Aula 5

Projeto do MIPS

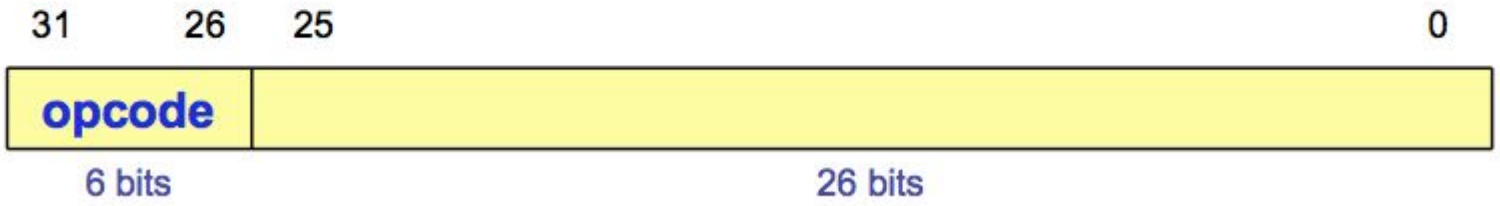
Arquitetura MIPS

Instruções Principais

tipo	Instrução	exemplo	significado
Aritmética	Add immediate	addi $s1, $s2, 100	$s1 = $s2 + 100
Aritmética	Add	add $s1, $s2, $s3	$s1 = $s2 + $s3
Aritmética	Subtract	sub $s1, $s2, $s3	$s1 = $s2 - $s3
Aritmética	OR	or $s1, $s2, $s3	$s1 = $s2 OR $s3
Aritmética	AND	and $s1, $s2, $s3	$s1 = $s2 AND $s3
Transf. de dados	Load word	lw $s1, 100($s2)	$s1 = Mem[$s2+100]
Transf. de dados	Store word	sw $s1, 100($s2)	Mem[$s2+100] = $s1
Transf. de dados	Load Upper Immediate	lui $s1, 100	$s1 = 100 * 2^{16}
Desvio cond.	Branch on equal	beq $s1, $s2, L	if($s1 == $s2) go to L
Desvio cond.	Branch on not equal	bne $s1, $s2, L	if($s1 != $s2) go to L
Desvio cond.	Set on less than	slt $s1, $s2, $s3	if($s2 < $s3) $s=1; else $s1=0
Desvio cond.	Set less than immediate	slti $s1, $s2, 100	if($s2 < 100) $s=1; else $s1=0
Desvio incond.	Jump	j 2500	Desvia para 10000
Desvio incond.	Jump register	jr s$1	Desvia para $t1
Desvio incond.	Jump and link	jal 2500	$ra = PC+4; desvia para 10000

Instruções

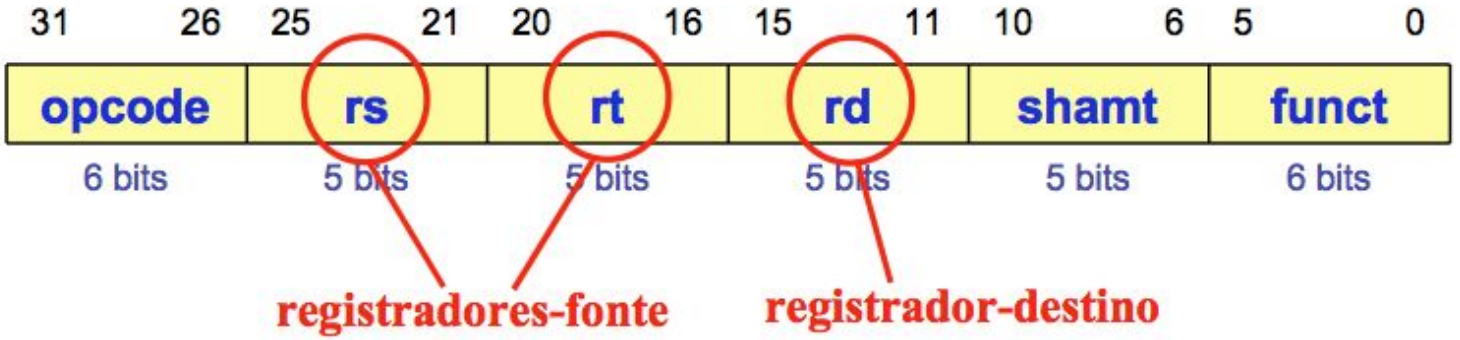
- todas as instruções têm 32 bits
- todas têm opcode de 6 bits
- o modo de endereçamento é codificado juntamente com o opcode



Arquitetura MIPS

Instruções formato R: add, sub, or, and

- opcode = 0
- "funct" define a operação a ser feita pela ALU
- "shamt" (shift amount) é usado em instruções de deslocamento

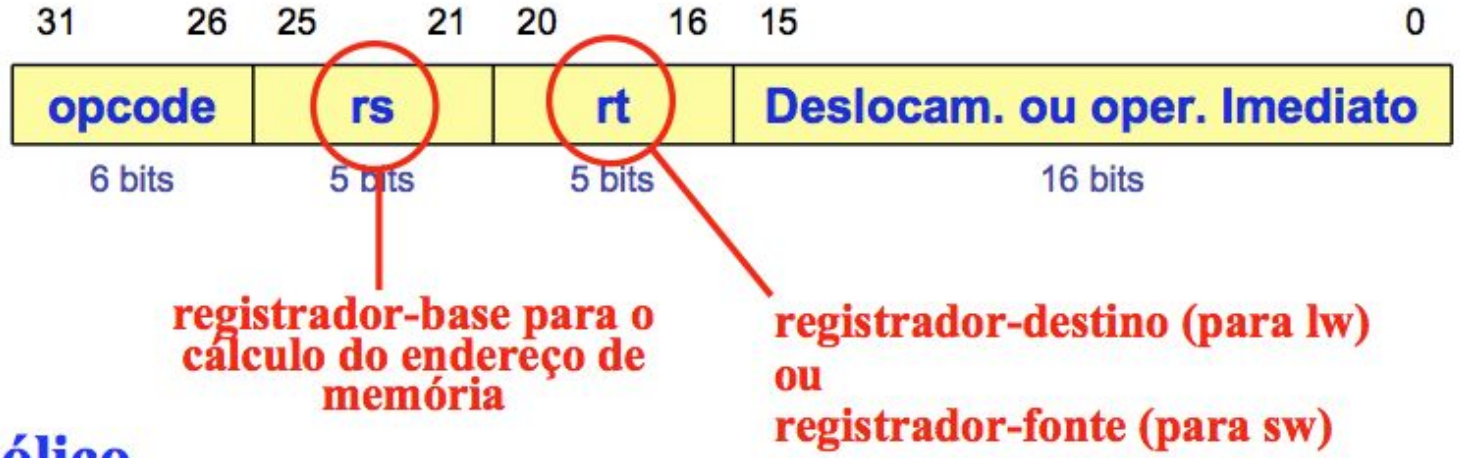


Simbólico (exemplo): `add $s1,$s2, $s3` (`$s1 ← $s2 + $s3`)

Arquitetura

Instruções formato I: load word (ld) e store word (sw)

