Department of Computer Science

COS284 Practical and Assignment 3: Jump Around

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

1 Introduction

This document contains both practical 3 and assignment 3. In general the assignments will build upon the work of the current practical.

1.1 Submission

The practical will be fitchfork only and is due at 17:00 on Friday 11 September, where as the assignment will also be fitchfork only and due at 17:00 on Friday 18 September. You may use the Discord server to ask for assistance with the assignment and practical components.

1.2 Online Resources

- Euclidean Algorithm: `https://www.whitman.edu/mathematics/higher_math_online/section03.03.html`
- Determining if a number is a prime number: `https://www.geeksforgeeks.org/wilsons-theorem/`

1.3 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`

1.4 Practical component [28%]

1.4.1 Task 1: GCD [6%]

For the first task of the practical you will be required to implement a program to calculate the GCD of two numbers. An example output would be:

```
Enter the first number: 20
Enter the second number: 15
05

Enter the first number: 50
Enter the second number: 10
10
```

You can assume that input and output will always be 2 digits. When you have completed the task you must create a tarball containing your source code file named task1.asm and upload it to the Practical: Task 1 upload slot on the CS website.

1.4.2 Task 2: Prime numbers [8%]

For this task you will be required to implement a program to calculate the largest prime number that is less than or equal to an input value. An example output would be:

```
Please input a number: 05
05

Please input a number: 15
13
```

You can assume that input and output will always be 2 digits and input values will not be greater that 20. When you have completed the task you must create a tarball containing your source code file named task2.asm and upload it to the Practical: Task 1 upload slot on the CS website.

1.4.3 Task 3: Towers of Hanoi [14%]

For this task you will be required to implement a program that receives a number as input and displays the minimum number of moves necessary for the discs to be moved from one pole to any other pole.

```
Please input a number: 03
7

Please input a number: 15
32767
```

All input values will be two digits and output can be of variable length. When you have completed the task you must create a tarball containing your source code file named task3.asm and upload it to the Practical: Task 3 upload slot on the CS website.

1.5 Assignment component [72%]

In this assignment you will be required to implemented some parts of Booths algorithm. Specfically, you are required to print out all the intermediary numbers that are added together to produce the final result in the algorithm.

For input you will be given 2 signed integers in base 10 which are representable as 16-bit 2's complement binary numbers. The numbers will be separated by a space '0x20'. Below is an example of input you may expect where \n represents the newline and ... represents your programs output:

```
5634 -345\n
...
```

Your programs output must be a list of numbers in 32-bit 2's complement representation separated by spaces. You must output all **non-zero** numbers produced by Booths algorithm.

You **must** use the algorithm given in the textbook or slides provided. Below we present psuedocode for the part of Booths algorithm required for this assignment:

```
α = multiplicand
β = multiplier
m = 0

while True:
        n = β & 1

        if n = 1 and m = 0:
                print −α in 32−bit 2s complement
        elif n = 0 and m = 1:
                print α in 32−bit 2s complement

        α *= 2
        m = n
        if β >> 1 = β:
                break
        else:
                β >>= 1
```

Below we provided examples of how the program should run if 53 and 126 are given as input:

```
53 126
11111111111111111111111110010110 00000000000000000001101010000000
```

Another example using -7 and 2 as input:

```
-7 2
00000000000000000000000000001110 11111111111111111111111111100100
```

When you are finished, create a tarball containing your source code file named **main.asm** and upload it to the assignments.cs.up.ac.za website, under the **Assignment 3** submission slot.

2 Mark Distribution

Activity	**Mark**
Practical	28
Assignment	72
Total	**100**