ELP780

Software Lab

Aghil Sabu

2018EET2865

A report presented for the assignment on
Assignment 8 - Python and Github

Electrical Engineering
IIT Delhi
India
September 18, 2019

Contents

1 Problem Satement 1

1.1 Problem Statement

Parity Check

The simplest way of error detection is to append a single bit, called a parity check, to a string of data bits. This parity check bit has the value 1 if the number of 1's in the bit string is even and has the value 0 otherwise, i.e., Odd Parity Check.

Bit-Oriented Framing

Data Link Layer needs to pack bits into frames so that each frame is distinguishable from another. Frames can be fixed or variable size. In variable size framing, we define the end of the frame using a bit-oriented approach. It uses a special string of bits, called a flag for both idle fills and to indicate the beginning and the ending of frames.
The bit stuffing rule is to insert a 0 after each appearance of 010 in the original data.
The string 0101 is used as the bit string or flag to indicate the end of the frame.

1.2 Input Format

Enter binary bit data that has to be transmitted.

1.3 Output Format

Print binary bit data with parity bit.
Print the modified string that is to be transmitted

1.4 Sample Input

010101110100101

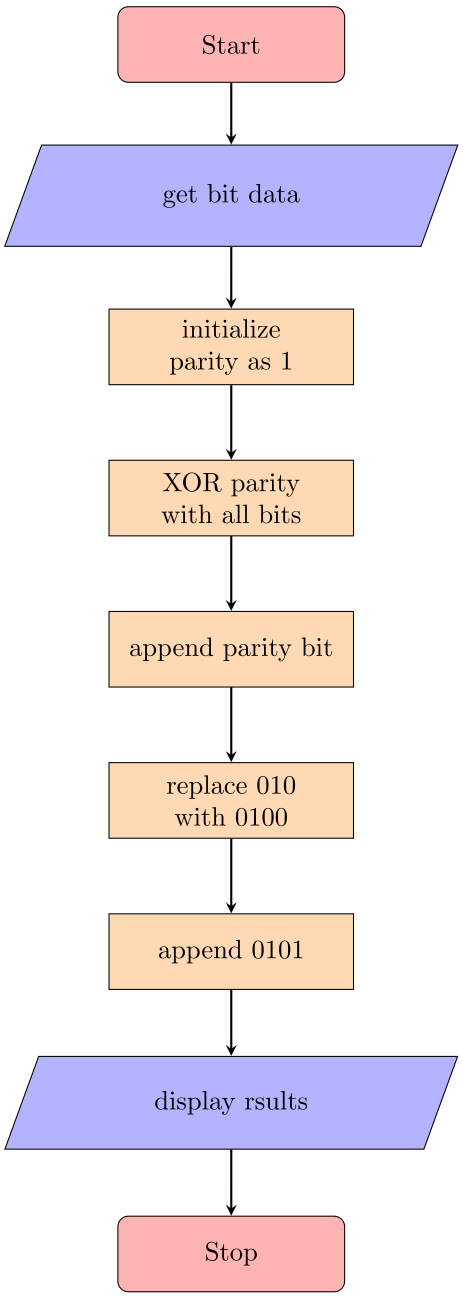
1.5 Sample Output

Parity bit data : 0101011101001011
Transmitting data: 01001011101000100110101

1.6 PS1 algorithm

- Read bit data from user
- Initialize parity as 1
- XOR parity with all bits to get parity value
- append parity bit
- replace pattern 010 with 0100
- append 0101 at the end
- display rsults in the required format

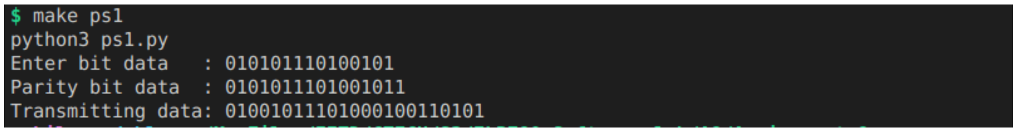
1.7 PS1 Flow Chart



1.8 PS1 Solution-Code

```
# reading bit data from user
data=input("Enter bit data   : ")
# initialising parity to 1 as initial parity value
parity=1
# finding parity of data stream
for i in data:
    parity=parity^int(i)
# adding parity bit to data
data+=str(parity)
print("Parity bit data  :",data)
# replacing 010 in data with 0100 to get data to be transmitted
trans=data.replace("010","0100")
# appending 0101 to the data to be transmitted
trans+="0101"
print("Transmitting data:",trans)
```

1.9 PS1 Output Screenshots



```
$ make ps1
python3 ps1.py
Enter bit data   : 010101110100101
Parity bit data  : 0101011101001011
Transmitting data: 01001011101000100110101
```

