“If you were **plowing** a field, which would you rather use? Two strong oxen or 1024 **chickens**?”

- Seymour Cray

SPEEDRUN INTO MASSIVE DATA

SPEEDRUN INTO MASSIVE DATA

SPEEDRUN INTO MASSIVE DATA

> START GAME

OPTIONS

EXIT GAME

SPEEDRUN INTO MASSIVE DATA

> START GAME

OPTIONS

EXIT GAME

Name

> _ _ _ _ _ _

BitSpire

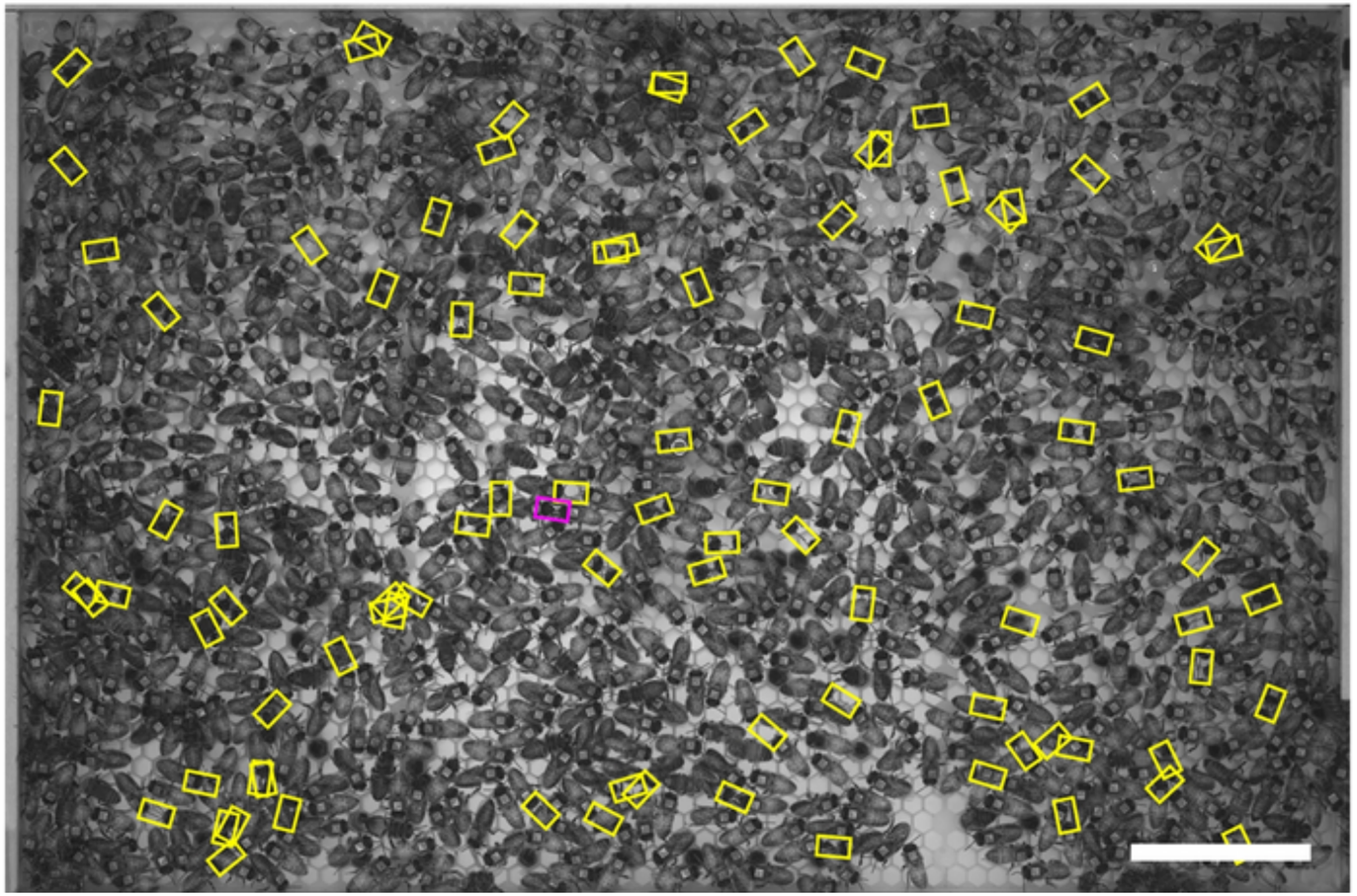
MVP

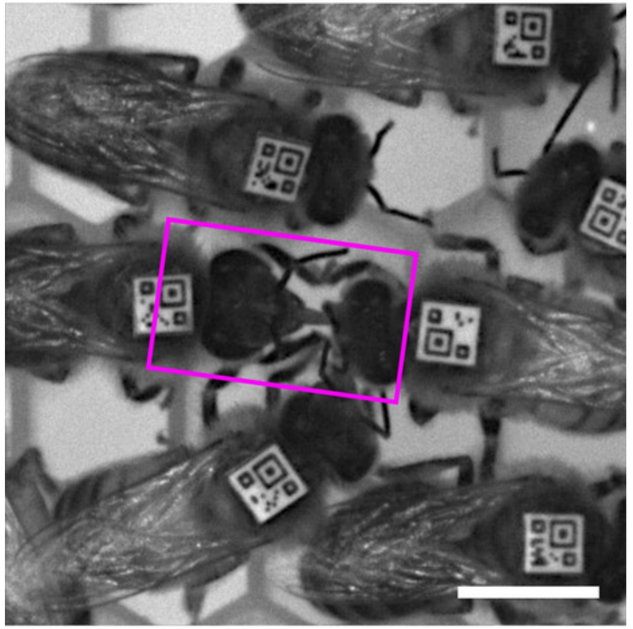
Name

> S I E V E N

https://steven-giesel.com

MOTIVATION





MOTIVATION

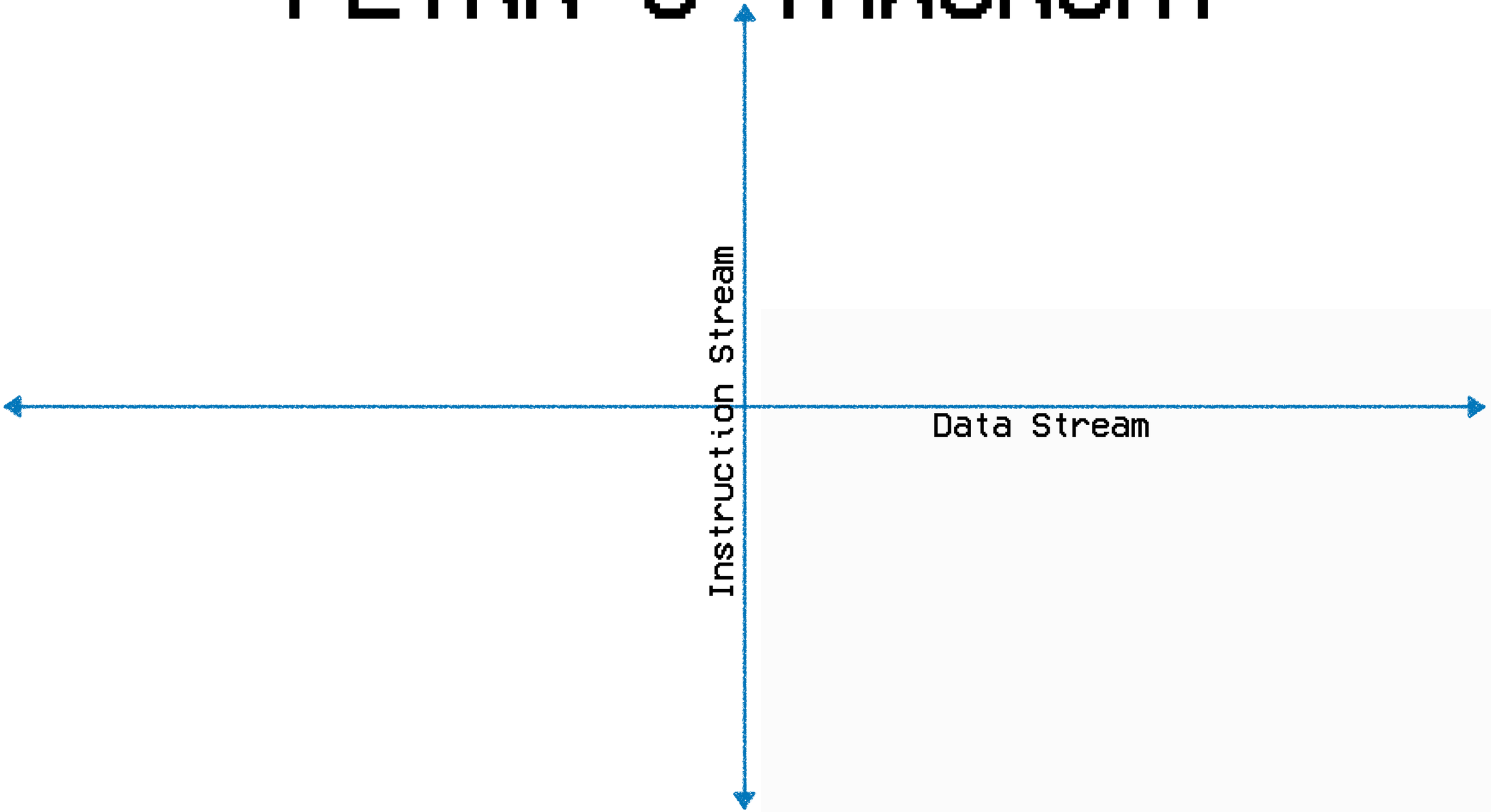
```
public readonly partial struct Guid
    private static bool EqualsCore(in Guid left, in Guid right)
        if (Vector128.IsHardwareAccelerated)
        {
            return Vector128.LoadUnsafe( source: ref Unsafe.As<Guid, byte>
        }
```

```
public static partial class Enumerable
    public static double Average(this IEnumerable<int> source)
        if (source.TryGetSpan(out ReadOnlySpan<int> span))
            if (Vector.IsHardwareAccelerated && span.Length >=
            {
```

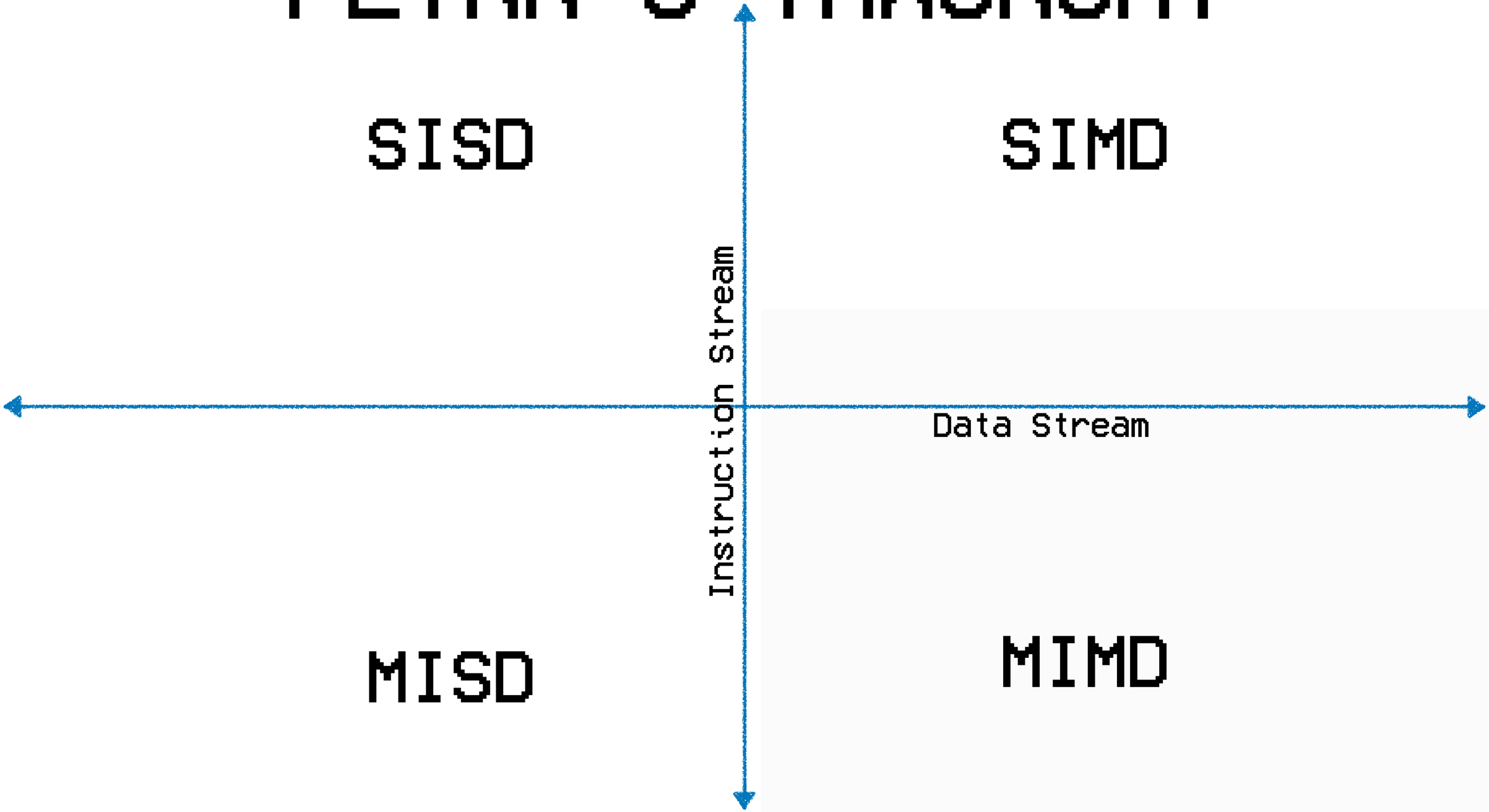
```
public static partial class Convert
    public static string ToBase64String(Rea
        if (Vector128.IsHardwareAccelerated
        {
```

