MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2013 Certified)

SUMMER – 19 EXAMINATION

Subject Name: Microprocessor and Programming **Subject Code: 17431**

Model Answer

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. No.	Sub Q. N.	Answer	Marking Scheme
1	a)	**Attempt any Six of the following :**	**12 M**
	i	**List the general purpose register in 8085 micro processor.**	**2 M**
	Ans:	8-bit general purpose registers. B, C, D, E, H and L.(8-bit) **(OR)** Pair of two 8 bit register such as BC, DE and HL are used as 16 bit registers.	Correct list : 2M
	ii	**State number of data lines and number of address lines used in 8086 microprocessor**	**2 M**
	Ans:	Data lines: 16 (AD0-AD15 multiplexed address and data) Address lines :20	Each : 1M
	iii	**List the four addressing mode of 8086 microprocessor**	**2 M**
	Ans:	**Addressing modes of 8086** 1. Immediate 2. Direct 3. Register 4. Register indirect	Any 4:For each mode : ½ M

<table>
<tr><td></td><td></td><td>5. Indexed
6. Register relative
7. Based indexed
8. Relative based indexed
9. Implied</td><td></td></tr>
<tr><td></td><td>iv</td><td>Define flowchart and algorithim.</td><td>2 M</td></tr>
<tr><td></td><td>Ans:</td><td>Algorithm: Algorithm is a task or sequence of operations performed by program.

Flowchart: Flow chart is a graphical representation of task or algorithm.</td><td>Each definition : 1M</td></tr>
<tr><td></td><td>v</td><td>State the use of following pin of 8085

1)HOLD

2)ALE</td><td>2 M</td></tr>
<tr><td></td><td>Ans:</td><td>1) HOLD: Hold is an active high input signal line used by the other master controller for gaining the control of address, data and control buses

2)ALE: It is an Address Latch Enable signal. It is used to separate address signals from data signals It goes high during first T state of a machine cycle and enables the lower 8-bits of the address, if its value is 1, otherwise data bus is activated.</td><td>Each signal: 1M</td></tr>
<tr><td></td><td>vi</td><td>List the names of segment register of 8086.</td><td>2 M</td></tr>
<tr><td></td><td>Ans:</td><td>Segment registers in 8086 microprocessor:

1. Code Segment register(CS)
2. Data Segment register(DS)
3. Stack Segment register(SS)
4. Extra Segment register(ES)</td><td>For each register : ½ M</td></tr>
<tr><td></td><td>vii</td><td>Give any two difference between NEAR and FAR procedure.</td><td>2 M</td></tr>
<tr><td></td><td>Ans:</td><td>
<table>
<tr><th>Sr.no</th><th>Near procedure</th><th>Far Procedure</th></tr>
<tr><td>1.</td><td>A near procedure refers to a procedure which is in the same code segment from that of the call instruction</td><td>A far procedure refers to a procedure which is in the different code segment from that of the call instruction.</td></tr>
<tr><td>2.</td><td>It is also called intra-segment procedure</td><td>It is also called inter-segment procedure call</td></tr>
<tr><td>3</td><td>A near procedure call replaces the old IP with new IP.</td><td>A far procedure call replaces the old CS:IP pairs with new CS:IP pairs</td></tr>
</table>
</td><td>Any 2 points : 1 M each</td></tr>
</table>

		4.	The value of old IP is pushed on to the stack. SP=SP-2 ;Save IP on stack(address of procedure)	The value of the old CS:IP pairs are pushed on to the stack SP=SP-2 ;Save CS on stack SP=SP-2 ;Save IP (new offset address of called procedure)
		5.	Less stack locations are required	More stack locations are required
		6.	Example :- Call Delay	Example :- Call FAR PTR Delay
	viii	**Write instruction of 8086 microprocessor to:** **1)Substract 50H from the content of AX register.** **2)Rotate the content of AX towards right by 2 bit position.**		**2 M**
	Ans:	1) SUB AX,50H 2) MOV CL, 02H RCR AX, CL Or MOV CL, 02H ROR AX, CL		Each correct instruction:1 M
	b)	**Attempt any Six of the following :**		**8 M**
	i	**State the function of** **1) Editor** **2)Assembler** **3)Linker** **4)Debugger**		**4 M**
	Ans:	**1)Editor** 1. It is a program which helps to construct assembly language program with a file extension .asm, in right format so that the assembler will translate it to machine language. 2. It enables one to create, edit, save, copy and make modification in source file.		One function of each :1M

