Department Of Software Engineering

Computer Architecture & logic design

Lab Manual

Submitted to:

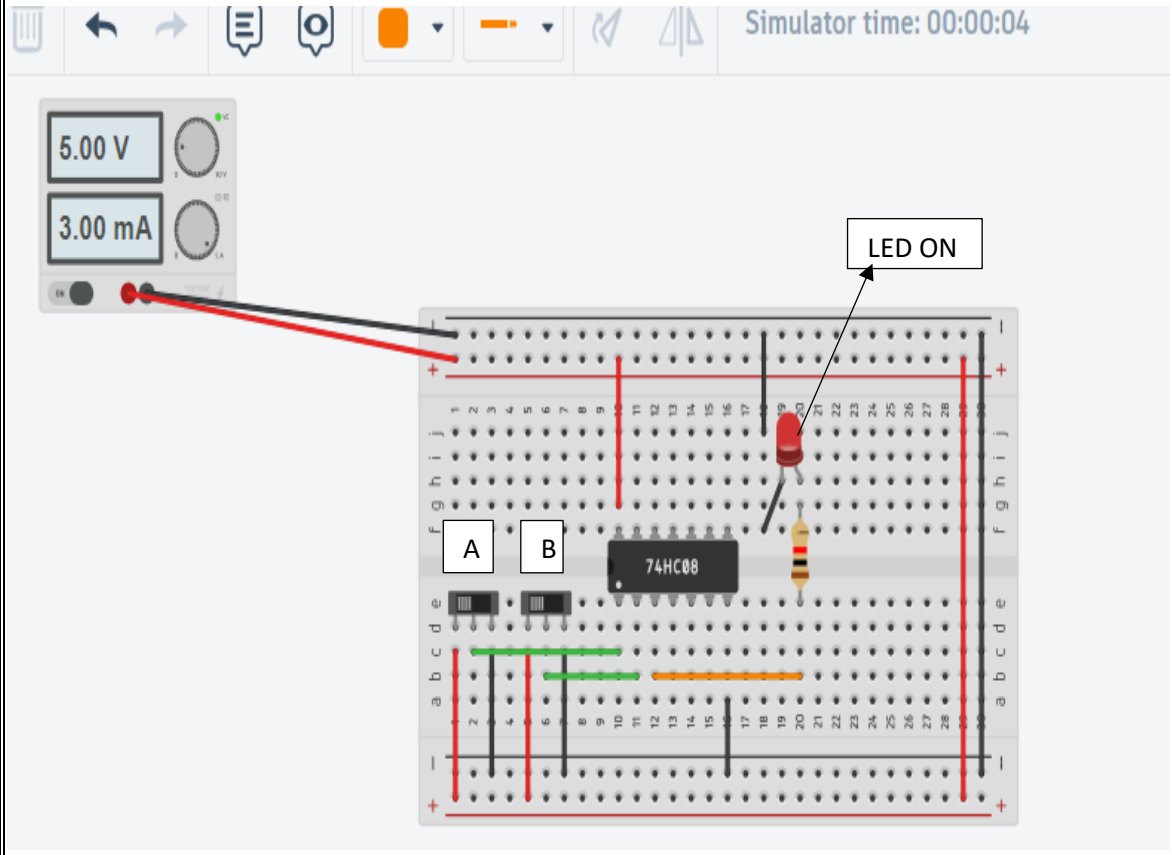
Sir Shoaib

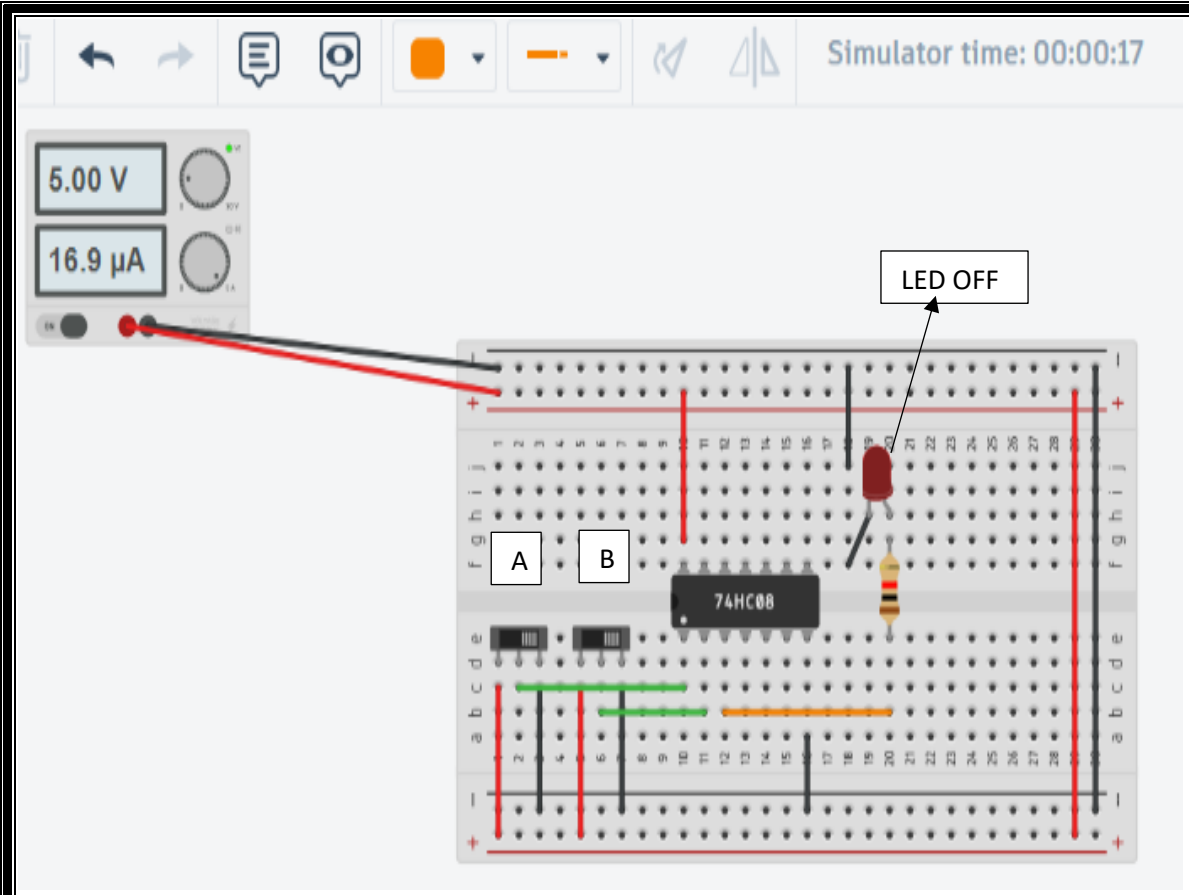
Submitted by:

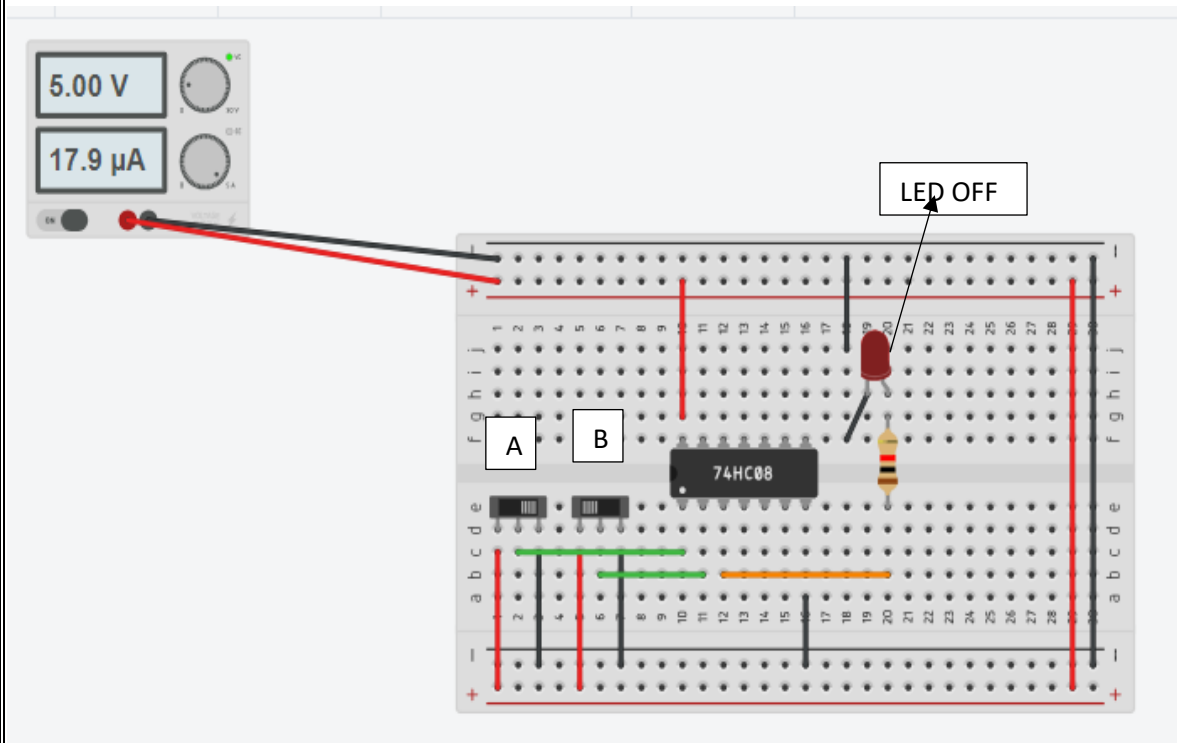
Rabia batool (2022-BSE-064)

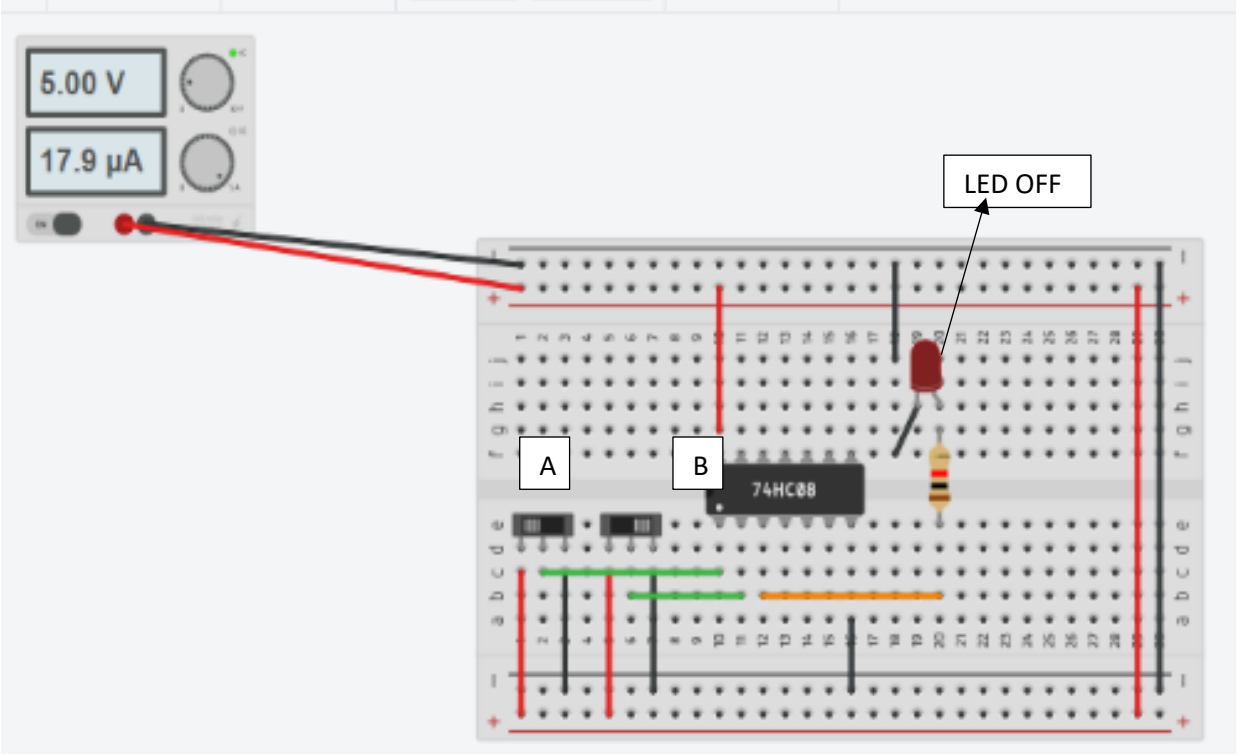
FAMILIARIZATION WITH LOGIC TRAINER AND VERIFICATION OF TRUTH TABLE OF BASIC LOGIC GATES.

AND GATE:

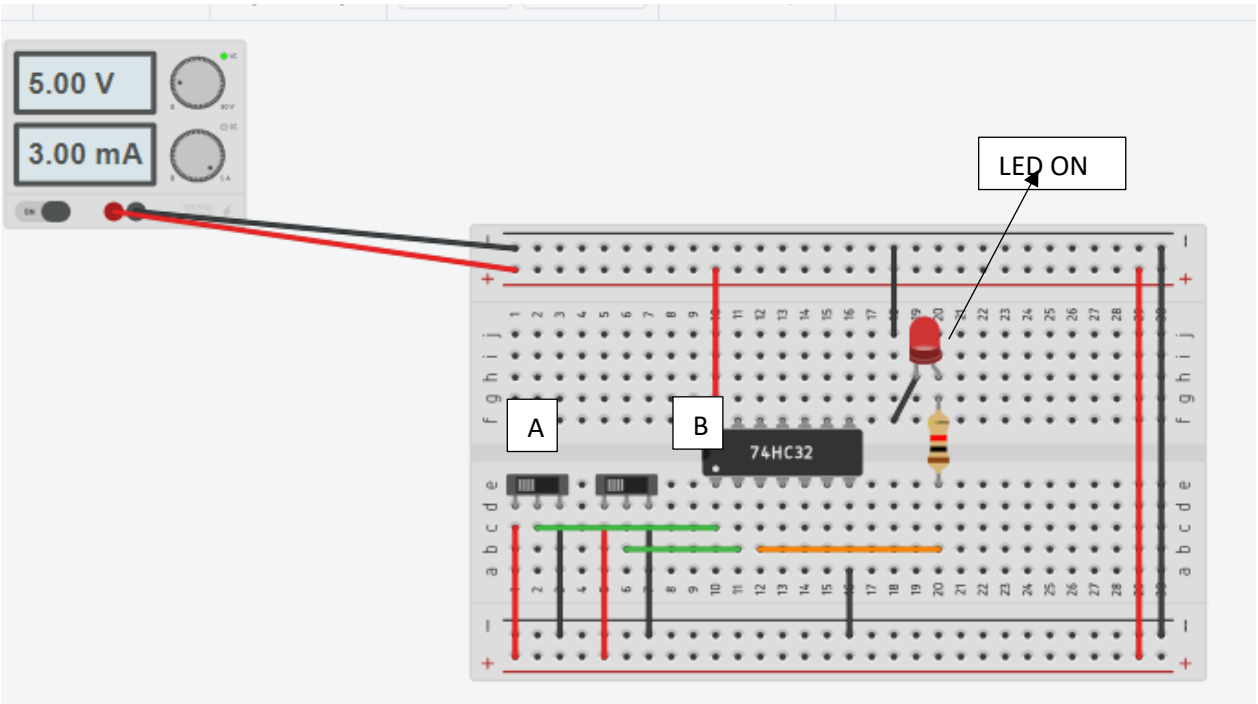


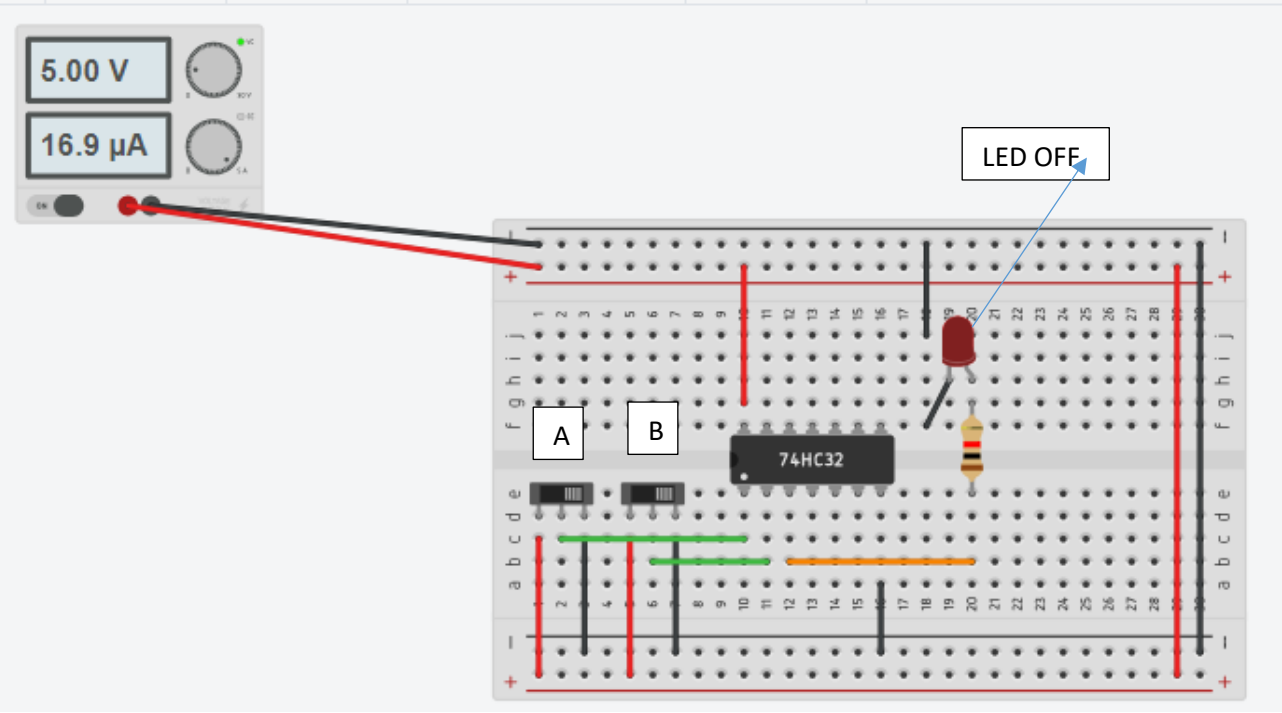

Simulator time: 00:00:17
5.00 V
16.9 µA
LED OFF
A
B
74HC08

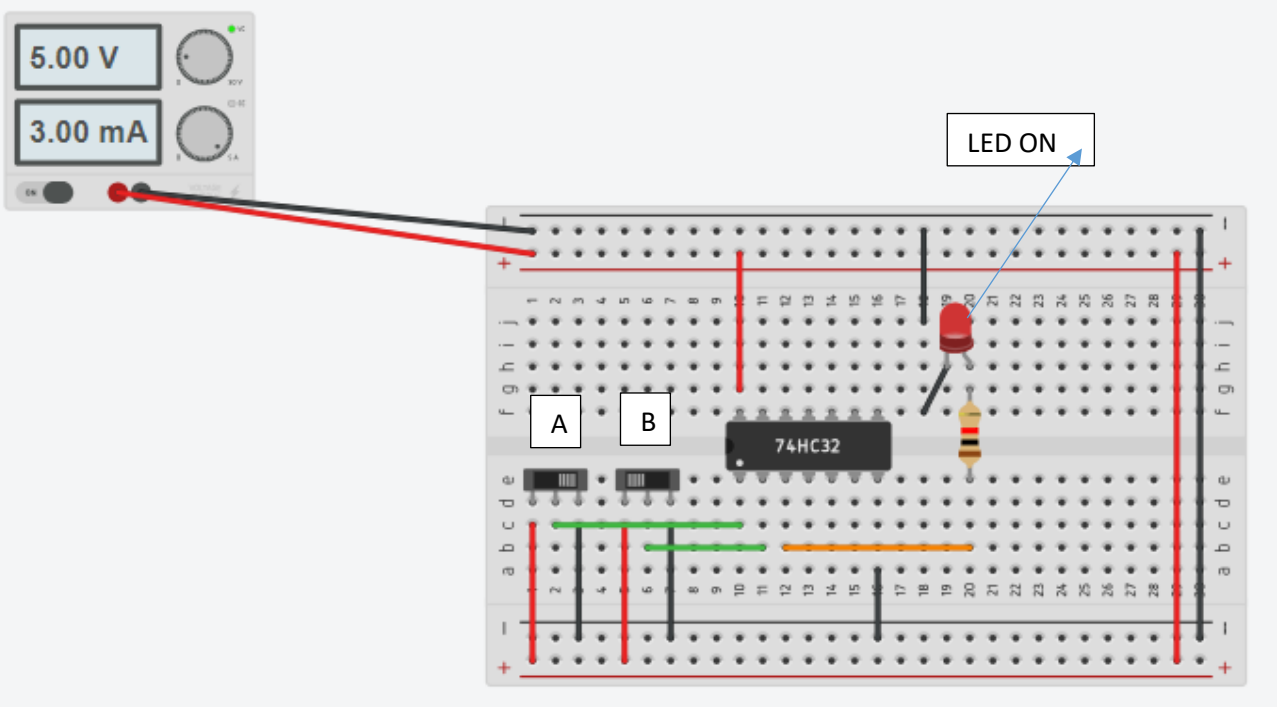

5.00 V
17.9 µA
LED OFF
A
B
74HC08

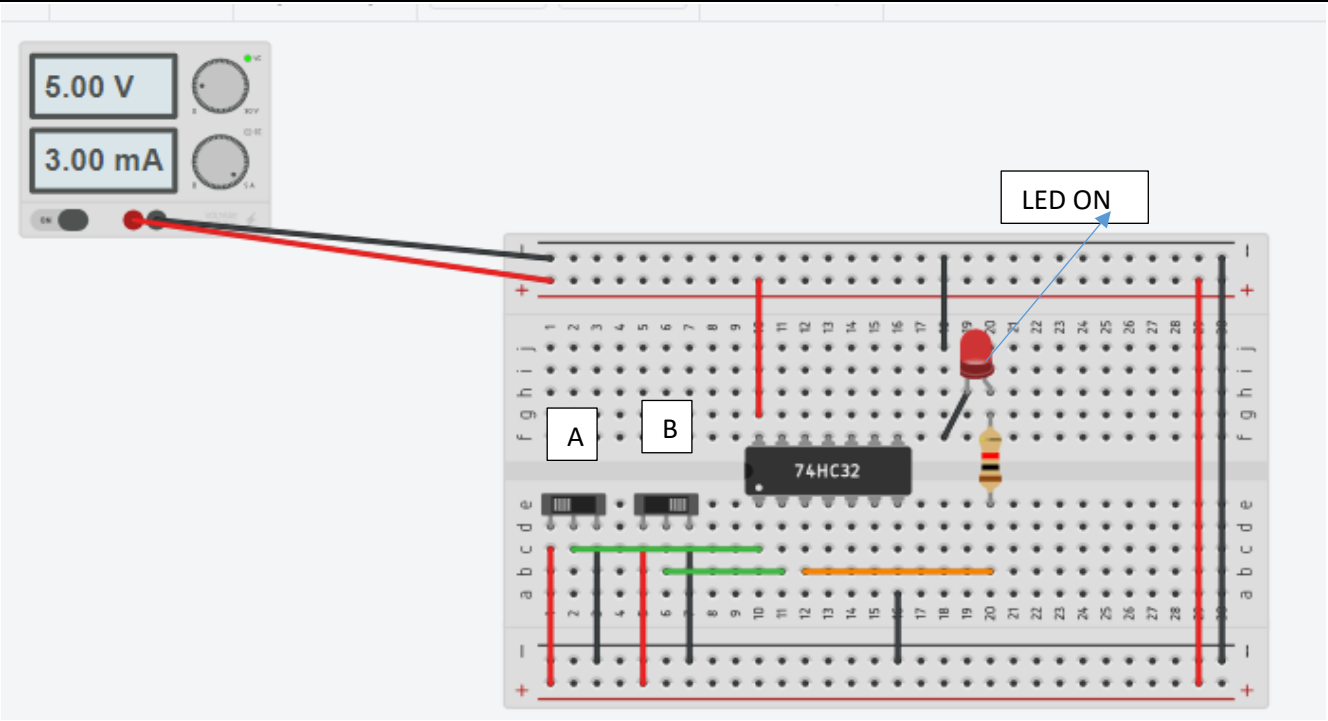


OR GATE:

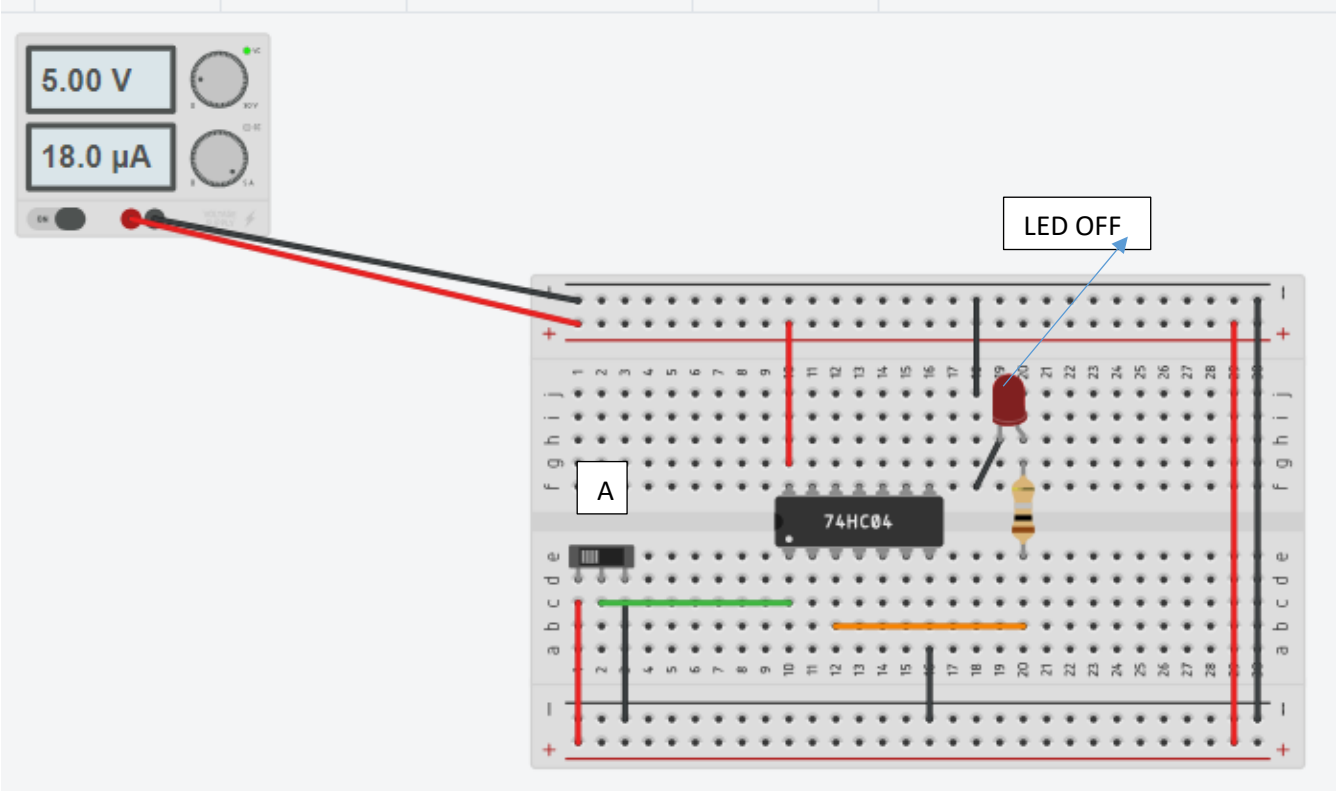


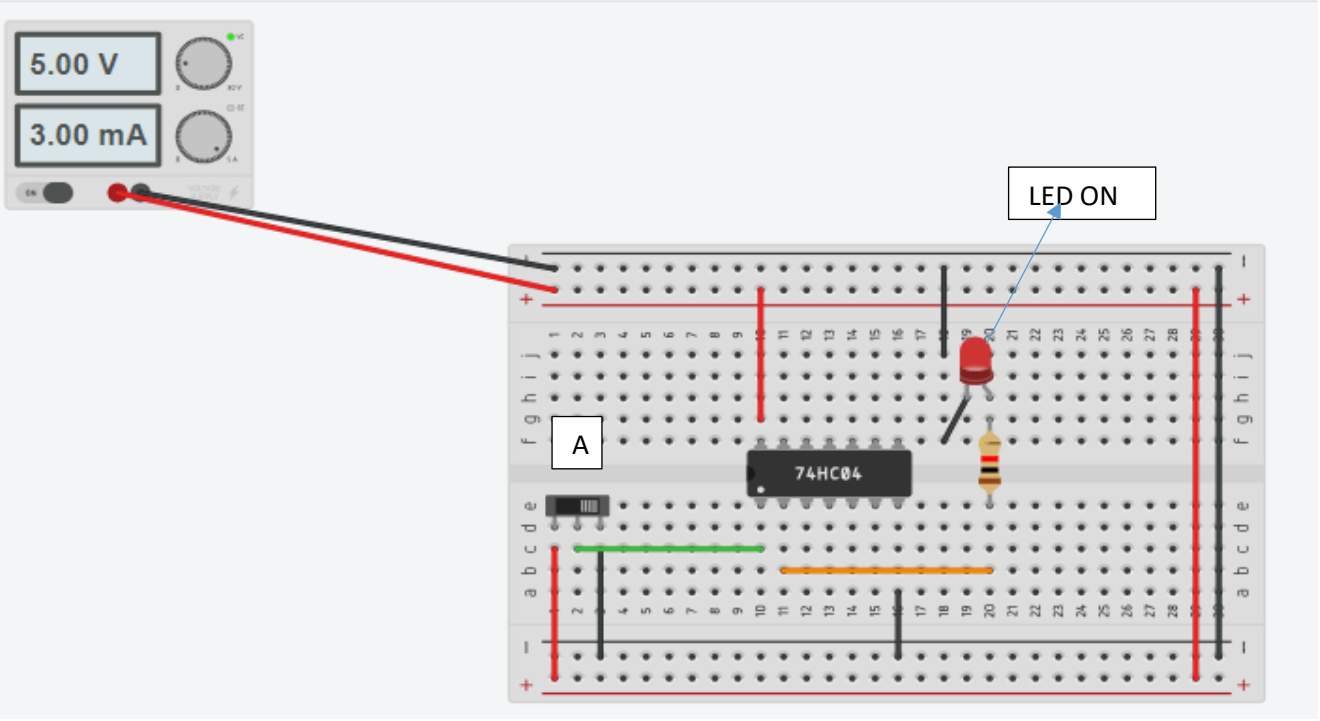

5.00 V
16.9 µA
LED OFF
A
B
74HC32


5.00 V
3.00 mA
LED ON
A
B
74HC32

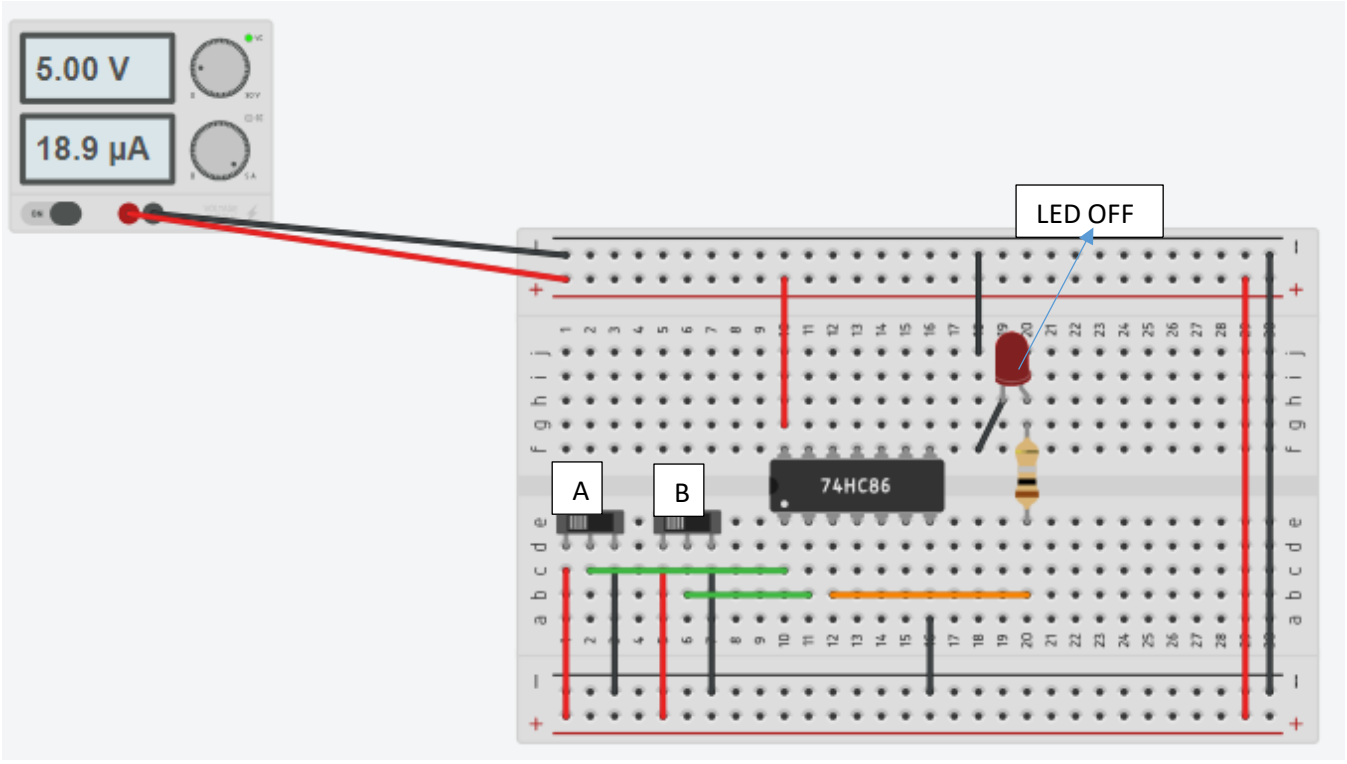


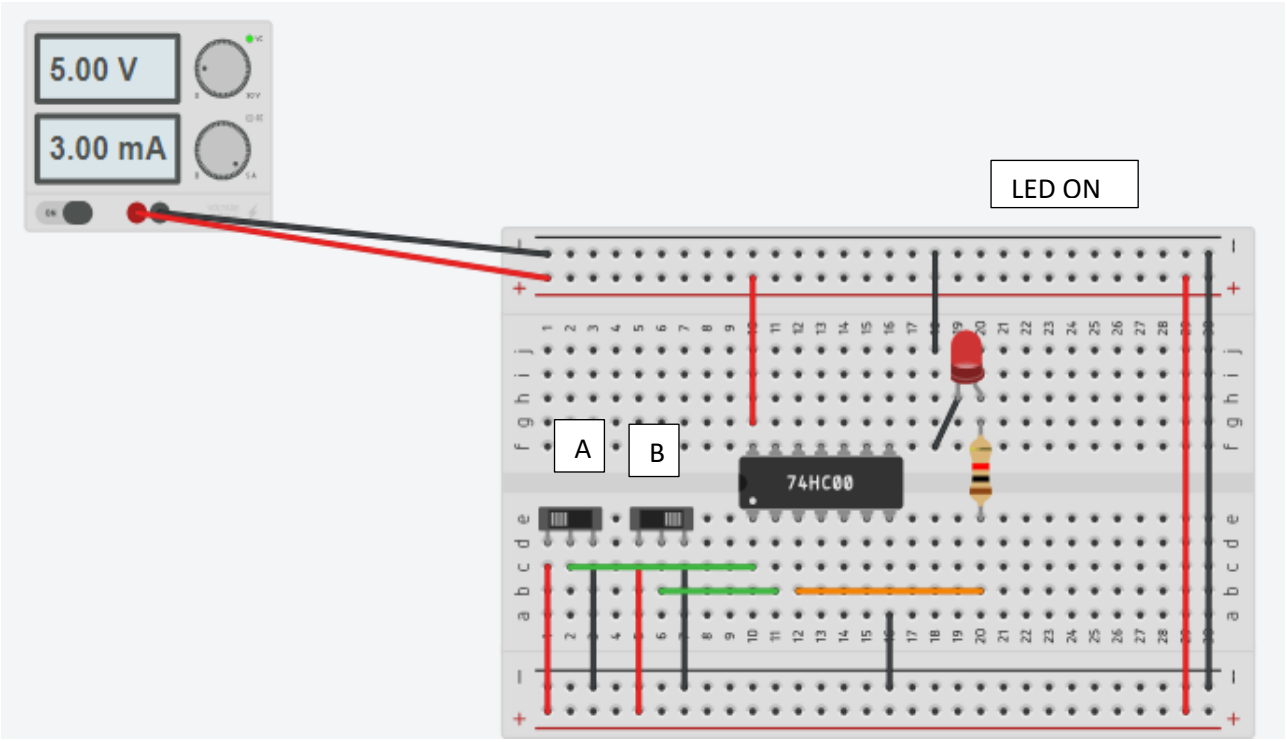
NOT GATE:

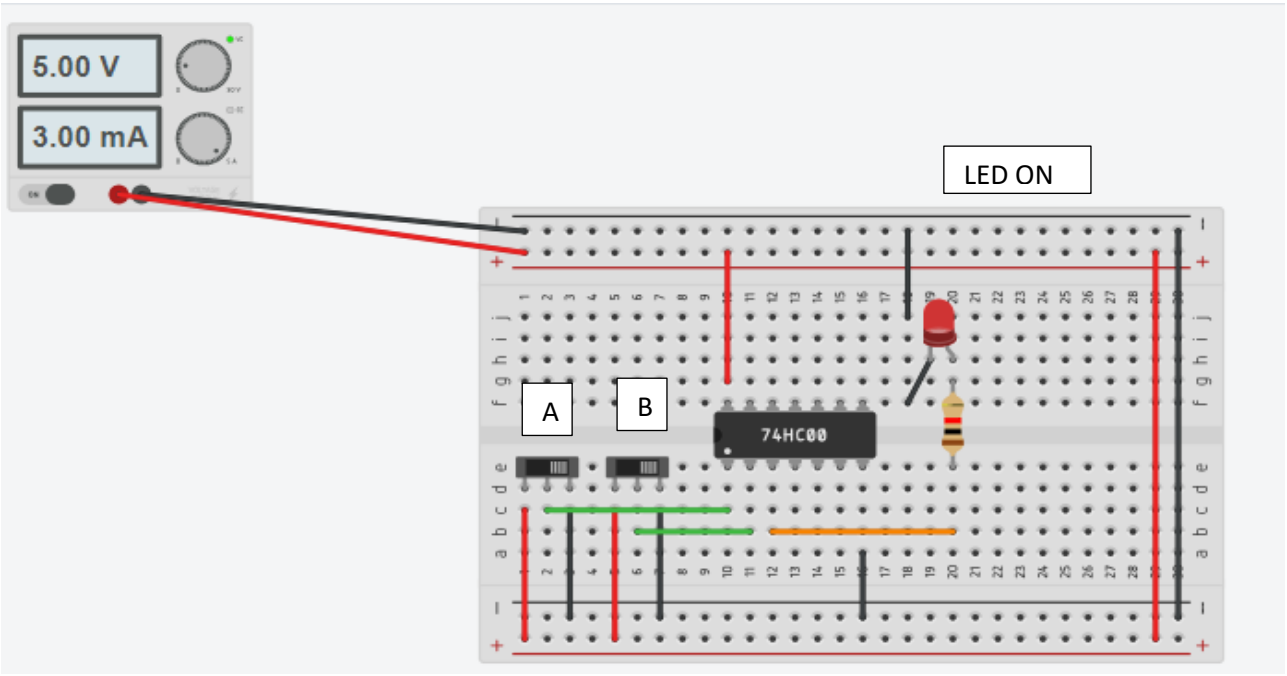


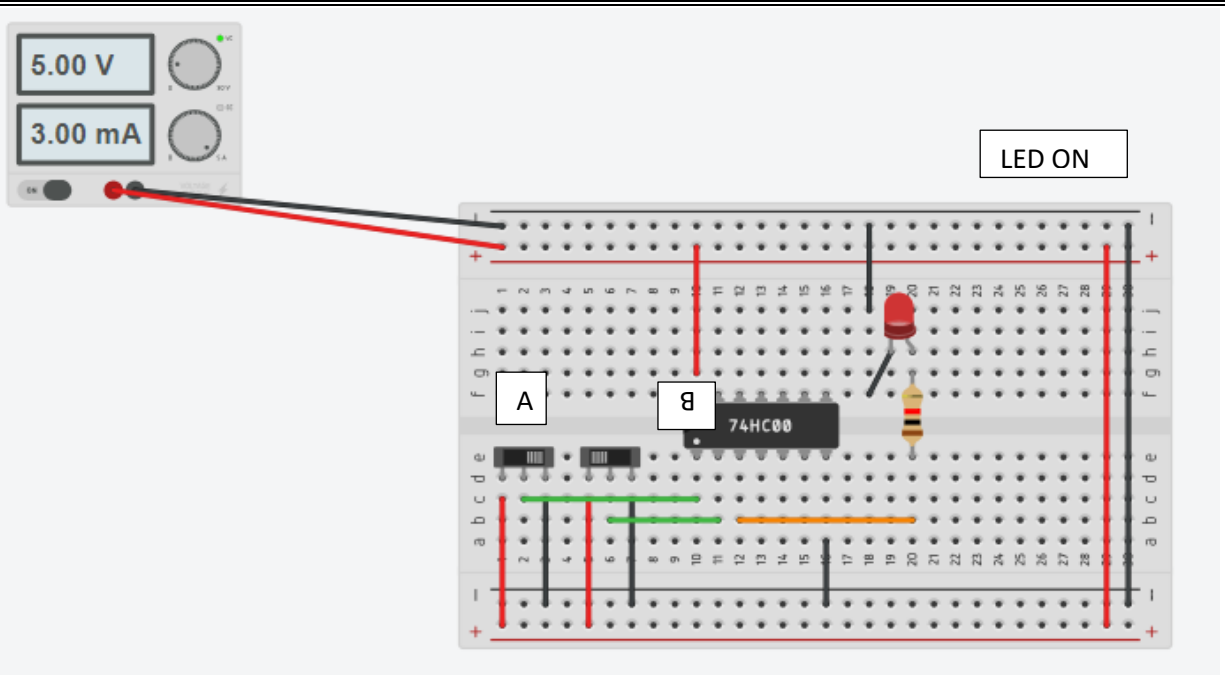


NAND GATE:

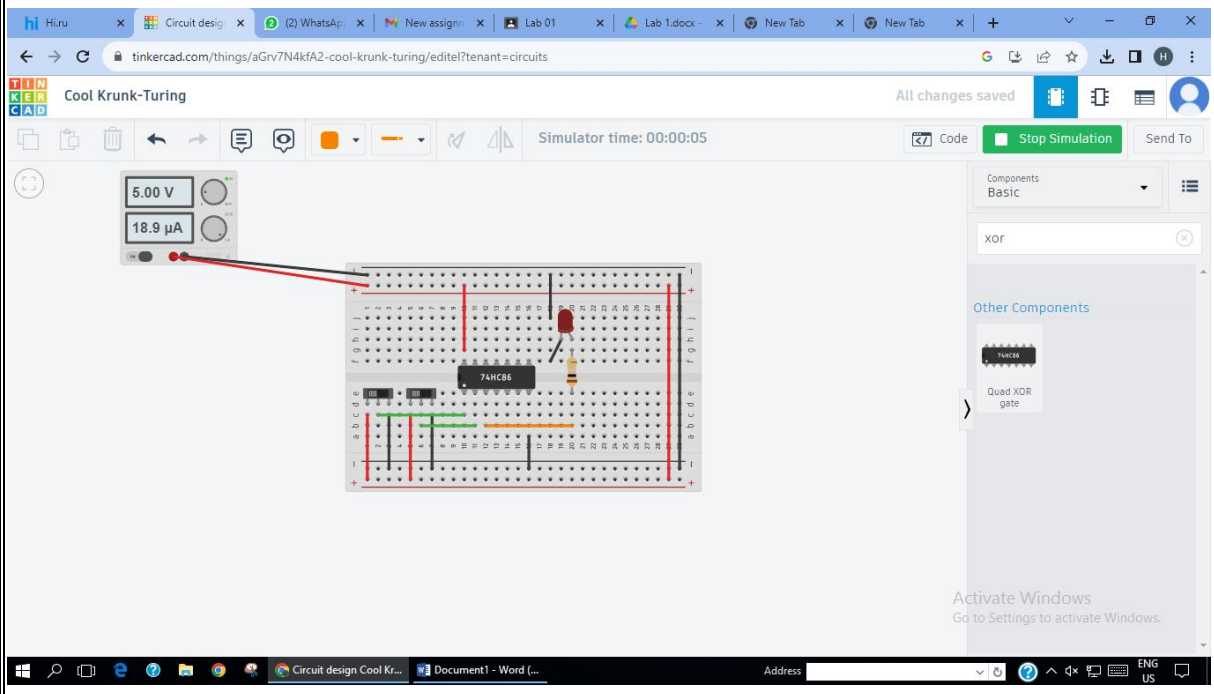


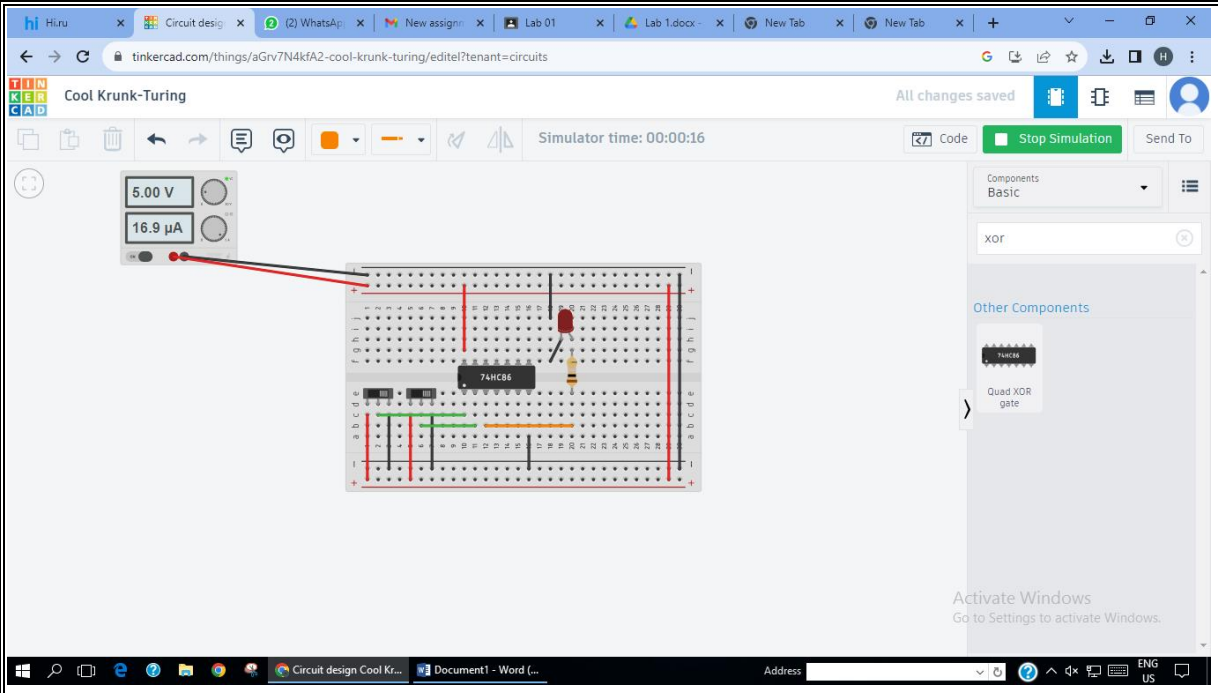

5.00 V
3.00 mA
LED ON
A
B
74HC00

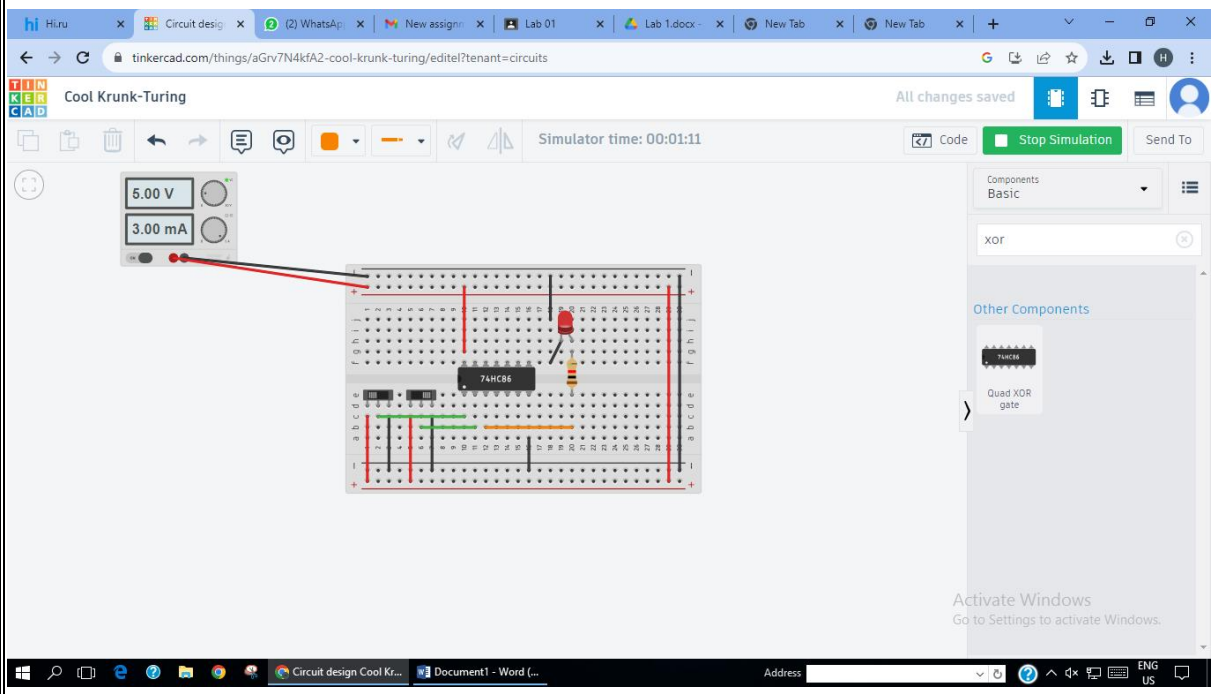

5.00 V
3.00 mA
LED ON
A
B
74HC00

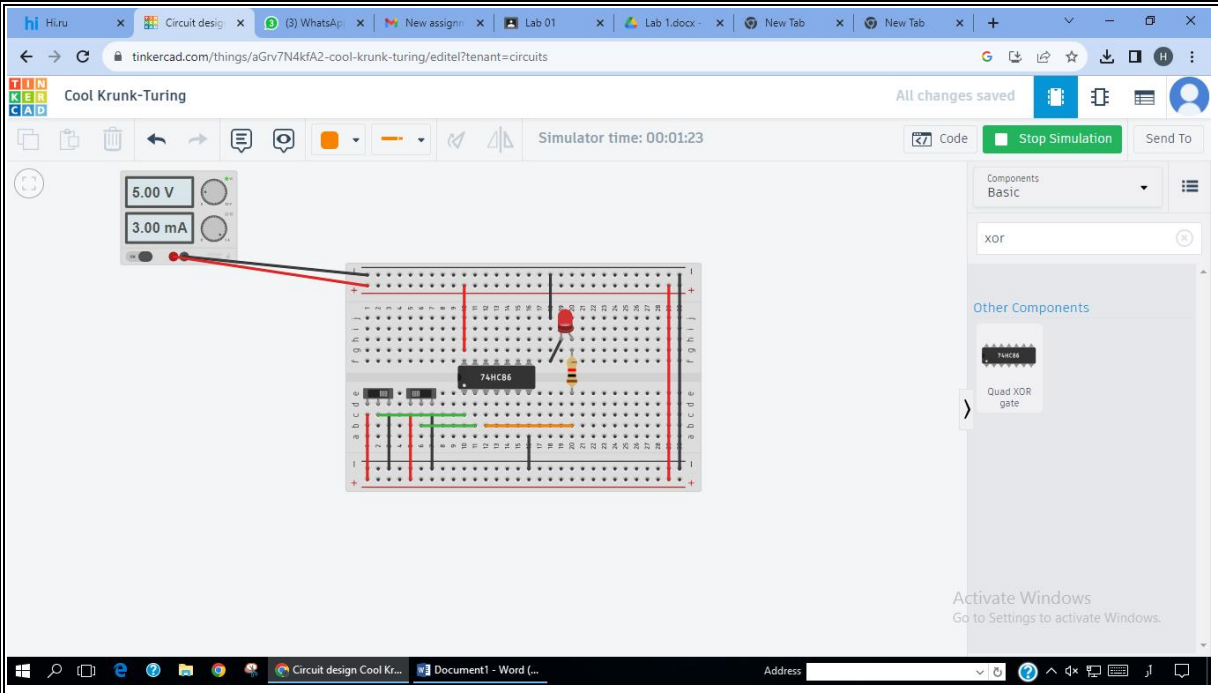

5.00 V
3.00 mA
LED ON
A
B
74HC00

EX-OR

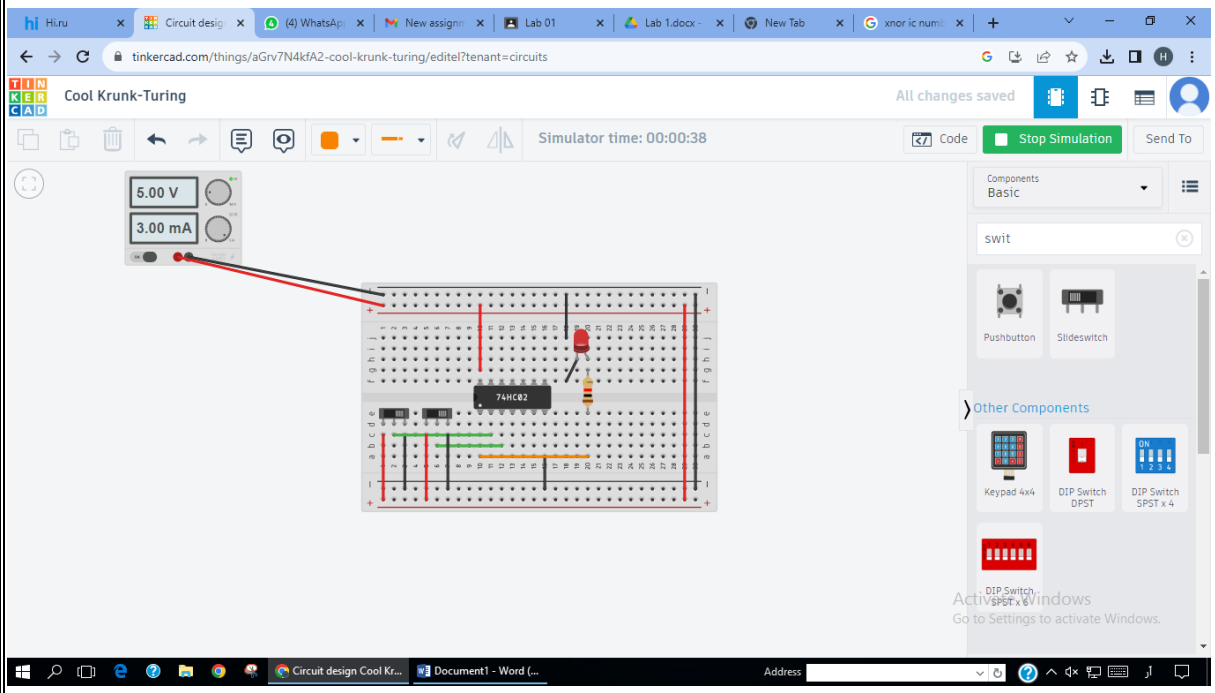

tinkercad.com/things/aGrv7N4kfA2-cool-krunk-turing/editel?tenant=circuits
Cool Krunk-Turing
All changes saved
Simulator time: 00:00:05
Code
Stop Simulation
Send To
5.00 V
18.9 µA
74HC86
Components
Basic
xor
Other Components
Quad XOR gate
Activate Windows
Go to Settings to activate Windows.

