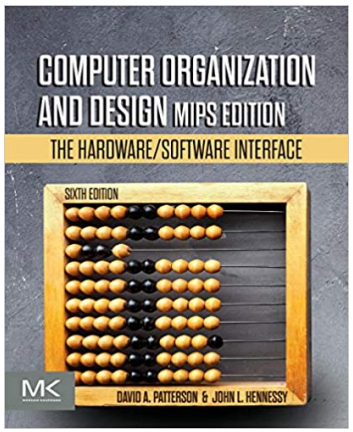
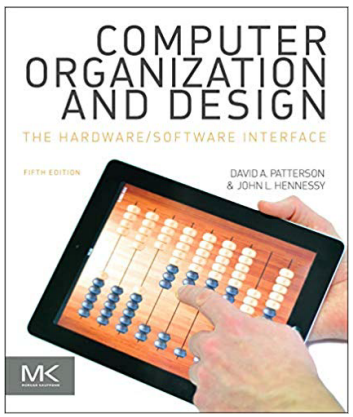
Aula 06 –Caminho de Dados

Prof. Dr. Clodoaldo A. de Moraes Lima

Material baseado no livro "Patterson, David A., Hennessy, J. L. - Computer Organization And Design: The Hardware/Software Interface"





Sumário

Introdução

Até o momento, analisamos as instruções MIPS básicas:

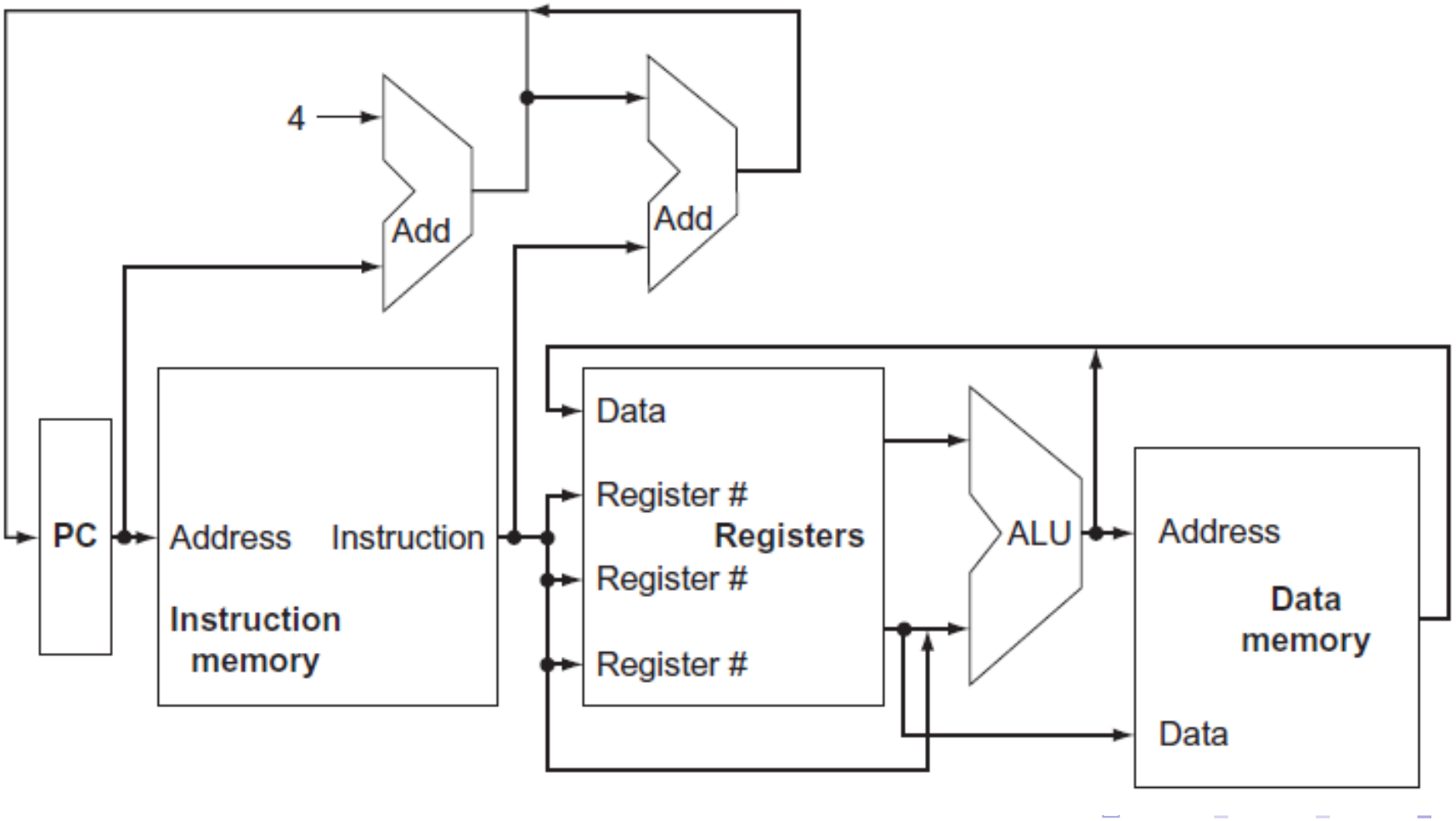
- Aritméticas (add, sub);
- Lógicas(or, slt);
- De Referência à memória (lw, sw);
- De desvio(beq, j, jr).

Para cada instrução, duas etapas são idênticas:

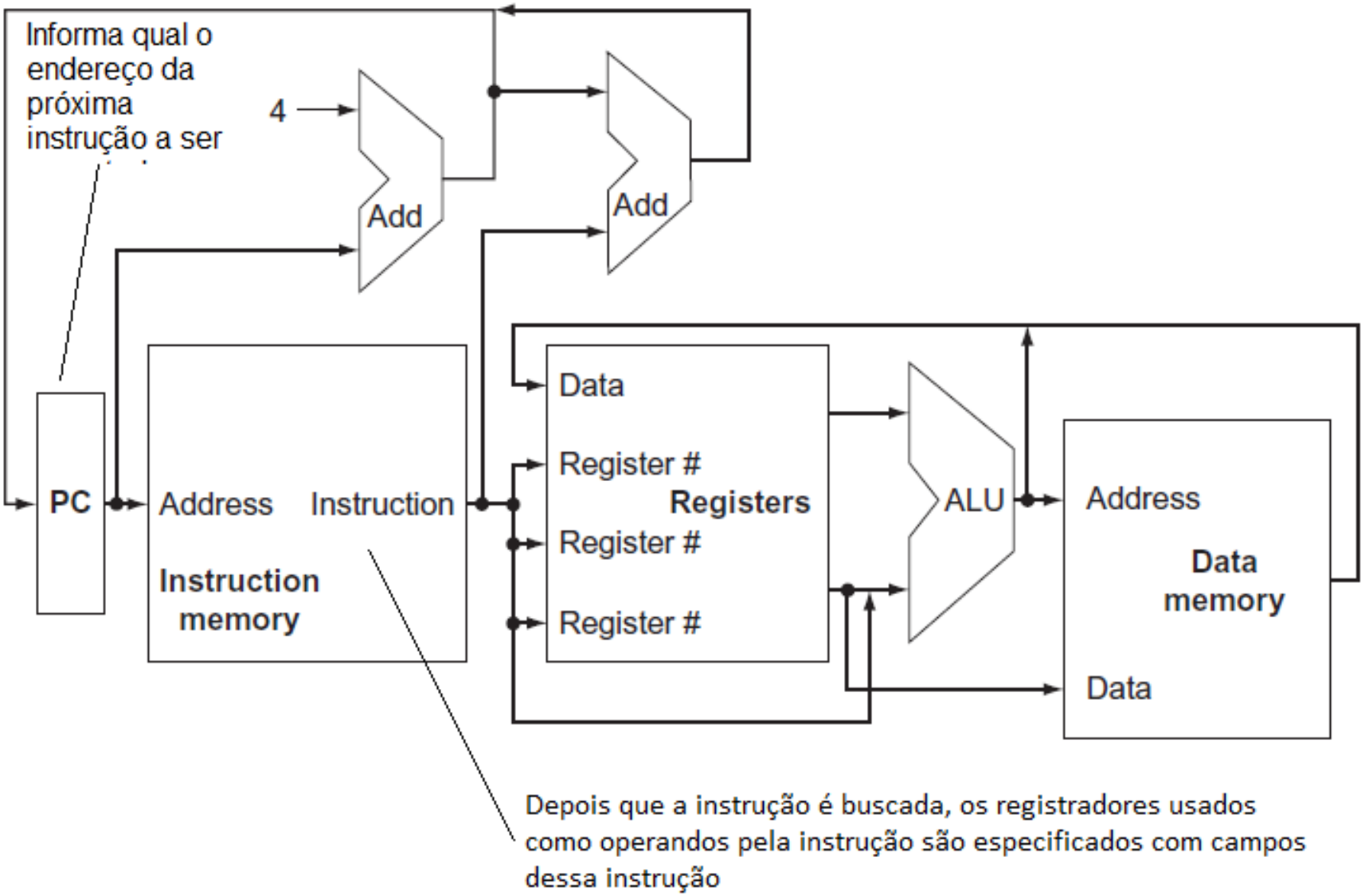
- Enviar o contador de Programa (PC) à memória que contém o código e buscar a instrução dessa memória;
- Ler um ou mais registradores, usando campos da instrução para selecionar os registradores a serem lidos

Introdução

- Após essas duas etapas, as ações necessárias para completar a instrução dependem da classe de instrução:



Introdução


Informa qual o endereço da próxima instrução a ser
4
Add
Add
PC
Address
Instruction
Instruction memory
Data
Register #
Registers
Register #
Register #
ALU
Address
Data memory
Data
Depois que a instrução é buscada, os registradores usados como operandos pela instrução são especificados com campos dessa instrução

Introdução

- Registradores
  - Podem ser operadores para calcular endereço (lw e sw);
  - Podem ser operadores para calcular um resultado aritmético (lógica e aritmética);
  - Podem ser operadores para calcular uma comparação (desvio).

