CMPE 110 Computer Architecture
Fall 2017, Homework #3

Computer Engineering
UC Santa Cruz

November 20, 2017

Submission Guidelines:

- This homework is due on Tuesday 11/28/2017.
- The homework must be submitted to Canvas by 11:59 pm.
  - Anything later is a late submission
- **Please include your name and your UCSC email address**
- **The homework should be "readable" without too much effort**
  - The homework must be typed and submitted as a single file in PDF format
  - Please name your homework file cmpe110-hw#-yourcruzid.pdf
  - Please keep your responses coherent and organized or you may lose points
- Provide details on how to reach a solution. **An answer without explanation gets no credit. Clearly state all assumptions.**
- Points: 64 = 24 + 24 + 16

Question 1. Cache Mapping and Overhead (24 Points)

In this problem, we will explore the different cache mapping schemes, and analyse how they impact hardware design choices. You are to design a 256-KByte byte addressable cache, with 16-word cachelines, with 4-Byte word size. The size of the address is 32-bits. Using these values, you are to design the following.

Question 1.A Direct Mapped (8 Points)

Assume the cache is Direct Mapped. Using the information given to you, calculate the number of bits used for the offset, index, and tag, and fill the table below. Calculate the tag overhead, using 1 bit for Valid, and the number of bits you calculated for the tag.

Field	**Size (bits)**
Offset	
Index	
Tag	

Question 1.B Fully-Associative (8 Points)

Assume the cache is fully-associative. Using the information given to you, calculate the number of bits used for the offset, index, and tag, and fill the table below. Calculate the tag overhead, using 1 bit for Valid, and the number of bits you calculated for the tag.

Field	**Size (bits)**
Offset	
Index	
Tag	

Question 1.C 4-Way Set-Associative (8 Points)

Assume the cache is 4-way set-associative. Using the information given to you, calculate the number of bits used for the offset, index, and tag, and fill the table below. Calculate the tag overhead, using 1 bit for Valid, and the number of bits you calculated for the tag.

Field	**Size (bits)**
Offset	
Index	
Tag	

Question 2. Cache Misses and Latency (24 Points)

In this question, we will explore the different types of cache misses, and see how they affect memory latency. Consider the MIPS assembly code shown below, which contains the address and content of each instruction. The instruction cache is Direct Mapped with eight 16-Byte cache lines.

Address	Instruction	Iteration 1	Iteration 2
	`loop:`		
0x108	`addiu r1, r1, -1`		
0x11c	`addiu r2, r2, 1`		
0x110	`j foo`		
	`...`		
	`foo:`		
0x218	`addiu r6, r6, 1`		
0x21c	`bne r1, r0, loop`		

The register R1 is initially set to 32, which means there will be 32 iterations of this loop.

Question 2.A Catagorizing Cache Misses (14 Points)

Assume that the cache is initially empty. For the first two iterations of the loop, fill the appropriate column (iteration 1 and iteration 2), with the appropriate type of miss that will occur (compulsory, conflict, capacity), or leave it blank if you think there is a cache hit. Rank the type of misses based on their frequency(most common to least common) for the above code.

Question 2.B Latency (10 Points)

Based on the code shown above, calculate the instruction cache miss rate for 32 iterations of the loop. Also calculate the average instruction cache access latency for 32 iterations of the loop, in terms of cycles, given that the hit time is 1 cycle and miss penalty is 6 cycles. Show your work, as this calculation has more than one step.

Question 3. Average Memory Access Time (16 Points)

Consider two processors with different ISAs with the following configurations:

Property	**Computer A**	**Computer B**
ISA	ARM	x86
Clock Frequency	3 GHz	4 GHz
Base CPI	1 cycle/instruction	2 cycles/instruction
L1 Instruction Cache Hit Time	1 cycle	1 cycle
L1 Instruction Cache Miss Rate	3%	2%
L1 Data Cache Hit Time	1 cycle	1 cycle
L1 Data CAche Miss Rate	9%	5%
L2 Hit Time	15 cycles	12 cycles
L2 Global Miss Rate	3%	4%
Main Memory Access Time	250 cycles	300 cycles

Question 3.A Ideal System (8 Points)

Assuming there is perfect branch prediction and ideal memory sub-system, which processor is faster, and by how much(speedup), given that ARM programs have 1.5x as may instructions as x86? In an ideal memory sub-system, memory accesses never miss (100% of memory accesses hit L1).

Question 3.B Full system (8 Points)

Now assume that both processors have all the features listed in the table above(separate instruction and data L1 caches, and unified L2 cache). If 30% of instructions are loads/stores, which processor is faster, and what is the speedup?