1. **What is the difference between DATA LABEL and CODE LABEL?**

Data Label is the label that we use to define data as we defined memory locations num1,num2 ....etc in our programs. Code Label is the label that we have on code as we see in the case of conditional jump (Label l1) and is normally used for loop control statements.

2. **List the seven addressing modes available in the 8088 architecture.**

    1. Direct
    2. Base register Indirect
    3. Indexed register Indirect
    4. Base register indirect + offset
    5. Index register indirect +offset
    6. Base+index+offset

3. Differentiate between effective address and physical address.

The **effective address** is the address generated by the program, after all, transformations, such as index registers, offsets, addressing mode, etc. have been made. The **physical address** is the address generated by the hardware, after performing whatever lookups through the page table, etc. have been made. The effective address, or a virtual address, is the concern of the program. The physical address, or real address, is the concern of the operating system.

4. What is the effective address generated by the following instructions? Every instruction is independent of others. Initially BX=0x0100, num1=0x1001, [num1]=0x0000, and SI=0x0100

a. mov ax, [bx+12]
b. mov ax, [bx+num1]
c. mov ax, [num1+bx]
d. mov ax, [bx+si]

a. bx+12 = 0x0100 + 0xc = 0x010c
b. bx+num1 = 0x0100 + 0x1001 = 0x1101
c. num1+bx = 0x1001 + 0x0100 = 0x1101
d. bx+si = 0x0100 + 0x100 = 0x0200

5. What is the effective address generated by the following combinations if they are valid. If not give reason. Initially BX=0x0100, SI=0x0010, DI=0x0001, BP=0x0200, and SP=0xFFFF

a. bx-si
b. bx-bp
c. bx+10

d. bx-10
e. bx+sp
f. Bx+di

6.Identify the problems in the following instructions and correct them by replacing them with one or two instructions having the same effect.

a. mov [02], [22]
b. mov [wordvar], 20
c. mov bx, al
d. mov ax, [si+di+100]

a)
mov ax, [22]
mov [02], ax
OR
mov al, [22]
mov [02], [al]
b)
mov al, 22
mov [wordvar], al
c)
mov bl, al
d)
Mov bx, si
Mov ax, [bx+di+100]

7. What is the function of segment override prefix and what changes it brings to the opcode?
Solution:
The function of Segment Override Prefix:
The segment override prefix, which may be added to almost any instruction in any memory addressing mode, allows the programmer to deviate from the default segment. The segment override prefix is an additional byte that appends the front of an instruction to select an alternate segment register. The only instructions that cannot be prefixed are the jump and call instructions that must use the code segment register for address generation. For example, the MOV AX, [DI] instruction accesses data within the data segment by default. If required by a program it can be changed by prefixing the

instruction. Suppose that the data are in the extra segment instead of in the data segment. This instruction addresses the extra segment if changed to MOV AX, ES:[DI].

Changes to Opcode:

Opcode never changes, but the prefix byte modifies the default association to association with the desired segment register for the instruction.

8. What are the two types of address wraparound? What physical address is accessed with [BX+SI] if FFFF is loaded in BX, SI, and DS.

Solution:

There are two types of address wraparounds:

1. Within a single segment
2. Inside the whole megabyte/physical memory

Within a single segment

During the effective address calculation when a carry is generated then Segment Wraparound occurs. This carry is dropped giving the effect that when we try to access beyond the segment limit, we are actually wrapped around to the first cell in the segment. Just like a circle when we reached the end then we started again from the beginning. An arc at 370 degrees is the same as an arc at 10 degrees. We tried to cross the segment boundary and it pushed us back to the start. This is called segment wraparound.

For example if BX=9100, DS=1500 and the access is [bx+0x7000] we form the effective address 9100 + 7000 =10100. The carry generated is dropped giving the actual effective address of 0100. The physical address in this example will be 15100.

Inside the whole physical memory: Just like segment wraparound, the whole physical memory wraparound occurs. In Segment Wraparound memory address is pushed back to the start of the segment while in other cases address is pushed back to the very start of the physical memory. The following example, best describes the idea of a whole memory wraparound.

EXAMPLE: BX=0100, DS=FFF0 and the access under consideration is [bx+0x0100]. The effective address will be 0200 and the physical address will be 100100. This is a 21bit answer and cannot be sent on the address bus which is 20 bits wide. The carry is dropped and just like the segment wraparound, our physical memory has wrapped around at its very top. When we tried to access beyond limits the actual access is made at the very start.

Physical Address Calculation:

As

SI = FFFF

BX = FFFF

Segment : DS = FFFF

SI+BX= FFFF+FFFF= FFFE= offset address

Physical address = segment x 10 + offset address

= FFFF0+FFFE

= 0FFEE

9. Write instructions to do the following.

a. Copy contents of memory location with offset 0025 in the current data segment into AX.

b. Copy AX into memory location with offset 0FFF in the current data segment.

c. Move contents of memory location with offset 0010 to memory location with offset 002F in the current data segment.

Solution:

```
a. mov SI, 0x0025
 mov AX, [SI]
b. mov [DI], 0x0FFF
 mov [DI], AX
c. mov DI, 0x002F
 mov SI, 0x0010
 mov AX, [SI]
 mov [DI], AX
```

10. Write a program to calculate the square of 20 by using a loop that adds 20 to the accumulator 20 times.

Solution:

```
[org 0x0100]
mov bx,20
mov cx,20
mov ax,0
l1:
 add ax, bx
 sub cx, 1
 jnz l1
 mov [total], ax
```

```
 mov ax,0x4c00
 int 0x21
total: dw 0
```