FLYNN'S TAXONOMY

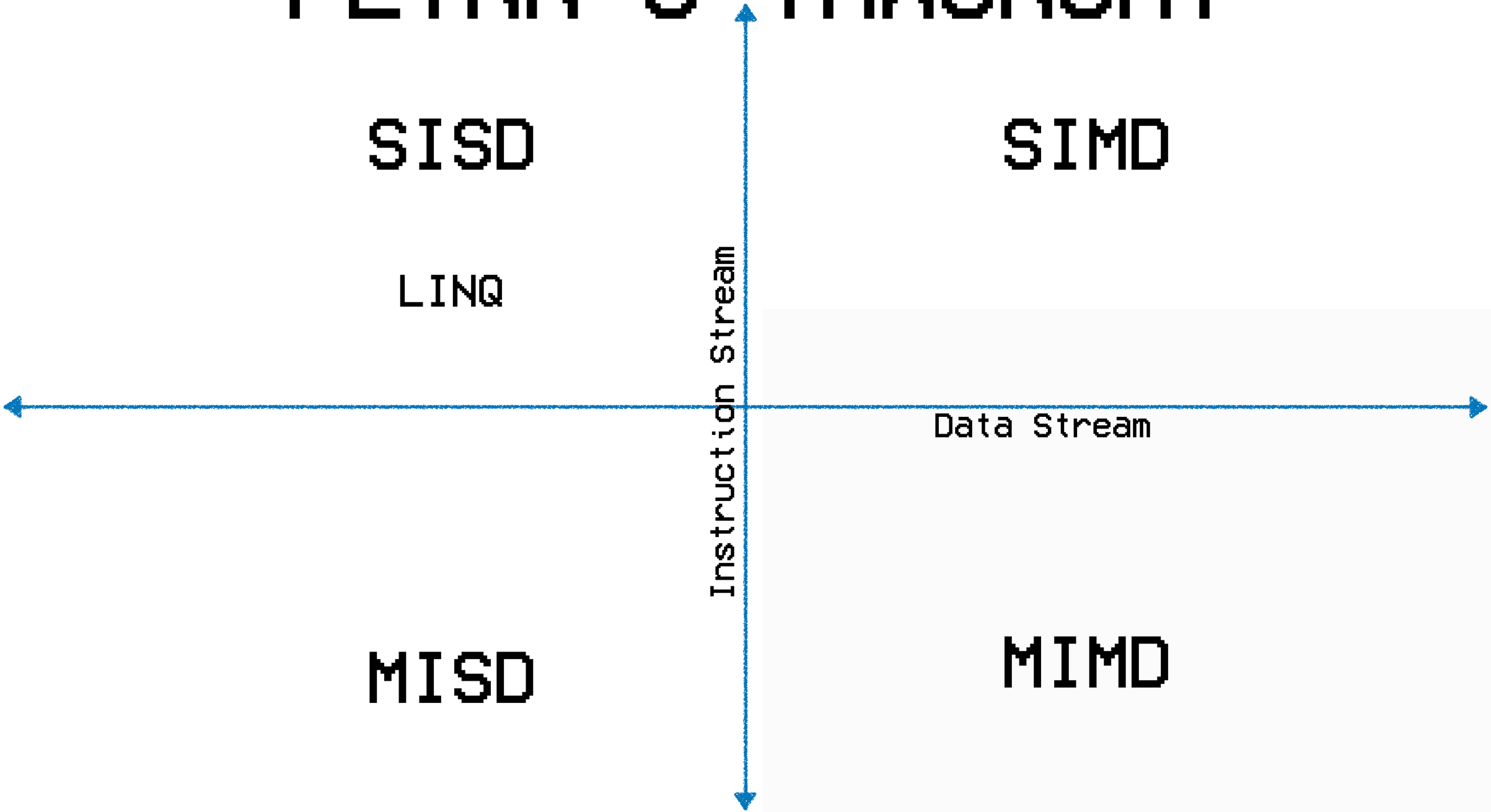
FLYNN'S TAXONOMY



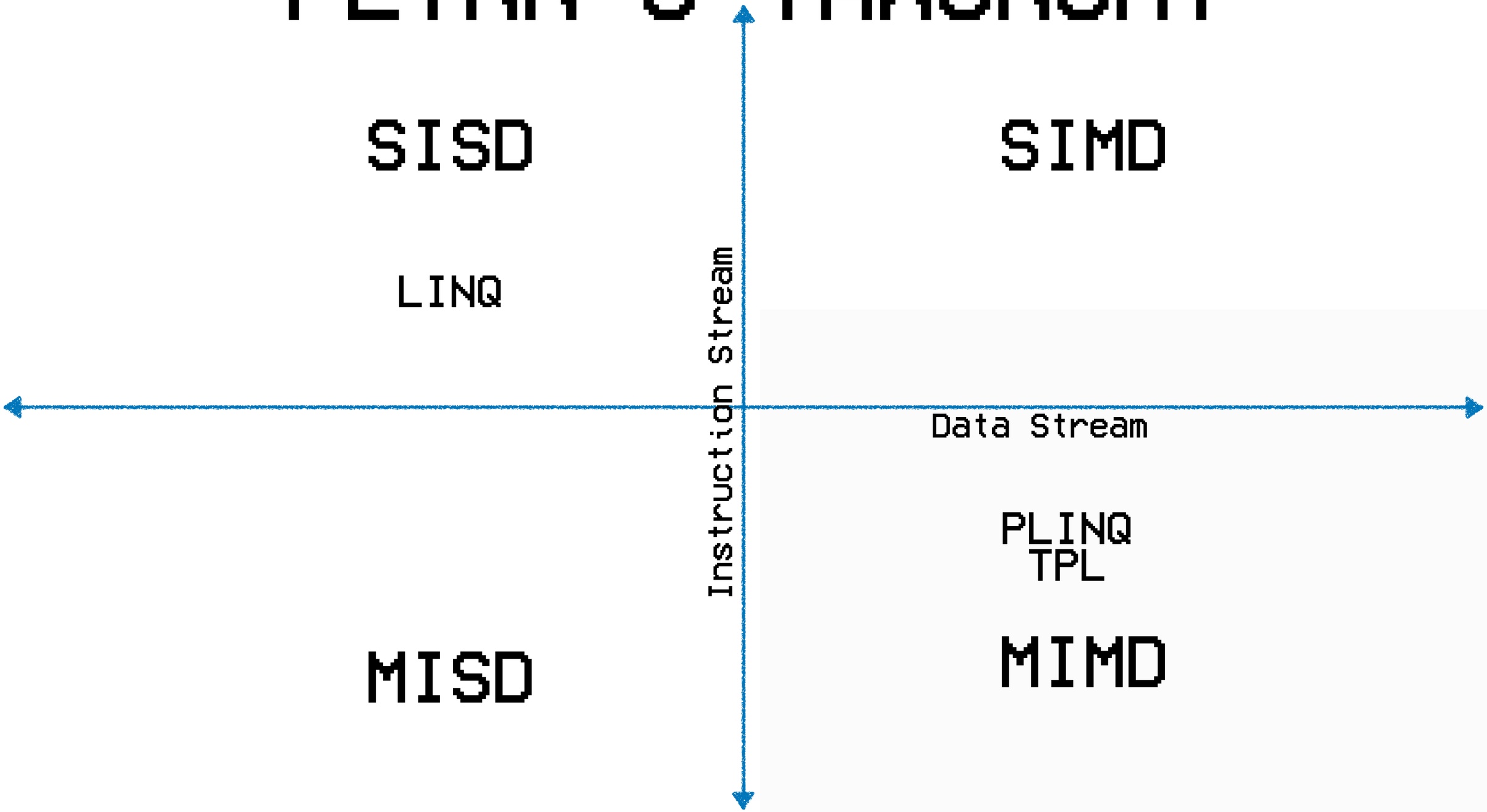
FLYNN'S TAXONOMY



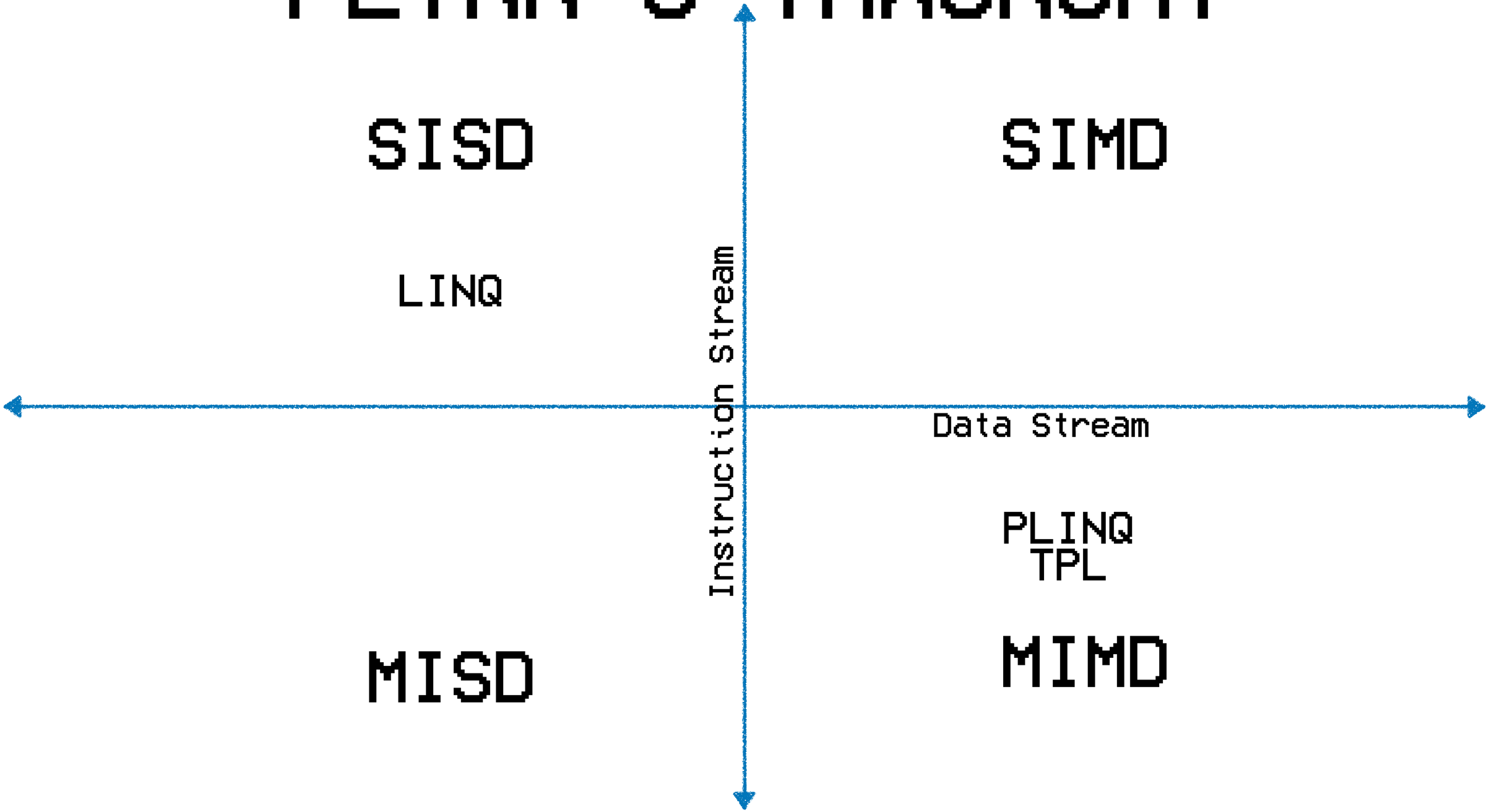
FLYNN'S TAXONOMY



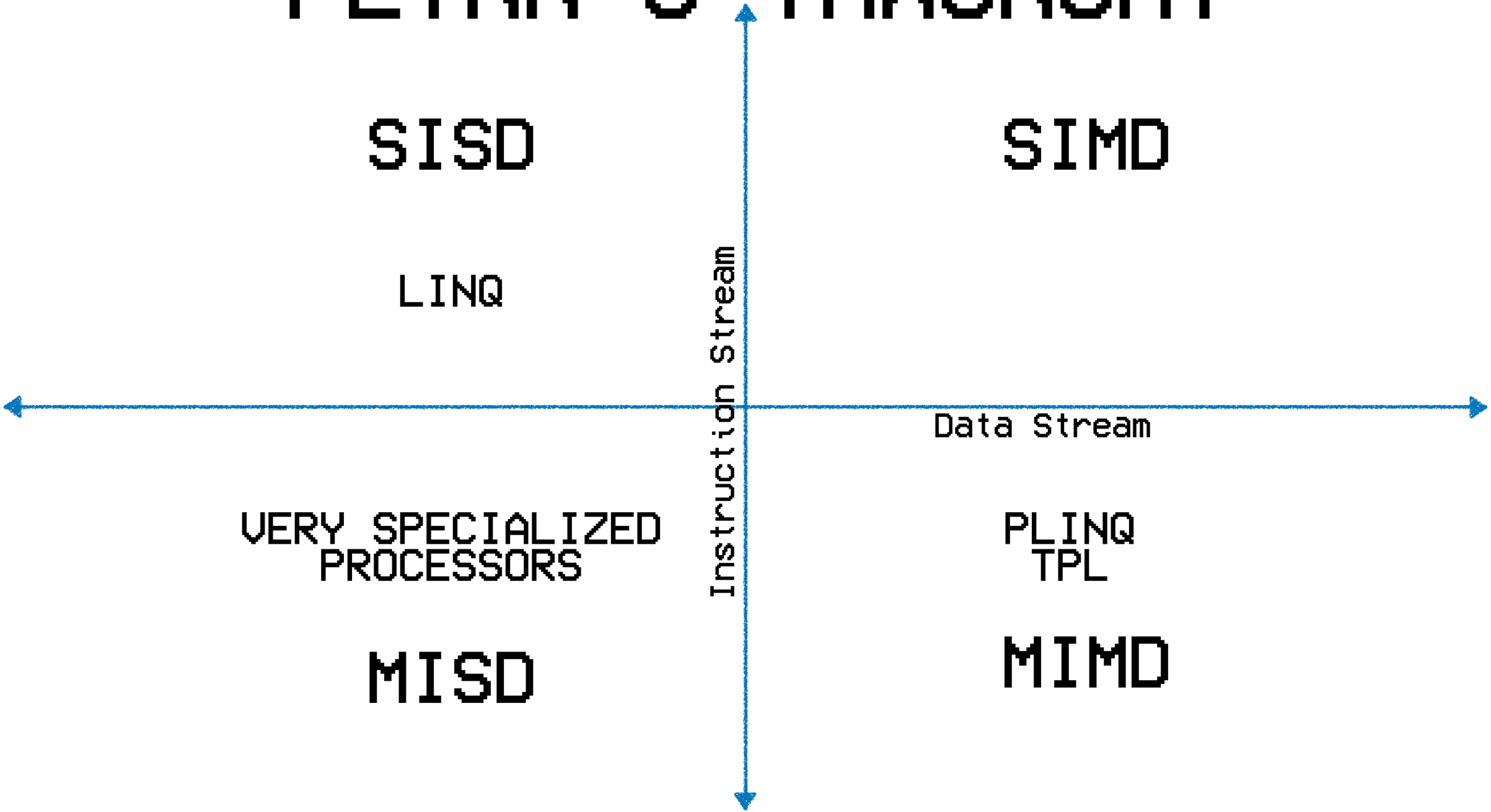
FLYNN'S TAXONOMY



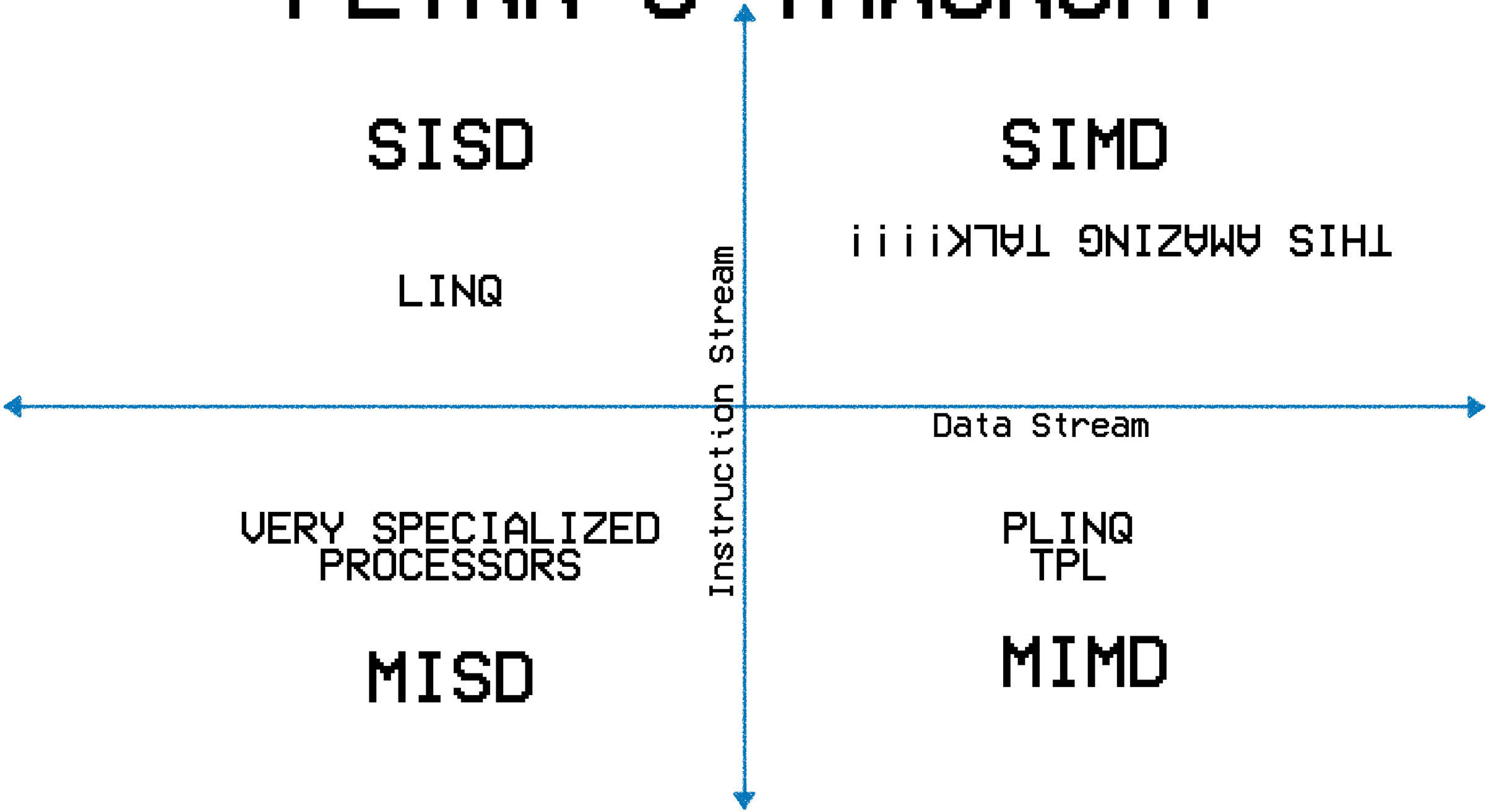
FLYNN'S TAXONOMY



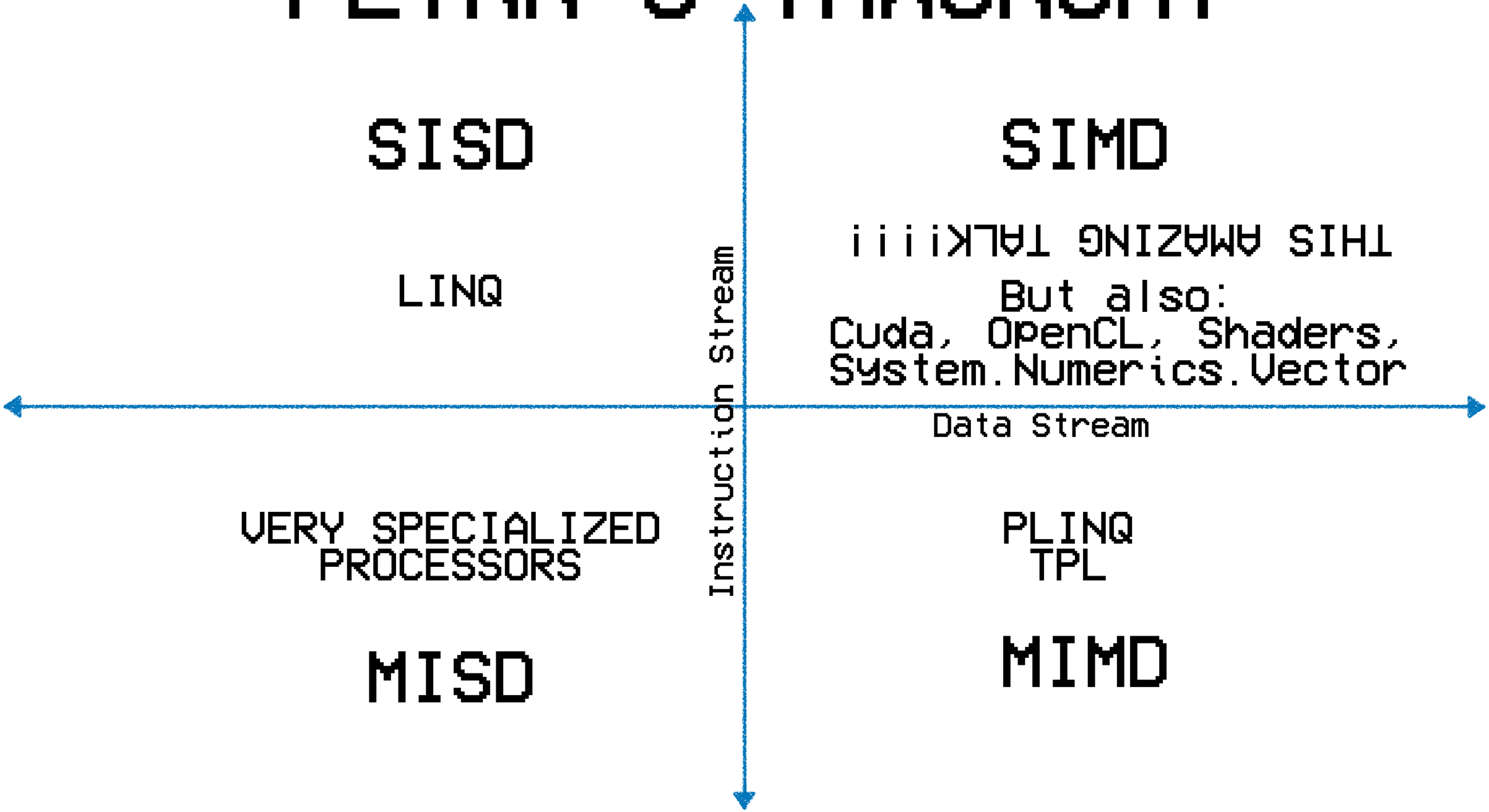
FLYNN'S TAXONOMY

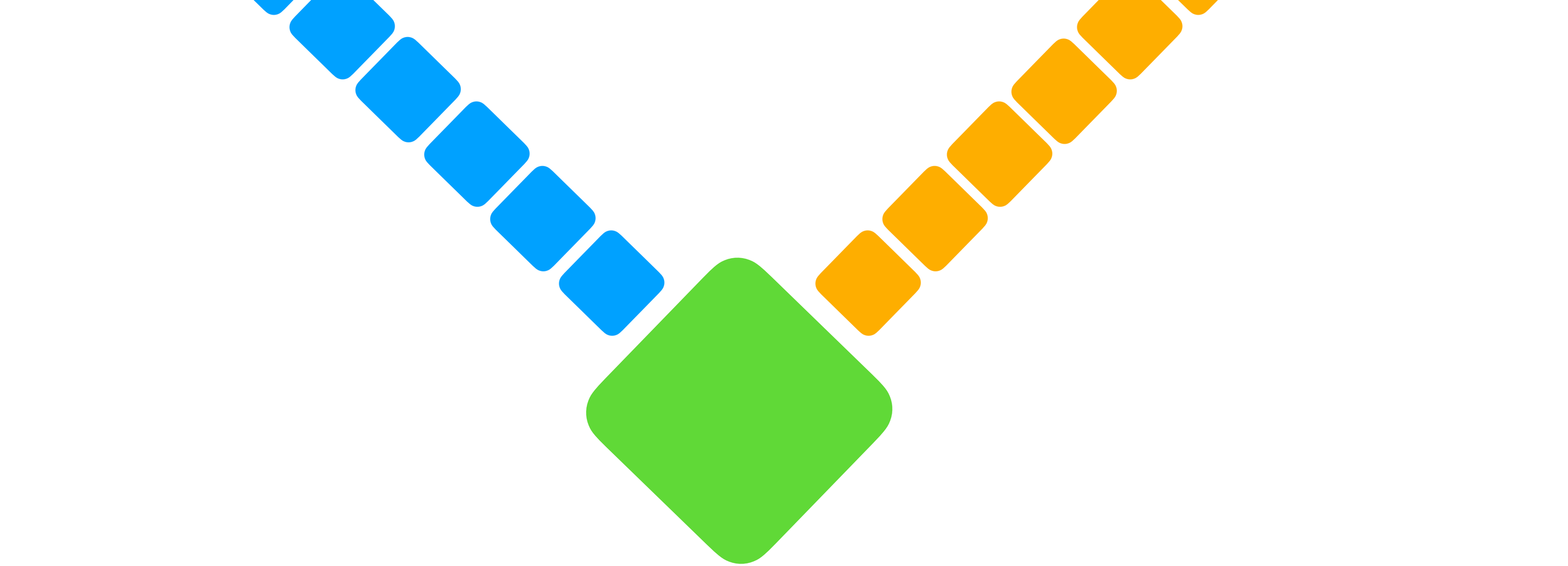


