MIPS

1 de mayo de 2017

1. Introducción

MIPS (*Microprocessor without Interlocked Pipeline Stages*, Microprocesador sin Interbloqueos de Segmentación) es un procesador de tipo RISC de 32 bits. Un proyecto original de la Universidad de Stanford liderado por John L. Hennessy. La idea era desarrollar un procesador con un rendimiento mejorado utilizando la técnica de segmentación.

2. Registros

Nombre	Numero	Uso	Se preserva
$zero	$0	Constante 0	Sí
$at	$1	Reservado ensamblador	No
$v0–$v1	$2–$3	Valores de retorno y expresiones	No
$a0–$a3	$4–$7	Argumentos de funciones	No
$t0–$t7	$8–$15	Temporales	No
$s0–$s7	$16–$23	Temporales preservados	Sí
$t8–$t9	$24–$25	Temporales	No
$k0–$k1	$26–$27	Reservados para el Núcleo del S.O.	No
$gp	$28	Puntero Global	Sí
$fp	$29	Puntero Base Reg. Act.	Sí
$sp	$30	Puntero de Pila	Sí
$ra	$31	Dirección de Retorno	No

3. Formatos de Instrucción

3.1. Tipo R

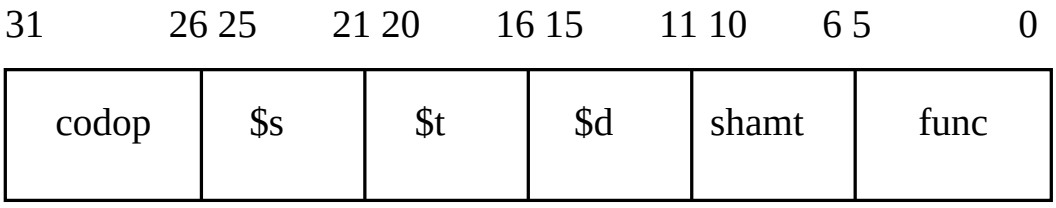

Figura 1: formato de instrucción R.

3.2. Tipo I

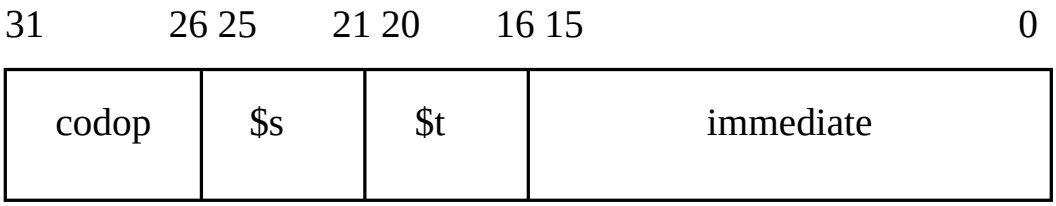

Figura 2: formato de instrucción I.

3.3. Tipo J

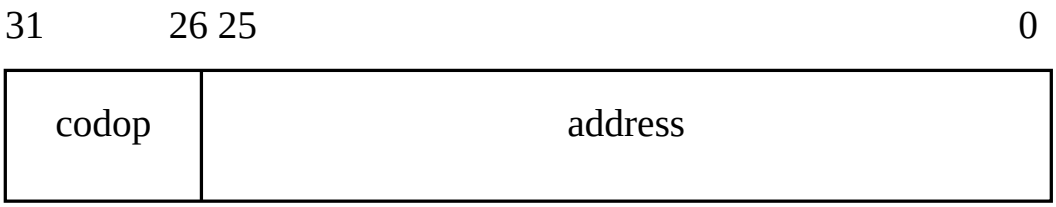

Figura 3: formato de instrucción J.

4. Instrucciones

4.1. Carga y almacenaje en memoria

Nombre	Sintaxis	Formato	Código	Semántica
Load Byte	LB $t,I($s)	I	0x20	$t = M($s+I) (signed)
Load HalfWord	LH $t,I($s)	I	0x21	$t = M($s+I) (signed)
Load Word Left	LWL $t,I($s)	I	0x22	$t = M($s+I) (left part)
Load Word	LW $t,I($s)	I	0x23	$t = M($s+I)
Load Byte Unsigned	LBU $t,I($s)	I	0x24	$t = M($s+I) (unsigned)
Load HalfWord Unsigned	LHU $t,I($s)	I	0x25	$t = M($s+I) (unsigned)
Load Word Right	LWR $t,I($s)	I	0x26	$t = M($s+I) (right part)
Store Byte	SB $t,I($s)	I	0x28	M($s + I) = $t
Store HalfWord	SH $t,I($s)	I	0x29	M($s + I) = $t
Store Word Left	SWL $t,I($s)	I	0x2A	M($s + I) = $t (left part)
Store Word	SW $t,I($s)	I	0x2B	M($s + I) = $t
Store Word Right	SWR $t,I($s)	I	0x2E	M($s + I) = $t (right part)