- load word (lw): opcode = 35
- store word (sw): opcode = 43



Simbólico

```
lw $s1, deslocam($s2)   ($s1 ← Mem[$s2 + deslocam] )
sw $s1, deslocam($s2)   (Mem[$s2 + deslocam] ← $s1 )
```

Instrução formato I: Desvio Condicional

beq: branch on equal

- Opcode = 4
- Campo deslocamento usado para calcular o endereço-alvo
- Se o conteúdo do registrador cujo endereço está no campo rs for igual ao conteúdo do registrador cujo endereço está em rt, então salta para a posição endereço+PC+4

Simbólico

```
beq $s1, $s2, deslocm   ( if ($s1== $s2) then PC←PC+4+deslocam)
```

Instrução formato J: Desvio Incondicional

j: jump

- Opcode = 2
- Campo deslocamento usado para calcular o endereço-alvo
- Se o conteúdo do registrador cujo endereço está no campo rs for igual ao conteúdo do registrador cujo endereço está em rt, então salta para a posição endereço+PC+4

Simbólico

```
j endereço   ( PC←endereço )
```

Instruções Aritméticas e Lógicas

- operação entre 2 registradores, resultado num terceiro registrador
  - tipo R
- add, and, nor, ori, or, sub, xor
- existem versões com endereçamento imediato
  - addi, andi, ori, xori
  - tipo I
- comparação – compara dois registradores e coloca valor 1 (ou 0) em registrador destino
  - slt, slti
- $0 usado para sintetizar operações populares
  - carga de constante = soma imediata onde $0 é um dos operandos
  - mover de registrador para registrador = soma com $0

Princípios do Projeto Eficiente

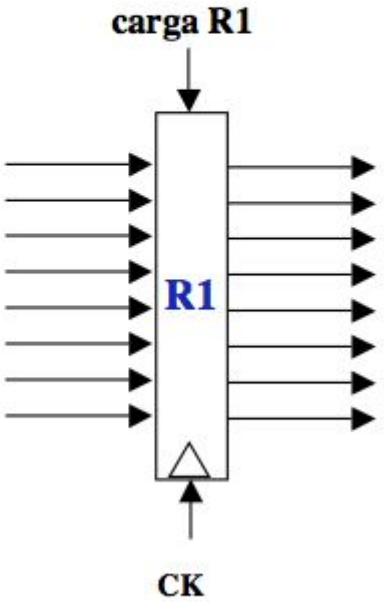
“Faça o caso comum executar mais rápido”

“A simplicidade favorece a regularidade”

Regime de Clock (Temporização)

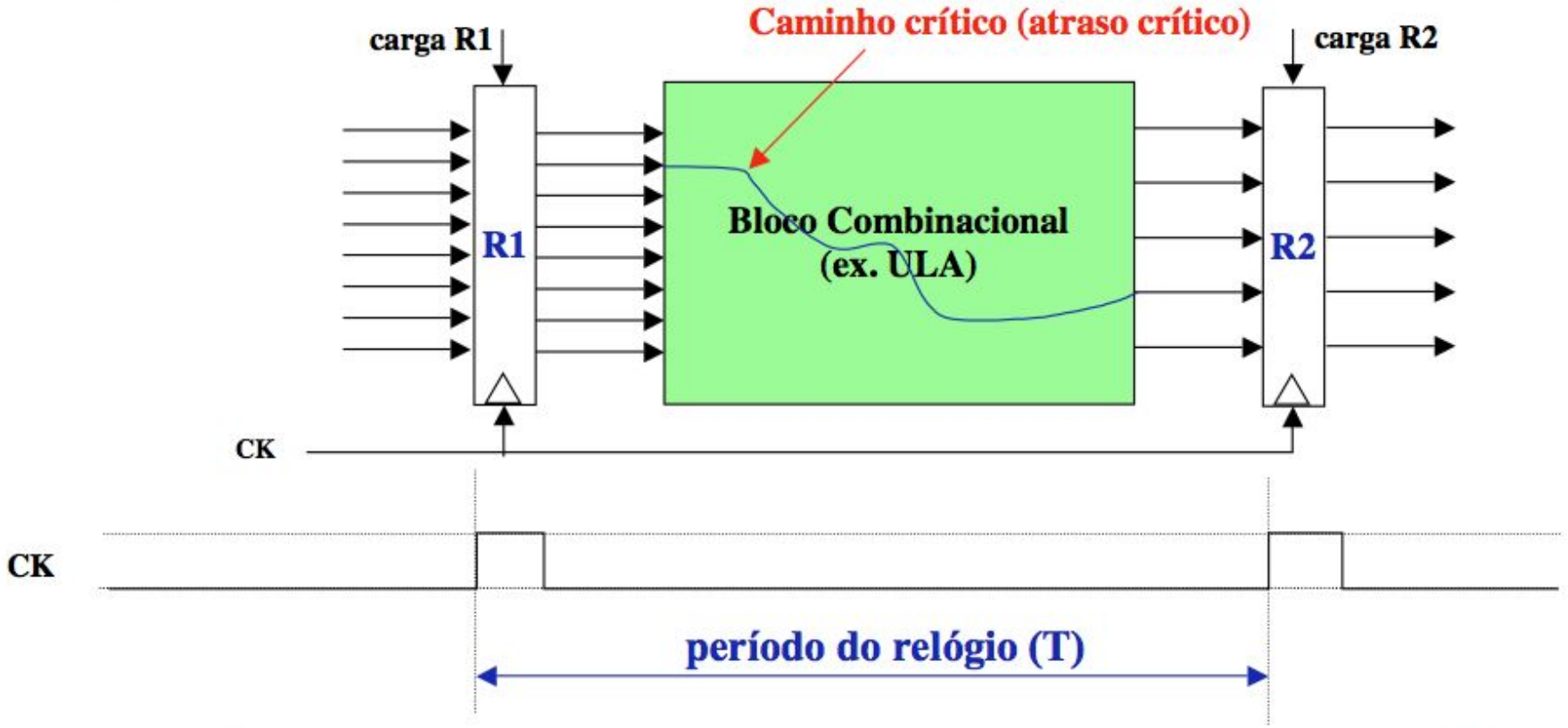
Iremos supor que:

- **Cada registrador possui um sinal de carga particular que é ativado com lógica direta**
- **Os registradores são "disparados" pela borda ascendente do relógio**
- **No desenho ao lado, quando CK=↑, se cargaR1=1, então R1←entradas**



Regime de Clock (Temporização)

