CSE 332 PROJECT

Course: CSE 332

Section: 12

Submitted to : Dr Khaleda Ali (KDA1)

Developed by

Md Yearat Hossain

ID: 1712275642

Mostak Khan Khadem

ID: 1621777042

Md Abir

ID: 1621775042

Pranto Shaha

ID: 1530261642

Summary

A 10 bit processor was made as the project of the course CSE 332. It has 10 bit instruction set.

The processor is based on RISC architecture. It takes instructions from an instruction memory and can store them in a data memory. It has 4 registers and it can perform both R type and I type operations.

Instruction Formats:

R type:

opcode	rs	rt	rd
4bit	2bit	2bit	2bit

I type:

opcode	rs	rt	immediate
4bit	2bit	2bit	2bit

Features :

The processor can perform various arithmetic operations.
Example: **addition, subtraction**

It can perform various logical operations too.
Example: **and, or**

It can perform branch operation.
Example: **Branch on Equal**

It can also perform some immediate operations.
Example: **addi, ori** etc.

It can load and store values from memory.

It works on unsigned values.

It takes 4 bit opcode for selecting operations.

It can access 4 different registers and use them for various operations.

Operation Table:

Opcode	Operation	Format
0000	and	R
0001	or	R
0010	add	R
0011	sub	R
1000	andi	I
1001	ori	I
1010	addi	I
1011	subi	I
1101	store	I
1110	load	I
1111	Branch on Equal	I

The processor can perform **11** different types of operations. Among these **4** are from **R** type and **7** are from **I** type

Main Components

The processor was built combining various components. Each of the components has their own functionality.

These are:

 # Program Counter: It holds the address of the next instruction.

 # Instruction Memory: It holds the instructions to be executed.

 # Register File: It holds the registers for the processor.

 # ALU: The ALU does all the arithmetic and logical operations.

 # Control Unit: It generates control signals for various operations.

 # Data Memory: It is used for loading and storing data.

ALU

The ALU is a 10 bit ALU which was made combining **1** bit ALUs. The ALU takes **10** bit inputs and can perform ***and, or , subtract*** and ***addition*** operations. Operations are specified by opcode and control unit.

