CSN-221

ARM Processor

Problem statement

The basic idea of this project was to implement a CPU that includes ALU , register files, control circuitry, instructions flow etc. on Logisim simulator. Rigorous benchmark evaluation on different instructions has to be performed while working on this project.

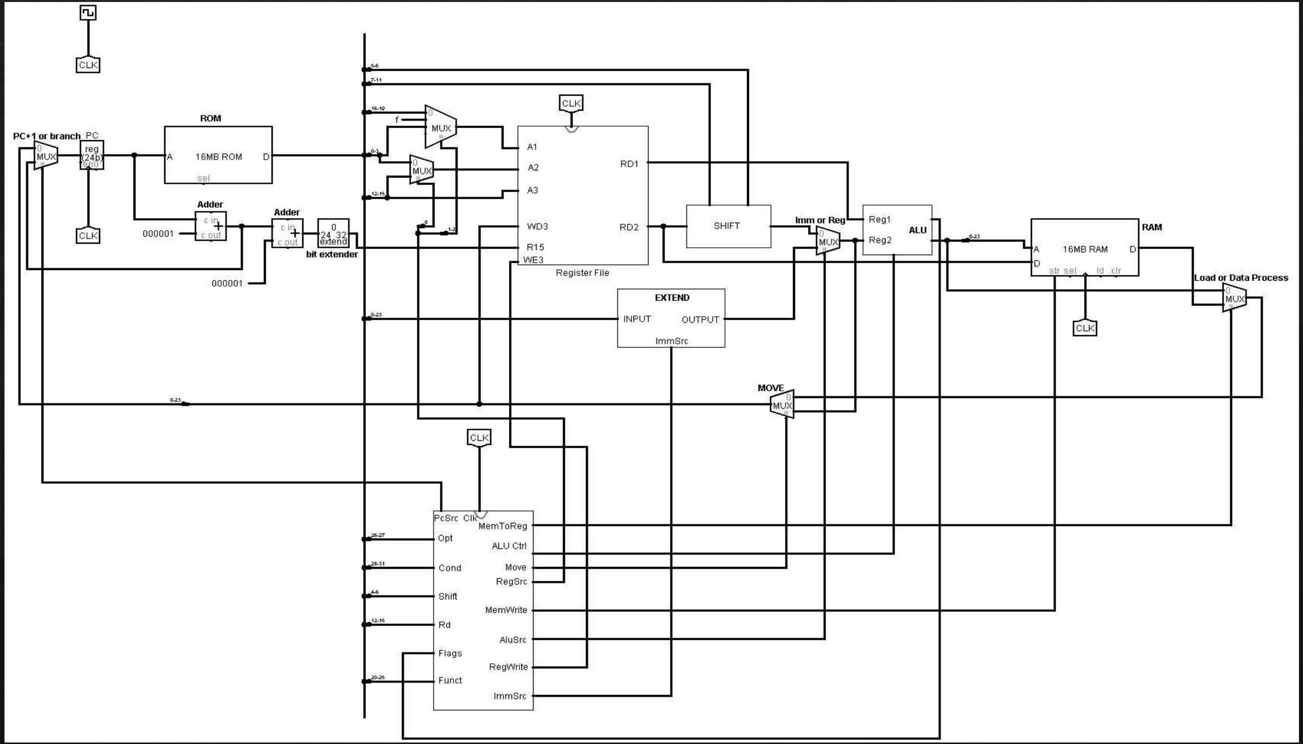
Our Solution

We designed a CPU using the ARM instruction set. Different components of the CPU such as ALU (Arithmetic and Logic unit), control unit, instruction memory, data memory were designed in the Logisim simulator. Also, benchmark evaluations were done regularly in the Logisim simulator itself by testing the processor on some custom instructions.

