CMSC 411 Practice Exam 2

General instructions.
Be complete, yet concise. You may leave arithmetic expressions in any form that a calculator could evaluate.

1. Cache

   (a) Suppose we have a byte addressable memory of size 4GB(2^{32} bytes) with a cache of size 256KB (2^{18} bytes), not including tag bits. Also suppose the cache block size is 32 bytes. For each of the following cache organizations, compute the length in number of bits for the tag, index and offset fields of the 32-bit memory address (show your calculations):

      i. direct mapped
      ii. 2-way set associative
      iii. 8-way set associative
      iv. fully associative

   (b) Given the following parameters, which performs better in terms of average memory access time, the direct mapped or the 2-way set associative cache? Assume a cache hit takes 1 cycle for either organization, and the cache miss penalty is 20 nanoseconds (both use the same memory system). Justify your answer.

   Direct mapped - miss rate 10%, 1GHz clock
   2-way set associative - miss rate 5%, 800MHz clock

   (c) Consider adding a fast second-level cache to a computer with only a single level cache. Suppose data can be accessed from the second level cache 12 times faster than it can be accessed from the original cache, and that the second level cache can be used 30% of the time. How much speedup can we gain by adding the second-level cache? Assume all data accesses hit in the original cache.

2. Branch prediction
   Consider the following code fragment:

```
if( d == 1 )
  d = 2;
if( d == 2 )
  d = 3;
if( d == 3)
  d = 4;
```

   A typical code sequence generated for this fragment, assuming that d is assigned to R1, looks like the following:

```
        DSUBUI R2, R1, #1
        BNEZ  R2, L1   ; Branch B1
        DADDIU  R1, R0, #2
L1:     DSUBUI  R3, R1, #2
        BNEZ  R3, L2 ; Branch B2
        DADDIU  R1, R0, #3
L2:     DSUBUI R4, R1, #3
        BNEZ R4, L3 ; Branch B3
        DADDIU R1, R0, #4
        . . .
L3:
```

   For the given program fragment, construct a (1, 1) predictor table, given that the code executes inside a loop, with the value of d alternating between 2 and 0. Fill in the table as given below. Circle the prediction bit used for each branch executed. Assume that all predictors are initialized to not taken, and that the correlation bit is initially set to not taken.

   Finally, calculate the total number of mispredicted branches.

d=?	B1 prediction	B1 action	New B1 prediction	B2 prediction	B2 action	New B2 prediction	B3 prediction	B3 action	New B3 prediction
2									
0									
2									
0									

3. Dynamic scheduling (25 points)
The following code implements the vector operation $Y = aX + Y$ for a vector of length 100, and assumes that initially R1=0 and F0 contains the constant a:

```
foo:  L.D      F2,0(R1)     ;load X(i)
      MUL.D    F4,F2,F0     ;multiply a*X(i)
      L.D      F6,0(R2)     ;load Y(i)
      ADD.D    F6,F4,F6     ;add a*X(i)+Y(i)
      S.D      F6,0(R2)     ;store Y(i)
      DADDUI   R1,R1,#8     ;increment X index
      DADDUI   R2,R2,#8     ;increment Y index
      DSGTUI   R3,R1,#800   ;test if done
      BEQZ     R3,foo       ;loop if not done
```

The DSGTUI instruction is an integer instruction that sets the result register, R3, to a non-zero value if the value in R1 is greater than 800.

You are going to describe the execution of the loop for a single-issue Tomasulo-style MIPS pipeline, with functional units as described in the following table:

FU type	Cycles in EX	# of FUs	# of reservation stations
Integer	1	1	5
FP adder	4	1	3
FP multiplier	15	1	2

Also assume the following:

- Functional units are not pipelined.
- No forwarding between functional units; results can only come from the CDB.
- The EX stage does both the effective address calculation and the memory access for loads and stores. So the pipeline is IF/ID/IS/EX/WB.
- Loads take 1 clock cycle.
- The issue(IS) and write result (WB) stages each take 1 clock cycle.
- There are 5 load buffer slots and 5 store buffer slots.
- The BEQZ instruction takes 0 clock cycles.

Show the number of stall cycles for each instruction and what clock cycle each instruction begins execution (i.e., enters its first EX cycle), for two iterations of the loop. Also show when each instruction writes the CDB (store and branch instructions don't write the CDB). Use the following table as a template:

Iteration number	Instructions	Issues at cycle #	Executes at cycle	Write CDB at	Comment (stalls)

How many clock cycles does each iteration take?