

MICRO L'AFENTIKA

The Best

Outline

01

Instruction Set Architecture

02

ALU Operations

03

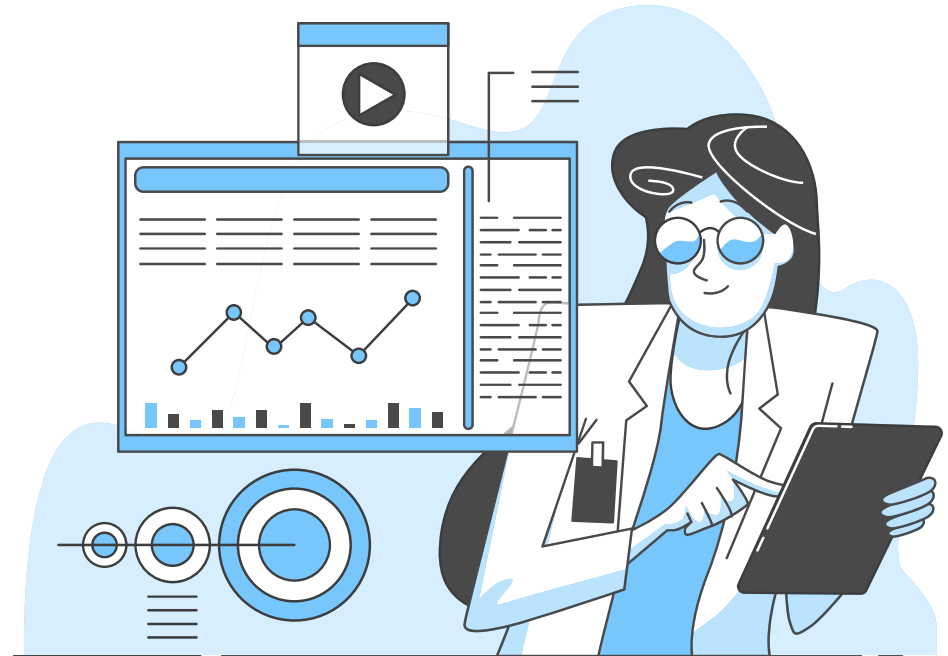
Memory Instructions

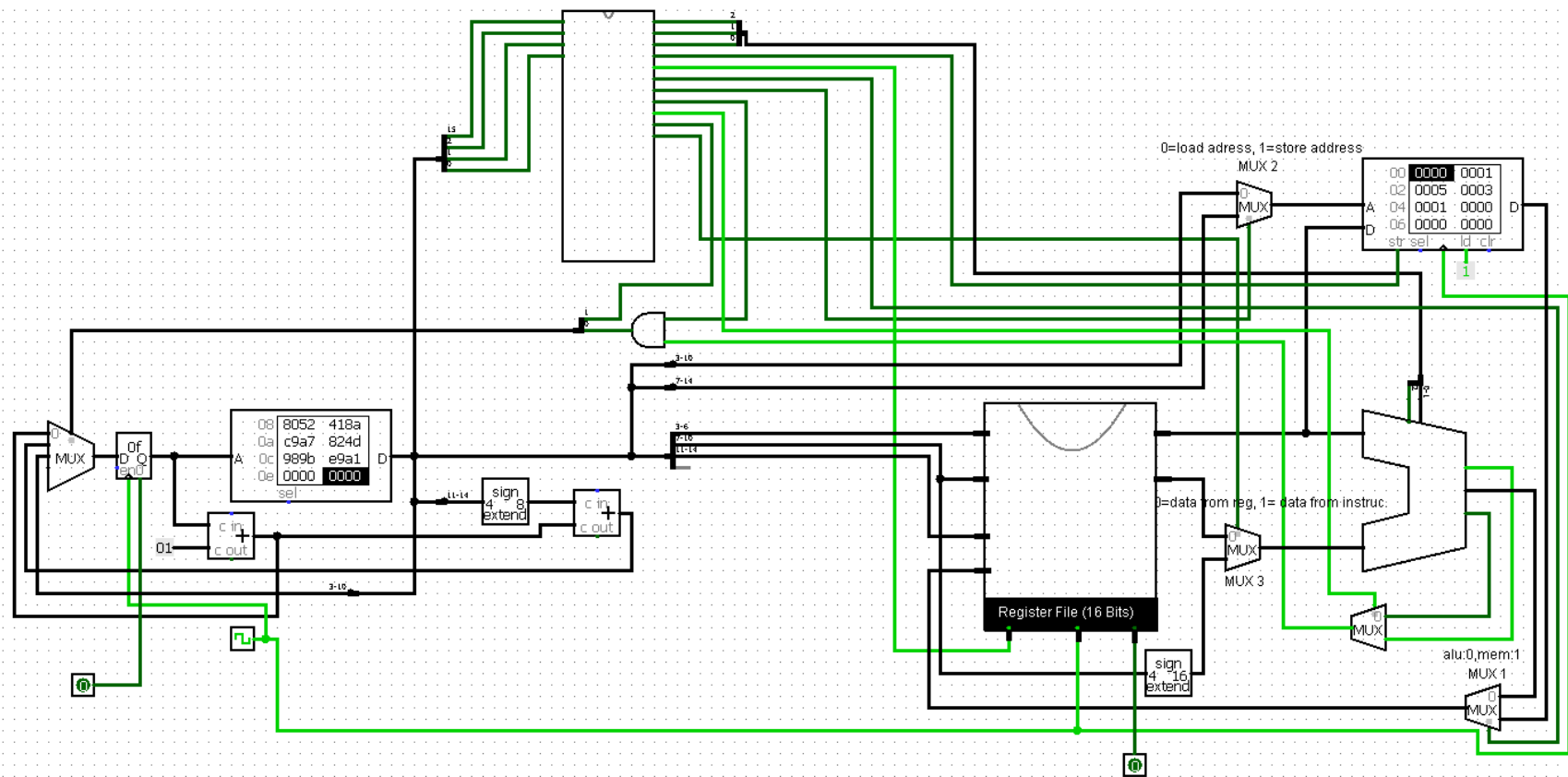
04

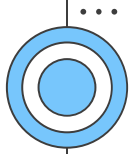
Flow Control Instructions

05

Data Paths







Instruction Set Architecture

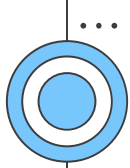
- Micro L'Afentika is a 16-bit microprocessor
- 16 registers
- It is interfaced with a data memory that has 2^8 addressable locations (2 bytes each)
- It operates on 16-bit instructions
- It can perform 16 distinct instructions that spread across 3 categories
- We will look at that in a bit (1,0)



...



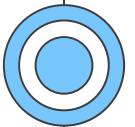
