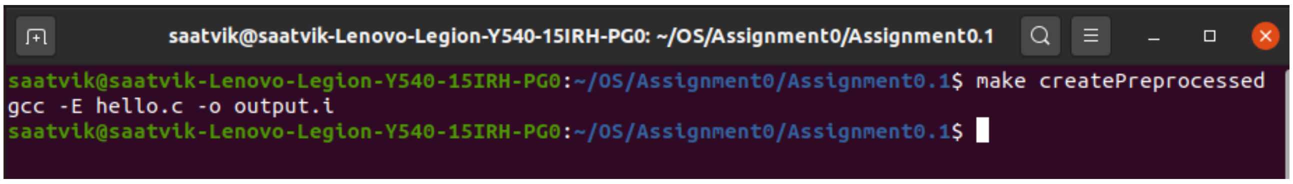
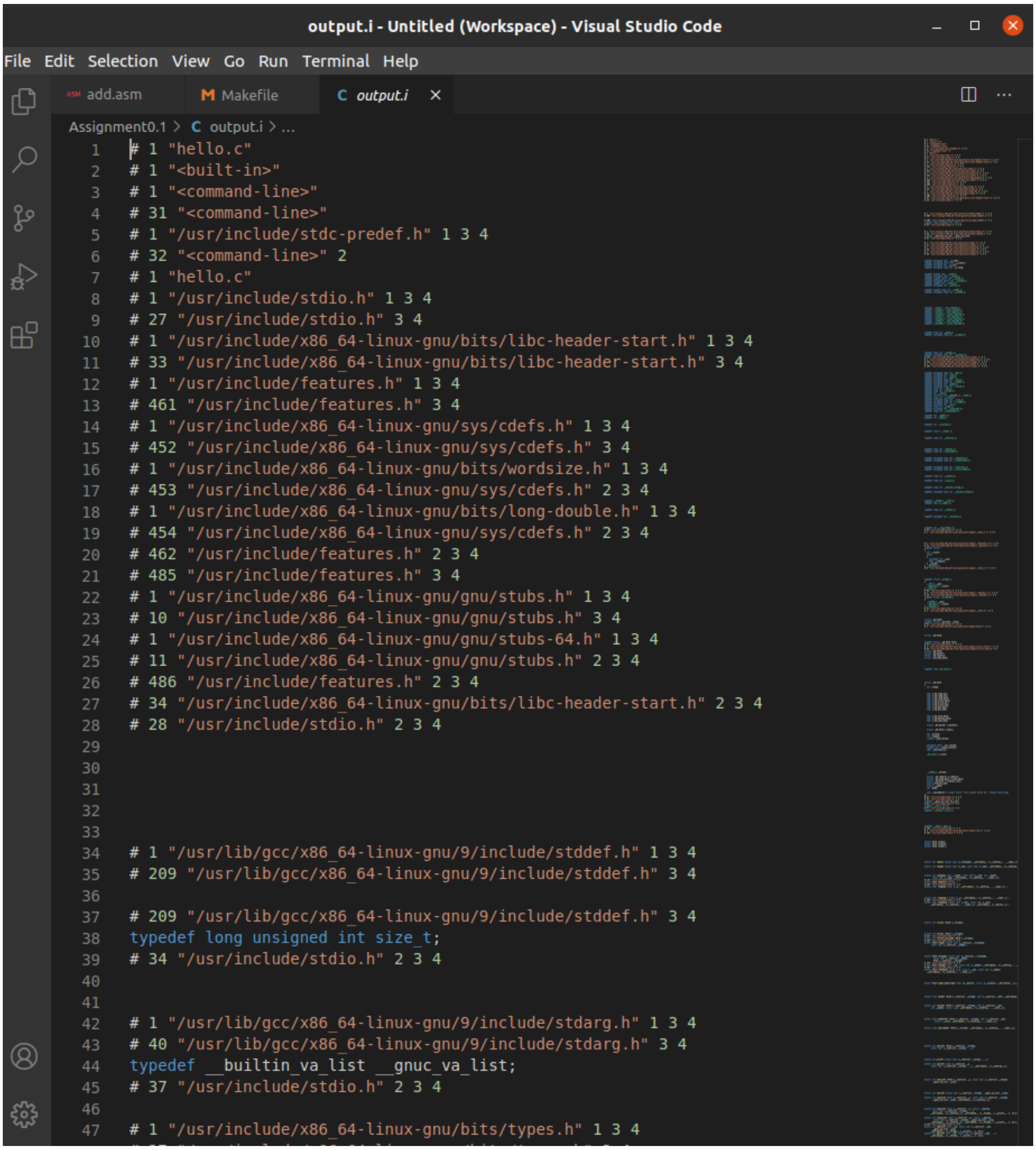
Preprocessing Phase:
To get the preprocessed file (.i) from the source code (.c) i have used the **createPreprocessed** target in the Makefile, which uses the command :

