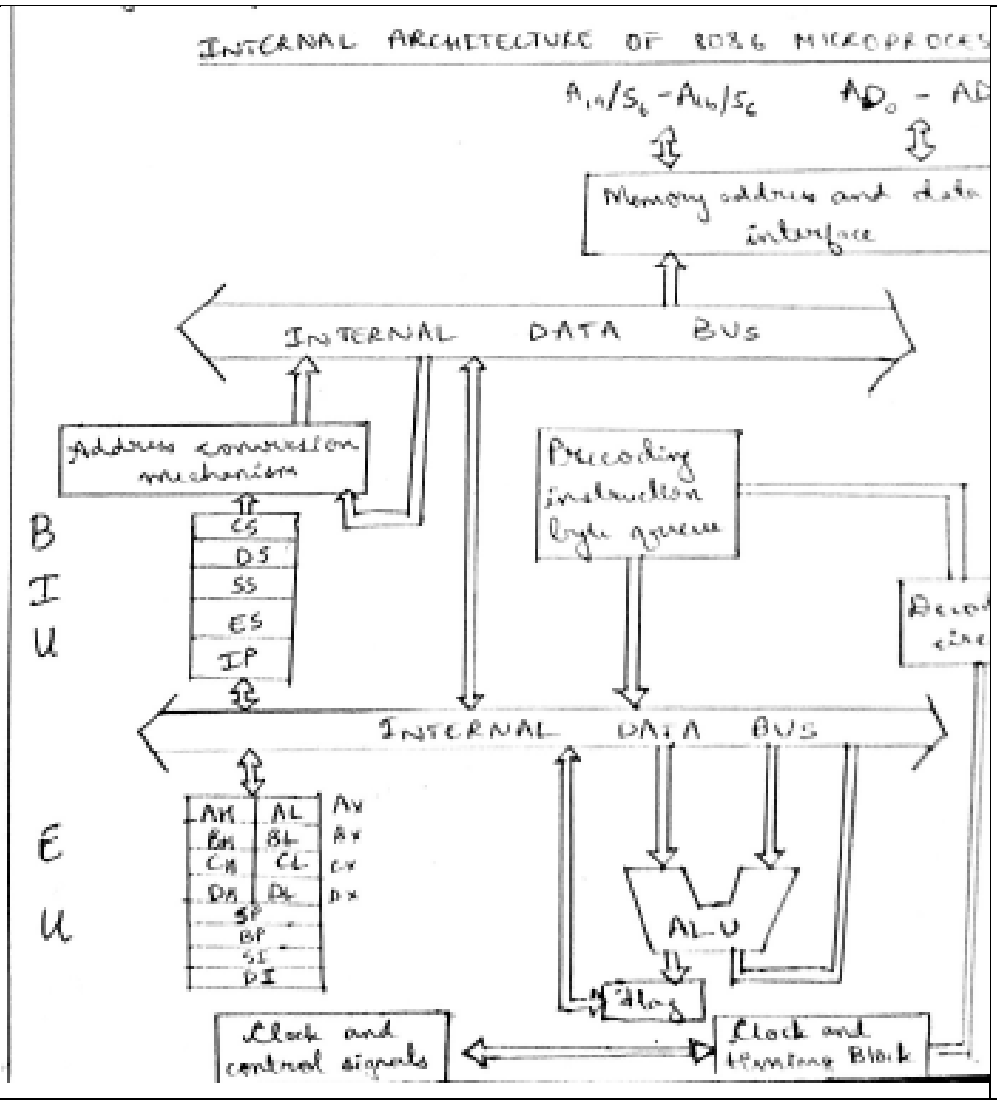
Scheme Of Evaluation
Internal Assessment Test 1 – Sept.2018

Sub:	Microprocessor							**Code:**	17EC46
Date**:**	07/03/2018	Duration:	90mins	Max Marks:	50	**Sem:**	IV	**Branch:**	ECE(A,B,C,D)/ TCE

Note: Answer Any Five Questions

Question #		**Description**	**Marks Distribution**		**Max Marks**
1		**Describe with neat diagram, the internal architecture of 8086 microprocessor.** • Diagram • BIU • EU	4 M 3 M 3 M	10 M	10 M


INTERNAL ARCHITECTURE OF 8086 MICROPROCES
$A_{19}/S_6 - A_{16}/S_3$
AD_0 - AD
Memory address and data interface
INTERNAL DATA BUS
Address generation mechanism
Decoding instruction byte queue
CS
DS
SS
ES
IP
B
I
U
INTERNAL DATA BUS
AH
AL
AX
BH
BL
BX
CH
CL
CX
DH
DL
DX
SP
BP
SI
DI
E
U
ALU
Flags
Clock and control signals
Clock and timing Block

Architecture of 8086
- (a) Bus Interface Unit (BI
- (b) Execution unit (EU)

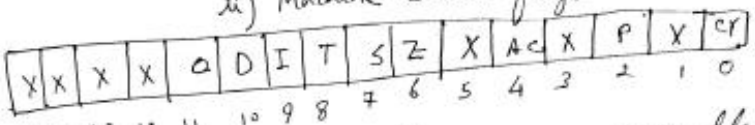
BIU: It contains circuit for physical address calculations and predecoding instruction byte queue (6

→ This unit is responsible for establishing communications with external devices and peripherals including memory via the bus.

- When the opcode is fetched and decoded, the external bus remains free for some time. This time slot is utilised in 8086 to achieve the overlapped fetch and execution cycles.
- While the fetched instruction is executed internally the external bus is used to fetch the machine code of the next instruction and arrange it in a queue known as predecoded instruction byte queue.
- It is a 6 bytes long FIFO structure.
- Once a byte is decoded, the queue is rearranged by pushing it out and the queue status is checked for the possibility of the next opcode fetch cycle.
- while the opcode is fetched by BIU, the EU executes the previously decoded instructions concurrently.

		(2) – The BIU along with the EU forms a pipe – BIU works under the control of timing control unit. EU: It contains the register set of 8086 except segment registers and IP. – It has 16-bit ALU, able to perform arith and logic operations. – 16-bit flag register reflects the results o execution by the ALU. Decoding unit: It decodes the opcode bytes issued from the instruction byte queue. – The timing and control unit derives the necessary control signals to execute the instr opcode received from the queue. – The EU may pass the results to the BIU for storing them in memory.			
2	a)	**Describe the flag register structure of 8086 with a neat diagram.** • Diagram • Explanation	2 M 6 M	8 M	10 M

Flag Register — i) condition codes or status flags
ii) Machine control flags



O – overflow flag, D → Direction flag, I → Interrupt flag, T → Trap flag
S – Sign flag, Z → zero flag, Ac → Auxiliary carry flag, P → Parity flag
CY → carry flag, X → not used.

S – Sign flag :– If result of any computation is –ve. For signed computations sign flag is MSB of the result.

Z – zero flag :– If the result of computation or comparison performed by the previous instructions is zero.

P – parity flag : flag is set if lower byte of the result contains even number of 1s.

C – carry flag :– set if carry out of MSB for addition or a borrow for subtraction.

T – Trap flag : If it is set processor enters single step execution mode. Trap interrupt is generated after execution of each instruction. The processor executes the current instruction and control is transferred to Trap interrupt service routine.

I – Interrupt flag : – The maskable interrupts are recognised by CPU if it is set.

