#5 : MIPS Programming I

Computer Architecture 2020/2021

Ricardo Rocha

Computer Science Department, Faculty of Sciences, University of Porto

Arithmetic Instructions

add $s1, $s2, $s3	$s1 = $s2 + $s3	(add)
addu $s1, $s2, $s3	$s1 = $s2 + $s3	(add unsigned, no overflow)
addi $s1, $s2, 20	$s1 = $s2 + 20	(add immediate, sign-extend)
addiu $s1, $s2, 20	$s1 = $s2 + 20	(add immediate, no overflow)
sub $s1, $s2, $s3	$s1 = $s2 – $s3	(subtract)
mul $s1, $s2, $s3	$s1 = $s2 * $s3	(multiply)

Logical Instructions

and $s1, $s2, $s3	$s1 = $s2 & $s3	(and, bit-by-bit)
andi $s1, $s2, 20	$s1 = $s2 & 20	(and immediate)
or $s1, $s2, $s3	$s1 = $s2 | $s3	(or)
nor $s1, $s2, $s3	$s1 = ~ ($s2 | $s3)	(nor)
sll $s1, $s2, 10	$s1 = $s2 << 10	(shift left logical)
srl $s1, $s2, 10	$s1 = $s2 >> 10	(shift right logical)

Load Instructions

lw \$s1, 20(\$s2)	\$s1 = Mem[\$s2 + 20]	(load word, from memory)
lh \$s1, 20(\$s2)	\$s1 = Mem[\$s2 + 20]	(load half word, sign-extend)
lhu \$s1, 20(\$s2)	\$s1 = Mem[\$s2 + 20]	(load half word, zero-extend)
lb \$s1, 20(\$s2)	\$s1 = Mem[\$s2 + 20]	(load byte, sign-extend)
lbu \$s1, 20(\$s2)	\$s1 = Mem[\$s2 + 20]	(load byte, zero-extend)
li \$s1, 20	\$s1 = 20	(load immediate)
la \$s1, L	\$s1 = L	(load address)

Store Instructions

sw $s1, 20($s2)	Mem[$s2 + 20] = $s1	(store word, to memory)
sh $s1, 20($s2)	Mem[$s2 + 20] = $s1	(store half word)
sb $s1, 20($s2)	Mem[$s2 + 20] = $s1	(store byte)

Branch Instructions

beq $s1, $s2, 25	if ($s1 == $s2) go to (PC+4+100)	(branch on equal)
beq $s1, $s2, L	if ($s1 == $s2) go to L	(branch on equal)
bne $s1, $s2, L	if ($s1 != $s2) go to L	(branch on not equal)
blt $s1, $s2, L	if ($s1 < $s2) go to L	(branch on less than)
bgt $s1, $s2, L	if ($s1 > $s2) go to L	(branch on greater than)
ble $s1, $s2, L	if ($s1 <= $s2) go to L	(branch on less than or equal)

slt $s1, $s2, $s3	if ($s2 < $s3) $s1 = 1
	else $s1 = 0
		(set on less than,
		for use with beq/bne)

slti $s1, $s2, 20	if ($s2 < 20) $s1 = 1
	else $s1 = 0
		(set on less than immediate)

Jump Instructions

j 2500	go to 10000	(jump to target address)
j L	go to L	(jump to target address)
jal L	$ra = PC+4; go to L	(jump and link, for procedure call)
jr $ra	go to $ra	(jump register, for procedure return)

Pseudo-Instructions

Most assembler instructions represent machine instructions one-to-one. To **simplify programming**, the assembler can also treat common variations of machine instructions as if they were instructions in their own right. Such instructions are called **pseudo-instructions**. The hardware need not implement the pseudo-instructions and register $at (assembler temporary) is reserved for this purpose.

```
li $s1, 20          →   addiu $s1, $zero, 20
move $t0, $t1       →   addu $t0, $zero, $t1
blt $s1, $s2, L     →   slt $at, $s1, $s2
                        bne $at, $zero, L
```