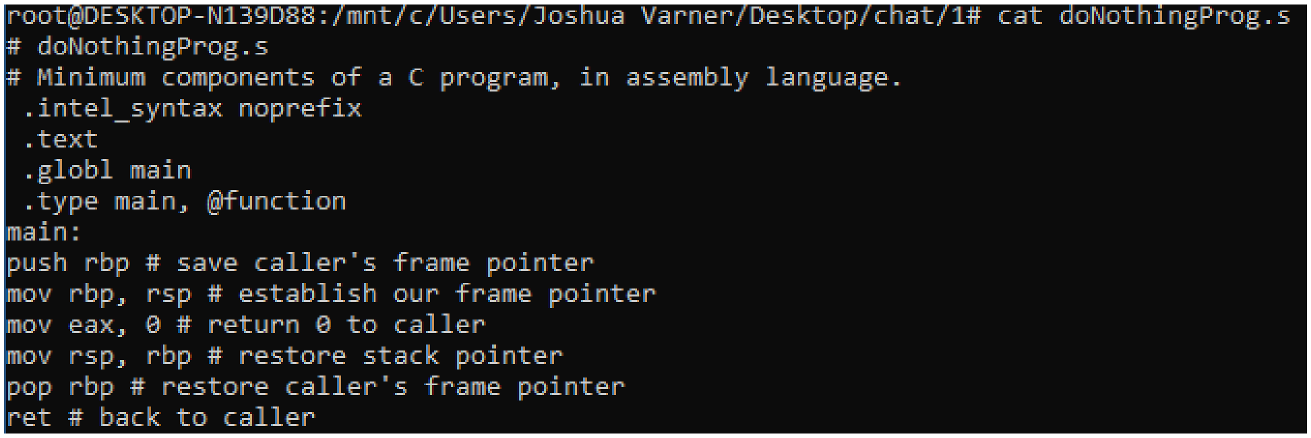
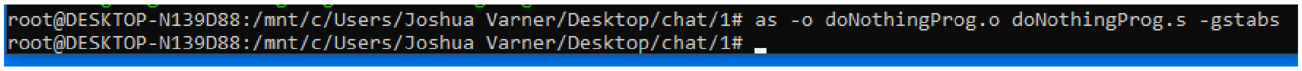
DoNothingProg.s



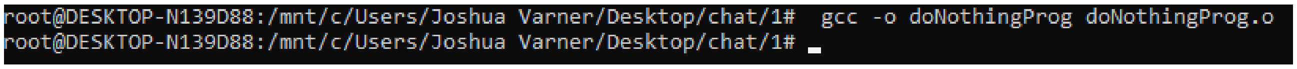
```
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# cat doNothingProg.s
# doNothingProg.s
# Minimum components of a C program, in assembly language.
 .intel_syntax noprefix
 .text
 .globl main
 .type main, @function
main:
push rbp # save caller's frame pointer
mov rbp, rsp # establish our frame pointer
mov eax, 0 # return 0 to caller
mov rsp, rbp # restore stack pointer
pop rbp # restore caller's frame pointer
ret # back to caller
```



```
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# as -o doNothingProg.o doNothingProg.s -gstabs
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1#
```

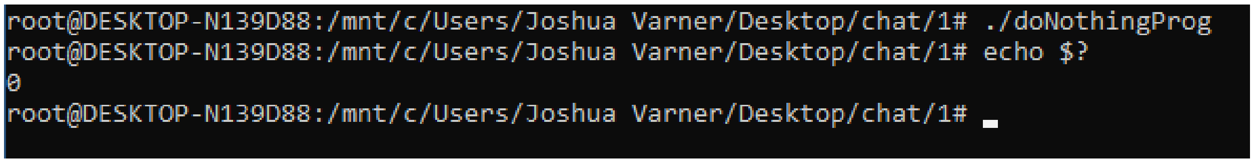
Running as -o doNothingProg.o doNothingProg.s –gstabs to compile the assembler’s Minimum components of a C program, in assembly language

-gstabs added the debugging information in the object file code for gdb



```
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1#  gcc -o doNothingProg doNothingProg.o
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1#
```

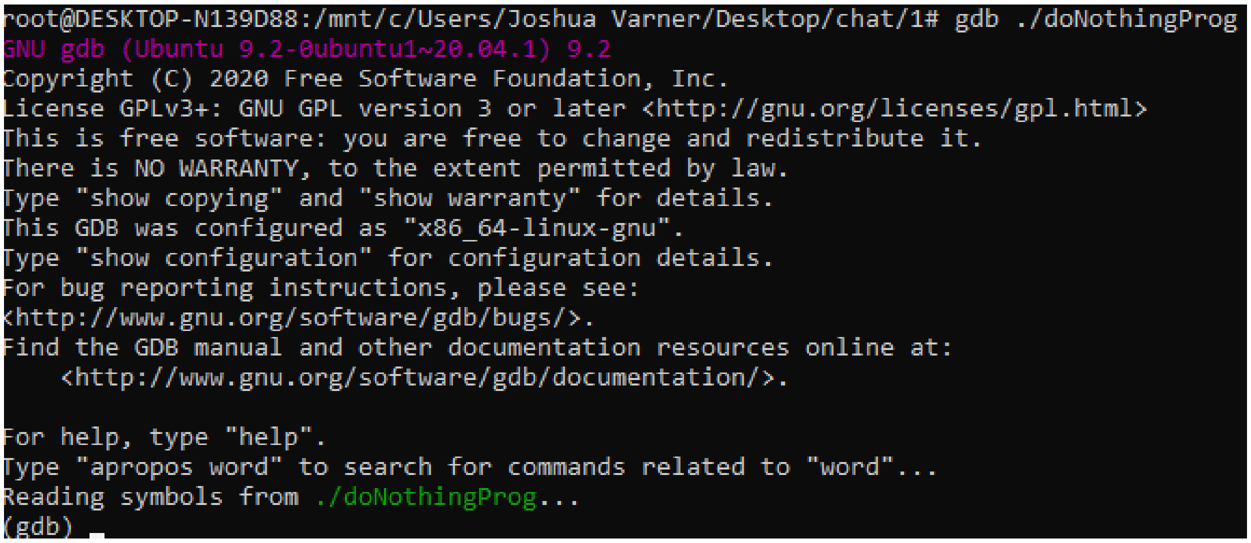
gcc -o doNothingProg doNothingProg.o



```
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# ./doNothingProg
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# echo $?
0
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1#
```

