Instruções da Máquina Nativa				Instruções da Máquina Virtual				ASM-SIL	
Transferência Memória-Registo (<i>Load</i>)		Cálculo c/ Inteiros: Operações Aritméticas		Transferência Memória-Registo (Load)		Salto Relativo (Branch)			
lb	Rdst,addr	add	Rdst,Rsrc1,Rsrc2	la	Rdst, ADDR	b	Label		
lbu	Rdst,addr	addi	Rdst, Rsrc, Imm	ld	Rdst, ADDR	beqz Rsrc, Label			
lh	Rdst,addr	addiu	Rdst, Rsrc, Imm	ulh	Rdst, ADDR	bge :	Rsrc,Src,La		
lhu	Rdst,addr	addu	Rdst, Rsrc1, Rsrc2	ulhu	Rdst,ADDR	bgeu Rsrc,Src,Label			
lw	Rdst,addr	div	Rsrc1, Rsrc2	ulw	Rdst, ADDR	bgt Rsrc,Src,Label			
lwl	Rdst, addr	divu	Rsrc1, Rsrc2	l.d	FPdst, ADDR	bgtu Rsrc, Src, Label			
lwr	Rdst,addr	mult	Rsrc1,Rsrc2	1.s	FPdst, ADDR	ble Rsrc, Src, Label			
lwcz	CReg, addr	multu	Rsrc1,Rsrc2		rência Registo-Memória (Store)	bleu Rsrc, Src, Label			
	ncia Registo-Memória (Store)	sub	Rdst, Rsrc1, Rsrc2	sd	Rdst, ADDR	blt Rsrc,Src,Label			
sb	Rsrc, addr	subu	Rdst, Rsrc1, Rsrc2	ush	Rdst, ADDR	bltu Rsrc,Src,Label			
sh	Rsrc, addr		Inteiros: Op. Lógicas <i>Bitwise</i>	usw	Rdst, ADDR	bnez Rsrc, Label			
SW	Rsrc, addr	and	Rdst, Rsrc1, Rsrc2	s.d	FPsrc, ADDR	oner nordy rader			
swl	Rsrc,addr	andi	Rdst, Rsrc, Imm	s.s	FPsrc, ADDR		Tabela I: Modos de Endereçamento		
	Rsrc, addr	nor	Rdst, Rsrc1, Rsrc2		rência Regist-Registo (Move)	Modo de	Endereçamento	Ī	ulo do Endereç
swr	·	or	·		Rdst, Rsrc		Ender eçamend	Conteúdo do registo	•
SWCZ	Creg, addr		Rdst, Rsrc1, Rsrc2	move	•	(reg)			reg
	ncia Regist-Registo (Move)	ori	Rdst, Rsrc, Imm		Rdst, FPsrc	Imm		Constante imm	
mfhi	Rdst	xor	Rdst,Rsrc1,Rsrc2		lação de Const. (Load Immediate)	imm(reg) Conteúdo do registo		0	
mflo	Rdst	xori	Rdst, Rsrc, Imm	li 	Rdst, IMM	sym		(label) sym	
mthi	Rsrc		/ Inteiros: Operações de Shift	l.d	FPdst, ADDR		sym +/- imm Endereço do símbol		•
mtlo	Rsrc	sll	Rdst, Rsrc1, Rsrc2	1.s	FPdst, ADDR	sym +/- im	ım (reg)	Conteúdo de reg + e	ndereço de sym +/- const. imm
mfcz	Rdst,Creg	sllv	Rdst, Rsrc1, Rsrc2		c/ Inteiros: Op. Aritmétic			~ -	
mtcz	Rsrc,Creg	sra	Rdst, Rsrc1, Rsrc2	abs	Rdst,Rsrc		1	egistos do MIPS e o	
mov.d	FPdst, FPsrc	srav	Rdst, Rsrc1, Rsrc2	div	Rdst, Rsrc, Src	Nome Lóg		Uso Convenciona	do
mov.s	FPdst, FPsrc	srl	Rdst, Rsrc1, Rsrc2	divu	Rdst, Rsrc, Src	\$zero	\$0	Constante 0	
Manipula	ção de Const. (Load Immediate)	srlv	Rdst, Rsrc1, Rsrc2	mul	Rdst, Rsrc, Src	\$at	\$1	Reservado pelo asse	
lui	Rdst,Imm	Cálculo e	m Vírgula Flutuant	mulo	Rdst, Rsrc, Src	\$v0\$v1	\$2\$3	Cálculo de expressõ	es e valor de retorno das funçõe s.
Instruçõe	s de Comparação	abs.p	FPdst, FPsrc	mulou	Rdst, Rsrc, Src	\$a0\$a3	\$4\$7	Primeiros 4 parâmet	os das funções
slt	Rdst, Rsrc1, Rsrc2	add.p	FPdst, FPsrc1, FPsrc2	neg	Rdst, Rsrc	\$t0\$t7	\$8\$15	Geral (não são prese	rvados pelas funções)
sltu	Rdst, Rsrc1, Rsrc2	c.eq.p	FPsrc1, FPsrc2	negu	Rdst, Rsrc	\$s0\$s7	\$16\$23	Geral (não podem se	r alterados pelas funções)
slti	Rdst, Rsrc, Imm	c.le.p	FPsrc1, FPsrc2	rem	Rdst, Rsrc, Src	\$t8\$t9	\$24\$25	Geral (não são prese	rvados pelas funções)
sltiu	Rdst, Rsrc, Imm	c.lt.p	FPsrc1, FPsrc2	remu	Rdst, Rsrc, Src	\$k0\$k1	\$26\$27	Reservado pelo kern	el do S.O.
Salto Rela	ativo (Branch) e Absoluto (Jump)	cvt.d.s	FPdst, FPsrc	Cálculo	c/ Inteiros: Op. Lógicas <i>Bitwise</i>	\$gp	\$28	Ponteiro para área gl	obal Global(Pointer)
bczf	Label	cvt.d.w	FPdst, FPsrc	not	Rdst,Rsrc	\$sp	\$29	Stack Pointer	, ,
bczt	Label	cvt.s.d	FPdst, FPsrc	Cálculo	c/ Inteiros: Operações de <i>Rotate</i>	\$fp	\$30	Frame Pointer	
beq	Rsrc1, Rsrc2, Label		FPdst, FPsrc	rol	Rdst, Rsrc, Src	\$ra	\$31	Endereço de retorno	das funções (Return Address)
bgez	Rsrc,Label	cvt.w.d	FPdst, FPsrc	ror	Rdst, Rsrc, Src		•	•	,
bgezal	Rsrc, Label	cvt.w.s	FPdst, FPsrc	Instruçõ	es de Comparação	Ta	Tabela III: Registos da FPU do MIPS		S e convenção de uso
bgtz		div.p	FPdst, FPsrc1, FPsrc2	seq	Rdst, Rsrc, Src	Nome Lóg	gico	Uso Convencionad	0
blez	Rsrc, Label	mul.p	FPdst, FPsrc1, FPsrc2	sge	Rdst, Rsrc, Src	\$f0(\$f1)			e valor de retorno das funçõe s
bltz	Rsrc,Label	neg.p	FPdst, FPsrc	sgeu	Rdst, Rsrc, Src			Geral (não são preserv	
bltzal	Rsrc, Label	sub.p	FPdst, FPsrc1, FPsrc2	sgt	Rdst, Rsrc, Src	\$f12(\$f13) \$f14(\$f15) Passagem de parâmetros			
bne	Rsrc1, Rsrc2, Label		ção de Excepções e <i>Traps</i>	sgtu	Rdst, Rsrc, Src		\$f16(\$f17) \$f18(\$f19) Geral (não são preservados pe		
j	Label	break	n	sle	Rdst, Rsrc, Src		\$120(\$121) \$130(\$131) Geral (não são preservados pelas funços \$120(\$121) \$130(\$131) Geral (não podem ser alterados pelas fu		
jal	Label	nop		sleu	Rdst, Rsrc, Src	φ120 (ψ121)	, φιο σ (φιο ι)	cara (mao podem ser t	permo rangoco,
jalr	Rsrc	eret		sne	Rdst, Rsrc, Src				
jr	Rsrc	syscall			,				

