Organização de Computadores I
DCC006

Aula 6 – O Processador – Ciclo Único

Prof. Omar Paranaiba Vilela Neto

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Introdução

- Fatores de performance da CPU
  - Número de Instruções
    - Determinado pelo ISA e compilador
  - CPI e tempo de Ciclo
    - Determinado pela CPU hardware
- Nós vamos examinar duas implementações RISC-V + Um MIPS extra
  - Uma versão simplificada
  - Uma versão mais realista - Pipeline
- Subconjunto simples, mostra muitos aspectos
  - Referência à Memória: `ld`, `sd`
  - Aritmética/lógica: `add`, `sub`, `and`, `or`
  - Transferência de Controle: `beq`

Execução de Instrução

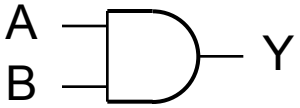
- PC → memória de instrução, fetch instrução
- Números de Registradores→ arquivo de registradores, lê registradores
- Dependendo da Classe da Instrução
  - Usa a ALU para calcular
    - Resultados Aritmético
    - Endereço de Memória para load/store
    - Comparação para o Branch
  - Acessa dado na memória para load/store
  - PC ← endereço de destino ou PC + 4

Básico do Design Lógico

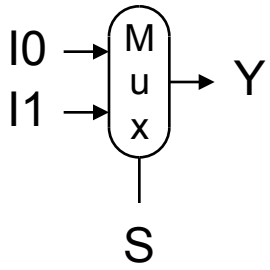
- Informação codificada em binário
  - Low voltage = 0, High voltage = 1
  - Um bit por fio
  - Dado multi-bit codificado em barramento de multi-fios
- Elemento combinacional
  - Opera em dados
  - Saída é em função das entradas
- Elementos de estado (sequencial)
  - Informação armazenada

Elementos Combinacionais

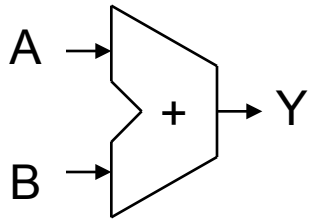
- AND-gate
  - Y = A & B



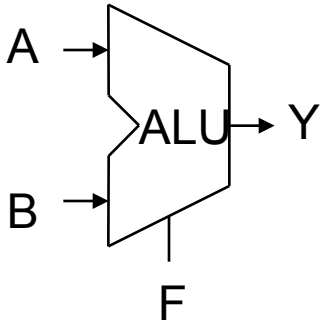
- Multiplexer
  - Y = S ? I1 : I0



- Adder
  - Y = A + B

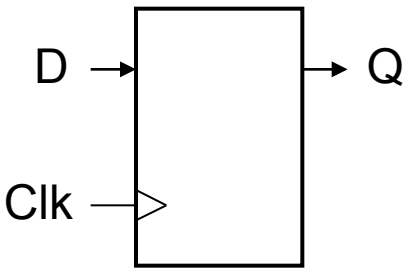


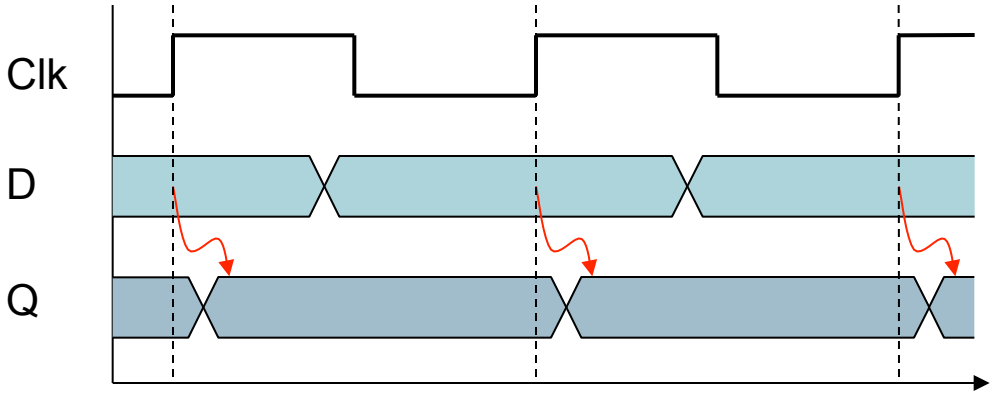
- Arithmetic/Logic Unit
  - Y = F(A, B)



Elementos Sequenciais

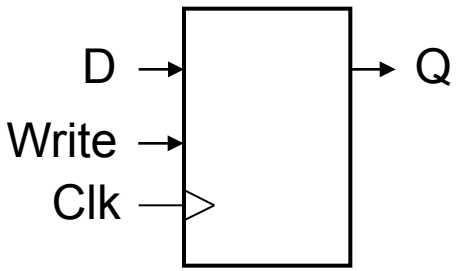
- **Registrador**: armazena dado em um circuito
  - Usa o sinal do clock para determinar quando o dado deve ser atualizado.
  - Edge-triggered: atualiza quando o Clk muda de 0 pra 1

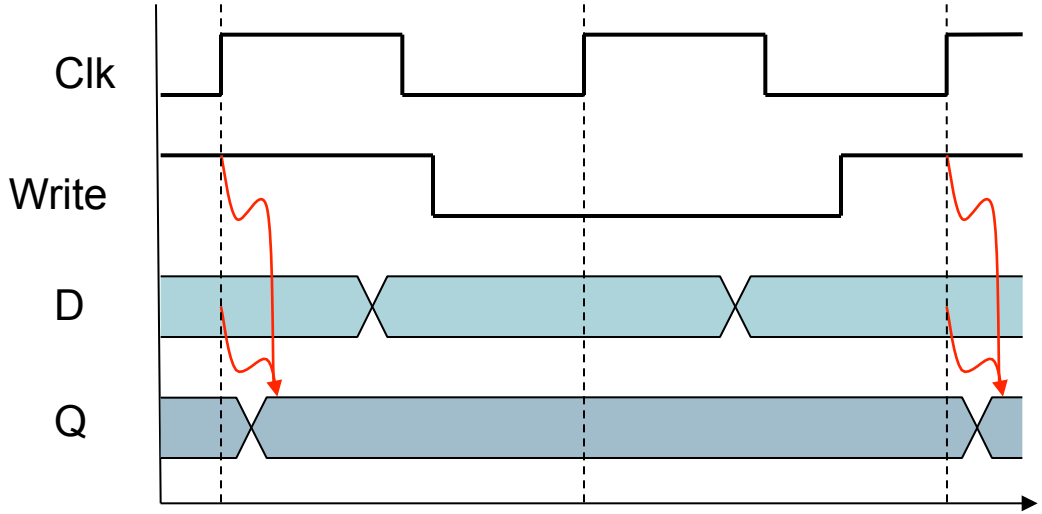




Elementos Sequenciais

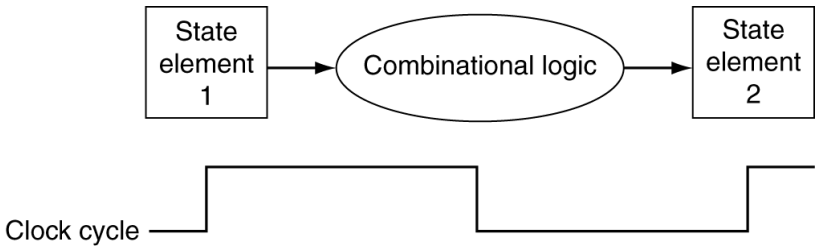
- **Registrador** com controle de escrita
  - Somente atualiza na transição do CLK quando o controle de escrita é 1.
  - Usado quando o valor armazenado é requerido depois.

