Datapath

Feb 22, 1996

Topics

- **datapath building blocks**
- **simple datapath design**

MIPS architecture subset

R-type instructions (add, sub, and, or, slt): rd <-- rs funct rt

0	rs	rt	rd	shamt	funct
31-26	25-21	20-16	15-11	10-6	5-0

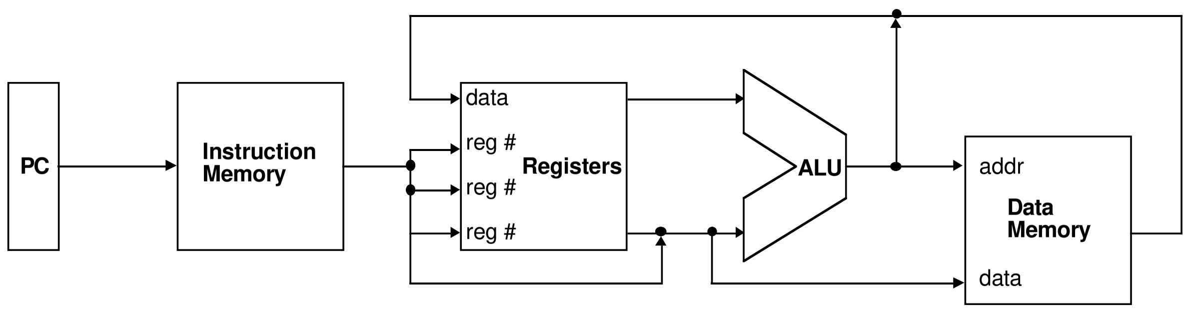
Load: rt <-- Mem[rs + address] **Store:** Mem[rs + address] <-- rt

35 or 43	rs	rt	address
31-26	25-21	20-16	15-0

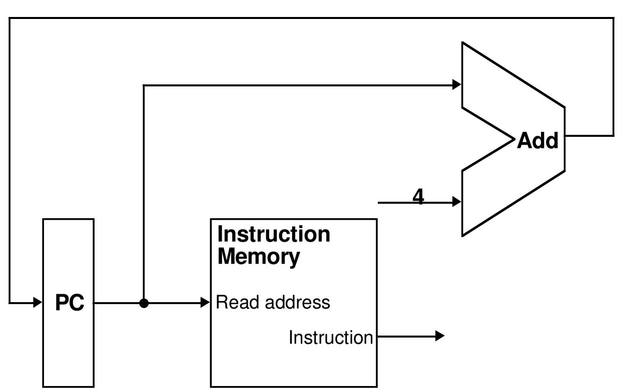
Branch equal: PC <-- ((rs - rt) == 0) ? PC + 4 + address : PC + 4

35 or 43	rs	rt	address
31-26	25-21	20-16	15-0

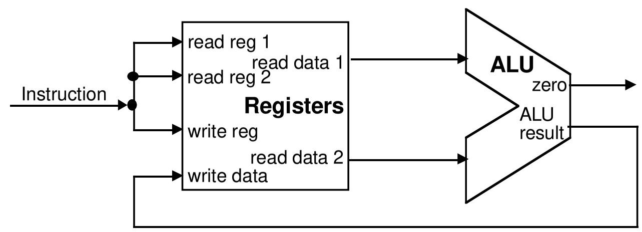
Abstract View of Datapath



Instruction fetch and PC increment



R-type instructions



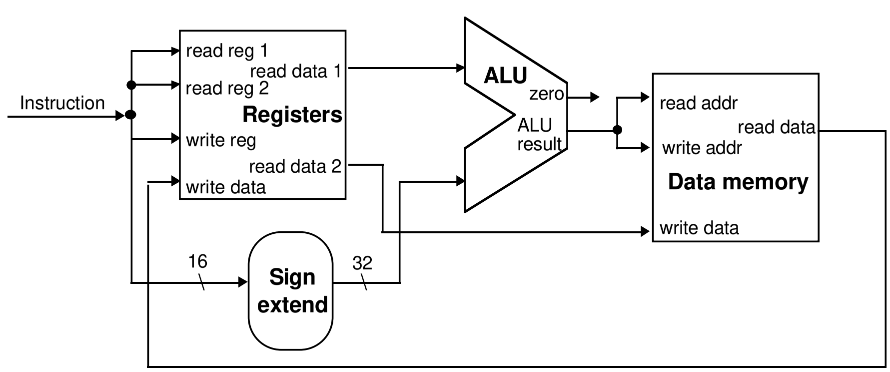
R-type instructions (add, sub, and, or, slt)