...



The Instructions and Their Opcodes

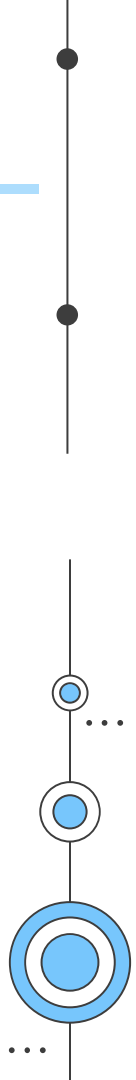
Add	Subtract	Multiply	Divide	AND	OR	XOR	Negate
0000	0001	0010	0011	0100	0101	0110	0111

Beq	Bne	Jump	ADDi	load	store	ORi	ANDi
1000	1001	1010	1011	1100	1101	1110	1111





Control Unit

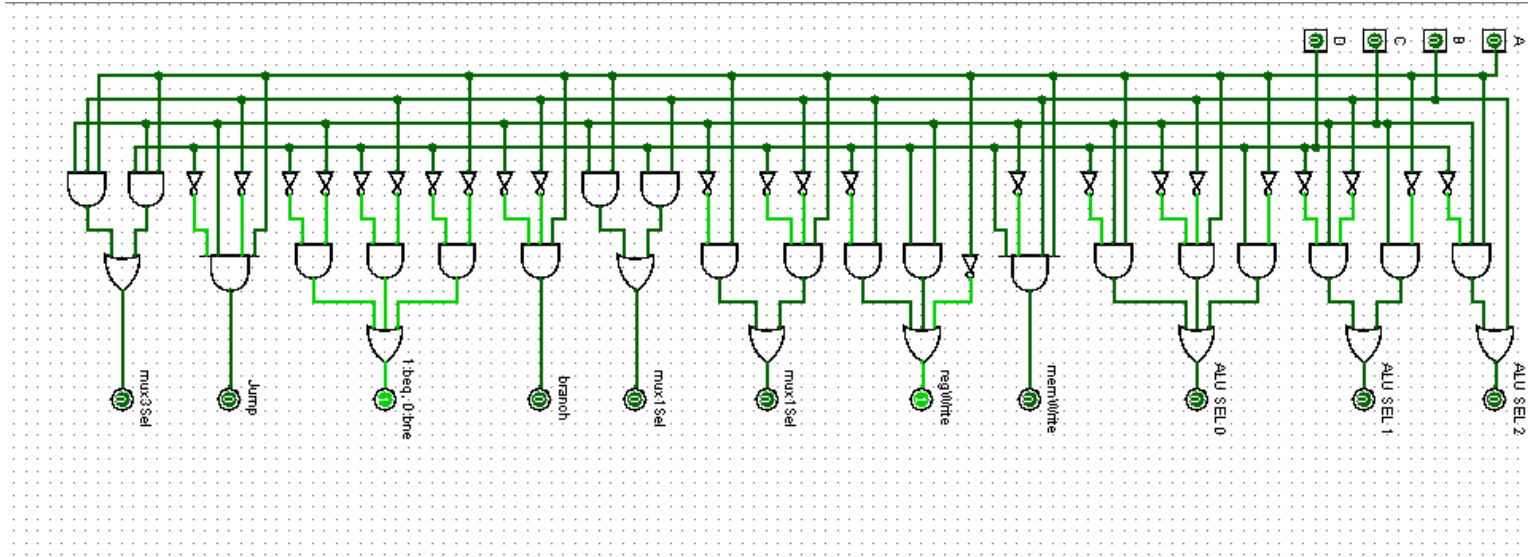
1. The CU is the “brain” of the brain, to put it.
 2. It gives signals and controls what exactly happens when instructions are executed
 3. The control unit takes bits 0, 1, 2 and 15 (opcodes) of all instructions and generate appropriate signals to control the following:
 - ALU Operations
 - Register Write Enable
 - Memory Write Enable
 - MUX Select Controls
 - Flow control signals
- 

