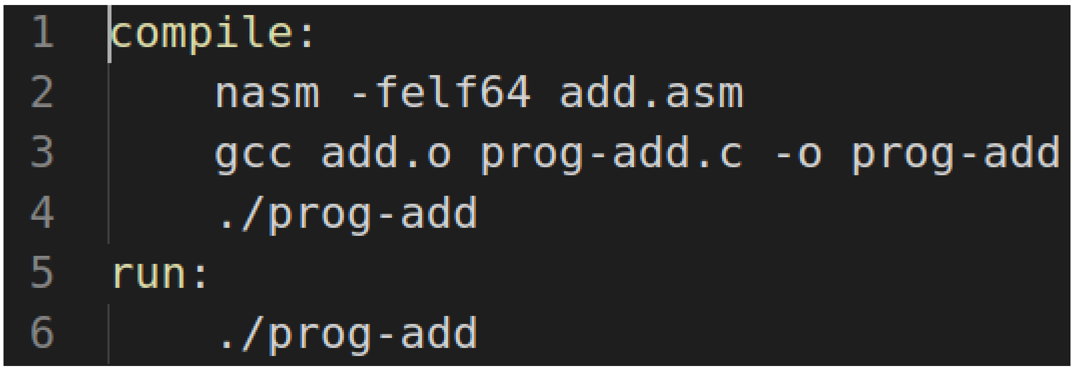
Refresher Module Assignment 0.2

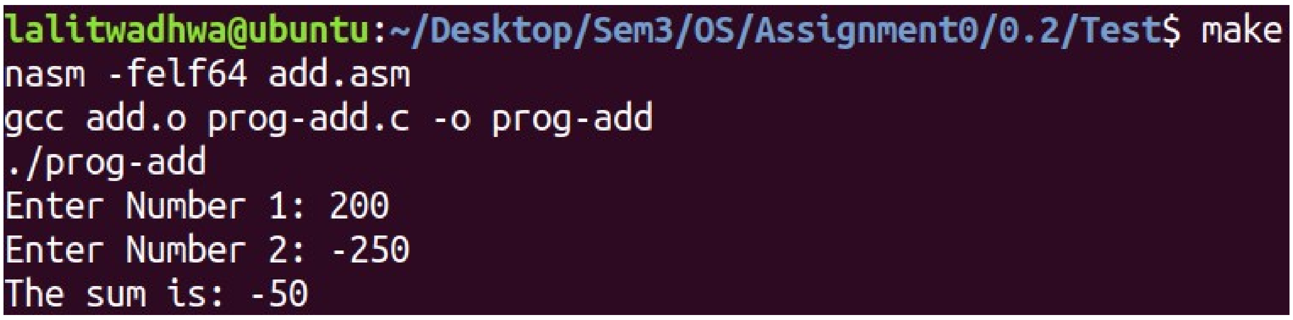
Report

The Makefile contains a total of 2 flags:



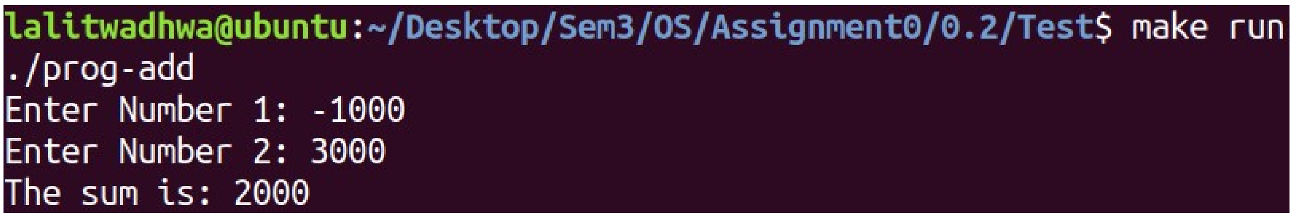
```
compile:
    nasm -felf64 add.asm
    gcc add.o prog-add.c -o prog-add
    ./prog-add
run:
    ./prog-add
```

1. **compile**
   a) This flag will run by default in the makefile if no other flag is specified
   b) This flag will:
      - first assemble the assembly code and convert it to .o format
      - then it will link the c code and the assembly code and create an executable file by the name of prog-add
      - at the end it will run this executable file
   c) The codes used under this flag are:
      nasm -felf64 add.asm
      gcc add.o prog-add.c -o prog-add
      ./prog-add