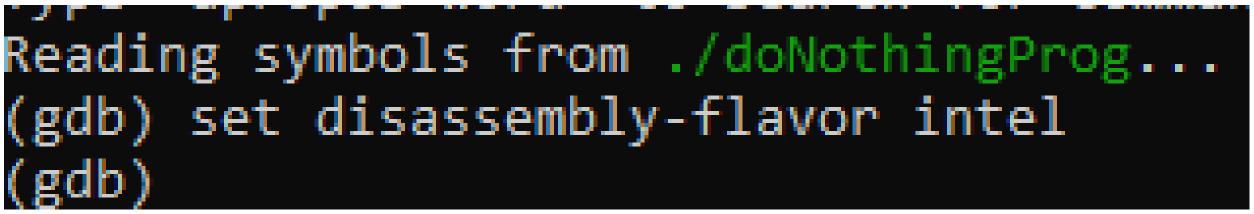
Running the executable with the epected output of 0

echo $? Prints out the value of the program from the operating system



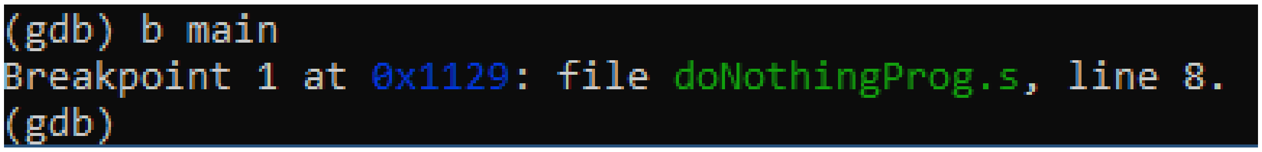
```
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# gdb ./doNothingProg
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./doNothingProg...
(gdb)
```

Adding debugging information to gdb from doNothingProg



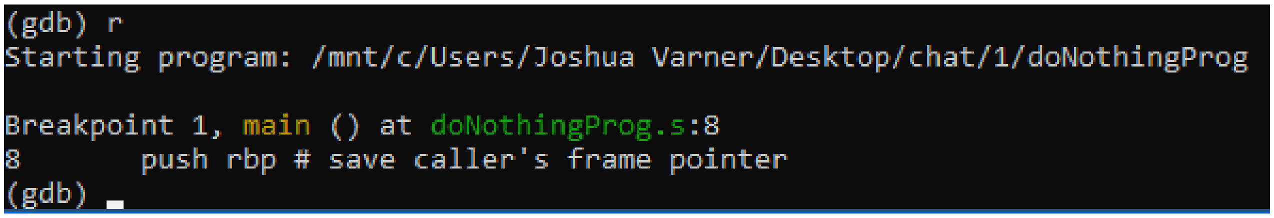
```
Reading symbols from ./doNothingProg...
(gdb) set disassembly-flavor intel
(gdb)
```

Setting the disassembly to the intel syntax



```
(gdb) b main
Breakpoint 1 at 0x1129: file doNothingProg.s, line 8.
(gdb)
```

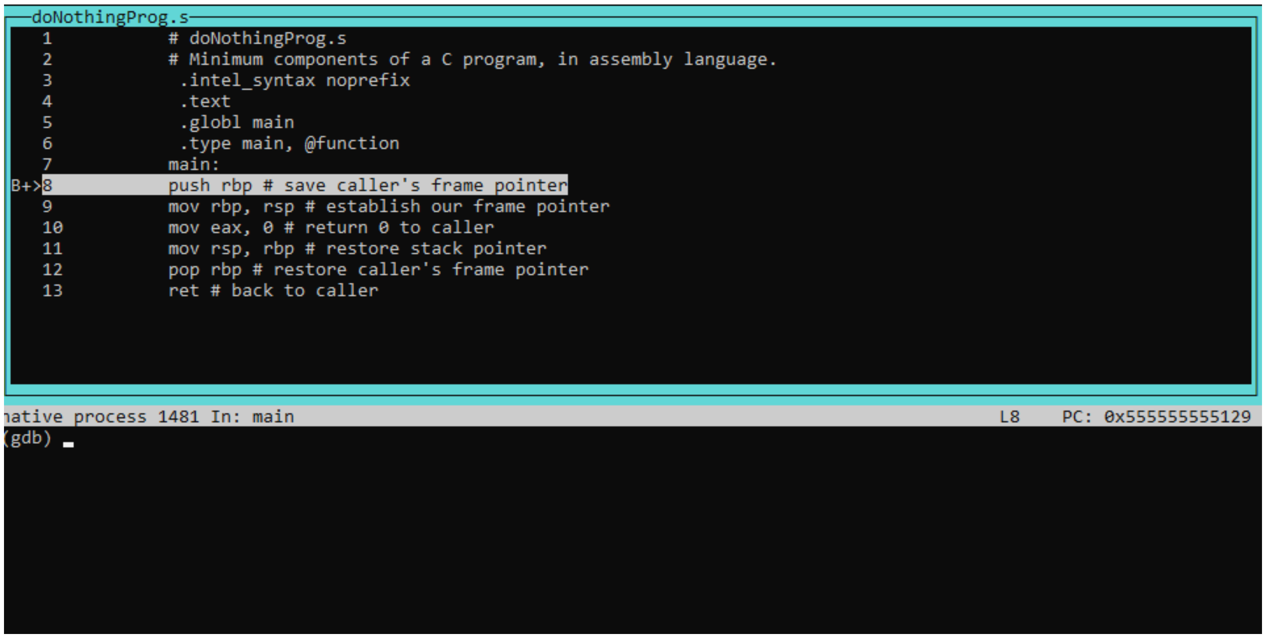
Creating a break point at memory space 0x1129 or line 8 in the doNothingProg.s file



```
(gdb) r
Starting program: /mnt/c/Users/Joshua Varner/Desktop/chat/1/doNothingProg

Breakpoint 1, main () at doNothingProg.s:8
8       push rbp # save caller's frame pointer
(gdb)
```

