CS398 Exam 3
December 6th, 2012

Name: ____________________________________________
NETID: __________________________________________

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

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>AYA (1-3pm) Craig</td>
<td>AYE (9-11am) Maria</td>
</tr>
<tr>
<td>AYB (2-4pm) Jon</td>
<td>AYF (10am-noon) Ting</td>
</tr>
<tr>
<td>AYC (3-5pm) Michael</td>
<td>AYG (11am-1pm) Ting</td>
</tr>
<tr>
<td>AYD (4-6pm) Ting</td>
<td>AYH (noon-2pm) Jon</td>
</tr>
<tr>
<td></td>
<td>AYI (1-3pm) Ryan</td>
</tr>
<tr>
<td></td>
<td>AYJ (2-4pm) Michael</td>
</tr>
<tr>
<td></td>
<td>AYK (3-5pm) Michael</td>
</tr>
<tr>
<td></td>
<td>AYL (4-6pm) Ryan</td>
</tr>
<tr>
<td></td>
<td>AYM (5-7pm) Ryan</td>
</tr>
</tbody>
</table>

- This exam has 7 pages; the final sheet is provided as a reference to you.
- You have 120 minutes.
- No calculators or other electronics are allowed. You may bring one 8.5” x 11” sheet of handwritten notes.
- 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>25</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Multiple choice</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Question 1: Pipelining (40 points)

Consider the 5-stage pipeline that we discussed in class as shown below. Assume that all branches and jumps are predicted as not-taken. Conditional branches and indirect branches (e.g., \texttt{jr}) are resolved in the EX stage. Unconditional jumps are resolved in the ID stage.

For all of this question consider the MIPS assembly code on the following page. Corresponding C code is shown below.

**Part (a)** Annotate the MIPS assembly to indicate all of the true data dependences. (5 points)

For the next two parts, use your scantron form. We recommend that you first answer on the next page and then copy your answers to the scantron form. In any case, we won't give credit for any answers not on the scantron.

**Part (b)** Indicate which instructions will be stalled: a) no stall b) stall (5 points)

**Part (c)** Indicate how each of the forwarding muxes (\texttt{forwA}, \texttt{forwB}) will set in the cycles when each instruction is in the EX stage. Answer a-c as labelled in the diagram above. (10 point)

```c
typedef struct pixel{
    int x, y, z;
    int r, g, b;
} pixel_t;

void filter4(pixel_t **image1,
    pixel_t **image2, int i) {
    int x = image1[i-1]->x;
    int new_x;
    if (x < 0) {
        new_x = image1[i-1]->y;
    } else {
        new_x = image1[i-1]->z;
    }
    image2[i]->x = new_x;
}
```
Part (d) Compute how many cycles this function will take to execute starting from the cycle that the sll is fetched until the cycle the jr is completed. Assume the branch is not taken. Explain your answer for partial credit. (10 points)

Part (e) Re-schedule/re-write the function to make it faster. Faster code will achieve more points, but your answer must fit in the space below. (10 points)
Question 2: Cache Analysis (25 points)

Part (a) For an 8KB 2-way set-associative, write-back cache with 32B blocks on a machine with 32-bit address spaces (both virtual and physical) and no hardware prefetching, compute the average number of misses per inner-loop iteration for the following loop nest (and explain how you computed it!). Assume everything is in registers, except the matrix A. (20 points)

```c
const int N = 900;
float A[N][N];    // floats are 4 bytes big

for (int i = 0; i < N/2; i++) {
    for (int j = i+1; j < N; j++) {
        float temp1 = A[j][i];
        float temp2 = A[i][j];
        A[i][j] = temp1;
        A[j][i] = temp2;
    }
}
```

Part (b) Would your answer be different if N = 1024? Explain. (5 points)
Question 3: Cache-aware Programming (20 points)

Rewrite the following code to optimize its cache performance on a system with no hardware prefetching and a single-level cache. The cache is a 2-way set associative 16KB cache with 32B blocks. Software prefetch syntax, if you choose to use it, is shown on the right.

```
#define N 5000
float A[N][N], B[N], C[N][N], D[N];

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

for (int k = 0 ; k < N ; k ++) {
    D[k] = B[k]*B[k];
}
```
Multiple Choice (15 points)

22) You are downloading a large file over the internet and it is taking too long. Which should you be more concerned with?
   a) The latency of the packets.
   b) The throughput of the packets.

