S.L. Karthik
IMS18CS055
IV 'B'
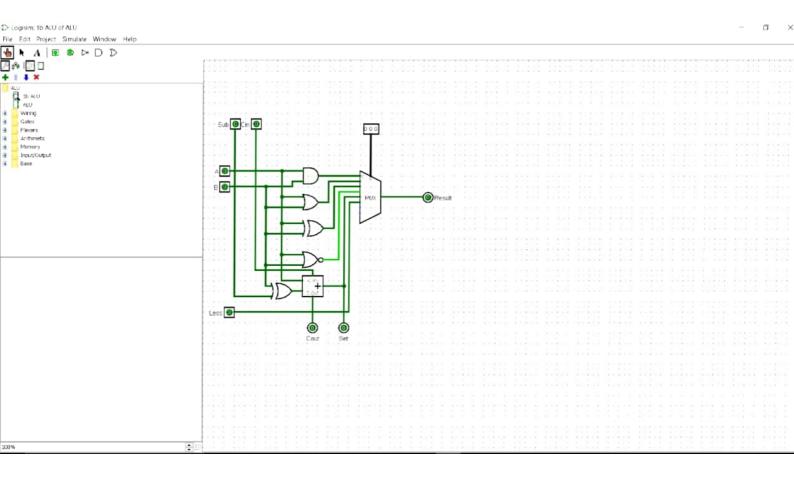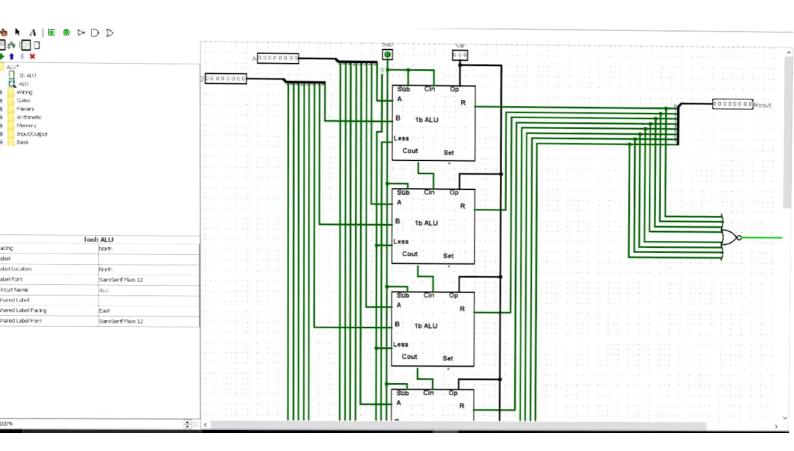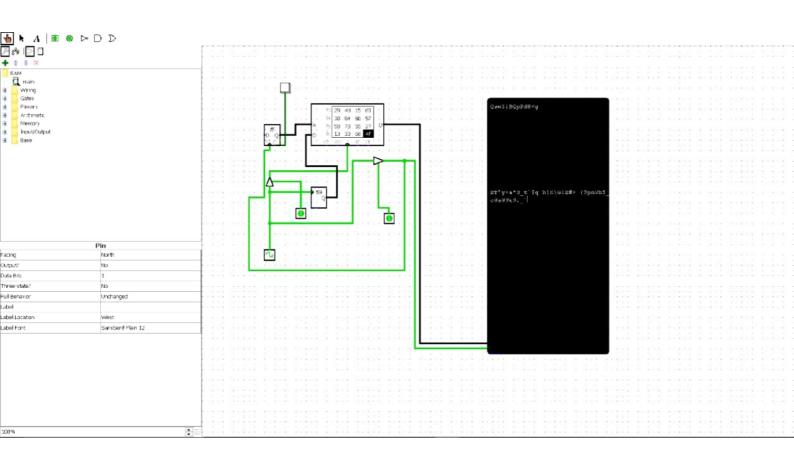
C.O. LAB - ACTIVITY V

Objective: To simulate working of ALU.

List out the steps in designing ALU

1. Add the two i/p pins. Name them A and B.
2. Add or, and, ex-or, nor gates and a 1-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 result.
5. Add a 1-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 i/p pin to C_{in}, and output pin to C_{out}.
9. Add an ex-or gate, connect its i/p to C_{out}. The first i/p must be connected to B and the second one to another input 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.

S.L. Karthik
IMS18CS055
IV   'B'

C.O. LAB - ACTIVITY VI

Objective: To simulate working operation on memory.

List out the steps in designing memory system:

1. Add a RAM with separate load and share selected.
2. Add a counter and connect Q and to A of the RAM.
3. Add a controller buffer and connect its o/p to the RAM.
4. Add a clock and connect it 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 controlled buffer, connect to TTY. Also add an i/p pin to the buffer.
8. Connect the output of the second buffer to the counter.
9. Connect a button to the counter.

S.L. Karthik
IMS18CS055
IV   B

C.O. LAB - ACTIVITY VII

Objective: To learn, analyze performance of CPU by overlapping of instructions using CPUOS simulator.

With diagram demonstrate execution of fol. instructions using pipelining

lw $10, 20($1)
sub $11, $2, $3
add $12, $3, $4
lw $13, 24($1)
add $14, $5, $6

Program execution order (in instruction) | Time (in clock cycles)

	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 access	Write back	
add $14, $5, $6					Instruction Fetch	Instruction decode	Execution	Data access	Write back

Time (in clock cycles) →

Program execution (in instruction)

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9
lw $10, 20($1)	IM	Reg	ALU	DM	Reg				
Sub $11, $2, $3		IM	Reg	ALU	DM	Reg			
add $12, $3, $4			IM	Reg	ALU	DM	Reg		
lw $13, 24($1)				IM	Reg	ALU	DM	Reg	
add $14, $5, $6					IM	Reg	ALU	DM	Reg

CPU INSTRUCTIONS IN MEMORY (RAM)

PAdd	LAdd	Instruction	Base	T
1000	0000	LDW @R04, R10	1000	0
1005	0005	SUB #42, R03	1000	1
1011	0011	MOV R03, R11	1000	0
1016	0016	ADD R03, R04	1000	1
1021	0021	MOV R04, R12	1000	0
1026	0026	LDW @R07, R13	1000	0
1031	0031	ADD R05, R06	1000	1
1036	0036	MOV R06, R14	1000	0
1041	0041	HLT	1000	2

Program | Instructions | Optimize - Assemble |

ADD NEW...	SHOW...	UND
INSERT ABOVE...	MOVE DOWN	EDIT..
INSERT BELOW...	MOVE UP	DELE
COPY	PASTE ABOVE	PASTE BELOW

Cache - Pipeline   Execution Unit

Pipeline
- (•) Single pipeline
- ( ) Dual pipeline

Select pipeline

SR Status Flag

SPECIAL CPU REGISTERS

| PC | 41 | SR | 1 |
| SP | 8096 | BR | 1000 |

CPU Mod:

GENERAL PURPOSE CPU REGISTERS

Reg	Val (D)	C	Val (D)
R03	-42		
R04	-42		
R05	0		
06	0		
7	0		
8	0		
9	0		
0	0		
1	-42		
2	-42		
3	0		
4	0		
5	0		
6	0		
7	0		
8	0		
9	0		
0	0		
1	0		
2	0		
3	0		
4	0		
5	0		
6	0		
7	0		
8	0		
9	0		
0	0		
1	0		

Instruction Pipeline 0 CPU 0

LAdd	Instruction	Pipeline Stages
1011	MOV R03, R11	
1016	ADD R03, R04	
1021	MOV R04, R12	
1026	LDW @R07, R13	
1031	ADD R05, R06	
1036	MOV R06, R14	
1041	HLT	
1041	HLT	

Statistics
Clocks	60	Busy Stages	0
Inst. Count	10	Data Hazards	0
CPI	6	Control Hazards	0
SF	.83	Inst. Syncs	2

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

Program Stack | Watch

g Value [0]   CHANGE   RESET ALL

w Reg Access Status [ ]

Select Register Set Size [32 ▾]

STOP   Slow | SHOW PCB...   INTERRUPTS...