Arquitetura e Organização de Computadores

Capítulo 4 – O Processador

- Material adaptado de:

  Patterson e Henessy,
  Computer Organization and Design 4th Edition

  Prof. Onur Mutlu
  Carnegie Mellon University

- Capítulo 01 - Básico de arquitetura de computadores e desempenho:
  - Definições, CPI, MIPS, Energia, etc...

- Capítulo 02 - Conjunto de instruções e ISA do MIPS:
  - Tamanho da instrução;
  - Decodificação da instrução;
  - Número de registradores;
  - Modos de endereçamento;
  - Acesso alinhado vs. desalinhado;
  - Propriedades do CISC vs. RISC;
  - Visão geral do ISA do MIPS.

Relembrando

- Notas estão no AVA, verifiquem...
- Apenas 4 submissões...
  - O que está acontecendo?
- Aumento do peso dos projetos em relação as atividades de fixação:
  - 3x

Projeto 01 - Notas

CAPÍTULO 4

- Fatores de desempenho da CPU:
  - Número de instruções:
    - Determinado pelo ISA e compilador;
  - CPI e Tempo do ciclo:
    - Determinado pelo hardware da CPU;

- Nós iremos examinar duas implementações do MIPS:
  - Uma versão simplificada – multi-ciclo;
  - Uma versão mais realista - pipeline;

- Subconjunto simples, que mostra a maioria dos aspectos:
  - Referência à memória: lw, sw;
  - Aritmética/Lógica: add, sub, and, or, slt;
  - Transferência de controle: beq, j;

Introdução

- O que significa processar uma instrução?
- Relembre o modelo de von Neumann:

AS = Estado da arquitetura antes de uma instrução ser processada

⇩

Processar instrução

⇩

AS' = Estado da arquitetura depois de uma instrução ser processada

- Processar uma instrução: Transformar AS em AS' de acordo com a especificação da instrução.

Como uma máquina processa instruções?

- ISA define de maneira abstrata o que AS' deveria ser, dado uma instrução e um AS:
  - Define uma máquina de estados finita onde:
    - State = estado visível ao programador;
    - Next-state logic = especificação da execução da instrução
  - Do ponto de vista do ISA, não há estados intermediários entre AS e AS' durante a execução de uma instrução:
    - Uma mudança de estado por instrução.

Processar instruções - ISA

Processar instruções - Microarquitetura

- Microarquitetura implementa como o AS é transformado em AS':
  - Existem muitas escolhas possíveis:
    - Cada um nós fará uma diferente;
  - Podemos utilizar estados invisíveis ao programador para otimizar a velocidade de execução da instrução:
    - Escolha 1: AS → AS' (transformar AS em AS' em um único ciclo de clock);
    - Escolha 2: AS → AS+MS1 → AS + MS2 → AS' (transformar AS em AS' em vários ciclos de clock).

- Cada instrução gasta um ciclo de clock para executar;
- É utilizada apenas lógica combinacional para implementar a execução da instrução:
  - Não há estados intermediários (invisíveis ao programador);

AS = Estado da arquitetura antes de uma instrução ser processada

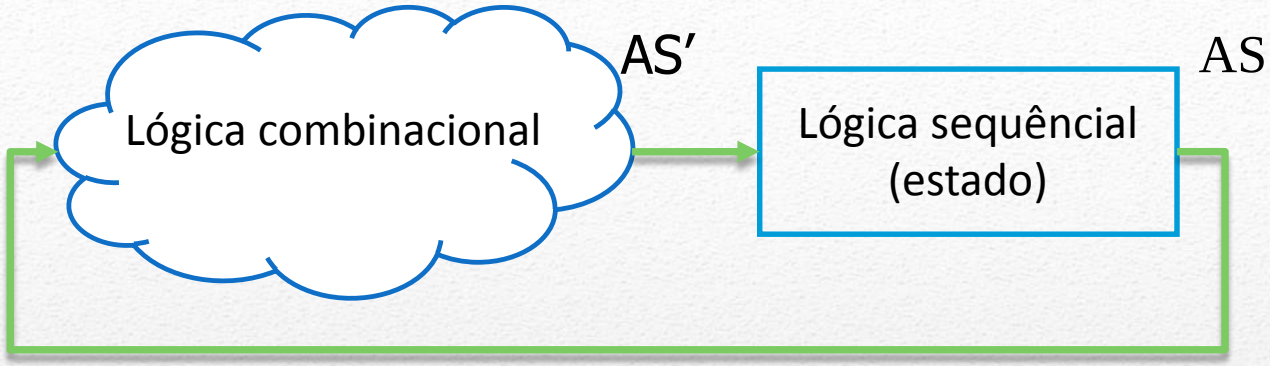
⇩

Processar instrução em um ciclo de clock

⇩

AS' = Estado da arquitetura depois de uma instrução ser processada

Single-cycle archtecture



- O que determina o tempo do ciclo de clock?
- O que determina o caminho crítico da lógica combinacional?

Single-cycle archtecture

- Máquinas single-cycle:
  - Cada instrução gasta um único ciclo de clock;
  - Toda a atualização do estado é realizada ao final do ciclo de clock;
  - Desvantagem: instrução com ciclo mais lento determina o tempo do ciclo.

- Máquinas multi-cycle:
  - Processamento da instrução dividido em múltiplos ciclos/estágios;
  - Atualizações de estado são realizadas durante a execução da instrução;
  - Atualização do estado da arquitetura é realizado no último ciclo de clock;
  - Vantagem: estágio mais lento determina o tempo do ciclo.

Single-cycle vs. Multi-cycle

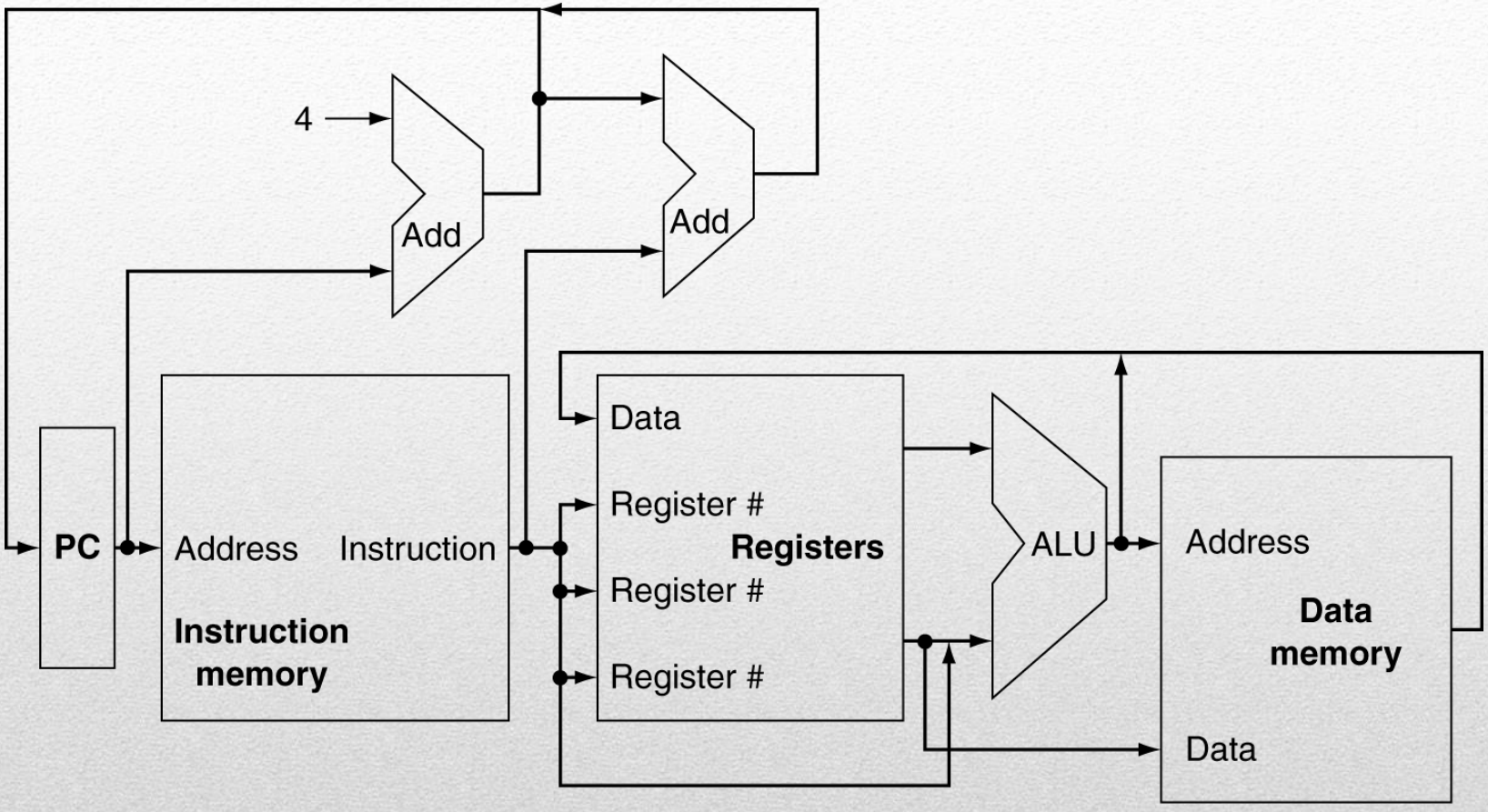
Ciclo da Instrução

- Ciclo da Instrução: sequência de passos para processar uma instrução;
- Instruções são executadas sob a direção da Unidade de Controle;
- Geralmente seis passos:
  1. Buscar (Fetch);
  2. Decodificar (Decode);
  3. Avaliar Endereços (Evaluate Address);
  4. Buscar Operandos (Fetch Operands);
  5. Executar (Execute);
  6. Armazenar Resultados (Write Result);

