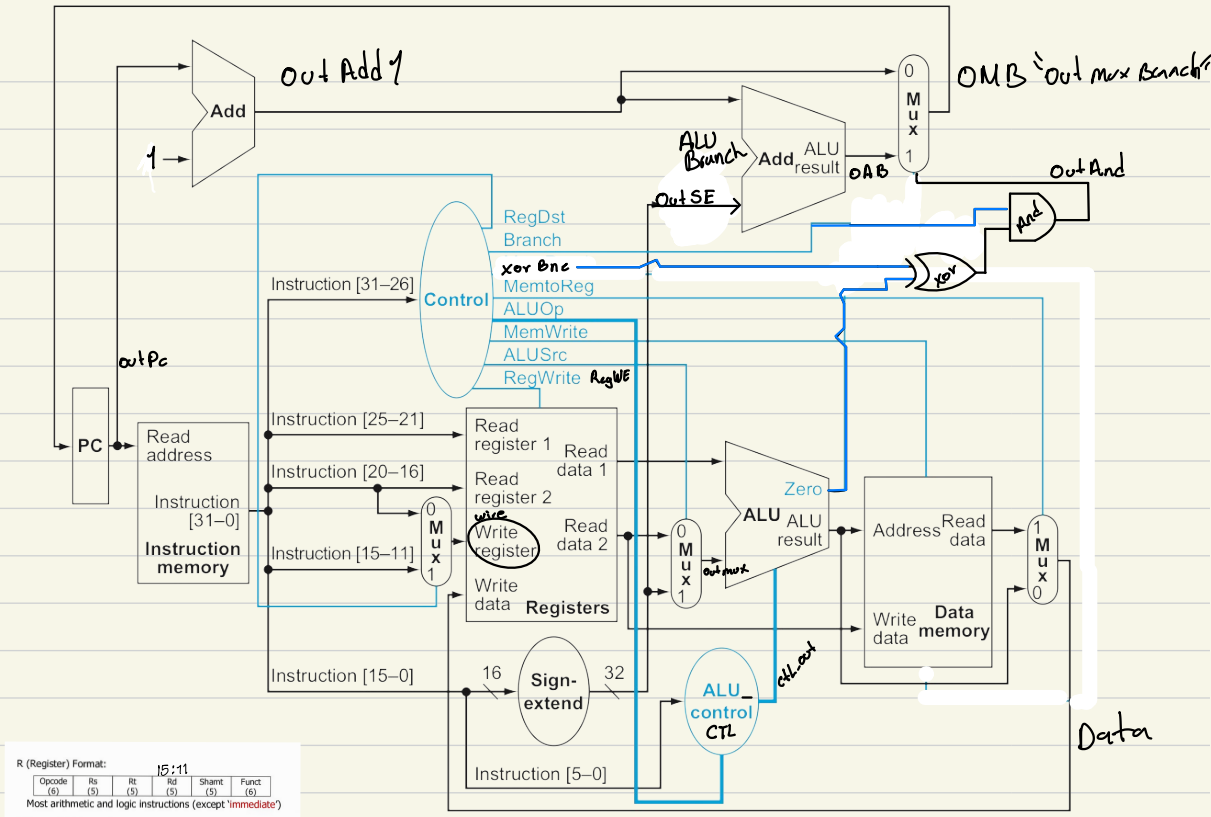
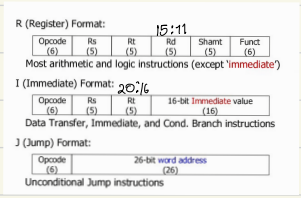
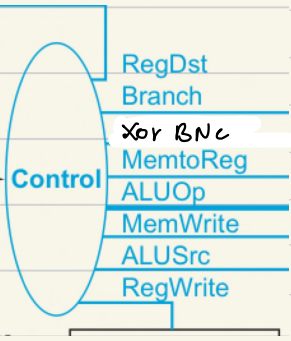




Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111



R	I Lw	I Sw	I Beq	I Bne
1	0	0	0	0
0	0	0	1	1
0	0	0	0	1
0	1	1	0	0
000	001	010	011	100
1/0	0	1	0	0
0	1	1	0	0
1	1	0	0	0

Don't care? Because we only branch to the other instructions

100011 (35) LW rt, imm(rs) // load word
101011 (43) SW rt, imm(rs) // store word
rt: Destiny

xor

0 0	0
0 1	1
1 0	1
1 1	0

000100 (4)
beq rs, rt, label
branch if equal

000101 (5)
bne rs, rt, label
branch if not equal

SW Instruction

Instruction:10101100000100000000000000000011

RT

CONTROL

RegDst:0	Branch:0	XorBNE:0
MemtoReg:1	ALUOp:010	MemWrite:1
ALUSrc:1	RegWrite:0	Zero:0

