

Question Number 1

(17/04/20 Marks)

What are the questions in the following questions

1. Difference between computer architecture and computer organization

Computer architecture is study of system
in which we study how input data is fed
into the system and output
Computer organization is a study of system
which studies which data is transferable and
how

2. What is overflow in the CPU and how is it detected

Overflow and underflow is overflow because

if we want to add the data then the
memory is over and the data transfer
is not correct

3. What are the advantages of using sign-magnitude representation to represent negative
integers?

The main advantage for signed magnitude
is one computer data to represent the signed
value while operation is done we first convert
the data to its complement and then we operate

4. What are the steps involved in execution of CPU instruction and in which order they are
executed?

Fetch instruction
Decode instruction
Execute instruction
Write back

5. If we are signed overflow during the multiplication, then is there any chance that the result will
not fit in the destination register of CPU? Explain briefly.

When we operate first the result
will not fit in the register because the size
of operand is small
As we know that when we multiply
2 operand the size will increase

- vi. What happens to SP and IP when CALL and RET instructions are executed?

When CALL and RET instructions are executed the pointers SP and IP push the instruction which are called.
and when the RET instruction is executed the pointers call the function as which they are pushed.

- vii. Write a single assembly language instruction (rather than multiple instructions) that moves all the bits of a value stored in AX register.

ROL AX, 4

- viii. Draw ROP - 3 instruction format for the memory referenced instructions. Also define the basic difference between ROP - 3 and ROP - 10 instruction formats.



- ix. What are the major functions of an I/O Module?

The major functions of I/O module
Read
write
send data
receive data



- x. What are the two basic design issues in instruction formats?

1) It is how to process
2) CPU designed for how instructions

Section Number 2

(5+5=10 Marks)

17. Write an Assembly language program (according to syntax of assembly) to calculate the factorial of a number stored in AX.

• Data

```

fact db 1
mult db 1
loop1 db 1
loop2 db 1

```

• Code

```

mov ax, 5
mov cx, ax
mov bx, 1
loop1:
    mov dx, bx
    mul cx
    inc bx
    dec cx
    jnz loop1
    mov fact, dx
    loop2:
        jmp loop2

```

5

18. Divide 2 signed numbers (according to syntax of assembly) in such a way that the probability of divide overflow is minimum.

• Data

```

div1 db 10
div2 db 5

```

• Code

```

mov ax, div1
mov cx, div2
div cx, ax

```

2

Question Number 2

100 Marks

Write the names of five available addressing modes and draw a diagram to illustrate the register indirect addressing.

- (1) Immediate Addressing
- (2) Direct Addressing
- (3) Indirect Addressing
- (4) Register Addressing
- (5) Register Indirect Addressing
- (6) Indexed

Register Indirect Addressing



5

Question Number 4

(10 Marks)

Write an assembly language program to count the number of vowels in a given string.

x86-64

str: DB "aslo"

str2: DB "This is a cat."

code:

mov ~~eax~~ ^{rsi} rsi~~mov~~ ^{rsi} rsi

Found:

CMP ~~AL~~ ^{AL} ~~0~~JNE ~~next~~ ^{next}

Not Found:

JMP ~~next~~ ^{next}

2

Question Number 3

(10 Marks)

Locate the errors in the following codes and correct them by re-writing the code.

Incorrect Code	Errors	Corrected Code
46 <pre> COUNT = 500 COUNT1 EQU 400 COUNT = 1 MOV AL, COUNT COUNT = 10.1 MOV AL, COUNT COUNT1 EQU 100 MOV AL, COUNT1 </pre>	Count = 500 Count = 10	<pre> Count = 500 Count 1 EQU 400 Count = 1 mov al, Count mov al, Count Count 1 EQU 100 mov al, Count1 </pre>
47 <pre> MOV AX, 1000H MOV CX, 100H DIV CX </pre>	mov ax, 1000h mov cx, 100h div cx	<pre> mov ax, 1000h mov cx, 100h div cx </pre>
48 <pre> data var DB 48 code mov al, var mov bl, 45 div bl </pre>	data code mov al, var div bl	<pre> data var DB 48 code mov al, var mov bl, 45 div bl </pre>

Question Number 4

Suppose that the following string has been defined:

(10 Marks)

String DB: "THIS IS A GOOD DAY" 100

Write instructions that will cause each '*' to be replaced with 'I'.

