Prof. Cristina Silvano	ACA2022_EXAM_FORM1 Data: 21/07/2022

ACA2022_FORM1_21July2022_SILVANO

ACA Course -- Prof. SILVANO

FORM1 is composed of 15 QUESTIONS to get UP TO 21 POINTS

Duration is 35 minutes!

IMPORTANT: What to do before leaving MS FORM1:

1. **Please click on:** “Inviami una conferma tramite posta elettronica delle risposte” (Send me a confirmation by email of the answers). This is **fundamental** to get an email with subject: **“My responses".** Open the email and click on the link to **VIEW YOUR RESPONSES.**

2. **Please click on SUBMIT** (Click on **INVIA**) and you will get the message: **“Your answer has been sent”.** Points will be assigned **manually** to **FORM1**. Therefore, the view of your results is **not** immediate: it will be enabled by prof. Silvano **after** results publication phase.

Prof. Cristina Silvano	ACA2022_EXAM_FORM1 Data: 21/07/2022

Question 1 (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: 1001

Which is the request sent for the block B1 to become:

Directory N1 Block B1 | State: Modified | Sharer Bits: 0100

(SINGLE ANSWER)
1 point

Answer 1: Write Hit B1 sent from N1 to N1
Answer 2: Write Miss B1 from N1 to N1 **(TRUE)**
Answer 3: Read Hit B1 from N1 to N1
Answer 4: Read Miss B1 from N1 to N1

Question 2 (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 = 0.5 & Ideal per-thread CPI = 4
Answer 5: Ideal CPI = 0.25 & Ideal per-thread CPI = 2 **(TRUE)**

Question 3 (format Multiple Choice – Single answer)

Let's consider a 1-bit 1-entry Branch History Table initialized as not-taken and a loop iterated 100 times. How many branch mispredictions will be incurred by the conditional branch that takes the execution back to the beginning of the loop (loop-backward branch)?
(SINGLE ANSWER)
1 point

Answer 1: 1%
Answer 2: 2% **(TRUE)**
Answer 3: 99 %
Answer 4: 100%
Answer 5: 98 %

Feedback
Assume initial state NT, we have 2 mispredictions: once when the loop-back branch is first executed and once when the loop exits (98% success and 2% misprediction).

Question 4 (format Multiple answers)

In a distributed shared memory multiprocessor, which of the following statements are true?

(Multiple answers)
2 points

Answer 1: The access time to a shared memory location is uniform for all the processors (UMA)

Answer 2: The access time to a shared memory location is non uniform for all the processors (NUMA) **TRUE**

Answer 3: Multiple processors communicate through load/store operations on shared memory variables. **TRUE**

Answer 4: The shared memory model generates the cache coherence problem among multiple processors. **TRUE**

Answer 5: The shared memory can be either centralized on a single node or distributed over the nodes.

Question 5 (format Multiple answers)

The prediction bit in a Branch History Table could belong to another branch with the same low-order address bits than the current branch, therefore this can generate a misprediction. *To reduce this collision problem, what are the solutions that can be proposed?*

(Multiple answers)
2 points

Answer 1: Increase the number of rows in the BHT, even if this is costly in terms of HW. **True**

Answer 2: Introduce a hashing function that maps the branch address on k-bits, even if this can increase the access time to the BHT. **True**

Answer 3: Introduce more prediction bits in each entry of the Branch History Table, even if this is costly in terms of HW.

Answer 4: Combine the BHT to the Branch Target Buffer, that uses tags. **True**

Question 6 (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, RX                # load vector from memory address RX into V1
MULVS.D V1, V1, F0        # FP multiply vector V1 to scalar F0
L.V V2, RY                # load vector from memory address RY into V2
MULVS.D V2, V2, F0        # FP multiply vector V2 to scalar F0
ADDVV.D V3, V1, V2        # FP add vectors V1 and V2
S.V V3, RZ                # store vector V3 into memory address RZ
```

How many convoys? How many clock cycles to execute the code?

(SINGLE ANSWER)
2 points

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
Answer 5: 3 convoys; 25 clock cycles

Question 7 (format Multiple Choice – Multiple answers)

Let's consider the following loop code:

```
for (i=1; i<=100, i++) {
        X[i] = X[i-1] + Y[i-1] + Z[i-1];     /*S1*/
        Z[i] = Y[i-1] + W[i-1]               /*S2*/
}
```

What are the loop-carried dependences in the code?

(MULTIPLE ANSWERS)

2 points

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

Answer 2: One in S1 because X[i] depends on Y[i-1];

Answer 2: One in S1 because X[i] depends on Z[i-1]; (**TRUE**)

Answer 4: One in S2 because Z[i] depends on Y[i-1];

Answer 5: One in S2 because Z[i] depends on W[i-1];

Feedback
Please note that the dependences of Z[i] on Y[i-1] and on W[i-1] are not loop-carried dependences because the vectors Y[ ] and W[] are never modified in the loop.

Question 8 (format Multiple Choice – Single answer)

Let's consider the following loop kernel to be repeated for 100 iterations:

```
LOOP:  ld.d $FP2,VECTA($s1)
IF:    beq $s1, $s2, ELSE
THEN:  ld.d $FP6, 8($s1)        // more probable path
       sd.d $FP6, 8($s6)
       j INC
ELSE:  ld.d $FP2, 8($s1)       // less probable path
       add.d $FP2, $FP2, $FP0
       mul.d $FP4, $FP2, $FP2
       sd.d $FP2, 12($s1)
INC:   addi $s1,$s1,8
       bne $s0,$s7, LOOP
```

What is the best static branch prediction technique to be applied?

(SINGLE ANSWER)
1 point

Answer 1: Backward Taken Forward Not Taken (**TRUE**)
Answer 2: Always Taken
Answer 3: Always Not Taken
Answer 4: Delayed Branch

Question 9 (Format Multiple Choice – Multiple answers)

A 4-issue VLIW processor exploits:

(MULTIPLE ANSWERS)
2 points
Answer 1: Data-level parallelism by applying a single instruction to multiple data.
Answer 2: Insertion of NOPs to solve true data dependencies. **(TRUE)**
Answer 3: Dynamic scheduling techniques
Answer 4: Multiple-issue instruction level parallelism. **(TRUE)**
Answer 5: Static scheduling techniques **(TRUE)**

Prof. Cristina Silvano	ACA2022_EXAM_FORM1 Data: 21/07/2022

Question 10 (format True/False)
To speed up the access time, the level-1 cache can be virtually indexed and physically tagged in order to overlap the cache tag access with virtual address translation.

(True/False)
1 point

Answer: **TRUE**

Question 11 (format True/False)
In the Speculative Tomasulo architecture, the Store Path is designed to connect the store buffers to the memory unit.

(format True/False)
1 point

Answer 1: False

Feedback
In the Speculative Tomasulo architecture, there are no Store Buffers because they are substituted by ReOrder Buffers with the Store Path directly connecting to the memory unit.

Question 12 (format Multiple Choice – Reorder options)

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: 0010

After a **Write Miss on B1 from node N0**, reorder the sequence of messages sent among the nodes to get:

Directory N1 Block B1 | State: Modified | Sharer Bits: 1000

(REORDER OPTIONS)
1 point

1. **Fetch/Invalidate** message sent from home node N1 to remote node N2
2. **Data Write Back** message from past owner N2 to home node N1
3. **Data Value Reply** from home node N1 to local cache N0.

Feedback:

Due to the Write Miss on B1 from the Local Node N0 to home node N1, first there is **a Fetch/Invalidate** message sent from the home node N1 to the remote node N2 **(past owner)** to get the most recent copy of the block in the home node N1 through a **Data Write Back** message from past owner N2 to home node N1and invalidating the state of the block B1 in the past owner's cache C2.
Then there is a **Data Value Reply** from home node N1 to local cache N0.
The state of the block B1 in the directory N1 stays **Modified** but the **owner is changed** from N2 to N0 as follows: Directory N1 Block B1 | State: **Modified** | Sharer Bits: **1000.**

Question 13 (format Multiple Choice – Single answer)

Let's consider a **Speculative Tomasulo** architecture with 2 load buffers (Load1, Load2), 2 multiply reservation stations (Mult1 and Mult2) and 6-entry ROB (ROB0, ROB1, ROB2, ROB3, ROB4, ROB5).
Let's consider the following LOOP when ROB is full after the issue of the first loop iteration (while the first load is executing a cache miss) and the speculative issue of the LD of the second iteration.

```
LOOP: LD $F0, 0 ($R1)
        MULTD $F4, $F0, $F2
        SD $F4, 0 ($R1)
        SUBI $R1, $R1, 8
        BNEZ $R1, LOOP   // branch prediction taken
```

In the Rename Table, what are the pointer values used for $F0 and $F4?

(SINGLE ANSWER)
2 points

Answer 1: ROB5 for $F0 and ROB1 for $F4 **(TRUE)**
Answer 2: ROB5 for $F0 and ROB2 for $F4
Answer 3: ROB0 for $F0 and ROB1 for $F4
Answer 4: ROB0 for $F0 and ROB2 for $F4

Feedback:
The 6-entry ROB is FULL:

ROB#	**Instruction**	**Dest.**	**Ready**	
ROB0	LD $F0, 0 ($R1) (1^ iteration exec. cache miss)	$F0	No	HEAD
ROB1	MULTD $F4, $F0, $F2 (1^ iteration issued)	$F4	No	
ROB2	SD $F4, 0 ($R1) (1^ iteration issued)	MEM	No	
ROB3	SUBI $R1, $R1, 8 (1^ iteration issued)	$R1	No	
ROB4	BNEZ $R1, LOOP (1^ branch predicted as taken)		No	
ROB5	LD $F0, 0 ($R1) (2^ iteration issued speculatively)	$F0	No	

Rename Table:

$F0	ROB5
$F2	

$F4	ROB1

Therefore, in the Rename Table, $F0 points to ROB5 because the WAW $F0 is solved while $F4 points to ROB1 because the second MULTD has not yet been issued because the ROB is full. See also L07: Reorder Buffer & Speculation

Question 14 (format Multiple Choice – Single answer)

In a directory-based protocol for cache coherency, the message called "Data Value Reply" is sent:
(SINGLE ANSWER)
2 points

Answer 1: From the local node cache to return data to the home directory
Answer 2: From the home directory to return data to the local node cache **(TRUE)**
Answer 3: From the remote node cache to return data to the home directory

Feedback:
See Directory-based protocols messages on the slides on L13: Multiprocessors.

Question 15 (format True/False)
To obtain a loop unrolling version of a code, we need to use register renaming if there are some true data dependences in the original code.

(format True/False)
1 point

Answer 1: **False**