# COMP1562, Lab #3 Processing programs

|  |  |  |  |
| --- | --- | --- | --- |
| Lecturer | **Mariusz Pelc** | Phone | **020 83318588** |
| Office | **QM366** | e-mail address | **m.pelc@gre.ac.uk** |
| Office hours | **Mon 4-5pm, Wed 9-10am** |  |  |

**(by M. Pelc and K. McManus and C. Ierotheou)**

**Description:**

**This exercise is mainly focused on tracing the execution of a real program. Students are supposed to do all the tasks specified in the section *Tasks*. All results (preferably in the form of screenshots) documenting your activity should be included into your final report.**

**Learning Outcomes:**

**Students will understand how processor processes a program and how the execution affects the processor’s registers, memory and I/O devices. Student will also learn how to assemble and execute assembly programs in the Linux operating system.**

**EXERCISES**

1. **Assembly / debugging Exercise 1.**

Login to your student Linux server (*stulinux.cms.gre.ac.uk*) using putty program.

* 1. Create a file called program\_name.asm (for example 1.asm) using pico (or, if pico is not installed, use nano or vim editor instead) with the following content

section .data

hello: db 'hello world!',10,13

helloLen equ $-hello

section .text

global main

main:

mov eax,4

mov ebx,1

mov ecx,hello

mov edx,helloLen

int 80h

mov eax,1

mov ebx,0

int 80h

* 1. Save the file.
  2. Create a script file called for example do\_asm using any editor and make the file content to be:

nasm -f elf -l $1.lst $1.asm

gcc -m32 -o $1 $1.o

* 1. Save the file.
  2. Change the do\_asm script rights to make it executable:

chmod +x do\_asm

* 1. Run the script to build the program:

./do\_asm your\_program\_file\_name

* 1. Run the program by typing:

./your\_program\_file\_name

* 1. Run the GNU debugger to debug the program:

gdb your\_program\_file\_name

* 1. Type the following sequence of commands in the command line of gdb program (for what to pay attention at just read *man gdb*):

set a breakpoint at symbol *main*

run the program

set a specific disassembly style

disassemble main

show information about registers

display esp register

display eax register

on step forward

display esp register

display eax register

go to next breakpoint (if present)

backtrace of the stack

quit

confirm quit

break main

run

set disassembly-flavor intel

disassemble main

info registers

print/x $esp

print/x $eax

nexti

print/x $esp

print/x $eax

next

info stack

q

y

For your assembly program try to:

* + display registers values after every **nexti** instruction
  + watch program counter values (EIP register)
  + watch flag register values (EFLAGS register)
  + discuss / comment on relevant results / screenshots

## Assembly / debugging Exercise 2.

Repeat exactly the same steps as in **Exercise.1** for the following assembly program. The program name can be for example 2.asm.

section .data

NUMBER1: dw 2

NUMBER2: dw 4

section .text

global main

main:

mov ax,[NUMBER1]

add ax,[NUMBER2]

mov [NUMBER2],ax

mov eax,1

mov ebx,0

int 80h

## Assembly / debugging Exercise 3.

Based on **Exercise 1 & 2** try to make changes to the code so that it would divide NUMBER2 by NUMBER1 instead of adding the two numbers together (you may need to use the documentation for nasm – see table). Run the program in the debugger and check if the division operation result is correct.

**The modified program and screenshots documenting its operation should be included into your upload document.**

**TASKS**

* 1. **Based on the example programs and the information you found on the Internet try to write part of the program that calculates area of trapezoid of the following parameters:**

- shorter side **a**=3,

- longer side **b**=7,

- height **h**=4.

You are not required to write the whole code. All you need to do is to show how you would declare all variables that you would need to calculate the area of trapezoid (a, b, h), for example, if your task were to calculate area of a square which side r=3, you would define the following variables:

r: dw 1

result: dw 1

and also you will need to write only the instructions that will do the requested calculations. For example, if your task were to calculate the mentioned area of square of side r=5, your implementation might look like:

mov word[r],5

mov ax,[r]

mov bx,[r]

mul bx

mov word[result],ax ;the last instruction should be moving

;calculation result to the **result** variable.

**You’ll be able to enter independently the variables declaration part and the code part into the scriptcheck.cms.gre.ac.uk system (using appropriate text areas provided) and the system will check if your solution is correct. The marking script will check whether your code contains all instructions required to implement the trapezoid area equation. This means that even if your program returns correct result (in the numerical meaning), your code structure still matters and you will not be awarded full marks for the code part unless you implement the equation appropriately.**

You can use available registers and the variables you will declare yourself. All the information you may need can be found at:

|  |
| --- |
| **Nasm documentation** |
| 1. Short introduction: <http://docs.cs.up.ac.za/programming/asm/derick_tut/> 2. Quite detailed description of NASM:   <http://www.pravaraengg.org.in/Download/MA/assembly_tutorial.pdf>   1. Full **nasm** documentation: <http://www.nasm.us/doc/> |

**Techniques/resources:**

**Solution of all the above tasks will require using Linux operating system which is available on the university Intranet.**

For the future you may also give a go to your Linux virtualboxes (there is one per group created). The whole above exercise is also doable on your virtual boxes once you do the following (virtual machines names are os-01, os-02, ...os-38):

a) from student.cms.gre.ac.uk

- ssh vme

- vme:~> ssh comp1562-lin-your\_group\_no -l root

and then use root password which is simply ‘root’

b) Update date yum

- yum update

c) Install nfs-utils

- yum install nfs-utils

d) install nasm

- yum -y install nasm

e) install gcc & gdb & libs

- yum -y install gcc gdb glibc.i686 libstdc++-devel.i686

- yum install /usr/bin/debuginfo-install

f) you may want to change root password (be VERY VERY careful and most of all REMEMBER THE PASSWORD); the system will ask you to type the password twice:

- passwd

g) the below solution is only in case the above will not allow you to nasm and gcc the programs.

just in case the three following commands

- yum -y install gcc

- yum -y install glibc-devel.i686 glibc-devel ibstdc++-devel.i686

- yum -y install glibc-devel.i686 libstdc++-devel.i686

- yum install compat-gcc-34

- yum install gdb

- yum install /usr/bin/debuginfo-install

**Marking:**

**The solutions will be marked in the range 0-100%.**

**Deadline:**

**The solutions should be uploaded into Teachmat within one week from the lab date (by Sunday midnight).**