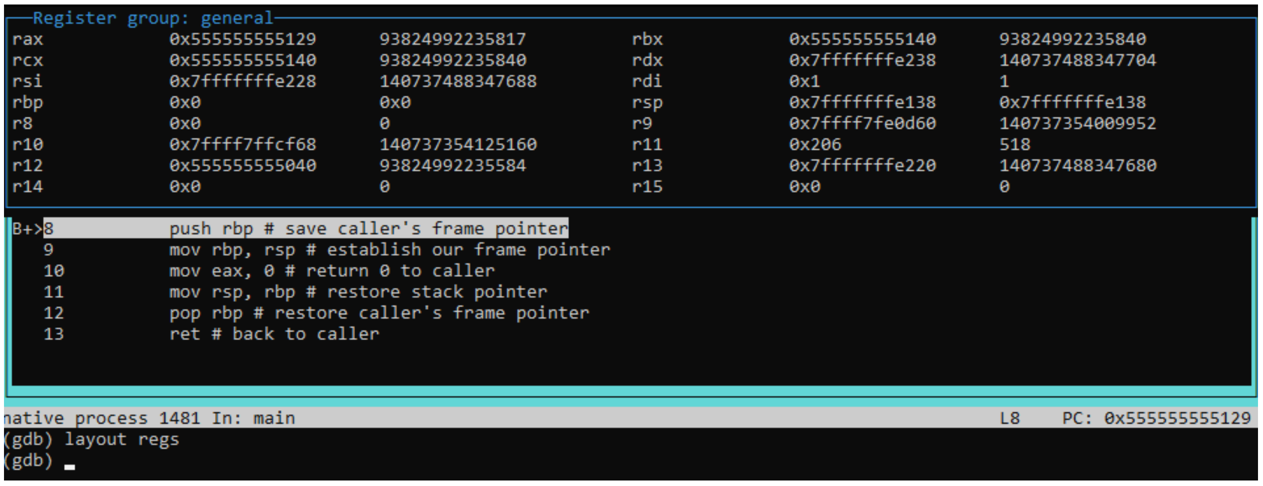
Running doNothingProg.s in gdb

Program stops at breakpoint



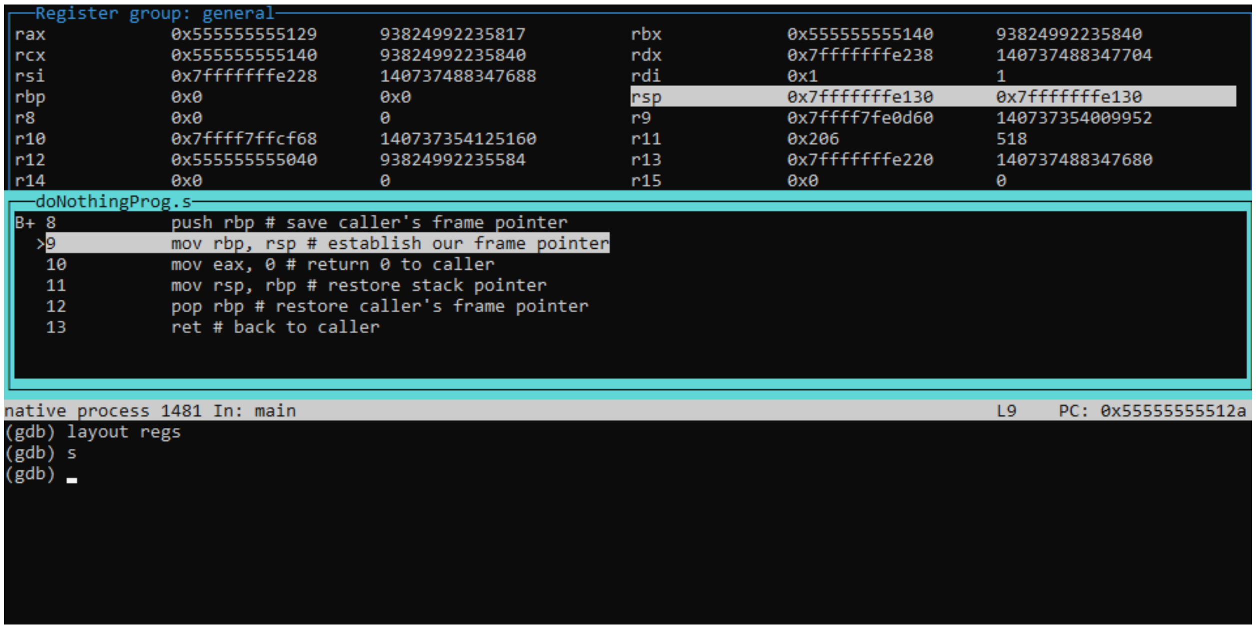
```
doNothingProg.s
          # doNothingProg.s
          # Minimum components of a C program, in assembly language.
           .intel_syntax noprefix
           .text
           .globl main
           .type main, @function
          main:
B+>8      push rbp # save caller's frame pointer
          mov rbp, rsp # establish our frame pointer
          mov eax, 0 # return 0 to caller
          mov rsp, rbp # restore stack pointer
          pop rbp # restore caller's frame pointer
          ret # back to caller

native process 1481 In: main                                          L8    PC: 0x555555555129
(gdb)
```

TUI enables the src display



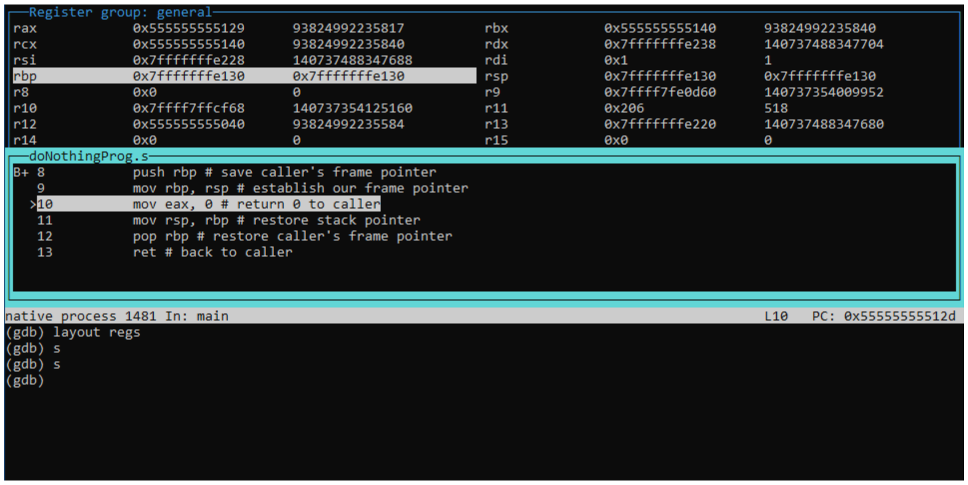
```
Register group: general
rax            0x555555555129      93824992235817           rbx            0x555555555140      93824992235840
rcx            0x555555555140      93824992235840           rdx            0x7fffffffe238      140737488347704
rsi            0x7fffffffe228      140737488347688          rdi            0x1                 1
rbp            0x0                 0x0                      rsp            0x7fffffffe138      0x7fffffffe138
r8             0x0                 0                        r9             0x7ffff7fe0d60      140737354009952
r10            0x7ffff7ffcf68      140737354125160          r11            0x206               518
r12            0x555555555040      93824992235584           r13            0x7fffffffe220      140737488347680
r14            0x0                 0                        r15            0x0                 0

B+>8      push rbp # save caller's frame pointer
          mov rbp, rsp # establish our frame pointer
          mov eax, 0 # return 0 to caller
          mov rsp, rbp # restore stack pointer
          pop rbp # restore caller's frame pointer
          ret # back to caller

native process 1481 In: main                                          L8    PC: 0x555555555129
(gdb) layout regs
(gdb)
```

