Ayesha Mehmood
ID # 6832

Assignment #6th

Answers

1) and ax, 00FFh

2) or ax, OFF00h

3) xor eax, OFFFFFFFFh

4) test eax, 1 : (low bit set if eax is odd)

5) or al, 00100000b

26) and al, 00001111b

27)
```
.data
memval DWORD ?
.code
mov al, BYTE PTR memval
xor al, BYTE PTR memval+1
```

xor al, BYTE PTR memval + 2

xor al, BYTE PTR memval + 3

6) JA , JNBE, JAE , JNB ,
JB , JNAE , JBE , JNA .

7) JG, JNLE, JGE, JNL, JL,
JNGE, JLE, JNG.

8) No, because the jg is used with signed value and (809h is negative, and 26h is positive.)

24) Yes

25) Yes (the unsigned representation of -42 is compared to 26)

28) Cmp dx, cx

jbe L1

29) Cmp ax, cx

jg L2

30) and al, 11111110b

Jz L3
jmp L4

16) (a) 01101101

(b) 01001000

(c) 01101111

(d) 10100011

18) (a) CF = 0, ZF = 0, SF = 0

(b) CF = 0, ZF = 0, SF = 0

(c) CF = 1, ZF = 0, SF = 1

19) JECXZ

17) (a) 85h

(b) 34h

(c) BFh

(d) AEh

34) INCLUDE Irvine32.inc

N = 10

.data

array SDWORD N DUP (-10, -8, -6, -4,

-2, -1, 1, 3, 5, 7)

```
j DWORD ?
K DWORD?
.code
main PROC
Call Clrscr
mov j,0
mov K ,10
mov ESI, OFFSET array
mov ECX, N
Call SummingArrayElementsInRange
Call WriteInt
Call crlf
mov j, -10
mov K, 0
```

```
mov ESI, OFFSET array
mov ECX, N
Call Summing Array Elements In Range
Call writeInt
Call Crlf
Call waitMsg
exit
main ENDP
Summing Array Elements In Range PROC
Push ecx
Push esi
mov eax, 0
L1:
mov ebx, [esi]
```

(7)

```
Cmp   ebx , j
jge   true1
Jmp   next
true1:
    Cmp ebx , K
    jle  true2
    jmp  next
true2:
    add eax , ebx
next:
    add esi , 4
Loop l1
    Pop esi
    Pop ecx
```

ret

Summing Array Elements In Range ENDP

END main

35)

Include Irvine32.inc

.data

byte1 BYTE 11111110b, 11011101b, 10001111b, 11001100b, 11001010b, 11001010b, 11001010b, 1100

byte2 BYTE 11111110b, 11011111b, 10001110b, 11101001b, 11001100b, 11001011b, 11001010b, 11001010b, 1100

.code

```
main PROC
    mov esi, OFFSET byte1
mov ecx, SIZEOF byte2
 Call PECheck
 Call WriteInt
mov esi, OFFSET byte2
mov ecx, SIZEOF byte2
 Call PFCheck
 call writeInt
exit
main ENDP
PFCheck PROC
; eax PF=1 TRUE PF=0 False
; esi, ecx
    Push esi
    push eax
```

```
Sub ecx, 1
mov al, 0
xor al, 0

mov al, [esi]

L1;

inc esi
xor al, [esi]

mov bl, [esi]
Loop L1

jP LPE1
mov eax, 0
jmp LEND
LPF1:

mov eax, 1
LEND:
pop ecx
pop esi

ret

PF check ENDP
END main
```

32) INCLUDE32.inc

```
.data

    N DWORD 10
    A DWORD 9
    B DWORD 8

.code
main PROC

    mov eax, N
    mov ebx, A
    mov eex, B

TOP:

    Cmp eax ; 0
    jbe Next
    cmp eax, 3
    jne L1
    jmp L4

L1:

    Cmp eax, ebx
```

```
        jb L3
        ja L2

L2:
        cmp eax, ecx
        ja L3
        jb L4

L3:
        sub eax, 2
        jmp Top

L4:
        sub eax, 1
        jmp top

Next:

    Invoke ExitProcess, 0

main endp

end main
```

31) (a) CMP EBX, ECX

JBE L1; if(ebx<=ecx)

CMP EBX, val1

JBE L1

MOV X, 1

JMP L2

L1: MOV X, 2 : else, X=2

L2:

(b) CMP EBX, ECX

JBE L1

CMP EBX, EDX

JBE L1

JMP L3; both true, go to L2

L1: CMP EDX, EAX

JBE L3 ; if (edx <= eax), go to L3

L2: MOV X, 1

10) BX = 006Bh

11) BX = 092h

12) BX = 064BBh

13) BX = A857h

14) EBX = BFAFF69Fh

15) RBX = 00000000505098B64h

9) (a)
```
cmp ebx, ecx
jna next
mov x, 1
next:
```

(b)
```
cmp edx, ecx
jnbe else
mov x, 1
jmp next
else: mov x, 2
```

33)
```
INCLUDE Irvine32.inc
N = 10
.data
Array DWORD N DUP(?)
```

```
j DWORD ?
K DWORD ?

. Code

  main PROC
  Call Clrscr
  mov j, -10
  mov K, 10
  mov ESI, OFFSET array
  mov ECX, N
  Call FillingAnArray
  mov j, 100
  mov K, 1000
  mov ESI, OFFSET array
  mov ECX, N
  Call FillingAnArray
  Call waitmsg
```

```
exit

main ENDP

Filling An Array PROC

    Push ecx
    Push esi

l1:

    mov eax, j
    mov ebx, k

    dec ebx

    sub ebx, eax ; create range from
                   0 to N

    xchg, ebx, eax ; randoms works
                     with eax

    call RandomRange ; generate random
                       with range 0 to
                       N.

    neg ebx          ; change sign of ebx
```

```
sub eax, ebx    ; sub from eax to
                  define range

call Crlf
Call waitmsg
mov [esi], eax
add esi, 4
loop L1
  Pop esi
  Pop ecx
  ret
FillingAnArray ENDP

END main
```

20) JA) JNBE (Jump if above / jump if not below or equal)

Condition for jump:-

CF=0 and ZF=0

The JA / JNBE conditional jump instruction with cause program execution to transfer to another location in a rang from +127 bytes to -128 bytes from the instruction following the Jump instruction CF=0 and if ZF=0. (both must be 0). if this ~~ZF=0~~ condition is not true no jump occurs. when used after cmp. this instruction is refering to unsigned values of the operands used by the CMP instruction. DEBUG Notes:

Regardless of which mnemonic is used during assembly, DEBUG always [Flag effected- none.].

21) The final value of Edx =

100000000 , 00000000 , 01111111

11111111 . 00000000 00000000

100000000 00000000 = 25614.

23) The final value of Edx =

00000000 00000000 01111111

11111111 1111 1111 1111 1111

10000000 00000000

Eax = 016

22) operations on eax, do not directly affect the value of edx but since it has been initialized to 1 and the zeroing depend on the outcome of an operation on eax. It is affected indirectly.

jb is an unsigned operation and does what you said not that ffffh is below 8000h so the jump will be taken, thereby skipping the mov edx, 0 thus the final value in edx will be (1).