SAEED AHMAD ASAD

#14273

MICROPROCESSOR AND ASSEMBLY LANGUAGE

SIR AMIN

Q 5.

Answers

AL : 40h

AX : A200 38h

EDX : 3

Q1.

Answers:

(a) CX = 009Bh

(b) CX = 009Bh

(c) val2 = 1000h

(d) AL = 20h

(e) AX = 200h

(f) EAX = 20000h

Q2.

(a) CX = 0, ZF = 1

(b) CX = -1, SF = 1

(c) AL = 00, CF = 1

(d) AL = FF, CF = 1

(e) AL = 80h, OF = 1

(f) CF = 0, OF = 1

Q3.

```
mov eax, TYPE myBytes ;  a . 1
mov eax, LENGTH OF myBytes;  b . 4
mov eax, SIZEOF myBytes;  c . 4
mov eax, TYPE myBytes ;  d . 2
mov eax, LENGTH OF myBytes; e . 4
mov eax, SIZEOF myBytes; f . 8
mov eax, SIZEOF myBytes;  g . 5
```

Q6.

a)

```
mov al, 'a'  ; al = 01100001b
and al, 11011111b ; al = 01000001b
```

b)

```
mov al, 6   ;   al = 00000110b
or al, 01100000b ;  al = 01100110b
```

c)

```
and al, 0000b ; clear unwanted bits
cmp al, 00001011b ; check remaining bits
  je C1 ; all set? jump to L1
```

Q4.

```
.data
Val32   LABEL   DWORD
valB    BYTE    78h, 56h, 34h, 12h
Val8    LABEL   BYTE
varD    DWORD   12345678h
.code
    mov bl, BYTE PTR varD ; (a) BL=78h
    mov eax, DWORD PTR varB ; (b) EAX=
                                78563412h
    mov al, valB ; (c) AL=78h
    mov eax, val32 ; (d) EAX=12345678h
```

Q7.

(a)

```
    cmp   val1, eax
    jna   else
    cmp   eax, edx
    jna   else
    mov   x, 1
    jmp   next
    else ; mov x, 2
    next
```

Q7

(b).

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

Q7

(c).

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

Q 8.

(a).

Statements that use only PUSH and POP instructions to exchange the value in the EAX and EBX registers are:

PUSH EBX

(It pushes the contents of EBX register into the top of the stack)

PUSH EAX

(It pushes the contents of EAX register into the top of the stack now. the contents of EAX register were are on top of the stack)

POP EBX

(POP -the last thing pushed on the stack to EBX. Here the contents of EAX register were last pushed on the stack so the contents EAX are popped in EBX)

POP EAX

(POP the last thing pushed on the stack to EAX. Here the contents of EBX registers are only left in the stack so the contents of EBX are popped in EAX).

```
push ebx
push eax
pop ebx
pop eax
```

Q8.

(b)

```
INCLUDE Irvine32.inc
.data
    source BYTE "This is the source string", 0
    target BYTE SIZEOF source DUP ('#')
.cod
main PROC
    mov esi OFFSET source ; offset of variable
    mov ebx, 1            ; byte format
    mov ecx SIZEOF source ; counter
```

```
    mov edx offset target ; counter
    call DumpMem ; require the initial
                   values of esi, ebx, etxi:
L1
    mov al, [esi]
    mov [edx+ecx], al
    inc esi
    loop L1
    mov esi OFFSET target; offset of variable
    mov ebx 1
    mov ecx SIZEOF target ; counter
    call DumpMem ; require the initial
                   values of esi, ebx, ecx
main ENDP
END main
```

Q 8.

(c)

```
INCLUDE Irvine32.inc
.data
 loopcount DWORD?
 foreground DWORD?
 background DWORD?
```

```
.cod
main PROC
        main PROC
        mov ecx, 16 ; outer loop count
L1:
    mov loopcount, ecx
    mov foreground, ecx
    dec foreground ; foreground initial
          value = 15 decrements by 1 each
            time loop repeats
    mov ecx, 16 ; Inner loop count
L2:
    mov background, ecx
    dec background ; Background initial
          value = 15 descrements by 1
            each time loop repeats
    mov eax background ; set EAX = background
    shi eax, 4 ; shift left, equivatent to
                multiplying EAX by 16
    add eax, foreground ; Add
                foreground to EAX
    call settextcolor
    mov al, "A" ; set AL to character to
                be written to screen
    call writechar ; Write the character
                in AL to screen
```

```
Loop L2
  call crlf ; Advances cursor to
  beginning of next line
  mov ecx, loopcount
Loop L1
  exit
  main ENDP
  END main
```