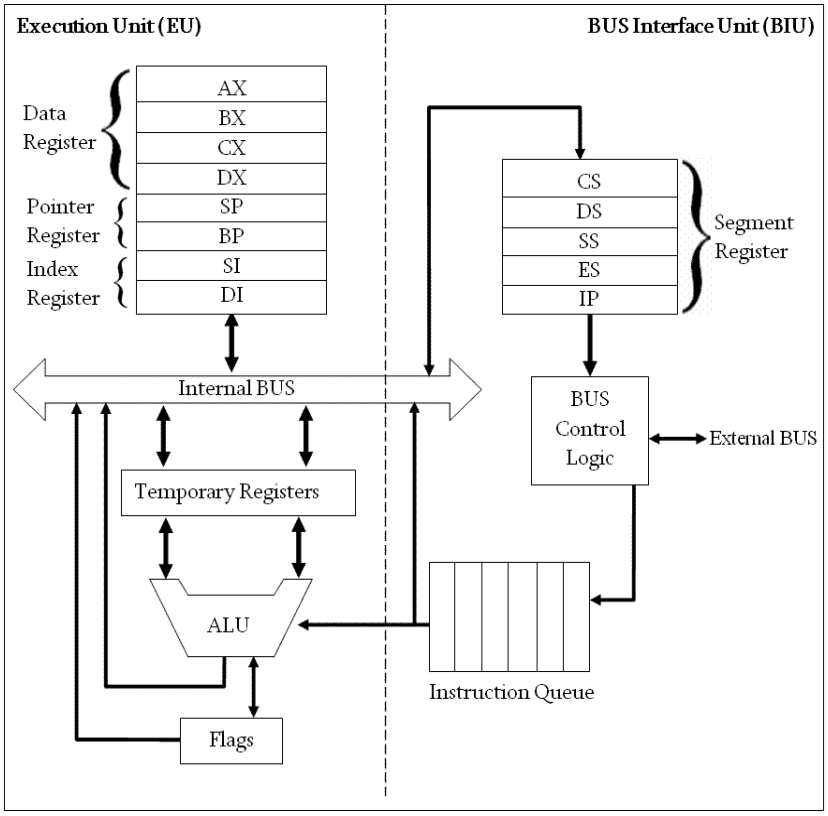
BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT
YELAHANKA – BANGALORE - 64

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

I INTERNAL ASSESSMENT TEST SCHEME & SOLUTIONS, MARCH – 2017

Subject: Microprocessors & Microcontrollers	***Subject Code: 15CS44***	***Branch & Semester : CSE - 4 A & B***
Max. Marks : 30 Marks	***Date: 17/03/2017*** ***Time: 2 PM - 3:30 PM***	***Faculty: Mr. Shankar R***

Answer FIVE full questions, selecting ONE full question from each Part.
(Part D & Part E are compulsory questions.)

Q. No	Question	CO, PO, K level	Marks
	PART-A		
1.	**Describe in detail with a neat figure the working of the internal architecture of the 8086 MP.** *Diagram 2 marks* *Explanation 6 marks*  *The 8086 CPU is divided into two parts. They are:* *1.Bus Interface Unit (BIU)* *2. Execution Unit (EU)* ***Bus Interface Unit (BIU):*** *The BIU handles all data and addresses on the buses for the execution unit such as fetching the instruction or data, writing*	***CO1 (PO1) K1***	06

