# Government Engineering College, Thrissur

CS331 – System Software Lab

Documentation -

# Exp 9 – Pass 2 of Two Pass Assembler

Date of Submission
28 October 2020

Submitted By
**Kowsik Nandagopan D**
Roll No 31
TCR18CS031
GECT CSE S5

# Experiment 9

## AIM

Implement pass two of a two pass assembler

## Compiling of Code

### Prerequisite

- The code is provided in the **pass1.c** & **pass2.c** along with this documentation. You can open the terminal in Linux (Ubuntu 18.04 tested). Then run the command

  *gcc program.c*

  *./a.out*

  <u>**PASS 1**</u>

- Compile and run **pass1.c** using the above code.
- We have **two** input files. One is *optab.txt* which denote the OPTAB of the assemblers. Also most importantly the source code is stored in the *input,txt.* Note in reality this file will be of *.asm extension. For the simplicity here we use *.txt extension
  1. *optab.txt:* It stores the operation codes allowed in the assembly language. Format for the input is

     *<Opcode (String)> <Tab> <Hex Code corresponding to opcode (String)>*

  2. *input.txt:* We store the assembly language in this file. The assembly language used is SIC (Simple Instruction Computer)
- Output of the code will be printed on the **console** and the SYMTAB will be stored to ***symtab.txt*** and ***inter.txt***
- **Length of the program is stored to the flen.txt**
- **Note: Please see the inter.txt and symtab.txt will be the input for the next step.**

  <u>**PASS 2**</u>

- Compile and run **pass2.c** as usually we do with any normal c file.
- **Input for this program will be the out put of the pass 1. So no need of special input and renaming of auto generated files.**
- This program uses the intermediate file of the source code, SYMTAB, OPTAB and flen file (File containing the length of program) to generate the output which is the object code. The object code is out put as **output.txt**

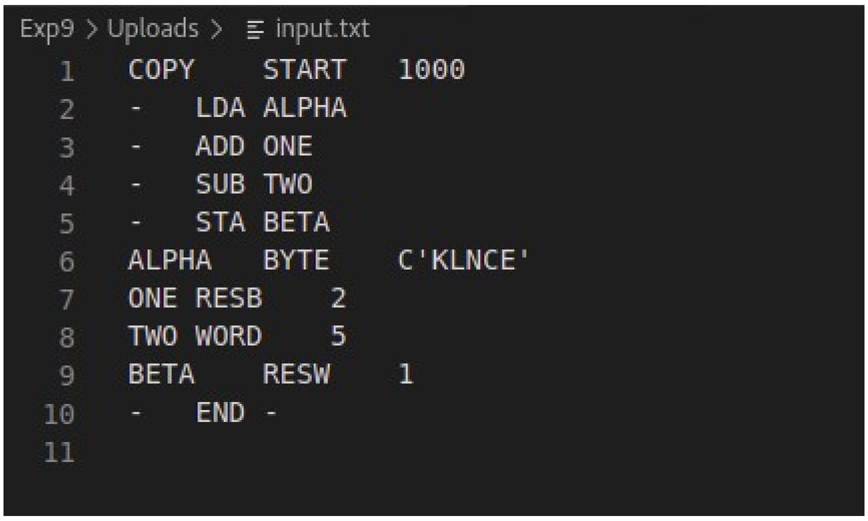
- The output as 3 record sections. First line is the Header Record starting with H. This record contains the name of the program, starting address and the length of the entire program. **Note in the actual object code ^ will not be used. But here for the simplicity of reading the instructions are separated by ^ symbol.**
- Second part is the Text record which contains the program itself in the object code format. This is generated using the SYMTAB and OPTAB
- Last part or the last line is the End record which denotes the end of the program and provides the address of the first executable instruction. Any instruction written after the End record will not be executed.
- **Output of the program are printed into a file named *output.txt***
- To see the output in console use the code *cat output.txt*

