

COMP 322: Fundamentals of Parallel Programming

Lecture 1: Task Creation & Termination (async, finish)

Mack Joyner
mjoyner@rice.edu

<http://comp322.rice.edu>



Special Thanks to Vivek Sarkar!



Your Teaching Staff!

- TAs
 - Elian Ahmar, Timothy Goh, Kelly Park, Tucker Reinhardt, Mantej Singh, Minh Vu, Thanh Vu, Robert Walsh, Frederick Wang, Xincheng Wang, Yidi Wang
- Instructor
 - Mack Joyner



Course Syllabus

- Fundamentals of Parallel Programming taught in three modules
 - 1.Parallelism
 - 2.Concurrency
 - 3.Locality & Distribution
- Each module is subdivided into units, and each unit into topics
- Lecture and lecture handouts will introduce concepts using pseudocode notations
- Labs and programming assignments will be in Java 8 (not moving to Java 11 this semester)
 - Initially, we will use the Habanero-Java (HJ) library developed at Rice as a pedagogic parallel programming model
 - HJ-lib is a Java 8 library (no special compiler support needed)
 - HJ-lib contains many features that are easier to use than standard Java threads/tasks, and are also being added to future parallel programming models
 - Later, we will learn parallel programming using standard Java libraries, and combinations of Java libs + HJ-lib



