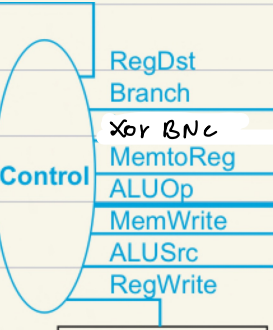


R (Register) Format: 15:11
Opcode (6) Rs (5) Rt (5) Rd (5) Shift (5) Funct (6)
Most arithmetic and logic instructions (except 'immediate')

I (Immediate) Format: 20:16
Opcode (6) Rs (5) Rt (5) 16-bit Immediate value (16)
Data Transfer, Immediate, and Cond. Branch instructions

J (Jump) Format: 26-bit word address (26)
Opcode (6)
Unconditional Jump instructions

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	Lw	Sw	Beq	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

1000 11 (35) LW $r_d, imm(rs)$ // load word
1010 11 (43) SW $r_s, imm(rs)$ // store word
↳ Destination

xor
00 0
01 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

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

SW Instruction

Instruction: 10101100000010000000000000000011

CONTROL

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

RSData:00000000000000000000000000000000

RTData:00000000000000000000000000000100

OutALU:00000000000000000000000000000011

OutDataMem:00000000000000000000000000000100

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

```
0 00000000000000000000000000000000
1 00000000000000000000000000000000
2 00000000000000000000000000000000
3 00000000000000000000000000000001
00000000000000000000000000000001
00000000000000000000000000000001
00000000000000000000000000000001
00000000000000000000000000000001
```

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

Both registers contain the same data

+1

this comes from the adder

```
0 00010000010000010000000000000101//Beq Register 1,2,br
1 10101100000010000000000000000011//sw $t0,$0 -DONE
2 00010100011000010000000000000101//Bne Register 1,2,br
3 10001101000010010000000000000100//lw $t1,4($t0) -DONE
4 00000001000010010100000000000000//R ADD $s0,$t0,$t1
5 10101101001010100000000000000111//sw $t2,$3($t1) -DONE
6 0000000100001010100000000100100//R AND $s0,$t0,$t1
1000110000001011111111111110000//lw $t3,-16($0) -DONE
10101110001011000000000000011010//sw $t4,26($s1) -DONE
00000001000010010100000000000010//R SUB $s0,$t0,$t1
000000010000100101000000000000101//R OR $s0,$t0,$t1
0000000100001001010000000000001010//R SLT $s0,$t0,$t1
```

R=Instruction AND

KERNEL: Instruction: 00000000100001010100000000100100

CONTROL

KERNEL: RegDst:1 Branch:0 XorBNE:0
KERNEL: MemtoReg:0 ALUOp:000 MemWrite:1
KERNEL: ALUSrc:0 RegWrite:1 Zero:0

KERNEL: RSData:00000000000000000000000000000100

KERNEL: RTData:00000000000000000000000000000101

KERNEL: OutALU:00000000000000000000000000000100

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

both registers are equal, so it executes the branch.

BNE Instruction

Instruction: 0001010000100000000000000000000101

CONTROL

RegDst:0 Branch:1 XorBNE:1
MemtoReg:0 ALUOp:100 MemWrite:0
ALUSrc:0 RegWrite:0 Zero:0

RSData: 0000000000000000000000000000000001

RTData: 0000000000000000000000000000000000

OutALU: 0000000000000000000000000000000001

OutDataMem: 0000000000000000000000000000000000

OutALUBranch: 00000000000000000000000000000000110

R Instruction AND

Instruction: 000000001000101010000000100100

CONTROL

RegDst:1 Branch:0 XorBNE:0
MemtoReg:0 ALUOp:000 MemWrite:1
ALUSrc:0 RegWrite:1 Zero:0

RSData: 000000000000000000000000000000001000

RTData: 000000000000000000000000000000001001

OutALU: 000000000000000000000000000000001000

OutDataMem: 00000000000000000000000000000000101

OutALUBranch: 00000000000000000000000010000000101011

0 0001010000100000000000000000000101 //Bne Register 1,2,bran
1 0001000001000001000000000000000101 //Beq Register 1,2,bran
2 101011000000100000000000000000011 //sw \$t0,3(\$0) -DONE
3 1000110100001001000000000000000100 //lw \$t1,4(\$t0) -DONE
4 00000001000010010100000000100000 //R ADD \$s0 \$t0\$t1
5 10101101001010100000000000000011 //sw \$t2,3(\$t1) -DONE---
6 0000000100001010100000000100100 //R AND \$s0 \$t0\$t1
8 1000110000001011111111111110000 //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

0000000000000000000000000000000000
0000000000000000000000000000000001
0000000000000000000000000000000010
0000000000000000000000000000000011
000000000000000000000000000000001000
000000000000000000000000000000001001
000000000000000000000000000000001100
00000000000000000000000000000000111
000000000000000000000000000000001000