Surname (COGNOME)	SOLUTION
Name	
POLIMI Personal Code	
Signature	

Politecnico di Milano, 19 June, 2024

Course on Advanced Computer Architectures

Prof. C. Silvano

EX1	(5 points)	
EX2	(5 points)	
EX3	(5 points)	
Q1	(4 points)	
Q2	(6 points)	
QUIZZES	(8 points)	
TOTAL	(33 points)	

EXERCISE 1 – SCOREBOARD (5 points)

1. *Let's consider the following assembly code containing multiple types of intra-loop dependences. Complete the following table by inserting all types of true-data-dependences, anti-dependences and output dependences for each instruction:*

I#	TYPE OF INSTRUCTION	**ANALYSIS OF DEPENDECES:** 1. True data dependence with I# for $Fx 2. Anti-dependence with I# for $Fy 3. Output-dependence with I# for $Fz
I0	`LOOP: LD $F2, 0($R1)`	None
I1	`LD $F4, 0($R2)`	None
I2	`FADD $F4,$F2,$F4`	True data dependence with I0 for $F2 True data dependence with I1 for $F4 Output dependence with I1 for $F4
I3	`SD $F4,0($R1)`	True data dependence with I2 for $F4
I4	`ADDUI $R1,$R1,4`	Anti dependence with I0 for $R1 Anti dependence with I3 for $R1
I5	`ADDUI $R2,$R2,4`	Anti dependence with I1 for $R2
I6	`BNE $R1,$R3,LOOP`	True data dependence with I4 for $R1

2. *Schedule the code on a CPU with dynamic scheduling based on* ***OPTIMIZED SCOREBOARD*** *with the following assumptions:*
   - 2 LOAD/STORE Units (LDU1, LDU2) with latency 3 cycles
   - 1 FP Unit (FPU1) with latency 4 cycles
   - 2 ALU/BR Units (ALU1, ALU2) with latency 1 cycle
   - Register File with 2 read ports and 1 write port
   - **Check for WAR and WAW hazards postponed to the WRITE BACK phase**
   - **Forwarding**

Feedback

INSTRUCTION	ISSUE	READ OPs	EXEC COMPL.	WRITE BACK	Hazards Type	UNIT
`LOOP:LD $F2, 0($R1)`	1	2	5	6	--	LDU1
`LD $F4, 0($R2)`	2	3	6	7	--	LDU2
`FADD $F4,$F2,$F4`	3	**7**	11	12	**RAW $F4 I1 solved by forw.** (Check RAW $F2 I0 in WB ok) (Check WAW $F4 I1 in WB ok)	FPU1
`SD $F4,0($R1)`	7	**12**	15	16	**Check STRUCT LDU1 in ISSUE** **RAW $F4 I2 solved by forw.**	LDU1
`ADDUI $R1,$R1,4`	8	9	10	**13**	**WAR $R1 with I3** (Check WAR $R1 I0 in WB ok)	ALU1
`ADDUI $R2,$R2,4`	9	10	11	**14**	**Check STRUCT RF write** (Check WAR $R2 I1 in WB ok)	ALU2
`BNE $R1,$R3,LOOP`	**14**	15	16	17	**Check STRUCT ALU1 in ISSUE** (Check RAW $R1 I5 ok)	ALU1

Calculate the CPI: CPI = (#clock cyles / IC) = 17/7 = 2.43

EXERCISE 2 – VLIW SCHEDULING (5 points)

Let's consider the following LOOP code, where $**Ri** are integer registers and **Fi** are floating-point registers.

```
L1: ADDI R1, R2, R3
    SUBI R4, R1, R5
    MULI R6, R7, R8
    FADD F1, F2, F3
    FSUB F4, F1, F5
    LD R9, MEM[R4]
    SD R6, MEM[R10]
    SD F4, MEM[R11]
    BNEZ R9, L1
```

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/L/S & 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. *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 *4-issue VLIW machine including the BRANCH DELAY SLOT.* Please do not write in NOPs.

	Memory Unit	Floating Point Unit	Integer Unit
C1		FADD F1, F2, F3	ADDI R1, R2, R3
C2			SUBI R4, R1, R5
C3	LD R9, MEM[R4]*		MULI R6, R7, R8
C4	SD R6, MEM[R10]*	FSUB F4, F1, F5	
C5			
C6			BNEZ R9, L1
C7	SD F4, MEM[R11]		(br. delay slot)
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

(*) The pair of LD and SD instructions can be anticipated by one cycle without impacting the critical path.

