Based on the disassembled bytecode, you can tell that it’s produced by CPython’s virtual machine. The instructions such as RESUME, LOAD\_CONST, IMPORT\_NAME, MAKE\_FUNCTION, and the overall stack-oriented operations are characteristic of CPython’s stack-based bytecode interpreter, especially in recent versions (like Python 3.11, where you see new instructions such as RESUME).

Thus, the virtual machine you’re looking at is CPython’s virtual machine.

**1. SIMD (Single Instruction, Multiple Data) ISA**

✅ **Best Fit** for Matrix Multiplication.

**Why?**

* You're doing the *same operation* (+= A[i][k] \* B[k][j]) across large arrays of data.
* No data dependencies between loop iterations ⇒ perfect for vectorization.

**Instruction Architecture Features:**

* Vector ALUs (SIMD add, multiply)
* Data prefetching
* ISA examples: **Intel AVX**, **ARM NEON**, **RISC-V Vector Extension**

**How it works:**

text

CopyEdit

SIMD Instruction → Load vector of A[i][\*] and B[\*][j] → Multiply → Accumulate

**Ideal Strategy Based on Your Code:**

| **Workload Feature** | **Best Match ISA** |
| --- | --- |
| Loop-level independence | SIMD, MIMD |
| Uniform operations per row | SIMD (AVX, NEON) |
| Multi-core potential | MIMD (x86, CUDA) |
| Regular memory access | All above |
| Static schedule friendly | VLIW |

A close-up of a colorful box

AI-generated content may be incorrect.