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)	Conteudo de Rsrc	FPsrc(1,2)	Registo fonte do coprocessador aritmético (1 ou 2)	
Rdst	Registo destino	$C_{\mathbf{z}}$	Coprocessador nº z	
CReg	Registo do Coprocessador Cz	ADDR	Um dos modos de endereçamento da Tabela I	
		Src	Rsrc ou IMM	

Tabela V - Directivas do Assembler					
Directivas	Descrição				
Para controlo dos Segmentos					
.data [address]	Coloca os próximos items no segmento de dados do ut ilizador (opcionalmente a partir de address).				
.text [address]	Coloca os próximos items no segmento de código do u tilizador (opcionalmente a partir de address).				
.kdata [address]	Coloca os próximos items no segmento de dados do kernel (opcionalmente a partir de address).				
.ktext [address]	Coloca os próximos items no segmento de código do kernel (opcionalmente a partir de address).				
Para criação de constantes e	variáveis em memória:				
.ascii str	Armazena uma string em memória sem lhe acrescentar o terminador '\0'.				
.asciiz str	Armazena uma string 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, \ldots, h_n	Armazena as grandezas de 16 bits h_1, \ldots, h_n em sucessivas meias palavras de memória.				
.word w_1 ,, w_n	Armazena as grandezas de 32 bits w_1, \ldots, w_n em sucessivas palavras de memória.				
.float f_1 ,, f_n	Armazena f ₁ ,, f _n em vírgula flutuante, precisão simples (32 bits) n o 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	Aloca n bytes (SPIM só deixa usar esta directiva no segmen to .data).				
Para controlo do alinhamen	to:				
.align n	Alinha o próximo item num endereço múltiplo de 2 n.				
Para referências externas:					
.globl sym	Declara que o símbolo sym é global e pode ser referenciado em outros ficheiros.				
.extern sym size	Declara que o item associado a sym ocupa size bytes e é um símbolo global.				

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