VLSI Architecture Design Course (048853)
Final Exam
July 13th, 2003

Electrical engineering Department

Student name:______________________ Student number:________________

This exam contains TWO questions.
The exam duration is 2:30 hours.
Please fill the answers ON THE EXAM forms.

Please explain or provide a formula for each computation!

TAKE YOUR TIME, READ THE QUESTIONS THOUROUGHLY, UNDERSTAND THE CONTENT AND, *ONLY THEN,* START TO ANSWER

Good luck!

Q1	
Q2	
Total	

Question 1: Multi-Threading (50%)

A processor with the following characteristics is given:

- Fine-Grain Interleaved Multithreading (processing different thread each cycle)
- In-order issue, single pipe
- The processor can run in one of two modes: either Single-threaded or Dual-threaded. In the single-threaded mode, the processor can issue one instruction every other cycle. In the dual-threaded mode, the processor can issue one instruction every cycle.
  That is: the ideal CPI in the single threaded mode is 2, while in the dual-threaded mode it is 1.
- 1-bit local branch predictor (BP) – Unified for both threads, indexed by the IP,
- Branch penalty of 20 cycles per misprediction (measured from execution to execution)
- The BP is idealized. It has infinite entries and is updated immediately after each branch
- Perfect instruction and data caches (100% cache hit rate)

For example:

Single-threaded (ideal CPI=2), executing thread A:

Cycle	Thread	Instruction number
1	A	1
2		
3	A	2
4		
5	A	3
6		

Dual-threaded (ideal CPI=1), executing threads A+B:

Cycle	Thread	Instruction number
1	A	1
2	B	1
3	A	2
4	B	2
5	A	3
6	B	3

The branch predictor table structure is as follows:

Tag Value	1-bit history counter

The tag value is a function of the Program Counter of the branch. The 1-bit counter indicates the last branch direction.

A function *quantize()* reads through a byte vector and checks for each element whether its value is above or below a given threshold. Elements with values below the threshold are rounded to Zero (0), while elements with values above the threshold are rounded to 255.
Consider the following software implementation of the function *quantize*:

quantize(int *vector, int *size, int threshold, int *quantized_vector)

```
         push   …
         mov    r1, vector
         mov    r2, size
         mov    r3, threshold
         mov    r4, quantized_vector
loop:    cmp    [r1],r3
         jbt    Bigger
         mov    [r4],0
         jmp    End
bigger:  mov    [r4],255
         jmp    End
end:     inc    r1
         inc    r4
         dec    r2
         jnz    loop
         pop    …
         ret
```

The blocked portion designates the main loop of *quantize*.

A1. What is fine-grain interleaved multithreading?

A2. Compare its advantages and disadvantages over other ways of multithreading. Fill in the following table on the next page:

	Simultaneous MT	Blocked MT	Interleaved MT
Instruction scheduling cost			
Context switch cost			
Device utilization			
Single-thread throughput			

B. Assume that the processor is running in single-threaded mode. Consider the following program that quantizes two pictures:

```
quantize (Picture, 1000x1000, 16, New_Picture1);
quantize (Picture, 1000x1000, 240, New_Picture2);
```

B1. Compute the number of times the inner loop is executed.

B2. Given that all elements in the input array Picture have a value of 128, compute the CPI and the number of cycles required to execute the inner loops of the program. Ignore the initialization overhead.

C. In order to speed up the computation, the programmer decided to exploit the multithreading capability of the processor, and to split the computation into two separate threads:

```
Thread 1:   quantize (Picture, 1000x1000, 16, New_Picture1)
Thread 2:   quantize (Picture, 1000x1000, 240, New_Picture2)
```

The two threads enter the function *quantize()* simultaneously. Assume that all elements in the input array Picture have a value of 128 and that the first access to the BP is a miss.

C1. Fill in the following table.

Cycle	Instruction Thread A	Instruction Thread B
1	loop: cmp [r1],r3	
2		loop: cmp [r1],r3
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		
40		
51		
52		
53		

C2. Based on your results for C1, compute the CPI of the processor within the loops.

D. What is the cause of the performance gap between single-threaded (B2) and dual-threaded (C2) modes? Explain your results.

E. If all the elements in the input array Picture were all zeros, what would your results be for questions B2 and C2?

F. Without computing, how will a 2-bit branch predictor affect the results of questions C2 and E?

G. Suggest at least one microarchitectural improvement that will improve the results of the above benchmarks. Discuss the implications of your improvements.