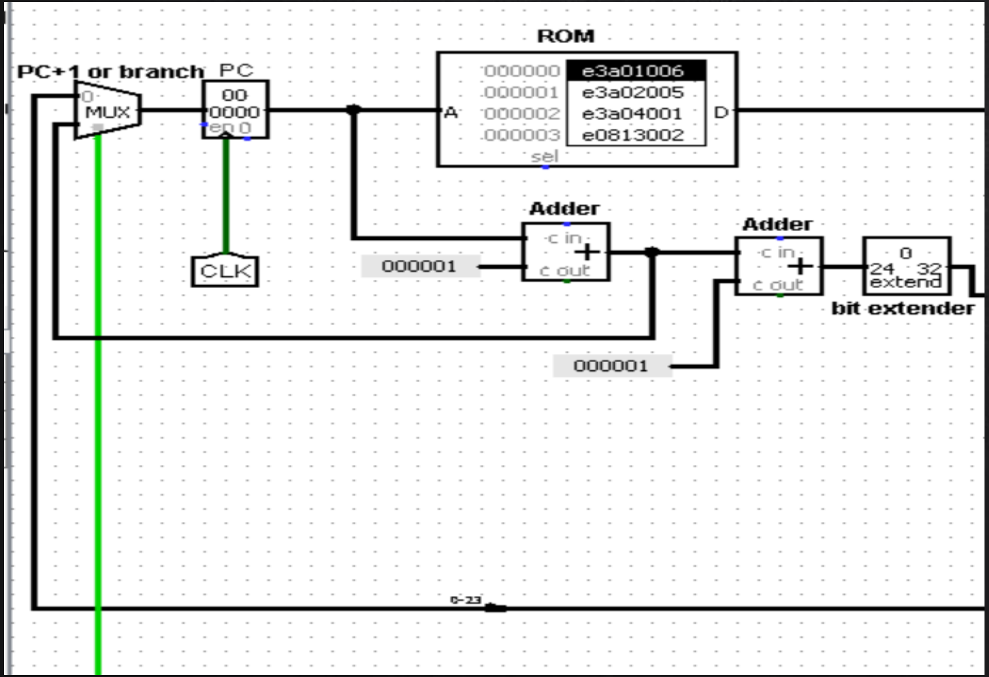
The following report describes the various Architectural implementations and the flow of instructions and data with all the diagrams being made by us as Drawings or Screenshots of our implementation to help describe the same.

The Novelty of Work Done

Main Circuit



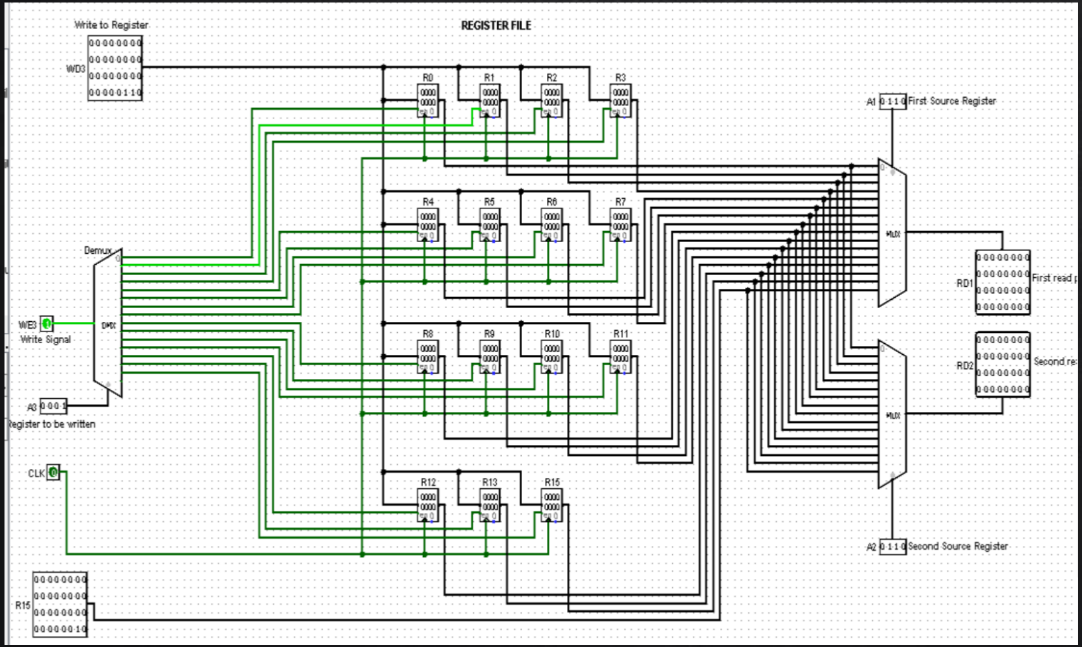
Program Counter And Instruction Memory (ROM)



- ➢ Controls flow of program
- ➢ Uses a multiplexer to decide the next instruction