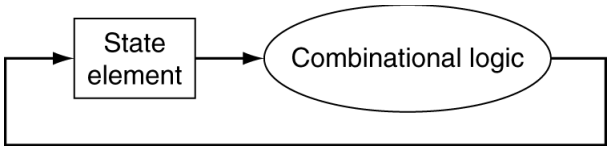




Metodologia de Clocking

- Lógica combinacional transforma o dado durante o ciclo de clock
  - Entre as transições do clock
  - Entradas de elemento de estados, Saídas para elemento de estados
  - Maior demora determina o período de clock

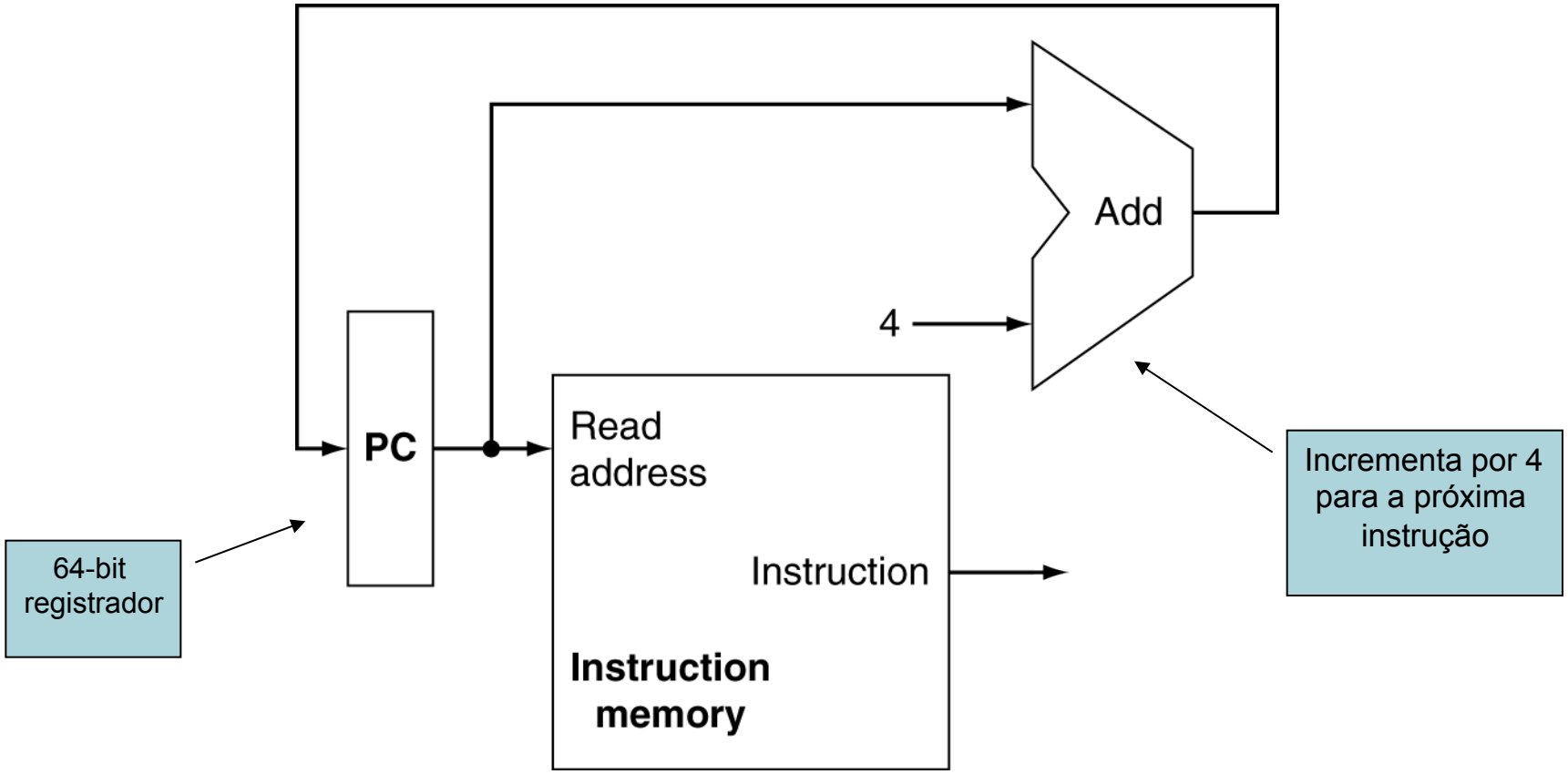




Construindo um Datapath

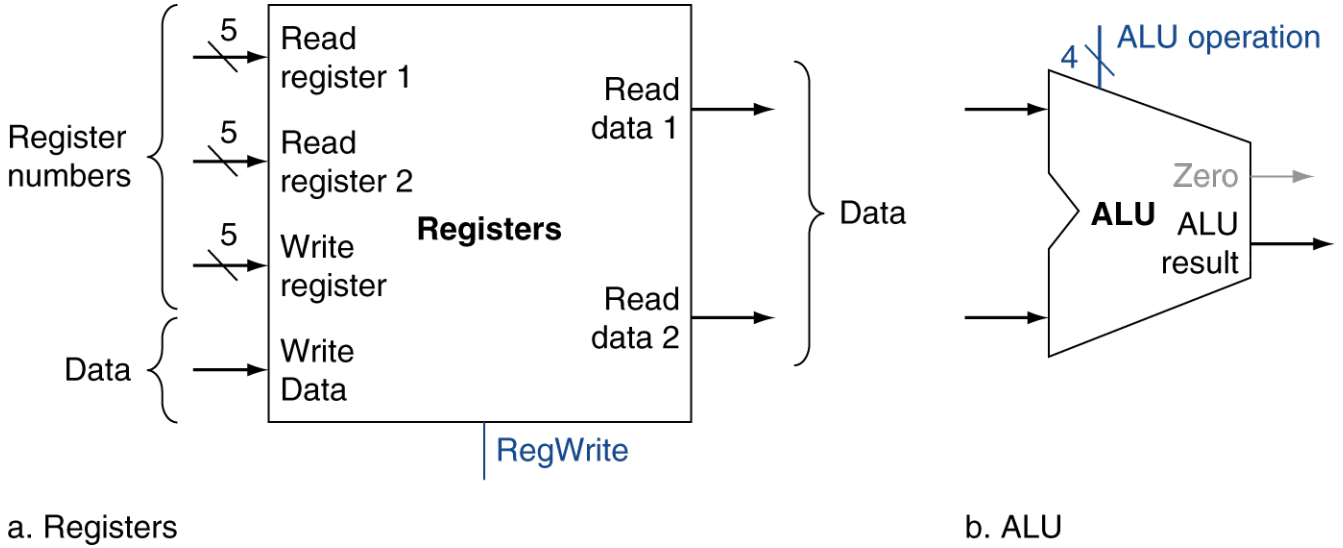
- Datapath (Caminho de dados)
  - Elementos que processam dados e endereços na CPU
    - Registradores, ALUs, mux's, memórias, …
- Nós vamos construir o datapath do RISC-V incrementalmente
  - Refinando a visão geral do desing

