Ensamblado y desensamblado de LEGv8 + Implementación de la ISA

Arquitectura de Computadoras 2023

Arithmetic Operations

- Three operands: two sources and one destination

  ADD a, b, c   // a gets b + c

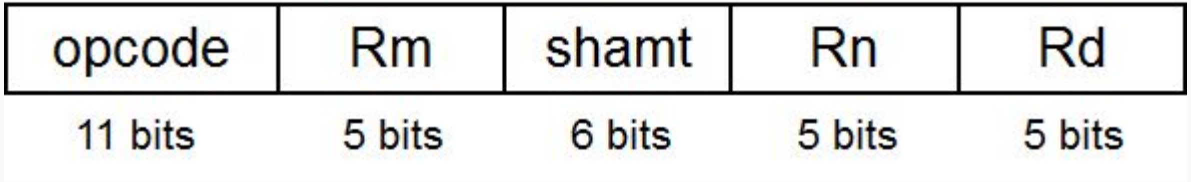
- C code:

```
f = (g + h) - (i + j); // f-> X19 g->X20, h->X21
                       // i->X22, j->X23
```

- Compiled LEGv8 code:

```
ADD   X0,   X20,   X21          //   X0   =   g   +   h
ADD   X1,   X22,   X23          //   X1   =   i   +   j
SUB X19, X0, X1    // f = X0 - X1
```

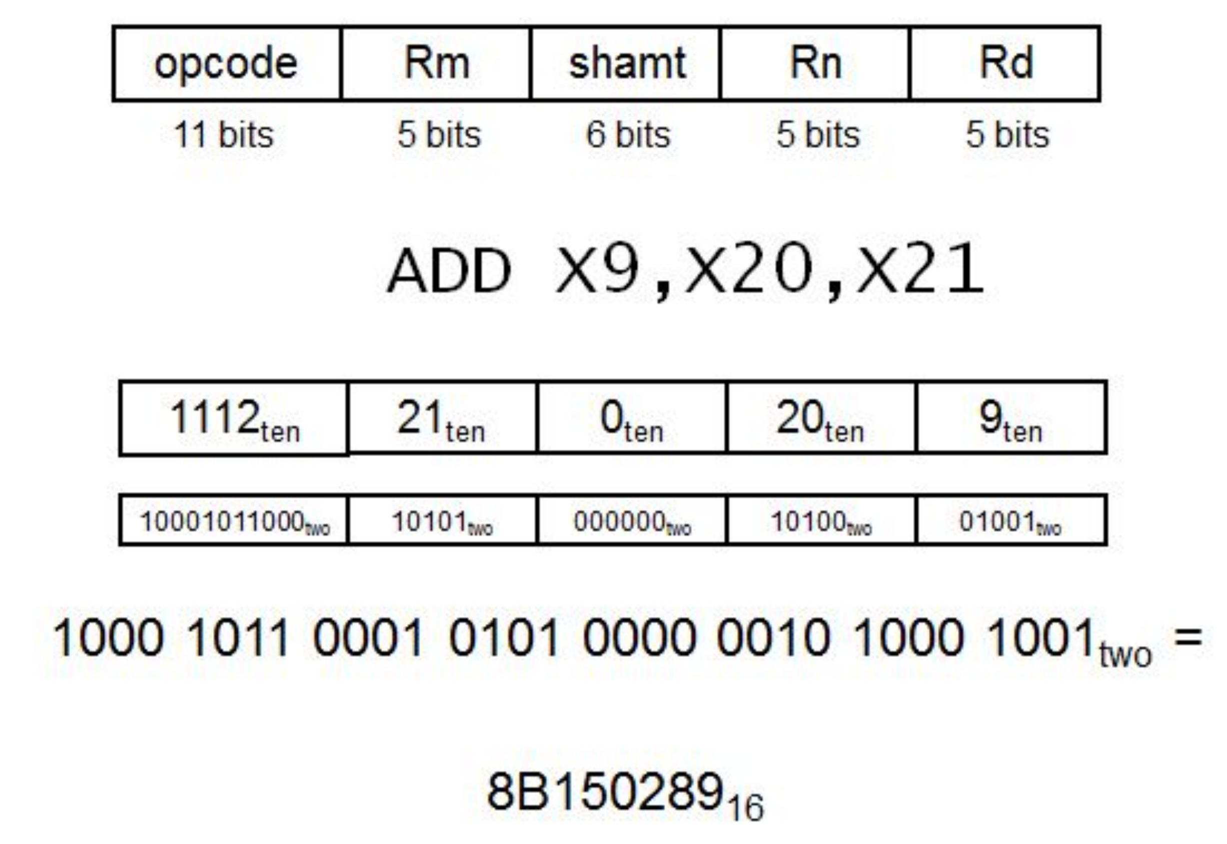
LEGv8 R-format Instructions



opcode	Rm	shamt	Rn	Rd
11 bits	5 bits	6 bits	5 bits	5 bits

- Instruction fields
  - opcode: operation code
  - Rm: the second register source operand
  - shamt: shift amount (00000 for now)
  - Rn: the first register source operand
  - Rd: the register destination