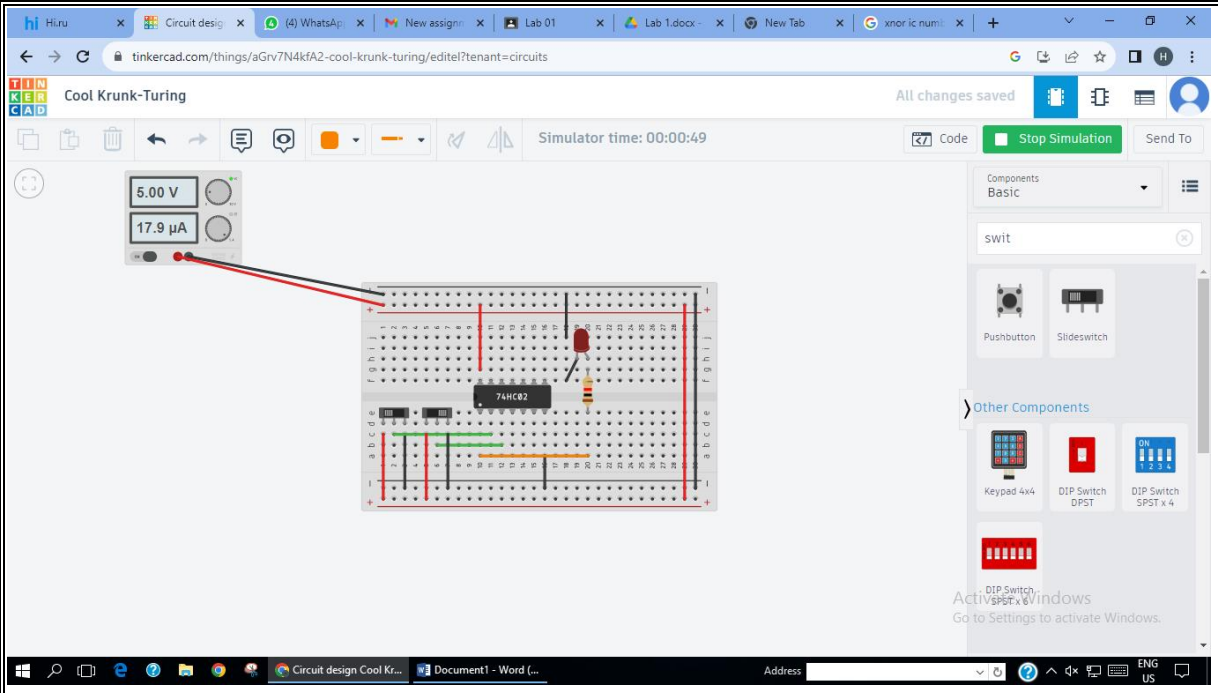

Cool Krunk-Turing
All changes saved
Simulator time: 00:00:16
Code
Stop Simulation
Send To
5.00 V
16.9 µA
74HC86
Components
Basic
xor
Other Components
Quad XOR gate
Activate Windows
Go to Settings to activate Windows.

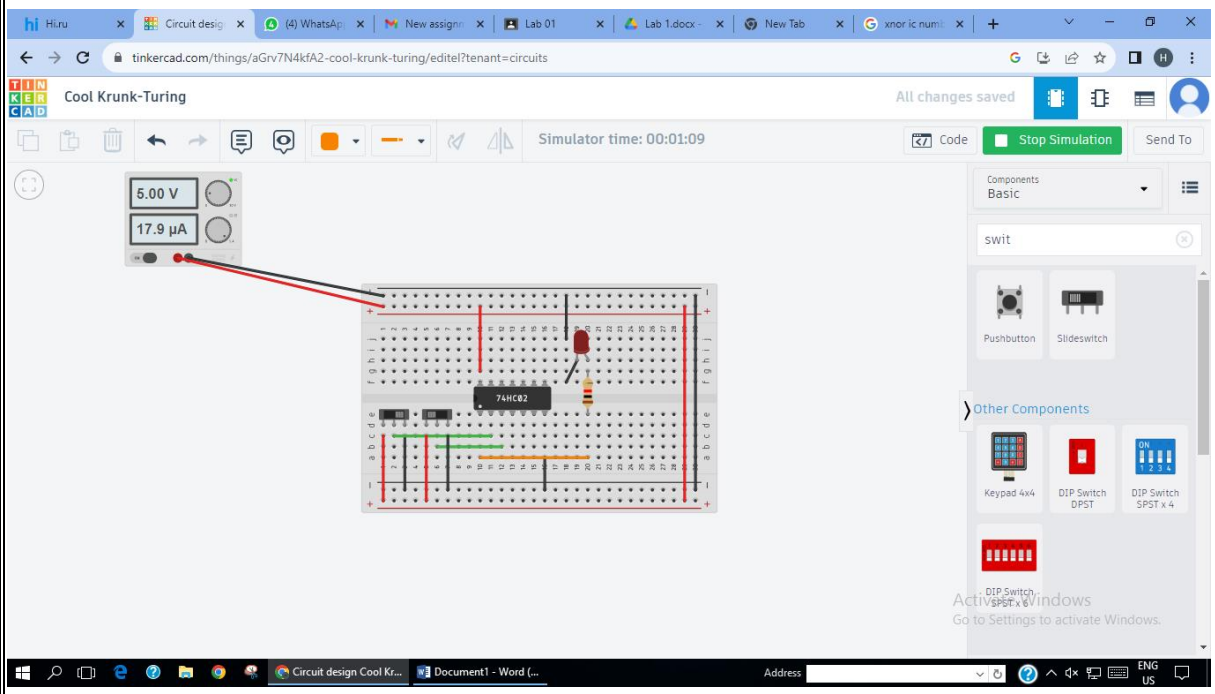

Cool Krunk-Turing
All changes saved
Simulator time: 00:01:11
Code
Stop Simulation
Send To
5.00 V
3.00 mA
74HC86
Components
Basic
xor
Other Components
Quad XOR gate
Activate Windows
Go to Settings to activate Windows.

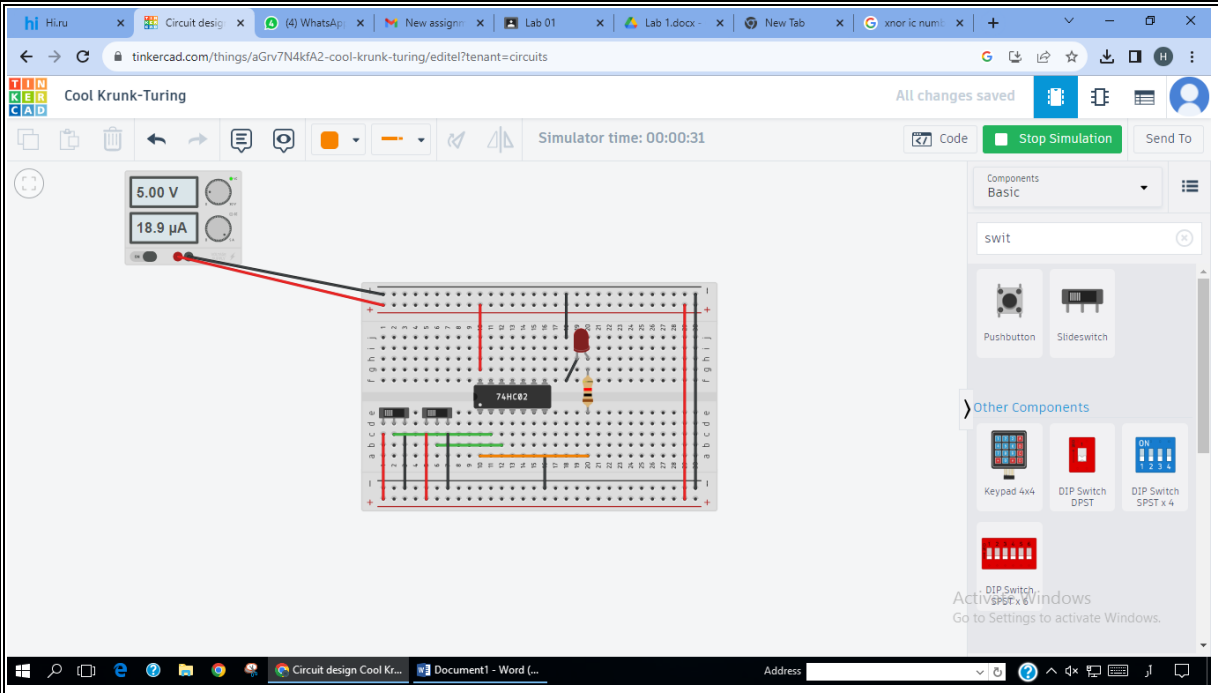

Cool Krunk-Turing
All changes saved
Simulator time: 00:01:23
Code
Stop Simulation
Send To
5.00 V
3.00 mA
Components
Basic
xor
Other Components
74HC86
Quad XOR gate
Activate Windows
Go to Settings to activate Windows.

NOR GATE:


Cool Krunk-Turing
All changes saved
Simulator time: 00:00:38
Code
Stop Simulation
Send To
5.00 V
3.00 mA
Components
Basic
swit
Pushbutton
Slideswitch
Other Components
Keypad 4x4
DIP Switch DPST
DIP Switch SPST x 4
DIP Switch SPST x 6
74HC02
Activate Windows
Go to Settings to activate Windows.


Cool Krunk-Turing
All changes saved
Simulator time: 00:00:49
Code
Stop Simulation
Send To
5.00 V
17.9 µA
74HC02
Components
Basic
swit
Pushbutton
Slideswitch
Other Components
Keypad 4x4
DIP Switch DPST
DIP Switch SPST x 4
DIP Switch SPST x 6
Activate Windows
Go to Settings to activate Windows.


Cool Krunk-Turing
All changes saved
Simulator time: 00:01:09
Code
Stop Simulation
Send To
5.00 V
17.9 µA
74HC02
Components
Basic
swit
Pushbutton
Slideswitch
Other Components
Keypad 4x4
DIP Switch DPST
DIP Switch SPST x 4
DIP Switch SPST x 6
Activate Windows
Go to Settings to activate Windows.



Fatima Jinnah Women University

Sawaira saeed

2022-BSE-067

Rabia Batool

2022-BSE-064

Hafsah tahir

2022-BSE-052

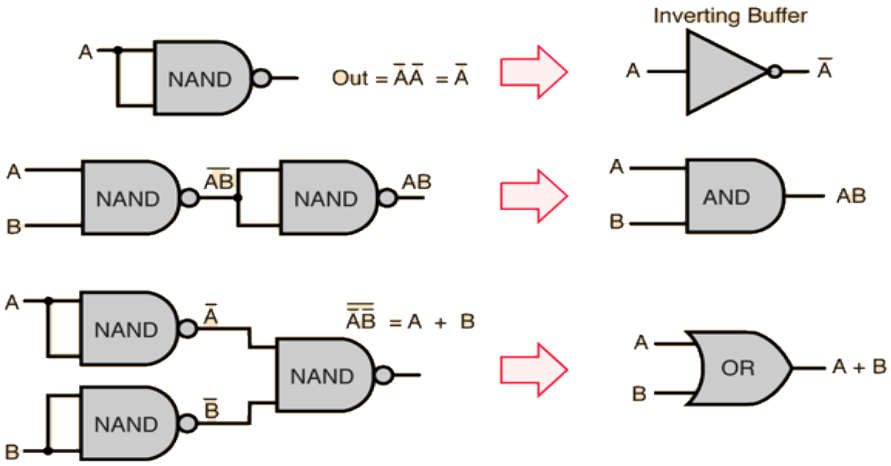
Group#B

Computer architecture and logic design

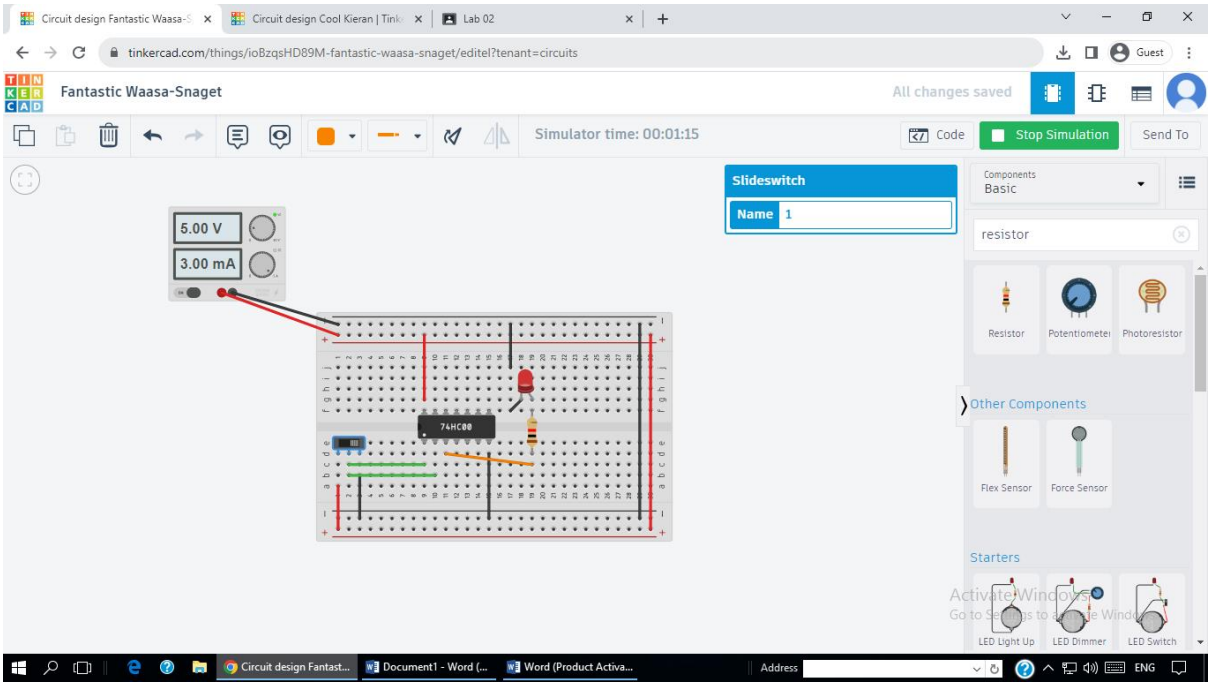
LAB#02

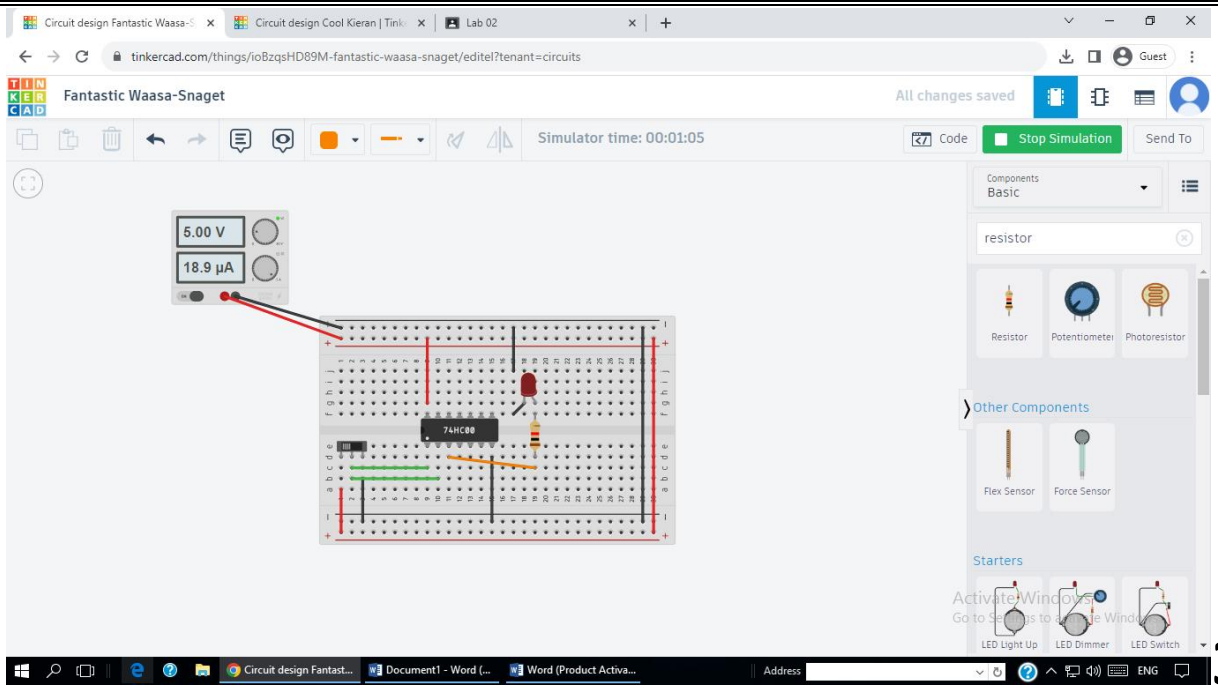
Submitted to Sir Shoaib

3.1. Working with NAND Gate:



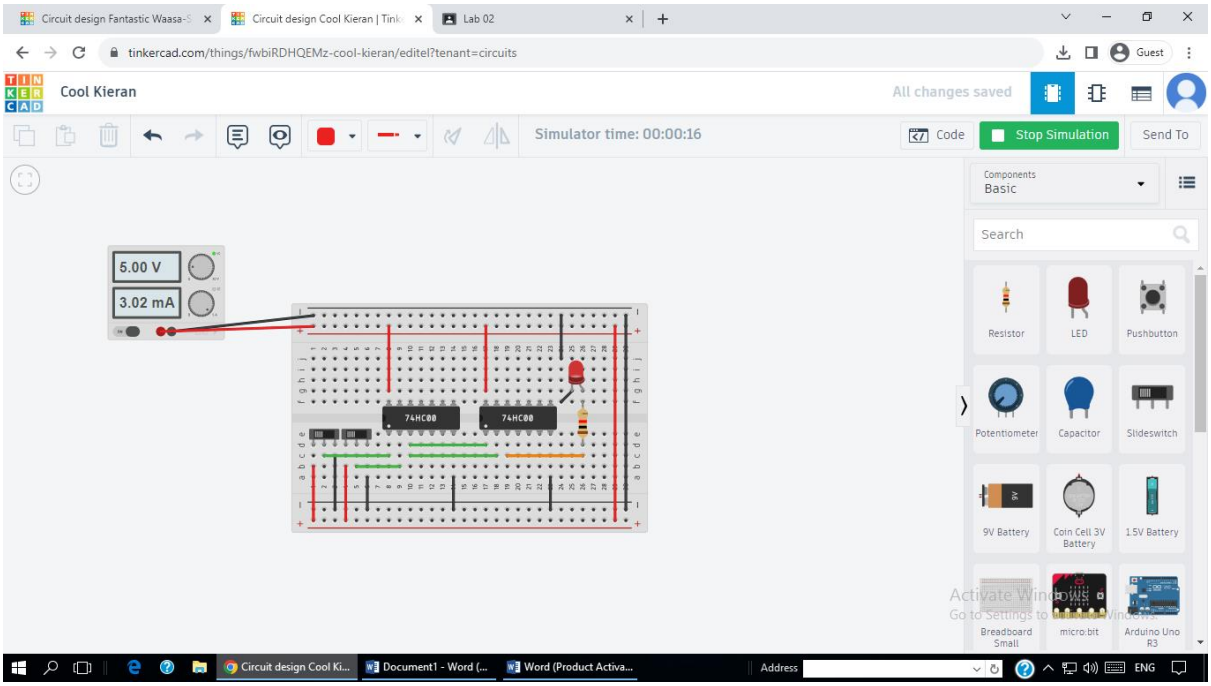
: 1a)- NAND to NOT

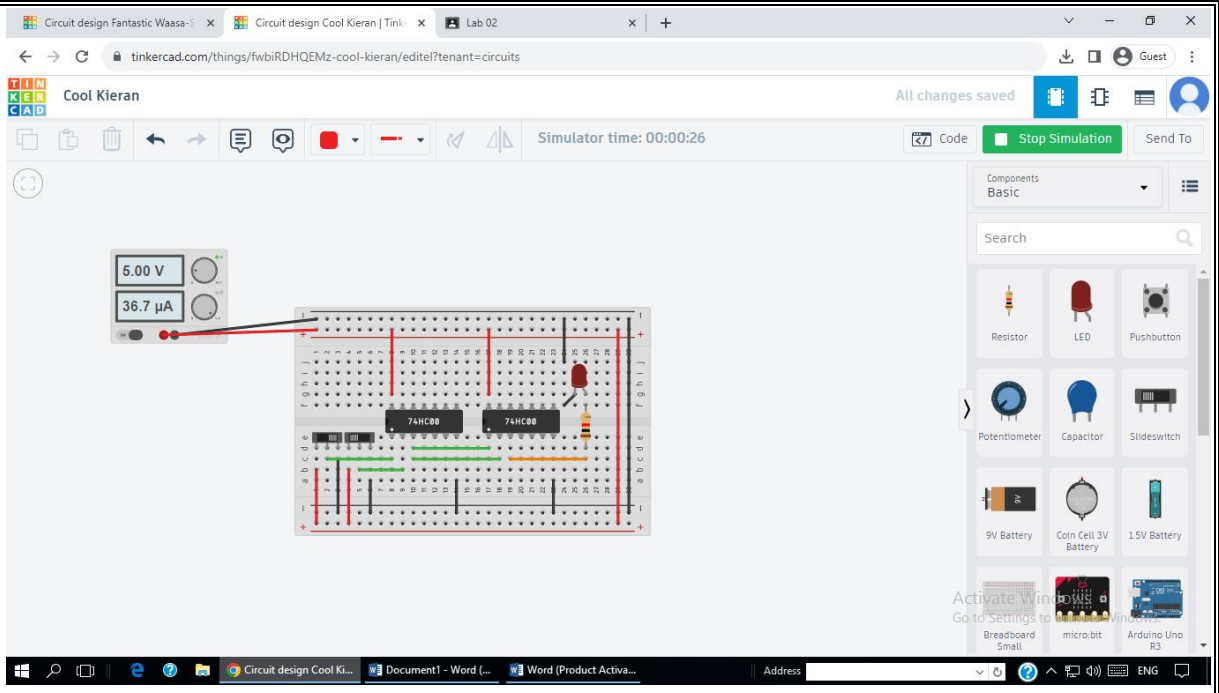


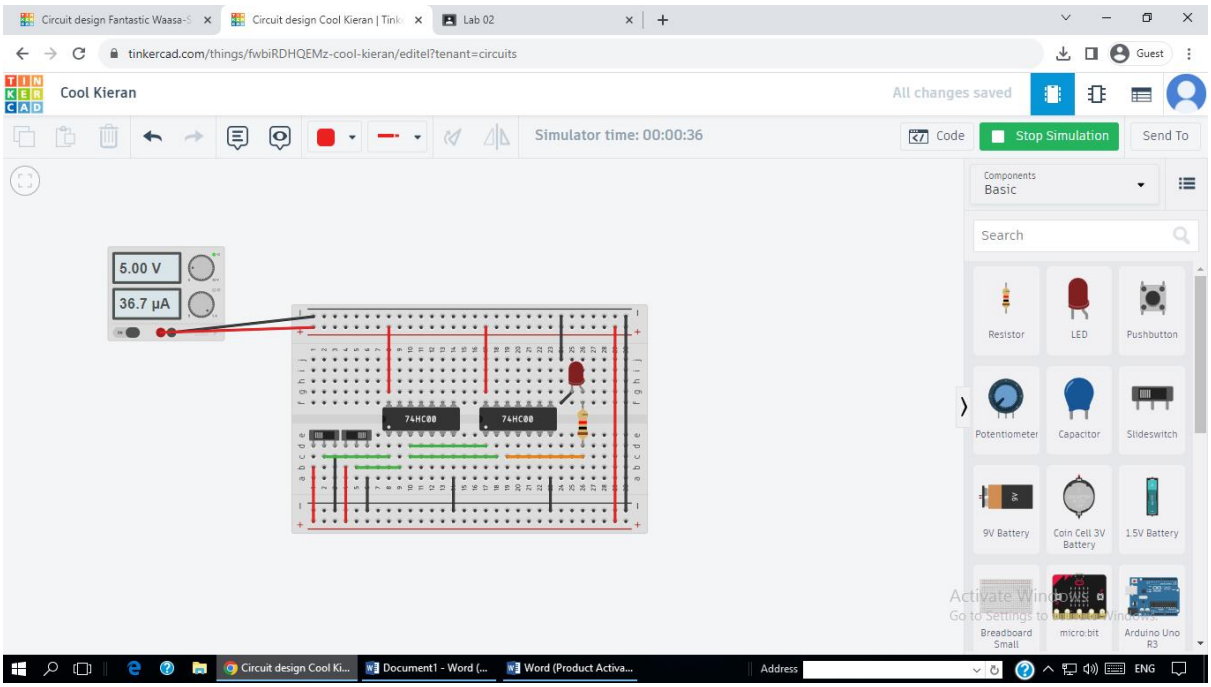


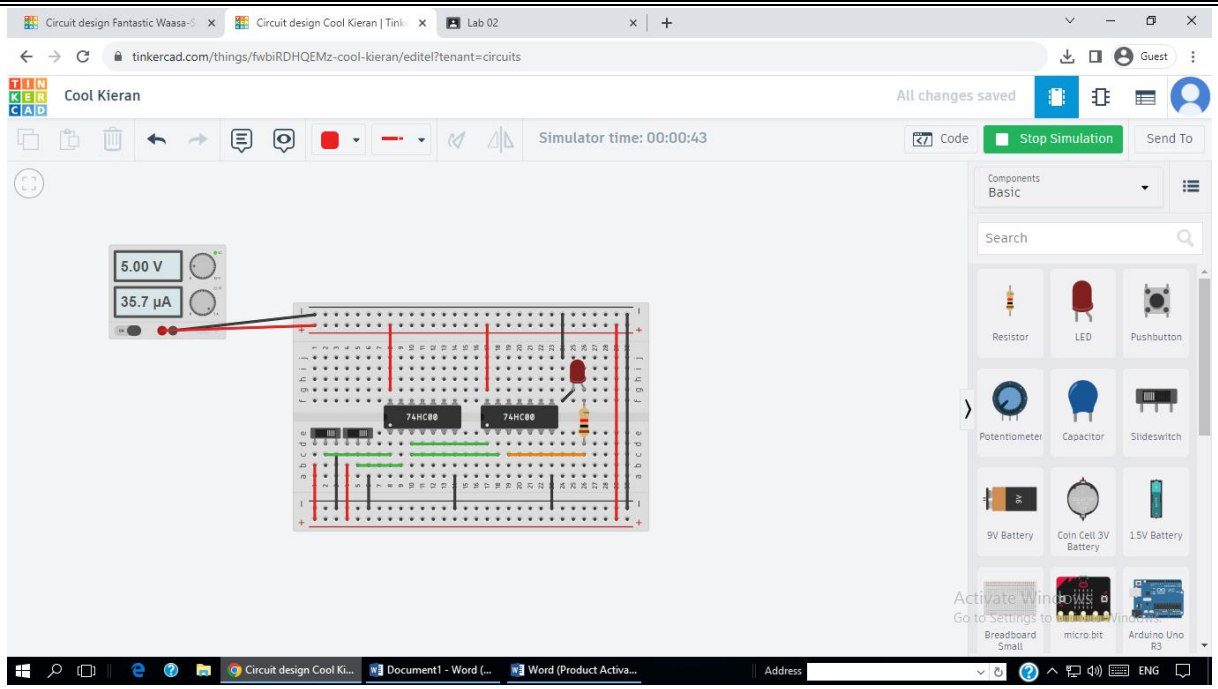
3b)

NAND to AND



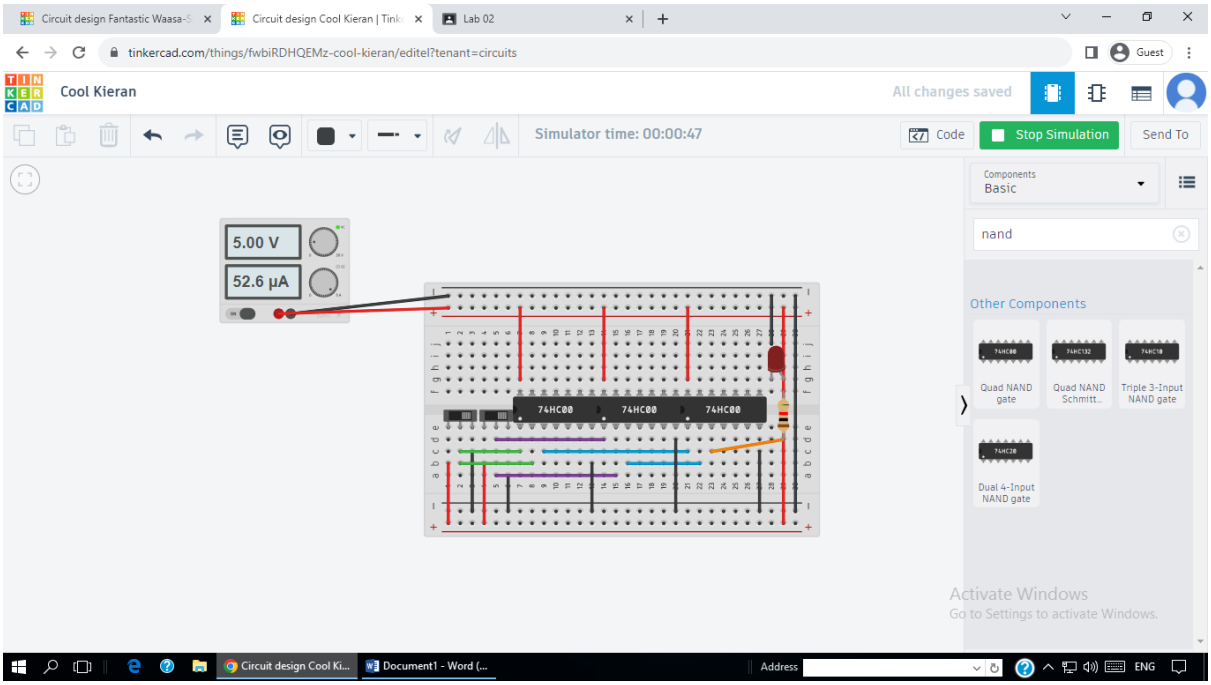

Circuit design Fantastic Waasa-S
Circuit design Cool Kieran | Tink
Lab 02
tinkercad.com/things/fwbiRDHQEMz-cool-kieran/editel?tenant=circuits
Guest
Cool Kieran
All changes saved
Simulator time: 00:00:26
Code
Stop Simulation
Send To
Components
Basic
Search
5.00 V
36.7 µA
74HC00
74HC00
Resistor
LED
Pushbutton
Potentiometer
Capacitor
Slideswitch
9V Battery
Coin Cell 3V Battery
1.5V Battery
Breadboard Small
micro:bit
Arduino Uno R3
Activate Windows
Go to Settings to activate Windows.
Circuit design Cool Ki...
Document1 - Word (...
Word (Product Activa...
Address
ENG

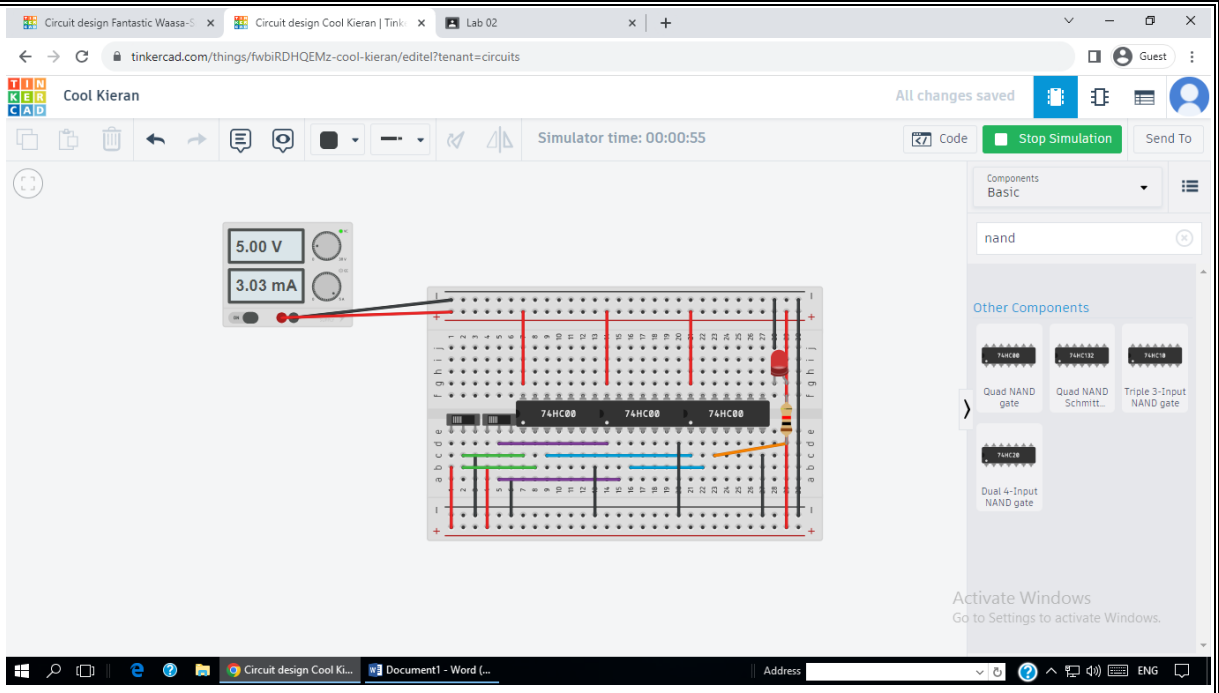

Circuit design Fantastic Waasa-S
Circuit design Cool Kieran | Tink
Lab 02
tinkercad.com/things/fwbiRDHQEMz-cool-kieran/editel?tenant=circuits
Guest
Cool Kieran
All changes saved
Simulator time: 00:00:36
Code
Stop Simulation
Send To
Components
Basic
Search
5.00 V
36.7 µA
74HC00
74HC00
Resistor
LED
Pushbutton
Potentiometer
Capacitor
Slideswitch
9V Battery
Coin Cell 3V Battery
1.5V Battery
Breadboard Small
micro:bit
Arduino Uno R3
Activate Windows
Go to Settings to activate Windows.
Circuit design Cool Ki...
Document1 - Word (...
Word (Product Activa...
Address
ENG

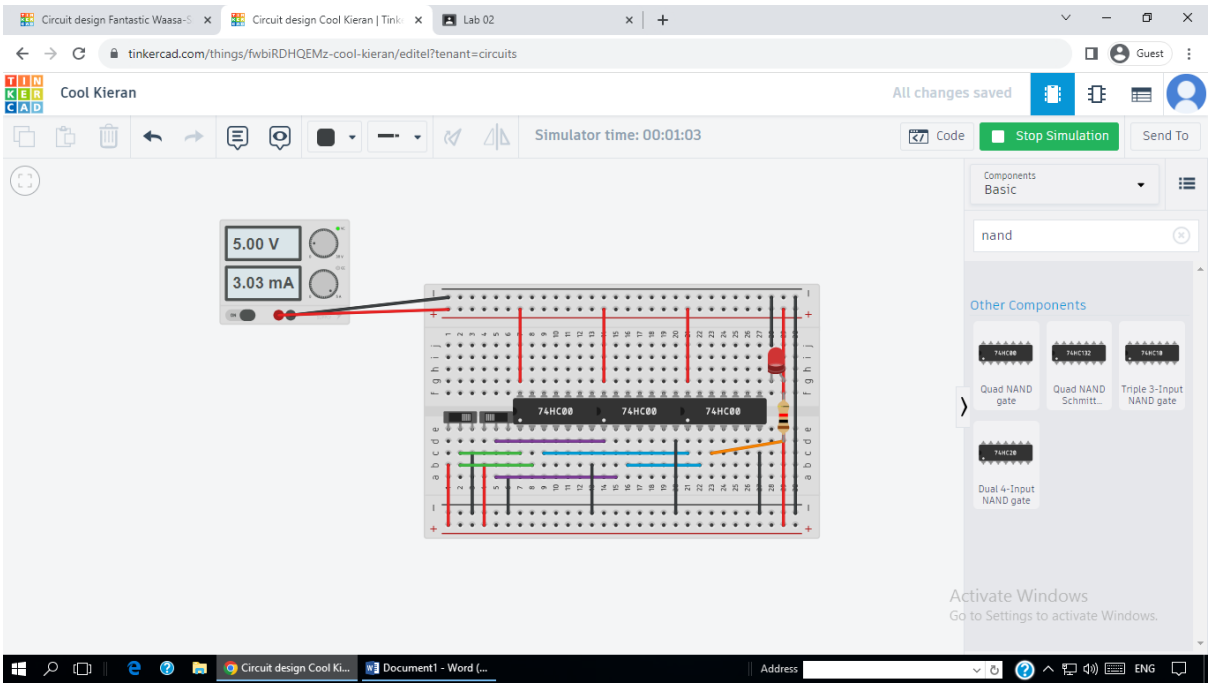

Cool Kieran
All changes saved
Simulator time: 00:00:43
Code
Stop Simulation
Send To
Components
Basic
Search
5.00 V
35.7 µA
74HC00
74HC00
Resistor
LED
Pushbutton
Potentiometer
Capacitor
Slideswitch
9V Battery
Coin Cell 3V Battery
1.5V Battery
Breadboard Small
micro:bit
Arduino Uno R3
Activate Windows
Go to Settings to activate Windows.
Circuit design Cool Ki...
Document1 - Word (...
Word (Product Activa...
Address
ENG

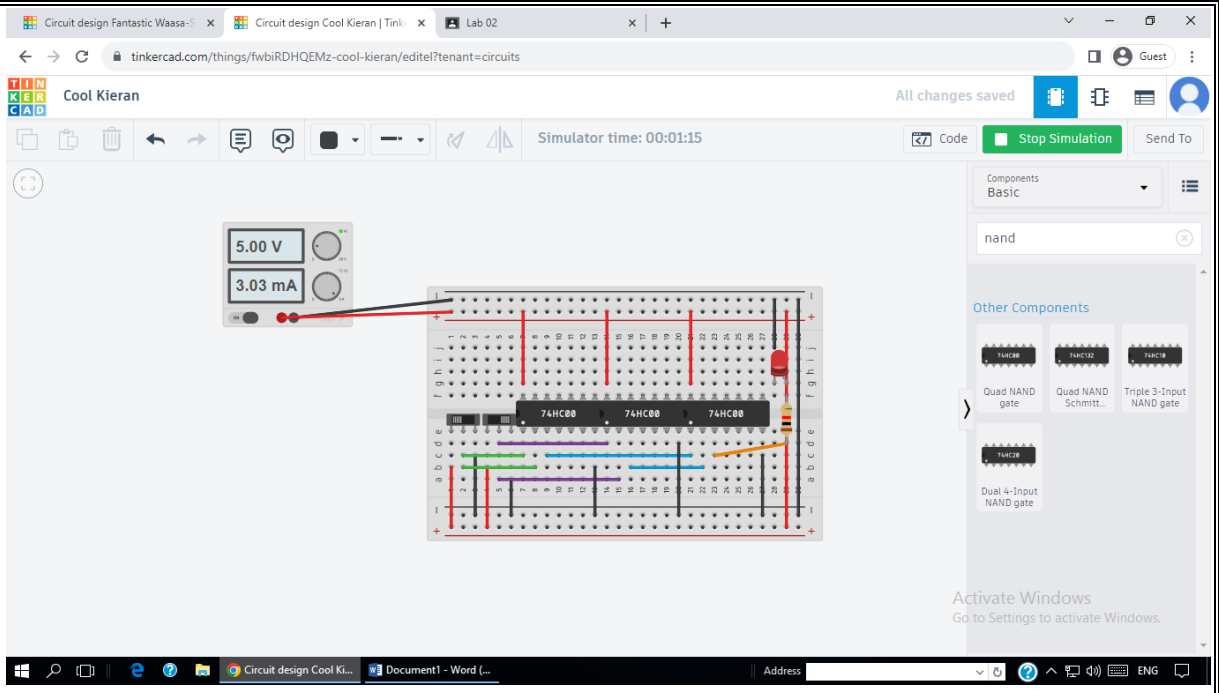
3b)

NAND to OR

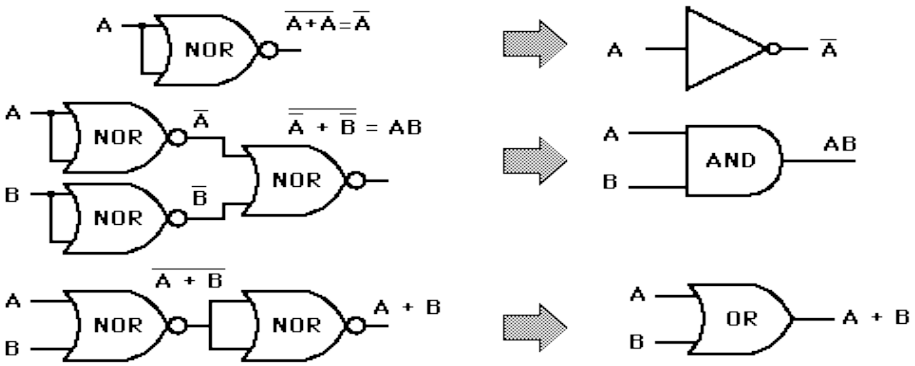

Cool Kieran
All changes saved
Simulator time: 00:00:47
Code
Stop Simulation
Send To
Components
Basic
nand
5.00 V
52.6 µA
74HC00
74HC00
74HC00
Other Components
Quad NAND gate
Quad NAND Schmitt...
Triple 3-Input NAND gate
Dual 4-Input NAND gate
Activate Windows
Go to Settings to activate Windows.
Circuit design Cool Ki...
Document1 - Word (...
Address
ENG


Circuit design Fantastic Waasa-S
Circuit design Cool Kieran | Tink
Lab 02
tinkercad.com/things/fwbiRDHQEMz-cool-kieran/editel?tenant=circuits
Guest
Cool Kieran
All changes saved
Simulator time: 00:00:55
Code
Stop Simulation
Send To
Components
Basic
nand
Other Components
5.00 V
3.03 mA
74HC00
74HC00
74HC00
Quad NAND gate
Quad NAND Schmitt...
Triple 3-Input NAND gate
Dual 4-Input NAND gate
Activate Windows
Go to Settings to activate Windows.
Circuit design Cool Ki...
Document1 - Word (...
Address
ENG

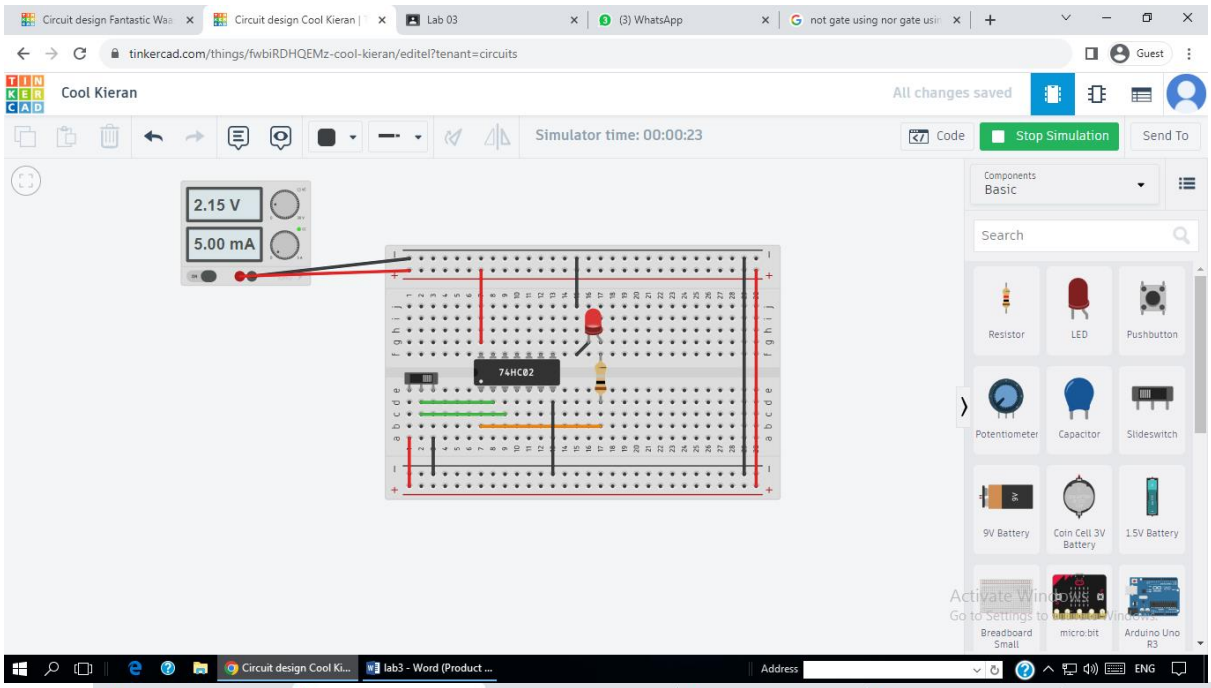

Circuit design Fantastic Waasa-S
Circuit design Cool Kieran | Tink
Lab 02
tinkercad.com/things/fwbiRDHQEMz-cool-kieran/editel?tenant=circuits
Guest
Cool Kieran
All changes saved
Simulator time: 00:01:03
Code
Stop Simulation
Send To
Components
Basic
nand
Other Components
5.00 V
3.03 mA
74HC00
74HC00
74HC00
Quad NAND gate
Quad NAND Schmitt...
Triple 3-Input NAND gate
Dual 4-Input NAND gate
Activate Windows
Go to Settings to activate Windows.
Circuit design Cool Ki...
Document1 - Word (...
Address
ENG

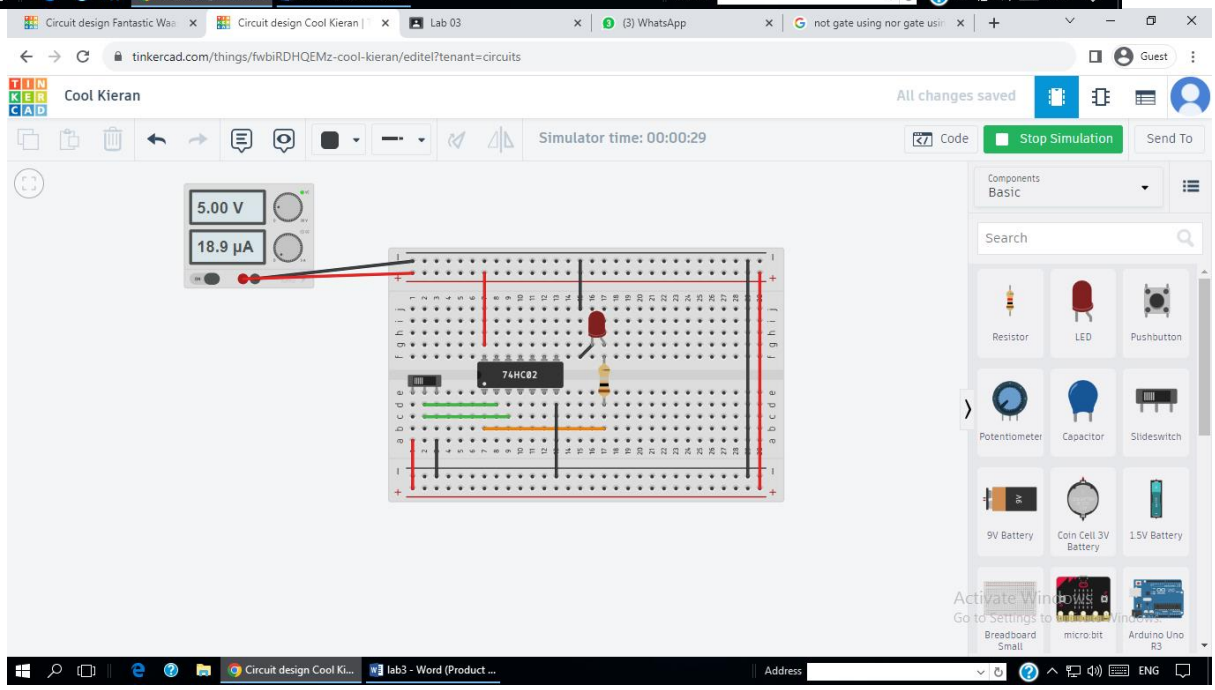


3.2. Working with NOR GATE

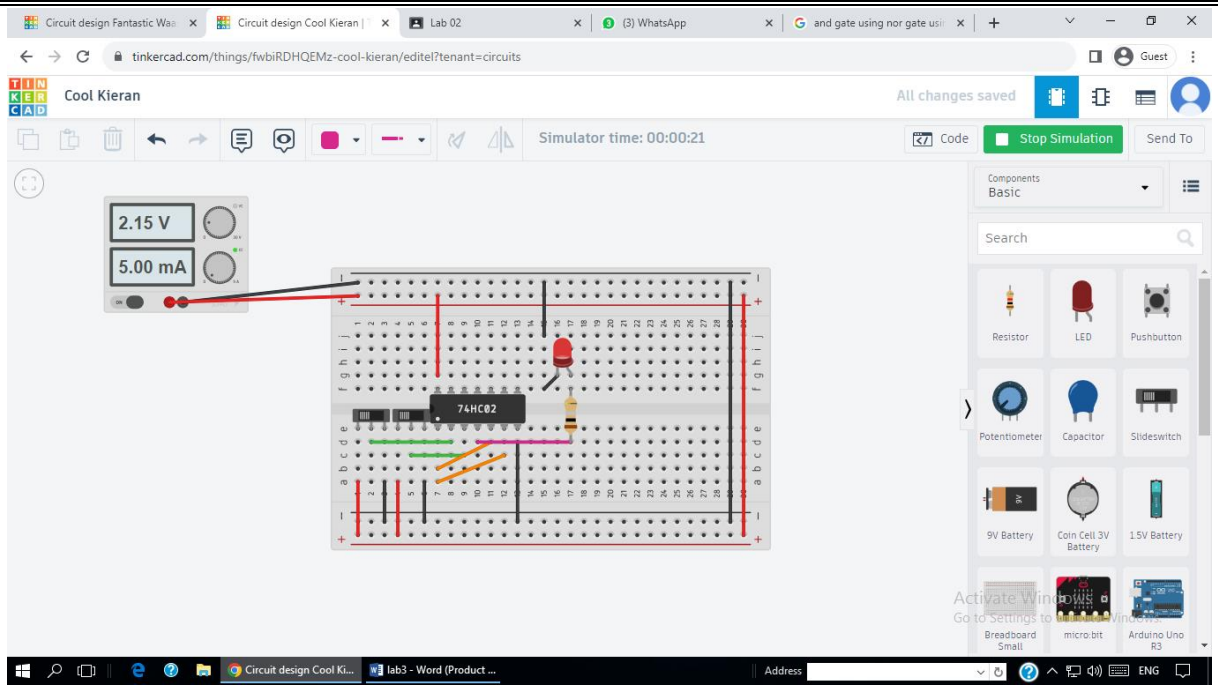


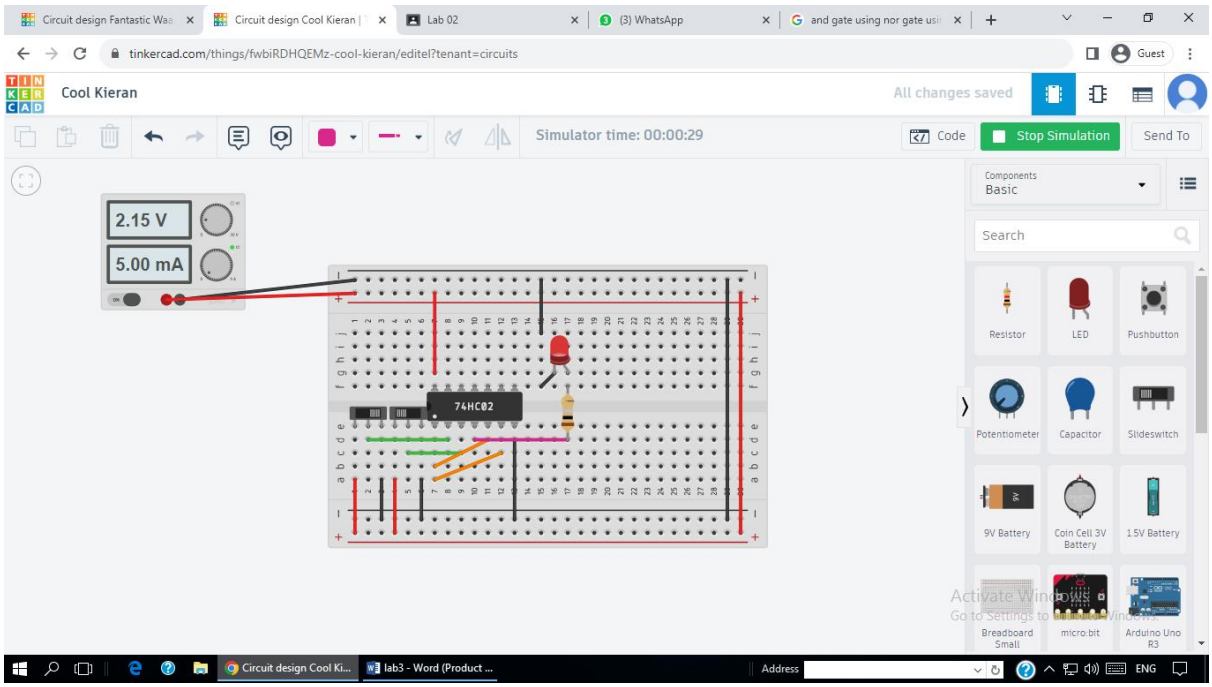
3.4 a Nand to NOT:

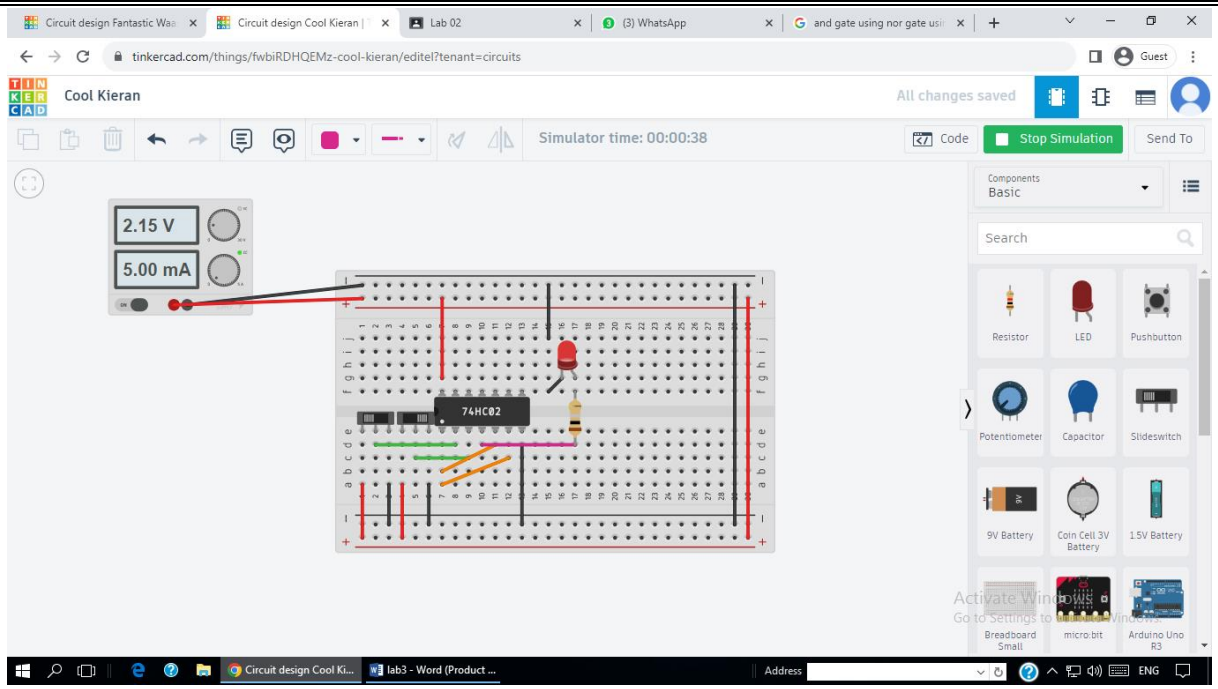


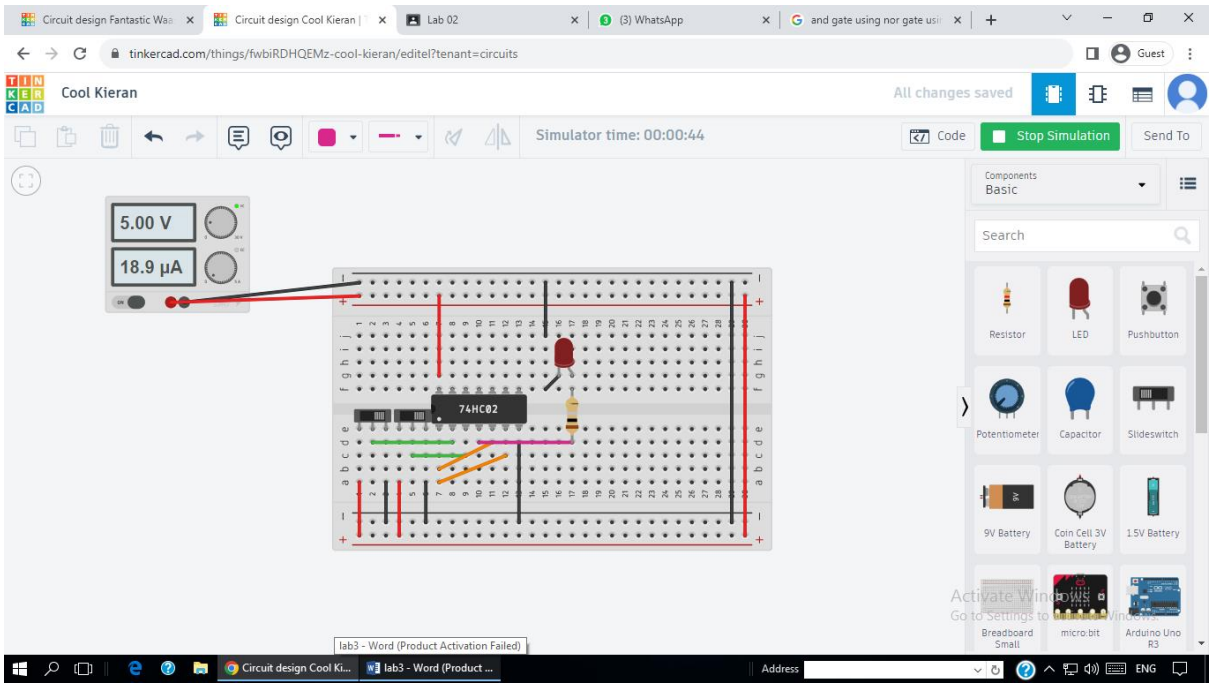


3.4 b) Nand to OR

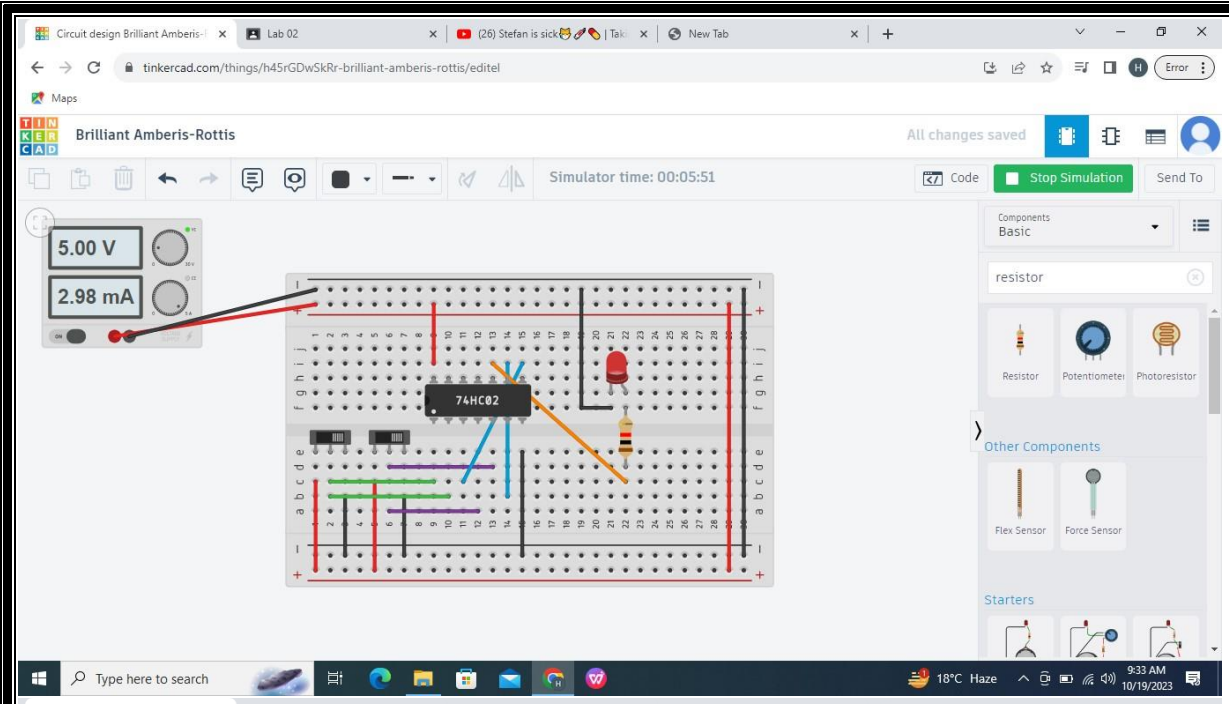

Circuit design Fantastic Wa
Circuit design Cool Kieran
Lab 02
(3) WhatsApp
and gate using nor gate usi
tinkercad.com/things/fwbiRDHQEMz-cool-kieran/editel?tenant=circuits
Guest
Cool Kieran
All changes saved
Simulator time: 00:00:21
Code
Stop Simulation
Send To
Components
Basic
Search
2.15 V
5.00 mA
74HC02
Resistor
LED
Pushbutton
Potentiometer
Capacitor
Slideswitch
9V Battery
Coin Cell 3V Battery
1.5V Battery
Breadboard Small
micro:bit
Arduino Uno R3
Activate Windows
Go to Settings to activate Windows.
Circuit design Cool Ki...
lab3 - Word (Product ...
Address
ENG

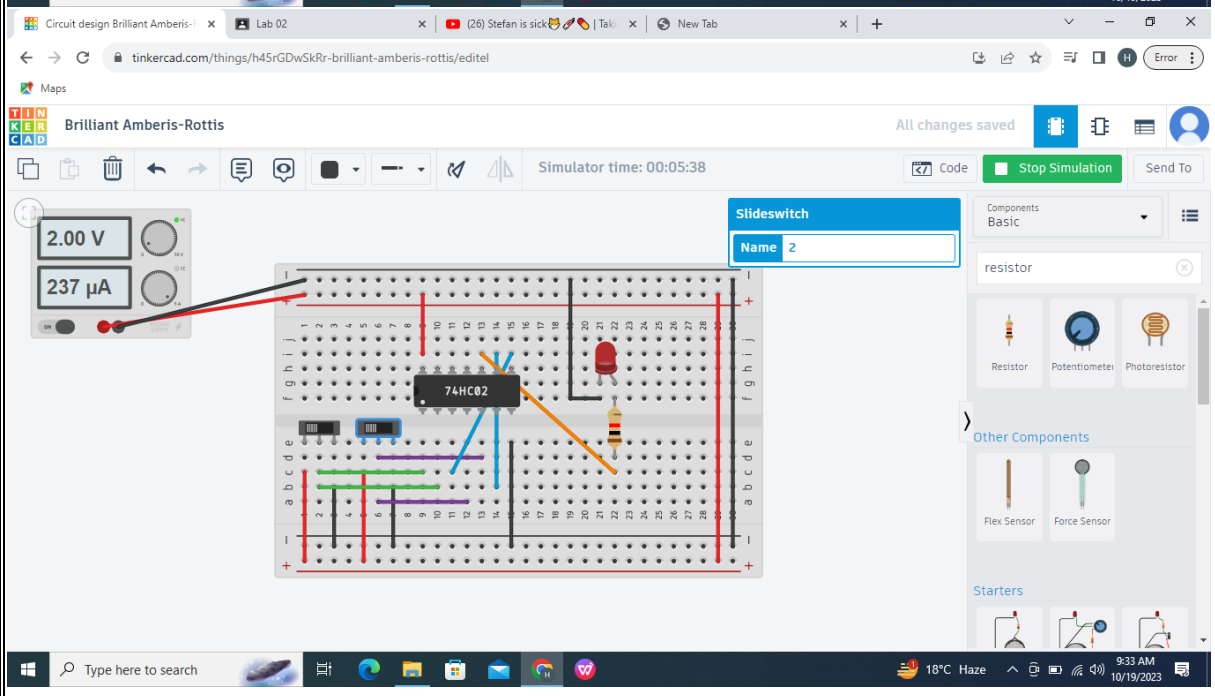

Circuit design Fantastic Wa
Circuit design Cool Kieran
Lab 02
(3) WhatsApp
and gate using nor gate usi
tinkercad.com/things/fwbiRDHQEMz-cool-kieran/editel?tenant=circuits
Guest
Cool Kieran
All changes saved
Simulator time: 00:00:29
Code
Stop Simulation
Send To
Components
Basic
Search
2.15 V
5.00 mA
74HC02
Resistor
LED
Pushbutton
Potentiometer
Capacitor
Slideswitch
9V Battery
Coin Cell 3V Battery
1.5V Battery
Breadboard Small
micro:bit
Arduino Uno R3
Activate Windows
Go to Settings to activate Windows.
Circuit design Cool Ki...
lab3 - Word (Product ...
Address
ENG

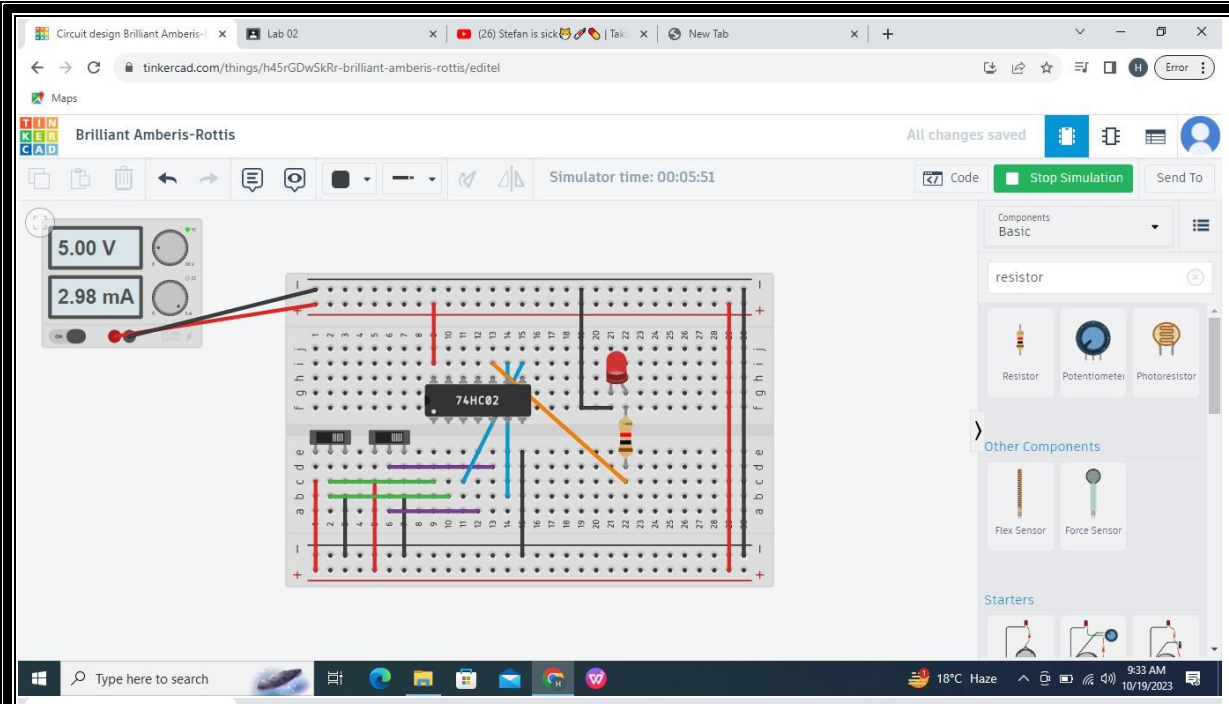


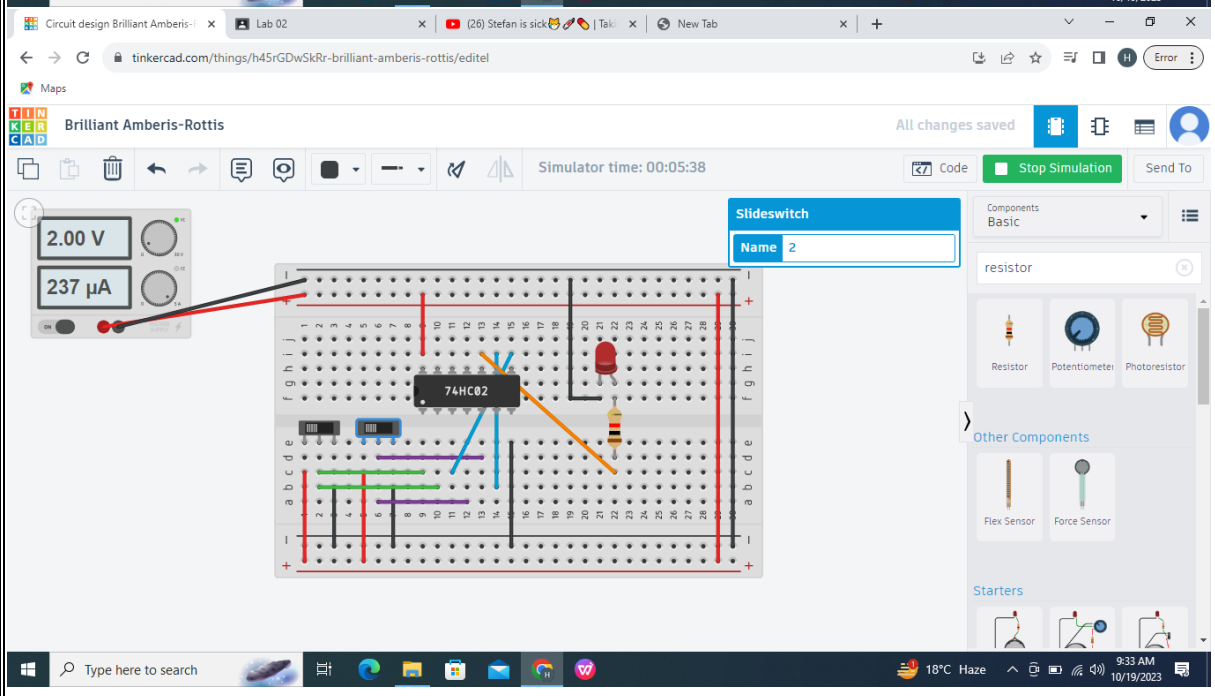


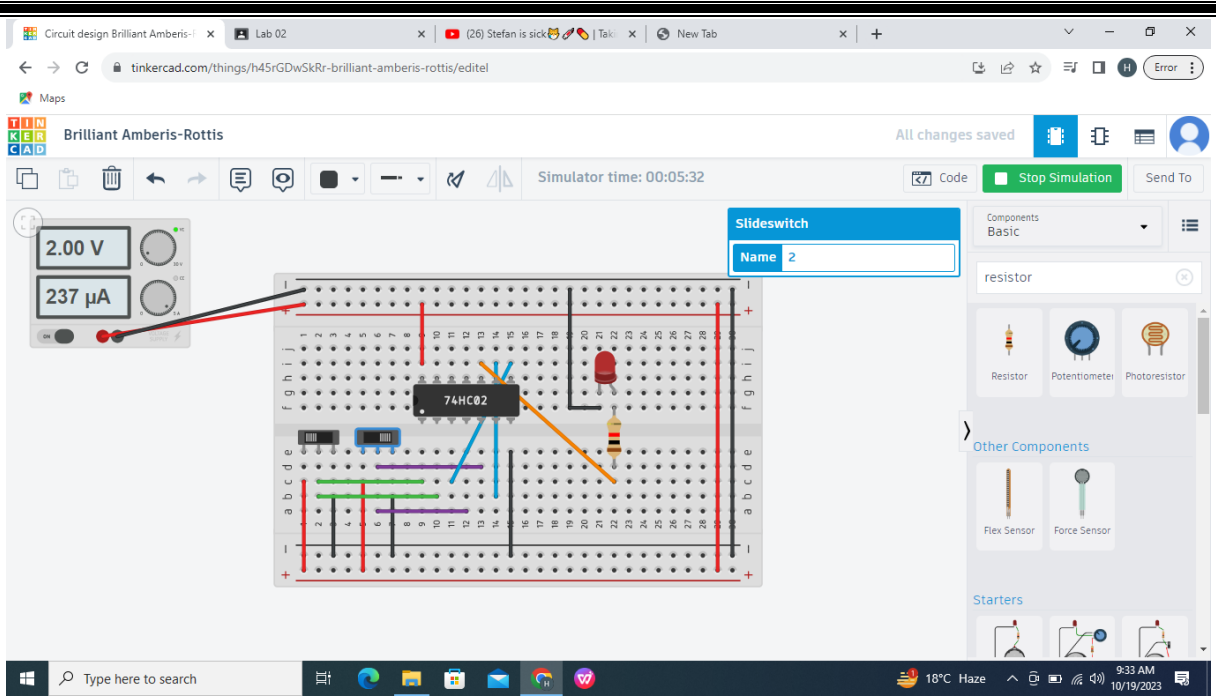
3.4 b) Nand to AND


Circuit design Brilliant Amberis-
Lab 02
(26) Stefan is sick
New Tab
tinkercad.com/things/h45rGDwSkRr-brilliant-amberis-rottis/editel
Error
Maps
Brilliant Amberis-Rottis
All changes saved
Simulator time: 00:05:51
Code
Stop Simulation
Send To
5.00 V
2.98 mA
74HC02
Components
Basic
resistor
Resistor
Potentiometer
Photoresistor
Other Components
Flex Sensor
Force Sensor
Starters
Type here to search
18°C Haze
9:33 AM
10/19/2023


Circuit design Brilliant Amberis-
Lab 02
(26) Stefan is sick
New Tab
tinkercad.com/things/h45rGDwSkRr-brilliant-amberis-rottis/editel
Error
Maps
Brilliant Amberis-Rottis
All changes saved
Simulator time: 00:05:38
Code
Stop Simulation
Send To
2.00 V
237 µA
Slideswitch
Name 2
74HC02
Components
Basic
resistor
Resistor
Potentiometer
Photoresistor
Other Components
Flex Sensor
Force Sensor
Starters
Type here to search
18°C Haze
9:33 AM
10/19/2023


Circuit design Brilliant Amberis-
Lab 02
(26) Stefan is sick
New Tab
tinkercad.com/things/h45rGDwSkRr-brilliant-amberis-rottis/editel
Error
Maps
Brilliant Amberis-Rottis
All changes saved
Simulator time: 00:05:51
Code
Stop Simulation
Send To
5.00 V
2.98 mA
74HC02
Components
Basic
resistor
Resistor
Potentiometer
Photoresistor
Other Components
Flex Sensor
Force Sensor
Starters
Type here to search
18°C Haze
9:33 AM
10/19/2023


Circuit design Brilliant Amberis-
Lab 02
(26) Stefan is sick
New Tab
tinkercad.com/things/h45rGDwSkRr-brilliant-amberis-rottis/editel
Error
Maps
Brilliant Amberis-Rottis
All changes saved
Simulator time: 00:05:38
Code
Stop Simulation
Send To
2.00 V
237 µA
Slideswitch
Name 2
74HC02
Components
Basic
resistor
Resistor
Potentiometer
Photoresistor
Other Components
Flex Sensor
Force Sensor
Starters
Type here to search
18°C Haze
9:33 AM
10/19/2023



Fatima Jinnah Women University

Sawaira saeed

2022-BSE-067

Rabia Batool

2022-BSE-064

Hafsah tahir

2022-BSE-052

Group#B

Computer architecture and logic design

LAB#03

Submitted to Sir Shoaib

NAND to XOR:

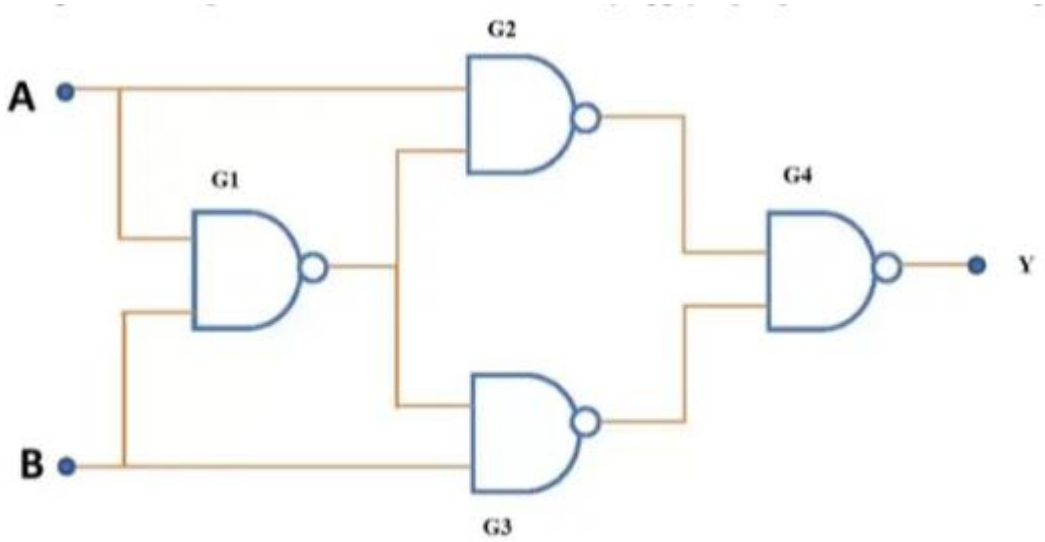
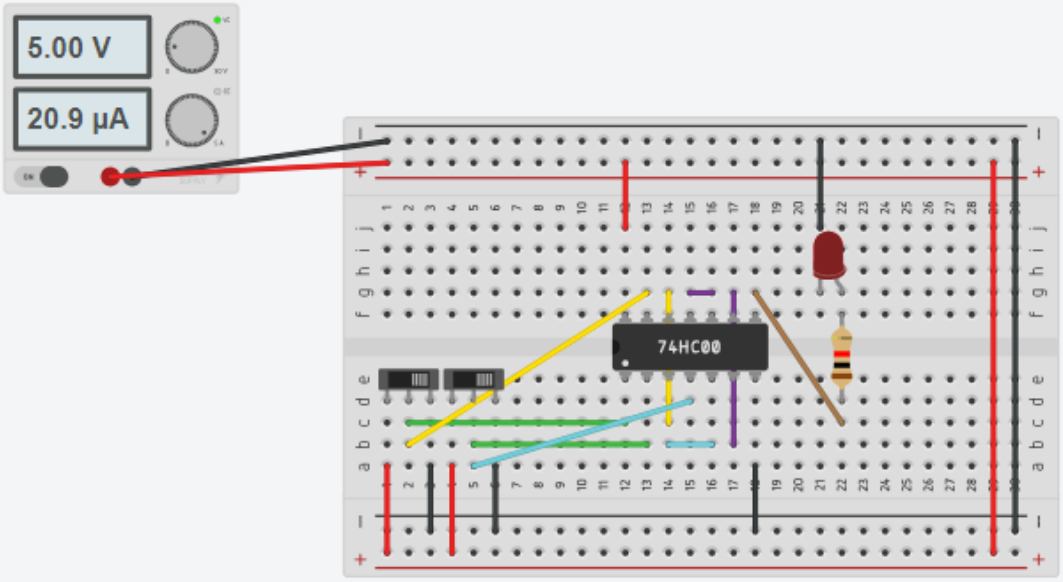
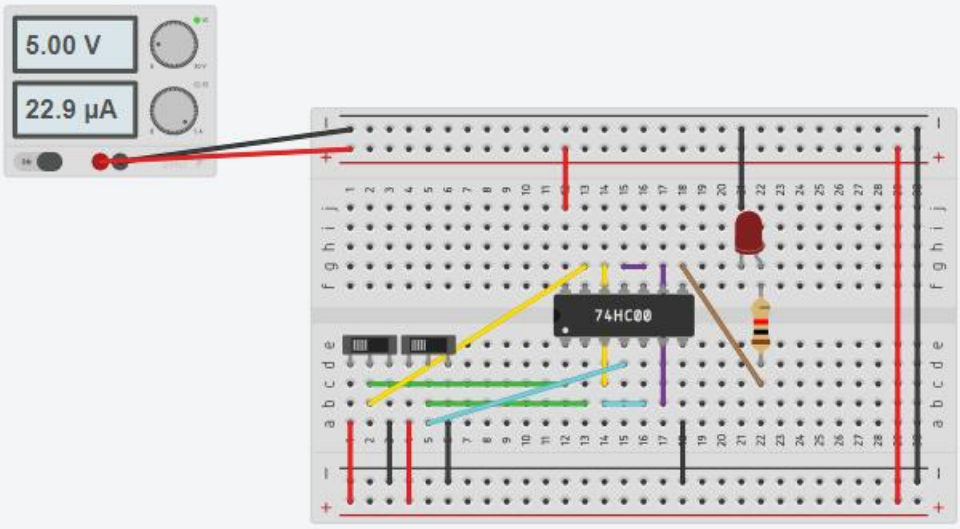
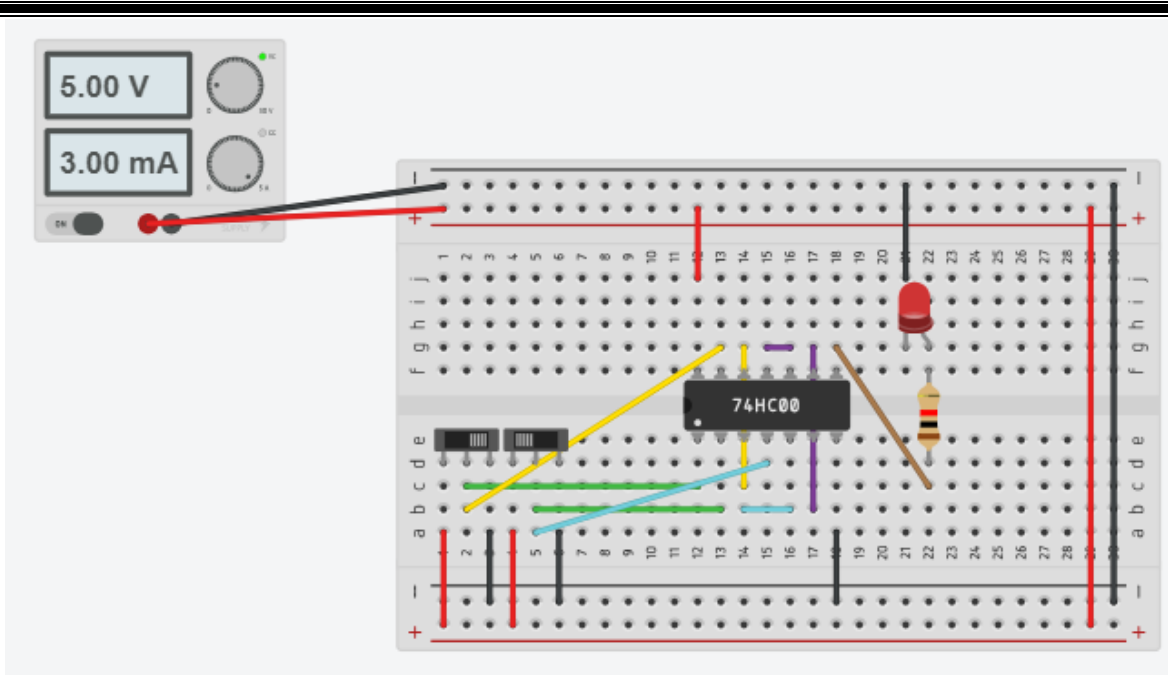
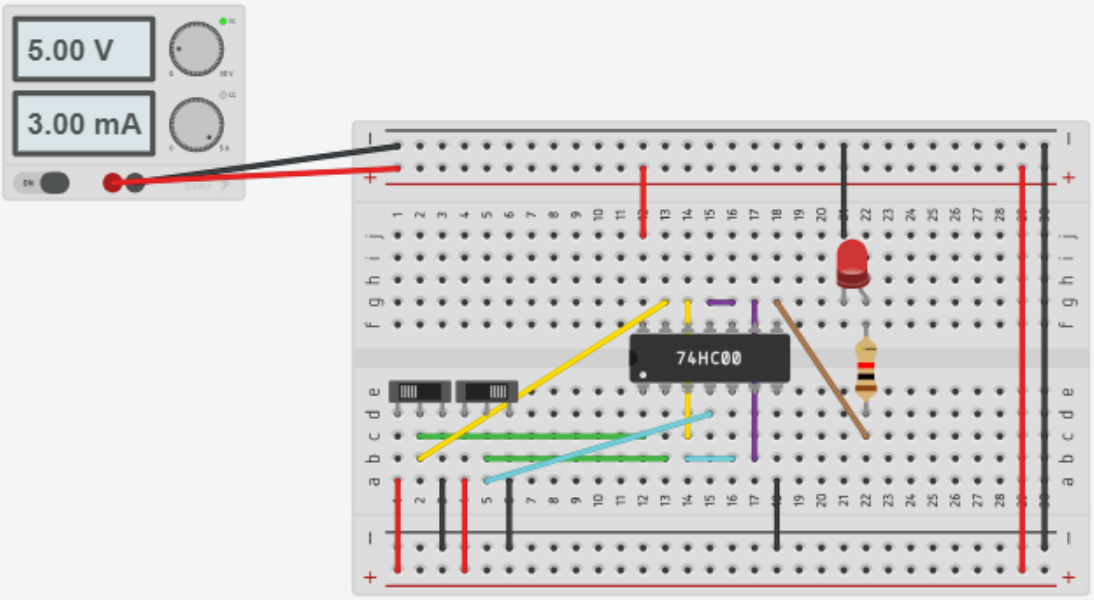


Figure 4.4: *NAND implementation of XOR gate*


5.00 V
20.9 µA
74HC00

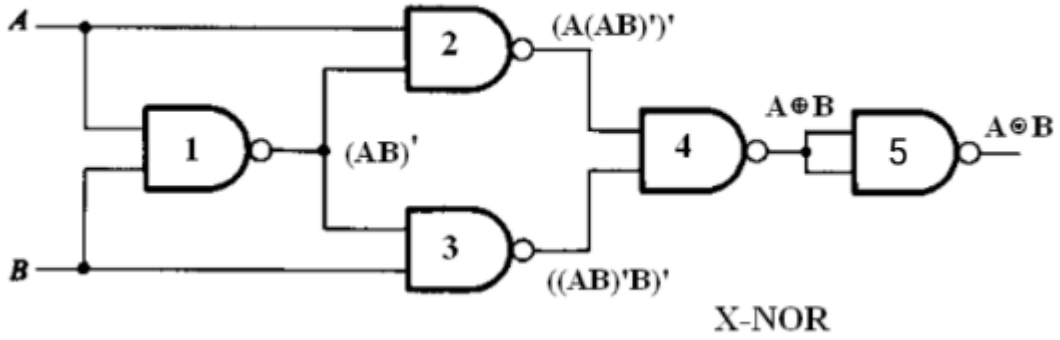

5.00 V
22.9 µA
74HC00



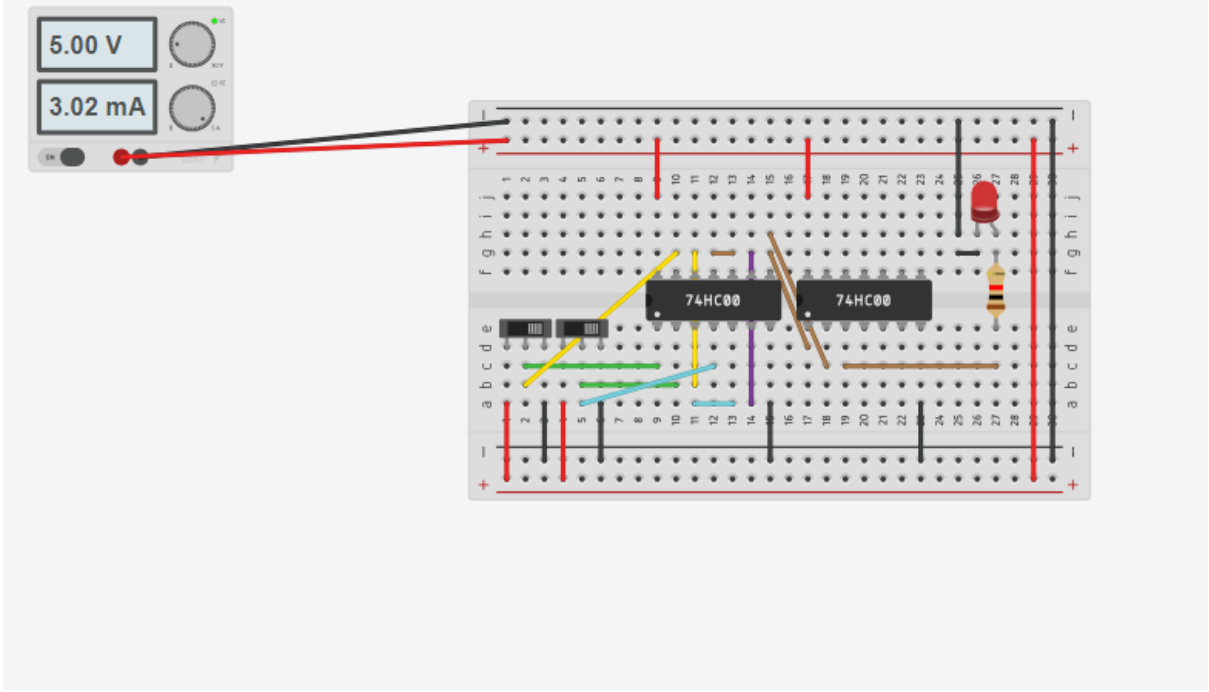


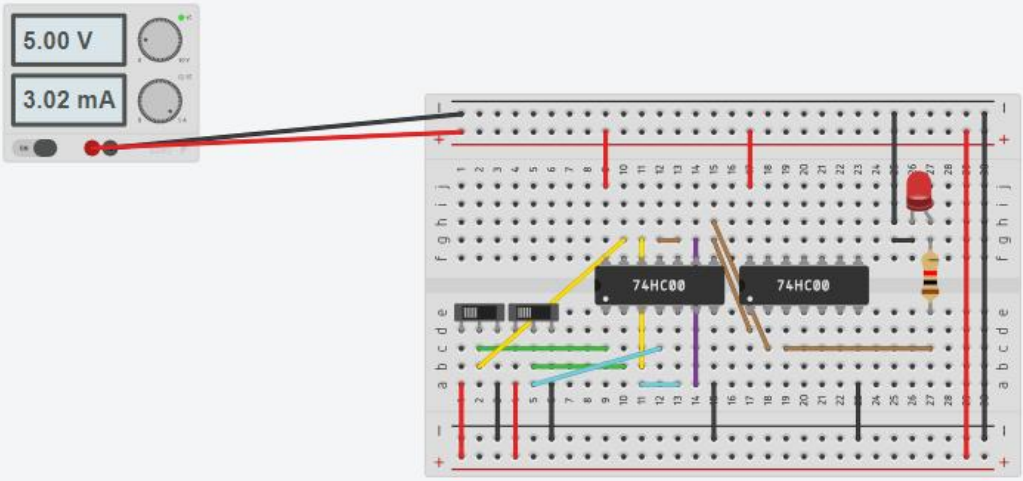
NAND to XNOR:

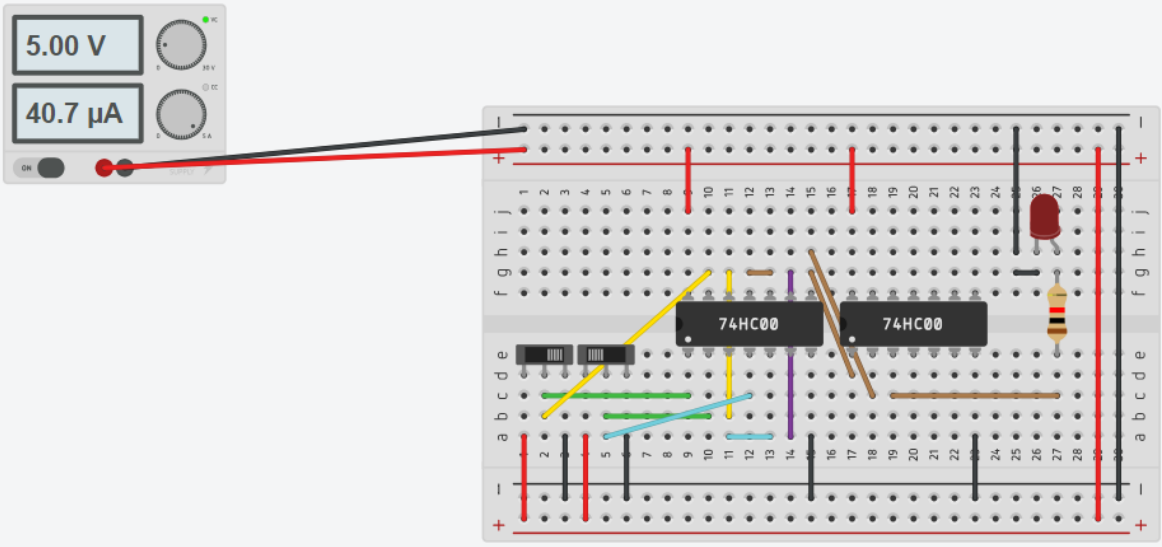
NAND Gates as EX-NOR Gate



X-NOR







Fatima Jinnah Women University

Computer architecture and logic design

Sawaira saeed

2022-BSE-067

Rabia Batool

2022-BSE-064

Hafsah tahir

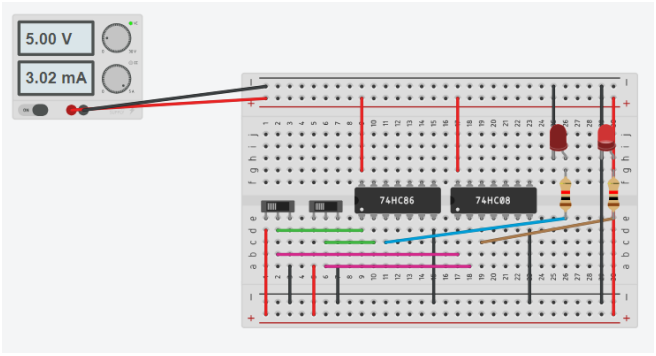
2022-BSE-052

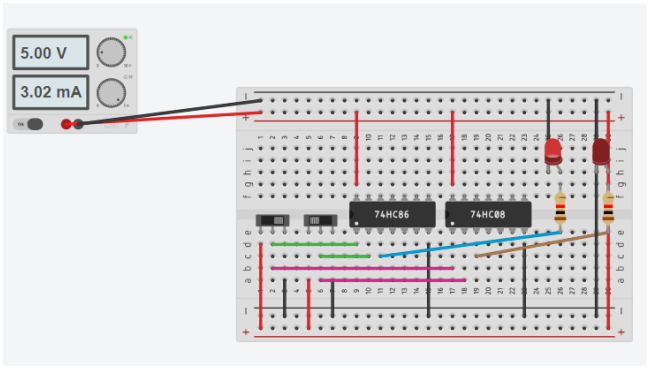
Group#B

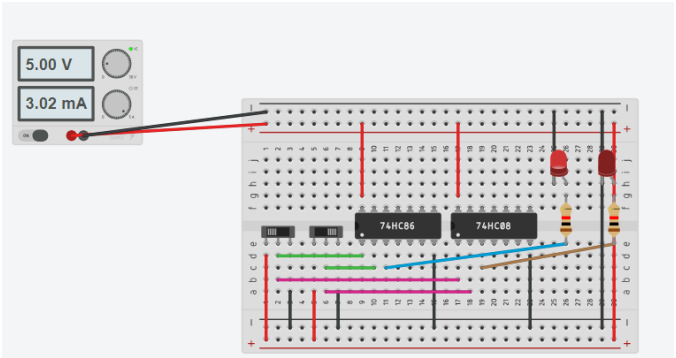
Lab 4

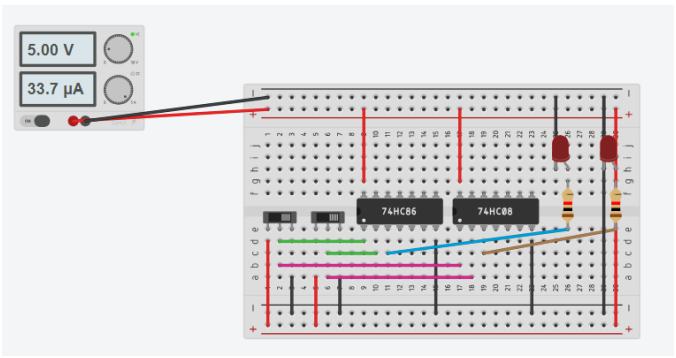
Submitted to Sir Shoaib

Half Adder

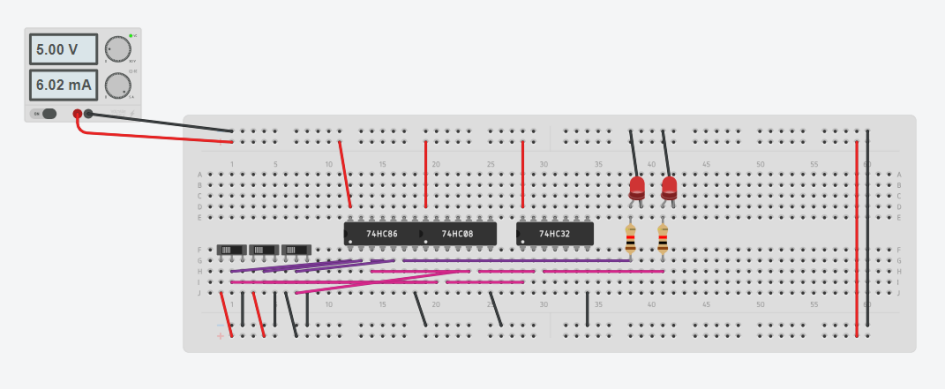


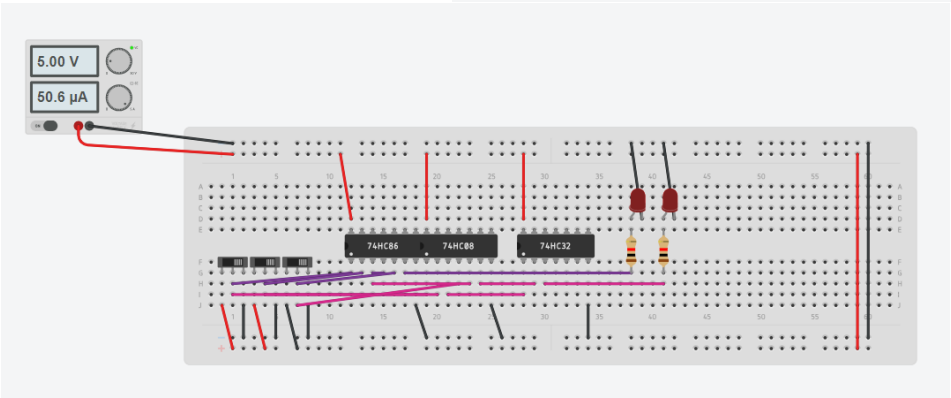


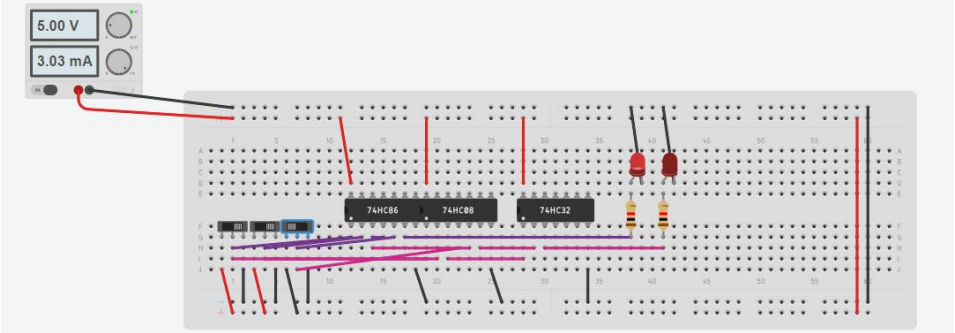


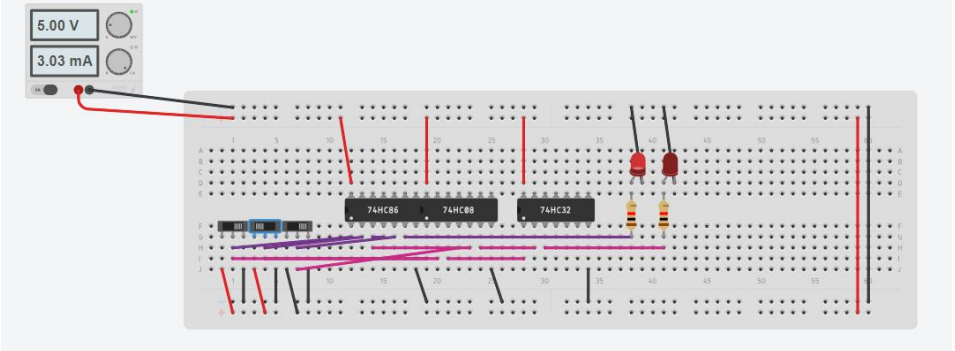


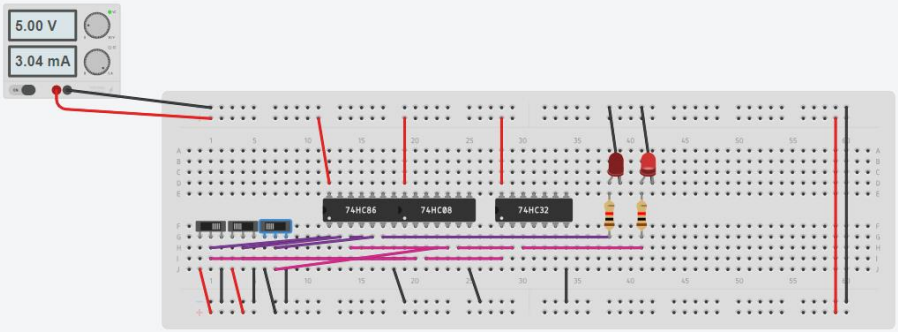
FULL ADDER

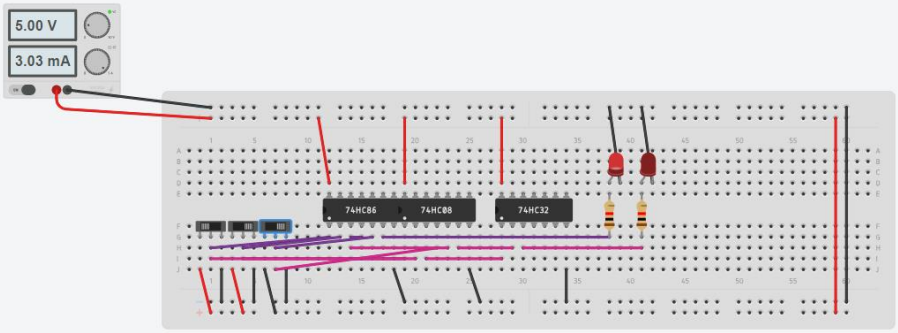

5.00 V
6.02 mA
74HC86
74HC08
74HC32

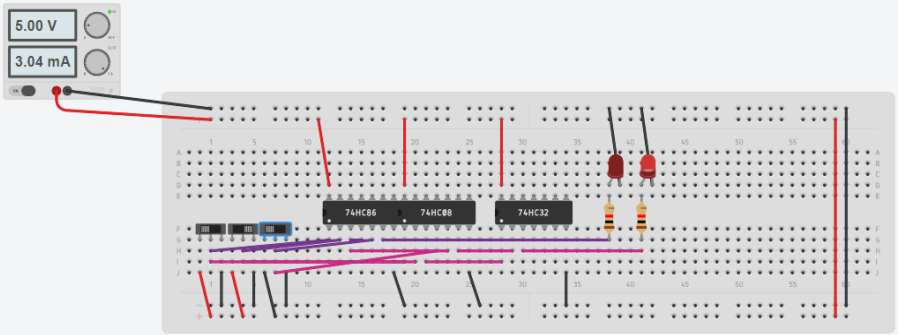

5.00 V
50.6 µA
74HC86
74HC08
74HC32

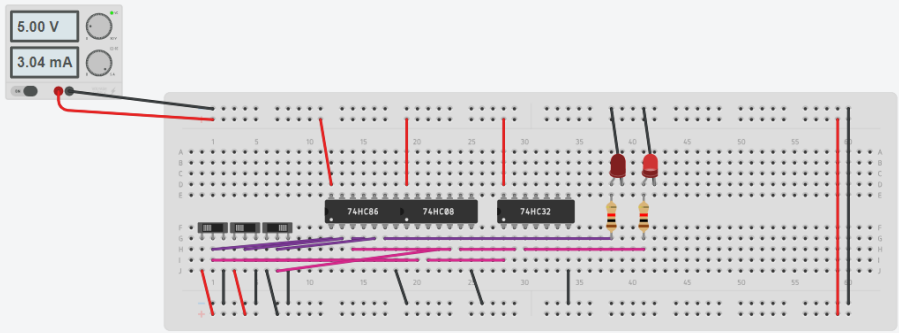

5.00 V
3.03 mA
74HC86
74HC08
74HC32


5.00 V
3.03 mA
74HC86
74HC08
74HC32


5.00 V
3.04 mA
74HC86
74HC08
74HC32


5.00 V
3.03 mA
74HC86
74HC08
74HC32


5.00 V
3.04 mA
74HC86
74HC08
74HC32


5.00 V
3.04 mA
74HC86
74HC08
74HC32

Fatima Jinnah Women University

Computer architecture and logic design

Sawaira saeed

2022-BSE-067

Rabia Batool

2022-BSE-064

Hafsah tahir

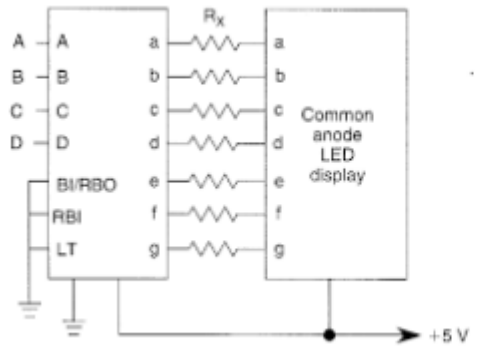
2022-BSE-052

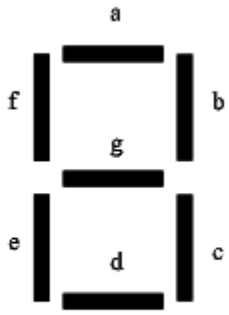
Group#B

LAB#05

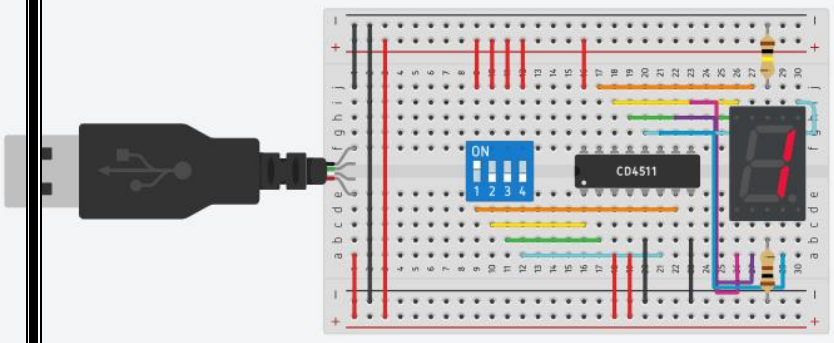
Submitted to Sir Shoaib

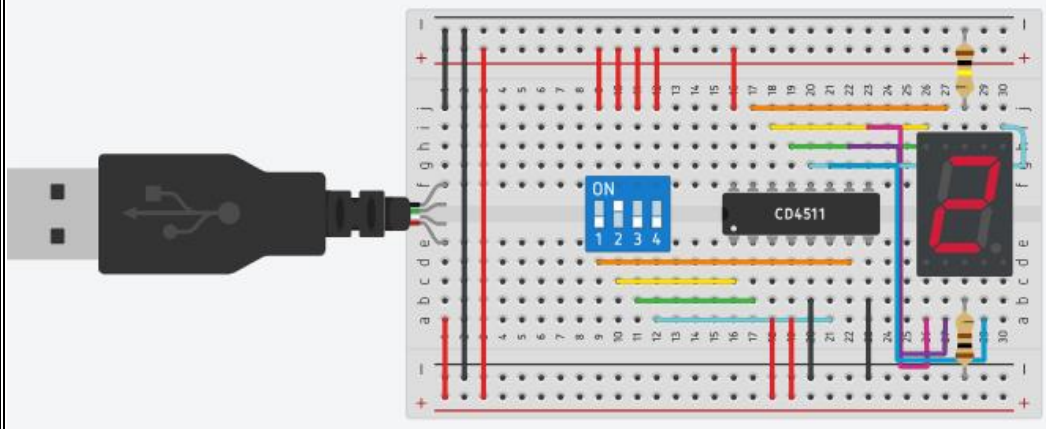
the basic concept of BCD to Seven segment decode.Understand

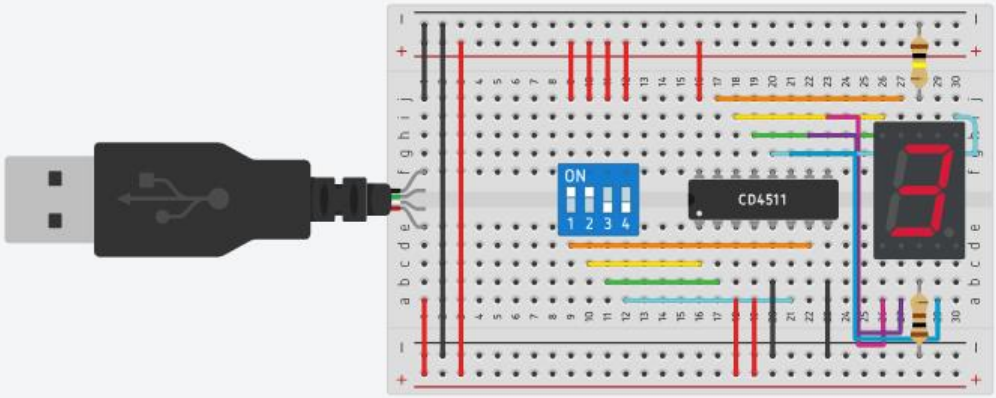


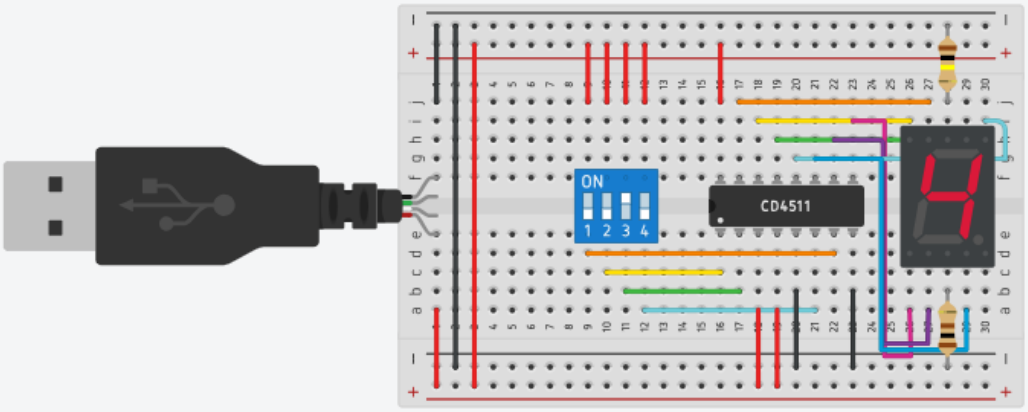


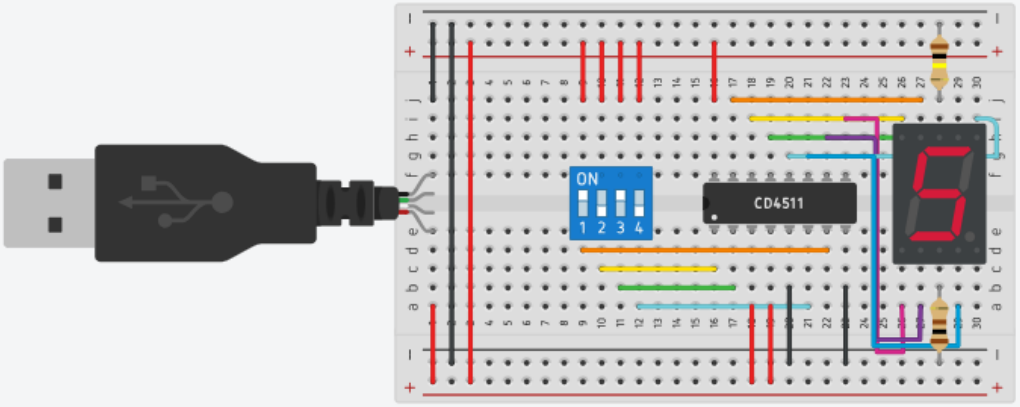
INPUT				Output
1	2	4	8	
0	0	0	0	0
1	0	0	0	1
0	1	0	0	2
1	1	0	0	3
0	0	1	0	4
1	0	1	0	5
0	1	1	0	6
1	1	1	0	7
0	0	0	1	8
1	0	0	1	9

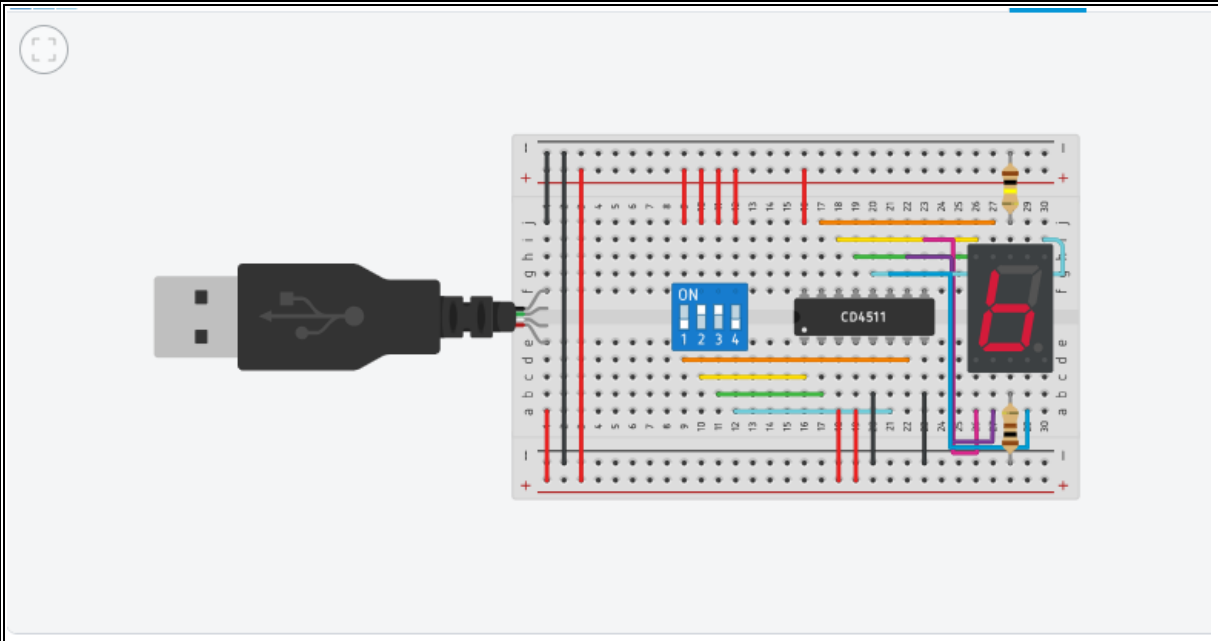

ON
1 2 3 4
CD4511

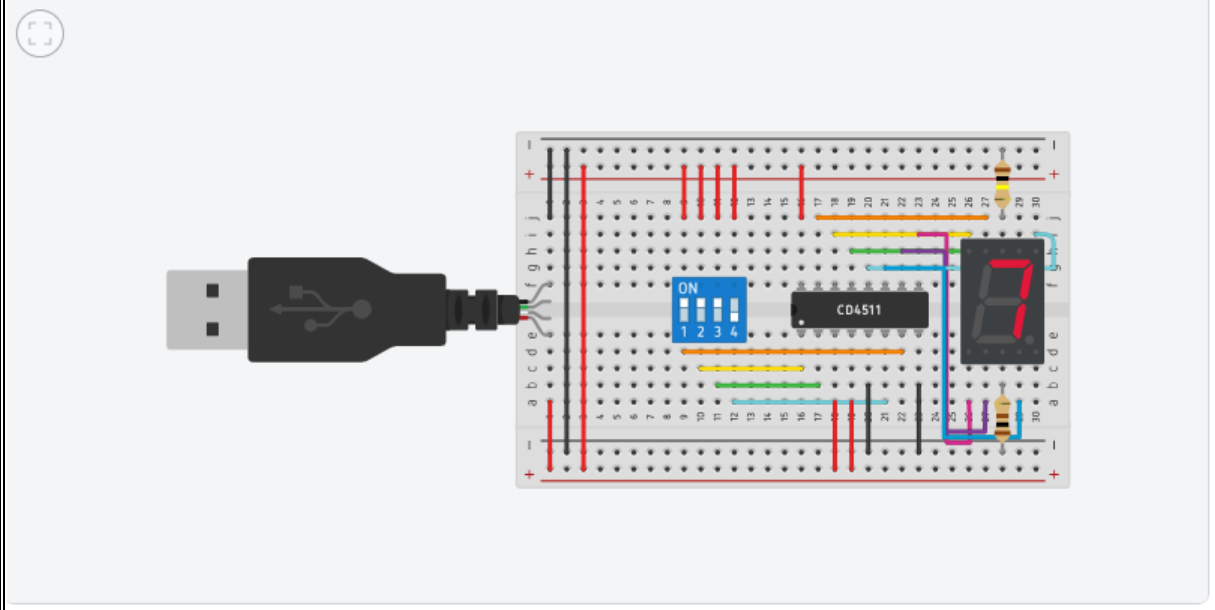

ON
1 2 3 4
CD4511

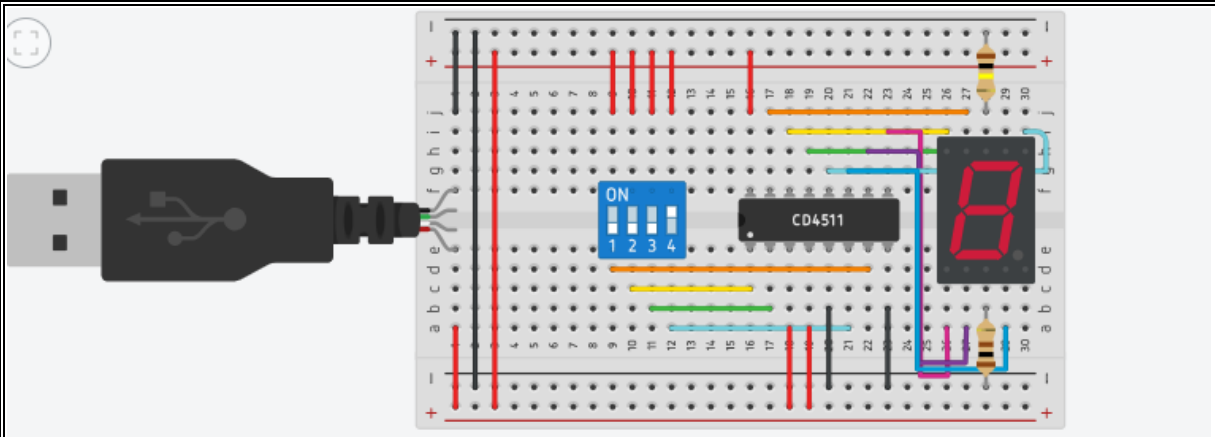

ON
1 2 3 4
CD4511

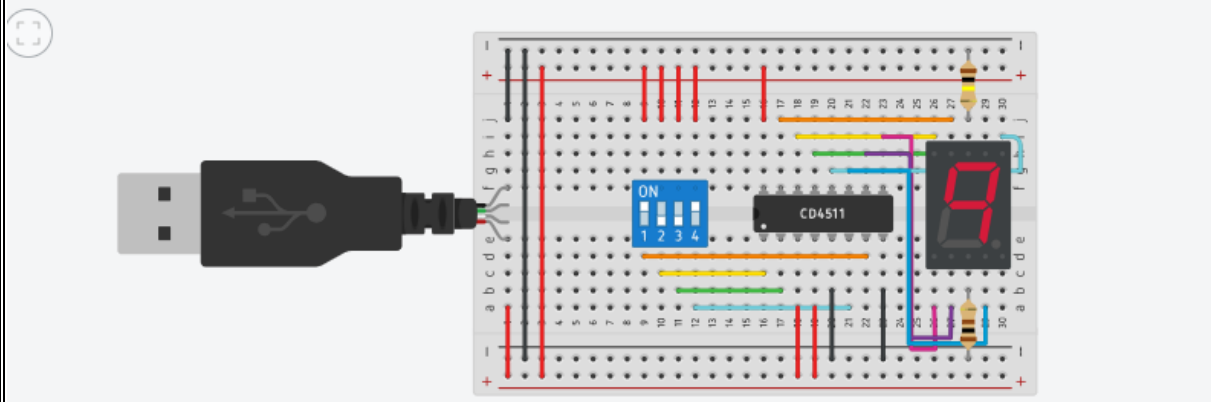

ON
1 2 3 4
CD4511


ON
1 2 3 4
CD4511


ON
1 2 3 4
CD4511


ON
1 2 3 4
CD4511





Fatima Jinnah Women University

Sawaira saeed

2022-BSE-067

Rabia Batool

2022-BSE-064

Hafsah tahir

2022-BSE-052

Group#B

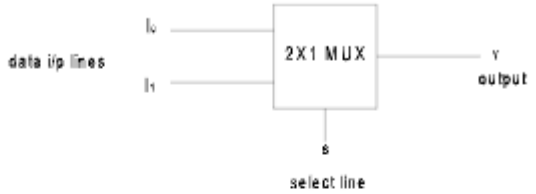
Computer architecture and logic design

LAB#03

Submitted to Sir Shoaib

Open with Google Docs

Block diagram of 2x1 MUX

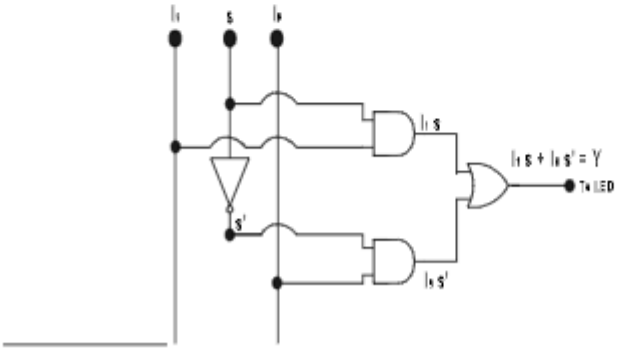


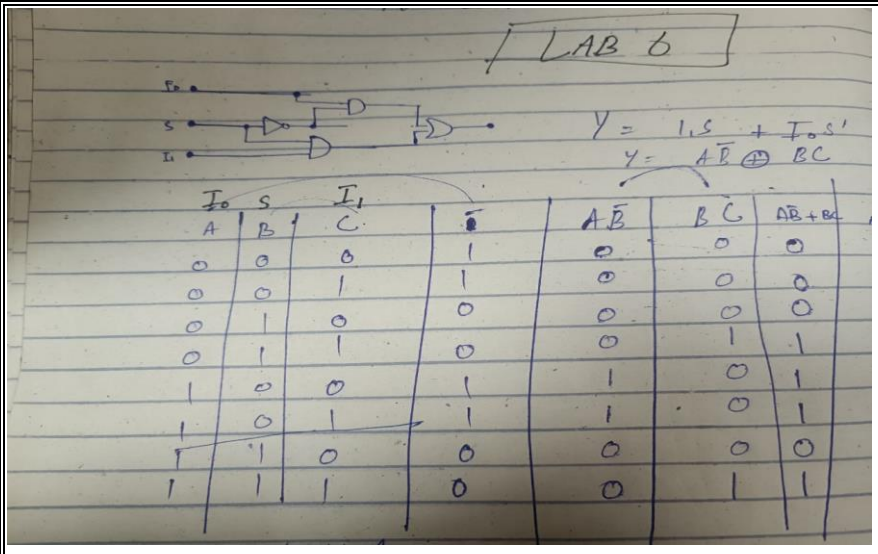
The function table of 2x1 Mux is

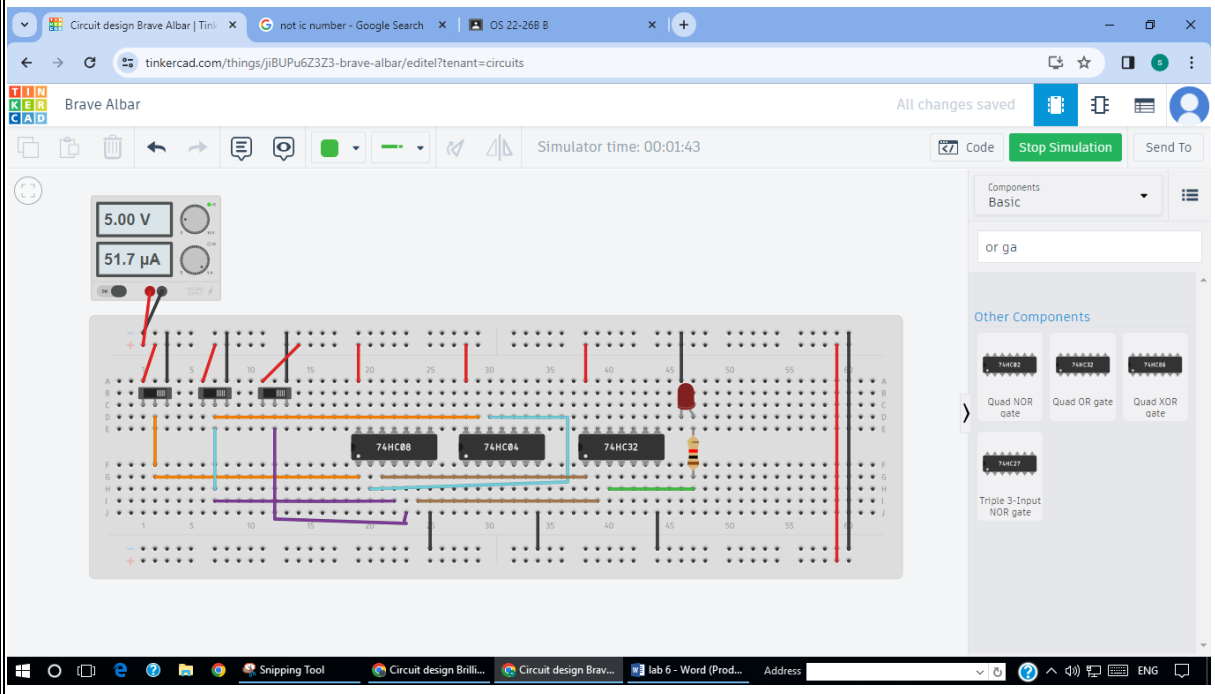
Select line	o/p
S	Y
0	I_0
1	I_1

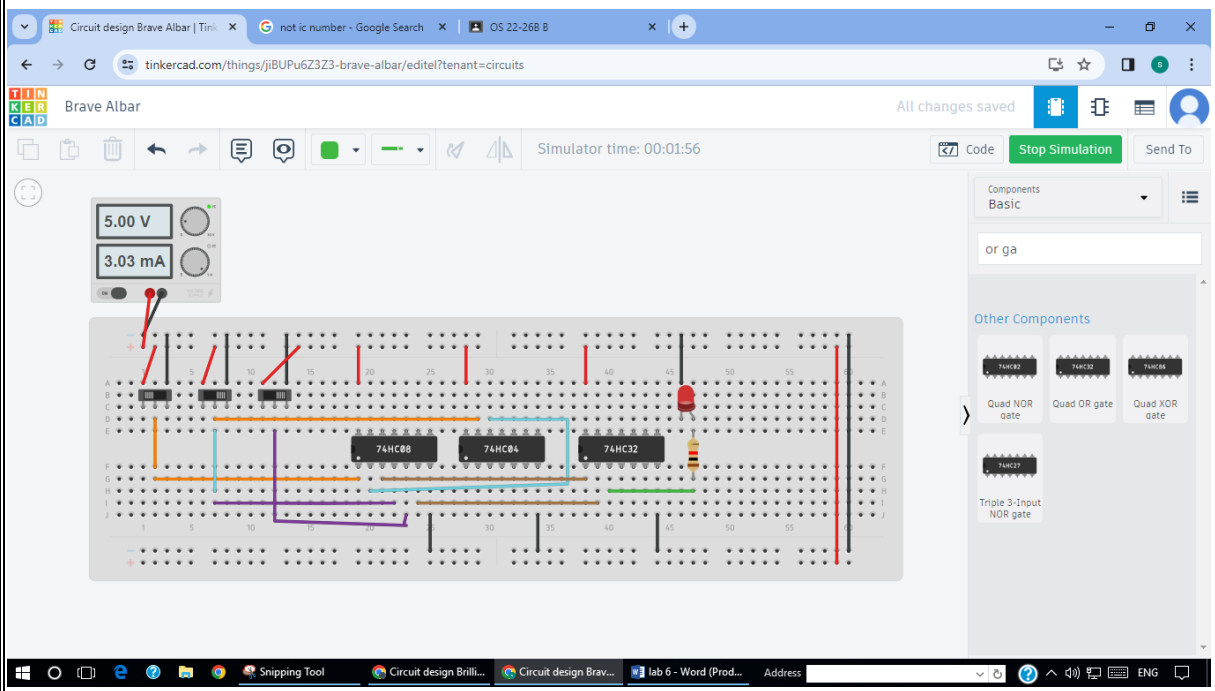
The Boolean function for 2x1 Mux is: $Y = I_1\, s + I_0\, s'$

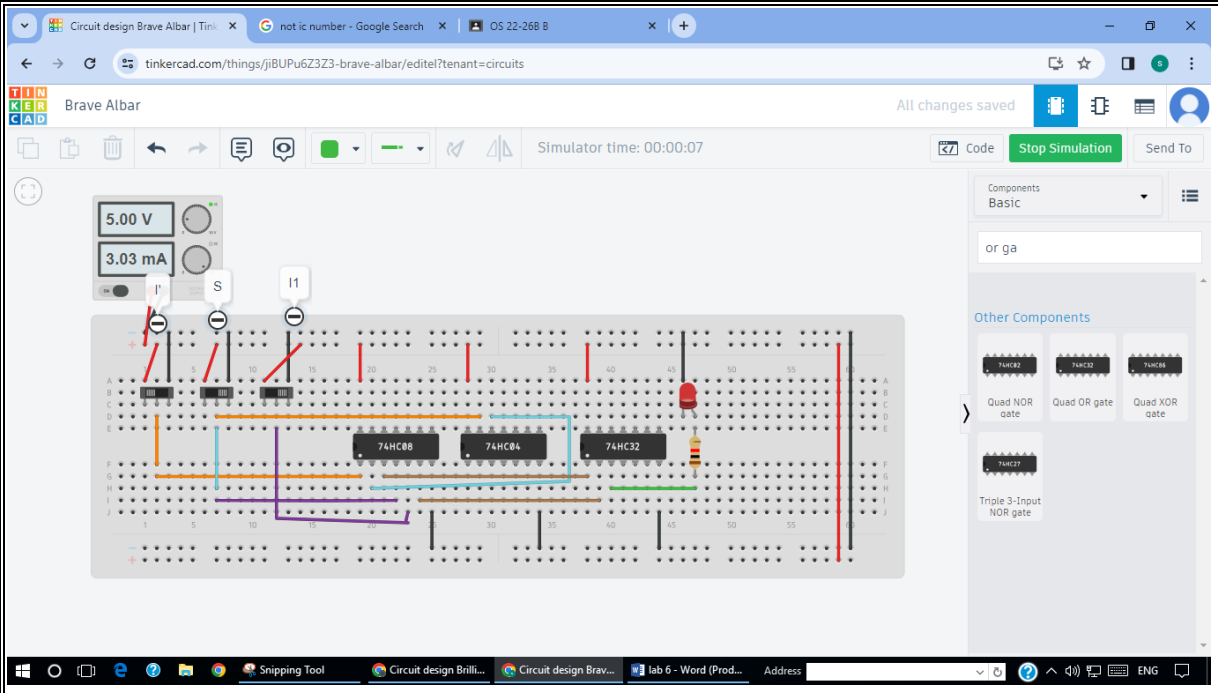
Logic Diagram of 2x1 Mux is



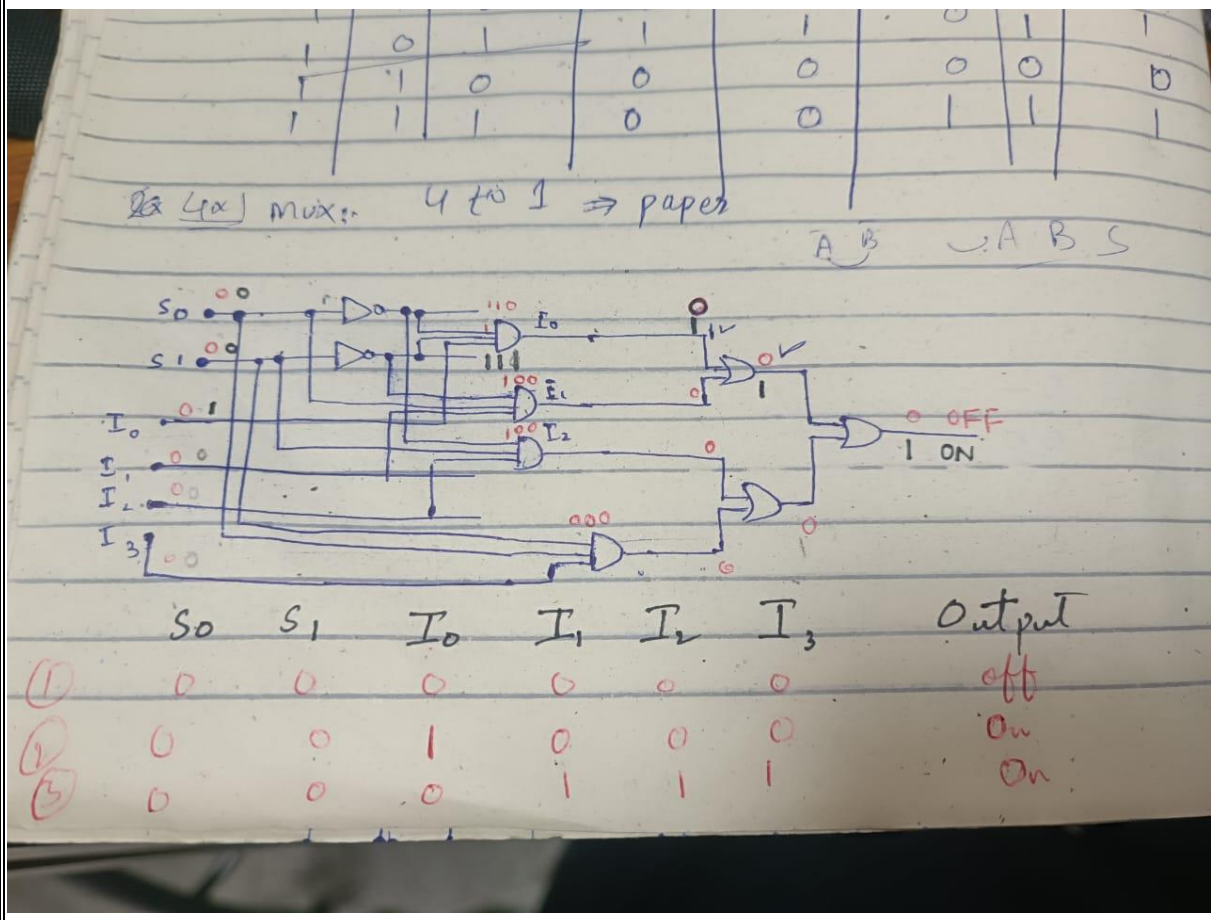

LAB 6
Y = I1.S + I0.S'
Y = AB̄ ⊕ BC
I0 S I1 F
A B C
AB̄ BC AB+BC


Circuit design Brave Albar | Tinkercad
not ic number - Google Search
OS 22-26B B
tinkercad.com/things/jiBUPu6Z3Z3-brave-albar/editel?tenant=circuits
Brave Albar
All changes saved
Simulator time: 00:01:43
Code
Stop Simulation
Send To
5.00 V
51.7 µA
74HC08
74HC04
74HC32
Components
Basic
or ga
Other Components
Quad NOR gate
Quad OR gate
Quad XOR gate
Triple 3-Input NOR gate
Snipping Tool
Circuit design Brilli...
Circuit design Brav...
lab 6 - Word (Prod...
Address
ENG

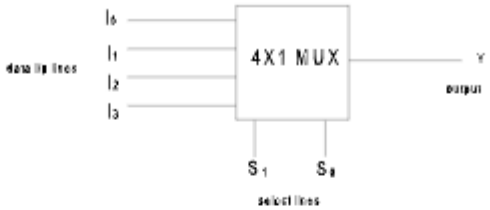

Circuit design Brave Albar | Tinkercad
not ic number - Google Search
OS 22-26B B
tinkercad.com/things/jiBUPu6Z3Z3-brave-albar/editel?tenant=circuits
Brave Albar
All changes saved
Simulator time: 00:01:56
Code
Stop Simulation
Send To
5.00 V
3.03 mA
74HC08
74HC04
74HC32
Components
Basic
or ga
Other Components
Quad NOR gate
Quad OR gate
Quad XOR gate
Triple 3-Input NOR gate
Snipping Tool
Circuit design Brilli...
Circuit design Brav...
lab 6 - Word (Prod...
Address
ENG



Multiplexer 4 to2:



Block diagram of 4x1 MUX

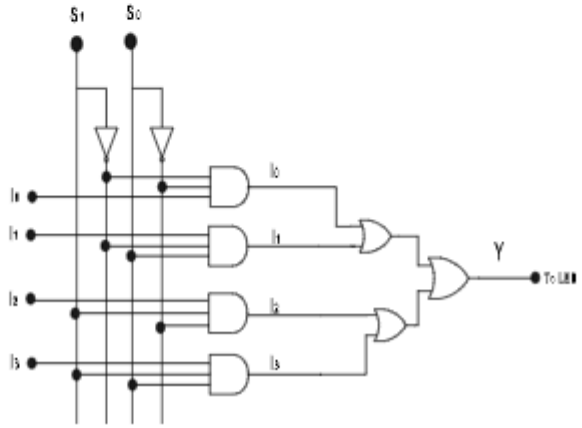


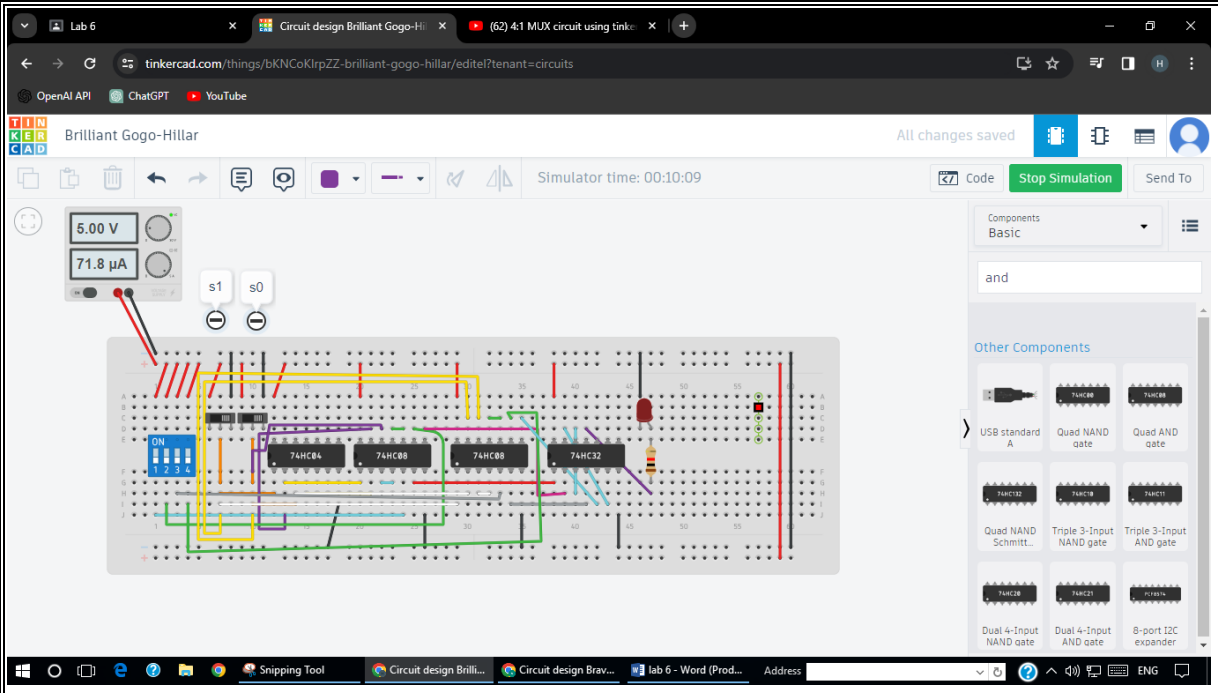
I_0, I_1, I_2 and I_3 are inputs of Mux
S_1 and S_0 are select lines
Y is output

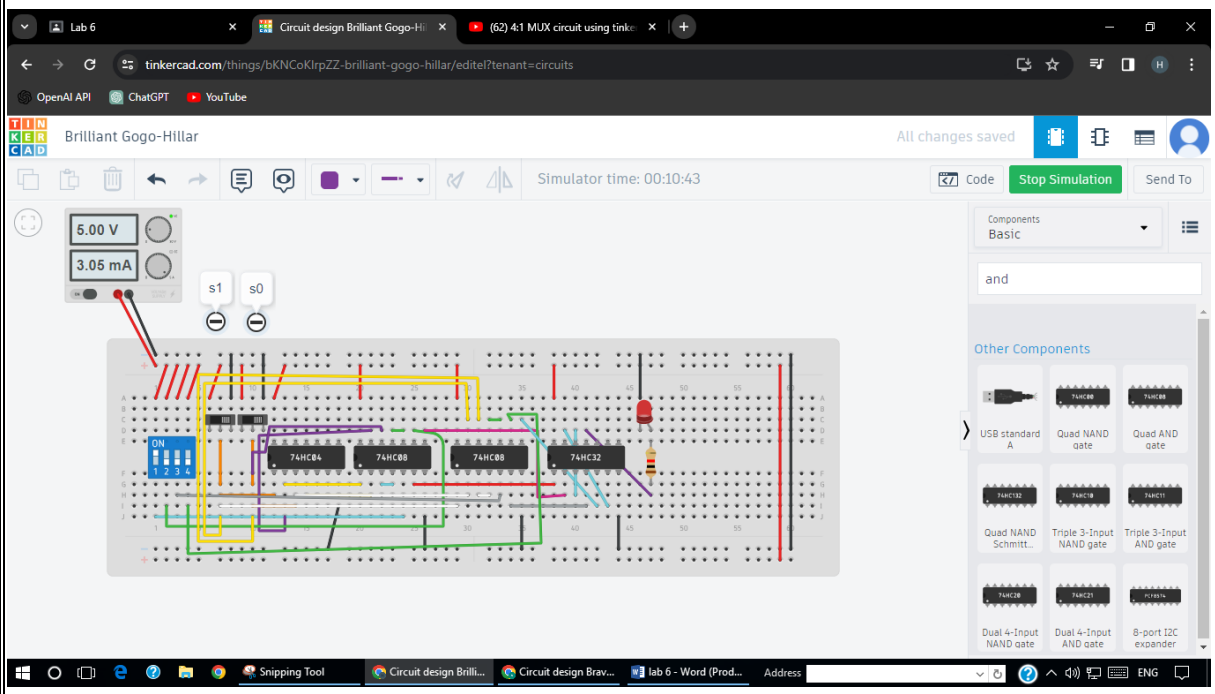
The Boolean function for 4x1 Mux is

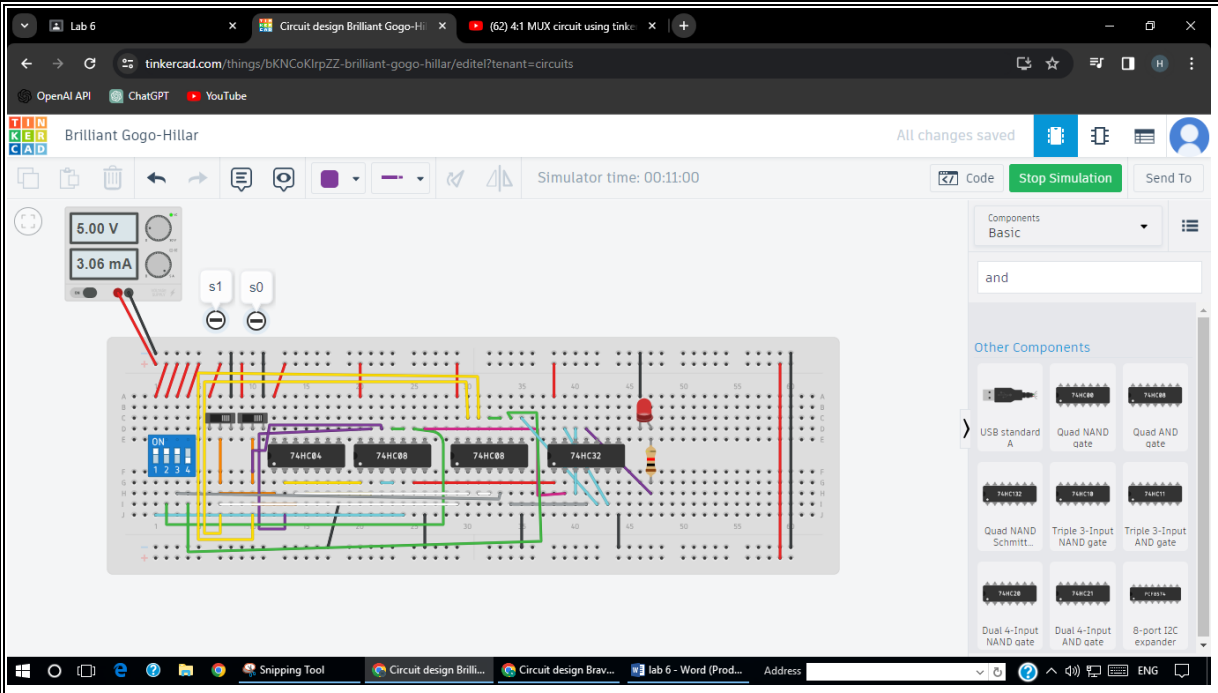
$$Y = I_0 S_1' S_0' + S_1' S_0 I_1 + S_1 S_0' I_2 + S_1 S_0 I_3$$

