myMIPS Instruction Set Architecture Documentation

Led Robster

August 25, 2025

Overview

myMIPS is an ISA that imitates the classic MIPS architecture. It is a byproduct of my interest in electronics and software and the result of my learning journey. May it be useful for at least one person other than me.

Instruction Format

myMIPS is a 16-bit architetcure. Instructions are 16 bits wide, data is 16 bits wide. Describe the instruction format, including fields such as opcode, operands, and immediate values. Use a table to illustrate this.

15–12	**11–9**	**8–6**	**5–3**	**2–0**	
opcode	rs	rt	rd/shamt	Fcode	R-format
opcode	rs	rt	offset/immediate		I-format
opcode	address				J-format

Table 1: Instruction Format Example

Formal meaning.
In R-format: rs = alu_op1, rt = alu_op2, rd = dest reg
In R-format SLL or SRL : rs = des reg, rt = alu_op1, shamt = alu_op2
In I-format: rt = dest reg, rs = alu_op1, imm = alu_op2 ;
In I-format BEQ : rs = alu op1, rt = alu op2

Instruction Set

List all instructions, grouped by category (e.g., arithmetic, logical, memory, control). Include opcode, function codes, and descriptions.

Arithmetic Instructions

Instructions involving arithmetic operators +, -.

Format	**Instruction**	**Opcode**	**Fcode**	**Description**	**Example (CASM)**	
R	ADD	0000	000	Add rs and rt, store in rd	ADD $rd, $rs, $rt	
R	SUB	0000	001	Subtract rt from rs, store in rd	SUB $rd, $rs, $rt	
I	ADDI	0001	—	Add rs and immediate, store in rt	ADDI $rs, $rt, d'10	

Logical Instructions

Instructions involving logical operators &, |, <<, >>.

Format	Instruction	Opcode	Fcode	Description	Example (CASM)	
R	AND	0000	010	Bitwise AND rs and rt, store in rd	AND $rd, $rs, $rt	
R	OR	0000	011	Bitwise OR rs and rt, store in rd	OR $rd, $rs, $rt	
R	SLL	0000	101	Shift-left-logical rt and shamt, store in rs	SLL $rs, $rt, shamt	
R	SRL	0000	110	Shift-left-logical rt and shamt, store in rs	SRL $rs, $rt, shamt	

Comparison Instructions

Instructions involving relational operators >, ==.

Instruction	Opcode	Fcode	Description	Example (CASM)	Effect
SLT	0000	100	Set rd=1 if rs<rt	SLT $rd, $rs, $rt	rs<rt ? rd=
SLTI	0011	—	Set rt=1 if rs<IMM	SLTI $rs, $rt, IMM	rs<IMM ?

Control Flow Instructions

List instructions used for branching and jumps, including encoding and behavior.

Instruction	Opcode	Fcode	Description	Example (CASM)	Effect
BEQ	0110	—	Branch to PC+OFFSET+1 if rs == rt	BEQ $rs, $rt, OFFSET	rs==rt ?
J	0111	—	Jump to ADDR	J h'F	PC=IM
JAL	1000	—	Same as J but PC+1 is stored in $ra	JAL h'F	PC=IM
JR	0000	—	Jump to address in register rs	JR $rs	PC=rs

Memory Instructions

Load/store instructions to access data memory.

Instruction	Opcode	Fcode	Description	Example (CASM)	Effect
LW	0100	—	Load RAM[rs+OFFSET] in rt	LW $rs, $rt, OFFSET	rt=RAM[
SW	0101	—	Store rt in RAM[rs+offset]	SW $rs, $rt, OFFSET	RAM[rs+

Register Set

Describe the registers available in your ISA, including their names, sizes, and special purposes.

Register Name	Size (bits)	Purpose/Description
r0	32	Always zero & CONTROL reg
r1-r7	32	General Purpose
r8	32	Stack pointer
r14	32	Return address

Table 7: Register Set

Encoding Examples

Provide encoding examples for a few representative instructions.

Instruction	Opcode	rs	rt	Effect
ADD $rd, $rs, $rt	0110011	00010	00011	rd=rs + rt
SUB $rd, $rs, $rt	TBD	TBD	TBD	rd=rs - rt
ADDI $rs, $rt, d'10	TBD	TBD	TBD	rt=rs + 10
AND $rd, $rs, $rt	TBD	TBD	TBD	rd=rs AND rt
OR $rd, $rs, $rt	TBD	TBD	TBD	rd=rs OR rt
ANDI $rs, $rt, h'0A	TBD	TBD	TBD	rt=rs AND 0x0A
SLL $rs, $rt, d'2	TBD	TBD	TBD	rs=rt sll 2
SRL $rs, $rt, d'2	TBD	TBD	TBD	rs=rt srl 2
LW $rs, $rt, d'11	TBD	TBD	TBD	rt=RAM[rs+11]
SW $rs, $rt, d'11	TBD	TBD	TBD	RAM[rs+11]=rt
BEQ $rs, $rt, d'16	TBD	TBD	TBD	rs==rt ? PC=PC+16+1 : PC=PC+1
J d'11	TBD	TBD	TBD	PC=11
JR $rs	TBD	TBD	TBD	PC=rs
JAL d'7	TBD	TBD	TBD	PC=7 , ra=PC+1
SLT $rd, $rs, $rt	TBD	TBD	TBD	rs<rt ? rd=1 : rd=0
SLTI $rs, $rt, h'0A	TBD	TBD	TBD	rs<0x0A ? rt=1 : rt=0

Table 8: Encoding Examples

Additional Notes

CASM Immediate : immediates can be written in the following format b'XXX, h'YYY, d'ZZZ. Depends on the number base preferred. b' is for binary format, h' for hexadecimal format and 'd for decimal format. Immediates are 6-bits long, writing anything higher than 2**6-1 will result in truncation.
Examples: b'111; d'10; h'3F

REGISTER ZERO : MIPS arch includes a zero-register. Reading at zero-reg returns 0. Writing at zero-reg has no effect. This last sentence is not valid in myMIPS. Writing to zero-reg has the effect of writing to CONTROL-reg. CONTROL-reg handles minor configurations for the CPU.

REGISTER 0: CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	TO	PD	Z	DC	C
bit15							bit0

bit 15-1 Unimplemented: Maintain as '0'.

bit 0 C: Regfile bank bit
1 = BANK1, opens access to r9-r15
0 = BANK0, opens access to r1-r7

Note: A subtraction is executed by adding the two's complement of the second operand.
For rotate (`RRF, RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

R	Readable bit
W	Writable bit
U	Unimplemented bit, read as ‘0’
n	Value at POR
‘1’	Bit is set
‘0’	Bit is cleared
x	Bit is unknown