National University of Computer and Emerging Sciences

FAST School of Computing **Fall-2023** **Islamabad Campus**

EE-2003 Computer Organization and Assembly Language

Serial No:

Final Exam

Total Time: 3 Hour
Total Marks:

Tuesday, 7th November 2023.

_____
Signature of Invigilator

Course Instructor

Mr. Farrukh Bashir, Mr. Aqib Rehman, Mr. Taimur Shahzad, Mr. Obaid Ullah, Mr. Shams Farooq.

_____ _____

Student Name Roll No. Section Signature

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Attempt all questions on the question-book. Read the question carefully, understand the question, and then attempt it.
2. No additional sheet will be provided for rough work. Use the provided space for rough work
3. After asked to commence the exam, please verify that you have **14** different printed pages including this title page. There are a total **8** of questions.
4. Calculator sharing is strictly prohibited.
5. Use permanent ink pens only. Any part done using soft pencil will not be marked and cannot be claimed for rechecking.

	Q-1	Q-2	Q-3	Q-4	Q-5	Q-6	Q-7	Q-8	Total
Marks Obtained									
Total Marks	**20**	**15**	**15**	**15**	**20**	**40**	**15**	**20**	**160**

Question 1 [10+10 = 20]

i. Dry run 32-bit assembly code given below, and only reflect changes done to array in given space.

```
.data
  array dword -5, 1, 0, 3, 4, -2
  swap db 0
.code

main Proc
 mov ecx, 24
 sub ecx, 4

start:
      mov swap, 0
      mov ebx, 0
loop1:
      mov eax,[ebx+array]
      cmp eax,[ebx+array+4]
      jle noswap

      mov edx,[ebx+array+4]
      mov [ebx+array+4],eax
      mov [ebx+array],edx
      mov swap, 1
noswap:
      add ebx, 4
      cmp ebx, ecx
      jne loop1
      cmp swap, 1
je start
main ENDP
END main
```

What Program is doing
Bubble sort

Array						swap
-5	1	0	3	4	-2	0
-5	0	1	3	4	-2	1
-5	0	1	3	4	-2	1
-5	0	1	3	-2	4	1
-5	0	1	3	-2	4	0
-5	0	1	-2	3	4	1
-5	0	1	-2	3	4	0
-5	0	-2	1	3	4	1
-5	0	-2	1	3	4	0
-5	-2	0	1	3	4	1
-5	-2	0	1	3	4	0

ii. Complete the missing code given below.

- **a.** Complete the **main** that must pass arrays to a common function to process passes array. Procedure must receive parameters (array **offset** first then array **length**) through stack.
- **b.** Creating local variable(swap) on the stack in **Function**.
- **c.** After **Ret** from procedure main should retain same old values of Registers as before **CALL** and stack should be empty as well.
- **d.** Create local variable on stack for **Swap** (rather than created on memory as shown in **part i**)

Note: Filling stack will help you to complete the program also notice line carefully before return

```
.data
   array1 db 8,-6,3,4
   array2 db 9,7,6,5,4,14

.code
main Proc
   mov eax,1
   mov ebx,2
   mov ecx,3
   mov ebp,4
;function calls para passng

mov ebx, offset array1
push ebx
mov ebx,sizeof array1
push ebx
Call Function

mov ebx, offset array2
push ebx
mov ebx,sizeof array2
push ebx
Call Function

main ENDP
INVOKE ExitProcess,0
```

Address	Content
0100	
00FC	~~OFFSET ARRAY1~~
00F8	~~SIZE ARRAY1~~
00F4	~~CALL ADDRESS~~
00F0	~~EBP~~
00EC	~~LOCAL VARIABLE~~
00E8	~~EAX~~
00E4	~~EBX~~
00E0	~~ECX~~
00CC	~~EDX~~
00C8	~~ESI~~
00C4	
00B0	
00BC	
00B8	
00B4	

```
Function proc
;Create local variable
;push register

push ebp
mov ebp,esp
sub esp,4
push eax
push ebx
push ecx
push edx
push esi

iteration:
;accessing local variables
;initialization

 dec ecx
loop1:
 mov al,[ebx+esi]
 cmp al,[ebx+esi+1]
 jbe noswap

mov dl,[ebx+esi+1]

mov [ebx+esi+1],al
mov [ebx+esi],dl
mov dword ptr[ebp-4],1

noswap:
add esi,1
cmp esi,ecx
jne loop1

cmp dword ptr[ebp-4],1
je iteration

pop esi
pop edx
pop ecx
pop ebx
pop eax
mov esp,ebp
pop ebp
ret 8
Function endp
END main
```

