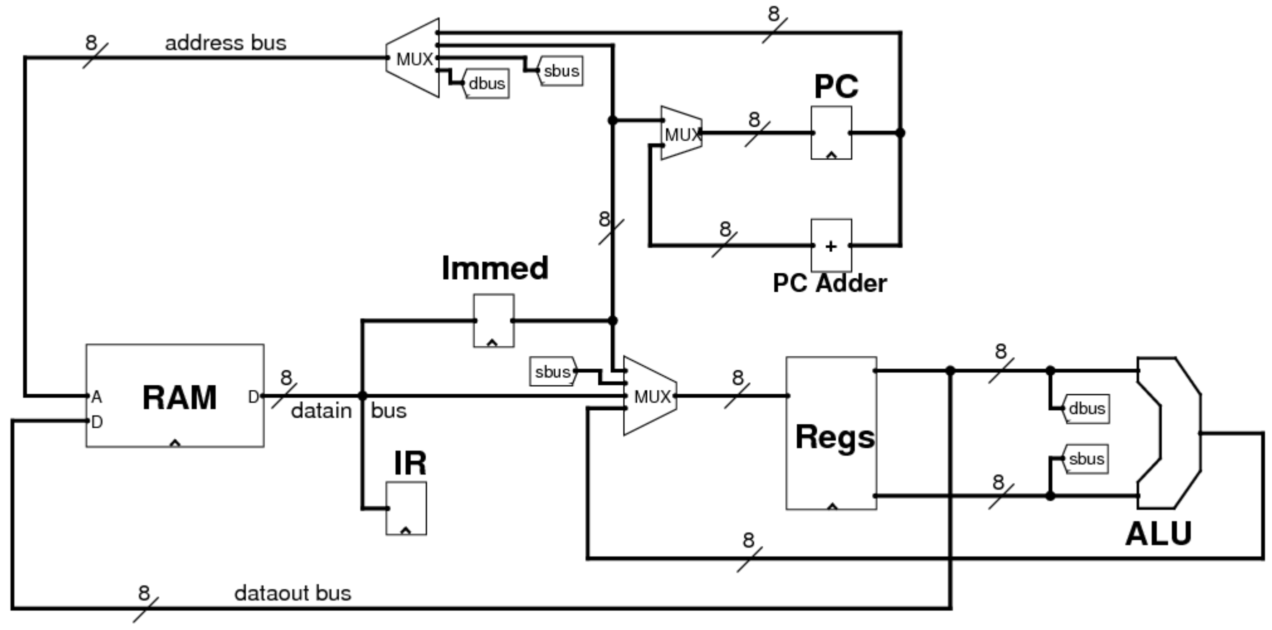
8-bit CPU

1 Design

- The CPU can support 256 bytes(2^8) of memory to hold both instructions and data.
- It is load/store kind of architecture where results are stored back into the the registers.
- The ALU performs the four operations AND, XOR, ADD and SUB on two 8-bit values.
- Internally, there are four 8-bit registers, R0 to R3, plus an Instruction Register, the Program Counter, and an 8-bit register which holds immediate values.
- The ALU operations have two operands: one register is a source register, and the second register is both source and destination register, i.e. destination register = destination register OP source register.
- There are 3 MUXs:
    - The address bus MUX: - It gets a memory address from the PC, the immediate register or from the source or destination registers
    - The PC multiplexor either lets the PC increment, or jump to the value in the immediate register.
    - The multiplexor before RegFile:- It determines where a register write comes from: the ALU, the immediate register, another register or the data bus.
- The *dbus* is destination registers and *sbus* is source registers
- The dataout bus is only connected to the *dbus* line, so that only value which can be written to memory is the destination register.
- The datapath of our CPU:



2 How to Run

Load the Alu.circ file into the main file before running the main file.

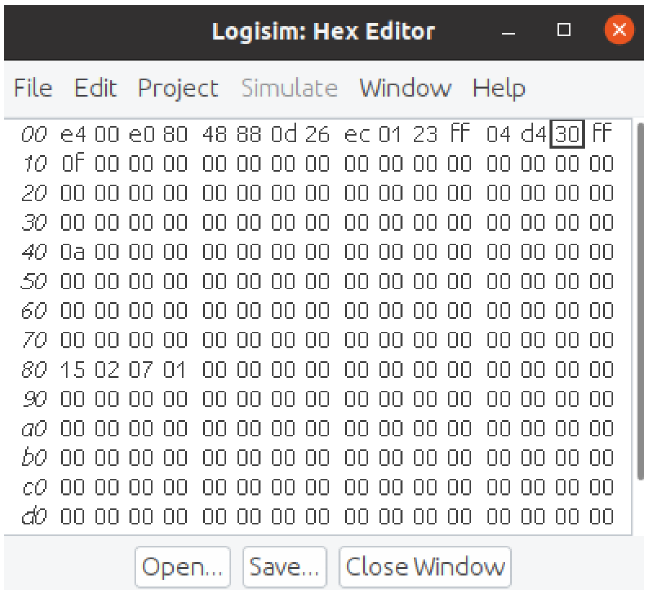
For any given set of instruction in mips convert it into binary instruction as per the following instruction table

