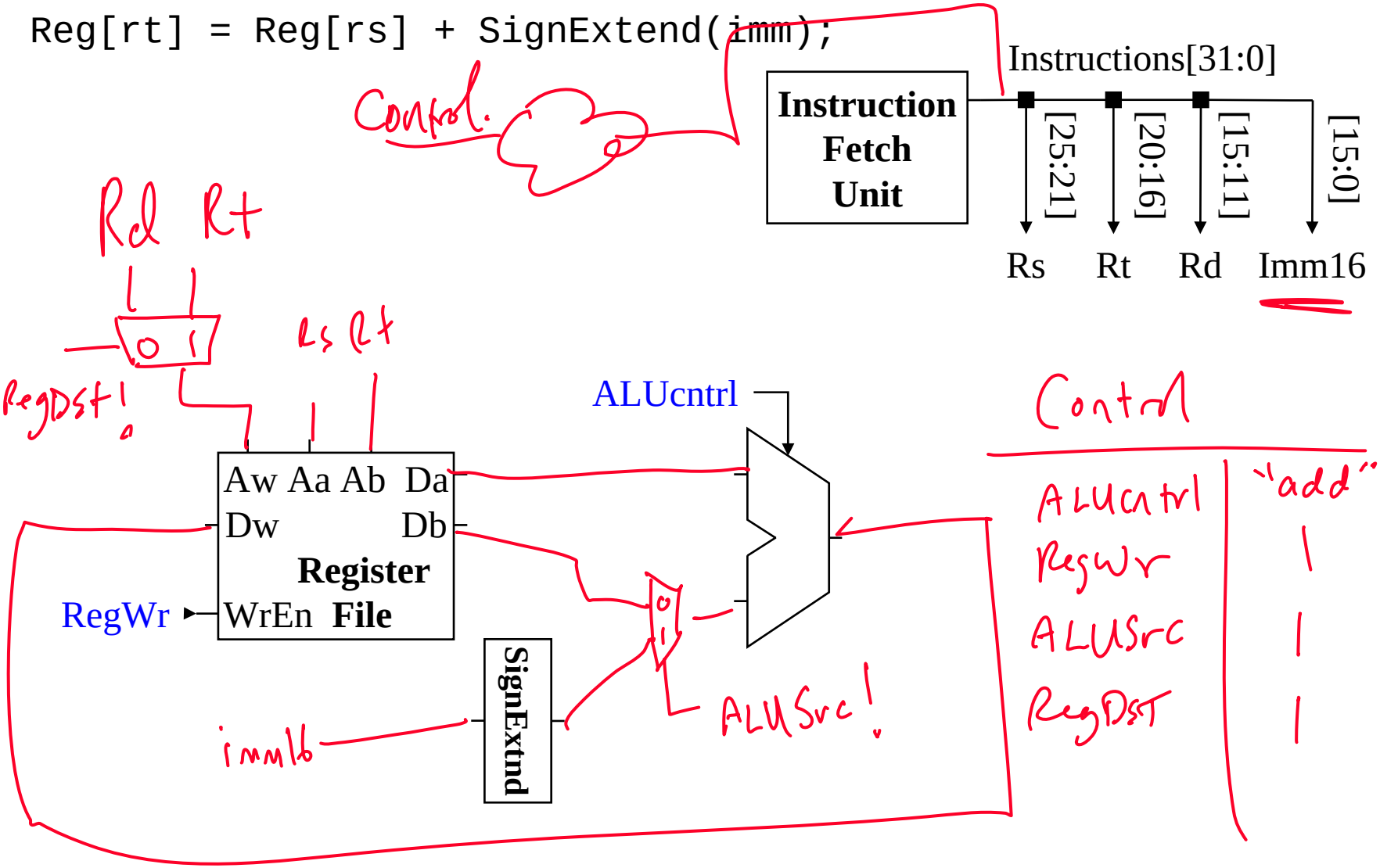
Datapath + Immediate Ops

```
Reg[rt] = Reg[rs] + SignExtend(imm);
```

