NAME SAEEDA NAZ

ID 14332

Semester 5th semester

Deparment BS (CS)

ASSIGNMENT

NO

(6)

ASSIGNMENT No 6

Question No 1.

1) Write a single instruction using 16-bit operands that clears the high 8-bit of AX and does not change the low 8 bits.

Ans :- AND AX , 00FFh

Question No 2.

Ans :- OR ax , 0FF0Bh

Question No 3

Ans :- Xor eax , 0FFFFFFFFh

Question No 4

Ans :- test eax , 1 ; (Low bit set if eax is odd

Question No 5

Ans :- or al , 01000000b

Question No 10

Ans :- BX = 0063h

Question No 11

Ans :- BX = 092h

Question No 12

Ans :- BX = 064BBh

Question No 13

Ans :- BX = A857h

Question No 14

Ans :- EBX = BFAFF69Fh

Question No 15

Ans :- RBX = 0000000050509B64h

X Question No 16

Ans :- AL = 2Dh + 48h, 6Fh, A3h X

Question No 17

Ans :- AL = 85h, 34h, BFh, AEh

Question No 18

Ans :-
a. CF = 0 ZF = 0 SF = 0.
b. CF = 0 ZF = 0 SF = 0
c. CF = 1 ZF = 0 SF = 1

Question No 19

Ans :- JECX

Question No 6

Ans :- JA, JNBE, JAE, JNB, JB, JNAE, JBE, JNA.

Question No 7

Ans :- JG, JNLE, JGE, JNL, JL, JNGE, JLE, JNG.

Question No 8

Ans :- No will the following code jump to the label named Target.
or No, berause the jg is used with signed value and (8109h is negative, and 36h is positive).

Question No 22

Ans :- Operating 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 on operation on eax. it is affected indirectly.

jb is an unsigned operation and does what you said. Note that 7ffh is below 8000h so the jump will be taken, thereby skipping the mov edx, 0. Thus, the final value in edx will be (1).

Question No 24

Ans :- Yes : will the following code jump to the label named Target?

```
mov ax, +30
cmp ax, -50
jg Target
```

Question No 25

Ans : Yes : will the following code jump to the label named Target.

```
mov ax, -43
cmp ax, 26
ja Target.
```

Yes (the unsigned represented of -43 is compared to 26)

Question No 28

Ans :-
```
cmp dx, cx
jbe L1
```

Question No 26

Ans :-
```
and al, 00001111b
```

Question No 29

Ans :-
```
cmp ax, cx
jg L2.
```

Question No 30

Ans :-
```
and al, 11111100b
jz L3
jmp L4
```

Question No 27

Ans :- Use the formula presented earlier :

B_0 XOR B_1 XOR B_2 XOR B_3.

```
• 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
```

Question No 21

Ans :- The final value of EdX =

10000000 , 00000000 ~~1000000~~ 01111111
11111111 00000000 00000000 10000000
00000000 = 256 14.

Question No 23

Ans :- The final value of EdX =
00000000 00000000 01111111 11111111
1111 1111 1111 1111 10000000 00000000

Eax = 016.

Question No 9

Ans :- a) if ebx > ecx
x = 1

```
cmp  ebx , ecx
jna  next
mov  x, 1
next :
```

b) if edx <= ecx
x = 1
else
x = 2

```
comp  edx , ecx
jnbe  else
mov   x, 1
jmp   next
else  : move x, 2
next :
```

Question No 31

a) if (val1 > ecx) AND (ecx > edx)
x = 1
else
x = 2

```
cmp  val1 , ecx
jna  else
cmp  ecx , edx
jna  else
mov  x , 1
jmp  next
else : mov  x, 2
next :
```

b) if (ebx > ecx) OR (ebx > val1)

```
X = 1
else
X = 2
```

```
cmp  ebx , ecx
ja   L1
cmp  ebx, val1
ja   L1
mov  x, 2
jmp  next
L1 : mov  x, 1
next :
```

c) if (ebx > ecx AND ebx > edx) OR (edx > eax)

```
X = 1
else
X = 2
```

```
cmp  ebx , ecx     ; ebx > ecx ?
jna  L1            ; no : try condition after
                     OR
cmp ebx , edx      ; yes : is ebx > edx ?
jna  L1            ; no : try condition after OR
jmp  L2            ; yes : set x to 1

; ..... OR (edx > eax) ----------

L1 : cmp edx , eax     ; edx > eax ?
jna else               ; no : set x to 2
L2 : mov x, 1          ; yes : set x to 1.
jmp next               ; and quit
else : mov x, 2        ; set x to 2
next :
```

