**Homework 4 (Due: Nov 22)**

# **Instructions**

* Complete the following questions to the best of your ability.
* Answers should be clear, concise, and justified with work.
* Please write your name and NetID on the hardcopy of your solution, and bring it to the lecture on the due day. Please submit your solution before the lecture starts.

# **Collaboration**

* Professor/TAs
  + You **may** discuss any content with any professor or TA.
* Students
  + You **may** discuss high-level concepts, techniques, jargon, keywords, related problems, or course content that is relevant to the material.
  + You **may not** discuss particular solutions to any of the questions.
  + If you have substantial conversations with any students, please note their name and, if you feel it necessary, the extent of your collaboration.
* Outside Sources (Internet, books)
  + You **may** use external references for any course content.
  + You **may** use an external tool to verify your solutions where appropriate, but not to generate solutions to any questions.
  + List any outside sources that you use. Formal citations are not necessary; links are fine.

# **Question 1 (20 points)**

Consider the following code:

1: addi $1, $0, 56

2: addi $2, $0, 112

3: sub $2, $2, 1

4: bne $2, $1, 2

5: lw $2, 0($3)

6: sw $1, 0($3)

a) Fill out the table below by stating the type of register data dependency (i.e., write-after-read, read-after-write, write-after-write, or read-after-read) that exists between the two “Dependent Instructions” and concerns the “Concerned Register”. Note that some data dependencies might be missing in the table; do not include new rows for those dependencies.

|  |  |  |
| --- | --- | --- |
| Dependent Instructions | Concerned Register | Type of Data Dependency |
| 1-2 | 0 | Read-after-read |
| 1-4 | 1 | Read-after-write |
| 2-3 | 2 | Read-after-write |
| 2-3 | 2 | Write-after-write |
| 2-4 | 2 | Write-after-write |
| 3-4 | 2 | Read-after-read |
| 3-4 | 2 | Read-after-write |
| 4-5 | 2 | Write-after-read |
| 4-6 | 1 | Write-after-read |
| 5-6 | 3 | Read-after-read |

b) Early versions of MIPS processors did not implement hardware interlocks. In those cases, data dependencies would be checked in software and NOPs would be inserted. **Write down** the minimum number of NOPs that need to be inserted to avoid data hazards. **Rewrite** the above sequence of instructions with the proper NOPs in place. You should not modify the instructions’ order (no rescheduling). Assume a five-stage pipeline, where jumps/branches are computed in the execute stage, but latched and resolved in the memory stage; no multiplier/divider unit; all registers are initialized to 0; and the value of $0 is always 0. We will also assume that there is no bypass, stall, or flush logic.

1: addi $1, $0, 56

2: addi $2, $0, 112

3: nops

4: nops

5: sub $2, $2, 1

6: nops

7: nops

8: bne $2, $1, 2

9: lw $2, 0($3)

10: sw $1, 0($3)

# **Question 2 (15 points)**

The following program is running on a MIPS machine **with stall and flush logic but no bypassing.** Fill in the table below to indicate which stage of the traditional 5-stage pipeline each instruction is in at each clock cycle. The first instruction is filled in as an example. Assume a five-stage pipeline, where jumps/branches are computed in the execute stage, but latched and resolve in the memory stage; no multiplier/divider unit; all registers are initialized to 0; and the value of $0 is always 0. Use the conventions in the Pipelining slides #38 and #51 to show your stalls (adding stars) and flushing (crossing off instructions while still showing where they reached in the pipeline).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| lw $r3, 0($r1) | F | D | X | M | W |  |  |  |  |  |  |  |  |
| addi $r2, $r1, 6 |  | F | D | X | M | W |  |  |  |  |  |  |  |
| and $r4, $r3, $r5 |  |  | F | D | d\* | X | M | W |  |  |  |  |  |
| sub $r5, $r6, $r3 |  |  |  | F | d\* | D | X | M | W |  |  |  |  |
| sw $r5, 4($r1) |  |  |  |  |  | F | D | d\* | d\* | X | M | W |  |
| add $r1, $r1, $r4 |  |  |  |  |  |  | F | d\* | d\* | D | X | M | W |

# **Question 3 (30 points)**

You are designing the memory hierarchy for a processor which has a memory access latency of 150 ns.

1. The Level 3 cache can be accessed in 30 ns on a hit. What hit rate does it need to achieve an average access latency of 60ns?
2. Assuming the Level 3 cache achieves the hit rate in part 1 (thus its average access latency is 60ns), and that the Level 2 cache has 75% hit rate, what hit latency does the L2 cache need in order to achieve an average access time of 25 ns?

c) You have two choices for the Level 1 data cache design:

* Option A 64KB, 8-way set associative, 2.5 ns hit latency, 95% hit rate
* Option B 32KB, 4-way set associative, 1 ns hit latency, 90% hit rate

Assuming the 60ns L3 and 25ns L2 average access times above, which option would you pick? Why? Show your work.

* Option A:
* Option B:

**Because average time of option B is less than option A, I will pick option B**

# **Question 4 (35 points)**

A 16-byte cache has 4-byte blocks, has 2 sets, and is 2-way set-associative. The cache is initially empty (all valid bits are off: indicated by a blank box in the table below). The cache receives requests in the sequence listed below. For each address in the sequence (a) split it into the tag, index, and offset; (b) categorize the access as a hit, a compulsory miss, a conflict miss, or a capacity miss (You can abbreviate hit=H, Compulsory=O, Conflict=F, Capacity=P); (c) show the new contents of the cache after the access—write the tags for each way, and note which way is LRU. The first one is done for you:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Split Address | | | Result | Set 0 | | | Set 1 | | |
| Tag | Index | Offset | Way 0 | Way 1 | LRU Way | Way 0 | Way 1 | LRU Way |
| FF | 1F | 1 | 3 | O |  |  | 0 | 1F |  | 1 |
| 22 | 04 | 0 | 2 | O | 04 |  | 1 |  |  | 1 |
| BD | 17 | 1 | 1 | O | 04 |  | 1 |  | 17 | 0 |
| 31 | 06 | 0 | 1 | O | 04 | 06 | 0 |  | 17 | 0 |
| 42 | 08 | 0 | 2 | O | 08 | 06 | 1 |  | 17 | 0 |
| 41 | 08 | 0 | 1 | H | 08 | 06 | 1 |  | 17 | 0 |
| FC | 1F | 1 | 0 | O | 08 | 06 | 1 | 1F | 17 | 1 |
| 23 | 04 | 0 | 3 | P | 08 | 04 | 0 | 1F | 17 | 1 |
| 32 | 06 | 0 | 2 | P | 06 | 04 | 1 | 1F | 17 | 1 |
| 10 | 02 | 0 | 0 | O | 06 | 02 | 0 | 1F | 17 | 1 |
| BC | 17 | 1 | 0 | H | 06 | 02 | 0 | 1F | 17 | 0 |