NAME RaFi ULLAH

ID 14283

Department · Bs(cs)

5th semester

SUBJECT ASSEMBLY LANGUAGE

Assighment No:

4

Page # 1

i) QNo: 1

Ans: inc val 2

2) QNo: 2

Ans Sub eax. val 3

3) QNo: 3

Ans: mov ax. val 4

Sub val 2, ax

4) QNo: 4

ANs: CF = 0, SE = 1

5) QNo: 5

Ans OF = 1, SE = 1

6) QNo: 6

ANs: movax, 7FF0h

add al, 10h; a: CF = 1 SE = 0 2F = 1 OF = 0

add ah, 1; b: CF = 0 SE = 1 2F = 0 OF = 0

add ax, 2; C: CF = 0 SE = 0 2F = 0 OF = 0

7) QNo: 7 Ans:

Ans. mov esi. OFFSET my Bytes

mov al, [esi] ; a. AL = 10h

mov al, [esi+3] ; b. AL = 40h

mov esi, OFFSET my word+2 ; C. Ax = 003Bh

mov edi, 8

```
mov edx,[myDoubles+edi] ; d. EDX = 3
mov edx, myDoubles[edi] ; e. EDX = 3

mov eax, [ebx+4]  ; F. EAX = 2
```

Q No # 19

Ans:
```
mov al, 80h
add al, 80h
```

Q No # 20:

Ans:
```
mov al, 0FFh
inc al
jz INC_overflow
mov bl, 1:2
dec bl
jz DEC_overflow
INC_overflow;
DEC_overflow.
```

Q No: # 21

Ans: mov eax, TYPE myBytes; a. 1

mov eax, LENGTHOF myBytes; b. 4

mov eax, SIZEOF myBytes; c. 4

mov eax, TYPE myword; d. 2

mov eax, LENGTHOF myword; e. 4

mov eax, SIZEOF myword; f. 8

mov eax, SIZEOF myString; g. 5

Q No: # 22

Ans: mov dx, WORD PTR myBytes

Q No # 23

Ans mov al, BYTE PTR myWords + 1

Q No # 24:

Ans: mov eax, DWORD PTR myBytes.

Q No: 8

ANS:
```
mov esi, OFFSET myBytes
mov ax, [esi]      ; a. Ax = 2010h
mov eax, DWORD PTR myWords; b. EAX =
                                  003B008Ah
xmov ax, [esi]x
x mov esi, MyWoR x

mav esi, my pointer    ;
mov ax, [esi+2]        ; c. Ax=0000
mov ax, [esi+6]        ; d. Ax=0000
mov ax, [esi-4]        ; e. Ax=0044h
```

Q No # 9:

Ans:

The programm does not stop, because the first loop instuction deerement ECX to zero. The second loop instruction deerements ECX to FFFFFFFFh, causing the outer loop to repeat.

Question #No 10:

Ans:

```
.DATA
    count DWORD ?
.CODE
mov eax,0
mov ecx, 10; outer loop counter
 L1:
mov count, ecx
 mov eax, 3
 mov ecx, 5 inner loop counter.
L2:
add eax.5
Loop L2; repeat inner loop
mov ecx, count
Loop L1; repeat outer loop.
```

QNo: # 11

Ans:

```
mov ax, word ptr three
mov bx, word ptr three+2
 mov three, bx
mov word ptr three+2.ax
```

Page # 6

QNo: # 12:

ANS: Xchg A,B
Xchg A,C
Xchg A,D

QNo # 13

ANS: * Parity flag (PF) will be set if there is an even number of 1 bits in the message byte.

* parity flag (PF) will be Zero for the message byte having an odd number of 1 bits.

• Code

```
mov al, 01110101b
add al, 00000000; AL=01110101, PF=0
```

After the executing of the ADD instruction Al Contain the vaclue of the message byte since, there are ~~three~~ five(5) odd number of ones in the AL register. Thus, PF=0

Page # 7:

Q No # 14:

ANS: Any non-zero operand causes the carry flag to be set.