Layout regs registers



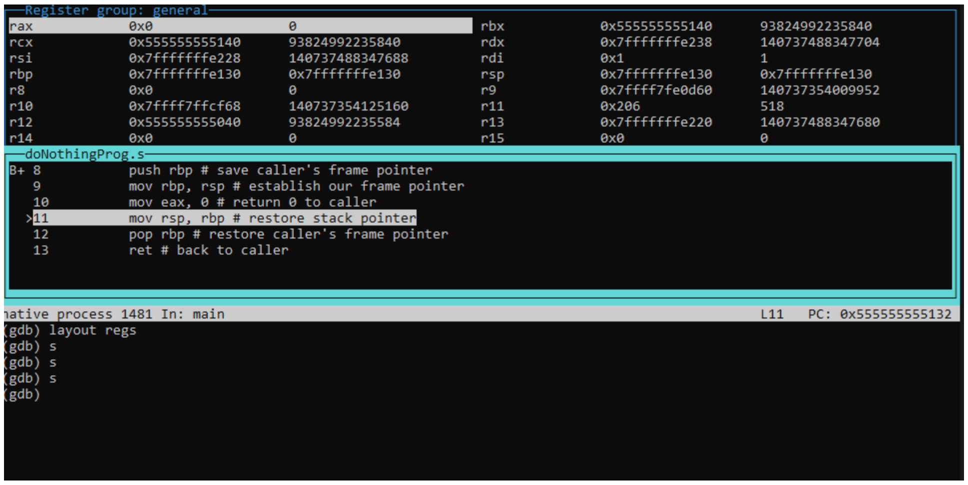
```
Register group: general
rax            0x555555555129      93824992235817       rbx            0x555555555140      93824992235840
rcx            0x555555555140      93824992235840       rdx            0x7fffffffe238      140737488347704
rsi            0x7fffffffe228      140737488347688      rdi            0x1                 1
rbp            0x0                 0x0                  rsp            0x7fffffffe130      0x7fffffffe130
r8             0x0                 0                    r9             0x7ffff7fe0d60      140737354009952
r10            0x7ffff7ffcf68      140737354125160      r11            0x206               518
r12            0x555555555040      93824992235584       r13            0x7fffffffe220      140737488347680
r14            0x0                 0                    r15            0x0                 0
doNothingProg.s
B+ 8           push rbp # save caller's frame pointer
  >9           mov rbp, rsp # establish our frame pointer
   10          mov eax, 0 # return 0 to caller
   11          mov rsp, rbp # restore stack pointer
   12          pop rbp # restore caller's frame pointer
   13          ret # back to caller

native process 1481 In: main                                                          L9     PC: 0x55555555512a
(gdb) layout regs
(gdb) s
(gdb)
```

s command executes the first line of code



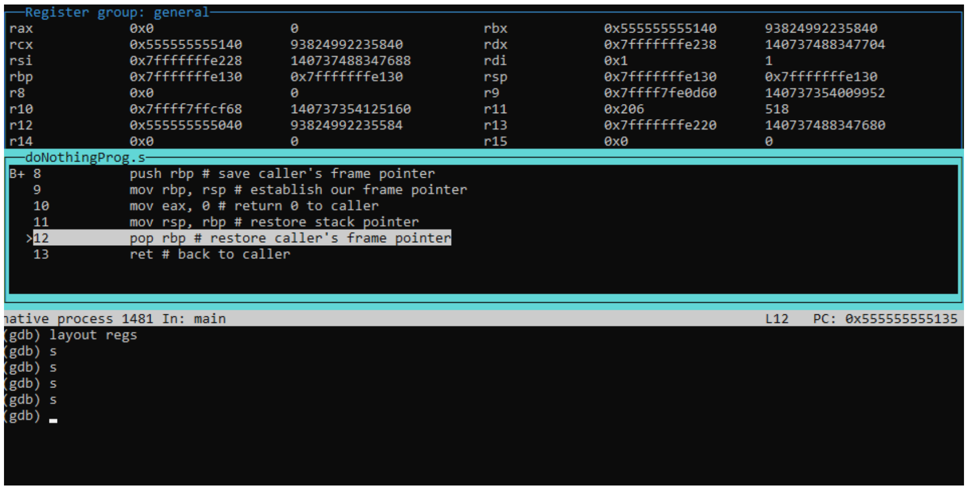
```
Register group: general
rax            0x555555555129      93824992235817       rbx            0x555555555140      93824992235840
rcx            0x555555555140      93824992235840       rdx            0x7fffffffe238      140737488347704
rsi            0x7fffffffe228      140737488347688      rdi            0x1                 1
rbp            0x7fffffffe130      0x7fffffffe130       rsp            0x7fffffffe130      0x7fffffffe130
r8             0x0                 0                    r9             0x7ffff7fe0d60      140737354009952
r10            0x7ffff7ffcf68      140737354125160      r11            0x206               518
r12            0x555555555040      93824992235584       r13            0x7fffffffe220      140737488347680
r14            0x0                 0                    r15            0x0                 0
doNothingProg.s
B+ 8           push rbp # save caller's frame pointer
   9           mov rbp, rsp # establish our frame pointer
  >10          mov eax, 0 # return 0 to caller
   11          mov rsp, rbp # restore stack pointer
   12          pop rbp # restore caller's frame pointer
   13          ret # back to caller

native process 1481 In: main                                                          L10    PC: 0x55555555512d
(gdb) layout regs
(gdb) s
(gdb) s
(gdb)
```

