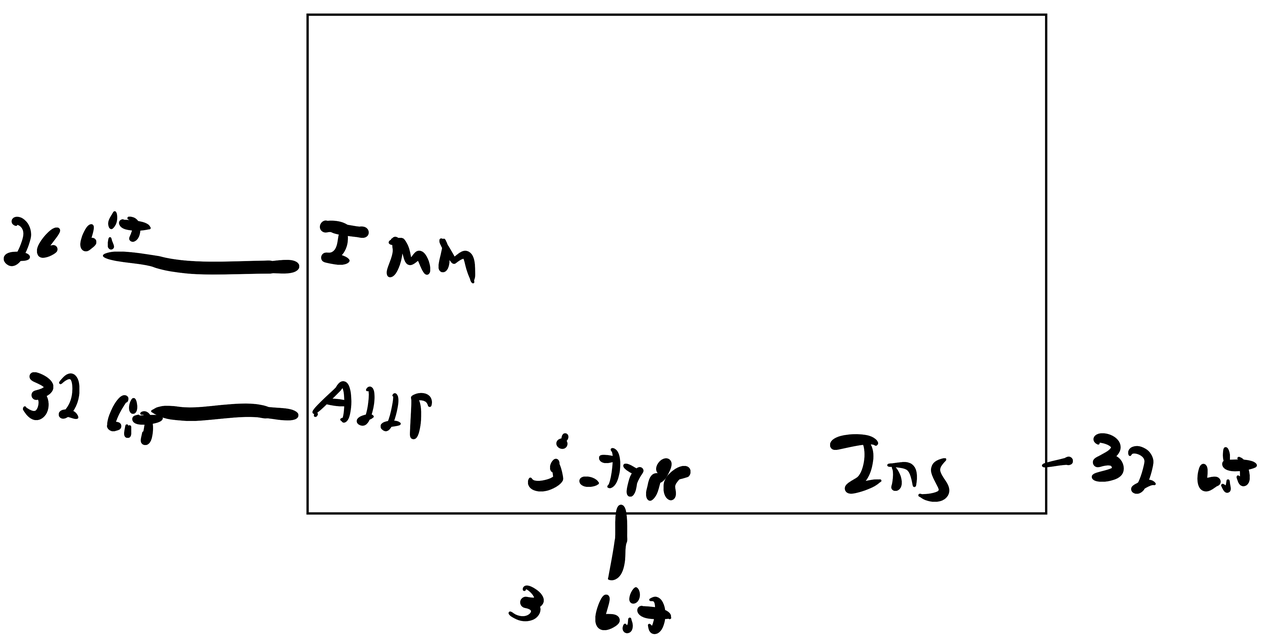
Fetch design

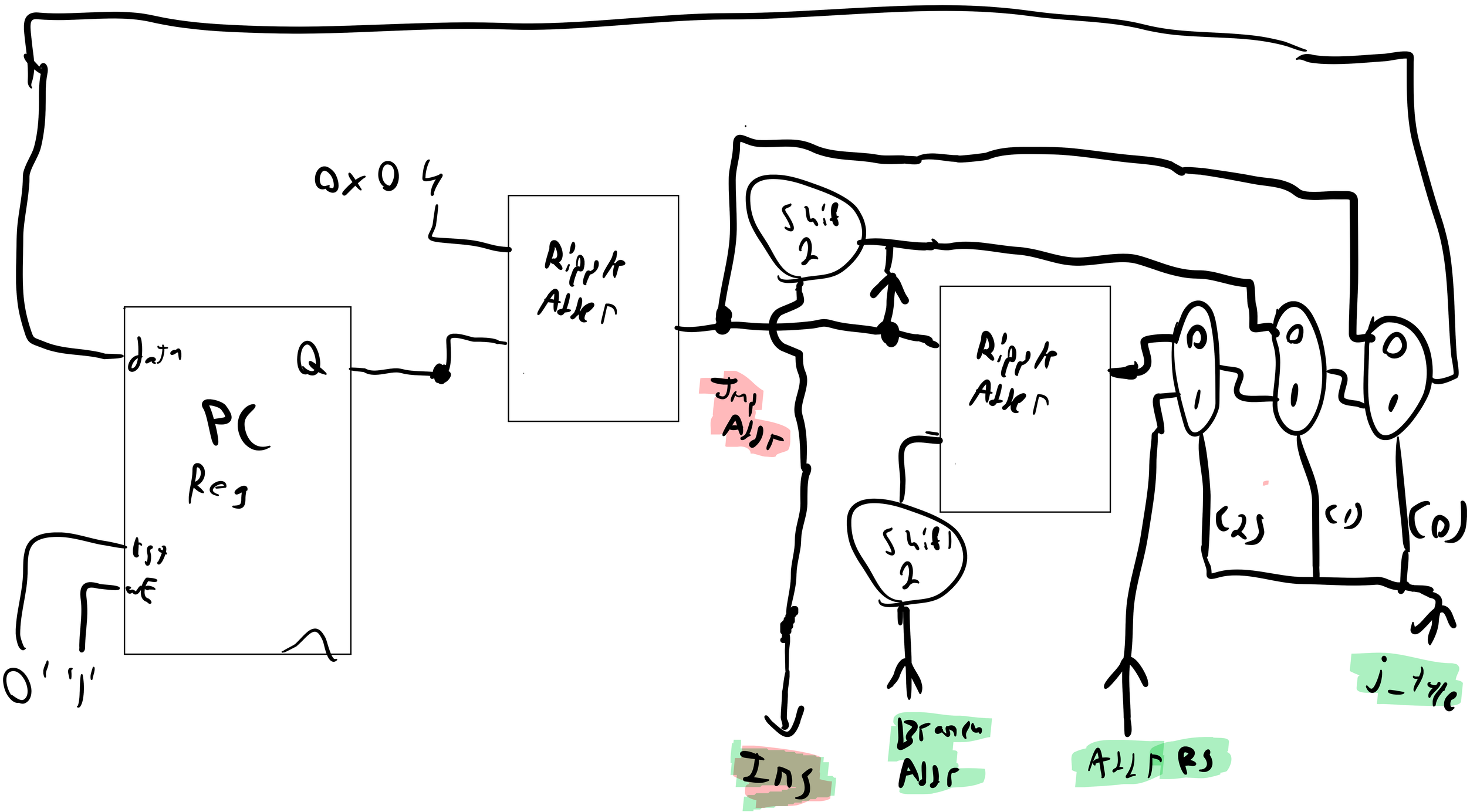
Functionality: Calculate and move to next instruction based on current instruction

j_type	lookup
Signals	Operation
x x 0	PC = PC + 4
x 0 1	PC = jmpAddr
0 1 1	PC = branchAddr
1 1 1	PC = R[rs]

Interface symbol



Fetch Schematic



Decode design

R type - JR

RegDST = 1

link = 0

ALUSrc = 0

Shamt = Shamt

memWrite = 0

memtoReg = 1

Jmp Ins = 0

branch = 0

:

JumpIns = 0

I type op =

8, 9, C, f, d, a, b

24, 00, 02

branch = 0

jump = 0

RegDST = 1

link = 0

ALUSrc = 1

Shamt = Shamt

memWrite = 0

memtoReg = 1

Jmp Ins = 0

Special Case instructions

lw ✓

sw ✓

j ✓

jal ✓

jr ✓

beq ✓

bne ✓

lw

RegDst = 0
Jump = 0
Branch = 0
MemToReg = 0
ALUOp = add
MemWrite = 0
ALUSrc = 1
JumpIns = 0
RegWrite = 1
Link = 0

sw

RegDst = x
Jump = 0
Branch = 0
MemToReg = x
ALUOp = add
MemWrite = 1
ALUSrc = 1
JumpIns = 0
RegWrite = 0
Link = 0

beq / bne

RegDst = x
Jump = 0
Branch = 1
MemToReg = x
ALUOp = Sub
MemWrite = 0
ALUSrc = 1
JumpIns = 0
RegWrite = 0
Link = 0

