

Instruções da Máquina Nativa				Instruções da Máquina Virtual				DETI-UA - ACI		
Transferência Memória-Registro (<i>Load</i>)		Cálculo c/ Inteiros: Operações Aritméticas		Transferência Memória-Registro (<i>Load</i>)		Salto Relativo (<i>Branch</i>)				
lb	Rdst, addr	add	Rdst, Rsrc1, Rsrc2	l.d	FPdst, addr	b	Label			
lbu	Rdst, addr	addi	Rdst, Rsrc, Imm	l.s	FPdst, addr	beqz	Rsrc, Label			
lw	Rdst, addr	addiu	Rdst, Rsrc, Imm			bnez	Rsrc, Label			
lwcz	CReg, addr	addu	Rdst, Rsrc1, Rsrc2	Transferência Registro-Memória (<i>Store</i>)		bge	Rsrc, Src, Label			
Transferência Registro-Memória (<i>Store</i>)		div	Rsrc1, Rsrc2	s.d	FPsrc, addr	bgeu	Rsrc, Src, Label			
sb	Rsrc, addr	divu	Rsrc1, Rsrc2	s.s	FPsrc, addr	bgt	Rsrc, Src, Label			
sw	Rsrc, addr	mult	Rsrc1, Rsrc2	Transferência Registro-Registro (<i>Move</i>)		bgtu	Rsrc, Src, Label			
swcz	Creg, addr	multu	Rsrc1, Rsrc2	move	Rdst, Rsrc	ble	Rsrc, Src, Label			
		sub	Rdst, Rsrc1, Rsrc2			bleu	Rsrc, Src, Label			
		subu	Rdst, Rsrc1, Rsrc2			blt	Rsrc, Src, Label			
Transferência Registro-Registro (<i>Move</i>)		Cálculo c/ Inteiros: Op. Lógicas Bitwise		Manipulação de Const. (<i>Load Imm/sym</i>)		bltu	Rsrc, Src, Label			
mfhi	Rdst	and	Rdst, Rsrc1, Rsrc2	la	Rdst, sym	beq	Rsrc, Src, Label			
mflo	Rdst	andi	Rdst, Rsrc, Imm	li	Rdst, IMM	bne	Rsrc, Src, Label			
mthi	Rsrc	nor	Rdst, Rsrc1, Rsrc2	l.d	FPdst, sym					
mtlo	Rsrc	or	Rdst, Rsrc1, Rsrc2	l.s	FPdst, sym					
mfcz	Rdst, Creg	ori	Rdst, Rsrc, Imm	Cálculo c/ Inteiros: Op. Aritméticas						
mtcz	Rsrc, Creg	xor	Rdst, Rsrc1, Rsrc2	abs	Rdst, Rsrc					
mov.d	FPdst, FPsrc	xori	Rdst, Rsrc, Imm	div	Rdst, Rsrc, Src					
mov.s	FPdst, FPsrc	Cálculo c/ Inteiros: Operações de Shift		divu	Rdst, Rsrc, Src					
		sll	Rdst, Rsrc1, Imm5	mul	Rdst, Rsrc, Src					
Manipulação de Const. (<i>Load Immediate</i>)		sllv	Rdst, Rsrc1, Rsrc2	mulu	Rdst, Rsrc, Src					
lui	Rdst, Imm	sra	Rdst, Rsrc1, Imm5	mulo	Rdst, Rsrc, Src					
Instruções de Comparação		srav	Rdst, Rsrc1, Rsrc2	mulou	Rdst, Rsrc, Src					
slt	Rdst, Rsrc1, Rsrc2	srl	Rdst, Rsrc1, Imm5	neg	Rdst, Rsrc					
sltu	Rdst, Rsrc1, Rsrc2	srlv	Rdst, Rsrc1, Rsrc2	negu	Rdst, Rsrc					
slti	Rdst, Rsrc, Imm	Cálculo em Vírgula Flutuante		rem	Rdst, Rsrc, Src					
sltiu	Rdst, Rsrc, Imm	abs.p	FPdst, FPsrc	remu	Rdst, Rsrc, Src					
Salto Relativo (<i>Branch</i>) e Absoluto (<i>Jump</i>)		add.p	FPdst, FPsrc1, FPsrc2	Cálculo c/ Inteiros: Op. Lógicas Bitwise						
bczf	Label	c.eq.p	FPsrc1, FPsrc2	not	Rdst, Rsrc					
bczt	Label	c.le.p	FPsrc1, FPsrc2	Cálculo c/ Inteiros: Operações de Rotate						
beq	Rsrc1, Rsrc2, Label	c.lt.p	FPsrc1, FPsrc2	rol	Rdst, Rsrc, Src					
bgez	Rsrc, Label	cvt.d.s	FPdst, FPsrc	ror	Rdst, Rsrc, Src					
bgezal	Rsrc, Label	cvt.d.w	FPdst, FPsrc	Instruções de Comparação						
bgtz	Rsrc, Label	cvt.s.d	FPdst, FPsrc	seq	Rdst, Rsrc, Src					
blez	Rsrc, Label	cvt.s.w	FPdst, FPsrc	sge	Rdst, Rsrc, Src					
bltz	Rsrc, Label	cvt.w.d	FPdst, FPsrc	sgeu	Rdst, Rsrc, Src					
bltzal	Rsrc, Label	cvt.w.s	FPdst, FPsrc	sgt	Rdst, Rsrc, Src					
bne	Rsrc1, Rsrc2, Label	div.p	FPdst, FPsrc1, FPsrc2	sgtu	Rdst, Rsrc, Src					
j	Label	mul.p	FPdst, FPsrc1, FPsrc2	sle	Rdst, Rsrc, Src					
jal	Label	neg.p	FPdst, FPsrc	sleu	Rdst, Rsrc, Src					
jalr	Rsrc	sub.p	FPdst, FPsrc1, FPsrc2	sne	Rdst, Rsrc, Src					
jr	Rsrc	Manipulação de Exceções e Traps								
		break	n							
		nop								
		eret								
		syscall								

