MIPS CPU (Central Processing Unit) Design Document

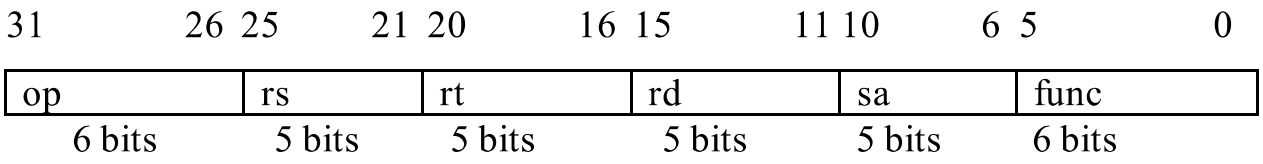
Project: Design, Verification, and Implementation of a High-Performance Multi-core CPU.

CPU ISA Version: MIPS32

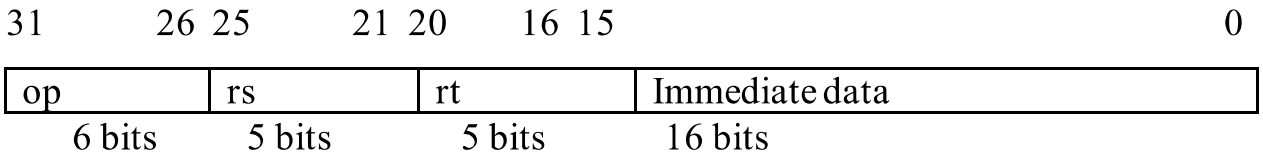
MIPS Instruction Format:

All the MIPS instructions following the MIPS32 ISA are 32 bit long. Below are 3 instruction formats of MIPS instructions where op stands for opcode of the instruction and the func stands for function.

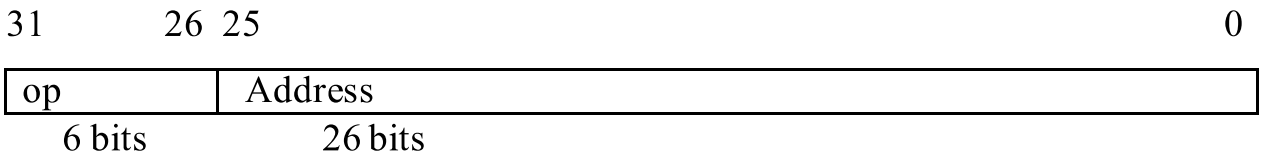
R Format (Register)



I Format (Immediate)



J Format (Jump)



Note

1. The op in R-Format is 000000. We would use func to perform the operation.

2. Each of rt, rs, rd is a 5-bit register number. The shift instruction uses the sa area to specify the amount of shift. The contents of the rt register would be shifted by the value of the sa register and the result would be stored in the rd register. Only the R-Format instructions operate on two operands sourced from rs, rt registers and store the results in the rd register.

3. The I-Format instructions use the immediate data as the second operand. The immediate data should either be sign or zero extended to 32 bits. The first source operand is in the rs register and the result is written to the rt register.

4. The conditional branches using the I-Format instructions compare the operands in the rs, and rt registers to determine whether to branch or not and then use the immediate data as a signed word offset which is used to calculate the branch target address.

5. The load instructions of the I-Format load the contents of the memory to the rt register and the store instructions store the contents of the rt register into the memory. For both the load and store instructions the signed extended immediate data + operand in the rs register is used as the memory address.

6. The address in the J-Format instructions will be shifted by 2 bits to the left to form the low 28 bits of the jump target address.

MIPS General Purpose Registers

Register Name	Register Number	Use
$zero	0	Constant 0
$at	1	Assembler Temporary
$v0 to $v1	2 to 3	Function Return Value
$a0 to $a3	4 to 7	Function Parameters
$t0 to $t7	8 to 15	Temporaries
$s0 to $s7	16 to 23	Saved Temporaries
$t8 to $t9	24 to 25	Temporaries
$k0 to $k1	26 to 27	Reserved for OS Kernel
$gp	28	Global Pointer
$sp	29	Stack Pointer
$fp	30	Frame Pointer
$ra	31	Return Address

Note:

1. The above table gives a base for developing MIPS assembly applications e.g., GNU C Compiler, etc.

2. The $zero register has its content as 0 always.

3. Though the usage of these registers is different their design is the same looking at a hardware point of view.

Major MIPS Integer Instructions

Inst.	[31:26]	[25:21]	[20:16]	[15:11]	[10:6]	[5:0]	Meaning
add	000000	rs	rt	rd	00000	100000	Register add
sub	000000	rs	rt	rd	00000	100010	Register sub
and	000000	rs	rt	rd	00000	100100	Register and
or	000000	rs	rt	rd	00000	100101	Register or
xor	000000	rs	rt	rd	00000	100110	Register xor
div	000000	rs	rt	rd	00000	100001	Register div
sqrt	000000	rs	rt	rd	00000	100011	Register sqrt
sll	000000	00000	rt	rd	sa	000000	Shift left
srl	000000	00000	rt	rd	sa	000010	Logical sft right
sra	000000	00000	rt	rd	sa	000011	Arith sft right
jr	000000	rs	00000	00000	00000	001000	Register jump
addi	001000	rs	rt	Immediate Data			Imm add
andi	001100	rs	rt	Immediate Data			Imm and
ori	001101	rs	rt	Immediate Data			Imm or
xori	001110	rs	rt	Immediate Data			Imm xor
lw	100011	rs	rt	Offset			Load mem
sw	101011	rs	rt	Offset			Store mem
beq	000100	rs	rt	Offset			Branch on equal

bne	000101	rs	rt	Offset			Branch on not eq
lui	001111	00000	rt	Immediate Data			Ld Uppr Imm
j	000010			address			Jump
jal	000011			address			Call

Note:

1. Add/sub/and/or/xor/div/sqrt rd, rs, rt # rd <=== rs op rt. These five instructions have the same format. The rs, rt registers are the source registers numbers and the rd is the destination register number.
2. Sll/srl/sra rd,rt,sa # rd <=== rt shift sa. These 3 shift instructions would shift by the amount given by the 5-bit sa.
3. Addi (add immediate) #rt <=== rs + (sign) immediate data. The addi instruction adds a 16-bit signed immediate data to the 32-bit value in the rs register. The result is saved to the rt register. The 16-bit immediate data is sign-extended to 32-bits.
4. Andi/ori/xori rt, rs , immediate #rt <=== rs op (zero)immediate. The andi, ori, xori instructions perform bitwise logical AND, OR, and XOR respectively on the value in registers rs and a 16-bit unsigned immediate data and the result is written to the rt register. The 16-bit immediate data is zero extended to 32-bits.
5. Rt <=== immediate << 16. The lui (load upper immediate) instruction shifts immediate data to the left by 16bits. By using the lui and ori instructions we can set a register to any 32-bit constant (lui sets high 16 bits and ori sets low 16 bits).
6. Lw, rt, offset(rs) #rt <=== memory[rs+offset]. The load word instruction loads a 32-bit word from memory. The memory address is calculated by adding a 16-bit signed immediate (offset) to the 32-bit value in register rs.
7. Sw rt, offset(rs) #memory[rs+offset] <=== rt. The store word stores a 32-bit word to the memory. The memory address is calculated by adding a 16-bit signed offset to the 32-bit value in the register rs. The word to be stored is in register rt.

8. Beq rs, rt, label #if (rs ==rt) PC <=== label. The beq (branch on equal) instruction transfers control to a PC-relative target address (label) if the values in registers rs and rt are equal. The branch target address is calculated by adding an 18-bit signed constant (the 16-bit offset shifted to left by 2 bits) to the address of the instruction following beq.
9. Bne rs, rt, label #if (rs != rt) PC <=== label. The bne (branch on not equal) instruction transfers control to a PC relative target address (label) if the values in rs and rt are not equal. The branch target address is calculated by adding an 18-bit signed constant (the 16-bit offset shifted to left by 2 bits) to the address of the instruction following bne.
10. J target #PC <=== target. The jump instruction transfers control to a target address who's low 28 bits are the 26-bit target shifted by 2 bits. The remaining upper bits are the corresponding bits of the instruction following the j instruction.
11. Jal target #$31 <=== PC; PC <=== target. The jump and link instruction that does the same job as the J instruction and saves the return address to register $31. The return address is the location at which execution continues after returning from the subroutine. MIPS uses a delayed branch technique; the return address is PC + 8.