EECS 470 Final Exam
Fall 2005

Name: ___________________________  unique name: _____________

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

___________________________________

Scores:

<table>
<thead>
<tr>
<th>#</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 1</td>
<td>/30</td>
</tr>
<tr>
<td>Section 2</td>
<td>/30</td>
</tr>
<tr>
<td>Section 3</td>
<td>/40</td>
</tr>
<tr>
<td>Total</td>
<td>/100</td>
</tr>
</tbody>
</table>

NOTES:
1. Open book and Open notes
2. Calculators are allowed, but no PDAs, Portables, Cell phones, etc.
3. Don’t spend too much time on any one problem.
4. You have about 120 minutes for the exam.
5. Be sure to show work and explain what you’ve done when asked to do so.
Part I – Short answer/fill in the blank – 30 Points.

1. Consider your solution to project 3, but ignore the structural memory hazard (so we can load and store to memory while doing a fetch). A given program has an instruction mix of 20% loads, 10% stores, 20% branches, with the remainder being ALU operations. The branch predictor is correct 80% of the time. 50% of all ALU operations are followed by a dependent instruction. 25% of all loads are followed by a dependent instruction. What CPI is achieved on this program? Show your work. [5]

2. Given a virtually indexed, physically-tagged cache that is eight-way associative and has 64-byte blocks, what is the largest total size the cache could have if pages were 4KB in size? [3]

3. Consider an architecture which consists of 64 architected registers. Now say we have the following instruction formats:
   - Jump: Takes two register arguments.
   - Branch/load/store: Takes two register and one 12-bit immediate as arguments.
   - Register ALU: Takes 3 registers as arguments.
   - Immediate ALU: Takes 2 registers and a 16-bit immediate as arguments.

Which of the following combinations of instruction types could be encoded in a 32-bit instruction format? LIST all correct answers. [6 points, -2 per wrong/missing answer]
   a) 16 Jumps, 64 branch/load/store, 128 Register ALU, 12 immediate ALU
   b) 1024 Jumps, 32 branch/load/store, 64 Register ALU, 8 immediate ALU
   c) 16 of each
   d) 32 Jumps, 256 branch/load/store, 0 ALU (both types)
   e) 4 Jumps, 32 branch/load/store, 128 Register ALU, 4 immediate ALU
   f) 1023 Jumps, 4 branch/load/store, 4 Register ALU, 32 immediate ALU

Place your answer here
4. When using the MESI coherence protocol and bus protocol described in class, which of the following are true? Assume the cache is write-back and write-allocate. **LIST all correct answers.**  
[6 points, -2 per wrong/missing answer]  
- a) If a given cache line is held in the “E” state and that cache’s processor wishes to do a **write**, the transaction need not be sent on the bus.  
- b) If a given cache line is held in the “E” state and that cache’s processor wishes to do a **read**, the transaction need not be sent on the bus.  
- c) If a given cache line is held in the “S” state and that cache’s processor wishes to do a **write**, a BRIL transaction is placed on the bus.  
- d) The only state which indicates that the data is “dirty” is the “M” state.  
- e) When a cache-line held in the “S” state is evicted, a BWL transaction is placed on the bus.  

Place your answer here  

5. Provide the shortest possible reference stream where a direct-mapped cache will get a hit, while a 4-way associative cache will get a miss. Assume both are 1KB caches with 32-byte lines. [3]  

6. Fill-in-the-blank or circle the best answer. [7 points, -1 for each wrong or blank answer, minimum of zero]  
- a. As wires get narrower their resistance goes **Up or Down.** As wires get shorter their resistance goes **Up or Down.**  
- b. Adding **architected or physical** registers can provide the compiler more flexibility in avoiding ___________ dependencies.  
- c. If the multiplier is in the critical path for the clock cycle, decreasing the number of pipeline stages used to do a multiply can be expected to **increase or decrease or not effect** the clock period while **increasing or decreasing** the CPI.  
- d. If the BTB is in the critical path for the clock cycle, making it more associative can be expected to **increase or decrease or not effect** the clock period while **increasing or decreasing** the CPI.  
- e. A ___________ predictor actually consists of two (or more) predictors, where one is usually global and the other local. The predictor that chooses which predictor to use only is updated when the two underling predictors _________________.
Part II – Longer Answer – 30 Points.