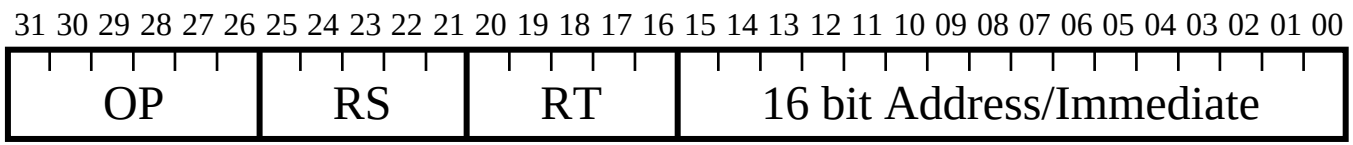


Load RTL

Load Instruction: `lw rt, imm(rs)`

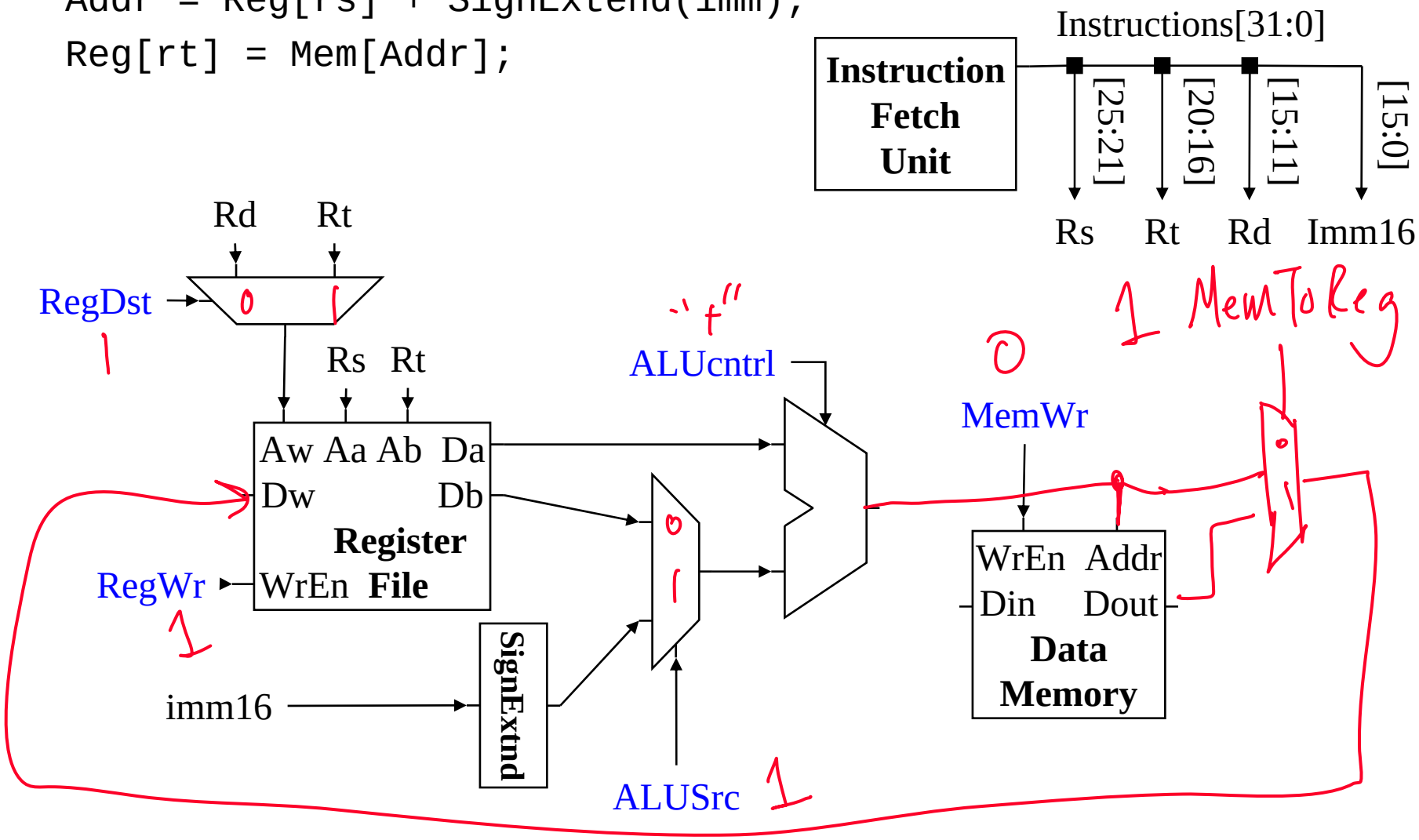
```
Instruction = Mem[PC];
Addr = Reg[rs] + SignExtend(imm);
Reg[rt] = Mem[Addr];
PC = PC + 4;
```

