Single Cycle Architecture

Part 2

Office hours : this week

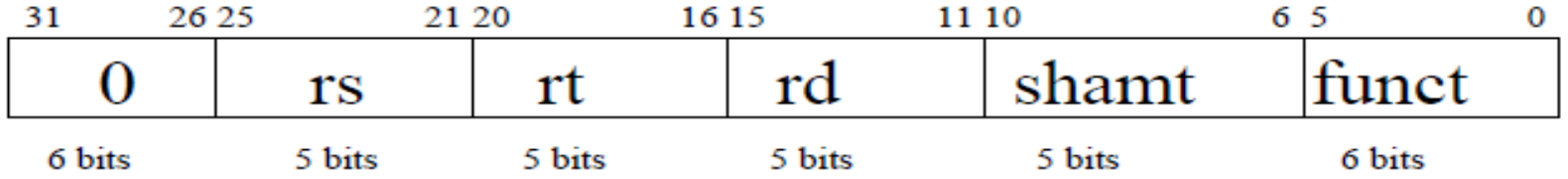
- A) Today 1-2pm my office
- B) Wed: 1-2:30
- C) Thursday 1:30-2:30
- Check Piazza for Sean's Office Hours Also and midterm related info

What Statement about memory in the Datapath is NOT true ☺

- A) the instruction and data memory are a type of RAM
- B) Data memory can be read from and written to
- C) The Registers contain memory in flip flops
- D) The Register file allows reading and writing to the registers
- E) The Instruction memory in the datapath allows reading of instructions and writing new instructions

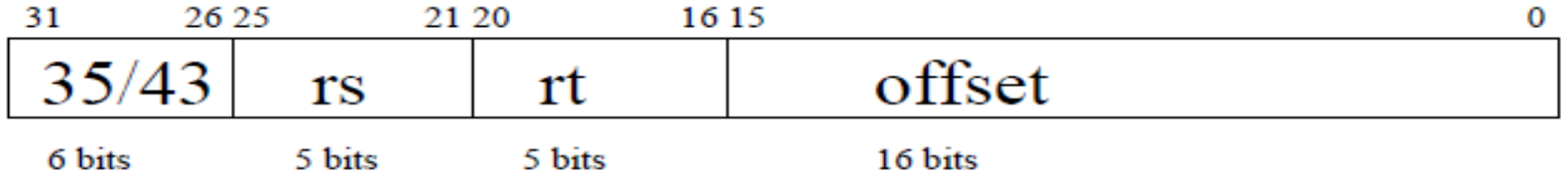
How the Instruction Bits are Broken up:

R-format: `add $t1, $t2, $t3` ⇒ `add rd,rs,rt`



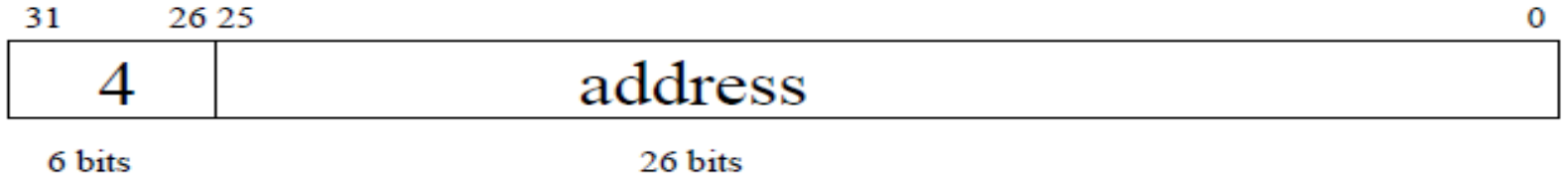
sll, sub

Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`



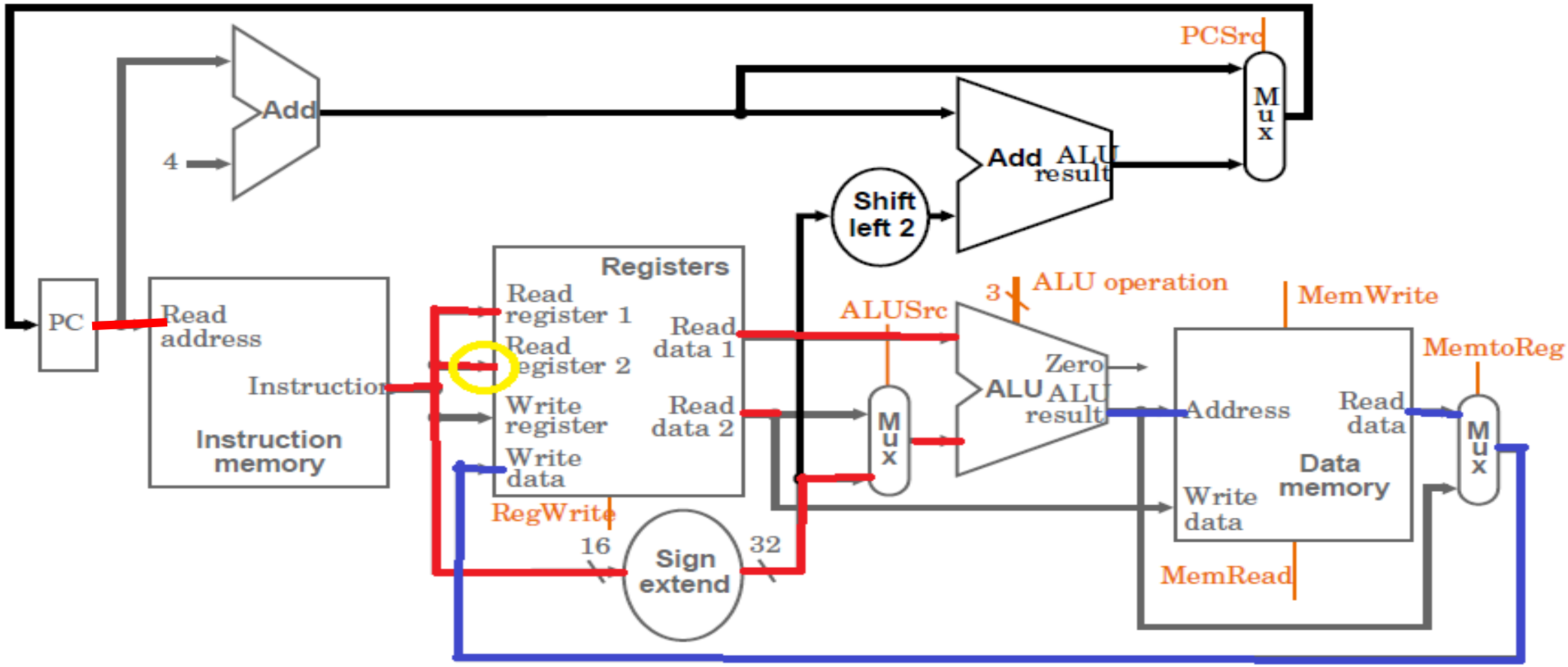
beq, subi, addi

Jump: `j 3000`



special

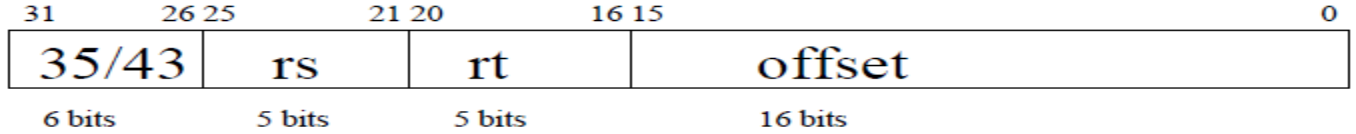
lw $t1, 100($t2)



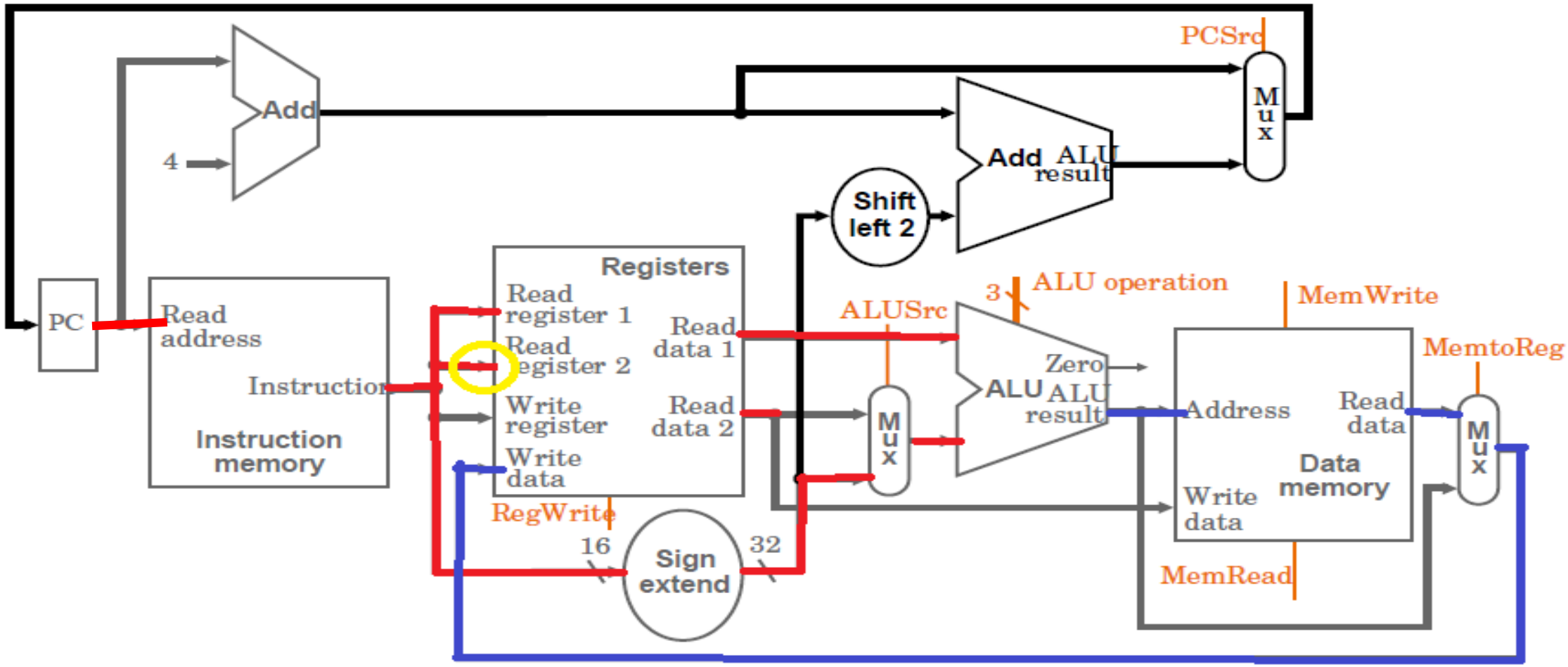
Once we access data memory → we now have 32 bits of data

* this data needs to be written to destination register (rt)
* therefore send data back to register file

Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`



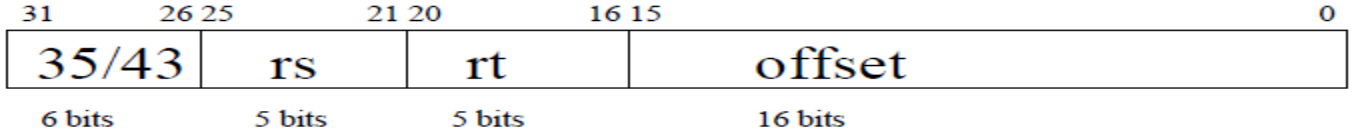
lw $t1, 100($t2)



