Universidade Federal de Santa Catarina

Centro Tecnológico

Departamento de Informática e Estatística

Ciências da Computação & Engenharia Eletrônica

Sistemas Digitais

INE 5406

Aula 7-T

O Processador MIPS: conjunto de instruções e exemplos de uso (noções de programação assembly).

Prof. José Luís Güntzel e Cristina Meinhardt

& Est. André Beims Bräscher

{j.guntzel, cristina.meinhardt }@inf.ufsc.br,

andre.brascher@grad.ufsc.br

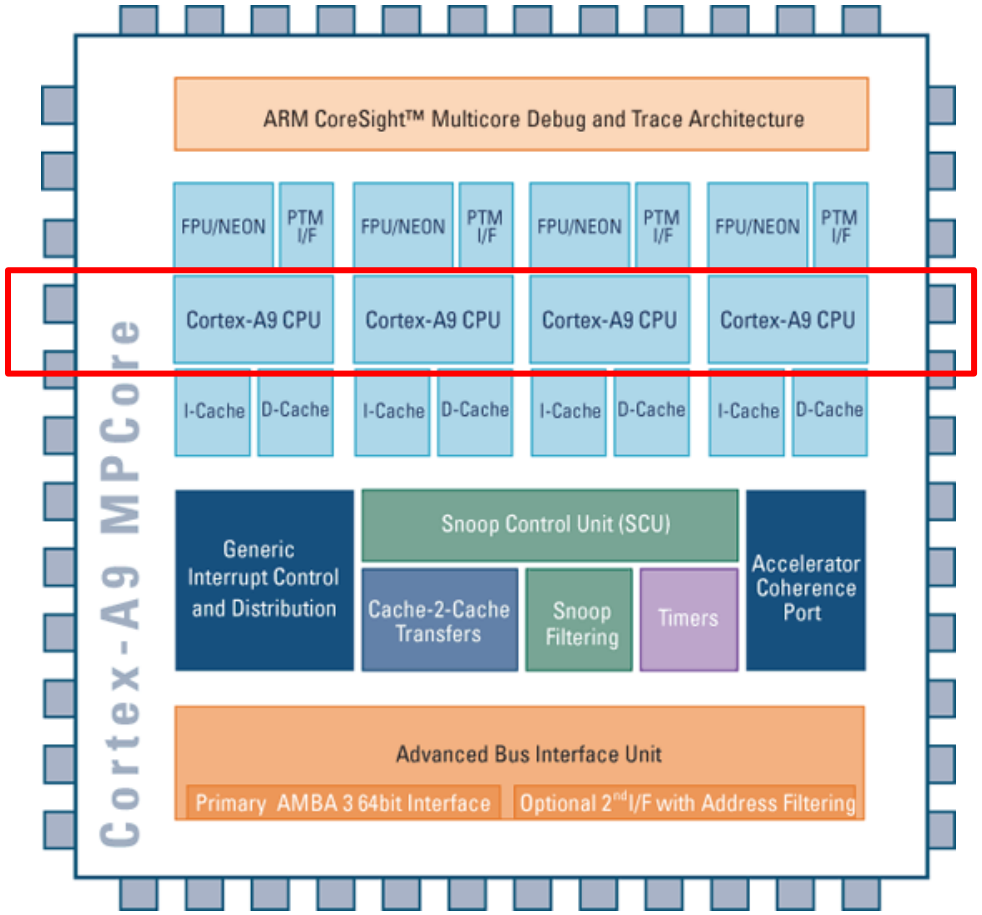
1. Sistemas Digitais e Nível RT

Classificação dos Sistemas Digitais Quanto à Aplicação

1. Processadores de Propósito Geral (CPUs* ou GPPs**):

- Podem ser programados para executar (virtualmente) **qualquer algoritmo**
- Para tanto, são projetados para executar um **conjunto de instruções**
- Otimizados para realizar o **conjunto de instruções** para o qual são projetados (e não um algoritmo ou uma classe de algoritmo)

** Central Processing Units*
*** General-Purpose Processors*



1. Sistemas Digitais e Nível RT

Usando uma CPU (GPP): do programa à execução

Programa em Linguagem de Alto Nível

```
swap (int v[ ] , int k)
{
  int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

* Específico para a linguagem de alto nível e CPU escolhidos

Compilador*

Programa em Linguagem de montagem

```
swap:   muli $2,$5, 4
        add $2,$4,$2
        lw $15, 0($2)
        lw $16, 4($2)
        sw $16, 0($2)
        sw $15, 4($2)
        jr $31
```

Montador**

** Específico da CPU escolhida

Programa em Linguagem de máquina (i.e., "executável")**

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100000000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
```

O Processador MIPS Monociclo

Características Arquiteturais (RISC*)

