NAVEENA NARAYANA POOJARI
1MS18CS081

List out the steps in designing ALU

Step 1: Add the two input pins,
Drop two East facing input pins on the canvas 4 bits Each label A and B ensure that input is 4 bits

Step 2: Add the adder / subtractor and gates Now we add the sub circuits created earlier Select Circuits under main project handler folder

Step 3: Add the multipliers
Take one or more inputs and generates a single output in logisim multiplexers and under Plexer folder click multiplexer leor and drop two of them into canvas

Step 4: Add controls
Drop two pins on the canvas north facing with 1 data bit. Label them 0 and 1 Respectively

Step 5: Add a splitter
Next we add a splitter into our circuit that takes one line from the second multiplexer and split to 4 inputs to an OR gate for a 4-bit ALU

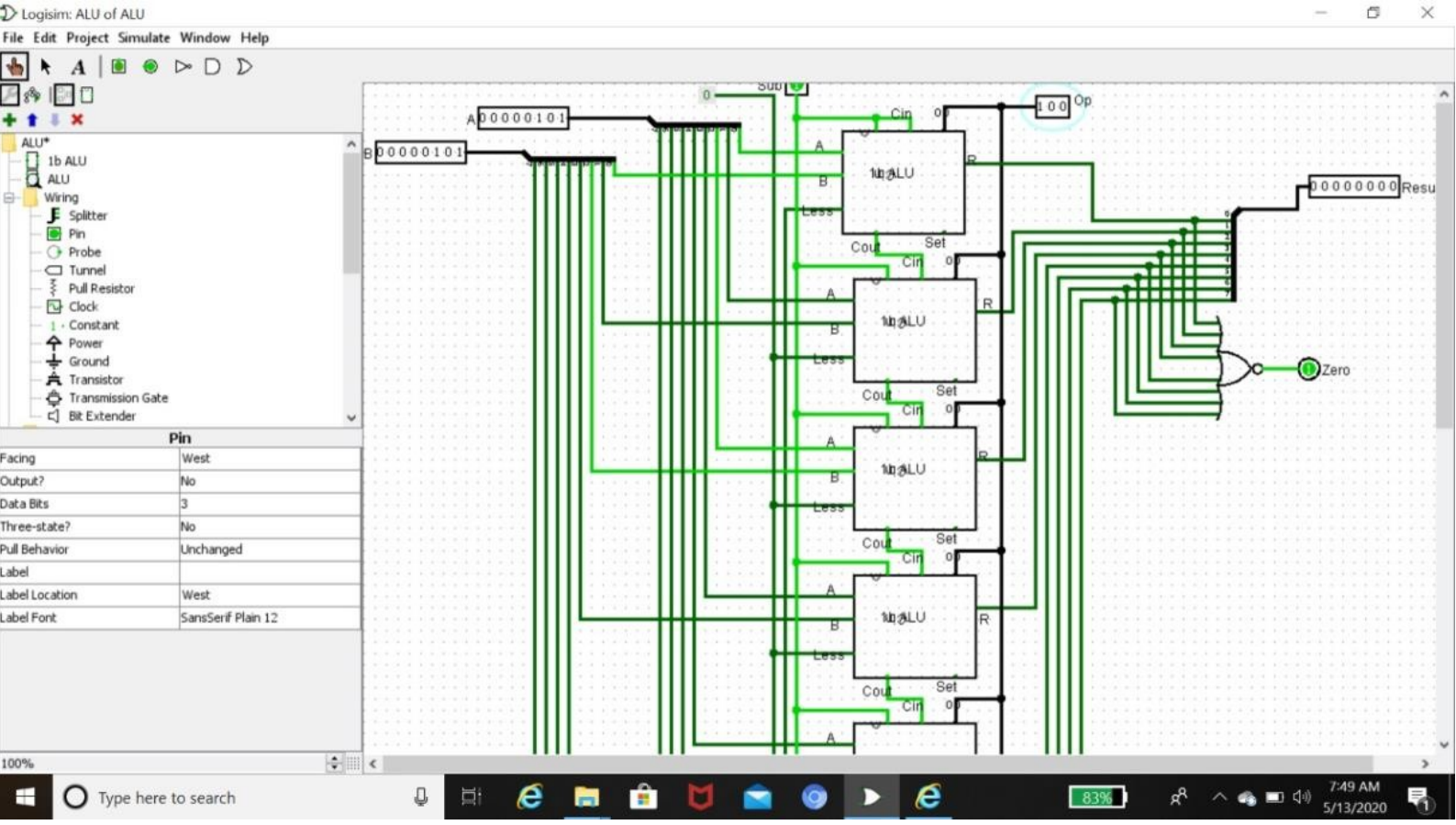
Step 6: Add another OR gate and NOT gate Now we add an OR gate after the splitter which has 4 inputs. TO right of the OR gate and a NOT gate