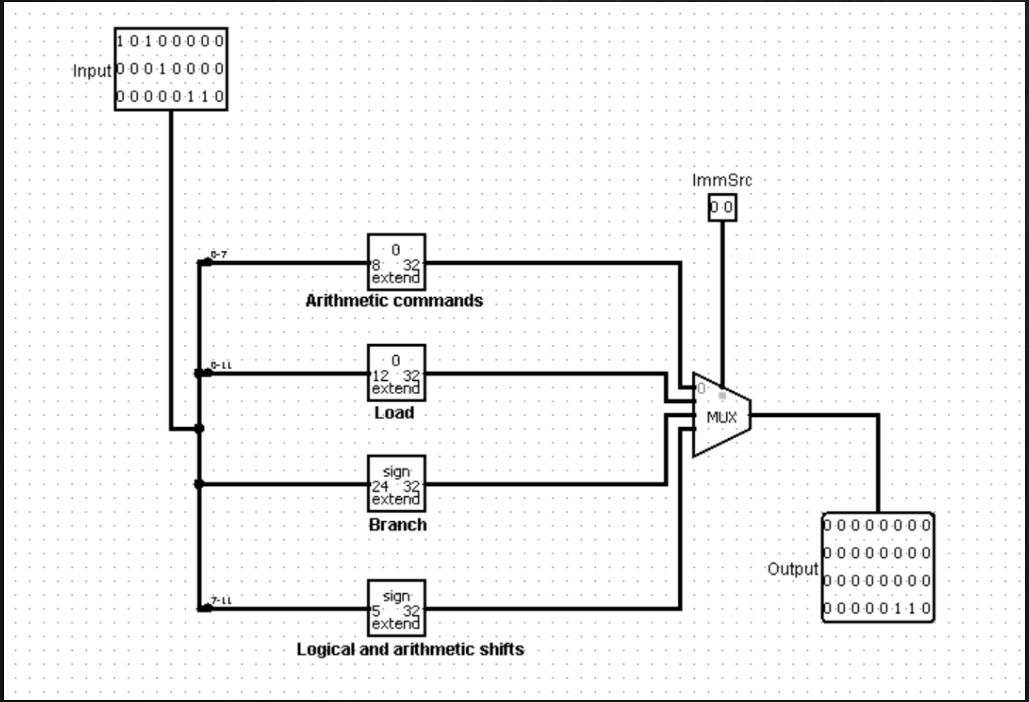
The ROM holds the instructions used for benchmark evaluations.

Register File



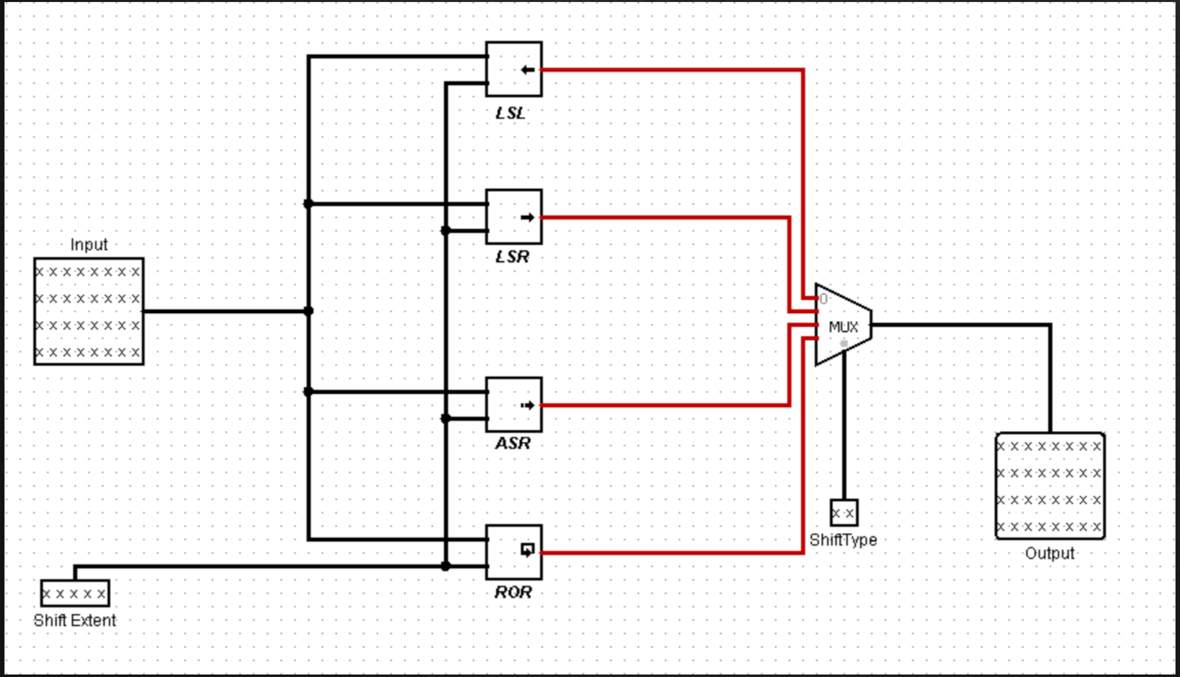
- Consist of several ports used for accessing the registers.
- WD3 brings the data to be written in the register files.