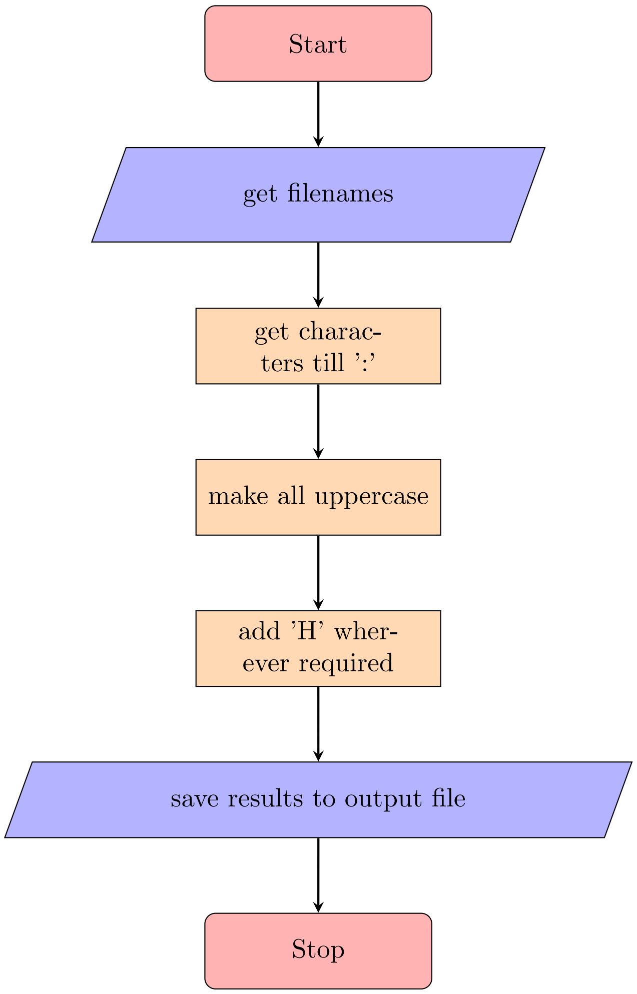
2 Problem Satement 2

A 8085 assembler is designed in a way that it takes input assembly source code instructions in all capital letters and numbers in hexadecimal format suffixed by H. You have to create preprocessor for assembler that will translate any 8085 assembly code into full capital letters and all numerals into hexadecimal format suffixed by H. Use lex(flex) and Yacc(bison).

2.1 PS2 algorithm

- Get filenames from user
- for each line get all characters before colon ':'
- make all uppercase
- add hexadecimal 'H' wherever required
- save result to output file

2.2 PS2 Flow Chart



2.3 PS2 Solution-Code

2.4 PS2 Output

2.4.1 input file

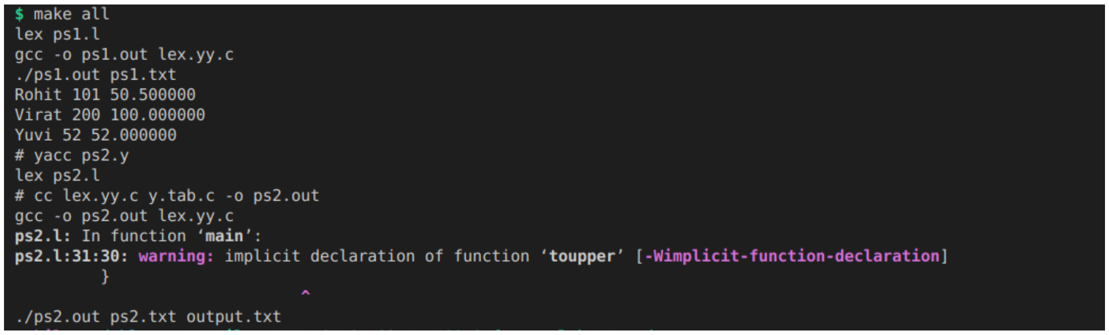
2.4.2 output file

3 makefile

3.1 makefile code

```
all: ps1 ps2
ps1:
python3 ps1.py
ps2:
python3 ps2.py

# clean:
#  rm *.out
```