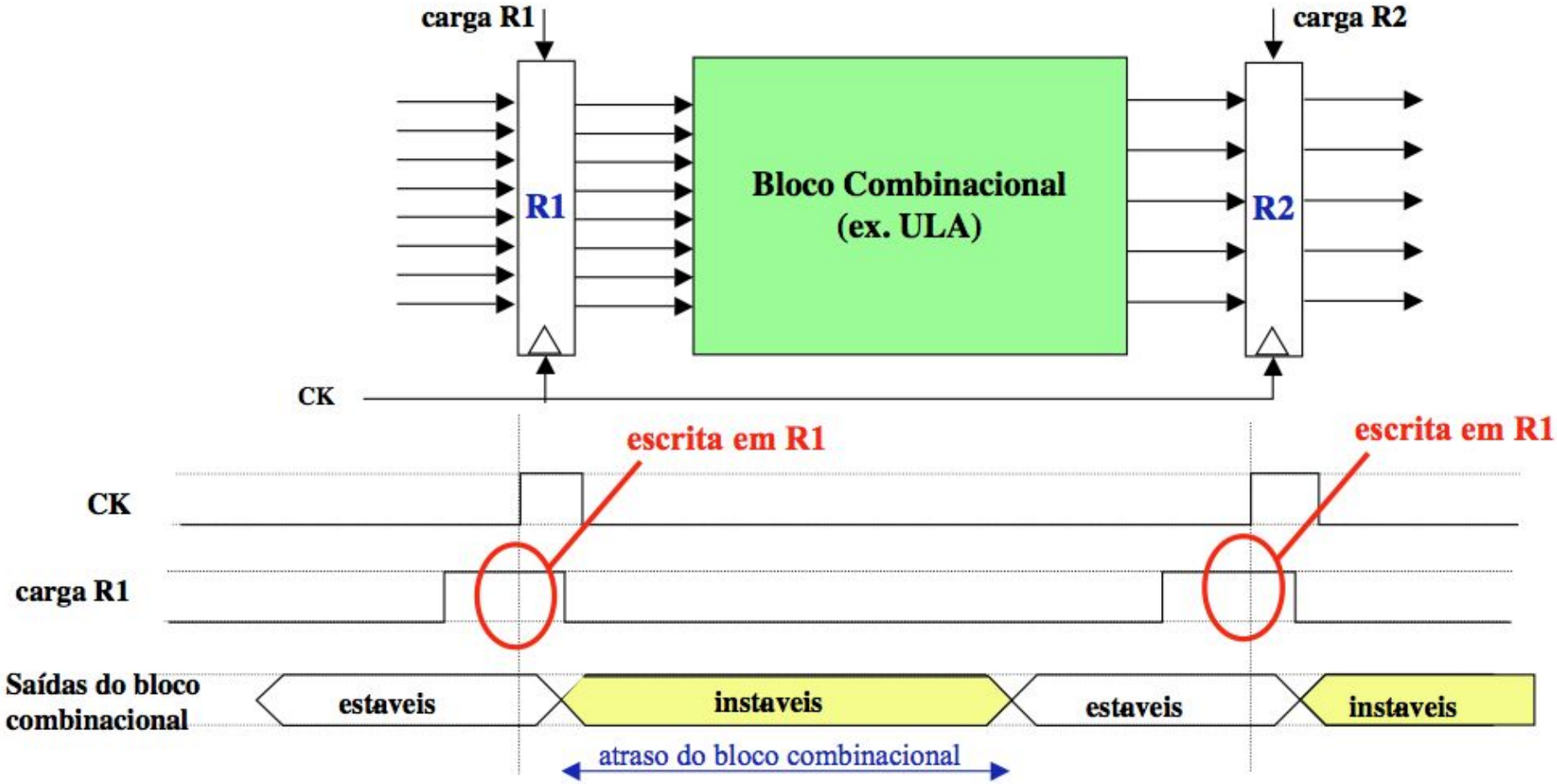
Condições para o Correto Funcionamento do Bloco Comb.



O atraso crítico do bloco combinacional deve ser menor que o período do relógio

Regime de Clock (Temporização)

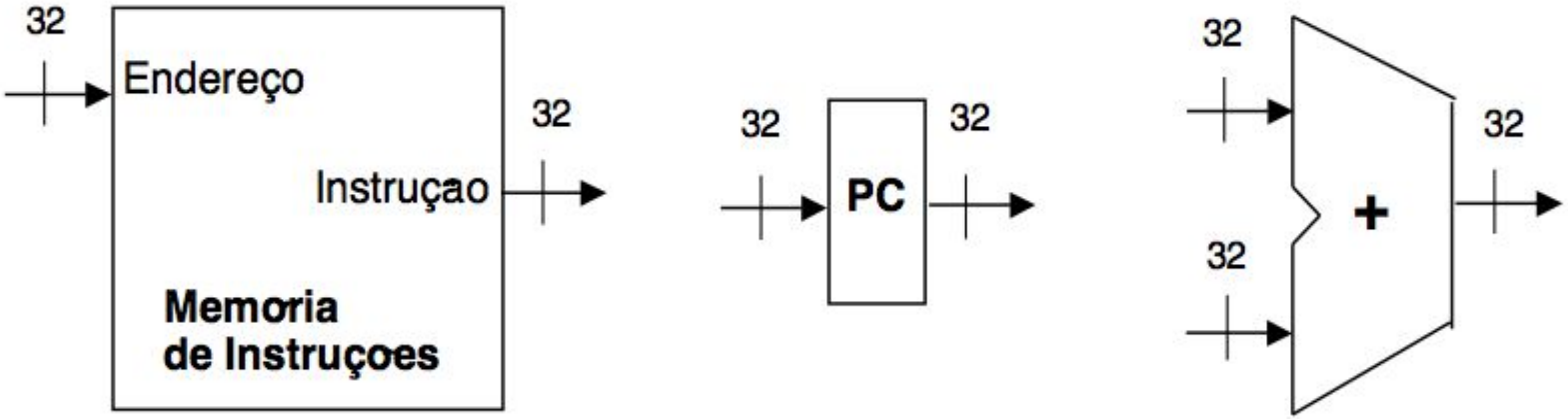
Temporização para o Correto Funcionamento do Bloco Comb.



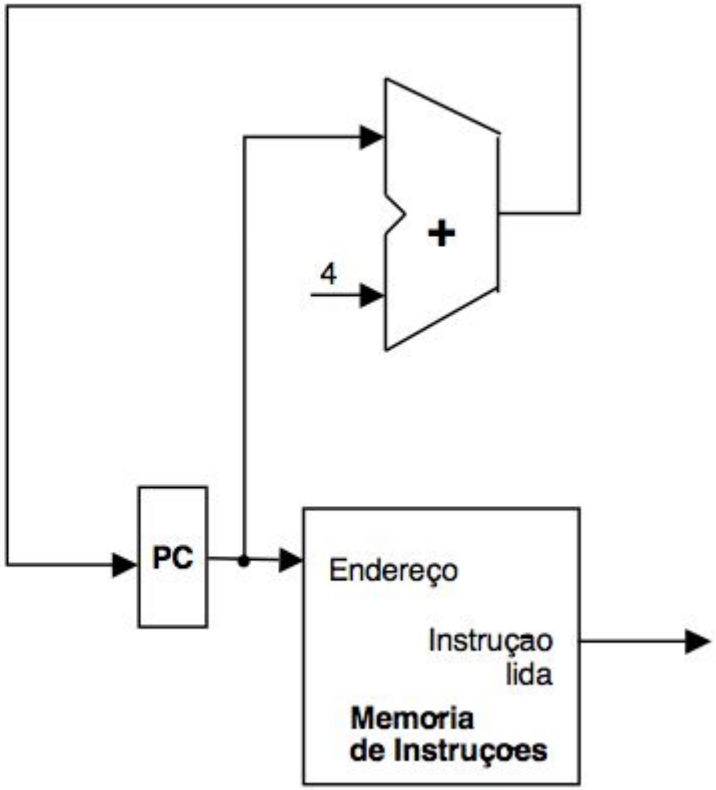
- **Qualquer instrução é executada em um único ciclo de relógio**
- **O período do relógio será longo o suficiente para acomodar qualquer instrução**
  - Na verdade, o período do relógio será função da instrução mais demorada

