



Universidade Tecnológica Federal do Paraná – UTFPR  
Bacharelado em Ciência da Computação

BCC33B – Arquitetura e Organização de Computadores

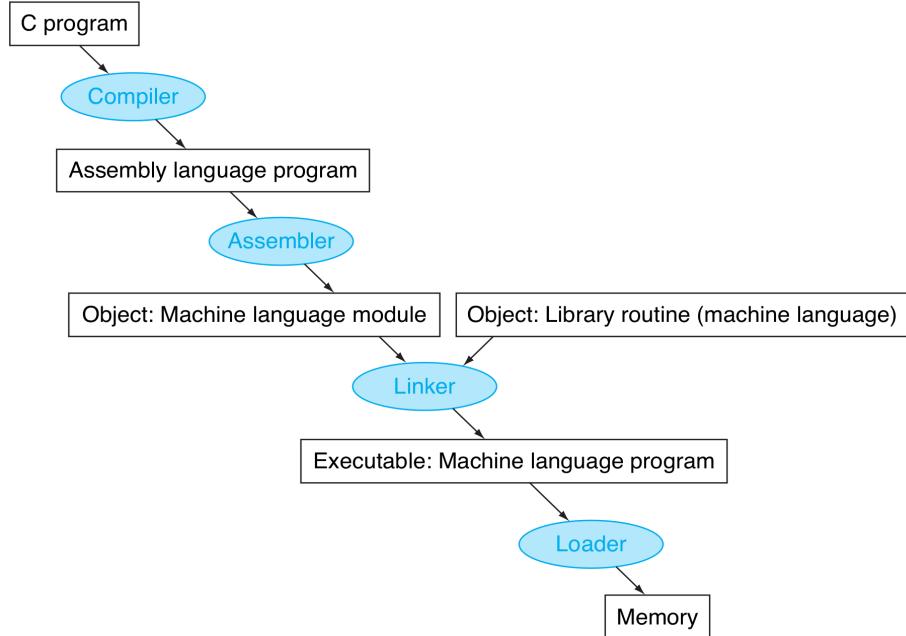
Prof. Rogério A. Gonçalves

*rogerioag@utfpr.edu.br*

## Aula 012

**Codificação do Assembly do MIPS**

# Processo de Compilação



## A tradução

- Quando da tradução de C para assembly deve-se fazer:
  - alokar registradores para as variáveis do programa
  - produzir código para o corpo do procedimento
  - preservar os registradores durante a chamada do procedimento

# Programa em Linguagem C

```
#include <stdio.h>
int x;      // Variáveis Globais
int y;
int z;

int somar(int a, int b) {
    return (a + b);
}

void main (void) {
    printf ("Digite um número : ");
    scanf ("%d", &x);
    printf ("\nDigite um número : ");
    scanf ("%d", &y);
    z = somar (x,y);
    printf ("Soma dos números: %d\n", z);

    return 0;
}
```

06/04/2017

BCC33B – Arquitetura e Organização de Computadores



5

## Estrutura de um Programa em Assembly do MIPS

```
.globl main
.data
# int x
x:    .word 0
# int y
y:    .word 0
# int z
z:    .word 0

str_in:   .asciiz "Digite um número: "
str_out:  .asciiz "Soma dos números: "

.text
j main

# função de soma.
somar: nop
        add $v0, $a0, $a1 # $v0 = a + b.
        jr $ra             # retorna.
```

06/04/2017

BCC33B – Arquitetura e Organização de Computadores



6

# Estrutura de um Programa em Assembly do MIPS

```
# programa principal.
main: addi $sp, $sp, -4    # stack frame
      sw   $ra, 0($sp)    # store $ra
      # Imprimir a string solicitacao numero.
      # print_string é system call 4
      li $v0, 4
      # load no endereço da string a ser impressa.
      la $a0, str_in
      syscall
      # Leitura de um inteiro.
      # read_int é system call 5
      li $v0, 5
      syscall
      # read_int retorna um valor, o resultado é colocado em $v0.
      # copia o valor lido para x.
      sw $v0, x
      # imprime a quebra de linha '\n'.
      # print_char é syscall 11 e '\n' is ASCII 10.
      li $v0, 11
      li $a0, 10
      syscall
```