Control Unit

1. Truth table for creating the circuit of the control unit:

A	B	C	D	ALUSEL2	ALUSEL1	ALUSEL0	memWrite	regWrite	mux1Sel	mux1Sel2	branch	beq0bne1	Jump	mux3Sel
0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	1	0	1	0	0	0	0	0	0
0	0	1	0	0	1	0	0	1	0	0	0	1	0	0
0	0	1	1	0	1	1	0	1	0	0	0	0	0	0
0	1	0	0	1	0	0	0	1	0	0	0	1	0	0
0	1	0	1	1	0	1	0	1	0	1	0	0	0	0
0	1	1	0	1	1	0	0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	0	1	0	1	0	0	0	0
1	0	0	0	0	0	1	0	0	1	0	1	1	0	0
1	0	0	1	0	0	1	0	0	1	0	1	0	0	0
1	0	1	0	1	1	1	0	0	1	1	0	1	1	0
1	0	1	1	0	0	0	0	1	0	1	0	0	0	1
1	1	0	0	1	0	0	0	1	1	0	0	1	0	0
1	1	0	1	1	0	0	1	0	1	1	0	0	0	0
1	1	1	0	1	0	1	0	1	0	1	0	0	0	1
1	1	1	1	1	0	0	0	1	0	1	0	0	0	1

Hardware Realization of the CU



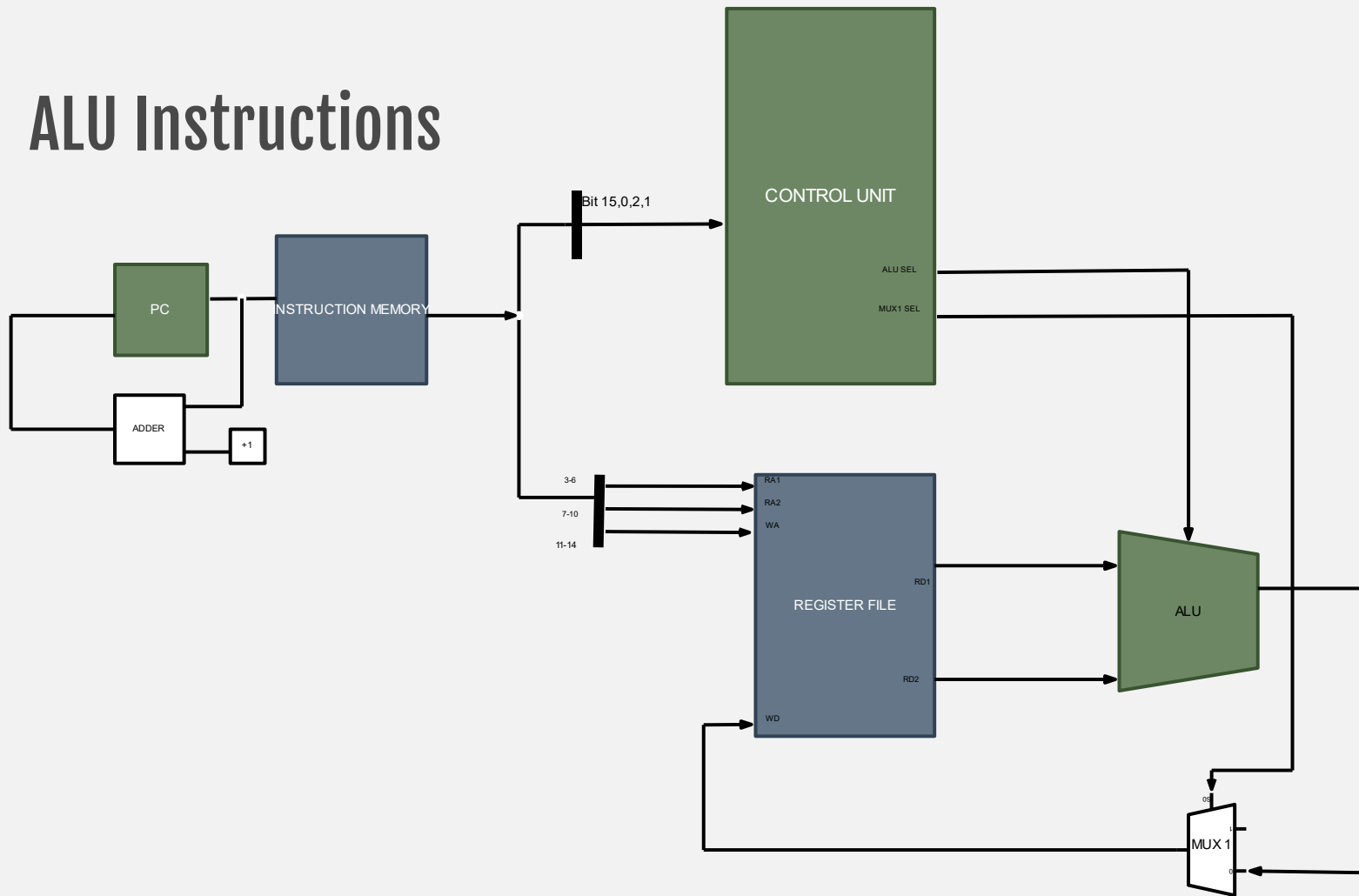
General ALU Operations

(With only Register Locations)

1. All ALU instructions have 4 different parts
 - The opcode
 - 3 register locations specified with 4 bits each
2. The format