	the data to memory, writing the data to the port, reading data from the port. ***Execution Unit (EU):*** *The execution unit (EU) tells the BIU where to fetch instructions or data from, decodes instructions, and executes instructions. The functional parts of the execution unit are control circuitry or system, instruction decoder, and Arithmetic logic unit (ALU).* *There are 5 types of Registers. They are: 1. General purpose registers* *2. Pointer Registers* *3. Index registers* *4. Flag registers* *5. Segment registers* ***General Purpose Registers:*** *They are divided into 4 types. They are: 1.AX (accumulator)* *2. BX (base)* *3. CX (counter)* *4. DX (data)* ***Pointer Registers:*** *They are divided into 3 types. They are: 1.SP (Stack pointer)* *2. BP (Base pointer)* *3. IP (Instruction pointer)* ***Index Registers:*** *Index group consists of SI (Source Index), DI (Destination index)* ***Segment Registers:*** *Segment group consists of ES (Extra Segment), CS (Code Segment), DS (Data Segment) and SS (Stack Segment).* ***Flag Registers:*** *Control flag group consists of a single 16-bit flag register.*		
2.	**Describe in detail the Register Organization & the various bits of a Flag Register for 8086 MP.** *Register Organization of 8086 Microprocessor – 3 marks* *Flags – 3 marks* *The 8086 has a powerful set of registers. It includes general purpose registers, segment registers, pointers and index registers and flag register. All the registers of 8086 are 16-bit registers.* ***General Data Registers:*** *The registers AX, BX, CX and DX are the general purpose 16-bit registers.AX is used as 16-bit accumulator, with the lower 8-bits of AX designated as AL and higher 8-bits as AH. .BX is used as 16-bit base, with the lower 8-bits of BX designated as BL and higher 8-bits as BH.CX is used as 16-bit counter, with the lower 8-bits of CX designated as CL and higher 8-bits as CH.DX is used as 16-bit data, with the lower 8-bits of DX designated as DL and higher 8-bits as DH.*	***CO1 (PO1) K1***	06

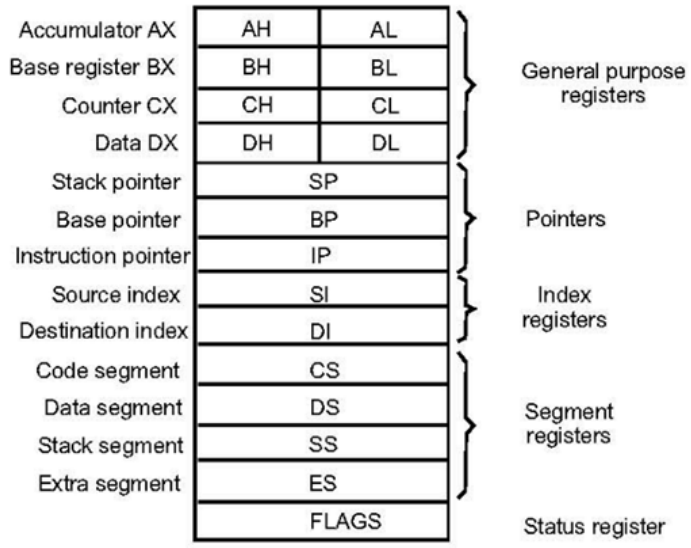


Fig.11.6: Schematic diagram of intel 8086 registers

Flag Register: *8086 has a 16-bit flag register which is divided into two parts.They are: 1.Status flags*

2. Control flags

The description of each flag bit is as follows:

S-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.*

Z-Zero Flag: *This flag is set, if the result of the computation or comparison performed by the previous instruction/instructions is zero.*

P-Parity Flag: *This flag is set to 1, if the lower byte of the result contains even number of 1s.*

C-Carry Flag: *This flag is set, when there is a carry out of MSB in case of addition or borrow in case of subtraction.*