j

RegDst = x

Jump = 1

Branch = x

MemToReg = x

ALUOp = x

MemWrite = 0

ALUSrc = x

JumpIns = 0

RegWrite = 0

Link = 0

jal

RegDst = x

Jump = 1

Branch = x

MemToReg = x

ALUOp = x

MemWrite = 0

ALUSrc = x

JumpIns = 0

RegWrite = 1

Link = 1

jR

RegDst = x

Jump = x

Branch = x

MemToReg = x

ALUOp = x

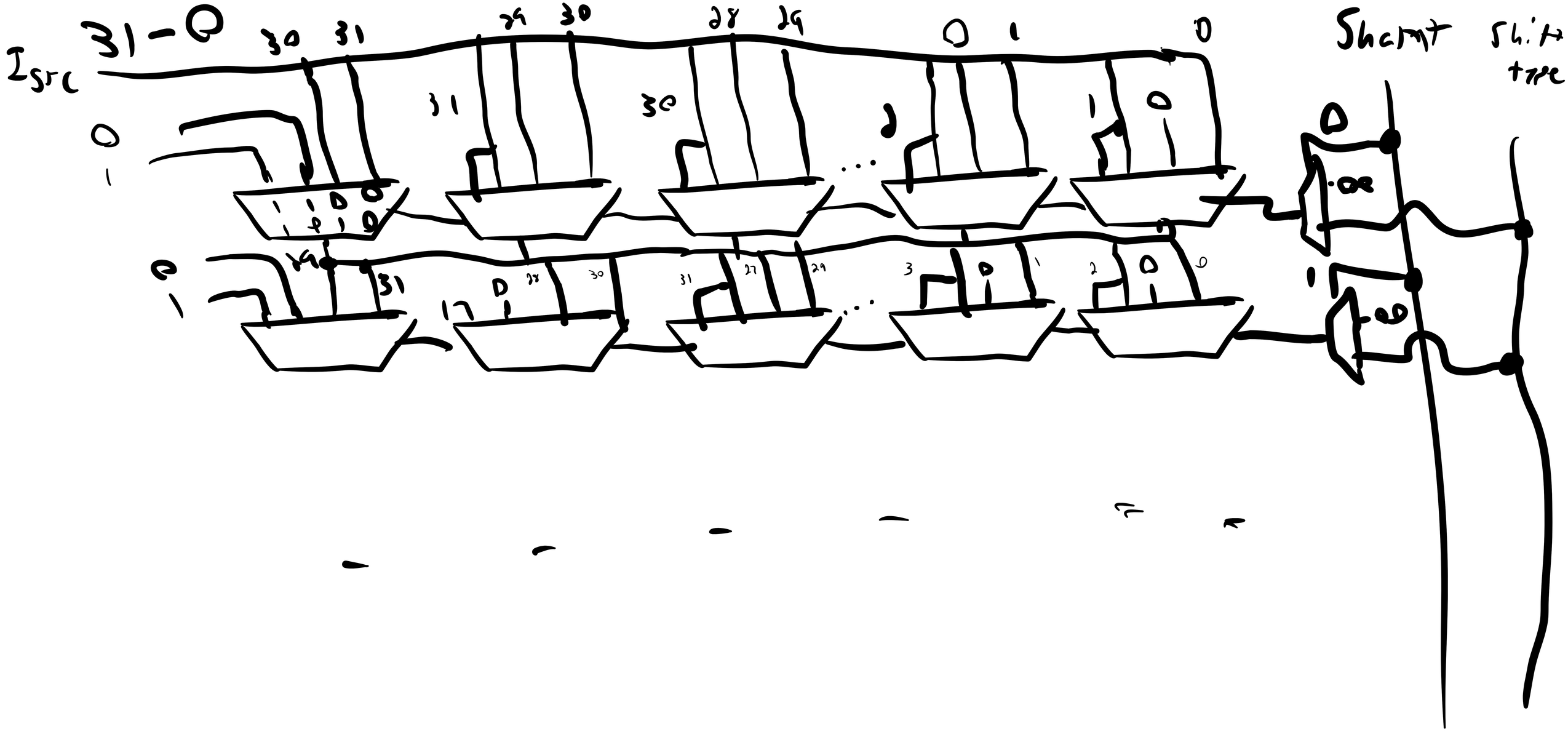
MemWrite = 0

ALUSrc = x

JumpIns = 1

RegWrite = 0

Link = 0


31-0
Shamt
Shift type

Sub Components

32 bit register