



中国科学技术大学  
University of Science and Technology of China

# 1. 计算机系统概论

## 王 超

中国科学技术大学计算机学院  
嵌入式系统实验室  
2019年春

# 本章提纲



中国科学技术大学  
University of Science and Technology of China

## 口计算机系统概述

- ✓ 计算机系统简介
- ✓ 计算机系统的层次结构

## 口计算机的基本组成

- ✓ 计算机硬件组成
- ✓ 计算机组装与体系结构
- ✓ 计算机软件组成

## 口计算机硬件性能指标

- ✓ 机器字长
- ✓ 存储容量
- ✓ 运行速度

## 口计算机/CPU发展历史

- ✓ 计算机的分代
- ✓ 计算机设计面临的挑战



嵌入式系统实验室  
EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC

# 1.1 计算机系统简介



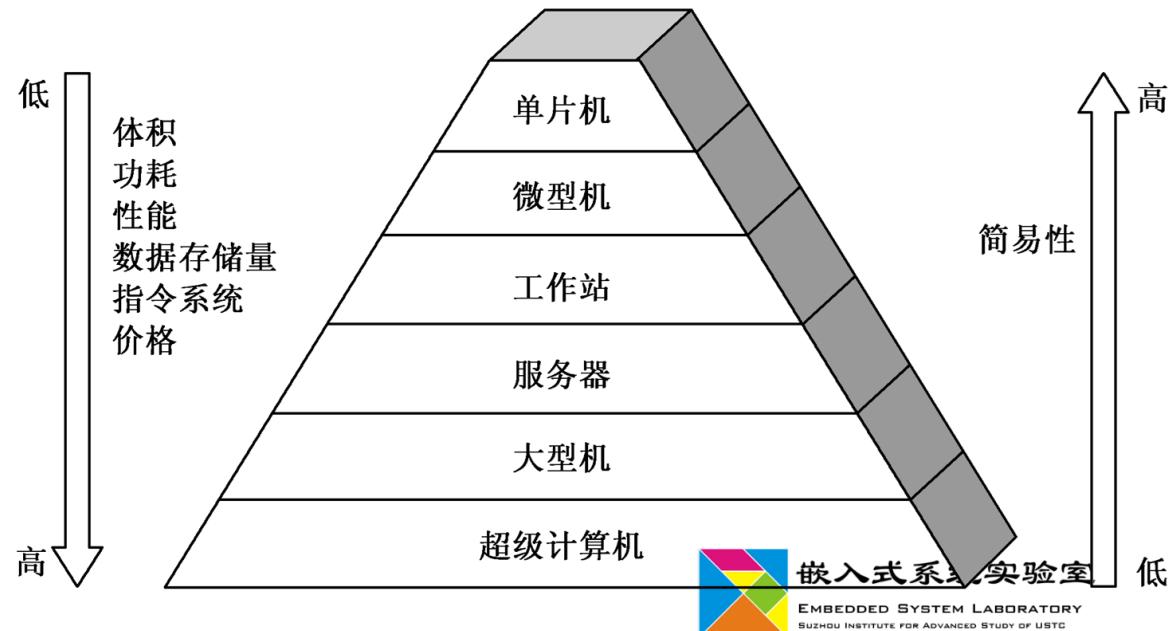
中国科学技术大学  
University of Science and Technology of China

□ 计算机的分类 按性能和灵活性划分  
(数字) 计算机

专用计算机 嵌入式系统

通用计算机 普通PC机

□ 通用计算机类型





103型计算机(即DJS-1型, 1958)

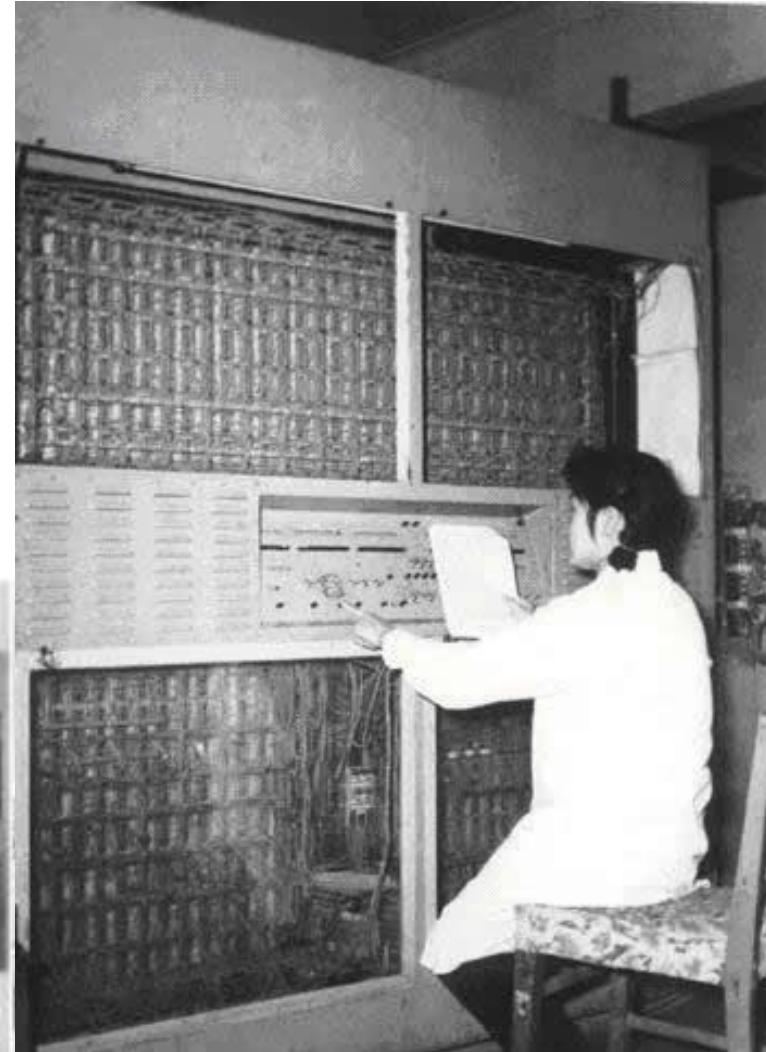
## □中国制造的第一台 计算机

- ✓ 运行速度1500次/s
- ✓ 内存容量1KB

## □1960年第一台通用 计算机-107机



夏培肃院士



# 2017年11月全球超级计算机TOP500



中国科学技术大学  
University of Science and Technology of China

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	<b>Sunway TaihuLight</b> - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
2	<b>Tianhe-2 (MilkyWay-2)</b> - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P , NUDT National Super Computer Center in Guangzhou China	3,120,000	33,862.7	54,902.4	17,808
3	<b>Piz Daint</b> - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	361,760	19,590.0	25,326.3	2,272
4	<b>Gyoukou</b> - ZettaScaler-2.2 HPC system, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 700Mhz , ExaScaler Japan Agency for Marine-Earth Science and Technology Japan	19,860,000	19,135.8	28,192.0	1,350
5	<b>Titan</b> - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x , Cray Inc. DOE/SC/Oak Ridge National Laboratory United States	560,640	17,590.0	27,112.5	8,209

# 2018年11月全球超级计算机TOP500



中国科学技术大学  
University of Science and Technology of China

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	DOE/SC/Oak Ridge National Laboratory United States	<b>Summit</b> - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM	2,397,824	143,500.0	200,794.9	9,783
2	DOE/NNSA/LLNL United States	<b>Sierra</b> - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM / NVIDIA / Mellanox	1,572,480	94,640.0	125,712.0	7,438
3	National Supercomputing Center in Wuxi China	<b>Sunway TaihuLight</b> - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
4	National Super Computer Center in Guangzhou China	<b>Tianhe-2A</b> - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 NUDT	4,981,760	61,444.5	100,678.7	18,482
5	Swiss National Supercomputing Centre (CSCS) Switzerland	<b>Piz Daint</b> - Cray XC50, Xeon E5- 2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	387,872	21,230.0	27,154.3	2,384

实验室



# 天河一号A (2010年)

- CPU+GPU异构体系：实测运算能力倍增至2507万亿次
- “自主可控”原则：
  - ✓ 2048颗我国自主研发的飞腾FT-1000八核处理器
    - 基于SUN公司UltraSparc T2处理器
  - ✓ 14336颗Intel Xeon (至强) X5670 2.93GHz六核处理器
  - ✓ 7168块NVIDIA Tesla M2050高性能计算卡 (\$2500/块)

## 中国研制成功每秒运算逾千万亿次超级计算机

天河一号



# 神威·太湖之光 (2016)



中国科学技术大学  
University of Science and Technology of China

□ 2016年6月，Sunway Taihu Light超算服务器在Lin基准测试中，以93 petaflop/s（注：petaflop：每秒执行1千万亿次浮点运算）的性能指标成为TOP500超算计算机列表的第一名



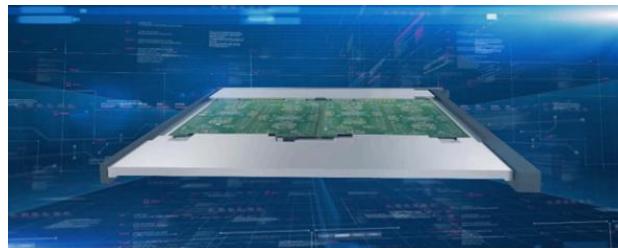
单节点260核



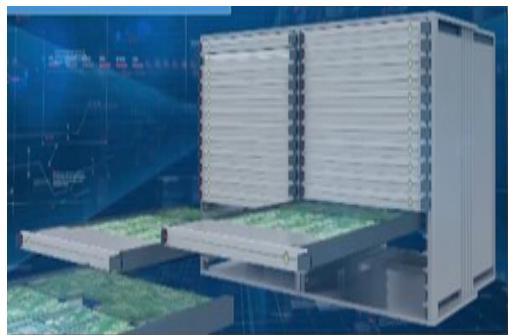
1个卡上2个节点

Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway

Site:	National Supercomputing Center in Wuxi
Manufacturer:	NRCPG
Cores:	10,649,600
Linpack Performance (Rmax)	93,014.6 TFlop/s
Theoretical Peak (Rpeak)	125,436 TFlop/s
Nmax	12,288,000
Power:	15,371.00 kW
Memory:	1,310,720 GB
Processor:	Sunway SW26010 260C 1.45GHz
Interconnect:	Sunway
Operating System:	Sunway RaiseOS 2.0.5



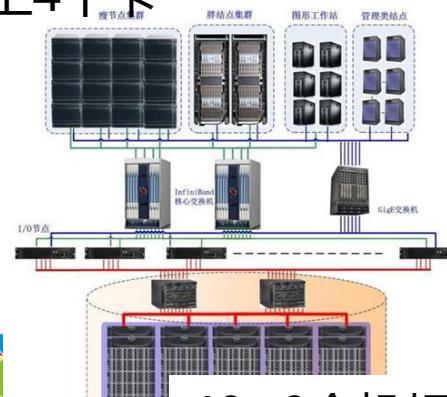
一个板上4个卡



一个超级节点32个板 (256节点)