06/04/2017

BCC33B – Arquitetura e Organização de Computadores

UNIVERSIDADE TECNOLÓGICA DE MARília

7

# Estrutura de um Programa em Assembly do MIPS

```
# Imprime solicitação de outro número.
li $v0, 4
# load no endereço da string a ser impressa.
la $a0, str_in
syscall

# leitura de y.
li $v0, 5
syscall
sw $v0, y

# imprime a quebra de linha '\n'.
# print_char é syscall 11 e '\n' is ASCII 10.
li $v0, 11
li $a0, 10
syscall

lw  $a0, x      # carrega x como argumento.
lw  $a1, y      # carrega y como argumento.
jal somar      # call somar
sw  $v0, z      # z = somar(x,y)
```

06/04/2017

BCC33B – Arquitetura e Organização de Computadores

8

# Estrutura de um Programa em Assembly do MIPS

```
# imprimir a string do resultado.  
# print_str system call 4  
li $v0, 4  
la $a0, str_out  
syscall  
  
# carrega o valor de z da memória para um registrador.  
lw $t0, z  
add $a0, $0, $t0  
# $a0 agora contem o valor de z a ser impresso.  
li $v0, 1  
syscall  
  
# now print the '\n'  
# print_char is system call 11 and '\n' is ASCII 10  
li $v0, 11  
li $a0, 10  
syscall  
  
lw $ra, 0($sp)    # restaura $ra.  
addi $sp, $sp, 4    # restaura $sp.  
jr $ra             # retorna para o S0.
```

06/04/2017

BCC33B – Arquitetura e Organização de Computadores

10

## Formato de instruções

Name	Fields						Comments
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions 32 bits
R-format	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	op	rs	rt	address/immediate			Transfer, branch, imm. format
J-format	op	target address					Jump instruction format

**FIGURE 3.19 MIPS instruction formats in Chapter 3.** Highlighted portions show instruction formats introduced in this section.

# Codificação das instruções do MIPS

op(31:26)								
28-26 31-29	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	R-format	<u>Bltz/gez</u>	jump	jump & link	branch eq	branch ne	blez	bgtz
1(001)	add immediate	addiu	set less than imm.	sltiu	andi	ori	xori	load upper imm
2(010)	TLB	<u>F1Pt</u>						
3(011)								
4(100)	load byte	lh	lw1	load word	lbu	lhu	lw1	
5(101)	store byte	sh	sw1	store word			swr	
6(110)	lwc0	<u>lwc1</u>						
7(111)	swc0	<u>swc1</u>						

op(31:26)=010000 (TLB), rs(25:21)								
23-21 25-24	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(00)	mfc0		cfc0		mtc0		ctc0	
1(01)								
2(10)								
3(11)								

06/04/2017

BCC33B – Arquitetura e Organização de Computadores

# Codificação das instruções do MIPS

op(31:26)=000000 (R-format), funct(5:0)								
2-0 5-3	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	sll		srl	sra	sllv		srlv	sra
1(001)	jump reg.	jalr			syscall	break		
2(010)	mfhi	mtfi	mflo	mtlo				
3(011)	mult	multu	div	divu				
4(100)	add	addu	subtract	subu	and	or	xor	nor
5(101)			set l.t.	situ				
6(110)								
7(111)								

**FIGURE 3.18 MIPS instruction encoding.** This notation gives the value of a field by row and by column. For example, in the top portion of the figure load word is found in row number 4 (100<sub>two</sub> for bits 31–29 of the instruction) and column number 3 (011<sub>two</sub> for bits 28–26 of the instruction), so the corresponding value of the op field (bits 31–26) is 100011<sub>two</sub>. Underscore means the field is used elsewhere. For example, R-format in row 0 and column 0 (op = 000000<sub>two</sub>) is defined in the bottom part of the figure. Hence subtract in row 4 and column 2 of the bottom section means that the funct field (bits 5–0) of the instruction is 100010<sub>two</sub> and the op field (bits 31–26) is 000000<sub>two</sub>. The F1Pt value in row 2, column 1 is defined in Figure 4.48 on page 292 in Chapter 4. Bltz/gez is the opcode for four instructions found in Appendix A: bltz, bgez, bltzal, and bgezal. Instructions given in full name using color are described in Chapter 3, while instructions given in mnemonics using color are described in Chapter 4. Appendix A covers all instructions.

