

CS8625

Chapter 1 from textbook
**CS8625 High Performance and
Parallel Computing**
Dr. Ken Hoganson

Class

Will

Start

Momentarily...

Single Instruction, Multiple Data

- Presumes a computer architecture containing many processing elements that all receive the same stream of instructions, and all can operate on different data.
- Each processor contains small temporary storage (registers)
- Data may need to be pre-loaded to the processing elements, prior to the start of processing.
- All processors are connected by an instruction bus

Small array example

- Say you have two 1-D arrays (vectors) that need to be manipulated to create a new resulting 1-D array.
- Arrays A, B, C, each contain 16 elements
- $C = A \text{ op } B$, in general

Details of the operation $C[I] = A[I] \text{ op } B[I]$

- How many operations are required? _____
- How many data reads? _____
- How many result stores? _____

$C[I] = A[I] \text{ op } B[I]$

- Any data conflicts?
 - Read and write same value?
 - Concurrent reads?
 - Concurrent writes?

$$C[I] = A[I] \text{ op } B[I]$$

How do we pre-load the data, in order to minimize the number of accesses over the interconnect that connects the processing elements (PEs) (and takes time/latency).

- Assuming 16 PEs, then preload each PE_i with $A[I]$ and $B[I]$.
- Where does the C array reside?

$$C[I] = A[I] \text{ op } B[I]$$

- What kind of speedup do we get, compared to doing the work with a single processor? _____
- Speedup = Serial-Time/Parallel-Time

- What are we leaving off in our calculation?

Pre-Load of Data
Extraction of Results

$$C[I] = A[I] \text{ op } B[I]$$

Speedup = serial time/parallel time

Time to distributed data +
Calculation time +
Time to return result

Time to distributed data = a serial operation from a single source = $16 * \text{data distribution time}$

Note: latency to move data always slower than time for one computation

Time to return result = a serial operation back to a single source = $16 * \text{latency}$

$$C[I] = A[I] \text{ op } B[I]$$

- Lets say data movement time is equal to time for one computation (it is usually worse).
- Parallel time = $16 + 1[16\text{parallelops}] + 16 = 33$
- Serial time = 16 ops
- $16/33 = \frac{1}{2}$ **Slower on parallel machine?!**
- **When might this process make sense?** – if more operations where to be performed on each data items, for the same setup and collate-results times.
 - Say had 1024 operations to perform, on 16 processors, with same data movement. **ParT=16 + 1024/16 + 16**
- Speedup = $1024/96$ or about 10
- Note, many author's "fudge" and discount problem setup and results collation.

- Example to talk through

```
/* Matrix multiply,  $C := A \cdot B$ . Compute elements of  $C$  by
```

$$c_{ij} = \sum_{k=0}^{N-1} a_{ik} b_{kj} \quad */$$

```
for i := 0 step 1 until N-1
```

```
  begin /* Compute one row of C. */
```

```
    /* Initialize the sums for each element of a row of C. */
```

```
    C[i, j] := 0, (0 ≤ j ≤ N-1);
```

```
    /* Loop over the terms of the inner product. */
```

```
    for k := 0 step 1 until N-1
```

```
      /* Add the kth inner product term across columns in parallel. */
```

```
      C[i, j] := C[i, j] + A[i, k]*B[k, j], (0 ≤ j ≤ N-1);
```

```
    /* End of product term loop. */
```

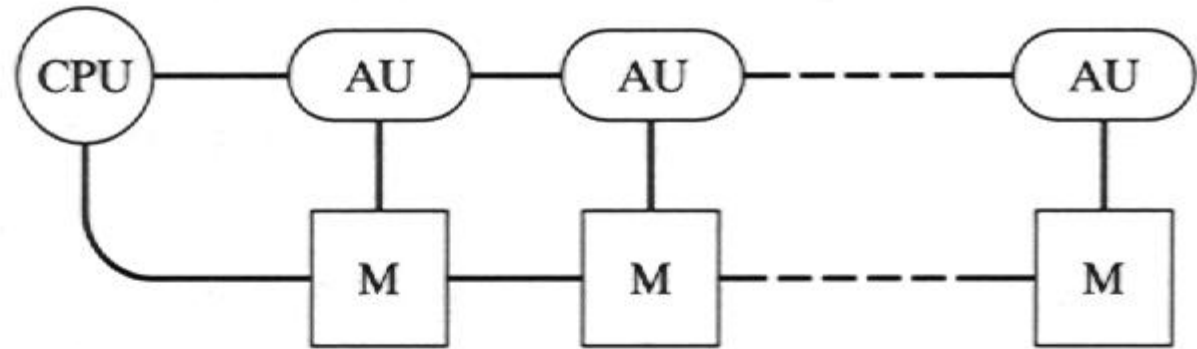
```
  end /* of all rows. */
```

```
/* Loop over the terms of the inner product. */  
for k := 0 step 1 until N-1  
  /* Add the kth inner product term across columns in parallel. */  
  C[i, j] := C[i, j] + A[i, k]*B[k, j], (0 ≤ j ≤ N-1);  
/* End of product term loop. */
```

