LSU EE 4720 **Homework 3** Solution **Due: 15 March 2004**

Problem 1: The MIPS program below copies a region of memory and runs on the illustrated implementation. In the sub-problems below use only the bypass connections shown in the illustration.

(*a*) Show a pipeline execution diagram for the code running on the illustrated implementation for two iterations.

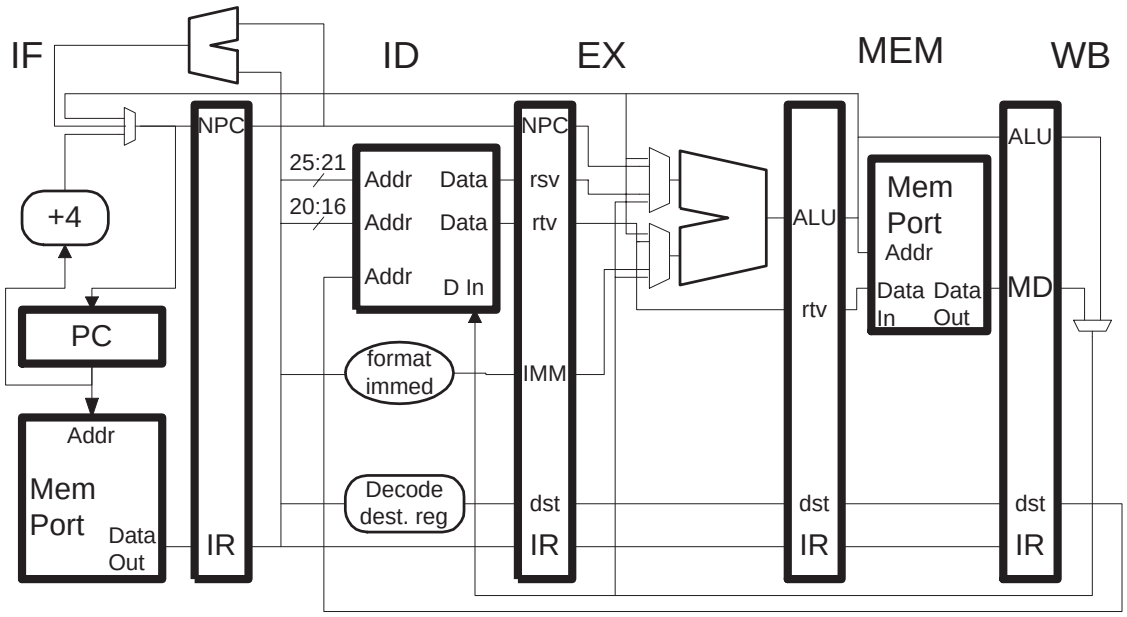
See below.

(*b*) Compute the CPI and the rate at which memory is copied in bytes per cycle assuming a large number of iterations.

Each iteration takes 9 cycles and contains 5 instructions so the CPI is $\frac{9}{5} = 1.8$. Each iteration copies four bytes of data and so the data copy rate is $\frac{4}{9}$ bytes per cycle.

- Don't forget, when computing the number of cycles per iteration be sure not to count a cycle more, or less, than once.

```
LOOP:  # Cycle       0  1  2  3  4  5  6  7  8  9  10 11 12
 lw $t0, 0($a0)     IF ID EX ME WB             IF ID EX ME
 sw 0($a1), $t0        IF ID ----> EX ME WB       IF ID ->
 addi $a0, $a0, 4      IF ----> ID EX ME WB       IF ->
 bne $a0, $a2 LOOP                IF ID ----> EX ME WB
 addi $a1, $a1, 4                    IF ----> ID EX ME WB
```



Problem 2: Execution should be inefficient in the problem above.

(*a*) Add **exactly** the bypass connections needed so that the program above executes as fast as possible.

