PY410 / 505
Computational Physics 1

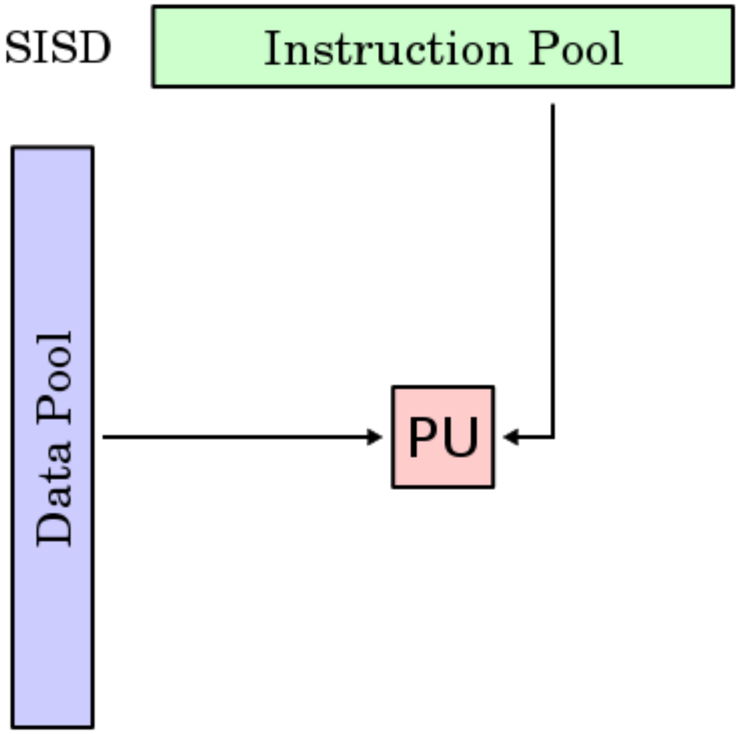
Salvatore Rappoccio

Vectorization

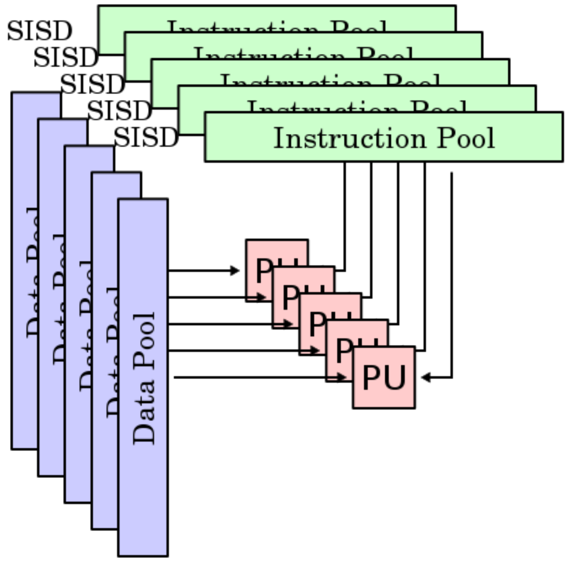
- Example in the “Vectorization” directory
- “Advanced” topic, but critical for numeric programming these days

Flynn’s Taxonomy

- Up until now, we have basically assumed we have a single instruction, single data (SISD) model:



- 1 CPU processes data from 1 location
- Parallelization is achieved by just repeating this:
  - Single program, multiple data (SPMD)



