| | |
|---|---|
| **Surname (COGNOME)** | **SOLUTION** |
| **Name** | |
| **POLIMI Personal Code** | |
| **Signature** | |

**Politecnico di Milano, 15 February, 2024**

# Course on Advanced Computer Architectures

Prof. C. Silvano

| | | |
|---|---|---|
| **EX1** | **( 6 points)** | |
| **EX2** | **( 4 points)** | |
| **EX3** | **( 3 points)** | |
| **EX4** | **( 2 points)** | |
| **Q1** | **( 5 points)** | |
| **Q2** | **( 5 points)** | |
| **QUIZZES** | **( 8 points)** | |
| **TOTAL** | **(33 points)** | |

## EXERCISE 1 – Tomasulo (6 points)

*Please consider one iteration of the following assembly loop code:*

```
LOOP:  lw $t1,VECTA($t0)
I2:    lw $t2,VECTB($t0)
I3:    add $t3,$t1,$t2
I4:    add $t4,$t3,$t3
I5:    sw $t3,VECTC($t0)
I6:    sw $t4,VECTD($t0)
I7:    addi $t0,$t0,4
I8:    bne $t0,$t5,LOOP
```

to be executed on a ***CPU1*** with dynamic scheduling based on **TOMASULO algorithm** with all cache HITS, a single Common Data Bus and:

- *2 RESERVATION STATIONS **(RS1, RS2)** for 2 LOAD/STORE units **(LDU1, LDU2)** with latency **6***
- *2 RESERVATION STATIONS **(RS3, RS4)** for 2 ALU/BR FUs **(ALU1, ALU2)** with latency **2***

1. *Please complete the following table* for ***CPU1***:

| INSTRUCTION | ISSUE | START EXEC | WRITE RESULT | Hazards Type | RSi | UNIT |
|---|---|---|---|---|---|---|
| `LOOP:lw $t1,VECTA($t0)` | 1 | 2 | 8 | None | RS1 | LDU1 |
| `I2: lw $t2,VECTB($t0)` | 2 | 3 | 9 | None | RS2 | LDU2 |
| `I3: add $t3,$t1,$t2` | | | | | | |
| `I4: add $t4,$t3,$t3` | | | | | | |
| `I5: sw $t3,VECTC($t0)` | | | | | | |
| `I6: sw $t4,VECTD($t0)` | | | | | | |
| `I7: addi $t0,$t0,4` | | | | | | |
| `I8: bne $t0,$t5,LOOP` | | | | | | |

2. *Calculate the **CPI1** :*

***CPI1*** **= _____**

Let's assume to execute the same loop iteration on a ***CPU$_2$*** with dynamic scheduling based on **TOMASULO algorithm** with all cache HITS, a single Common Data Bus and **more reservation stations** as follows:

- *4 RESERVATION STATIONS **(RS1, RS2, RS3, RS4)** for 2 LOAD/STORE units **(LDU1, LDU2)** with latency **6***
- *4 RESERVATION STATIONS **(RS5, RS6, RS7, RS8)** for 2 ALU/BR FUs **(ALU1, ALU2)** with latency **2***

3. *Please complete the following table* for ***CPU$_2$***:

| INSTRUCTION | ISSUE | START EXEC | WRITE RESULT | Hazards Type | RSi | UNIT |
|---|---|---|---|---|---|---|
| `LOOP:lw $t1,VECTA($t0)` | 1 | 2 | 8 | None | RS1 | LDU1 |
| `I2: lw $t2,VECTB($t0)` | 2 | 3 | 9 | None | RS2 | LDU2 |
| `I3: add $t3,$t1,$t2` | | | | | | |
| `I4: add $t4,$t3,$t3` | | | | | | |
| `I5: sw $t3,VECTC($t0)` | | | | | | |
| `I6: sw $t4,VECTD($t0)` | | | | | | |
| `I7: addi $t0,$t0,4` | | | | | | |
| `I8: bne $t0,$t5,LOOP` | | | | | | |

