VHDL

Home About Contact

MIPS ISA 32-BIT MICROPROCESSOR

HARDWARE

We will deeply dive into the construction of MIPS microprocessor and learn the functionality of it and its instructions.

Get Started Play Video

What is MIPS ?

The MIPS 32-bit Instruction Set Architecture (ISA) is a RISC-based design that uses fixed 32-bit instructions and a load/store architecture. It features three instruction formats—R-type for register operations, I-type for immediate and memory access, and J-type for jump instructions.

Learn More

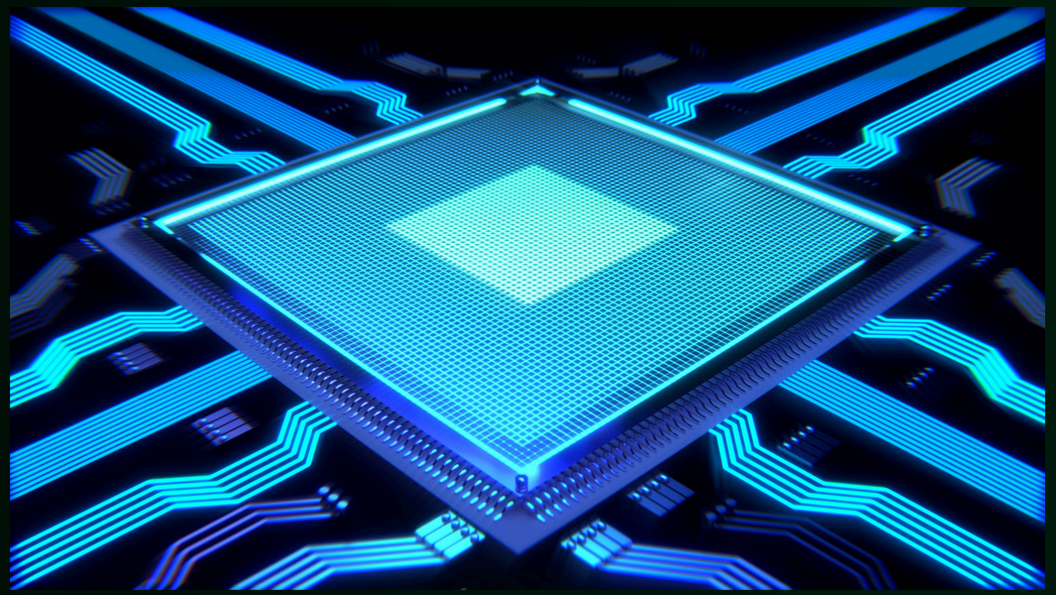
Introduction

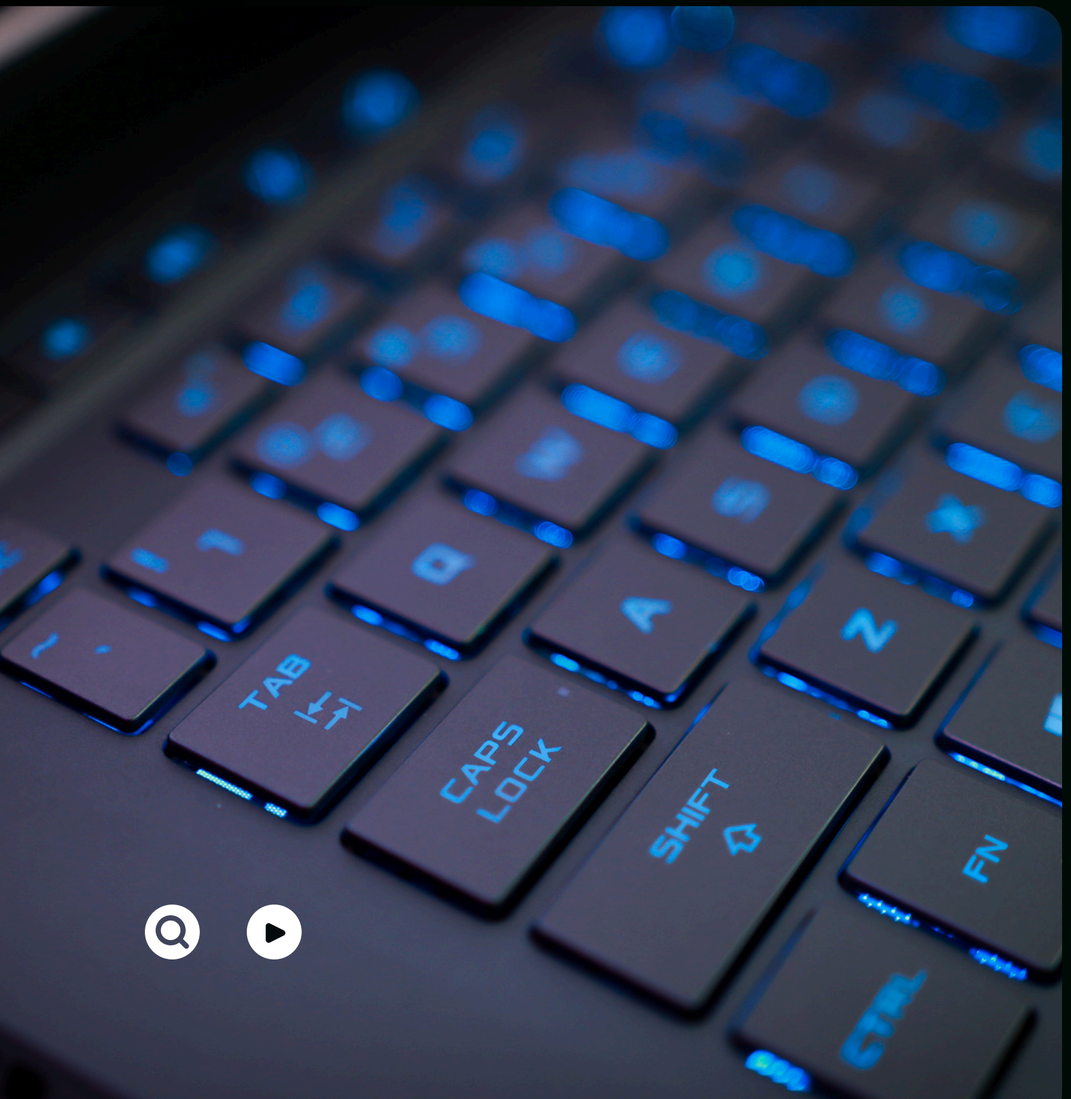
MIPS is a 32-bit RISC architecture known for its simplicity, fixed instruction format, and efficient execution. It uses a load/store model and is widely used in education and embedded systems.

Learn More

Key Features of MIPS

- 32-bit instruction width (fixed size)
- Load/store architecture
- 32 general-purpose registers ($0 to $31)
- Three instruction formats: R-type, I-type, J-type
