Name-Last Name: _____ **Student ID:** _____

Section (Check one)

☐ **Section 1 (Wednesday)**

☐ **Section 2 (Friday)**

Hacettepe University	Computer Engineering Department
BBM234 Computer Organization	Instructor: Assoc. Prof. Dr. Suleyman TOSUN
Midterm Exam	Exam Date: 15.04.2016
Duration: 100 minutes	

Questions	**1**	**2**	**3**	**4**	**5**	**Total**
Marks	**20**	**20**	**20**	**20**	**20**	**100**
Earned						

Q1. Suppose we have an array A with 100 elements. The memory address of the first array element is 0xA1B2C3D4. **Write a MIPS function named *ordered*, which checks if the elements of the array A are in ascending order.** If they are, the function should return 1. Otherwise, it returns 0. Call the function *ordered* from the main function. And then, store the return value in register s0.
Note that you do not have to store the variables in the stack when function is called.

Important note: You are allowed to use only the instruction we have seen in our lectures. You cannot use any other pseudo instructions. If you use any, you will lose 5 points for each pseudo instruction.

```
Main:
            lui $a0, 0xA1B2
            ori $a0, $a0, 0xC3D4
            jal ordered
            add $s0, $v0, $0

ordered:    addi $t0, $0, 99      #t0=array_size-1 (we have 99 comparisons)
            addi $t1, $0, 0       #i=0
            addi $v0, $0, 1       #v0=1 means array is ordered.
      for:  beq $t0, $t1, done    #loop 99 times (99 comparisons)
            lw $t2, 0($a0)        #load the first element to t2
            lw $t3, 4($a0)        #load the next element to t3
            slt $t4, $t3, $t2     #t4=1 if t3<t2
            beq $t4, $0, else     #if t4=0 (means that t3 is not less than t2), go to next comparison
            addi $v0, $0, 0       #v0=0 (array is not ordered) since t3<t2
            jr $ra                #we do not need to check the array further. Return to main function.
      else: addi $t1, $t1, 1      #i=i+1
            addi $a0, $a0, 4      #point next array element
            j ordered             #go to loop to compare next two numbers
      done: jr $ra                # Go to main.  (If we reach this line, that means array is ordered. )
```

Q2. You are given a MIPS program below.

a) Fill the given table for this program by entering the following:
- How many times does each instruction execute (fetched) in the program?
- What is the clock cycles of each instruction when executes on multi-cycle processor? Diagram of the multi-cycle processor is given in Question 5 if you need it
- What are the total clock cycles for each instruction and the whole program when executed on multi-cycle processor?

	Instruction	**How many times does an instruction execute?**	**Clock cycles of the instruction for multi-cycle processor**	**Total clock cycles for multi-cycle processors**
	addi $s0, $0, 2	1	4	4
loop:	beq $s0, $0, done	2	3	6
	srl $s0, $s0, 1	1	4	4
	bne $s0, $0, else	1	3	3
	lw $s5, 0($a0)	0	5	0
	add $s5, $s5, $s0	0	4	0
	sw $s5, 0($a0)	0	4	0
else:	addi $s0, $s0,-1	1	4	4
	j loop	1	3	3
done:	jr $ra	1	3	3
Total (Single-Cycle):		**8**	**Total (Multi-Cycle):**	27

b) Show the formats of the instructions beq (opcode=4) and srl (funct=2). Then, write machine codes for these instructions in hexadecimal format. (s0=16)

beq $16, $0, done (It is an I-type instruction)

opcode	rs	rt	immediate
000100	00**000**	**1**0000	**0000** 0000 **0000** 0111
1 0	**1**	0	**0** 0 **0** 7

When we determine the value of immediate field, we count the number of instruction between the srl (next instruction after beq) and the instrcution at the done line (jr $ra). It is 7.

srl $16, $16, 1 (It is an R-type instruction.)

opcode	rs	rt	rd	shamt	funct
000000	00**000**	**1**0000	**1000**0	000**01**	**00**0010
0 0	**1**	0	**8** 0	**4**	2

Instruction	Hexadecimal Machine Code
beq $s0, $0, done	0x1010_0007
srl $s0, $s0, 1	0x0010_8042

