Arquitetura e Organização de Computadores

Linguagem dos Computadores

Prof. André D'Amato

andredamato@utfpr.edu.br

A Linguagem dos Computadores

- Conjunto de Instruções (ISA)
  - O vocabulário dos comandos entendidos por uma determinada arquitetura.
- Princípios básicos semelhantes
  - Operações básicas devem ser fornecidas por todos os hardwares
  - Exemplo: adição
    - add a, b, c # a = b + c
  - E para somar b + c + d?

A Linguagem dos Computadores

- Princípios básicos semelhantes
  - Operações básicas devem ser fornecidas por todos hardware
  - Exemplo: adição
    - add a, b, c # a = b + c
  - E para somar b + c + d?
    - add a, b, c

A Linguagem dos Computadores

- Princípios básicos semelhantes
  - Operações básicas devem ser fornecidas por todos hardware
  - Exemplo: adição
    - add a, b, c # a = b + c
  - E para somar b + c + d?
    - add a, b, c
    - add a, a, d

A Linguagem dos Computadores

- Princípios básicos semelhantes
  - Operações básicas devem ser fornecidas por todos hardware
  - Exemplo: adição
    - add a, b, c # a = b + c
  - E para somar b + c + d?
    - add a, b, c
    - add a, a, d
  - Simplicidade favorece regularidade

A Linguagem dos Computadores

- Outro exemplo:
  - a = (b + c) – (d + e)?

A Linguagem dos Computadores

- Outro exemplo:
  - a = (b + c) – (d + e)?

  - add $t0, b, c
  - add $t1, d, e
  - sub $t3, $t0, $t1

A Linguagem dos Computadores

- Outro exemplo:
  - a = (b + c) – (d + e)?

  - add $t0, b, c
  - add $t1, d, e
  - sub $t3, $t0, $t1

O Simulador QTSPIM

A Linguagem dos Computadores

Categoria	Instrução	Exemplo	Significado
Aritmética	add	add $s1,$s2,$s3	$s1 = $s2 + $s3
	subtract	sub $s1,$s2,$s3	$s1 = $s2 – $s3
	add immediate	addi $s1,$s2,20	$s1 = $s2 + 20
Transferência de dados	load word	lw $s1,20($s2)	$s1 = Memória[$s2 + 20]
	store word	sw $s1,20($s2)	Memória[$s2 + 20] = $s1
	load half	lh $s1,20($s2)	$s1 = Memória[$s2 + 20]
	load half unsigned	lhu $s1,20($s2)	$s1 = Memória[$s2 + 20]
	store half	sh $s1,20($s2)	Memória[$s2 + 20] = $s1
	load byte	lb $s1,20($s2)	$s1 = Memória[$s2 + 20]
	load byte unsigned	lbu $s1,20($s2)	$s1 = Memória[$s2 + 20]
	store byte	sb $s1,20($s2)	Memória[$s2 + 20] = $s1
	load linked word	ll $s1,20($s2)	$s1 = Memória[$s2 + 20]
	store condition, word	sc $s1,20($s2)	Memória[$s2+20]=$s1;$s1=0 or 1
	load upper immed.	lui $s1,20	$s1 = 20 * 2^{16}
Lógica	and	and $s1,$s2,$s3	$s1 = $s2 & $s3
	or	or $s1,$s2,$s3	$s1 = $s2 | $s3
	nor	nor $s1,$s2,$s3	$s1 = ~ ($s2 | $s3)
	and immediate	andi $s1,$s2,20	$s1 = $s2 & 20
	or immediate	ori $s1,$s2,20	$s1 = $s2 | 20
	shift left logical	sll $s1,$s2,10	$s1 = $s2 << 10
	shift right logical	srl $s1,$s2,10	$s1 = $s2 >> 10
Desvio condicional	branch on equal	beq $s1,$s2,25	if ($s1 == $s2) go to PC + 4 + 100
	branch on not equal	bne $s1,$s2,25	if ($s1!= $s2) go to PC + 4 + 100
	set on less than	slt $s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1 = 0
	set on less than unsigned	sltu $s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1 = 0
	set less than immediate	slti $s1,$s2,20	if ($s2 < 20) $s1 = 1; else $s1 = 0
	set less than immediate unsigned	sltiu $s1,$s2,20	if ($s2 < 20) $s1 = 1; else $s1 = 0