23) CPI (average Cycles Per Instruction) is always a number greater than or equal to 1, because an instruction can’t be executed in less than one cycle.  
   A) TRUE    B) FALSE

24) Pipelining a processor’s datapath can increase the latency of executing individual instructions compared to a non-pipelined implementation.  
   A) TRUE     B) FALSE

25) Which of the following is a valid stall control signal for the pipeline on page 2 of this exam:
   a. stall = (IF/ID.MemRead && ((IF/ID.rt == ID/EX.rs) || (IF/ID.rt == ID/EX.rt)));
   b. stall = (ID/EX.MemRead && ((IF/ID.rt == ID/EX.rs) || (IF/ID.rt == ID/EX.rt)));
   c. stall = (IF/ID.MemRead && ((ID/EX.rt == IF/ID.rs) || (ID/EX.rt == IF/ID.rt)));
   d. stall = (ID/EX.MemRead && ((ID/EX.rt == IF/ID.rs) || (ID/EX.rt == IF/ID.rt)));

26) Which of the following is false, or select (d) if all of the statements are true:
   a. a structural hazard occurs when a part of the processor's hardware is needed by two or more instructions at the same time.
   b. a data hazard occurs when two instructions that exhibit a data dependence are in pipeline stages that preclude the satisfaction of that dependence.
   c. a control hazard occurs when the instruction to execute after a control instruction is not known at the time that that instruction needs to be fetched.
   d. all of the statements are true.

27) Which of the following statements is false:
   a. Predicting branches as taken is more difficult than predicting them as not taken because it requires knowing the branch target.
   b. Predicting branches taken has the same prediction accuracy as predicting them not taken: 50%.
   c. Branch target buffers allow the processor to know that an instruction is a branch before it is fetched.
   d. A longer pipeline typically increases the branch misprediction penalty.
   e. Unconditional branches can be resolved earlier than conditional branches.

For a 32KB 4-way set-associative, write-back cache with 64B blocks on a machine with 32-bit address spaces (both virtual and physical): (for questions 28-30)

28) How many bits is the block offset?    a) 4     b) 5     c) 6     d) 7     e) 8
29) How many bits is the index?    a) 4     b) 5     c) 6     d) 7     e) 8

30) Can the data associated with the following addresses be in the cache simultaneously?    a) Yes    b) No

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hexadecimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>65536</td>
<td>0x10000</td>
<td>000100000000000000000000</td>
</tr>
<tr>
<td>131072</td>
<td>0x20000</td>
<td>001000000000000000000000</td>
</tr>
</tbody>
</table>
Multiple Choice, cont.

For the following pieces of code, indicate which forms of locality are definitely exhibited for all of the accesses to the bolded data structure:

a) temporal locality  b) spatial locality  c) both  d) neither

31) for (int i = 0 ; i < NUM ; i ++) {
    x += data_items[i];
}

32) for (list_t *list = list_begin ; list != NULL ; list = list->next) {
    sum += list->data;
}

33) for (int k = 0 ; k < 1024 ; k ++) {
    for (int l = 0 ; l < 1024 ; l ++) {
        rays[k][l] = chords[k][0] * scaling[l];
    }
}

34) Which of the following is false, or select (d) if all of the statements are true:
   a. virtual memory uses indirection to allow pages of a program’s memory to be migrated between physical memory and disk.
   b. virtual memory systems always use a write-back policy to minimize the number of I/O accesses performed.
   c. virtual memory provides security between programs by making sure that no two programs use the same virtual addresses.
   d. all of the statements are true.

35) In a virtual memory system, the “page offset” refers to:
   a. the number that has to be added to the page table base pointer when indexing into the page table.
   b. the number concatenated to the physical page number to compute the physical address.
   c. the number of bit positions that the virtual address has to be shifted to create the virtual page number.
   d. the number that has to be added to the virtual address to compute the physical address.

36) Which of the following is false, or select (e) if all of the statements are true:
   a. a program’s virtual addresses can be mapped to physical memory, disk, or nothing at all.
   b. multi-level page tables are used because they are more space efficient than monolithic page tables.
   c. translation look-aside buffers (TLBs) are caches that hold virtual address to physical address translations.
   d. virtual memory systems rely on high levels of spatial and temporal locality to be efficient.
   e. all of the statements are true.
Scratch paper (will not be graded).
Scratch paper (will not be graded).
Scratch paper (will not be graded).

<table>
<thead>
<tr>
<th>Instruction</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>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>