Question 2 [10+5 = 15]

Execute complete code and fill/empty the given **STACK**. Update the progress register after executing each line using comma separator. Stack starts at address **0AB0H** All registers initially have **0000H**. Marks will be awarded on detailed execution of the given code.

```
1  .data
2  .code
3  main Proc
4              mov ebx,5
5              mov ecx,4
6              push 2
7              Call Function
8
9  main ENDP
10 INVOKE ExitProcess,0
11
12 Function proc
13    push ebp
14    mov ebp,esp
15    push ebx
16    push ecx
17    mov eax,[ebp+8]
18    cmp eax,0
19    ja L1
20    mov eax,1
21    jmp L2
22    L1:
23    dec eax
24    push eax
25    inc ecx
26    call Function
27    Return1:
28    mov ebx,[ebp+8]
29    mul ebx
30    L2:
31    pop ecx
32    pop ebx
33    pop ebp
34 ret 4
   Function ENDP
   END main
```

STACK	
Address	**Content**
00AB0H	
00AACH	~~PUSH 2~~
00AA8H	~~RET(ADD 8)~~
00AA4H	~~PUSH EBP=0~~
00AA0H	~~EBX=5~~
00A9CH	~~ECX=4~~
00A98H	~~EAX=1~~
00A94H	~~RET(ADD 24~~
00A90H	~~PUSH-EBP-0AA4~~
00A9CH	~~EBX=5~~
00A88H	~~ECX=5~~
00A84H	~~PUSH EAX=0~~
00A80H	~~RET=25~~
00A7CH	~~PUSH EBP= 0A90~~
00A78H	~~PUSH EBX=5~~
00A74H	~~PUSH ECX=6~~
00A70H	
00A6CH	
00A68H	
00A64H	
00A60H	
00A5CH	
00A58H	
00A54H	
00A50H	

EBP	0,0AA4, 00A90, 0A7C, A90, AA4,
EAX	2,1,1,0,0,1,1,2
EBX	5,1,2
ECX	4,5,6

Question 3 [5+5+5 =15]

Consider the following data declaration, Copy **string1** to **string2** using **STACK** fill given memory after execution of the program.

```
.data
     string1 db "eman srotcurtsni ruoy elcric sunob erocs ot"
     size1=$-string1
     string2 db size1 dup('p')
     last dd 0ABCDH
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
100	‘e’	‘m’	‘a’	‘n’	‘ ‘	‘s’	‘r’	‘o’	‘t’	‘c’	‘u’	‘r’	‘t’	‘s’	‘n’	‘i’
110	‘ ‘	‘r’	‘u	‘o’	‘y’	‘ ‘	e	l	c	r	i	c		s	u	n
120	o	b		e	r	o	c	s		o	t	t	o		s	c
130	o	r	e		b	o	n	u	s		c	i	r	l	e	
140	y	o	u	r		i	n	s	t	r	u	c	t	o	r	s
150		n	a	m	e											

```
.data
      ary1 db "eman srotcurtsni ruoy elcric sunob erocs ot"
       size1=$-ary1
       copyary db size1 dup('p')
       last dd 0ABCDH

.code
main Proc

    mov ecx, lengthof ary1
     mov esi,OFFSET ary1
     mov edi,OFFSET copyary
     L1:
          mov al,[esi]
          push ax
          inc esi
     LOOP L1

     mov ecx, lengthof copyary
     L2:
          pop ax
          mov [edi],al
          inc edi
     LOOP L2
```

Question 4[12 + 3 = 15]

Dry run the given assembly code below? Update memory at every iteration? Write the purpose of the program.