R-format Example



opcode	Rm	shamt	Rn	Rd
11 bits	5 bits	6 bits	5 bits	5 bits

ADD X9,X20,X21

1112_{ten}	21_{ten}	0_{ten}	20_{ten}	9_{ten}

10001011000_{two}	10101_{two}	000000_{two}	10100_{two}	01001_{two}

$1000\ 1011\ 0001\ 0101\ 0000\ 0010\ 1000\ 1001_{two}$ =

$8B150289_{16}$

Memory Operands

- Load values from memory into registers
- Store result from register to memory
- Memory is byte addressed

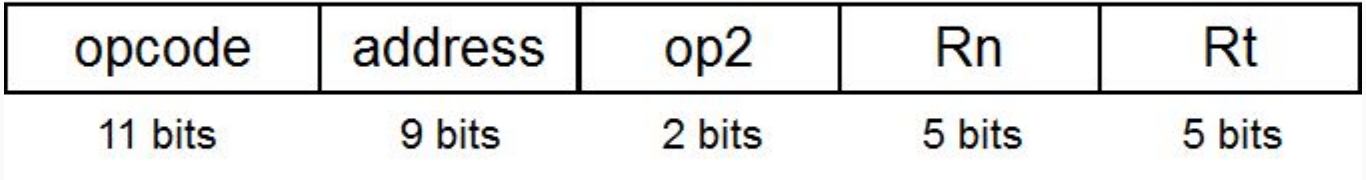
- C code: (elementos de A = DobleWord)

```
A[12] = h + A[8]; // h->X21, base address of A->X22
```

- Compiled LEGv8 code:

```
LDUR    X9, [X22,#64] // Index 8 requires offset of 64
ADD     X9, X21, X9
STUR    X9, [X22,#96]
```

LEGv8 D-format Instructions



opcode	address	op2	Rn	Rt
11 bits	9 bits	2 bits	5 bits	5 bits

- Load/store instructions
  - Rn: base register
  - address: constant offset from contents of base register (-256 to 255)
  - Rt: destination (load) or source (store) register number
  - op2 = “00”

Branch Operations

Branch to a labeled instruction if a **condition** is true. Otherwise, continue sequentially:

- CBZ register, L1

  if (register == 0) branch to instruction labeled L1;

- CBNZ register, L1

  if (register != 0) branch to instruction labeled L1;

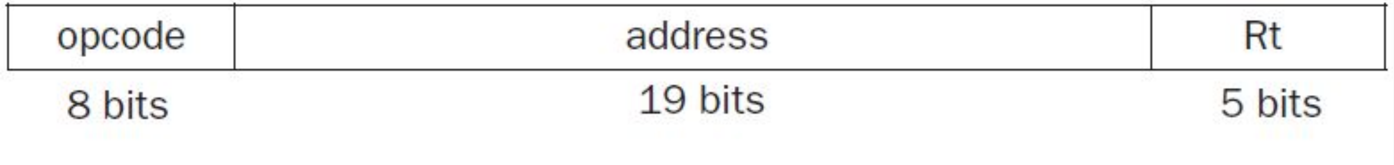
Branch **unconditionally** to instruction labeled L1:

- B L1

Branch Addressing

- CB-type

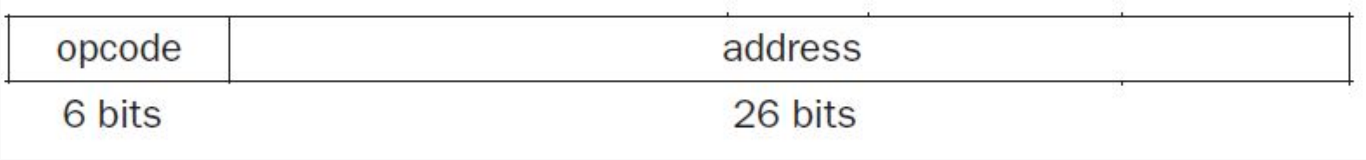
```
CBNZ X19, Exit    // go to Exit if X19 != 0
```



opcode	address	Rt
8 bits	19 bits	5 bits

- B-type

```
B loop     // go to loop
```



opcode	address
6 bits	26 bits

Both addresses (n° instructions) are PC-relative

New_PC = Current_PC + (n° instructions * 4)

