MIPS Instruction Set (core)

<table>
<tr><th><i>instruction</i></th><th><i>example</i></th><th><i>meaning</i></th></tr>
<tr><td colspan="3"><b>arithmetic</b></td></tr>
<tr><td>add</td><td>add $1,$2,$3</td><td>$1 = $2 + $3</td></tr>
<tr><td>subtract</td><td>sub $1,$2,$3</td><td>$1 = $2 - $3</td></tr>
<tr><td>add immediate</td><td>addi $1,$2,100</td><td>$1 = $2 + sign_extend(100)</td></tr>
<tr><td>add unsigned</td><td>addu $1,$2,$3</td><td>$1 = $2 + $3</td></tr>
<tr><td>subtract unsigned</td><td>subu $1,$2,$3</td><td>$1 = $2 - $3</td></tr>
<tr><td>add immediate unsigned</td><td>addiu $1,$2,100</td><td>$1 = $2 + sign_extend(100)</td></tr>
<tr><td>set if less than</td><td>slt $1, $2, $3</td><td>if ($2 < $3), $1 = 1 else $1 = 0</td></tr>
<tr><td>set if less than immediate</td><td>slti $1, $2, 100</td><td>if ($2 < sign_extend(100)), $1 = 1 else $1 = 0</td></tr>
<tr><td>set if less than unsigned</td><td>sltu $1, $2, $3</td><td>if ($2 < $3), $1 = 1 else $1 = 0</td></tr>
<tr><td>set if < immediate unsigned</td><td>sltui $1, $2, 100</td><td>if ($2 < 100), $1 = 1 else $1 = 0</td></tr>
<tr><td>multiply</td><td>mult $2,$3</td><td>Hi, Lo = $2 * $3, 64-bit signed product</td></tr>
<tr><td>multiply unsigned</td><td>multu $2,$3</td><td>Hi, Lo = $2 * $3, 64-bit unsigned product</td></tr>
<tr><td>divide</td><td>div $2,$3</td><td>Lo = $2 / $3, Hi = $2 mod $3</td></tr>
<tr><td>divide unsigned</td><td>divu $2,$3</td><td>Lo = $2 / $3, Hi = $2 mod $3, unsigned</td></tr>
<tr><td colspan="3"><b>transfer</b></td></tr>
<tr><td>move from Hi</td><td>mfhi $1</td><td>$1 = Hi</td></tr>
<tr><td>move from Lo</td><td>mflo $1</td><td>$1 = Lo</td></tr>
<tr><td>load upper immediate</td><td>lui $1,100</td><td>$1 = 100 x 2^{16}</td></tr>
<tr><td colspan="3"><b>logic</b></td></tr>
<tr><td>and</td><td>and $1,$2,$3</td><td>$1 = $2 & $3</td></tr>
<tr><td>or</td><td>or $1,$2,$3</td><td>$1 = $2 | $3</td></tr>
<tr><td>and immediate</td><td>andi $1,$2,100</td><td>$1 = $2 & zero_extend(100)</td></tr>
<tr><td>or immediate</td><td>ori $1,$2,100</td><td>$1 = $2 | zero_extend(100)</td></tr>
<tr><td>nor</td><td>nor $1,$2,$3</td><td>$1 = not($2 | $3)</td></tr>
<tr><td>xor</td><td>xor $1, $2, $3</td><td>$1 = $2 ⊕ $3</td></tr>
<tr><td>xor immediate</td><td>xori $1, $2, 255</td><td>$1 = $2 ⊕ zero_extend(255)</td></tr>
<tr><td colspan="3"><b>shift</b></td></tr>
<tr><td>shift left logical</td><td>sll $1,$2,5</td><td>$1 = $2 << 5 (logical)</td></tr>
<tr><td>shift left logical variable</td><td>sllv $1,$2,$3</td><td>$1 = $2 << $3 (logical), variable shift amt</td></tr>
<tr><td>shift right logical</td><td>srl $1,$2,5</td><td>$1 = $2 >> 5 (logical)</td></tr>
<tr><td>shift right logical variable</td><td>srlv $1,$2,$3</td><td>$1 = $2 >> $3 (logical), variable shift amt</td></tr>
<tr><td>shift right arithmetic</td><td>sra $1,$2,5</td><td>$1 = $2 >> 5 (arithmetic)</td></tr>
<tr><td>shift right arithmetic variable</td><td>srav $1,$2,$3</td><td>$1 = $2 >> $3 (arithmetic), variable shift amt</td></tr>
<tr><td colspan="3"><b>memory</b></td></tr>
<tr><td>load word</td><td>lw $1, 1000($2)</td><td>$1 = memory [$2+1000]</td></tr>
<tr><td>store word</td><td>sw $1, 1000($2)</td><td>memory [$2+1000] = $1</td></tr>
<tr><td>load byte</td><td>lb $1, 1002($2)</td><td>$1 = memory[$2+1002] in least sig. byte, sign extended (sign bit extends 24 bits to left)</td></tr>
<tr><td>load byte unsigned</td><td>lbu $1, 1002($2)</td><td>$1 = memory[$2+1002] in least sig. byte, zero extended (padded with 24 zeros to the left)</td></tr>
<tr><td>store byte</td><td>sb $1, 1002($2)</td><td>memory[$2+1002] = $1 (byte modified only)</td></tr>
<tr><td colspan="3"><b>branch</b></td></tr>
<tr><td>branch if equal</td><td>beq $1,$2,100</td><td>if ($1 = $2), PC = PC + 4 + (100*4)</td></tr>
<tr><td>branch if not equal</td><td>bne $1,$2,100</td><td>if ($1 ≠ $2), PC = PC + 4 + (100*4)</td></tr>
<tr><td colspan="3"><b>jump</b></td></tr>
<tr><td>jump</td><td>j 10000</td><td>PC = 10000*4</td></tr>
<tr><td>jump register</td><td>jr $31</td><td>PC = $31</td></tr>
<tr><td>jump and link</td><td>jal 10000</td><td>$31 = PC + 4; PC = 10000*4</td></tr>
</table>