1) 5 flags thy onki defines likhni the

1) C = Carry = This extra bit that won't fit in the target register is placed in the carry flag where it can be used and tested.
2) P = Parity = Parity is the number of "one" bits in a binary number. Parity is either odd or even.
3) A = Auxiliary Carry = auxiliary carry is the carry from the first nibble to the second.
4) Z = Zero Flag = The Zero flag is set if the last mathematical or logical instruction has produced a zero in its destination
5) S = Sign Flag = A signed number is represented in its two's complement form in the computer.
6) T = Trap Flag = The trap flag has a special role in debugging which will be discussed later.
7) I = Interrupt Flag = It tells whether the processor can be interrupted from outside or not.
8) D = Direction Flag = this flag tells whether the current operation has to be done from bottom to top of the block (D=0) or from top to bottom of the block (D=1).
9) O = Overflow Flag = The overflow flag is set during signed arithmetic, e.g. addition or subtraction, when the sign of the destination changes unexpectedly

2) ASCII code likhna tha a ka

A = 065, a = 097

3) How MOVS work

MOVS is the instruction that allows memory to memory moves. MOVS transfers a byte or word from the source location DS:SI to the destination ES:DI and updates SI and DI to point to the next locations. MOVS is used to move a block of memory.

4) Difference near and short jump

A short JMP is the relative JMP that you refer to. It is encoded as a two bytes; the actual JMP and the number of bytes +/- relative to the current IP. A near jump **allows you to jump within the current "segment**" (using real mode terms) or within the currently selected memory area in the CS selector.

5) Local variable and where to store it

We modify the bubble sort subroutine to use a local variable to store the swap flag. A local variable should be used to store the return value from the first recursive call.

6) selective bit inversion on lowest nibble 100110b.

Selective Bit Inversion

XOR can also be used as a masking operation to invert selective bits. The bits in the mask are cleared at positions, which are to retain their values, and are set at positions, which are to be inverted.

7) What is the function of segment override prefix and what changes it brings to the opcode?

Solution: 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.

8) 2 long thy or dono ma instructions di hwi thi wo execute kr k onky components btany aur attribute bytes ky display colours / Background and foreground colors k attributes
The attribute byte as detailed below has the RGB for the foreground and the background. It has an intensity bit for the foreground color as well thus making 16 possible colors of the foreground and 8 possible colors for the background.

7 6 5 4 3 2 1 0
7 – Blinking of foreground character
6 – Red component of background color
5 – Green component of background color
4 – Blue component of background color
3 – Intensity component of foreground color
2 – Red component of foreground color
1 – Green component of foreground color
0 – Blue component of foreground color

9) How are local variables created?
Local variables are created when the method, constructor or block is entered and the variable will be destroyed once it exits the method, constructor, or block. ... Local variables are visible only within the declared method, constructor, or block. Local variables are implemented at stack level internally.

10) Process of calling a subroutine keeping in mind stack data structure and using inter segment direct call?
Subroutine nesting is a common Programming practice In which one Subroutine call another Subroutine. As the last Subroutine called is the first one to be returned (Last in first out format). So stack data structure is the most efficient way to store the return addresses of the Subroutines.

11) LDS ka aya tha
LES loads ES while LDS loads DS. Both these instructions have two parameters, one is the general purpose register to be loaded and the other is the memory location from which to load these registers.

12) Inter or intra ka ayaa tha
Near calls are also called intra segment calls, while far calls are called inter-segment calls.

13) LOADS ki definition:
LODS transfers a byte or word from the source location DS:SI to AL or AX and updates SI to point to the next location. LODS is generally used in a loop

14) REP intsruction working
REP Prefix: REP repeats the following string instruction CX times. The use of CX is implied with the REP prefix. The decrement in CX doesn't affect any flags and the jump is also independent of the flags, just like JCXZ.

15) movement of busses between memory and processor
Address bus - carries memory addresses from the processor to other components such as primary storage and input/output devices. The address bus is unidirectional .
Data bus - carries the data between the processor and other components.