		D – Direction flag : If this bit is '0' the string is processed beginning from the lowest address, to the highest address, i.e. auto incrementating mode. – otherwise string is processed from the highest address towards the lowest address, auto decrementing mode. Ac – Auxiliary Carry Flag : It is set if there's a carry from the lowest nibble, during addition, or borrow from the lowest nibble, i.e. bit 3, during subtraction. O – Overflow flag : Set if overflow occurs, i.e. the result of a signed operation is large enough to be accomodated in a destination register. eg. addition of two signed numbers if the result overflows into the sign bit, i.e the result is of more than 7-bits in size in case of 8 bit signed operations and more than 15-bits in size in case of 16-bit signed operations, then the overflow flag will be set.			
	b)	**Differentiate between SUB and CMP instructions.** • SUB definition • CMP definition **Each carry 1 Mark** SUB: Subtracts source operand from destination operand and stores the result of subtraction in destination. CMP: Compares source and destination operand by subtracting source operand from destination operand. But does not store the result of subtraction.	2 M	2 M	
3		**Determine the physical address for the following instructions, if DS=2000h, SS=3000h, ES=4000h,BP=0010h, BX=0020h, SP=0030h, SI=0040h, DI=0050h,** **I. MOV AL, [DI]** **II. MOV CX, [BX]** **III. MOV AL,[BP+SI]** **IV. MOV DS:[BX], AL** **V. MOV AL, [BX+ 1200H]**	2 M 2 M 2 M 2 M 2 M	10 M	10 M

		• Physical address claculation for each command\ 3. I. MOV AL, [DI] Physical address = DS * 10H + 0050H = 20000H + 0050H = 20050H II. MOV CX, [BX] Physical address = DS * 10H + 0020H = 20020 H III. MOV AL, [BP+SI] Physical address = 55 * 10H + 0040H + 0010 H = 30000H + 0040H + 0010 H = 30050 H IV. MOV DS: [BX], AL Physical address = DS * 10H + BX = 20000H + 0020H = 20020 H V. MOV AL, [BX+1200H] Physical address = DS * 10H + 0020H + 1200 = 20000H + 1220H = 21220H			
4		**Describe briefly any 5 addressing modes of 8086 with an example for each.** • Any 5 addressing modes with example	2 M 2 M 2 M 2 M 2 M	10 M	10 M

Addressing modes of 8086

1. Immediate: Data is a part of the instruction appears as successive byte or bytes.

   MOV AX, 0005H → 16 bit immediate.
   MOV BL, 06H → 8bit immediate

2. Direct: A 16-bit memory address (offset) I/O address is directly specified in the i... as a part of it.

   MOV AX, [5000H] ; 10H * 0.5 + 5000H → off... [5000H] → offset addres...
   IN 80H ; ; 80H is IO address

		3. Register
		MOV BX, AX
		ADC AL, BL
		→ The operands in these instructions are provided in the registers BX, AX, AL, BL respectively.
		4. Register Indirect
		In this addressing mode, the offset address of data is in BX, SI, DI register.
		The address of the memory location which contains the data or operand is determined using the offset registers.
		MOV AX, [BX]
		Here data is present in a memory location in DS whose offset address is in BX.
		effective address 10H*DS+[BX]
		5. Indexed:
		offset of the operand in one of the index registers.
		DS is the default segment for index registers SI and DI.
		In case of string operations DS and ES are the default segments for SI ad DI respectively.
		MOV AX, [SI]
		MOV CX, [DI]
		Here, data is available at an offset address stored in SI in DS.
		The effective address, is, 10H * DS+[SI]			

5	a)	**Briefly describe the memory segmentation of 8086.** • Diagram • Explanation The 8086 architecture uses the concept of segmented memory. The size of address bus of 8086 is 20 bits and is able to address 1 Mbytes (2^{20}) of physical memory. This 1 megabyte memory is divided into 16 segments and	2 M 4 M	6 M	10 M

each segment is of 64 Kbytes. However, at any given time the 8086 works with only four segments namely code segment, data segment, extra segment and stack segment. The complete physical address of a memory location which is 20-bits long is generated using segment and offset registers, each 16-bits long.

The segment register indicates the base address of a particular segment and CS, DS, SS and ES are used to keep the base address (starting address) of a segment.

The offset indicates the distance of the required memory location in the segment from the base address, and the offset may be the content of register IP, BP, BX, SI, DI and SP.

Generating a physical address:

- The content of segment register (segment address) is shifted left bit-wise four times.
- The content of an offset register (offset address) is added to the result of the previous shift operation.