gcc -E hello.c -o output.i

After runing **make createPreprocessed** in the terminal an output.i file is made in the current directory and in the this file the header file is expanded into the functions and utilities it contains (for example printf), but the int main function is untouched and it is present at the end of the file. Also the comments are removed and the macros are expanded (as they are preceeded by # preprocessor directive sign) and typedefs are replaced


saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0: ~/OS/Assignment0/Assignment0.1

```
saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0:~/OS/Assignment0/Assignment0.1$ make createPreprocessed
gcc -E hello.c -o output.i
saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0:~/OS/Assignment0/Assignment0.1$ 
```

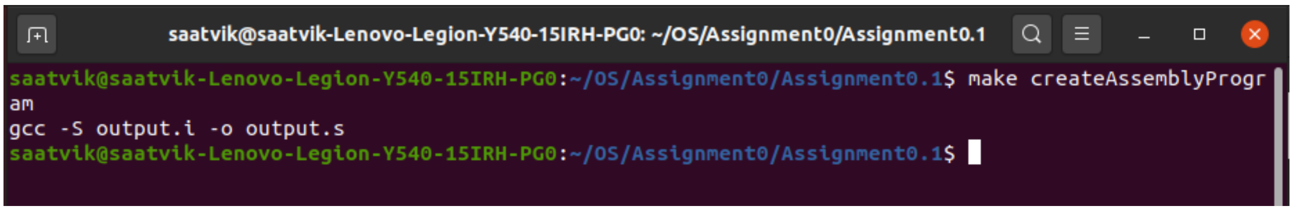

output.i - Untitled (Workspace) - Visual Studio Code
File Edit Selection View Go Run Terminal Help
add.asm
Makefile
output.i
Assignment0.1 > output.i > ...

```
# 1 "hello.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 31 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 32 "<command-line>" 2
# 1 "hello.c"
# 1 "/usr/include/stdio.h" 1 3 4
# 27 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 1 3 4
# 33 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 3 4
# 1 "/usr/include/features.h" 1 3 4
# 461 "/usr/include/features.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 1 3 4
# 452 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 453 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/long-double.h" 1 3 4
# 454 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
# 462 "/usr/include/features.h" 2 3 4
# 485 "/usr/include/features.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 1 3 4
# 10 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/gnu/stubs-64.h" 1 3 4
# 11 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 2 3 4
# 486 "/usr/include/features.h" 2 3 4
# 34 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 2 3 4
# 28 "/usr/include/stdio.h" 2 3 4





# 1 "/usr/lib/gcc/x86_64-linux-gnu/9/include/stddef.h" 1 3 4
# 209 "/usr/lib/gcc/x86_64-linux-gnu/9/include/stddef.h" 3 4

# 209 "/usr/lib/gcc/x86_64-linux-gnu/9/include/stddef.h" 3 4
typedef long unsigned int size_t;
# 34 "/usr/include/stdio.h" 2 3 4


# 1 "/usr/lib/gcc/x86_64-linux-gnu/9/include/stdarg.h" 1 3 4
# 40 "/usr/lib/gcc/x86_64-linux-gnu/9/include/stdarg.h" 3 4
typedef __builtin_va_list __gnuc_va_list;
# 37 "/usr/include/stdio.h" 2 3 4

# 1 "/usr/include/x86_64-linux-gnu/bits/types.h" 1 3 4
```