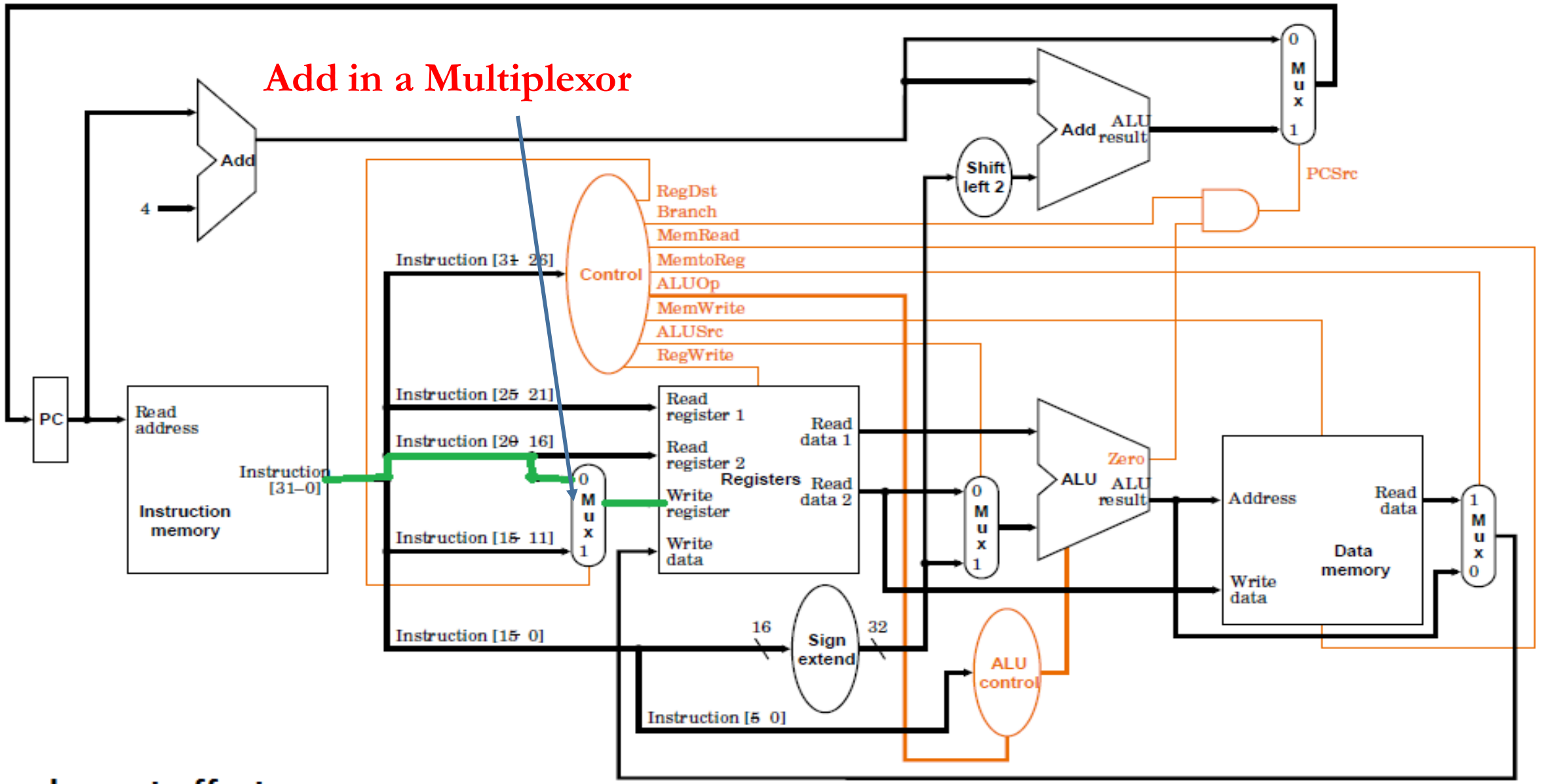
We need rt to be the Write Register

Need more hardware here : to connect rt as also an option for a write register

Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`

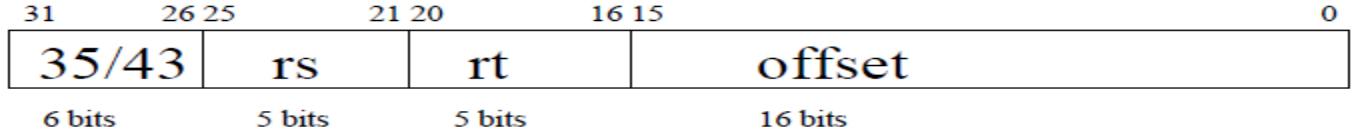


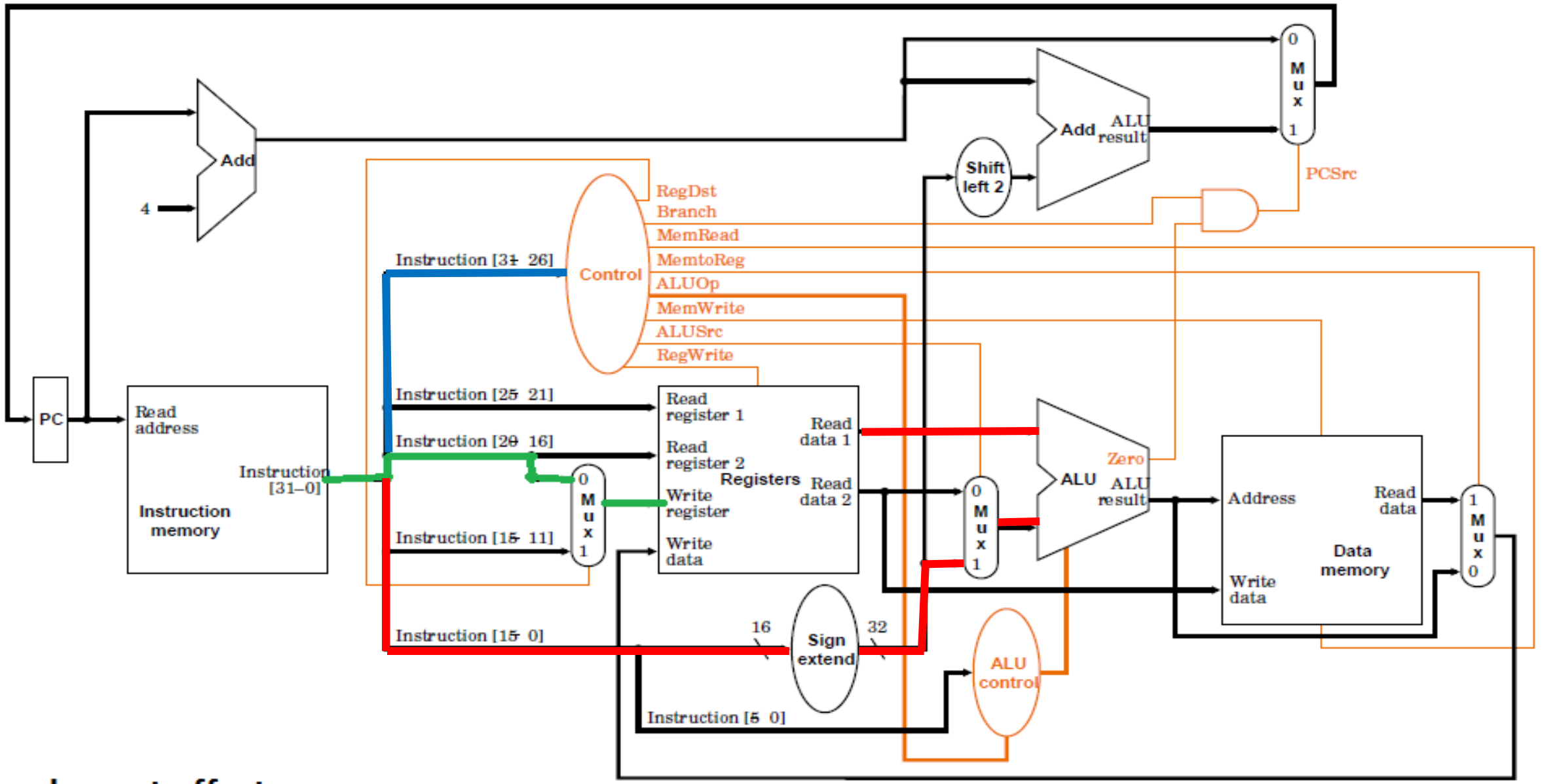
Add in a Multiplexor



lw rs rt offset

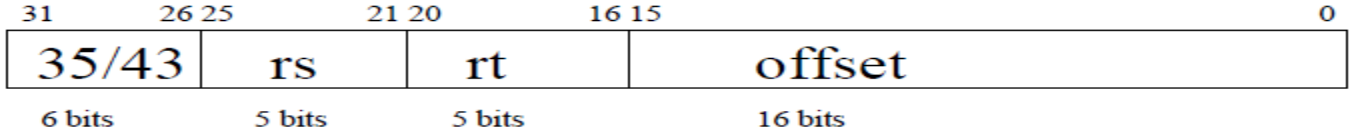
Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`

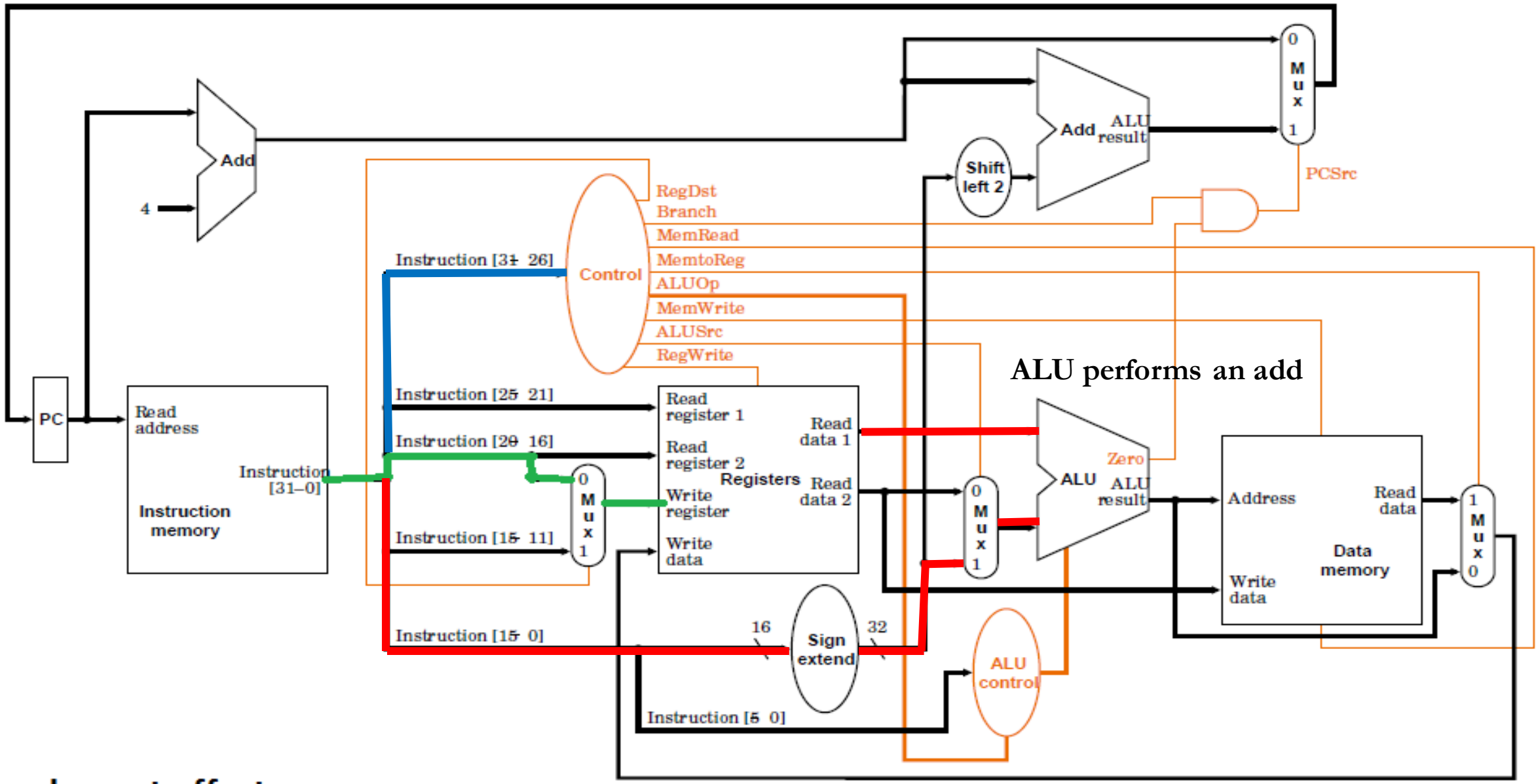




lw rs rt offset

Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`

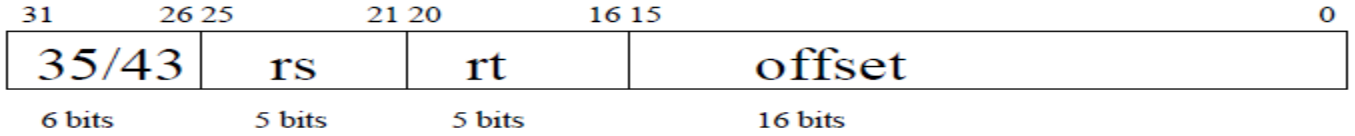


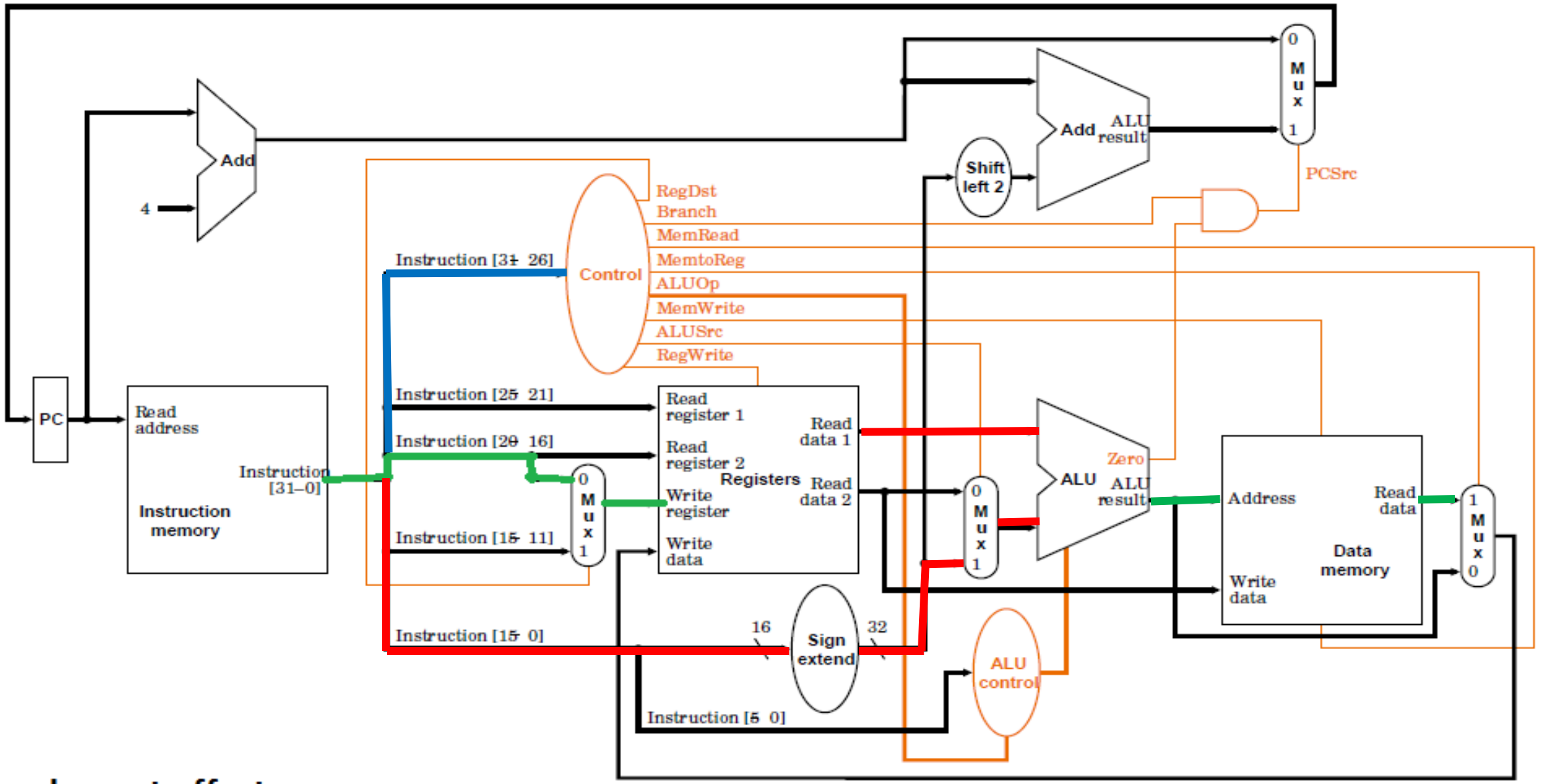


ALU performs an add

lw rs rt offset

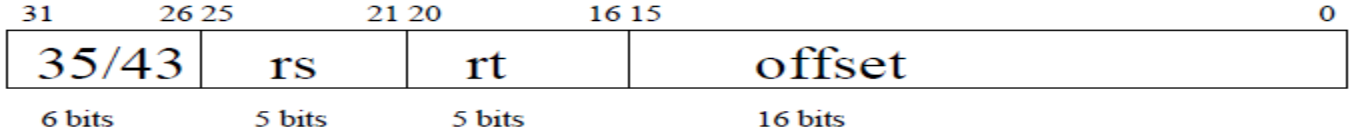
Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`

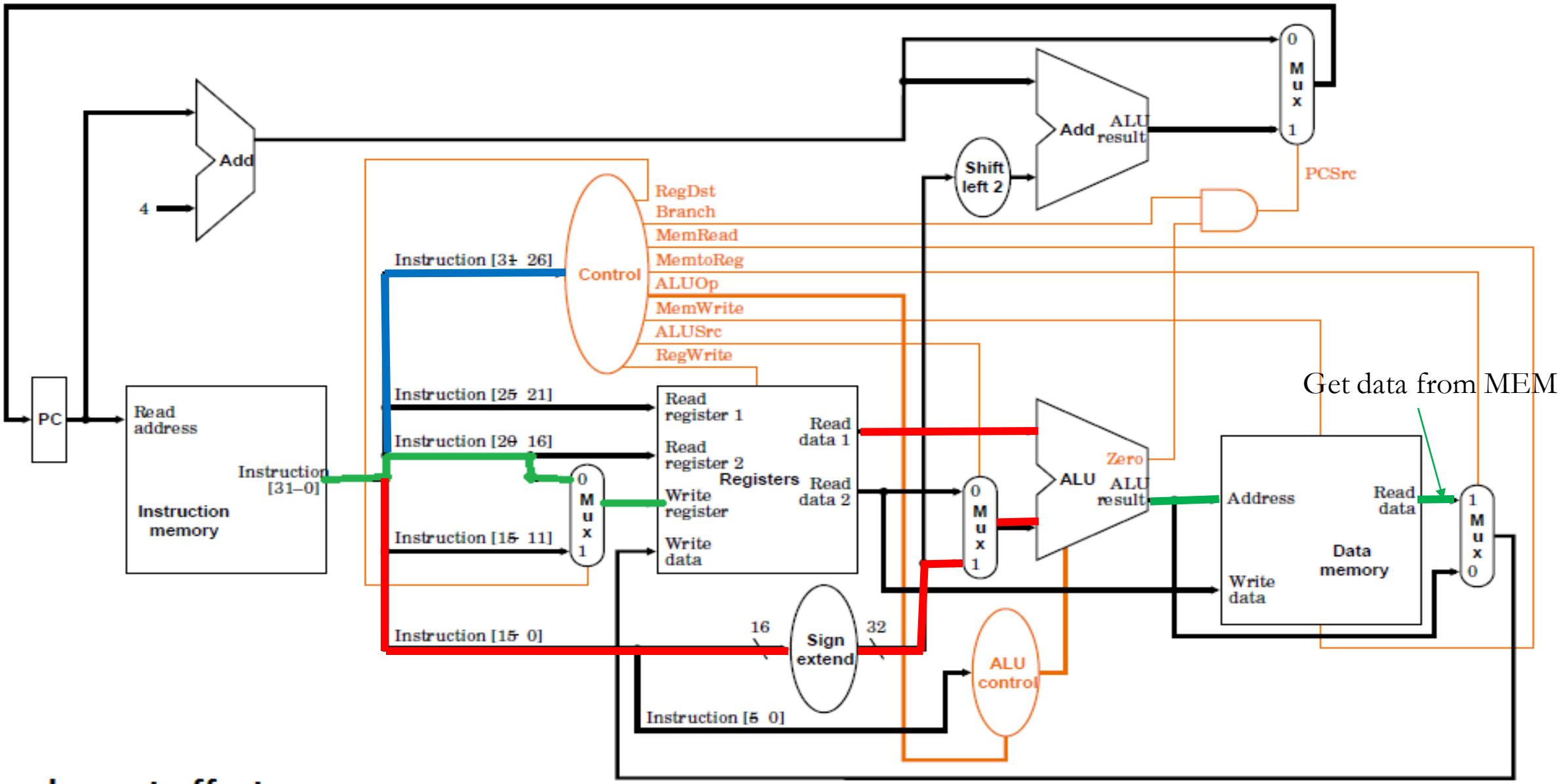




lw rs rt offset

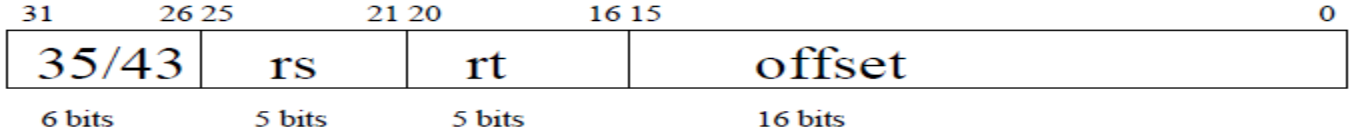
Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`

