

CS1217 – Spring 2020 — Homework 1 1

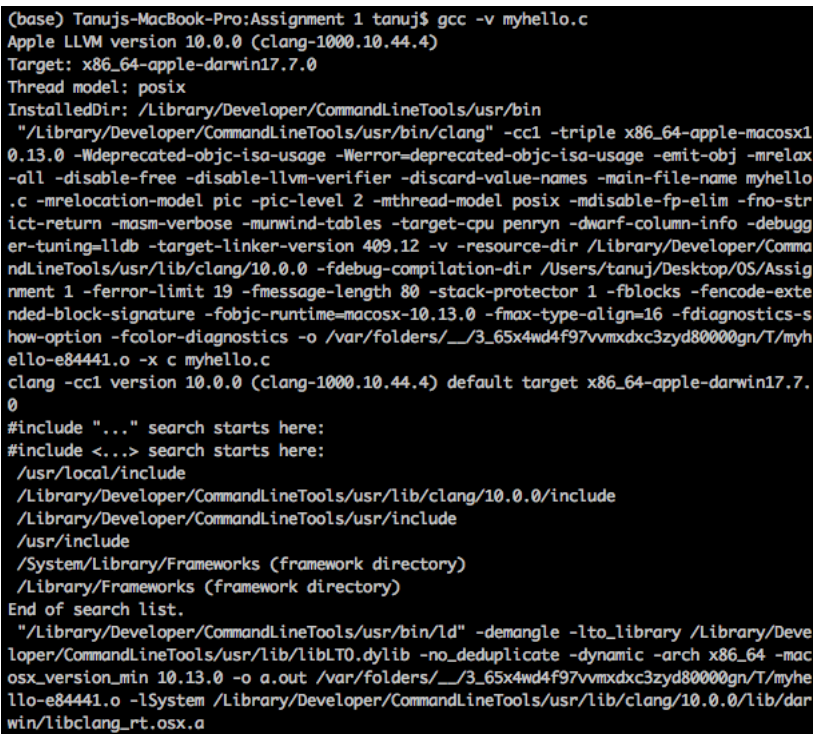
Tanuj Sood — Pratyush Kumar

Collaborators: NONE

1. First Question

- (a) The **#include** command is a directive (a language construct that specifies how a compiler would its input) to the pre-processor that tells the compiler to include a header file in the program.
- (b) The `stdio.h` file used with the **#include** function is a directive to the pre-processor that tells the compiler to include a header file **"stdio.h"** in the program. This specific header file stands for standard input output and its use is to take inputs from the keyboard and output them on a screen or monitor. Using this with the **#include** directive allows all the functionalities defined in the file within placed between the braces to be used in current file.

2. Second Question

- (a) 
- ```
(base) Tanuj-MacBook-Pro:Assignment 1 tanuj$ gcc -v myhello.c
Apple LLVM version 10.0.0 (clang-1000.10.44.4)
Target: x86_64-apple-darwin17.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
"/Library/Developer/CommandLineTools/usr/bin/clang" -cc1 -triple x86_64-apple-macosx10.13.0 -Wdeprecated-objc-isa-usage -Werror=deprecated-objc-isa-usage -emit-obj -mrelax-all -disable-free -disable-llvm-verifier -discard-value-names -main-file-name myhello.c -mrelocation-model pic -pic-level 2 -mthread-model posix -mdisable-fp-elim -fno-strict-return -masm-verbose -munwind-tables -target-cpu penryn -dwarf-column-info -debugger-tuning=lldb -target-linker-version 409.12 -v -resource-dir /Library/Developer/CommandLineTools/usr/lib/clang/10.0.0 -fdebug-compilation-dir /Users/tanuj/Desktop/OS/Assignment 1 -ferror-limit 19 -fmessage-length 80 -stack-protector 1 -fblocks -fencode-extended-block-signature -fobjc-runtime=macosx-10.13.0 -fmax-type-align=16 -fdiagnostics-show-option -fcolor-diagnostics -o /var/folders/___/3_65x4wd4f97vmmxdbc3zyd80000gn/T/myhello-e84441.o -x c myhello.c
clang -cc1 version 10.0.0 (clang-1000.10.44.4) default target x86_64-apple-darwin17.0
#include "...": search starts here:
#include <...> search starts here:
 /usr/local/include
 /Library/Developer/CommandLineTools/usr/lib/clang/10.0.0/include
 /Library/Developer/CommandLineTools/usr/include
 /usr/include
 /System/Library/Frameworks (framework directory)
 /Library/Frameworks (framework directory)
End of search list.
"/Library/Developer/CommandLineTools/usr/bin/ld" -demangle -lto_library /Library/Developer/CommandLineTools/usr/lib/libLT0.dylib -no_deduplicate -dynamic -arch x86_64 -macosx_version_min 10.13.0 -o a.out /var/folders/___/3_65x4wd4f97vmmxdbc3zyd80000gn/T/myhello-e84441.o -lSystem /Library/Developer/CommandLineTools/usr/lib/clang/10.0.0/lib/darwin/libclang_rt.osx.a
```

### 3. Third Question

- (a) The `./` symbol tells the compiler that the program file that we're trying to run is located in the current directory. The current working directory (**cwd**) is not a part of the **PATH** environment variable by default, we need to use `./` to specify where the executable is located. The **PATH** environment variables are the directories that contain executable programs that can be started without knowing and typing the whole path to the file on the command line.
- (b) If `./` is not included in path, the shell will only search for the executable in **PATH** directory, not find the file and will give an error called : **command not found**.

### 4. Fourth Question

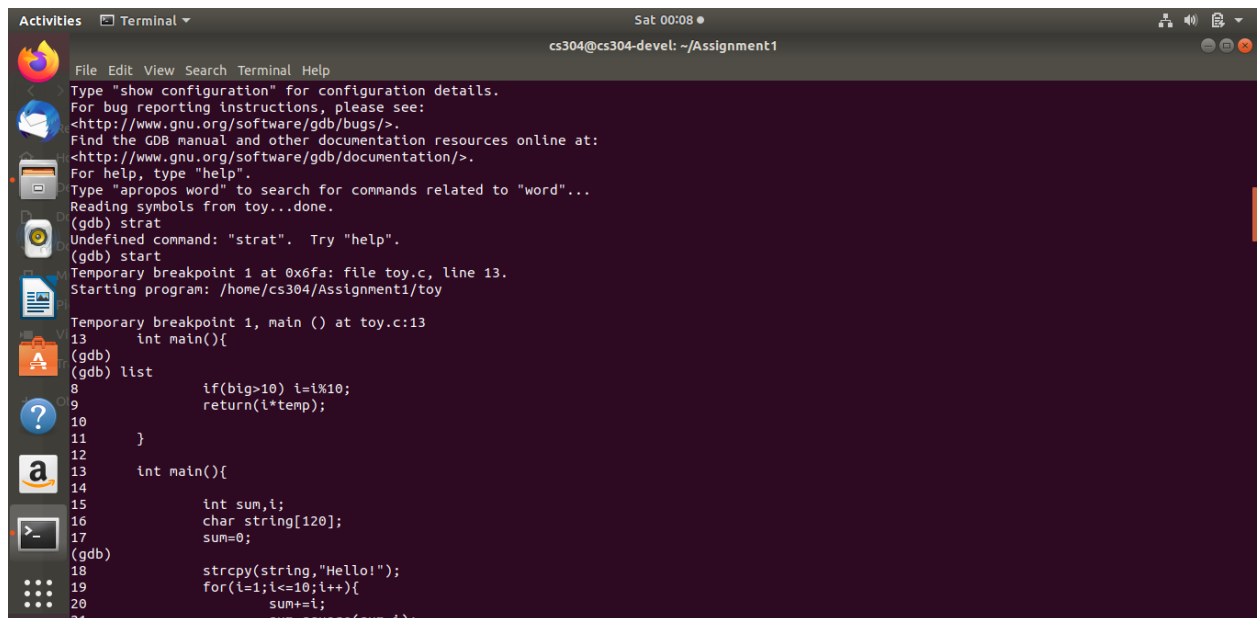
- (a) The last line of Makefile: `gcc -o myhello myhello.c` is responsible for creating the myhello executable.  
Changing the name is a function of **gcc**.

