Prajwal
1MS19CS091

list out the steps in designig ALU

Step1: Add the two input pins, Drop two East facing input
s pins on the canwas 4 bits each label A and B ensure
that input is 4 bits

Step2: Add the Adder/subtractor and gates Now we add
the Sub Circuits created earliear Select Circuits under main
Project handler folder

Step3: Add the multipliers
Take on of mole inputs and generate a single output in
logisim multiplexers and under plexer folder. click
Multiplexer Icon and drop two of them into canlaas

Step4: Add controls
Drop two pins on the canwas north facing with 1 data
bit. Label them 0 & 1 respectively

Step5: 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.

This Arrangement accounts for zero output when
all the bits result in zero.

Prajwal
1MS18CS091

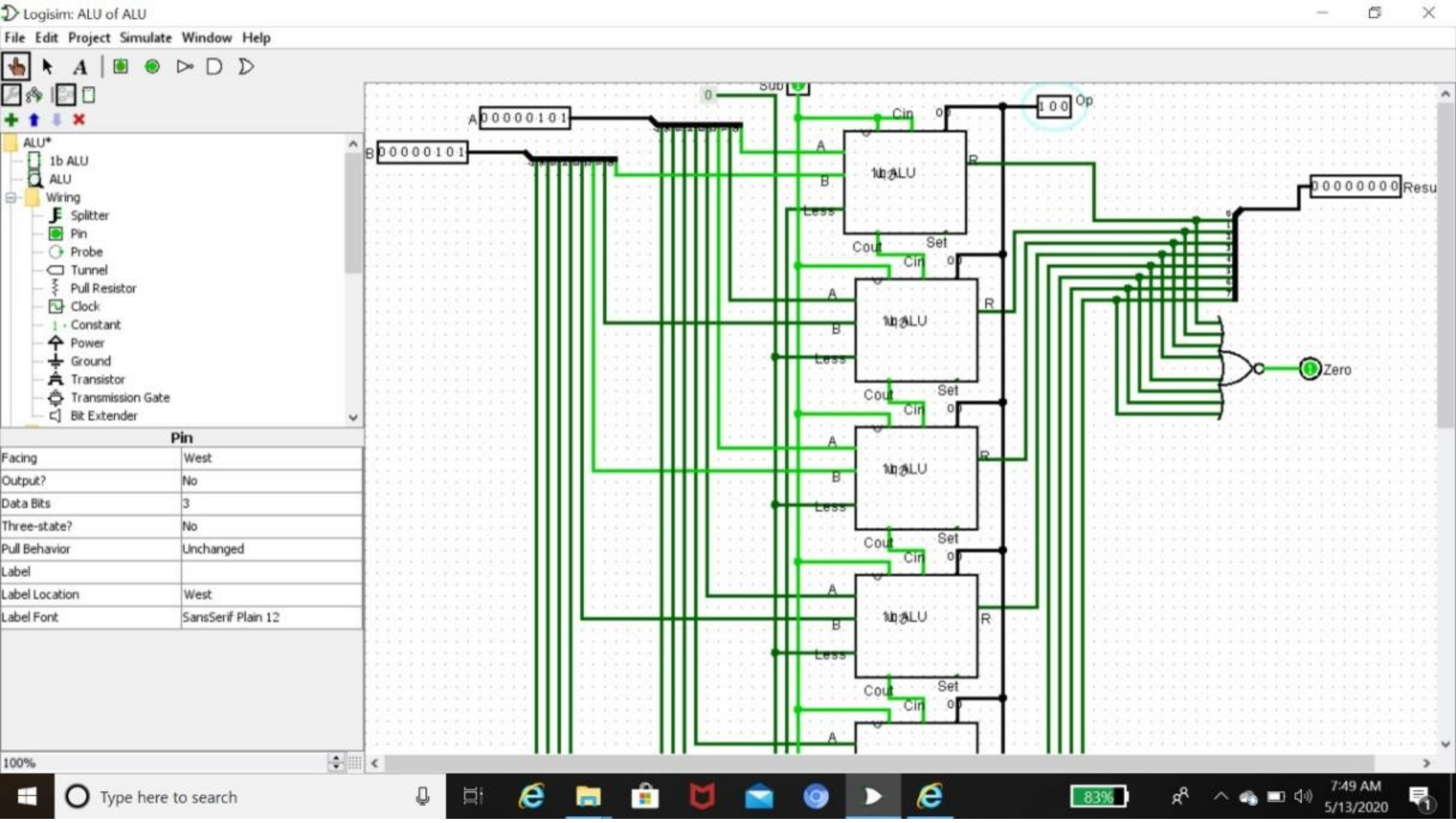
The Not gate following The OR gate achieves this

Finally add a single bit pin after the NOT gate to Store the result, label it zero