4.2. Acceso a la ALU

Nombre	Sintaxis	Código	Función	Semántica
Add	add $d,$s,$t	0	0x20	$d = $s+$t
Add Unsigned	addu $d,$s,$t	0	0x21	$d=$s+$t
Substract	sub $d,$s,$t	0	0x22	$d=$s-$t
Substract Unsigned	subu $d,$s,$t	0	0x23	$d=$s-$t
Add Immediate	addi $t,$s,I	0x8	—	$t=$s+I (signed)
Add Immediate Unsigned	addiu $t,$s,I	0x9	—	$t=$s+I (unsigned)
And	and $d,$s,$t	0	0x24	$d = $s and $t
And Immediate	andi $t,$s,I	0xC	—	$t = $s and I (unsigned)
Or	or $d,$s,$t	0	0x25	$d = $s or $t
Or Immediate	ori $t,$s,I	0xD	—	$t = $s or I (unsigned)
Xor	xor $d,$s,$t	0	0x26	$d = $s xor $t
Xor Immediate	xori $t,$s,I	0xE	—	$t = $s and I (unsigned)
Nor	nor $d,$s,$t	0	0x27	$d = $s nor $t
Set on Less Than	slt $d,$s,$t	0	0x2A	$d = $s < $t ? 1 : 0
Set on Less Than Unsigned	sltu $d,$s,$t	0	0x2B	$d = $s < $t ? 1 : 0 (unsigned)
Set on Less Than Immediate	slti $t,$s,I	0xA	—	$t = $s < I ? 1 : 0
Set on Less Than Immediate Unsigned	sltiu $d,$s,$t	0xB	—	$t = $s < I ? 1 : 0 (unsigned)
Load Upper Immediate	lui $t,I	0xF	—	$t = I << 16

4.3. Desplazamientos

Nombre	Sintaxis	Código	Función	Semántica
Shift Left Logic	sll $d,$t,S	0	0	$d = $t << S
Shift Right Logic	srl $d,$t,S	0	0x2	$d = $t >> S
Shift Right Arithmetic	sra $d,$t,S	0	0x3	$d = $t >> S (signed)
Shift Left Logic Variable	sllv $d,$t,$s	0	0x4	$d = $t << $s
Shift Right Logic Variable	srlv $d,$t,$s	0	0x6	$d = $t >> $s
Shift Right Arithmetic Variable	srav $d,$t,$s	0	0x7	$d = $t >> $s (signed)

4.4. Multiplicaciones y divisiones

Nombre	Sintaxis	Código	Función	Semántica
Multiply	mult $s,$t	0	0x18	(HI , LO) = $s * $t
Multiply Unsigned	multu $s,$t	0	0x19	(HI , LO) = $s * $t (unsigned)
Divide	div $s,$t	0	0x1A	LO= $s % $t, HI = $s / $t
Divide Unsigned	divu $s,$t	0	0x1B	LO= $s % $t, HI = $s / $t (unsigned)
Move from HI	mfhi $d	0	0x10	$d = HI (no modifcar HI por 2 ciclos)
Move to HI	mthi $s	0	0x11	HI = $s
Move from LO	mflo $d	0	0x12	$d = LO (no modifcar LO por 2 ciclos)
Move to LO	mfhi $s	0	0x13	LO = $s

4.5. Saltos

Nombre	Sintaxis	Código	Función	Semántica
Branch on Equal	beq $s,$t,I	0x4	–	if($s=$t) PC = PC + 4*I
Branch on Not Equal	bne $s,$t,I	0x5	–	if($s<>$t) PC = PC + 4*I
Jump	j A	(J) 0x2	–	PC[27:0] = A * 4
Jump Register	jr $s	0	0x8	PC = $s
Jump and Link	jal A	(J) 0x3	–	$ra = PC ; PC[27:0] = A*4
Jump and Link Register	jalr $s	0	0x9	$ra = PC ; PC = $s