## Output / Screenshots

## PASS 1

### Input

Source code – input.txt

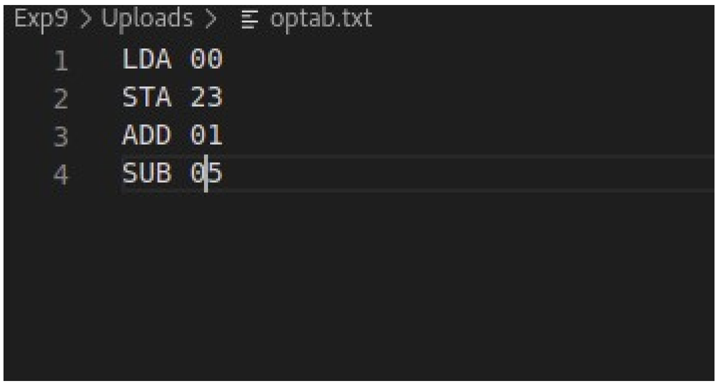


Exp9 > Uploads > input.txt

```
COPY    START   1000
-   LDA ALPHA
-   ADD ONE
-   SUB TWO
-   STA BETA
ALPHA   BYTE    C'KLNCE'
ONE RESB    2
TWO WORD    5
BETA    RESW    1
-   END -