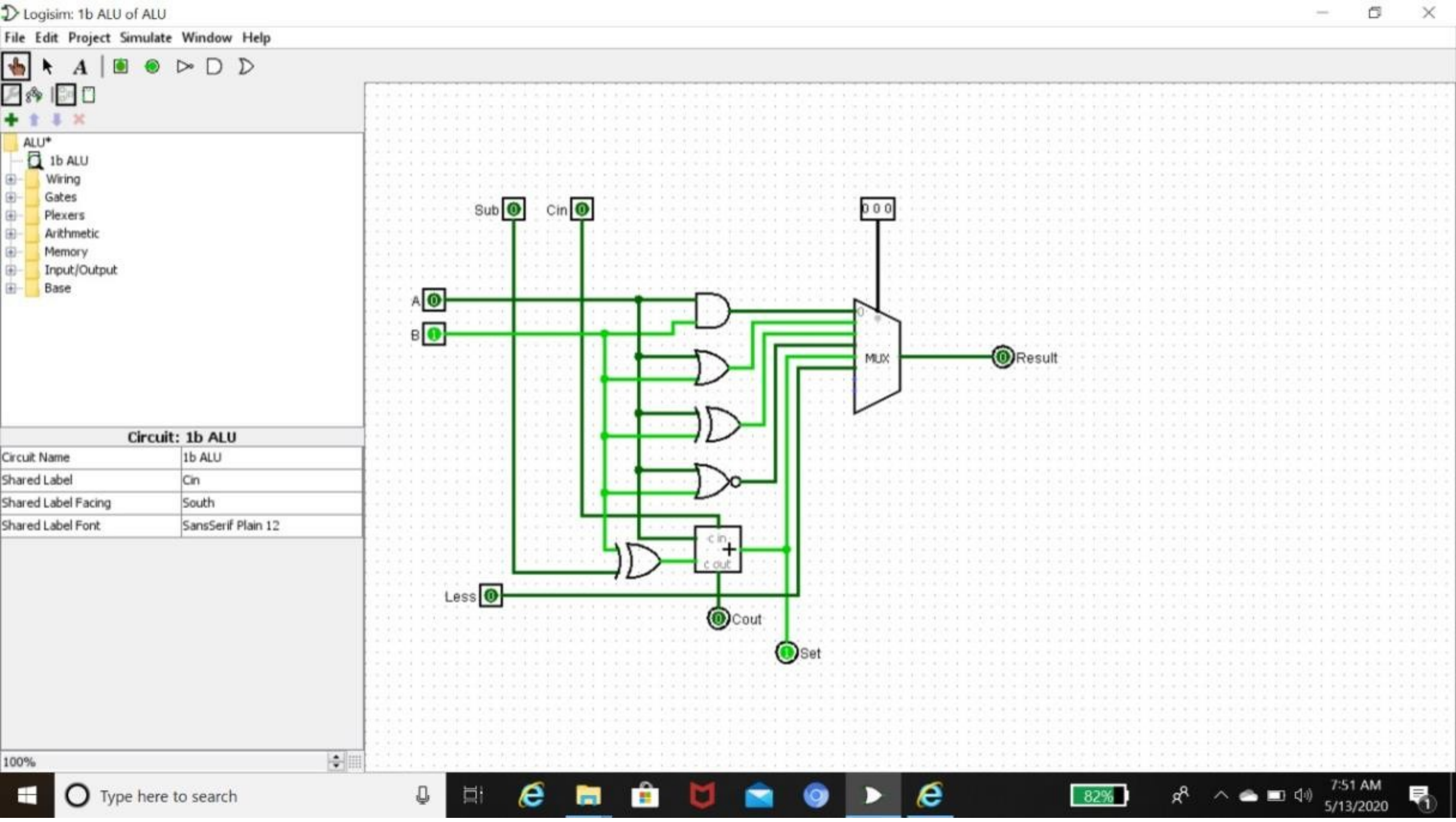
Step 7: Add a result pin for the Mux

we handled the Zeroes carring from the Mux but we also need to account for valid combinations Inputs from A, B and the Control inputs

in



Pin	
Facing	West
Output?	No
Data Bits	3
Three-state?	No
Pull Behavior	Unchanged
Label	
Label Location	West
Label Font	SansSerif Plain 12



Circuit: 1b ALU

Circuit Name	1b ALU
Shared Label	Cin
Shared Label Facing	South
Shared Label Font	SansSerif Plain 12

