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

Politecnico di Milano, 31 August, 2023

Course on Advanced Computer Architectures

Prof. C. Silvano

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

EXERCISE 1 – VLIW (5 points)

Let's consider the following LOOP code:

```
Loop: LD F0,0(R2)
      LD F4,0(R3)
      FMUL F0,F0,F4
      FADD F2,F2,F4
      ADDUI R2,R2,#8
      ADDUI R3,R3,#8
      SUBUI R5,R4,R2
      BNEZ R5,Loop
```

Given a **3-issue VLIW** machine with **fully pipelined functional units:**

- **1 Memory Units with 3 cycles latency**
- **1 FP ALUs with 3 cycles latency**
- **1 Integer ALU with 1 cycle latency to next Int/FP & 2 cycle latency to next Branch**

The branch is completed with 1 cycle delay slot (branch solved in ID stage). **No branch prediction.** In the Register File, it is possible to read and write at the same address at the same clock cycle.

1) *Considering one iteration of the loop, complete the following table by using the **list-based scheduling*** (do NOT introduce any software pipelining, loop unrolling and modifications to loop indexes) on the *3-issue VLIW machine including the BRANCH DELAY SLOT.* Please do not write in NOPs.

	Memory Unit	Floating Point Unit	Integer Unit
C1	`LD F0,0(R2)`		
C2			
C3			
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

2. *How long is the critical path for a single iteration?*

   _____

3. *What performance did you achieve in CPI?*

   _____

4. *What performance did you achieve in FP ops per cycles?*

   _____

5. *How much is the code efficiency?*

   _____

6. *Assuming to have **2 FLOATING POINT UNITS** instead of one, how much is the impact on CPI and code efficiency?*

_____

_____

_____

_____

_____

_____

Feedback

	Memory Unit	Floating Point Unit	Integer Unit
C1	`LD F0,0(R2)`		`ADDUI R2,R2,#8`
C2	`LD F4,0(R3)`		`ADDUI R3,R3,#8`
C3			`SUBUI R5,R4,R2`
C4			
C5		`FMUL F0,F0,F4`	`BNEZ R5,Loop`
C6		`FADD F2,F2,F4`	`(br. delay slot)`
C7			
C8			

1. *How long is the critical path for a single iteration? (*)*

   8 cycles (needed to conclude the instruction FADD F2, F2,F4)

   (*) When considering many iterations, the next LF F0 could start at cycle C8, therefore the critical path would be 7 cycles and the $CPI_{AS} = 7 / 8$

2. *What performance did you achieve in CPI?* CPI = # cycles / IC = 8 / 8 = 1

3. *What performance did you achieve in FP ops per cycles?*

   FP ops per cycles = # FP ops / # cycles = 2 / 8 = 0.25

4. *How much is the code efficiency?*

   Code efficiency = IC / (# cycles x # issues) = 8 / (8 x 3) = 8 / 24 = 1 / 3 = 0.33

5. *Assuming to have **2 FLOATING POINT UNITS** instead of one, how much is the impact on CPI and code efficiency?* FMUL and FADD can be scheduled in parallel, therefore we have a critical path for a single iteration of 7 cycles with a slightly improved CPI = 7 / 8 = 0.875 and a slightly worst code efficiency = 8 / (7 x 4) = 2 / 7 = 0.29

EXERCISE 2 – CACHE MEMORIES (5 points)

Let's consider 32-block main memory with a **2-way set associative** 8-block cache based on a ***write allocate*** with ***write-back*** protocol.

The addresses are expressed as:
Memory Address: [0, 1, 2, ……. 31] $_{10}$
Cache Address: [a, b, c, d, e, f, g, h]
and Cache Tags are expressed as binary numbers.

