Instruções Nativas				Instruções Virtuais					DETI-UA - ACI
Transfer	ência Memória-Registo ( <i>Load</i> )	Cálculo c/ Inteiros: Operações Aritméticas		Transferência Memória-Registo (Load)		Salto Relativo (Branch)			
lb	Rdst, addr	add	Rdst, Rsrc1, Rsrc2	1.d	FPdst,addr	b		bel	
lbu	Rdst,addr	addi	Rdst, Rsrc, Imm	1.s	FPdst,addr	bge	Rsi	rc,Src,Lak	pel
lw	Rdst,addr	addiu	Rdst, Rsrc, Imm			bgeu	Rsi	rc,Src,Lab	pel
lwcz	CReg, addr	addu	Rdst, Rsrc1, Rsrc2	Transfe	rência Registo-Memória (Store)	bgt	Rsi	rc,Src,Lak	pel
ldcz	CReg, addr	div	Rsrc1, Rsrc2	s.d	FPsrc,addr	bgtu	Rs	rc,Src,Lal	pel
Transfer	ência Registo-Memória (Store)	divu	Rsrc1, Rsrc2	s.s	FPsrc,addr	ble	Rsi	rc,Src,Lak	pel
sb	Rsrc,addr	mult	Rsrc1, Rsrc2			bleu	Rsi	rc,Src,Lak	pel
sw	Rsrc,addr	multu	Rsrc1, Rsrc2	Transfe	rência Registo-Registo (Move)	blt	Rs	rc,Src,Lal	pel
SWCZ	Creg,addr	sub	Rdst, Rsrc1, Rsrc2	move	Rdst, Rsrc	bltu	Rs	rc,Src,Lal	pel
sdcz	Creg,addr	subu	Rdst, Rsrc1, Rsrc2			beq	Rsi	rc,Src,Lak	pel
Transfer	ência Registo-Registo (Move)	Cálculo c/	Inteiros: Op. Lógicas Bitwise	Manipu	lação de Const. (Load Imm/sym)	bne	Rs	rc,Src,La	pel
mfhi	Rdst	and	Rdst, Rsrc1, Rsrc2	la	Rdst,sym				
mflo	Rdst	andi	Rdst, Rsrc, Imm	li	Rdst,IMM				
mthi	Rsrc	nor	Rdst, Rsrc1, Rsrc2						
mtlo	Rsrc	or	Rdst, Rsrc1, Rsrc2						
${\tt mfc} z$	Rdst,Creg	ori	Rdst, Rsrc, Imm						
mtcz	Rsrc, Creg	xor	Rdst, Rsrc1, Rsrc2	Cálculo	c/ Inteiros: Op. Aritméticas				
mov.d	FPdst, FPsrc	xori	Rdst, Rsrc, Imm	mul	Rdst, Rsrc, Src				
mov.s	FPdst, FPsrc	Cálculo c/	Inteiros: Operações de Shift	mulu	Rdst, Rsrc, Src				
		sll	Rdst,Rsrc1,Imm5	mulo	Rdst, Rsrc, Src				
Manipula	ação de Const. (Load Immediate)	sllv	Rdst, Rsrc1, Rsrc2	mulou	Rdst, Rsrc, Src				
lui	Rdst,Imm	sra	Rdst, Rsrc1, Imm5				Т	abela I: Regi	stos do MIPS e convenção de uso
		srav	Rdst, Rsrc1, Rsrc2	div	Rdst, Rsrc, Src	Nome I			Uso Convencionado
Instruçõe	es de Comparação	srl	Rdst, Rsrc1, Imm5	divu	Rdst, Rsrc, Src	\$zero	10 <b>5</b> .	\$0	Constante 0
slt	Rdst, Rsrc1, Rsrc2	srlv	Rdst, Rsrc1, Rsrc2			\$at		\$1	Reservado pelo assemblador
sltu	Rdst, Rsrc1, Rsrc2	Cálculo er	n Vírgula Flutuante	rem	Rdst, Rsrc, Src	\$v0	\$v1	\$2\$3	Geral / valor de retorno das funções
slti	Rdst, Rsrc, Imm	add.p	FPdst, FPsrc1, FPsrc2	remu	Rdst, Rsrc, Src	\$a0:		\$4\$7	Geral / primeiros 4 parâmetros das funções
sltiu	Rdst, Rsrc, Imm	sub.p	FPdst, FPsrc1, FPsrc2			\$t0:		\$8\$15	Geral
-	lativo (Branch) e Absoluto (Jump)	div.p	FPdst, FPsrc1, FPsrc2			\$s0			Geral - não podem ser alterados pelas funções
bczf	Label	mul.p	FPdst, FPsrc1, FPsrc2			\$t8:			Geral
bczt	Label	_	·			\$k0			Reservado pelo kernel do S.O.
beq	Rsrc1, Rsrc2, Label	neg.p	FPdst, FPsrc	Cálculo	c/ Inteiros: Operações de <i>Rotate</i>	\$gp		\$28	Ponteiro para área global (Global Pointer)
bne	Rsrc1,Rsrc2,Label	abs.p	FPdst, FPsrc	rol	Rdst, Rsrc, Src	\$sp		\$29	Stack Pointer
bgez	Rsrc, Label		·	ror	Rdst, Rsrc, Src	\$fp		\$30	Frame Pointer
bgtz	Rsrc, Label	cvt.d.s	FPdst, FPsrc		, ,	\$ra		\$31	Endereço de retorno das funções ( <i>Return Address</i> )
blez	Rsrc,Label		FPdst, FPsrc						*
bltz	Rsrc, Label		FPdst, FPsrc			,	Tabel	a II: Registos	da FPU do MIPS e convenção de uso
j	Label		FPdst, FPsrc			Nome I			Uso Convencionado
jal	Label		FPdst, FPsrc			\$f0	JUZICU		Geral / valor de retorno das funções
jalr	Rsrc		FPdst, FPsrc			\$f2	\$f10		Geral
jr	Rsrc		,			\$f12.			Geral / passagem de parâmetros para funções
Manipulação de Excepções e Traps		Comparação em Vírgula Flutuante		1					Geral
break	n		FPsrc1, FPsrc2	1		\$f20.			Geral - não podem ser alterados pelas funções
nop	**		FPsrc1, FPsrc2			7120.	• 4 T O (	<u> </u>	nuo poucii sei aiteratus peias itinções
eret			FPsrc1, FPsrc2						Rev 2023 - MBC, JLA, AO, LAU, ACP
syscal	1	[	1101,110102						1 104 2020 IVIDO, ULA, AO, LAO, AOI
Jyscar.	<b>-</b>								