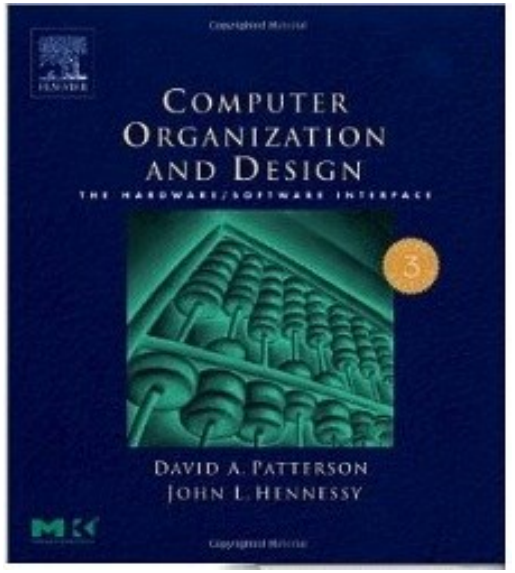
1. Poucas instruções, instruções simples
2. Acesso à memória somente com **LOAD** e **STORE**
3. Operações lógicas e aritméticas entre registradores (instruções com três endereços de registrador)
4. Todas as instruções do mesmo tamanho (com pouca variação de formato)
5. Poucos modos de endereçamento (e codificados junto com as instruções)
6. Uso de instruções *compare-and-branch*
7. Poucos tipos de dados

* Reduced Instruction Set Computer

O Processador MIPS Monociclo

Processador MIPS

- **Arquitetura didática desenvolvida por D. Patterson & J. Hennessy para ensinar Organização de Computadores no nível de graduação**
- **Processadores comerciais da empresa MIPS Technologies (fundada em 1984 por um grupo de pesquisadores de Stanford University, entre eles J. Hennessy).**
  - Estes processadores equiparam diversos produtos: computadores pessoais, estações de trabalho e servidores, roteadores e consoles de jogos



O Processador MIPS Monociclo

Registradores Visíveis ao Programador

32 registradores (de 32 bits) de propósito geral

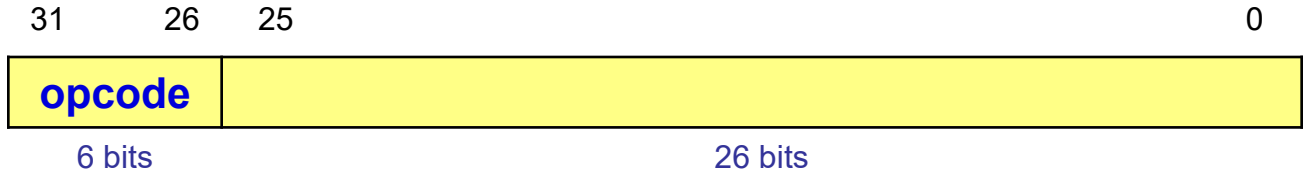
Na linguagem simbólica (assembly) os registradores são designados por:

- **$s0, $s1, …, $s7**
  - registradores que correpondem às variáveis dos programas escritos em linguagem de alto nível (C, por exemplo)
  - São mapeados nos registradores reais de número 16 a 23
- **$t0, $t1,…, $t9**
  - registradores temporários, necessários à tradução dos programas em linguagem de alto nível em instruções do MIPS
  - São mapeados nos registradores de número 8 a 15, 24 e 25

O Processador MIPS Monociclo

Formato das Instruções

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



O Processador MIPS Monociclo

Instruções Consideradas

Instrução	Formato	Linguagem de Montagem	Significado
Adição	R	`add $s1, $s2, $s3`	`$s1 ← $s2 + $s3`
Subtração	R	`sub $s1, $s2, $s3`	`$s1 ← $s2 - $s3`
AND bit a bit	R	`and $s1, $s2, $s3`	`$s1 ← $s2 and $s3`
OR bit a bit	R	`or $s1, $s2, $s3`	`$s1 ← $s2 or $s3`
Load word	I	`lw $s1, desl($s2)`	`$s1 ← Mem[$s2 + desl]`
Store word	I	`sw $s1, desl($s2)`	`Mem[$s2 + desl] ←$s1`
Salto condicional	I	`beq $s1, $s2, desl`	`if ($s1==$s2)` `then PC ← PC+4+(desl<<2)`
Salto incondicional	J	`j L`	`PC ← L onde` `L = ((PC+4)[31-28]) || (constante << 2)`

A interface Hardware/Software

Sobre o compilador

- O Compilador é responsável por
  - associar variáveis a registradores
  - alocar em endereços de memória certas estruturas de dados (tais como os arrays)
  - otimizar o código gerado
- Assim, é fácil para o compilador colocar o endereço inicial nas instruções de transferência de dados

Endereçamento de Memória no MIPS

- Quase todas as arquiteturas endereçam a memória a bytes
- O endereço de uma palavra deve ser igual ao endereço de um de seus bytes (mas sempre o mesmo)

endereço	dados
0	100
4	10
8	101
12	1
⋮	⋮

Endereçamento de Memória no MIPS

- O endereço de duas palavras consecutivas na memória se difere sempre de 4 unidades
- O espaço de endereçamento de memória do MIPS é de 2^{30} palavras (de 32 bits):

endereço	dados
0	100
4	10
8	101
12	1
⋮	⋮
4294967292	77

Endereçamento de Memória no MIPS

- No MIPS as palavras sempre começam em endereços múltiplos de 4 (restrição de alinhamento)
- MIPS usa o endereçamento *big endian*

End.	Memória
i	byte3 (mais sig.)
i+1	byte2
i+2	byte1
i+3	byte0 (menos sig.)

