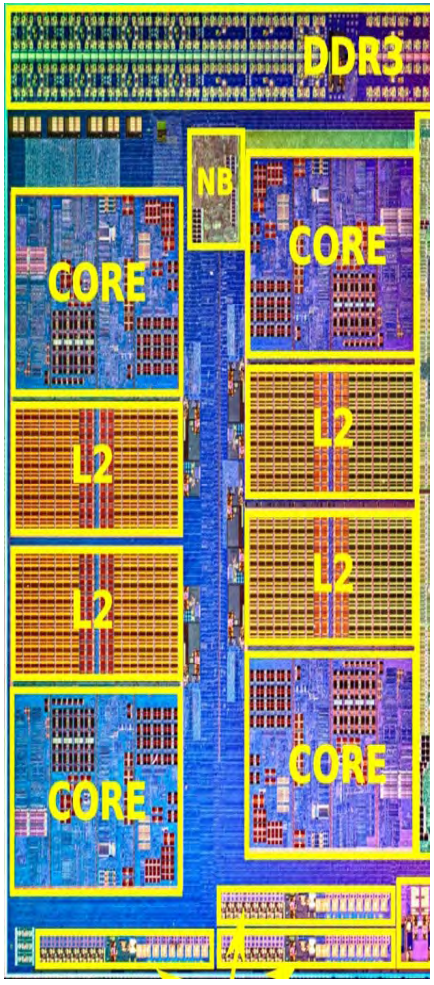
0273359 – Arquitetura e Organização de Computadores 1



Fonte: http://www.techspot.com/article/904-history-of-the-personal-computer-part-5/

Construindo um Datapath e Controle para MIPS

Luciano de Oliveira Neris
luciano@dc.ufscar.br

Adaptado de slides do prof. Marcio Merino Fernandes

Departamento de Computação
Universidade Federal de São Carlos

2

Implementando um Processador

Implementando um Processador

- Vamos ver como um processador é implementado
  - Iniciamos c/ um processador muito simples, e adicionamos recursos extras passo-a-passo

- ISA adotada: subconjunto do MIPS:
  - Instruções de acesso à memória: lw and sw
  - Instruções da ALU: add, sub, or, slt
  - Instruções p/ fluxo de controle: beq and j

Implementando um Processador

- Subconjunto MIPS adotado:

lw: load word
sw: store word

add: adição
sub: subtração
or: ou bit-a-bit
slt: set if less than

beq: branch if equal
jump: desvio incondicional

Funcionamento do Processador

- Três Passos:

  —(FETCH / Busca): Acessar o endereço de memória correspondente para carregar a próxima instrução (32 bits)

  —(DECODE / Decodificação): Ler 0, 1 ou 2 registradores de acordo com os campos correspondentes na instrução

  —(EXECUTE / Execução): Executar a instrução

Funcionamento do Processador

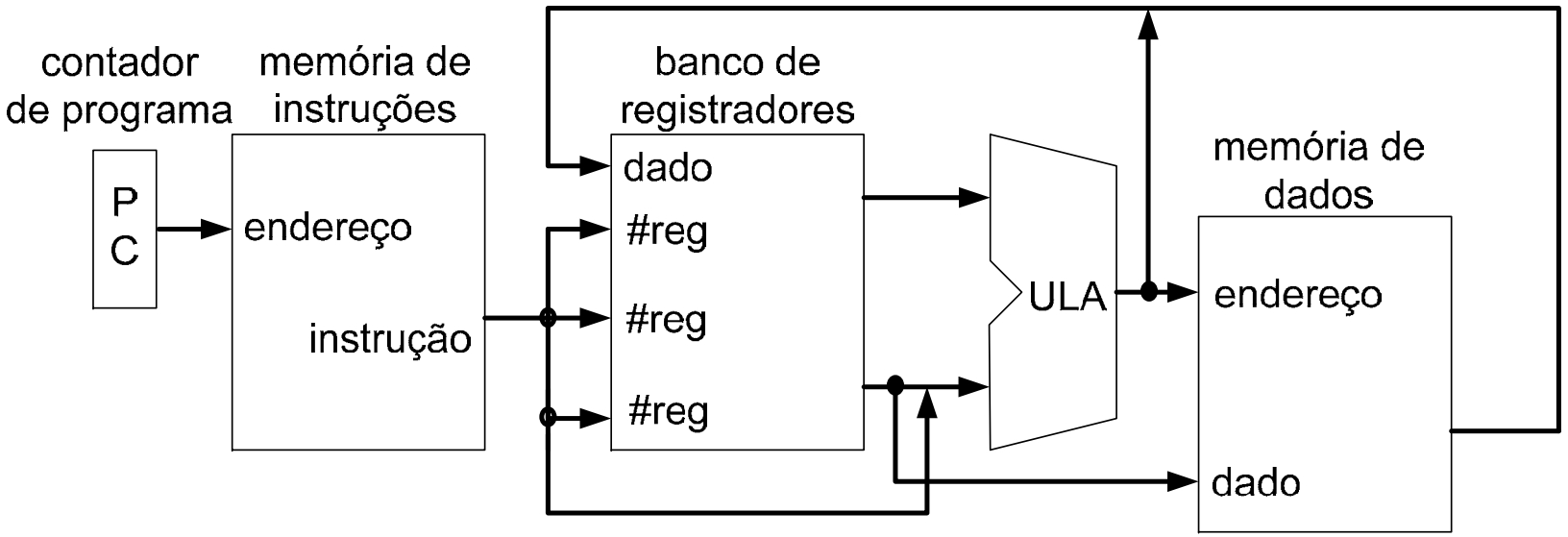
- Executar a instrução
  - Felizmente, existem algumas similaridades na execução de diferentes instruções: a maioria delas utiliza a ULA p/ calcular um valor numérico, ou um endereço de memória.

- Também existem diferenças:
  - Apenas duas delas acessam a memória
  - Store não escreve em nenhum registrador
  - Apenas branch e jump modificam o valor do PC

Processador Simplificado

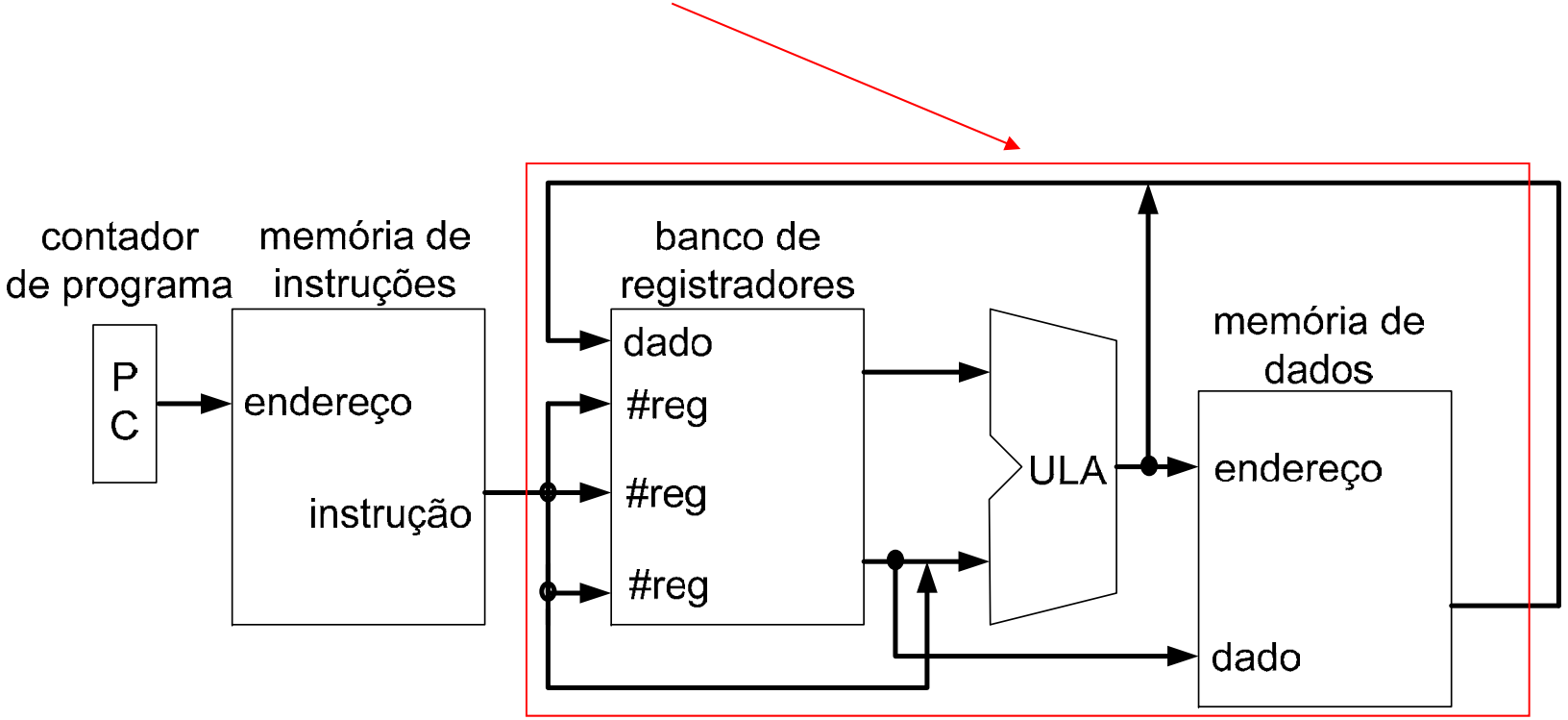
- Para implementar os 3 passos anteriores, nosso processador terá 5 componentes:
  - —Registrador PC (Program Counter) fornece o endereço da instrução
  - —Memória* onde as *instruções* são lidas (o campo de opcode é usado para executar a operação correspondente)
  - —Memória* onde os *dados* são lidos ou gravados
  - —ULA
  - —Conjunto de registradores
- **Obs: Computadores c/ Memórias separadas p/ instruções e dados são conhecidos como "Arquitetura Harvard"*

Processador Simplificado

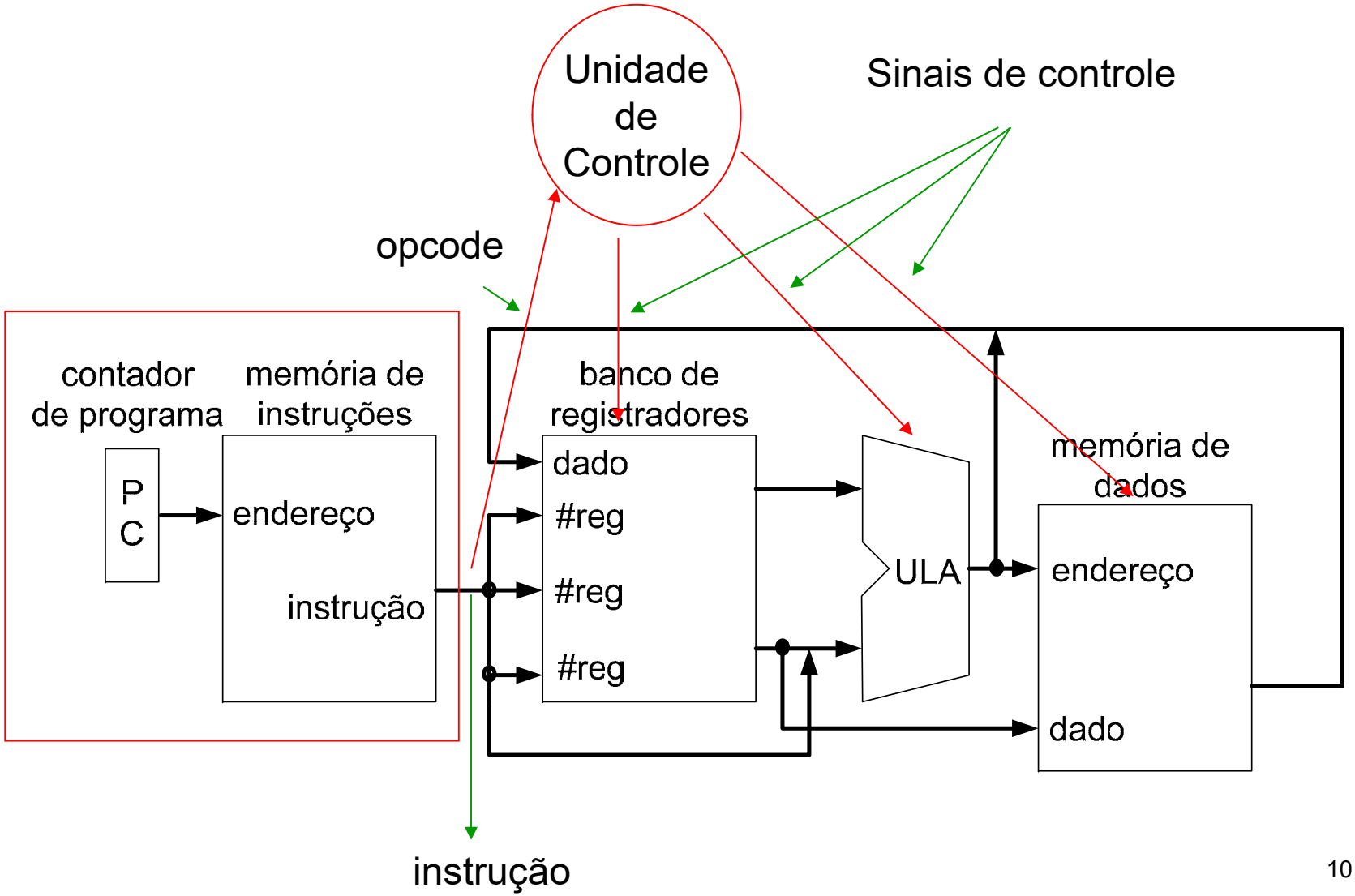


Processador Simplificado

À direita da figura vemos os elementos principais do fluxo de dados do MIPS



Processador Simplificado



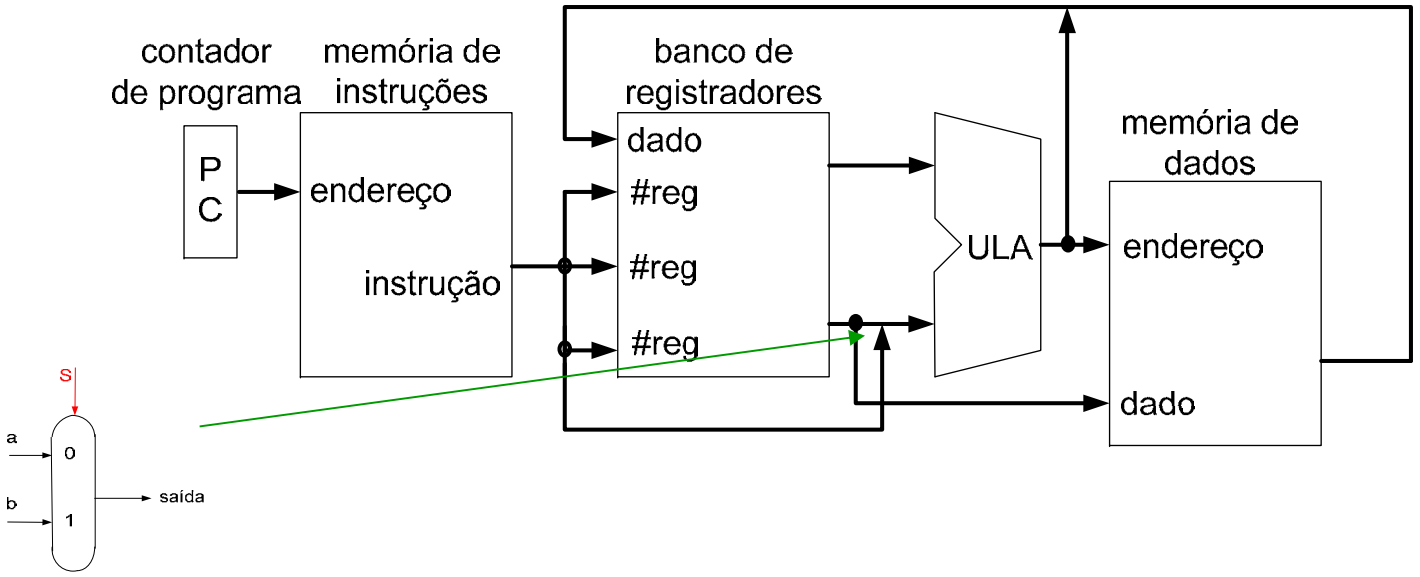
Unidade de Controle

- Envia os bits de controle apropriados p/ todos os multiplexadores e componentes da CPU, de acordo c/ a instrução sendo executada.

- Isso é feito baseado nos bits do opcode da instrução

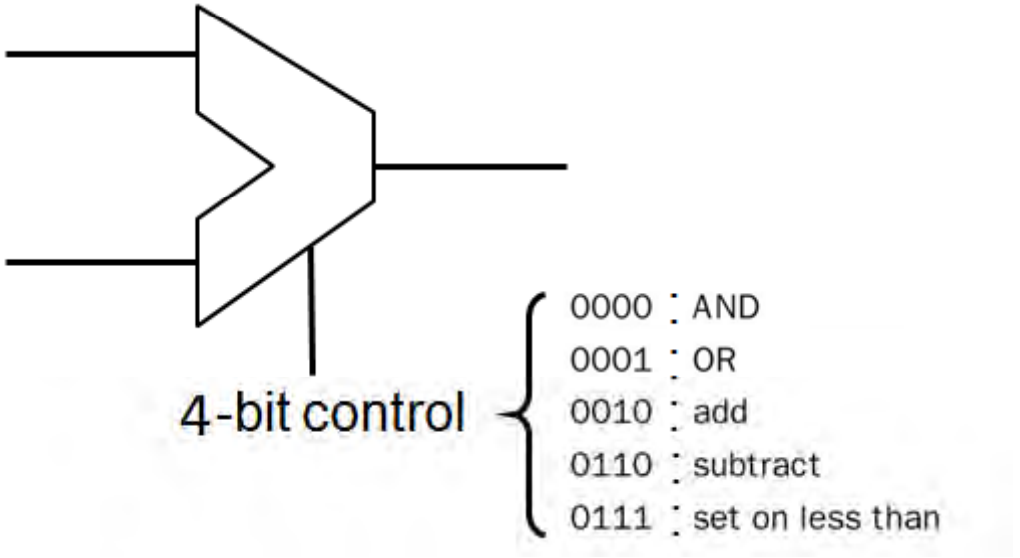
Unidade de Controle

- Dependendo da instrução sendo executada, diferentes sinais alimentam o componente.
- Assim, dependendo da instrução, devemos decidir qual entrada será selecionada
- Isto é feito pelo Multiplexador (MUX)



Unidade de Controle

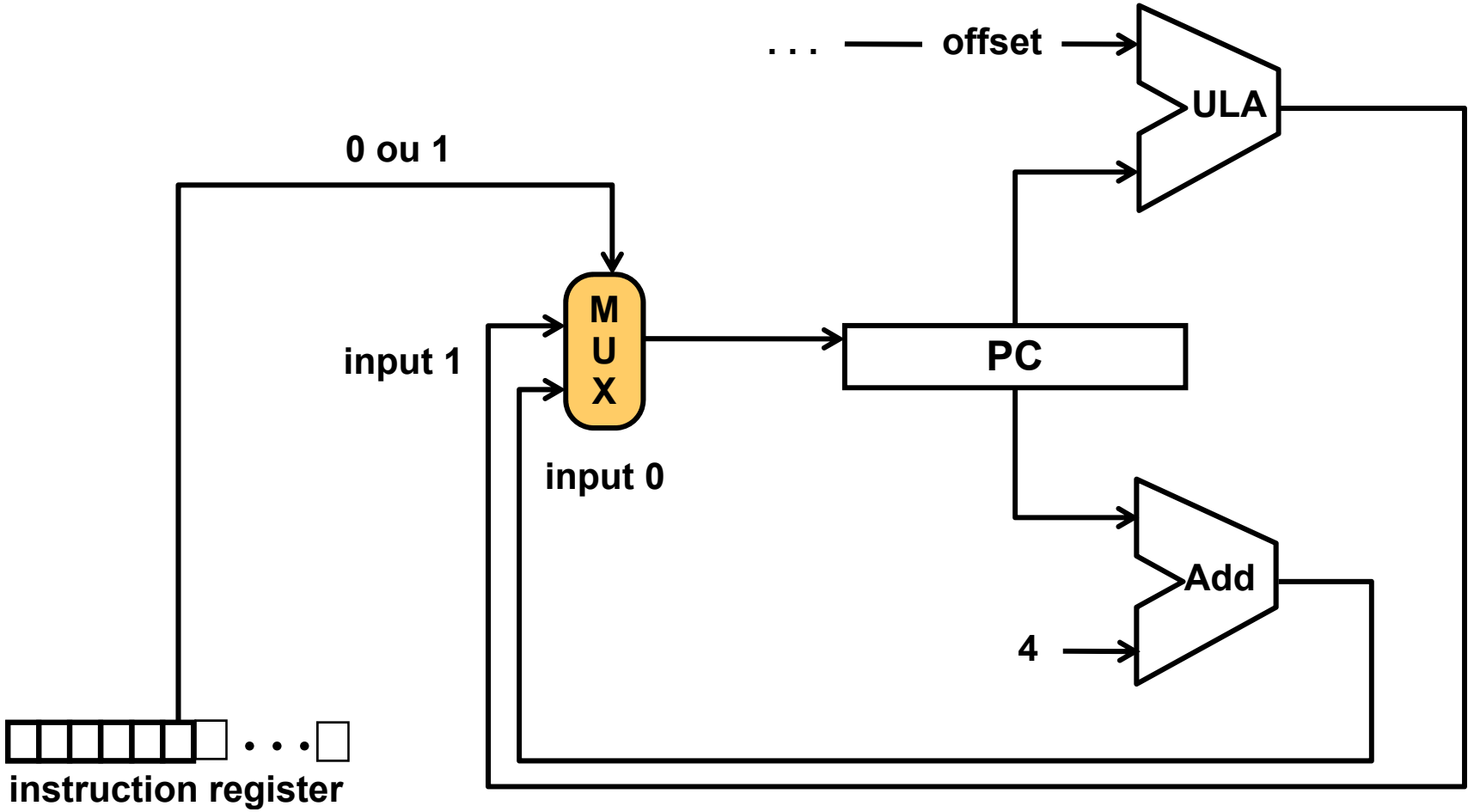
- Controladores
  - A ULA possui uma série de bits de controle p/ determinar o que ela deve fazer



Unidade de Controle

- Exemplo
  - Imagine que a instrução branch tem um opcode do tipo XXXXX1, e todos os demais opcodes são do tipo XXXXX0
  - Assim, a unidade de controle pode decidir se o valor do PC será atualizado c/ o valor "default" = (PC+4), ou da saída da ULA

Unidade de Controle

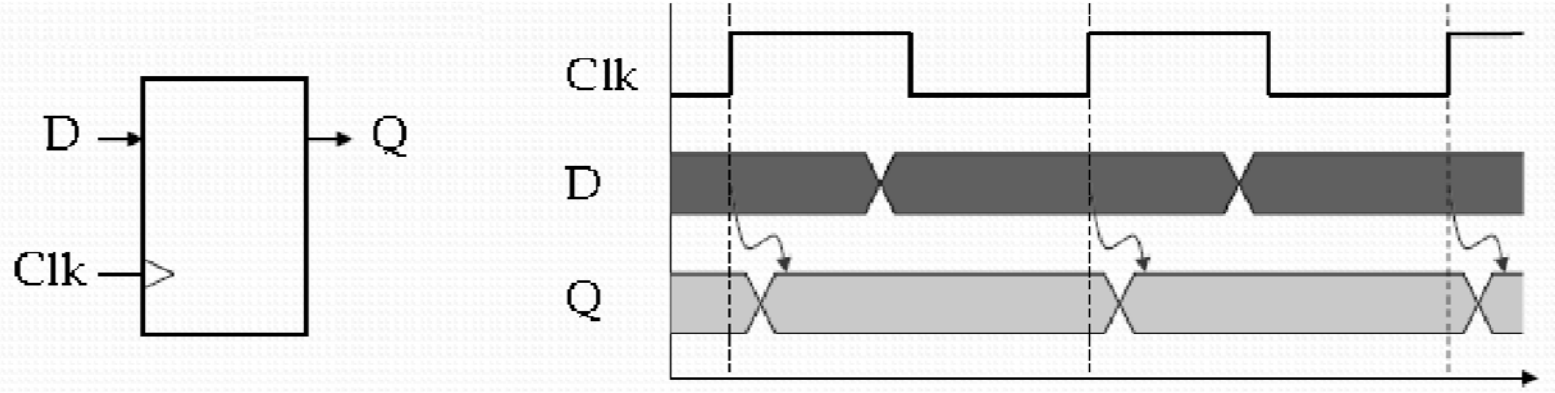


Circuitos Lógicos Digitais

- Circuito combinacional: as saídas dependem apenas dos níveis lógicos atuais das entradas.
  - Dada a mesma entrada, sempre produzem a mesma saída
  - Não possuem armazenamento interno, estado (memória)
- Circuito sequencial: as saídas dependem dos níveis lógicos das entradas e das informações armazenadas na memória interna (estado).

