NAME – KAMAL CHOUDHURY
ROLL NO. – 1801CS27

PROJECT CS321

TWO-PASS ASSEMBLER FOR SIMPLE ASSEMBLY LANGUAGE IN C

COMMAND LINE ARGUMENTS USED: →

1) First give the below command in linux terminal:
   gcc asm.c -o asm
2) Then asm executable file will be created which will be processed on asm files.
3) Suppose the test filename is test1.asm. Now to assemble it give the following command.
   ./asm test1.asm
4) Then 3 files will be created whose details are mentioned below.

FILE DETAILS: →

I have made a 2-pass assembler which takes file with asm extension & creates output 3 files.

If the name of the asm file is filename.asm, then these 3 files will be created →

i) **filename.log** file --> It tells all the errors in the file & it is a text file.

ii**) filename.lst** file --> It has the program listing of the file & it is a text file.

iii**) filename.o** file --> It has the machine code of the asm file & is a binary file.

iv) Now we can access each of them using the below 3 commands in command-line interface -->

more filename.log

more filename.lst

hexdump filename.o

CLAIMS FILE: →

I have submitted a text file named Claims_File.txt which lists all my claims & assumptions I have made during the making of the code.

TEST FILES: →

I have tested my code on the following asm files & for each file there are 3 more files created (log file, lst file, machine code file) which I have submitted in the folder. Out of these test1.asm, test2.asm, test3.asm & test4.asm were already provided & I myself created additional 4 more test files namely: test5.asm, test6.asm, test7.asm where test5.asm & test6.asm are asm files having errors while test7.asm are assembled successfully. Also, test7.asm is an asm program to subtract 2 numbers. Also, I created bubble.asm as it was required in the document.

LIST OF ASM FILES: ->

1. test1.asm
2. test2.asm
3. test3.asm
4. test4.asm
5. test5.asm
6. test6.asm
7. test7.asm
8. bubble.asm

For each of the above files (.lst), (.log) & (.o) files will be created.

Example: - for test1.asm the following files will be created →

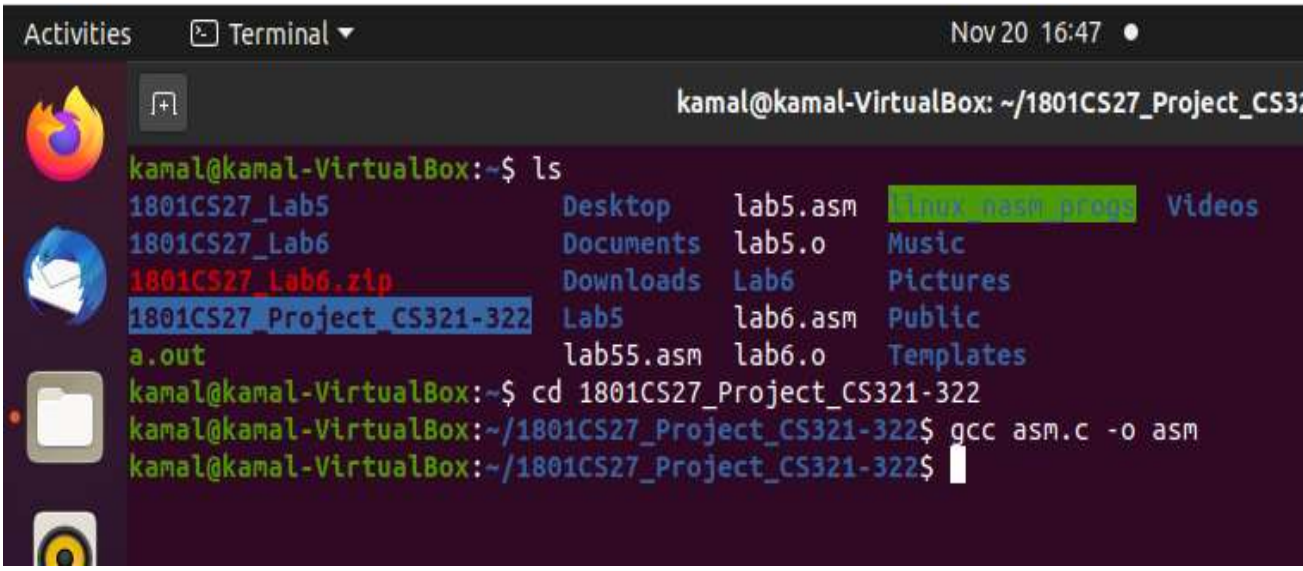
1. test1.log (Containing errors or warnings in test1.asm, if there)
2. test1.lst (Listing file)
3. test1.o (Machine code)

SCREENSHOTS OF ASSEMBLER AS A WORKING MODEL: →

I will show that my assembler is working via screenshots by taking 2 test files (test2.asm & test4.asm)

Compile asm.c & make executable file named asm

1) First compile using **gcc asm.c -o asm.** It compiles without any warnings or errors.



