National University of Computer and Emerging Sciences, Lahore Campus



Course:

Program:

BSCS,BSDS,BSR 3 Hour

Duration: Paper Date:

18-Dec-2023

COAL

Section: All Course Code:

Semester: Total Marks:

Your Section:

Roll No.

Page(s):

Solution

EE2003

13

Fall 2023

Exam: Final

Instruction/Notes:

This is an open notes/book exam. Sharing notes and calculators is NOT ALLOWED. All the answers should be written in provided space on this paper. Rough sheets can be used but will not be collected and checked. In case of any ambiguity, make reasonable assumptions. Questions during exams are not allowed.

Question 1 [CLO 1] [1+1+1+2+5 = 10 Marks]: Answer following questions:

Which of the following bus is unidirectional? (i)

Show your working to get credit:

- Data Bus
- (b) Address Bus
- c. Control Bus
- Which of the following instruction can change the value of IP register? (ii)
 - a. Mov
- Call (b)
- c. push
- Which of the following instruction does not change the flag register? (iii)

What is the last physical address of the segment 0xB432? (iv)

(v)

Content of Memory starting from offset 0x139D is given below. Write the updated memory content after execution of the following code. There is not syntax error in this code.

mov bx, 0x319E mov si, 4 mov cx, 0xAB01 mov [bx], cx mov al, [0x31A4] mov [bx+3], al mov word [bx+si], 10

Memory Content before Execution of Code:

	ntent beron	20	15	F5	89	9A	BC	A0
12	34	ZU Turneution of	Code:					
Memory Co	ntent after	AQ	15	BC	09	00	BC	AO
12	0	(D		I DC	0.1	.0		
	0	0		0	-0			

Question 2 [CLO 2] [2+3+5+5+5 = 20 Marks]: (i) Following Code is trying to print '*' on top left cel memory with black background and white foregree Complete the code. (Ascii of '*' is 0x2A.)	II of display ound (0x07).	[ORG 0x0100] mov ax, 0xb800 mov es, ax
		mov byte[es:0], 0x 2A mov byte[es:1], 0x 07 MOV AX, 0x4C00 ; Terminate Program
		INT 0x21
(ii) Following code is trying to highlight the first five characters from top left of the video memory. Originally the whole screen is white foreground on blue background, without blinking. After highlight the specified characters should be blinking with red background, rest should remain the same. Write the mask instruction to accomplish this task.	add si,2	si] Apply required mask in one instruction ax runs the loop cx times
(iii)		
Suppose when the following program loads, the Buffer is at offset 0x0103 in memory. CS=DS=ES=SS=0x19F5	[ORG 0x0100 jmp start buffer: times	o) s 80 dw 0 ;reserves 80 words in memory by placing zeros.
What will be the content of 'buffer' after	start:	

execution of code? Explain in one line only.

Nothing is changed.

At what Logical Address (Base:Offset) the above code is starting copying the data?

O. 6010XO

start: mov ax, 0xb800 mov ds, ax

mov bx, buffer mov es, bx

mov si,0

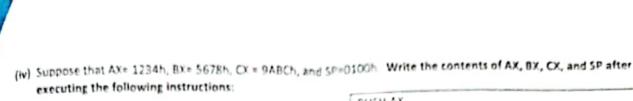
mov cx, 80

cld

mov di, 0

rep movsw

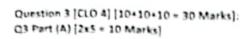
MOV AX, 0x4C00 INT 0x21 ; Terminate Program



ax: <u>0x9ABC</u> bx: <u>0x9ABC</u> cx: <u>0x5678</u> PUSH AX
PUSH BX
XCHG AX, CX
POP CX
PUSH AX
POP BX

(v) We are required to write a program that disables space key on Dosbox. After running our program if user tries to write anything on dosbox, he should not be able to add space, rest of the keys should run normal with command prompt. Partial code is given; complete the following program such that it fulfills the required functionality.

