Grand Assignment

21K-3153

Q1

(i) ~~mov ecx, 0FFF200h~~
~~FADD dword ptr [ecx]~~

(ii) (a) 67 66 89 A4 EA 01

(b) 67 02 00

(c) 0F 85 02 00 00 00

(d) 66 50

(e) 66 A1 0F 00 00 00

(f) ~~66 50~~ 66 2B 0D 00 00 00 0F

(g) 66 42

(iii) (a) AL = 85h

(b) AL = 34h

(c) AL = 0BFH

(d) AL = 0AEH

(iv) in 1st instruction, eax will hold address of var1
in 2nd, eax will hold the value of var1

(v) (1) Pass parameter to procedures
(2) store return addresses of procedures during
(3) preserve register & memory values of procedures
(4) create and store local variables

(vi) EAX: 9ABCH EBX = 9ABCH
ECX: 567F H ESP = 0FCH

(vii) additional:

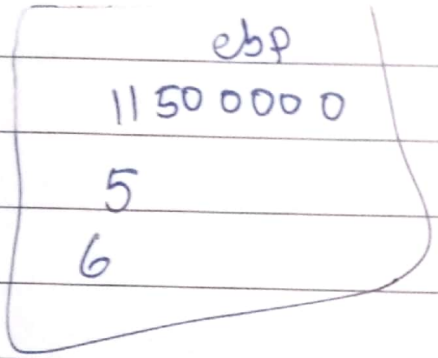
```
PUSH ESI
PUSH ECX
mov ecx,10
L1:
    add eax,[esi]
    sub esi,4
    dec ecx
    Jnz L1
    POP ecx
    POP esi
    ret
```

(viii)

(ix)

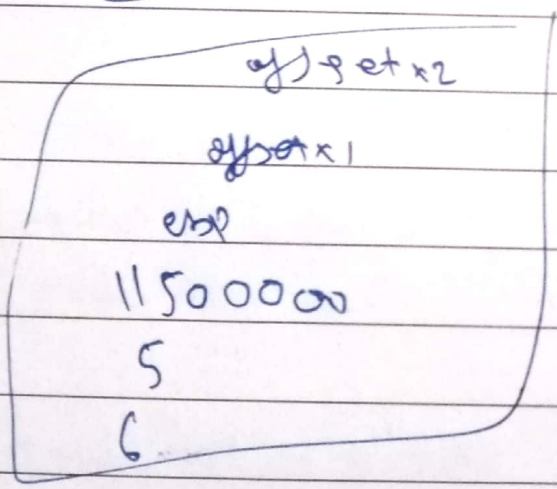
	AH = 1	AH = 2	AH = 3
v1 = v2	→		✓
v1 < v2	✓	✓	
v1 > v2	✓		

(x)

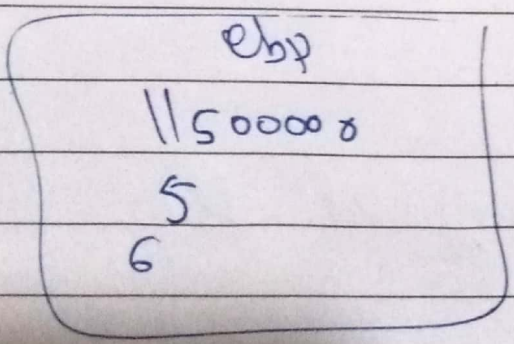


mov esi, eax = ESP = 1FF8

mov ebp, esp = EBP = 1FE2

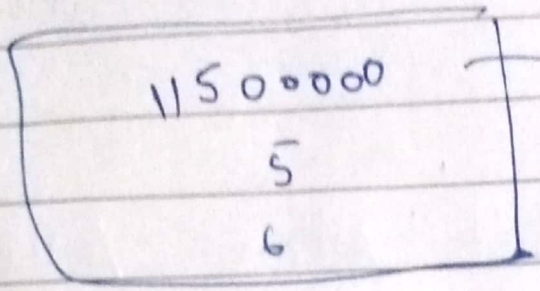


PUSH OFFSET x2 =
ESP = 1FD4



x2 = 32h

x1 = 30h



EP = 11500000 h

(Q4)

(i) eax = 2h

(ii) ~~eax, length of base y~~ eax = 9h

(iii) eax = 30h

(iv) ~~eax~~ eax = 40400 Ch

(v) eax = 0A3020104

(vi) AL = 0ABh

(vii) No because it is not a logical or arithmetic operation

(viii) If the ~~largest~~ MSB is negated (like in -1y)

10000000 ,

↓
MSB

this negation will set off OF flag.

(ix) invalid as 32 bit value cannot be stored in an 8 bit register

(x) AL = 38h

(xi) EAX = 00006534h

(xii) ebx = 00030000h

(xiii) invalid as INC operator needs a size for eg BYTE PTR

(xiv) memory cannot be moved to memory

(xv) edx = 393f3736h

(xvi) memory cannot be moved to memory

(xvii) L is 16 bit, esi is 32 bit, sizes should match

(xviii) AX = fff6h

(xix) C
D
E
A
B

(xx) (a) AL = 6A h
(b) AL: 6A h

(xxi)
(i) Cf=0 Of=0 Sf=1 Zf=0 AX=0F1EB h

(xxii) Cf=1 Of=0 Sf=0 Zf=0 CX=1600 h

xxiii Cf=1 Of=0 Sf=1 Zf=0 BX=0FFE h

(xx) EAX = ???? FFFF h

(xxi) AND eax, 0fffff0f0 h

(xxii) CF 1 After cmp, Cf would be 0 so JB would not execute.

0 The OF flag would be 1 so JL would execute

(Q3)

```
.data
OP1 dword 10
OP2 dword 15
OP3 dword 20
X dword 5
Y dword 20

.code
main PROC
while:
    mov eax, op1
    cmp eax, op2
    JGE end
    inc op1
    mov eax, Y
    mov X, eax
    add x, 2
    mov eax, OP3
    cmp eax, op2
    JE while
    Add X, 8
    JMP while
end:
    exit
main endp
end main
```

(b)

```
.data
val1    dword 4
val2    dword 2
val3    dword 2
x       dword ?

.code
main PROC
    mov eax, val1
    cmp eax, val2
    JLE else
    mov eax, val2
    cmp eax, val3
    JLE else
    mov x, 10
    JMP exit
else:
    mov x, 20
exit:
    exit
main endp
end main
```

(iv) .data

x dword ?
y dword ?
z dword ?

Minimum PROC

```
    cmp  eax, ebx
    JLE  L1
    xchg eax, ebx
L1:
    cmp  eax, ebx
    JLE  L2
    xchg eax, ebx
L2:
    cmp  eax, ebx
    JLE  L3
    xchg eax, ebx
L3:
    ret
Minimum endp
```

(iii)

(a)

```
CopyTable PROC uses esi edi ecx
mov esi, offset table1
mov edi, offset table2
movecx, lengthof Table1
cld
rep movsb
ret
CopyTable endp
```

(b)

```
mov ecx, num
mov esi, offset table
mov eax, size of table
call search
```

```
Search PROC uses ecx esi eax
    push ebp
    mov ebp, esp
    mov ecx, [ebp + 16]
    mov esi, [ebp + 20]
    mov eax, [esp + 24]
    mov edi, -1
L1:
    inc edi
    cmpal, [esi + edi]
    LOOPNZ L1
    JZ L2
    inc edi
L2: pop ebp
    ret 12
Search endp
```

(a) (c)

```
BCoff PROC
n: dword, k: dword
cmp k,0
JE baseCase
mov ecx, n
cmp ecx, k
JE baseCase
dec n
INVOKE BCoff, n, k
DEC k
INVOKE BCoff, n, k
ret
BASECASE:
INC eax
ret
BCoff endp
```

```
main PROC
INVOKE BCoff, 5,2
call readint
mov n, eax
call readint
mov k, eax
INVOKE bcoff, n,k
call writedec
exit
```

(b)

```
Power PROC x: dword, n: dword
cmp n,0
JE baseCase
dec n
INVOKE power, x, n
mul x
baseCase:
ret
```

```
main PROC
INVOKE power, 5,2
call writedec
exit
main endp
```

(v) ~~fibonacci~~ pro

```
fib proc,
  n: dword
  cmp n,0
  Jmp basecase
  DEC n
  Invoke fib,n
  dec n
  Invoke fib,n
  ret

basecase:
        add eax,n
        ret

fib endp

main PROC
  call readint
  mov ebx,eax   mov eax,0
    Invoke fib,0
      call writedec
      exit
main endp
```

(vi) ~~exchange sort uses~~ .data array dword . . . .

```
main PROC
    mov ecx, length array -1
L1:
    mov esi, 0
    mov edx, length of array -1
L2:
    cmp edx, 0
    JE L5
    mov eax, array[esi]
    add esi, TYPE array
    cmp eax, array[esi]
    JB L3
    mov array[esi], eax
    sub esi, TYPE array
    mov array[esi], ebx
    dec edx
    add esi, type array
    JMP L2
L3:
    dec edx
    JMP L2
L5:
    LOOP L1
    mov esi, offset array
    mov ecx, length of array
    mov ebx, type array
    call dumpmem
    exit
main endp
END main
```

(vii) .data dword

array 60, 4, 17, 45, 17

i dword ?

.code

mov ecx, length of array

sub ecx, 1

mov edi, 0

L1:

mov ebx, ecx

mov esi, 0

mov ecx, length of array

e sub ecx, edi

mov eax, 0

L2:

cmp al, array[esi]

jnc L3

mov i, esi

mov al, array[esi]

L3:

inc esi

LOOP L1

call swap

inc edi

mov ecx, ebx

Loop L

mov esi, offset array

mov ecx, length of array

call dumpmem

exit

main endp

Date: M T W T F S S

Part 3

(Q4)

(i)

```
.data
seq_num     word ?
Rev count   byte 0
status      byte 0
Sens data   WORD 0

.code
main PROC
  mov seq_num, ax
  and seq num, 0000 1111 1111 1111 b
  shr eax, 12
  mov rev count, al
  and rev count, 00000111 b
  shr eax, 3
  mov status, al
  and status, 00000001 b
  SHR eax, 1
    mov sens data, ax
  exit
main endp
end main
```

(ii) .code

```
main PROC
    mov eax, x
    mov cx, ax
    shl ax, 4
    mov bx, cx
    shl bx, 2
    add ax, bx
    shl bx, 1
    add ax, bx
    add ax, cx
    call write dec
main endp
```

(iii) Save in dl

→