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:

i. a) **List maskable and non maskable hardware interrupt of 8085. (1 M each)**

Ans.: i) Maskable **interrupts** are RST 7.5 ,RST 6.5,RST 5.5,INTR

Non maskable interrupts are TRAP

ii) **State the functions of following pins of 8085 microprocessor (1 M each)**

1. **SOD**
2. **HOLD**

Function of pins:

1) **SOD** :-Serial Output data
SOD pin is used to transmit data serially from accumulator to the external devices connected to the pin.

2) **HOLD** –HOLD is an active high input signal. It is used to request microprocessor for gaining the control of buses i.e. address bus data bus and control bus.

iii) **State the maximum size of memory that can be interfaced with microprocessor 8086. Why? (Size of Memory 1M, Explanation 1 M)**

Ans.: In 8086 microprocessor the total memory addressing capability is 1 mega bytes
For representing 1 MB there are minimum 4 hex digits are required i.e., 20 bits. but 8086 has fourteen 16-bit registers. That is there are no registers for representing 20 bit address. So, the total memory is divided into 16 logical segments and each segment capacity is 64 KB (kilo bytes). That is 16*64kb=1 MB. So, for representing 64 kb only 16 bit register is sufficient.

iv) **State any four examples of immediate addressing mode. (1/2 Marks for each)**

Ans.: Immediate addressing mode example are as
1) MOV AX, 0005H: AX is loaded with 16bitimmediate data i.e.0005H

2) MOV AL, 25H: Move 25H to AL or AL= 25H

3) MOV [SI], 0B29H – Move 0B29H (16 bit data) to two consecutive memory location which effective address or offset is in SI and base address is in DS.

4) MOV CX, 1234H: Move 16 bit data to CX or CX =1234H

v) **State the function of "Assembler".**

(2M for function)

Ans.: Assembler is a program that translate assembly language program to the correct binary code

for each instruction.

vi**) State the concept of pipelining of 8086.**

Ans.: concept of pipeline: (2 Mark)

- In pipelined processor, fetch, decode and execute operation are performed simultaneously or in parallel.
- When first instruction is being decoded ,same time next instruction's code is fetches. When first instruction is getting executed, second one's is decoded and third instruction code is fetches from memory. This process is known as **pipelining**. It improves speed of operation to great extent.

vii) **State the steps involved in ALP using procedure.**

Ans.: Step involved in ALP: **(2 Mark)**

1. Defining the problem
2. Algorithm
3. Flowchart
4. Initialization of checklist
5. Choosing instructions
6. Converting algorithms to assembly language program

OR

- When microprocessor executes the CALL instruction, the control is transferred to the procedure, but before it goes to the subroutine, it saves the returning address on the top of the stack.
- The stack pointer is decremented by TWO and copies the offset of next instruction after the CALL on the stack.
- The control is again transferred to the main program when microprocessor comes across RET instruction at the end of the procedure. This done by popping up the offset saved on the stack back to IP

viii) **Explain any one logical instruction of 8086 with example.**

Ans.: Logical instructions are : (Explanation 1 Mark , Example 1 Mark)

(Students may also write similar answer for other logical instructions)

1) **AND** : This instruction bit by bit ANDS the source operand that may be an immediate a register or a memory location to the destination operand that may be a register or a memory location

EX: AND AX, 0008H

2) OR 3) NOT 4) XOR

5) TEST 6) CMP

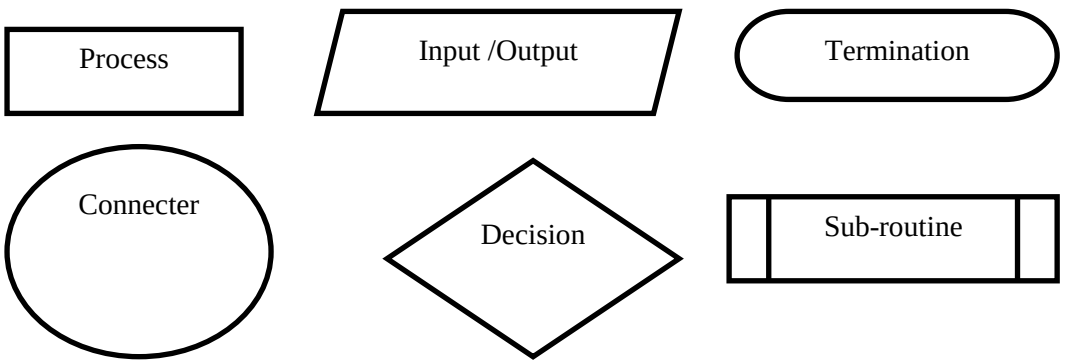
b) Attempt any TWO of the following:

i) What is an algorithm? What is a Flow chart? Sketch any four symbols used in flow chart.

Ans.: i) Algorithm: Algorithm is a task or sequence of operations perform by program can be specified as a step called as algorithm. **(1 Mark)**

ii) Flow chart is a graphical representation of task**.** **(1 Mark)**

iii) Symbols: **(2 Mark)**



ii) List any four assembler directives and explain any two of them.
(Any other assembler directives written by students can be considered)

(Students may also write directives from segment define directives, segment combine directives and processor directives)

(Listing of any four 2 marks, Explanation of any two 1 M each)

Ans.: Assembler directives are: 1) DB 2) DW 3) DQ 4) DT 5) ? (Uninitialized value)

6) PTR 7) OFFSET 8)EQU 9) DUP 10) LABEL 11)ALIGN
12) EVEN 13) ORG 14) DOSSEG 15) STRUCT 16) DD

1) DB: Define byte (8 -bit)

The db directive is used to reserve byte or bytes of memory locations in the available memory.

2) DW: Define word (16-bits)

The dw directive is used to reserve the memory words instead of bytes.

iii) Difference between NEAR and FAR calls (four points)
(1 Mark for each point)
Ans.:

	NEAR CALLS	FAR CALLS
1	A Near call called as Intra-segment call	Far call called as Intersegment call
2	A Near call refers to procedure call which is in the same segment as the CALL instruction	A Near call refers to procedure call which is in the different code segment as the CALL instruction
3	A Near call replaces the old IP with new IP.	A Far call replaces the old CS:IP pairs With new CS:IP pairs.
4	Less stack location are required	More stack location are required.

