COE 308 – Computer Architecture

Final Exam – Fall 2008

Saturday, February 7, 2009

7:30 – 10:00 AM

Computer Engineering Department
College of Computer Sciences & Engineering
King Fahd University of Petroleum & Minerals

Student Name: **SOLUTION**

Student ID: _____

Q1	/ 20	Q2	/ 15
Q3	/ 20	Q4	/ 20
Q5	/ 25		
Total	/ 100		

Important Reminder on Academic Honesty

Using unauthorized information on an exam, peeking at others work, or altering graded exams to claim more credit are severe violations of academic honesty. Detected cases will receive a failing grade in the course.

Q1. (20 points) True or False? Explain or give the right answer for a full mark.

a) On a read, the value returned by the cache depends on which blocks are in the cache.

False, it depends on the last value written to the same memory location

b) Most of the cost of the memory hierarchy is at the L1 cache.

False, most of the cost is at the L2 cache (takes more area on chip)

c) The higher the memory bandwidth, the larger the cache block size should be.

True, if the memory bandwidth is high then a larger block size can be transferred in same amount of time

d) In reducing cache misses, capacity is more important than associativity.

True, increasing the capacity eliminates more cache misses than higher associativity

e) Compulsory cache misses can be reduced.

True, compulsory misses can be reduced with prefetching

f) Allowing ALU and branch instructions to take fewer stages and complete earlier than other instructions does not improve the performance of a pipeline.

True, pipeline performance is related to throughput, not the latency of instructions

g) Increasing the depth of pipelining by splitting stages always improves performance.

False, not always, at some point pipeline register delays become significant, and more bubbles or stall cycles must be introduced if the pipeline depth is increased

h) The single-cycle datapath must have separate instruction and data memories because the format of instructions and data is different.

False reason, it is because both memories should be accessed during the same cycle and both are single ported.

i) A given application runs in 15 seconds. A new compiler is released that requires only 0.6 as many instructions as the old compiler. Unfortunately, it increases the CPI by 1.1. We expect the application to run using this new compiler in $15 \times 0.6/1.1 = 8.18$ sec

False, it should be $15 \times 0.6 \times 1.1 = 9.9$ sec

j) If Computer A has a higher MIPS rating than computer B, then A is faster than B.

False, it is possible to have higher MIPS rating and worse execution time.

Q2. (15 pts) Consider a **direct-mapped** cache with **128 blocks**. The block size is **32 bytes**.

a) (3 pts) Find the number of tag bits, index bits, and offset bits in a 32-bit address.

Offset bits = 5

Index bits = 7

Tag bits = 32 – 12 = 20 bits

b) (4 pts) Find the number of bits required to store all the valid and tag bits in the cache.

Total number of tag and valid bits = 128 * (20 + 1) = 2688 bits

c) (8 pts) Given the following sequence of address references in decimal:

20000, 20004, 20008, 20016, 24108, 24112, 24116, 24120

Starting with an **empty cache**, show the **index** and **tag** for each address and indicate whether a hit or a miss.

Address = Hex	**Offset (5 bits)**	**Index (7 bits)**	**Tag**	**Hit or Miss**
20000 = 0x4E20	0x00 = 0	0x71 = 113	4	Miss (initially empty)
20004 = 0x4E24	0x04 = 4	0x71 = 113	4	Hit
20008 = 0x4E28	0x08 = 8	0x71 = 113	4	Hit
20016 = 0x4E30	0x10 = 16	0x71 = 113	4	Hit
24108 = 0x5E2C	0x0C = 12	0x71 = 113	5	Miss (different tag)
24112 = 0x5E30	0x10 = 16	0x71 = 113	5	Hit
24116 = 0x5E34	0x14 = 20	0x71 = 113	5	Hit
24120 = 0x5E38	0x18 = 24	0x71 = 113	5	Hit

Q3. (20 pts) A processor runs at 2 GHz and has a CPI of 1.2 without including the stall cycles due to cache misses. Load and store instructions count 30% of all instructions.

The processor has an I-cache and a D-cache. The hit time is 1 clock cycle. The I-cache has a 2% miss rate. The D-cache has a 5% miss rate on load and store instructions.

The miss penalty is 50 ns, which is the time to access and transfer a cache block between main memory and the processor.

a) (3 pts) What is the average memory access time for instruction access in clock cycles?

Miss penalty = 50 ns * 2 GHz = 100 clock cycles

AMAT = hit time + miss rate * miss penalty = 1 + 0.02 * 100 = 3 clock cycles

b) (3 pts) What is the average memory access time for data access in clock cycles?

AMAT = 1 + 0.05 * 100 = 6 clock cycles

c) (4 pts) What is the number of stall cycles per instruction and the overall CPI?

Stall cycles per instruction = 1 * 0.02 * 100 + 0.3 * 0.05 * 100 = 3.5 cycles

Overall CPI = 1.2 + 3.5 = 4.7 cycles per instruction

Suppose we add now an L2 cache that has a hit time of 5 ns, which is the time to access and transfer a block between the L2 and the L1 cache. Of all the memory references sent to the L2 cache, 80% are satisfied without going to main memory.

d) (4 pts) What is the average memory access time for instruction access in clock cycles?

Hit time in the L2 cache = 5 ns * 2 GHz = 10 clock cycles

AMAT = 1 + 0.02 * (10 + 0.2 * 100) = 1.6 clock cycles

e) (4 pts) What is the number of stall cycles per instruction and the overall CPI?

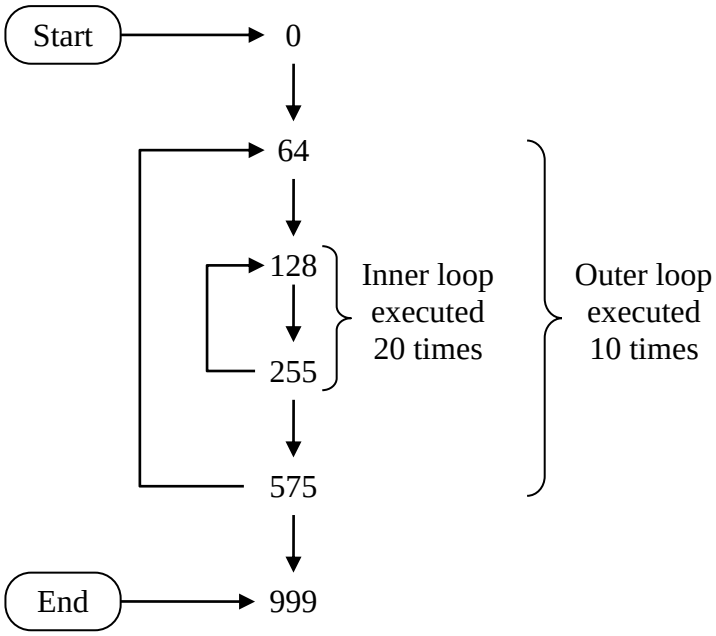
Stall cycles per instruction = (1 * 0.02 + 0.3 * 0.05) * (10 + 0.2 * 100) = 1.05 cycles

Overall CPI = 1.2 + 1.05 = 2.25 cycles per instruction

f) (2 pts) How much faster will the machine be after adding the L2 cache?

Speedup = CPI_c / CPI_e = 4.7 / 2.25 = 2.09

Q4. (20 pts) A program consists of two nested loops: a smaller inner loop and a larger outer loop. The general structure of the program is shown below. All memory addresses are shown in decimal. Each decimal address points to an instruction in memory. All memory locations in the various sections contain instructions to be executed in straight-line sequencing. Instructions 128→255 form the inner loop and are repeated 20 times for each pass of the outer loop. Instructions 64→127 and 256→575 form the outer loop and are repeated 10 times. The program is to be run on a computer that has a direct-mapped instruction-cache with 64 blocks, where each block can store 4 instructions. The hit time is 1 clock cycle and the miss penalty is 20 cycles.



a) (6 pts) What is the total instruction count and how many I-cache misses are caused by the program?

0 → 63: 64 instruction = 16 I-blocks => 16 I-cache misses

64 → 127: 64 instructions = 16 I-blocks => 16 I-cache misses

128 → 255: 128 instructions = 32 I-blocks => 32 I-cache misses (all 20 iterations)

Inner loop can fit inside I-cache (only first pass causes I-cache misses)

256 → 575: 320 instructions = 80 I-blocks => 80 I-cache misses

Outer loop I-cache misses = 10 * (16 + 32 + 80) = 1280

576 → 999: 424 instruction = 106 I-blocks => 106 I-cache misses

Total I-cache misses = 16 + 1280 + 106 = 1402

Total I-count = 64 + 64*10 + 128*20*10 + 320*10 + 424 = 29,928

b) (2 pts) What is the I-cache miss rate?

Miss Rate = 1402 / 29,928 = 0.0468 = 4.68%

c) (4 pts) If only cache misses stall the processor, what is the execution time (in nanoseconds) of the above program on a 2 GHz pipelined processor?

Total clock cycles = 29,928 + 1402 * 20 = 57,968 cycles

Execution time = 57,968 * 0.5 ns = 28,984 ns

d) (8 pts) Repeat (a) thru (c) if a bigger block size that can store 8 instructions is used. The total number of blocks in the I-cache is still 64. What is the speedup factor?

0 → 63: 64 instruction = 8 I-blocks => 8 I-cache misses

64 → 127: 64 instructions = 8 I-blocks => 8 I-cache misses

128 → 255: 128 instructions = 16 I-blocks => 16 I-cache misses (all 20 iterations)

Inner loop can fit inside I-cache. Only first pass causes I-cache misses

256 → 575: 320 instructions = 40 I-blocks => 40 I-cache misses

Outer loop I-cache misses = 8 + 16 + 40 = 64

Outer loop can fit inside I-cache. Only first pass causes I-cache misses

576 → 999: 424 instruction = 53 I-blocks => 53 I-cache misses

Total I-cache misses = 8 + 64 + 53 = 125

Total I-count = still the same = 29,928

I-cache Miss Rate = 125 / 29,928 = 0.418%

Total clock cycles = 29,928 + 125 * 20 = 32,428

Execution time = 32,428 * 0.5 ns = 16,214 ns

Speedup = 28984 / 16214 = 1.79

Q5 (25 pts) Use the following MIPS code fragment:

```
I1:    ADDI    $3, $0, 100          # $3 = 100
I2:    ADD     $4, $0, $0           # $4 = 0
Loop:
I3:    LW      $5, 0($1)            # $5 = MEM[$1]
I4:    ADD     $4, $4, $5           # $4 = $4 + $5
I5:    LW      $6, 0($2)            # $6 = MEM[$2]
I6:    SUB     $4, $4, $6           # $4 = $4 - $6
I7:    ADDI    $1, $1, 4            # $1 = $1 + 4
I8:    ADDI    $2, $2, 4            # $2 = $2 + 4
I9:    ADDI    $3, $3, -1           # $3 = $3 - 1
I10:   BNE     $3, $0, Loop         # if ($3 != 0) goto Loop
```

a) (10 pts) Show the timing of one loop iteration on the 5-stage MIPS pipeline **without forwarding hardware**. Complete the timing table, showing all the stall cycles. Assume that the branch will stall the pipeline for 1 clock cycle only.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
I1: ADDI	**IF**	**ID**	**EX**	**M**	**WB**																				
I2: ADD		**IF**	**ID**	**EX**	**M**	**WB**																			
I3: LW			**IF**	**ID**	**EX**	**M**	**WB**																		
I4: ADD				**IF**	**stall**	**stall**	**ID**	**EX**	**M**	**WB**															
I5: LW					2 stall cycles		**IF**	**ID**	**EX**	**M**	**WB**														
I6: SUB								**IF**	**stall**	**stall**	**ID**	**EX**	**M**	**WB**											
I7: ADDI									2 stall cycles		**IF**	**ID**	**EX**	**M**	**WB**										
I8: ADDI												**IF**	**ID**	**EX**	**M**	**WB**									
I9: ADDI													**IF**	**ID**	**EX**	**M**	**WB**								
I10: BNE														**IF**	**stall**	**stall**	**ID**								
I3: LW															2 stall cycles		**IF**	**IF**	**ID**	**EX**	**M**	**WB**			
I4: ADD																1 delay cycle			**IF**	**stall**	**stall**	**ID**	**EX**	**M**	**WB**

Time of one loop iteration = 15 cycles

b) (5 pts) According to the timing diagram of part **(a)**, compute the number of clock cycles and the average CPI to execute ALL the iterations of the above loop.

There are 100 iterations
Each iteration requires 15 cycles =
8 cycles to start the 8 instructions in loop body + 7 stall cycles
There are 2 additional cycles to start the first 2 instructions before the loop.
Therefore, total cycles = 100 * 15 + 2 (can be ignored) = 1502 cycles ≈ 1500 cycles
Total instruction executed = 2 + 8 * 100 = 802 instructions (counting first two)
Average CPI = 1502 / 802 = 1.87
If we ignore first two instructions and the time to terminate last iteration then
Average CPI = 1500/800 = 1.88 (almost same answer)

c) (5 pts) Reorder the instructions of the above loop to fill the load-delay and the branch-delay slots, without changing the computation. Write the code of the modified loop.

```
    ADDI    $3, $0, 100     # $3 = 100
    ADD     $4, $0, $0      # $4 = 0
Loop:
    LW      $5, 0($1)       # $5 = MEM[$1]
    LW      $6, 0($2)       # Moved earlier to avoid load-delay
    ADDI    $3, $3, -1      # Moved earlier
    ADD     $4, $4, $5      # $4 = $4 + $5
    ADDI    $1, $1, 4       # $1 = $1 + 4
    ADDI    $2, $2, 4       # $2 = $2 + 4
    BNE     $3, $0, Loop    # if ($3 != 0) goto Loop
    SUB     $4, $4, $6      # Fills branch delay slot
```

Other re-orderings are possible as long as we avoid the load delay and we fill branch delay slot with an independent instruction. We should be able to reduce the stall cycles to 0.

d) (5 pts) Compute the number of cycles and the average CPI to execute ALL the iteration of the modified loop. What is the speedup factor?

There are 100 iterations
Each iteration requires 8 cycles =
8 cycles to start the 8 instructions in loop body + 0 stall cycles
There are 2 additional cycles to start the first 2 instructions before the loop
+ 4 additional cycles to terminate the ADDI instruction in the last iteration.
Therefore, total cycles = 100 * 8 + 6 (can be ignored) = 806 cycles ≈ 800 cycles
Total instruction executed = 2 + 8 * 100 = 802 instructions (counting first two)
Average CPI = 806 / 802 = 1.00
If we ignore first two instructions and the time to terminate last iteration then
Average CPI = 800/800 = 1.00 (almost same answer)
Speedup Factor = $CPI_{part\text{-}b}/CPI_{part\text{-}d}$ = 1.88/1.00 = 1.88