- Se a instrução for uma instrução lógica ou aritmética, o resultado da ULA precisa ser escrito em um registrador.
- Se a operação for um load ou store, o resultado da ULA é usado como um endereço para armazenar o valor de um registrador ou ler um valor da memória para um registrador. O resultado da ULA ou memória é escrito de volta no banco de registradores.
- Os desvios exigem o uso da saída da ULA para determinar o próximo endereço de instrução, que vem da ULA ou de um somador que incrementa PC atual em 4.

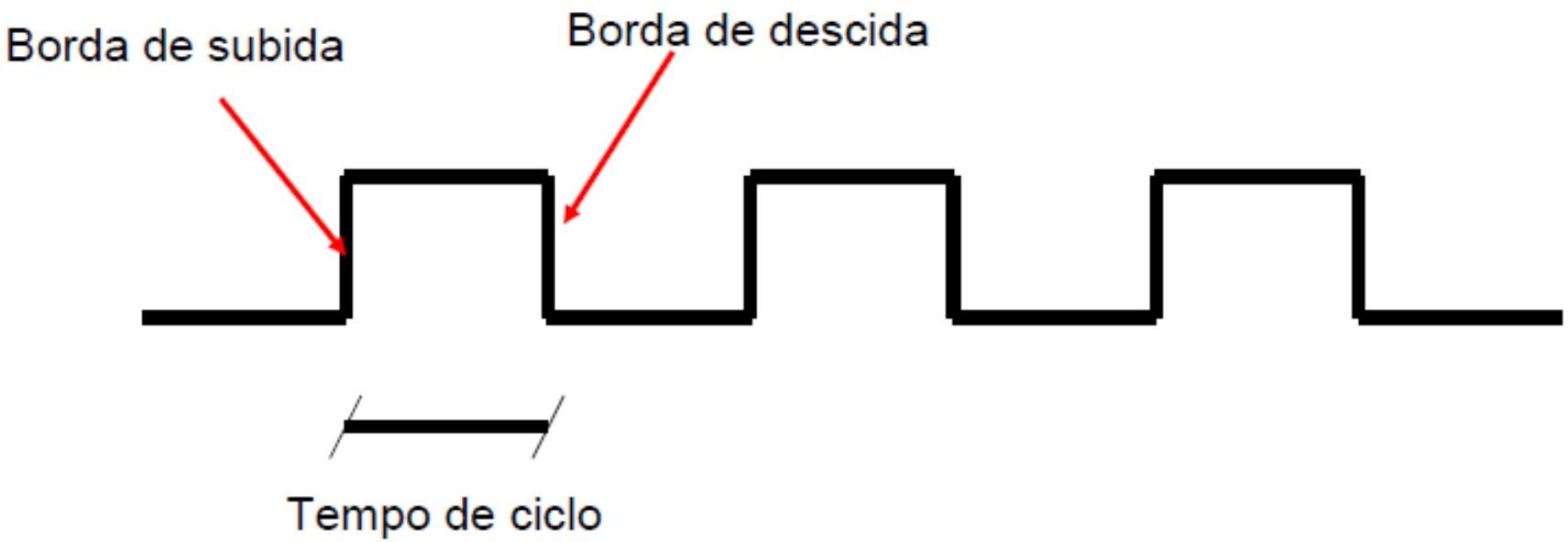
Convenções Lógicas de Projeto

- As unidades funcionais na implementação MIPS consistem em dois tipos diferentes de elementos lógicos:
  - Combinacionais: saídas dependem apenas das entradas atuais (Portas Lógicas);
  - Sequenciais (De Estado): que contém estado se tiver algum armazenamento interno (Flip-Flop).

Convenções Lógicas de Projeto

- Uma metodologia de clocking define quando os sinais podem ser lidos e quando podem ser escritos. É importante para especificar a sincronização das leituras e escritas, evitando que as mesmas ocorram simultaneamente, o que geraria um erro.
- Uma metodologia de sincronização acionada por transição, significa que quaisquer valores armazenados em um elemento lógico sequencial são atualizados apenas em uma transição do clock.

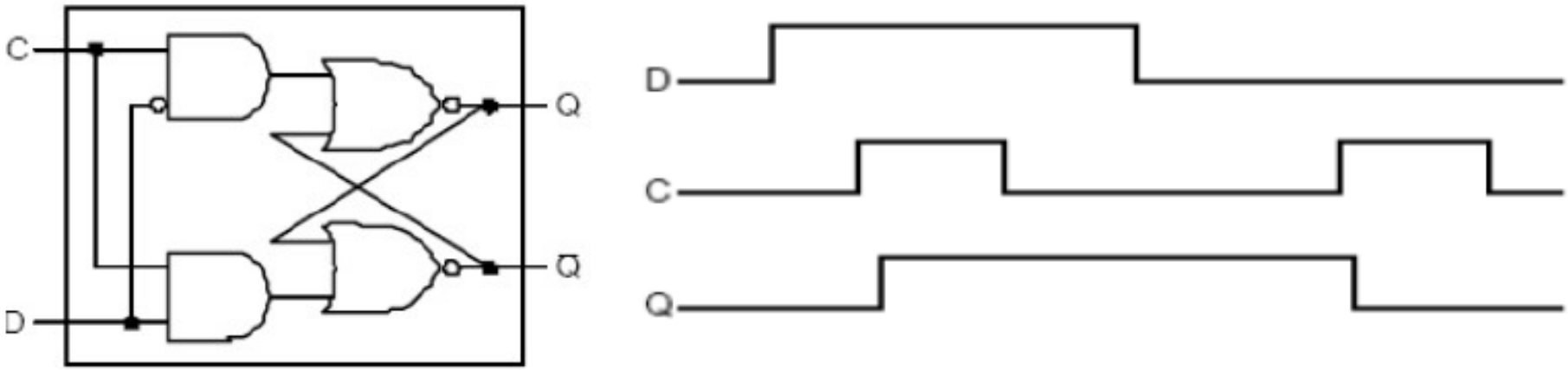
Convenções Lógicas de Projeto



Convenções Lógicas de Projeto

- Uma metodologia acionada por transição permite que um elemento de estado seja lido e escrito no mesmo ciclo de clock sem criar uma disputa que poderia levar a valores de dados indeterminados.

Convenções Lógicas de Projeto

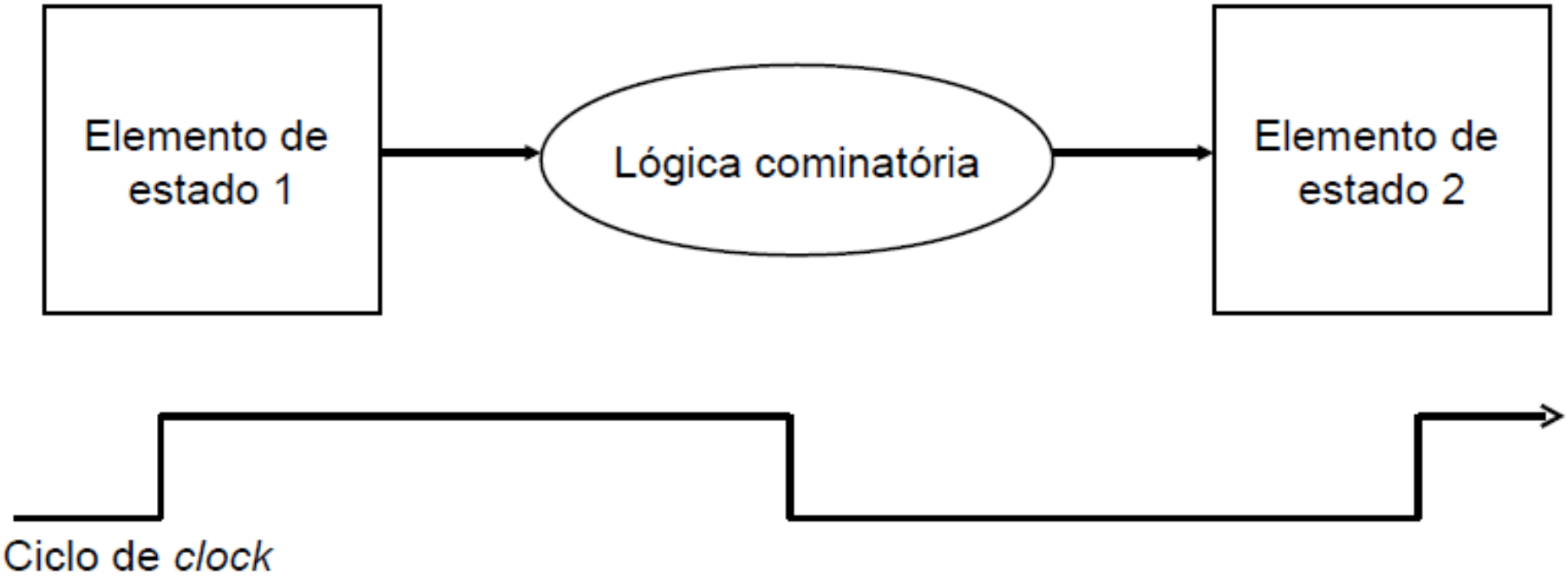


- O valor a ser armazenado (D);
- O sinal do relógio (C) indicando leitura ou escrita;
- O valor do estado interno (Q) e o seu complemento (Q).

