1 Introduction

This document contains both practical 1 and assignment 1. In general the assignments will build upon the work of the current practical.

1.1 Submission

The practical component must be marked within the session by a tutor and an upload to fitchfork must be made in the session.

You have until Friday the 7th of August at 17:00 to complete the assignment component. The assignment must be uploaded to the CS site. **No late submissions will be accepted.** The upload slots will become available on the Monday the 27th of July at 11:00.

1.2 Plagiarism policy

It is in your own interest that you, at all times, act responsible and ethically. As with any work done for the purpose of your university degree, remember that the University of Pretoria will not tolerate plagiarism. Do not copy a friend’s work or allow a friend to copy yours. Doing so constitutes plagiarism, and apart from not gaining the experience intended, you may face disciplinary action as a result.

For more on the University of Pretoria’s plagiarism policy, you may visit the following webpage: http://www.library.up.ac.za/plagiarism/index.htm
1.3 Practical component [25%]

You must first complete all 4 task of this practical. Once you have done so you must get marked by a tutor in the session. You will be asked to show what you have done for each task, and why. You must be ready to be marked 30-minutes before the end of the session.

1.3.1 Task 1: Exit program [10 %]

For this task you must implement the following simple assembler program:

```assembly
segment .text
global _start

_start: 
    mov eax,1
    mov ebx,5
    int 0x80
```

You must assemble and link the program. Once you have done this change the program such that the return code is 0.

1.3.2 Task 2: Data example [10 %]

For this task you must use the yasm feature of generating listings to obtain the way the following data items are stored in memory (use dd):

```
0.0
-0.0
0
-0
3.3651
123.1
-3.6
1.5
```

1.3.3 Task 3: Hello World [5%]

For this task you have to implement the following 64-bit hello world assembly program in a file called `hello.asm`:

```assembly
segment .data
hello: db "hello world!",0x0a

segment .text
global _start

_start:
    mov eax,1
    mov edi,1
```
mov edx, 13 ; The number of characters
lea rsi,[hello]
syscall
mov eax,60
xor edi, edi
syscall

You are not at all expected to fully understand the code at this point in time. You are simply required to alter it to display the following:

My student number is SXXXXXXXX

Where the X’s must be replaced with your student number.

When you are have finished, create a tarball containing your source code file and upload it to Fitchfork on the CS website, using the Practical 2 upload link. You have 5 uploads for this task.

1.4 Assignment component [75%]

There exist many claims about the speed and space utilization of assembler. This assignment will focus on trying to validate or invalidate these claim in a very simplified context.

You are required to implement a "hello world” esque program where the string The quick brown fox jumps over the lazy dog. is output using a number of programing languages. For each language you must record the execution time and the size of the "executable” file.

The languages and compiler version to be used are:

- Java using javac 1.8.
- Fortran using gfortran gcc version 5.1.0-5
- Cobol using cobc OpenCOBOL 1.1.0
- Lisp using GNU CLISP 2.49-4
- c++ using gcc version 5.1.0-5
- Assembler using yasm version 1.3.0-1

These are all installed in the lab. For c++ you must test using cout (not printf). For all languages utilize the default optimization flag (i.e. don’t set a optimization flag).

Given that the program that you plan on executing is so small many external factors could substantially alter the completion time. For example your program being prioritized lower than some system maintenance task. To mitigate this, calculate the time required for the program to execute 1000 times, call this value $\lambda$. Now the time one execution would take is on average $\lambda/1000$. It is still possible that one run out of the 1000 took drastically longer to execute due to external influences, and as a result the average time might be biased
heavily. It is advised that you calculate 100 $\lambda$s and report the minimum and average of these 100 $\lambda$s. It is also highly advisable to use a high precision timer.

You must write a 2 page report summarizing your experimental set up (i.e. run environment), results, and conclusions.

One you have completed this assignment you must create an archive containing everything that would be required to run your program and obtain your empirical results as well as your report. This archive must be uploaded to the slot Assignment 1 Task 1. This assignment will be hand marked by the course assistants.

2 Mark Distribution

<table>
<thead>
<tr>
<th>Activity</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>10</td>
</tr>
<tr>
<td>Task 2</td>
<td>10</td>
</tr>
<tr>
<td>Task 3</td>
<td>5</td>
</tr>
<tr>
<td>Assignment component</td>
<td>75</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>