一个机柜4个超级节点 (1024节点)



40+8个机柜

# 神威太湖之光

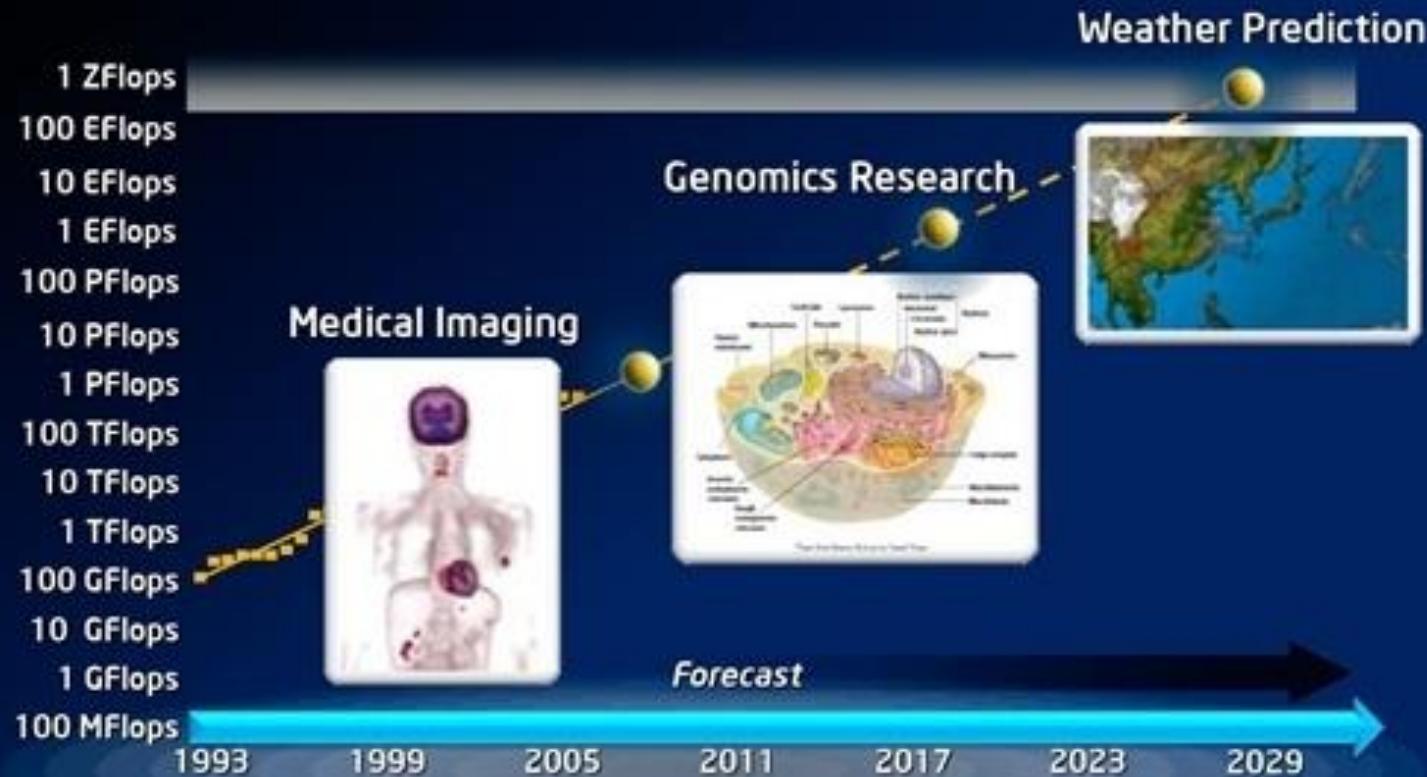


中国科学技术大学  
University of Science and Technology of China





## An Insatiable Need For Computing



*Exascale Problems Cannot Be Solved Using the  
Computing Power Available Today*

# 1.1 计算机系统简介 (2)



中国科学技术大学  
University of Science and Technology of China

## □计算机系统

- ✓ 一个由“硬件 + 软件”组成的复杂的自动化电子设备
- ✓ 包含硬件系统和软件系统两大部分
  - 硬件：计算机的实体部分，由看得见摸得着的各种电子元器件，各类光、电、机设备的实物组成
  - 软件：由人们事先编制的具有各类特殊功能的程序组成，看不见摸不着，一般通过各类媒介（如光盘、磁盘等）存放

## □硬件与软件的关系

- ✓ 硬件是躯体，是物质基础
- ✓ 软件是灵魂，是硬件功能的完善和补充

计算机性能的好坏取决于“软”、“硬”件功能的总和



## □ 计算机硬件和软件的逻辑等价

- ✓ 计算机中，许多功能既可以直接由硬件实现，也可以在基本硬件的支持下利用软件来实现
  - 如乘法运算，可以使用硬件乘法电路实现，也可以用乘法子程序实现
- ✓ 硬件和软件间的功能分配，随技术发展而变化，没有固定的模式
  - 取决于设计目标、系统性价比以及当时的技术水平
  - 早期，“硬件软化”，降低计算机造价
  - 集成电路技术的发展，“软件硬化”
- ✓ 软件固化：将软件的一些功能固化地保存在只读存储器中，称为**固件**
  - 功能上是软件，形态上是硬件
  - 一些系统软件的核心部分（如操作系统内核、常用软件的固定部分）常被固化在存储芯片中

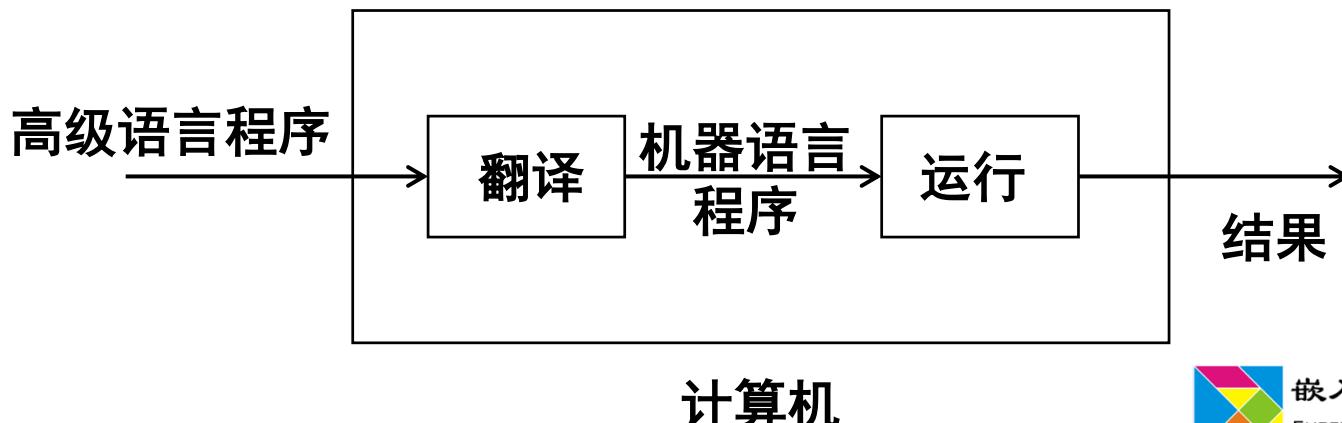
# 1.2 计算机系统的层次结构



中国科学技术大学  
University of Science and Technology of China

## 口现代计算机的解题处理过程

- ✓ 首先，用户用高级语言编写程序，描述问题的求解过程
- ✓ 然后，将程序与数据一起送入计算机中，由计算机将其翻译成机器语言程序
- ✓ 最后，计算机自动运行机器语言程序，输出计算结果



# 1.2 计算机系统的层次结构



中国科学技术大学  
University of Science and Technology of China

应用语言

虚拟机器  $M_4$

XML等，为满足某种用途而专门设计——应用程序包

高级语言

虚拟机器  $M_3$

C/C++/Java等，对问题描述接近人类习惯，较强的通用性，翻译成汇编语言——编译器或解释器

汇编语言

虚拟机器  $M_2$

符号式语言，不具有通用性，与机器指令系统相关——汇编器

操作系统

虚拟机器  $M$

提供汇编语言/高级语言使用与实现中所需的基本操作，管理计算机软硬件资源，方便使用

机器语言

传统机器  $M_1$

二进制代码（0、1串），直接在物理机器上执行

微指令系统

微程序机器  $M_0$

将每条机器指令翻译成一组微指令，构成一个微程序系统实验室



## 1.2 计算机系统的层次结构

软件

硬件



# 本章提纲



中国科学技术大学  
University of Science and Technology of China

## 口计算机系统概述

- ✓ 计算机系统简介
- ✓ 计算机系统的层次结构

## 口计算机的基本组成

- ✓ 计算机硬件组成
- ✓ 计算机组装与体系结构
- ✓ 计算机软件组成

## 口计算机硬件性能指标

- ✓ 机器字长
- ✓ 存储容量
- ✓ 运行速度

## 口计算机/CPU发展历史

- ✓ 计算机的分代
- ✓ 计算机设计面临的挑战



嵌入式系统实验室  
EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC

## 2.1 计算机的硬件组成



□从计算  $y = ax + b - c$  开始

行数	解题步骤与数据	说明
1	取数 (9)	(9) 表示第9行的数a, 下同
2	乘法 (12)	完成ax
3	加法 (10)	完成ax+b
4	减法 (11)	完成y=ax+b-c
5	存数 y → 13	结果y的值记录到第13行
6	输出	把y的值输出
7	停止	运算完毕, 停止
8		
9	a	数据
10	b	数据
11	c	数据
12	x	数据
13	y	数据

## 2.1 计算机的硬件组成 (2)



中国科学技术大学  
University of Science and Technology of China

### □ 解题过程的解释

- ✓ 把原始数据及解题步骤记录在纸上，即在纸上“存储”了解题的原始信息，纸作为具有记忆功能的部件——**存储器**
- ✓ 对数据进行了加、减、乘等算术运算，人脑作为具有计算功能的部件——**运算器**
- ✓ 用笔将解题信息写在纸上或将运算结果写出来——**输入/输出设备**
- ✓ 整个过程在人脑的控制下有序进行——**控制器**

## 2.1 计算机的硬件组成 (3)



中国科学技术大学  
University of Science and Technology of China

### □计算机硬件的五大组成部件

- ✓ 运算器
  - 完成算术运算和逻辑运算，暂存中间结果
- ✓ 存储器
  - 存放数据和程序
- ✓ 控制器
  - 控制、指挥程序和数据的输入、运行以及处理运算结果
- ✓ 输入设备
  - 将人们熟悉的信息形式转换成机器能识别的形式
- ✓ 输出设备
  - 将机器运算结果转换为人们熟悉的信息形式展示出来



