Universidade Tecnológica Federal do Paraná – UTFPR
Bacharelado em Ciência da Computação

BCC33B – Arquitetura e Organização de Computadores

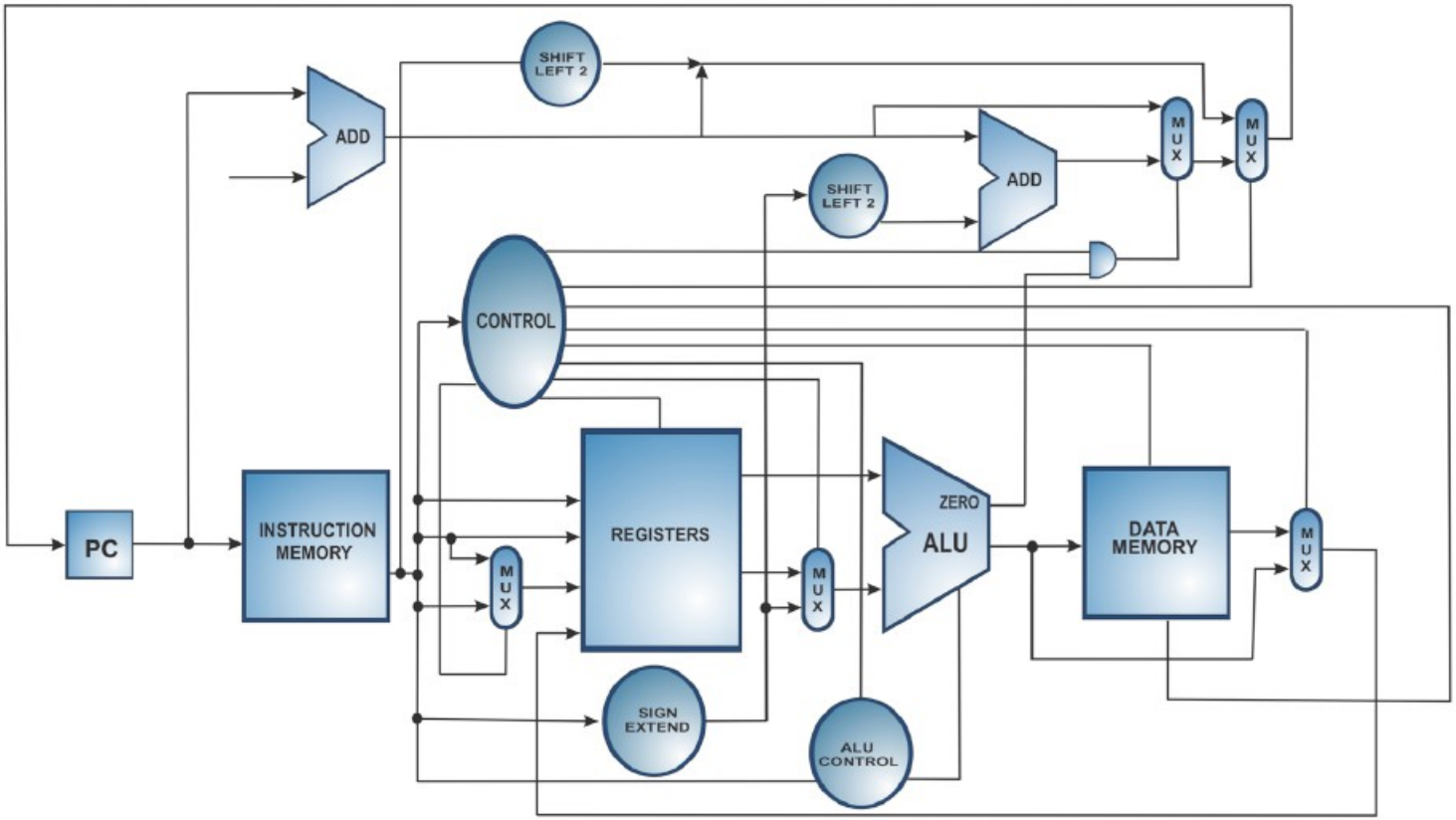
Prof. Paulo Gonçalves

paulogoncalves@utfpr.edu.br

Aula 006

Aula de Hoje

- Datapath



Roteiro

- **Conceito de Datapath**
- **Construção do Datapath da Arquitetura MIPS**
  - (Capítulo 4 do livro texto)
  - **Formatos das Instruções**
    - Instruções Aritméticas e Lógicas
    - Instruções de Referência à Memória
    - Instruções de Desvios
  - **Componentes do Datapath**
  - **Datapath Completo**

Datapath

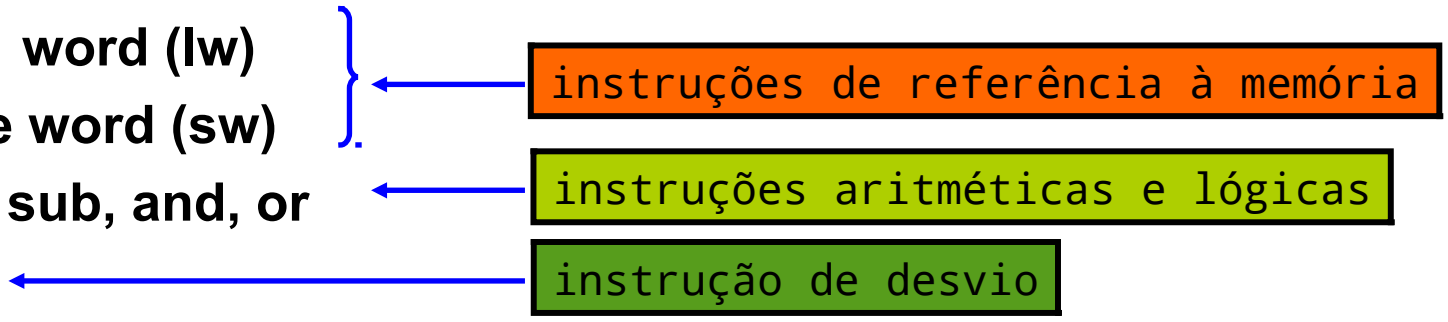
Datapath

Corresponde aos componentes e suas interligações que possibilitam a execução das instruções de um processador

Exemplo de Estudo: Processador MIPS

Para o estudo: selecionamos um grupo básico de instruções que representa todas as demais instruções do processador

- **load word (lw)**
- **store word (sw)**
- **add, sub, and, or**
- **beq**



Datapath

Construção do Datapath

Para determinar o datapath devemos examinar cada um dos componentes necessários à execução de cada uma das classes de instruções do processador

Instruções do MIPS

Formatos das Instruções da Arquitetura MIPS

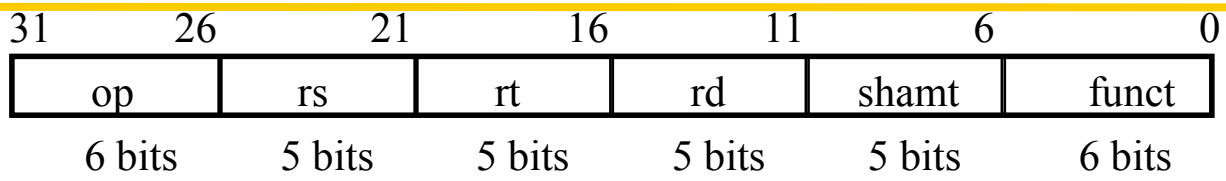
- Instruções de Registradores: Tipo R
- Instruções Imediatas: Tipo I
- Instruções de Desvios: Tipo J

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Instruções do MIPS

Tipo-R (registrador)

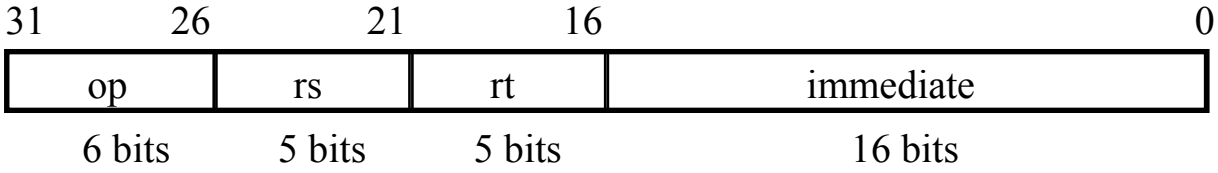
- **add rd, rs, rt**
- **sub, and, or, slt**



1. Lê registradores rs e rt
2. Fornece dados para a ULA
3. Atualiza conjunto de registradores

Tipo-I (imediato)

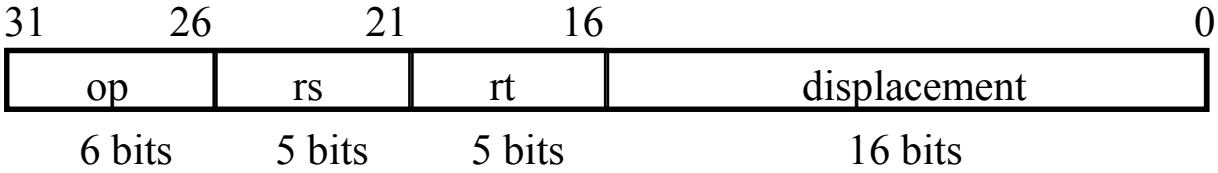
- **lw rt, rs, imm**
- **sw rt, rs, imm**



1. Lê registrador rs (e rt para armazenar)
2. Fornece rs e imediato para a ULA
3. Move dado entre memória e registrador

Tipo-J (jump/desvio)

- **beq rs, rt, imm**



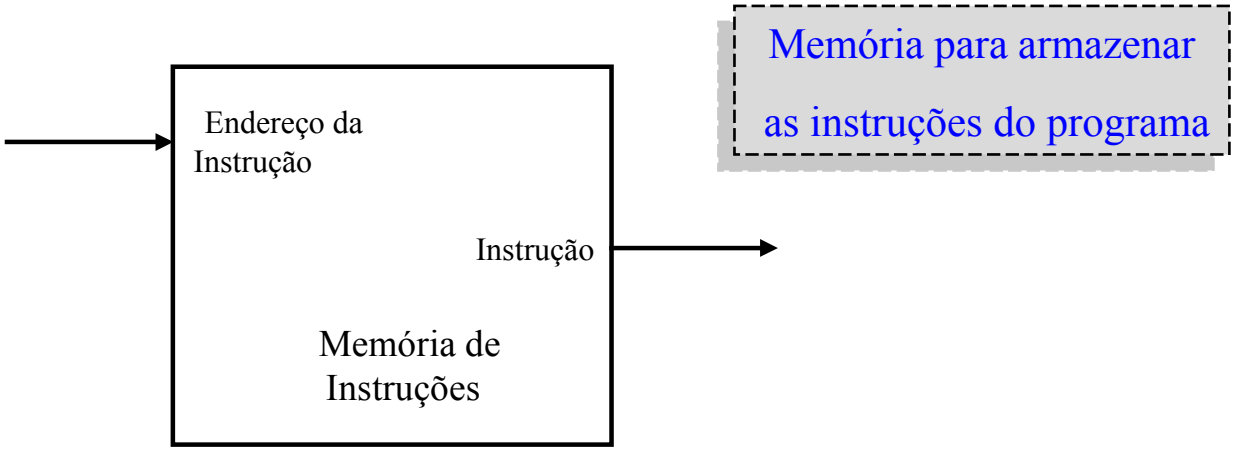
1. Lê registradores rs e rt
2. Fornece-os para a ULA comparar
3. Soma PC ao deslocamento; atualiza PC

Datapath

Simbologia usada no Datapath

Unidades funcionais necessárias para cada instrução:

Busca de Instruções (ação executada para todas as instruções)

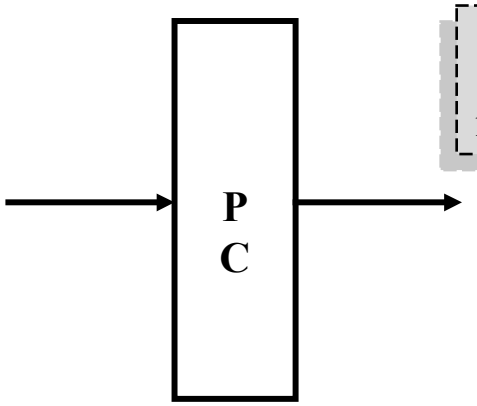


Datapath

Simbologia usada no Datapath

Unidades funcionais necessárias para cada instrução:

Busca de Instruções (ação executada para todas as instruções)



Contador de Programa

PC - Program Counter

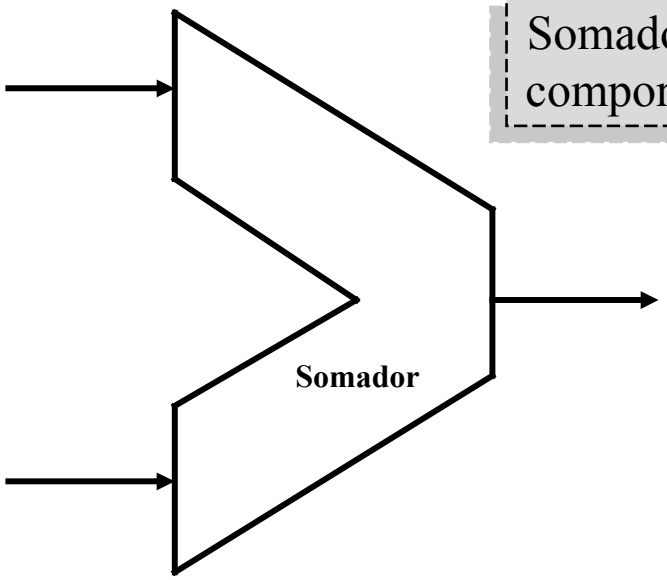
mantém o endereço da próxima instrução

Datapath

Simbologia usada no Datapath

Unidades funcionais necessárias para cada instrução:

Busca de Instruções (ação executada para todas as instruções)