Elementos Necessários Para a Busca da Instrução:

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

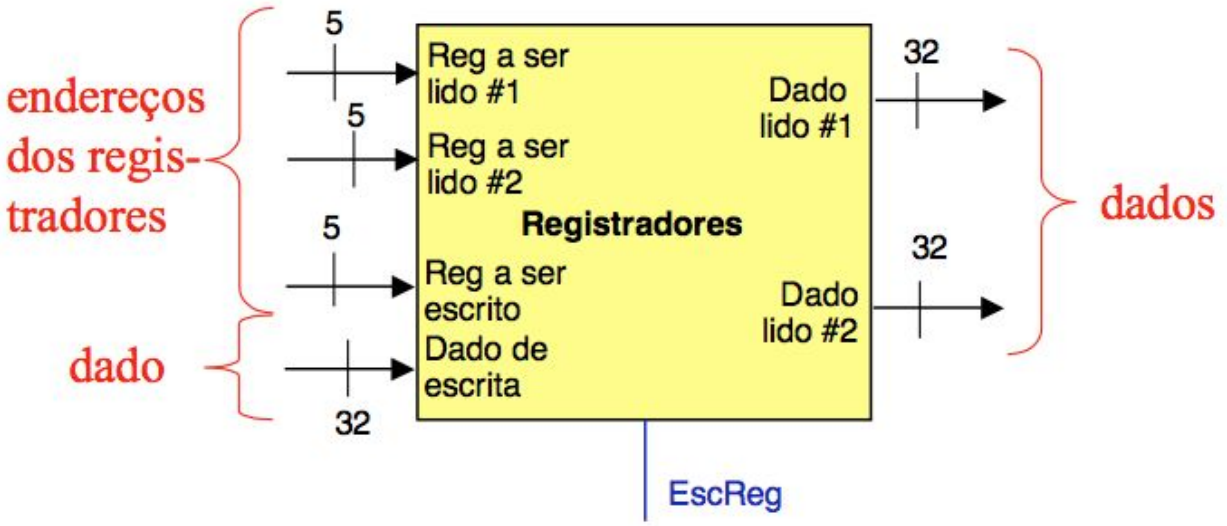


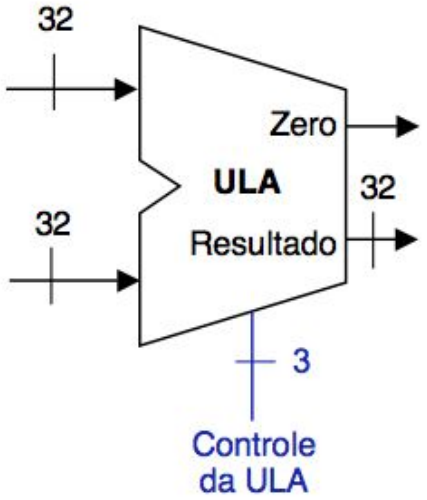
- O contador de programa contém o endereço da instrução em execução
- O endereço da próxima instrução é obtido pela soma de 4 posições ao contador de programa
- A instrução lida é usada por outras porções do bloco operativo



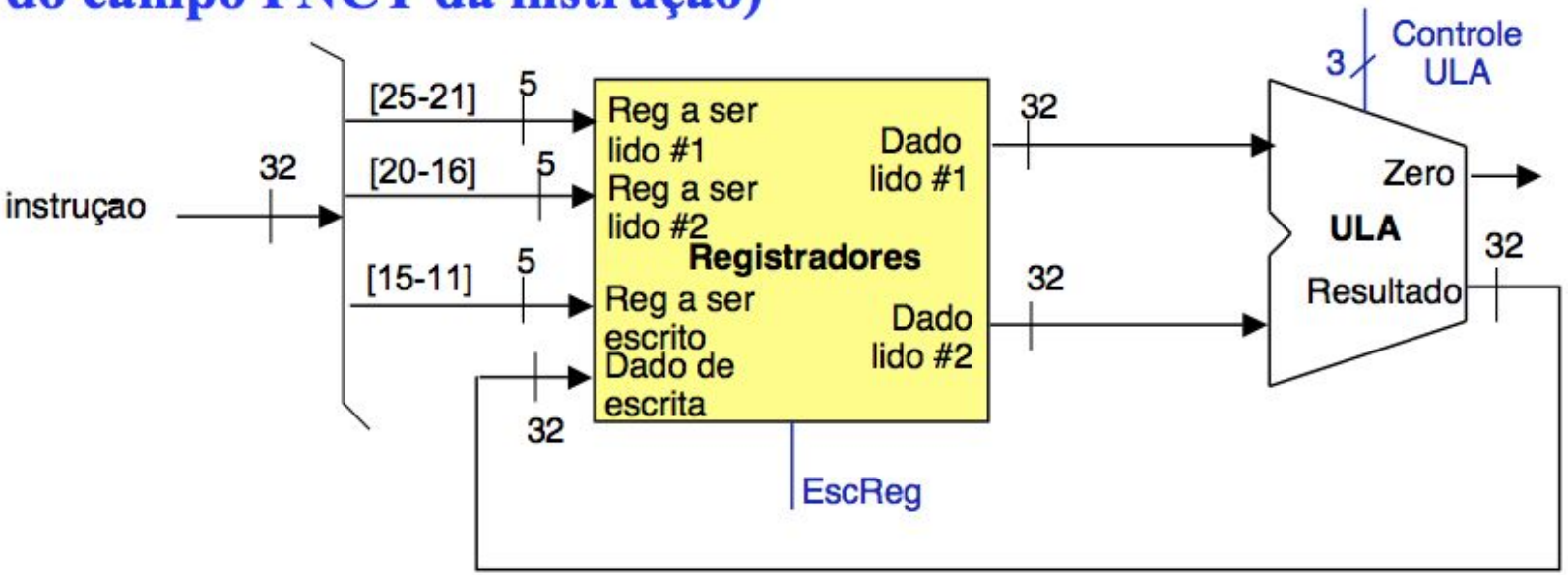
Elementos Necessários para Execução de Instruções tipo R:

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



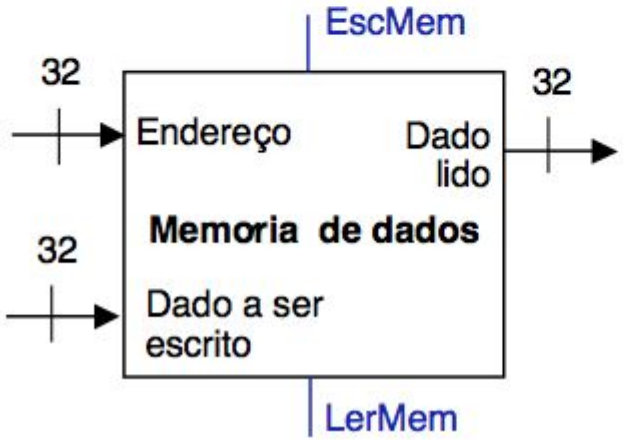


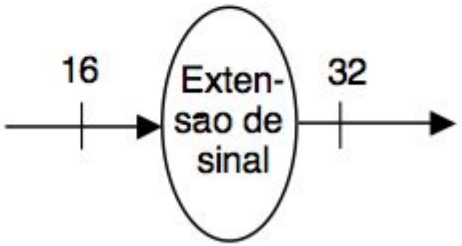
- **A instrução (fornecida pelo hardware de busca de instruções) contém o endereço de três registradores**
- **Dois destes registradores são lidos e passados para a ULA realizar a operação**
- **O resultado é armazenado em um terceiro registrador**
- **O controle da ULA determina a operação que será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