16) define byte with example
A byte is a unit of memory data equal to either seven or eight bits depending on whether or not it needs error correction (parity). ... You can think of a byte as one letter, for example, the letter 'h' is one byte or eight bits and the word 'hope' as four bytes or 32 bits (4*8).

17) Assembly Directives kia hte hain detail mn btana tha
Directives are commands that are part of the assembler syntax but are not related to the x86 processor instruction set. All assembler directives begin with a period (.) align directive causes the next data generated to be aligned modulo integer bytes.

18) Vectored kia hte hai 8088 architecture mn
The software interrupt mechanism in 8088 uses vectored interrupts meaning that the address of the interrupt routine is not directly mentioned in an interrupt call, rather the address is lookup up from a table. 8088 provides a mechanism for mapping interrupts to interrupt handlers.

19) Type The Assembly Language Program Of Add 10 Numbers 10,20,30,40,50,10,20,30,40,50

Example 2.7

```
001     ; a program to add ten numbers
002     [org 0x0100]
003                 mov  bx, num1           ; point bx to first number
004                 mov  cx, 10             ; load count of numbers in cx
005                 mov  ax, 0              ; initialize sum to zero
006
007     l1:         add  ax, [bx]           ; add number to ax
008                 add  bx, 2              ; advance bx to next number
009                 sub  cx, 1              ; numbers to be added reduced
010                 jnz  l1                 ; if numbers remain add next
011
012                 mov  [total], ax        ; write back sum in memory
013
014                 mov  ax, 0x4c00         ; terminate program
015                 int  0x21
016
017     num1:       dw   10, 20, 30, 40, 50, 10, 20, 30, 40, 50
018     total:      dw   0
```

20) Write Three name of assembly logical part.
We have two logical parts of our program, the code and the data, and actually there is a third part called the program stack as well, but higher level languages make this invisible to us. These three logical parts of a program should appear as three distinct units in memory, but making this division is not possible in the linear memory model.

21) working of linear memory model?

In earlier processors like 8080 and 8085 the linear memory model was used to access memory. In linear memory model the whole memory appears like a single array of data.

22) Difference between shift logical right and shift arithmetic right?

Shift Logical Right (SHR)

The shift logical right operation inserts a zero from the left and moves every bit one position to the right and copies the rightmost bit in the carry flag.

Shift Arithmetic Right (SAR)

A signed number holds the sign in its most significant bit. If this bit was one a logical right shifting will change the sign of this number because of insertion of a zero from the left.

23) Write a program for multiplication of Two 4 bits number

Example 4.1

```
01      ; 4bit multiplication algorithm
02      [org 0x100]
03                  jmp  start
04
05      multiplicand: db   13                ; 4bit multiplicand (8bit space)
06      multiplier:   db   5                 ; 4bit multiplier
07      result:       db   0                 ; 8bit result
08
09      start:        mov  cl, 4             ; initialize bit count to four
10                    mov  bl, [multiplicand] ; load multiplicand in bl
11                    mov  dl, [multiplier]  ; load multiplier in dl
12
13      checkbit:     shr  dl, 1             ; move right most bit in carry
14                    jnc  skip              ; skip addition if bit is zero
15
16                    add  [result], bl      ; accumulate result
17
18      skip:         shl  bl, 1             ; shift multiplicand left
19                    dec  cl                ; decrement bit count
20                    jnz  checkbit          ; repeat if bits left
21
22                    mov  ax, 0x4c00        ; terminate program
23                    int  0x21
```

24) The* following code is taken from bubble sort program fill the missing lines in which you much have and restore register value using stack operation bubbles sort .

_____;save old value of ax

_____; save old value of cs

_____;save old value of si

```
bubblesort:   push ax                  ; save old value of ax
              push cx                  ; save old value of cx
              push si                  ; save old value of si
```

25) difference between logical address and physical address

The logical address is used like a reference, to access the physical address. The fundamental difference between logical and physical address is that logical address is generated by CPU during a program execution whereas, the physical address refers to a location in the memory unit.