Building a Datapath

Datapath

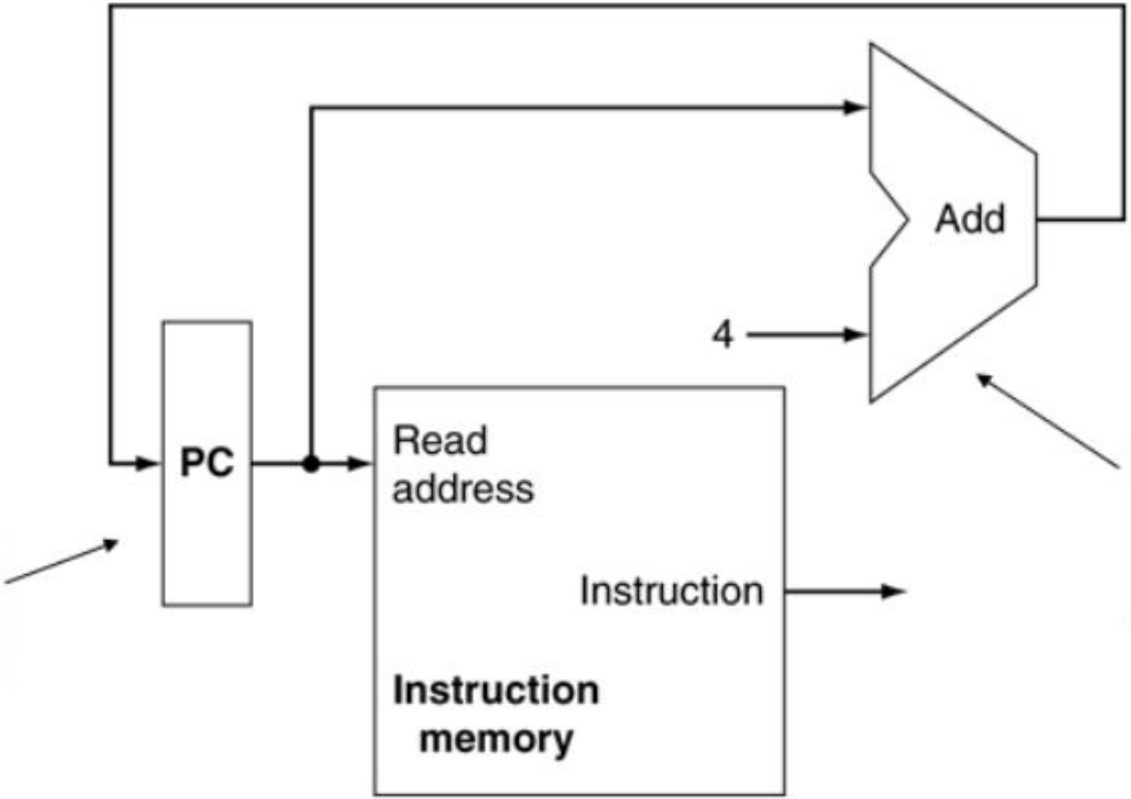
- Elements that process data and addresses in the CPU

