EE 4720 **Homework 4 & 5** Solution **Due: 23 April 1999**

In all problems below assume there are no cache misses and that all register values are available at the beginning of execution.

Problem 1: Show a pipeline execution diagram for the first 41 cycles of the code below on a dynamically scheduled implementation of DLX in which:

- There is one floating point multiply unit with a latency of 5 and an initiation interval of **2**.
- There is a load/store functional unit with a latency of 1. The segments are labeled `L1` and `L2`.
- The FP add functional unit has a latency of 3 and an initiation interval of 1.
- The integer functional unit has a latency of 0 and an initiation interval of 1.
- The functional units have reservation stations with the following numbers: integer, 6-9; fp add, 0-1; fp multiply, 2-3; load/store, 4-5.
- There is no reorder buffer.
- The branch delay is one. (There are no branch delay slots.)
- Ignore load/store ordering.

Initially all reservation stations are available.

```
LOOP:
 addi  r1, r1, #8
 sub   r2, r1, r3
 lf    f0, 0(r1)
 multf f1, f0, f0
 multf f2, f0, f1
 sf    4(r1), f1
 bneq  r2, LOOP   ! Assume always taken.
 xor   r4, r5, r6
```

```
LOOP:
Cycle              0    1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16   17   18   19   20   21   22
 addi  r1, r1, #8  IF   ID 6:EX 6:WB                      IF   ID 6:EX 6:WB                                                IF   ID
 sub   r2, r1, r3       IF   ID 7:EX 7:WB                      IF   ID 7:EX 7:EX 7:WB                                           IF
 lf    f0, 0(r1)             IF   ID 4:L1 4:L2 4:WB                 IF   ID 5:RS 5:L1 5:L2 5:WB
 multf f1, f0, f0                 IF   ID 2:RS 2:M1 2:M1 2:M2 2:M2 2:M3 2:M3 2:WB
                                                                   IF   ID 2:RS 2:RS 2:M1 2:M1 2:M2 2:M2 2:M3 2:M3 2:WB
 multf f2, f0, f1                      IF   ID 3:RS 3:RS 3:RS 3:RS 3:RS 3:RS 3:M1 3:M1 3:M2 3:M2 3:M3 3:M3 3:WB
                                                                        IF   ID ----------------------> 3:RS 3:RS 3:M1 3:M1
 sf    4(r1), f1                            IF   ID 4:RS 4:RS 4:RS 4:RS 4:RS 4:L1 4:L2 4:WB
                                                                             IF ----------------------->   ID 4:RS 4:L1 4:L2
 bneq  r2, LOOP                                  IF   ID                                                   IF   ID
 xor   r4, r5, r6                                     IF   x                                                     IF   x


LOOP:
Cycle              20   21   22   23   24   25   26   27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42
 addi  r1, r1, #8       IF   ID 6:EX 6:WB                           IF   ID 6:EX 6:WB
 sub   r2, r1, r3            IF   ID 7:EX 7:WB                           IF   ID 7:EX 7:EX 7:WB
 lf    f0, 0(r1)                 IF   ID 4:L1 4:L2 4:L2 4:WB                 IF   ID 5:RS 5:L1 5:L2 5:WB
 multf f1, f0,f0 2:M3   WB            IF   ID 2:RS 2:RS 2:M1 2:M1 2:M2 2:M2 2:M3 2:M3 2:WB
                                                                             IF   ID 2:RS 2:RS 2:M1 2:M1 2:M2 2:M2 2:M3 2:M3
 multf f2, f0,f1 3:RS 3:M1 3:M1 3:M2 3:M2 3:M3 3:M3 3:WB
                                            IF   ID ---> 3:RS 3:RS 3:RS 3:RS 3:RS 3:RS 3:M1 3:M1 3:M2 3:M2 3:M3 3:M3 3:WB
                                                                                IF   ID ----------------------> 3:RS 3:RS
 sf    4(r1), f1 4:RS 4:L1 4:L2 4:WB            IF ---->   ID 4:RS 4:RS 4:RS 4:RS 4:RS 4:L1 4:L2 4:WB
                                                                                     IF ----------------------->   ID 4:L1
 bneq  r2, LOOP    ID                                     IF   ID                                                   IF   ID
 xor   r4, r5, r6  IF   x                                      IF   x                                                     IF
```

Note: In cycle 12 the load waits an extra cycle because L1 is being used by the store. (As a general rule, the instruction waiting longer should start first. When contending for the CDB, the functional unit with the longer latency gets priority.)

Problem 2: Determine the CPI for a large number of iterations of the loop above (or give a good reason why it would be very difficult to determine the CPI).

