# Introduction to Computer Architecture

## Chapter 1


Krerk Piromsopa, Ph.D. @ 2016

# Introduction to Computer Architecture

- ★ Why should we take this class?
- ★ What is Computer Architecture?
- ★ Instruction Set Architecture
- ★ Computer Organization
- ★ Design Tradeoffs/Design Goals
- ★ Requirements and Technology Trends
- ★ Cost and Trends
- ★ How to design?

# Why should we take this class?

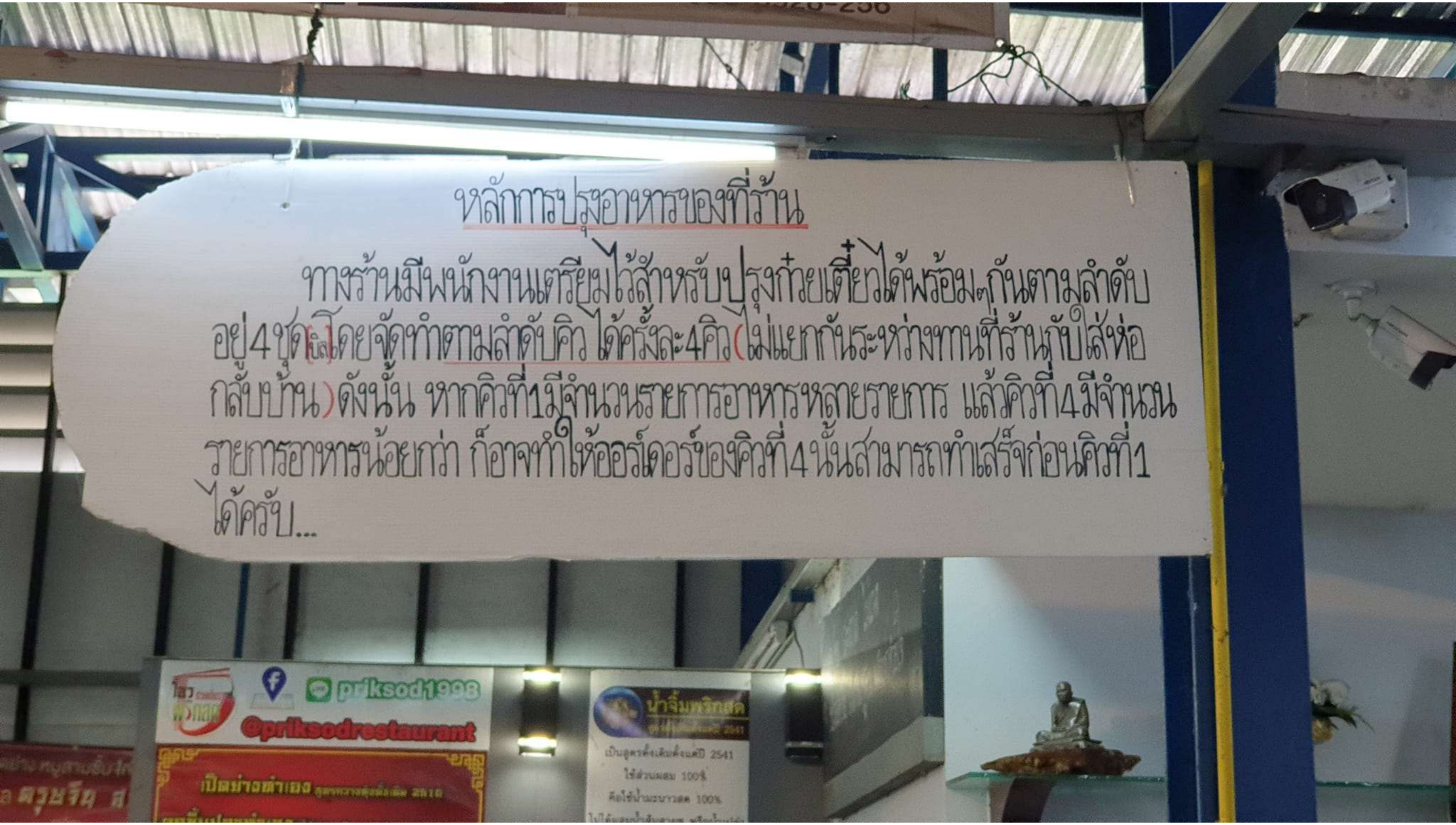
- ★ To design the next great architecture? …. Well … Maybe ….
  - ○ Architecture (especially desktop/server/laptop) has largely converged.
  - ○ Dictated by big name in the market
  - ○ How many of you will be an architect at the processor companies? (e.g. Intel, AMD, ARM)
- ★ Current Instruction set architecture (ISA) abstractions include enormous organizational innovation
  - ○ Differences in Intel Core I3, I5, and I7 are organizations (number of cores, Cache, clock/Turbo boost, Hyper-Threading).