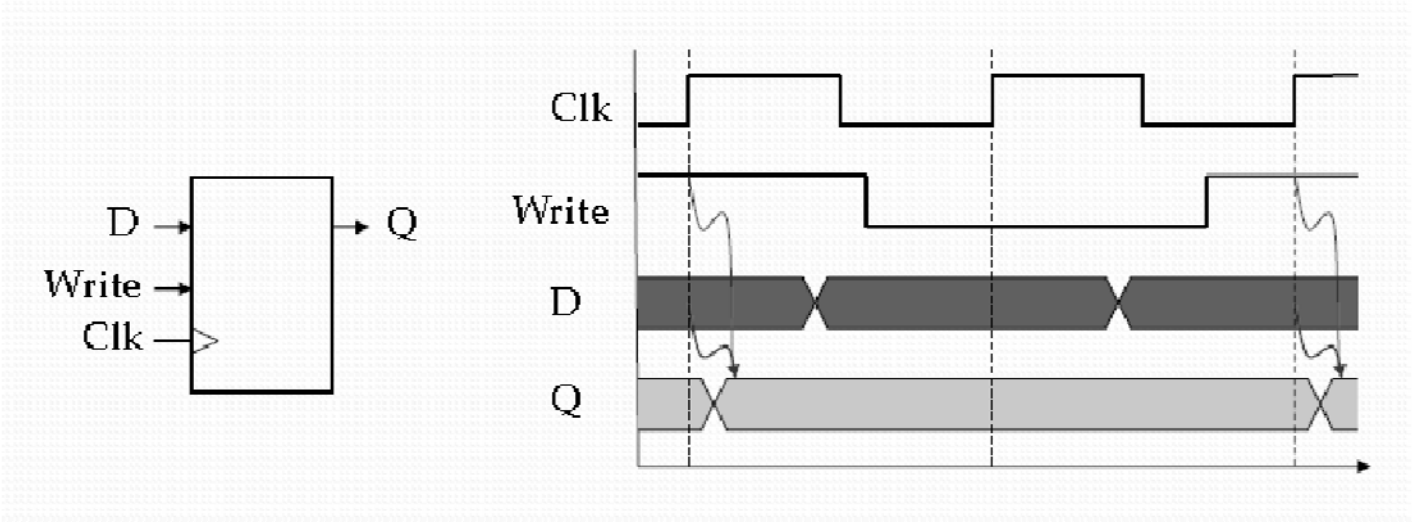
Circuitos Lógicos Digitais

- Unidade de armazenamento:
  - Sensível à borda: as saídas são atualizadas quando o clock muda de 0 para 1.

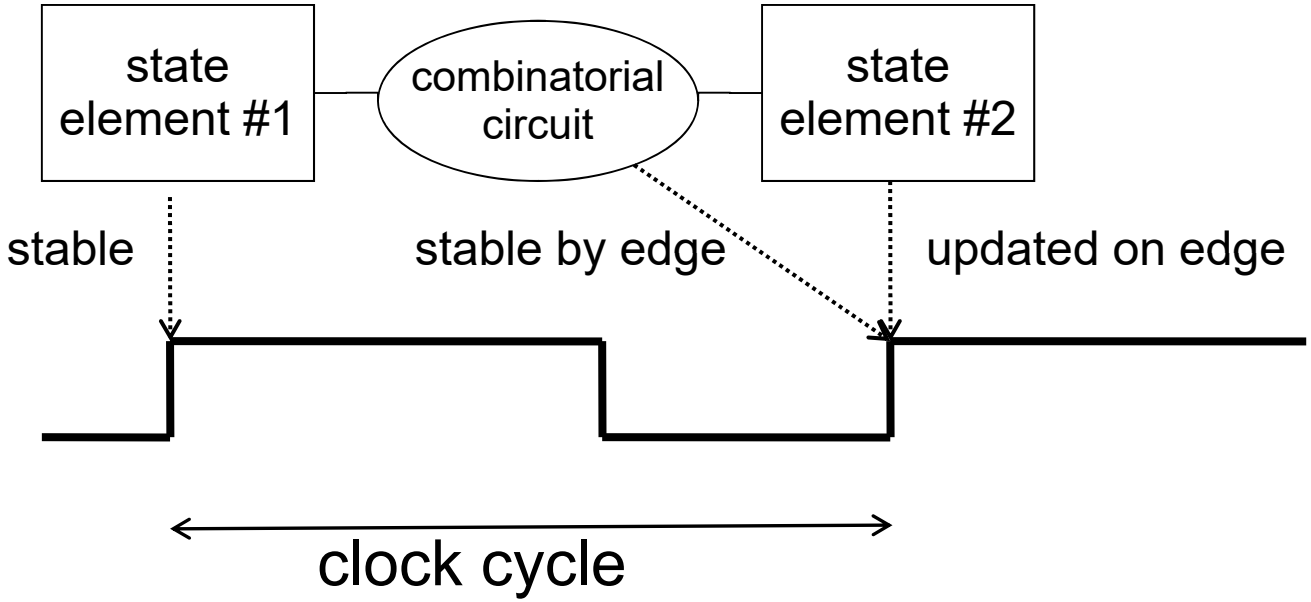


Circuitos Lógicos Digitais

- Registrador com entrada de controle:
  - Atualiza o valor armazenado apenas na subida de relógio quando a entrada de controle (Write) está ativa (nesse caso, no nível ALTO).



Clock



- Podemos usar o clock para controlar a leitura de um dado, seu uso em um circuito combinacional, e gravar a saída desse circuito
- O maior atraso determina o período do relógio.

Clock

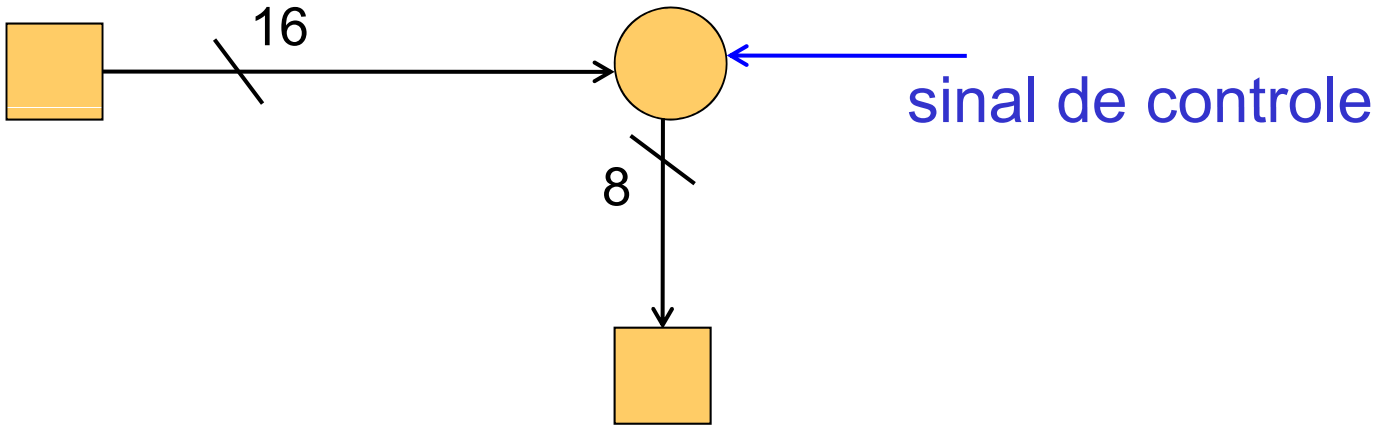
- Leitura e Escrita no mesmo Ciclo
  - Esse esquema permite que um mesmo elemento de estado possa ser lido e escrito no mesmo ciclo, sem que ocorram "condições de corrida".
    - Ex: leituras ocorrem na subida de borda, e escritas na descida.

Clock

- Opção: implementação monociclo (uniciclo)
  - Toda instrução inicia sua execução em uma borda ativa do sinal de relógio e termina na próxima borda.
  - Vantagem: simples de entender.
  - Desvantagem: o ciclo de relógio precisa ser alongado o suficiente para acomodar a instrução mais longa.

Barramentos

- A maioria dos elementos combinacionais e sequenciais possuem vários bits de entrada (ex: 32-bit inputs)
- O termo "barramento" (bus) refere-se a um fio composto por vários bits em paralelo.
- Indicamos a largura do barramento como se segue:

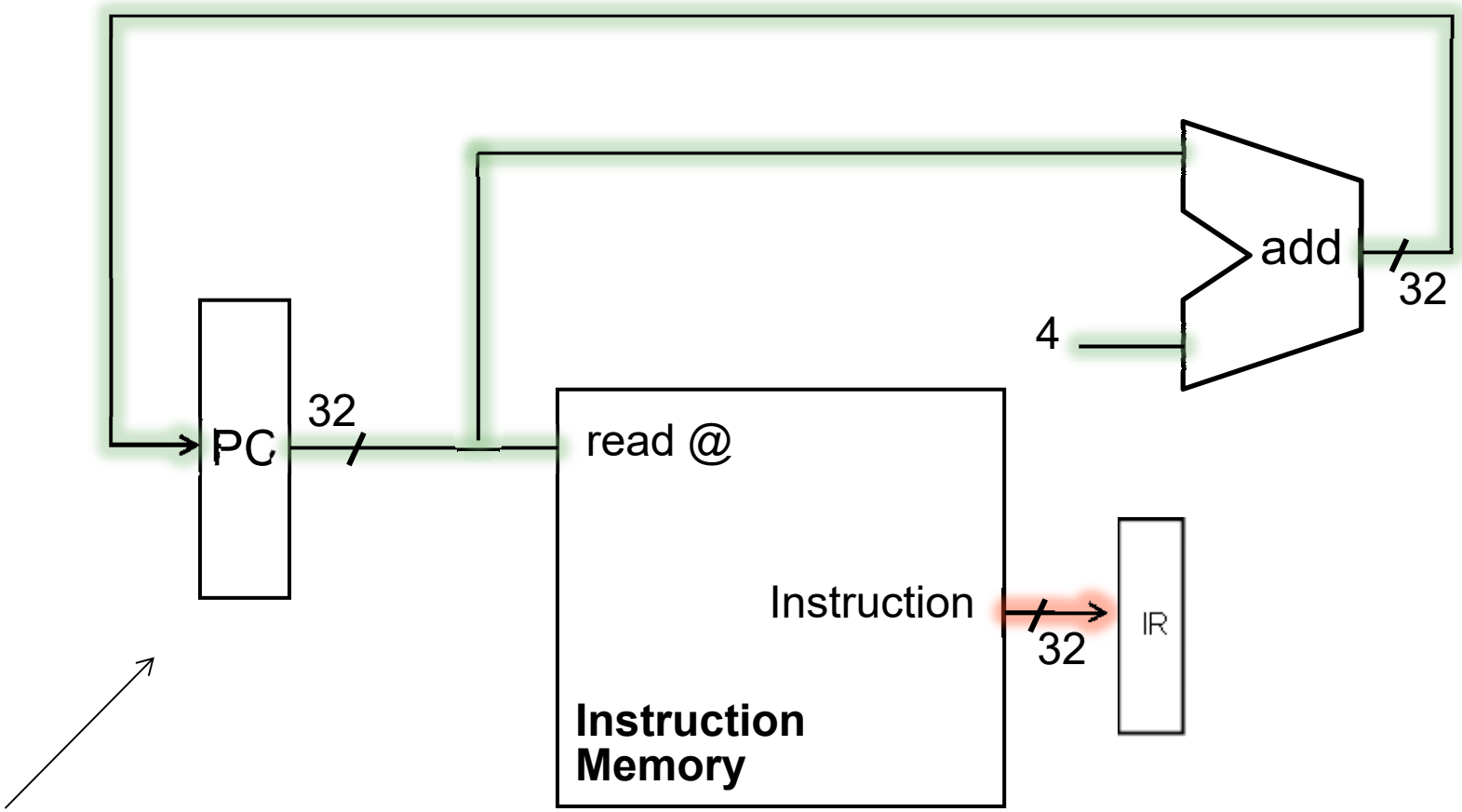


Construindo um Processador Simplificado

- Para executar qualquer instrução, precisamos buscá-la na memória
- O registrador Program Counter (PC) é utilizado para ler a instrução da memória e armazená-la no Registrador de Instrução (IR)
- Um somador incrementa PC em 4 (uma word) e coloca o resultado de volta em PC

Construindo um Processador Simplificado

- Busca de Instrução (Fetch)



O PC é atualizado em 1 ciclo de clock

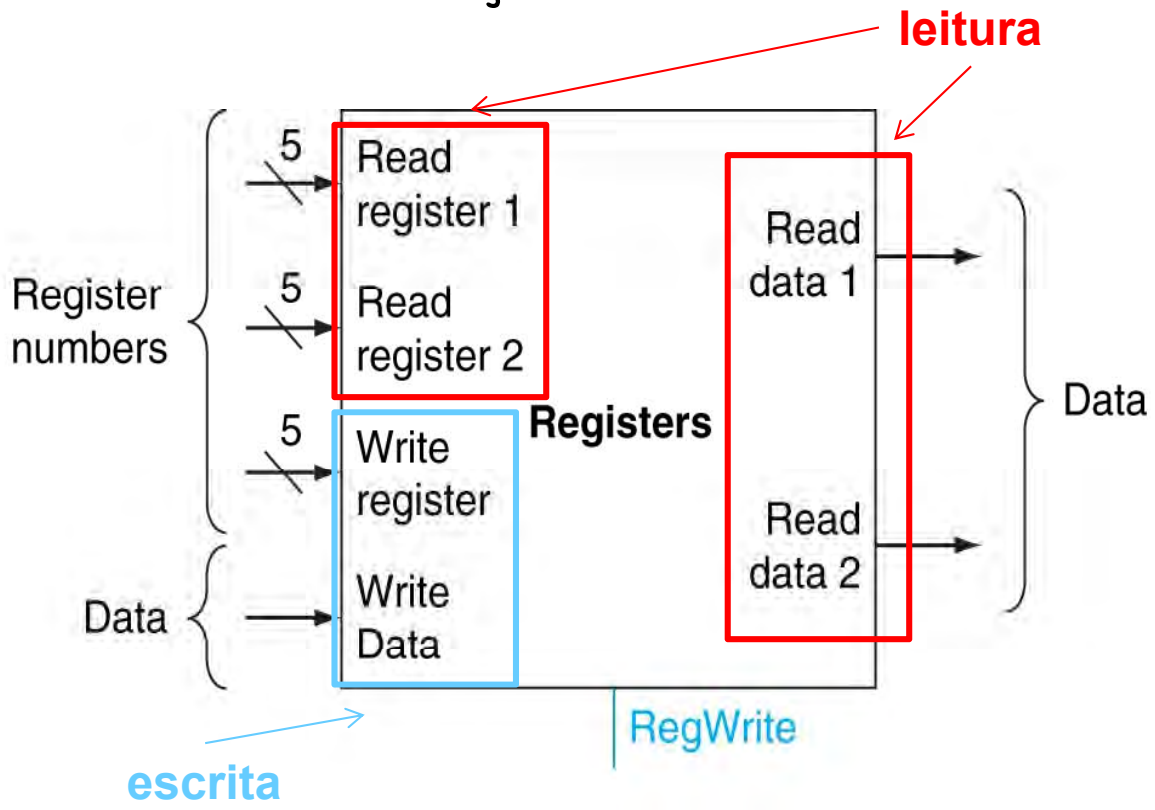
Construindo um Processador Simplificado

- Instruções do Tipo-R
  - Possuem 3 registradores como operandos:
    - 2 registradores para leitura (Rs, Rt)
    - 1 registrador para escrita (Rd)
  - Ex: add t1, t1, t2

- Banco de Registradores
  - Leitura:
    - Endereços dos registradores a serem lidos
    - Saída de dados para os conteúdos lidos
  - Escrita
    - Endereço do registrador a ser escrito
    - Entrada de dados a serem escritos

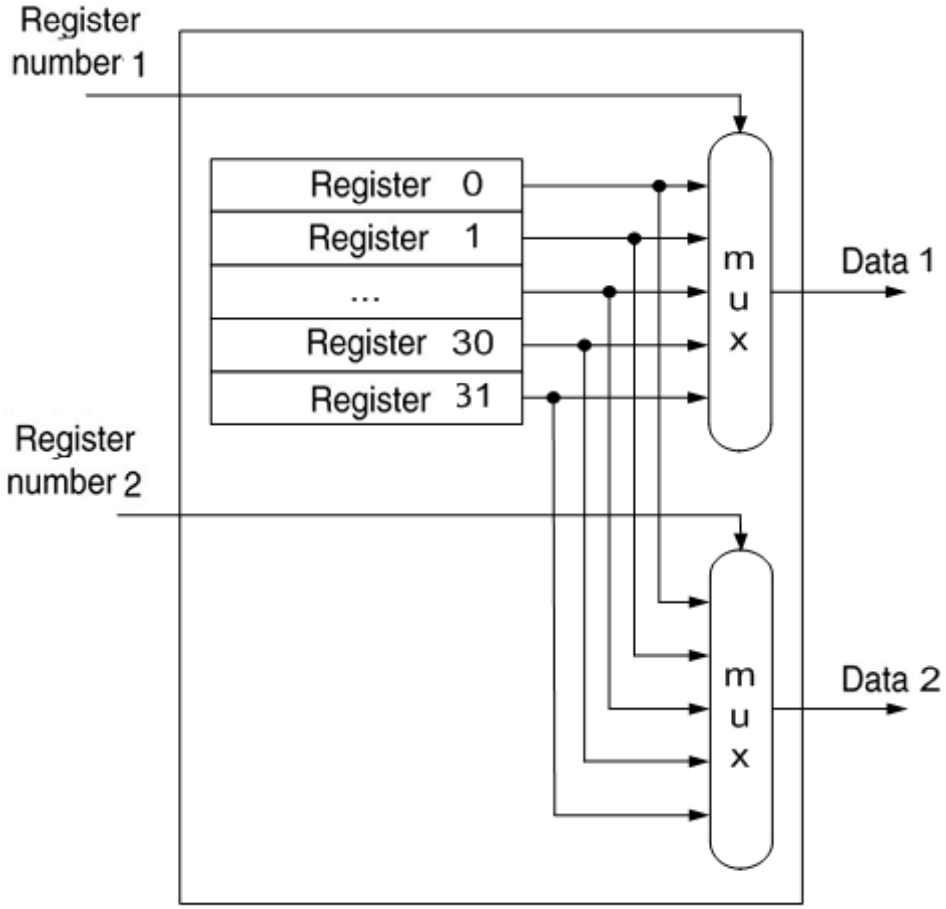
Construindo um Processador Simplificado

- Banco de Registradores
  - Cada registrador é representado por um código de 5-bits, que é extraído da instrução de 32 bits.



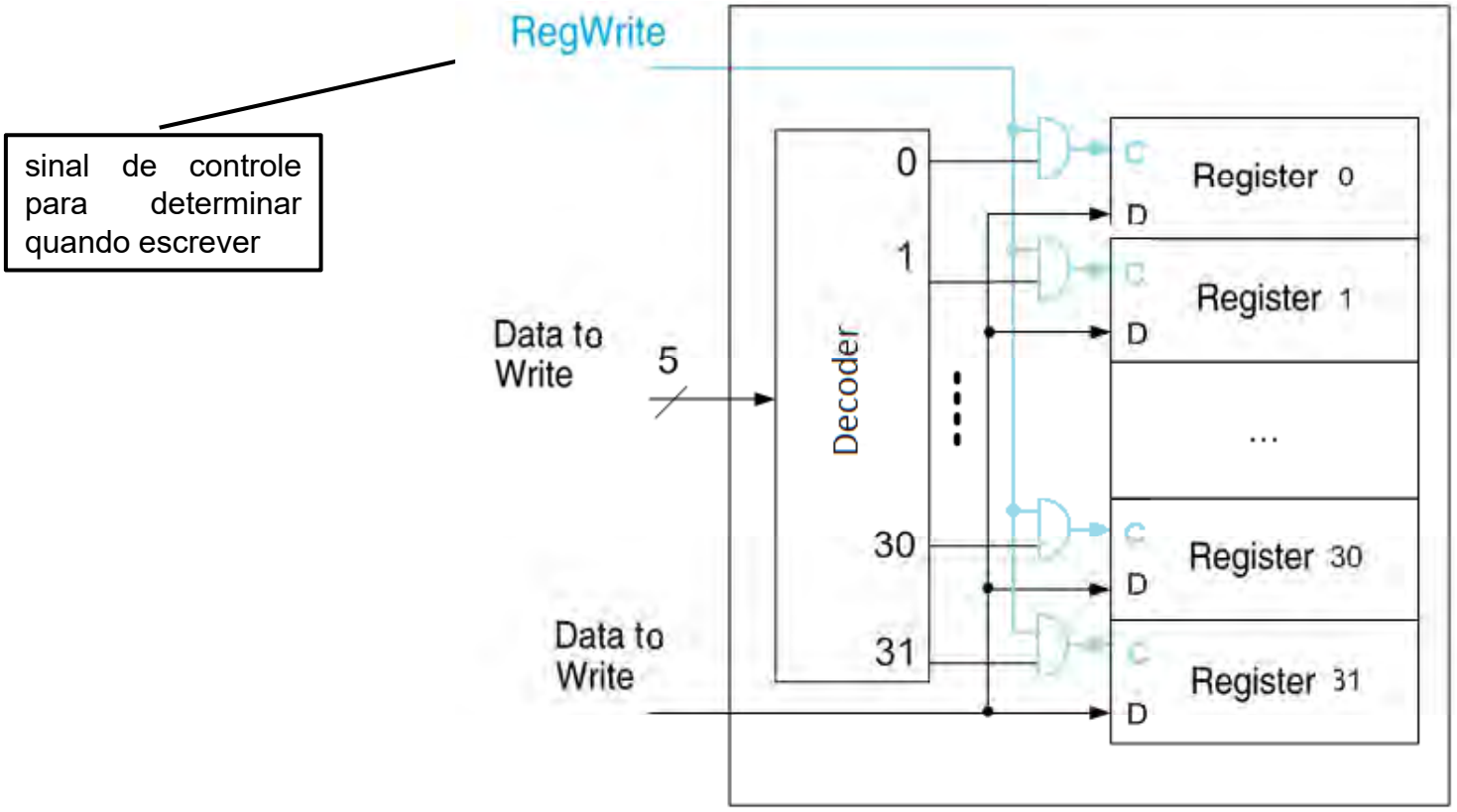
Construindo um Processador Simplificado

- Banco de Registradores
  - Diagrama do circuito de leitura do banco de registradores