xor ax, ax start: [org 0x0100] mov es, ax imp start oldisr: dd 0 mov ax, [es:9*4] mov (oldisr), ax push ax kbisr: mov ax, [es:9*4+2] in al, 0x60 mov [oldisr+2], ax ;Add your code here (if required) ine nomatch mov word [es:9*4], kbisr mov [es:9*4+2], cs sti imp exit ;Add your code here (if required) jmp TSR no match: pap ax jrmp for (cs.oldisr) Unhook: mov ax, [oldisr] mov bx (oldisr+2) moy [es:9*4], ax mgv [es:9*4+2], bx mov ax. 0x4c00 int 0x21 mov al, 0x20 exit: TSR: out 0x20, al mov dx, start pop ax add dx, 15 iret mov cl. 4 shr dx, cl nomatch: pop ax jmp far (cs:oldisr) mov ax, 0x3100 int 0x21





Consider the cache has 4 blocks/indexes and we have following memory references:

5, 12, 13, 17, 4, 12, 13, 6, 4

What is the hit ratio for the following caches? Show your working below to get credit.

i.	Fully/Simple Associative with FIFO Replacement Algorithm	3/9
li.	Fully/Simple Associative with LRU Replacement Algorithm	3/9
III.	Direct mapping	0
îν.	2-Way Set Associative using LRU Replacement Algorithm	3/0

v. Consider a pipelined and a non-pipelined architecture. In both, one instruction completes in 6 stages and each stage takes 2 us.

Fill-in following information:

Pipelined Architecture	Clock Cycle Time	Frequency = 5×105 H7
Non-Pipelined Architecture	12 45	1/12/10-4 = 83,333 HZ

Show your working here:		
	7	

he inde	e have declared an array in memory of 200 bytes and we have an (fully) associative cache of size 8 bytes and e existores 1 byte of data. For the following code, answer the questions below:
v ax, 0	
v bx, 0	ad Jaconstabul
p1: add	al, [array1+bx] ray1+bx+1]
bx, 1	
bx, 10	00
loop1	
i.	What is the total number of memory references in the code?
ii.	If we have total memory size of 1Mb, how many bits are required for the tag in the cache? 20
	What is the hit rate for the cache for the above code? Use FIFO replacement algorithm. Assume that the
iii.	to in initially ampty
	cache is initially empty. 99/200
W YOU	UR WORKING HERE TO GET CREDIT:
	3 √
	78.00

Q3 Part (C) - Pipelining Question FOR ALL THE SECTIONS EXCEPT BCS-3A, BCS-3B, BCS-3C:

Let us consider the following decomposition of the instruction processing. All stages take equal Set of instructions amount of time i.e 1µs I1: jmp start Fetch Instruction (FI): Read the next expected instruction into a buffer. 12: n1: db 10.20 Decode Instruction (DI): Determine the opcode and the operand specifiers. 13: start: mov ax, 1 Fetch Operands (FO): Calculate the effective address of each source operand and fetch each 14: mov cx, 8 operand from the memory. Operand in registers need not to be fetched. 15: jox 11 Execute Instruction (EI): Perform the indicated operation and store the result if any, in the 16: add ax, cx specified destination operand location. 17: mov bx, 8 Write Operand (WO): Store the result in memory. Following is a set of instructions. Their implementation through pipelining has some control 18: 11: add cx, bx hazards. You have to solve those hazards by using Branch prediction (Predict taken) method. 19: add ax, 4 110: add si, 4 Also calculate latency and throughput. Note: Normal execution will be done in case of simple instructions. 'Jump instructions calculation is done in the execution phase.

Solution:																	_			
Instruction No.	1												_	,						
	-	2	3	4	5	6	7	8	9	10	11	12	13	14	15					
II	FI	DI	FO	EX	WO									14	15	16	17	18	19	20
I3		FI	PI	FO	EX	un						-	-	_						
T4		F	F	DI	EO	EX					-		_							
IS				FI	DI	FO	EX	740		-		-	_	_						
TO				1-		FU	LX	1 5										1	i	
10					FI	DI	FO	5	710									-		
I9						FI	DT			W)			-					_		
IIO							FT.			-								_		
I6								FI	17											
TT						-	-	1	DI	HO	EX	WC								
- 1									FI	DT	FO	A	\Ah							_

	1 1	1)) [FO EX W
Latency: 5,19	18	FIDIFOEXWO
Throughput: 10/16	Iq	FIDEFOEX WO
Fig.	T10	FI IN FO A WO

中的