FLYNN'S TAXONOMY

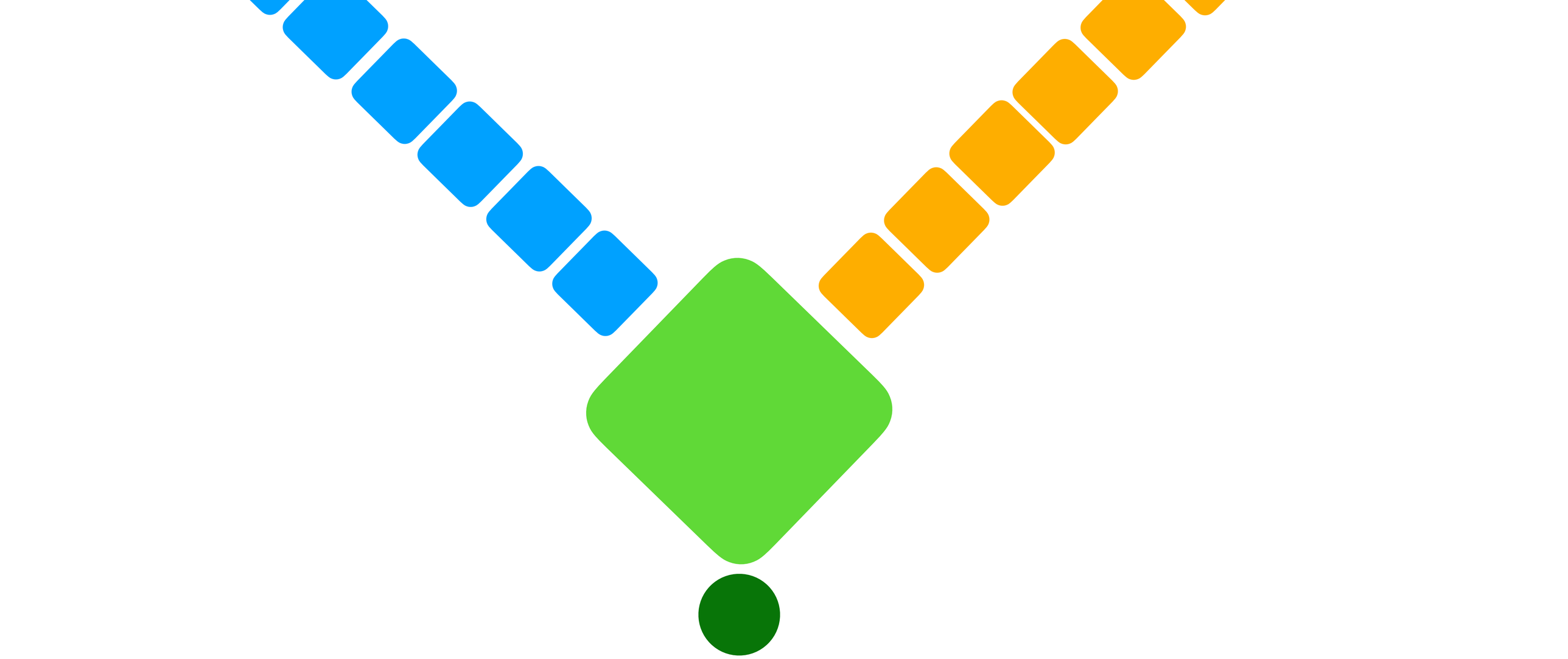


FLYNN'S TAXONOMY

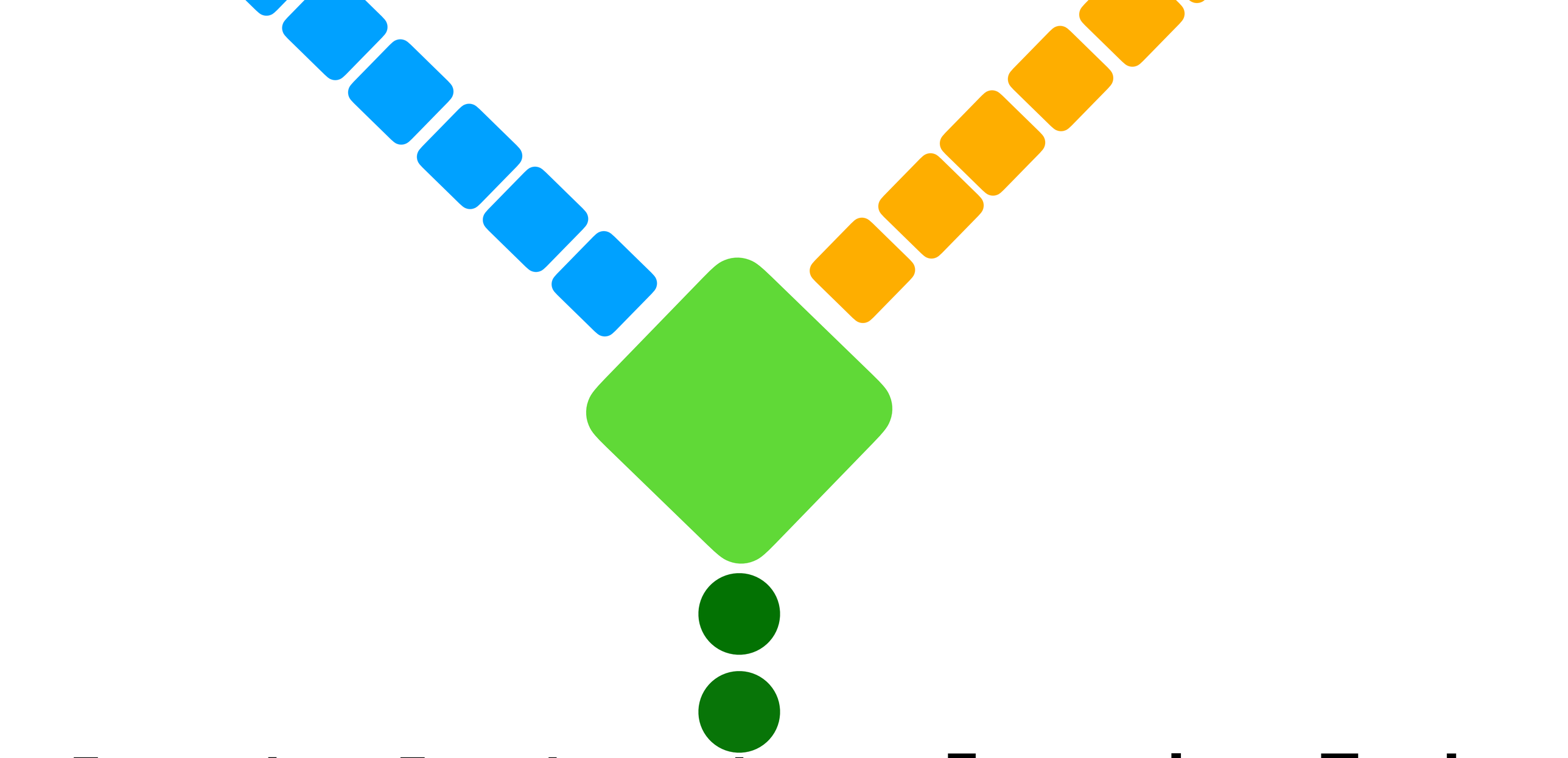


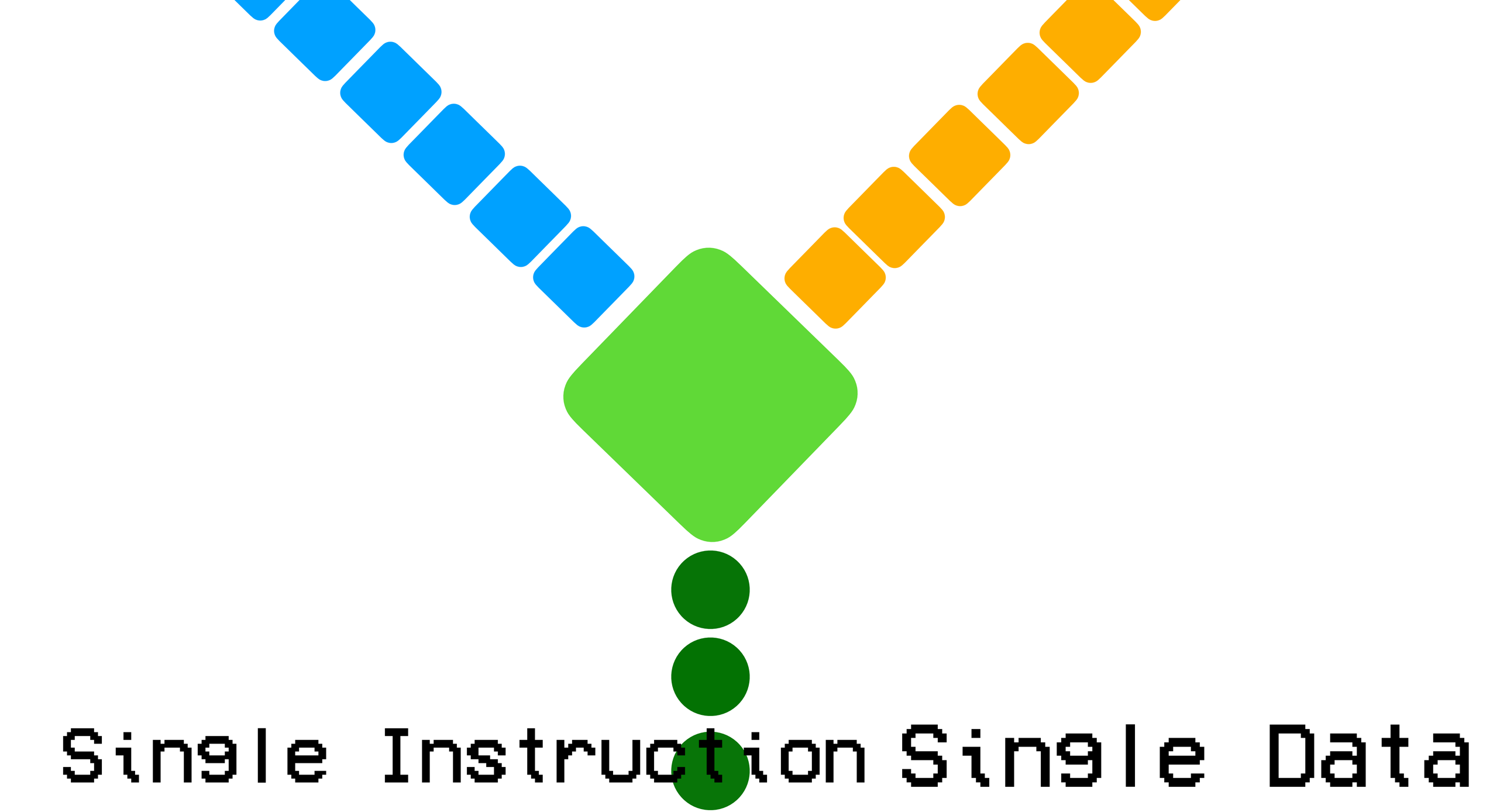


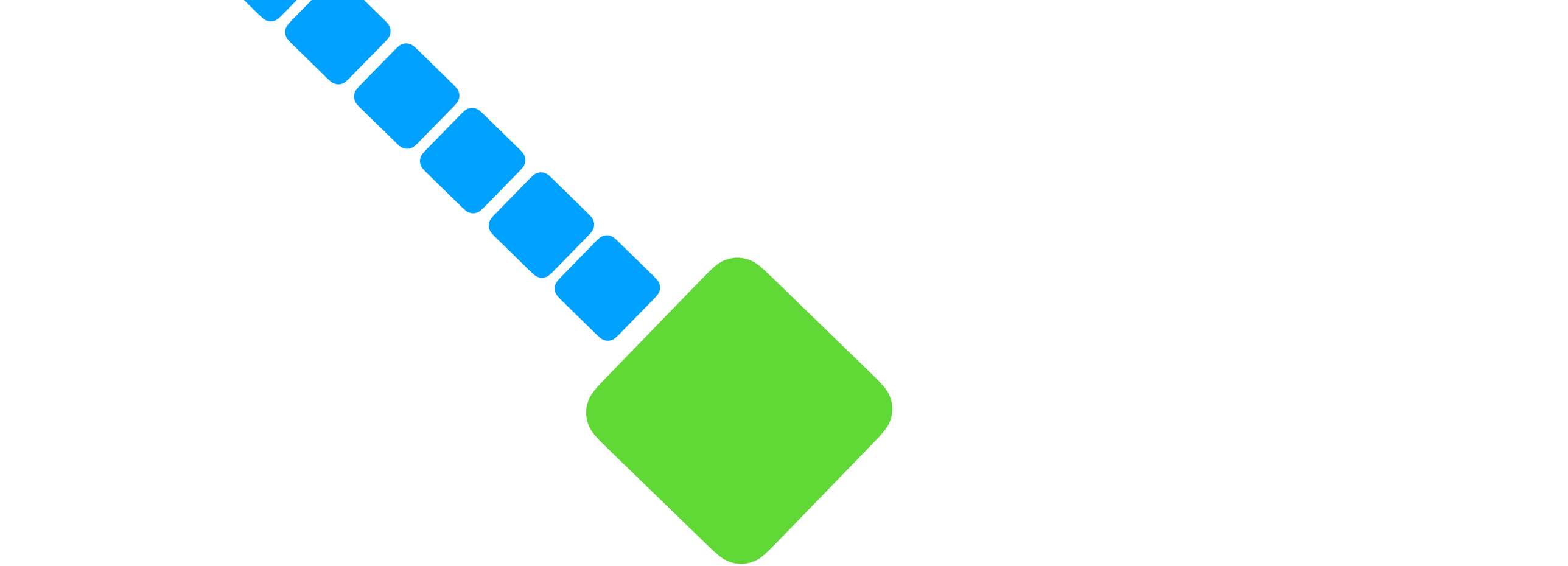
Single Instruction Single Data



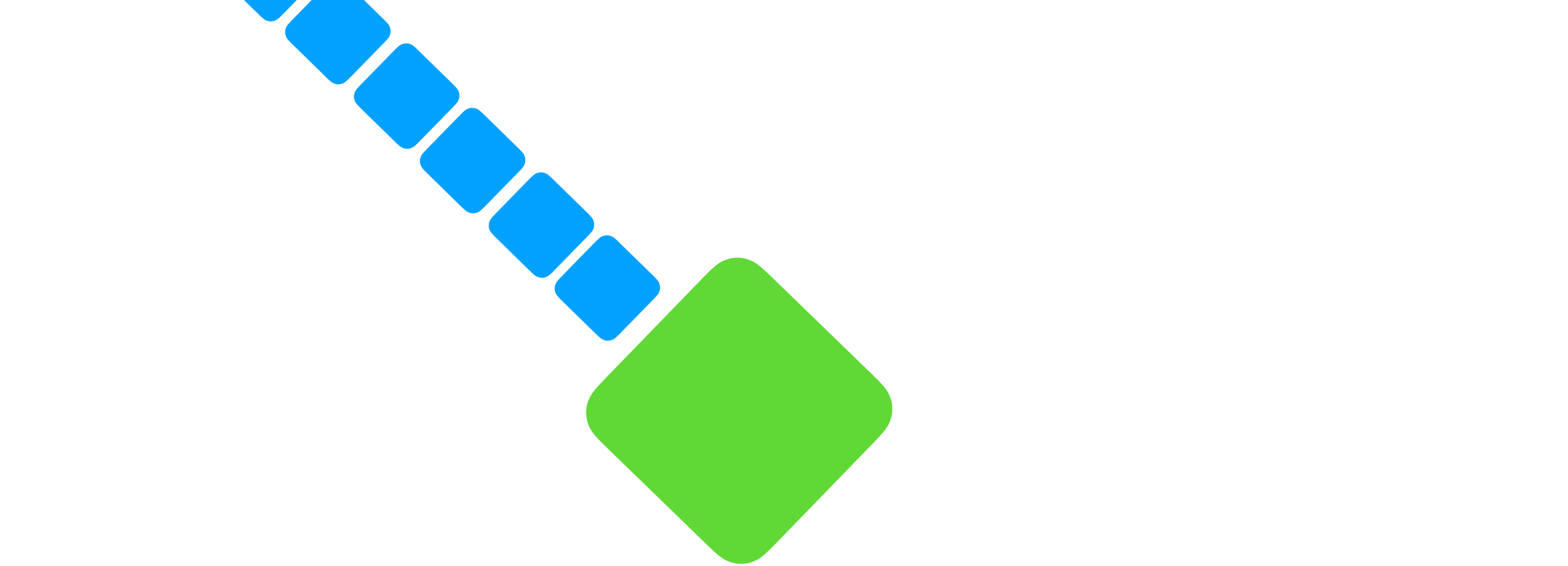
Single Instruction Single Data


Single Instruction Single Data

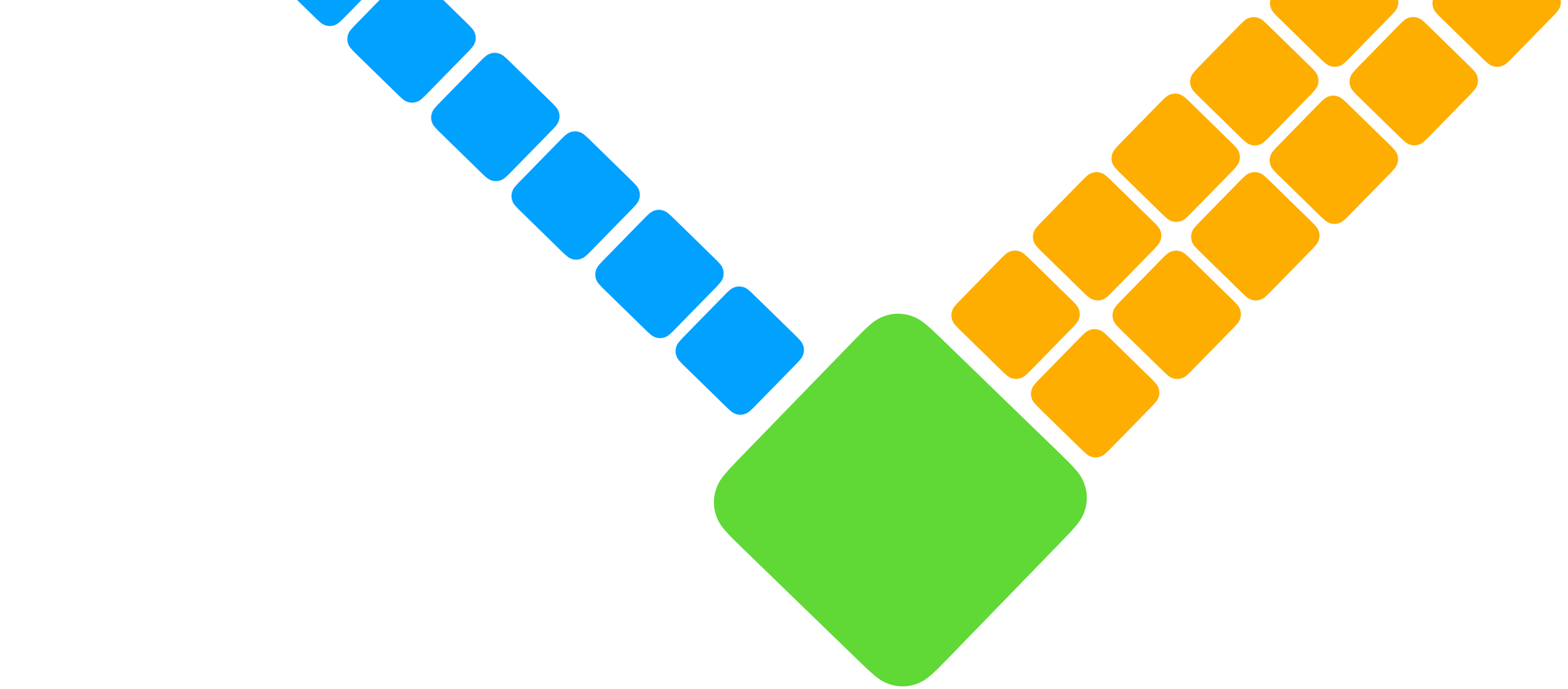

Single Instruction Single Data



