Arquitetura MIPS
Instruções especiais e paralelismo

Tópicos da aula

- **Transferências de Bytes e Half Words**
- **Operações *signed* e *unsigned***
- **Operações em ponto flutuante**
- **Paralelismo subword**
- **Operações atômicas – *multicore processor***

Movimentação entre registradores e memória

Função	Mnemônico		Exemplo
Carrega da memória para o registrador: MEMÓRIA → REG	LB	Load Byte signed	`lb $t1, 0($s1)`
	LBU	Load Byte Unsigned	`lbu $t1, 0($s1)`
	LH	Load Halfword	`lh $t1, 0($s1)`
	LHU	Load Halfword Unsigned	`lhu $t1, 0($s1)`
	LW	Load Word	`lw $t1, 0($s1)`
Carrega imediato [31:16]	LUI	Load Upper Immediate	`lui $t1, 25`
Move conteúdo de registradores	MOVE	Movement	`move $t1, $t2`
Armazena conteúdo do registrador na memória REG → MEMÓRIA	SB	Store Byte	`sb $t1, 0($s1)`
	SH	Store Halfword	`sh $t1, 0($s1)`
	SW	Store Word	`sw $t1, 0($s1)`

Transferência: Memória → Registrador

- **Load Byte** – LB
  - Instrução no formato **I**
  - Carregar o byte inferior com **sinal estendido** do conteúdo da memória apontado em rs no registrador rt
  - São endereçados 1 a 1 dentro de uma posição de memória
- **Exemplo**

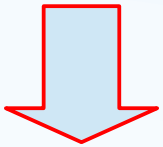
```
LB   rt, imm(rs)

lb   $t2, 0($s0)    # Carrega t2 com conteúdo apontado pelo endereço em s0
```

31:26	25:21	20:16	15:0
100000	rs	rt	imm

Exemplo 1 – LB de número (+)

```
1   .data
2           DADO_B: .byte 2 3 4 5 6
3
4   .text
5   main:
6           la      $t1, DADO_B
7           lb      $t2, 1($t1)
```



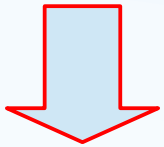
Data Segment

Address	Value (+0)	Value (+4)
0x10010000	0x05040302	0x00000006
0x10010020	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000

$t1	9	0x10010000
$t2	10	0x00000003
$t3	11	0x00000000

Exemplo 2 – LB de número (-)

```
1   .data
2           DADO_B: .byte 2 -3 4 5 6
3
4   .text
5   main:
6           la      $t1, DADO_B
7           lb      $t2, 1($t1)
```



Data Segment

Address	Value (+0)	Value (+4)
0x10010000	0x0504fd02	0x00000006
0x10010020	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000

$t1	9	0x10010000
$t2	10	0xfffffffd
$t3	11	0x00000000

Transferência: Memória → Registrador

- **Load Byte Unsigned** – LBU
  - Instrução no formato **I**
  - Carregar o byte inferior com **extensão de zeros** do conteúdo da memória no endereço apontado por rs no registrador rt
  - São endereçados 1 a 1 dentro de uma posição de memória
- **Exemplo**

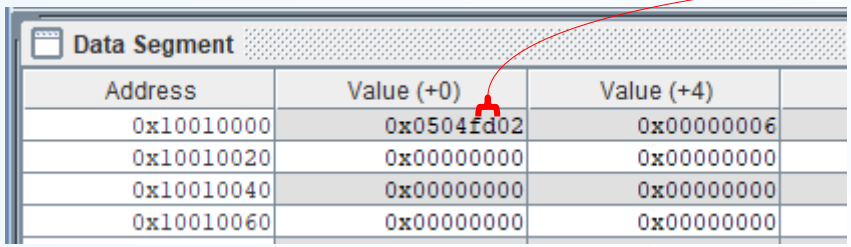
```
LBU     rt, imm(rs)

lbu   $t2, 0($s0)     # Carrega t2 com conteúdo apontado pelo endereço em s0
```