These two operations together produce a 20-bit physical address. For example, consider the segment address is 2000H and the offset address is 3000H. The physical address is calculated as:

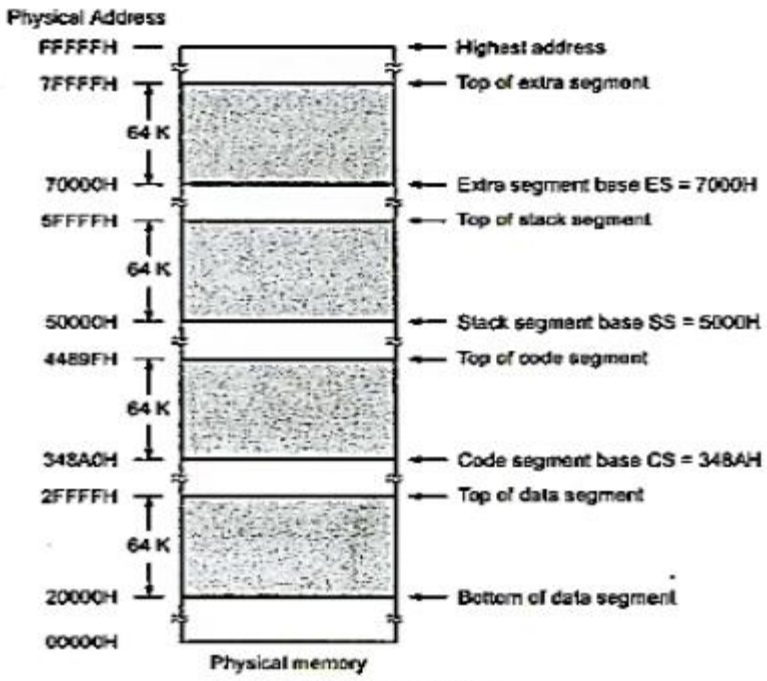
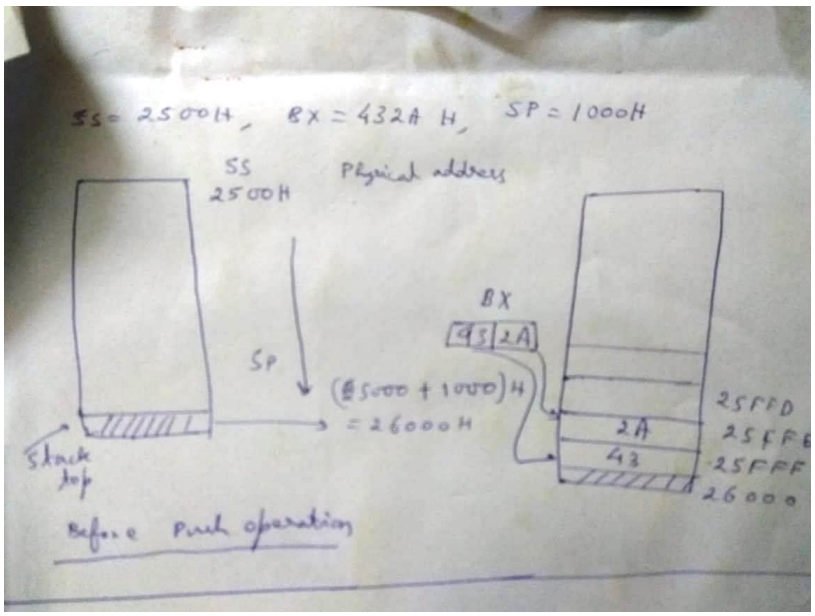
PA=Segment Address*10H + offset address

=2000*10+3000=23000H

The main advantages of the segmented memory scheme are as follows:

- Allows the memory capacity to be 1 Mbyte although the actual addresses to be handled are of 16-bit size
- Allows the placing of code data and stack portions of the same program in different parts (segments) of memory, for data and code protection.

Diagram of memory segmentation is given below.

