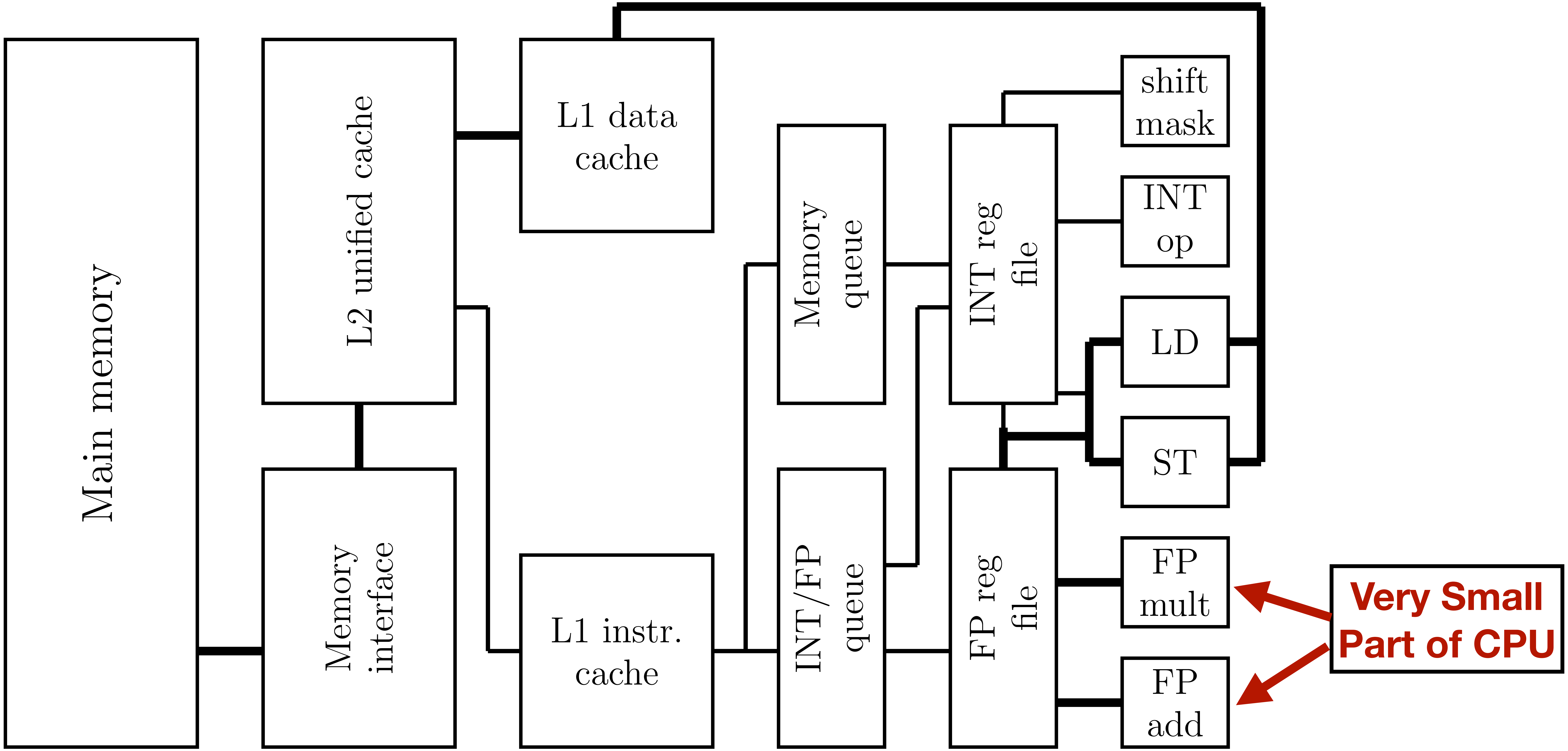
Introduction to Parallel Processing

Lecture 4 : Vectorization

09/02/2022

Professor Amanda Bienz

Cache-Based Microprocessor



