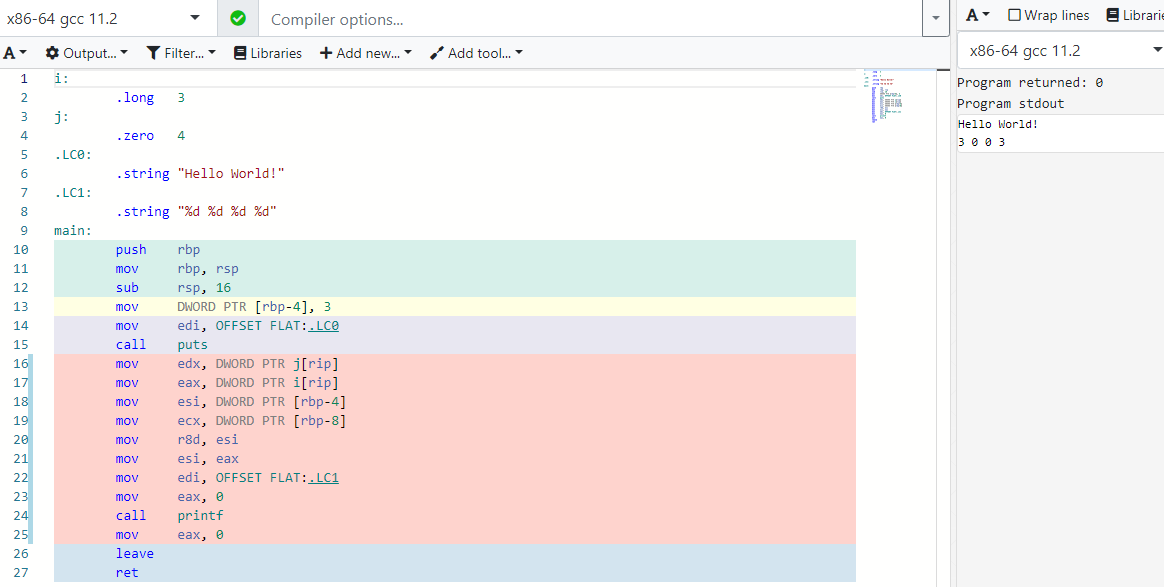
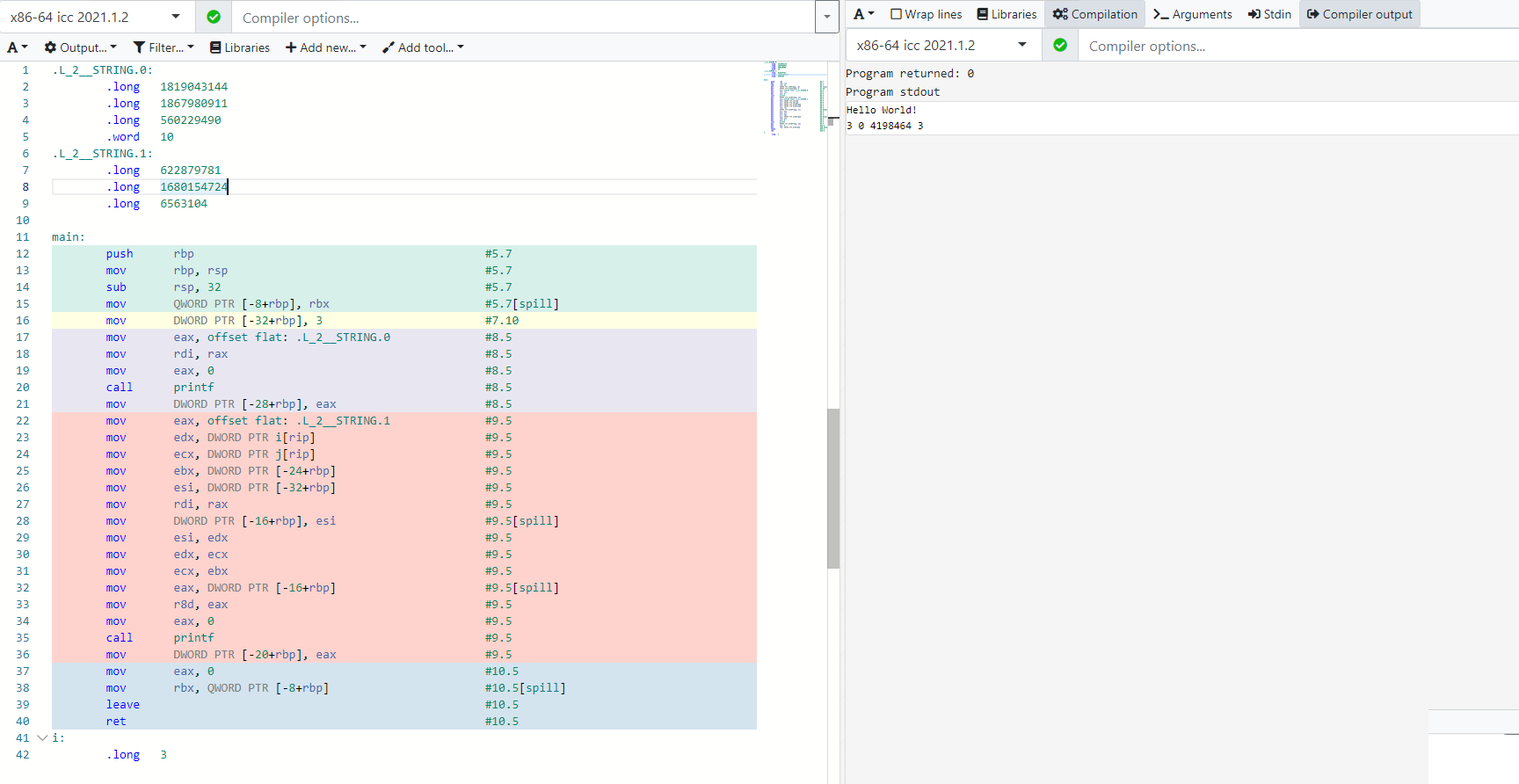
**Sistemas Operacionais**