Extender



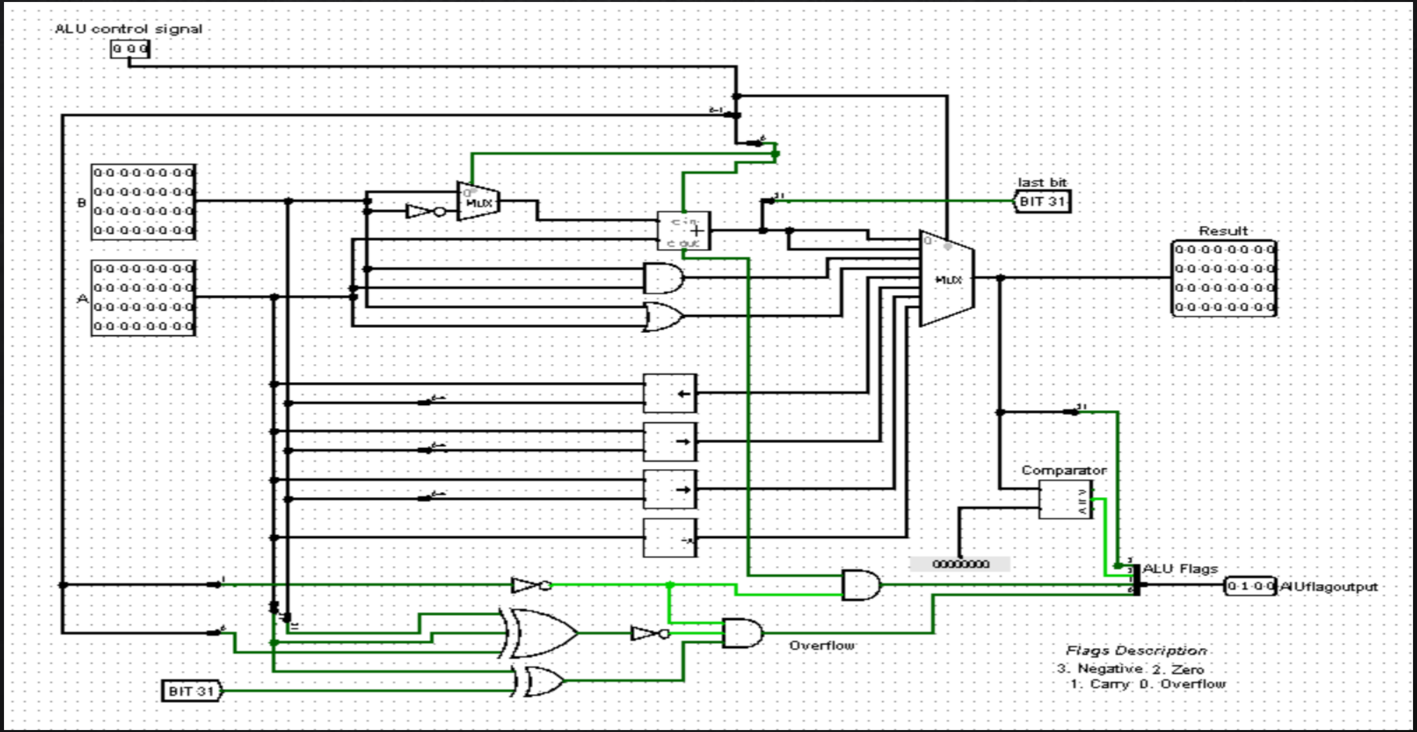
Extends the immediate according to the **ImmSrc** signal

Shifter



If the second source might have some shift associated with it, thus we pass the value through a shifter

ALU


ALU control signal
000
B
A
MUX
last bit
BIT 31
Result
MUX
Comparator
00000000
ALU Flags
0100
ALUflagoutput
Overflow
BIT 31
Flags Description
3. Negative 2. Zero
1. Carry 0. Overflow

Instruction Encoding

General Format

Cond	Type	
32 29	28 27	

Data Processing Instruction

Cond	00	I	opcode	S	rs	rd	Shifter operand/ immediate
32 29	28 27	26	25 22	21	20 17	16 13	12 0

