EECS 470 *Final Exam*

Name: ___________________________________

Scores:

Page #	Points
Section I	/20
Section II	======
1	/7
2	/10
3	/18
4	/10
5	/15
6	/8
7	/12
Total	**/100**

NOTES:

- Open book and Open notes
- Calculators are allowed, but no PDAs, Portables, Cell phones, etc.
- ***There are a lot of questions***. Don't spend too much time on any one problem. It might also be a good idea to look over the problems. Some may be much more time consuming than others.
- You have about 120 minutes for the exam.

Section I – Fill in the blank/short answer – 20 points

Each of the 10 questions is worth 2 points each. Use no more than 4 words for each blank. Either provide your answer in the blank or directly below the question. Some questions may have more than one right answer; pick the one you think is best.

1. WAR and WAW are types of ____________ dependencies that Tomasulo's algorithm removes by _______________.

2. In the second Tomasulo's algorithm a ROB is introduced in addition to the reservations stations. The reason we don't remove the RS is that the RS is _________ than the ROB, thus reducing the fan-out of the _________________.

3. In an n-way super-scalar processor, the number of checks required to determine if there exists any dependencies between any of the n instructions being decoded is on the order of ____________________ (in terms of n).

4. The LRU replacement algorithm in a cache is an attempt to take advantage of what characteristic of memory reference streams? Be precise but answer in no more than 4 words.

5. Loop unrolling will generally reduce the I-cache _______________ but usually increases the amount of ____________ available in the program.

6. IA-64 uses stop-bits to indicate ____________________________________.

7. A ______________________ data cache doesn't need dirty bits.

8. The reason store buffers are used is to reduce the amount of time it takes a ___________ to get to main memory.

9. Victim caches are most effective when added to ___________________ caches.

10. In a single-processor computer, there are three categories often used to describe cache misses: Capacity, Conflict, and Compulsory. In a multi-processor system a forth type of cache miss, called _________________ can also occur.

Section II – Longer questions – 80 points

In this section you should be sure to show your work. *Each page has a single question usually in multiple parts.*

1. AHI is a computer company developing a new ISA from scratch. In their new ISA they have *all* ALU and branch instructions be of the format “Op-code, register, register, immediate”
    - All ALU operations are of the form reg1=reg1⊕ reg2 ⊕ immediate where reg1 and reg2 can be any of the GPRs.
    - All branches are of the form if(reg1 ⊕ reg2) branch to PC + immediate

   (⊕ is used to indicate an arbitrary operation such as addition or greater than.)

   If the ALU and branch instructions take up three-fourths of the instruction encodings, and there are 32 GPRs, and there are 48 such instructions, and each instruction is 32 bits in size, how big can the immediate be assuming all ALU and branch instructions use the same size immediate? [7]

2. Consider a virtually addressed, 32KB, 2-way associative, write-back data cache on a computer where the page size is 4 KB. The machine is byte-addressed, the physical addresses are 32 bits and the virtual addresses are 36 bits. The TLB holds 16 entries and is fully-associative.
   a) How many bits are there in the tag of the TLB? [3]

   b) How many bits are there in the tag of the cache? [3]

   c) Due to *aliasing* it may be necessary to search more than one set for a given block. How many *sets* will need to be searched? [4]

3. Say you have an out-of-order processor (using the second of Tomasulo's algorithms found in the text) which has a total of 16 reservation stations (usable by any execution unit) a ROB of size 48, 32 architected registers, 2 CDBs, and the ability to fetch, issue and commit 2 instructions per cycle (in an ideal case the processor can sustain 2 instructions committing per cycle.)

   a) If a high-latency load were to occur, what is the largest number of instructions behind that instruction (in program order) which could *commit* before the load finished executing? [4]

   b) If a high-latency load were to occur, what is the largest number of instructions behind that instruction (in program order) which could *execute* before the load finished executing? [4]

   c) If a high-latency load were to occur, what is the largest number of instructions that could *execute* in the time between when the load starts execution and when the load finishes execution? [4]

   d) What is the *best achievable* CPI for this processor if:
      - A given program executes 200,000 instructions, 200 of which are high-latency loads. ***These loads are evenly spread throughout the program.***
      - Each high-latency loads requires 100 cycles to complete

      For this problem you must correctly explain your answer to receive *any* credit. Answers which do not have a clear and precise justification will not be accepted. [6]

4. Consider a bus-based multi-processor system using the MESI protocol. As in class, each processor can issue the following bus transactions: BRL, BRIL, BIL, BWL. Assume the block size is 64 bytes, the word size is 4 bytes, and the addresses are all 32-bits in size.