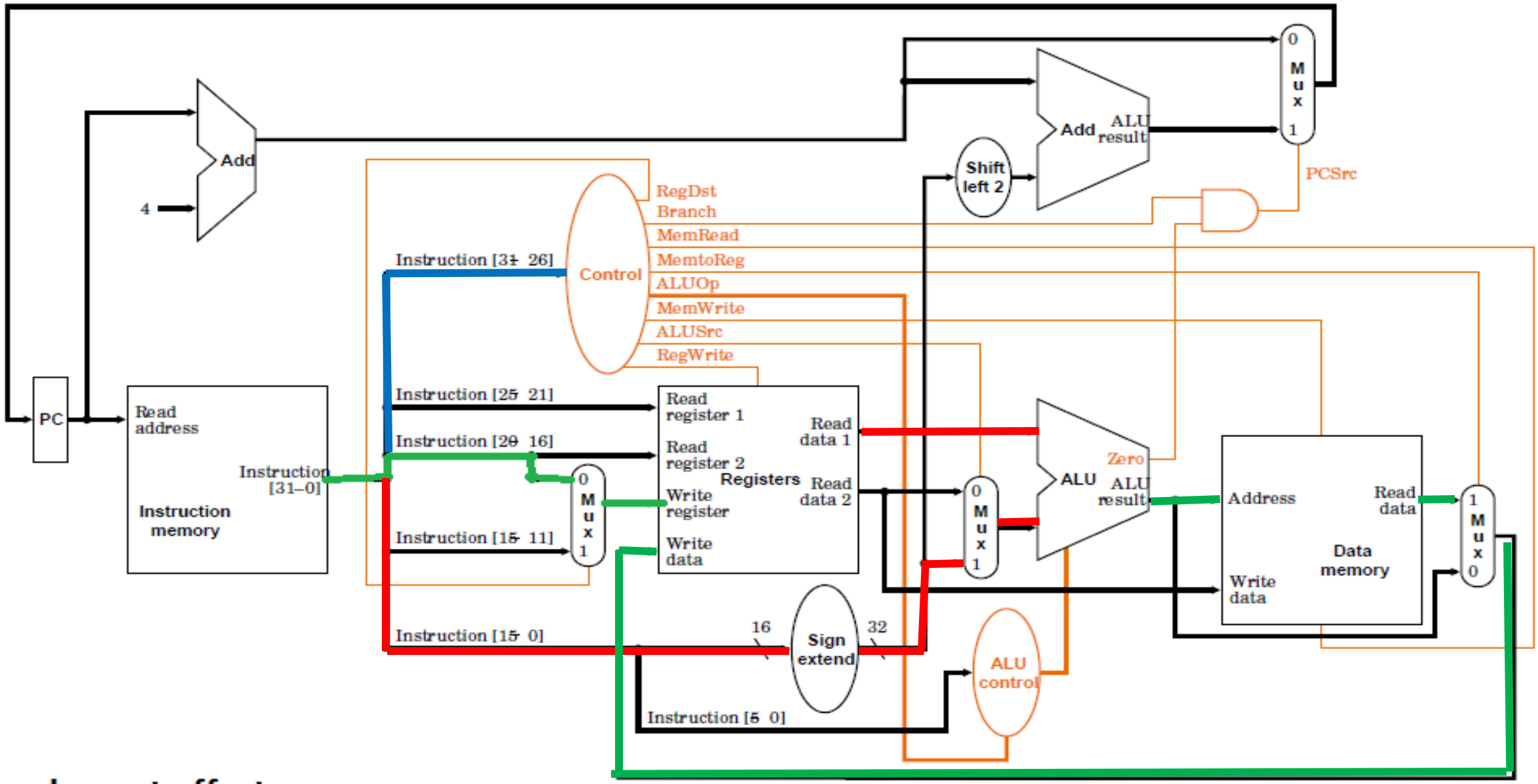




lw rs rt offset

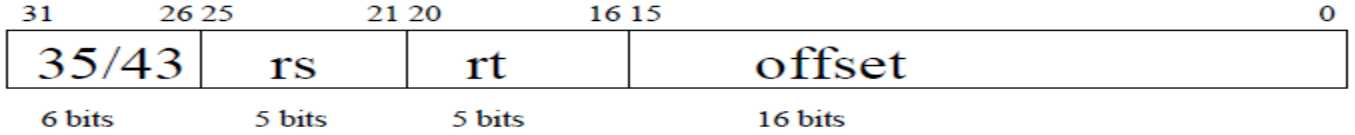
Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`

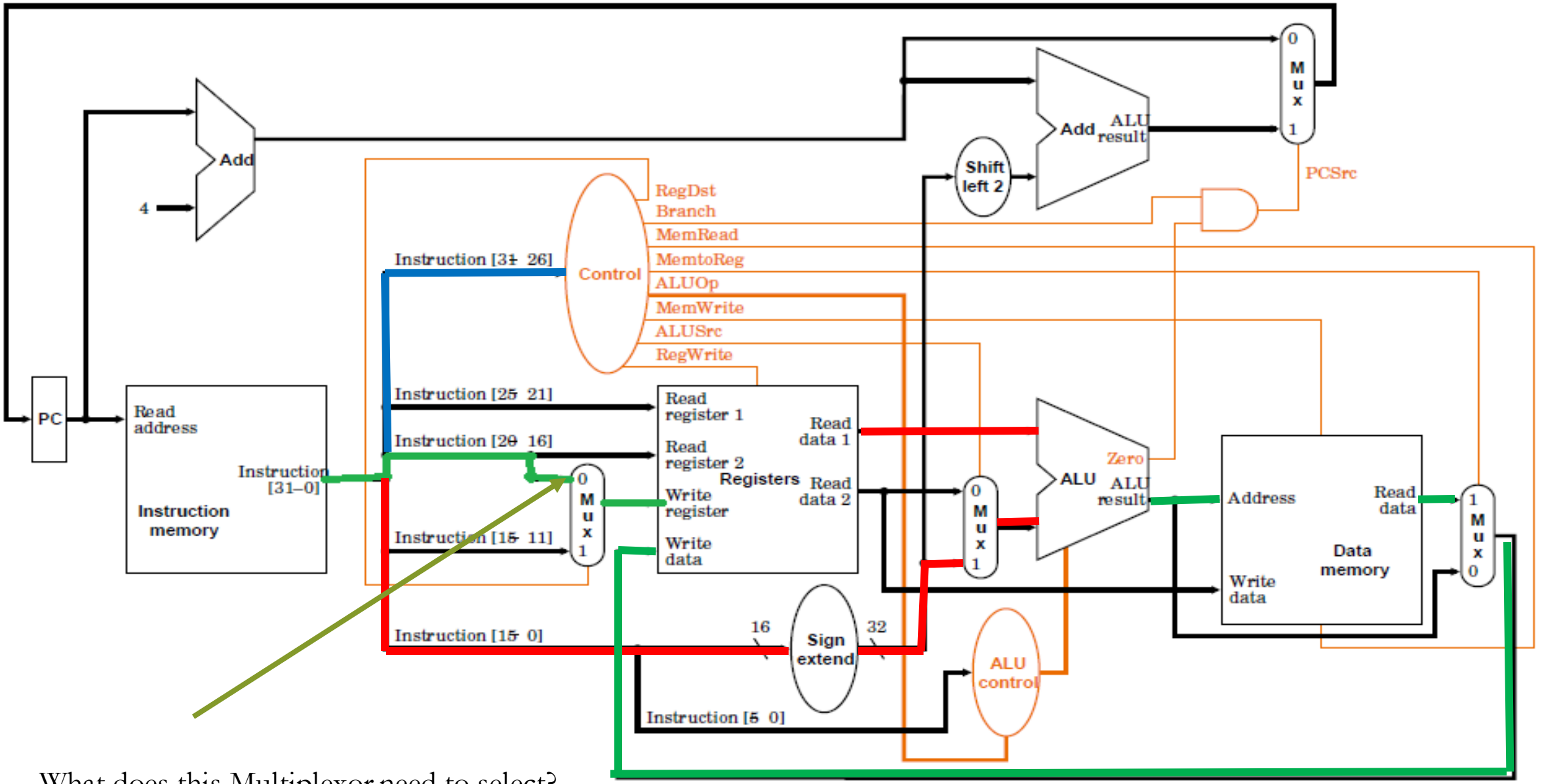




lw rs rt offset

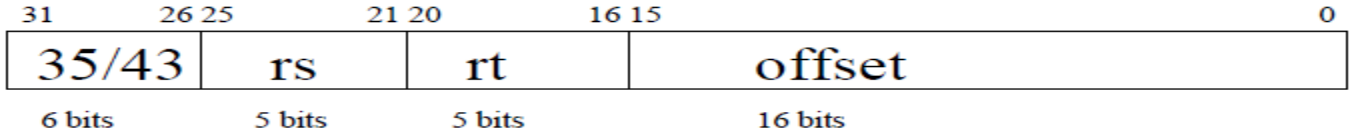
Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`

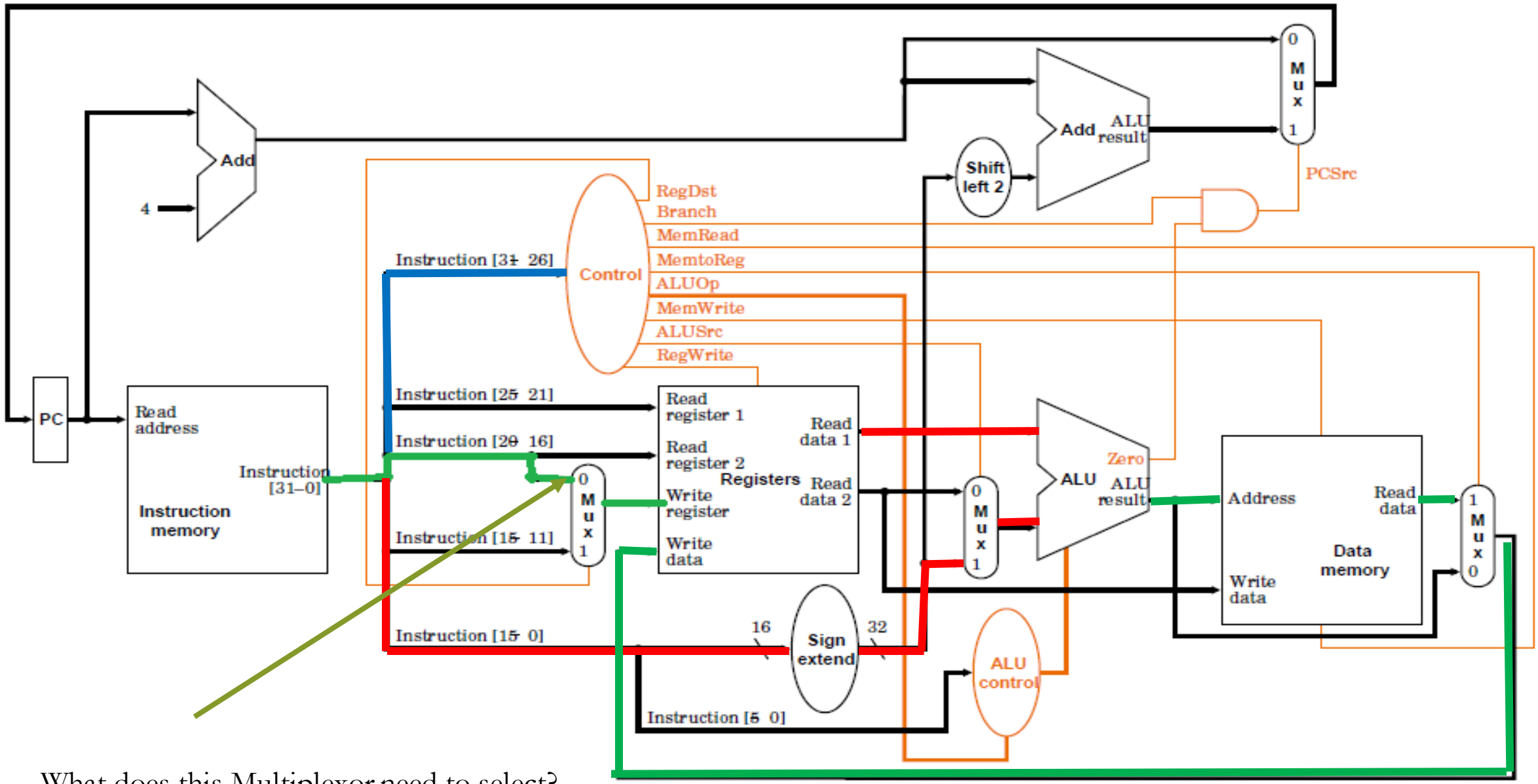




What does this Multiplexor need to select?

Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`



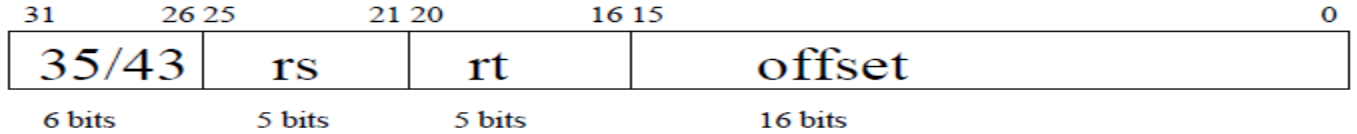


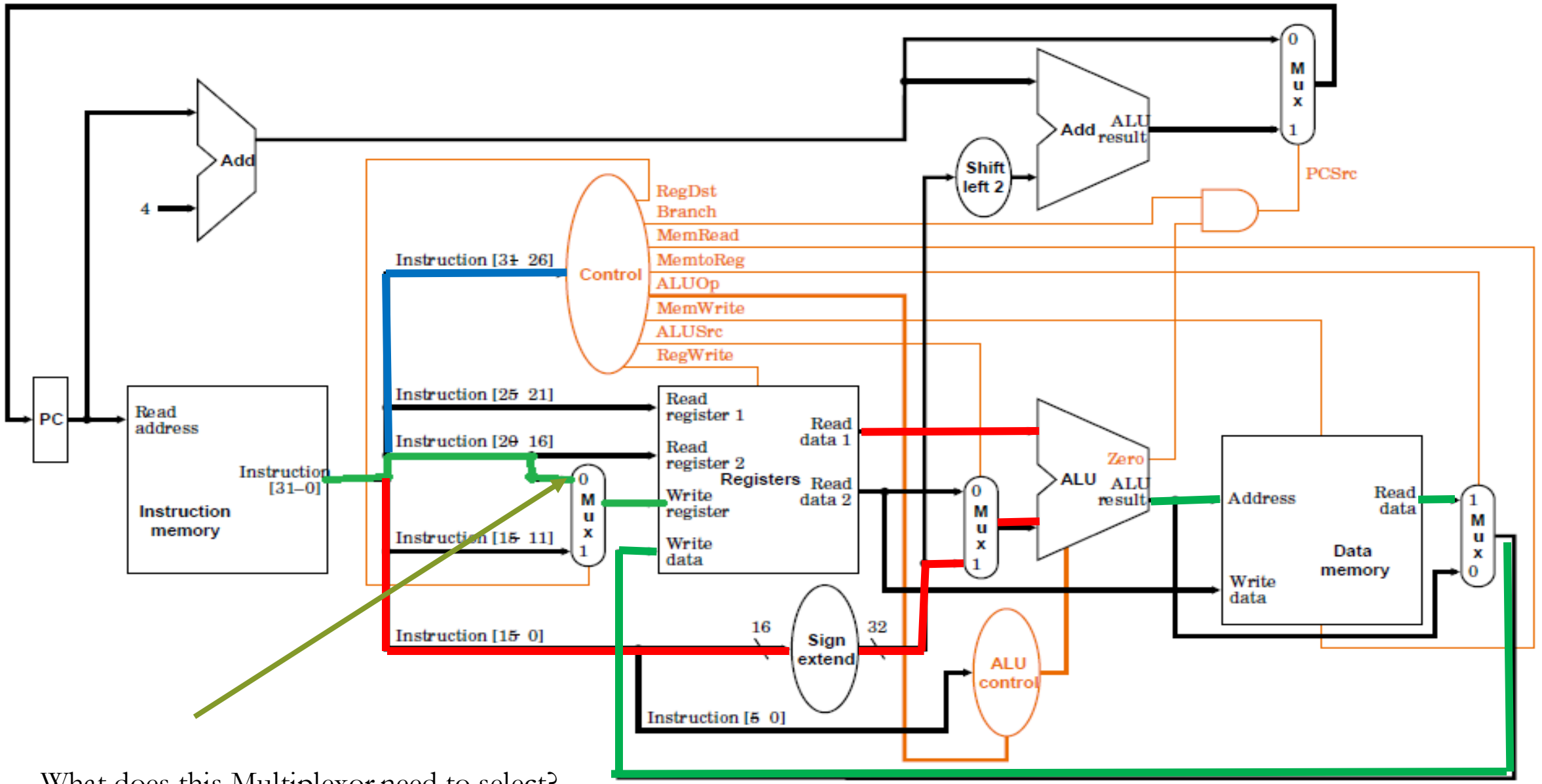
What does this Multiplexor need to select?

Zero

Option 1 is used for which instructions?

Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`





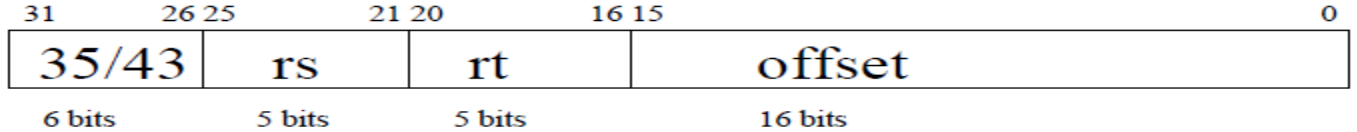
What does this Multiplexor need to select?