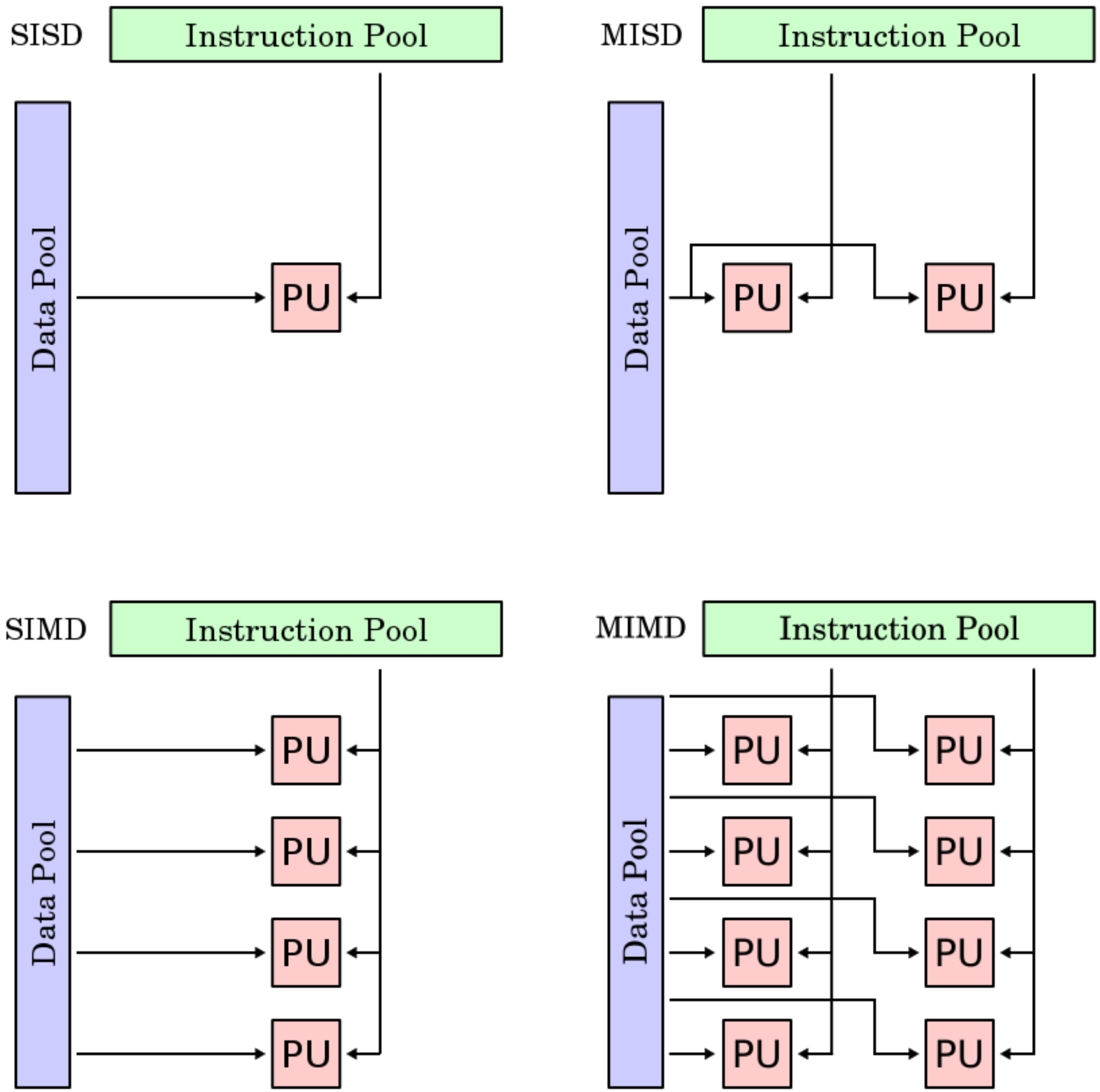
https://en.wikipedia.org/wiki/Flynn%27s_taxonomy