   a) If a processor currently has a given address in its cache in the shared state and that processor performs a store to that address, what bus instruction will be sent from that processor? What state will that address be in that processor once the bus transaction has finished? State any assumptions. [3]

   b) If a processor currently doesn't have a given address in its cache and that processor performs a load of that address, what bus instruction will be sent from that processor? What state will that address be in that processor once the bus transaction has finished? State any assumptions. [3]

   c) Say that the current bus traffic is evenly split between the four bus instructions. List all types of bus transactions that would increase in frequency if the caches were made smaller. Provide a brief explanation. [4]

5. Consider the following C code:

```
for ( i = 0; i < MAX; i++ ) {
            a[ i ] = a[ i ] + b[ i ];
      } //end for
```

That C code is translated into the following x86-like assembly language:
(note: the ++ indicates the "autoincrement" addressing mode. See page 98 if needed.)

```
      mov   r1, addr( a )  -- address of a[ 0 ] into r1
      mov   r2, addr( b )  -- address of b[ 0 ] into r2
      mov   rx, MAX        -- Number of iterations into rx
l1:
      ld    r3, (r1)       -- load indirect into r3 through r1
      ld    r4, (r2)++     -- what r2 points to loaded in r4
      fadd  r5, r3, r4     -- r5 holds sum of two elements
      st    r5, (r1)++     -- store result and post-increment
      loop  l1             -- does an autodecrement (by 1) of rx
                           -- if rx isn't zero branches to l1
```

And then that assembly code is software pipelined.

```
-- Initialization:
      mov   r0, addr( a )     -- r0 is pointer to a[0]
      mov   r1, r0            -- copy address of a[0] into r1
      mov   r2, addr( b )     -- r2 is pointer to b[0]
      ___blank A_______
      ___blank B_______
      ___blank C_______
      fadd  r5, r3, r4
      ld    r3, (r1)++
      ld    r4, (r2)++
l2:   st    r5, (r0)++
      fadd  r5, r3, r4
      ld    r3, (r1)++
      ld    r4, (r2)++
      loop  l2                -- decrement rx, if != 0 jump to l2
      ___blank D_______
      fadd  r5, r3, r4
      st    r5, (r0)
```

Supply the missing code [3 each]

blank A: ______________________ blank B: ______________________

blank C: ______________________ blank D: ______________________

If, in the original C code, MAX is ***less than*** ________ the software-pipelined loop will behave incorrectly. [3]

(The above code was taken from an example by Herb Mayer at Portland State University)

6. You are working on a project in Verilog that isn't quite working and you think you have narrowed it down to the following mux. The code compiles and most of the time it seems to work.

```
always @(mult_done or ld_result or alu_result or mult_result)
 if(mult_done)
   top_result = mult_result;
 else if(ld_done)
   top_result = ld_result;
 else top_result = alu_result;
```

   a) Either provide correct code to do the same thing or describe the edit required to fix it. [5]

   b) Provide one technique to avoid or detect this type of error. [3]

7. Say we have a deeply pipelined in-order, two-way super-scalar processor with a total of 14 stages that runs at 1GHz. Branches are predicted not-taken and resolved in the 10th stage as are *all* other instructions, and no operation needs its arguments before the 10th stage. Further:

    - Memory accesses have a hit rate of 90%, a T_{miss} of 10 cycles and a T_{hit} of 0 cycles (so no more time than normally required by the pipeline).
    - Stores are placed into a store buffer that, on our benchmark, never fills up.
    - In this benchmark, 20% of all instructions are loads, 10% are stores, 10% are branches, and the rest are ALU instructions.
    - Branches are taken 60% of the time.
    - If an instruction stalls, all instructions after it in program order also stall.

    a) If the *only* source of stalling (or otherwise having to have less than a CPI of 0.5) were branch mis-predictions, what would be the CPI of the benchmark running on this processor? [3]

    b) If the *only* source of stalling (or otherwise having to have less than a CPI of 0.5) were due to load stalls, what would be the CPI of the benchmark running on the processor? [3]

    c) If no branches were ever taken, what would be the actual CPI of the benchmark running on the processor? Assume:

        - 50% of all instructions are dependent on the instruction in front of them.
        - There are two pipes. Instructions are initially placed into both pipes without regard to dependencies.
        - In stage 6 dependencies are checked. Data hazards are resolved by stalling the dependent instruction and all instructions which follow the dependent instruction.

        Show and explain your work carefully. State any assumptions you feel you need to make. [6]