Consider the state of the machine when fetching the first instruction of the loop, `addi`. It is the same at cycle 8 and 30 (for example, in both cases the first multiply from the previous iteration started the second multiply stage, the `sf` and second `multf` are sitting in reservation stations, and the number of free reservation stations at each functional unit is the same in both cycles). There are 7 instructions per iteration, so the CPI is $(30 - 8)/(2 \times 7) = 1.571$ CPI.

Problem 3: What are the minimum number of reservation stations of each type needed so that the code above executes at maximum speed? What is the CPI at maximum speed? *(This part was not in the problem as originally assigned:)* The CDB can handle any number of writebacks per cycle and there are an unlimited number of functional units.

The problem as originally assigned was more tedious than intended. To solve it one would need to find a repeating pattern of iterations. Because of contention for the CDB, the repeating pattern does not occur in the first few iterations and so one would have to tediously construct the diagram for many iterations.

To solve this problem construct a pipeline execution diagram assuming an unlimited number of reservation stations. The diagram should continue until every instruction in the first iteration completes. (This loop does not have inter-iteration dependencies, but if it did [*e.g.*, if the second multiply were `multf f2, f1, f2`] the diagram would continue until every instruction in the second iteration finished.) From the diagram find the maximum number of reservation stations used. For the code above the diagram should be continued until cycle 18 (a few extra cycles are shown):

```
LOOP:
Cycle              0    1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16   17   18   19   20   21   22
 addi  r1, r1, #8  IF   ID   EX   WB                        IF   ID   EX   WB                        IF   ID   EX   WB
 sub   r2, r1, r3       IF   ID   EX   WB                        IF   ID   EX   WB                        IF   ID   EX   WB
 lf    f0, 0(r1)             IF   ID   L1   L2   WB                   IF   ID   L1   L2   WB                   IF   ID   L1   L2   WB
 multf f1, f0, f0                IF   ID   RS   M1   M1   M2   M2   M3   M3   WB
                                                                      IF   ID   RS   M1   M1   M2   M2   M3   M3   WB
                                                                                                                IF   ID   RS   M1
 multf f2, f0, f1                     IF   ID   RS   RS   RS   RS   RS   RS   M1   M1   M2   M2   M3   M3   WB
                                                                           IF   ID   RS   RS   RS   RS   RS   RS   M1   M1   M2
                                                                                                                     IF   ID   RS
 sf    4(r1), f1                           IF   ID   RS   RS   RS   RS   RS   L1   L2   WB
                                                                                IF   ID   RS   RS   RS   RS   RS   L1   L2   WB
                                                                                                                          IF   ID
 bneq  r2, LOOP                                 IF   ID                                IF   ID                                    IF
 xor   r4, r5, r6                                    IF   x                                   IF   x
```

Two integer RS are needed (cycle 3), zero FP add RS are needed, three FP multiply units are needed (cycle 14),

Problem 4: The code below executes on a machine similar to the type described in the first problem except that it uses a reorder buffer. Draw a pipeline execution diagram for the code below, be sure to show when each instruction commits. Remember that instructions stall in the functional unit if they are not granted access to the CDB.

```
LOOP:
 lf    f0, 0(r1)
 multf f1, f0, f0
 multf f2, f0, f1
 addf  f3, f3, f0
 lf    f4, 8(r1)
 sf    4(r1), f1
 multf f1, f4, f4
 multf f2, f4, f1
 addi  r1, r1, #16
 sub   r3, r4, r5
 xor   r6, r7, r8
 or    r9, r10, r11
```

Since a re-order buffer is being used instruction results will be identified by their reorder buffer entry number rather than their reservation station number. For that reason reservation stations are only held until execution initiation. For example, the first `multf` only needs a RS in cycle 3.

```
LOOP:
Cycle              0    1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16   17   18   19   20   21 22 23 24 25
 lf    f0, 0(r1)   IF   ID   L1   L2   WC
 multf f1, f0, f0       IF   ID 2:RS   M1   M1   M2   M2   M3   M3   WC
 multf f2, f0, f1            IF   ID 3:RS 3:RS 3:RS 3:RS 3:RS 3:RS   M1   M1   M2   M2   M3   M3   WC
 addf  f3, f3, f0                 IF   ID   A1   A2   A3   A4   WB                                     C
 lf    f4, 8(r1)                       IF   ID   L1   L2   WB                                          C
 sf    4(r1), f1                            IF   ID 4:RS 4:RS 4:RS   L1   L2   WB                              C
 multf f1, f4, f4                                IF   ID   M1   M1   M2   M2   M3   M3   WB                          C
 multf f2, f4, f1                                     IF   ID 3:RS 3:RS 3:RS 3:RS 3:RS   M1   M1   M2   M2   M3   M3   WB  C
 addi  r1, r1, #16                                         IF   ID   EX   WB                                                C
 sub   r3, r4, r5                                               IF   ID   EX ---->   WB                                        C
 xor   r6, r7, r8                                                    IF   ID 7:RS   EX ---->   WB                                   C
 or    r9, r10, r11                                                       IF   ID 6:RS 6:RS   EX ---->   WB                              C
```

