Instruções da Máquina Nativa

Transferência Memória-Registo (*Load*)

```
lb      Rdst,addr
lbu     Rdst,addr
lh      Rdst,addr
lhu     Rdst,addr
lw      Rdst,addr
lwl     Rdst,addr
lwr     Rdst,addr
lwcz    CReg,addr
```

Transferência Registo-Memória (*Store*)

```
sb      Rsrc,addr
sh      Rsrc,addr
sw      Rsrc,addr
swl     Rsrc,addr
swr     Rsrc,addr
swcz    Creg,addr
```

Transferência Registo-Registo (*Move*)

```
mfhi    Rdst
mflo    Rdst
mthi    Rsrc
mtlo    Rsrc
mfcz    Rdst,Creg
mtcz    Rsrc,Creg
mov.d   FPdst,FPsrc
mov.s   FPdst,FPsrc
```

Manipulação de Const. (*Load Immediate*)

```
lui     Rdst,Imm
```

Instruções de Comparação

```
slt     Rdst,Rsrc1,Rsrc2
sltu    Rdst,Rsrc1,Rsrc2
slti    Rdst,Rsrc,Imm
sltiu   Rdst,Rsrc,Imm
```

Salto Relativo (*Branch*) e Absoluto (*Jump*)

```
bczf    Label
bczt    Label
beq     Rsrc1,Rsrc2,Label
bgez    Rsrc,Label
bgezal  Rsrc,Label
bgtz    Rsrc,Label
blez    Rsrc,Label
bltz    Rsrc,Label
bltzal  Rsrc,Label
bne     Rsrc1,Rsrc2,Label
j       Label
jal     Label
jalr    Rsrc
jr      Rsrc
```

Cálculo c/ Inteiros: Operações Aritméticas

```
add     Rdst,Rsrc1,Rsrc2
addi    Rdst,Rsrc,Imm
addiu   Rdst,Rsrc,Imm
addu    Rdst,Rsrc1,Rsrc2
div     Rsrc1,Rsrc2
divu    Rsrc1,Rsrc2
mult    Rsrc1,Rsrc2
multu   Rsrc1,Rsrc2
sub     Rdst,Rsrc1,Rsrc2
subu    Rdst,Rsrc1,Rsrc2
```

Cálculo c/ Inteiros: Op. Lógicas *Bitwise*

```
and     Rdst,Rsrc1,Rsrc2
andi    Rdst,Rsrc,Imm
nor     Rdst,Rsrc1,Rsrc2
or      Rdst,Rsrc1,Rsrc2
ori     Rdst,Rsrc,Imm
xor     Rdst,Rsrc1,Rsrc2
xori    Rdst,Rsrc,Imm
```

Cálculo c/ Inteiros: Operações de *Shift*

```
sll     Rdst,Rsrc1,Rsrc2
sllv    Rdst,Rsrc1,Rsrc2
sra     Rdst,Rsrc1,Rsrc2
srav    Rdst,Rsrc1,Rsrc2
srl     Rdst,Rsrc1,Rsrc2
srlv    Rdst,Rsrc1,Rsrc2
```

Cálculo em Vírgula Flutuante

```
abs.p   FPdst,FPsrc
add.p   FPdst,FPsrc1,FPsrc2
c.eq.p  FPsrc1,FPsrc2
c.le.p  FPsrc1,FPsrc2
c.lt.p  FPsrc1,FPsrc2
cvt.d.s FPdst,FPsrc
cvt.d.w FPdst,FPsrc
cvt.s.d FPdst,FPsrc
cvt.s.w FPdst,FPsrc
cvt.w.d FPdst,FPsrc
cvt.w.s FPdst,FPsrc
div.p   FPdst,FPsrc1, FPsrc2
mul.p   FPdst,FPsrc1, FPsrc2
neg.p   FPdst,FPsrc
sub.p   FPdst,FPsrc1, FPsrc2
```

Manipulação de Excepções e *Traps*

```
break   n
nop
eret
syscall
```

Instruções da Máquina Virtual

Transferência Memória-Registo (*Load*)

```
la      Rdst,ADDR
ld      Rdst,ADDR
ulh     Rdst,ADDR
ulhu    Rdst,ADDR
ulw     Rdst,ADDR
l.d     FPdst,ADDR
l.s     FPdst,ADDR
```

Transferência Registo-Memória (*Store*)

```
sd      Rdst,ADDR
ush     Rdst,ADDR
usw     Rdst,ADDR
s.d     FPsrc,ADDR
s.s     FPsrc,ADDR
```

Transferência Registo-Registo (*Move*)

```
move    Rdst,Rsrc
mfc1.d  Rdst,FPsrc
```

Manipulação de Const. (*Load Immediate*)

```
li      Rdst,IMM
l.d     FPdst,ADDR
l.s     FPdst,ADDR
```