Instruction Encoding

Encoding shifter operand

Shift immediate	type	0	rt
12 8	7 6	5	4 1

Shift immediate		type	1	rt
12 9		7 6	5	4 1

Instruction Encoding

Load And Store Instruction

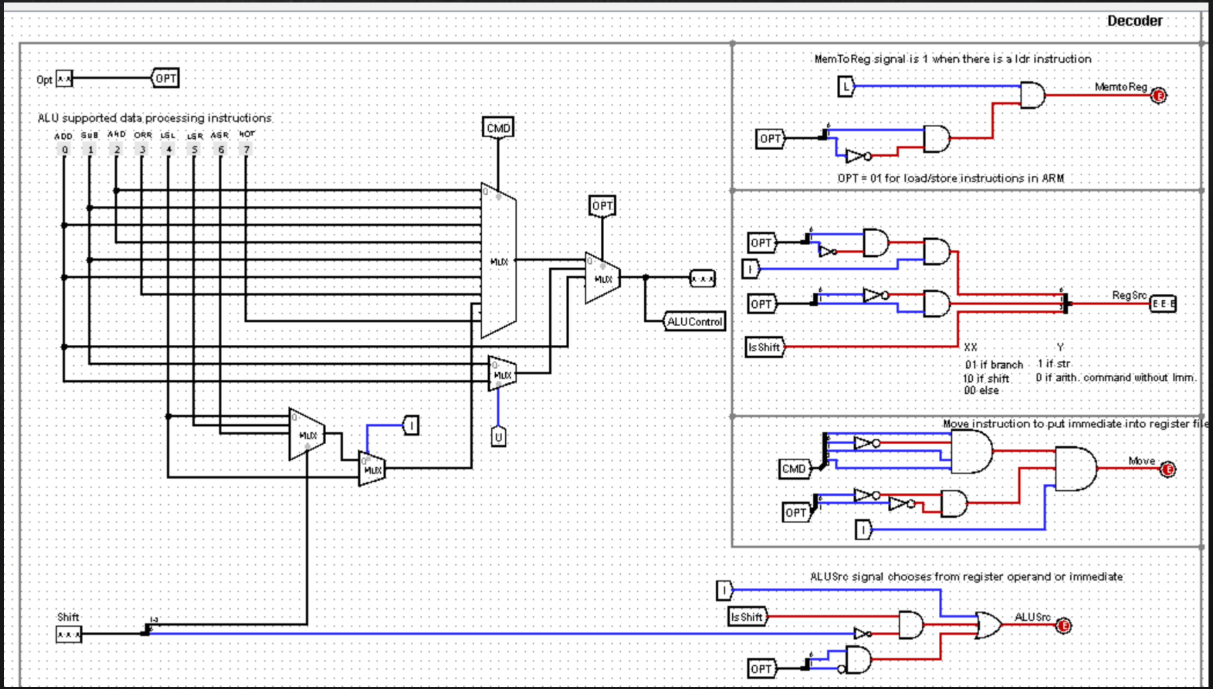
COND	01	I	P	U	B	W	L	RS	RD	Shifter operand /immediate
32 29	28 27	26	25	24	23	22	21	20 17	16 13	12 0