- Simple and efficient pipelining





Instruction Types

R-type :
Register operations (e.g., add, sub)

I-type :
Immediate and memory operations (e.g., lw, addi)

J-type :
Jump operations (j, jal)

Instruction Format

- **R-type : opcode (6) | rs (5) | rt (5) | rd (5) | shamt (5) | funct (6)**
- **I-type : opcode (6) | rs (5) | rt (5) | immediate (16)**
- **J-type : opcode (6) | address (26)**

Learn More

MIPS Registers

32 registers: each 32 bits :-

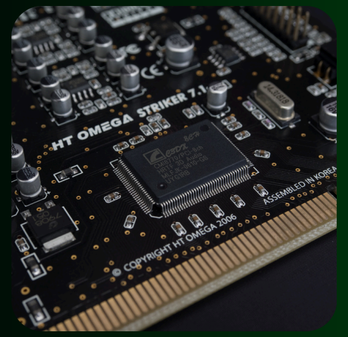
- $0: always 0
- $t0–$t9: temporaries
- $s0–$s7: saved across calls
- $a0–$a3: arguments
- $v0–$v1: return values
- $sp, $ra: stack pointer, return address

ALU Instructions

- add, sub, and, or, slt
- addi, andi, ori for immediate versions
- All use registers as operands
- No flag registers (unlike x86)

Memory Access Instructions

Only lw (load word), sw (store word) access memory
Load/store model
Syntax: lw $t0, 4($sp)
Word-aligned addresses (multiples of 4)