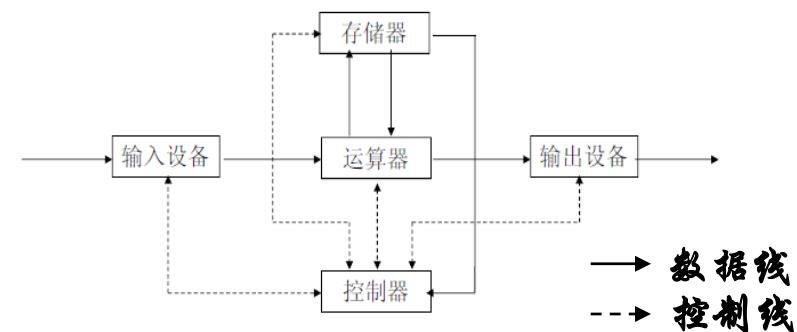
## 冯·诺依曼机

✓ 1945年，冯·诺依曼在研究EDVAC时提出“存储程序”概念，以此概念为基础研制的计算机统称为冯·诺依曼机



✓ 特点：

- 五大组成部件，以运算器为中心
- 数据和指令用二进制数表示，以同等地位存放于存储器中，按地址访问
- 指令由操作码和地址码组成，在存储器中按顺序存放





□ 冯·诺依曼结构也叫做普林斯顿结构

□ 哈佛结构

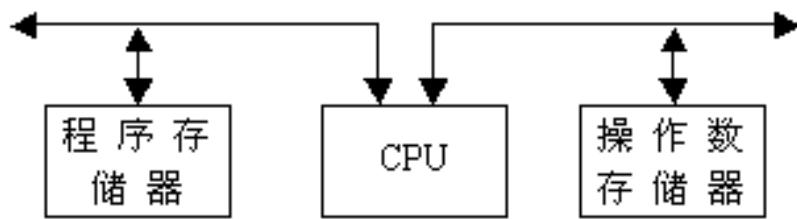
- ✓ 将指令和数据分开存储的结构

程序总线

数据总线

□ 两者的主要区别

- ✓ 指令和数据是否分别存储
- ✓ 是否使用两条独立的总线，分别作为CPU与每个存储器间的专用通信通道，两条总线间毫无关联
- ✓ 哈佛结构目前较多出现在嵌入式应用中



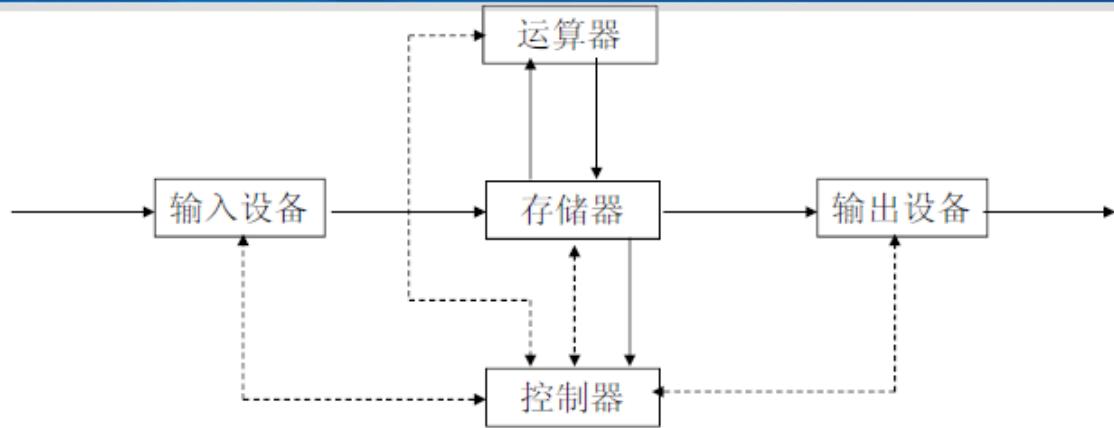
嵌入式系统实验室

EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC

# 现代计算机的组成结构

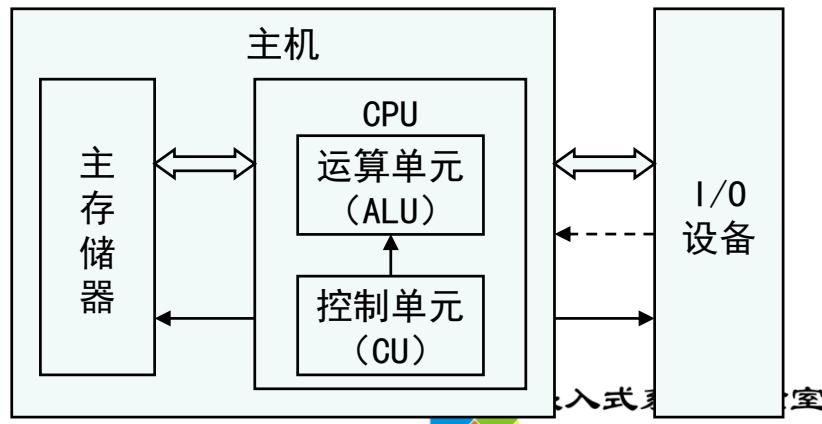


以存储器为中心



- ✓ 运算器和控制器在逻辑关系和电路结构上联系紧密，将其集成在同一芯片上，称为**中央处理器CPU**
- ✓ 把输入设备与输出设备简称I/O设备
- ✓ 现代计算机三大组成部件：

CPU ————— 主机  
主存储器 ————— 外部设备  
I/O设备 ————— 外部设备





## □ 运算器

- ✓ 常称为算术逻辑运算部件ALU
- ✓ 运算数据格式
  - 二进制数：0，低电压或无脉冲；1，高电压或有脉冲
  - 长度一般都为2的幂数
- ✓ 运算器最少包括3个寄存器
  - ACC，累加器
  - MQ，乘商寄存器
  - X，操作数寄存器
- ✓ 数据传送
  - 将运算结果从ACC送至存储器中的MDR
  - 存储器的数据从MDR送至ACC、MQ或X

操作数 寄存器	运算	加法	减法	乘法	除法
ACC		被加数及和	被减数及和	乘积高位	被除数及余数
MQ				乘数及乘积低位	商
X		加数	减数	被乘数	除数

# 运算器的四则运算过程



中国科学技术大学  
University of Science and Technology of China

## 口表示规定

- ✓ M表示存储器地址号，[M]表示其中内容值；X/MQ/ACC表示寄存器，[X]/[MQ]/[ACC]表示对应的内容值
- ✓ 假设ACC中已存有前一时刻的运算结果，并作为下一个运算的操作数

### ➤ 加法

$[M] \rightarrow X$

$[ACC] + [X] \rightarrow ACC$

### ➤ 乘法

$[M] \rightarrow MQ$

$[ACC] \rightarrow X$

$0 \rightarrow ACC$

$[X] \times [MQ] \rightarrow ACC//MQ$

操作数 寄存器	运算	加法	减法	乘法	除法
ACC		被加数及和	被减数及和	乘积高位	被除数及余数
MQ				乘数及乘积低位	商
X		加数	减数	被乘数	除数

### ➤ 减法

$[M] \rightarrow X$

$[ACC] - [X] \rightarrow ACC$

### ➤ 除法

$[M] \rightarrow X$

$[ACC] \div [X] \rightarrow MQ$

余数R在ACC中



嵌入式系统实验室  
EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC



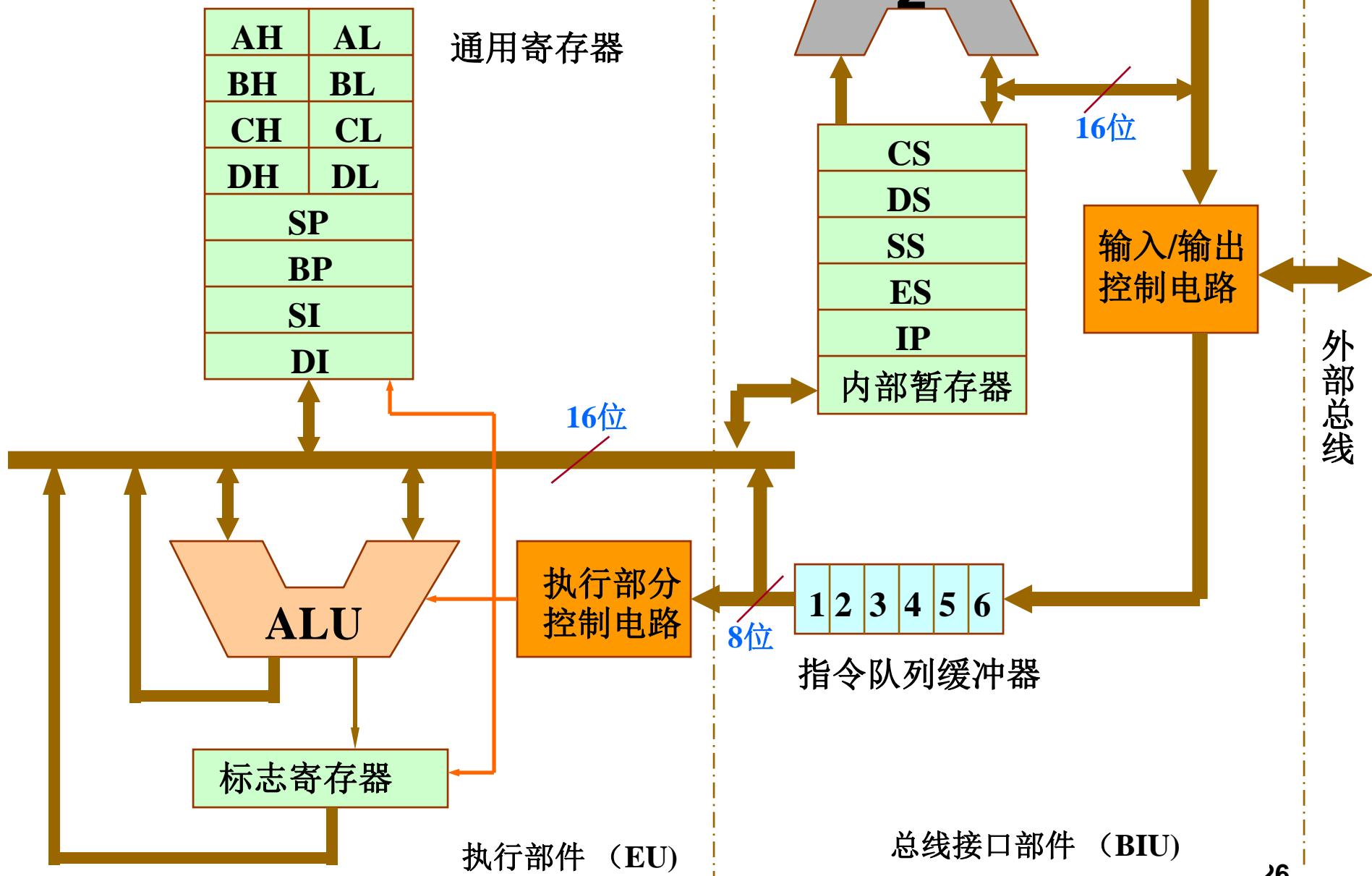
## □ 控制器

- ✓ 计算机的神经中枢，控制计算机各部件有条不紊地工作
- ✓ 控制器的基本任务：周而复始地完成下列三个过程：
  - 按照计算机程序编排好的指令序列，从存储器中取出**一条指令**——**取指过程**
  - 对指令进行分析，确定要完成的**操作（操作码）**，并按寻址特征找出**操作数**的地址——**分析过程**
  - 根据**操作数地址**及指令的**操作码**，完成相应的操作——**执行过程**