Grade Policies

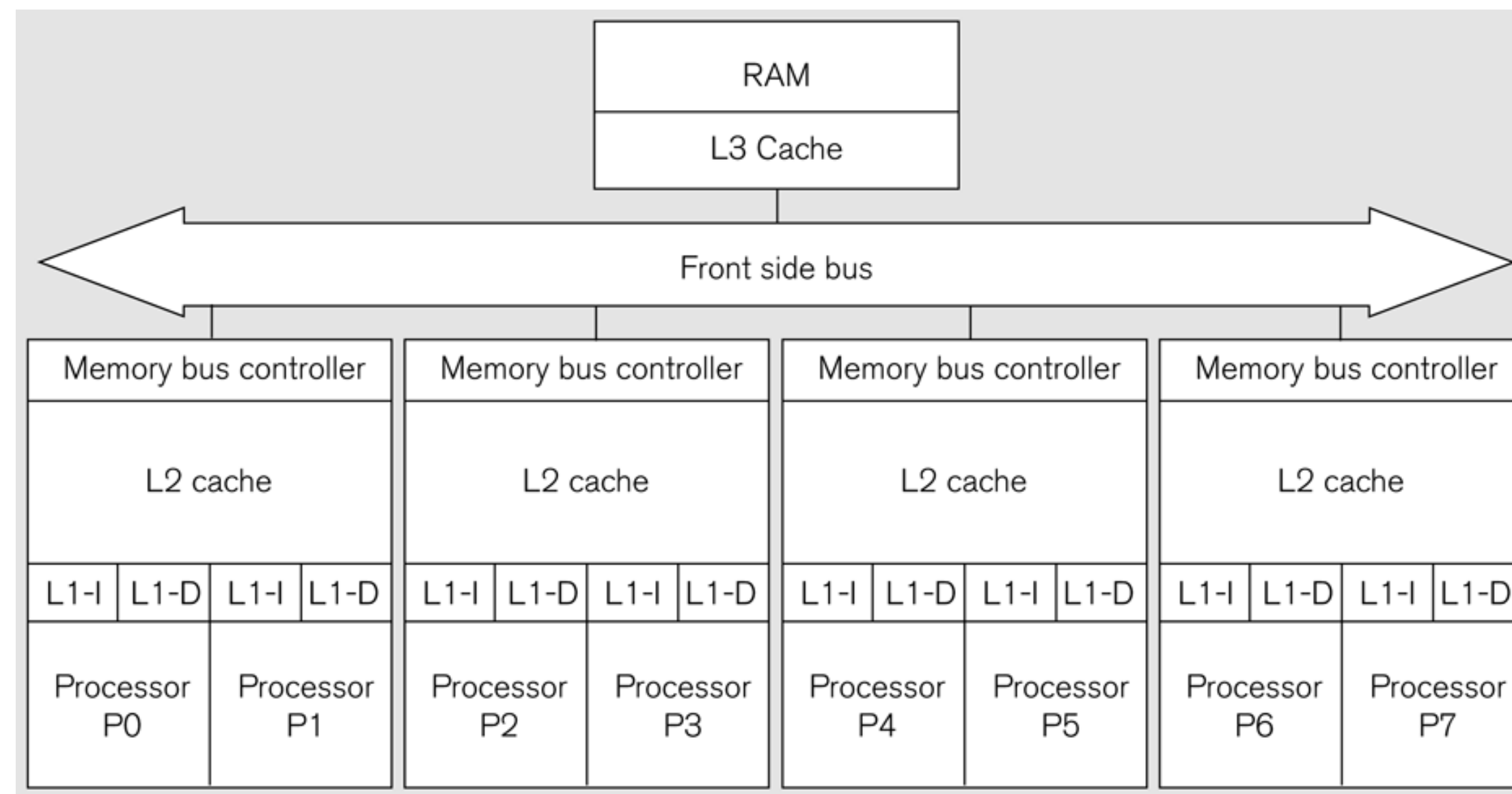
Course Rubric

- Homework (4) 40% (written + programming components)
 - Weightage proportional to # weeks for homework
- Exams (2) 40% (scheduled midterm + scheduled final)
- Labs 10% (labs need to be submitted by Monday)
- Quizzes 5% (on-line quizzes on Canvas)
- Class Participation 5% (in-class worksheets)



What is Parallel Computing

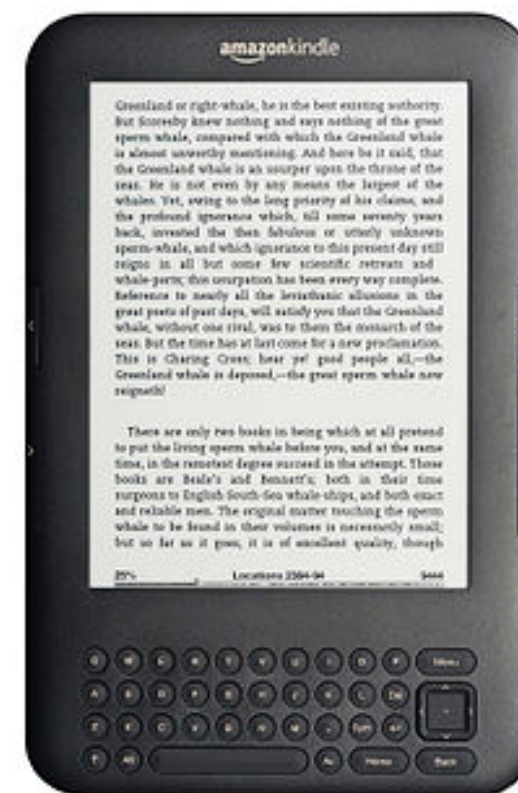
- **Parallel computing:** using multiple processors in parallel to solve problems more quickly than with a single processor and/or with less energy
- Example of a parallel computer
 - An 8-core Symmetric Multi-Processor (SMP) consisting of four dual-core chip microprocessors (CMPs)