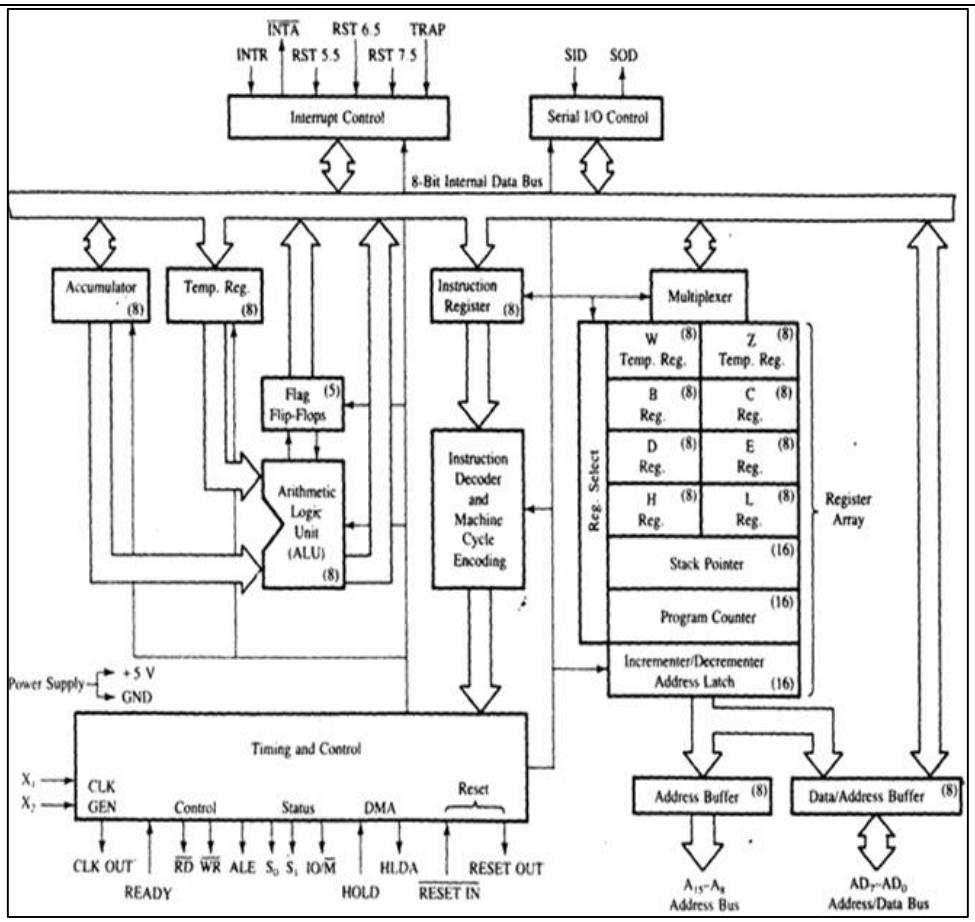
		2)Assembler 1. Assembler is a program that translates assembly language program to the correct binary code. 2. It also generates the file called as object file with extension .obj. 3. It also displays syntax errors in the program, if any. 4. It can be also be used to produce list(.lst) and .crf files **3)Linker** 1. It is a programming tool used to convert Object code (.OBJ) into executable (.EXE) program. 2. It combines, if requested, more than one separated assembled modules into one executable module such as two or more assembly programs or an assembly language with C program. **4)Debugger:** 1. A debugger is a program which allows us to load object code program into system memory execute the program, and debug it.	
	ii	**Describe the function of following directives:** **1)DD** **2)ASSUME** **3)ORG** **4)INCLUDE**	**4 M**
	Ans:	**1) DD: -Define Double word (32-bits)** It is used to declare a variable of type doubleword or to reserve memory locations which can be accessed as type doubleword (32-bits.) **Example:** NUMBER DD 1,2,3,4,9 ; allocated 20 memory locations. NUM DD? ;Allocate 4 memory locations **2) ASSUME: -** Assume directive is used to tell Assembler the name of the logical segment it should use for the specified segment. respective logical segments. **Example: -** Assume CS: MSBTE_CODE, DS: MSBTE_DATA	Correct function of each : 1M (example : optional)

		3) ORG: The directive ORG assigns the location counter with value specified in the directive. It helps in placing the machine code in the specified location while translating instructions into machine codes by the assembler. $ is used to indicate current value of location counter **Syntax: ORG [$+] Numeric_value** **Example:** ORG 2000H ; set location counter to 2000H 4) **INCLUDE-Include source code code from file** This directive used to tell the assembler to insert a block of source code from named file into current source module. **Syntax** :INCLUDE <file path specification with file name> **Example INCLUDE C:\Tasm\Macro.lib**	
	iii	**Write an assembly program using recursive procedure to find factorial of a number**	**4 M**

	Ans:	DATA SEGMENT
		N DB 04H
		RES DW?
		DATA ENDS
		CODE SEGMENT
		ASSUME CS:CODE,DS:DATA
		START: MOV AX,DATA
		MOV DS,AX
		MOV AL,N
		MOV AH,00H
		CALL FACT
		MOV AH,4CH
		INT 21H
		
		FACT PROC
		CMP AX,01 ;IF N=1,FACT=1 ELSE FACT=N*FACT(N-1)
		JZ EXIT
		PUSH AX
		DEC AX ; N-1
		CALL FACT ; N*FACT(N-1)
		POP AX
		MUL RES
		MOV RES,AX ;RES=FACTORIAL
		RET
		EXIT:
		MOV RES,01
			Correct logic : 2M
			
			Correct syntax : 2M

		RET **FACT ENDP** CODE ENDS END START	
2.		**Attempt any Four of the following**	**16 M**
	a	**Enlist interrupt pins of 8085 microprocessor with its function.**	**4 M**
	Ans:	The 8085 has five interrupt signals that can be used to interrupt a program execution. 1. INTR 2. RST 7.5 3. RST 6.5 4. RST 5. 5. TRAP **1. TRAP** It is a non-maskable interrupt, having the highest priority among all interrupts. Bydefault, it is enabled until it gets acknowledged. In case of failure, it executes as ISR and sends the data to backup memory. This interrupt transfers the control to the location 0024H. **2. RST7.5** It is a maskable interrupt, having the second highest priority among all interrupts. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 003CH address. **3. RST 6.5** It is a maskable interrupt, having the third highest priority among all interrupts. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 0034H address. **4. RST 5.5** It is a maskable interrupt. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 002CH address.	List : 1M, function of any 3: 1M each

		5. INTR It is a maskable interrupt, having the lowest priority among all interrupts. It can be disabled by resetting the microprocessor.	
	b	**State any eight features of 8086 microprocessor.**	**4 M**
	Ans:	1. 20 bit address lines so 2^{20} = 1Mbyte of memory can be addressed and data bus is 16 bit,. 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. The instruction set is powerful, flexible and can be programmed in high level language like C language. 5. Can operate in single processor and multiprocessor configuration i.e. operating modes. 6. Provides 6-bytes instruction queue for pipelining of instructions executions. 7. Provides 256 types of vectored software interrupts. 8. Operate in maximum and minimum mode to achieve high performance level. 9. Provides separate instructions for string manipulation. 10. Generate 8 bit of 16 bit I/O address so it can access maximum 64K I/O devices. 11. Operate in maximum and minimum mode to achieve high performance. 12. **8086 uses memory banks:-**The 8086 uses a memory banking system. It means entire data is not stored sequentially in a single memory of 1 MB but memory is divided into two banks of 512KB. **Interrupts:-**8086 has 256 vectored interrupts.	Any 8 features : 1/2 Mark each
	c	**Describe memory segmentation in 8086 microprocessor. Give any two advantages of segmentation.**	**4 M**
	Ans:	**Memory Segmentation**: The memory in 8086 based system is organized as segmented memory. 8086 can access 1Mbyte memory which is divided into number of logical segments. Each segment is 64KB in size and addressed by one of the segment register. The 4 segment register in BIU hold the 16-bit starting address of 4 segments. CS holds program instruction code. Stack segment stores interrupt & subroutine address. Data segment stores data for program. Extra segment is used for string data. **Advantages of segmentation** 1) With the use of segmentation the instruction and data is never overlapped. 2) The major advantage of segmentation is Dynamic relocation of program (code segment) which means that a program can easily be	Description: 2 M, Any 2 Advantages : 1 M each