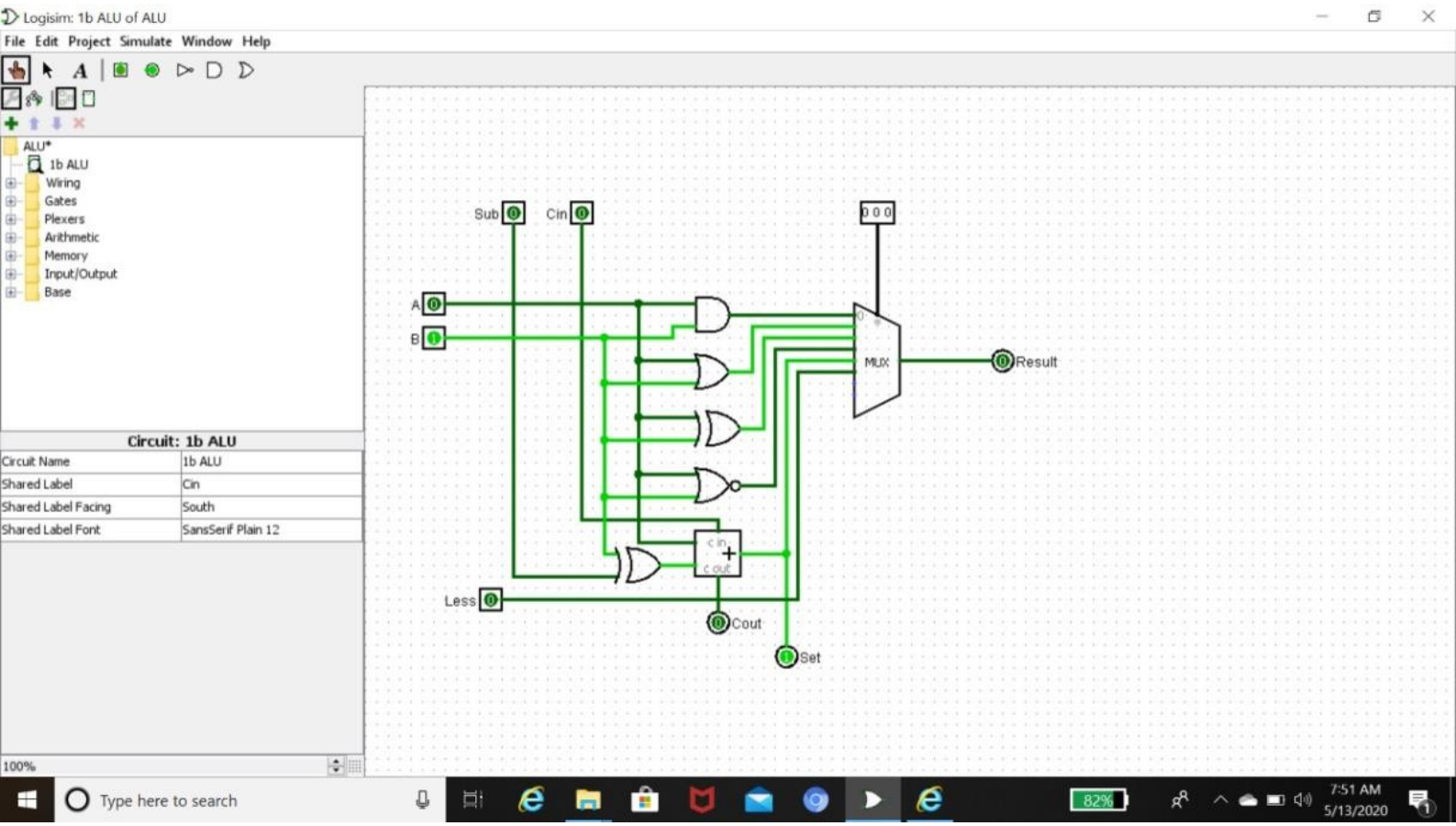
Naveena Narayana Poojari
1MS18CS081

This Arrangement accounts for zero output when all the bits results in zero.
The Not gate following The OR gate achieves this
Finally add a single bit pin after the NOT gate to store the result, label it zero.

Step 7: Add a result pin for the Mux

We handled the zeroes carrying from the MUX but we also need to account for valid combinations inputs from A, B and the Control inputs


Logisim: ALU of ALU
File Edit Project Simulate Window Help
ALU*
1b ALU
ALU
Wiring
Splitter
Pin
Probe
Tunnel
Pull Resistor
Clock
Constant
Power
Ground
Transistor
Transmission Gate
Bit Extender
Pin
Facing West
Output? No
Data Bits 3
Three-state? No
Pull Behavior Unchanged
Label
Label Location West
Label Font SansSerif Plain 12
100%
A 00000101
B 00000101
Sub
0
Cin
Op
100
A
B
Less
Cout
Set
R
1b ALU
00000000
Zero
Type here to search
83%
7:49 AM
5/13/2020


Logisim: 1b ALU of ALU
File Edit Project Simulate Window Help
ALU*
1b ALU
Wiring
Gates
Plexers
Arithmetic
Memory
Input/Output
Base
Circuit: 1b ALU
Circuit Name 1b ALU
Shared Label Cin
Shared Label Facing South
Shared Label Font SansSerif Plain 12
100%
Sub
Cin
000
A
B
MUX
Result
c in
c out
Less
Cout
Set
Type here to search
82%
7:51 AM
5/13/2020