06/04/2017

BCC33B – Arquitetura e Organização de Computadores

# Instruções do MIPS

MIPS assembly language				
Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add \$s1,\$s2,\$s3	$s1 = s2 + s3$	Three operands; data in registers
	subtract	sub \$s1,\$s2,\$s3	$s1 = s2 - s3$	Three operands; data in registers
	add immediate	addi \$s1,\$s2,100	$s1 = s2 + 100$	Used to add constants
Data transfer	load word	lw \$s1,100(\$s2)	$s1 = \text{Memory}[s2 + 100]$	Word from memory to register
	store word	sw \$s1,100(\$s2)	$\text{Memory}[s2 + 100] = s1$	Word from register to memory
	load byte	lb \$s1,100(\$s2)	$s1 = \text{Memory}[s2 + 100]$	Byte from memory to register
	store byte	sb \$s1,100(\$s2)	$\text{Memory}[s2 + 100] = s1$	Byte from register to memory
Conditional branch	load upper immediate	lui \$s1,100	$s1 = 100 * 2^{16}$	Loads constant in upper 16 bits
	branch on equal	beq \$s1,\$s2,25	if ( $s1 == s2$ ) go to PC + 4 + 100	Equal test; PC-relative branch
	branch on not equal	bne \$s1,\$s2,25	if ( $s1 != s2$ ) go to PC + 4 + 100	Not equal test; PC-relative
Unconditional jump	set on less than	slt \$s1,\$s2,\$s3	if ( $s2 < s3$ ) $s1 = 1$ ; else $s1 = 0$	Compare less than; for beq, bne
	set less than immediate	slti \$s1,\$s2,100	if ( $s2 < 100$ ) $s1 = 1$ ; else $s1 = 0$	Compare less than constant
	jump	j 2500	go to 10000	Jump to target address
Conditional jump	jump register	jr \$ra	go to \$ra	For switch, procedure return
	jump and link	jal 2500	$$ra = PC + 4$ ; go to 10000	For procedure call

FIGURE 3.20 MIPS assembly language revealed in Chapter 3. Highlighted portions show portions from sections 3.7 and 3.8.

06/04/2017

BCC33B – Arquitetura e Organização de Computadores



13

# Instruções do MIPS

Operandos MIPS		
Nome	Exemplo	Comentários
32 registradores	\$s0-\$s7, \$t0-\$t9, \$zero, \$a0-\$a3, \$v0-\$v1, \$gp, \$fp, \$sp, \$ra	Locais rápidos para dados. No MIPS, os dados precisam estar em registradores para a realização de operações aritméticas. O registrador MIPS \$zero sempre é igual a 0. \$gp (28) é o ponteiro global. \$sp (29) é o stack pointer. \$fp é o frame pointer, e \$ra (31) é o endereço de retorno.
2 <sup>30</sup> words na memória	Memória[0], Memória[4]...., Memória[4294967292]	Acessadas apenas por instruções de transferência de dados no MIPS. O MIPS utiliza endereços em bytes, de modo que os endereços em words sequenciais diferem em 4 vezes. A memória contém estruturas de dados, arrays e spilled registers, como aqueles salvos nas chamadas de procedimento.