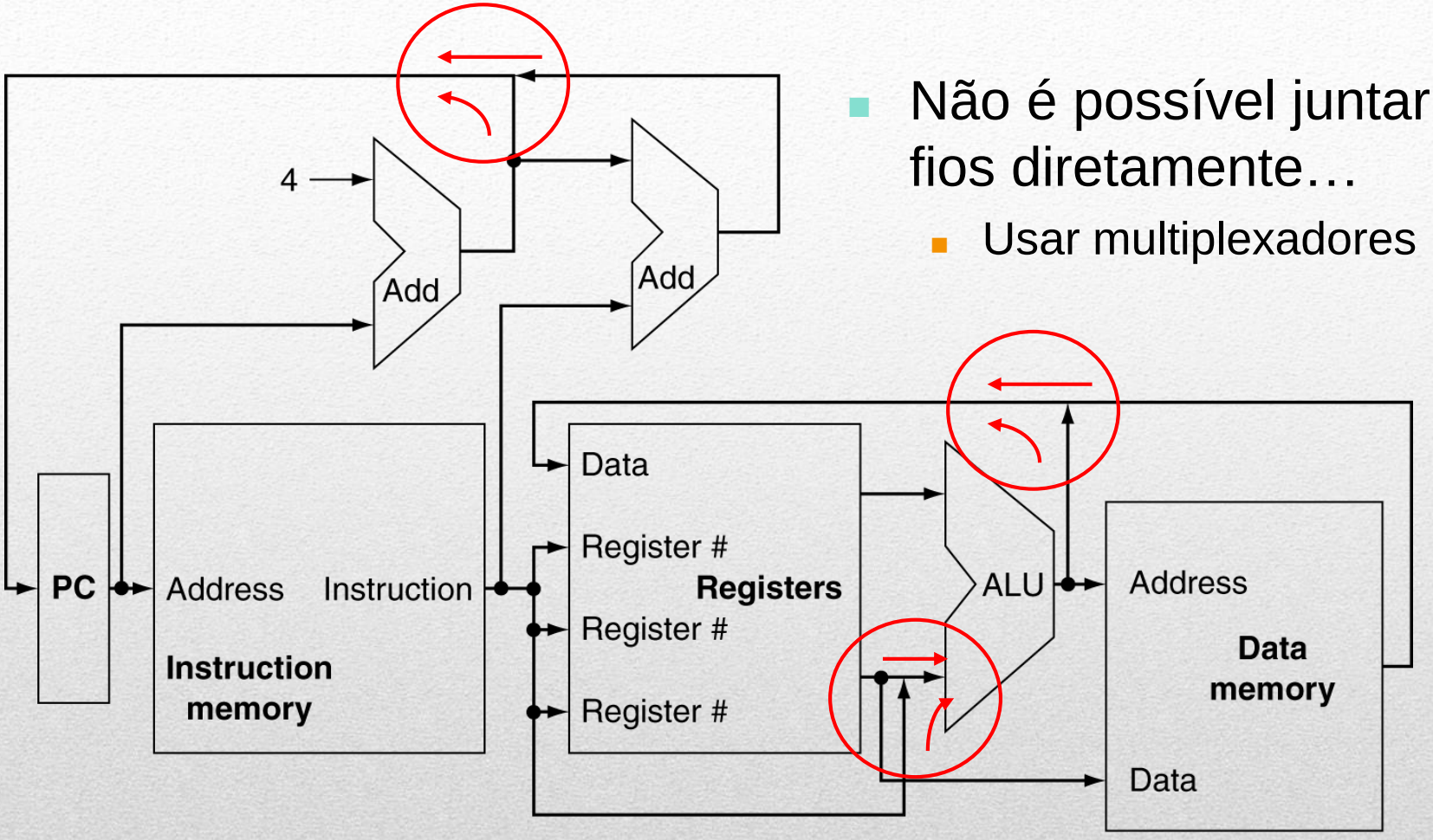
EXECUÇÃO DA INSTRUÇÃO - MIPS

1. PC → memória de instrução, buscar instrução;
2. Número dos registradores → banco de registradores, ler registradores
3. Depende da classe da instrução:
   - Usa a ULA para calcular:
     - Resultado das aritméticas;
     - Endereço da memória para load/store;
     - Endereço alvo de um desvio;
   - Acessar memória de dados para load/store;
   - PC <- endereço alvo ou PC + 4.

Execução da instrução

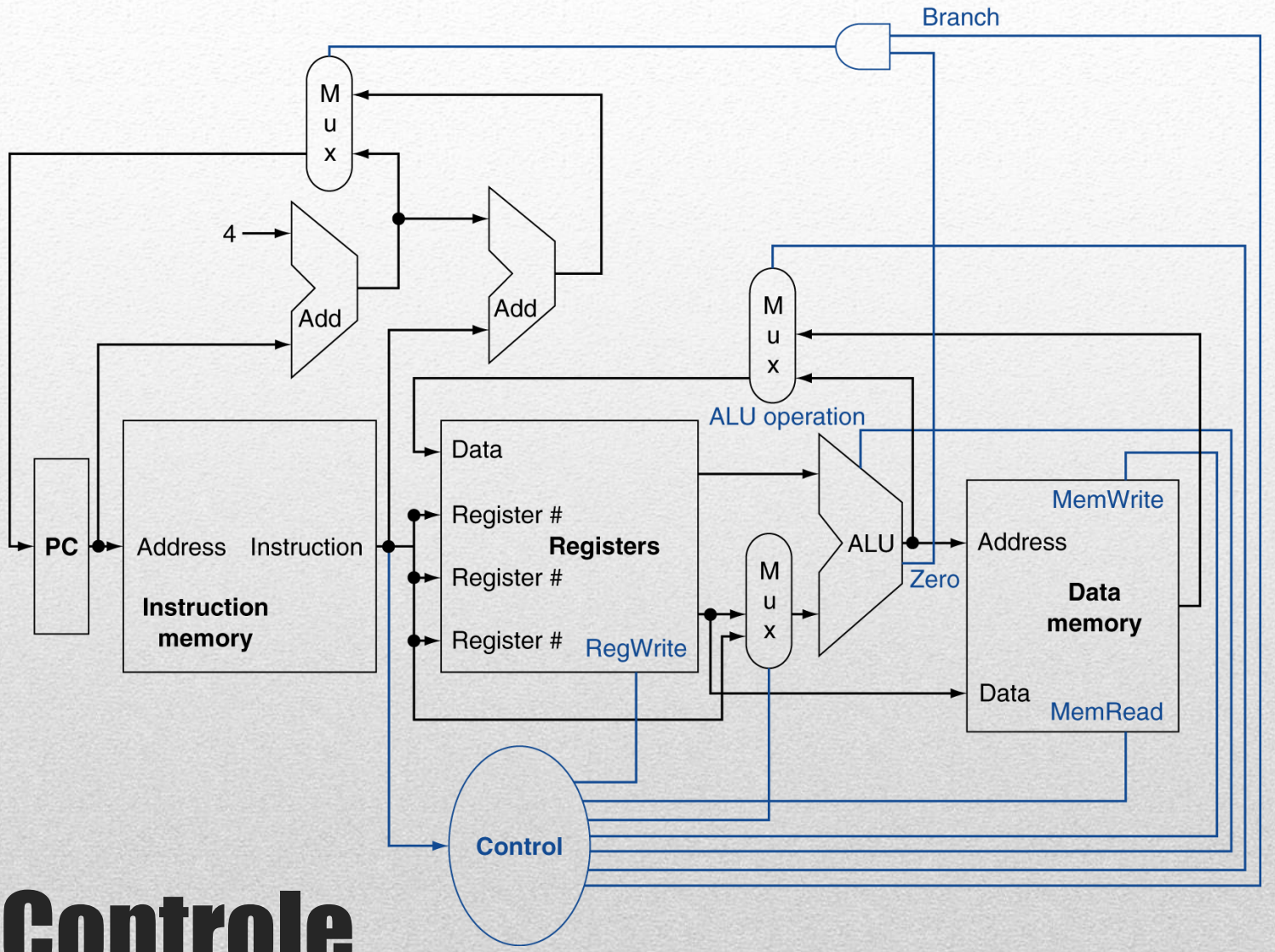


Visão geral da CPU



- Não é possível juntar fios diretamente…
  - Usar multiplexadores

Multiplexadores

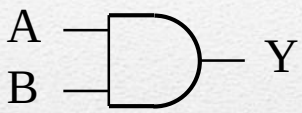

Branch
M
u
x
4
Add
Add
M
u
x
ALU operation
PC
Address
Instruction
Instruction memory
Data
Register #
Registers
Register #
Register #
RegWrite
M
u
x
ALU
Zero
MemWrite
Address
Data memory
Data
MemRead
Control

