UNIT I

The 8086 MICROPROCESSOR

Part A

1. What are the functional parts of 8086 CPU?

The two independent functional parts of the 8086 CPU are:

i. Bus Interface Unit (BIU):

BIU sends out addresses, fetches instruction from memory, reads data from ports and memory and writes data to ports and memory.

ii. Execution Unit (EU):

EU tells the BIU where to fetch instructions or data, decodes instructions and executes instructions.

2. What is the purpose of a decoder in EU?

The decoder in EU translates instructions fetched from memory into a series of actions, which the EU carries out.

3. Give the register classification of 8086.

The 8086 contains:

i. General purpose registers:

They are used for holding data, variables and intermediate results temporarily.

ii. Special purpose registers:

They are used as segment registers, pointers, index register or as offset storage registers for particular addressing modes.

4. What are general data registers?

The registers AX,BX,CX and DX are the general data registers.

AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

L and H represents the lower and higher bytes of particular register.
AX register is used as 16-bit accumulator.
BX register is used as offset storage for forming physical addresses in case of certain addressing modes.
CX register is used as a default counter in case of string and loop instructions.

DX register is used as an implicit operand or destination in case of a few instructions.

5. Give the different segment registers.

The four segment registers are:

i. Code segment register:

It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.

ii. Data segment register:

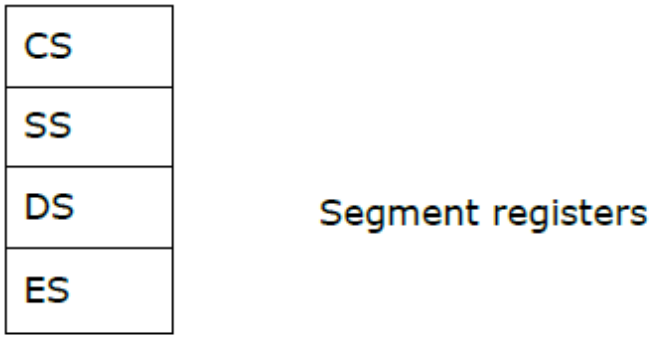
It points to the data segment of the memory, where data is resided.

iii. Extra segment register:

It also contains data.

iv. Stack segment register:

It is used for addressing stock segment of memory. It is used to store stack data.



6. What are pointers and index registers?

IP, BP and SP are the pointers and contain offsets within the code, data and stack segments respectively. SI and DI are the index registers, which are used as general purpose registers and also for offset storage in case of indexed, based indexed and relative based indexed addressing modes.

7. If the execution unit generates an effective address of 43A2H and the DS register contains 4000H. What will be the physical address generated by the BIU? What is the maximum size of the data segment.

Effective Address = 43A2

Physical Address = 40000H + 43A2H= 443A2H

8. What are the different types of addressing modes of 8086 instruction set ?

The different addressing modes are:

i. Immediate
ii. Direct
iii. Register
iv. Register indirect
v. Indexed
vi. Register relative
vii. Based indexed
viii. Relative based indexed

9. What are the different types of instructions in 8086 microprocessor?

The different types of instructions in 8086 microprocessor are:

i. Data copy / transfer instructions
ii. Arithmetic and logical instructions
iii. Branch instructions
iv. Loop instruction
v. Machine control instruction
vi. Flag manipulation instruction
vii. Shift and rotate instruction
viii. String instruction.

10. Mention the addressing modes of the following 8086 instructions
Mov al, disp(ax)- Register Relative Addressing Mode
Mov AH, DISP [bx][si] – Relative Based Indexed Addressing Mode

11. What are the different flag available in status register of 8086 ?
There are 6 one bit flags are present. They are,
AF - Auxiliary Carry Flag
CF - Carry Flag
OF - Overflow Flag
SF - Sign Flag
PF - Parity Flag
ZF - Zero Flag
Flag register format and functions refer book (architecture of 8086)

12. How Will Carry And Zero Flags Reflect the Result of The Instruction **Cmp Bx,Cx?**

	CF	ZF
BX=CX	0	1
BX>CX	0	0
BX<CX	1	0

13. What do these instructions do?
STD, IRET
STD- Set the direction flag register(D=1)
IRET- Interrupt on return. It is used to exist any interrupt procedure, whenever activated by hardware and software

14. What is the use of HOLD and HLDA signals?

HOLD- The signal indicates the another master is requesting the host 8086 to handover the system bus
HLDA- On receiving the hold signal, 8086 outputs HLDA signal high as an acknowledgement

15. What is an assembler directive?
An assembler directive is a statement to give direction to the assembler to perform task of assembly process

16. How single stepping can be done in 8086?
By setting the Trace Flag (TF) the 8086 goes to single-step mode. In this mode, after the execution of each instruction s 8086 generates an internal interrupt and by writing some interrupt service routine we can display the content of desired registers and memory locations. So it is useful for debugging the program.

