

1.1. After -Og compile and objdump -d test1.exe (from <main>):

```
sub  $0x28,%rsp
call 140001520 <__main>
mov  $0x0,%eax
add  $0x28,%rsp
ret
```

1.2. After -O0 compile and objdump:

```
push %rbp
mov  %rsp,%rbp
sub  $0x30,%rsp
call 140001530 <__main>
movl $0x7,-0x4(%rbp)      # int x = 7
movl $0x2,-0x8(%rbp)      # int y = 2
mov  -0x4(%rbp),%eax
imul -0x8(%rbp),%eax
mov  %eax,-0xc(%rbp)
mov  $0x0,%eax            # int z
add  $0x30,%rsp           # z = y*x
pop  %rbp
ret
```

1.3. Changing the optimization flag from -Og to -O0 increased the length of our disassembly instructions. -Og is used for faster code while -O0 is used for debugging.

2.1. After -Og compile and objdump -d test2.exe (from <main>):

```
sub  $0x28,%rsp
call 140001520 <__main>
mov  $0x0,%eax
add  $0x28,%rsp
ret
```

2.2 After -O0 compile and objdump:

```
push %rbp
mov  %rsp,%rbp
sub  $0x30,%rsp
call 140001540 <__main>
movl $0x7,-0x4(%rbp)      # int x = 7
movl $0x0,-0x8(%rbp)      # int y = 0
mov  -0x4(%rbp),%eax
```

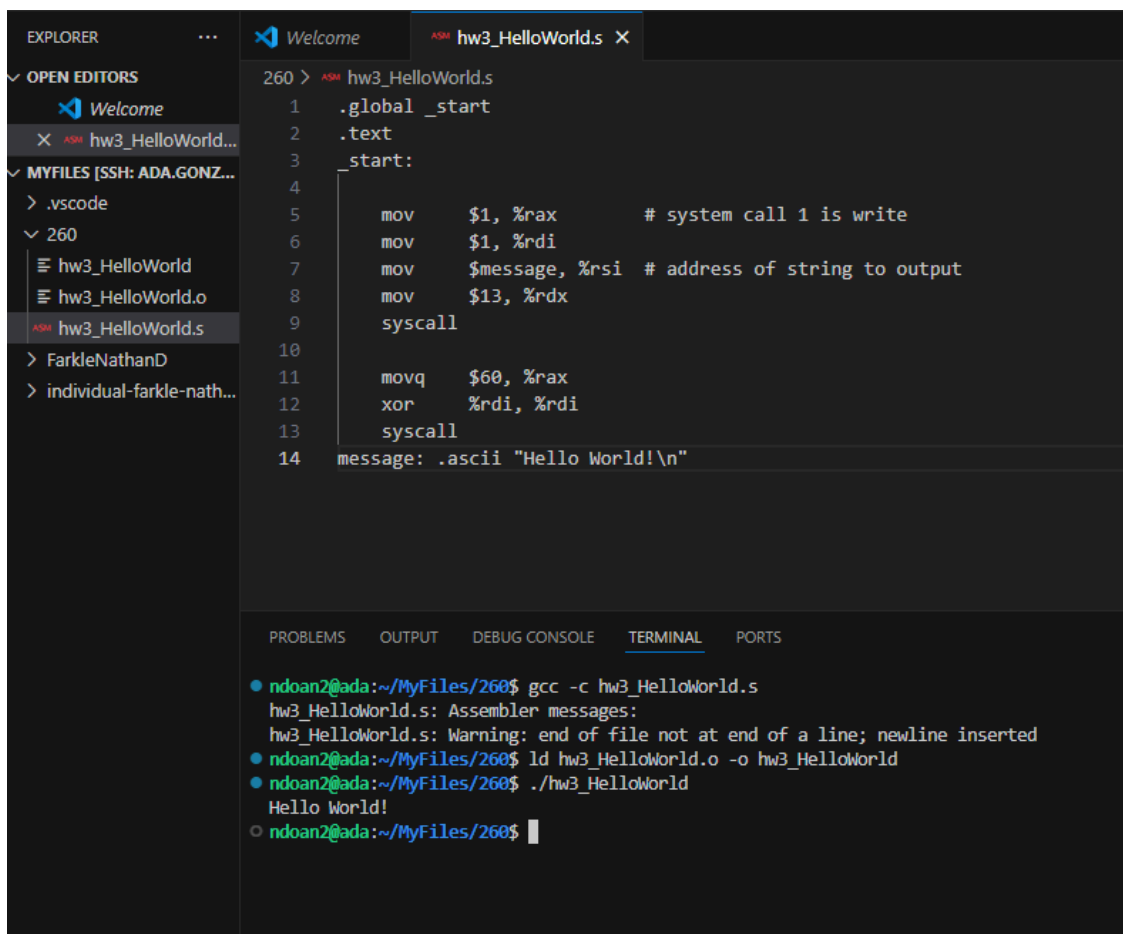
```

and    $0x1,%eax
test   %eax,%eax           # if x % 2
jne     14000147e <main+0x2e> # if true
movl    $0x2,-0x8(%rbp)     # set y = 2 & return
jmp     140001485 <main+0x35> # if false
movl    $0x1,-0x8(%rbp)     # set y = 1 & return
mov     $0x0,%eax
add     $0x30,%rsp
pop     %rbp
ret

```

2.3. Like my previous observations, -O0 increases the length of the disassembly instructions. We can confirm this by looking at the outputs. With -O0, no optimization. The compiler will keep all variables and the if-else structure. Even though y isn't used, all steps are kept. With -Og, the compiler removes dead code. Since y is assigned but not used, the entire if-else is eliminated. The assembly would just return 0 without any of the variable assignments or condition checks. -Og helps eliminate any unnecessary operations.

3.1 Assembly code, using Linux ada.gonzaga server (Originally on Windows 11)



The screenshot shows a VS Code editor with the file `hw3_HelloWorld.s` open. The assembly code is as follows:

```

260 > asm hw3_HelloWorld.s
1  .global _start
2  .text
3  _start:
4
5      mov     $1, %rax      # system call 1 is write
6      mov     $1, %rdi
7      mov     $message, %rsi # address of string to output
8      mov     $13, %rdx
9      syscall
10
11     movq    $60, %rax
12     xor     %rdi, %rdi
13     syscall
14     message: .ascii "Hello World!\n"

```

The terminal window shows the following commands and output:

```

ndloan2@ada:~/MyFiles/260$ gcc -c hw3_HelloWorld.s
hw3_HelloWorld.s: Assembler messages:
hw3_HelloWorld.s: Warning: end of file not at end of a line; newline inserted
ndloan2@ada:~/MyFiles/260$ ld hw3_HelloWorld.o -o hw3_HelloWorld
ndloan2@ada:~/MyFiles/260$ ./hw3_HelloWorld
Hello World!
ndloan2@ada:~/MyFiles/260$

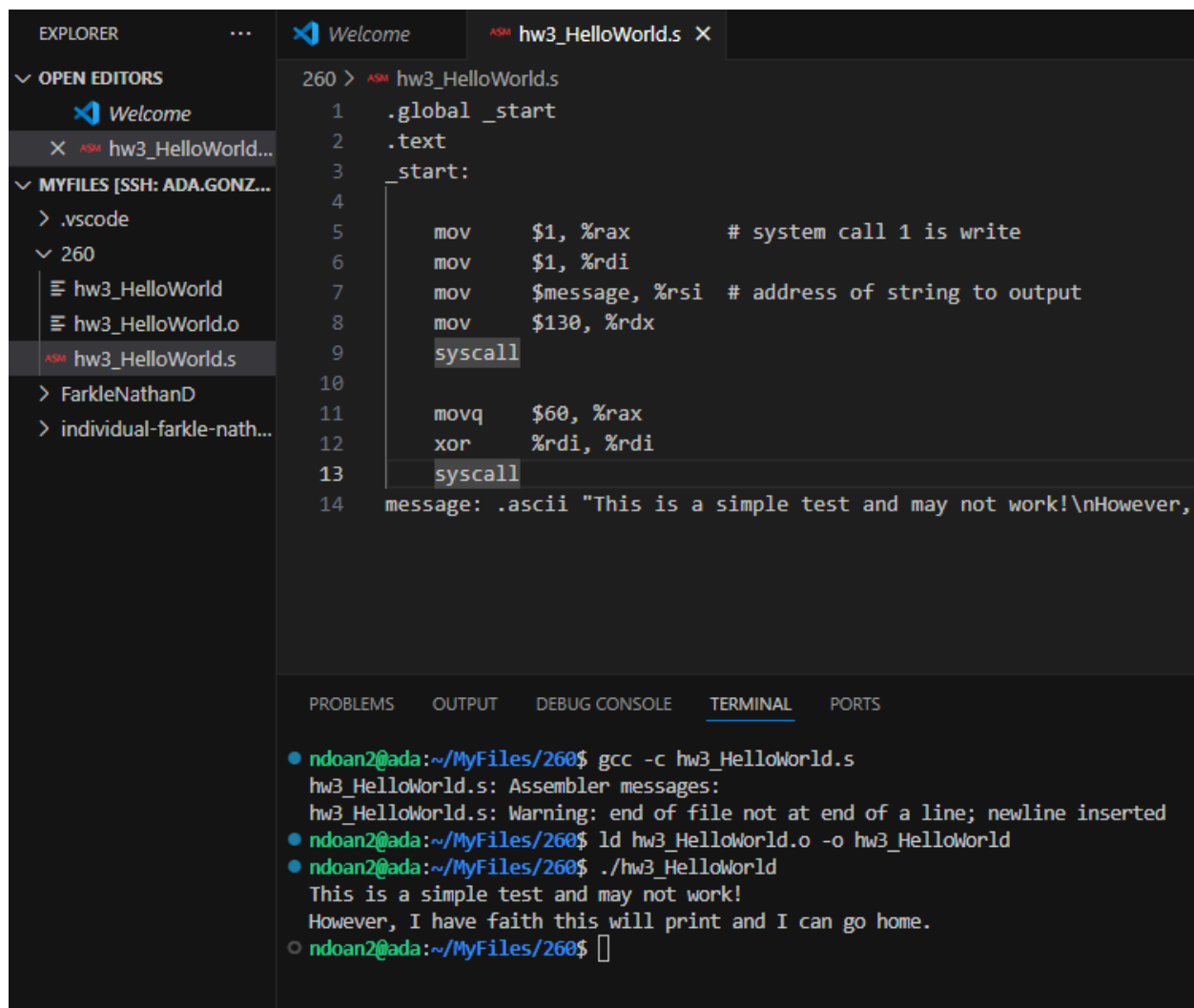
```

3.2 Modifying the original HelloWorld.s file

