it must include this additional linkage information. This means that all labels in the module that have to be visible to other modules must be specified as public symbols. Similarly, all labels that are defined in other modules must be specified as external symbols.
    - At link time, the linker combines separately assembled modules into a single load module. The linker will also add any initialization or finalization code to allow the operating system to start the program or return control to the OS, once the program has completed.
- Loader and Debugger: At load time, the program loader copies the program into computer's main memory. A debugger or an emulator can be used to run and debug the program.
- At run-time, the program execution begins.

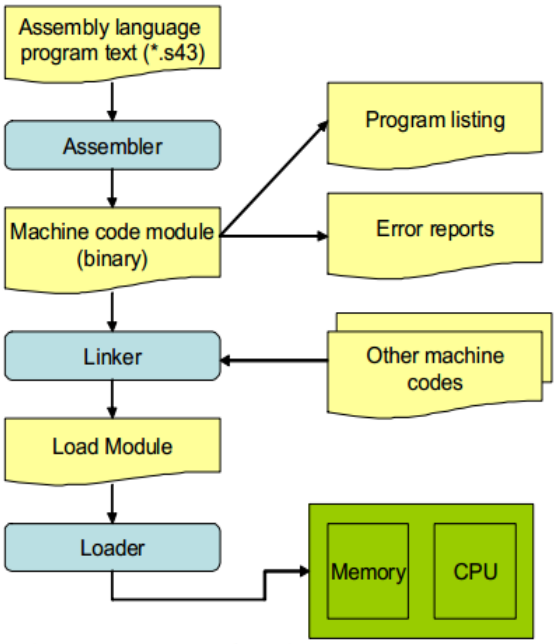


Figure 1. Program development flow using assembly language programming.

b) Compare Minimum Mode and Maximum Mode operation of 8086?

Ans:

(Any Four Points One Mark for Each Point)

Minimum Mode	**Maximum Mode**
1. MN/MX = 1 for minimum mode	1. MN/MX = 0 for maximum mode.
2. Single μp in the minimum mode system.	2. Multiprocessor configuration.
3. Bus controller IC 8288 not present.	3. Bus controller IC 8288 is used to generate control signal.
4. Advanced I/O write signal is not there	4. Advanced I/O write command is also there.
5.Status pins S0 S1 S2 are not used	5. Status pins S0 S1 S2 are used to specify the type of transfer that is to be carried out.

c) What memory will be accessed if CS = 124C H, IP - 4621H and SS = 3344 H, SP = 1342H?

Ans:

1st case: **2 Marks**

CS:

1	2	4	C	**0**

IP: +

4	6	2	1

20 BIT ADDRESS:

1	**6**	**A**	**E**	**1**

2nd case: **2 Marks**

SS:

3	3	4	4	**0**

SP:

1	3	4	2

20 BIT ADDRESS:

3	**4**	**7**	**8**	**2**

Q.2 Attempt any four of the following: **16M**

a) What are the Limitations of 8 bit Microprocessor? **4M**

Ans: Any 4 points: 1 Mark for each point.

The Limitation of 8 bit Microprocessor is as follows.

1) It is an 8 bit processor hence the size of ALU is 8 bit.
2) Pipelining is not available; hence only one instruction is executed at a time.
3) Operating frequency is less, so the speed of execution is slow
4) 8 bit processors cannot be used in multiprocessor mode
5) There is no memory management unit available in 8 bit processors
6) While reading and writing 16 bit data extra cycles are required to read and write this data bits
7) There is no flag available condition to indicate overflow.

b) Write ALP to multiply the content of BX by 4 using shift instructions. **4M**

Ans:

Correct Program : **4 M**

```
DATA SEGMENT
      Multiplier db 02h
      Multiplicand dw 0100h
DATA ENDS
          CODE  SEGMENT
ASSUME CS:CODE,DS:DATA
START:Mov AX, DATA      ;initialize data segment
       Mov DS, AX

       Mov BX, Multiplicand
       Mov CL, Multiplier
       SAL BX, CL
        MOV AX,4C00h
        INT 21h
CODE ENDS
END START
```

c) Describe the function of following assembly language programming tools. **4M**

i) Linker ii) Debugger

LINKER: 2 Marks

A linker is a program used to join together several objects 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 indivisually written, tested and debugged. When all the module work, they can be linked togethet to form a large functioning program.

The linker produces a link file which contains the binary codes for all the combined modules. The linker also produces a link map which contains the address information to the program, it only assigns relative address es starting from zero. This form of the program is said to be relocatable, because it can be put anywhere in memory to be run.

The command on command prompt for converting .obj file to ,EXE file is as given below:

C : \ MASM \ BIN \ > LINK myprog.obj;

DEBUGGER: 2 Marks

A debugger is a program which allows us to load object code program into system memory execute the program, and debug it.

1. The debugger allows us to look at the contents of registers and memory locations after our program runs.
2. It allows us to change the contents of register and memory location and return the program.
3. Some debugger allows us to stop execution after each instruction so we can check or alter memory and register content.
4. A debugger also allows us to set a breakdown at any point in our program. When we run a program, the system will execute instructions up to this breakpoint and stop. We can then examine register and memory contents to see if the result are correct at that point. If the result are correct, we can move the break point to a later point in our program. If result are not correct we can check the program up to that point to find out why are not correct.