NAVEENA NARAYANA POOJARI
1MS18CS081

List out the steps in designing memory system

Step 1: Add ram

Select a separate load and store operation for RAM

Step 2: Add Counter

Step 3: Connect Counter clock, and Controlled Buffer to the RAM

Step 3: Add TTY

To display Data Read on Memory

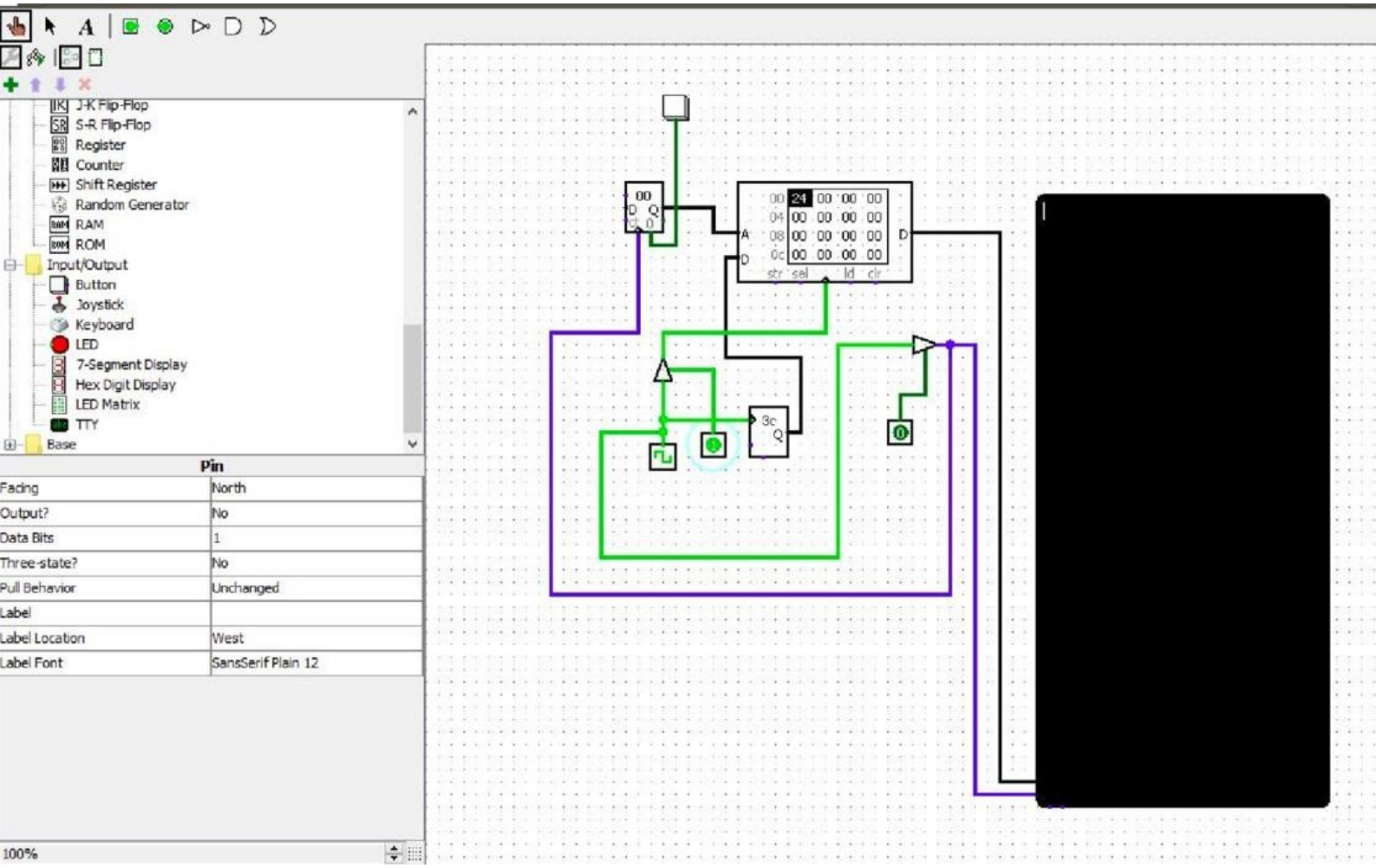
Step 4: Add Random Generator

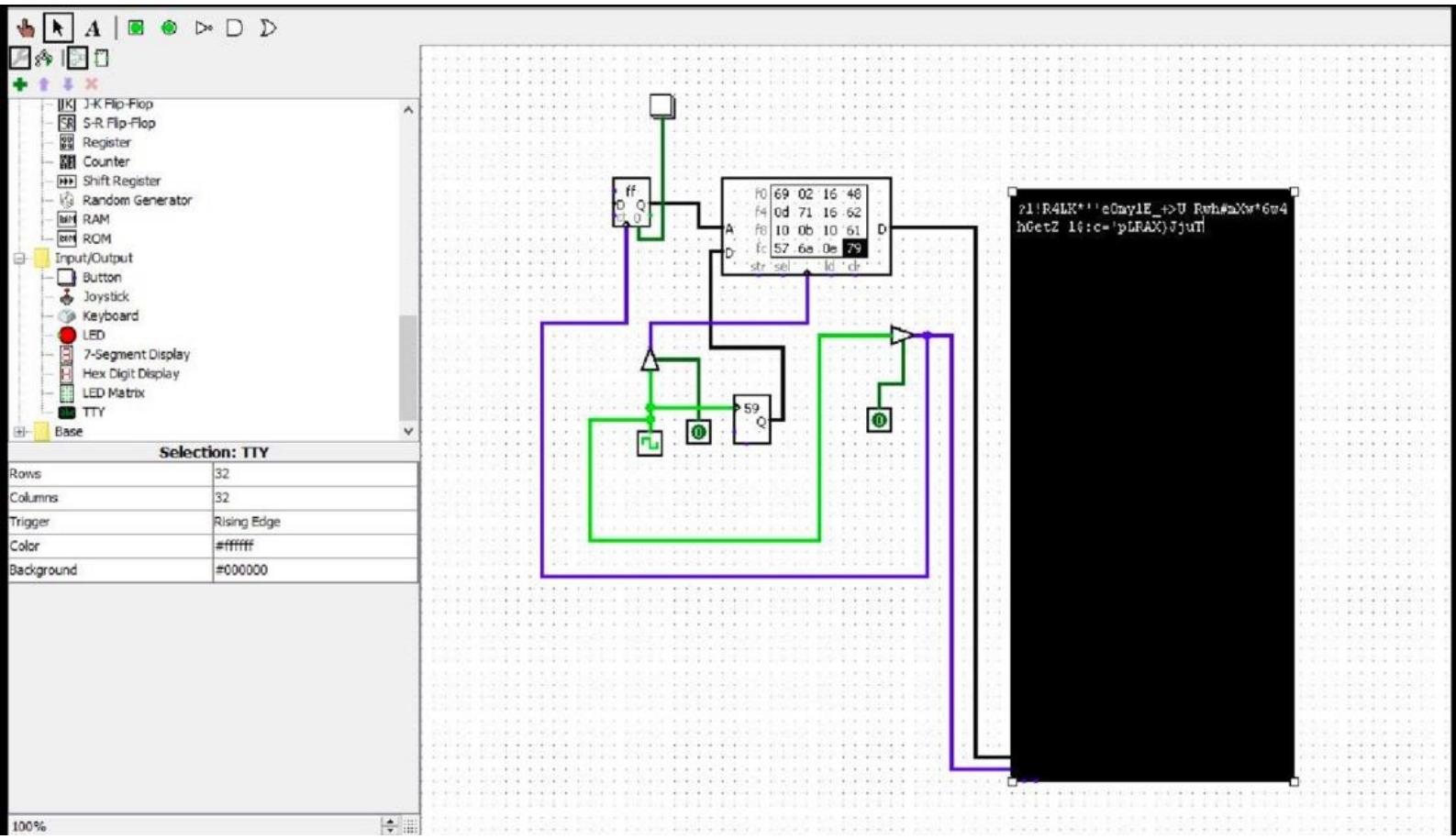
To Generate different address location

Add input and another Controlled Buffer to the Random Generator

Step 5: Add Button

Connect Button to Counter


J-K Flip-Flop
S-R Flip-Flop
Register
Counter
Shift Register
Random Generator
RAM
ROM
Input/Output
Button
Joystick
Keyboard
LED
7-Segment Display
Hex Digit Display
LED Matrix
TTY
Base
Pin
Facing North
Output? No
Data Bits 1
Three-state? No
Pull Behavior Unchanged
Label
Label Location West
Label Font SansSerif Plain 12
100%