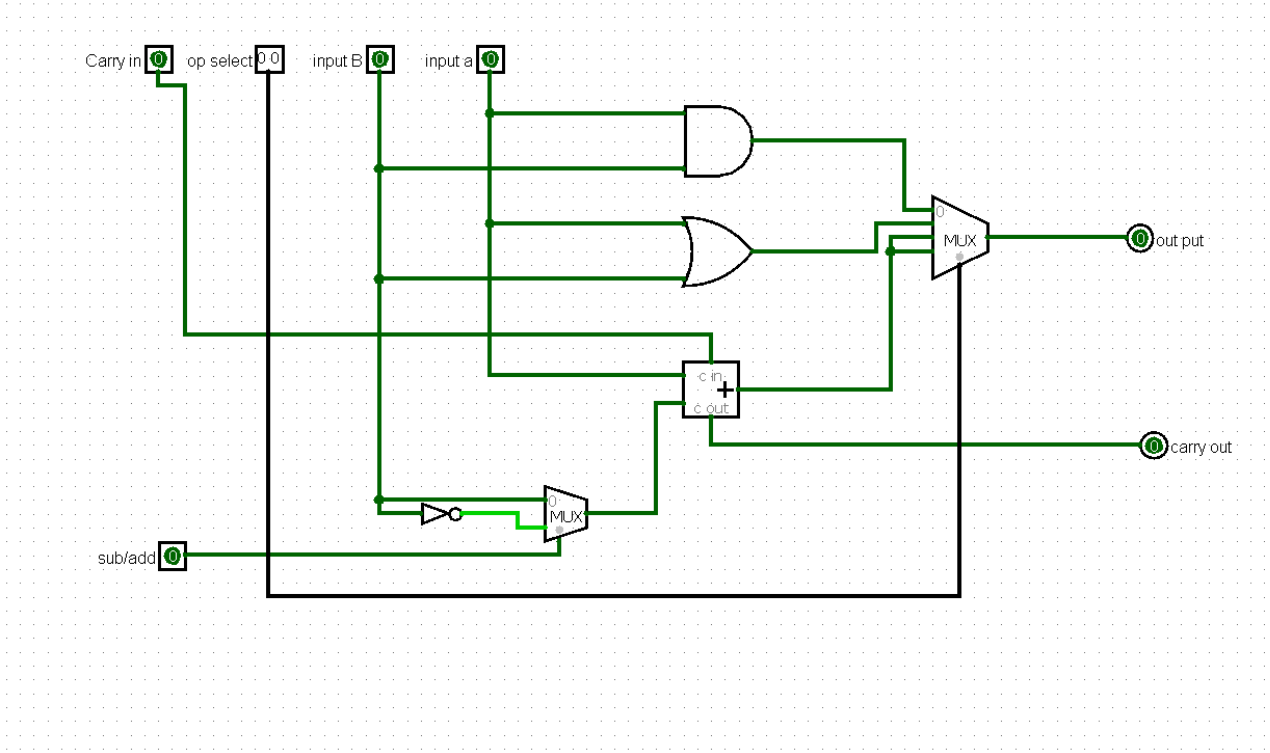


Figure: 1 bit ALU

Instruction Memory:

It holds the **instructions** and passes the instructions to the **register file** according to Program counter and Clock Pulse. It holds the instructions in **Hexadecimal** number system which is provided by the assembler/compiler.

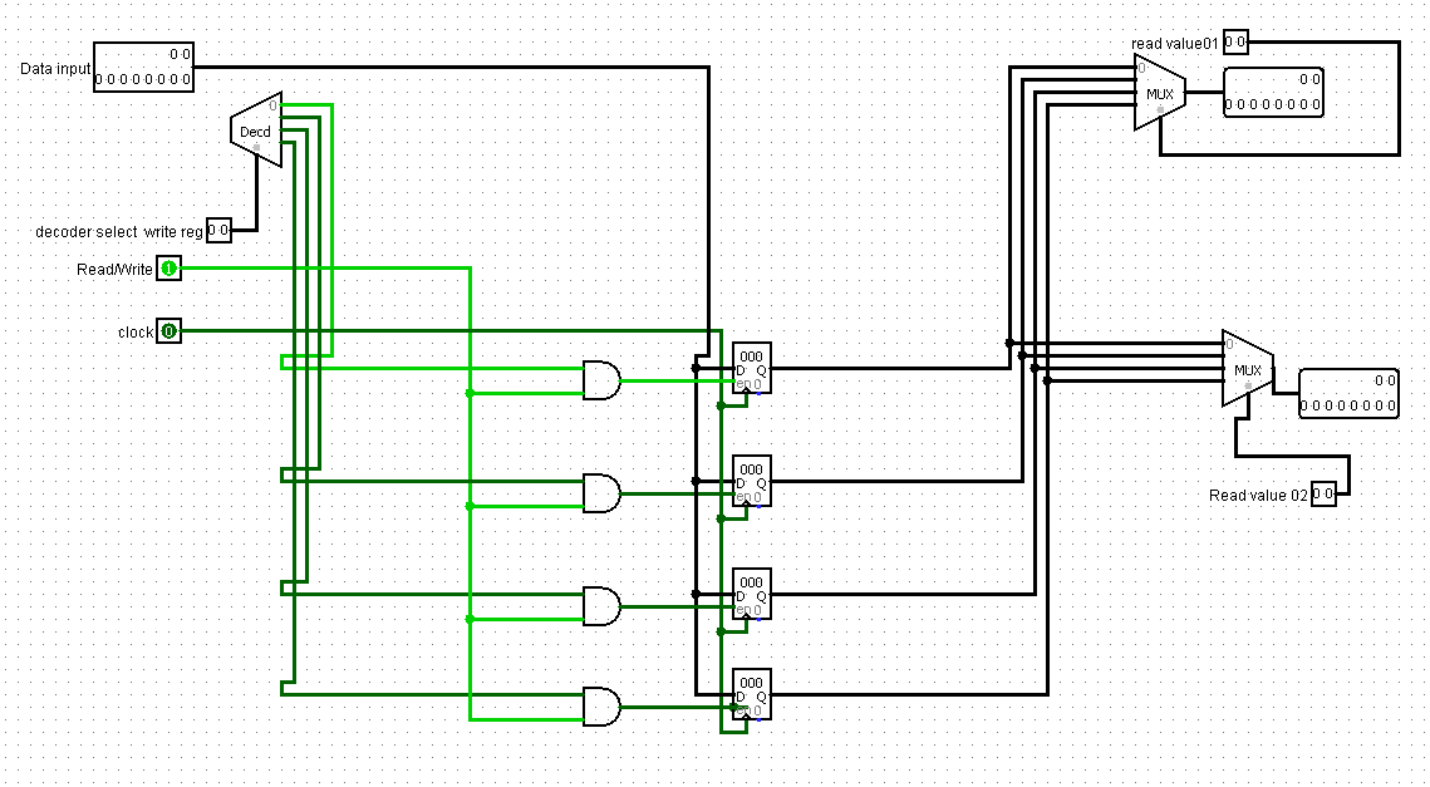
Data Memory:

It is used for ***loading*** and ***storing*** values.
During load operation, values are grabbed from **Data Memory** and loaded into **registers**.
During store operation, values are taken from **registers** and stored in **Data Memory.**
The operation mode is selected by memory's **load/store** control section which is generated by the **control unit**.

Register File:

The register file contains all the registers used by the processor. This register file contains 4 different 10 bit registers. The registers are: $R0, $R1, $R2 and $R3.

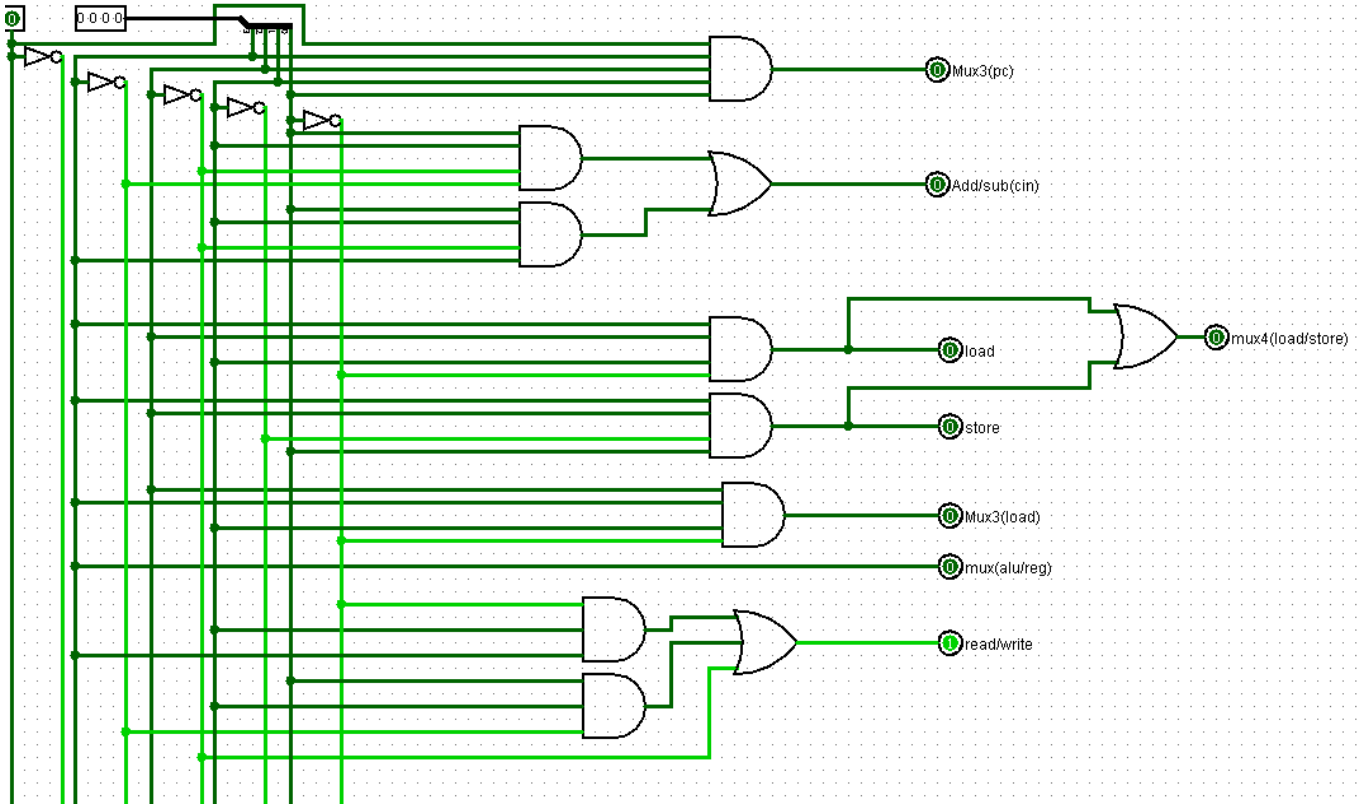
The register file contains a decoder which is used for selecting a specific register. As there are only 4 registers , the registers can be accessed by 2 bit binary values from 00 to 11.