Examples:

```
.data
valB  BYTE 1,0
valC  SBYTE -128
.code
neg valB          ; CF=1, OF=0
neg [valB+1]      ; CF=0, OF=0
neg valC          ; CF=1; OF=1
```

Q No # 15:

ANS:

```
mov al, 0FFh
add al, 1
```

QUEStion No: 16:

ANS: mov al, OFFh

add al, 1; CF = 1, AL = 00

; Try to go below zero:

mov al, 0

sub al, 1; CF = 1 AL = FF

QNo: # 17

ANS: Solution:

```
INCLUDE Irvine32.inc
.data
Val1 SDWORD 8
Val2 SDWORD -15
Val3 SDWORD 20

Final SDWORD ?
.CODE
main PROC
mov eax Val2
neg eax; eax = -15
add eax, 7; _Val2 + 7
mov ebx, Val3
```

```
add ebx, val 1; val 3 + val 1
Sub eax, ebx
mov finalval, eax
call DumpRegs; display the register
exixt
main ENDP
END main.
```

QNo # 18:

ANS:

```
.data
intarray DWORD 10000h, 20000h,
30000h, 40000h
.code
main proc
mov edi, OFFSET intarray
mov ecx, LENGTHOF intarray
mov eax, 0
L1
add eax, [edi]
add edi, TYPE intarray
```

PAGE # 10

```
mov  eax, 0
  L1
add  eax, [edi]
add  edi. TYPE intarray
  Loop L1
invoke Exixt process, 0
main endp
end main
```

Q No # 19

ANS: mov al, 80h

```
    add al, 80h
```

QUESTION # No: 25

ANS:

```
Mywords LABEL DWORD
mywords WORD 3 DUP ( ? ), 2000h
```

- data

```
mov eax mywords D
```

QUESTION # 26

ANS:

- data

```
myByte BYTE 10h, 20h, 30h, 40h
mywords WORD 3 DUP ( ? ). 200h
mywords DLABEL DWORD
mywords WORD 3 DUP ( ? ), 2000h
```

- code

```
mov eax. mywords D
```

QUESTION # 27

ANS: programming Name: big Endian to Little Endian.

```
.386
model Flat, std call
.stack 4096
Exixt process pROTo, dwExixt:DWORD
.data
big Endian BYTE 12h, 34h, 56h, 78h
little Endian DWORD?
.code
main pROC
mov al, [big Endian+3]
mov BYTE PTR [little Endian], al
mov al, [big Endian+2]
mov BYTE pTR [little Endian+1], al
mov BYTE. pTR [little Endian+2], al
mov al, [big Endian]
mov BYTE PTR [little Endian+3], al
INVOKE Exixtprocess, 0
main ENDP
END main
```

QUESTION No # 28

ANS:

```
.386
model flat, stdcall
.stack 4096
ExitProcess PROTO, dwExitCode:DWORD
.data
array WORD 0,2,5,9,10
newArray DWORD LENGTHOF array DUP(?)
.code
main PROC
mov ecx, LENGTHOF array
mov ESI, OFFSET array
mov EDI, OFFSET newarray
L1:
mov EAX,0
Mov Ax, [ESI]
mov [EDI], EAx
ADD ESI, TYPE array
ADD EDI, TYPE newArray
  Loop L1
INVOKE ExitProcess,0
main ENDP
END main
```

QUESTION # 29

ANS: Solution:

```
• 386
• model flat, stdcall
• stack 4096
ExitProcess PROTO, dwExitCode: DWORD
• data
decimalArray DWORD 1,2,3,4,5,6,7,8
• code
main PROC
MOV ESI, OFFSET decimalArray
MOV EDI, OFFSET decimalArray
mov ecx, LENGTHOF decimalArray-1
L1:
ADD EDI, TYPE decimalArray
  LOOP L1
  mov ecx, LENGTHOF decimalArray
L2:
MOV EAX, [ESI]
MOV EBX, [EDI]
XCHG EAX, EBX
ADD ESI, TYPE decimalArray
SUB EDI, TYPE decimalArray
  LOOP L2
```

```
INVOKE ExitProcess, 0
main ENDP
END main
```

QUESTION # 30

Ans:

```
.386
.model flat, stdcall
.stack 4096
ExitProcess PROTO, dwExitCode:DWORD
.data
source BYTE "This is the source string", 0
target BYTE SIZEOF source DUP
.code
Main proc
mov esi, 0
mov edi, LENGTHOF source - 1
mov ecx, SIZEOF source
L1:
mov eax, 0
mov edi al, source[esi]
mov target[edi], al
inc esi
dec edi
loop L1
```

```
INVOKE ExitProcess,0
main ENDP
ENDP main
```