1) *How many sets are in the cache?*

_____

2) *Being set-associative, what is the most used block replacement policy?*

_____

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

Please complete the following table:

	Read / Write	Memory Address	HIT/MISS Type	Cache Tag	Cache Address	Set	Write in memory
1	Write	[12] $_{10}$	Cold-start Miss				
2	Read	[12] $_{10}$					
3	Read	[10] $_{10}$					
4	Write	[10] $_{10}$					
5	Write	[26] $_{10}$					
6	Read	[14] $_{10}$					
7	Write	[30] $_{10}$					
8	Write	[16] $_{10}$					
9	Read	[18] $_{10}$					
10	Read	[24] $_{10}$					

4) *How much is the Miss Rate?*

_____

Feedback

1) *How many sets are in the cache?*

The 2-way set-associative cache has 8-blocks [a, b, c, d, e, f, g, h] organized in **4 sets,** where each set contains **2 blocks** as follows: **Set_0: [a , b] Set_1 [c, d]; Set_2: [e, f] Set_ 3 [g, h]**

Memory Address Mapping:

Tags	Set_0 [a , b]	Set_1 [c, d]	Set_2 [e, f]	Set_1 [g, h]
000	0	1	2	3
001	4	5	6	7
010	8	9	10	11
011	12	13	14	15
	…	…	…	…
111	28	29	30	31

2) *Being set-associative, what is the most used block replacement policy?*

The block replacement policy in each set uses the **Least Recently Used (LRU) algorithm**.

3) *Please complete the following table:*

	Read / Write	Memory Address	HIT / MISS Type	Cache Tag	Cache Address	Set	Write in memory
1	Write	$[12]_{10}$	Cold-start Miss	$[011]_2$	a	$[0]_{10}$	No
2	Read	$[12]_{10}$	Hit	$[011]_2$	a	$[0]_{10}$	No
3	Read	$[10]_{10}$	Cold-start Miss	$[010]_2$	e	$[2]_{10}$	No
4	Write	$[10]_{10}$	Hit	$[010]_2$	e	$[2]_{10}$	No
5	Write	$[26]_{10}$	Cold-start Miss	$[110]_2$	f	$[2]_{10}$	No
6	Read	$[14]_{10}$	Conflict Miss	$[011]_2$	e	$[2]_{10}$	Yes Wr. in $M[10]_{10}$
7	Write	$[30]_{10}$	Conflict Miss	$[111]_2$	f	$[2]_{10}$	Yes Wr. in $M[26]_{10}$
8	Write	$[16]_{10}$	Cold-start Miss	$[100]_2$	b	$[0]_{10}$	No
9	Read	$[18]_{10}$	Conflict Miss	$[100]_2$	e	$[2]_{10}$	No
10	Read	$[24]_{10}$	Conflict Miss	$[110]_2$	a	$[0]_{10}$	Yes Wr. in $M[12]_{10}$

4) *How much is the Miss Rate?*

Miss Rate = Number of misses / Number of memory access = 8/10 = 0.8

EXERCISE 3 - SOFTWARE PIPELINING (2 points)

Given the following software pipelined loop, reconstruct the original (non-pipelined) code:

```
START_UP: LD $F0, 0 ($R1)
          LD $F2, 0 ($R2)
          FADD $F4, $F0, $F0
          FADD $F6, $F2, $F2
          LD $F0, 8 ($R1)
          LD $F2, 8 ($R2)

LOOP: SD $F4, 0 ($R1)
      SD $F6, 0 ($R2)
      FADD $F4, $F0, $F0
      FADD $F6, $F2, $F2
      LD $F0, 16 ($R1)
      LD $F2, 16 ($R2)
      ADDUI $R1, $R1, 8
      ADDUI $R2, $R2, 8
      BNE $R1, $R3, LOOP

FINISH-UP: SD $F4, 8 ($R1)
           SD $F6, 8 ($R2)
           FADD $F4, $F0, $F0
           FADD $F6, $F2, $F2
           SD $F4, 16 ($R1)
           SD $F6, 16 ($R2)
```

