Description of our ISA:

- We are following the Von Neumann architecture.
- We have only direct access of memory
- We have 32 general purpose registers.
- Each register is 32 bits each
- The operand types are signed Integers
- We have in total 13 instructions in our ISA :

Arithmetic :

Add $rd $rt $rs (3 registers arguments)

$rd = $rs + $rt

Addi rd rs val(2 registers and 1 immediate value arguments)

$rd = $rs + val

Sub $rd $rt $rs(3 registers arguments)

$rd = $rs - $rt

Mult $rd $rt $rs(3 registers arguments)

$rd = $rs * $rt

Move $rd $rs(2 registers arguments)

$rd = $rs

Clear $rd (1 register arguments)

$rd = 0

Logical :

And $rd $rt $rs (3 registers arguments)

$rd = $rs & $rt

Or $rd $rt $rs (3 registers arguments)

$rd = $rs | $rt

Memory :

Lw $rd $rs (2 registers arguments)

$rd = mem[$rs]

Sw $rd $rs(2 registers arguments)

mem[$rs] = $rd

Conditions:

Beq $rd $rt $rs (3 registers arguments)

If $rs == $rt, branch to instruction at mem[$rd]

Slt $rd $rt $rs (3 registers arguments)

If $rs < $rt, let $rd = 1

Jr $rd (1 register arguments)

Jump to memory of $rd

The instructions are divided into two types: immediate value instructions and non-immediate value instructions. However, we have one division for the fields of our 32 bit instructions.

1 bit	16 bits	5 bits	5 bits	5 bits
Flag	opcode/immediate value	Src reg 1	Src reg 2	Dest reg

The opcodes for the instructions were chosen following the “One Hot Assignment” method as follows :

1. Mult : 0000000000000001
2. Add : 0000000000000010
3. Sub : 0000000000000100
4. Move : 0000000000001000
5. Clear : 0000000000010000
6. And : 0000000000100000
7. Or : 0000000001000000
8. Beq : 0000000010000000
9. Slt : 0000000100000000
10. Lw : 0000001000000000
11. Sw : 0000010000000000
12. Jr : 0000100000000000

Hardware Design