COMP 322 / ELEC 323: Fundamentals of Parallel Programming

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

Instructors: Mack Joyner, Zoran Budimlić
Department of Computer Science, Rice University
{mjoyner, zoran}@rice.edu

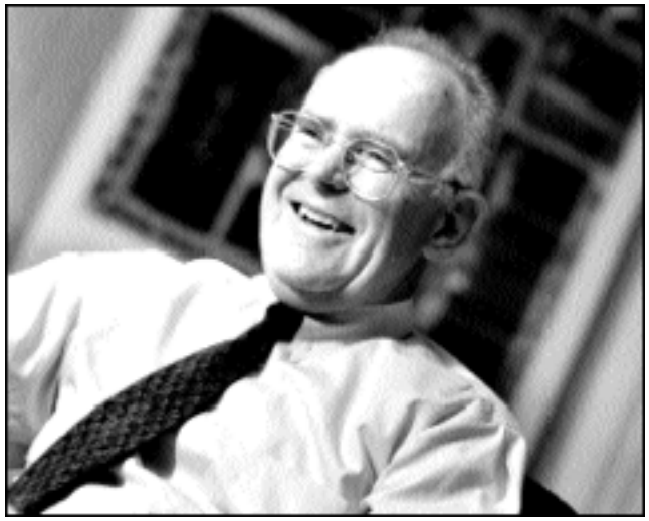
http://comp322.rice.edu

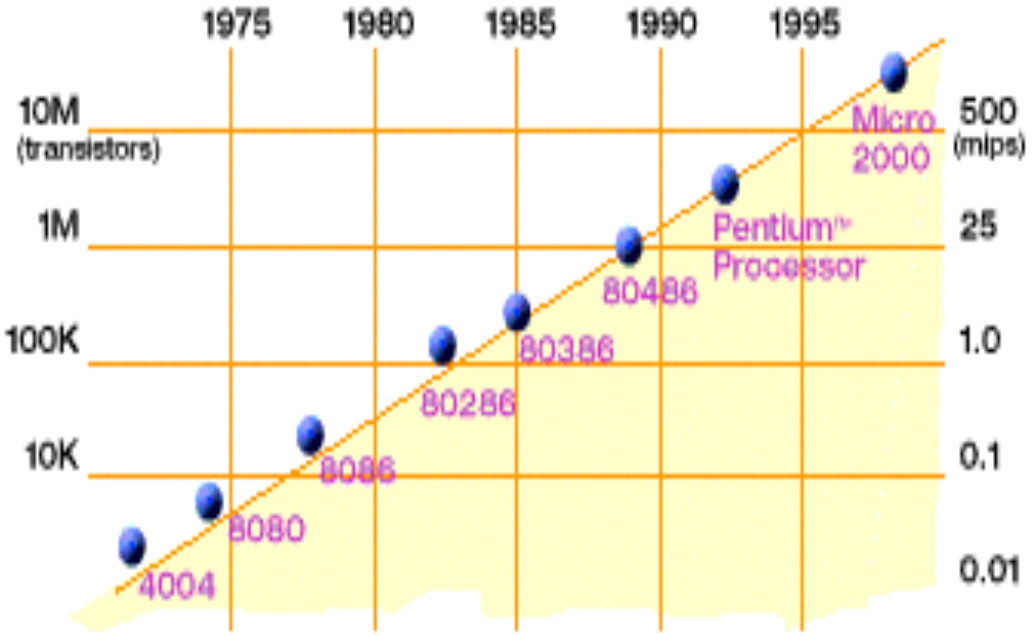
Special thanks to Vivek Sarkar!

Your Teaching Staff!

- **Undergraduate TAs**
  - — Liam Bonnage, Harrison Brown, Mustafa El-Gamal, Krishna Goel, Ryan Green, Ryan Han, Rishu Harpavat, Namanh Kapur, Tian Lan, Tam Le, Will LeVine, Eva Ma, Hamza Nauman, Rutvik Patel, Aryan Sefidi, Jeemin Sim, Tory Songyang, Jiaqi Wang, Erik Yamada, Yifan Yang, Aydin Zanagar
- **Graduate TAs**
  - — Jonathan Sharman, Srdjan Milakovic
- **Instructors**
  - — Mack Joyner, Zoran Budimlić

