

MIPS Guia de Referência

①



CONJUNTO BÁSICO DE INSTRUÇÕES

NOME, MNEMÔNICO	FOR-	OPERAÇÃO (em Verilog)	OPCODE/FUNÇÃO (Hexa)
Add	add	R R[rd] = R[rs] + R[rt]	(1) 0 / 20 _{hex}
Add Immediate	addi	I R[rt] = R[rs] + SignExtImm	(1,2) 8 _{hex}
Add Imm. Unsigned	addiu	I R[rt] = R[rs] + SignExtImm	(2) 9 _{hex}
Add Unsigned	addu	R R[rd] = R[rs] + R[rt]	0/21 _{hex}
And	and	R R[rd] = R[rs] & R[rt]	0/24 _{hex}
And Immediate	andi	I R[rt] = R[rs] & ZeroExtImm	(3) c _{hexa}
Branch On Equal	beq	I if(R[rs]==R[rt]) PC=PC+4+BranchAddr	(4) 4 _{hexa}
Branch On Not Equal	bne	I if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	(4) 5 _{hexa}
Jump	j	J PC=JumpAddr	(5) 2 _{hexa}
Jump And Link	jal	J R[31]=PC+8; PC=JumpAddr	(5) 3 _{hexa}
Jump Register	jr	R PC=R[rs]	0 / 08 _{hexa}
Load Byte Unsigned	lbu	I R[rt]={24'b0,M[R[rs]+SignExtImm](7:0)}	(2) 24 _{hexa}
Load Halfword Unsigned	lhu	I R[rt]={16'b0,M[R[rs]+SignExtImm](15:0)}	(2) 25 _{hexa}
Load Linked	ll	I R[rt]=M[R[rs]+SignExtImm] (2,7)	30 _{hexa}
Load Upper Imm.	lui	I R[rt]={imm, 16'b0}	f _{hexa}
Load Word	lw	I R[rt]=M[R[rs]+SignExtImm]	(2) 23 _{hexa}
Nor	nor	R R[rd]=~(R[rs] R[rt])	0 / 27 _{hexa}
Or	or	R R[rd]=R[rs] R[rt]	0 / 25 _{hexa}
Or Immediate	ori	I R[rt]=R[rs] ZeroExtImm	(3) d _{hexa}
Set Less Than	slt	R R[rd]=(R[rs] < R[rt]) ? 1 : 0	0 / 2a _{hexa}
Set Less Than Imm.	slti	I R[rt]=(R[rs] < SignExtImm) ? 1 : 0	(2) a _{hexa}
Set Less Than Imm. Unsigned	sltiu	I R[rt]=(R[rs] < SignExtImm) ? 1 : 0	(2,6) b _{hexa}
Set Less Than Unsig.	sltu	R R[rd]=(R[rs] < R[rt]) ? 1 : 0	(6) 0 / 2b _{hexa}
Shift Left Logical	sll	R R[rd]=R[rt] << shamt	0/00 _{hexa}
Shift Right Logical	srl	R R[rd]=R[rt] >>> shamt	0/02 _{hexa}
Store Byte	sb	I M[R[rs]+SignExtImm](7:0) = R[rt](7:0)	(2) 28 _{hexa}
Store Conditional	sc	I M[R[rs]+SignExtImm] = R[rt]; R[rt] = (atomic) ? 1 : 0	(2,7) 38 _{hexa}
Store Halfword	sh	I M[R[rs]+SignExtImm](15:0) = R[rt](15:0)	(2) 29 _{hexa}
Store Word	sw	I M[R[rs]+SignExtImm] = R[rt]	(2) 2b _{hexa}
Subtract	sub	R R[rd]=R[rs] - R[rt]	(1) 0/22 _{hexa}
Subtract Unsigned	subu	R R[rd]=R[rs] - R[rt]	0/23 _{hexa}

- (1) Pode causar exceção de overflow
- (2) SignExtImm = { 16{imediato[15]}, imediato }
- (3) ZeroExtImm = { 16{lb'0}, imediato }
- (4) BranchAddr = { 14{imediato[15]}, imediato, 2'b0 }
- (5) JumpAddr = { PC+4[31:28], endereço, 2'b0 }
- (6) Operandos considerados números sem sinal (versus complemento de 2)
- (7) Par indivisível testar&definir; R[rt] = 1 se par indivisível, 0 se não indivisível

FORMATOS BÁSICOS DE INSTRUÇÃO

R	opcode	rs	rt	rd	shamt	função
	31	26 25	21 20	16 15	11 10	6 5
I	opcode	rs	rt	imediato		
	31	26 25	21 20	16 15	0	
J	opcode	endereço				
	31	26 25	0			

