EE 4720 **Homework 4** Solution **Due: 3 November 2000**

Problem 1: Show a pipeline execution diagram for the execution of the DLX program below on a single-issue statically scheduled (plain old chapter 3) fully bypassed implementation in which the add functional unit is two stages (`A1`, `A2`) with an initiation interval of 2 (latency 3) and the multiply unit is six stages (`M1` through `M6`) with an initiation interval of 1 (latency 5). (This problem is very similar to Spring 2000 homework 3 problem 1. Check the solution to that assignment only if completely lost.)

```
 ! Solution
 addd  f0, f2, f4    IF ID A1 A1 A2 A2 WB
 addd  f6, f0, f8       IF ID -------> A1 A1 A2 A2 WB
 addd  f10, f12, f14       IF -------> ID -> A1 A1 A2 A2 WB
 multd f16, f18, f20                   IF -> ID M1 M2 M3 M4 M5 M6 WB
```

Problem 2: Show a pipeline execution diagram for the execution of the DLX program below on a single-issue statically scheduled fully bypassed implementation in which there are two add units, both consisting of one stage with an initiation interval of 4 (latency 3, unpipelined). Use symbol `A` for one adder and `B` for the other. The program below is slightly different than the one above.

```
 ! Solution
 addd  f0, f2, f4    IF ID A  A  A  A  WB
 addd  f6, f0, f8       IF ID -------> A  A  A  A  WB
 addd  f10, f12, f14       IF -------> ID B  B  B  B  WB
 addd  f16, f18, f20                   IF ID ----> A  A  A  A  WB
```

Problem 3: Show a pipeline execution diagram for the execution of the DLX program below on a two-way superscalar statically scheduled fully bypassed implementation in which there are two add units, both consisting of one stage with an initiation interval of 4 (latency 3, unpipelined). Use symbol `A` for one adder and `B` for the other.

```
 ! Solution
LINE1: ! LINE1 = 0x1000
 addd  f0, f2, f4    IF ID A  A  A  A  WB
 addd  f6, f0, f8    IF ID ----------> A  A  A  A  WB
 addd  f10, f12, f14    IF ----------> ID B  B  B  B  WB
 addd  f16, f18, f20    IF ----------> ID -------> A  A  A  A  WB
```

Problem 4: Show a pipeline execution diagram for the DLX code below executing on a processor with the following characteristics:

- Statically scheduled two-way superscalar.
- Unlimited number of functional units.
- Six stage fully pipelined multiply.
- Can handle an **unlimited** number of write backs per cycle. (Unrealistic, but reduces adidactic tedium.)
- Fully bypassed, including the branch condition.

The diagram should start at the first iteration and end after 30 cycles or until a repeating pattern is encountered, whichever is sooner. Note that there is a floating-point loop-carried dependency (`f2`). What is the CPI for a large number of iterations?

```
LOOP: ! LOOP = 0x1004
! Cycle           0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22
 ld   f0, 0(r1)   IF ID EX ME WB    IF ID EX ME WB    IF ID EX ME WB    IF ID EX ME WB
 muld f2, f0, f2     IF ID -> M1 M2 M3 M4 M5 M6 WB
                                    IF ID -> M1 M2 M3 M4 M5 M6 WB
                                                       IF ID -> M1 M2 M3 M4 M5 M6 WB
 addi r1, r1, #8     IF ID EX ME WB    IF ID EX ME WB    IF ID EX ME WB
 sub  r2, r1, r3        IF -> ID EX ME WB IF -> ID EX ME WB IF -> ID EX ME WB
 bneq r2, LOOP          IF -> ID -> EX ME WB
                                       IF -> ID -> EX ME WB
                                                          IF -> ID -> EX ME WB
 xor  r10, r11, r12           IF ->x
 and  r13, r14, r15           IF ->x
```

Problem 5: Unroll and schedule the loop from the problem above for maximum efficiency. Unroll the loop four times; the number of iterations will always be a multiple of four. Use software pipelining and take advantage of associativity to overlap the multiply latency. (In software pipelining a computation is spread over several iterations.) Code may be added before the `LOOP` label.

```
 nop
LOOP: ! LOOP = 0x1008
! Cycle           0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22
 ld   f0, 0(r1)   IF ID EX ME WB       IF ID EX ME WB
 ld   f10, 8(r1)  IF ID EX ME WB       IF ID EX ME WB
 ld   f12, 12(r1)    IF ID EX ME WB       IF ID EX ME WB
 ld   f14, 16(r1)    IF ID EX ME WB       IF ID EX ME WB
 addi r1, r1, #32       IF ID EX ME WB       IF ID EX ME WB
 muld f0, f0, f24       IF ID M1 M2 M3 M4 M5 M6 WB
                                          IF ID M1 M2 M3 M4 M5 M6 WB
 sub  r2, r1, r3           IF ID EX ME WB
 muld f24, f20, f22        IF ID M1 M2 M3 M4 M5 M6 WB
 muld f20, f0, f10            IF ID M1 M2 M3 M4 M5 M6 WB
 muld f22, f12, f14           IF ID M1 M2 M3 M4 M5 M6 WB
 bneq r2, LOOP                   IF ID EX ME WB
 xor                             IF IDx
                                    IFx
```