ECE220: Computer Systems & Programming **Fall 2020 ZJUI**
Lab 12 **Tuesday 8 December**

Basic Blocks for LC-3 Programs

In today's lab, you must build basic blocks from a set of LC-3 instructions. A basic block is a sequence of instructions that are always executed in order, allowing a compiler to perform optimization and reordering amongst the instructions. In particular, only the first instruction in a basic block can be targeted by any branch or JSR instructions—other instructions must be reached by executing the first instruction. And only the last instruction can have multiple possible next instructions—instructions within the block (except for the last) are always followed by other instructions within the block. The goal is for you to gain more experience with dynamic allocation and dynamic resizing, both of which are needed for MP10.

Begin by checking out the **`lab12`** subdirectory in your Subversion repository. The directory contains a copy of this document (**`lab12.pdf`**), a C header file **`lab12.h`**, a C source file **`lab12main.c`** that provides most of the code, a **`Makefile`**, and a C source file, **`lab12.c`**, with which you can start this lab. Finally, the subdirectory **`samples`** includes three sample LC-3 object files, and the subdirectory **`outputs`** contains the correct outputs for those files. You can immediately "**`make`**" the executable, but without your code, the analysis of LC-3 programs will neither be correct nor particularly useful.

Next, copy one of your group's **`lab11.c`** solutions from last week and edit the file to include **`lab12.h`** instead of **`lab11.h`**. The solution need not be perfect, but it is needed to fully explore subroutines, and also to match the sample outputs.

What the Given Code is Doing

The code given to you is nearly the same as last week, but a few new flags have been added, a new field has been added to each instruction (to point to the basic block that contains the instruction), and a new structure has been added for basic blocks. The only additional activity by the main program is to call the code that you must write in **`lab12.c`** and then print the results.

The Task

You must implement the function **`build_blocks`**, in **`lab12.c`**. The function is invoked for the main LC-3 program and on every subroutine reachable from the main program. Note that an LC-3 program may appear to have subroutines that are not actually subroutines, since data values may appear to be JSR instructions.

The signature for your function is as follows:

```
basic_block_t* build_blocks (code_stat_t* cs, int32_t first,
                             uint16_t block_start);
```

Your function will be called once for the main program and once for each subroutine. Use recursion to dynamically allocate and fill in basic blocks for the whole main program or for the whole subroutine.

The **`cs`** parameter provides information on the code, as discussed previously. The **`first`** parameter indicates the starting point for the basic block to be built, and is an array index for the array of instructions in the **`cs`**. The **`block_start`** parameter is the address of the first instruction of the code segment, and is used recursively to mark all blocks belonging to the main program or to a subroutine with the starting address of that part of the code.

The function must first check for an existing basic block, and otherwise allocate one and find the end by checking instructions until an appropriate ending point is found, then recursively finding the next block(s) for the basic block just built. Basic blocks are allocated dynamically, and the array of instruction pointers in each block uses dynamic resizing. Read the changes to the header file and the comments in **`lab12.c`** for more specific directions as to how your function must operate.