Controle

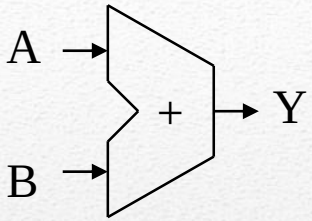
- Informação codificada em binário:
  - Baixa tensão = 0, Alta tensão = 1;
  - Um fio por bit;
  - Dados de vários bits codificados com conjuntos (barramentos) de vários fios;
- Elementos combinacionais:
  - Opera sobre os dados;
  - Saída é função da entrada;
- Elementos de estado (sequêncial):
  - Armazena informações.

Projeto de circuito digital

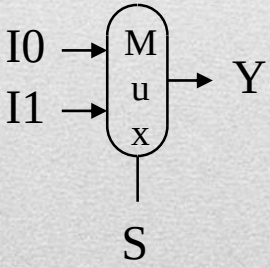
- AND-gate:
  - Y = A & B



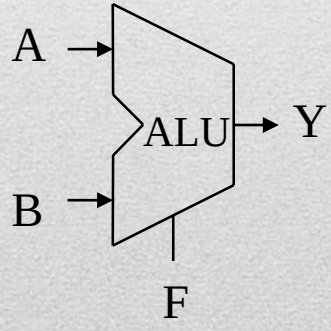
- Somador:
  - Y = A + B



- Multiplexador:
  - Y = S ? I1 : I0

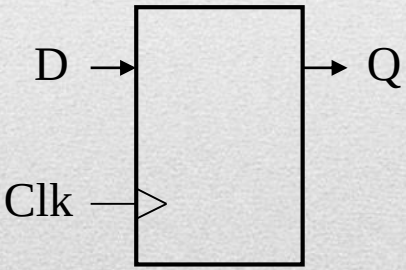


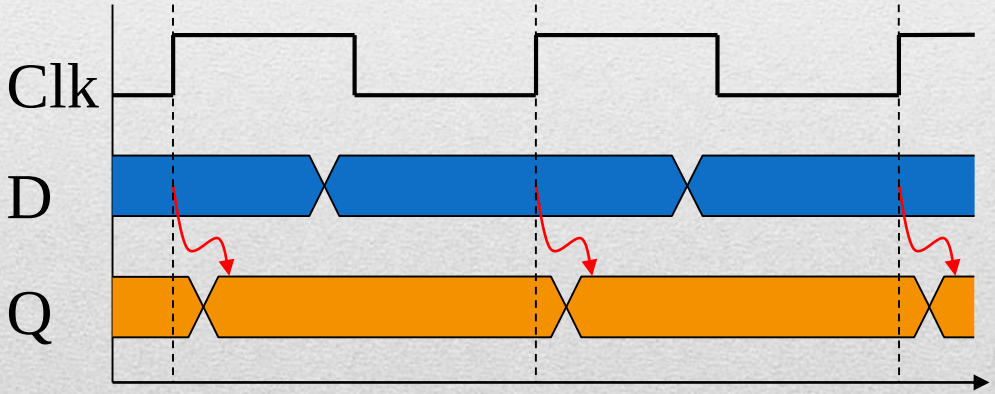
- Unidade Aritmética/Lógica:
  - Y = F(A, B)



Elementos combinacionais

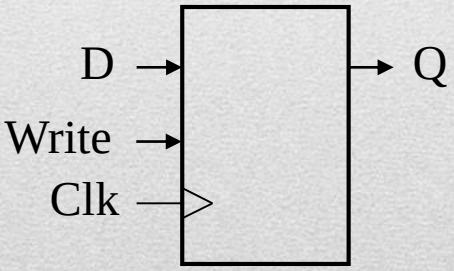
- Registrador: armazena dados em um circuito:
  - Usa o sinal de clock para determinar quando atualizar o valor armazenado;
  - Trigado na borda: atualiza quando o clock muda de 0 para 1.

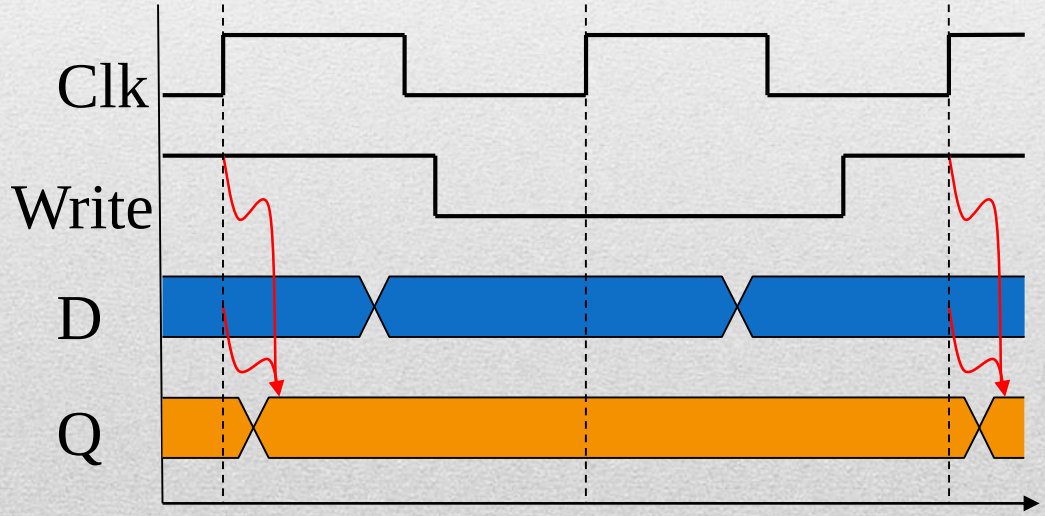




Elementos sequenciais

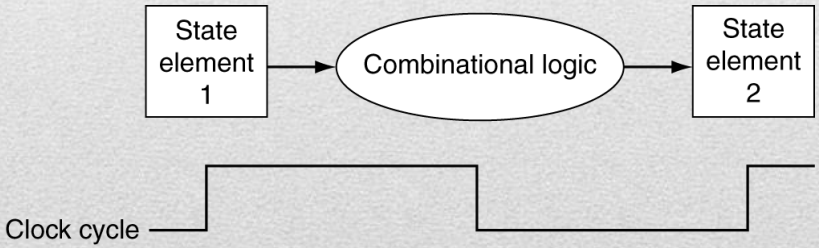
- Registrador com sinal para controle de escrita (sinal de Write):
  - Só atualiza na borda do clock quando o sinal para controle de escrita é 1;
  - Utilizado quando o dado armazenado será necessário no futuro.

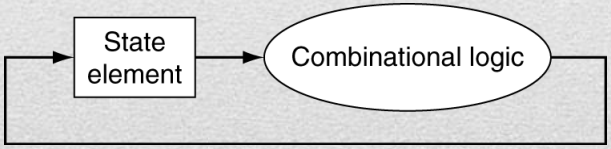




Elementos sequenciais

- Lógica combinacional transforma os dados durante o ciclo de clock:
  - Entre duas bordas do clock;
  - Entradas vem de elementos de estados, saída vai para elementos de estado;
  - Circuito com maior atraso determina o período do clock.