26) Explain the working of STOS
STOS transfers a byte or word from register AL or AX to the string element addressed by ES:DI and updates DI to point to the next location. STOS is often used to clear a block of memory or fill it with a constant.

27) write alternative instructions for CF,JNZ with character
CF = 1: This jump is taken if the last arithmetic operation generated a carry or required a borrow.
CF = 0: This jump is taken if the last arithmetic operation did not generated a carry or required a borrow.
JNZ = Jump if not zero = ZF = 0: This jump is taken if the last arithmetic operation did not produce a zero in its destination.

28) how to invert nibble?
Selective Bit Inversion
XOR can also be used as a masking operation to invert selective bits. For example to invert the lower nibble of the destination operand, the operand should be applied with a mask of 0F in the source. The upper nibble will retain its value and the lower nibble will be set as a result.

29) REPE and REPME with zero flag
REPE or REPZ repeat the following string instruction while the zero flag is set and REPNE or REPNZ repeat the following instruction while the zero flag is not set. REPE or REPNE are used with the SCAS or CMPS instructions.

30) used btyes by db vs dw

Directive	Purpose	Storage Space
DB	Define Byte	allocates **1 byte**
DW	Define Word	allocates 2 bytes
DD	Define Doubleword	allocates 4 bytes
DQ	Define Quadword	allocates 8 bytes

31) Name the registers which have higher and lower halves. Only names
one operand must be in register, ax is used to read the lower and upper halves of the source one by one. The destination is directly updated.

32What is test instructions?

The test instruction is used for bit testing. BX holds the mask and in every next iteration it is shifting left, as our concerned bit is now the next bit

33) Keeping in view of stack data structure and using integer segment how subroutine direct call.

The stack is an area of memory; the stack pointer is the address of the last value pushed onto the stack. Usually, the stack is used for storing data when subroutines are called. The stack is a last-in-first-out, i.e., LIFO structure so the last thing stored in the stack is the first thing retrieved.

34) Computer Bus and their functionality

A bus is a high-speed internal connection. Buses are used to send control signals and data between the processor and other components. Three types of bus are used. Address bus - carries memory addresses from the processor to other components such as primary storage and input/output devices.

To add 50 numbers, which method you will prefer indirect or direct and why?

Ill prefer indirect addressing mode bcz by using this method i can save first element of an array then by using index register i can move 2nd element. On the other hand memory is fixed in direct addressing method and we need more registers for addition.

Q2. Explain peek operation in stack? (3).

Peek operation is used to get value from top of the stack without removing or incremented stack pointer.

Q3. Write instructions of following.(5)

copy dl to al.
copy ax to dx.
store 0x12 in al.
store 0x1234 in ax.
store 0xFFFF in ax.
Ans:
mov al, dl
mov dx, ax
mov al, 18
mov ax, 4660
mov ax, 65535
used scientific cal for hexa to dec

Q4.write behaviour of zero flag in REPE and REPNE (5).

Ans:
REPE.
repeat while equal prefix repeat block of instruction while zero flag is set or z=1
REPNE.
repeat while not equal repeat block of instruction while zero flag is not set or z=0

Q5. write stack behaviour in intra segment and inter segment (5).
Ans:
In intra segment sp is decremented by 2 and ip is pushed on top of the stack.
In inter segment sp is decremented by 2 and CS is pushed on top of the stack

What does interrupt flag tells when it is not set and when it is set

The Interrupt flag (IF) is a flag bit in the CPU's FLAGS register, which determines whether or not the (CPU) will respond immediately to maskable hardware interrupts. If reset (set to 0) such interrupts will be disabled until interrupts are enabled.

Rotate Through Carry Left (RCL)
The exact opposite of rotate through carry right instruction is the rotate through carry left instruction. In its operation the carry flag is inserted from the right causing every bit to move one location to its left and the most significant bit occupying the carry flag.