Convenções Lógicas de Projeto

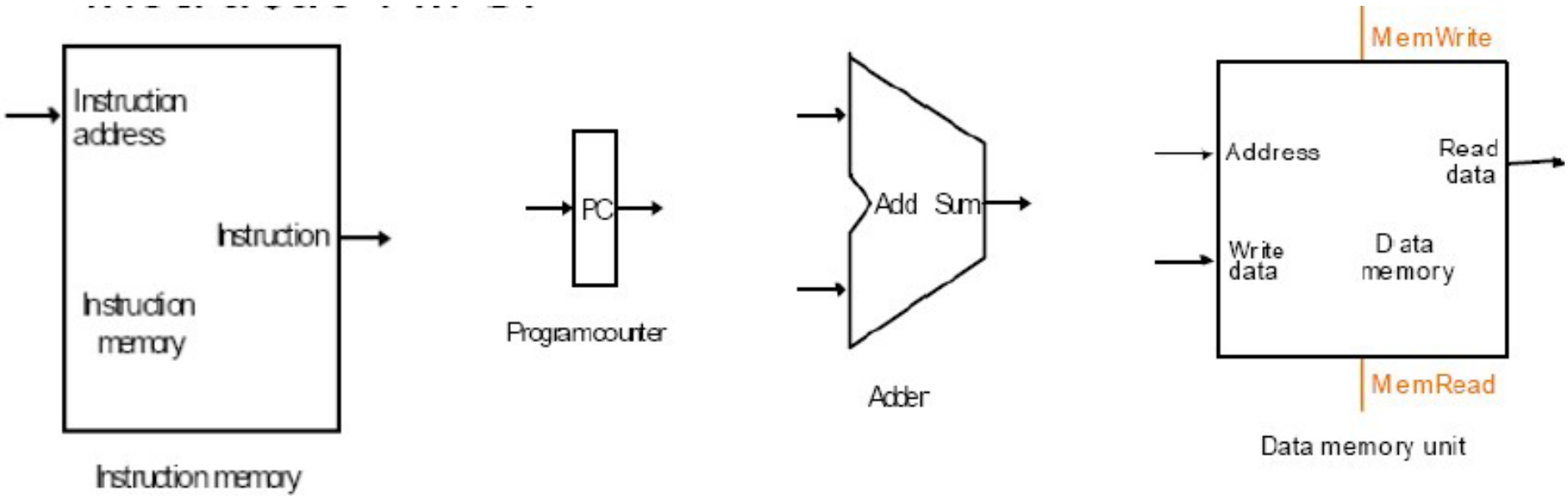
- Como apenas os elementos de estado podem armazenar valores de dados, qualquer coleção de lógica combinatória precisa ter suas entradas vindo de um conjunto de elementos de estados e suas saídas escritas em um conjunto de elementos de estado
- As entradas são valores escritos em um ciclo de clock anterior, enquanto as saídas são valores que podem ser usados em um ciclo de clock seguinte.

Convenções Lógicas de Projeto



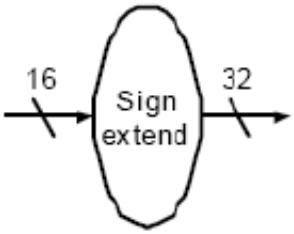
Construindo um Caminho de Dados

- Principais componentes necessários para executar cada classe de instrução MIPS.
- Elementos do caminho de dados:

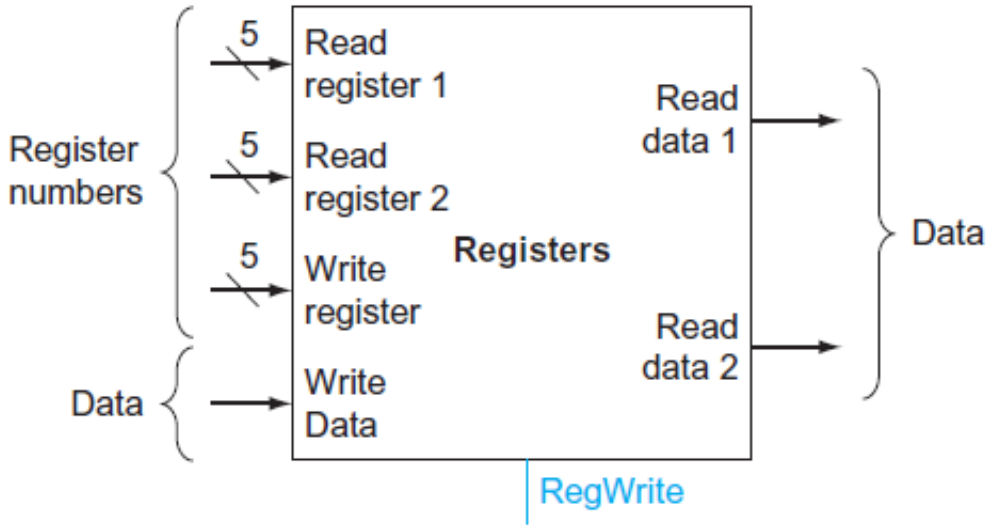


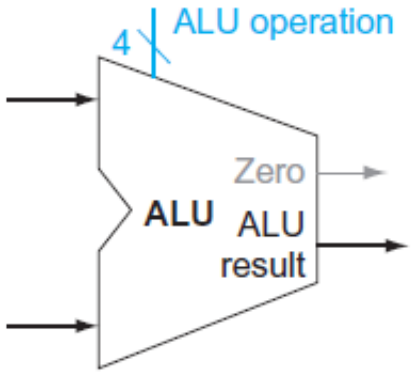
Construindo um Caminho de Dados

Elementos do caminho de dados:



Sign-extension unit





Construindo um Caminho de Dados

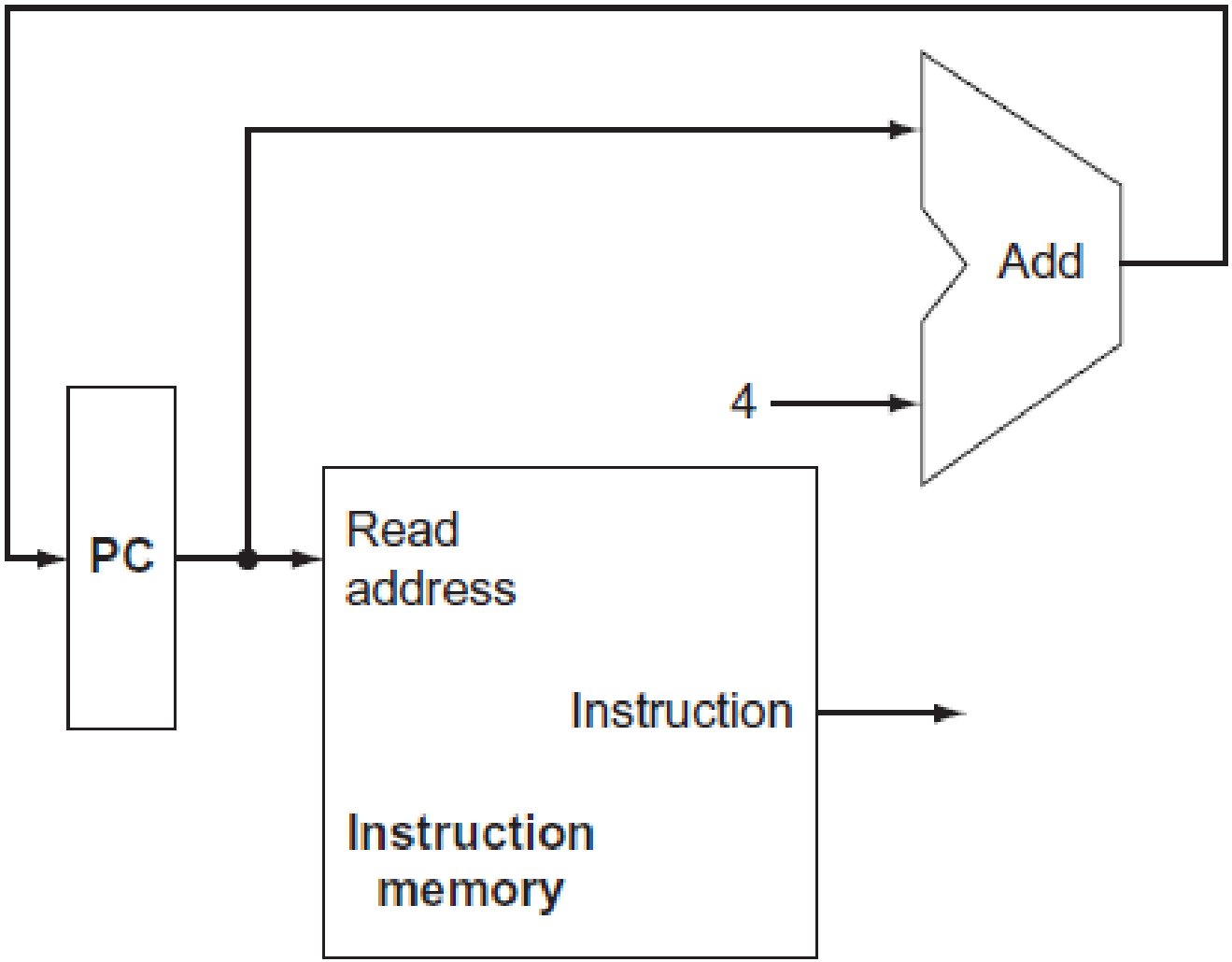
- A memória de Instruções (instruction memory) armazena as instruções de um programa e fornece instruções dado um endereço.
- O contador de programa (program counter – PC) é usado para conter o endereço da instrução atual.
- O somador (Adder) é utilizado para incrementar o PC para o endereço da próxima instrução, é um somador combinatório.

Construindo um Caminho de Dados

Para executar qualquer instrução, precisamos começar buscando a instrução na memória de instruções.

Para preparar para executar a próxima instrução, também temos de incrementar o contador de programa de modo que aponte para a próxima instrução.