Moore's Law and Dennard Scaling





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

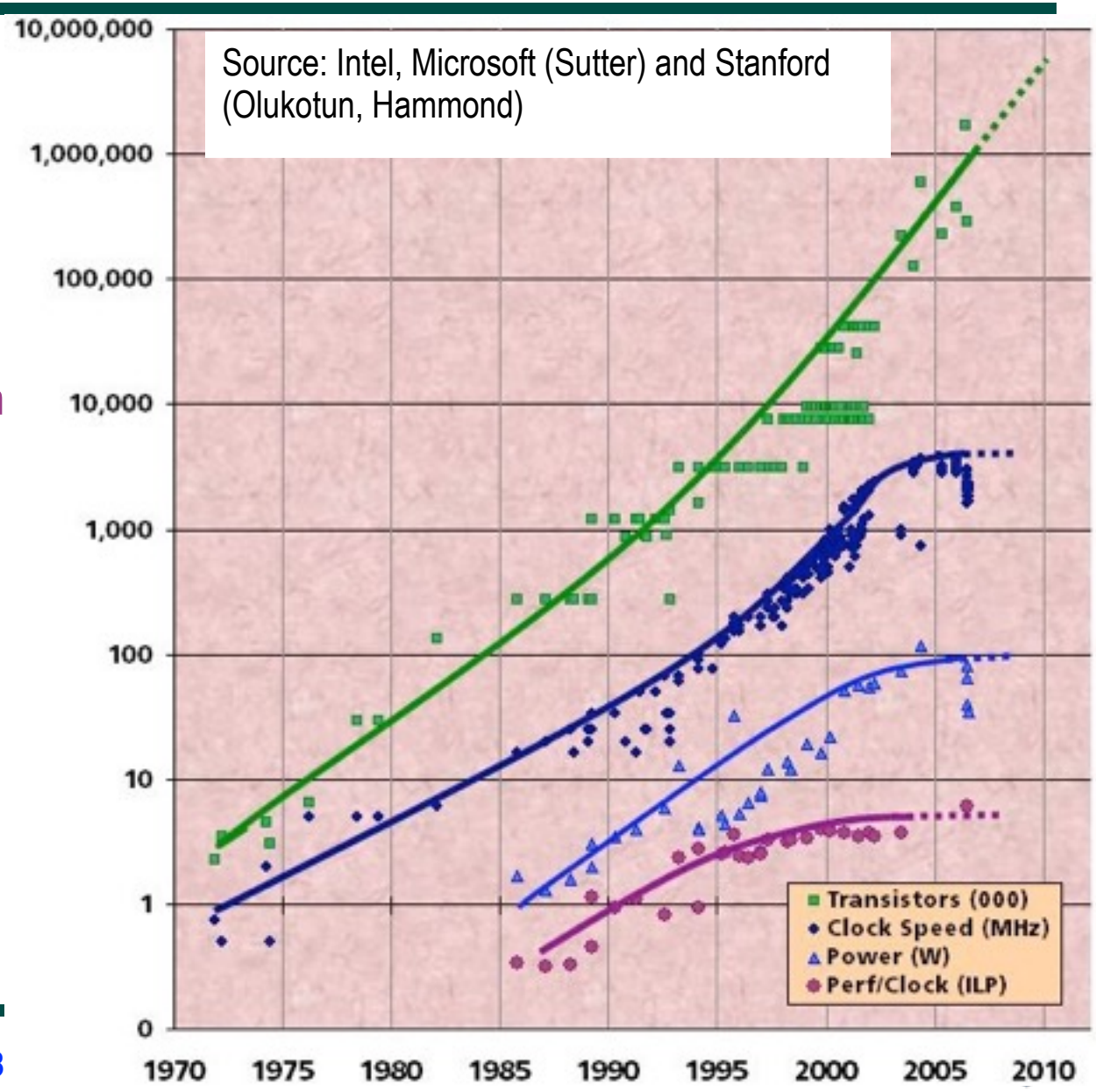
⇒ feature size reduces by √2 every 1-2 years