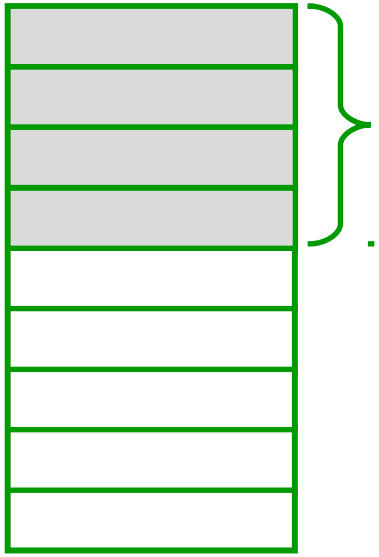
Somador incrementa o valor do PC para compor o endereço da próxima instrução

Datapath

Visão Lógica da Memória no MIPS

Palavras de 4 bytes posicionadas sequencialmente

Memória no MIPS



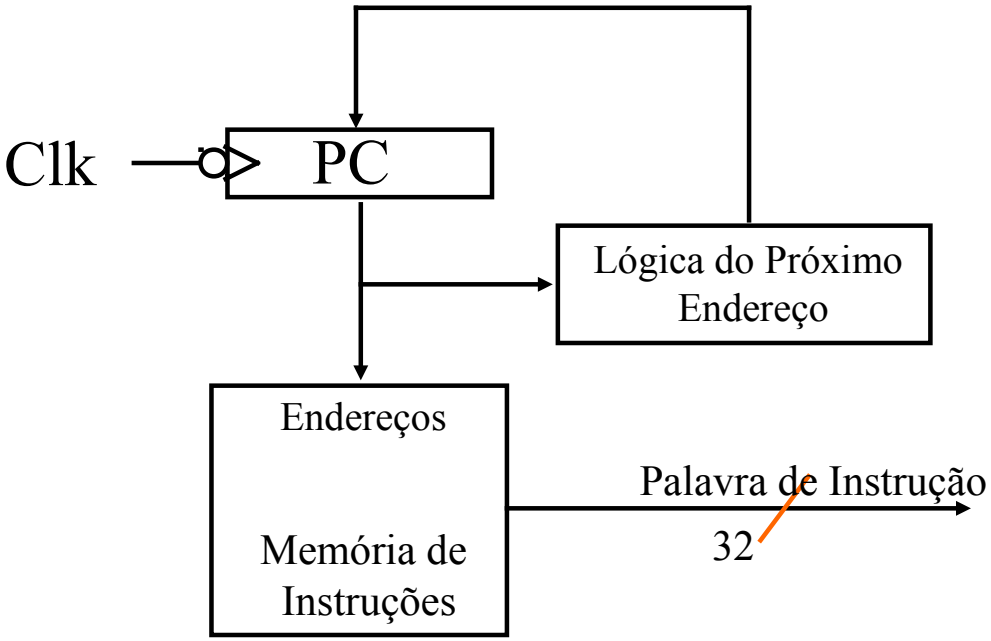
Palavras de

4 bytes

Datapath

Datapath

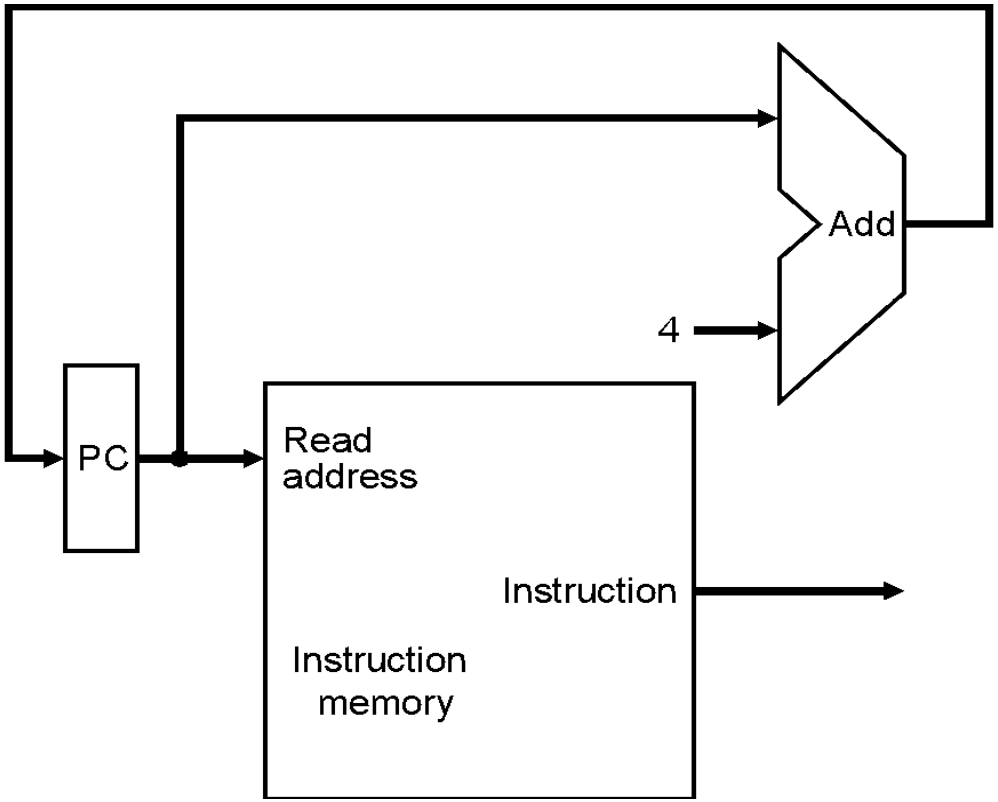
Parte do Datapath para Busca de Instruções



Datapath

Datapath

Parte do Datapath para Busca de Instruções



Datapath

Unidades funcionais necessárias para operação da ULA (instruções do tipo Registrador):

Banco de Registradores (Register File) e a própria ULA

Todas as instruções do Tipo-R precisam:

- Ler 2 registradores
- Realizar a operação na ULA
- Escrever o resultado num registrador

Escrita precisa:

- Especificar o número do registrador a ser escrito
- Entrada do dado a ser escrito

Leitura precisa:

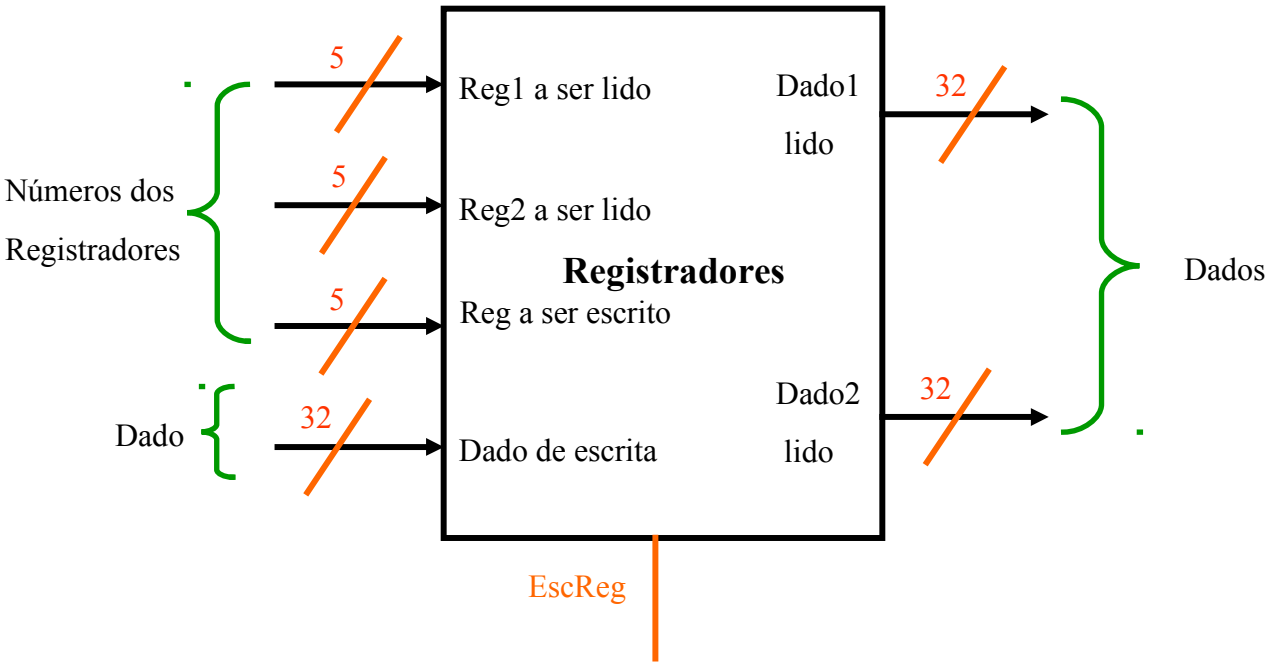
- Especificar o número do registrador a ser lido
- Saída para o registrador lido

Datapath

Simbologia usada no Datapath

Unidades funcionais necessárias para operação da ULA (instruções do tipo Registrador):

Banco de Registradores (Register File) e a própria ULA

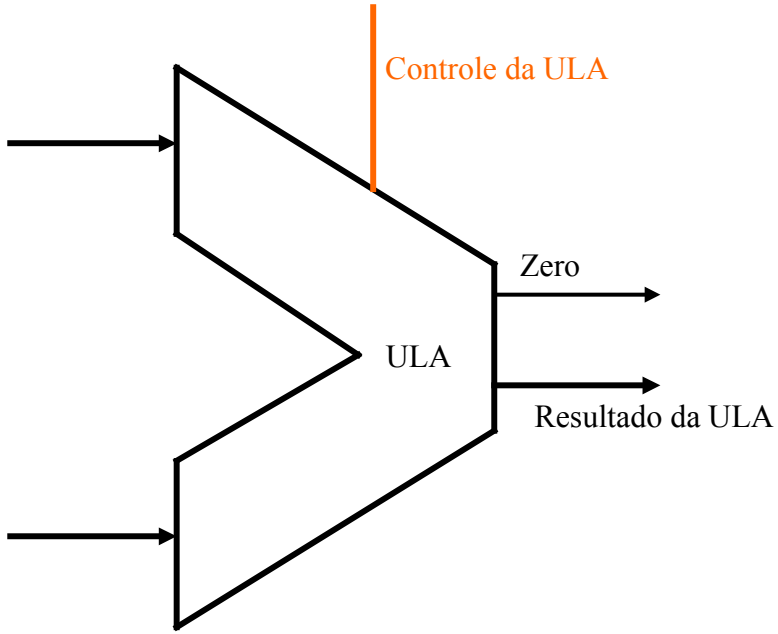


Datapath

Simbologia usada no Datapath

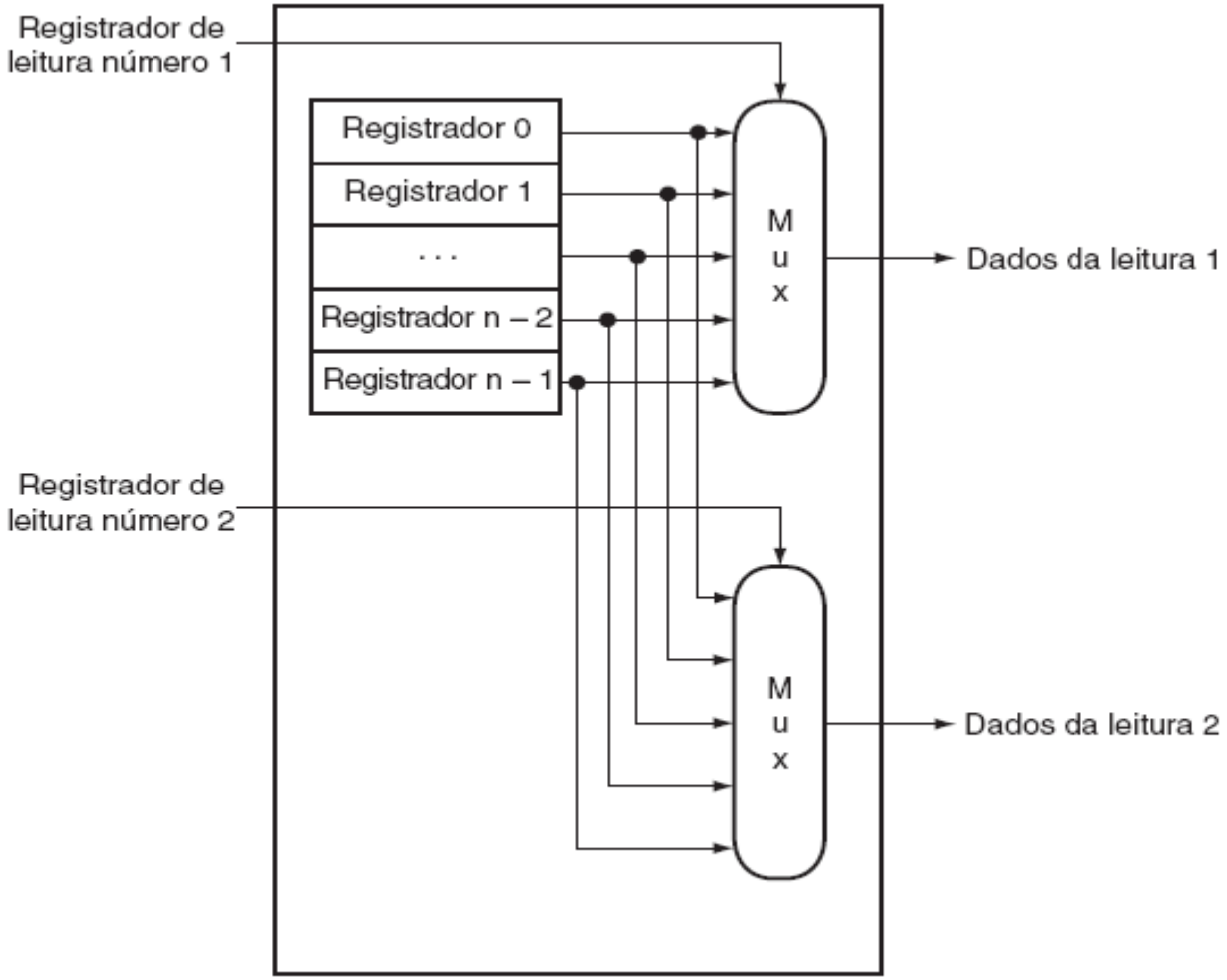
Unidades funcionais necessárias para operação da ULA (instruções do tipo Registrador):

