

CSE305 Computer Architecture

Introduction

Daehoon Kim
Department of EECS, DGIST

Course Information

- **Instructor**
 - Daehoon Kim (dkim@dgist.ac.kr)
 - Office Hours: by appointment (E3-312)
- **TA**
 - Seonmu Oh (smoh@dgist.ac.kr)
 - Jungbo Kim (jbkim@dgist.ac.kr)
 - Sungju Kim (sungju_kim@dgist.ac.kr)
 - Kyeonghyeon Ryu (khryu@dgist.ac.kr)
 - Office Hours: by appointment
- **Textbook**
 - Computer Organization and Design, The Hardware/Software Interface, 5th (or 6th) Edition (MIPS edition), David Patterson, John Hennessy, Elsevier.
 - Computer Architecture: A Quantitative Approach (6th Edition)
- **Website**
 - lms.dgist.ac.kr
- **Prerequisite: Programming skills for class projects**
 - *Must use C, C++ for projects*

More Course Information

- **Grading (tentative)**
 - **Exams (midterm+final): 50% (25%+25%)**
 - **Projects: 40% (4 programming projects)**
 - **Attendance and participation: 10%**

Topics

- **Performance metric**
- **Instruction set architecture**
- **Processor architecture**
 - Single cycle data path
 - Pipelining
 - Instruction-level Parallelism, Out-of-order Execution
- **Memory hierarchy**
 - Caches
 - Virtual Memory
 - TLB
- **Multi-core processor architecture**

Problem Specification to Physics

Specification

compute the fibonacci sequence

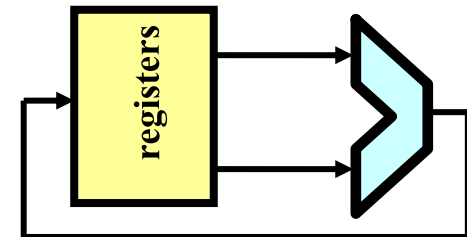
Program

```
for(i=2; i<100; i++) {
    a[i] = a[i-1]+a[i-2];}
```

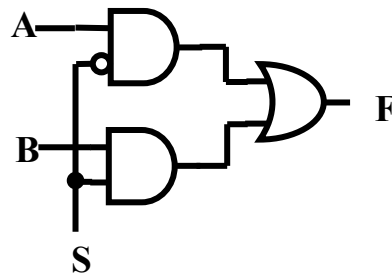
ISA (Instruction Set Architecture)

```
load r1, a[i];
add  r2, r2, r1;
```