In short, debugger tools can help us to isolate problems in our program.

d) Explain re-entrant procedure with sketch? 4M

Ans:

Explanation : 2M

In some situation it may happen that

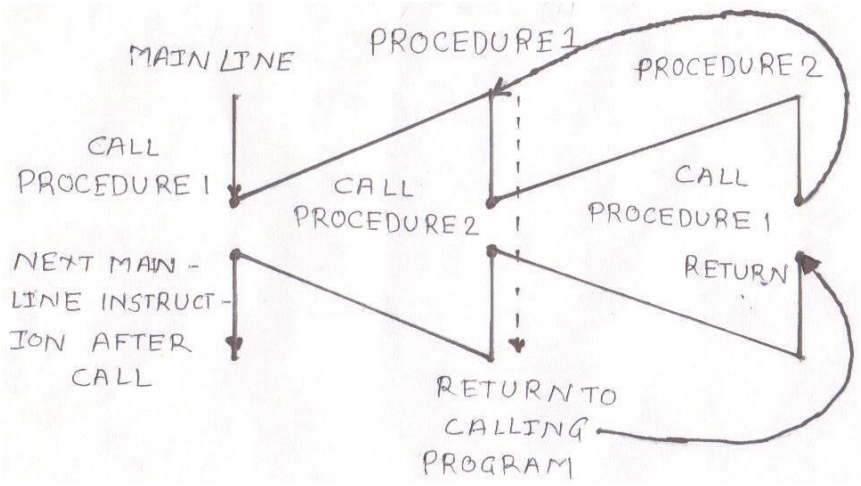
- Procedure1 is called from main program,
- Procedure2 is called from procedure1
- And procedure1 is again called from procedure2.

In this situation program execution flow reenters in the procedure1. These types of procedures are called reentrant procedures.

The RET instruction at the end of procedure1 returns to procedure2. The RET instruction at the end of procedure2 will return the execution to procedure1. Procedure1 will be again executed from where it had stopped at the time of calling procedure2 and the RET instruction at the end of this will return the program execution to main program.

The flow of program execution for reentrant procedure is shown in fig.

Sketch : **2M**



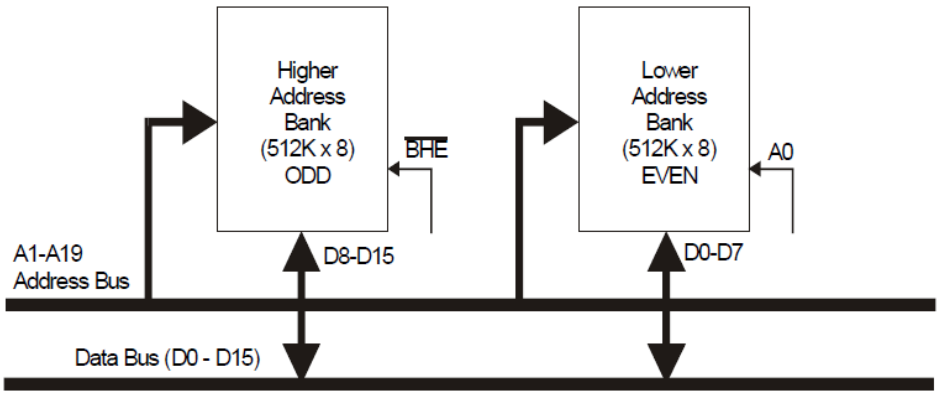
e) Why 8086 memory is divided into two banks? **4M**

Ans: Diagram – 1M Explanation: 3M

The 8086 memory address space can be viewed as a sequence of one million bytes in which any byte may contain an 8-bit data element and any two consecutive bytes may contain a 16-bit data element. There is no constraint on byte or word address boundaries. The address space is physically connected to a 16-bit data bus by dividing the address space into two 8-bit banks of up to 512K bytes each.

One bank called as even bank, is connected to the lower half of the 16-bit data bus (D0 – D7) and contains even address bytes. *i.e.*, when A0 bit is low, the bank is selected. The other bank called as odd bank,is connected to the upper half of the data bus (D8 - D15) and contains odd address bytes. *i.e.*, when A0 is high and BHE (Bus High Enable) is low, the odd bank is selected. A specific byte within each bank is selected by address lines A1-A19.

This enables the 8086 to read 16 bits from the memory by simultaneously reading an odd addressed byte and even addressed byte. i.e, 8086 need to perform only one memory read cycle.



f) What is Procedure? Give its Format. **4M**

Ans: Explanation: 3M Format: 1M

The repeated group of instructions in a large program can be written separately from the main program. This subprogram is called as Procedure in an assembly language programming i.e. Procedure is a set of statements that can be processed independently from the main program.

For defining procedure. PROC & ENDP assembler directives are used. The PROC indicates the beginning of the procedure and ENDP directive indicates the end of procedure to the assembler. The procedure must be defined in the codes segment only.

Advantages:

i. Large programs can be split in to smaller modules.
ii. Reduces the size of the program.
iii. Procedures can be accused from other segments/programs also.
iv. Easy to debug the program.

The general format for procedure is

Procedure name PROC

Procedure Statements

Procedure Name ENDP.

Q.3) Attempt any four of the following: **16 Marks**

a) State the important features of 8085.

Ans: ***(Any 4 features 1 Mark for each)***

1. 8085 microprocessor can read or write or perform arithmetic and logical operations on 8-bit data at time.
2. It is a single chip NMOS device implemented with 6200 transistors.
3. It requires +5V power supply.
4. It provides on chip clock generator.
5. Maximum clock frequency is 3MHZ and minimum clock frequency is 500KHZ.
6. It provides 74 instructions with five addressing modes.
7. It provides 5 hardware interrupt and 8 software interrupts.
8. It has 8 data lines and 16 address lines hence capacity is 2^{16} = 64KB of memory.
9. It provides two serial I/O lines SID and SOD so that serial peripherals can be interfaced directly with 8085 microprocessor.
10. It can generate 8-bit I/O address so 2^8=256 input and 256 output ports can be accessed.

b) Explain MOVS instruction.

