CS270 Recitation 9
“Working with pointers in LC-3”

Goals

Work with pointers in assembly language where you can watch how the values are used.

The recitation:

Download http://www.cs.colostate.edu/~cs270/.Fall13/recitations/R9/LINKED_LIST.asm

You will see a large number of .FILL directives toward the top of this file. This lays out a linked list in memory. Each element of our list consists of two adjacent words of memory. The first word contains the ASCII code for a character. The second contains the address of the next element of our list (except the last element of our list which is just 0x0).

Your job is to write some assembly code to print the sequence of characters to the screen. There is a .FILL directive labeled HEAD that points to the first element of our list.

Up till now, we’ve only used registers to store data values of interest, such as bit patterns and operands for arithmetic. This recitation will involve loading addresses of other words of memory into registers. When a register has the address of something in memory, we can say it contains a pointer to that memory.

Step 1: Figure out which LC-3 instruction will load a word from memory into one register, using an address which is in a register. You might want to write a small test program to see if you can successfully load the first element of the linked list into registers. (i.e. The first word (the ASCII character value) into one register and the second word (the address of the next element) into another).

Step 2: Use this to write a loop that repeatedly does what your Step1 test program did until it encounters a null (0x0) value where the pointer to the next element would normally go (we call that a null pointer). Load the character value into R0, and use the “OUT” macro to print it. OUT already expects R0 to be in ASCII, so you don’t need to do any conversion. The memory labeled HEAD is not really any different from the second field of our list elements, so you should be able to have a single check for the null pointer that works for both.

When you get it to print “Helo World!”, show it to your instructor. The spelling error is deliberate.

Step 3 (worth one recitation of extra credit): Add/modify the .fill values at the top of the file to fix the spelling error.

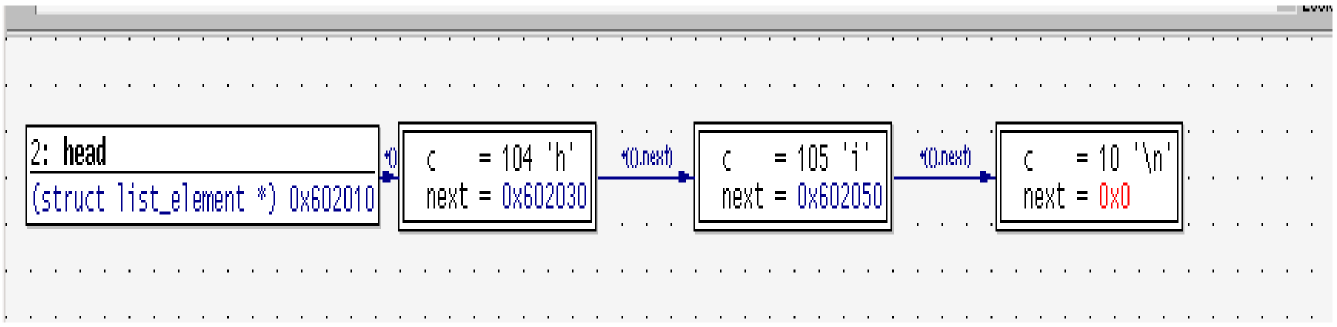
For convenience, here are the commands you’ll need:

```
~cs270/lc3tools/lc3as LINKED_LIST.asm
```

```
~cs270/lc3tools/lc3sim-tk
```

After you have this working, you might find it interesting to look at some C code that does the same thing (and a useful example to refer back to). You can find this at http://www.cs.colostate.edu/~cs270/.Fall13/recitations/R9/ll.c

Here is a screenshot of using `ddd` to visualize the C data structure in memory:



```
2: head
(struct list_element *) 0x602010
```

```
c    = 104 'h'
next = 0x602030
```

```
c    = 105 'i'
next = 0x602050
```

```
c    = 10 '\n'
next = 0x0
```