AKGEC/IAP/FM/02

Ajay Kumar Garg Engineering College, Ghaziabad

Department of ECE

ST-2 Model Solution

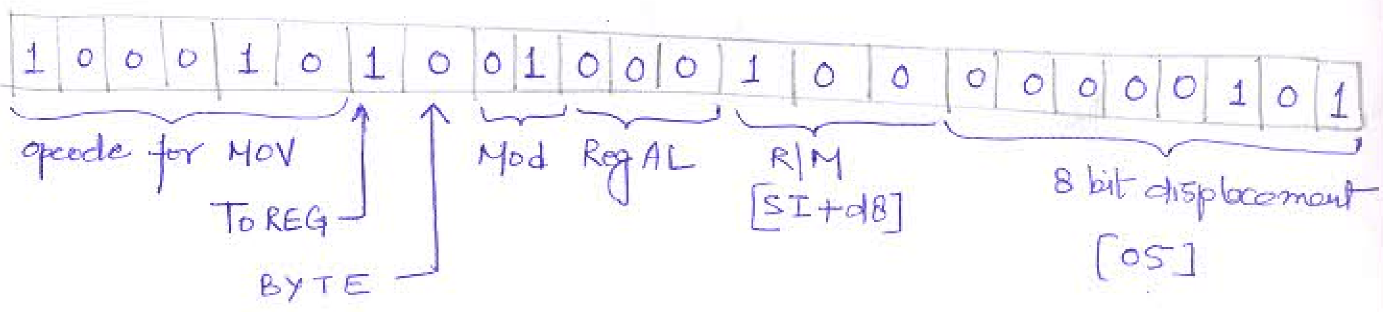
Course:	B.Tech	Semester:	V
Session:	2017-18	Section:	EC-1,2,3, EI-K
Subject:	Microprocessors	Sub. Code:	NEC-503
Max Marks:	50	Time:	2 hour

SECTION-A

QA

1. Generate Machine code for following instruction assuming the opcode for MOV as 100010 — MOV AL, [SI+05]

Ans.



2. What is the function of 8086 instruction queue? How does it speed up the processing?

Ans. To speed up the program execution, the BIU (Bus Interface Unit) fetches six instruction bytes ahead of time from the memory. These prefetched instruction bytes are held for the execution unit in a group of registers called Instruction Queue.

With the help of this queue, it is possible to fetch next instruction when current instruction is in ~~queue~~ execution. The BIU continues this process as long as the queue is not full. Due to this, execution unit gets the ready instruction in the queue & instruction fetch time is eliminated. feature of fetching the next instruction while the current ~~ex~~ instruction is executing is called pipelining

3. State the functions of control flags of 8086.

Ans. Three flags are control flags— They are used to control certain operations of the processor.

TRAP FLAG (TG) — To run a program one instruction at a time & see the contents of used registers & memory variables after execution of every instruction. This is called Single stepping through a program. If set, a trap (i.e. interrupt service routine) is executed after execution of each instruction, which displays various registers & memory variables. contents on the display after execution of each instruction. Programmer can easily trace & correct errors in the program.

INTERRUPT FLAG (IF) - It is used to allow / prohibit the interruption of a program. If set, a certain type of interrupt (a maskable interrupt) can be recognized by 8086, otherwise these interrupts are ignored.

DIRECTION FLAG (DF) — It is used with string instructions. If DF = 0, string is processed from its beginning with the first element having the lowest address. Otherwise, string is processed from high address towards the low address.

4 Define the functions of the following pins of 8086.
a) $\overline{\text{TEST}}$ b) $\overline{\text{LOCK}}$

Ans. $\overline{\text{TEST}}$ - This signal is used only by WAIT instruction. 8086 enters into wait state after execution of WAIT instruction until a low signal on $\overline{\text{TEST}}$ pin. $\overline{\text{TEST}}$ signal is synchronized internally during each clock cycle on the leading edge of clock cycle.