Ans: ***(Syntax: 1 Mark, Operation: 3 Marks)***

MOVS - Move String byte or word from memory source to memory destination. The source is always data segment and destination is extra segment.

Syntax:

MOVS destination, source

Operation:

ES:[DI]<----- DS:[SI]

It copies a byte a location in data segment to a location in extra segment. The offset of source is pointed by SI and offset of destination is pointed by DI. Direction Flag (DF) can be set or reset to auto increment or auto decrement pointers after one move.

c**) Write a ALP to divide 16-bit unsigned number with 8-bit unsigned number.**

Ans: (***Data segment: 1 Mark, Code segment [correct logic]: 3 Marks***)

```
DATA_DIV SEGMENT
 DIVIDEND   DW   0123H
 DIVISOR    DB   12H
 QUO        DB   0
 REM        DB   0
 DATA_DIV ENDS
CODE_DIV SEGMENT
ASSUME   CS: CODE_DIV, DS: DATA_DIV
START:   MOV AX, DATA          ; initialize data segment
 MOV DS, AX
 MOV AX, DIVIDEND      ; divide word bit by bit
 DIV DIVISOR
 MOV QUO, AL           ; store quotient
 MOV REM, AH           ; store remainder
 MOV AX, 4COOH
 INT 21H
 CODE_DIV ENDS
 END  START
```

d) Compare Far and Near Procedures.

Ans: ***(Any four point : 1 Mark each)***

Sr.no	Far Procedure	Near procedure
1.	A far procedure refers to a procedure which is in the different code segment from that of the call instruction.	A near procedure refers to a procedure which is in the same code segment from that of the call instruction
2.	It is also called inter-segment procedure call	It is also called intra-segment procedure
3	A far procedure call replaces the old CS:IP pairs with new CS:IP pairs	A near procedure call replaces the old IP with new IP.
4.	The value of the old CS:IP pairs are pushed on to the stack	The value of old IP is pushed on to the stack.
5.	More stack locations are required	Less stack locations are required
6.	Example :- Call FAR PTR Delay	Example :- Call Delay

e) Differentiate between I/O Mapped I/O and Memory Mapped I/O.

Ans: ***(Any four points: 1 Mark each)***

Sr.No	**I/O Mapped I/O**	**Memory Mapped I/O**
1.	In this technique I/O is treated as I/O and memory is treated as memory.	In this technique I/O and memory both are treated as Memory
2	$\overline{IOR}$ and $\overline{IOW}$ control signals are used to control read and write I/O operation.	$\overline{MEMR}$ and $\overline{MEMW}$ control signals are used to control read and write I/O operations.
3.	Data transfer is between accumulator and I/O device.	Data transfer is between any reg,ister and I/O device
4.	IN and OUT instructions are required for I/O read and write operation.	All memory related instructions are used to I/O devices.
5.	In this case I/O has an 8 bit address and memory has 16-bit address.	In this case both I/O and memory have a 16-bit address.
6.	In this, device address is 8-bit .Thus A0 to A7 or A8 to A15 lines are used to generate device address.	In this, device address is 16-bit.Thus A0 to A15 lines are used to generate the device address.
7.	Address decoding logic is simple as less hardware is requited.	Address decoding logic is complicated and expensive.
8.	I/O devices and memory are distinguished by control signals and addresses.	I/O devices and memory are distinguished by only addresses.
9.	Arithmetic and logical operations cannot be performed on I/O ports	Arithmetic and logical operations can be performed on I/O ports.

10.	Can interface maximum memory of 64KB and 256 I/O ports.	Can interface maximum memory of 64KB which also includes the I/O ports.
11.	Size of memory is not reduced.	Size of memory is reduced

f) Write a program in assembly language for addition of two 8-bit nos. using macro.

Ans: ***(Macro Declaration: 2 Mark, Main Program: 2 Marks)***

```
ADD_NO    MACRO    NO1, NO2, RESULT    ; Macro declaration
          MOV AL, NO1
          ADD AL, NO2
          MOV RESULT, AL
 ENDM
DATA_ADD SEGMENT
NUM1      DB    34H
NUM2      DB    21H
DATA_ADD ENDS
CODE_ADD SEGMENT
ASSUME    CS: CODE_ADD, DS: DATA_ADD
START:    MOV AX, DATA          ; initialize data segment
          MOV DS, AX
          ADD_NO    NUM1, NUM2, RES    ; call Macro
          MOV AX, 4C00H
          INT 21H
CODE_ADD ENDS
END START
```

Q4) Attempt any four of the following: 16 Marks

a) State four important features of 8086.

Ans: ***(Any four points: 1 Mark each)***

1. 20 bit address lines so 2^{20} = 1Mbyte of memory can be addressed.
2. Operating clock frequencies 5MHz, 8MHz, 10MHz.
3. Arithmetic operation can be performed on 8-bit or 16-bit signed & unsigned data including multiplication and division.
4. Provide 20 address lines so, 1 Mbytes of memory can be addressed.
5. The instruction set is powerful, flexible and can be programmed in high level language like C language.
6. Can operate in single processor and multiprocessor configuration i.e. operating modes.
7. Provides 6-bytes instruction queue for pipelining of instructions executions.
8. Provides 256 types of vectored software interrupts.

9. Operate in maximum and minimum mode to achieve high performance level.
10. Provides separate instructions for string manipulation.

b) Explain DAA instruction with example.