Flynn’s Taxonomy

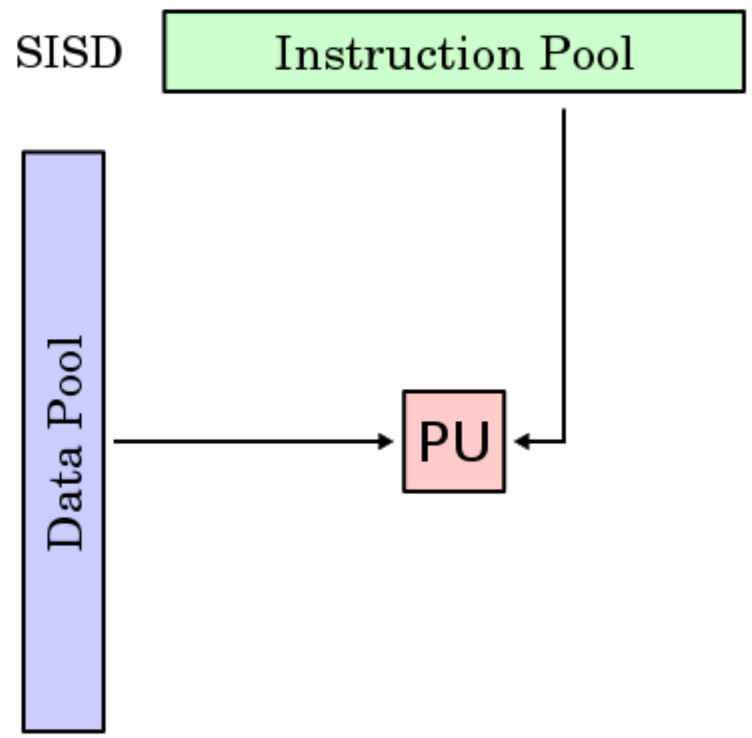
Single instruction | Multiple instruction