Assembly do MIPS				
Categoria	Instrução	Exemplo	Significado	Comentários
Aritmética	add	add \$s1,\$s2,\$s3	$s1 = s2 + s3$	Três operandos; dados nos registradores
	subtract	sub \$s1,\$s2,\$s3	$s1 = s2 - s3$	Três operandos; dados nos registradores
Transferência de dados	load word	lw \$s1,100(\$s2)	$s1 = \text{Memória}[s2 + 100]$	Dados da memória para o registrador
	store word	sw \$s1,100(\$s2)	$\text{Memória}[s2 + 100] = s1$	Dados do registrador para a memória
Lógica	and	and \$s1,\$s2,\$s3	$s1 = s2 \& s3$	Três operadores em registrador; AND bit a bit
	or	or \$s1,\$s2,\$s3	$s1 = s2   s3$	Três operadores em registrador; OR bit a bit
	nor	nor \$s1,\$s2,\$s3	$s1 = \neg(s2   s3)$	Três operadores em registrador; NOR bit a bit
	and immediate	andi \$s1,\$s2,100	$s1 = s2 \& 100$	AND bit a bit registrador com constante
	or immediate	ori \$s1,\$s2,100	$s1 = s2   100$	OR bit a bit registrador com constante
Desvio condicional	shift left logical	sll \$s1,\$s2,10	$s1 = s2 << 10$	Deslocamento à esquerda por constante
	shift right logical	srl \$s1,\$s2,10	$s1 = s2 >> 10$	Deslocamento à direita por constante
	branch on equal	beq \$s1,\$s2,L	if ( $s1 == s2$ ) go to L	Testa igualdade e desvia
	branch on not equal	bne \$s1,\$s2,L	if ( $s1 != s2$ ) go to L	Testa desigualdade e desvia
	set on less than	slt \$s1,\$s2,\$s3	if ( $s2 < s3$ ) $s1 = 1$ ; else $s1 = 0$	Compara menor que; usado com beq, bne
Desvio incondicional	set on less than immediate	slt \$s1,\$s2,100	if ( $s2 < 100$ ) $s1 = 1$ ; else $s1 = 0$	Compara menor que imediato; usado com beq, bne
	jump	j L	go to L	Desvia para endereço de destino
	jump register	jr \$ra	go to \$ra	Para retorno de procedimento
	jump and link	jal L	$$ra = PC + 4$ , go to L	Para chamada de procedimento

06/04/2017

BCC33B – Arquitetura e Organização de Computadores



14

Linguagem de máquina do MIPS

Nome	Formato	Exemplo						Comentários
add	R	0	18	19	17	0	32	add \$s1,\$s2,\$s3
sub	R	0	18	19	17	0	34	sub \$s1,\$s2,\$s3
lw	I	35	18	17	100			lw \$s1,100(\$s2)
sw	I	43	18	17	100			sw \$s1,100(\$s2)
and	R	0	18	19	17	0	36	and \$s1,\$s2,\$s3
or	R	0	18	19	17	0	37	or \$s1,\$s2,\$s3
nor	R	0	18	19	17	0	39	nor \$s1,\$s2,\$s3
andi	I	12	18	17	100			andi \$s1,\$s2,100
ori	I	13	18	17	100			ori \$s1,\$s2,100
sll	R	0	0	18	17	10	0	sll \$s1,\$s2,10
srl	R	0	0	18	17	10	2	srl \$s1,\$s2,10
beq	I	4	17	18	25			beq \$s1,\$s2,100
bne	I	5	17	18	25			bne \$s1,\$s2,100
slt	R	0	18	19	17	0	42	slt \$s1,\$s2,\$s3
j	J	2	2500					j 10000 (ver Seção 2.9)
jr	R	0	31	0	0	0	8	jr \$ra
jal	J	3	2500					jal 10000 (ver Seção 2.9)
Tamanho do campo		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	Todas as instruções MIPS de 32 bits
Formato R	R	op	rs	rt	rd	shamt	funct	Formato das instruções aritméticas
Formato I	I	op	rs	rt	endereço			Formato para transferências de dados e desvios

**FIGURA 2.20 A linguagem de máquina do MIPS revelada até a Seção 2.7.** As partes destacadas mostram as estruturas em assembly do MIPS introduzidas na Seção 2.7. O formato J, usado para instruções de jump e jump-and-link, é explicado na Seção 2.9, que também explica por que colocar 25 no campo de endereço das instruções em linguagem de máquina beq e bne é equivalente a 100 em assembly.

