This exam consists of five questions, and is worth a total of 30 points. Take your time and make sure to show all work. Answers without justification will not receive full credit. Good luck!

"The Hierarchical Detective" [7 points]

1. For one uniprocessor system, the time to access main memory is 200 nanoseconds. Assume that there is no contention in the memory hierarchy. Measured from the time that it begins execution, a single load instruction can take many different times. The following measurements are sorted from fastest to slowest. For each of the following timings, give the most likely explanation of why the load took that long. Assume any caches are write through.

a) 10 nanoseconds

b) 50 nanoseconds

c) 200 nanoseconds

d) 400 nanoseconds

e) 20 milliseconds (20 x 10^6 nanoseconds)

f) Give 3 details about the organization and implementation of the uniprocessor system that you can deduce from these measurements.
"Branching Out" [4 points]

2. The classic 5-stage pipeline presented in Chapter 3 has a 1 clock cycle branch delay provided the branch condition is checked in the second stage and the branch address is calculated in the second stage. This one-cycle delay is part of the DLX architecture. More recent implementations have gone to longer pipelines, such as the 8-stage pipeline of the R4000 presented in Chapter 3. R4000 instruction fetch takes 2 stages and data fetch takes 3 stages. The R4000 checks the branch condition and calculates the branch address in the fourth stage (EX).

a) Suppose the machine is going to use static branch prediction for the 8-stage pipeline implementation. You choice is to predict taken or not taken. Assuming that you cannot change the pipeline, which would you chose? Why?

b) For the next part of the problem, we will focus on dynamic branch prediction.

*Here is the code in DLX assembly language:*

```assembly
LI R2 #80 ; R2 = 80ten
foo:  
LD F2 X(R2) ; load X(i)
MULTD F4 F2 F0 ; multiply a*X(i)
SD Y(R2) F6 ; store Y(i)
ADDI R2 R2 -#8 ; decrement index
BEQZ R2 foo ; loop if not done
```

For this code, how would a one-bit dynamic branch predictor perform differently from a two-bit dynamic branch predictor? Why?
"Caching out" [8 points]

3. Given a system with the following:

a standard RISC instruction set (e.g., DLX)
64-bit, 250 MHz uniprocessor CPU
48-bit virtual address
45-bit physical address
8 KB, virtually-indexed, direct-mapped, L1 Instruction Cache with 32B blocks
8 KB, virtually-indexed, direct-mapped, write-through, no-write-allocate, L1 Data Cache with 32B blocks
128 KB, physically-indexed, 4-way set associative, write-back, write-allocate, L2 Unified cache with 64B blocks
Inclusion is always maintained between the Data Cache and the L2 Cache (this means that the Data Cache is always a subset of the L2 Cache).
On average, 20% of the blocks in the L2 cache at any time are dirty
Time for to transfer a full cache block between L2 and main memory is 200 ns in either direction.

CPI execution (with no memory stalls) is 1.2
Hits to L1 take 1 CPU clock cycle
Hits to L2 take 3 CPU clock cycles
(Assume that you can't start an L2 access until you decide whether or not L1 is a hit. Therefore, an access that misses in L1 but hits in L2 takes a total of 4 CPU clock cycles.)
15% of instructions are loads; 10% of instructions are stores.
L1 Instruction Cache miss rate = 8%; L1 Data Cache miss rate = 15%; L2 local miss rate = 20%

a) How many total cache blocks are there in the L1 Instruction Cache?
   How many instructions does can be in the L1 Instruction Cache?

b) How much memory is needed beyond the 128 KB of data (e.g, tags) for the L2 cache?
   What percent is this overhead of 128 KB?

c) What is the average memory access time in clock cycles? Show the formula first.

d) What is the **global** miss rate of the L2 cache?
**CDC 6600 vs. IBM 360/91 [6 points]**

4. This question involves the difference in performance between the a scoreboard based machine and a Tomasulo algorithm based machine. We assume the same latencies as in the book.

<table>
<thead>
<tr>
<th>Instruction producing result</th>
<th>Instruction using result</th>
<th>Latency in clock cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP ALU op</td>
<td>FP ALU op</td>
<td>3</td>
</tr>
<tr>
<td>FP ALU op</td>
<td>Store Double</td>
<td>2</td>
</tr>
<tr>
<td>Load Double</td>
<td>FP ALU op</td>
<td>1</td>
</tr>
<tr>
<td>Load Double</td>
<td>Store Double</td>
<td>0</td>
</tr>
</tbody>
</table>

We will concentrate on this code fragment:

```plaintext
foo:  LD  F2  X(R2) ; load X(i)
      MULTD F4  F2  F0 ; multiply a*X(i)
      SD  Y(R2) F6 ; store Y(i)
      ADDI R2  R2  -#8 ; decrement index
      BEQZ R2 foo ; loop if not done
```

a) Which dynamic pipeline control scheme would yield faster execution? Why?

b) Write a software pipelined version of the loop above.

c) List two advantages of software pipelining over loop unrolling.
“Cray memorial” [10 points]

5. This question involves the difference in using mask registers and using scatter-gather. Consider the following code fragment:

```
   do 10 I = 1, 64
       if (B(i) .ne. 0) then
           A(i) = A(i) / B(i)
       continue
```

Assume that the addresses of A and B are in Ra and Rb. Using the vector mask capability, the corresponding DLXV code is

```
vmloop:  LD  F0  #0.0 ;F0 = 0.0
         LV  V0  Ra ;V0 = A vector
         LV  V1  Rb ;V1 = B vector
         SNESV F0  V1 ;VM(i) = 1 iff V1(i) != F0
         DIVV V0  V0,V1 ;V0(i) = V0(i)/V1(i) iff VM(i) = 1
         SV  V0  Ra ;A(i) = V0(i) iff VM(i) = 1
         CVM ;Clear VM to 1’s
```

Using the scatter-gather capability, the corresponding DLXV code is

```
sgloop:  LD  F0  #0.0 ;F0 = 0.0
         LV  V0  Rb ;V0 = B vector
         SNESV F0  V0 ;VM(i) = 1 iff V0(i) != F0
         CVI  V1  #8 ;V1 = indices to use
         POP  R1  VM ;R1 = # of 1’s in VM
         MOVI2S VLR R1 ;vector length = R1
         CVM ;Clear VM to 1’s
         LVI  V3  (Rb+V1);V3(i) = B(j) iff B(j) != 0.0
         LVI  V2  (Ra+V1);V2(i) = B(j) iff A(j) != 0.0
         DIVV V2  V2,V3 ;V2 = V2 / V3
         SVI  V2  (Ra+V1);A(i) = V2(j) iff B(i) != 0.0
         LI  R1  #64 ;vector length = 64
         MOVI2S VLR R1 ;
```

Assumptions about the machine and the program:
1) The machine has one memory pipeline and does not support chaining.
2) The LVI and SVI instructions have the same performance parameters (e.g., startup overhead) as LV and SV.
3) The vector mask register is an implied source in all relevant vector instructions; that is, there is a RAW data hazard between an instruction that changes VM and the following vector instructions.
4) The SNESV and CVI instructions have a startup overhead of one clock cycle and produce one result per cycle thereafter.
5) Instructions that do not operate on a vector register (e.g., CVM, MOVI2S) can be ignored when calculating T_{chime} or T_{start}.
6) The startup overheads are 6 clock cycles for vector add, 7 for vector multiply, 20 for vector divide, and 12 for vector load or vector store.
7) Assume that 50% of the entries of B are 0.
8) Assume that T_{loop} is 15.
9) The maximum vector length is 64.
10) As in the Appendix B, the functional units are FP add/subtract, FP multiply, FP divide, Integer, and Logical.
11) The initiation rate is one result per clock cycle.
12) For this code fragment, assume that the vector length is 64.
Name:__________________

Conditional vector, continued

a) Identify the convoys in the vector-mask code sequence. How many are there?

```
vmloop: LD F0 #0.0 ; F0 = 0.0
LV V0 Ra ; V0 = A vector
LV V1 Rb ; V1 = B vector
SNESV F0 V1 ; VM(i) = 1 iff V1(i) != F0
DIVV V0 V0, V1 ; V0(i) = V0(i)/V1(i) iff VM(i) = 1
SV V0 Ra ; A(i) = V0(i) iff VM(i) = 1
CVM ; Clear VM to 1's
```

b) Identify the convoys in the scatter-gather code sequence. How many are there?

```
sgloop: LD F0 #0.0 ; F0 = 0.0
LV V0 Rb ; V0 = B vector
SNESV F0 V0 ; VM(i) = 1 iff V0(i) != F0
CVI V1 #8 ; V1 = indices to use
POP R1 VM ; R1 = # of 1’s in VM
MOVI2S VLR R1 ; vector length = R1
CVM ; Clear VM to 1’s
LVI V3 (Rb+V1) ; V3(i) = B(j) iff B(j) != 0.0
LVI V2 (Ra+V1) ; V2(i) = B(j) iff A(j) != 0.0
DIVV V2 V2, V3 ; V2 = V2 / V3
SVI V2 (Ra+V1) ; A(i) = V2(j) iff B(i) != 0.0
LI R1 #64 ; vector length = 64
MOVI2S VLR R1 ;
```

c) What is T_start for the vector-mask code sequence?

d) What is T_start for the scatter-gather code sequence?
Conditional vector, continued

e) What is $T_{64}$ for the vector-mask code sequence? Be sure to state any additional assumptions.

f) What is $T_{64}$ for the scatter-gather code sequence? Be sure to state any additional assumptions.

g) For what percentage of zeros in B would $T_{64}$ be about the same for both code sequences?