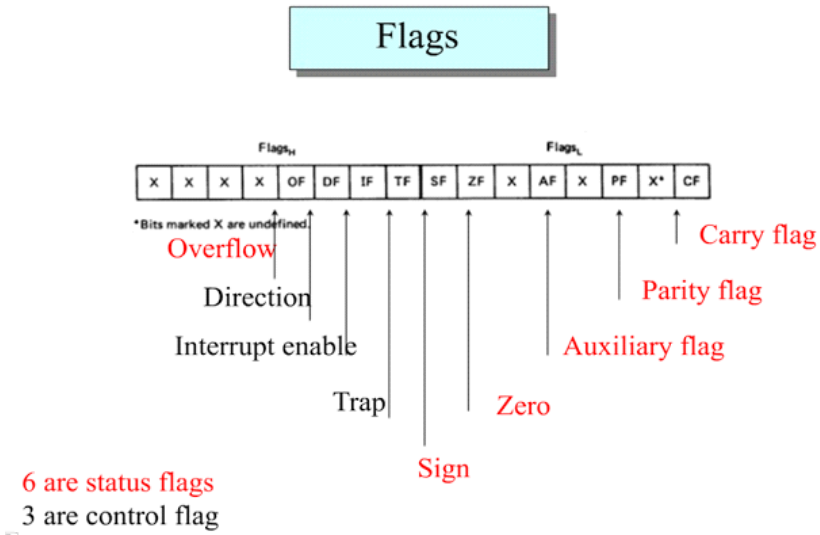
T-Trap Flag: *If this flag is set, the processor enters the single step execution mode. In other words, a trap interrupt is generated after execution of each instruction. The processor executes the current instruction and the control is transferred to the Trap interrupt service routine.*

I-interrupt Flag: *If this flag is set, the maskable interrupts are recognized by the CPU, otherwise, they are ignored.*

D-Direction Flag: *This 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, i.e. auto incrementing mode. Otherwise, the string is processed from the highest address towards the lowest address, i.e. auto decrementing mode.*

AC-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.*

O-Overflow Flag: *This flag is set, if an overflow occurs, i.e. if the result of a signed operation is large enough to be accommodated in a destination register. For example, in case of the 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 than15-bits in size in case of 16-bit signed operations, and then the overflow flag will be set.*



PART-B

3a.	**Describe real mode addressing. Recite default segment and offset registers.**	***CO1 (PO1) K1***	03

Real mode – 1 marks
default segment and offset registers – 2 marks

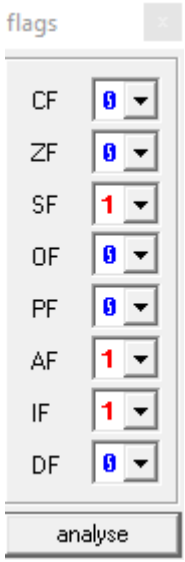
Real mode, also called real address mode, is an operating mode of all x86-compatible CPUs. Real mode is characterized by a 20-bit segmented memory address space.
Real mode operation allows the microprocessor to address only the first 1M byte of memory space-even if it is the Pentium II microprocessor. Note that the first 1 M byte of memory is called either the real memory or conventional memory system. The DOS operating system requires the microprocessor to operate in the real mode. Real mode operation allows application software written for the 8086/8088, which contain only 1 M byte of memory.
Table 2-1 for addressing memory using any Intel microprocessor with 16-bit registers. Table 2-2 shows the defaults assumed in the 80386 and above when using 32-bit registers. Note that the 80386 and above have a far greater selection of segment offset address combinations than do the 8086 through the 80286 microprocessors.

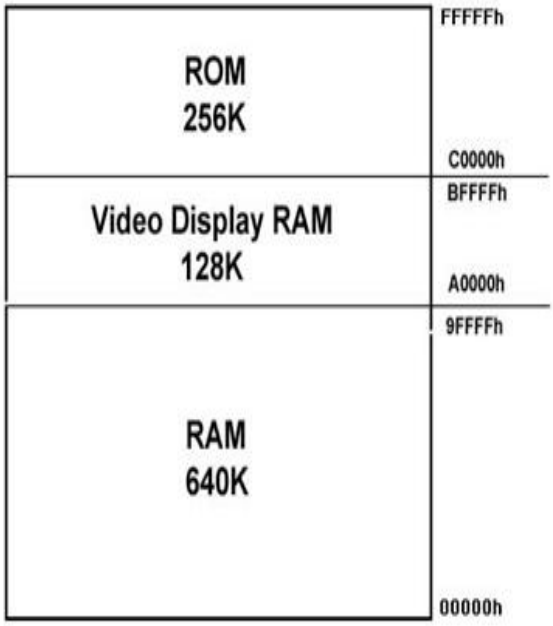
Segment	Offset	Special Purpose
CS	IP	Instruction address
SS	SP or BP	Stack address
DS	BX, DI, SI, an 8-bit number, or a 16-bit number	Data address
ES	DI for string instructions	String destination address