Cálculo c/ Inteiros: Op. Aritméticas

```
abs     Rdst,Rsrc
div     Rdst,Rsrc,Src
divu    Rdst,Rsrc,Src
mul     Rdst,Rsrc,Src
mulo    Rdst,Rsrc,Src
mulou   Rdst,Rsrc,Src
neg     Rdst,Rsrc
negu    Rdst,Rsrc
rem     Rdst,Rsrc,Src
remu    Rdst,Rsrc,Src
```

Cálculo c/ Inteiros: Op. Lógicas *Bitwise*

```
not     Rdst,Rsrc
```

Cálculo c/ Inteiros: Operações de *Rotate*

```
rol     Rdst,Rsrc,Src
ror     Rdst,Rsrc,Src
```

Instruções de Comparação

```
seq     Rdst,Rsrc,Src
sge     Rdst,Rsrc,Src
sgeu    Rdst,Rsrc,Src
sgt     Rdst,Rsrc,Src
sgtu    Rdst,Rsrc,Src
sle     Rdst,Rsrc,Src
sleu    Rdst,Rsrc,Src
sne     Rdst,Rsrc,Src
```

Salto Relativo (*Branch*)

```
b       Label
beqz    Rsrc, Label
bge     Rsrc,Src,Label
bgeu    Rsrc,Src,Label
bgt     Rsrc,Src,Label
bgtu    Rsrc,Src,Label
ble     Rsrc,Src,Label
bleu    Rsrc,Src,Label
blt     Rsrc,Src,Label
bltu    Rsrc,Src,Label
bnez    Rsrc,Label
```

Tabela I: Modos de Endereçamento

Modo de Endereçamento	Cálculo do Endereço
(reg)	Conteúdo do registo reg
Imm	Constante imm
imm(reg)	Conteúdo do registo reg + constante imm
sym	Endereço do símbolo (*label*) sym
sym +/- imm	Endereço do símbolo sym +/- constante imm
sym +/- imm (reg)	Conteúdo de reg + endereço de sym +/- const. imm

Tabela II: Registos do MIPS e convenção de uso

Nome Lóg.	Nome Real	Uso Convencionado
$zero	$0	Constante 0
$at	$1	Reservado pelo assemblador
$v0..$v1	$2..$3	Cálculo de expressões e valor de retorno das funções.
$a0..$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 *kernel* do S.O.
$gp	$28	Ponteiro para área global (*Global Pointer*)
$sp	$29	*Stack Pointer*
$fp	$30	*Frame Pointer*
$ra	$31	Endereço de retorno das funções (*Return Address*)

Tabela III: Registos da FPU do MIPS e convenção de uso

Nome Lógico	Uso Convencionado
$f0($f1) ... $f2($f3)	Cálculo de expressões e valor de retorno das funções
$f4($f5) ... $f10($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)

MBC, JLA, AO, LAU, ACP

Tabela VIII: 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)`	Conteudo de **Rsrc**	`FPsrc(1,2)`	Registo fonte do coprocessador aritmético (1 ou 2)
`Rdst`	Registo destino	C_z	Coprocessador nº **z**
`CReg`	Registo do Coprocessador C_z	`ADDR`	Um dos modos de endereçamento da Tabela I
		`Src`	**Rsrc** ou **IMM**
([1]) Estas instruções podem usar operandos na forma "**`Rrdt,IMM`**" sendo **Rrdt** simultaneamente o registo fonte e destino			

Tabela IV: Registos de I/O mapeado em memória				
Nome	**Endereço**	**Bit 7-3**	**Bit 1**	**Bit 0**
Controlo de Recepção	0xffff0000	Não usados	Int Enable	Ready
Dados do Receptor	0xffff0004	Byte recebido		
Controlo de Emissão	0xffff0008	Não usados	Int Enable	Ready
Dados do Emissor	0xffff000c	Byte a enviar		

Tabela IX - 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 *string* em memória sem lhe acrescentar o terminador '\0'.
`.asciiz str`	Armazena uma *string* em memória acrescentando-lhe o terminador '\0'.
`.byte` $b_1, \ldots, b_n$	Armazena as grandezas de 8 bits $b_1, \ldots, b_n$ 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` $w_1, \ldots, w_n$	Armazena as grandezas de 32 bits $w_1, \ldots, w_n$ em sucessivas palavras de memória.
`.float` $f_1, \ldots, f_n$	Armazena $f_1, \ldots, f_n$ em vírgula flutuante, precisão simples (32 bits) no seg. de dados.
`.double` $d_1, \ldots, d_n$	Armazena $d_1, \ldots, d_n$ em vírgula flutuante, precisão dupla (64 bits) no seg. de dados.
`.space n`	Aloca *n* bytes (SPIM só deixa usar esta directiva no segmento `.data`).
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 `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 V: *System Calls* do MARS			
Protótipo equivalent em C	**$v0**	**Parâmetros de entrada**	**Retorno**
void print_int10(**int** value)	1	$a0 = int	
void print_float(**float** value)	2	$f12 = float	
void print_double(**double** value)	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
void read_string(**char** *buf, **int** len)	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
void print_int16(**unsigned int** value)	34	$a0	
void print_int2(**unsigned int** value)	35	$a0	
void print_intu10(**unsigned int** value)	36	$a0	