### ✓ 控制器的组成

- **程序计数器PC**：用于存放当前将要执行的指令，具有**自动加1**的功能，能够自动形成下一条指令的地址
- **指令寄存器IR**：存放CPU当前正在执行的指令内容
- **控制单元CU**：分析当前指令所要完成的操作，并发出各种微操作命令序列

# 8086内部结构





# CPU功能与指令的执行过程

## □ 功能

- ✓ 算术逻辑运算
- ✓ 指令译码、执行
- ✓ 数据暂存
- ✓ 与MEM、I/O交换数据
- ✓ 提供整个系统的定时和控制
- ✓ 响应中断请求

## □ 过程

### □ 取指

- 根据PC访存读取当前要执行的指令； PC+1

### □ 译码

- 识别指令字中的操作类型，产生相应的控制信号

- 停机指令？

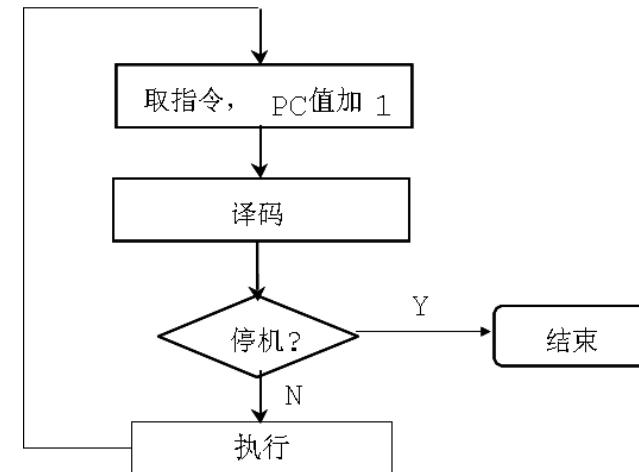
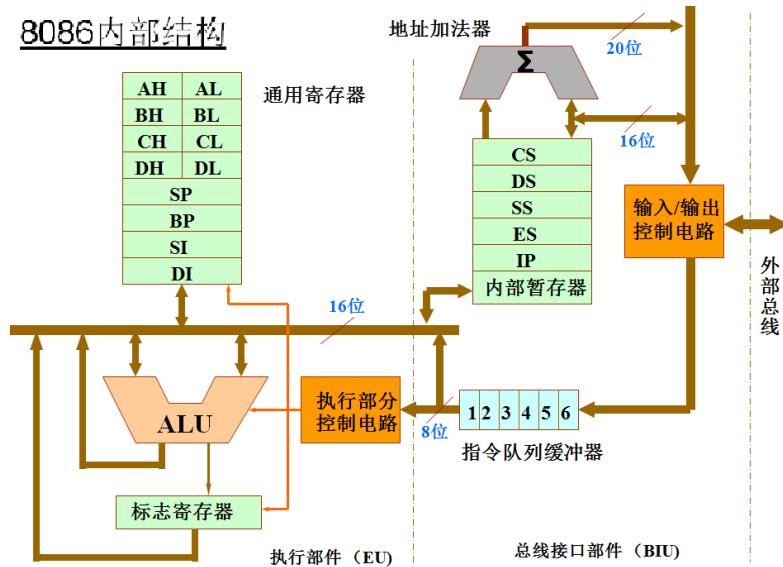
### □ 取操作数

- 根据指令字的地址域访存

### □ 执行

### □ 写回

8086内部结构





## 口指令—计算机实现某种操作（控制或运算）的一条代码

- ✓ 由**操作码**和**操作数地址**组成，以**二进制码**的形式顺序存放在存储器中
- ✓ 解决某个问题的一串指令的序列，形成该问题的**程序**
- ✓ “**程序控制**”的概念
  - 控制器依据存储的程序来控制全机协调地完成计算任务

### ➤ 指令系统

- 一台计算机通常有几十种基本指令，它们构成了该计算机的指令系统
- 指令系统是衡量计算机性能的重要标志

指令	操作码
加法	001
减法	010
乘法	011
除法	100
取数	101
存数	110
打印	111
停机	000





# 示例—指令

## 口指令格式

- ✓ 指令字长16位：操作码6位，地址码10位
- ✓ 例：000001 0000001000

## 口指令操作码

- ✓ load (000001) : [M]→ACC
- ✓ store (000010) : ACC→M
- ✓ add (000011) : ACC+[M]→ACC
- ✓ mul (000100) : ACC\*[M]→ACC
- ✓ print (000101) : 打印[M]
- ✓ hlt (000110) : 停机



# 示例— $y=ax^2+bx+c$

□  $y=(ax+b)x+c$

- ✓  $x \rightarrow ACC$
- ✓  $x * a \rightarrow ACC$
- ✓  $ax + b \rightarrow ACC$
- ✓  $x * (ax + b) \rightarrow ACC$
- ✓  $(ax + b)x + c \rightarrow ACC$
- ✓  $ACC \rightarrow y$
- ✓ Print(y)
- ✓ Hit

地址	op	opr
0000	000001(Id)	0000001000
0001	000100(mul)	0000001001
0010	000011(add)	0000001010
0011	000100(mul)	0000001000
0100	000011	0000001011
0101	000010	0000001100
0110	000101	0000001100
0111	000110	
1000	x	
1001	a	
1010	b	
1011	c	
1100	y	



## □ 存储器的功能

- ✓ 保存或“记忆”解决问题所需的数据和解题方法步骤

## □ 存储器的组成

- ✓ 以主存储器为例，包括存储体、各种逻辑部件及控制电路等
- ✓ 存储体 ← 存储单元 ← 存储元件
  - 存储元件，也称存储元，每个能存储一位二进制代码0或1
  - 存储单元，可存储一串二进制数，称为一个存储字（字节），其位数称为**存储字长**。
  - 存储字代表的二进制码，可以是数值、字符等，也可以表示指令
- ✓ 存储地址
  - 对每个存储单元进行编号，其编号称为该存储单元的地址



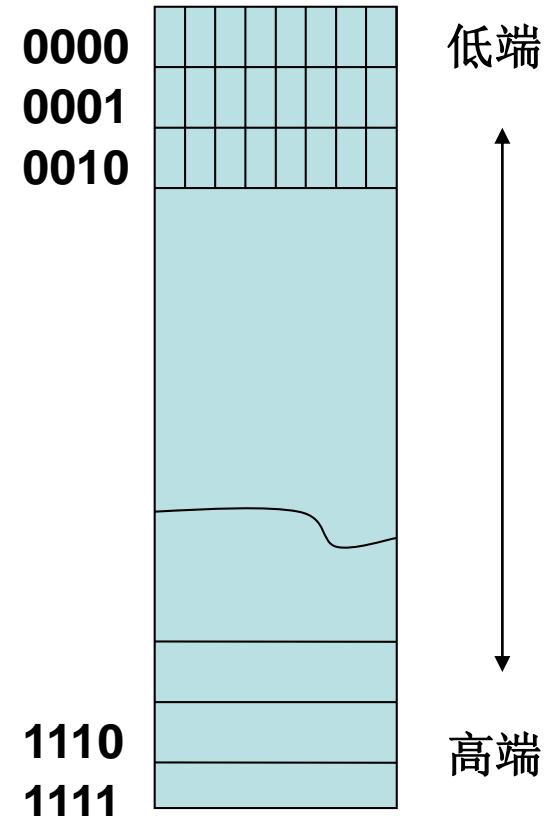
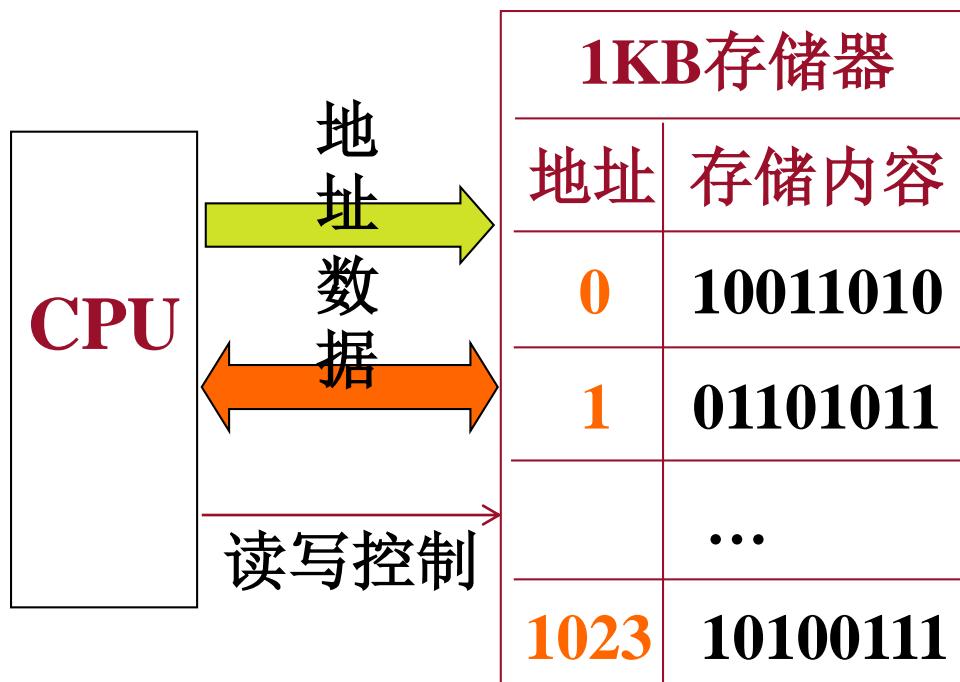
# 存储器的组织

□ 存储单元按字节或字寻址

□ 程序和数据顺序存放

- ✓ 数据段
- ✓ 代码段

□ 读写操作以数据总线宽度为单位





## 口主存储器的工作方式

- ✓ 按存储单元的地址号来实现对存储字各位的写入和读出——按地址存取方式（访存）
- ✓ 为实现上述访存方式，需配置两个寄存器
  - MAR (Memory Address Register)，存储器地址寄存器
  - MDR (Memory Data Register)，存储器数据寄存器
- ✓ MAR
  - 存放要访问的存储单元的地址，其位数对应存储单元的个数
  - 设MAR为16位，则有 $2^{16} = 64K$ 个存储单元
- ✓ MDR
  - 存放从某个存储单元取出的或准备往某个存储单元存入的内容，其位数与存储字长相等



