Bloco Operacional Monociclo

Luciano L. Caimi

lcaimi@uffs.edu.br

Roteiro

Introdução

1. Busca de instrução
2. Instruções aritméticas
3. Instruções de acesso à memória
4. Instruções de salto
5. Combinando instruções
6. Bloco operacional completo

O Processador: Datapath & Controle

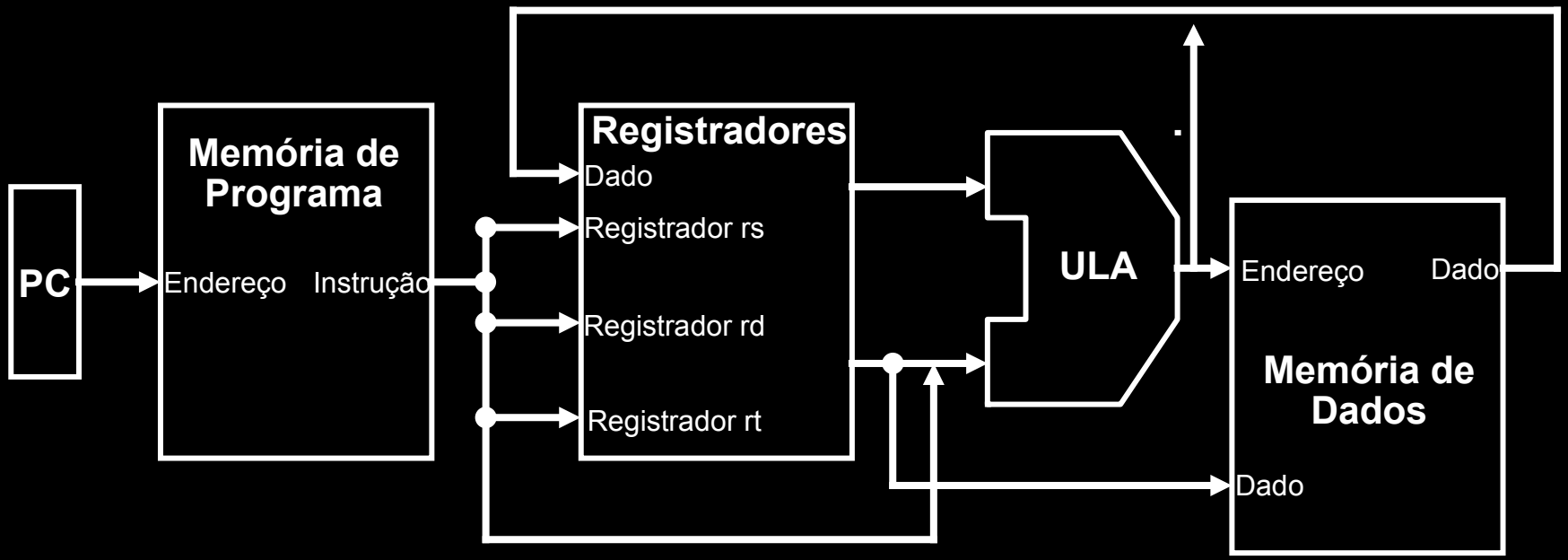
- **Simplificado contendo somente:**
  - instruções de referência a memória: `lw, sw`
  - instruções lógico-aritméticas: `add, sub, and, slt`
  - instruções de controle de fluxo: `beq, j`
- **Implementação Genérica:**
  - uso do contador de programa (PC) para endereçar instruções
  - busca de instruções da memória
  - leitura de registradores
  - uso de instruções para decidir exatamente o que fazer

Todas as instruções usam a ALU após a leitura dos registradores

Porque? Referência a memória!
Aritmética! Controle de fluxo!

Mais Detalhes de Implementação

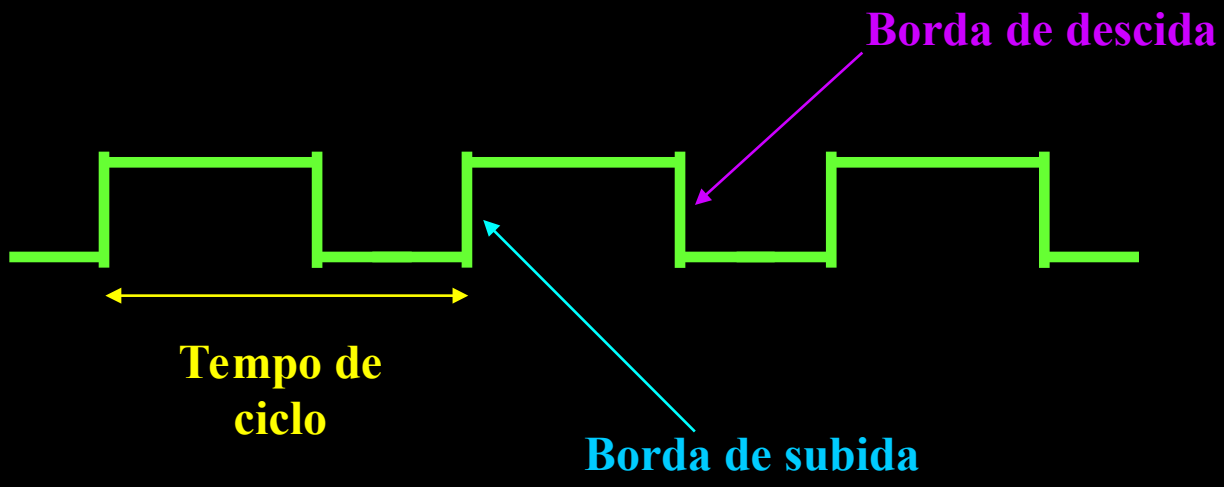
Abstração / Vista Simplificada:



- **Dois tipos de unidades funcionais:**
  - **elementos que operam com valores (combinatório)**
  - **elementos que contém estados (seqüenciais)**

Elementos de Estado

- Unclocked vs. Clocked
- Clocks usados em lógica síncrona
- Quando o elemento que contém o estado poderia ser atualizado?

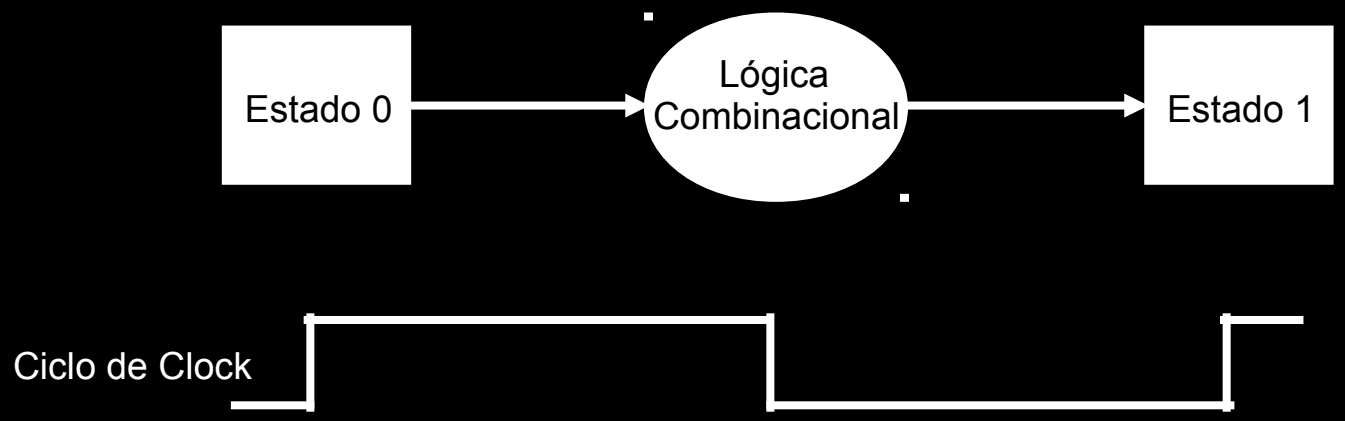


Latches e Flip-flops

- Saída é igual ao valor armazenado no elemento
  (não é necessário permissão para ler o valor)
- Latches: entradas mudam → estado muda
- Flip-flop: estados mudam somente na borda do clock
  (edge-triggered metodologia)