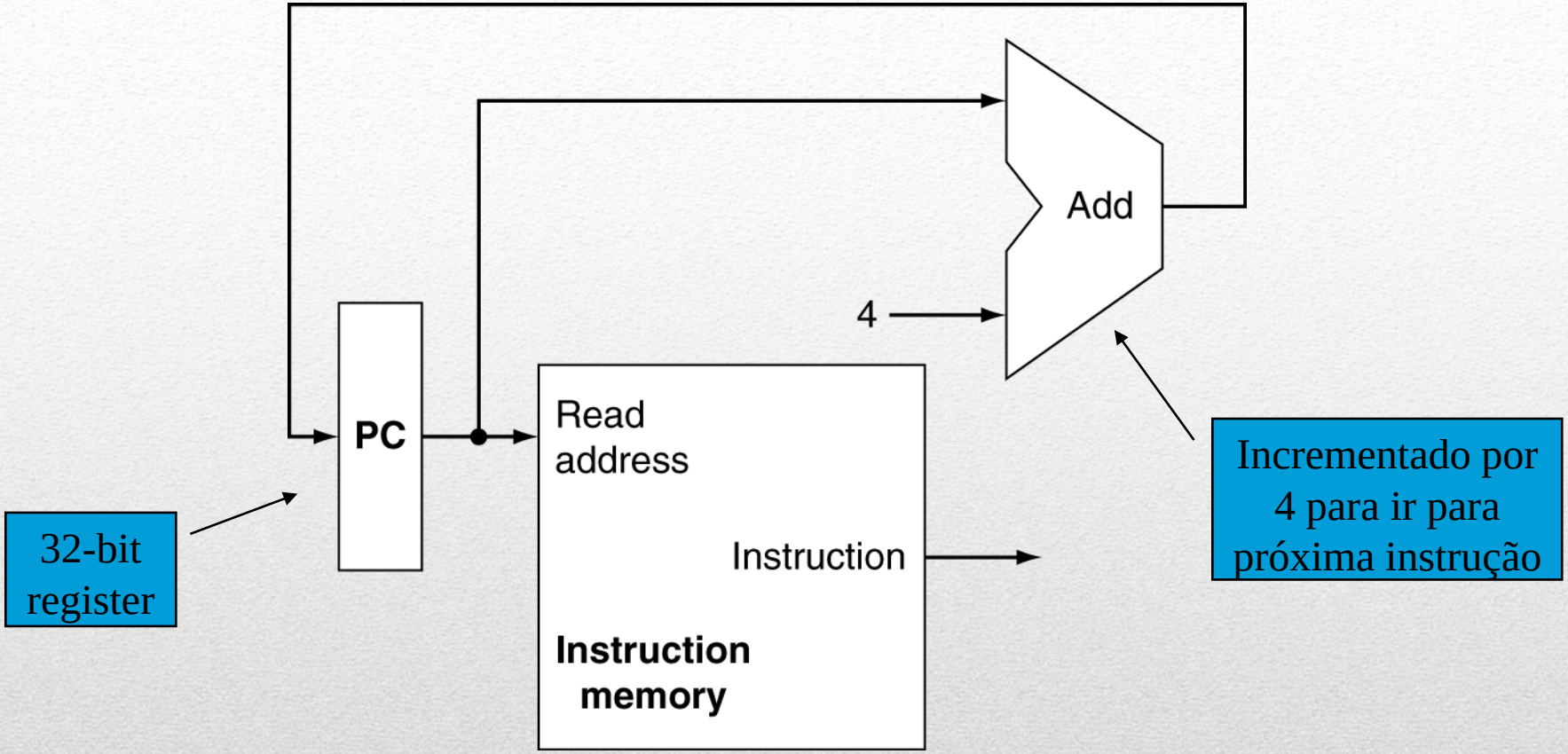
Metodologia de clock

Construindo uma Unidade de Processamento (datapath)

- Unidade de Processamento (Datapath):
  - Elementos que processam dados e endereços na CPU:
    - Registradores, ALUs, multiplexadores, memórias, etc...
- Nós iremos construir uma Unidade de Processamentos para o MIPS incrementalmente:
  - Refinando a visão geral...

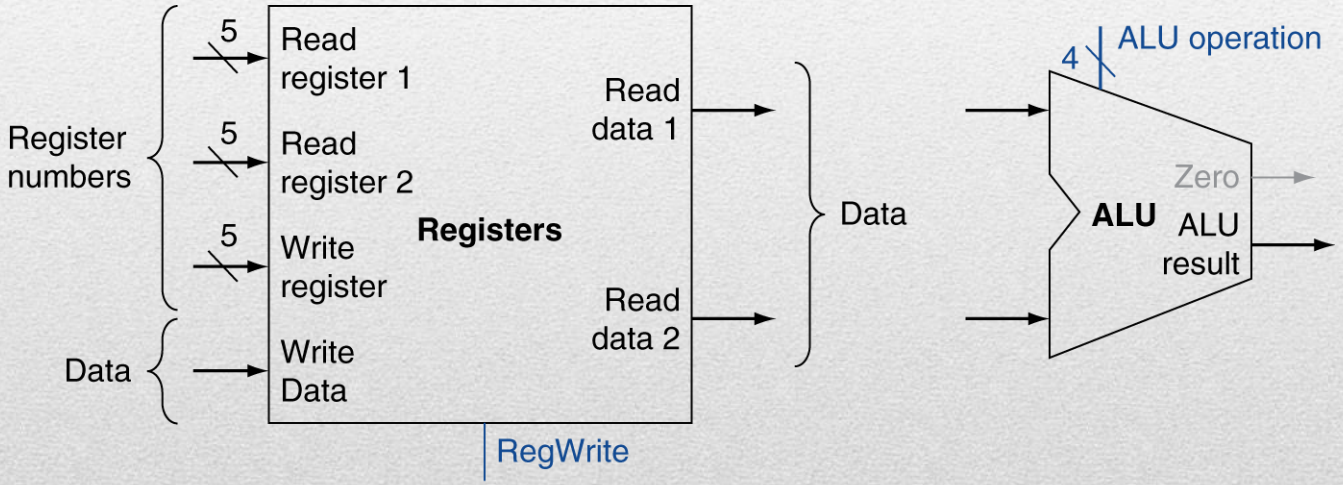
1. PC → memória de instrução, buscar instrução;
2. Número dos registradores → banco de registradores, ler registradores
3. Depende da classe da instrução:
   - Usa a ULA para calcular:
     - Resultado das aritméticas;
     - Endereço da memória para load/store;
     - Endereço alvo de um desvio;
   - Acessar memória de dados para load/store;
   - PC <- endereço alvo ou PC + 4.

Relembrando: Execução da instrução



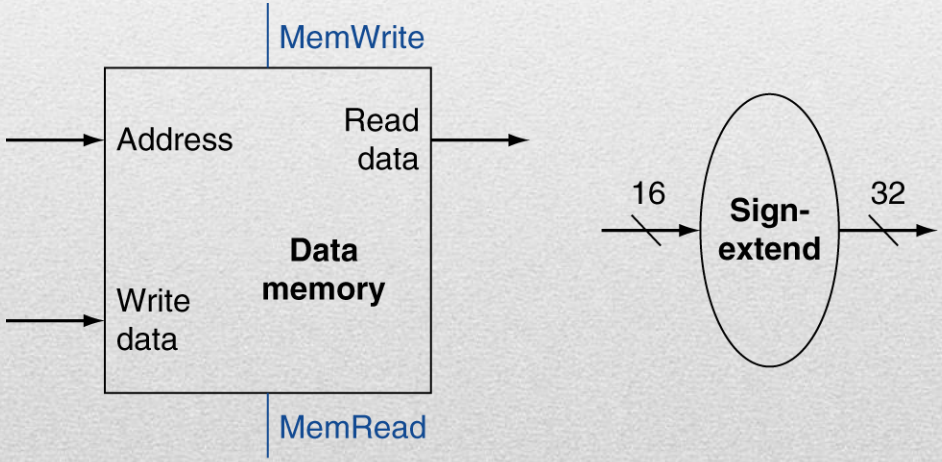
Buscar Instrução (Instruction Fetch – IF)

1. Lê dois operandos dos registradores;
2. Executa uma operação lógica/aritmética;
3. Escreve o resultado no registrador.



Instruções do formato R

1. Lê operando do registrador;
2. Calcula o endereço usando deslocamento (constante) de 16-bits:
   - Usa ULA, mas com deslocamento com sinal estendido;
3. Load: Lê a memória e escreve no registrador;
4. Store: Lê do registrador e escreve na memória.