17. What are the functions of bus interface unit (BIU) in 8086?
(a) Fetch instructions from memory.
(b) Fetch data from memory and I/O ports.
(c) Write data to memory and I/O ports.
(d) To communicate with outside world.
(e) Provide external bus operations and bus control signals.

18. Explain the process of control instructions
STC – It sets the carry flag & does not affect any other flag
CLC – it resets the carry flag to zero &does not affect any other flag
CMC – It complements the carry flag & does not affect any other flag
STD – It sets the direction flag to 1 so that SI and/or DI can be decremented automatically after execution of string instruction & does not affect other flags
CLD – It resets the direction flag to 0 so that SI and/or DI can be incremented automatically after execution of string instruction & does not affect other flags
STI – Sets the interrupt flag to 1. Enables INTR of 8086.
CLI – Resets the interrupt flagto0. 8086 will not respond to INTR.

19. Discuss the function of instruction queue in 8086?
In 8086, a 6-byte instruction queue is presented at the Bus Interface Unit (BIU). It is used to prefetch and store at the maximum of 6 bytes of instruction code from the memory. Due to this, overlapping instruction fetch with instruction execution increases the processing speed.

20. What is the maximum memory size that can be addressed by 8086?
In 8086, an memory location is addressed by 20 bit address and the address bus is 20 bit address and the address bus is 20 bits. So it can address up to one mega byte (2^20) of memory space.

21. Explain DUP

The DUP directive can be used to initialize several locations & to assign values to these locations. Format Name Data_Type Num DUP (value)
Example TABLE DW 10 DUP (0). Reserves an array of 10 words of memory and initializes all 10 words with 0. array name is TABLE.

22. Explain PUBLIC

For large programs several small modules are linked together. In order that the modules link together correctly any variable name or label referred to in other modules must be declared public in the module where it is defined. The PUBLIC directive is used to tell the assembler that a specified name or label will be accessed from other modules

23. What is Microcontroller and Microcomputer

Microcontroller is a device that includes microprocessor; memory and I/O signal lines on a single chip, fabricated using VLSI technology. Microcomputer is a computer that is designed using microprocessor as its CPU. It includes microprocessor, memory and I/O.

24. What is assembler

The assembler translates the assembly language program text which is given as input to the assembler to their binary equivalents known as object code. The time required to translate the assembly code to object code is called access time. The assembler checks for syntax errors & displays them before giving the object code.

25. What is loader

The loader copies the program into the computer's main memory at load time and begins the program execution at execution time.

26. What is linker

A linker is a program used to join together several object files into one large object file. For large programs it is more efficient to divide the large program modules into smaller modules. Each module is individually written, tested & debugged. When all the modules work they are linked together to form a large functioning program.

27. Explain PROC & ENDP

PROC directive defines the procedures in the program. The procedure name must be unique. After PROC the term NEAR or FAR are used to specify the type of procedure. Example FACT PROC FAR. ENDP is used along with PROC and defines the end of the
procedure.

28. Explain SEGMENT & ENDS

An assembly program in .EXE format consists of one or more segments. The starts of these segments are defined by SEGMENT and the end of the segment is indicated by ENDS directive. Format Name SEGMENT Name ENDS.

29. What are procedures

Procedures are a group of instructions stored as a separate program in memory and it is called from the main program whenever required. The type of procedure depends on where the procedures are stored in memory. If it is in the same code segment as that of the main program then it is a near procedure otherwise it is a far procedure.

30. What are libraries

Library files are collection of procedures that can be used in other programs. These procedures are assembled and compiled into a library file by the LIB program. The library file is invoked when a program is linked with linker program. when a library file is linked only the required procedures are copied into the program. Use of library files increase s/w reusability & reduce s/w development time.

