**ANS_01**

I. **Register Parameters: Easy to use, by name access / Limited number of registers.**
**Stack Parameters: More parameters than register / Difficult to access.**

II. **Local directive Creates local variable by name.**
**Local directive can create un-equal size local variables.**

III. **Both are invalid, parameters/Local variable address is not static , so can't used ADDR/OFFSET.**
**LEA esi, A1 ; Instruction 1**
**LEA edi, A2 ; Instruction 2**

IV. **FFF6 h**

V. **MOV EBX,EAX**
**MOV ECX, EAX**
**SHL EAX, 5 ; 32**
**SHL EBX, 2 ; 4**
**SUB EAX,EBX ; 32-4= 28**
**SUB EAX, ECX ; 28-1 =27**

VI. **B.**

VII. **CMP is non-destructive instruction, only updates flag**
**SUB is destructive instruction, updates both flag and operand.**

VIII. **The SCASW instruction compare a value in AX to a word, addressed by EDI.**
**LODSW load a word from memory at ESI into AX.**
**Array WORD 1,2,3,4,5,6**
**MOV ECX , LENGTHOF Array**
**MOV AX, 3**
**MOV EDI, OFFSET Array**
**REP SCASW**
**MOV ECX , LENGTHOF Array**
**MOV ESI, OFFSET Array**
**REP LODSW**

IX. **The MOVSX instruction (move with sign-extend) copies the contents of a source operand into a destination operand and fills the upper half of the destination with a copy of the source operand's sign bit.**
**The CBW instruction (convert byte to word) extends the sign bit of AL into AH.**

X. **PUSHFD Pushes EFLAG Register on STACK**
**PUSHAD Pushes 32-bit General Purpose Registers on STACK.**

**ANS_02: Assembly to MachineCodes**

I. MOVZX CX, VAR1 ; Var1= 18h → **0F B7 0E H**
II. POP 0F12C [BX] →**8F 87 2C F1 H**
III. MOV BX, DX → **89 D3 H**
IV. INC [BP][DI]+ 2C → **FF 43 2C H**
V. RCL [BX] , 3 → **C1 17 3 H**

**ANS_03: MachineCodes to Assembly**

I. 03 84 2B 1A → **ADD AX , [SI +1A2B]**
II. 2B 1D → **SUB BX,[DI]**
III. F6 D3 → **NOT BL**
IV. 56 → **PUSH SI**
V. 0B 0A → **OR CX,[BP][SI]**

**ANS_04:**

```
.DATA
input BYTE 0,0,1,0,1,0,0,0
dVal BYTE  128,64,32,16,8,4,2,1
Decimal  DWORD ?

.CODE
MAIN PROC
        MOV ESI , OFFSET input
        MOV EDI ,OFFSET dVal
        MOV ECX, 8
        L1:
                MOV AL,[ESI]
                MOV BL,[EDI]
                MUL BL
                MOVZX EDX,AX
                ADD Decimal , EDX
                INC ESI
                INC EDI
                LOOP L1
MAIN ENDP
END MAIN
```

**ANS_05:**

```
op1  QWORD    4 DUP(1234567812345678h)
op2  QWORD    4 DUP(8765432187654321h)
.code
main PROC
        mov esi, OFFSET op1
        mov edi,OFFSET op2
        mov ebx, OFFSET diff
        mov ecx,8 ; number of doublewords
        call Extended_Sub
        mov esi, offset diff
        mov ebx,4
        mov ecx, LENGTHOF diff
        call DumpMem
        exit
main ENDP

Extended_Add Proc
        Pushad
        mov eax,0
        clc
```

```
        L1:
        mov eax,[esi] ; get the first integer
        ADC eax,[edi] ; subtract the second integer
        pushfd ; save the Carry flag
        mov [ebx],eax ; store partial result
        add esi,4 ; advance all 3 pointers
        add edi,4
        add ebx,4
        popfd ; restore the Carry flag
        loop L1 ; repeat the loop
        ; repeat the loop
        ADC word ptr [ebx],0 ; left over borrow
        popad
        ret
Extended_Add ENDP
End main
```

**ANS_06:**

```
.stack 100h
.code
MAIN PROC
        mov ecx, lengthOF string_1
        dec ecx
        mov esi,0
        ;------ Push all the elements of String_1 into the STACK --------
        L1:
        push string_1[esi]
        Loop L1
        ;---------------- word Count ----------------
        mov ecx, length
        L2:
        pop eax
        cmp eax,20h
        jnz FR
        inc count
        FR:
        Loop L2
        inc count
        ;---------------- Display word Count ----------------
        mov edx, offset msg0
        call WriteString
        mov eax, count
        call WriteInt
MAIN ENDP
```

**ANS_07:**

```
.data
        Destination BYTE   "abcdef",0
        Source      BYTE   "cfqe",0
        Result      BYTE  4 DUP(?)
.code
Main PROC
```

```
        Mov ebx, offset Result
        mov esi, offset Source
        L1:
                Mov edi, offset Destination
                mov ecx, lengthof Destination
                LODSB
                repne SCASB
                jnz quit
                Mov [EBX],AL
                inc EBX
                quit:
        LOOP L1
Main ENDP
END Main
```

**ANS_08(a):**

```
while N > 0
{
  if (N != 3) AND (N < A OR N > B)
        N = N − 2
  else
        N = N − 1
}
```

```
L_WHILE:
CMP N,0
JG L1
JMP L_EXIT
L1:
CMP N,3
JNE L_C2
JMP L_FALSE
L_C2:
        MOV EAX,A
        CMP N, EAX
        JL  L_TRUE

        MOV EAX,B
        CMP N, EAX
        JG L_TRUE
        JMP L_FALSE
L_TRUE:

SUB N,2
JMP L_WHILE

L_FALSE:
DEC N
JMP L_WHILE
L_EXIT:
        EXIT
```

**ANS_08(b):**

```
MOV AL,0D4H
SHL AL,3                ; a. A0 H
MOV AL,0E4H
SAR AL,3                ; b. FC H
STC
MOV AL,0ABH
ROL AL,27               ; c. 5D H
STC
MOV AL,10H
RCR AL,1                ; d. 88 H
```