Banco de Registradores (Register File) e a própria ULA



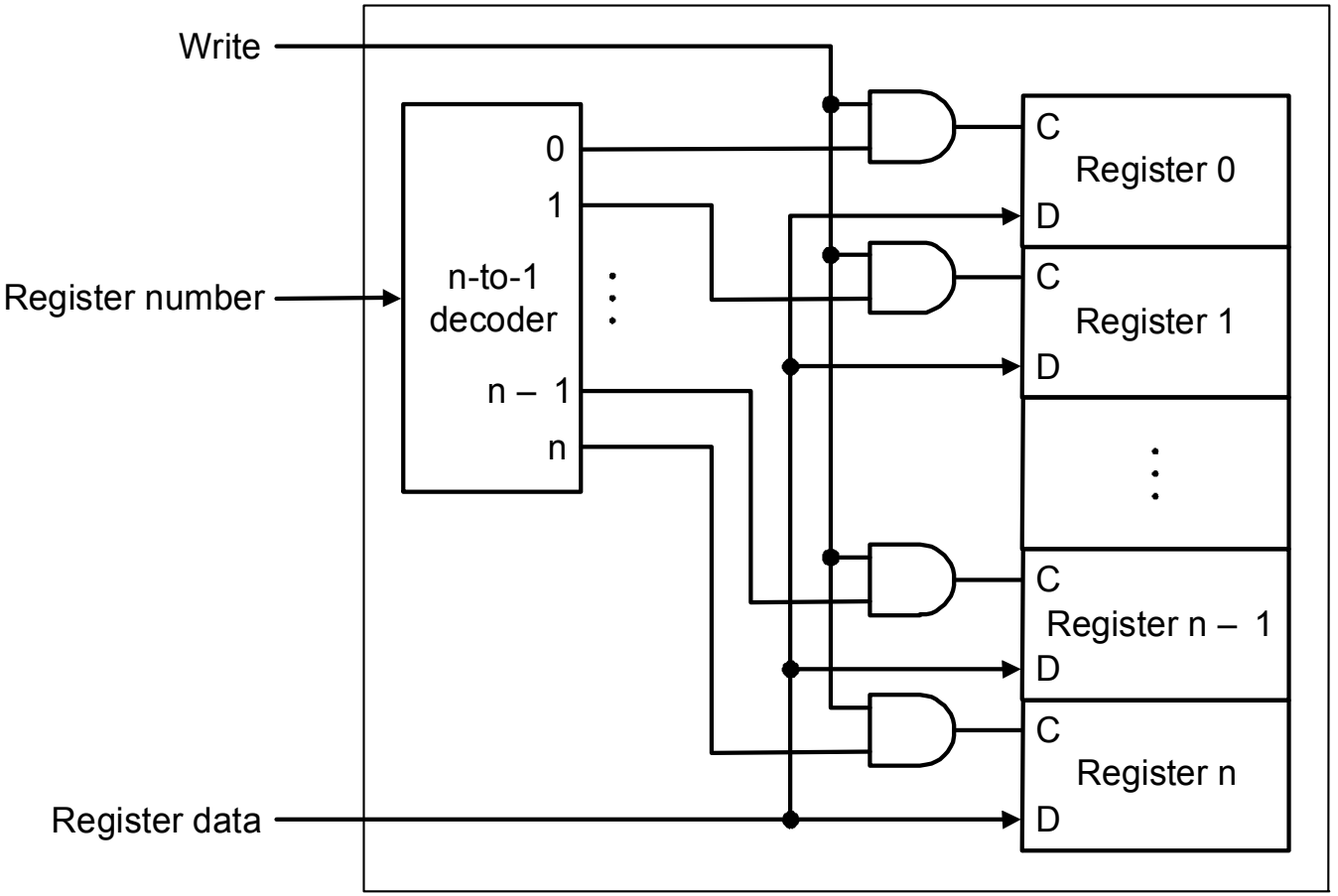
Datapath

Estrutura do Banco de Registradores para Leitura



Datapath

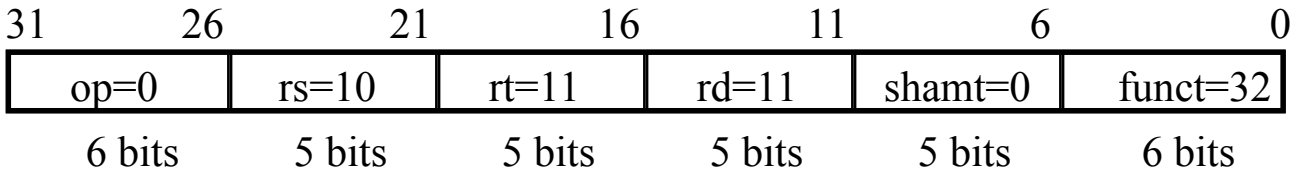
Estrutura do Banco de Registradores para Escrita

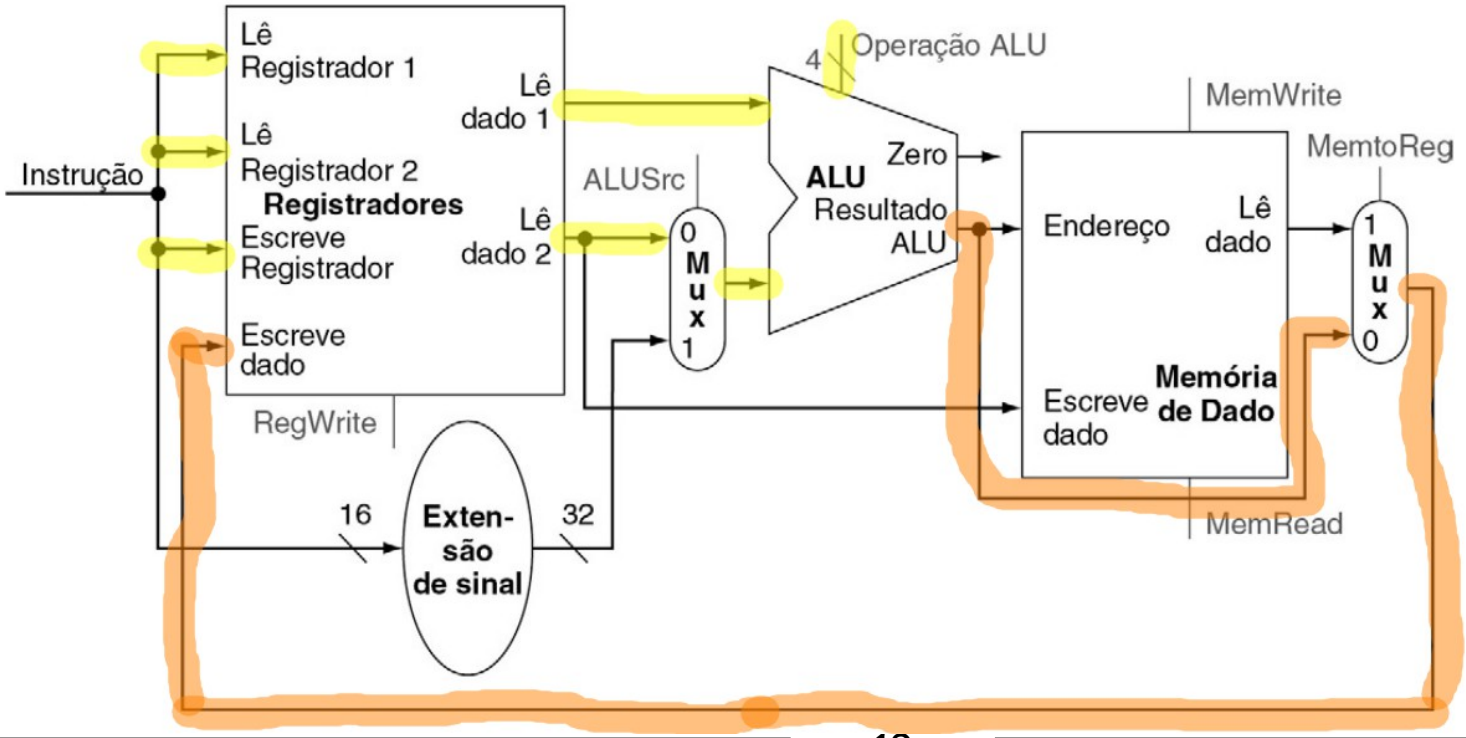


Datapath

Parte do Datapath para Instruções do Tipo Registrador

add rd, rs, rt **Exemplo: *add* *$t1, $t2, $t3***





Datapath

Unidades funcionais necessárias para as instruções de load e store:

Memória de Dados e Unidade de Extensão de Sinal

Todas as instruções do Tipo-I precisam:

- Ler 1 registrador base para calcular o endereço do operando
- Somar o conteúdo do registrador base ao endereço de 16 bits na instrução

Instrução de Store:

- Especificar o número do registrador a ser lido
- Escrever o conteúdo do registrador lido na posição de memória calculada

Instrução de Load:

- Ler o conteúdo da posição de memória no endereço calculado
- Escrever o conteúdo da posição de memória no registrador especificado

Elementos necessários:

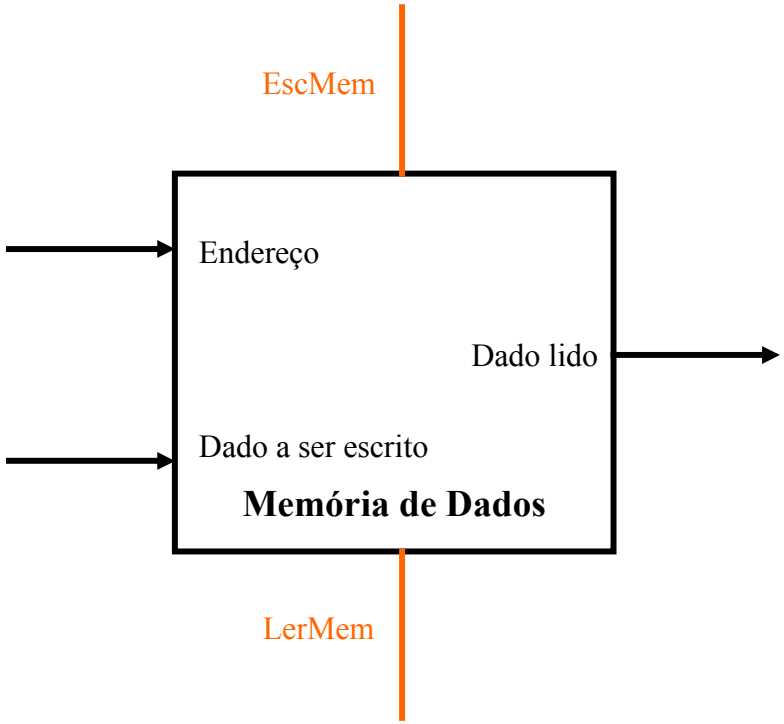
- Todas as instruções usam a ULA, o banco de registradores e a memória de dados

Datapath

Simbologia usada no Datapath

Unidades funcionais necessárias para as instruções de load e store:

Memória de Dados e Unidade de Extensão de Sinal



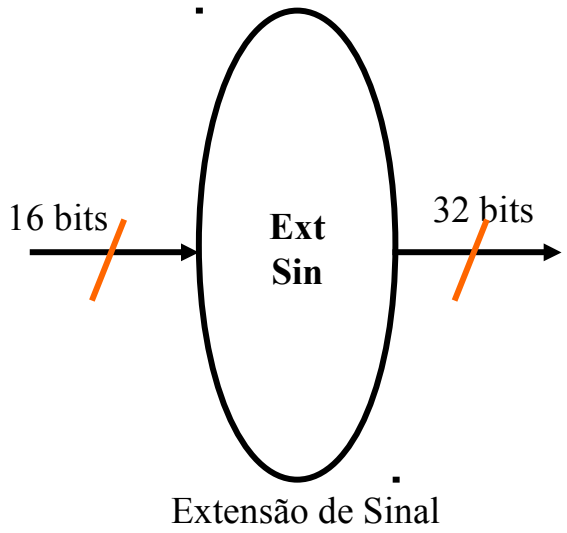
A memória precisa ter sinais de controle para leitura e escrita, um endereço de entrada, além de uma entrada para dados e uma saída para dados

Datapath

Simbologia usada no Datapath

Unidades funcionais necessárias para as instruções de load e store:

Memória de Dados e Unidade de Extensão de Sinal



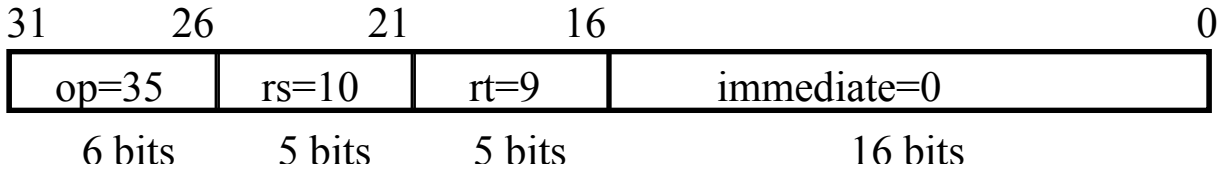
Unidade de Extensão de Sinal:

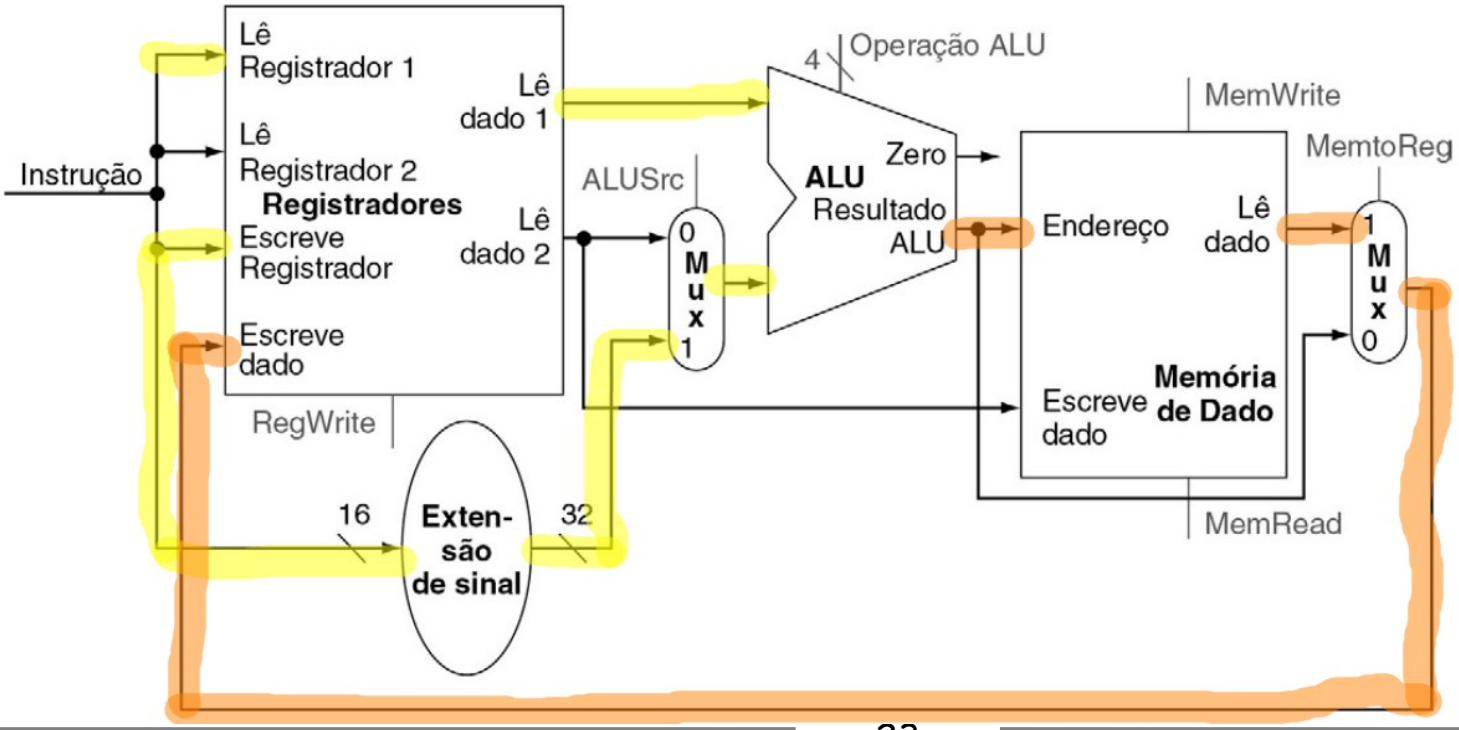
Tem entrada de 16 bits e estende o sinal para 32 bits para calcular o endereço do operando

Datapath

Parte do Datapath para Instruções do Tipo Load

lw rt, imm16(rs) **Exemplo: *lw $t1, 0($t2)***

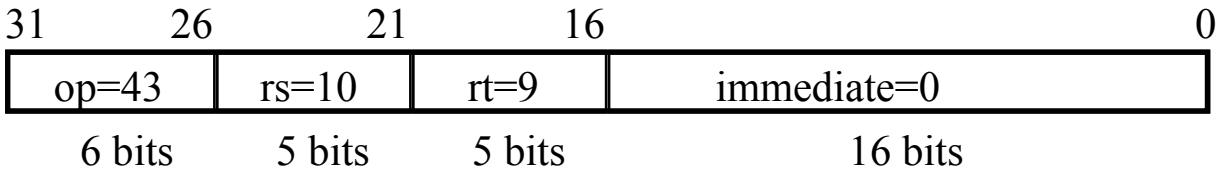


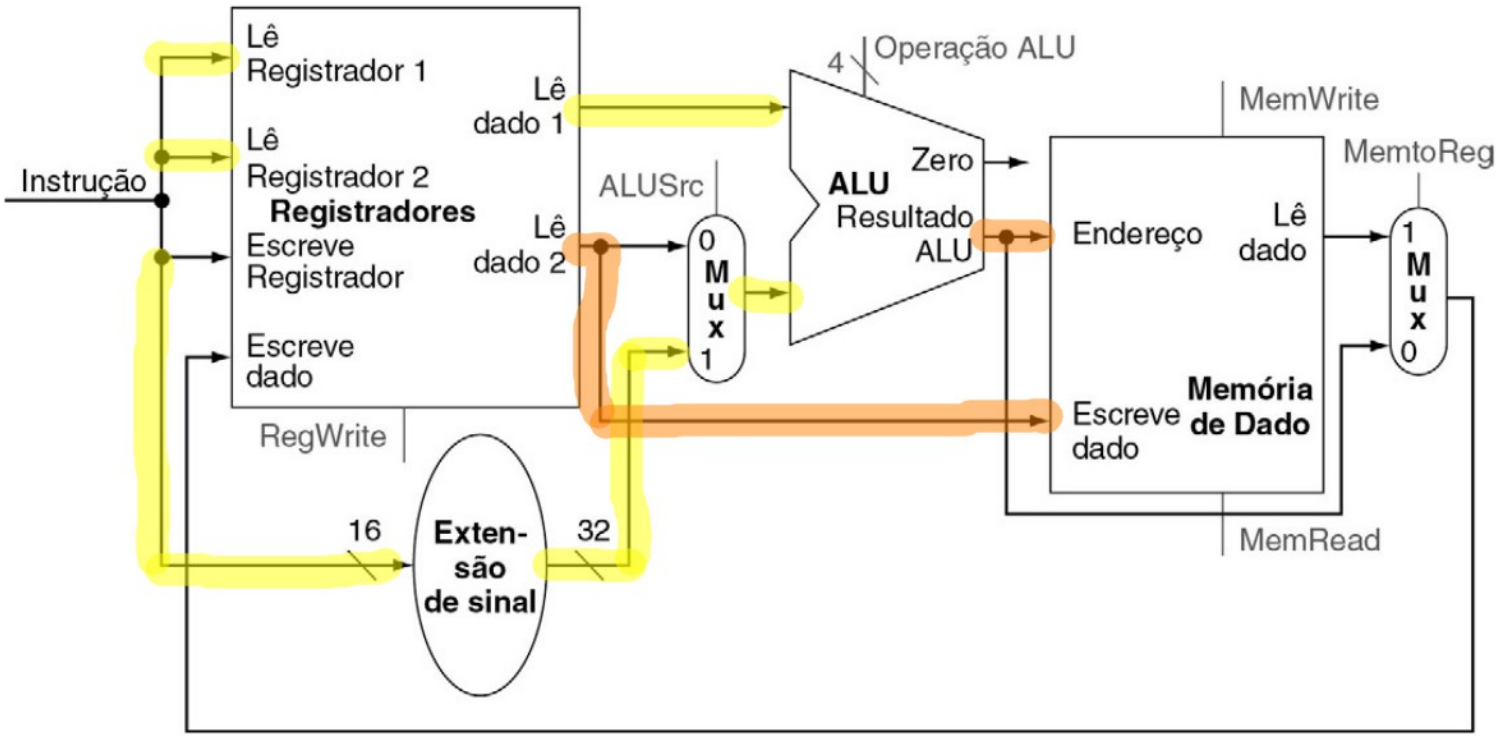


Datapath

Parte do Datapath para Instruções do Tipo Store

sw rt, imm16(rs) **Exemplo: s*w $t1, 0($t2)***





Datapath

Unidades funcionais necessárias para as instruções de branch:

Registradores, ULA, Unidade de Extensão de Sinal, Unidade de Deslocamento de bits

Instruções de Desvios Condicionais precisam:

Ler 2 registradores para testar o desvio

Somar o conteúdo do PC ao endereço-alvo do desvio (16 bits na instrução)

Deslocar 2 bits à esquerda do endereço-alvo do desvio para endereçar a palavra por um fator de 4

Se o desvio é tomado: PC ⇐ endereço-alvo do desvio

- Se o desvio é não-tomado: PC ⇐ PC + 4

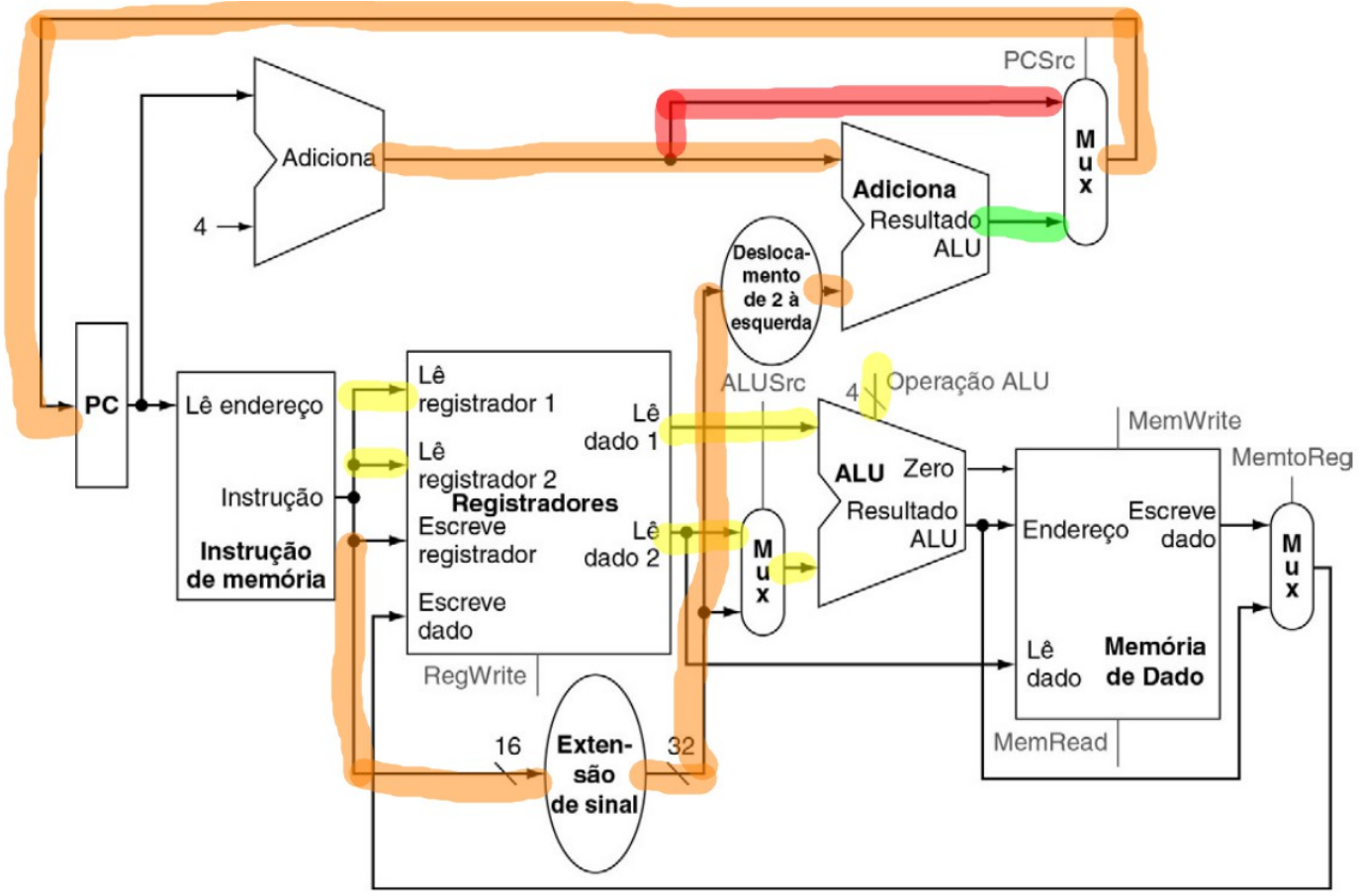
Elementos necessários:

ULA para comparar os registradores, Somador para calcular endereço-alvo e o banco de registradores

Datapath

Parte do Datapath para Instruções do Tipo Branch

beq rs, rt, imm16 **Exemplo: beq $t1, $t2, offset**



Datapath

Unidades funcionais necessárias para as instruções de jump:

Registradores, ULA, Unidade de Extensão de Sinal, Unidade de Deslocamento de bits

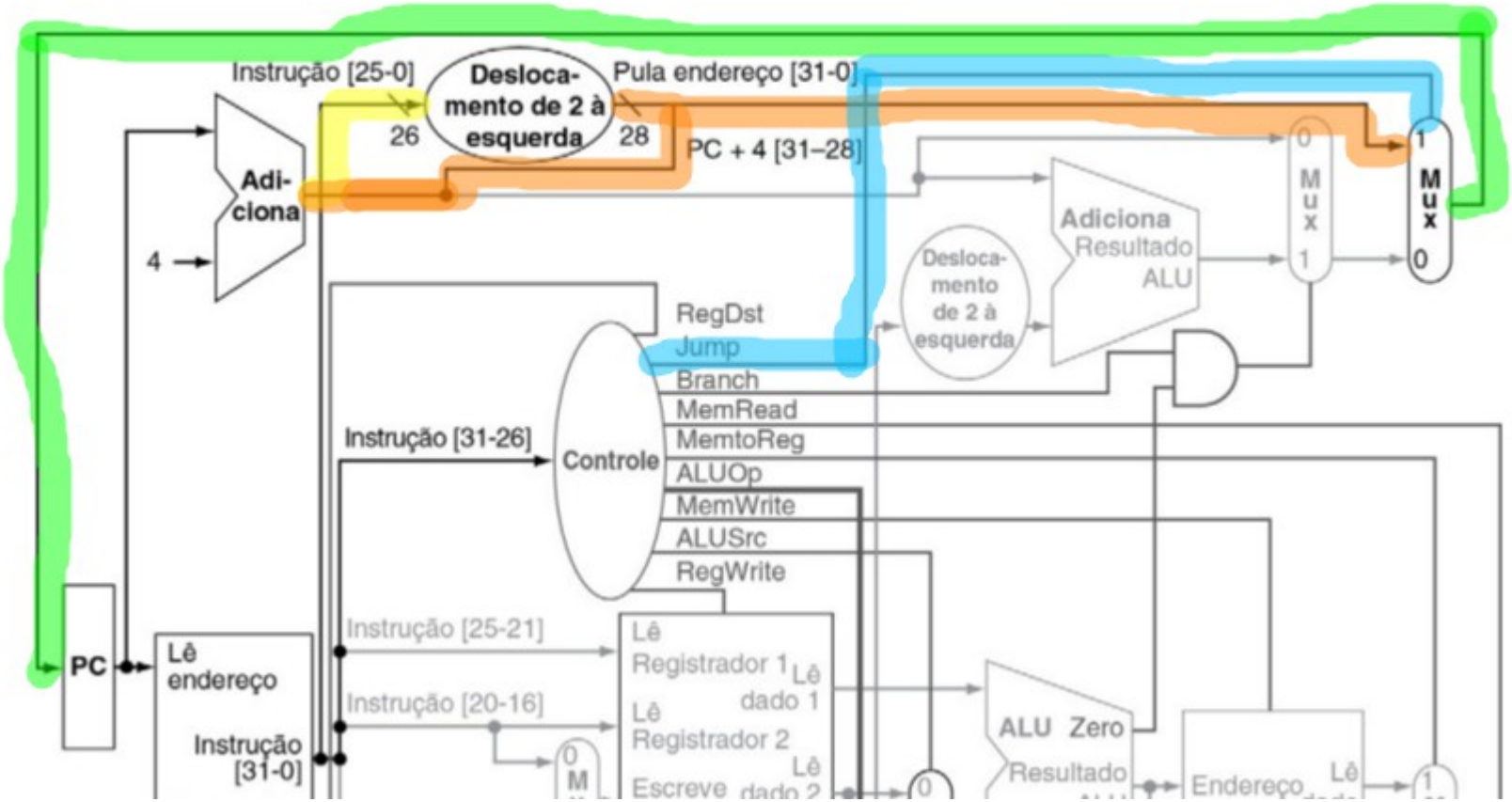
Instruções jump precisam:

- Os 4 bits superiores do PC atual + 4 (esses são bits 31:28 do endereço da instrução imediatamente seguinte.
- O campo de 26 bits imediato da instrução jump
- os bits 00_2
- O endereço de destino do jump é obtido deslocando-se os 26 bits inferiores da instrução jump de 2 bits para a esquerda, efetivamente adicionando 00 como os bits menos significativos
- Depois concatenando os 4 bits mais significativos dos PC + 4 com os bits mais significativos, produzindo um endereço de 32 bits.

Datapath

Parte do Datapath para Instruções do Tipo jump

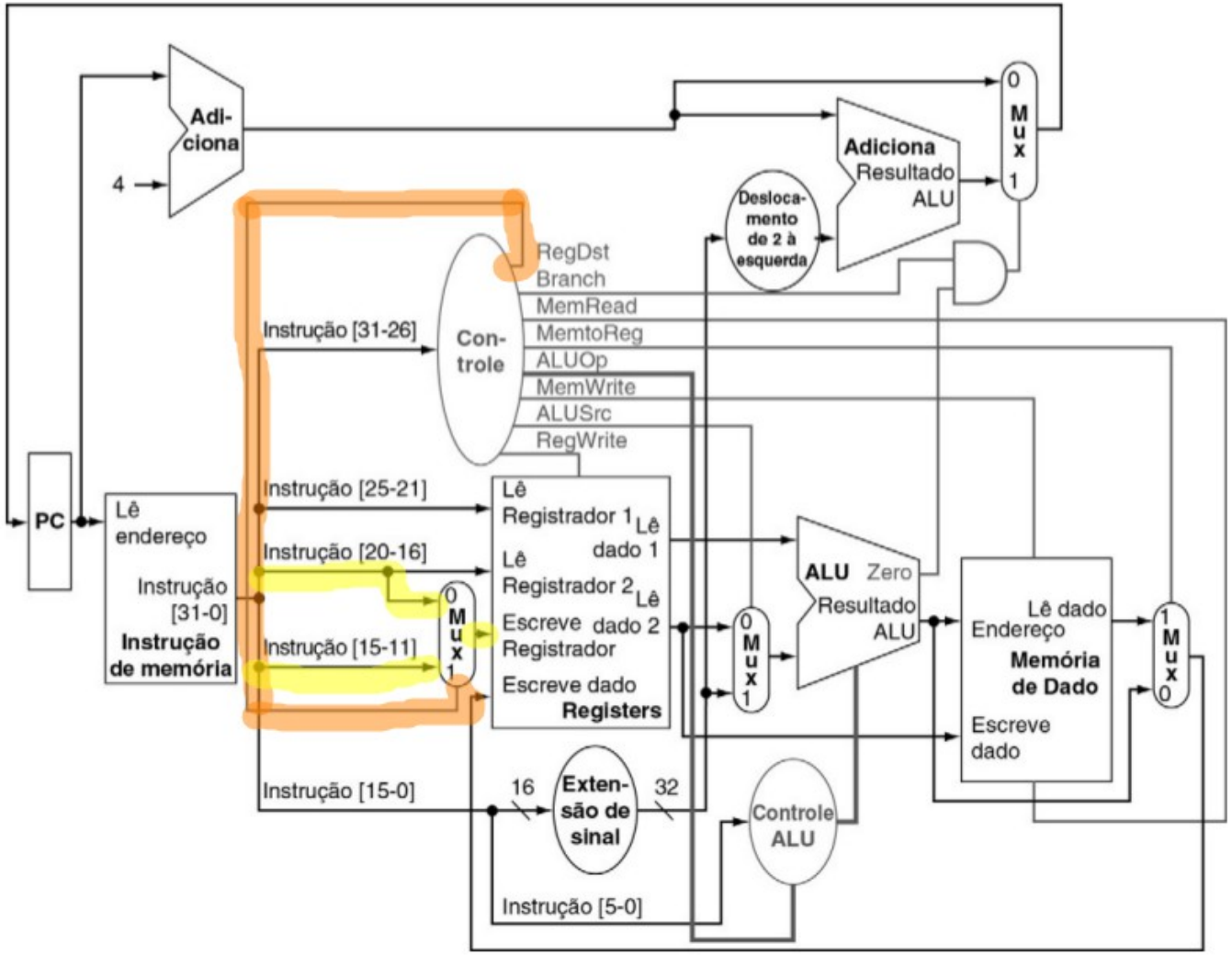
jump imm26



Sinais Controle - RegDst

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador destino para entrada Registrador para escrita vem do campo rt (bits 20:16).	O número do registrador destino para a entrada Registrador para escrita vem do campo rd (bits 15:11).
EscreveReg	Nenhum.	O registrador na entrada Registrador para escrita é escrito com o valor da entrada Dados para escrita.
OrigALU	O segundo operando da ALU vem da segunda saída do banco de registradores (Dados da leitura 2).	O segundo operando da ALU consiste nos 16 bits mais baixos da instrução com sinal estendido.
OrigPC	O PC é substituído pela saída do somador que calcula o valor de PC + 4.	O PC é substituído pela saída do somador que calcula o destino do desvio.
LeMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é colocado na saída Dados da leitura.
EscreveMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é substituído pelo valor na entrada Dados para escrita.
MemparaReg	O valor enviado à entrada Dados para escrita do banco de registradores vem da ALU.	O valor enviado à entrada Dados para escrita do banco de registradores vem da memória de dados.

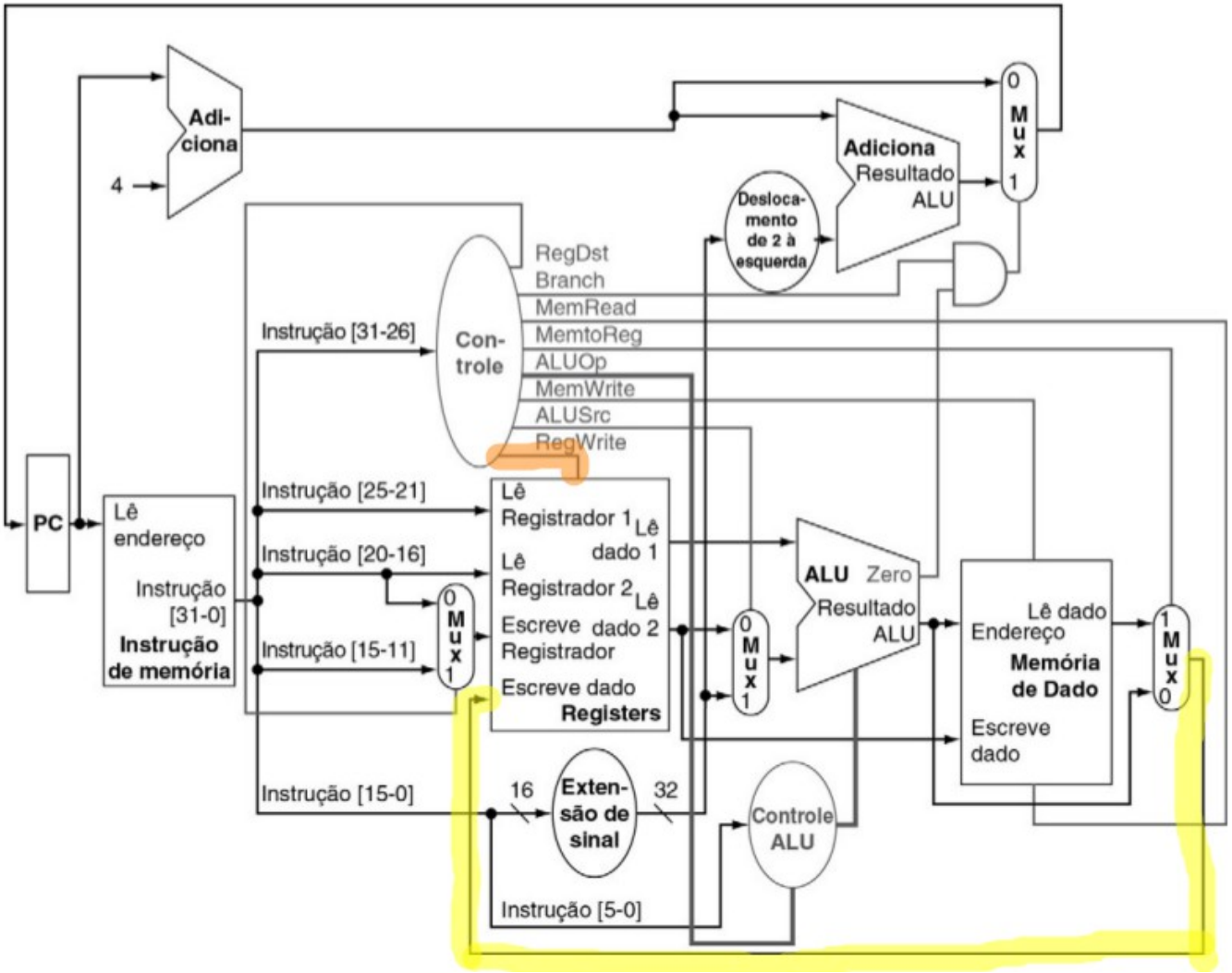
Sinais Controle - RegDst



Sinais Controle – EscreveReg (RegWrite)

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador destino para entrada Registrador para escrita vem do campo rt (bits 20:16).	O número do registrador destino para a entrada Registrador para escrita vem do campo rd (bits 15:11).
EscreveReg	Nenhum.	O registrador na entrada Registrador para escrita é escrito com o valor da entrada Dados para escrita.
OrigALU	O segundo operando da ALU vem da segunda saída do banco de registradores (Dados da leitura 2).	O segundo operando da ALU consiste nos 16 bits mais baixos da instrução com sinal estendido.
OrigPC	O PC é substituído pela saída do somador que calcula o valor de PC + 4.	O PC é substituído pela saída do somador que calcula o destino do desvio.
LeMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é colocado na saída Dados da leitura.
EscreveMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é substituído pelo valor na entrada Dados para escrita.
MemparaReg	O valor enviado à entrada Dados para escrita do banco de registradores vem da ALU.	O valor enviado à entrada Dados para escrita do banco de registradores vem da memória de dados.