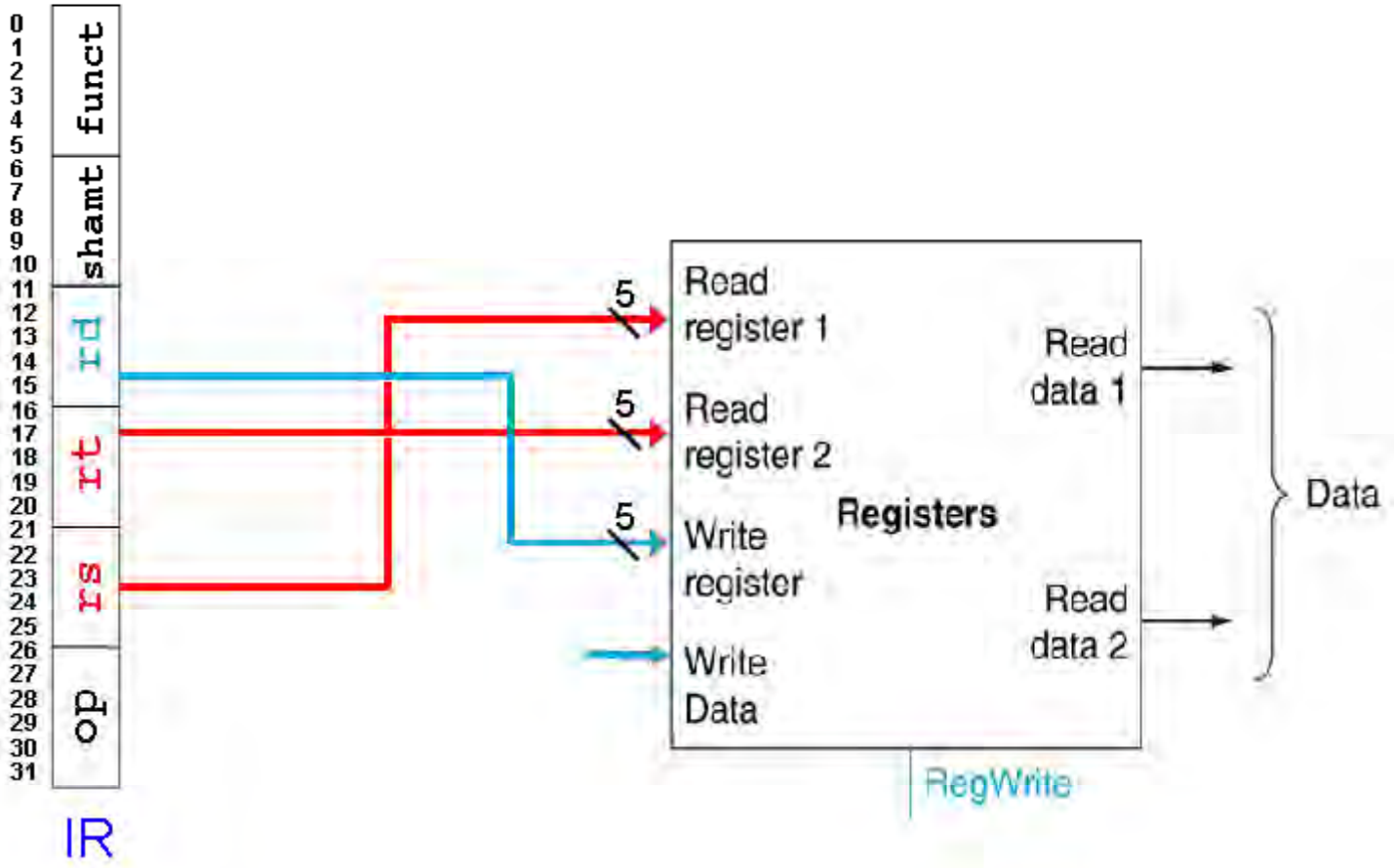
Construindo um Processador Simplificado

- Banco de Registradores
  - Diagrama do circuito de escrita no banco de registradores



Construindo um Processador Simplificado

- Instruções do Tipo-R

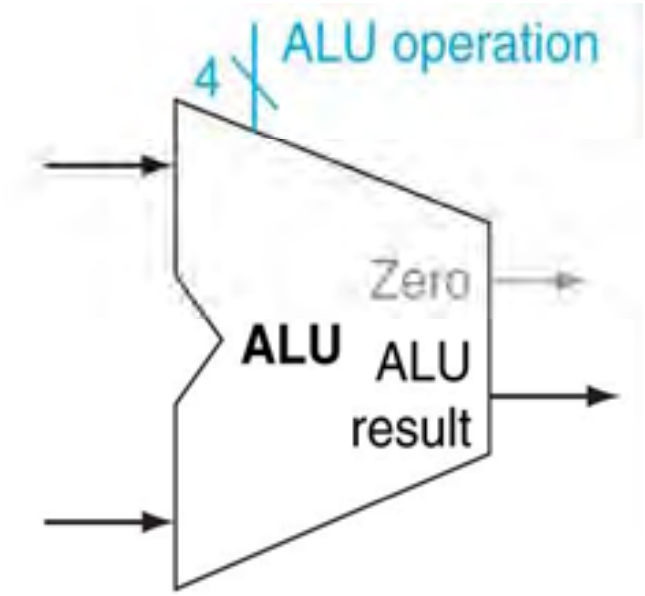


Construindo um Processador Simplificado

- Instruções do Tipo-R
  - Precisamos de um componente que forneça os dados a serem gravados no registrador indicado para a escrita
  - Esse componente frequentemente é a ULA
  - Precisamos de um sinal de escrita para disparar a gravação do registrador de saída na próxima mudança de borda do ciclo de clock

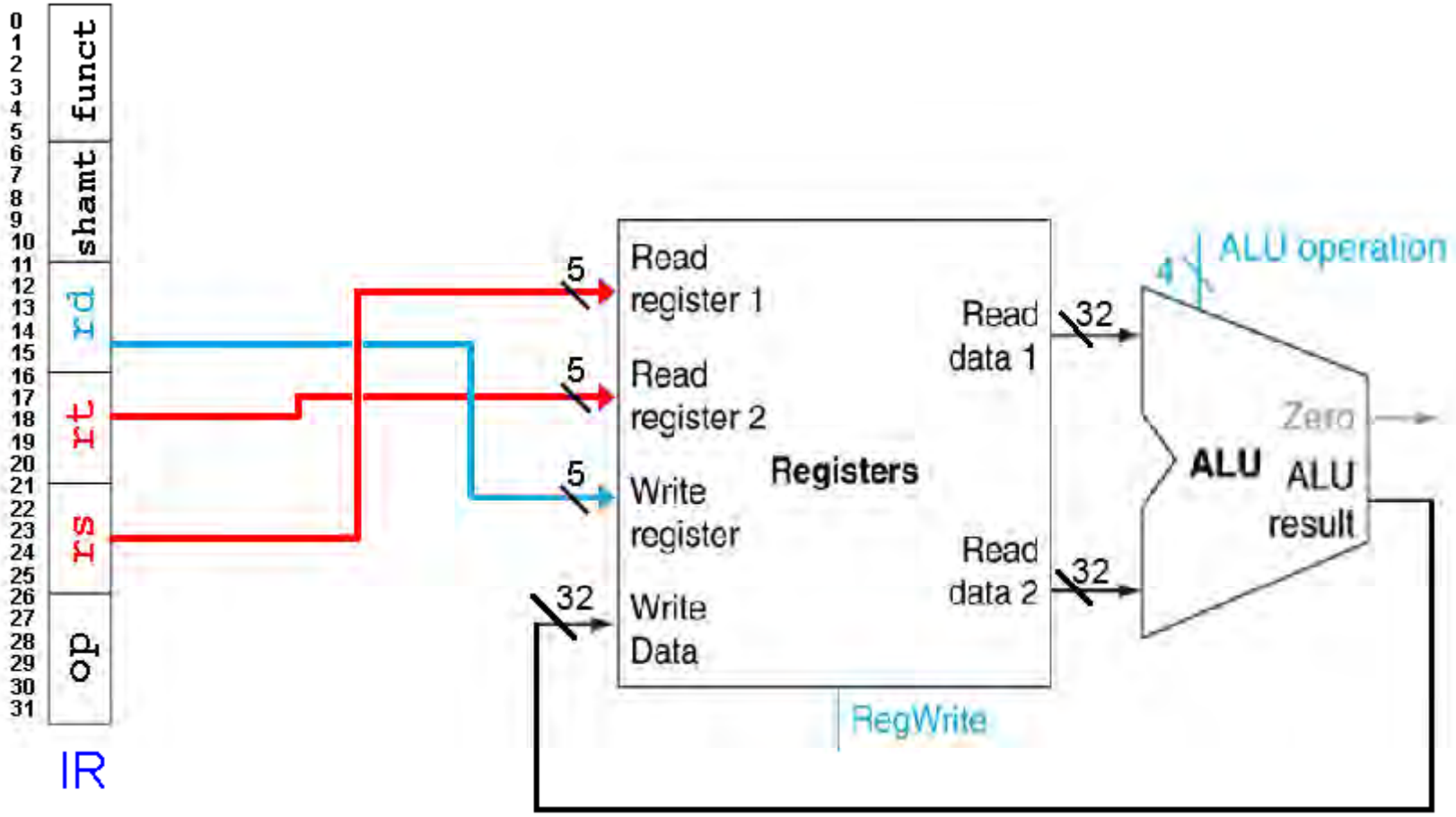
Construindo um Processador Simplificado

- Instruções do Tipo-R



Construindo um Processador Simplificado

- Instruções do Tipo-R

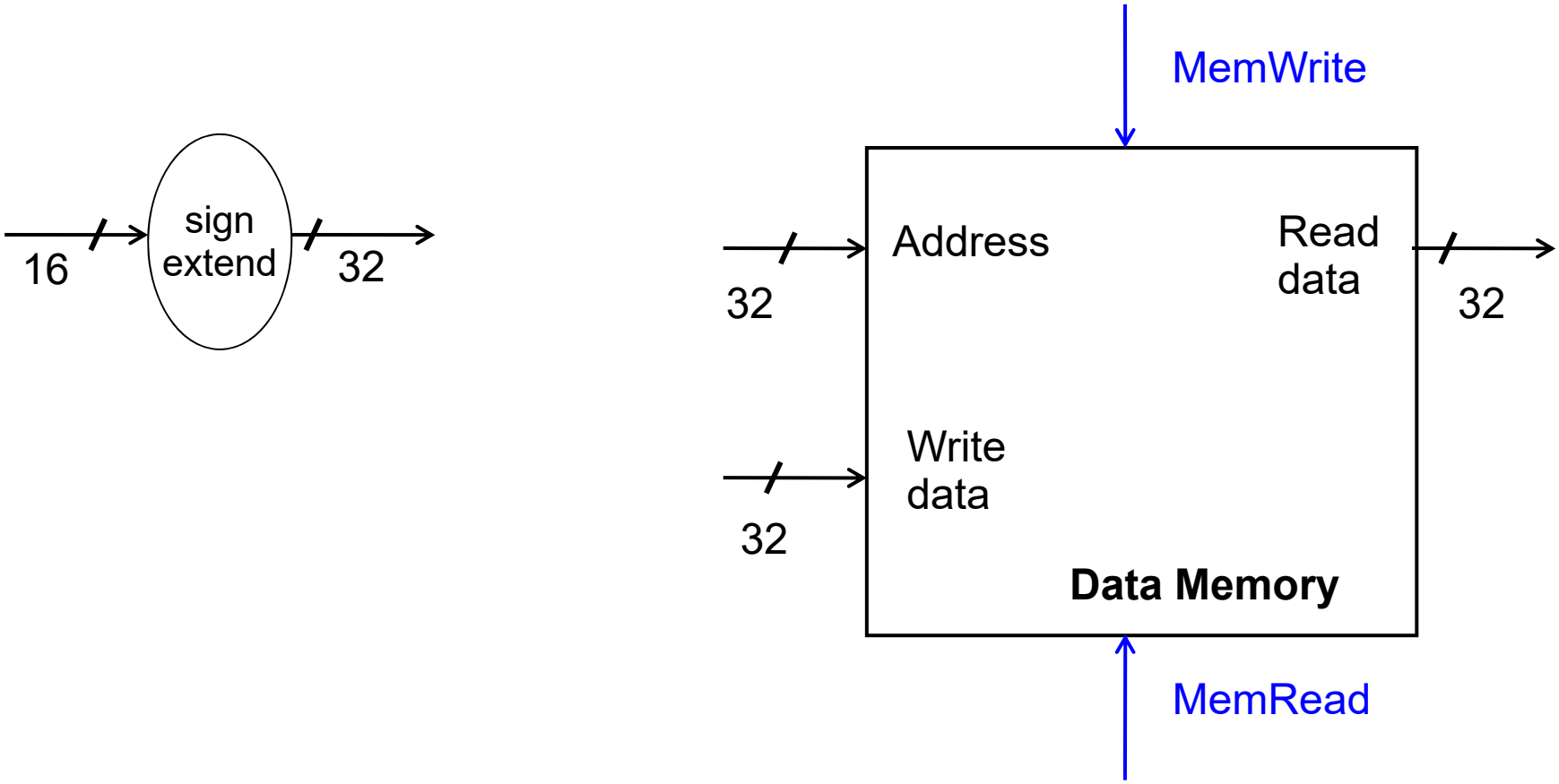


Construindo um Processador Simplificado

- Instruções Load / Store
  - O endereço de memória é computado somando o campo offset de 16 bits ao registrador de entrada (base) de 32 bits (Rs)
  - O offset possui 16-bits, mas os endereços de memória são de 32 bits
    - Assim, o offset deve ser estendido (sign-extended) p/ 32 bits, antes de ser somado ao registrador de entrada
  - A memória de dados tem sinais de controle p/ leitura e escrita
  - Essa instrução utiliza o banco de registradores (ler e armazenar dados) e a ULA para o cálculo do endereço

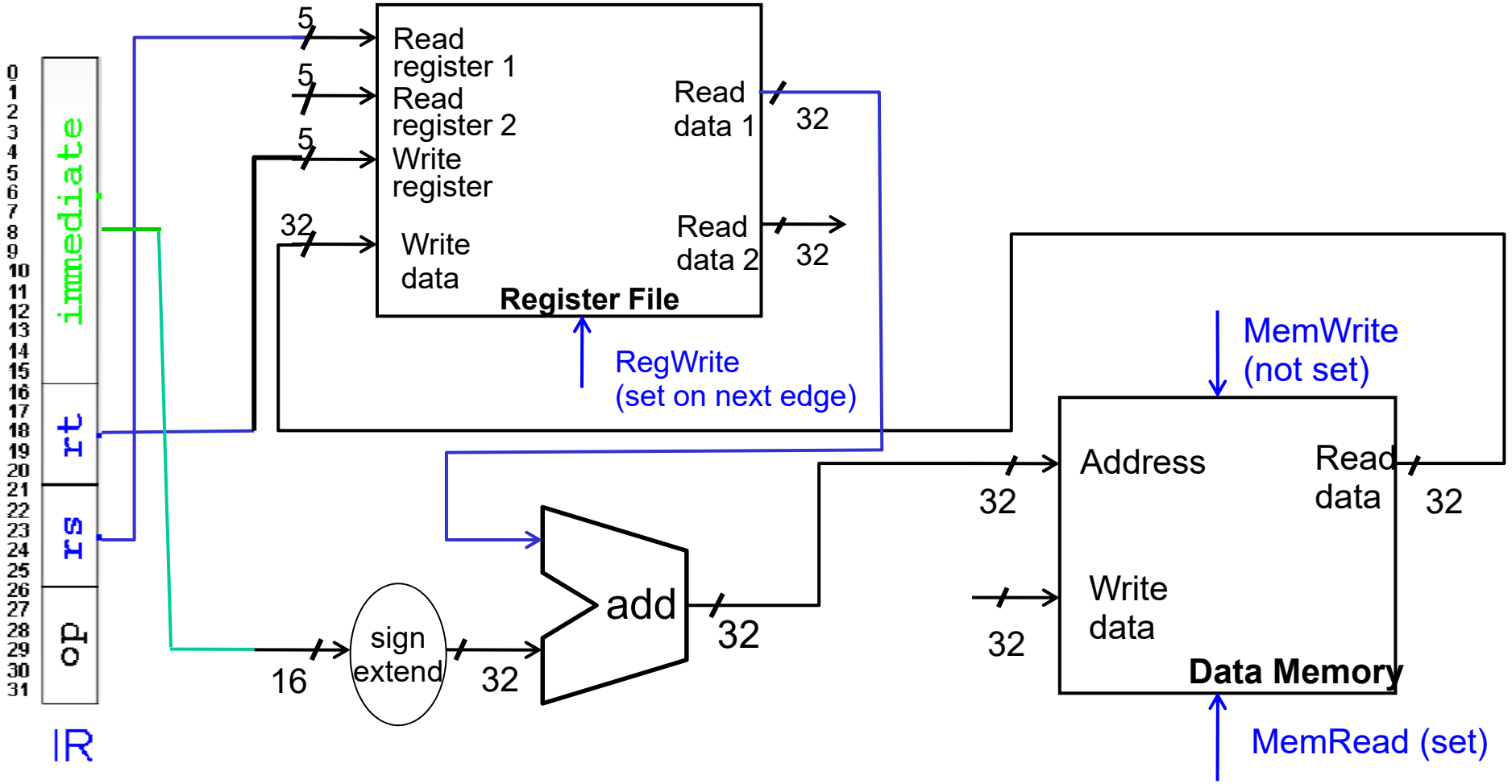
Construindo um Processador Simplificado

- Instruções Load / Store



Construindo um Processador Simplificado

- Instruções Load

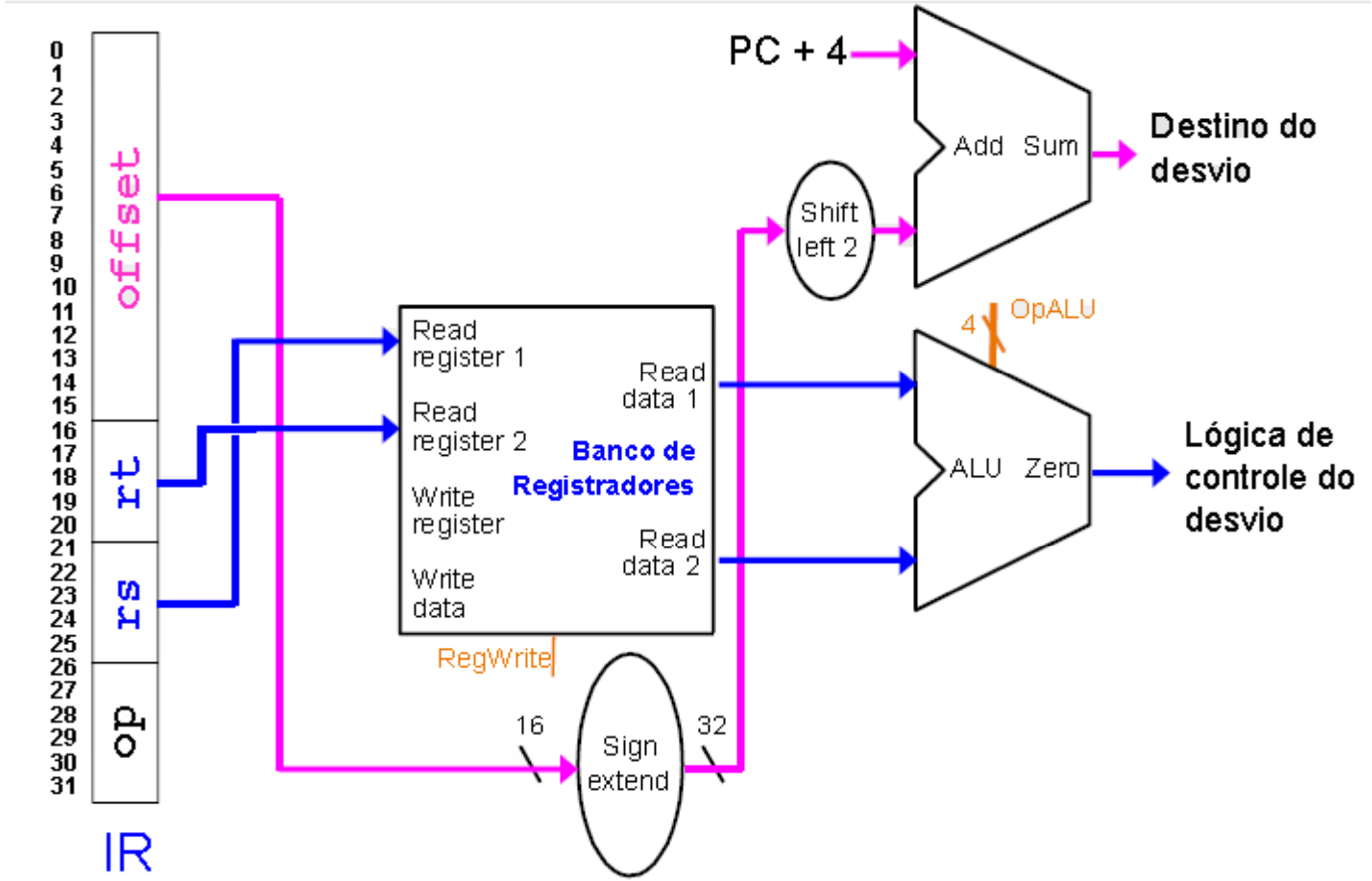


Construindo um Processador Simplificado

- Instruções Branch (Desvio Condicional)
  - Possuem três operandos:
    - Dois registradores a serem comparados
    - Offset de 16 bits para calcular o endereço de destino
    - Ex.: beq t1, t2, offset (Se t1 = t2, desviar p/ a instrução no endereço (PC + offset))
  - Como o deslocamento é constante, não se utiliza um circuito shifter, mas sim um deslocamento de sinais que acrescenta 00 à extremidade direita
  - A comparação é implementada como uma subtração
    - Para operandos iguais, a ALU sinaliza resultado ZERO
  - A depender do resultado da comparação, o novo valor de PC pode ser (PC + 4) ou (PC + 4 + (offset << 2))

Construindo um Processador Simplificado

- Instruções Branch (Desvio Condicional)



Construindo um Processador Simplificado

- Instruções Jump (Desvio Incondicional)
  - É preciso substituir os 28 bits menos significativos do PC pelos 26 bits contidos no campo da instrução deslocados dois dígitos à esquerda (i.e., concatenados com 00).
  - Esta instrução é bastante simples: o endereço do desvio (target) é a concatenação de:
    - 4 bits de ordem mais alta de (PC+4)
    - 26 bits do campo imediato da instrução
    - 00

Concatena 00 à direita do target - Por que ?