$\overline{\text{LOCK}}$ - This signal indicates that an instruction with a LOCK prefix is being executed and the bus is not to be used by another processor.

5 What are the functions of following Assembler directives? Explain with examples -
a) EXTRN
b) DT.

Ans. EXTRN - This is used to tell the assembler that the names or labels following the directive are in some other assembly module. If we want to call a procedure while is in a program module assembled at a different time from that which contains CALL instr, you must tell the assembler that procedure is external. The assembler will then put information in the object code file so that the linker can connect the two modules together.

eg EXTRN DIVISOR : WORD

DT — Define Ten Bytes.

This assembler directive tells the assembler to define a variable which is 10 bytes in length or to reserve 10 bytes of storage in memory.

eg RESULTS DT 20H DUP(0)

Array of 20H blocks of 10 bytes each & initialize all 320 bytes to 00.

SECTION-B

6. What is the need of memory segmentation in 8086? How the 20 bit effective address is calculated - Explain with example

Ans. Physical address of 8086 is 20 bits wide to access 1 MByte memory locations. However its registers & memory locations which contains logical address are just 16 bits wide. Hence 8086 uses memory segmentation. It treats 1 MByte of memory as divided into segments. Maximum size of segment is 64 KBytes. Thus any location within the segment can be accessed using 16 bits.

Segment registers are used to hold the upper 16 bits of starting addresses of the four segments of memory, on which 8086 works at a particular time. Starting address is also known as Base Address or Segment Base.

BIU always inserts zeros for the lower 4 bits (nibble) in the contents of segment register to generate 20-bit base address.

for eg - If Code segment registers contains 348A H, then code segment will start at address 348A0 H

The complete physical address (20 bits long) is generated using segment & offset registers, each 16 bit long. for generating a physical address, Contents of segment register are shifted left bitwise four times & to this result, content of an offset register is added, to produce a 20-bit physical Address.

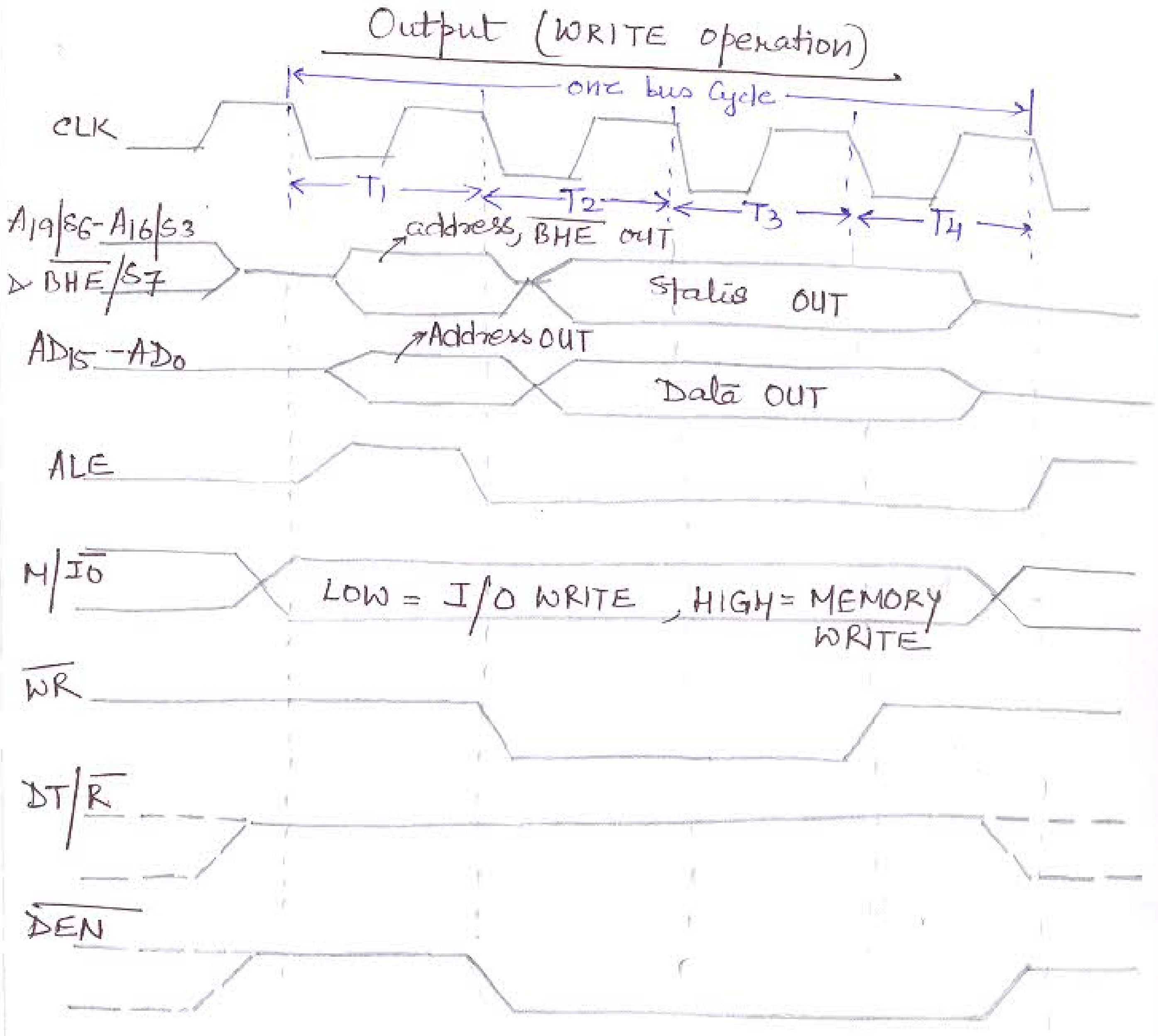
eg

Segment Address = 1005H
offset address = 5555H
Physical Address is calculated as—
1005H shifted by 4 bit positions

```
  ⇒        0001  0000  0000  0101  0000
then offset ⇒
address added    0101  0101  0101  0101
          ------------------------------
           0001  0101  0101  1010  0101
          ------------------------------
              ⇒  155A5 H  ⇒ 20 bit
                              physical Addr
```

7. Draw & explain the ~~[illegible]~~ Write Cycle timing diagram of 8086 microprocessor in minimum mode:—

Ans.



1. When processor is ready to initiate the bus cycle, it applies a pulse to ALE during T_1. Before the falling edge of ALE, the address, BHE, M/IO, DEN and DT/R must be stable. DEN = high & DT/R = 1

2. During T_2, the address signals are disabled & S3–S7 are available on AD16/S3 – AD19/S6 & BHE/S7. Also DEN is lowered to enable transceiver.

3. For an output operation, processor applies $\overline{WR} = 0$, and then the data on the data bus during T_2.

4. In T_4, $\overline{WR}$ is raised high & data signals are disabled.

5. $\overline{DEN}$ is raised during T_4 to disable the transceiver. Also $M/\overline{IO}$ is set according to the next tranfer at this time or during next T_1 state. Thus, length of bus cycle is 4 clock Cycles.

8. Explain the difference between SHORT JUMP, NEAR JUMP and FAR JUMP.

Ans. JUMP instructions are under control Transfer Group. This group of instructions will always cause 8086 to fetch its next instructions from location specified by instructions.

SHORT JUMP — This is two byte instruction that allows jumps or branches to memory locations within +127 and -128 bytes from address following the jump.

NEAR JUMP – Branch or jump within ± 32 KByte or anywhere in Current Code segment. The segments are cyclic in nature, ie. one location above offset address FFFFH is 0000H.

FAR JUMP – Allows a jump to any memory location within real memory system.