```

OPTAB – optab.txt

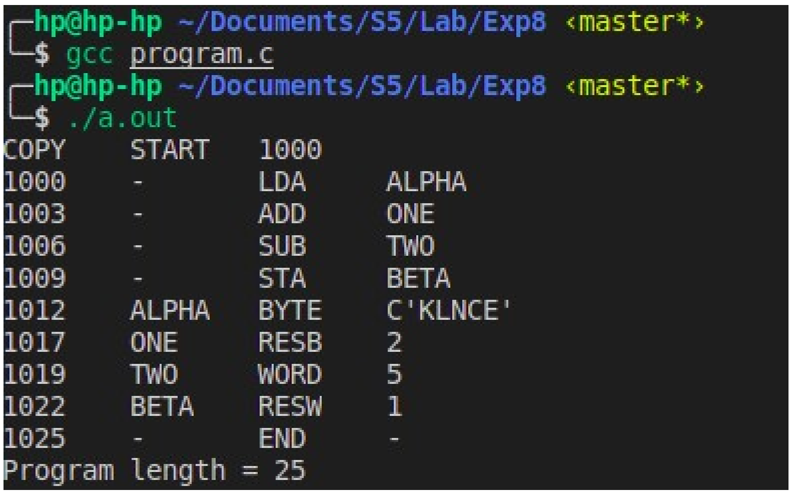


Exp9 > Uploads > optab.txt

```
LDA 00
STA 23
ADD 01
SUB 05
```

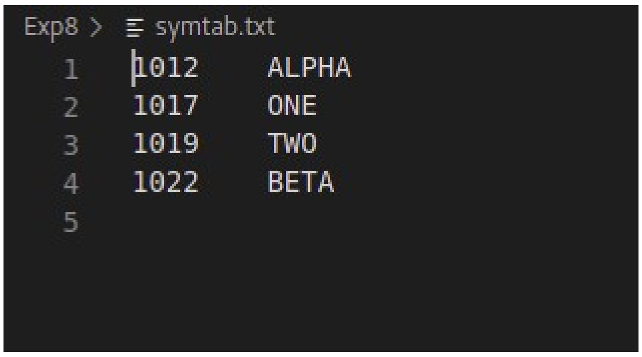
**Output**

Output to console



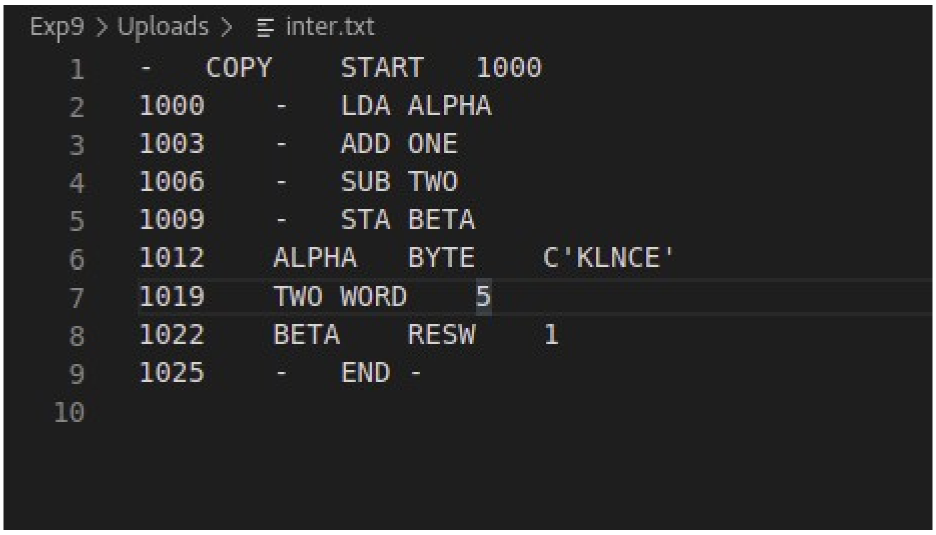
```
┌─hp@hp-hp ~/Documents/S5/Lab/Exp8 ‹master*›
└─$ gcc program.c
┌─hp@hp-hp ~/Documents/S5/Lab/Exp8 ‹master*›
└─$ ./a.out
COPY    START   1000
1000    -       LDA     ALPHA
1003    -       ADD     ONE
1006    -       SUB     TWO
1009    -       STA     BETA
1012    ALPHA   BYTE    C'KLNCE'
1017    ONE     RESB    2
1019    TWO     WORD    5
1022    BETA    RESW    1
1025    -       END     -
Program length = 25
```

SYMTAB – symtab.txt



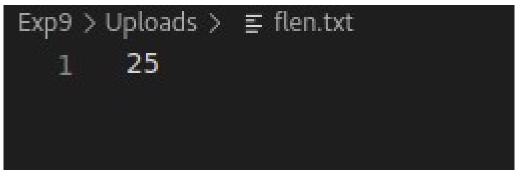
```
1012    ALPHA
1017    ONE
1019    TWO
1022    BETA
```

Intermediate file – inter.txt



```
-   COPY    START   1000
1000    -   LDA ALPHA
1003    -   ADD ONE
1006    -   SUB TWO
1009    -   STA BETA
1012    ALPHA   BYTE    C'KLNCE'
1019    TWO WORD    5
1022    BETA    RESW    1
1025    -   END -
```

Program Length – flen.txt



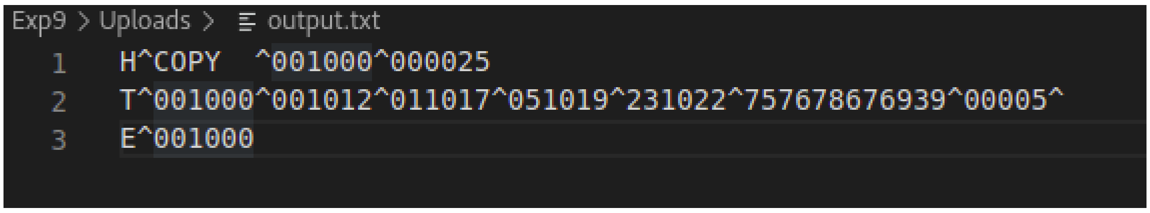
Exp9 > Uploads > flen.txt

```
25
```

## PASS 2

**Inputs are inter.txt, symtab.txt, optab.txt and flen.txt** (Screenshots are provided above)

**Output**
Object code – output.txt



Exp9 > Uploads > output.txt

```
H^COPY  ^001000^000025
T^001000^001012^011017^051019^231022^757678676939^00005^
E^001000
```