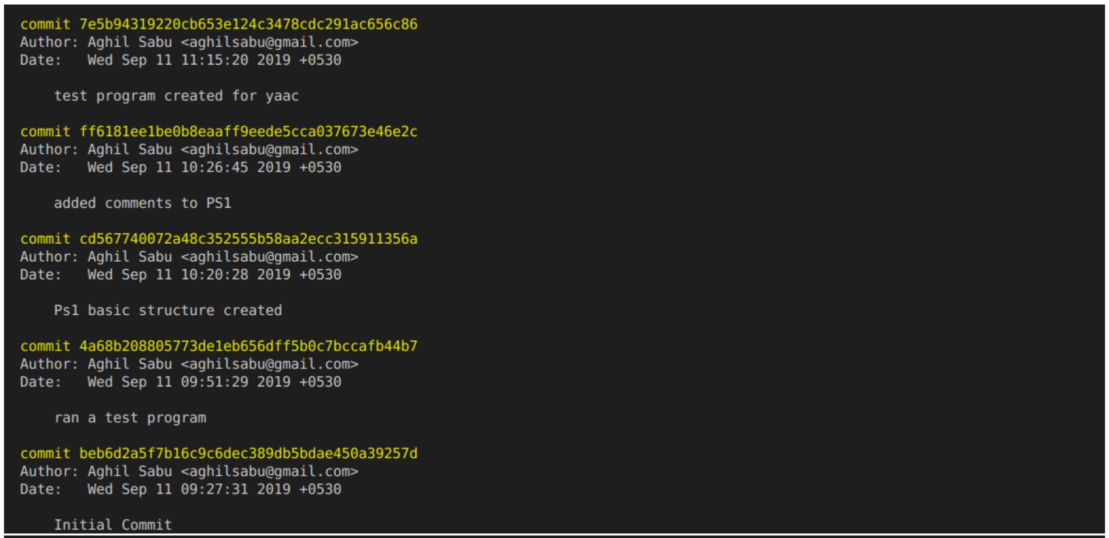
3.2 makefile output



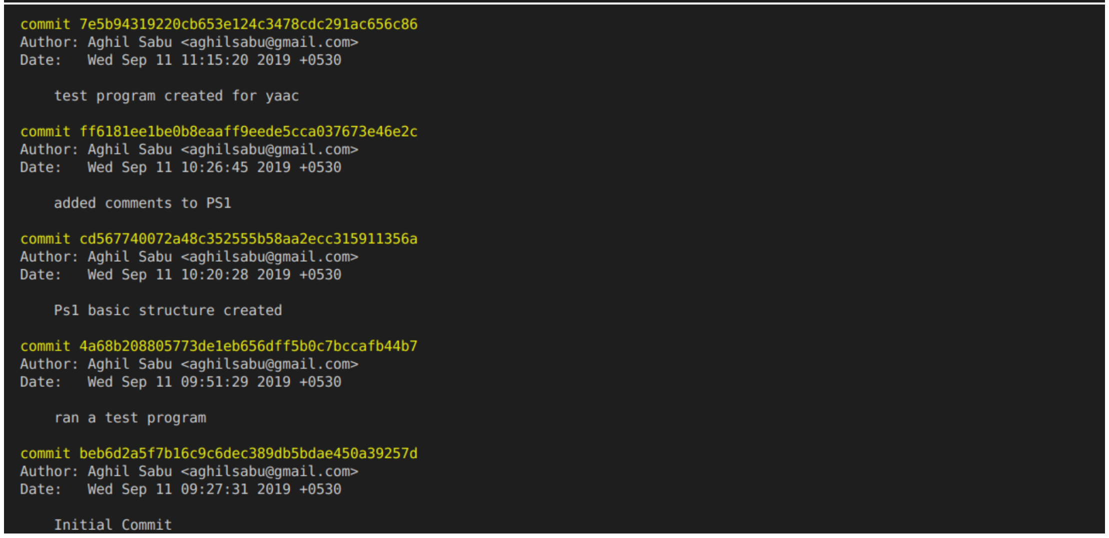
```
$ make all
lex ps1.l
gcc -o ps1.out lex.yy.c
./ps1.out ps1.txt
Rohit 101 50.500000
Virat 200 100.000000
Yuvi 52 52.000000
# yacc ps2.y
lex ps2.l
# cc lex.yy.c y.tab.c -o ps2.out
gcc -o ps2.out lex.yy.c
ps2.l: In function ‘main’:
ps2.l:31:30: warning: implicit declaration of function ‘toupper’ [-Wimplicit-function-declaration]
         }
                              ^

./ps2.out ps2.txt output.txt
```

4 GIT

4.1 GIT Commit Screenshots



```
commit 7e5b94319220cb653e124c3478cdc291ac656c86
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 11:15:20 2019 +0530

    test program created for yaac

commit ff6181ee1be0b8eaaff9eede5cca037673e46e2c
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 10:26:45 2019 +0530

    added comments to PS1

commit cd567740072a48c352555b58aa2ecc315911356a
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 10:20:28 2019 +0530

    Ps1 basic structure created

commit 4a68b208805773de1eb656dff5b0c7bccafb44b7
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 09:51:29 2019 +0530

    ran a test program

commit beb6d2a5f7b16c9c6dec389db5bdae450a39257d
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 09:27:31 2019 +0530

    Initial Commit
```



```
commit 7e5b94319220cb653e124c3478cdc291ac656c86
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 11:15:20 2019 +0530

    test program created for yaac

commit ff6181ee1be0b8eaaff9eede5cca037673e46e2c
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 10:26:45 2019 +0530

    added comments to PS1

commit cd567740072a48c352555b58aa2ecc315911356a
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 10:20:28 2019 +0530

    Ps1 basic structure created

commit 4a68b208805773de1eb656dff5b0c7bccafb44b7
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 09:51:29 2019 +0530

    ran a test program

commit beb6d2a5f7b16c9c6dec389db5bdae450a39257d
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 09:27:31 2019 +0530

    Initial Commit
```