Prasad
1MS18CS091

List out the steps in designing memory system

Step1: Add ram

Select a separate Load and Store operation for RAM

Step2: Add Counter

Step3: Connect Counter Clock, and Controlled Buffer to the RAM

Step 3: Add TTY

To display Data Read on Memory

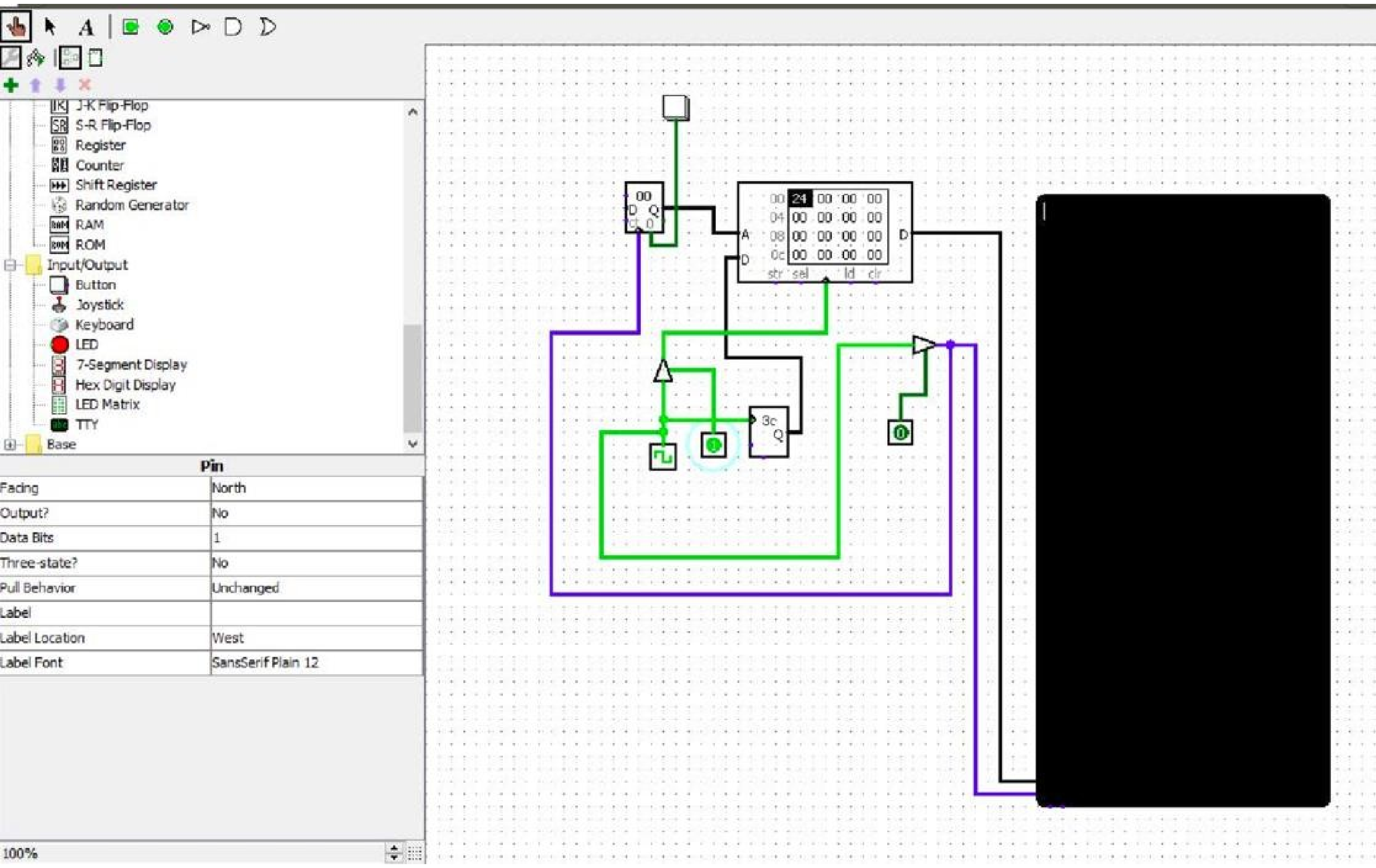
Step 4: Add Random Generator

