MIPS Pseudoinstructions

Pseudoinstructions are legal MIPS assembly language instructions that do not translate directly to binary machine code; that is, they do not have a direct hardware implementation. They are provided as a convenience for the programmer. The assembler will translate a pseudoinstruction to an equivalent single, more complex instruction or a short sequence of instructions.

Below is a list of MIPS pseudoinstructions and their functions. Note that such instructions do not include binary encoding information, since they do not translate to machine code directly.

Pseudoinstruction	Meaning	Alternative MIPS
`abs rdest, rsrc`	absolute value	several instructions
`div rdest, rsrc1, rsrc2`	divide (with overflow)	`div rs, rt`, etc.
`divu rdest, rsrc1, rsrc2`	divide (without overflow)	`divu rs, rt`, etc.
`mulo rdest, rsrc1, rsrc2`	multiply (with overflow)	`mult rs, rt`, etc.
`mulou rdest, rsrc1, rsrc2`	unsigned multiply (with overflow)	`multu rs, rt`, etc.
`neg rdest, rsrc`	negate value (with overflow)	`sub rd, $0, rt`
`negu rdest, rsrc`	negate value (without overflow)	`subu rd, $0, rt`
`not rdest, rsrc`	logical not	`nor rd, rs, $0`
`rem rdest, rsrc1, rscr2`	remainder	`div rs, rt`, etc.
`remu rdest, rsrc1, rsrc2`	unsigned remainder	`divu rs, rt`, etc.
`li rdest, imm`	load immediate	`addiu rs,$0,imm`
`seq rdest, rsrc1, rsrc2`	set rdest to 1 if rsrc1 $=$ rsrc2	several instructions
`sge rdest, rsrc1, rscr2`	set rdest to 1 if rsrc1 $\geq$ rsrc2	several instructions
`sgeu rdest, rsrc1, rscr2`	set rdest to 1 if rsrc1 $\geq$ rsrc2	several instructions
`sgt rdest, rsrc1, rscr2`	set rdest to 1 if rsrc1 $>$ rsrc2	several instructions
`sgtu rdest, rsrc1, rscr2`	set rdest to 1 if rsrc1 $>$ rsrc2	several instructions
`sle rdest, rsrc1, rscr2`	set rdest to 1 if rsrc1 $\leq$ rsrc2	several instructions
`sleu rdest, rsrc1, rscr2`	set rdest to 1 if rsrc1 $\leq$ rsrc2	several instructions
`sne rdest, rsrc1, rscr2`	set rdest to 1 if rsrc1 $\neq$ rsrc2	several instructions
`b label`	branch to label	`bc1t label`
`beqz rsrc, label`	branch to label if rsrc $= 0$	`beq rs, $0, label`
`bge rsrc1, rsrc2, label`	branch to label if rsrc1 $\geq$ rsrc2	`slt`, etc.
`bgeu rsrc1, rsrc2, label`	branch to label if rsrc1 $\geq$ rsrc2	`sltu`, etc.
`bgt rsrc1, rsrc2, label`	branch to label if rsrc1 $>$ rsrc2	`slt`, etc.
`bgtu rsrc1, rsrc2, label`	branch to label if rsrc1 $>$ rsrc2	`sltu`, etc.
`ble rsrc1, rsrc2, label`	branch to label if rsrc1 $\leq$ rsrc2	`slt`, etc.
`bleu rsrc1, rsrc2, label`	branch to label if rsrc1 $\leq$ rsrc2	`sltu`, etc.
`blt rsrc1, rsrc2, label`	branch to label if rsrc1 $<$ rsrc2	`slt`, etc.
`bltu rsrc1, rsrc2, label`	branch to label if rsrc1 $<$ rsrc2	`sltu`, etc.
`bnez rsrc, label`	branch to label if rsrc $\neq 0$	`bnez rs, $0, label`
`la rdest, address`	load address into rdest	`lui`, etc.
`ld rdest, address`	load 64-bit address into register pair	several instructions
`ulh rdest, address`	load 16-bit address into rdest*	big mess
`ulhu rdest, address`	load 16-bit address into rdest*	big mess
`sd rsrc, address`	store 64-bit quantity at address	big mess
`sdu rsrc, address`	store 64-bit quantity at address	big mess
`usw rsrc, address`	store 16-bit quantity at address*	still a big mess
`move rdest, rsrc`	copy value in rsrc to rdest	`addu rd, $0, rs`
`mfc1.d rdest, frsrc1`	mv fl. pt. register pair to rdest	`mfc1 rd, fs`, etc.

*Possibly unaligned.