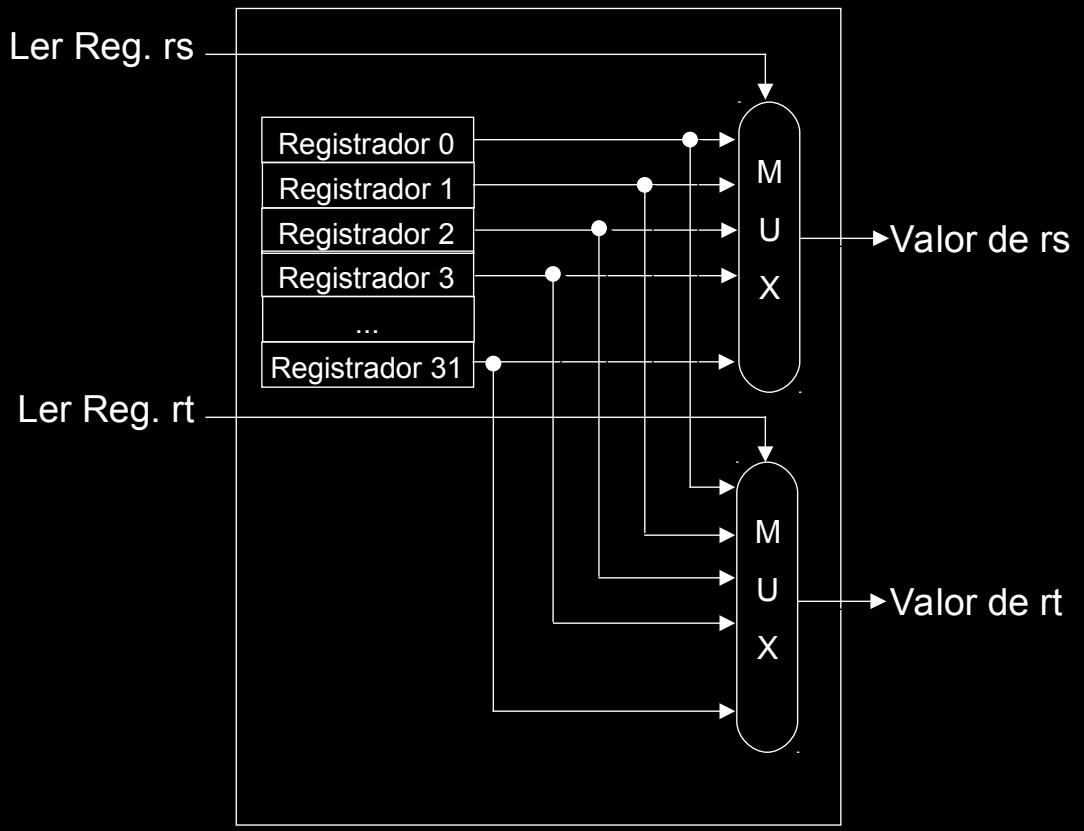
Nossa Implementação

- Metodologia edge triggered
- Execução típica:
  - Ler conteúdos dos vários elementos de estado,
  - Enviar valores através da lógica combinatória
  - Escrever resultados em um ou mais elementos de estado