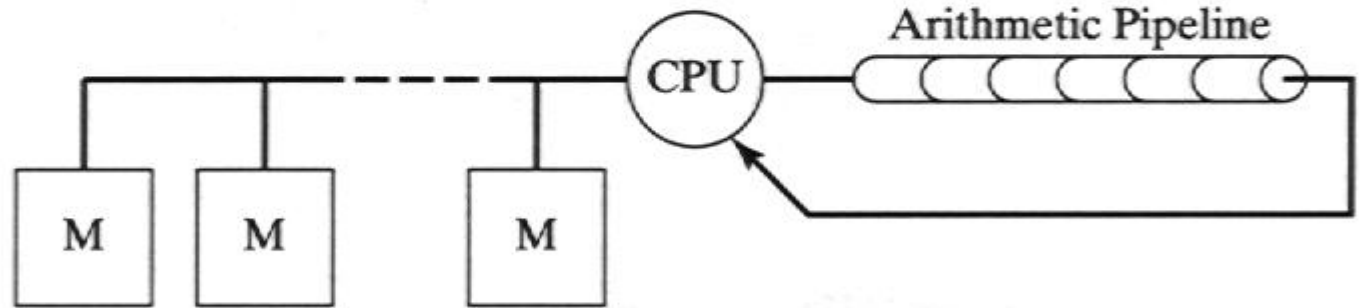
Focus: For each element there are n steps, each of which is a multiply and an add. Here n is the size of column=row (assumes square).

- 16 element example, $n=4$.
- To compute each element requires $4[\text{mult}+\text{add}]$. Multiplies often take longer than adds, lets say just twice as long.
- 16 elements, so $16[4(2+1)] = 192$ operation times

- But what about data access, problem setup, results collation?
- Note that each element needs access to a different subset of the matrix (its entire row and entire column). There are 16 subsets.
- Do we duplicate and distribute the data to the processing elements? If so, each element goes to _____ different PEs? Can we broadcast a data item to multiple PEs at the same time?
- Do we need special hardware architecture to handle the movement of data?



(a) True SIMD or vector computer



(b) Pipelined SIMD computer

AU — Arithmetic unit CPU — Central processing unit M — Memory

Figure 1-8 True and pipelined SIMD architectures.

/* Matrix multiply, $C := A \cdot B$. Compute elements of C by

$$c_{ij} = \sum_0^{N-1} a_{ik} b_{kj} \quad */$$

```
private i, j, k;
shared A[N,N], B[N,N], C[N,N], N;
/* Start N-1 new processes, each for a different column of C. */
for j := 0 step 1 until N-2 fork DOCOL;
/* The original process reaches this point and does column N-1. */
j := N-1;
DOCOL: /* Executed by N processes, each doing one column of C. */
for i := 0 step 1 until N-1
  begin /* Compute a row element of C. */
    /* Initialize the sum for the inner product. */
    C[i, j] := 0;
    /* Loop over the terms of the inner product. */
    for k := 0 step 1 until N-1
      /* Add the kth inner product term. */
      C[i, j] := C[i, j] + A[i, k]*B[k, j];
    /* End of product term loop. */
  end /* of all rows. */
join N;
```

- Cannot just focus on computation. Analysis must also consider:
 - Data movement
 - Data sharing/contention
 - Data dependencies
 - Problem setup
 - Result collation
 - Some specialized problems need special architectures, i.e. matrix multiply on vector computers (vector is like a column or row of a 2-D array) – these are “traditional” supercomputers.

**End
Of
Today's
Lecture.**

This Slide Left Intentionally Blank