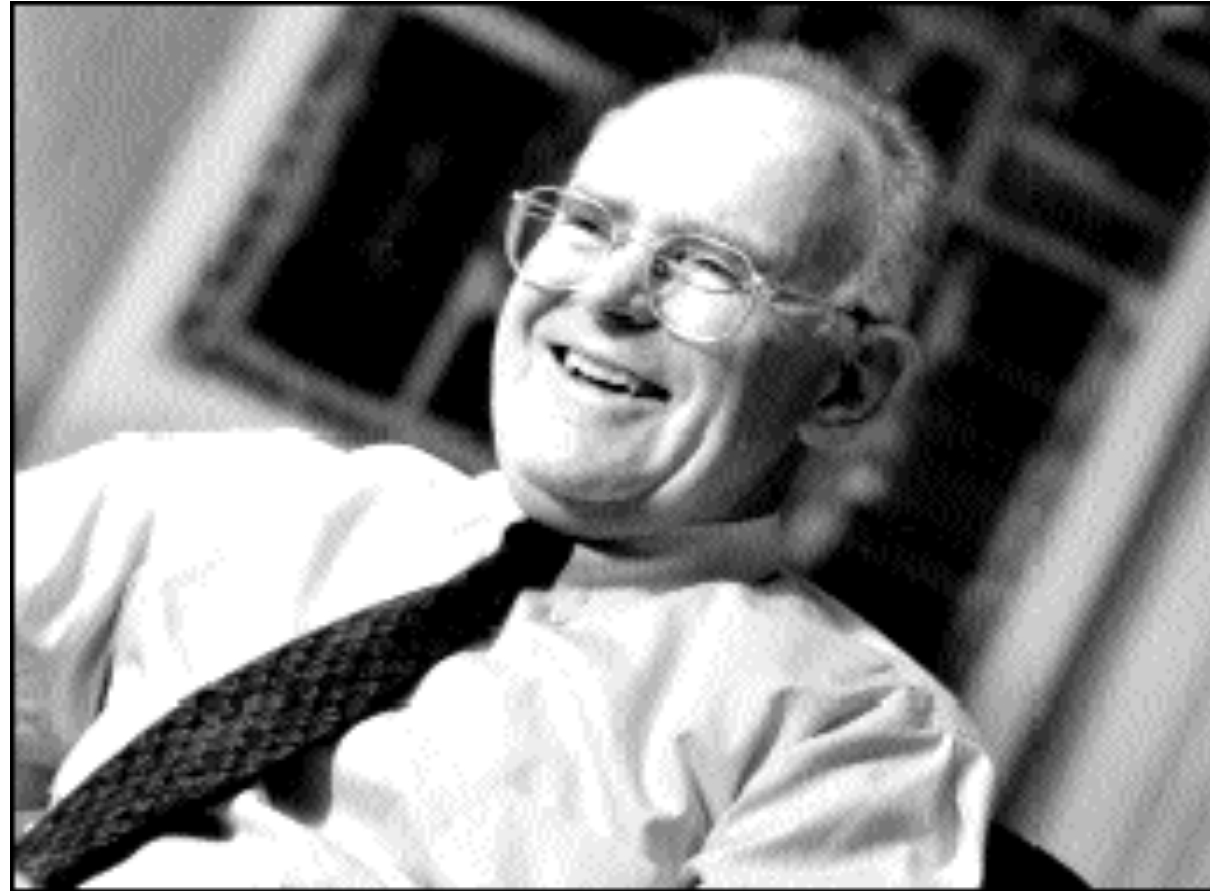


Source: Figure 1.5 of Lin & Snyder book, Addison-Wesley, 2009



All Computers are Parallel Computers



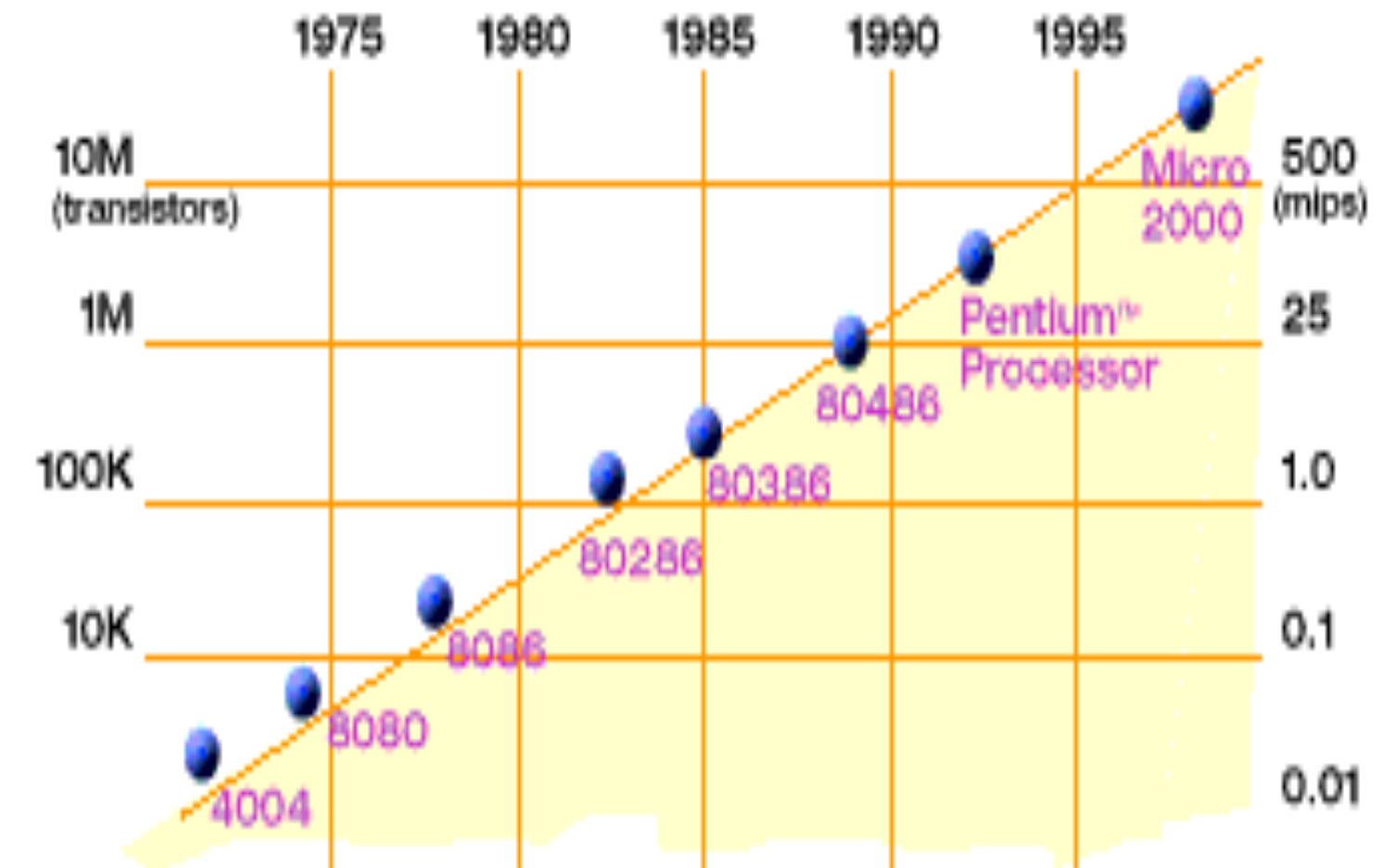


Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 1-2 years (Moore's Law)