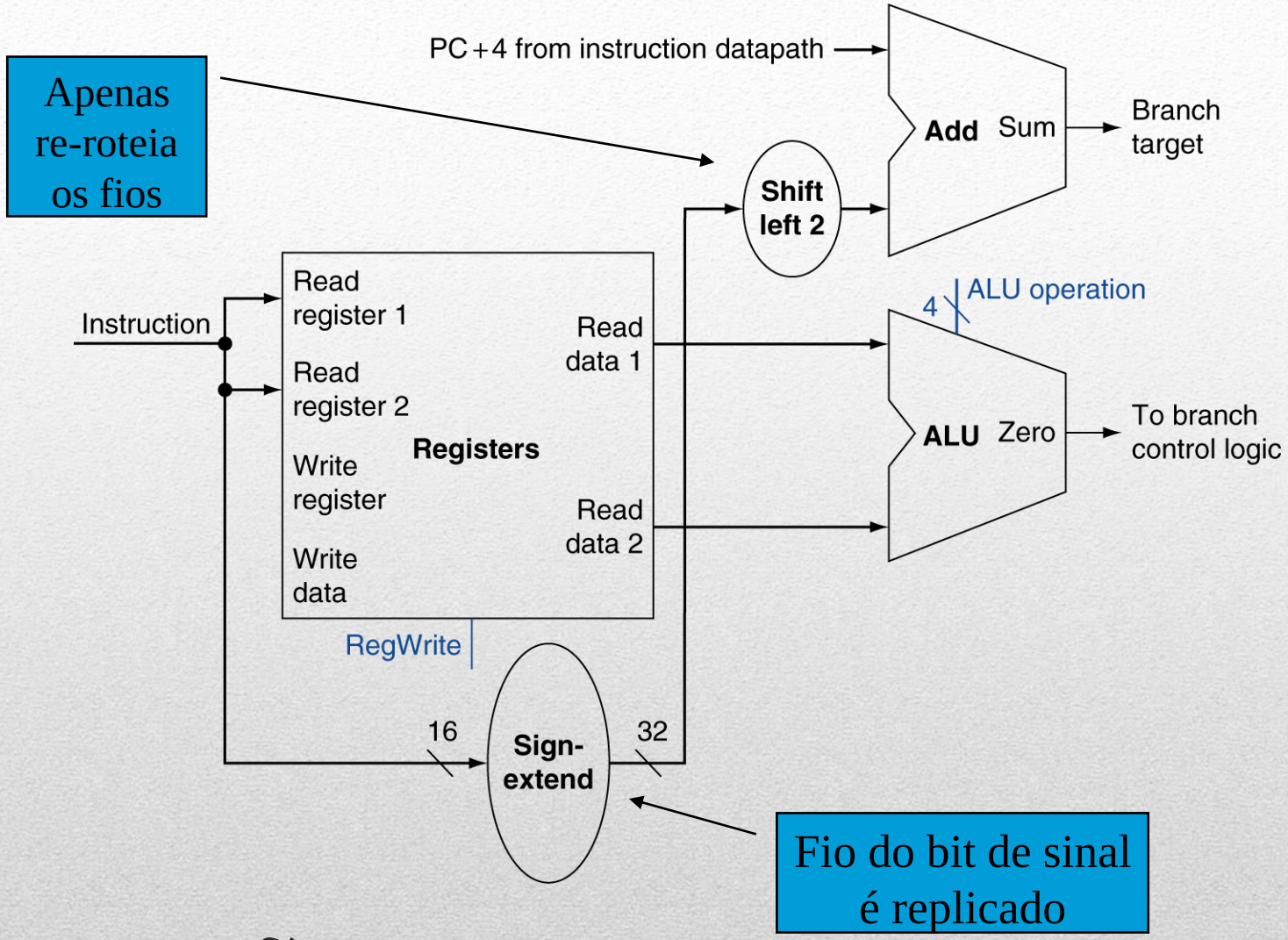


a. Data memory unit b. Sign extension unit

Instruções de Load/Store

1. Lê os operandos dos registradores;
2. Compara os operandos:
   - Usa a ULA, subtrai e verifica a saída Zero;
3. Calcula o endereço alvo:
   1. Estende o sinal do deslocamento;
   2. Shift para esquerda de 2 posições (deslocamento por palavra);
   3. Soma com PC+4:
      - Já calculado pela busca de instrução.

