Review Problem 1

- Programming languages have many instructions, but they fall under a few basic types. One is arithmetic (+, -, *, /, etc). What are the others?

  logical (and, or, etc)
  shift
  variables, structures, classes
  iteration: for, while, etc.
  control flow: if, goto, procedure call
  I/O: print, files, ...
  storage allocation: malloc, new
Assembly Language

Readings: 2.1-2.7, 2.9-2.10, 2.14
Green reference card

Assembly language
Simple, regular instructions – building blocks of C, Java & other languages
Typically one-to-one mapping to machine language

Our goal
Understand the basics of assembly language
Help figure out what the processor needs to be able to do

Not our goal to teach complete assembly/machine language programming
Floating point
Procedure calls
Stacks & local variables
Aside: C/C++ Primer

```c
struct coord { int x, y; }; /* Declares a type */
struct coord start;       /* Object with two slots, x and y */
start.x = 1;               /* For objects "." accesses a slot */
struct coord *myLoc;       /* "*" is a pointer to objects */
myLoc = &start;            /* "&" returns thing's location */
myLoc->y = 2;              /* "->" is "*" plus "." */
```

```
int scores[8];             /* 8 ints, from 0..7 */
scores[1]=5;                /* Access locations in array */
int *index = scores;       /* Points to scores[0] */
index++;                    /* Next scores location */
(*index)++;                 /* "*" works in arays as well */
index = &(scores[3]);       /* Points to scores[3] */
*index = 9;
```

```
Scores: [0 1 2 3 4 5 6 7]
index: [1]
```
ARM Assembly Language

The basic instructions have four components:
  Operator name
  Destination
  1st operand
  2nd operand

ADD <dst>, <src1>, <src2> // <dst> = <src1> + <src2>
SUB <dst>, <src1>, <src2> // <dst> = <src1> - <src2>

Simple format: easy to implement in hardware

More complex: A = B + C + D - E

    ADD t1, B, C
    SUB t2, D, E
    ADD A, t1, t2
Operands & Storage

For speed, CPU has 32 general-purpose registers for storing most operands
For capacity, computer has large memory (multi-GB)

Load/store operation moves information between registers and main memory
All other operations work on registers
# Registers

32x 64-bit registers for operands

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0-X7</td>
<td>Function arguments/Results</td>
<td></td>
</tr>
<tr>
<td>X8</td>
<td>Result, if a pointer</td>
<td></td>
</tr>
<tr>
<td>X9-X15</td>
<td>Volatile Temporaries</td>
<td>Not saved on call</td>
</tr>
<tr>
<td>X16-X17</td>
<td>Linker scratch registers</td>
<td>Don't use them</td>
</tr>
<tr>
<td>X18</td>
<td>Platform register</td>
<td>Don't use this</td>
</tr>
<tr>
<td>X19-X27</td>
<td>Temporaries (saved across calls)</td>
<td>Saved on call</td>
</tr>
<tr>
<td>X28</td>
<td>Stack Pointer</td>
<td></td>
</tr>
<tr>
<td>X29</td>
<td>Frame Pointer</td>
<td></td>
</tr>
<tr>
<td>X30</td>
<td>Return Address</td>
<td></td>
</tr>
<tr>
<td>X31</td>
<td>Always 0</td>
<td>No-op on write</td>
</tr>
</tbody>
</table>

Lab #1 Due Oct 11th