CMPE 110 Computer Architecture
Winter 2016, Homework #3

Computer Engineering
UC Santa Cruz

February 15, 2016

Name: Gurpreet Singh

Email: gsingh13@ucsc.edu

Submission Guidelines:

- This homework is due on Friday 2/26/16by 11:59pm.
- The homework must be submitted only on ecommons. Late submission are accepted on ecommons till2/28/16, 11:59pmwith penalty. Do not email your HW to the instructor or TAs.
- **Please write 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 or .doc format
  - Please name your homework file cmpe110-hw3-yourcruzid.pdf
- Provide details on how to reach a solution. **An answer without explanation gets no credit. Clearly state all assumptions.**
- Points: 78 = 18 + 30 + 30

Question	Part A	Part B	Part C	Total
1				
2				
3				
Total				

Question 1. Cache Mapping and Access (18 points)

Consider a 1 MB cache with 16-word cachelines. Each word is 4-Bytes. This cache uses write-back scheme, and the address is 32 bits wide.

Question 1.A Direct-Mapped Cache Fields (6 points)

Assume the cache is direct-mapped. Fill in the table below to specify the size of each address field of the cache.

Field	Size (bits)
Cacheline Offset	
Cacheline Index	
Tag	

- Cacheline offset = $\log_2(16*4) = 6$
- Cachline Index = $\log_2(\frac{1}{16})*4 = 2$
- Teg = 32 – 6 – 2 = 24 bits

Question 1.B Fully-Associative Cache Fields (6 points)

Assume the cache is fully-associative. Fill in the table below to specify the size of each address field of the cache.

Field	Size (bits)
Cacheline Offset	
Cacheline Index	
Tag	

- Cacheline Offset = $\log_2(16*4) = 6$
- Chacheline Index = Index field not used since cache is fully associated
- Tag = 32 – 6 = 26 bits

Question 1.C 16-Way Set-Associative Cache Fields (6 point)

Assume the cache is 16-way set-associative. Fill in the table below to specify the size of each address field.

Field	Size (bits)
Cacheline Offset	
Cacheline Index	
Tag	

- Cacheline Offset = $\log_2(16*4) = 6$
- Cachline Index = $\log_2(\frac{1}{16}*4*16) = -2$
- Tag = 32 – 6 + 2 = 28 bits

Question 2. Average Memory Access Time(30 points)

Assume the following processor configuration: a dedicated L1 cache for instructions (IL1) and a L1 cache for data (DL1), a shared L2 cache that serves as an intermediate level between each of the L1 caches and the main memory. The figure below shows the hierarchy:

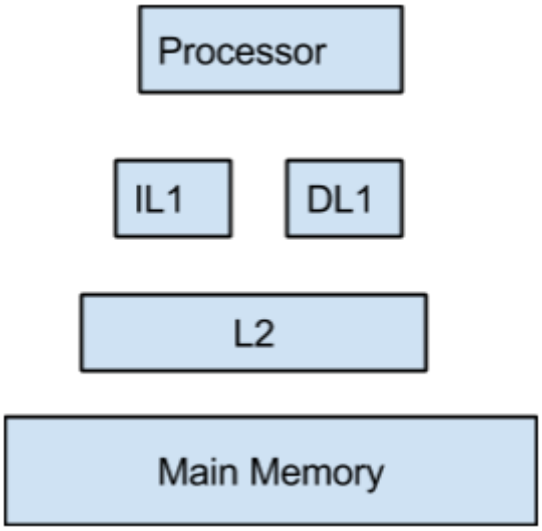

Figure 1: Cache Hierarchy

Processor Spec	Computer A
Cycle Time	1ns
Hit Time to L1(I-L1 or D-L1) and return the data to the processor	1 cycle
IL1 miss rate	8%
DL1 miss rate	15%
Hit Time to L2 and return the data to the L1(I-L1 or D-L1)	6 cycles
L2 miss rate	30%
Main Memory Access Time from L2	50 cycles

Figure 2: Table containing the processor cache Specifications

Out of the total instructions executed in this processor, **assume load/store instructions comprise of 25% of the total instructions.** Answer the following questions.

Question 2.A Calculate AMAT (10 points)

What is the average memory access time?

- Average memory access time (AMAT)
  - AMAT = thit(L1) + Miss%(L1) * tmiss(L1) = thit(L1) + Miss%(L1) * (thit(L2) + Miss%(L2)*tmiss(L2)) and it must split the tmiss into I$ miss and D$ miss
  - We then have, AMAT = 1 + 8%(6+35%*50) + 25%*(8%*(6+35%*50) = 3.1696 cycles

Question 2.B Processor with no caches (10 points)

Assume CPI = 1 if the processor has no memory stalls. Without the caches, each memory access would take 52 cycles. What is the CPI of the processor without any caches?

- CPI = CPIbase + miss%(L1I) * tmiss(L1I) + miss%(L1D) * tmiss(L1D)
- CPI = 1 + (8%*52) + (25%*8%*52) = 6.3

Question 2.C Processor with two levels of caches (as in Figure 1) (10 points)

Assume CPI = 1 if the processor has no memory stalls. What is the CPI of processor with the all the caches? Remember that it takes 50 cycles to access memory from L2.