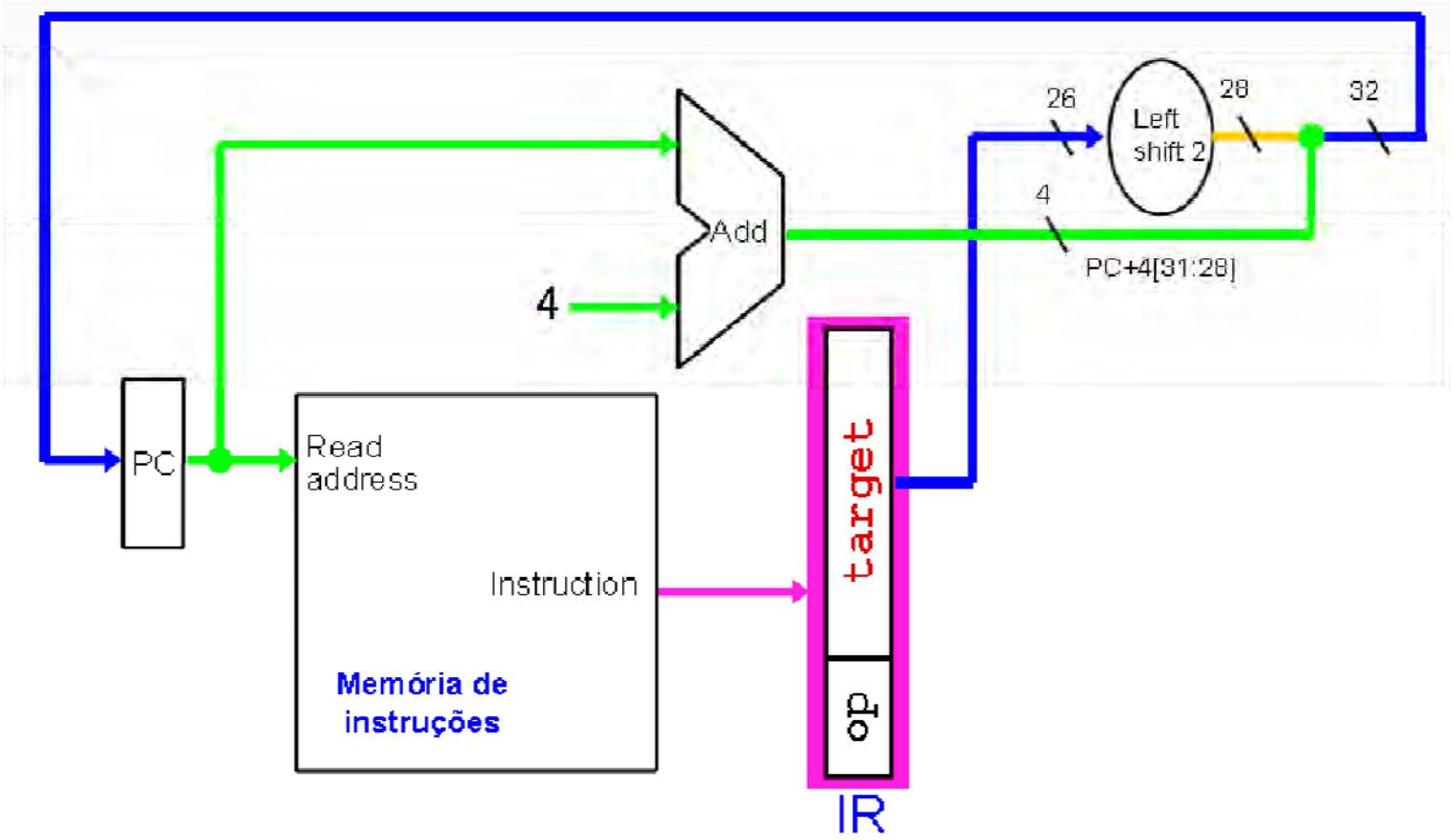
Na instrução, target é especificado em words (4 bytes).

O endereço de desvio computado tem os 2 últimos bits= 00, pois é um múltiplo de words (0, 4, 8, 12, ..).

Por isso, o target é concatenado c/ 00 à direita (shift left), p... produzir um endereço em bytes p/ o PC.

Construindo um Processador Simplificado

- Instruções Jump (Desvio Incondicional)



Construindo um Processador Simplificado

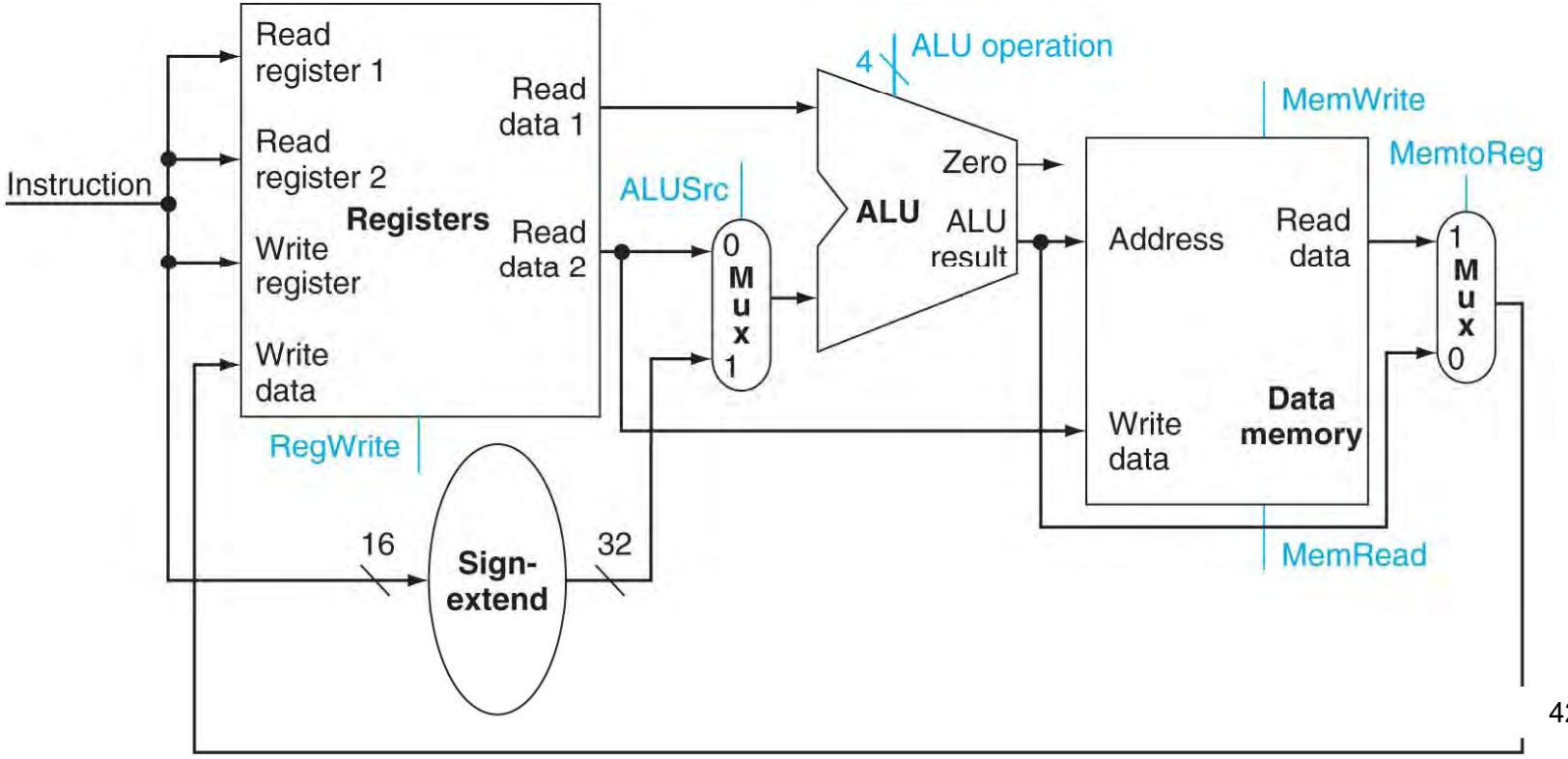
- Combinando Todas as Instruções
  - Os projetos anteriores podem ser combinados para formar um único datapath
  - Em sua forma mais simples, ele será capaz de executar qualquer uma das instruções em um único ciclo.
  - Nesse caso, todo elemento do datapath é usado no máximo uma vez a cada ciclo de clock
  - Nenhuma duplicação de hardware é necessária (talvez apenas alguns somadores)

Construindo um Processador Simplificado

- Combinando Todas as Instruções
  - Incialmente, vamos combinar os datapaths p/ instruções R-Type (ALU) e de acesso à memória (L/S)
  - Para isso, adicionamos um multiplexador para escolher o datapath p/ a ALU, ou p/ a Memória (determinando ainda os sinais de controle correspondentes)

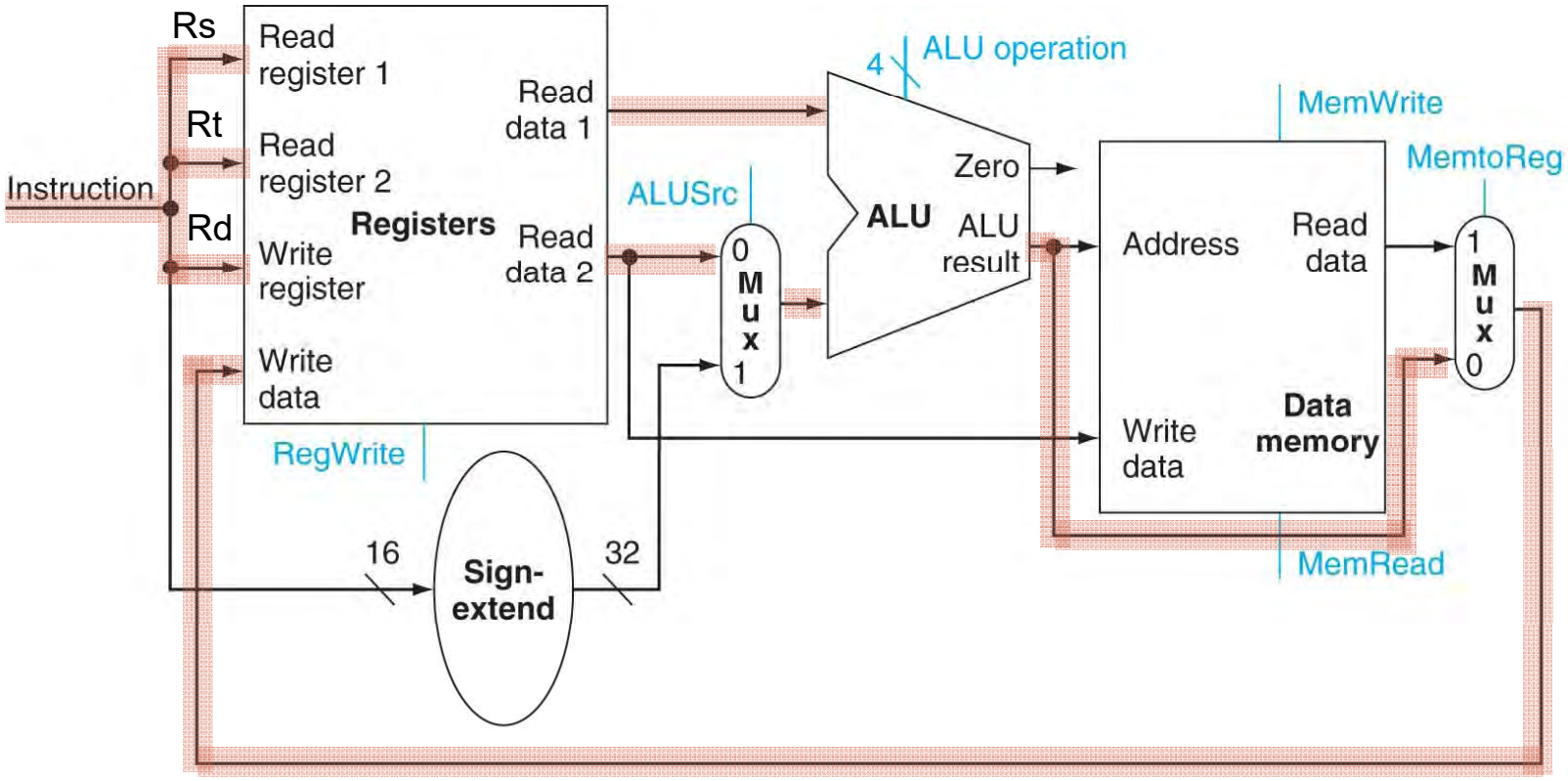
Construindo um Processador Simplificado

- Combinando Todas as Instruções
  - R-Type (ALU) e de acesso à memória (L/S)



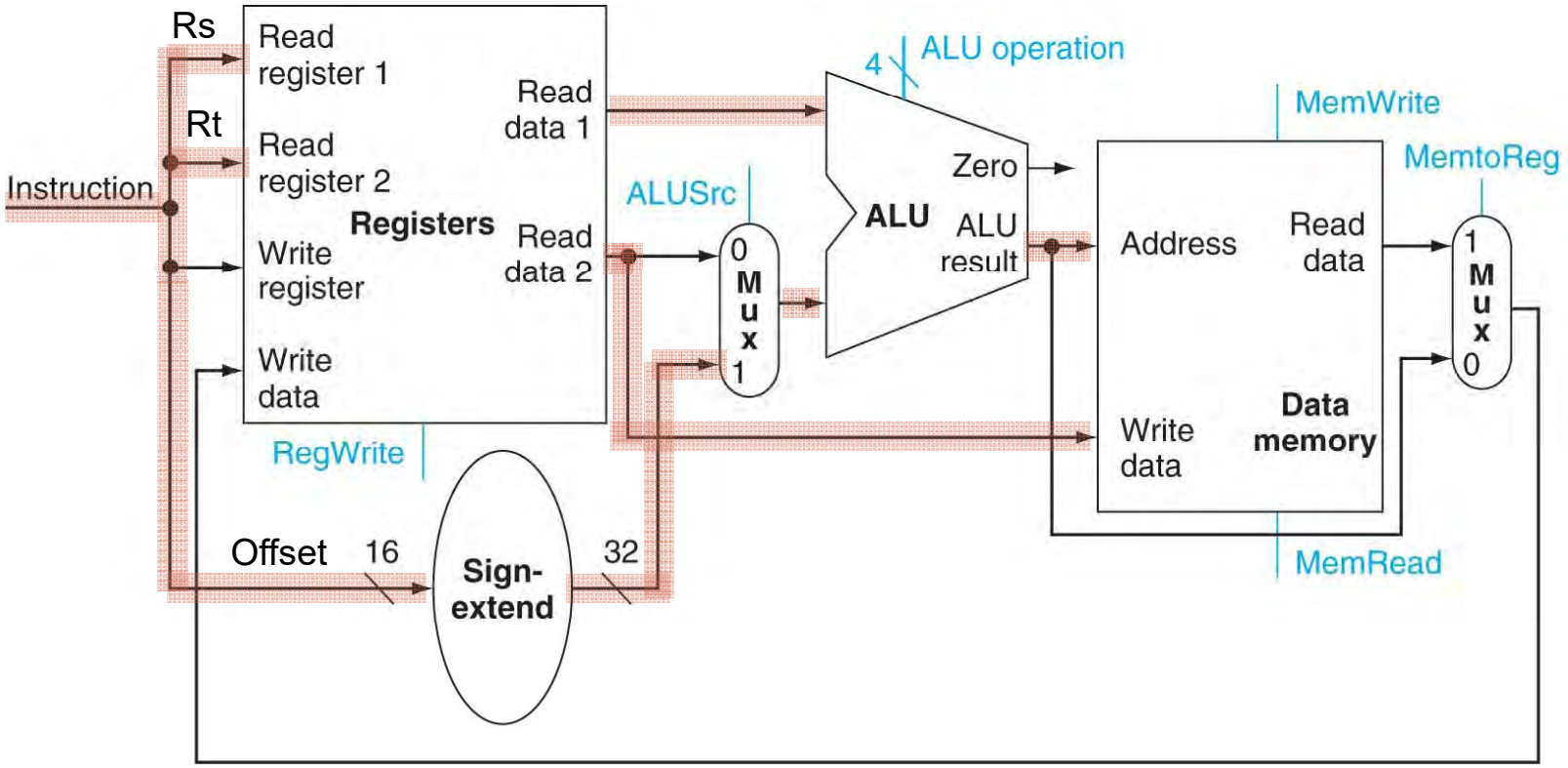
Construindo um Processador Simplificado

- Combinando Todas as Instruções
  - add, $t0, $s1, $s2



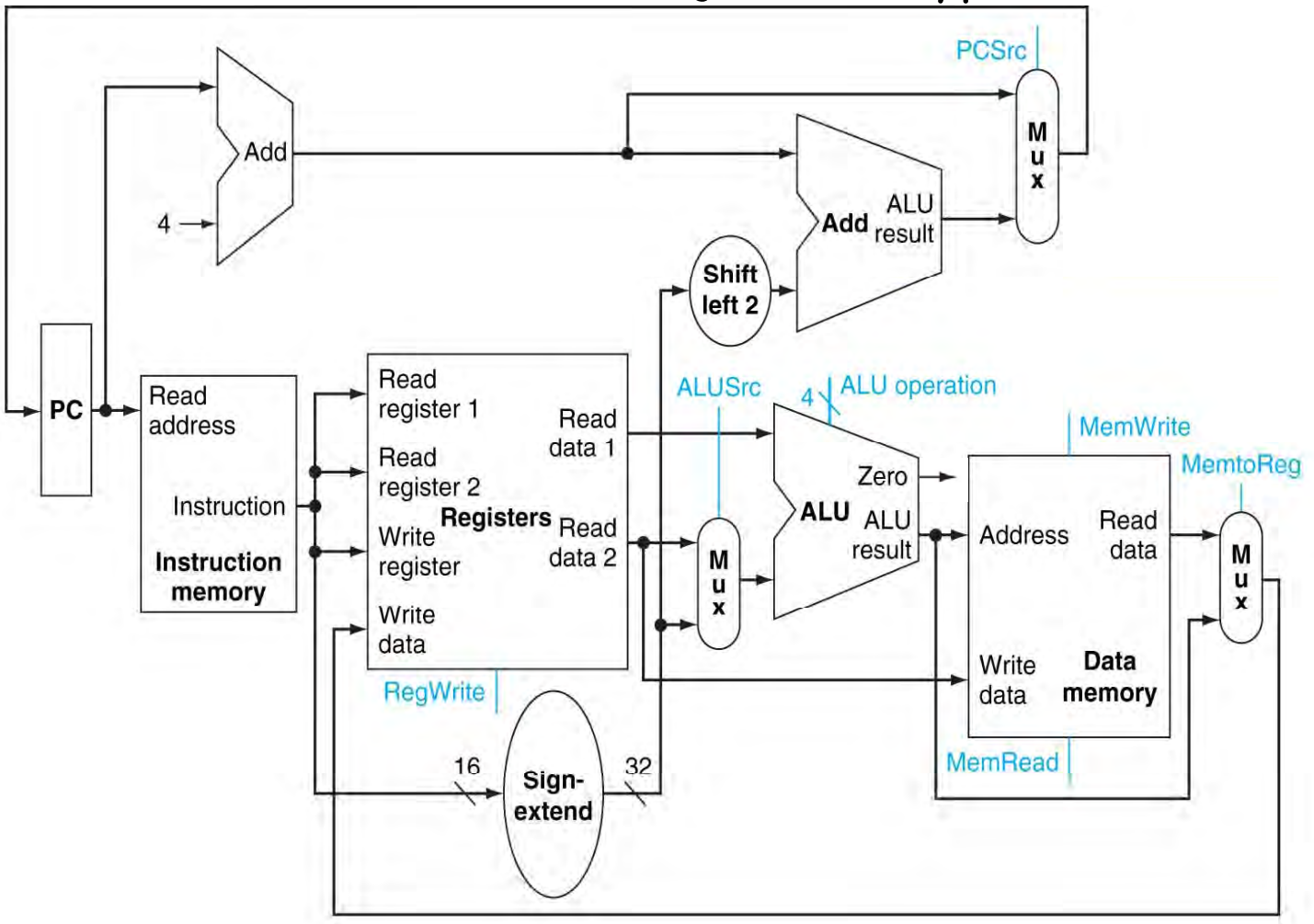
Construindo um Processador Simplificado

- Combinando Todas as Instruções
  - sw, $t0, 16($s2)



