Name-Last Name: ______________________________ **Student ID:** ________________________

Hacettepe University	Computer Engineering Department
BBM234 Computer Organization	Instructors: Suleyman TOSUN, Mehmet KOSEOGLU
Midterm Exam	
Duration: 3 hours (13:00-16:00)	Exam Date: 19.06.2019

Questions	**1**	**2**	**3**	**4**	**Total**
Marks	**25**	**25**	**30**	**20**	**100**
Earned					

Question 0: Write the following statements on a separate sheet by hand and sign. Include the sheet as the first page of your pdf file. If you do not include these statements in your final submission, your exam will not be graded.

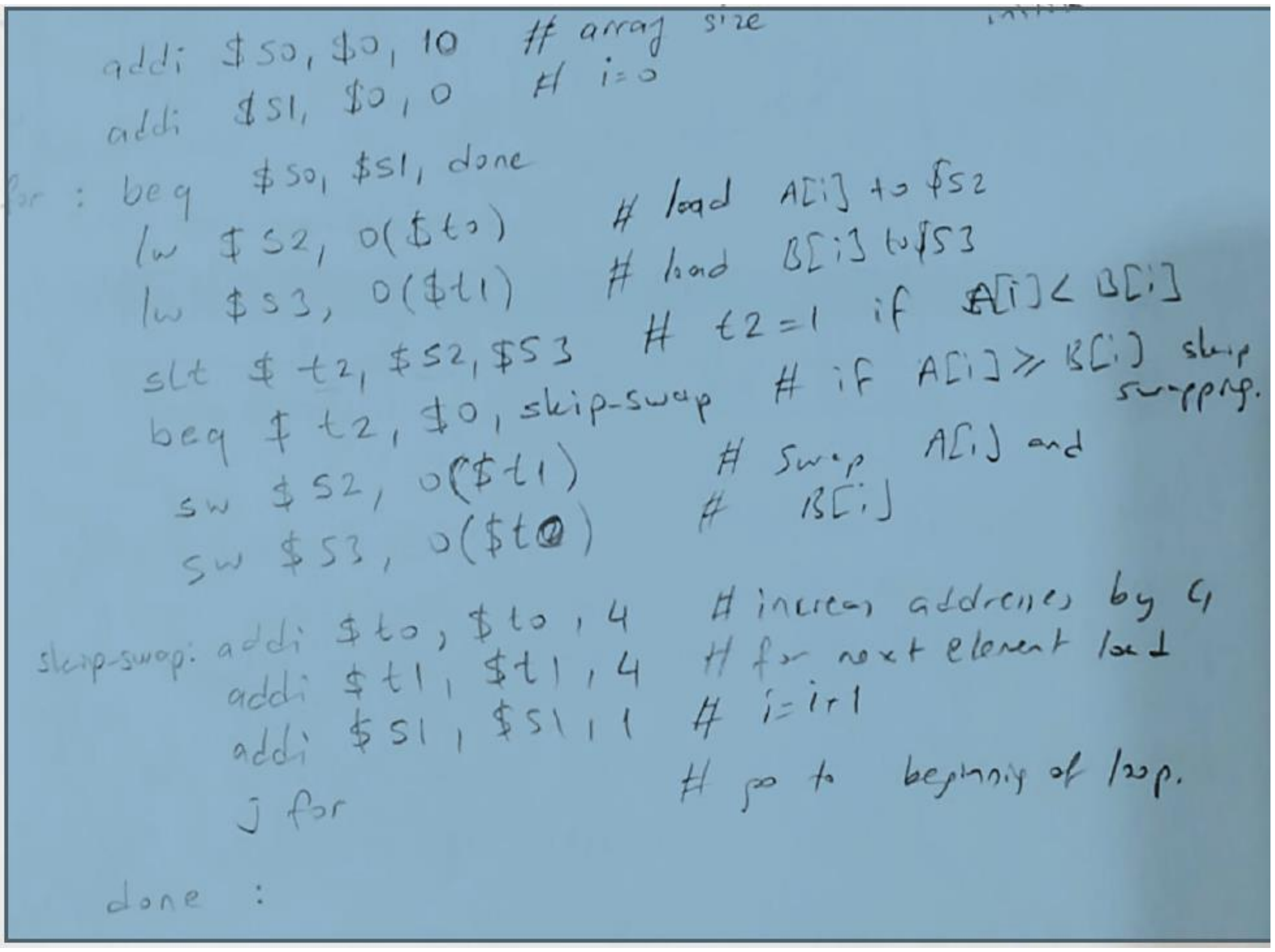
1) I will solve each question on a seperate sheet. I will write my name on top of each sheet.
2) I will make a single pdf file in the order of the questions.
3) I will upload the pdf file through submit.cs.hacettepe.edu.tr.
4) I understand that computer-typed solutions will not be accepted. I must handwrite.
5) I will not attempt any form of cheating.
6) I am aware that this is an open-book exam.

Q1. Suppose we have two arrays A[10] and B[10]. Write a MIPS code that does the following:

It compares A[i] and B[i]. If A[i] <B[i], it swaps their values. Otherwise, there is no swap. There is an example below. Suppose the addresses of A[0] and B[0] are in registers t0 and t1, respectively.

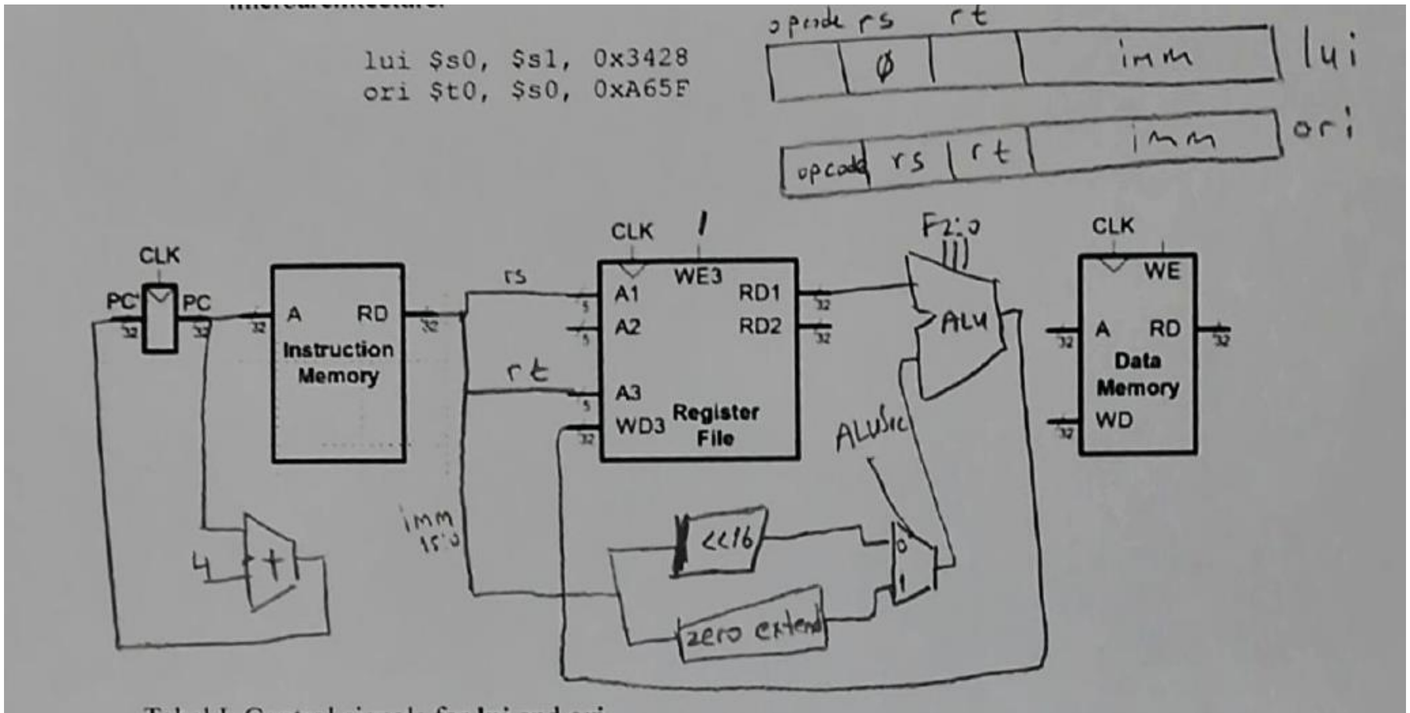
Inputs			**Outputs**	
A	**B**		**A**	**B**
5	6	5<6; swap	6	5
10	8	10>8; no swap	10	8
2	5	2<5; swap	5	2
8	8	8=8; no swap	8	8
-5	4	-5<4; swap	4	-5
4	-2	4>-2; no swap	4	-2
20	10	20>10; no swap	20	10
3	5	3<5; swap	5	3
10	11	10<11;swap	11	10
5	3	5>3; no swap	5	3

