EE 4720 **Homework 4** **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.)

```
addd  f0, f2, f4
addd  f6, f0, f8
addd  f10, f12, f14
multd f16, f18, f20
```

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.

```
addd  f0, f2, f4
addd  f6, f0, f8
addd  f10, f12, f14
addd  f16, f18, f20
```

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.

```
LINE1: ! LINE1 = 0x1000
 addd  f0, f2, f4
 addd  f6, f0, f8
 addd  f10, f12, f14
 addd  f16, f18, f20
```

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
 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 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.