Arthur do Prado Labaki – 11821BCC017

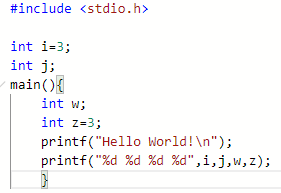
Máquina virtual Kali Linux

Atividade Prática – Unidade 2

**1-** Utilizando o site *godbolt.org*, o código em C pode ser compilado por diversos compiladores. Usando os compiladores x86-64 gcc 11.2 e x86-64 icc 2021.1.2 temos respectivamente:



Código:



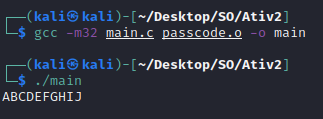
Comparando as duas compilações, a diferença vista entre os compiladores é em suas atribuições das variáveis.

No GCC, o w é atribuído a pilha no espaço [rbp-8], logo na parte de seu respectivo printf (área vermelha). Assim o local do registrador de w tem como valor o 0.

Já no o ICC, o w que é atribuído a pilha no espaço [-24+rbp] não recebe nenhum valor, devido a escolha de variáveis do tipo long, tendo como valor do registrador “lixo de memória”.

Em suma, a diferença é que o GCC armazena a variável antes da main enquanto o ICC armazena a variável depois da main.