⇒ area of transistor halves every 1-2 years

⇒ feature size reduces by $\sqrt{2}$ every 1-2 years

Slide source: Jack Dongarra



Dennard Scaling states that power for a fixed chip area remains constant as transistors grow smaller



Parallelism Saves Power (Simplified Analysis)

Maximum Frequency is capped by Voltage

→ Power is proportional to (Frequency)³

Baseline example: single 1GHz core with power P

Option A: Increase clock frequency to 2GHz → Power = 8P

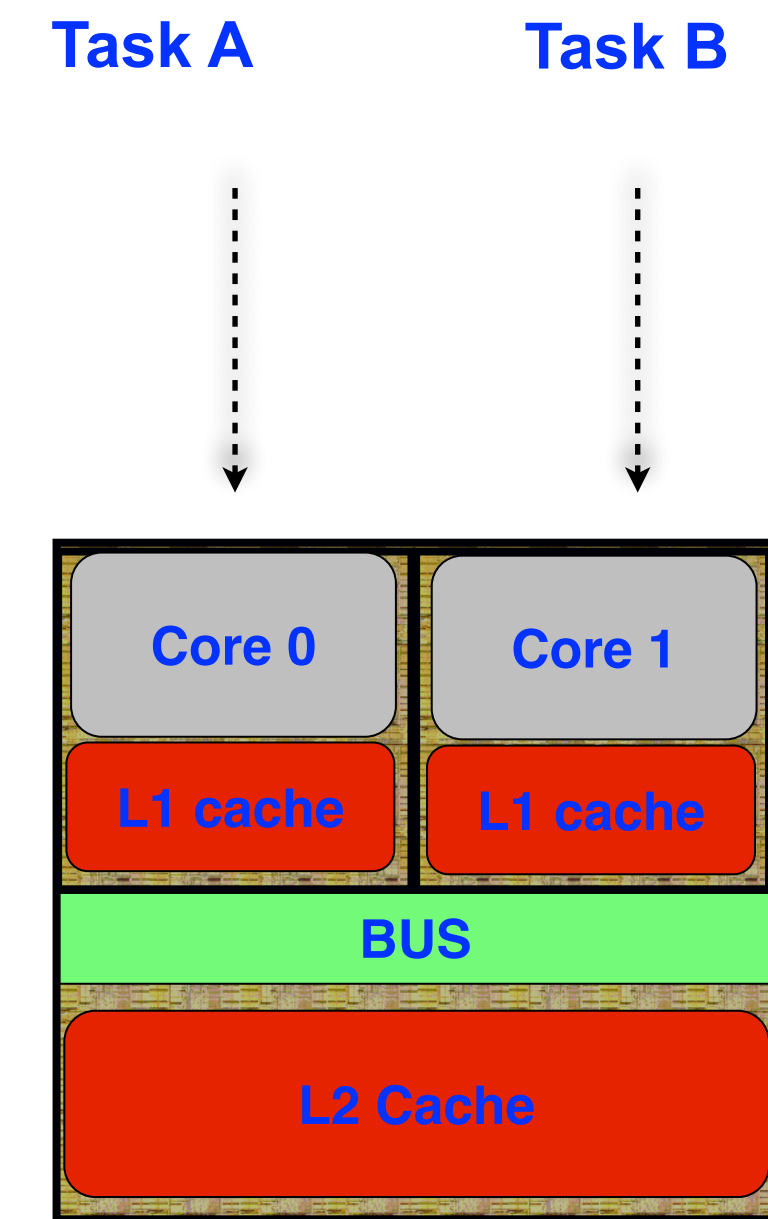
Option B: Use 2 cores at 1 GHz each → Power = 2P

- Option B delivers same performance as Option A with 4x less power ... provided software can be decomposed to run in parallel!



What is Parallel Programming

- Specification of operations that can be executed in parallel
- A parallel program is decomposed into sequential subcomputations called *tasks*
- Parallel programming constructs define task creation, termination, and interaction



Schematic of a dual-core Processor



Example of a Sequential Program: Computing sum of array elements

Algorithm 1: Sequential ArraySum

Input: Array of numbers, X .

Output: $sum =$ sum of elements in array X .

