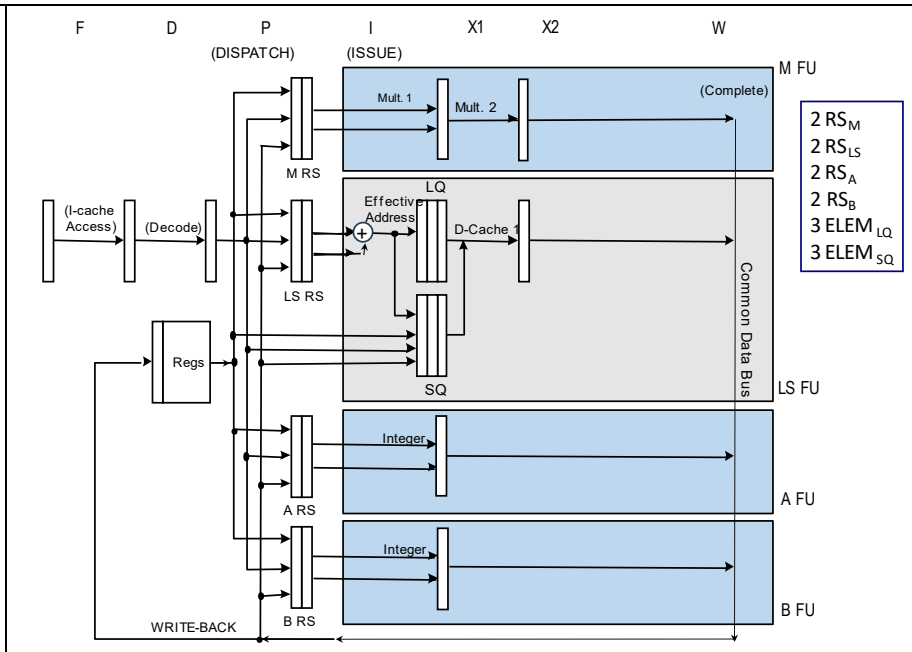
**HIGH PERFORMANCE COMPUTER ARCHITECTURE midterm exam 30-10-2023** **MATR.NO.**_____

**SURNAME**_____

**FIRST NAME**_____

1) (POINTS 27/30) Consider a **single-dispatch (1 instruction per cycle)** processor using Tomasulo's algorithm to perform the dynamic scheduling of instructions on the pipeline shown in the following figure. This pipeline is executing the following program, which performs a search within a vector (initially, R1=0).

```
etic: LW   R2, 0(R1)     ; read Xi
      MULI R2, R2, 3     ; multiplies Xi by 3
      SW   R2, 0(R1)     ; write Xi
      ADDI R1, R1, 4     ; update  R1
      BNE  R2, R0, etic  ; continue to loop if
false
```




Working hypothesis:

- the loop executes speculatively in terms of direction (always taken) and regarding the branch condition; high-performance fetch breaks after fetching a branch
- the issue stage (I) calculates the address of the actual read/write and push it into load/store queues; only 1 instruction is issued per cycle
- **reads require 2 clock cycles**; writes take 0 cycles (they are written in a write-buffer + split-cache)
- when accessing memory (M), **writes** have precedence over reads and must be executed in-order
- **there is a single CDB**
- dispatch stage (P) and complete stage (W) require 1 clock cycle
- **ASSUME** that the reservation stations could be freed right before the start of issue phase (therefore extending the duration of P stage)
- only 1 instruction is committed (C stage) per cycle
- there are separated integer units: one for the calculation of the actual address, one for arithmetic and logical operations, one of the integer multiplication and one for the evaluation of the branch condition, as illustrated in this table:

| Type of Functional Unit | No. of Functional Units | Cycles for stage X | No. of reservation stations |
|---|---|---|---|
| LS: Integer (effective addr.) | 1 | 1 | 2 |
| A: Integer (op. A-L) | 1 | 1 | 2 |
| B: Integer (branch calc.) | 1 | 1 | 2 |
| M: Integer Multiplication | 1 | **2** | 2 |

- the functional units TAKE advantage of pipelining techniques internally
- the load queue has 3 slots; the store queue has 3 slots (writes wait for the operand in the store queue, i.e., in the execution stage)

Complete the following chart until the end of the FOURTH iteration of the above code fragment in the case of dynamic scheduling with speculation. Also add the instruction that occupies a certain reservation station (one of the 8) as indicated:

| Instr. No.. | Instruction name | ALU RS1 | ALU RS2 | LS RS1 | LS RS2 | BU RS1 | BU RS2 | MU RS1 | MU RS2 | P: disPatch (clock) | I+X:Issue+Exec (start-stop) | M: MEM.ACCESS (start-stop) | W: CDB-write (clock) | C: Commit (clock) | Comments |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I01 | **LW R2,0(R1)** | | | I01 1-1 | | | | | | 1 | 2-2 | 3-4 | 5 | 6 | |
| ... | ... | | | | | | | | | | | | | | |
| ... | ... | | | | | | | | | | | | | | |

2) (POINTS 3/30) On a Linux system, write the SINGLE command line to perform at the BASH shell prompt the following operation (please note that no intermediate files should be used):

- Print on the standard output the lines of all files containing both “gh” and “ab” and whose filename contains “fil” in any position of the filename.

