University of Algiers 1

Faculty of Sciences
Department of Computer Science

Make-up Exam L2-ADO
14-June-2026

Duration : 90 minutes

Surname: First name: Group: Student ID No:	/ 20

Exercise 1: (10 points)

Consider the following instruction sequence:

```
.data
tab:    .word 10, 20, 30, 40, 50, 60, 70
.text
        addi  $s1, $0, 0
        addi  $s3, $0, 3
        la    $s4, tab
Loop:   add   $t1, $s1, $s1
        add   $t1, $t1, $t1
        add   $t1, $t1, $s4
        lw    $t0, 0($t1)
        add   $s5, $s5, $t0
        addi  $s1, $s1, 1
        bne   $s1, $s3, Loop
Exit:   li    $v0, 10
        syscall
```

Assume tab starts at 0x10010000.

1. Translate the instruction (**bne $s1, $s3, Loop**) into its 32-bit machine code format, then put the answer in hexadecimal format. List all the details.

(1.5 pts. missing details = no mark)

Type I, Opcode for bne = 5 (hex) = 00 0101 binary
$s1 (rt) = 17 (decimal) = 10001 binary
$s3 (rs) = 19 (decimal) = 10011 binary
Offset = -7 = 1111 1111 1111 1001 binary
(0x0040000C – (0x00400024 + 4))/4 = −0x1C/4= -7
Binary: 0001 01 10 001 1 0011 1111 1111 1111 1001
Hexadecimal: 0x1633FFF9

2. What addressing mode is used in this instruction?

(0.5 pt. missing details = no mark)

PC indexed addressing -> PCnew=(PCcurrent+4)+(immediate×4)

3. List all active Datapath components for this instruction, assuming a 5-stage MIPS pipeline (IF, ID, EX, MEM, WB).

(1.5 pts. missing details = no mark)

PC, Instruction Memory, Register File (read port for rs, rt), Sign Extender, ALU (to subtract)
(Plus adder related to PC, Shift Left 2, Adder (PC+4 + offset))

4. What is the **final value** of register **$s4** after running the program? Explain.

(1 pts. missing details = no mark)

$s4 = 0x10010000
Explanation:
The instruction "**la $s4, tab"** loads the base address of the array tab into $s4.
No instruction in the program modifies $s4 after that. Therefore, its final value remains 0x10010000

5. During the **second iteration** (just before **bne**), what is the value of register **$t1** (as an address in hexadecimals)? Explain.

0x10010004 **(1.5 pts. missing details = no mark)**

Explanation (calculations):

- Iteration 1: $s1 = 0 → after addi $s1, $s1, 1 → $s1 = 1
- Start of iteration 2, before bne:
  - add $t1, $s1, $s1 → $t1 = 1 + 1 = 2
  - add $t1, $t1, $t1 → $t1 = 2 + 2 = 4 (byte offset)
  - add $t1, $t1, $s4 → $t1 = 4 + 0x10010000 = 0x10010004

6. Fill in the table below for the first three iterations **(1.5 pts.)**

Iteration	$s1	Loaded address	Array index	Loaded value	$s5 after
1	0	0x10010000	tab[0]	10	10
2	1	0x10010004	tab[1]	20	30
3	2	0x10010008	tab[2]	30	60

7. What is the final value of **$s5** after the program terminates?

value of $s5: 60 **(0.5 pt. missing details = no mark)**

explanation:
$s5 accumulates the sum of tab[0] + tab[1] + tab[2] = 10 + 20 + 30 = 60.
The loop runs for $s1 = 0, 1, 2 (three iterations). When $s1 = 3, bne condition fails (3 ≠ 3 is false), and the program exits.

8. The **bne** instruction at the end of the loop creates a control hazard. Assuming the pipeline always predicts "not taken", how many penalty cycles occur when the branch is taken? Assume branch resolution happens in EX stage

Since the pipeline always predicts “no branching” and since the branch resolution happens during the EX stage this means there are two instructions that have been wrongly fetched and decoded during the IF and ID stages -> So **2 cycles are lost (penalty)** when the branch is taken. **(1 pt. missing details = no mark)**

9. How many total cycles are wasted over all iterations of the loop

Iteration 1: branch taken (to Loop) → 2 penalty cycles **(1 pt. missing details = no mark)**
Iteration 2: branch taken → 2 penalty cycles
Iteration 3: branch **not taken** (falls through to Exit) → 0 penalty cycles
Total = 4 penalty cycles

Exercise 2: (10 points)

Consider an L1 cache memory with the following characteristics:

- **32 words per line** (1 word = 4 bytes)
- Total cache size = **128 KB**
- **4-way set-associative**
- Replacement policy: **LRU**
- Physical address bus width: **20 bits**

1. How many **lines** are there in this cache?

1 Number of lines=Total cache size/block size **(1.5 pts. missing details = no mark)**
Number of lines= (128×1024 bytes)/ (32x4 bytes/line) =1024 lines

2. How many **sets** are there in this cache?

Number of sets= Number of lines/ Associativity **(1.5 pts. missing details = no mark)**
=1024/4= 256 sets

3. Calculate the **tag size** for this cache.

- **Block offset bits** = $\log_2(\text{Block size}) = \log_2(128) = 7$ bits
- **Index bits** = $\log_2(\text{Number of sets}) = \log_2(256) = 8$ bits
- **Tag bits** = Total address bits − Index bits − Block offset bits
  $= 20 - 8 - 7 = 5$ bits

(2 pt. missing details = no mark)

4. If the main memory is **512 MB**, how many **different memory blocks** map to the same cache set?

Main memory = 512 MB = $512 \times 1024 \times 1024 = 2^{29}$ bytes

(2 pts. missing details = no mark)

Block size = 128 bytes = 2^7 bytes

$$\text{Total memory blocks} = \frac{2^{29}}{2^7} = 2^{22} \text{ blocks}$$

Number of sets = 256 = 2^8

$$\text{Blocks per set} = \frac{2^{22}}{2^8} = 2^{14} = 16{,}384 \text{ memory blocks}$$

Now assume we have an L2 cache. On an L1 miss, the system accesses L2. On an L2 miss, the system pays a 10-cycle penalty to access main memory. Use the table below. Clock cycle duration is T ns.

Level	Hit time (ns)	Hit rate	Miss penalty	Size
L1	3	80%	(goes to L2)	256 KB
L2	6	90%	10 cycles	2 MB

5. Calculate the average memory access time (AMAT) in ns as a function of T.

(2 pt. missing details = no mark)

$$\text{AMAT} = \text{L1 hit time} + (\text{L1 miss rate} \times \text{L2 hit time} + (\text{L2 miss rate} \times \text{L2 miss penalty in ns}))$$
$$\text{AMAT} = 3 + 0.20 \times (6 + (0.10 \times 10 \times T))$$
$$\text{AMAT} = 3 + 1.2 + 0.2T$$
$$\text{AMAT} = 4.2 + 0.2T \text{ ns}$$

6. If $T = 0.5$ ns, calculate:
   a. The AMAT in nanoseconds.
   b. The AMAT in cycles.

$AMAT = 4.2 + 0.2 \times 0.5 = 4.2 + 0.1 = 4.3$ ns

(1 pt. missing details = no mark)

In cycles: $4.3 \div 0.5 = 8.6$ cycles