- O endereçamento a bytes também afeta a indexação dos arrays: a instrução lw do exemplo anterior precisa ser

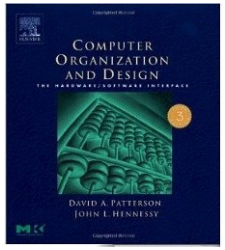
```
lw      $t0, 32 ($s3)      # registrador temporário $t0 recebe A[8]
```

O Processador MIPS Monociclo

Leitura da Semana

Livro:

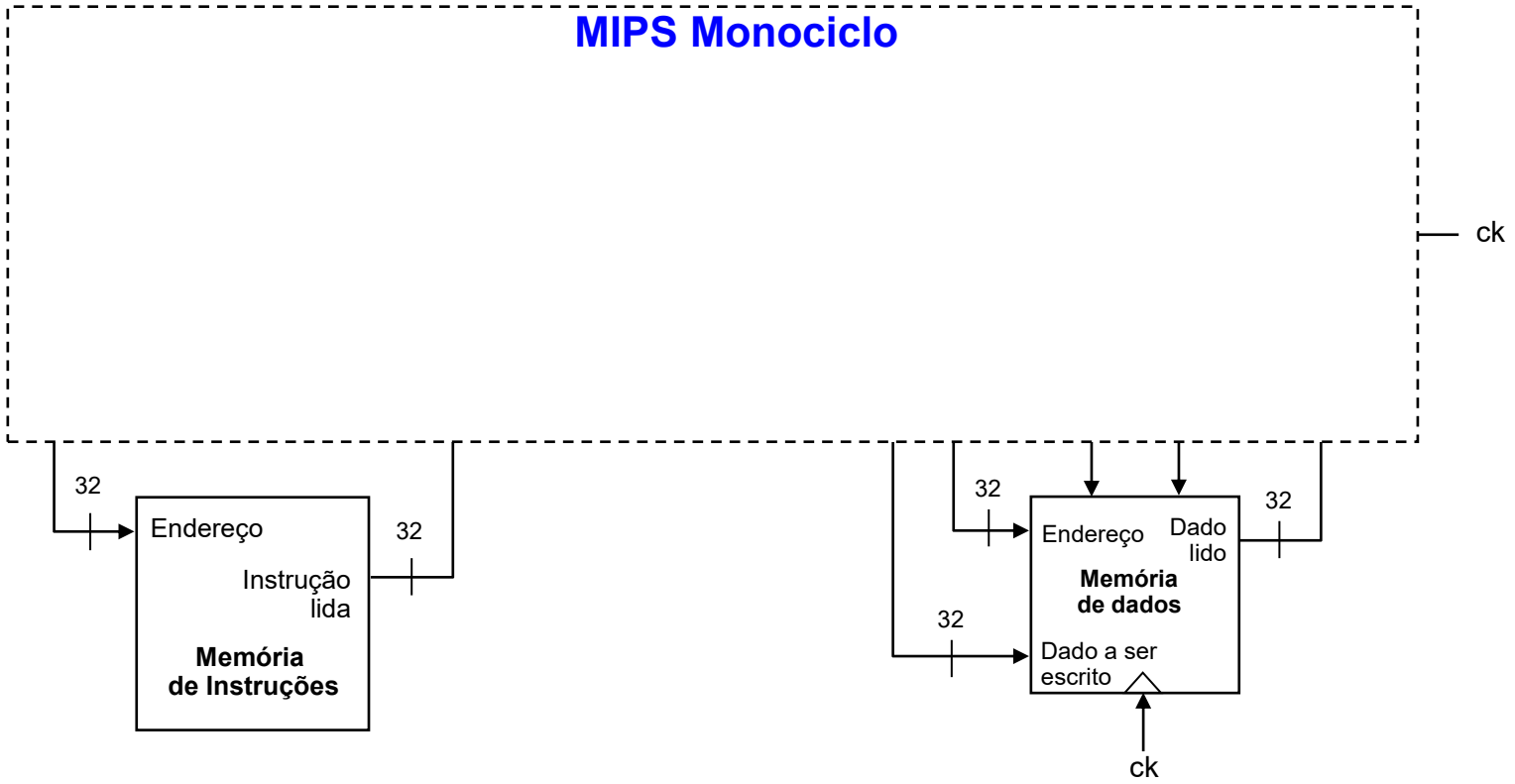
PATTERSON, David A.; HENNESSY, John L. "Computer Organization and Design: the hardware/software Interface", 3rd edition, Morgan Kaufmann Publishers, San Francisco, California, USA, 2007.



Se usar a 2ª Edição: Seções 3.3 a 3.5.
Se usar a 3ª Edição: Seções 2.3 a 2.6.