```
.data
      M db 13
      Q db -5
      A db 0
      result dw 0
      Q_l db 0
.code
main Proc
      mov ax,0
      mov cx,8
      mov al,A
      mov bl,M
      mov dl,Q
      clc
l1:
      mov Q_l,dl
      RCL Q_l,1
      And Q_l,3

      cmp Q_l,1
      jne skipadd

      add al,bl
      jmp shift

skipadd:
      cmp Q_l,2
      jne shift
      sub al,bl
shift:
      SAR al,1
      RCR dl,1
LOOP L1
mov byte PTR result, dl
mov byte PTR result+1,al
```

Purpose of Program
Unsigned Multiplication

M (13) =bl	0000 1101	-M (-13)	1111 0011

A =al	Q = dl	Q_1	Carry	Steps
00000000	1111 1011	0000 0000	0	Initialization Cx=8
1111 0011 1111 1001	1111 1011 1111 1101	1111 1011 1111 0110 10	 1	Q_1= dl RCL dl AND dl,3 A-M SAR, cx=7
   1111 1100	 1111 1110	1111 1101 1111 1011 11	 1	Q_1= dl RCL dl AND dl,3 SAR, cx=6
1111 1100 +0000 1101 0000 1001 0000 0100	 1111 1111	1111 1110 1111 1101 11	 0	Q_1= dl RCL dl AND dl,3 A+M SAR, cx=5
 0000 0100 1111 0011 1111 0111 **1111 1011**	 **1111 1111**	1111 1111 1111 1110 11	 1	Q_1= dl RCL dl AND dl,3 A-M SAR, cx=4
   1111 1101	 **1111 1111**	**1111 1111** **1111 1111** **11**	 1	Q_1= dl RCL dl AND dl,3 SAR, cx=3
   1111 1110	 **1111 1111**	**1111 1111** **1111 1111** **11**	 1	Q_1= dl RCL dl AND dl,3 SAR, cx=2
   1111 1111	 **0111 1111**	**1111 1111** **1111 1111** **11**	 1	Q_1= dl RCL dl AND dl,3 SAR, cx=1
   1111 1111	 **1011 1111**	**1111 1111** **1111 1111** **11**	 1	Q_1= dl RCL dl AND dl,3 SAR, cx=0

Question 5[4x3+ 8 = 20]

Consider the given assembly code and update fill memory**.** For part I, II, III after mov instruction not No marks will be given for mov instruction. You are supposed to understand the given code Part IV fill given memory

Fill Memory and Registers

```
;Part I
.code
 mov ax,0
 mov al,'9'
 add al,'8'
 aaa
 or ax,3030h
```

ax	0107
ax	3137

```
;Part II
.data
 val1 db '5'
 val2 db '7'
.code
 mov ax,0
 mov al,val1
 sub al,val2
 aas
 or al,30h
```

ax	FF08
al	38

```
;Part III

.data
 val1 db 5h
 val2 db 6h
.code

 mov ax,0
 mov bl,val1
 mov al,val2
 mul bl
 aam
```

ax	1E
ax	0300

```
;Part IV
.data
  string1 BYTE "987"
  string2 BYTE "789"
  s3 dB (SIZEOF string1+1)DUP(0),0

.code
main Proc

mov esi,SIZEOF string1 - 1
mov edi,SIZEOF string1
mov ecx,SIZEOF string1
mov bh,0

L1:
  mov ah,0
  mov al, string1[esi]
  add al,bh
  aaa
  mov bh,ah
  add al,string2[esi]
  aaa
  or bh,ah
  or bh,30h
  or al,30h
  mov s3[edi],al
  dec esi
  dec edi
loop L1

mov s3[edi],bh

main ENDP
END main
```

	0	1	2	3	4	5	6	7	8	9
0100	9	8	7	7	8	9	1	7	7	6
	39	38	37	37	38	39	31	37	37	36

Question 6 [10+5+10+5+10= 40]

i. Consider following data declaration and fill memory accordingly. ASCII for **A= 41h**.Data memory start at **0100**

```
.DATA
     v8 label word
     v6 Qword  AB12CD34EF78H,2
     v7 label byte
     v5 DD 0ABCDH, 'ABCD'
     v3 word 0ABH,'AB', 0CDEFH,
               5
     v2 db -1 ,255,-1,12o,16t,0010001b,2
     v1 db 1,2,3,'ABCD',
           0BH,0CH,0DH
        byte "string"
```

```
.CODE
    mov al,sizeof v5
     mov bl,lengthof v5
     mov cl,type v5

     mov al,sizeof v3
     mov bl,lengthof v3
     mov cl,type v3

     mov al,sizeof v1
     mov bl,lengthof v1
     mov cl,type v1

     mov al,v7+1
     mov bx,v8

     mov al, byte ptr v2+3
     mov bx, word ptr v5+2
     mov ecx, dword ptr v6+1
```

AL	08
BL	02
CL	04
AL	08
BL	04
CL	02
AL	0A
BL	0A
CL	01

AL	AB
BX	EF78
Al	0A
BX	0000
ECX	12CD34EF

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4000	78	ef	34	cd	12	ab	00	00	02	00	00	00	00	00	00	00
4010	cd	ab	00	00	44	43	42	41	ab	00	42	41	ef	Cd	05	00
4020	ff	ff	ff	0a	10	11	02	01	02	03	41	42	43	44	0b	0c
4030	0d	73	74	72	69	6e	67									

ii. Update registers and memory after executing the following code. Highlight space in memory created using align.

```
.DATA
    count=3
    b1 byte count dup(1)
    align word
    count=1
    b2 db count dup(count dup(1b))
    count=2
    w1 dw count dup('A','B','C')
    count=1
    align word
    d1 DD count dup('ABCD')
.CODE
    mov eax, offset b2
    mov ebx, offset w1
    mov ecx, offset d1
```

REGISTER

EAX	0104
EBX	0105
ECX	0112

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0100	01	01	01	00	01	41	00	42	00	43	00	41	00	42	00	43
0110	00	00	44	43	42	41										
0120																

iii. Fill in the following table for different caches. Assume all caches are direct mapped. Also, rough work is mandatory for each part to score marks.

No.	Address Bits	Cache Size	Block Size	Tag Bits	Index Bits	Offset Bits	Bits per Row
a.	16	16KB	8B	2	11	3	67
b.	32	32KB	16B	17	11	4	146
c.	64	1024	64B	44	14	6	557
d.	32	512KB	32B	13	14	5	270

Rough Work:

iv. Convert given C++ Code to assembly equivalent for any value of x and y

```
if(x>=9&&y<15)
      {
            if(x!=7||y>5)
            {
                  ax=777;
            }
      }
```