- CPI = thit(L1) + Miss%(L1) * tmiss(L1) = thit(L1) + Miss%(L1) * (thit(L2)+Miss%(L2)*tmiss(L2)) → split the tmiss into I$ miss and D$ miss
- CPI = 1 + 8%(6+35%(50)) + 25%(8%(6+35%(50)) = 3.1696

Question 3. Two-Cycle Instruction Cache (30 points)

In this problem, we will be examining the performance of the instruction cache on the MIPS assembly program shown on the next page. The first column shows the instruction address for each instruction. Note that these addresses are byte addresses. The value of r1 is initially 64, meaning that there are 64 iterations in the loop. In this problem, we will be considering the execution of this loop with a direct-mapped instruction cache microarchitecture with eight cache lines, and each cacheline is 16B. This means each cache line can hold four instructions and the bottom four bits of an instruction address are the block offset. **Hint**: The first instruction in the code segment (i.e., `addiu r1, r1, -1`), is in the middle of a cache line with starting address 0x100.

For this problem, the instruction cache hit time is two cycles, but it is fully pipelined. Tag check occurs in the first cycle, and if it is a hit, the instruction is read in the second cycle. Essentially, this creates a six-stage pipelined processor with the following stages: instruction cache tag check (F0), instruction cache data access (F1), decode (D), execute (X), memory (M), and write-back (W). This also implies the data cache hit time is one cycle.

Assume that jumps are resolved in the decode stage and that branches are resolved in the execute stage. Assume the miss penalty is three cycles so on a cache miss the pipeline will stay in F0 for a total of three cycles, go into F1 for one cycle, and then continue as normal. You should assume that in every other way, the processor pipeline follows the classic fully-bypassed five-stage pipeline. Assume that the processor does not include a branch delay slot. Assume the processor speculatively predicts all jumps and branches are not taken.

Question 3.A Control Hazards (4 points)

Draw a pipeline diagram illustrating the first iteration of the loop assuming there are no instruction cache misses. Remember that there are two fetch stages (F0 and F1). Show stalls by simply repeating the pipeline stage character (e.g., D) for multiple consecutive cycles.

Use a dash (-) to indicate pipeline bubbles caused by killing instructions (pipeline flushes). You should show all instructions in the first iteration of the loop and the first instruction of the second iteration that you can properly draw the control dependency for the backwards branch.

Address	Instruction	Q3.B Iteration 1 ICache Miss Type	Q3.C Iteration 2 ICache Miss Type
	loop:		
0x108	addiu r1, r1, -1	Compulsory	
0x10c	j foo	Compulsory	Conflict
0x110	addiu r2, r2, 1		
0x114	addiu r3, r3, 1		
0x118	addiu r4, r4, 1		
0x11c	addiu r5, r5, 1		
	...		
	foo:		
0x218	bgtz r1, loop		
0x21c	addiu r6, r6, 1	Compulsory	Conflict
0x220	addiu r7, r7, 1		
0x224	addiu r8, r8, 1		
0x228	addiu r9, r9, 1		

Clock	1	2	3	4	5	6	7	8	9
I1 B [pc, #x]	F	D	C	M	W				
I2		F	D	X					
I3			F	X					
I1+x				F	D	C	M	W	
I2+x					F	D	C	M	W

Question 3.B First Iteration of the Loop (6 points)

Fill in the table above. In the appropriate column, write *compulsory*, *conflict*, or *capacity* next to each instruction which misses in the instruction cache to indicate the type of instruction cache misses that occur in the first iteration of the loop. **Assume that the instruction cache is initially completely empty.** Now draw a pipeline diagram illustrating the first iteration of the loop including instruction cache misses. Clearly indicate the number of cycles it takes to execute the first iteration.

Loop	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
addiu r1, r1, -1	F	D	-	-	-										
1 foo		F	D	-	-	-									
addiu r2, r2, 1			F	D	E	M	W								
addiu r3, r3, 1				F	D	E	M	W							
addiu r4, r4, 1					F	D	E	M	W						
addiu r5, r5, 1						F	D	E	M	W					
…															
foo:															
bgtz r1, loop							F	-	-	-	-				
addiu r6, r6, 1								F	D	E	M	W			
addiu r7, r7, 1									F	D	E	M	W		
addiu r8, r8, 1										F	D	E	M	W	
addiu r9, r9, 1											F	D	W	M	W

- Execution Time: 15cycles * 11instructions = 165 cycles

Question 3.C Second Iteration of the Loop and overall CPI (20 points)

Continue to fill in the table above. Write *compulsory*, *conflict*, or *capacity* next to each instruction which misses in the instruction cache to indicate the type of instruction caches misses that occur in the second iteration of the loop. Now draw a pipeline diagram illustrating the second iteration of the loop. Clearly indicate the number of cycles it takes to execute the second iteration. Calculate the CPI for this processor executing all 64 iteration of the loop **(Note: The CPI calculation should not include instructions that are fetched but then later squashed).**

Loop	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
addiu r1, r1, -1	F	-	-	-	-										
1 foo		F	-	-	-	-									
addiu r2, r2, 1			F	D	E	M	W								
addiu r3, r3, 1				F	D	E	M	W							
addiu r4, r4, 1					F	D	E	M	W						
addiu r5, r5, 1						F	D	E	M	W					
…															
foo:															
bgtz r1, loop							F	-	-	-	-				
addiu r6, r6, 1								F	D	E	M	W			
addiu r7, r7, 1									F	D	E	M	W		
addiu r8, r8, 1										F	D	E	M	W	
addiu r9, r9, 1											F	D	W	M	W

- Execution Time: 15cycles * 64 iterations = 960 cycles