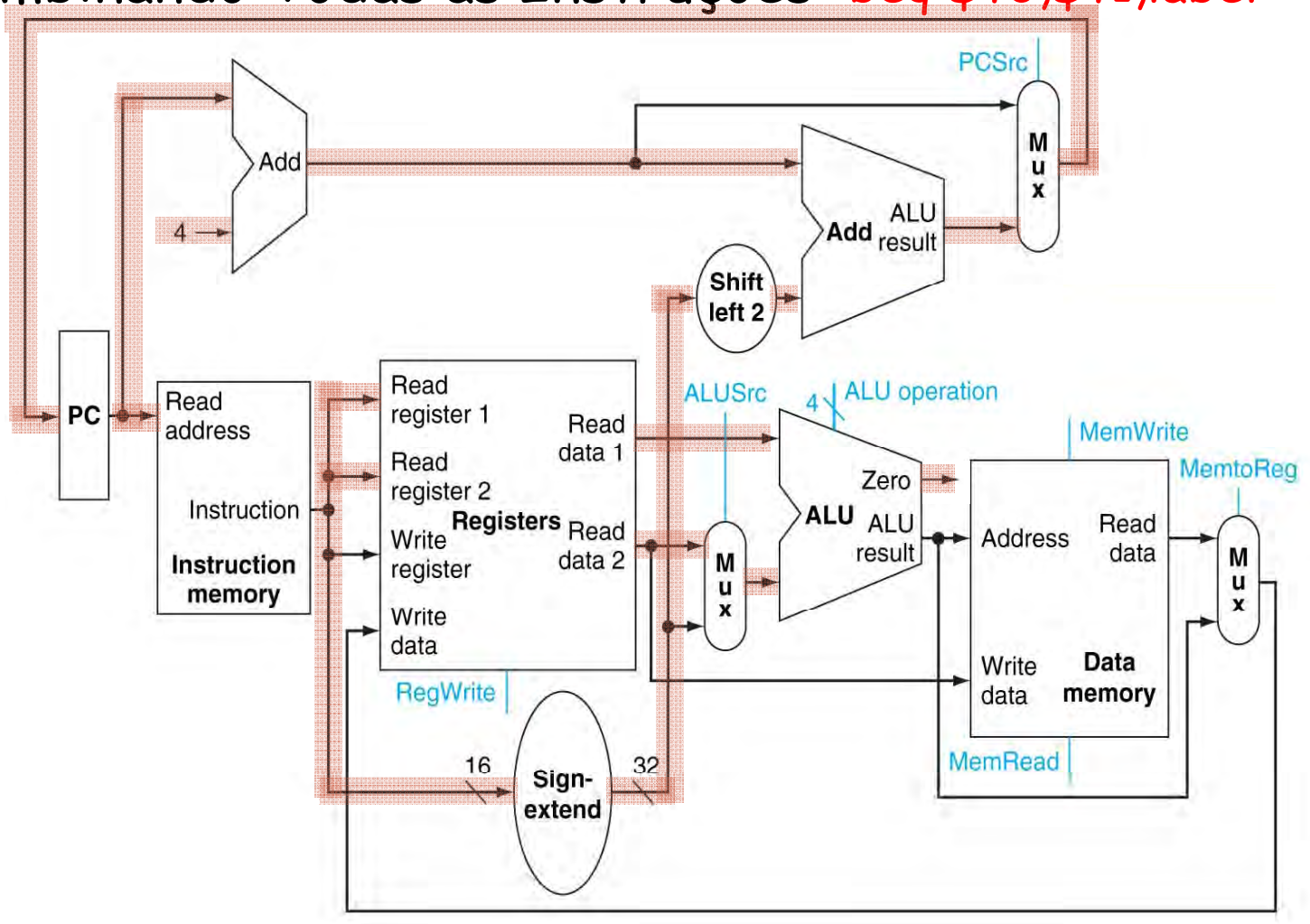
Construindo um Processador Simplificado

- Combinando Todas as Instruções: R-Type, L/S e Branch



Construindo um Processador Simplificado

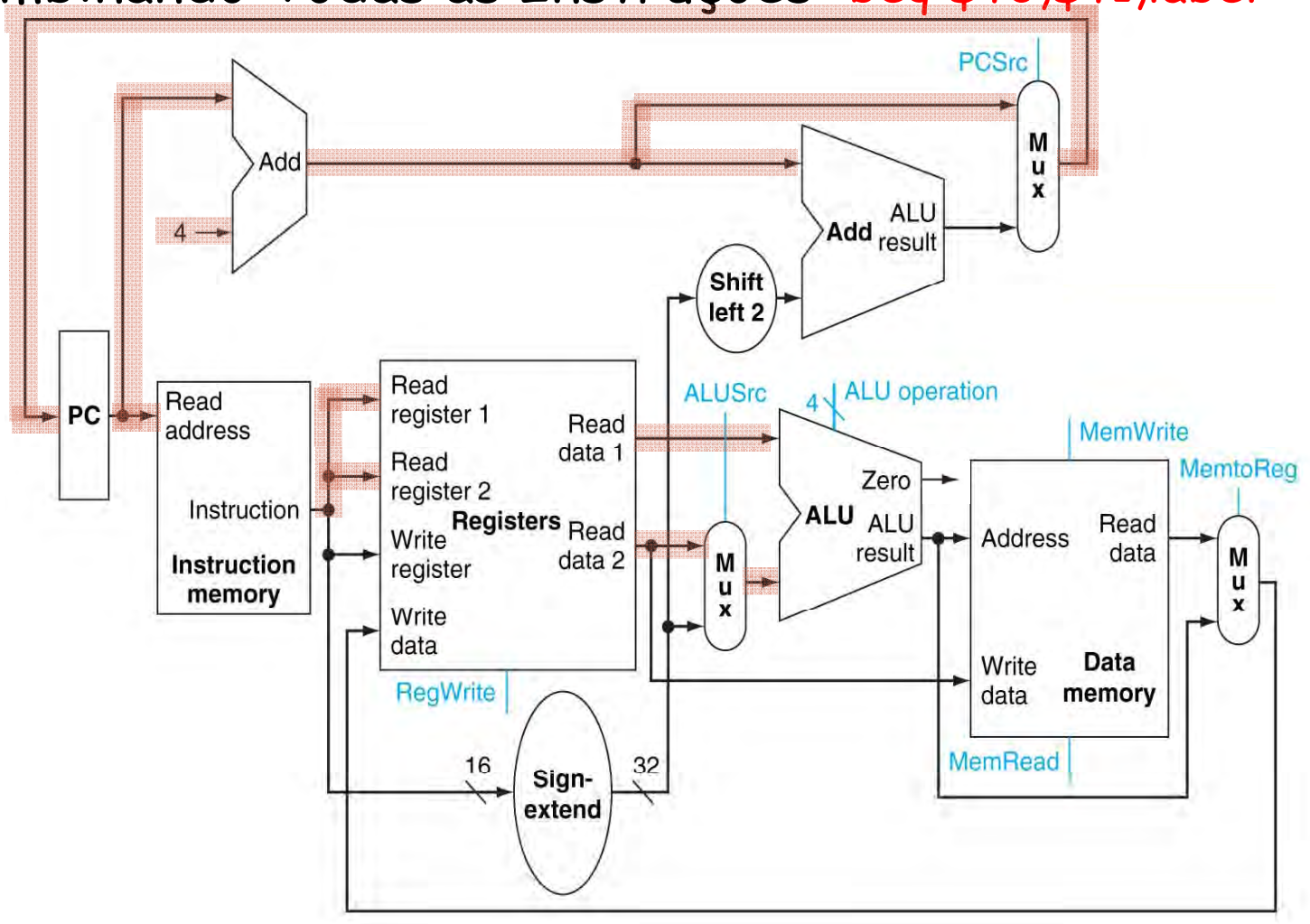
- Combinando Todas as Instruções: beq $t0,$t1,label



$t0 = $t1

Construindo um Processador Simplificado

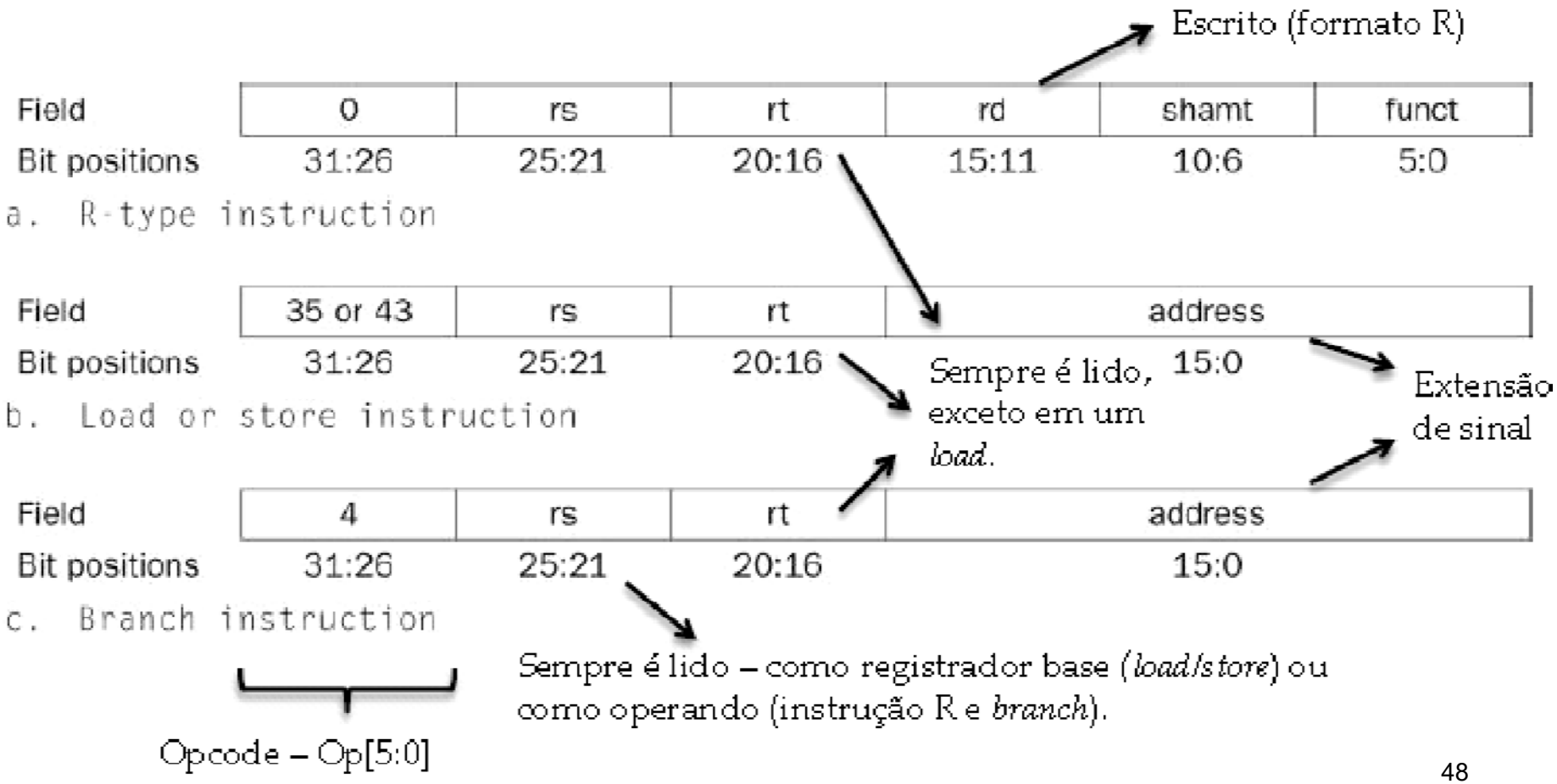
- Combinando Todas as Instruções: beq $t0,$t1,label



$t0 != $t1

Construindo um Processador Simplificado

- Relembrando...

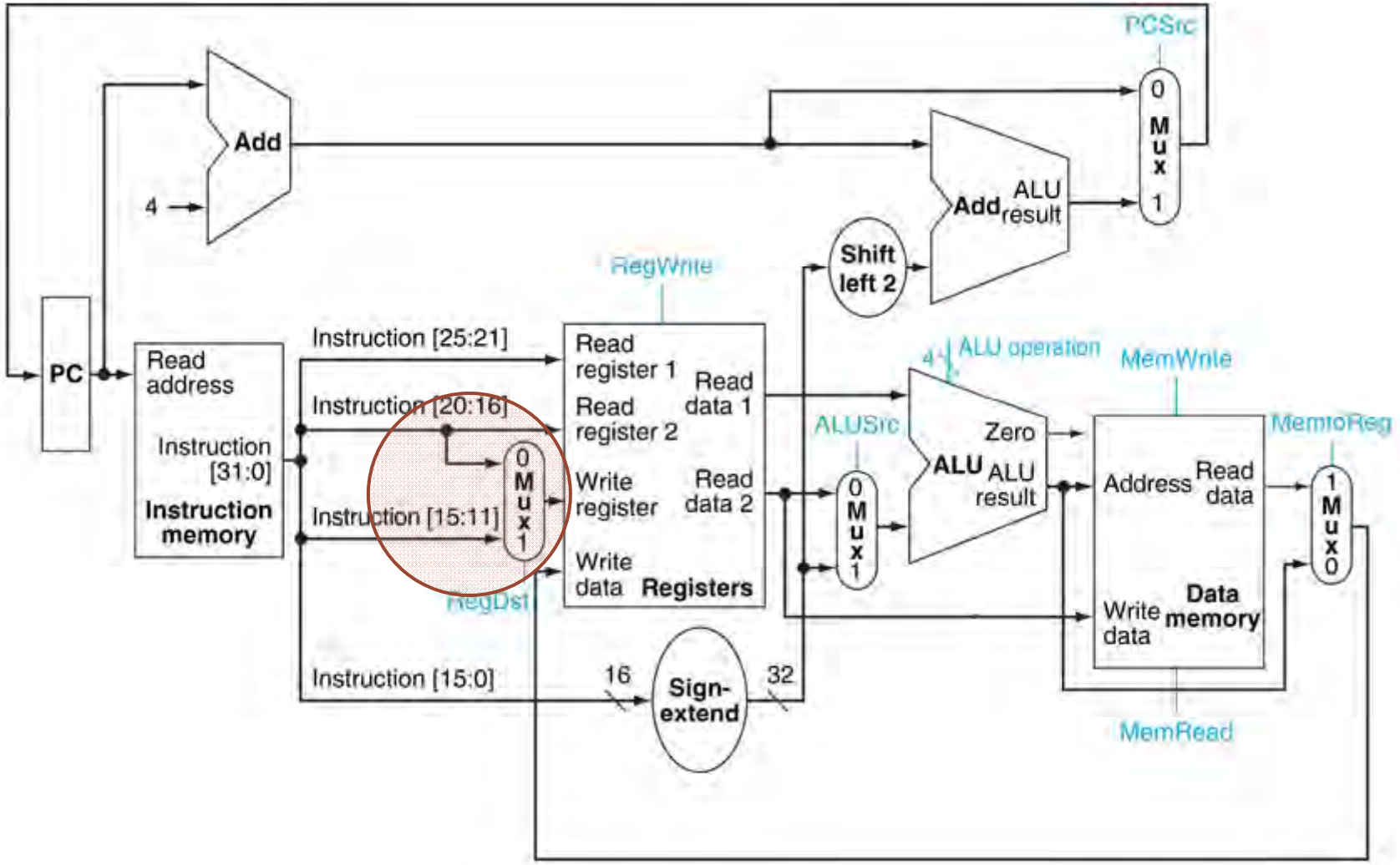


Construindo um Processador Simplificado

- Relembrando...
  - Para sw, campo Rt (bits 20-16) designam registrador cujo conteúdo será escrito na memória de dados
  - Para lw, Rt (bits 20-16) designam registrador que será carregado com valor lido da memória de dados
  - Para lw, o endereço do registrador a ser escrito está no campo Rt (bits 20-16)
  - Para instruções tipo R, o endereço do registrador a ser escrito está no campo Rd (bits 15-11)

Construindo um Processador Simplificado

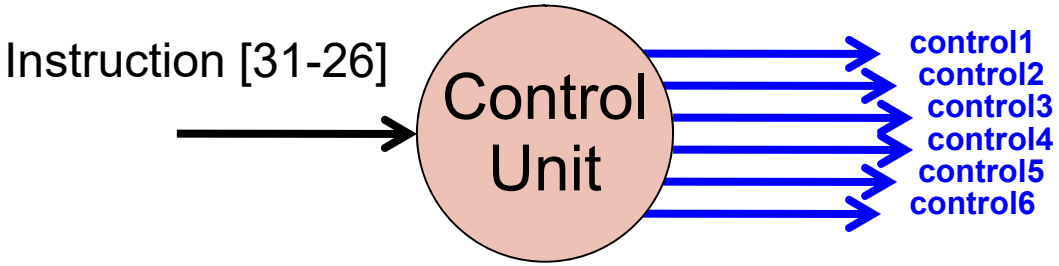
- Ajustando...



Unidade de Controle

Unidade de Controle

- A Unidade de Controle deve através dos sinais de controle:
  - Selecionar as operações para executar (ULA, leitura/escrita, etc.)
  - Controlar o fluxo de dados (entradas para o multiplexador)
- Baseia-se no opcode da instrução para determinar o valor de uma série de bits de controle
- Sua operação é definida por uma tabela verdade



opcode	c1	c2	c3	c4	c5	c6
000000	1	X	0	X	0	1
000001	1	1	X	0	0	0
. . . . . .						
111110	1	x	0	x	x	0
111111	0	0	x	0	1	x

X = não importa

Unidade de Controle

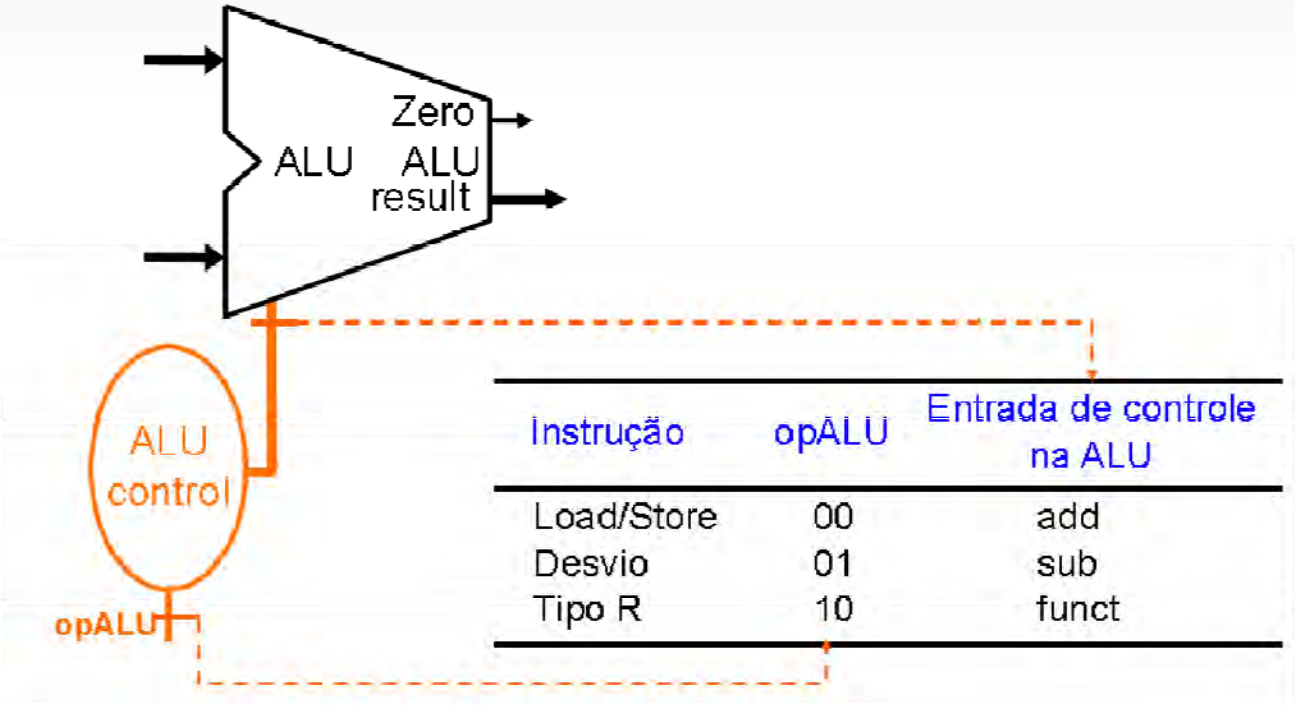
- Controle da ULA

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than

  - Instruções do tipo R: há cinco ações possíveis para a ALU.
    - Todas possuem opcode igual a 00000.
    - Diferem no campo funct (6 bits).
  - Instruções load/store: usam a ALU para uma adição (soma conteúdo do registrador base com offset para obter o endereço de memória)
  - Instrução branch: usa a ALU para uma subtração – que, por sua vez, implementa uma comparação.

Unidade de Controle

- Controle da ULA
  - Dois bits (ULAop) são usados para indicar se a operação esperada da ULA é uma soma (load/store - 00), subtração (branch - 01) ou se é determinada pelo campo funct (10).

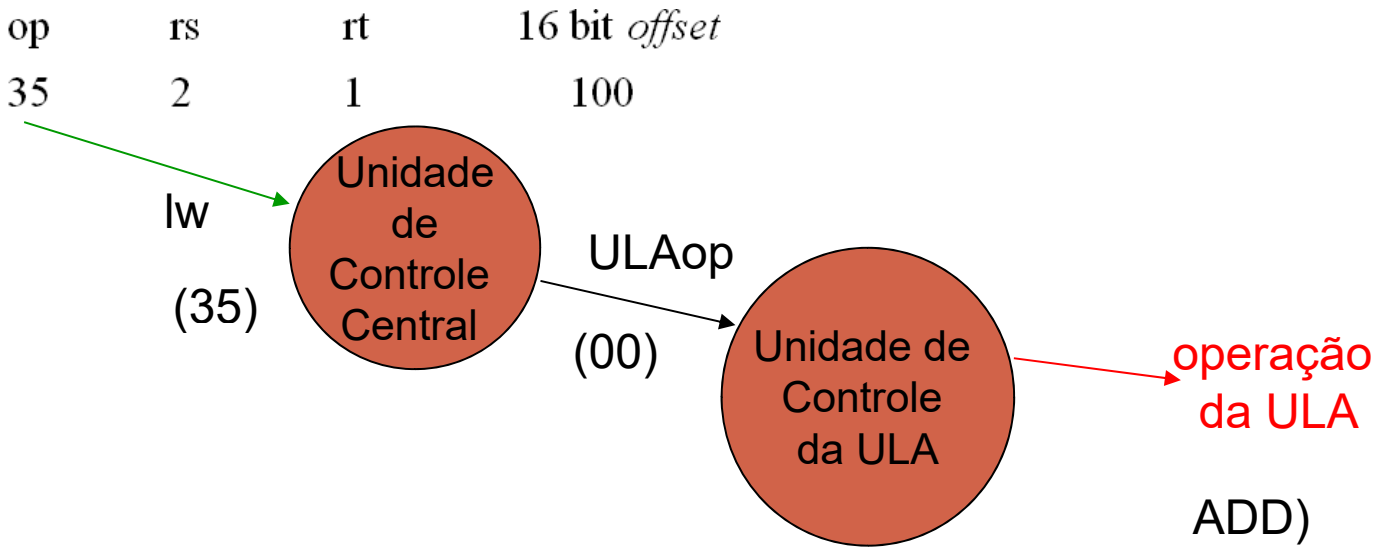


Instrução	opALU	Entrada de controle na ALU
Load/Store	00	add
Desvio	01	sub
Tipo R	10	funct

Unidade de Controle

- Controle da ULA
  - Ex 1.: Para a instrução load-word a operação da ULA só depende do opcode.

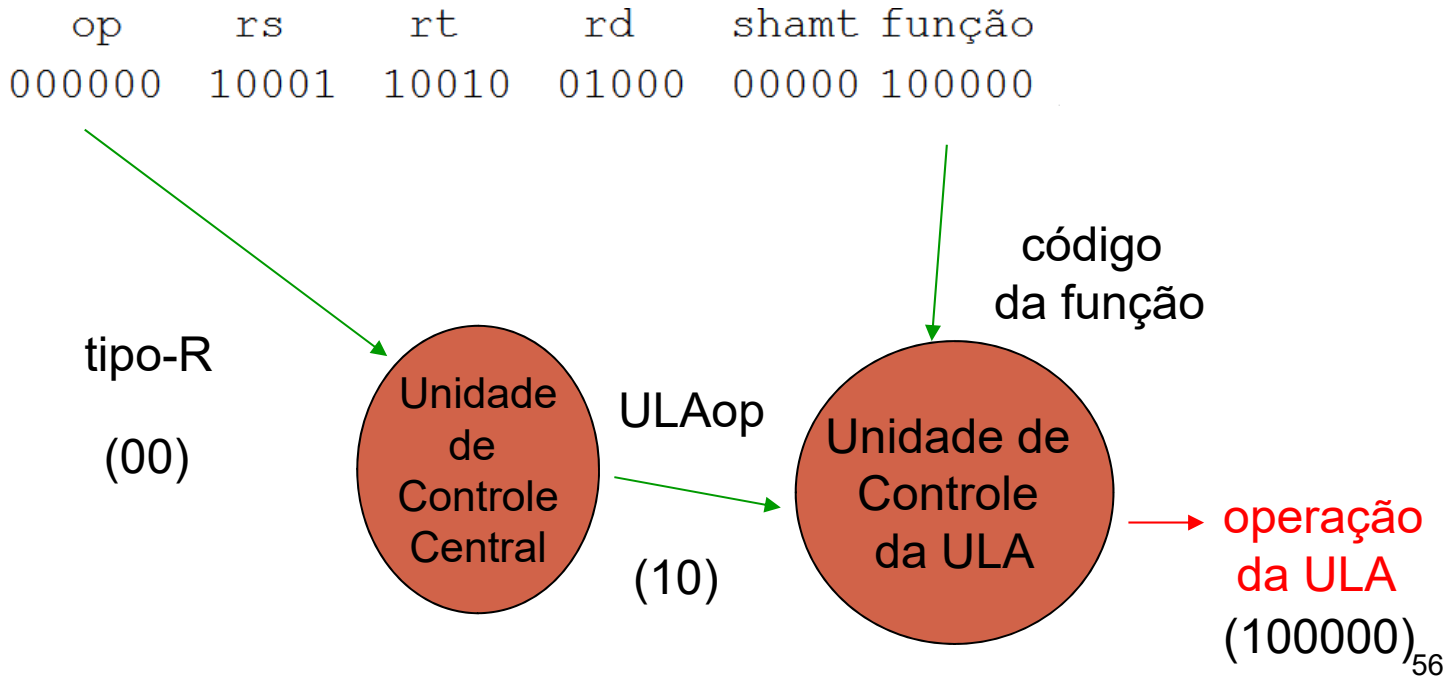
lw $1, 100($2)



Unidade de Controle

- Controle da ULA
  - Ex 2.: Para a instrução add a operação da ULA também depende do código da função

add $8, $17, $18



Unidade de Controle

- Controle da ULA

Mnemonic	Meaning	Type	Opcode	Funct
add	Add	R	0x00	0x20
addu	Add Unsigned	R	0x00	0x21
and	Bitwise AND	R	0x00	0x24
div	Divide	R	0x00	0x1A
divu	Unsigned Divide	R	0x00	0x1B
jr	Jump to Address in Register	R	0x00	0x08
mfhi	Move from HI Register	R	0x00	0x10
mflo	Move from LO Register	R	0x00	0x12
mfc0	Move from Coprocessor 0	R	0x10	NA
mult	Multiply	R	0x00	0x18
multu	Unsigned Multiply	R	0x00	0x19
nor	Bitwise NOR (NOT-OR)	R	0x00	0x27
xor	Bitwise XOR (Exclusive-OR)	R	0x00	0x26
or	Bitwise OR	R	0x00	0x25
slt	Set to 1 if Less Than	R	0x00	0x2A
sltu	Set to 1 if Less Than Unsigned	R	0x00	0x2B
sll	Logical Shift Left	R	0x00	0x00
srl	Logical Shift Right (0-extended)	R	0x00	0x02

Mnemonic	Meaning	Type	Opcode	Funct
sra	Arithmetic Shift Right (sign-extended)	R	0x00	0x03
sub	Subtract	R	0x00	0x22
subu	Unsigned Subtract	R	0x00	0x23
j	Jump to Address	J	0x02	NA
jal	Jump and Link	J	0x03	NA
addi	Add Immediate	I	0x08	NA
addiu	Add Unsigned Immediate	I	0x09	NA
andi	Bitwise AND Immediate	I	0x0C	NA
beq	Branch if Equal	I	0x04	NA
bne	Branch if Not Equal	I	0x05	NA
lbu	Load Byte Unsigned	I	0x24	NA
lhu	Load Halfword Unsigned	I	0x25	NA
lui	Load Upper Immediate	I	0x0F	NA
lw	Load Word	I	0x23	NA
ori	Bitwise OR Immediate	I	0x0D	NA
sb	Store Byte	I	0x28	NA
sh	Store Halfword	I	0x29	NA
slti	Set to 1 if Less Than Immediate	I	0x0A	NA
sltiu	Set to 1 if Less Than Unsigned Immediate	I	0x0B	NA
sw	Store Word	I	0x2B	NA

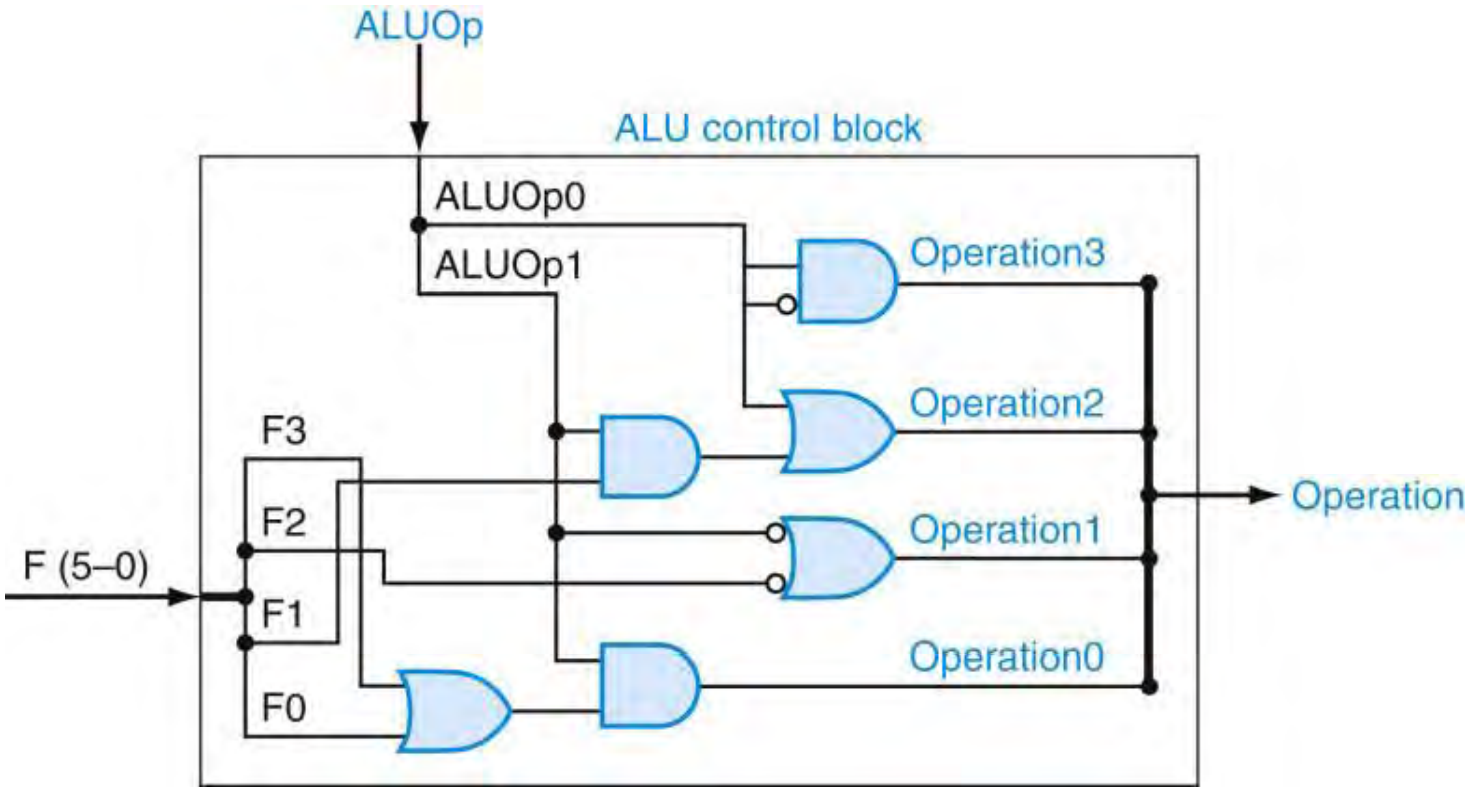
Unidade de Controle

- Controle da ULA
  - É possível simplificar a geração dos bits de controle da ALU a partir de ALUOp e do campo funct.

ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
0	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

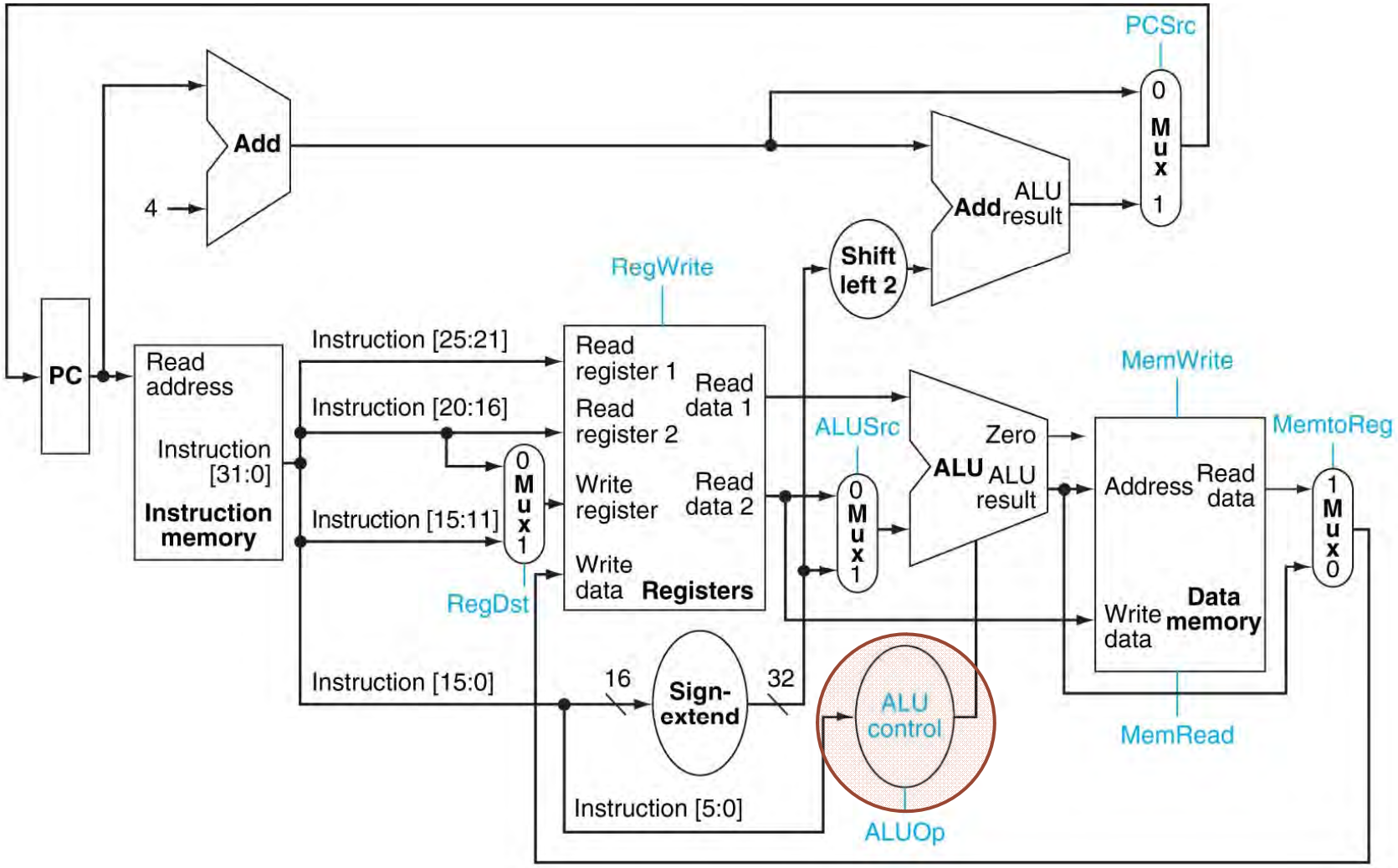
Unidade de Controle

- Controle da ULA
  - Fazendo a simplificação das expressões lógicas, é possível chegar ao seguinte circuito combinacional:



Unidade de Controle

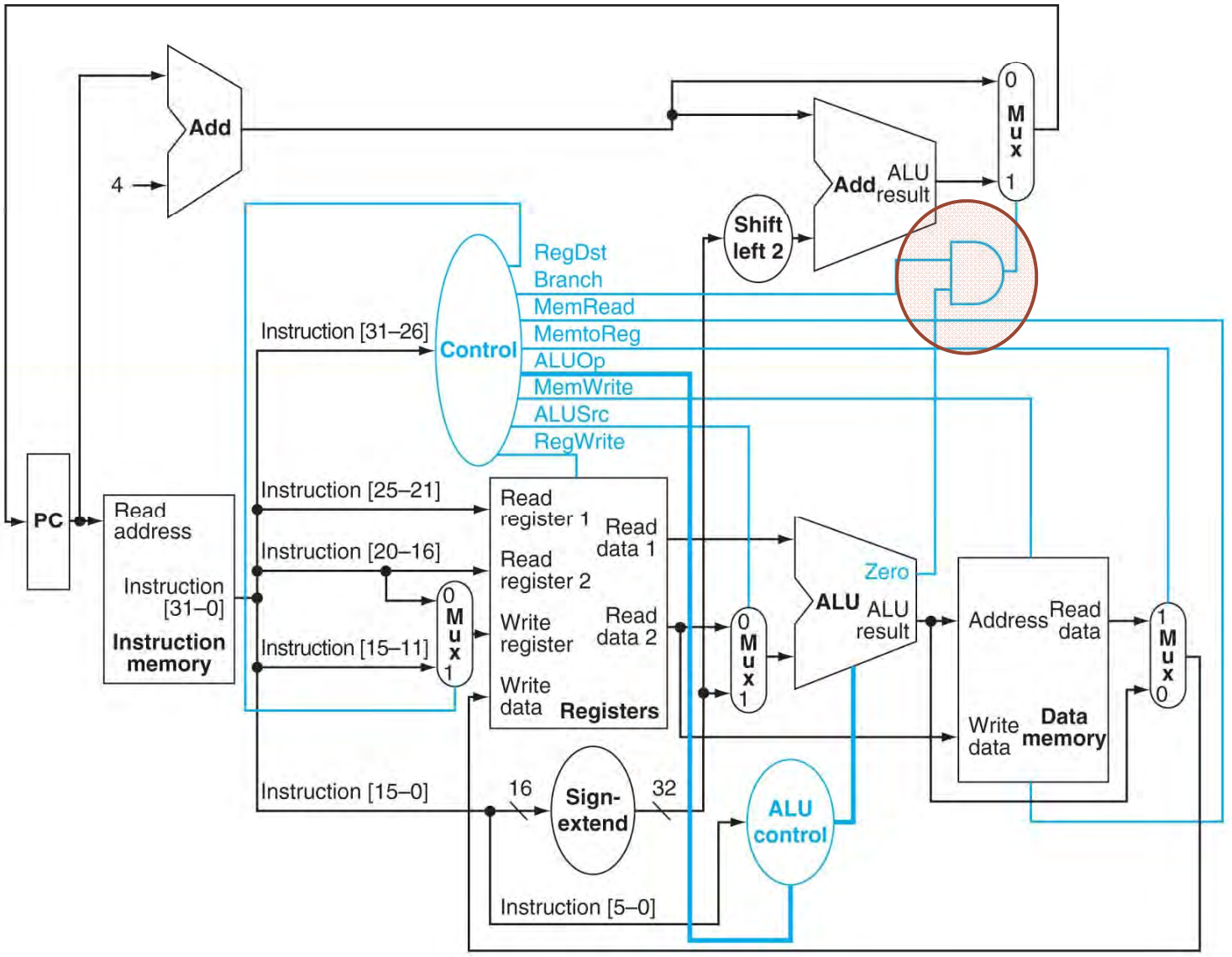
- Ajustando...



Unidade de Controle

- Linhas de Controle: valores
  - Todos os elementos de estado (registradores, memória) têm o clock como uma entrada implícita
  - Unidade de controle pode definir todos os sinais de controle baseada no campo opcode da instrução, exceto Branch, que depende da saída zero da ALU no caso de instruções de desvio

Unidade de Controle

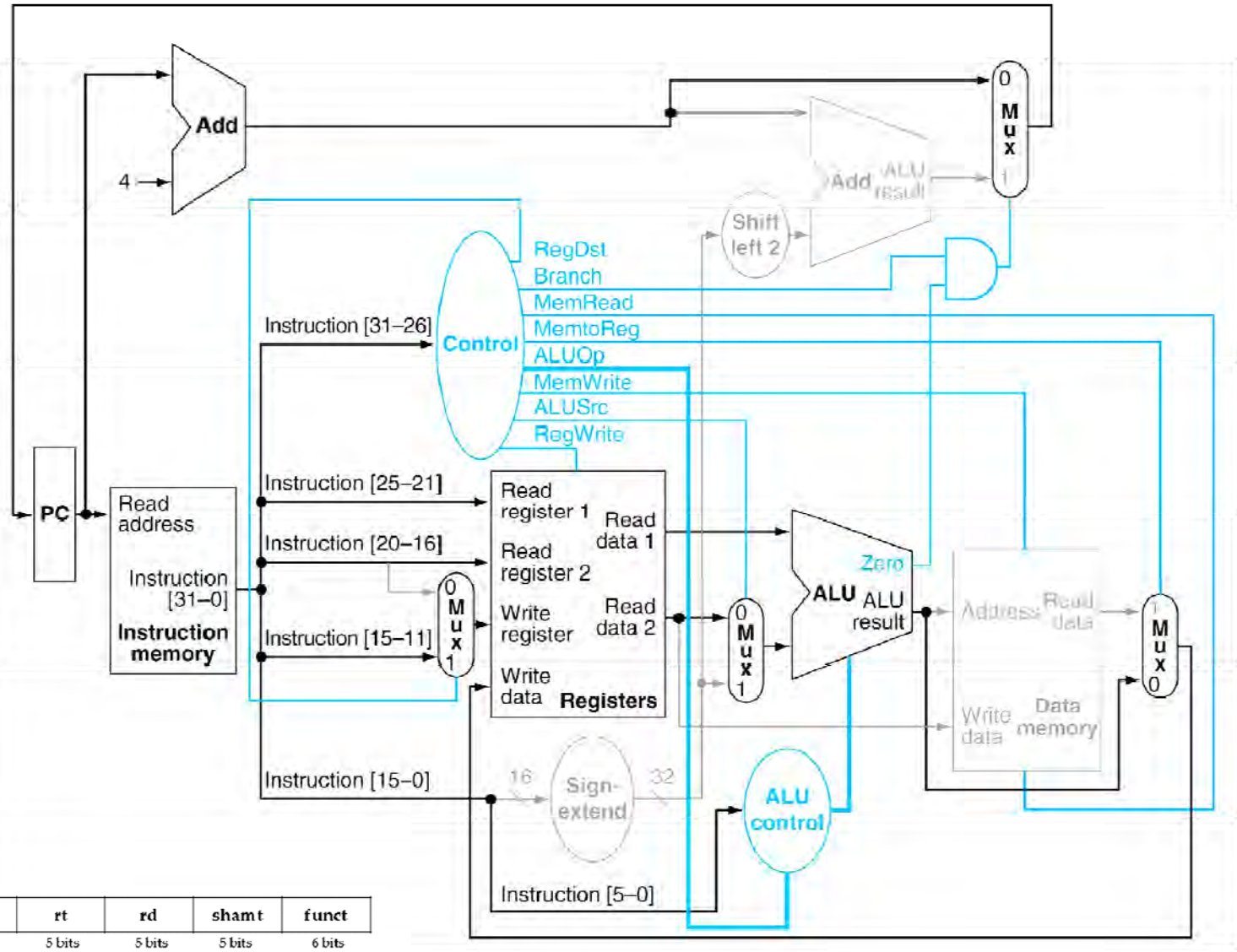


Unidade de Controle

- Linhas de Controle

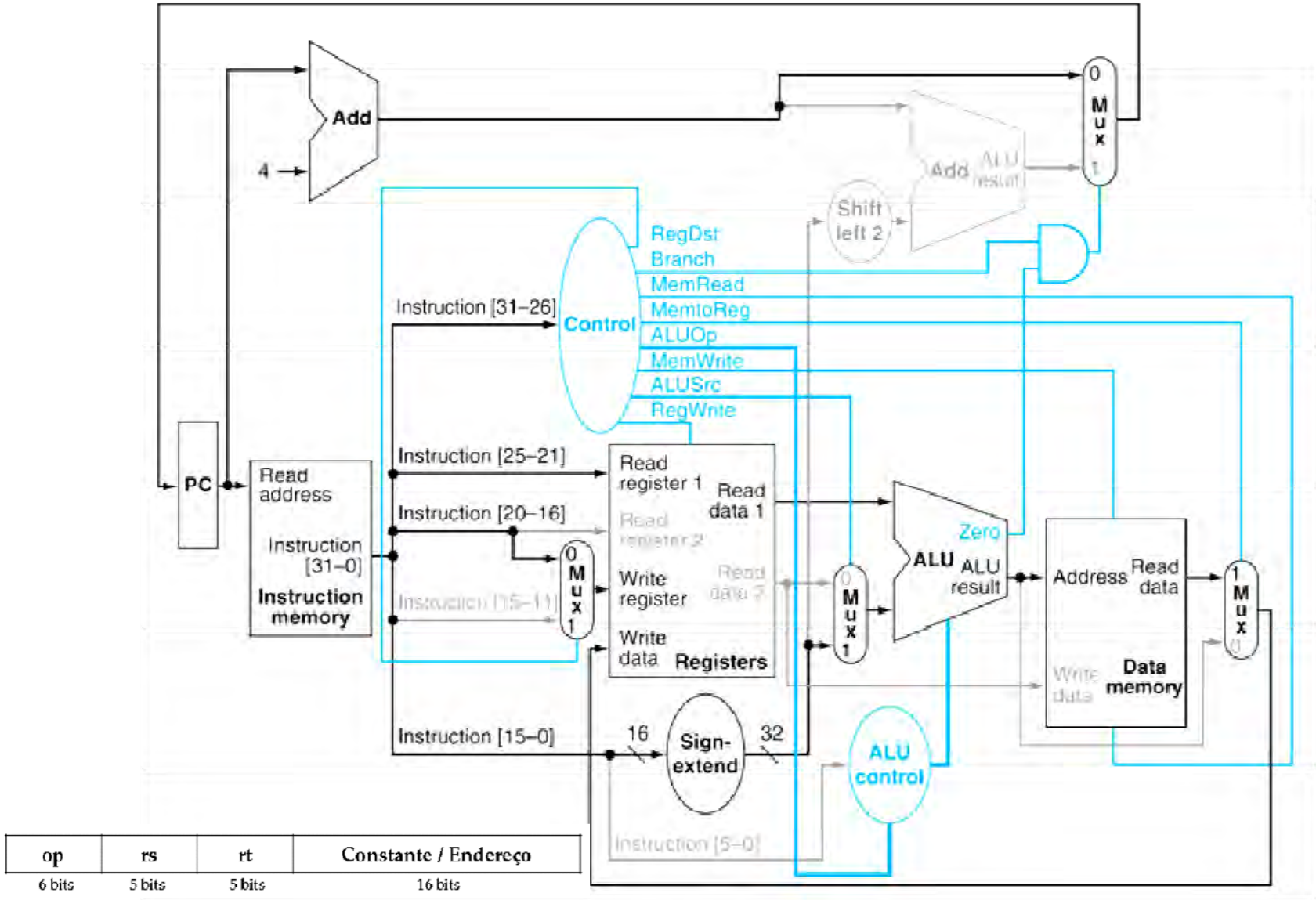
Sinal de Controle	Efeito quando inativo (zero)	Efeito quando ativo (um)
RegDst	O registrador de destino é indicado no campo **Rt** (bits 20:16)	O registrador de destino é indicado no campo **Rd** (bits 15:11)
RegWrite	Nenhum	O registrador destino é escrito com o valor da entrada Write data
OrigALU	O 2º operando da ALU é um registrador (Read data 2)	O 2º operando da ALU é o campo imm estendido para 32 *bits*
Branch	O PC é substituído pelo *output* do somador que calcula **PC+4**	O PC é substituído pelo *output* do somador que calcula o destino do salto
MemRead	Nenhum	O conteúdo da posição de memória endereçada é colocado nas saídas
MemWrite	Nenhum	O conteúdo da posição de memória endereçada é reescrito
MemtoReg	O valor a escrever no registrador é o resultado da ALU	O valor a escrever no registrador é lido da memória de dados