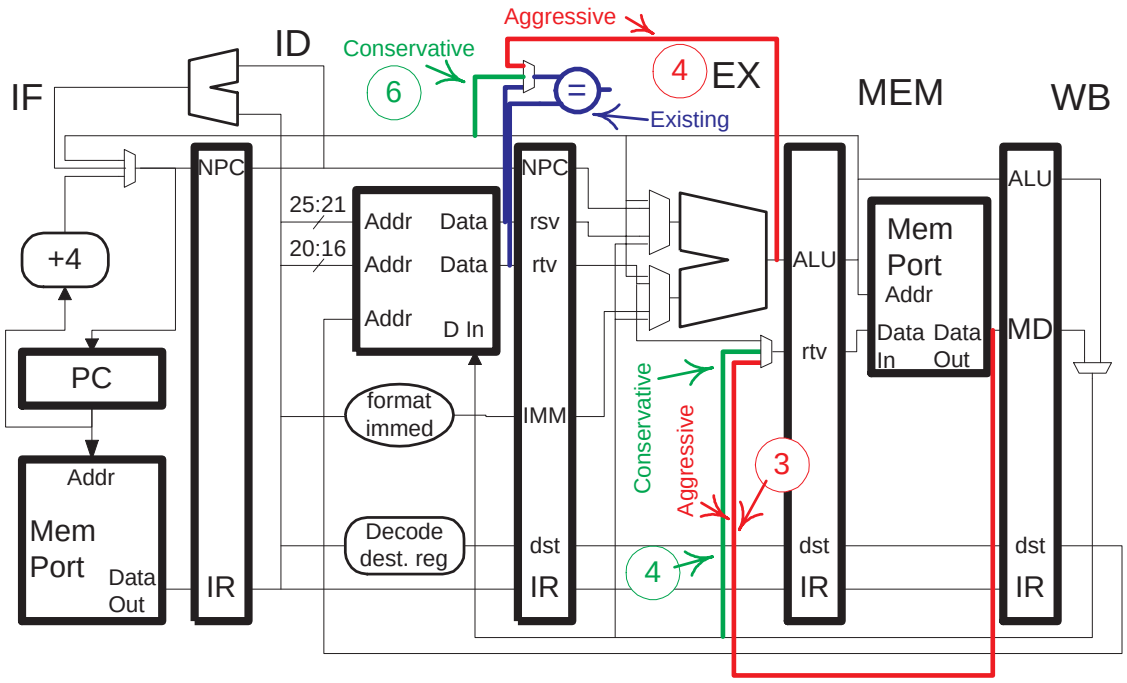
- Don't forget that branch uses ID-stage comparison units.
- Don't forget the store.

There are two solutions, conservative and aggressive. With the conservative solution the number of stall cycles will be reduced from two to one (zero would be better) in each case and the critical path length will not be increased (which is good, of course). In the aggressive solution all stalls will be eliminated but there might be critical-path impact, and so the clock frequency might be reduced.

The aggressive solution improves performance on the program above because the number of cycles per iteration is reduced from 9 to 5, while any reduction in clock frequency would not be as drastic. For programs which do not make frequent use of the new bypasses performance would be lower because there would not be enough of a reduction in stall cycles to compensate for the lower clock frequency.

Note that one could have an aggressive load-store bypass and a conservative branch condition bypass, and vice versa.

In the illustration below the bypass paths for the conservative solution are shown in green, the bypass paths for the aggressive solution are shown in red, and the comparison unit (which was present but not shown in the original diagram) appears in blue.



(*b*) Show a pipeline execution diagram of the code on the improved implementation.

```
# Aggressive.
LOOP:  # Cycle       0  1  2  3  4  5  6  7  8  9
 lw $t0, 0($a0)      IF ID EX ME WB IF ID EX ME ...
 sw 0($a1), $t0         IF ID EX ME WB IF ID EX ...
 addi $a0, $a0, 4          IF ID EX ME WB IF ID ...
 bne $a0, $a2 LOOP            IF ID EX ME WB IF ...
 addi $a1, $a1, 4                 IF ID EX ME WB

# Conservative
LOOP:  # Cycle       0  1  2  3  4  5  6  7  8  9
 lw $t0, 0($a0)      IF ID EX ME WB       IF ID EX ME ...
 sw 0($a1), $t0         IF ID -> EX ME WB    IF ID -> ...
 addi $a0, $a0, 4          IF -> ID EX ME WB    IF -> ...
 bne $a0, $a2 LOOP               IF ID -> EX ME WB
 addi $a1, $a1, 4                   IF -> ID EX ME WB
```

(*c*) For each bypass path that you've added show the cycles in which it will be used by writing the cycle number near the bypass path. If a bypass path goes to several places (for example, both ALU muxen) put the cycle number at the place(s) that use the signal.

(*d*) Re-compute the CPI and the rate at which memory is copied.

With the aggressive solution the CPI is $\frac{5}{5} = 1$ and the copy rate is $\frac{4}{5}$ bytes per cycle. With the conservative solution the CPI is $\frac{7}{5} = 1.4$ and the copy rate is $\frac{4}{7}$ bytes per cycle.