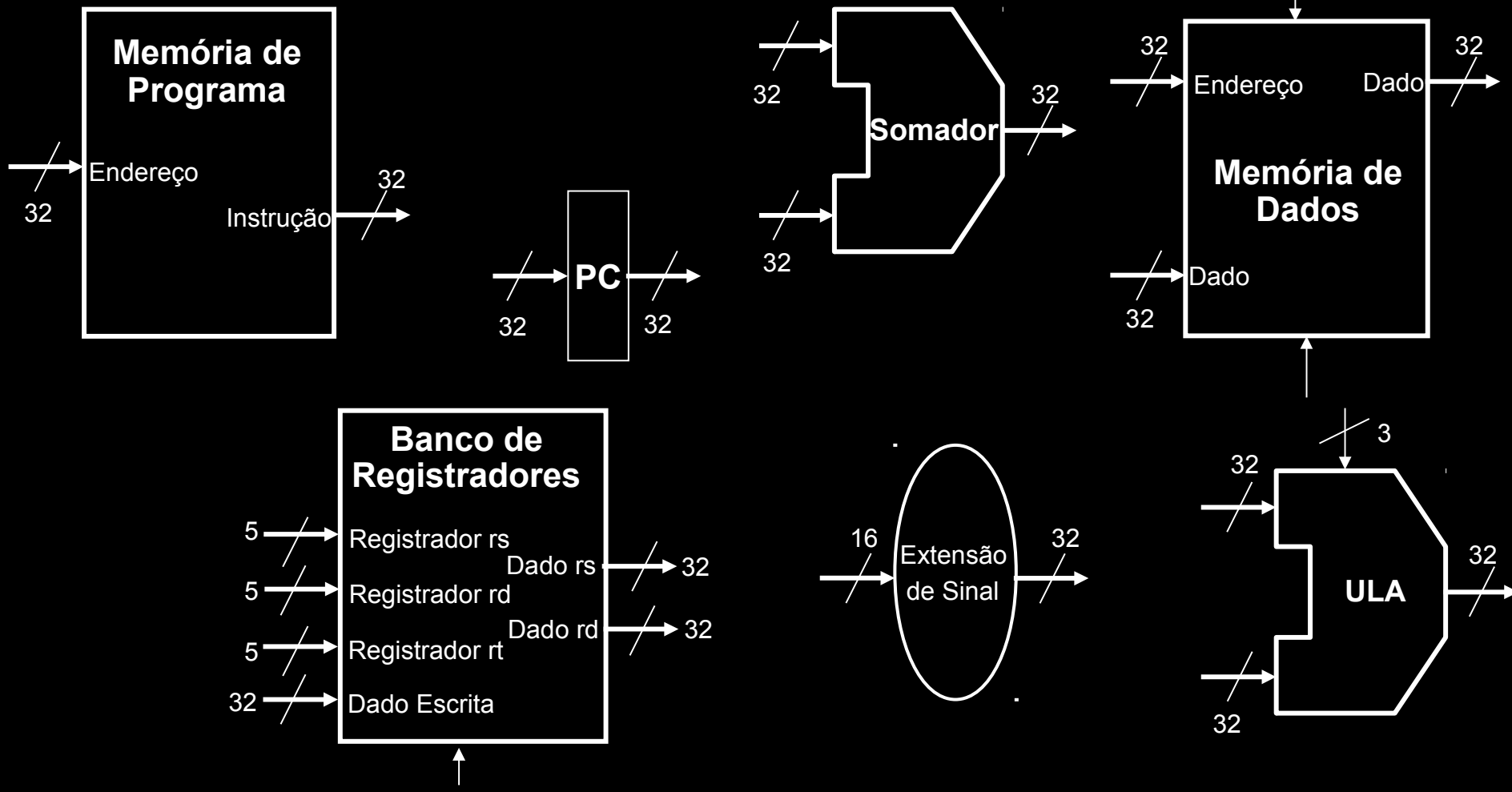
Registradores

- Integração usando flip-flops D



Implementação Simples

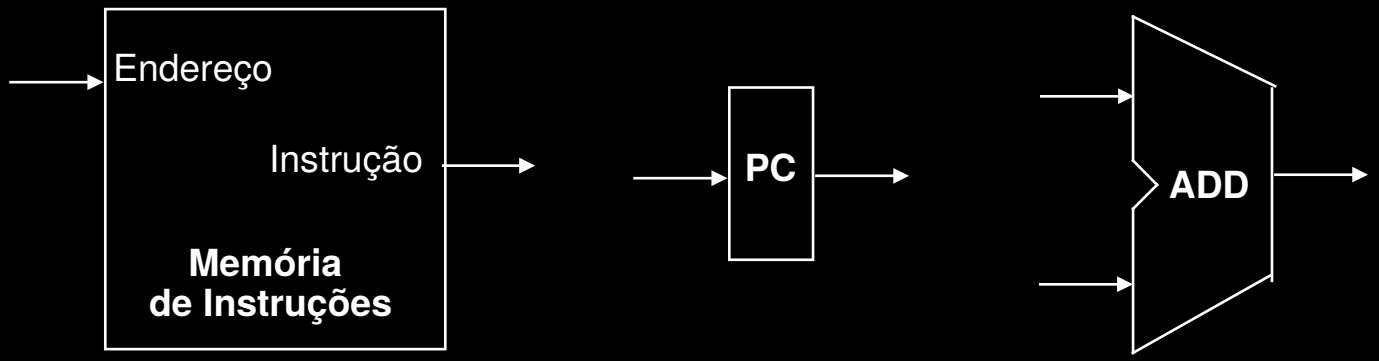
- unidades funcionais necessárias



1. Busca de instrução

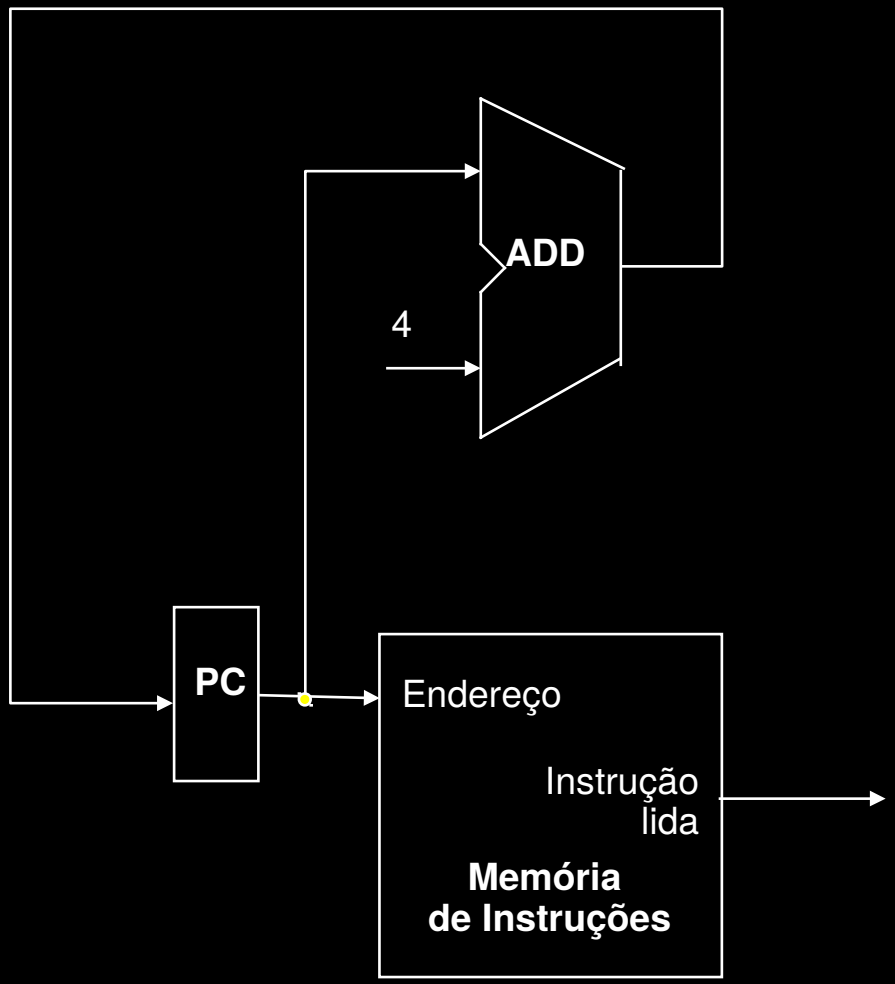
Três elementos são necessários para executar uma busca de instrução:

- **a memória onde estão armazenadas as instruções**
- **o contador de programa (PC) para armazenar o endereço da instrução**
- **um somador é necessário para calcular o endereço da próxima instrução**



Busca de instrução

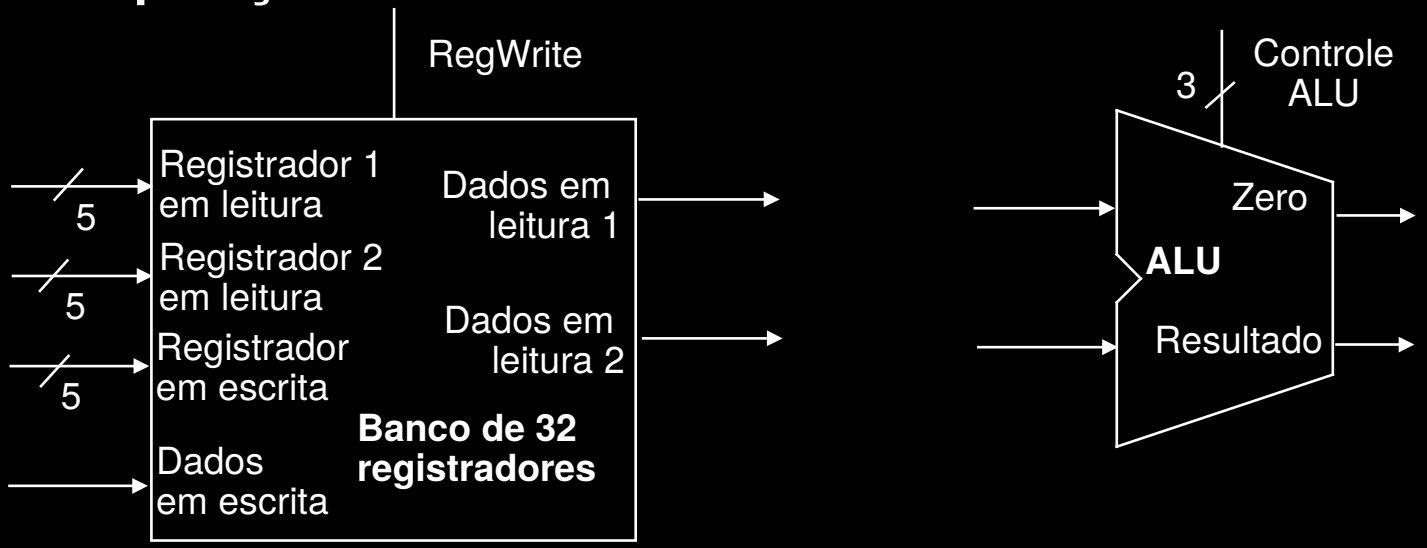
- O contador de programa contém o endereço da instrução em execução
- O endereço da próxima instrução é obtido pela soma de 4 posições ao contador de programa
- A instrução lida é usada por outras porções da parte operativa



2. Instruções aritméticas

Dois elementos são necessários para a execução de operações aritméticas:

- **Um banco de registradores para armazenar os operandos e o resultado das operações**
- **Uma Unidade Lógica/Aritmética (ALU) que será utilizada para realizar as operações**



op	rs	rt	rd	shamt	funct

2. Instruções aritméticas

O hardware para a execução de instruções aritméticas e lógicas (formato R) segue o esquema abaixo:

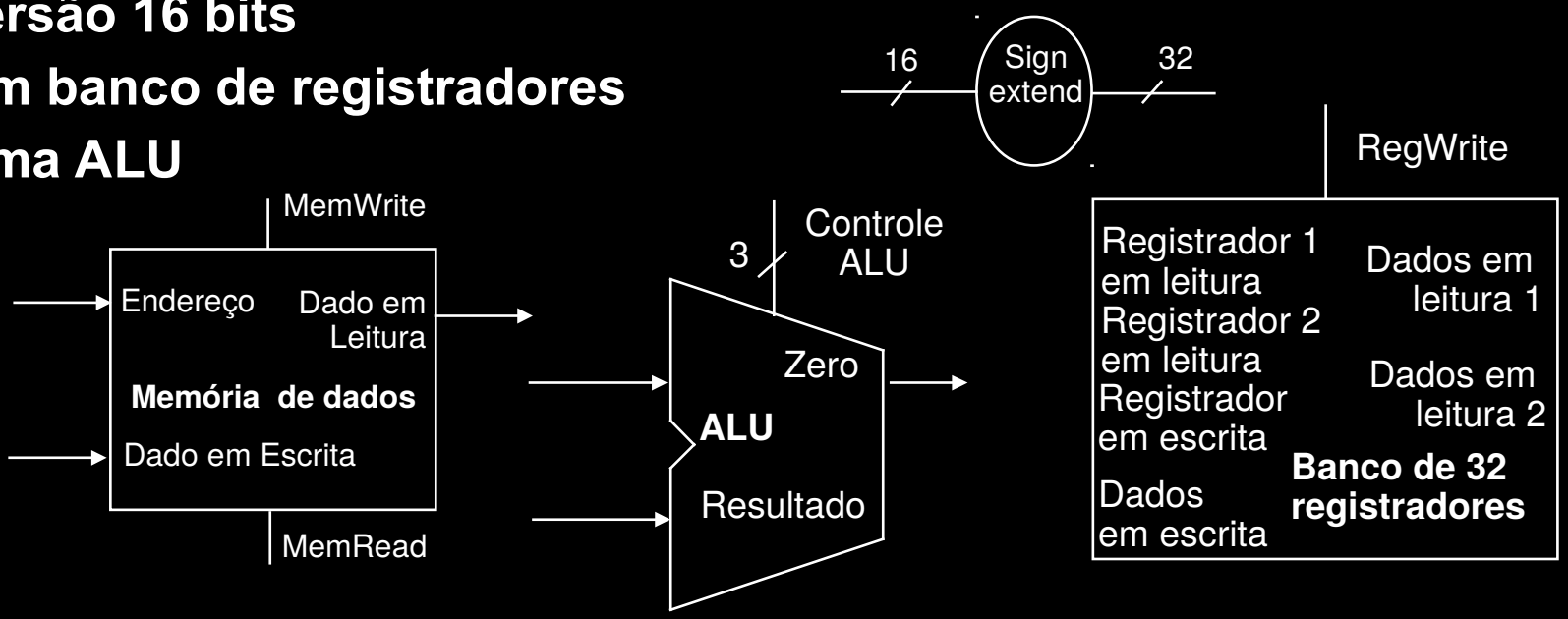
- **A instrução (fornecida pelo hardware de busca de instruções) contém o endereço de três registradores**
- **Dois destes registradores são lidos e passados para a ALU realizar a operação**
- **O resultado é armazenado em um terceiro registrador**
- **O controle da ALU determina a operação que será realizada (a partir do código da instrução)**

op	rs	rt	rd	shamt	funct

3. Instruções de acesso à memória

Para executar instruções de acesso à memória do tipo load e store são necessários:

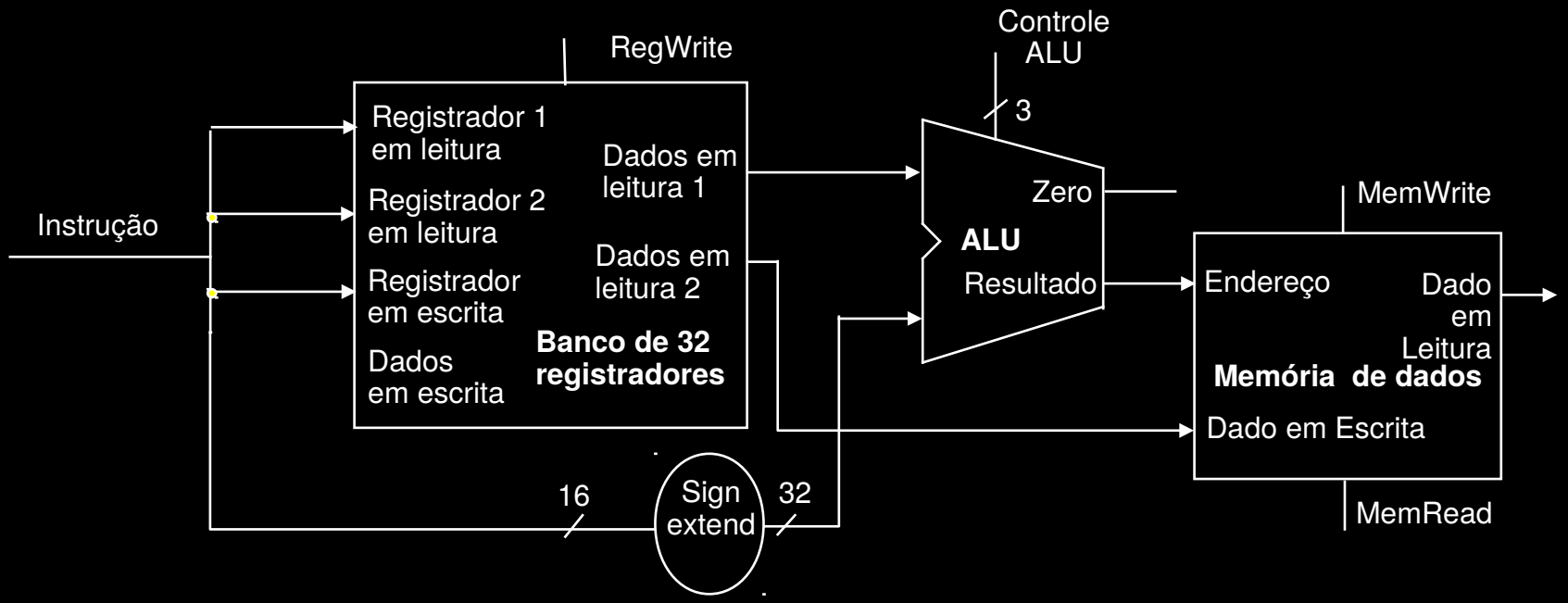
- **Uma memória de dados**
- **Um módulo de extensão de sinal (sign extend) para calcular números negativos e positivos em 32 bits a partir de sua versão 16 bits**
- **Um banco de registradores**
- **Uma ALU**



op	rs	rt	offset

Escrita em memória

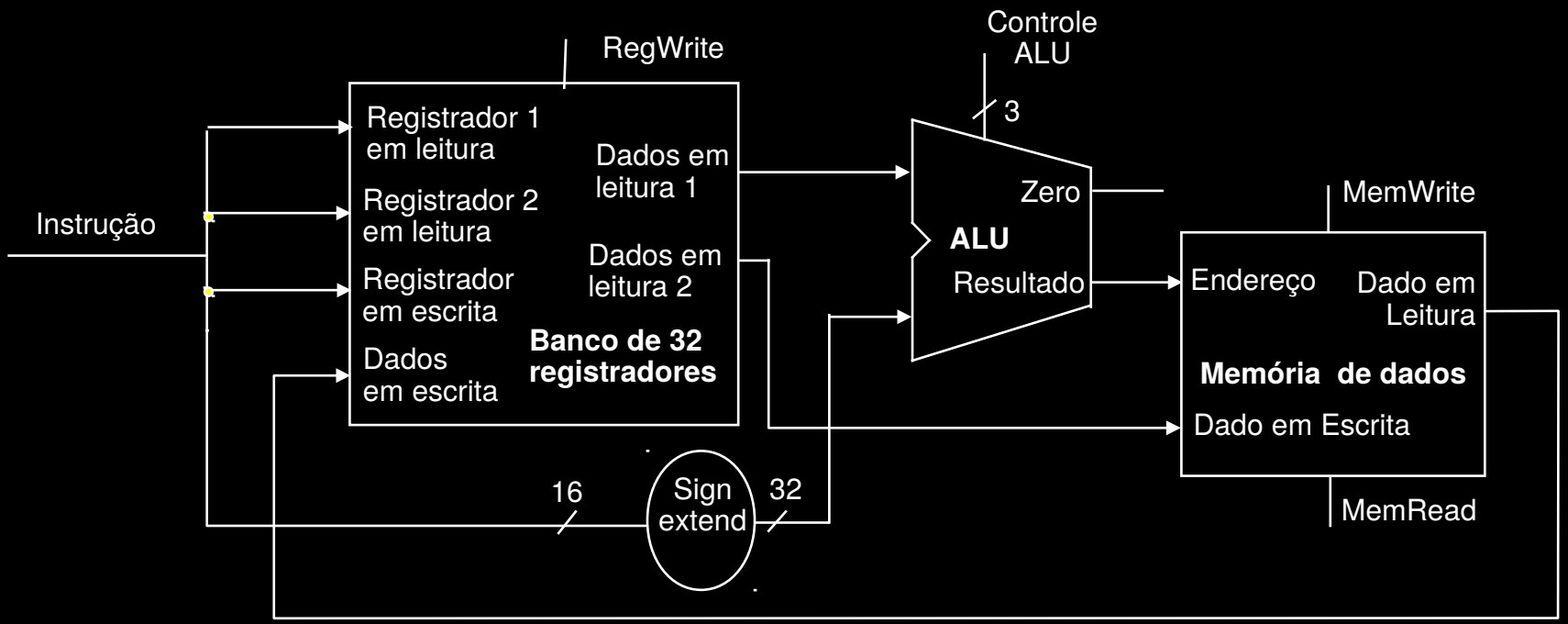
- O endereço de escrita é obtido pela soma de um registrador de base (registrador 1) com um deslocamento
- O registrador 2 é escrito na memória



op	rs	rt	offset

Leitura da memória

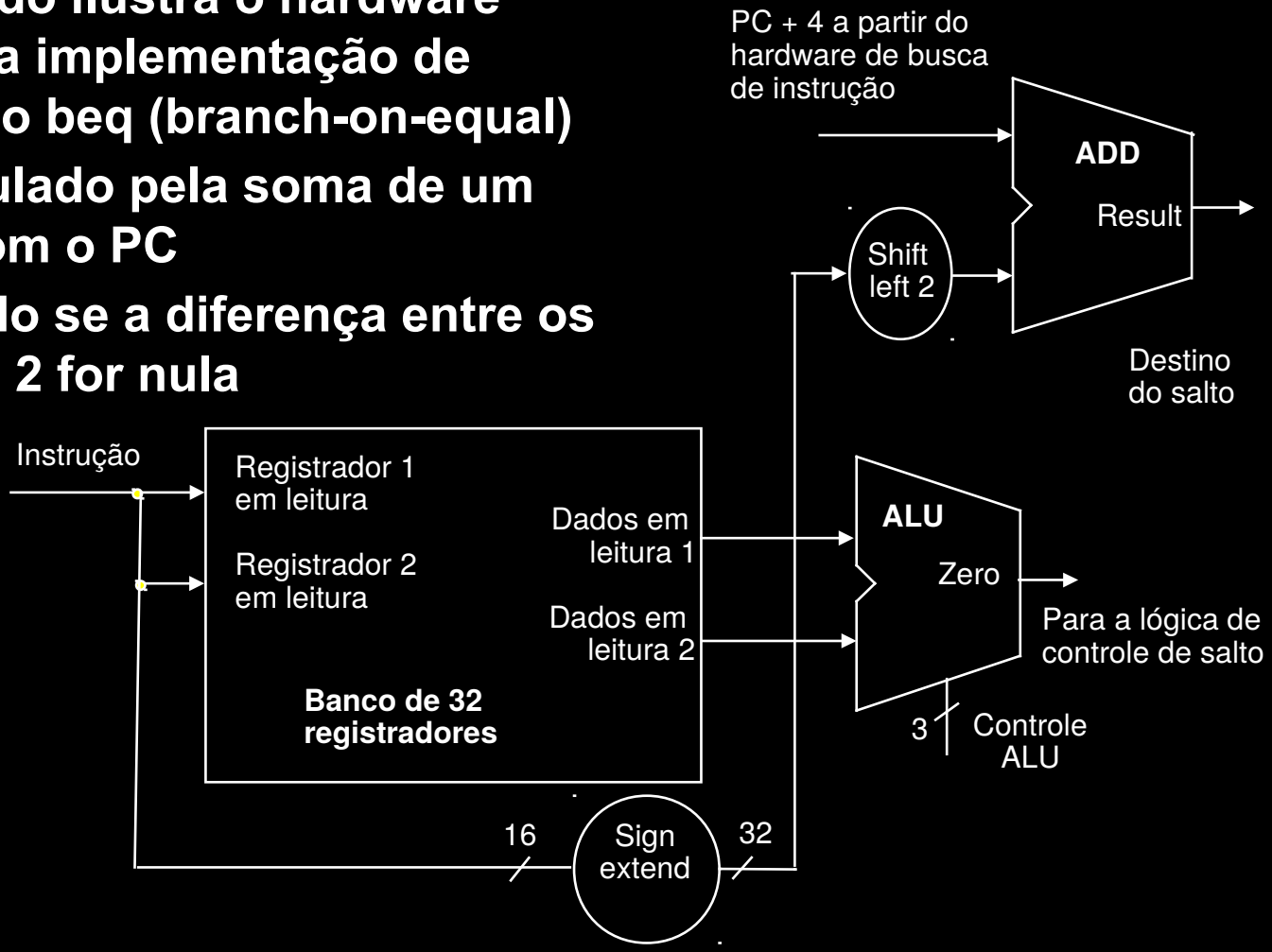
- O processo de leitura é semelhante ao de escrita
- A diferença básica é a existência de um caminho para escrever o valor lido no banco de registradores