Q.2. Attempt any Four of the following:

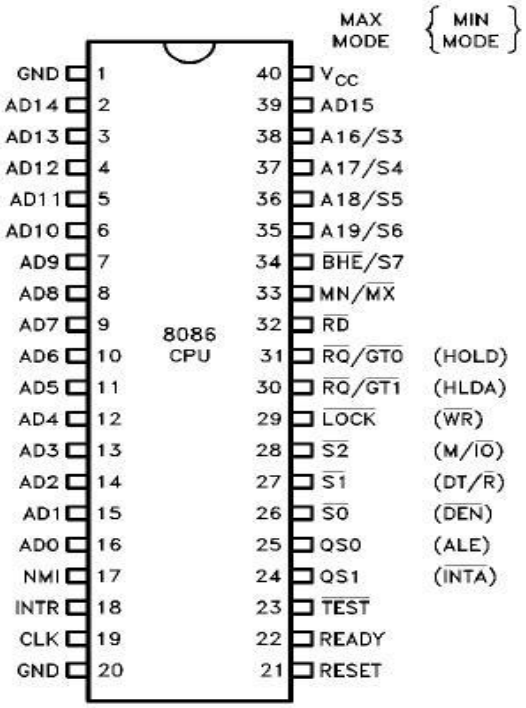
a) List salient features of microprocessor 8085 (any 8).

Ans.: Silent features of 8085 :(1/2 Marks for each)

1. 16 address line so 2^{16}=64 Kbytes of memory can be addressed.
2. Operating clock frequency is 3MHz and minimum clock frequency is 500KHz.
3. On chip bus controller.
4. Provide 74 instructions with five addressing modes.
5. 8085 is 8 bit microprocessor.
6. Provides 5 level hardware interrupts and 8 software interrupts.
7. It can generate 8 bit I/O address so 2^8=256 input and 256 output ports can be accessed.
8. Requires a single +5 volt supply.

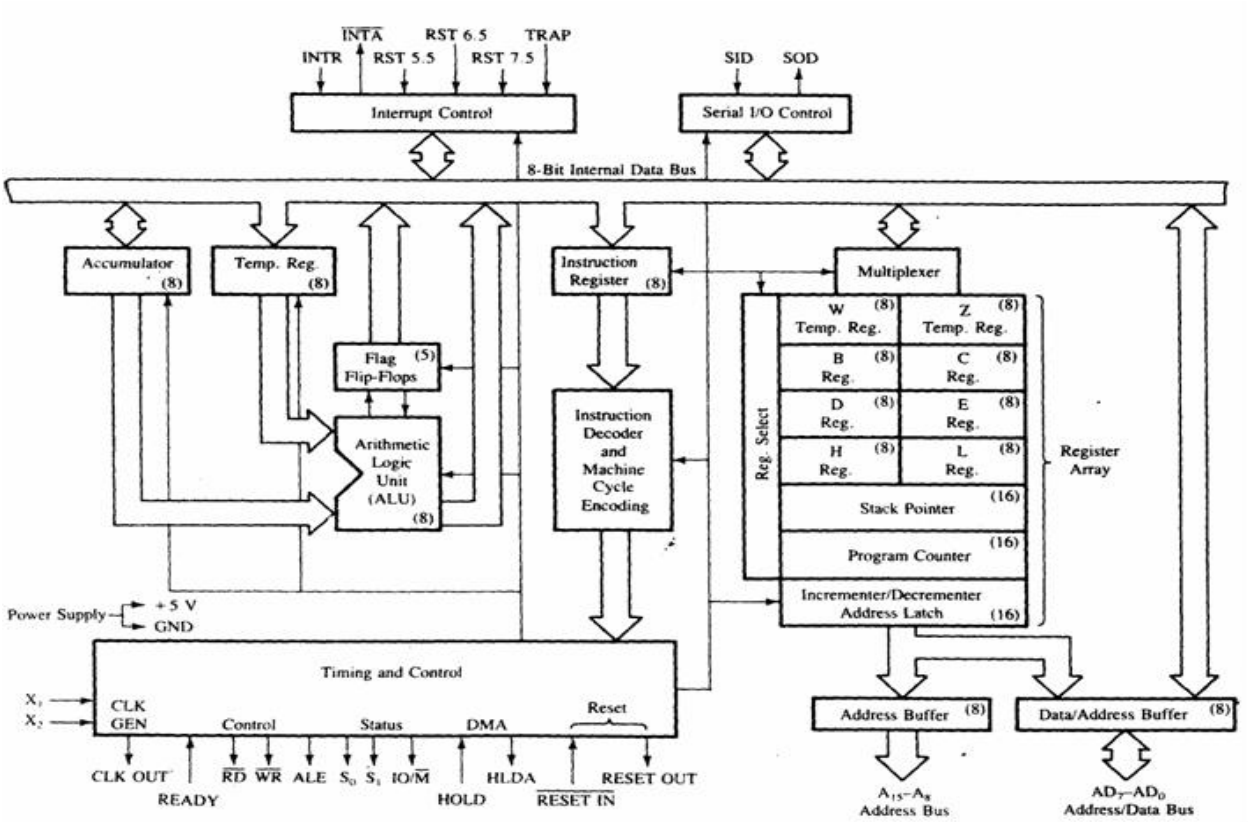
b) Draw the pin diagram of 8086. (4M)

Ans.