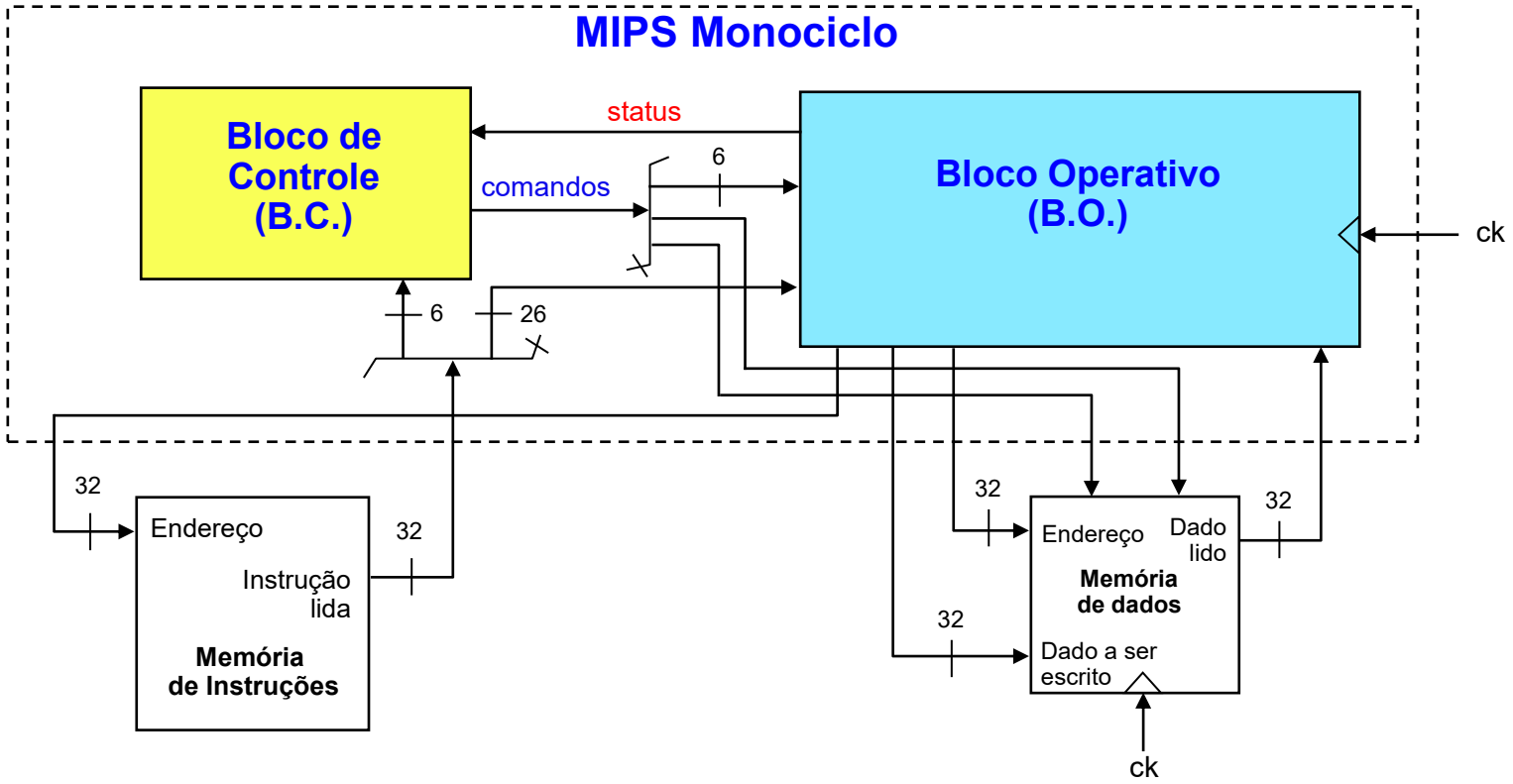
O Processador MIPS Monociclo

Diagrama de Blocos do Sistema



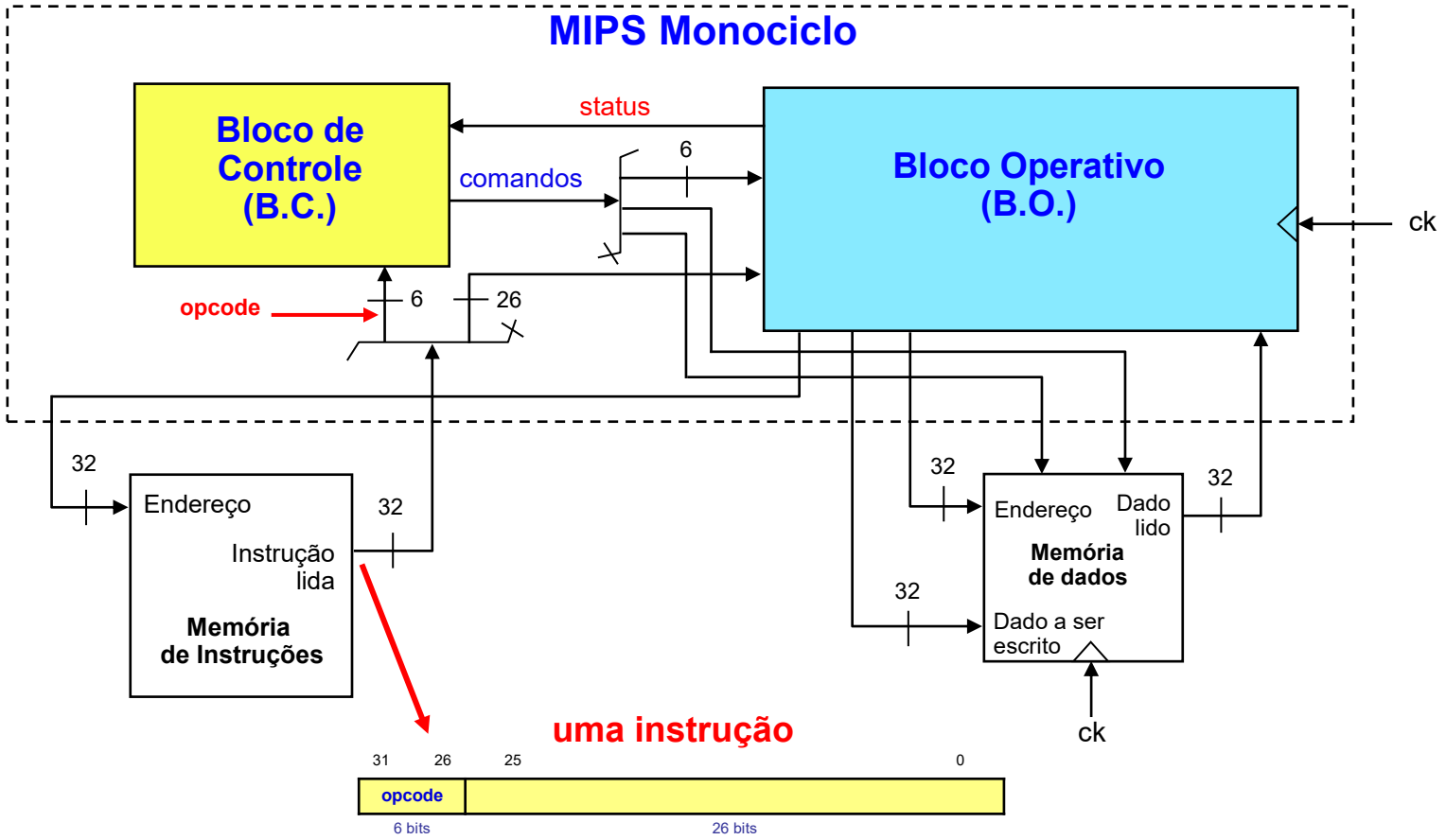
O Processador MIPS Monociclo

Diagrama de Blocos do Sistema



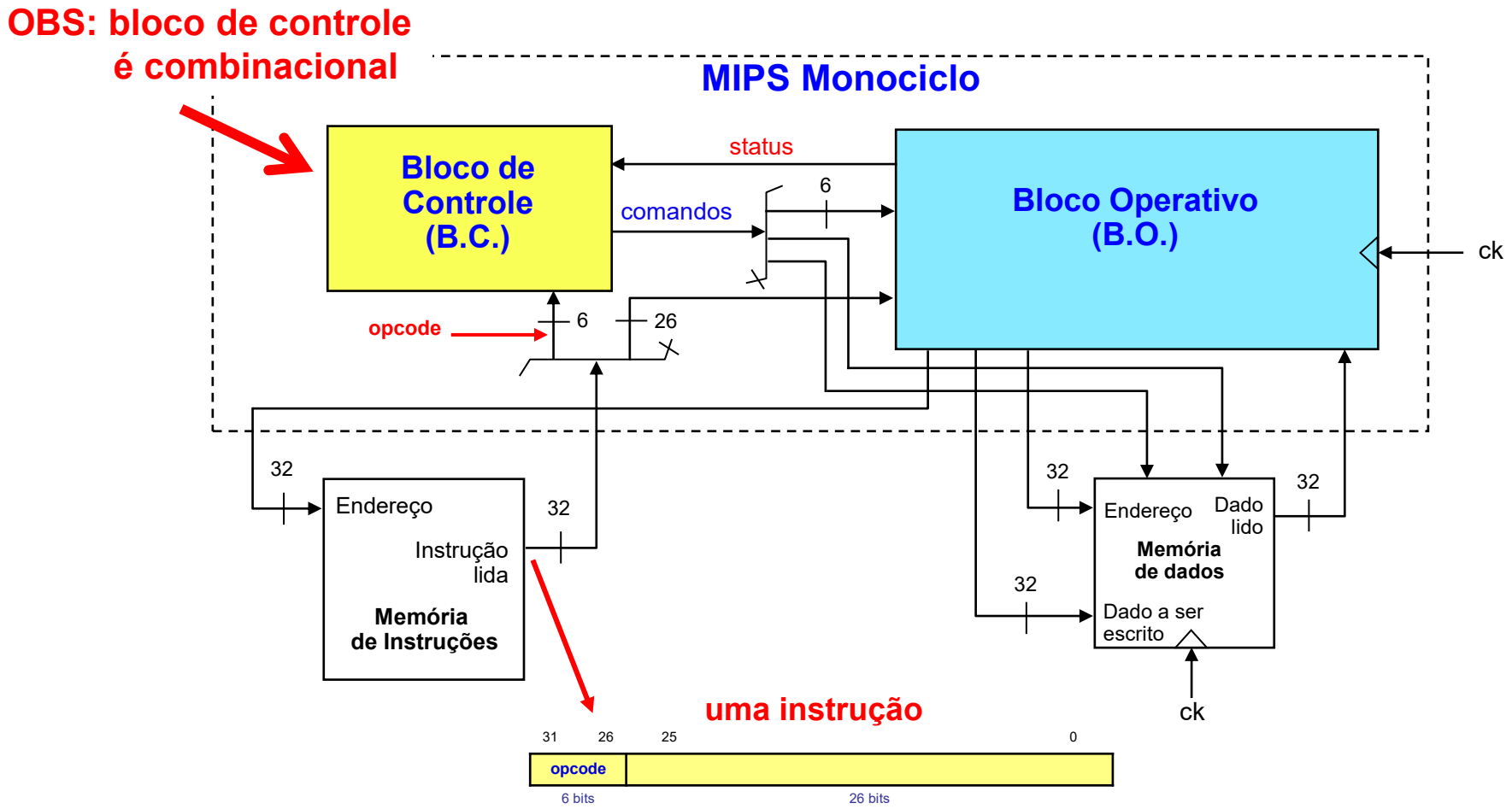