### 5. Fifth Question

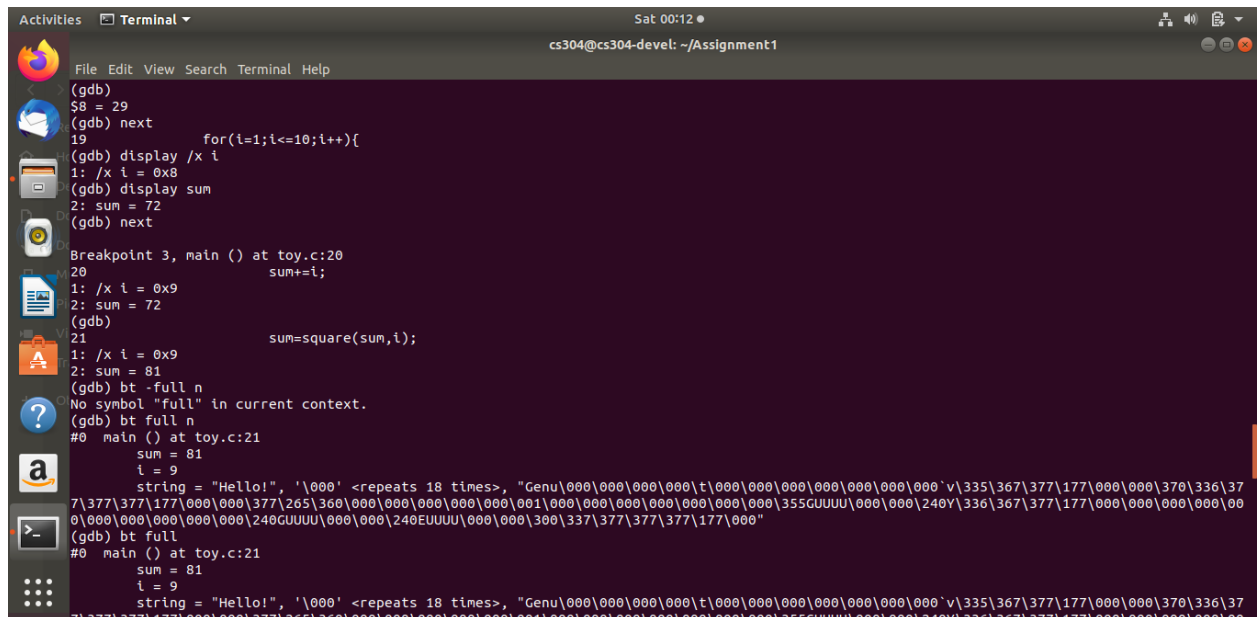
- (a) The strange looking rule is used to collectively compile all the files in the directory ending in **.c** using the gcc compiler.  
A change in either of the **.c** files doesn't force the re-compilation of another **.c** file, i.e. if `main.c` is changed, then only `main.c` and the `dumb.h` file is recompiled. Similar is the case when `dump.c` is changed; only the `dumb.c` file and `dumb.h` file is recompiled. But if **dumb.h** is changed, this forces all the **.c** files to be recompiled. Also, if no changes are made, the files are not recompiled and a message notifying that the output file is up to data is printed on the terminal.

### 6. Sixth Question

- (a) The `list` command is used to print a fixed number of lines of from the program. If a `linenumber` is provided with the `list` command, a fixed number of lines from around that line number is printed on the shell.



Display command allows us to display the values of certain variables, expression at break-points or at each step of the program.



Where or the backtrace command is useful in determinig where a particular part of code is called from. For a function, it will show where the function was called from up until the last end if the call chain.



## Breakpoints

```

Activities Terminal Sat 00:14
cs304@cs304-devel: ~/Assignment1

(gdb) start
Temporary breakpoint 1 at 0x60e: file main.c, line 6.
Starting program: /home/cs304/Assignment1/hello

Temporary breakpoint 1, main () at main.c:6
6 printf("%s","Hello World!\n");
(gdb) b 2
Breakpoint 2 at 0x5555555460e: file main.c, line 2.
(gdb) b 5
Note: breakpoint 2 also set at pc 0x5555555460e.
Breakpoint 3 at 0x5555555460e: file main.c, line 5.
(gdb) b 10
Breakpoint 4 at 0x555555546a9: file main.c, line 10.
(gdb) b 12
No line 12 in the current file.
Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) start
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Temporary breakpoint 5 at 0x5555555460e: file main.c, line 6.
Starting program: /home/cs304/Assignment1/hello

Breakpoint 2, main () at main.c:6
6 printf("%s","Hello World!\n");
(gdb)
(gdb)
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/cs304/Assignment1/hello

Breakpoint 2, main () at main.c:6
6 printf("%s","Hello World!\n");