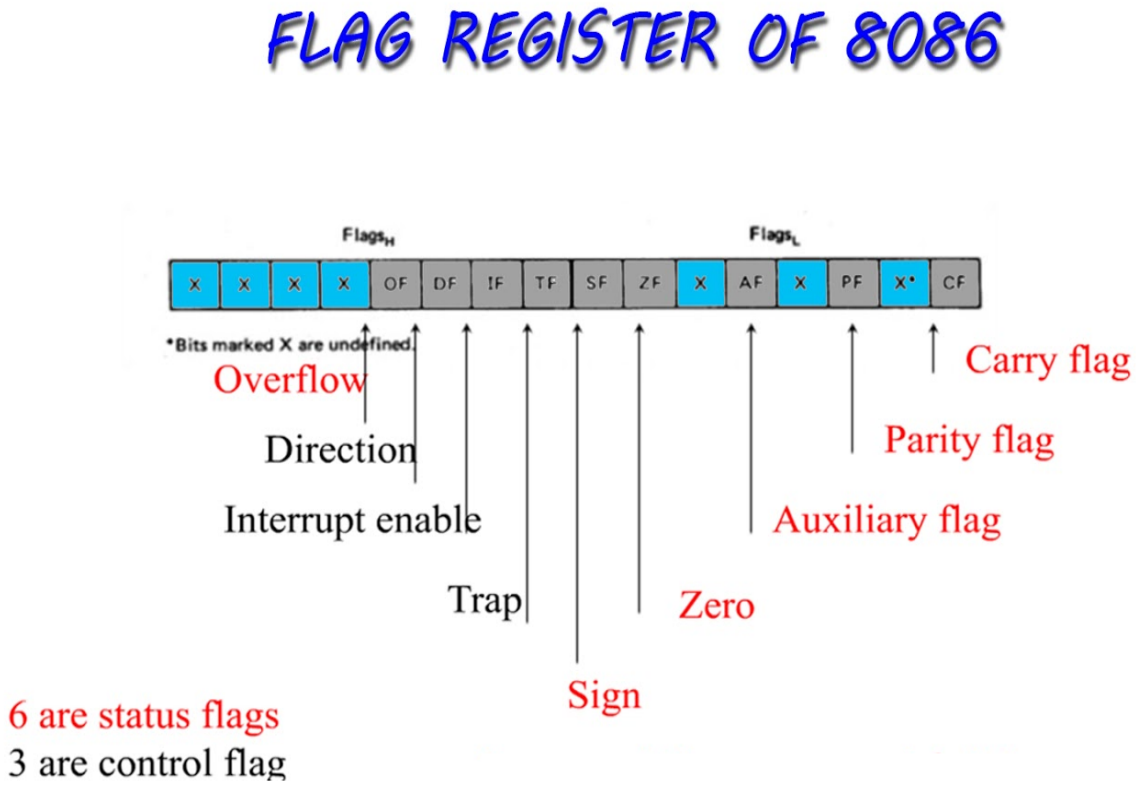


c) Draw the neat labeled functional block diagram of 8086.

Ans.:



d) Draw the flag register format of microprocessor 8086 and explain any two flags.
Ans.: (2 Mark for format, 2 Mark for explanation)



(Any two Flag explanation can be considered in Answer as per format.)

- **Carry flag:** This flag is set when there is a carry out of MSB in case of addition or a borrow in case of subtraction.
- **Parity flag:** This flag is set to 1if the lower byte of the result contains even numbers of 1s
- **Auxiliary carry flag :** this is set if there is a carry from the lowest nibble, i.e. bit three during addition or borrow for the lowest nibble i.e. bit three during subtraction.
- **Zero flag:** This flag is set if the result of the computation or comparison performed by previous instruction is zero.
- **Sign flag:** this flag is set when the result of any computation is negative. For signed computations the sign flag equals the MSB of the result.
- **Trap flag** If this flag is set the processor enters the single step execution mode.
- Interrupt flag: If this flag is set the maskable interrupts are recognized by the cpu, otherwise they are ignored.
- **Direction flag:** This flag is used by string manipulation instructions. If this flag bit is '0' the string is processed beginning from the lowest address to the highest address, otherwise the string is processed from the highest address towards lowest address.
- **Overflow flag:** This flag is set if an overflow occurs.

e) If AL,BL and CL content 10H, 10H and 20H respectively .state the effects of following instructions

i) CMP BL,CL

ii) XCHG AL,CL

Ans.: (2M each)

AL 10H, BL 10H CL 20H

i) CMP BL, CL:

This instruction compares the source operand with a destination operand. For comparison, it subtracts the source operand from destination operand .i.e. it compare 20H TO 10 in above CL subtracted from BL I.E 20H from 10H

ii) XCHG AL, CL: Exchange the content of CLTO AL i.e. 20H TO 10H (2 Mark)

f) Calculate physical address in the following cases: (2M each)

i) CS: 1200H, IP:DE00H

ii) DS: 1F00, BX:1A00 for MOV AX,[BX]

Ans.:

1) Segment address ----→ 1200H ------→ 0001 0010 0000 0000

Offset address ------→DE00H -----→ 1101 1110 0000 0000

Segment address shifted by 4 bit position-→ 1101 1110 0000 0000 0000

\+ 0001 0010 0000 0000

Physical address-----→1101 1111 0010 0000 0000

D F 2 0 0

2) Segment address ----→ 1F00H ------→ 0001 1111 0000 0000

Offset address ------→1A00H -----→ 1101 1010 0000 0000

Segment address shifted by 4 bit position-→ 0001 1111 0000 0000 0000

\+ 1101 1010 0000 0000

Physical address-----→0001 1111 1010 0000 0000

1 F A 0 0

Q.3. Attempt any Four of the following:

a) Compare the following instruction(2 Points)

i. AND and TEST

ii. AAA and DAA

(each Point 1 Mark)

i. AND and TEST

AND	TEST
The source and destination are ANDed and the result is stored in the destination.	The test instruction ANDs the source and the destination operands but does not store the result nor does it change the operands
The result in the destination after ANDing is used for further calculations.	This instruction is used to set the flags before conditional jump instructions.

ii. AAA and DAA

AAA	DAA
AAA stands for ASCII adjust after addition used in ASCII Addition.	DAA stands for Decimal adjust accumulator used in BCD addition
After the addition of ASCII equivalent of two decimal numbers, result is stored in AL register.	After the addition of two packed BCD numbers, the DAA instruction is used to adjust the hexadecimal result to its BCD equivalent.

b) State the functions for the following Pins of 8086.

i. $\overline{TEST}$

ii. $\overline{BHE}$

iii. $\overline{INTA}$

iv. DT / $\overline{R}$

Ans.:

Functions of each pin – 1M

i) $\overline{TEST}$ **:**

This input is examined by the Wait instruction. If this is low, execution continues otherwise processor waits in idle state.

ii) $\overline{BHE}$**:**

Bus High Enable is used to enable data onto most significant half of the data bus. This signal also signifies the byte which is read from even address or odd address.

iii)$\overline{INTA}$**:**

Interrupt Acknowledge is used as read signal for interrupt acknowledge given by the microprocessor.

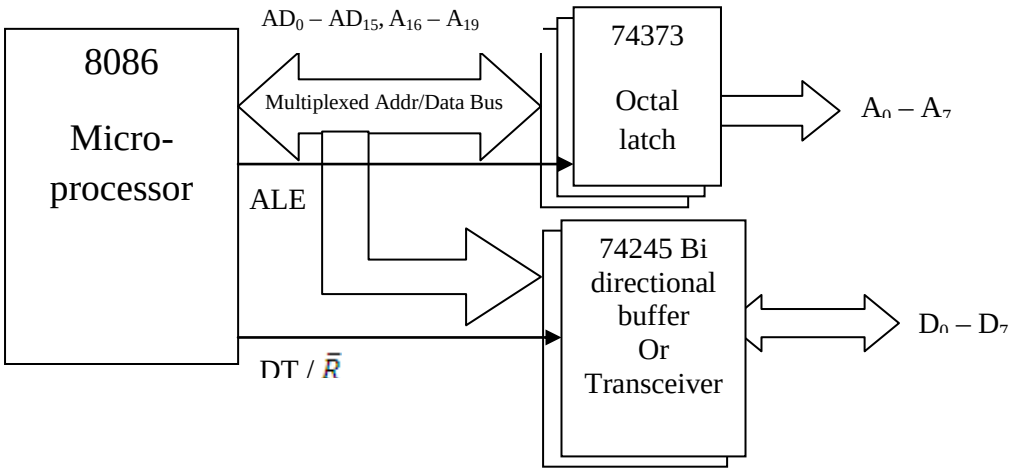
iv) DT / $\overline{R}$

Data Transmit / Receive. It is used to control the direction of data flow through the transceiver 8286 or 74LS245.When the processor sends the data out, this signal is high and when this signal is low the data is received.

c) Explain demultiplexing of address and data bus using a neat labeled diagram.
Explanation – 1M; Diagram: multiplexed lines at microprocessor drawn -1M; ALE & DT/$\overline{R}$ signals -1M; overall connections -1M)

Note: Demultiplexing in 8085 or 8086 or in general can also be considered.)

Ans.:



The data and address buses are multiplexed as AD_0- AD_7 in 8085 and AD_0 – AD_{15} in 8086. In the first clock cycle of a read or write cycle, the address is taken on the address/data bus using the Octal latch 74373. The ALE signal is used to strobe the address, after which the address/data bus becomes the data bus. When ALE is zero, the data lines are taken through the transceiver. The diagram above shows the de-multiplexing of address line and data lines in 8086.

d) Compare maximum mode and minimum mode configurations of 8086(any four points). (each point 1 Mark)

Ans.:

Sr. No.	**Minimum mode**	**Maximum mode**
1.	MN/$\overline{MX}$ pin is connected to VCC i.e. MN/$\overline{MX}$ = 1.	MN/$\overline{MX}$ pin is connected to ground i.e. MN/$\overline{MX}$ = 0.
2.	Control system M/$\overline{IO}$, $\overline{RD}$, $\overline{WR}$ is available on 8086 directly.	Control system M/$\overline{IO}$, $\overline{RD}$, $\overline{WR}$ is not available directly in 8086.
3.	Single processor in the minimum mode system.	Multiprocessor configuration in maximum mode system.
4.	In this mode, no separate bus controller is required.	Separate bus controller (8288) is required in maximum mode.
5.	Control signals such as $\overline{IOR}$, $\overline{IOW}$, $\overline{MEMW}$, $\overline{MEMR}$ can be generated using control signals M/$\overline{IO}$, $\overline{RD}$, $\overline{WR}$ which are available on 8086 directly.	Control signals such as $\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$ and $\overline{AIOWC}$ are generated by bus controller 8288.
6.	ALE, $\overline{DEN}$, DT/$\overline{R}$ and $\overline{INTA}$ signals are directly available.	ALE, $\overline{DEN}$, DT/$\overline{R}$ and $\overline{INTA}$ signals are not directly available and are generated by bus controller 8288.

7.	HOLD and HLDA signals are available to interface another master in system such as DMA controller.	$\overline{RQ}/GT0$ and $\overline{RQ}/GT1$ signals are available to interface another master in system such as DMA controller and coprocessor 8087.
8.	Status of the instruction queue is not available.	Status of the instruction queue is available on pins QS0 and QS1 .

e) Write an ALP for comparing two strings of 10 bytes each.
(General program outline – 1M; correct Logic & instructions -3M)

Note: Program from the manual or any simple program may be considered.
Ans.:

```
PRINT MACRO MES
 MOV AH,09H
 LEA DX,.ES
 INT 21H
ENDM

Data segment
MS1 DB 10,13,"Enter First string:$"
MS2 DB 10,13,"Enter Second string:$"
MS3 DB 10,13,"Equal Strings$"
MS4 DB 10,13,"Unequal Strings$"
MS5 DB 10,13,"$"
BUFF DB 25,?, 25 DUP ('$')
BUFF1 DB 25,?,25 DUP('$')

Data Ends

Code Segment
Assume cs:code, ds:data
Start:  MOV AX,DATA
 MOV DS,AX
 MOV ES,AX
 PRINT MS1     ;1ST NO.
 MOV AH,0AH
 LEA DX,BUFF
 INT 21H
 LEA SI,BUFF
 PRINT MS2     ;2ND NO.
 MOV AH,0AH
 LEA DX,BUFF1
 INT 21H
 MOV CL,BUFF+1       ;CHK LENGTH
 MOV CH, BUFF1+1
 CMP CH,CL
 JNZ uneq
 MOV CH,00H  ;COMPARE
 LEA DI,BUFF1
 CLD
REPE    CMPSB
 JNZ uneq
 PRINT MS3     ;PRINT RESULT
 JMP FINISH
```

```
uneq:    PRINT MS4
FINISH:MOV AH,4CH
 INT 21H

Code ends
End start
```

OR

```
DSEG SEGMENT
        ST1 DB 'TEXT'
        ST2 DB 'TEST'
        R DB ?
DSEG ENDS

CSEG SEGMENT
ASSUME CS:CSEG,DS:DSEG
START:MOV AX,DSEG
                MOV DS,AX
                MOV CX, 04H
                MOV BX,0000H
    BACK:       MOV AL,ST1[BX]
                CMP AL,ST2[BX]
                JNZ UNEQ
                INC BX
                LOOP BACK
                MOV R,'Y'
                 JMP EN
    UNEQ:       MOV R,'N'
    EN:         MOV AH,4CH
                INT 21H
CSEG ENDS
END START
```