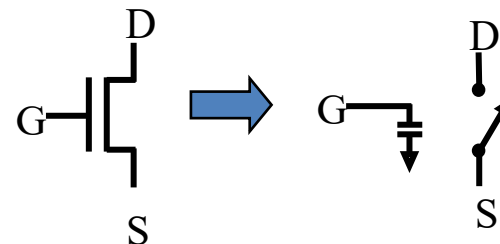
microArchitecture



Logic

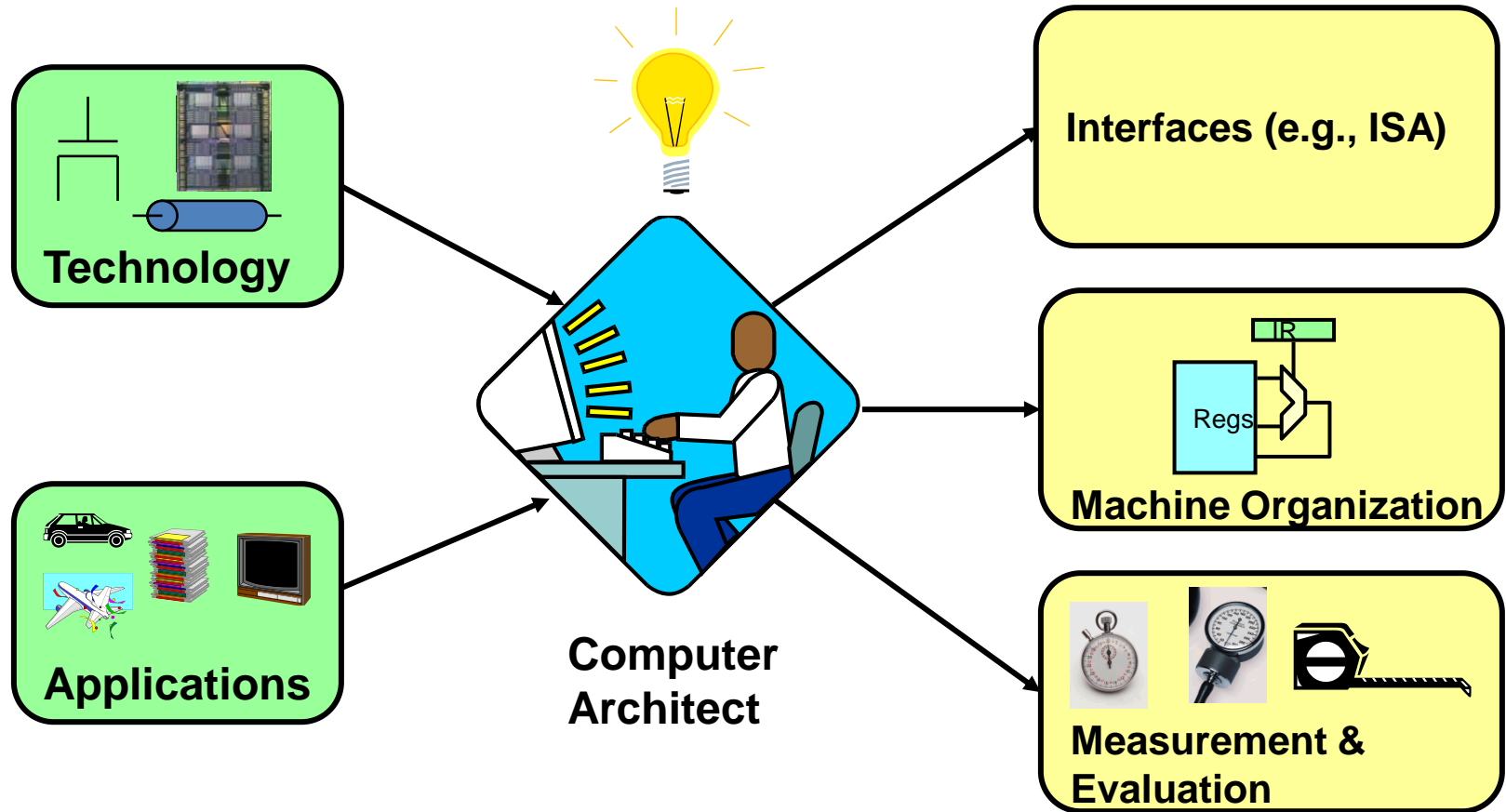


Transistors



Physics/Chemistry

What Is Computer Architecture?