Datapath Utilizado p/ Instruções do Tipo-R



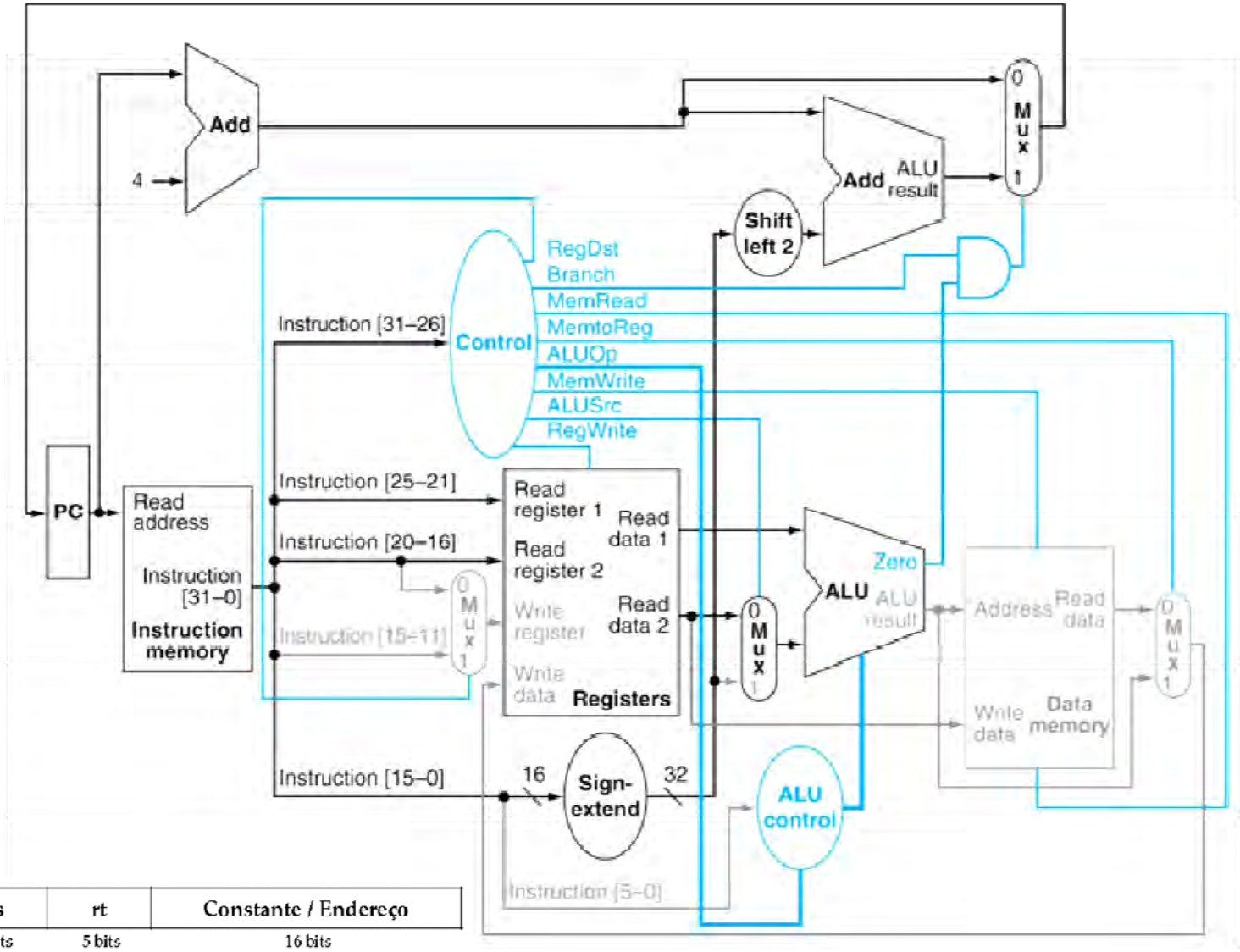
op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Datapath Utilizado p/ Instruções Load



op	rs	rt	Constante / Endereço
6 bits	5 bits	5 bits	16 bits

Datapath Utilizado p/ Instrução BEQ



op	rs	rt	Constante / Endereço
6 bits	5 bits	5 bits	16 bits

Unidade de Controle

- Linhas de Controle: valores

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

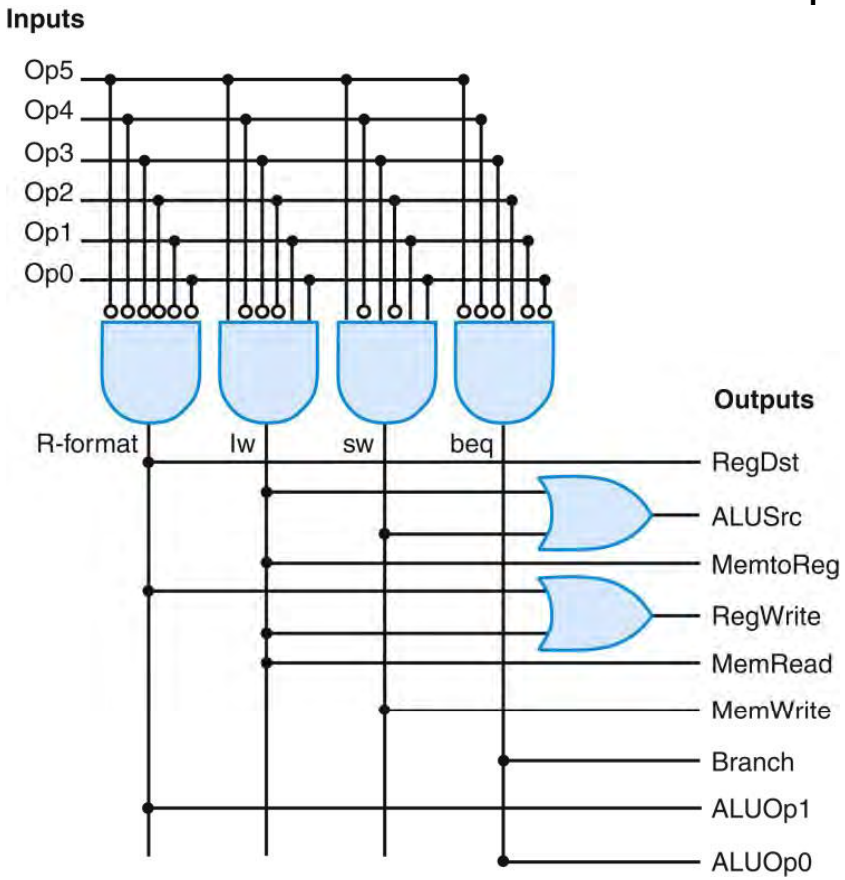
Unidade de Controle

- Linhas de Controle
  - Os níveis lógicos dos sinais de controle gerados pela unidade central dependem exclusivamente do opcode da instrução.

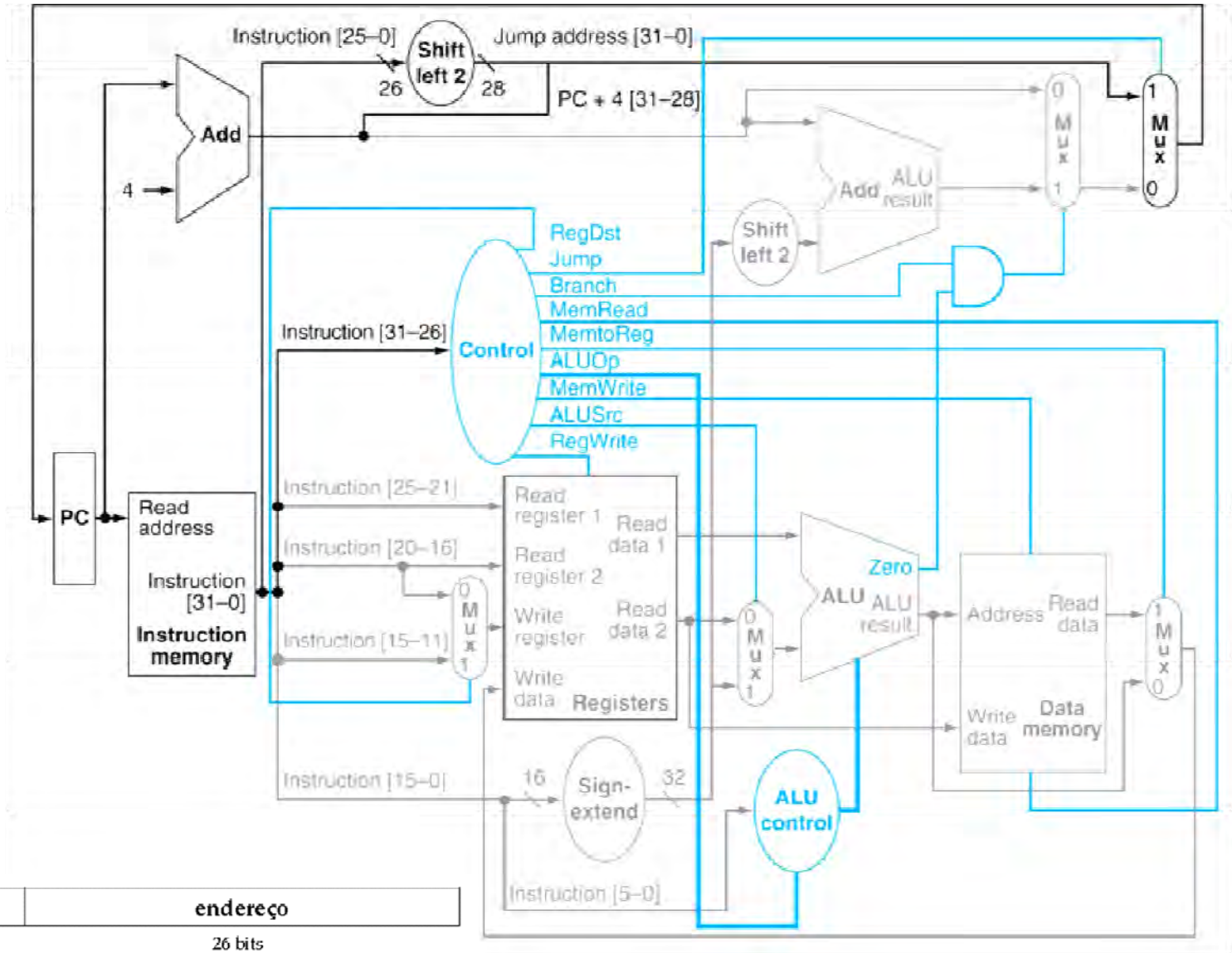
Input or output	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

Unidade de Controle

- Linhas de Controle
  - A partir das informação da tabela verdade, é possível implementar diretamente a unidade central de controle usando portas lógicas:



Unidade de Controle



Execução de uma Instrução Tipo R

- Seja a instrução tipo R add $t1, $t2, $t3, podemos imaginar que esta instrução é executada em 4 etapas:

  1. Busca da instrução (na memória de instruções) e incremento do PC

  2. Leitura de dois registradores (no caso, $t2 e $t3, ou Rs e Rt) e geração dos sinais de controle para o resto do bloco operativo (decodificação da instrução)

  3. Operação na ULA

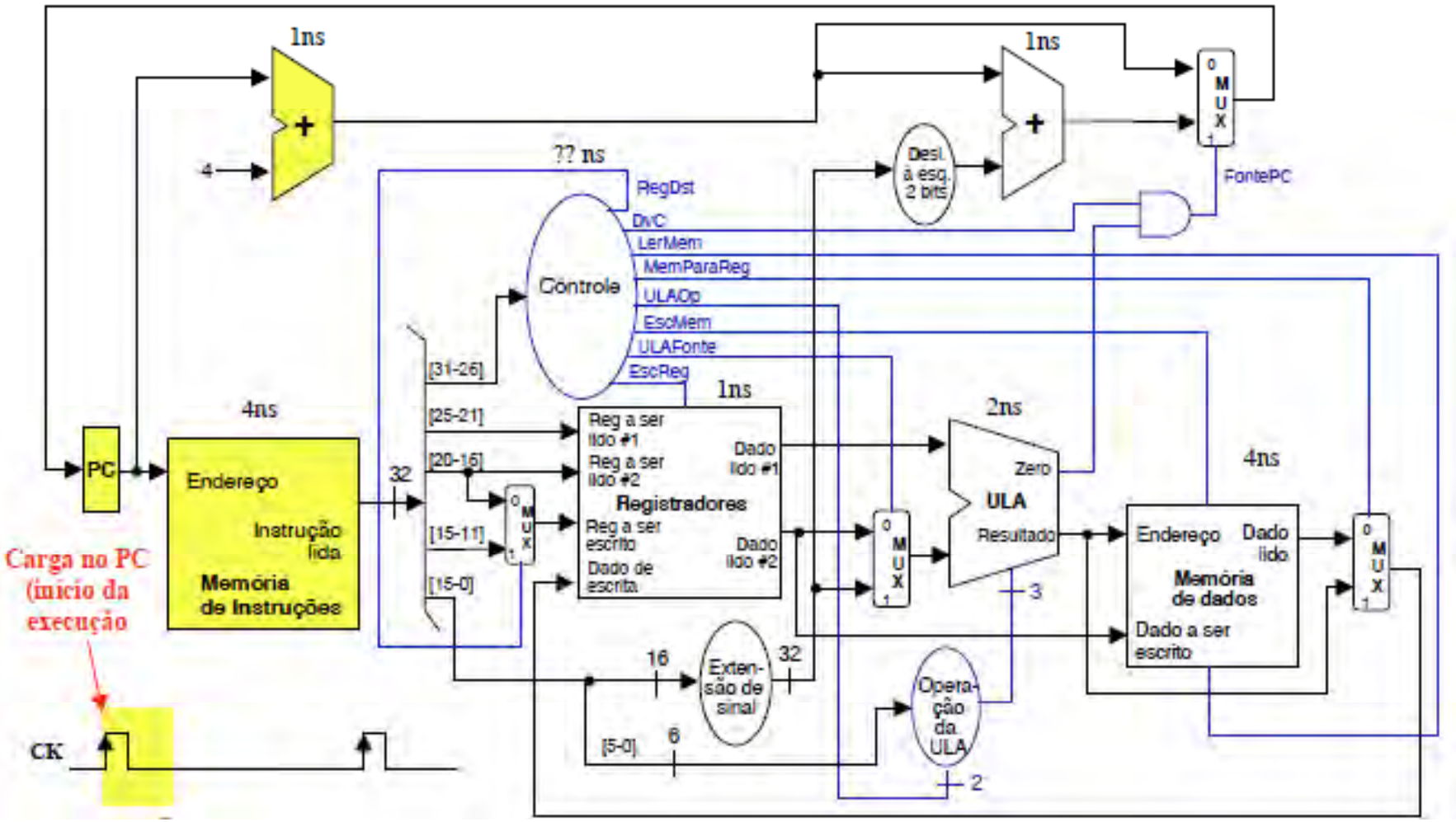
  4. Escrita (do resultado da operação realizada na ULA) no registrador destino ($t1 ou Rd)

Como estes passos ocorrem dentro do mesmo ciclo de relógio (regime monociclo), a ordem real irá depender do atraso de cada componente.

Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Execução de uma Instrução Tipo R

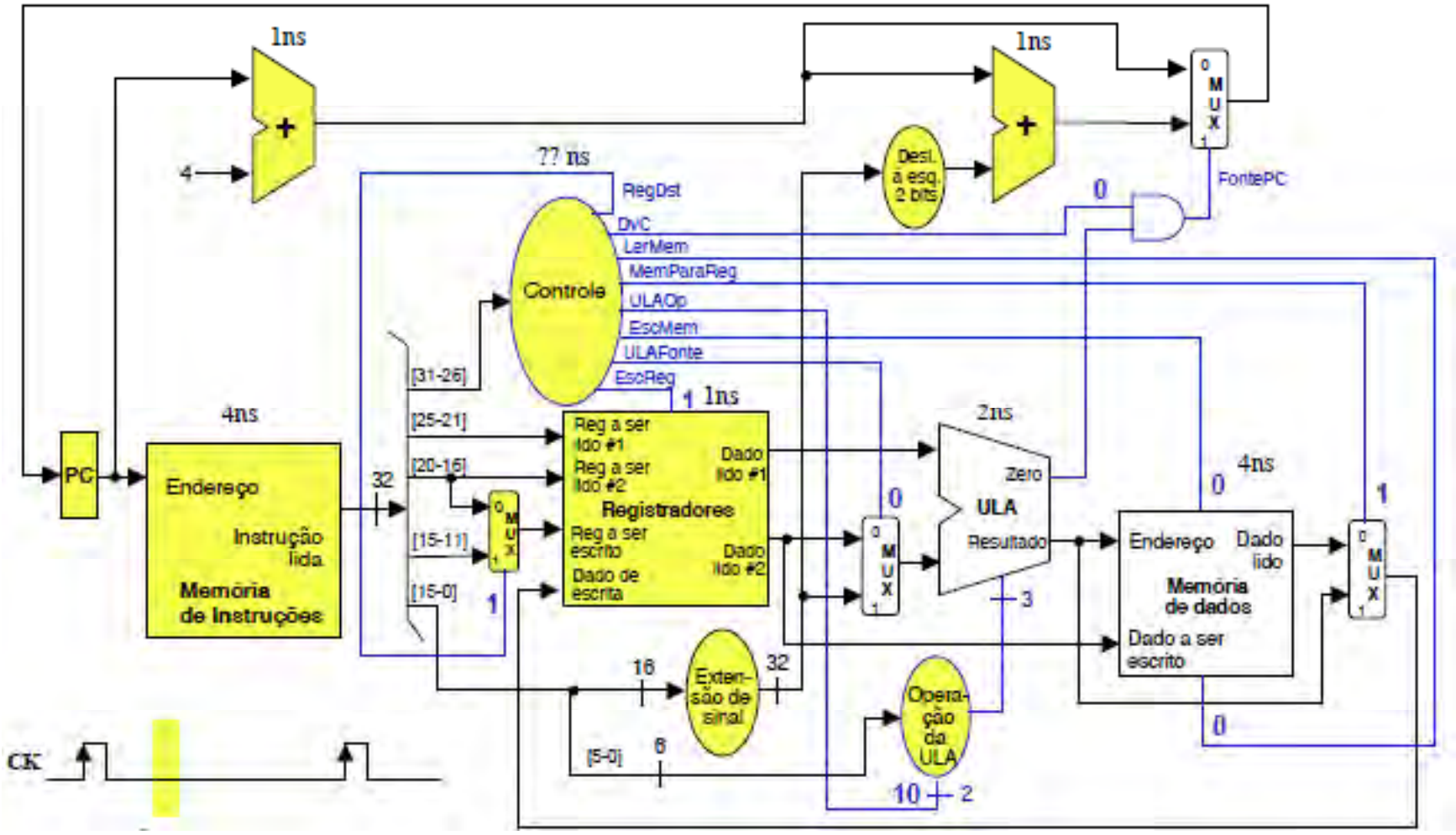
- Busca da instrução e cálculo de PC+4



Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Execução de uma Instrução Tipo R

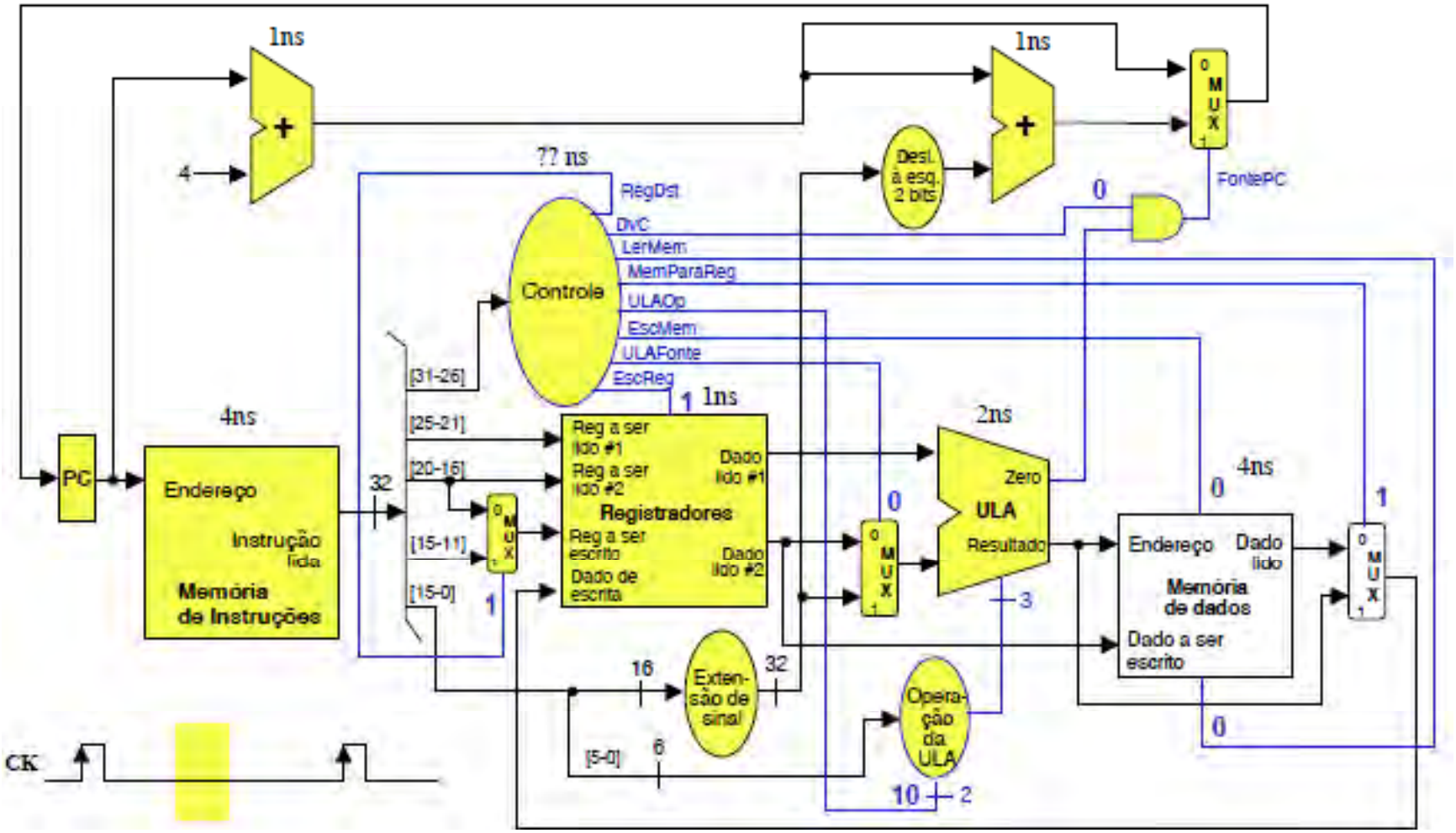
- Leitura de Rs e Rt e geração sinais de controle



Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Execução de uma Instrução Tipo R

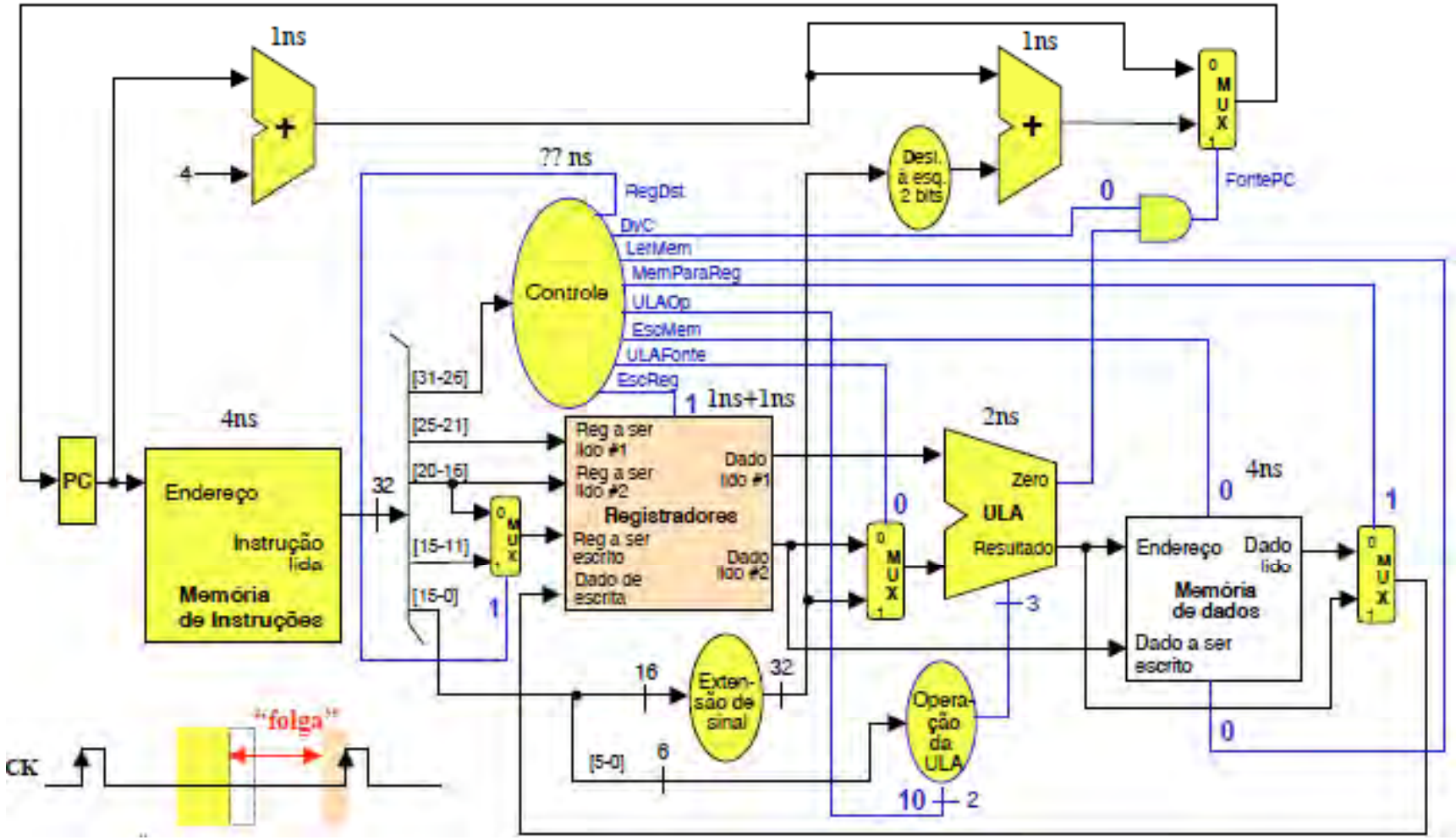
- Operação na ULA (depende de "funct")



Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Execução de uma Instrução Tipo R

- Escrita no registrador-destino



Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Execução de uma Instrução lw

- Seja a instrução tipo I lw $t1, deslocamento($t2), podemos imaginar que esta instrução é executada em 5 etapas:

1. Busca da instrução (na memória de instruções) e incremento do PC

2. Leitura de dois registradores (no caso, $t1 e $t2, ou Rs e Rt) e geração dos sinais de controle para o resto do bloco operativo (decodificação da instrução). Apenas o registrador $t2 (Rs) interessa, pois é o registrador-base. Rt será desprezado…

3. Cálculo do endereço usando a ULA (adição)

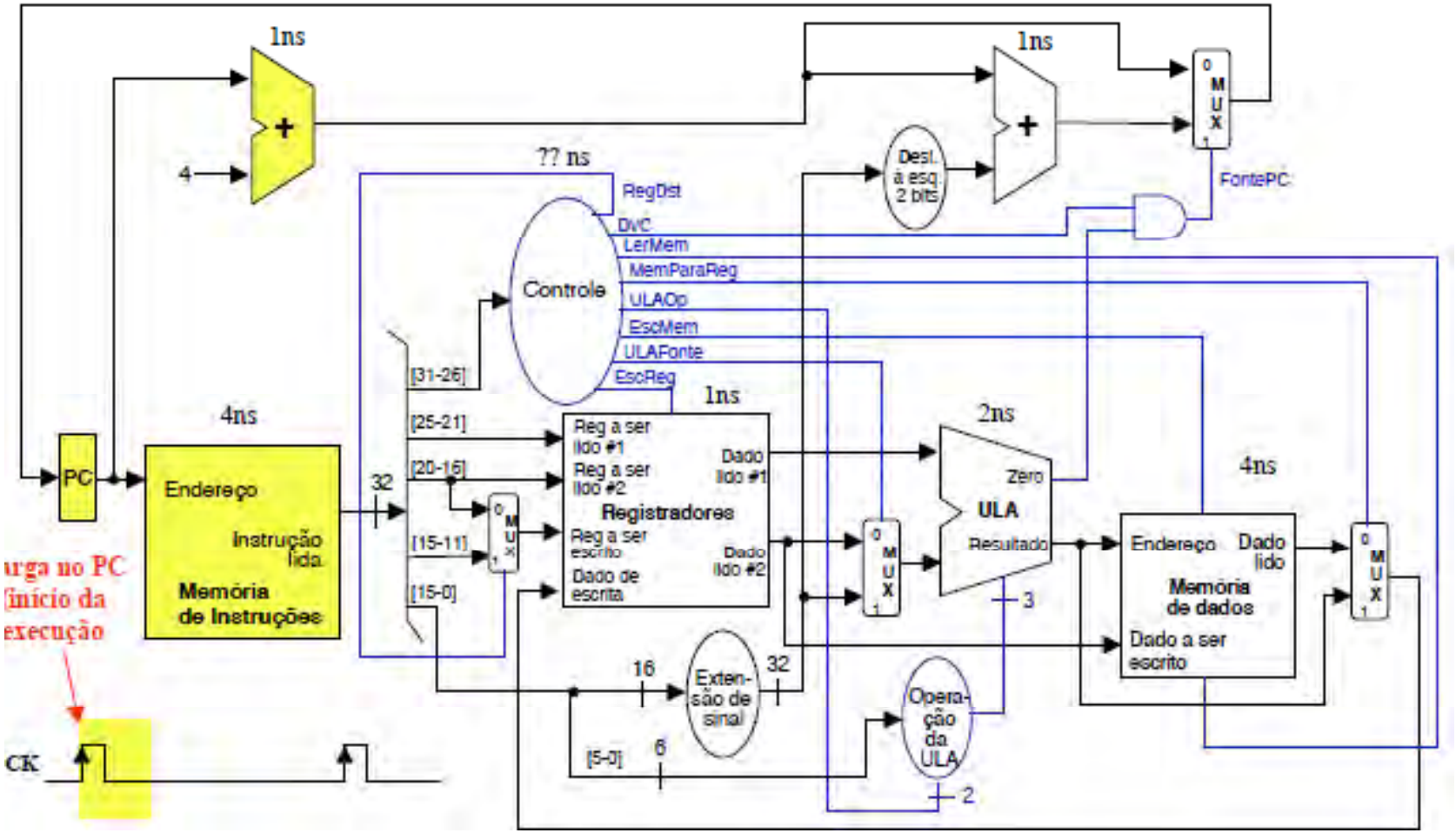
4. Acesso à memória de dados para uma leitura (endereço = resultado da ULA)

5. Escrita (do valor lido da memória de dados) no registrador destino ($t1, que neste caso corresponde ao campo Rt)

Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Execução de uma Instrução lw

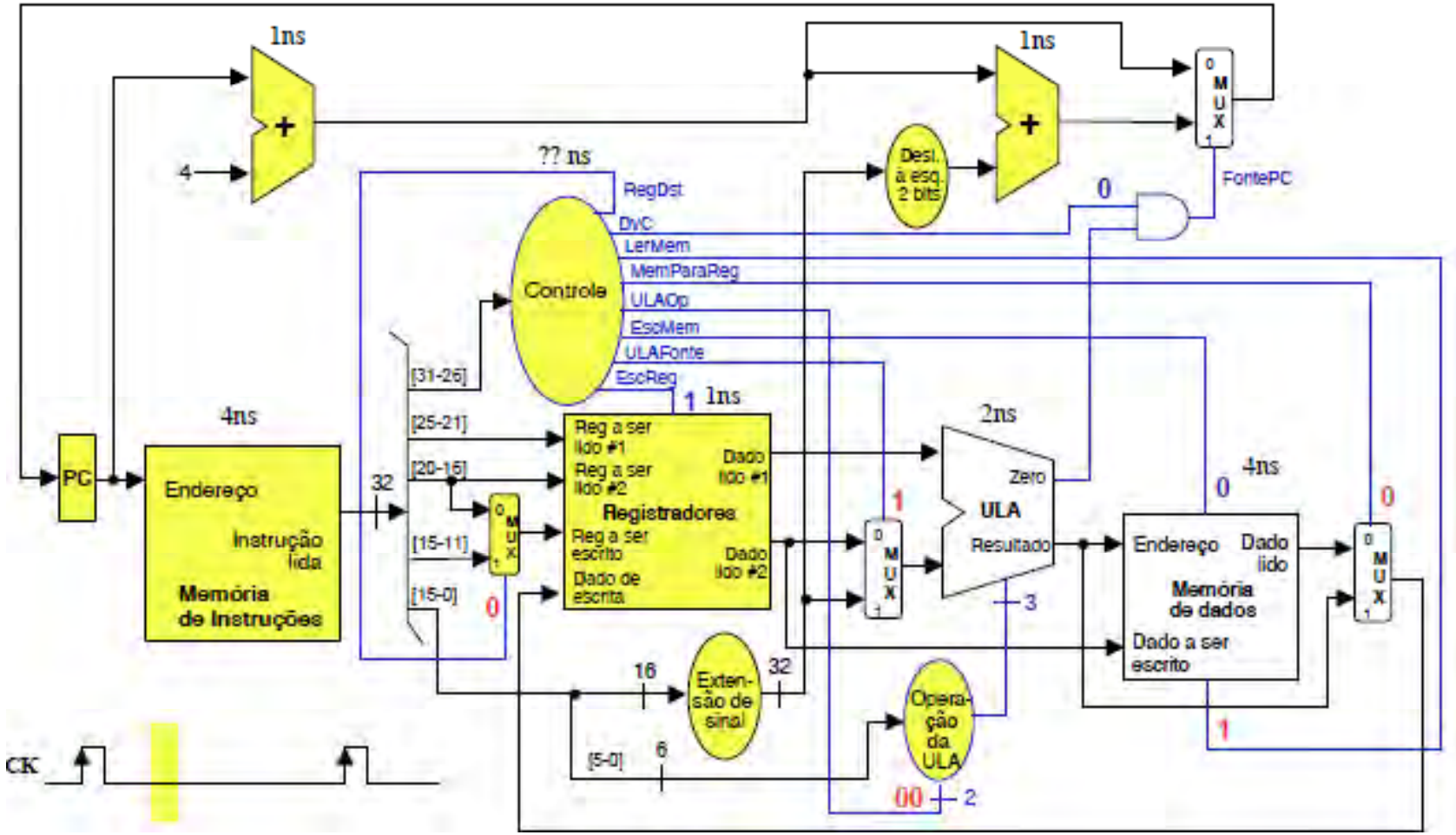
- Busca da instrução e cálculo de PC+4



Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Execução de uma Instrução lw

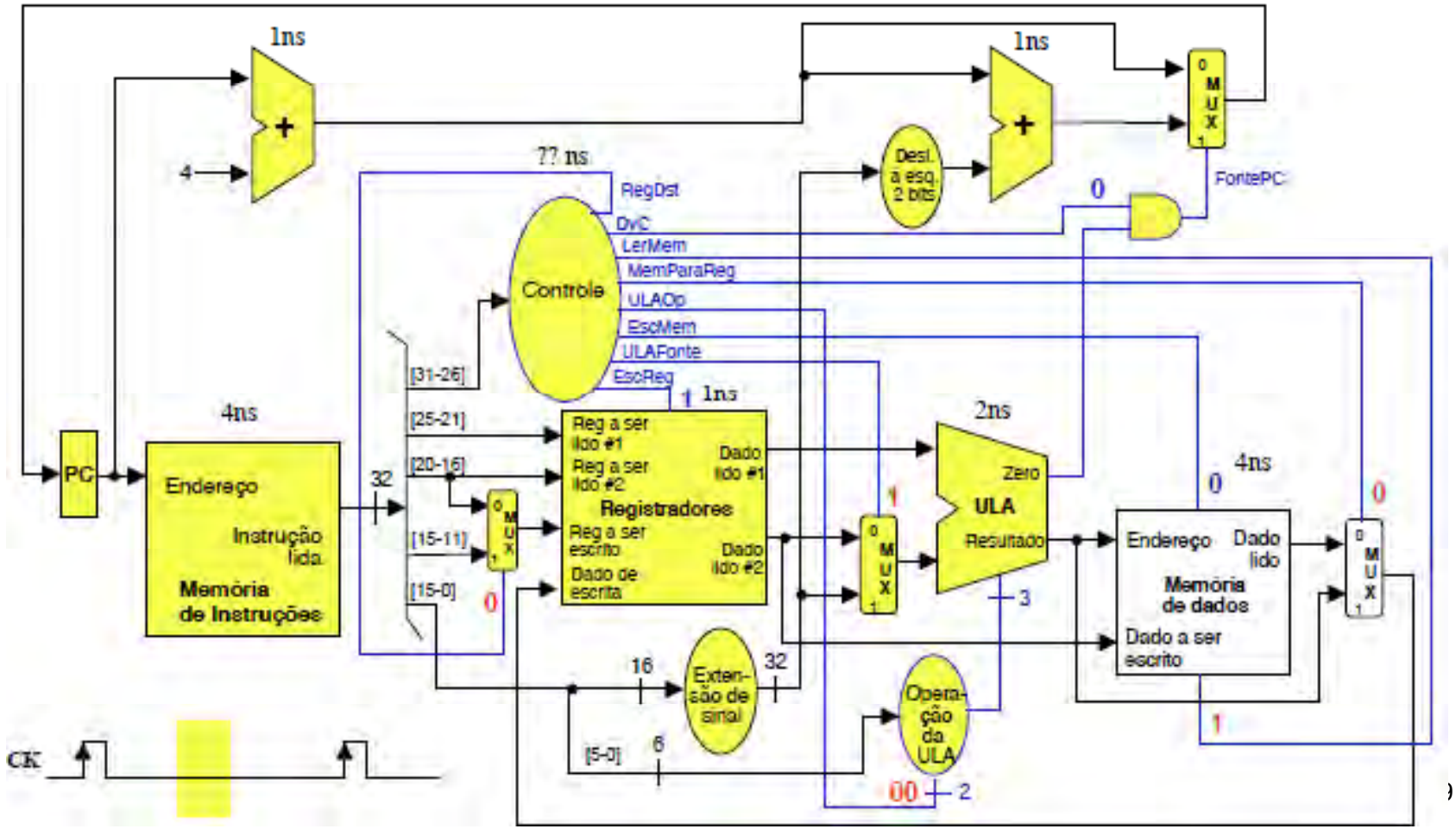
- Leitura de Rs (e Rt) e geração sinais de controle



Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Execução de uma Instrução lw

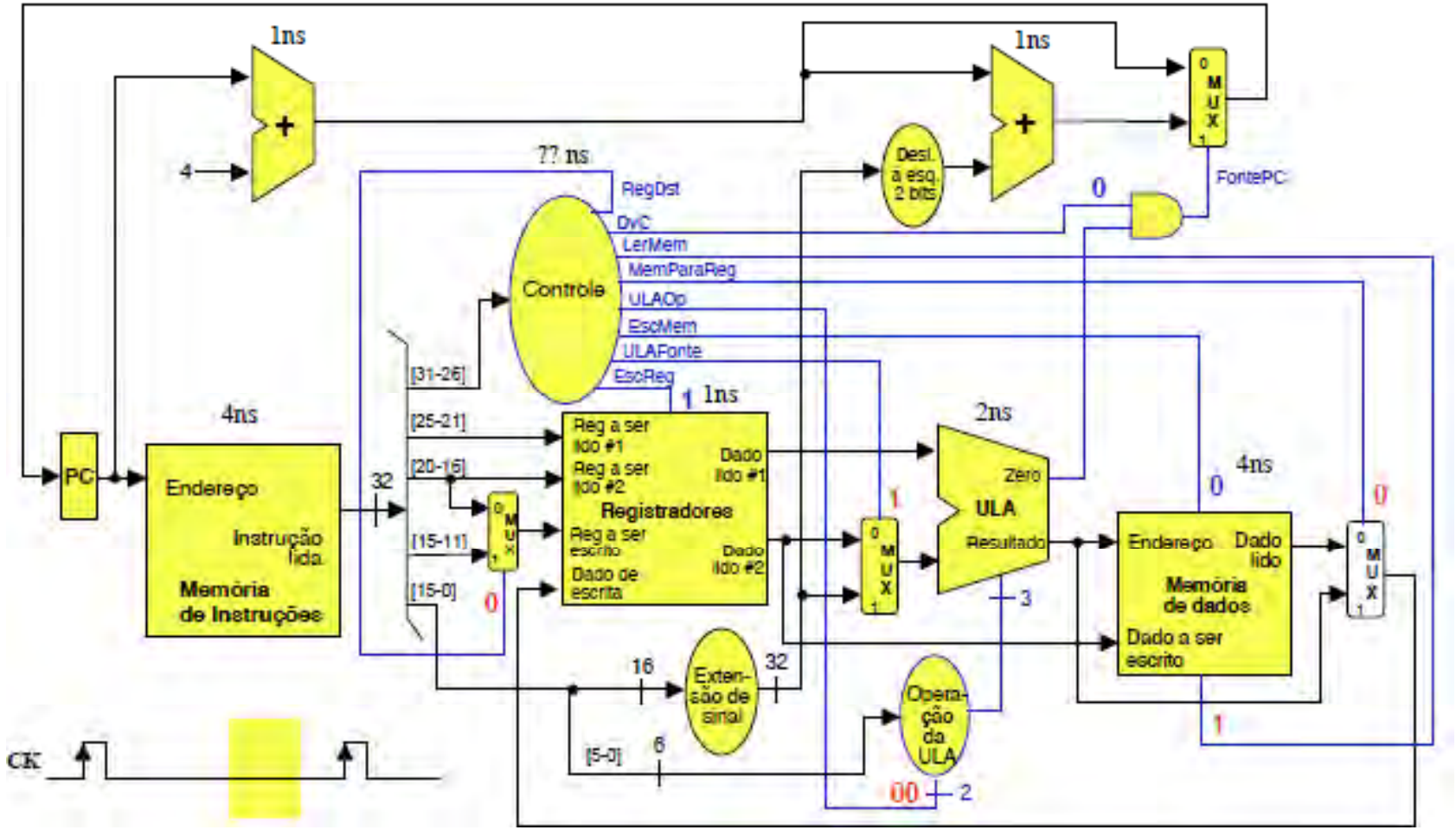
- Cálculo do endereço usando a ULA (adição)



Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Execução de uma Instrução lw

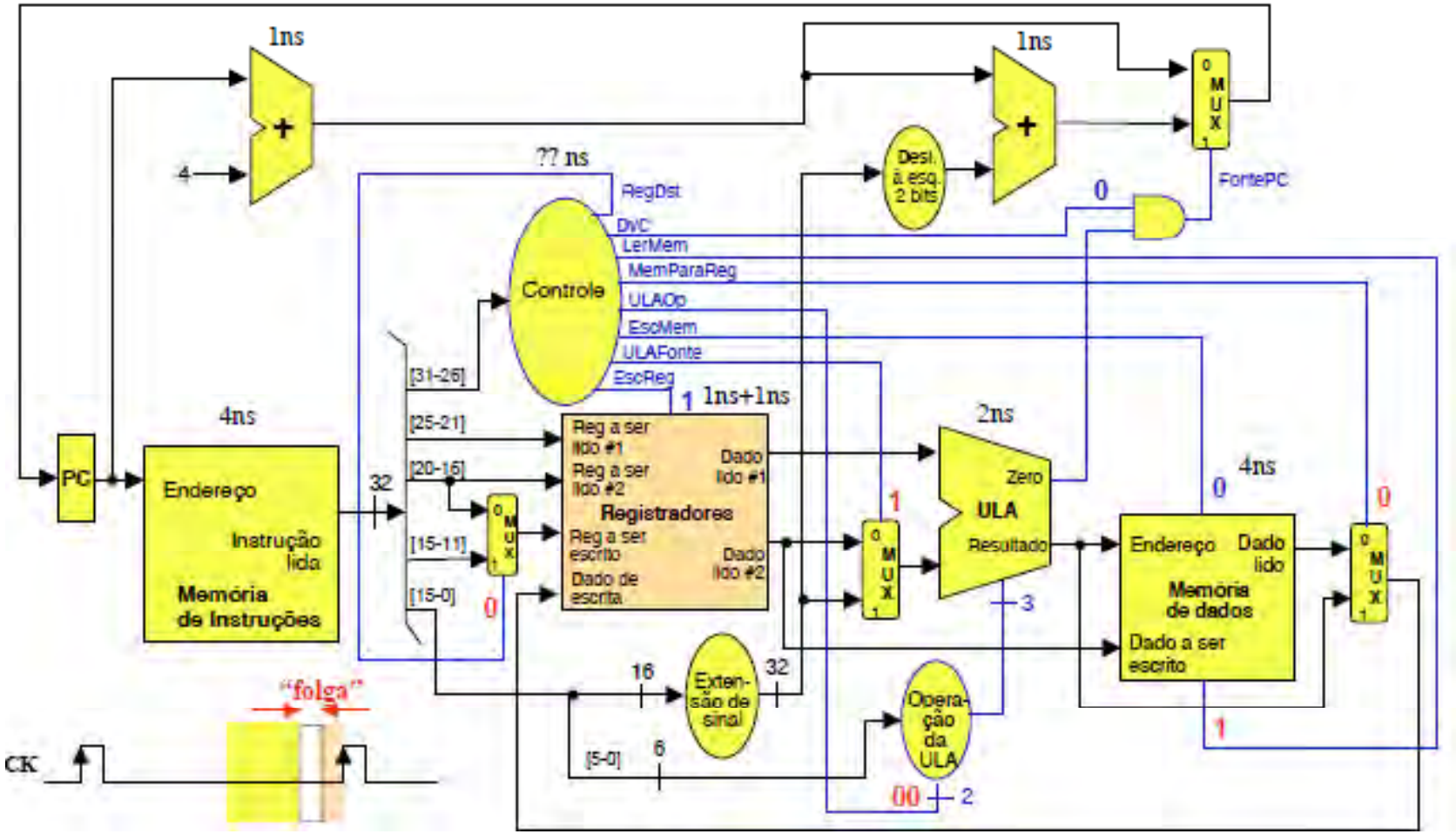
- Acesso à memória de dados para uma leitura



Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Execução de uma Instrução lw

- Escrita no registrador-destino



Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Desempenho de Máquinas Monociclo

- Unidades funcionais utilizadas por cada instrução

instrução	Etapa 1	Etapa 2	Etapa3	Etapa 4	Etapa 5
Tipo R	Busca da instrução	Lê registrador(es)	ULA	Escreve registrador	
lw	Busca da instrução	Lê registrador(es)	ULA	Lê memória	Escreve registrador
sw	Busca da instrução	Lê registrador(es)	ULA	Escreve na memória	
beq	Busca da instrução	Lê registrador(es)	ULA		
jump	Busca da instrução				

Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Desempenho de Máquinas Monociclo

- Tempo de execução de cada instrução (com valores hipotéticos de atraso para cada etapa)

instrução	Acesso à memória de instruções	Leitura de registradores	Operação na ULA	Acesso à memória de dados	Escrita no registrador	Total
Tipo R	4 ns	1 ns	2 ns	---	1 ns	8 ns
lw	4 ns	1 ns	2 ns	4 ns	1 ns	12 ns
sw	4 ns	1 ns	2 ns	4 ns	---	11 ns
beq	4 ns	1 ns	2 ns	---	---	7 ns
jump	4 ns	---	---	---	---	4 ns

Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Conclusões

- Vimos como é possível construir um datapath relativamente simples para executar instruções da arquitetura MIPS
- Os mesmos princípios se aplicam para se implementar outras instruções
- O datapath apresentado pode ser otimizado (para execução mais rápida) em vários aspectos
- Uma de suas limitações é o fato de ser mono-ciclo, ou seja, executar qualquer instrução em um ciclo de clock, porém esse ciclo é bastante demorado.
- Solução: implementação multi-ciclo desse datapath.

Conclusões

- A duração de um ciclo de clock deve ser longa o bastante para comportar o caminho mais longo no processador.
- A penalidade por utilizar um projeto uniciclo é significativa, mas pode ser considerada aceitável para um conjunto bastante reduzido de instruções – foi explorada inicialmente em computadores com conjuntos bastante simples de instruções.
- Observação:
  - Como o ciclo de relógio é determinado pelo atraso de pior caso entre todas as instruções, é absolutamente inútil o uso de técnicas que reduzem o atraso do caso comum, mas que não trazem qualquer otimização do pior caso.