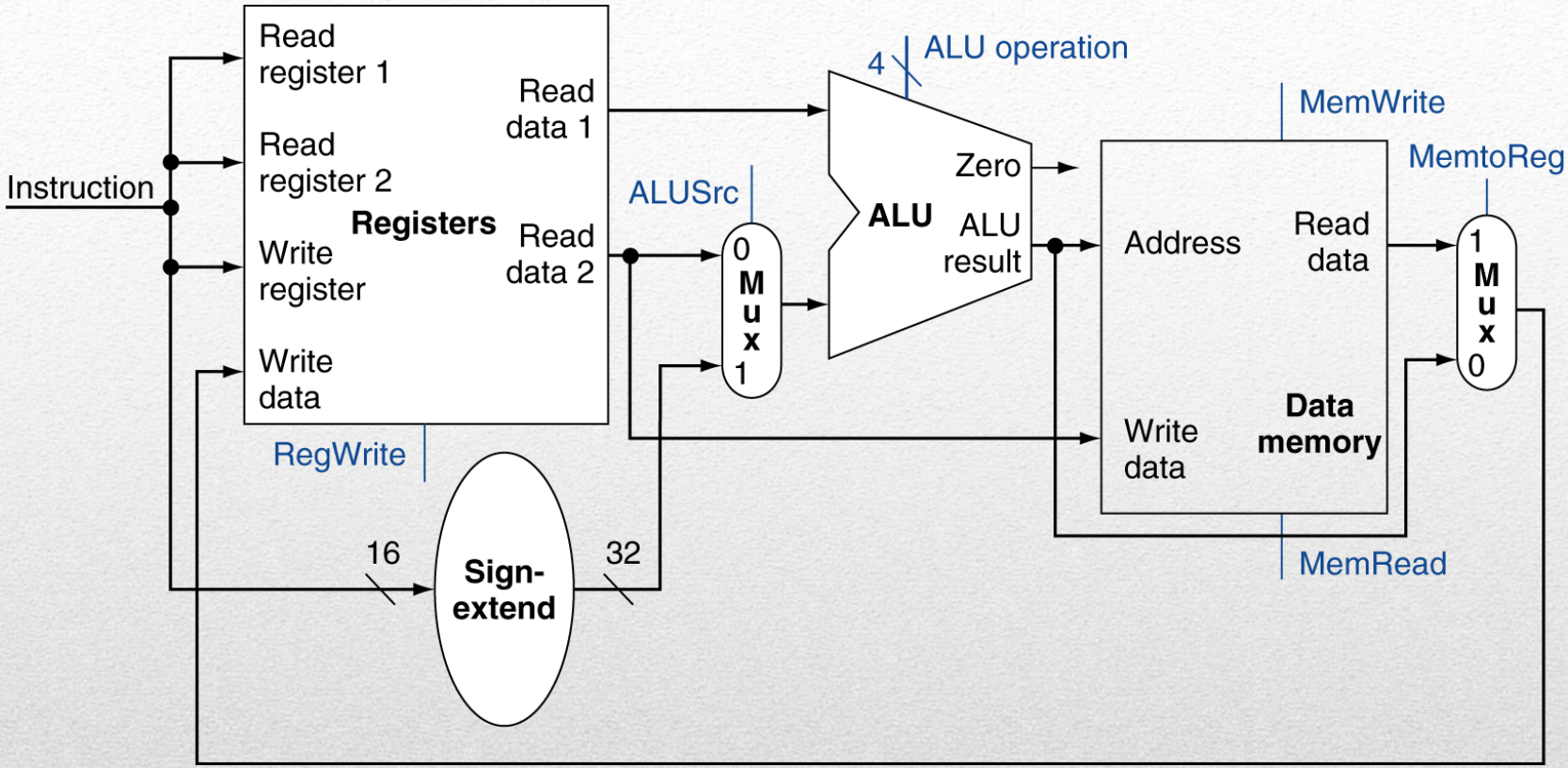
Instruções de desvio



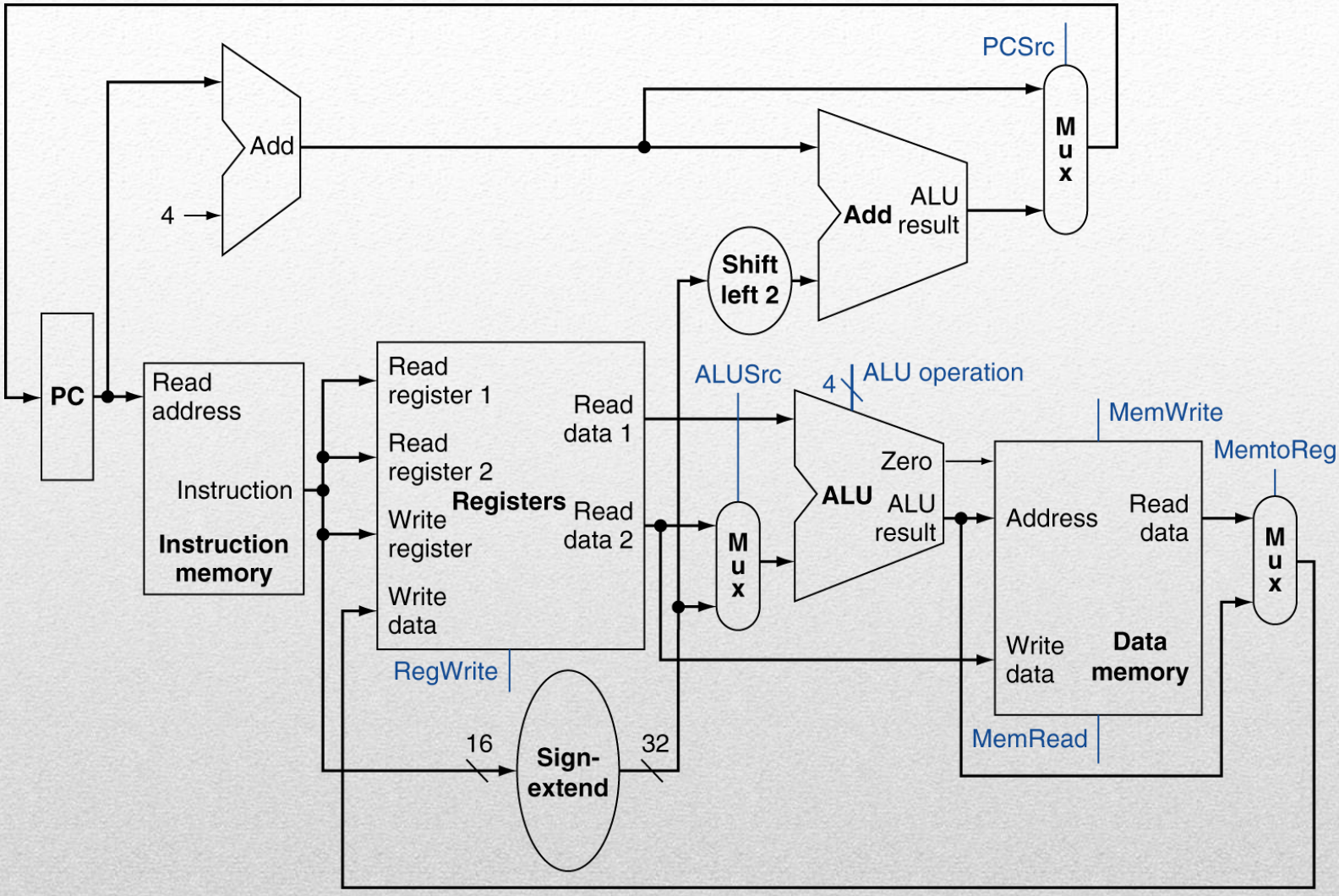
Instruções de desvio

- Primeiro: Unidade de Processamento faz uma instrução em um ciclo de clock:
  - Cada elemento da unidade pode fazer apenas uma função por vez;
  - Desta forma, precisamos separar as memórias de instruções e dados;
- Use multiplexadores onde fonte de dados alternativas são utilizadas para diferentes instruções,

Compondo os elementos da Unidade de Processamento



Unidade de Processamento: Formato R/Loda/Store



Unidade de Processamento: Completo

- ULA usada para:
  - Load/Store: F = somar;
  - Branch: F = subtrair;
  - Formato R: F depende do campo FUNCT

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set-on-less-than
1100	NOR

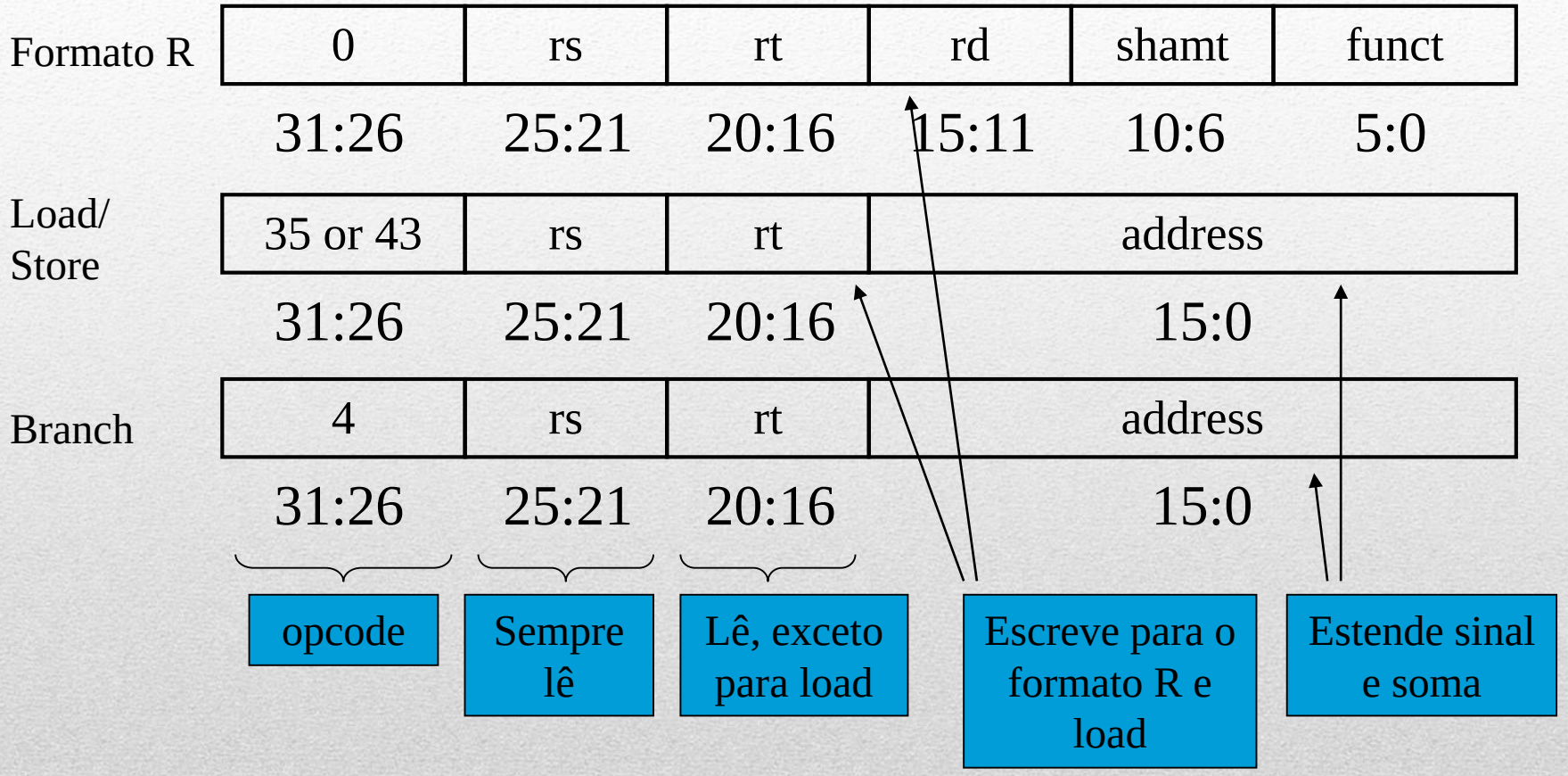
Controle de ULA

- Assuma ALUOp com 2 bits derivado do opcode:
  - Lógica combinacional determina o controle da ULA

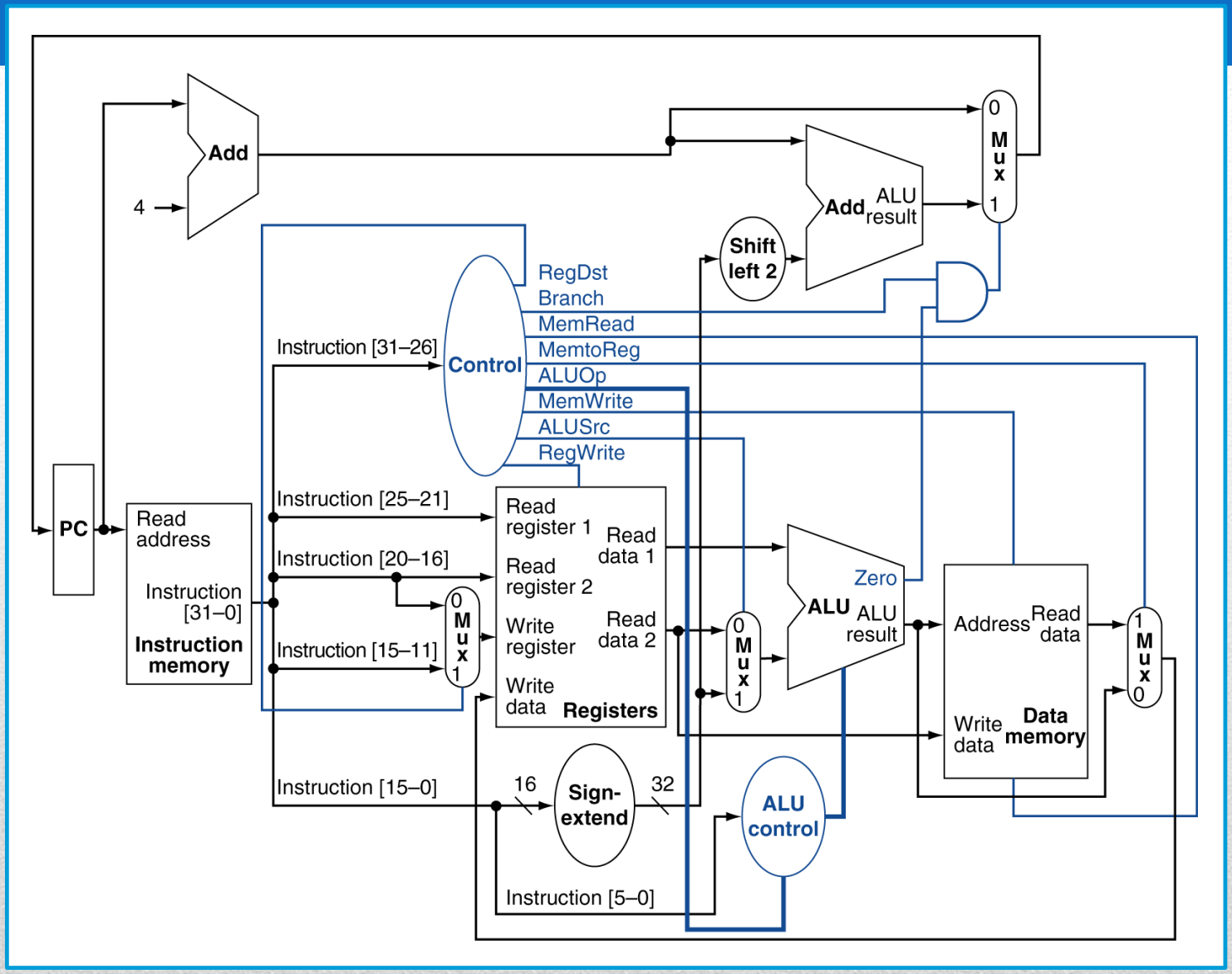
opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111