Construindo um Caminho de Dados

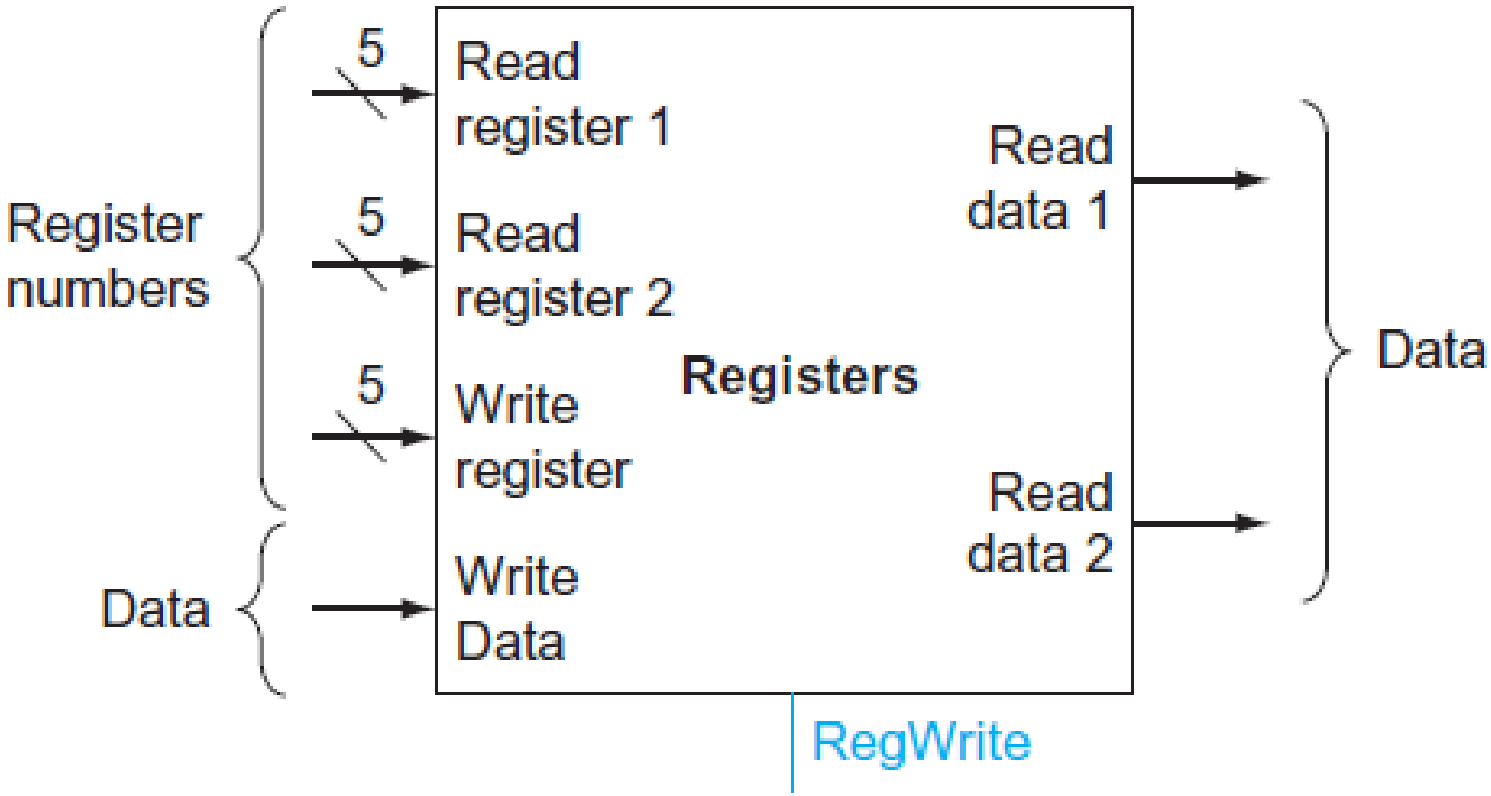


Construindo um Caminho de Dados

Banco de registradores:

- Armazena os registradores de uso geral de 32 bits do processador;
- Os registradores são identificados por um número e podem ser lidos ou escritos;
- Contém o estado dos registradores da máquina que poderá ser passado para a ULA.

Construindo um Caminho de Dados



Construindo um Caminho de Dados

- Uma instrução do formato R lê dois registradores, realiza uma operação na ULA com o conteúdo dos registradores e escreve o resultado em um terceiro registrador.
- Devido às instruções de formato R terem três operandos de registrador, precisaremos ler duas palavras de dados do banco de registradores eescrever uma palavra de dados no banco de registradores para cada instrução.

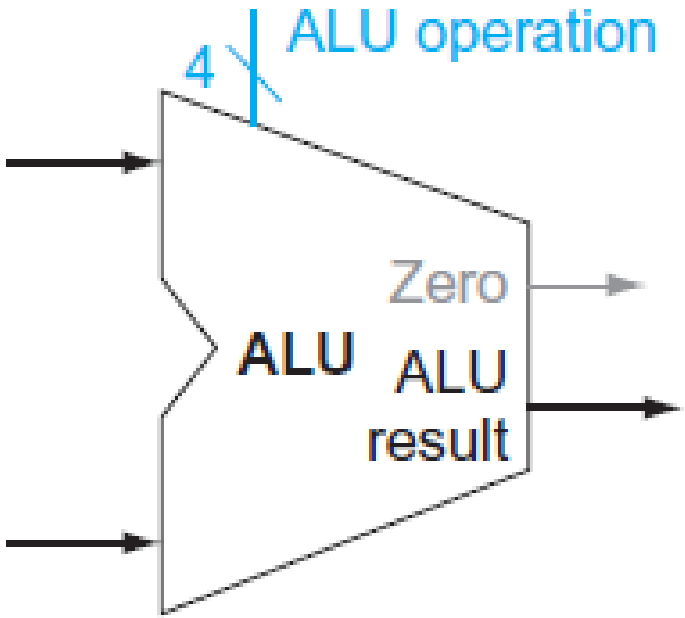
Construindo um Caminho de Dados

- Para ler cada palavra de dados precisamos de uma entrada no banco de registradores que especifique o numero do registrador a ser lido e uma saida que conduzira o valor lido dos registradores.
- Para escrever uma palavra de dados, precisaremos de duas entradas:
  - Uma especificando o numero do registrador a ser escrito;
  - Uma com os dados a serem escritos no registrador.
- As escritas, no entanto, sao controladas pelo sinal de controle de escrita, que precisa ser ativo para que uma escrita ocorra na transicao do clock.

Construindo um Caminho de Dados

ULA

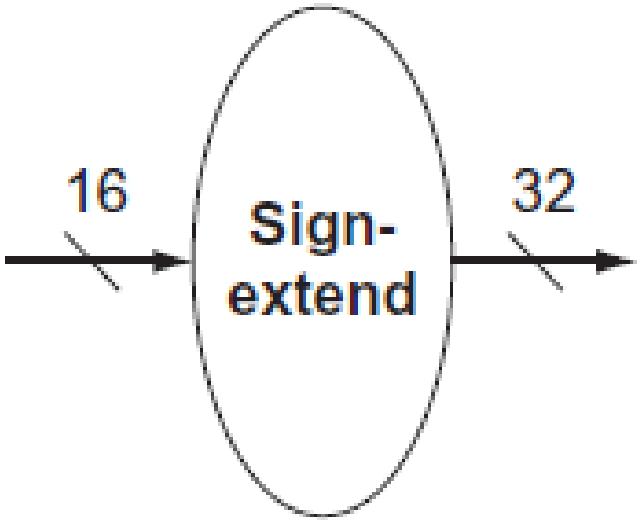
- Usa duas entradas de 32 bits e produz um resultado de 32 bits, bem como um sinal de 1 bit se o resultado for 0;
- A ULA possui um entrada de 4 bits de controle (ALU control) para determinar a operação.



Construindo um Caminho de Dados

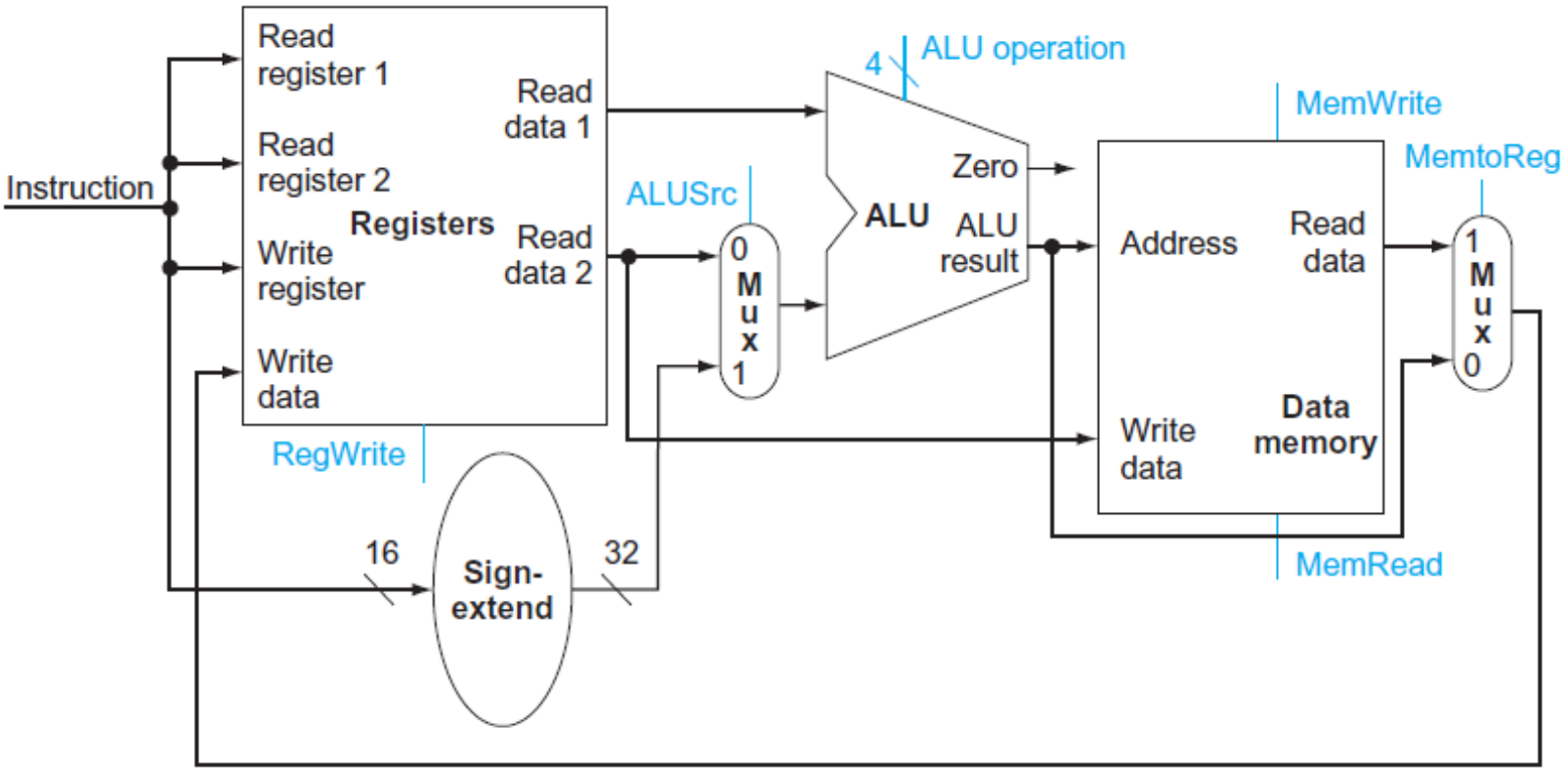
Unidade de extensão de sinal

- Utilizada para estender o sinal de um valor de 16 bits para 32 bits;
- Útil para quando for ler ou escrever na memória utilizando instruções que somam um valor de deslocamento (offset) ao endereço base armazenado em um registrador (ex: lw e sw).