op	rs	rt	offset

4. Instruções de desvio condicional

- O esquema ao lado ilustra o hardware necessário para a implementação de instruções do tipo beq (branch-on-equal)
- O destino é calculado pela soma de um deslocamento com o PC
- O salto é realizado se a diferença entre os registradores 1 e 2 for nula



op	rs	rt	offset

5. Combinando instruções

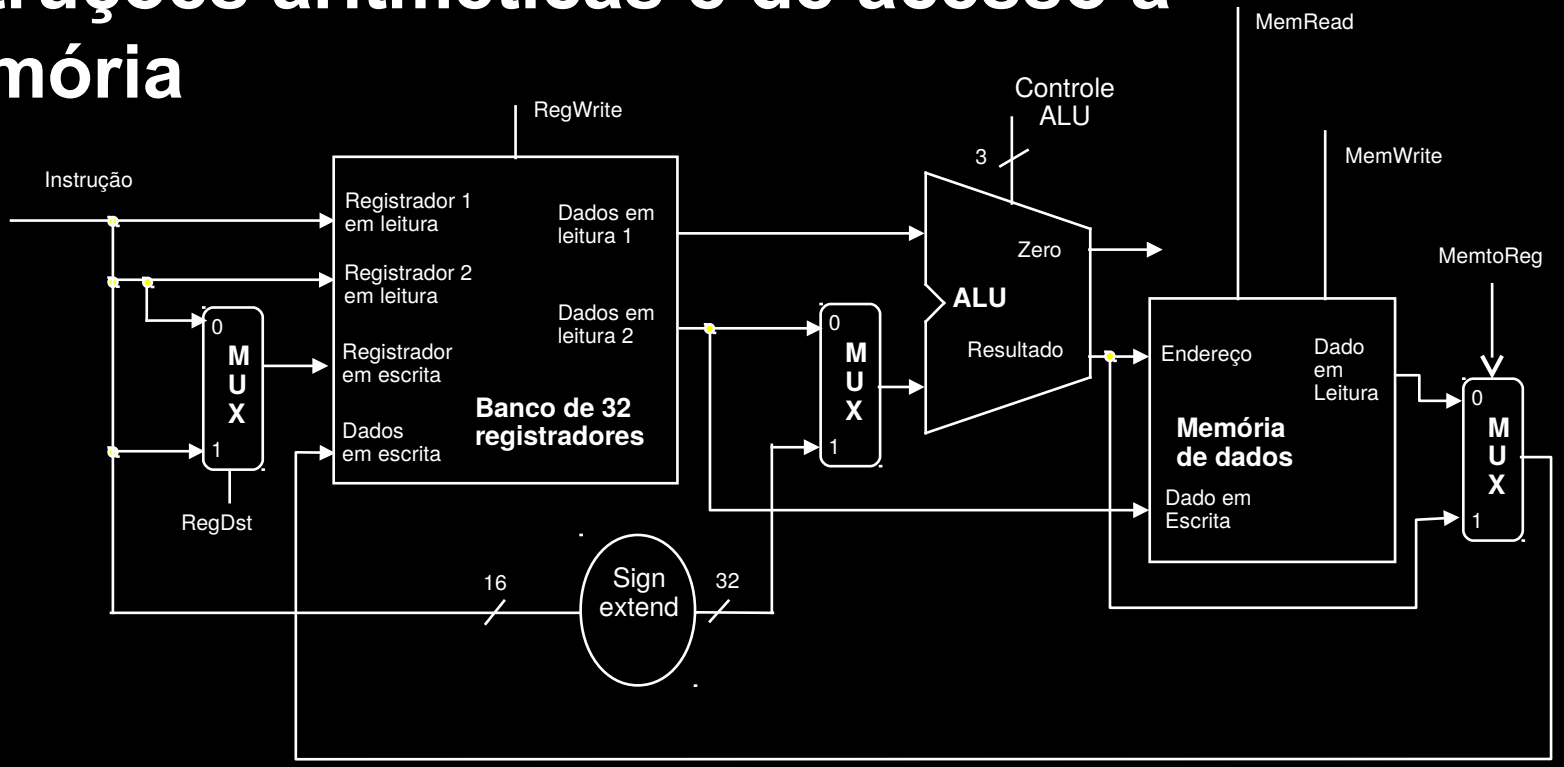
- Os recursos para as instruções aritméticas e para as instruções de acesso à memória são semelhantes. Seria possível combiná-los em um só?
- A segunda entrada da ALU ...
  - é um registrador, no caso de instruções aritméticas
  - é obtida a partir da extensão dos 16 bits inferiores da instrução, no caso de instruções de acesso à memória
- O valor a ser escrito no registrador destino ...
  - vem da saída da ALU no caso de uma operação aritmética
  - vem da memória nas instruções de acesso à memória

5. Combinando instruções

- **Uma parte operativa combinada pode ser obtida através da inserção de multiplexadores nestes pontos**