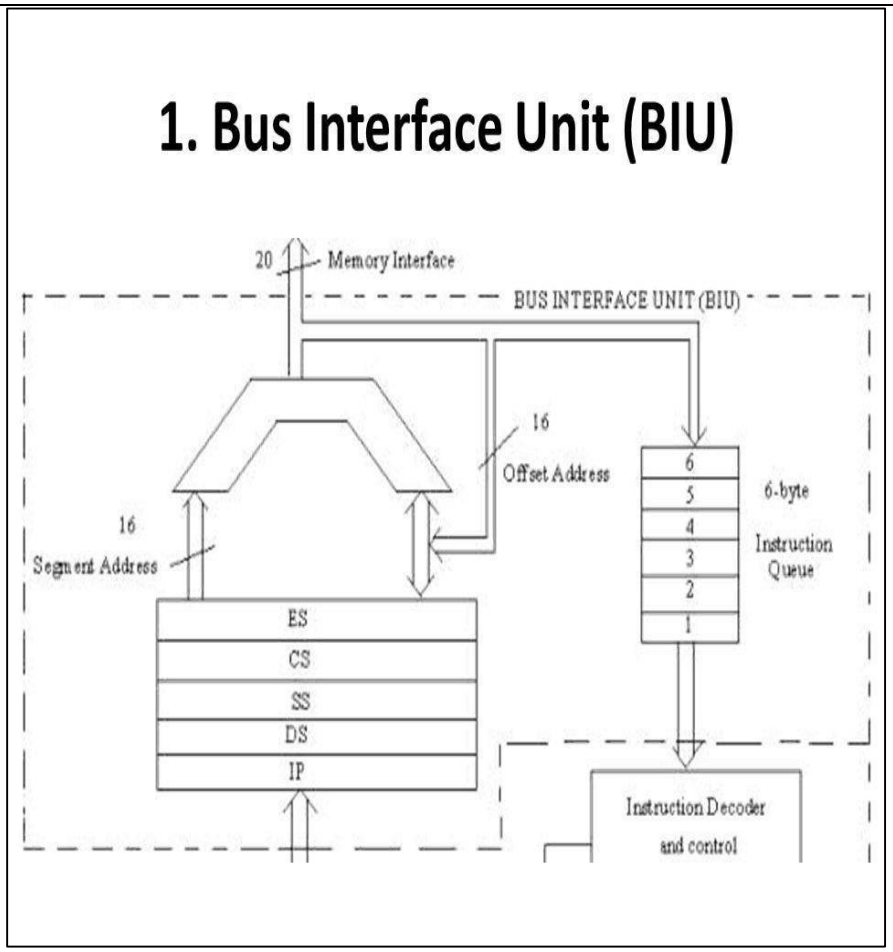
		transferred from one code memory segment to another code memory segment without changing the effective address. 3) Segmentation can be used in multi-user time shared system. 4) Segmentation allows two processes to share data. 5) Segmentation allows you to extend the addressability of a processor i.e., address up to 1MB although the actual addresses to be handled are of 16 bit size. 6) Programs and data can be stored separately from each other in segmentation.	
	d	**Draw the detailed architecture of 8085.**	**4 M**
	Ans:		Correct diagram : 4M
	e	**Write 8086 instruction for following** **i) Multiply AL by 4 using shift rotation.** **ii)Move 1234H into DSregister.**	**4 M**
	Ans:	i)MOV CL,02H SHL AL,CL	Correct instruction : 2M each