Mips: Conjunto de instruções

A Linguagem dos Computadores

Nome do registrador	Número	Uso
$zero	0	constante 0
$at	1	reservado para o montador
$v0	2	avaliação de expressão e resultados de uma função
$v1	3	avaliação de expressão e resultados de uma função
$a0	4	argumento 1
$a1	5	argumento 2
$a2	6	argumento 3
$a3	7	argumento 4
$t0	8	temporário (não preservado pela chamada)
$t1	9	temporário (não preservado pela chamada)
$t2	10	temporário (não preservado pela chamada)
$t3	11	temporário (não preservado pela chamada)
$t4	12	temporário (não preservado pela chamada)
$t5	13	temporário (não preservado pela chamada)
$t6	14	temporário (não preservado pela chamada)
$t7	15	temporário (não preservado pela chamada)
$s0	16	temporário salvo (preservado pela chamada)
$s1	17	temporário salvo (preservado pela chamada)
$s2	18	temporário salvo (preservado pela chamada)
$s3	19	temporário salvo (preservado pela chamada)
$s4	20	temporário salvo (preservado pela chamada)
$s5	21	temporário salvo (preservado pela chamada)
$s6	22	temporário salvo (preservado pela chamada)
$s7	23	temporário salvo (preservado pela chamada)
$t8	24	temporário (não preservado pela chamada)
$t9	25	temporário (não preservado pela chamada)
$k0	26	reservado para o kernel do sistema operacional
$k1	27	reservado para o kernel do sistema operacional
$gp	28	ponteiro para área global
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	endereço de retorno (usado por chamada de função)

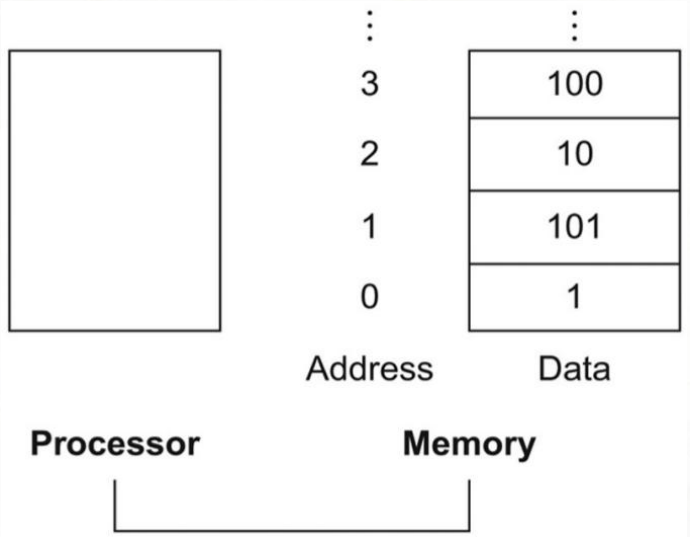
Conjunto de Registradores do MIPS

Exercícios Introdução: Instruções Aritméticas

Faça os exercícios disponibilizados no moodle

A Linguagem dos Computadores:

Operações Load & Store



Mips: Endereços de memória

Vetor A[] = {1, 101, 10, 100 …}

Supondo que o reg S0 contenha a posição inicial de A. Vamos carregar em $t0 o valor 10.