OR

```
DATA SEGMENT
ST1 DB "TESTING PG"
ST2 DB "TEXTING PG"
L DB 0AH
R DB ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA, ES:DATA
  START:        MOV AX,DATA
                MOV DS,AX
                MOV ES,AX
                MOV CL,L
                 CLD
                MOV SI, OFFSET ST1
                 MOV DI, OFFSET ST2
                 REPE CMPSB
                 JNZ NEQ
                 MOV R,'E'
                 JMP LAST
```

```
    NEQ:    MOV R, 'N'
    LAST:   HLT
CODE ENDS
END START
```

f) Compare microprocessors 8085 and 8086(any four points) (Each comparison -1M)

	8085	8086
1	8 bit microprocessor	16 bit microprocessor
2	16 address lines	20 address lines
3	Total memory supported 64 KB	Total memory supported 1MB
4	8 bit operations can be performed using 8 bit registers an ALU	16 bit as well as 8 bit operations can be performed using 16 bit registers an ALU
5	Operates in single mode	Designed to operate in two modes, Maximum and Minimum
6	Doesn't support multiprogramming	Supports Multiprogramming
7	Doesn't support pipelining	Supports pipelining
8	Memory segmentation is not implemented	Memory segmentation is implemented
9	Supports 5 hardware interrupts	Supports 2 hardware interrupts
10	Has Five Addressing modes	Has 12 addressing modes
11	Single unit Microprocessor	Architecture has 2 units EU and BIU

Q.4. Attempt any Four of the following:

a) Explain following string instructions and respective prefix:

i. REP MOV SW

ii. REPE CMP SB

Ans.: (2M each)

i. MOVSW

- The MOVSW instruction is used to transfer a word from the source string to the destination string.
- The source must be in the data segment and the destination in the extra segment.
- The offset of the source word must be placed in SI register, which is represented as DS:SI and the offset of the destination word must be placed in DI register, which is represented as ES:DI
- On the execution of the instruction, SI and DI are automatically adjusted by one to the next element of the source and destination.
- If the Direction Flag is reset (DF = 0), the registers SI and DI will be incremented by two for the word movement.
- If the Direction Flag is set (DF = 1), the registers SI and DI will be decremented by two for the word movement.
- In multiple byte or word moves, the count must be loaded in CX register which functions as a counter.

ii. REP (Instruction Prefix)

- The REP instruction prefix is used in string instructions and interpreted as "Repeat while not end of string"
- In REP prefix, CX register is loaded with the count.
  Operation performed :--
  While CX <>0, perform the string operation
  CX ←CX - 1

iii. CMPSB

- The CMPSB instruction is used to compare a byte in the source string with a byte in the destination string.
- The source must be in the data segment and the destination in the extra segment.
- The offset of the source byte must be placed in SI register, which is represented as DS:SI and the offset of the destination byte must be placed in DI register, which is represented as ES:DI
- On the execution of the instruction, SI and DI are automatically adjusted by one to the next element of the source and destination.
- If the Direction Flag is reset (DF = 0), the registers SI and DI will be incremented by one for the byte comparison. If the Direction Flag is set (DF = 1), the registers SI and DI will be decremented by one for the byte comparison.
- In multiple byte or word comparison, the count must be loaded in CX register which functions as a counter.
- All conditional flags are affected

REPE/REPZ (Instruction Prefix)

- The REPE instruction prefix is used in string instructions and interpreted as "Repeat while not end of string and string equal" (CX<>0 and ZF =1)
- In REPE prefix, CX register is loaded with the count.
  Operation performed :--
  While CX <>0 & ZF -1 ,
  perform the string operation CX ←CX - 1

b) Identify the addressing modes used in the following instructions:

i. MOV DS,AX
ii. MOV AX, [4172H]
iii. ADD AX, [SI]
iv. ADD AX, [SI] [BX] [04H]

Ans.: (Each Addressing Mode -1M)

i) MOV DS,AX
Mode : Register Addressing Mode

ii) MOV AX, [4172H]
Mode: Direct Addressing Mode

iii) ADD AX, [SI]
Mode: Register Indirect or Indexed Mode

iv) ADD AX, [SI] [BX] [04H]
Mode: Relative Based Indexed Mode

c) Write an ALP to Divide a 16 bit number by a 8 bit number.(Program 4Marks)
Ans.:

```
DATA SEGMENT
        N1 DW 0101H
        N2  DB 10H
        Q DB ?
        R DB ?
DATA ENDS
CODE SEGMENT
ASSUME DS:DATA, CS:CODE
START:MOV DX,DATA
      MOV DS,DX
      MOV AX, N1
      MOV BL,N2
      DIV BL
      MOV Q,AL
      MOV R,AH
      HLT
CODE ENDS
END START
```

d) Write an ALP to find sum of first 10 integers.(Program 4Marks)
Ans.:

```
DATA SEGMENT
        SUM DB 0
DATA ENDS
CODE SEGMENT
ASSUME DS:DATA, CS:CODE
START:MOV DX,DATA
      MOV DS,DX
      MOV AL, 0AH
NEXT:   ADD SUM,AL
      DEC AL
      JNZ NEXT
      HLT
CODE ENDS
END START
```

e) What is a MACRO? Define a MACRO with an example.
Ans.: (Definition- 1M; Example- 3M)
(Note: any example with proper declaration and call from main program may be considered)

Small sequences of codes of the same pattern repeated frequently at different places which perform the same operation on the different data of the same data type are called MACRO.
Macro is also called as an Open subroutine. When called, the code written within macro are executed automatically. Macros should be used when it has few program statements. This simplifies the programming process.

