## RTL

> R: opcode=0110011
> I1: opcode= 0010011
> I2: opcode=0000011
> s: opcode=0100011
> B: opcode=1100011
> J: opcode=1101111
> IJ: opcode=1100111
> U1: opcode=0110111
> U2: opcode=0010111

> f31: funct3=0x0
> f32: funct3=0x1
> f33: funct3=0x2
> f34: funct3=0x3
> f35: funct3=0x4
> f36: funct3=0x5
> f37: funct3=0x6
> f38: funct3=0x7

> f71: funct7=0x00
> f72: funct7=0x20
> f73: funct7=0x01

## Type R

- ADD
  T0: `IR <- M[PC]`
  R.T1: `PC <- PC + 4`
  R.f31.f71.T2: `Reg[rd]<- Reg[rs1] + Reg[rs2]`
- SUB
  T0: `IR <- M[PC]`
  R.T1: `PC <- PC + 4`
  R.f31.f72.T2: `Reg[rd]<- Reg[rs1] - Reg[rs2]`
- XOR
  T0: `IR <- M[PC]`
  R.T1: `PC <- PC + 4`
  R.f35.f71.T2: `Reg[rd]<- Reg[rs1] ^ Reg[rs2]`
- OR
  T0: `IR <- M[PC]`
  R.T1: `PC <- PC + 4`
  R.f37.f71.T2: `Reg[rd]<- Reg[rs1] | Reg[rs2]`
- AND
  T0: `IR <- M[PC]`
  R.T1: `PC <- PC + 4`
  R.f38.f71.T2: `Reg[rd]<- Reg[rs1] & Reg[rs2]`
- SLL
  T0: `IR <- M[PC]`
  R.T1: `PC <- PC + 4`
  R.f32.f71.T2: `Reg[rd]<- Reg[rs1] << Reg[rs2][4:0]
- SRL
  T0: `IR <- M[PC]`
  R.T1: `PC <- PC + 4`
  R.f36.f71.T2: `Reg[rd]<- Reg[rs1] >> Reg[rs2][4:0]`

- SRA
  T0: `IR <- M[PC]`
  R.T1: `PC <- PC + 4`
  R.f36.f72.T2: `Reg[rd]<- signed(Reg[rs1]) >> Reg[rs2][4:0]`
- SLT
  T0: `IR <- M[PC]`
  T1: `PC <- PC + 4`
  R.f33.f71.T2: `Reg[rd] <- (signed(Reg[rs1]) < signed(Reg[rs2])) ? 1 : 0`
- SLTU
  T0: `IR <- M[PC]`
  R.T1: `PC <- PC + 4`
  R.f34.f71.T2: `Reg[rd] <- (Reg[rs1] < Reg[rs2]) ? 1 : 0`
- Multiply Extension

  ```
  - MUL
   T0: `IR <- M[PC]`
   R.T1: `PC <- PC + 4`
   R.f31.f73.T2: `Reg[rd]<- (Reg[rs1] * Reg[rs2])[31:0]`

  - MULH
   T0: `IR <- M[PC]`
   R.T1: `PC <- PC + 4`
   R.f32.f73.T2: `Reg[rd]<- (Reg[rs1] * Reg[rs2])[63:32]`



  - MULHSU
   T0: `IR <- M[PC]`
   R.T1: `PC <- PC + 4`
   R.f33.f73.T2: `Reg[rd] <- (signed(Reg[rs1]) * unsigned(Reg[rs2]))[63:32]`
  ```

- MULHU
 T0: `IR <- M[PC]`
 R.T1: `PC <- PC + 4`
 R.f34.f73.T2: `Reg[rd] <- (unsigned(Reg[rs1]) * unsigned(Reg[rs2]))[63:32]`

- DIV
 T0: `IR <- M[PC]`
 R.T1: `PC <- PC + 4`
 R.f35.f73.T2: `Reg[rd] <- signed(Reg[rs1]) / signed(Reg[rs2])`

- DIVU
 T0: `IR <- M[PC]`
 R.T1: `PC <- PC + 4`
 R.f36.f73.T2: `Reg[rd] <- unsigned(Reg[rs1]) / unsigned(Reg[rs2])`

- REM
 T0: `IR <- M[PC]`
 R.T1: `PC <- PC + 4`
 R.f37.f73.T2: `Reg[rd] <- signed(Reg[rs1]) % signed(Reg[rs2])`

- REMU
 T0: `IR <- M[PC]`

```
R.T1: `PC <- PC + 4`
R.f38.f73.T2: `Reg[rd] <- unsigned(Reg[rs1]) % unsigned(Reg[rs2])`
```