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

Slide source: Jack Dongarra

Recent Technology Trends

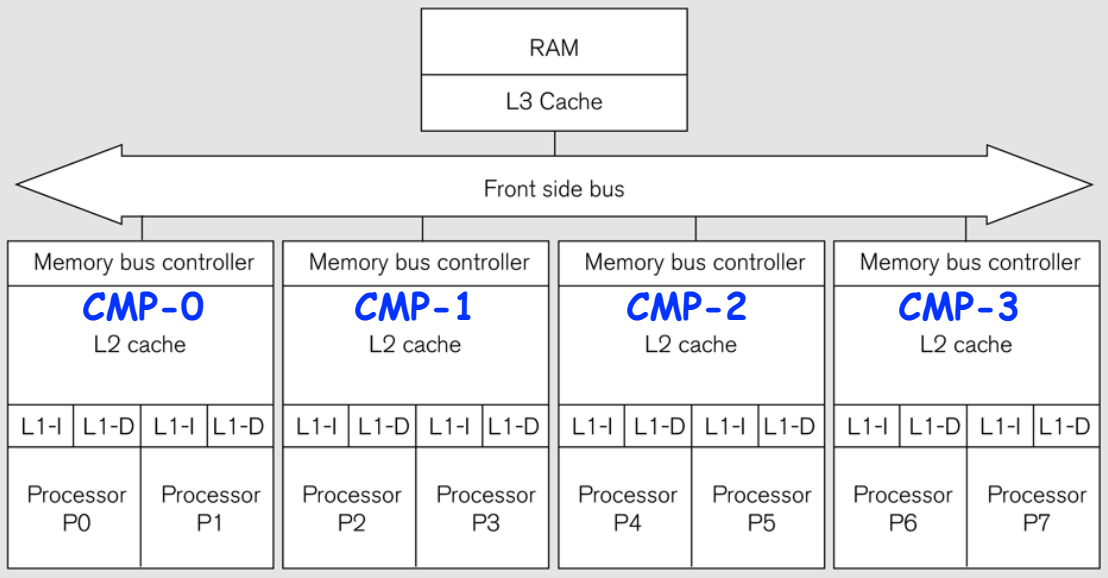
- **Chip density (transistors)** is increasing ~2x every 2 years
- **Clock speed** is plateauing below 10 GHz so that **chip power** stays below 100W
- **Instruction-level parallelism (ILP)** in hardware has also plateaued below 10 instructions/cycle
- ⇒ ***Parallelism must be managed by software!***



What is Parallel Computing?

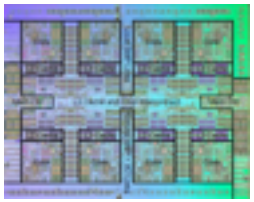
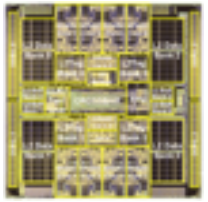
- **Parallel computing:** using multiple processors in parallel to solve problems more quickly than with a single processor and/or with less energy

- Example of a parallel computer
  - —An 8-core Symmetric Multi-Processor (SMP) consisting of four dual-core chip microprocessors (CMPs)

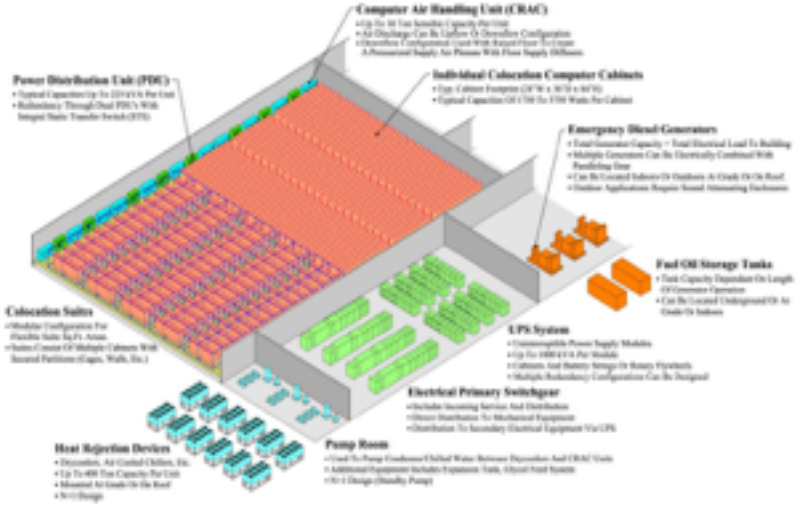


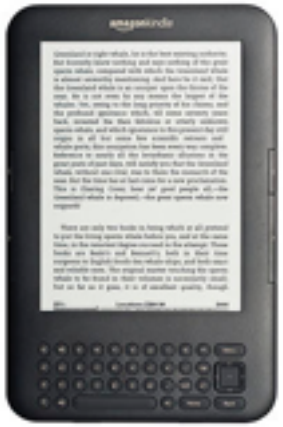
Source: Figure 1.5 of Lin & Snyder book, Addison-Wesley, 2009

All Computers are Parallel Computers --- Why?