1. Consider processor which, when running a given application performs 1 billion loads and 1 billion stores per second. Assume the following is true:
   
   - The processor’s multi-level cache system gets a 95% hit rate on both loads and stores.
   - Cache lines are 32-bytes in size
   - There is no prefetching and the instruction cache never misses.
   - 20% of all lines evicted from the last level of the cache are dirty.
   - The cache is write-back and write-allocate.
   - There are no coherence misses.
   - All loads and stores are to 4-byte values.
   - The bus only supports an operation size of 32-bytes.

   a) What is the read bandwidth (bytes/second) on the bus? Show your work. \[3\]

   b) What is the write bandwidth (bytes/second) on the bus? Show your work. \[3\]

   c) How would the above values change if the cache were non-write allocate (assume the hit rates remain constant)? \[4\]
2. For this problem, assume the base processor’s power consumption is 50% static and 50% dynamic.

   a) Say that Bob has proposed a processor improvement. This improvement takes a base processor and drops the total power consumption by 50% while increasing runtime by a factor of 1.2. What percent of the original processor’s energy is used by a processor using Bob’s improvement to perform a given task? Assume that his solution continues to have a 50/50 split between dynamic and static power consumption. [5]

   b) Say we instead use voltage scaling and reduce the frequency and voltage to 80% of its old value, and in so doing we increase the runtime by a factor of 1.2. What percent of the original processor’s energy would we expect this scheme to use when performing a given task? [5]
3. Consider the following C code:

```c
int SIZE, STRIDE;
int A[SIZE];

// Initialize SIZE and STRIDE here
for(j=0;j<N;j++)
    for(i=0;i<SIZE;i=i+STRIDE)
        X=A[i];
```

Assume N is a very large number. What hit rates would you expect to get on a 1KB, direct-mapped data cache with 32-byte lines given the following values for STRIDE and SIZE? You are to assume that every value other than the array A is kept in registers and that ints are 4 bytes. [10 -1.5 per wrong or empty box, min 0]

<table>
<thead>
<tr>
<th>SIZE</th>
<th>STRIDE=2</th>
<th>STRIDE=4</th>
<th>STRIDE=24</th>
<th>STRIDE=32</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Part III – Longest answer – 40 Points

1. In this problem we are using what we are calling Tomasulo’s third algorithm. Say the ROB, RS, RAT, PRF, and RRAT are as follows:

<table>
<thead>
<tr>
<th>RAT</th>
<th>ROB</th>
<th>RRAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RS</th>
<th>GR1</th>
<th>GR2</th>
<th>OP1 PRN/value</th>
<th>OP2 PRN/value</th>
<th>Dest. PRN</th>
<th>ROB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Key:
- **Op1 PRN/value** is the value of the first argument if “Op1 ready?” is yes; otherwise it is the Physical Register Number that is being waited upon.
- **Op2 PRN/value** is the same as above but for the second argument.
- **Dest. PRN** is the destination Physical Register Number.
- **ROB** is the associated ROB entry for this instruction.
- **Free/Valid** indicates if the PRF entry is currently available for allocation and if the valid in it is valid.