1 Bit	4 bits	4 bits	4 bits	3 bits
Opcode	Destination	Source 2	Source 1	Opcode
15	14:11	10:7	6:3	2:0

ALU Instructions



ALU Instructions with Immediate

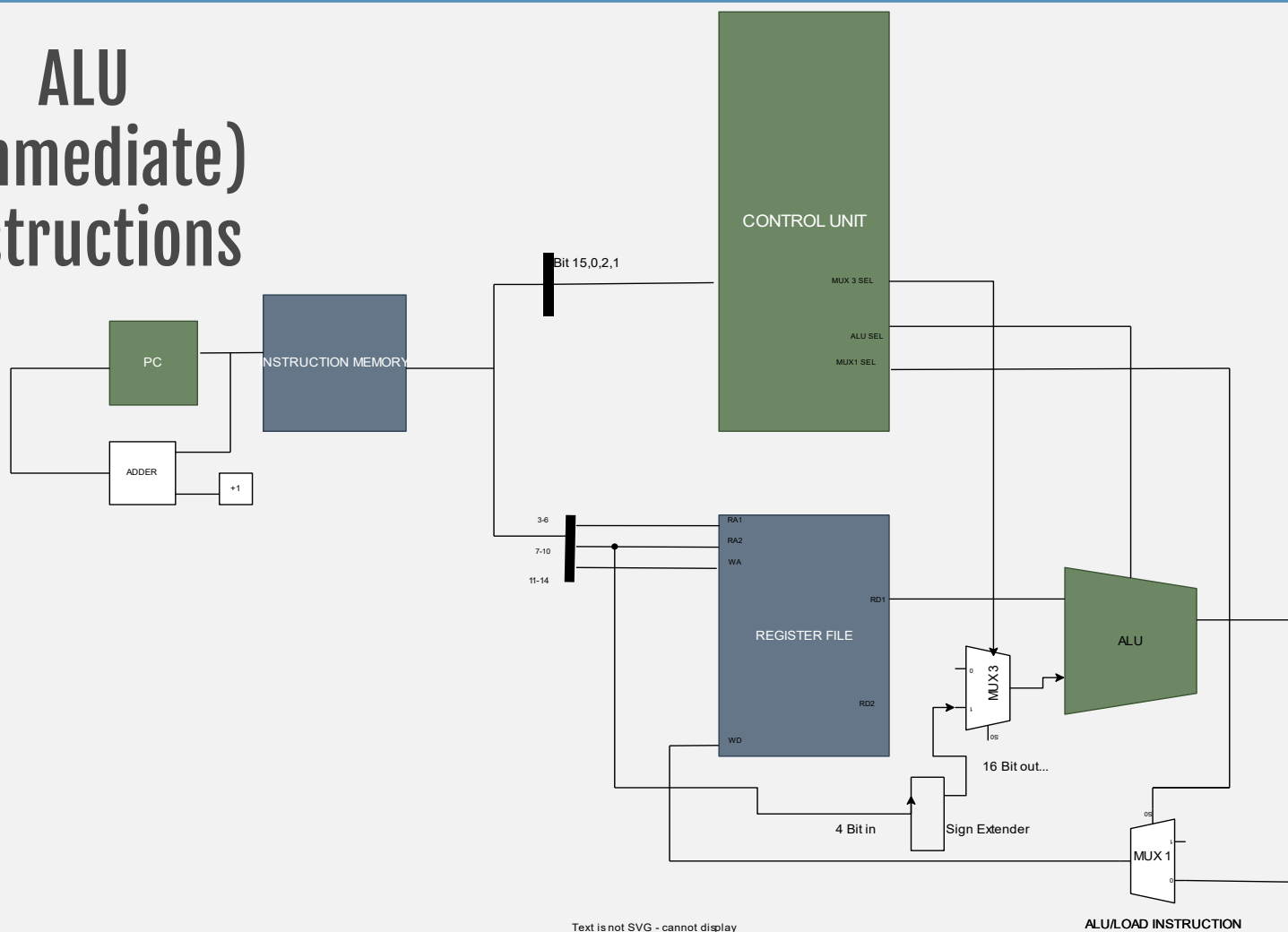
1. These allow the user to also enter at most one operand as part of the instructions
2. This is possible for only three ALU operations:
 - AND
 - OR
 - ADD

3. Instruction Format

Bits	15	14:11	10:7	6:3	2:0
Description	Opcode	Destination	Immediate	Source Operand 1	Opcode
Number of bits	1	4	4	4	3

ANDi	1111	ORi	1110	ADDi	1011
-------------	-------------	------------	-------------	-------------	-------------

