CS2110 Fall 2014
Homework 6

Author: Thanh Ky Quan

This assignment is due by:

Day: Tuesday September 30th
Time: 11:54:59pm

Rules and Regulations

Academic Misconduct
Academic misconduct is taken very seriously in this class. Homework assignments are collaborative. However, each of these assignments should be coded by you and only you. This means you may not copy code from your peers, someone who has already taken this course, or from the Internet. You may work with others **who are enrolled in the course,** but each student should be turning in their own version of the assignment. We will be using automated code analysis and comparison tools to enforce these rules. **If you are caught you will receive a zero and will be reported to Dean of Students.**

Submission Guidelines
1. You are responsible for turning in assignments on time. This includes allowing for unforeseen circumstances. If you have an emergency let us know ***IN ADVANCE*** of the due time supplying documentation (i.e. note from the dean, doctor's note, etc). Extensions will only be granted to those who contact us in advance of the deadline and no extensions will be made after the due date.

2. You are also responsible for ensuring that what you turned in is what you meant to turn in. No excuses, what you turn in is what we grade. In addition, your assignment must be turned in via T-Square. When you submit the assignment you should get an email from T-Square telling you that you submitted the assignment. If you do not get this email that means that you did not complete the submission process correctly. Under no circumstances whatsoever we will accept any email submission of an assignment. Note: if you were granted an extension you will still turn in the assignment over T-Square.

3. There is a random grace period added to all assignments and the TA who posts the assignment determines it. The grace period will last **at least one hour** and may be up to 6 hours and can end on a 5 minute interval; therefore, you are guaranteed to be able to submit your assignment before 12:55AM and you may have up to 5:55AM. As stated it can end on a 5 minute interval so valid ending times are 1AM, 1:05AM, 1:10AM, etc. **Do not ask us what the grace period is we will not tell you**. *So what you should take from this is not to start assignments on the last day and depend on this grace period past 12:55AM.* There is no late penalty for submitting within the grace period. If you can not submit your assignment on T-Square due to the grace period ending then you will receive a zero, no exceptions.

General Rules

1. Any code you write (if any) must be clearly commented and the comments must be meaningful. You should comment your code in terms of the algorithm you are implementing we all know what the line of code does.
2. Although you may ask TAs for clarification, you are ultimately responsible for what you submit.
3. Please read the assignment in its entirety before asking questions.
4. Please start assignments early, and ask for help early. Do not email us the night the assignment is due with questions.
5. If you find any problems with the assignment it would be greatly appreciated if you reported them to the author (which can be found at the top of the assignment). Announcements will be posted if the assignment changes.

Submission Conventions

1. Failure to follow these may result in a max of 5 points taken off

2. All files you submit for assignments in this course should have your name at the top of the file as a comment for any source code file, and somewhere in the file, near the top, for other files unless otherwise noted.
3. When preparing your submission you may either submit the files individually to T-Square or you may submit an archive (zip or tar.gz only please) of the files (preferred). You can create an archive by right clicking on files and selecting the appropriate compress option in on your system.
4. If you choose to submit an archive please don't zip up a folder with the files, only submit an archive of the files we want. (See Deliverables).
5. Do not submit compiled files that is .class files for Java code and .o files for C code. Only submit the files we ask for in the assignment.
6. Do not submit links to files. We will not grade assignments submitted this way as it is easy to change the files after the submission period ends.

Overview

For this assignment, you will be writing 3 assembly programs. The purpose of this assignment is to get you familiar with and more comfortable with writing low-level assembly code. The tools you need to do this assignment are on T-Square under Resources > Assembly. Once you get there, read Complx tools installation instructions.htm.

This will tell you how to install the tools for the class.

Note that you may only use Complx to simulate your LC-3 code.

For each part of the assignment, you will write an assembly program that implements the given C (or Java) code. Haven't seen C before? It's okay, the syntax is very similar to Java, so you should be able to figure it out. I'll explain the bit in the comments you need for each problem, but we'll go over it more once we get to that part of class. Think of this as a little taste.

How to write an assembly program

You've been given boiler plates for this assignment, but I'll tell you little bit about how the assembly files in this class look.

```
.orig x3000
LEA R0, HW ;Load the address of the string
PUTS ;Output the string
HALT ;Stop running
HW
.stringz “Hello world.\n”
.end
```

This is a small program that will output “Hello world.” followed by a new-line.

The .orig is a pseudo-op. Pseudo-ops are special instructions for the assembler that are not actually assembly. .orig x3000 tells the assembler to place this block of code at x
3000, which is where the LC3
starts code execution from.
Next is your assembly program. You've seen this before. PUTS is just a pseudonym for a trap that prints a string whose address is stored in R0 just like HALT is a trap that stops the LC3.

.stringz is a pseudo-op that stores the following string at that set of memory locations, followed by a zero. (That’s what the z is for.) For example, 'H' is stored at x3003, 'e' is stored at x3004, etc.
.end tells the assembler where to stop reading code for the current block. Every .orig statement must have a corresponding .end statement.

A couple other pseudo-ops are

- .fill <value> will put the value at that memory location.
- .blkw <num> will reserve the next num locations, filling them with 0.

Abstract

For this assignment you will be working with memory and state machines. This is a 3-part assignment,

In Part 1, you will implement division in LC3.

In Part 2, I will give you a positive number, call it k. Your task is to find the smallest prime number that greater than k.

In Part 3, you will be given an array of size n. Your job is to sort in a descending order.

Proper comments are required. Please comment in such a way that you can understand what does it mean when you look at it a year later!

Part 1

So I have asked George P. Burdell if he would like to suggest any assignment for our awesome CS2110 students, he told me: “Let's them do division!”. So yep, this is your first task to implement.

You will be given two **non-negative** numbers A, B. All you have to do is to implement an integer division A divides B. If B is zero, then you just need to return -1 for quotient and 0 for remainder. Your result of division must be stored in quotient label and remainder in remainder label.

Example in C:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
	int a = 87;
	int b = 10;
	int remainder = 0;
	int quotient = 0;
	if (b == 0) {
		quotient = -1;
	}
	else {
		while (a >= b) {
			quotient++;
			a -= b;
		}
		remainder = a;
	}
	printf("Quotient: %d\n", quotient);
	printf("Remainder: %d\n", remainder);
}
```

Part 2

You will be given a **positive** number K, i.e. 5. Your job is to find the smallest prime number that is greater than K, i.e. 7. Your answer must be stored in nextprime label.

Example in C:

```
#include <stdio.h>

int main() {
	int n = 99	;
	int found = 0;
	int i;
	int k = n;
	while (!found) {
		k++;
		int isPrime = 1;
		for (i=2; i<k/2 && isPrime; i++) {
			isPrime = k%i;
		}
		found = isPrime;
	}
	int nextPrime = k;
	printf("%d\n", nextPrime);
}
```

Part 3

You will be given **non-negative** number N, and an array of size N. Your task is to sort it in place and in descending order.

For example:
N = 5
Array: 4 10 2 3 5
Answer: 10 5 4 3 2

Example in C:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
	int n = 8;
	int a[8] = {1,2,3,4,5,100,7,40};
	int i,j,temp;
	for (i=0; i<n-1; i++) {
		for (j=i+1; j<n; j++) {
			if (a[i] < a[j]) {
				temp = a[i];
				a[i] = a[j];
				a[j] = temp;
			}
		}
	}
	for (i=0; i<n; i++) {
		printf("%d ", a[i]);
	}
	printf("\n");
}
```

Deliverables

Part 1
Your modified version of division.asm

Part 2
Your modified version of nextprime.asm

Part 3
Your modified version of sort.asm