Parallelism Saves Power (Simplified Analysis)

Nowadays (post Dennard Scaling), Power ~ (Capacitance) * (Voltage)^2 * (Frequency)

and maximum Frequency is capped by Voltage

➔ Power is proportional to (Frequency)^3

Baseline example: single 1GHz core with power P

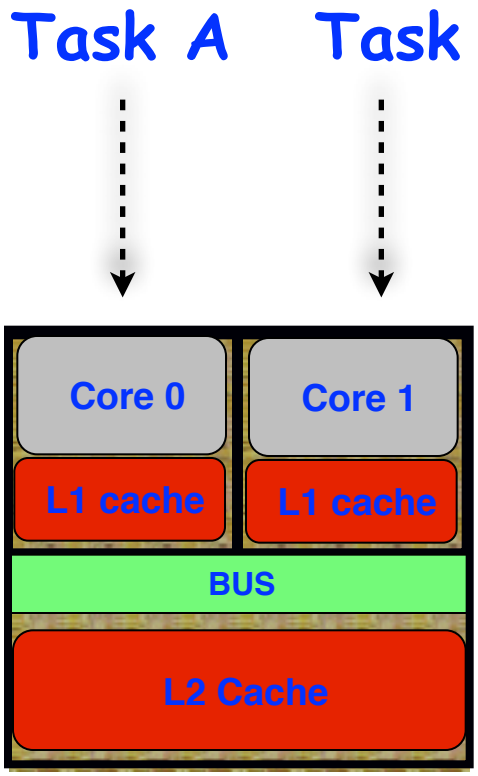
Option A: Increase clock frequency to 2GHz ➔ Power = 8P

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

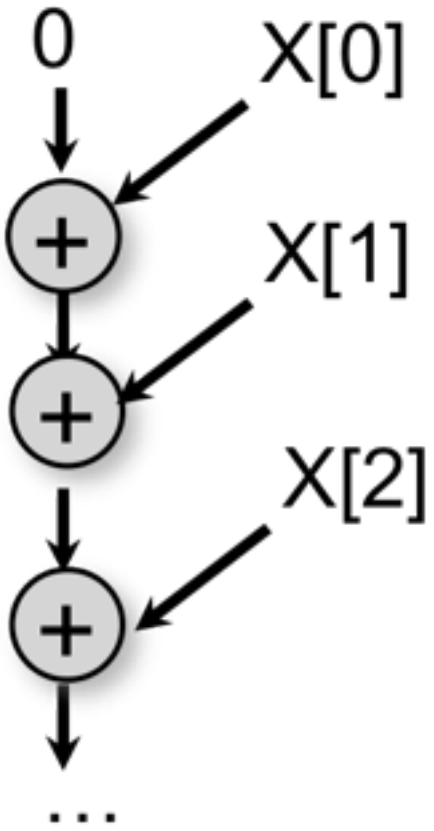
Example of a Sequential Program: Computing the sum of array elements

Algorithm 1: Sequential ArraySum

Input: Array of numbers, X.
Output: sum = sum of elements in array X.

```
sum ← 0;
for i ← 0 to X.length − 1 do
    sum ← sum + X[i];
return sum;
```