$sum \leftarrow 0;$

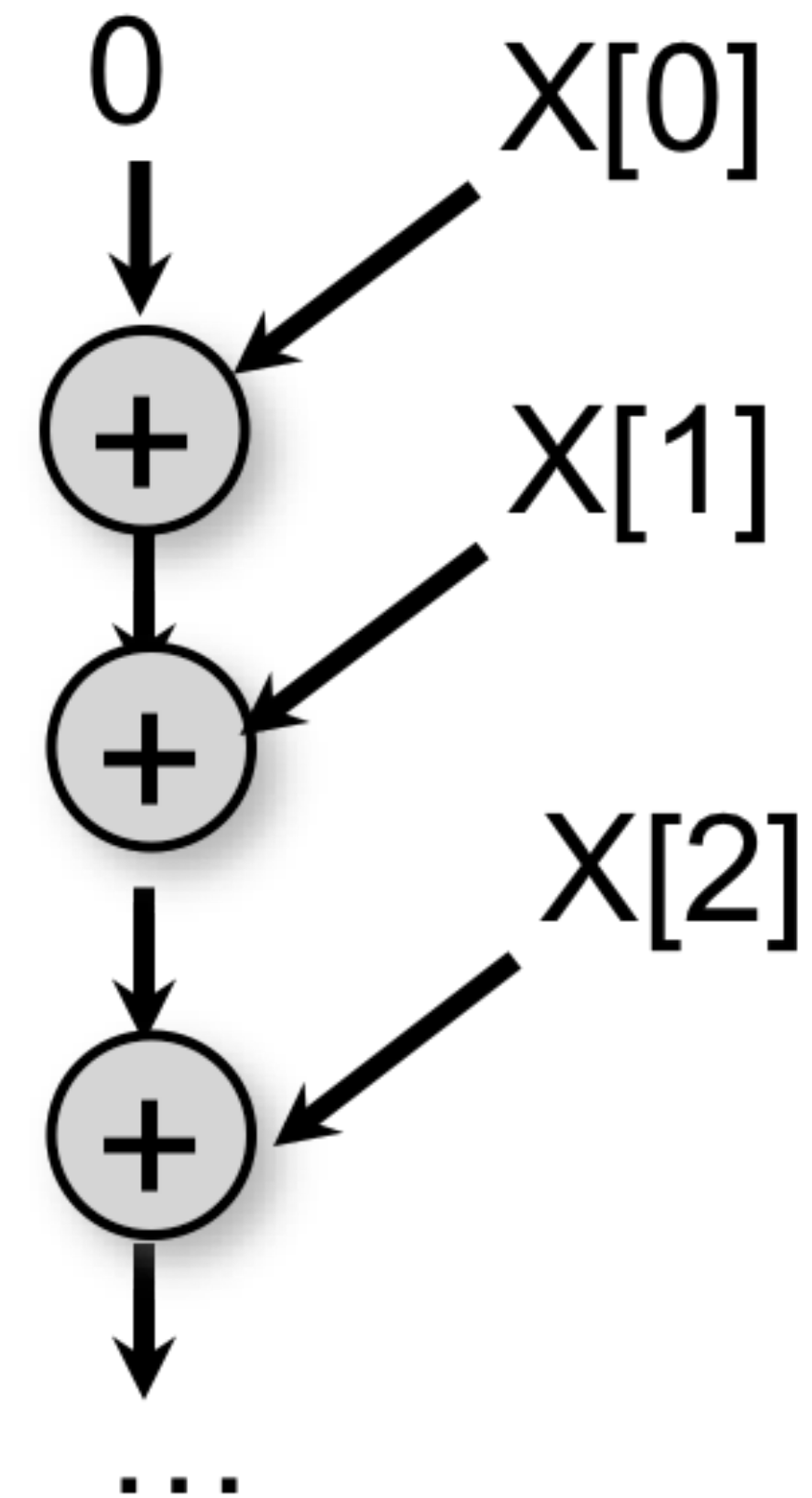
for $i \leftarrow 0$ **to** $X.length - 1$ **do**

$sum \leftarrow sum + X[i];$

return $sum;$

Observations:

- The decision to sum up the elements from left to right was arbitrary
- The computation graph shows that all operations must be executed sequentially