SHORT JUMP & NEAR JUMP are intrasegment jumps. FAR JUMP is intersegment jump

9. Write a program in assembly language using 8086 to convert a BCD number into a binary number.

Ans.
```
ASSUME CS: CODE   DS: DATA
DATA SEGMENT
OPER1  DB  89H
RESULT  DB  01  DUP (?)
DATA  ENDS

CODE  SEGMENT
 MOV  AX, 1000H
 MOV  DS, AX
 MOV  BX, OFFSET  OPER1
 MOV  BH, [BX]
 MOV  BL, [BX]
 AND  BH, F0
 ROR  BH, OH
 MOV  AL, 0A

 MUL  BH
 AND  BL, OF
 ADD  AL, BL
 MOV  DX, AL
 MOV  AH, 4CH
 INT  21H
 CODE  ENDS
  END  START
```

10. Differentiate between DOS & BIOS interrupts. Describe any two function calls of INT 21H with usage of registers & returns.

Ans Comparison b/w DOS & BIOS interrupts —

DOS	BIOS
1) DOS is loaded from the bootable disk.	1) BIOS is located in 8 kbyte ROM.
2) DOS programs offer higher level services, allowing more flexibility, portability & hardware independence.	2) Programs within ROM-BIOS provide the most direct, lowest level interaction with devices in the system.
3) DOS has ability to load & execute programs directly.	3) ROM-BIOS does not have ability to load & execute programs directly.
4) DOS can store data on disks organized as logical files.	4) ROM-BIOS cannot store data on disks organized as logical files.
5) DOS has a command interpreter to allow us to copy files, print files & delete files.	5) ROM BIOS has no command-interpreter to allow us to copy files, print files & delete files.

function calls of INT 21H

1. INT 21H function 01H
(character input with echo) — reads a character from standard input device & echos it to standard output device. If no character is ready, waits until one is available.

Calling parameter → AH = 01H
Returns AL = 8 bit input data

eg
```
Char db = 0
    mov ah, 01h
    int 21h      // transfer to MS DOS
    mov char, al
```

2. INT 21H (character Output)
function 02H.
Outputs the character to standard output device.
Calling parameter AH = 02H
DL = 8 bit data for output.
Returns Nothing.

eg
```
    Mov ah, 2
    mov dl, '*'
    int 21h.   // transfer to MSDOS.
```

(11)

SECTION-C

11. What are the different ways of specifying Effective Address (EA) in the instructions [illegible]. Explain addressing modes (with proper examples) for each.

Ans. Different [illegible] specifying EA [illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

a) Direct Addressing Mode – [illegible]

[illegible]

[illegible] MOV AL, [illegible]

[illegible] [illegible] = EA

b) Register Indirect Addressing Mode –

EA [illegible]

[illegible]

[illegible] BX [illegible] BP

[illegible] DI

[illegible]

c) Base-plus-index Addressing :— Similar to indirect addressing. This uses one Base register (BP or BX) and one index register (DI or SI) to indirectly access memory.

eg MOV CX, [BX+DI]

d) Register Relative Addressing — Similar to base-plus-index addressing mode. Data in the segment of memory are addressed by adding the displacement to the contents of a base or an index register,

eg MOV CX, [BX + 0003H]