Combinando instruções

Bloco operacional considerando instruções aritméticas e de acesso à memória

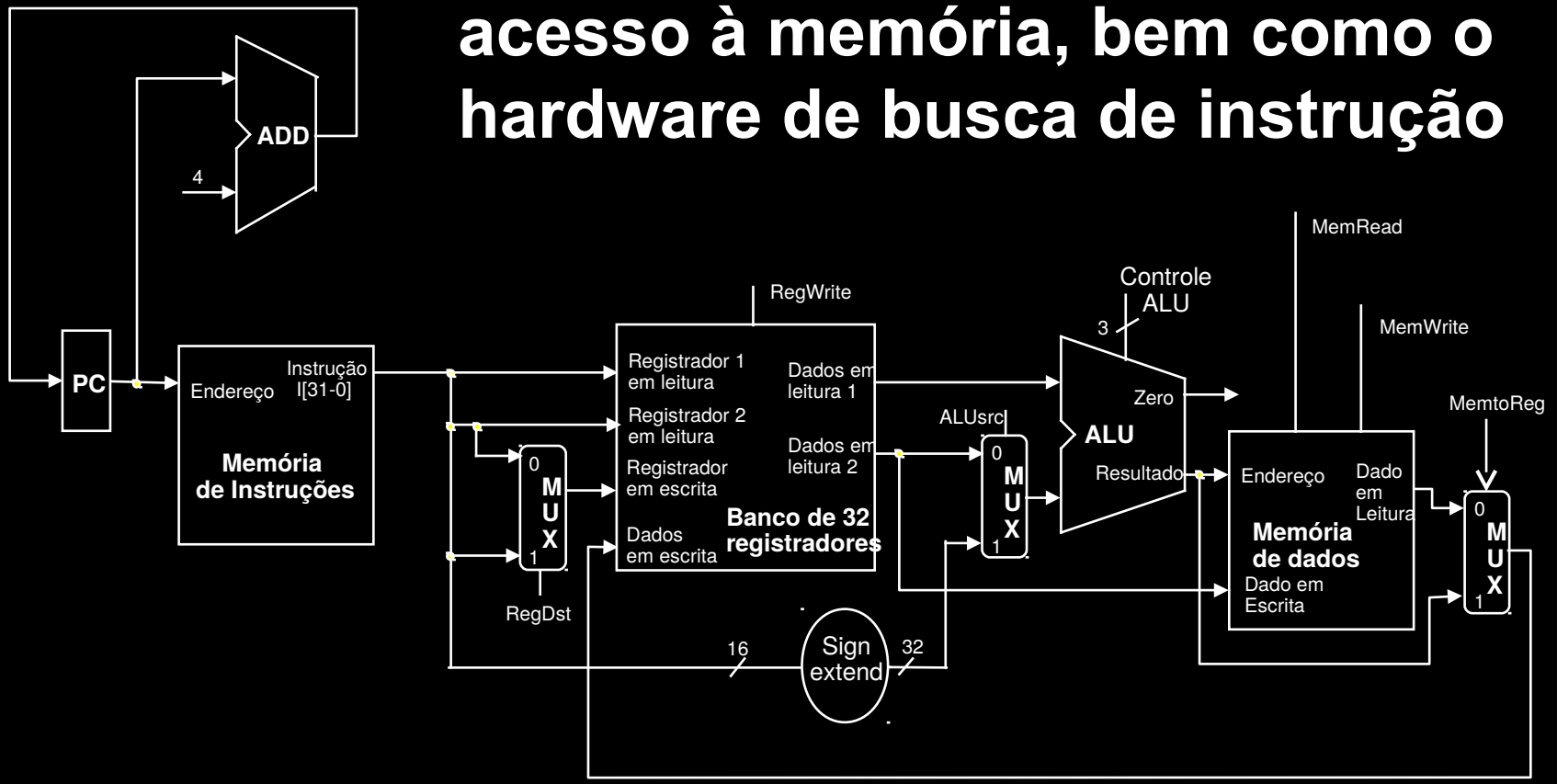


op	rs	rt	rd	shamt	funct

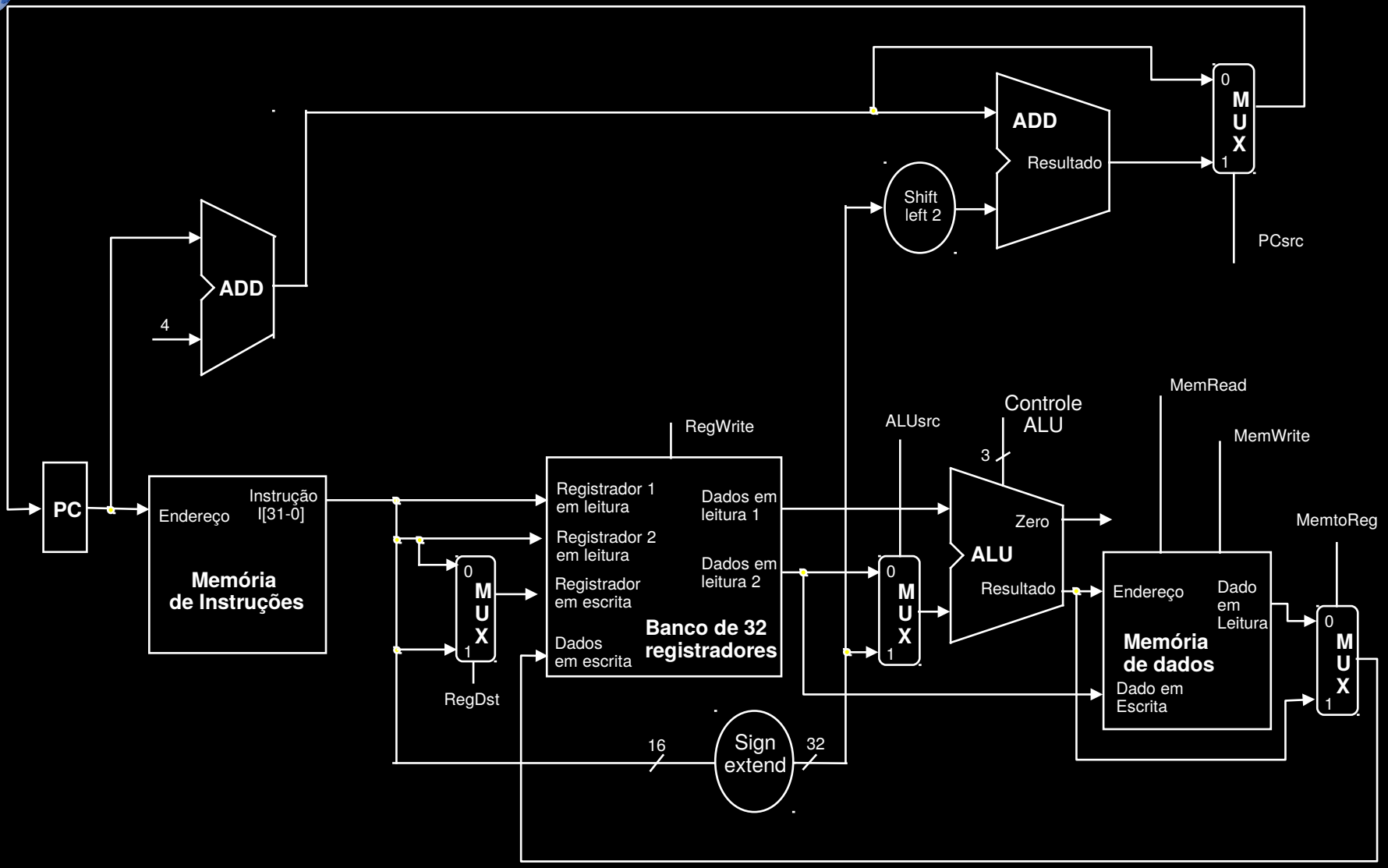
op	rs	rt	offset

Combinando instruções

Bloco operacional considerando instruções aritméticas e de acesso à memória, bem como o hardware de busca de instrução



6. Bloco operacional completo



Controle

- Seleção de operações (ALU, read/write, etc.)
- Controle do fluxo de dados (multiplexadores)
- Informações extraídas dos 32 bits da instrução
- Ex: add $8, $17, $18

  Formato da Instrução:

op	rs	rt	rd	shamt	funct
000000	10001	10010	01000	00000	100000

- Operações das ALU's são baseadas nos campos tipo de instrução e no código da função

Controle

- **Exemplo: lw $1, 100($2)**

op	rs	rt	offset
35	2	1	100

- **Entradas de controle da ALU**

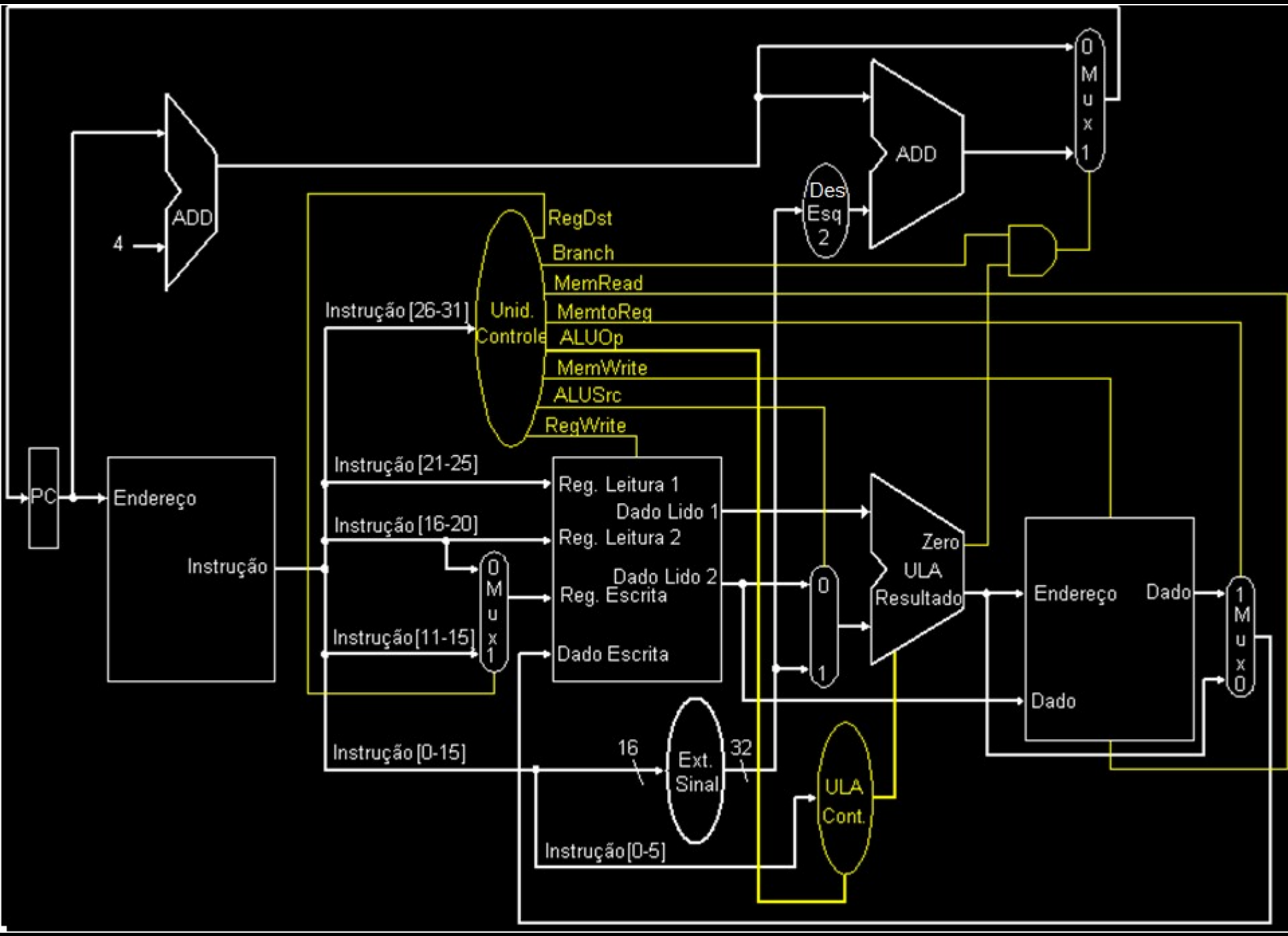
```
000   AND
001   OR
010   add
110   subtract
111   set-on-less-than
```