		 Memry Segmentation			
	b)	**Sketch the content of stack memory indicating the value of SP register before PUSH BX operation and after PUSH BX operation. Assume SS = 2500H, BX = 432AH, SP = 1000H.** • Before execution • After execution 	2 M 2 M	4 M	

6		**Describe the following instructions with example. i) ADC ii) LEA iii) IMUL iv) POP sv) DIV.** • Explanation with example for each I. **ADC:** Syntax: **ADC destination, source** ; destination = destination + source +CF The ADC instruction adds the **source operand** and the **destination operand**along with **CF** (which may be set as a result of previous calculation).The **result** will be stored in the **destination operand.** **Source operand** may be a register, memory location or immediate data. **Destination operand** may be a register or memory location. Both destination and source operands must not be **memory location.** Destination operand must not be an **immediate operand.** **All the status flags will be affected by this instruction.** Examples: ADC AX,0100H ; Immediate addressing mode ADC AX,BX ; Register addressing mode ADC AX,[5000H] ; Direct addressing mode II. **LEA:** (Load Effective Address) Instruction loads the address of the source label into destination register (SI, DI, BX) Ex: LEA SI, SRC Here address of the label SRC is loaded into SI register. III. **IMUL:** Syntax: **IMUL SOURCE** This instruction multiplies a signed byte or signed word in source operand by signed byte in AL or signed word in AX respectively . For signed byte multiplication the result will be stored in AX. For signed word multiplication the most significant word of the result is stored in DX while the least significant word is stored in AX The source Operand can be a general purpose register or memory. Cannot be a constant or immediate data. Only CF and OF are affected SF, ZF, AF, PF are unpredictable **EX:** Let AL = FFh and BL = 02h **After Execution** MUL BL; AX = 01FEh	2 M 2 M 2 M 2 M 2 M	10 M	10 M

		IV. **POP:** **POP destination: Pop from Stack** The POP instruction copies a word from the stack location pointed by the stack pointer to a destination specified in the instruction. The destination can be a general-purpose register, a segment register or a memory location. After the word is copied to the specified destination, the stack pointer is automatically incremented by 2 to point to the next word on the stack. The POP instruction does not affect any flag. EX: POP AX POP DS POP [5000H] V. **DIV:** **Syntax: DIV SOURCE** It divides an unsigned word or double word by a byte or word operand respectively. The source Operand can be a general purpose register or memory. Cannot be a constant or immediate data. CF,OF,SF, ZF, AF, PF are unpredictable EX: Let **DX=0000h, AX=0005h**, and **BX=FFFEh** – DIV BX; AX=0000 DX=0005			
7		**Write an ALP to copy a block of 10 data bytes from location SRC to location** • Template • Algorithm .mode Small .stack 64H .data src db 10H,20H,30H,40H,50H,60H,70H,80H,90H,0A0H count equ ($-src) lap equ 2 dst db count-lap dup(00h) .code	4 M 6 M	10 M	10 M

		Mov ax,@data
		Mov ds,ax
		Mov bx,count
		Up: Mov al,src[bx-1]
		Mov dst[bx-lap-1], al
		Dec bx
		Jnz up
		
		Mov ax,4c01h
		Int 21h
		end			

8		**Verify the following instructions and correct them if any instruction is wrong. Explain the operation performed by all the instructions.** **i. ADD [2345H], AL** **ii. INC BX,2** **iii. SUB 0AH,AL** **iv. MOV AX,12H** **v. MUL AL** • For each instruction validation and reasoning **I. ADD [2345H],AL** **Instruction is correct.** Here the contents of register AL is added with the contents of memory location whose offset address is 2345H of data segment and the result stored in DS:[2345H]. II. **INC BX,2**	2 M 2 M 2 M 2 M 2 M	10 M	10 M

		Instruction is wrong. It should be INC BX twice to increment the contents of BX by 2. **III. SUB 0AH,AL** Instruction is wrong. Destination cannot be immediate data. It should be SUB AL,0AH. AL=AL-0AH **IV. MOV AX,12H** **Instruction is correct.** Here AX will get the immediate value 0012H. AX=0012H V. MUL AL **Instruction is correct.** Here the contents of register AL is multiplied with the contents AL register and the result is stored in AX register.			