  Registers, ALUs, mux’s, memories…

- We will build a LEGv8 datapath incrementally

  Refining the overview design

Instruction Fetch

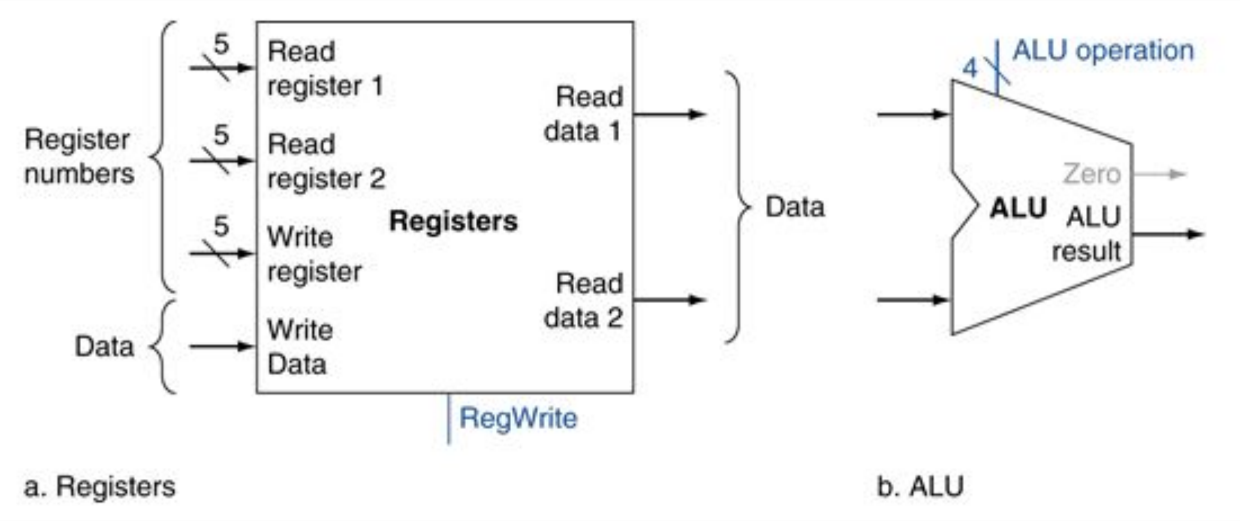


64-bit register

Increment by 4 for next instruction

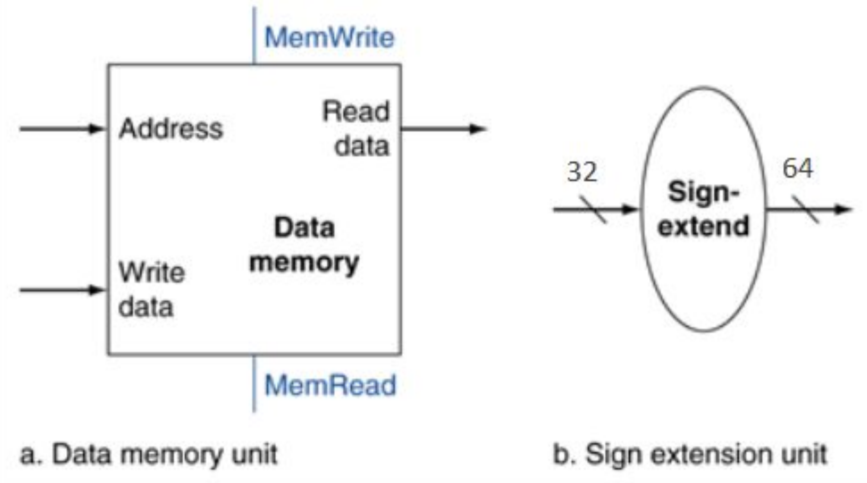
R-Format Instructions

- Read two register operands
- Perform arithmetic/logical operation
- Write register result



Load/Store Instructions (64 bits)

- Read register operands
- Calculate address using 9-bit offset (use ALU)
- Load: Read memory and update register
- Store: Write register value to memory



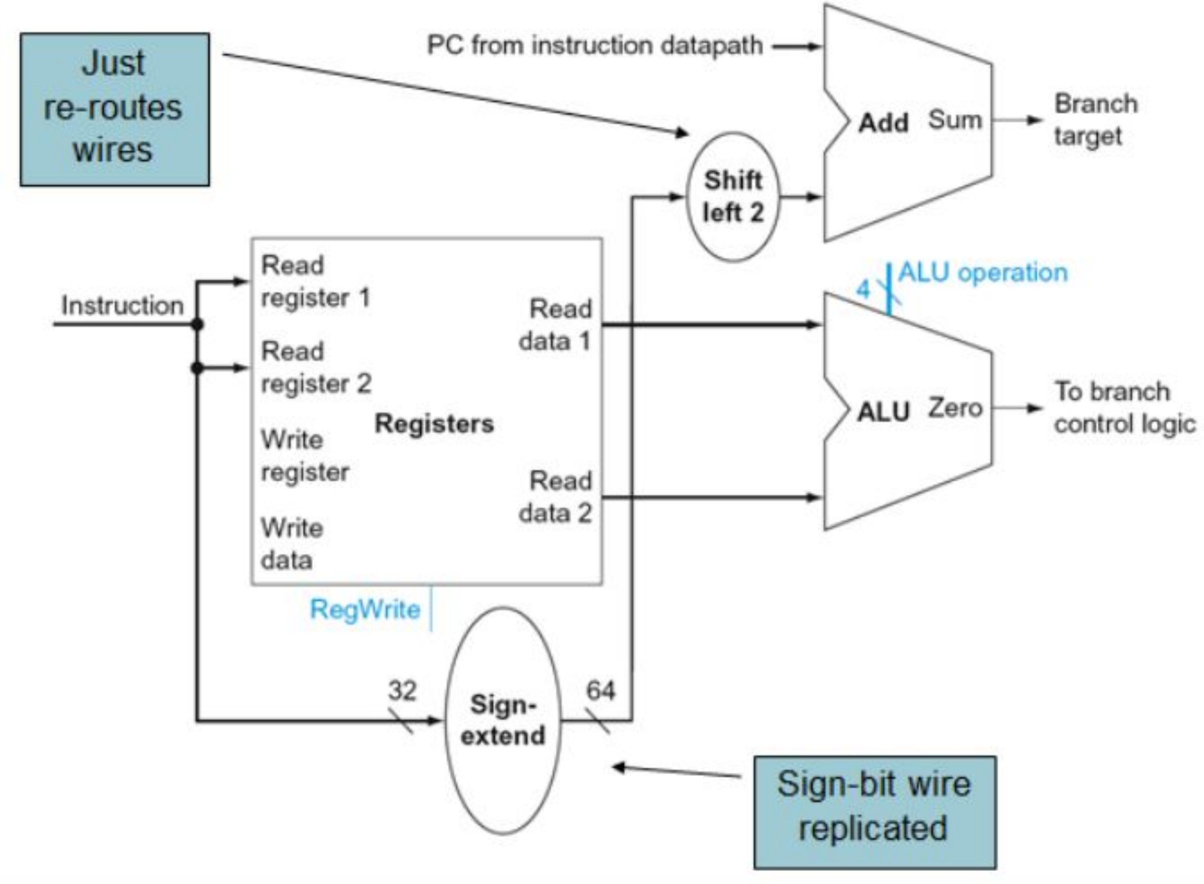
a. Data memory unit

b. Sign extension unit

Branch Instruction (CBZ)

