

MIPS64 Instruction Set (ver 3.2)

Legend:

- * **double-word (64-bit)** = $b_0 b_1 b_2 b_3 \dots b_{60} b_{61} b_{62} b_{63}$
- * **##** = concatenate
- * \leftarrow_n = n-bit of data to be transferred
- * 0^n = replicate 0 n-times (zero-extend)
- * **[Mem[offset+ RS]₀]ⁿ** = replicate sign bit n-times (sign-extension)
- * **RD, RS, RT** = General Purpose Register (Integer), RD = usually destination, RS=usually source, RT=usually another source
- * **FD, FS, FT, FR** = Floating-point Register, FD=destination, FS=source, FT=another source, FR=another source
- * **hi/lo** = integer multiply/divide result register
- * **c0** = CPU status register / **c1** = FPU status register

***Note: the list is not comprehensive. For full listing, please refer to the book [See MIPS run, 2nd Edition, chapter 8.2.3].**

• Load-Store Instructions:

Mnemonic	Meaning	Format	Operation
LB	Load byte (signed)	LB RD, offset(RS)	$RD \leftarrow_{64} \text{Mem}[\text{offset}+ \text{RS}]_0^{56} \text{##} \text{Mem}[\text{offset}+ \text{RS}]$
LBU	Load byte (unsigned)	LBU RD, offset(RS)	$RD \leftarrow_{64} 0^{56} \text{##} \text{Mem}[\text{offset}+ \text{RS}]$
LH	Load half-word (signed)	LH RD, offset(RS)	$RD \leftarrow_{64} [\text{Mem}[\text{offset}+ \text{RS}]_0^{48} \text{##} \text{Mem}[\text{offset}+ \text{RS}]]$
LHU	Load half-word (unsigned)	LHU RD, offset(RS)	$RD \leftarrow_{64} 0^{48} \text{##} \text{Mem}[\text{offset}+ \text{RS}]$
LW	Load word (signed)	LW RD, offset(RS)	$RD \leftarrow_{64} [\text{Mem}[\text{offset}+ \text{RS}]_0^{32} \text{##} \text{Mem}[\text{offset}+ \text{Regs}[\text{RS}]]]$
LWU	Load word (unsigned)	LWU RD, offset(RS)	$RD \leftarrow_{64} 0^{32} \text{##} \text{Mem}[\text{offset}+ \text{RS}]$
LD	Load double-word	LD RD, offset(RS)	$RD \leftarrow_{64} \text{Mem}[\text{offset}+ \text{RS}]$
SB	Store byte	SB RT, offset(RS)	$\text{Mem}[\text{offset}+ \text{RS}] \leftarrow_8 \text{RT}_{56..63}$
SH	Store half-word	SH RT, offset(RS)	$\text{Mem}[\text{offset}+ \text{RS}] \leftarrow_{16} \text{RT}_{48..63}$
SW	Store word	SW RT, offset(RS)	$\text{Mem}[\text{offset}+ \text{RS}] \leftarrow_{32} \text{RT}_{32..63}$
SD	Store double-word	SD RT, offset(RS)	$\text{Mem}[\text{offset}+ \text{RS}] \leftarrow_{64} \text{RT}_{0..63}$
L.S	Load single-precision	L.S FD, offset(RS)	$\text{FD} \leftarrow_{64} \text{Mem}[\text{offset}+ \text{RS}] \text{##} 0^{32}$
L.D	Load double-precision	L.D FD, offset(RS)	$\text{FD} \leftarrow_{64} \text{Mem}[\text{offset}+ \text{RS}]$
LUI	Load upper immediate	LUI RD, imm	$RD \leftarrow (\text{imm}_0)^{32} \text{##} \text{imm} \text{##} 0^{16}$
S.S	Store single-precision	S.S FT, offset(RS)	$\text{Mem}[\text{offset}+ \text{RS}] \leftarrow_{32} \text{FT}_{0..31}$
S.D	Store double-precision	S.D FT, offset(RS)	$\text{Mem}[\text{offset}+ \text{RS}] \leftarrow_{64} \text{FT}_{0..63}$

• Arithmetic Instructions

Mnemonic	Meaning	Format	Operation
DADDU	Double-word addition	DADDU RD, RS, RT	$RD \leftarrow RS + RT$
DADDIU	Double-word add w/immediate (signed constant)	DADDIU RD, RS, imm	$RD \leftarrow RS + \text{imm}$
DSUBU	Double-word subtraction	DSUBU RD, RS, RT	$RD \leftarrow RS - RT$
DMULT	Double-word multiplication (signed)	DMULT RS, RT	$\text{hi/lo} \leftarrow RS * RT$
DMULTU	Double-word multiplication (unsigned)	DMULTU RS, RT	$\text{hi/lo} \leftarrow RS * RT$
DDIV	Double-word division (signed)	DDIV RS, RT	$\text{lo} = RS \text{ div } RT; \text{hi} = RS \text{ mod } RT$
DDIVU	Double-word division (unsigned)	DDIVU RS, RT	$\text{lo} = RS \text{ div } RT; \text{hi} = RS \text{ mod } RT$
MADD	Multiply-Accumulate	MADD RS, RT	$\text{hilo} = \text{hilo} + (RS * RT)$

- **Special move Instructions:**

Mnemonic	Meaning	Format	Operation
MFHI	Move from hi	MFHI RD	$RD \leftarrow hi$
MFLO	Move from lo	MFLO RD	$RD \leftarrow lo$
MTHI	Move to hi	MTHI RS	$hi \leftarrow RS$
MTLO	Move to lo	MTLO RS	$lo \leftarrow RS$
MFC0	Move from C0 to RD (32-bit)	MFC0 RD, CS	$RD \leftarrow CS$
MTC0	Move to C0 from RS (32-bit)	MTC0 RS, CD	$CD \leftarrow RS$
DMFC0	Move from C0 to RD (64-bit)	DMFC0 RD, CS	$RD \leftarrow CS$
DMTC0	Move to C0 from RS (64-bit)	DMTC0 RS, CD	$CD \leftarrow RS$
MFC1	Move from FPR to GPR (32-bit)	MFC1 RD, FS	$RD \leftarrow FS$
MTC1	Move to FPR from GPR (32-bit)	MTC1 RS, FD	$FD \leftarrow RS$
DMFC1	Move from FPR to GPR (64-bit)	DMFC1 RD, FS	$RD \leftarrow FS$
DMTC1	Move to FPR from GPR (64-bit)	DMTC1 RS, FD	$FD \leftarrow RS$
MOV.S	Move from one Single Precision FPR to another	MOV.S FD, FS	$FD \leftarrow FS$
MOV.D	Move from one Double Precision FPR to another	MOV.D FD, FS	$FD \leftarrow FS$

- **Logical Instructions**

Mnemonic	Meaning	Format	Operation
AND	Logical AND	AND RD, RS, RT	$RD \leftarrow RS \bullet RT$
OR	Logical OR	OR RD, RS, RT	$RD \leftarrow RS \mid RT$
XOR	Logical XOR	XOR RD, RS, RT	$RD \leftarrow RS \oplus RT$
NOR	Logical NOR	NOR RD, RS, RT	$RD \leftarrow \sim (RS \mid RT)$
ANDI	Logical AND with immediate	ANDI RD, RS, imm	$RD \leftarrow RS \bullet imm$
ORI	Logical OR with immediate	ORI RD, RS, imm	$RD \leftarrow RS \mid imm$
XORI	Logical XOR with immediate	XORI RD, RS, imm	$RD \leftarrow RS \oplus imm$

- **Shift Instructions**

Mnemonic	Meaning	Format	Operation
DSLLV	Double-word shift left logical variable	DSLLV RD, RS, RT	$RD \leftarrow RS \ll RT_{58..63}$ (only the lower 6 bits of RT is considered) Shift logical left: as bits shift left, the LSB of the register is filled with zero
DSRLV	Double-word shift right logical variable	DSRLV RD, RS, RT	$RD \leftarrow RS \gg RT_{58..63}$ (only the lower 6 bits of RT is considered) Shift logical right: as bits shift right, the MSB of the register is filled with zero
DSRAV	Double-word shift right arithmetic variable	DSRLV RD, RS, RT	$RD \leftarrow RS \gg RT_{58..63}$ (only the lower 6 bits of RT is considered) Shift arithmetic right: as bits shift right, the MSB of the register is filled with sign bit
DSLL	Double-word shift left logical	DSLL RD, RS, imm	$RD \leftarrow RS \ll imm_{11..15}$ (only the low 5 bits of imm is considered) Shift logical left: as bits shift left, the LSB of the register is filled with zero
DSRL	Double-word shift right logical	DSRL RD, RS, imm	$RD \leftarrow RS \gg imm_{11..15}$ (only the low 5 bits of imm is considered) Shift logical right: as bits shift right, the MSB of the register is filled with zero
DSRA	Double-word shift right arithmetic	DSRA RD, RS, imm	$RD \leftarrow RS \gg imm_{11..15}$ (only the low 5 bits of imm is considered) Shift arithmetic right: as bits shift right, the MSB of the register is filled with sign bit

• **Control & Transfer Instructions**

J	Unconditional Jump	J Label	$PC_{36..63} \leftarrow \text{Label}$ Internally, Label (which is 26-bit) is left shifted twice and then replaces the lower 28 bits of the PC. Thus, Label can be viewed as target address div 4
JR	Unconditional Jump	JR RS	$PC \leftarrow RS$
JAL	Jump and link ("call")	JAL Label	$R31 \leftarrow PC+4$ (no delay branch slot) $R31 \leftarrow PC+8$ (with delay branch slot) $PC_{36..63} \leftarrow \text{Label}$ Internally, Label (which is 26-bit) is left shifted twice and then replaces the lower 28 bits of the PC. Thus, Label can be viewed as target address div 4
JALR	Jump and Link ("call")	JALR RS	$R31 \leftarrow PC+4$ (no delay branch slot) $R31 \leftarrow PC+8$ (with delay branch slot) $PC \leftarrow RS$
BLEZ	Branch if less than or equal to zero	BLEZ RS, Label ($RS \leq 0$)*	If ($RS \leq 0$)* then $PC \leftarrow PC + \text{Label}$ Internally, label (which is 16-bit signed offset) is left shifted twice and then added to the PC+4. Thus, the offset can also be viewed as the distance of instruction to a label. The next instruction after the branch is viewed as offset 0
BLTZ	Branch if less than zero	BLTZ RS, Label ($RS < 0$)*	
BGEZ	Branch if greater than or equal to zero	BGEZ RS, Label ($RS \geq 0$)*	If ($RS \geq 0$)* then $PC \leftarrow PC + \text{Label}$ Internally, label (which is 16-bit signed offset) is left shifted twice and then added to the PC+4. Thus, the offset can also be viewed as the distance of instruction to a label. The next instruction after the branch is viewed as offset 0
BGTZ	Branch if greater than zero	BGTZ RS, Label ($RS > 0$)*	
BEQ	Branch if equal	BEQ RS, RT, Label	If ($RS=RT$) then $PC \leftarrow PC + \text{Label}$ Internally, label (which is 16-bit signed offset) is left shifted twice and then added to the PC+4. Thus, the offset can also be viewed as the distance of instruction to a label. The next instruction after the branch is viewed as offset 0
BNE	Branch if not equal	BNE RS, RT, Label	If ($RS \neq RT$) then $PC \leftarrow PC + \text{Label}$ Internally, label (which is 16-bit signed offset) is left shifted twice and then added to the PC+4. Thus, the offset can also be viewed as the distance of instruction to a label. The next instruction after the branch is viewed as offset 0
MOVZ	Conditional Move if zero	MOVZ RD, RS, RT	If ($RT = 0$) $RD \leftarrow RS$
MOVN	Conditional Move if Not zero	MOVN RD, RS, RT	If ($RT \neq 0$) $RD \leftarrow RS$

- **Set-on-condition Instructions**

Mnemonic	Meaning	Format	Operation
SLT	Set if less than (Signed)	SLT RD, RS, RT	$RD \leftarrow RS < RT ? 1 : 0$
SLTU	Set if less than (Unsigned)	SLTU RD, RS, RT	$RD \leftarrow RS < RT ? 1 : 0$
SLTI	Set if less than immediate (Signed)	SLTI RD, RS, imm	$RD \leftarrow RS < \text{imm} ? 1 : 0$
SLTIU	Set if less than immediate (Unsigned)	SLTIU RD, RS, imm	$RD \leftarrow RS < \text{imm} ? 1 : 0$

- **Floating Point Instructions**

Mnemonic	Meaning	Format	Operation
ADD.D	Add double precision	ADD.D FD, FS, FT	$FD \leftarrow FS + FT$
ADD.S	Add single precision	ADD.S FD, FS, FT	$FD \leftarrow FS + FT$
ADD.PS	Add pair of single precision	ADD.PS FD, FS, FT	$FD_{0..31} \leftarrow FS_{0..31} + FT_{0..31}$ $FD_{32..63} \leftarrow FS_{32..63} + FT_{32..63}$
SUB.D	Subtract double precision	SUB.D FD, FS, FT	$FD \leftarrow FS - FT$
SUB.S	Subtract single precision	SUB.S FD, FS, FT	$FD \leftarrow FS - FT$
SUB.PS	Subtract pair of single precision	SUB.PS FD, FS, FT	$FD_{0..31} \leftarrow FS_{0..31} - FT_{0..31}$ $FD_{32..63} \leftarrow FS_{32..63} - FT_{32..63}$
MUL.D	Multiply double precision	MUL.D FD, FS, FT	$FD \leftarrow FS * FT$
MUL.S	Multiply single precision	MUL.S FD, FS, FT	$FD \leftarrow FS * FT$
MUL.PS	Multiply pair of single precision	MUL.PS FD, FS, FT	$FD_{0..31} \leftarrow FS_{0..31} * FT_{0..31}$ $FD_{32..63} \leftarrow FS_{32..63} * FT_{32..63}$
DIV.D	Divide double precision	DIV.D FD, FS, FT	$FD \leftarrow FS / FT$
DIV.S	Divide single precision	DIV.S FD, FS, FT	$FD \leftarrow FS / FT$
DIV.PS	Divide pair of single precision	DIV.PS FD, FS, FT	$FD_{0..31} \leftarrow FS_{0..31} / FT_{0..31}$ $FD_{32..63} \leftarrow FS_{32..63} / FT_{32..63}$
MADD.D	Multiply-ADD double precision	MADD.D FD, FR, FS, FT	$FD \leftarrow FR + (FS * FT)$
MADD.S	Multiply-ADD single precision	MADD.S FD, FR, FS, FT	$FD \leftarrow FR + (FS * FT)$
MADD.PS	Multiply-ADD pair of single precision	MADD.PS FD, FR, FS, FT	$FD_{0..31} \leftarrow FR_{0..31} + (FS_{0..31} * FT_{0..31})$ $FD_{32..63} \leftarrow FR_{32..63} + (FS_{32..63} * FT_{32..63})$
CVT.d.s	Convert from type y to type x; where x & y can be s (SP floating point), d (DP floating point), w (32-bit integer), l (64-bit integer)	CVT.x.y FD, FS	$FD \leftarrow FS$
C._.S	Set floating point status register (FCC) if condition is _; where “_” can be LT, LE, GT, GE, EQ, NE Default is 0 (0-7). For Single Precision comparison	C._.S FS, FT C._.S cc, FS, FT	$FCC(cc) = FS < FT ? 1 : 0$
C._.D	Set floating point status register (FCC) if condition is _; where “_” can be LT, LE, GT, GE, EQ, NE Default is 0 (0-7). For Double Precision comparison	C._.D FS, FT C._.D cc, FS, FT	$FCC(CC) = FS < FT ? 1 : 0$

Mnemonic	Meaning	Format	Operation
BC1T	Branch if FP status register is 1	BC1T Label BC1T cc, Label	If $FCC(cc) = 1$ then $PC \leftarrow PC + \text{Label}$ Internally, label (which is 16-bit signed offset) is left shifted twice and then added to the PC+4. Thus, the offset can also be viewed as the distance of instruction to a label. The next instruction after the branch is viewed as offset 0
BC1F	Branch if FP status register is 0	BC1F Label	If $FCC(cc) = 0$ then $PC \leftarrow PC + \text{Label}$ Internally, label (which is 16-bit signed offset) is left shifted twice and then added to the PC+4. Thus, the offset can also be viewed as the distance of instruction to a label. The next instruction after the branch is viewed as offset 0

- Special Instructions**

Mnemonic	Meaning	Format	Operation
TRAP	Transfer to operating at a vectored address	TRAP	
ERET	Return to user code from an exception; restore user mode	ERET	
NOP	No Operation	NOP	