Problem 5: Consider the code execution from the problem above. Suppose there is an exception in the `L2` segment executing the second `lf`. At what cycle would the trap instruction be inserted? What might go wrong if a reorder buffer had not been used?

The trap will be inserted when `lf` reaches the head of the reorder buffer, at cycle 18. If a reorder buffer were not used and the preceding multiply raised an exception, its trap handler might run before the one for `lf`.

Problem 6: Show the execution of the code below on a dynamically scheduled 4-way superscalar machine using a reorder buffer. Instruction fetch is aligned. There is one of each floating-point functional unit, with latencies and initiation intervals given in the first problem. There are four integer execution units. The reservation station numbers are as given in the first problem.

```
LOOP: = 0x1008
 lf    f0, 0(r1)
 multf f1, f0, f0
 multf f2, f0, f1
 addf  f3, f3, f0
 lf    f4, 8(r1)
 sf    4(r1), f1
 multf f1, f4, f4
 multf f2, f4, f1
 addi  r1, r1, #16
 sub   r3, r4, r5
 xor   r6, r7, r8
 or    r9, r10, r1
```

```
LOOP: = 0x1008 = 1 0000 0000 1000
Cycle              0    1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16   17   18   19   20   21   22
 lf    f0, 0(r1)   IF   ID   L1   L2   WC
 multf f1, f0, f0  IF   ID 2:RS 2:RS   M1   M1   M2   M2   M3   M3   WC
 multf f2, f0, f1       IF   ID 3:RS 3:RS 3:RS 3:RS 3:RS 3:RS 3:RS   M1   M1   M2   M2   M3   M3   WC
 addf  f3, f3, f0       IF   ID 0:RS   A1   A2   A3   A4   WB                                       C
 lf    f4, 8(r1)        IF   ID   L1   L2   WB                                                      C
 sf    4(r1), f1        IF   ID 4:RS 4:RS 4:RS 4:RS 4:RS 4:RS 4:RS   L1   L2   WB                  C
 multf f1, f4, f4            IF   ID 2:RS 2:RS   M1   M1   M2   M2   M3   M3   WB                       C
 multf f2, f4, f1            IF   ID --------> 2:RS 2:RS 2:RS 2:RS 2:RS 2:RS   M1   M1   M2   M2   M3   M3   WC
 addi  r1, r1, #16           IF   ID   EX   WB                                                           C
 sub   r3, r4, r5            IF   ID   EX   WB                                                           C
 xor   r6, r7, r8                 IF --------->   ID   EX   WB                                           C
 or    r9, r10, r11               IF --------->   ID   EX   WB                                                C
```

In the diagram above, WC indicates that writeback and commit occur in the same cycle. Note that since instructions are fetched in aligned blocks of four, only two useful instructions are fetched in cycle 0.

Problem 7: Rewrite the code below for the VLIW DLX ISA presented in class. Instructions can be rearranged and register numbers changed. Try to minimize the number of bundles, minimize the use of the serial bit, and maximize the value of the lookahead field. Use a "?" for the lookahead field in the last bundle.

```
LOOP:
 lf    f0, 0(r1)
 multf f1, f0, f0
 multf f2, f0, f1
 addf  f3, f3, f0
 lf    f4, 8(r1)
 sf    4(r1), f1
 multf f1, f4, f4
 multf f2, f4, f1
 addi  r1, r1, #16
 sub   r3, r4, r5
 xor   r6, r7, r8
 or    r9, r10, r11
```

Solution:

```
LOOP:
{ P 0
 lf    f0, 0(r1)
 lf    f4, 8(r1)
 addi  r1, r1, #16
}
{ P 0
 multf f1, f0, f0
 multf f11, f4, f4
 sub   r3, r4, r5
}
{ P 1
 multf f2, f0, f1
 addf  f3, f3, f0
 multf f2, f4, f11
}
{ P ?
 sf    -12(r1), f1
 xor   r6, r7, r8
 or    r9, r10, r11
}
```