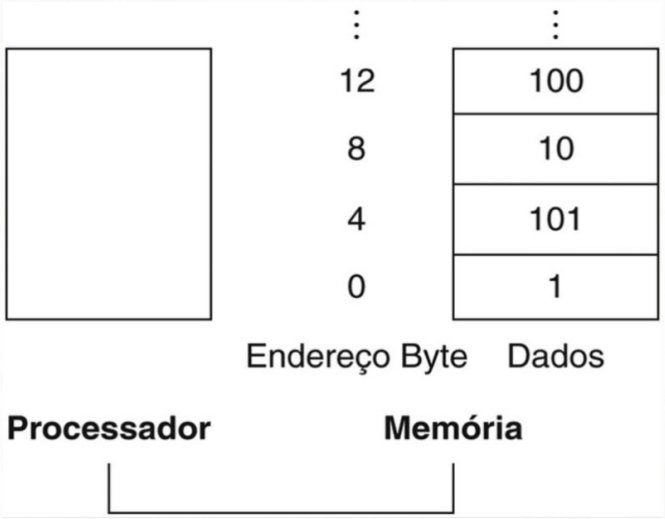
```
lw $t0, 2($s0)

addi $t0, $t0, 92

sw  $t0, 1($s0)
```

A Linguagem dos Computadores:

Operações Load & Store



Mips: Endereços reais

Vetor A[] = {1, 101, 10, 100 …}

Supondo que o reg S0 contenha a posição inicial de A.
Vamos carregar em $t0 o valor 10.

```
lw $t0, 8($s0)

addi $t0, $t0, 92

sw  $t0, 4($s0)
```

Exercícios de Introdução: Instruções Load/store

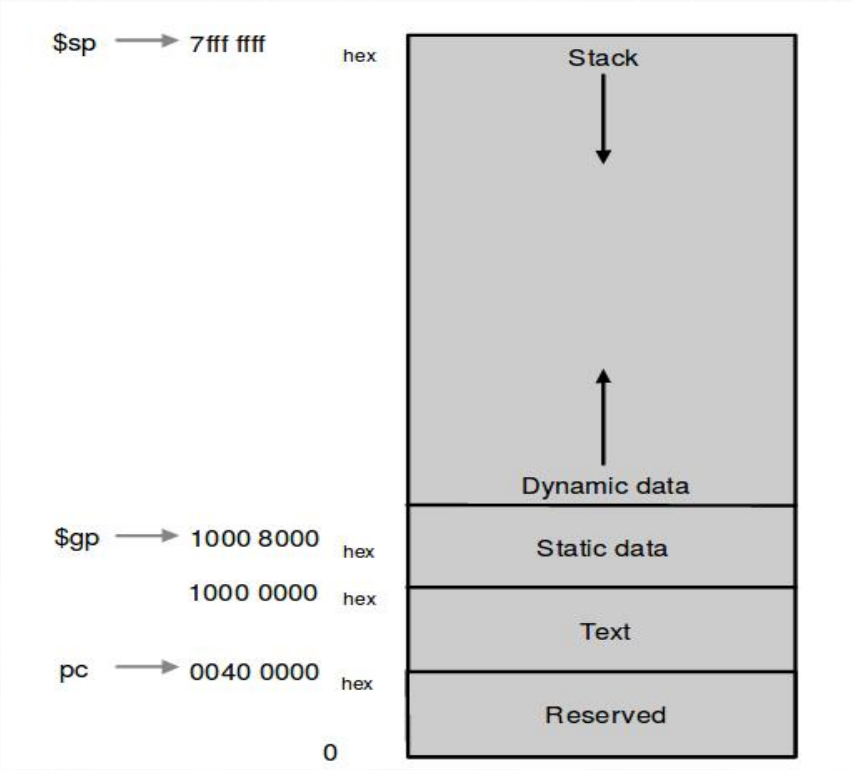
Faça os exercícios disponibilizados no moodle