## EXERCIZE 1

| Instr. No.. | Instruction name | ALU RS1 (start-stop) | ALU RS2 (start-stop) | LS RS1 (start-stop) | LS RS2 (start-stop) | BU RS1 (start-stop) | BU RS2 (start-stop) | MU RS (start-stop)1 | MU RS2 (start-stop) | P: Dispatch (clock) | I+X: Issue (start-stop) | MEM. ACC. (start-stop) | W: CDB-write (clock) | C: Commit (clock) | Comments |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I01 | LW R2,0(R1) | | | I01 1-1 | | | | | | 1 | 2-2 | 3-4 | 5 | 6 | |
| I02 | MULI R2,R2,3 | | | | | | | I02 2-5 | | 2 | 6-7 | -- | 8 | 9 | I waits R2 from 1/LW |
| I03 | SW R2,0(R1) | | | | I03 3-3 | | | | | 3 | 4-4 | 10 | -- | 11 | M waits R2 from 1/MULI; M waits mem |
| I04 | ADDI R1,R1,4 | I04 4-4 | | | | | | | | 4 | 5-5 | -- | 6 | 12 | |
| I05 | BNE R2,R0,etic | | | | | I05 5-8 | | | | 5 | 9 | -- | -- | 13 | I waits R2 from 1/MULI |
| I06 | LW R2,0(R1) | | | I06 6-6 | | | | | | 6 | 7 | 8-9 | 10 | 14 | I waits R1 from 1/ADDI |
| I07 | MULI R2,R2,3 | | | | | | | I07 7-10 | | 7 | 11-12 | -- | 13 | 15 | I waits R2 from 2/LW |
| I08 | SW R2,0(R1) | | | I08 8-9 | | | | | | 8 | 10 | 14 | -- | 16 | I waits R1 from 1/ADDI; I waits issue logic; M waits R2 |
| I09 | ADDI R1,R1,4 | I09 9-11 | | | | | | | | 9 | 12 | -- | 14 | 17 | I waits R1 from 1/ADDI; I waits issue logic; |
| I10 | BNE R2,R0,etic | | | | | | I10 10-13 | | | 10 | 14 | -- | -- | 18 | I waits R2 from 2/MULI; I waits issue logic; |
| I11 | LW R2,0(R1) | | | I11 11-14 | | | | | | 11 | 15 | 16-17 | 18 | 19 | I waits issue logic; I waits R1 from 2/ADDI; |
| I12 | MULI R2,R2,3 | | | | | | | I12 12-18 | | 12 | 19-20 | -- | 21 | 22 | I waits R2 from 3/LW |
| I13 | SW R2,0(R1) | | | | I13 13-15 | | | | | 13 | 16 | 23 | -- | 24 | I waits R1; I waits issue logic; M waits R2; M waits mem |
| I14 | ADDI R1,R1,4 | | I14 14-16 | | | | | | | 14 | 17 | -- | 18 | 25 | I waits issue logic; |
| I15 | BNE R2,R0,etic | | | | | I15 15-21 | | | | 15 | 22 | -- | -- | 26 | I waits R2 from 3/MULI I waits issue logic; |
| I16 | LW R2,0(R1) | | | I16 16-19 | | | | | | 16 | 20 | 21-22 | 23 | 27 | I waits R1; I waits issue logic; |
| I17 | MULI R2,R2,3 | | | | | | | | I17 17-23 | 17 | 24-25 | -- | 26 | 28 | I waits R2 from 4/LW; I waits issue logic; |
| I18 | SW R2,0(R1) | | | | I18 18-20 | | | | | 18 | 21 | 27 | -- | 29 | I waits R1; I waits issue logic; M waits R2; |
| I19 | ADDI R1,R1,4 | I19 19-22 | | | | | | | | 19 | 23 | -- | 24 | 30 | I waits issue logic; |
| I20 | BNE R2,R0,etic | | | | | | I20 20-25 | | | 20 | 27 | -- | -- | 31 | I waits R2 from 4/MULI |

## EXERCIZE 2

The requested command line is:

```
cat "*fil*"|grep "ab"|grep "gh"
```