Then I will test my asm executable on two test files. (test2.asm & test4.asm). test2.asm is an asm file containing errors while test4.asm assembles successfully & thus assembler works in both cases.

Test2.asm

1) Then use the command → **./asm test2.asm**
2) Now number of errors & warnings will be shown as shown below.

```
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$ gcc asm.c -o asm
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$ ./asm test2.asm
There are 2 warnings and there are 12 errors (mentioned in the log file).
Since there are errors in the asm file the listing & object files are made only till the first
Now 3 files have been created ::
1) test2.log :: (Telling all the errors)
2) test2.lst :: (Listing file)
3) test2.o :: (Object file)
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$
```

3) Now as we can see above there are 2 warnings & 12 errors in the file which has been reported by the assembler.
4) 3 files have been created & errors are there in the **test2.log** file.
5) To check that let's use command → **more test2.log**

```
3) test2.o :: (Object file)
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$ more test2.log
WARNING on Line No. 3 :: Label is not used
ERROR on Line No. 4 :: There is a label named label which is a Duplicate Label
ERROR on Line No. 5 :: The label used as the operand does not exist
ERROR on Line No. 6 :: The operand is non-numerical
ERROR on Line No. 7 :: A Numerical Value or a label was expected
ERROR on Line No. 8 :: No numerical Value or label was expected
ERROR on Line No. 9 :: Only one numerical value or label was expected
ERROR on Line No. 10 :: Label must be starting with an alphabet
ERROR on Line No. 11 :: fibble is not a valid mnemonic
ERROR on Line No. 12 :: 0def is not a valid mnemonic
```

6) The listing file is there in the **test2.lst** file.
7) Since there are errors in this file listing file cannot be created properly, hence I have made the listing file until the first error is encountered.
8) To open the listing file we can use command → **more test2.lst** which can be seen below in the screenshot.

```
ERROR on Line No. 10 :: Label must be starting with an alphabet
ERROR on Line No. 11 :: fibble is not a valid mnemonic
ERROR on Line No. 12 :: 0def is not a valid mnemonic
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$ more test2.lst
; test2.asm
; Test error handling
00000000 00000000 label:
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$
```

9) Also the machine code is created which is the file **test2.o** which is a binary file & could be opened using hexdump command which reads

the binary file & gives hexadecimal values for each byte. Since there are errors in this file object file cannot be created properly, hence I have made the object file until the first error is encountered. Here Nothing is being created.

10) Command used will be → **hexdump test2.o**

```
; Test error handling
00000000 00000000 label:
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$ hexdump test2.o
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$
```

Test4.asm

1) Use the command → **./asm test4.asm**
2) Now number of errors & warnings will be shown as shown below.

```
0000004
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$ ./asm test4.asm
There is no warnings or errors.
Now 3 files have been created ::
1) test4.log :: (Telling all the errors)
2) test4.lst :: (Listing file)
3) test4.o :: (Object file)
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$
```

3) Now as we can see above there are no warnings & no errors in the file it has been reported by the assembler.
4) 3 files have been created & errors are there in the **test4.log** file. (test4.log file is empty since there are no errors)
5) To check that let's use command → **more test4.log**

```
ERROR on Line No. 15 :: cada is not a valid mnemonic
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$ more test4.log
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$
```

