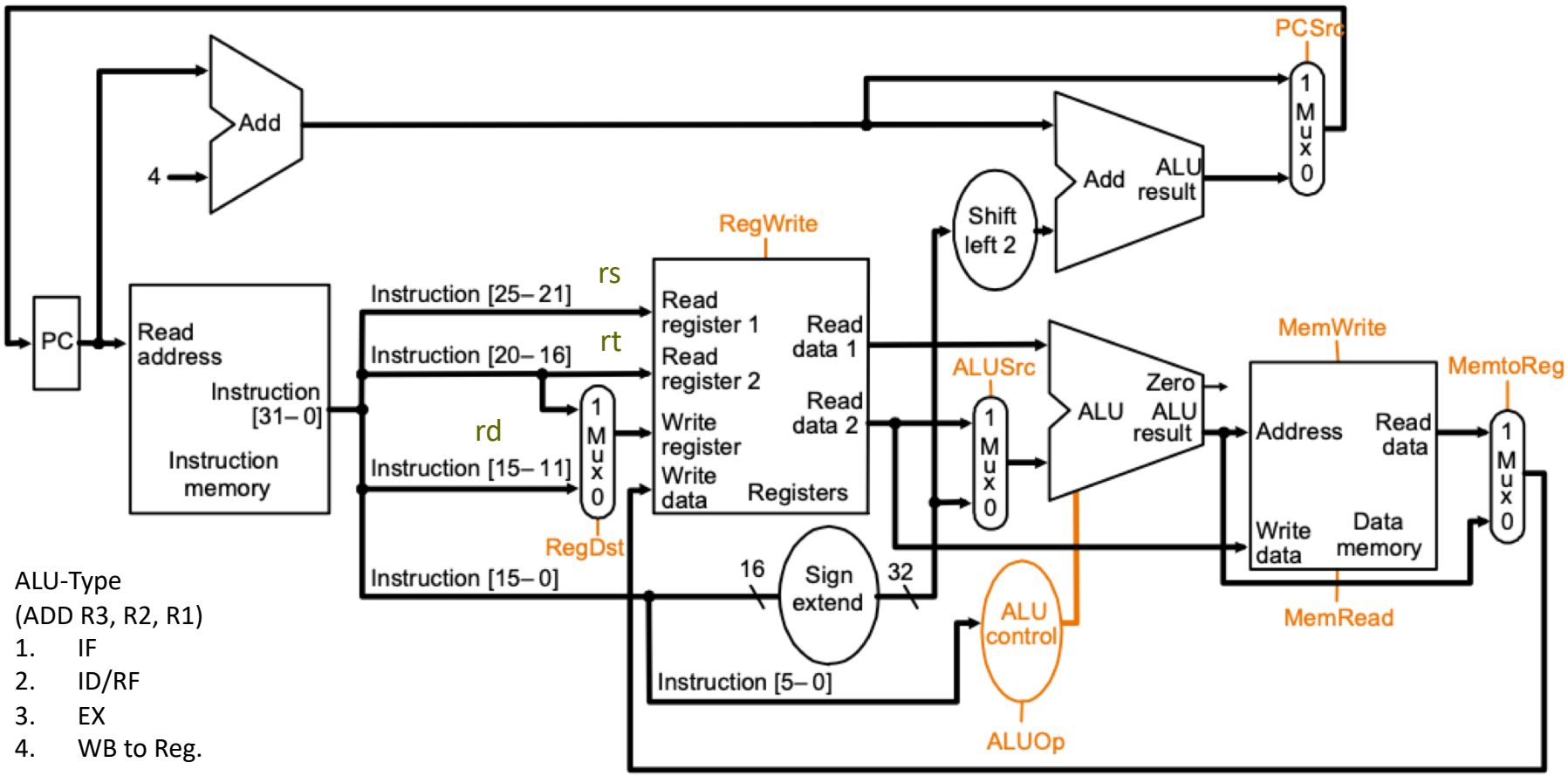
R-type Instructions – ADD Instruction



ALU-Type
(ADD R3, R2, R1)

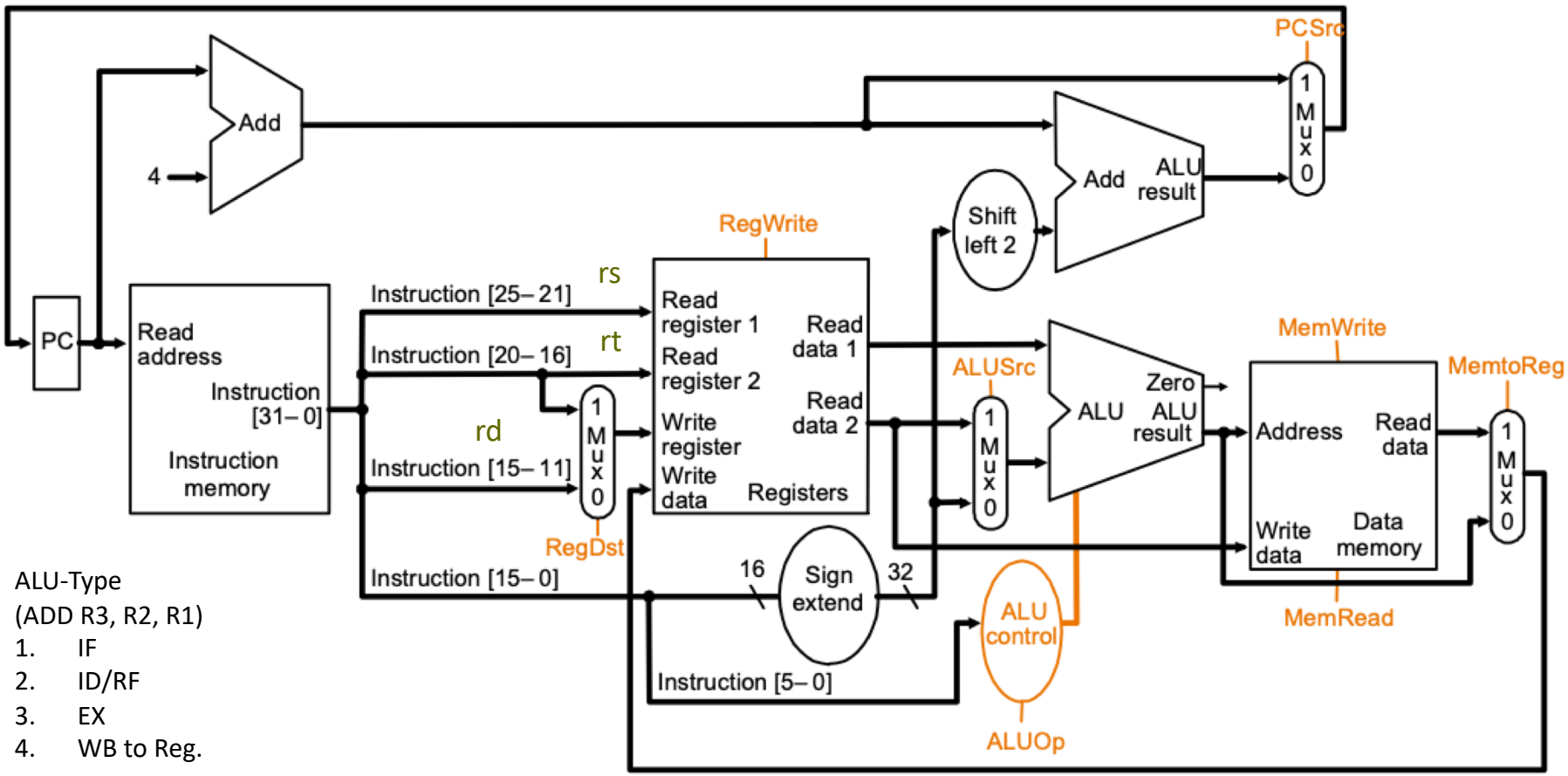
1. IF
2. ID/RF
3. EX
4. WB to Reg.

```
if MEM[PC] == ADD rd rs rt
                GPR[rd] ← GPR[rs] +
GPR[rt]
                PC ← PC + 4
```

0	rs	rt	rd	0	ADD
6-bit	5-bit	5-bit	5-bit	5-bit	6-bit

ADD rd, rs, rt

I-type Instructions - ADDI **Instruction**



ALU-Type
(ADD R3, R2, R1)