Ans: ***(Explanation: 2 Marks, Example: 2Marks)***

DAA – (Decimal Adjust AL after BCD Addition)

Syntax- DAA

Explanation:

This instruction is used to make sure the result of adding two packed BCD numbers is adjusted to be a correct BCD number.

The result of the addition must be in AL for DAA instruction to work correctly.

If the lower nibble in AL after addition is > 9 or Auxiliary Carry Flag is set, then add 6 to lower nibble of AL. If the upper nibble in AL is > 9 or Carry Flag is set, and then add 6 to upper nibble of AL.

Example: - (Any Same Type of Example)

if AL=99 BCD and BL=99 BCD

Then ADD AL, BL

1001 1001 = AL= 99 BCD
+ 1001 1001 = BL = 99 BCD
0011 0010 = AL =32 H and CF=1, AF=1

After the execution of DAA instruction, the result is

CF = 1
0011 0010 =AL =32 H AF =1
+ **0110 0110**

1001 1000 =AL =98 in BCD

c) Explain ASSUME and EVEN directives.

Ans: ***(2 Marks for each directive)***

ASSUME: - Assume directive is used to tell Assembler the name of the logical segment it should use for the specified segment.

When program is loaded the processor segment register should point to the respective logical segments.

Example: - Assume CS: MSBTE_CODE, DS: MSBTE_DATA

EVEN: - The directive even is used to tell assembler to increment the location counter to the next even memory address, if it is not already at an even address.

If the location counter is already pointing to even address it should not be increment. The 8086 reads/writes data in one machine cycle, if data is to be accessed from even location. It requires two bus cycle to access a word from odd memory location. This directive can be used in both Code and Data segment.

Example: - DATA SEGMENT
Array db 9 DUP (?)

```
        Even
        Block   dw   75H   DUP (0)
DATA ENDS
```

d) Explain CALL and RET instruction.

Ans: ***(2 Marks for each instruction)***

The CALL instruction is used to transfer execution to a subprogram or procedure .

There are two types of calls-NEAR and FAR

Syntax: CALL procedure name

(direct/indirect)

Operation-: Steps executed during CALL

1) For Near CALL

SP ⟵ SP - 2

Save IP on stack

IP ⟵ address of procedure

2) For Far call

SP SP-2

Save CS on stack

CS ⟵ New segment base containing procedure

SP ⟵ SP - 2

Save IP on stack

IP ⟵ Starting address of called procedure

RET: (Return from procedure)

The instruction RET is used to transfer program control from the procedure back to the calling program.

Syntax:- RET

Operation:- Steps executed during RET

1) For Near Return

IP ⟵ Content from top of stack

SP ⟵ SP + 2

2) For Far Return

IP ⟵ Contents from top of stack

SP ⟵ SP+2

CS ⟵ Contents of top of stack

SP ⟵ SP+2

e) Describe TEST and XCHG instruction.

Ans: ***(2 Marks for each instruction)***

Syntax: TEST Destination, Source

This instruction AND's the contents of a source byte or word with the contents of specified destination byte or word and flags are updated, , flags are updated as result ,but neither operands are changed.

Operation performed:

Flags ⟵ set for result of (destination AND source)

Example: (Any 1)

TEST AL, BL ; AND byte in BL with byte in AL, no result, Update PF, SF, ZF.

TEST CX, 0001H ; AND word in CX with immediate data 0001H, no result, Update PF, SF, ZF .

TEST CX, [SI] ; AND word at offset[SI] in data segment with word in CX, no result, Update PF,SF,ZF.

XCHG Destination, Source

This instruction exchanges the contents of a register with the contents of another register or memory location.

Operation performed:

Destination ⟷ Source

Example: (Any 1)

XCHG AX, BX ; Exchange the word in AX with word in BX.

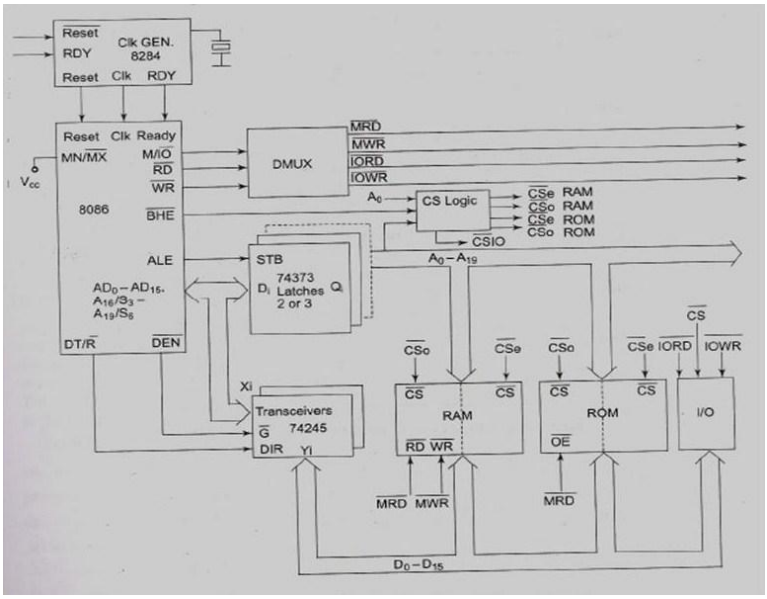
XCHG BL, CL ; Exchange the byte in BL with byte in CL.

XCHG AL, Array [BX] ; Exchange the AL with byte in memory at effective address Array[BX].

Q5) Attempt any two of the following. (16 Marks)

a) Draw and explain the minimum system configuration of 8086.

