MIPS: Repertoire d'instructions

Instructions arithmétiques et logiques

Instruction	Example	Meaning
add	`add $s1, $s2, $s3`	$s1 = $s2 + $s3
add w/o overflow	`addu $s1, $s2, $s3`	$s1 = $s2 + $s3
add immediate	`addi $s1, $s2, 10`	$s1 = $s2 + 10
add imm. w/o overflow	`addiu $s1, $s2, 10`	$s1 = $s2 + 10
AND	`and $s1, $s2, $s3`	$s1 = $s2 ∧ $s3
AND immediate	`andi $s1, $s2, 0xFD`	$s1 = $s2 ∧ 0xFD
NOR	`nor $s1, $s2, $s3`	$s1 = ¬ ($s2 ∨ $s3)
NOT*	`not $s1, $s2`	$s1 = ¬ $s2
OR	`or $s1, $s2, $s3`	$s1 = $s2 ∨ $s3
OR immediate	`ori $s1, $s2, 0xFD`	$s1 = $s2 ∧ 0xFD
shift left logical	`sll $s1, $s2, 4`	$s1 = $s2 << 4
shift left logical var.	`sllv $s1, $s2, $s3`	$s1 = $s2 << $s3
shift right logical	`srl $s1, $s2, 4`	$s1 = $s2 >> 4
shift right logical var.	`srlv $s1, $s2, $s3`	$s1 = $s2 >> $s3
subtract	`sub $s1, $s2, $s3`	$s1 = $s2 − $s3
subtract w/o overflow	`subu $s1, $s2, $s3`	$s1 = $s2 − $s3
subtract immediate*	`subi $s1, $s2, 10`	$s1 = $s2 − 10
exclusive OR	`xor $s1, $s2, $s3`	$s1 = $s2 ⊕ $s3
exclusive OR imm.	`xori $s1, $s2, 0xFD`	$s1 = $s2 ⊕ 0xFD

Instructions de comparaison

Instruction	Example	Meaning
set on less than	`slt $s1, $s2, $s3`	if ($s2 < $s3) $s1 = 1; else $s1 = 0
set on less than unsgn.	`sltu $s1, $s2, $s3`	if ($s2 < $s3) $s1 = 1; else $s1 = 0
set on less than imm.	`slti $s1, $s2, 10`	if ($s2 < 10) $s1 = 1; else $s1 = 0
slt immediate unsgn.	`sltiu $s1, $s2, 10`	if ($s2 < 10) $s1 = 1; else $s1 = 0

Instructions de saut conditionnel

Instruction	Example	Meaning
branch on equal	`beq $s1, $s2, L`	if ($s1 == $s2) go to L
branch on not equal	`bne $s1, $s2, L`	if ($s1 != $s2) go to L
branch on greater than equal zero	`bgez $s1, L`	if ($s1 ≥ 0) go to L
branch on greater than zero	`bgtz $s1, L`	if ($s1 > 0) go to L
branch on less than equal zero	`blez $s1, L`	if ($s1 ≤ 0) go to L
br. on less than zero	`bltz $s1, L`	if ($s1 < 0) go to L
branch on greater than equal zero and link	`bgezal $s1, L`	if ($s1 ≥ 0) $ra = PC + 4; go to L
branch on less than zero and link	`bltzal $s1, L`	if ($s1 < 0) $ra = PC + 4; go to L

Instructions de saut inconditionnel

Instruction	Example	Meaning
jump	`j 0x2500`	go to addr 10,000
jump and link	`jal 0x2500`	$ra = PC + 4; go to addr 10,000
jump and link register	`jalr $s1, $s2`	$s2 = PC + 4; go to $s1
jump register	`jr $ra`	go to $ra

Instructions d'accès à la mémoire

Instruction	Example	Meaning
load byte (sign-extended)	`lb $s1, 96($s2)`	$s1 = Memory[$s2 + 96]
load unsigned byte	`lbu $s1, 96($s2)`	$s1 = Memory[$s2 + 96]
load halfword (sign-extended)	`lh $s1, 96($s2)`	$s1 = Memory[$s2 + 96]
load unsigned halfword	`lhu $s1, 96($s2)`	$s1 = Memory[$s2 + 96]
load word	`lw $s1, 96($s2)`	$s1 = Memory[$s2 + 96]
store byte	`sb $s1, 96($s2)`	Memory[$s2 + 96] = $s1
store halfword	`sh $s1, 96($s2)`	Memory[$s2 + 96] = $s1
store word	`sw $s1, 96($s2)`	Memory[$s2 + 96] = $s1

Registres

Name	Register number	Usage
$zero	0	the constant value 0
$at	1	reserved for the assembler
$v0–$v1	2–3	values for results and expression evaluation
$a0–$a3	4–7	arguments
$t0–$t7	8–15	temporaries
$s0–$s7	16–23	saved
$t8–$t9	24–25	more temporaries
$k0–$k1	26–27	reserved for the operating system
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address