Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)

Department of CSE

Programme: B.E **Term: Jan to May 2019**
Course: Computer Organization **Course Code: CS45**

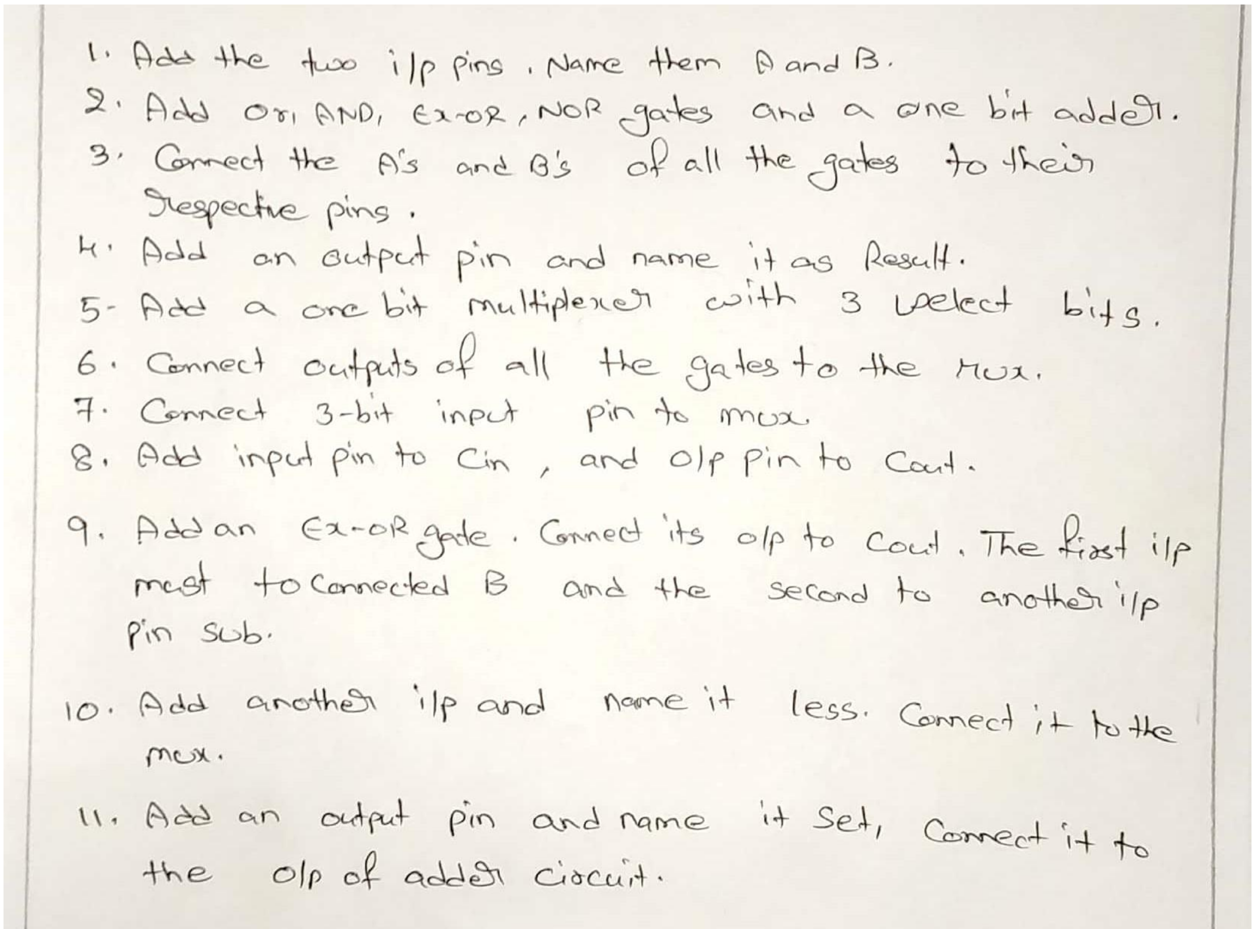
Activity V: Designing an ALU to perform arithmetic and logical functions using Logisim simulator.

Name:MK NIKHIL	**Marks: /10**	**Date:20-05-2020**
USN:1MS18CS076	**Signature of the Faculty:**	

Objective: To simulate the working of Arithmetic and Logical Unit using simulator.

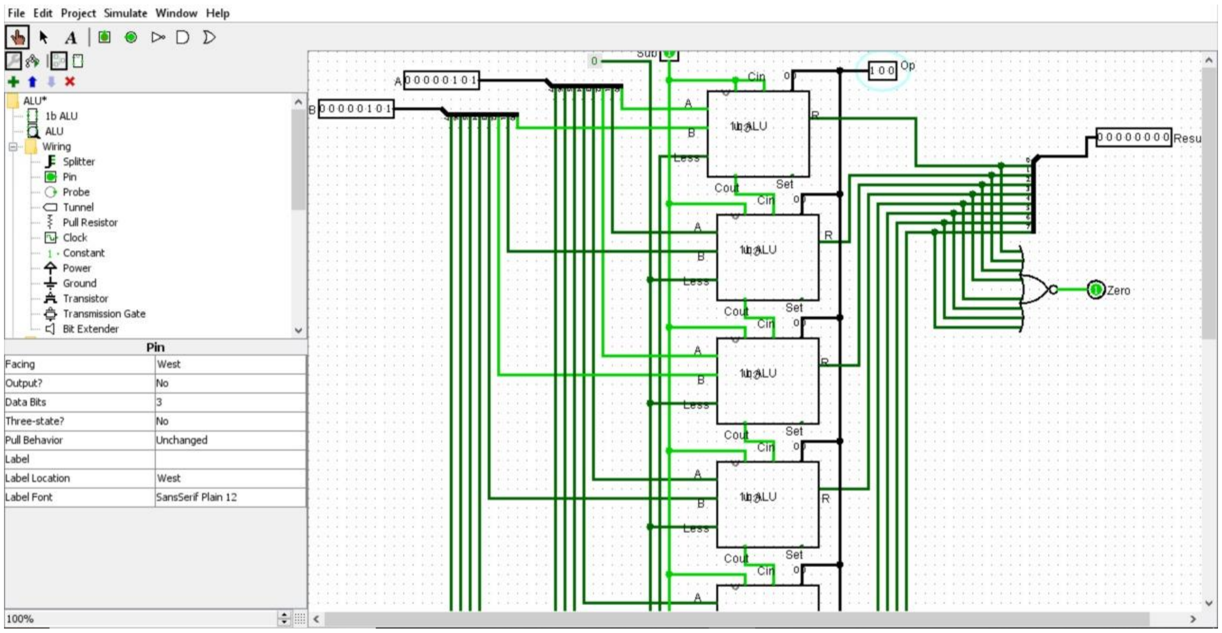
Simulator Description: Logisim is an educational tool for designing and simulating digital logic circuits. With its simple toolbar interface and simulation of circuits as you build them, it is simple enough to facilitate learning the most basic concepts related to logic circuits. With the capacity to build larger circuits from smaller sub circuits, and to draw bundles of wires with a single mouse drag, Logisim can be used (and is used) to design and simulate entire CPUs for educational purposes.

Activity to be performed by students:



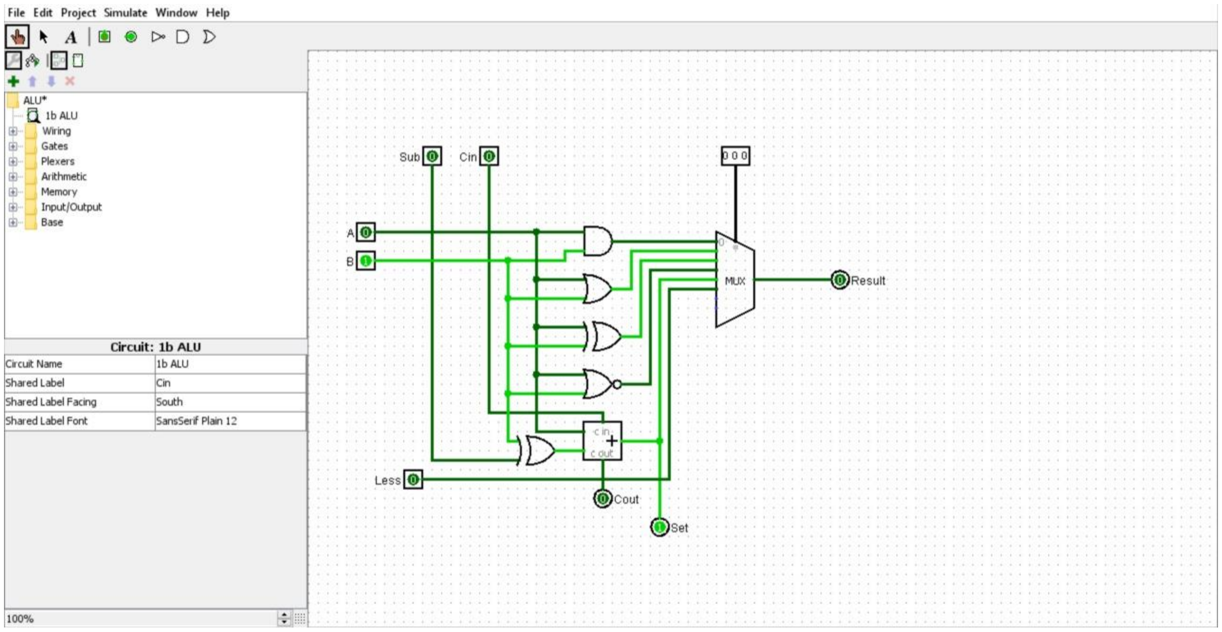
1. Add the two i/p pins. Name them A and B.
2. Add Or, AND, Ex-OR, NOR gates and a one bit adder.
3. Connect the A's and B's of all the gates to their respective pins.
4. Add an output pin and name it as Result.
5. Add a one bit multiplexer with 3 select bits.
6. Connect outputs of all the gates to the mux.
7. Connect 3-bit input pin to mux.
8. Add input pin to Cin, and o/p pin to Cout.
9. Add an Ex-OR gate. Connect its o/p to Cout. The first i/p must to connected B and the second to another i/p pin sub.
10. Add another i/p and name it less. Connect it to the mux.
11. Add an output pin and name it Set, Connect it to the o/p of adder circuit.

SNAPSHOTS



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

Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)

Department of CSE

Programme: B.E **Term: Jan to May 2019**
Course: Computer Organization **Course Code: CS45**

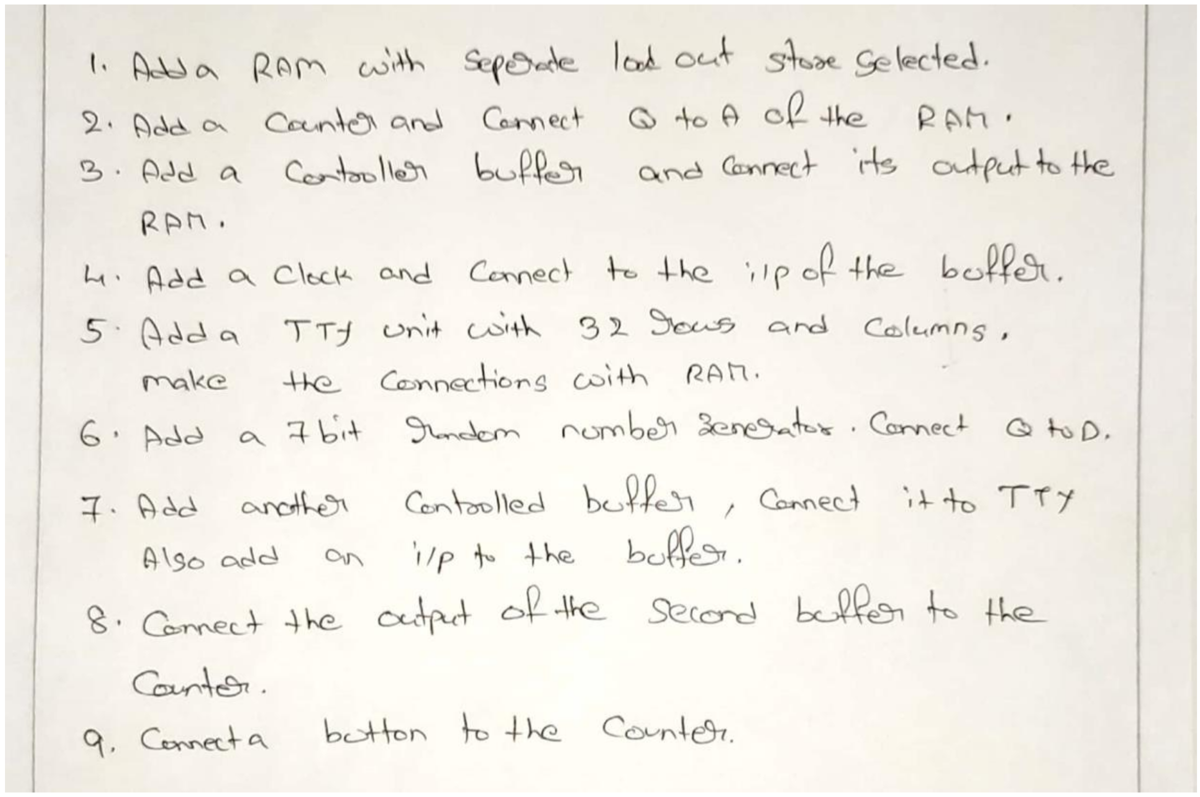
Activity VI: Designing memory system using Logisim simulator.

Name: MK NIKHIL	**Marks: /10**	**Date:20-05-2020**
USN: 1MS18CS076	**Signature of the Faculty:**	

Objective: To simulate the writing operation on memory.

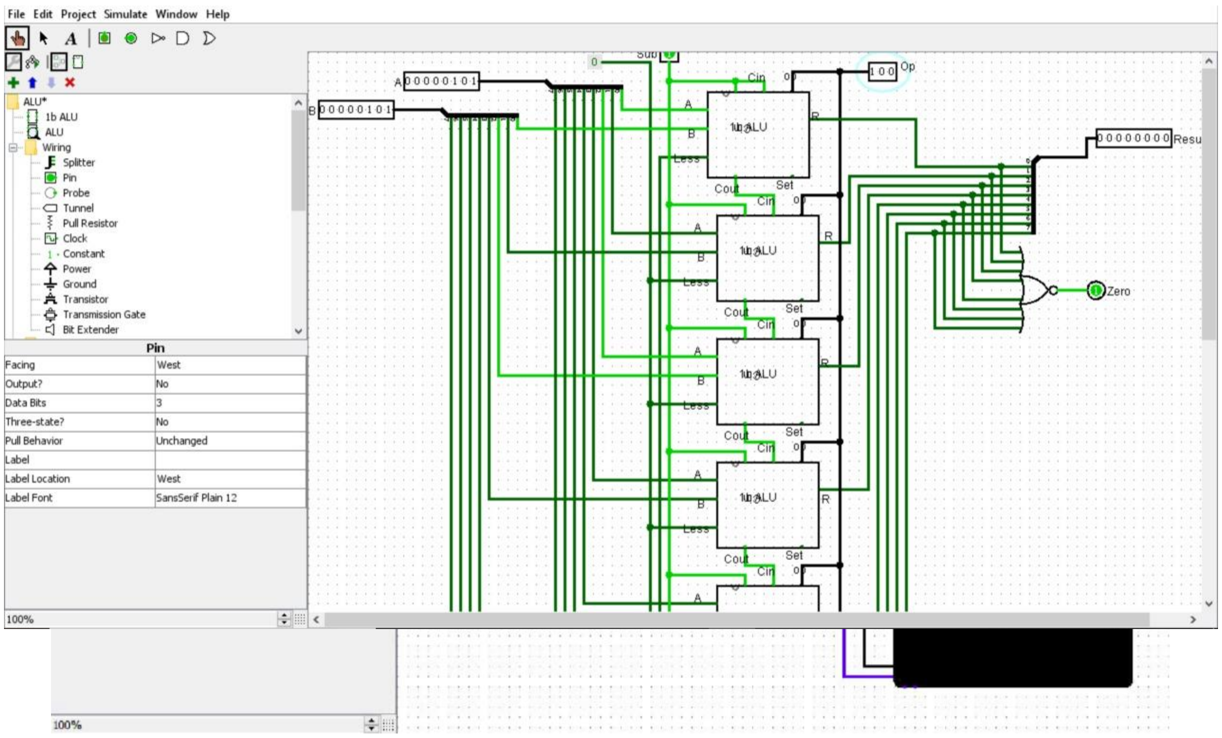
Simulator Description: Logisim is an educational tool for designing and simulating digital logic circuits. With its simple toolbar interface and simulation of circuits as you build them, it is simple enough to facilitate learning the most basic concepts related to logic circuits. With the capacity to build larger circuits from smaller sub circuits, and to draw bundles of wires with a single mouse drag, Logisim can be used (and is used) to design and simulate entire CPUs for educational purposes.

Activity to be performed by students:

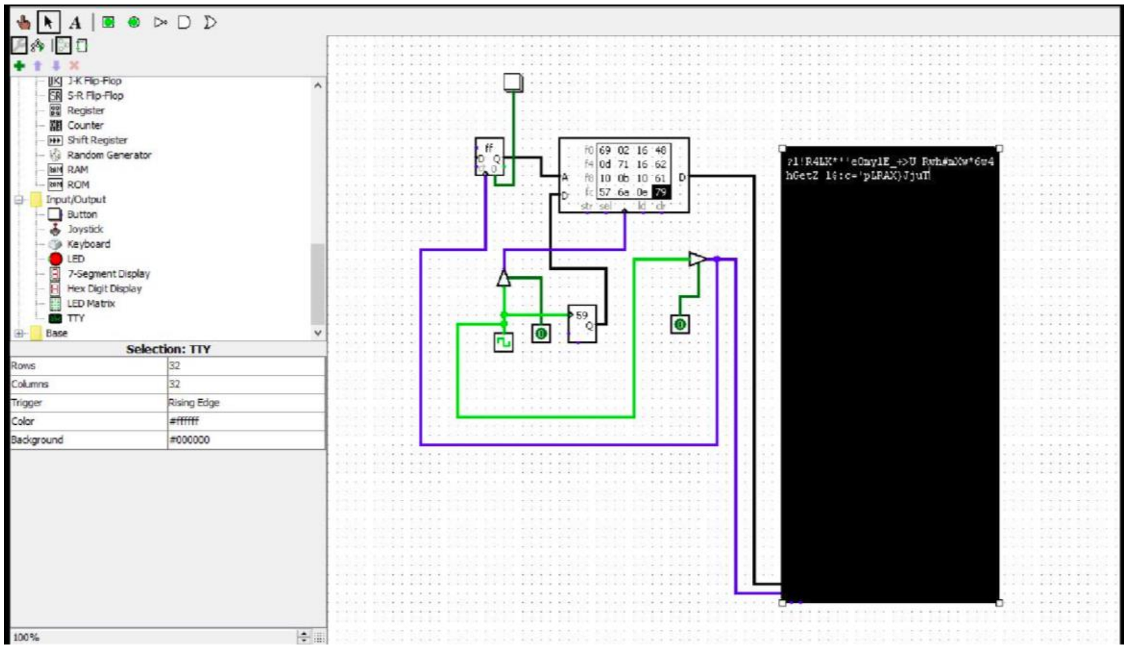


1. Add a RAM with seperate load out store selected.
2. Add a Counter and Connect Q to A of the RAM.
3. Add a Contoller buffer and Connect its output to the RAM.
4. Add a Clock and Connect to the i/p of the buffer.
5. Add a TTY unit with 32 rows and Columns, make the Connections with RAM.
6. Add a 7 bit random number generator. Connect Q to D.
7. Add another Contolled buffer, Connect it to TTY Also add an i/p to the buffer.
8. Connect the output of the second buffer to the Counter.
9. Connect a button to the Counter.

SNAPSHOTS



Pin	
Facing	West
Output?	No
Data Bits	3
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

```
?1!R4LK*''e0my1E_+>U Rwh#mXw*6w4
hGetZ 1¿:c='pLRAX}JjuT
```

Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)

Department of CSE

Programme: B.E **Term: Jan to May 2019**
Course: Computer Organization **Course Code: CS45**

Activity VII: To simulate advantages of using pipeline technique in executing a program.

Name:MK NIKHIL	**Marks: /10**	**Date:22-05-2020**
USN:1MS18CS076	**Signature of the Faculty:**	

Objective: To learn and analyze the performance of the CPU by overlapping of instructions using CPUOS-SIM simulator.

Simulator Used: CPUOS-SIM is a software development environment for the simulation of simple computers. It was developed by Dale Skrien to help users to understand computer architectures.

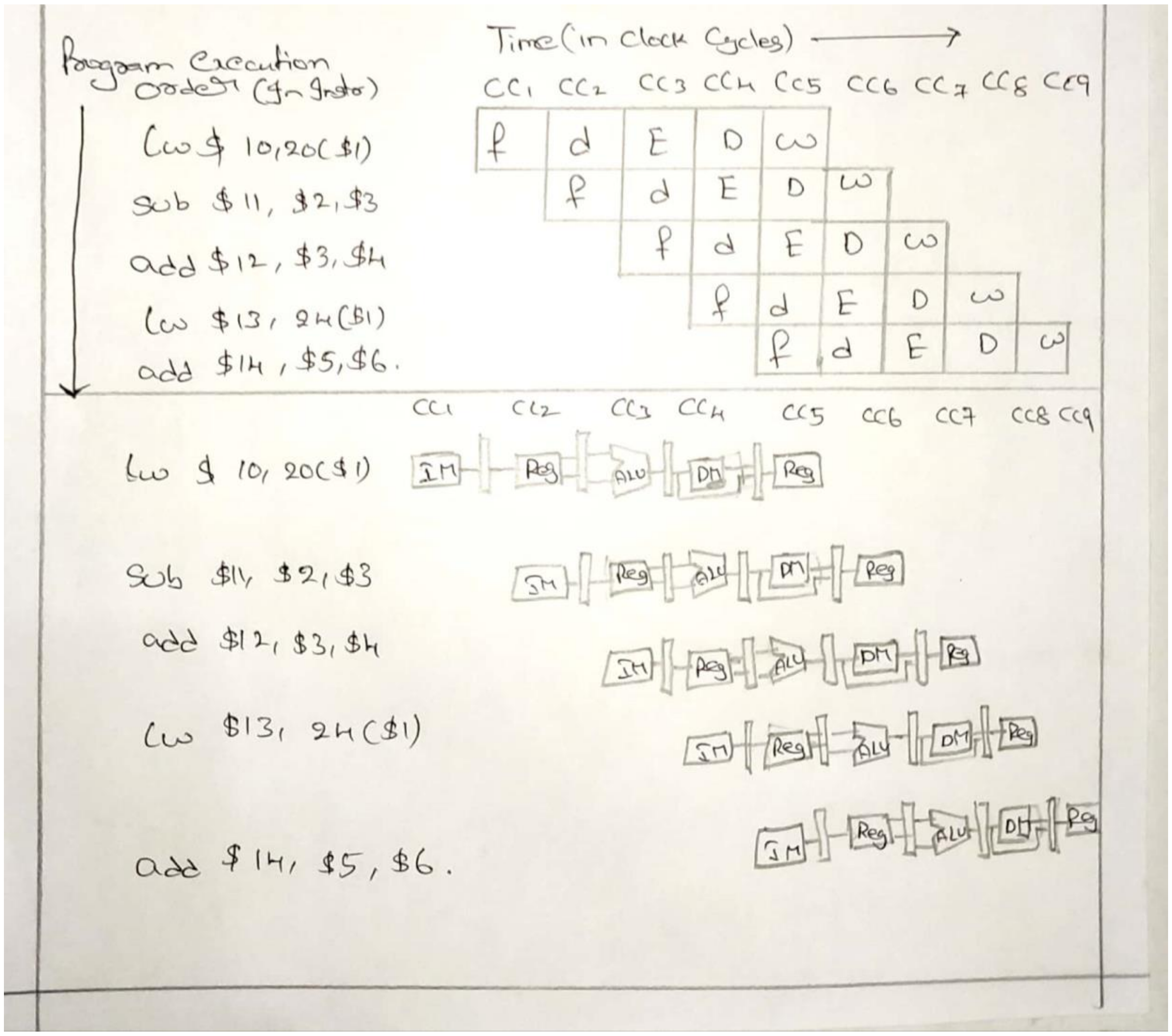
Modern CPU's contain several semi-independent circuits involved in decoding and executing each machine instruction. Separate circuit elements perform each of these typical steps:

- Fetch the next instruction from memory into an internal CPU register.
- Decode the instruction to determine which function sub-circuits it requires.
- Read any input operands required from high-speed registers or directly from memory.
- Execute the operation using the selected sub-circuits.
- Write any output results to high-speed registers or directly to memory.

Separate sections of the CPU circuitry are used for each of these steps. This allows these circuit sections to be arranged into a sequential pipeline, with the output of one step feeding into the next step.

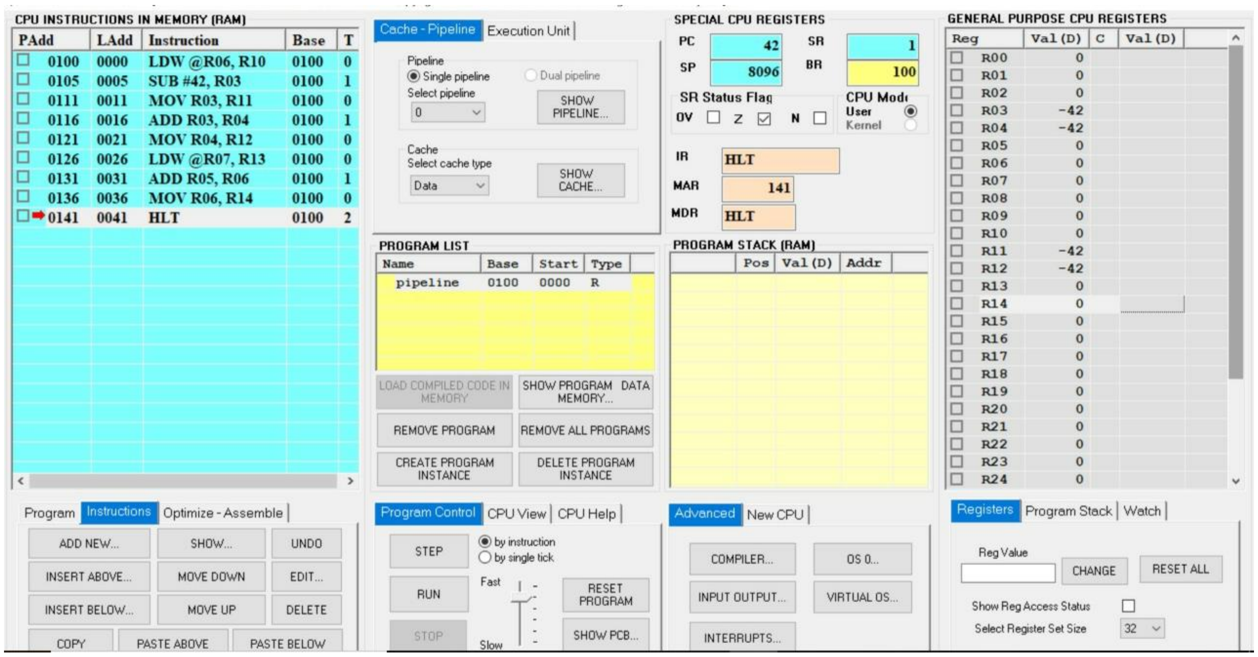
With diagram demonstrate the execution of the following instructions using pipelining technique.

lw $10,20($1)
sub $11, 42, $3
add $12, $3, $4
lw $13, 24($1)
add $14, $5, $6



	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9
lw $10,20($1)	f	d	E	D	w				
sub $11, $2, $3		f	d	E	D	w			
add $12, $3, $4			f	d	E	D	w		
lw $13, 24($1)				f	d	E	D	w	
add $14, $5, $6.					f	d	E	D	w

SNAPSHOTS



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)

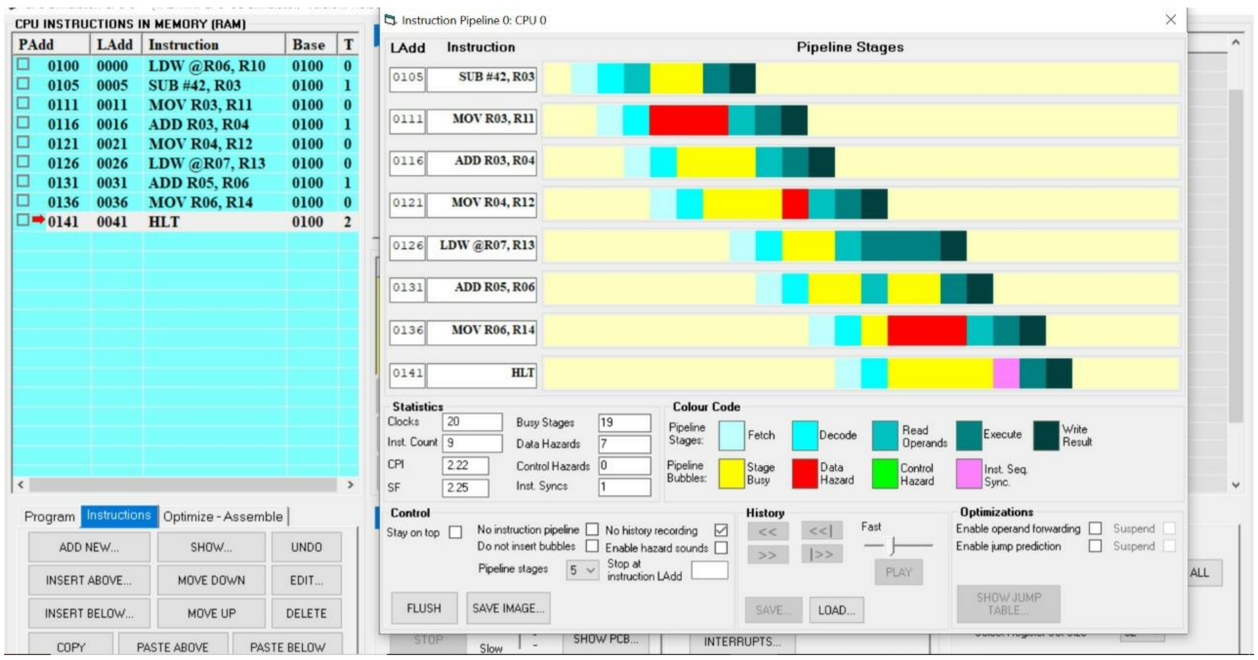
Pos	Val (D)	Addr

GENERAL PURPOSE CPU REGISTERS

Reg	Val (D)
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: STEP [✓ by instruction] by single tick

Select Register Set Size: 32



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

Instruction Pipeline 0: CPU 0

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

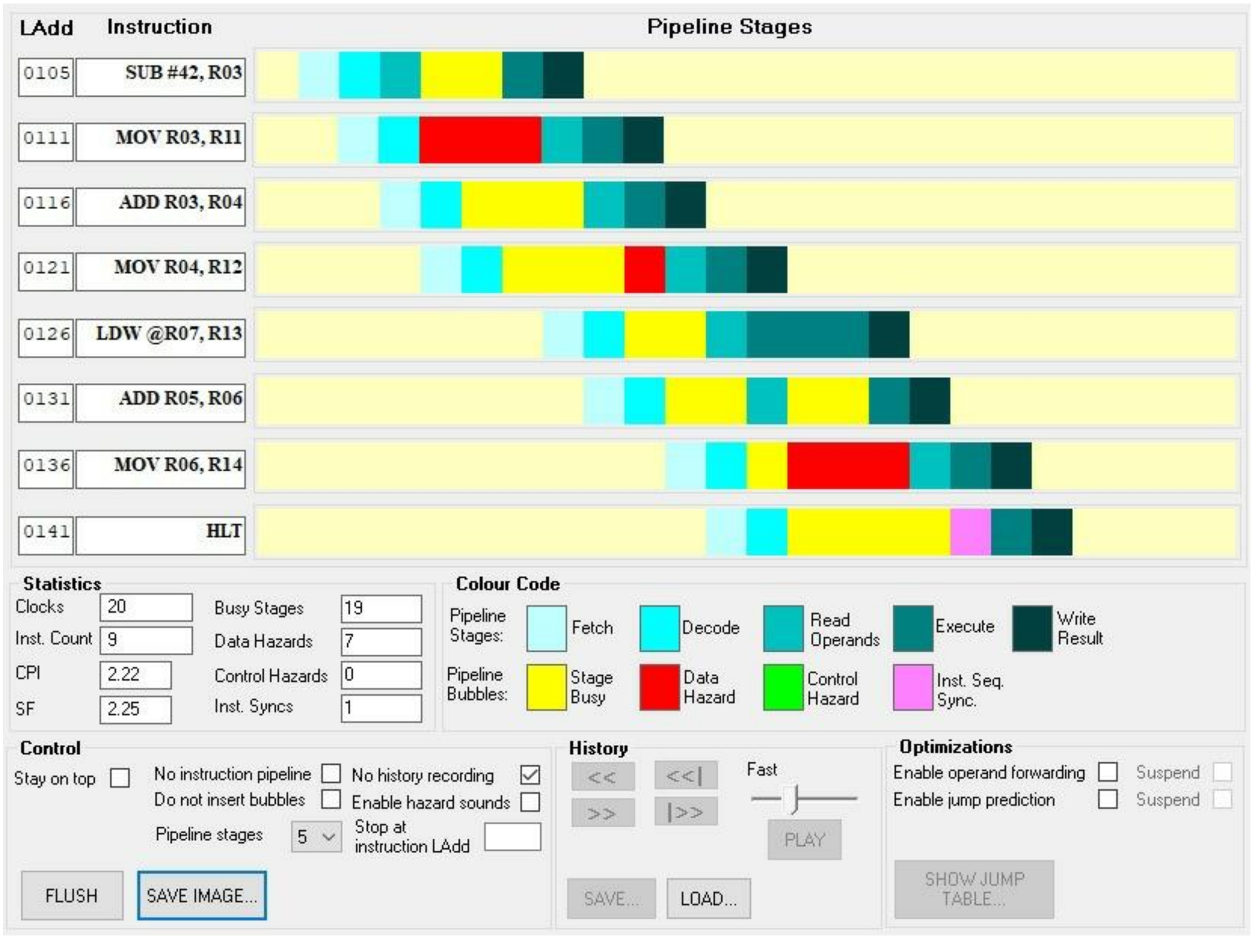
Colour Code

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 ☐; Do not insert bubbles ☐; No history recording ☑; Enable hazard sounds ☐; Pipeline stages 5; Stop at instruction LAdd

Optimizations: Enable operand forwarding ☐ Suspend; Enable jump prediction ☐ Suspend



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

Pipeline Stages

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 ☐

Do not insert bubbles ☐

No history recording ☑

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...