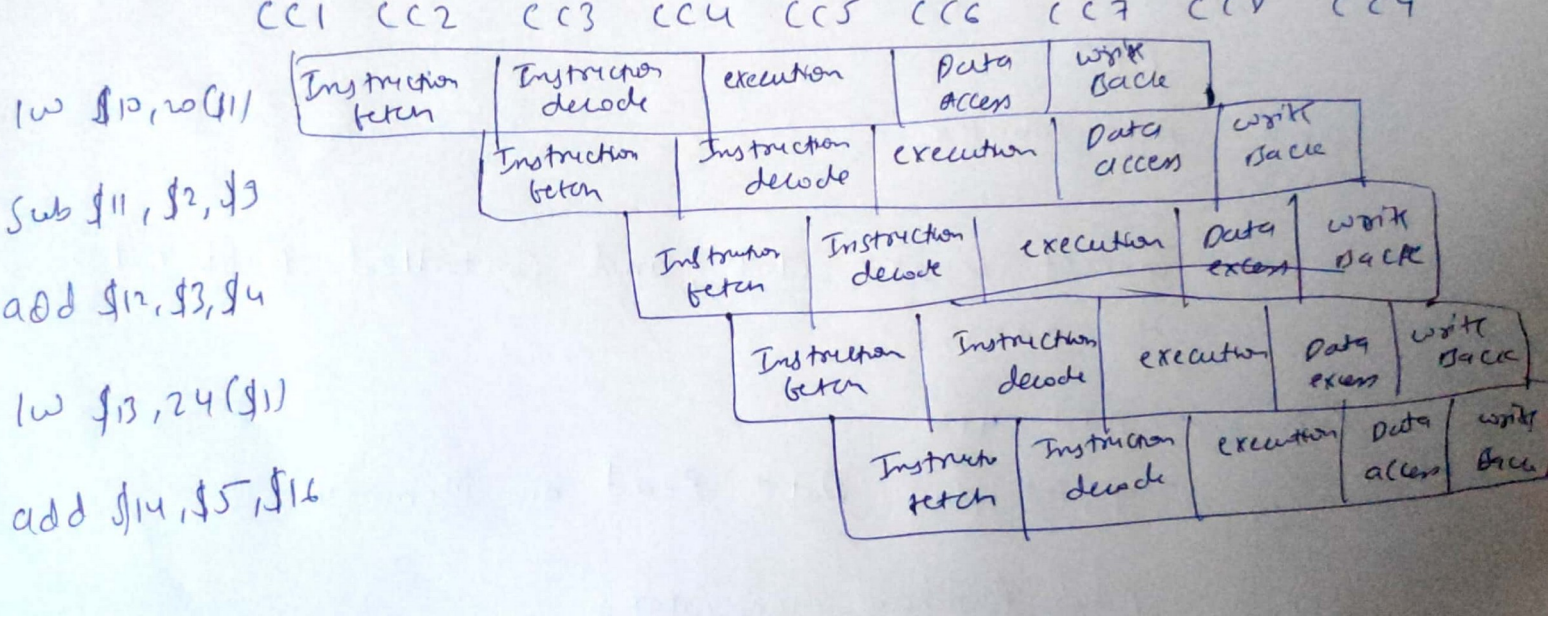

J-K Flip-Flop
S-R Flip-Flop
Register
Counter
Shift Register
Random Generator
RAM
ROM
Input/Output
Button
Joystick
Keyboard
LED
7-Segment Display
Hex Digit Display
LED Matrix
TTY
Base
Selection: TTY
Rows 32
Columns 32
Trigger Rising Edge
Color #ffffff
Background #000000
100%
ff
59
f0 69 02 16 48
f4 0d 71 16 62
f8 10 0b 10 61
fc 57 6a 0e 79
str sel ld clr
?l!R4LK*''e0myiE_+>U Rvrh#aXw*6w4
hGetZ 1@:c='pLRAX)JjuT

Naveena Narayana Poojari
1MS18CS081

Time C in Clock cycles)

