EE 4720 **Homework 6** **Due: 4 May 1998**

Problem 1: The program below executes on a single-issue (not superscalar) DLX implementation that uses Tomasulo's algorithm for dynamic execution. It also includes dynamic branch prediction using an ID-stage BHT (but no target prediction) and a reorder buffer to support speculative execution.

Instruction execution proceeds speculatively using a predicted path when a branch condition cannot be immediately determined. When the condition is determined the instructions following the branch are cancelled if the prediction was wrong (otherwise execution proceeds normally). Branch instructions use a special branch functional unit, including reservation stations, located in the ID stage. If a branch condition is available when a branch instruction is in the ID stage (the register value is in the register file or can be bypassed to the ID stage that cycle) the branch is executed normally. Otherwise it waits in the branch functional unit until the register value it needs is ready while following instructions execute. Those following instructions may start at the branch target (as soon as available) or the fall through (the instruction after the branch), depending on how the branch is predicted. When the branch outcome is determined it is compared to the prediction, instructions following the branch are cancelled if the prediction was wrong, otherwise execution proceeds normally.

Assume that the reorder buffer has an unlimited capacity. At most one entry per cycle can be retired from the reorder buffer, but any number of elements can be deleted in one cycle. The system has five reservation stations per functional unit, including the integer functional unit, EX, and the branch functional unit, BR.

Show a pipeline execution diagram for the code below when the branch is mispredicted as taken. (That is, the outcome of the branch is not-taken, but it is predicted taken.) Show the contents of the re-order buffer at each cycle, include only the instructions shown below. Show execution until the reorder buffer is empty of the instructions encountered in the execution of the code below. For each entry in the reorder buffer show the instruction mnemonic and place a check next to it if it has completed execution.

The `eqf` instruction uses the floating-point add functional unit and bfpt uses the branch functional unit described above. Note that there is a dependency between `bfpt` and `eqf`.

```
multf f4, f5, f6
addf  f0, f1, f2
eqf   f0, f3     ! Set floating point condition code to true iff f0=f3
bfpt  TARG       ! Branch if floating point condition true.
add   r1, r2, r3
sub   r4, r5, r6
...

TARG:
 and  r1, r2, r3
 or   r4, r5, r6
 ...
```

More problems on next page.

Problem 2: The program below runs on a system using a 3-bit branch history branch predictor, with a branch predicted taken if the count is 5 or greater. Initially all entries in the branch history table are zero. What will the prediction accuracy be during the execution of this program? (The program never finishes, for simplicity consider execution until r1 reaches $2^{31} - 1$.)

```
 add r1, r0, r0
LOOP:
 andi r2, r1, #0x10
 beqz r2, CONTINUE
 addi r3, r3, #1
CONTINUE:
 addi r1, r1, #1
 j LOOP
```

Problem 3: Maybe, just maybe, the behavior of a branch in a procedure depends on where the procedure was called from. Suppose it does. Show how to implement a branch predictor that would use this information (the identity of the branch instruction and the caller of the "procedure" containing the branch instruction) to select branch history. The size of the BHT should be limited to 2^{12} entries. Assume that procedures are always called using the jal and jalr instructions. Be sure to show where the address lines for the BHT come from. The solution does not have to show details of the counter predict and update hardware.

Problem 4: Show how $32\,\text{b} \times 2^{23}$ memory devices can be connected to implement a 27-bit, byte-addressed address space in which the CPU fetches 64-bit aligned doublewords. (Using the notation from class, $a = 27$, $w = 64\,\text{b}$, $c = 8\,\text{b}$.) Indicate which memory devices store each of these addresses: `0x0, 0x7e33b8e, 0x3396891.`