Busca de Instrução - Fetch



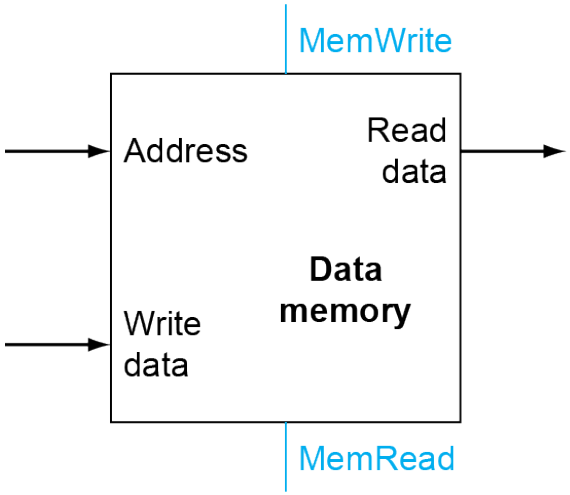
Instruções Tipo-R

- Lê dois operandos do Registradores
- Executa operação aritmética/lógica
- Escreve resultado no Registrador

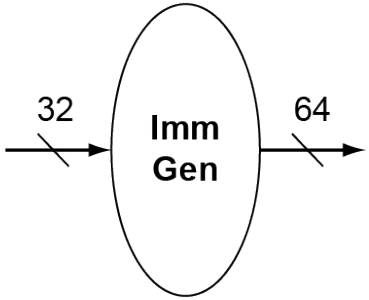


Instruções Load/Store

- Lê operando do registrador
- Calcula endereço usando 12-bit offset
  - Usa ALU, mas com offset de sinal estendido
- Load: Lê memória e atualiza registrador
- Store: Escreve valor de registrador na memória



a. Data memory unit

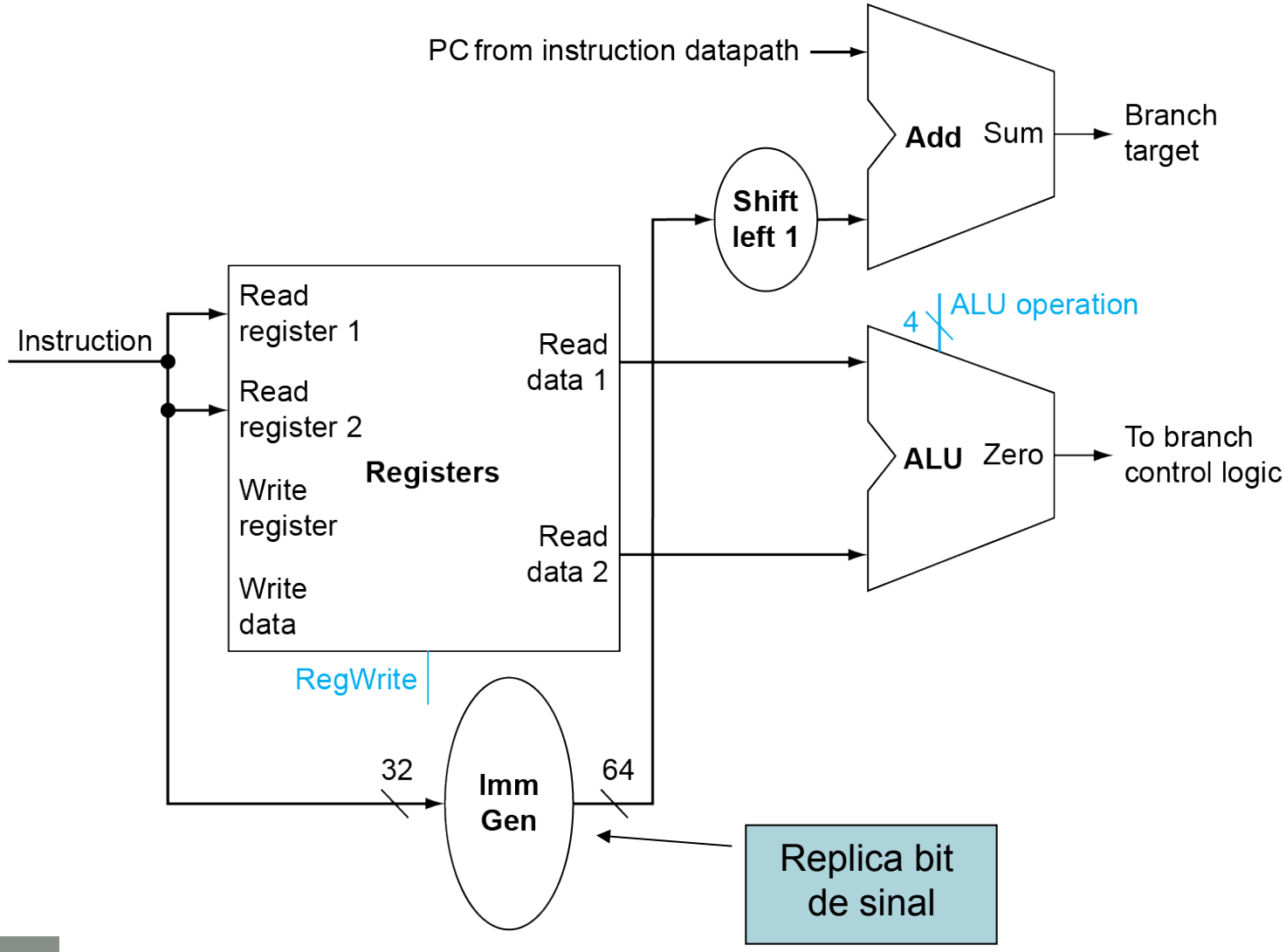


b. Immediate generation unit

Instruções Branch

- Lê operando do registrador
- Compara operandos
  - Usa ALU, subtrator e checa saída Zero
- Calcula endereço de destino
  - Sinal-estendido deslocado
  - Shift a esquerda 1 posição (halfword deslocamento)
  - Adiciona o valor do PC