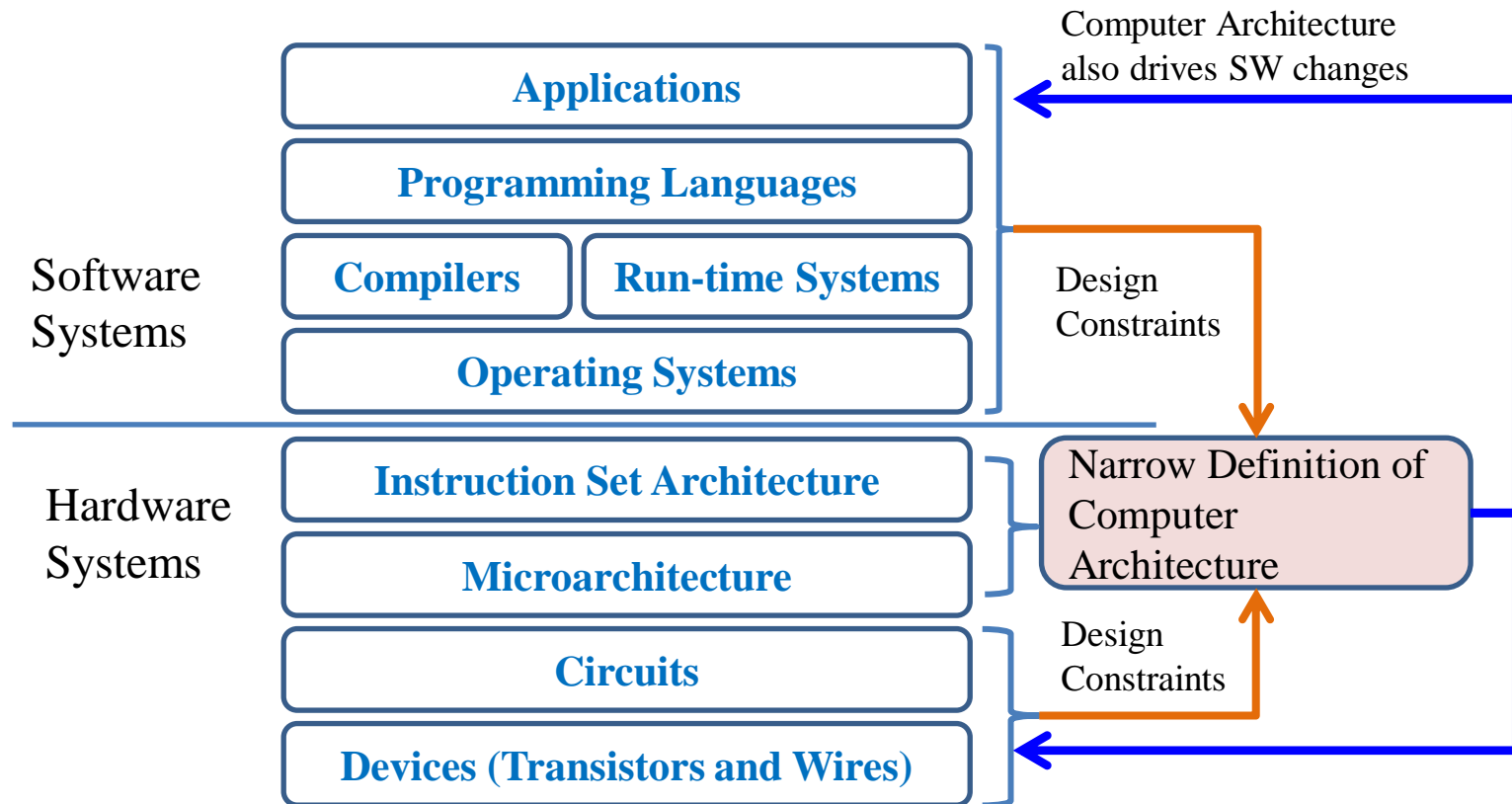


Great Ideas In Computer Architecture

- **Abstraction**
- **Moore's Law**
- **Principle of Locality: Memory Hierarchy**
- **Parallelism**
- **Amdahl's Law**

Computer System Abstraction

- **Abstraction helps us deal with complexity**
 - Hide lower-level detail
- *Instruction set architecture (ISA)*
 - The hardware/software interface



Computer Architecture

- **Instruction Set Architecture (ISA)**
 - Interface between HW and SW systems
 - Hard to change due to compatibility issues
 - Various ISAs: MIPS, x86, Power, ARM etc
 - ISA evolves: Intel and AMD have been extending x86 (32 and 64bit support, VM support)
- **Microarchitecture**
 - Implementation of ISA
 - Organization of processors (and I/O subsystems)
 - Achieve design goal with multiple constraints
 - Examples
 - Depth of pipelines (and hazard resolution)
 - Cache sizes and organization
 - How to implement out-of-order execution
 - Various speculative mechanisms for performance

Program to Machine Code

- High-level language program (in C)

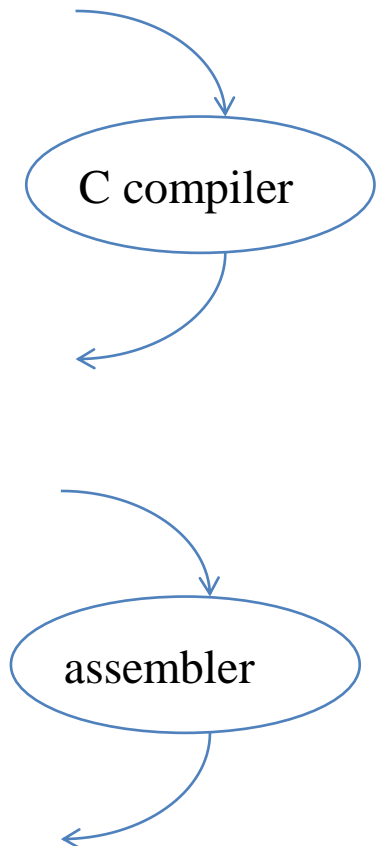
```
swap (int v[], int k)
(
    int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
)
```

- Assembly language program (for MIPS)

```
swap:      sll      $2, $5, 2
           add      $2, $4, $2
           lw       $15, 0($2)
           lw       $16, 4($2)
           sw       $16, 0($2)
           sw       $15, 4($2)
           jr       $31
```

- Machine (object, binary) code (for MIPS)

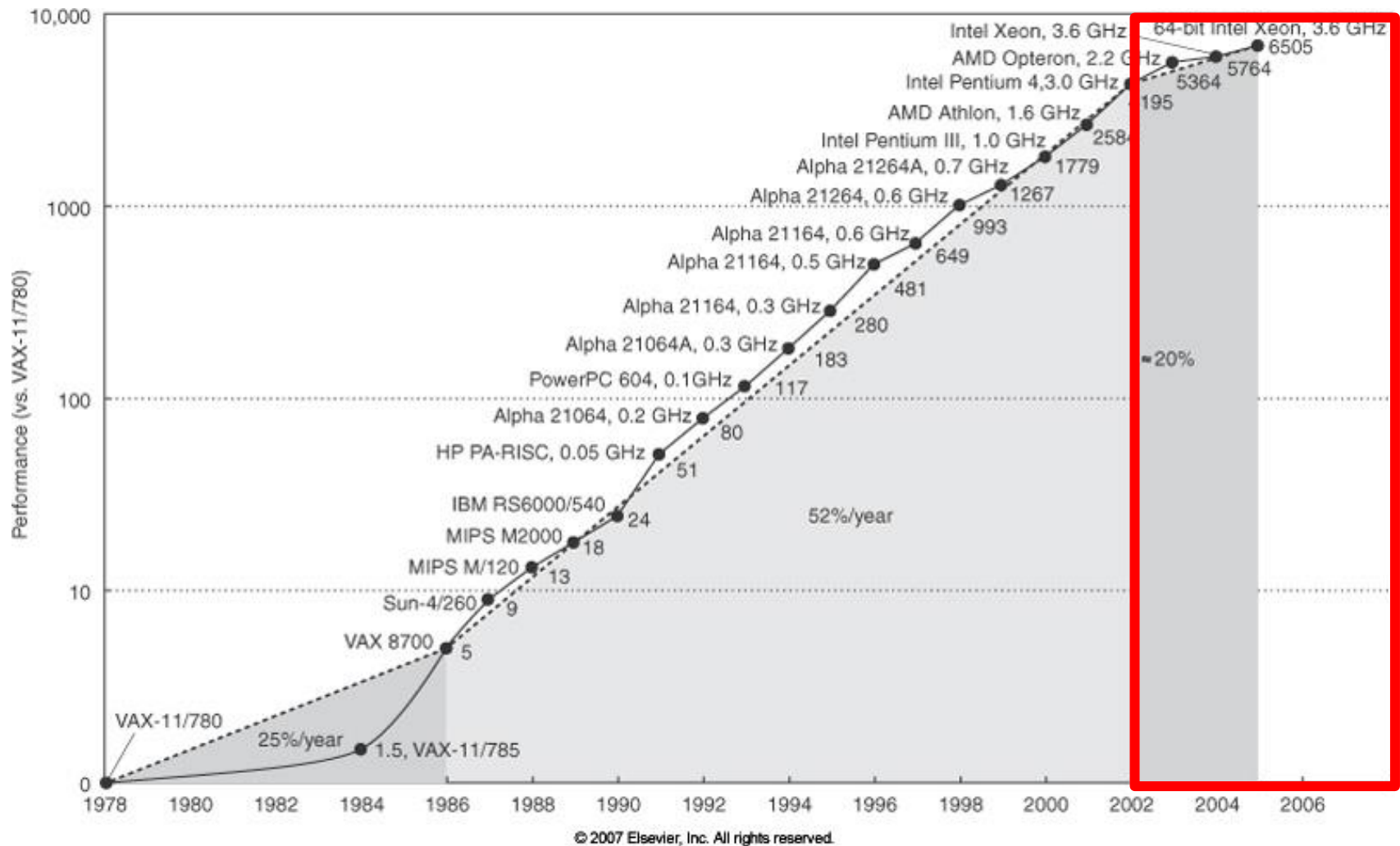
```
000000 00000 00101 0001000010000000
000000 00100 00010 00010000000100000
. . .
```