Zero

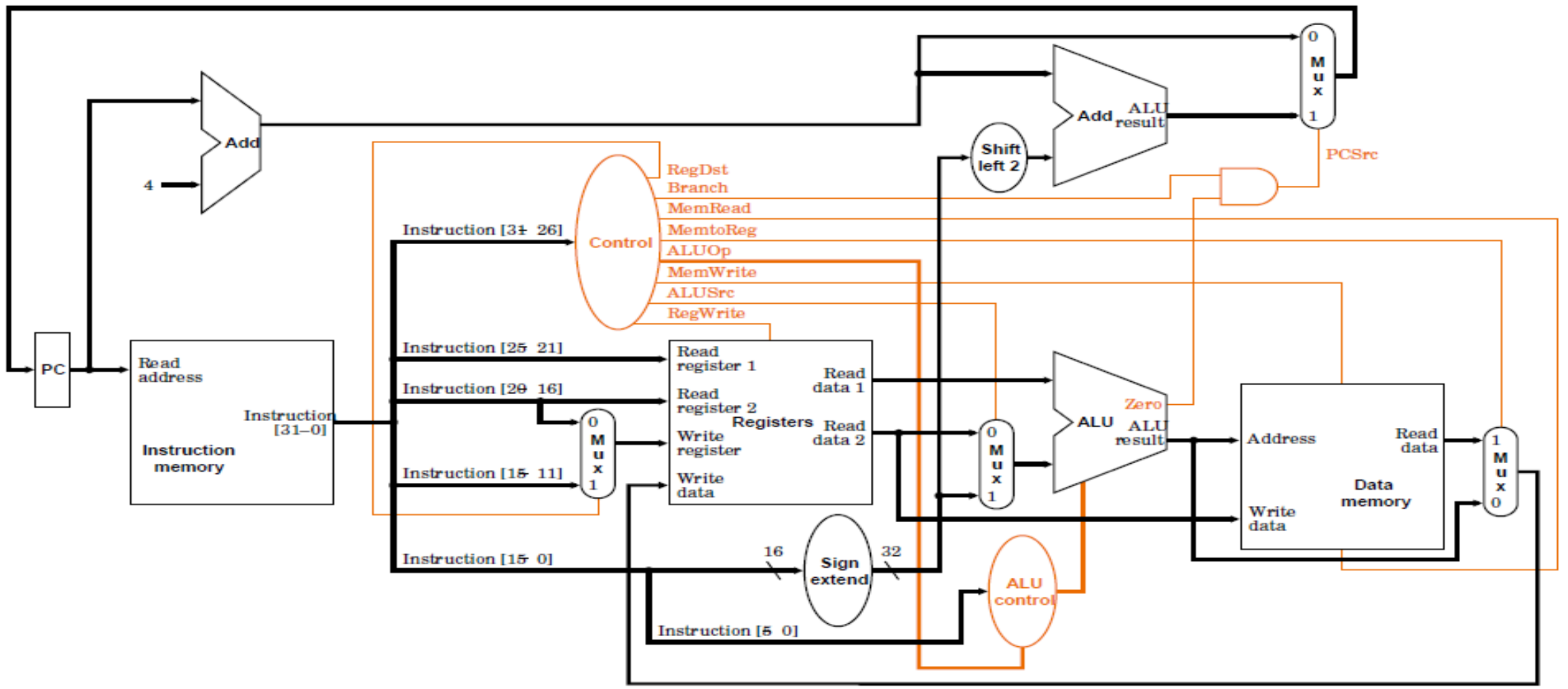
Option 1 is used for which instructions?

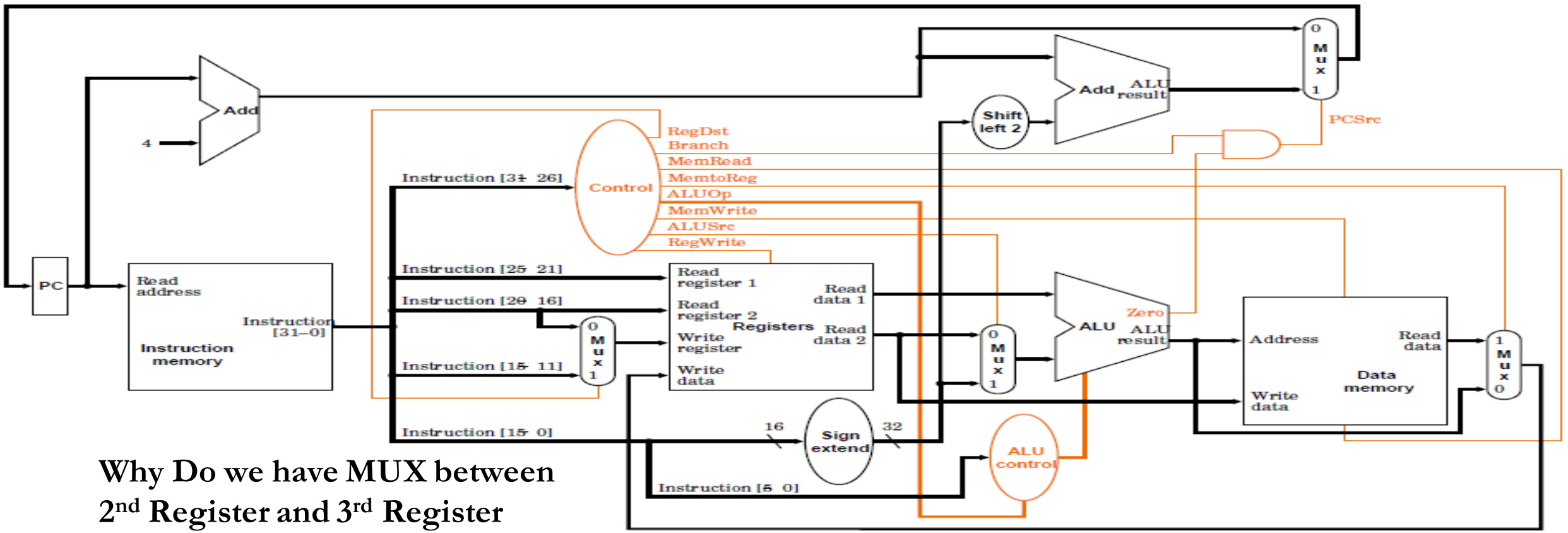
Rformat

Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`



BRANCHING


Add
4
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [31–26]
Control
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
0 M u x 1
16
Sign extend
32
Shift left 2
Add
ALU result
PCSrc
Zero
ALU
ALU result
ALU control
Address
Read data
Data memory
Write data
1 M u x 0



Why Do we have MUX between 2nd Register and 3rd Register (RegDst Mux)

A) Write Register for SW and Beq instruction

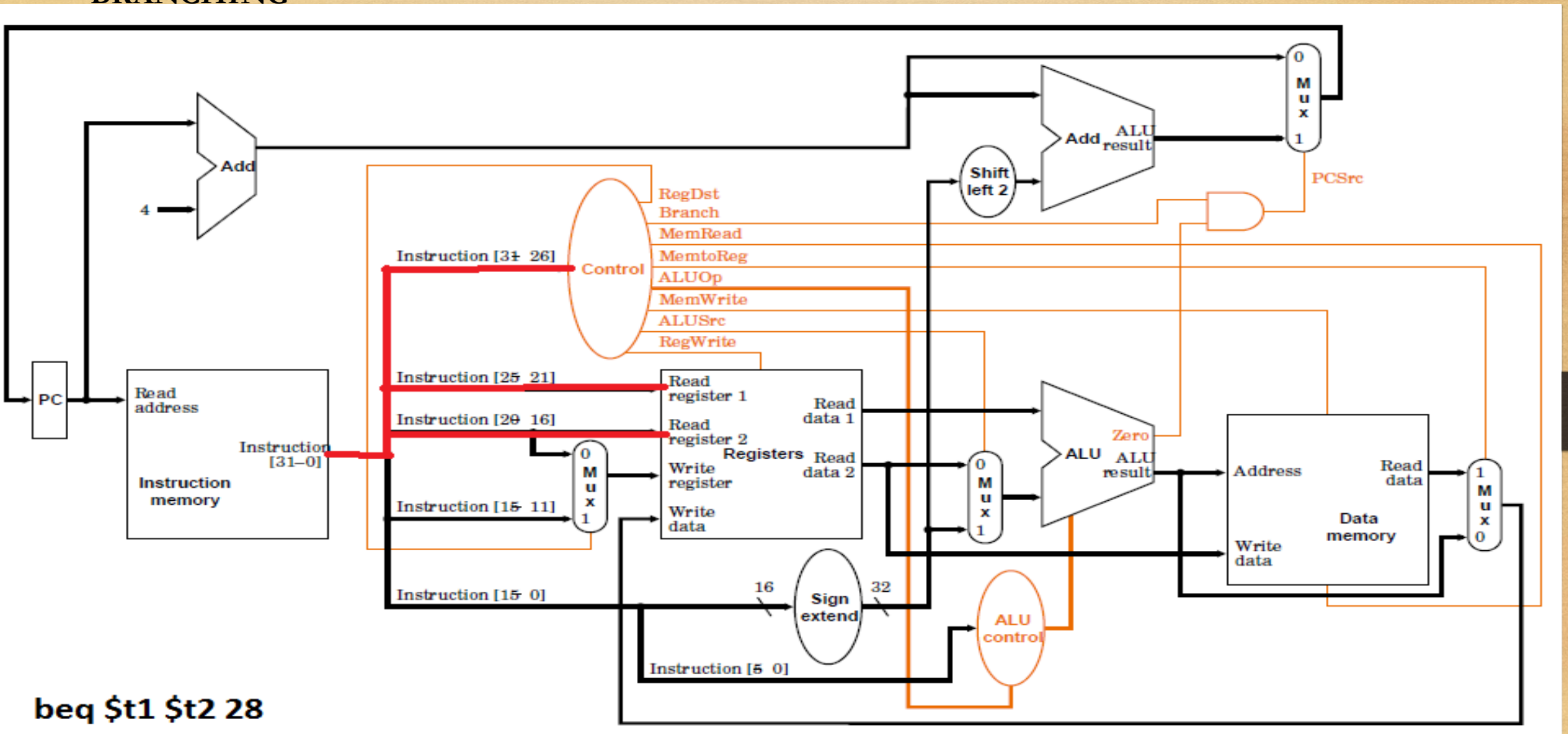
B) Write Register for LW and SW

C) Write Register for Data Memory

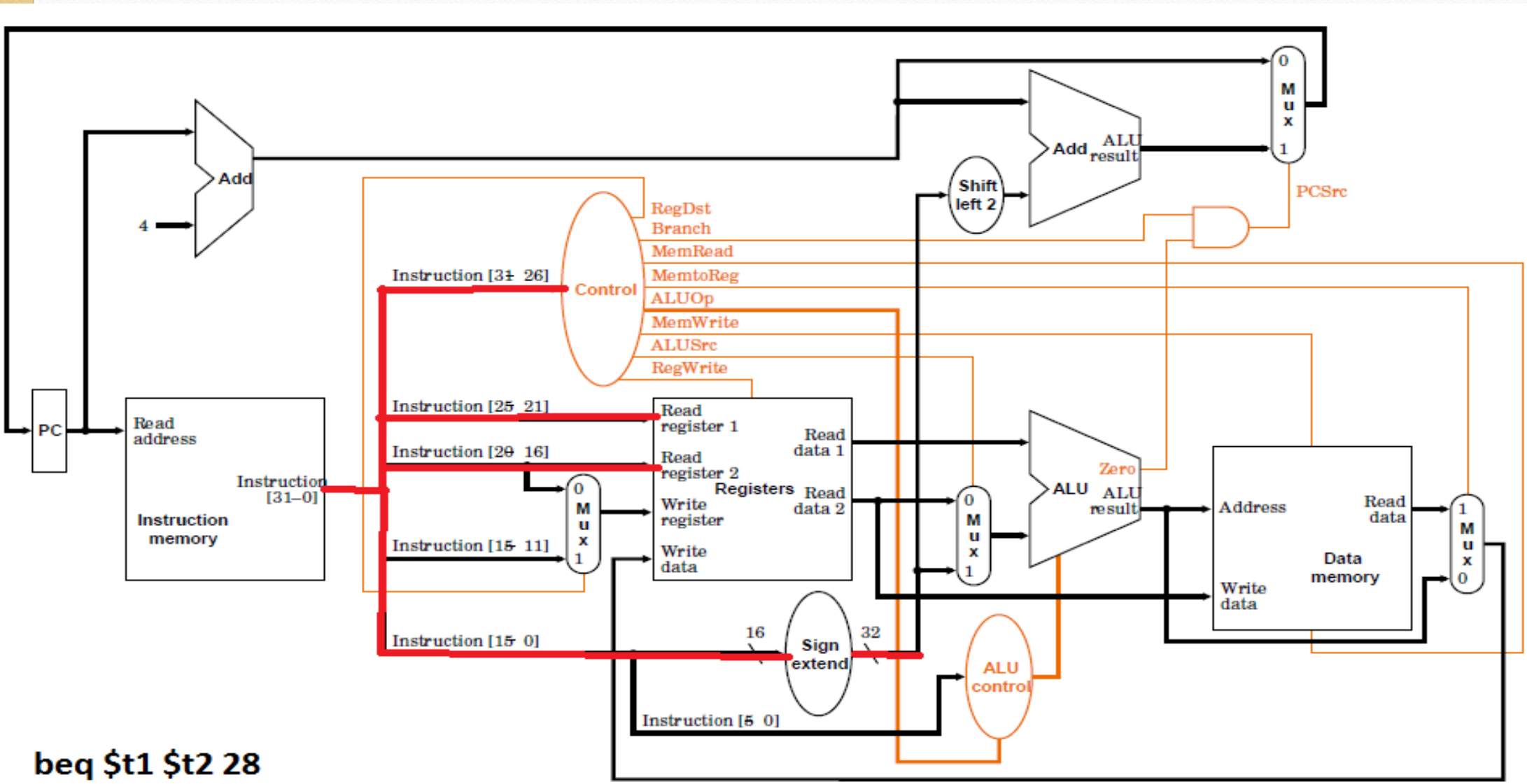
D) Write Register between an Rformat Instruction and LW

E) Write Register between Add instruction and Sub Instruction

BRANCHING

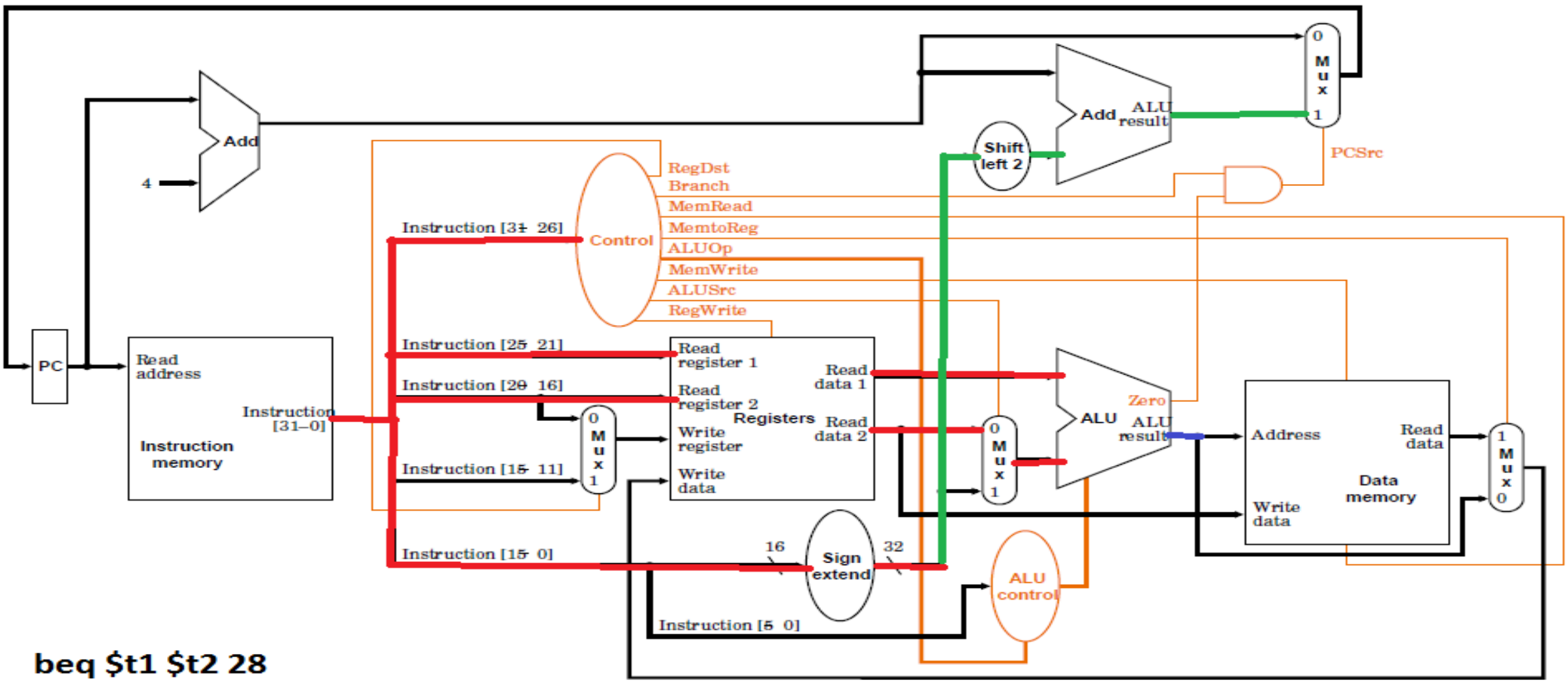

Add
4
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [31–26]
Control
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
0 M u x 1
16
Sign extend
32
Shift left 2
Add
ALU result
PCSrc
ALU
Zero
ALU control
Address
Read data
Data memory
Write data
beq $t1 $t2 28

