CS Basics - Exercises
Introduction to Assembler

E. Benoist C. Grothoff P. Mainini

Fall Term 2022-23

1 Update and Upgrade Your System

Each time you start your Debian GNU/Linux system, you should first update it:

```
$ sudo apt update
$ sudo apt upgrade
```

2 Playing with the Sandbox and a Debugger

The following steps can be done with `gdb`, `ddd` or another front-end of your choice. Refer to the cheat-sheet provided in the course and the documentation found in the man pages or at

- `https://sourceware.org/gdb/onlinedocs/gdb/`
- `https://www.gnu.org/software/ddd/manual/`

To run a program in the debugger, use one of the following commands:

```
$ gdb ./eatsyscall
```

or

```
$ ddd ./eatsyscall &
```

Create a sandbox for playing with assembly language instructions (as seen in the slides). Then, perform the steps described below. For every step, observe the relevant values in the debugger.

1. Store the value 75 to register RAX.
2. Store the value -4 to register RBX.

3. Store the value of register RSP to register RCX.
4. Store the value of register RSP to register R8.
5. Store the value -1 to register EAX (32-bit register)
6. Store the value 10 to register AX (16-bit register)
7. Store the value 35.0 in memory as a doubleword (`dd`). Copy its value to register EDX. Check in the debugger that the value and not the address has been copied.

3 Working with `eatsyscall.asm`

For this exercise, work with a copy of `eatsyscall.asm` from directory `asm-4-programming/examples/eatsyscall/`.

Execute the required assemble and link steps given in the file manually to build an executable.

Then, use the command `touch` to change the modification date of the file and rebuild it using `make`. Which way do you prefer?

3.1 Inspect Memory with `gdb` or `ddd`

1. Place a breakpoint on label "`_start`"
2. Run the program. It should be stopped at the breakpoint by the debugger.
3. Inspect the register values.
4. Step through all the instructions, until you have the memory address of the string in a register.
5. Examine the memory at the address found in the previous step (`gdb`: "`x`").

3.2 Edit an Assembly Source File

1. In the comments of the program, add a "modified by" comment with your name.
2. Change the output text to "`Eat at Pete's!`\n", and build the file again.
3. As before, debug the executable and print out the relevant memory addresses.

Do not forget to compare the values stored in the registers with your expectation.

3.3 Access Memory

Copy the address of the string into register R8. Copy 8 bytes of the content of the string to R9. Inspect with the debugger.

3.4 Implement a Loop

Extend the program to contain a loop. Use the loop to output the text 5 times. For this, use register R15 to store the loop counter. Use a label and a jump instruction to repeat the output.

Note: You could also use another register, like RCX for example. However, the "classic" registers are often modified by `syscall` and it would therefore be required to save them somewhere (typically on the stack, as we will see next time).