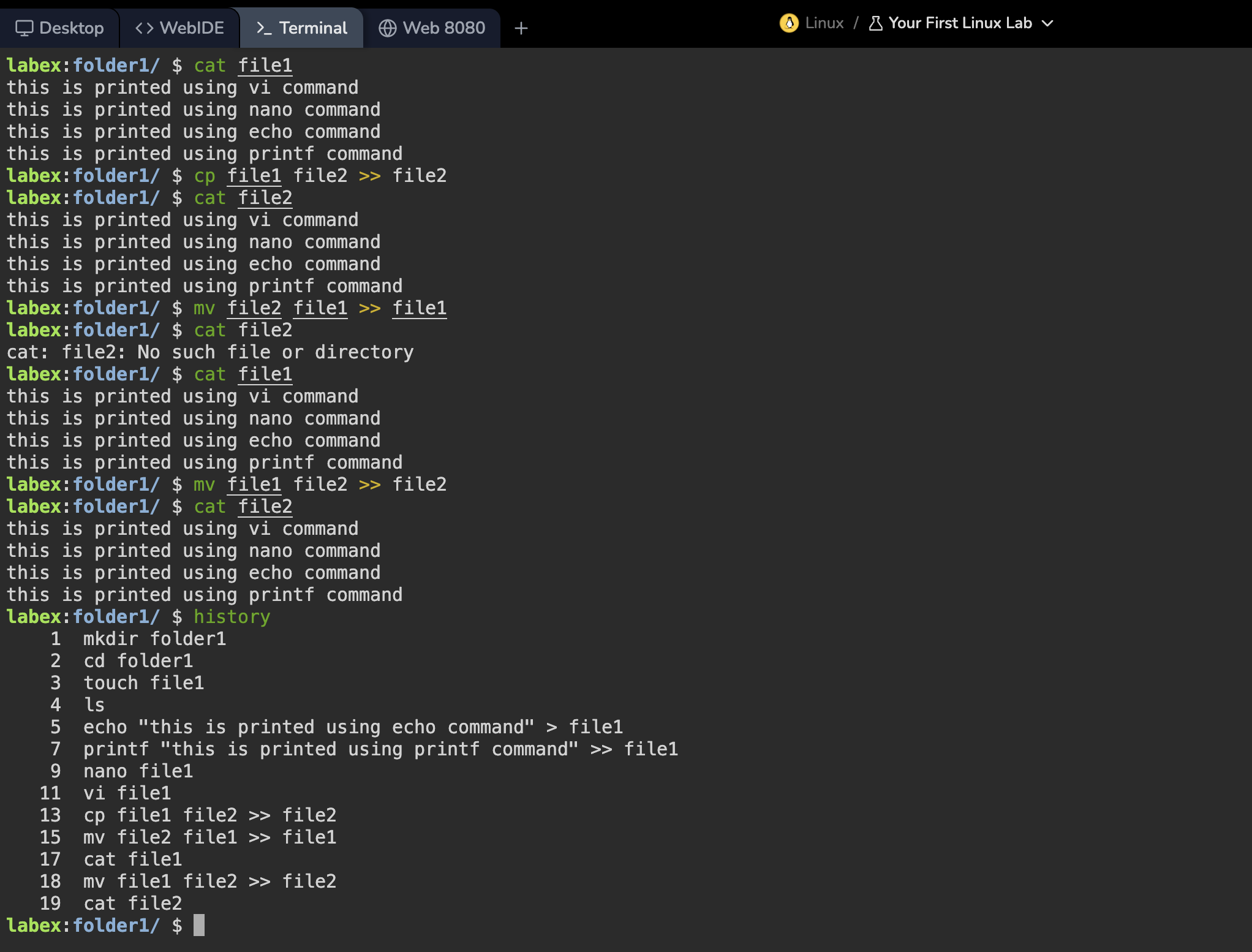
DAY 1

ls : list command

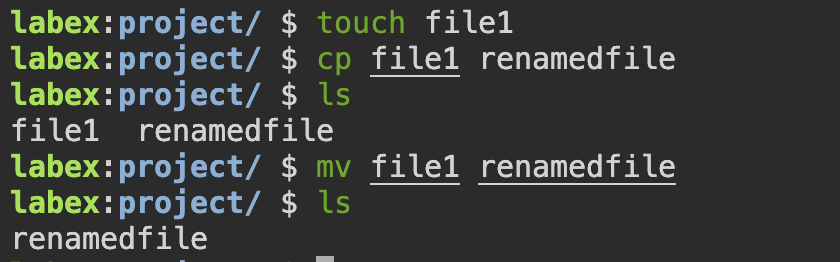
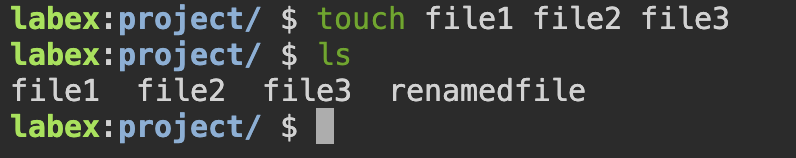
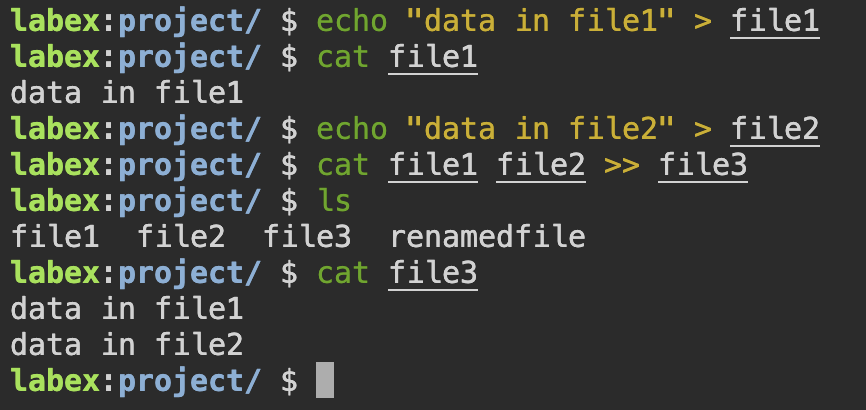
cd : Change directory command

mkdir : Create a Directory command  
rmdir : remove an empty directory  
rm -rf directory\_name : remove a directory with contents

touch : create a blank file   
echo : to add input to a file   
echo “file\_content” > file\_name : to create a file with content in it   
printf : to print the content into a file   
nano : to edit a file in nano editor  
vi : to edit a file in vi editor  
mv : to move the contents of a file or folder to another folder  
cp : to copy the contents of a file or folder to another file or folder  
cp -r : to copy the contents of a folder to another folder  
cat : to print the contents of a file or folder   
history : to print the history of commands used   
history > out.txt : to save the history into a file as input  
  
“>” : this allows to add something from source to destination but the destination is overwritten with the contents of3e source  
“>>” : to overcome the problem of overwriting, we use the double greater than sign This preserves the content of the destination file



Tasks of the DAY

* Rename one of the files.
* Create multiple Files in single command
* Combine multiple files in a single command and redirect their output to a single file
* Create a directory and a few text files in it.

DAY 2

NUMBER SYSTEMS

1. Unary
2. Binary
3. Ternery
4. Octal
5. Decimal
6. Hexadecimal

Hexadecimal:

Representation of numbers using base 10, ranging from 0 to 9

Binary:

Representation of numbers using 2 bits 0 and 1, or base 2

Task: write the range, and representations of bits ranging from 1 to 8

No. series range number of sets

1 0,1 (0 - 1) 2

2 0,1,2,3 (0 - 3) 4

3 0,1,2,3,...7 (0 - 7) 8

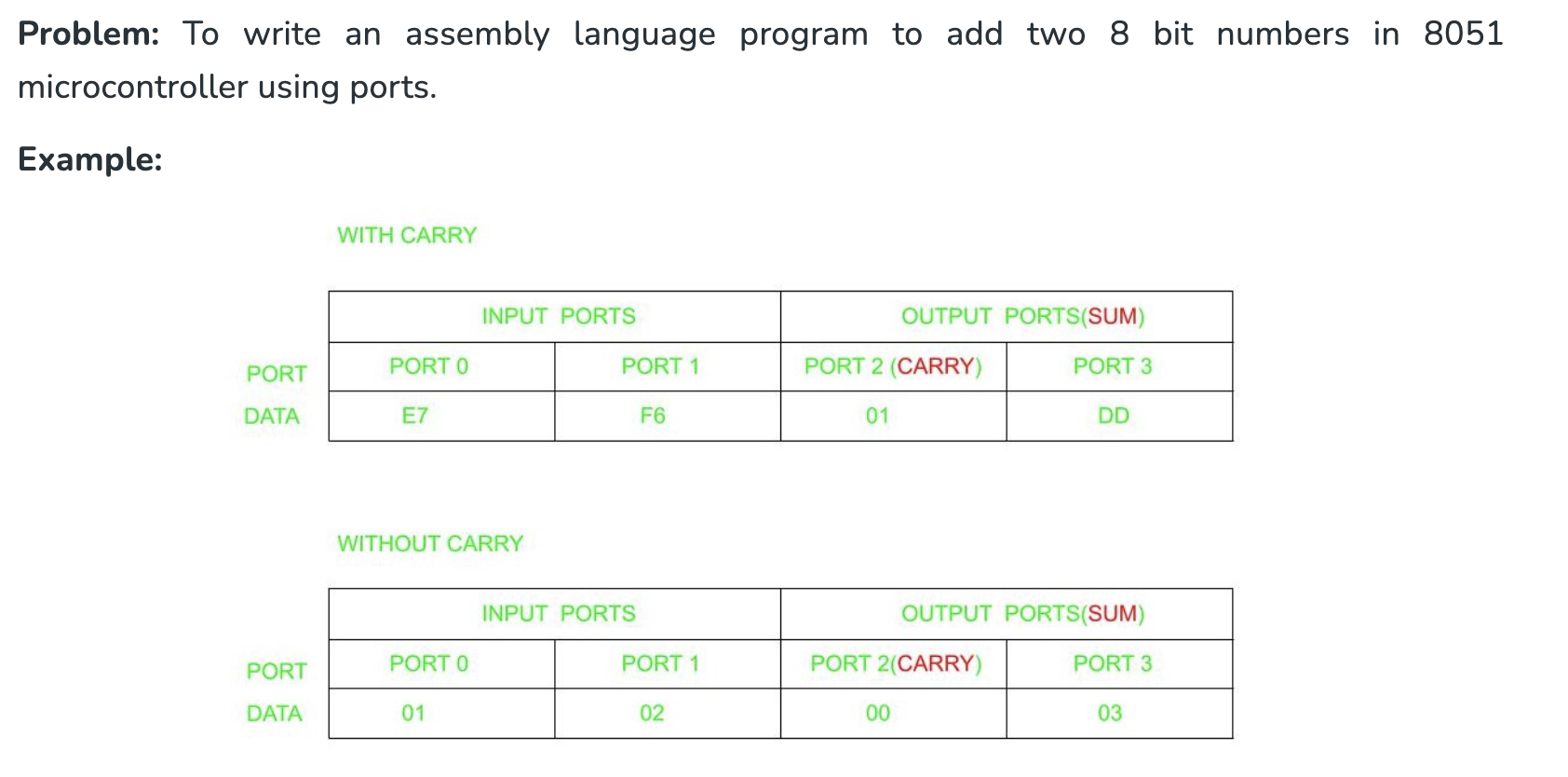
4 0,1,2,3,...15 (0 - 15) 16

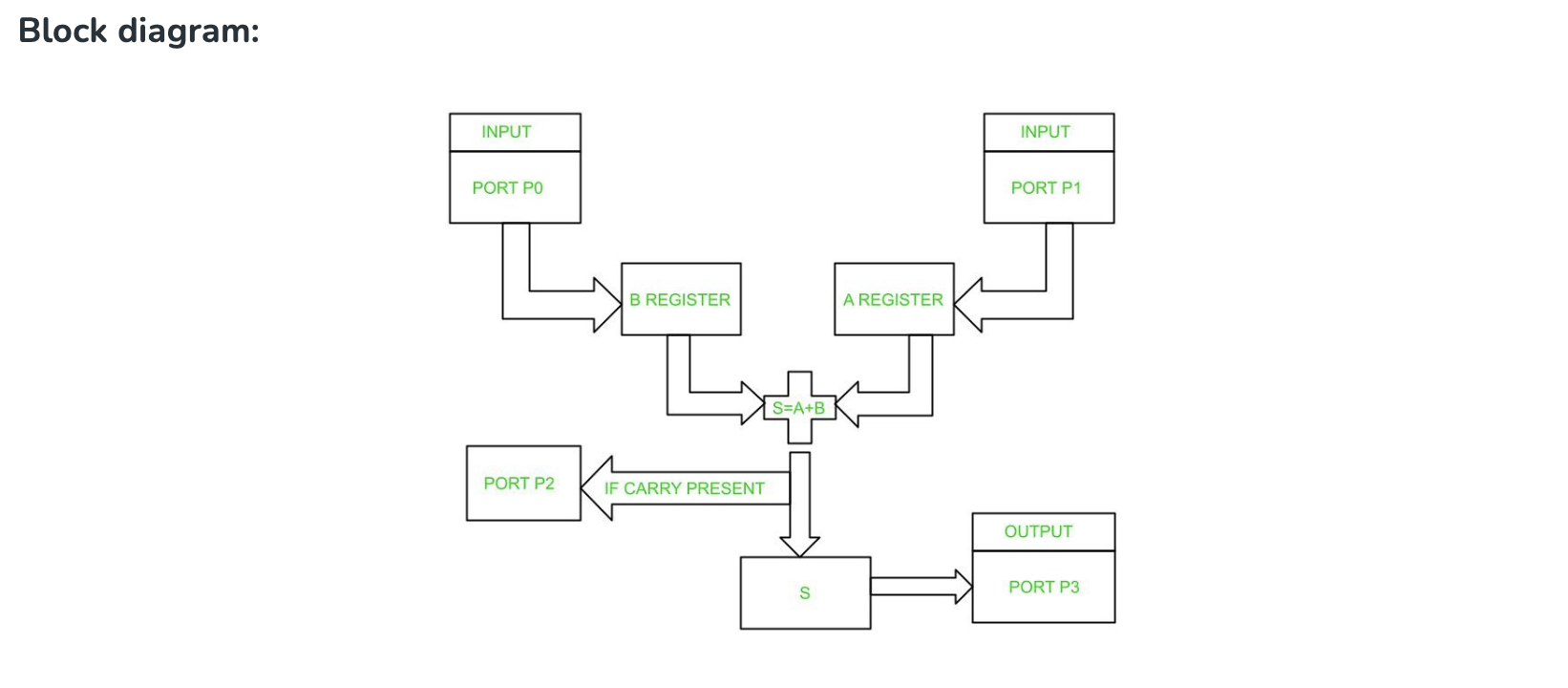
5 0,1,2,3,...31 (0 - 31) 32

6 0,1,2,3,...63 (0 - 63) 64

7 0,1,2,3,...127 (0 - 127) 128

8 0,1,2,3,...255 (0 - 255) 256





**Algorithm:**

* Initialize Ports P0 and P1 as input ports.
* Initialize Ports P2 and P3 as output ports.
* Initialize the R1 register.
* Move the contents from Port 0 to B register.
* Move the contents from Port 1 to A register.
* Add contents in A and B.
* If carry is present increment R1.
* Move contents in R1 to Port 2.
* Move the sum in step 6 to Port 3.

**PROGRAM**

ORG 00H // Indicates starting address

MOV P0,#0FFH // Initializes P0 as input port

MOV P1,#0FFH // Initializes P1 as input port

MOV P2,#00H // Initializes P2 as output port

MOV P3,#00H // Initializes P3 as output port

L1:MOV R1, #00H // Initializes Register R1

MOV B,P0 // Moves content of P0 to B

MOV A,P1 // Moves content of P1 to A

CLR C // Clears carry flag

ADD A,B // Add the content of A and B and store result in A

JNC L2 // If carry is not set, jump to label L2

INC R1 // Increment Register R1 if carry present

L2: MOV P2, R1 // Moves the content from Register R1 to Port2

MOV P3,A // Moves the content from A to Port3

SJMP L1 // Jumps to label L1

END

Here's an 8051 assembly code example to multiply two 8-bit numbers, with the result stored in 16-bit registers:

; Assume 'num1' and 'num2' are 8-bit numbers stored in memory locations 0x40 and 0x41 respectively

; The 16-bit result will be stored in R2 (low byte) and R1 (high byte)

**PROGRAM**

MOV R0, #0x40 ; Set R0 to point to num1

MOV R1, #0x41 ; Set R1 to point to num2

MOV A, [R0] ; Move num1 into the accumulator(A)

MOV B, [R1] ; Move num2 into register B

MUL AB ; Multiply the contents of A and B,  
 storing the 16-bit result

; (low byte in A, high byte in B)

MOV R2, A ; Store the low byte of the result in R2

MOV R1, B ; Store the high byte of the result in R1

|  | |
| --- | --- |