BRANCHING



beq $t1 $t2 28

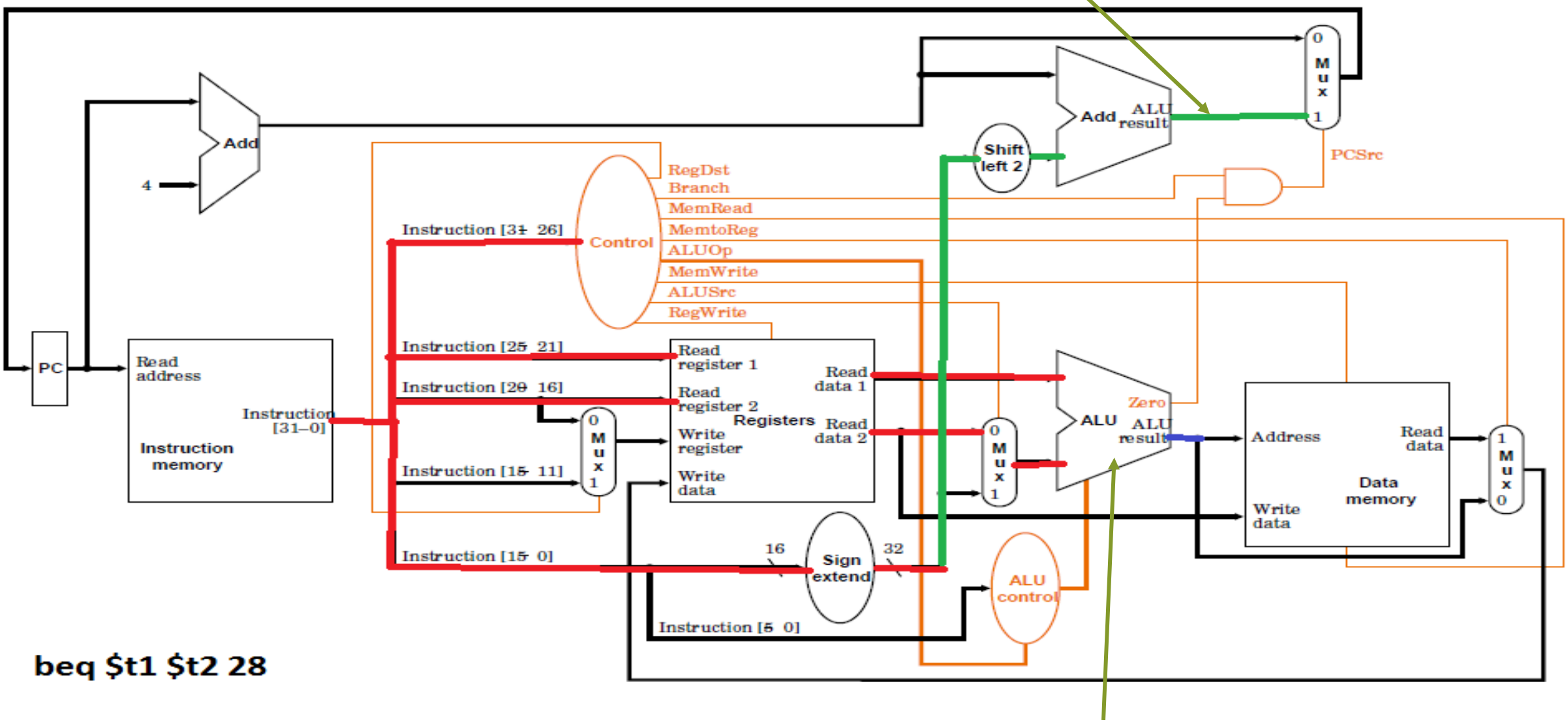
BRANCHING



beq $t1 $t2 28

BRANCHING

Computation of Branch Target address

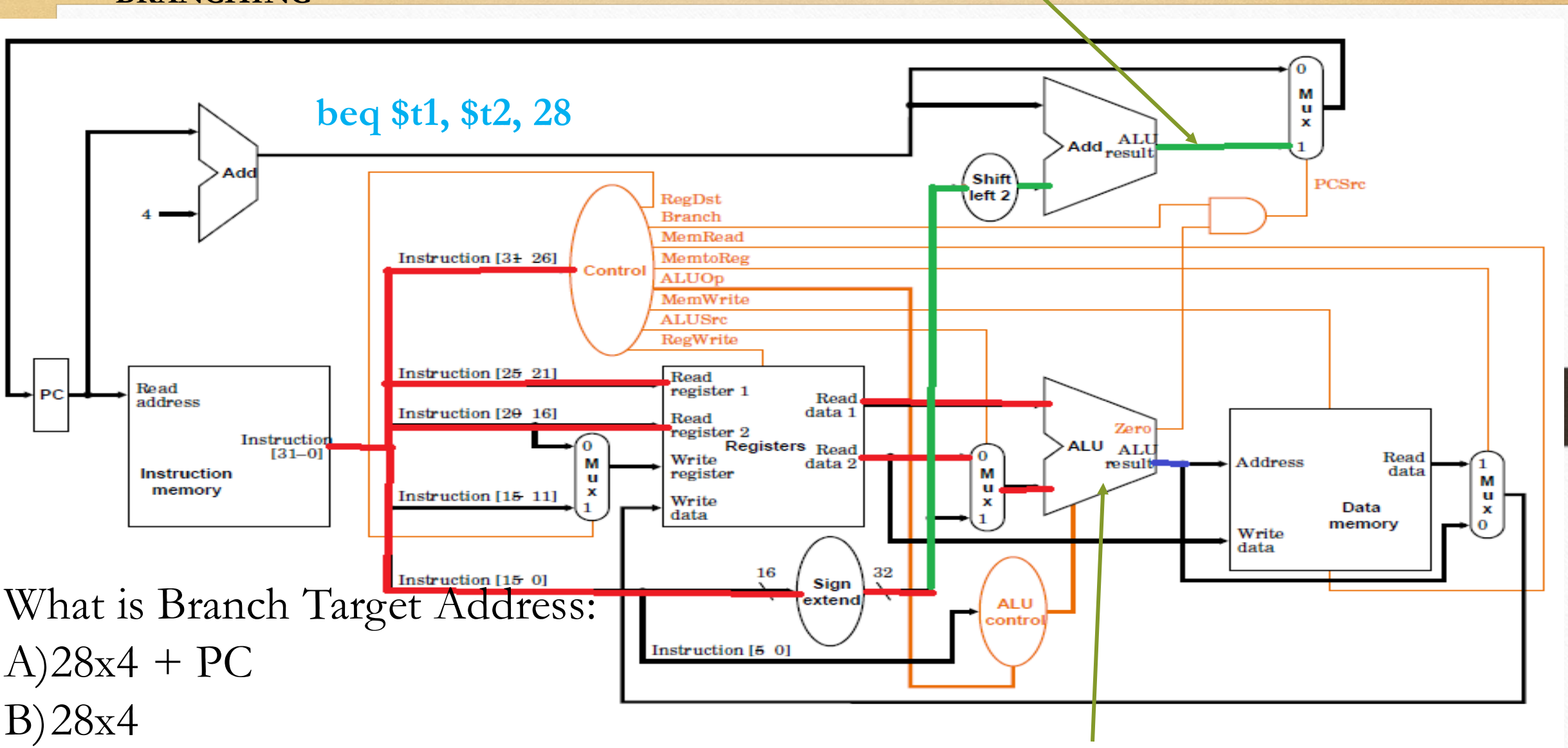


beq $t1 $t2 28

Branch instruction ALU will subtract Two Source Registers

BRANCHING

Computation of Branch Target address



What is Branch Target Address:

A)28x4 + PC

B)28x4

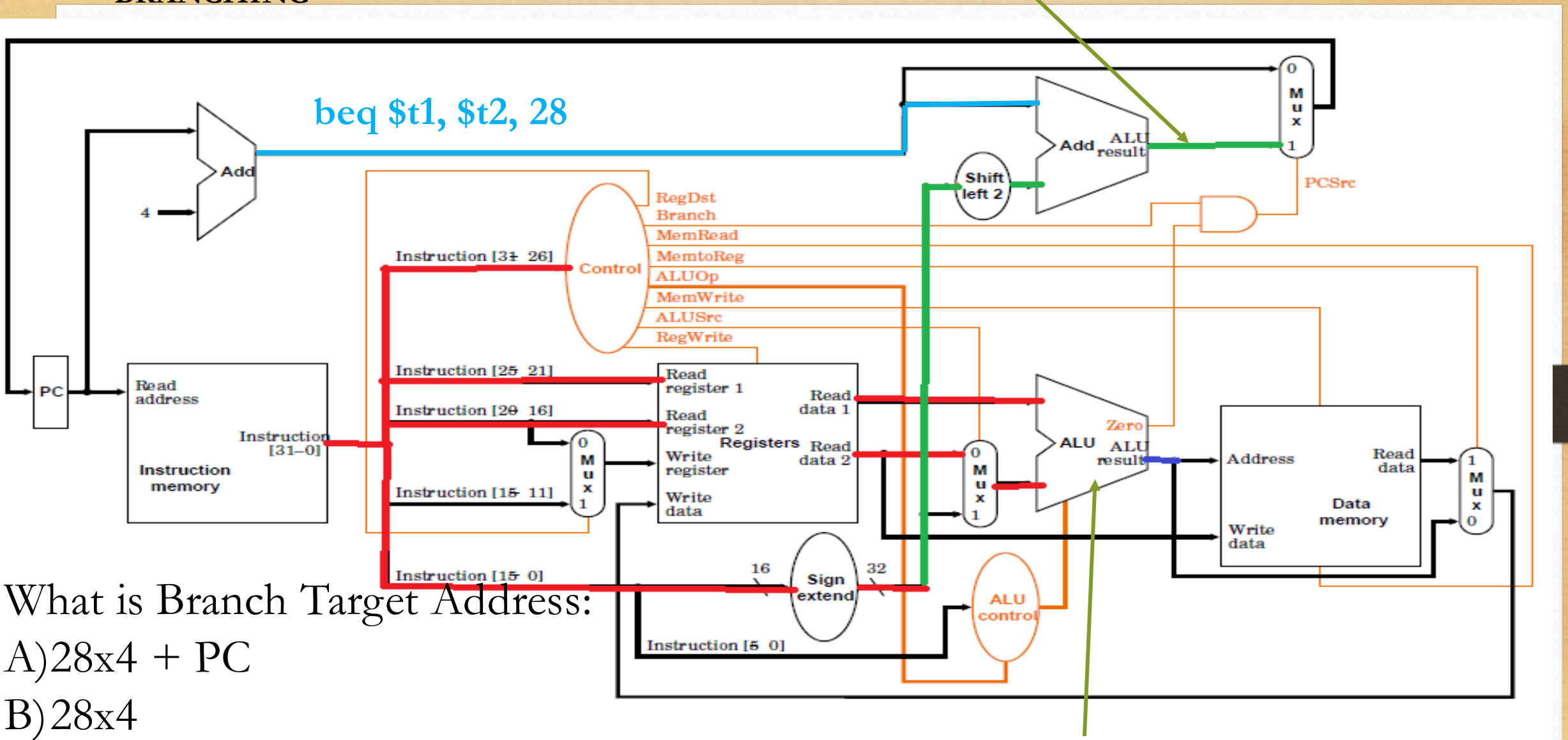
C)28x32 +PC+4

D)28x4 + (PC+4)

Branch instruction ALU will subtract Two Source Registers

BRANCHING

Computation of Branch Target address



What is Branch Target Address:

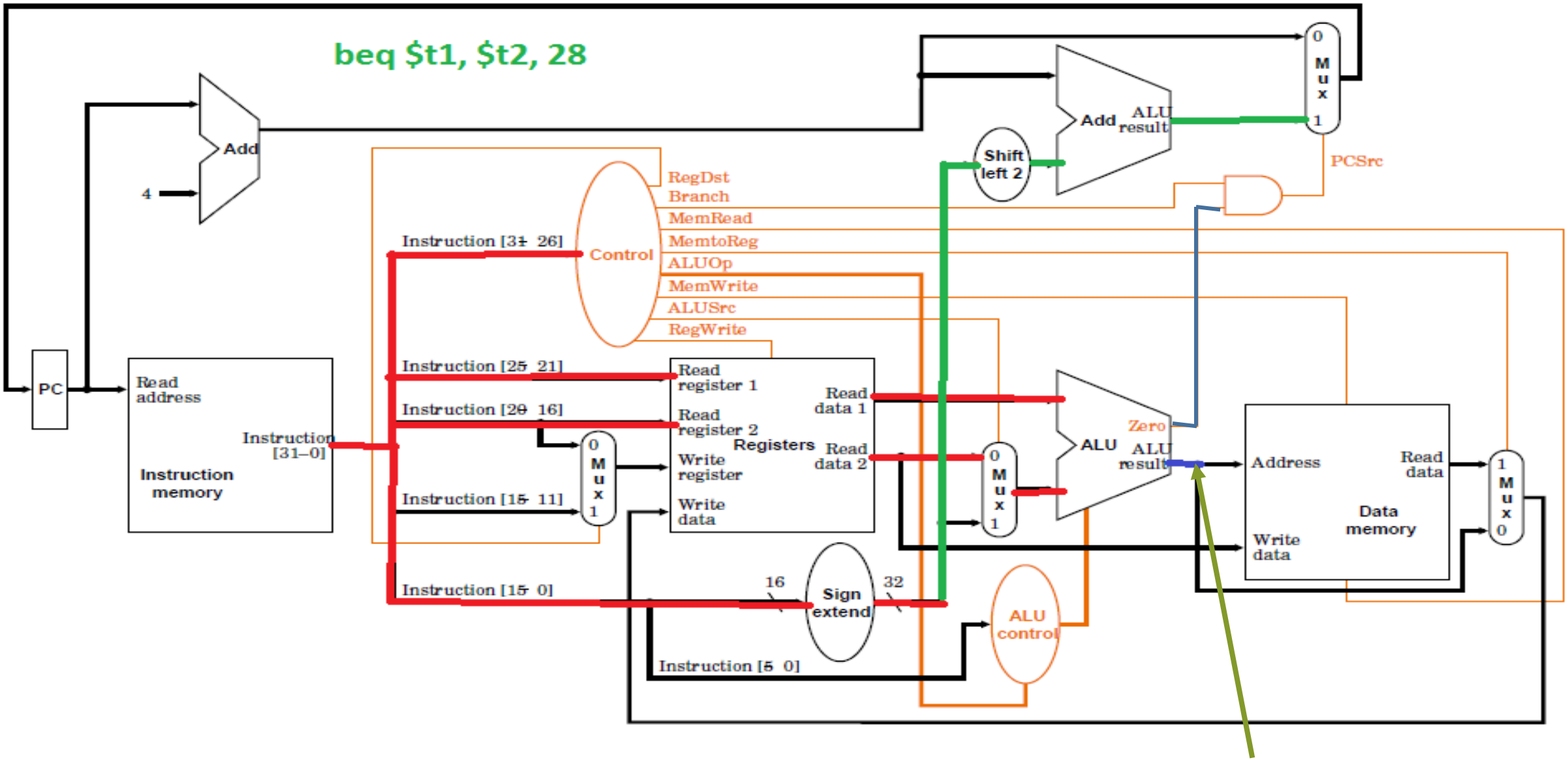
A) 28x4 + PC

B) 28x4

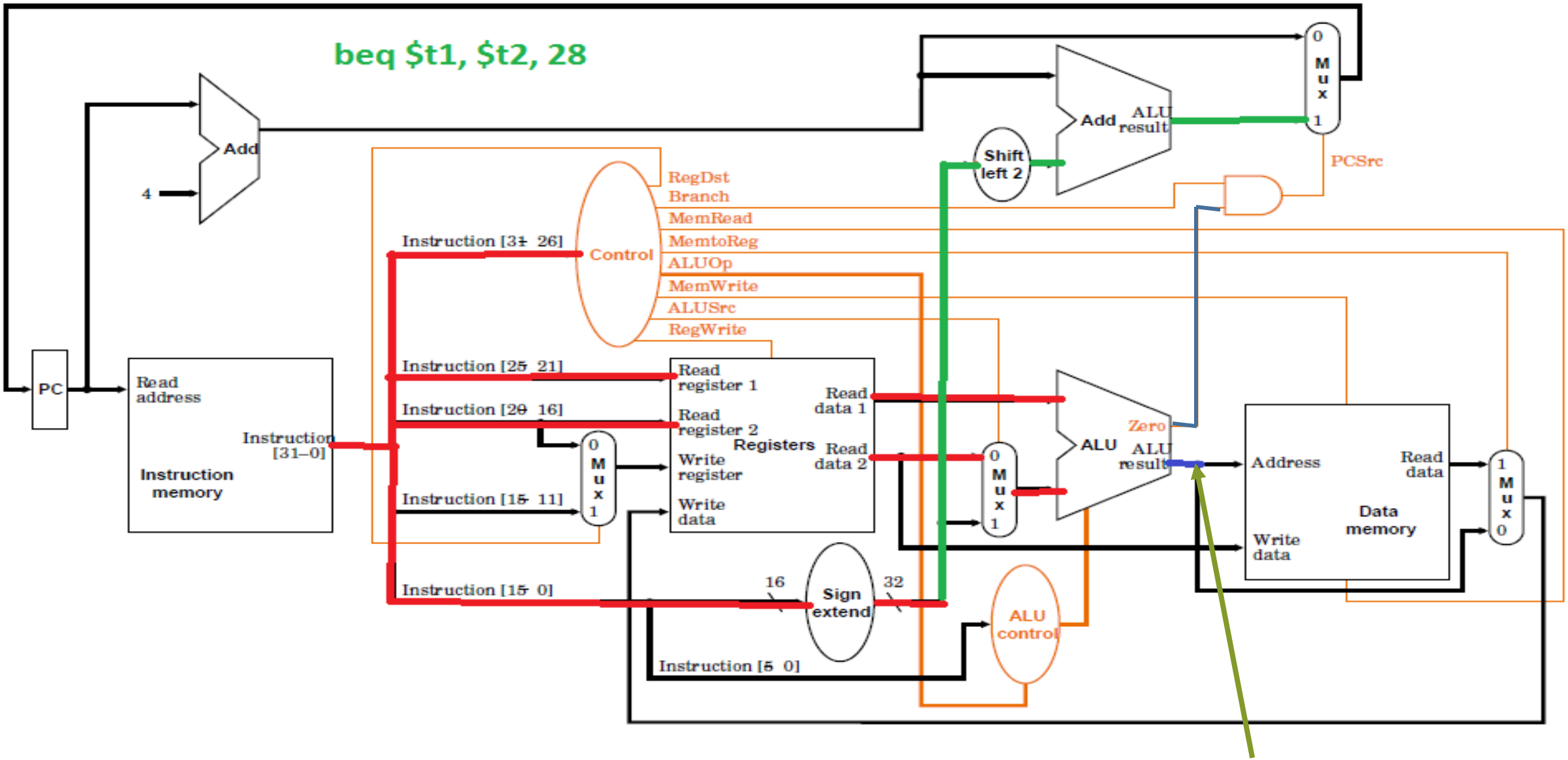
C) 28x32 +PC+4

D) 28x4 + (PC+4)

Branch instruction ALU will subtract Two Source Registers



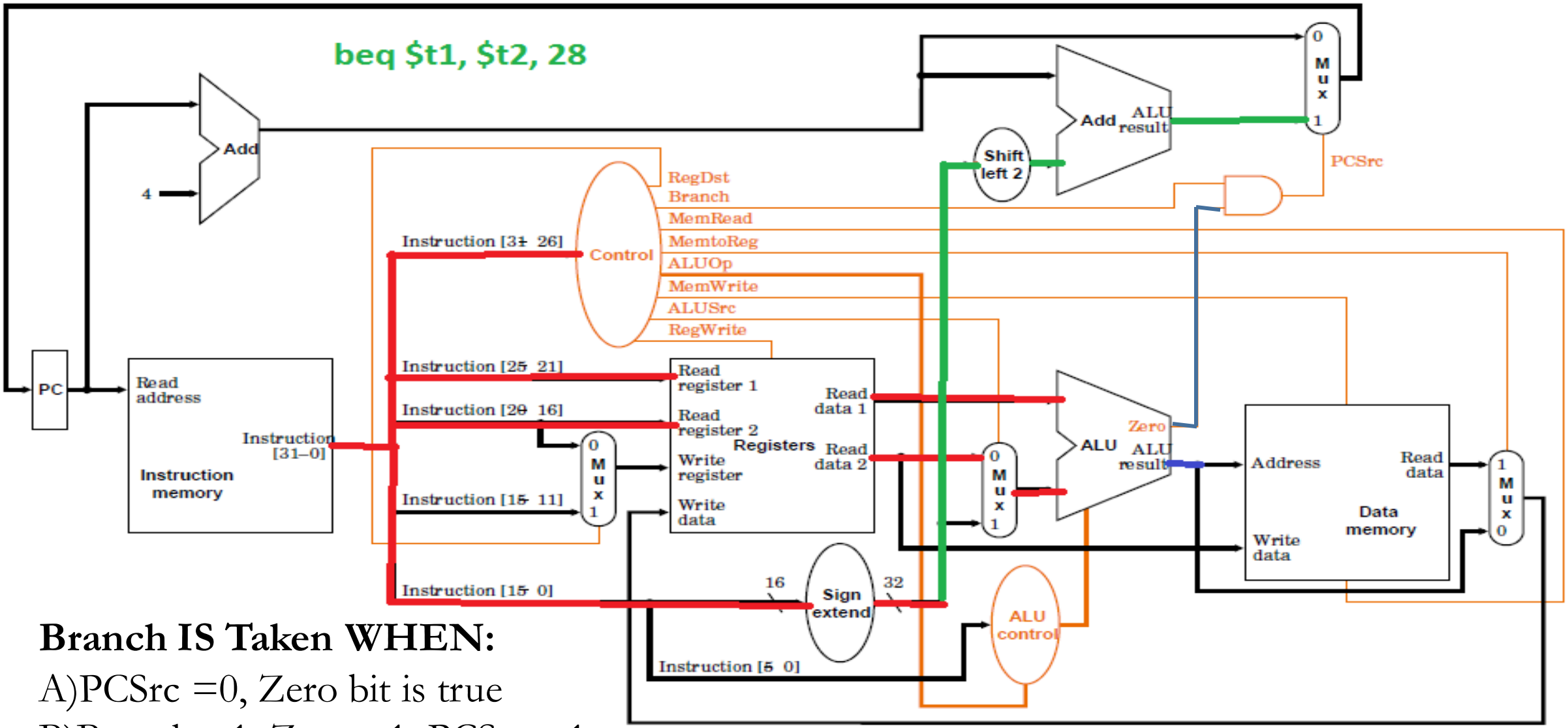
What is the value here?



What is the value here?

If I know $t1==$t2

Result is zero



Branch IS Taken WHEN:

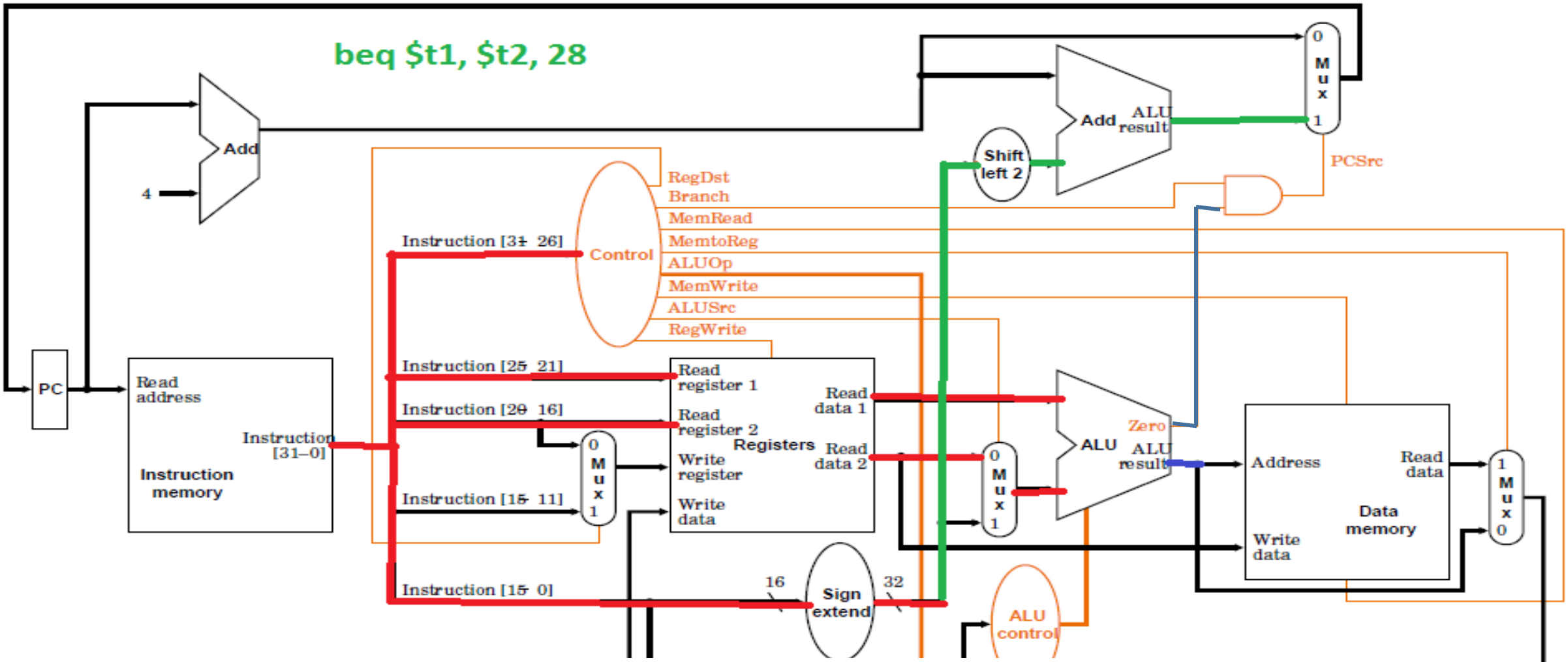
A)PCSrc =0, Zero bit is true

B)Branch =1, Zero =1, PCSrc =1

C) Branch =0, Zero=1, PCSrc =1

D) Need more hardware to decide

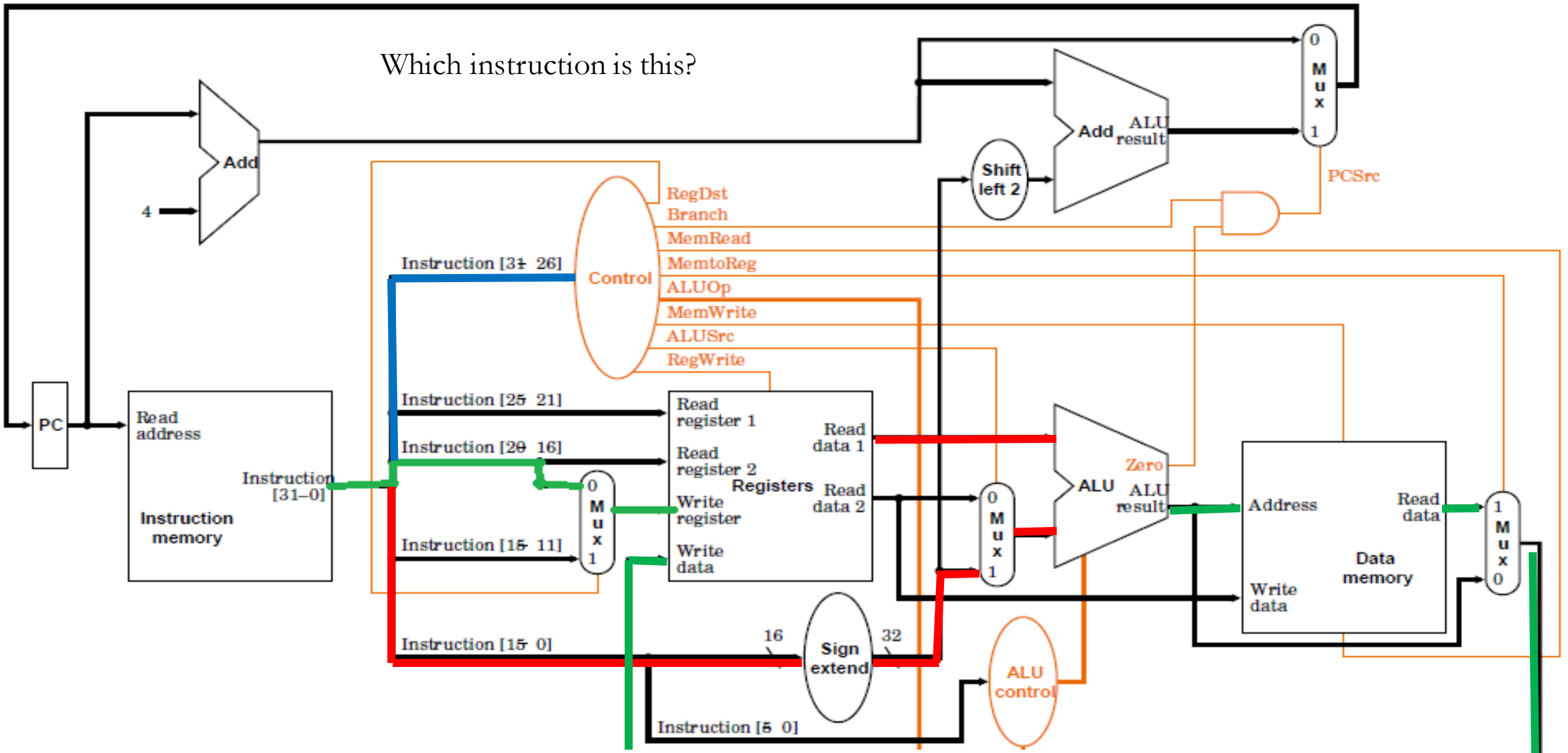
E) ALUSrc =0, Branch =1, Zero =0



Type	Reg Dst	ALU Src	Mem ToReg	Reg Write	Mem Read	Mem Write	Branch	ALU op1	ALU op0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

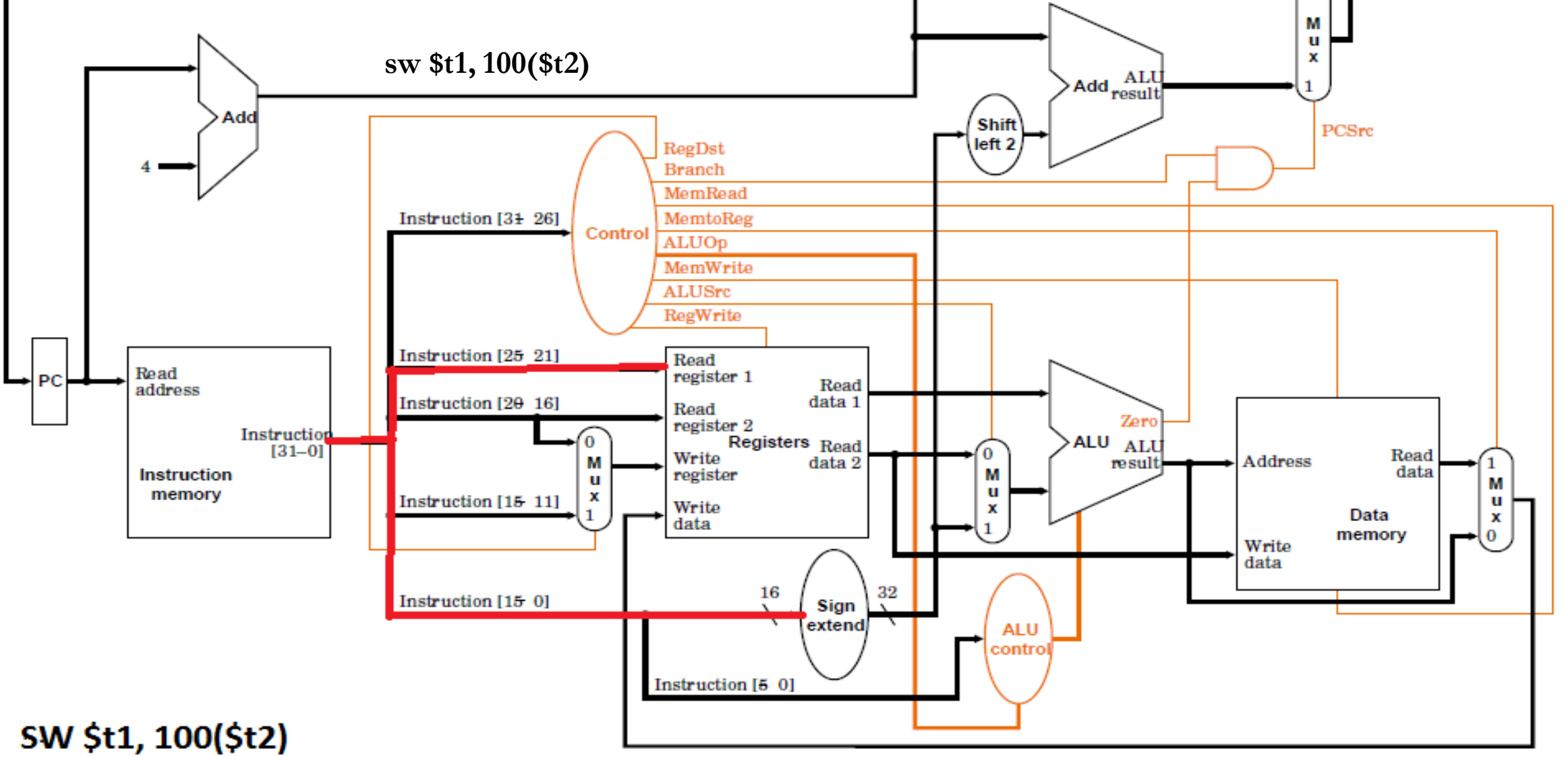
Tells you exactly what the control lines Need to be for each instruction

Which instruction is this?



