COL216 Computer Architecture

Lab Assignment 3 : Expression Evaluation with I/O

The objective of this exercise is to augment the program of assignment 2 with convenient input/output. There are three learning goals here –

- Understand how input/output is performed at assembly level
- Learn how to use predefined functions in assembly
- Learn how to work with assembly code spread over multiple files

Remove the main program (the code which makes the top level call to expression evaluation function) from the assignment 2 code and create a new main in a separate file. This new main is required to loop through the following sequence.

1. input an expression from `stdin` (standard input device)
2. call the evaluation function of assignment 2
3. output the result on `stdout` (standard output device)

The loop should terminate when an empty expression is read from `stdin`. The keyboard, together with a window displayed by the simulator, represent the `stdin` and `stdout`.

How to perform input/output?

System calls (`swi` instruction) for I/O discussed in class are for ARMSim# version 1.91. ARMSim# version 2.1 installed in the lab has a different I/O mechanism. It supports a single call `swi #0x123456` (they have termed it "Angel SWI instruction") and a parameter passed through register `r0` defines the specific I/O or other operation to be performed. The file "AngelInstructions.pdf" (uploaded on moodle) gives the details. On the whole, the new I/O mechanism is more versatile. For example, random access files are supported. However, the operations performed by individual calls are more primitive. For example, reading or writing of a string or an integer can't be done by one `swi` instruction, but requires some non-trivial code. For the purpose of this assignment, you can use some predefined I/O functions described in the next section. You will still need to use `swi` instruction for exiting the program in the following manner.

```
mov   r0, #0x18
mov   r1, #0
swi   0x123456
```

Predefined I/O functions

The file "UsefulFunctions.s" supplied by ARMSim# developers has been uploaded on moodle. It contains the following functions that will fulfill the needs of this assignment.

- `prints` : print a null-terminated ASCII string to standard output
- `fgets` : read one line from a text file or standard input
- `itoa` : convert an integer to a null-terminated ASCII string

Use `fgets` to input an expression string from keyboard. The result of expression evaluation needs to be converted into a string using `itoa` and then displayed using `prints`. Also, to make your program more friendly, use `prints` to display some prompt messages such as – "Enter an expression to be evaluated", "Expression evaluates to ", "Exiting the program now", etc.

One tricky question is – how do you terminate a string during input? Pressing enter or return key is likely to work. The `fgets` function actually looks for '\n' character (ASCII code 10) to identify end of a string. However, depending upon the system, the enter/return key may give out '\r', '\n' character sequence or just '\r' character (ASCII code 13). In such a case, press CTRL J at the end of the string, instead of enter/return.

In order to use the above I/O functions, you don't need to understand their detailed implementations. It is enough to know the definition of the parameters, clearly given in the file "UsefulFunctions.s". However, some effort to understand how these are built over the primitive operations, will give you a good insight.

Working with multiple assembly source files

It is a good practice to keep independent pieces of code in separate files. ARMSim# has a provision to specify multiple source files at the time of loading. All you need to do is to define appropriate linkages among the files. This is done using two assembler directives - `.global` and `.extern`, which have the following syntax and meaning.

> `.global` *comma separated list of symbols*
>
> This declares the symbols defined in the current file and used in some other file(s).
>
> `.extern` *comma separated list of symbols*
>
> This declares the symbols used in the current file and defined in some other file.

In the present assignment, you will have three files – (i) "UsefulFunctions.s", (ii) your expression evaluation code of assignment 2 (minus the main) and (iii) your new main program with I/O. The first one already has the required `.global` statements. In the second file, include a `.global` statement mentioning the top level function, that will be called by main. The third one will require an `.extern` statement listing all the functions to be called from the previous two files.