Program execution order

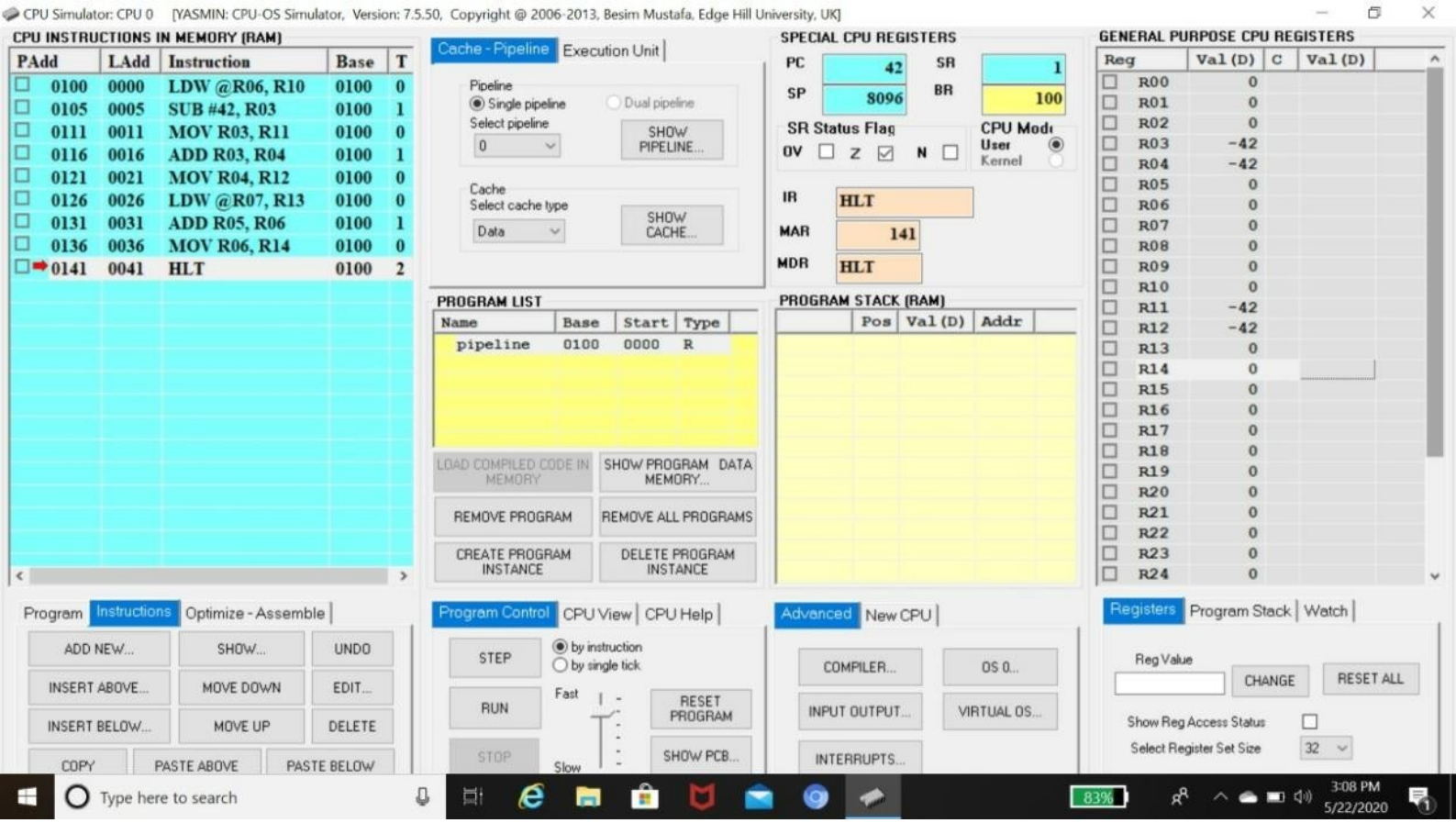
in instruction

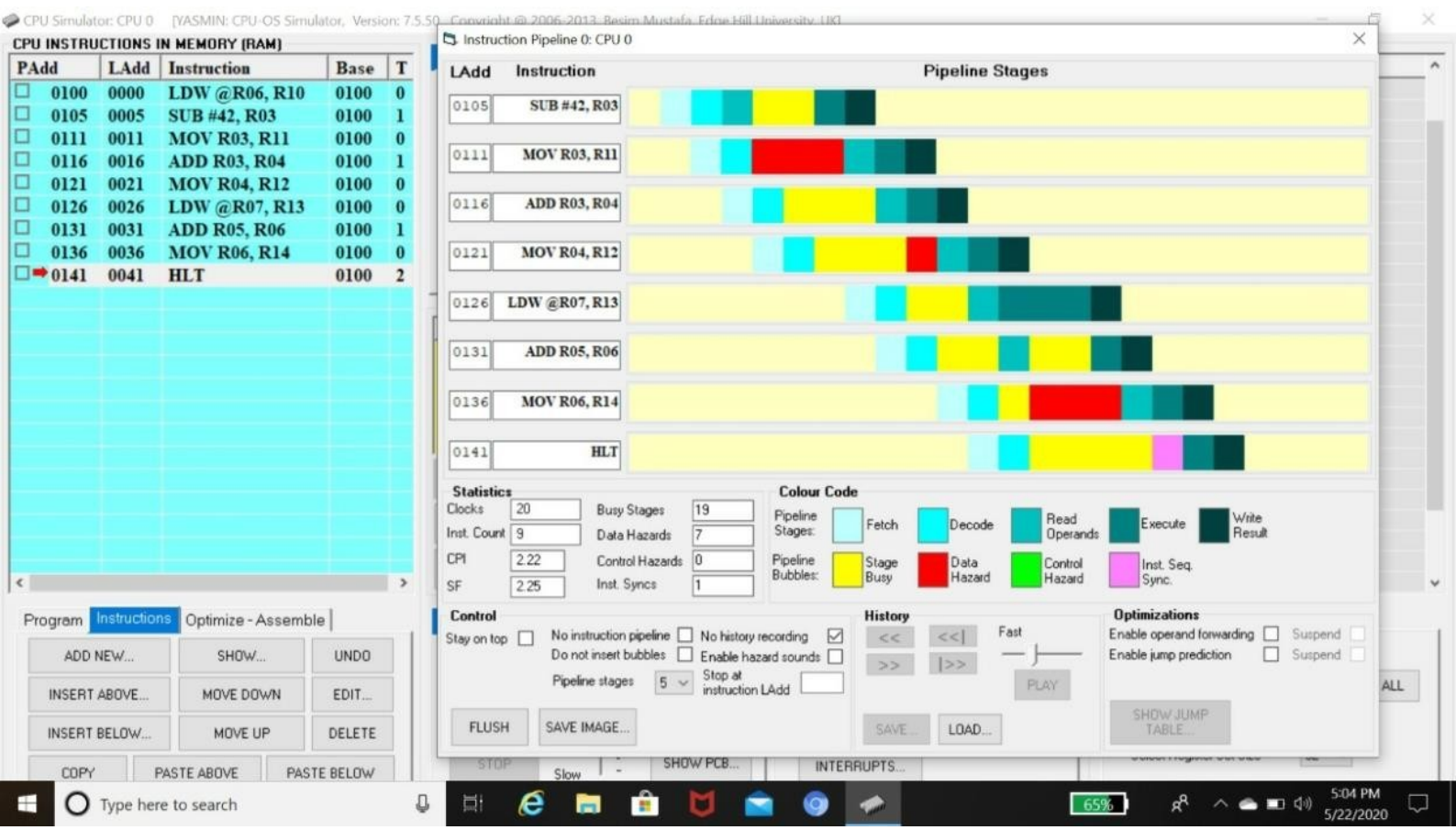


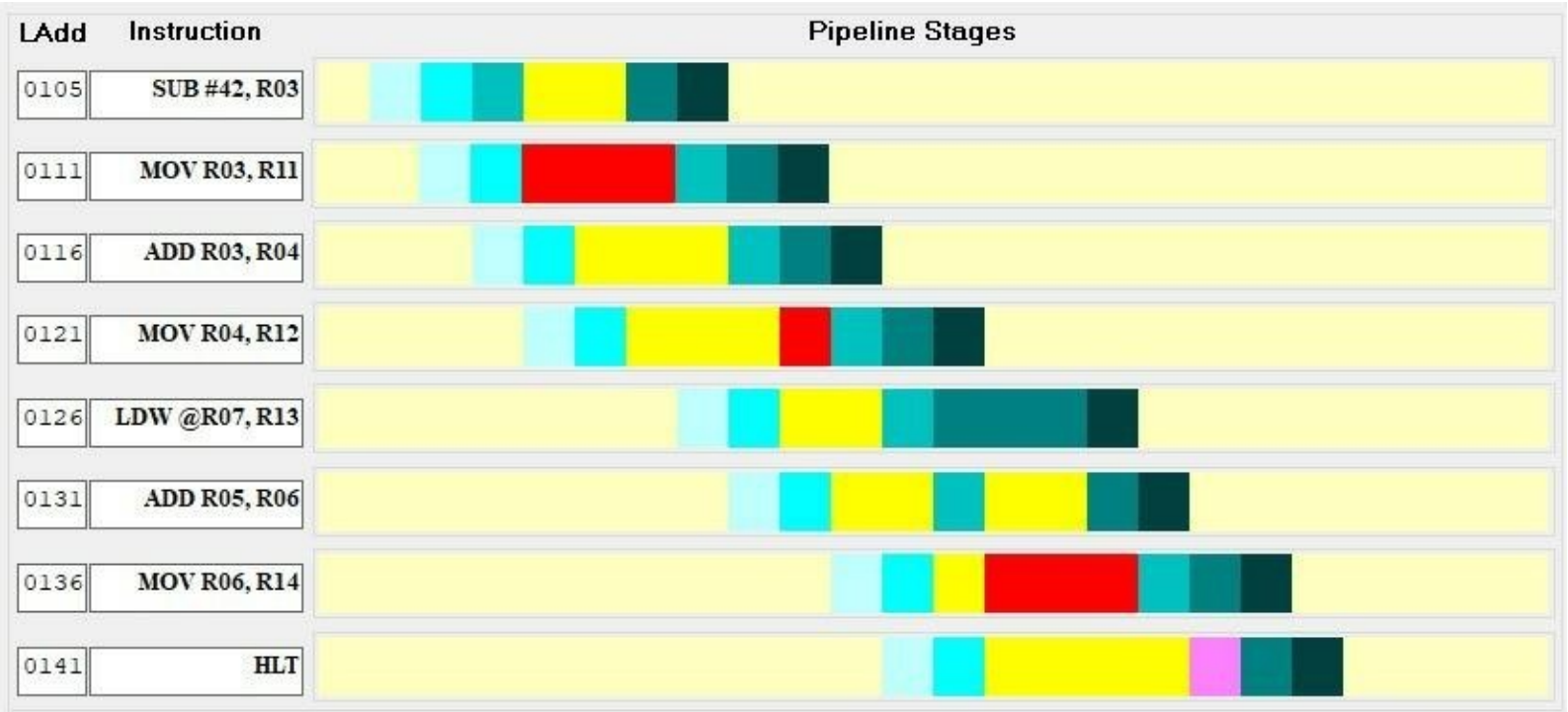
Naveena Narayana Poojari
1MS13CS081

Time in clock cycles

	CC1	CC2	CC3	CC4	CC5	CC6	CC7
Program execution order							
lw $10,20 ($1)	Im	Reg	A	Dm	Reg		
Sub $11, $2, $3		Im	Reg	A	Dm	Reg	
add $12, $3, $4			Im	Reg	A	Dm	Reg
lw $13, 24, ($1)				Im	Reg	A	Dm
add $14, $5, $6					Im	Reg	A