Q3. We would like to add **blez** instruction to single cycle MIPS processor. blez instruction copies the branch target address (BTA, which is *PCBranch* in Figure 2) to the program counter (PC) if the value in register [rs] is less than or equal to zero. In other words, if [rs] ≤ 0, PC = BTA.

a) First, add an output signal to the ALU (given in Figure 1) and name this signal as LTEZ (Less Than Equal to Zero). LTEZ=1 when the Result is less than or equal to zero (i.e., Result ≤ 0). Otherwise, LTEZ=0. [5]

b) By using LTEZ output signal, show the necessary changes on data-path of single-cycle processors given in Figure 2 and explain your changes. [10]

c) Fill the control signals in Table I. [5]

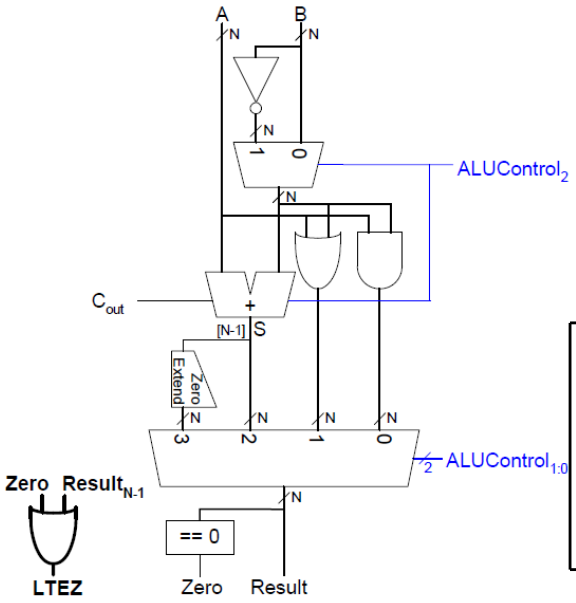


Figure 1: ALU

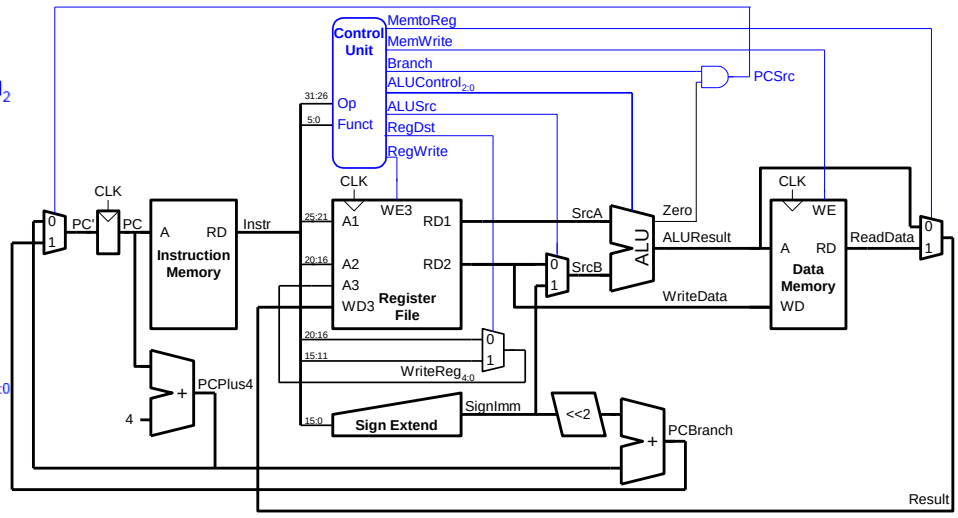
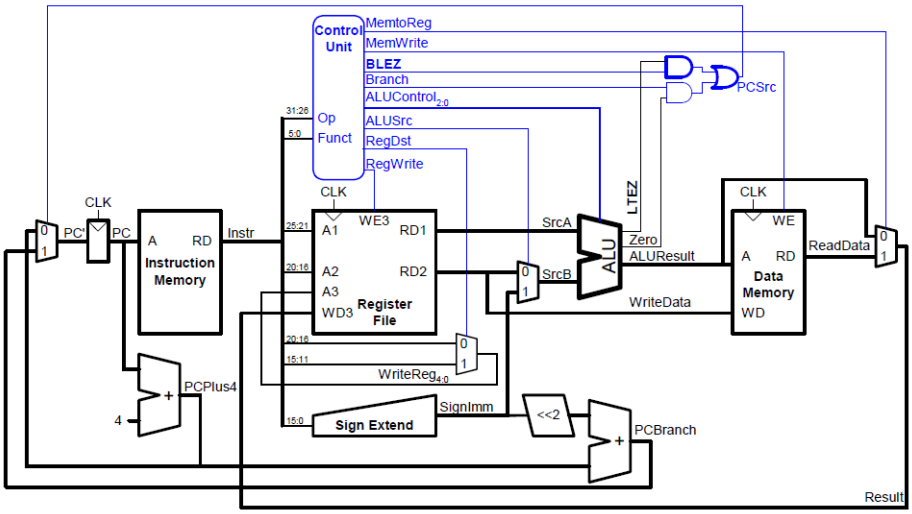


