Fetch States

```
F0: memreq.addr ← PC; A ← PC;
      RD ← M[memreq.addr]
F1: IR ← RD
F2: PC ← A + 4; goto inst
```

add rd, rs1, rs2

```
A0: A ← R[rs1]
A1: B ← R[rs2]
A2: R[rd] ← A + B
```

addi rd, rs1, imm

```
AI0: A ← R[rs1]
AI1: B ← sext(imm_i)
AI2: R[rd] ← A + B
```

mul rd, rs1, rs2

```
 M0: A ← R[x0]
 M1: B ← R[rs1]
 M2: C ← R[rs2]
 M3: A ← C[0] ? A + B : A;
        B ← B << 1; C ← C >> 1
 M4: A ← C[0] ? A + B : A;
        B ← B << 1; C ← C >> 1
 ...
M35: R[rd] ← C[0] ? A + B : A;
        goto F0
```

lw rd, imm(rs1)

```
L0: A ← R[rs1]
L1: B ← sext(imm_i)
L2: memreq.addr ← A + B;
      RD ← M[memreq.addr]
L3: R[rd] ← RD; goto F0
```

sw rs2, imm(rs1)

```
S0: WD ← R[rs2]
S1: A ← R[rs1]
S2: B ← sext(imm_s)
S3: memreq.addr ← A + B
      M[memreq.addr] ← WD;
      goto F0
```

jal rd, imm

```
JA0: R[rd] ← PC
JA1: B ← sext(imm_j)
JA2: PC ← A + B; goto F0
```

jr rs1

```
JR0: PC ← R[rs1]; goto F0
```

bne rs1, rs2, imm

```
B0: A ← R[rs1]
B1: B ← R[rs2]
B2: B ← sext(imm_b);
      if A == B goto F0
B3: A ← PC
B4: A ← A − 4
B5: PC ← A + B; goto F0
```