IF



Datapath + Load

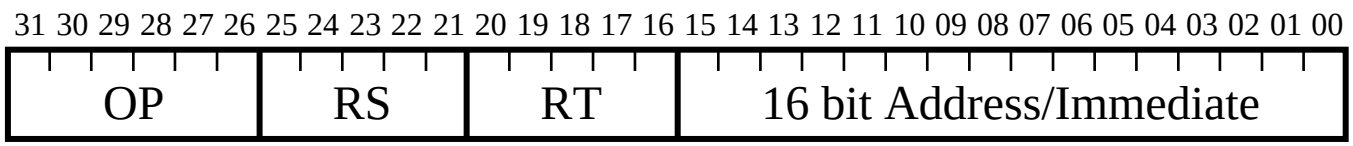
```
Addr = Reg[rs] + SignExtend(imm);
Reg[rt] = Mem[Addr];
```



Store RTL

Store Instruction: `sw rt, imm(rs)`

```
Instruction = Mem[PC];
Addr = Reg[rs] + SignExtend(imm);
Mem[Addr] = Reg[rt];
PC = PC + 4;
```

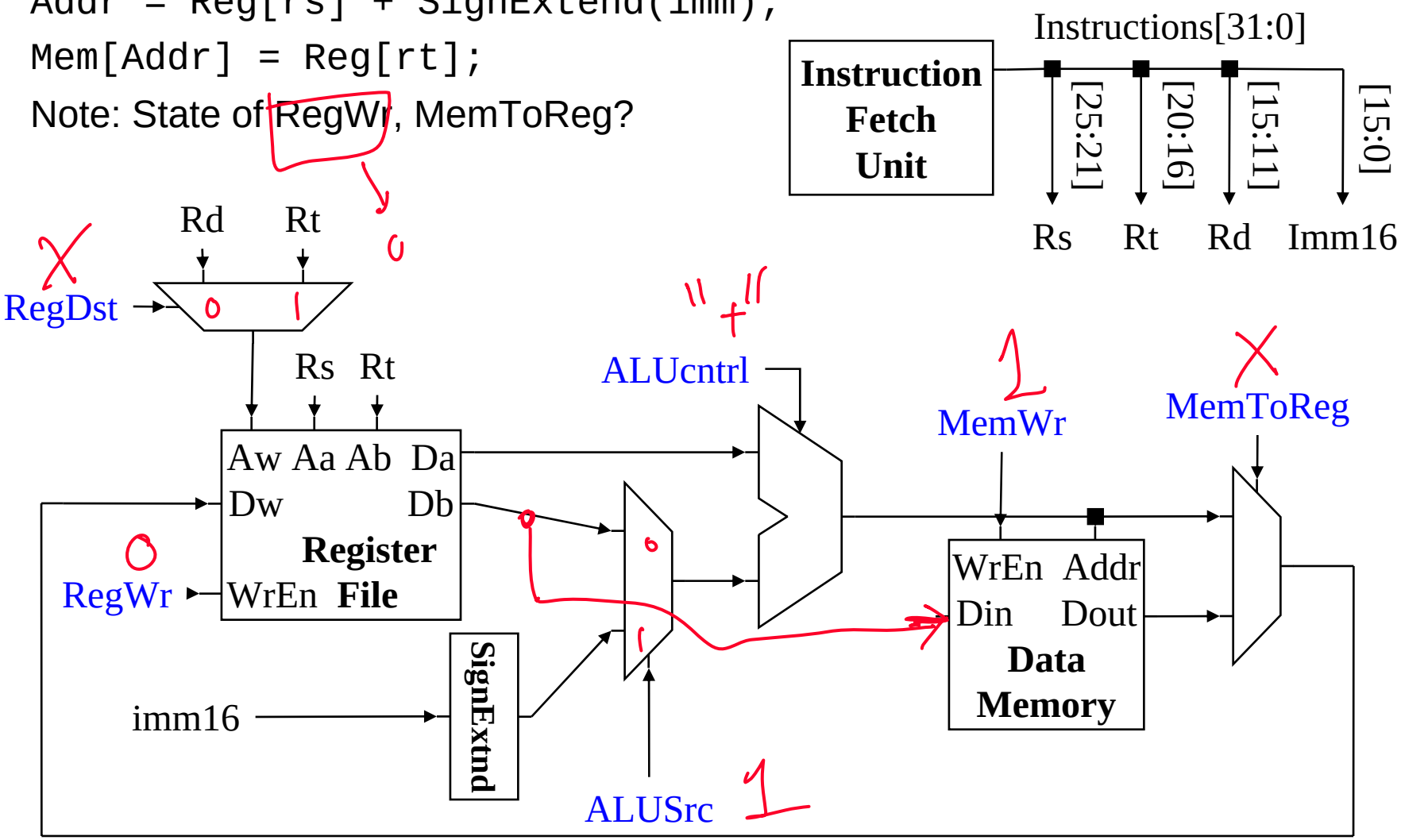


IF

Datapath + Store

```
Addr = Reg[rs] + SignExtend(imm);
Mem[Addr] = Reg[rt];
```

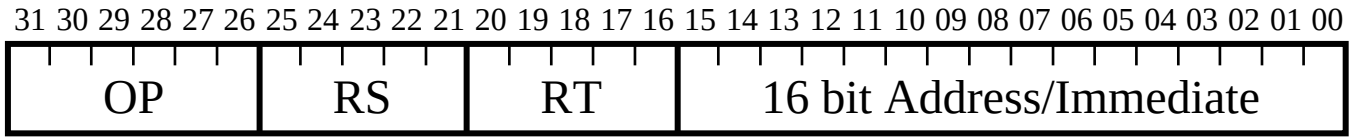
Note: State of RegWr, MemToReg?