## Type I

- addi
  T0: `IR <- M[PC]`
  I1.T1: `PC <- PC + 4`
  I1.f31.T2: `Reg[rd] <- Reg[rs1]+ imm`
- ori
  T0: `IR <- M[PC]`
  I1.T1: `PC <- PC + 4`
  I1.f37.T2: `Reg[rd] <- Reg[rs1]| imm`
- andi
  T0: `IR <- M[PC]`
  I1.T1: `PC <- PC + 4`
  I1.f38.T2: `Reg[rd] <- Reg[rs1]& imm`
- slli
  T0: `IR <- M[PC]`
  I1.T1: `PC <- PC + 4`
  I1.f32.T2: `Reg[rd] <- Reg[rs1]<< imm[4:0]`
- srli
  T0: `IR <- M[PC]`
  I1.T1: `PC <- PC + 4`
  I1.f36.T2: `Reg[rd] <- Reg[rs1]>> imm[4:0]`
- lb
  T0: `IR <- M[PC]`
  I2.T1: `PC <- PC + 4`

I2.f31.T2: `MAR <- Reg[rs1] + imm`
I2.f31.T3: `MDR <- M[MAR]`
I2.f31.T4: `Reg[rd] <- sign_extend(MDR[7:0])`

- lh
  T0: `IR <- M[PC]`
  I2.T1: `PC <- PC + 4`
  I2.f32.T2: `MAR <- Reg[rs1] + imm`
  I2.f32.T3: `MDR <- M[MAR]`
  I2.f32.T4: `Reg[rd] <- sign_extend(MDR[15:0])`
- lw
  T0: `IR <- M[PC]`
  I2.T1: `PC <- PC + 4`
  I2.f33.T2: `MAR <- Reg[rs1] + imm`
  I2.f33.T3: `MDR <- M[MAR]`
  I2.f33.T4: `Reg[rd] <- sign_extend(MDR[31:0])`
- lbu
  T0: `IR <- M[PC]`
  I2.T1: `PC <- PC + 4`
  I2.f35.T2: `MAR <- Reg[rs1] + imm`
  I2.f35.T3: `MDR <- M[MAR]`
  I2.f35.T4: `Reg[rd] <- zero_extend(MDR[7:0])`
- lhu
  T0: `IR <- M[PC]`
  I2.T1: `PC <- PC + 4`
  I2.f36.T2: `MAR <- Reg[rs1] + imm`
  I2.f36.T3: `MDR <- M[MAR]`
  I2.f36.T4: `Reg[rd] <- zero_extend(MDR[15:0])`
- jalr
  T0: `IR <- M[PC]`

IJ.f31.T1: `Temp <- PC + 4, A <- Reg[rs1]`
IJ.f31.T2: ``Reg[rd] <- Temp, PC <- (A + imm) & ~1

## Type S

- sb
  T0: `IR <- M[PC]`
  s.T1: `PC <- PC + 4`
  s.f31.T2: `MAR <- Reg[rs1] + imm`
  s.f31.T3: `M[MAR] <- Reg[rs2][7:0]`
- sh
  T0: `IR <- M[PC]`
  s.T1: `PC <- PC + 4`
  s.f32.T2: `MAR <- Reg[rs1] + imm`
  s.f32.T3: `M[MAR] <- Reg[rs2][15:0]`
- sw
  T0: `IR <- M[PC]`
  s.T1: `PC <- PC + 4`
  s.f33.T2: `MAR <- Reg[rs1] + imm`
  s.f33.T3: `M[MAR] <- Reg[rs2][31:0]`

## Type B

- beq
  T0: `IR <- M[PC]`
  B.T1: `PC_temp <- PC`
  B.f31.T2: `if (Reg[rs1] == Reg[rs2]) PC <- PC_temp + imm else PC <- PC_temp + 4`
- bne
  T0: `IR <- M[PC]`

B.T1: `PC_temp <- PC`
B.f32.T2: `if (Reg[rs1] != Reg[rs2]) PC <- PC_temp + imm else PC <- PC_temp + 4`

- blt
  T0: `IR <- M[PC]`
  B.T1: `PC_temp <- PC`
  B.f35.T2: `if (signed(Reg[rs1]) < signed(Reg[rs2])) PC <- PC_temp + imm else PC <- PC_temp + 4`
- bge
  T0: `IR <- M[PC]`
  B.T1: `PC_temp <- PC`
  B.f36.T2: `if (signed(Reg[rs1]) >= signed(Reg[rs2])) PC <- PC_temp + imm else PC <- PC_temp + 4`
- bltu
  T0: `IR <- M[PC]`
  B.T1: `PC_temp <- PC`
  B.f37.T2: `if (Reg[rs1] < Reg[rs2]) PC <- PC_temp + imm else PC <- PC_temp + 4`
- bgeu
  T0: `IR <- M[PC]`
  B.T1: `PC_temp <- PC`
  B.f38.T2: `if (Reg[rs1] >= Reg[rs2]) PC <- PC_temp + imm else PC <- PC_temp + 4`

## Type U

- lui
  T0: `IR <- M[PC]`
  T1: `PC <- PC + 4`
  ** u1.T2: `Reg[rd] <- imm << 12`
- auipc
  T0: `IR <- M[PC]`
  u2.T1: `PC_temp <- PC, PC <- PC_temp + 4`
  **u2.T2: `Reg[rd] <-PC_temp + (imm << 12)`

## Type J

- jal
  T0: `IR <- M[PC]`
  J.T1: `Temp <- PC + 4`
  J.T2: `PC <- PC + imm`, `Reg[rd] <- Temp