Samuel Boulton

169039695

Assignment c and assembly

Assembly Language vs C

Table of Contents

[Introduction 2](#_Toc29740624)

[Description 2](#_Toc29740625)

[C 2](#_Toc29740626)

[Assembly 2](#_Toc29740627)

[How Data is stored in Memory 3](#_Toc29740628)

[C 3](#_Toc29740629)

[Assembly 3](#_Toc29740630)

[Comparison of the two Programs 3](#_Toc29740631)

[Conclusion 4](#_Toc29740632)

# Introduction

This document will explain how the code I have created works, the way it uses memory and provides a comparison of the two programs. The document is in sections, you can find your way around the document by using the table of contents to assist you.

# Description

## C

The C code that I have written is a short, simple and effective program to display the FizzBang method. Firstly print the FizzBang Demo title then a simple for loop with 3 if statements inside it. The if statements go in order 15, 5 then 3 testing if the value in the variable a is divisible by any of these numbers. If the value is divisible then the printf will the string to the console. Otherwise if the number is not the divisible then print the value that is the variable a. It does this loop to the max times of 20.

## Assembly

The Assembly Code works in a similar way with testing the values, running a loop system with the registers and printing the output to the console. Firstly declaring all the registers to the correct values and for example the amount of the characters used to print certain messages. Also in the section .text we are placing the title FizzBang Demo here making sure it isn’t inside any part of the loop.

Next I defined the sections of comparisons, in this the code needs to compare the value to which will be used by the main loop further into the code to compare the certain number to the divisible numbers. For example: the label five includes the code that the system will run through to see if it is divisible by 5. The loop is made before the number sections are created as the loop needs to start and check if the value of the register is equal to 20, because if it is then the loop will end.

Next the section .data is important as this contains all the data information that the system needs to run the code, for example: I have written in this section the hexadecimal numbers as the system needs to know what to output. However as it is in hexadecimal, letters print as well as numbers.

# How Data is stored in Memory

## C

In the C program the memory is stored in the stack in frames. In my code I have written a function called main(), which includes a primitive variable called int a. This creates a frame on the stack names main() with the local variable a stored inside it and all data is stored in binary. Firstly the data puts the integer a with the value 1 inside the stack because within the for loop it is defined as 1. The first if statement comes and asks if a is divisible by 15, 5 or 3, which it isn’t so the loop adds 1 onto the integer a making it 2. Then the value of 2 is placed on top of the first value inside the stack. The system does this till the for loop is completed and the recursion process continues by printing the memory stored out from the top of the stack first.

## Assembly

The Assembly program uses various registers and has many functions. The main function in the program brings words/commands out of the memory in an order which is written. The program will define the data into certain registers inside the memory and the functions of the code will bring each part of that register out of the memory, use the information inside and change that same information and place it back inside the memory. The code that is written the sections named ‘five, three and fifteen’, just compares what information is in the registers. However the loop process pulls the data from the registers which is in the memory and uses the comparing sections (three, five and fifteen) to compare the statements to the data in the registers. The section .data is the part which stores the useful data in the memory. This include the hexadecimal numbers which will be used in the output.

# Comparison of the two Programs

I believe that the two different programs use memory in a different way. In the C language the registers are only used when the programmer defines it in the code. Putting ‘register’ before the integer for example, will allows the code faster access to the storage. Registers are faster than memory to access, so the variables that are used frequently in a C program can be put into registers to allow the code to run faster. However the code that I have written doesn’t include this ‘register’ command.

The Assembly program on the other hand, does use registers for fast access to memory. The program will fill the registers with data and use them as a quick in and out access.

Programs like the Assembly program can execute faster; while programs in C are easier to develop and maintain.

The memory layout for each program works near enough the same. The text section and the data section is allocated in the same low address section in the memory, then command line and the stack segment is in the high address.

# Conclusion

Overall the two programs are both different and the same concurrently. Both of the programs test and compare the values of data and output the correct information to the console. However there is a main difference between the two programs: the memory allocated in the Assembly program allows the program to be executed faster because the memory is stored in the registers. Compared to the C language which doesn’t use registers.