31. What are Macros

Macro is a group of instruction. The macro assembler generates the code in the program each time where the macro is called. Macros are defined by MACRO & ENDM directives. Creating macro is similar to creating new opcodes that can be used in the program

```
INIT MACRO
MOV AX, data
MOV DS
MOV ES, AX
ENDM.
```

32. How do 8086 interrupts occur

An 8086 interrupt can come from any of the following three sources
External signals
Special instructions in the program
Condition produced by instruction

33. What are the 8086 interrupt types

Dedicated interrupts
Type 0: Divide by zero interrupt
Type 1: Single step interrupt
Type 2:Non maskable interrupt
Type 3: Breakpoint
Type 4: Overflow interrupt
Software interrupts
Type 0-255

34. What is interrupt service routine

Interrupt means to break the sequence of operation. While the CPU is executing a program an interrupt breaks the normal sequence of execution of instructions & diverts its execution to some other program. This program to which the control is transferred is called the interrupt service routine.

35. What is multiple interrupt processing capability?

Whenever a number of devices interrupt a CPU at a time, and id the processor is able to handle them properly, it is said to have multiple interrupt processing capability.

36. What is hardware interrupt?

An 8086 interrupt can come from any one of three sources. One sources is an external signal applied to the nonmarkable interrupt(NMI) input in or to the interrupt (INTR) input pin. An interrupt caused by the signal applied to one of these input is referred to as a hardware interrupt.

37. What is software interrupt?

The interrupt caused due to execution of interrupt instruction is called software interrupt.

38. What is assembly level programming?

A program called „assembler" is used to convert the mnemonics of instruction and data into their equivalent object code modules. The object code modules are further converted into executable code using linker and loader programs. This type of programming is called assembly level programming.

39. What is a stack ?

Stack is a top-down data structure, whose elements are accessed using a pointer that is implemented using the SS and SP registers. It is a LIFO data segment.

40. What is an assembler directive?

An Assembler directive is a statement to give direction to the assembler to perform task of assembly process

41. How is the physical address calculated ? Give an example.

The physical address, which is 20-bits long is calculated using the segment and offset registers, each 16-bits long. The segment address is shifted left bit-wise four times and offset address is added to this to produce a 20 bit physical address.

Eg: segment address - > 1005H

Offset address - > 5555H

Segment address - > 1005H - > 0001 0000 0000 0101

Shifted by 4 bit position - > 0001 0000 0000 0101 0000

Offset address - > + 0101 0101 0101 0101

Physical address - > 0001 0101 0101 1010 0101

1 5 5 A 5

42. Explain the three machine control flags.

i. Trap flag:

If this flag is set, the processor enters the single step execution.

ii. Interrupt flag:

If this flag is set, the markable interrupts are recognized by the CPU, otherwise they are ignored.

iii. Direction flag:

This is used by string manipulation instructions. If this flag bit is „0", the string is processed from the lowest to the highest address i.e., auto incrementing mode. Otherwise, the string is processed from highest address to lowest address, i.e., auto decrementing mode.

43. Differentiate Macros and Procedures

Procedures	Macros
To use procedure use call and ret instructions are needed	To use macro type its name
It occupies less memory	It occupies More Memory
Stack is used	Stack is not used
End of the procedure: ENDP	End of the procedure: ENDM
Overhead time is needed to call and return	No overhead time during execution

44. Define Linking and Relocation. When it will happen.

The linking program links the different object modules of a source program and function library routines to generate an integrated executable code of the source program. The assignment of segment address is called relocation and is done after the linking process has determined exactly where each segment is to be put in memory.

45. Define Modular Programming

It is defined as subdivision of computer program into separate programs.

46. Give the features of 8086 (Refer Book)

Part B

1. Discuss in detail the architecture of 8086 microprocessor

Key points

Basic components:

Bus Interface unit, Execution Unit.

Architecture Block Diagram

Bus Interface Unit

Four numbers of 16 bit segment registers

a. Code Segment
b. Data Segment
c. Stack Segment
d. Extra Segment

Execution Unit

Eight numbers of 16- bit general purpose registers.
AX, BX, CX, DX, SP, BP, SI and DI.

