Arithmetic Register Intructions

Name	Opcode/Funct	Format	Mnemonic	Usage
Add	0/20	R	add	add $r $o1 $o2 => r = o1 + o2
Add Unsigned	0/21	R	addu	addu $r $o1 $o2 => r = o1 + o2 Números são tratados como sem sinal, e não como 2's Complement
Subtract	0/22	R	sub	sub $r $o1 $o2 => r = o1 - o2
Subtract Unsigned	0/23	R	subu	subu $r $o1 $o2 => r = o1 - o2 Números são tratados como sem sinal, e não como 2's Complement
Divide	0/1a	R	div	div $o1 $o2 => $hi = o1 % o2 $low = o1 / o2
Divide Unsigned	0/1b	R	divu	div $o1 $o2 => $hi = o1 % o2 $low = o1 / o2 Números são tratados como sem sinal, e não como 2's Complement
Multiply	0/18	R	mult	mult $o1 $o2 => {$hi, $low} = o1 * o2
Multiply Unsigned	0/19	R	multu	mult $o1 $o2 => {$hi, $low} = o1 * o2 Números são tratados como sem sinal, e não como 2's Complement

Arithmetic Immediate Intructions

Name	Opcode	Format	Mnemonic	Usage
Add Immediate	8	I	addi	addi $r $o1 100 => r = o1 + 100
Subtract Immediate		P	subi	subi $r $o1 100 => r = o1 - 100
Add Immediate Unsigned	9	I	addiu	addiu $r $o1 100 => r = o1 + 100 Números são tratados como sem sinal, e não como 2's Complement
Subtract Immediate Unsigned	-	P	subiu	subi $r $o1 100 => r = o1 - 100 Números são tratados como sem sinal, e não como 2's Complement

Logical Register Intructions

Name	Opcode/Funct	Format	Mnemonic	Usage
And	0/24	R	and	and $r $o1 $o2 => r = o1 && o2
Nor	0/27	R	nor	nor $r $o1 $o2 => r = !(o1 || o2)
Or	0/25	R	or	or $r $o1 $o2 => r = o1 || o2
Xor	0/26	R	xor	xor $r $o1 $o2 => r = o1 ^ o2
Shift Left Logical	0/00	R	sll	Sll $r $o1 2 => r = o1 >> 2
Shift Right Logical	0/02	R	srl	srl $r $o1 2 => r = o1 >> 2

Logical Immediate Intructions

Name	Opcode	Format	Mnemonic	Usage
And Immediate	c	I	andi	andi $r $o1 64 => r = o1 && (bin 64)
Or Immediate	d	I	nori	nori $r $o1 64 => r = o1 || (bin 64)
Xor Immediate	e	I	ori	ori $r $o1 64 => r = o1 ^ (bin 64)

Memory Storing Intructions

Name	Opcode	Format	Mnemonic	Usage
Store Byte	28	I	sb	sb $o1 label => o byte (8 bits) de o1 fica guardado no endereço de “label”
Store Halfword	29	I	sh	sh $o1 label => o byte (16 bits) de o1 fica guardado no endereço de “label”
Store Word	2b	I	sw	sw $o1 label => o byte (32 bits) de o1 fica guardado no endereço de “label”

Memory Loading Intructions

Name	Opcode	Format	Mnemonic	Usage
Load Byte	20	I	lb	lb $o1 label => carrega o byte em “label” para o1
Load Byte Unsigned	24	I	lbu	lbu $o1 label => carrega o byte em “label” para o1 Números são tratados como sem sinal, e não como 2’s Complement
Load Halfword	25	I	lh	lh $o1 label => carrega a halfword em “label” para o1
Load Halfword Unsigned	25	I	lhu	lhu $o1 label => carrega o halfword em “label” para o1 Números são tratados como sem sinal, e não como 2’s Complement
Load Word	23	I	lw	lw $o1 label => carrega o word em “label” para o1
Load Upper Immediate	f	I	lui	lui $o1 64 => carrega 64 (em bin) para os primeiros 16 bits de o1
Load Immediate	-	P	li	li $o1 64 => carrega 64 (em bin) para o1
Load Address	-	P	la	la $o1 label => carrega o endereço label para o1