NOT operation
NOT inverts the bits (forms the one's complement) of the byte or word operand. Unlike the other logical operations, this is a single operand instruction, and is not purely a logical operation in the sense the others are, but it is still traditionally counted in the same set.
control bus: A control bus is a computer bus that is used by the CPU to communicate with devices that are contained within the computer. This occurs through physical connections such as cables or printed circuits.

Assembly language:
An assembly language is **a type of low-level programming language that is intended to communicate directly with a computer's hardware**. Unlike machine language, which consists of binary and hexadecimal characters, assembly languages are designed to be readable by humans.

Which registers are used to access the stack?

The register used to access the stack is known as **the stack pointer register**. The stack pointer in the 8051 is 8-bits wide, and it can take a value of 00 to FFH.
If a large number is divided by a very small number it is possible that the quotient is larger than the space provided for it in the implied destination. In this case an interrupt is automatically generated and the program is usually terminated as a result. This is called a divide overflow error

23) mov ax, 10 mov ax 0x10 mov ax [0 x10] ko apni opnion my btana ta...3 marks ka
mov ax, 10 (store constant 10 in ax register)
mov ax, 0x10 (store address 0x10 in ax register)
mov ax, [0x10] (store content of address 0x10 in ax register)

25) Ascii code ka syntax btana according to given db'abc'
The three ways used below are identical in their meaning.
db 0x61, 0x62, 0x63
db 'a', 'b', 'c'

db 'abc'

26) List down corresponding logical segment IP SI and DI BP
CS is associated to IP by default; rather it is tied with it. It cannot access memory in any other segment. if the register used is one of SI, DI, or BX the default segment is DS. If BP the default segment used is SS.IP is tied to CS while SP is tied to SS.

31) Write assembly language code of 32 bit by using subtraction extended.

```
dest: dd 40000 --- src: dd 80000 --- mov ax, [src] --- sub word [dest], ax --- mov ax, [src+2] ---
- sbb word [dest+2], ax
```

```
01      ; clear the screen
02      [org 0x0100]
03                  mov  ax, 0xb800         ; load video base in ax
04                  mov  es, ax             ; point es to video base
05                  mov  di, 0              ; point di to top left column
06
07      nextchar:   mov  word [es:di], 0x0720 ; clear next char on screen
08                  add  di, 2              ; move to next screen location
09                  cmp  di, 4000           ; has the whole screen cleared
10                  jne  nextchar           ; if no clear next position
11
12                  mov  ax, 0x4c00         ; terminate program
13                  int  0x21
```

Example 7.1

```
001     ; clear screen using string instructions
002     [org 0x0100]
003                 jmp  start
004
005     ; subroutine to clear the screen
006     clrscr:     push es
007                 push ax
008                 push cx
009                 push di
010
011                 mov  ax, 0xb800
012                 mov  es, ax             ; point es to video base
013                 xor  di, di             ; point di to top left column
014                 mov  ax, 0x0720         ; space char in normal attribute
015                 mov  cx, 2000           ; number of screen locations
016
017                 cld                     ; auto increment mode
018                 rep  stosw              ; clear the whole screen
019
020                 pop  di
```

```
021                 pop  cx
022                 pop  ax
023                 pop  es
024                 ret
025
026     start:      call clrscr             ; call clrscr subroutine
027
028                 mov  ax, 0x4c00         ; terminate program
029                 int  0x21
```

why we use JAE in assembly language

The mnemonics jae is recommended after compare instruction (cmp) which does subtraction. You can get more details in Intel or AMD software developer manuals. jae means Jump if above or equal. It will jump if the carry flag is equal to 0.

Which two instructions are used to check particular bits of binary are set or not? How the differ.

Selective Bit Testing
AND can be used to check whether particular bits of a number are set or not. AND can be applied on a destination with a 1-bit in the desired position and a source, which is to be checked.

