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

Hacettepe University	Computer Engineering Department
BBM234 Computer Organization	Instructor: Assoc. Prof. Dr. Suleyman TOSUN
Midterm Exam	Exam Date: 1.6.2016
Duration: 120 minutes	

Questions	1	2	3	4	Total
Marks	20	40	20	20	100
Earned					

Q1. You are given the following MIPS assembly code for a procedure called proc1.

```
proc1:  add  $s1, $0, $0
        add  $v0, $0, $0
count:  beq  $a0, $0, done
        andi $s0, $a0, 0x1
        beq  $s0, $0, shift
        addi $s1, $s1, 1
shift:  srl  $a0, $a0, 1
        jal  proc1
done:   add  $v0, $v0, $s1
        jr   $ra
```

(a) proc1 function does not successfully return when it is called from the main function. Fix the code so that it successfully returns. (Hint: You should modify one of the instructions.) [5]

When jal proc1 instruction executes, we loose the content of register ra, which is the return address to the main. We should change this instruction to "j count" so that the loop can executes again.

(b) Using the corrected code, write the return value ($v0) for given values of $a0. Write your answer in decimal. [10]

a0 | 0x000F | v0 | 4

a0 | 0x0013 | v0 | 3

(c) Describe in words what proc1 does. [5]

It counts the number of 1's in register a0.

Q2. Following parameters are given for byte addressable memories:

Capacity of virtual memory	8 pages
Capacity of main memory	2 pages
Capacity of cache	4 words
TLB access time	1 ns
Cache access time	50 ns

Page size	4 words
Block size	1 word
Word size	4 bytes
TLB/Cache associativity	2-way
Replacement type	LRU

Suppose the page table, TLB, and cache have the following data in them.

Page Table

VPN	V	PPN
7		
6		
5	1	0
4		
3		
2	1	1
1		
0		

Cache

	Way 1				Way 0		
	U	V	Tag	Data	V	Tag	Data
Set 1	1	1	11	D	1	10	C
Set 0	1	1	01	B	1	00	A

TLB

	Way 1				Way 0		
U	V	VPN	PPN	V	VPN	PPN	
1	1	010	1	1	101	0	

Based on the above information, answer the following questions: [Each is 5 points.]

(a) How many bits do we need to address the virtual memory? 7

The number of bytes in virtual memory = number of pages in VM x page size in words x word size in bytes = 8x4x4=128. Thus, we can use 7 bits to address 128 bytes.

(b) How many bits do we need to address the main memory? 5

The number of bytes in main memory = number of pages in MM x page size in words x word size in bytes = 2x4x4=32. Thus, we can use 5 bits to address 32 bytes.

You are given the virtual address **0x24**.

(c) What is the physical address for this virtual address?

Since the last two bits are byte offset, and the next two bits are page offset (4 words in a page), the remaning bits are used to translate virtual pages to physical pages.

VA= 0x24 is 010 0100 in binary. The physical page number for page 010 ise 1. Then, the physical address is 10100.

(d) Is this memory access is a hit or miss on the cache? Why? Show your work.

The physical address is 10100. The last two bits are byte offset. Since we have two sets in cache, the next bit is for set. We should look to set 1. The tag is 10, which is the tag in way 0. And the valid bit (V) is 1. Thus, it is a hit.

(e) If it is hit, which data is accessed? (A,B,C,D?) If not, where do we bring the new data?

C

(f) After the memory access 0x24, is there any changes necessary on the TLB or cache? If yes, show the changes made on TLB and/or cache.

Cache

		Way 1			Way 0		
	U	V	Tag	Data	V	Tag	Data
Set 1	1	1	11	D	1	10	C
Set 0	1	1	01	B	1	00	A

TLB

	Way 1				Way 0		
U	V	VPN	PPN	V	VPN	PPN	
~~1~~0	1	010	1	1	101	0	

In the TLB, Least Recently Used (LRU) data now is in way 0. Thus, U bit should be changed to 0.

(g) What is the total access time of data?