5 ponits for each correct for loop, loads (lw), comparing A[i] and B[i], swapped stores (sw) and iteraton increaments.



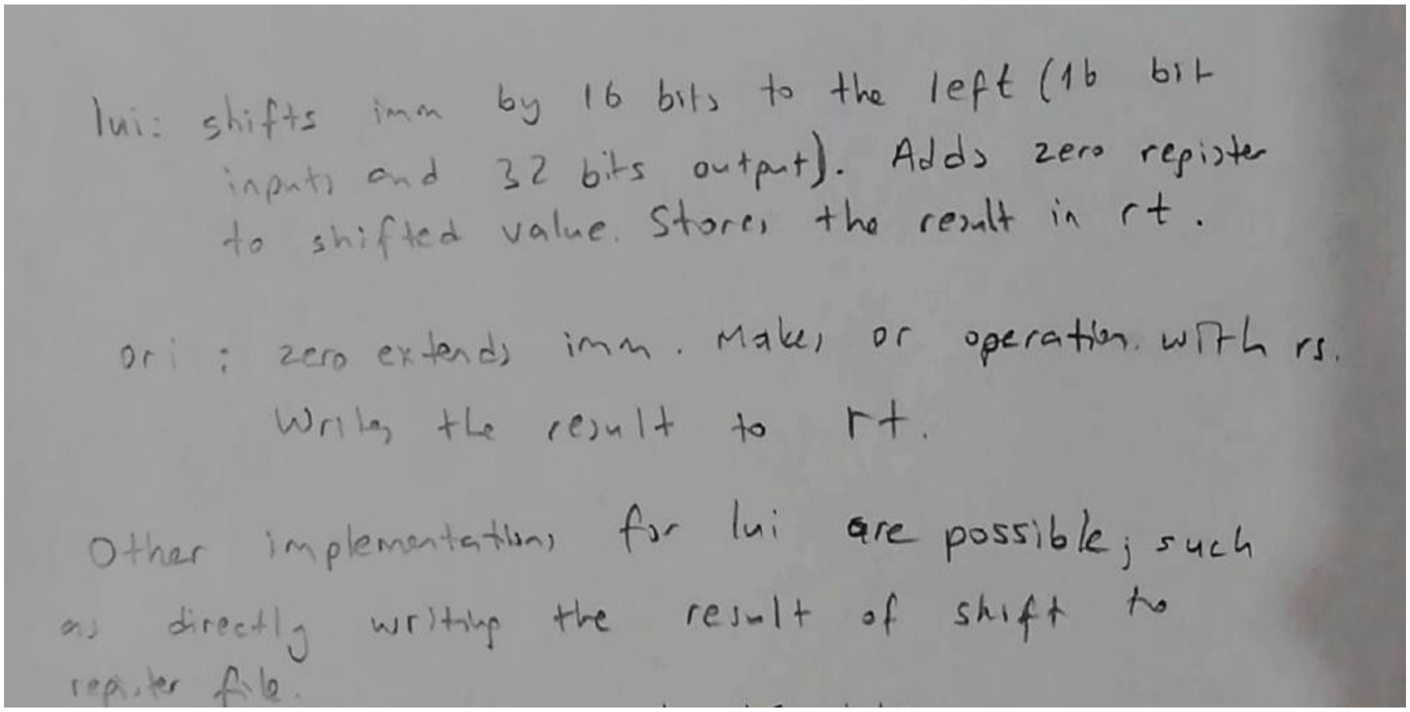
```
      addi $s0, $0, 10        # array size
      addi $s1, $0, 0         # i=0
for : beq $s0, $s1, done
      lw $s2, 0($t0)          # load A[i] to $s2
      lw $s3, 0($t1)          # load B[i] to $s3
      slt $t2, $s2, $s3       # t2=1 if A[i]< B[i]
      beq $t2, $0, skip-swap  # if A[i] ≥ B[i] skip
                              #   swapping.
      sw $s2, 0($t1)          # Swap A[i] and
      sw $s3, 0($t0)          # B[i]

skip-swap: addi $t0, $t0, 4   # increas addresses by 4
      addi $t1, $t1, 4        # for next element load
      addi $s1, $s1, 1        # i=i+1
      j for                   # go to beginning of loop.

done :
```