Elementos Necessários para Executar lw e sw

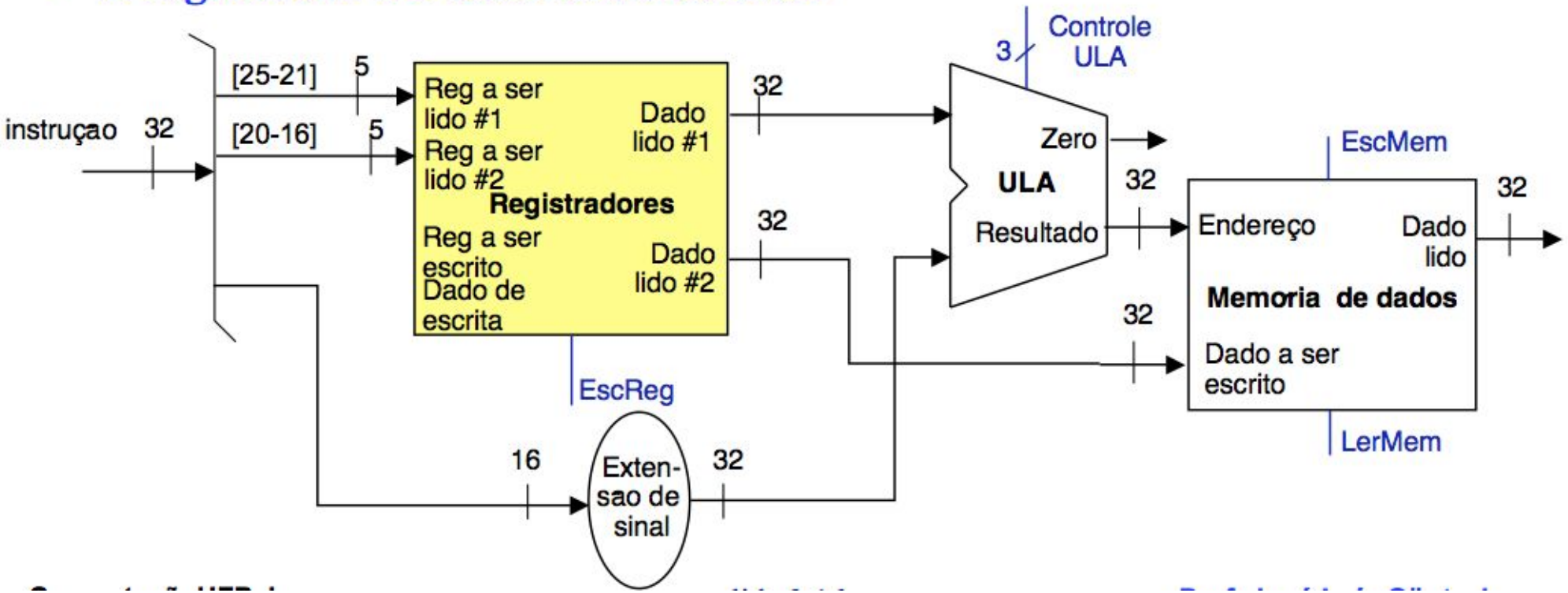
- Uma memória de dados
- Um módulo de extensão de sinal
- Um banco de registradores (já mostrado)
- Uma ULA (já mostrada)





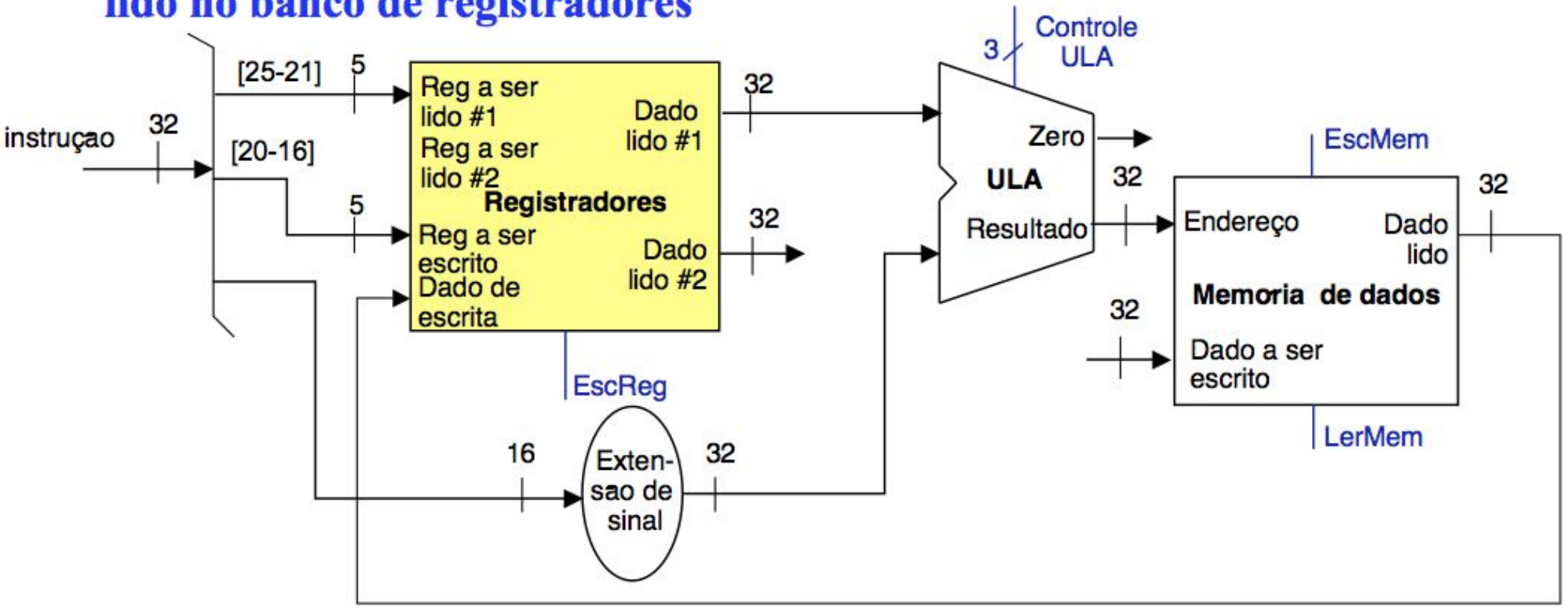
Combinando os Elementos para uma Escrita na Memória (sw)

- O endereço de escrita é obtido pela soma de um registrador de base (registrador 1) com um deslocamento de 16 bits estendido para 32 bits
- O registrador 2 é escrito na memória