## □ 输入设备

- ✓ 把人们熟悉的某种信息形式转换为机器内部能接收和识别的二进制信息形式
- ✓ 常见的输入设备：键盘、鼠标、扫描仪、模数转换器等

## □ 输出设备

- ✓ 把计算机处理的结果转换为人或其他机器能接收和识别的信息形式
- ✓ 常见的输出设备：显示器、打印机、绘图仪、语言语音设备等

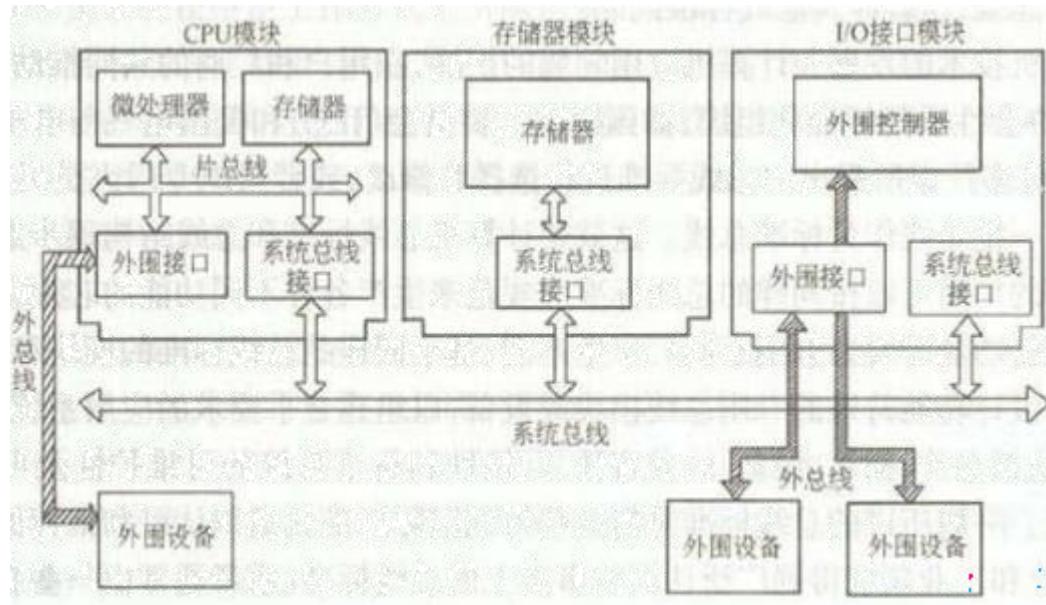
## □ 适配器

- ✓ I/O设备类型繁多、速度各异、数据形式不一，因此，要实现I/O设备与主机之间的连接和信息交换，必须经过一个数据转换和传输的设备，称为**I/O接口**或者**I/O适配器**
- ✓ 类似于一个转换器
  - 用以保证I/O设备用主机要求的形式发送或接收信息



## □ 总线

- ✓ 构成计算机系统的骨架，是计算机各个组成部件间进行数据传送的公用通路
- ✓ 从连接角度看，可以分为内部总线、系统总线和I/O总线
- ✓ 从功能角度看，可以分为数据总线、地址总线和控制总线

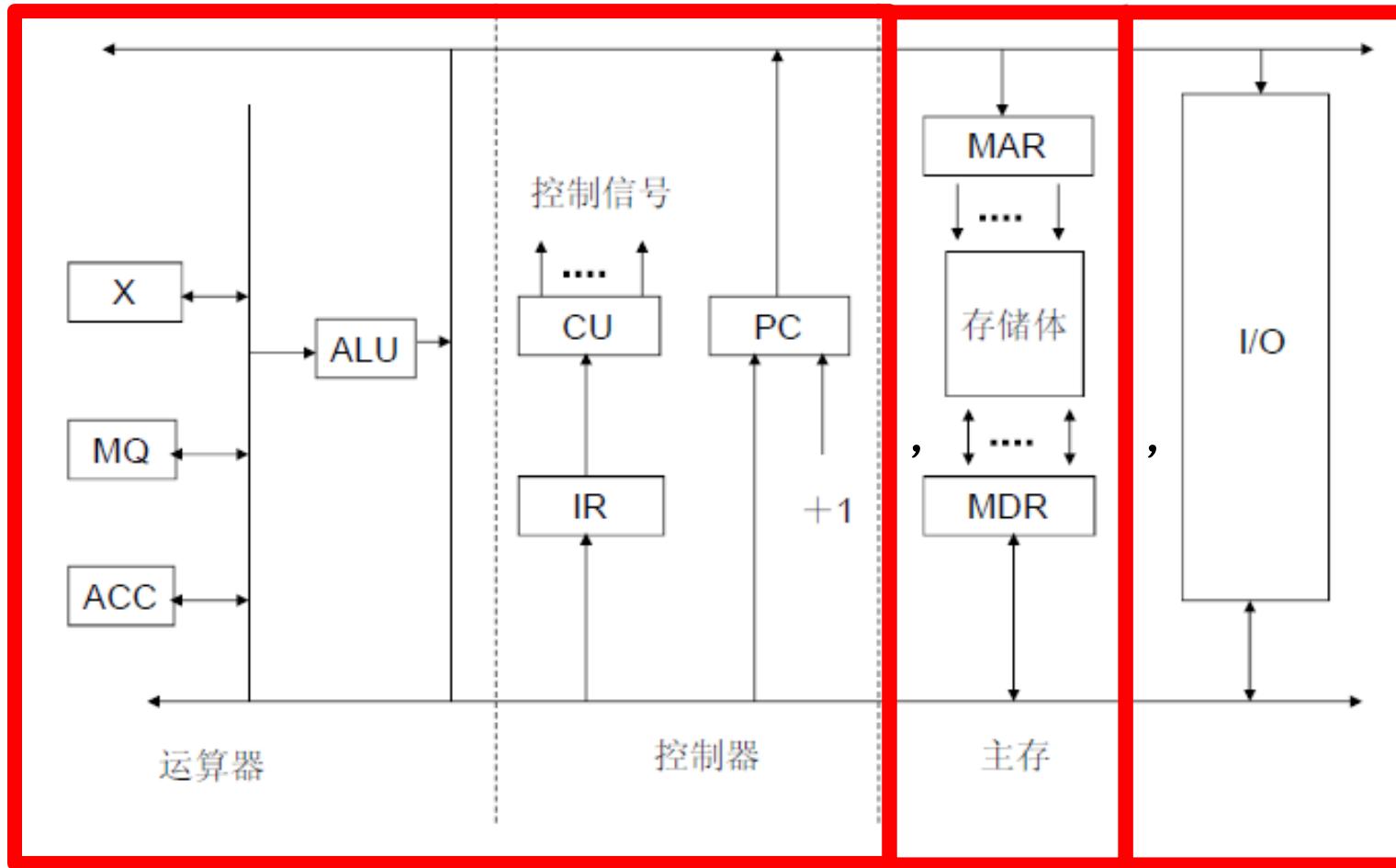


# 细化的计算机组成框图



中国科学技术大学  
University of Science and Technology of China

## 口了解计算机的工作过程





## 主要分为3个阶段

- ✓ 取指：根据当前PC值，从存储地址中取得指令，通过MDR送至IR

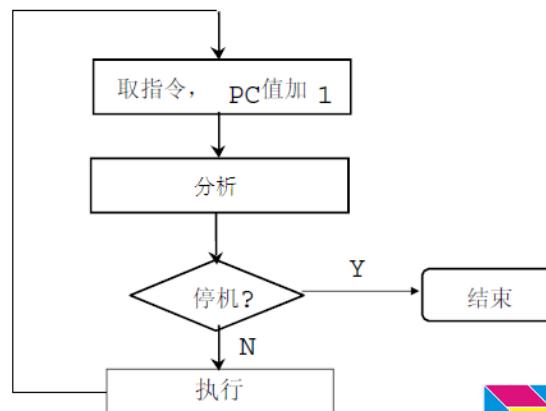
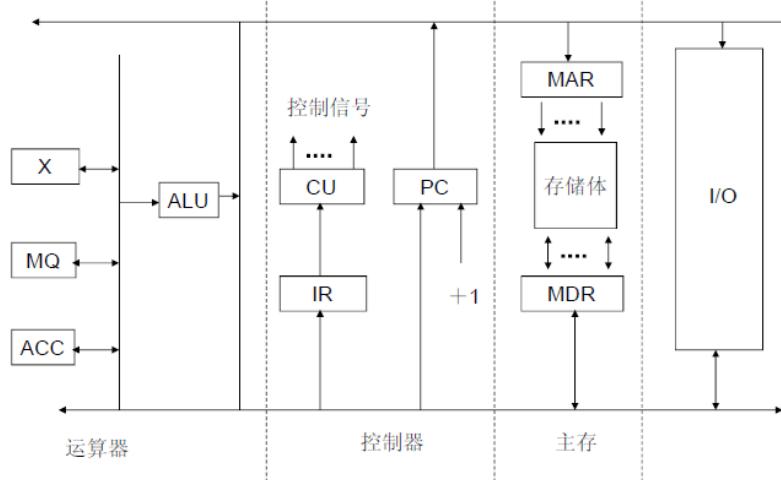
$[PC] \rightarrow MAR, [PC]+1 \rightarrow PC; [MAR] \rightarrow MDR; [MDR] \rightarrow IR$

- ✓ 分析：CU分析IR中的指令，确定操作类型和操作数地址

$[IR] \rightarrow CU: OP[IR] \rightarrow ALU, AD[IR] \rightarrow MAR;$

$[MAR] \rightarrow MDR; [MDR] \rightarrow X \text{ 或 } MQ \text{ 或 } ACC$

- ✓ 执行：ALU部件处理数据，完成操作



计算机的工作过程，就是按存储器中存放的指令顺序反复不断地取指、分析、执行，直到遇到停机指令为止



# 复习 计算机的硬件组成



中国科学技术大学  
University of Science and Technology of China

## 口计算机硬件的五大组成部件

- ✓ 运算器
  - 完成算术运算和逻辑运算，暂存中间结果
- ✓ 存储器
  - 存放数据和程序
- ✓ 控制器
  - 控制、指挥程序和数据的输入、运行以及处理运算结果
- ✓ 输入设备
  - 将人们熟悉的信息形式转换成机器能识别的形式
- ✓ 输出设备
  - 将机器运算结果转换为人们熟悉的信息形式展示出来

## 2. 2计算机组成与计算机体系结构



中国科学技术大学  
University of Science and Technology of China

### □ 计算机体系结构

- ✓ 程序员可见的机器属性，即**概念性的结构与功能特性**
  - 程序员特指机器语言/汇编语言程序员
- ✓ 机器属性包括
  - 指令集、数据类型、存储器寻址技术、I/O机制等

### □ 计算机组成

- ✓ 计算机体系结构的逻辑实现
  - 具有对程序员透明的实现细节
- ✓ 包含实际机器的数据流和控制流的组成和逻辑设计等
- ✓ 着眼于物理机器内部各事件的排序方式与控制方式，各部件的功能以及相互间的联系