```

```

Activities Terminal Sat 00:15
cs304@cs304-devel: ~/Assignment1

Starting program: /home/cs304/Assignment1/hello

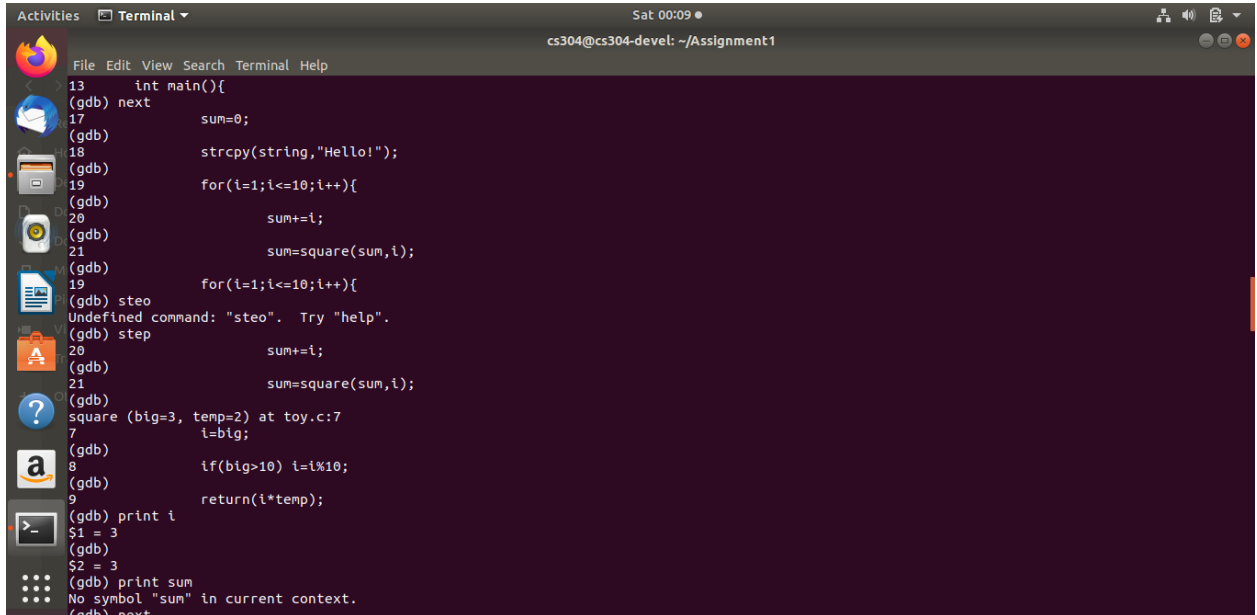
Breakpoint 2, main () at main.c:6
6 printf("%s","Hello World!\n");
(gdb)
(gdb)
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/cs304/Assignment1/hello

Breakpoint 2, main () at main.c:6
6 printf("%s","Hello World!\n");
(gdb) next
Hello World!
7 dumb(10);
(gdb) next
n = 10
9 return 0;
(gdb) next

Breakpoint 4, main () at main.c:11
11 }
(gdb) next
__libc_start_main (main=0x5555555460a <main>, argc=1, argv=0x7fffffffdfc8, init=<optimized out>, fini=<optimized out>,
rtld_fini=<optimized out>, stack_end=0x7fffffffdfb8) at ../csu/libc-start.c:344
344 ../csu/libc-start.c: No such file or directory.
(gdb) next
[Inferior 1 (process 15436) exited normally]
(gdb) next
The program is not being run.
(gdb)

```

## Stepping



```

Sat 00:09
cs304@cs304-devel: ~/Assignment1
File Edit View Search Terminal Help
13 int main(){
(gdb) next
17 sum=0;
(gdb)
18 strcpy(string,"Hello!");
(gdb)
19 for(i=1;i<=10;i++){
(gdb)
20 sum+=i;
(gdb)
21 sum=square(sum,i);
(gdb)
19 for(i=1;i<=10;i++){
(gdb) steo
Undefined command: "steo". Try "help".
(gdb) step
20 sum+=i;
(gdb)
21 sum=square(sum,i);
(gdb)
square (big=3, temp=2) at toy.c:7
7 i=big;
(gdb)
8 if(big>10) i=i%10;
(gdb)
9 return(i*temp);
(gdb) print i
$1 = 3
(gdb)
(gdb) print sum
$2 = 3
(gdb) print sum
No symbol "sum" in current context.
(gdb) next