Table 2-1: Default 16-bit segment and offset address combinations

Segment	Offset	Special Purpose
CS	EIP	Instruction address
SS	ESP or EBP	Stack address
DS	EBX, EDI, ESI, EAX, ECX, EDX an 8-bit number, or a 32-bit number	Data address
ES	EDI for string instructions	String destination address
FS	No default	General address
GS	No default	General address

Table 2-2: Default 32-bit segment and offset address combinations

3b.	**Restate the Flag register after executing the following code:** *Computation – 1 mark* *Flags – 2 marks* *MOV AX,34F5H* *ADD AX,95EBH* *Answer is : CAE0H* *Flags affected:* 	***CO1 (PO1,PO2) K2***	03
4a.	**Identify the addressing modes of the following instructions and explain them briefly:** *addressing mode each 1 mark* *MOV WORD PTR [SI], 20H* *MOV ES: [1000H], 10H* *MOV CX, NUM [BX + DI]* *1. MOV WORD PTR [SI], 20H* *This is immediate addressing mode.* *Immediate addressing mode: In this mode of addressing the data to be manipulated is part of the instruction.Immediate data should be in source field. And the destination can be register or memory location.* *2. MOV ES: [1000H], 10H* *This is immediate addressing mode.* *Immediate addressing mode: In this mode of addressing the data to be manipulated is part of the instruction.Immediate data should be in source field. And the destination can be register or memory location.* *3. MOV CX, NUM [BX + DI]* *This is base index addressing mode* *Base index addressing mode: Operand offset is given by sum of either BX or BP with either SI or DI.*	***CO1 (PO1,PO2) K1***	03
4b.	**Recall the Memory Map of IBM PC.** *Diagram – 1 mark* *Explanation – 2 marks* *1. For a program to be executed on the PC, windows must first load it into*	***CO1 (PO1) K1***	03

	RAM.The 20-bit address of the 8086/88 allows a total of 1 megabyte (1024k bytes) of memory space with address range 00000-FFFFF. *2. During the design of first IBM PC, engineers has to decide on the allocation of 1-megabyte memory space to various sections of the PC. This memory allocation is called MEMORY MAP.* *3. Of this 1-megabyte, 640k bytes from addresses 00000-9FFFFH were set aside for RAM.* *4. The 128K bytes from A0000H to BFFFFH were allocated for video memory. The remaining 256K bytes from C0000H to FFFFFH were aside for ROM.*  Memory Map		
	PART-C		
5.	**Demonstrate an assembly language program to reverse a given string and verify whether it is a palindrome or not. Display the appropriate message.** *Program – 6 marks* *.model small* *Initds macro* *Mov ax, @data; initializing the data segment* *Mov ds, ax ; it is ds, not dx* *Endm* *Inites macro* *Mov es, ax ; initializing the extra segment* *Endm* *Printf macro msg*	***CO1,CO4 (PO1,PO3) K3***	06

```
 Lea dx, msg       ; load the effective address to dx
 Mov ah, 9         ; function number is 9
 Int 21h           ; using dos interrupt 21h
Endm

Getchar macro
 Mov ah, 1         ; this macro takes 1 key input,
 Int 21h           ; its ASCII value in hex stores in al
Endm

Exit macro
 Mov ah, 4ch    ; to terminate
 Int 21h
Endm

.data
 Original db 30 dup (?)       ; 1st array
 Reverse db 30 dup (?)       ; 2nd array to store the reversed array

 Ask db 10, 13,"String please: $"
 Palindromemsg db 10, 13,"Palindrome$"
 Notpalindromemsg db 10, 13,"Not Palindrome
.code

 Initds

 Inites   ; initializing extra segment (b'coz we are playing with strings)

 Lea si, original   ; 1st array starting index to si
 Lea di, reverse   ; 2nd array starting index to di
 Printf ask
 Mov cx, 00     ; counter. Right now it's 0 (we haven't taken any i/p)

 Takeinput:

 Getchar                ; takes single character
 Cmp al, 13             ; compare with ENTER key
 Je done                ; if you press ENTER key, then goto done
 Mov [si], al           ; else, store your key in array
 Inc cx                 ; keeps the no. of elements in array
 Inc si                 ; move to next position
 Jmp takeinput          ; repeat till you press ENTER key

 Done: dec si           ; point to the last position

 Reversingtask:

 Mov al, [si]           ; last element of si
 Mov [di], al           ; put that to first element of di
 Inc di                 ; Inc 2nd array position
 Dec si                 ; dec 1st array position
 Jnz reversingtask

 Lea si, original       ; comparison part
 Lea di, reverse
 Cld                    ; clear direction flag
```

