University of Western Ontario, Computer Science Department
CS3350B, Computer Architecture
Quiz 1 (30 minutes)
January 21, 2015

Student ID number:
Student Last Name:

Exercise 1. In the following list of performance metrics, identify those that can be used to calculate the CPU execution time of a program. For each such performance metric, indicate a formula estimating CPU execution time where this performance metric is used. Your formulas should be as simple as possible, that is, should not contain any unnecessary variables.

- Cycles per instruction (CPI),
- Clock rate (CR in Hz),
- Clock cycle time (CC in sec.),
- Instruction count (IC).

Answer:

All the above variables are performance metrics. Corresponding formulas are shown below:

$$\text{CPU\_time} = \text{IC} \times \text{CPI} \times \text{CC},$$

$$\text{CPU\_time} = \frac{\text{IC} \times \text{CPI}}{\text{CR}}.$$

Exercise 2. To capture the fact that the time to access data for both hits and misses affects performance, designers use average memory access time (AMAT) as a way to examine alternative cache designs. Average memory access time is the average time to access memory considering both hits and misses and the frequency of different accesses. In the list below, please circle the correct formula for calculating the AMAT of a program on a computer with one level cache (L1) and a main memory (DRAM).

Answer box:

- AMAT = L1_hit_time + L1_miss_rate * L1_miss_penalty
- AMAT = L1_miss_penalty + L1_hit_rate * L1_hit_time
- AMAT = L1_miss_penalty + L1_miss_rate * L1_hit_time

Answer:
AMAT = L1_hit_time + L1_miss_rate * L1_miss_penalty

Exercise 3. Computing the overall effective CPI is done by looking at the different types of instructions and their individual cycle counts and averaging. Please use the information in the table to answer questions:

$Operation$	$Frequency$	CPI_i
ALU	50%	1
Load	20%	5
Store	10%	3
Branch	20%	2

Question 1: What is the overall effective CPI of the machine according to the table?

Answer: The overall effective is given by

$$\text{CPI} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{IC}_i), \quad \text{where}$$

- IC_i is the percentage of the number of instructions of class i executed,
- CPI_i is the (average) number of clock cycles per instruction for that instruction class,
- n is the number of instruction classes.

Therefore, from the above data, we have

$$\begin{aligned} \text{CPI} &= 50\% \times 1 + 20\% \times 5 + 10\% \times 3 + 20\% \times 2 \\ &= 2.2 \end{aligned}$$

Questing 2: What is the overall effective CPI of the machine if a better data cache reduces the average load time from 5 to 2 cycles?

Answer:

$$\begin{aligned}\text{CPI} &= 50\% \times 1 + 20\% \times 2 + 10\% \times 3 + 20\% \\ &= 1.6\end{aligned}$$

Question 3: What if two ALU instructions could be executed at once?

Answer:

$$\begin{aligned}\text{CPI} &= 50\% \times 1/2 + 20\% \times 5 + 10\% \times 3 + 20\% \times 2 \\ &= 1.95\end{aligned}$$

Exercise 4. In this exercise, we consider a direct-mapped cache memory where each cache block holds two words. We assume that each word is a one byte and that each memory address is a 4-bit number where

- the first 2 bits (from left to right) are the tag bits,
- the third bit is the set address (index), and
- the last bit is the offset from the beginning of the block.

Hence, the main memory of this computer holds 8 blocks, thus, 16 words (or bytes).

We assume that the following words are accessed in sequence, according to the following access pattern (from left to right):

Word number:	0	1	2	3	4	3	4	15
Memory address:	0000	0001	0010	0011	0100	0011	0100	1111

Table 1: Sequence of accessed words

We start with an empty cache and all blocks initially marked as not valid. (Valid bits are not shown on the pictures below.)

set	tag	block	
0			
1			

Table 2: Initially, the cache is empty.

Questions:

- Use Table 3 to Table 10 to depict the contents of the cache when the processor requests the 8 words, in sequence, as specified in Table 1. For each request of the processor to the cache, indicate whether this is a cache miss or cache hit.
- Calculate the cache miss rate.

Important remark: In the above statement, the *endianess* was left unspecified, on purpose, for simplicity. In the correction below, we first use the *Big-endian* system. Then, as an alternative solution, we use the *Little-endian* version. For learning about endianess, please refer to `http://en.wikipedia.org/wiki/Endianness`.

Both systems yield, of course, the same requests: 4 hits, 4 misses. Thus a 50% hit rate!

Big-endian answer:

set	tag	block	
0	00	Mem(0)	Mem(1)
1			

Table 3: Accessing word number 0 : miss

set	tag	block	
0	00	Mem(0)	Mem(1)
1			

Table 4: Accessing word number 1: hit

set	tag	block	
0	00	Mem(0)	Mem(1)
0	00	Mem(2)	Mem(3)

Table 5: Accessing word number 2 : miss

set	tag	block	
0	00	Mem(0)	Mem(1)
0	00	Mem(2)	Mem(3)

Table 6: Accessing word number 3 : hit

set	tag	block	
0	01	Mem(4)	Mem(5)
0	00	Mem(2)	Mem(3)

Table 7: Accessing word number 4: miss

set	tag	block	
0	01	Mem(4)	Mem(5)
0	00	Mem(2)	Mem(3)

Table 8: Accessing word number 3: hit

set	tag	block	
0	01	Mem(4)	Mem(5)
0	00	Mem(2)	Mem(3)

Table 9: Accessing word number 4 : hit

set	tag	block	
0	01	Mem(4)	Mem(5)
0	11	Mem(14)	Mem(15)

Table 10: Accessing word number 15 : miss

Little-endian answer:

set	tag	block	
0	00	Mem(1)	Mem(0)
1			

Table 11: Accessing word number 0 : miss

set	tag	block	
0	00	Mem(1)	Mem(0)
1			

Table 12: Accessing word number 1: hit

set	tag	block	
0	00	Mem(1)	Mem(0)
0	00	Mem(3)	Mem(2)

Table 13: Accessing word number 2 : miss

set	tag	block	
0	00	Mem(1)	Mem(0)
0	00	Mem(3)	Mem(2)

Table 14: Accessing word number 3 : hit

set	tag	block	
0	01	Mem(5)	Mem(4)
0	00	Mem(3)	Mem(2)

Table 15: Accessing word number 4: miss

set	tag	block	
0	01	Mem(5)	Mem(4)
0	00	Mem(3)	Mem(2)

Table 16: Accessing word number 3: hit

set	tag	block	
0	01	Mem(5)	Mem(4)
0	00	Mem(3)	Mem(2)

Table 17: Accessing word number 4 : hit

set	tag	block	
0	01	Mem(5)	Mem(4)
0	11	Mem(15)	Mem(16)

Table 18: Accessing word number 15 : miss