```

## 7. Seventh Question

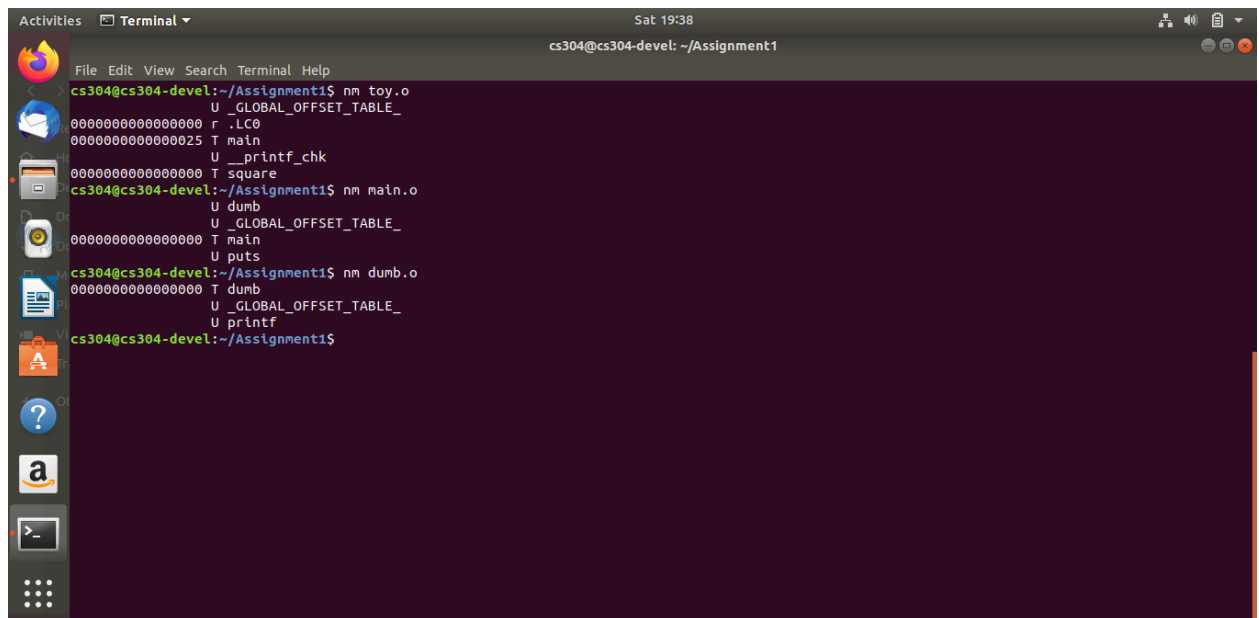
- (a) **nm** is a utility provided in UNIX based systems to examine binary files given out by programs, executable files(.exe) that are in windows, library files that generally end in .lib and other such files. The command displays the symbol table associated with these files. The output is generally in 3 columns by default:

First: the virtual address by default

Second: the symbol type

Third: the symbol name

The output is a mix of lowercase and uppercase symbols types where uppercase means "External symbols" and lowercase means "Local symbols". It also tells us about different functions that are imported from external libraries. Below is a screenshot showing a use nm:



```

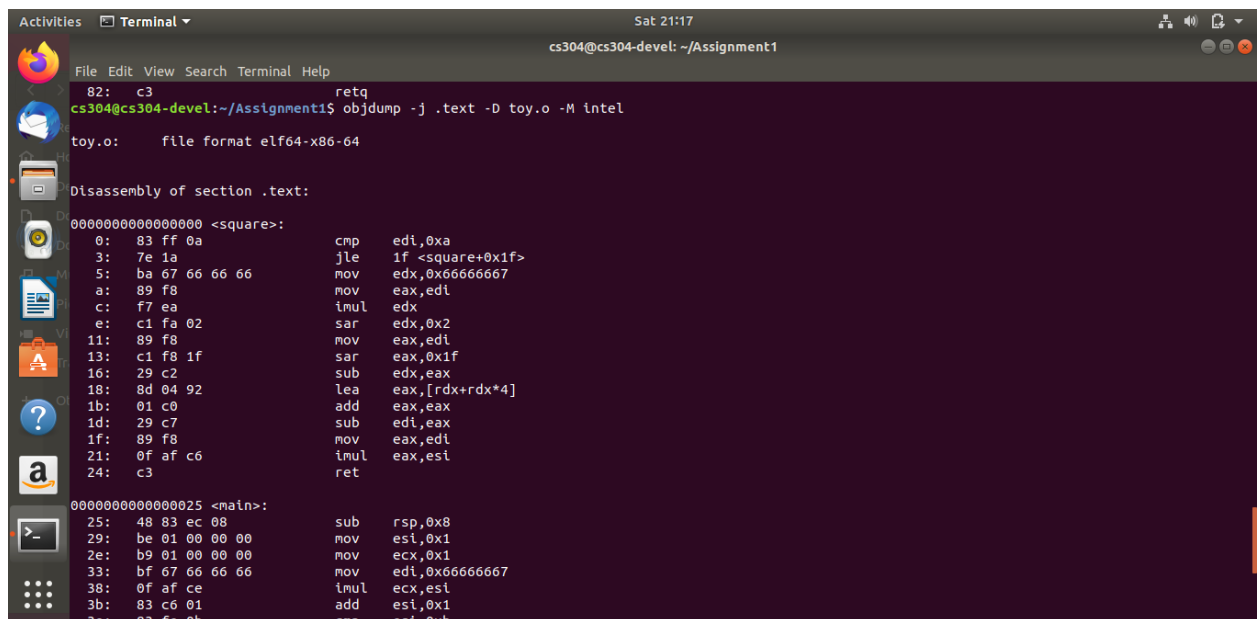
cs304@cs304-devel:~/Assignment1$ nm toy.o
0000000000000000 U _GLOBAL_OFFSET_TABLE_
0000000000000000 r .LC0
0000000000000025 T main
0000000000000000 U __printf_chk
0000000000000000 T square
cs304@cs304-devel:~/Assignment1$ nm main.o
0000000000000000 U dumb
0000000000000000 U _GLOBAL_OFFSET_TABLE_
0000000000000000 T main
0000000000000000 U puts
cs304@cs304-devel:~/Assignment1$ nm dumb.o
0000000000000000 T dumb
0000000000000000 U _GLOBAL_OFFSET_TABLE_
0000000000000000 U printf
cs304@cs304-devel:~/Assignment1$

```

**objdump** is another command-line program to display various info about object files on UNIX-like OS systems. This disassembles the program and allows us to look at the assembly code of the file. The information can be used to get an overview of the control flow of the program via a series of assembly commands. The output generally has four columns:

- . The virtual address of the instruction
- . The actual machine code
- . Last 2 columns: The actual assembly code that matches the machine code