Construindo um Caminho de Dados

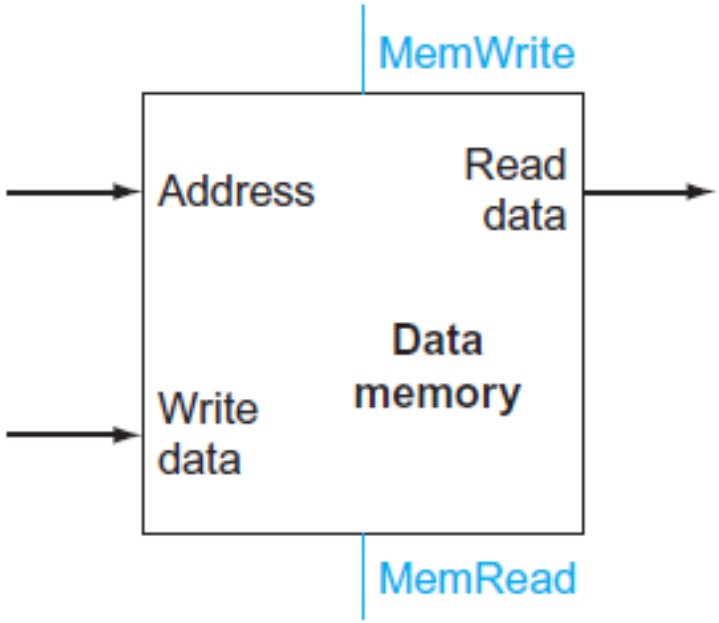
Instrução do tipo R



Construindo um Caminho de Dados

Memória de Dados

- Uma unidade de memória de dados é um elemento de estado com entradas para os endereços e os dados de escrita, e uma única saída para o resultado da leitura.
- Possui controles de leitura e escrita separados, embora apenas um deles pode ser ativado em qualquer clock específico.

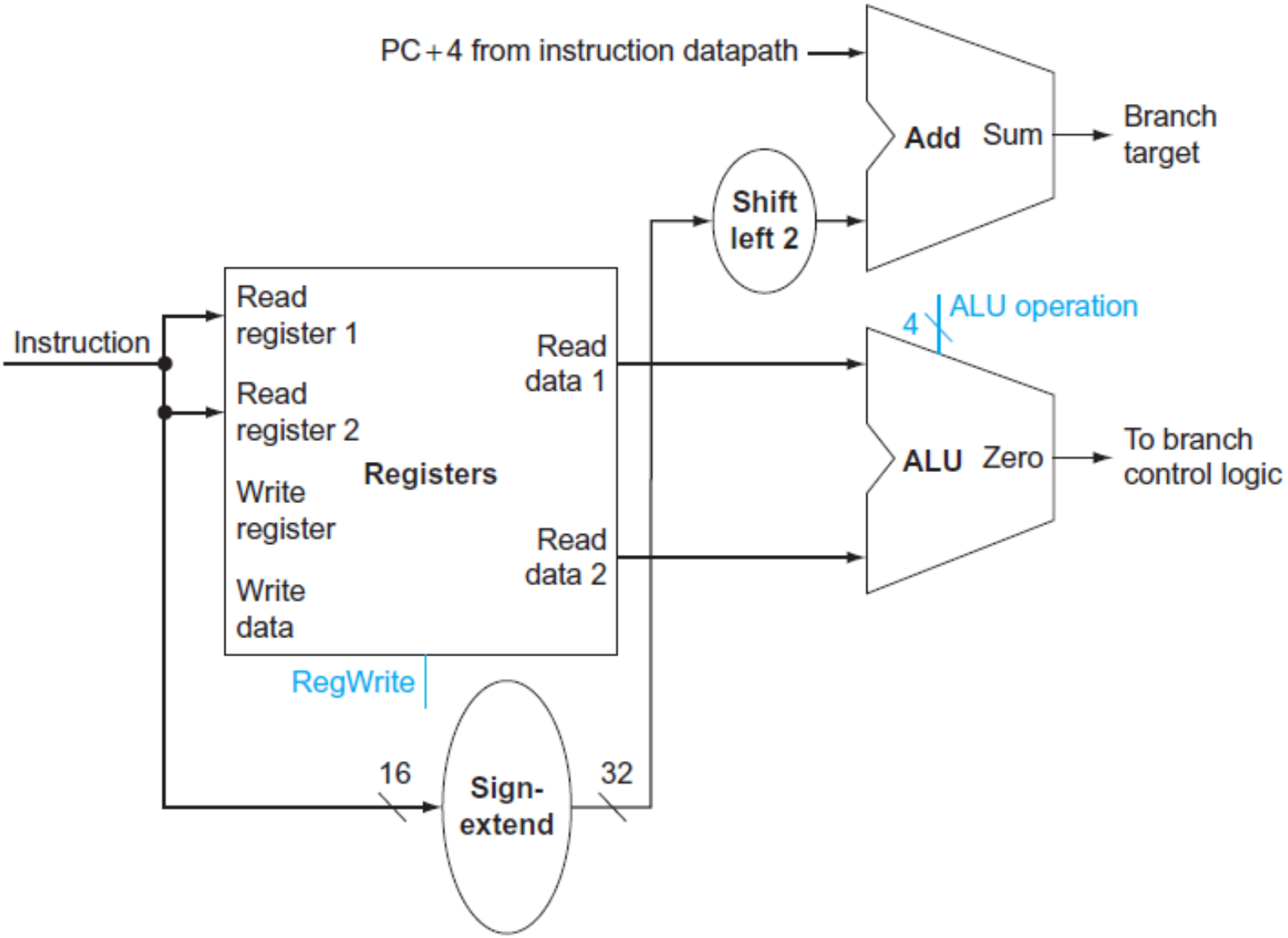


Construindo um Caminho de Dados

Endereco de destino do desvio:

- Endereco especificado em um desvio, que se torna o novo contador de programa (PC) se o desvio for tomado;
- O destino do desvio e dado pela soma do campo offset da instrucao e o endereco da instrucao seguinte ao desvio;
- Forma de instrucao de desvio beq $t1, $t2, offset.

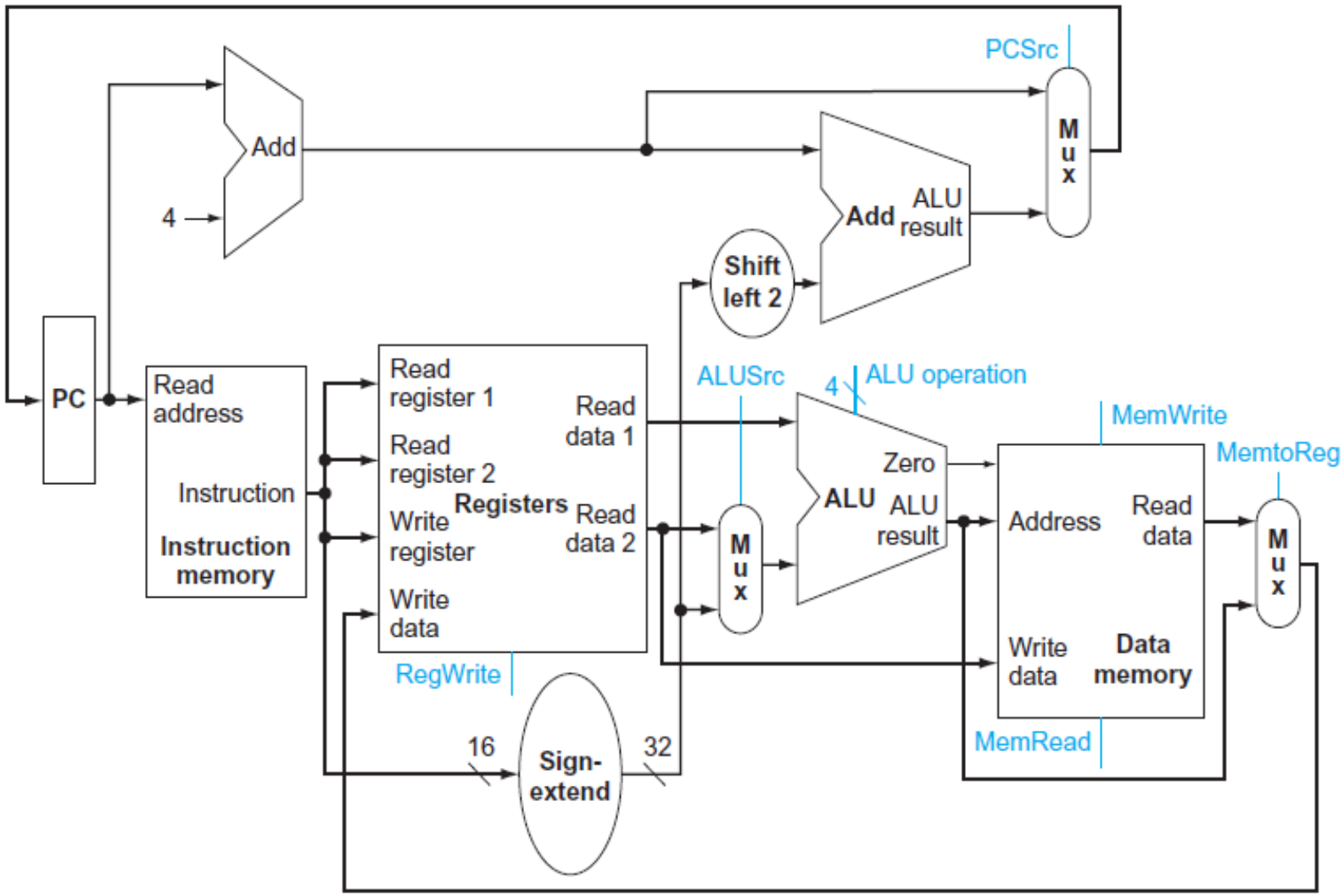
Construindo um Caminho de Dados



Construindo um Caminho de Dados

- Além de calcular o endereço de destino do desvio, também precisamos determinar se a próxima instrução é a instrução que segue sequencialmente ou a instrução no endereço de destino do desvio.
- Quando a condição é verdadeira, o endereço de destino do desvio se torna o novo PC, dizemos que o desvio é tomado.
- Quando a condição é falsa, o PC incrementado deve substituir o PC atual, dizemos que o desvio é não tomado.