Ans: ***(4 Marks for (Any relevant diagram), 4 Marks for Explanation)***



Explanation:

In minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/$\overline{MX}$ pin to logic 1 .There is a single microprocessor in the minimum mode system. The remaining components in the system are latches, transceivers, clock generator, memory and I/O devices.

The latches are generally buffered output D-type flip-flop like 74LS373 of 8282.Latches are used for separating the valid addresses from the multiplexer address/ data signals and controlled by ALE signal generated by 8086.

Transceivers are the bidirectional buffers and also called as data amplifiers. They are required to separate the valid data from time multiplexed address/ data signal. They are controlled by two signals namely $\overline{DEN}$ and DT/$\overline{R}$. the $\overline{DEN}$ signal indicated that the valid data is available on the data bus. DT/$\overline{R}$ indicates the direction of data, i.e., from or to the processor.

The clock generator generates the clock from the crystal oscillator. The clock generator also synchronizes some external signals with the system clock. Since 8086 has 20 address lines & 16 data lines, the 8086 CPU requires 3 octal latches & two octal data buffers (transceivers) for the complete address & data separation.

b) i) Identify the addressing modes of each of the following instruction.

i. MOV BX, OA42H
ii. MOV AL,CL
iii. DAA
iv. MOV AL, [3000 H]

(Correct Addressing Mode 1 Mark each)

Ans: **i) MOV BX, OA42H: Immediate Addressing mode**

ii) MOV AL, CL: Resister Addressing mode

iii) DAA: Resister Addressing mode

iv) MOV AL, [3000 H]: Direct Addressing mode

ii) Define and explain four logical instructions.

Ans: ***(Any 4 instructions 1Mark each)***

(NOTE: Either explanation or detailed example :1 Mark)

1) AND- Logical AND

Syntax **(optional)**

AND destination, source

Operation

Destination ←destination AND source

This instruction AND's each bit in a source byte or word with the same number bit in a destination byte or word. The result is put in destination.

(Or)

Example: AND AX, BX

2) OR – Logical OR

Syntax **(optional)**

OR destination, source

Operation

Destination ⟵ destination OR source

This instruction OR's each bit in a source byte or word with the corresponding bit in a destination byte or word. The result is put in a specified destination.

(OR)

Example: OR AH, CH

3) NOT – Logical Invert

Syntax: (optional)

NOT destination

Operation

Destination ⟵ NOT destination

The NOT instruction inverts each bit of the byte or words at the specified destination.

(OR)

Example **(optional)**

NOT AH

4) XOR – Logical Exclusive OR

Syntax

XOR destination, source

Operation

Destination⟵ destination XOR source

This instruction exclusive, OR's each bit in a source byte or word with the same number bit in a destination byte or word.

Example (optional)

XOR CL, BH

5) TEST – Logical Compare to update flags

Syntax **(optional)**

TEST destination, source

Operation

Flags update ⟵ destination AND source.

This instruction AND's the contents of a source byte or word with the contents of the specified destination byte or word.

Example (optional)

TEST CX, 0001H

6) SHL/SAL – shift operand bits Left, Put zero in LSB(S)

Syntax

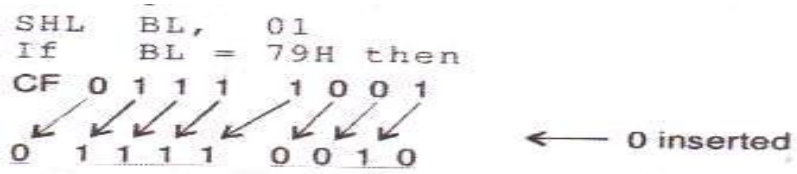
SHL/SAL destination, count

Operation

CF←MSB ←—— LSB ←——— 0

These instructions shift the operand word or Byte bit by bit to the left and insert zeros in the newly introduced LSB position.

Example (optional)



7) SHR – Shift operand bits right, put zero in MSB(s)

Syntax

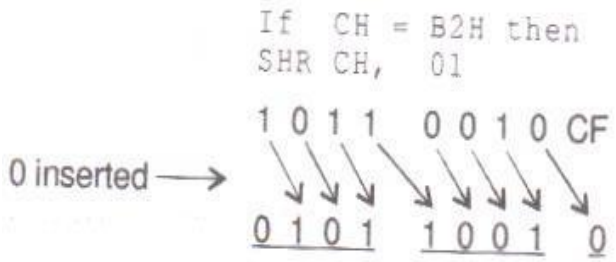
SHR destination, count

Operation

0⟶ MSB⟶ SB ⟶ CF

This instruction shifts each bit in byte or word destination to the right & insert zero in the newly introduced MSB position.

Example (optional)



8) SAR – Shift operand bits right, new MSB = Old MSB.

Syntax :

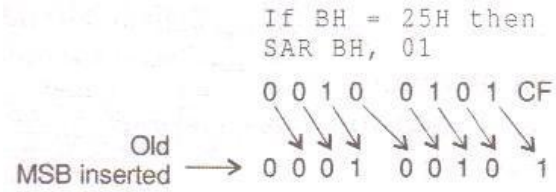
SAR destination, count

Operation

MSB ⟶ MSB ⟶ LSB ⟶ CF

This instruction performs right shifts on the operand word or byte, that may be a register or a memory location.

Example (optional)



9) ROL [Rotate left without carry]

Syntax: ROL Destination, Count

This instruction rotates all of the bits of the specified byte or word count times towards left. The bit moves out of MSB is rotated around into LSB and copied to CF.

Operation

Example: (optional) CF=0 BL=1011 1010

ROL BL, 1 ; Rotate all bits in BL left by one bit position.