31:26	25:21	20:16	15:0
100100	rs	rt	imm

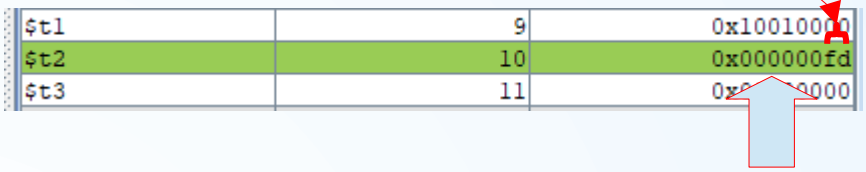
Exemplo 3 – LBU de número (-)

```
1   .data
2           DADO_B: .byte 2 -3 4 5 6
3
4   .text
5   main:
6           la      $t1, DADO_B
7           lbu     $t2, 1($t1)
```



Data Segment

Address	Value (+0)	Value (+4)
0x10010000	0x0504fd02	0x00000006
0x10010020	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000



$t1	9	0x10010000
$t2	10	0x000000fd
$t3	11	0x00000000

Transferência: Memória → Registrador

- **Load Halfword** – LH
  - Instrução no formato **I**
  - Carregar os dois bytes inferiores com **sinal estendido** do conteúdo da memória apontado por rs no endereço indicado por rt
  - São endereçados 2 a 2 dentro de uma posição de memória
- **Exemplo**

```
LH   rt, imm(rs)

lh   $t0, 4($s0)    # Carrega t2 com conteúdo apontado pelo endereço em s0
```

31:26	25:21	20:16	15:0
100001	rs	rt	imm

Transferência: Memória → Registrador

- **Load Halfword Unsigned** – LHU
  - Instrução no formato **I**
  - Carregar os dois bytes inferiores com **extensão de zeros** do conteúdo da memória no endereço
  - São endereçados 2 a 2 dentro de uma posição de memória
- **Exemplo**

  LHU rt, imm(rs)

  lhu $t2, 0($s0) # Carrega t2 com conteúdo apontado pelo endereço em s0

31:26	25:21	20:16	15:0
100100	rs	rt	imm

Transferência: Memória → Registrador

- **Load Word** – LW
  - Instrução no formato **I**
  - Carregar o conteúdo da memória que está no endereço apontado por rs no registrador rt
  - São endereçados 4 e 4 na memória
- **Exemplo**

```
LW   rt, imm(rs)

lw   $t2, 0($s0)     # Carrega t2 com conteúdo apontado pelo endereço em s0
```

31:26	25:21	20:16	15:0
100001	rs	rt	imm

Transferência: Imediato → Registrador

- **Load Upper Immediate** – LUI
  - Instrução no formato **I**
  - Carrega a constante imediata de 16 bits nos 16 bits superiores do registrador rt

- **Exemplo**

```
LUI   rt, imm

lui   $t2, 0x55     # Carrega t2 com conteúdo imediato 55 hexadecimal
```

31:26	25:21	20:16	15:0
001111	xxxxx	rt	imm

Transferência: Registrador → Memória

- **Store Byte** – SB
  - Instrução no formato **I**
  - Armazena os 8 bits inferiores contidos em rt nos 8 bits inferiores da memória localizada no endereço armazenado em rs com extensão de sinal.

- **Exemplo**

```
SB   rt, rs

sb   $t2, 0($s0)    # Carrega conteúdo de $t2 em posição de memória ($S0)
```

31:26	25:21	20:16	15:0
101000	rs	rt	imm

Transferência: Registrador → Memória

- **Store Halfword** – SH
  - Instrução no formato **I**
  - Armazena os 16 bits inferiores contidos em rt nos 16 bits inferiores da memória localizada no endereço armazenado em rs com extensão de sinal.