Tabela I: Registos do MIPS e convenção de uso		
Nome Lóg.	Nome Real	Uso Convencionado
\$zero	\$0	Constante 0
\$at	\$1	Reservado pelo assembler
\$v0..\$v1	\$2..\$3	Cálculo de expressões e valor de retorno das
\$a0..\$a3	\$4..\$7	Primeiros 4 parâmetros das funções
\$t0..\$t7	\$8..\$15	Geral (não são preservados pelas funções)
\$s0..\$s7	\$16..\$23	Geral (não podem ser alterados pelas funções)
\$t8..\$t9	\$24..\$25	Geral (não são preservados pelas funções)
\$k0..\$k1	\$26..\$27	Reservado pelo <i>kernel</i> do S.O.
\$gp	\$28	Ponteiro para área global (<i>Global Pointer</i>)
\$sp	\$29	<i>Stack Pointer</i>
\$fp	\$30	<i>Frame Pointer</i>
\$ra	\$31	Endereço de retorno das funções (<i>Return Address</i>)

Tabela II: Registos da FPU do MIPS e convenção de uso	
Nome Lógico	Uso Convencionado
\$f0(\$f1)	Cálculo de expressões e valor de retorno das funções
\$f2(\$f3) ... \$f10(\$f11)	Geral (não são preservados pelas funções)
\$f12(\$f13) ... \$f14(\$f15)	Passagem de parâmetros para funções.
\$f16(\$f17) ... \$f18(\$f19)	Geral (não são preservados pelas funções)
\$f20(\$f21) ... \$f30(\$f31)	Geral (não podem ser alterados pelas funções)