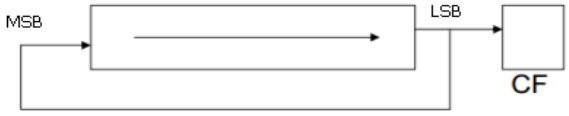
CF=1 BL=0111 0101

10) ROR [Rotate right without carry]

Syntax: ROR Destination, Count

This instruction rotates all of the bits of the specified byte or word count times towards right. The bit moves out of LSB is rotates around into the MSB and also copied to CF.

Operation



Example (optional)

CF=0 BL=0011 1011

ROL BL, 1 ; Rotate all bits in BL right by one bit position.

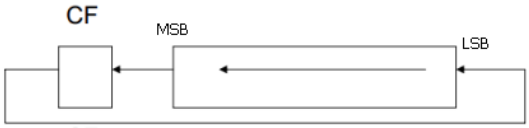
CF=1 BL=1001 1101

11) **RCL [Rotate left with carry]**

Syntax: RCL Destination, Count

This instruction is used to rotate all bits in a specified byte or word with carry by count times towards left. The operation is circular because the MSB of the operand is rotated into the carry flag and bit in the carry flag is rotated around into the LSB of the operand.

Operation



Example. (Optional)

CF=1 BL=0011 1011

RCL BL, 1 ; Rotate all bits in BL left by one bit position.

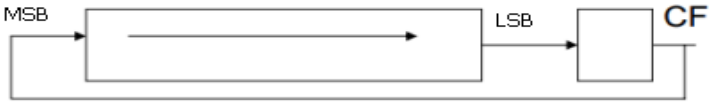
CF=0 BL=0111 0111

12) **RCR [Rotate right with carry]**

Syntax: RCR Destination, Count

This instruction is used to rotate all bits in a specified byte or word with carry by count times towards right. The operation is circular because the LSB of the operand is rotated into the carry flag and bit in the carry flag is rotated around into the MSB of the operand.

Operation



Example (optional). CF=0 BL=0011 1011

RCR BL, 1 ; Rotate all bits in BL right by one bit position.

CF=1 BL=0001 1101

c) Write an assembly language program to arrange five data bytes in ascending order. Draw flow chart.

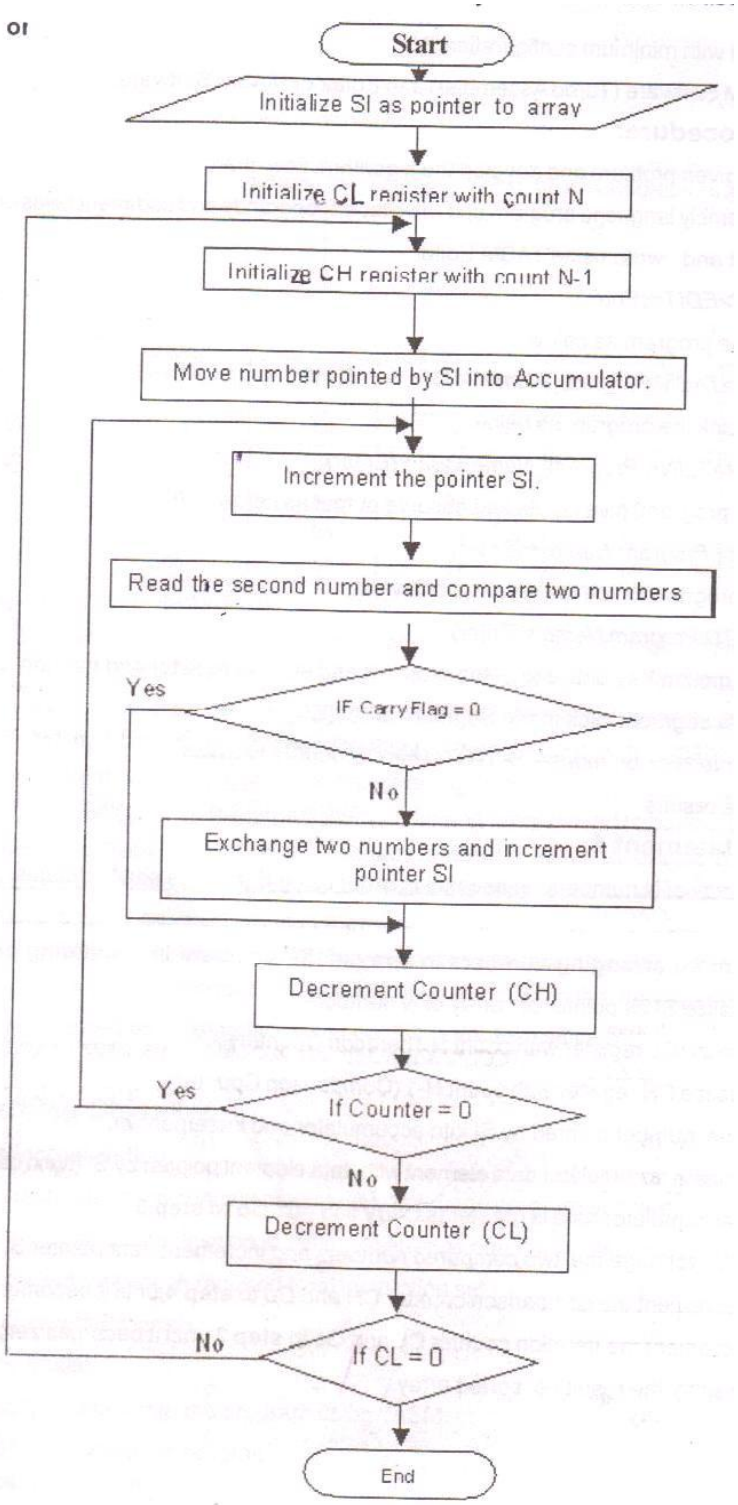
Ans: ***(Data segment 1mark, Code segment (correct logic) 4 Marks, Flowchart 3Mark)***

