Computer Architecture Project 3: R-ASCII

Jon Durbin

How to run

In ubuntu, I can type the following to simulate all 4 programs (the -nc flag is necessary for ubuntu, but apparently not for windows - unsure why):

```
$ scala -nc ram_files/bin-sort.ram 1
$ scala -nc ram_files/bin-sort.ram 2
$ scala -nc ram_files/quicksort.ram 1
$ scala -nc ram_files/quicksort.ram 2
$ scala -nc ram_files/quicksort.ram 4
$ scala -nc ram_files/adventure-io.ram 3; 0; *CRTL+C*
$ scala -nc ram_files/rule-io.ram 3; 10
$ scala -nc ram_files/rule-io.ram 4; 10
```

Output

The output of running each of the lines above is shown here:

```
*output*
```

Reflection

Computer architecture is hard, but this project greatly helped cement the ideas we went over in class in my head. We certainly made mane simplifying assumptions while defining and working on this project, but I still think I gained a lot from taking abstract ideas out of my head and writing them down in a program. In particular, I have a much better understanding of the control flow of a program's execution. Splitting the process into two steps - loading the program into memory, then executing the instructions - definitely switched on a few light bulbs in my head.

An emulator like this would be good (best?) for tracing programs. Because it is only a simulation, it's going to run slower, so one wouldn't want to use it for anything that's time-sensitive. A user could potentially generate a custom trace using a program like what we created. Users could also add to this program, specifying costs associated with accessing memory or ram to better understand where potential bottlenecks in their program are.

Personal note:

I compiled this document to pdf using:

```
pandoc writeup.md --pdf-engine=xelatex -o writeup.pdf
```