Single data

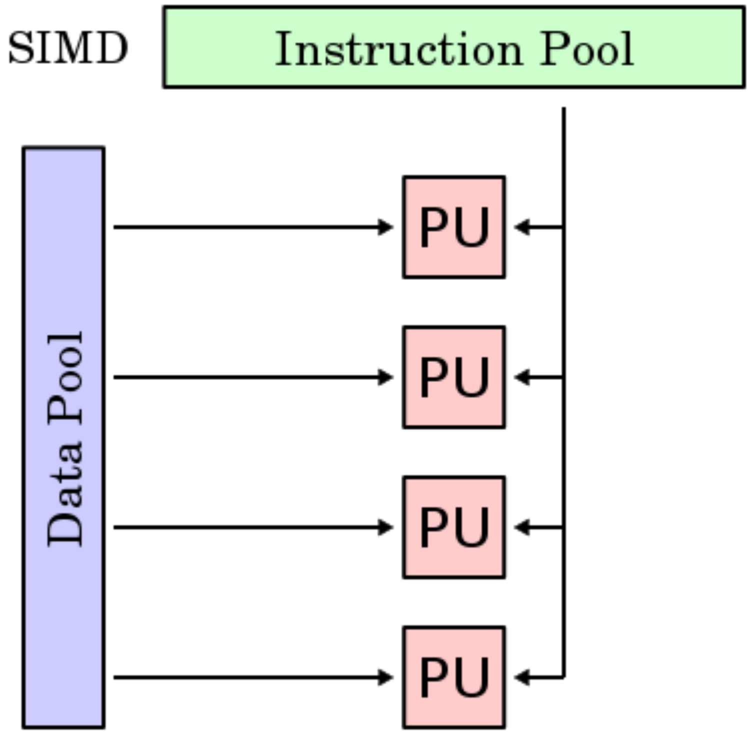
Multiple data



Flynn’s Taxonomy



=



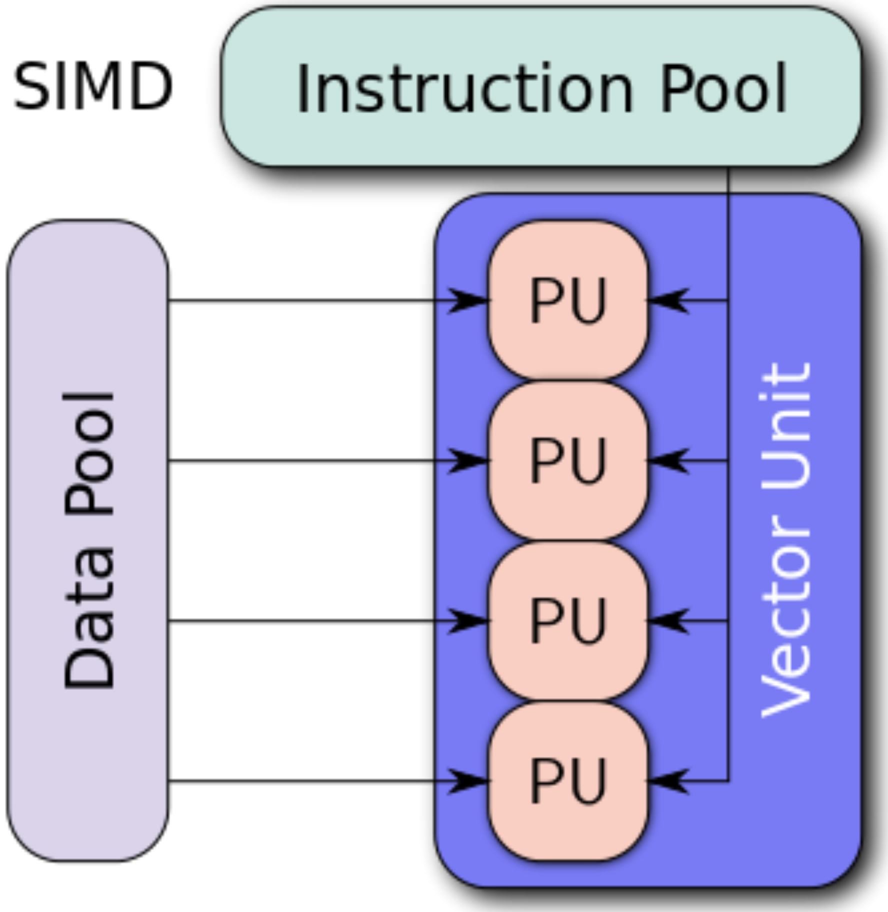
=

SIMD

- SIMD has become more popular because of vectorized processing units
  - GPUs
  - New CPUs

- Vector units do N identical operations in the same amount of time that it takes to do one of them, on different addresses in memory

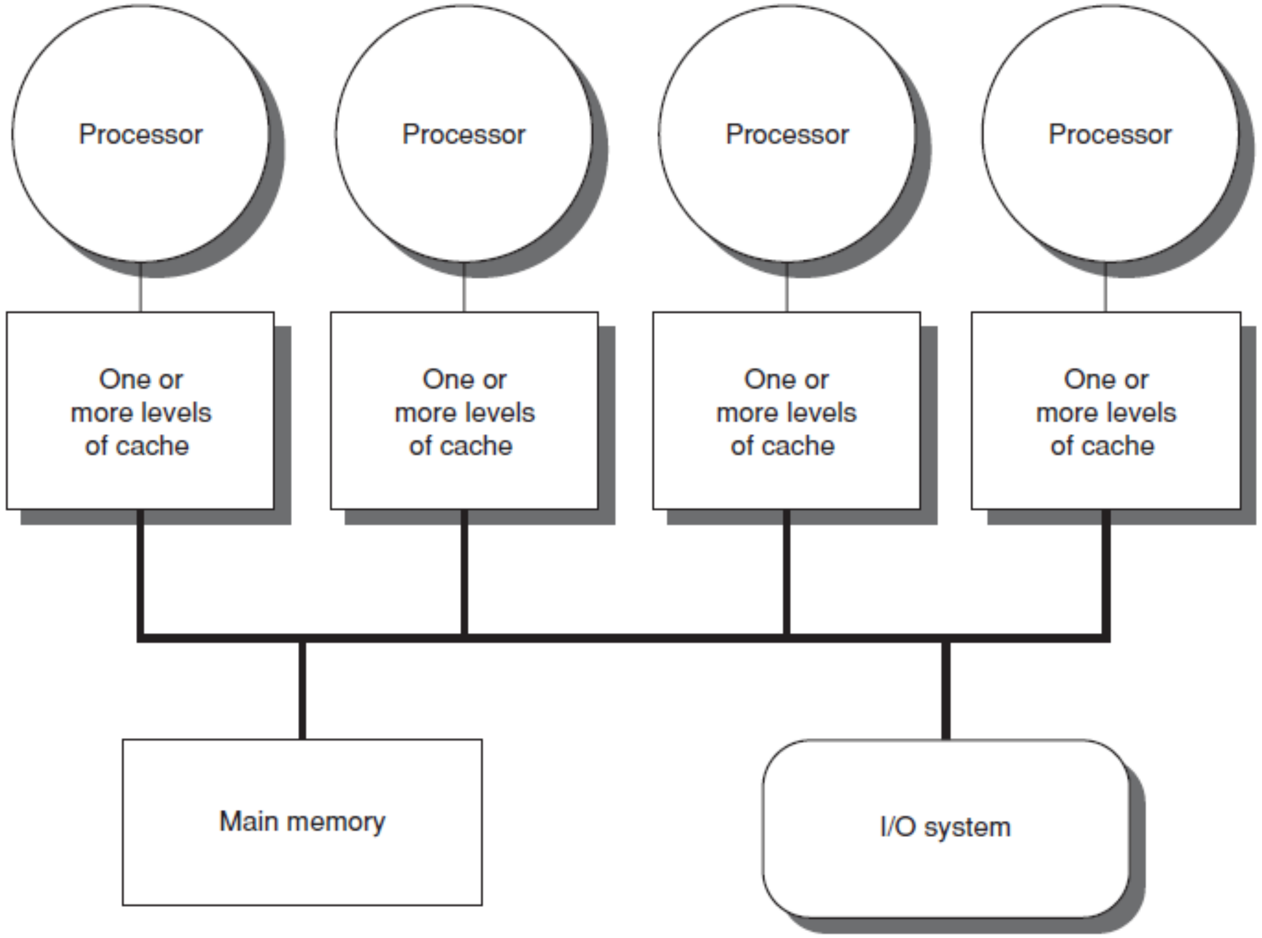
- Speed comes from data access
  - “Get me N ints” versus “Get me an int”, N times