(Note: Any Other Logic can be considered)

```
DATA_ASORT SEGMENT
        NUM DB 87H, 67H, 22H, 45H, 83H
        DATA_ASORT ENDS
        CODE_ASORT SEGMENT
        ASSUME CS: CODE_ASORT, DS: DATA_ASORT
        START:      MOV AX, DATA            ; initialize data segment
        MOV DS, AX
        MOV CL, 05H             ; iteration counter
LOOP1:  LEA BX, NUM
        MOV CH, 04H             ; comparison counter
LOOP2:  MOV AL, [BX]
        INC BX
        CMP AL, [BX]
        JC  DOWN                ; if number is smaller skips exchange procedure
        MOV DL, [BX]            ; exchange two numbers
        MOV [BX], AL
        DEC BX
        MOV [BX], DL
        INC BX
```

```
DOWN:   DEC CH
        JNZ LOOP2   ; if comparison counter is not zero, jump to LOOP1
        DEC CL
        JNZ LOOP1   ; if iteration counter is not zero, jump to LOOP1
        MOV AX, 4C00H
        INT 21H
CODE_ASORT ENDS
```

or



END START

Q.6) Attempt ant two of the following **16 Marks**

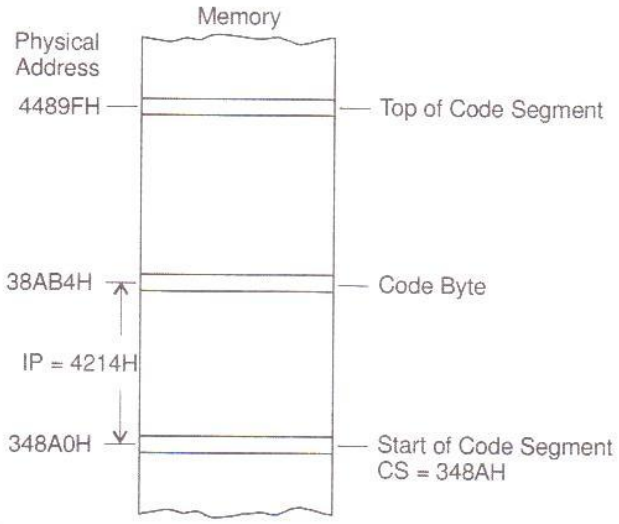
a) Define logical address and physical address. Explain 20-bit physical address generation with example.

[Logical address :2 Marks , Physical address :2 Marks, Address generation:2Marks ,Example 2Mark(any 1])

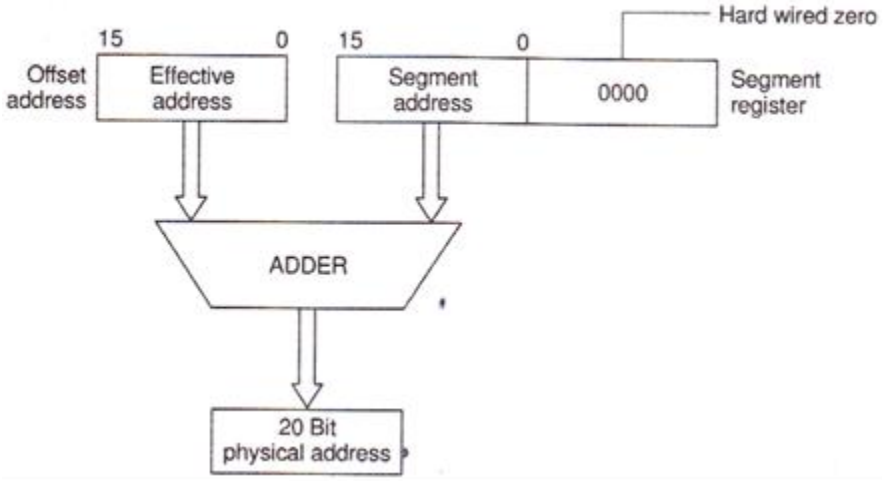
Physical Address:- The 8086 has 20 address lines so each memory location have 20 bit address which is actually put on address bus known as physical address. The address can have range of 00000H to FFFFFH for 8086. Physical address is calculated with the help of logical address and base address which is always present in segment register.

Logical Address:- the logical address is normally stored in the 16-bit base register(SP, BP) or pointer register(BP, SP) or index register(SI, DI) which is an offset from location 0 of a respective segment(CS,DS,SS,ES).

Or (Relevant diagram with explanation can be given full mark)



Formation of a physical address:- segment registers carry 16 bit data, which is also known as base address. BIU attaches 0 as LSB of the base address.so now this address become 20 bit address. Any base/pointer or index register carry 16 bit offset. Offset address is added into 20 bit base address which finally forms 20 bit physical address of memory location.