Branch RTL

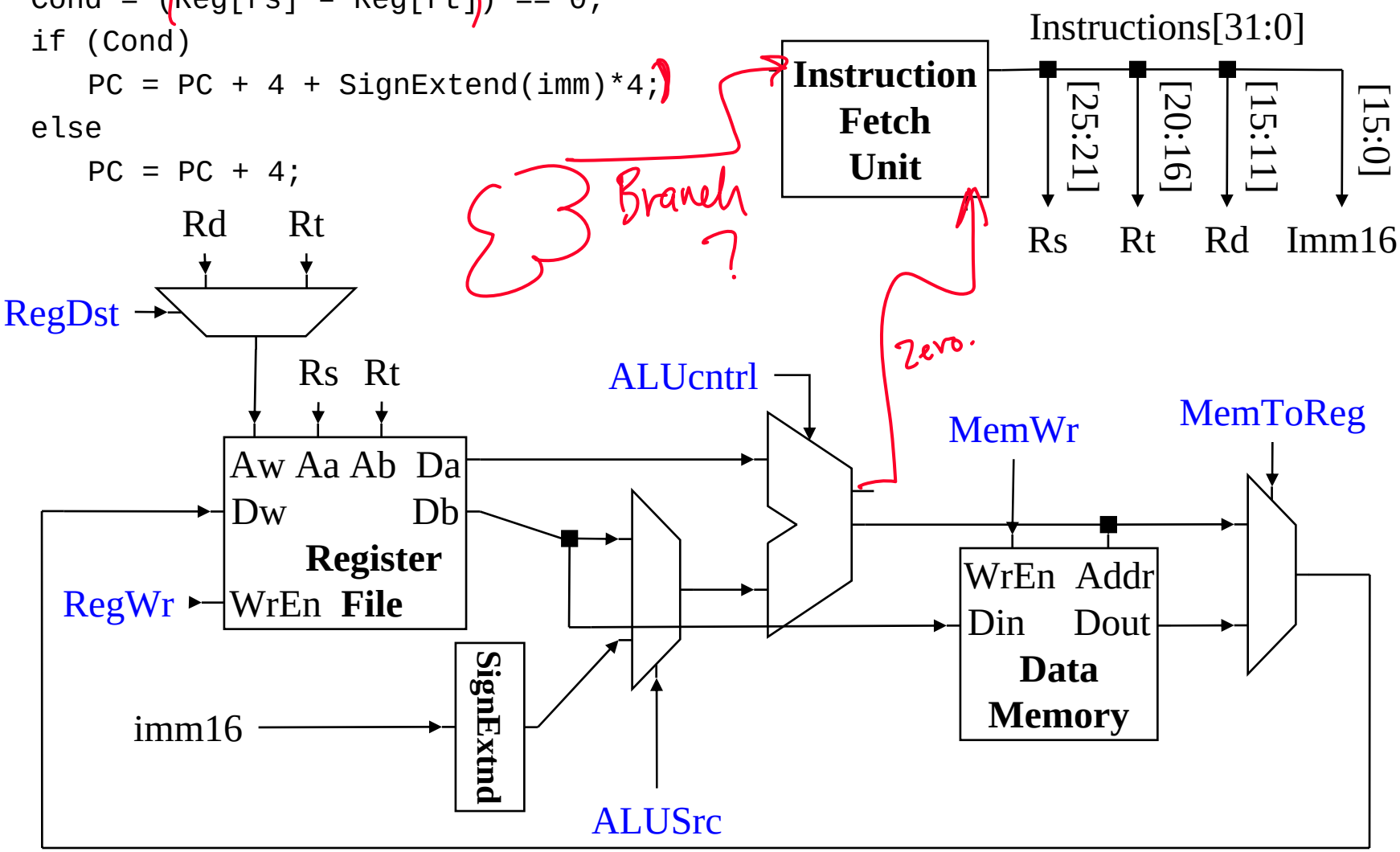
Branch Instruction: `beq rs, rt, imm`

```
Instruction = Mem[PC];
Cond = (Reg[rs] – Reg[rt]) == 0;      // Test equality
if (Cond)
    PC = PC + 4 + SignExtend(imm)*4; // Neg for backward
                                     // *4: LSbits == 00
else
    PC = PC + 4;
```



Datapath + Branch

```
Cond = (Reg[rs] – Reg[rt]) == 0;
if (Cond)
   PC = PC + 4 + SignExtend(imm)*4;
else
   PC = PC + 4;
```



Instruction Fetch + Branch

```
Cond = (Reg[rs] – Reg[rt]) == 0;
if (Cond)
   PC = PC + 4 + SignExtend(imm)*4;
else
   PC = PC + 4;
```