	; (so that si & di are auto incremented) *Repe cmpsb ; comparing [si] & [di]* *Je palin ; if all the characters are equal, then goto palin* *Printf notpalindromemsg ; else, not palindrome case* *Exit ; bye bye!* *Palin: printf palindromemsg; palindrome* *Exit ; bye bye!* *End* *OUTPUT 1:* *3. EXE* *String please: MADAM* *Palindrome* *OUTPUT 2:* *3. EXE* *String please: COLLEGE* *Not Palindrome*		
6.	**Demonstrate an assembly language program to read the current time and Date from the system and display it in the standard format on the screen.** *Program – 6 marks* *.model small* *Initds macro* *Mov ax,@data ; initializing the data segment* *Mov ds, ax ; it is ds, not dx* *Endm* *Printf macro msg* *Lea dx, msg ; load the effective address to dx* *Mov ah, 9 ; function number is 9* *Int 21h ; using dos interrupt 21h* *Endm* *Putchar macro char* *Mov dl, char ; load the printable character's hex value in dl* *Mov ah, 2 ; function number is 9* *Int 21h ; using dos interrupt 21h* *Endm* *Accesstime macro* *Mov ah, 2ch ; time interrupt ch=hours; cl=minutes* *Int 21h ; dh=seconds; dl=milliseconds* *Endm* *Accessdate macro ; date interrupt dl=day; dh=month; cx=year* *Mov ah, 2ah*	***CO1,CO4 (PO1,PO3) K3***	06

```
   Int 21h
 Endm

 Display macro value
    Mov al, value; copy the passed value to AL
    Aam              ; split al into ah & al
   Add ax, 3030h; convert ah & al to ASCII
   Mov bx, ax       ; copy ax to bx to be safe
   Putchar bh       ; print first digit
   Putchar bl       ; print second digit
 Endm

 Exit macro
   Mov ah, 4ch      ; to terminate
   Int 21h
 Endm

Time macro
   Printf timemsg     ; print "current time is"
   Accesstime         ; call accesstime macro
   Display ch         ; display hours
   Putchar ':'        ; print ':
   Display cl         ; display minutes
 Endm

Date macro
   Printf datemsg     ; print "current date is"
   Accessdate         ; call accessdate macro
   Display dl         ; display day
   Putchar ':'        ; print ':'
    Display dh        ; display month
Endm

.data
 Timemsg db 10, 13,"current time is $"
 Datemsg db 10, 13,"current date is $"

.code
  Initds    ; initialze data segment
  Time      ; time task
  Date      ; date task
  Exit      ; bye bye!
End
```

OUTPUT:
5. EXE
Current time is 10:37
Current date is 14:03

PART-D

7.	**Recognize the Processor we use in Microprocessor Lab at BMSIT&M. Also, recall its brief history.** *Technically, we use Intel® Core™2 Quad Processor Q9500 (6M Cache, 2.83*	***CO1 (PO1,PO2) K1***	08