1. How long is the critical path? ______________________________

   9 cycles (to complete the SD)

2. What performance did you achieve in CPIas?

   ____

   CPIas = #cycles / IC = 7 / 9 = 0.78

   Note that the CPI of one single iteration is CPI = #cycles / IC = 9 / 9 = 1

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

   ____

   FPops / cycles = 2 / 9

4. How much is the code efficiency?

   ____

   Code_eff = IC / (#cycles * #issues) = 9 / (9 x 3) = 1 / 3 = 0.3 (for one single iteration)

5. Assuming to have **2 INTEGER UNITS** how much is the impact on CPI and code efficiency?

The critical path is still the same, therefore the CPI is not impacted.

The code efficiency becomes worst:

Code_eff = IC / (#cycles * #issues) = 9 / (9 x 4) = 1 / 4 = 0.25

EXERCISE 3 – MESI PROTOCOL (5 points)

Let's consider the following access patterns on a **2-processor** system with a direct-mapped, write-back cache with one cache block per processor and a two-cache block memory.

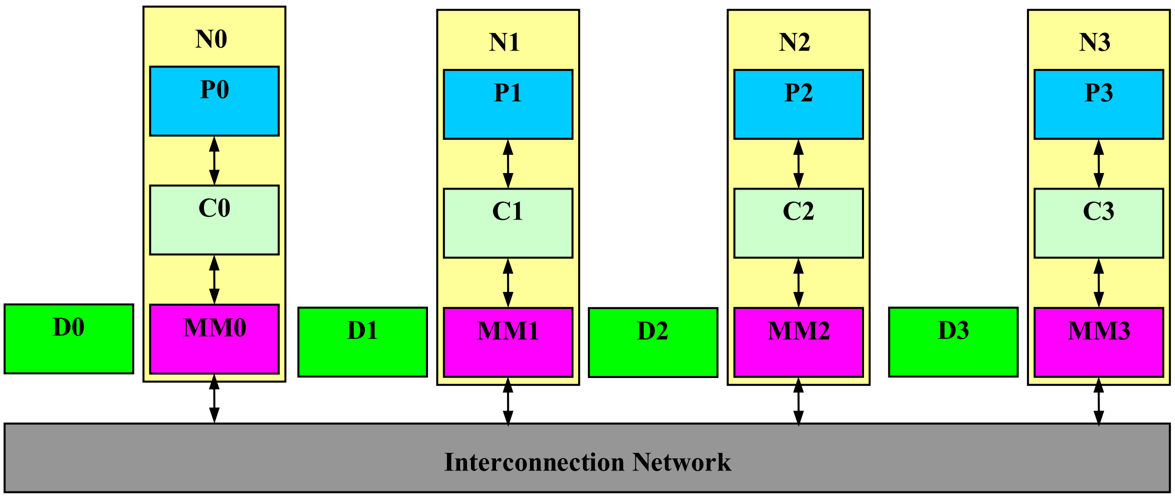
Assume the **MESI protocol** is used, with **write-back** caches, **write-allocate,** and **write-invalidate** of other caches.

Please complete the following table:

Cycle	**After Operation**	**Hit/ Miss**	**P0 cache block state**	**P1 cache block state**	**Memory at bl. 0 up to date?**	**Memory at bl. 1 up to date?**	**Write Invalidation sent?**
0	**P0: Read Bl. 1**	Miss	Exclusive(1)	Invalid	Yes	Yes	No
1	**P1: Read Bl. 0**	Miss	Exclusive(1)	Exclusive(0)	Yes	Yes	No
2	**P0: Write Bl. 1**	Hit	Modified (1)	Exclusive(0)	Yes	No	No
3	**P0: Write Bl. 0**	Miss	Modified (0)	Invalid	No	Yes	Yes
4	**P1: Read Bl. 1**	Miss	Modified (0)	Exclusive(1)	No	Yes	No
5	**P1: Write Bl. 1**	Hit	Modified (0)	Modified (1)	No	No	No
6	**P0: Read Bl. 1**	Miss	Shared (1)	Shared (1)	Yes	Yes	No
7	**P0: Write Bl. 1**	Hit	Modified (1)	Invalid	Yes	No	Yes
8	**P1: Write Bl. 1**	Miss	Invalid	Modified (1)	Yes	No	Yes

QUESTION 1: DIRECTORY-BASED PROTOCOL (*4 points)*

Let's consider a directory-based protocol for a distributed shared memory system with 4 nodes (N0, N1, N2, N3) where: **Directory N1 Block B1 | State: Modified | Sharer Bits: 0001**



1. **After the message Write Miss on B1 sent by node N2,** please answer to the following questions:

Which is the home node?	**N1**
Which is the local node?	**N2** is the local node sending the request
Which is the remote node?	**N3** is the remote node where there is the most updated copy of the block B1
What is the message sent between the home node and the remote note?	**Fetch/Invalidate** message is sent from home N1 to remote N3 (past owner) to ask to fetch the most updated copy of the block B1 in the home directory of N1 and to invalidate the block B1 in the past owner's cache C3 in N3.
What is are the other messages sent between the nodes?	**1)** **Data Write Back** message reply from remote cache N3 to the home node N1 to write back the most updated copy of the block B1 in the home directory of N1.