Architecture of processors

- Processing units now contain caches of memory with fast access.
- Can have different levels of cache (small but fast versus large but slow)
- Then memory
- Then disk

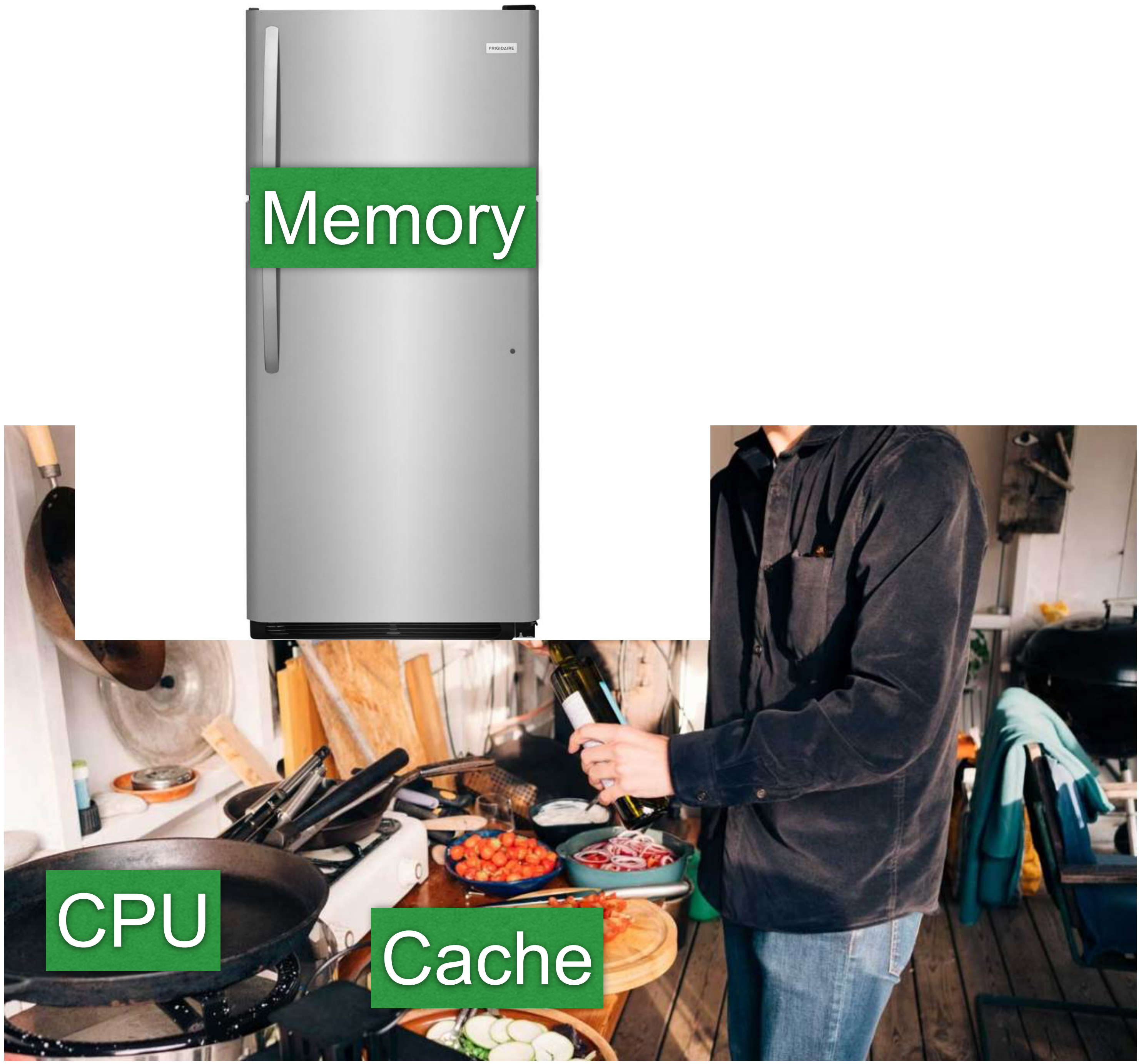
- Accessing cache is much faster than accessing memory
- Accessing memory is much faster than accessing disk