# 体系结构的8种属性

- 数据表示
  - ✓ 硬件能直接辨识和操作的数据类型和格式
- 寻址方式
  - ✓ 最小可寻址单位、寻址方式的种类、地址运算
- 寄存器组织
  - ✓ 操作寄存器、变址寄存器、控制寄存器及专用寄存器的定义、数量和使用规则
- 指令系统
  - ✓ 机器指令的操作类型、格式、指令间排序和控制机构
- 存储系统
  - ✓ 最小编址单位、编址方式、主存容量、最大可编址空间
- 输入输出结构
  - ✓ 输入输出的连接方式、处理机/存储器与输入输出设备间的数据交换方式、数据交换过程的控制
- 中断机构
  - ✓ 中断类型、中断级别，以及中断响应方式等
- 信息保护
  - ✓ 信息保护方式、硬件信息保护机制



## □ 对于 Intel 386 与Core i7两种机器

- ✓ 高级语言程序员：认为是同一属性的机器
- ✓ 汇编语言程序员：两种截然不同的机器
  - 指令集、数据类型、寻址技术等不同
  - 因此，这两种机器的体系结构各不相同

## □ 指令系统的问题

- ✓ 计算机**有哪些指令**，属于体系结构的问题
- ✓ **指令如何实现**，即怎么取指、分析指令、取操作数、运算等，属于计算机组成的问题

## □ 例子

- ✓ 一台机器是否具备乘法指令，这是计算机体系统结构的问题
- ✓ 怎么实现乘法指令的功能，则是计算机组成的问题
- ✓ 可以用专门的乘法电路或者用连续相加的加法电路

## 2.3 计算机的软件组成



- 凡用于一台计算机的各种程序，统称为这台计算机的程序或软件系统
- 计算机软件的分类
  - ✓ 系统软件
    - 主要用来管理计算机系统，简化程序设计，简化系统使用方法，提高计算机使用效率，发挥和扩大计算机的功能及用途
    - 可分为四类：
      - 1) 各种服务程序：诊断程序、调试程序、标准函数库等
      - 2) 语言处理程序：汇编器、编译器等
      - 3) 操作系统；
      - 4) 数据库管理系统
  - ✓ 应用软件
    - 用户根据任务需要，利用计算机来解决相应问题而编制的程序
    - 如ERP软件、浏览器、游戏软件、聊天工具等等

# 本章提纲



中国科学技术大学  
University of Science and Technology of China

## □计算机系统概述

- ✓ 计算机系统简介
- ✓ 计算机系统的层次结构

## □计算机的基本组成

- ✓ 计算机硬件组成
- ✓ 计算机组装与体系结构
- ✓ 计算机软件组成

## □计算机硬件性能指标

- ✓ 机器字长
- ✓ 存储容量
- ✓ 运行速度

## □计算机/CPU发展历史

- ✓ 计算机的分代
- ✓ 计算机设计面临的挑战



嵌入式系统实验室  
EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC

### 3.1 机器字长



#### □ 机器字长——CPU一次能处理的二进制数据的位数

- ✓ 通常与CPU的寄存器位数有关，也反映运算部件和数据总线的位数
  - 日常所说的32位、64位机器，这里的32和64就是指机器字长
- ✓ 影响运算速度
  - CPU字长较短，要运算位数较多数据时，则需多次运算才能完成
- ✓ 影响计算精度
  - 字长越长，数的表示范围越大，精度越高
- ✓ 影响硬件成本
  - 字长越长，硬件部件的成本随之增高

为协调计算精度和硬件成本间的制约关系，  
多数计算机采用变字长运算

## 3.2 存储容量



中国科学技术大学  
University of Science and Technology of China

### □ 存储容量——存储器中所有存储单元的总位数

- ✓ 分为主存容量和辅存容量
- ✓ 存储容量 = 存储单元个数 × 存储字长
  - 存储器的地址寄存器MAR的位数反映了存储单元个数
  - 存储器的数据寄存器MDR的位数反映了存储字长
  - 假设MAR为16位、MDR为32位，则表示该存储部件有 $2^{16}=65536$ 个存储单元，总的存储容量为 $2^{16} \times 32b = 2^{21}b = 2Mb = 256KB$

### □ 存储容量表示

- ✓ 通常用字节数来描述容量大小，如B、KB、MB、GB、TB
- ✓ 字节：一个字节B，表示8位二进制数

### 3.3 存储容量 (2)



#### □ 数据通路宽度——数据总线一次能并行传送的信息的位数

- ✓ 影响计算机的有效处理速度
- ✓ 分为CPU内部和CPU外部两种情况
  - 内部数据通路宽度一般等于机器字长，即内部数据线的位数
  - 外部数据通路宽度等于系统数据总线的位数，即CPU与主存、I/O设备之间一次数据传送的信息位数，即**数据字长**

#### □ 字的概念

- ✓ 不同机器其字的位数可能不一样，但一定是字节的倍数
- ✓ 对于某一系列的计算机来说，字的长度是固定的
  - 80x86系列，一个字为16位，单字；32位数据，双字
  - IBM 303X系列，一个字为32位，单字；16位数据，半字

## 3.2 存储容量 (3)



中国科学技术大学  
University of Science and Technology of China

### 口机器字长 VS. 存储字长 VS. 数据字长

#### ✓ 三者可以相等，也可以不等

- 早期计算机，三者一般相等，这样一次访存可以取一条指令或一个数据
- 随着技术的发展，对机器字长和数据字长有了可变性需求，这样三者可以不相等，但必须是字节的整数倍

#### ✓ 典型例子

- 8086处理器，机器字长/存储字长/数据字长都是16位
- Pentium处理器，机器字长32位，存储字长64位

### 口存储器带宽

#### ✓ 单位时间内从存储器读出的二进制数信息量，一般用 B/s 表示

### 3.3 运算速度



- 衡量计算机运算速度的指标，与许多因素有关，如机器主频、操作类型、主存速度等。
- 基准程序法 (benchmark)
  - ✓ 考虑CPU、I/O结构、操作系统、编译器效率等
  - ✓ 评价计算机的实际工作能力
- 目前常用的测试程序：
  - ✓ 真实程序：C编译器、Matlab、Photoshop… (移植带来问题)
  - ✓ **核心程序**：由从真实程序中提取的较短但很关键的代码构成。  
Livermore loops、Linpack
  - ✓ 合成测试程序 ( Whetstone/Dhrystone )
  - ✓ **测试程序组** ( 测试程序包：Benchmarks ) : 选择一组各个方面有代表性的测试程序，组成的一个通用测试程序集合。

# 基准测试套 (Benchmark)



中国科学技术大学  
University of Science and Technology of China

## □ Dhrystone

- ✓ 整数测试程序
- ✓ 一个综合性的基准测试程序
- ✓ 适于比较同一家族的机器



## □ Linpack

- ✓ 测试向量性能和高速缓存性能

## □ Whetstone

- ✓ 一个综合性测试程序
- ✓ 测试浮点操作、整数计算和功能调用等性能

## □ TPC (Transaction Processing Council)

- ✓ 计算机（服务器）事务处理性能

## □ SPEC

- ✓ System Performance Evaluation Cooperative



# 测试程序包：[www.SPEC.org](http://www.SPEC.org)

AutoBench：工业控制中常见的计算的评测，比较简单

## □ 桌面

- ✓ : Angle to time conversion
- Basic interger/floating point = arctan(x)
- Bit manipulation
- Cache buster
- Can remote data request
- Fft
- Fir filter
- Iir filter
- Idct
- Ifft
- Matrix arithmetic

## □ 服务

- ✓ : Pulse width modulation
- Road speed calculation
- Table lookup and interpolation
- Tooth to spark
- ✓ : Consumer :
  - JPEG
- ✓ : High-Pass Grey-scale filter
- RGB to CMYK Conversion
- RGB to YIQ Conversion

## □ 嵌入

Java GrinderBench : J2ME的测试

- ✓ |

## □ 大数

Networking : 网络传输

- ✓ | IP Reassembly
- IP Packet Check
- IP NAT

## □ 人工

- OSPF(open shortest path first)/Dijkstra
- QOS
- Route Lookup
- TCP
- Packet Flow

Digital Entertainment : 娱乐/消费电子中常见的应用，复杂度较高

- AES
- DES
- High-pass grey scale filter
- Huffman decoding
- Mp3 decode
- Mpeg-2 decode
- Mpeg-2 encode
- Mpeg-4 decode
- Mpeg-4 encode
- RGB-CMYK Conversion
- RGB-YIQ Conversion
- RSA

OA :

- Bezier : 画贝塞尔曲线
- Dithering : 图像输出到打印机
- Ghostscript : 跑ps printer
- Rotate
- Text parsing : 处理PCL/Postscript脚本

定

Telecomm :

- Autocorrelation : 数字信号处理中的常用计算 ;
- Bit allocation : DSL modem中的常用功能
- Convolutional encoder : 无线通信系统中使用 ; 纠错
- Fft :
- Viterbi decoder : 纠错

Multibench : 18个测试，专测多核性能

### 3.3 运算速度



口 早期衡量运算速度的普通方法：完成一次加法或乘法所需的时间

✓ CPI (Cycle per Instruction)

- 指令周期，表示执行一条指令所需的平均时钟周期数

$$CPI = \frac{\text{执行程序所需的CPU时钟周期数}}{\text{程序包含的指令条数}}$$

✓ MIPS (Million Instruction Per Second)

- 百万条指令每秒，即单位时间内执行的指令数

$$MIPS = \frac{\text{程序的指令条数}}{\text{程序的执行时间} \times 10^6}$$

✓ MFLOPS (Million Floating Point Operation Per Second)

- 百万次浮点操作每秒，用来衡量机器浮点运算的性能

$$MFLOPS = \frac{\text{程序的浮点操作次数}}{\text{程序的执行时间} \times 10^6}$$



# 度量单位



## Computer Performance

Name	FLOPS
<u>yotta</u> FLOPS	$10^{24}$
<u>zetta</u> FLOPS	$10^{21}$
<u>exa</u> FLOPS	$10^{18}$
<u>peta</u> FLOPS	$10^{15}$
<u>tera</u> FLOPS	$10^{12}$
<u>giga</u> FLOPS	$10^9$
<u>mega</u> FLOPS	$10^6$
<u>kilo</u> FLOPS	$10^3$





例1：一微处理器，主频为**20MHz**，请计算其**时钟周期**。若一个**机器周期**由**2**个时钟周期组成，平均每条指令用**3**个机器周期的时间(即“**指令执行时间**”),请计算该处理器的平均运行速度**MIPS**。

[解]

$$\begin{aligned}\text{时钟周期} &= \frac{1}{\text{时钟频率}} = \frac{1}{20MHz} \\ &= 0.05 \times 10^{-6} S\end{aligned}$$

$$\begin{aligned}\text{平均速度} &= \frac{1}{\text{指令执行时间}(S)} \\ &= \frac{1}{0.05 \times 10^{-6} \times 6} = 3.33MIPS\end{aligned}$$



# 详细分析CPU的性能

□ 将程序执行的时间进行分解。

- ✓ 将计算机系统中与实现技术和工艺有关的因素提取出来。这个因素就是计算机工作的时钟频率 ( $f$ )，单位是MHz

- ✓ 可以测量执行程序使用的总时钟周期数 (CLK)