Tabela III: Notação						
Imm	Valor imediato (constante) de 16 bits	addr	Endereço na forma <b>Imm</b> ( <b>Rsrc</b> ) = ( <b>Rsrc</b> ) + <b>Imm</b>			
IMM	Valor imediato de 32 bits	B <sub>k</sub> (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	Src	Rsrc ou IMM			
CReg	Registo do Coprocessador Cz	cz	Coprocessador nº z (0 or 1)			
sym	Endereço do símbolo (label) sym	Imm5	Valor imediato (constante) de 5 bits			
Label	Endereço de uma instrução	.p	Precisão: substituir por .s ou .d			

Tabela IV: System Calls do MARS					
Protótipo equivalent em C	\$v0	Parâmetros de entrada	Retorno		
<pre>void print_int10(int value)</pre>	1	\$a0 = value			
<pre>void print_float (float value)</pre>	2	\$f12 = value			
<pre>void print_double(double value)</pre>	ო	\$f12 = value			
<pre>void print_string(char *str)</pre>	4	\$a0 = str			
<pre>int read_int(void)</pre>	5		\$v0		
<pre>float read_float(void)</pre>	6		\$f0		
double read_double(void)	7		\$f0		
<pre>void read_string(char *buf, int length)</pre>	8	\$a0=buf,\$a1=length			
<pre>void *sbrk(int amount)</pre>	9	\$a0 = amount	\$v0		
<pre>void exit(void)</pre>	10				
<pre>void print_char(char value)</pre>	11	\$a0 = value			
char read_char(void)	12		\$v0		
<pre>void print_int16(unsigned int value)</pre>	34	\$a0 = value			
<pre>void print_int2(unsigned int value)</pre>	35	\$a0 = value			
<pre>void print_intul0(unsigned int value)</pre>	36	\$a0 = value			

Tabela V - Directivas do Assembler					
Directivas	Descrição				
Para controlo dos Segmentos					
.data [address]	Coloca os próximos items no segmento de dados do utilizador (opcionalmente a partir de address).				
.text [address]	Coloca os próximos items no segmento de código do utilizador (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 <i>string</i> em memória sem lhe acrescentar o terminador '\0'.				
.asciiz str	Armazena uma string em memória acrescentando-lhe o terminador '\0'.				
.space n	Reserva <i>n</i> bytes no segmento de dados, sem inicializar				
.byte $b_1$ ,, $b_n$	Armazena as grandezas de 8 bits b <sub>1</sub> ,, b <sub>n</sub> em sucessivos bytes de memória.				
.word $w_1$ ,, $w_n$	Armazena as grandezas de 32 bits w <sub>1</sub> ,, w <sub>n</sub> em sucessivas palavras de memória.				
.float $f_1$ ,, $f_n$	Armazena f <sub>1</sub> ,, f <sub>n</sub> em vírgula flutuante, precisão simples (32 bits) no seg. de dados.				
.double $d_1$ ,, $d_n$	Armazena d <sub>1</sub> ,, d <sub>n</sub> em vírgula flutuante, precisão dupla (64 bits) no seg. de dados.				
.eqv label, valor	Substitui todas as ocorrências de label no programa por valor.				
Para controlo do alinhament	to:				
.align n	Alinha o próximo item num endereço múltiplo de 2 <sup>n</sup> .				
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.				
.include filename	Insere o conteúdo do ficheiro especificado (o nome do ficheiro é colocado entre aspas).				