2. How many functional units does 8086 contain? Discuss them in brief

Key points

Bus Interface Unit and Execution Unit.
Four numbers of 16 bit segment registers

a. Code Segment
b. Data Segment
c. Stack Segment
d. Extra Segment

Execution Unit

Eight numbers of 16- bit general purpose registers.
AX, BX, CX, DX, SP, BP, SI and DI.

3. Illustrate various addressing modes with examples

Key points

Addressing Modes:

Addressing modes indicates the way in which the operand or data for an instruction is accessed and way in which microprocessor calculates the branch address for the jump, call and return instructions.

1. Register addressing Mode
2. Immediate Addressing Mode
3. Data Memory Addressing Mode
4. Program Memory Addressing Mode
5. Stack Memory Addressing Mode.

4. List the types of instruction set related with 8086 microprocessor and explain each of its operation indetail.

1. Data Transfer Instructions

2. Arithmetic Instructions
3. Logical Instructions
4. Flag Manipulation Instruction
5. Control Transfer Instructions

5. Define Assembler and Give an example for assembly language programming
Key points

Assembler
An assembler is a program that is used to convert assembly language program to machine level language program.
Example for assembly language programming

```
PCL EQU 2 ; programme counter lower byte
STATUS EQU 3 ;status register
PORTA EQU 5 ;Port A
PORTB EQU 6 ;Port B
```

6. Explain various modes of assembler directives

Segment control
- Generic segment (SEGMENT, RSEG)
- Absolute segment (CSEG, DSEG and XSEG)

Address control
- ORG, USING, END

Symbol definition
- EQU, SET, CODE, DATA, IDATA, XDATA

Memory initialization/reservation
- DB, DW, DD, DS

Generic segments are created using the SEGMENT directive
The final memory location for a generic segment is assigned by the linker
The format is as follows:

<*Symbol* > SEGMENT <*segment_memory_class*>

Example:

MYDATA SEGMENT DATA

The above directive defines a reloadable segment named as MYDATA, with a memory class of DATA

- Once the above segment name has been defined, the next step is to select that segment by using the RSEG directive as shown in the example below

RSEG MYDATA

Whenever the above statement is encountered, the MYDATA segment will become the current active segment until the assembler comes across another RSEG directive, which will then define another segment area
Absolute segment means a fixed memory segment. Absolute segments are created by CSEG, DSEG and XSEG directives.
The final location of the segment is known at compile time

The format of this directive is as follows:

CSEG AT <*address*>*;* defines an absolute code segment

DSEG AT <*address*>; defines an absolute data segment

XSEG AT <*address*>; defines an absolute external data segment

The specified format for the ORG directive is:

ORG ***<expression>***

The ORG directive is used to set the location counter in the current segment to an offset address specified by the *expression*

The specified format for the directive is:

END

The END directive indicates the end of the source file

7. Describe procedure and macros in detail with necessary examples.

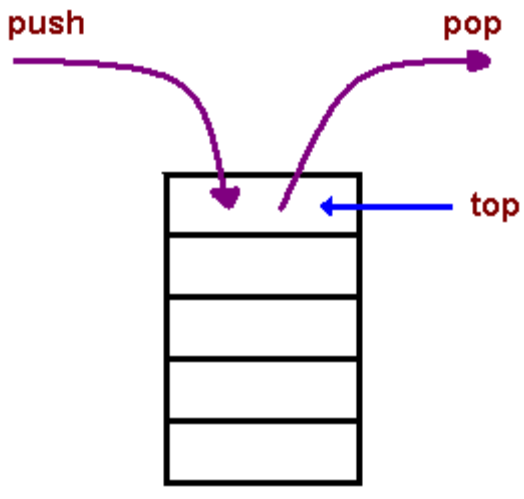
Difference between macros and procedures

Macros	Procedures
1. Accessed during assembly when name given to macro is written as an instruction in the assembly program.	Accessed by CALL and RET instructions during program execution.
2. Machine code is generated for instructions each time a macro is called.	Machine code for instructions is put only once in the memory.
3. This due to repeated generation of machine code requires more memory.	This as all machine code is defined only once so less memory is required.
4. Parameters are passed as a part of the statement in which macro is called.	Parameters can be passed in register memory location or stack.