```
mov bl,x
mov cl,y
cmp bl,9
jb exit

cmp cl,5
jae exit


  INCOMPLETE  or Not implemented

    a77:
        mov ax,777

exit:
```

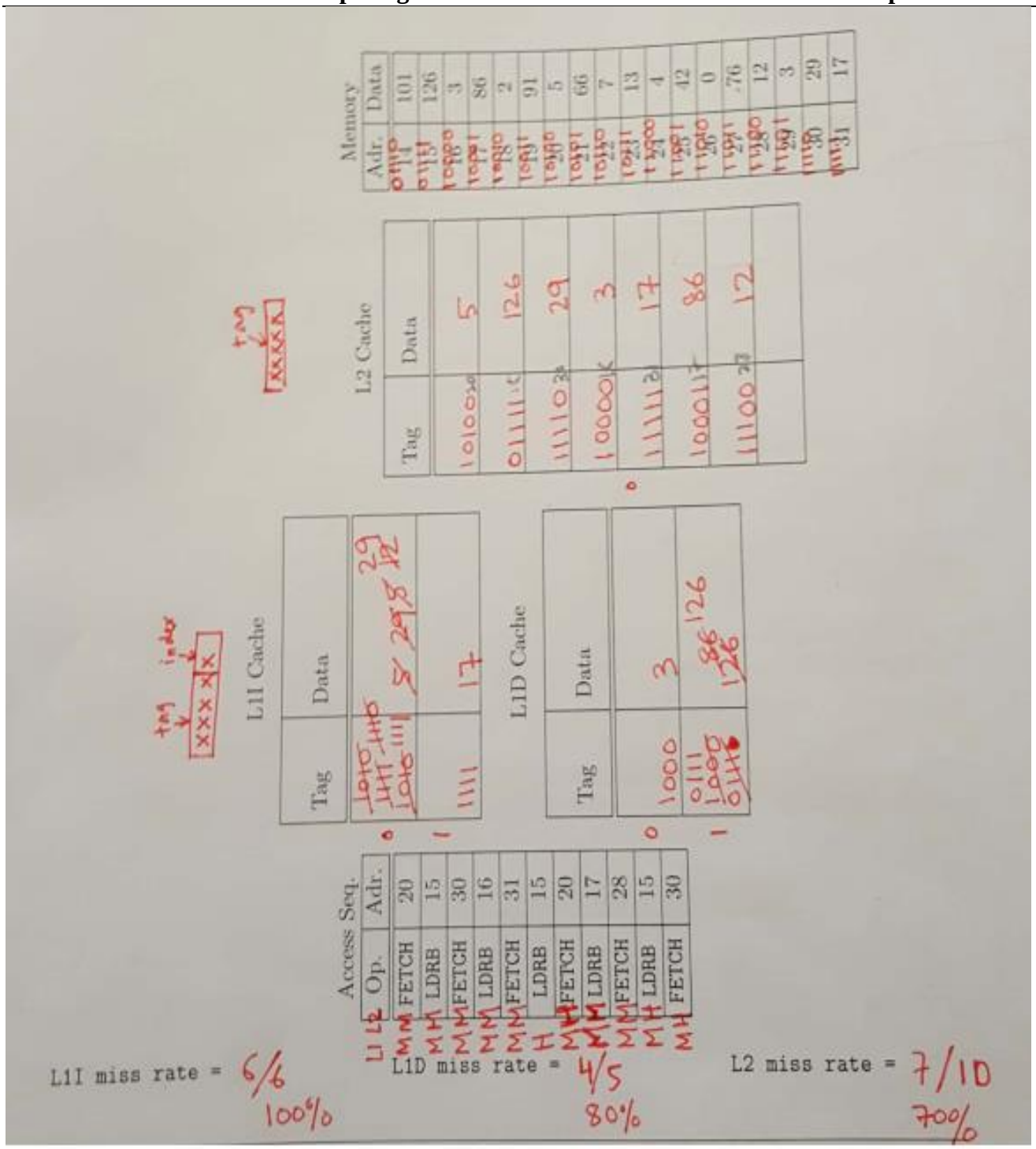
v. Write a program that add **R1=A+B** and subtract **R2=C - D** save results. Your computer is 32-bit processor.

```
.data
      A dq 034ABCDEF12H
      B dq 078ABCDEF12H
      C1 dq 0111230000H
      D dq 00ABCDEF12H
      R1 dq 0
      R2 dq 0
.code
     mov eax,dword ptr A
     add eax,dword ptr B
     mov dword ptr R1,eax
     mov eax,dword ptr A+4
     adc eax,dword ptr B+4
     mov dword ptr R1+4,eax
     mov eax,dword ptr C1
     sub eax,dword ptr D
     mov dword ptr R2,eax
```