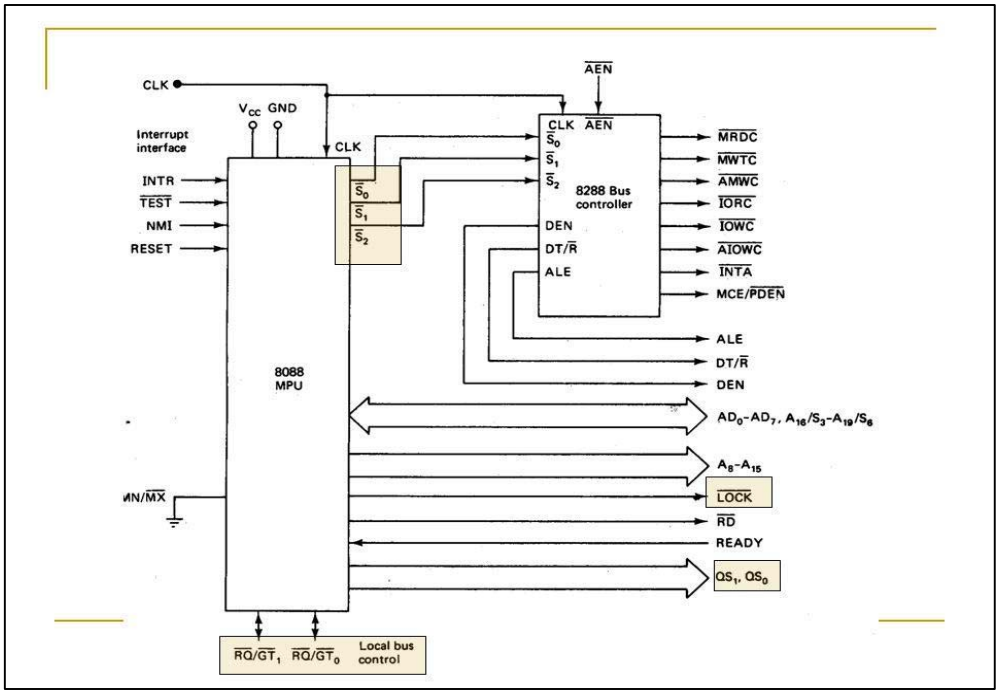
		(OR)
		MOV CL,02H
		SAL AL,CL
		
		ii)MOV AX,1234H
		MOV DS,AX	

	f	**Calculate physical address in following cases:** **i)CS=79FBH and IP=8437H** **ii)DS:1FABH,BX:1A77H for MOV AX,(BX)**	**4 M**
	Ans:	i) CS =79FBH, IP =8437h CS= 79FB0H……. 0 added by BIU + IP= 8437H ---------- **823E7H** ii) DS =1FABH, BX =1A77H MOV AX, [BX] = MOV AX, DS:[BX] BX is used as address offset to a memory operand Physical address can be calculated as DS * 10H + BX. DS= 1FAB0H……. 0 added by BIU + BX= 1A77H ---------- **21527H**	Each correct address calculation : 2M
3.		**Attempt any Four of the following**	**16 M**
	a	**Explain any two string instruction with example.**	**4 M**
	Ans:	1] MOVS/ MOVSB/ MOVSW - Move String byte or word. Syntax MOVS destination, source MOVSBdestination, source MOVSWdestination, source Operation: ES:[DI]<----- DS:[SI] It copies a byte or word 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.CX register contain counter and direction flag (DF) will be set	for Any two String instructions: for each Instruction ½ mark for List and 1 mark for Syntax with Explanation and ½ Mark

		or reset to auto increment or auto decrement pointers after one move. e.g. MOVS m8, m8 Move byte at address DS:(E) SI to address ES :(E) DI. MOVS m16, m16 Move word at address DS:(E)SI to address ES:(E)DI. MOVSB Move byte at address DS:(E)SI to address ES:(E)DI. MOVSW Move word at address DS:(E)SI to address ES:(E)DI. MOVSD Move doubleword at address DS:(E)SI to address 2] CMPS /CMPSB/CMPSW: Compare string byte or Words. Syntax CMPS destination, source CMPSBdestination, source CMPSWdestination, source Operation: Flags affected < ----- DS:[SI]- ES:[DI] It compares a byte or word in one string with a byte or word in another string. SI holds the offset of source and DI holds offset of destination strings. CX contains counter and DF=0 or 1 to auto increment or auto decrement pointer after comparing one byte/word. e.g. CMPS m8, m8 Compares byte at address DS:(E)SI with byte at address ES:(E)DI and sets the status flags accordingly. CMPS m16, m16 Compares word at address DS:(E)SI with word at address ES:(E)DI and sets the status flags accordingly. CMPSB Compares byte at address DS:(E)SI with byte at address ES:(E)DI and sets the status flags accordingly. CMPSW Compares word at address DS:(E)SI with word at address ES:(E)DI and sets the status flags accordingly. 3] SCAS/SCASB/SCASW: Scan a string byte or word. *Syntax* SCAS/SCASB/SCASW Operation: Flags affected < ----- AL/AX-ES: [DI] It compares a byte or word in AL/AX with a byte /word pointed by ES:DI. The string	for Example

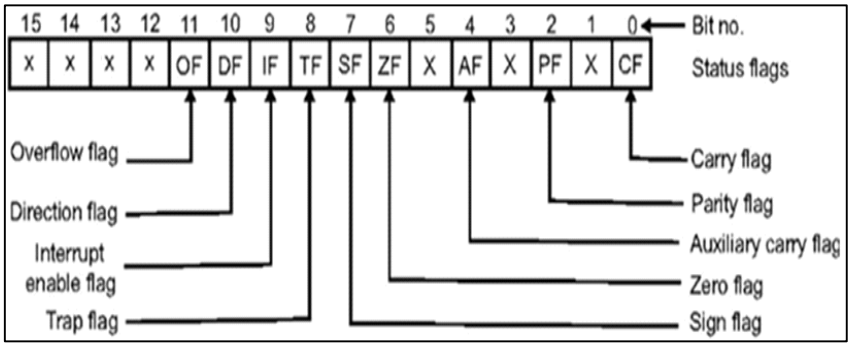
		to be scanned must be in the extra segment and pointed by DI. CX contains counter and DF may be 0 or 1. When the match is found in the string execution stops and ZF=1 otherwise ZF=0 e.g. SCAS m8 Compare AL with byte at ES:(E)DI and set status flags. SCAS m16 Compare AX with word at ES:(E)DI and set status flags. SCASB Compare AL with byte at ES:(E)DI and set status flags. SCASW Compare AX with word at ES:(E)DI and set status flags. 4] LODS/LODSB/LODSW: Load String byte into AL or Load String word into AX. Syntax: LODS/LODSB/LODSW Operation: AL/AX < ----- DS: [SI] IT copies a byte or word from string pointed by SI in data segment into AL or AX.CX may contain the counter and DF may be either 0 or 1 e.g. LODS m8 Load byte at address DS:(E)SI into AL. LODS m16 Load word at address DS:(E)SI into AX. LODSB Load byte at address DS:(E)SI into AL. LODSW Load word at address DS:(E)SI into AX. 5] STOS/STOSB/STOSW (Store Byte or Word in AL/AX) Syntax STOS/STOSB/STOSW Operation: ES:[DI] < -----AL/AX It copies a byte or word from AL or AX to a memory location pointed by DI in extra segment CX may contain the counter and DF may either set or reset. • Operation: ES:[DI] < -----AL/AX e.g. STOS m8 Store AL at address ES:(E)DI. STOS m16 Store AX at address ES:(E)DI. STOSB Store AL at address ES:(E)DI. STOSW Store AX at address ES:(E)DI.	
	b	**Draw and explain bus interface unit of 8086 microprocessor.**	**4 M**