Branching Intructions

Name	Opcode	Format	Mnemonic	Usage
Branch On Equal	4	I	beq	beq $o1 $o2 label => vai para “label” se o1 == o2
Branch On Not Equal	5	I	bne	bne $o1 $o2 label => vai para “label” se o1 != o2
Branch Less Than	-	P	blt	blt $o1 $o2 label => vai para “label” se o1 < o2
Branch Greater Than	-	P	bgt	bgt $o1 $o2 label => vai para “label” se o1 > o2
Branch Less Than or Equal	-	P	ble	ble $o1 $o2 label => vai para “label” se o1 <= o2
Branch Greater Than or Equal	-	P	bge	bge $o1 $o2 label => vai para “label” se o1 >= o2

Moving/Copying Intructions

Name	Opcode/Funct	Format	Mnemonic	Usage
Move	-	P	move	move $o1 $o2 => o1 = o2
Move From Hi	0/10	R	mfhi	mfhi $o1 => o1 = hi
Move From Lo	0/12	R	mflo	mflo $o1 => o1 = lo

Floating Point Arithmetic Intructions

Name	Opcode/fmt/Funct	Format	Mnemonic	Usage
FP Add Single	11/10/0	FR	add.s	add.s $fr $fo1 $fo2 => fr = fo1 + fo2
FP Subtract Single	11/10/1	FR	sub.s	sub.s $fr $fo1 $fo2 => fr = fo1 - fo2
FP Multiply Single	11/10/2	FR	mul.s	mul.s $fr $fo1 $fo2 => fr = fo1 * fo2
FP Divide Single	11/10/3	FR	div.s	div.s $fr $fo1 $fo2 => fr = fo1 / fo2
FP Add Double	11/11/0	FR	add.d	add.d $fr $fo1 $fo2 => fr = fo1 + fo2 Registos de precisão dupla
FP Subtract Double	11/11/1	FR	sub.d	sub.d $fr $fo1 $fo2 => fr = fo1 - fo2 Registos de precisão dupla
FP Multiply Double	11/11/2	FR	mul.d	mul.d $fr $fo1 $fo2 => fr = fo1 * fo2 Registos de precisão dupla
FP Divide Double	11/11/3	FR	div.d	div.d $fr $fo1 $fo2 => fr = fo1 / fo2 Registos de precisão dupla

Floating Point Branching and Comparing Intructions

Name	Opcode/fmt/ft/Funct	Format	Mnemonic	Usage
Branch On FP True	11/8/1/-	FI	bc1t	bc1t 0 label => vai para label se a flag 0 for true
Branch On FP Flase	11/8/0/-	FR	bc1f	bc1f 0 label => vai para label se a flag 0 for false
FP Compare Single Equal	11/10/-/32	FR	c.qe.s	c.eq.s 0 $fo1 $fo2 => se fo1 == fo2 a flag 0 fica verdadeira
FP Compare Single Less Than	11/10/-/3c	FR	c.lt.s	c.lt.s 0 $fo1 $fo2 => se fo1 < fo2 a flag 0 fica verdadeira
FP Compare Single Less Than or Equal	11/10/-/3e	FR	c.le.s	c.le.s 0 $fo1 $fo2 => se fo1 <= fo2 a flag 0 fica verdadeira
FP Compare Double Equal	11/11/-/32	FR	c.qe.d	c.eq.d 0 $fo1 $fo2 => se fo1 == fo2 a flag 0 fica verdadeira Registos de precisão dupla
FP Compare Double Less Than	11/11/-/3c	FR	c.lt.d	c.lt.d 0 $fo1 $fo2 => se fo1 < fo2 a flag 0 fica verdadeira Registos de precisão dupla
FP Compare Double Less Than or Equal	11/11/-/3e	FR	c.le.d	c.le.d 0 $fo1 $fo2 => se fo1 <= fo2 a flag 0 fica verdadeira Registos de precisão dupla

Floating Point Loading and Storing Intructions

Name	Opcode	Format	Mnemonic	Usage
Load FP Single	31	I	lwc1	lwc1 $fo1 label => carrega label para fo1
Load FP Double	35	I	ldc1	ldc1 $fo1 label => carrega label para fo1
Store FP Single	39	I	swc1	swc1 $fo1 label => guarda fo1 em label
Store FP Double	3d	I	sdc1	sdc1 $fo1 label => guarda fo1 em label

System Calls (Syscalls)

Service	$v0	Arguments	Result
Print Integer	1	integer to print in $a0	
Print Float	2	float to print in $f12	
Print Double	3	double to print in $f12/f13	
Print String	4	string's adderess in $a0	
Read Integer	5		integer read (in $v0)
Read Float	6		float read (in $f0)
Read Double	7		double read (in $f0)
Read String	8	address of buffer in $a0, size of the buffer in $a1	(String gets stored in dynamic memory)
Exit Program	10		