To Generate different address location

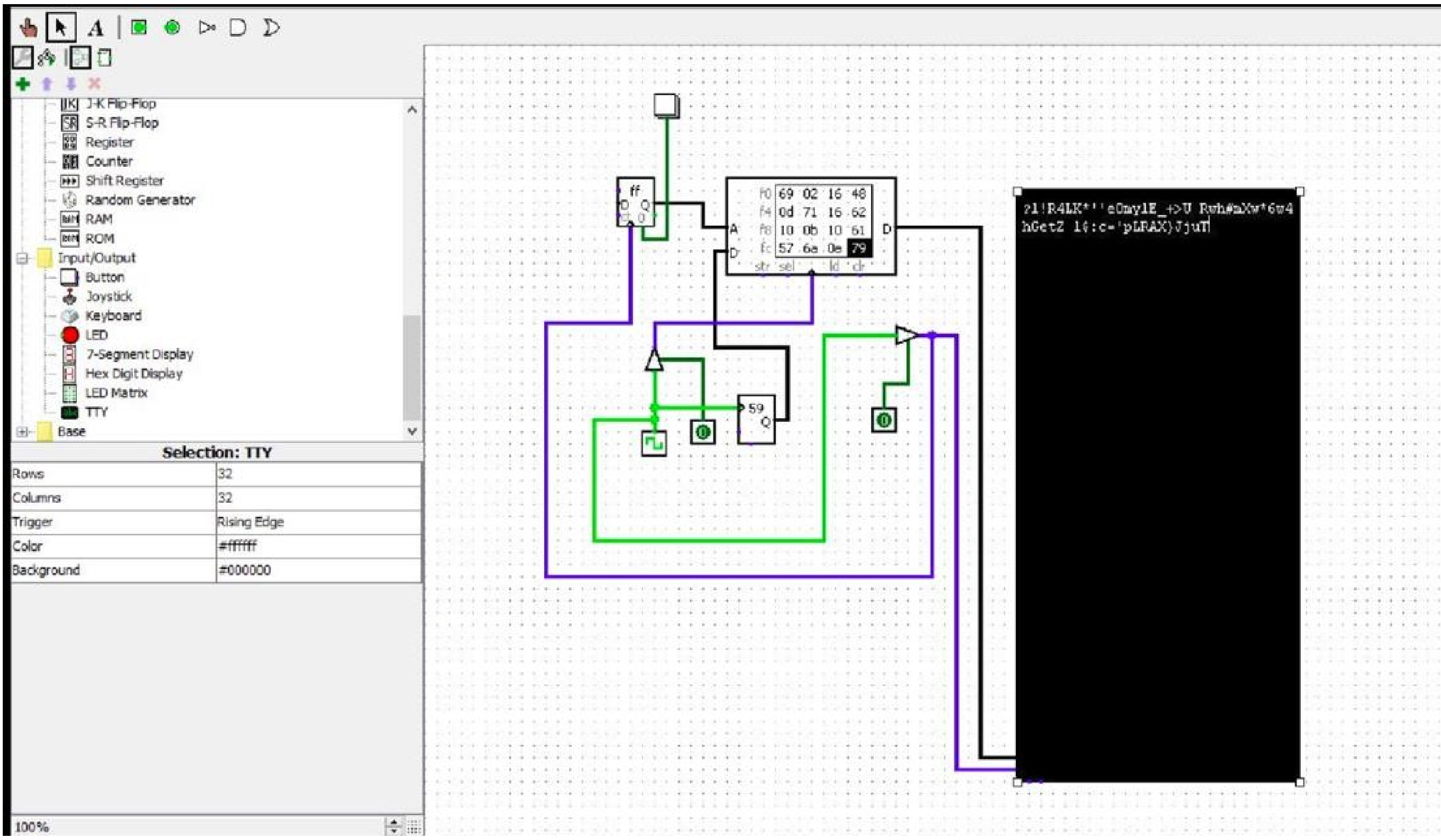
Add input and another Controlled Buffer to the Random Generator.

Step5: Add Button

Connect Button to Counter.



Pin	
Facing	North
Output?	No
Data Bits	1
Three-state?	No
Pull Behavior	Unchanged
Label	
Label Location	West
Label Font	SansSerif Plain 12