Usar o arquivo: mars-opcodes.pdf

Computadores



15

## Resumo da Aula de Hoje

### Tópicos mais importantes:

Linguagem Assembly

Microprocessador MIPS

- Codificação do Assembly



Universidade Tecnológica Federal do Paraná – UTFPR  
Bacharelado em Ciência da Computação

BCC33B – Arquitetura e Organização de Computadores

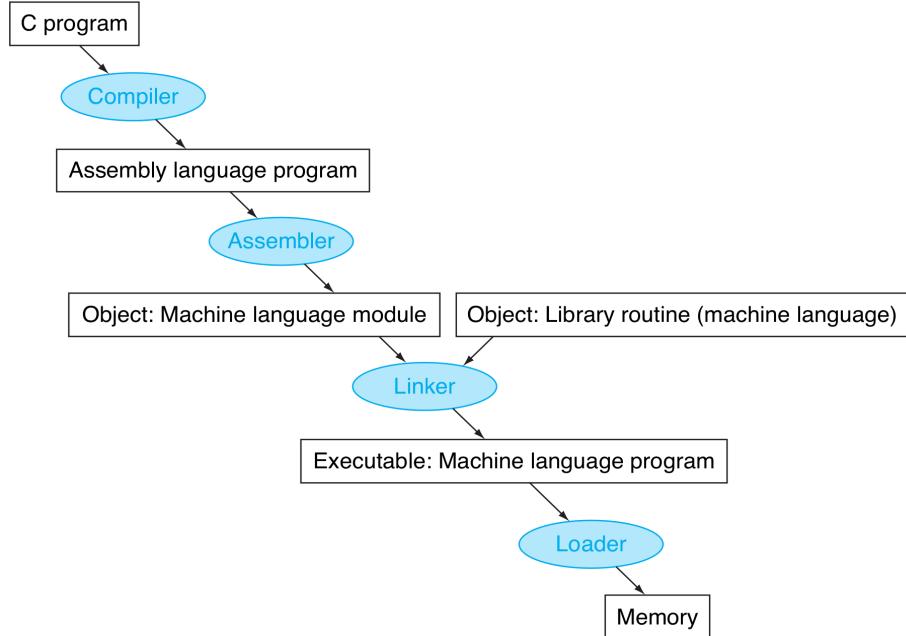
Prof. Rogério A. Gonçalves

*rogerioag@utfpr.edu.br*

## Aula 012

**Codificação do Assembly do MIPS**

# Processo de Compilação



## A tradução

- Quando da tradução de C para assembly deve-se fazer:
  - alokar registradores para as variáveis do programa
  - produzir código para o corpo do procedimento
  - preservar os registradores durante a chamada do procedimento

# Programa em Linguagem C

```
#include <stdio.h>
int x;      // Variáveis Globais
int y;
int z;

int somar(int a, int b) {
    return (a + b);
}

void main (void) {
    printf ("Digite um número : ");
    scanf ("%d", &x);
    printf ("\nDigite um número : ");
    scanf ("%d", &y);
    z = somar (x,y);
    printf ("Soma dos números: %d\n", z);

    return 0;
}
```

06/04/2017

BCC33B – Arquitetura e Organização de Computadores



5

## Estrutura de um Programa em Assembly do MIPS

```
.globl main
.data
# int x
x:   .word 0
# int y
y:   .word 0
# int z
z:   .word 0

str_in:  .asciiz "Digite um número: "
str_out: .asciiz "Soma dos números: "

.text
j main

# função de soma.
somar: nop
        add $v0, $a0, $a1 # $v0 = a + b.
        jr $ra             # retorna.
```

06/04/2017

BCC33B – Arquitetura e Organização de Computadores



6

# Estrutura de um Programa em Assembly do MIPS

```
# programa principal.
main: addi $sp, $sp, -4    # stack frame
      sw   $ra, 0($sp)    # store $ra
      # Imprimir a string solicitacao numero.
      # print_string é system call 4
      li $v0, 4
      # load no endereço da string a ser impressa.
      la $a0, str_in
      syscall
      # Leitura de um inteiro.
      # read_int é system call 5
      li $v0, 5
      syscall
      # read_int retorna um valor, o resultado é colocado em $v0.
      # copia o valor lido para x.
      sw $v0, x
      # imprime a quebra de linha '\n'.
      # print_char é syscall 11 e '\n' is ASCII 10.
      li $v0, 11
      li $a0, 10
      syscall
```