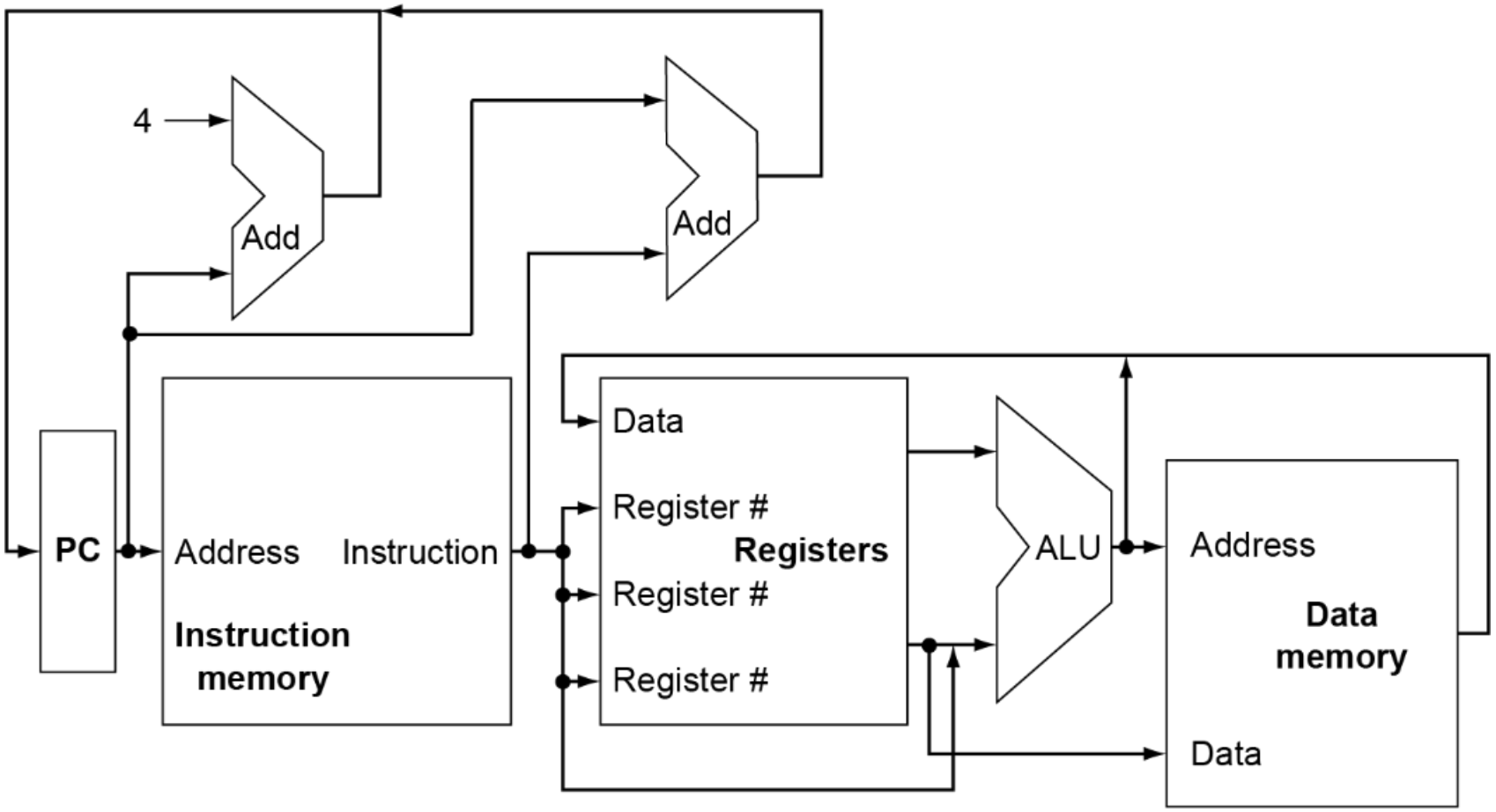
Instruções Branch



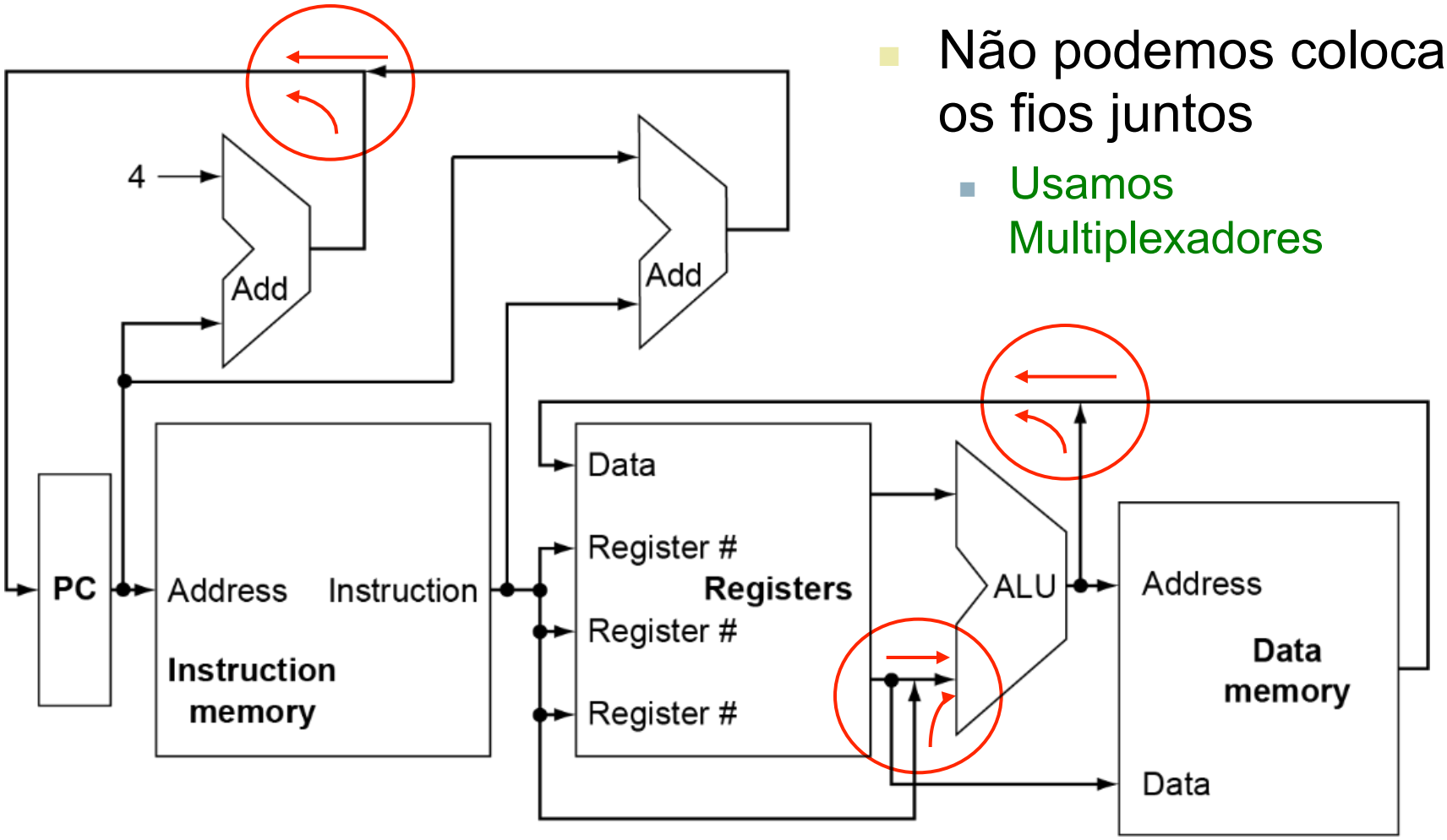
Compondo os elementos

- Primeira versão executa uma instrução em um ciclo de clock
  - Cada elemento do datapath só pode executar uma função por vez
  - Logo, precisamos separar memória de instruções e dados
- Usar multiplexadores onde fontes de dados alternados são usados para instruções diferentes