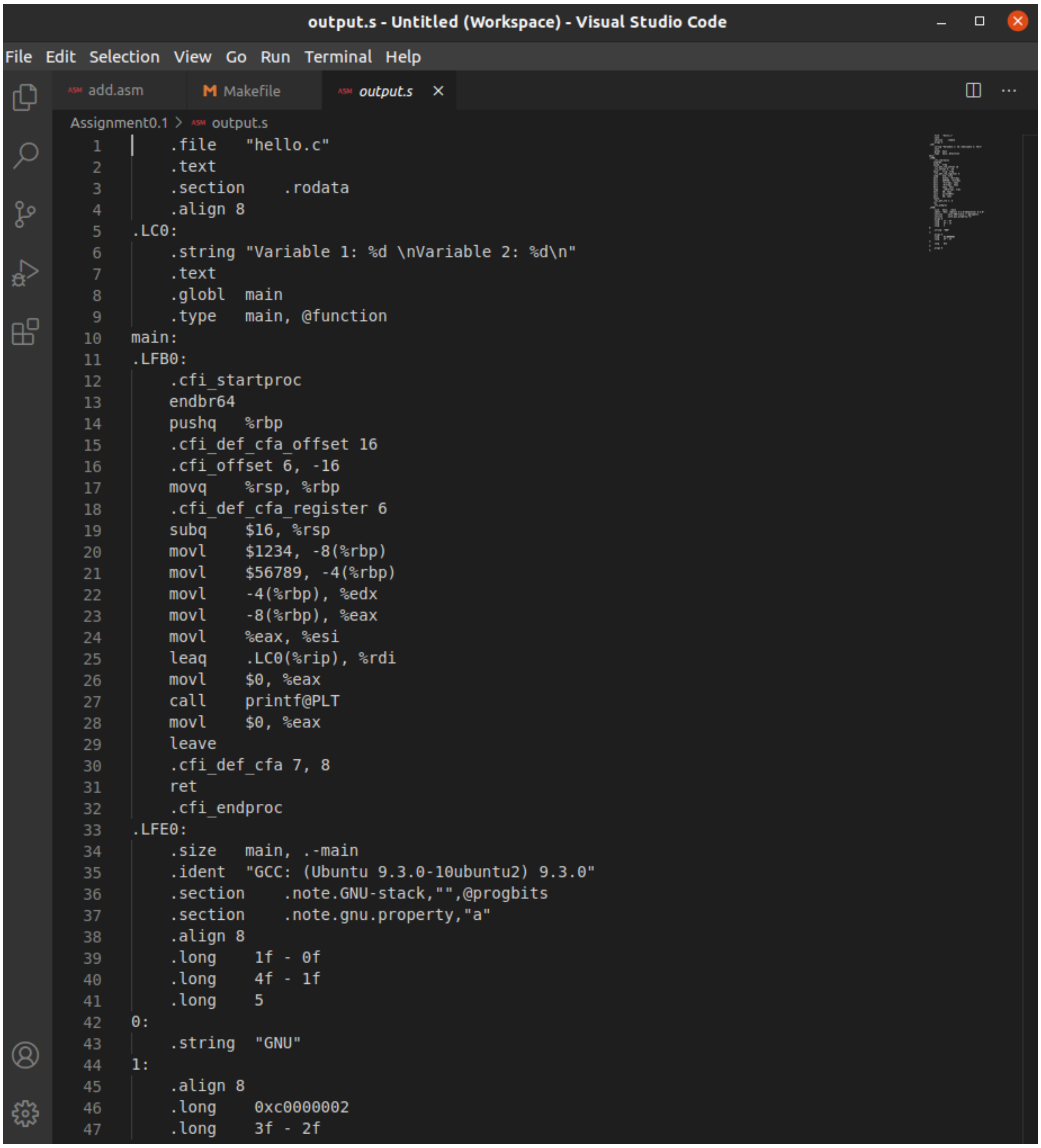
Compiling Phase:
To get the .s file from the preprocessed file (.i) i have used the c**reateAssemblyProgram** target in the Makefile, which uses the command :

```
gcc -S output.i -o output.s
```

After running **make createAssemblyProgram** in the terminal an output.s file is made in the current directory and the file contains assembly code corresponding to the preprocessed file(.i)


saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0: ~/OS/Assignment0/Assignment0.1