Below is a use case of the objdump command:



```

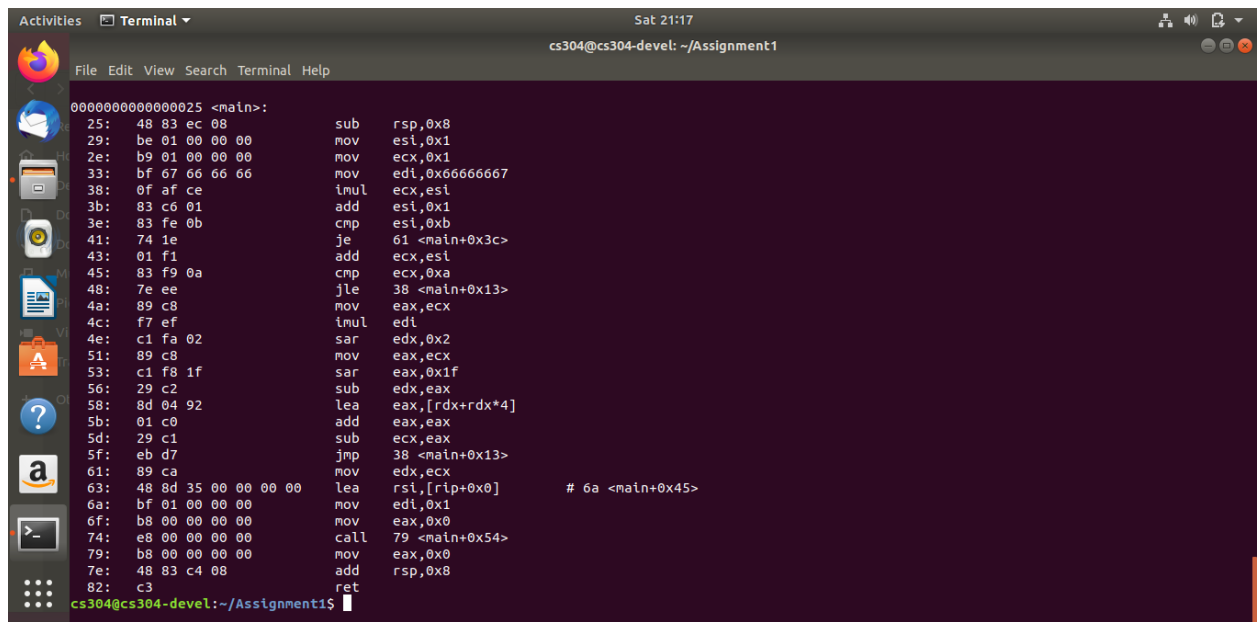
cs304@cs304-devel:~/Assignment1$ objdump -j .text -D toy.o -M intel
toy.o: file format elf64-x86-64

Disassembly of section .text:

0000000000000000 <square>:
0: 83 ff 0a cmp edi,0xa
3: 7e 1a jle 1f <square+0x1f>
5: ba 67 66 66 66 mov edx,0x66666667
a: 89 f8 mov eax,edi
c: f7 ea imul edx
e: c1 fa 02 sar edx,0x2
11: 89 f8 mov eax,edi
13: c1 f8 1f sar eax,0x1f
16: 29 c2 sub edx,eax
18: 8d 04 92 lea eax,[rdx+rdx*4]
1b: 01 c0 add eax,eax
1d: 29 c7 sub edi,eax
1f: 89 f8 mov eax,edi
21: 0f af c6 imul eax,esi
24: c3 ret

0000000000000025 <main>:
25: 48 83 ec 08 sub rsp,0x8
29: be 01 00 00 00 mov esi,0x1
2e: b9 01 00 00 00 mov ecx,0x1
33: bf 67 66 66 66 mov edi,0x66666667
38: 0f af ce imul ecx,esi
3b: 83 c6 01 add esi,0x1
3e: 83 fe 0b cmp esi,0xb