4. *Calculate the **CPI$_2$** :*

***CPI$_2$ =*** _____

5. *Calculate the **Speedup** obtained by adding more reservation stations:*

***Speedup =*** _____

**Feedback:**

| INSTRUCTION | ISSUE | START EXEC | WRITE RESULT | Hazards Type | RSi | UNIT |
|---|---|---|---|---|---|---|
| `LOOP:lw $t1,VECTA($t0)` | 1 | 2 | 8 | None | RS1 | LDU1 |
| `I2: lw $t2,VECTB($t0)` | 2 | 3 | 9 | None | RS2 | LDU2 |
| `I3: add $t3,$t1,$t2` | 3 | 10 | 12 | (RAW $t1) **RAW $t2** | RS3 | ALU1 |
| `I4: add $t4,$t3,$t3` | 4 | 13 | 15 | **RAW $t3** | RS4 | ALU2 |
| `I5: sw $t3,VECTC($t0)` | 9 | 13 | 19 | **STRUCT RS1 RAW $t3** | RS1 | LDU1 |
| `I6: sw $t4,VECTD($t0)` | 10 | 16 | 22 | **RAW $t4** | RS2 | LDU2 |
| `I7: addi $t0,$t0,4` | 13 | 14 | 16 | **STRUCT RS3** | RS3 | ALU1 |
| `I8: bne $t0,$t5,LOOP` | 16 | 17 | 19 | **STRUCT RS4** (RAW $t0) | RS4 | ALU2 |

***$CPI_1$ = # clock cycles / IC = 22 / 8 = 2,75***

| INSTRUCTION | ISSUE | START EXEC | WRITE RESULT | Hazards Type | RSi | UNIT |
|---|---|---|---|---|---|---|
| `LOOP:lw $t1,VECTA($t0)` | 1 | 2 | 8 | None | RS1 | LDU1 |
| `I2: lw $t2,VECTB($t0)` | 2 | 3 | 9 | None | RS2 | LDU2 |
| `I3: add $t3,$t1,$t2` | 3 | 10 | 12 | (RAW $t1) **RAW $t2** | RS5 | ALU1 |
| `I4: add $t4,$t3,$t3` | 4 | 13 | 15 | **RAW $t3** | RS6 | ALU2 |
| `I5: sw $t3,VECTC($t0)` | 5 | 13 | 19 | **RAW $t3** | RS3 | LDU1 |
| `I6: sw $t4,VECTD($t0)` | 6 | 16 | 22 | **RAW $t4** | RS4 | LDU2 |
| `I7: addi $t0,$t0,4` | 7 | 13 | 16 | **STRUCT ALU1 + CDB** | RS7 | ALU1 |
| `I8: bne $t0,$t5,LOOP` | 8 | 17 | 19 | **RAW $t0** | RS8 | ALU2 |

***$CPI_2$ = # clock cycles / IC = 22 / 8 = 2,75***

*There is **no speedup** in adding more reservation stations.*

*In Tomasulo, the potential WAW and WAR hazards are already solved by Register Renaming*

*Only the hazards that have caused the introduction of some stalls are reported in the table, while other potential hazards are either omitted or put into brackets.*

## EXERCISE 2 on REORDER BUFFER (4 points)

*Let's consider the following assembly loop where registers* **$R1** *and* **$R2** *are initialized at 0 and 40 respectively:*

```
LOOP:  LD $F2, 0 ($R1)
I2:    ADDD $F4, $F2, $F2
I3:    SD $F4, 0 ($R1)
I4:    ADDI $R1, $R1, 4
I5:    BNE $R1, $R2, LOOP # branch predicted ad taken
```

- *How many loop iterations? 10 iterations*

---

*Then execute the loop by the* ***Speculative Tomasulo*** *architecture with an* ***8-entry ROB*** *and:*

- *4 Load Buffers (Load1, Load2, Load3, Load4);*
- *2 FP Reservation Stations (FP1, FP2)*
- *2 Integer Reservation Stations (Int1, Int2)*

- *Please update the following ROB and Rename Tables until the ROB becomes* ***full*** *while the first instruction is still in execution due to a cache miss (*)*

***ROB Table***

| ROB# | Instruction | Dest. | Resource Allocation | Ready /Status | Spec. | |
|---|---|---|---|---|---|---|
| ROB0 | LOOP: LD $F2, 0 ($R1) | $F2 | Load1 | No, exec. (*) | No | HEAD |
| ROB1 | I2: ADDD $F4, $F2, $F2 | $F4 | FP1 | No, issued | No | |
| ROB2 | I3: SD $F4, 0 ($R1) | Mem | -- | No, issued | No | |
| ROB3 | I4: ADDI $R1, $R1, 8 | $R1 | Int1 | No, issued | No | |
| ROB4 | I5: BNE $R1, $R2, LOOP | -- | Int2 | No, issued | No | |
| ROB5 | LOOP: LD $F2, 0 ($R1) | $F2 | Load 2 | No, issued | Yes | |
| ROB6 | I2: ADDD $F4, $F2, $F2 | $F4 | FP2 | No, issued | Yes | |
| ROB7 | I3: SD $F4, 0 ($R1) | Mem | -- | No, issued | Yes | |

