Program 1: Hello World!

1 Objective

This first activity is just to get you familiar with NASM (and also your Debian development environment). If you can run this first program of ours, then we know that your development environment is working correctly, and that you can assemble, build, and run programs. We'll first do some final configuration for your Debian machine. We will then code, assemble, and run a simple program. This will allow you to get a feel for how everything works.

2 Due

Before Midnight, Thursday, 16 September 14 February

3 First Program

Our first program will be pretty simple. We are going to code up the seminole "Hello World!" program. Fire up your Debian machine and let's get started!

3.1 Step 1: Build NASM Libraries and Example Code

I recommend that you create a directory to organize your work. You can have your NASM machine keep all of your work on your host OS by using the shared folder in VirtualBox (this makes it easier to submit assignments, etc.). You can use the "link" command in Debian to provide a shortcut to your shared directory (and any special subdirectory that you want to use for your assembler work).

Example: I want to keep all of my work within "Documents/CPSC232/Assignments", under my home directory (and I shared my home directory with my Debian guest). On my Linux machine, I would open a Terminal, and type the following:
`ln -s /media/sf_jochen/Documents/CPSC232/Assignments` (obviously change this to match your setup). This will create a link (shortcut) on your linux machine (in your home directory) to the folder on your host OS.

Note: If you get a permission error when you try to access that directory, this means you need to go back to the instructions on setting up the shared folder. You need to add the "nasm" user to the "vboxsf" group.

On your host OS, dowload the linux-ex.zip archive file from D2L and copy it to the folder that you are sharing with your Linux machine. Now go to your Linux machine, in that same terminal window, and use the "change directory" command to navigate to where the library/example code is: `cd Assignments` (assuming that you set up the shared directory like I did above).

Tye the command `ls -al` and you should see a directory listing that includes the `linux-ex.zip` file. To expand the archive, type `unzip linux-ex.zip` and use the `cd` command to change directories into the linux-ex directory. Let's build the example code and libraries (type the following):

- `make clean` (This will remove all of the old build files.)
- `make all` (This builds all of the libraries and the example programs - you should not see any errors. If you do see errors, ask for help.)

Try one of the example programs (like maybe "math"). You can do a directory listing to see them all (`ls -al`). To run a program in Linux that resides in the same directory that you are currently working within, you need to provide the path: (example: `./math`) Type that exactly as you see it. It should prompt you for a number. Enter a number and see that the output makes sense. If all goes well, then you successfully built the example programs and the libraries!

3.2 Step 2: Edit NASM Source Code

Create a subdirectory in your NASM work folder for Assignment 1. Within that directory, you need to copy the following files to your assignment directory so that you can build and run your program (these will live in the linux-ex directory):

- asm_io.inc
- asm_io.o
- cdecl.h
- driver.c

Now let's edit the source code file. You have many editors available to you within Linux. Let's start with something really simple like either nano or pico (these are installed by default). However, I encourage you to experiment with other editors, like emacs (or xemacs), eclipse, and vi.

Create a new file (in the directory for this assignment) with the following contents:

```
%include "asm_io.inc"

segment .data

        mesg    db    "Hello Assembly World!",10,0

segment .text
        global  asm_main

asm_main:
        enter   0,0
        pusha

        mov     eax, mesg
        call    print_string

        popa
        mov     eax, 0
        leave
        ret
```

Save this file as "hello.asm" (note the .asm extension)

Time to compile/assemble! `nasm -f elf hello.asm`

You should not see any complaints or errors. If you get a file not found error, at the command window, type: `ls -al` and see if you can see a file named `hello.asm`. If you can't, you may have

saved it to the wrong place, or you may have mistakenly saved it as a text file. If you see hello.txt, then rename the file to `hello.asm` and try again.

Once the file assembles correctly, type the following:

```
gcc -m32 -o hello hello.o driver.c asm_io.o
```

Again, if all goes well you will see no errors. Type:

```
./hello
```

and you should see your message displayed!

If you run into any trouble along any of the steps, contact me and/or post on the class help forum as soon as possible. We need to get your environment set up quickly so that we may do the rest of the work for the semester.

4 What to submit:

You must submit you assignment two ways:

1. Submit the source code (`.asm` file) electronically to the D2L dropbox
2. Print out your source code, nicely formatted, and attach the cover sheet from the last page of this lab sheet. Bring this with you to class immediately after the due date (the date that your assignment is submitted electronically to D2L will be the date/time that I use to determine "on time" submission) .
3. Be sure to include your name in the top of your program file in the comment block!

Program 1: Hello World!

I certify that the submitted program is the result of my own efforts, and that I have not shared any source code with others. Further, my program assembles correctly and has been tested for correctness (check each box as appropriate in the table below and sign).

Checkbox	Description
	The program assembles and compiles correctly
	I have tested the program, and it runs correctly with my test input

Notes:

Student Signature: _____

This table is where your scores for grading will be recorded:

Earned Points	Description
	Program printout is neat and readable, code style promotes understanding (1 point)
	Syntactically Correct: Assembles/Links correctly (5 points)
	Semantically Correct: Runs correctly, produces correct output (4 points)
	Penalty points: Lateness, failure to submit printout / source file / completed checklist not stapled
	Total points (10)