ICS 233 - Computer Architecture & Assembly Language

Final Exam – Fall 2007

Wednesday, January 23, 2007
7:30 am – 10:00 am

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

Student Name: ________________________________

Student ID: ________________________________

Q1	/ 10	Q2	/ 20
Q3	/ 15	Q4	/ 15
Q5	/ 20	Q6	/ 25
Total	/ 105		

Important Reminder on Academic Honesty

Using unauthorized information or notes 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. (10 pts) Consider the following MIPS code sequence:

```
lw  $5, 100($2)
add $2, $3, $5
sub $5, $5, $2
sw  $5, 100($2)
```

a) (5 pts) Identify all the RAW dependencies between pairs of instructions.

b) (3 pts) Identify all the WAR dependencies between pairs of instructions

c) (2 pts) Identify all the WAW dependencies between pairs of instructions

Q2. (20 pts) We have a program core consisting of five conditional branches. The program core will be executed millions of times. Below are the outcomes of each branch for one execution of the program core (T for taken and N for not taken).

Branch 1: T-T-T-T-T
Branch 2: N-N-N
Branch 3: T-N-T-N-T-N-T-N
Branch 4: T-T-T-N-N-N
Branch 5: T-T-T-N-T-T-T-N-T

Assume that the behavior of each branch remains the same for each program core execution. For dynamic branch prediction schemes, assume that each branch has its own prediction buffer and each buffer is initialized to the same state before each execution. List the predictions and the accuracies for each of the following branch prediction schemes:

a) Always taken
b) Always not taken
c) 1-bit predictor, initialized to predict taken
d) 2-bit predictor, initialized to weakly predict taken

Q3. (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.

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

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.

Q4. (15 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?

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

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

d) (5 pts) You are considering replacing the 2 GHz CPU with one that runs at 4 GHz, but is otherwise identical. How much faster does the new processor run? Assume that hit time in the I-cache and the D-cache is 1 clock cycle in the new processor, and the time to access and transfer a cache block between main memory and the processor is still 50 ns.

Q5. (20 pts) Consider the following idea: we want to modify all **load** and **store** instructions in the instruction set such that the offset is always 0. The **load** and **store** instructions can be of the R-type and there is NO need for the ALU to compute the memory address. This means that all load and store instructions will have the following format, where **Rs** is the register that contains the memory address.

```
LW  Rt, (Rs)    # No immediate constant used
SW  Rt, (Rs)    # No immediate constant used
```

a) (10 pts) Draw the modified **single-cycle** datapath. **Identify the changes** that you are making to the single-cycle datapath.

b) (4 pts) Assume that the operation delays for the major components are as follows:

Instruction Memory: 200 ps

Data Memory: 200 ps

ALU: 150 ps

Register file (read or write): 100 ps

Ignore the delays in the multiplexers, control, PC access, extension logic, and wires.

What is the cycle time for the single-cycle datapath BEFORE and AFTER making the modification?

c) (6 pts) Because we have removed the offset in all load and store instructions, all original load-store instructions with non-zero offsets would now require an additional **ADDI** instruction to compute the address. This will increase the instruction count.

Suppose we have a program in which 20% of the instructions are load-store instructions. Assume further that only 10% of the original load-store instructions have a non-zero offset and would require an additional **ADDI** instruction to compute the address.

What is the percent increase in the instruction count when additional **ADDI** instructions are used?

Which design is better, the original one that allowed non-zero offsets, or the modified one with zero offsets, and why?

What is the speedup factor?

Q6. (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																									
I3: LW																									
I4: ADD																									
I5: LW																									
I6: SUB																									
I7: ADDI																									
I8: ADDI																									
I9: ADDI																									
I10: BNE																									
I3: LW																									
I4: ADD																									

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.

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.

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?

Additional Page if Needed