NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 21/07/2022

ACA2022_FORM2_21July2022_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	P3: Read Bl. 0						
4	P1: Write Bl. 0						
5	P0: Write Bl. 1						
6	P3: Read Bl. 1						
7	P2: Read Bl. 0						

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 21/07/2022

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	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
7	P2: Read Bl. 0	Shared (1)	Shared (0)	Shared (0)	Shared (1)	Yes	Yes

See MESI protocols on the slides on L13: Multiprocessors.

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 21/07/2022

Question 2 (complete table format)
2 points

Let's consider the following assembly code:

```
FOR:LD $F2, A($R1)
    LD $F4, B($R1)
    FADD $F2, $F2, $F2
    FADD $F4, $F4, $F4
    LD $F6, C($R1)
    LD $F8, D($R1)
    FADD $F6, $F2, $F6
    FADD $F8, $F4, $F8
    SD $F2, A($R1)
    SD $F4, B($R1)
    SD $F6, C($R1)
    SD $F8, D($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: 21/07/2022

Feedback

	Integer ALU	Mem Unit	FP ADDER
C1		LD F2, A(R1)	
C2		LD F4, B(R1)	
C3		LD F6, C(R1)	FADD F2, F2, F2
C4		LD F8, D(R1)	FADD F4, F4, F4
C5			
C6		SD F2, A(R1)	FADD F6, F2, F6
C7		SD $F4, B($R1)	FADD F8, F4, F8
C8			
C9		SD $F6, C($R1)	
C10	ADD R1, R1, 4	SD $F8, D($R1)	
C11			
C12	BNE $R1, $R2, FOR		
C13	Br. delay slot		

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 21/07/2022

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: 21/07/2022

Feedback

1. How long is the critical path?

   13 cycles

2. What performance did you achieve in CPIas?

   CPIas = (# cycles) / IC = 13 / 14 = 0.93

3. How much is the code efficiency?

   Code_eff = IC/ (# cycles * # issues) = 14 / (13 * 3) = 0.36

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, C(R1)`	`LD F8, D(R1)`	
C3				`FADD F2, F2, F2`
C4				`FADD F4, F4, F4`
C5				
C6		`SD F2, A(R1)`		`FADD F6, F2, F6`
C7		`SD $F4,B($R1)`		`FADD F8, F4, F8`
C8				
C9		`SD $F6,C($R1)`		
C10	`ADD R1,R1,4`	`SD $F8,D($R1)`		
C11				
C12	`BNE $R1,$R2, FOR`			
C13	`Br. delay slot`			

- How much has the critical path been improved?

  The critical path is the same as before (13 cycles)

- Has the code efficiency been improved?

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

  Code_eff = IC/ (# cycles * # issues) = 14 / (13 * 4) = 0.27

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 21/07/2022

Question 4 (format complete table)
2 points

Let's consider 32-block main memory with a direct-mapped 8-block cache based on a ***write allocate*** with ***write-through*** protocol.

The addresses are expressed as decimal numbers:
Memory Address: $[0, 1, 2, \ldots\ldots\ldots 31]_{10}$
Cache Address: $[0, 1, 2, \ldots 7]_{10}$
and Cache Tags are expressed as binary numbers.

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

	Type of memory access	Memory Address	HIT/MISS Type	Cache Tag	Cache Address	Write in memory
1	Read	$[24]_{10}$	Cold-start Miss	$[11]_2$	$[0]_{10}$	No
2	Write	$[24]_{10}$	Hit	$[11]_2$	$[0]_{10}$	Yes, Wr. in M$[24]_{10}$
3	Read	$[24]_{10}$	Hit	$[11]_2$	$[0]_{10}$	No
4	Read	$[10]_{10}$				
5	Read	$[16]_{10}$				
6	Write	$[10]_{10}$				
7	Write	$[24]_{10}$				
8	Read	$[12]_{10}$				
9	Read	$[21]_{10}$				
10	Write	$[18]_{10}$				

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 21/07/2022

Feedback

	Type of memory access	Memory Address	HIT/MISS Type	Cache Tag	Cache Address	Write in memory
1	Read	$[24]_{10}$	Cold-start Miss	$[11]_{2}$	$[0]_{10}$	No
2	Write	$[24]_{10}$	Hit	$[11]_{2}$	$[0]_{10}$	Yes, Wr. in M$[24]_{10}$
3	Read	$[24]_{10}$	Hit	$[11]_{2}$	$[0]_{10}$	No
4	Read	$[10]_{10}$	Cold-start Miss	$[01]_{2}$	$[2]_{10}$	No
5	Read	$[16]_{10}$	Conflict Miss	$[10]_{2}$	$[0]_{10}$	No
6	Write	$[10]_{10}$	Hit	$[01]_{2}$	$[2]_{10}$	Yes, Wr. in M$[10]_{10}$
7	Write	$[24]_{10}$	Conflict Miss	$[11]_{2}$	$[0]_{10}$	Yes, Wr. in M$[24]_{10}$
8	Read	$[12]_{10}$	Cold-start Miss	$[01]_{2}$	$[4]_{10}$	No
9	Read	$[21]_{10}$	Cold-start Miss	$[10]_{2}$	$[5]_{10}$	No
10	Write	$[18]_{10}$	Conflict Miss	$[10]_{2}$	$[2]_{10}$	Yes, W in M$[18]_{10}$

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 21/07/2022

Question 5 (format open text)
2 points

Let's consider the following assembly code:

```
FOR:LD $F2, A($R1)
    LD $F4, B($R1)
    FADD $F2, $F2, $F2
    FADD $F4, $F4, $F4
    LD $F6, C($R1)
    LD $F8, D($R1)
    FADD $F6, $F2, $F6
    FADD $F8, $F4, $F8
    SD $F2, A($R1)
    SD $F4, B($R1)
    SD $F6, C($R1)
    SD $F8, D($R1)
    ADDUI $R1, $R1, 4
    BNE $R1, $R2, FOR
```

Write a software-pipelined version of this loop without the startup and finish up code.

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 21/07/2022

Feedback:

```
FOR:SD $F2, A($R1)          /* store from iteration [i] corr. to index 0 */
    SD $F4, B($R1)          /* store from iteration [i] corr. to index 0 */
    SD $F6, C($R1)          /* store from iteration [i] corr. to index 0 */
    SD $F8, D($R1)          /* store from iteration [i] corr. to index 0 */
    FADD $F2, $F2, $F2      /* add from iteration [i+1] corr. to index 4 */
    FADD $F4, $F4, $F4      /* add from iteration [i+1] corr. to index 4 */
    FADD $F6, $F2, $F6      /* add from iteration [i+1] corr. to index 4 */
    FADD $F8, $F4, $F8      /* add from iteration [i+1] corr. to index 4 */
    LD $F2, A+8($R1)          /* load from iteration [i+2] corr. to index 8 */
    LD $F4, B+8($R1)          /* load from iteration [i+2] corr. to index 8 */
    LD $F6, C+8($R1)          /* load from iteration [i+2] corr. to index 8 */
    LD $F8, D+8($R1)          /* load from iteration [i+2] corr. to index 8 */
    ADDUI $R1, $R1, 4
    BNE $R1, $R2, FOR
```

See slides on L09 VLIW Code Scheduling

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 21/07/2022

Question 6 (complete table format)
2 points

Let's consider to accelerate a processor architecture by adding a vector mode to it: When executing in vector mode, the processor is 20 times faster than the normal execution mode of.
We indicate as *percentage of vectorization* the percentage of computation time that could be spent using the vector mode.

1. What *percentage of vectorization* F_v is needed to achieve an overall speedup of 2?

2. Given the following percentages of vectorization for a target program, what are the corresponding overall speedups that could be achieved?

F_v	$Speedup_{overall}$
0%	
10%	
25%	
50%	
75%	
90%	
100%	

NAME SURNAME PERSONAL CODE	ACA2022_EXAM_FORM2 Data: 21/07/2022

1. What *percentage of vectorization* F_v is needed to achieve an overall speedup of 2?

We need to apply the Amdahl's law:

$$Speedup_{overall} = 1 \ / \ [(1-F_v) + F_v / S_v]$$

Therefore in our case we have:

$2 = 1 / [(1-F_v) + F_v / 20]$

$\Rightarrow 2 = 20 / [20(1-F_v) + F_v]$

$\Rightarrow 40 (1-F_v) + 2F_v = 20$

$\Rightarrow F_v = 20 / 38 \Rightarrow F_v = 52.6\%$

3. Given the following percentages of vectorization for a target program, what are the corresponding overall speedups that could be achieved?

For each value of F_v, we need to apply the Amdahl's law where $S_v = 20$:

$$Speedup_{overall} = 1 / [(1-F_v) + F_v / 20]$$

F_v	$Speedup_{overall}$
0%	1
10%	1.10
25%	1.31
50%	1.90
75%	3.48
90%	6.90
100%	20