Going through the steps of code with the s command



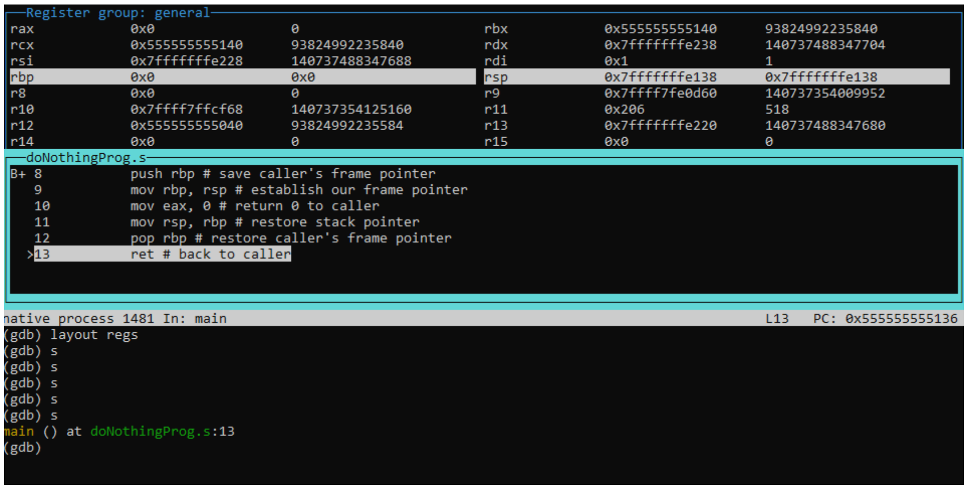
```
─Register group: general─────────────────────────────────────────────────────────────────────────────
rax            0x0                 0                     rbx            0x555555555140      93824992235840
rcx            0x555555555140      93824992235840        rdx            0x7fffffffe238      140737488347704
rsi            0x7fffffffe228      140737488347688       rdi            0x1                 1
rbp            0x7fffffffe130      0x7fffffffe130        rsp            0x7fffffffe130      0x7fffffffe130
r8             0x0                 0                     r9             0x7ffff7fe0d60      140737354009952
r10            0x7ffff7ffcf68      140737354125160       r11            0x206               518
r12            0x555555555040      93824992235584        r13            0x7fffffffe220      140737488347680
r14            0x0                 0                     r15            0x0                 0
─doNothingProg.s─────────────────────────────────────────────────────────────────────────────────────
B+ 8            push rbp # save caller's frame pointer
   9            mov rbp, rsp # establish our frame pointer
   10           mov eax, 0 # return 0 to caller
  >11           mov rsp, rbp # restore stack pointer
   12           pop rbp # restore caller's frame pointer
   13           ret # back to caller

ative process 1481 In: main                                                         L11    PC: 0x555555555132
(gdb) layout regs
(gdb) s
(gdb) s
(gdb) s
(gdb)
```

Rax register



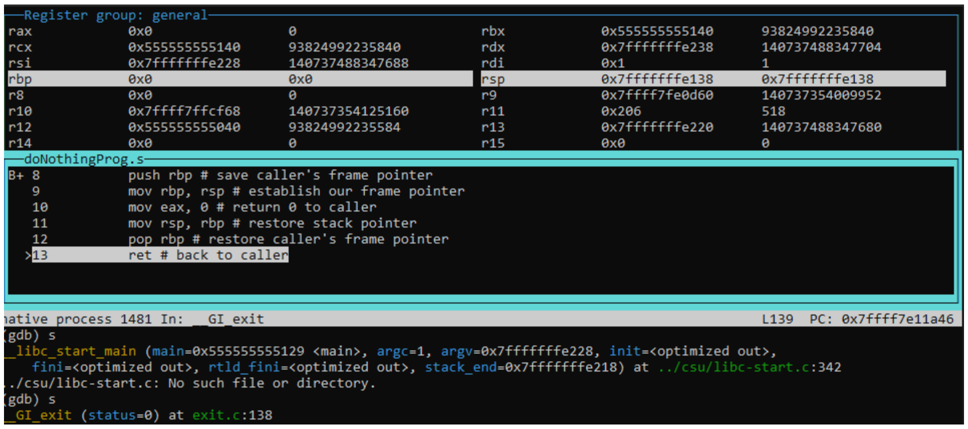
```
─Register group: general─────────────────────────────────────────────────────────────────────────────
rax            0x0                 0                     rbx            0x555555555140      93824992235840
rcx            0x555555555140      93824992235840        rdx            0x7fffffffe238      140737488347704
rsi            0x7fffffffe228      140737488347688       rdi            0x1                 1
rbp            0x7fffffffe130      0x7fffffffe130        rsp            0x7fffffffe130      0x7fffffffe130
r8             0x0                 0                     r9             0x7ffff7fe0d60      140737354009952
r10            0x7ffff7ffcf68      140737354125160       r11            0x206               518
r12            0x555555555040      93824992235584        r13            0x7fffffffe220      140737488347680
r14            0x0                 0                     r15            0x0                 0
─doNothingProg.s─────────────────────────────────────────────────────────────────────────────────────
B+ 8            push rbp # save caller's frame pointer
   9            mov rbp, rsp # establish our frame pointer
   10           mov eax, 0 # return 0 to caller
   11           mov rsp, rbp # restore stack pointer
  >12           pop rbp # restore caller's frame pointer
   13           ret # back to caller

ative process 1481 In: main                                                         L12    PC: 0x555555555135
(gdb) layout regs
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) _
```

Rbp is restored



```
─Register group: general─────────────────────────────────────────────────────────────────────────────
rax            0x0                 0                     rbx            0x555555555140      93824992235840
rcx            0x555555555140      93824992235840        rdx            0x7fffffffe238      140737488347704
rsi            0x7fffffffe228      140737488347688       rdi            0x1                 1
rbp            0x0                 0x0                   rsp            0x7fffffffe138      0x7fffffffe138
r8             0x0                 0                     r9             0x7ffff7fe0d60      140737354009952
r10            0x7ffff7ffcf68      140737354125160       r11            0x206               518
r12            0x555555555040      93824992235584        r13            0x7fffffffe220      140737488347680
r14            0x0                 0                     r15            0x0                 0
─doNothingProg.s─────────────────────────────────────────────────────────────────────────────────────
B+ 8            push rbp # save caller's frame pointer
   9            mov rbp, rsp # establish our frame pointer
   10           mov eax, 0 # return 0 to caller
   11           mov rsp, rbp # restore stack pointer
   12           pop rbp # restore caller's frame pointer
  >13           ret # back to caller

ative process 1481 In: main                                                         L13    PC: 0x555555555136
(gdb) layout regs
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s
main () at doNothingProg.s:13
(gdb)
```