Visão Geral da CPU

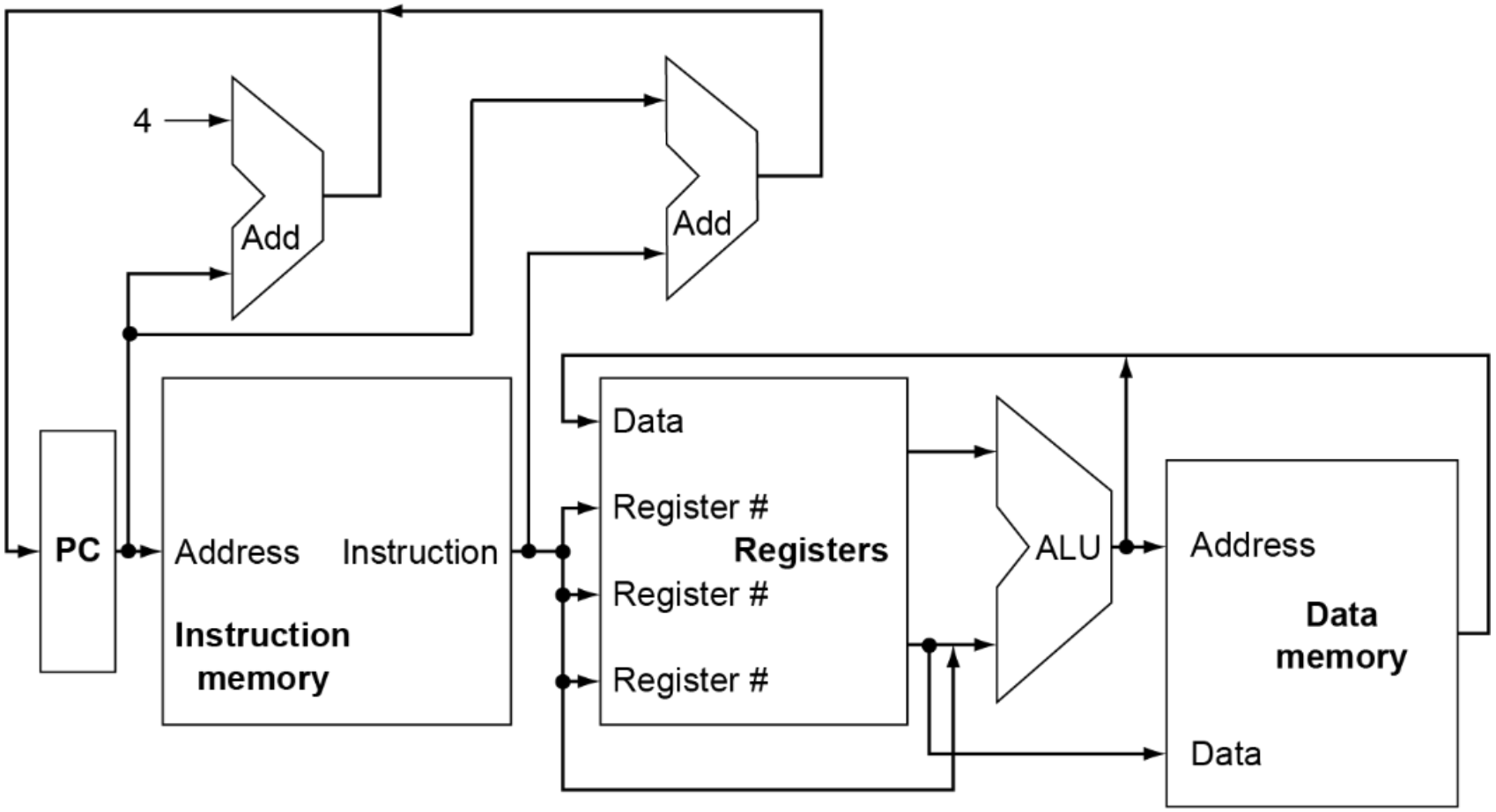


Multiplexadores

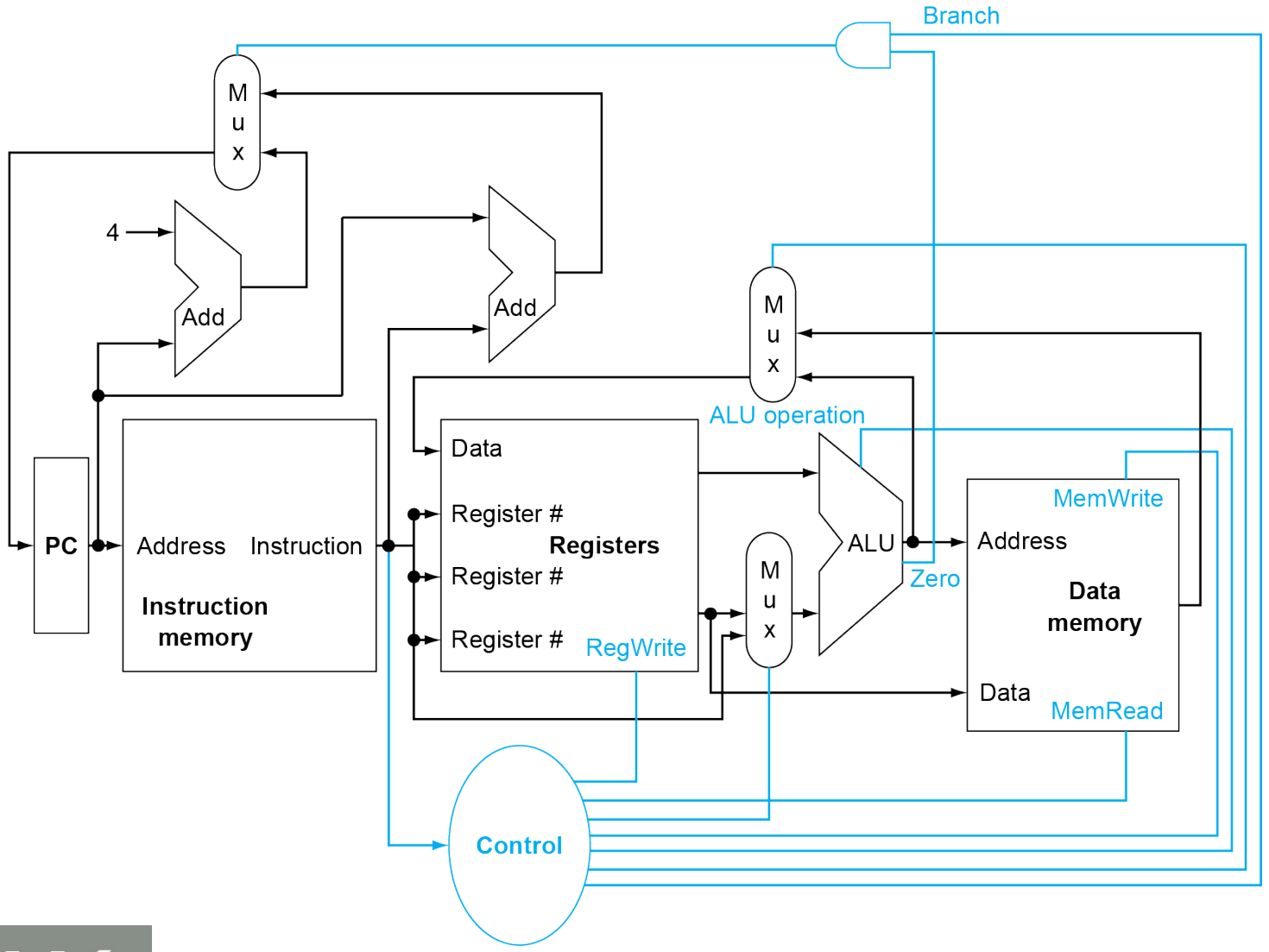


- Não podemos colocar os fios juntos
  - Usamos Multiplexadores

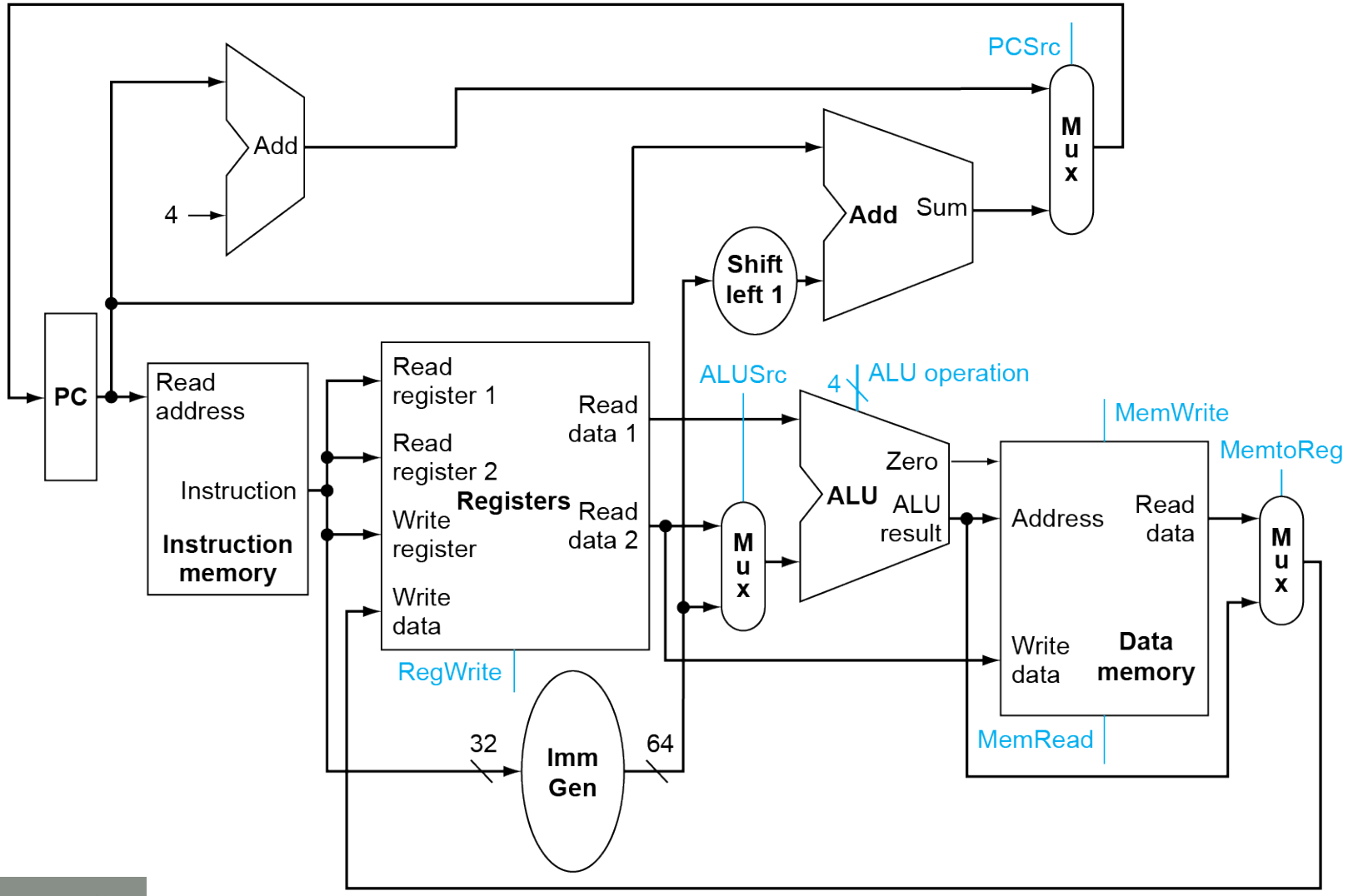
Quem controla tudo isso?



Controle


Branch
M
u
x
4
Add
Add
M
u
x
ALU operation
PC
Address
Instruction
Instruction
memory
Data
Register #
Registers
Register #
Register #
RegWrite
M
u
x
ALU
Zero
MemWrite
Address
Data
memory
Data
MemRead
Control

Datapath Completo



Controle da ALU

- ALU usada para
  - Load/Store: F = add
  - Branch: F = subtrai
  - R-type: F depende do opcode

Controle da ALU	Função
0000	AND
0001	OR
0010	add
0110	subtract

Controle da ALU

- Assuma 2-bit ALUOp derivado do opcode
  - Lógica combinacional deriva o controle da ALU

opcode	ALUOp	Operation	Opcode field	ALU function	ALU control
ld	00	load register	XXXXXXXXXXX	add	0010
sd	00	store register	XXXXXXXXXXX	add	0010
beq	01	branch on equal	XXXXXXXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001