CONJUNTO BÁSICO DE INSTRUÇÕES ARITMÉTICAS ② OPCODE/FMT/FT/FUNÇÃO

NOME, MNEMÔNICO	FOR-	OPERAÇÃO	OPCODE/FMT/FT/FUNÇÃO (Hexa)
Branch On FP True	bclt	FI if(FPcond)PC = PC + 4 + Branch-Addr	(4) 11/8/0/--
Branch On FP False	bclt	FI if(! FPcond)PC = PC + 4 + Branch-Addr	(4) 11/8/0/--
Divide	div	R Lo = R[rs]/R[rt]; Hi = R[rs]%R[rt]	0/--/--/1a
Divide Unsigned	divu	R Lo = R[rs]/R[rt]; Hi = R[rs]%R[rt]	(6) 0/--/--/1b
FP Add Single	add.s	FR F[fd] = F[fs] + F[ft]	11/10/--/0
FP Add Double	add.d	FR {F[fd],F[fd+1]} = {F[fs],F[fs+1]} + {F[ft],F[ft+1]}	11/11/--/0
FP Compare Single	c.x.s*	FR FPcond = (F[fs] op F[ft]) ? 1 : 0	11/10/--/y
FP Compare Double	c.x.d*	FR FPcond = ((F[fs],F[fs+1]) op {F[ft],F[ft+1]}) ? 1 : 0	11/11/--/y
* (x é eq, lt ou le) (op pe ==, < ou <=) (y é 32, 3c ou 3e)			
FP Divide Single	div.s	FR F[fd] = F[fs] / F[ft]	11/10/--/3
FP Divide Double	div.d	FR {F[fd],F[fd+1]} = {F[fs],F[fs+1]}/ {F[ft],F[ft+1]}	11/11/--/3
FP Multiply Single	mul.s	FR F[fd] = F[fs] * F[ft]	11/10/--/2
FP Multiply Double	mul.d	FR {F[fd],F[fd+1]} = {F[fs],F[fs+1]} * {F[ft],F[ft+1]}	11/11/--/2
FP Subtract Single	sub.s	FR F[fd] = F[fs] - F[ft]	11/10/--/1
FP Subtract Double	sub.d	FR {F[fd],F[fd+1]} = {F[fs],F[fs+1]} - {F[ft],F[ft+1]}	11/11/--/1
Load FP Single	lwc1	I F[rt] = M[R[rs] + SignExtImm]	(2) 31/--/--/1
Load FP Double	ldc1	I F[rt] = M[R[rs] + SignExtImm]; F[rt+1] = M[R[rs] + SignExtImm + 4]	(2) 35/--/--/1
Move From Hi	mghi	R R[rd] = Hi	0/--/--/10
Move From Lo	mflr	R R[rd] = Lo	0/--/--/12
Move From Control	mfc0	R R[rd] = CR[rs]	10/0/--/0
Multiply	mult	R {Hi,Lo} = R[rs] * R[rt]	0/--/--/18
Multiply Unsigned	multu	R {Hi,Lo} = R[rs] * R[rt]	(6) 0/--/--/19
Shift Right Arith.	sra	R R[rd] = R[rt] >> shamt	0/--/--/3
Store FP Single	swc1	I M[R[rs] + SignExtImm] = F[rt]	(2) 39/--/--/1
Store FP Double	sdc1	I M[R[rs] + SignExtImm] = F[rt]; M[R[rs] + SignExtImm + 4] = F[rt + 1]	(2) 3d/--/--/1

FORMATOS DE INSTRUÇÃO DE PONTO FLUTUANTE

FR	opcode	fnt	ft	fs	fd	função
	31	26 25	21 20	16 15	11 10	6 5
FI	opcode	fnt	ft	imediato		
	31	26 25	21 20	16 15		

CONJUNTO DE PSEUDOINSTRUÇÕES

NOME	MNEMÔNICO	OPERAÇÃO
Branch Less Than	blt	if(R[rs] < R[rt]) PC = Label
Branch Greater Than	bgt	if(R[rs] > R[rt]) PC = Label
Branch Less Than or Equal	bte	if(R[rs] <= R[rt]) PC = Label
Branch Greater Than or Equal	bge	if(R[rs] >= R[rt]) PC = Label
Load Immediate	li	R[rd] = imediato
Move	move	R[rd] = R[rs]