Instruction Set

op1	op2	Mnemonic	Purpose
00	00	AND Rd, Rs	Rd = Rd AND Rs
00	01	XOR Rd, Rs	Rd = Rd XOR Rs
00	10	ADD Rd, Rs	Rd = Rd + Rs
00	11	SUB Rd, Rs	Rd = Rd - Rs
01	00	LW Rd, (Rs)	Rd = Mem[Rs]
01	01	SW Rd, (Rs)	Mem[Rs] = Rd
01	10	MOV Rd, Rs	Rd = Rs
01	11	NOP	Do nothing
10	00	JEQ Rd, immed	PC = immed if Rd == 0
10	01	JNE Rd, immed	PC = immed if Rd != 0
10	10	JGT Rd, immed	PC = immed if Rd > 0
10	11	JLT Rd, immed	PC = immed if Rd < 0
11	00	LW Rd, immed	Rd = Mem[immed]
11	01	SW Rd, immed	Mem[immed] = Rd
11	10	LI Rd, immed	Rd = immed
11	11	JMP immed	PC = immed

A sample program:- A program to add numbers given in an array.

MIPS Code	Binary instruction
LI R1,0x00	e4 00
LI R0,0x80	e0 80
LW R2, (R0)	48
JEQ R2, end	88 0d
ADD R1, R2	26
LI R3, 0x01	ec 01
ADD R0, R3	23
JMP loop	ff 04
SW R1, 0x30	d4 30
JMP inf	ff 0f

