LSU EE 4720 Homework 1 Due: 20 February 2015

Problem 1: Answer each MIPS code question below. Try to answer these by hand (without running code).

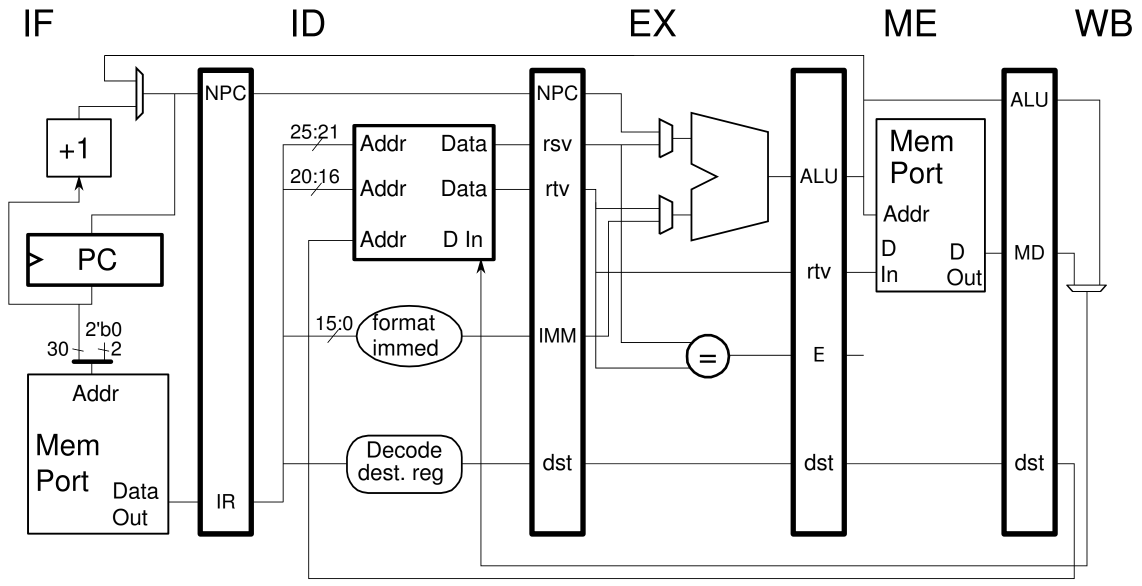
(*a*) Where indicated, show the changed register in the following simple code fragments:

```
# r1 = 10, r2 = 20, r3 = 30, etc.
#
add r1, r2, r3
#
# Changed register, new value:

# r1 = 10, r2 = 20, etc.
#
add r1, r1, r2
#
# Changed register, new value:
```

(*b*) Show the values assigned to registers `s1` through `s6` in the code below. Correctly answering this question requires an understanding of MIPS big-endian byte ordering and of the differences between `lw`, `lbu`, and `lb`. Refer to the MIPS review notes and MIPS documentation for details.

```
        .data
values: .word 0x11121314
        .word 0xaabbccdd
        .word 0x99887766
        .word 0x41424344

        .text
        la  $s0, values  # Load $s0 with the address of the first value above.
        lw  $s1, 0($s0)
        lw  $s2, 4($s0)
        sh  $s2, 0($s0)  # Note: this is a store *half*.
        lbu $s3, 0($s0)
        lbu $s4, 3($s0)
        lb  $s5, 4($s0)
        lb  $s6, 7($s0)
```

Problem 2: *Note: The following problem was assigned last year and its solution is available. DO NOT look at the solution unless you are lost and can't get help elsewhere. Even in that case just glimpse.* Appearing below are **incorrect** executions on the illustrated implementation. For each one explain why it is wrong and show the correct execution.



(*a*) Explain error and show correct execution.

```
LOOP: # Cycles     0  1  2  3  4  5  6  7
 lw r2, 0(r4)      IF ID EX ME WB
 add r1, r2, r7       IF ID EX ME WB
LOOP: # Cycles     0  1  2  3  4  5  6  7
```

(*b*) Explain error and show correct execution.

```
LOOP: # Cycles     0  1  2  3  4  5  6  7
 add r1, r2, r3    IF ID EX ME WB
 lw r1, 0(r4)         IF ID -> EX ME WB
LOOP: # Cycles     0  1  2  3  4  5  6  7
```

(*c*) Explain error and show correct execution.