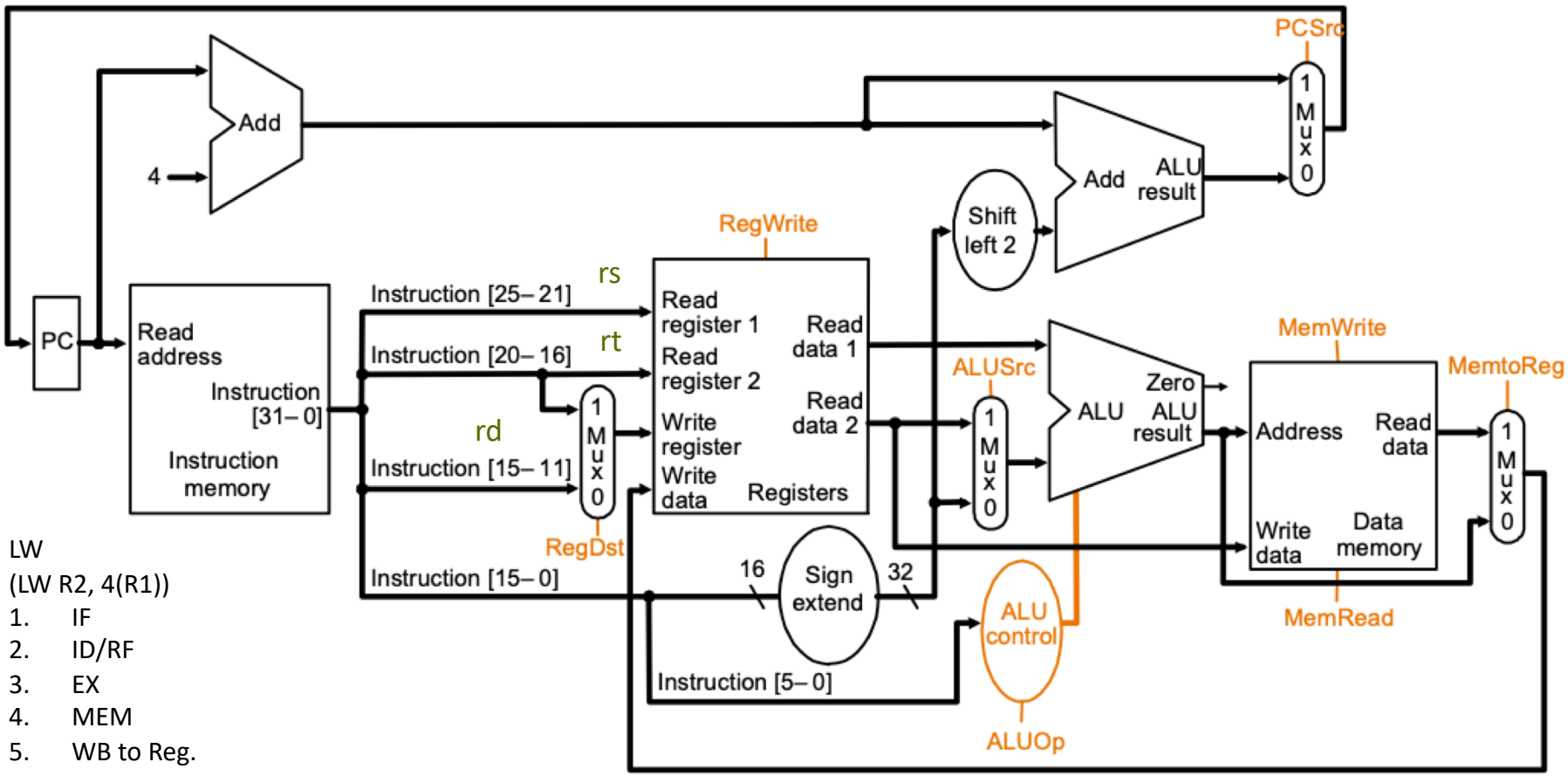
1. IF
2. ID/RF
3. EX
4. WB to Reg.

if MEM[PC] == ADDI rt rs immediate
GPR[rt] ← GPR[rs] + sign-extend (imm)
PC ← PC + 4

ADDI	rs	rt	immediate
6-bit	5-bit	5-bit	16-bit

ADDI rt, rs, immediate

I-type Instructions - **LW Instruction**



LW
(LW R2, 4(R1))