![Diagram of ROB and RRAT](image)

The following instructions have been issued.

```
top: R1=R2*R3            // A
      R3=R4+R5            // B
      if(R3==R4) goto top // C
      R5=R1+R2            // D
      R3=R5+R3            // E
```

A has committed, D has completed execution and the other instructions (B,C,E) have issued but have not yet started execution. A commits after C is issued, but before D is issued and the branch is predicted not-taken. Fill the different structures. The PRF will always allocate the lowest numbered free PRF entry. The PC of instruction A is 20 and each instruction is 4 bytes in size. [20]
2. The following code implements a single reservation station. There are five missing lines and a large missing block. Fill them in. You may not add or modify any wires/regs/inputs/outputs. [20]

```
`define SD #1

module rs1(rs1_dest_in, rs1_opa_in, rs1_opa_valid, rs1_opb_in, rs1_opb_valid,
rs1_cdb_in, rs1_cdb_tag, rs1_cdb_valid, rs1_load_in, rs1_avail_out,
rs1_ready_out, rs1_opa_out, rs1_opb_out, rs1_dest_tag_out, rs1_use_enable,
reset, clock);

input  [4:0] rs1_dest_in;    // The destination of this instruction
input [63:0] rs1_cdb_in;     // CDB bus from functional units
input  [4:0] rs1_cdb_tag;    // CDB tag bus from functional units
input        rs1_cdb_valid;  // The data on the CDB is valid
input [63:0] rs1_opa_in;     // Operand a from Rename
input [63:0] rs1_opb_in;     // Operand b from Rename
input        rs1_opa_valid;  // Is Opa a Tag or immediate data (READ THIS COMMENT)
input        rs1_opb_valid;  // Is Opb a tag or immediate data (READ THIS COMMENT)
input        rs1_load_in;    // Signal from rename to flop opa/b
input        rs1_use_enable; // Signal to send data to Func units AND to free this RS
input        reset;          // reset signal
input        clock;          // the clock

output        rs1_ready_out;     // This RS is in use and ready to go to EX
output [63:0] rs1_opa_out;       // This RS' opa
output [63:0] rs1_opb_out;       // This RS' opb
output [4:0] rs1_dest_tag_out;  // This RS' destination tag
output        rs1_avail_out;     // Is this RS is available to be issued to

wor    [63:0] rs1_opa_out;
wor    [63:0] rs1_opb_out;
wor     [4:0] rs1_dest_tag_out;

reg    [63:0] OPa;              // Operand A
reg    [63:0] OPb;              // Operand B
reg    OPaValid;               // Operand a Tag/Value
reg    OPbValid;               // Operand B Tag/Value
reg        InUse;              // InUse bit
reg     [4:0] DestTag;         // Destination Tag bit

wire          LoadAFromCDB;  // signal to load from the CDB
wire          LoadBFromCDB;  // signal to load from the CDB

assign rs1_avail_out = ______________________________________________
assign rs1_ready_out = ______________________________________________
assign rs1_opa_out = _______________________________________________
assign rs1_opb_out = _______________________________________________
assign rs1_dest_tag_out = ____________________________________________

// Has the tag we are waiting for shown up on the CDB
assign LoadAFromCDB = (rs1_cdb_tag[4:0] == OPa) && OPaValid && InUse && rs1_cdb_valid;
assign LoadBFromCDB = (rs1_cdb_tag[4:0] == OPb) && OPbValid && InUse && rs1_cdb_valid;
```

EECS 470 Fall ‘05 Final Page 8/9
always @(posedge clock)
begin
  if (reset)
    begin
      OPa <= `SD 0;
      OPb <= `SD 0;
      OPaValid <= `SD 0;
      OPbValid <= `SD 0;
      InUse <= `SD 1'b0;
      DestTag <= `SD 0;
    end
  else
    begin
      if (rs1_load_in)
        begin
          OPa <= `SD rs1_opa_in;
          OPb <= `SD rs1_opb_in;
          OPaValid <= `SD rs1_opa_valid;
          OPbValid <= `SD rs1_opb_valid;
          InUse <= `SD 1'b1;
          DestTag <= `SD rs1_dest_in;
        end
      else
        begin
          // else rs1_load_in
        end
    end // else rs1_load_in
end // else !reset
end // always @
endmodule