Figure 2: Single Cycle MIPS Processor

If $Result_{N-1}$ is 1, that means the result is negative and it is less than zero. If the Result is zero, the *zero* output will be 1. If we OR these two signals, we get LTEZ.

We add BLEZ control signal to the controller. If BLEZ is 1 and LTEZ is 1, the PCSrc will be 1 and the BTA will be stored in PC. The contol for beq does not change and we can OR them together.
There is no other changes to the data path.



c) Tabel I: Control signals for **blez** instruction

Inst.	$Op_{31:26}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$	BLEZ
blez	000110	0	X	0	0	0	X	01	1

Q4. **a)** In a MIPS architecture, indicate if the following instructions use sign extend logic or ALU. If an instruction uses ALU, what is the type of operation (add, subtract, and, or, …etc)? [10]

Instruction	Uses sign extend (YES or NO)	Uses ALU (YES or NO)	Type of ALU operation
lw $s0, 0($a0)	YES	YES	ADD
sllv $s0, $s1, $s2	NO	YES	SHIFT
sw $s0, 0($a0)	YES	YES	ADD
jal target	NO	NO	---
bne $s0, $s1, target	YES	YES	SUBTRACT

b) Suppose following delays have been determined for the elements of the single-cycle processor. You can ignore the delays of other elements in the design.

Element	Parameter	Delay (ps)
Register clock-to-Q	t_{pcq_PC}	30
Register setup	t_{setup}	20
Multiplexer	t_{mux}	25
ALU	t_{ALU}	200
Memory read	t_{mem}	250
Register file read	t_{RFread}	150
Register file setup	$t_{RFsetup}$	20

b1) What is the worst case delay for R-type instructions? Show how you determine this value. [5]

$T_{R\text{-}Type} = t_{pcq\_PC} + t_{mem} + t_{RFread} + t_{mux} + t_{ALU} + t_{mux} + t_{RFsetup}$ =(30+250+150+25+200+25+20)ps=700ps

After reading the instrcution, register values are read and then the ALU operation takes place. Then, the result is written to regster file. The last mux is the mux in front of PC and T_setup is for writing to PC.

b2) What is the worst case delay for beq instruction? Show how you determine this value. [5]

$T_{beq} = t_{pcq\_PC} + t_{mem} + t_{RFread} + t_{mux} + t_{ALU} + t_{mux} + t_{setup}$ =(30+250+150+25+200+25+20)ps=700ps

After reading the instrcution from instruction memory, tow register values are read and they are subtracted. If the result is zero, then the BTA address is written to the PC. The last mux is the mux in front of PC and T_setup is for writing to PC. AND operation is ignored.

Q5.

Q5. In the multi-cycle processor given in Figure 3, lw instruction takes 5 clock cycles.

1. Fetch: Fetch the instruction from instruction memory and store it into the instruction register.
2. Decode: Read the operands such as registers and immediate values.
3. Execute: Add operands (add rs and sign extended immediate value.)
4. Read memory: Read the data from memory and store it into data register.
5. Write result: Write the data from data register to destination register in register file.