	2) **Data Value Reply** message sent from the home node N1 to the local node N2 to get the most updated copy of the block B1 in the cache C2 to make the write. Then the node N2 becomes the new owner of block B1.
At the end of the above sequence of messages:	
Which is the coherence state and sharer bits of the block B1 in the home directory?	Directory N1 Block B1 | State: **Modified** | Sharer Bits: **0010**
Which node is the new owner of the block B1?	**N2**
Which is the coherence state of the block B1 in the Local Cache?	**Modified** in the Local Cache C2 of node N2.

QUESTION 2: LOOP UNROLLING (6 points)

Let's consider the following loop code where registers **$R1** and **$R2** are initialized to **0** and **$R3** is initialized to **400:**

```
LOOP:  LD $F2, 0 ($R1)
       LD $F4, 0 ($R2)
       FADD $F6, $F2, $F4
       SD $F6, 0 ($R1)
       ADDDUI $R1, $R1, 4
       ADDDUI $R2, $R2, 4
       BNE $R1, $R3, LOOP
```

Answer to the following questions:

How many iterations of the loop?	100 iterations
How many instructions per iteration?	7 instructions per iteration
How many instructions dedicated to the loop overhead per iteration?	3 out of 7 instructions per iteration
How many instructions are executed globally?	700 instructions
How many branch instructions are executed globally?	100 branches (one branch per iteration)
Let's consider a loop unrolling version of the code with unrolling factor 2.	
How many iterations of the unrolled loop?	50 iterations
How many instructions per iteration?	(4 x 2) + 3 = 11 instructions per iteration
How many instructions dedicated to the loop overhead per iteration?	3 out of 11 instructions per iteration
How many instructions are executed globally?	550 instructions

How many branch instructions are executed globally?	50 branches (one branch per iteration)
How many more registers per iteration are needed due to register renaming?	3 more FP registers are needed to avoid the WAW hazards on \$F2, \$F4 and \$F6
What are the registers that need to be renamed?	Registers \$F2, \$F4 and \$F6 must be first used and then renamed in the second replication of the original iteration in the unrolled loop.
How much is the global instruction count decrease?	(700 – 550) / 700 = 21.43%
Let's consider a loop unrolling version of the code with unrolling factor 4.	
How many iterations of the unrolled loop?	25 iterations
How many instructions per iteration?	(4 x 4) + 3 = 19 instructions per iteration
How many instructions due to the loop overhead per iteration?	3 out of 19 instructions per iteration
How many instructions are executed globally?	475 instructions
How many branches are executed?	25 branches (1 branch per iteration)
How many more registers per iteration are needed due to register renaming?	(3 x 3) = 9 more FP registers are needed to avoid the WAW hazards on \$F2, \$F4 and \$F6
What are the registers that need to be renamed?	Registers \$F2, \$F4 and \$F6 must be first used and then renamed in the next three replications of the original iteration in the unrolled loop.
How much is the global instruction count decrease?	(700 – 475) / 700 = 32%
What is the best unrolling factor between 2 and 4? Explain why	The best is the unrolling factor 4 if we consider the global instruction count decrease, the reduced number of branches, the increase of the length of the basic block that can be used to better exploit the ILP. The drawback is the higher register pressure due to the higher register renaming.

MULTIPLE-CHOICE QUESTIONS: (8 points)

Question 1 (format Multiple Choice – Multiple answer)

Let's consider the following loop:

```
for (i=1; i<=100, i++) {
     A[i] = A[i] + A[i-1];    /*S1*/
     C[i] = A[i] + B[i-1]     /*S2*/
     D[i] = A[i] + D[i-1]     /*S3*/
}
```

How many loop-carried dependencies are in the code?

(MULTIPLE ANSWERS)
1 point

Answer 1: One in S2 because C[i] depends on B[i-1];

Answer 2: One in S3 because D[i] depends on D[i-1]; (**TRUE**)

Answer 3: One in S1 because A[i] depends on A[i-1]; (**TRUE**)

Answer 4: One in S2 because C[i] depends on A[i];

Motivate your answers:
1 point

Feedback:
The dependence of C[i] on B[i-1] in S2 is not a loop-carried dependence because the vector B[ ] is never modified in the loop.
The dependence of D[i] on D[i-1] in S3 is a loop-carried dependence
The dependence of A[i] on A[i-1] in S1 is a loop-carried dependence
Dependences of C[i] on A[i] in S2 and D[i] on A[i] in S3 are not loop-carried dependences.