Q2. You have 32-bit Program Counter (PC), instruction memory, register file, and data memory. Draw a single-cycle processor that can execute <u>ONLY the following type of instructions</u>. Define necessary control signals and write their values in the table. Note that you do not need some of the control signals in the table. Write NA (Not Applicable) to them if they are not used in your microarchitecture.



```
lui $s0, $s1, 0x3428
ori $t0, $s0, 0xA65F
```

Tabel I: Control signals for **lui** and **ori**

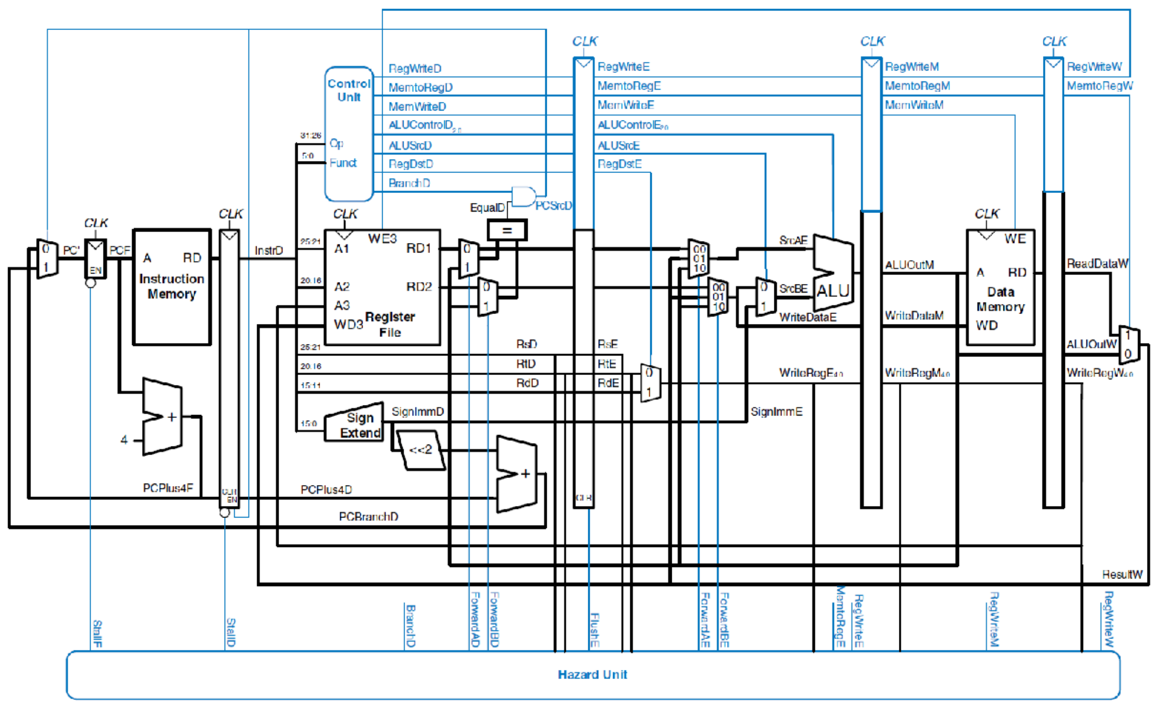


lui: shifts imm by 16 bits to the left (16 bit inputs and 32 bits output). Adds zero register to shifted value. Stores the result in rt.

ori : zero extends imm. Makes or operation with rs. Writes the result to rt.

Other implementations for lui are possible; such as directly writing the result of shift to register file.

15 points to datapath. 10 points to control signals if data path is correct or control signals matched the control signals of your design.