□ 通过这两个参数我们就可以知道程序执行的CPU时间：

$$T_{CPU} = CLK / f$$



# CPU的性能

- 程序执行过程中所处理的指令数，记为IC。
- 每条指令执行所需要的时钟周期数CPI (Cycles Per Instruction)：每条指令执行所需要的平均时钟周期数。

$$CPI = CLK / IC$$

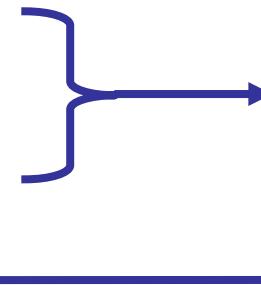


# CPU性能公式

$$\boxed{CPI = CLK / IC}$$

$$\rightarrow IC \times CPI = CLK$$

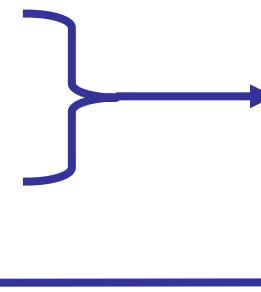
$$T_{CPU} = CLK / f$$



时  
钟  
周  
期  
时  
间

$$\rightarrow T_{CPU} = CPI \times IC / f$$

$$T_{CLK} = 1 / f$$



$$\rightarrow T_{CPU} = CPI \times IC \times T_{CLK}$$



# CPU性能公式和体系结构

- 时钟频率 ( $f$ ) : 反映了计算机实现技术、生产工艺和计算机组织。
- 指令的平均时钟周期数 ( $CPI$ ) : 反映了计算机组织(如流水线) 和计算机指令集的结构。
- 程序的指令数 ( $IC$ ) : 反映了计算机指令集的结构和编译技术。



# 深入CPU性能公式

口假设计算机系统有 $n$  种指令， 其中第 $i$  种指令的处理时间为 $CPI_i$ ， 在程序中第 $i$  种指令出现的次数为  $IC_i$ 。

吉普森（Gibson）法

$$T_{CPU} = CLK / f$$

$$CLK = \sum (IC_i \times CPI_i)$$

$$\begin{aligned} T_{CPU} &= \sum (IC_i \times CPI_i) / f \\ &= \sum (IC_i \times CPI_i) \times T_{CLK} \end{aligned}$$



# 深入CPU性能公式

$$CPI = CLK / IC$$

$$CLK = \sum (IC_i \times CPI_i)$$

$$\begin{aligned} CPI &= \sum (IC_i \times CPI_i) / IC \\ &= \sum [(IC_i / IC) \times CPI_i] \end{aligned}$$

其中： $IC_i / IC$  反映了第*i*种指令在程序中所占的比例。

# 练习 1



□ 某台计算机只有Load/Store 指令能对存储器进行读/写操作，其它指令只对寄存器进行操作。根据对某程序跟踪实验结果，已知每种指令所占的比例及CPI数如下：

指令类型	指令所占比例	CPI
算逻指令	43%	1
Load指令	21%	2
Store指令	12%	2
转移指令	24%	2



- 求上述情况下的平均CPI。
- 假设该程序由M条指令组成。算逻运算中25%的指令两个操作数中的一个已在寄存器中，另一个必须在算逻指令执行前用Load指令从存储器取到寄存器。因此有人建议增加另一种算逻指令，其特点是一个操作数取自寄存器，另一个操作数取自存储器，即寄存器—存储器类型，假设这种指令的CPI等于2。同时，转移指令的CPI变为3。求新指令系统的平均CPI。

答



- $CPI = (43 \times 1 + 21 \times 2 + 12 \times 2 + 24 \times 2) \% = 1.57$
- $43\% \times 25\% = 11\%$

指令类型	原比例	原CPI	新比例	CPI
新算逻指令	0%	0	11%	2
原算逻指令	43%	1	32%	1
Load指令	21%	2	10%	2
Store指令	12%	2	12%	2
转移指令	24%	2	24%	3

- $CPI_{new} = (11 \times 2 + 32 \times 1 + 10 \times 2 + 12 \times 2 + 24 \times 3) \% = 1.70$

$$CPI_{new} = (11 \times 2 + 32 \times 1 + 10 \times 2 + 12 \times 2 + 24 \times 3) / 89\% = 1.91$$



# 计算机性能的评测

□ 利用**时间**的概念评测一台计算机的性能和测试者所处的角度有关。

- ✓ **响应时间**（执行时间）：从事件开始到结束之间的时间。
- ✓ **吞吐率** (Throughput)：在单位时间内所能完成的工作量（任务）。（多用户系统）

□ 用户以响应时间为标准，多道程序系统以吞吐率为标准。



# “快” (Fast) 的含义

“**X比Y快**” 的含义是：对于给定任务，X的响应时间比Y少。 “**X比Y快n倍**” 是指：

$$\frac{\text{响应时间}_Y}{\text{响应时间}_X} = n$$

由于响应时间与性能成反比，又有：

$$\frac{\text{响应时间}_Y}{\text{响应时间}_X} = \frac{\frac{1}{\text{性能}_Y}}{\frac{1}{\text{性能}_X}} = \frac{\text{性能}_X}{\text{性能}_Y}$$



## □ 可靠性

- ✓ 平均无故障运行时间，MTBF (Mean Time Between Failures)

## □ 可用率

- ✓ 度量系统的可用性
- ✓ 计算式子： $\text{MTBF} / (\text{MTBF} + \text{MTTR})$ 
  - MTTR (Mean Time To Repair) , 平均修复时间



- 执行时间 (CPU时间、Elapsed Time)
- 峰值速度 (Peak Performance)
- 负载 (load)
- 开销 (Overhead)
- 利用率 (Utilization Ratio)
- 饱和性能 (Saturate Performance)
- 带宽 (Bandwidth)
- 延迟 (Latency)
- 加速比 (Speedup)
- 效率 (Efficiency)



# 本章提纲



中国科学技术大学  
University of Science and Technology of China

## 口计算机系统概述

- ✓ 计算机系统简介
- ✓ 计算机系统的层次结构

## 口计算机的基本组成

- ✓ 计算机硬件组成
- ✓ 计算机组装与体系结构
- ✓ 计算机软件组成

## 口计算机硬件性能指标

- ✓ 机器字长
- ✓ 存储容量
- ✓ 运行速度

## 口计算机/CPU发展历史

- ✓ 计算机的分代
- ✓ 计算机设计面临的挑战

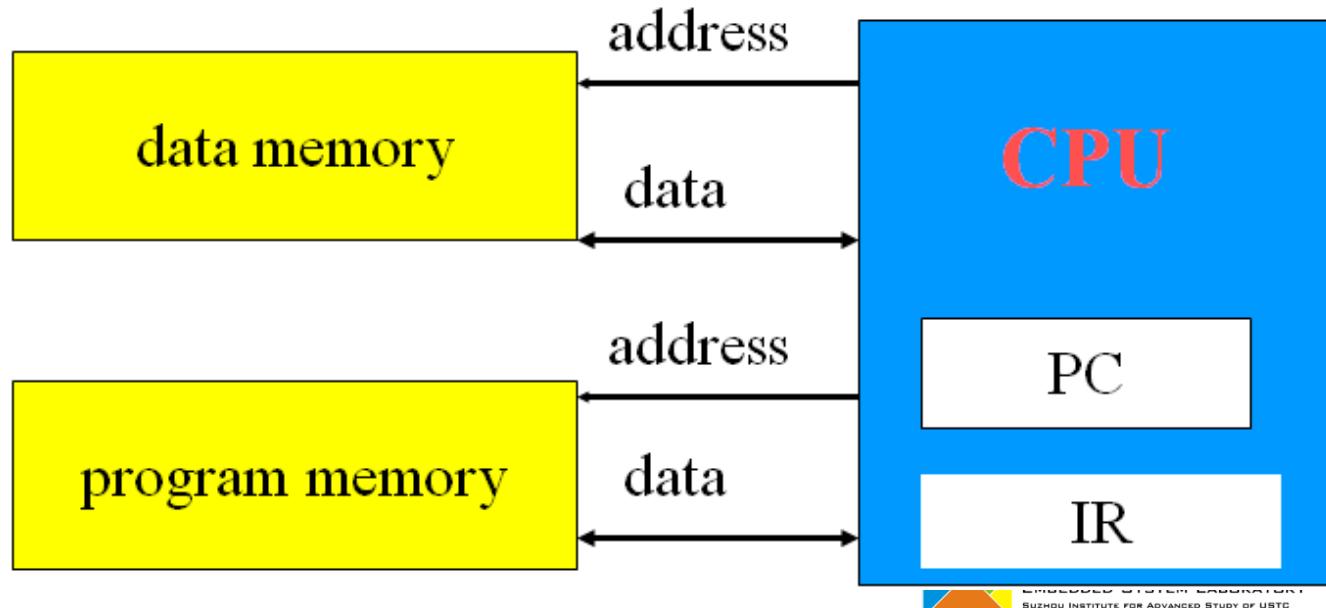


嵌入式系统实验室  
EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC

# 机械式计算机=>自动计算机



- 1642年，法国哲学家和数学家帕斯卡 (Blaise Pascal)
  - ✓ 第一台加减法计算机
- 1833年，英国科学家巴贝奇 (Charles Babbage)
  - ✓ 提出制造“自动化计算机”的设想和**程序控制**概念，但未实现
- 1944年，艾肯 (Howard Aiken, IBM)
  - ✓ 第一台数字式**自动**计算机Mark I (**继电器**计算机)
  - ✓ 实现了巴贝奇设想，并提出**哈佛结构** (Harvard architecture)



# 第一台现代电子计算机，1946



- ENIAC( Electronic Numerical Integrator and Computer, 电子数字积分器和计算机), 1946年美国宾夕法尼亚大学, 运算速度 5000次/秒, 功耗150kw/h, 占地 $170\text{m}^2$  , 造价100万美元。用于测定氢弹可靠性。
- 不具备存储程序能力, 程序要通过外接电路板输入(wired)。对于每种类型的题目, 都要设计相应的外接插板。

