University of California, Berkeley
College of Engineering
Computer Science Division — EECS

Spring 1999 John Kubiatowicz

Homework Quiz (HW #4)

March 15, 1999
CS152 Computer Architecture and Engineering

This quiz covers one of the problems from homework #4.
Good Luck!

Your Name:	
SID Number:	
Discussion Section:	

Total:	

The VSCALE instrucition

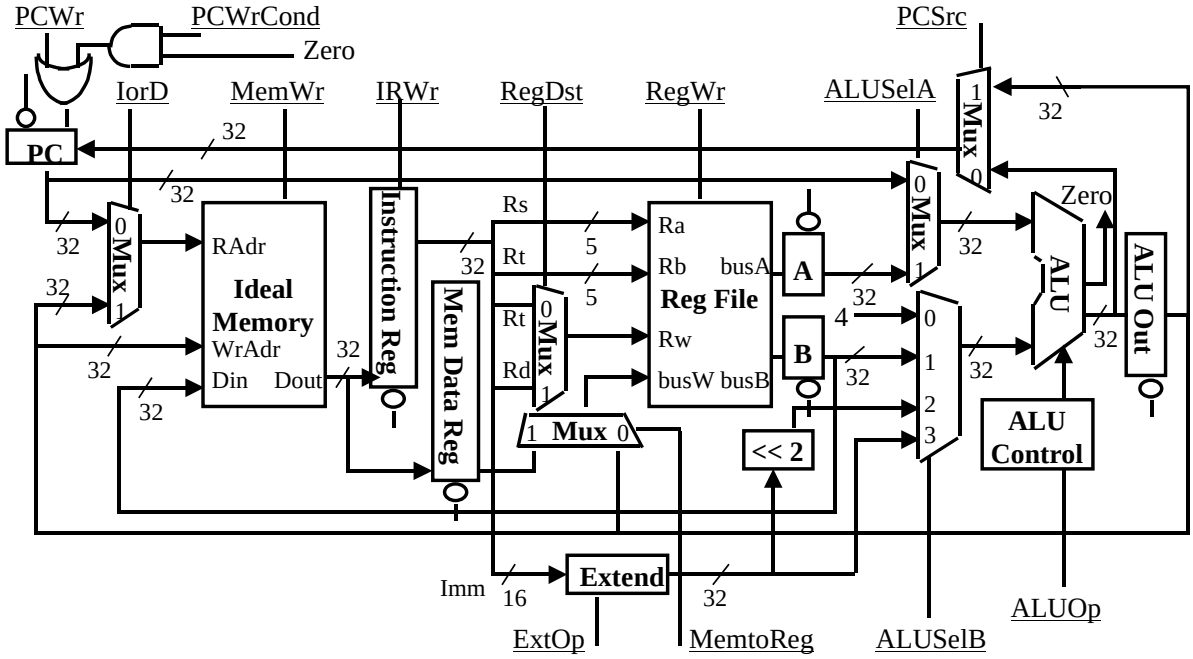

Figure 1: A multicycle data path

In your homework, you implemented the `bcp` (block copy) instruction. Here is the pseudo-code for a related instruction, called `vscale`. It is the same as block copy, except that it multiplies every source word by a constant before copying it to its destination. Let *v1* be the starting address of the input vector, *v2* be the starting address of the destination vector, *length* be the length of both vectors (in words), and *scale* be the scale factor:

```
vscale(v1,v2,length,scale) {
    int index = (length-1)×4;    /* mult ×4 since we want byte addr */

    while (index >= 0) {
        M[v2+index] = scale × M[v1+index];
        index = index – 4;
    }
}
```

Your job is to implement the `vscale` instruction in the multicycle data path. Here is the coding of `vscale` instruction:

opcode = Instruction[31:26] = 0x31 (an unused opcode)
R[rs]⇒v1, R[rt]⇒v2, R[rd]⇒length-1, Instruction[10:0]⇒scale

The *scale* value is hard-coded into the lower 11 bits of the instruction. Assume that it is a signed number. Note that we have to read *three* registers to make this work, i.e. rs, rt, *and* rd. Since scaling a vector of zero length isn't very interesting, assume that R[rd] contains the length –1.

Problem 1a: Start with Figure 1. Assume that you are not allowed to add additional ALUs or adders, but that the current ALU supports multiply. What is the lowest number of cycles that you might expect to spend per vector element? Why? *Hint: use the ALU every cycle.*

Problem 1b: What needs to be added to the datapath to implement `vscale` with this number of cycles in the inner loop? Add *only* registers, muxes, and simple logic. You can modify existing components. Assume that the ALU supports a multiply operation and ignore overflow. Note that the ALU takes a complete cycle to compute and memory takes a complete cycle to read or write. *Hint: you will need to add new registers and may need to add enable lines to existing registers.*

[Over]

Problem 1c: Use pseudo-code and register-transfer language to describe what is happening in each cycle of your instruction. *Be very clear about which things are happening at the same time!*