CPU Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright @ 2006-2013, Besim Mustafa, Edge Hill University, UK]
CPU INSTRUCTIONS IN MEMORY (RAM)
PAdd LAdd Instruction Base T
0100 0000 LDW @R06, R10 0100 0
0105 0005 SUB #42, R03 0100 1
0111 0011 MOV R03, R11 0100 0
0116 0016 ADD R03, R04 0100 1
0121 0021 MOV R04, R12 0100 0
0126 0026 LDW @R07, R13 0100 0
0131 0031 ADD R05, R06 0100 1
0136 0036 MOV R06, R14 0100 0
0141 0041 HLT 0100 2
Program Instructions Optimize - Assemble
ADD NEW... SHOW... UNDO
INSERT ABOVE... MOVE DOWN EDIT...
INSERT BELOW... MOVE UP DELETE
COPY PASTE ABOVE PASTE BELOW
Cache - Pipeline Execution Unit
Pipeline
Single pipeline Dual pipeline
Select pipeline
0
SHOW PIPELINE...
Cache
Select cache type
Data
SHOW CACHE...
SPECIAL CPU REGISTERS
PC 42
SR 1
SP 8096
BR 100
SR Status Flag
OV Z N
CPU Mode
User
Kernel
IR HLT
MAR 141
MDR HLT
PROGRAM LIST
Name Base Start Type
pipeline 0100 0000 R
LOAD COMPILED CODE IN MEMORY
SHOW PROGRAM DATA MEMORY...
REMOVE PROGRAM
REMOVE ALL PROGRAMS
CREATE PROGRAM INSTANCE
DELETE PROGRAM INSTANCE
PROGRAM STACK (RAM)
Pos Val (D) Addr
GENERAL PURPOSE CPU REGISTERS
Reg Val (D) C Val (D)
R00 0
R01 0
R02 0
R03 -42
R04 -42
R05 0
R06 0
R07 0
R08 0
R09 0
R10 0
R11 -42
R12 -42
R13 0
R14 0
R15 0
R16 0
R17 0
R18 0
R19 0
R20 0
R21 0
R22 0
R23 0
R24 0
Program Control CPU View CPU Help
STEP
by instruction
by single tick
RUN
Fast
Slow
RESET PROGRAM
STOP
SHOW PCB...
Advanced New CPU
COMPILER...
OS 0...
INPUT OUTPUT...
VIRTUAL OS...
INTERRUPTS...
Registers Program Stack Watch
Reg Value
CHANGE
RESET ALL
Show Reg Access Status
Select Register Set Size
32
Type here to search
83%
3:08 PM
5/22/2020


CPU Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright @ 2006-2013, Besim Mustafa, Edge Hill University, UK]
CPU INSTRUCTIONS IN MEMORY (RAM)
PAdd LAdd Instruction Base T
0100 0000 LDW @R06, R10 0100 0
0105 0005 SUB #42, R03 0100 1
0111 0011 MOV R03, R11 0100 0
0116 0016 ADD R03, R04 0100 1
0121 0021 MOV R04, R12 0100 0
0126 0026 LDW @R07, R13 0100 0
0131 0031 ADD R05, R06 0100 1
0136 0036 MOV R06, R14 0100 0
0141 0041 HLT 0100 2
Program Instructions Optimize - Assemble
ADD NEW... SHOW... UNDO
INSERT ABOVE... MOVE DOWN EDIT...
INSERT BELOW... MOVE UP DELETE
COPY PASTE ABOVE PASTE BELOW
Instruction Pipeline 0: CPU 0
LAdd Instruction Pipeline Stages
0105 SUB #42, R03
0111 MOV R03, R11
0116 ADD R03, R04
0121 MOV R04, R12
0126 LDW @R07, R13
0131 ADD R05, R06
0136 MOV R06, R14
0141 HLT
Statistics
Clocks 20
Busy Stages 19
Inst. Count 9
Data Hazards 7
CPI 2.22
Control Hazards 0
SF 2.25
Inst. Syncs 1
Colour Code
Pipeline Stages: Fetch Decode Read Operands Execute Write Result
Pipeline Bubbles: Stage Busy Data Hazard Control Hazard Inst. Seq. Sync.
Control
Stay on top
No instruction pipeline
No history recording
Do not insert bubbles
Enable hazard sounds
Pipeline stages 5
Stop at instruction LAdd
FLUSH
SAVE IMAGE...
History
Fast
PLAY
SAVE...
LOAD...
Optimizations
Enable operand forwarding Suspend
Enable jump prediction Suspend
SHOW JUMP TABLE...
STOP
Slow
SHOW PCB...
INTERRUPTS...
Type here to search
65%
5:04 PM
5/22/2020



Statistics

Clocks	20	Busy Stages	19
Inst. Count	9	Data Hazards	7
CPI	2.22	Control Hazards	0
SF	2.25	Inst. Syncs	1

Colour Code

Pipeline Stages: Fetch, Decode, Read Operands, Execute, Write Result

Pipeline Bubbles: Stage Busy, Data Hazard, Control Hazard, Inst. Seq. Sync.

Control

Stay on top ☐ | No instruction pipeline ☐ | No history recording ☑ | Do not insert bubbles ☐ | Enable hazard sounds ☐

Pipeline stages 5 | Stop at instruction LAdd

FLUSH | SAVE IMAGE...

History

<< | <<| | >> | |>> | Fast | PLAY | SAVE... | LOAD...

Optimizations

Enable operand forwarding ☐ Suspend ☐

Enable jump prediction ☐ Suspend ☐

SHOW JUMP TABLE...