	GHz, 1333 MHz FSB). Its complete specifications are found in this URL. Students need to go through it and understand the various details that has gone in manufacturing this processor. *https://ark.intel.com/products/37159/Intel-Core2-Quad-Processor-Q9500-6M-Cache-2_83-GHz-1333-MHz-FSB* *Its history tracks down to 1971 where Intel came up with 4004 Processor which is a 4 bit Processor. Then in 1972, they released 8008 which is a 8 bit processor. They tasted tremendous success in 1978 by releasing 8086 microprocessor which had 16-bit data bus, internally and externally. All registers are 16 bits wide, and there is a 16-bit data bus to transfer data in and out of the CPU. There was resistance to a 16-bit external bus as peripherals were designed around 8-bit processors. A printed circuit board with a 16-bit data also bus cost more. As a result, Intel came out with the 8088 version.Identical to the 8086, but with an 8-bit data bus. Picked up by IBM as the microprocessor in designing the PC. Intel introduced the 80286 in 1982, which IBM picked up for the design of the PC AT. In 1985 Intel introduced 80386 (or 80386DX). 32-bit internally/externally, with a 32-bit address bus. Capable of handling memory of up to 4 gigabytes.Virtual memory increased to 64 terabytes.Later Intel introduced 386SX, internally identical, but with a 16-bit external data bus & 24-bit address bus. This makes the 386SX system much cheaper. On the 80486, in 1989, Intel put a greatly enhanced 80386 & math coprocessor on a single chip. In 1992, Intel released the Pentium®. (not 80586).* *In 1995 Intel Pentium® Pro was released—the sixth generation x86. It had 5.5 million transistors. Designed primarily for 32-bit servers & workstations. In 1997 Intel introduced the Pentium® II processor. In 1998 the Pentium® II Xeon was released. In 1999, Celeron® was released. In 1999 Intel released Pentium® III. In 1999 Intel introduced the Pentium® III Xeon. Designed more for servers and business workstations with multiprocessor configurations.The Pentium® 4 debuted late in 1999. Intel has selected Itanium® as the new brand name for the first product in its 64-bit family of processors. Formerly called Merced. The evolution of microprocessors is increasingly influenced by the evolution of the Internet.*		
	PART - E		
8.	**We came across how difficult it is to print year while accessing date using 2AH function call. Construct a solution which addresses the above problem.** *DATA SEGMENT* *NUM DW 1234H* *RES DB 10 DUP ('$')* *DATA ENDS* *CODE SEGMENT* *ASSUME DS:DATA,CS:CODE* *START:* *MOV AX,DATA* *MOV DS,AX* *MOV AX,NUM* *LEA SI,RES*	***CO1,CO5 (PO2,PO3) K3***	06

```
   CALL HEX2DEC

   LEA DX,RES
   MOV AH,9
   INT 21H

   MOV AH,4CH
   INT 21H
CODE ENDS
HEX2DEC PROC NEAR
   MOV CX,0
   MOV BX,10

LOOP1: MOV DX,0
      DIV BX
      ADD DL,30H
      PUSH DX
      INC CX
      CMP AX,9
      JG LOOP1

      ADD AL,30H
      MOV [SI],AL

LOOP2: POP AX
      INC SI
      MOV [SI],AL
      LOOP LOOP2
      RET
HEX2DEC ENDP

END START
```

Course Outcomes: Students will be able to	
CO1	*Describe the architecture of X86 Microprocessors and have an introduction to Assembly Language Programming.*
CO2	*Discuss the Instruction Set of X86 Microprocessors and extend it to interface various devices to X86 families*
CO3	*Understand ARM philosophy and its Instruction Set.*
CO4	*Demonstrate the skills to code in Assembly Language, ARM.*
CO5	*Construct software and hardware programs using Assembly Language Programming, ARM.*

K1: Remember	***K2:Understand***
K3: Apply	***K4: Analyze***
K5: Evaluate	***K6: Creation***