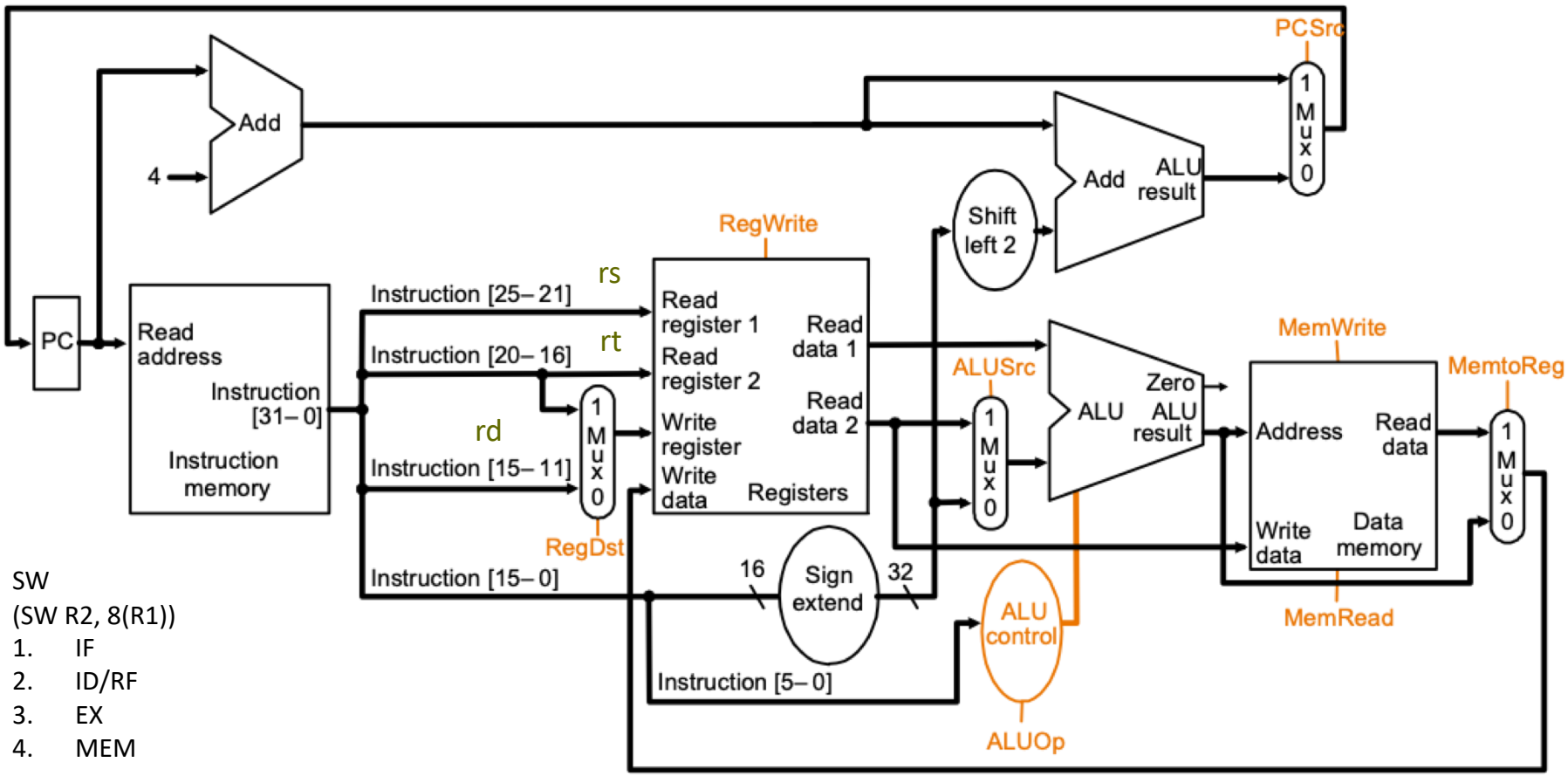
1. IF
2. ID/RF
3. EX
4. MEM
5. WB to Reg.

if MEM[PC]==LW rt offset$_{16}$ (base)
Addr = sign-extend(offset) + GPR[base]
GPR[rt] ← MEM[Addr]
PC ← PC + 4

LW	rs	rt	offset
6-bit	5-bit	5-bit	16-bit

LW rt, offset(rs)

