

Set de Instruções da Máquina Nativa (1)

DETI-UA - ACI

Menemônica		Descrição	OAMV	E/O	Algoritmo
Instruções de Transferência Memória-Registo (Load)					
lb	Rdst, addr	Load Byte	Rdst, ADDR		Rdst = *(byte)addr
lbu	Rdst, addr	Load Byte Unsigned	Rdst, ADDR		Rdst = *(unsigned byte)addr
lh	Rdst, addr	Load Halfword	Rdst, ADDR		Rdst = *(halfword)addr
lhu	Rdst, addr	Load Halfword Unsigned	Rdst, ADDR		Rdst = *(unsigned halfword)addr
lw	Rdst, addr	Load Word	Rdst, ADDR		Rdst = *(word)addr
lwl	Rdst, addr	Load Word Left	Rdst, ADDR		n=3-(addr&0x3); addr&=0xffffffffc; for (k=3;k>=n;k--,addr++) B _k (Rsrc)=*addr;
lwr	Rdst, addr	Load Word Right	Rdst, ADDR		n=3-(addr&0x3); for (k=n;k>=0;k--,addr++) B _k (Rsrc)=*addr;
lwcz	CReg, addr	Load Word Coprocessor z	Rdst, ADDR		C _z Reg = *(word)addr
Instruções de Transferência Registo-Memória (Store)					
sb	Rsrc, addr	Store Byte	Rsrc, ADDR		*addr = (byte)Rsrc
sh	Rsrc, addr	Store Halfword	Rsrc, ADDR		*addr = (halfword)Rsrc
sw	Rsrc, addr	Store Word	Rsrc, ADDR		*addr = (word)Rsrc
swl	Rsrc, addr	Store Word Left	Rsrc, ADDR		n=3-(addr&0x3); addr&=0xffffffffc; for (k=3;k>=n;k--,addr++) *addr=B _k (Rsrc);
swr	Rsrc, addr	Store Word Right	Rsrc, ADDR		n=3-(addr&0x3); for (k=n;k>=0;k--,addr++) *addr=B _k (Rsrc);
swcz	Creg, addr	Store Word Coprocessor z	Rsrc, ADDR		*(word)addr = C _z Reg
Instruções de Transferência Registo-Registo (Move)					
mghi	Rdst	Move From HI			Rdst = HI
mflo	Rdst	Move From LO			Rdst = LO
mthi	Rsrc	Move To HI			HI = Rsrc
mtlo	Rsrc	Move To LO			LO = Rsrc
mfcz	Rdst, CReg	Move From Coprocessor C _z			Rdst = C _z Reg
mtcz	Rsrc, CReg	Move To Coprocessor C _z			C _z Reg = Rsrc
mov.d	FPdst, FPsrc	Move Double			FPdst = FPsrc
mov.s	FPdst, FPsrc	Move Single			FPdst = FPsrc
Instruções de Manipulação de Constantes (Load Immediate)					
lui	Rdst, Imm	Load Upper Immediate			Rdst = Imm << 16
Instruções de Cálculo sobre Inteiros: Operações Aritméticas					
add	Rdst, Rsrc1, Rsrc2	Add	Rdst, Rsrc, IMM ⁽¹⁾	✓	Rdst = Rsrc + Src
addi	Rdst, Rsrc, Imm	Add Immediate	Rdst, Rsrc, IMM ⁽¹⁾	✓	Rdst = Rsrc + Imm
addiu	Rdst, Rsrc, Imm	Add Immediate Unsigned	Rdst, Rsrc, IMM ⁽¹⁾		Rdst = Rsrc + Imm
addu	Rdst, Rsrc1, Rsrc2	Add Unsigned	Rdst, Rsrc, IMM ⁽¹⁾		Rdst = Rsrc + Src
div	Rsrc1, Rsrc2	Divide		✓	HI = Rsrc % Rsrc2; LO = Rsrc / Rsrc2
divu	Rsrc1, Rsrc2	Divide Unsigned			HI = Rsrc % Rsrc2; LO = Rsrc / Rsrc2
mult	Rsrc1, Rsrc2	Multiply			HI, LO = Rsrc1 * Rsrc2
multu	Rsrc1, Rsrc2	Multiply Unsigned			HI, LO = Rsrc1 * Rsrc2
sub	Rdst, Rsrc1, Rsrc2	Subtract	Rdst, Rsrc, IMM ⁽¹⁾	✓	Rdst = Rsrc - Src
subu	Rdst, Rsrc1, Rsrc2	Subtract Unsigned	Rdst, Rsrc, IMM ⁽¹⁾		Rdst = Rsrc - Src