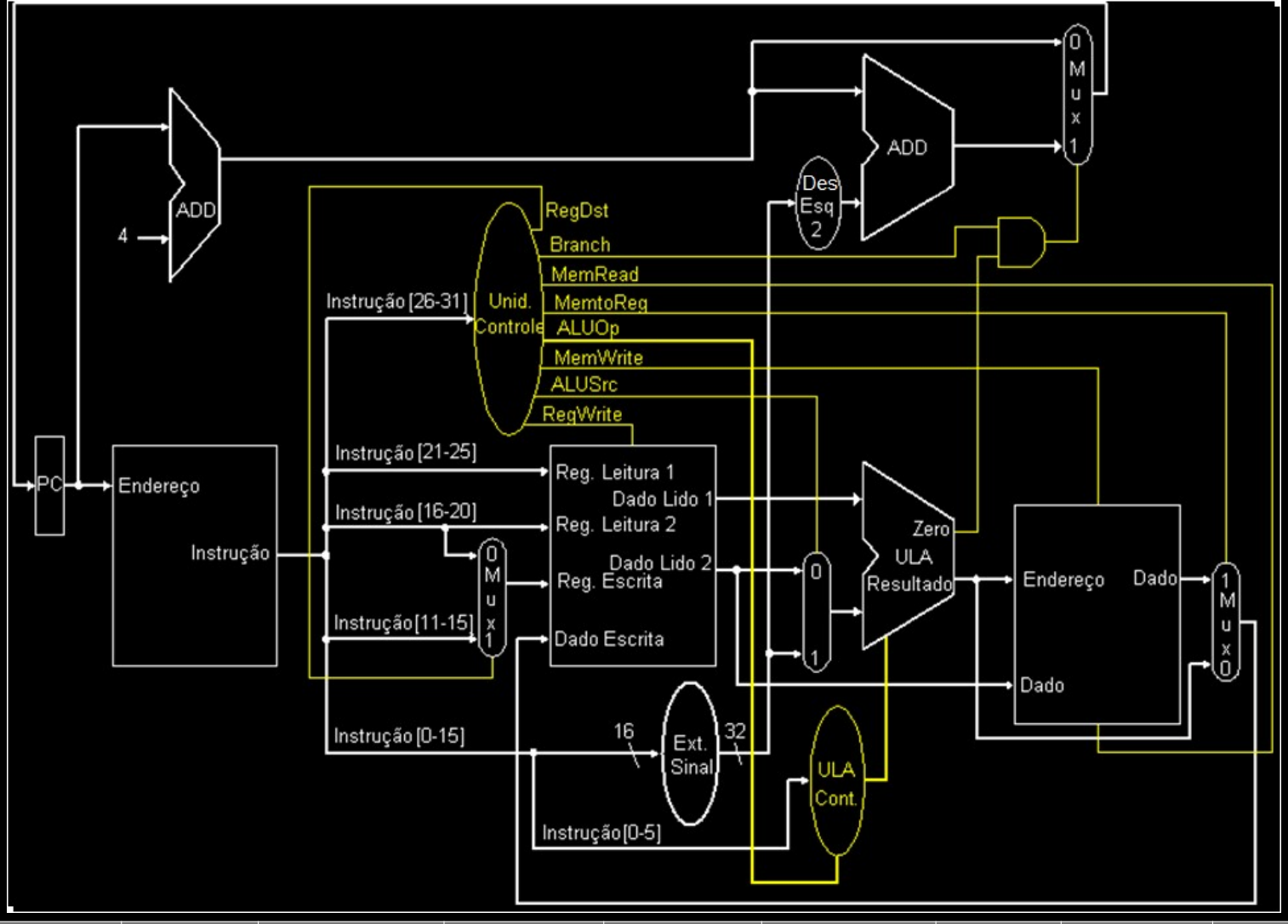
Controle

- O hardware deve permitir executar o conjunto de instruções
  - dado o tipo de instrução
    - 00 = lw, sw
    - 01 = beq,
    - 11 = aritmética

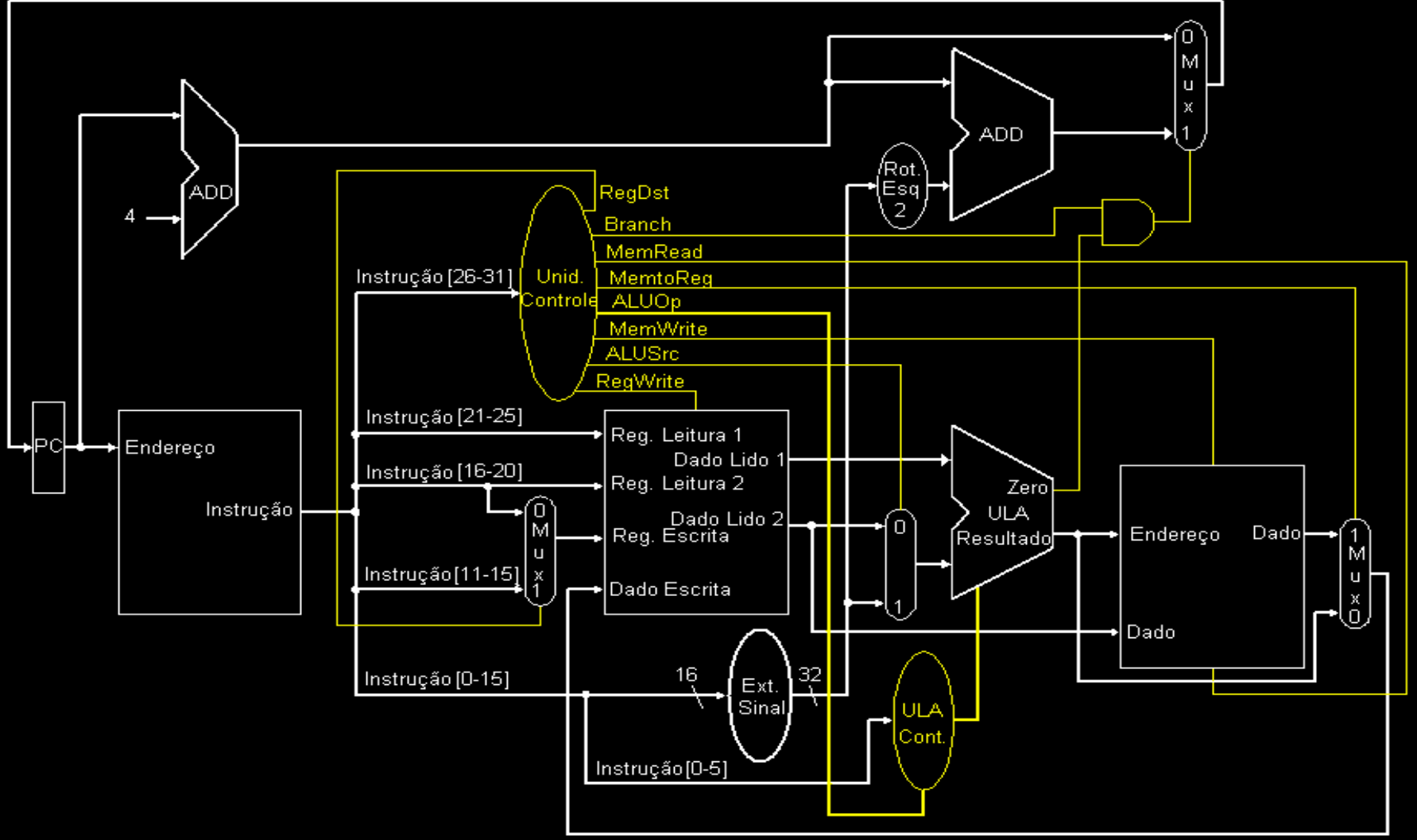
    } ALUOp Obtido do tipo de instrução
  - código da função aritmética

AluOp		Funct						Operação
Aluop1	AluOp2	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
0	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111


ADD
4
ADD
0
M
u
x
1
Des
Esq
2
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instrução [26-31]
Unid.
Controle
PC
Endereço
Instrução
Instrução [21-25]
Instrução [16-20]
Instrução [11-15]
0
M
u
x
1
Reg. Leitura 1
Dado Lido 1
Reg. Leitura 2
Dado Lido 2
Reg. Escrita
Dado Escrita
0
1
Zero
ULA
Resultado
Endereço
Dado
Dado
1
M
u
x
0
Instrução [0-15]
16
Ext.
Sinal
32
ULA
Cont.
Instrução [0-5]