O Processador MIPS Monociclo

Diagrama de Blocos do Sistema



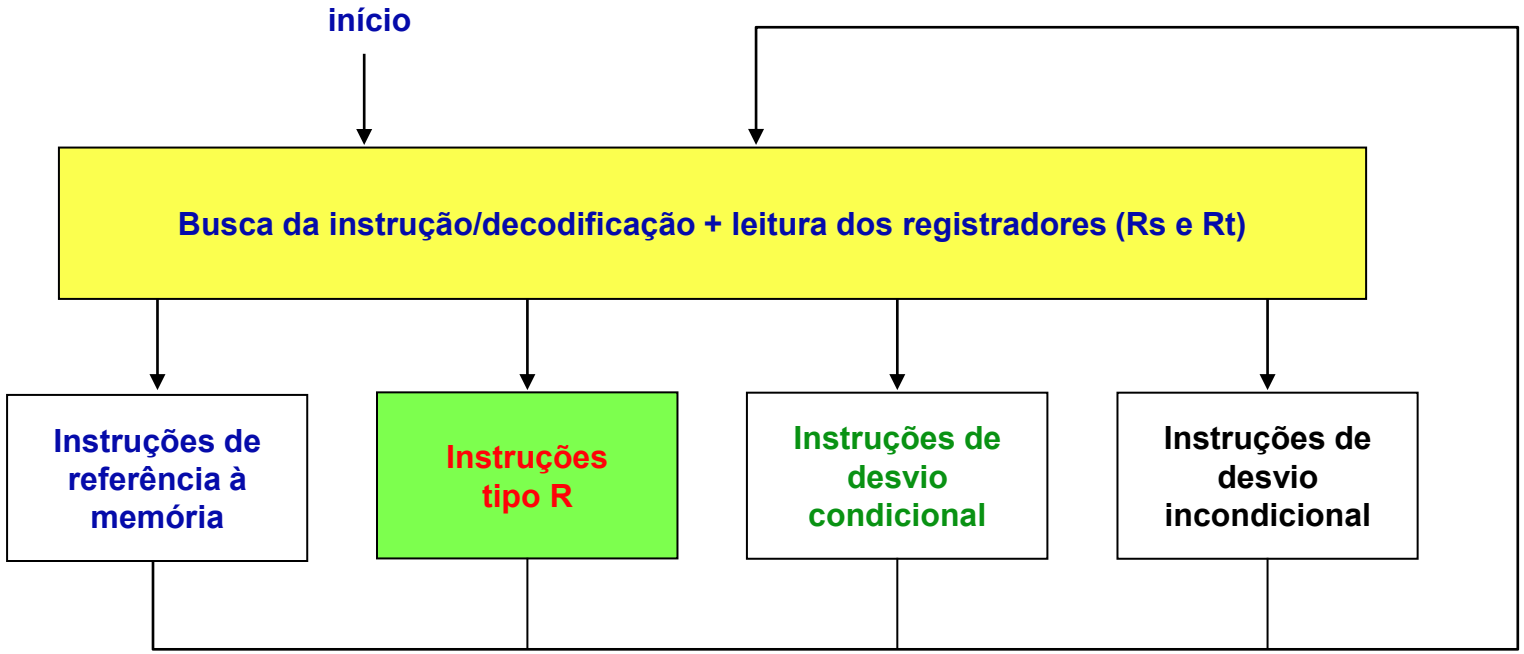
O Processador MIPS Monociclo

Diagrama de Blocos do Sistema



O Processador MIPS Monociclo

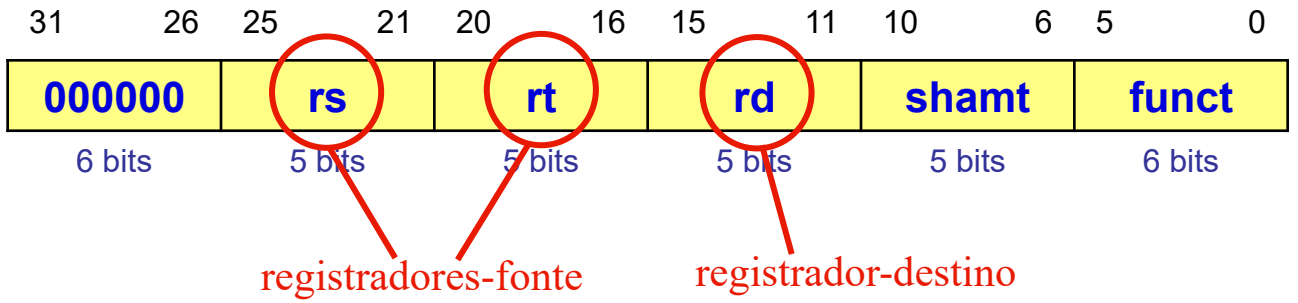
Execução das Instruções ("Ciclo de *Fetch*")



O Processador MIPS Monociclo

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`)

O Processador MIPS Monociclo

Uso de Instruções Tipo R (Aritméticas e Lógicas)

Seja o comando C mostrado abaixo.