RSData:00000000000000000000000000000000

RTData:00000000000000000000000000001000

OutALU:00000000000000000000000000000011

OutDataMem:00000000000000000000000000001000

In the direction 3 puts the data 8, then is saved in dataprom and it will be what will out of the data mem.

```
0 00000000000000000000000000000000
1 00000000000000000000000000000000
2 00000000000000000000000000000000
3 00000000000000000000000000001000
  00000000000000000000000000000001
  00000000000000000000000000000010
  00000000000000000000000000000011
```

```
# KERNEL:                    BEQ Instruction
# KERNEL: Instruction:00010000010000010000000000000101
# KERNEL:                    CONTROL
# KERNEL: RegDst:0        Branch:1        XorBNE:0
# KERNEL: MemtoReg:0      ALUOp:011       MemWrite:0
# KERNEL: ALUSrc:0        RegWrite:0      Zero:1
# KERNEL: RSData:00000000000000000000000000000011
# KERNEL: RTData:00000000000000000000000000000011
# KERNEL: OutALU:00000000000000000000000000000000
# KERNEL: OutDataMem:00000000000000000000000000000000
# KERNEL: OutALUBranch:00000000000000000000000000000110
# KERNEL:
# KERNEL:                    R-Instruction AND
# KERNEL: Instruction:00000000100001010100000000100100
# KERNEL:                    CONTROL
# KERNEL: RegDst:1        Branch:0        XorBNE:0
# KERNEL: MemtoReg:0      ALUOp:000       MemWrite:1
# KERNEL: ALUSrc:0        RegWrite:1      Zero:0
# KERNEL: RSData:00000000000000000000000000001000
# KERNEL: RTData:00000000000000000000000000001001
# KERNEL: OutALU:00000000000000000000000000001000
```

Both registers content the same data

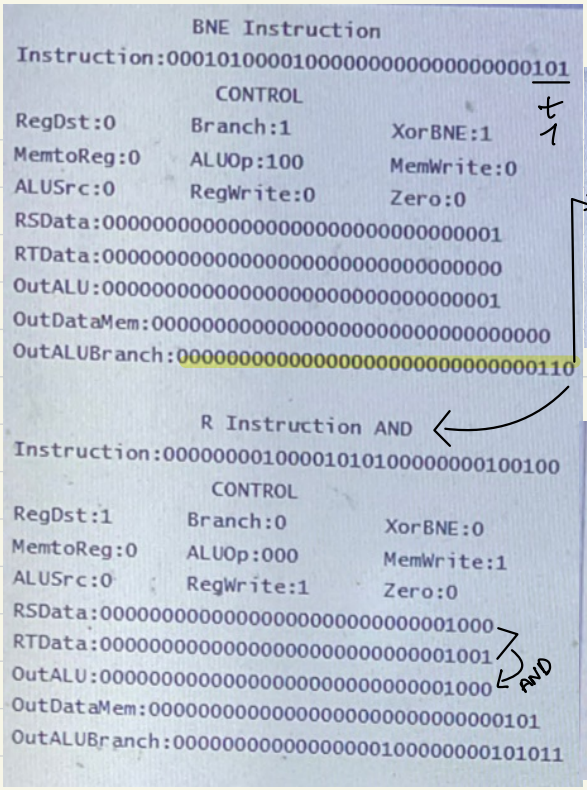
(+1)

this comes from the adder

```
0 00010000010000010000000000000101//Beq Register 1,2,br
1 10101100000100000000000000000011//sw $t0,3($0) -DONE
2 00010100011000010000000000000101//Bne Register 1,2,br
3 10001101000001001000000000000100//lw $t1,4($t0) -DONE
4 00000001000010010100000000100000//R ADD $s0 $t0$t1
5 10101101001010100000000000000011//sw $t2,3($t1) -DONE-
6 00000001000010101000000000100100//R AND $s0 $t0$t1
  10001100000010111111111111110000//lw $t3,-16($0)-DONE
  10101110001011000000000000011010//sw $t4,26($s1) -DONE
  00000001000010010100000000100010//R SUB  $s0 $t0$t1
  00000001000010010100000000100101//R OR  $s0 $t0$t1
  00000001000010010100000000101010//R SLT $s0 $t0$t1
```

```
0 00000000000000000000000000000000
1 00000000000000000000000000000011
2 00000000000000000000000000000011
3 00000000000000000000000000000011
```

both registers are equal, so it execute the branch.




```
0 00010100001000000000000000000101//Bne Register 1,2,branc
1 00010000010000010000000000000101//Beq Register 1,2,branc
2 10101100000010000000000000000011//sw $t0,3($0) -DONE
3 10001101000010010000000000000100//lw $t1,4($t0) -DONE
4 00000001000010010100000000100000//R ADD $s0 $t0$t1
5 10101101001010100000000000000011//sw $t2,3($t1) -DONE---
6 00000001000010101000000000100100//R AND $s0 $t0$t1
8 10001100000010111111111111110000//lw $t3,-16($0)-DONE
9 10101110001011000000000000011010//sw $t4,26($s1) -DONE
10 00000001000010010100000000100010//R SUB  $s0 $t0$t1
11 00000001000010010100000000100101//R OR  $s0 $t0$t1
12 00000001000010010100000000101010//R SLT $s0 $t0$t1
13       //$t0 //$t1 //$s0
```

Registers