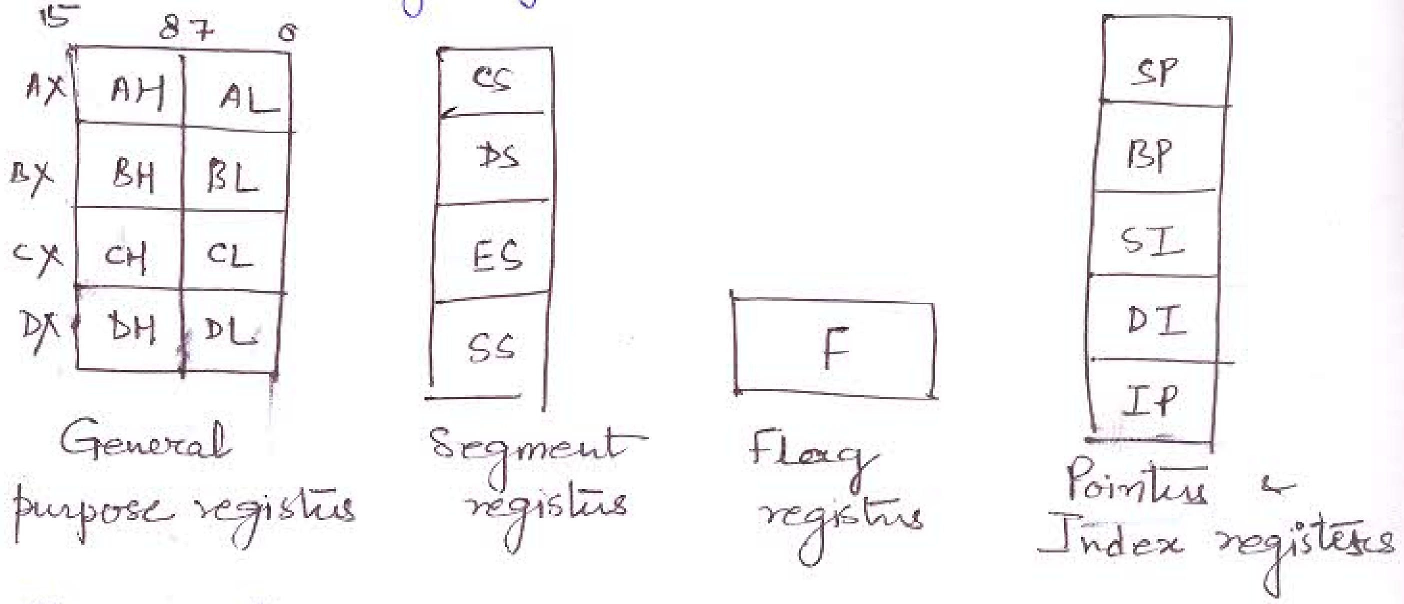
e) Base Relative plus Index Addressing — Similar to base plus Index addressing but it adds a displacement, besides using a base register and an index register to generate a physical address of the memory.

eg MOV AL, [BX + SI + 10H]

12. Draw the register organization of 8086 and explain typical application of each register. Also list out the signals of 8086 which have different meaning in minimum and maximum modes.

Ans. 8086 has powerful set of registers. It includes -

1. General purpose registers
2. Segment Registers
3. Pointers.
4. Index registers
5. flag register.



General Purpose Registers - 4 general purpose reg - AX, BX, CX, DX. Each can be split into two 8 bit registers. These registers are used for storing offset address for some particular addressing. AX is used as accumulator. BX is used for storing offset for generating physical ~~addressinges~~ in case of certain addressing modes. CX is used as default counter. DX is concatenated with AX to form 32 bit register for MUL & DIV operations.