Branch Instruction

COND	101	L	OFFSET
32 29	28 26	25	24 0

Control Unit

Decoder



ALU supported data processing instructions

MemToReg signal is 1 when there is a ldr instruction

OPT = 01 for load/store instructions in ARM

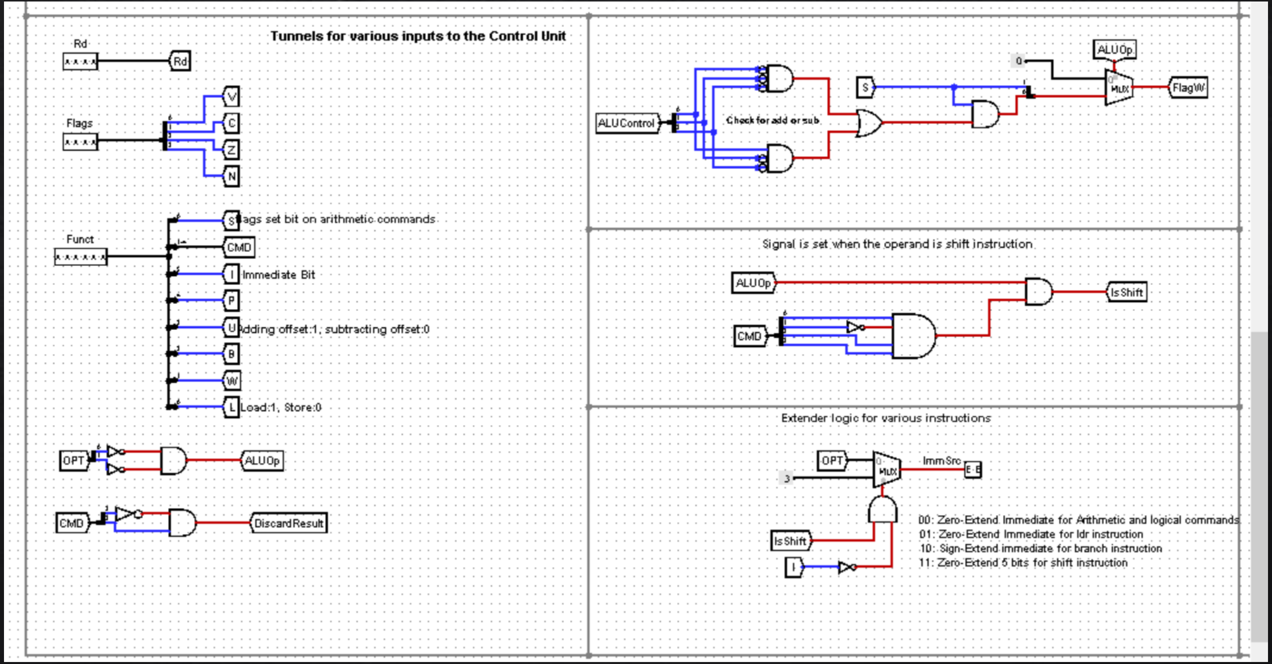
XX
01 if branch
10 if shift
00 else

Y
1 if str
0 if arith. command without Imm.

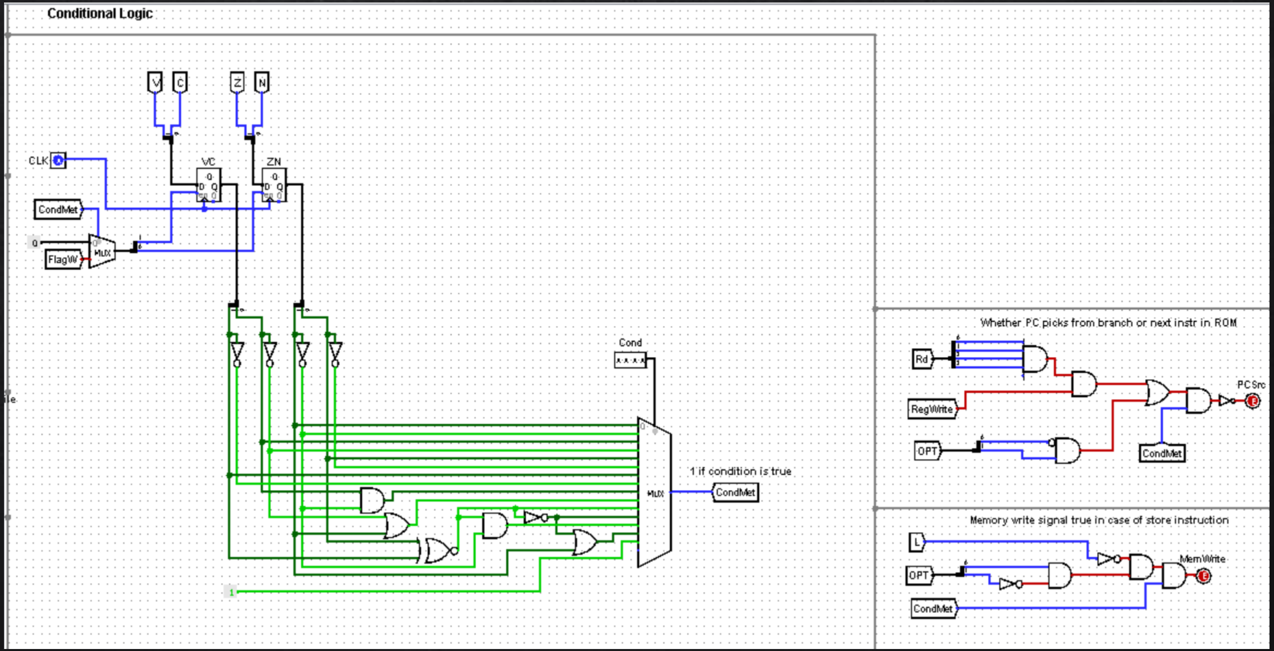
Move instruction to put immediate into register file