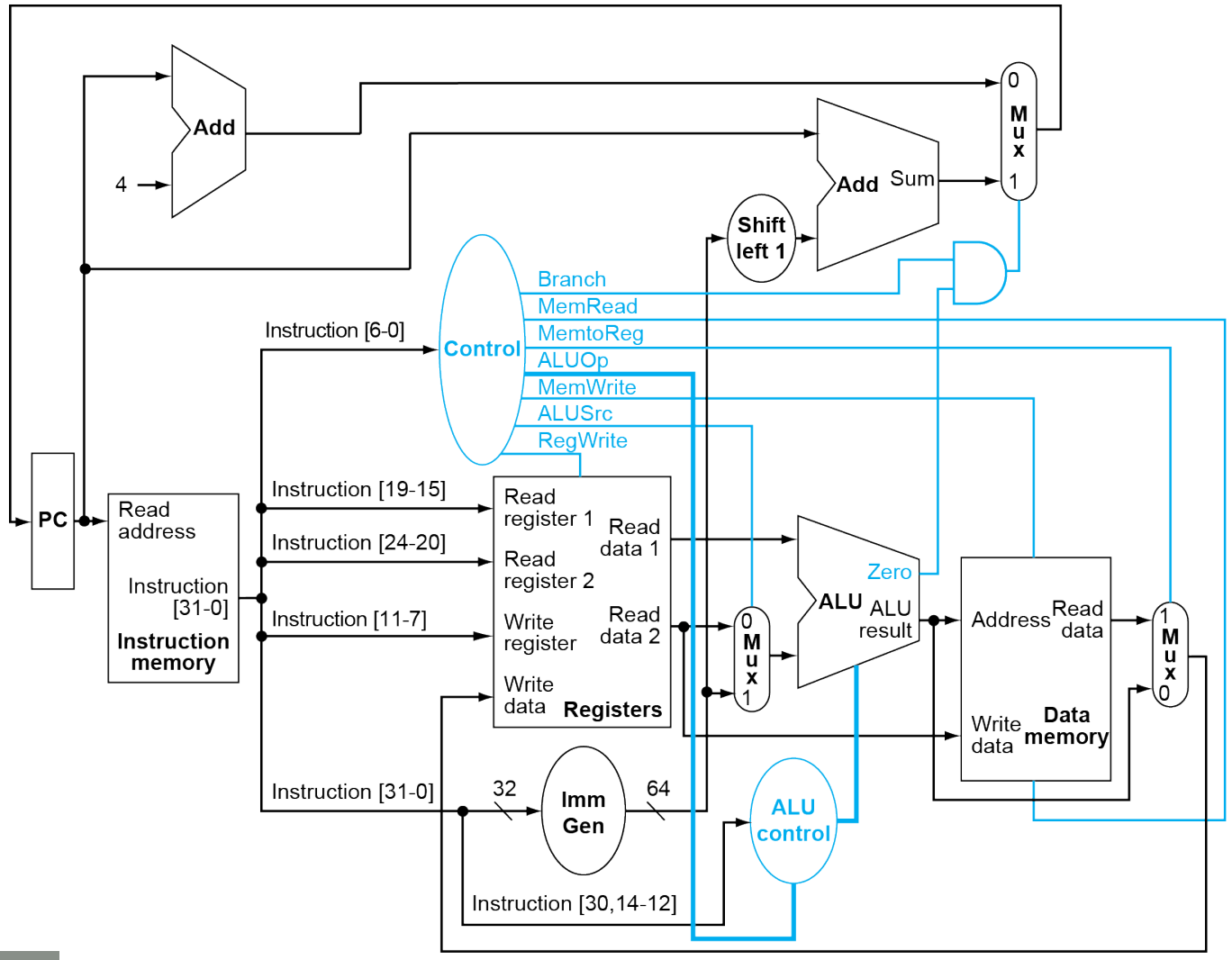
A Unidade de Controle Principal

- Sinais de controle derivados da instrução

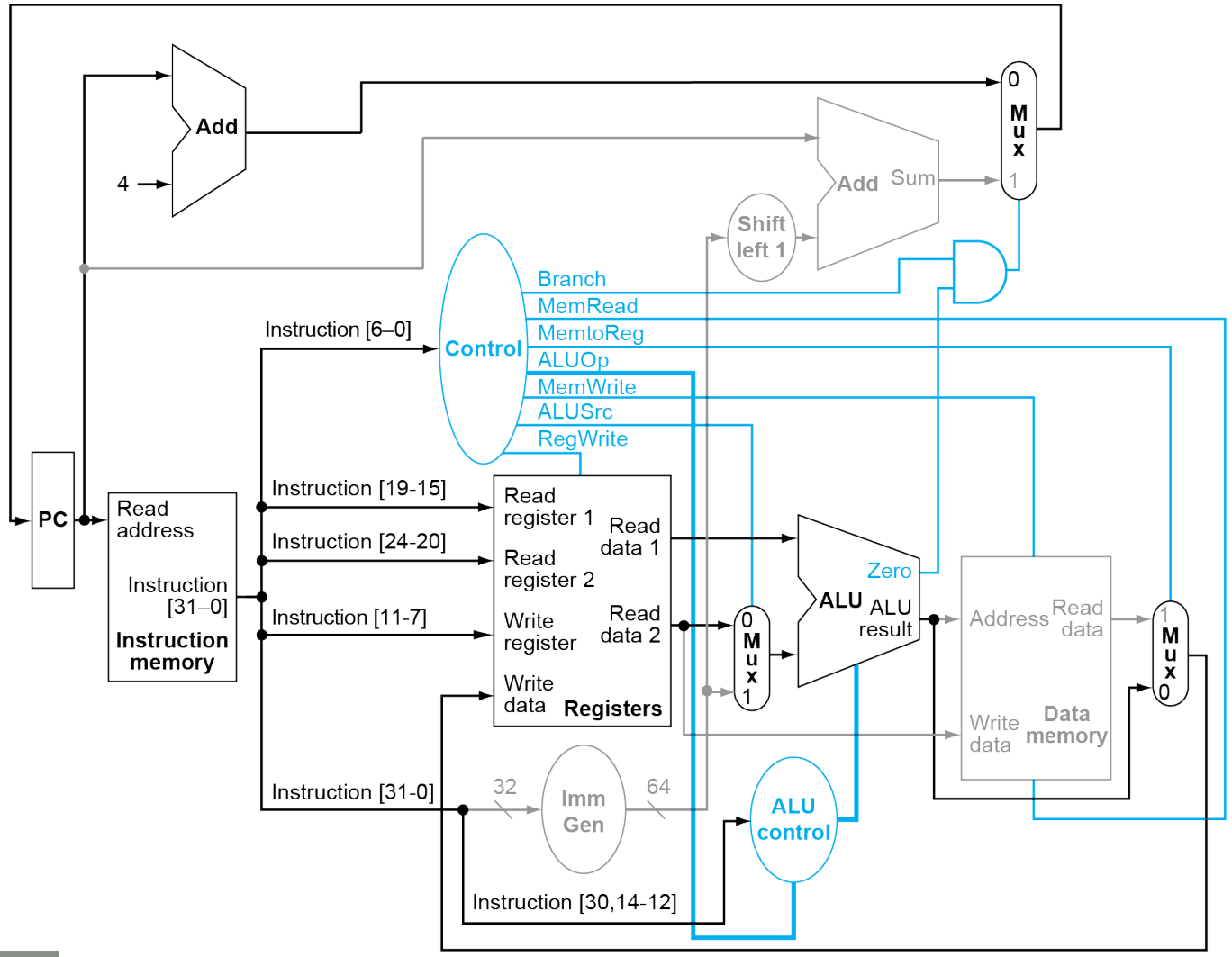
Name (Bit position)	Fields 31:25	24:20	19:15	14:12	11:7	6:0
(a) R-type	funct7	rs2	rs1	funct3	rd	opcode
(b) I-type	immediate[11:0]		rs1	funct3	rd	opcode
(c) S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode
(d) SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode

ALUOp		Funct7 field							Funct3 field			
ALUOp1	ALUOp0	I[31]	I[30]	I[29]	I[28]	I[27]	I[26]	I[25]	I[14]	I[13]	I[12]	Operation
0	0	X	X	X	X	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	X	X	X	X	0110
1	X	0	0	0	0	0	0	0	0	0	0	0010
1	X	0	1	0	0	0	0	0	0	0	0	0110
1	X	0	0	0	0	0	0	0	1	1	1	0000
1	X	0	0	0	0	0	0	0	1	1	0	0001