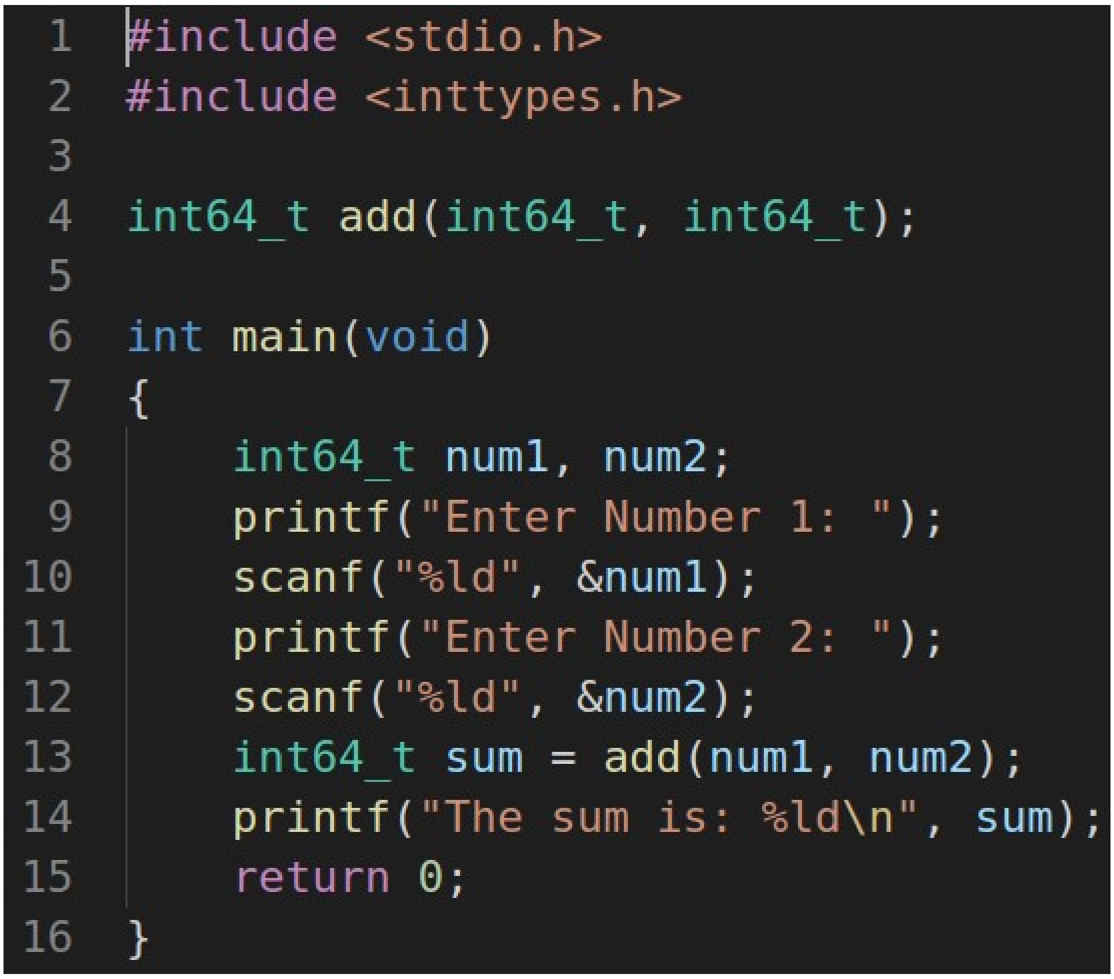
```
lalitwadhwa@ubuntu:~/Desktop/Sem3/OS/Assignment0/0.2/Test$ make
nasm -felf64 add.asm
gcc add.o prog-add.c -o prog-add
./prog-add
Enter Number 1: 200
Enter Number 2: -250
The sum is: -50
```

2. **run**
   a) This flag will be used to run the already existing executable file prog-add
   b) This flag can only be used once the default flag is used as the default flag is the one which creates the executable file prog-add.
   c) The code used under this flag is:
      ./prog-add



```
lalitwadhwa@ubuntu:~/Desktop/Sem3/OS/Assignment0/0.2/Test$ make run
./prog-add
Enter Number 1: -1000
Enter Number 2: 3000
The sum is: 2000
```

The c program consists of the following code:

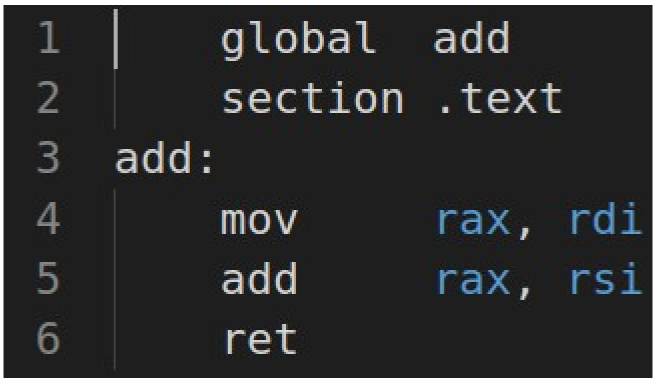


```c
#include <stdio.h>
#include <inttypes.h>

int64_t add(int64_t, int64_t);

int main(void)
{
    int64_t num1, num2;
    printf("Enter Number 1: ");
    scanf("%ld", &num1);
    printf("Enter Number 2: ");
    scanf("%ld", &num2);
    int64_t sum = add(num1, num2);
    printf("The sum is: %ld\n", sum);
    return 0;
}
```

The assembly program consists of the following code:



```asm
    global  add
    section .text
add:
    mov     rax, rdi
    add     rax, rsi
    ret
```

- The C program has defined a function with the prototype of “int64_t add(int64_t, int64_t);” before the main function. This tells the c program that this function is defined somewhere and can be accessed.
- This function in turn is defined in the file add.asm
- After the two files are linked and compiled, in the execution stage, whenever this function is called by the C program it finds the function definition from the add.asm file(which has by this time got converted into add.o for linking) and executes accordingly.
- The function prototype specifies that the return type of the function is int64_t and it takes in two arguments both of which have the type int64_t.

- Whenever the function is called, the arguments passed in the function called are stored in the rdi and rsi registers of x86_64 respectively.
- The value stored in rdi is moved to the rax register in line 4.
- in line 5 the value stored in rax is added to the value stored in rsi and the sum in turn is stored in rax register.
- The ret command is then used to return the value stored in the rax register.
- The returned value is stored in the “sum” variable (line 13, c code)
- This value is in turn printed using the printf statement in the c program.