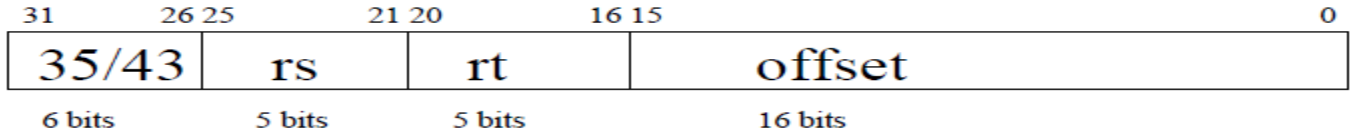
Type	Reg Dst	ALU Src	Mem ToReg	Reg Write	Mem Read	Mem Write	Branch	ALU op1	ALU op0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
	X	1	X	0	0	1	0	0	0

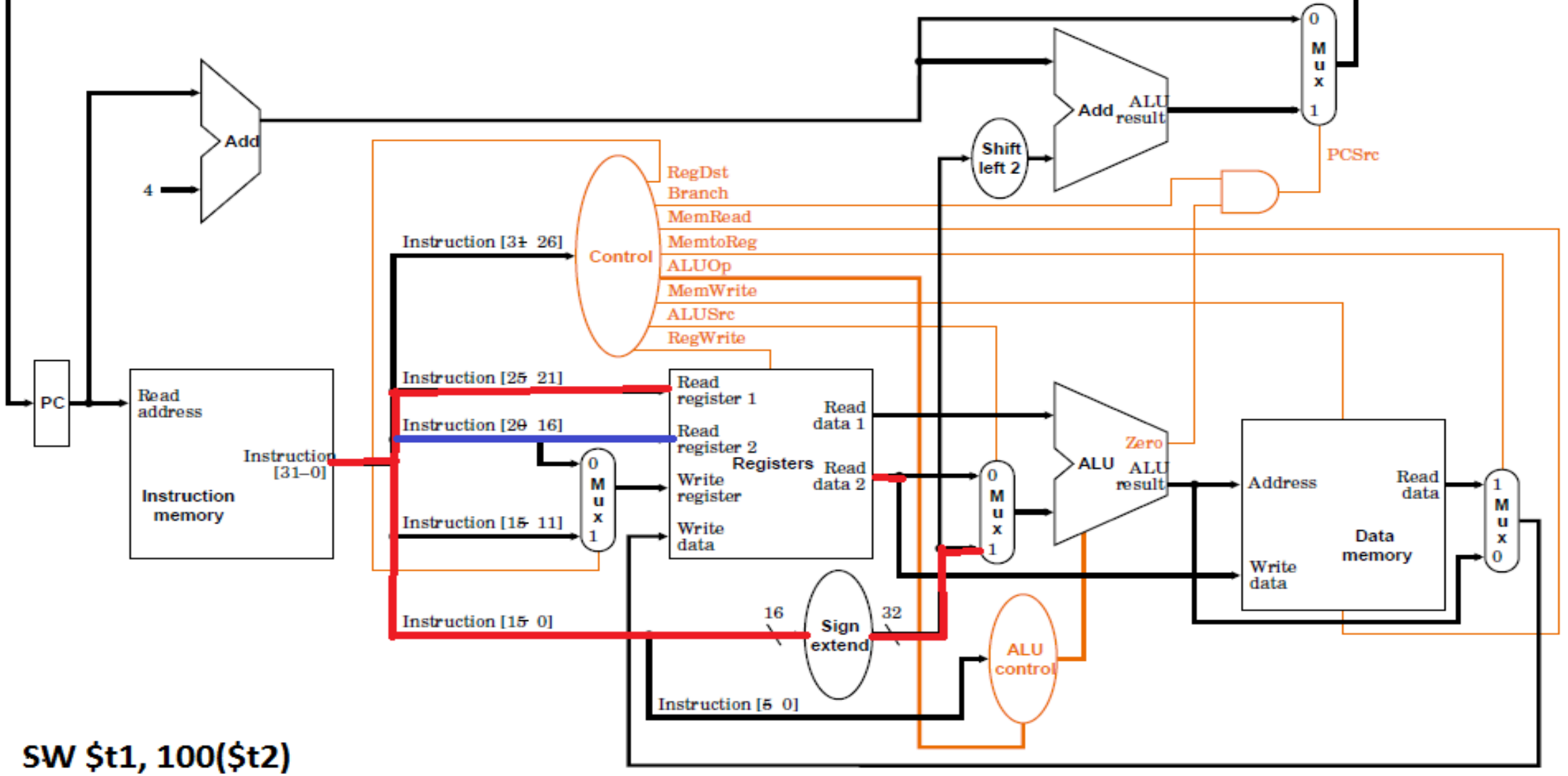
Control Units : Course Notes



SW $t1, 100($t2)

Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`

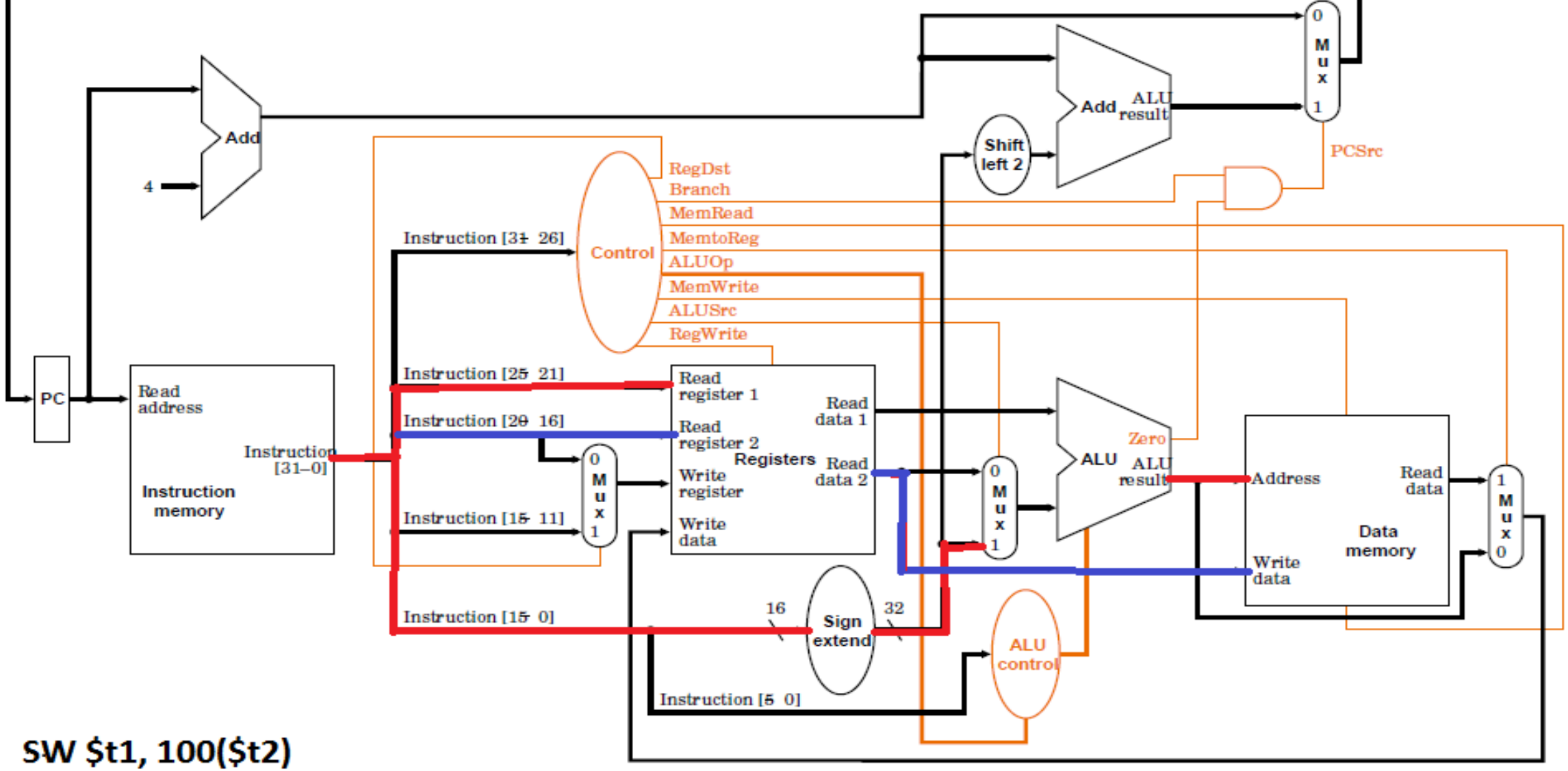




SW $t1, 100($t2)

Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`

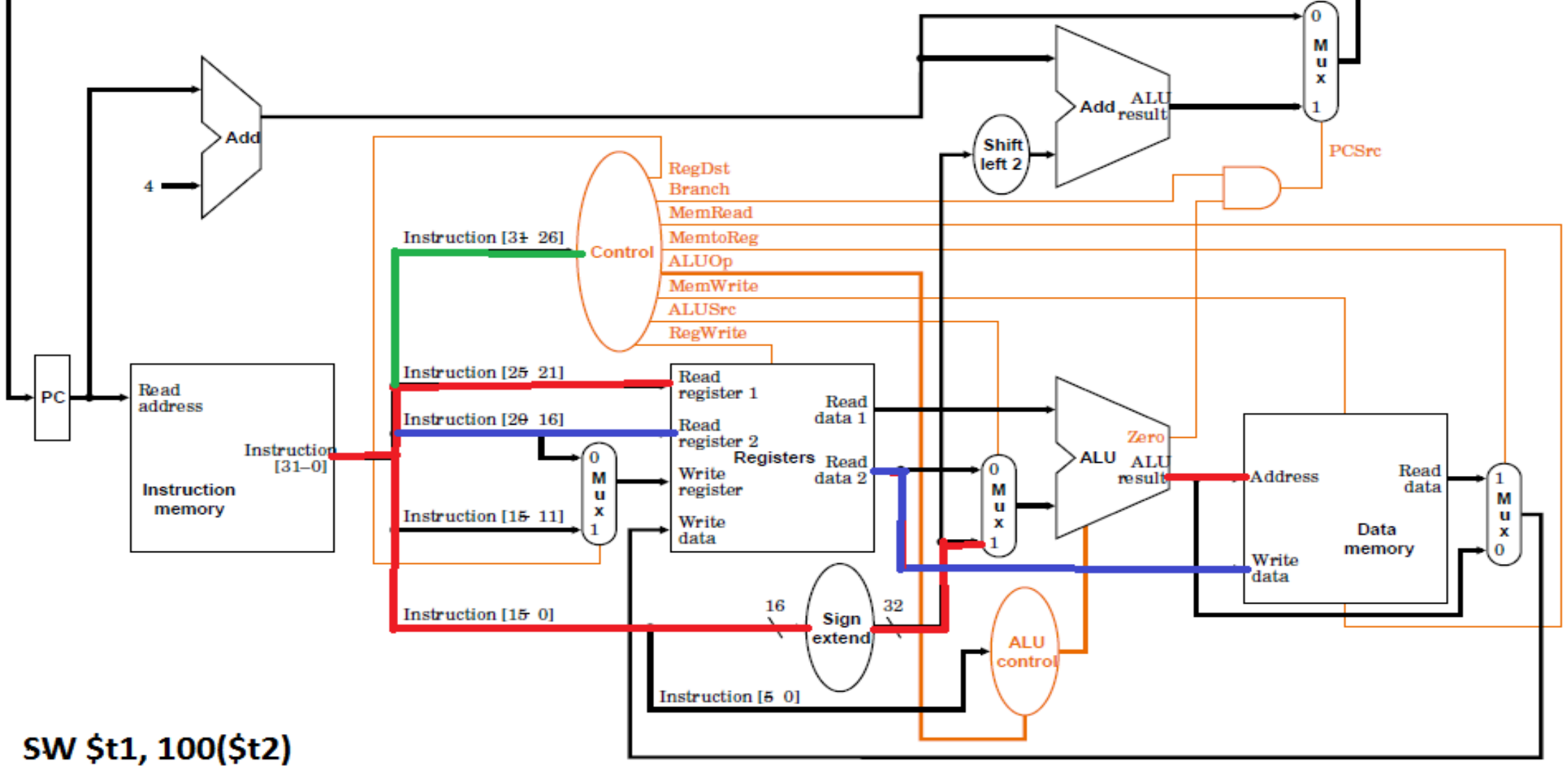
31–26	25–21	20–16	15–0
35/43	rs	rt	offset
6 bits	5 bits	5 bits	16 bits



SW $t1, 100($t2)

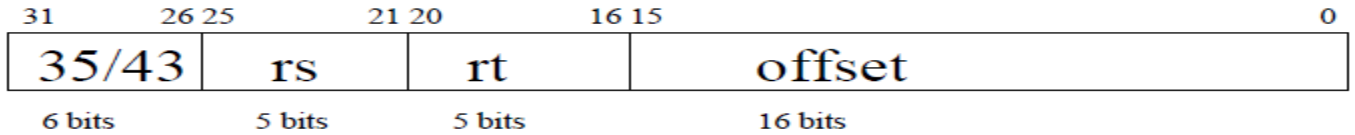
Load/store: `lw $t1, 100($t2)` ⇒ `lw rt, 100(rs)`

31–26	25–21	20–16	15–0
35/43	rs	rt	offset
6 bits	5 bits	5 bits	16 bits

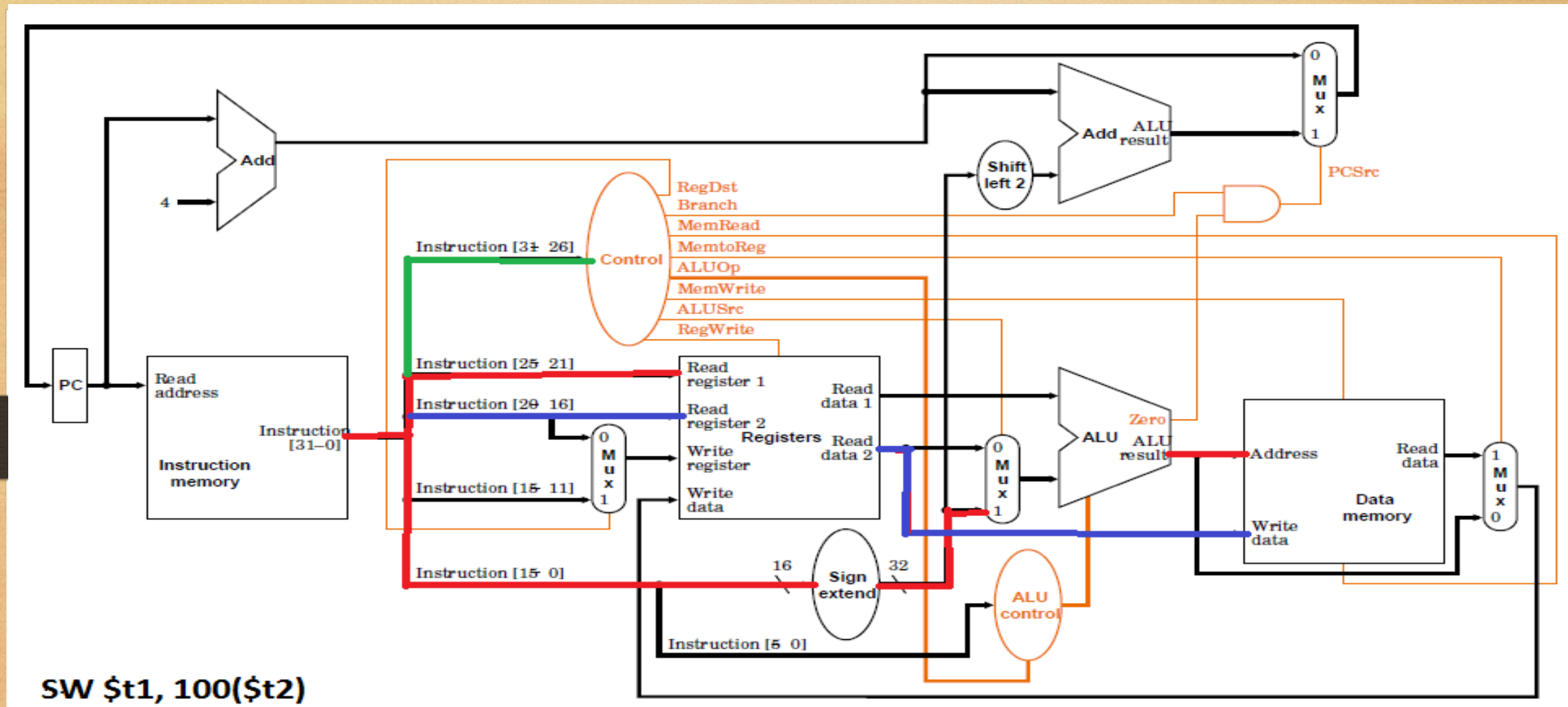


SW $t1, 100($t2)