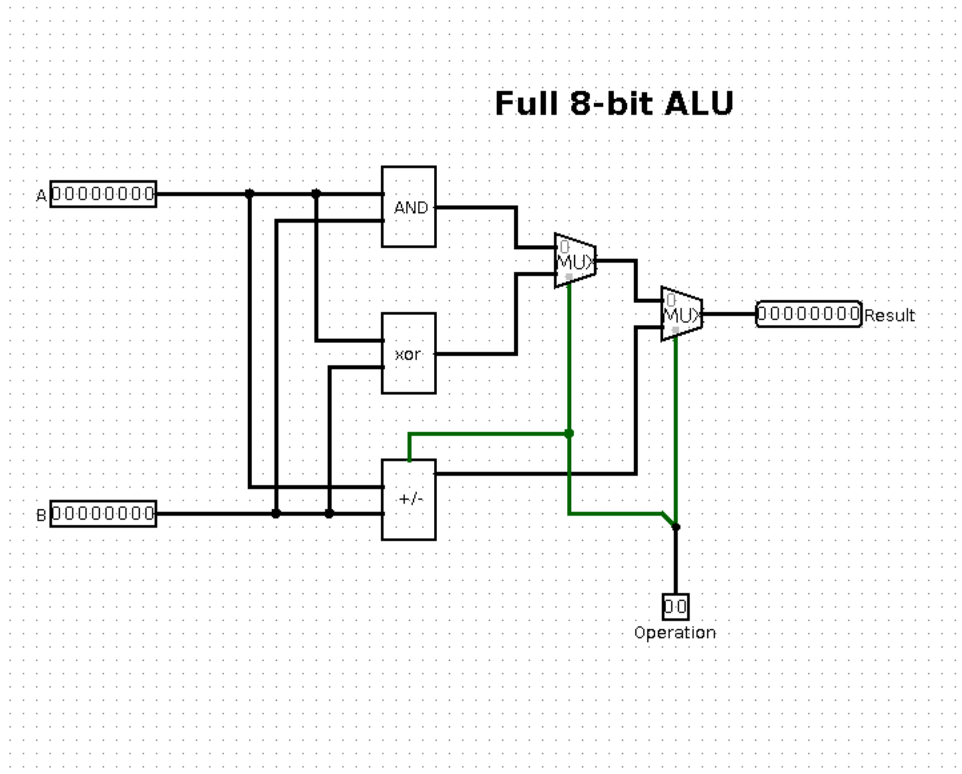

Logisim: Hex Editor
File Edit Project Simulate Window Help
Open... Save... Close Window

```
00 e4 00 e0 80 48 88 0d 26 ec 01 23 ff 04 d4 30 ff
10 0f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40 0a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80 15 02 07 01 00 00 00 00 00 00 00 00 00 00 00 00
90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

3 Control lines

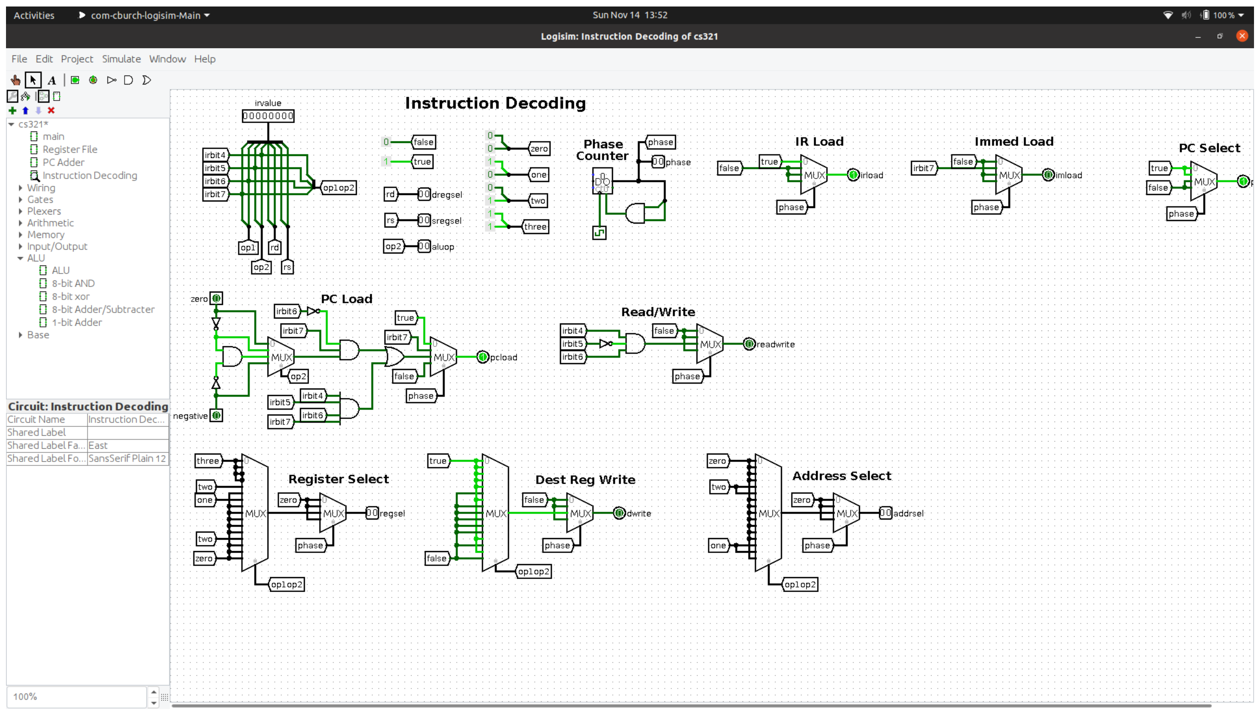
- There are several 1-bit control lines:
  - *pcsel*, increment PC or load the jump value from the Immediate Register.
  - *pcload*, load the PC with a new value, or don't load a new value.
  - *irload*, load the Instruction Register with a new instruction.
  - *imload*, load the Immediate Register with a new value.
  - *readwrite*, read from memory, or write to memory.
  - *dwrite*, write a value back to a register, or don't write a value.
- There are also several 2-bit control lines:
  - *addrsel*, select an address from the PC, the Immediate Register, the source register or the destination register.
  - *regsel*, select a value to write to a register from the Immediate Register, another register, the data bus or from the ALU.
  - *dregsel* and *sregsel*, select two registers whose values are sent to the ALU.
  - *aluop*, which are the *op2* bits that control the operation of the ALU.

4 8 Bit ALU



4 The Decoder

It basically decodes the instruction and sets correspnding control lines.



REFRENCES:

https://www.mips.com/products/architectures/
https://tams.informatik.uni-hamburg.de/applets/hades/webdemos/mips.html
http://cmosedu.com/cmos1/electric/Senior%20Design%20Report.pdf
https://github.com/yxwangcs/MIPS-CPU
https://en.wikipedia.org/wiki/MIPS_architecture
https://minnie.tuhs.org/CompArch/Tutes/week03.html
https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/mips/index
.html