- Read register operand
- Compare operand
    - Use ALU and check Zero output
- Calculate target address
    - Sign-extend displacement
    - Shift left 2 places (word displacement)
- Add to PC

Branch Instruction (CBZ)

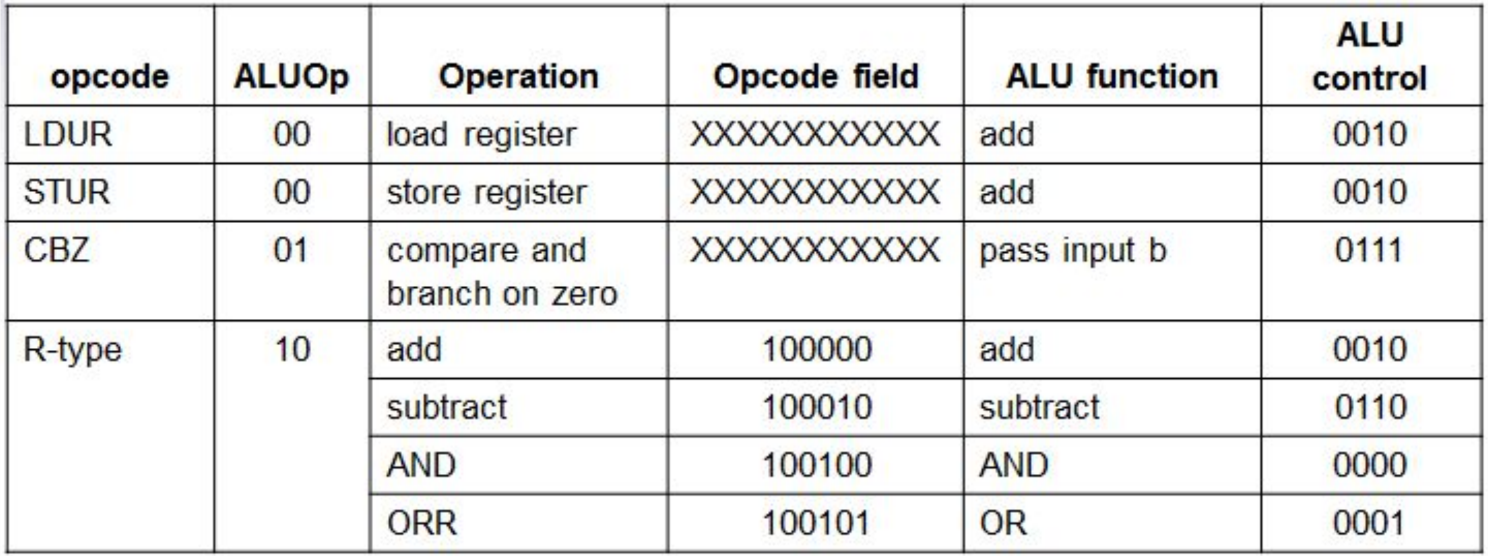


Just re-routes wires

Sign-bit wire replicated

ALU Control

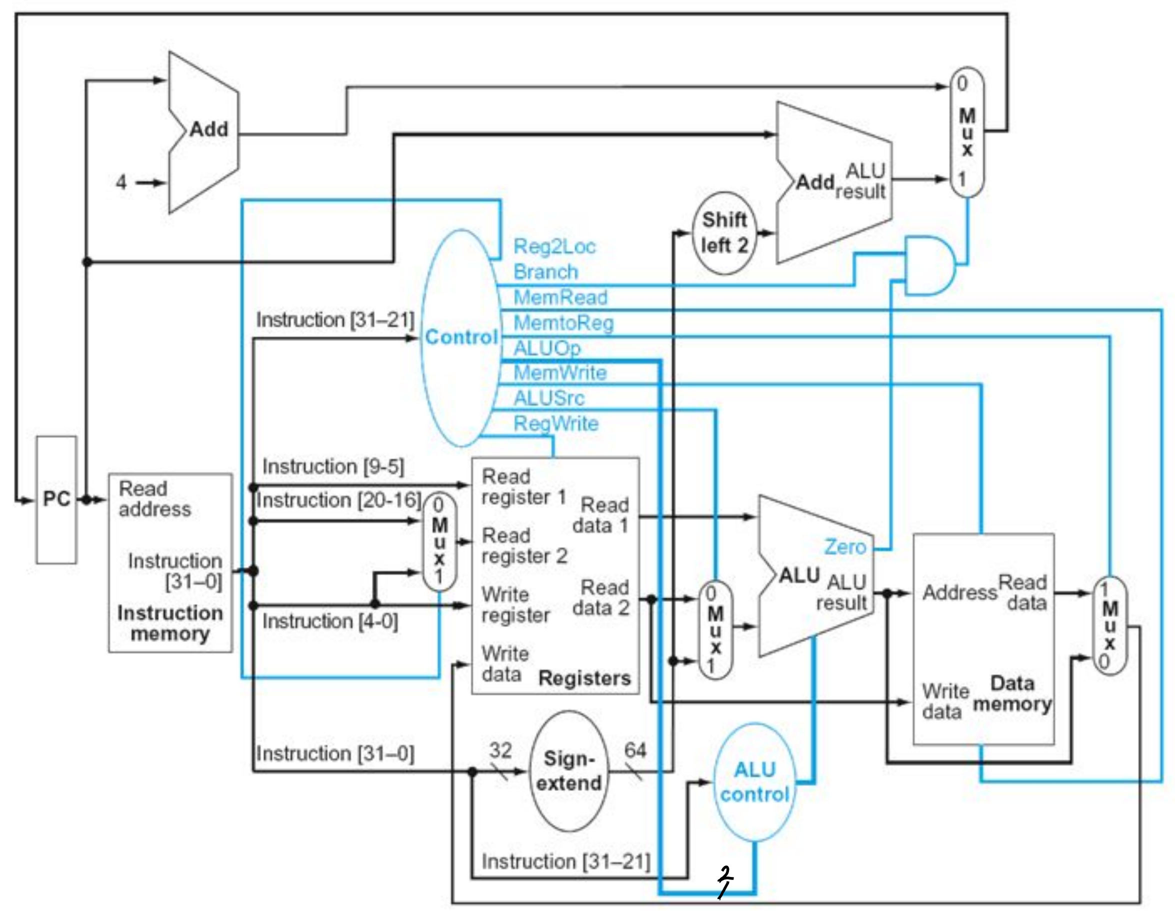
- ALU used for
  - Load/Store: F = add
  - Branch: F = pass input b
  - R-type: F depends on opcode