Set de Instruções da Máquina Nativa (2)

Menemónica		Descrição	OAMV	E/O	Algoritmo
Instruções de Cálculo sobre Inteiros: Operações Lógicas Bit-a-Bit (<i>Bitwise</i>)					
and	Rdst, Rsrc1, Rsrc2	And (Bitwise)	Rdst, Rsrc, IMM ⁽¹⁾		Rdst = Rsrc & Src
andi	Rdst, Rsrc, Imm	And Immediate (Bitwise)	Rdst, Rsrc, IMM ⁽¹⁾		Rdst = Rsrc & Imm
nor	Rdst, Rsrc1, Rsrc2	Nor (Bitwise)	Rdst, Rsrc, IMM ⁽¹⁾		Rdst = ~(Rsrc Src)
or	Rdst, Rsrc1, Rsrc2	Or (Bitwise)	Rdst, Rsrc, IMM ⁽¹⁾		Rdst = Rsrc Src
ori	Rdst, Rsrc, Imm	Or Immediate (Bitwise)	Rdst, Rsrc, IMM ⁽¹⁾		Rdst = Rsrc Imm
xor	Rdst, Rsrc1, Rsrc2	XOR	Rdst, Rsrc, IMM ⁽¹⁾		Rdst = Rsrc ^ Src
xori	Rdst, Rsrc, Imm	XOR Immediate	Rdst, Rsrc, IMM ⁽¹⁾		Rdst = Rsrc ^ Imm
Instruções de Cálculo sobre Inteiros: Operações de Deslocamento (<i>Shift</i>) e de Rotação (<i>Rotate</i>)					
sll	Rdst, Rsrc1, Rsrc2	Shift Left Logical	Rdst, Rsrc, IMM		Rdst = Rsrc << Src
sllv	Rdst, Rsrc1, Rsrc2	Shift Left Logical Variable			Rdst = Rsrc << Rsrc2
sra	Rdst, Rsrc1, Rsrc2	Shift Right Arithmetic	Rdst, Rsrc, IMM		Rdst = Rsrc >> Src
srav	Rdst, Rsrc1, Rsrc2	Shift Right Arithmetic Variable			Rdst = Rsrc >> Rsrc2
srl	Rdst, Rsrc1, Rsrc2	Shift Right Logical	Rdst, Rsrc, IMM		Rdst = Rsrc >> Src
srlv	Rdst, Rsrc1, Rsrc2	Shift Right Logical Variable			Rdst = Rsrc >> Rsrc2
Instruções de Comparação					
slt	Rdst, Rsrc1, Rsrc2	Set on Less Than	Rdst, Rsrc, IMM		Rdst = (Rsrc < Src) ? 1 : 0
sltu	Rdst, Rsrc1, Rsrc2	Set on Less Than Unsigned	Rdst, Rsrc, IMM		Rdst = (Rsrc < Src) ? 1 : 0
slti	Rdst, Rsrc, Imm	Set on Less Than Immediate	Rdst, Rsrc, IMM		Rdst = (Rsrc < Imm) ? 1 : 0
sltiu	Rdst, Rsrc, Imm	Set on Less Than Imm. Unsigned	Rdst, Rsrc, IMM		Rdst = (Rsrc < Imm) ? 1 : 0
Instruções de Salto Relativo (<i>Branch</i>) e Salto Absoluto (<i>Jump</i>)					
bczf	Label	Branch Coprocessor z FALSE			if (c _z _flag == FALSE) goto label
bczt	Label	Branch Coprocessor z TRUE			if (c _z _flag == TRUE) goto label
beq	Rsrc1, Rsrc2, Label	Branch on Equal	Rsrc1, IMM, Label		if (Rsrc1 == Src2) goto label
bgez	Rsrc, Label	Branch on Greater Than or Equal Zero			if (Rsrc1 >= 0) goto label
bgezal	Rsrc, Label	Branch on Greater Than or Equal Zero and Link			if (Rsrc1 >= 0) {\$31=PC; goto label }
bgtz	Rsrc, Label	Branch on Greater Than Zero			if (Rsrc1 > 0) goto label
blez	Rsrc, Label	Branch on Less Than or Equal Zero			if (Rsrc1 <= 0) goto label
bltz	Rsrc, Label	Branch on Less Than Zero			if (Rsrc1 < 0) goto label
bltzal	Rsrc, Label	Branch on Less Than Zero and Link			if (Rsrc1 < 0) {\$31=PC; goto label }
bne	Rsrc1, Rsrc2, Label	Branch on Not Equal	Rsrc1, IMM, Label		if (Rsrc1 != Src2) goto label
j	Label	Jump			goto label
jal	Label	Jump and Link			\$31=PC; goto label
jalr	Rsrc	Jump and Link Register			\$31=PC; goto Rsrc
jr	Rsrc	Jump Register			goto Rsrc