Read operands from register file

Compute value

Write back to register file

Load and store logic



Load

Compute address

Read from memory

Write to register

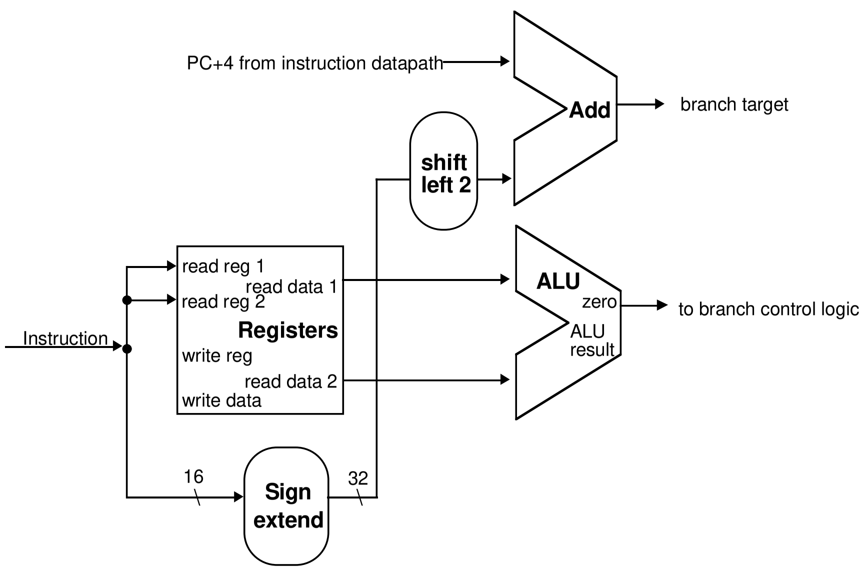
Store

Compute address

Read from register

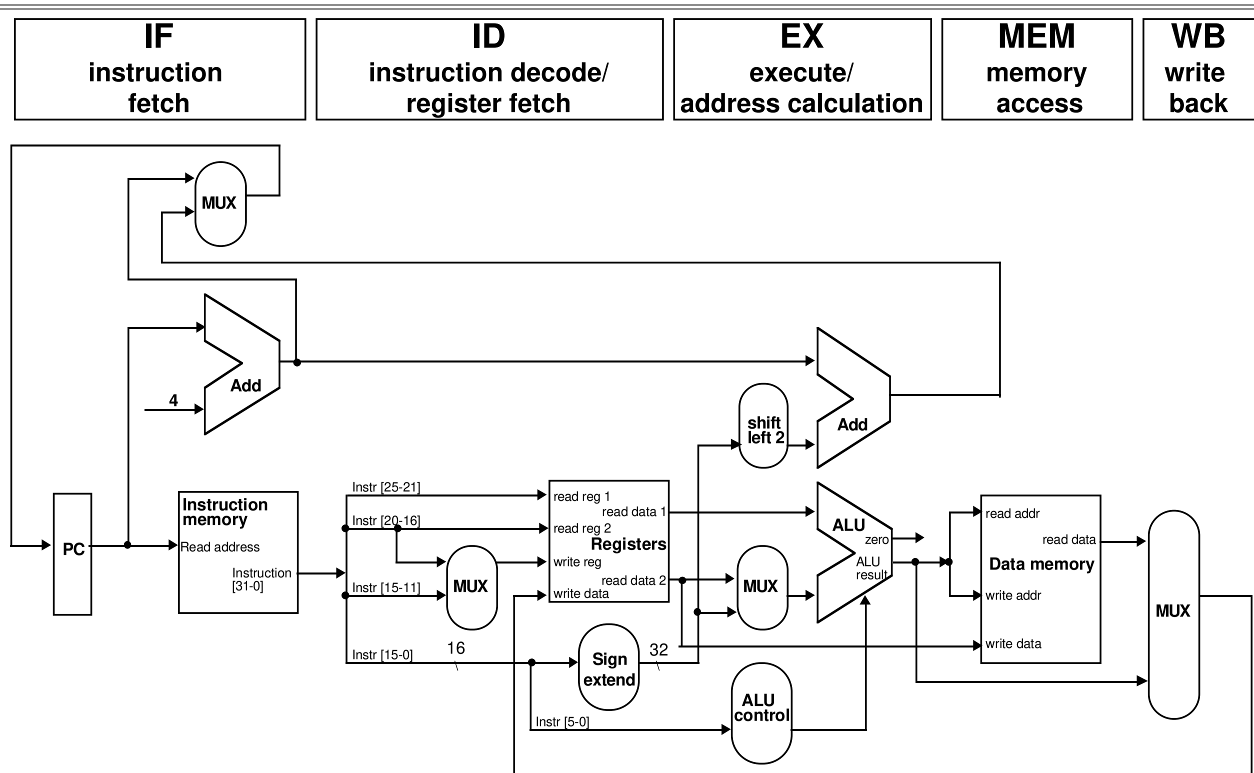
Write to memory

Branch on equal logic



fetch arguments
subract
compute target
select new PC

Datapath



R-type instructions

IF: Instruction fetch

- IR <-- IMemory[PC]
- PC <-- PC + 4

ID: Instruction decode/register fetch

- A <-- Register[IR[25:21]]
- B <-- Register[IR[20-16]]

Ex: Execute

- ALUOutput <-- A op B

MEM: Memory

- nop

WB: Write back

- Register[IR[15:11]] <-- ALUOutput

Load instruction

IF: Instruction fetch

- IR <-- IMemory[PC]
- PC <-- PC + 4

ID: Instruction decode/register fetch

- A <-- Register[IR[25:21]]
- B <-- Register[IR[20:16]]

Ex: Execute

- ALUOutput <-- A + <IR[15]^17 # IR[14:0]>

MEM: Memory

- Mem-Data <-- DMemory[ALUOutput]

WB: Write back

- Register[IR[20:16]] <-- Mem-Data

(note: # -> concatenate, ^ -> sign extend)

Store instruction

IF: Instruction fetch

- IR <-- IMemory[PC]
- PC <-- PC + 4

ID: Instruction decode/register fetch

- A <-- Register[IR[25:21]]
- B <-- Register[IR[20:16]]

Ex: Execute

- ALUOutput <-- A + <IR[15]^17 # IR[14:0]>

MEM: Memory

- DMemory[ALUOutput] <-- B

WB: Write back

- nop

Branch on equal

IF: Instruction fetch

- IR <-- IMemory[PC]
- PC <-- PC + 4

ID: Instruction decode/register fetch

- A <-- Register[IR[25:21]]
- B <-- Register[IR[20:16]]

Ex: Execute

- Target <-- PC + <IR[15]^15 # IR[14:0] # 00>
- Z <-- (A - B == 0)

MEM: Memory

- If (Z) PC <-- target

WB: Write back

- nop

Adding pipe registers