Control Unit:

The control unit generates values for various control sections according to the opcode.

The control unit was made using NOT, AND & OR gates.
K-map was used for designing the control unit.



Control Unit Table :

Opcode	Operation	mux	mux2	mux3	carry	add / sub	read / write	load	store
0000	and	0	0	0	0	0	1	0	0
0001	or	0	0	0	0	0	1	0	0
0010	add	0	0	0	0	0	1	0	0
0011	sub	0	0	0	1	1	1	0	0
1000	andi	0	1	0	0	0	1	0	0
1001	ori	0	1	0	0	0	1	0	0
1010	addi	0	1	0	0	0	1	0	0
1011	subi	0	1	0	1	1	1	0	0
1101	store	0	1	0	0	0	0	0	1
1110	load	0	1	1	0	0	1	1	0
1111	Branch on Equal	1	1	0	0	0	0	0	0

How it Works :

The instruction memory passes the 10 bit instruction set to the register file according to the clock pulse. Here, most significant 4 bits are used for opcode and they also enables the control unit which generates control signals for various control sections based on operations.
But the least significant 2 bits are used as destination register for R type instructions and as immediate for I type instructions. A control section determines when and how to handle the last 2 bits.

For R type operations values are used from registers and also stored in register. Here ALU performs operations on the values of corresponding registers.

In I type format, the immediate 2 bit value is sign extended to 10 bit value and then sent into the ALU for operations. For load/store operations the ALU output is a memory address for the data memory , otherwise it can be just a value to be stored in the registers.For branch on equal operation two register values are compared and if they are equal the program counter is incremented by the offset value.

Several control sections determine when to store , load and write values depending on the opcode and the control unit.

Demo :

Code	Binary Instruction	Hexa Instruction
LW $R1 , $R2 (0) rt rs	1110100100	3A4
LW $R2, $R0 (1) rt rs	1110001001	389
Beq $R1, $R2, 2	1111011010	3DA
ADD $R3, $R1, $R2 rd rs rt	0010011011	09B
SW $R3, $R0(3) rs rt	1101110011	373
Else: SUB $R3, $R1, $R2 rd rs rt	0011011011	0DB
SW $R3, $R0 (3)	1101110011	373

```
If ( a == b) {
      F = a – b
  }
Else {
      F = a + b
}
```

Disadvantages :

It doesn't work with Signed values.

Instruction set doesn't support any functionality.

It can't perform branch not equal operation.

Immediate value can't exceed 3.