Feedback:

```
LOOP: LD $F0, 0 ($R1)
      LD $F2, 0 ($R2)
      FADD $F4, $F0, $F0
      FADD $F6, $F2, $F2
      SD $F4, 0 ($R1)
      SD $F6, 0 ($R2)
      ADDUI $R1, $R1, 8
      ADDUI $R2, $R2, 8
      BNE $R1, $R3, LOOP
```

EXERCISE 4 – VECTOR PROCESSORS (3 points)

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

- *Vector Register File composed of 32 vectors of 64 elements per 8 bits/element;*
- *Scalar FP Register File composed of 32 registers of 8 bits;*
- *One Load/Store Vector Unit with operation chaining and memory bandwidth 8 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 V2, RY          # Load vector from memory address RY into V2
MULVS.D V1, V1, F0  # FP multiply vector V1 to scalar F0
ADDVV.D V2, V2, V1  # FP add vectors V1 and V2
MULVS.D V3, V2, F1  # FP multiply vector V2 to scalar F1
S.V V3, RZ          # Store vector V3 into memory address RZ
```

1) *How many convoys?* _____

2) *Please complete the following scheme:*

0	LV V1	63

3) *How many clock cycles to execute the code?*

_____

4) *In case of NO OPERATION CHAINING, how many convoys and clock cycles would be needed to execute the code?*

_____

Feedback:

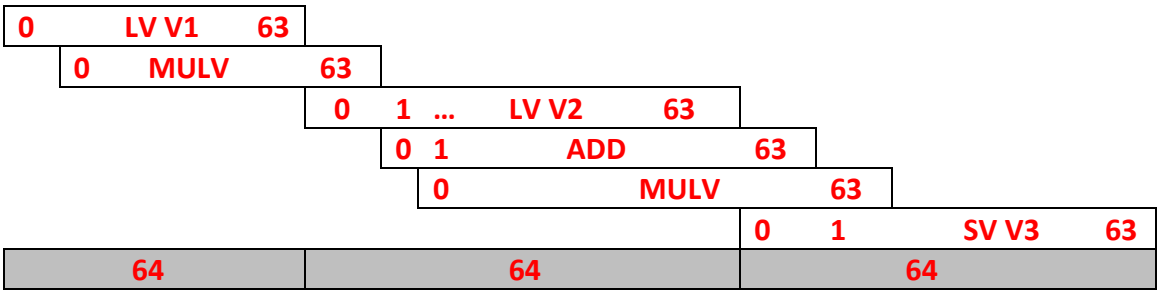
1) *How many convoys?*

   There are 3 convoys as follows:

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

MULVS.D V1, V1, F0
ADDVV.D V2, V2, V1
MULVS.D V3, V2, F1

2) *Please complete the following scheme:*



3) *How many clock cycles to execute the code?*

   3 X 64 = 192 clock cycles

4) *In case of NO OPERATION CHAINING, how many convoys and clock cycles would be needed to execute the code?*

   There are 5 convoys and 5 X 64 = 330 clock cycles as follows:

1) L.V V1, RX;
2) L.V V2, RY
3) ADDVV.D V2, V2, V1
4) MULVS.D V3, V2, F1
5) S.V V3, RZ

MULVS.D V1, V1, F0

QUESTION 1: PROCESSOR PIPELINING (5 points)

1) *What is the difference between a dependency and a hazard?*	
2) *What is the difference between a data dependency and a control dependency?*	
3) *What is the difference between a true data dependency and a name dependency?*	
4) *Which of the WAW, RAW and WAW hazards are generated by true data dependencies and by name dependencies?*	

5) *For each statement of the assembly code, please complete the following table by reporting data dependencies (corresponding to RAW hazards) and name dependencies (WAR/WAW hazards):*

Assembly Code:	Data dependencies (RAW Hazards)	Name dependencies (WAR / WAW hazards)
`Loop: LD F0, 0(R1) # S1`	None	None
`LD F2, 0(R3) # S2`	None	None
`FMUL F0, F0, F2 # S3`	RAW F0 w.S1; RAW F2 w.S2	. . . . .
`FADD F2, F0, F2 # S4`	. . . . .	. . . . .
`SD F2, 0(R3) # S5`	. . . . .	. . . . .
`ADDUI R1, R1, 8 # S6`	. . . . .	. . . . .
`ADDUI R3, R3, 8 # S7`	. . . . .	. . . . .
`BNE R1, R6, Loop # S8`	. . . . .	. . . . .

6) *Use the above Loop code to make an practical example on how* ***Register Renaming*** *can be applied to avoid name dependences:*