Example: To find the square of a number N.

```
SQUARE MACRO NUM
            MOV AL,NUM
            MUL AL
            MOV BX,AX
SQUARE ENDM

CODE SEGMENT
                :
            SQUARE N    ; CALL MACRO SQUARE
            MOV R,BX
                :
```

f) Write an ALP to count number of 0's in AL register.(Program 4 Marks)

Ans.:

```
DATA SEGMENT
    N DB 37H
    Z DB 0
DATA ENDS
CODE SEGMENT
ASSUME DS:DATA, CS:CODE
START:MOV DX,DATA
      MOV DS,DX
      MOV AL, N
      MOV CL,08
NEXT:  ROL AL,01
      JC ONE
      INC Z
ONE:   LOOP NEXT
      HLT
CODE ENDS
END START
```

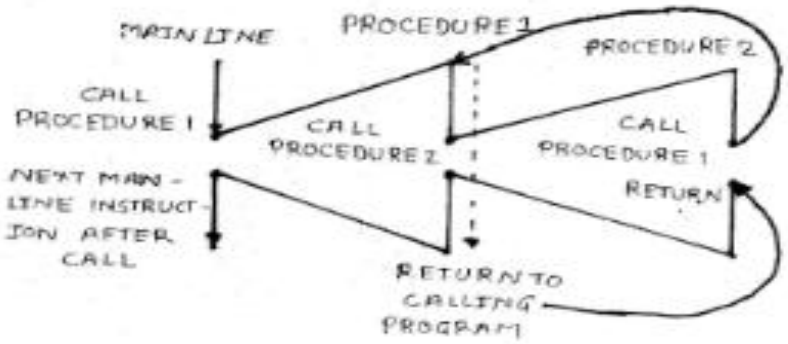
Q.5. Attempt any FOUR of the following:

a) Differentiate between re-enterant and recursive procedures.(two point)
(Any relevant 2 points 2 marks each)

Ans.: In some situation it may happen that Procedure 1is called from main program
Procrdure2 is called from procedure1And procrdure1 is again called from procdure2.
In this situation program execution flow re enters in the procedure1. These types of procedures are called re enterant procedures.

The RET instruction at the end of procrdure1 returns to procedure2. The RET instruction at the end of procedure2 will return the execution to procedure1.Procedure1 will again executed from where it had stopped at the time of calling procrdure2 and the RET instruction at the end of this will return the program execution to main program.
The flow of program execution for re-entrant procedure is as shown in FIG.

Sketch :



Recursive Procedure

A recursive procedure is a procedure which calls itself. Recursive procedures are used to work with complex data structures called trees. If the procedures is called with N (recursion depth) = 3. Then the n is decremented by one after each procedure CALL and the procedure is called until n = 0. Fig. shows the flow diagram and pseudo-code for recursive procedure.

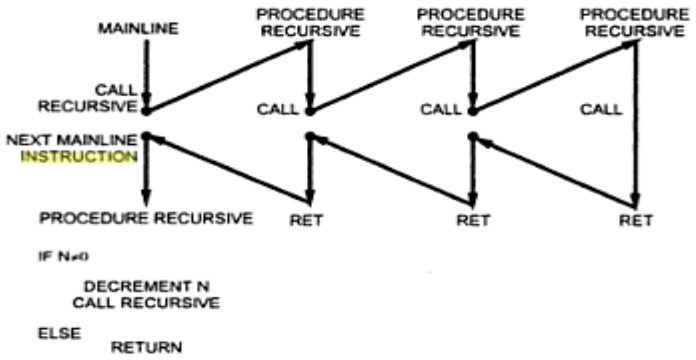


Fig. Flow diagram and pseudo-code for recursive procedure

b) Write appropriates 8086 instructions to perform following operations

i. Initialize stack at 42000 H

Ans. : (Instruction to initialize stack segment 1 mark, instruction to initialize stack pointer 1 mark)

```
MOV AX,,4200h
MOV  SS,AX
MOV SP,0000H
```

ii. Rotate register BX right 4 times.

(instruction to initialize counter 1 mark, rotate instruction 1 mark)

```
MOV CL,04H
RCR BX,CL
or
MOV CL,04H
ROR BX,CL
```

c) Write on ALP to transfer block to 10 numbers from one location to another location. (Program 4 marks)

Ans.: (Note: ALP to transfer data with string or without string can considered)

ALP to transfer block of data without using string instruction

```
DATA   SEGMENT
BLOCK1 DB 01H,02H,03H,04H,05H,06H,07H,08H,09H,10H
BLOCK2  DB 10 DUP(0)
DATA ENDS
CODE   SEGMENT
ASSUME CS:CODE ,DS:DATA

START:   MOV DX,DATA
        MOV DS, DX
        LEA SI, BLOCK1
        LEA DI, BLOCK2
        MOV CX, 000AH
       BACK:MOV AH,[SI]
        MOV [DI],AH
        INC SI
        INC DI
        DEC CX
        JNZ BACK
        MOV AH, 4CH
        INT 21H
        CODE ENDS
        END START
```

OR

An ALP to transfer a block of 10 data bytes using string instructions

```
DATA SEGMENT
STRNO1 DB 10 DUP(?)
DATA ENDS
EXTRA SEGMENT
STRNO2 DB 10 DUP(0)
EXTRA ENDS
CODE SEGEMENT
ASSUME CS: CODE, DS: DATA, ES: EXTRA
MOV DX, DATA
MOV DS, DX
MOV DX, EXTRA
MOV ES, DX
LEA SI, STRNO1
LEA DI, STRNO2
MOV CX, 000AH
CLD
REP MOVSB
MOV AH, 4CH
INT 21H
CODE ENDS
END
```

d) Find the error in the following program and correct them

```
MOV AX, 1000 H
MOV DS, AX
MOV BX, 2000 H
MOV ES, BX
MOV SI, 3000 H
MOV DI, 4000 H
REPE MOV SB
```

Ans.: (Listing errors 1 mark each, corrected error 1 mark each)
Errors: Counter register not initialized (logical error)
Syntax error in MOV SB (no space between MOV and SB)
(**Note: space between number and H (1000 H, also considered as error**)

Corrected program

```
MOV AX,1000H
MOV DS,AX
MOV BX,2000H
MOV ES,BX
MOV CL,05H ( counter initialization)
MOV SI,3000H
MOV DI,4000H
REPE MOVSB
```

e) Write an ALP to check a number to be odd or even
(Program 4 marks)

Ans.:

```
DATA SEGMENT
NUM DB 02H
EVEN DB 00H
ODD DB 00H
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:
MOV DX,DATA
MOV DS,DX
MOV CL,01H
MOV AL,NUM
 ROR AL,1
JC DN
INC EVEN
JMP T1
DN : INC ODD
T1: CODE ENDS
END START
```

f) List instruction formats of 8086 and explain any one.(4 marks)

Ans.: Machine Language instruction format:

A machine language instruction format has one or more number of fields associated with it.