Example:-

CS- 348A0 ⟵ Hardwired zero
IP - 4214 Offset

Physical address 38AB4

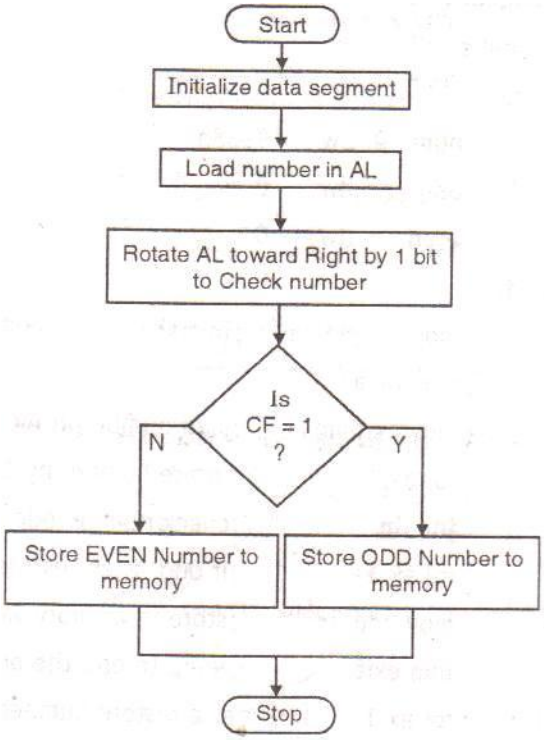
(OR Any Same Type of Example)

6(b) Write an assembly language program to test whether a given number is ODD or EVEN. Draw Flowchart.

(Data segment 2mark, Code segment (correct logic) 3Marks, Flowchart 3Mark)

```
DATA_EVENODD SEGMENT
NUM  DB   89H
ODD  DB   0
EVEN  DB  0
DATA_EVENODD ENDS
CODE_EVENODD SEGMENT
ASSUME     CS: CODE_EVENODD, DS: DATA_EVENODD
START:      MOV AX, DATA          ; initialize data segment
        MOV DS, AX
        MOV AL, NUM             ; load number in AL
        ROR AL, 1               ; rotate number by 1 bit towards right
        JNC  DN                 ; check number odd or even
        ROL AL, 1               ; if odd, then restore the number
        MOV ODD,AL              ; store in memory variable ODD
        JMP EXIT            ; jump to end program
DN:     ROL AL, 1           ; else restore number
        MOV EVEN,  AL       ; store in memory variable EVEN.
        MOV AX, 4COOH
        INT 21H
```

```
EXIT:        CODE_EVENODD ENDS
             END START
```



c) Interface two 8K RAM chips with 8086 as EVEN and ODD memory banks. Give complete memory map.
Ans:

Address Mapping

2^{13} = 8KB **(01 Mark)**

EVEN RAM - Initial address **(01 Mark)** EVEN RAM - Final address **(1 Mark)**

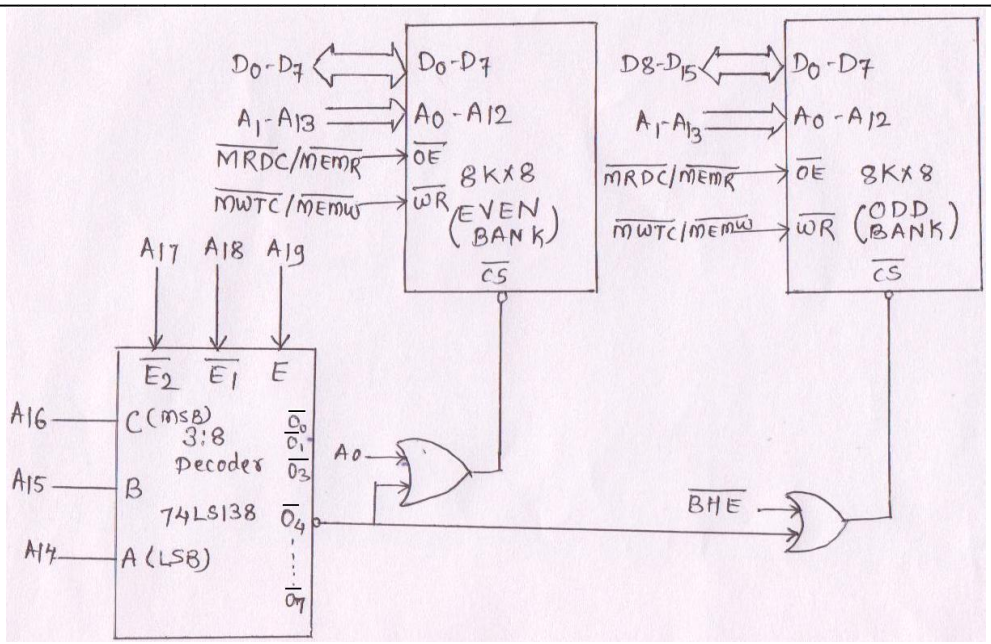
ODD RAM - Initial address **(01 Mark)** ODD RAM - Final address **(1 Mark)**

(Note: Students may assume any starting address which will also change decoder connections, should be considered.)

Interfacing diagram: **(03 Marks)**

A_{19} A_{18} A_{17} A_{16} A_{15} A_{14}	A_{13} A_{12} A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0	20 bit physical Address	
1 0 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0	90000H	EVEN RAM
1 0 0 1 0 0	1 1 1 1 1 1 1 1 1 1 1 1 1 0	93FFEH	
1 0 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 1	90001H	ODD RAM
1 0 0 1 0 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1	93FFFH	

Used for chip select decoder logic


D0-D7
A1-A13
MRDC/MEMR
MWTC/MEMW
D0-D7
A0-A12
OE
WR
8K×8
(EVEN BANK)
CS
D8-D15
D0-D7
A1-A13
A0-A12
MRDC/MEMR
MWTC/MEMW
OE
WR
8K×8
(ODD BANK)
CS
A17
A18
A19
E2
E1
E
A16
C (msB)
3:8
Decoder
74LS138
A15
B
A14
A (LSB)
O0
O1
O3
O4
O7
A0
BHE