Async and Finish Statements for Task Creation and Termination

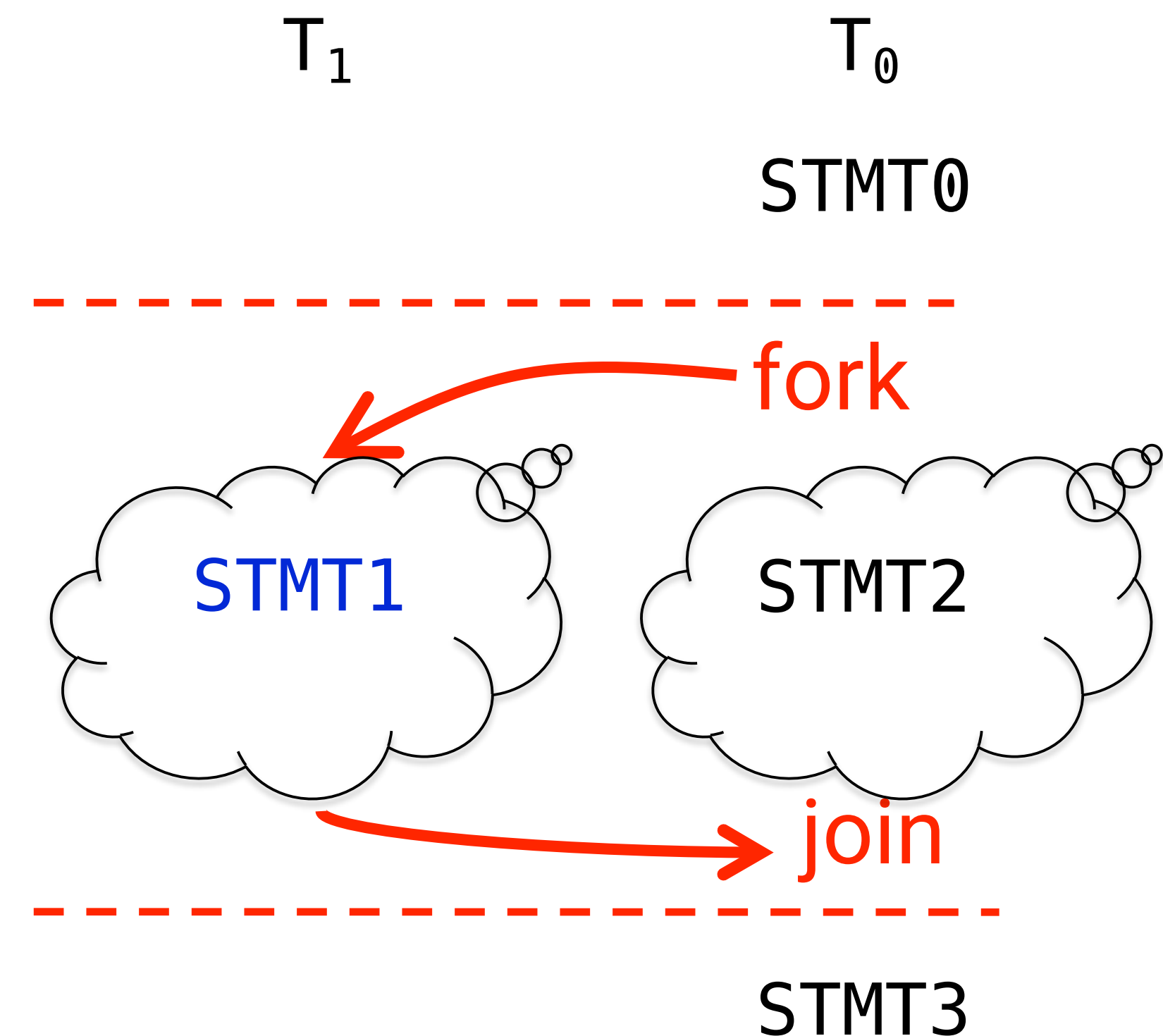
async S

- Creates a new child task that executes statement S

```
// T0(Parent task)
STMT0;
finish { //Begin finish
  async {
    STMT1; //T1(Child task)
  }
  STMT2; //Continue in T0
} //End finish (wait for T1)
STMT3; //Continue in T0
```

finish S

- Execute S, but wait until *all* asyncs in S's scope have terminated.



Example of a Sequential Program: Computing sum of array elements

Algorithm 1: Sequential ArraySum

Input: Array of numbers, X .

Output: $sum =$ sum of elements in array X .

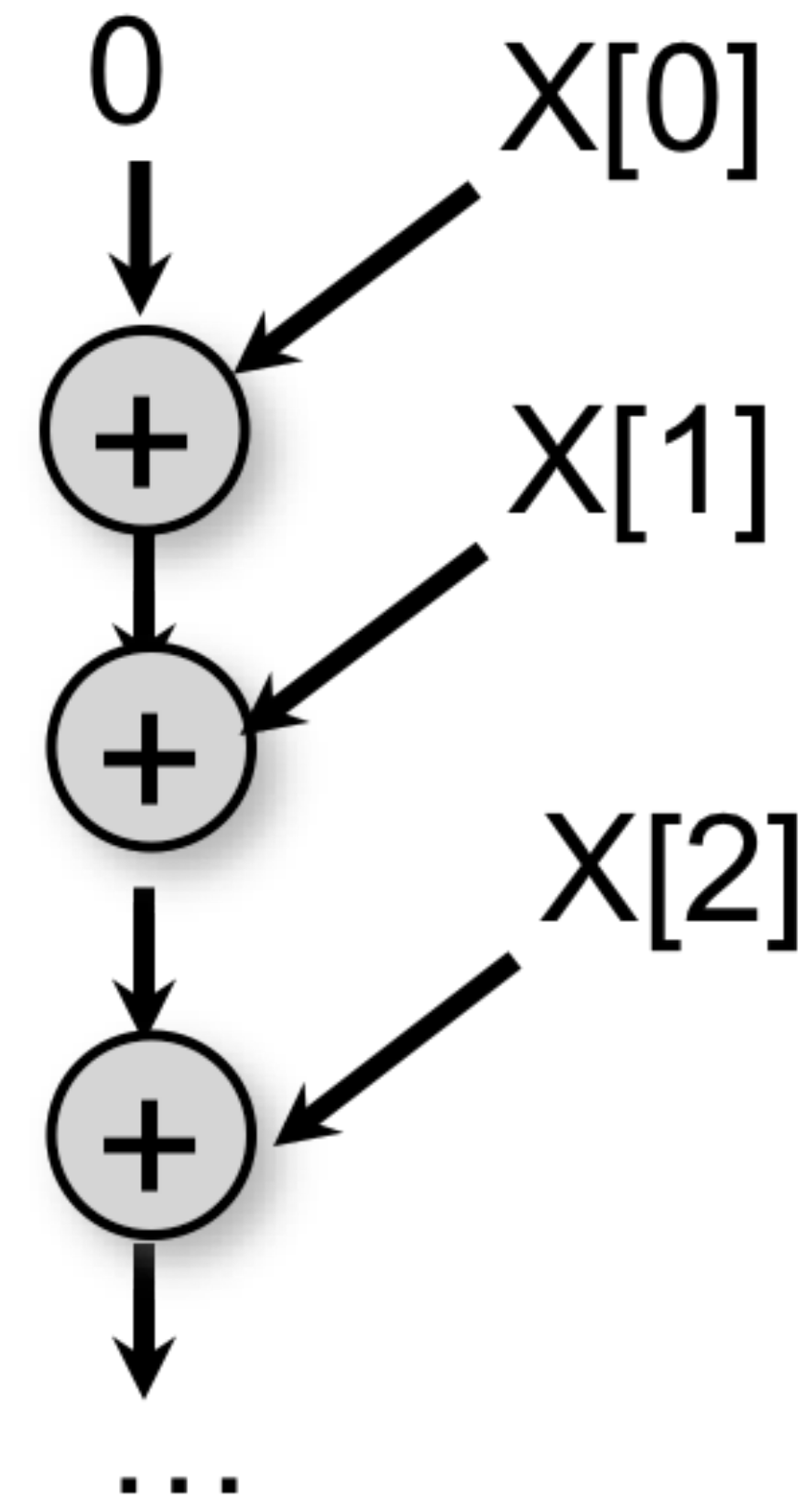
$sum \leftarrow 0;$

for $i \leftarrow 0$ **to** $X.length - 1$ **do**

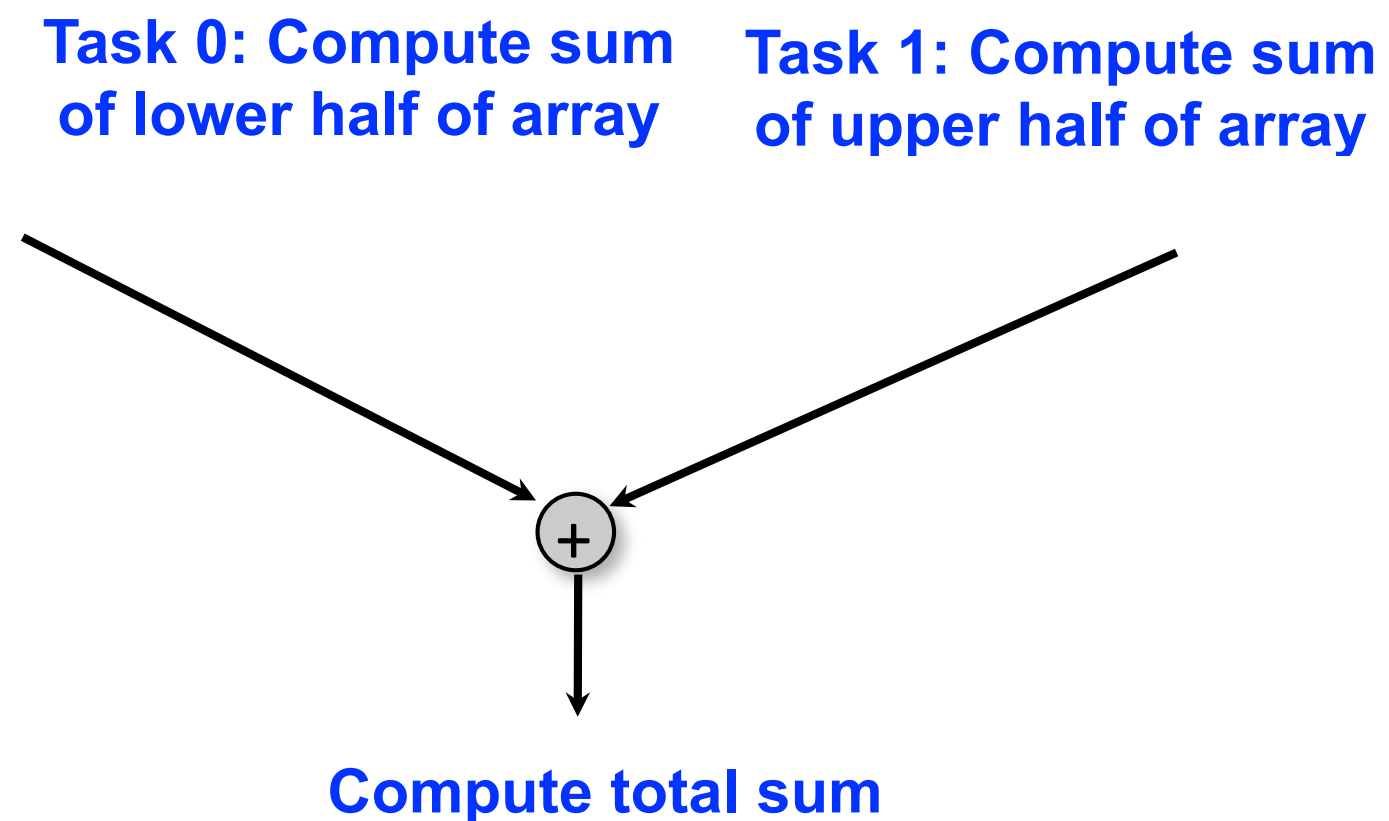
$sum \leftarrow sum + X[i];$

return $sum;$

Can you insert an `async/finish` anywhere to improve performance?



Parallelization Strategy for 2 cores (Two-way Parallel Array Sum)



Basic idea:

- Decompose problem into two tasks for partial sums
- Combine results to obtain final answer
- Parallel divide-and-conquer pattern



Two-way Parallel Array Sum using async & finish constructs

Algorithm 2: Two-way Parallel ArraySum

Input: Array of numbers, X .

Output: $sum =$ sum of elements in array X .

// Start of Task T1 (main program)

$sum1 \leftarrow 0$; $sum2 \leftarrow 0$;

// Compute $sum1$ (lower half) and $sum2$ (upper half) in parallel.

finish{

async{

 // Task T2

for $i \leftarrow 0$ **to** $X.length/2 - 1$ **do**

$sum1 \leftarrow sum1 + X[i]$;

 };

async{

 // Task T3

for $i \leftarrow X.length/2$ **to** $X.length - 1$ **do**

$sum2 \leftarrow sum2 + X[i]$;

 };

};

// Task T1 waits for Tasks T2 and T3 to complete

// Continuation of Task T1

$sum \leftarrow sum1 + sum2$;

return sum ;



Next Steps

- IMPORTANT:
 - This week's lab is at 1:30pm on Tuesday or 4:50pm on Thursday (Zoom)
 - Watch [videos](#) for topics 1.2 & 1.3 for next lecture on Wednesday
- HW1 will be assigned on Jan 27th and be due on Feb 10th. (Homework is normally due on Wednesdays.)
- Each quiz (to be taken online on Canvas) will be due on the Friday after the unit is covered in class. The first quiz for Unit 1 (topics 1.1 - 1.5) is due by Feb 5th.
- See course web site for syllabus, work assignments, due dates, ...
 - <http://comp322.rice.edu>



Office Hours