	Ans:	 **BIU (Bus Interface Unit):** BIU takes care of all data and addresses transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direction connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus. It has the following functional parts – **Instruction queue:** BIU contains the instruction queue. BIU gets up to 6 bytes of next instructions and stores them in the instruction queue. When EU executes instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased execution speed. Fetching the next instruction while the current instruction executes is called pipelining. **Segment register:** BIU has 4 segment buses, i.e. CS, DS, SS & ES. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations it also contains 1 pointer register IP, which holds the address of the next instruction to executed by the EU. **(i) CS:** It stands for Code Segment it is used for addressing a memory location in the code segment of the memory, where the executable program	Diagram : 2 Marks, Explanation : 2 Marks

		is stored. **(ii) DS:** It stands for Data Segment. It consists of data used by the program and is accessed in the data segment by an offset address or the content of other register that holds the offset address. **(iii) SS:** It stands for Stack Segment. It handles memory to store data and addresses during execution. **(iii) ES:** It stands for Extra Segment. ES is additional data segment, which is used by the string to hold the extra destination data. **Instruction pointer:** It is a 16-bit register used to hold the address of the next instruction to be executed.	
	c	**Draw the interfacing of 8288 Bus controller with 8086. List and explain interfacing signal**.	**4 M**
	Ans:	 **Bus Status Codes** **Status Inputs** / **CPU Cycles** / **8288 Command** $\overline{S_2}$ $\overline{S_1}$ $\overline{S_0}$ 0 0 0 — Interrupt Acknowledge — $\overline{INTA}$ 0 0 1 — Read I/O Port — $\overline{IORC}$ 0 1 0 — Write I/O Port — $\overline{IOWC}$, $\overline{AIOWC}$ 0 1 1 — Halt — None 1 0 0 — Instruction Fetch — $\overline{MRDC}$ 1 0 1 — Read Memory — $\overline{MRDC}$ 1 1 0 — Write Memory — $\overline{MWTC}$, $\overline{AMWC}$ 1 1 1 — Passive — None	Correct Interfacing diagram: 2 Marks, List and Explanation : 2 Marks

	d	**Explain function of following instruction with one example.** **i)XLAT** **ii) LAHF**	**4 M**
	Ans:	**i)XLAT :** • XLAT replaces a byte in AL register with a byte from 256 byte lookup table beginning at [BX]. • AL is used as offset into this table. • Flags are not affected • operation :- AL=[BX+AL] Example : XLAT **ii) LAHF:** LAHF Instruction in 8086: Load lower byte of flag register in AH. This instruction copies the contents of lower byte of 8086 flag register to AH register. Moves the low byte of the EFLAGS register (which includes status flags SF, ZF, AF, PF, and CF) to the AH register. Reserved bits 1, 3, and 5 of the EFLAGS register are set in the AH register as shown in the "Operation" section below. Operation AH = EFLAGS(SF:ZF:0:AF:0:PF:1:CF); Example: LAHF	For each instruction: Explanation :1 ½ Marks, example : ½ Mark
	e	**Write an assembly language program to find the string length of a given string.**	**4 M**

	Ans:	DATA SEGMENT
		STR1 DB 'STUDENT$'
		LENGTH_STRING DB?
		DATA ENDS
		ASSUME CS:CODE, DS:DATA
		CODE SEGMENT
		MOV AX, DATA
		MOV DS, AX
		MOV AL, '$'
		MOV CX, 00H
		MOV SI, OFFSET STR1
		BACK: CMP AL, [SI]
		JE DOWN
		INC CL
		INC SI
			Correct program : 4M

		JMP BACK DOWN: MOV LENGTH_STRING, CL MOV AX, 4C00H INT 21H CODE ENDS END	
	f	**Draw flag register format of 8086. Explain Trap and Overflow flag.**	**4 M**
	Ans:	 1) **Trap flag (TF)-** It is used to set the trace mode i.e. start single stepping mode. Here the microprocessor is interrupted after every instruction so that the program can be debugged. 2) **Overflow flag (OF)-** It will be set if the result of a signed operation is too large to fit in the number of bits available to represent it.It can be checked using the instruction INTO (Interrupt on Overflow).	Diagram : 2 M, Explanation of each flag : 1 M each
4.		**Attempt any Four of the following**	**16 M**
	a	**Differentiate between following instruction.** **i) AAA,AAM** **ii) POP,POPF** **iii) LDS,LES** **iv) ROL,RCL**	**4 M**
	Ans:	**i)AAA,AAM** **AAA** | **AAM** ASCII Adjust after Addition | ASCII Adjust after Multiplication	1 M for 1 Difference for each point

		AAM is used adjust the addition to two unpacked BCD numbers | AAM is used adjust the product to two unpacked BCD numbers **ii) POP,POPF** **POP** | **POPF** Copies word from the stack pointed by the stack pointer to the destination | Copies word from the stack to the flag register Ex: POP CX | Ex: POPF **iii) LDS,LES** **LDS** | **LES** Load register and DS with words from memory location | Load register and ES with words from memory location LDS register, memory address of the first word | LES register, memory address of the first word Ex: LDS CX,[391AH] | Ex: LES CX,[391AH] **iv) ROL,RCL** **ROL** | **RCL** Rotate left byte or word | Rotate through carry left byte or word Syntax: ROL Destination, Count | Syntax: RCL Destination, Count Can be used to Swap the nibbles | Cannot be used to swap the nibbles.	
	b	**State the function of process control instruction.** **i. STC** **ii. CMC** **iii. STD** **iv. CLD**	**4 M**
	Ans:	i) **STC**- Set carry flag: This instruction Set Carry Flag. CF=1, STC does not affect any other flag. ii) **CMC**- Complement carry flag: This instruction Complement Carry Flag. CMC does not affect any other flag. iii) **STD**- Set direction flag: his instruction is used to set the direction flag to one so that SI and/or DI can be decremented automatically after execution of	For Each Instruction's function: 1 M (Maximum 4 M)

		string instructions. STD does not affect any other flags. This instruction Set Direction Flag. DF=1, iv) **CLD-** Clear direction flag: This instruction is used to reset the direction flag to zero, so that SI and/or DI can be incremented automatically after execution of string instructions. CLD does not affect any other flag. This instruction Clear Direction Flag. DF=0	
	c	**Write an assembly language program to mask the lower nibble of 8-bit number.**	**4 M**