Question 2: Trace Cache (50%)

Given a machine with a trace cache with the following structure:

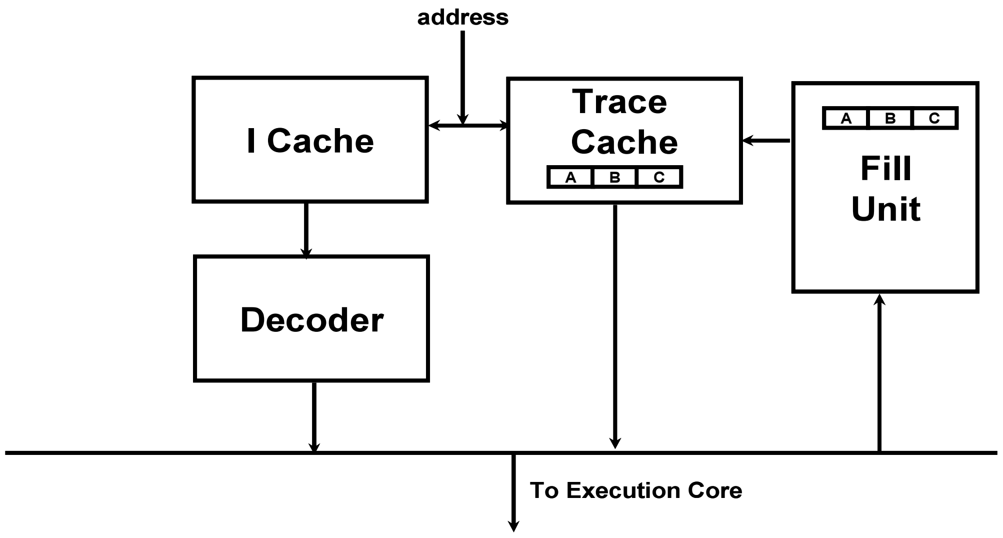


Fig1: the machine with a trace cache

The Trace cache fields are shown in Fig 2:

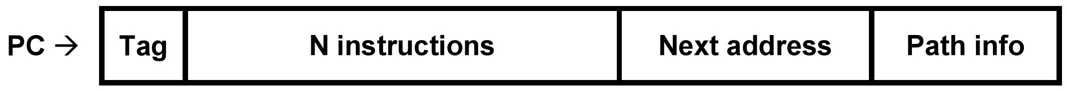
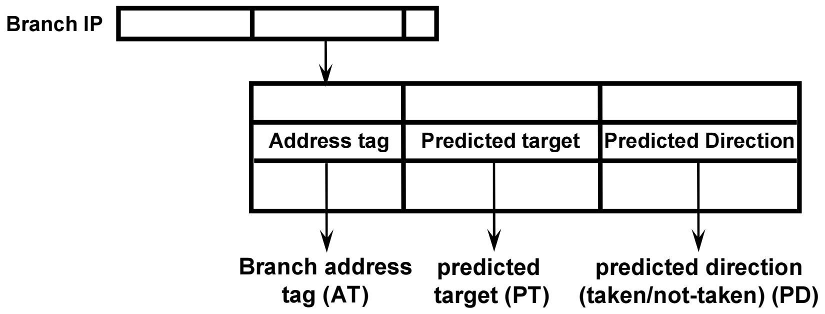


Fig2: Trace cache fields

There is also Branch predictor with the following information:

1. **Trace cache Address Tag (AT)**
2. **Predicted Target (PT)**



3. **Predicted direction (PD)**

Fig3: Branch Predictor

A1. What is the purpose of the Trace Cache Tag information?
What does it contain?
In what ways does this tag differ from a traditional Instruction cache?

A2. What is the purpose of next address field? What does it contain? How many bits are needed?

A3. What is the purpose of the Path information field. What does it contain?

B. Consider that your processor architecture supports Self Modifying Code (SMC). SMC means that an agent (DMA, a processor...) can write and alter instructions. In case that an SMC action is detected, traces that contain affected instructions should be invalidated.

B1. How would you detect an SMC write that alters an instruction in the trace cache? What information do you need to have?

There are two design options for SMC invalidation:

a. Precise invalidation (only traces that contain altered instructions are invalidated)

b. Non-precise invalidation (e.g. invalidate all instructions in the page of the altered instruction), with minimal die area impact

B2. Please suggest two invalidation solutions for the above options. Provide some "back of the envelope" estimations of the amount of false invalidation for each of the above options.

Comment [r1]: If we want 3, let leave space for 3… We have only a and b.

a.

b.

B3. Can you think about a more radical solution in case where SMC occurs very very seldom?