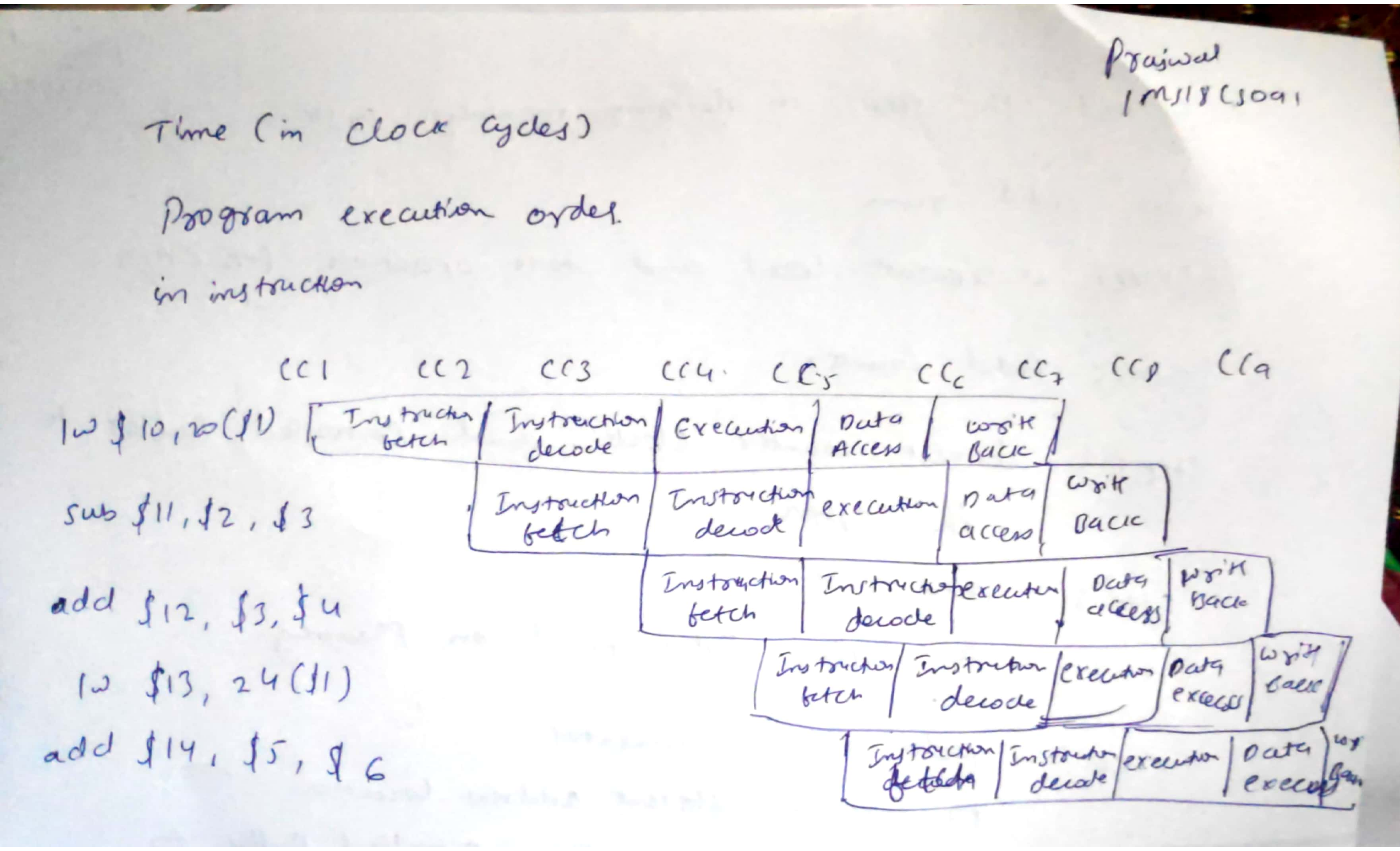


Selection: TTY

Rows	32
Columns	32
Trigger	Rising Edge
Color	#ffffff
Background	#000000

```
f0 69 02 16 48
f4 0d 71 16 62
f8 10 0b 10 61
fc 57 6a 0e 79
```

```
?1!R4LK*''eOmylE_+>U Rwh#mXw*6w4
hGetZ 1{:c='pLRAX)JjuT
```



Prajwal
1MS18CS091

Time (in clock cycles)

Program execution order

in instruction

	CC1	CC2	CC3	CC4.	CC5	CC6	CC7	CC8	CC9
lw $10, 20($1)	Instruction fetch	Instruction decode	Execution	Data Access	write Back				
sub $11, $2, $3		Instruction fetch	Instruction decode	execution	Data access	write Back			
add $12, $3, $4			Instruction fetch	Instruction decode	execution	Data access	write Back		
lw $13, 24($1)				Instruction fetch	Instruction decode	execution	Data excess	write Back	
add $14, $5, $6					Instruction fetch	Instruction decode	execution	Data excess	write Back

