81 0D C0 64 93 AB

0110

# National University of Computer and Emerging Sciences

FAST School of Computing Spring-2023 Islamabad Campus

**Question 2 [6+2+4 =12 Marks]**

A. Consider the following array:
**wordArray WORD 810Dh, 0C064h,93ABh**

**Note:** Write the updated **values** of wordArray in the respective block, not the code.

a) Shift three memory words to the left by 1 bit position.

81 0D
1000 0D01 0000 1101
CF=1 0000 0010 0001 1010 ← ans

C0 64
1100 0000 0110 0100
CF=1 1000 0000 1100 1000 ← ans

93 AB
1001 0011 1010 1011
CF=1 0010 0111 0101 0110 ← ans

b) Logical Shift Right three memory words by 1 bit position.

81 0D
1000 0001 0000 1101
0100 0000 1000 0110 CF=1
↑ ans

C0 64
1100 0000 0110 0100
0110 0000 0011 0010 CF=0
↑ ans

93 AB
1001 0011 1010 1011
0100 1001 1101 0101 CF=1
↑ ans.

B. Suppose the instruction set contained **no rotate** instructions. Show how we might use **SHR and a conditional jump** instruction to rotate the contents of the AL register one position to the **right.**

```
.data                    .code
arr BYTE 8 dup(0)        mov CX, 8
                         mov si, offset arr
                    L1:  shr al, 1
                         mov [si],0
                         JNC next
                         mov [si], 1
                    next: inc si
                         Loop L1
```

C. **To multiply eax*29, how many shift left operations are required. Write how the multiplier (29) is factored?**

16+8+4+1

$\Rightarrow EAX * (16+8+4+1)$

$= (EAX * 2^4) + (EAX * 2^3) + (EAX * 2^2) + (EAX * 2^0)$

Shifts required = 4 + 3 + 2 + 0 = 9

14

```
MOV EBX, EAX
MOV ECX, EAX
MOV EDX, EAX
SHL EAX, 4
SHL EBX, 3
SHL ECX, 2
ADD EAX, EBX
ADD EAX, ECX
ADD EAX, EDX
```

# National University of Computer and Emerging Sciences

FAST School of Computing | Spring-2023 | Islamabad Campus

## Question 1 [10 Marks]

**Update Flag register value after execution of the CMP statement. Mark ✓ in taken or not taken box for each jump instruction respectively.**

| Instructions | Taken | Not Taken |
|---|---|---|
| Mov al, 72 | | |
| Cmp al,-15 | | |
| Ja L1 | | ✓ |
| L1: jg L2 | ✓ | |
| L2: jnb L3 | | ✓ |
| L3: jnge L4 | | ✓ |
| L4: jbe L5 | ✓ | |
| L5: jng L7 | | ✓ |
| L6:jmp end | ✓ | |
| L7: | | |
| End: | | |

| Flags | | |
|---|---|---|
| | Sign | 0 |
| | Zero | 0 |
| | Carry | 1 |
| | Overflow | 0 |
| | Parity | 0 |
| | Auxiliary | 0 |

10

Calculations:

FF
0048
− FFF1
0057

0101 0111

| | | |
|---|---|---|
| JG/JNLE | ZF = 0 and SF = OF | Greater / Not Less nor Equal |
| JGE/JNL | SF = OF | Grater or Equal / Not Less |
| JL / JNGE | SF <> OF | Less / Not Greater nor Equal |
| JLE/JNG | (ZF = 1) or (SF <> OF) | Less or equal / not greater |
| JA/JNBE | (CF and ZF) = 0 | Above / Not Below or Equal |
| JAE/JNB | CF = 0 | Above or Equal / Not Below |
| JB/JNAE/JC | CF = 1 | Below / Not Above or Equal / Carry |
| JBE/JNA | (CF or ZF) = 1 | Below or Equal / Not Above |
| JE/JZ | ZF = 1 | Equal / Zero |

A 10
B 11
C 12
D 13
E 14
F 15

## Question 2 [6+10 Marks]

**(a) Update flags(carry, overflow, auxiliary, parity and zero flag) after following operations:**

1) 75BDh + FEFEh **[3 Marks]**

```
  1 1 1
  7 5 B D
+ F E F E
-------
  7 4 B B
```

1011 1011

3

CF = 1
OF = 0
AF = 1
PF = 1
ZF = 0
SF = 0

16 | 213 | 13
−16
5

16 | 27 | 1
−16
11

16 | 28 | 1
−16
4

2) 12B0h - 8347h [3 Marks]