Question No 33

Ans Solution :-

```
INCLUDE Irvine32.inc

N = 10
.data
array DWORD N DUP( ?:)
j     DWORD ?
ic    DWORD ?

• code
  main PROC
  call Clrscr
  mov j, -10
```

```
mov k, 10
mov ESI, OFFSET array
mov ECX, N
call Filling An Array

mov j, 100
mov k, 1000
mov ESI, OFFSET array
mov ECX, N
call Filling An Array

call waiting Msg
exit
main ENDP

Filling An Array PROC
push ecx
push esi

h:
   mov eax, j
   mov ebx, k

   dec ebx
   sub ebx, eax      ; create range from 0 to N
   xchg ebx, eax     ; randam work with eax

   call RandomRange  ; generate random int within
                     rang 0 to N

   neg ebx           ; change sign of ebx
   sub eax, ebx      ; sub from eax to
                     define range
```

```
call crlF
call Write Int

mov [esi], eax
add esi, 4

loop h

pop esi
pop ecx
ret
filling ArrayEndP

END main
```

X Question No. 32

Ans :- Solution :-

```
N mov ebx
A mov ecx
B Top cmp eax
O jbe Next cmp eax
3 jne L1 jmp L4 L1 cmp eax, ebx.
```

Question No 30

Ans :- 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 will cause program execution to transfer to another location in a rang from +127 bytes to -128 bytes from the instruction following the jump instruction if CF=0 and if ZF=0 (both must be 0). if this condition is not true no jump occurs. when used after CMP. this instruction is refering to the unsigned values of the operands used by the CMP instruction. DEBUG Notes: Regardless of which mnemonic is used during assembly, DEBUG always disassembles this op code as JA. [Flag affected - none].

Question No 16

Ans :- a = 00101101
b = 01001000
c = 01101111
d = 10100011

Question No 34

Program :-

```
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 Summing Array Elements In Range
call WriteInt
call Crlf

mov j,-10
mov k,0
mov ESI,OFFSET array
mov ECX,N
call Summing Array Element In Range
call WriteInt
call CrlF

call WaitMsg
exit
```

```
main ENDP

Summing Array Elements In Range PROC

push ecx
push esi
mov eax ,0

h:
  mov ebx, [esi]
  cmp ebx , j
  jge true1
  jmp next

  true 1 :

      cmp ebx ,k
      jle true2
      jmp next

  true 2 :
      add eax , ebx

  next :
      add esi, 4

  loop h

    pop esi
    pop ecx
    ret
  Summing Array Element In Range ENDP

End main.
```

Question No 35:

Ans :- Programming :-

1)

```
.386.
INCLUDE Irvine32.inc:

.data
byte1 BYTE 11111110b, 11011110b,
      10001110b, 11011100b, 11001100b,
11001010b, 11001010b, 11001010b;

byte2 BYTE 11111110b, 11011110b, 10001110b,
      11011100b, 11001100b, 11001010b,
11001010b , 11001010b ,


.code
main PROC
    mov esi , OFFSET byte1
    mov ecx, SIZEOF bytes1
    call PFcheck
    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 FAISE
; esi , ecx
        push esi
        push ecx
        sub ecx, 1
        mov al, 0
        xor al, 0
        mov al, [esi]

L1:
    inc esi
    xor al, [esi]
    mov bl, [esi]
    loop L1

    jp LPF1
    mov eax, 0
    jmp LEND
    LPF1:
    mov eax, 1
    LEND:
    pop esi

    ret

PFcheck ENDP

END main.
```

Question No 32

Ans

```
INCLUDE 32.inc

.data

    N DWORD 10
    A DWORD 9
    B DWORD 8

.code
main PROC

    mov eax, N
    mov ebx, A
    mov ecx, 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 Exit process , 0

        main endp
        end  main
```

(completed)