ALUSrc signal chooses from register operand or immediate

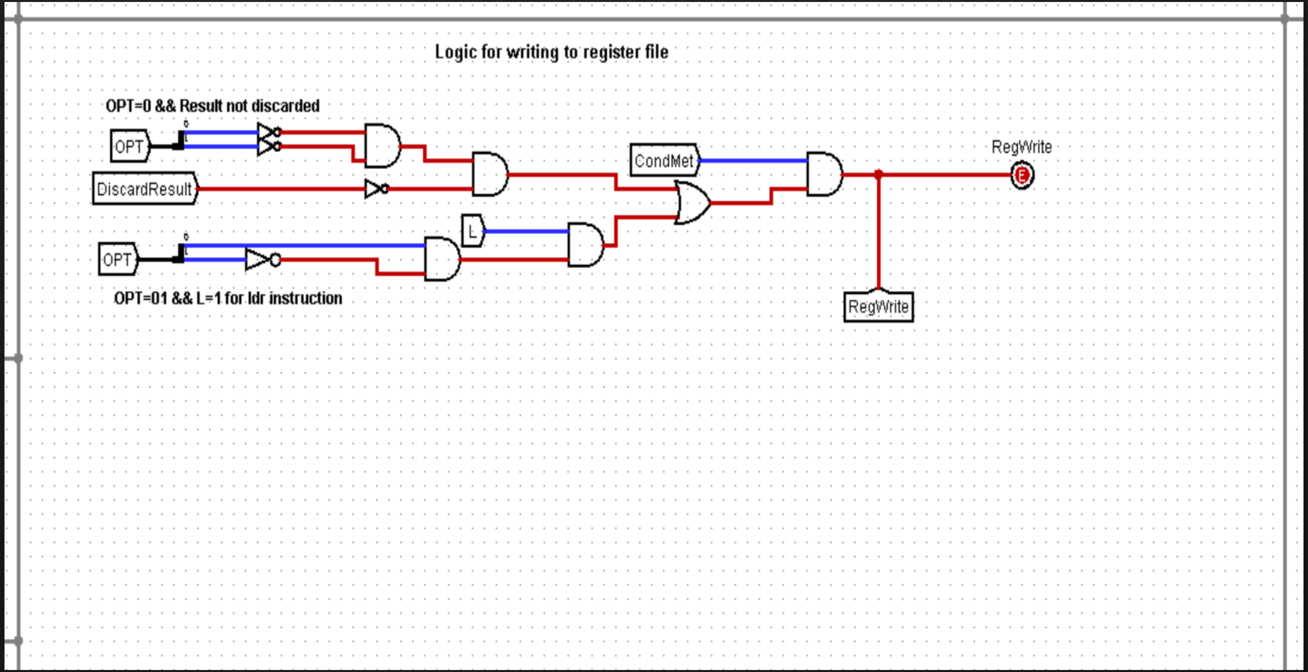
Decoder


Tunnels for various inputs to the Control Unit
Rd
Flags
V
C
Z
N
Funct
Flags set bit on arithmetic commands
CMD
Immediate Bit
P
Adding offset:1; subtracting offset:0
B
W
Load:1, Store:0
OPT
ALUOp
DiscardResult
ALUControl
Check for add or sub
S
FlagW
MUX
Signal is set when the operand is shift instruction
IsShift
Extender logic for various instructions
ImmSrc
00: Zero-Extend Immediate for Arithmetic and logical commands
01: Zero-Extend Immediate for ldr instruction
10: Sign-Extend immediate for branch instruction
11: Zero-Extend 5 bits for shift instruction