Set de Instruções da Máquina Nativa (3)

Menemónica	Descrição	E/O	Algoritmo
Instruções de Cálculo em Vírgula Flutuante			
abs.p Fpdst,FPsrc	Absolute Value		FPdst = FPsrc
add.p Fpdst,FPsrc1,FPsrc2	Addition		FPdst = FPsrc1 + FPsrc2
c.eq.p FPsrc1,FPsrc2	Compare Equal		fp_flag = (FPsrc1 == FPsrc2) ? true : false
c.le.p FPsrc1,FPsrc2	Compare Less Than or Equal		fp_flag = (FPsrc1 <= FPsrc2) ? true : false
c.lt.p FPsrc1,FPsrc2	Compare Less Than		fp_flag = (FPsrc1 < FPsrc2) ? true : false
cvt.d.s Fpdst,FPsrc	Convert Single to Double		FPdst = (double)FPsrc
cvt.d.w Fpdst,FPsrc	Convert Integer to Double		FPdst = (double)FPsrc
cvt.s.d Fpdst,FPsrc	Convert Double to Single		FPdst = (float)FPsrc
cvt.s.w Fpdst,FPsrc	Convert Integer to Single		FPdst = (float)FPsrc
cvt.w.d Fpdst,FPsrc	Convert Double to Integer		FPdst = (int)FPsrc
cvt.w.s Fpdst,FPsrc	Convert Single to Integer		FPdst = (int)FPsrc
div.p Fpdst,FPsrc1,FPsrc2	Divide		FPdst = FPsrc1 / FPsrc2
mul.p Fpdst,FPsrc1,FPsrc2	Multiply		FPdst = FPsrc1 * FPsrc2
neg.p Fpdst,FPsrc	Negate		FPdst = 0 - FPsrc
sub.p Fpdst,FPsrc1,FPsrc2	Subtract		FPdst = FPsrc1 - FPsrc2
Instruções para Manipulação de Exceções e Traps			
break n	Break		
Nop	No Operation		
Rfe	Return From Exception		
Syscall	System Call		

Set de Instruções da Máquina Virtual (1)

