Nome: _____

Lab 01: Compilar um programa em C e produzir o output em assembly. Verificar que o mesmo programa, quando compilado para ISAs e/ou compiladores diferentes, produz código assembly diferente.

1. Use sua máquina Linux (ou https://repl.it/languages/c).
2. Use “gcc --help” e “man gcc”, para mostrar como usar o compilador gcc para compilar programas escritos em linguagem C de alto nível em código de máquina.

```
yanyh@vm:~$ gcc --help
Usage: gcc [options] file...
Options:
  -pass-exit-codes         Exit with highest error code from a phase
  --help                   Display this information
  --target-help            Display target specific command line options
  --help={common|optimizers|params|target|warnings|[^]{joined|separate|undocumented}}[,...]
                           Display specific types of command line options
  (Use '-v --help' to display command line options of sub-processes)
  --version                Display compiler version information
...
  -E                       Preprocess only; do not compile, assemble or link
  -S                       Compile only; do not assemble or link
  -c                       Compile and assemble, but do not link
  -o <file>                Place the output into <file>
```

3. Crie o arquivo “ola.c” e digite o “Olá Mundo” ao lado:

4. Compile e execute o programa ola.c.

```c
#include <stdio.h>

int main(void)
{
  printf("Olá, mundo!\n");
  printf("Aqui é ArqComp!\n");
  return 0;
}
```

5. Crie o arquivo “bubble.c” e digite o código ao lado:

```c
void bubble_sort(int list[], int n) {
  int i, j, t;

  for (i = 0 ; i < n - 1; i++) {
    for (j = 0 ; j < n - i - 1; j++) {
      if (list[j] > list[j+1]) {
        /* Swapping */
        t        = list[j];
        list[j]  = list[j+1];
        list[j+1] = t;
      }
    }
  }
}
```

6. Compile o arquivo “bubble.c” com a opção -S, que gera o arquivo “bubble.s” para o ISA de seu computador (x86). Verifique o conteúdo do arquivo “bubble.s”. Conte quantas instruções existem. Obs.: Linhas com símbolos que começam com . (.file, .text, .cfi_startproc), linhas com “labels” que contém : (bubble:, LFB0:) ou linhas com assinaturas de funções ou comentários NÃO SÃO INSTRUÇÕES.

7. Utilize o Compiler Explorer (https://godbolt.org/) para explorer outros assemblys gerados para outros compiladores e/ou ISAs, usando o código “bubble.c”.

```
yyan7@yyan7-Ubuntu:~$ vi bubble.c
yyan7@yyan7-Ubuntu:~$ uname -a
Linux yyan7-Ubuntu 5.4.0-72-generic #80~18.04.1
yyan7@yyan7-Ubuntu:~$ gcc -S bubble.c
yyan7@yyan7-Ubuntu:~$ cat bubble.s
        .file   "bubble.c"
        .text
        .globl  bubble_sort
        .type   bubble_sort, @function
bubble_sort:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movq    %rdi, -24(%rbp)
        movl    %esi, -28(%rbp)
        movl    $0, -12(%rbp)
        jmp     .L2
.L6:
        movl    $0, -8(%rbp)
        jmp     .L3
.L5:
        movl    -8(%rbp), %eax
        cltq
        leaq    0(,%rax,4), %rdx
        movq    -24(%rbp), %rax
        addq    %rdx, %rax
        movl    (%rax), %edx
        movl    -8(%rbp), %eax
        cltq
        addq    $1, %rax
        leaq    0(,%rax,4), %rcx
```

TAREFA: conte o número de instruções para o compilador/ISA especificado na tabela abaixo, usando o código do “bubble.c”. Labels (terminam com :) e diretivas (começam com .) não são instruções. Você deve contar manualmente o número de instruções do output para cada compilador/ISA especificado na tabela abaixo, usando o https://godbolt.org.

	Compilador e ISA	**Número de Instruções**
https://godbolt.org/	RISC-V rv32gc gcc latest	
https://godbolt.org/	MIPS64 gcc 5.4	
https://godbolt.org/	x86-64 clang 12.0.0	
https://godbolt.org/	ARM gcc 8.2	
https://godbolt.org/	RISC-V rv32gc clang(trunk)	
https://godbolt.org/	RISC-V rv64gc clang(trunk)	

TAREFA: o que você conclui analisando a tabela acima? Responda aqui.