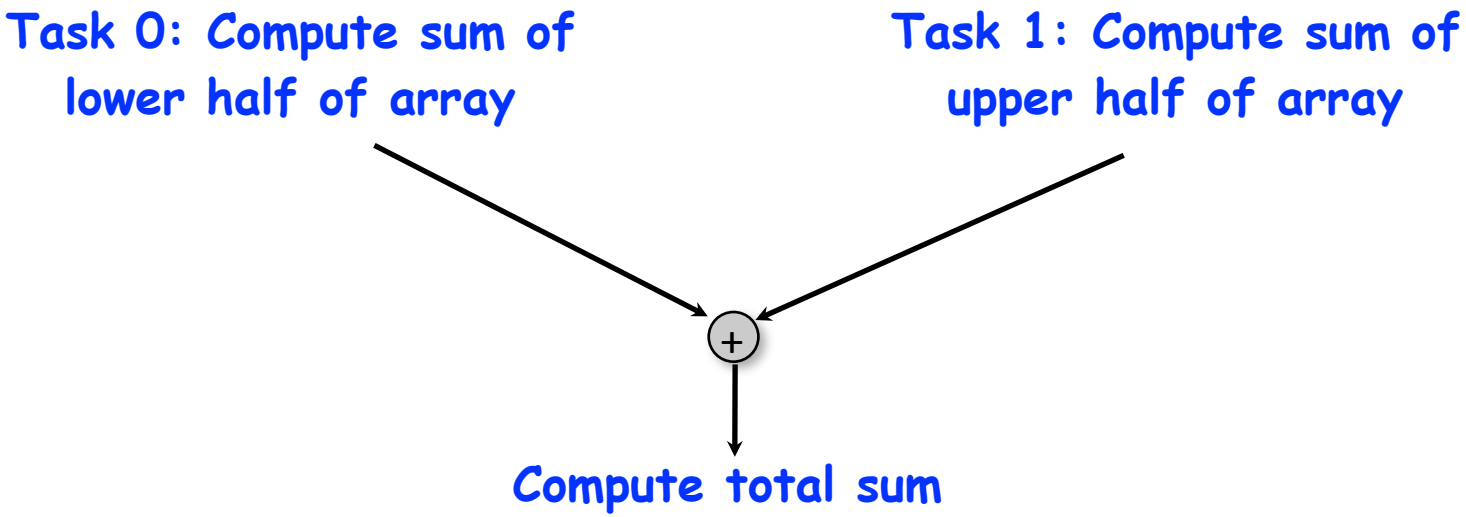
Computation Graph



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

Parallelization Strategy for two 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

Async and Finish Statements for Task Creation and Termination (Pseudocode)

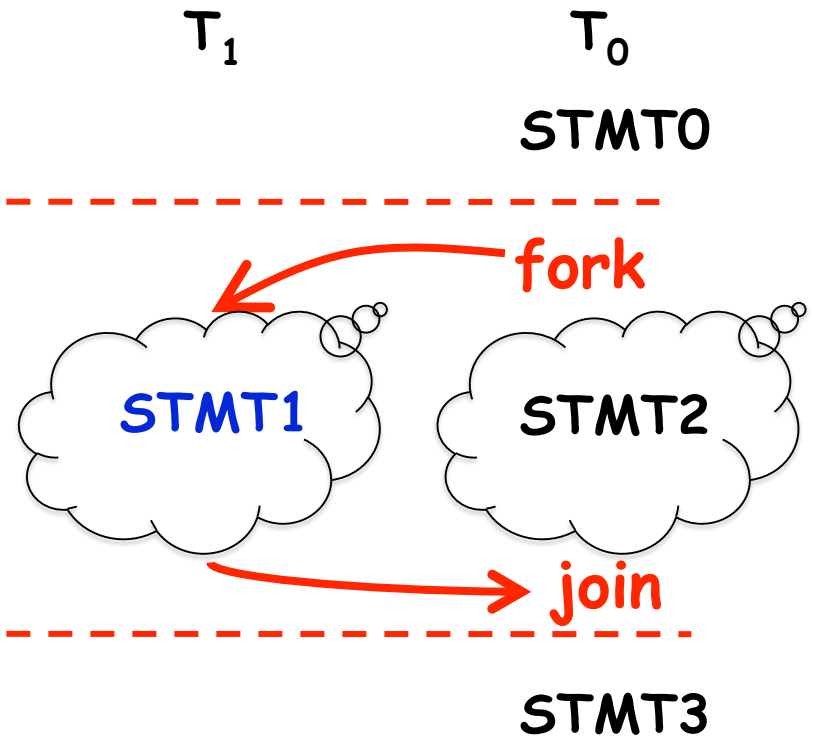
async S

- Creates a new child task that executes statement S

finish S

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

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



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 ← 0; sum2 ← 0;
// Compute sum1 (lower half) and sum2 (upper half) in parallel.
finish{
    async{
        // Task T2
        for i ← 0 to X.length/2 − 1 do
            sum1 ← sum1 + X[i];
    };
    async{
        // Task T3
        for i ← X.length/2 to X.length − 1 do
            sum2 ← sum2 + X[i];
    };
};
// Task T1 waits for Tasks T2 and T3 to complete
// Continuation of Task T1
sum ← sum1 + sum2;
return sum;
```


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

Next Steps

- **IMPORTANT:**
  - —Bring your laptop to this week's lab at 4pm on Thursday (HH 100)
  - —Watch videos for topics 1.2 & 1.3 for next lecture on Wednesday
- **HW1 will be assigned on Jan 9th and be due on Jan 23rd. (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 Jan 25.**
- **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 instructors (mjoyner@rice.edu, zoran@rice.edu) if you need to meet some other time this week
- And remember to post questions on Piazza!