06/04/2017

BCC33B – Arquitetura e Organização de Computadores

UNIVERSIDADE TECNOLÓGICA DE MARília

7

# Estrutura de um Programa em Assembly do MIPS

```
# Imprime solicitação de outro número.
li $v0, 4
# load no endereço da string a ser impressa.
la $a0, str_in
syscall

# leitura de y.
li $v0, 5
syscall
sw $v0, y

# imprime a quebra de linha '\n'.
# print_char é syscall 11 e '\n' is ASCII 10.
li $v0, 11
li $a0, 10
syscall

lw  $a0, x      # carrega x como argumento.
lw  $a1, y      # carrega y como argumento.
jal somar      # call somar
sw  $v0, z      # z = somar(x,y)
```

06/04/2017

BCC33B – Arquitetura e Organização de Computadores

8

# Estrutura de um Programa em Assembly do MIPS

```
# imprimir a string do resultado.  
# print_str system call 4  
li $v0, 4  
la $a0, str_out  
syscall  
  
# carrega o valor de z da memória para um registrador.  
lw $t0, z  
add $a0, $0, $t0  
# $a0 agora contem o valor de z a ser impresso.  
li $v0, 1  
syscall  
  
# now print the '\n'  
# print_char is system call 11 and '\n' is ASCII 10  
li $v0, 11  
li $a0, 10  
syscall  
  
lw $ra, 0($sp)    # restaura $ra.  
addi $sp, $sp, 4    # restaura $sp.  
jr $ra             # retorna para o S0.
```

06/04/2017

BCC33B – Arquitetura e Organização de Computadores

10

## Formato de instruções

Name	Fields						Comments
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions 32 bits
R-format	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	op	rs	rt	address/immediate			Transfer, branch, imm. format
J-format	op	target address					Jump instruction format

**FIGURE 3.19 MIPS instruction formats in Chapter 3.** Highlighted portions show instruction formats introduced in this section.

# Codificação das instruções do MIPS

op(31:26)								
28-26 31-29	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	R-format	<u>Bltz/gez</u>	jump	jump & link	branch eq	branch ne	blez	bgtz
1(001)	add immediate	addiu	set less than imm.	sltiu	andi	ori	xori	load upper imm
2(010)	TLB	<u>F1Pt</u>						
3(011)								
4(100)	load byte	lh	lw1	load word	lbu	lhu	lw1	
5(101)	store byte	sh	sw1	store word			swr	
6(110)	lwc0	<u>lwc1</u>						
7(111)	swc0	<u>swc1</u>						

op(31:26)=010000 (TLB), rs(25:21)								
23-21 25-24	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(00)	mfc0		cfc0		mtc0		ctc0	
1(01)								
2(10)								
3(11)								

06/04/2017

BCC33B – Arquitetura e Organização de Computadores

# Codificação das instruções do MIPS

op(31:26)=000000 (R-format), funct(5:0)								
2-0 5-3	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	sll		srl	sra	sllv		srlv	sra
1(001)	jump reg.	jalr			syscall	break		
2(010)	mfhi	mtfi	mflo	mtlo				
3(011)	mult	multu	div	divu				
4(100)	add	addu	subtract	subu	and	or	xor	nor
5(101)			set l.t.	situ				
6(110)								
7(111)								

**FIGURE 3.18 MIPS instruction encoding.** This notation gives the value of a field by row and by column. For example, in the top portion of the figure load word is found in row number 4 (100<sub>two</sub> for bits 31–29 of the instruction) and column number 3 (011<sub>two</sub> for bits 28–26 of the instruction), so the corresponding value of the op field (bits 31–26) is 100011<sub>two</sub>. Underscore means the field is used elsewhere. For example, R-format in row 0 and column 0 (op = 000000<sub>two</sub>) is defined in the bottom part of the figure. Hence subtract in row 4 and column 2 of the bottom section means that the funct field (bits 5–0) of the instruction is 100010<sub>two</sub> and the op field (bits 31–26) is 000000<sub>two</sub>. The F1Pt value in row 2, column 1 is defined in Figure 4.48 on page 292 in Chapter 4. Bltz/gez is the opcode for four instructions found in Appendix A: bltz, bgez, bltzal, and bgezal. Instructions given in full name using color are described in Chapter 3, while instructions given in mnemonics using color are described in Chapter 4. Appendix A covers all instructions.