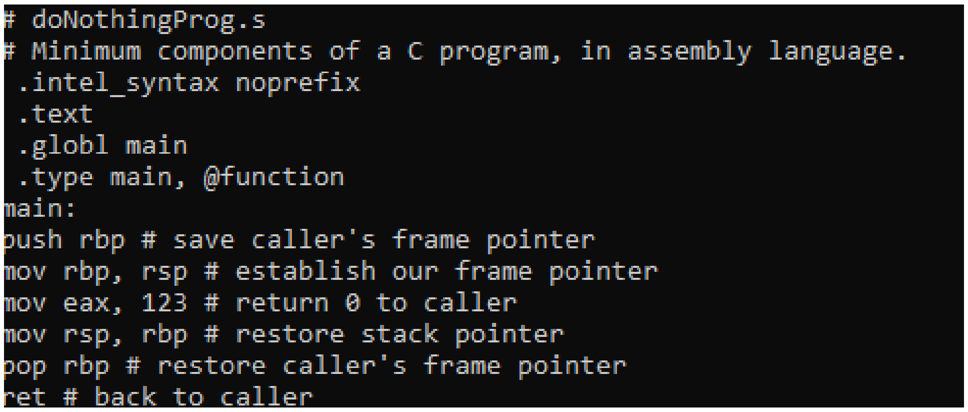
Rbp register



```
—Register group: general—
rax            0x0                 0                    rbx            0x555555555140      93824992235840
rcx            0x555555555140      93824992235840       rdx            0x7fffffffe238      140737488347704
rsi            0x7fffffffe228      140737488347688      rdi            0x1                 1
rbp            0x0                 0x0                  rsp            0x7fffffffe138      0x7fffffffe138
r8             0x0                 0                    r9             0x7ffff7fe0d60      140737354009952
r10            0x7ffff7ffcf68      140737354125160      r11            0x206               518
r12            0x555555555040      93824992235584       r13            0x7fffffffe220      140737488347680
r14            0x0                 0                    r15            0x0                 0
—doNothingProg.s—
B+             push rbp # save caller's frame pointer
               mov rbp, rsp # establish our frame pointer
               mov eax, 0 # return 0 to caller
               mov rsp, rbp # restore stack pointer
               pop rbp # restore caller's frame pointer
  >            ret # back to caller

native process 1481 In: __GI_exit                                                  L139   PC: 0x7ffff7e11a46
(gdb) s
__libc_start_main (main=0x555555555129 <main>, argc=1, argv=0x7fffffffe228, init=<optimized out>,
    fini=<optimized out>, rtld_fini=<optimized out>, stack_end=0x7fffffffe218) at ../csu/libc-start.c:342
../csu/libc-start.c: No such file or directory.
(gdb) s
__GI_exit (status=0) at exit.c:138
```

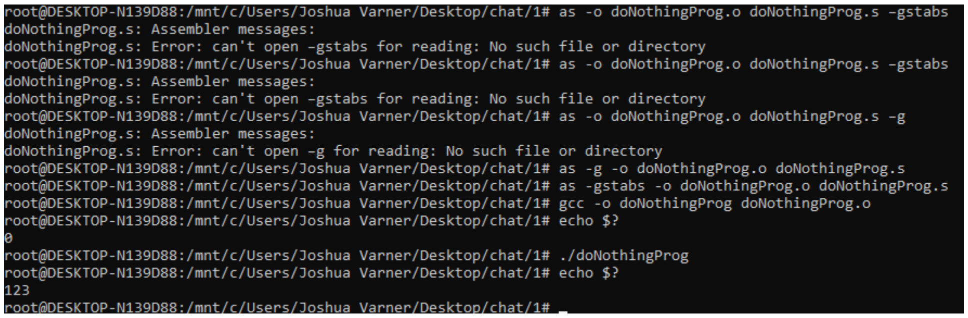
Ret does not use anymore registers

Program is completed



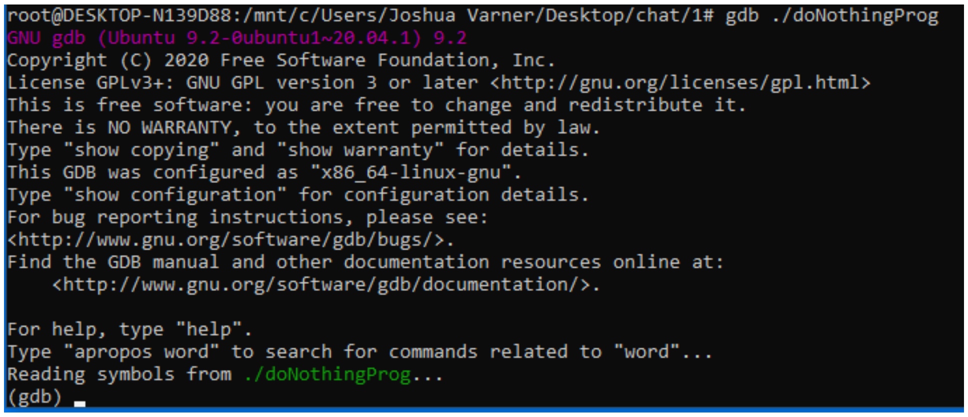
```
# doNothingProg.s
# Minimum components of a C program, in assembly language.
 .intel_syntax noprefix
 .text
 .globl main
 .type main, @function
main:
push rbp # save caller's frame pointer
mov rbp, rsp # establish our frame pointer
mov eax, 123 # return 0 to caller
mov rsp, rbp # restore stack pointer
pop rbp # restore caller's frame pointer
ret # back to caller
```

Changing the value to 123 instead of 0



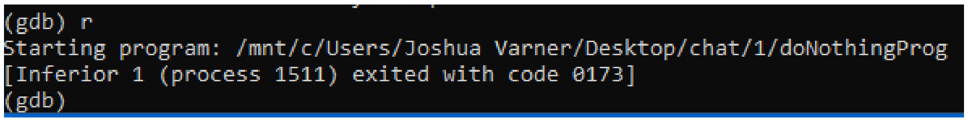
```
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# as -o doNothingProg.o doNothingProg.s -gstabs
doNothingProg.s: Assembler messages:
doNothingProg.s: Error: can't open -gstabs for reading: No such file or directory
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# as -o doNothingProg.o doNothingProg.s -gstabs
doNothingProg.s: Assembler messages:
doNothingProg.s: Error: can't open -gstabs for reading: No such file or directory
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# as -o doNothingProg.o doNothingProg.s -g
doNothingProg.s: Assembler messages:
doNothingProg.s: Error: can't open -g for reading: No such file or directory
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# as -g -o doNothingProg.o doNothingProg.s
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# as -gstabs -o doNothingProg.o doNothingProg.s
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# gcc -o doNothingProg doNothingProg.o
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# echo $?
0
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# ./doNothingProg
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# echo $?
123
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# _
```

Program returns value of 123



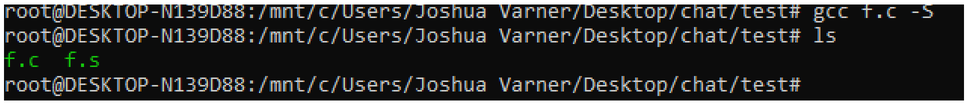
```
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/1# gdb ./doNothingProg
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./doNothingProg...
(gdb)
```

Running the newly compiled executable into gdb



```
(gdb) r
Starting program: /mnt/c/Users/Joshua Varner/Desktop/chat/1/doNothingProg
[Inferior 1 (process 1511) exited with code 0173]
(gdb)
```

exited with code 0173



```
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/test# gcc f.c -S
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/test# ls
f.c  f.s
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/test#
```

f.c to assembly with gcc –S

Outputted this code in f.s

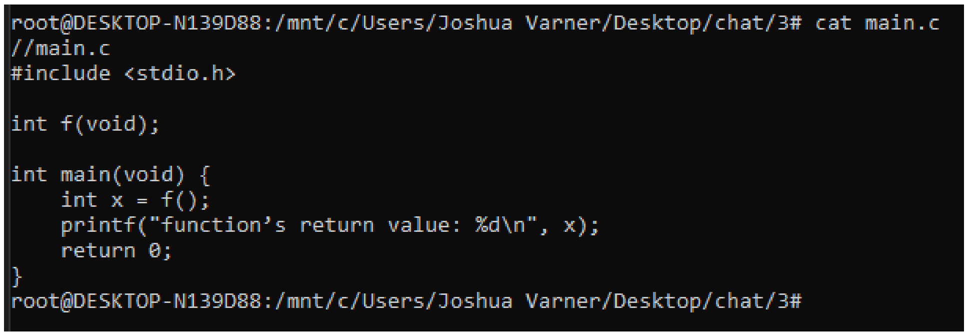
```
.file	"f.c"
.text
.globl	f
.type	f, @function
f:
.LFB0:
.cfi_startproc
endbr64
pushq	%rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq	%rsp, %rbp
.cfi_def_cfa_register 6
```

```
movl    $0, %eax
popq    %rbp
.cfi_def_cfa 7, 8
ret
.cfi_endproc
.LFE0:
.size    f, .-f
.ident   "GCC: (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0"
.section            .note.GNU-stack,"",@progbits
.section            .note.gnu.property,"a"
.align 8
.long     1f - 0f
.long     4f - 1f
.long     5
0:
.string   "GNU"
1:
.align 8
.long     0xc0000002
.long     3f - 2f
2:
.long     0x3
3:
.align 8
4:
```

My f.s code in comparison with gcc

```
.globl f
```

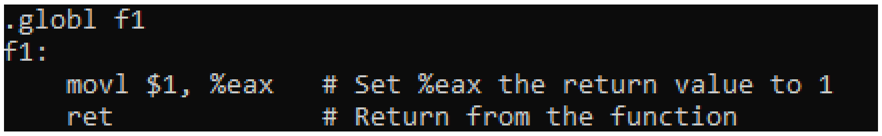
```
f:
    xorl %eax, %eax   # Set %eax (the return value) to 0
    ret               # Return from the function
```



```
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/3# cat main.c
//main.c
#include <stdio.h>

int f(void);

int main(void) {
    int x = f();
    printf("function’s return value: %d\n", x);
    return 0;
}
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/3#
```

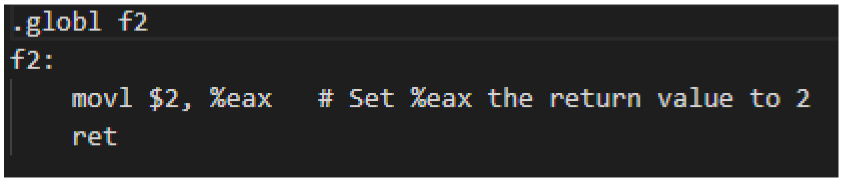
My main.c function calls the function from the assembly code. printf returns the function's value (returns 0)

Three assembly functions that return an int value stored in eax



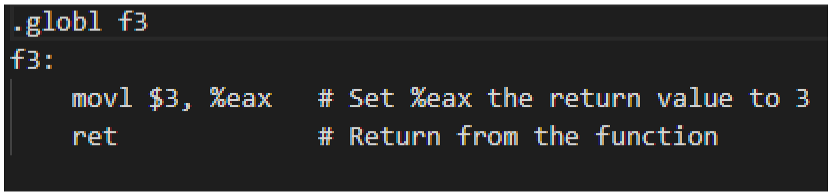
```
.globl f1
f1:
    movl $1, %eax    # Set %eax the return value to 1
    ret              # Return from the function
```

Returns 1



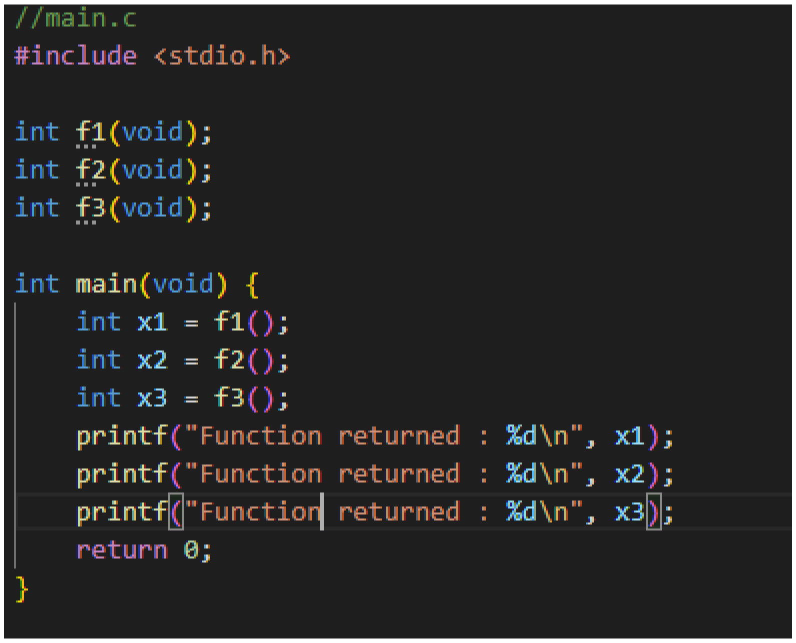
```
.globl f2
f2:
    movl $2, %eax   # Set %eax the return value to 2
    ret
```

Returns 2



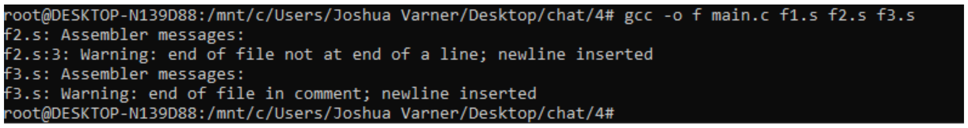
```
.globl f3
f3:
    movl $3, %eax   # Set %eax the return value to 3
    ret             # Return from the function
```

Returns 3



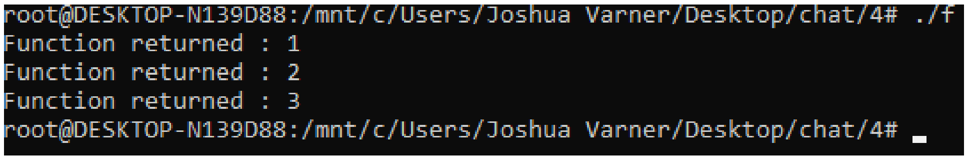
```c
//main.c
#include <stdio.h>

int f1(void);
int f2(void);
int f3(void);

int main(void) {
    int x1 = f1();
    int x2 = f2();
    int x3 = f3();
    printf("Function returned : %d\n", x1);
    printf("Function returned : %d\n", x2);
    printf("Function returned : %d\n", x3);
    return 0;
}
```