***Rename Table:***

| Reg# | ROB# |
|---|---|
| $F0 | -- |
| $F2 | ~~ROB0,~~ ROB5 |
| $F4 | ~~ROB1,~~ ROB6 |
| $F6 | -- |
| $F8 | -- |

*Please complete the unrolled version of the loop with unrolling factor 2:*

```
LOOP: LD $F2, 0 ($R1)
I2:   ADDD $F4, $F2, $F2
I3:   SD $F4, 0 ($R1)
I4:   LD $F6, 4 ($R1)
I5:   ADDD $F8, $F6, $F6
I6:   SD $F8, 4 ($R1)
I7:   ADDI $R1, $R1, 8
I8:   BNE $R1, $R2, LOOP
```

- *How many loop iterations? 5 iterations*

---

*Then execute the unrolled version of the loop by the **Speculative Tomasulo** architecture with an **8-entry ROB** and:*

- *4 Load Buffers (Load1, Load2, Load3, Load4);*
- *2 FP Reservation Stations (FP1, FP2)*
- *2 Integer Reservation Stations (Int1, Int2)*

- *Please update the following ROB and Rename Tables until the ROB becomes full while the first instruction is still in execution due to a cache miss:*

***ROB Table***

| ROB# | Instruction | Dest. | Resource Allocation | Ready /Status | Spec. | |
|---|---|---|---|---|---|---|
| ROB0 | LOOP:LD $F2, 0 ($R1) | $F2 | Load1 | No, exec. (*) | No | HEAD |
| ROB1 | I2: ADDD $F4, $F2, $F2 | $F4 | FP1 | No, issued | No | |
| ROB2 | I3: SD $F4, 0 ($R1) | Mem | -- | No, issued | No | |
| ROB3 | I4: LD $F6, 4 ($R1) | $F6 | Load2 | No, issued | No | |
| ROB4 | I5: ADDD $F8, $F6, $F6 | $F8 | FP2 | No, issued | No | |
| ROB5 | I6: SD $F8, 4 ($R1) | Mem | -- | No, issued | No | |
| ROB6 | I7: ADDI $R1, $R1, 8 | $R1 | Int1 | No, issued | No | |
| ROB7 | I8: BNE $R1, $R2, LOOP | -- | Int2 | No, issued | No | |

***Rename Table:***

| Reg# | ROB# |
|---|---|
| $F0 | -- |
| $F2 | ROB0 |
| $F4 | ROB1 |
| $F6 | ROB3 |
| $F8 | ROB4 |

## EXERCISE 3 – CACHE MEMORIES (3 points)

Let's consider 32-block main memory with a **fully-associative** 4-block cache and **LRU** cache replacement policy. The addresses are expressed as:

Memory Block Addresses: $[0, 1, 2, \ldots\ldots\ldots\ldots 31]_{10}$

Cache Block Addresses: [a, b, c, d]

At cold start the cache is empty, then there is the following sequence of memory accesses.

- *Assuming the **Write Allocate** and **Write Back** cache policies, complete the following table:*

| | Type of memory access | Memory Address | HIT / MISS Type | Cache Tag | Cache Block Address | Dirty bit | Write-back in memory |
|---|---|---|---|---|---|---|---|
| 1 | read | $[12]_{10}$ | Cold start MISS | $[01100]_2$ | a | 0 | no |
| 2 | write | $[12]_{10}$ | HIT | $[01100]_2$ | a | 1 | no |
| 3 | write | $[10]_{10}$ | *Cold start MISS* | $[01010]_2$ | *b* | *1* | *no* |
| 4 | read | $[10]_{10}$ | *HIT* | $[01010]_2$ | *b* | *1* | *no* |
| 5 | read | $[26]_{10}$ | *Cold start MISS* | $[11010]_2$ | *c* | *0* | *no* |
| 6 | write | $[7]_{10}$ | *Cold start MISS* | $[00111]_2$ | *d* | *1* | *no* |
| 7 | read | $[12]_{10}$ | *HIT* | $[01100]_2$ | *a* | *1* | *no* |
| 8 | read | $[4]_{10}$ | *Conflict MISS* | $[00100]_2$ | *b* | *0* | *Yes: WB in* $[10]_{10}$ |
| 9 | write | $[26]_{10}$ | *HIT* | $[11010]_2$ | *c* | *1* | *no* |
| 10 | write | $[18]_{10}$ | *Conflict MISS* | $[10010]_2$ | *d* | *1* | *Yes: WB in* $[7]_{10}$ |
| 11 | read | $[5]_{10}$ | *Conflict MISS* | $[00101]_2$ | *a* | *0* | *Yes: WB in* $[12]_{10}$ |
| 12 | write | $[24]_{10}$ | *Conflict MISS* | $[11000]_2$ | *b* | *1* | *No* |