ALU (Immediate) Instructions



ALU/LOAD INSTRUCTION

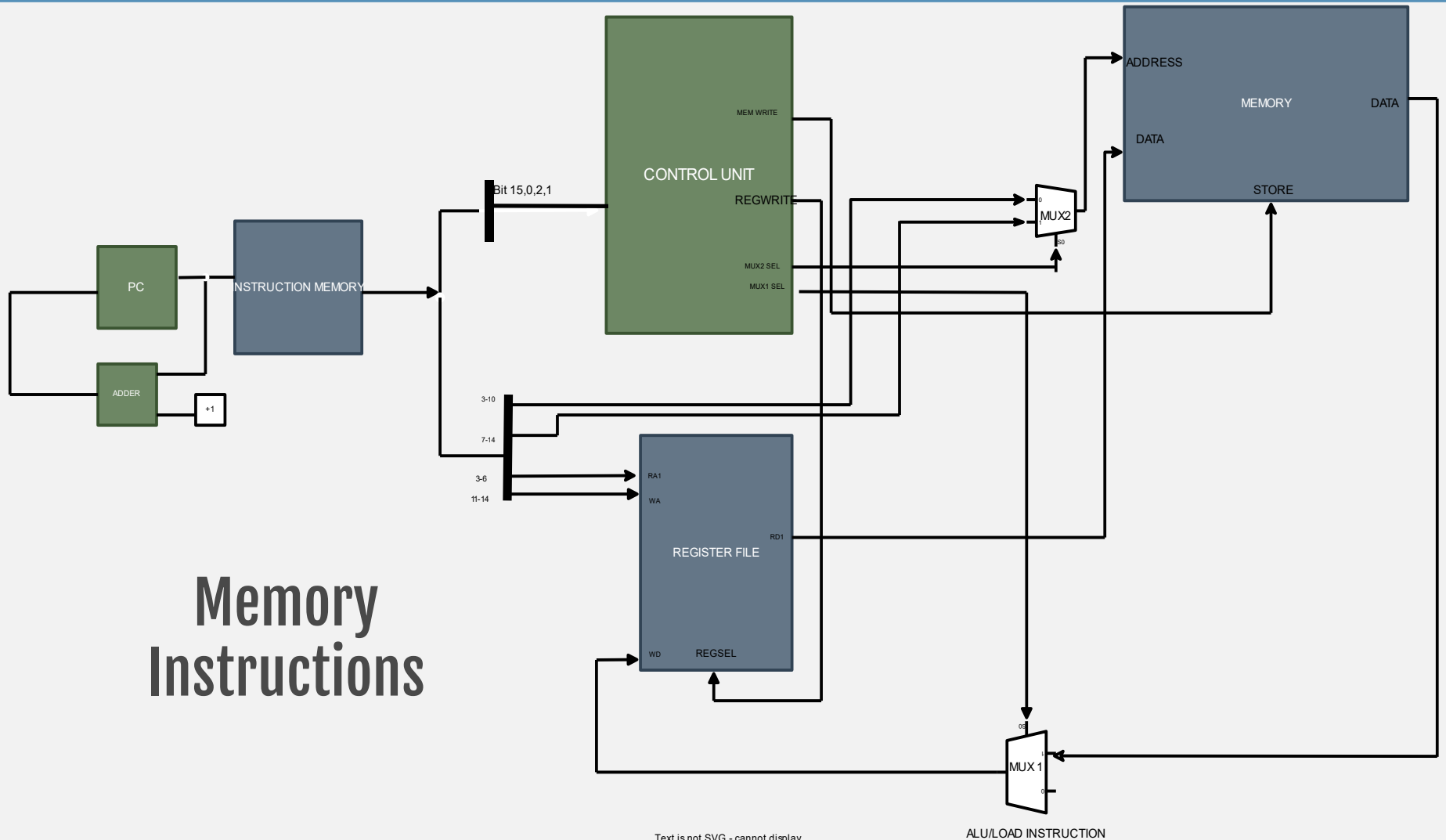
Memory Instructions

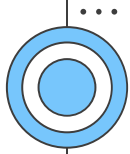
1. There are two instructions that allow the microprocessor to communicate with memory
2. The memory instructions are
 - Load instruction: moving data from memory to register
 - Store instruction: moving data from register to memory
3. Instruction format of memory instructions:

Bits	15	14:11	10:3	2:0
Description	Opcode	Register (Destination)	Memory Address (Source)	Opcode
Number of bits	1	4	8	3

Bits	15	14:7	6:3	2:0
Description	Opcode	Memory Address (Destination)	Register (Source)	Opcode
Number of bits	1	8	4	3

Memory Instructions





Flow Control Instructions

1. There are two major instructions used to control the flow of the instructions
 - Branch Instructions
 - Unconditional Jump instruction
- 2.



...



...