- Regular office hour schedule can be found at [Office Hours](#) link on course web site
- Send email to instructor (mjoyner@rice.edu) if you need to meet some other time this week
- And remember to post and answer questions on Piazza!



Worksheet #1

Honor Code Policy for Worksheets: You are free to discuss all aspects of in-class worksheets with your other classmates, the teaching assistants and the professor during the class. You can work in a group and write down the solution that you obtained as a group. If you work on the worksheet outside of class (e.g., due to an absence), then it must be entirely your individual effort, without discussion with any other students. If you use any material from external sources, you must provide proper attribution. You should submit the worksheet in Canvas.

1) Parallelizing your weekday/weekend tasks!

Consider the sequential list of weekday/weekend tasks below. Assume that you have an unbounded number of helpers to help you with your chores and tasks. Insert async and finish pseudocode annotations to maximize parallelism, while ensuring that the parallel version has no unintended/undesirable outcomes. Make any reasonable assumptions e.g., you only have one fridge, you need to watch videos in order, you have access to multiple washers & dryers, you can reorder statements so long as you don't change the outcome, etc.

Watch COMP 322 video for topic 1.2 by 1:30pm on Wednesday

Watch COMP 322 video for topic 1.3 by 1:30pm on Wednesday

Make your bed

Clean out your fridge

Buy food supplies and store them in fridge

Run load 1 in washer

Run load 2 in washer

Run load 1 in dryer

Run load 2 in dryer

Call your family

Post on Facebook that you're done with all your tasks!