```
mov eax,dword ptr C1+4
sbb eax,dword ptr D+4
mov dword ptr R2+4,eax
```

Question 7[15]

Show the final data and tags of the two levels of cache, the given memory contents, and data access sequence. Consider 5-bit memory addresses and single byte instructions. Assume that the L1 caches are direct mapped and the L2 cache is fully associative, using LRU for replacement. All caches can place a single byte of data in each block. Also assume that **addresses greater than 19 contain instructions whereas addresses smaller than 19 contain data.**



Question 8[20]

Note: Data is same as address for all part of this question

i. Fill in the **Direct-mapped cache** for the address given below. Calculate Hit and Miss Rate.

Addresses	30H	3FH	32H	38H	39H	33H	3EH
Address	110000	111111	110010	111000	111001	110011	111110
Hit/Miss	M	M	M	M	H	H	H
Addresses	31H	10H	1EH	11H	38H	33H	
Address	110001	010000	011110	010001	001000	110011	

Hit/Miss	H	M	M	H	H	H	

TAG		0	1	V
~~11~~, 01	000	~~30H~~, 10H	~~31H~~, 11H	1
11	001	32H,	33H	1
	010			
	011			
11	100	38H	39H	1
	101			
	110			
~~11~~,01	111	~~3EH~~,1EH	~~3FH~~,1FH	1

HIT RATE = 7/13

MISS RATE =6 /13

Capacity= bytes

ii. Fill in the **Direct-mapped cache** for the address given below. Calculate Hit and Miss Rate

Addresses	18H	20H	2FH	1AH	28H	0AH	1BH
Address	011000	100000	101111	011010	101000	001010	011011
Hit/Miss	M	M	M	H	H	M	H
Addresses	18H	0EH	2BH	27H	1FH	2CH	
Address	011000	001110	101011	100111	011111	101100	
Hit/Miss	H	H	M	H	H	H	

TAG	LINE	000	001	010	011	100	101	110	111
1	00	20H	21H	22H	23H	24H	25H	26H	27H
~~1, 0,~~ 1	01	~~28, 08,~~ 28	~~29, 09,~~ 29	~~2A, 0A,~~ 2A	~~2B, 0B,~~ 2B,	~~2C, 0C,~~ 2C	~~2D, 0D,~~ 2D	~~2E, 0E,~~ 2E	~~2F, 0F,~~ 2F
	10								
0	11	18h	19h	1Ah	1Bh	1Ch	1Dh	1Eh	1Fh

HIT Rate = 8/13 **Miss Rate=** **5** /13 **Capacity=** **Byte**

iii. Fill in the **4 Way Set Associative Cache** for the address given below. Calculate Hit and Miss Rate.

Addresses	3FH	24H	1FH	0FH	3CH	1EH	27H
Address	111111	010100	011111	001111	111100	011110	100111
Hit/Miss	M	M	M	M	H	H	H
Addresses	3DH	05H	3EH	26H	0EH	24H	
Address	111101	000101	111110	100110	001110	100100	
Hit/Miss	H	M	H	H	M	H	

TAG		00	01	10	11	V	LRU	LRu (binary)
	0							
111	1	3C	3D	3E	3F	1	0,1,2,3,0,1,2,0,1,0, 1, 2	~~00,01, 10,~~ ~~11~~, 00, 01, 10, 00, 01,00, 01,10
010		24	25	26	27	1	0,1,2,3,0,1,2,0,1,0	~~00, 01~~, ~~10,~~ 11, 00 ,01, 10, 00
011, 001		1C, OC	1D, OD	1E, OE	1F, OF	1	0,1,2,0,1,2,3,0,1	00, 01, 10, 00, 01, 10, 11,00,01
~~001~~, 000		~~0C,~~ 04	~~0D~~, 05	~~0E,~~ 06	~~0F,~~ 07	1	0,1,2,3,0,1,2,3	00, 01,10 ,11, 00,01, 10,11

HIT RATE = 6/13 MISS RATE = 7 /13

iv. Fill in the **8 Way Fully Associative Cache** for the address given below. Calculate Hit and Miss Rate.

Addresses	**3FH**	**24H**	**1FH**	**0FH**	**3CH**	**1EH**	**27H**
Address	111111	010100	011111	001111	111100	011110	100111
Hit/Miss	M	M	M	M	H	H	H
Addresses	**3DH**	**05H**	**3EH**	**26H**	**0EH**	**24H**	
Address	111101	000101	111110	100110	001110	100100	
Hit/Miss	**H**	**M**	**H**	**H**	**H**	**H**	

TAG		00	01	10	11	V	LRU
1111		3C	3D	3E	3F	1	0,1,2,3,0,1,2,O,1,0
0101		24	25	26	27	1	0,1,2,3,0,1,2
0111		1C	1D	1E	1F	1	0,1,2,0,1,2,3
0011		0C	0D	0E	0F	1	0,1,2,3,4
0001		04	05	06	07	1	0,1

HIT RATE = 8/13 **MISS RATE = 5/13**

ROUGH