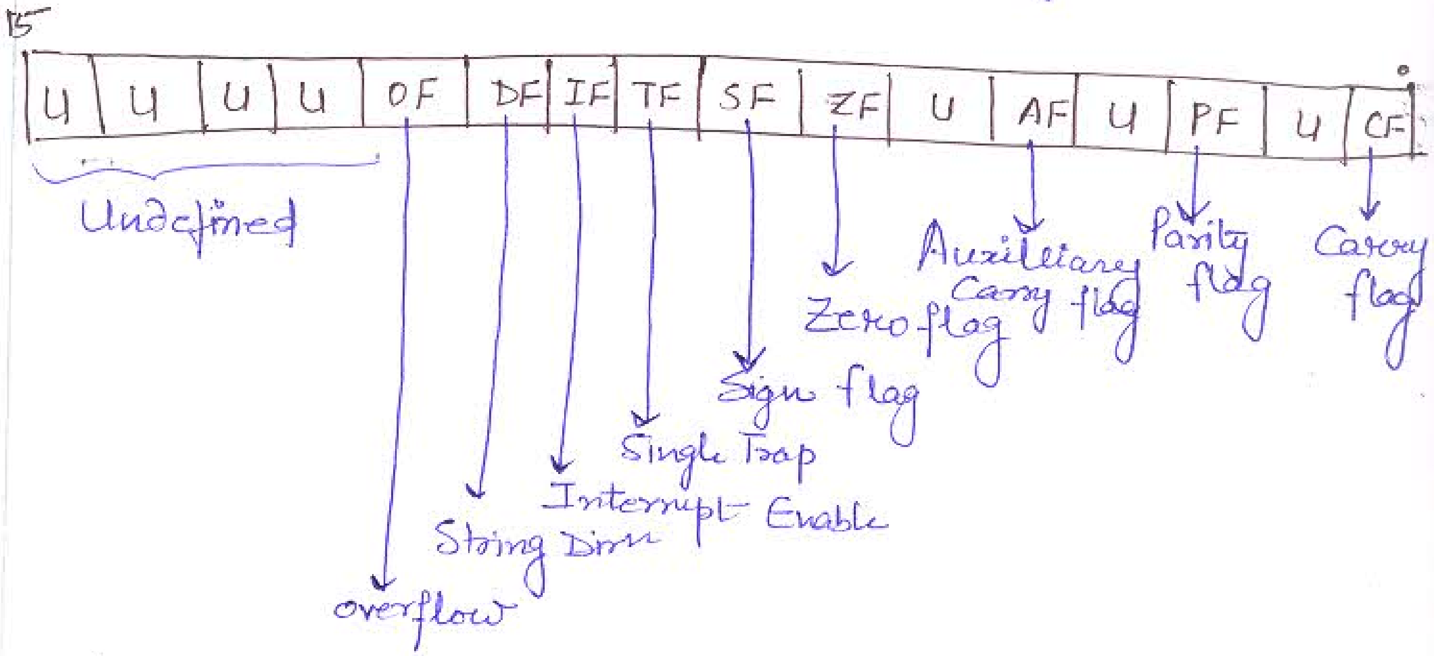
Segment Registers — for selection of four segments, 16 bit registers are provided by BIU of 8086. four segment registers are Code segment (CS) register, Data segment (DS) register, Stack segment (SS) registers, Extra (ES) registers. These are used to hold the upper 16 bits of the starting addresses of the four segments of memory, on which 8086 works at a particular time.

Pointers & Index registers — To get 20 bit physical address one or more pointer or index registers are associated with each ~~other~~ segment register. Pointer registers — Instruction pointer (IP), Base pointer (BP), Stack pointer (SP) are associated with Code, Data & Stack segments. They contain offset within the code, data & stack segments respectively.

Index registers (DI & SI) — Destination Index & Source Index registers are general purpose as well as for offset storage.

Flag Register — It is a flip flop which indicates some condition produced by the execution of an instruction or controls

certain operation of Execution Unit.
Flag registu contains nine active flags.



Signals having different meaning in minimum and maximum modes -

Signals for Minimum Mode (pin 24 to 31)

$\overline{INTA}$ (Interrupt acknowledge)
ALE (Address Latch enable)
$\overline{DEN}$ (Data Enable)
$DT/\overline{R}$ (Data Transmit / Receive)
$M/\overline{IO}$ (Memory / Input Output)
$\overline{WR}$ (Write Output)
HOLD
HLDA (HOLD Acknowledge)

Signals for Maximum Mode (pin 24 to 31)

QS_1, QS_0 (Queue status)

$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ (~~output~~ status signals)

$\overline{LOCK}$

$\overline{RQ}/\overline{GT_1}$ and $\overline{RQ}/\overline{GT_0}$ (Request / Grant)

END