Branch and Jump Instructions

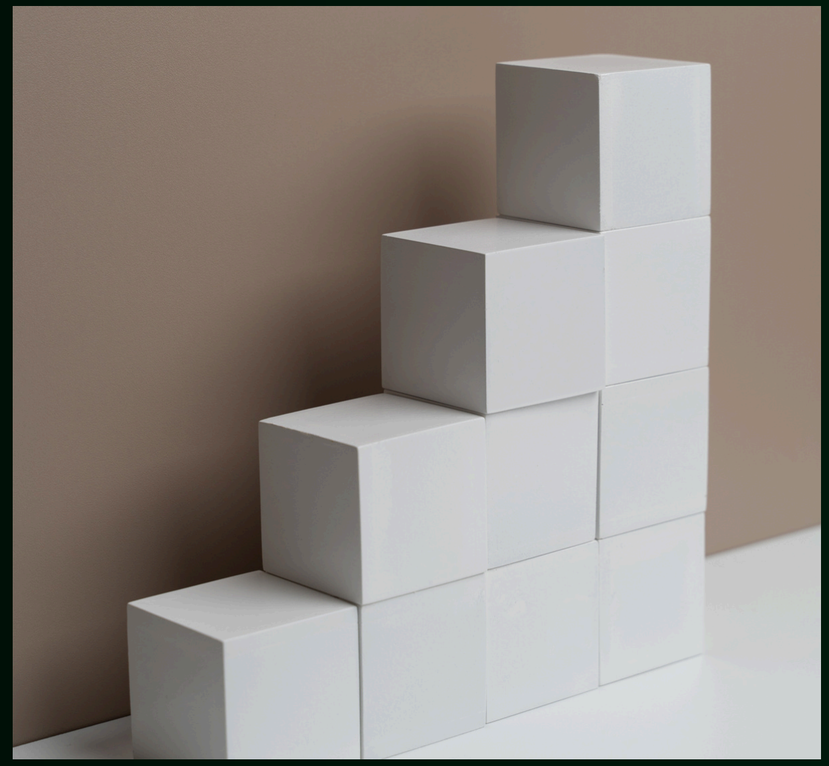
1. Conditional branches: beq, bne
2. Unconditional jumps: j, jal, jr
3. PC-relative addressing for branche
4. Jump register (jr) for function returns

Function Calls and Stack

- Calling convention
- $a0–$a3 for arguments
- $v0, $v1 for return values
- $ra stores return address (jal)
- Stack used for local storage and saving registers