To print out a larger string, I had to modify the `mov $num, %rdx` line. The 'num' indicates any integer and that will create an empty string based on the size of the integer. For my message to work, 13 characters was not enough so I multiplied it by 10.

Explanation of %rax, %rdi, %rsi, and %rdx

The uses of `%rax`, `%rdi`, `%rsi`, and `%rdx` are used to call some sort of registry in the computer. I assume these registries allow the computer to access memory or information about the task they're supposed to perform. I tried changing the `%rsi` to `%rdi` on the `$message` line, and vice versa – however that failed to compile the executable. So, the `%rax`, `%rdi`, `%rsi`, and `%rdx` does affect the code.



The screenshot shows the Visual Studio Code editor with the file `hw3_HelloWorld.s` open. The code is written in assembly and includes comments. The terminal at the bottom shows the compilation and execution process.

```

260 > ASM hw3_HelloWorld.s
1  .global _start
2  .text
3  _start:
4
5      mov     $1, %rax        # system call 1 is write
6      mov     $1, %rdi
7      mov     $message, %rsi # address of string to output
8      mov     $130, %rdx
9      syscall
10
11     movq    $60, %rax
12     xor     %rdi, %rdi
13     syscall
14 message: .ascii "This is a simple test and may not work!\nHowever,

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

● ndoan2@ada:~/MyFiles/260$ gcc -c hw3_HelloWorld.s
hw3_HelloWorld.s: Assembler messages:
hw3_HelloWorld.s: Warning: end of file not at end of a line; newline inserted
● ndoan2@ada:~/MyFiles/260$ ld hw3_HelloWorld.o -o hw3_HelloWorld
● ndoan2@ada:~/MyFiles/260$ ./hw3_HelloWorld
This is a simple test and may not work!
However, I have faith this will print and I can go home.
○ ndoan2@ada:~/MyFiles/260$ 

```