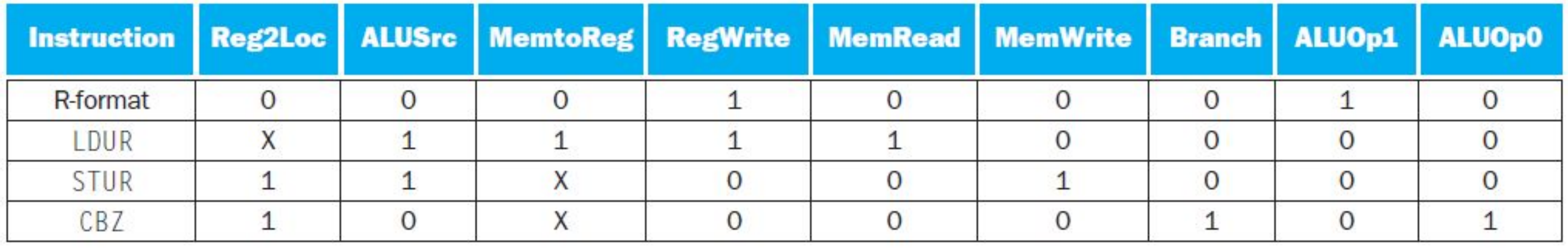
opcode	ALUOp	Operation	Opcode field	ALU function	ALU control
LDUR	00	load register	XXXXXXXXXXX	add	0010
STUR	00	store register	XXXXXXXXXXX	add	0010
CBZ	01	compare and branch on zero	XXXXXXXXXXX	pass input b	0111
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		ORR	100101	OR	0001