The first field is called the opcode field, which indicate the type of operation performed by the CPU. The second field is called as an operand field which specifies the data on which the operation is to be performed.

The general format of 8086 instruction with different fields is as given below.

8086 Instructions are represented as binary numbers Instructions require between 1 and 6 bytes.

byte	7	6	5	4	3	2	1	0	
1	opcode						d	w	Opcode byte
2	mod		reg			r/m			Addressing mode byte
3	[optional]								low disp, addr, or data
4	[optional]								high disp, addr, or data
5	[optional]								low data
6	[optional]								high data

This is the general instruction format used by the majority of 2-operand instructions. Bytes 1 and 2 are divided up into 6 fields:

opcode
d -direction (or s = sign extension)
w -word/byte
mod -mode
reg -register
r/m -register/memory

opcode field specifies the operation performed (mov, xchg,etc)

d (direction) field specifies the direction of data movement:

- d = 1 data moves from operand specified by R/M field to operand specified by REG field
- d = 0 data moves from operand specified by REG field to operand specified by R/M field

W (word/byte) specifies operand size

W = 1 data is word

W = 0 data is byte

Q.6. Attempt any TWO of the following:

a) Draw and describe the maximum mode diagram of 8086.

Ans.: (Diagram 4 marks, explanation 4 marks)

- In the maximum mode, the 8086 is operated by strapping the MN/$\overline{MX}$ pin to ground.
- In this mode, the processor derives the status signal S2, S1, S0. Another chip called bus controller derives the control signal using this status information.
- In the maximum mode, there may be more than one microprocessor in the system configuration.
- The components in the system are latches, trans-receiver and clock generator, memory and I/O devices.
- Latches are generally buffered output D-type flip-flop like 74LS373 of 8282 .latches are used for separating valid addresses from the multiplexer address /data signal and controlled by ALE signal generated by 8086.
- Transceiver 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 DEN- and DT-/R . DEN signal indicated that the valid data is available on data bus. DT/R indicates the direction of data i.e. from or to the processor

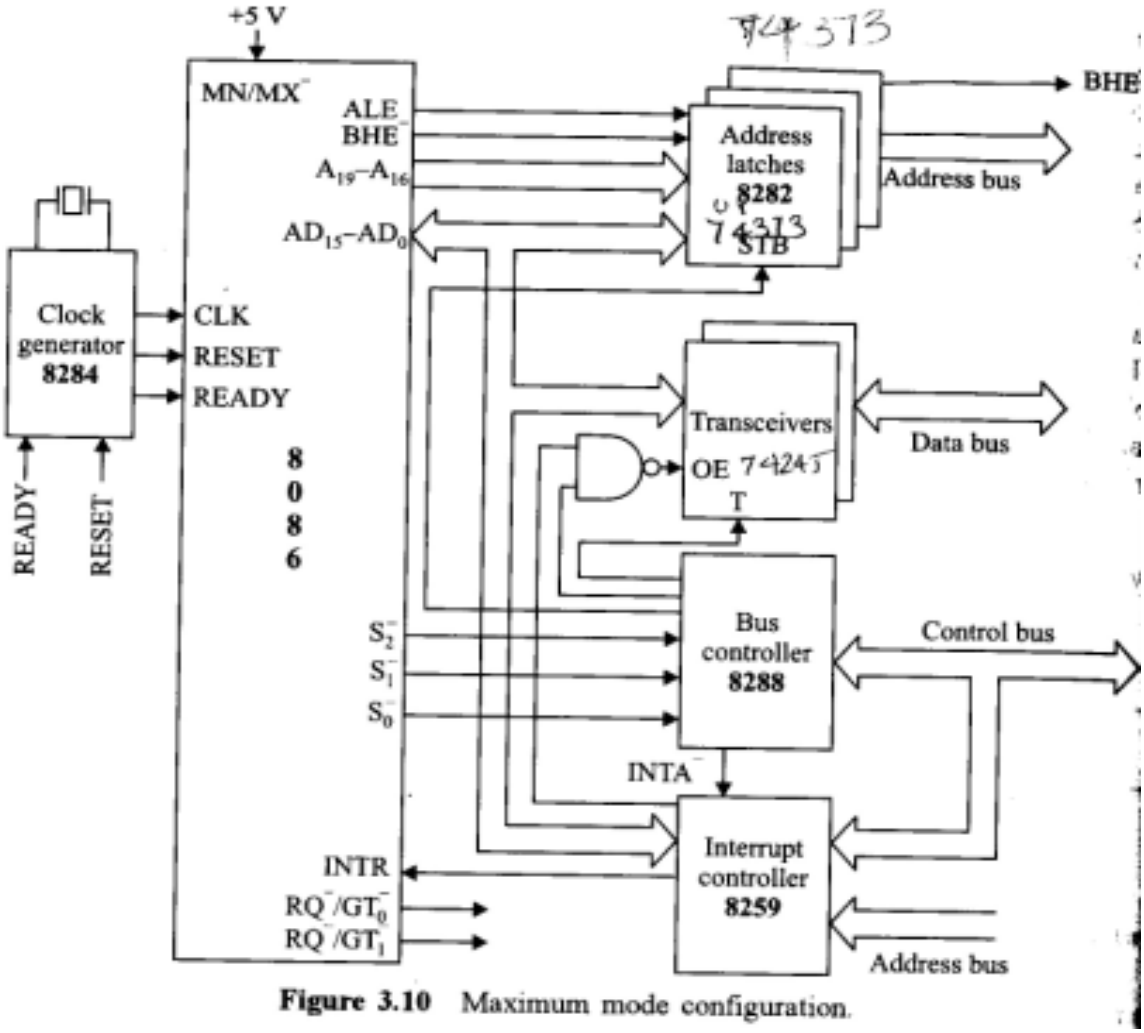


Figure 3.10 Maximum mode configuration.

The basic function of the bus controller chip IC8288, is to derive control signals like RD and WR (for memory and I/O devices), DEN, DT/R, ALE etc. using the information by the processor on the status lines.