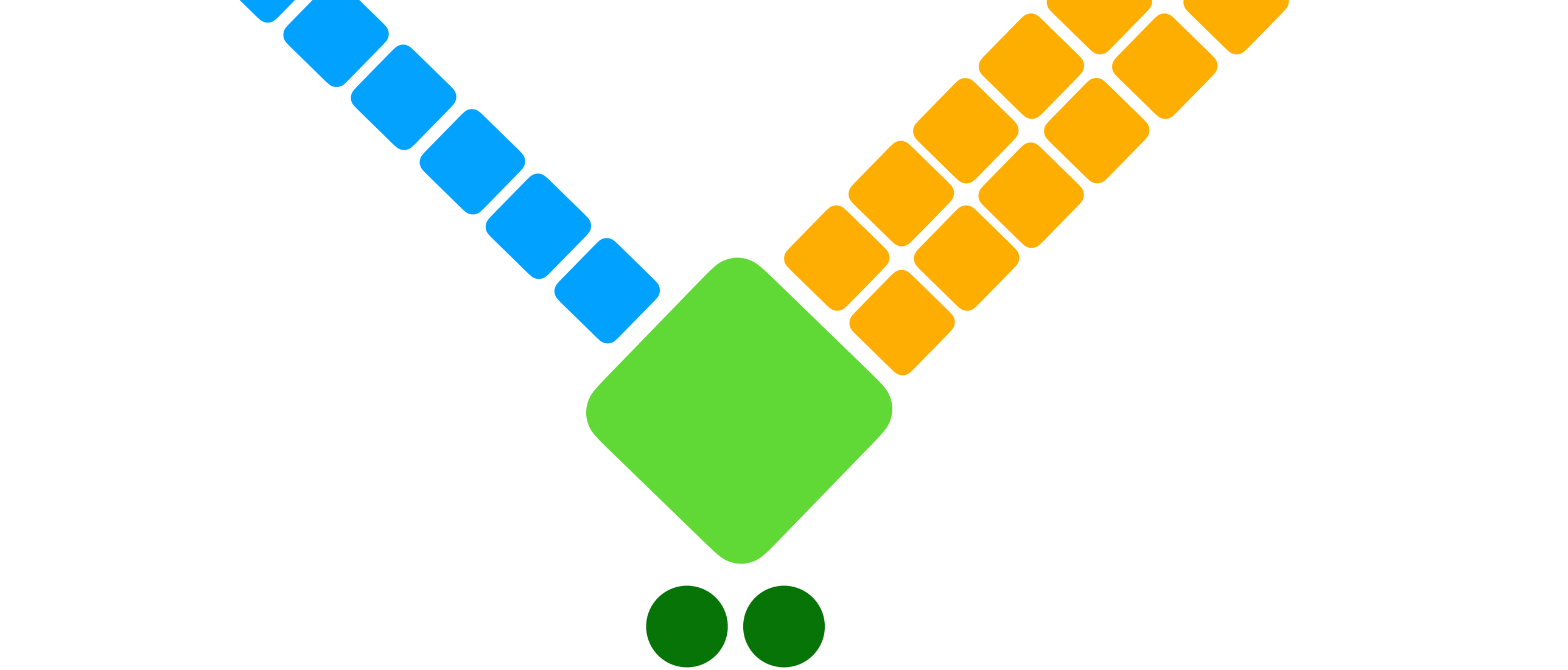
Single Instruction Multiple Data



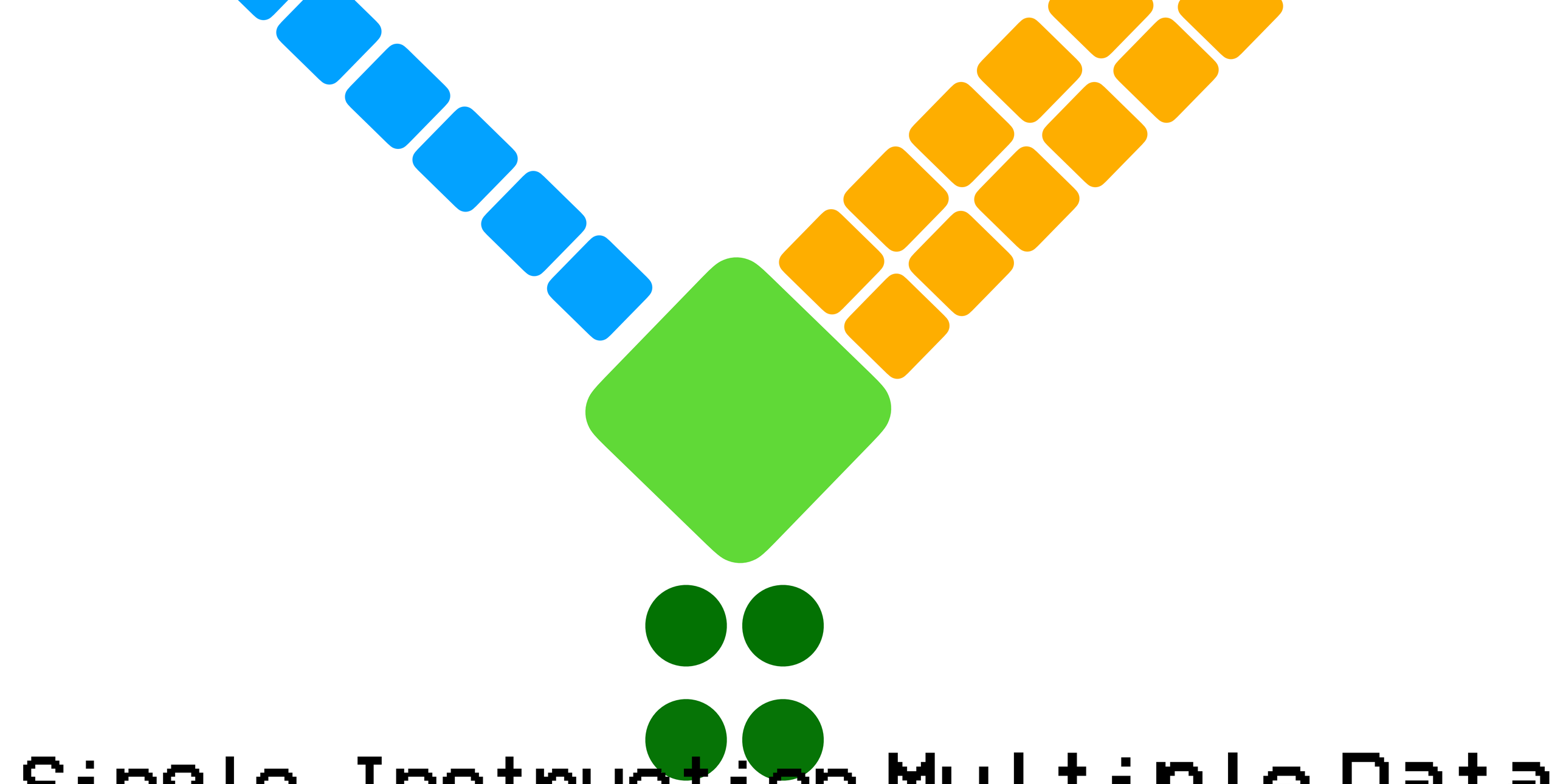
Single Instruction Multiple Data

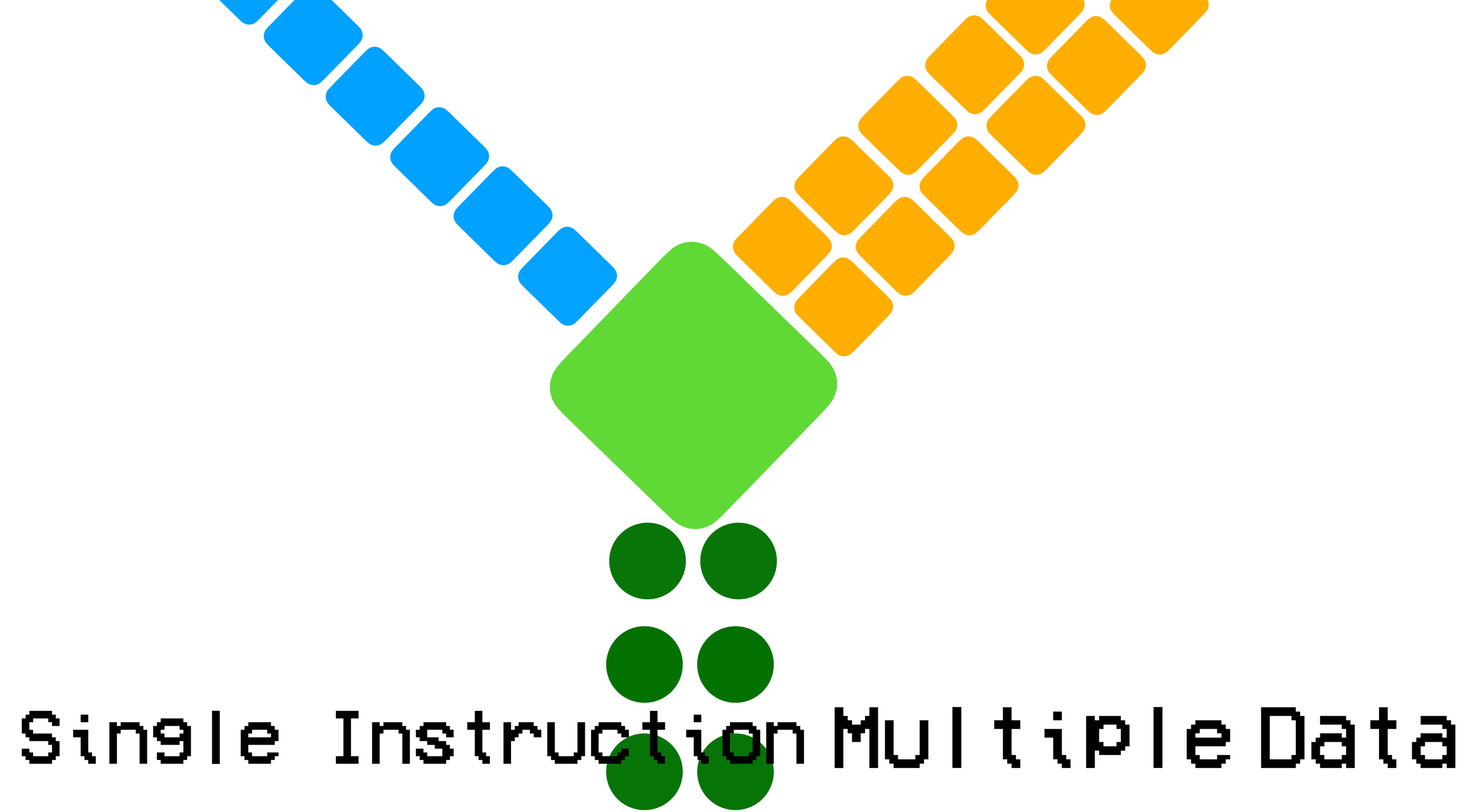


Single Instruction Multiple Data



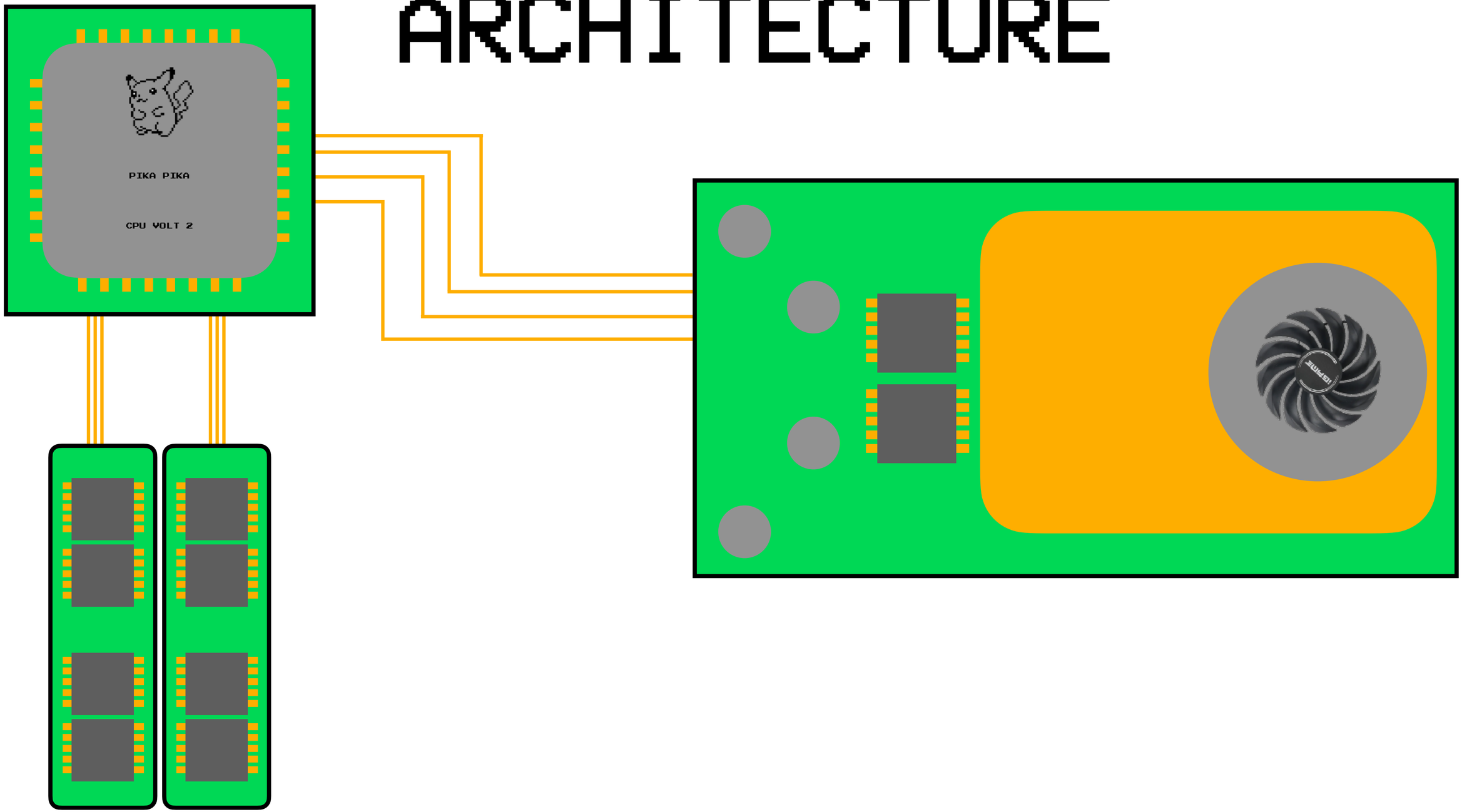
Single Instruction Multiple Data


Single Instruction Multiple Data

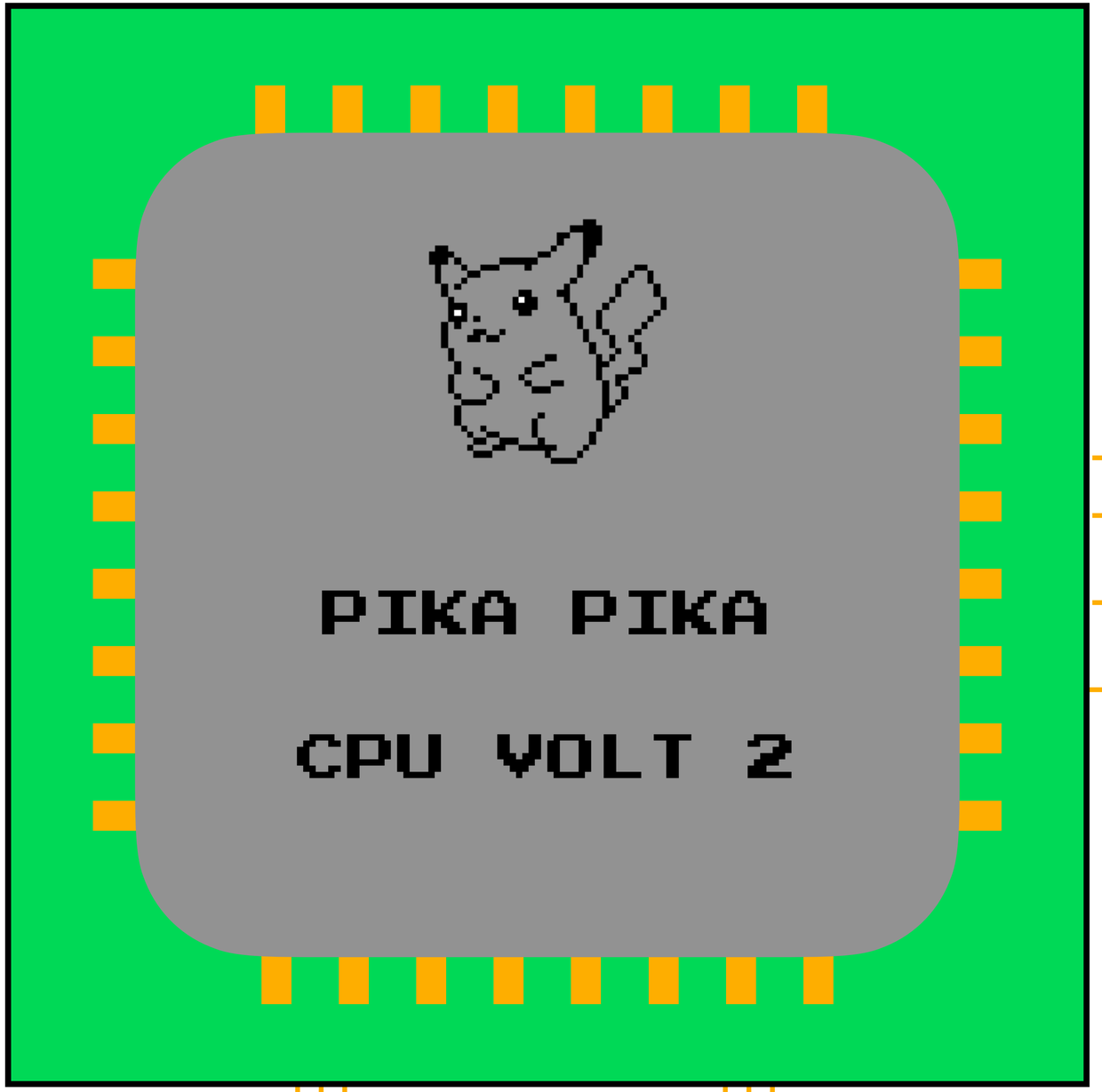

Single Instruction Multiple Data

ARCHITECTURE