- **Exemplo**

```
SH   rt, rs

sw   $t2, 0($s0)     # Carrega conteúdo de $t2 em posição de memória ($S0)
```

31:26	25:21	20:16	15:0
101001	rs	rt	imm

Transferência: Registrador → Memória

- **Store Word** – SW
  - Instrução no formato **I**
  - Armazena os o valor contido em rt na posição de memória localizada no endereço armazenado em rs

- **Exemplo**

SW rt, rs

```
sw   $t2, 0($s0)    # Carrega conteúdo de $t2 em posição de memória ($S0)
```

31:26	25:21	20:16	15:0
101011	rs	rt	imm

Movimentação entre registradores, memória e Coprocessador 1

Função	Mnemônico		Exemplo
Carrega da memória para o registrador: MEMÓRIA → REG	LWC1	Load Word into Coprocessor 1	`lwc1 $f1, 0($s1)`
	L.S	Load single precision into Coprocessor 1	`l.s $f1, 0($s1)`
	L.D	Load double precision into Coprocessor 1	`l.d $f2, 0($s1)`
Movimentação entre registradores	MTC1	Move To Coprocessor 1	`mtc1 $t2, $f1`
	MFC1	Move From Coprocessor 1	`mfc1 $t2, $f1`
	MOV.S	Movement between single precision	`mov.s $f1, $f0`
	MOV.D	Movement between double precision	`mov.d $f2, $f0`
Armazena conteúdo do registrador na memória REG → MEMÓRIA	SWC1	Store Word from Coprocessor 1	`sb $t1, 0($s1)`
	S.S	Store Single precison	`sh $t1, 0($s1)`
	S.D	Store Double precison	`sw $t1, 0($s1)`

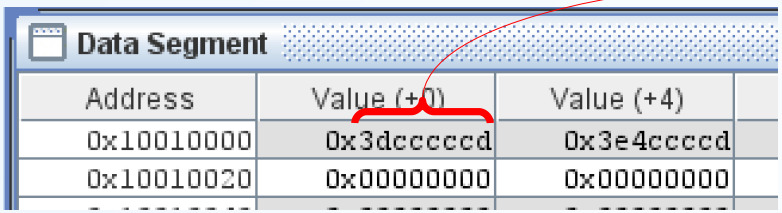
Transferência: Memória → Registrador Ponto Flutuante

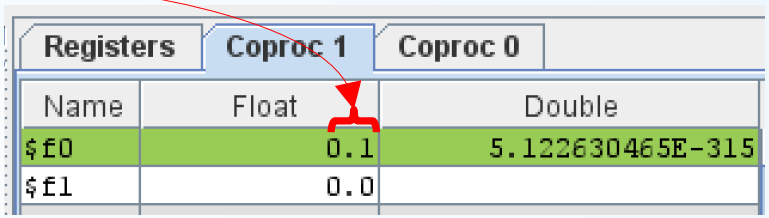
- **Load Word into Coprocessor 1** – LWC1
  - Carrega conteúdo da memória em registrador $fx
  - **Importante**: carrega apenas uma word (precisão simples)

- **Load Single / Double** – L.S / L.D
  - **L.S** - Carrega conteúdo de precisão simples da memória em registrador $fx
  - **L.D**. - Carrega conteúdo de precisão dupla da memória em registrador $fx
  - **Importante**: L.D. só pode ser usado com registradores **pares** ($f0, $f2, $f4, etc.)

Exemplo 4 – l.s

```
1   .data
2           DADO_F: .float 0.1 0.2 0.3 -0.4
3
4   .text
5   main:
6           la      $s0, DADO_F
7           l.s     $f0, 0($s0)
8
```



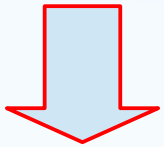


Movimentação entre registradores Ponto Flutuante

- **Entre registradores comuns → Ponto Flutuante** – MTC1, MFC1
  - **MTC1**: carrega conteúdo de registrador comum ($sx, $tx) para registrador $fx
  - **MFC1**: carrega conteúdo de registrador $fx em registrador comum ($sx, $tx)
- **Movimenta Single / Double** – MOV.S / MOV.D
  - **MOV.S** - Movimenta conteúdo de precisão simples entre quaisquer registradores $fx
  - **MOL.D**. - Movimenta conteúdo de precisão dupla apenas entre registradores $fx pares

Exemplo 5 – mov.d

```
1   .data
2           DADO_F: .double 0.01 0.02 0.03 -0.04
3
4   .text
5   main:
6           la      $s0, DADO_F
7           l.d     $f0, 8($s0)
8           mov.d   $f2, $f0
```



Data Segment

Address	Value (+0)	Value (+4)
0x10010000	1202590843	1065646817
0x10010020	0	0
0x10010040	0	0

Registers | **Coproc 1** | **Coproc 0**

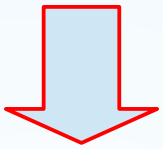
Name	Float	Double
$f0	89128.96	0.02
$f1	1.16	
$f2	89128.96	0.02
$f3	1.16	

Transferência: Registrador → Memória Ponto Flutuante

- **Store Word into Coprocessor 1** – SWC1
  - **SWC1**: carrega conteúdo de registrador comum $fx para memória
  - **Importante**: carrega apenas precisão simples
- **Store Single / Double** – S.S / S.D
  - **S.S** - carrega conteúdo de precisão simples de registrador $fx para posição de memória
  - **S.D**. - carrega conteúdo de precisão dupla de registradores $fx em posição de memória

Exemplo 6 – s.d

```
1   .data
2           DADO_F: .double 0.01 0.02
3           DADO_D: .double
4
5   .text
6   main:
7           la      $s0, DADO_F
8           la      $s1, DADO_D
9           l.d     $f0, 0($s0)
10          s.d     $f0, 0($s1)
```



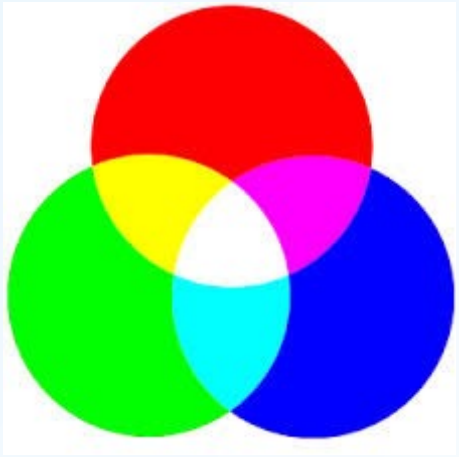
Data Segment

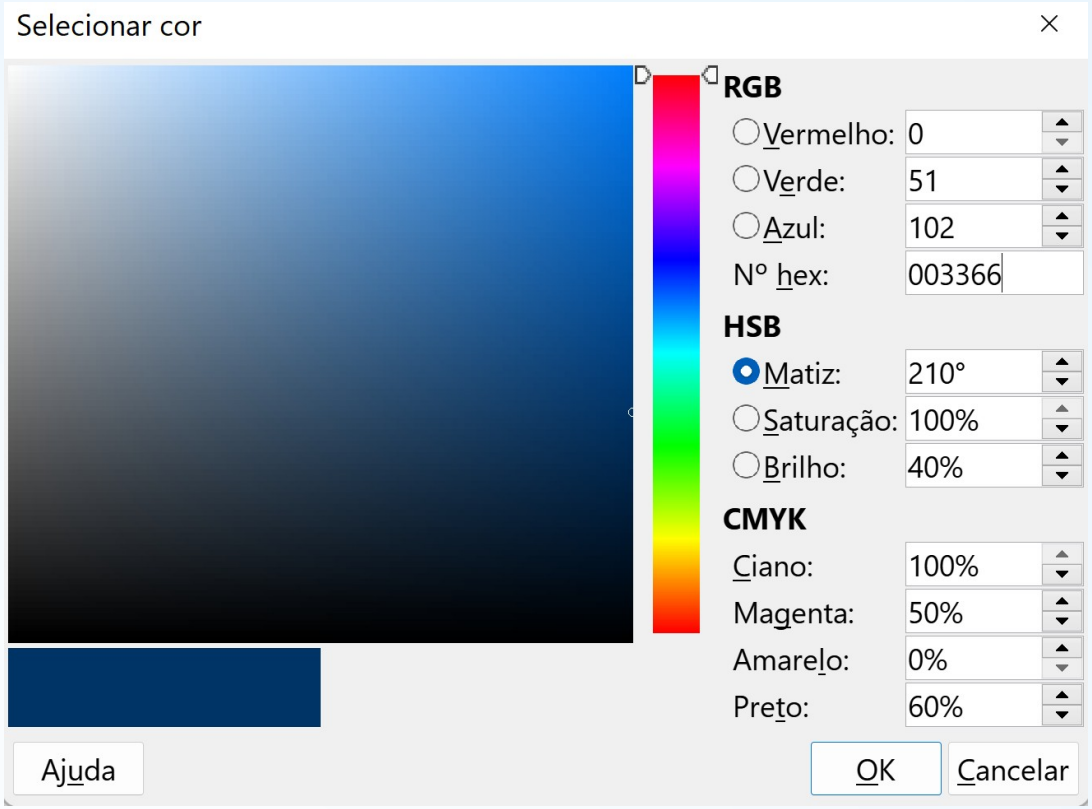
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x10010000	1202590843	1065646817	1202590843	1066695393	1202590843	1065646817
0x10010020	0	0	0	0	0	0

Paralelismo subword

- Em geral, computadores mais elaborados possuem suas próprias telas gráficas
- Os primeiros sistemas gráficos utilizavam sistemas de representação em 8 bits
  - uma imagem **GIF** de **8 bits** pode conter até **256 cores**
  - uma imagem **JPEG** de **24 bits** pode conter aproximadamente **16 milhões** de cores
- Padrões na formação de cores atuais
  - **RGB** - 24 bits (8 bits x 3 canais)
  - **CMYK** - 32 bits (8 bits x 4 canais)