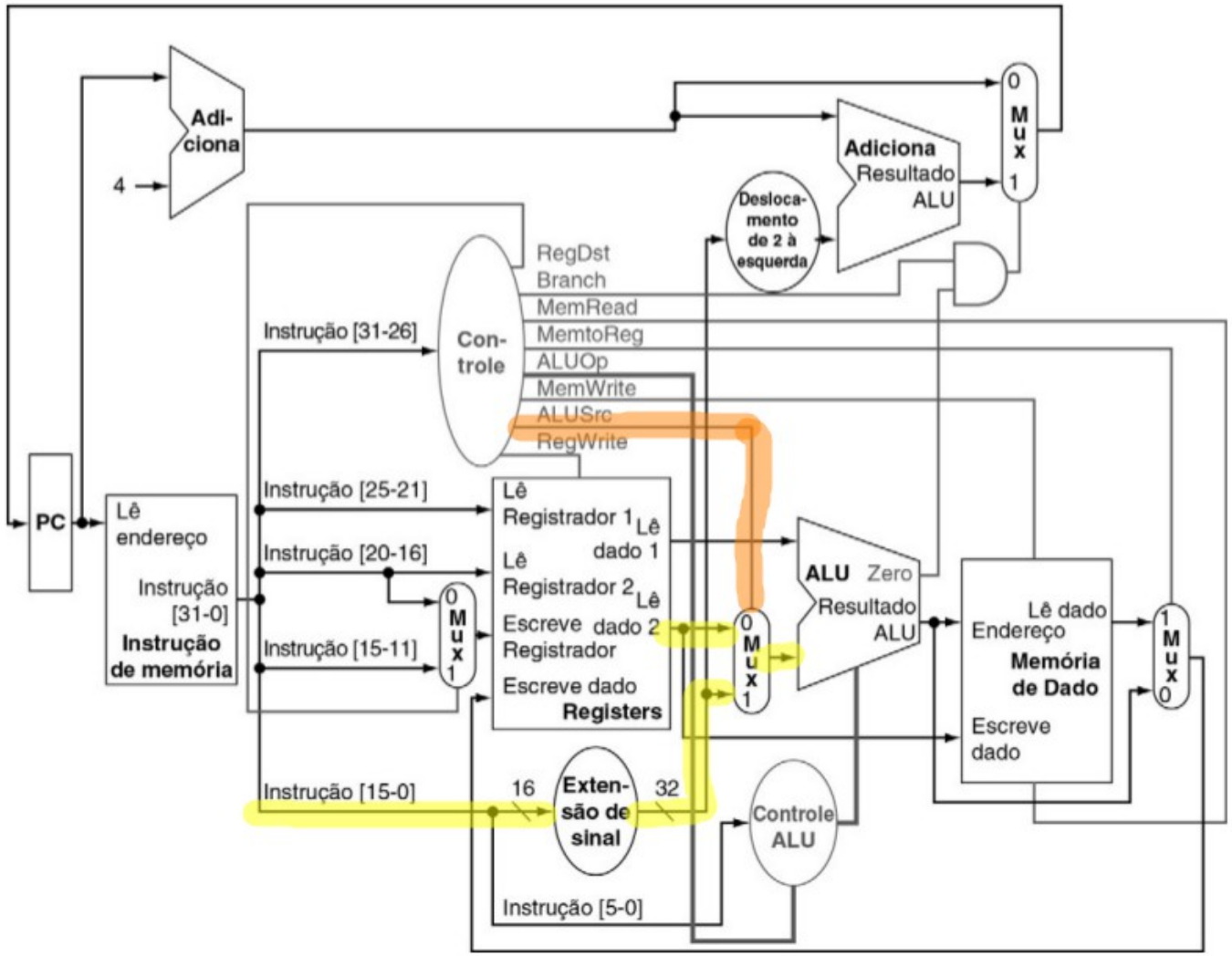
Sinais Controle – EscreveReg (RegWrite)



Sinais Controle – OrigALU (ALUSrc)

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador destino para entrada Registrador para escrita vem do campo rt (bits 20:16).	O número do registrador destino para a entrada Registrador para escrita vem do campo rd (bits 15:11).
EscreveReg	Nenhum.	O registrador na entrada Registrador para escrita é escrito com o valor da entrada Dados para escrita.
OrigALU	O segundo operando da ALU vem da segunda saída do banco de registradores (Dados da leitura 2).	O segundo operando da ALU consiste nos 16 bits mais baixos da instrução com sinal estendido.
OrigPC	O PC é substituído pela saída do somador que calcula o valor de PC + 4.	O PC é substituído pela saída do somador que calcula o destino do desvio.
LeMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é colocado na saída Dados da leitura.
EscreveMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é substituído pelo valor na entrada Dados para escrita.
MemparaReg	O valor enviado à entrada Dados para escrita do banco de registradores vem da ALU.	O valor enviado à entrada Dados para escrita do banco de registradores vem da memória de dados.

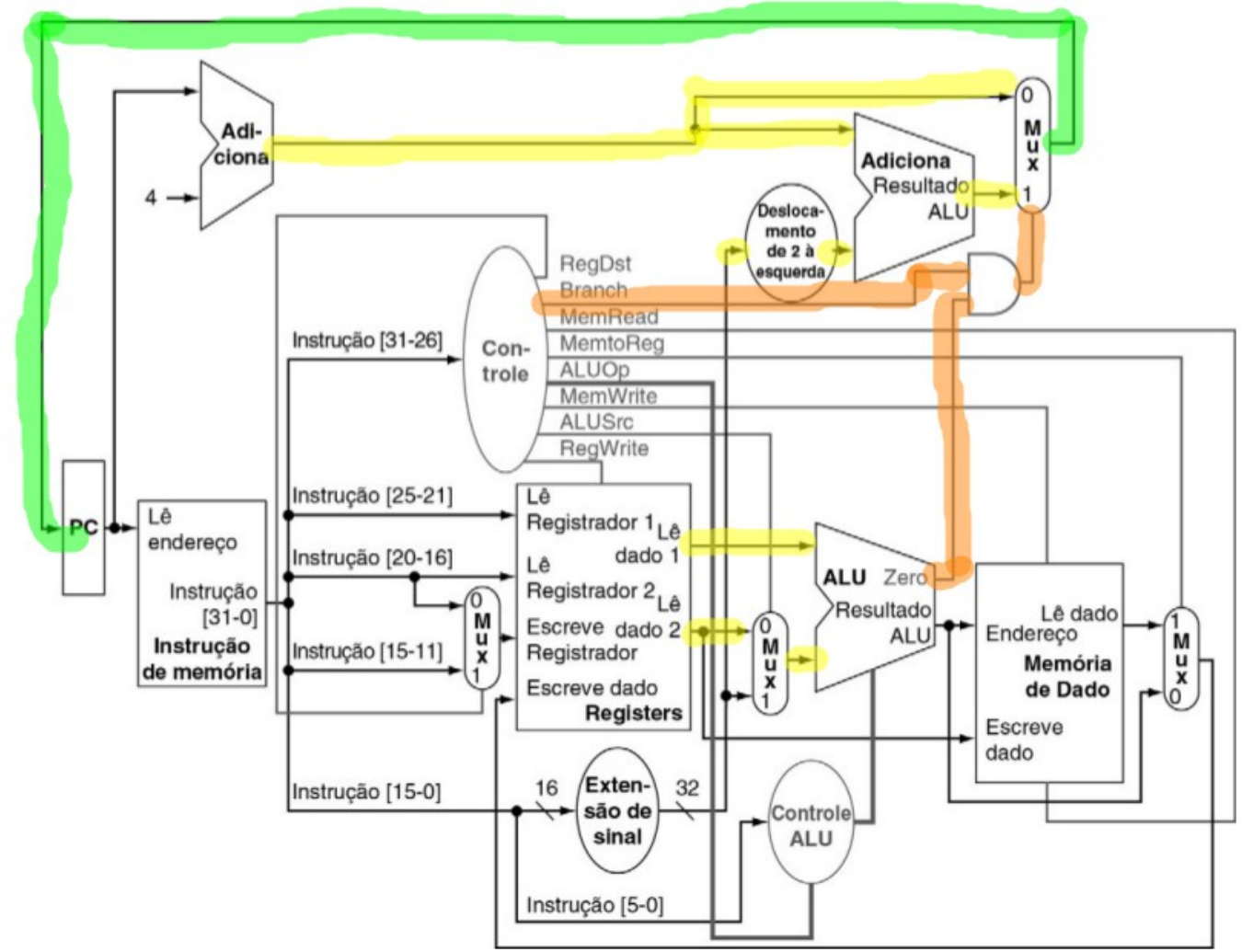
Sinais Controle – OrigALU (ALUSrc)



Sinais Controle – OrigPC (Branch & Zero)

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador destino para entrada Registrador para escrita vem do campo rt (bits 20:16).	O número do registrador destino para a entrada Registrador para escrita vem do campo rd (bits 15:11).
EscreveReg	Nenhum.	O registrador na entrada Registrador para escrita é escrito com o valor da entrada Dados para escrita.
OrigALU	O segundo operando da ALU vem da segunda saída do banco de registradores (Dados da leitura 2).	O segundo operando da ALU consiste nos 16 bits mais baixos da instrução com sinal estendido.
OrigPC	O PC é substituído pela saída do somador que calcula o valor de PC + 4.	O PC é substituído pela saída do somador que calcula o destino do desvio.
LeMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é colocado na saída Dados da leitura.
EscreveMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é substituído pelo valor na entrada Dados para escrita.
MemparaReg	O valor enviado à entrada Dados para escrita do banco de registradores vem da ALU.	O valor enviado à entrada Dados para escrita do banco de registradores vem da memória de dados.

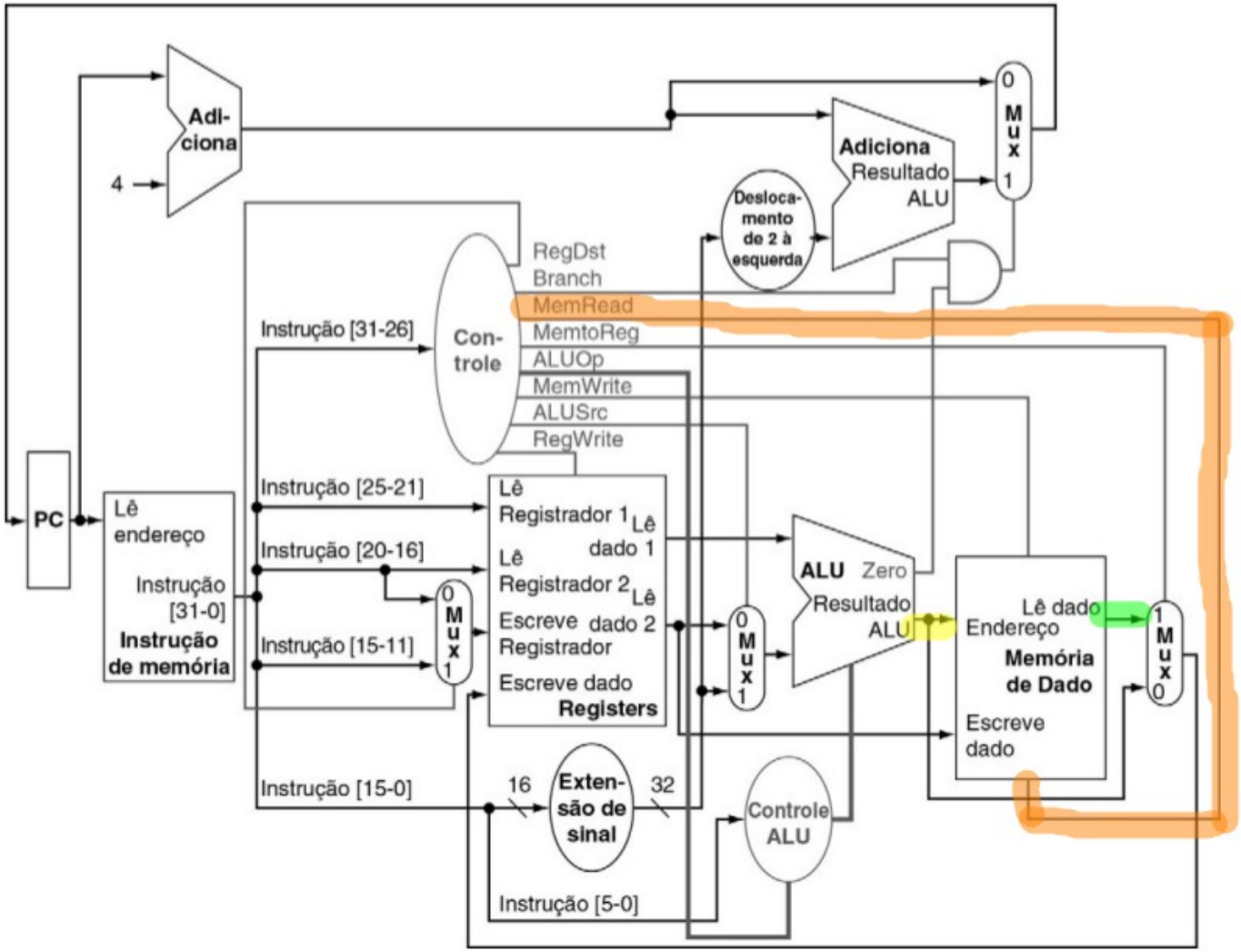
Sinais Controle – OrigPC (Branch & Zero)



Sinais Controle – LeMem (MemRead)

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador destino para entrada Registrador para escrita vem do campo rt (bits 20:16).	O número do registrador destino para a entrada Registrador para escrita vem do campo rd (bits 15:11).
EscreveReg	Nenhum.	O registrador na entrada Registrador para escrita é escrito com o valor da entrada Dados para escrita.
OrigALU	O segundo operando da ALU vem da segunda saída do banco de registradores (Dados da leitura 2).	O segundo operando da ALU consiste nos 16 bits mais baixos da instrução com sinal estendido.
OrigPC	O PC é substituído pela saída do somador que calcula o valor de PC + 4.	O PC é substituído pela saída do somador que calcula o destino do desvio.
LeMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é colocado na saída Dados da leitura.
EscreveMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é substituído pelo valor na entrada Dados para escrita.
MemparaReg	O valor enviado à entrada Dados para escrita do banco de registradores vem da ALU.	O valor enviado à entrada Dados para escrita do banco de registradores vem da memória de dados.