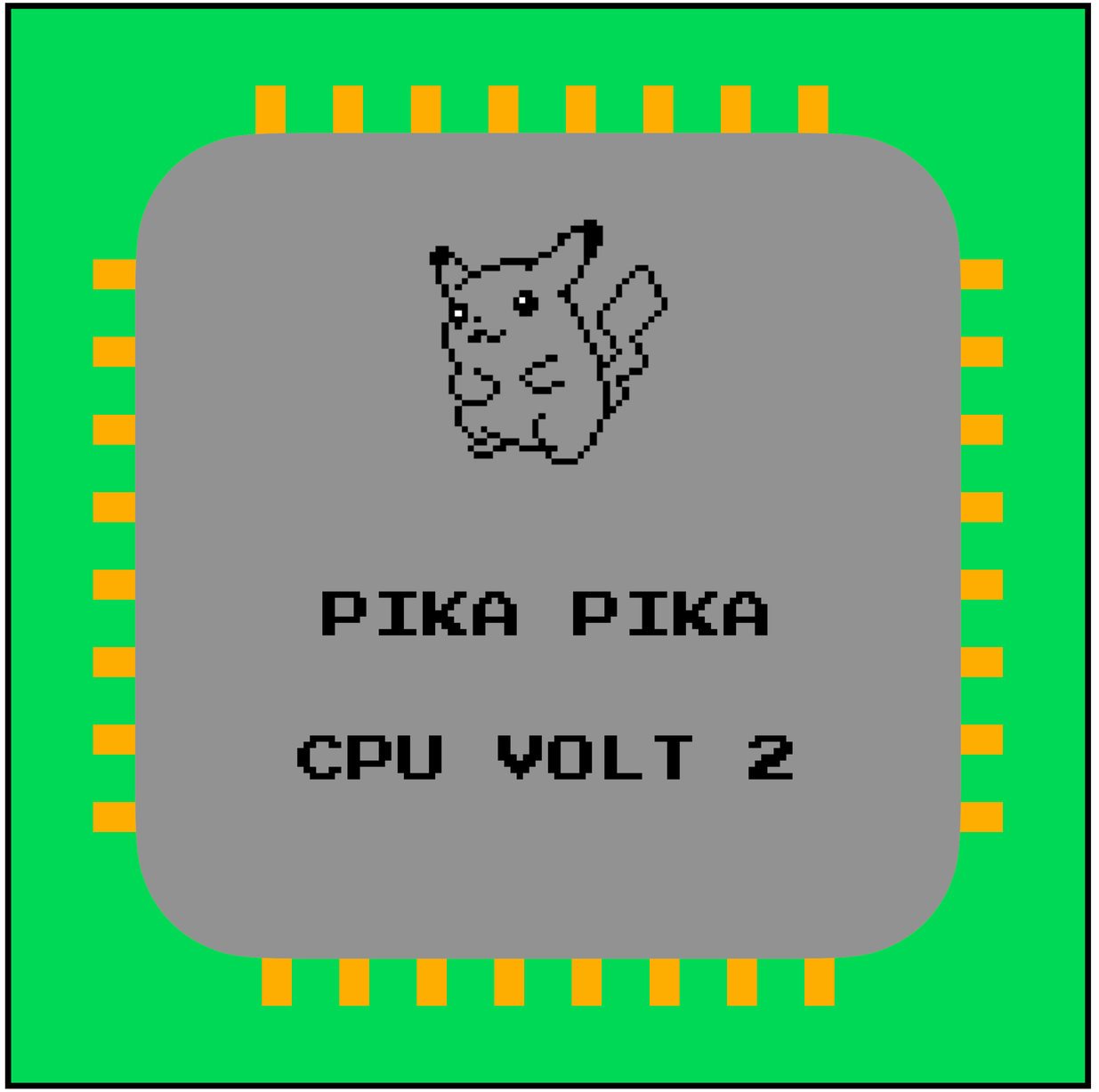
ARCHITECTURE



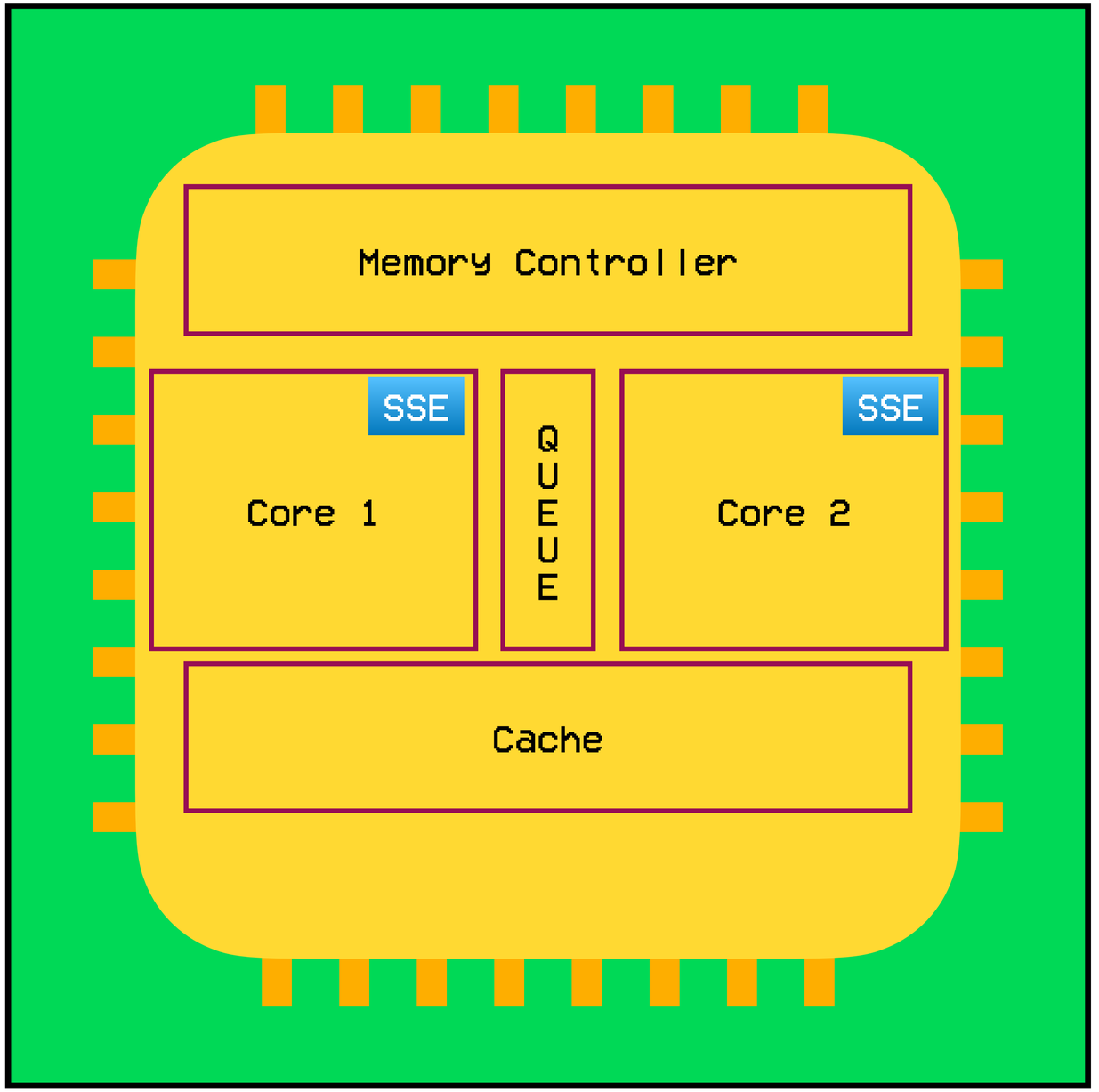
ARCHITECTURE



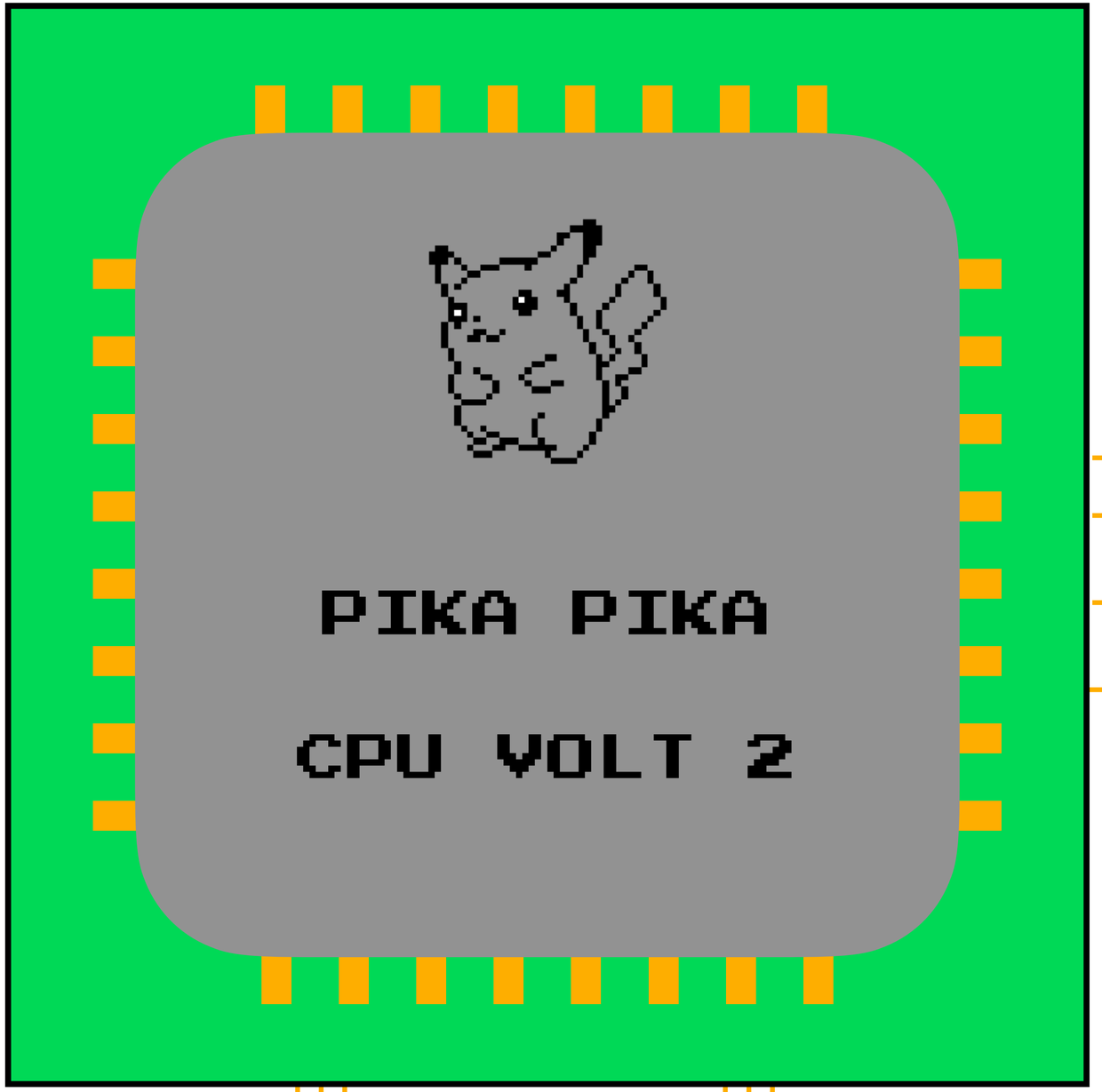
ARCHITECTURE



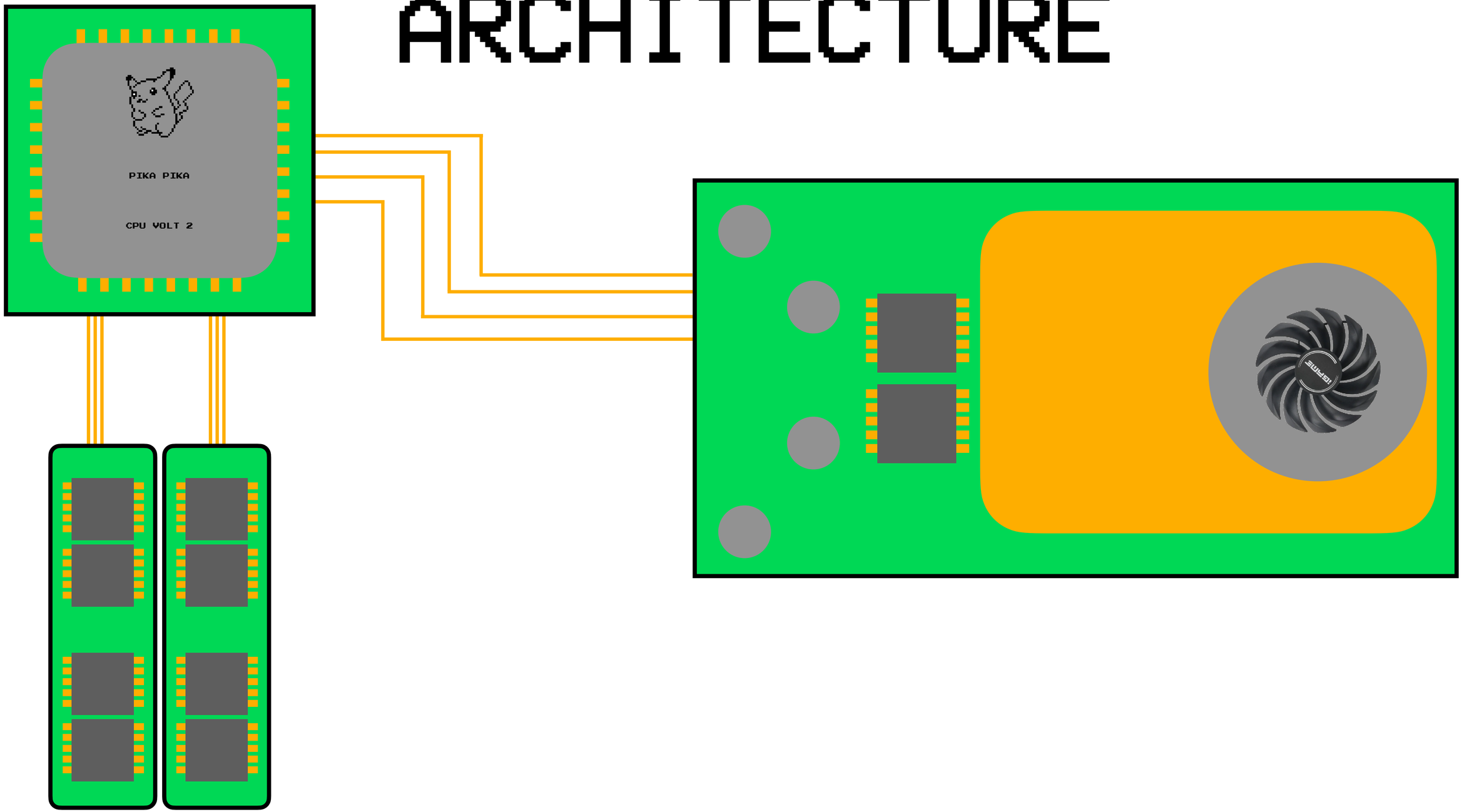
ARCHITECTURE



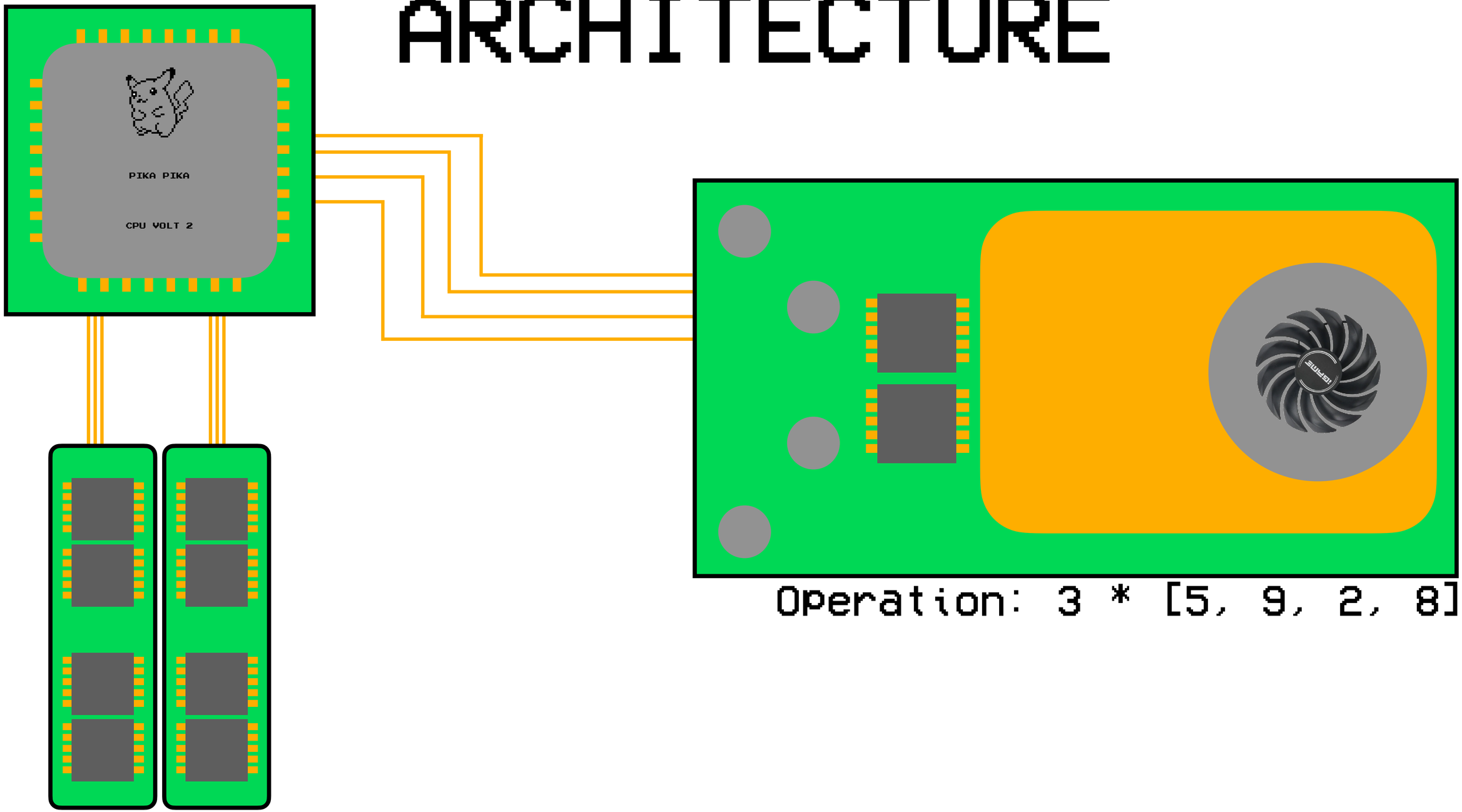
ARCHITECTURE



ARCHITECTURE

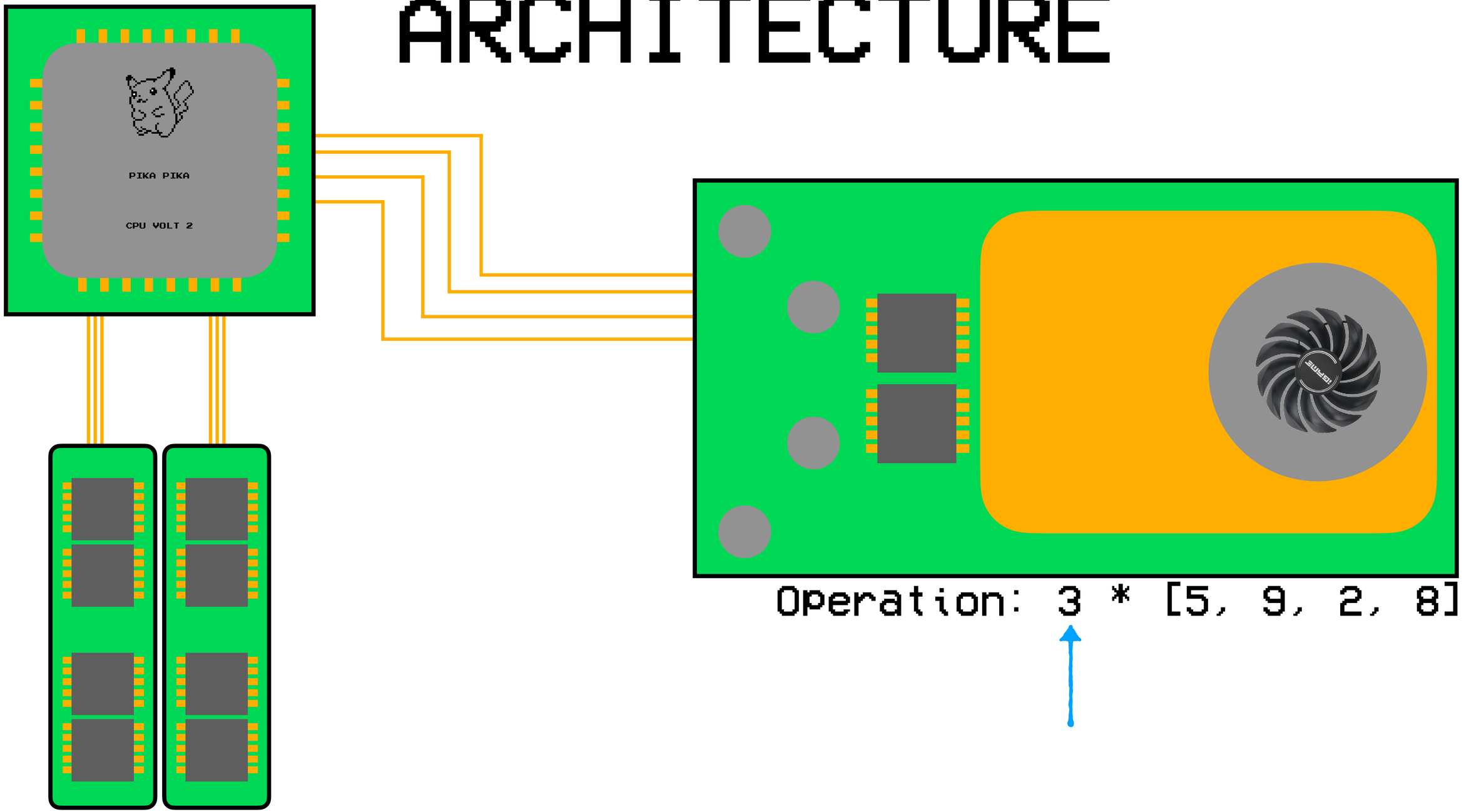


ARCHITECTURE

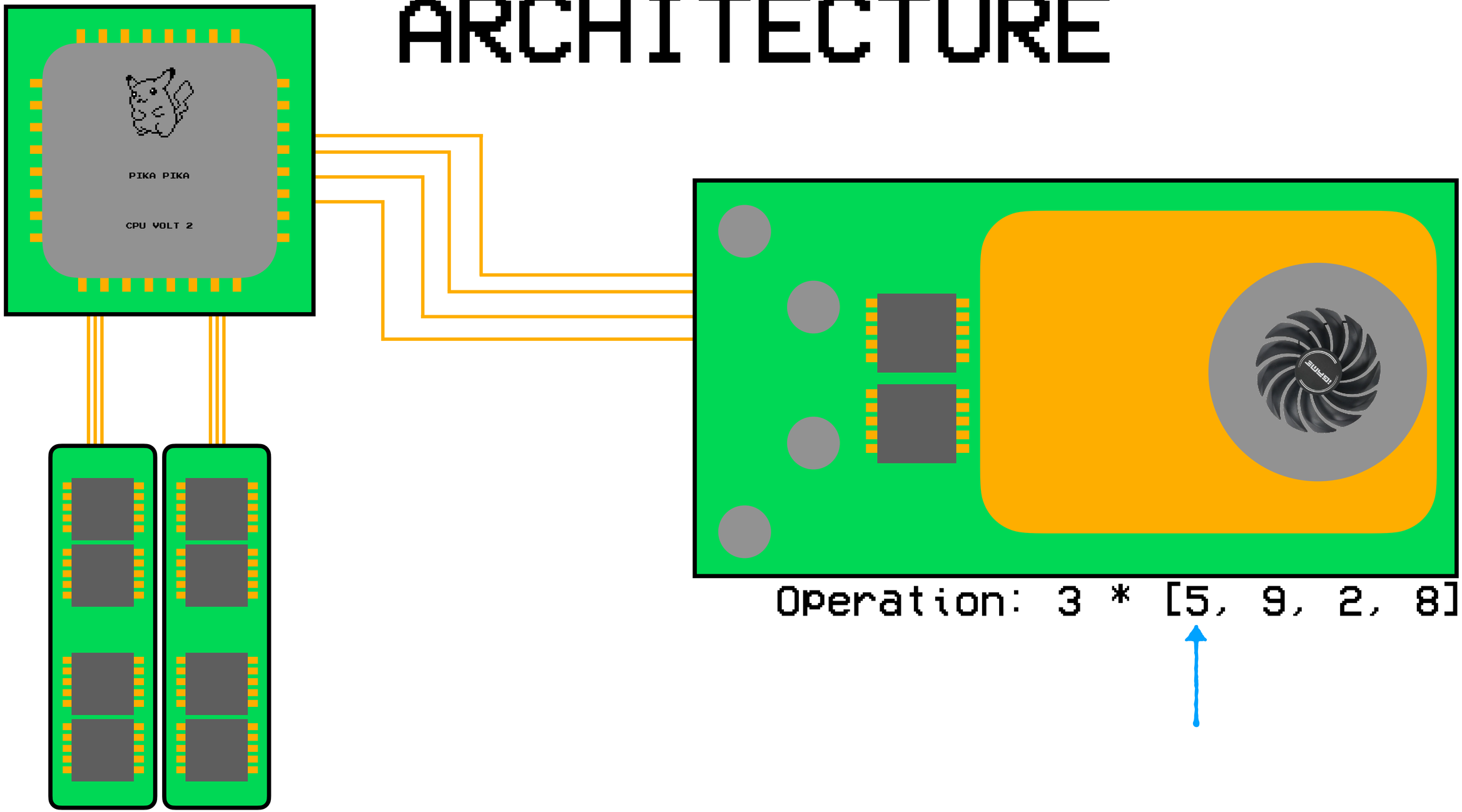


Operation: 3 * [5, 9, 2, 8]

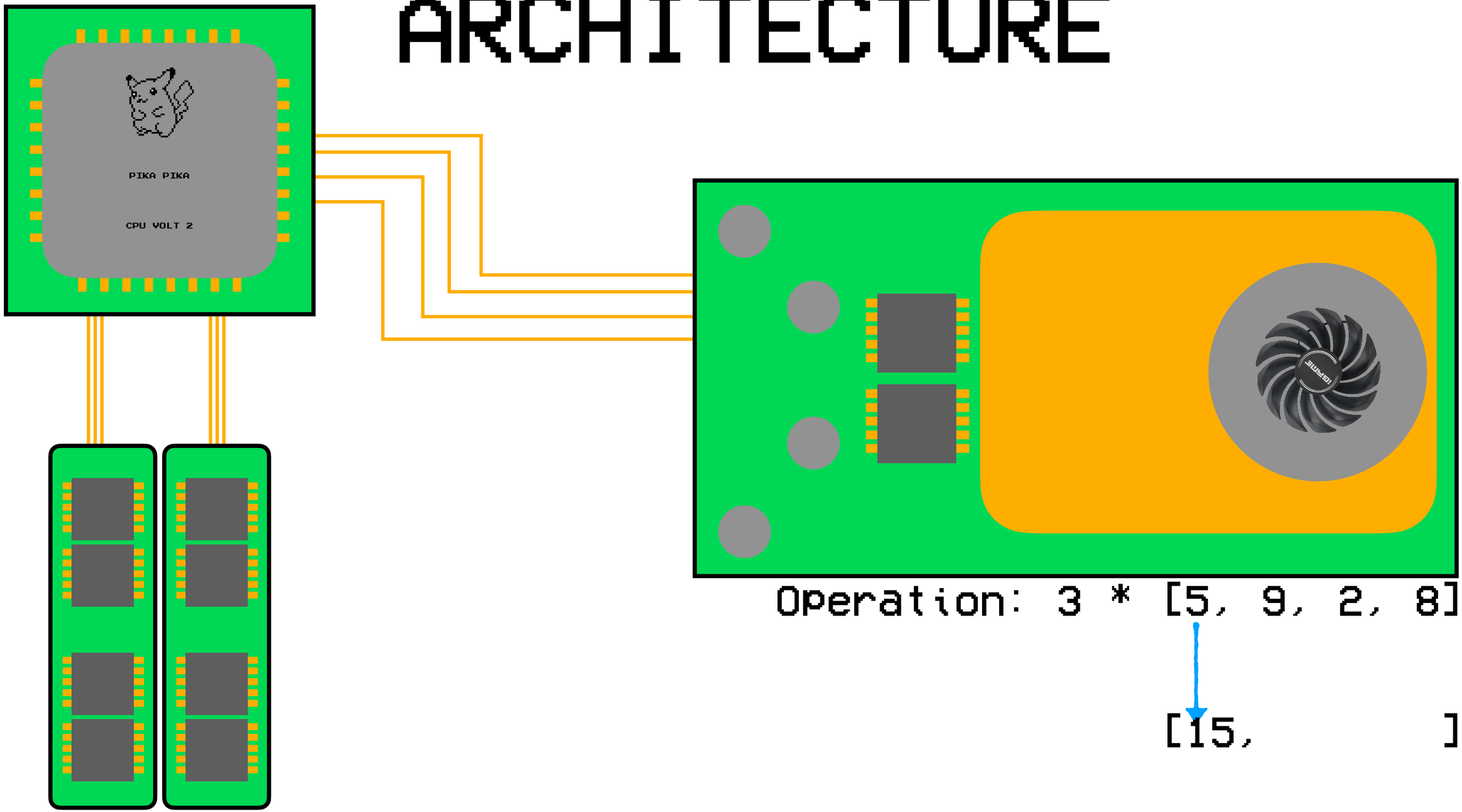
ARCHITECTURE



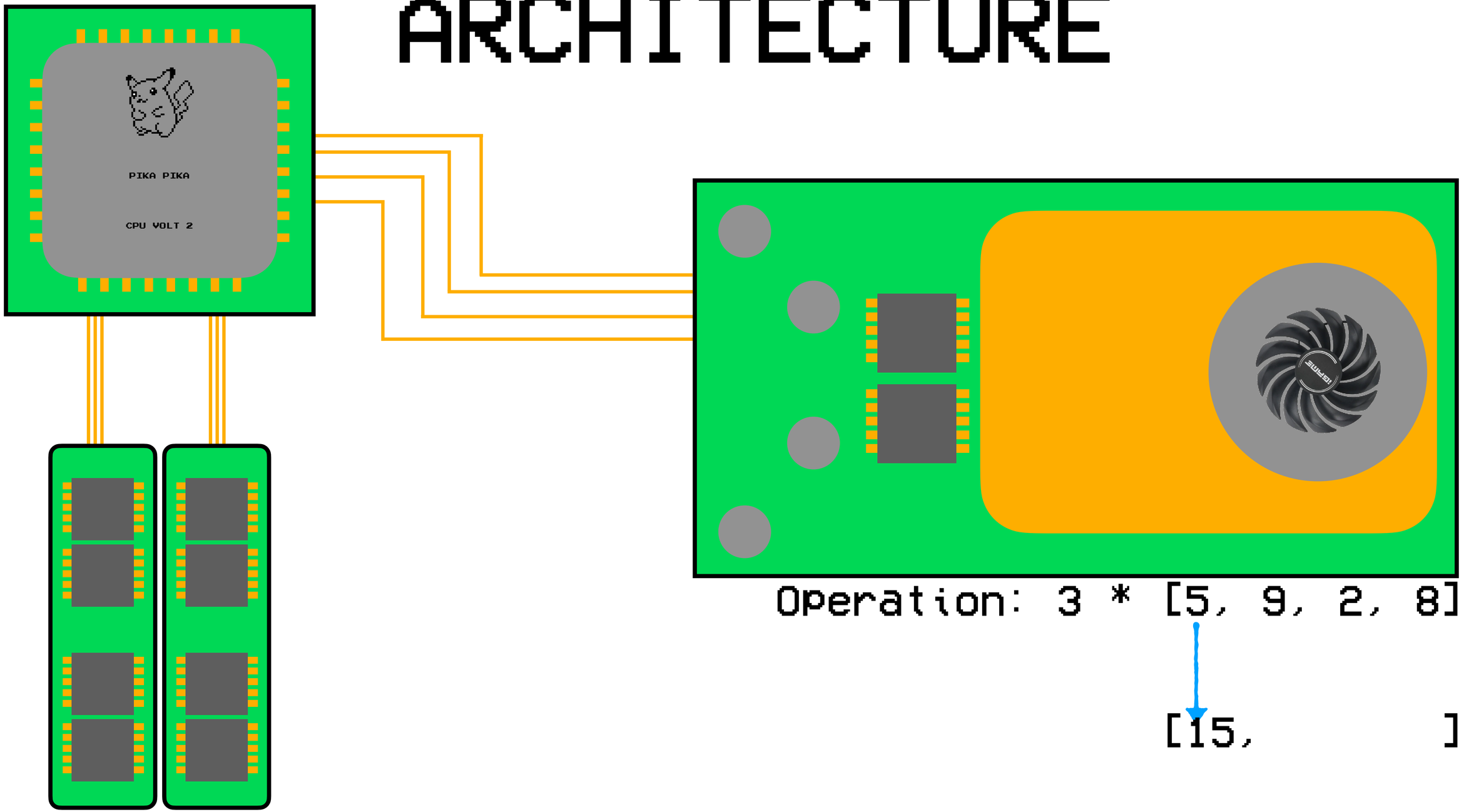
ARCHITECTURE



ARCHITECTURE

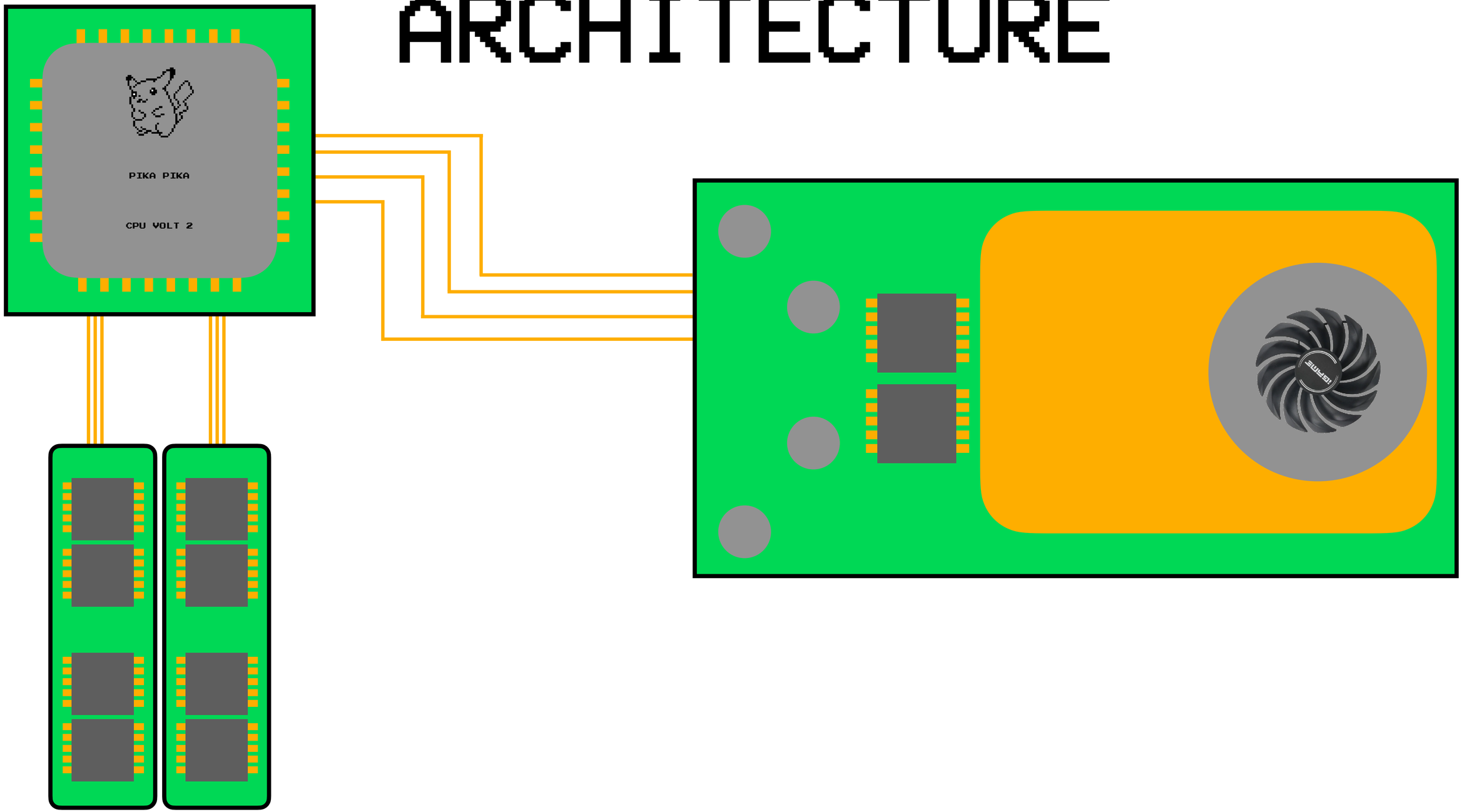


ARCHITECTURE

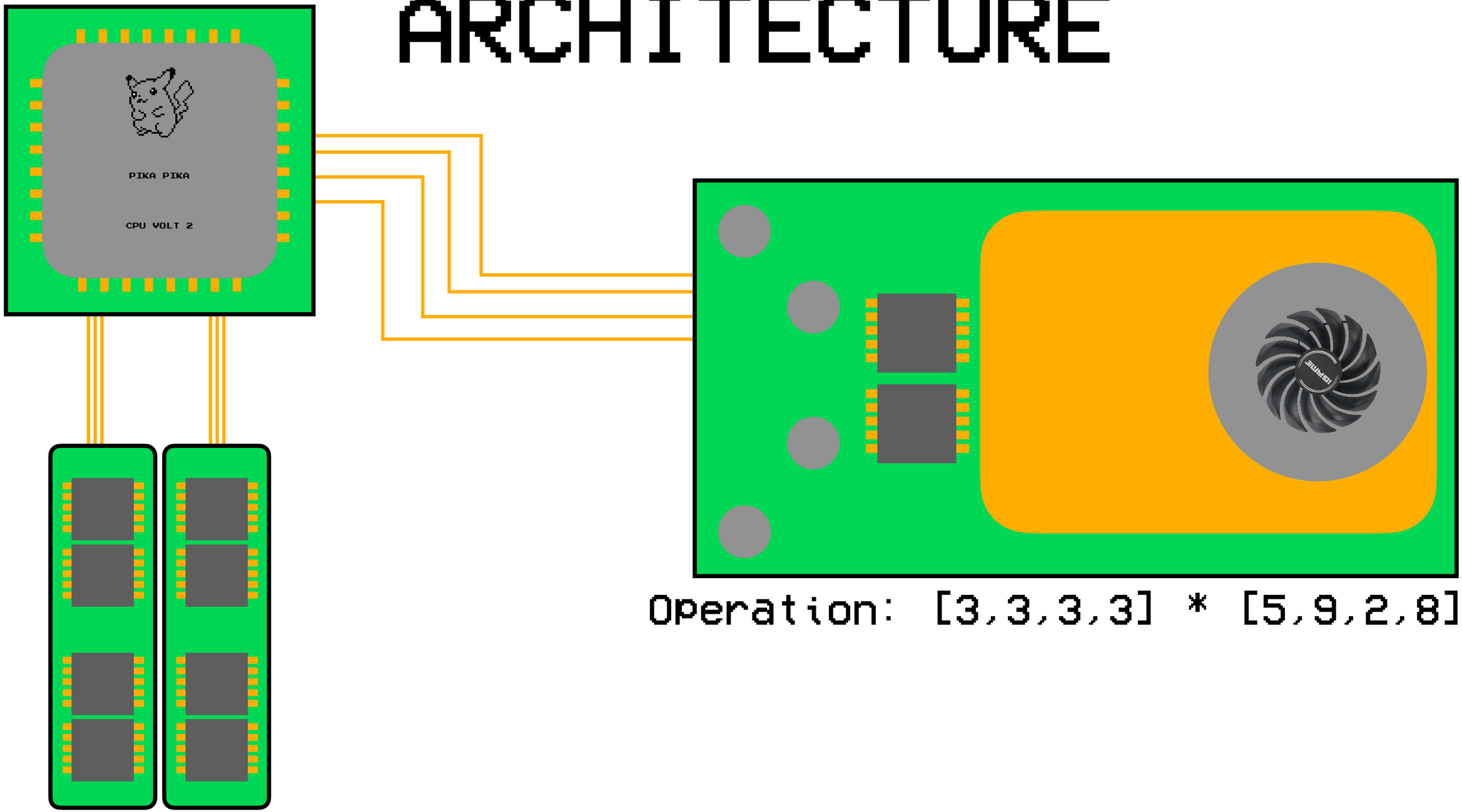


Total: 5 reads, 4ops, 4 writes

ARCHITECTURE

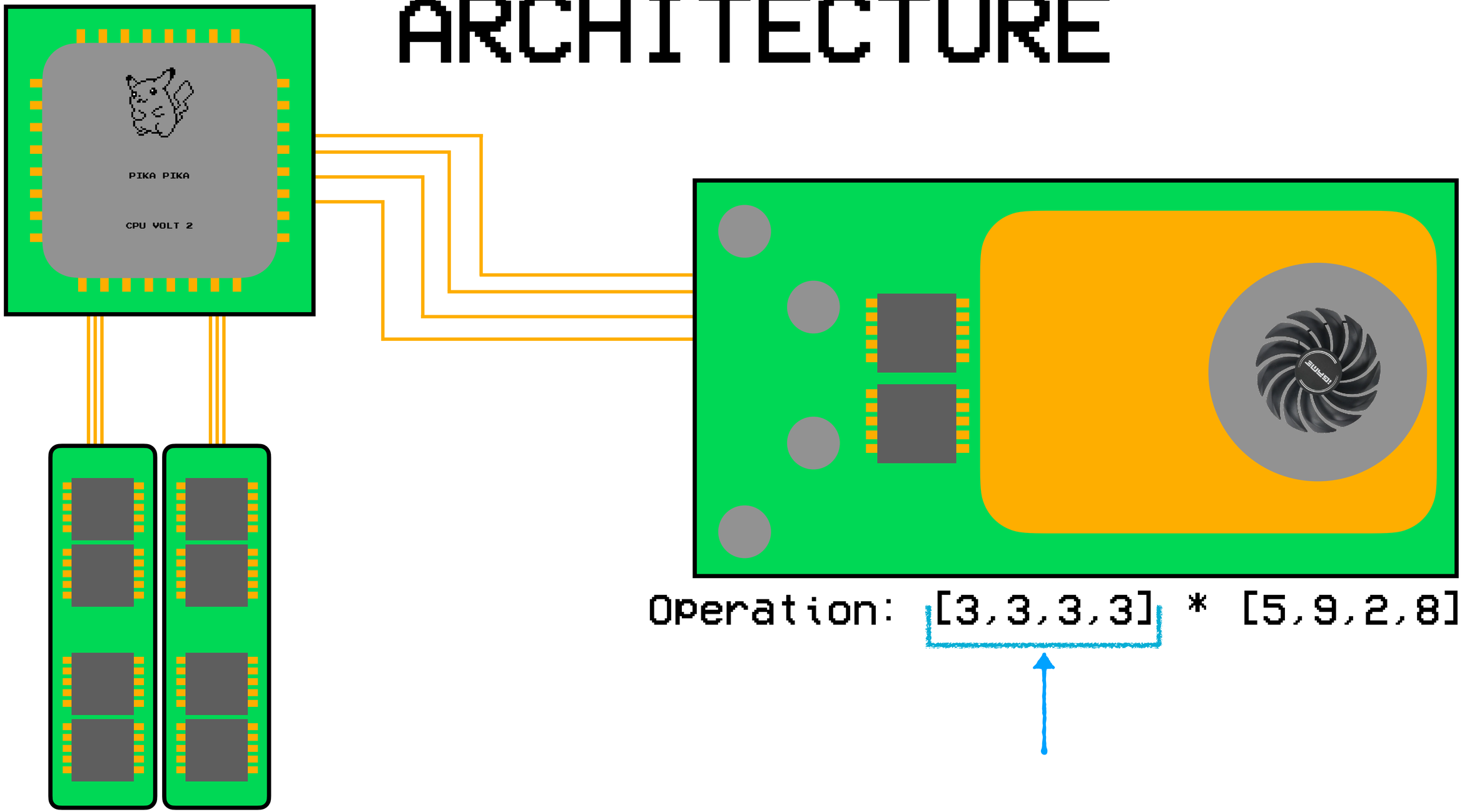


ARCHITECTURE

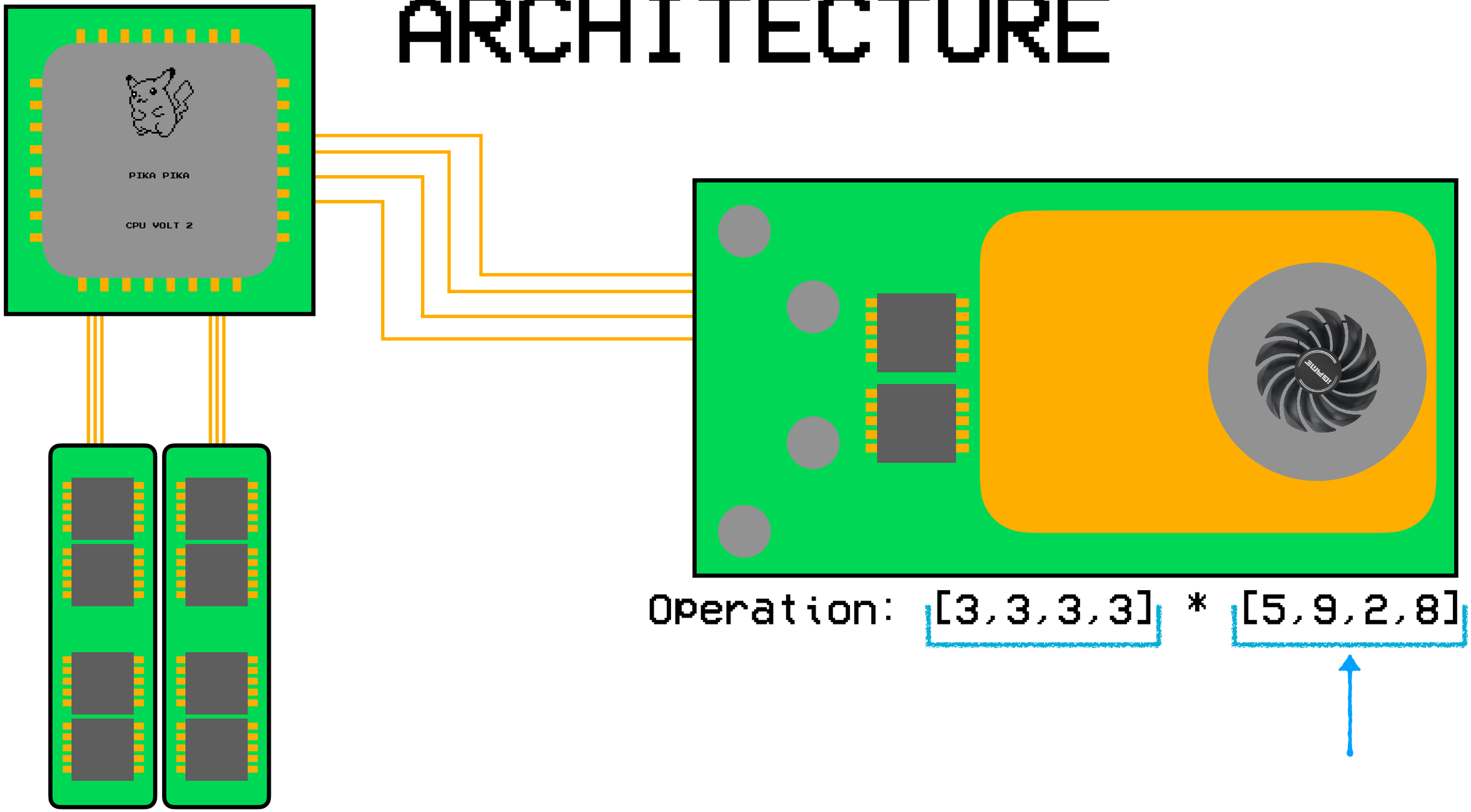


Operation: [3,3,3,3] * [5,9,2,8]