https://en.wikipedia.org/wiki/Cache_hierarchy

Architecture of processors

- Cooking analogy:



Vector processing

- Same operation done many times over a vector of data (SIMD!)
- Traditionally: done with loops
- Now: done with vector unit, if possible

```
for (i = 0; i < 1024; i++)
    C[i] = A[i]*B[i];
```

Get next bread.
Slice.
Get next bread.
Slice.
Get next bread.
Slice.

```
    C = A*B;
```

Slice all the bread.

https://en.wikipedia.org/wiki/Vector_processor

Unrolling loops

- There are also advantages to unrolling loops:

```
for (i = 0; i < 1024; i++)
    C[i] = A[i]*B[i];
```

Get next bread.
Slice.
Get next bread.
Slice.
Get next bread.
Slice.

```
for (i = 0; i < 1024; i+=4){
    C[i+0] = A[i+0]*B[i+0];
    C[i+1] = A[i+1]*B[i+1];
    C[i+2] = A[i+2]*B[i+2];
    C[i+3] = A[i+3]*B[i+3];
}
```

Get four slices of bread.
Slice.

https://en.wikipedia.org/wiki/Loop_unrolling

Vectorization in Practice

- Scalar:

```
for (i = 0; i < 1024; i++)
   C[i] = A[i]*B[i];
```

- Vectorized:

```
for (i = 0; i < 1024; i+=4)
   C[i:i+3] = A[i:i+3]*B[i:i+3];
```

Implicitly or explicitly unrolling loops can
allow the compiler to appropriately vectorize the operation

https://en.wikipedia.org/wiki/Automatic_vectorization

Test case

- We will use the simple addition of two large vectors (100000 elements) as a demonstration in various scenarios:
  - C++ without optimization
  - C++ with non-vectorization optimizations
  - C++ adding vectorization optimizations
  - Python itself
  - Numpy within python

Comparisons

	C++, -O0	C++, -O1	C++, -O2	python	python+ numpy
Vector, unknown size	825	585	150	31862	460
Vector, known size	736	564	93		
Static array	576	564	525		
Static array, manual unroll	132	97	93		