Prajwal
1MS18CS091

Time in clock cycles

CC1 CC2 CC3 CC4 CC5 CC6 CC7

program execution order

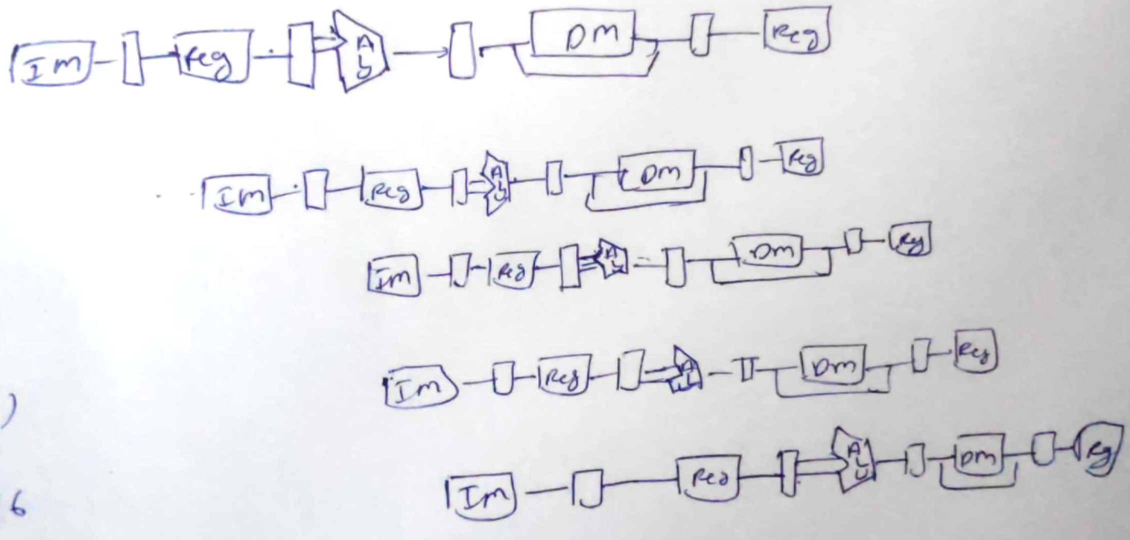
lw $10, 20($1)

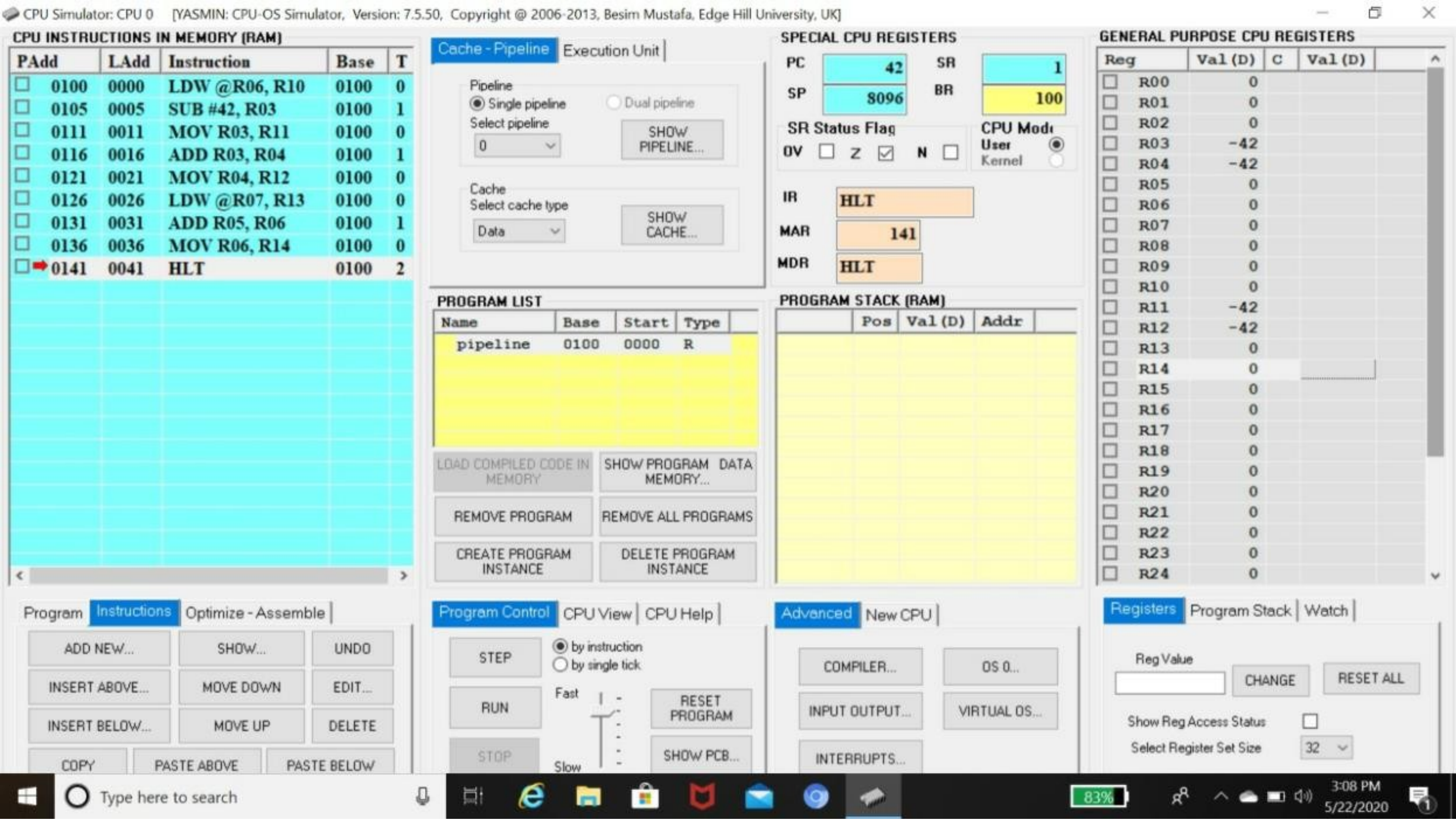
sub $11, $2, $3

add $12, $3, $4

lw $13, 24($1)

add $14, $5, $6





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

Pipeline
● Single pipeline ○ Dual pipeline
Select pipeline: 0

Cache
Select cache type: Data

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

PROGRAM STACK (RAM)

