MIPS I

The first version of the MIPS architecture was designed by MIPS Computer Systems for its R2000microprocessor, the first MIPS implementation. Both MIPS and the R2000 were introduced together in 1985.When MIPS II was introduced, *MIPS* was renamed *MIPS I* to distinguish it from the new version.[3]:32

MIPS is a load/store architecture (also known as a *register-register architecture*); except for the load/store instructions used to access memory, all instructions operate on the registers.

Registers

MIPS I has thirty-two 32-bit general-purpose registers (GPR). Register $0 is hardwired to zero and writes to it are discarded. Register $31 is the link register. For integer multiplication and division instructions, which run asynchronously from other instructions, a pair of 32-bit registers, *HI* and *LO*, are provided. There is a small set of instructions for copying data between the general-purpose registers and the HI/LO registers.

The program counter has 32 bits. The two low-order bits always contain zero since MIPS I instructions are 32 bits long and are aligned to their natural word boundaries.

Instruction formats

Instructions are divided into three types: R, I and J. Every instruction starts with a 6-bit opcode. In addition to the opcode, R-type instructions specify three registers, a shift amount field, and a function field; I-type instructions specify two registers and a 16-bit immediate value; J-type instructions follow the opcode with a 26-bit jump target.[2]:A-174

The following are the three formats used for the core instruction set:

Type	-31-			format (bits)		-0-
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	rt (5)	immediate (16)		
J	opcode (6)	address (26)				

CPU instructions

Loads and stores

MIPS I has instructions that load and store 8-bit bytes, 16-bit halfwords, and 32-bit words. Only one addressing mode is supported: base + displacement. Since MIPS I is a 32-bit architecture, loading quantities fewer than 32 bits requires the datum to be either signed- or zero-extended to 32 bits. The load instructions suffixed by "unsigned" perform zero extension; otherwise sign extension is performed. Load instructions source the base from the contents of a GPR (rs) and write the result to another GPR (rt). Store instructions source the base from the contents of a GPR (rs) and the store data from another GPR (rt). All load and store instructions compute the memory address by summing the base with the sign-extended 16-bit immediate. MIPS I requires all memory accesses to be aligned to their natural word boundaries, otherwise an exception is signaled. To support efficient unaligned memory accesses, there are load/store word instructions suffixed by "left" or "right". All load instructions are followed by a load delay slot. The instruction in the load delay slot cannot use the data loaded by the load instruction. The load delay slot can be filled with an instruction that is not dependent on the load; a nop is substituted if such an instruction cannot be found.

Instruction name	Mnemonic	Format	Encoding			
Load Byte	LB	I	32_{10}	rs	rt	offset
Load Halfword	LH	I	33_{10}	rs	rt	offset
Load Word Left	LWL	I	34_{10}	rs	rt	offset
Load Word	LW	I	35_{10}	rs	rt	offset
Load Byte Unsigned	LBU	I	36_{10}	rs	rt	offset
Load Halfword Unsigned	LHU	I	37_{10}	rs	rt	offset
Load Word Right	LWR	I	38_{10}	rs	rt	offset
Store Byte	SB	I	40_{10}	rs	rt	offset
Store Halfword	SH	I	41_{10}	rs	rt	offset
Store Word Left	SWL	I	42_{10}	rs	rt	offset
Store Word	SW	I	43_{10}	rs	rt	offset
Store Word Right	SWR	I	46_{10}	rs	rt	offset

ALU

MIPS I has instructions to perform addition and subtraction. These instructions source their operands from two GPRs (rs and rt), and write the result to a third GPR (rd). Alternatively, addition can source one of the operands from a 16-bit immediate (which is sign-extended to 32 bits). The instructions for addition and subtraction have two variants: by default, an exception is signaled if the result overflows; instructions with the "unsigned" suffix do not signal an exception. The overflow check interprets the result as a 32-bit two's complement integer.

MIPS I has instructions to perform bitwise logical AND, OR, XOR, and NOR. These instructions source their operands from two GPRs and write the result to a third GPR. The AND, OR, and XOR instructions can alternatively source one of the operands from a 16-bit immediate (which is zero-extended to 32 bits).

The Set on *relation* instructions write one or zero to the destination register if the specified relation is true or false. These instructions source their operands from two GPRs or one GPR and a 16-bit immediate (which is sign-extended to 32 bits), and write the result to a third GPR. By default, the operands are interpreted as signed integers. The variants of these instructions that are suffixed with "unsigned" interpret the operands as unsigned integers (even those that source an operand from the sign-extended 16-bit immediate).

The Load Immediate Upper instruction copies the 16-bit immediate into the high-order 16 bits of a GPR. It is used in conjunction with the Or Immediate instruction to load a 32-bit immediate into a register.

Instruction name	Mnemonic	Format	Encoding					
Add	ADD	R	0_{10}	rs	rt	rd	0_{10}	32_{10}
Add Unsigned	ADDU	R	0_{10}	rs	rt	rd	0_{10}	33_{10}
Subtract	SUB	R	0_{10}	rs	rt	rd	0_{10}	34_{10}
Subtract Unsigned	SUBU	R	0_{10}	rs	rt	rd	0_{10}	35_{10}
And	AND	R	0_{10}	rs	rt	rd	0_{10}	36_{10}
Or	OR	R	0_{10}	rs	rt	rd	0_{10}	37_{10}
Exclusive Or	XOR	R	0_{10}	rs	rt	rd	0_{10}	38_{10}
Nor	NOR	R	0_{10}	rs	rt	rd	0_{10}	39_{10}
Set on Less Than	SLT	R	0_{10}	rs	rt	rd	0_{10}	42_{10}
Set on Less Than Unsigned	SLTU	R	0_{10}	rs	rt	rd	0_{10}	43_{10}
Add Immediate	ADDI	I	8_{10}	rs	rd	immediate		
Add Immediate Unsigned	ADDIU	I	9_{10}	$s	$d	immediate		
Set on Less Than Immediate	SLTI	I	10_{10}	$s	$d	immediate		
Set on Less Than Immediate Unsigned	SLTIU	I	11_{10}	$s	$d	immediate		
And Immediate	ANDI	I	12_{10}	$s	$d	immediate		
Or Immediate	ORI	I	13_{10}	$s	$d	immediate		
Exclusive Or Immediate	XORI	I	14_{10}	$s	$d	immediate		
Load Upper Immediate	LUI	I	15_{10}	0_{10}	$d	immediate		

Shifts

MIPS I has instructions to perform left and right logical shifts and right arithmetic shifts. The operand is obtained from a GPR (rt), and the result is written to another GPR (rd). The shift distance is obtained from either a GPR (rs) or a 5-bit "shift amount" (the "sa" field).

Instruction name	Mnemonic	Format	Encoding					
Shift Left Logical	SLL	R	0_{10}	0_{10}	rt	rd	sa	0_{10}
Shift Right Logical	SRL	R	0_{10}	0_{10}	rt	rd	sa	2_{10}
Shift Right Arithmetic	SRA	R	0_{10}	0_{10}	rt	rd	sa	3_{10}
Shift Left Logical Variable	SLLV	R	0_{10}	rs	rt	rd	0_{10}	4_{10}
Shift Right Logical Variable	SRLV	R	0_{10}	rs	rt	rd	0_{10}	6_{10}
Shift Right Arithmetic Variable	SRAV	R	0_{10}	rs	rt	rd	0_{10}	7_{10}

Multiplication and division

MIPS I has instructions for signed and unsigned integer multiplication and division. These instructions source their operands from two GPRs and write their results to a pair of 32-bit registers called HI and LO, since they may execute separately from (and concurrently with) the other CPU instructions. For multiplication, the high- and low-order halves of the 64-bit product is written to HI and LO (respectively). For division, the quotient is written to LO and the remainder to HI. To access the results, a pair of instructions (Move from HI and Move from LO) is provided to copy the contents of HI or LO to a GPR. These instructions are interlocked: reads of HI and LO do not proceed past an unfinished arithmetic instruction that will write to HI and LO. Another pair of instructions (Move to HI or Move to LO) copies the contents of a GPR to HI and LO. These instructions are used to restore HI and LO to their original state after exception handling. Instructions that read HI or LO must be separated by two instructions that do not write to HI or LO.