Construindo um Caminho de Dados



Controle da ULA

- A ULA possui quatro entradas de controle.
- Esses bits não foram codificados, portanto, apenas 6 das 16 combinações possíveis são usadas neste subconjunto:

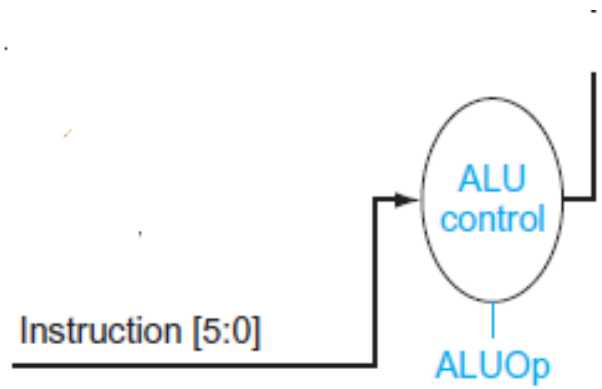
ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

Controle da ULA

- Para instruções do tipo R, a ULA precisa realizar uma das cinco ações (AND, OR, add, subtract ou set on less than) dependendo do campo funct (função) de 6 bits.
- A função NOR é necessária para outras partes do conjunto de instruções MIPS.
- Para as instruções lw ou sw, a ULA é usada para calcular o endereço de memória por adição.
- Para instruções branch equal, a ULA precisa realizar uma subtração.

Controle da ULA

- Podemos gerar a entrada do controle da ULA de 4 bits usando uma pequena unidade de controle que tenha como entradas o campo funct da instrução e um campo control de 2 bits, que chamamos OpALU.
- OpALU indica se a operação a ser realizada deve ser:
  - add (00) para loads e stores;
  - subtract (01) para beq;
  - Determinada pela operação codificada no campo funct (10);



Controle da ULA

- É utilizado uma técnica comum, a unidade de controle principal gera os bits de OpALU, que então, são usados como entrada para o controle da ULA que gera os sinais reais para controlar a ULA.
- Essa técnica visa diminuir a complexidade da unidade de controle central e aumentar a sua velocidade utilizando várias unidades de controle menores.

Controle da ULA

Tabela verdade para os três bits de controle da ULA (OpALU):

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111

ALUOp		Funct field						
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	Operation
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

Projeto da Unidade de Controle Principal

- Para começar o projeto, precisamos identificar os campos de uma instrução e as linhas de controle necessárias para o caminho de dados.
- Para entender como conectar os campos de uma instrução:
  - Instruções do tipo R;
  - Instruções de desvio;
  - Instruções de acesso à memória.

Projeto da Unidade de Controle Principal

- O campo op (opcode), está sempre contido nos bits [31:26]. Iremos nos referir a esse campo como Op[5:0].

Field	0	rs	rt	rd	shamt	funct
Bit positions	31:26	25:21	20:16	15:11	10:6	5:0

a. R-type instruction

Field	35 or 43	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

b. Load or store instruction

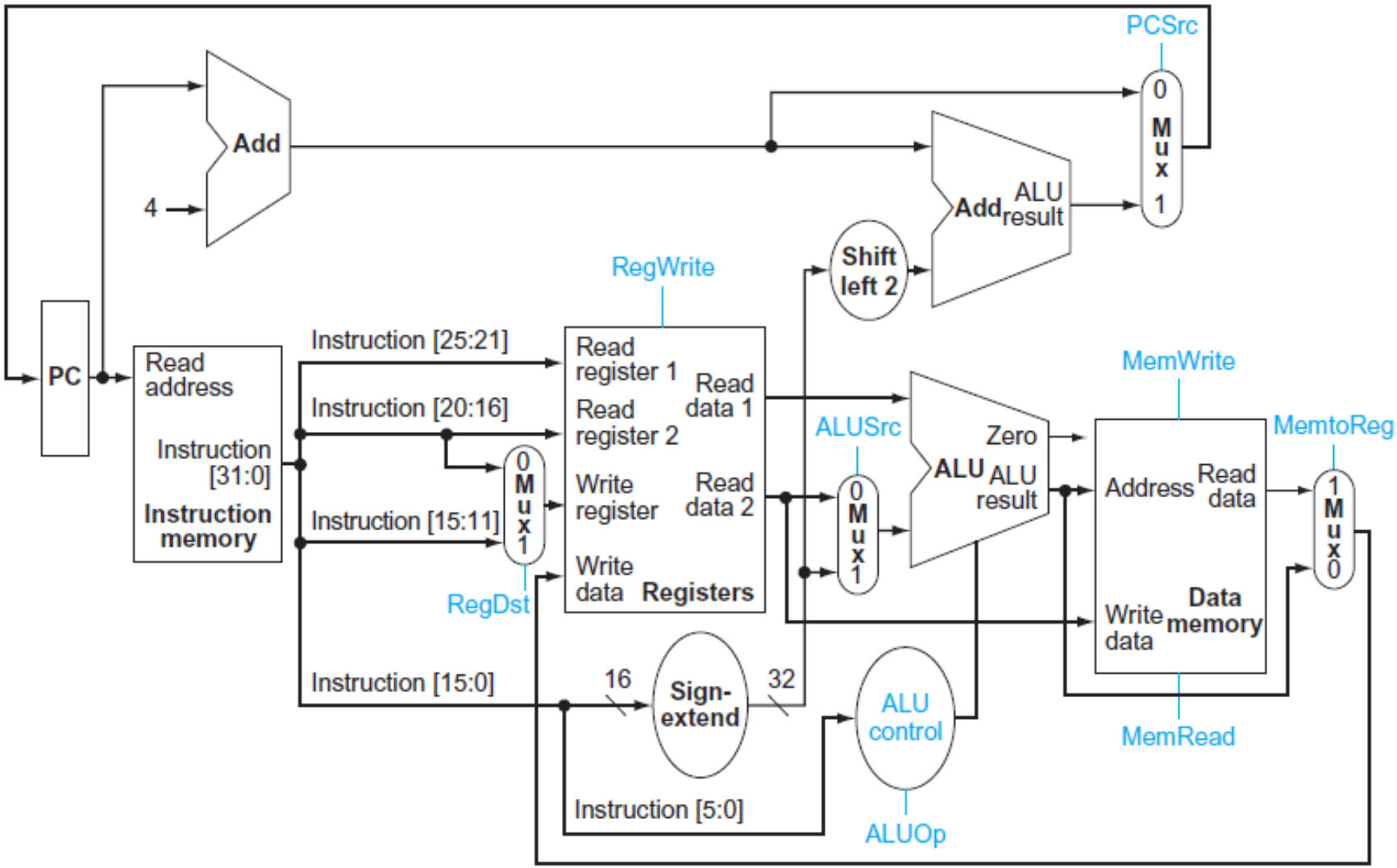
Field	4	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

c. Branch instruction

Projeto da Unidade de Controle Principal

- Os dois registradores a serem lidos sempre são especificados pelos campos rs e rt, nas posições [25:21] e [20:16], nas instruções do tipo R, branch equal ou store.
- O registrador de base para a instrução de load ou store está sempre nas posições de bit [25:21].
- O deslocamento (16 bits) para beq, load e store está sempre nas posições [15:0].
- O registrador de destino está em um dos dois lugares. Para load ele está nas posições [20:16] (rt). Para instruções do tipo R ele está nas posições [15:11] (rd). Portanto, precisamos incluir um multiplexador para selecionar que campo da instrução será usado para indicar o número de registrador a ser escrito.

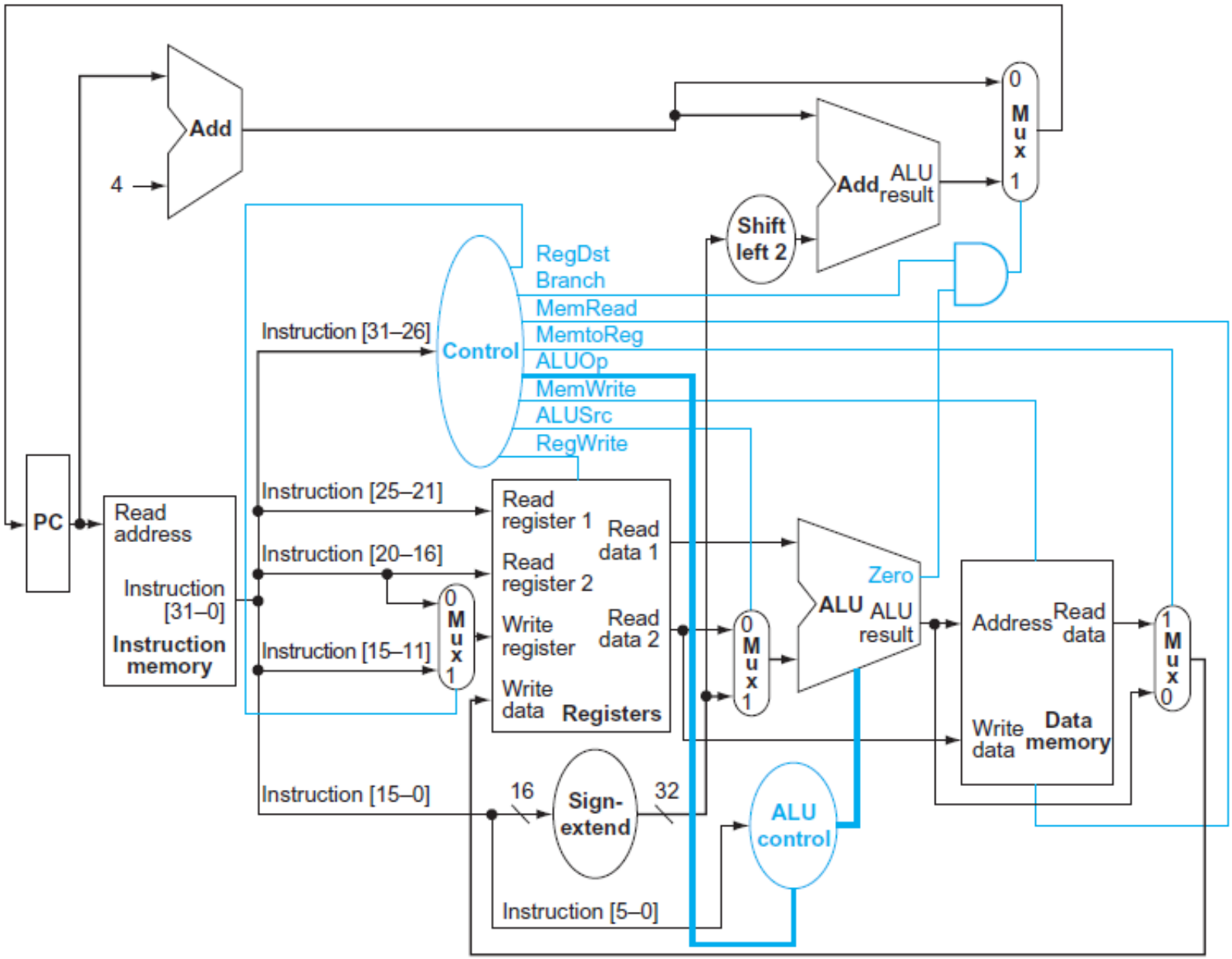
Projeto da Unidade de Controle Principal