I-type Instructions - **SW Instruction**



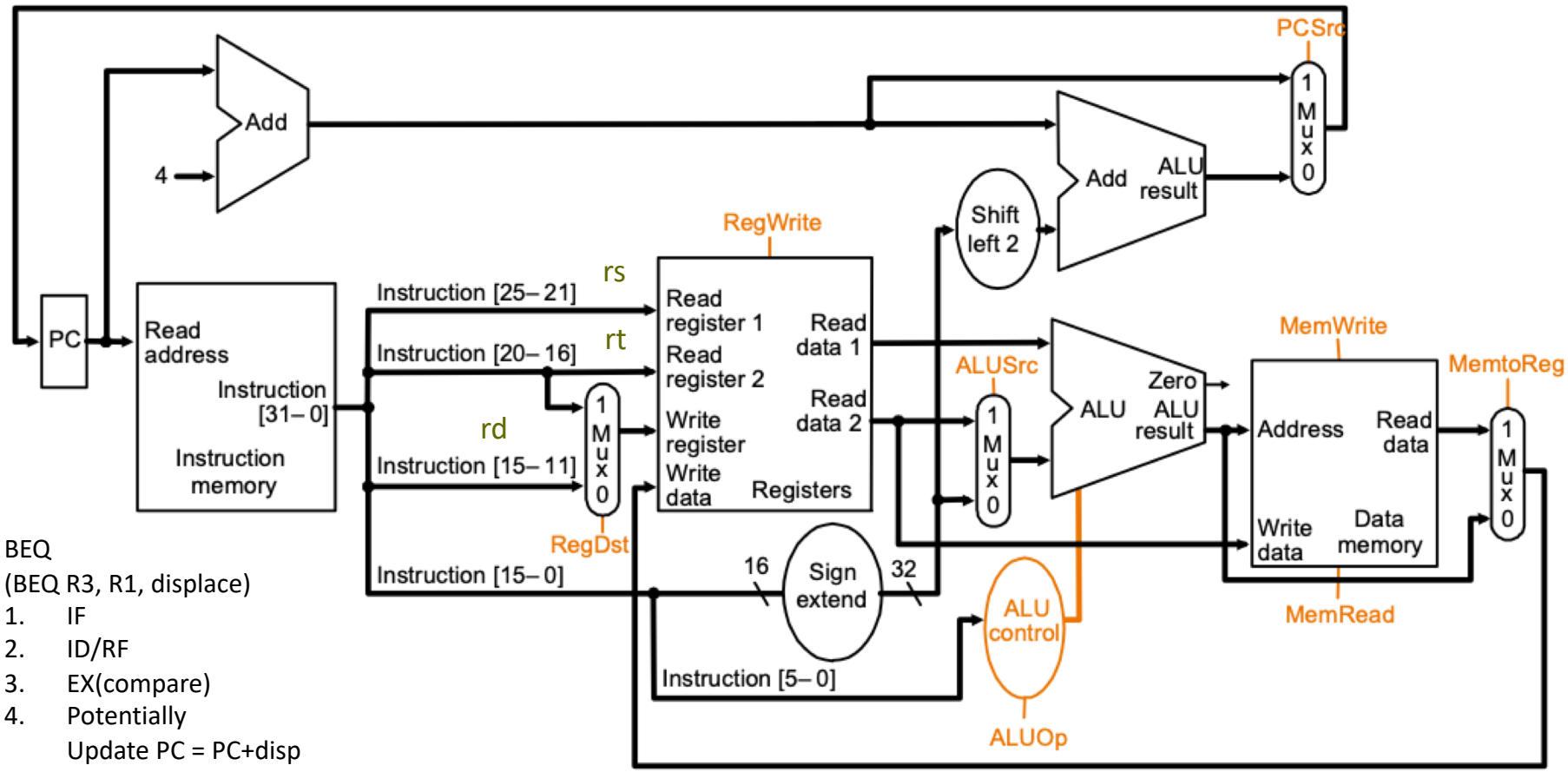
SW
(SW R2, 8(R1))
1. IF
2. ID/RF
3. EX
4. MEM

if MEM[PC]==SW rt offset$_{16}$ (base)
Addr = sign-extend(offset) + GPR[base]
MEM[Addr] ← GPR[rt]
PC ← PC + 4

SW	rs	rt	offset
6-bit	5-bit	5-bit	16-bit

SW rt, offset(rs)

I-type Instructions - **Branch Instructions**



BEQ
(BEQ R3, R1, displace)

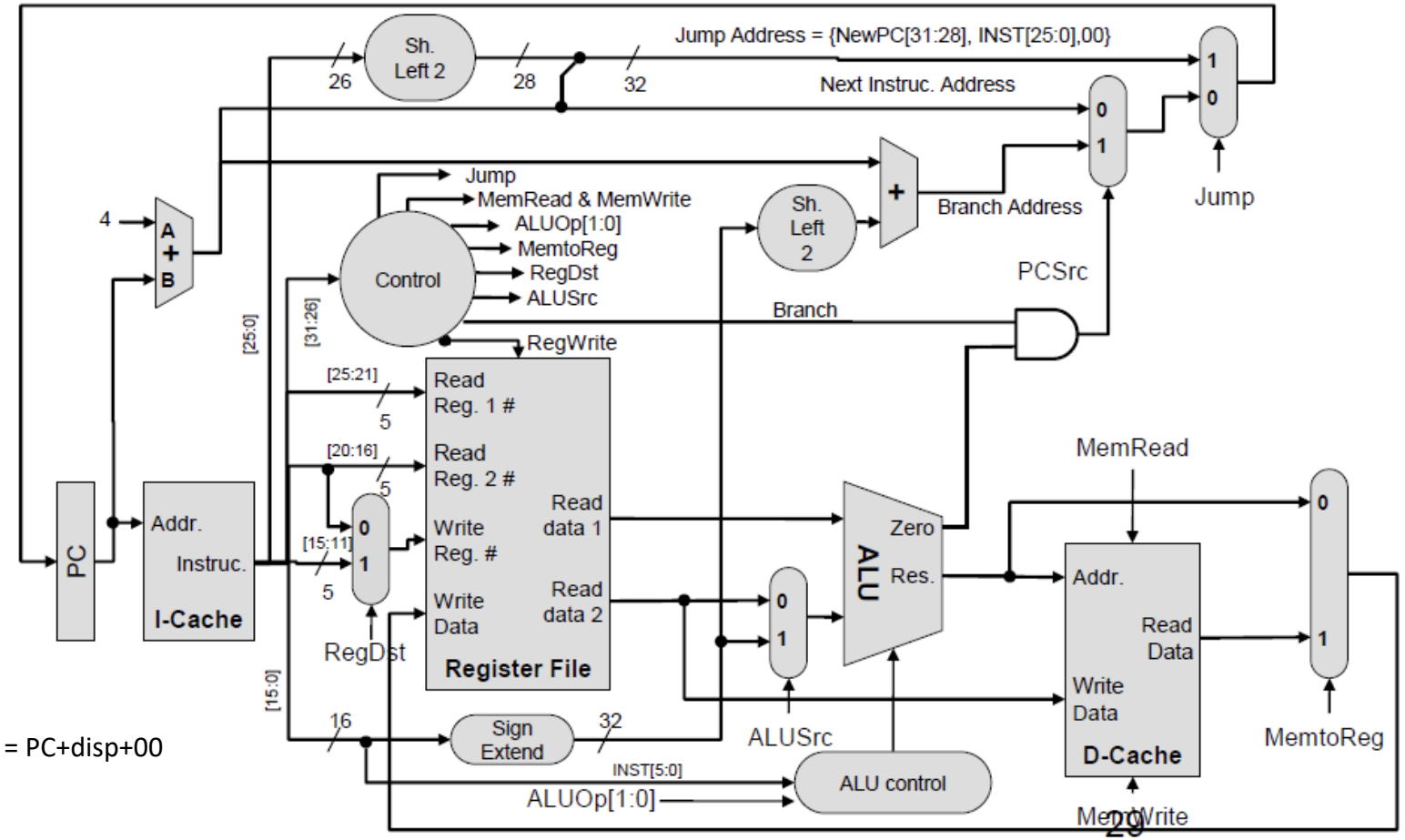
1. IF
2. ID/RF
3. EX(compare)
4. Potentially
   Update PC = PC+disp

if MEM[PC]==BEQ rs rt immediate$_{16}$
target = PC + 4 + sign-extend(immediate) x 4
if (GPR[rs]==GPR[rt]) PC ← target
else PC ← PC + 4

BEQ	rs	rt	immediate
6-bit	5-bit	5-bit	16-bit

BEQ rs, rt, immediate

J-type Instructions - **Jump Instructions**



J
(J displace)

1. IF
2. ID/RF
3. Update PC = PC+disp+00

if MEM[PC]==J immediate$_{26}$
target = { PC[31:28], immediate$_{26}$, 2'b00 }
PC ← target

J	immediate
6-bit	26-bit

J immediate