Logic Diagram of 4x1 Mux is

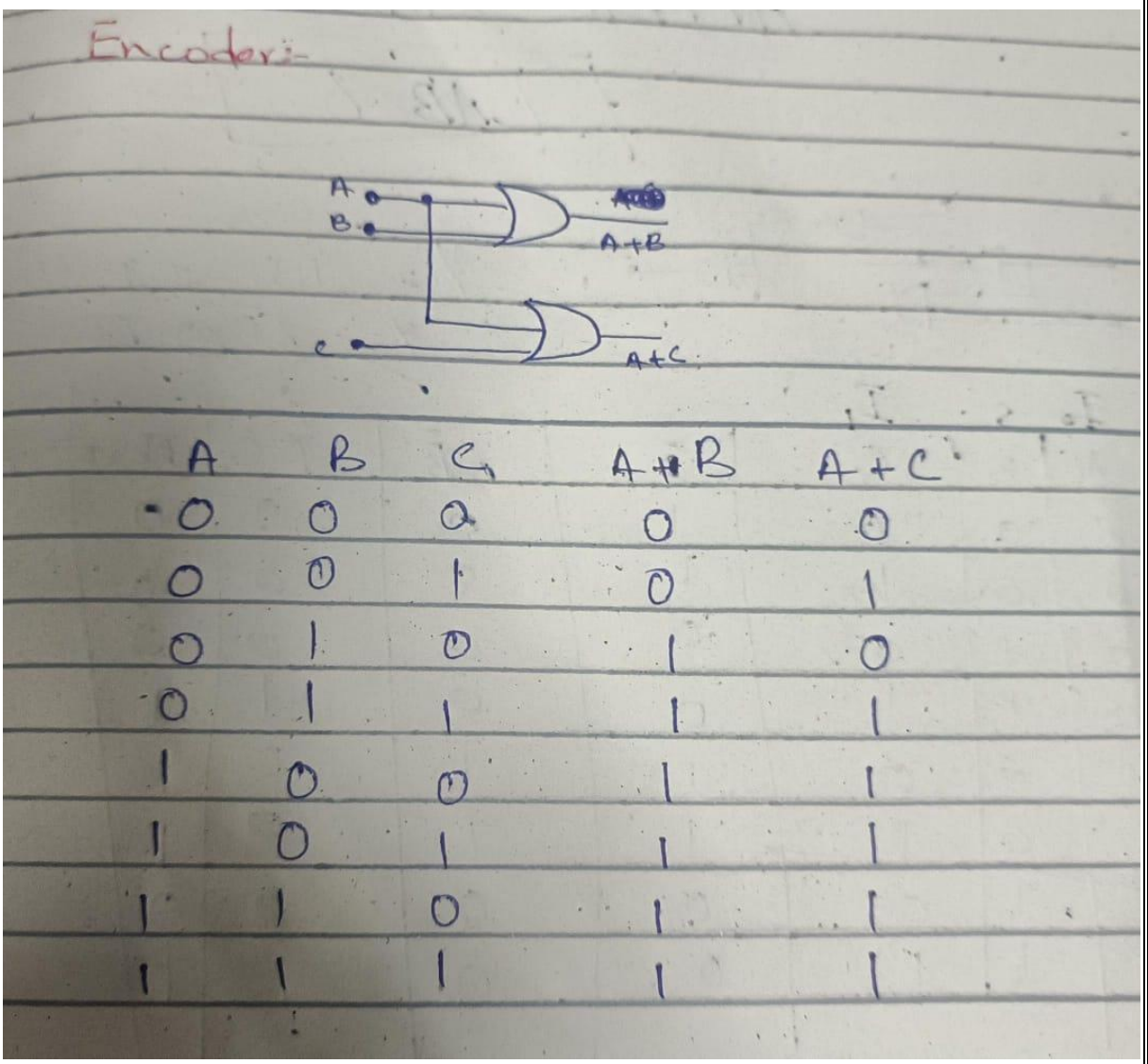


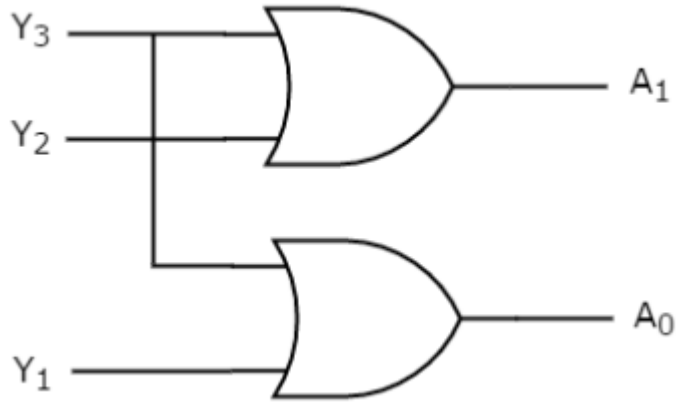

Lab 6
Circuit design Brilliant Gogo-Hi
(62) 4:1 MUX circuit using tinke
tinkercad.com/things/bKNCoKlrpZZ-brilliant-gogo-hillar/editel?tenant=circuits
OpenAI API
ChatGPT
YouTube
Brilliant Gogo-Hillar
All changes saved
Simulator time: 00:10:09
Code
Stop Simulation
Send To
5.00 V
71.8 µA
s1
s0
74HC04
74HC08
74HC08
74HC32
Components
Basic
and
Other Components
USB standard A
Quad NAND gate
Quad AND gate
Quad NAND Schmitt...
Triple 3-Input NAND gate
Triple 3-Input AND gate
Dual 4-Input NAND gate
Dual 4-Input AND gate
8-port I2C expander
Snipping Tool
Circuit design Brilli...
Circuit design Brav...
lab 6 - Word (Prod...
Address
ENG

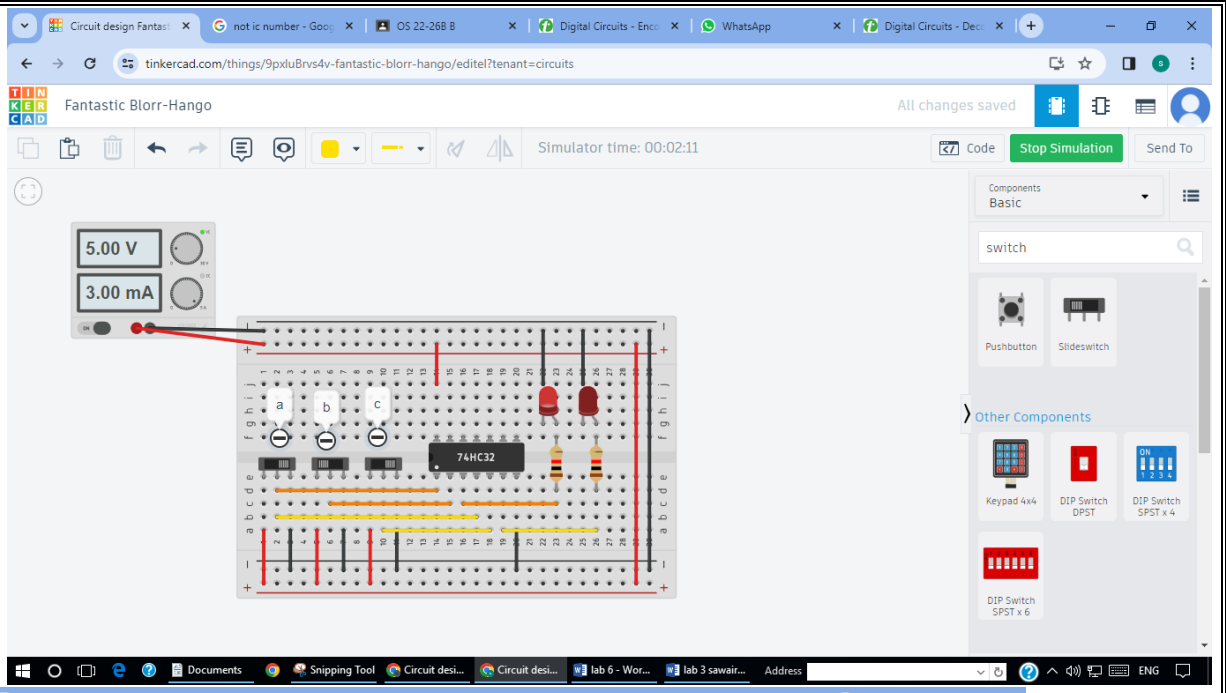

Lab 6
Circuit design Brilliant Gogo-Hi
(62) 4:1 MUX circuit using tinke
tinkercad.com/things/bKNCoKlrpZZ-brilliant-gogo-hillar/editel?tenant=circuits
OpenAI API
ChatGPT
YouTube
Brilliant Gogo-Hillar
All changes saved
Simulator time: 00:10:43
Code
Stop Simulation
Send To
5.00 V
3.05 mA
s1
s0
74HC04
74HC08
74HC08
74HC32
Components
Basic
and
Other Components
USB standard A
Quad NAND gate
Quad AND gate
Quad NAND Schmitt...
Triple 3-Input NAND gate
Triple 3-Input AND gate
Dual 4-Input NAND gate
Dual 4-Input AND gate
8-port I2C expander
Snipping Tool
Circuit design Brilli...
Circuit design Brav...
lab 6 - Word (Prod...
Address
ENG

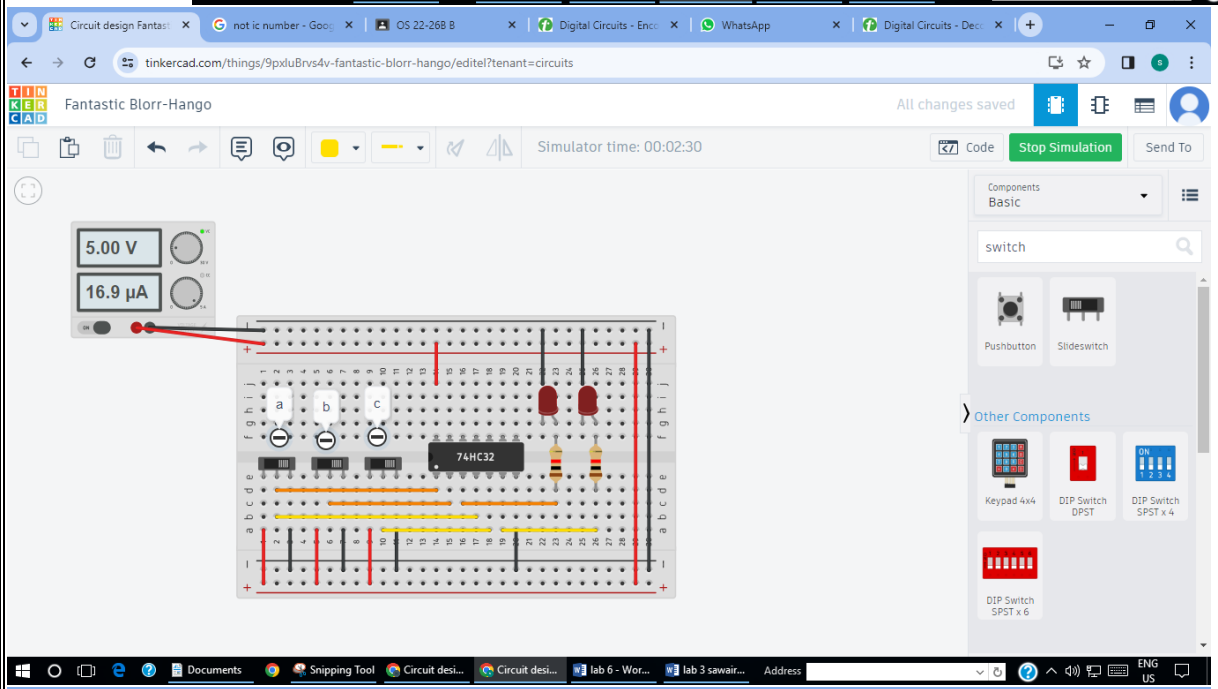

Lab 6
Circuit design Brilliant Gogo-Hi
(62) 4:1 MUX circuit using tinke
tinkercad.com/things/bKNCoKlrpZZ-brilliant-gogo-hillar/editel?tenant=circuits
OpenAI API
ChatGPT
YouTube
Brilliant Gogo-Hillar
All changes saved
Simulator time: 00:11:00
Code
Stop Simulation
Send To
5.00 V
3.06 mA
s1
s0
74HC04
74HC08
74HC08
74HC32
Components
Basic
and
Other Components
USB standard A
Quad NAND gate
Quad AND gate
Quad NAND Schmitt...
Triple 3-Input NAND gate
Triple 3-Input AND gate
Dual 4-Input NAND gate
Dual 4-Input AND gate
8-port I2C expander
Snipping Tool
Circuit design Brilli...
Circuit design Brav...
lab 6 - Word (Prod...
Address
ENG

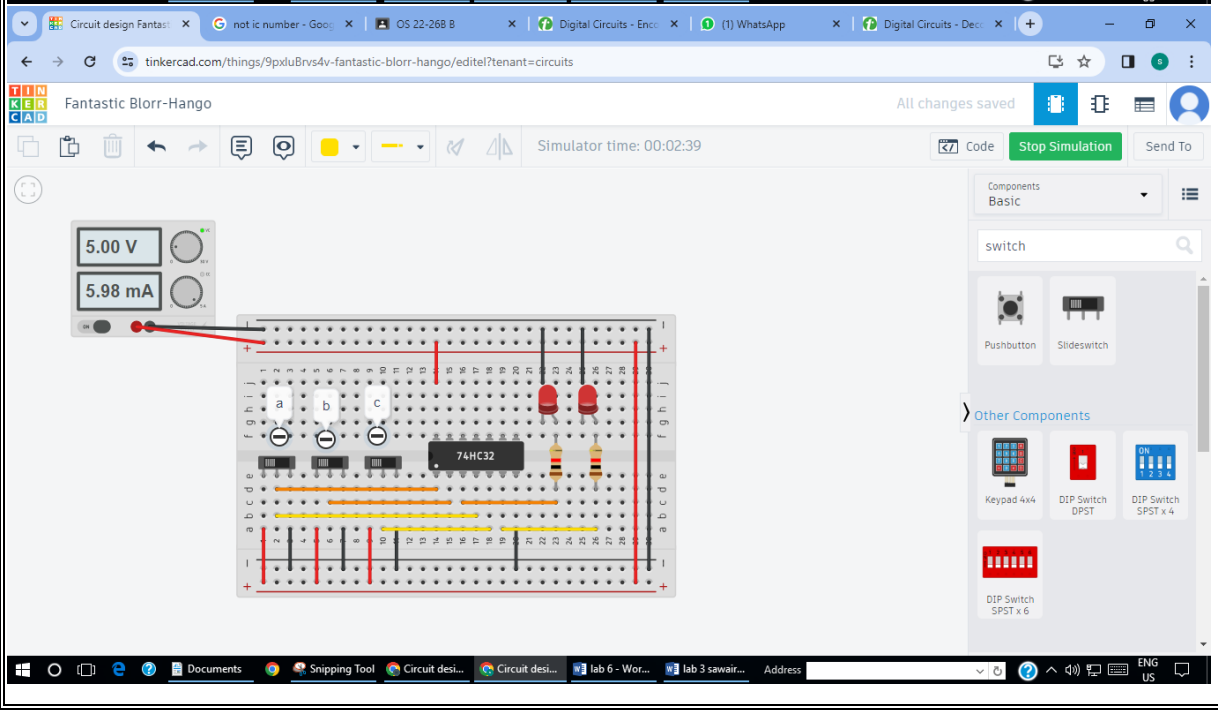
Encoder:


Encoder:-
A
B
A+B
C
A+C
A B C A+B A+C
0 0 0 0 0
0 0 1 0 1
0 1 0 1 0
0 1 1 1 1
1 0 0 1 1
1 0 1 1 1
1 1 0 1 1
1 1 1 1 1

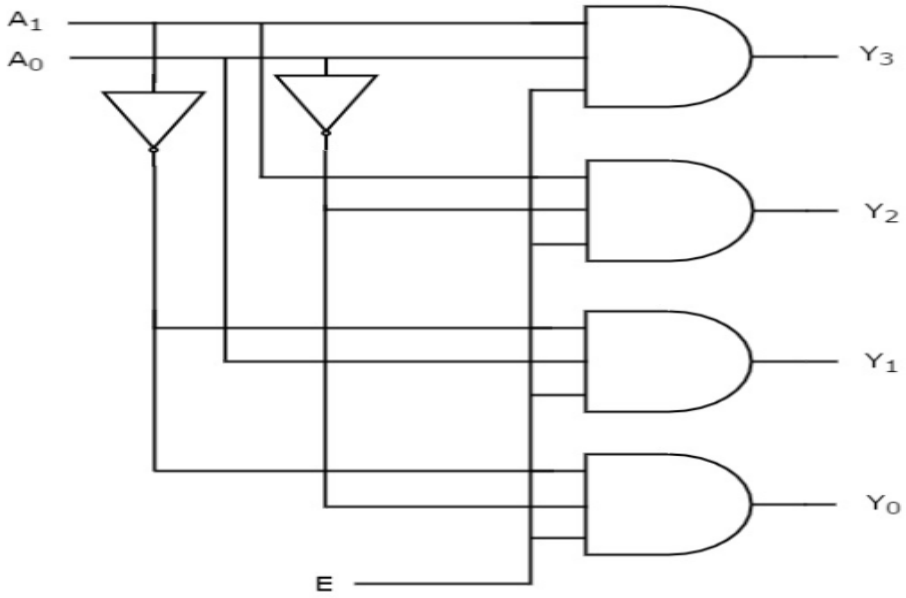

Y3
Y2
Y1
A1
A0

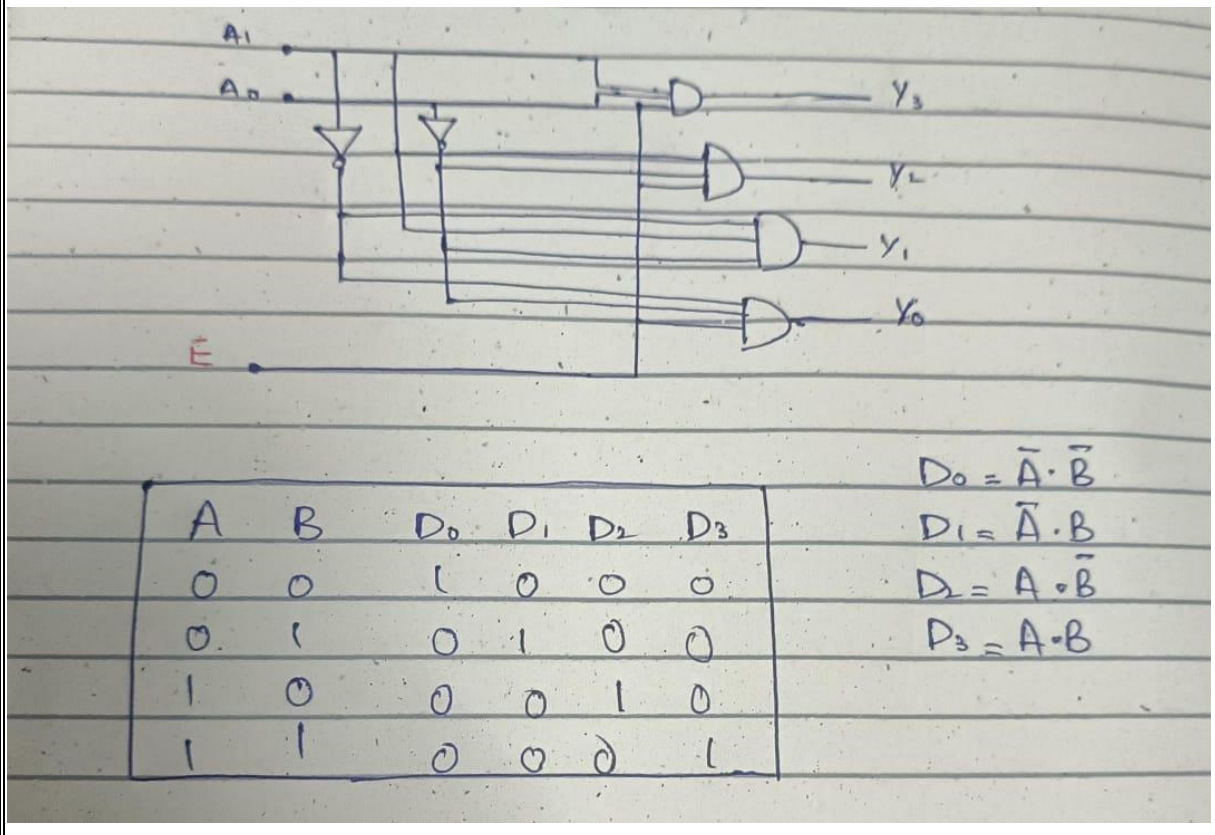

Fantastic Blorr-Hango
All changes saved
Simulator time: 00:02:11
Code
Stop Simulation
Send To
5.00 V
3.00 mA
74HC32
Components
Basic
switch
Pushbutton
Slideswitch
Other Components
Keypad 4x4
DIP Switch DPST
DIP Switch SPST x 4
DIP Switch SPST x 6

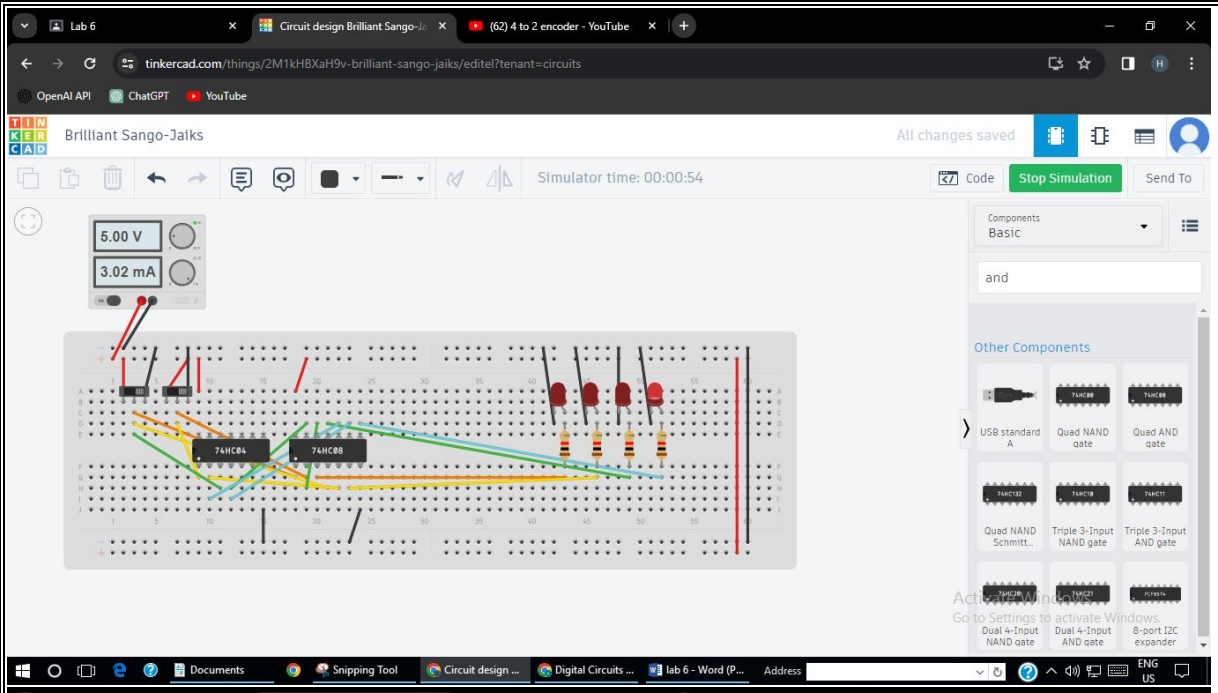

Fantastic Blorr-Hango
All changes saved
Simulator time: 00:02:30
Code
Stop Simulation
Send To
5.00 V
16.9 µA
74HC32
Components
Basic
switch
Pushbutton
Slideswitch
Other Components
Keypad 4x4
DIP Switch DPST
DIP Switch SPST x 4
DIP Switch SPST x 6

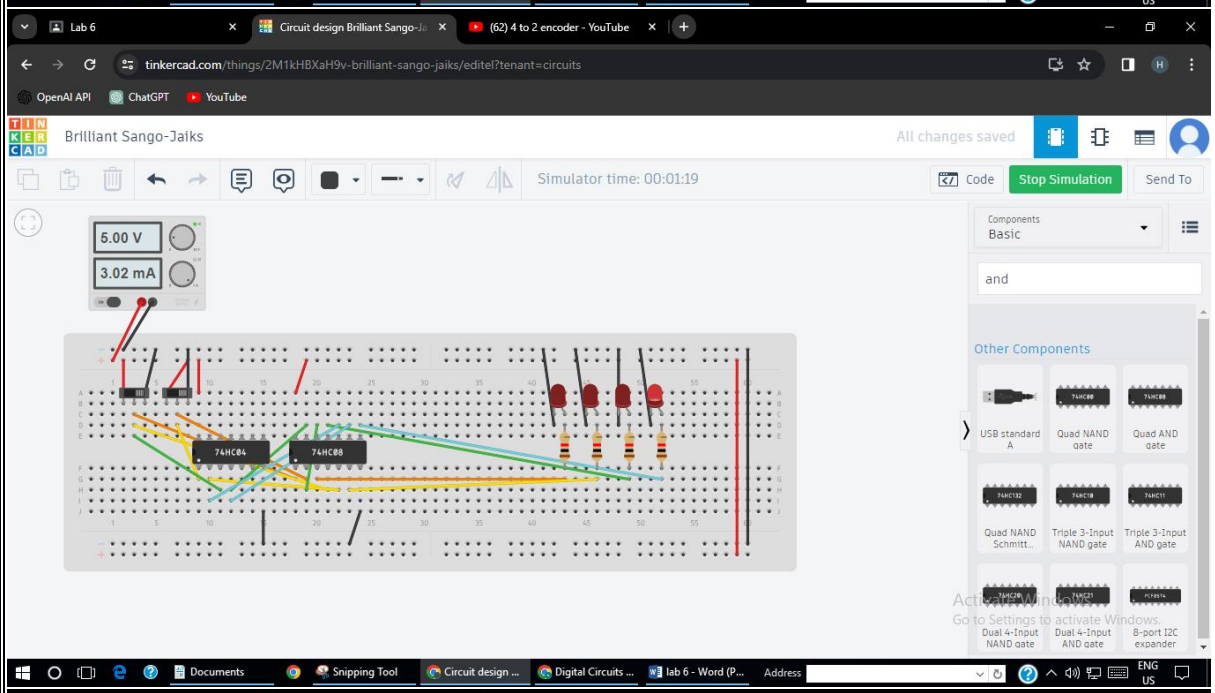

Fantastic Blorr-Hango
All changes saved
Simulator time: 00:02:39
Code
Stop Simulation
Send To
5.00 V
5.98 mA
74HC32
Components
Basic
switch
Pushbutton
Slideswitch
Other Components
Keypad 4x4
DIP Switch DPST
DIP Switch SPST x 4
DIP Switch SPST x 6

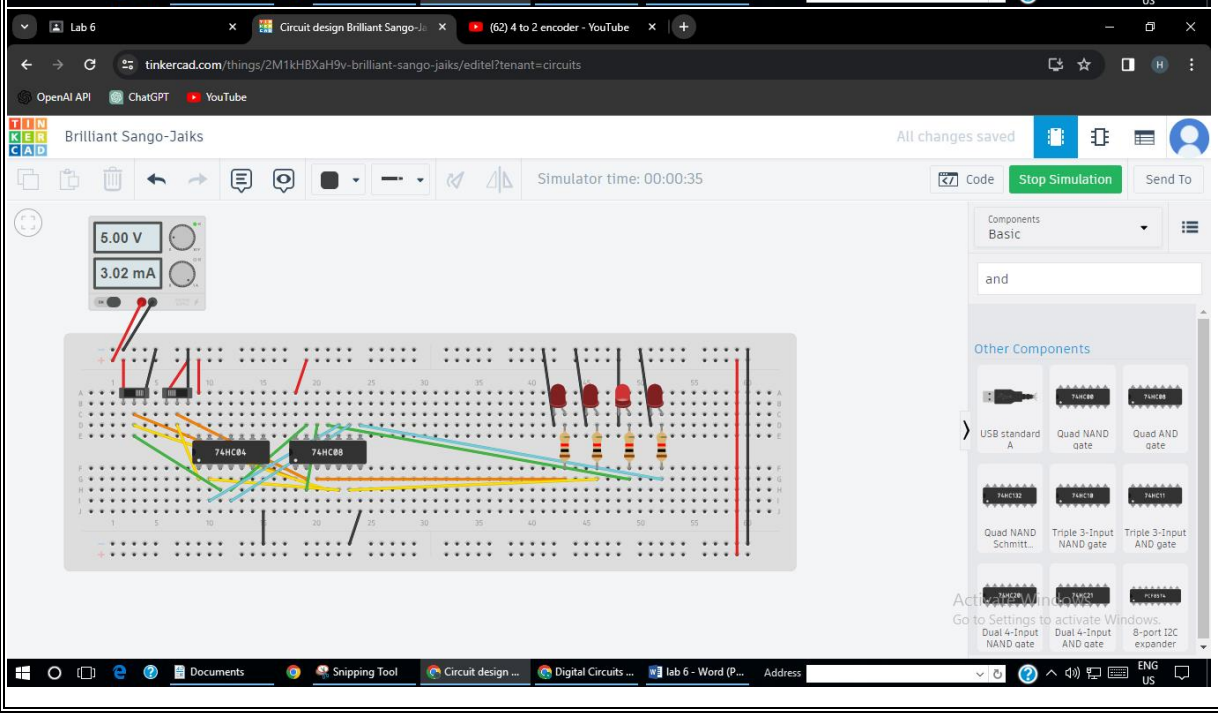
Decoder:


A1
A0
Y3
Y2
Y1
Y0
E


A1
A0
Y3
Y2
Y1
Y0
E
A B D0 D1 D2 D3
0 0 1 0 0 0
0 1 0 1 0 0
1 0 0 0 1 0
1 1 0 0 0 1
D0 = Ā · B̄
D1 = Ā · B
D2 = A · B̄
D3 = A · B


Lab 6
Circuit design Brilliant Sango-Ja
(62) 4 to 2 encoder - YouTube
tinkercad.com/things/2M1kHBXaH9v-brilliant-sango-jaiks/editel?tenant=circuits
OpenAI API
ChatGPT
YouTube
Brilliant Sango-Jaiks
All changes saved
Simulator time: 00:00:54
Code
Stop Simulation
Send To
5.00 V
3.02 mA
74HC04
74HC08
Components
Basic
and
Other Components
USB standard A
Quad NAND gate
Quad AND gate
Quad NAND Schmitt...
Triple 3-Input NAND gate
Triple 3-Input AND gate
Dual 4-Input NAND gate
Dual 4-Input AND gate
8-port I2C expander
Activate Windows
Go to Settings to activate Windows.
Documents
Snipping Tool
Circuit design ...
Digital Circuits ...
lab 6 - Word (P...
Address
ENG
US


Lab 6
Circuit design Brilliant Sango-Ja
(62) 4 to 2 encoder - YouTube
tinkercad.com/things/2M1kHBXaH9v-brilliant-sango-jaiks/editel?tenant=circuits
OpenAI API
ChatGPT
YouTube
Brilliant Sango-Jaiks
All changes saved
Simulator time: 00:01:19
Code
Stop Simulation
Send To
5.00 V
3.02 mA
74HC04
74HC08
Components
Basic
and
Other Components
USB standard A
Quad NAND gate
Quad AND gate
Quad NAND Schmitt...
Triple 3-Input NAND gate
Triple 3-Input AND gate
Dual 4-Input NAND gate
Dual 4-Input AND gate
8-port I2C expander
Activate Windows
Go to Settings to activate Windows.
Documents
Snipping Tool
Circuit design ...
Digital Circuits ...
lab 6 - Word (P...
Address
ENG
US


Lab 6
Circuit design Brilliant Sango-Ja
(62) 4 to 2 encoder - YouTube
tinkercad.com/things/2M1kHBXaH9v-brilliant-sango-jaiks/editel?tenant=circuits
OpenAI API
ChatGPT
YouTube
Brilliant Sango-Jaiks
All changes saved
Simulator time: 00:00:35
Code
Stop Simulation
Send To
5.00 V
3.02 mA
74HC04
74HC08
Components
Basic
and
Other Components
USB standard A
Quad NAND gate
Quad AND gate
Quad NAND Schmitt...
Triple 3-Input NAND gate
Triple 3-Input AND gate
Dual 4-Input NAND gate
Dual 4-Input AND gate
8-port I2C expander
Activate Windows
Go to Settings to activate Windows.
Documents
Snipping Tool
Circuit design ...
Digital Circuits ...
lab 6 - Word (P...
Address
ENG
US

COMPUTER ARCHITECTURE AND LOGIC DESIGN

8086 EMULATOR

LAB 07

SUBMITTED BY: : Hafsah tahir

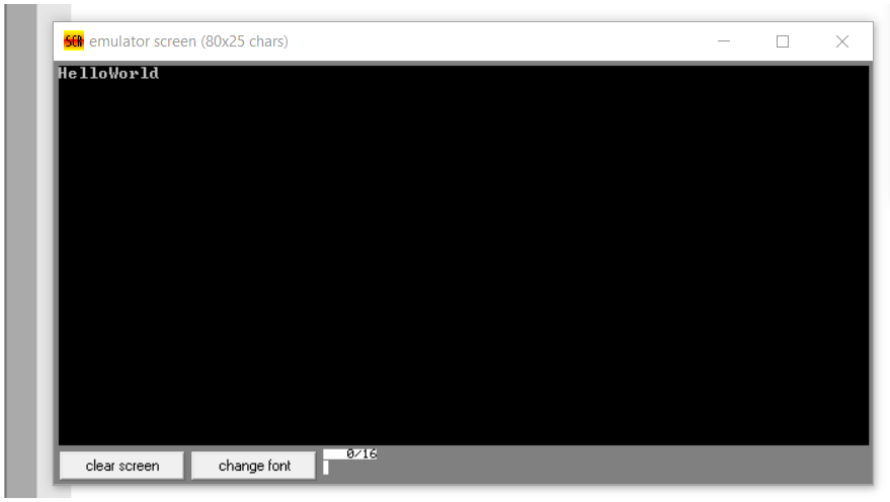
Sawaira saeed

Rabia batool

Hello World Program 1

```
org 100h
.data                                                    ; Declare variables
hello_message db 'Hello','World',0dh,0ah,'$'
.code                                                    ; Write code
main proc
mov ax,@data                          ; Copy the address of data
mov ds,ax                                    ; Segment into DS register
mov dx,offset hello_message
mov ah,9                                  ; MS-Dos Function to display string
INT 21H
mov ax,4C00h              ; Halt the program and return control to OS
INT 21H
main endp
```

Output console:



Hello World Program 2

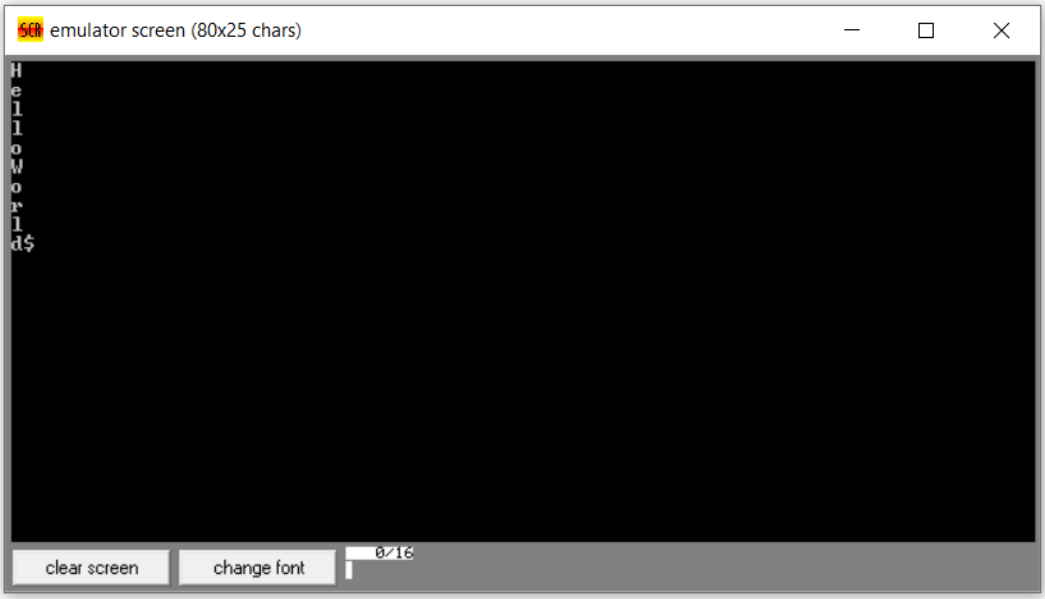
Assembly language code for vertically displaying hello world

```
org 100h

mov ah, 02h      ; Function to display character
mov dx, 'H'      ; Display 'H'
int 21h

mov dx, 0Dh      ; Move to a new line
int 21h
mov dx, 0Ah      ; Move to a new line
int 21h

mov dx, 'e'      ; Display 'e'
int 21h

mov dx, 0Dh      ; Move to a new line
int 21h
mov dx, 0Ah      ; Move to a new line
int 21h

mov dx, 'l'      ; Display 'l'
int 21h

mov dx, 0Dh      ; Move to a new line
int 21h
```

```
mov dx, 0Ah       ; Move to a new line
int 21h

mov dx, 'l'       ; Display 'l'
int 21h

mov dx, 0Dh       ; Move to a new line
int 21h
mov dx, 0Ah       ; Move to a new line
int 21h

mov dx, 'o'       ; Display 'o'
int 21h

mov dx, 0Dh       ; Move to a new line
int 21h
mov dx, 0Ah       ; Move to a new line
int 21h

mov dx, 'W'       ; Display 'W'
int 21h

mov dx, 0Dh       ; Move to a new line
int 21h
mov dx, 0Ah       ; Move to a new line
int 21h

mov dx, 'o'       ; Display 'o'
int 21h
```

```
mov dx, 0Dh      ; Move to a new line
int 21h
mov dx, 0Ah      ; Move to a new line
int 21h

mov dx, 'r'      ; Display 'r'
int 21h

mov dx, 0Dh      ; Move to a new line
int 21h
mov dx, 0Ah      ; Move to a new line
int 21h

mov dx, 'l'      ; Display 'l'
int 21h

mov dx, 0Dh      ; Move to a new line
int 21h
mov dx, 0Ah      ; Move to a new line
int 21h

mov dx, 'd'      ; Display 'd'
int 21h

mov dx, '$'      ; End the program
int 21h
mov ah, 4Ch      ; Halt the program and return control to OS
int 21h
```

Output console:



Hello World Program 3

Spaced hello world

```
org 100h

mov ah, 02h      ; Function to display character

mov dx, 'H'      ; Display 'H'
int 21h
```

```
mov dx, ' '       ; Display space
int 21h

mov dx, ' '       ; Display space
int 21h

mov dx, ' '       ; Display space
int 21h

mov dx, 'e'       ; Display 'e'
int 21h

mov dx, ' '       ; Display space
int 21h

mov dx, ' '       ; Display space
int 21h

mov dx, ' '       ; Display space
int 21h

mov dx, 'l'       ; Display 'l'
int 21h

mov dx, ' '       ; Display space
int 21h

mov dx, ' '       ; Display space
int 21h
```

```
mov dx, ' '      ; Display space
int 21h

mov dx, 'l'      ; Display 'l'
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, 'o'      ; Display 'o'
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, ' '      ; Display space
int 21h
```

```
mov dx, 'W'      ; Display 'W'
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, 'o'      ; Display 'o'
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, 'r'      ; Display 'r'
int 21h

mov dx, ' '      ; Display space
```

```
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, 'l'      ; Display 'l'
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, 'd'      ; Display 'd'
int 21h

mov dx, ' '      ; Display space
int 21h

mov dx, ' '      ; Display space
int 21h
```

```
mov dx, ' '       ; Display space
int 21h


mov dx, '$'       ; End the program
int 21h


mov ah, 4Ch       ; Halt the program and return control to OS
int 21h
```

Output console:

Hello World Program 4

Text on two different lines

```
org 100h

mov ah, 09h        ; Function to display string

mov dx, offset hello_message   ; Display 'Hello' on the first line
int 21h

mov dx, offset newline      ; Move to a new line
int 21h

mov dx, offset world_message   ; Display 'World' on the second line
int 21h

mov ah, 4Ch        ; Function to exit the program
int 21h

hello_message db 'Hello$'     ; 'Hello' text
world_message db 'World$'     ; 'World' text
newline db 0Dh, 0Ah, '$'      ; Newline characters with a terminating dollar sign
```

Output Console:

COURSE TITTLE:

Computer Architecture and logic design

Submitted to:

Eng. Shoaib

Submitted by:

Rabia batool

(2022-BSE-064)

TASKS

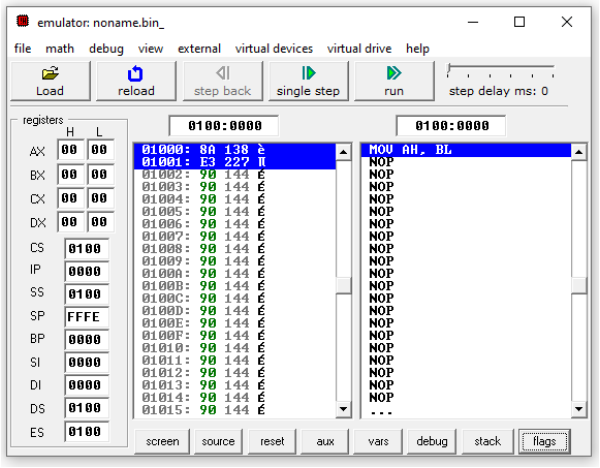
Assemble and trace following programs using DEBUG. For each program, write down the contents of only those registers and flag bits which have been modified by the given program.

❖ Program 1:

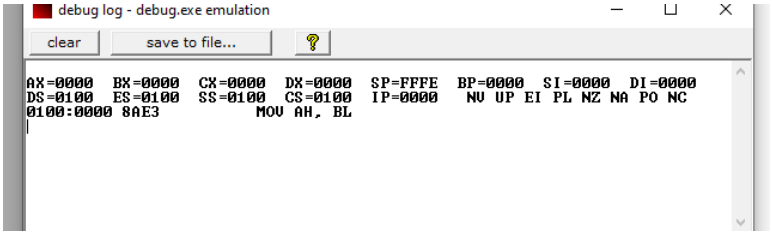
[AX=0020, BX=00AA]

mov ah, bl

FLAG BIT UPDATE



DEBUG

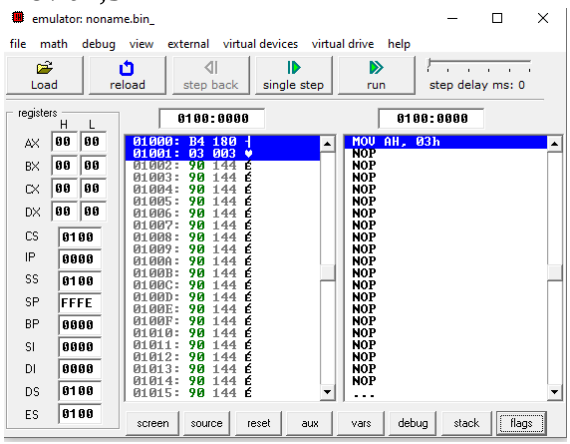


❖Program 2:

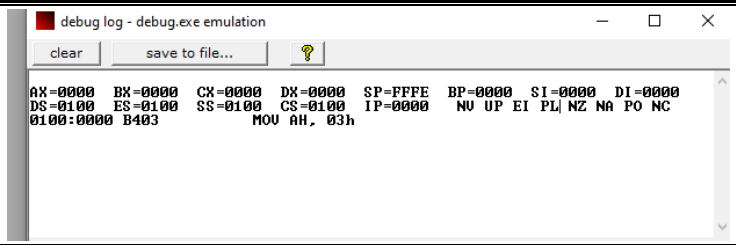
[AX=06AF]

mov ah,3

FLAG BIT



DEBUG;

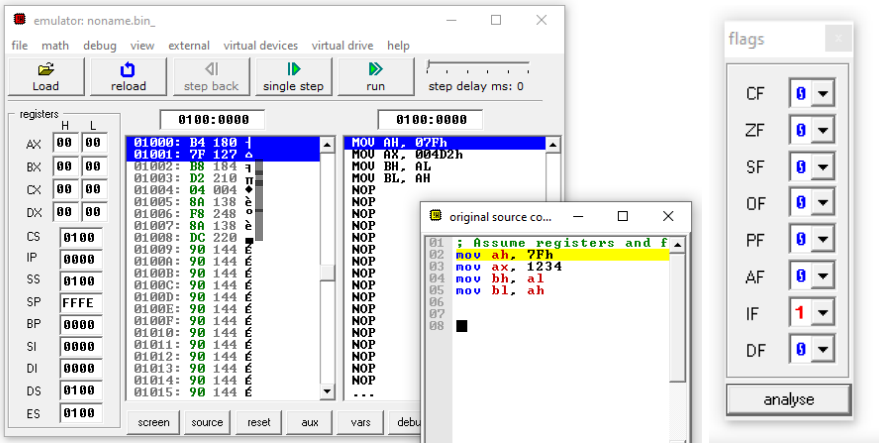


❖Program 3:

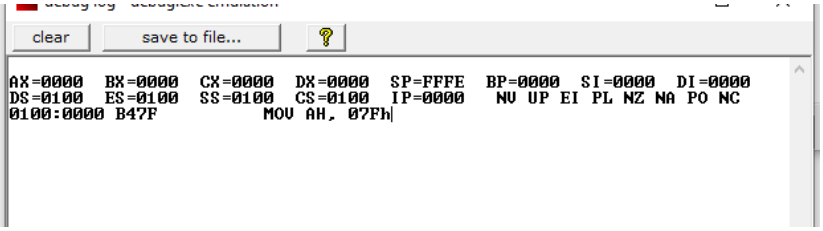
[Assume registers and flag bits are set to default values]

```
mov     ah,7Fh
mov     ax,1234
mov     bh, al
mov     bl, ah
```

FLAG BIT



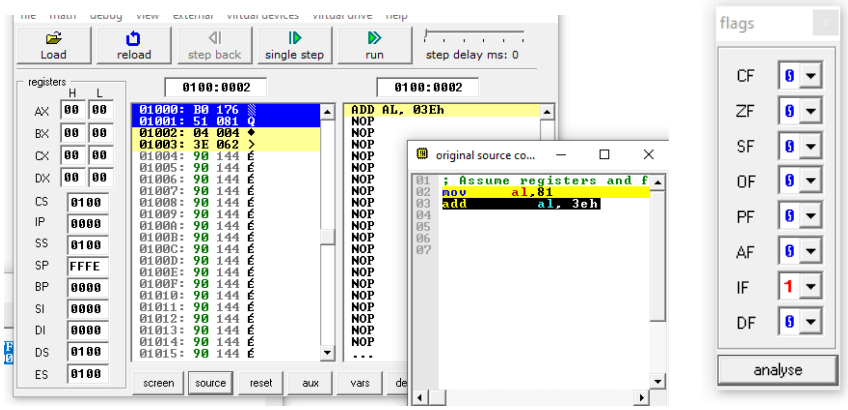
DEBUG:



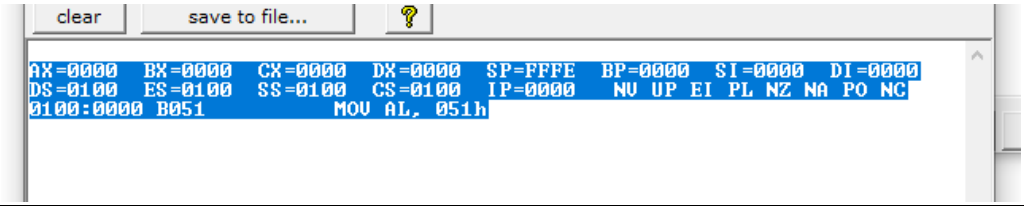
❖Program 4:

[Assume registers and flag bits are set to default values]

```
mov     al,81
add     al,3Eh
```



DEBUGG:

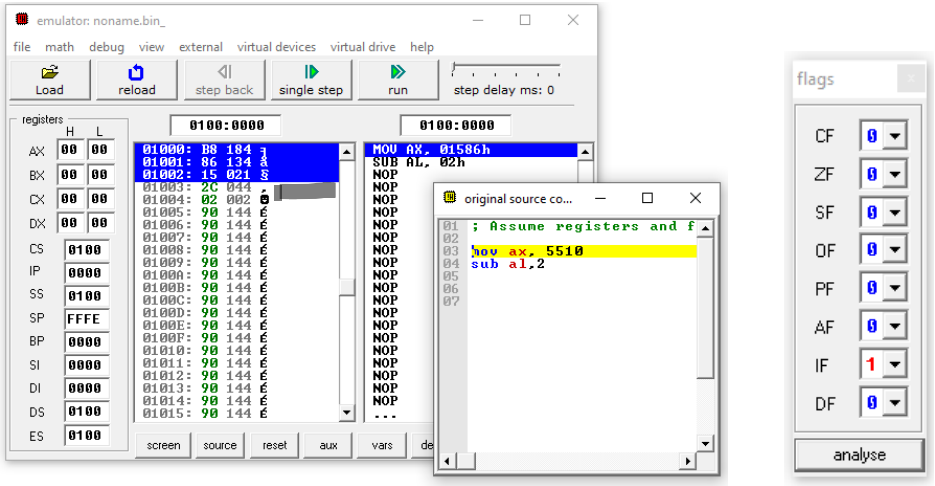


❖Program 5:

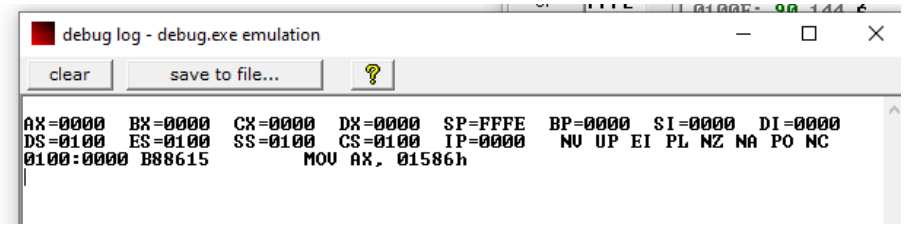
[Assume registers and flag bits are set to default values]

mov ax, 5510

sub al,2



DEBUG:

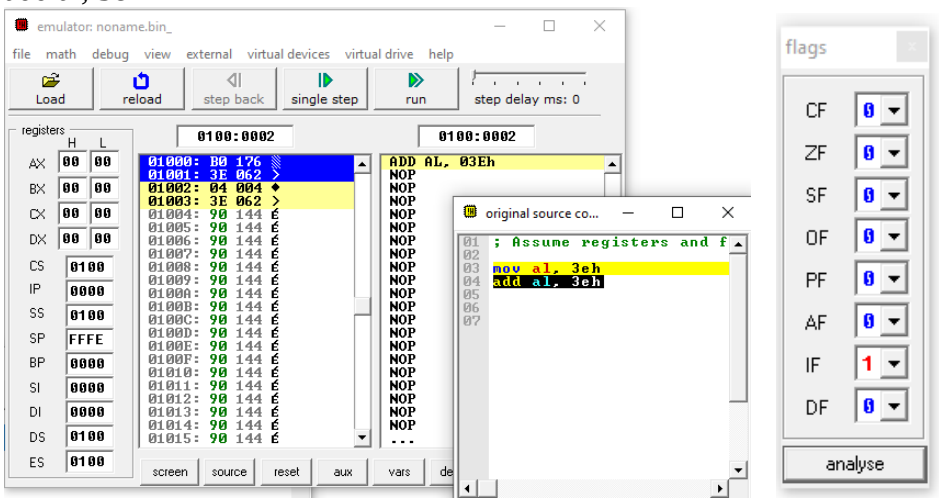


❖Program 6:

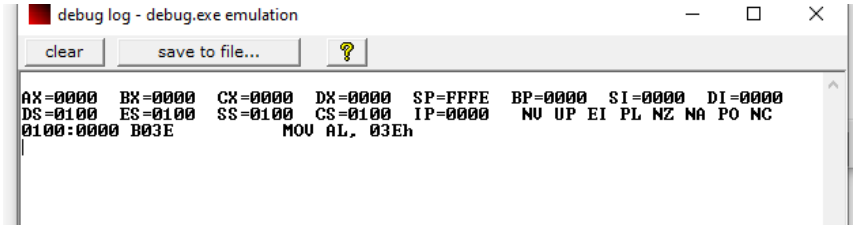
[Assume registers and flag bits are set to default values]

mov al, 3eh

add al, 3eh



DEBUG:

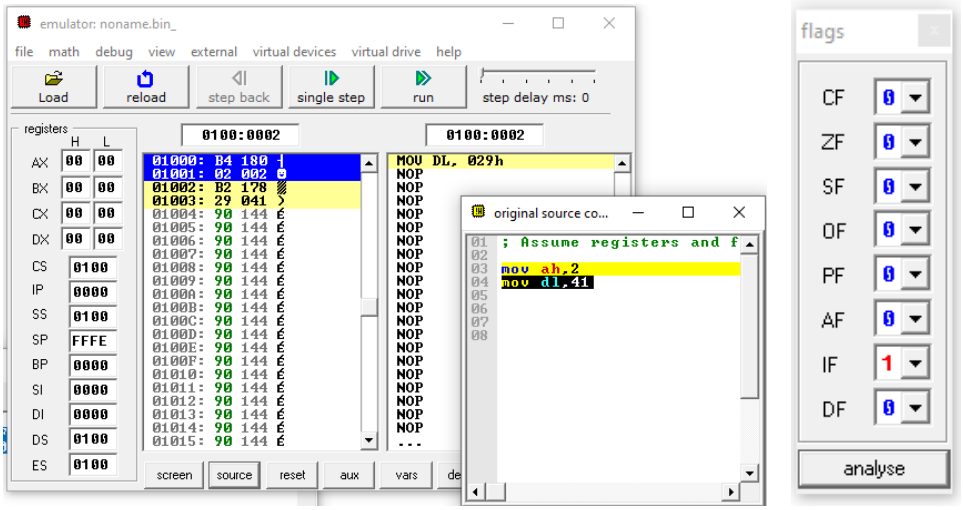


❖Program 7:

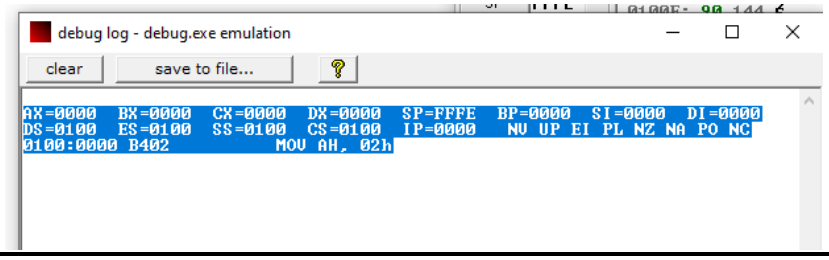
[Assume registers and flag bits are set to default values]

mov ah,2

mov dl,41



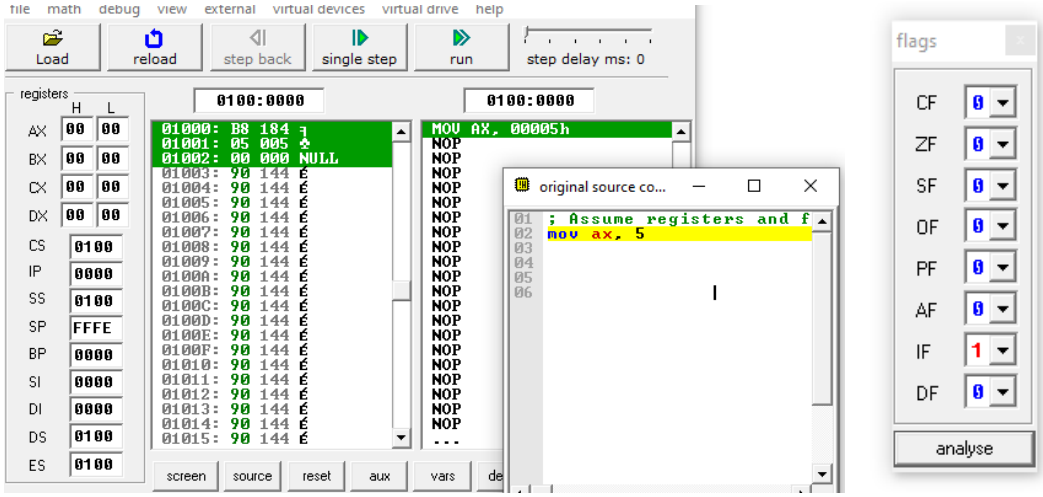
DEBUG:



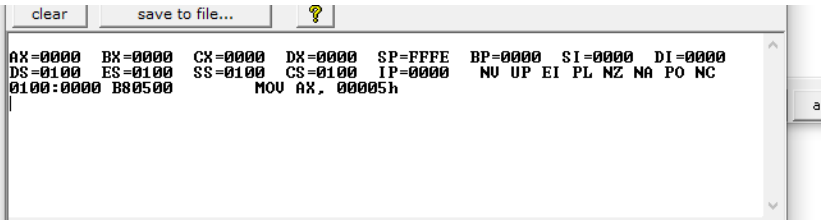
❖Program 8:

[Assume registers and flag bits are set to default values]

mov ax, 5



DEBUG:

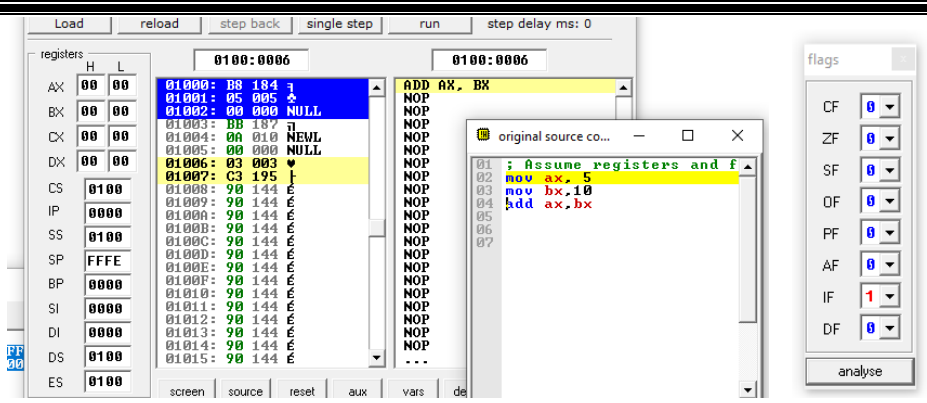


❖Program 9: [Assume registers and flag bits are set to default values]

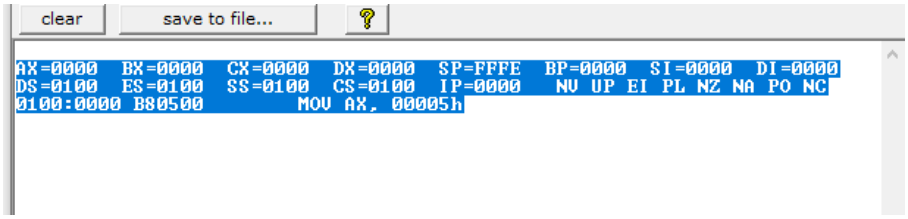
mov ax, 5

mov bx,10

add ax,bx



DEBUGG:



Program 10: [Assume registers and flag bits are set to default values]

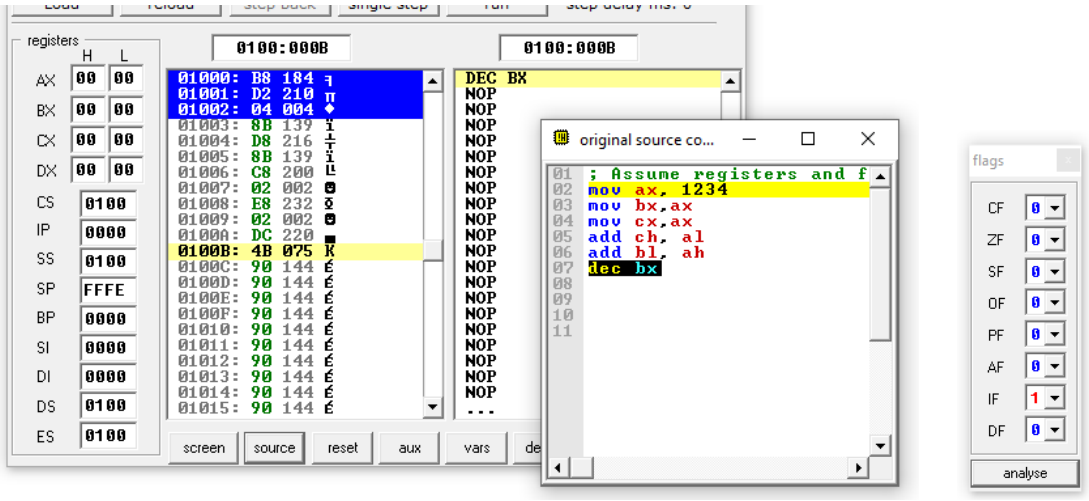
mov ax, 1234

mov bx,ax mov cx,ax add ch, al

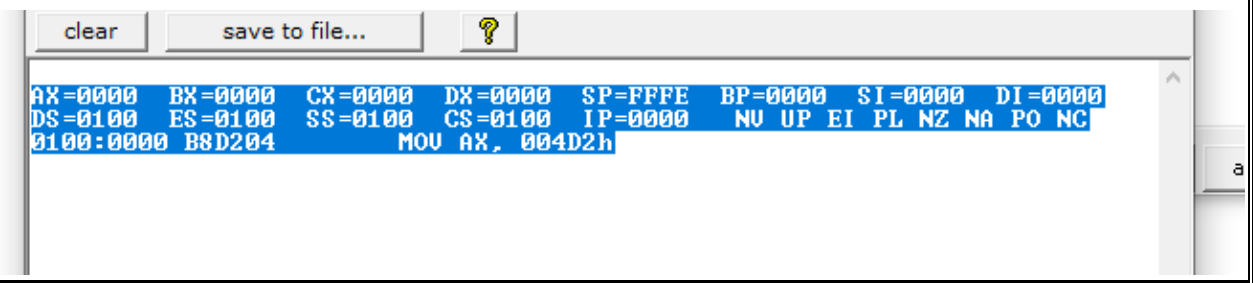
add bl, ah

add ax, FFFF

dec bx



DEBUGG:


clear
save to file...
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0100 ES=0100 SS=0100 CS=0100 IP=0000 NV UP EI PL NZ NA PO NC
0100:0000 B8D204 MOV AX, 004D2h

COURSE TITTLE:

Computer Architecture and logic design

Submitted to:

Eng. Shoaib

Submitted by:

Rabia batool

(2022-BSE-064)

TASKS

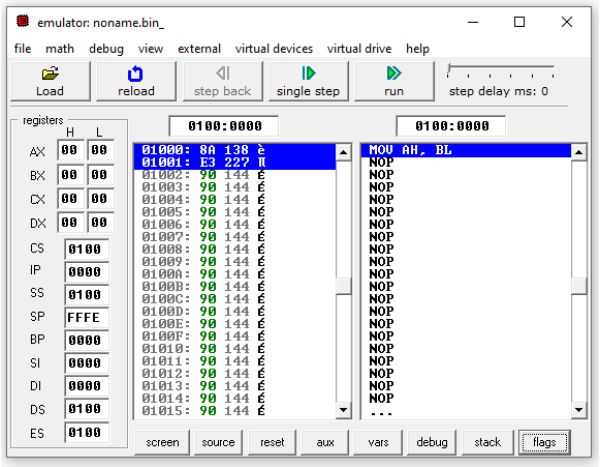
Assemble and trace following programs using DEBUG. For each program, write down the contents of only those registers and flag bits which have been modified by the given program.

❖ Program 1:

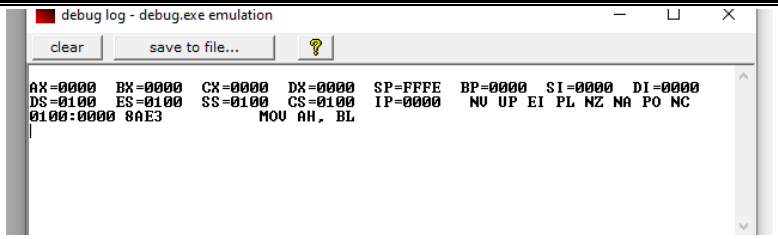
[AX=0020, BX=00AA]

mov ah, bl

FLAG BIT UPDATE



DEBUG

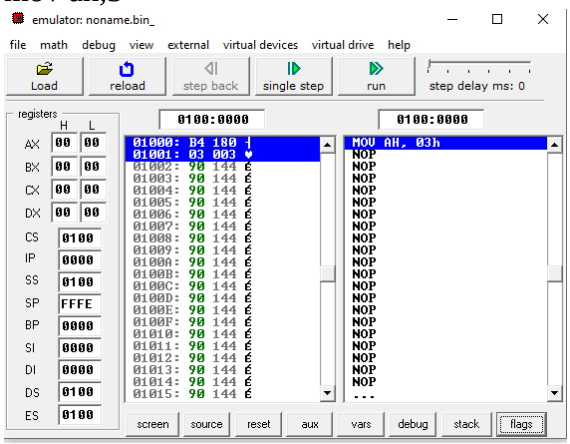


❖Program 2:

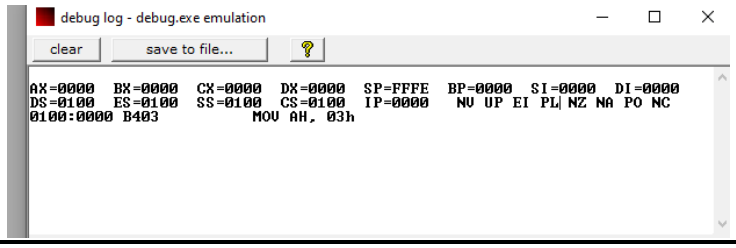
[AX=06AF]

mov ah,3

FLAG BIT



DEBUG;

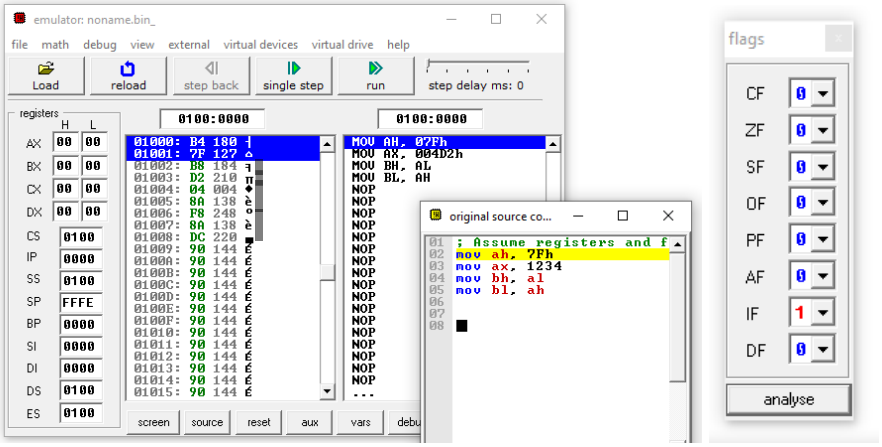


❖Program 3:

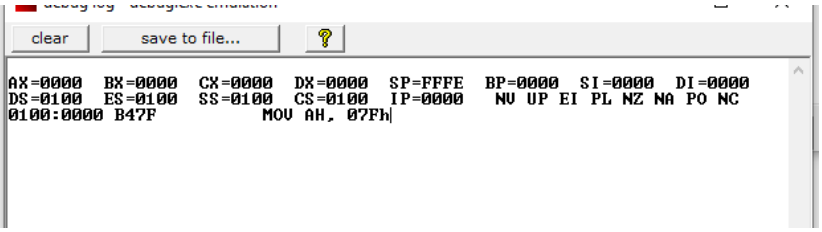
[Assume registers and flag bits are set to default values]

mov ah,7Fh
mov ax,1234
mov bh, al
mov bl, ah

FLAG BIT



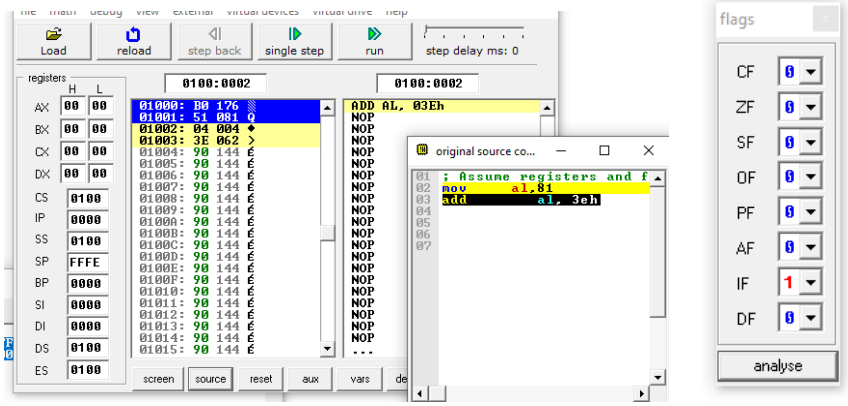
DEBUG:



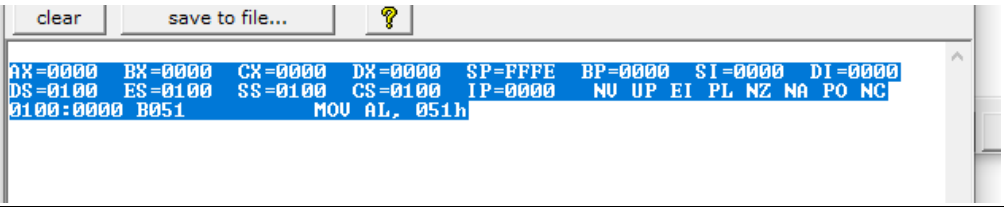
❖Program 4:

[Assume registers and flag bits are set to default values]

```
mov     al,81
add     al,3Eh
```



DEBUGG:

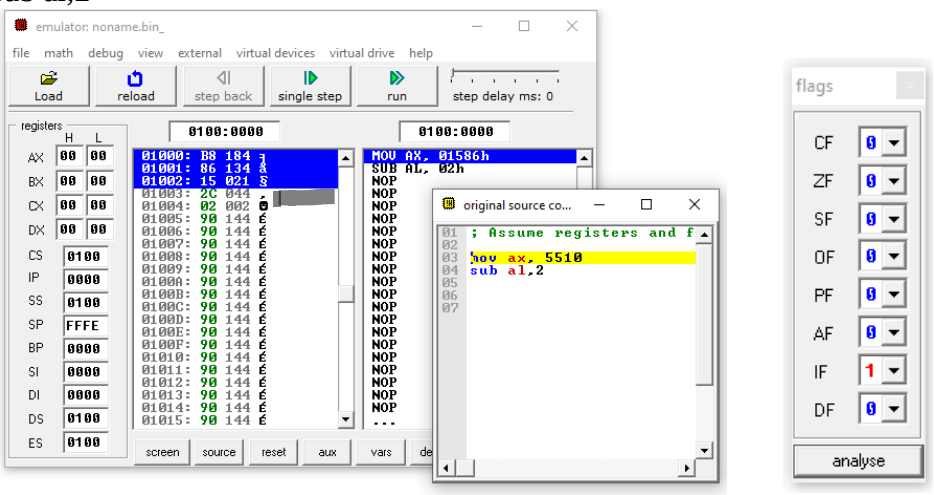


❖Program 5:

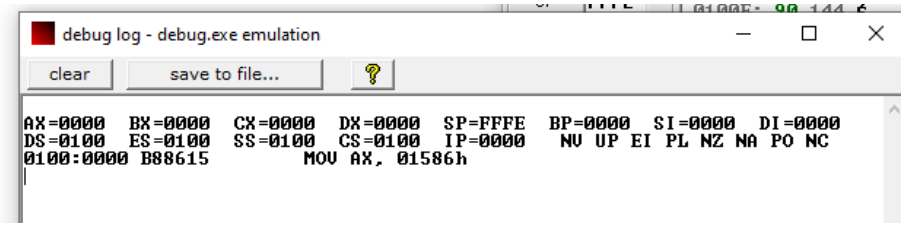
[Assume registers and flag bits are set to default values]

mov ax, 5510

sub al,2



DEBUG:

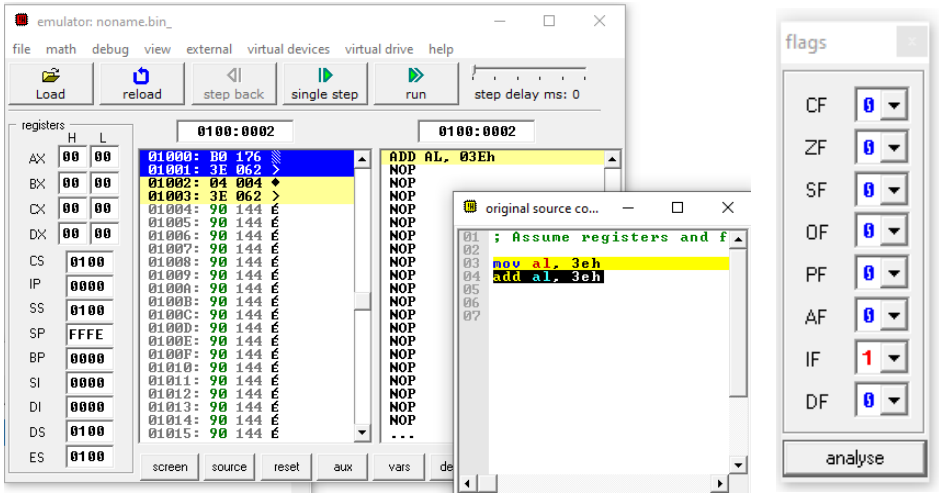


❖Program 6:

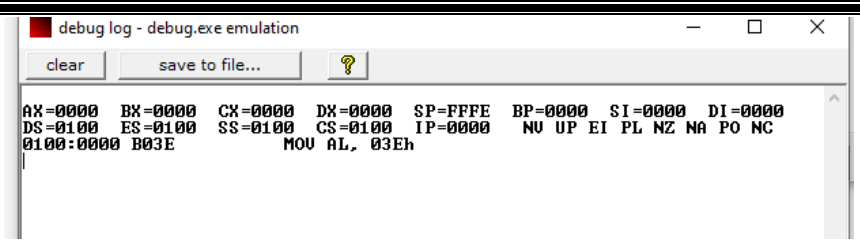
[Assume registers and flag bits are set to default values]

mov al, 3eh

add al, 3eh



DEBUG:

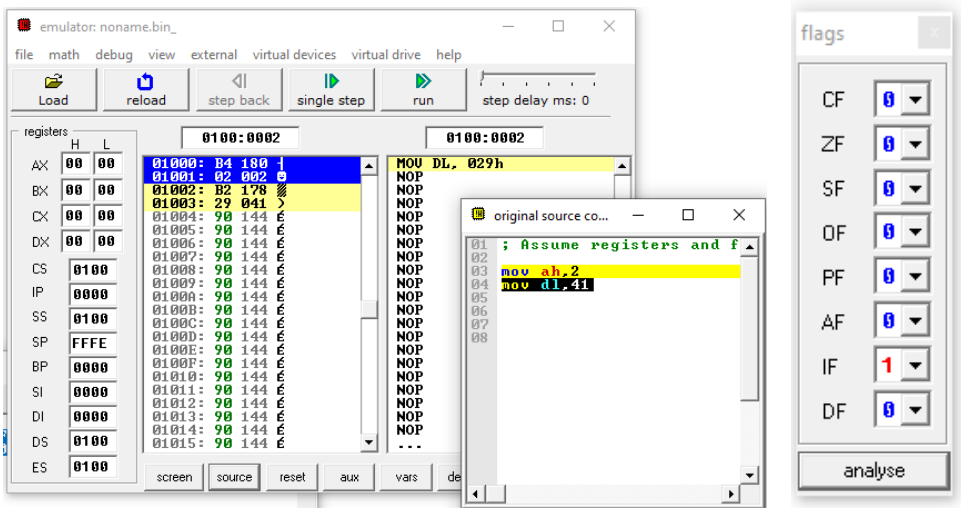


❖Program 7:

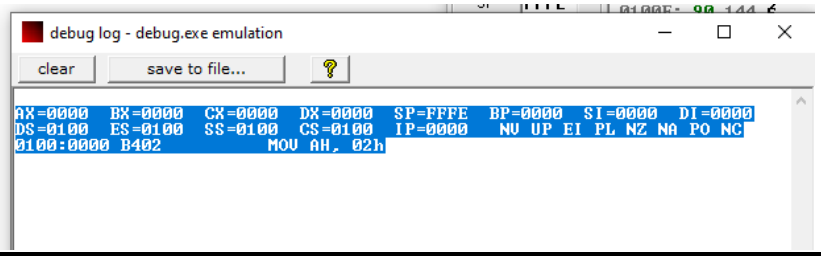
[Assume registers and flag bits are set to default values]

mov ah,2

mov dl,41



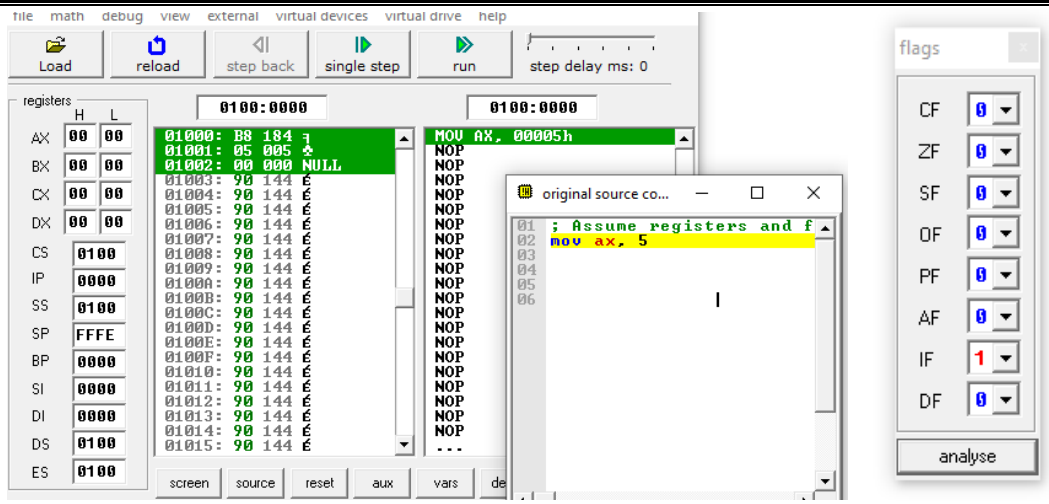
DEBUG:



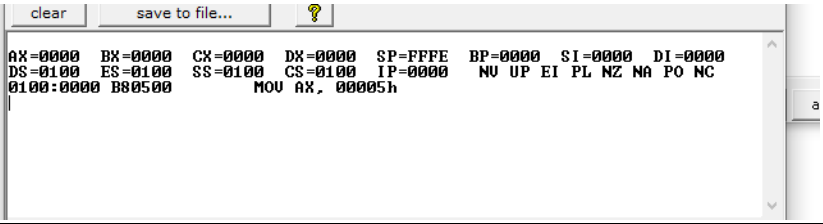
❖Program 8:

[Assume registers and flag bits are set to default values]

mov ax, 5



DEBUG:

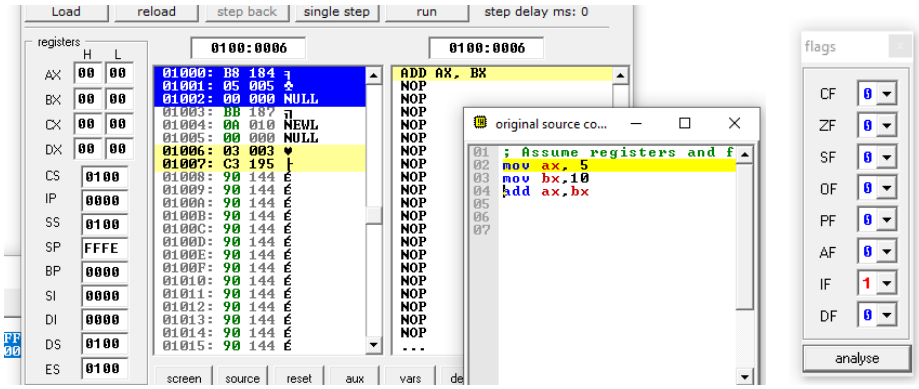


❖Program 9: [Assume registers and flag bits are set to default values]

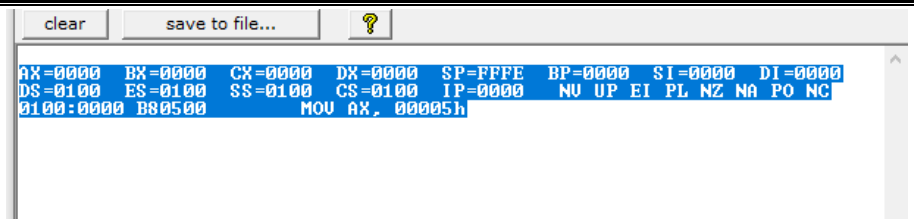
mov ax, 5

mov bx,10

add ax,bx



DEBUGG:



Program 10: [Assume registers and flag bits are set to default values]

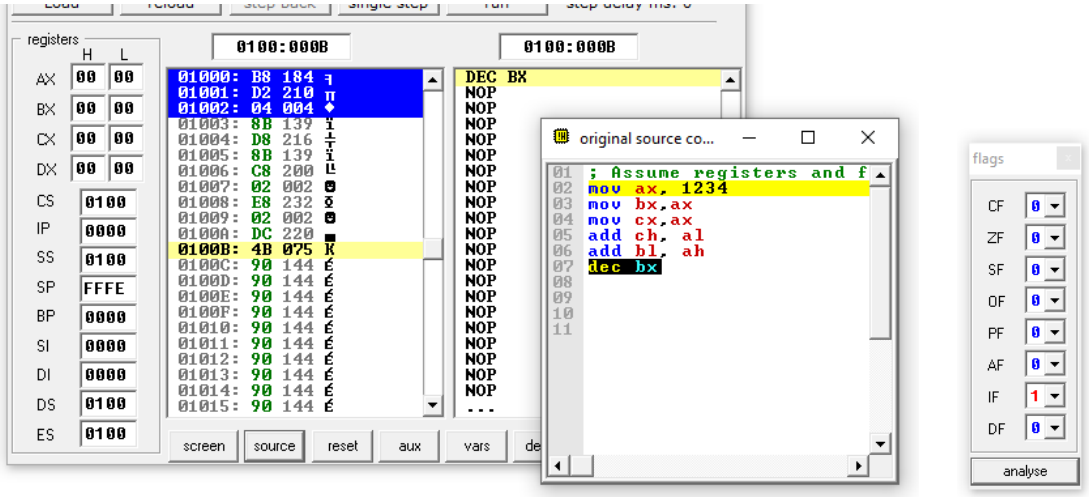
mov ax, 1234

mov bx,ax mov cx,ax add ch, al

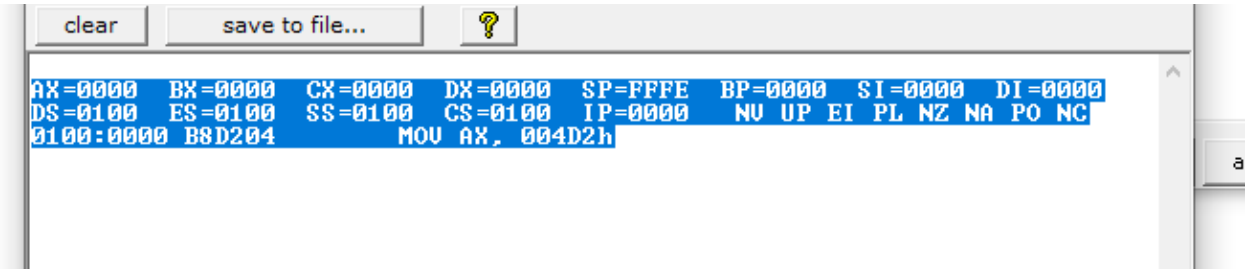
add bl, ah

add ax, FFFF

dec bx



DEBUGG:



Computer architecture

LAB # 08

Submitted to:

Sir shoib

Submitted by:

Rabia Batool

2022-BSE-064

Exercise 9.1

Write a program that input a character from user is in lowercase, the program will convert it to uppercase and will display it on console after conversion.

Hint: - The ASCII codes for lowercase letters (a-z) are 97-122. In order to convert a lowercase letter to uppercase letter, just subtract 32 from its ASCII code.

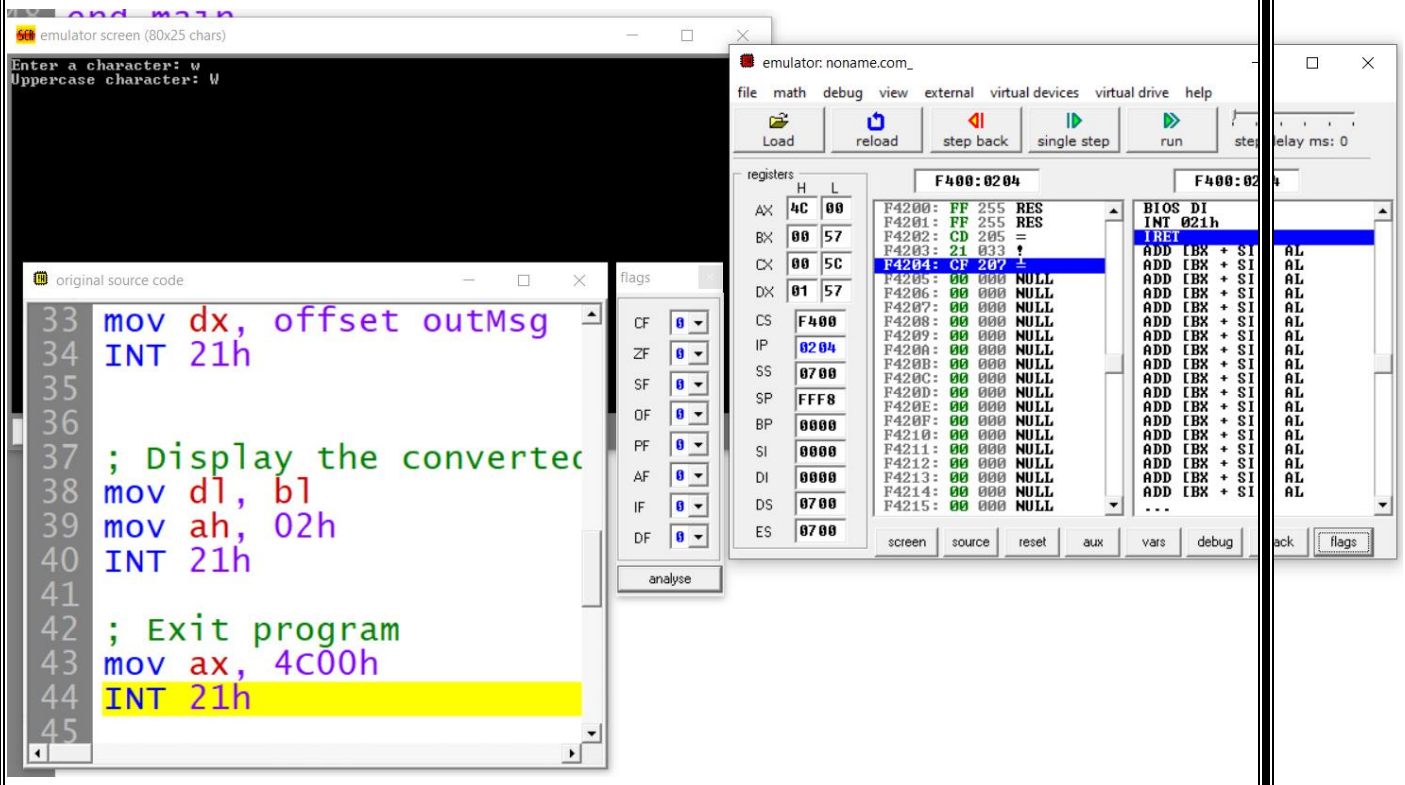
Code:

```
org 100h

.data
   inMsg db "Enter a character: $"
   outMsg db "Uppercase character: $"
   newline db 0dh, 0ah, '$'

.code
main proc
```

```
; Display input message
mov ah, 09h
mov dx, offset inMsg
INT 21h

; Get input from keyboard
mov ah, 01h
INT 21h
mov bl, al  ; Store the input character in bl

; Convert to uppercase by subtracting 32 from ASCII code
sub bl, 32



; Move cursor to new line
mov ax, @data
mov ds, ax
mov dx, offset newline
mov ah, 9
INT 21H

; Display output message
mov ah, 09h
mov dx, offset outMsg
INT 21h



; Display the converted character
mov dl, bl
```

```
    mov ah, 02h
    INT 21h

    ; Exit program
    mov ax, 4C00h
    INT 21h

main endp

end main
```

Output:



Exercise 9.2

Write a program that input a character from user. The program will display it ten times

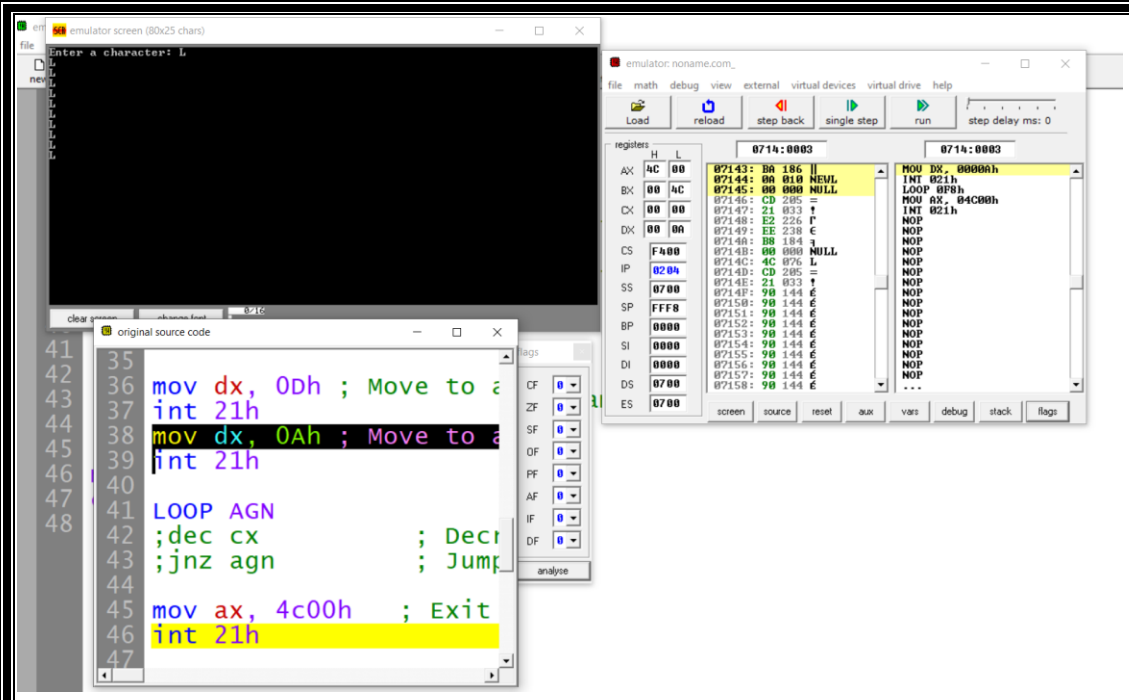
on screen in newline.

Code:

```
;To read a character from keyboard
org 100h
.data
   inMsg db "Enter a character: ",'$'
   newline db 0dh,0ah,'$'

.code
main proc

   ;To display input message
   mov ax, @data
   mov ds, ax
   mov dx, offset inMsg
   mov ah, 9
   INT 21H

   ;Get input from keyboard
   mov ah, 1h
   INT 21H
   mov bl, al

   ;Move cursor to new line
   mov ax, @data
   mov ds, ax
   mov dx, offset newline
```

```
    mov ah, 9
    INT 21H

    mov cx, 10  ; Counter for displaying the character ten times

agn:
    mov dl, bl      ; Display the character
    mov ah, 2h
    INT 21H

    mov dx, 0Dh ; Move to a new line
    INT 21h
    mov dx, 0Ah ; Move to a new line
    INT 21h

    LOOP AGN

    mov ax, 4c00h   ; Exit program
    INT 21h

main endp
end main
```

OUTPUT:


emulator screen (80x25 chars)
Enter a character: L
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
AX 4C 00
BX 00 4C
CX 00 00
DX 00 0A
CS F400
IP 0204
SS 0700
SP FFF8
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0714:0003
0714:0003
07143: BA 186 º
07144: 0A 010 NEWL
07145: 00 000 NULL
07146: CD 205 =
07147: 21 033 !
07148: E2 226 Γ
07149: EE 238 €
0714A: B8 184 ¸
0714B: 00 000 NULL
0714C: 4C 076 L
0714D: CD 205 =
0714E: 21 033 !
0714F: 90 144 É
07150: 90 144 É
07151: 90 144 É
07152: 90 144 É
07153: 90 144 É
07154: 90 144 É
07155: 90 144 É
07156: 90 144 É
07157: 90 144 É
07158: 90 144 É
MOV DX, 0000Ah
INT 021h
LOOP 0F8h
MOV AX, 04C00h
INT 021h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen source reset aux vars debug stack flags
original source code
mov dx, 0Dh ; Move to a
int 21h
mov dx, 0Ah ; Move to a
int 21h
LOOP AGN
;dec cx ; Decr
;jnz agn ; Jump
mov ax, 4c00h ; Exit
int 21h
CF ZF SF OF PF AF IF DF
analyse

“lab 10”

COURSE :

CALD

SUBMITTED TO :

Sir shoaib

SUBMITTED BY :

Rabia Batool (2022-BSE-064)

SECTION :

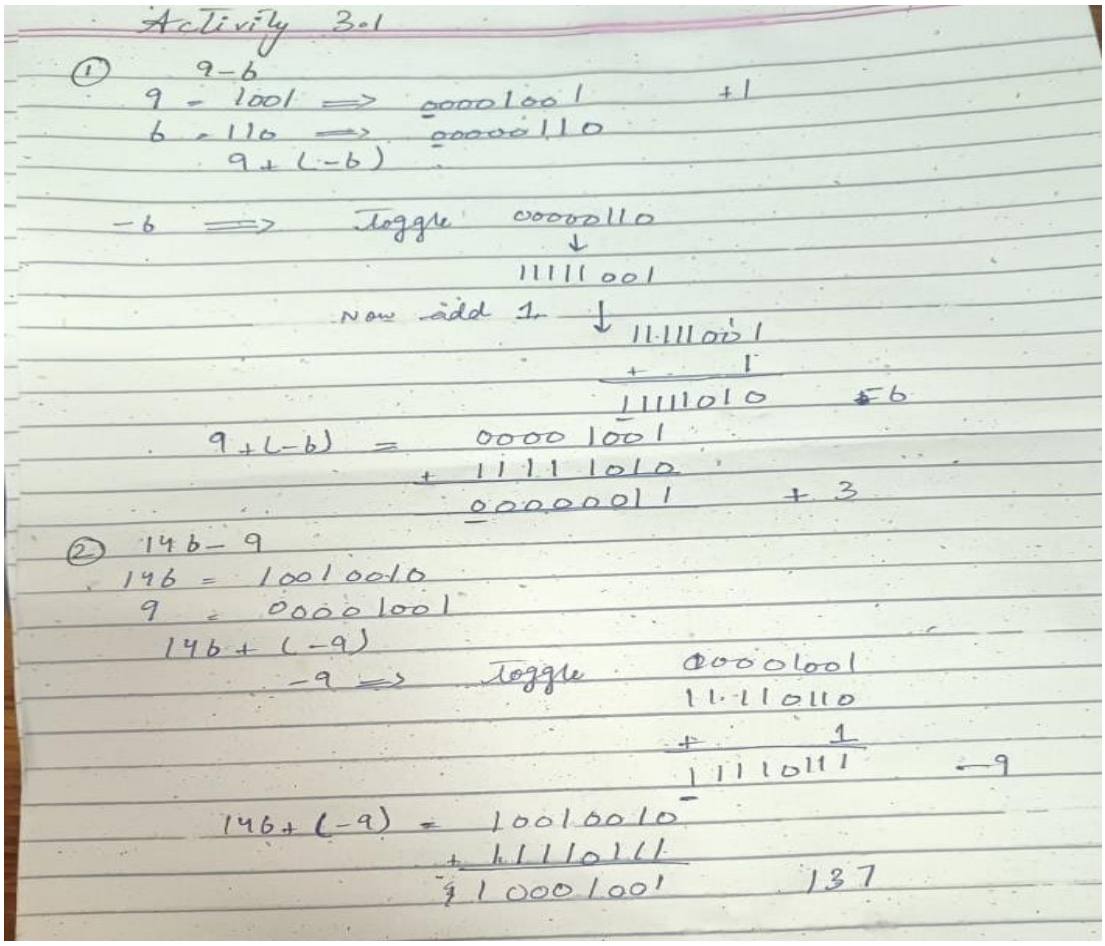
B

Activity #3.1

Activity 3.1:

Show the working of following instruction by taking the two's complement

- 9 – 6 _____ - _____ = _____
- 146 – 9_____ - _____ = _____

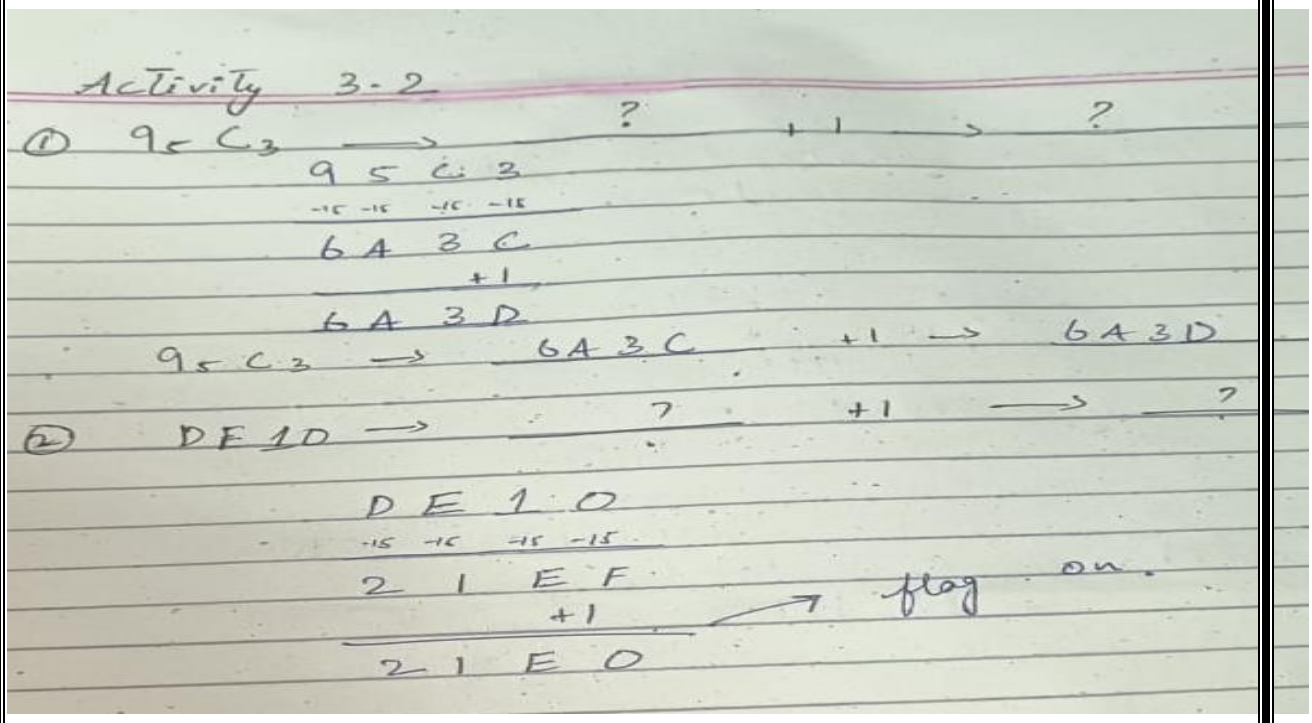


Activity #3.2

Activity 3.2:

$95C_3$ --> -------- + 1 -->--------

DE10 --> -------- + 1 --> -------



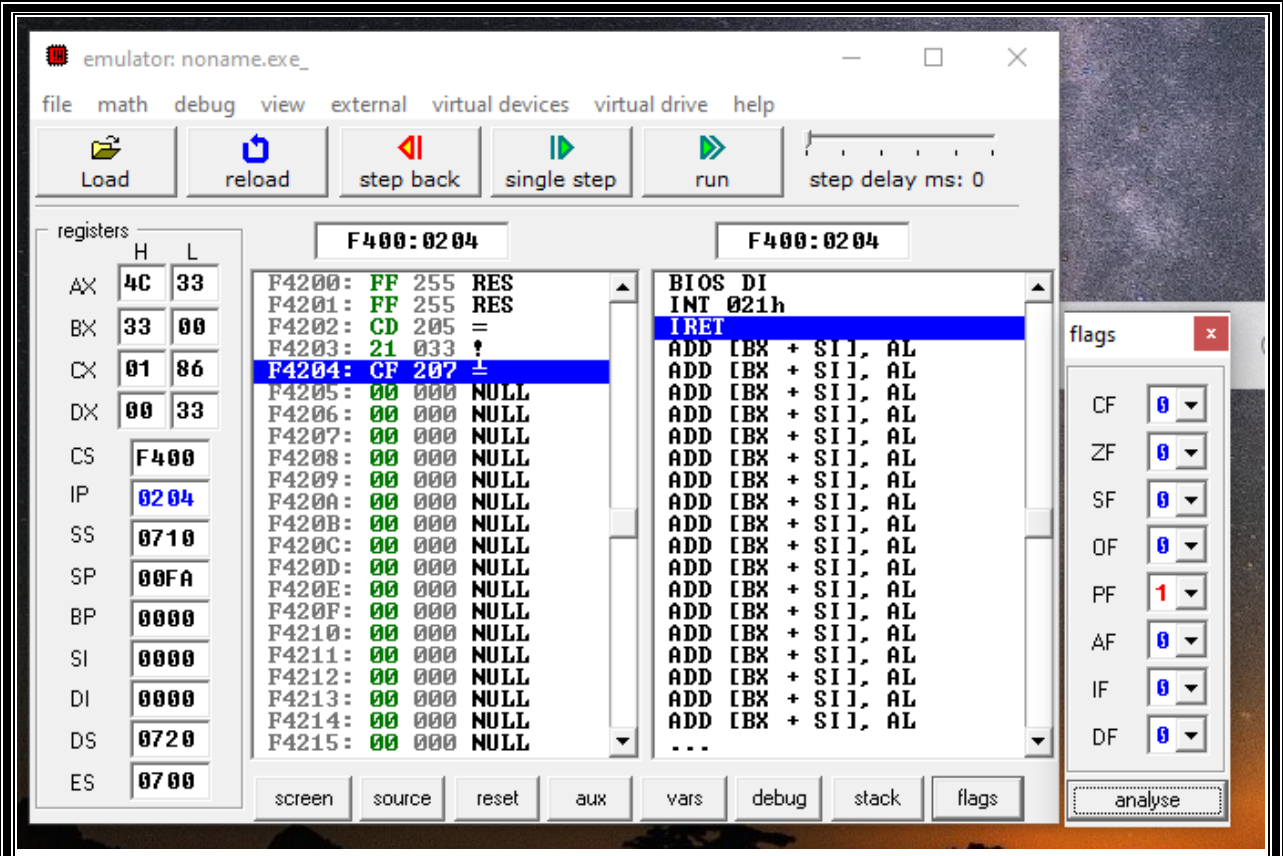
Activity #3.3:

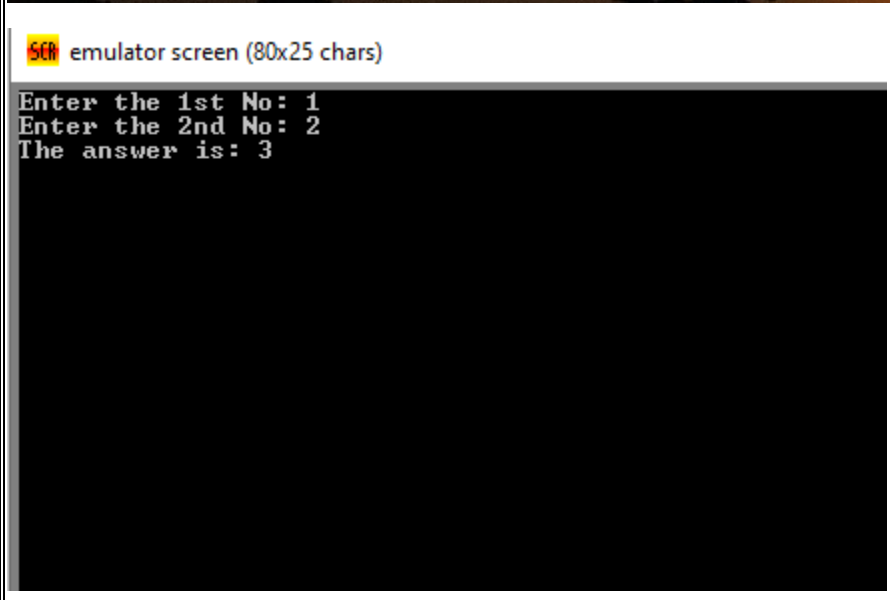
Activity 3.3: Addition of two single digits numbers

```
.model small

.stack 100h


.data

Num1 db 0

Num2 db 0

msgNum1 db "Enter the 1st No: $"
```

```
msgNum2 db 10,13,"Enter the 2nd No: $"
msgAns db 10,13,"The answer is: $"

.code
main proc
mov ax, @data
mov ds, ax  ; Initialize data segment once

; Input for Num1
mov dx, offset msgNum1
mov ah, 09h
int 21h
mov ah, 1
int 21h
sub al, 30h
mov Num1, al

; Input for Num2
mov dx, offset msgNum2
mov ah, 09h
int 21h
mov ah, 1
int 21h
sub al, 30h
mov Num2, al

; Addition
mov bh, Num1
add bh, Num2
```

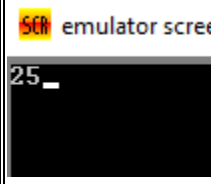
```
; Check for overflow (optional)
cmp bh, 9
ja overflow  ; Jump to overflow handler if needed

; Print the result
add bh, 30h
mov dx, offset msgAns
mov ah, 09h
int 21h
mov dl, bh
mov ah, 2
int 21h

; Exit
mov ah, 4ch
int 21h

overflow:  ; Overflow handling (if implemented)
; ...

main endp
end main
```

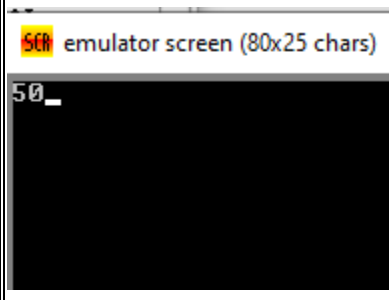

emulator: noname.exe_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 4C 33
BX 33 00
CX 01 86
DX 00 33
CS F400
IP 0204
SS 0710
SP 00FA
BP 0000
SI 0000
DI 0000
DS 0720
ES 0700
F400:0204
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ╧
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
...
flags
CF 0
ZF 0
SF 0
OF 0
PF 1
AF 0
IF 0
DF 0
analyse
screen source reset aux vars debug stack flags

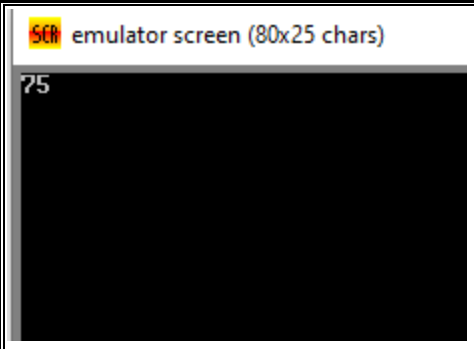

emulator screen (80x25 chars)
Enter the 1st No: 1
Enter the 2nd No: 2
The answer is: 3

Activity #3.4

```
include emu8086.inc ;include library
.data
num1 dw 0
num2 dw 0
; get the multi-digit signed number from the keyboard, and store the
; result in cx register:
call scan_num ; Get first number
mov num1, cx ; store first number:
call scan_num ; Get second number
mov num2, cx ; store second number:
mov ax, num1 ; Can not add two memory variables
add ax, num2 ; add num2 in ax and store result in ax
call print_num ; print ax value.
DEFINE_PRINT_NUM ; used by print_num proc
DEFINE_PRINT_NUM_UNS ; used by print_num proc
DEFINE_SCAN_NUM ; used by scan_num proc
END
```



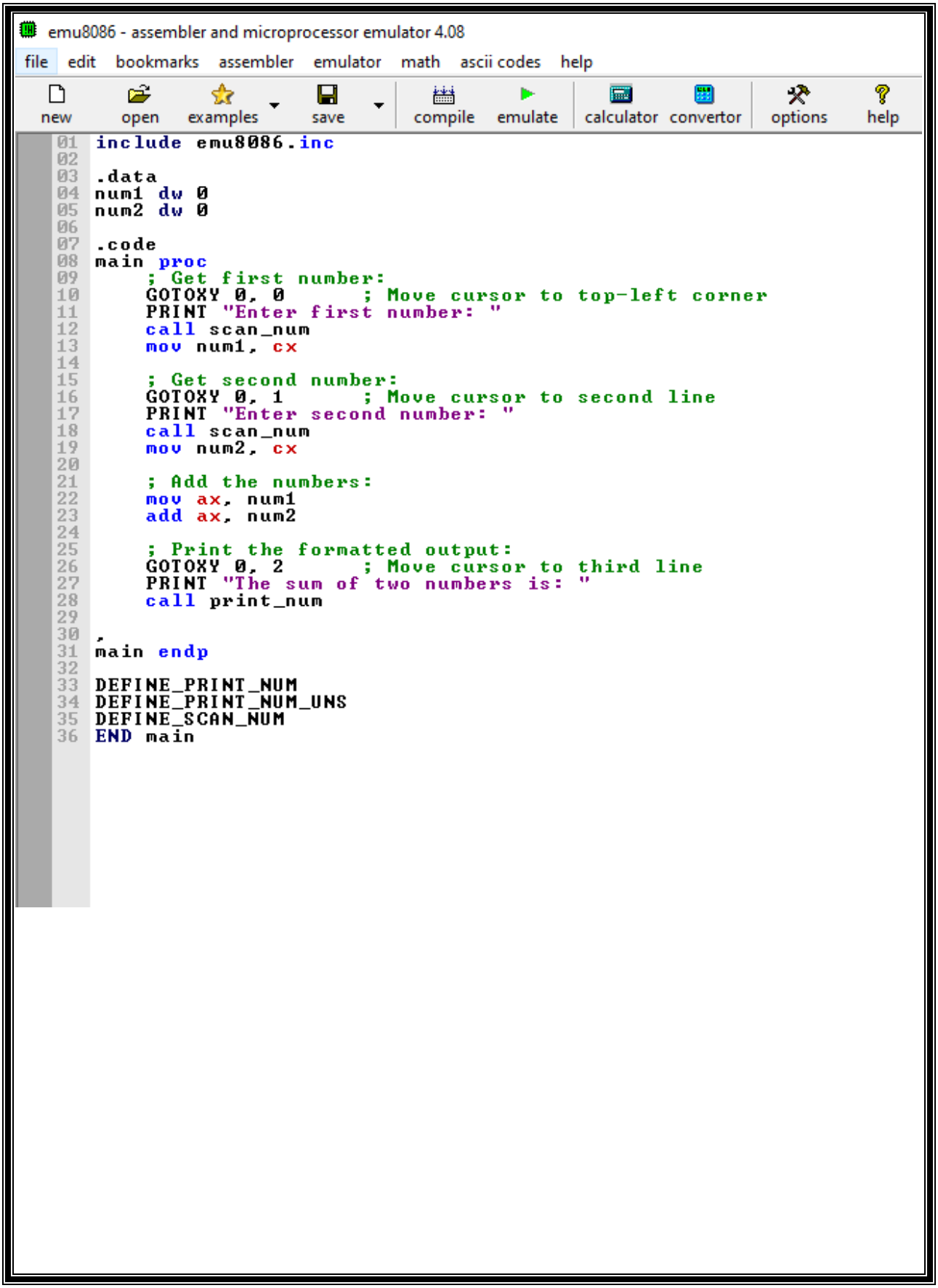


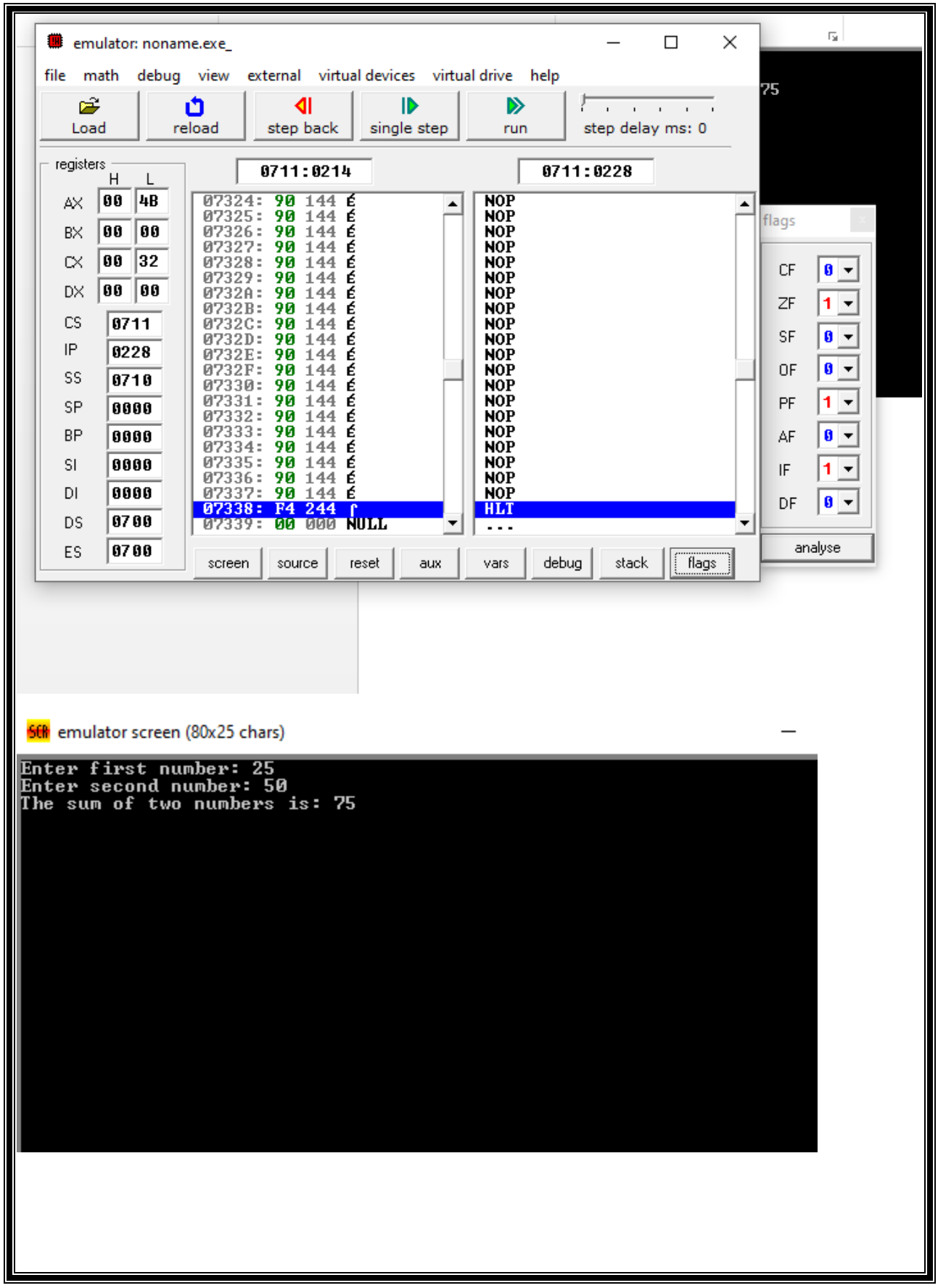


Exercise #3.1:

```
include emu8086.inc

.data
num1 dw 0
num2 dw 0

.code
main proc
    ; Get first number:
    GOTOXY 0, 0     ; Move cursor to top-left corner
    PRINT "Enter first number: "
    call scan_num
    mov num1, cx

    ; Get second number:
    GOTOXY 0, 1     ; Move cursor to second line
    PRINT "Enter second number: "
    call scan_num
    mov num2, cx
```

```
    ; Add the numbers:
    mov ax, num1
    add ax, num2

    ; Print the formatted output:
    GOTOXY 0, 2      ; Move cursor to third line
    PRINT "The sum of two numbers is: "
    call print_num

,
main endp

DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS
DEFINE_SCAN_NUM
END main
```


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help
01 include emu8086.inc
02
03 .data
04 num1 dw 0
05 num2 dw 0
06
07 .code
08 main proc
09 ; Get first number:
10 GOTOXY 0, 0 ; Move cursor to top-left corner
11 PRINT "Enter first number: "
12 call scan_num
13 mov num1, cx
14
15 ; Get second number:
16 GOTOXY 0, 1 ; Move cursor to second line
17 PRINT "Enter second number: "
18 call scan_num
19 mov num2, cx
20
21 ; Add the numbers:
22 mov ax, num1
23 add ax, num2
24
25 ; Print the formatted output:
26 GOTOXY 0, 2 ; Move cursor to third line
27 PRINT "The sum of two numbers is: "
28 call print_num
29
30
31 main endp
32
33 DEFINE_PRINT_NUM
34 DEFINE_PRINT_NUM_UNS
35 DEFINE_SCAN_NUM
36 END main


emulator: noname.exe_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 4B
BX 00 00
CX 00 32
DX 00 00
CS 0711
IP 0228
SS 0710
SP 0000
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0711:0214
0711:0228
07324: 90 144 É NOP
07325: 90 144 É NOP
07326: 90 144 É NOP
07327: 90 144 É NOP
07328: 90 144 É NOP
07329: 90 144 É NOP
0732A: 90 144 É NOP
0732B: 90 144 É NOP
0732C: 90 144 É NOP
0732D: 90 144 É NOP
0732E: 90 144 É NOP
0732F: 90 144 É NOP
07330: 90 144 É NOP
07331: 90 144 É NOP
07332: 90 144 É NOP
07333: 90 144 É NOP
07334: 90 144 É NOP
07335: 90 144 É NOP
07336: 90 144 É NOP
07337: 90 144 É NOP
07338: F4 244 ⌠ HLT
07339: 00 000 NULL ...
screen source reset aux vars debug stack flags
flags
CF 0
ZF 1
SF 0
OF 0
PF 1
AF 0
IF 1
DF 0
analyse
75
emulator screen (80x25 chars)
Enter first number: 25
Enter second number: 50
The sum of two numbers is: 75

Exercise#3.2

Exercise 3.2:

Write a program that reads five unsigned integers from user and display sum of those five integers on the screen.

Code:

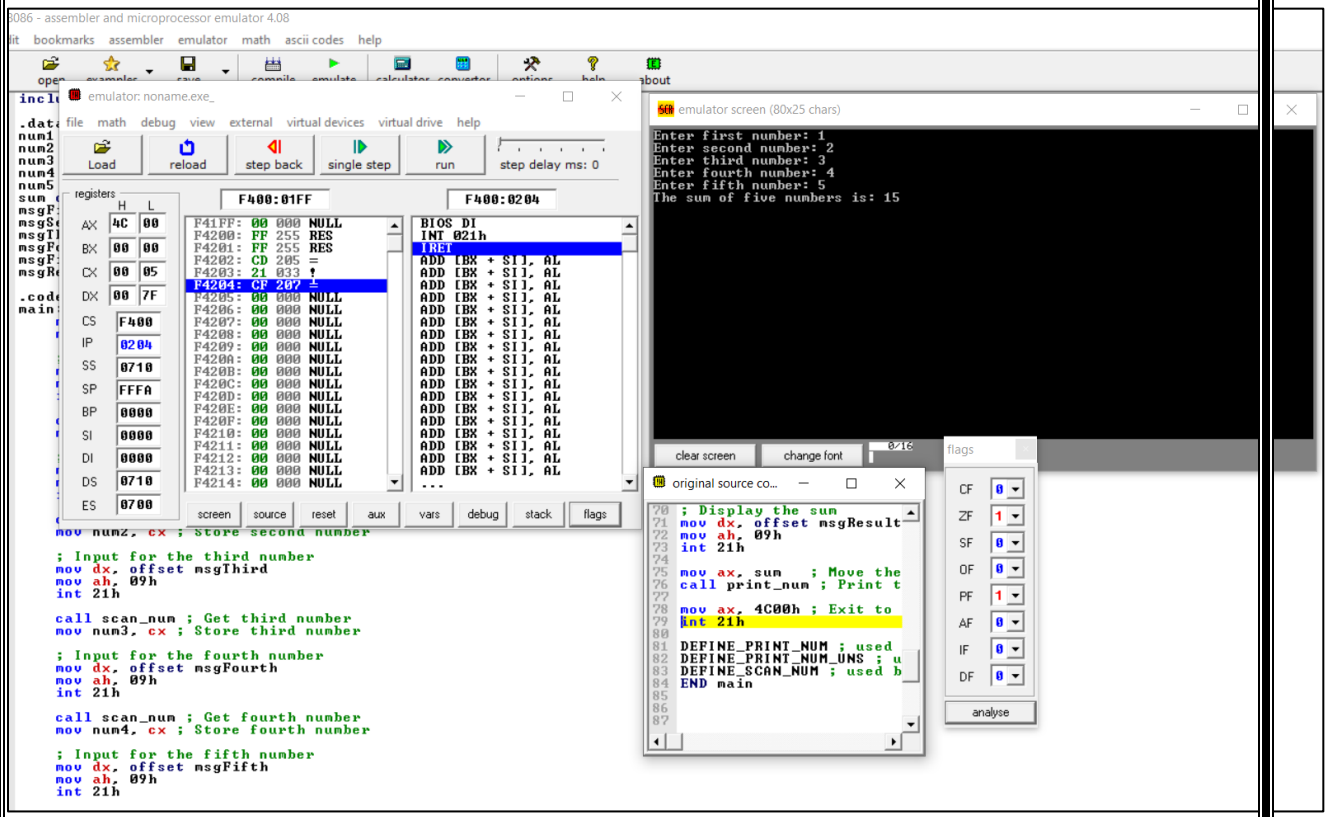
```
include emu8086.inc ;include library

.data
num1 dw 0   ; Define variables for each number
num2 dw 0
num3 dw 0
num4 dw 0
num5 dw 0
sum dw 0    ; Variable to store the sum
msgFirst db "Enter first number: $"
msgSecond db 10, 13, "Enter second number: $"
msgThird db 10, 13, "Enter third number: $"
msgFourth db 10, 13, "Enter fourth number: $"
msgFifth db 10, 13, "Enter fifth number: $"
msgResult db 10, 13, "The sum of five numbers is: $"
```

```
.code
main:
    mov ax, @data
    mov ds, ax

    ; Input for the first number
    mov dx, offset msgFirst
    mov ah, 09h
    int 21h

    call scan_num ; Get first number
    mov num1, cx ; Store first number

    ; Input for the second number
    mov dx, offset msgSecond
    mov ah, 09h
    int 21h

    call scan_num ; Get second number
    mov num2, cx ; Store second number

    ; Input for the third number
    mov dx, offset msgThird
    mov ah, 09h
    int 21h

    call scan_num ; Get third number
    mov num3, cx ; Store third number
```

```
; Input for the fourth number
mov dx, offset msgFourth
mov ah, 09h
int 21h

call scan_num ; Get fourth number
mov num4, cx ; Store fourth number

; Input for the fifth number
mov dx, offset msgFifth
mov ah, 09h
int 21h

call scan_num ; Get fifth number
mov num5, cx ; Store fifth number

; Calculate the sum
mov ax, num1 ; Move the first number to ax
add ax, num2 ; Add the second number
add ax, num3 ; Add the third number
add ax, num4 ; Add the fourth number
add ax, num5 ; Add the fifth number
mov sum, ax  ; Store the final sum

; Display the sum
mov dx, offset msgResult
mov ah, 09h
int 21h
```

```
    mov ax, sum   ; Move the sum to ax for printing
    call print_num ; Print the sum

    mov ax, 4C00h ; Exit to DOS
    int 21h

DEFINE_PRINT_NUM ; used by print_num proc
DEFINE_PRINT_NUM_UNS ; used by print_num proc
DEFINE_SCAN_NUM ; used by scan_num proc
END main
```

Output:



Exercise 3.3:

Write a program using instruction shown in table and fill in the table with the offsets of the instructions in the code segment.

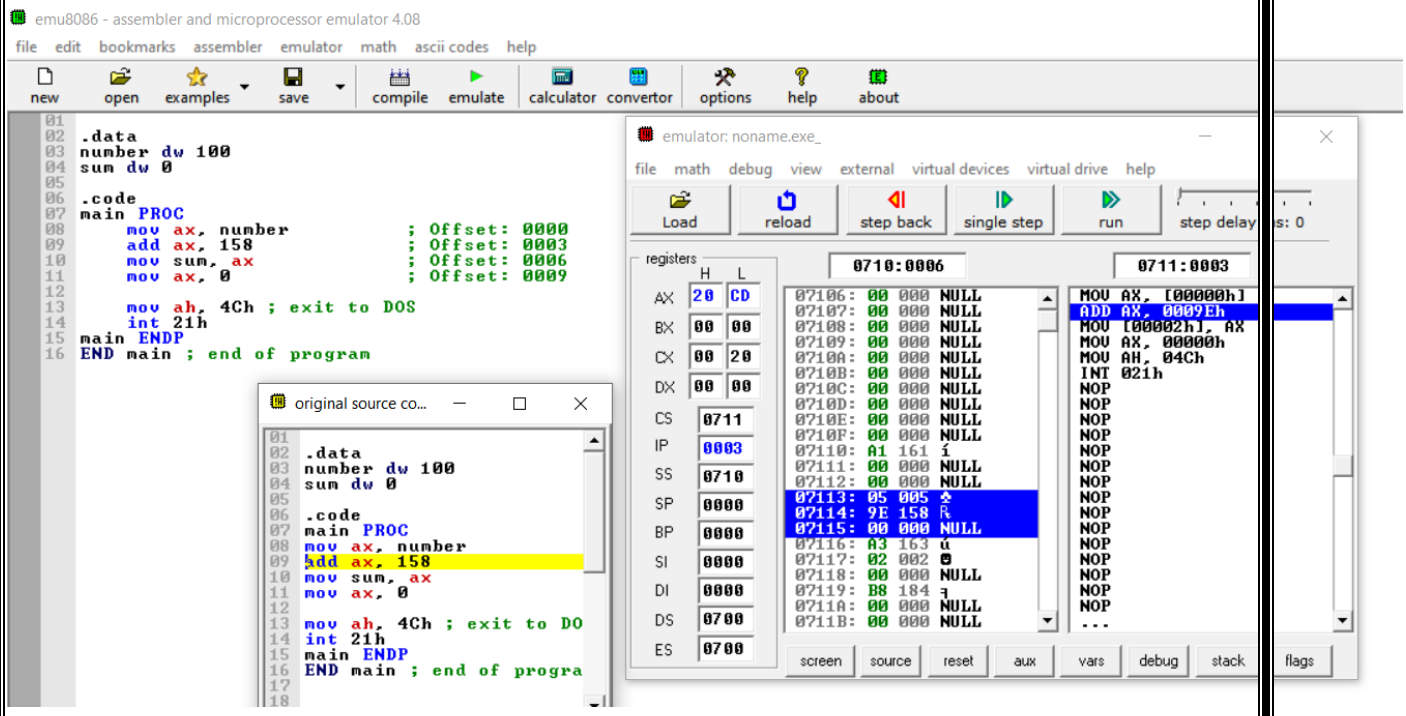
Code:

```
.data
number dw 100
sum dw 0

.code
main PROC
   mov ax, number
   add ax, 158
   mov sum, ax
   mov ax, 0

   mov ah, 4Ch ; exit to DOS
   int 21h
main ENDP
END main ; end of program
```

Output:

Instruction	Segment and Offset
mov ax, number	0000
add ax, 158	0003

mov sum, ax	0006
mov ax, 0	0009



“lab 11”

COURSE :

CALD

SUBMITTED TO :

Sir shoaib

SUBMITTED BY :

Rabia Batool (2022-BSE-064)

SECTION :

B

Activity

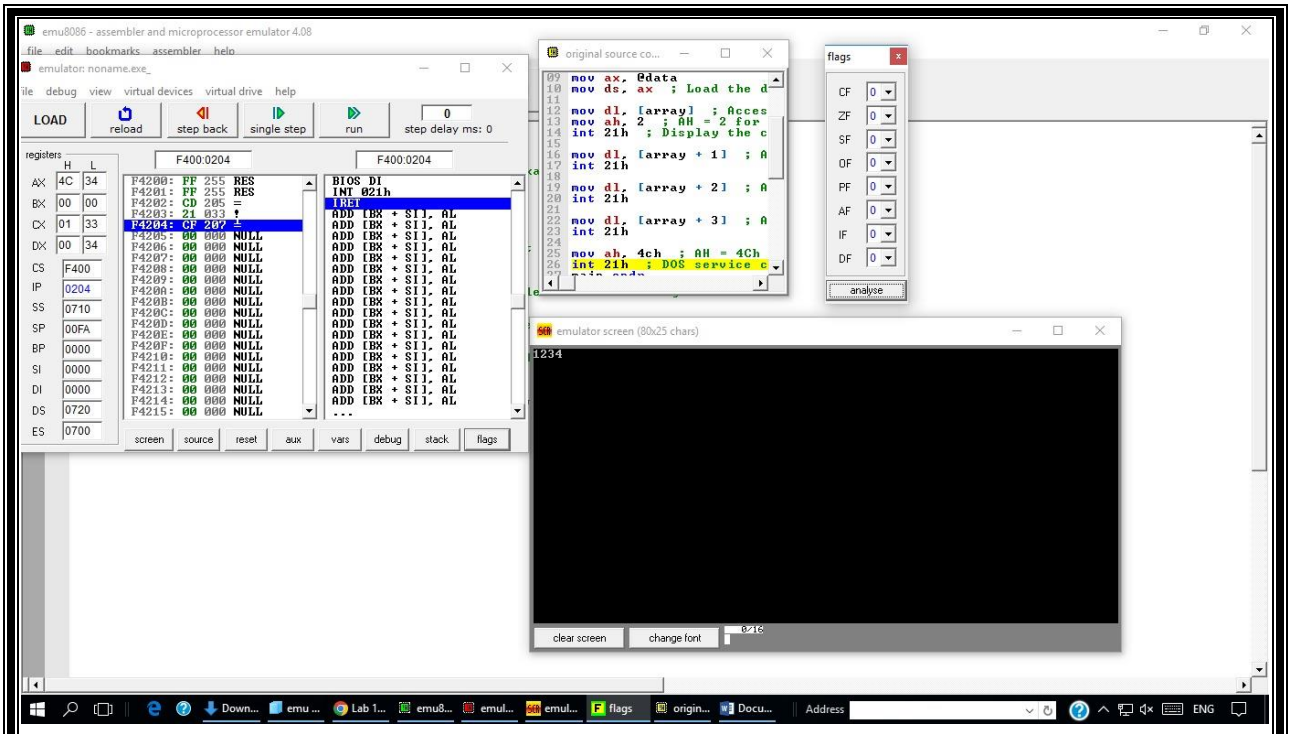
Activity 4.1

Define and Display 8-bit Numbers. Write a program to do the following:

- Use the DB directive to define the following list of numbers and name it array.
  - 31h, 32h, 33h, 34h
- Use the int 21h to display the array.

Code

```
.model small
.stack 100h
.data
array db 31h, 32h, 33h, 34h  ; Define the array of 4 hexadecimal numbers
.code
main proc
    mov ax, @data
    mov ds, ax  ; Load the data segment
    mov dl, [array]  ; Access and load the first element of the array
    mov ah, 2  ; AH = 2 for displaying a character
    int 21h  ; Display the character
    mov dl, [array + 1]  ; Access and load the second element of the array
    int 21h
    mov dl, [array + 2]  ; Access and load the third element of the array
    int 21h
    mov dl, [array + 3]  ; Access and load the fourth element of the array
    int 21h
    mov ah, 4ch  ; AH = 4Ch for terminating the program
    int 21h  ; DOS service call to terminate the program
main endp
end main
```



Exercise

Exercise 4.1

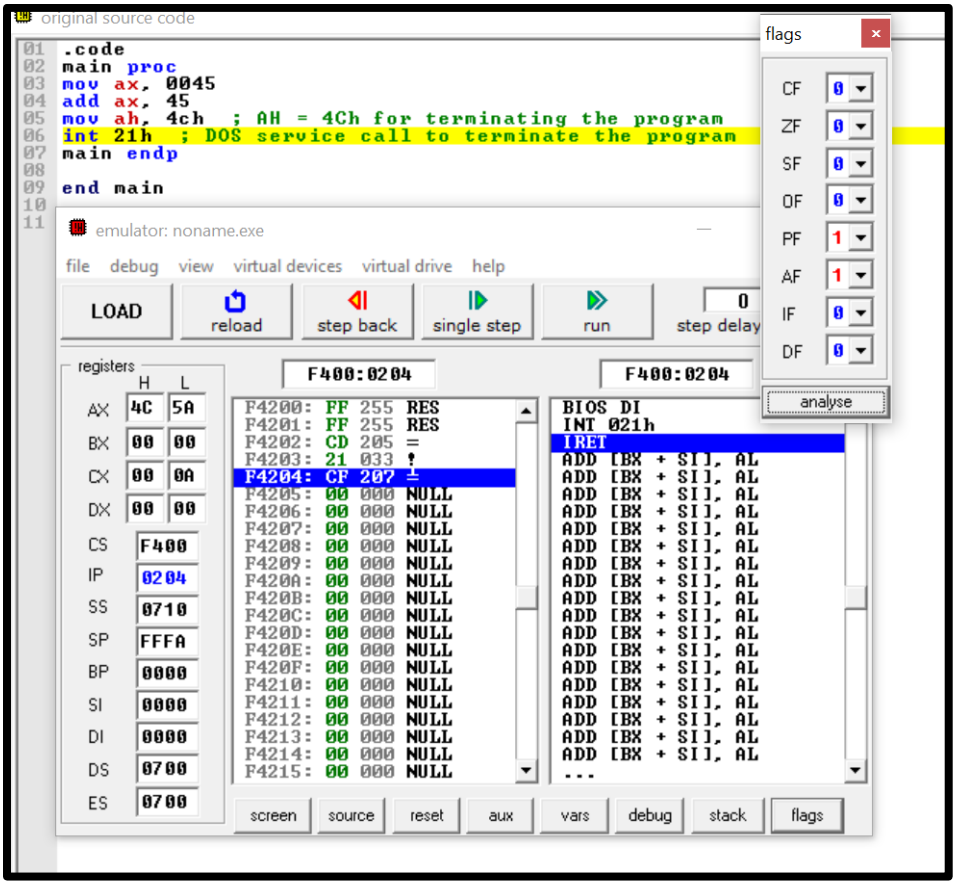
For each add instruction in this exercise, assume that AX contains the given contents before the instruction is executed. Give the contents of AX as well as the values of the CF, OF, SF, PF, AF and ZF after the instruction is executed. All numbers are in hex. (Hint: add ax, 45 adds 45 to the contents of register ax and stores the result back in ax)

Contents of AX (Before)	Instruction	Contents of AX (After)	CF	OF	SF	PF	AF	ZF
0045	add ax, 45							
FF45	add ax, 45							
0045	add ax, -45							
FF45	add ax, -45							
FFFF	add ax, 1							

Code for add ax, 45:

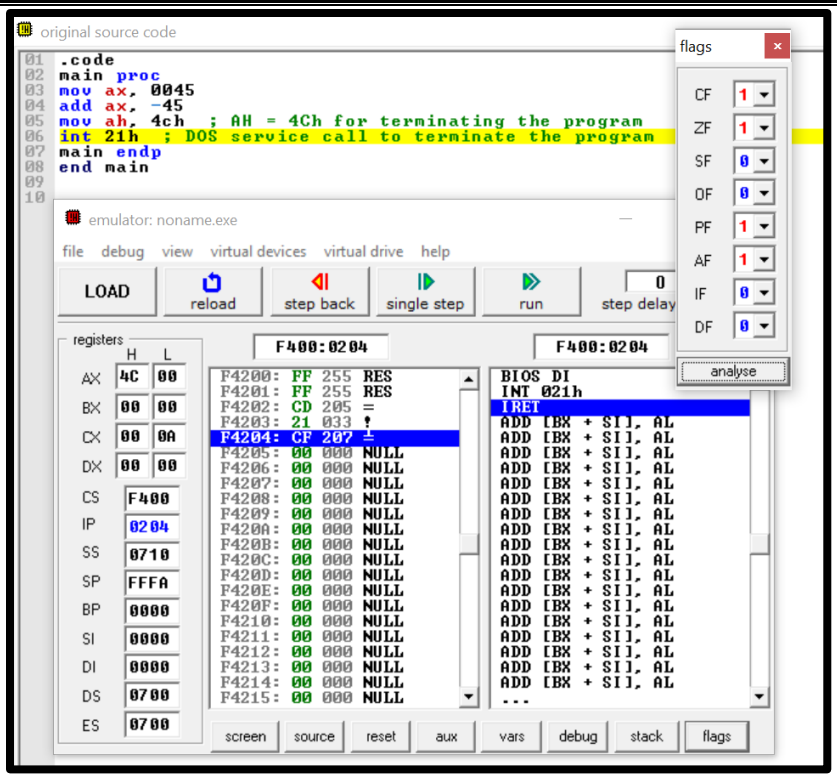
```
.code
main proc
    mov ax, 0045
    add ax, 45
```

```
    mov ah, 4ch  ; AH = 4Ch for terminating the program
    int 21h  ; DOS service call to terminate the program
main endp
end main
```



Code for add ax, -45:

```
.code
main proc
    mov ax, 0045
    add ax, -45
    mov ah, 4ch  ; AH = 4Ch for terminating the program
    int 21h  ; DOS service call to terminate the program
main endp
end main
```



PROGRAM-03

PROGRAM-04

Exercise 4.4

Define 5 different 8-bit Numbers. Write a programme to do the following :

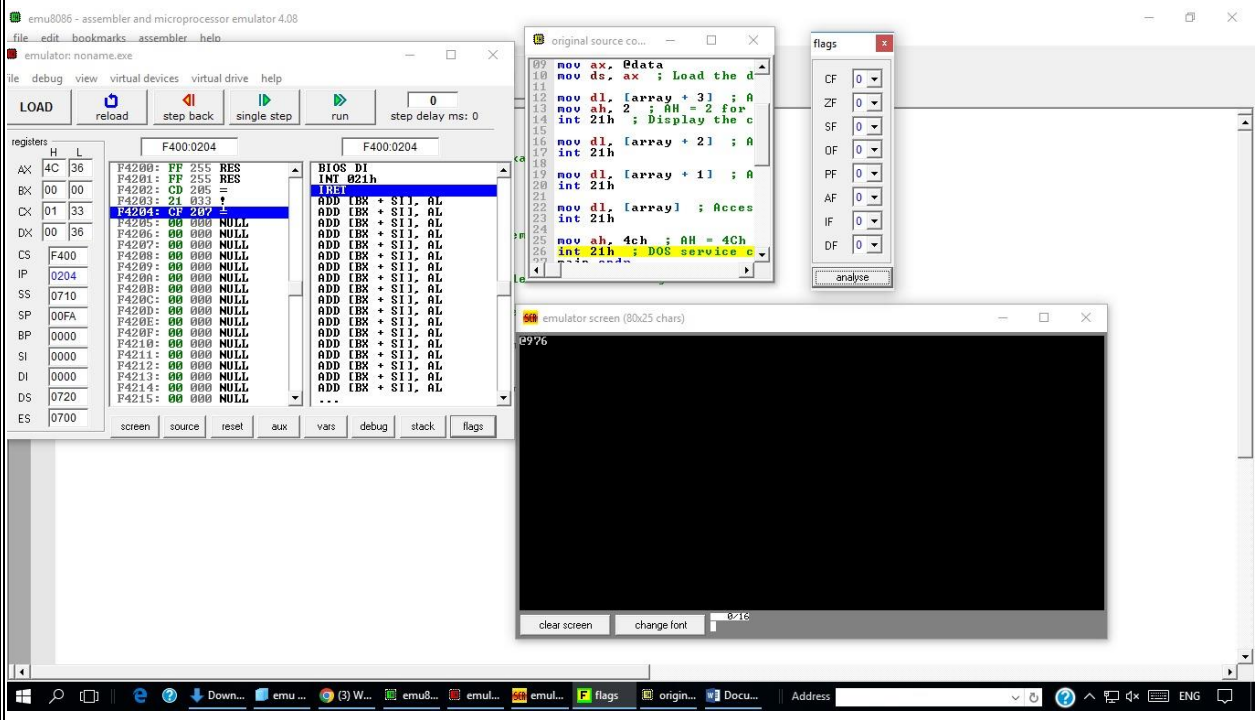
Use the DB directive to define the following list of numbers and name it myarray.

- 36h, 37h, 39h, 40h
- Display the array in reverse order.

Code:

```
.model small
.stack 100h
.data
array db 36h, 37h, 39h, 40h  ; Define the array of 4 hexadecimal numbers
.code
main proc
   mov ax, @data
   mov ds, ax  ; Load the data segment
   mov dl, [array + 3]  ; Access and load the first element of the array
   mov ah, 2  ; AH = 2 for displaying a character
   int 21h  ; Display the character
   mov dl, [array + 2]  ; Access and load the second element of the array
   int 21h
   mov dl, [array + 1]  ; Access and load the third element of the array
   int 21h
   mov dl, [array]  ; Access and load the fourth element of the array
   int 21h
   mov ah, 4ch  ; AH = 4Ch for terminating the program
   int 21h  ; DOS service call to terminate the program
main endp
```

end main



Exercise 4.2

Calculate the range of signed and unsigned numbers that db, dw, dd, dq and d directives can specify. Mention the adopted method i.e how you will calculate th range.

db (define byte):

- **Signed Range:** -128 to 127
- **Unsigned Range:** 0 to 255
- **Calculation method**: For signed, it's -2^(n-1) to 2^(n-1)-1, and for unsigned, it's 0 to 2^n-1, where n is the number of bits (8 bits in this case).

dw (define word):

- **Signed Range:** -32,768 to 32,767
- **Unsigned Range:** 0 to 65,535
- **Calculation method:** Similar to db, but with 16 bits.

dd (define doubleword):

- **Signed Range:** -2,147,483,648 to 2,147,483,647
- **Unsigned Range:** 0 to 4,294,967,295

- **Calculation method:** Similar to db but with 32 bits.

dq (define quadword):

- **Signed Range:** -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
- **Unsigned Range:** 0 to 18,446,744,073,709,551,615
- **Calculation method:** Similar to db but with 64 bits.

dt (define ten-byte):

- **Signed Range:** Approximately -1.8 x 10^308 to 1.8 x 10^308
- **Unsigned Range:** 0 to approximately 3.6 x 10^308
- **Calculation method:** The dt directive is not as standard as the others, and the range can vary depending on the floating-point format used. In general, it's designed to handle floating-point values with higher precision.

Exercise 4.3

Find the initial values that the assembler will generate for each of the directives below.

byte1 db 10110111b _____

byte2 db 35q _____

byte3 db 0E8h _____

byte4 db 150 _____

byte5 db -91 _____

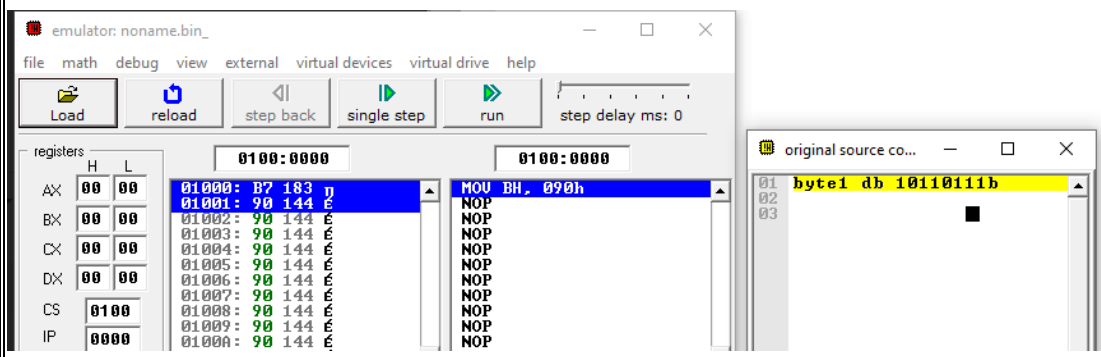
byte6 db 'K' _____

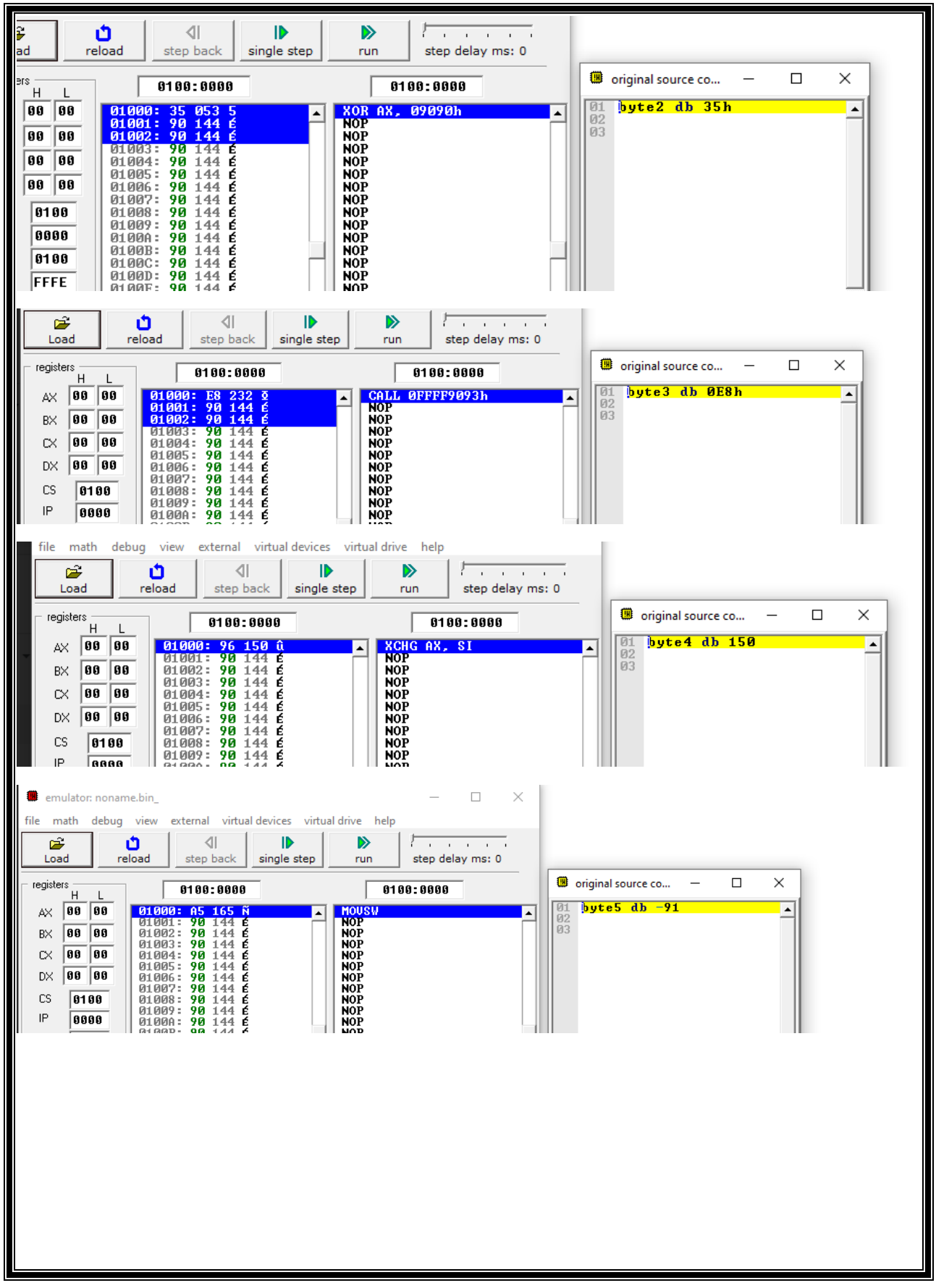
byte7 db 'k' _____

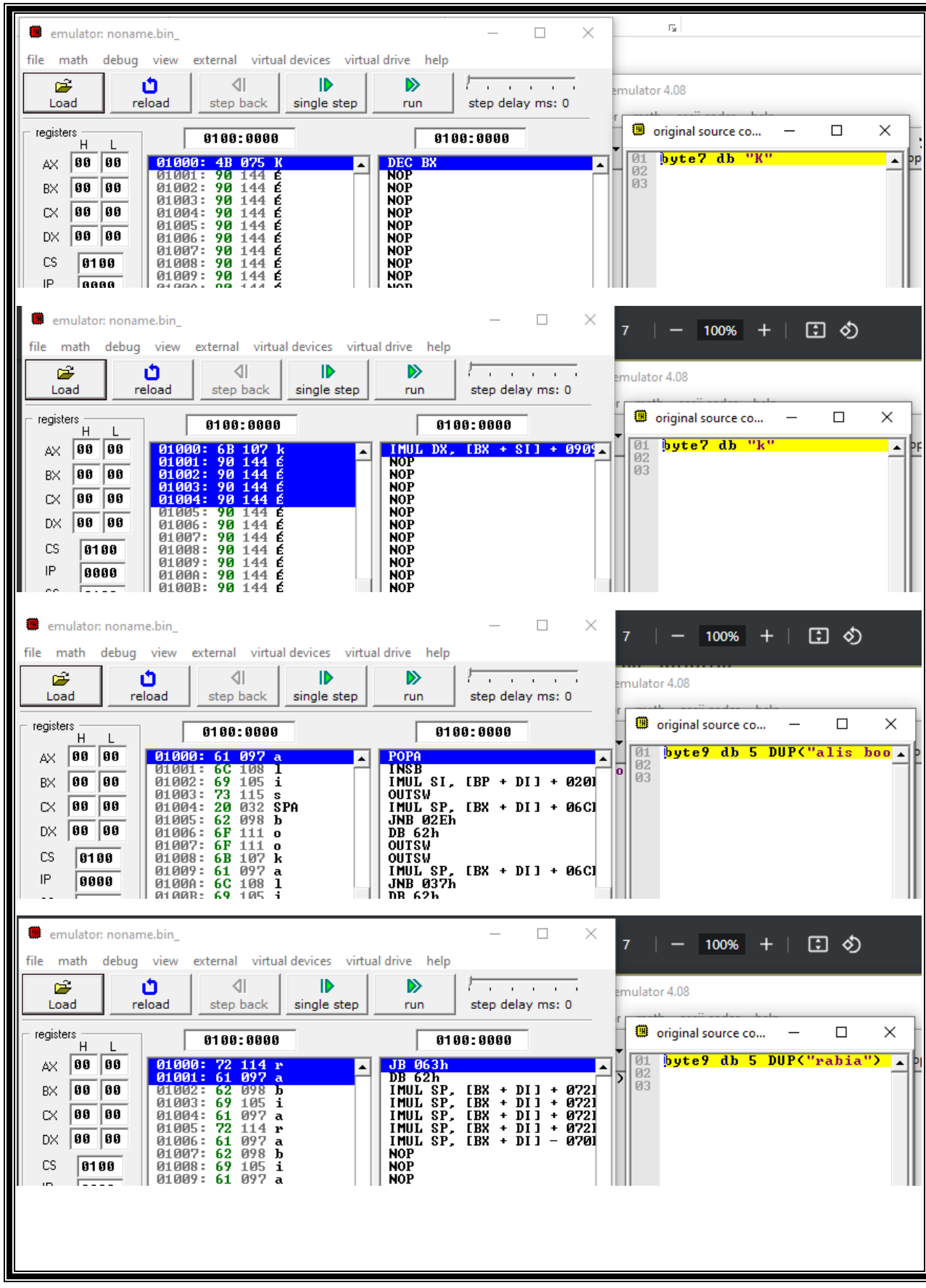
byte8 db "Ali's book" _____

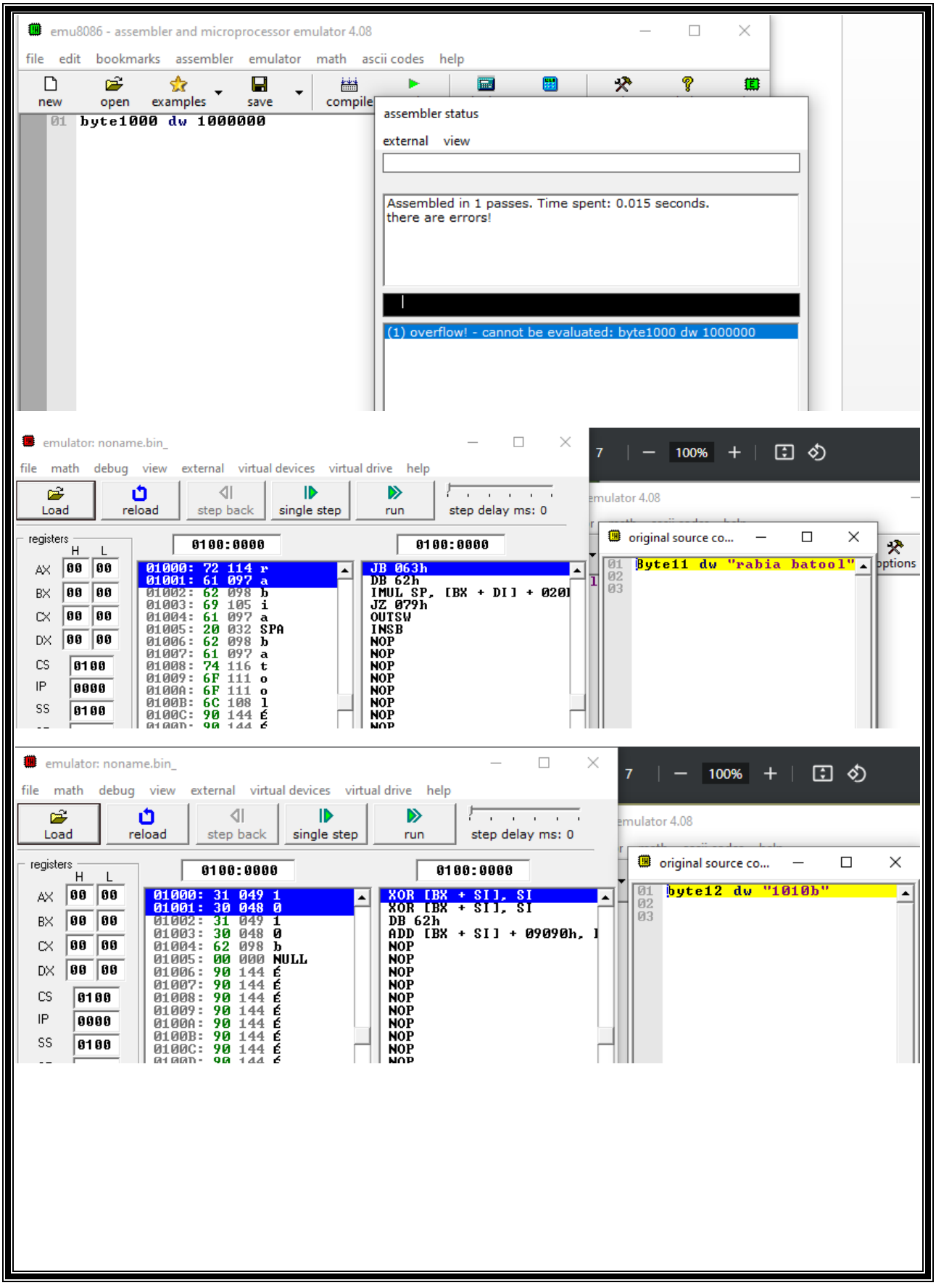
byte9 db 5 DUP("< >") _____

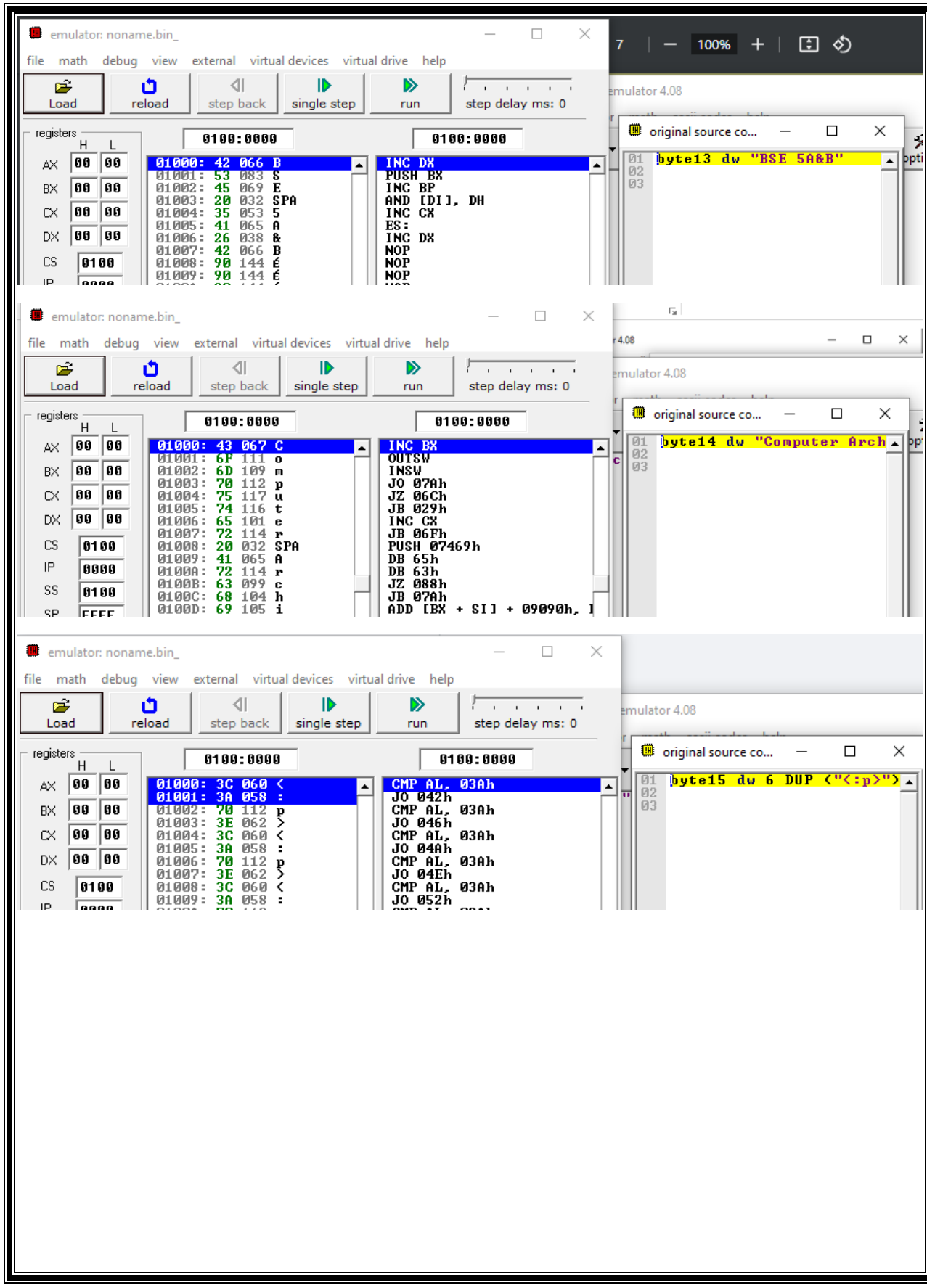
byte10 dw 1000000 _____




reload
step back
single step
run
step delay ms: 0
0100:0000
0100:0000
01000: 35 053 5
01001: 90 144 É
01002: 90 144 É
XOR AX, 09090h
NOP
original source co...
byte2 db 35h
Load
CALL 0FFFF9093h
byte3 db 0E8h
file math debug view external virtual devices virtual drive help
registers
AX
BX
CX
DX
CS
IP
01000: 96 150 û
XCHG AX, SI
byte4 db 150
emulator: noname.bin_
01000: A5 165 Ñ
MOVSW
byte5 db -91