Pos	Val (D)	Addr

GENERAL PURPOSE CPU REGISTERS

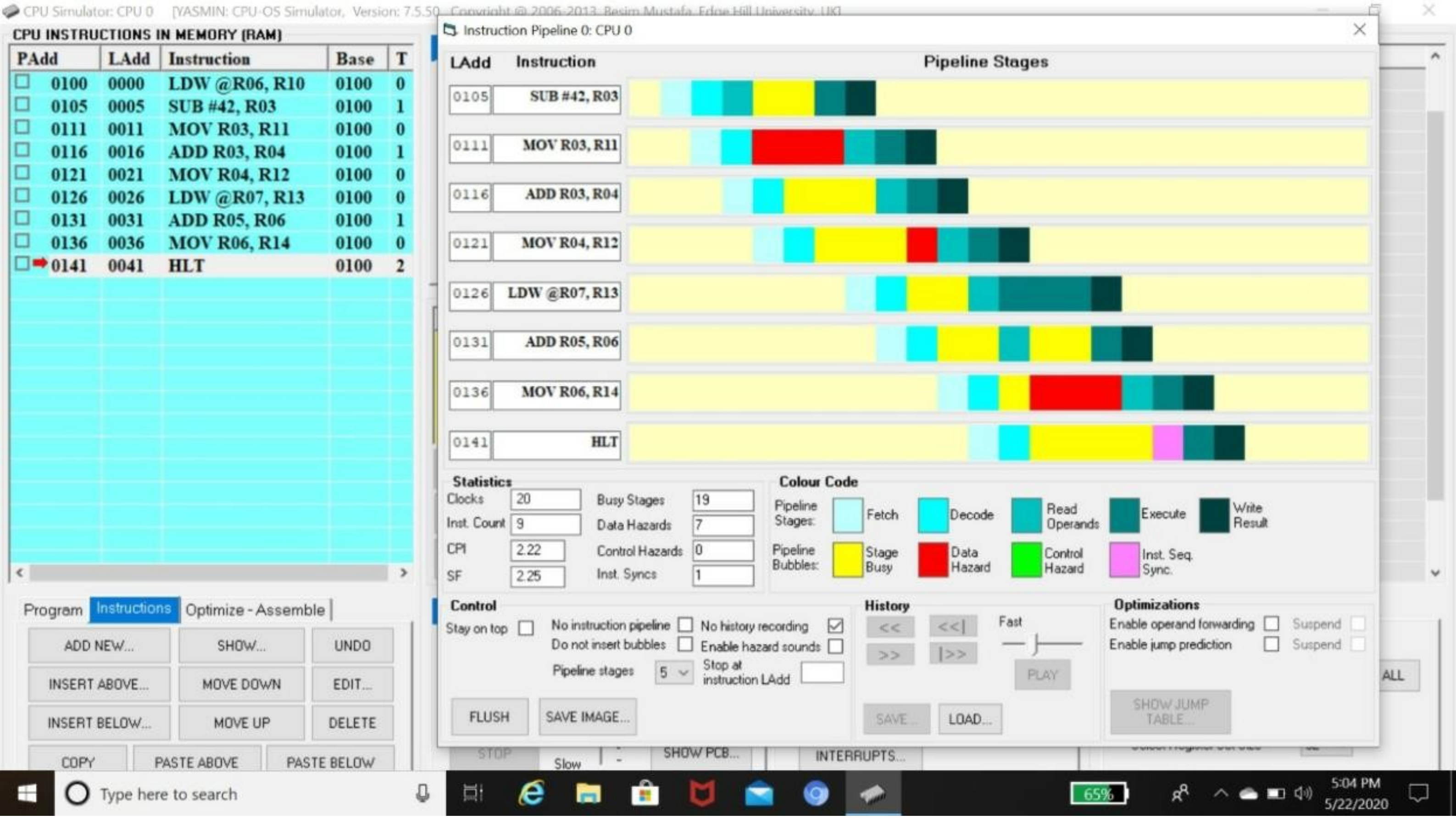
Reg	Val (D)	C	Val (D)
R00	0		
R01	0		
R02	0		
R03	-42		
R04	-42		
R05	0		
R06	0		
R07	0		
R08	0		
R09	0		
R10	0		
R11	-42		
R12	-42		
R13	0		
R14	0		
R15	0		
R16	0		
R17	0		
R18	0		
R19	0		
R20	0		
R21	0		
R22	0		
R23	0		
R24	0		