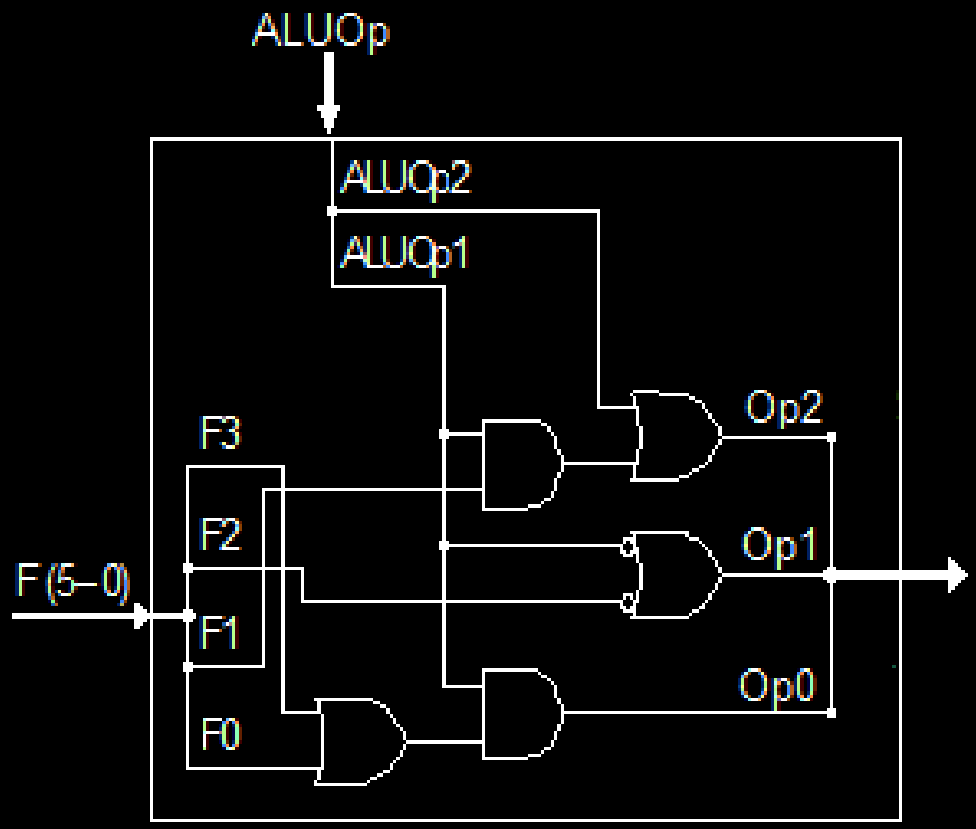
Instrução	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp2
R-format									
lw									
sw									
beq									

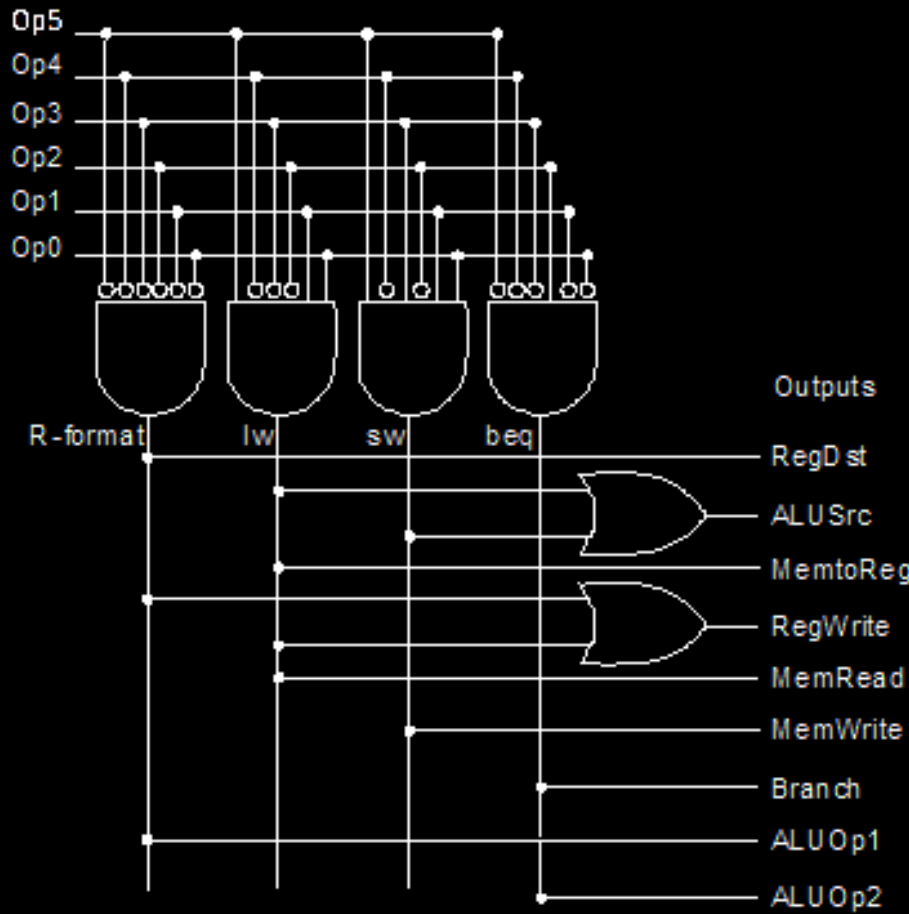


Instrução	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp2
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Controle

- **Implementação da Unidade de Controle usando Lógica combinatória (tabela verdade)**



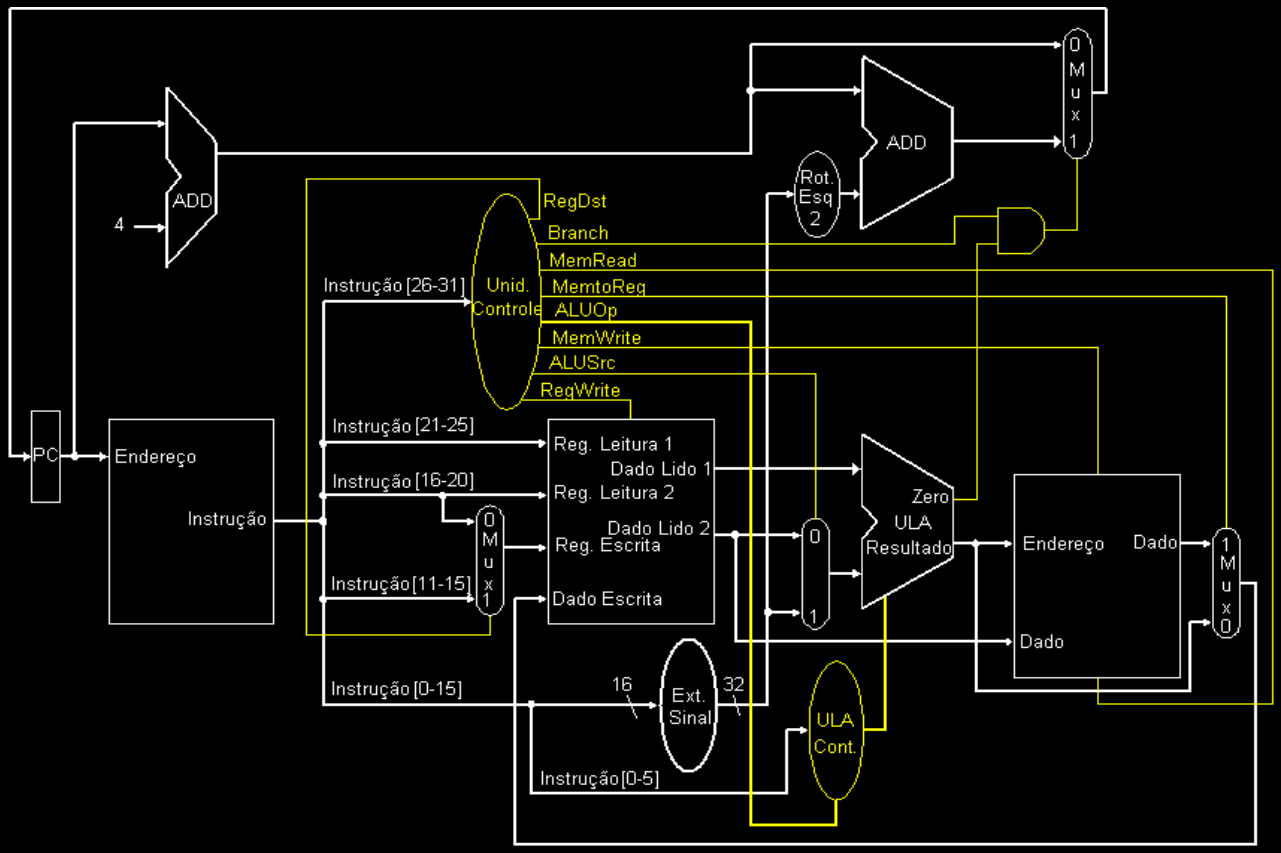


Nossa estrutura de Controle

- Esperamos que os valores sejam estáveis
  - ALU poderia não produzir a resposta correta
  - Nós usamos sinais que são controlados por um clock que determina quando escrever
- Tempo de Ciclo é determinado pelo caminho mais longo

Implementação único Ciclo

- **Cálculo do *tempo de ciclo* assumindo atrasos desprezíveis exceto:**
  - **memória (2ns), ALU and somadores (2ns), acesso a registradores (1ns)**



Onde Nós Estamos

- Problemas do ciclo único:
  - Instruções complicadas como ponto flutuante, como fazer?
  - Gasto de área
- Uma solução:
  - usar o menor *tempo de ciclo*
  - Ter diferentes números de ciclos para instruções diferentes
  - datapath "multiciclo":