Controle da ULA

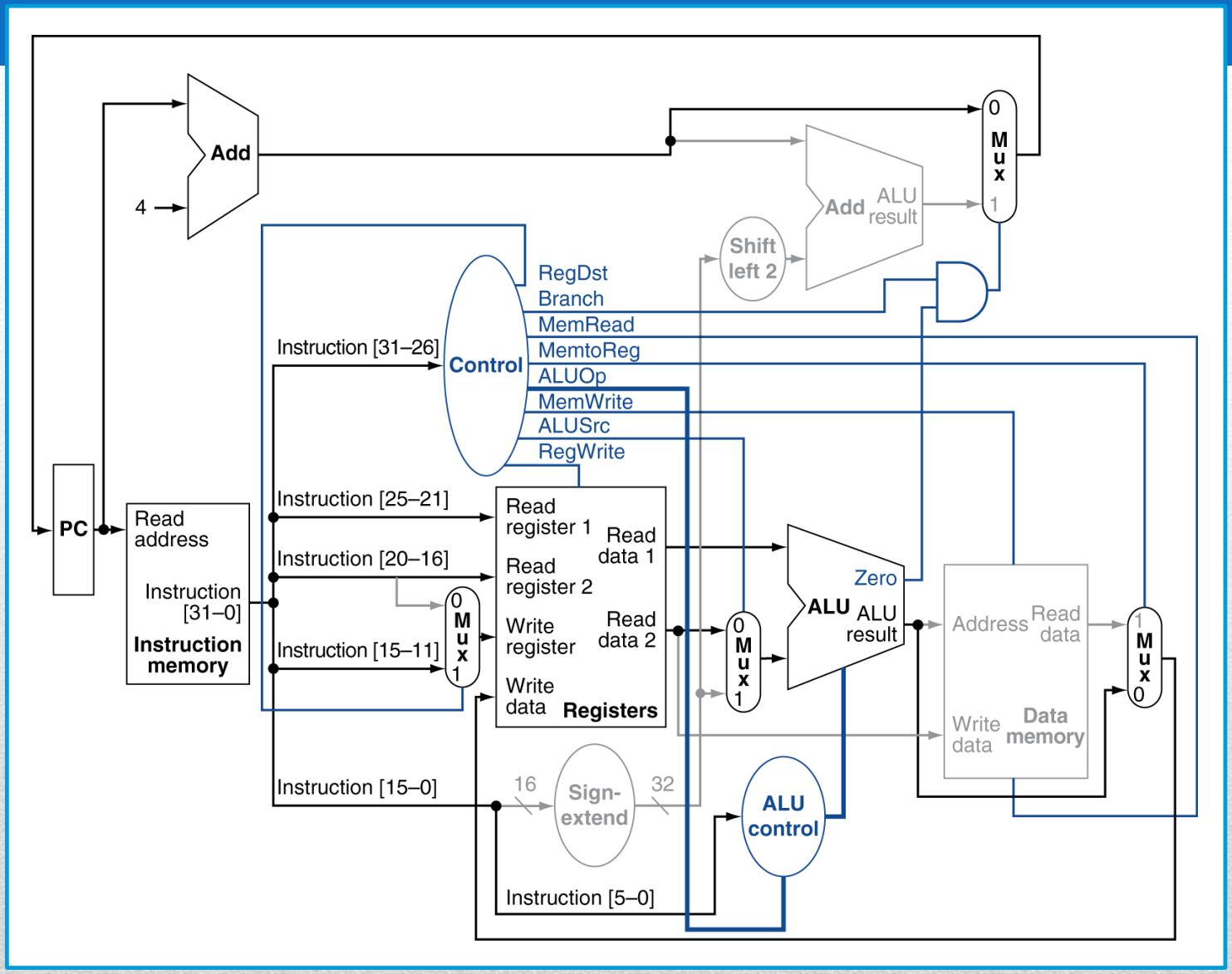
- Sinais de controle são derivados da instrução:



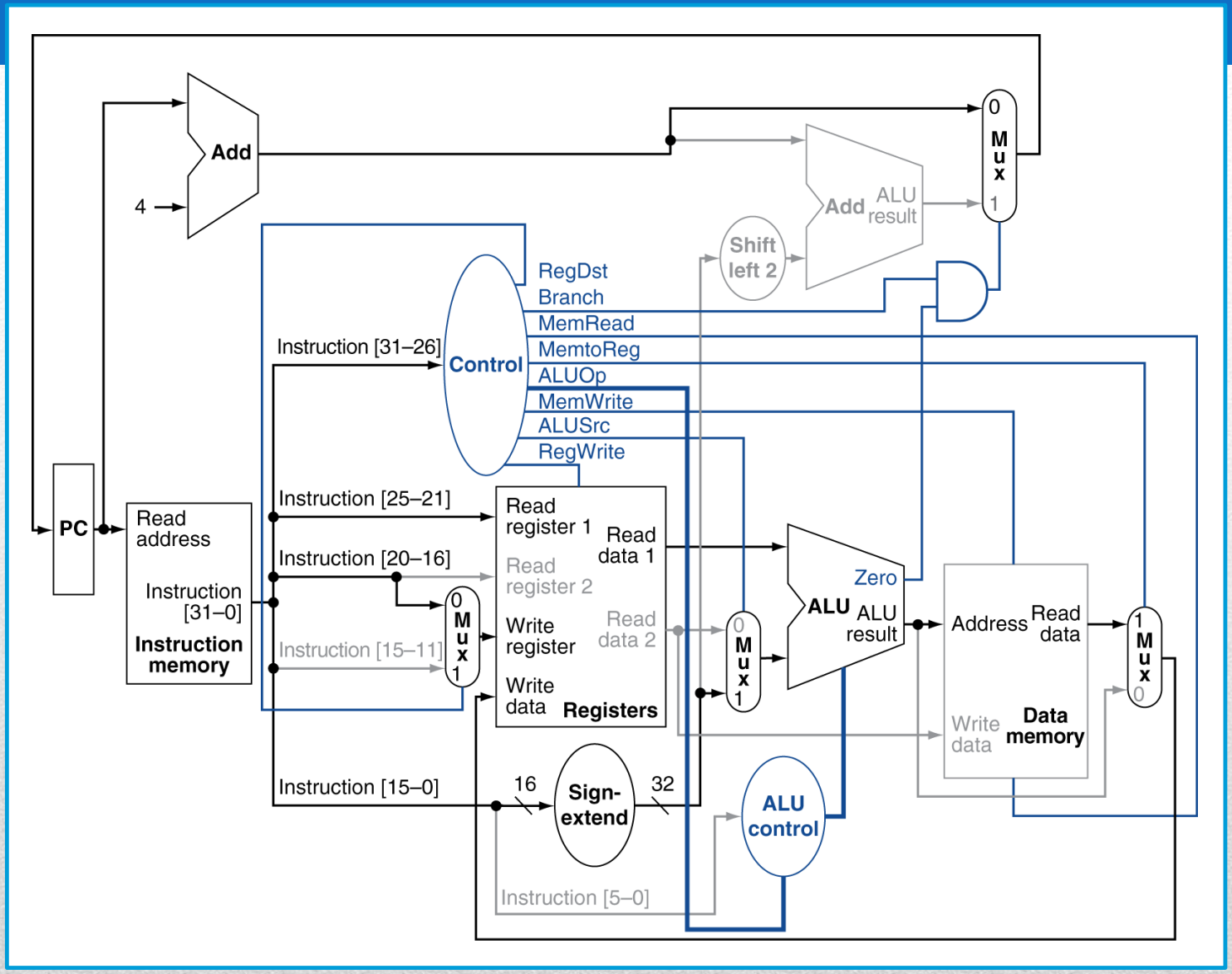
Unidade de Controle - Decodificação



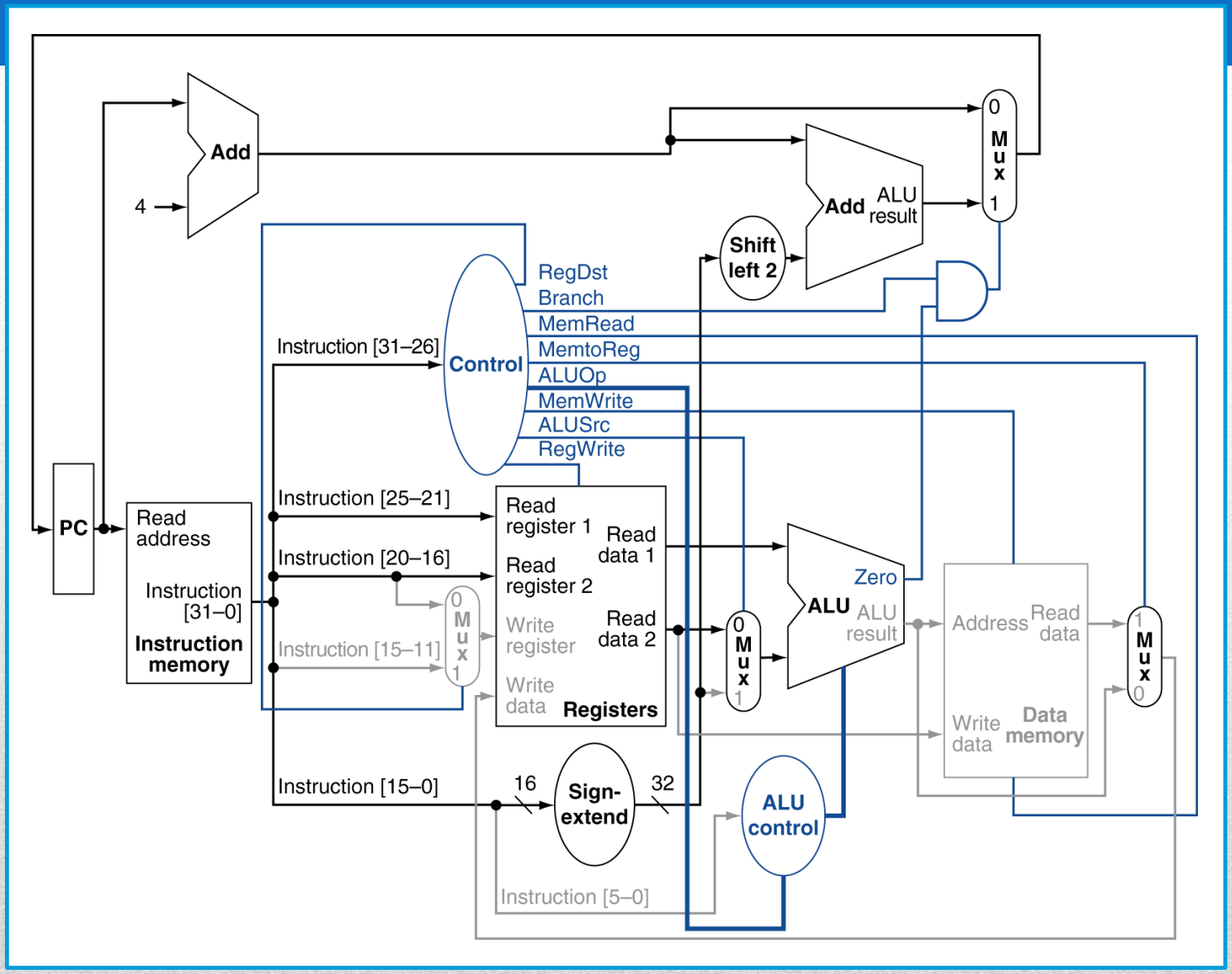
Unid. Process. + Unid. de Controle



Instrução do formato R



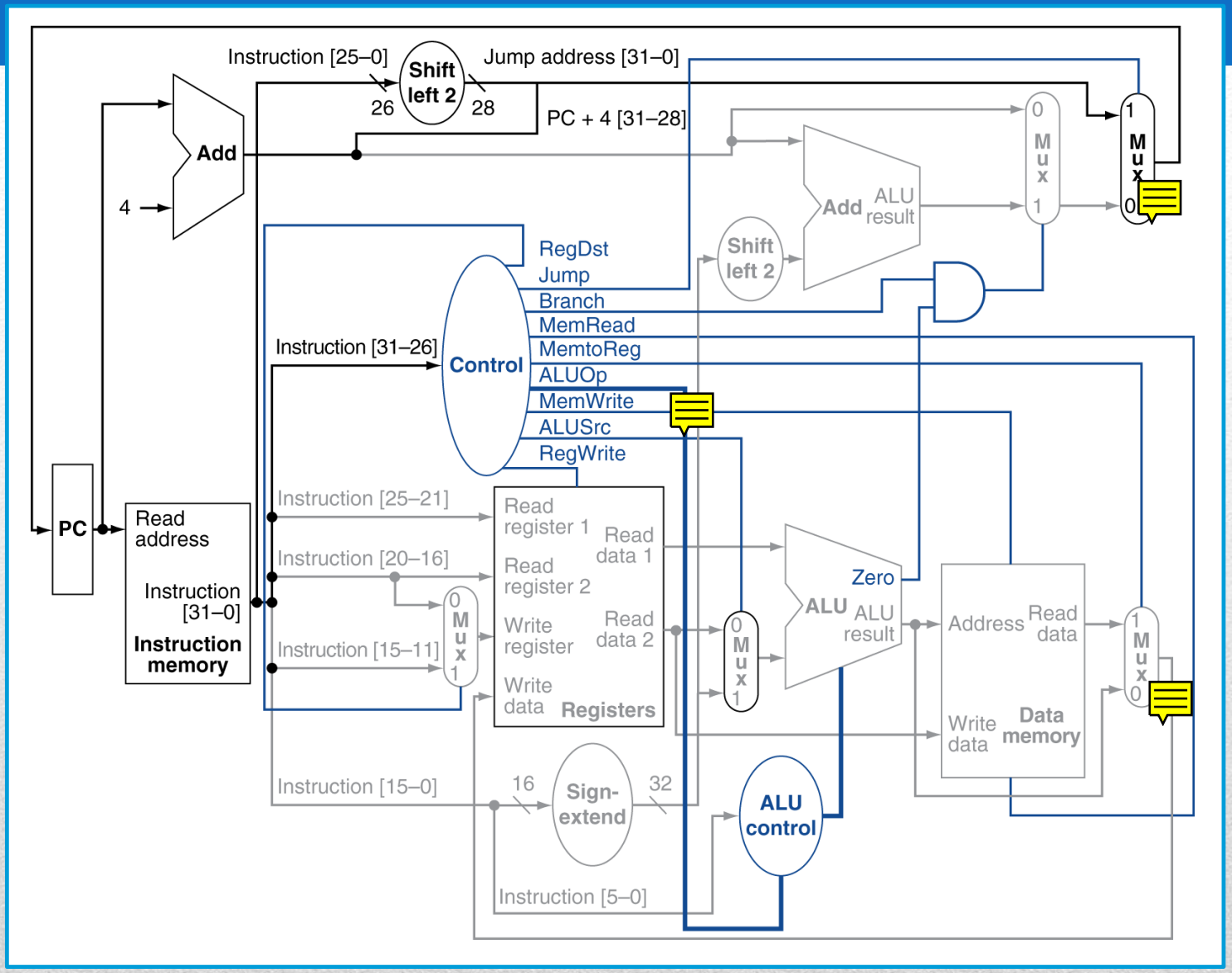
Instrução de Load



Instrução de Beq

- Jump usa o endereço da palavra;
- Atualiza o PC com a concatenação de:
  - 4 bits mais significativos do PC anterior;
  - 26 bits do endereço do jump
  - 00
- Precisa de um sinal extra do controle decodificado do opcode.

Implementado os Jumps



Unid. Process. + Jumps

- Circuito mais longo determina o período de clock:
  - Caminho crítico: instrução de load;
  - Memória de instruções -> banco de registradores -> ULA -> memória de dados -> banco de registradores;
- Não é viável para variar período para diferentes instruções;
- Viola o princípio de projeto:
  - Faça o caso comum mais rápido;
- Nós iremos aumentar o desempenho através do pipeline.

Problemas de desempenho

Arquitetura e Organização de Computadores

Capítulo 4 – O Processador