Paralelismo subword



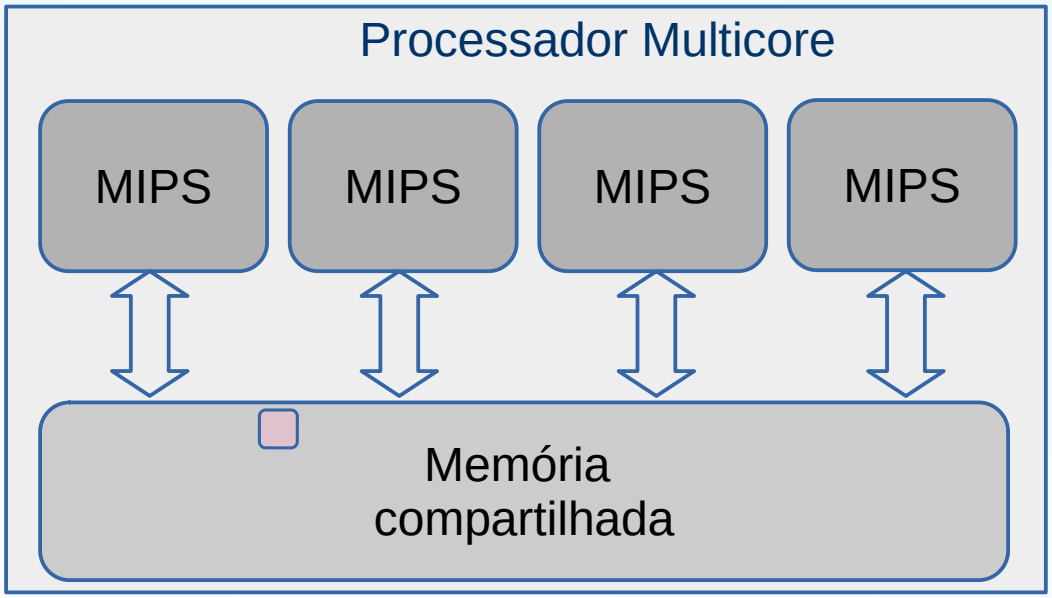


Paralelismo subword

- Processadores dão suporte a operações com Bytes, Halfwords e Words
- Arquitetos de computadores perceberam que operações de Bytes e Halfword poderiam usar de paralelismo para operações. Ou seja: em uma única instrução, realizar múltiplas ações operações como soma de bytes
- Foram criadas instruções específicas para realizar operações paralelas em números com quantidades menores de bits
  - Inteiros com e sem sinal de 8 bits, 16 bits, 32 bits e 64 bits
  - Exemplos: NEO → ARM MMX e SSE → Intel

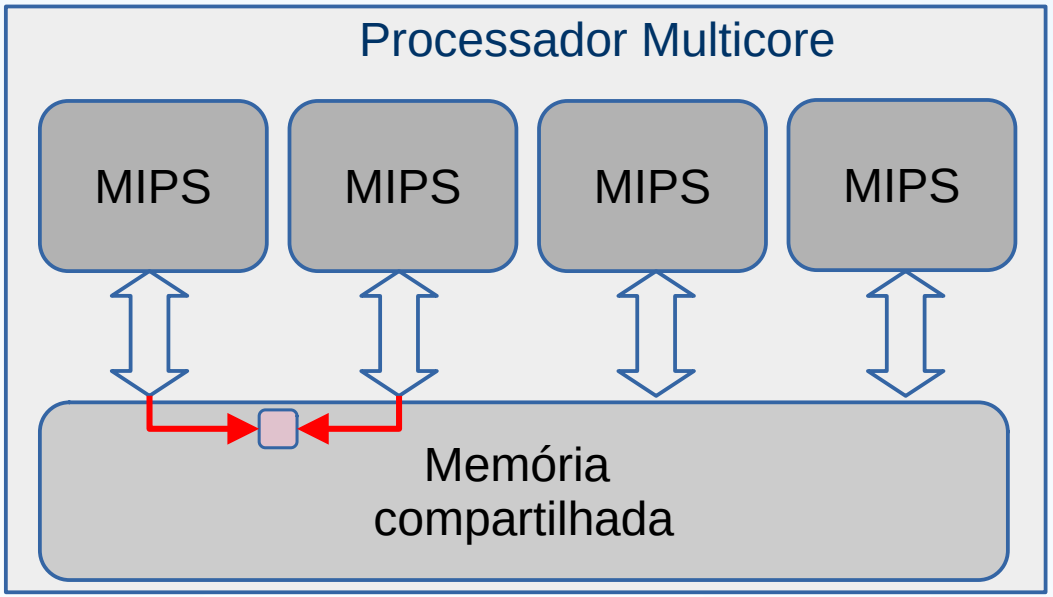
Paralelismos - concorrência por recursos

- Sistemas computacionais modernos normalmente usam mais de um processador, executando tarefas em modo paralelo
- Ocorre uma natural competição por recursos da máquina



Paralelismos - concorrência por recursos

- Acesso a memória cache compartilhada é um exemplo de concorrência (será visto em aulas posteriores)
- Exemplo: dois processadores querem escrever numa mesma posição de memória



Operações atômicas

- Importante no contexto de Sistemas Operacionais
- Garantia de que um recursos será acessado por apenas um processador num determinado instante
- MIPS possui instruções de transferência de dados MEMÓRIA → REGISTRADOR → MEMÓRIA com garantia de acesso atômico
  - **Load Linked** (`LL`) - busca dado de memória compartilhada
  - **Store Conditioned** (`SC`): escreve dado em memória compartilhada

FIM MÓDULO 10