16 → 16 16 A 16
0 12B0
-8347
8F69

0110 1001

CF = 1
OF = 1
AF = 1
PF = 1
ZF = 0
SF = 1

(b) You may use jumps for the conversion of **IF ELSE** statement and **WHILE** loop. Write proper data segment for initializing variables used in code and code segment for writing code. **[10 marks]**

*Note*: all variables are 16-bit unsigned integers.

```
Int array[ ]={10,60,20,33,72};
Int index=0;
Int sample=40;
var1=10;
array_size=sizeof array/sizeof sample;
while(index<array_size){
        if(array[index]<sample)
            {
                var1= var1+1;
                index++; }
        else
            {
                Index--;
            }

}
```

- Model Small
- Stack 100h

- data
array dw 10,60,20,33,72
index dw 0
sample dw 40
var1 dw 10

- code

cmp index, arraysize
jae exit
startloop:
cmp [array+index], sample
jae elsecond
add var1, 1
add index, 1
jmp startloop

jo else end
sub index, 1

```
mov bx, SIZEOF array
startloop: cmp index, bx
jae exit
mov ax, WORD PTR (array + index)
cmp ax, sample
.....
jae elsecond
add var1, 1
add index, 1
jmp startloop

elsecond:
sub index, 1
jmp startloop

exit:
mov ah, 4ch
int 21h
```

**Question 3 [20+6+4=30 Marks]**

**(a) Consider the following data declaration and fill instruction after every instruction. Assume all registers have zero value at the start of execution of code. Also fill memory of these data declaration [20 marks]**

```
L1 LABEL BYTE
L2 LABEL WORD
L3 DD 66ab77efh,2,3,4,5
L4 LABEL WORD
Byte1 SBYTE -4,-2,3,1
L6 db sizeof L3 DUP(type Byte1 DUP(1))
```

| | |
|---|---|
| Mov AL , L1 | AL= ef |
| Mov AX , L4 | AX= FE FC |
| Mov AX,(L4+2) | AH= 01 03 |
| Mov BL , SIZEOF L3 | BL= 20 (in hex = 14) |
| Mov BH , BYTE PTR (L3 +1) | BH= 77 |
| Mov CL , (L1+2) | CL= ab |
| Mov BX , L2 | BX= 77 ef |
| Mov AX, sizeof L6 | AX= 00 20 (in hex = 00 14) |
| Mov DX, (L2+4) | DX= 00 02 |
| Mov esi, Offset L3<br>Mov ax,[esi-4] | AX= XX XX (garbage) |

| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | ef | 77 | ab | 66 | 02 | 00 | 00 | 00 | 03 | 00 | 00 | 00 | 04 | 00 | 00 | 00 |
| 0010 | 05 | 00 | 00 | 00 | FC | FE | 03 | 01 | 01 | 01 | 01 | 01 | 01 | 01 | 01 | 01 |
| 0020 | 01 | 01 | 01 | 01 | 01 | 01 | 01 | 01 | 01 | 01 | 01 | 01 | | | | |

80

(b) Find the values of SizeOf, LengthOf and Type operators. [6 marks]

| .data | SizeOf | LengthOf | Type |
|---|---|---|---|
| V1 byte 11,22,33,44,55 | 5 | 5 | 1 |
| V2 word 15 Dup(0),5,7 | 34 | 17 | 2 |
| V3 dword 4 Dup(10 Dup(4)) | 160 | 40 | 4 |
| V4 word 1,2,3,4,5,6, 7,8,8 Word 7,8,6,9,8,9 | 18 | 9 | 2 |

(c) Write a code that adds **v1** and **v2** and store result in **sum**. Your code should be x86 architecture.
**[4 marks]**

```
.data
V1 db -5
V2 dw 0FFFCh
Sum dw 0

.code
mov ax, TYPE V1
add ax, V2
mov Sum, ax
```

**Question 4 [4 marks]**

**Update value of registers after each line of code.**

*Bval1 db 034h, 022h*

*Wval2 dw 0ABCDh*

*Dval3 dd 0AB22EF12h*

| | | H | L |
|---|---|---|---|
| *Mov ax, WORD PTR bval1* | Ax | 22 | 34 |
| *Mov al, BYTE PTR wval2* | Ax | 22 | CD |
| *Mov bx, WORD PTR dval3* | Bx | EF | 12 |
| *Mov cx, WORD PTR [dval3+2]* | Cx | AB | 22 |

4