Your goal is to ***design a microarchitecture for lw instruction that takes only 3 clock cycles***. You should merge cycles Fetch and Decode into FetchDec and merge Execute and Read memory into (ExecMem). Write result cycle will be the same.

a) Draw the data path using the 32-bit Program Counter (PC), instruction/data memory, register file, and additional hardware if necessary. You should clearly show the extra hardware and the bit numbers of each connection. Note that PC must be incremented in the first cycle. [10]

b) Draw the FSM (Finite State Machine) for new lw instruction. [10]

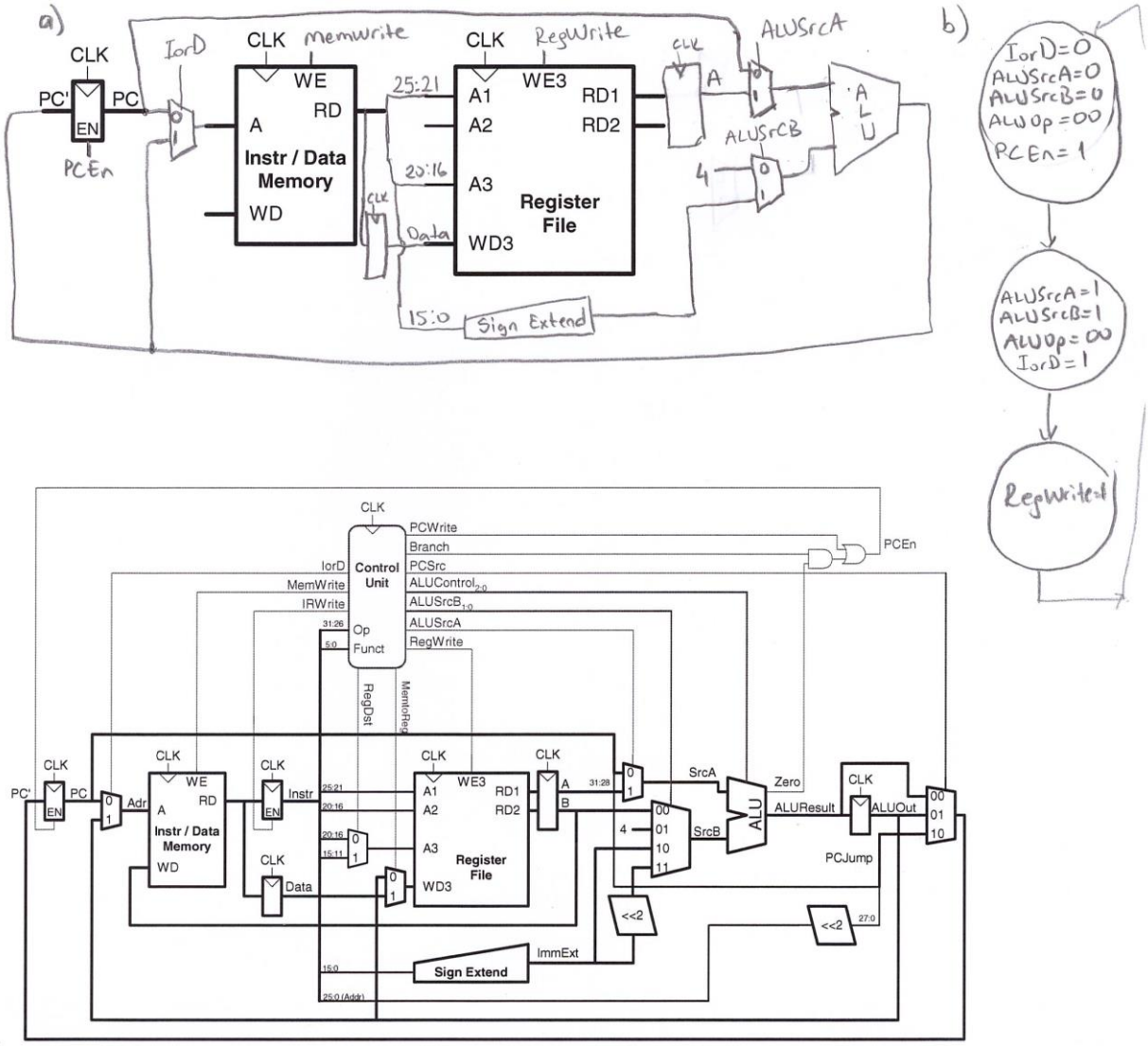


Figure 3:Multi cycle processor