Operação do Caminho de Dados

- Antes de mais nada, é necessário entender como cada instrução usa o caminho de dados.
- Todas as operações ocorrem em 1 ciclo de clock
- Podemos pensar em quatro etapas para executar uma instrução do tipo R (add $t1, $t2, $t3); essas etapas são ordenadas pelo fluxo de informação:
  - A instrução é buscada e o PC é incrementado;
  - Dois registradores, $t2 e t3$, são lidos do banco de registradores e a unidade de controle principal calcula a definição das linhas de controle também durante essa etapa;
  - A ULA opera nos dados lidos do banco de registradores, usando o código de função (bits 5:0, campo funct) para gerar a função da ULA;
  - O resultado da ULA é escrito no banco de registradores usando os bits 15:11 da instrução para selecionar o registrador de destino ($t1).

Operação do Caminho de Dados



Operação do Caminho de Dados

Linhas de Controle

OpALU

Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0

Operação do Caminho de Dados

Linhas de controle para uma instrução do tipo R:

- Os campos registradores de origem são rs e rt e o campo registrador de destino é rd, isso especifica como os sinais OrigALU e RegDst são definidos;
- Esse tipo de instrução escreve em um registrador (EscreveReg=1), mas não escreve ou lê a memória de dados (LeMem=0, EscreveMem=0);
- Sendo Branch=0, o PC=PC+4;

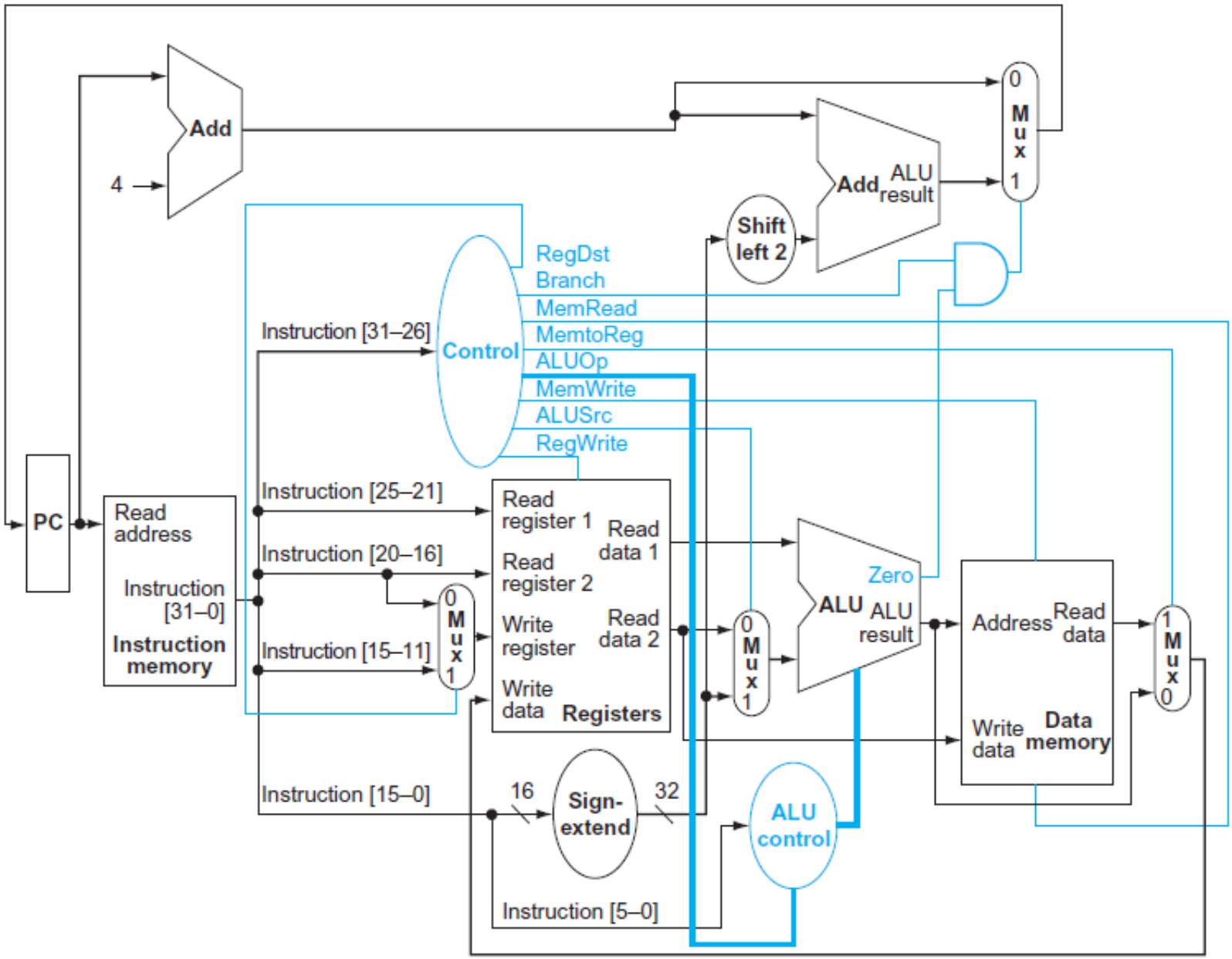
Operação do Caminho de Dados

Como exemplo vejamos a instrução abaixo:

lw $t1, offset($t2)

- A instrução é buscada e o PC é incrementado;
- Um valor do registrador ($t2) é lido do banco de registradores;
- A ULA calcula a soma do valor lido do banco de registradores com os 16 bits menos significativos com sinal estendido da instrução offset;
- A soma da ULA é usada como o endereço para a memória de dados;
- Os dados da unidade de memória são escritos no banco de registradores, o registrador de destino é fornecido pelos bits 20:16 da instrução ($t1).

Operação do Caminho de Dados



Operação do Caminho de Dados

Linhas de Controle

Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
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

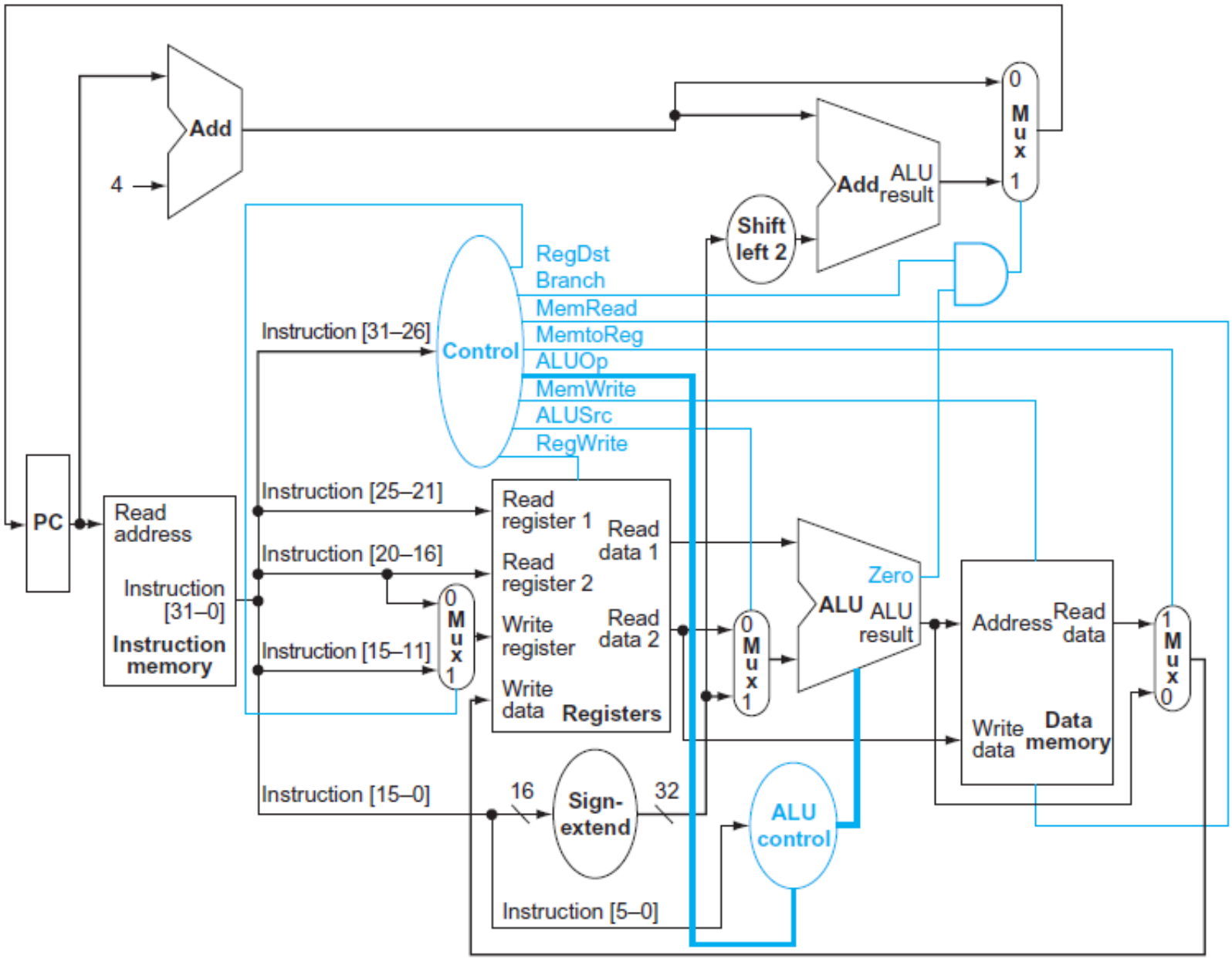
Operação do Caminho de Dados

Como exemplo vejamos a instrução abaixo:

beq $t1,$t2, offset

- A instrucao e buscada e o PC e incrementado;
- Dois registradores, $t1 e t2$, sao lidos do banco de registradores;
- A ULA realiza uma subtracao dos valores de dados lidos do banco de registradores. O valor de PC + 4 e somado aos bits menos significativos com sinal estendido da instrucao offset, deslocados de dois para a esquerda; O resultado e o endereco de destino do desvio;
- O resultado Zero da ULA e usado para decidir o resultado de que somador deve ser armazenado no PC.

Operação do Caminho de Dados

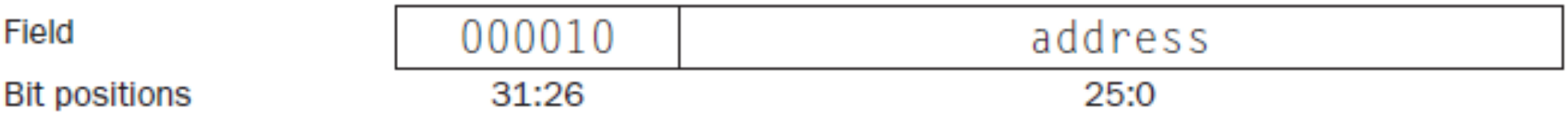


Operação do Caminho de Dados

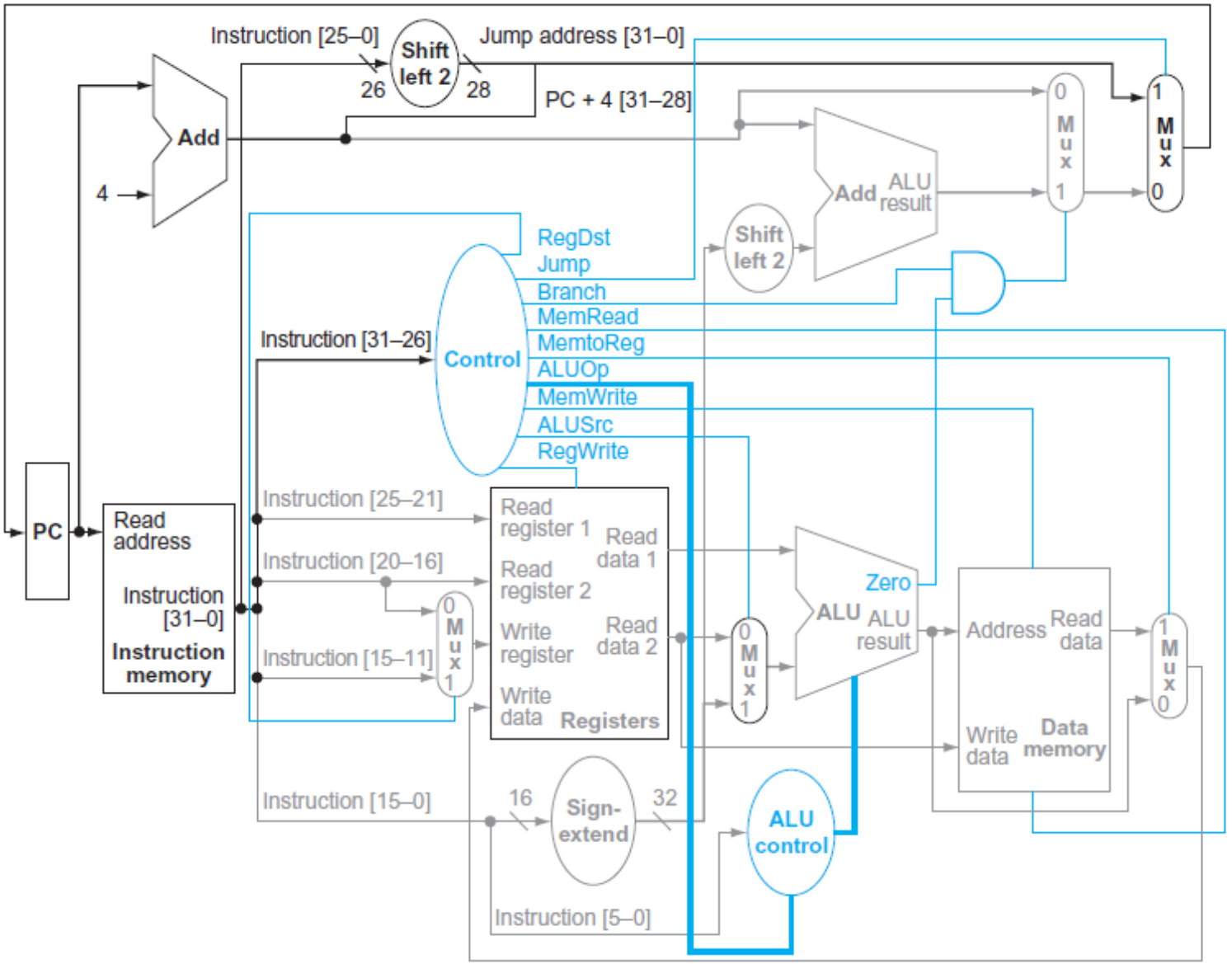
Linhas de Controle

Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
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

Implementando Jumps

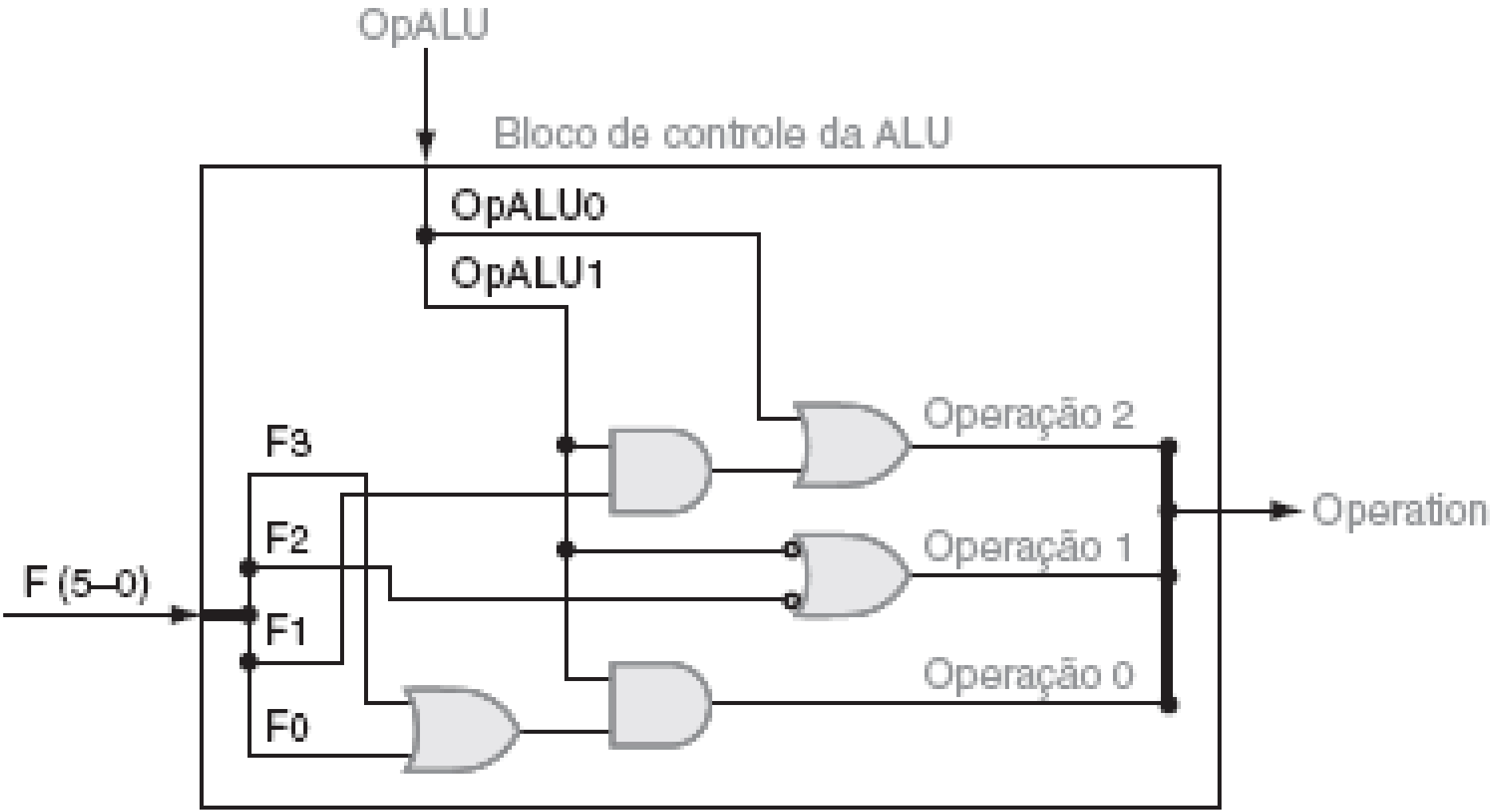


Implementando Jumps



Operação do Caminho de Dados

Detalhes da lógica combinacional do controle da ALU:



Operação do Caminho de Dados

Detalhes da lógica combinacional do controle da ALU: