

6.5900 Computer System Architecture

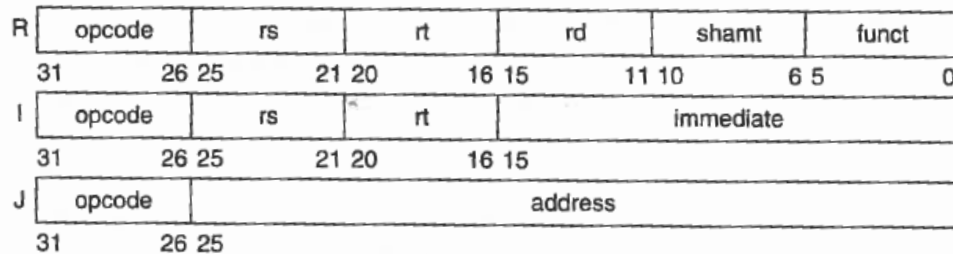
RISC ISA – MIPS64

<http://csg.csail.mit.edu/6.823/>

Here is a brief summary of the MIPS64 instructions used in the self-assessment test. All general-purpose registers (GPRs) are assumed to be 64 bits. (R0 is hardwired to zero.) For more information, please check out either Section 1.3 or Appendix A of the Hennessy and Patterson book.

Instruction Encoding

Basic instruction formats



Floating-point instruction formats

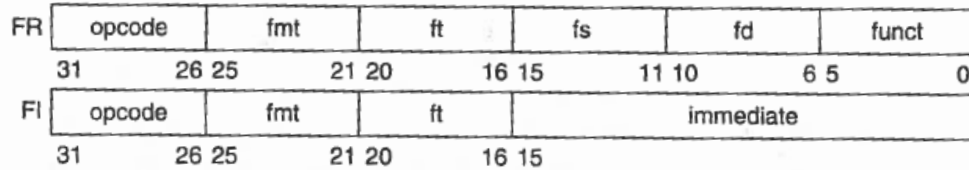


Figure 1.6 MIPS64 instruction set architecture formats. All instructions are 32 bits long. The R format is for integer register-to-register operations, such as DADDU, DSUBU, and so on. The I format is for data transfers, branches, and immediate instructions, such as LD, SD, BEQZ, and DADDIs. The J format is for jumps, the FR format for floating-point operations, and the FI format for floating-point branches.

Instruction Types

Instruction type/opcode	Instruction meaning
<i>Data transfers</i>	<i>Move data between registers and memory, or between the integer and FP or special registers; only memory address mode is 16-bit displacement + contents of a GPR</i>
LB, LBU, SB	Load byte, load byte unsigned, store byte (to/from integer registers)
LH, LHU, SH	Load half word, load half word unsigned, store half word (to/from integer registers)
LW, LWU, SW	Load word, load word unsigned, store word (to/from integer registers)
LD, SD	Load double word, store double word (to/from integer registers)
L.S, L.D, S.S, S.D	Load SP float, load DP float, store SP float, store DP float
MFC0, MTC0	Copy from/to GPR to/from a special register
MOV.S, MOV.D	Copy one SP or DP FP register to another FP register
MFC1, MTC1	Copy 32 bits to/from FP registers from/to integer registers
<i>Arithmetic/logical</i>	<i>Operations on integer or logical data in GPRs; signed arithmetic trap on overflow</i>
DADD, DADDI, DADDU, DADDIU	Add, add immediate (all immediates are 16 bits); signed and unsigned
DSUB, DSUBU	Subtract, signed and unsigned
DMUL, DMULU, DDIV, DDIVU, MADD	Multiply and divide, signed and unsigned; multiply-add; all operations take and yield 64-bit values
AND, ANDI	And, and immediate
OR, ORI, XOR, XORI	Or, or immediate, exclusive or, exclusive or immediate
LUI	Load upper immediate; loads bits 32 to 47 of register with immediate, then sign-extends
DSLL, DSRL, DSRA, DSLLV, DSRLV, DSRAV	Shifts: both immediate (DS__) and variable form (DS__V); shifts are shift left logical, right logical, right arithmetic
SLT, SLTI, SLTU, SLTIU	Set less than, set less than immediate, signed and unsigned
<i>Control</i>	<i>Conditional branches and jumps; PC-relative or through register</i>
BEQZ, BNEZ	Branch GPRs equal/not equal to zero; 16-bit offset from PC + 4
BEQ, BNE	Branch GPR equal/not equal; 16-bit offset from PC + 4
BC1T, BC1F	Test comparison bit in the FP status register and branch; 16-bit offset from PC + 4
MOVN, MOVZ	Copy GPR to another GPR if third GPR is negative, zero
J, JR	Jumps: 26-bit offset from PC + 4 (J) or target in register (JR)
JAL, JALR	Jump and link: save PC + 4 in R31, target is PC-relative (JAL) or a register (JALR)
TRAP	Transfer to operating system at a vectored address
ERET	Return to user code from an exception; restore user mode
<i>Floating point</i>	<i>FP operations on DP and SP formats</i>
ADD.D, ADD.S, ADD.PS	Add DP, SP numbers, and pairs of SP numbers
SUB.D, SUB.S, SUB.PS	Subtract DP, SP numbers, and pairs of SP numbers
MUL.D, MUL.S, MUL.PS	Multiply DP, SP floating point, and pairs of SP numbers
MADD.D, MADD.S, MADD.PS	Multiply-add DP, SP numbers, and pairs of SP numbers
DIV.D, DIV.S, DIV.PS	Divide DP, SP floating point, and pairs of SP numbers
CVT.___	Convert instructions: CVT.x.y converts from type x to type y, where x and y are L (64-bit integer), W (32-bit integer), D (DP), or S (SP). Both operands are FPRs.
C.___.D, C.___.S	DP and SP compares: “___” = LT,GT,LE,GE,EQ,NE; sets bit in FP status register

Figure 1.5 Subset of the instructions in MIPS64. SP = single precision; DP = double precision. Appendix A gives much more detail on MIPS64. For data, the most significant bit number is 0; least is 63.