A Linguagem dos Computadores

Serviço	Código de chamada ao sistema	Argumentos	Resultado
print_int	1	$a0 = integer	
print_float	2	$f12 = float	
print_double	3	$f12 = double	
print_string	4	$a0 = string	
read_int	5		integer (in $v0)
read_float	6		float (in $f0)
read_double	7		double (in $f0)
read_string	8	$a0 = buffer, $a1 = length	
sbrk	9	$a0 = amount	endereço (em $v0)
exit	10		
print_char	11	$a0 = char	
read_char	12		char (in $v0)
open	13	$a0 = nome de arquivo (string), $a1 = flags, $a2 = modo	descritor de arquivo (em $a0)
read	14	$a0 = descritor de arquivo, $a1 = buffer, $a2 = tamanho	número de caracteres lidos (em $a0)
write	15	$a0 = descritor de arquivo, $a1 = buffer, $a2 = tamanho	número de caracteres escritos (em $a0)
close	16	$a0 = descritor de arquivo	
exit2	17	$a0 = resultado	

Mips: Serviços de sistema

A Linguagem dos Computadores

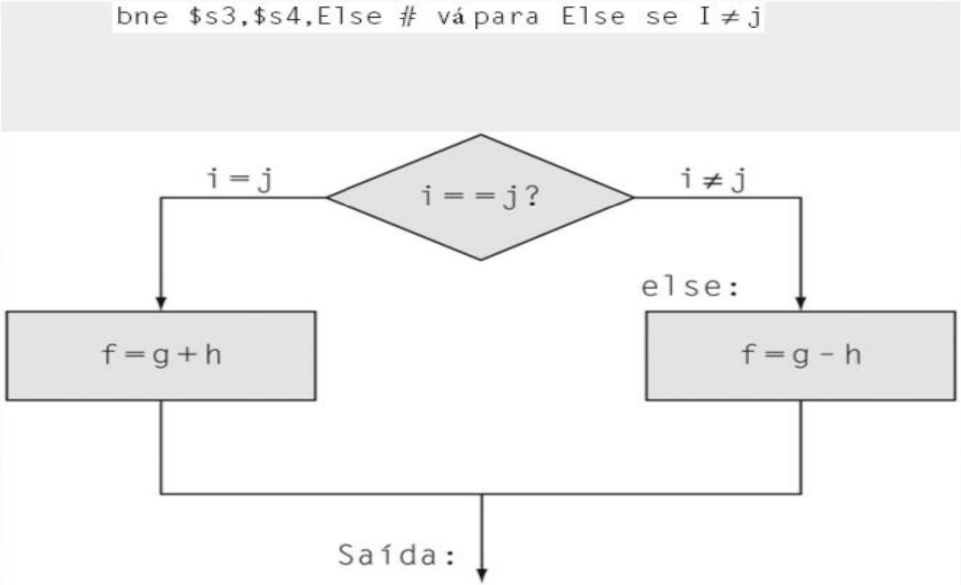


Exemplo: Pilha de execução

Exercícios de Introdução: Syscalls

Faça os exercícios disponibilizados no moodle

Instruções de Controle



Exemplo: Instrução Condicional

Instruções de Controle

#Considerando: **$s3 = I e $s4 = j**

```
bne  $s3, $s4, Else
    add f, g, h


Else:
    sub f, g, h
```

Instruções de Controle

Exemplo:

#Considerando:

$s3 = i

$s5 = k

$s6 = save

```
while(save[i] == k)
    I +=1;
```

Como você implementaria em assembly? (mips)

Instruções de Controle

#Considerando:

$s3 = i

$s5 = k

$s6 = save

```
Loop:
    sll  $t0, $s3, 2
    add $t0, $t0, $s6
```

Instruções de Controle

#Considerando:

$s3 = i

$s5 = k

$s6 = save

```
Loop:
    sll  $t0, $s3, 2
    add $t0, $t0, $s6
    lw   $t1, 0($t0)
```

Instruções de Controle

```
#Considerando:
        $s3 = i
        $s5 = k
        $s6 = save
Loop:
    sll  $t0, $s3, 2
    add $t0, $t0, $s6
    lw   $t1, 0($t0)
    bne $t1, $s5, Sair


Sair:
```

Instruções de Controle

#Considerando:

$s3 = i

$s5 = k

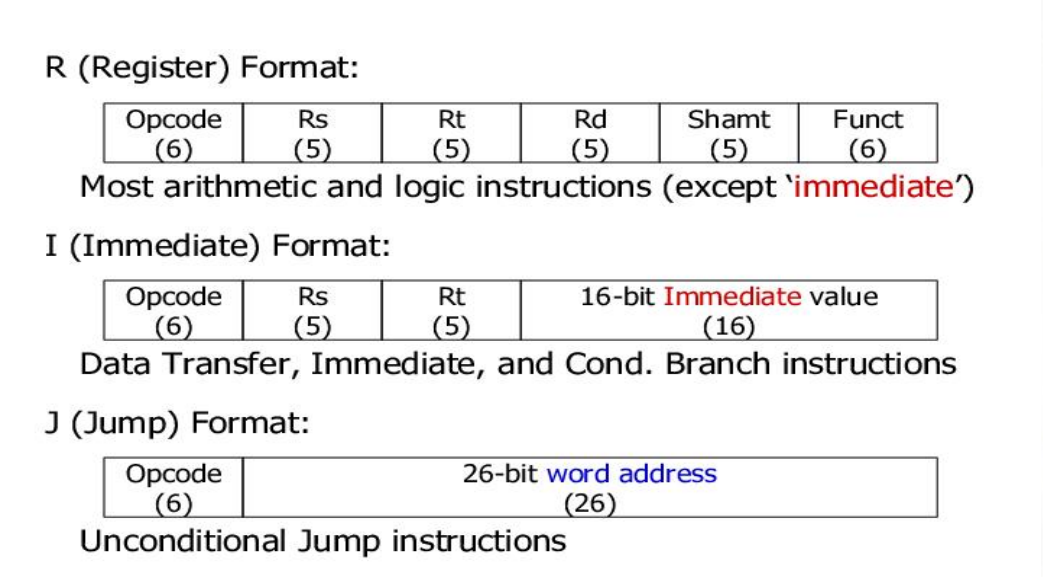
$s6 = save

```
Loop:
    sll  $t0, $s3, 2
    add $t0, $t0, $s6
    lw   $t1, 0($t0)
    bne $t1, $s5, Sair
    addi $s3, $s3, 1
    J  Loop
Sair:
```

Exercícios de Introdução: Instruções de Desvio e Saltos

Faça os exercícios disponibilizados no moodle

A Linguagem dos Computadores



Mips: Formatos de instruções (fonte Yul Chu*)

* https://www.researchgate.net/figure/Instruction-formats-for-MIPS-architecture-1_fig2_228942202

A Linguagem dos Computadores

Load and Store Instructions				
100000	base	dest	signed offset	LB rt, offset(rs)
100001	base	dest	signed offset	LH rt, offset(rs)
100011	base	dest	signed offset	LW rt, offset(rs)
100100	base	dest	signed offset	LBU rt, offset(rs)
100101	base	dest	signed offset	LHU rt, offset(rs)
101000	base	dest	signed offset	SB rt, offset(rs)
101001	base	dest	signed offset	SH rt, offset(rs)
101011	base	dest	signed offset	SW rt, offset(rs)

Mips: Opcodes (FonteBerkeley*)

*https://inst.eecs.berkeley.edu/~cs150/sp11/checkpoint_1/

A Linguagem dos Computadores

I-Type Computational Instructions				
001001	src	dest	signed immediate	ADDIU rt, rs, signed-imm.
001010	src	dest	signed immediate	SLTI rt, rs, signed-imm.
001011	src	dest	signed immediate	SLTIU rt, rs, signed-imm.
001100	src	dest	zero-ext. immediate	ANDI rt, rs, zero-ext-imm.
001101	src	dest	zero-ext. immediate	ORI rt, rs, zero-ext-imm.
001110	src	dest	zero-ext. immediate	XORI rt, rs, zero-ext-imm.
001111	00000	dest	zero-ext. immediate	LUI rt, zero-ext-imm.

Mips: Opcodes (FonteBerkeley*)

*https://inst.eecs.berkeley.edu/~cs150/sp11/checkpoint_1/

A Linguagem dos Computadores

R-Type Computational Instructions

000000	00000	src	dest	shamt	000000	SLL rd, rt, shamt
000000	00000	src	dest	shamt	000010	SRL rd, rt, shamt
000000	00000	src	dest	shamt	000011	SRA rd, rt, shamt
000000	rshamt	src	dest	00000	000100	SLLV rd, rt, rs
000000	rshamt	src	dest	00000	000110	SRLV rd, rt, rs
000000	rshamt	src	dest	00000	000111	SRAV rd, rt, rs
000000	src1	src2	dest	00000	100001	ADDU rd, rs, rt
000000	src1	src2	dest	00000	100011	SUBU rd, rs, rt
000000	src1	src2	dest	00000	100100	AND rd, rs, rt
000000	src1	src2	dest	00000	100101	OR rd, rs, rt
000000	src1	src2	dest	00000	100110	XOR rd, rs, rt
000000	src1	src2	dest	00000	100111	NOR rd, rs, rt
000000	src1	src2	dest	00000	101010	SLT rd, rs, rt
000000	src1	src2	dest	00000	101011	SLTU rd, rs, rt

Mips: Opcodes (Fonte Berkeley*)

*https://inst.eecs.berkeley.edu/~cs150/sp11/checkpoint_1/

A Linguagem dos Computadores

Jump and Branch Instructions						
000010	target					J target
000011	target					JAL target
000000	src	00000	00000	00000	001000	JR rs
000000	src	00000	dest	00000	001001	JALR rd, rs
000100	src1	src2	signed offset			BEQ rs, rt, offset
000101	src1	src2	signed offset			BNE rs, rt, offset
000110	src	00000	signed offset			BLEZ rs, offset
000111	src	00000	signed offset			BGTZ rs, offset
000001	src	00000	signed offset			BLTZ rs, offset
000001	src	00001	signed offset			BGEZ rs, offset

Mips: Opcodes (FonteBerkeley*)

*https://inst.eecs.berkeley.edu/~cs150/sp11/checkpoint_1/

A Linguagem dos Computadores

Exemplo:

Como a instrução subu **$s0, $s1, $s2** ficaria
Em binário?

Opcode:
0

A Linguagem dos Computadores

Exemplo:

Como a instrução subu **$s0, $s1, $s2** ficaria
Em binário?

Opcode:
0 17 18 16

A Linguagem dos Computadores

Exemplo:

Como a instrução subu **$s0, $s1, $s2** ficaria
Em binário?

Opcode:

0 17 18 16 0 35$_{(dec)}$

A Linguagem dos Computadores

Exemplo:

Como a instrução subu **$s0, $s1, $s2** ficaria
Em binário?

Opcode:

0 17 18 16 0 $35_{(dec)}$

000000 10001 10010 10000 00000 $100011_{(bin)}$

Exercícios de Introdução: Tradução

Faça os exercícios disponibilizados no moodle

Referências

- PATTERSON, David A.; HENNESSY, John L.Organizac˜¸ao e projeto de computadores: ainterface hardware/software, 5. ed. Rio de Janeiro: Elsevier, 2017. ISBN 9788535287936.

- STALLINGS, William.; Arquitetura e Organização de Computadores. 10. ed. São Paulo:Pearson Education do Brasil, 2017. ISBN 9788543020532.

- TANENBAUM, A. S.; AUSTIN, T. Organização Estruturada de Computadores. 6. ed. SãoPaulo: Pearson Education do Brasil, 2013. ISBN 9788581435398.