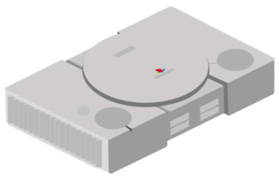
pikuma.com



MIPS Instruction Set

- simplified -

Data Transfer Instructions

Instruction	Example	Description
li $reg, *imm*	**li** $t0, 0x100	Loads an Immediate *(constant)* value into a register.
la $reg, *label*	**la** $a1, *Label*	Loads the Address that a label points to into a register.
lui $reg0, *imm*	**lui** $a0, 0x1F80	Loads Upper Immediate loads a value in the upper part of a register [0x**UUUU**——]
move $reg0, $reg1	**move** $a0, $s1	Copies the value from one register to another.

Load Instructions (From Memory)

Instruction	Example	Description
lw $reg, offset(baseaddr)	**lw** $t1, 0x0060($t0)	Loads Word (32 bits) from an address, where the final address is an immediate *offset* from a baseaddr that must be stored in a register.
lh $reg, offset(baseaddr)	**lh** $t1, 0x0060($t0)	Loads Half (16 bits) from an address, where the final address is an immediate *offset* from a baseaddr that must be stored in a register.
lb $reg, offset(baseaddr)	**lb** $t1, 0x0060($t0)	Loads Byte (8 bits) from an address, where the final address is an immediate *offset* from a baseaddr that must be stored in a register.
lwu $reg, offset(baseaddr)	**lwu** $t1, 0x0060($t0)	Loads Word (32 bits) Unsigned from an address, where the final address is an immediate *offset* from a baseaddr that must be stored in a register.
lhu $reg, offset(baseaddr)	**lhu** $t1, 0x0060($t0)	Loads Half (16 bits) Unsigned from an address, where the final address is an immediate *offset* from a baseaddr that must be stored in a register.
lbu $reg, offset(baseaddr)	**lbu** $t1, 0x0060($t0)	Loads Byte (8 bits) Unsigned from an address, where the final address is an immediate *offset* from a baseaddr that must be stored in a register.

Store Instructions (To Memory)

Instruction	Example	Description
sw $reg, offset(baseaddr)	**sw** $t1, 0x0060($t0)	Store Word (32 bits) from a register to an address, where the final address is an immediate *offset* from a baseaddr that must be stored in a register.
sh $reg, offset(baseaddr)	**sh** $t1, 0x0060($t0)	Store Half (16 bits) from a register to an address, where the final address is an immediate *offset* from a baseaddr that must be stored in a register.
sb $reg, offset(baseaddr)	**sb** $t1, 0x0060($t0)	Store Byte (8 bits) from a register to an address, where the final address is an immediate *offset* from a baseaddr that must be stored in a register.
swu $reg, offset(baseaddr)	**swu** $t1, 0x0060($t0)	Store Word (32 bits) Unsigned from a register to an address, where the final address is an immediate *offset* from a baseaddr that must be stored in a register.
shu $reg, offset(baseaddr)	**shu** $t1, 0x0060($t0)	Store Half (16 bits) Unsigned from a register to an address, where the final address is an immediate *offset* from a baseaddr that must be stored in a register.
sbu $reg, offset(baseaddr)	**sbu** $t1, 0x0060($t0)	Store Byte (8 bits) Unsigned from a register to an address, where the final address is an immediate *offset* from a baseaddr that must be stored in a register.

Shift Instructions

Instruction	Example	Description	
sll $a, $b, *imm*	**sll** $t0, $t0, 1	Shift Left Logical by a constant amount of bits.	$a = $b << *imm*
srl $a, $b, *imm*	**srl** $t0, $t0, 2	Shift Right Logical by a constant amount of bits.	$a = $b >>> *imm*
sra $a, $b, *imm*	**sra** $t0, $t0, 1	Shift Right Arithmetic by a constant amount of bits.	$a = $b >> *imm*
sllv $a, $b, $c	**sllv** $t0, $t0, $t1	Shift Left Logical by the amount in the register.	$a = $b << *$c*
srlv $a, $b, $c	**srlv** $t0, $t0, $t2	Shift Right Logical by the amount in the register.	$a = $b >>> *$c*
srav $a, $b, $c	**srav** $t0, $t1, $t2	Shift Right Arithmetic by the amount in the register.	$a = $b >> *$c*

Addition Instructions

Instruction	Example	Description	
add $a, $b, $c	**add** $t0, $t0, $t1	Adds (signed) numbers from registers.	$a = $b + $c
addu $a, $b, $c	**addu** $t0, $t1, $t2	Adds Unsigned numbers from registers.	$a = $b + $c
addi $a, $b, *imm*	**addi** $t0, $t1, 5	Adds an Immediate (signed) number.	$a = $b + *imm*
addiu $a, $b, ***imm***	**addiu** $t0, $t0, 1	Adds Immediate Unsigned numbers.	$a = $b + *imm*

Subtraction Instructions

Instruction	Example	Description	
sub $a, $b, $c	**sub** $t0, $t0, $t1	Subtracts (signed) numbers from registers.	$a = $b - $c
subu $a, $b, $c	**subu** $t0, $t1, $t2	Subtracts Unsigned numbers from registers.	$a = $b - $c

Multiplication Instructions

Instruction	Example	Description
mult $a, $b	**mult** $t0, $t1	Multiply (signed) numbers from two registers.
multu $a, $b	**multu** $t0, $t1	Multiply Unsigned numbers from two registers.

Division Instructions

Instruction	Example	Description
div $a, $b	**div** $t0, $t1	Divide (signed) numbers from two registers.
divu $a, $b	**divu** $t0, $t1	Divide Unsigned numbers from two registers.

Multiply & *Divide* instructions will store their results in two additional regiters, **HI** and **LO**. We can fetch the contents from **HI** and **LO** using the instructions **mfhi** and **mflo**.

mfhi $reg
mflo $reg

Logical Bitwise Instructions

Instruction	Example	Description	
and $a, $b, $c	**and** $t0, $t0, $t1	Logical bitwise AND	$a = $b & $c
andi $a, $b, $c	**andi** $t0, $t1, 0xF000	Logical bitwise AND Immediate	$a = $b & imm
or $a, $b, $c	**or** $t0, $t0, $t1	Logical bitwise OR	$a = $b | $c
ori $a, $b, $c	**ori** $t0, $t1, 0x00FF	Logical bitwise OR Immediate	$a = $b | imm
xor $a, $b, $c	**xor** $t0, $t0, $t1	Logical bitwise Exclusive-OR	$a = $b ^ $c
xori $a, $b, $c	**xori** $t0, $t0, 0xFFFF	Logical bitwise Exclusive-OR Immediate	$a = $b ^ imm

Branch Instructions

Instruction	Example	Description	
beq a, *b*, label	**beq** $t0, $t1, Label	Branch if Equals	if (a == b) *then jump to label*
bne a, *b*, label	**bne** $t0, 100, Label	Branch if Not Equals	if (a != b) *then jump to label*
blt a, *b*, label	**blt** $t0, $t2, Label	Branch if Less Than	if (a < b) *then jump to label*
ble a, *b*, label	**ble** $t0, $t2, Label	Branch if Less or Equals	if (a <= b) *then jump to label*
bgt a, *b*, label	**bgt** $t0, 5, Label	Branch if Greater Than	if (a > b) *then jump to label*
bge a, *b*, label	**bge** $t0, $t2, Label	Branch if Greater or Equals	if (a >= b) *then jump to label*
bltu a, *b*, label	**bltu** $t0, $t2, Label	Branch if Less Than (unsigned)	if (a < b) *then jump to label*
bleu a, *b*, label	**bleu** $t0, $t2, Label	Branch if Less or Equals (unsigned)	if (a <= b) *then jump to label*
bgtu a, *b*, label	**bgtu** $t0, 5, Label	Branch if Greater Than (unsigned)	if (a > b) *then jump to label*
bgeu a, *b*, label	**bgeu** $t0, $t2, Label	Branch if Greater or Equals (unsigned)	if (a >= b) *then jump to label*

b can be either a *register* or an *immediate*

Jump Instructions

Instruction	Example	Description
j label	**j** Loop	Unconditional jump to Label (address)
jal label	**jal** Subroutine	Jump & Link stores the return address in the register **$ra**.
jr $reg	**jr** $t1	Jump to Register jumps to an address in a register
jalr $reg	**jalr** $t0	Jump & Link Register jumps to an address in a register and stores the return address in **$ra**.