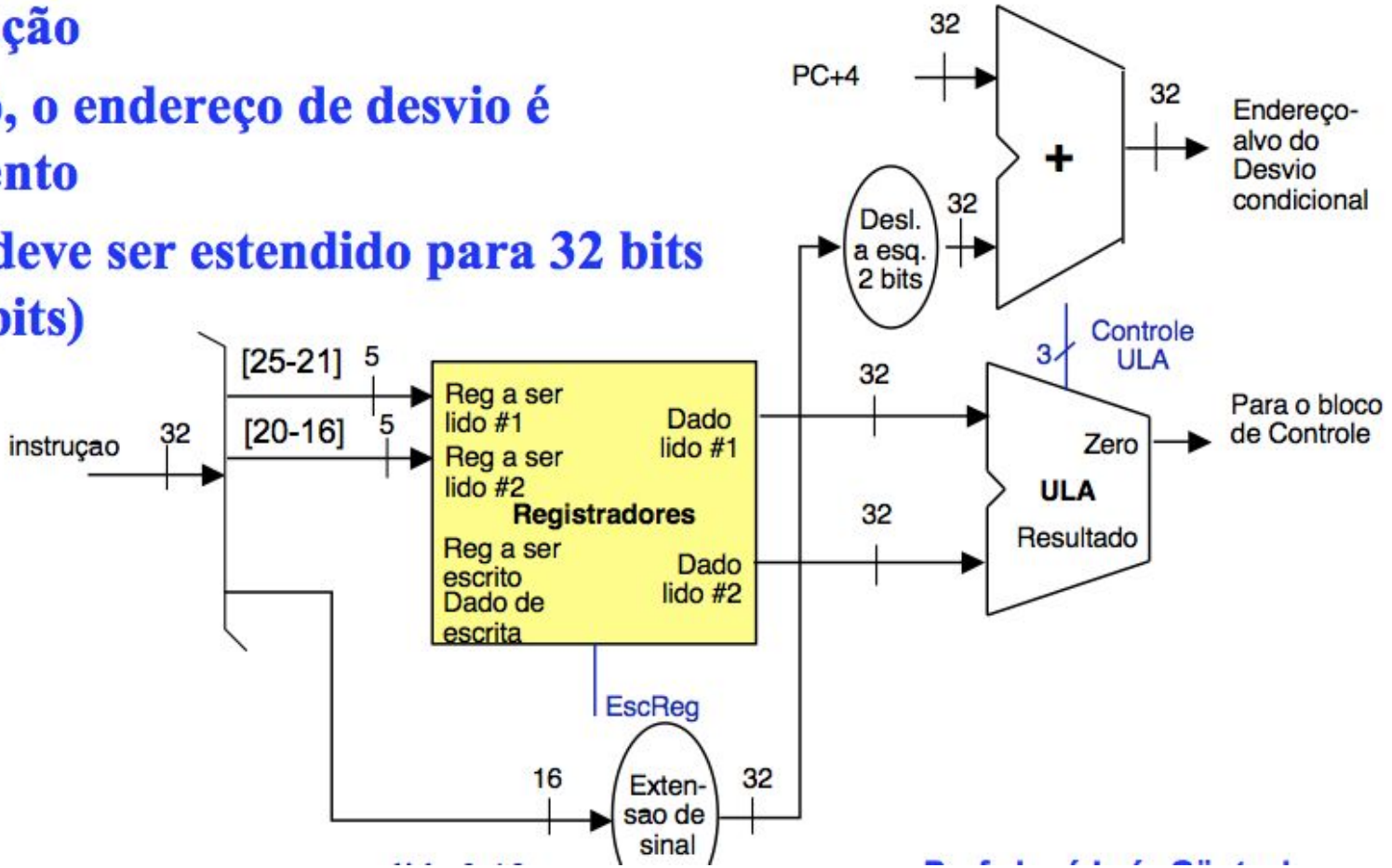
Combinando os Elementos para uma Leitura da Memória (lw)

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



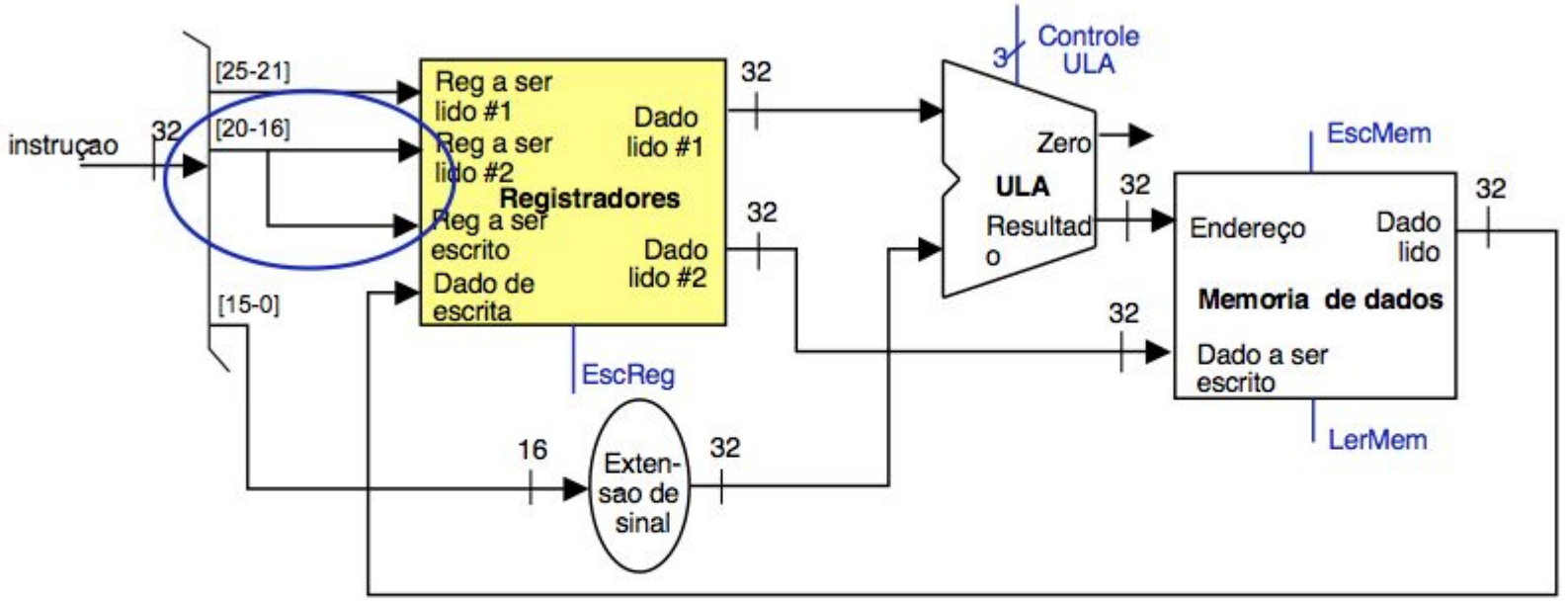
Elementos Necessários Implementar um Branch on Equal

- Comparar dois registradores usando a ULA para fazer uma subtração
- Se ocorrer desvio, o endereço de desvio é PC+4+deslocamento
- O deslocamento deve ser estendido para 32 bits (pois está em 16 bits)