```

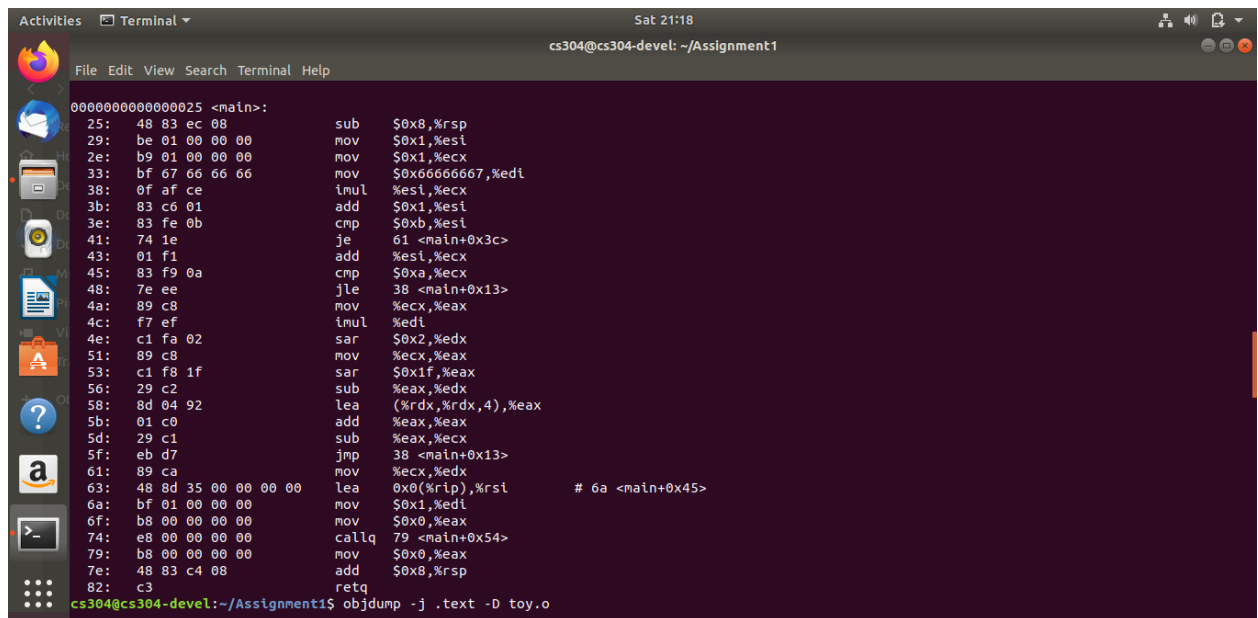


```

Sat 21:17
cs304@cs304-devel: ~/Assignment1
File Edit View Search Terminal Help
0000000000000025 <main>:
25: 48 83 ec 08 sub rsp,0x8
29: be 01 00 00 00 mov esi,0x1
2e: b9 01 00 00 00 mov ecx,0x1
33: bf 67 66 66 66 mov edi,0x66666667
38: 0f af ce imul ecx,esi
3b: 83 c6 01 add esi,0x1
3e: 83 fe 0b cmp esi,0xb
41: 74 1e je 61 <main+0x3c>
43: 01 f1 add ecx,esi
45: 83 f9 0a cmp ecx,0xa
48: 7e ee jle 38 <main+0x13>
4a: 89 c8 mov eax,ecx
4c: f7 ef imul edi
4e: c1 fa 02 sar edx,0x2
51: 89 c8 mov eax,ecx
53: c1 f8 1f sar eax,0x1f
56: 29 c2 sub edx,eax
58: 8d 04 92 lea eax,[rdx+rdx*4]
5b: 01 c0 add eax,eax
5d: 29 c1 sub ecx,eax
5f: eb d7 jmp 38 <main+0x13>
61: 89 ca mov edx,ecx
63: 48 8d 35 00 00 00 00 lea rsi,[rip+0x0] # 6a <main+0x45>
6a: bf 01 00 00 00 mov edi,0x1
6f: b8 00 00 00 00 mov eax,0x0
74: e8 00 00 00 00 call 79 <main+0x54>
79: b8 00 00 00 00 mov eax,0x0
7e: 48 83 c4 08 add rsp,0x8
82: c3 ret
cs304@cs304-devel:~/Assignment1$

```

Above two are in the Intel format which is easier to read. The default is the AT&T format below:



```

Sat 21:18
cs304@cs304-devel: ~/Assignment1
File Edit View Search Terminal Help
0000000000000025 <main>:
25: 48 83 ec 08 sub $0x8,%rsp
29: be 01 00 00 00 mov $0x1,%esi
2e: b9 01 00 00 00 mov $0x1,%ecx
33: bf 67 66 66 66 mov $0x66666667,%edi
38: 0f af ce imul %esi,%ecx
3b: 83 c6 01 add $0x1,%esi
3e: 83 fe 0b cmp $0xb,%esi
41: 74 1e je 61 <main+0x3c>
43: 01 f1 add %esi,%ecx
45: 83 f9 0a cmp $0xa,%ecx
48: 7e ee jle 38 <main+0x13>
4a: 89 c8 mov %ecx,%eax
4c: f7 ef imul %edi
4e: c1 fa 02 sar $0x2,%edx
51: 89 c8 mov %ecx,%eax
53: c1 f8 1f sar $0x1f,%eax
56: 29 c2 sub %eax,%edx
58: 8d 04 92 lea (%rdx,%rdx,4),%eax
5b: 01 c0 add %eax,%eax
5d: 29 c1 sub %eax,%ecx
5f: eb d7 jmp 38 <main+0x13>
61: 89 ca mov %ecx,%edx
63: 48 8d 35 00 00 00 00 lea 0x0(,%rip),%rsi # 6a <main+0x45>
6a: bf 01 00 00 00 mov $0x1,%edi
6f: b8 00 00 00 00 mov $0x0,%eax
74: e8 00 00 00 00 callq 79 <main+0x54>
79: b8 00 00 00 00 mov $0x0,%eax
7e: 48 83 c4 08 add $0x8,%rsp
82: c3 retq
cs304@cs304-devel:~/Assignment1$ objdump -j .text -D toy.o

```