	Ans:	.model small
		.data
		a dw 0012H
		.code
		mov ax, @data ; Initialize data section
		mov ds, ax
		mov ax, a ; Load number1 in ax
		and al, 0f0h ; mask lower nibble. Result in al
		mov ch, 02h ; Count of digits to be displayed
		mov cl, 04h ; Count to roll by 4 bits
		mov bh, al ; Result in reg bh
		up: rol bh, cl ; roll bl so that msb comes to lsb
		mov dl, bh ; load dl with data to be displayed
		and dl, 0fH ; get only lsb
		cmp dl, 09 ; check if digit is 0-9 or letter A-F
		jbe tr
		add dl, 07 ; if letter add 37H else only add 30H
		tr: add dl, 30H
		mov ah, 02 ; Function 2 under INT 21H (Display character)
		int 21H
		dec ch ; Decrement Count
		jnz up
		mov ah, 4ch
		int 21h
		end
			Correct program : 4M

	d	**Write an assembly language program to transfer block of 10 number from source i.e. 2000H to destination 3000H (No overlapped block transfer).**	**4 M**
	Ans:	. Model small . Data ORG 2000H Arr1 db 00h,01h,02h,03h,04h,05h,06h,07h,08h,09h	Correct program : 4M

		Count Equ 10 Dup
		Org 3000H
		Arr2 db 10 Dup(00h)
		Ends
		.code
		Start: Mov ax,@data
		Mov ds,ax
		Mov SI,2000H
		Mov DI,3000H
		Mov cx, count
		Back: Mov al, [SI]
		Mov [DI], al
		Inc SI
		Inc DI
		Dec cx
		Jnc Back
		Mov ah, 4ch
		Int 21h
		Ends
		End	

	e	**Write an assembly language program to add two 8bit BCD numbers.**	**4 M**

	Ans:	DATA SEGMENT
		NUM1 DB 09H
		NUM2 DB 09H
		SUM DB?
		DATA ENDS
		CODE SEGMENT
		START: ASSUME CS:CODE,DS:DATA
		MOV AX,DATA
		MOV DS,AX
			Correct program : 4M

		MOV AL,NUM1
		ADD AL,NUM2
		DAA Decimal adjust for addition
		
		MOV SUM,AL
		MOV AH,4CH
		INT 21H
		CODE ENDS
		END START
		
		(OR)
		.MODEL SMALL
		.DATA
		NUM1 DB 84H
		NUM2 DB 28H
		RES_LSB DB?
		RES_MSB DB?
		.CODE
		MOV AX,@DATA
		MOV DS,AX
		MOV AL,NUM1 ;
		MOV BL,NUM2
		ADD AL,BL ;Ans ACH
		DAA
		JNC DN
		INC RES_MSB
		DN:MOV RES_LSB,AL
		MOV AH,4CH
		INT 21H
		END	

	f	**Write an assembly language program to find largest number among block of data using macro.**	**4 M**
	Ans:	LrgMac MACRO Again: cmp al,[bl] Jnc skip Mov al ,[bl] Skip: inc bl Loop again Endm	Correct program : 4M

		.data
		Nums db 44h,55h,66h,77h,88h
		Count db 05h
		Largest db?
		.code
		Start: mov ax, @data
		Mov ds,ax
		Mov al,00h
		Mov cl, count
		Mov bl,nums
		LrgMac
		Mov largest , al
		Ends
		End	

5.		**Attempt any Four of the following**	**16 M**
	a	**Write an assembly language program to reverse string computer programming for 8086.**	**4M**

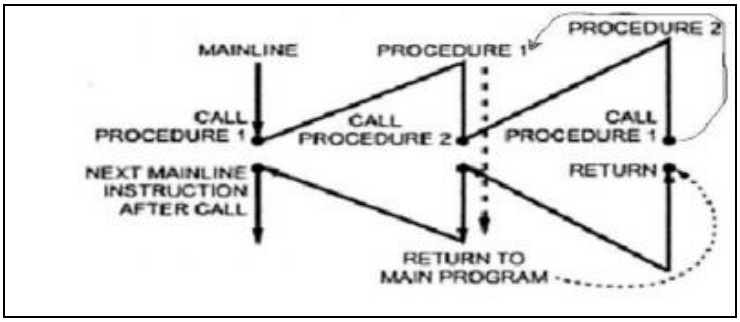
	Ans:	DATA SEGMENT
		STRB DB 'computer programming $'
		REV DB 0FH DUP (?)
		DATA ENDS
		CODE SEGMENT
		START:ASSUME CS:CODE,DS:DATA
		MOV DX,DATA
		MOV DS,DX
		LEA SI,STRB
		MOV CL,0FH
		LEA DI,REV
		ADD DI,0FH
		UP:MOV AL,[SI]
		MOV [DI],AL
		INC SI
		DEC DI
		LOOP UP
			Correct program : 4M

		MOV AH,4CH INT 21H CODE ENDS END START	
	b	**Write an assembly language program to multiply two 16-bit unsigned numbers.**	**4M**

	Ans:	DATA SEGMENT
		N1 DW 2401H
		N2 DW 1324H
		C DD?
		DATA ENDS
		CODE SEGMENT ASSUME CS: CODE, DS:DATA
		START:
		MOV AX,DATA
		MOV DS,AX
		MOV AX,N1
		MOV BX,N2
		MUL BX
		MOV WORD PTR C,AX
		MOV WORD PTR C+2,DX
		INT 21H
		CODE ENDS
		END START
			Correct program : 4M

	c	**Write an assembly language program to sort an array of 10 numbers in Descending order.**	**4M**
	Ans:	DATA SEGMENT ARRAY DB 15h,05h,08h,78h,56h, 60h, 54h, 35h, 24h, 67h DATA ENDS CODE SEGMENT START: ASSUME CS: CODE, DS:DATA MOV DX, DATA MOVDS,DX MOVBL,0AH step1:MOVSI,OFFSETARRAY MOVCL,09H step: MOV AL,[SI] CMP AL,[SI+1] JNC Down XCHG AL,[SI+1] XCHG AL,[SI] Down:Add SI,01h LOOP step DEC BL JNZ step1 CODE ENDS	Correct program : 4M