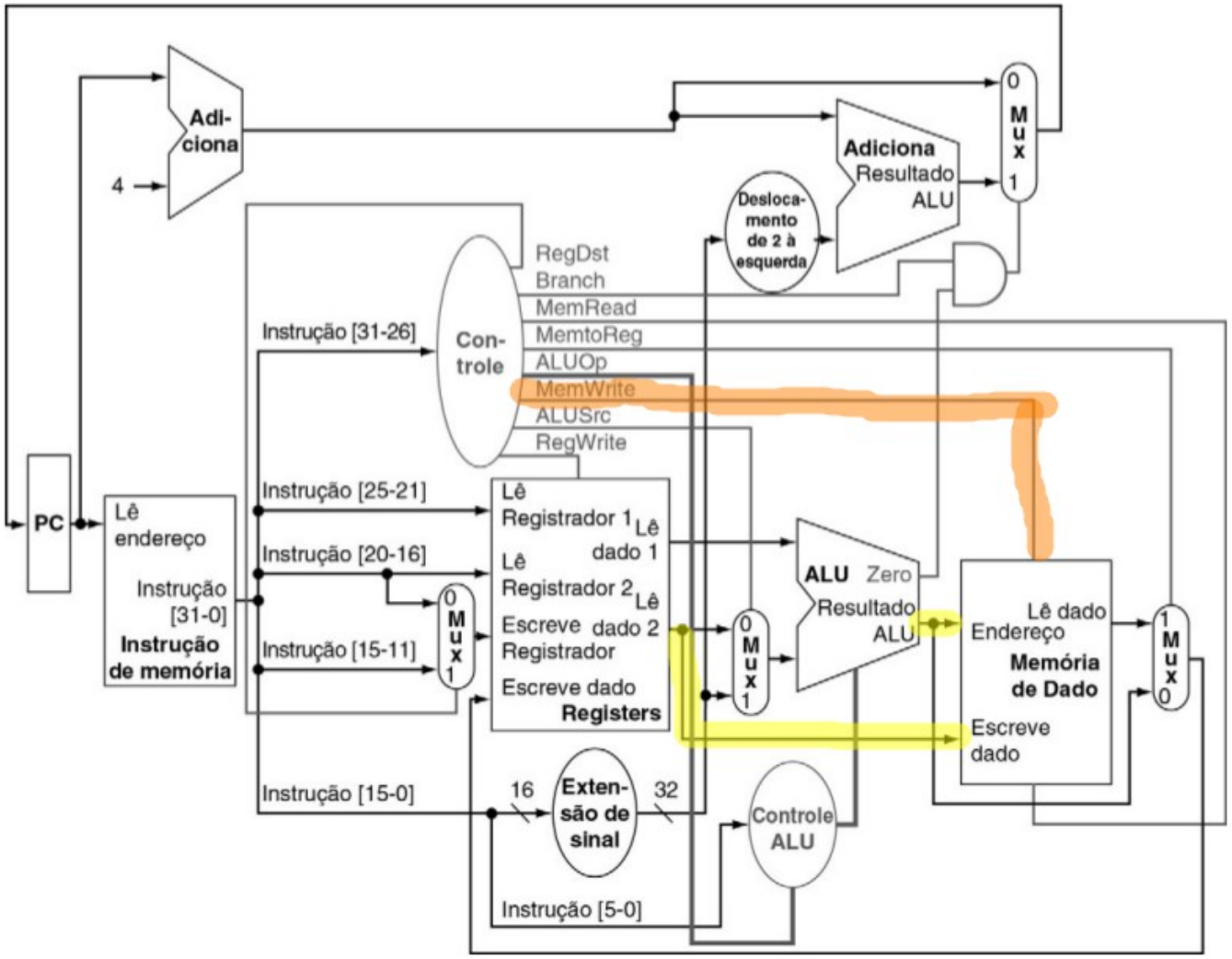
Sinais Controle – LeMem (MemRead)



Sinais Controle – EscreveMem (MemWrite)

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador destino para entrada Registrador para escrita vem do campo rt (bits 20:16).	O número do registrador destino para a entrada Registrador para escrita vem do campo rd (bits 15:11).
EscreveReg	Nenhum.	O registrador na entrada Registrador para escrita é escrito com o valor da entrada Dados para escrita.
OrigALU	O segundo operando da ALU vem da segunda saída do banco de registradores (Dados da leitura 2).	O segundo operando da ALU consiste nos 16 bits mais baixos da instrução com sinal estendido.
OrigPC	O PC é substituído pela saída do somador que calcula o valor de PC + 4.	O PC é substituído pela saída do somador que calcula o destino do desvio.
LeMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é colocado na saída Dados da leitura.
EscreveMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é substituído pelo valor na entrada Dados para escrita.
MemparaReg	O valor enviado à entrada Dados para escrita do banco de registradores vem da ALU.	O valor enviado à entrada Dados para escrita do banco de registradores vem da memória de dados.

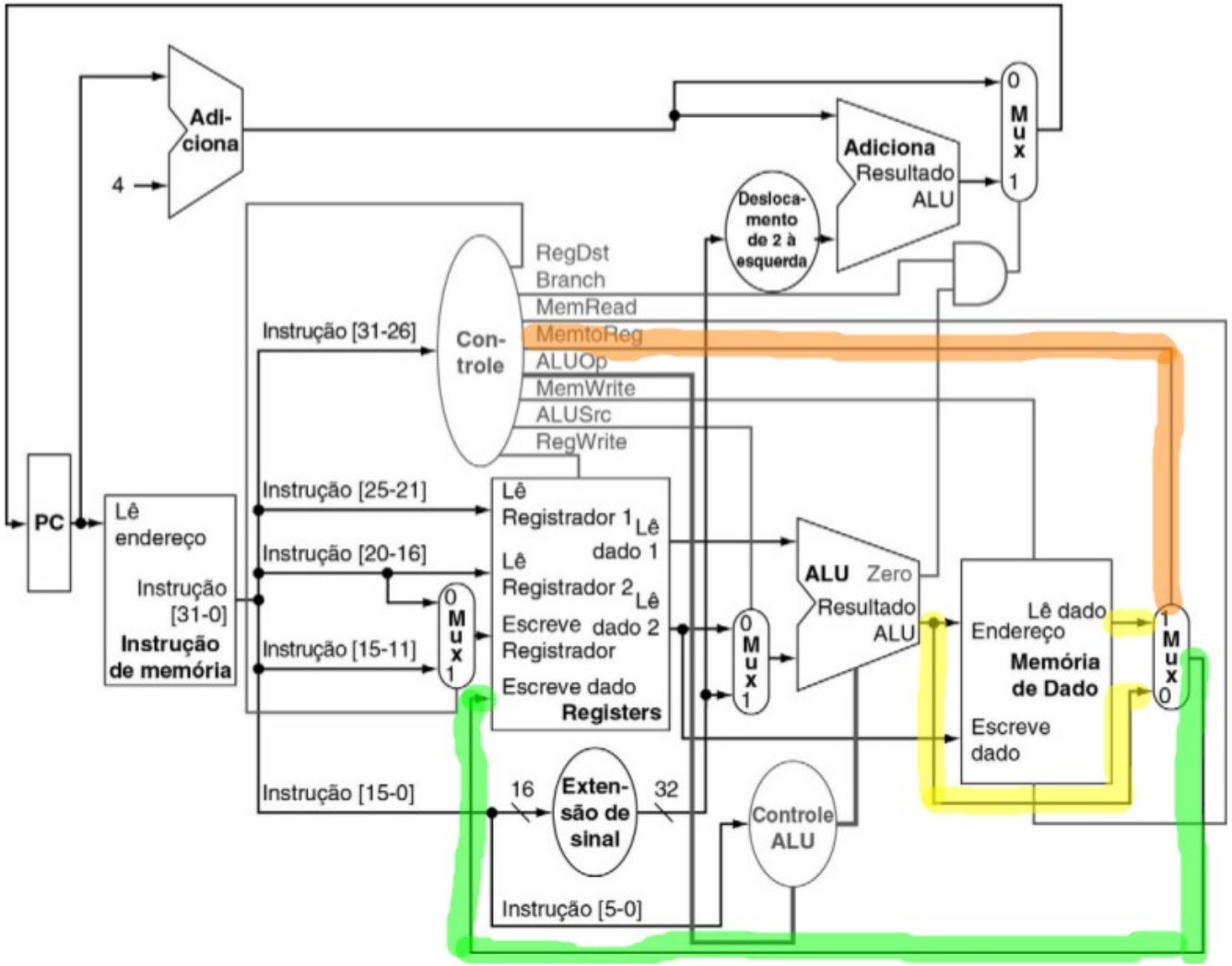
Sinais Controle – EscreveMem (MemWrite)



Sinais Controle – MemParaReg (MemToReg)

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador destino para entrada Registrador para escrita vem do campo rt (bits 20:16).	O número do registrador destino para a entrada Registrador para escrita vem do campo rd (bits 15:11).
EscreveReg	Nenhum.	O registrador na entrada Registrador para escrita é escrito com o valor da entrada Dados para escrita.
OrigALU	O segundo operando da ALU vem da segunda saída do banco de registradores (Dados da leitura 2).	O segundo operando da ALU consiste nos 16 bits mais baixos da instrução com sinal estendido.
OrigPC	O PC é substituído pela saída do somador que calcula o valor de PC + 4.	O PC é substituído pela saída do somador que calcula o destino do desvio.
LeMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é colocado na saída Dados da leitura.
EscreveMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é substituído pelo valor na entrada Dados para escrita.
MemparaReg	O valor enviado à entrada Dados para escrita do banco de registradores vem da ALU.	O valor enviado à entrada Dados para escrita do banco de registradores vem da memória de dados.

Sinais Controle – MemParaReg (MemToReg)



Unidade de Controle

- **UC da ULA:**
  - **Entradas:**
    - 6 bits do campo funct da instrução.
    - OpALU: 2 bits de controle (gerados pela UC principal).
      - » Se instrução é lw ou sw então OpALU = 00 (ALU deve realizar soma).
      - » Se instrução é beq então OpALU = 01 (ALU deve realizar subtração).
      - » Se instrução é lógica/aritmética então OpALU = 10 (ALU deve realizar operação determinada pelo campo funct).

OpALU		Campo funct						Operação
OpALU1	OpALU 2	F5	F4	F3	F2	F1	F0	
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

Unidade de Controle

- **UC da ULA:**
  - **OpALU: 2 bits de controle (gerados pela UC principal).**
    - » Se instrução é lw ou sw então OpALU = 00 (ALU deve realizar soma).
    - » Se instrução é beq então OpALU = 01 (ALU deve realizar subtração).
    - » Se instrução é lógica/aritmética então OpALU = 10 (ALU deve realizar operação determinada pelo campo funct).

Opcode da instrução	OpALU	Operação da instrução	Campo funct	Ação da ALU desejada	Entrada do controle da ALU
LW	00	load word		add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
tipo R	10	add	100000	add	0010
tipo R	10	subtract	100010	subtract	0110
tipo R	10	AND	100100	AND	0000
tipo R	10	OR	100101	OR	0001
tipo R	10	set on less than	101010	set on less than	0111

Unidade de Controle

Instrução	RegDst	OrigALU	MemparaReg	EscreveReg	LeMem	EscreveMem	Branch	ALUOp1	ALUOp0
formato R	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

- A primeira linha da tabela corresponde às instruções formato R (add, sub, AND, OR e slt).
- 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.
- Uma instrução tipo R escreve em um registrador (EscreveReg = 1), mas não escreve ou lê a memória de dados.

op(31:26)								
28–26 31–29	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	formato R	Bltz/gez	jump	jump & link	branch eq	branch ne	blez	bgtz
1(001)	add immediate	addiu	set less than imm.	set less than imm. unsigned	andi	ori	xori	load upper immediate
2(010)	TLB	FlPt						
3(011)								
4(100)	load byte	load half	lwl	load word	load byte unsigned	load half unsigned	lwr	
5(101)	store byte	store half	swl	store word			swr	
6(110)	load linked word	lwc1						
7(111)	store cond. word	swc1						

op(31:26)=010000 (TLB), rs(25:21)								
23–21 25–24	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(00)	mfc0		cfc0		mtc0		ctc0	
1(01)								
2(10)								
3(11)								

op(31:26)=000000 (format R), funct(5:0)								
2–0 5–3	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	shift left logical		shift right logical	sra	sllv		srlv	srav
1(001)	jump register	jalr			syscall	break		
2(010)	mfhi	mthi	mflo	mtlo				
3(011)	mult	multu	div	divu				
4(100)	add	addu	subtract	subu	and	or	xor	not or (nor)
5(101)			set l.t.	set l.t. unsigned				
6(110)								
7(111)								

FIGURA 2.19 Codificação de instruções do MIPS. Essa notação indica o valor de um campo por linha e por coluna. Por exemplo, a parte superior da figura mostra `load word` na linha número 4 (100_{bin} para os bits 31-29 da instrução) e a coluna número 3 (011_{bin} para os bits 28-26 da instrução), de modo que o valor correspondente do campo op (bits 31-26) é 100011_{bin}. `Formato R` na linha 0 e coluna 0 (op = 000000_{bin}) é definido na parte inferior da figura. Logo, `subtract` na linha 4 e coluna 2 da seção inferior significa que o campo funct (bits 5-0) da instrução é 100010_{bin} e o campo op (bits 31-26) é 000000_{bin}. O valor de ponto flutuante na linha 2, coluna 1, é definido na Figura 3.18, no Capítulo 3. `Bltz/gez` é o opcode para quatro instruções encontradas no Apêndice B: `bltz`, `bgez`, `bltzal` e `bgezal`. Este capítulo descreve as instruções indicadas com nome completo usando cor, enquanto o Capítulo 3 descreve as instruções indicadas em mnemônicos usando cor. O Apêndice B abrange todas as instruções.

Unidade de Controle

Instrução	RegDst	OrigALU	MemparaReg	EscreveReg	LeMem	EscreveMem	Branch	ALUOp1	ALUOp0
formato R	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

- A primeira linha da tabela corresponde às instruções formato R (add, sub, AND, OR e slt).
- 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.
- Uma instrução tipo R escreve em um registrador (EscreveReg = 1), mas não escreve ou lê a memória de dados.

Unidade de Controle

Instrução	RegDst	OrigALU	MemparaReg	EscreveReg	LeMem	EscreveMem	Branch	ALUOp1	ALUOp0
formato R	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

- Quando o sinal de controle Branch é 0, o PC é incondicionalmente substituído por PC + 4; caso contrário, o PC é substituído pelo destino do desvio se a saída Zero da ALU também está ativa.
- O campo OpALU para as instruções tipo R é definido como 10 a fim de indicar que o controle da ALU deve ser gerado do campo funct

Unidade de Controle

Instrução	RegDst	OrigALU	MemparaReg	EscreveReg	LeMem	EscreveMem	Branch	ALUOp1	ALUOp0
formato R	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

- A segunda e a terceira linhas dessa tabela fornecem as definições dos sinais de controle para lw e sw.
- Esses campos OrigALU e OpALU são ativados para realizar o cálculo do endereço. LeMem e EscreveMem são ativados para realizar o acesso à memória.
- Finalmente, RegDst e EscreveReg são ativados para que um load faça o resultado ser armazenado no registrador rt.

Unidade de Controle

Instrução	RegDst	OrigALU	MemparaReg	EscreveReg	LeMem	EscreveMem	Branch	ALUOp1	ALUOp0
formato R	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

- A instrução branch é semelhante à operação no formato R, já que ela envia os registradores rs e rt para a ALU.
- O campo OpALU para um desvio é definido como uma subtração (controle da ALU = 01), usada para testar a igualdade.
- Repare que o campo MemparaReg é irrelevante quando o sinal EscreveReg é 0: como o registrador não está sendo escrito, o valor dos dados na entrada Dados para escrita do banco de registradores não é usado.

Unidade de Controle

Instrução	RegDst	OrigALU	MemparaReg	EscreveReg	LeMem	EscreveMem	Branch	ALUOp1	ALUOp0
formato R	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

- Portanto, a entrada MemparaReg nas duas últimas linhas da tabela é substituída por X (don't care).
- Os don't care também podem ser adicionados a RegDst quando EscreveReg é 0.
- Esse tipo de don't care precisa ser acrescentado pelo projetista, uma vez que ele depende do conhecimento de como o caminho de dados funciona.

Unidade de Controle

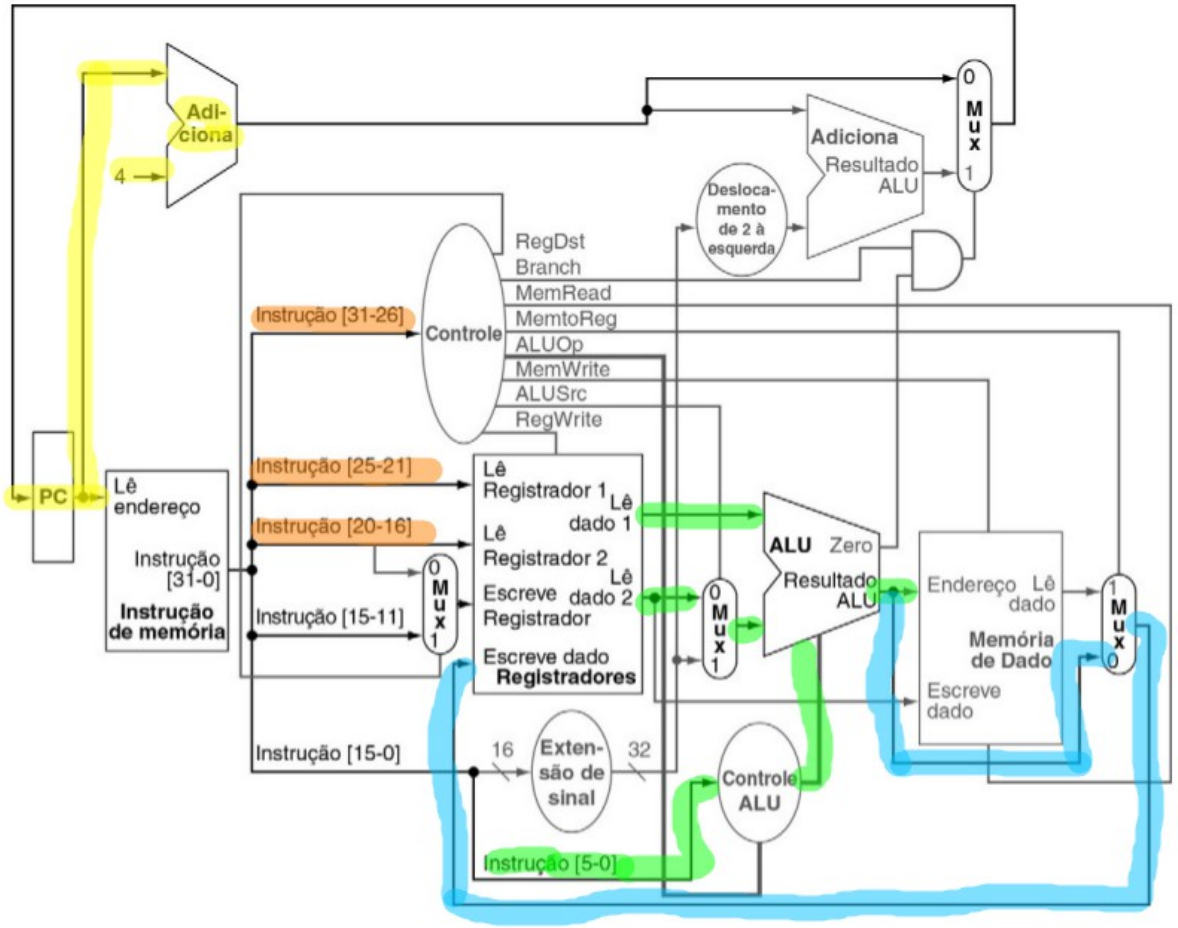


FIGURA 4.19 O caminho de dados em operação para uma instrução tipo R como `add $t1,$t2,$t3`**.** As linhas de controle, as unidades do caminho de dados e as conexões que estão ativas aparecem destacadas.

Unidade de Controle

1. A instrução é buscada e o PC é incrementado.
2. 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.
3. A ALU opera nos dados lidos do banco de registradores, usando o código de função (bits 5:0, que é o campo funct, da instrução) para gerar a função da ALU.
4. O resultado da ALU é escrito no banco de registradores usando os bits 15:11 da instrução para selecionar o registrador de destino (`$t1`).

Unidade de Controle

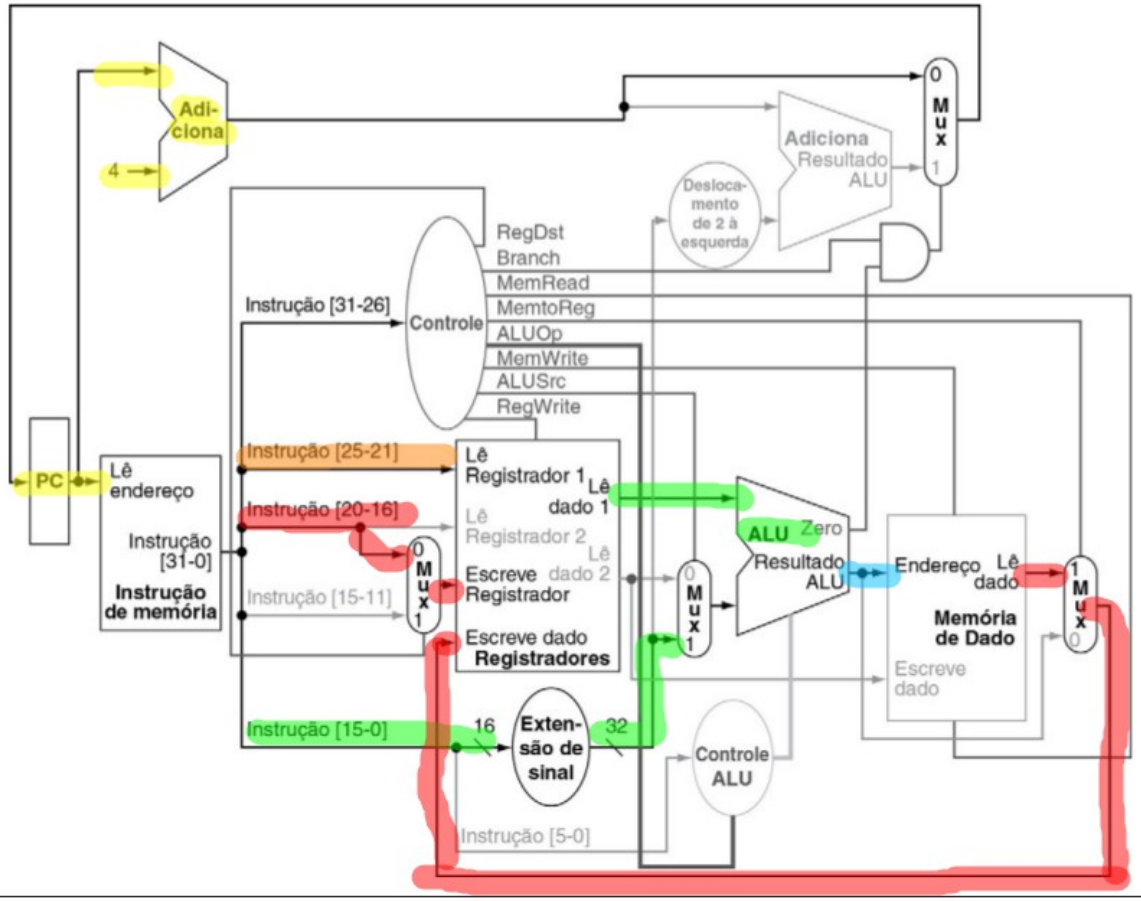


FIGURA 4.20 O caminho de dados em operação para uma instrução load. As linhas de controle, as unidades do caminho de dados e as conexões ativas aparecem destacadas. Uma instrução store operaria de maneira muito semelhante. A principal diferença seria que o controle da memória indicaria uma escrita em vez de uma leitura, a segunda leitura do valor de um registrador seria usada para os dados a serem armazenados e a operação de escrita do valor da memória de dados no banco de registradores não ocorreria.

Unidade de Controle

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

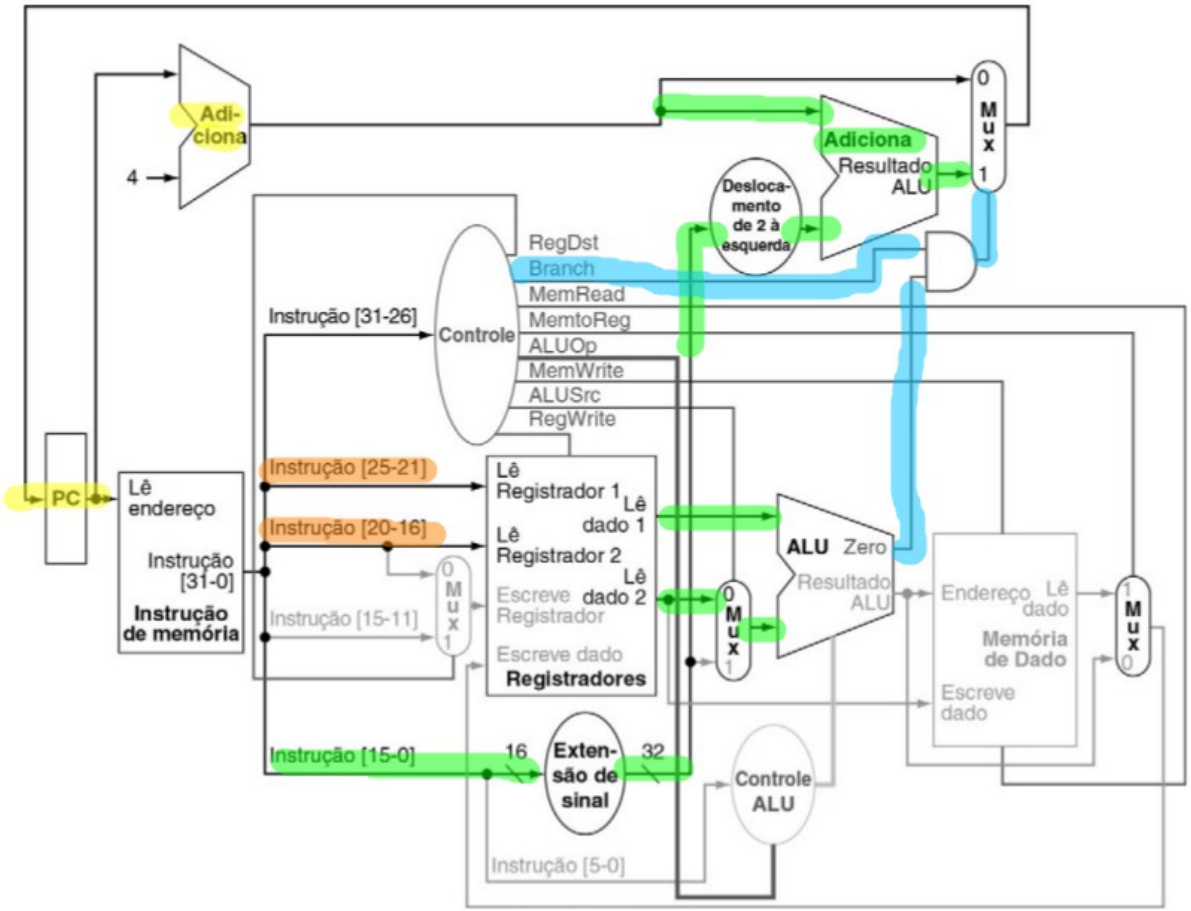


FIGURA 4.21 O caminho de dados em operação para uma instrução branch equal. As linhas de controle, as unidades do caminho de dados e as conexões que estão ativas aparecem destacadas. Após usar o banco de registradores e a ALU para realizar a comparação, a saída Zero é usada na seleção do próximo contador de programa dentre os dois candidatos.

Unidade de Controle

1. Uma instrução é buscada da memória de instruções e o PC é incrementado.
2. Dois registradores, `$t1` e `$t2`, são lidos do banco de registradores.
3. A ALU realiza uma subtração dos valores de dados lidos do banco de registradores. O valor de PC + 4 é somado aos 16 bits menos significativos com sinal estendido (`offset`) deslocados de dois para a esquerda; o resultado é o endereço de destino do desvio.
4. O resultado Zero da ALU é usado para decidir qual resultado do somador deve ser armazenado no PC.

Unidade de Controle

Entrada ou saída	Nome do sinal	formato R	lw	sw	beq
Entradas	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Saídas	RegDst	1	0	X	X
	OrigALU	0	1	1	0
	MemparaReg	0	1	X	X
	EscreveReg	1	1	0	0
	LeMem	0	1	0	0
	EscreveMem	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

- A Tabela especifica completamente a função de controle, e podemos implementá-la diretamente em portas lógicas de uma maneira automatizada.
- Seção C.2 no Apêndice C.

Referências

Notas de Aulas do Prof. João Angelo Martini, DIN-UEM.

ARAUJO, M. R. D. ; PÁDUA, F. L. C. ; ANDRADE, F. V. ; CORREA JUNIOR, F. L. . MIPS X-Ray: A MARS Simulator Plug-in for Teaching Computer Architecture. International Journal of Recent Contributions from Engineering, Science & IT (iJES), v. 2, p. 36, 2014.