LSU EE 4720 **Homework 6** Solution **Due: 25 April 2003**

Problem 1: Show the execution of the MIPS code fragment below for three iterations on a four-way dynamically machine using method 3 (physical register file) with a 256-entry reorder buffer. Though the machine is four-way, assume that there can be any number of write-backs per cycle.

- Assume that the branch and branch target are correctly predicted in IF so that when the branch is in ID the predicted target is being fetched.
- The FP multiply functional unit is three stages (M1, M2, and M3) with an initiation interval of 1.
- There are an unlimited number of functional units.

(*a*) Show the pipeline execution diagram, indicate where each instruction commits.

(*b*) Determine the CPI for a large number of iterations. (The method used for statically scheduled systems will work here but will be very inconvenient. There is a much easier way to determine the CPI.)

```
 # Solution
 # Cycle           0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
LOOP:
 ldc1 f0, 0(t1)    IF ID Q  L1 L2 WC
 mul  f2, f2, f0   IF ID Q        M1 M2 M3 WC
 bneq t1, t2 LOOP  IF ID Q  B  WB          C
 addi t1, t1, 8    IF ID Q  EX WB          C

 ldc1 f0, 0(t1)       IF ID Q  L1 L2 WB    C
 mul  f2, f2, f0      IF ID Q              M1 M2 M3 WC
 bneq t1, t2 LOOP     IF ID Q  B  WB                C
 addi t1, t1, 8       IF ID Q  EX WB                C

 ldc1 f0, 0(t1)          IF ID Q  L1 L2 WB          C
 mul  f2, f2, f0         IF ID Q                    M1 M2 M3 WC
 bneq t1, t2 LOOP        IF ID Q  B  WB                      C
 addi t1, t1, 8          IF ID Q  EX WB                      C
 # Cycle           0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
```

The CPI is $\frac{3}{4} = 0.75$. The hard way of computing the CPI is completing the pipeline execution diagram until there is a repeating pattern. With a 256-entry reorder buffer that will take a long time. (The ROB size was not given in the original problem.) The easy way is to find the critical path. The critical path must be through loop carried dependencies, for this loop there are two, `t1` and `f2`. There is a single instruction per iteration that updates `t1` and that has a latency of zero, so the path through `t1` can execute at a rate of one iteration per cycle, which is the same as the fetch rate. The path through `f2` is also through a single instruction, the multiply, however that has a latency of 2 (takes 3 cycles to compute) and so the fastest it can execute is 3 cycles per iteration. The processor will initially fetch one iteration per cycle and the `addi` instruction will be able to keep up, while the `mul.d` will fall behind. Eventually the reorder buffer will fill, when that happens instructions will only be fetched when new space opens up, which will be when the multiply instructions commit. Therefore fetch will drop to three cycles per iteration or a CPI of $\frac{3}{4}$.

Note that the load is not on the critical path. It does provide data for the multiply and it is dependent on data from a previous iteration, `t1`, but it has its data ready before the multiply needs it. (This is only so because of the assumption that the load always hits the cache. With cache misses the situation is more complex.)

Problem 2: The execution of a MIPS program on a one-way dynamically scheduled system is shown below. The value written into the destination register is shown to the right of each instruction. Below the program are tables showing the contents of the ID Map, Commit Map, and Physical Register File (PRF) at each cycle. The tables show initial values (before the first instruction is fetched), in the PRF table the right square bracket "]" indicates that the register is free. (Otherwise the right square bracket shows *when* the register is freed.)

(*a*) Show where each instruction commits.

(*b*) Complete the ID and Commit Map tables.

(*c*) Complete the PRF table. Show the values and use a "[" to indicate when a register is removed from the free list and a "]" to indicate when it is put back in the free list. Be sure to place these in the correct cycle.

```
# Solution

# Cycle          0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 (Result)
lw   r1, 0(r2)   IF ID Q  L1 L2             L2 WC                   (0x100)
ori  r1, r1, 6      IF ID Q                    EX WC                (0x106)
subi r2, r1, 2         IF ID Q                    EX WC             (0x104)
xor  r1, r3, r3           IF ID Q  EX WB                C          (0)
addi r2, r1, 0x700           IF ID Q  EX WB                C       (0x700)
subi r1, r2, 4                  IF ID Q  EX WB                C     (0x6fc)

# Cycle          0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
ID Map
r1       96         99 98    95    93
r2       92               97    94
# Cycle          0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
Commit Map
r1       96                                    99 98    95    93
r2       92                                          97    94
# Cycle          0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
Physical Register File
99       112 ]      [                          100]
98       583 ]         [                          106   ]
97       174 ]            [                          104   ]
96       309                                    ]
95       606 ]               [        0                     ]
94       058 ]                  [        700
93       285 ]                     [        6fc
92      1234                                          ]
91       518 ]
90       207 ]
# Cycle          0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
```