Fall 2016

CSc 256 Chapter 3 Lab Problem

(due 5pm Tue 9/27/2016; no late submissions)

(3% of your grade)

From unixlab.sfsu.edu, copy or download the files

```
~whsu/csc256/LABS/PROGS/Prob3.F16.cpp
~whsu/csc256/LABS/PROGS/Prob3.F16.s
```

Note: you can also access it through a web browser at

http://unixlab.sfsu.edu/~whsu/csc256/LABS/PROGS/Prob3.F16.cpp
http://unixlab.sfsu.edu/~whsu/csc256/LABS/PROGS/Prob3.F16.s

The C++ program steps through the array pos[]. For each i, if pos[i] >= 0, data[i] is saved in x[pos[i]]. (This is called a *scatter* operation; it's very common in a lot of scientific applications.) Compile and run the program; it should print:

```
0
73
0
47
0
0
23
0
0
26
```

In this exercise, you'll try to translate the C++ program to MIPS. Some of the more tedious parts are already given in Prob3.F16.s. You won't have to write the data allocation sections, some of the initializations, and the output loop at the end.

Here's what you have to fill in:

```
main:la   $s1, x
     la   $s2, pos
     la   $s3, data
     li   $s6, 6
                    #  for (int i=0; i<6; i++) {
for:                #    if (pos[i] >= 0) {
```

```
                        #       int temp = pos[i];

                        #       x[temp] = data[i];



                        #     }
                        #   }
```

As we did in Lab 2.3, we first identify the for loop, and translate it:

```
main:la   $s1, x
     la   $s2, pos
     la   $s3, data
     li   $s6, 6
     li   $s0, 0          #  for (int i=0; i<6; i++) {
for:                      #    if (pos[i] >= 0) {



                          #       int temp = pos[i];

                          #       x[temp] = data[i];



     addi $s0, $s0, 1     #    }
     blt  $s0, $s6, for   #  }
```

Note that we don't have to test i<6 at the start of the loop! This is because we know that 0<6.

Next, we deal with the if statement. We have to compare pos[i] with 0]. pos[i] has to be in a register before we can compare it with a conditional branch.

First we have to calculate the address of pos[i]. According to Chapter 3 slides, this is &pos[0] + i*4. (Remember earlier, we loaded &pos[0] into $s2.) So we compute &pos[i] and load pos[i] into a temporary register:

```
     li   $s0, 0          #  for (int i=0; i<6; i++) {
for: mul  $t0, $s0, 4     #    if (pos[i] >= 0) {
     add  $t0, $t0, $s2
     lw   $t1, ($t0)
```

```
                        #       int temp = pos[i];

                        #       x[temp] = data[i];



addi $s0, $s0, 1    #    }
blt  $s0, $s6, for  #  }
```

Then we can test pos[i] >= 0 using a conditional branch. The true part of the if statement is

```
                        #       int temp = pos[i];

                        #       x[temp] = data[i];
```

We already have pos[i] earlier. We'll need to 1) calculate &data[i], 2) calculate &x[temp], 3) load data[i], and finally 4) store x[temp] .

Fill out all the missing code in Prob3.F16.s; make sure your final version runs and produces the same results as the C++ program. Submit your code using the iLearn submission link.

Note: there are more efficient ways to translate this program. All the accesses to pos[i] and data[i] are actually *sequential* accesses, so you can rewrite them as pointer dereferences and save some address calculations. You are not required to make this optimization for this assignment, but you might want to try to get it to work as a challenge! (Only a few small changes are needed.)