MIPS Disc1

Tips

- In MIPS, words must start at addresses that are multiples of 4 *alignment restriction*
- Assembler directives
  - .text *indicating succeeding instruction*
  - .data *indicating succeeding data*
  - .word *4 bytes*
  - .align n *align on 2^n byte boundary*
  - .global *indicating global symbol visible in other file*
  - .asciiz *indicating null-terminated string in memory*

Register

Name	Register#	Usage	Preserved on call?
$zero	0	the constant value 0	n.a.
$v0-$v1	2-3	values for results & expr. evaluation	no
$a0-$a3	4-7	arguments	yes
$t0-$t7	8-15	temporaries	no
$s0-$s7	16-23	saved	yes
$t8-$t9	24-25	more temporaries	no
$gp	28	global pointer	yes
$sp	29	stack pointer	yes
$fp	30	frame pointer	yes
$ra	31	return address	yes

Instructions

srl $rd, $rt, shamt	R[$rd] ← R[$rt] >> shamt	Unsigned right shift
sra $rd, $rt, shamt	R[$rd] ← R[$rt] >> shamt	Signed right shift
mfhi $rd	R[$rd] ← HI	
mflo $rd	R[$rd] ← LO	
div $rs, $rt	LO ← R[$rs] / R[$rt] HI ← R[$rs] % R[$rt]	Signed division
slt $rd, $rs, $rt	R[$rd] ← R[$rs] < R[$rt]	Signed comparison

jal address	R[31] ← PC + 8 PC ← {(PC + 4)[31:28], address, 00}	return address

lui $rt, imm	R[$rt] ← {(imm)[15:0], 0 × 16}	
ori $rt, $rs, imm	R[$rt] ← R[$rs] | {0 × 16, imm}	
lw $rt, imm($rs)	R[$rt] ← Mem4B(R[$rs]+SignExt16b(imm))	Computed address must be a multiple of 4
sw $rt, imm($rs)	Mem4B(R[$rs] + SignExt16b(imm)) ← R[$rt]	Computed address must be a multiple of 4

Links

- http://www-inst.eecs.berkeley.edu/~cs61c/resources/MIPS_help.html

Keep in Mind

- Address or Value ??
- Register or Memory ??
- Signed or Unsigned ??
- Word or Half Word or Byte ??

Try

- int* arr = {1,2,3,4,5,6,0} *lie in memory, the value of* arr *be a multiple of 4 and stored in register* $s0

```
lw $t0, 12($s0)
add $t1, $t0, $s0
sw $t0, 4($t1)
```

```
addiu $s1, $s0, 10
addiu $t0, $0, 6
sw $t0, 2($s1)
```

```
addiu $s1, $s0, 27
lh $t0, -3($s1)
```

```
addiu $s1, $s0, 24
lh $t0, -3($s1)
```

```
addiu $t0, $0, 12
sw $t0, 6($s0)
```

Try

- int* arr = {1,2,3,4,5,6,0} *lie in memory, the value of* arr *be a multiple of 4 and stored in register* $s0

```
lw $t0, 12($s0)
add $t1, $t0, $s0
sw $t0, 4($t1)  // arr[2] <- 4;
```

```
addiu $s1, $s0, 10
addiu $t0, $0, 6
sw $t0, 2($s1)  // arr[3] <- 6;
```

```
addiu $s1, $s0, 27
lh $t0, -3($s1)   // $t0 <- 0;
```

```
addiu $s1, $s0, 24
lh $t0, -3($s1)   // alignment error;
```

```
addiu $t0, $0, 12
sw $t0, 6($s0)  // alignment error;
```

Try

$s0 < $s1	$s0 <= $s1	$s0 > 1	$s0 >= 1
slt, bne	slt, beq	sltiu, beq	bgtz

Try

$s0 < $s1	$s0 <= $s1	$s0 > 1	$s0 >= 1
slt $t0, $s0, $s1 bne $t0, $0, label	slt $t0, $s1, $s0 beq $t0, $0, label	sltiu $t0, $s0, 2 beq $t0, $0, label	bgtz $s0, label

Try

C	MIPS

// $s0 -> a, $s1 -> b
// $s2 -> c, $s3 -> z
int a = 4, b = 5, c = 6, z;
z = a + b + c + 10;
	addiu $s0, $0, 4
	addiu $s1, $0, 5
	addiu $s2, $0, 6

Try

C	MIPS

// $s0 -> a, $s1 -> b
// $s2 -> c, $s3 -> z
int a = 4, b = 5, c = 6, z;
z = a + b + c + 10;
	addiu $s0, $0, 4
	addiu $s1, $0, 5
	addiu $s2, $0, 6
	
	addu $s3, $s0, $s1
	addu $s3, $s3, $s2
	addiu $s3, $s3, 10

Try

C	MIPS

// $s0 -> int * p = intArr;
// $s1 -> a;
*p = 0;
int a = 2;
p[1] = p[a] = a;
	la $s0 intArr

Try

C	MIPS

// $s0 -> int * p = intArr;
// $s1 -> a;
*p = 0;
int a = 2;
p[1] = p[a] = a;
	la $s0 intArr
	sw $0, 0($s0)
	addiu $s1, $0, 2
	sw $s1, 4($s0)
	sll $t0, $s1, 2
	add $t0, $t0, $s0
	sw $s1, 0($t0)

Try

C	MIPS

// $s0 -> a, $s1 -> b
int a = 5, b = 10;
if (a + a == b) {
  a = 0;
} else {
  b = a - 1;
}	

Try

C	MIPS

// $s0 -> a, $s1 -> b
int a = 5, b = 10;
if (a + a == b) {
  a = 0;
} else {
  b = a - 1;
}
	addiu $s0, $0, 5
	addiu $s1, $0, 10
	addu $t0, $s0, $s0
	bne $t0, $s1, else
	xor $s0, $0, $0
	j exit
	else:
	addiu $s1, $s0, -1
	exit:

Try

C	MIPS

	addiu $s0, $0, 0
	addiu $s1, $0, 1
	addiu $t0, $0, 30
	loop:
	beq $s0, $t0, exit
	addu $s1, $s1, $s1
	addiu $s0, $s0, 1
	j loop
	exit:

Try

C	MIPS

// computes s1 = 2^30
s1 = 1;
for(s0=0;s0<30;s0++) {
  s1 *= 2;
}
	addiu $s0, $0, 0
	addiu $s1, $0, 1
	addiu $t0, $0, 30
	loop:
	beq $s0, $t0, exit
	addu $s1, $s1, $s1
	addiu $s0, $s0, 1
	j loop
	exit: