Prof. Taylor’s
CSE 141 Winter 2014
Final Exam

Architect (your name here): ________________________________

General Instructions

- To receive full credit, write your answers as neatly as you can.
- Show your work. You may be wrong, but you could get partial credit.
- Be succinct but thorough. Say exactly what you mean, and explain it well. Extraneous stuff may count against your score.
- If you have a question, raise your hand and either the TA or the Prof will come over.
- This is a closed book exam.

You have ninety (90) minutes.
Work quickly and do not check your answers until you get to the end.

<table>
<thead>
<tr>
<th>Question</th>
<th>Time</th>
<th>Points</th>
<th>Your Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>45</td>
<td>400</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>1000</td>
<td></td>
</tr>
</tbody>
</table>
1. Imagine running the same program on a 3-stage pipeline rather than a 5-stage pipeline. Assuming there are no stalls in execution in either case, the new average CPI would be \( \frac{3}{5} \) of the original.

2. Google’s cloud is mostly deployed as Infrastructure as a service (IaaS) while Amazon’s EC2 is mostly software as a service (SaaS).

3. Warehouse-scale computers (WSC) are distinguished from data centers by being relatively heterogeneous in terms of the kinds of hardware being deployed.

4. A cache coherence protocol requires at least three states to work.

5. For disk-based systems, the latency of accessing a remote disk in a lightly-loaded WSC is almost the same as a local disk.

6. In the MESI protocol, if a cache line is in the dirty state on one core, and then another cache reads that line, the cache line will first be written back to DRAM.

7. In power-optimized WSC’s, the energy of CPU+DRAM can be almost 50% the total energy budget.

8. Large multiprocessors make sense for SaaS apps for which the number of nodes required is close to the size of the multiprocessor and where there are a significant number of non-local accesses.

9. Branch prediction is less important if the number of pipeline stages in the front end is small.

10. In a 5-stage MIPS pipeline, WAW (write after write) and WAR (write after read) dependencies on general purpose registers do not cause stalls.

11. In the 5-stage MIPS pipeline with full forwarding, the only data hazard that leads to a RAW (read after write) stall is one where result of a load instruction is used in the very next instruction.

12. In a multiprocessor architecture with a MESI protocol, only one processor may have an address in its cache in the modified state.

13. The best static branch predictors tend to perform better than the best dynamic branch predictors.

14. Prefetching data can help reduce capacity misses in a cache.

15. The page table is used to cache TLB entries.
16. Stack accesses have significant temporal locality, but not much spatial locality.

17. RAID-1 typically decreases the chances of data loss.

18. Writes on RAID-1 are likely to be faster than on RAID-0.

19. Caching is harder to get right on message passing multiprocessors than on shared memory multiprocessors.

20. Graphics processors attain speedup by having many small processors that are individually less powerful than the general purpose processor in your cell phone or laptops.
Question 2 [15 minutes, 200 points, points divided evenly]

Consider the following MIPS code:

Start:
  li R2, 17
Loop:  srl R2, R2, 1
  bgt R2, R0, Loop
  J Start

You may assume that R0 is hardwired to zero.

A. If we denote taken by T and not taken by N, what is the branching pattern (only consider bgt)? Explain.

B. If branch “always NOT taken” predictor is used, what is the steady-state (warmed up) mispredict rate?

C. If branch “always taken” predictor is used, what is the mispredict rate? Explain briefly.

D. What is the steady-state mispredict rate for a 1-bit dynamic predictor? Explain briefly.

E. What is the steady-state mispredict rate for a 2-bit dynamic predictor? Explain briefly.
F. What is the steady-state mispredict rate for a predictor that considers only global history? Assume the global history predictor has 5 bits of history and those bits are used to index into a table of thirty-two 2-bit counters. Assume at the start of the loop, the counters are all primed to 00, and the history is 1 1 1 1 0, where 0 is the most recent branch outcome (not-taken). What is the steady-state mispredict rate? What are the final steady state values of the counters?
**Question 3 [ 10 minutes , 200 points ]**

Consider a system with 14 bit virtual address and a 9 bit physical address with a page size of 128 bytes. There is just one level of page table.

A. What is the maximum addressable size of physical memory?

B. Fill the following bits with O if that bit is an offset bit, V if it belongs to Virtual Page Number.

<table>
<thead>
<tr>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

C. Fill the following bits with O if that bit is an offset bit, P if it belongs to Physical Page Number.

<table>
<thead>
<tr>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

D. How many entries and how many bits wide does the page table need to be at a minimum? Assume that each page table entry has a valid bit. Explain your answer.

E. If the system has a 7-way associative 896-byte cache, will it be easy or hard to implement virtually-indexed physically-tagged caching, and why?
Question 4 [ 45 minutes, 400 points ]

**el GOOG**, a large, secret multinational corporation (motto: “do no good”), has hired you to optimize the following code because it is the core of their “SPYMAST3R” web indexing algorithm.

```
loop:

lw $r2, 0($r4)
lw $r4, 4($r2)
lw $r6, 8($r4)
bnez $r2, loop
addu $r7, $r4, $r7
```

a. Given a standard 5-stage MIPS pipeline with one delay slot, with forwarding, draw the pipeline diagram that shows what stage each instruction is in on each cycle, for one and half iterations of the loop. (Instruction should be rows and cycles should be columns.) Draw arrows for bypasses. (75 points)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BNE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b. Compute the “steady state” CPI for the loop. That is, the CPI if the loop executed an infinite number of times. (75 points)
c. **(200 points)** If you can optimize the above code, it will improve latency and allow *El Goog* to spy on people even better. (You've been promised a small tropical island and your very own SR-71 in exchange for your services…) Ordinarily, a “pointer chasing” application like the above is very difficult to optimize because many of the instructions are serially dependent and some take multiple cycles to execute. However, you have a secret weapon: you know that three loads are always to three separate arrays (load #1 is to array #1, load #2 is to array #2, etc). To speedup up this code, your plan is to have three separate memories (M1..M3), one for each array, and to add three instructions, LW1, LW2, and LW3. Here is the code:

```
loop:
  lw1 $r2, 0($r4)
  lw2 $r4, 4($r2)
  lw3 $r6, 8($r4)
  bnez $r2, loop
  addu $r7, $r4,$r7
```

Design and draw a pipelined datapath, based on the 5-stage MIPS pipeline (but with more stages) that can attain a true steady-state CPI of 1.

Some hints:

- You only need to support the above instructions; do not draw logic for anything else.
- You only need to draw the datapath.
- Make registers very clear.
- Do not draw control.
- Each memory should use its own effective address adder.
- SRAM accesses must begin at the beginning of the cycle and take the full cycle.
- Adders take almost a full cycle; they may have bypass muxes on an input though.
- Adders cannot be placed in the same cycle as SRAMs.
- You should draw all bypass muxes that are used in the execution of the code.
(Q. 4C continued)
4D. **(50 points)** Show the new pipeline execution diagram for your new pipeline for the above code. Draw arrows to show the bypassed values. Remember, they can only go forward in time.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LW2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LW3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BNE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LW1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LW2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>