Lab 7: Calling Convention - C to Assembly and Assembly to C

Overview

In previous labs you have come across the way to call existing c functions from assembly such as printf and scanf. To learn more about calling conventions from C to assembly and vice versa, it is better to write your own C and assembly functions and call one from another language.
In this lab, you will develop two projects(each will have two source files, one in c and another in assembly). The task of both the projects is to extract a specific sub-string from a given string. However, the difference between the two projects is the way they are implmented.
- In project 1, main code is written in assembly (main.s) and sub-routine to extract the sub-string is written in C (sub_string.c)
- In project 2, main code is written in C (main.c) and sub-routine to extract the sub-string is written in assembly (sub_string.s)

You will use ARM assembly for this lab and hence use raspberry pi to run the programs.

Details

1. Main code does the following tasks:

1. Reads a string from the user Ex: “publication”
2. Reads the start index and end index as integers. Ex: start index = 3, end index = 7.
3. Calls the function or sub-routine “sub_string” and pass the above three inputs as arguments
4. Prints the returned sub-string from the function. Ex: “blica”. This is the sub-string starting from character 3 and ending at 7.

2. Sub-routine code does the following tasks:

1. Reads the three arguments from the caller (main)
2. Extracts the sub-string from the given string
3. Returns the sub-string to the caller (main)

In project 1, the sub-routine in c may look like this:

```
char* sub_string(char *in_string, int start_index, int end_index)
{
     char *out_string;
     /* code to extract the sub-string */
     return out_string
}
```

The main.s code will call the above sub-routine in the following way:

```
/* Data declarations as necessary */
/* Code to receive inputs from user */
```

```
  bl sub_string
  */ Code to print the sub-string */
```

In project 2, the sub-routine in asm may look like this:

```
    /* assembly declarations and code as required */
    sub_string:
     /* Code to extract the sub-string */
     /* Suitable code to return the sub-string */
```

The main.c code will call the above sub-routine in the following way:

```
  int main()
  {
       int start_index, end_index;
       char *in_string;
       char *out_string;
       /* Code to receive inputs from user */
       out_string = sub_string(in_string, start_index, end_index);
       */ Code to print the sub-string */
       return 0
  }
```

Your task is to use these as templates and fill up the required code.

Compiling instructions

For project 1, you will have two files : “main.s” and “sub_string.c”. Use the below command:

```
                              gcc -o asm2c main.s sub_string.c
```

For project 2, you will have two files : “main.c” and “sub_string.s”. Use the below command:

```
                              gcc -o c2asm main.c sub_string.s
```

You are required to use the above given file names for consistency among the class.

Instructions to execute

To run project 1, execute the “asm2c” executable as `./asm2c` To run project 2, execute the “c2asm” executable as `./c2asm`

The output for both the projects shall be similar to the one given below:

```
Enter a string: Publication
Enter the start index: 3
Enter the end index: 7
The substring of the given string is 'blica'
```

Again, you are required to maintain the above given output format.

Turnin

Credit for this lab will be given based on functionality of your program, useful comments on every line, simplicity/organization of your code and regularity in the filenames and the output format shown above.
Follow these instructions to turnin lab7.

1. Your four program files need to be named exactly as written above: `main.s`, `sub_string.c`, and `main.c`, `sub_string.s`.
2. Put all four of these files in a folder named `lab7-src`.
3. Submit this folder electronically by following these instructions. (You have to be ssh'd into data.cs.purdue.edu first, the "turnin" command is only on those machines)

```
$ turnin -c cs250 -p lab7 lab7-src
$ turnin -c cs250 -p lab7 -v
```

The second command verifies that you have successfully submitted your files.

References

Here are some good references that can help you in your programming

- ARM Assembly Tutorial 1
- ARM Assembly Tutorial 2 (Raspberry Pi specific!)[1)]
- ARM Assembly Language Reference Sheet *OR* http://ozark.hendrix.edu/~burch/cs/230/arm-ref.pdf
- ARM Assembly Language Slides
- http://www.pp4s.co.uk/main/tu-trans-asm-arm.html
- Procedure Call Standard for the ARM Architecture

[1)] Thanks to Joosep Jaagosild for finding this tutorial.