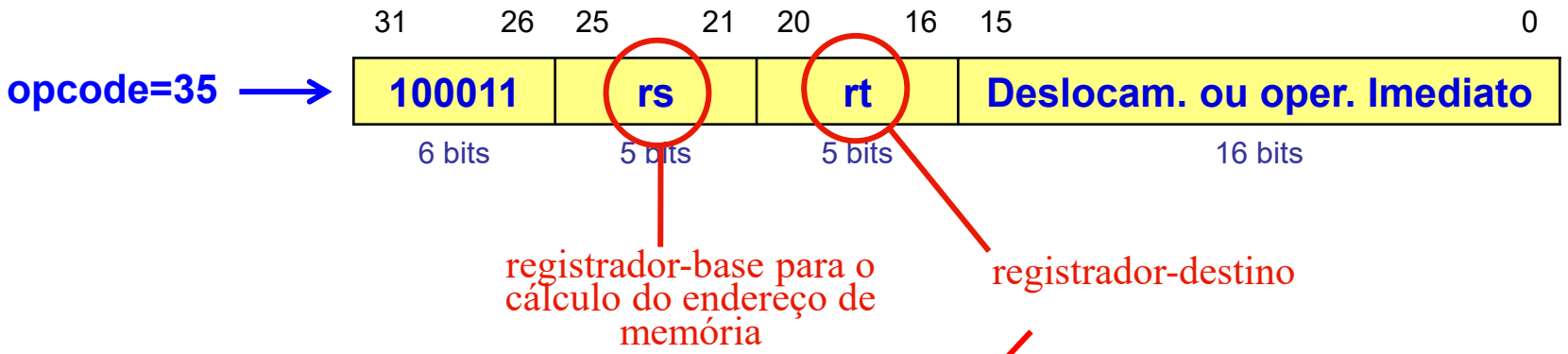
f = (g + h) - (i + j);

Um possível resultado da compilação deste comando para o MIPS seria:

```
add   $t0, $s1, $s2     # registrador $t0 contém g + h
add   $t1, $s3, $s4     # registrador $t1 contém i + j
sub   $s0, $t0, $t1     # f recebe $t0 - $t1, resultado final
```

O Processador MIPS Monociclo

Instruções formato I: load word (lw)

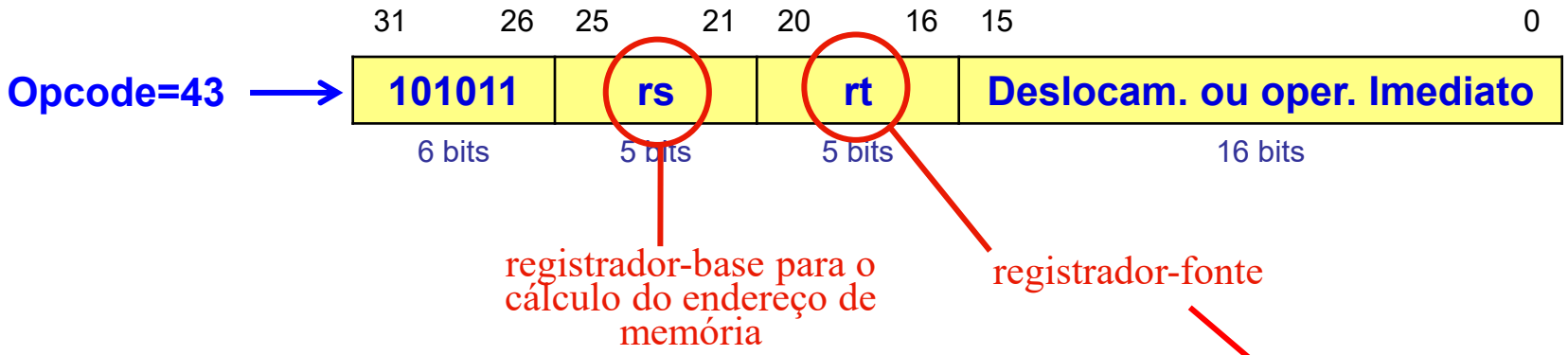


Simbólico

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

O Processador MIPS Monociclo

Instruções formato I: store word (sw)