06/04/2017

BCC33B – Arquitetura e Organização de Computadores

# Instruções do MIPS

MIPS assembly language				
Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add \$s1,\$s2,\$s3	$s1 = s2 + s3$	Three operands; data in registers
	subtract	sub \$s1,\$s2,\$s3	$s1 = s2 - s3$	Three operands; data in registers
	add immediate	addi \$s1,\$s2,100	$s1 = s2 + 100$	Used to add constants
Data transfer	load word	lw \$s1,100(\$s2)	$s1 = \text{Memory}[s2 + 100]$	Word from memory to register
	store word	sw \$s1,100(\$s2)	$\text{Memory}[s2 + 100] = s1$	Word from register to memory
	load byte	lb \$s1,100(\$s2)	$s1 = \text{Memory}[s2 + 100]$	Byte from memory to register
	store byte	sb \$s1,100(\$s2)	$\text{Memory}[s2 + 100] = s1$	Byte from register to memory
Conditional branch	load upper immediate	lui \$s1,100	$s1 = 100 * 2^{16}$	Loads constant in upper 16 bits
	branch on equal	beq \$s1,\$s2,25	if ( $s1 == s2$ ) go to PC + 4 + 100	Equal test; PC-relative branch
	branch on not equal	bne \$s1,\$s2,25	if ( $s1 != s2$ ) go to PC + 4 + 100	Not equal test; PC-relative
Unconditional jump	set on less than	slt \$s1,\$s2,\$s3	if ( $s2 < s3$ ) $s1 = 1$ ; else $s1 = 0$	Compare less than; for beq, bne
	set less than immediate	slti \$s1,\$s2,100	if ( $s2 < 100$ ) $s1 = 1$ ; else $s1 = 0$	Compare less than constant
	jump	j 2500	go to 10000	Jump to target address
Conditional jump	jump register	jr \$ra	go to \$ra	For switch, procedure return
	jump and link	jal 2500	$$ra = PC + 4$ ; go to 10000	For procedure call

FIGURE 3.20 MIPS assembly language revealed in Chapter 3. Highlighted portions show portions from sections 3.7 and 3.8.

06/04/2017

BCC33B – Arquitetura e Organização de Computadores



13

# Instruções do MIPS

Operandos MIPS		
Nome	Exemplo	Comentários
32 registradores	\$s0-\$s7, \$t0-\$t9, \$zero, \$a0-\$a3, \$v0-\$v1, \$gp, \$fp, \$sp, \$ra	Locais rápidos para dados. No MIPS, os dados precisam estar em registradores para a realização de operações aritméticas. O registrador MIPS \$zero sempre é igual a 0. \$gp (28) é o ponteiro global. \$sp (29) é o stack pointer. \$fp é o frame pointer, e \$ra (31) é o endereço de retorno.
2 <sup>30</sup> words na memória	Memória[0], Memória[4]...., Memória[4294967292]	Acessadas apenas por instruções de transferência de dados no MIPS. O MIPS utiliza endereços em bytes, de modo que os endereços em words sequenciais diferem em 4 vezes. A memória contém estruturas de dados, arrays e spilled registers, como aqueles salvos nas chamadas de procedimento.

