Department of Computer Science

COS284 Practical and Assignment 5: Assembly and C

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Contents

1 Introduction

This document contains the both practical 5 and assignment 5. In general the assignments will build upon the work of the current practical.

1.1 Submission

The practical component must be uploaded to fitchfork during or before your practical session. The assignment component will be marked in the practical session by your tutor.

You have until Friday the 7th of October at 17:00 to complete the assignment component. **No late submissions will be accepted.** The upload slots for the practical will become available on Friday the 23rd of September and will close on Friday the 30th of September at 17:00. The upload slots for the assignment will become available on Friday the 30th of September.

1.2 Plagiarism policy

It is in your own interest that you, at all times, act responsibly 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 both tasks of this practical.

1.3.1 Task 1: Run Length Encoding - 10 Marks [10 %]

Carefully examine the given source file task1.c. The main method utilizes a method called **encode**, this method must be implemented in assembly. The task1.c will be given access to the method when it is linked with **encode.o**. The **encode** method should perform the following:

1. Accepts a string;
2. Encodes the String using Run Length Encoding;
3. Outputs the encoded String to the terminal;

Refer to `http://www.geeksforgeeks.org/run-length-encoding/` for details on Run Length Encoding.

Please pay attention to the conventions when calling an assembler program from c. Use the given task1.c to test that your assembly method works correctly.

When you are have finished, create a tarball or zip (uXXXXXXXX-task1.tar or uXXXXXXXX-task1.zip, where XXXXXXXX is your student number) containing your source code file and upload it to the CS website, using the **Practical 5 Task 1** upload link. Name your assembler file **encode.asm**.

1.3.2 Task 2: Shapes - 15 Marks [15 %]

Carefully examine the given source file **task2.c**. The main method utilizes a method called **shaper**, this method must be implemented in assembly . The **task2.c** will be given access to the **shaper** function when it is linked with **shaper.o**. The shaper method should perform the following:

1. Accepts the address of a **Shape struct**
2. Determines the name of the shape based on the non 0 dimensions.
3. Inserts the shape name into the struct (lower case)
4. Inserts the corresponding formulafor calculating the area into the struct (lower case)
5. Calculates the area
6. Inserts the area into the struct

given the following C struct:

```
struct Shape {
    int id;
    char name[19];
    char formula[67];
    double area;
    double A;
    double B;
    double C;
    double D;
    double radius;
};
```

There are only 4 valid shapes: Triangle, Circle, Square and Rectangle. The lengths of the sides of the shapes are represented by the struct properties A,B,C and D. Only the radius is used in the case of a circle. You may assume all triangles are right-angled triangles.

Please pay attention to the conventions when calling an assembler program from c as well the conventions for handling structs.

Use the given task2.c to test that your assembly method works correctly. task2.c is responsible for displaying and formatting the output.

Example execution of your program is shown:

```
alex@linux-PC:/home/Documents$./task2.out
ID: 0
Name: Triangle
Formula: 0.5*h*b
Area: 0.000000
Side A: 1.000000
Side B: 4.000000
Side C: 3.300000
Side D: 0.000000
Radius: 0.000000
```

You may assume all shapes passed to the method are valid shapes.

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 5 Task 2** upload link. You have 10 uploads for this task. Name your assembler file **shaper.asm**.

1.4 Assignment component - 40 Marks [75%]

All source images in this practical will be bitmap images with 8 bits per pixel bitmap images (8-bit true colour). The following urls may be useful: https://en.wikipedia.org/wiki/BMP_file_format and https://en.wikipedia.org/wiki/8-bit_color

This assignment consists of three parts:

1.4.1 Part 1: Border - 10 Marks

Carefully examine the given source file part1.c. The main method utilizes a method called **border**, this method must be implemented in assembly. The part1.c file will be given access to the **border** method when it is linked with **border.o**. The **border** method should perform the following:

1. Accepts two strings; the first being the file name of the source bitmap image, and the second the name of the destination bitmap image
2. Reads the image file and places a black 1 px border around the edge of the image (the image size must remain the same).
3. Writes new image to a bitmap file

Please pay attention to the conventions when calling an assembler program from c.

Use the given part1.c to test that your assembly method works correctly.

When you are have finished, create a tarball or zip (uXXXXXXXX-part1.tar or uXXXXXXXX-part1.zip, where XXXXXXXX is your student number) containing your source code file and upload it to the CS website, using the **Assignment 5 Part 1** upload link. Name your assembler file **border.asm**.

1.4.2 Part 2: Crop - 10 Marks

Carefully examine the given source file part2.c. The main method utilizes a method called **crop**, this method must be implemented in assembly. The part2.c file will be given access to the **crop** method when it is linked with **crop.o**. The **crop** method should perform the following:

1. Accepts two strings and two integers; the first string being the file name of the source bitmap image, and the second string the name of the destination bitmap image, the first integer being the new width of the image, and the second integer being the new height of the image.
2. Crops the image according to the input dimensions
3. Writes cropped image to a bitmap file

Please pay attention to the conventions when calling an assembler program from c.

Use the given part2.c to test that your assembly method works correctly.

When you are have finished, create a tarball or zip (uXXXXXXXX-part2.tar or uXXXXXXXX-part2.zip, where XXXXXXXX is your student number) containing your source code file and upload it to the CS website, using the **Practical 5 Part 2** upload link. Name your assembler file **crop.asm**.

1.4.3 Part 3: Color intensity - 20 Marks

Carefully examine the given source file part3.c. The main method utilizes a method called **intensity**, this method must be implemented in assembly. The part3.c file will be given access to the **intensity** method when it is linked with **intensity.o**. The **intensity** method should perform the following:

1. Accepts two strings and one integer; the first string being the file name of the source bitmap image, the second the name of the destination bitmap image and the integer representing red (0), green (1) or blue (2).
2. Reads the image file and maximizes the intensity of the selected color
   e.g ('source.bmp','dest.bmp',2) will maximize the intensity of the blue in the image.
3. Writes new image to a bitmap file

Please pay attention to the conventions when calling an assembler program from c.

Use the given part3.c to test that your assembly method works correctly.

When you are have finished, create a tarball or zip (uXXXXXXXX-part3.tar or uXXXXXXXX-part3.zip, where XXXXXXXX is your student number) containing your source code file and upload it to the CS website, using the **Practical 5 Part 3** upload link. Name your assembler file **intensity.asm**.

2 Mark Distribution

Activity	**Mark**
Task 1	10
Task 2	15
Part 1	10
Part 2	10
Part 3	20
Total	**65**