Flow Control Instructions

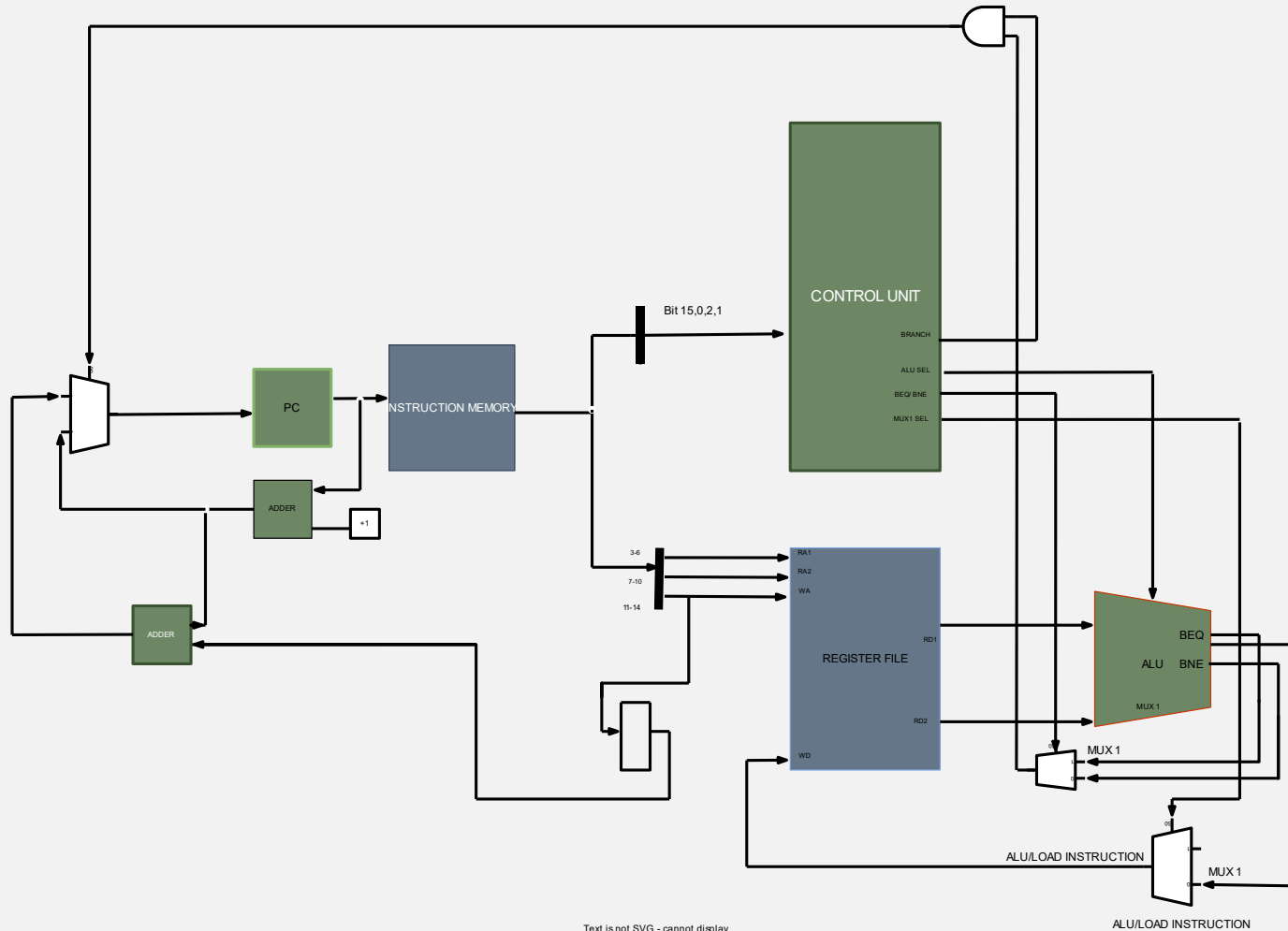
Branch Operations

The branch operations take two operands and a 4 bit program counter offset.

1. Branch Equal (Beq): if the operands are equal
2. Branch Not Equal (Bne): if the operands are not equal

Bits	15	14:11	10:7	6:3	2:0
Description	Opcode	Offset (value to add to (PC + 1))	Source Operand 2	Source Operand 1	Opcode
Number of bits	1	4	4	4	3