uso opcode p/ direct. Tipe R

Datapath With Control

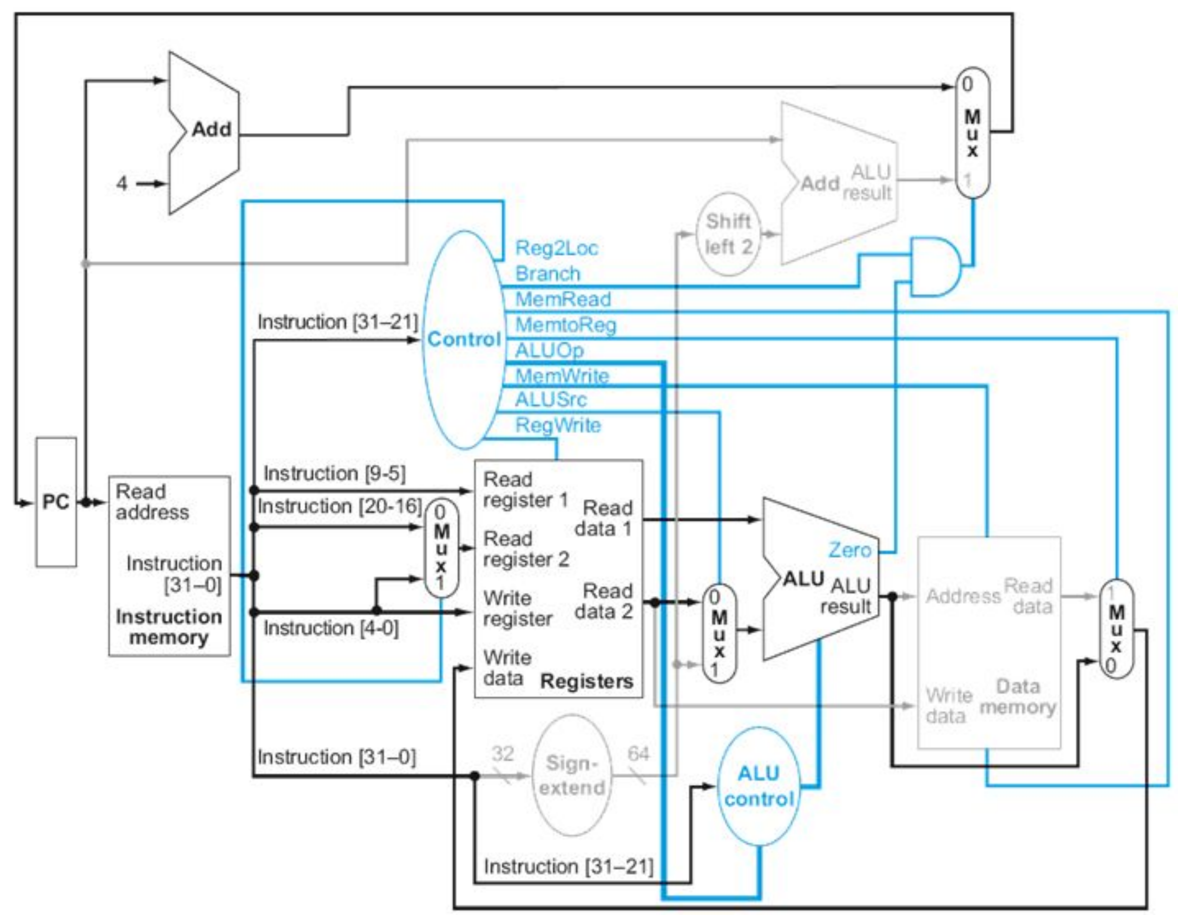


Setting of the control lines

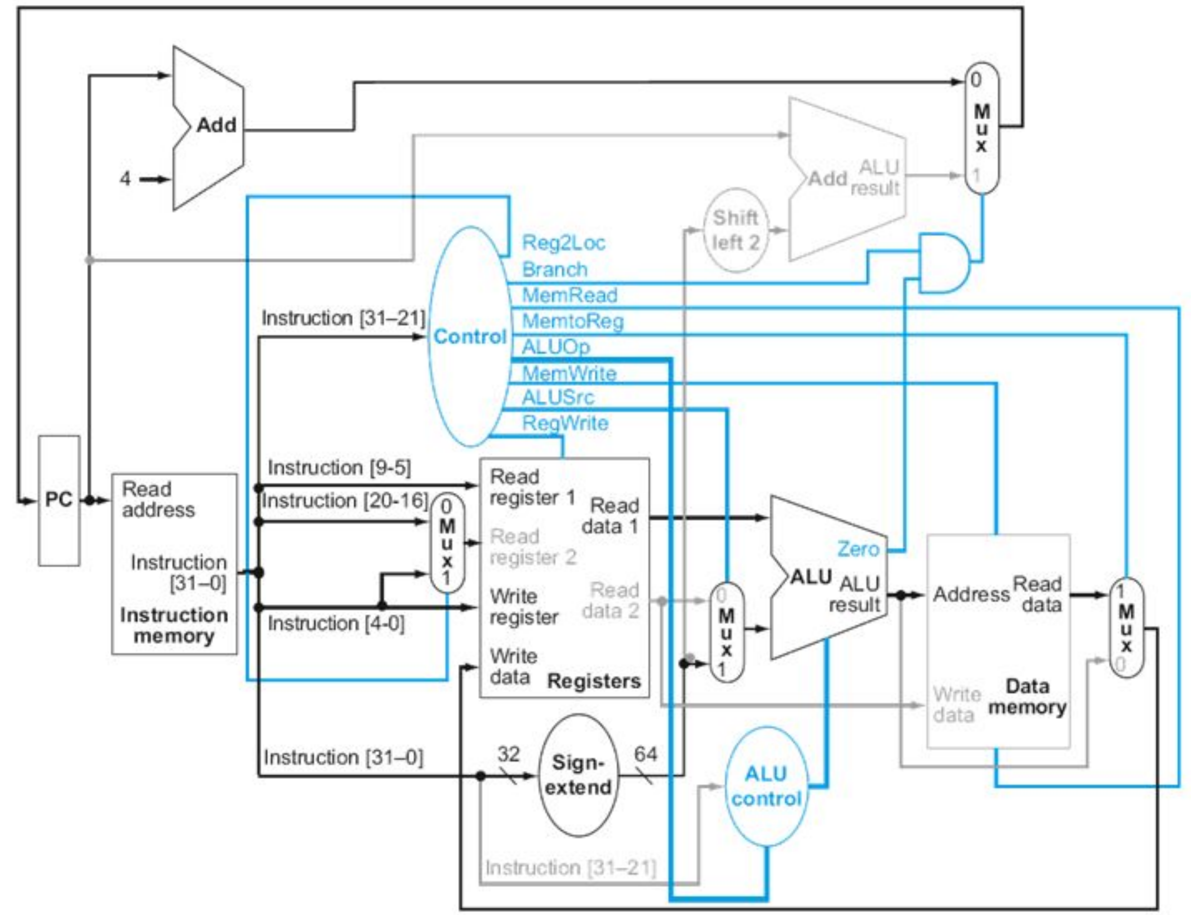


Instruction	Reg2Loc	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
R-format	0	0	0	1	0	0	0	1	0
LDUR	X	1	1	1	1	0	0	0	0
STUR	1	1	X	0	0	1	0	0	0
CBZ	1	0	X	0	0	0	1	0	1

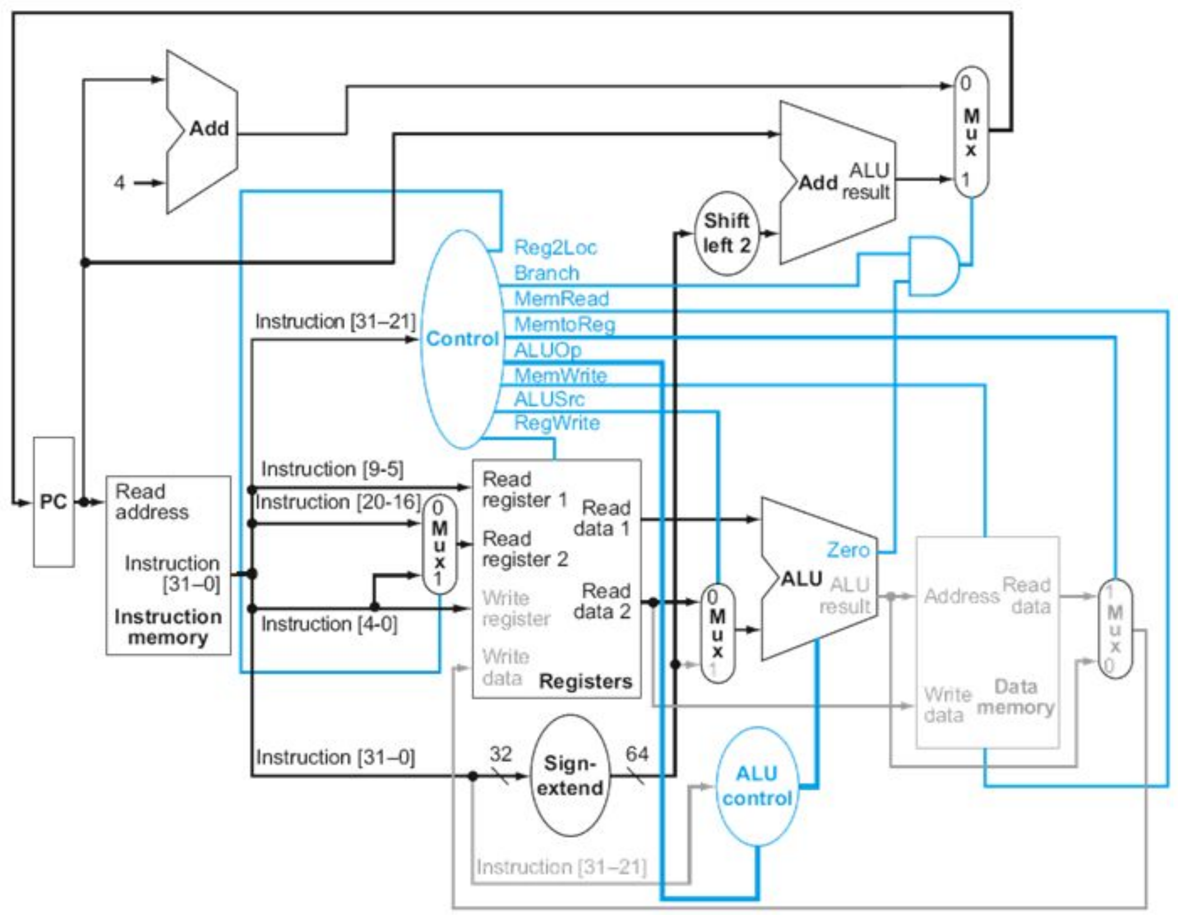
R-Type Instruction



Load Instruction



CBZ Instruction


Add
4
Add
ALU
result
Shift
left 2
0
M
u
x
1
Reg2Loc
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–21]
Instruction [9-5]
Instruction [20-16]
Instruction [4-0]
Instruction [31–0]
PC
Read
address
Instruction
[31–0]
Instruction
memory
0
M
u
x
1
Read
register 1
Read
register 2
Write
register
Write
data
Read
data 1
Read
data 2
Registers
0
M
u
x
1
Zero
ALU
ALU
result
Address
Read
data
Write
data
Data
memory
1
M
u
x
0
32
Sign-
extend
64
ALU
control
Instruction [31–21]