NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 29/06/2022

ACA2022_FORM2_29June2022_SILVANO

ACA Course -- Prof. SILVANO

FORM2 is composed of 6 QUESTIONS to get UP TO 12 POINTS

Duration is 24 minutes!

Question 1 (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	Memory at bl. 0 up to date?	Memory at bl. 1 up to date?
0	Initial state	Invalid	Invalid	Invalid	Invalid	Yes	Yes
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	P1: Read Bl. 0						
4	P3: Write Bl. 0						
5	P0: Write Bl. 1						
6	P2: Write Bl. 1						
7	P1: Read Bl. 0						

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 29/06/2022

Feedback

Cycle	After Operation	P0 cache block state	P1 cache block state	P2 cache block state	P3 cache block state	Memory at bl. 0 up to date?	Memory at bl. 1 up to date?
0	Initial state	Invalid	Invalid	Invalid	Invalid	Yes	Yes
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	P1: Read Bl. 0	Excl (1)	Shared (0)	Shared (0)	Invalid	Yes	Yes
4	P3: Write Bl. 0	Excl (1)	Invalid	Invalid	Mod (0)	No	Yes
5	P0: Write Bl. 1	Mod (1)	Invalid	Invalid	Mod (0)	No	No
6	P2: Write Bl. 1	Invalid	Invalid	Mod (1)	Mod (0)	No	No
7	P1: Read Bl. 0	Invalid	Shared (0)	Mod (1)	Shared (0)	Yes	No

See MESI protocols on the slides on L13: Multiprocessors.

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 29/06/2022

Question 2 (complete table format)
2 points

Given the following assembly code:

```
FOR:LD $F2, A($R1)
    LD $F4, B($R1)
    LD $F6, C($R1)
    FADD $F2,$F2,$F2
    SD $F2,B($R1)
    FADD $F4,$F2,$F4
    SD $F4,C($R1)
    FADD $F6,$F4,$F6
    SD $F6,D($R1)
    ADDUI $R1,$R1,4
    BNE $R1,$R2, FOR
```

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

- 1 Integer ALU with 1 cycle latency to next Int/FP & 2 cycle latency to next Branch
- 1 Memory Unit with 2 cycle latency
- 1 FP ALU with 3 cycle latency

The branch is completed with 1 cycle delay slot (branch solved in ID stage).
In the Register File, it is possible to read and write at the same address at the same clock cycle.
Complete the following table with the list-based scheduling on the 3-issue VLIW machine including the BRANCH DELAY SLOT (NOPs are not written).

	Integer ALU	Mem Unit	FP ADD
C1		`LD F2, A(R1)`	
C2			
C3			
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 29/06/2022

Feedback

	Integer ALU	Mem Unit	FP ADDER
C1		LD F2, A(R1)	
C2		LD F4, B(R1)	
C3		LD F6, B(R1)	FADD F2, F0, F2
C4			
C5			
C6		SD F2, B(R1)	FADD F4, F2, F4
C7			
C8			
C9		SD $F4,C($R1)	FADD F6, F4, F6
C10			
C11			
C12	ADD R1,R1,4	SD $F6,D($R1)	
C13			
C14	BNE $R1,$R2, FOR		
C15	Br. delay slot		

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 29/06/2022

Question 3 (format open text)
2 points

1. How long is the critical path? _____
2. What performance did you achieve in CPIas?

   _____

3. How much is the code efficiency?

   _____

4. Let's assume the same code to be rescheduled on a **4-issue VLIW** with the same characteristics as before but with **2 Memory Units** instead of 1 Memory Unit:
   - How much has the critical path been improved?

     _____

   - Has the code efficiency been improved?

     _____

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 29/06/2022

Feedback

1. How long is the critical path? **15 cycles**
2. What performance did you achieve in CPIas?

   CPIas = (# cycles) / IC = 15 / 11 = 1,14
3. How much is the code efficiency?

   Code_eff = IC/ (# cycles * # issues) = 11 / (15 * 3) = 0,24

Given a **4-issue VLIW** machine with with the same characteristics as before but with **2 Memory Units** instead of 1 Memory Unit we have had the following schedule:

	Integer ALU	Mem Unit	Mem Unit	FP ADDER
C1		`LD F2, A(R1)`	`LD F4, B(R1)`	
C2		`LD F6, B(R1)`		
C3				`FADD F2, F0, F2`
C4				
C5				
C6		`SD F2, B(R1)`		`FADD F4, F2, F4`
C7				
C8				
C9		`SD $F4,C($R1)`		`FADD F6, F4, F6`
C10				
C11				
C12	`ADD R1,R1,4`	`SD $F6,D($R1)`		
C13				
C14	`BNE $R1,$R2, FOR`			
C15	`Br. delay slot`			

- How much has the critical path been improved?

  The critical path is the same as before
- Has the code efficiency been improved?