Branch Instructions



Text is not SVG - cannot display

ALU/LOAD INSTRUCTION

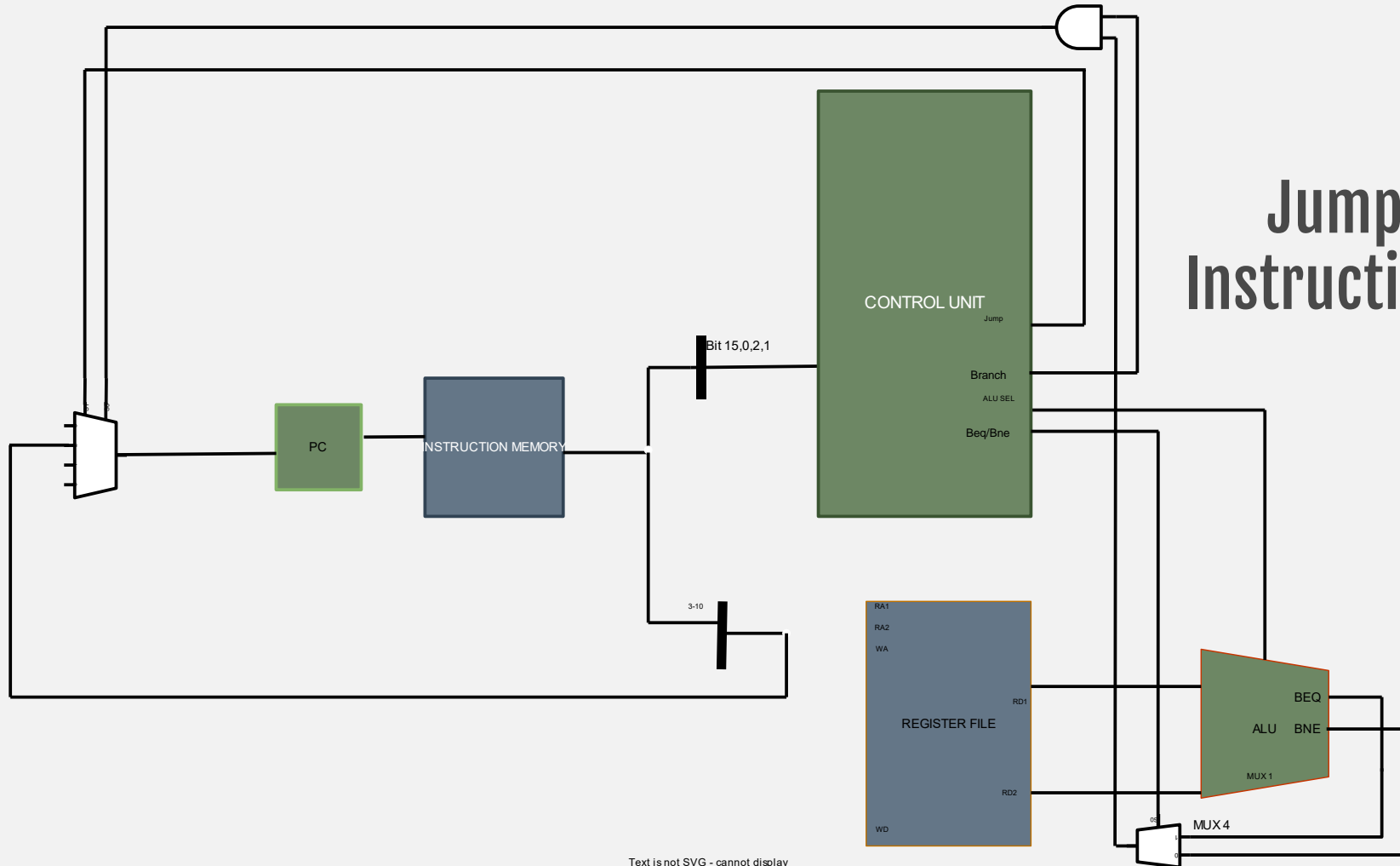
Flow Control Instructions

Jump Operation

1. The Jump operations allow the program counter to point to a specific address of the instruction memory.
2. This allows the program to skip over some number of instructions

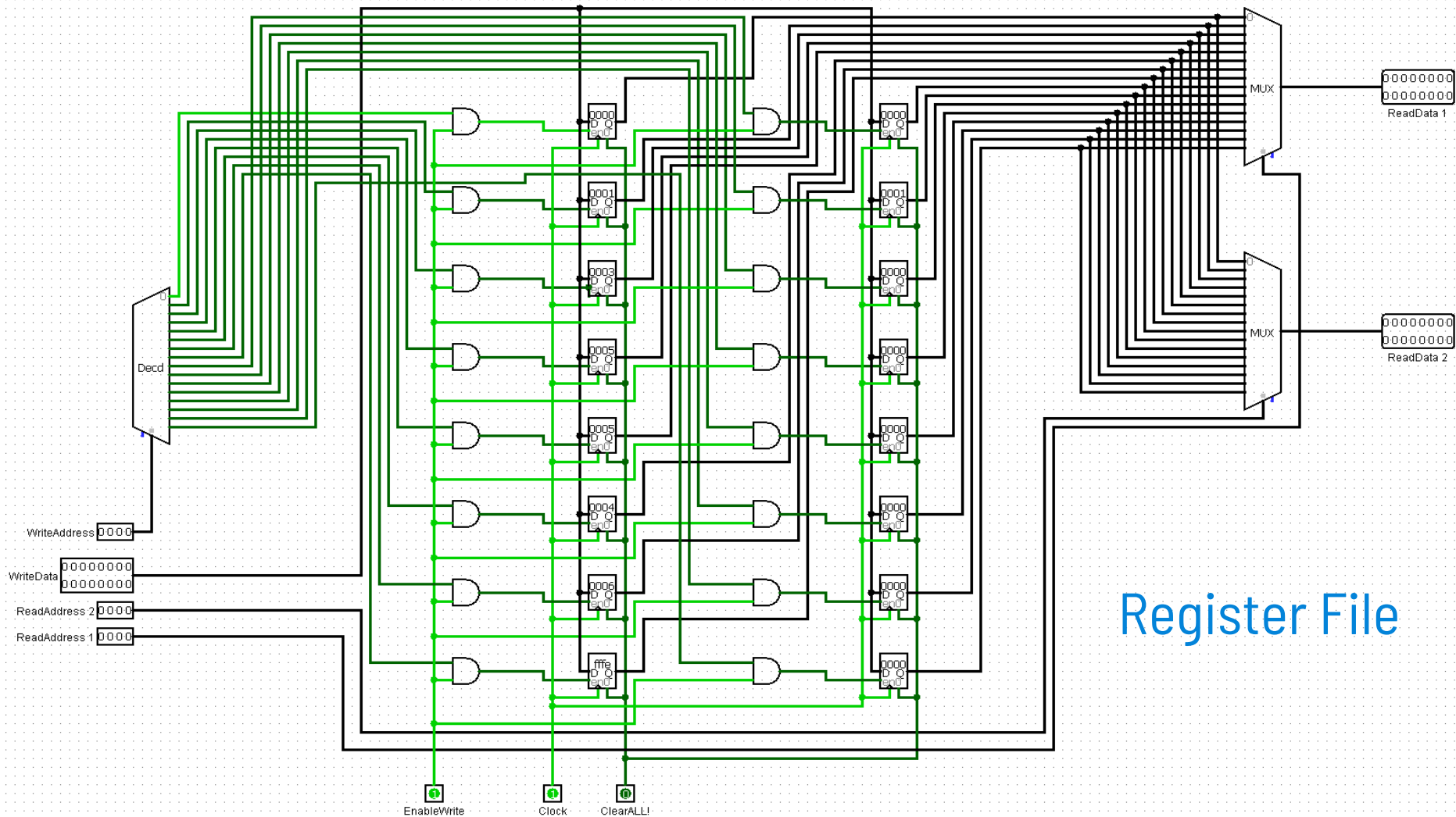
Bits	15	14:11	10:3	2:0
Description	Opcode	Unused	New address of the program counter	Opcode
Number of bits	1	4	8	3