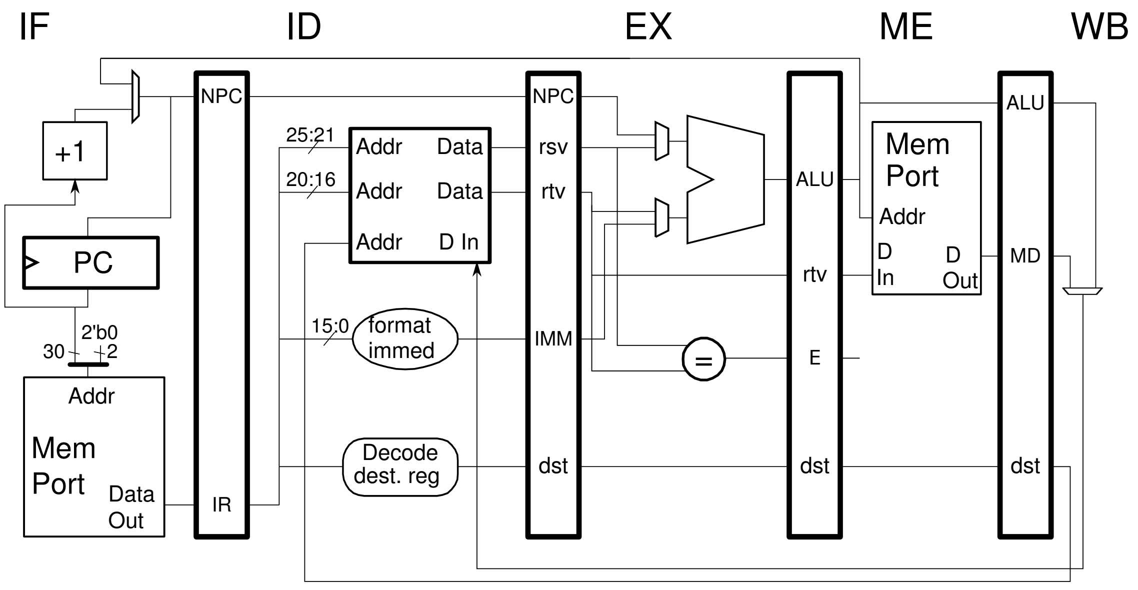
```
LOOP: # Cycles     0  1  2  3  4  5  6  7
 add r1, r2, r3    IF ID EX ME WB
 sw r1, 0(r4)         IF ID -> EX ME WB
LOOP: # Cycles     0  1  2  3  4  5  6  7
```

(*d*) Explain error and show correct execution.

```
LOOP: # Cycles     0  1  2  3  4  5  6  7
 add r1, r2, r3    IF ID EX ME WB
 xor r4, r1, r5       IF ----> ID EX ME WB
LOOP: # Cycles     0  1  2  3  4  5  6  7
```

Problem 3: Show the execution of the MIPS code below on the illustrated implementation. The register file is designed so that if the same register is simultaneously written and read, the value that will be read will be value being written. (In class we called such a register file *internally bypassed.*)

- Check carefully for dependencies.
- Pay attention to which registers are sources and which are destinations, especially for the `sw` instruction.
- Be sure to stall when necessary.

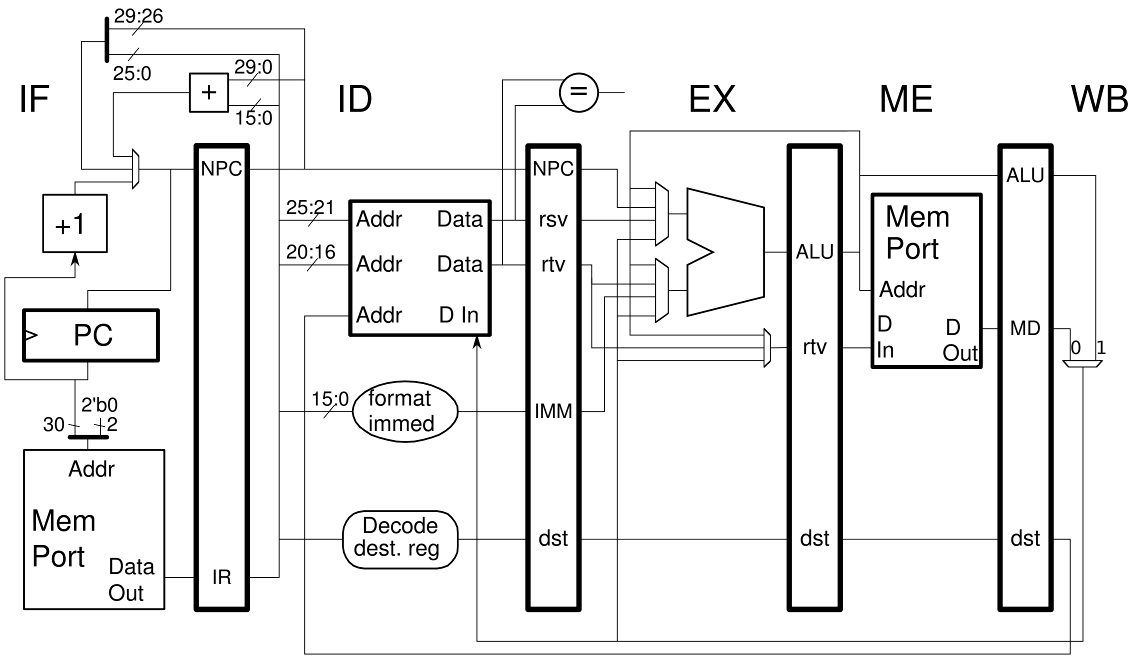


```
add r1, r2, r3

lw r4, 0(r1)

sw r1, 0(r1)

sub r5, r4, r1

sh r5, 4(r1)
```

Problem 4: The code below is the same as the code used in the previous problem, but the MIPS implementation is different.

(*a*) Show the execution of the MIPS code below on the illustrated implementation.

(*b*) On the diagram label multiplexor data inputs connecting to bypass paths that are used in the execution of this code. The label should include the cycle number, the register, and the instruction consuming the value. For example, the label $\boxed{\text{3:r1:lw}}$ might be placed next to one of the data inputs on the ALU's upper mux.



```
add r1, r2, r3

lw r4, 0(r1)

sw r1, 0(r1)

sub r5, r4, r1

sh r5, 4(r1)
```