Computer Architecture

Richard Röttger – SDU, Fall 2017

Lab 1

Lab 1 – Get used to assembly

This first lab course is dedicated to get used to the process of writing **assembly** code. This includes a couple of simple exercises and should enable you feeling comfortable using assembler (I mean as comfortable as assembler can be ;-). We will build upon these small first steps and start writing little programs in future lab assignments.

Before you start, you can make yourself familiar with the **GNU debugger**. In order to be able to run this debugger, you have to include dubbing flags into the code. This is simply done by the assembler with the flag `--gstabs`. Assume we have our program test_program.s:

```
> as --gstabs test_program.s -o test_program.o
> ld test_program.o –o test_program
```

Now, you can use the GNU debugger by simply calling

```
> gdb test_program
```

Which leaves you in a console like environment:

```
(gdb)
```

If you want to go through the program step by step, you first set a breakpoint, run the program and then use the *next* command to execute the next instruction:

```
(gdb) break _start
(gdb) run <input>
(gdb) next
(gdb) next
(gdb) …
```

next treats subroutine calls as a single line. To follow the execution into calls use *stepi*:

```
(gdb) stepi
```

Hint: Hitting enter without a command on the line executes the last input. Use this to move quickly through instructions.

continue continues the execution until the next breakpoint is encountered.

```
(gdb) continue
```

You can display the current contents of the registers with
```
(gdb) info registers
(gdb) info all-registers
```
or
```
(gdb) info register <name>
```

For further information about gdb features use the help command.
```
(gdb) help
(gdb) help <command group>
(gdb) help <command>
```

1. Output
Objectives:

- Download the small snippet from the website.
- Use a `mov` instruction to put a constant in RAX.
- Compile and run the program. This should output the value of the RAX register.

2. Looping
Objectives:

- Put a constant in, e.g., RAX
- Calculate the sum of 1+2+…+RAX
- Output that sum.

Hint: Use labels and the conditional branches you learned in the lecture.

3. Multiplication
Objectives:

- Put a constant in, e.g., RAX.
- Calculate the product of 1*2*…*RAX
- Output that sum.

4. More Sums
Objectives:

- As before, but this time calculate the sum of each multiple of 3 or 5 from 1 to the chosen constant. (e.g. 3+5+6+9+10+12+…)

5. GDB
Objectives:

- Try out gdb. Step through the program and note the changes of the registers. Find the solution from the last task through the debugger.