		END START	
	d	**Explain any four rotation instructions with example**	**4M**
	Ans:	**1. ROL –** Rotate bits of byte or word left, MSB to LSB and to CF **Syntax**: ROL destination, count **Eg**: **ROL BL,2** ; Rotate all bits in BL left by 1 bit, copy MSB to LSB and to CF IF BL = 11110000 After Execution 11000011, CF= 1 **2. ROR –** Rotate bits of byte or word right, LSB to MSB and to CF **Syntax**: ROR destination, count **Eg**: **ROR BL, 2** ; Rotate all bits in BL right by 1 bit ,copy LSB to MSB and to CF IF BL = 11110000 After Execution 00111100, CF= 0 **3. RCL –** Rotate bits of byte or word left, MSB to CF and CF to LSB. **Syntax**: RCL destination, count **Eg**: **RCL BL, 2** ; Rotate all bits in BL left by 1 bit ,copy MSB to CF and CF to LSB IF BL = 11110000, CF=0 After Execution 11000001 , CF= 1 **4. RCR –** Rotate bits of byte or word right, LSB to CF and CF to MSB. **Syntax**: RCR destination, count **Eg**: **RCR BL, 1;** Rotate all bits in BL right by 1 bit, copy LSB to CF and CF to MSB. IF BL = 11110000, CF= 0 After Execution 00111100 , CF= 0	Each instruction: 1 M
	e	**Explain re-entrant procedures with help of schematic diagram**	**4M**
	Ans:	The procedure which can be interrupted used and "reentered" without losing or writing over anything is called re-entrant procedure. In some situation it may happen that procedure1 is called from main program, procedure2 is called from 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 flow of program execution for reentrant procedure is shown in the diagram.	Explanation: 2M, Diagram:2M

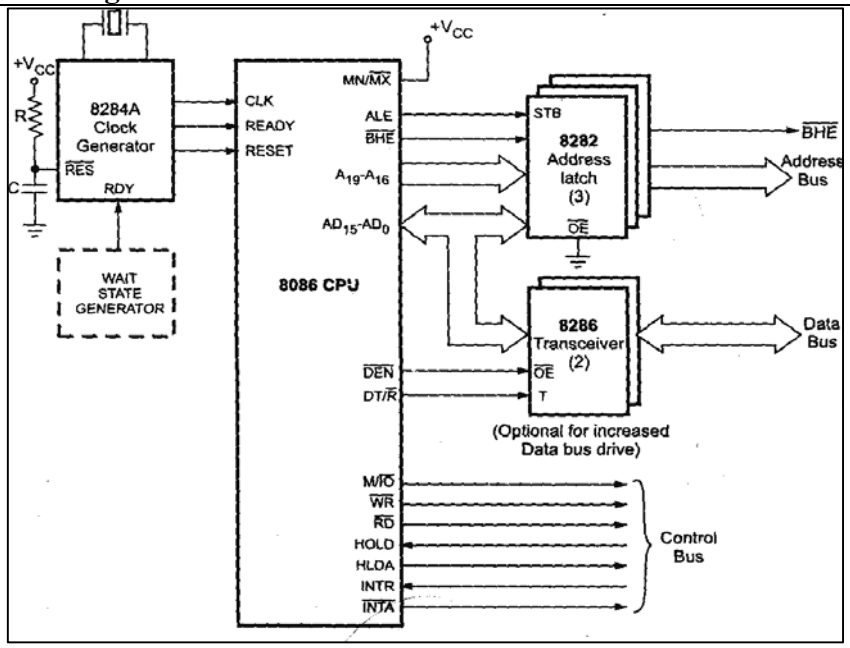


f **Differentiate between procedure and Macro**

Ans:

Sr.No.	MACRO	PROCEDURE
1	Macro is a small sequence of code of the same pattern, repeated frequently at different places, which perform the same operation on different data of the same data type.	Procedure is a series of instructions is to be executed several times in a program, and called whenever required.
2	The MACRO code is inserted into the program, wherever MACRO is called, by the assembler.	Program control is transferred to the procedure, when CALL instruction is executed at run time.
3	Memory required is more, as the code is inserted at each MACRO call	Memory required is less, as the program control is transferred to procedure.
4	Stack is not required at the MACRO call.	Stack is required at Procedure CALL.
5	No overhead time required.	Extra overhead time is required for linkage between the calling program and called procedure.
6	Parameter passed as the part of statement which calls macro.	Parameters passed in registers, memory locations or stack.
7	RET is not used	RET is required at the end of the procedure
8	Macro is called <Macro_name> [argument	Procedure is called using: CALL <Procedure_name>

Any 4 points 1M each

			list]		
		9	Directives used: MACRO, ENDM, LOCAL	Directives used: PROC, ENDP, FAR, NEAR	
6.		**Attempt any Two of the following**			**16 M**
	a	**With neat diagram describe minimum mode operation of 8086 microprocessor. List signals of maximum mode of 8086.**			**8M**
	Ans:			Diagram 3M ,Explanation 3M,List 2M

• When MN/ pin is in logic 1, the 8086 microprocessor operates in minimum mode system.

• In this mode, the microprocessor chip itself gives out all the control signals.

• This is a single processor mode.

The remaining components in the system are latches, transceivers, clock generator, memory or I/O devices.

• This system has three address latches (8282) and two octal data buffers (8286) for the Complete 20-bit address and 16 bit data Separation.

• The latches are used for separating the valid address from the multiplexed address/data signals and the controlled by the ALE signal generated by 8086.

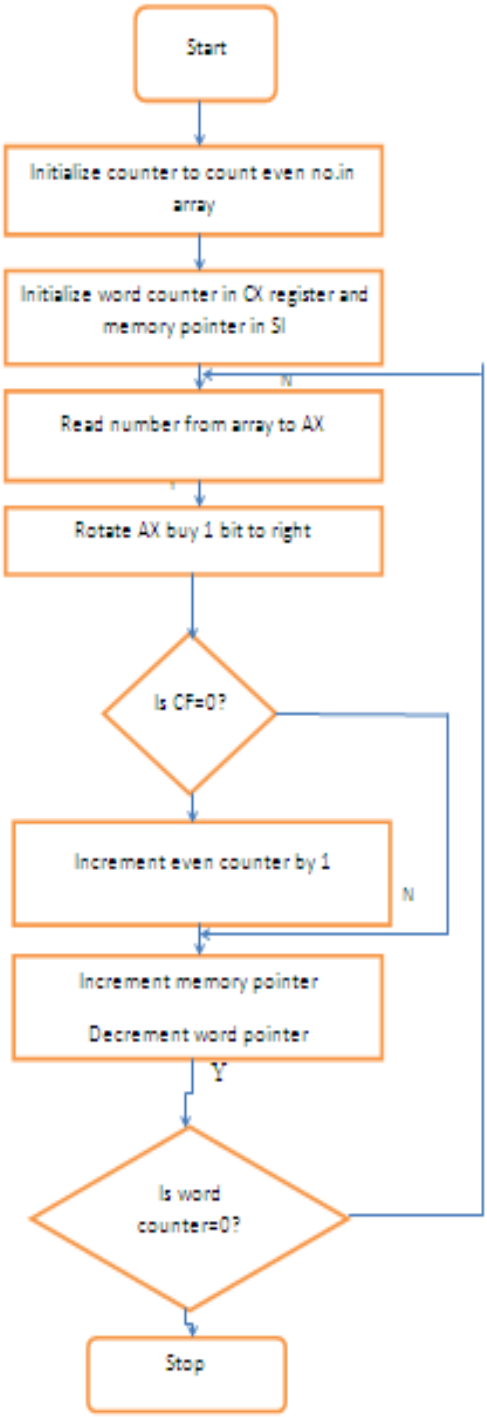
• Transceivers are the bi-directional buffers. They are required to separate the valid data from the time multiplexed address/data signal. This is

