

# CSE30 - HW #4

Assembly and Hand Compilation!

Due Friday, March 14th, 6:00 pm

## 1 Introduction

This assignment will give you a lot more practice with assembly programming. You will find all the code required for this assignment in the VM present in this [public location](#).

## 2 Getting started

This homework will be completely done in a Raspberry Pi (Raspbian on ARM) virtual machine. Following are the steps to get the setup working :

1. Install VMWare Player on your local machine. [Download Link](#)
2. Download the testbed setup from the public location mentioned above. Extract the downloaded tarball on your local machine.
3. Open VMware Player → Choose "Open Virtual Machine" option.
4. Choose CSE30.vmx file from the directory you extracted in step 3. The testbed VM will start booting now.
5. Once the VM boots, log in using password cse30.
6. Open up Accessories → Terminal Emulator and change directory to `~/Desktop/HW4`.
7. Execute `./run_emulator`. This will open the Qemu installation with Raspbian running in it. This process can take a while to complete.

8. Once the Raspian environment is up, open up LxTerminal and start coding! (`~/Desktop/CSE30` contains all the code for this homework)
9. Use the keyboard shortcut `Ctrl+Alt+G` to use the mouse correctly inside Qemu.

### 3 Thinking in Assembly

The function `atoi()` is used to convert a string of digits to integers (e.g., "25" to 25). This function takes a string, and converts it to an integer. Your task for this part of the assignment is to rewrite `atoi()` as the function `str_to_int()`. The function must be written entirely in ARM assembly.

We've provided the files "`str_to_int_ARM.s`" and "`str_to_int_test.c`" for you. Please write your function in the `.s` file, and test it thoroughly in the `.c` file.

We've provided you with a function signature. Please do not change it.

### 4 Hand Compilation Fun!

In this section, you are given a problem, and you are expected to solve it iteratively and recursively in assembly. For full credit, your ARM functions *must* use the same recursive implementation as the provided C functions do.

#### 4.1 Recursion in ARM

Implement the `print100()` function shown below in C and ARM assembly.

This source code should be written in `print100_recur.c` and `print100_iter.c`.

Please put your ARM implementation in `print100_recur_ARM.s` and `print100_iter_ARM.s`, and your testing code in `print100_test.c`.

Listing 1: `print100()`

```
1 /* int print100_recursive(int n)
2 *   Args:
3 *       int n: Clue - you will need this argument
4 *   Returns:
5 *       void:
6 */
7 void print100_recursive(int n)
```

```

8 | {
9 |     // TODO: IMPLEMENT ME
10 | }
11 |
12 | /* int print100_iterative()
13 | *   Args:
14 | *       No arguments
15 | *   Returns:
16 | *       void:
17 | */
18 | void print100_iterative()
19 | {
20 |     // TODO: IMPLEMENT ME
21 | }

```

## 4.2 Binary Search in ARM

Implement the `binary_search()` function below in ARM assembly. Your solution **MUST** find the index of the desired element in the sorted array recursively, as described below. This source code is provided in `binary_search.c`. Please put your implementation in `binary_search_ARM.s`, and your testing code in `binary_search_test.c`.

Listing 2: `binary_search()`

```

1 | /* int binary_search(int *data, int toFind,
2 | *                   int start, int end)
3 | *   Args:
4 | *       int *data: address of element 0 of an integer
5 | *                   array sorted in ascending order.
6 | *       int toFind: integer to find index of
7 | *       int start: index of start of search area
8 | *       int end: index of end of search area
9 | *   Returns:
10 | *       int: index of element in array
11 | *           OR
12 | *       -1 if element not in array
13 | */
14 | int binary_search(int *data, int toFind,
15 |                 int start, int end)
16 | {
17 |     int mid = start + (end-start)/2;
18 |     if (start > end)
19 |         return -1;

```

```
20     else if (data[mid] == toFind)
21         return (mid);
22     else if (data[mid] > toFind)
23         return binary_search(data, toFind, start, mid-1);
24     else
25         return binary_search(data, toFind, mid+1, end);
26 }
```

## 5 Submission

As soon as you're finished with the assignment, you may use the `turnin` command to submit your work. The `turnin` command accepts a single file as argument. First, compress the folder into a `tar.gz` file using the command `tar -czf hw4.tar.gz hw4` where `hw4` is the name of your homework directory. Next, execute `turnin -p hw4 hw4.tar.gz` to submit your homework.

You may submit your homework as many times as you'd like, but only the final submission will be recorded. **Make sure that all files needed for the assignment are in the directory given to turnin.** No other files will be submitted.