```
loop:LD   F0, 0(R1)      # S1
     LD   F2, 0(R3)      # S2
     . . . . . .         # S3
     . . . . . .         # S4
     . . . . . .        .# S5
     . . . . . .       ..# S6
     . . . . . .       ..# S7
     . . . . . .        .# S8
```

Feedback:

Assembly Code:	Data dependencies (RAW Hazards)	Name dependencies (WAR / WAW hazards)
`Loop: LD F0, 0(R1) # S1`	None	None
`LD F2, 0(R3) # S2`	None	None
`FMUL F0, F0, F2 # S3`	RAW F0 w.S1; RAW F2 w.S2	WAW F0 w. S1
`FADD F2, F0, F2 # S4`	RAW F0 w.S3; RAW F2 w.S2	WAW F2 w. S2 WAR F2 w.S3
`SD F2, 0(R3) # S5`	RAW F2 w.S4	None
`ADDUI R1, R1, 8 # S6`	None	WAR R1 w. S1
`ADDUI R3, R3, 8 # S7`	NONE	WAR R3 w. S2,S5
`BNE R1, R6, Loop # S8`	RAW R1 w.S6;	None

Example on how register renaming can be applied to avoid name dependences:

```
loop:LD   F0, 0(R1)      # S1
     LD   F2, 0(R3)      # S2
     FMUL F4, F0, F2     # S3 WAW F0 solved
     FADD F6, F4, F2     # S4 WAW/WAR F2 solved
     SD   F6, 0(R3)      # S5
     ADDUI R1, R1, 8     # S6
     ADDUI R3, R3, 8     # S7
     BNE  R1, R6, loop   # S8
```

QUESTION 2: BRANCH PREDICTION (5 points)

Let's consider the two types of branch prediction techniques: *static and dynamic.*

Answer to the following questions:

	Static Branch Prediction	**Dynamic Branch Prediction**
1) Explain the main concepts for each technique.		
2) What are the main benefits for each technique?		

3) What are the main drawbacks for each technique?		
4) What are the hardware resources needed to implement each technique in a pipelined processor?		

MULTIPLE-CHOICE QUESTIONS: (8 points)

Question 3 (format Multiple Choice – Single answer)

Please consider the following time schedule to be executed on a CPU with dynamic scheduling based on **TOMASULO algorithm** with all cache HITS, a single Common Data Bus and:

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

(SINGLE ANSWER)
2 points

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT
I1: lw $t2,VECTA($t6)	1	2	8
I2: lw $t3,VECTB($t6)	2	3	9
I3: lw $t4,VECTC($t6)	3	9	15
I4: addi $t2,$t2,k	4	10	12
I5: sw $t2,VECTA($t6)	5	13	19
I6: add $t4,$t3,$t4	6	16	18
I7: sw $t4,VECTC($t6)	7	19	25

Where is the error in the above schedule?

Answer 1: Line I3 must be: | 3 | 4 | 10 |
Answer 3: Line I4 must be: | 4 | 9 | 11 |
Answer 3: Line I5 must be: | 5 | 6 | 12 |
Answer 4: Line I6 must be: | 6 | 14 | 16 |
Answer 5: Line I7 must be: | 9 | 19 | 25 | **(TRUE)**

Feedback:

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT	Hazards Type	RSi	UNIT
I1: lw $t2,VECTA($t6)	1	2	8	None	RS1	LDU1
I2: lw $t3,VECTB($t6)	2	3	9	None	RS2	LDU2
I3: lw $t4,VECTC($t6)	3	9	15	**STRUCT LDU1**	RS3	LDU1
I4: addi $t2,$t2,k	4	10	12	**RAW $t2 + RF read**	RS5	ALU1
I5: sw $t2,VECTA($t6)	5	13	19	**RAW $t2**	RS4	LDU2
I6: add $t4,$t3,$t4	6	16	18	**RAW $t4**	RS6	ALU2
I7: sw $t4,VECTC($t6)	9	19	25	**STRUCT RS1 + RAW $t4**	RS1	LDU1

Question 4 (format Multiple Choice – Multiple answers)

*How does a MESI write-invalidate write-back protocol manage a **Write Hit** on an Exclusive cache block?*
(MULTIPLE ANSWERS)