MOVS

MOVS transfers a byte or word from the source location DS:SI to the destination ES:DI and updates SI and DI to point to the next locations.

Practical Questions:

1) show the illustration of RET and CALL value
2) 1 code tha uska effective address btana tha
3) 3 local variable ko create krny ki instructions ai th
4) Rewrite program and convert Dw To Db
5) basic commands likhin thin or un main error find krna tha like move ax, bx wrong ha or mov ax, bx right ha
6) selective bit inversion on lowest nibble 100110b.
7) 2 long thy or dono ma instructions di hwi thi wo execute kr k onky components btany thy...
8) 2 long thy or dono ma instructions di hwi thi wo execute kr k onky components btany thy...
9) Identify the operation and operands from the given codes.
Mov ax, 5
Mov bx, 10
10) Name the registers which have higher and lower halves. Only names
11) Calculate the physical memory address generated by the following segment offset pairs.
a. 1DDD:0436. b. 1234:7920. c. 74F0:2123
12) attribute bytes values
13) program tha registers wala lines put krni the aur error sahi krny
14) variable ki do function lekhy thy aur match krana tha set of each other with example
15) in video memory, what information is expressed by each byte?
16) CF=1 and AX=0x64,new value of AX register and carry flag after execute of RCL AX, 1?
17) stack data structure using direct call segment?
18) column matching thi functions ki
19) Ik instruction thi MOV [AX+BX+30] iska btana tha k is m konsa physical design ya strategy use hogi? Marks 5
20) ik statement Di v thi uski assembly languages code m lekhna tha 5marks
21) Suppose we have CX = 1 and Ax =0×64 what will be the new value of AX register and carry flag after the execution of RCL, AX ,1?
22) Write a program that define a word size array of two elements, calculate their sum and store the result in a new variable sum?
23) mov ax, 10 mov ax 0x10 mov ax [0 x10] ko apni opnion my btana ta...3 marks ka

24) 4 opertions diye ty usko detail mai btana just like store 0x123 into AX, store 0x1234 into AX store 0xFFFF into AX n copy DX into AX
25) Ascii code ka syntax btana according to given db'abc'
26) List down corresponding logical segment
IP
SI and DI
BP
27) Write a program in assembly language
HiIf(AX<BX<CX)&&(BX<DX<CX)
A==5. Marks5
28) Or ak sawal 5 Ka ta "inthook" say realted program ta vo bnana ta
29) write a program square of 3 accumulator terminates 2 times
30) write code to display big zero on dos screen
31) Write assembly language code of 32 bit by using subtraction extended.
Ek Question tha k 0741 ka matlab batana hai
1. Mov ax, 4
Not ax
What is value of ax now.
Write an assembly program to add two numbers in 32 bit memory
What is little endian and big endian . Iapx88 use which one?

3.
Ax = FF34 , Bx = F342
Cx = fefe , dx = fe43 stack = 0×0100
(stack k elawa kisi ki exact value yad nai)

Push Ax
Push Bx
Xchg ax, cx
Pop bx
Pop cx

Write the following position after these operations
Ax ?
Cx?
Bx?
Dx?

101000001 is trah se value thi is to convert krna tha

5 value d thi un ko ka error dekhna tha or explain krna tha

Addressing mode ki 3 ligel instructions likhain

Ik question ma physical address find krna tha?

32 binary ka code likhna tha.

a query was given we have to find the physical address

a code was given.we hv to find valid and invalid statement

1. Replace the following valid instruction with a single instruction that has the same effect. Don't consider the effect on flags.

```
    push word L1
    jmp L2
L1:
```

2. Replace the following invalid instructions with a single instruction that has the same effect.

```
a.  pop  ip
b.  mov  ip, L5
c.  sub  sp, 2
    mov  [ss:sp], ax
d.  mov ax, [ss:sp]
    add sp, 2
e.  add  sp, 6
    mov  ip, [ss:sp-6]
```