CS-382 Computer Architecture and Organization Fall 2022

Homework 2 · ARMv8 Assembly (Part I)

Lecturer: Shudong Hao *Date: See Canvas*

All update on the homework will be announced on Canvas. Please make sure to check it out often!

1 Task 1 (20 pts): Copy String

In this task, you will write an assembly code that completes the same task as in previous homework, *i.e.*, copy string `src_str` to another string `dst_str`. The `.data` and `.bss` segments are declared as follows, and you can just use them in your code:

```
.data
src_str:  .string  "I love 382 and assembly!" // source string

.bss
dst_str:  .skip    100                        // destination string
```

Once you have finished, do not print anything out. Simply start `gdb`, and make a screenshot to show that you have successfully copied the entire string into destination. You can use any screenshot you like, but it has to be persuasive and obvious enough to make the case.

1.1 Requirements

- You can add any data in `.data` segment you like, **except** the length of the string. Do not hard code the string length;
- Do not print anything out using `printf()` or system calls;
- You have to use loops;
- Name this file `strcpy.s`.

2 Task 2 (30 pts): Reversing Nibbles

In this task, you'll write an assembly code to reverse all the nibbles (4 bits) in an integer array. For example, say we have an integer array:

```
arr:    .word  0x12BFDA09, 0x9089CDBA, 0x56788910
length: .word  3
```

The nibbles in this array are:

```
1 2 B F D A 0 9 9 0 8 9 C D B A 5 6 7 8 8 9 1 0
```

After we reverse all of them, we have:

```
0 1 9 8 8 7 6 5 A B D C 9 8 0 9 9 0 A D F B 2 1
```

Then the array should contain the following three words:

```
0x01988765  0xABDC9809  0x90ADFB21
```

Again, in this task, do not print anything out. To verify your result, use `gdb`, and use the following command to print memory out:

```
x/3xw  &arr
```

You can change `arr` as you like and correspondingly `length`. However, the length of the array should **not** be hardcoded in your assembly code (`.text` segment). You can assume it's always an integer array, and the size of an integer **can** be hardcoded.

2.1 Requirements

- Do not hard code array length in your code;
- Do not use `printf()` or system calls to print anything;
- You have to use loops;
- Name this file `reverse.s`.

3 Task 3 (40 pts): Binary Search

In this task, you'll implement a binary search algorithm in ARM assembly. The `.data` segment includes a double word array (sorted), the length of the array, the target value we want to find, and output message:

```
.data
arr:     .quad    -40, -25, -1, 0, 100, 300
length:  .quad    6
```

```
target:  .quad    -25
msg1:    .string  "Target %ld is in the array."
msg2:    .string  "Target %ld is not in the array."
```

Always assume the numbers are signed, and the array is already sorted.

After the search, you need to print the messages correctly with the target value. For example, if the array is declared as above, and `target` is -25, your output should look like:

```
Target -25 is in the array.
```

If `target` is 20, your output should look like:

```
Target 20 is not in the array.
```

You need to make sure the code will exit successfully without any errors after printing out the messages.

3.1 Requirements

- Do not hard code array length in your code;
- You have to use loops;
- Name this file `bins.s`.

4 Report (10 pts)

For the first two tasks, use `gdb` to show that you have completed them. One screenshot for each task is enough, but more is fine.

5 General Requirements

Code that doesn't assemble will receive zero credits – no exception!

- Write your name and honor code pledge at the top of your code as comments;
- Do not use any multiplication and division instructions; you don't need them;
- Avoid using registers `X29` and `X30`;
- You have to put comments on each line. Without comments your code will be penalized heavily;

- Name your assembly code as mentioned. **Wrong file name will zero out your credits for this homework;**
- When in doubt, always ask.

Deliverable

(1) Assembly code for task 1, 2, and 3. Name them properly;
(2) A **PDF** report for the first two tasks. Any report in non-PDF format is not allowed. Note for the screenshots do not take pictures using your phone/camera.

Zip all files and submit on Canvas.