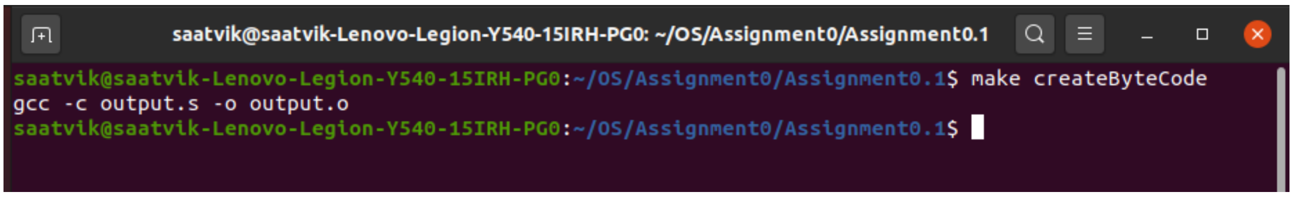
```
saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0:~/OS/Assignment0/Assignment0.1$ make createAssemblyProgr
am
gcc -S output.i -o output.s
saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0:~/OS/Assignment0/Assignment0.1$ 
```


output.s - Untitled (Workspace) - Visual Studio Code
File Edit Selection View Go Run Terminal Help
add.asm
Makefile
output.s
Assignment0.1 > output.s

```
	.file	"hello.c"
	.text
	.section	.rodata
	.align 8
.LC0:
	.string "Variable 1: %d \nVariable 2: %d\n"
	.text
	.globl	main
	.type	main, @function
main:
.LFB0:
	.cfi_startproc
	endbr64
	pushq	%rbp
	.cfi_def_cfa_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 6
	subq	$16, %rsp
	movl	$1234, -8(%rbp)
	movl	$56789, -4(%rbp)
	movl	-4(%rbp), %edx
	movl	-8(%rbp), %eax
	movl	%eax, %esi
	leaq	.LC0(%rip), %rdi
	movl	$0, %eax
	call	printf@PLT
	movl	$0, %eax
	leave
	.cfi_def_cfa 7, 8
	ret
	.cfi_endproc
.LFE0:
	.size	main, .-main
	.ident	"GCC: (Ubuntu 9.3.0-10ubuntu2) 9.3.0"
	.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
```

Assembling Phase:
To get the object (.o) file from the assembly source code (.s) I have used the **createByteCode** target in the Makefile, which uses the command :
gcc -c output.s -o output.o

After running **make createByteCode** in the terminal an output.o will be made and it would contain the machine code (object code or byte code) corresponding to the assembly language code. It is converted to object file (binary) so that the machine can understand it

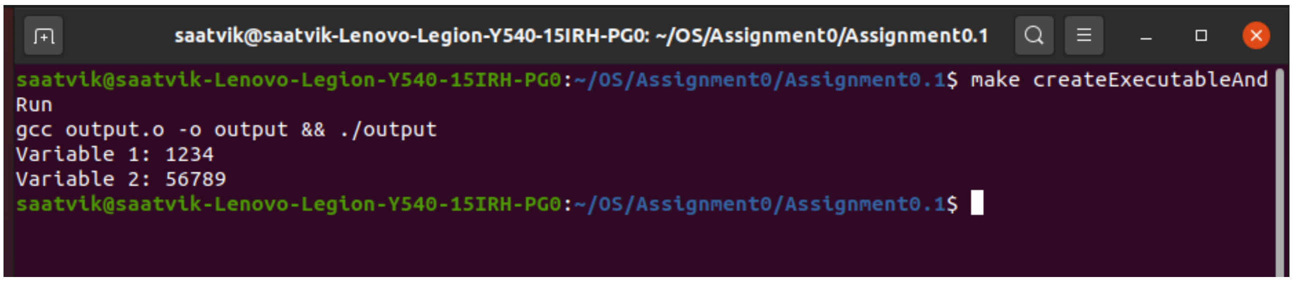


```
saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0: ~/OS/Assignment0/Assignment0.1
saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0:~/OS/Assignment0/Assignment0.1$ make createByteCode
gcc -c output.s -o output.o
saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0:~/OS/Assignment0/Assignment0.1$ 
```