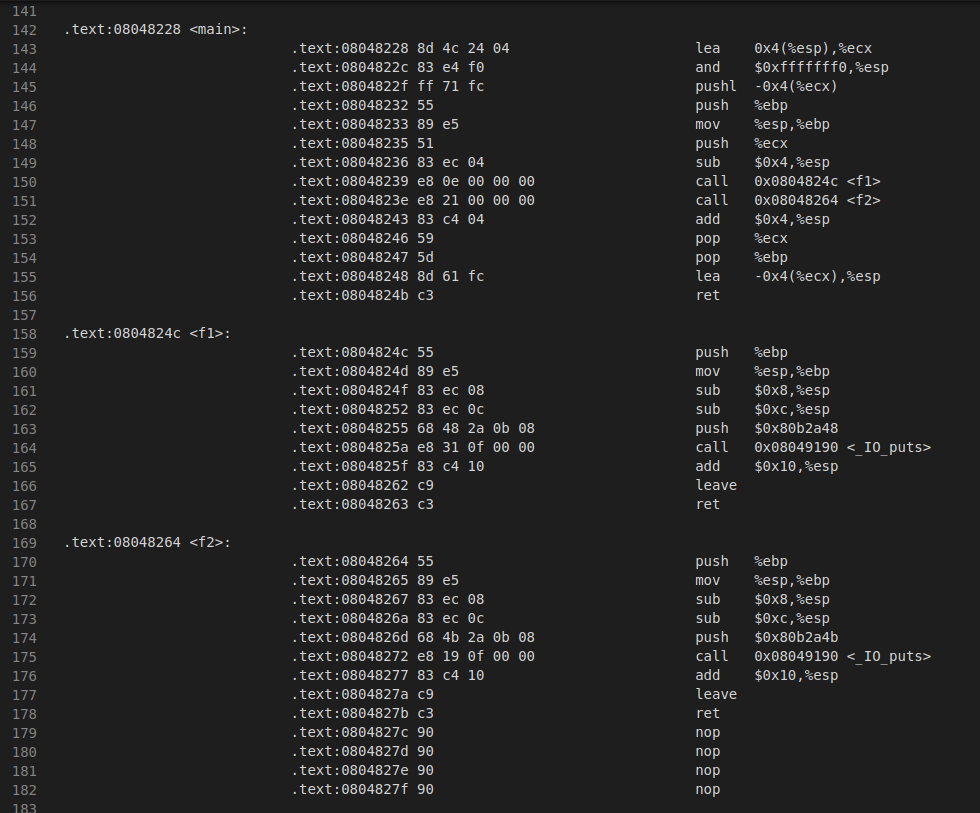
**2-** Para linkar os arquivos passcode com o código em c, foi necessário utilizar a biblioteca “gcc-miltilib”, pois o arquivo é de 32 bits e o sistema operacional usado é de 64 bits.



Com isso, obtemos a senha que é “ABCDEFGHIJ”.

**3-** Para conseguir modificar o arquivo executável, primeiro temos que identificar as chamadas das funções e seus endereços em seu código assembly. Para isso, foi necessário utilizar o site *onlinedisassembler.com* para fazer o disassebly.

Assim foi possível identificar a parte do código onde ocorre as chamadas das funções.



Para trocar as funções, temos que modificar seu código hexadecimal. O código hex “e8” remete a uma chamada relativa a instrução anterior e tanto o “0e” para f1 ou “21” para f2 refere-se a quantidade de bytes relativos a instrução anterior.

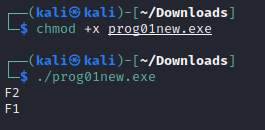
Para isso devemos subtrair o endereço da tag de f2 com o endereço da chamada da função f1 na main (08048264 na linha 169 – 08048239 da linha 150).

Também devemos fazer o mesmo com o endereço da tag de f1 com o endereço da chamada da função f2 na main (0804824c na linha 158 – 0804823e na linha 151).

Com isso temos “2b” e “0e” respectivamente. Porem a chamada da instrução é feita somando 5 bytes, então devemos retirar esses 5 bytes, resultando em “26” e “09”.

Utilizando o site *hexed.it* conseguimos editar o código hex do programa. Alterando o “0e” para “26” e “21” para ”09” temos:





**4-** Utilizando o comando time e anotando o tempo real de execução dos dois programas, temos:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **Média** |
| **Prog2** | 0,049 | 0,042 | 0,041 | 0,041 | 0,040 | 0,039 | 0,040 | 0,038 | 0,039 | 0,040 | 0,044 | 0,040 |
| **Prog3** | 0,020 | 0,018 | 0,017 | 0,017 | 0,015 | 0,014 | 0,018 | 0,017 | 0,017 | 0,017 | 0,010 | 0,016 |

Com isso, percebe-se que o prog3 é mais eficiente que o prog2, visto que, sua média é quase tres vezes menos.

**5-** Anexado