Lab Week #7
CIS 314

November 13, 2017

1 Pipeline Stalls

This week's assignment consists of two parts, a giant sort function and a question about pipeline stalls. For this part, I simply did the example from section 4.5 in the book (first mentioned on page 409), which was as follows:

```
#how many pipeline stalls (or bubbles) will be necessary for the following code
#   with and without forwarding
irmovl $3, %eax
irmovl $10, %edx
addl %eax, %edx
```

2 y86

Selection sort relies on the ability to find the index associated with the minimal element in the unsorted portion of the array. For this lab, (since I don't want to give away too much) I decided to write some function which would take an array and return the memory location of the least element. It's not directly applicable to the assignment, but it may help with some concepts[1].

```
.pos 0
Init:
    irmovl Stack, %ebp
    irmovl Stack, %esp
    call Main
    halt

findMin:   #int* findMin(int*, int)
    pushl %ebp
    rrmovl %esp, %ebp
    #prologue
    pushl %ebx
    pushl %esi
    #backing up callee save registers

    #grab the arguments
    mrmovl 8(%ebp), %ecx
    mrmovl 12(%ebp), %edx

    #make a guess for the min... a[0]!
    rrmovl %ecx, %eax

    #we generally always need 4 and 1 somewhere
    irmovl 4, %ebx
    irmovl 1, %esi

    #prime the loop
    subl %esi, %edx

    loop:
        addl %ebx, %ecx
        pushl %ecx
```

[1] I made sure to note that it wouldn't be terribly wise to use this procedure in the assignment, merely that I wanted to show my thought process for writing such a procedure. Nonetheless, some students will end up calling findMin...

```
        pushl %eax
        #since eax and ecx will get overwritten, need to back them up!
        mrmovl (%ecx), %ecx
        mrmovl (%eax), %eax
        subl %ecx, %eax #eax - ecx
        popl %eax
        popl %ecx
        cmovg %ecx, %eax
        subl %esi, %edx
        jg loop

    popl %esi
    popl %ebx
    popl %ebp
    ret

Main:
    pushl %ebp
    rrmovl %esp, %ebp

    irmovl 5, %eax
    pushl %eax
    #remember, the first argument we push will be the last argument
    #  from the perspective of the callee
    irmovl array, %eax
    pushl %eax
    call findMin

    rrmovl %ebp, %esp #clean up the stack
    popl %ebp
    ret


.pos 0x104 #out of the reach of our stack
array:
    .long 0x4
    .long 0x1
    .long 0x3
    .long 0x5
    .long 0x4

.pos 0x100
Stack:
```