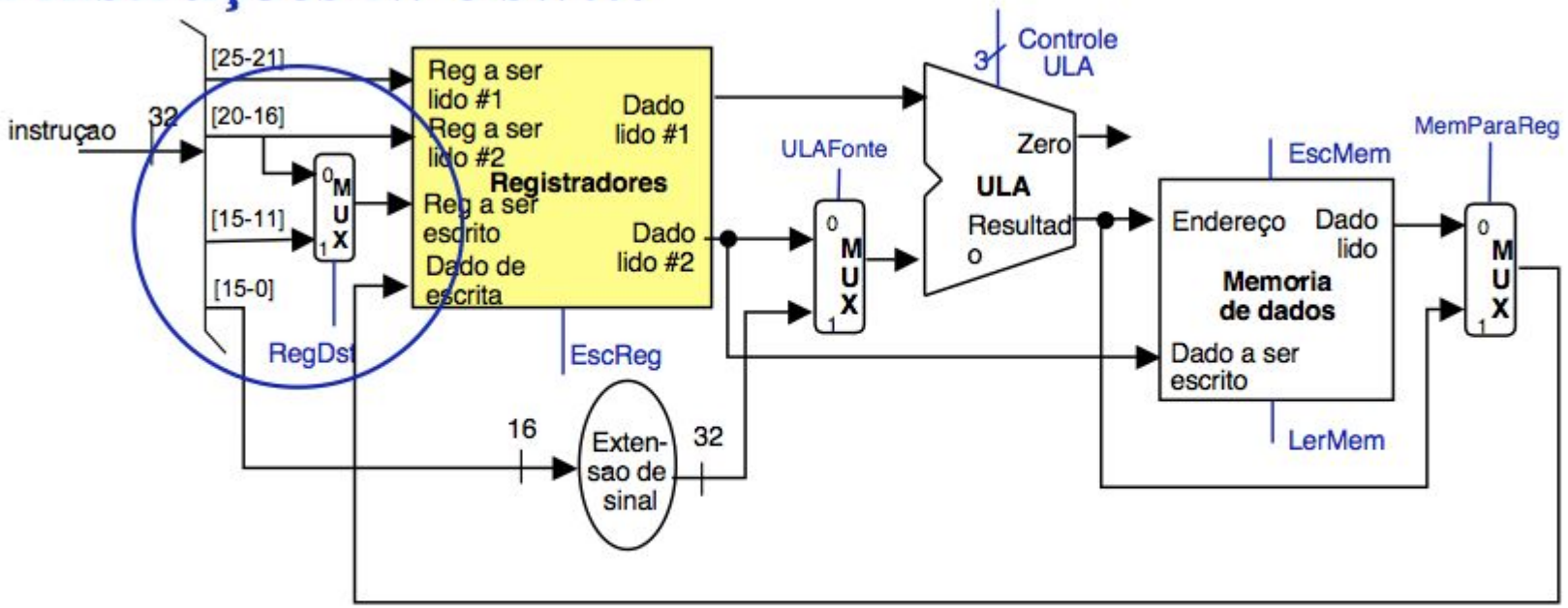
- Nesta versão de bloco operativo qualquer instrução executa em um ciclo de relógio
- Isto significa que o período do relógio deverá ser suficientemente longo para acomodar a instrução mais demorada
- Durante a execução de uma instrução qualquer, cada unidade funcional só pode ser usada uma única vez
- Por isso necessitamos de uma memória de instruções e outra de dados
- Ao combinarmos as porções de bloco operativo vistas anteriormente, veremos que muitas unidades funcionais podem ser compartilhadas

Recursos para executar instruções lw ou sw



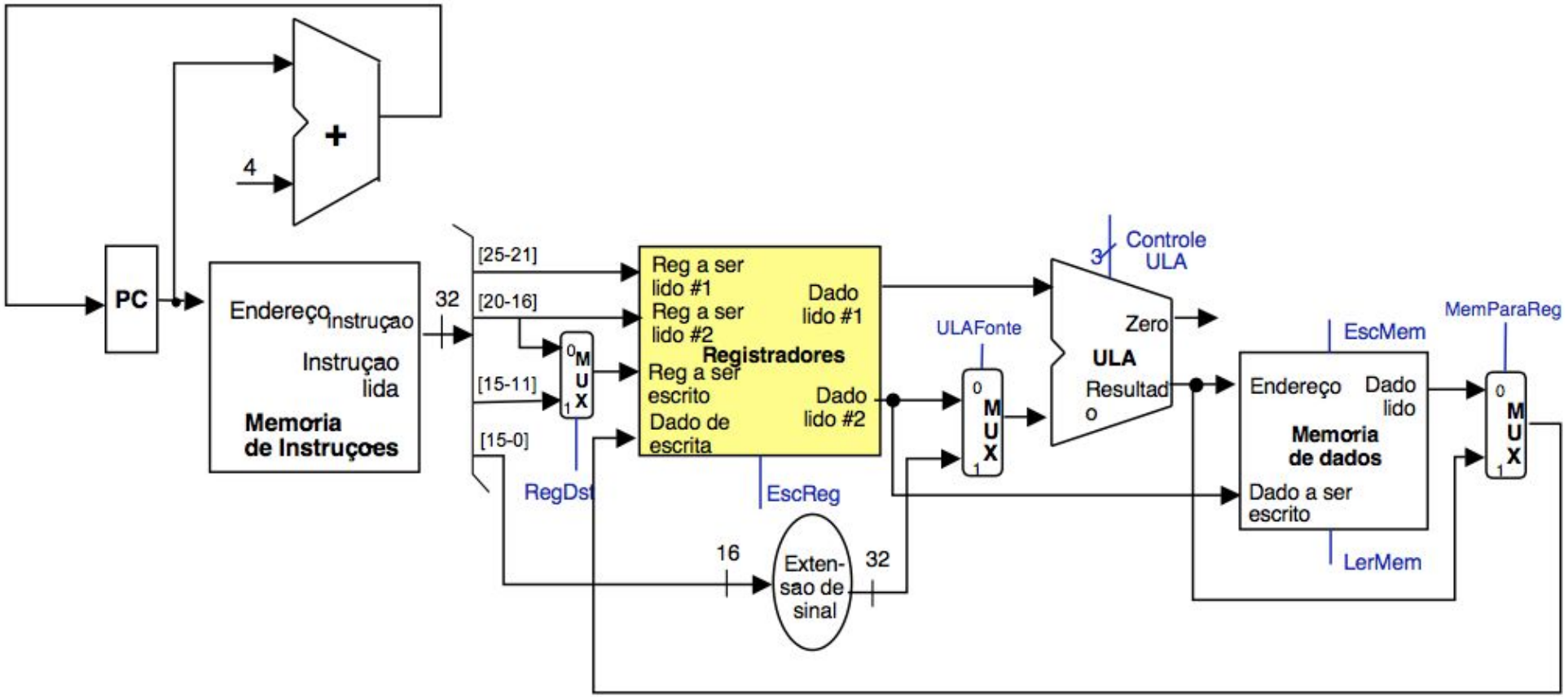
- Para **sw**, campo Rt (bits 20-16) designam registrador cujo conteúdo será escrito na memória de dados
- Para **lw**, Rt (bits 20-16) designam registrador que será carregado com valor lido da memória de dados

