Parallel Architecture 2

Module 3.2

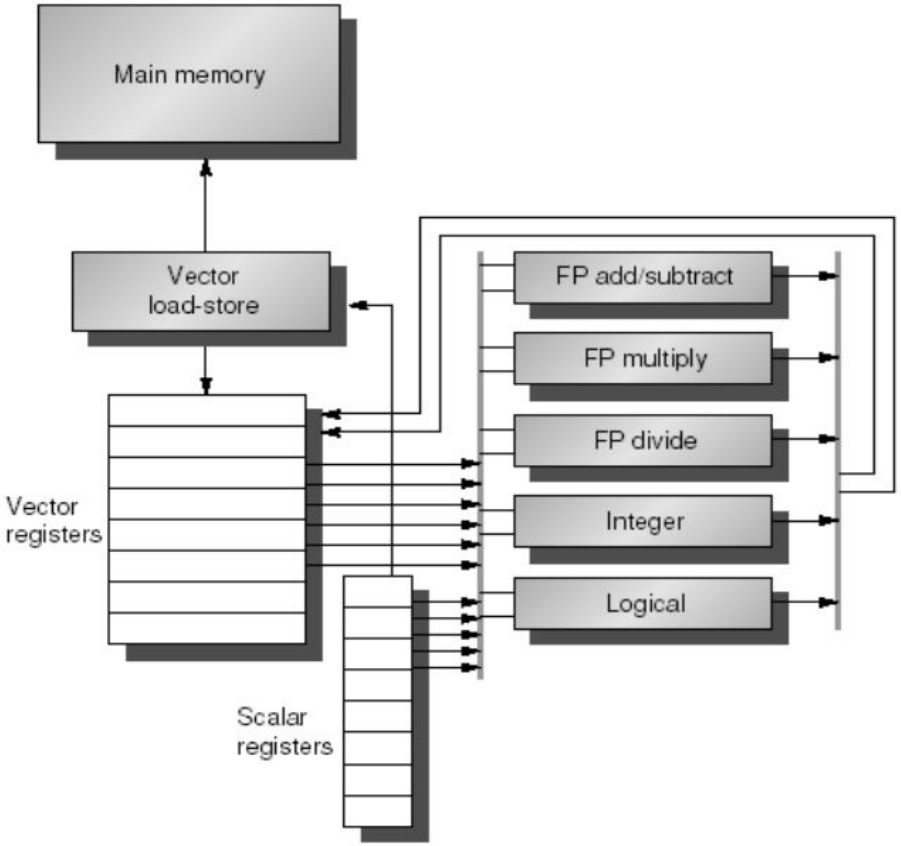
Peter J. Hawrylak

Module Learning Objectives

- Evaluate a loop to see if it is a candidate for parallelization using a vector architecture.
- Employ Foster's methodology (see: Designing and Building Parallel Programs, by Ian Foster, available at: https://www.mcs.anl.gov/~itf/dbpp/) for designing parallel programs for a given architecture.

Vector Architecture Review

- Same operation on multiple data
  □ SIMD approach
  - “Vector” contains the data
  - GPUs excel at this
- Loop-carried dependencies reduce parallelism
  - Must identify these □ parallelizing this will yield incorrect results
  - May be able to restructure code to eliminate □ try this



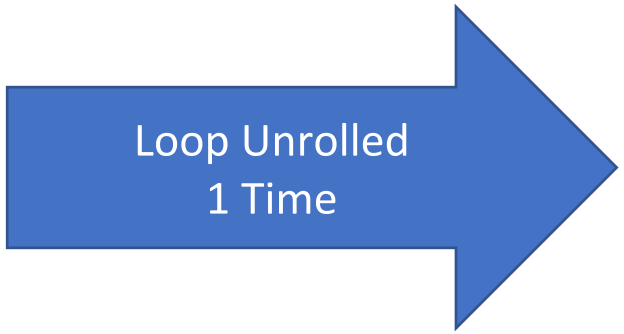
Loop Unrolling

- Vector architectures basically unroll loops
  - Goal is to completely unroll the loop
- Unrolling the loop does more work per iteration
  - Basically undoing the feature of the loop fewer lines of code
  - Small loops are more easily understood and coded let the tools do the behind-the-scenes work and focus on the algorithm

Original code - loop not unrolled

Array y is an array of integers

```
FOR x = 1 to n
      y[x]  y[x] * 7
END FOR
```



Loop unrolled 1 time

Array y is an array of integers

```
FOR x = 1 to n/2
   y[x]  y[x] * 7
   y[x+(n/2)]  y[x+(n/2)] * 7
END FOR
```

Why unroll a loop?

- Reduce/eliminate loop overhead
  - Instructions to update loop counters x in the example
  - Instructions to check loop exit or continue conditions Is x less than n/2?
    - Conditional Branches are VERY COSTLY in terms of hardware and performance
- Excellent fit for SIMD architectures --> GPUs
  - Many math-based problems fall into this category
  - Do not forget vector instructions on CPUs

Original code - loop not unrolled

Array y is an array of integers

```
FOR x = 1 to n
      y[x]  y[x] * 7
END FOR
```

Loop unrolled 1 time

Array y is an array of integers

```
FOR x = 1 to n/2
   y[x]  y[x] * 7
   y[x+(n/2)]  y[x+(n/2)] * 7
END FOR
```

Loop Carried Dependencies

- Iteration X depends on data from iteration (X-1)
  - Serialization is enforced previous iterations must finish first
  - Parallelizing this will yield incorrect results
- Parallel code MUST yield the SAME result as the serial code
  - It must be correct
- Compilers are getting better at identifying these cases and not parallelizing the loop
  - Not all cases can be identified
  - Not all compilers support this

Loop Carried Dependency Example:

Array y is an array of integers

FOR x = 1 to n

y[x] y[x-1]*y[x]

END FOR

Note: Dependencies may be multi-level and difficult to spot for the compiler and the human (coder)

Employing Foster's Methodology (1)

- Parallelize matrix multiplication
  - Multiply 2, N x N matrices, A □ B x C
- Step 1: Partitioning
  - Domain or Functional Decomposition? Both?
    - 1) Multiply row of B by a column of C
    - 2) Sum the product vector together to get 1 element of A

- Develop a partitioning strategy for this problem.

Employing Foster's Methodology (2)

- Parallelize matrix multiplication
  - Multiply 2, N x N matrices, A □ B x C
- Step 2: Communication
  - What messages must be communicated between tasks?
  - What data must be shared?
  - Is communication overhead too large?
  - Can communication overhead be reduced/minimized? If so, how?
    - Look at other methods for computing the product of 2 matrices.
- Develop a communication strategy for this problem.

Employing Foster's Methodology (3)

- Parallelize matrix multiplication
  - Multiply 2, N x N matrices, A □ B x C
- Step 3: Agglomeration
  - What architecture(s) would be well suited for this problem?
  - How will you map tasks to processes?
  - What are the data sharing and communication needs with the grouping?
  - Can the approach easily adapt to different numbers of processing elements? If not, what can be done to make it more adaptable?

- Develop an agglomeration strategy for this problem.

Employing Foster's Methodology (4)

- Parallelize matrix multiplication
  - Multiply 2, N x N matrices, A □ B x C
- Step 4: Mapping
  - How will you map processes to processing elements?
  - Are concurrent operations on different processing elements?
  - What are the data sharing and communication needs with the grouping?
  - What are the costs of communication and manager processes? Can these be reduced?

- Develop an agglomeration strategy for this problem.

Summary

- Evaluate a loop to see if it is a candidate for parallelization using a vector architecture.
  - No loop-carried dependencies
  - Loop-carried dependencies will lead to incorrect results
- Employ Foster's methodology for designing parallel programs for a given architecture.
  - Create multiple options at each step – select best option but have other options if later steps show approach does not scale well
  - Recheck scalability after each step.