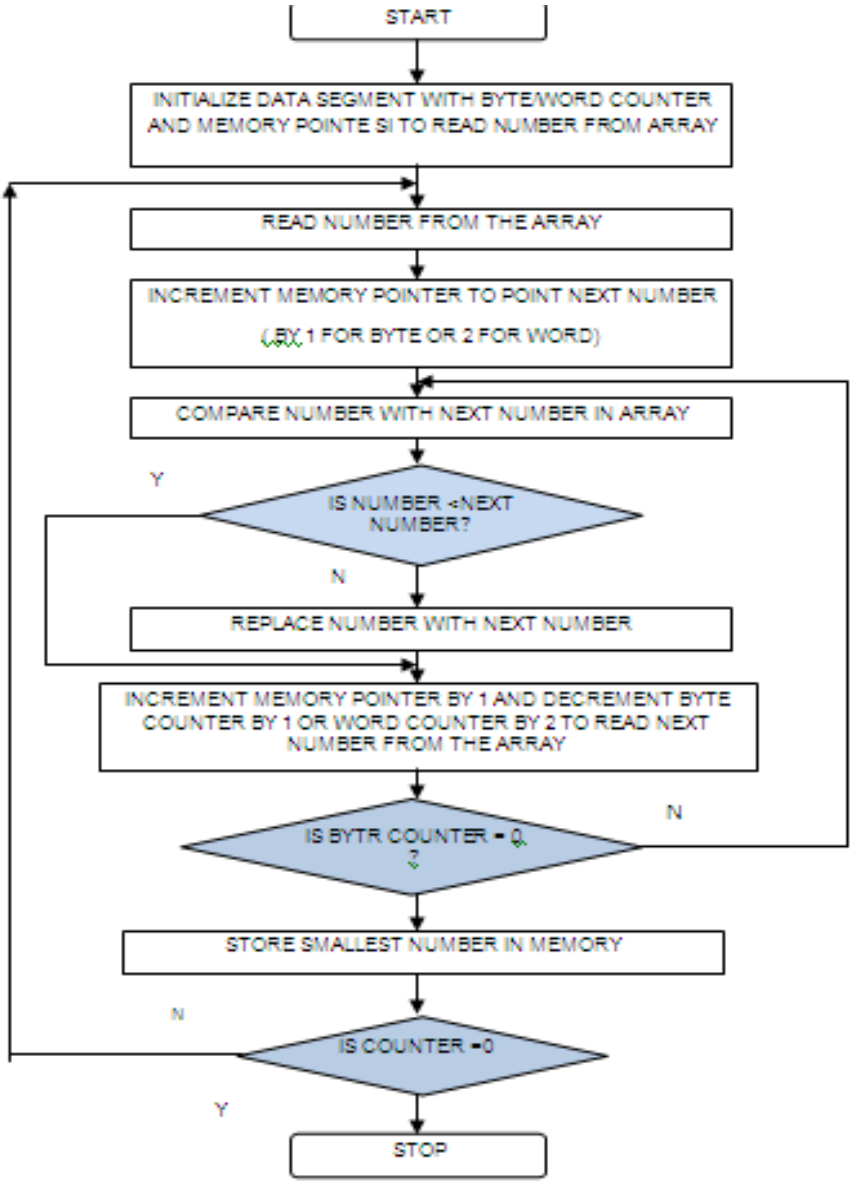
 The bus controller chip has input lines S2, S1, S0 and CLK. These inputs to 8288 are driven by CPU.

 It derives the outputs ALE, DEN, DT/R, MRDC, MWTC, AMWC, IORC, IOWC and AIOWC. The AEN, IOB and CEN pins are especially useful for multiprocessor systems.

b) **Write an ALP to arrange any array of 10 bytes in an ascending order. Also draw the flow chart for the same.**

Ans.: (program 6 marks, flowchart 2 marks)

```
DATA SEGMENT
ARRAY   DB   15H,05H,08H,78H,56H,   10H,   11H,
13H, 12H, 09H
DATA ENDS
CODE SEGMENT
START:   ASSUME CS: CODE, DS: DATA

        MOV DX, DATA
        MOV DS, DX
       MOV BL,0AH
 STEP1:   MOV SI,OFFSET ARRAY

             MOV  CL,09H

  STEP:     MOV AL,[SI]
            CMP AL,[SI+1]

     JC DOWN
      XCHG AL,[SI+1]
      XCHG AL,[SI]
DOWN :     ADD SI,1
       LOOP STEP
       DEC BL


       JNZ STEP1

CODE ENDS
END START
```



c) Explain various ways of parameter passing in 8086 assembly language procedure.
(For ways to pass parameter 2 marks each)

Ans.: Process of sending data/ parameter to a procedure while calling procedure in the main program is known as passing parameters to procedure.

There are four ways to pass the parameters:

1. Using registers: simple program with just few parameters
2. Using general memory: in case if we don't want to use registers
3. Using pointers: for passing arrays or other data structures
4. Using stack: for procedures in multiuser system program, procedures that will be called from a high level language program , or procedures that call themselves

1. Passing parameters through the registers:

Process of sending data/ parameter to a procedure using General purpose registers like AX, BX, CX, DX while calling the procedure in the main program is known as passing parameters through the registers to procedure.

The data to be processed is stored in registers and these registers are accessed by the procedure.

It is sued if the parameters are few in no.

2. **Passing parameters through memory:**

Process of sending data to a procedure using memory while calling the procedure in the main program is known as passing parameters through the registers to procedure.

The data is to be processed is stored in the memory locations and these memory locations are accessed in the procedure to process the data.

It is used when the parameters are few and passing through register is not possible

3. Passing parameters using pointers:

Process of sending data/ parameter to a procedure using address while calling the procedure in the main program is known as passing parameters using pointers to procedure.

The data to be processed is stored in the memory locations and these memory locations are accessed in the procedure using pointers to access the data.

Used When no. of parameters to be passed are more.

4. Passing parameters using stack:

Process of sending data to a procedure using stack memory while calling the procedure in the main program is known as passing parameter(s) using stack memory to procedure.

The data is to be processed is stored(PUSH) in the stack memory locations and these memory locations are accessed in the procedure to process the data.

Here stack diagram is very important