● by instruction ○ by single tick

Reg Value

Show Reg Access Status □
Select Register Set Size 32

83%
3:08 PM
5/22/2020



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

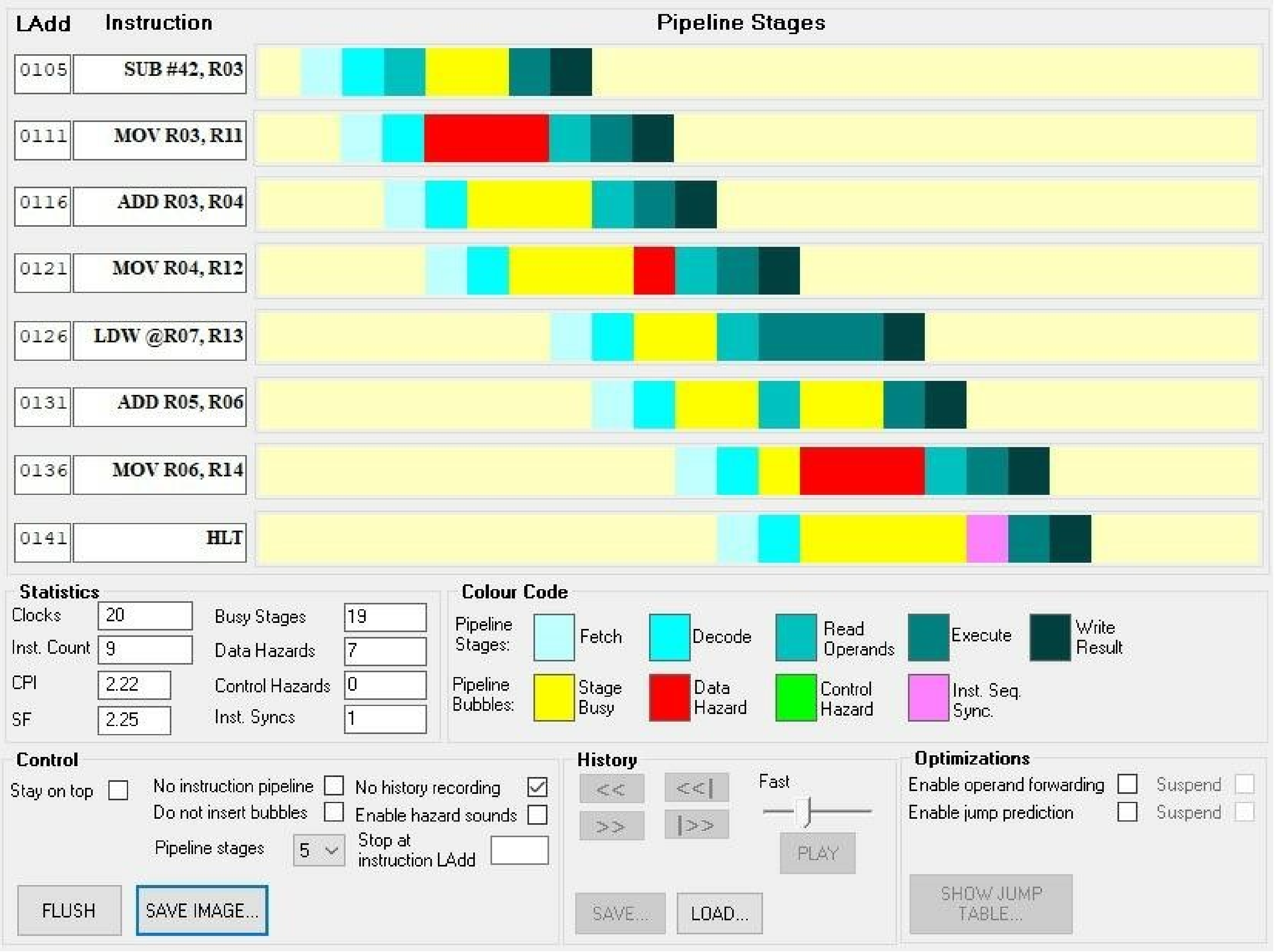
Colour Code

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

Optimizations
Enable operand forwarding □ Suspend □
Enable jump prediction □ Suspend □

65%
5:04 PM
5/22/2020



LAdd	Instruction
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

History

Optimizations

Enable operand forwarding ☐ Suspend ☐

Enable jump prediction ☐ Suspend ☐