6) The listing file is there in the **test4.lst** file.
7) Since there are no errors in this file listing file is created perfectly. I have made an advanced listing file with program counter, machine code & instructions as shown below.
8) To open the listing file we can use command → **more test4.lst** which can be seen below in the screenshot.

```
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$ more test4.log
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$ more test4.lst
00000000 00100000          ldc 0x1000
00000001 0000000B          a2sp
00000002 FFFFFF0A          adj -1
00000003 00004B00          ldc result
00000004 00000003          stl 0
00000005 00004A00          ldc count
00000006 00000004          ldnl 0
00000007 0000020D          call main
00000008 0000010A          adj 1
00000009 00000012          HALT
;
0000000A FFFFFD0A main:    adj -3
0000000B 00000103          stl 1
0000000C 00000203          stl 2
0000000D 00000000          ldc 0            ; zero accumulator
0000000E 00000003          stl 0
0000000F FFFFFF0A loop:    adj -1
00000010 00000302          ldl 3
00000011 00000003          stl 0
00000012 00000102          ldl 1
00000013 0000100D          call triangle
00000014 0000010A          adj 1
00000015 00000302          ldl 3
00000016 00000005          stnl 0
00000017 00000302          ldl 3
00000018 00000101          adc 1
00000019 00000303          stl 3
0000001A 00000002          ldl 0
0000001B 00000101          adc 1
0000001C 00000003          stl 0
0000001D 00000002          ldl 0            ; reload it
0000001E 00000202          ldl 2
0000001F 00000007          sub
00000020 FFFFEE10          brlz loop
00000021 00000102          ldl 1            ; get return address
00000022 0000030A          adj 3
00000023 0000000E          return
;
00000024 FFFFFD0A triangle:adj -3
00000025 00000103          stl 1
00000026 00000203          stl 2
00000027 00000100          ldc 1
00000028 00000008          shl
00000029 00000302          ldl 3
0000002A 00000007          sub
0000002B 00000410          brlz skip
0000002C 00000302          ldl 3
```

```
00000021 00000102          ldl 1            ; get retu
00000022 0000030A          adj 3
00000023 0000000E          return
;
00000024 FFFFFD0A triangle:adj -3
00000025 00000103          stl 1
00000026 00000203          stl 2
00000027 00000100          ldc 1
00000028 00000008          shl
00000029 00000302          ldl 3
0000002A 00000007          sub
0000002B 00000410          brlz skip
0000002C 00000302          ldl 3
0000002D 00000202          ldl 2
0000002E 00000007          sub
0000002F 00000203          stl 2
00000030 00000202 skip:    ldl 2
00000031 0000140F          brz one
00000032 00000302          ldl 3
00000033 FFFFFF01          adc -1
00000034 00000003          stl 0
00000035 FFFFFF0A          adj -1
00000036 00000102          ldl 1
00000037 00000003          stl 0
00000038 00000302          ldl 3
00000039 FFFFFF01          adc -1
0000003A FFFFE90D          call triangle
0000003B 00000102          ldl 1
0000003C 00000003          stl 0
0000003D 00000103          stl 1
0000003E 00000302          ldl 3
0000003F FFFFE40D          call triangle
00000040 0000010A          adj 1
00000041 00000002          ldl 0
00000042 00000006          add
00000043 00000102          ldl 1
00000044 0000030A          adj 3
00000045 0000000E          return
00000046 00000100 one:     ldc 1
00000047 00000102          ldl 1
00000048 0000030A          adj 3
00000049 0000000E          return
;
0000004A 0000000A count:   data 10
0000004B 00000000 result:  data 0
```

9) Also the machine code is created which is the file **test4.o** which is a binary file & could be opened using hexdump command which reads

the binary file & gives hexadecimal values for each byte. Since there are no errors in this file object file is created successfully.

10) Command used will be → **hexdump test4.o**

```
kamal@kamal-VirtualBox:~/1801CS27_Project_CS321-322$ hexdump test4.o
0000000 0000 0010 000b 0000 ff0a ffff 4b00 0000
0000010 0003 0000 4a00 0000 0004 0000 020d 0000
0000020 010a 0000 0012 0000 fd0a ffff 0103 0000
0000030 0203 0000 0000 0000 0003 0000 ff0a ffff
0000040 0302 0000 0003 0000 0102 0000 100d 0000
0000050 010a 0000 0302 0000 0005 0000 0302 0000
0000060 0101 0000 0303 0000 0002 0000 0101 0000
0000070 0003 0000 0002 0000 0202 0000 0007 0000
0000080 ee10 ffff 0102 0000 030a 0000 000e 0000
0000090 fd0a ffff 0103 0000 0203 0000 0100 0000
00000a0 0008 0000 0302 0000 0007 0000 0410 0000
00000b0 0302 0000 0202 0000 0007 0000 0203 0000
00000c0 0202 0000 140f 0000 0302 0000 ff01 ffff
00000d0 0003 0000 ff0a ffff 0102 0000 0003 0000
00000e0 0302 0000 ff01 ffff e90d ffff 0102 0000
00000f0 0003 0000 0103 0000 0302 0000 e40d ffff
0000100 010a 0000 0002 0000 0006 0000 0102 0000
0000110 030a 0000 000e 0000 0100 0000 0102 0000
0000120 030a 0000 000e 0000 000a 0000 0000 0000
0000130
```