Assembly do MIPS				
Categoria	Instrução	Exemplo	Significado	Comentários
Aritmética	add	add \$s1,\$s2,\$s3	$s1 = s2 + s3$	Três operandos; dados nos registradores
	subtract	sub \$s1,\$s2,\$s3	$s1 = s2 - s3$	Três operandos; dados nos registradores
Transferência de dados	load word	lw \$s1,100(\$s2)	$s1 = \text{Memória}[s2 + 100]$	Dados da memória para o registrador
	store word	sw \$s1,100(\$s2)	$\text{Memória}[s2 + 100] = s1$	Dados do registrador para a memória
Lógica	and	and \$s1,\$s2,\$s3	$s1 = s2 \& s3$	Três operadores em registrador; AND bit a bit
	or	or \$s1,\$s2,\$s3	$s1 = s2   s3$	Três operadores em registrador; OR bit a bit
	nor	nor \$s1,\$s2,\$s3	$s1 = \neg(s2   s3)$	Três operadores em registrador; NOR bit a bit
	and immediate	andi \$s1,\$s2,100	$s1 = s2 \& 100$	AND bit a bit registrador com constante
	or immediate	ori \$s1,\$s2,100	$s1 = s2   100$	OR bit a bit registrador com constante
Desvio condicional	shift left logical	sll \$s1,\$s2,10	$s1 = s2 \ll 10$	Deslocamento à esquerda por constante
	shift right logical	srl \$s1,\$s2,10	$s1 = s2 \gg 10$	Deslocamento à direita por constante
	branch on equal	beq \$s1,\$s2,L	if ( $s1 == s2$ ) go to L	Testa igualdade e desvia
Desvio condicional	branch on not equal	bne \$s1,\$s2,L	if ( $s1 != s2$ ) go to L	Testa desigualdade e desvia
	set on less than	slt \$s1,\$s2,\$s3	if ( $s2 < s3$ ) $s1 = 1$ ; else $s1 = 0$	Compara menor que; usado com beq, bne
	set on less than immediate	slt \$s1,\$s2,100	if ( $s2 < 100$ ) $s1 = 1$ ; else $s1 = 0$	Compara menor que imediato; usado com beq, bne
Desvio incondicional	jump	j L	go to L	Desvia para endereço de destino
	jump register	jr \$ra	go to \$ra	Para retorno de procedimento
	jump and link	jal L	$$ra = PC + 4$ , go to L	Para chamada de procedimento

06/04/2017

BCC33B – Arquitetura e Organização de Computadores



14

Linguagem de máquina do MIPS

Nome	Formato	Exemplo						Comentários
add	R	0	18	19	17	0	32	add \$s1,\$s2,\$s3
sub	R	0	18	19	17	0	34	sub \$s1,\$s2,\$s3
lw	I	35	18	17	100			lw \$s1,100(\$s2)
sw	I	43	18	17	100			sw \$s1,100(\$s2)
and	R	0	18	19	17	0	36	and \$s1,\$s2,\$s3
or	R	0	18	19	17	0	37	or \$s1,\$s2,\$s3
nor	R	0	18	19	17	0	39	nor \$s1,\$s2,\$s3
andi	I	12	18	17	100			andi \$s1,\$s2,100
ori	I	13	18	17	100			ori \$s1,\$s2,100
sll	R	0	0	18	17	10	0	sll \$s1,\$s2,10
srl	R	0	0	18	17	10	2	srl \$s1,\$s2,10
beq	I	4	17	18	25			beq \$s1,\$s2,100
bne	I	5	17	18	25			bne \$s1,\$s2,100
slt	R	0	18	19	17	0	42	slt \$s1,\$s2,\$s3
j	J	2	2500					j 10000 (ver Seção 2.9)
jr	R	0	31	0	0	0	8	jr \$ra
jal	J	3	2500					jal 10000 (ver Seção 2.9)
Tamanho do campo		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	Todas as instruções MIPS de 32 bits
Formato R	R	op	rs	rt	rd	shamt	funct	Formato das instruções aritméticas
Formato I	I	op	rs	rt	endereço			Formato para transferências de dados e desvios

**FIGURA 2.20 A linguagem de máquina do MIPS revelada até a Seção 2.7.** As partes destacadas mostram as estruturas em assembly do MIPS introduzidas na Seção 2.7. O formato J, usado para instruções de jump e jump-and-link, é explicado na Seção 2.9, que também explica por que colocar 25 no campo de endereço das instruções em linguagem de máquina beq e bne é equivalente a 100 em assembly.

Usar o arquivo: mars-opcodes.pdf

Computadores



15

## Resumo da Aula de Hoje

### Tópicos mais importantes:

Linguagem Assembly

Microprocessador MIPS

- Codificação do Assembly