Assignment 0.2

1. **Assembling the assembly code which contains add function:**

   nasm -f elf64 addfunction.s -o addfunction.o

   This command converts our code containing the assembly level code into machine level code but it is still not linked to our **prog-add.**

2. **Preprocessing**

   $(CC) -E prog-add.c > prog-add.i

   The -E option is used to stop the compilation after the preprocessing is done.The output of this command gives us the preprocessed file which is now stored in the **prog-add.i** file.
   In the preprocessed file the included files and macros expansion takes place.
   The comments are removed and the conditional compilation also takes place here.

3. **Compiling:**

   gcc -S prog-add.i

   The **-S** option is used to get the assembly language code from the preprocessed file and the contents gets stored in **prog-add.s** file.
   The **hello.s** file contains the corresponding code of **prog-add.c** in assembly level language**.**

4. **Assembly:**

   gcc -c prog-add.s

   The **-c** option assembles the code in the source file specified.(**prog-add.s)**
   This option does not do the linking.
   The **prog-add.o** file now contains the machine level instructions for the code provided by the source file. The function add() still remain to be resolved.

5. **Linking:**

gcc prog-add.o -static addfunction.o -o prog-add

For the linking step we can directly use **gcc hello.o to** to get the linking done and -o option is used to specify the output file. The **-static** option is used when special link options are needed to resolve the dependency.
The file **prog-add** now contains the machine level instructions with the add function resolved which were not one in the step 1 & 4.

Working

First of all the program prog-add.c takes input from the user using the scanf option. When the input is stored in variable a & b it calls an external routine add with these parameters passed.The control is transferred to the add function now in mentioned in addfunction.asm.

The values passed as parameters are now stored in the registers **rdi & rsi** .
First of all the value of rdi gets stored in rax and then we add the value of rsi to the corresponding value of rax.The answer is now stored in the register rax.Fisrt of all we print the output message and check if the number is negative.If the number is negative then we transfer the flow to **_negative label.** Here we print the symbol **“-” and** multiply our result with -1 and then call the printAnswer routine.If the number is positive we simply call the printAnswer routine, the control is transferred to printAnswer now.

Some memory is reserved beforehand for the answer to be displayed in **answerarray(**for storing every digits of answer as a string**)** **& answerarrayPos(**stores position of the string **)** in section .bss.

Our subroutine will breakdown the digits backward first so we need a new line character at the beginning.We start by storing the head of **answerarray** in rcx register and storing a new line in register rbx which then goes to the first position of rcx register.
Then we increment rcx to go to next address and store it in answerarrayPos.

Now in the first loop function first we initialize rdx to 0 and rbx to 10.Then we divide rax by rbx store it in rdx and store the updated rax to get last digit.We also add 48 to rdx to make it a character.Now we store the current position in rcx.Then we load the character in the current position with the help of **mov [rcx],dl.**We increment the rcx and move it again to the **answerarrayPos pointer.** We compare the value of rax with o if its non zero then we run over this function again.

In the second loop we move the current position stored in the **answerarrayPos pointer to rcx.** Then we use the syscall **sys_write** and print the digit which is currently pointed by the **answerarrayPos** pointer. We decrement the pointer to move backward and compare rcx to **answerarray** if it's not the same then we run over this routine again till all the digits are printed.

Sources:
https://pastebin.com/PN2jKVae

Navneet Agarwal
2018348