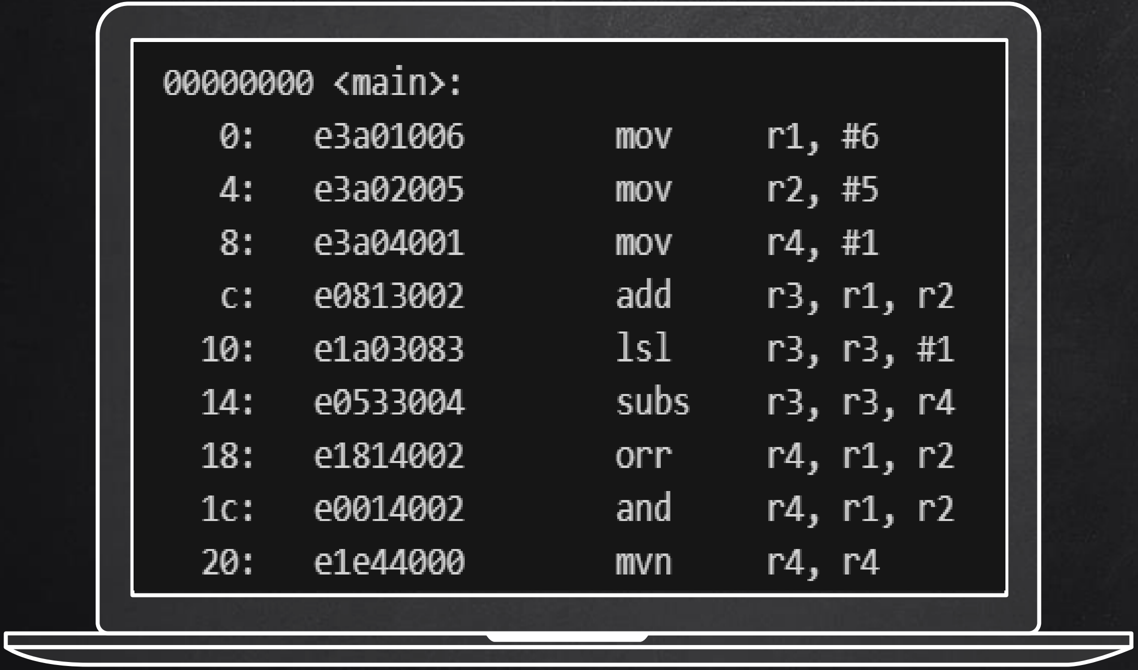
Conditional Logic



Conditional Logic

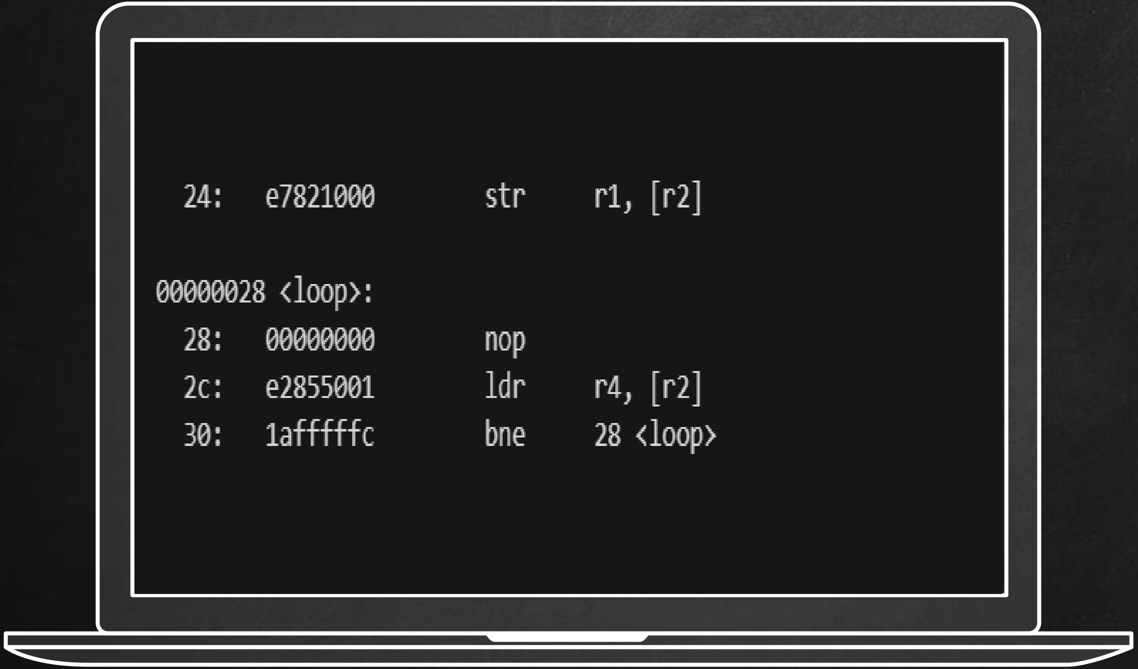

Logic for writing to register file
OPT=0 && Result not discarded
OPT
DiscardResult
OPT
OPT=01 && L=1 for ldr instruction
L
CondMet
RegWrite
RegWrite

Tests For Data Transfer Instructions



```
00000000 <main>:
   0:   e3a01006        mov     r1, #6
   4:   e3a02005        mov     r2, #5
   8:   e3a04001        mov     r4, #1
   c:   e0813002        add     r3, r1, r2
  10:   e1a03083        lsl     r3, r3, #1
  14:   e0533004        subs    r3, r3, r4
  18:   e1814002        orr     r4, r1, r2
  1c:   e0014002        and     r4, r1, r2
  20:   e1e44000        mvn     r4, r4
```

[redacted]



```
  24:   e7821000        str     r1, [r2]

00000028 <loop>:
  28:   00000000        nop
  2c:   e2855001        ldr     r4, [r2]
  30:   1afffffc        bne     28 <loop>
```

Tests For Load Store And Branch Instructions

Conclusion

Thus, we have designed an ARM processor with all the essential features and working. Our processor is capable of executing the following instructions in 24-bit format with 32-bit data:

- Branch Instructions
- LOAD & STORE Instructions
- ALU Instructions:-

• AND	• AND
• OR	• LSR
• SUB	• ASR
• NOT	• LSL

THANKS!