Operating Systems - Monsoon 2020 Take Home Quiz 1 (Total points: 40)

Sambuddho Chakravarty, Arani Bhattacharya

September 17, 2020

Submission deadline: September 19, 2020 @ 2359hrs.

Problem 1: 10 points.

It is "believed" that a program can have only one entry point. Your tasks is test out this hypothesis. You need to write a program in standalone assembly language program that has two functions – a `_start` function (not `main`) where the execution begins, and a second function, lets call it `checkGreater`. The program should begin execution from the `_start` function. The `_start` function should call the `checkGreater` along with two arguments x and y such that the function checks is $x > y$. If yes, then the function should print a string "1", else it should print a string "0" using the `write()`. You could call the `write()` system call using assembly language code snippet. For your convenience you could use the following reference:

`https://jameshfisher.com/2018/03/10/linux-assembly-hello-world/`

Your task is to explore if it is possible to write a regular C program, having a `main()` that should be able to call the `checkGreater` function, like you would normally do in other scenarios where a C program calls a assembly function.

Like previous cases, you need to compile (not link) the programs separately and thereafter link them together in a single binary.

What to submit:

- The source codes of the assembly and the C programs.
- Makefile to compile the programs and link them.
- Description of the programs and if and why they work together, i.e. if `main()` can call `checkGreater` or not, along with an explaination for the same.

Problem 2: 5 points.

Try to compile the following code snippet (no need to link).

```
add(float a, float b)
{
 return (float)(round(a)+round(b));
}
```

Would the compilation of this code snippet result in errors or warnings if any ? Accurately describe your answers and the reason as to why you observe what you observe.

What to submit:

- Command (with appropriate arguments) that were used to compile the snippet.
- A writeup showing all the compilation errors and warnings that you may have encountered and their possible explanations.

Problem 3: 5 points.

The following code snippet is being provided to you:

```
void main()
{
char x[64];
.
.
}
```

You need to complete the above code snippet to firstly store **eight** 64-bit integers using a pointer to type `long int`. Repeat the same exercise to store **sixteen** 32-bit integers using a pointer to type `int`.

What to submit:

- The modified program for the two different scenarios listed above, *i.e.* one where **eight** 64-bit integers, and the other where **sixteen** 32-bit integers are respectively saved in the array.
- A writeup describing the rationale behind your code.

Problem 4: 10 points.

1. Write a function called 'long_add' in x86 (32-bit) assembly, which takes four integers and returns their sum. Then, write a C program that calls this assembly function.
2. Is there any change needed in 'long_add' function if we want to run the assembly function in x86-64 (64-bit)? If change is needed, modify the function and name it 'extended_add'. Justify why such change is needed or is not needed.

What to submit:

- The source codes of the assembly and the C programs (with the names 'P4_1.asm', optionally 'P4_2.asm' and 'P4.c'.
- Makefile to compile the programs.
- A text file named 'A4.txt' with the justification as to why a change is needed or not needed.

Problem 5: 4 points.

Suppose you have an x86-64 processor in an embedded system with only 320KB of RAM. Then, would you ever use the long mode (64-bit mode) or would you prefer to use real mode (16-bit mode) for running your program. Please justify your answer.

What to submit

A text file named 'A5.txt' with the justification.

Problem 6: 6 points.

1. Write a program to store a string containing your name in main memory.
2. Would your program change if you want to store this string on the disk? If so, write the program. If not, justify why the program does not change.

What to submit:

- The C program in a file named 'P6_1.c' for the first part of the question.
- For the second part of the question, name the file as either 'P6_2.c' and 'A6_2.txt' depending on whether you submit a C program or give a justification.

Note: All your files and submissions must be enclosed in a zip file whose file name should be same as your roll number.