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 00
BX 00 00
CX 00 00
DX 00 00
CS 0100
IP 0000
0100:0000
0100:0000
01000: 4B 075 K
01001: 90 144 É
01002: 90 144 É
01003: 90 144 É
01004: 90 144 É
01005: 90 144 É
01006: 90 144 É
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
DEC BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
original source co...
01 byte7 db "K"
02
03
emulator 4.08
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 00
BX 00 00
CX 00 00
DX 00 00
CS 0100
IP 0000
0100:0000
0100:0000
01000: 6B 107 k
01001: 90 144 É
01002: 90 144 É
01003: 90 144 É
01004: 90 144 É
01005: 90 144 É
01006: 90 144 É
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
IMUL DX, [BX + SI] + 0909
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
original source co...
01 byte7 db "k"
02
03
7
100%
emulator 4.08
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 00
BX 00 00
CX 00 00
DX 00 00
CS 0100
IP 0000
0100:0000
0100:0000
01000: 61 097 a
01001: 6C 108 l
01002: 69 105 i
01003: 73 115 s
01004: 20 032 SPA
01005: 62 098 b
01006: 6F 111 o
01007: 6F 111 o
01008: 6B 107 k
01009: 61 097 a
0100A: 6C 108 l
POPA
INSB
IMUL SI, [BP + DI] + 020I
OUTSW
IMUL SP, [BX + DI] + 06CI
JNB 02Eh
DB 62h
OUTSW
OUTSW
IMUL SP, [BX + DI] + 06CI
JNB 037h
original source co...
01 byte9 db 5 DUP("alis boo
02
03
7
100%
emulator 4.08
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 00
BX 00 00
CX 00 00
DX 00 00
CS 0100
0100:0000
0100:0000
01000: 72 114 r
01001: 61 097 a
01002: 62 098 b
01003: 69 105 i
01004: 61 097 a
01005: 72 114 r
01006: 61 097 a
01007: 62 098 b
01008: 69 105 i
01009: 61 097 a
JB 063h
DB 62h
IMUL SP, [BX + DI] + 072I
IMUL SP, [BX + DI] + 072I
IMUL SP, [BX + DI] + 072I
IMUL SP, [BX + DI] + 072I
IMUL SP, [BX + DI] - 070I
NOP
NOP
NOP
original source co...
01 byte9 db 5 DUP("rabia")
02
03
7
100%
emulator 4.08


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile
01 byte1000 dw 1000000
assembler status
external view
Assembled in 1 passes. Time spent: 0.015 seconds.
there are errors!
(1) overflow! - cannot be evaluated: byte1000 dw 1000000
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
H L
AX 00 00
BX 00 00
CX 00 00
DX 00 00
CS 0100
IP 0000
SS 0100
0100:0000
0100:0000
01000: 72 114 r
01001: 61 097 a
01002: 62 098 b
01003: 69 105 i
01004: 61 097 a
01005: 20 032 SPA
01006: 62 098 b
01007: 61 097 a
01008: 74 116 t
01009: 6F 111 o
0100A: 6F 111 o
0100B: 6C 108 l
0100C: 90 144 É
JB 063h
DB 62h
IMUL SP, [BX + DI + 020]
JZ 079h
OUTSW
INSB
NOP
NOP
NOP
NOP
NOP
NOP
NOP
100%
emulator 4.08
original source co...
01 Byte11 dw "rabia batool"
02
03
options
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
H L
AX 00 00
BX 00 00
CX 00 00
DX 00 00
CS 0100
IP 0000
SS 0100
0100:0000
0100:0000
01000: 31 049 1
01001: 30 048 0
01002: 31 049 1
01003: 30 048 0
01004: 62 098 b
01005: 00 000 NULL
01006: 90 144 É
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
XOR [BX + SI], SI
XOR [BX + SI], SI
DB 62h
ADD [BX + SI] + 09090h, I
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
100%
emulator 4.08
original source co...
01 byte12 dw "1010b"
02
03


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 00
BX 00 00
CX 00 00
DX 00 00
CS 0100
IP 0000
SS 0100
SP FFFF
0100:0000
0100:0000
01000: 42 066 B
01001: 53 083 S
01002: 45 069 E
01003: 20 032 SPA
01004: 35 053 5
01005: 41 065 A
01006: 26 038 &
01007: 42 066 B
01008: 90 144 É
01009: 90 144 É
INC DX
PUSH BX
INC BP
AND [DI], DH
INC CX
ES:
INC DX
NOP
NOP
NOP
original source co...
byte13 dw "BSE 5A&B"
emulator 4.08
01000: 43 067 C
01001: 6F 111 o
01002: 6D 109 m
01003: 70 112 p
01004: 75 117 u
01005: 74 116 t
01006: 65 101 e
01007: 72 114 r
01008: 20 032 SPA
01009: 41 065 A
0100A: 72 114 r
0100B: 63 099 c
0100C: 68 104 h
0100D: 69 105 i
INC BX
OUTSW
INSW
JO 07Ah
JZ 06Ch
JB 029h
INC CX
JB 06Fh
PUSH 07469h
DB 65h
DB 63h
JZ 088h
JB 07Ah
ADD [BX + SI] + 09090h, I
byte14 dw "Computer Arch
01000: 3C 060 <
01001: 3A 058 :
01002: 70 112 p
01003: 3E 062 >
01004: 3C 060 <
01005: 3A 058 :
01006: 70 112 p
01007: 3E 062 >
01008: 3C 060 <
01009: 3A 058 :
CMP AL, 03Ah
JO 042h
CMP AL, 03Ah
JO 046h
CMP AL, 03Ah
JO 04Ah
CMP AL, 03Ah
JO 04Eh
CMP AL, 03Ah
JO 052h
byte15 dw 6 DUP ("<:p>")

“lab 13”

COURSE :

CALD

SUBMITTED TO :

Sir shoaib

SUBMITTED BY :

Rabia Batool (2022-BSE-064)

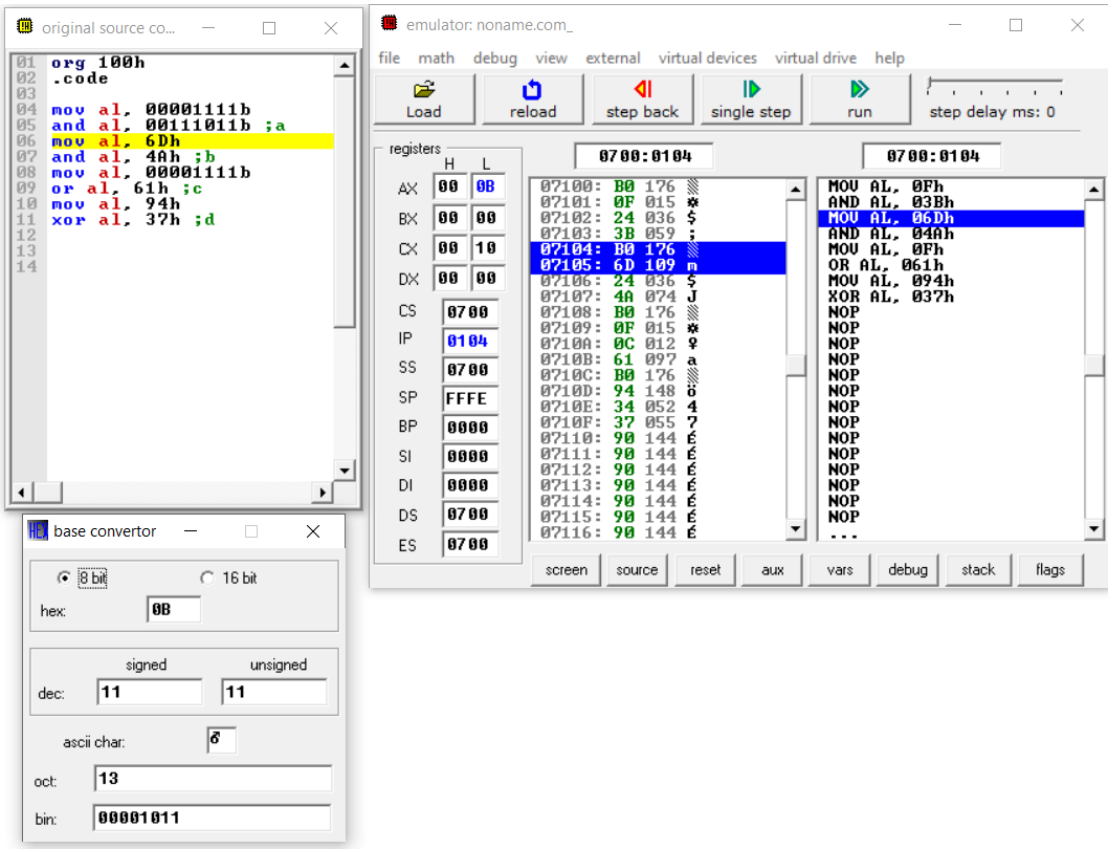
SECTION :

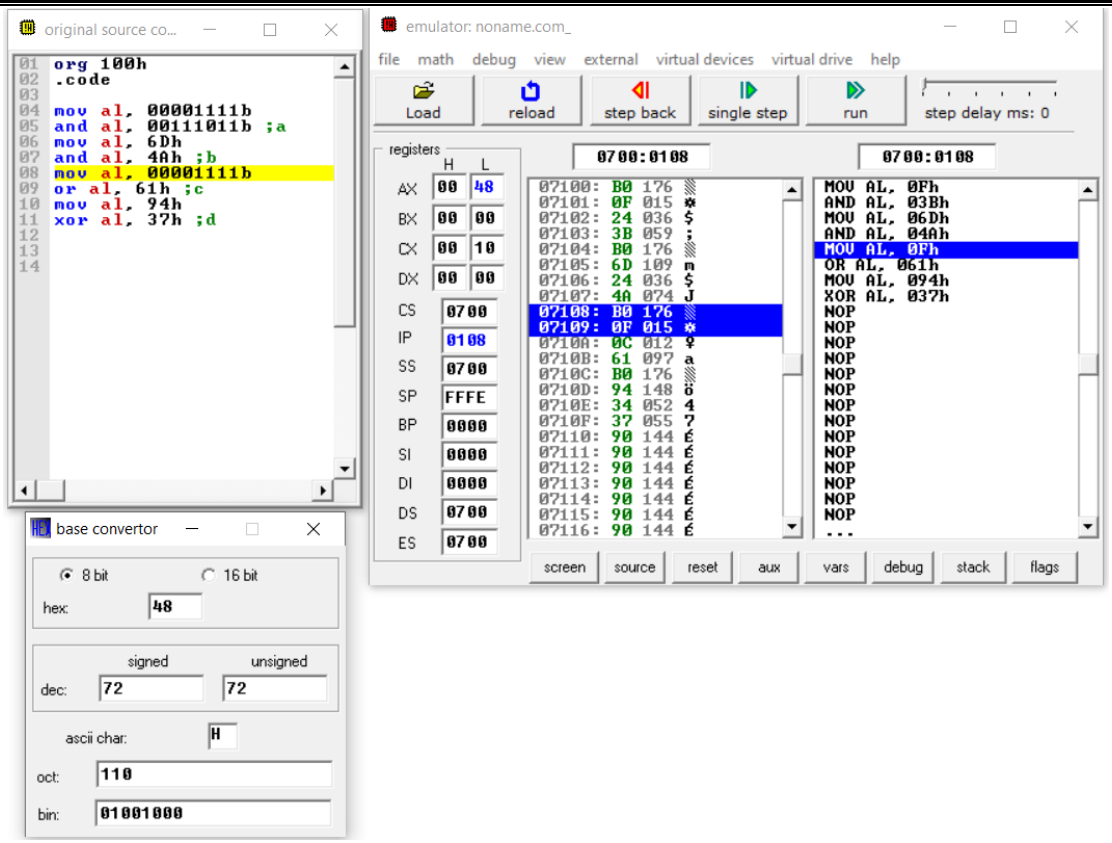
B

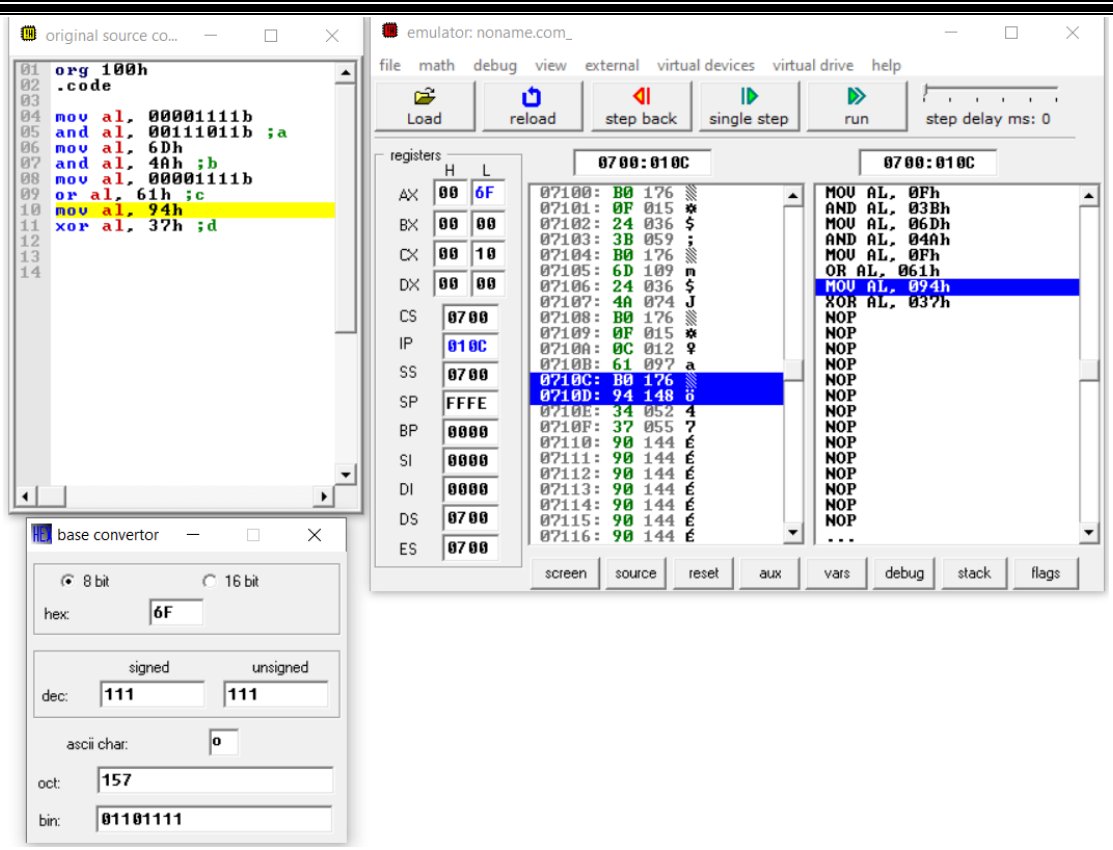
Exercise 7.1:

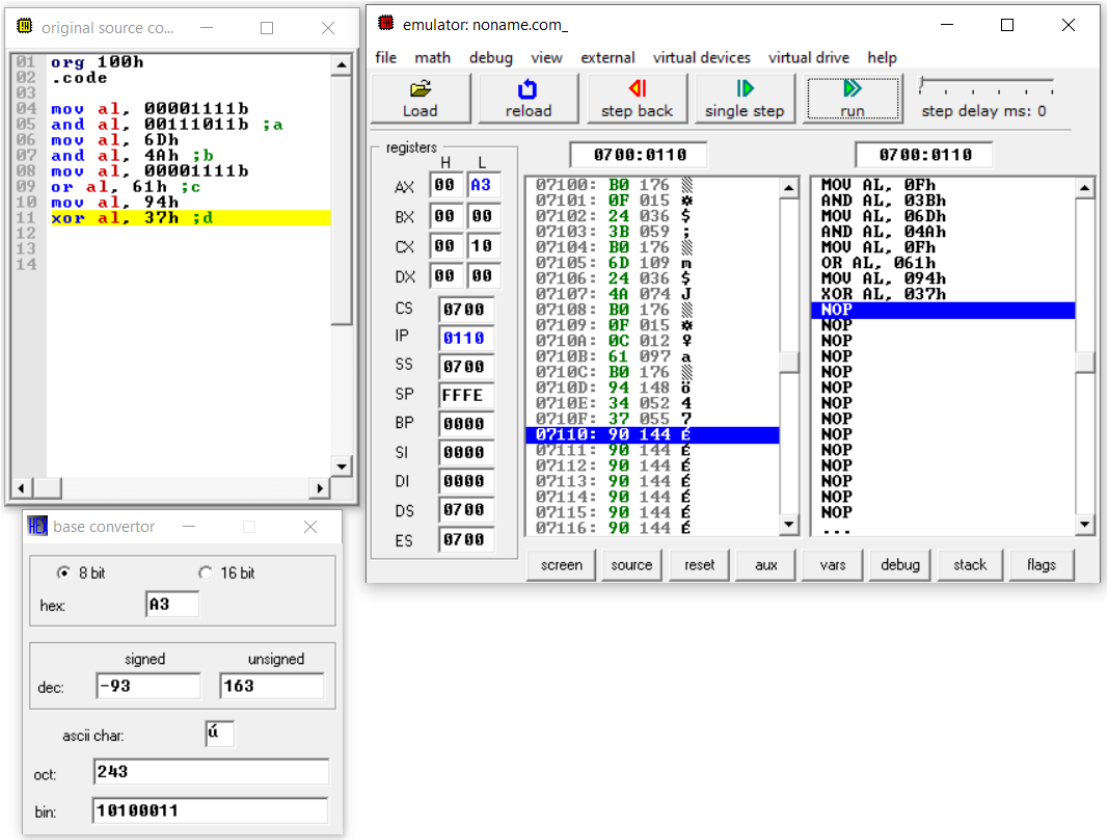
In the following instruction sequence. Show the changed value of AL where indicated,

in binary:




original source co...
01 org 100h
02 .code
03
04 mov al, 00001111b
05 and al, 00111011b ;a
06 mov al, 6Dh
07 and al, 4Ah ;b
08 mov al, 00001111b
09 or al, 61h ;c
10 mov al, 94h
11 xor al, 37h ;d
12
13
14
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 48
BX 00 00
CX 00 10
DX 00 00
CS 0700
IP 0108
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0700:0108
07100: B0 176
07101: 0F 015
07102: 24 036 $
07103: 3B 059 ;
07104: B0 176
07105: 6D 109 m
07106: 24 036 $
07107: 4A 074 J
07108: B0 176
07109: 0F 015
0710A: 0C 012
0710B: 61 097 a
0710C: B0 176
0710D: 94 148
0710E: 34 052 4
0710F: 37 055 7
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
0700:0108
MOV AL, 0Fh
AND AL, 03Bh
MOV AL, 06Dh
AND AL, 04Ah
MOV AL, 0Fh
OR AL, 061h
MOV AL, 094h
XOR AL, 037h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen
source
reset
aux
vars
debug
stack
flags
base convertor
8 bit
16 bit
hex: 48
signed
unsigned
dec: 72 72
ascii char: H
oct: 110
bin: 01001000

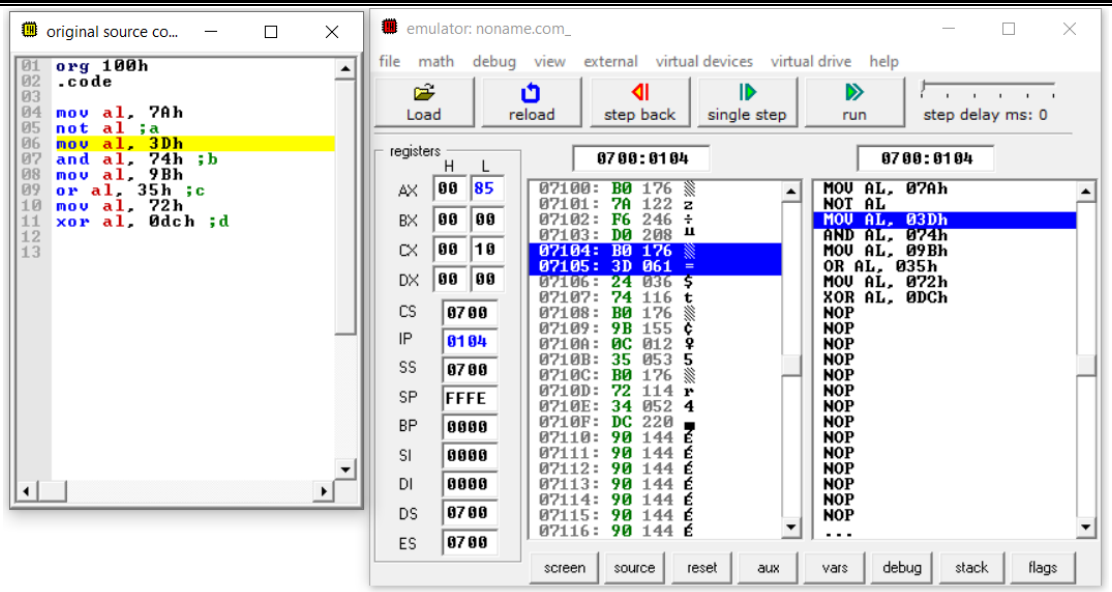

original source co...
01 org 100h
02 .code
03
04 mov al, 00001111b
05 and al, 00111011b ;a
06 mov al, 6Dh
07 and al, 4Ah ;b
08 mov al, 00001111b
09 or al, 61h ;c
10 mov al, 94h
11 xor al, 37h ;d
12
13
14
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 6F
BX 00 00
CX 00 10
DX 00 00
CS 0700
IP 010C
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0700:010C
07100: B0 176
07101: 0F 015
07102: 24 036 $
07103: 3B 059 ;
07104: B0 176
07105: 6D 109 m
07106: 24 036 $
07107: 4A 074 J
07108: B0 176
07109: 0F 015
0710A: 0C 012
0710B: 61 097 a
0710C: B0 176
0710D: 94 148
0710E: 34 052 4
0710F: 37 055 7
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
0700:010C
MOV AL, 0Fh
AND AL, 03Bh
MOV AL, 06Dh
AND AL, 04Ah
MOV AL, 0Fh
OR AL, 061h
MOV AL, 094h
XOR AL, 037h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen
source
reset
aux
vars
debug
stack
flags
base convertor
8 bit
16 bit
hex: 6F
signed
unsigned
dec: 111 111
ascii char: o
oct: 157
bin: 01101111

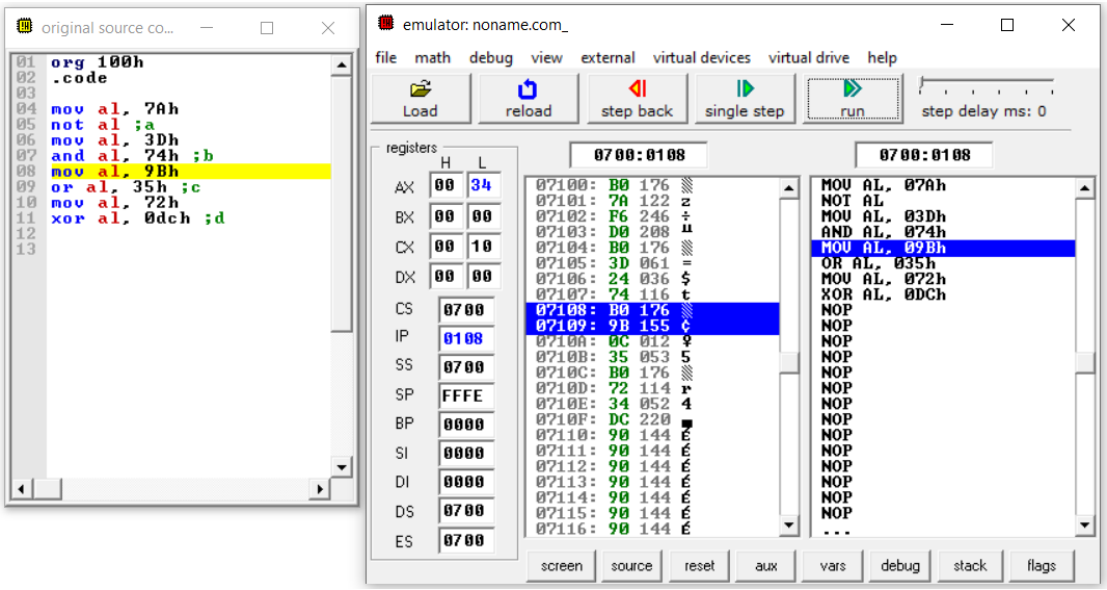


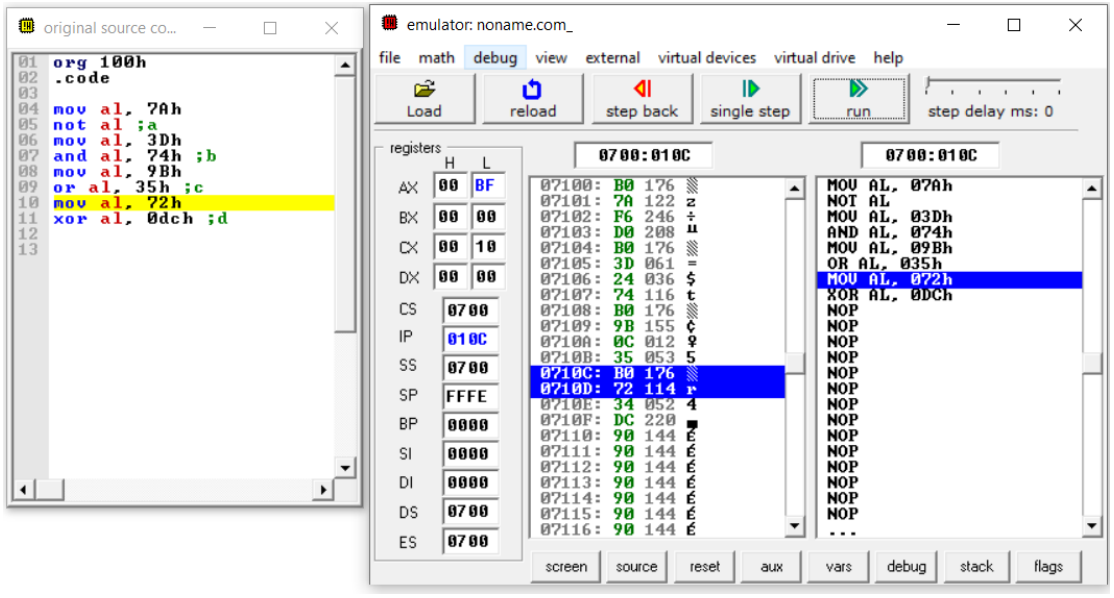
Exercise 7.2:

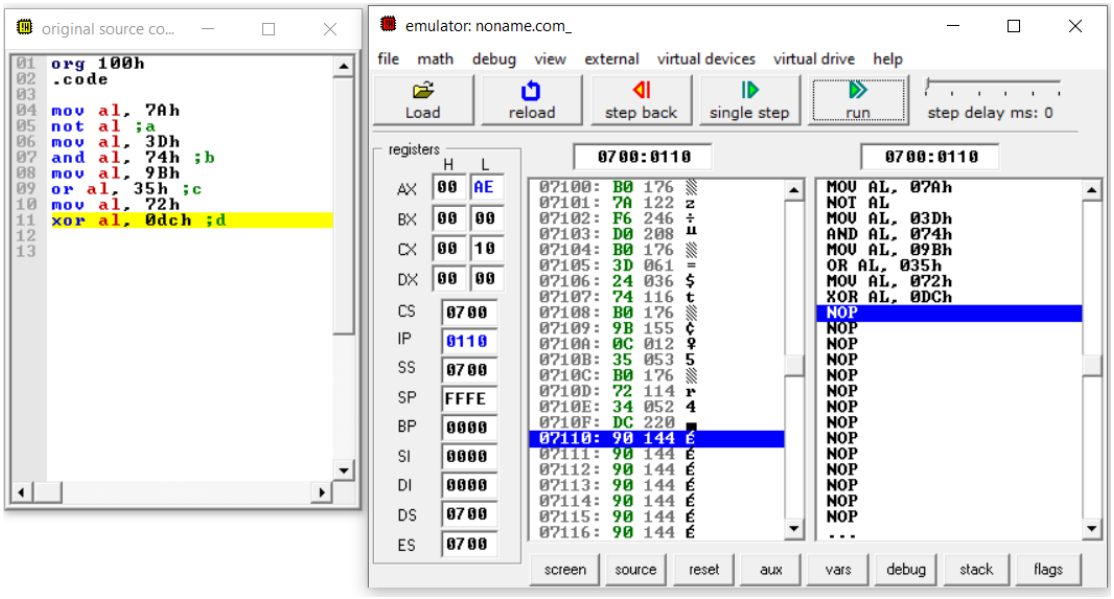
In the following instruction sequence, show the changed value of AL where indicated,

in hexadecimal:


original source co...
01 org 100h
02 .code
03
04 mov al, 7Ah
05 not al ;a
06 mov al, 3Dh
07 and al, 74h ;b
08 mov al, 9Bh
09 or al, 35h ;c
10 mov al, 72h
11 xor al, 0dch ;d
12
13
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 85
BX 00 00
CX 00 10
DX 00 00
CS 0700
IP 0104
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0700:0104
0700:0104
07100: B0 176
07101: 7A 122 z
07102: F6 246 ÷
07103: D0 208 ╨
07104: B0 176
07105: 3D 061 =
07106: 24 036 $
07107: 74 116 t
07108: B0 176
07109: 9B 155 ¢
0710A: 0C 012
0710B: 35 053 5
0710C: B0 176
0710D: 72 114 r
0710E: 34 052 4
0710F: DC 220
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
MOV AL, 07Ah
NOT AL
MOV AL, 03Dh
AND AL, 074h
MOV AL, 09Bh
OR AL, 035h
MOV AL, 072h
XOR AL, 0DCh
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen source reset aux vars debug stack flags

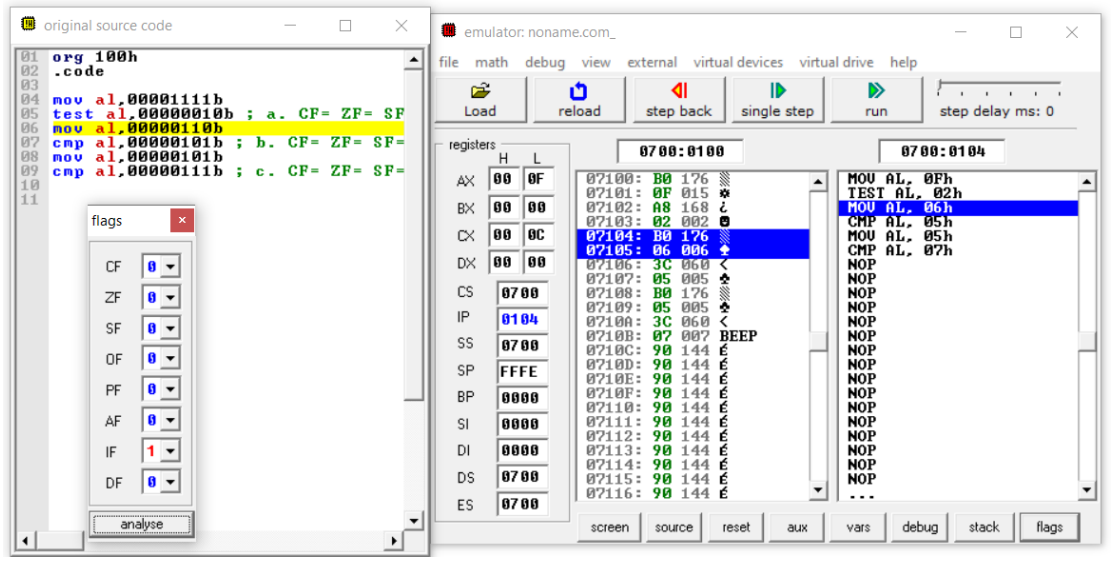

original source co...
01 org 100h
02 .code
03
04 mov al, 7Ah
05 not al ;a
06 mov al, 3Dh
07 and al, 74h ;b
08 mov al, 9Bh
09 or al, 35h ;c
10 mov al, 72h
11 xor al, 0dch ;d
12
13
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 34
BX 00 00
CX 00 10
DX 00 00
CS 0700
IP 0108
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0700:0108
0700:0108
07100: B0 176
07101: 7A 122 z
07102: F6 246 ÷
07103: D0 208 ╨
07104: B0 176
07105: 3D 061 =
07106: 24 036 $
07107: 74 116 t
07108: B0 176
07109: 9B 155 ¢
0710A: 0C 012
0710B: 35 053 5
0710C: B0 176
0710D: 72 114 r
0710E: 34 052 4
0710F: DC 220
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
MOV AL, 07Ah
NOT AL
MOV AL, 03Dh
AND AL, 074h
MOV AL, 09Bh
OR AL, 035h
MOV AL, 072h
XOR AL, 0DCh
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen source reset aux vars debug stack flags

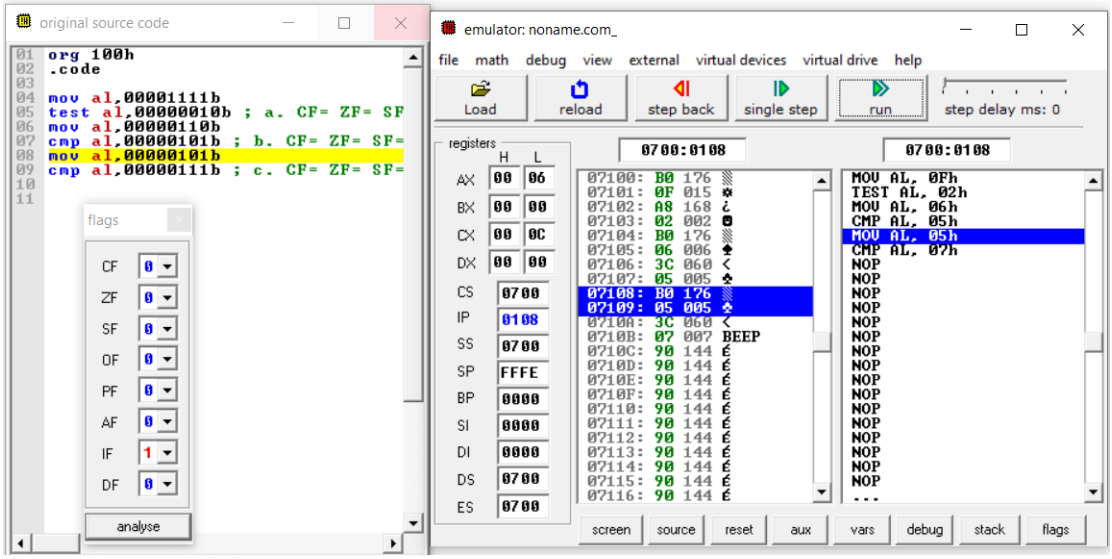


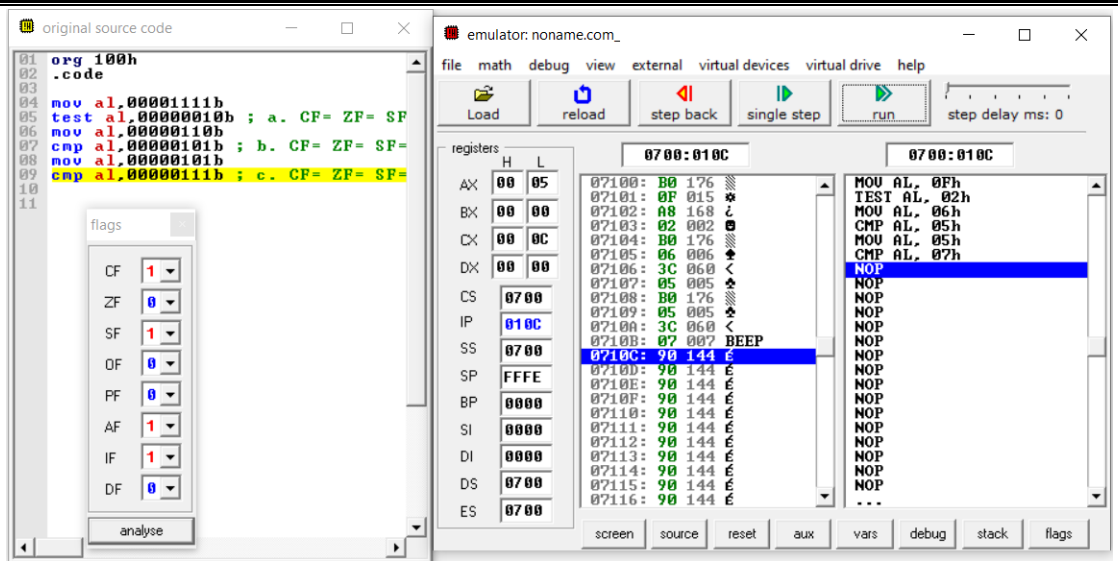


Exercise 7.3:

In the following instruction sequence, show the values of the Carry, Zero, and Sign flags where indicated:

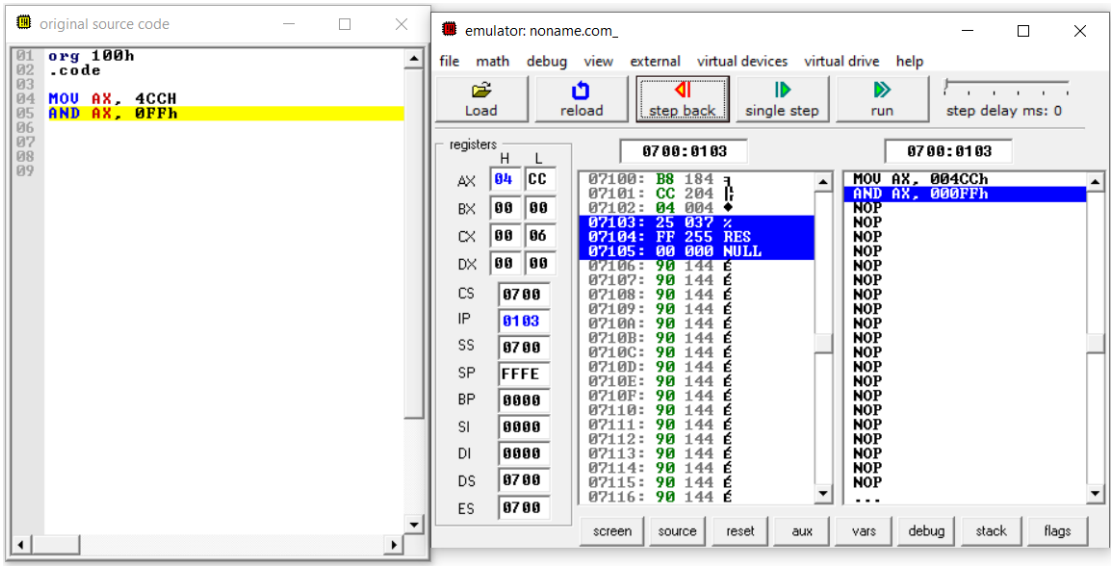

original source code
01 org 100h
02 .code
03
04 mov al,00001111b
05 test al,00000010b ; a. CF= ZF= SF
06 mov al,00000110b
07 cmp al,00000101b ; b. CF= ZF= SF=
08 mov al,00000101b
09 cmp al,00000111b ; c. CF= ZF= SF=
10
11
flags
CF 0
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 0
analyse
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 0F
BX 00 00
CX 00 0C
DX 00 00
CS 0700
IP 0104
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0700:0100
0700:0104
07100: B0 176
07101: 0F 015
07102: A8 168
07103: 02 002
07104: B0 176
07105: 06 006
07106: 3C 060
07107: 05 005
07108: B0 176
07109: 05 005
0710A: 3C 060
0710B: 07 007 BEEP
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
MOV AL, 0Fh
TEST AL, 02h
MOV AL, 06h
CMP AL, 05h
MOV AL, 05h
CMP AL, 07h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen source reset aux vars debug stack flags

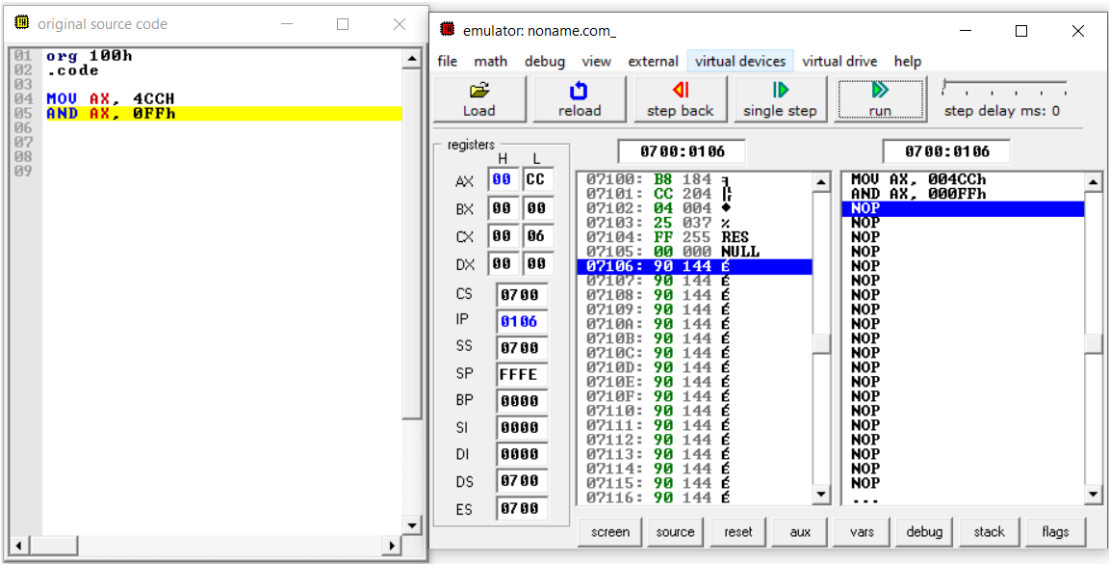

original source code
01 org 100h
02 .code
03
04 mov al,00001111b
05 test al,00000010b ; a. CF= ZF= SF
06 mov al,00000110b
07 cmp al,00000101b ; b. CF= ZF= SF=
08 mov al,00000101b
09 cmp al,00000111b ; c. CF= ZF= SF=
10
11
flags
CF 0
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 0
analyse
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 06
BX 00 00
CX 00 0C
DX 00 00
CS 0700
IP 0108
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0700:0108
0700:0108
07100: B0 176
07101: 0F 015
07102: A8 168
07103: 02 002
07104: B0 176
07105: 06 006
07106: 3C 060
07107: 05 005
07108: B0 176
07109: 05 005
0710A: 3C 060
0710B: 07 007 BEEP
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
MOV AL, 0Fh
TEST AL, 02h
MOV AL, 06h
CMP AL, 05h
MOV AL, 05h
CMP AL, 07h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen source reset aux vars debug stack flags



Exercise 7.4:

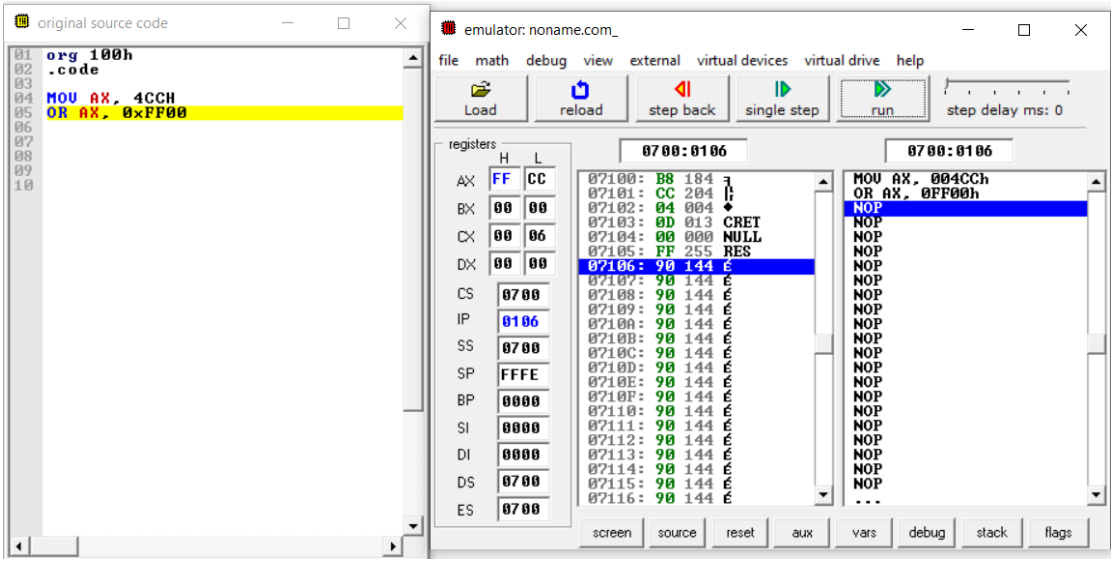
Write a single instruction that clears the high 8 bits of AX and does not change the low 8 bits.





Exercise 7.5:

Write a single instruction that sets the high 8 bits of AX and does not change the low 8 bits.


original source code
01 org 100h
02 .code
03
04 MOV AX, 4CCH
05 OR AX, 0xFF00
06
07
08
09
10
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX FF CC
BX 00 00
CX 00 06
DX 00 00
CS 0700
IP 0106
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0700:0106
07100: B8 184 ¸
07101: CC 204 Ì
07102: 04 004 ♦
07103: 0D 013 CRET
07104: 00 000 NULL
07105: FF 255 RES
07106: 90 144 É
07107: 90 144 É
07108: 90 144 É
07109: 90 144 É
0710A: 90 144 É
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
0700:0106
MOV AX, 004CCh
OR AX, 0FF00h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen source reset aux vars debug stack flags