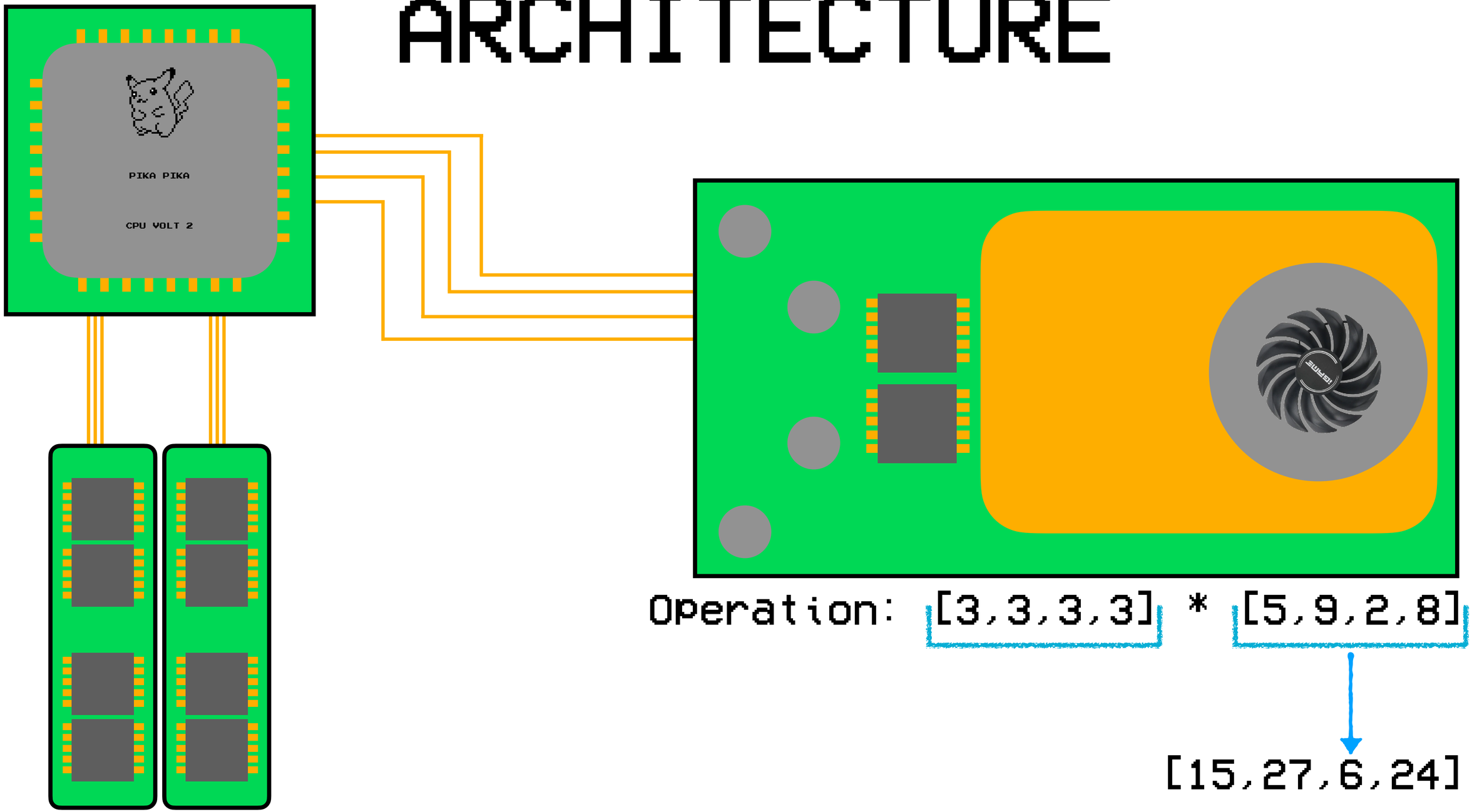
ARCHITECTURE



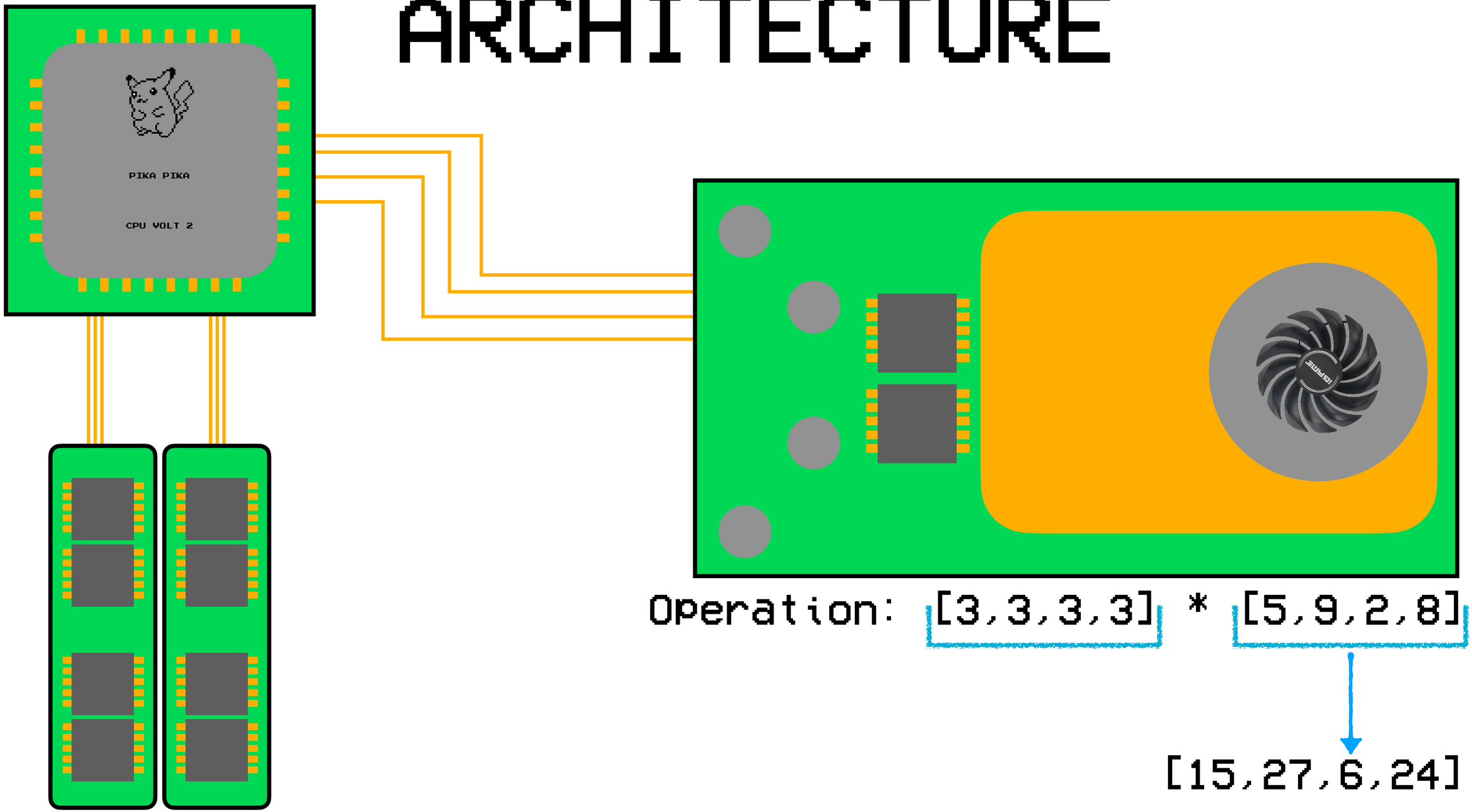
ARCHITECTURE



ARCHITECTURE



ARCHITECTURE



Operation: [3,3,3,3] * [5,9,2,8]

[15,27,6,24]

Total: 2 reads, 1op, 1 write

SIMD (CPU)

- Has to be continuous memory slice
  - Therefore not every object is suitable for SIMD
- SIMD registers (like “normal” ones) have width (like 128-bit, 256-bit and so on)
- SIMD has its own instruction set (like add, mil, div and so on)

IMPLEMENTATIONS

IMPLEMENTATIONS

SSE3

SSE4.2

NEON

SSE

SSSE3

AVX

AVX2

MDMX

IMPLEMENTATIONS

SSE4.1

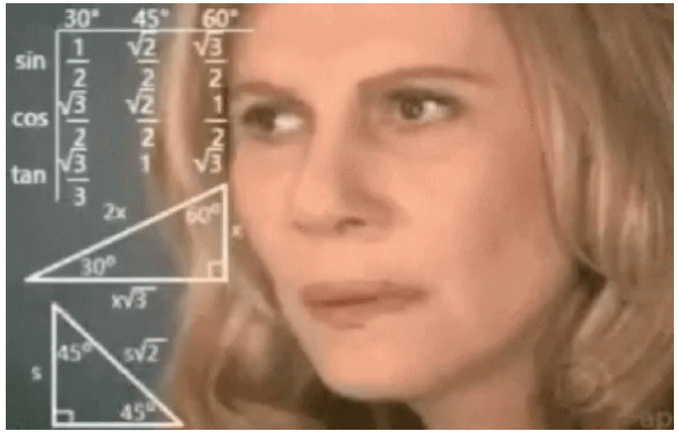
SSE4a

MMX

AdvSIMD

SSE2

AVX-512


30°
45°
60°
sin
cos
tan
2x
60°
30°
x√3
45°
s√2
s
45°

You can solve every problem with another level of indirection,

except for the problem of too many levels of indirection. - David Wheeler

"You can solve every problem with another level of indirection, except for the problem of too many layers of indirection. - David Wheeler"

Abstraction

Steven Giesel

Vector<T>

Vector<T>

Vector<T>

Vector<T>


M2
PRO


intel
CORE i5
inside™


INTEL® CORE™ i9
i9-14900K
SRN48
X334N481

Vector<T>

Vector128 Vector256 Vector512

Vector< >.Length

Vector128 Vector256 Vector512

Vector< int >.Length

Vector128

4

Vector256

8

Vector512

16

Vector< >.Length

Vector128 Vector256 Vector512

Vector<byte>.Length

Vector128
16

Vector256
32

Vector512
64

Vector< >.Length

Vector128 Vector256 Vector512

Vector<double>.Length

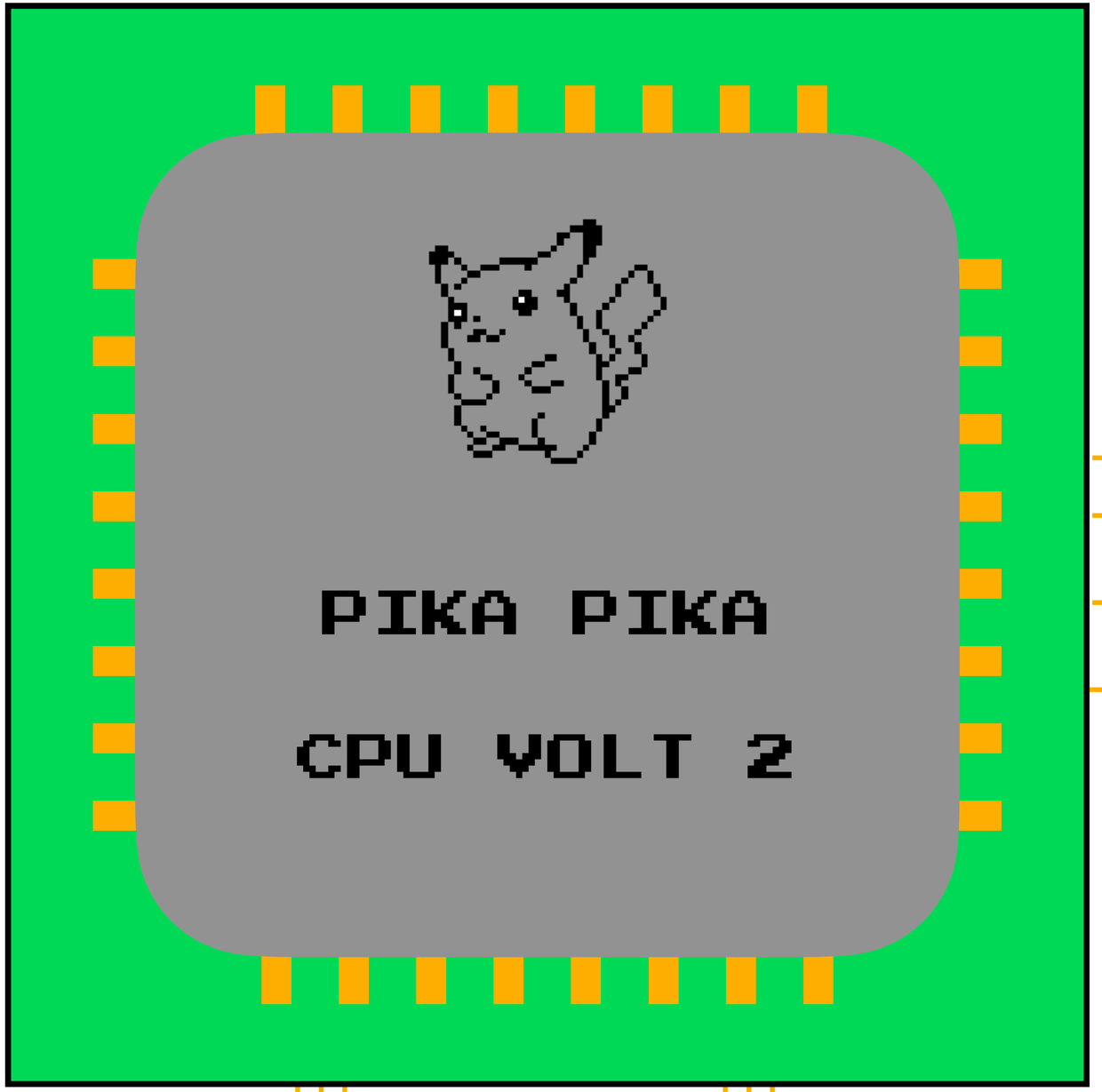
Vector128
2

Vector256
4

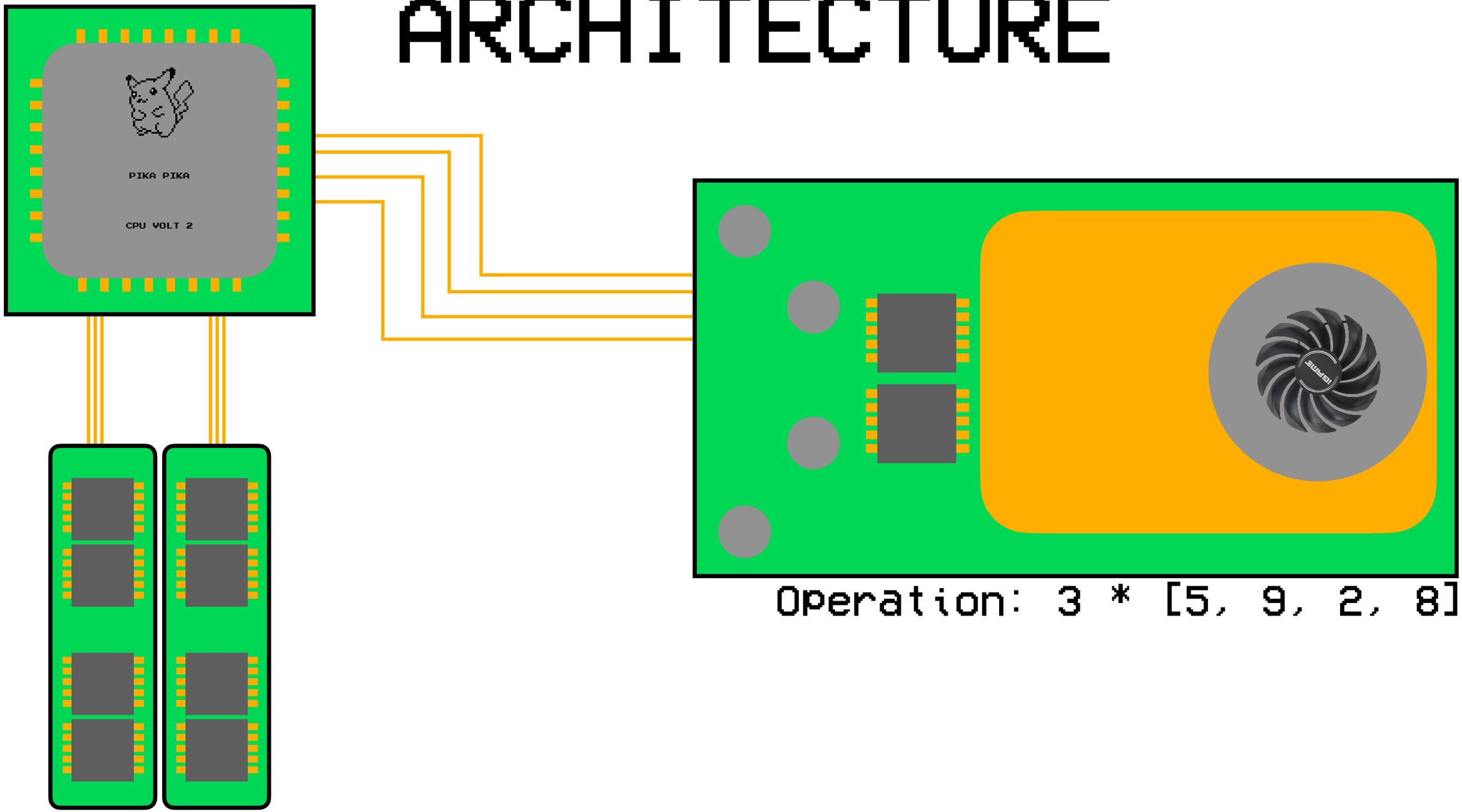
Vector512
8

CUDA / OPENCL

ARCHITECTURE

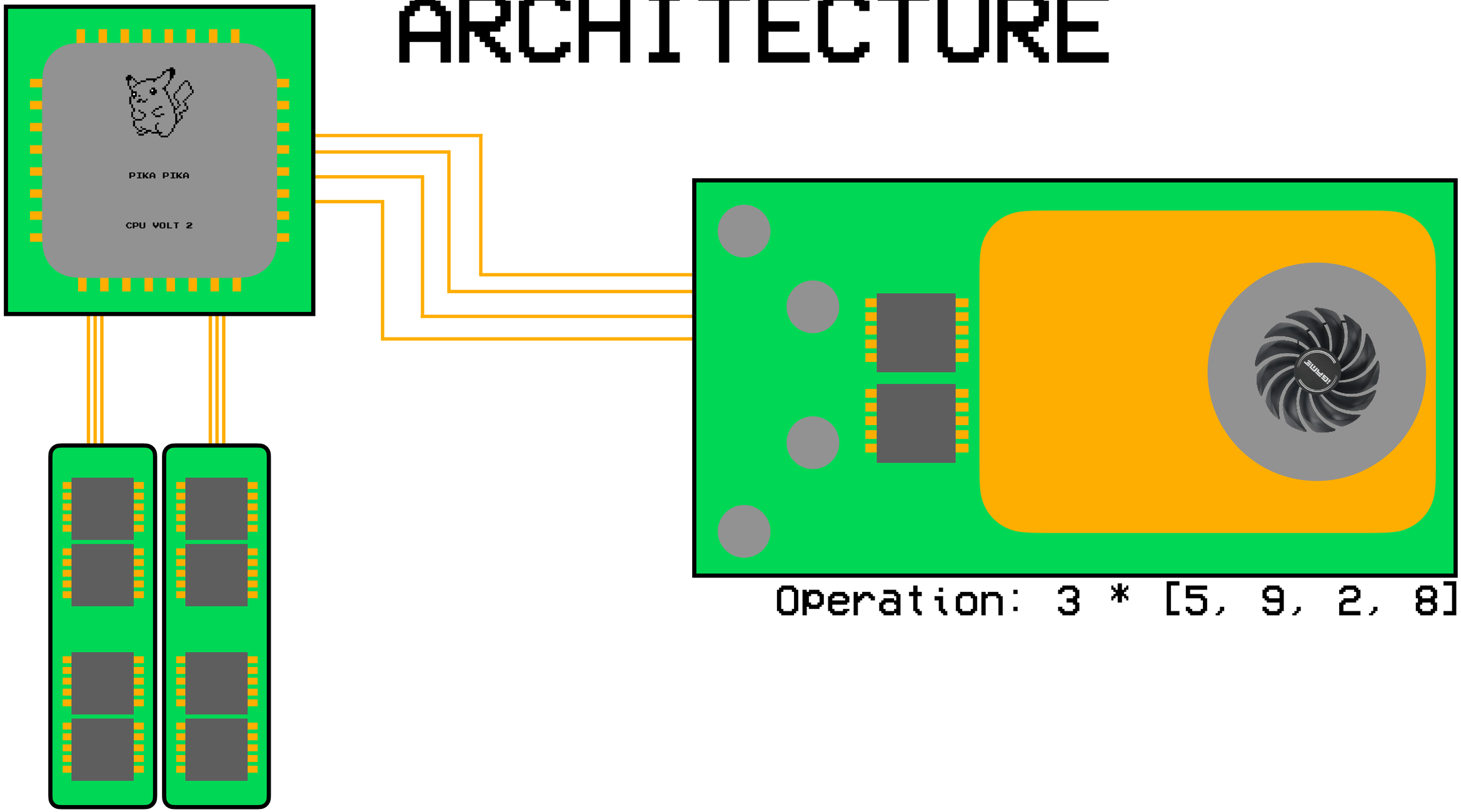


ARCHITECTURE



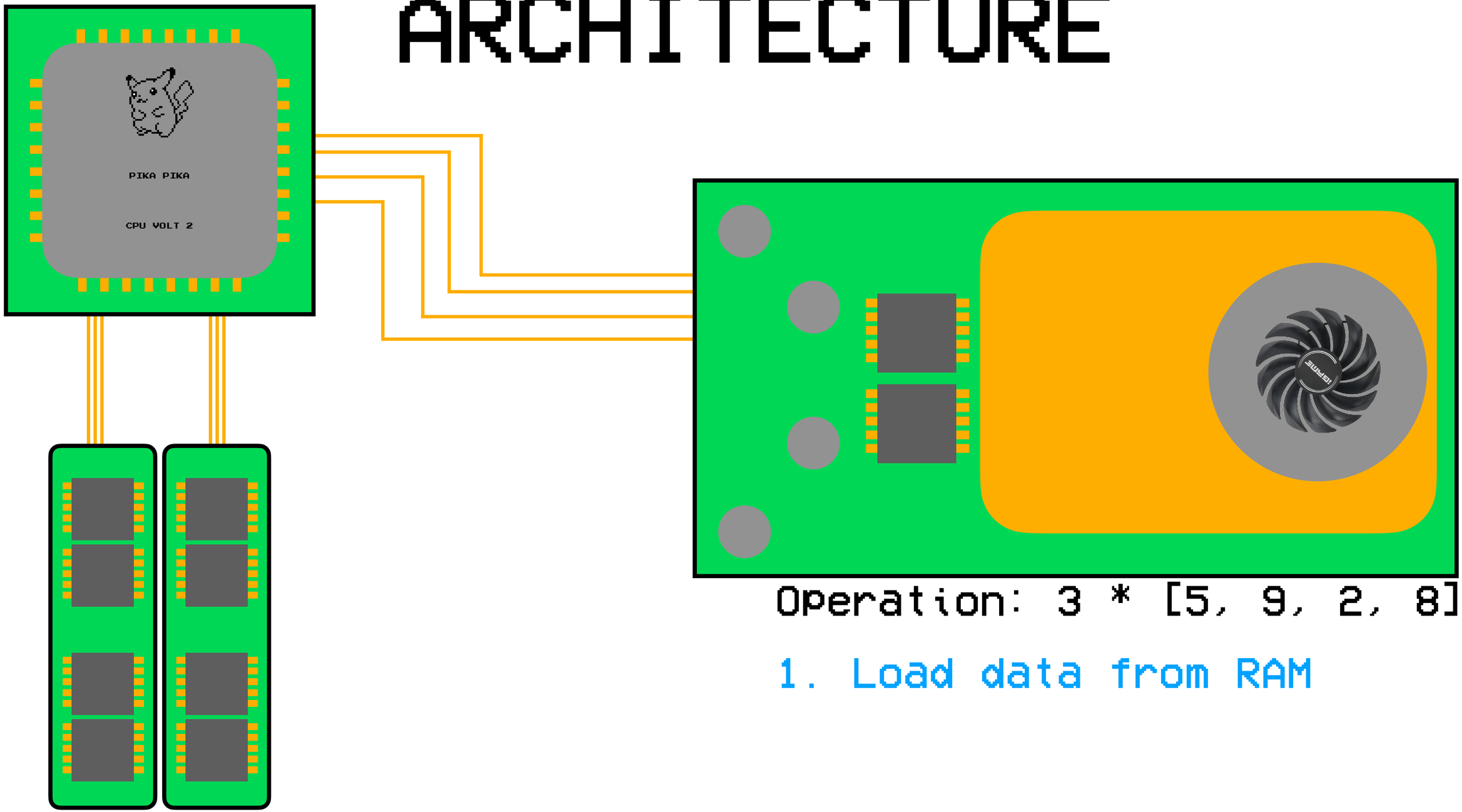
Operation: 3 * [5, 9, 2, 8]

ARCHITECTURE



Operation: 3 * [5, 9, 2, 8]

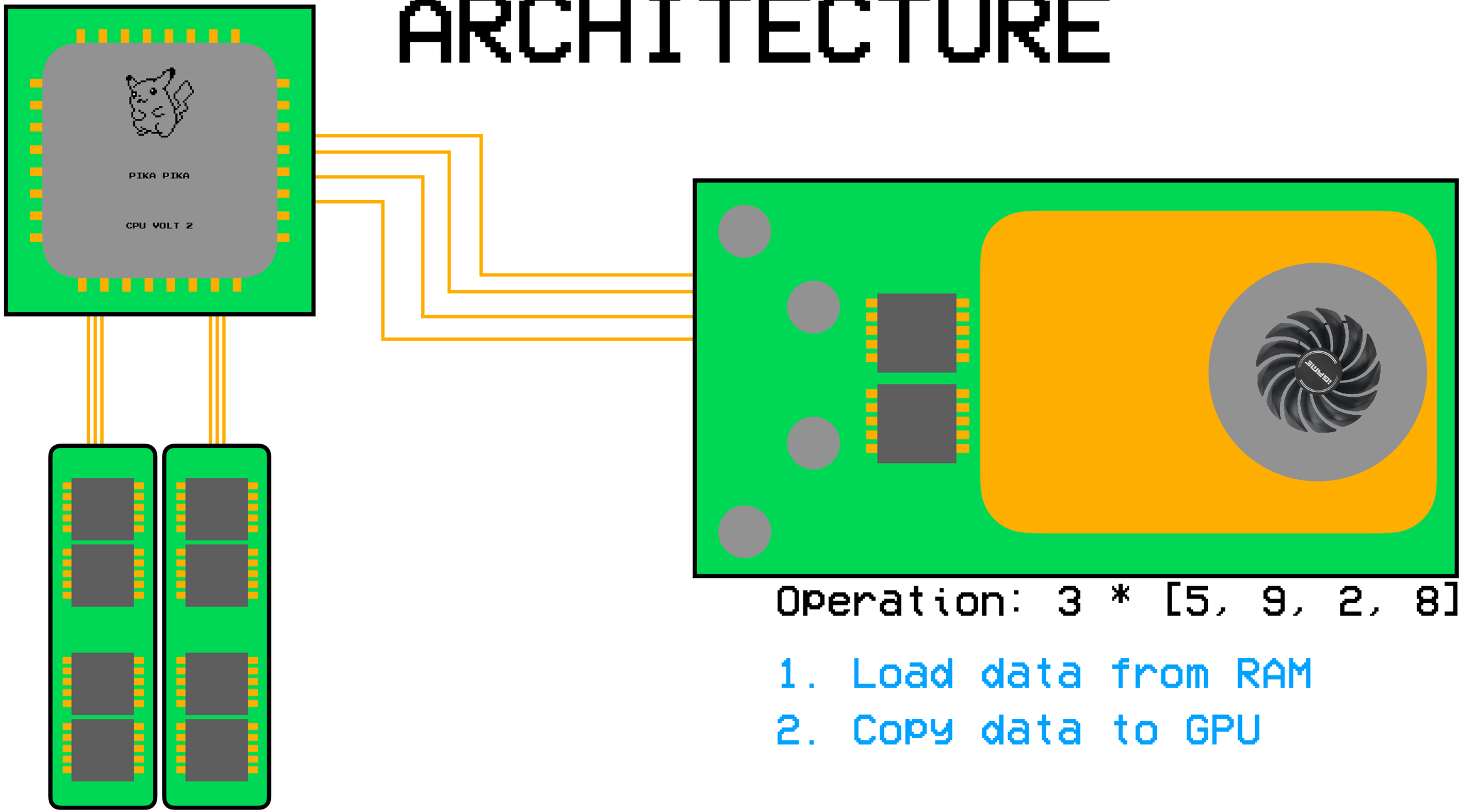
ARCHITECTURE



Operation: 3 * [5, 9, 2, 8]

1. Load data from RAM

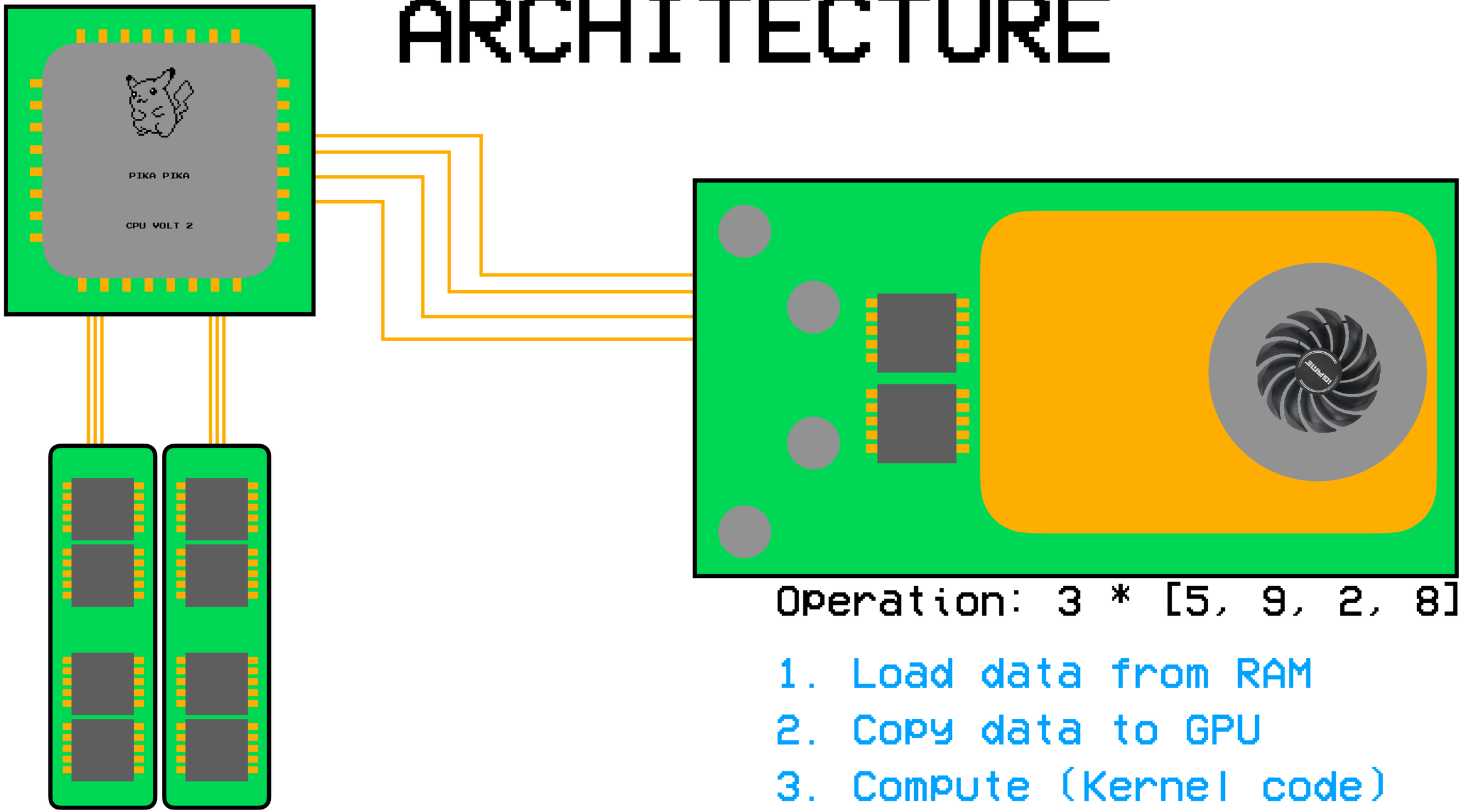
ARCHITECTURE



Operation: 3 * [5, 9, 2, 8]

1. Load data from RAM
2. Copy data to GPU

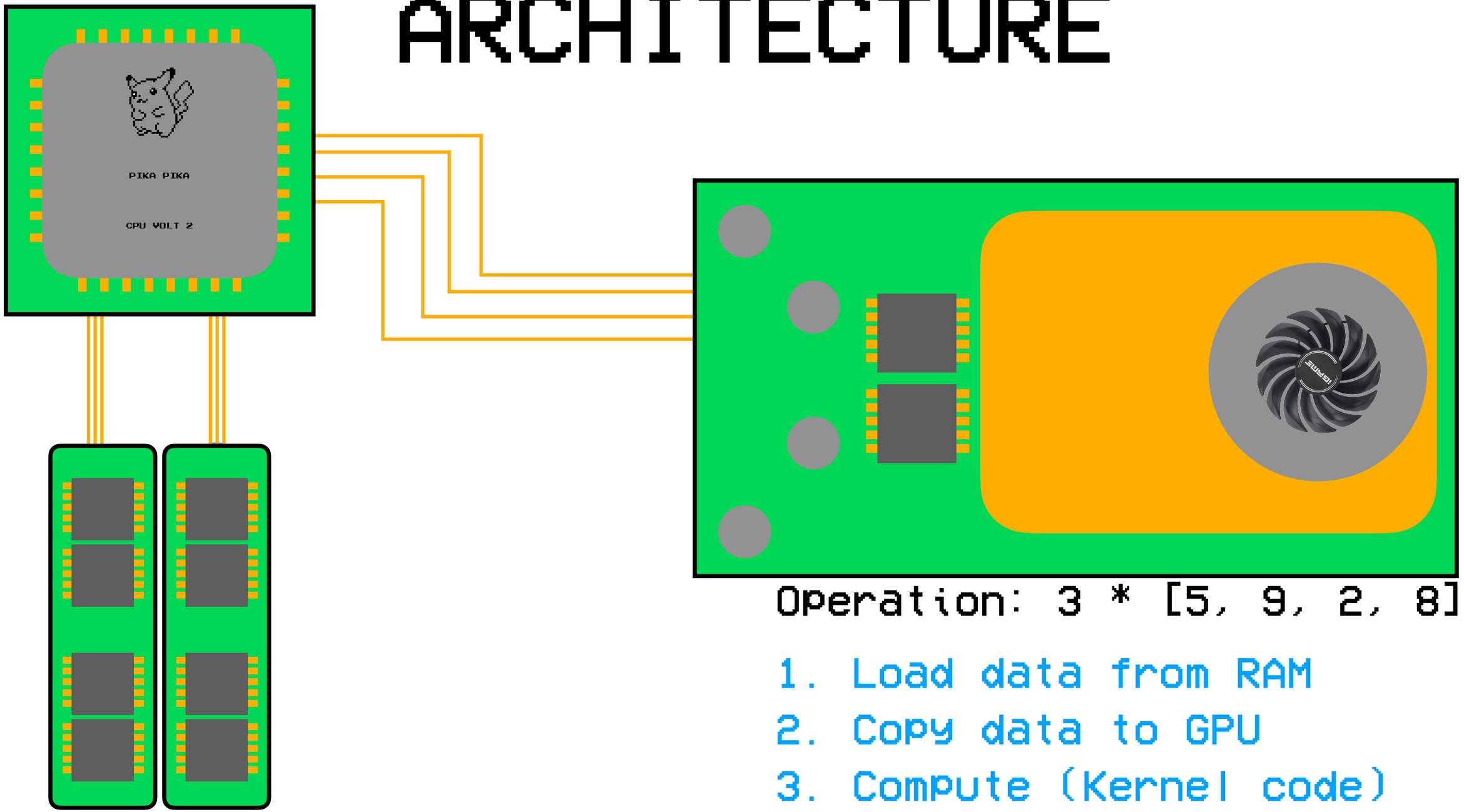
ARCHITECTURE



Operation: 3 * [5, 9, 2, 8]

1. Load data from RAM
2. Copy data to GPU
3. Compute (Kernel code)

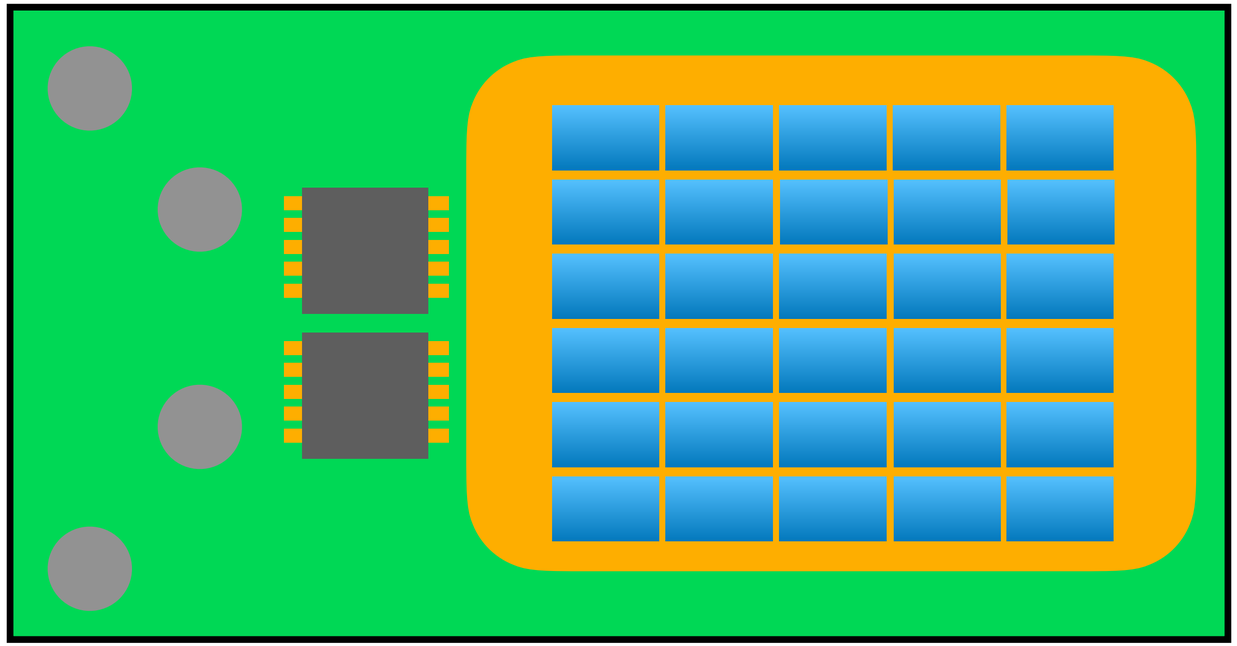
ARCHITECTURE



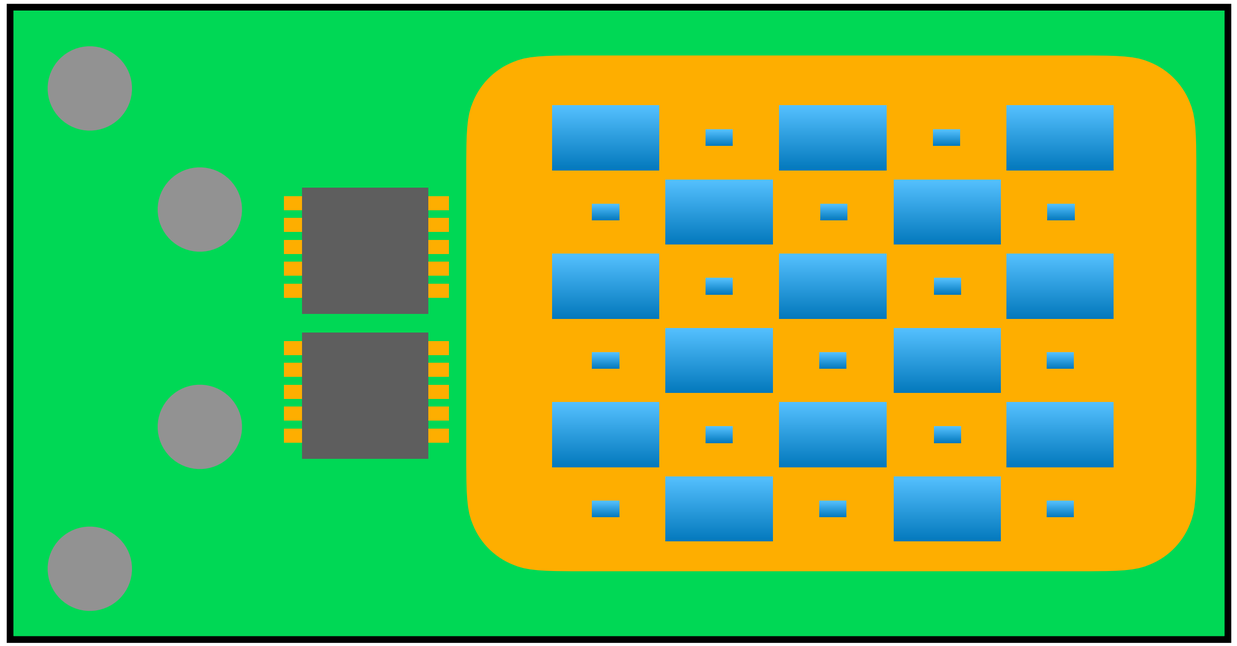
Operation: 3 * [5, 9, 2, 8]

1. Load data from RAM
2. Copy data to GPU
3. Compute (Kernel code)
4. All the way back

```
static void MyKernel(Index1D index, ArrayView<int> data)
{
    if (index % 2 == 0)
    {
        data[index] = index * index;
    }
    else
    {
        data[index] = index + 1;
    }
}
```



```
static void MyKernel(Index1D index, ArrayView<int> data)
{
    if (index % 2 == 0)
    {
        data[index] = index * index;
    }
    else
    {
        data[index] = index + 1;
    }
}
```



SIMD (GPU)

- The “problem” has to be big
  - Latency between CPU/GPU (Copy)
- Problem has to be parallelizable
  - Data and operation should be independent