8. Define stack and stack top address calculation and give example for its programming.

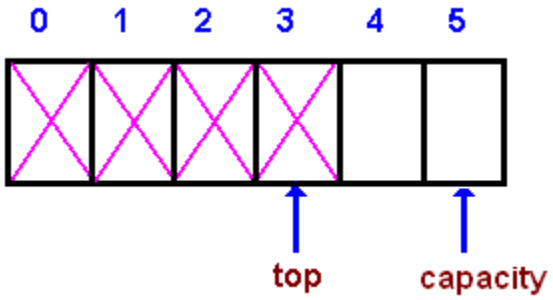
Stacks

A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. In the pushdown stacks only two operations are allowed: **push** the item into the stack, and **pop** the item out of the stack. A stack is a limited access data structure - elements can be added and removed from the stack only at the top. **Push** adds an item to the top of the stack, **pop** removes the item from the top. A helpful analogy is to think of a stack of books; you can remove only the top book, also you can add a new book on the top.



Eg :

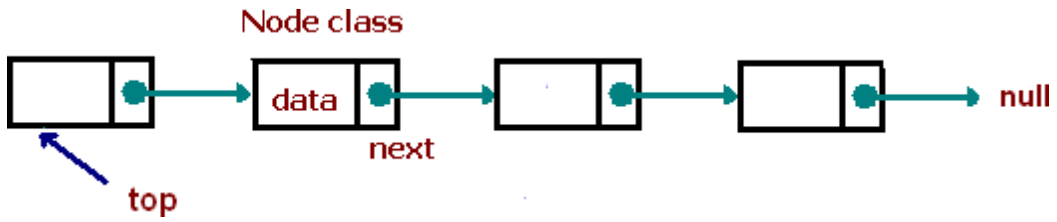
Array-based implementation



In an array-based implementation we maintain the following fields: an array A of a default size (≥ 1), the variable *top* that refers to the top element in the stack and the *capacity* that refers to the array size. The variable *top* changes from -1 to capacity - 1. We say that a stack is empty when top = -1, and the stack is full when top = capacity-1.
In a fixed-size stack abstraction, the capacity stays unchanged; therefore when *top* reaches *capacity*, the stack object throws an exception.

Linked List-based implementation

Linked List-based implementation provides the best (from the efficiency point of view) dynamic stack implementation.



9. Discuss in detail byte and string instruction manipulation.

STRING INSTRUCTIONS

The string instructions are summarized in Fig.5-1.Because the string instructions can operate on only a single byte or word unless they are used with the REP prefiks discussed in Sec. 5-2 they are often referred to as *string primitives*.

Figure 5-1 String primitives

Name	Mnemonic and Format*	Description**
Move string	MOVS DST, SRC	((DI)) <- ((SI)) Byte operands (SI) <- (SI)±1, (DI) <- (DI)±1 Word operands (SI) <- (SI)±2, (DI) <- (DI)±2
Move byte string	MOVSB	
Move word string	MOVSW	
Compare string	CMPS SRC, DST	((SI)) <- ((DI)) Byte operands (SI) <- (SI)±1, (DI) <- (DI)±1 Word operands (SI) <- (SI)±2, (DI) <- (DI)±2
Compare byte string	CMPSB	
Compare word string	CMPSW	
Scan string	SCAS DST	Byte operand ((AL)) - ((DI)), (DI) <- (DI)±1 Word operand ((AX)) - ((DI)), (DI) <- (DI)±2
Scan byte string	SCASB	
Scan word string	SCASW	

Load string	LODS SRC	Byte operand	
		(AL) <- (SI), (SI) <- (SI)±1	
		Word operand	

(AX) <- (SI), (SI) <- (SI)±2

Load byte string LODSB

Load word string LODSW

Store string STOS DST Byte operand

((DI)) <- ((AL)), (DI) <- (DI)±1

Word operand

((DI)) <- ((AX)), (DI) <- (DI)±2

Store byte string STOSB

Store word string STOSW

*The B suffix indicates byte operands and the W suffix indicates.

**Incrementing (+) is used if DF=0 and decrementing (-) is used if DF=1.

Flags: CMPS and SCAS affect all condition flags and

MOVS ,LODS and STOS affect no flags.

Addresing modes: Operands are implied.

All of the primitives are 1 byte long, with bit 0 indicating whether a byte (bit 0=0) or a word (bit 0=1) is being manipulated.