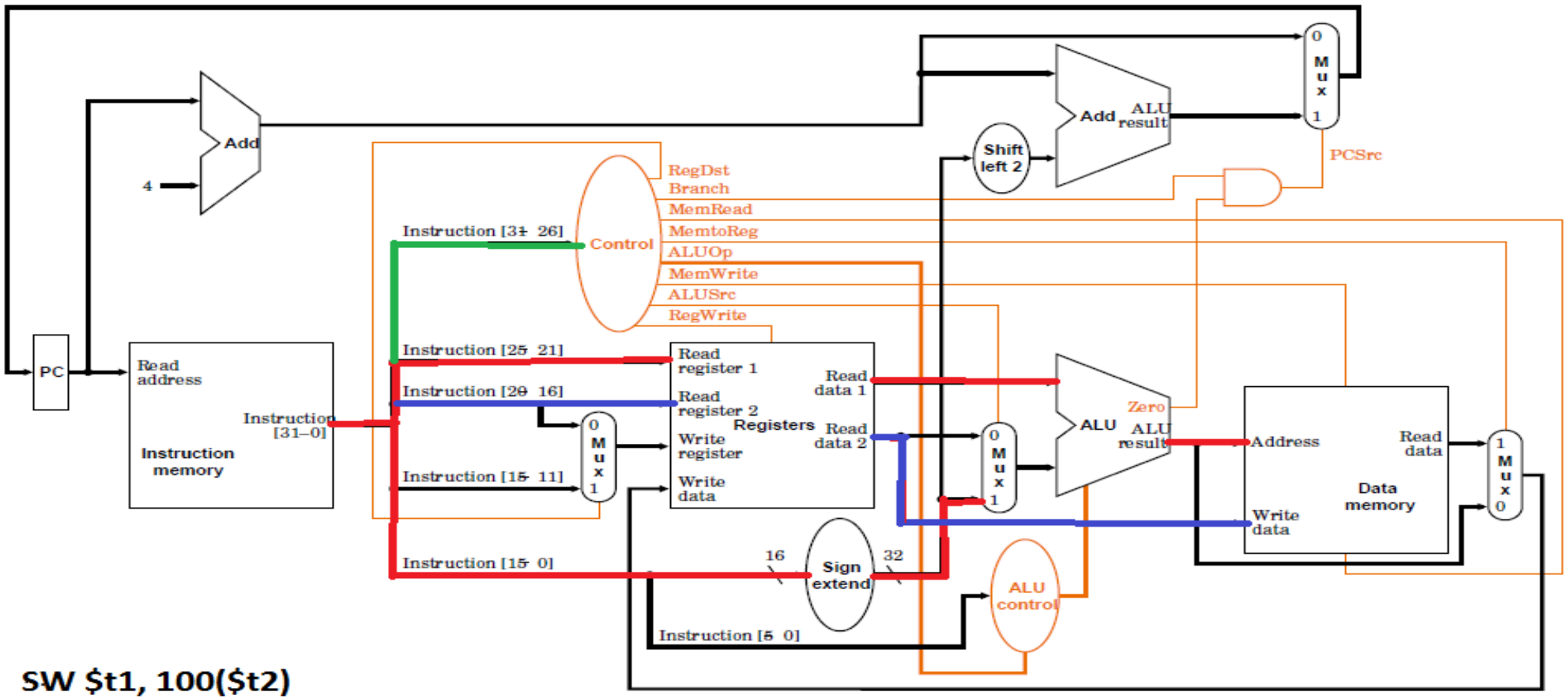
Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`



sw $t1, 100($t2)



SW $t1, 100($t2)



SW $t1, 100($t2)

WHICH ONE OF THE FOLLOWING MUST BE TRUE TO COMPLETE SW instruction:

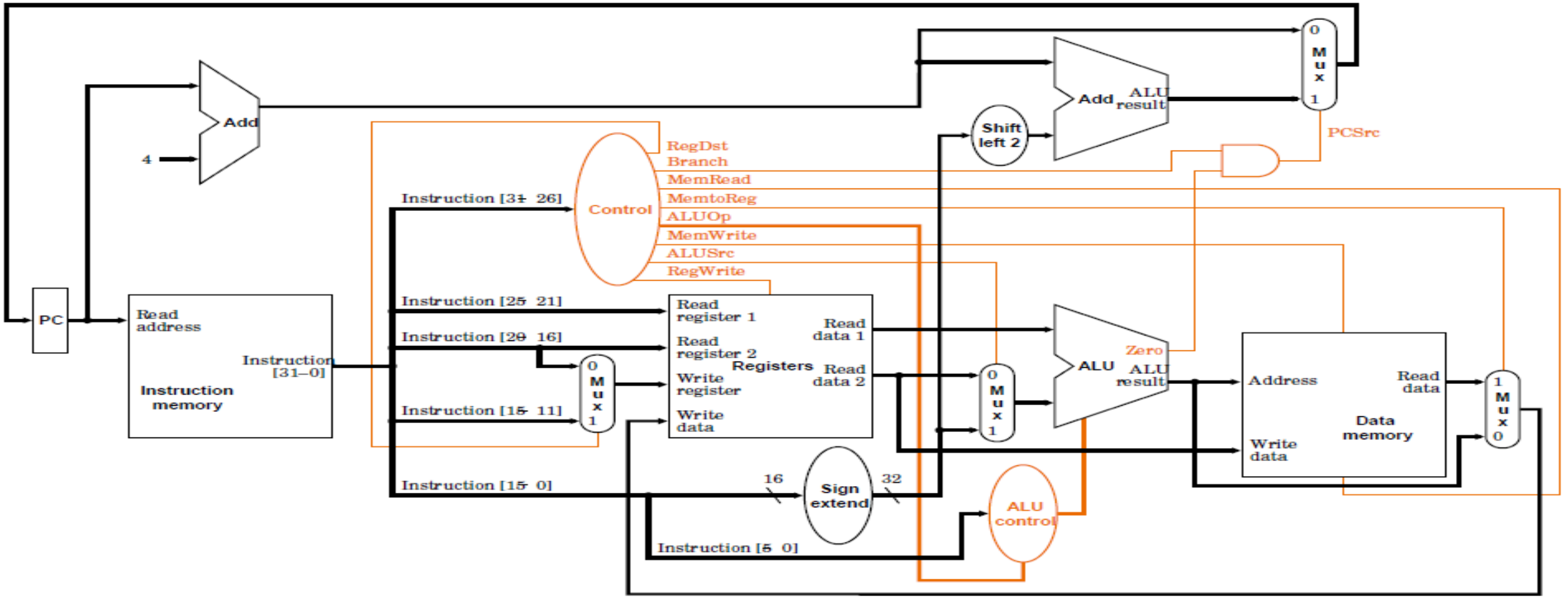
A) MemWrite = OFF (Low) , MemtoReg = 1

B) ALUSrc = 1 RegDst =1

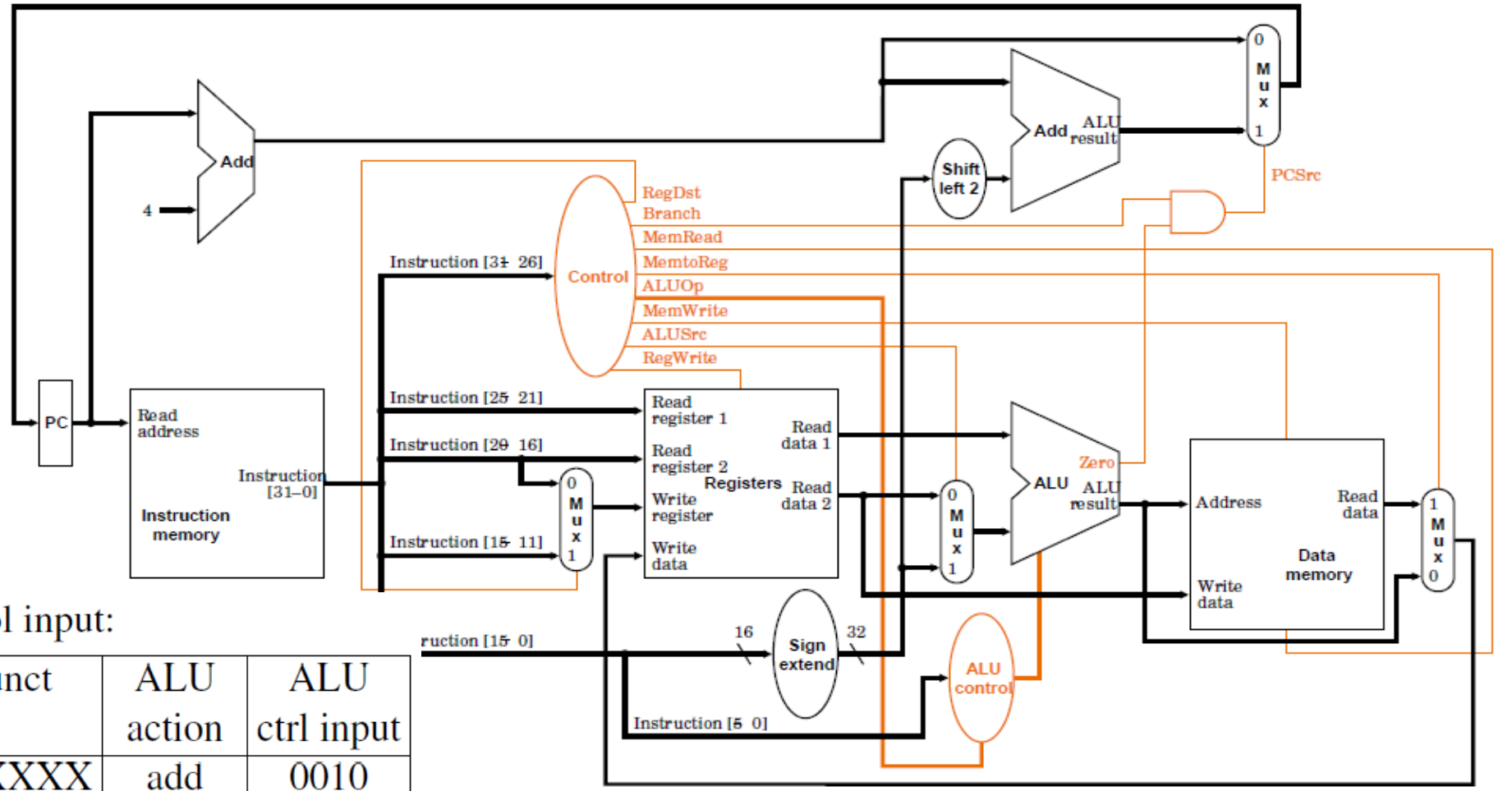
C) ALUSrc =1 , MemWrite =1

D) MemtoReg =1 , MemWrite =1

E) MemtoReg = X, ALUSrc =0



Type	Reg Dst	ALU Src	Mem ToReg	Reg Write	Mem Read	Mem Write	Branch	ALU op1	ALU op0
R-format	1	0	0	1	0	0	0	1	0
`lw`	0	1	1	1	1	0	0	0	0
`sw`	X	1	X	0	0	1	0	0	0
`beq`	X	0	X	0	0	0	1	0	1



Mapping of operation to ALU control input:

Opcode	Operation	ALUop	Funct	ALU action	ALU ctrl input
35	lw	00	XXXXXX	add	0010
43	sw	00	XXXXXX	add	0010
4	beq	01	XXXXXX	subtract	0110
0	add	10	100000	add	0010
0	sub	10	100010	subtract	0110
0	and	10	100100	AND	0000
0	or	10	100101	OR	0001
0	slt	10	101010	slt	0111

lapping of operation to ALU control input:

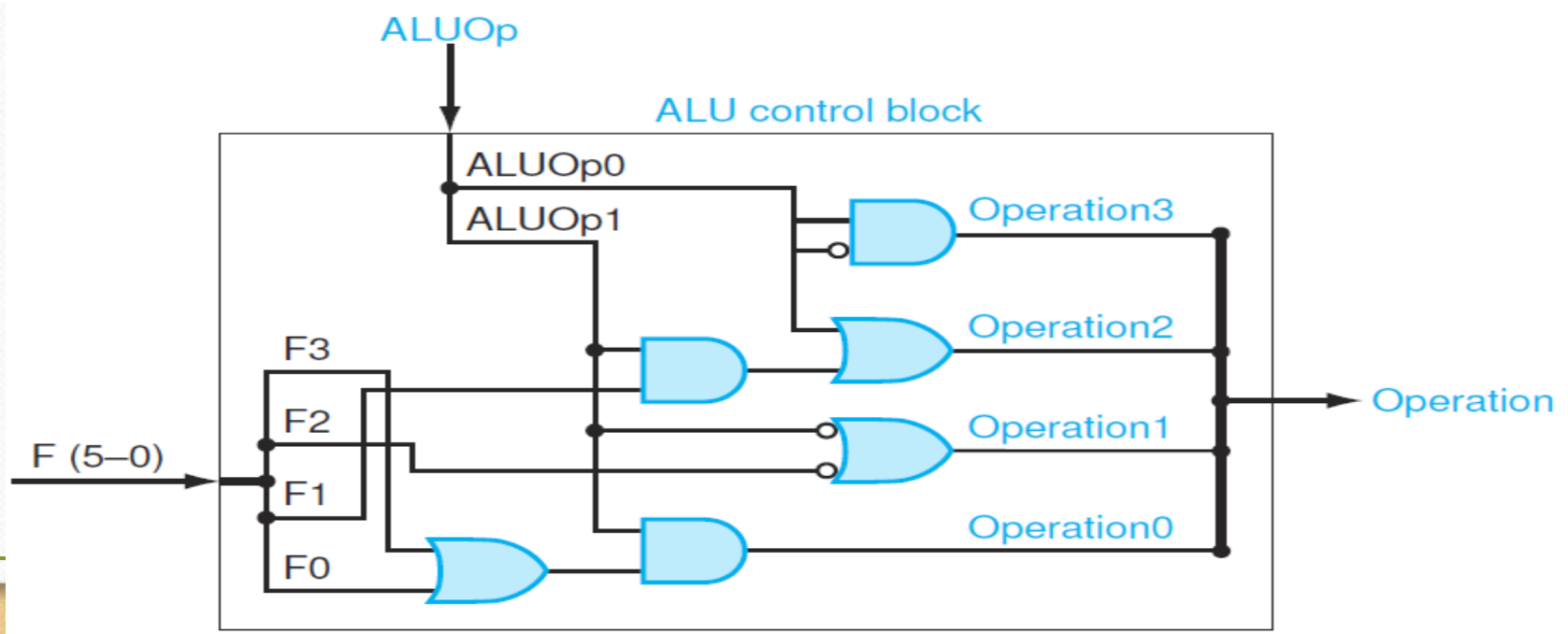
Opcode	Operation	ALUop	Funct	ALU action	ALU ctrl input
35	lw	00	XXXXXX	add	0010
43	sw	00	XXXXXX	add	0010
4	beq	01	XXXXXX	subtract	0110
0	add	10	100000	add	0010
0	sub	10	100010	subtract	0110
0	and	10	100100	AND	0000
0	or	10	100101	OR	0001
0	slt	10	101010	slt	0111

ALUop		Funct field						Operation
ALUop1	ALUop0	F5	F4	F3	F2	F1	F0	3210
0	0	X	X	X	X	X	X	0010
X(0)	1	X	X	X	X	X	X	0110
1	X(0)	X	X	0	0	0	0	0010
1	X(0)	X	X	0	0	1	0	0110
1	X(0)	X	X	0	1	0	0	0000
1	X(0)	X	X	0	1	0	1	0001
1	X(0)	X	X	1	0	1	0	0111

lapping of operation to ALU control input:

Opcode	Operation	ALUop	Funct	ALU action	ALU ctrl input
35	lw	00	XXXXXX	add	0010
43	sw	00	XXXXXX	add	0010
4	beq	01	XXXXXX	subtract	0110
0	add	10	100000	add	0010
0	sub	10	100010	subtract	0110
0	and	10	100100	AND	0000
0	or	10	100101	OR	0001
0	slt	10	101010	slt	0111

ALUop		Funct field						Operation
ALUop1	ALUop0	F5	F4	F3	F2	F1	F0	3210
0	0	X	X	X	X	X	X	0010
X(0)	1	X	X	X	X	X	X	0110
1	X(0)	X	X	0	0	0	0	0010
1	X(0)	X	X	0	0	1	0	0110
1	X(0)	X	X	0	1	0	0	0000
1	X(0)	X	X	0	1	0	1	0001
1	X(0)	X	X	1	0	1	0	0111



Type	Reg Dst	ALU Src	Mem ToReg	Reg Write	Mem Read	Mem Write	Branch	ALU op1	ALU op0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Type	Dec Opcode	Binary Opcode
R-format	0	000 000
lw	35	100 011
sw	43	101 011
beq	4	000 100