EE 4720 **Homework 5** **Due: 17 November 2000**

Problem 1: The familiar loop below executes on a dynamically scheduled machine using a reorder buffer to name destination registers. The machine has the following characteristics:

- Two-way superscalar. An unlimited number of write-backs per cycle.
- A 16-entry reorder buffer.
- A six-stage fully pipelined floating point multiply unit.
- Perfect branch target prediction. (Branch target in IF when branch is in ID.)

Show a pipeline execution diagram up to the fetch of the third iteration.

Explain why the first two iterations cannot be used to determine the CPI for a large number of iterations in this case. Estimate the CPI for a large number of iterations (a pipeline execution diagram is not necessary).

```
LOOP: ! LOOP = 0x1000
 ld   f0, 0(r1)
 muld f2, f0, f2
 addi r1, r1, #8
 sub  r2, r1, r3
 bneq r2, LOOP
 xor  r10, r11, r12
 and  r13, r14, r15
 or   r16, r17, r18
 sgt  r19, r20, r21
```

Problem 2: Unroll the loop in the problem above twice. (In the last homework it was unrolled four times.) Again exploiting the associativity of multiplication, rearrange the multiplies to improve the performance, but this time without using software pipelining. Why is software pipelining not necessary here?

Problem 3: The code below executes on a system using a one-level branch predictor with a 16-entry BHT. Which entries will the branches use?

If the number of iterations is large, the prediction accuracy will be high. If a certain number of additional nops are inserted before `SKIP1` the prediction accuracy will drop. How many and why?

```
 ! Note: r2 is not modified inside the loop.
LOOP:  ! LOOP = 0x1000
 subi r1, r1, #1
 bneq r2, SKIP1
 add r10, r10, r11
 nop
SKIP1:
 beqz r2, SKIP2
 add r12, r12, r13
SKIP2
 bneq r1, LOOP
```

Problem 4: Determine the prediction accuracy of a one-level branch predictor on each branch in the code below. The predictor uses a 1024-entry BHT. There is a .5 probability that a loaded value will be zero.

```
LOOP:
 addi r2, r2, #4
 lw r1, 0(r2)
 bneq r1, SKIP1
 add  r10, r10, r11
SKIP1:
 andi r3, r2, #4
 bneq r3, SKIP2
 add  r11, r11, r12
SKIP2:
 beqz r1, SKIP3
 add  r12, r12, r11
SKIP3:
 andi r4, r2, #12
 bneq r4, SKIP4
 add  r13, r13, r11
SKIP4:
 sub r5, r2, r6
 bneq r5, LOOP
```

Problem 5: How many BHT entries will the branches in the code above use in the middle of its execution (explained below) in a two-level gselect predictor that uses 10 bits of global branch history and 6 instruction address bits? The loop iterates many times, the middle of its execution starts after many iterations.

How many bits of global branch history are needed so that the branch following `SKIP3` is predicted very accurately?