LSU EE 4720 **Homework 1** **Due: 6 February 2017**

Problem 1: Follow the instructions for class account setup and for homework workflow. Review the comments in `hw01.s` and look for the area labeled "Problem 1," which has an empty procedure named `split`. The `split` routine itself is reproduced below, and for those who want to start before getting a class account the assembler for the entire assignment can be found at `http://www.ece.lsu.edu/ee4720/2017/hw01.s.html`.

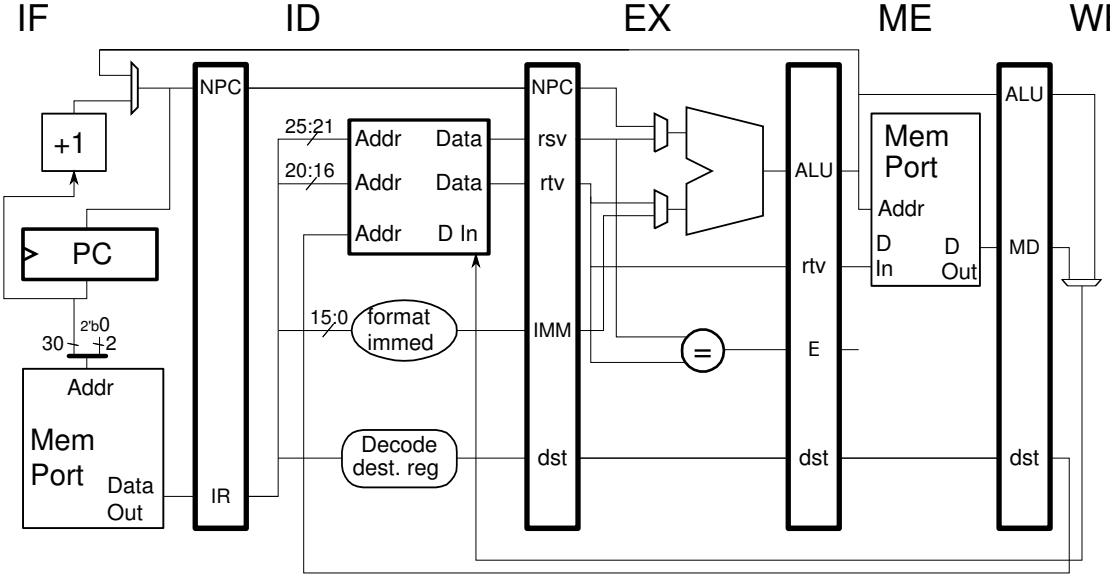
Routine `split` is called with four arguments. Register `a0` holds the number of elements in an input array, register `a1` holds the address of the input array. The input array consists of 4-byte elements. Register `a2` holds the address of a 4-byte-element output array and register `a3` holds the address of a 2-byte-element output array. Complete split so that it copies elements from the input array to the 2-byte output array (if it fits), otherwise to the 4-byte output array. The return value, in register `v0`, should be the number of elements in the 4-byte output array when copying is done.

When the code in the hw01.s file is run the split routine will be tested. The first line of output indicates whether the return value, `v0`, was correct. Below that the values written to the 4- and 2-byte arrays are shown side-by-side with correct values.

```
split:
        ## Register Usage
        #
        # CALL VALUES:
        #  $a0: Number of elements in input array.
        #  $a1: Address of input array. Elements are 4 bytes each.
        #  $a2: Address of output array, elements are 4 byte each.
        #  $a3: Address of output array, elements are 2 byte each.
        #
        # RETURN:
        #  [ ] $v0, Number of elements in 4-byte output array.
        #  [ ] Write small elements in $a3 array and large elements in $a2.
        #
        # Note:
        #  Can modify $t0-$t9, $a0-$a3

        # [ ] Code should be correct.
        # [ ] Code should be reasonably efficient.

        # SOLUTION GOES HERE









        jr $ra
        nop
```

There's another problem on the next page.

Problem 2: *Note: The following problem was assigned last year, two years ago, and three years ago 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
```