Q3. Solve the part a and b according to the following sample program, which is executed in the pipelined architecture above. The current stages of the instructions are shown next to the instructions in parentheses.

```
sw  $t2, $t3(0)  (WriteBack)
add $s0, $s2, $s3  (Memory)
or $s4, $s0, $s1  (Execute)
sw  $t5, $t4(0)  (Decode)
sub $s0, $s1, $s2  (InstructionFetch)
```

a) (5 points) Fill the control signals of the instructions in the following table. Note that you get points only if the majority of your answers are correct.

MemWriteD	ALUSrcE	RegDstE	MemWriteM	RegWriteW
1	0	1	0	0

When the code above is executed, the hazard unit sets ForwardAE to 10 because `add $s0, $s2, $s3` instruction changes the register $s0, which is used by the `or $s4, $s0, $s1` instruction. Considering this hint, answer the following questions:

b) (8 points) Modify the above program such that the hazard unit to set the ForwardBE to 10. Indicate the stages of each instruction similar to the example above. Explain your reasoning

sw $t2, $t3(0) (WriteBack)

add $s0, $s2, $s3 (Memory)

or $s4, $s1, $s0 (Execution) Similar to the example, but rt is forwarded instead of rs.

sw $t5, $t4(0) (Decode)

sub $s0, $s1, $s2 (InstructionFetch)

c) (8 points) Write a 5-line MIPS program such that the hazard unit to set the ForwardAE to 01. Indicate the stages of each instruction similar to the example above.

and $s0, $s5, $s6 (WriteBack) $s0 is forwarded from the writeback stage to the execution stage

add $s4, $s2, $s3 (Memory)

or $s4, **$s0**, $s1 (Execution)

sw $t5, $t4(0) (Decode)

sub $s0, $s1, $s2 (InstructionFetch)

d) (9 points) Write a 5-line MIPS program, which causes the hazard unit to set the ForwardAD to 1. Indicate the stages of each instruction similar to the example above.

sw $t2, $t3(0) (WriteBack)

add $s0, $s2, $s3 (Memory)

or $s4, $s1, $s0 (Execution)

beq $s0, $s5, somewhere (Decode) $s0 is forwarded from the Memory stage

sub $s0, $s1, $s2 (InstructionFetch)

Q4.

a) (3 points) Assume we have a 5-stage MIPS processor and it has a forwarding unit (but no stall and flush). For the given code executing on this processor, insert NOPS to the code if necessary. How many clock cycles does it take to finish to execute this code.

```
lw $s1, 0($s0)
lw $s2, 4($s0)
NOP
add $s3, $s1, $s2
sw $s3, 12($s0)      9 instructions + 4 = 13
lw $s4, 8($s0)
NOP
add $s5, $s1, $s4
sw $s5, 16($s0)
```

b) (5 points) Rearrange the code if possible to solve the data hazard(s). Is there any reduction on the total clock cycles?

```
lw $s1, 0($s0)
lw $s2, 4($s0)
lw $s4, 8($s0)
add $s3, $s1, $s2
sw $s3, 12($s0)        7 instructions + 4 = 11 cycles
add $s5, $s1, $s4
sw $s5, 16($s0)
```

c) (4 points) Assuming no hazard unit (no flush, no stall, no forwarding) and branch decision is done in the <u>memory</u> stage, insert NOPs to the following code to solve the hazard(s).

```
add $s4, $s5, $s6
beq $s1, $s2, Target
NOP
NOP
NOP
lw $s3, 300($s0)
sub $s7, $s8, $s9
sw $t1, 4($8)
Target:
```

d) (4 points) Assuming no hazard unit (no flush, no stall, no forwarding) and branch decision is done in the <u>decode</u> stage, insert NOPs to the following code to solve the hazard(s).

```
add $s4, $s5, $s6
beq $s1, $s2, Target
NOP
lw $s3, 300($s0)
sub $s7, $s8, $s9
sw $t1, 4($8)
Target:
```

e) (4 points) Assuming no hazard unit (no flush, no stall, no forwarding) and branch decision is done in the <u>decode</u> stage, insert NOPs to the following code to solve the hazard(s).

```
add $s4, $s5, $s1
NOP
NOP
lw $s1, 0($s4)
NOP
NOP
beq $s1, $s0, Target
NOP
lw $s3, 300($0)
.
.
.
Target:
```