  No, it hasn't: the Code efficient is worse than before:

  Code_eff = IC/ (# cycles * # issues) = 11 / (15 * 4) = 0,18

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 29/06/2022

Question 4 (format complete table and open text)
2 points

Let's consider the following assembly code:

```
I1: lw $f0,0($r1)
I2: fadd $f0,$f0,$f0
I3: sw $f0,0($r1)
I4: lw $f1,4($r1)
I5: lw $f2,8($r1)
I6: fadd $f1,$f1,$f2
I7: sw $f1,4($r1)
I8: sw $f2,C($r1)
```

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

- *2 RESERVATION STATIONS **(RS1, RS2)** for the LOAD/STORE unit **(LDU1)** with latency 4*
- *2 RESERVATION STATION **(RS3, RS4)** for the ALU/BR FUs **(ALU1)** with latency 2*

Please complete the following table:

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT	Hazards Type	RSi	UNIT
`I1: lw $f0,0($r1)`	1	2	6	None	RS1	LDU1
`I2: fadd $f0,$f0,$f0`	2	7	9	RAW $f0	RS3	ALU1
`I3: sw $f0,0($r1)`						
`I4: lw $f1,4($r1)`						
`I5: lw $f2,8($r1)`						
`I6: fadd $f1,$f1,$f2`						
`I7: sw $f1,4($r1)`						
`I8: sw $f2,C($r1)`						

Calculate the CPI:

CPI = _____

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 29/06/2022

Feedback:

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT	Hazards Type	RSi	UNIT
`I1: lw $f0,0($r1)`	1	2	6		RS1	LDU1
`I2: fadd $f0,$f0,$f0`	2	**7**	9	**RAW $f0**	RS3	ALU1
`I3: sw $f0,0($r1)`	3	**10**	14	**RAW $f0** (STRUCT LDU1)	RS2	LDU1
`I4: lw $f1,4($r1)`	**7**	**15**	19	**STRUCT RS1 + STRUCT LDU1**	RS1	LDU1
`I5: lw $f2,8($r1)`	**15**	**20**	24	**STRUCT RS2 + STRUCT LDU1**	RS2	LDU1
`I6: fadd $f1,$f1,$f2`	16	**25**	27	**RAW $f2** (RAW $f1)	RS3	ALU1
`I7: sw $f1,4($r1)`	**20**	**28**	32	**STRUCT RS1** + (STRUCT LDU1) + **RAW $f1**	RS1	LDU1
`I8: sw $f2,C($r1)`	**25**	**33**	37	**STRUCT RS2 + STRUCT LDU1** + (RAW $f2)	RS2	LDU1

CPI = # clock cycles / IC = 37 / 8 = 4.625

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 29/06/2022

Question 5 (format open text)
2 points

Let's consider the following assembly code:

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

Write a software-pipelined version of this loop by omitting the start-up and finish up code

Open text answer (max 15 rows)

Feedback:

```
LOOP: SD $F4, 0 ($R1)        /* store from iteration [i] corr. to index 0 */
       SD $F6, 0 ($R2)       /* store from iteration [i] corr. to index 0 */
      ADDD $F4, $F0, $F0     /* add from iteration [i+1] corr. to index 8 */
      ADDD $F6, $F2, $F2     /* add from iteration [i+1] corr. to index 8 */
      LD $F0, 16 ($R1)       /* load from iteration [i+2] corr. to index 16 */
      LD $F2, 16 ($R2)       /* load from iteration [i+2] corr. to index 16 */
      ADDDUI $R1, $R1, 8
      ADDDUI $R2, $R2, 8
      BNE $R1, $R3, LOOP
```

See slides on L09 VLIW Code Scheduling

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 29/06/2022

Question 6 (complete table format)
2 points

Let's consider the following assembly code:

Assembly Code:

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

For each statement of the assembly code, the following table reports data dependencies (corresponding to RAW hazards) and name dependencies (corresponding to WAR/WAW hazards):

Please complete the following table with dependencies (hazards):

Assembly Code:		Data dependencies (RAW Hazards)	Name depend. (WAR / WAW hazards)
LOOP: LD $F0, 0 ($R1)	#S1	None	None
LD $F2, 0 ($R2)	#S2	None	None
ADDD $F4, $F0, $F0	#S3	RAW $F0 w. S1	None
ADDD $F6, $F2, $F2	#S4	RAW $F2 w. S2	None
SD $F4, 0 ($R1)	#S5		
SD $F6, 0 ($R2)	#S6		
ADDDUI $R1, $R1, 8	#S7		
ADDDUI $R2, $R2, 8	#S8		
BNE $R1, $R3, LOOP	#S9		

Feedback

Assembly Code:		Data dependencies (RAW Hazards)	Name dependencies (WAR / WAW hazards)
LOOP: LD $F0, 0 ($R1)	#S1	None	None
LD $F2, 0 ($R2)	#S2	None	None
ADDD $F4, $F0, $F0	#S3	RAW $F0 w. S1	None
ADDD $F6, $F2, $F2	#S4	RAW $F2 w. S2	None
SD $F4, 0 ($R1)	#S5	RAW $F4 w. S3	None
SD $F6, 0 ($R2)	#S6	RAW $F6 w. S4	None
ADDDUI $R1, $R1, 8	#S7	None	WAR $R1 w.S5, S1
ADDDUI $R2, $R2, 8	#S8	None	WAR $R2 w.S6, S2
BNE $R1, $R3, LOOP	#S9	RAW $R1 w. S7	None