# Why should we take this class? (ctd)

★ We are Computer Engineering.
Design, Analysis, Implementation concepts are vital to us.

★ To deal with systems design challenges.
(To be able to choose appropriate architecture.)

★ To be a better developer.


หลักการปรุงอาหารของที่ร้าน
ทางร้านมีพนักงานเตรียมไว้สำหรับปรุงก๋วยเตี๋ยวได้พร้อมๆกันตามลำดับ
อยู่4ชุด(บิล)โดยจัดทำตามลำดับคิว ได้ครั้งละ4คิว(ไม่แยกกันระหว่างทานที่ร้านกับใส่ห่อ
กลับบ้าน)ดังนั้น หากคิวที่1มีจำนวนรายการอาหารหลายรายการ แล้วคิวที่4มีจำนวน
รายการอาหารน้อยกว่า ก็อาจทำให้ออร์เดอร์ของคิวที่4นั้นสามารถทำเสร็จก่อนคิวที่1
ได้ครับ...
prksod1998
@priksodrestaurant
น้ำจิ้มพริกสด
เป็นสูตรดั้งเดิมตั้งแต่ปี 2541
ใช้ส่วนผสม 100%
คือใช้น้ำมะนาวสด 100%

# Why one program is faster than another?

- ★ Same algorithm and same programming language
- ★ Two programmers ( A and B)
- ★ Same machine
- ★ One can run faster.

Why?

- ★ Different data structure?
- ★ Different compiler?
- ★ Different optimization?
- ★ -------- Good knowledge of architecture means better program --------

# Which one is faster? Why?

Same program, different data structure

```
unsigned short a[2],b[2],c[2];
unsigned short i=0;
a[0]=10+i;
a[1]=20+i;
b[0]=15+i;
b[1]=25+i;
c[0]=a[0]+b[0];
c[1]=a[1]+b[1];
printf("c1 = %d, c2 = %d\n",c[0],c[1]);
```

56 LOC

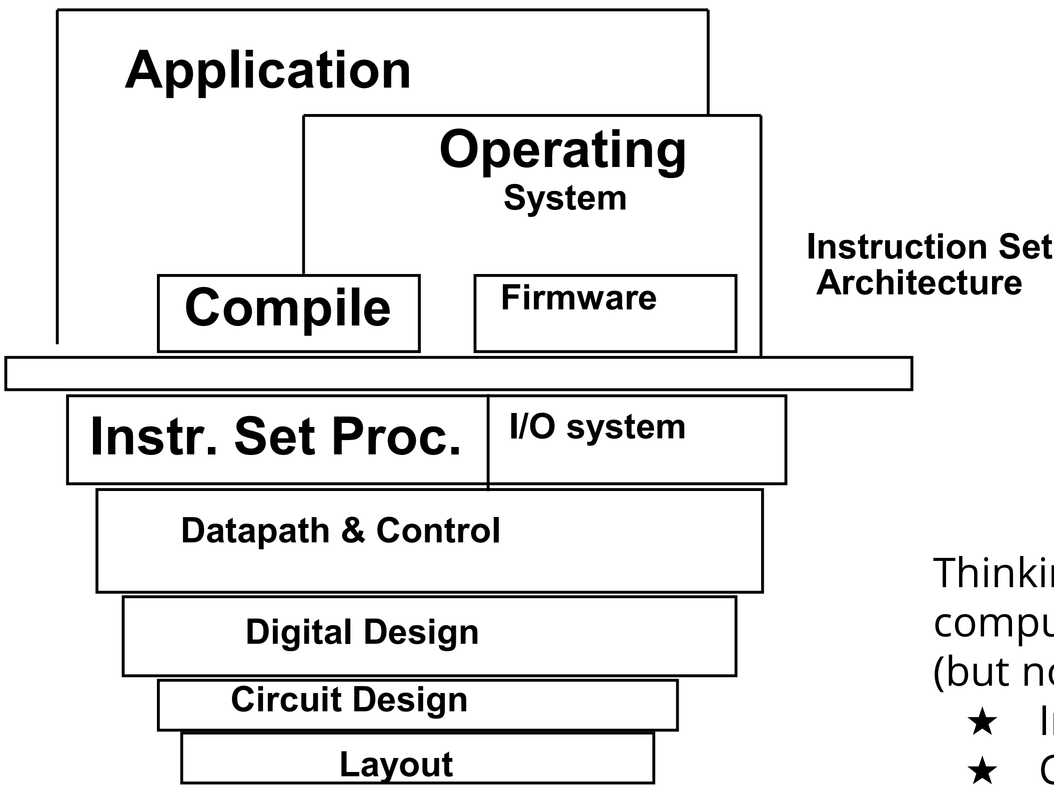
```
union i16 {
        unsigned int x;
        unsigned short h[2];
};

union i16 a;
union i16 b;
union i16 c;
unsigned short i=0;
a.h[0]=10+i;
a.h[1]=20+i;
b.h[0]=15+i;
b.h[1]=25+i;
c.x=a.x+b.x;
printf("c1 = %d, c2 = %d\n",c.h[0],c.h[1]);
```

52 LOC

Depending on version of compilers and operating systems, the result may vary.

# What is Computer Architecture?




- ★ Two words
  - ○ Computer
  - ○ Architecture - structure (usually building), concepts
- ★ Many levels of abstraction
- ★ Design, Measurement , and Evaluation

Thinking about a document explaining a computer Architecture. The book should include (but not limited to):

- ★ Instruction Set Architecture
- ★ Computer Organization
- ★ Application Binary Interface

# Instruction Set Architecture

- ★ Abstraction of machine for software
  - ○ Nowaday, we usually program in high-level programing languages (e.g. C). Compiler would translate to intermediate code for optimization. Eventually, the code will be translated to binary code.
    Programmers know nothing about underlying hardware.
- ★ Abstraction of software for hardware
  - ○ Legacy 8088 software still works on latest Intel core I7 architecture.

```
int average (int a, int b) {
    return (a+b)/2;
}
```

Compiler

Intermediate/Assembly Code

```
<average>:
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
   4:   89 7d fc                mov    %edi,-0x4(%rbp)
   7:   89 75 f8                mov    %esi,-0x8(%rbp)
   a:   8b 55 fc                mov    -0x4(%rbp),%edx
   d:   8b 45 f8                mov    -0x8(%rbp),%eax
  10:   01 d0                   add    %edx,%eax
  12:   89 c2                   mov    %eax,%edx
  14:   c1 ea 1f                shr    $0x1f,%edx
  17:   01 d0                   add    %edx,%eax
  19:   d1 f8                   sar    %eax
  1b:   5d                      pop    %rbp
  1c:   c3                      retq
```

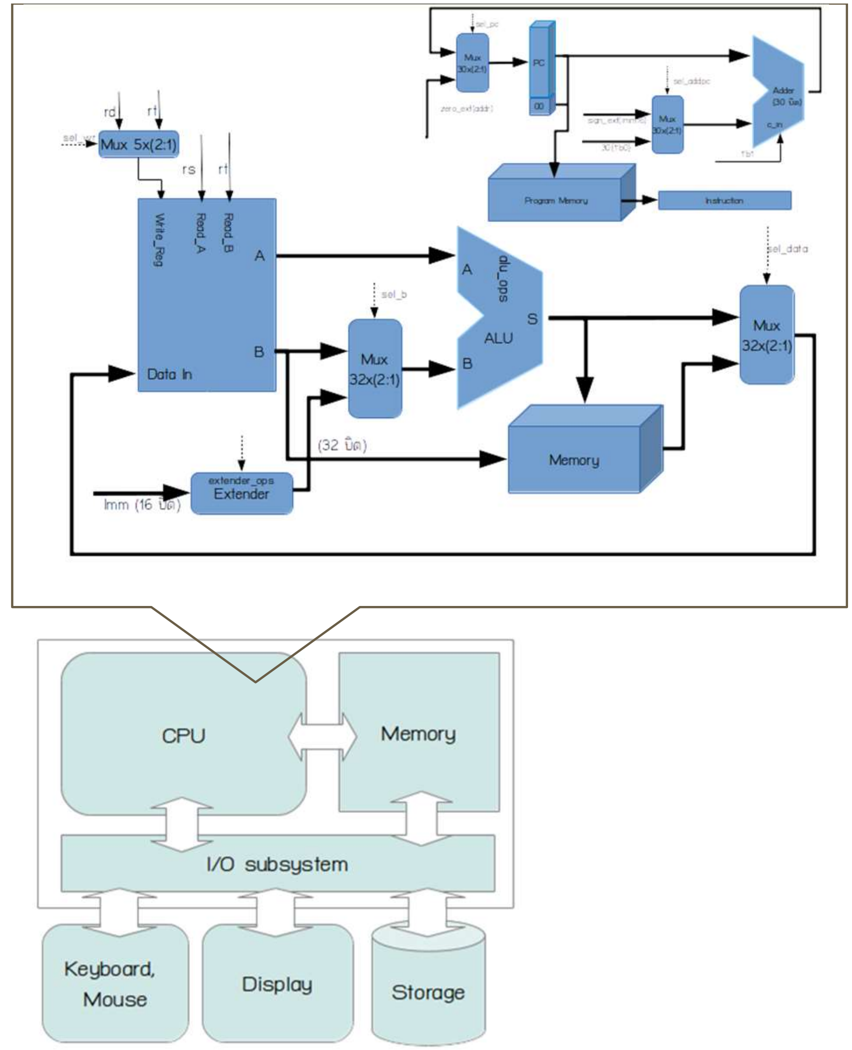
Assembler

Binary/Machine Code

```
5548 89e5 897d fc89 75f8 8b55 fc8b 45f8
01d0 89c2 c1ea 1f01 d0d1 f85d c3
```

# Computer Organization

- ★ ALU
- ★ Control Unit
- ★ Register
- ★ Bus
- ★ Memory subsystem
- ★ I/O subsystem
    - ○ Keyboard, Mouse
    - ○ Display
    - ○ Storage
    - ○ ...



# Why do we need temporary storage (register)?

★ How to do 2 + 3 in grade school?

★ Remember/Put 2 in your mind (temporary storage)
★ Put up 3 fingers
★ Count 3, 4, 5
★ The answer is 5.

Does having more registers beneficial for the calculation?

# Design Tradeoffs/Design Goals

- Performance ? at any cost ?
- Power ?
- Design Tradeoffs:
    - Cost vs. Performance (vs. Power)
    - Need for modeling and measurement

- Goals
    - Functional requirements
        - Market & application driven
    - Performance Goals
    - Cost constraints
    - Power constraints
    - … (you name it) …
- Involvements
    - ISA (software interface)
    - Organization (CPU internals, memory, buses,  ….)
    - Hardware - logic design, packaging, …

# Requirements and Technology Trends

- ★ Requirements
  - ○ Application area
    - ■ General purpose
    - ■ Scientific
    - ■ Commercial
    - ■ Multimedia
  - ○ Operating system requirements
    - ■ Memory Management
    - ■ Security, Protection
    - ■ Context switching
    - ■ Interrupts
  - ○ Standards
    - ■ IEEE floating-point ?
    - ■ Bus, I/O (PCI, USB, …)
    - ■ Support for programming languages
- ★ Time to market ?
- ★ Cost?

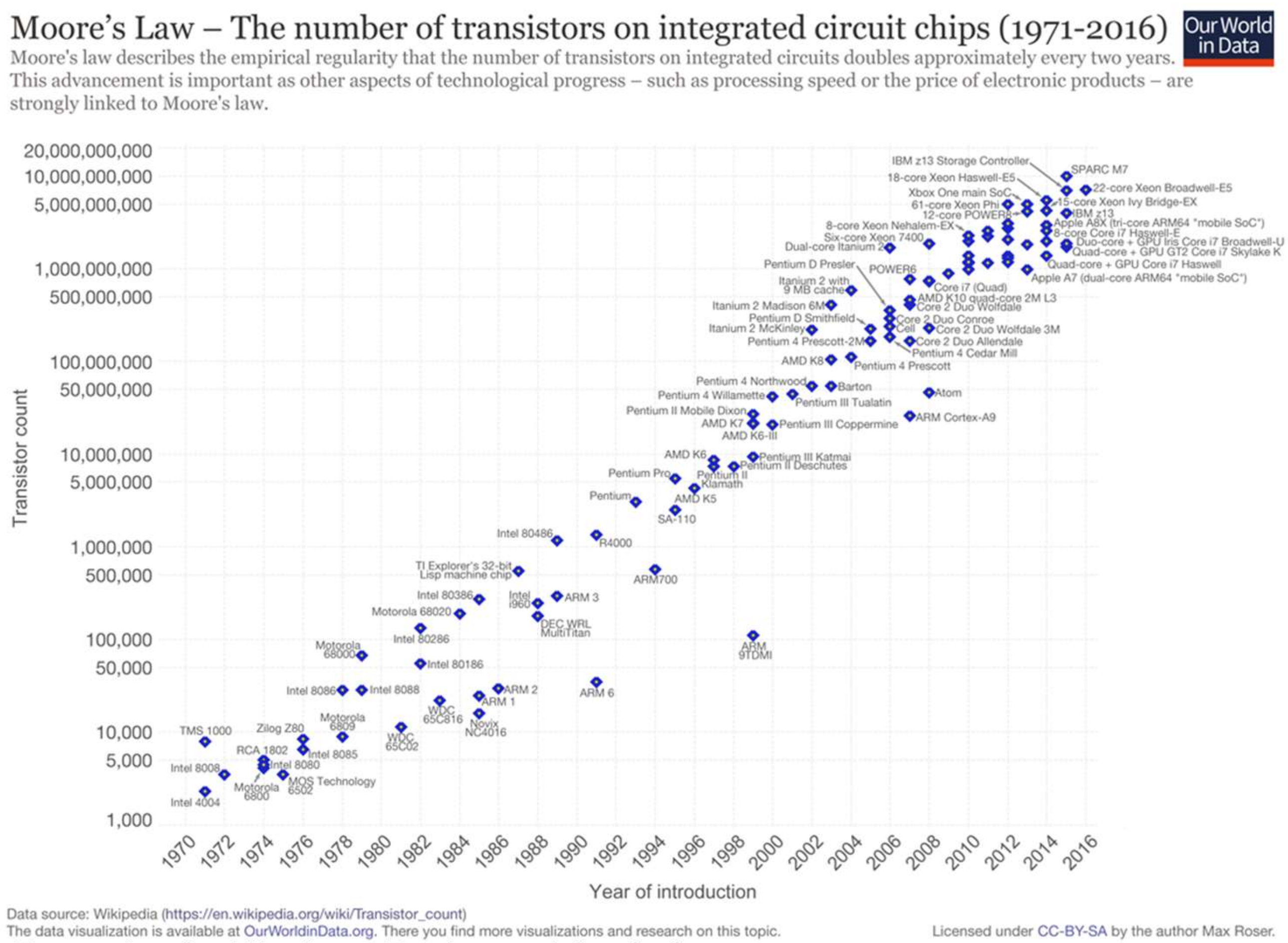
- ★ Software Trends
  - ○ Require more memory (2x per year)
    - ■ 1 bit of address
  - ○ Use of compilers
    - ■ (ISA for compiler, not programmer)
    - ■ Optimization
    - ■ Scheduling
- ★ Hardware Trends
  - ○ IC/Transistor technology - density, size, performance
  - ○ DRAM - capacity 4x per 3 years, slow performance improvement
  - ○ Disk - capacity 4x per 3 years

# Tradeoffs

- ★ Power


- ★ Performance

# Moore's Law




- ★ An observation by Gordon Moore (the co-founder of Fairchild Semiconductor and CEO of Intel)
- ★ the number of transistors in a dense integrated circuit doubles about every two years (18 months)

Images and charts are taken from wikipedia.

# Cost and Trends (DRAM)

- DRAM cost drops 40% per year
- Commoditization lowers the cost

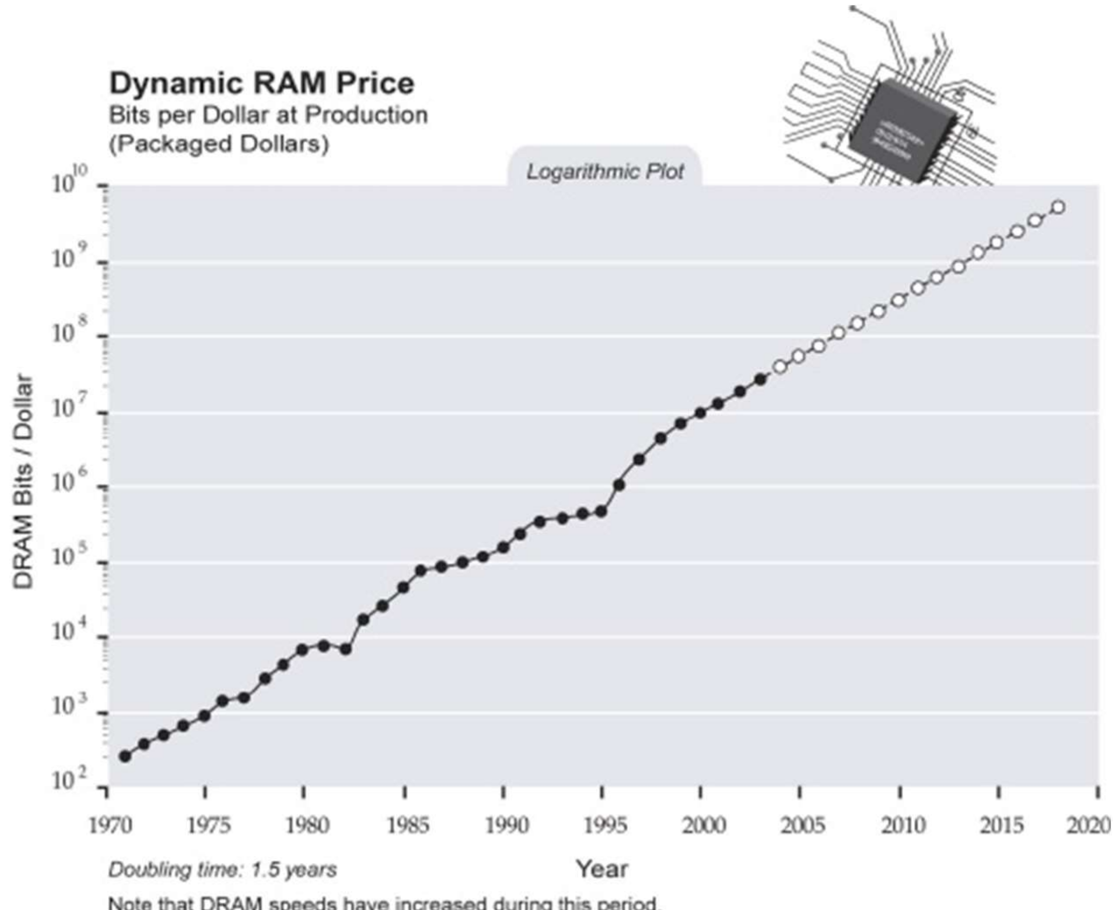


Chart is taken from http://www.singularity.com/charts/page58.html

# Cost and Trends (Watts per MIPS)

- ★ Watts per MIPS (Millions Instructions Per Second) is getting better

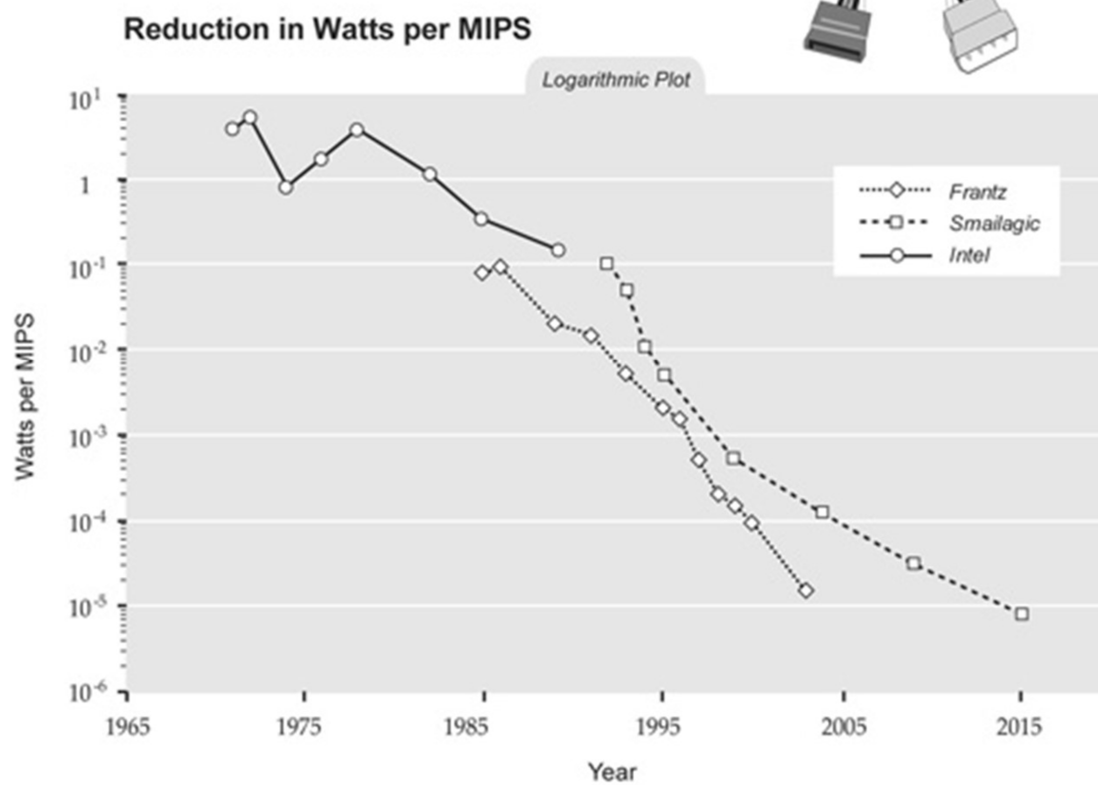


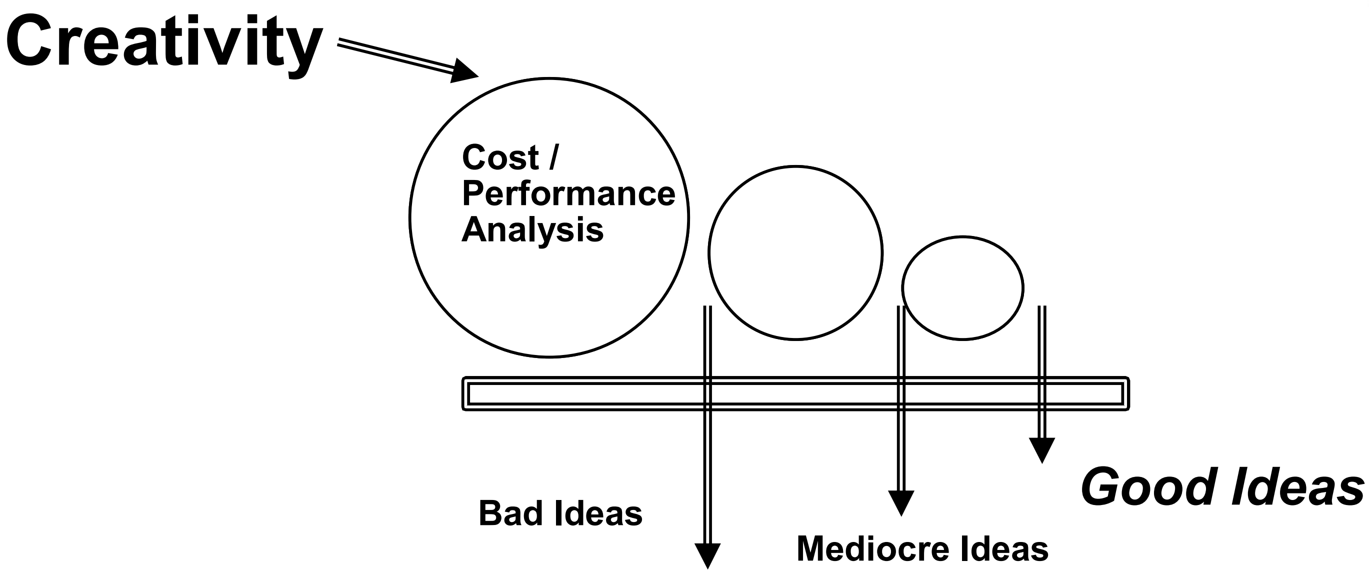

Chart is taken from http://www.singularity.com/charts/page129.html

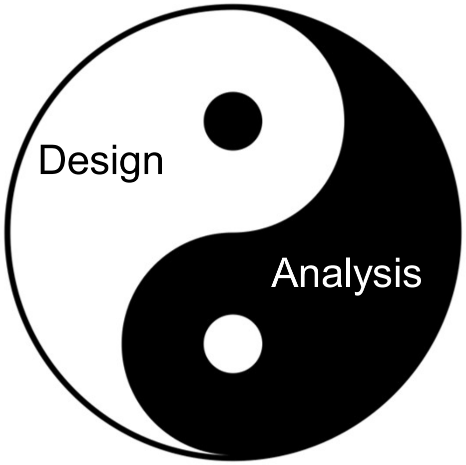
# Cost of Components

- ★ Price of a notebook computer
  - 6% for Case
  - 25% for Circuit/Processor
  - 10% for RAM
  - 5% for video system
  - 5% for I/O, PCB
  - 4% for keyboard, mouse
  - 20% for monitor
  - 20% for storage
  - 25% for battery
- ★ 20%+ to 30%+ margin

# How to design

★ Measurement and evaluation
★ Iterative process through possible designs



# End of Chapter 1