Moore's Law

- **Transistors**
 - Building block of integrated circuits
 - Electrical switch
- ***Moore's Law: Transistor count / chip : 40-55% increase per year***
 - In 1965, Intel's Gordon Moore predicted that the number of transistors that can be integrated on single chip would *double about every two years* (18~24 months)
- **Shrinking transistors: smaller, faster, and consuming less power at new generation technology**

Processor Performance



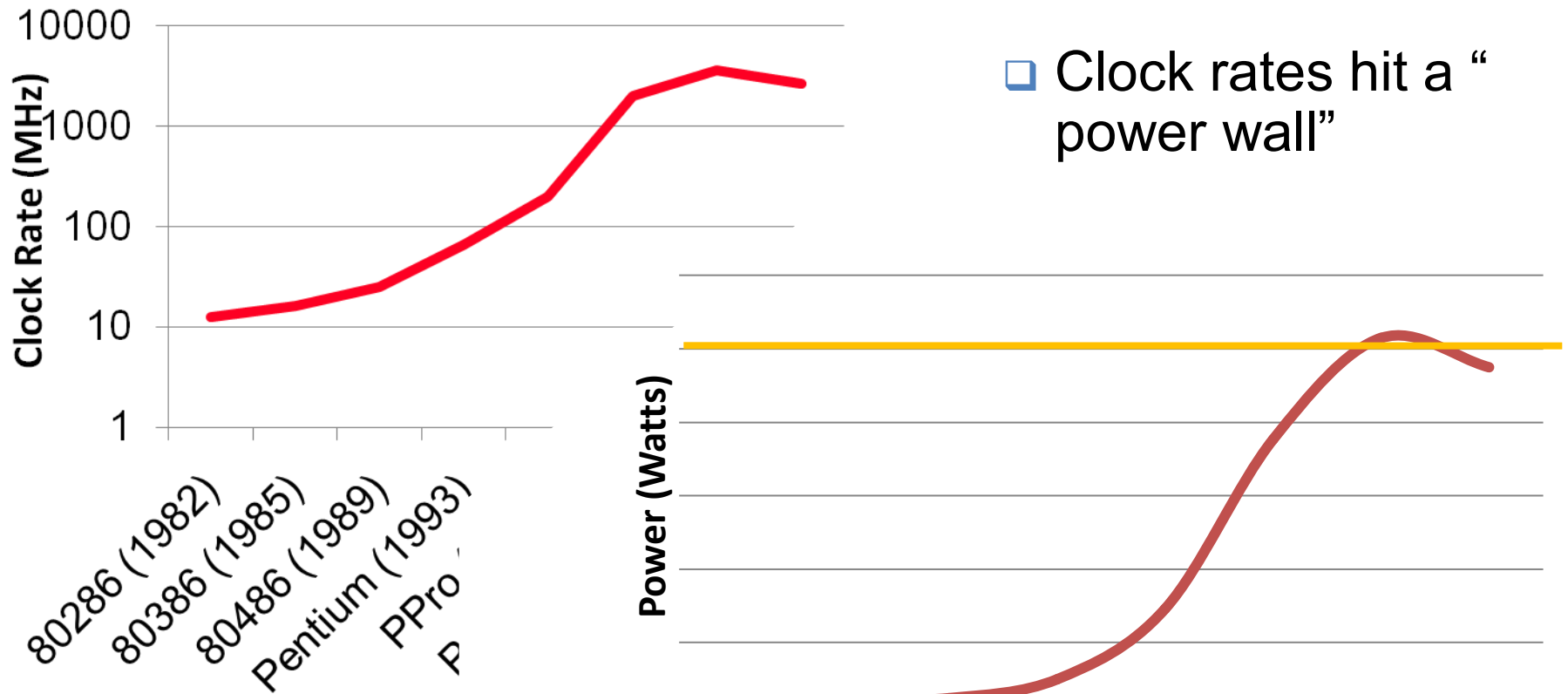
Era of 50%/Year Improvement

- **Performance improvement in the past decades**
 - Clock frequency: Faster and smaller transistors
 - Instruction-level parallelism (ILP)
 - Wide issue, out-of-order processor
 - Larger caches and better prefetch systems
- **Performance doubling every two years**

The End of 50%/year Era

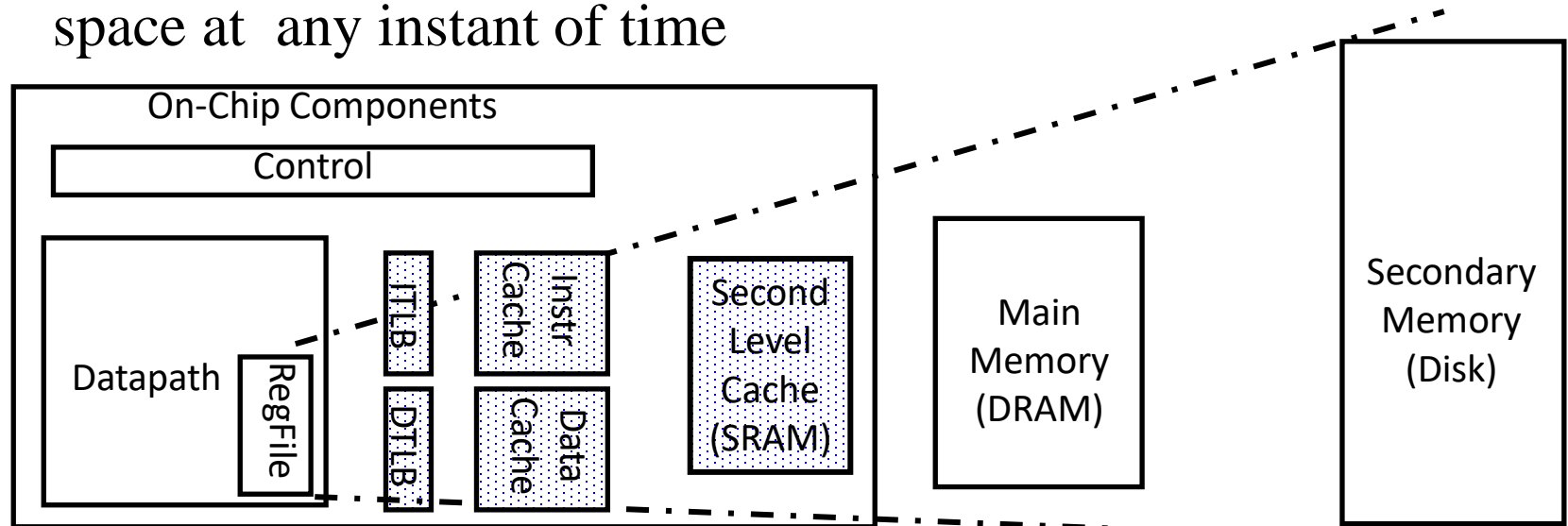
- Power (and heat) limits clock frequency
- Cannot keep increasing pipeline length
- Slow wires
- Diminishing returns of ILP features
 - 2x area to get 10-20% ILP improvement
- **Slowdown of single-core performance improvement**

But What Happened to Clock Rates and Why?



Locality

- Take advantage of the **principle of locality** to present the user with as much memory as is available in the *cheapest* technology at the speed offered by the *fastest* technology
- Programs access a relatively small portion of the address space at any instant of time



Speed (%cycles):	1/2's	1's	10's	100's	10,000's
Size (bytes):	100's	10K's	M's	G's	T's
Cost:	highest				lowest

Parallelism

- **HW systems are inherently parallel!**
 - Circuitry operates in parallel
 - CPU, memory, disk, and networks work in parallel
 - Pipelining, superscalar, vector operations, multi-cores
- **How to exploit parallelism has been one of the fundamental questions**
- **Throughout this course, you will learn how to exploit parallelism as a software engineer**

Performance

- **Tune applications to underlying hardware**
 - **Locality**
 - **Parallelism**
- **Architecture design based on performance**
 - **Study the behavior of applications**
 - **Determine the design based on performance, cost, and design complexity**
- **Our goal is to understand what factors in the architecture contribute to overall system performance and the relative importance (and cost) of these factors**