TLB is hit and cache is hit. TLB time + Cache time = 1ns+1ns=2ns

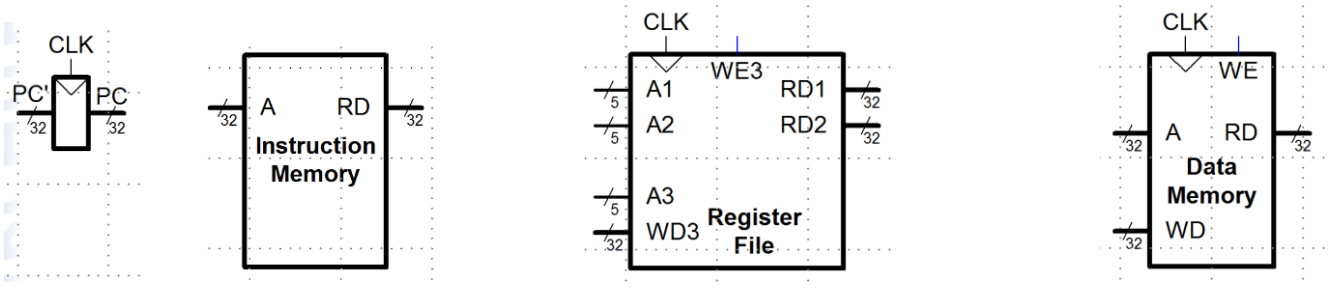
(h) What is TLB and why is it used?

TLB is the cache for the page table. It stores the page table entry for the recently accessed data. In this way, it reduces the number of main memory accesses.

Q3. You have 32-bit Program Counter (PC), instruction memory, register file and a data memory.

(a) Give an example R-type instruction and write its machine code format, including the bit numbers. (You do not need to give the binary values. You can write the labels of each field of the format.) [5]

(b) Draw the single cycle microarchitecture for ONLY R-type instructions. You can use extra hardware blocks if necessary. You should clearly show the bit numbers. [15]



Q4. You are given the following MIPS code:

```
                                  # of executions
      addi $t0, $0, 2             1
loop: beq $t0, $0, done           3
      lw $s0, 0($a0)              2
      lw $s1, 4($a0)              2
      add $s0, $s0, $s1           2
      sw $s0, 0($a0)              2
      addi $t0, $t0, -1           2
      addi $a0, $a0, 4            2
      j loop                      2
done:jr $ra                       1
```

a) How many cycles does it take to execute this code on a single-cycle processor? [3]

Total number of instructions fetched is 19. So, it takes 19 cycles.

b) How many cycles does it take to execute this code on a pipelined processor (with data hazard unit)? (If any NOPs necessary, write it on the above code.) [5]

```
                                  # of fetches
      addi $t0, $0, 2             1
loop: beq $t0, $0, done           3
      lw $s0, 0($a0)              3 (last fetch is flushed)
      lw $s1, 4($a0)              2
      NOP                         2
      add $s0, $s0, $s1           2
      sw $s0, 0($a0)              2
      addi $t0, $t0, -1           2
      addi $a0, $a0, 4            2
      j loop                      2
done:jr $ra                       3 (first three fetches are flushed)
```

Total number of fetches is 24. The last instruction needs 4 more cycles to finish. Then, the answers is 28 cycles.

c) Suggest a modification for the code (without adding or deleting any instructions) so that new code takes less time to execute. Explain your changes. [4]

# of fetches

```
        addi $t0, $0, 2        1
loop:   beq $t0, $0, done      3
        lw $s0, 0($a0)         3  (last fetch is flushed)
        lw $s1, 4($a0)         2
        addi $t0, $t0, -1      2
        add $s0, $s0, $s1      2
        sw $s0, 0($a0)         2
        addi $a0, $a0, 4       2
        j loop                 2
done:   jr $ra                 3  (first three fetches are flushed)
```

We move `addi $t0, $t0, -1` instruction in place of NOP. Then, we gain 2 cycles. The total number of cycles becomes 26.

d) In some cases, loop unrolling (writing the code sequentially instead of using loops) helps reduce the CPU time. Rewrite the above code without using any loops. You can use extra registers. Write your code in such a way that there is no stall or NOP penalty. Now, how many cycles does this code takes on pipelined processor? [8]

```
lw $s0, 0($a0)
lw $s1, 4($a0)
lw $s2, 8($a0)
add $s0, $s0, $s1
add $s1, $s1, $s2
sw $s0, 0($a0)
sw $s1, 4($a0)
```

Now, 7 instructions are fetched, plus 4 cc. The total execution time is 11 clock cycles.