The object file (output.o) wasn't opening in VSCode

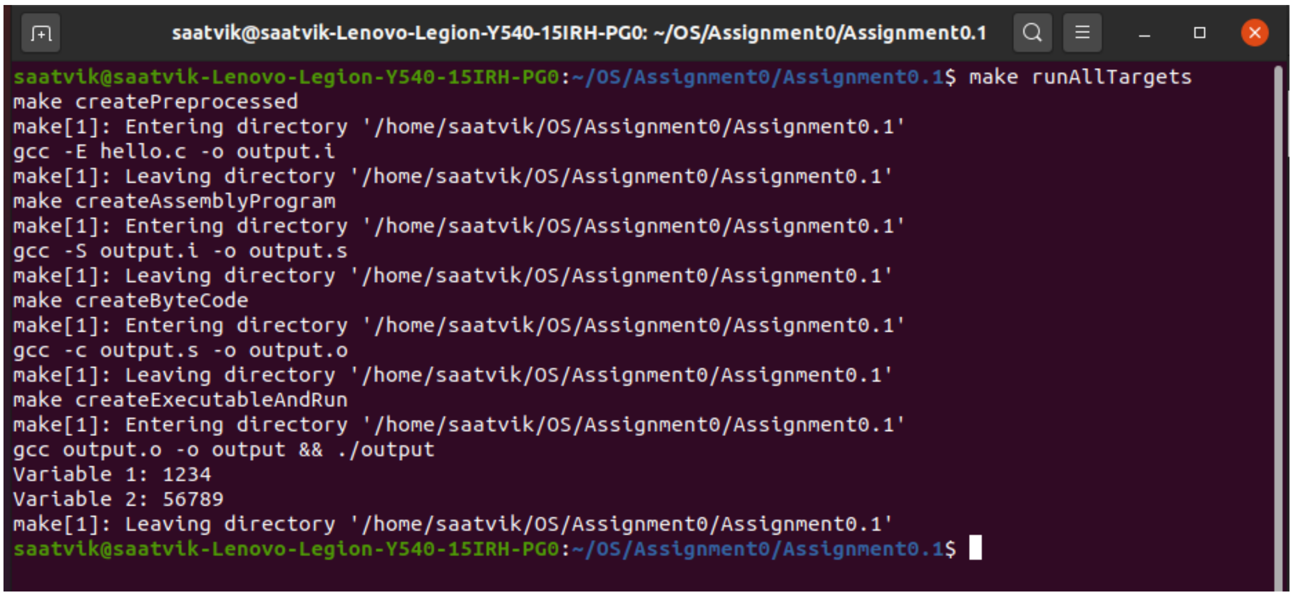
Linking Phase and Creating an executable:
To get the exectuable file (output) from the byte code i have used the **createExecutableAndRun** target in the Makefile, which uses the command:
gcc output.o -o output && ./output

After running **make createExecutableAndRun** in the terminal the linker links the object file (output.o) to all the required libraries and unresolved variables and functions (if any) and an output file is made which an executable (appears green in color when ls command is used) and the target also runs the file and thus we get our output



```
saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0: ~/OS/Assignment0/Assignment0.1
saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0:~/OS/Assignment0/Assignment0.1$ make createExecutableAnd
Run
gcc output.o -o output && ./output
Variable 1: 1234
Variable 2: 56789
saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0:~/OS/Assignment0/Assignment0.1$ 
```

In the end i have used the target **runAllTargets** which instead calls the steps of compilation in a particular order and runs the output file. It uses the commands:
make createPreprocessed
make createAssemblyProgram
make createByteCode
make createExectuableAndRun


saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0: ~/OS/Assignment0/Assignment0.1

```
saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0:~/OS/Assignment0/Assignment0.1$ make runAllTargets
make createPreprocessed
make[1]: Entering directory '/home/saatvik/OS/Assignment0/Assignment0.1'
gcc -E hello.c -o output.i
make[1]: Leaving directory '/home/saatvik/OS/Assignment0/Assignment0.1'
make createAssemblyProgram
make[1]: Entering directory '/home/saatvik/OS/Assignment0/Assignment0.1'
gcc -S output.i -o output.s
make[1]: Leaving directory '/home/saatvik/OS/Assignment0/Assignment0.1'
make createByteCode
make[1]: Entering directory '/home/saatvik/OS/Assignment0/Assignment0.1'
gcc -c output.s -o output.o
make[1]: Leaving directory '/home/saatvik/OS/Assignment0/Assignment0.1'
make createExecutableAndRun
make[1]: Entering directory '/home/saatvik/OS/Assignment0/Assignment0.1'
gcc output.o -o output && ./output
Variable 1: 1234
Variable 2: 56789
make[1]: Leaving directory '/home/saatvik/OS/Assignment0/Assignment0.1'
saatvik@saatvik-Lenovo-Legion-Y540-15IRH-PG0:~/OS/Assignment0/Assignment0.1$
```