Name:

Turn in this exam with your answers, or your score will be zero (0).

CMSC 411 Computer Architecture
Final Exam A
Summer, 1998

Remember, it is supposed to be HARD. Don't panic when you get stuck. Try something else and come back to it. If logic fails, try to think like an "intel chip". After that, thinking like an "ignorant user" will be a relief.

Problem 1: Matchless Moments (30)

For each numbered item, give the **Letter** of the item that is most closely associated with the numbered item. Note that there are more **Letters** than numbers, but letters should not be used more than once. Note that letter **(q. Other)** can be used **exactly once**, and requires an explanation.

1.1	Tomasulo's algorithm	a. scoreboard
1.2	Many MFLOPS	b. compiler optimization
1.3	Superpipelined	c. virtual memory
1.4	SRAM (2ns-10ns)	d. JAL POWER
1.5	CDB	e. scoreboard issue stalls
1.6	DEC Vax	f. variable instruction lengths
1.7	register-memory architecture	g. I-type instruction
1.8	WAW hazard	h. Branch Table Buffer
1.9	LD F0, 64(R0)	i. ILP
1.10	Alpha AXP memory hierarchy	j. structural hazard
1.11	Dynamic scheduling	k. small static code size
1.12	TLB	l. L1 Cache
1.13	BTB	l. branch delay slot
1.14	unconditional branch	m. multiple instruction issue
1.15	VLIW instruction	n. ITLB
		o. (1,3) ISA
		p. Cray X-MP
		q. Other (Please Specify)

Problem 2: Cache and Carry (24)

2.1 (6) As you may recall from your reading, the average memory access time is the sum of the **HIT TIME** and the product of the **MISS RATE** and the **MISS PENALTY**. Pick two (2) of the cache organizational parameters, written in **All CAPS** above, briefly define them, and describe a technique that can be used to improve the average memory access time by altering the value of the specific parameter.

2.2 (6) Suppose that Loads and Stores take 2 clock cycles each, and account for 40% of the instructions. Branches account for 15% of the instructions and take 3 clock cycles each. All other instructions take 1 clock cycle. Assume that the cache has a miss penalty of 50 clock cycles, with a 5% miss rate for instructions and a 15% miss rate for data references.

Determine how much slower the machine is with cache misses than it is if a perfect cache is assumed. Make sure you set up the problem clearly–I'm looking for how you do it, not what your numbers are!!!!

2.3 (6) What effect, if any, would you expect the addition of a multi-level cache to have upon a Multi-Cycle DLX pipeline's performance? Why?

2.4 (6) What effect, if any, would you expect the addition of a multi-level cache to have upon a Scoreboard-based DLX implementation? Why?

Problem 3: DLX-LAX: An ISA for the Millenium (30)

3.1 (10) Compare the implementation of the DLX ISA on three HW platforms: the standard Multicycle DLX pipeline (with full forwarding and bypassing, split memory, and WB writing in the first half of the clock cycle and ID reading in the second); the Scoreboard algorithm; and, Tomasulo's algorithm. For grading convenience, organize your answer in a chart. Compare the way data hazards are treated on each platform. Identify all potential structural hazards for each platform. For the Scoreboard and Tomasulo's algorithms, indicate benefits and drawbacks over the Multicycle DLX pipeline.

3.2 (20) Suppose you decide to change the DLX ISA to permit a single memory source operand in any ALU instruction. To accommodate this change, memory references are restricted to register indirect. That is, no offsets are allowed. To make life a bit easier, we introduce the instruction Load Immediate (LI), which takes a 16 bit signed two's complement integer as a source operand and places it in the integer destination register.

The code sequence below illustrates the required addressing mode change, the new ALU instruction format, and the new load instruction

```
       LI R1, #2000
       LI R2, #4000
       LI R3, #4000
       LI R4, #4000
Loop:  LD F0,(R1)
       MULTD F6, F0, (R2)
       ADDD F4, F0, (R2)
       ADD R4, R1, R3
       SD (R4), F6
       ADDI R1, R1, #8
       ADDI R2, R2, #8
       SUBI R5, R3, R4
       BNEZ R5, Loop
```

3.2.1 (10) Originally, 20% of the instructions were loads, 18% were stores, 12% were branches, and 30% were INT ALU instructions.

After modification to the ISA and the pipeline structure, 25% of the loads are converted to the "Load Immediate" mode, and half of the INT ALU instructions are discarded.

Load Immediate instructions take 1 clock cycle. Branches are increased from 3 clock cycles before the change to 2 clock cycles afterwards. Assume that the remaining instruction times remain unchanged: the average FP ALU operation takes 8 clock cycles, INT ALU instructions take 1 clock cycle, and all other instructions take 2 clock cycles.

What is the improvement in execution time, if any, expected from the new pipeline?

3.2.2 (10) How would you modify the MultiCycle DLX pipeline to take advantage of this new addressing mode? Identify the new sequence of stages, and show the execution of one iteration of this code sequence assuming your new stages. Indicate any forwarding or stalls needed to conteract potential hazards.

Problem 4: OOPS—Branch Delayed (24)

4.1 (6) Consider the following 8-stage variant of the basic 5-stage DLX pipeline, the mythomagical *OctoPipe Processing System*, or OOPS for short. Separate instruction and data memories are used. However, no forwarding occurs and the writing/reading of registers is **NOT** split across a single clock cycle.

F1	Start Instruction Fetch
F2	Complete Instruction Fetch
ID	Decode instruction and fetch register values
EX1	Start Execution; branch condition tested, branch target computed
EX2	Execution completes; effective address or ALU result ready
MEM1/ALUWB	First half of memory cycle plus WB of ALU instructions
MEM2	Memory access completes
LWB	Load write back completes

Suppose that you decide to use static branch prediction for the pipeline implementation, and must choose whether to predict taken or not taken. Assuming that you can make no changes to the OOPS pipeline, which option would you choose. Justify your answer for full credit.

4.2 (6) Suppose the OOPS machine is to be re-implemented using dynamic branch prediction and floating point operations. As with the Multicycle DLX Pipeline, FP instructions do not use the EX1 and EX2 stages, but, instead have separate paths between the ID and M1 stages. The FP ADDD command has stages A1 through A6 instead of the EX1 and EX2 stages.

Consider the following code sequence, where register R2 originally holds 80 (that's base 10, folks), and F6 is pre-loaded with a double precision constant.

```
Loop:  LD F4,4000(R2)
       ADDD F4, F4, F6
       SD 4000(R2),F4
       SUBI R2, R2, #8
       BNEZ R2, Loop
```

Explain how a one-bit dynamic branch predictor would behave differently from a two bit dynamic branch predictor on this code fragment? Why? Make sure you consider all possible cases!!!

4.3 (6) Suppose we use the same pipeline described in part [3.2], but we abandon dynamic branch prediction and use branch delay slots instead.

Pretend you are the compiler. Unroll the code sequence from part [3.2] as many times as possible, and reschedule the instructions to minimize the CPI. Make sure you explain how you are dealing with the branch delay. DO NOT compute the CPI. No credit will be given for doing so.

4.4 (6) Suppose that a BTB was used instead of the branch prediction techniques discussed in parts [3.1]–[3.3]. Specifically, explain one benefit and one drawback to BTB-based branch prediction.