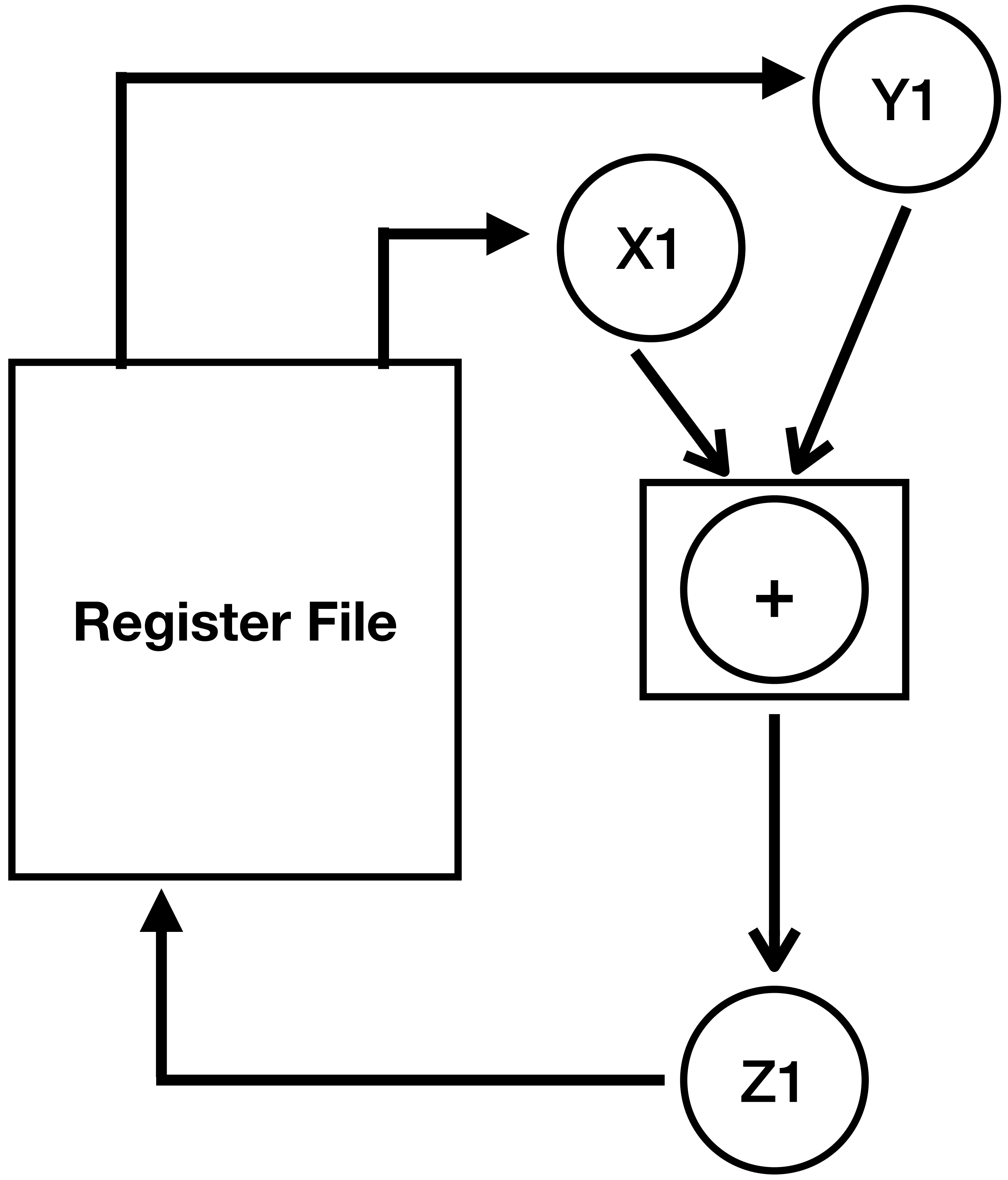
Want to improve performance, have more operations at once

Serial Architecture

Say we want to add two arrays together : Z = X + Y

For i = 0 to n
z[i] = x[i] + y[i]

scalar operations : n

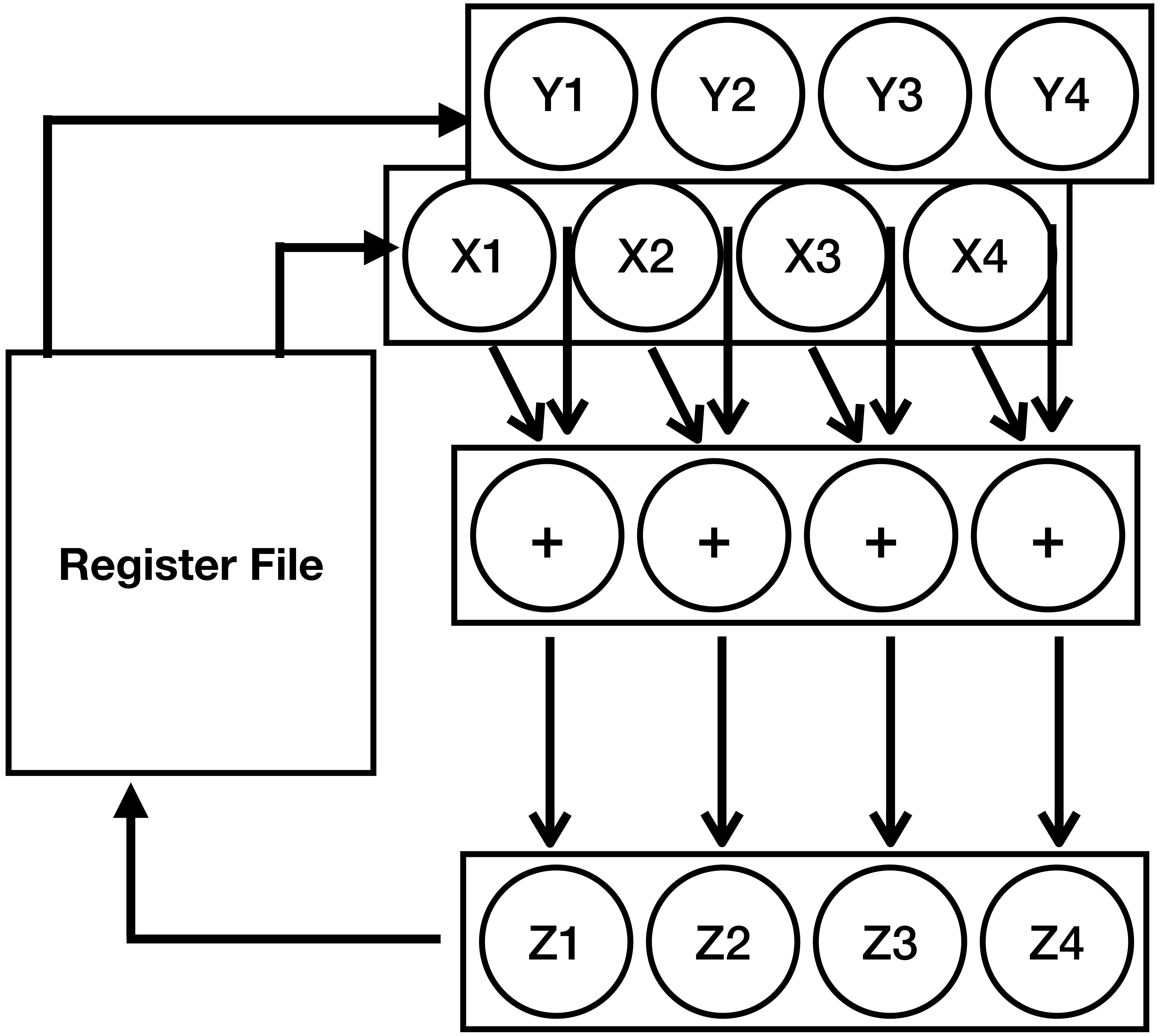


Vector Architecture

Say we want to add two arrays together : Z = X + Y

For i = 0 to n
z[i] = x[i] + y[i]

vector operations :
n/2 to n/4



Operate on 16 Bytes :

4 Floats / Ints
2 Doubles

Vectorized Loops

N	Bytes	Scalar	Vector	Speedup
1,000	4	5.5E-07	2.2E-07	2.6x
1,000	8	5.7E-07	2.8E-07	2x
1,000,000	4	8E-04	5.6E-04	1.4x
1,000,000	8	1.4E-03	1.3E-03	1.1x

Let's Step Through Matrix Multiplication

$$\underset{n \times n}{\begin{bmatrix} \quad A \quad \end{bmatrix}} \times \underset{n \times n}{\begin{bmatrix} \quad B \quad \end{bmatrix}} \rightarrow \underset{n \times n}{\begin{bmatrix} \quad C \quad \end{bmatrix}}$$

Data Dependencies

For i = 0 to n
z[i] = x[i] + y[i]

- Let’s look back at this addition operation
- Vector operations : load a block of four floats (or two doubles) into the vector register (e.g load x[0], x[1], x[2], x[3])
- Vector operation might look like the following
  - z[0:3] = x[0:3] + y[0:3]
  - z[4:7] = x[4:7] + y[4:7]

http://wgropp.cs.illinois.edu/courses/cs598-s16/lectures/lecture13.pdf

Data Dependencies

For i = 0 to n
z[i] = x[i] + y[i]

- What if &z[0] points to the same memory address as &x[1]
- Then:
  - z[0] = x[0] + y[0]
  - z[1] = x[1] + y[1] = **z[0]** + y[1]
  - z[2] = **z[1]** + y[2]

http://wgropp.cs.illinois.edu/courses/cs598-s16/lectures/lecture13.pdf

Data Dependencies

- A few examples:
  - X = A + B
    C = X + A
  - A = X + B
    X = C + D
  - X = A + B
    X = C + D

http://wgropp.cs.illinois.edu/courses/cs598-s16/lectures/lecture13.pdf

Data Dependencies

- If no data dependencies, instructions can be executed in any order, in parallel, or in a vector operation
- If there are data dependencies, the compiler (and user) cannot perform these optimizations
  - But, sometimes data can be rearranged to avoid dependencies
  - i.e. find two operations that are not dependent and optimize these

http://wgropp.cs.illinois.edu/courses/cs598-s16/lectures/lecture13.pdf