My main.c that calls the assembly functions f1,f2,f3 and prints the return integer values stored in x1,x2,x3 with printf



```
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/4# gcc -o f main.c f1.s f2.s f3.s
f2.s: Assembler messages:
f2.s:3: Warning: end of file not at end of a line; newline inserted
f3.s: Assembler messages:
f3.s: Warning: end of file in comment; newline inserted
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/4#
```

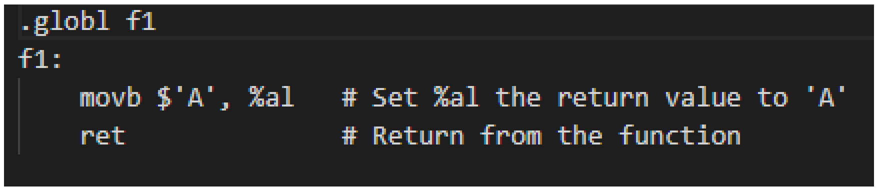
gcc -o f main.c f1.s f2.s f3.s complies the assembly code with the main.c



```
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/4# ./f
Function returned : 1
Function returned : 2
Function returned : 3
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/4#
```

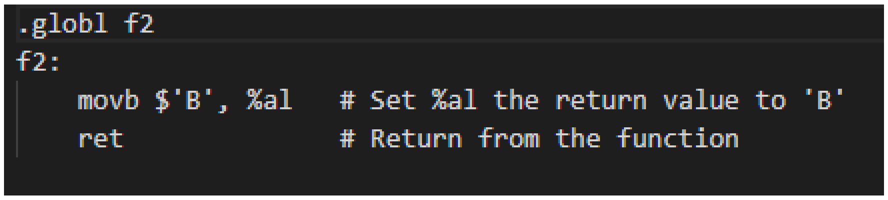
Program

Assembly functions that return char values A,B,C



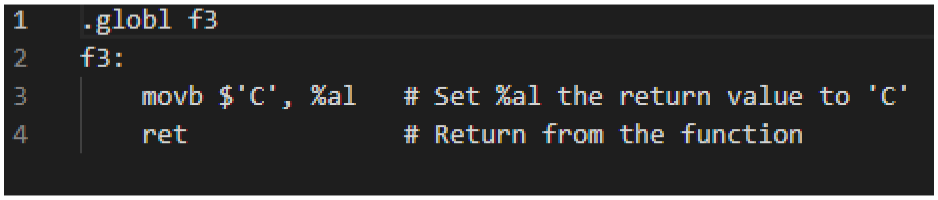
```asm
.globl f1
f1:
    movb $'A', %al    # Set %al the return value to 'A'
    ret               # Return from the function
```

Returns A



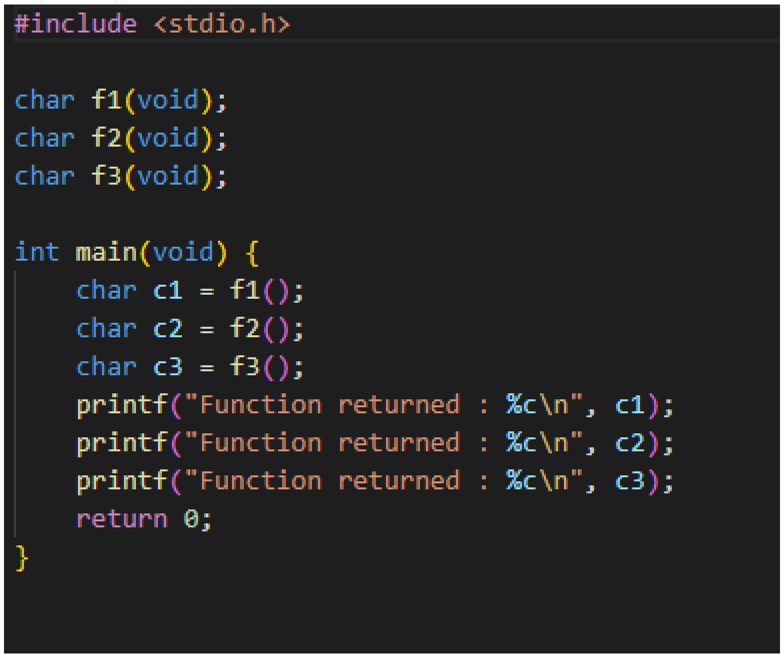
```asm
.globl f2
f2:
    movb $'B', %al    # Set %al the return value to 'B'
    ret               # Return from the function
```

Returns B



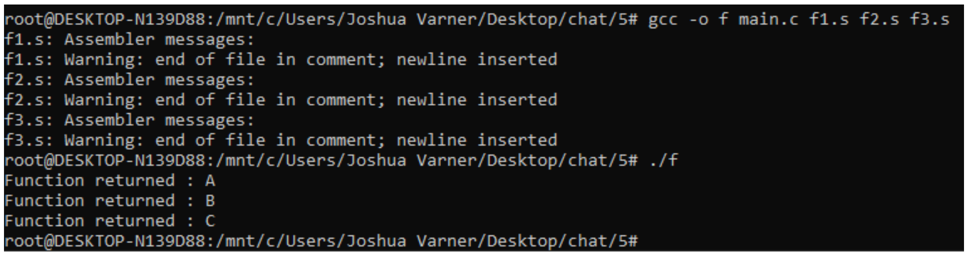
```
.globl f3
f3:
    movb $'C', %al    # Set %al the return value to 'C'
    ret               # Return from the function
```

Returns C



```c
#include <stdio.h>

char f1(void);
char f2(void);
char f3(void);

int main(void) {
    char c1 = f1();
    char c2 = f2();
    char c3 = f3();
    printf("Function returned : %c\n", c1);
    printf("Function returned : %c\n", c2);
    printf("Function returned : %c\n", c3);
    return 0;
}
```

Main.c returns the values of the assembly functions f1,f2,f3 that are stored in c1,c2,c3 via printf



```
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/5# gcc -o f main.c f1.s f2.s f3.s
f1.s: Assembler messages:
f1.s: Warning: end of file in comment; newline inserted
f2.s: Assembler messages:
f2.s: Warning: end of file in comment; newline inserted
f3.s: Assembler messages:
f3.s: Warning: end of file in comment; newline inserted
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/5# ./f
Function returned : A
Function returned : B
Function returned : C
root@DESKTOP-N139D88:/mnt/c/Users/Joshua Varner/Desktop/chat/5#
```

gcc -o f main.c f1.s f2.s f3.s complies the assembly code with the main.c

Program returns and prints the 3 char functions