CPSC 313: Computer Hardware and Operating Systems

Assignment #1, due Saturday September 22 at 11:59PM. Late penalty of 33.333% per day for up to 3 days.

1 Objectives

This assignment is to be done alone.

After completing this assignment, you should be able to

- read programs written using the x86-64 ISA, and explain what each instruction does,
- identify the correspondences between assembly and C code fragments,
- translate programs written in C or x86-64 assembly language into Y86 assembly language.

2 Introduction

In CPSC 213, you learned how a C compiler translates constructs into assembly language using SM213, a fictional ISA developed specifically for that course. The goals of this lab are to allow you to refresh your understanding of assembly language, deepen your knowledge of the IA32/x86-64 instruction set architectures (which you may have touched upon in CPSC 213, but without going into any depth), and help you become more familiar with the Y86 subset of the IA32 ISA that we will be using in class for the first part of the course, and the next several assignments.

Code for this assignment is provided in `code.zip` available in Piazza as part of the posting announcing this assignment.

3 Understanding x86-64 Assembly language

The procedure below, stored in file `heapsort.c`, implements the `heapsort` algorithm. Use the command `gcc -O2 -S heapsort.c` (do not forget any of the arguments) to get the gcc compiler to produce assembly code for this procedure. You must use a machine running a 64-bit linux distribution, such as `remote.ugrad.cs.ubc.ca`, or any one of the machines in room ICCS 005.

```
void heapsort(int last) {
    int i;
    if (last < 0) return;
    heapify_array(last);
    for (i = last; i >= 0; i--) {
        heap[i] = extract_max(i);
    }
}
```

Now, comment every line in the assembly code to explain what it does. Be sure to indicate the correspondence between C and Assembly code, i.e., which assembly instructions implement which lines of the C program. Ignore the setup and the tear-down code, that is, only comment the lines after

```
.cfi_startproc
```

and before

```
.cfi_endproc
```

and ignore all assembler directives such as `.cfi_def_cfa_offset` or `.p2align`.

A word of caution if you are using your own machine. Different compiler/OS combinations can result in different code being produced by the compiler. Even a slight change between compiler versions can result in different code. As a result, for consistency and ease of marking, the output being commented, and hence marked, must match the output as produced by the compilers on the department's undergraduate Linux machines.

4 Setting up the simulator

Instructions for installing the simulator form part of the information posted with the announcement of this assignment. Pay particular attention to the instructions at the end of the handout that describe how to run the reference simulators from the command line.

5 Writing Y86 Assembly language programs

In the next several assignments, you will write Y86 assembly language programs to test some modifications that you will be asked to make to the simulator. As a warm-up exercise for these assignments, rewrite the code from the file `heapsort.c` into Y86 assembly language. Start from the file `heapsort.s` you worked with in Section 3, and translate it more or less line by line. If you prefer, you can write the Y86 program from scratch.

In either case your function must assume that the value of `last` is provided in register `%eax`. Similarly, when you call `heapify_array` and `extract_max`, you should pass the argument through register `%eax`, and the value returned by function `extract_max` will be in `%eax` when it returns. Your function is allowed to modify registers `%eax`, `%ecx`, `%edx` and `%ebx` **only**. All other registers **must** have the same value when the function returns that they did when it was called (you can of course use these registers as long as you save them at the beginning of the function, and restore them to the old values at the end).

Do not forget that the x^{th} element of array `heap` is at address $\textit{address}(\texttt{heap}) + 4x$, not at address $\textit{address}(\texttt{heap}) + x$. Your code **must** be well-commented. Source code for the contents of file `heapsort-main.c` in file `heapsort-student.s` have been provided. Test your solution by inserting your code for `heapsort` at the location indicated by

```
###
### THIS PART TO BE COMPLETED BY THE STUDENT.
###
```

in file `heapsort-student.s`, and then running it through the simulator (use the reference simlator SimpleMachine313Seq.jar) to verify that it sorts the array `heap` correctly.

6 Deliverables

You should use the `handin` program to submit your assignment. Create a subdirectory ~`/cs313` in your home directory, and then create the directory ~`/cs313/a1`. You should place the following files (and no others) in your `a1` directory:

1. Your commented `heapsort.s` file.
2. A copy of the file `heapsort-student.s` that includes your Y86 implementation of the `heapsort` function (with comments!).
3. A text file `a1-info.txt`[1] that contains your name, your student number, and how long it took you to do the assignment. Only report the time you actually spent, not the time between when you started and finished. So "3 days" is not an acceptable answer unless you actually worked on the assignment for 72 hours.

That is, the structure of the directory you submit should be the following:

```
poirot> ls a1
a1-info.txt   heapsort.s  heapsort-student.s
```

Once the three files listed above are in a directory called `cs313/a1`, run the command `handin cs313 a1`. You can submit your assignment as often as you like up until 3 days after the due date. Handin will warn you if you handing in an assignment after the due date. Such assignments will be marked as late and penalized as indicated at the start of this assignment description.

The marking scheme will be broken down roughly as follows: 10 marks for commenting the `heapsort.s` file generated by the `gcc` compiler, 14 marks for the Y86 version of the `heapsort` function, and 1 mark for telling me how long you spent to complete the assignment.

[1]Not a Word document, PDF, HTML, or RTF document or other format that is not easily readable by a human!