Instruction name	Mnemonic	Format	Encoding					
Move from HI	MFHI	R	0_{10}	0_{10}	0_{10}	rd	0_{10}	16_{10}
Move to HI	MTHI	R	0_{10}	rs	0_{10}	0_{10}	0_{10}	17_{10}
Move from LO	MFLO	R	0_{10}	0_{10}	0_{10}	rd	0_{10}	18_{10}
Move to LO	MTLO	R	0_{10}	rs	0_{10}	0_{10}	0_{10}	19_{10}
Multiply	MULT	R	0_{10}	rs	rt	0_{10}	0_{10}	24_{10}
Multiply Unsigned	MULTU	R	0_{10}	rs	rt	0_{10}	0_{10}	25_{10}
Divide	DIV	R	0_{10}	rs	rt	0_{10}	0_{10}	26_{10}
Divide Unsigned	DIVU	R	0_{10}	rs	rt	0_{10}	0_{10}	27_{10}

Jump and branch

All MIPS I control flow instructions are followed by a branch delay slot. Unless the branch delay slot is filled by an instruction performing useful work, an nop is substituted. MIPS I branch instructions compare the contents of a GPR (rs) against zero or another GPR (rt) as signed integers and branch if the specified condition is true. Control is transferred to the address computed by shifting the 16-bit offset left by two bits, sign-extending the 18-bit result, and adding the 32-bit sign-extended result to the sum of the program counter (instruction address) and 8_{10}. Jumps have two versions: absolute and register-indirect. Absolute jumps ("Jump" and "Jump and Link") compute the address control is transferred to by shifting the 26-bit instr_index left by two bits and concatenating the 28-bit result with the four high-order bits of the address of the instruction in the branch delay slot. Register-indirect jumps transfer control to the instruction at the address sourced from a GPR (rs). The address sourced from the GPR must be word-aligned, else an exception is signaled after the instruction in the branch delay slot is executed. Branch and jump instructions that link (except for "Jump and Link Register") save the return address to GPR 31. The "Jump and Link Register" instruction permits the return address to be saved to any writable GPR.

Instruction name	Mnemonic	Format	Encoding					
Jump Register	JR	R	0_{10}	rs	0_{10}	0_{10}	0_{10}	8_{10}
Jump and Link Register	JALR	R	0_{10}	rs	0_{10}	rd	0_{10}	9_{10}
Branch on Less Than Zero	BLTZ	I	1_{10}	rs	0_{10}	offset		
Branch on Greater Than or Equal to Zero	BGEZ	I	1_{10}	rs	1_{10}	offset		
Branch on Less Than Zero and Link	BLTZAL	I	1_{10}	rs	16	offset		
Branch on Greater Than or Equal to Zero and Link	BGEZAL	I	1_{10}	rs	17	offset		
Jump	J	J	2_{10}	instr_index				
Jump and Link	JAL	J	3_{10}	instr_index				
Branch on Equal	BEQ	I	4_{10}	rs	rt	offset		
Branch on Not Equal	BNE	I	5_{10}	rs	rt	offset		
Branch on Less Than or Equal to Zero	BLEZ	I	6_{10}	rs	0_{10}	offset		
Branch on Greater Than Zero	BGTZ	I	7_{10}	rs	0_{10}	offset		

Exception

MIPS I has two instructions for software to signal an exception: System Call and Breakpoint. System Call is used by user mode software to make kernel calls; and Breakpoint is used to transfer control to a debugger via the kernel's exception handler. Both instructions have a 20-bit Code field that can contain operating environment-specific information for the exception handler.

Instruction name	Mnemonic	Format	Encoding		
System Call	SYSCALL	?	0_{10}	Code	12_{10}
Breakpoint	BREAK	?	0_{10}	Code	13_{10}