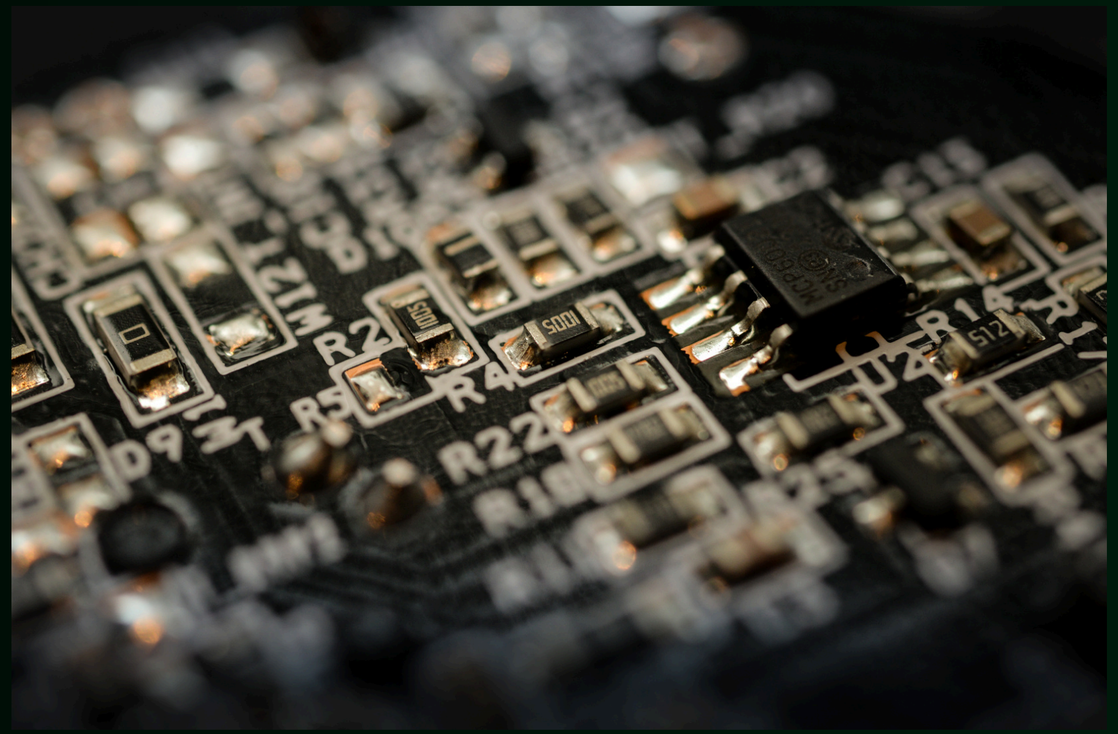
Pipelining in MIPS

- 5-stage pipeline: IF, ID, EX, MEM, WB
- Increases instruction throughput
- Simplifies control logic
- Hazards: data, control, structural
- Forwarding and hazard detection used



MIPS in Practice

- Widely used in embedded systems (routers, set-top boxes)
- Educational tool in computer architecture courses
- Implemented in simulators like MARS, QtSPIM
- Inspired modern RISC-V architecture



Advantages and Limitations

✅ Simplicity, speed, regular instruction format

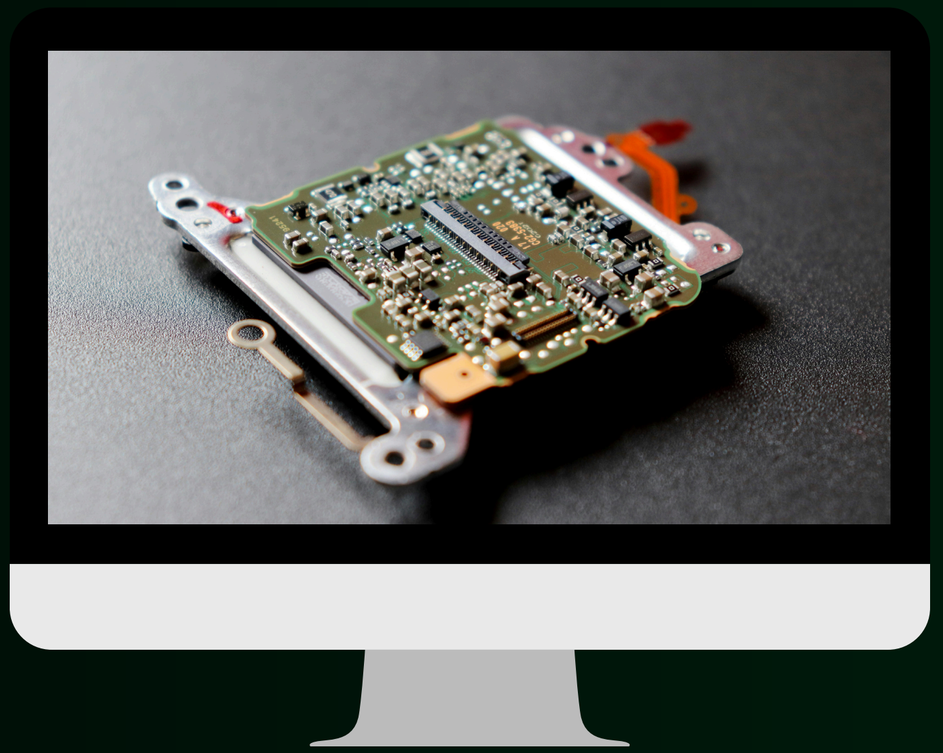
✅ Easy to learn and teach

❌ Not widely used in general-purpose desktop CPUs

❌ Less powerful than CISC in code density

Conclusion

- MIPS exemplifies the RISC design philosophy
- Strong foundation for understanding CPU architecture
- Continues to be relevant in teaching and specialized hardware
- Precursor to modern RISC architectures like RISC-V



THANK YOU

FOR YOUR ATTENTION