Rev 2022 - MBC, JLA, AO, LAU, ACP

Tabela III: Notação			
Imm	Valor imediato (constante) de 16 bits	addr	Endereço na forma Imm(Rsrc) = (Rsrc) + Imm
IMM	Valor imediato de 32 bits	B_k(Rsrc)	Byte índice k de Rsrc
Rsrc (1, 2)	Registo fonte (1 ou 2)	FPdst	Registo destino do coprocessador aritmético
(Rsrc)	Conteúdo de Rsrc	FPsrc (1, 2)	Registo fonte do coprocessador aritmético (1 ou 2)
Rdst	Registo destino	C_z	Coprocessador n° z
CReg	Registo do Coprocessador C_z	Src	Rsrc ou IMM
sym	Endereço do símbolo (label) sym	Imm5	Valor imediato (constante) de 5 bits

Tabela IV: <i>System Calls</i> do MARS			
Protótipo equivalent em C	\$v0	Parâmetros de entrada	Retorno
void print_int10(int value)	1	\$a0 = value (int)	
void print_float(float value)	2	\$f12 = value (float)	
void print_double(double value)	3	\$f12 = value (double)	
void print_string(char *str)	4	\$a0 = str	
int read_int(void)	5		\$v0
float read_float(void)	6		\$f0
double read_double(void)	7		\$f0
void read_string(char *buf, int length)	8	\$a0 = buf, \$a1 = length	
void *sbrk(int amount)	9	\$a0 = amount	\$v0
void exit(void)	10		
void _print_char(char value)	11	\$a0 = value (char)	
char read_char(void)	12		\$v0
void print_int16(unsigned int value)	34	\$a0 = value (unsigned int)	
void print_int2(unsigned int value)	35	\$a0 = value (unsigned int)	
void print_intu10(unsigned int value)	36	\$a0 = value (unsigned int)	

Tabela V - Directivas do Assembler	
Directivas	Descrição
Para controlo dos Segmentos	
.data [address]	Coloca os próximos itens no segmento de dados do utilizador (opcionalmente a partir de <i>address</i>).
.text [address]	Coloca os próximos itens no segmento de código do utilizador (opcionalmente a partir de <i>address</i>).
.kdata [address]	Coloca os próximos itens no segmento de dados do <i>kernel</i> (opcionalmente a partir de <i>address</i>).
.ktext [address]	Coloca os próximos itens no segmento de código do <i>kernel</i> (opcionalmente a partir de <i>address</i>).
Para criação de constantes e variáveis em memória:	
.ascii str	Armazena uma <i>string</i> em memória sem lhe acrescentar o terminador '\0'.
.eqv label, valor	Substitui todas as ocorrências de <i>label</i> no programa por <i>valor</i> .
.asciiz str	Armazena uma <i>string</i> em memória acrescentando-lhe o terminador '\0'.
.byte b ₁ , ..., b _n	Armazena as grandezas de 8 bits b ₁ , ..., b _n em sucessivos bytes de memória.
.half h ₁ , ..., h _n	Armazena as grandezas de 16 bits h ₁ , ..., h _n em sucessivas meias palavras de memória.
.word w ₁ , ..., w _n	Armazena as grandezas de 32 bits w ₁ , ..., w _n em sucessivas palavras de memória.
.float f ₁ , ..., f _n	Armazena f ₁ , ..., f _n em vírgula flutuante, precisão simples (32 bits) no seg. de dados.
.double d ₁ , ..., d _n	Armazena d ₁ , ..., d _n em vírgula flutuante, precisão dupla (64 bits) no seg. de dados.
.space n	Reserva <i>n</i> bytes no segmento de dados, sem inicializar
Para controlo do alinhamento:	
.align n	Alinha o próximo item num endereço múltiplo de 2 ⁿ .
Para referências externas:	
.globl sym	Declara que o símbolo <i>sym</i> é global e pode ser referenciado em outros ficheiros.
.extern sym size	Declara que o item associado a <i>sym</i> ocupa <i>size</i> bytes e é um símbolo global.