G51CSA Lab Exercise One: Converting the G51PRG drinks machine into ARM assembler

Steven R. Bagley

Introduction

The file `drinks.c` on the G51CSA website contains the example drinks vending machine menu that we saw in G51PRG. The idea of this exercise is to convert it into an ARM assembler program. A start was made in the lecture and can be found in `drink.s`.

This starting point only prints the message if `1` is pressed on the keyboard (read in using `SWI 1`), your task is to extend it to print each of the different messages based on the keyboard using the assembler equivalent of an `if…else` block. I strongly advise you to build this up bit by bit, if you attempt to write it all in one go then you will almost certainly get tangled up in spaghetti.

Therefore, I suggest the following approach. Firstly, insert the `else` clause block back into the `if` statement as shown in the lecture slides and check that that works (i.e. if you press `1` you get the message '`Have a bottle of coke`' and if you press anything else you get the error message.

Once that works, I□–¯d then introduce assembler equivalent for the second `if` statement in the program — this will start at the label you branch to for the first `if` equivalent's else clause, and the first if will now branch over the whole of the second `if`. Once that works, you should be able to do the same again to add the third `if`…

Tips

- The labels you need to make the if in assembler must be unique for each if statement, you cannot use the same label name twice
- The labels for a specific if in ASM will align with the same points in the C if every time. Specifically, you need a label aligned with the start of the else block (if there is one) and a label aligned with the last } in the C if statement (whether that is attached to the else block or the if itself)
- Test it often, and ask yourself is it doing what you expected before carrying on. If it isn't fix it so it does.
- You will need to use `DEFB` to define the strings for each message (don't forget the `\0` terminator!), at the start of the file.
- Use sensible label names so you know what's what '`cokeskip`' is a lot more helpful to you than `skip1`, `skip2` etc…
- If you need to remind yourself of how Komodo/`aasm` work, then take a look in the 'Getting started with UNIX/Linux for G51PRG and G51CSA' guide on the website.

Extension

Why not wrap the whole thing up in a loop that only quits if the user selects `0`.