Menemónica	Descrição	E/O	Algoritmo
Instruções de Transferência Memória-Registo (Load)			
la Rdst,ADDR	Load Address		Rdst = ADDR
ld Rdst,ADDR	Load Double-Word		Rdst,Rdst+1 = *(doubleword)ADDR
ulh Rdst,ADDR	Unaligned Load Halfword		Rdst = *(halfword)ADDR
ulhu Rdst,ADDR	Unaligned Load Halfword Unsigned		Rdst = *(halfword)ADDR
ulw Rdst,ADDR	Unaligned Load Word		Rdst = *(word)ADDR
l.d Fpdst,ADDR	Load Double		FPdst = *(double)ADDR
l.s Fpdst,ADDR	Load Single		FPdst = *(single)ADDR
Instruções de Transferência Registo-Memória (Store)			
sd Rdst,ADDR	Store Double-Word		*ADDR = Rdst,Rdst+1
ush Rdst,ADDR	Unaligned Store Halfword		*ADDR = (halfword)Rdst
usw Rdst,ADDR	Unaligned Store Word		*ADDR = (word)Rdst
s.d FPsrc,ADDR	Store Double		*(double)ADDR = FPsrc
s.s FPsrc,ADDR	Store Single		*(single)ADDR = FPsrc
Instruções de Transferência Registo-Registo (Move)			
move Rdst,Rsrc	Move		Rdst = Rsrc
mfc1.d Rdst,FPsrc	Move Double From Coprocessor C1		Rdst,Rdst+1 = FPsrc

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 do registo reg + endereço do símbolo sym +/- constante imm

Tabela II: Registos do MIPS e convenção de uso		
Nome Lógico	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 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 retornos das funções (Return Address)

Set de Instruções da Máquina Virtual (2)

DETI-UA - ACI

Menemónica	Descrição	E/O	Algoritmo
Instruções de Manipulação de Constantes (<i>Load Immediate</i>)			
li Rdst, IMM	Load Immediate		Rdst = IMM
l.d FPDst, ADDR	Load Immediate Double		FPdst = double
l.s FPDst, ADDR	Load Immediate Single		FPdst = float

Instruções de Cálculo sobre Inteiros: Operações Aritméticas			
abs Rdst, Rsrc	Absolute Value		Rdst = Rsrc
div Rdst, Rsrc, Src	Division	✓	Rdst = Rsrc / Src
divu Rdst, Rsrc, Src	Division Unsigned	✓	Rdst = Rsrc / Src
mul Rdst, Rsrc, Src	Multiply		Rdst = Rsrc * Src
mulo Rdst, Rsrc, Src	Multiply	✓	Rdst = Rsrc * Src
mulou Rdst, Rsrc, Src	Multiply Unsigned	✓	Rdst = Rsrc * Src
neg Rdst, Rsrc	Negate		Rdst = - Rsrc
negu Rdst, Rsrc	Negate		Rdst = 0 - Rsrc
rem Rdst, Rsrc, Src	Remainder	✓	Rdst = Rsrc % Src
remu Rdst, Rsrc, Src	Remainder Unsigned	✓	Rdst = Rsrc % Src

Instruções de Cálculo sobre Inteiros: Operações Lógicas Bit-a-Bit (<i>Bitwise</i>)			
not Rdst, Rsrc	Not (Bitwise)		

Instruções de Cálculo sobre Inteiros: Operações de Deslocamento (<i>Shift</i>) e de Rotação (<i>Rotate</i>)			
rol Rdst, Rsrc, Src	Rotate Left		Rdst = Rsrc <<° Src
ror Rdst, Rsrc, Src	Rotate Right		Rdst = Rsrc >>° Src

Instruções de Comparação			
seq Rdst, Rsrc, Src	Set on Equal		Rdst = (Rsrc == Src) ? 1 : 0
sge Rdst, Rsrc, Src	Set on Greater Than or Equal		Rdst = (Rsrc >= Src) ? 1 : 0
sgeu Rdst, Rsrc, Src	Set on Greater Than or Equal Unsigned		Rdst = (Rsrc >= Src) ? 1 : 0
sgt Rdst, Rsrc, Src	Set on Greater Than		Rdst = (Rsrc > Src) ? 1 : 0
sgtu Rdst, Rsrc, Src	Set on Greater Than Unsigned		Rdst = (Rsrc > Src) ? 1 : 0
sle Rdst, Rsrc, Src	Set on Less Than or Equal		Rdst = (Rsrc <= Src) ? 1 : 0
sleu Rdst, Rsrc, Src	Set on Less Than or Equal Unsigned		Rdst = (Rsrc <= Src) ? 1 : 0
sne Rdst, Rsrc, Src	Set on Not Equal		Rdst = (Rsrc != Src) ? 1 : 0