• 100
 mov DB, "THIS IS A GOOD DAY" 100
 lea di, (100-100)
 mov ax, 1
 lea di, 100
 mov DI, offset str
 mov AX, 100 - 100 - 100 - 100 - 100
 cld
 REPNE SCASB
 jz found
 jmp notfound.

4

Found:

mov SIB DB, 100
 mov ax, 100
 mov ax, 100
 mov ax, 100

Change:

set

Question Number 2

(10 Marks)

Write an assembly language code to count the number of 1's in AX register and then set or clear the parity flag accordingly.

• Data:

var: DB 01101111 •
 size var equ 8

• Code:

radix 16, 160

word [0] 00

word [4] 55

word [8] 44

word [12] 33

word [16] 22

word [20] 11

word [24] 00

word [28] 00

word [32] 00

word [36] 00

word [40] 00

word [44] 00

word [48] 00

word [52] 00

word [56] 00

word [60] 00

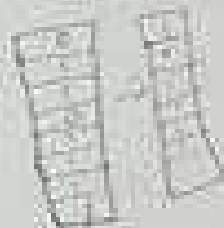
word [64] 00

word [68] 00

word [72] 00

word [76] 00

word [80] 00



2

Q. Number 8

(10 Marks)

Write code by using string primitive instructions to find the number stored in AX from an array of 10 unsigned words. If the required number is found then move that number to BX and using AX and word instruction otherwise search the entire array.

Q.8 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
 Range = $(9 - 1) / 2$
 code

mov 8, offset arr

~~mov 8, 2~~

mov AX, 5

CLD

~~mov CX, 10~~

~~JL~~

~~LODSW~~

~~STOSW~~

REPNE SCASW

JZ Found

JNZ NotFound

Found:

Sub 02, 2

MOV BX, [02]

NotFound:

ret

10

Question Number 2

(10 Marks)

If you have following variable defined in data segment

```
var1 DW 1234
var2 DW 56
var3 DW 23
var4 DW 1234
var5 DW 4567
var6 DW 12
var7 DW 7
```

Then write a program to implement the following arithmetic expression: (10 points)

$$res = var1 + var2 \times var3 - var4 + \left\lfloor \frac{var5}{var6} \right\rfloor \times (var7 \text{ AND } var6)$$

Then check the bit number 10 in the answer. If the bit is set then set the zero flag, otherwise set the parity flag. < 5-1 >

```
code
mov ax, var1
mov bx, var2
add ax, bx
mov bx, var3
cbw
mul ax
mov bx, var4
sub ax, bx
mov bx, var5
mov ax, var6
div bx
cdq
add ax, bx
mov bx, var7
not ax
par
add ax, bx
mov bx, var5
not ax
par
add ax, bx
```

```
MOV var, AX
JZ cond3
JNZ cond2
cond 2
STP
int 20h
cond 2
mov SI, 2
int 20h
ret
```

3

Test file 01/10

Question Number 18

Multiply -20 with -10 by using

(10 Marks)

- Two's complement method
- Booth's algorithm

And then compare the results generated by both the approaches.

-20

-10

Eg. Two's complement method

+20 = 10100

+10 = 1010

-20 By 2's complement

-10 By 2's complement

$$\begin{array}{r} 20 \ 20 \\ 2 \ 10100 \\ \hline 10100 \\ 2 \ 1010 \\ \hline 1010 \\ 2 \ 1010 \\ \hline 1010 \end{array}$$

$$-20 = \overline{01100} + 1 = 10100$$

↑
M

signed two's comp

$$\begin{array}{r} 1100 \\ 0110 \\ \hline 0000 \\ 0000 \\ 1100 \\ 1100 \\ \hline 10100 \\ 01100 \\ \hline 01100 \end{array}$$

01100

also 10100
By 2's complement
1011000