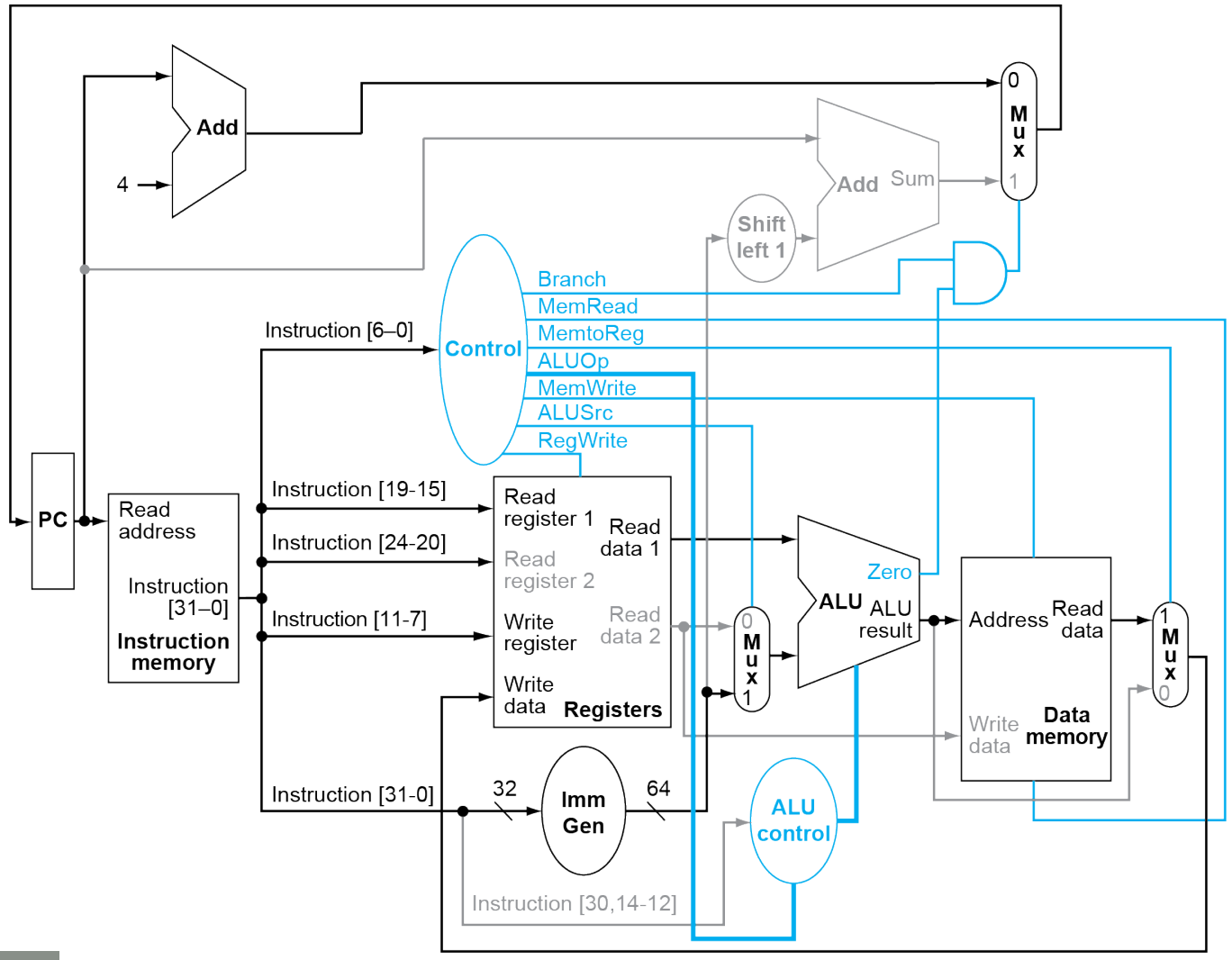
Datapath com Controle



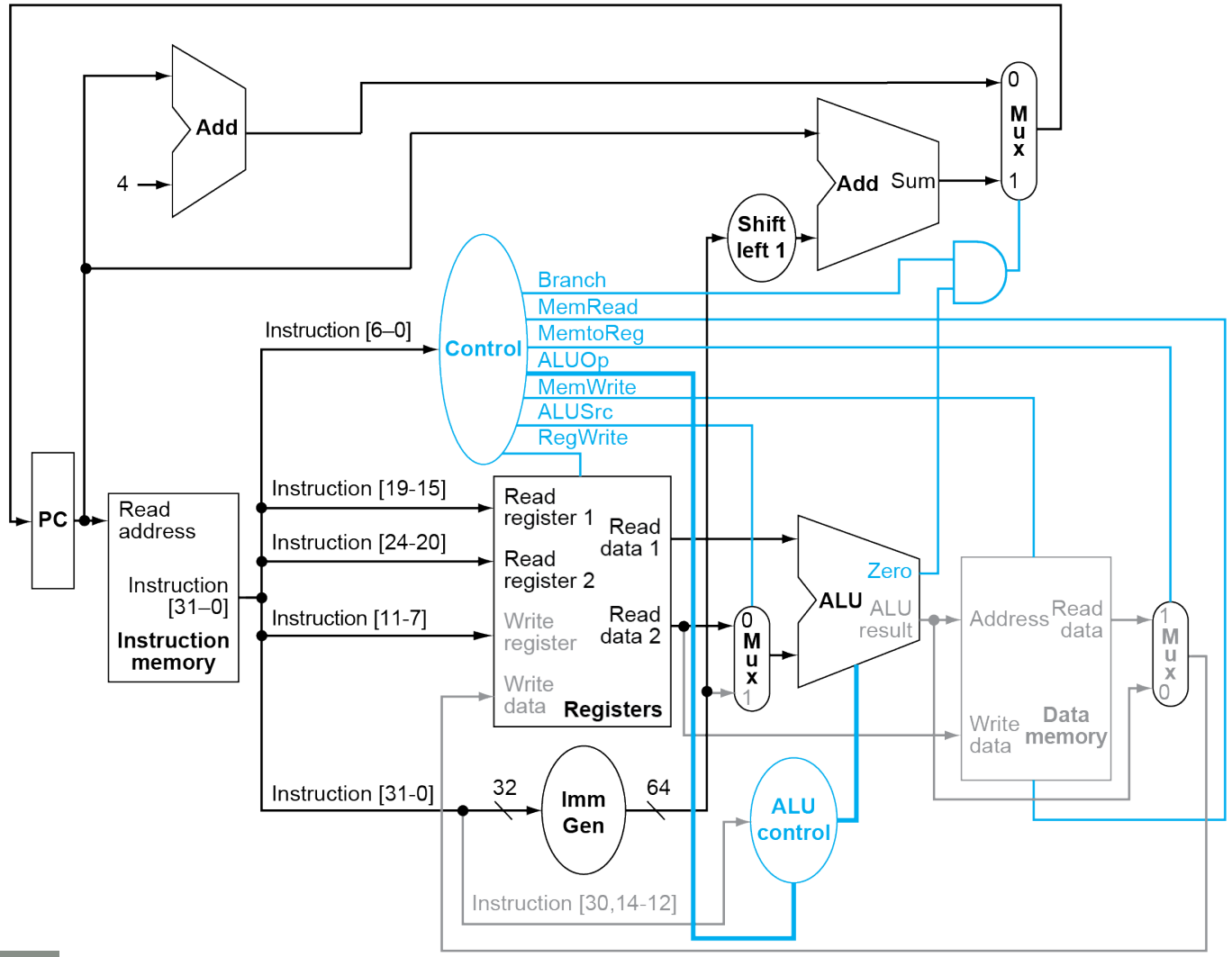
Instruções Tipo-R



Instruções Load



Instruções BEQ



Questões de Desempenho

- O maior atraso determina o período de clock
  - Caminho crítico: Instrução load
  - Instrução de memória → banco de registradores → ALU → memória de dados → banco de registradores
- Não é factível variar o período para diferentes instruções
- Viola princípio de design
  - Fazer o caso mais comum mais rápido