2 points

Answer 1: The status of the cache block becomes Modified; **(TRUE)**
Answer 2: The cache block is updated from the memory;
Answer 3: An invalidate is broadcasted on the bus to the other copies of the block;
Answer 4: The cache block is updated from another cache;
Answer 5: The cache block is update with the new value **(TRUE)**

Feedback:
On a Write Hit on an Exclusive cache block means that the block is clean and there are no other copies of the block in other caches. So there is no need to send an invalidate signal on the snooping bus to other caches. We have that:

- *The processor's cache is updated with the new data values.*
- *The status of the cache block becomes Modified*

Question 5 (format Multiple Choice – Single answer)

Let's consider a directory-based protocol for a distributed shared memory system with 4 Nodes (N0, N1, N2, N3) where we consider the block B1 in the directory of N1:

Directory N1 Block B1 | State: Shared | Sharer Bits: 0011 |

*Which are the State and the Sharer Bits of the block **B1** in **N1** after this sequence:*
Read Miss B1 from local cache N1;
Read Miss B1 from local cache N0;
(SINGLE ANSWER)
1 point

Answer 1: Directory N1 Block B1 | State: Shared | Sharer Bits: 1100 |
Answer 2: Directory N1 Block B1 | State: Modified | Sharer Bits: 0100 |
Answer 3: Directory N1 Block B1 | State: Modified | Sharer Bits: 1000 |
Answer 4: Directory N1 Block B1 | State: Shared | Sharer Bits: 1111 | **(TRUE)**

Question 6 (format Multiple Choice – Multiple answers)

What characteristics of the GPU architecture determine its high throughput?
(MULTIPLE ANSWERS)
1 point
Answer 1: The latency of GPU arithmetic units is lower than CPU ALUs;
Answer 2: A GPU has a massive number of parallel processing elements; **(TRUE)**
Answer 3: A GPU does not have sophisticated control units; **(TRUE)**
Answer 4: Switching threads on a GPU is less expensive than on a CPU; **(TRUE**

Question 7 (format True/False)

To make dynamic loop unrolling in the speculative Tomasulo architecture, the register renaming can be done by using the Reorder Buffer entries without the help of the compiler.
(True/False)
1 point

Answer 1: True (**TRUE**)

Answer 2: False

Feedback:
The speculative Tomasulo architecture supports Register Renaming by using ROB entries to eliminate WAR and WAW hazards and enable dynamic loop unrolling.

Question 8 (format Multiple Choice – Single answer)

*In the speculative Tomasulo architecture, what is done at the **Commit stage** when a not-speculative instruction is ready at the head of ROB?*

(SINGLE ANSWER)
1 point

Answer 1: Update destination into RF (or memory in case of store) with ROB result and free ROB entry; **(TRUE)**

Answer 2: Send the instruction result on CDB to update the ROB entry and awaiting Reservation Stations;

Answer 3: Wait for source operand values from CDB to compute the destination result;

Answer 4: Write the instruction result into the inter-stage forwarding registers;

Answer 5: Flush all ROB entries because of the branch misprediction;