Programming Assignment #3

Question 1

Step 1. Write a program in GNU assembly language that uses macros to print the following messages on the screen [20 marks]:

> Hello, programmers!
> Welcome to the world of,
> Linux assembly programming!

Before I show the assembly language, I just want to state that we are not really taught how to do what this question asks. In addition, trying to piece together the information with the various different sources such as the textbook, the powerpoint notes, the disassembler programs themselves, and online resources, is quite frustrating as there are always a variety of differenes between them.

For example, **copying and pasting the program in the module 6 notes into a vi file on the linux server supplied by the school and executing the commands as recommended gives an error** saying there is no main method*. Whether this is just a standard that is no longer used or not, I don't know, but assembly code derived from the gcc compiler on the linux machine supplied by the school does not use _start anywhere. This kind of surefire confusion is frustrating and leaves a student with more questions than answers.

*

```
[musfiqcomp213136@cs2 kuby]$ as Hello.s
Hello.s: Assembler messages:
Hello.s:3: Error: no such instruction: `_main'
[musfiqcomp213136@cs2 kuby]$
```

See the following page for my answer to the question.

C code:

```c
#include <stdio.h>
#include <stdlib.h>

#define msg1 printf("Hello, programmers!\n")
#define msg2 printf("Welcome to the world of,\n")
#define msg3 printf("Linux assembly programming!\n")

int main()
{
    msg1;
    msg2;
    msg3;
    return 0;
}
```

Assembly code derived from the GCC compiler on the supplied linux machine:

```
        .file   "question1.c"
        .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
        .string "Hello, programmers!"
.LC1:
        .string "Welcome to the world of,"
.LC2:
        .string "Linux assembly programming!"
        .text
.globl main
        .type   main, @function
main:
.LFB18:
        .cfi_startproc
        subq    $8, %rsp
        .cfi_def_cfa_offset 16
        movl    $.LC0, %edi
        call    puts
        movl    $.LC1, %edi
        call    puts
        movl    $.LC2, %edi
        call    puts
        movl    $0, %eax
        addq    $8, %rsp
        .cfi_def_cfa_offset 8
        ret
        .cfi_endproc
.LFE18:
        .size   main, .-main
        .ident  "GCC: (GNU) 4.4.7 20120313 (Red Hat 4.4.7-23)"
        .section        .note.GNU-stack,"",@progbits
```

Here we clearly see the read only data section, housing the macros, as well as the .text section that contains the main method. Main simply creates a bit of space on the run time stack, loads each macro address into %edi and calls puts, which sequentially prints the macros to screen. Finally, the return value is set to zero and %rsp incremented before execution finishes via ret.