CS233 Exam 6
November 29th, 2016

Name: ________________________________________________________________
NETID: _____________________________________________________________________

Circle the section that you attend (so we can hand back your exam).

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>(11am-noon) Craig</td>
<td>(11am-noon) Chamila</td>
</tr>
<tr>
<td>(noon-1pm) Litian</td>
<td>(noon-1pm) Vishaal</td>
</tr>
<tr>
<td>(1-2pm) Litian</td>
<td>(1-2pm) Chamila</td>
</tr>
<tr>
<td>(2-3pm) Ed</td>
<td>(2-3pm) Geoffrey</td>
</tr>
<tr>
<td>(3-4pm) Vishaal</td>
<td>(3-4pm) Dustyn</td>
</tr>
<tr>
<td>(4-5pm) Ed</td>
<td>(4-5pm) Dustyn</td>
</tr>
</tbody>
</table>

- This exam has 12 pages; the final sheets are provided as reference and scratch paper, but must be handed in with the exam.
- You have 120 minutes.
- No calculators or other electronics are allowed. You may bring one 8.5” x 11” sheet of handwritten notes that must be handed in with the exam.
- To make sure you receive credit, please write clearly and show your work.
- We will not answer questions regarding course material.

<table>
<thead>
<tr>
<th>Question</th>
<th>Maximum</th>
<th>Your Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>
Question 1: Pipelining (40 points)

Above is a pipelined architecture that is similar to the one discussed in class except that operations associated with the ALU have been spread out over two pipelined execution stages: EX1 and EX2. In addition, there is no forwarding from EX1 to any stage. Branches are predicted as not-taken and are resolved in the MEM stage. Unconditional branches are resolved in the ID stage.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Cache</td>
<td>5 ns</td>
</tr>
<tr>
<td>Data Cache</td>
<td>18 ns</td>
</tr>
<tr>
<td>Control</td>
<td>3 ns</td>
</tr>
<tr>
<td>Register File Read</td>
<td>7 ns</td>
</tr>
<tr>
<td>Register File Write</td>
<td>10 ns</td>
</tr>
<tr>
<td>Sign-extend Unit</td>
<td>2 ns</td>
</tr>
<tr>
<td>ALU stage 1</td>
<td>15 ns</td>
</tr>
<tr>
<td>ALU stage 2</td>
<td>8 ns</td>
</tr>
<tr>
<td>Pipeline Register (write)</td>
<td>2 ns</td>
</tr>
</tbody>
</table>

**Part(a) (10 points)**

Given the structure latencies (shown at the left), what is the speed up achieved by this pipelining compared to a non-pipelined version (you may assume a large number of instructions have executed and that there are no stalls or flushes). **Explain.**

**Note: the pipeline registers have a 2ns write latency!**
**Question 1: Pipelining (cont.)**

**Part (b) (10 points)**
Annotate the following code to identify as many true data dependencies as you can.

```assembly
add $t0, $zero, 2
sub $t1, $t0, $t0

loop: sub $t0, $t0, 1
beq $t0, 0, done
and $t2, $t0, 1
beq $t2, 1, odd
j even

odd: add $t3, $zero, $t0
j continue

even: add $t3, $zero, $zero

continue: add $t1, $t1, $t3
j loop

done: sub $t3, $t3, $t3
```

**Part (c) (15 points)**
Calculate the total number of cycles the above code takes to execute on the 6-stage pipeline on the previous page, from the IF stage of the first instruction to the WB stage of the last instruction. *Explain*. We must understand your computation to give you points.

**Part (d) (5 points)**
Could the running time of the above code be improved by removing one of the pipeline registers to make the machine into a 5-stage pipeline? If so, indicate which one and *explain* why/how performance would improve. If not, *explain* why not.
Question 2: Cache Analysis (30 points)

For a 8KB 2-way set associative L1 cache with LRU replacement and with blocks of size 16 bytes on a machine with 24-bit address space (both virtual and physical) with no hardware prefetching, consider the following code snippet. Given that the cache is initially empty, the memory is byte addressable, and that data for multi-dimensional arrays is stored in row-major order, answer the following questions. A is stored starting at address 0x110fe0 and B is stored starting at address 0xa80fe0. Assume that the function \( \text{ceil} \) makes no memory accesses.

```c
#define N 1024
int A[N][N]; // integers are 4B
double B[N]; // doubles are 8B
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        A[i][j] += \text{ceil}(B[i]);
    }
}
```

**Part (a) (5 points)**

Compute the address trace for the first 6 memory accesses and also mark if each access is a HIT or a MISS.

<table>
<thead>
<tr>
<th>Address</th>
<th>HIT/MISS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

**Part (b) (10 points)**

Compute the total number of accesses and the total number of misses for the above code for the specified cache configuration. Explain.
Question 2: Cache Analysis

Part (c) (5 points)
If we added a 64KB 2-way set associative L2 cache with 32B blocks, what would be the number of accesses and misses for the L2 cache? Explain.

Part (d) (5 points)
For the L1 cache on the previous page, compute the number of misses if the code was changed to the following: Explain.

```c
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        A[j][i] += ceil(B[i]);
    }
}
```

Part (e) (5 points)
For the original code, compute the number of misses that would occur if the L1 cache was changed to be direct mapped with all the other parameters unchanged. Explain. Feel free to explain qualitatively (for partial credit) even if you can’t compute a result.
Question 3: Cache-aware Programming (30 points)

Part (a) (10 points)
What must be true about the original version of any piece of code for tiling to be a profitable transformation? Explain. (You might find it helpful to write a loop where tiling would be helpful on the scratch paper to answer this question.)

Part (b) (20 points)
Rewrite the following code to optimize its cache performance on a system with software prefetching and a single-level cache. The cache is a 2-way set-associative 32KB cache with 64B blocks. Ints and floats are 4 byte data types. Software prefetch syntax is shown on the right. Assume that the cache is initially empty and that the presents array starts at an address divisible by 4096. Explain the transformations you applied and why. Your score will depend both on how much you optimize and how well you explain.

```c
#define NUM_PRESENTS 12000
#define NUM_STORES 7
struct item {
    float best_price;
    float prices[NUM_STORES];
}
float total_cost = 0;
struct item presents[NUM_PRESENTS]; // initialized elsewhere

for (int i = 0; i < NUM_STORES; i++) {
    for (int j = 0; j < NUM_PRESENTS; j++) {
        presents[j].best_price =
            MIN(presents[j].prices[i], presents[j].best_price);
    }
}
for (int j = 0; j < NUM_PRESENTS; j++) {
    total_cost += presents[j].best_price;
}
return total_cost;
```

```
void __builtin_prefetch (const void *addr,
    unsigned rw, unsigned locality);
addr: the address of the memory to prefetch.
rw: (optional) a compile-time constant 1 or 0; 1: the program anticipates writing the data soon, 0 (default) in the near term, the program expects to only read the data.
locality: (optional) a compile-time constant from 0 to 3. 0: data has no temporal locality, so need not be left in the cache after the access, 3: (default) data has a high degree of temporal locality and should be retained in all levels of cache if possible. Use 0 or 3 based on the expected reuse.
```

(write your answer on the next page)
Question 3: Cache-aware Programming (cont.)