Question 2 (format Multiple Choice – Single answer)

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, R1              # Load vector from mem. address R1 into V1
MULVS.D V1, V1, S1      # FP multiply vector V1 to scalar F1
ADDVV.D V2, V1, V1      # FP add vectors V1 and V1
MULVS.D V2, V2, S2      # FP multiply vector V2 to scalar F2
L.V V3, R2              # Load vector from mem. address R2 into V3
ADDVV.D V3, V2, V3      # FP add vectors V2 and V3
S.V V3, R1              # Store vector V3 into memory address R1
```

How many convoys? How many clock cycles to execute the code?
(SINGLE ANSWER)
1 point

Answer 1: 3 convoys; 24 clock cycles **(TRUE)**

Answer 2: 2 convoys; 16 clock cycles

Answer 3: 4 convoys; 32 clock cycles

Answer 4: 5 convoys; 40 clock cycles

Motivate your answer by completing the following figure:
1 point

0	*1*	*… L.V V1, RX …*	*7*

Question 3 (format Multiple Choice – Single answer)

*Let's consider a **Speculative Tomasulo** architecture with:*

- *2 load buffers (Load1, Load2),*
- *2 FP reservation stations (FPS1 and FPS2)*
- *2 Integer reservation stations (RS1 and RS2)*
- ***8-entry ROB** (ROB0, ROB1, …, ROB7).*

Let's consider the following LOOP after the issue of the instructions of the first iteration (while the first load is executing a cache miss).

```
LOOP: LD $F2, 0 ($R1)
      LD $F4, 0 ($R2)
      FADD $F4, $F2, $F4
      SD $F4, 0 ($R1)
      SUBI $R1, $R1, 8
      BNEZ $R1, LOOP   // branch prediction taken
```

In the Rename Table, what are the pointers used for **$F2** *and* **$F4?**

(SINGLE ANSWER)
1 point

Answer 1: ROB0 for $F2 and ROB1 for $F4

Answer 2: ROB0 for $F2 and ROB2 for $F4**(TRUE)**

Answer 3: ROB0 for $F2 and ROB3 for $F4

Answer 4: ROB0 for $F2 and ROB4 for $F4

Is the ROB full? If not, please explain whether or not the second iteration of the loop can start to issue speculatively. Motivate your answers:
1 point

Feedback:
The 8-entry ROB is not full because there are two free entries. However, the second iteration cannot be issued speculatively (assuming the branch is predicted as taken) because the structural hazard related to the two load buffers that are already busy.

ROB#	Instruction	Dest.	Ready	
ROB0	LD $F2, 0 ($R1) (1^ iteration exec. cache miss)	$F2	No	HEAD
ROB1	LD $F4, 0 ($R2) (1^ iteration issued)	$F4	No	
ROB2	FADD $F4, $F2, $F4 (1^ iteration issued)	$F4	No	
ROB3	SD $F4, 0 ($R1) (1^ iteration issued)	MEM	No	
ROB4	SUBI $R1, $R1, 8 (1^ iteration issued)	$R1	No	
ROB5	BNEZ $R1, LOOP (1^ branch predicted as taken)		No	
ROB6				
ROB7				

Rename Table:

$F0	
$F2	ROB0
$F4	~~ROB1~~ ROB2

Question 4 (format Multiple Choice – Single answer)

Let's consider a loop code iterated 50 times executed by a processor with a 1-entry 1-bit Branch History Table initialized as not-taken.
*How much is the **branch misprediction rate** of the loop code with one conditional branch that takes the execution back to the beginning of the loop (loop-backward branch)?*

(SINGLE ANSWER)
1 point

Answer 1: 2%

Answer 2: 4% (TRUE)

Answer 3: 96 %

Answer 4: 5%

Answer 5: 95 %

Feedback
Being the predictor initialized as NT, we have 2 mispredictions out of 50 branch predictions: one at the first execution of the loop-back branch and one at the loop exit (4% mispredictions).

Question 5 (format Multiple Choice – Single answer)

Let's consider a quad-issue SMT processor that can manage up to 8 simultaneous threads.
What are the values of the ideal CPI and the ideal per-thread CPI?

(SINGLE ANSWER)
1 point

Answer 1: Ideal CPI = 1 & Ideal per-thread CPI = 0.125
Answer 2: Ideal CPI = 0.25 & Ideal per-thread CPI =4
Answer 3: Ideal CPI = 0.5 & Ideal per-thread CPI = 2
Answer 4: Ideal CPI = 0.25 & Ideal per-thread CPI = 2 (TRUE)
Answer 5: Ideal CPI = 0.25 & Ideal per-thread CPI = 8