NOME DE REGISTRADOR, NÚMERO, USO, CONVENÇÃO DE CHAMADA

NOME	NÚMERO	USO	PRESERVADO EM UMA CHAMADA?
\$zero	0	O valor constante 0	N.A.
\$at	1	Temporário do assembler	Não
\$v0-\$v1	2-3	Valores para resultados de função e avaliação de expressão	Não
\$a0-\$a3	4-7	Argumentos	Não
\$t0-\$t7	8-15	Temporários	Não
\$s0-\$s7	16-23	Temporários salvos	Sim
\$t8-\$t9	24-25	Temporários	Não
\$k0-\$kl	26-27	Reservado para o kernel do sistema	Não
\$gp	28	Ponteiro global	Sim
\$sp	29	Ponteiro de pilha	Sim
\$fp	30	Ponteiro de frame	Sim
\$ra	31	Endereço de retorno	Sim

OPCODES, CONVERSÃO DE BASE, SÍMBOLOS ASCII ③

Op- code	(1) Função	(2) Função MIPS	Binário	Deci- mal	Hexa- mal	Car- rac- tere	Deci- mal	Hexa- mal	Carac- tere
(31:26)	(5:0)	MIPS			ASCII				ASCII
(1)	sll	add.f	00 0000	0	0	NUL	64	40	@
		sub.f	00 0001	1	1	SOH	65	41	A
j	srl	mul.f	00 0010	2	2	STX	66	42	B
jal	sra	div.f	00 0011	3	3	ETX	67	43	C
beq	sllv	sqrt.f	00 0100	4	4	EOT	68	44	D
bne		abs.f	00 0101	5	5	ENQ	69	45	E
blez	srlv	mov.f	00 0110	6	6	ACK	70	46	F
bgtz	sra	neg.f	00 0111	7	7	BEL	71	47	G
addi	j r		00 1000	8	8	BS	72	48	H
addiu	jalr		00 1001	9	9	HT	73	49	I
siti	movz		00 1010	10	a	LF	74	4a	J
sltiu	movn		00 1011	11	b	VT	75	4b	K
andi	syscall	round.w.f	00 1100	12	c	FF	76	4c	L
ori	break	trunc.w.f	00 1101	13	d	CR	77	4d	M
xori		ceil.w.f	00 1110	14	e	SO	78	4e	N
lui	sync	floor.w.f	00 1111	15	f	SI	79	4f	O
(2)	mfhi		01 0000	16	10	DLE	80	50	P
	mthi		01 0001	17	11	DC1	81	51	Q
	mflo	movz.f	01 0010	18	12	DC2	82	52	R
	mtlo	movn.f	01 0011	19	13	DC3	83	53	S
			01 0100	20	14	DC4	84	54	T
			01 0101	21	15	NAK	85	55	U
			01 0110	22	16	SYN	86	56	V
			01 0111	23	17	ETB	87	57	W
	mult		01 1000	24	18	CAN	88	58	X
	multu		01 1001	25	19	EM	89	59	Y
	div		01 1010	26	1a	SUB	90	5a	Z
	divu		01 1011	27	1b	ESC	91	5b	[
			01 1100	28	1c	FS	92	5c	\
			01 1101	29	1d	GS	93	5d]
			01 1110	30	1e	RS	94	5e	^
			01 1111	31	1f	US	95	5f	_
lb	add	cvt.s.f	10 0000	32	20	Space	96	60	,
lh	addu	cvt.d.f	10 0001	33	21	!	97	61	a
lwl	sub		10 0010	34	22	"	98	62	b
lwr	subu		10 0011	35	23	#	99	63	c
lbu	and	cvt.w.f	10 0100	36	24	\$	100	64	d
lhu	or		10 0101	37	25	%	101	65	e
lwr	xor		10 0110	38	26	&	102	66	f
	nor		10 0111	39	27	'	103	67	g
sb			10 1000	40	28	(104	68	h
sh			10 1001	41	29)	105	69	i
swl	slt		10 1010	42	2a	*	106	6a	j
sw	sltu		10 1011	43	2b	+	107	6b	k
			10 1100	44	2c	,	108	6c	l
			10 1101	45	2d	-	109	6d	m
swr			10 1110	46	2e	.	110	6e	n
cache			10 1111	47	2f	/	111	6f	o
ll	tge	c.f.f	11 0000	48	30	0	112	70	p
lwl	tgeu	c.un.f	11 0001	49	31	1	113	71	q
lwc2	tlr	c.eq.f	11 0010	50	32	2	114	72	r
pref	tlr	c.ueq.f	11 0011	51	33	3	115	73	s
	teq	c.olt.f	11 0100	52	34	4	116	74	t
ldc1		c.ult.f	11 0101	53	35	5	117	75	u
ldc2	tne	c.ole.f	11 0110	54	36	6	118	76	v
		c.ule.f	11 0111	55	37	7	119	77	w
sc		c.sf.f	11 1000	56	38	8	120	78	x
swc1		c.ngle.f	11 1001	57	39	9	121	79	y
swc2		c.seq.f	11 1010	58	3a	:	122	7a	z
		c.ngl.f	11 1011	59	3b	;	123	7b	{
		c.ltr.f	11 1100	60	3c	<	124	7c	
sdcl		c.nge.f	11 1101	61	3d	=	125	7d	}
sdcl		c.le.f	11 1110	62	3e	>	126	7e	~
		c.ngt.f	11 1111	63	3f	?	127	7f	DEL