		controlled by two signals, DEN & DT/· • DT/ indicates that the direction of data, ie. from or to the microprocessor. • Signal indicates the valid data is available on the data bus. • This system contains memory for the monitor and users program storage. It also contains I/O devices to communicate with the processor. • The clock generator in the system is used to generate the clock and to synchronize some external signals with the system clock. The signals in maximum mode are : $\overline{MRDC}, \overline{MWTC}, \overline{AMWC}, \overline{IORC}, \overline{IOWC}, \overline{AIOWC}, \overline{INTA}$	
	b	**Write an assembly language program to count even number in an array of five 16 bit number. Also draw the flowchart for the same**	**8M**

	Ans:	DATA SEGMENT
		NUM DW 1200h,2345h,4567h,7864h,2587h
		COUNT DB?
		DATA ENDS
		CODE SEGMENT
		ASSUME CS:CODE, DS:DATA
		START: MOV AX,DATA
		MOV DS,AX
		MOV CX,14H
		MOV SI, OFFSET NUM
		NEXT: MOV AL, [SI]
		INC SI
		MOV AH,[SI]
		ROL AX,01
		JNC DOWN
		INC COUNT
		DOWN: INC SI
		LOOP NEXT
		MOV AX, 4C00H
		INT 21H
		CODE ENDS
			Correct program 4M,Correct flowchart 4M

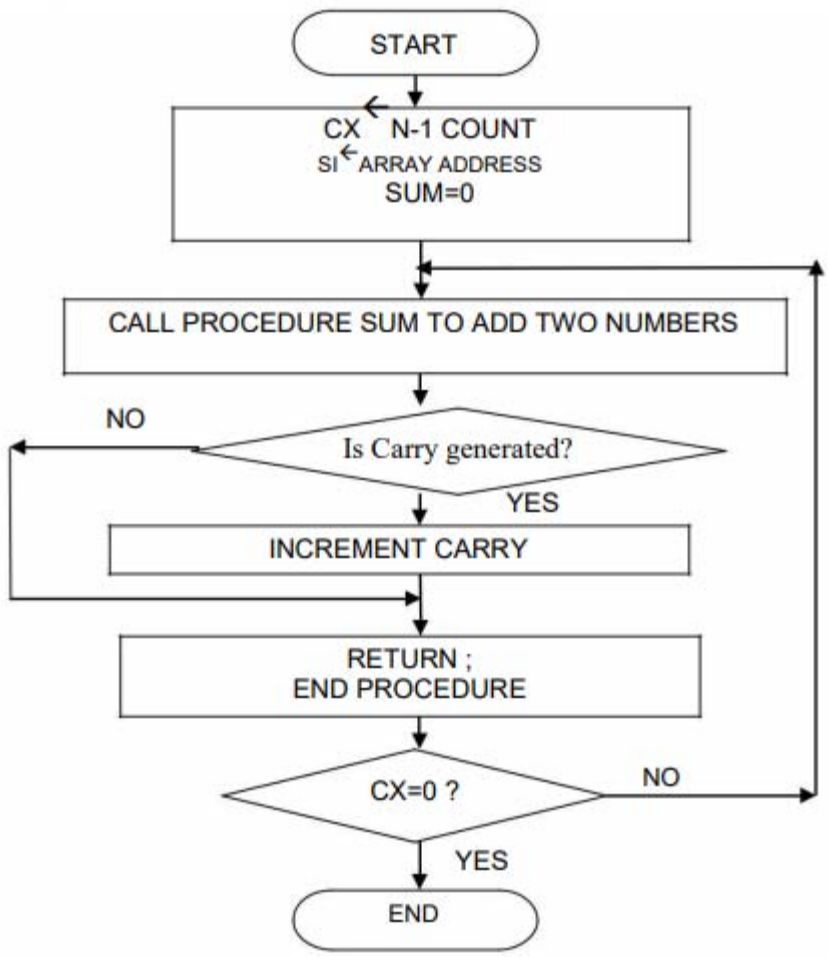
END START

Flowchart :



	c	**Write an assembly language program to add series of 5 number i.e. 8bit using FAR procedure. Also draw a flowchart for the same.**	**8M**

	Ans:	DATA SEGMENT
		NUM1 DB 10H,20H,30H,40H,50H
		RESULT DB 0H
		CARRY DB 0H
		DATA ENDS
		CODE SEGMENT ASSUME CS:CODE, DS:DATA
		START: MOV DX,DATA
		MOV DS, DX
		MOV CL,05H
		MOV SI, OFFSET NUM1
		UP:
		CALL SUM
		INC SI
		LOOP UP
		MOV AH,4CH
		INT 21H
		SUM PROC; Procedure to add two 8 bit numbers
		MOV AL,[SI]
		ADD RESULT, AL
		JNC NEXT
		INC CARRY
		NEXT: RET
		SUM ENDP
		CODE ENDS
		END START
		
		Flowchart :
			correct program 4M, correct flowchart 4M


START
CX ← N-1 COUNT
SI ← ARRAY ADDRESS
SUM=0
CALL PROCEDURE SUM TO ADD TWO NUMBERS
NO
Is Carry generated?
YES
INCREMENT CARRY
RETURN ;
END PROCEDURE
CX=0 ?
NO
YES
END