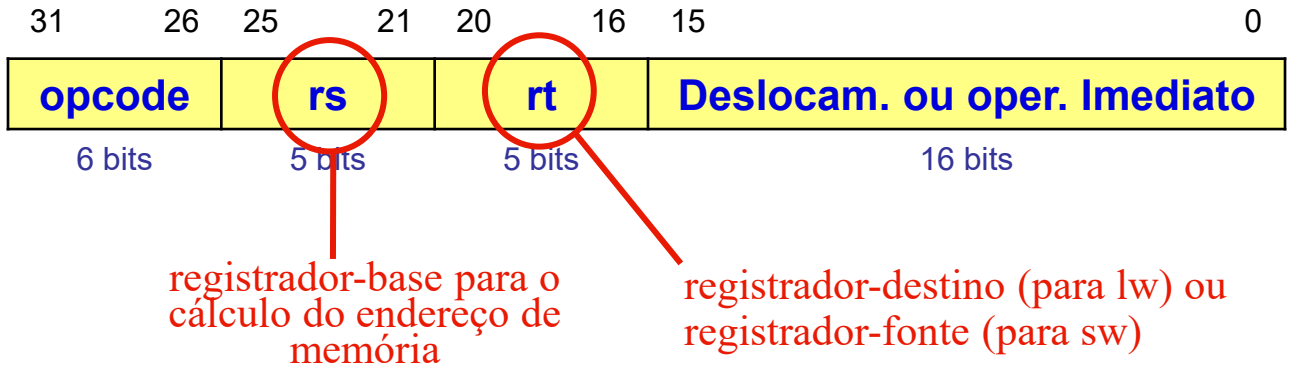
Simbólico

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

O Processador MIPS Monociclo

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 )
```

Uso da Instrução lw

Acesso a um operando que está na memória

Suponha que:

- A **seja um array de 100 palavras,**
- **O compilador associa as variáveis** g **e** h **aos registradores** $s1 **e** $s2
- **O endereço inicial do array (endereço-base) está armazenado em** $s3.

Traduza o seguinte comando de atribuição, escrito em C para a linguagem de montagem do MIPS.

```
g = h + A[8];
```

Uso da Instrução lw

Primeiramente, é necessário transferir A[8] **para um registrador:**

```
lw      $t0, 8 ($s3)        # registrador temporário $t0 recebe A[8]
```

A instrução seguinte pode operar normalmente com o valor trazido da memória, pois ele está armazenado no registrador temporário $t0:

```
add   $s1, $s2, $t0       # g recebe h + A[8]
```

O Processador MIPS Monociclo

Uso das Instruções lw e sw

- Suponha que a variável h esteja associada ao registrador $s2 e que o endereço do array A esteja armazenado em $s3
- Qual é o código de montagem do MIPS para o seguinte comando de atribuição, escrito em C?

A[12] = h + A[8]

```
lw    $t0, 32 ($s3)    # registrador temporário $t0 recebe A[8]
add   $t0, $s2, $t0    # registrador temporário $t0 recebe h + A[8]
sw    $t0, 48 ($s3)    # h + A[8] é armazenado em A[12]
```

O Processador MIPS Monociclo

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 << 2) + PC + 4**

Simbólico

```
beq $s1, $s2, deslocam   ( if ($s1== $s2) then PC←PC + 4 +
(deslocam << 2) )
```

O Processador MIPS Monociclo

Instruções de Desvio Condicional

- **Usadas para instruções que envolvem tomada de decisão (if, while, etc.)**
- **No MIPS:**

```
beq reg1, reg2, L1
```

Branch if equal: desvia para o comando com label L1**, se** reg1 == reg2

```
bne reg1, reg2, L1
```

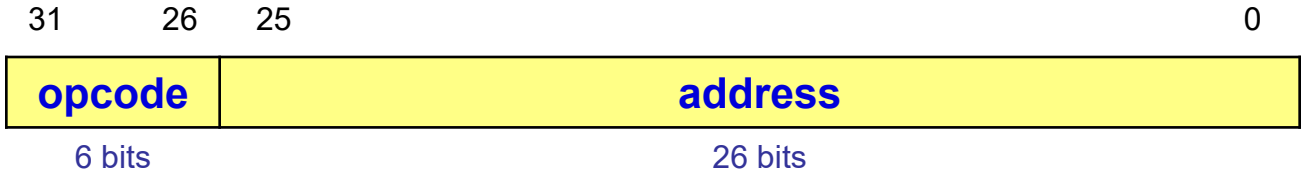
Branch if not equal: desvia para o comando com label L1**, se** reg1 != reg2

O Processador MIPS Monociclo

Instrução formato J: Desvio Incondicional

J: jump

- **Opcode = 2**
- **Campo address contém 26 bits do endereço de destino**
- **Os 2 bits menos significativos recebem "0"**
- **Os 4 bits mais significativos recebem os bits mais significativos de PC + 4**



Simbólico

```
j Label (PC← ((PC+4)[31-28]) || (endereço destino << 2))
```

Uso das Instruções beq e j

Compilação de um Comando *if-else* usando Desvio Condicional e Desvio Incondicional

- **Seja o código a seguir, escrito em linguagem C:**

```
if( i != j) f = g + h;
else f = g - h;
```

- **Suponha que as variáveis** f, g, h, i **e** j **sejam alocadas nos registradores** $s0, $s1, $s2, $s3 **e** $s4, **respectivamente**
- **Qual seria o código gerado pelo compilador do MIPS?**

Uso da Instrução beq

Compilação de um Comando *if* em uma Instrução de Desvio Condicional

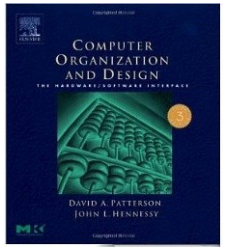
```
1            beq   $s3, $s4, ELSE    # desvia para ELSE se i == j
2            add   $s0, $s1, $s2     #  f = g + h (executa  se i != j)
3              j   EXIT              # Fim da condição if, vai para EXIT
4     ELSE:    sub   $s0, $s1, $s2     # f = g - h (executa se i == j)
5      EXIT:                         Fim da execução
```

O Processador MIPS Monociclo

Leitura da Semana

Livro:

PATTERSON, David A.; HENNESSY, John L. "Computer Organization and Design: the hardware/software Interface", 3rd edition, Morgan Kaufmann Publishers, San Francisco, California, USA, 2007.



Se usar a 2ª Edição: Seções 3.3 a 3.5.
Se usar a 3ª Edição: Seções 2.3 a 2.6.