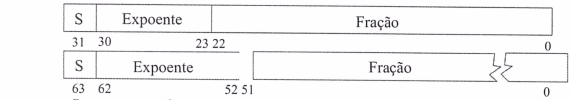
- (1) opcode(31:26) == 0
(2) opcode(31:26) == 17_{dec} (11_{hexa}); se fmt(25:21) == 16_{dec} (10_{hexa}) f = s (simples);
se fmt(25:21) == 17_{dec} (11_{hexa}) f = d (dupla)

PADRÃO DE PONTO FLUTUANTE IEEE 754

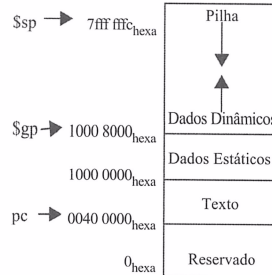
$$(-1)^s \times (1 + \text{Fração}) \times 2^{(\text{Expoente} - \text{Bias})}$$

onde Bias de Precisão Simples = 127,
Bias de Precisão Dupla = 1023

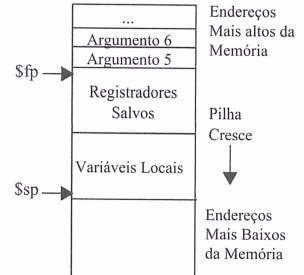
Formatos IEEE de Precisão Simples e Dupla:



ALOCÇÃO DE MEMÓRIA



FRAME DA PILHA

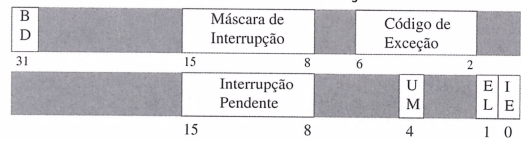


ALINHAMENTO DE DADOS

Dupla Word							
Word				Word			
Halfword		Halfword		Halfword		Halfword	
Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte
0	1	2	3	4	5	6	7

Valor dos três bits menos significativos do endereço de byte (Big Endian)

REGISTRADORES DE CONTROLE DE EXCEÇÃO: CAUSA E STATUS



BD = Branch Delay (desvio), UM = User Mode (modo usuário), EL = Exception Level (nível de exceção), IE = Interrupt Enable (interrupção habilitada)

CÓDIGOS DE EXCEÇÃO

Número	Nome	Causa da Exceção	Número	Nome	Causa da Exceção
0	Int	Interrupção (hardware)	9	Bp	Exceção de Ponto de Interrupção
4	AdEL	Exceção de Erro de Endereço (load ou busca de instrução)	10	RI	Exceção de Instrução Revertida
5	AdES	Exceção de Erro de Endereço (store)	11	CpU	Co-processor Não Implementado
6	IBE	Erro de Barramento ou Busca de Instrução	12	Ov	Exceção de Overflow Aritmético
7	DBE	Erro de Barramento no Load ou Store	13	Tr	Trap
8	Sys	Exceção de Syscall	15	FPE	Exceção de Ponto Flutuante

PREFIXOS DE TAMANHO

	PREFI- XO	SÍM- BOLO	TAMA- NHO	PREFI- XO	SÍM- BOLO	TAMA- NHO	PREFI- XO	SÍM- BOLO	TAMA- NHO	PREFI- FIXO	SÍM- BOLO
10 ³	Kilo-	K	2 ¹⁰	Kibi-	Ki	10 ¹⁵	Peta-	P	2 ⁵⁰	Pebi-	Pi
10 ⁶	Mega-	M	2 ²⁰	Mebi-	Mi	10 ¹⁸	Exa-	E	2 ⁶⁰	Exbi-	Ei
10 ⁹	Giga-	G	2 ³⁰	Gibi-	Gi	10 ²¹	Zetta-	Z	2 ⁷⁰	Zebi-	Zi
10 ¹²	Tera-	T	2 ⁴⁰	Tebi-	Ti	10 ²⁴	Yotta-	Y	2 ⁸⁰	Yobi-	Yi