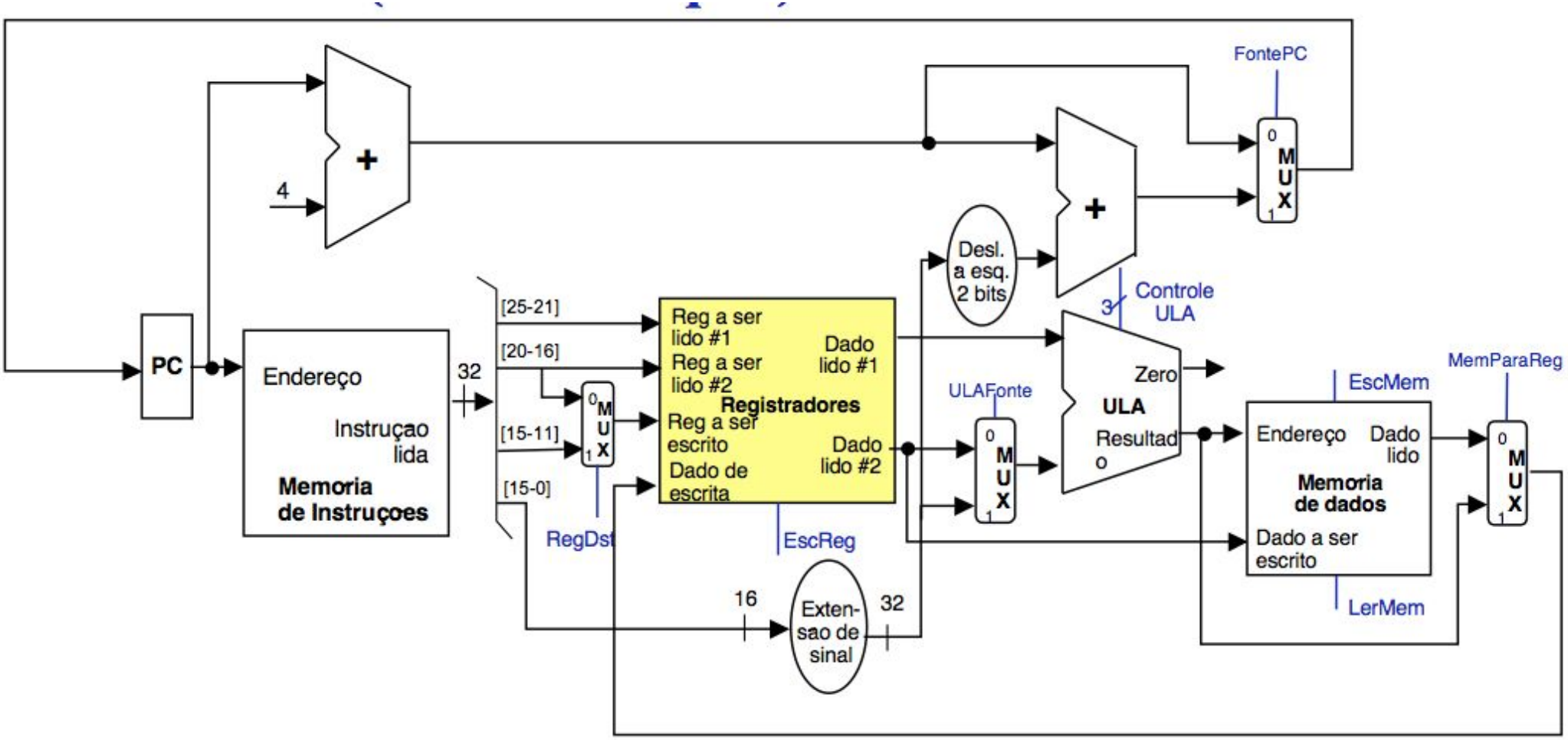
Combinando os recursos para executar instruções tipo R ou instruções lw e sw…



- 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)

Acrescentando os Recursos para a Busca da Instrução e o Cálculo do Próximo Endereço (exceto em desvios)




FontePC
4
+
+
PC
Endereço
Instruçao lida
Memoria de Instruçoes
32
[25-21]
[20-16]
[15-11]
[15-0]
0 M U X 1
RegDst
Reg a ser lido #1
Reg a ser lido #2
Registradores
Reg a ser escrito
Dado de escrita
Dado lido #1
Dado lido #2
EscReg
Desl. a esq. 2 bits
3
Controle ULA
ULAFonte
Zero
ULA
Resultado
Endereço
Dado lido
Memoria de dados
Dado a ser escrito
EscMem
LerMem
MemParaReg
16
Exten-sao de sinal
32

- ao final de cada ciclo de relógio o PC é carregado com um novo valor
- mudança no valor do PC se propaga através de uma grande lógica combinacional

  memória de instruções => banco de registradores => ULA =>
  memória de dados => banco de registradores
- período do ciclo de relógio deve ser maior do que máximo atraso de propagação através desta lógica combinacional

Exemplo de Cálculo

- supondo os seguintes atrasos:
  - memórias: 4 ns
  - ULA: 2 ns
  - banco de registradores: 1 ns
  - somadores: 1 ns
  - demais componentes: atraso desprezível

instrução	busca	Lê registradores	Cálculo na ULA	Acessa memória de dados	Escreve em registrador	Total
add, sub, and, or	4ns	1ns	2ns	--	1ns	8ns
beq	4ns	1ns	2ns	--	--	7ns
sw	4ns	1ns	2ns	4ns	--	11ns
lw	4ns	1ns	2ns	4ns	1ns	12ns

período do ciclo de relógio deve ser maior do que 12 ns (cerca de 83 MHz)

Arquitetura MIPS

Tipos de Dados

- **dados inteiros disponíveis em instruções load e store**
  - bytes
  - meias-palavras de 16 bits
  - palavras de 32 bits
- **dados inteiros disponíveis em instruções aritméticas e lógicas**
  - meias-palavras de 16 bits (estendidos para 32 bits)
  - palavras de 32 bits

Modos de Endereçamento

- **acessos à memória devem ser alinhados**
  - dados de 32 bits precisam iniciar em endereços múltiplos de 4
  - dados de 16 bits precisam estar em endereços múltiplos de 2
- **modo registrador**
  - para instruções aritméticas e lógicas: dado está em registrador
  - para instruções de desvio incondicional: endereço está em registrador

Modos de Endereçamento

- **modo base e deslocamento**
  - para instruções load e store
  - base é registrador inteiro de 32 bits
  - deslocamento de 16 bits contido na própria instrução
- **modo relativo ao PC**
  - para instruções de branch condicional
  - endereço é a soma do PC com deslocamento contido na instrução
  - deslocamento é dado em palavras e precisa ser multiplicado por 4

Modos de Endereçamento

- **modo imediato**
  - para instruções aritméticas e lógicas
  - dado imediato de 16 bits contido na própria instrução
  - dado é estendido para 32 bits
    - extensão com sinal nas instruções aritméticas
    - extensão sem sinal nas instruções lógicas
- **para que se possa especificar constantes de 32 bits**
  - instrução lui (load upper immediate)
  - carrega 16 bits imediatos na parte superior do registrador
  - parte inferior do registrador é zerada
  - instrução seguinte precisa fazer soma imediata do registrador com 16 bits da parte inferior