Calculate the Miss Rate:

*Miss Rate = Number of misses / Number of memory access = 8 / 12= 0,67*

## EXERCISE 4 – VECTOR PROCESSORS (2 points)

Let's consider the following code executed by a Vector Processor with:

- *Vector Register File composed of 32 vectors of 8 elements per 64 bits/element;*
- *Scalar FP Register File composed of 32 registers of 64 bits;*
- *One Load/Store Vector Unit with operation chaining and memory bandwidth 64 bits;*
- *One ADD/SUB Vector Unit with operation chaining;*
- *One MUL/DIV Vector Unit with operation chaining.*

```
L.V V1, RX          # Load vector from memory address RX into V1
L.V V3, RY          # Load vector from memory address RY into V3
MULVS.D V1, V1, F0  # FP multiply vector V1 to scalar F0
ADDVV.D V2, V1, V1  # FP add vectors V1 and V1
MULVS.D V2, V2, F0  # FP multiply vector V2 to scalar F0
ADDVV.D V3, V2, V3  # FP add vectors V2 and V3
S.V V3, RZ          # Store vector V3 into memory address RZ
```

*Please complete the following vector instruction scheduling:*

| | | | | |
|---|---|---|---|---|
| **0** | **1** | | **L.V V1, RX** | **7** |

*How many convoys? 3 convoys*

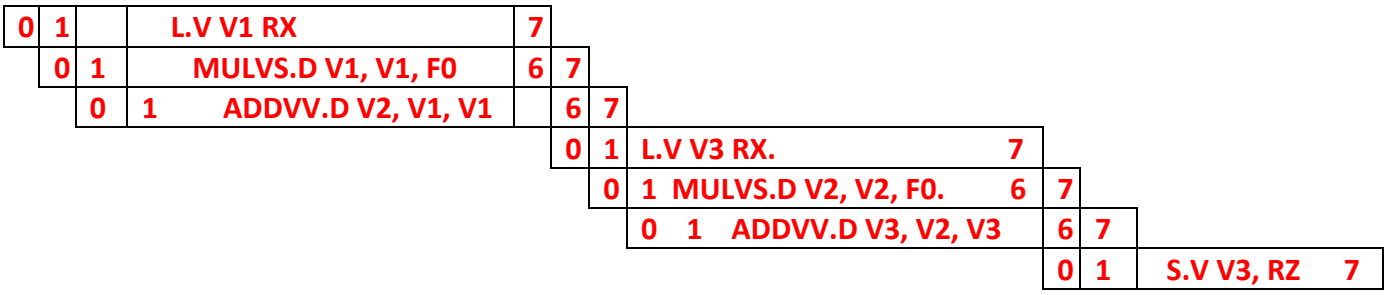
_____

*How many clock cycles to execute the code? 24 clock cycles*

_____

**Feedback:**

*Vector instruction scheduling:*




**There are 3 convoys as follows:**

1) L.V V1, RX; MULVS.D V1, V1, F0 ADDVV.D V2, V1, V1
2) L.V V3, RY MULVS.D V2, V2, F0 ADDVV.D V3, V2, V3
3) S.V V3, RZ

## QUESTION 1: VLIW ARCHITECTURES (5 points)

1. Describe the main concepts of Very Long Instruction Word processor architectures.

2. What are the main issues associated with VLIW architectures?
For each answer mark if it is either **TRUE** or **FALSE** and **motivate your answer**)

**Answer 1**: Compilers can detect parallelism only in basic blocks of the code **T** **F**

**Answer 2:** Larger code size **T** **F**

**Answer 3:** Increased HW complexity **T** **F**

**Answer 4:** Less code portability **T** **F**

**Answer 5:** Large number of registers required for register renaming **T** **F**

## QUESTION 2: MEMORY HIERARCHY (5 points)

Let's consider the following cache optimization techniques. *Answer to the following questions:*

| | **Pseudo-associativity and Way Prediction** | **Early Restart and Critical Word First** | **Introducing a Victim Cache** |
|---|---|---|---|
| *Explain the **main concepts** for each technique.* | | | |
| *What are the main effects of each technique on the **miss rate?*** | | | |
| *What are the main effects of each technique on the **miss penalty?*** | | | |

## MULTIPLE-CHOICE QUESTIONS: (8 points)

### Question 1 (format Multiple Choice – Single answer)

*In the **speculative Tomasulo architecture**, what are the hardware blocks needed to undo speculative instructions execution on mispredicted branches?*
***(SINGLE ANSWER)***
***1 point***

**Answer 1**: Reorder Buffer; **(TRUE)**

**Answer 2**: Instruction Dispatcher

**Answer 3**: Store Buffers;

**Answer 4:** Reservation Stations;

**Answer 5:** Load Buffers;

**Feedback:**
The speculative Tomasulo architecture supports speculative instructions execution and branch mispredictions by using ROB entries.