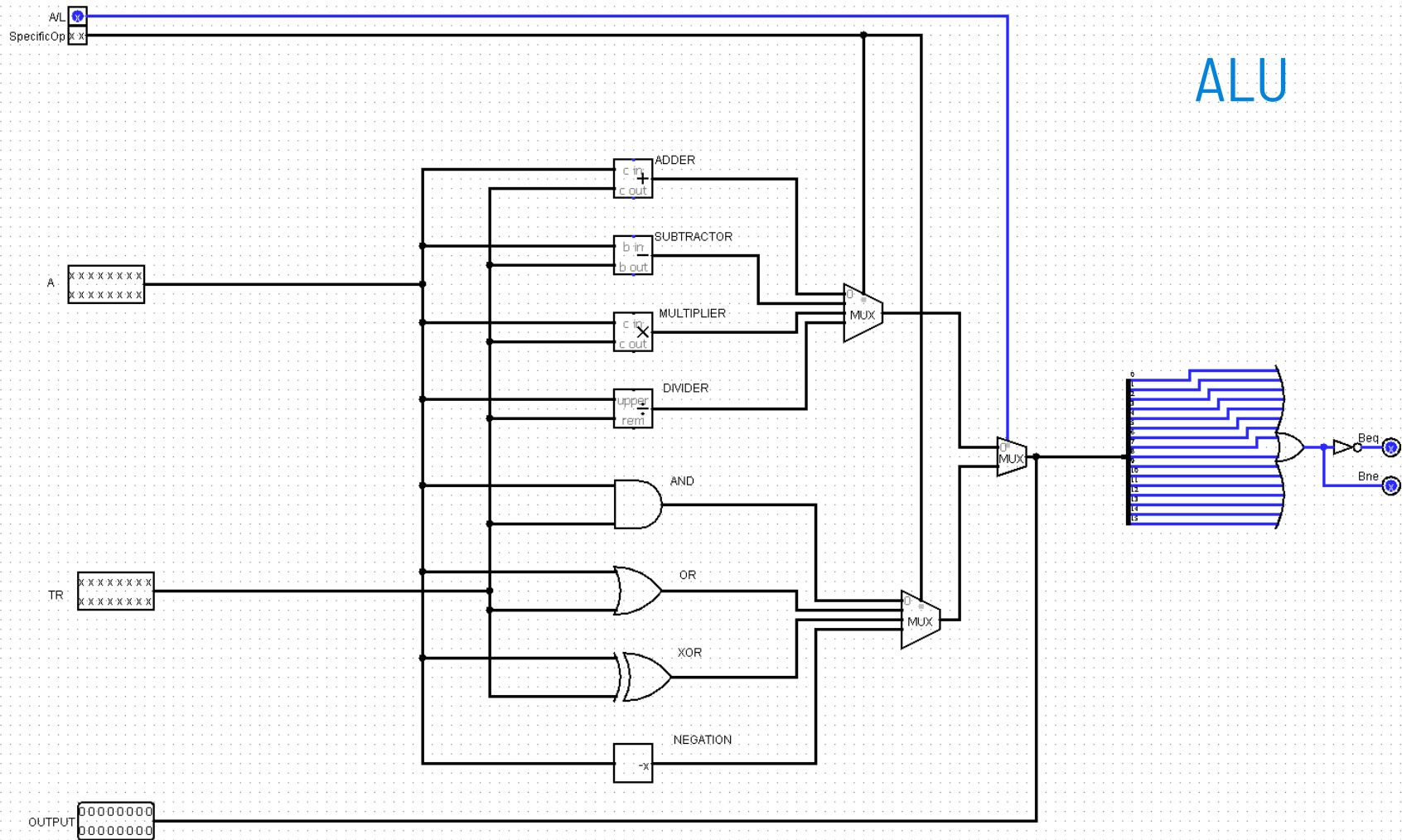
Jump Instructions



Now, The Eternals...

Oh sorry, “internals”





Sample Programme

880C	1.	load reg1 [mem1]	; reg1 = 1	mem1 = 1 mem2 = 5 mem3 = 3
901C	2.	load reg2 [mem3]	; reg2 = 3	
980C	3.	load reg3 [mem1]	; reg3 = 1	
A014	4.	load reg4 [mem2]	; reg4 = 5	
...				
2990	5.	add reg5, reg3, reg2	; reg5 = reg2 + reg3 = 4	
3208	6.	add reg6, reg4, reg1	; reg6 = reg1 + reg4 = 6	
3980	7.	beq -3, reg3, reg4	; goes back to line 5 if reg3 == reg4 but 1 != 5	
3A11	8.	sub reg7, reg4, reg2	; reg7 = reg2 - reg4 = -2	
8052	9.	jmp 10	; move to instruction line 11 (index starts at 0)	
418A	10.	mul reg8, reg3, reg1	; reg8 = reg1 x reg3 (never executed)	
C9A7	11.	ANDi, reg9, 3, reg4	; reg9 = reg4 AND 3 = 1	
824D	12.	store mem4, reg9	; mem4 = [reg9] = 1	
989B	13.	addi reg3, 1, reg3	; reg3 = reg3 + 1 = **	
E9A1	14.	bne -3, reg3, reg4	; loop: goes back to line 12 if [reg3] != [reg4]	

Thanks!

CREDITS: This presentation template was created by [Slidesgo](#), including icons by [Flaticon](#), infographics & images by [Freepik](#) and illustrations by [Stories](#)