Instruções de Salto Relativo (<i>Branch</i>) e Salto Absoluto (<i>Jump</i>)			
b Label	Branch		goto label
beqz Rsrc, Label	Branch on Equal Zero		if (Rsrc == 0) goto label
bge Rsrc, Src, Label	Branch on Greater Than or Equal		if (Rsrc >= Src) goto label
bgeu Rsrc, Src, Label	Branch on Greater Than or Equal Unsigned		if (Rsrc >= Src) goto label
bgt Rsrc, Src, Label	Branch on Greater Than		if (Rsrc > Src) goto label
bgtu Rsrc, Src, Label	Branch on Greater Than Unsigned		if (Rsrc > Src) goto label
ble Rsrc, Src, Label	Branch on Less Than or Equal		if (Rsrc <= Src) goto label
bleu Rsrc, Src, Label	Branch on Less Than or Equal Unsigned		if (Rsrc <= Src) goto label
blt Rsrc, Src, Label	Branch on Less Than		if (Rsrc < Src) goto label
bltu Rsrc, Src, Label	Branch on Less Than Unsigned		if (Rsrc < Src) goto label
bnez Rsrc, Label	Branch on Not Equal Zero		if (Rsrc != 0) goto label

Tabela III: 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 IV: 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)

Tabela V: Registos do CP0 do MIPS		
Nome Lógico	Nome Real	Conteúdo
\$BadVAddr	\$8	Endereço de memória inválido que causou a exceção
\$Status	\$12	Interrupt mask & Enable bits
\$Cause	\$13	Tipo de exceção e interrupt bits
\$EPC	\$14	Endereço da instrução que causou a exceção

Tabela VI: Valores dos bits [5..2] do registo Cause		
Valor	Nomel	Significado
0	INT	External Interrupt
4	ADDRL	Add error exception (load or store)
5	ADDRS	Add error exception (fetch)
6	IBUS	Bus error on instruction fetch
7	DBUS	Bus error on data load or store
8	SYSCALL	Syscall exception
9	BKPT	Break point exception
10	RI	Reserved instruction exception
12	OVF	Overflow exception

Tabela VII: 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 nº z
OAMV	Operandos Alternativos em Modo Virtual	ADDR	Um dos modos de endereçamento da Tabela I
E/O	Excepção gerada em caso de overflow	Src	Rsrc ou IMM
CReg	Registo do Coprocessador C_z		
(1) Estas instruções podem usar operandos na forma “ Rrdt, IMM ” sendo Rrdt simultaneamente o registo fonte e destino			

Tabela VIII: <i>System Calls</i> do MARS			
Protótipo equivalent em C	\$v0	Parâmetros de entrada	Retorno
void print_int10(int value)	1	\$a0 = value	
void print_float(float value)	2	\$f12 = value	
void print_double(double value)	3	\$f12 = value	
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 = 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	

Tabela IX - 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>). Todos os itens devem medir 32 bits, ou seja, serem instruções ou palavras (<i>words</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>). Todos os itens devem medir 32 bits, ou seja, serem instruções ou palavras (<i>words</i>).
Para criação de constantes e variáveis em memória:	
.ascii str	Armazena uma <i>string</i> em memória sem lhe acrescentar o terminador NULL.
.asciiz str	Armazena uma <i>string</i> em memória acrescentando-lhe o terminador NULL.
.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 os números em vírgula flutuante com precisão simples (32 bits) f ₁ , ..., f _n em posições de memória sucessivas.
.double d ₁ , ..., d _n	Armazena os números em vírgula flutuante com precisão dupla (64 bits) d ₁ , ..., d _n em posições de memória sucessivas.
.space n	Aloca <i>n</i> 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 . Por exemplo .align 2 seguido de .word xpto garante que a palavra xpto é armazenada num endereço múltiplo de 4.
.align 0	Desliga o alinhamento automático das directivas .half, .word, .float, e .double até à próxima directiva .data ou .kdata.
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. Esta directiva permite ao assembler armazenar o item numa porção do segmento de dados que seja eficientemente acedido através do registo \$gp.