LUT University

LUT Computer Vision and Pattern Recognition Laboratory 2022-02-22

BM40A0202 Foundations of Computer Science

Olli-Pekka Hämäläinen

Exercise 7 (week 10): GNU Assembly.

Tasks (1 p/task)

1. Task in Moodle.
2. Linux Ubuntu is needed in order to perform the following tasks. (Note: There is a separate help file in Moodle that will help those unfamiliar with Ubuntu to complete the task.)

   a) **Install the Linux Ubuntu operating system (64-bit) on your virtual machine (unless you already are running it on your computer).** See instruction file in Moodle.

   Let's learn a bit about the assembly programming language (more specifically, the x86-64 GNU Assembly language, using AT&T syntax and the Unix standard call convention System V ABI). Install the Linux Ubuntu operating system (64-bit) on your virtual machine. Create a file hello_world.s and copy the following to its contents:

```
.file "hello_world.s"
.global _start

.data
msg: .ascii "Hello world\n\0"
syscall_write_opcode: .quad 1

.text
_start:
movq syscall_write_opcode, %rax # syscall write
movq $1, %rdi # file stream stdout
mov $msg, %rsi # source string address
movq $13, %rdx # n.o. bytes to write
syscall

movq $60, %rax # syscall exit
movq $0, %rdi # return code 0
syscall
```

   Compile the program with the command: gcc -nostdlib -no-pie hello_world.s -o hello Run the program with ./hello and then answer the following question:

   b) What does the program print? **(Attach a screenshot)**

3. Let's analyze the program in previous task a bit more.

   a) What variables are defined in the source code?

PO BOX 20, FI-53851 LAPPEENRANTA, FINLAND
E-mail: olli-pekka.hamalainen@lut.fi

b) How is the exit from the program implemented?

4. See an example of a GNU Assembly language program (nj.s) on the course home page in Moodle.

   a) Modify that program so that you can measure the execution time of each operation. The execution time of a program can be measured (with some accuracy in a Unix environment), e.g. with the command

   $ time prog

   where the time command measures the time taken to execute the program "prog".

   Today's computers are very fast, so a large number of executions of each operation must be performed in order for the runtime measurement to show different results for different operations. For example, on some machine (Intel Core I7, 2.2Ghz) the operation `add` has to be executed 1000000000 times so that the "time" command shows an execution time of about one second.

   b) Test the execution times of various symbolic machine language operations `add, div, mul, sub` (on I86-64 hardware) and make a table of the results. **As a starting point, the aim would be to get a runtime of about 1 second when using the add operation with floating point numbers**

	add	div	mul	sub
floating point	(1.234 s)			
integer				

   c) Does it matter whether the operation is done with floating point or signed integers?

For example, from https://en.wikibooks.org/wiki/X86_Assembly/GAS_Syntax one can find more information on GNU Assembly programming.