### Question 2 (format TRUE/FALSE)

*A **fine-grain multi-threading** processor* exploits more parallelism than a simultaneous multi-threading processor
**(TRUE/FALSE)**
**1 point**

**Answer**: TRUE FALSE **(X)**

### Question 3 (format TRUE/FALSE)

*A **VLIW** processor needs multiple Program Counters to load the necessary multiple instructions.*
**(TRUE/FALSE)**
**1 point**

**Answer**: TRUE FALSE **(X)**

### Question 4 (format Multiple Choice – Single answer)

Let's consider a 5-issue processor that can manage up to 5 simultaneous threads.
*What are the values of the ideal CPI and the ideal per-thread CPI?*
***(SINGLE ANSWER)***
***1 point***

**Answer 1**: Ideal CPI = 5 & Ideal per-thread CPI = 1
**Answer 2**: Ideal CPI = 1 & Ideal per-thread CPI =0.2
**Answer 3**: Ideal CPI = 0.20 & Ideal per-thread CPI = 1 **(TRUE)**
**Answer 4**: Ideal CPI = 0.25 & Ideal per-thread CPI = 5
**Answer 5**: Ideal CPI = 5 & Ideal per-thread CPI = 0.20

**Question 5 (format Multiple Choice – Single answer)**

Let's consider a (2,1) Correlating Branch Predictor with 4K total entries.
*How many Branch History Tables?*
*How many entries per BHT?*
*How many bits per entry?*
***(SINGLE ANSWER)***
***1 point***

**Answer 1:** 4 BHTs | 1K entries | 1-bit per entry **(TRUE)**
**Answer 2:** 2 BHTs | 2K entries | 1-bit per entry
**Answer 3:** 2 BHTs | 2K entries | 2-bit per entry
**Answer 4:** 1 BHT | 4K entries | 1-bit per entry

**Question 6 (format Multiple Choice – Single answer)**

*Using a 5-stage **optimized pipeline with the "early evaluation of PC"** for branch instructions, how many stalls we need to introduce to execute the following assembly code?*

```
ld $t1, AA($t6)
beq $t1, $t0, LABEL
```

**Answer 1:** 1
**Answer 2**: 2 **(TRUE)**
**Answer 3**: 3
**Answer 4**: none

***(SINGLE ANSWER)***
***1 point***

*Complete the following pipeline scheme to motivate your answer:*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| `ld $t1, AA($t6)` | IF | ID | EX | ME | WB | | | | |
| `beq $t1, $t0, LABEL` | | IF | ID stall | ID stall | ID | EX | ME | WB | |

**Question 7 (complete table format)**
***2 points***

Let's consider the following access patterns on a **4-processor** system with a direct-mapped, write-back cache with one cache block per processor and a two-cache block memory.

Assume the **MESI protocol** is used with **write-back** caches, **write-allocate,** and **write-invalidate** of other caches.

*Please COMPLETE the following table:*

| Cycle | After Operation | P0 cache block state | P1 cache block state | P2 cache block state | P3 cache block state | Mem. at bl. 0 up to date? | Mem. at bl. 1 up to date? |
|---|---|---|---|---|---|---|---|
| **1** | P0: Read Bl. 1 | Excl (1) | Invalid | Invalid | Invalid | Yes | Yes |
| **2** | P2: Read Bl. 0 | Excl (1) | Invalid | Excl (0) | Invalid | Yes | Yes |
| **3** | P3: Read Bl. 0 | Excl (1) | Invalid | Shared (0) | Shared (0) | Yes | Yes |
| **4** | P1: Write Bl. 0 | Excl (1) | Mod (0) | Invalid | Invalid | No | Yes |
| **5** | P0: Write Bl. 1 | Mod (1) | Mod (0) | Invalid | Invalid | No | No |
| **6** | P3: Read Bl. 1 | Shared(1) | Mod (0) | Invalid | Shared (1) | No | Yes |