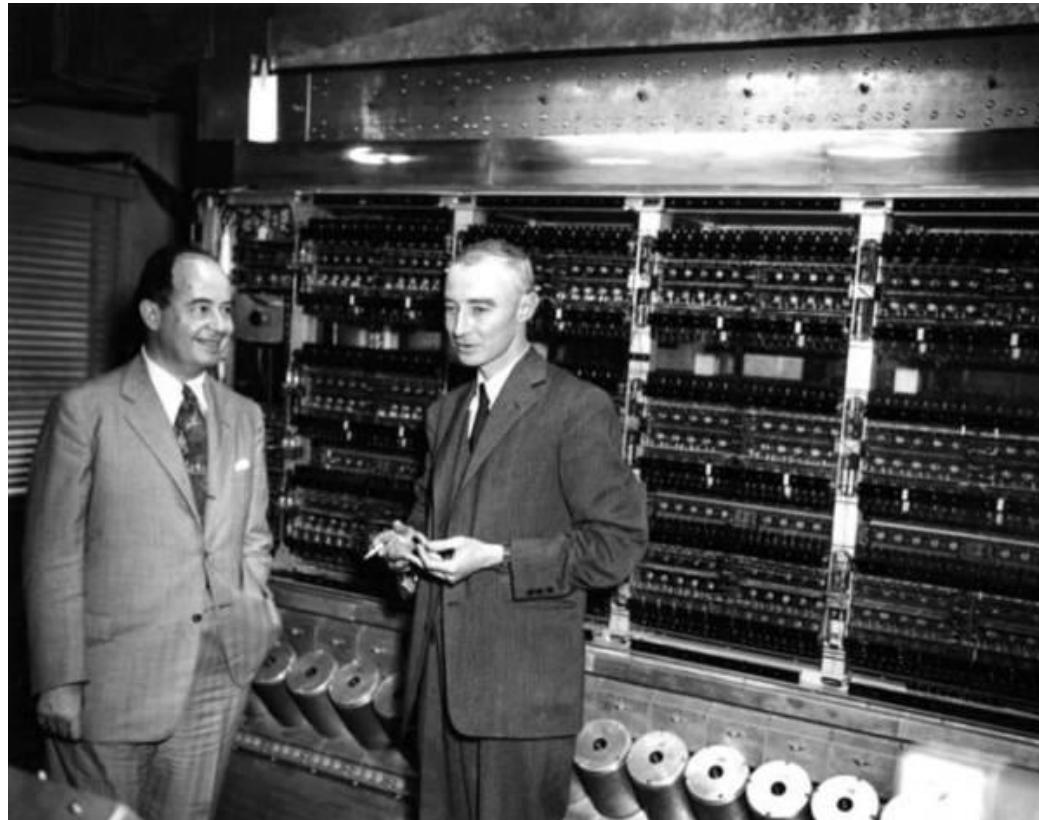
# EDVAC计算机, 1944~1952



中国科学技术大学  
University of Science and Technology of China

## □ Electronic Discrete Variable Automatic Computer 电子离散变量自动计算机

- ✓ 1MHz, 二进制, 字长32位, 串行
- ✓ 存储程序 (Stored Program)



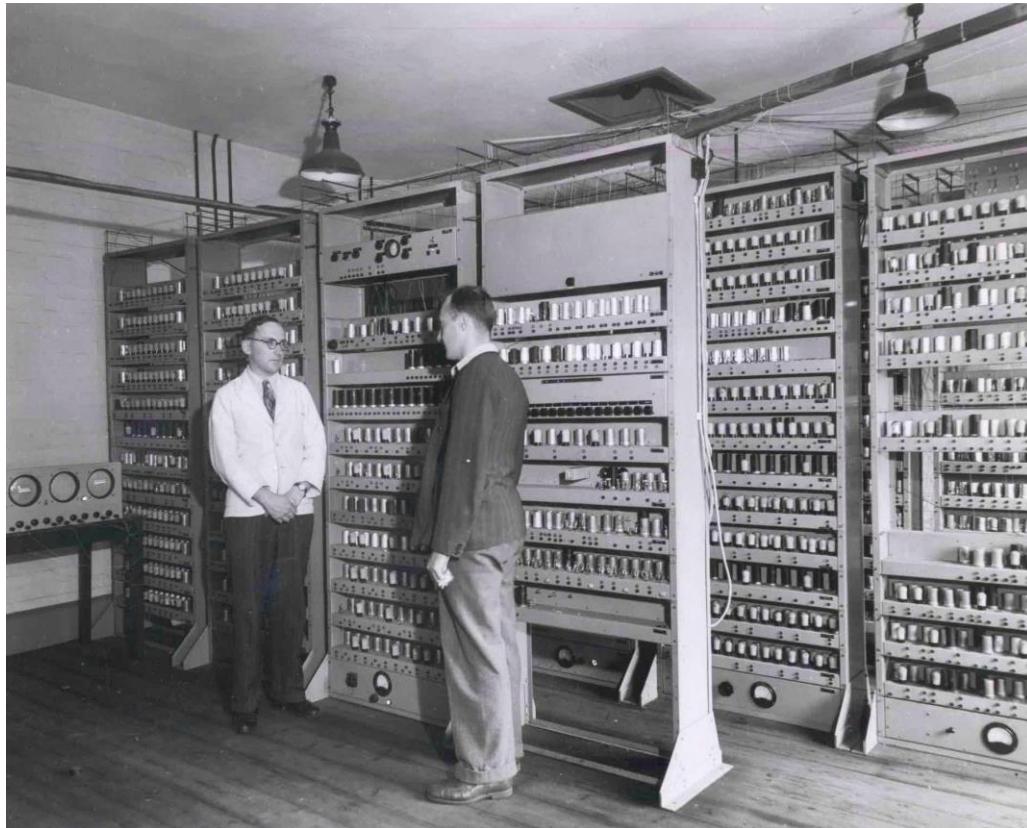
冯诺依曼 (John von Neumann, 1903~1957) 和奥本海默在普林斯顿计算机前

# 第一台存储程序计算机EDSAC



中国科学技术大学  
University of Science and Technology of China

- Electronic Delay Storage Automatic Calculator
- ✓ 参考EDVAC机，1946～1949年在剑桥实现



- 采用水银延迟线为存储器，可存储34b字长的**512字**。
- 加法时间**1.5ms**，乘法时间**4ms**。
- 串行**计算机（数据传输和运算按位逐一进行）
- 微程序，子程序，**Cache**, ...



第二届图灵奖  
(1967)

莫里斯·威尔克斯



嵌入式系统实验室  
EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC



# 第一台

□ 微机：Altair 8800，1975

✓ 显示器？

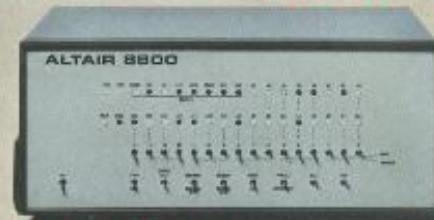
□ IBM PC机：IBM 5150，1981

- ✓ CPU: Intel 8088, 频率4.77MHz (浮点运算要加装Intel 8087加速卡)
- ✓ RAM: 16KB, 最大可到256KB
- ✓ 显示适配器: 16色输出与分辨率, 640 × 200 (CGA)
- ✓ 储存: 录音带, 5.25吋软盘驱动器要选购, 甚至没办法装硬盘
- ✓ OS: PC-DOS 1.0 (尚未改名为MS-DOS)
- ✓ 基础报价: \$1,565, 如果要全配 (内存全满、彩色屏幕等) 则是\$6,000

## PROJECT BREAKTHROUGH!

World's First Minicomputer Kit  
to Rival Commercial Models...

"ALTAIR 8800" SAVE OVER \$1000



## ALSO IN THIS ISSUE:

- An Under-\$90 Scientific Calculator Project
- CCD's—TV Camera Tube Successor?
- Thyristor-Controlled Photoflashers



## TEST REPORTS:

- Technics 200 Speaker System
- Pioneer RT-1011 Open-Reel Recorder
- Tram Diamond-40 CB AM Transceiver
- Edmund Scientific "Kirlian" Photo Kit
- Hewlett-Packard 5381 Frequency Counter





# 计算机的分代

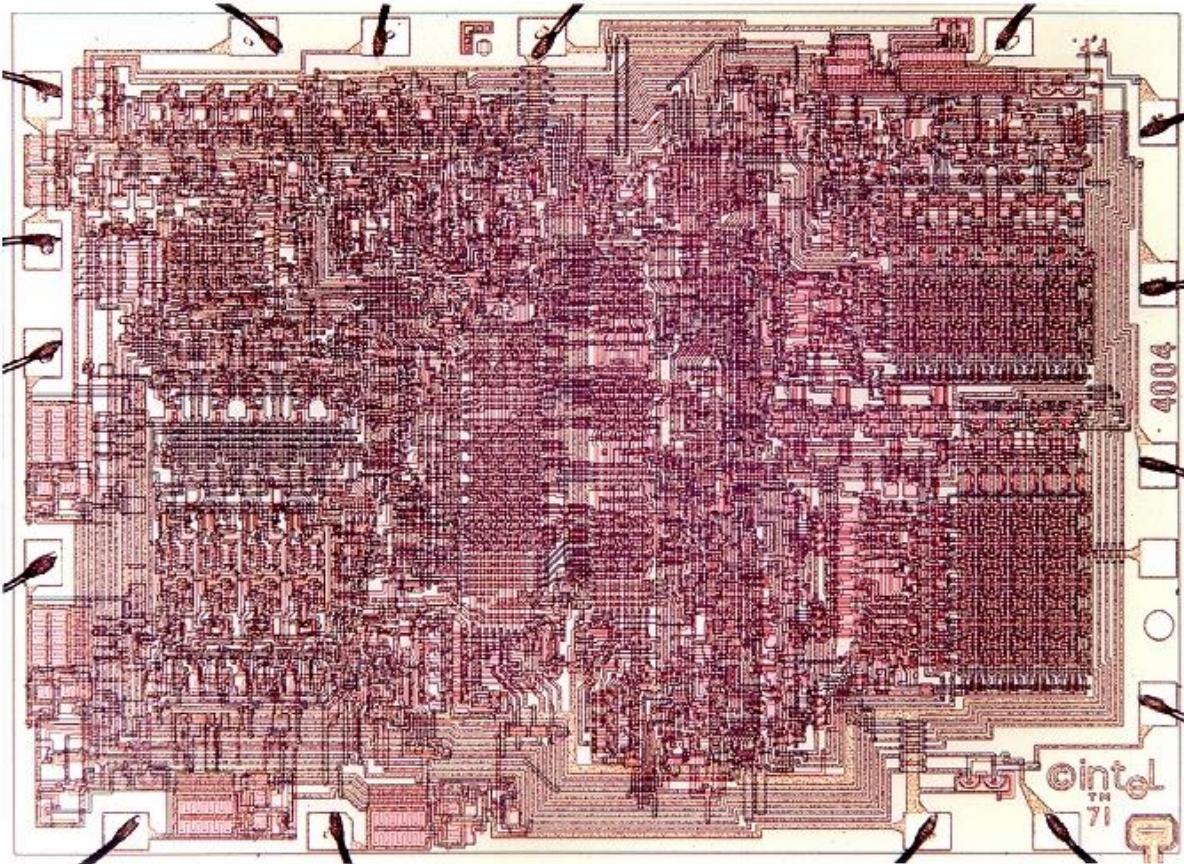
分代	器件	体系结构技术	软件技术	典型机器
第一代 (1945-1954)	电子管和继电器	存储程序计算机、程序控制I/O	机器语言和汇编语言	普林斯顿ISA、ENIAC IBM701, 650
第二代 (1955-1964)	晶体管、磁芯、印刷电路 PCB	浮点数据表示、寻址技术、中断、I/O处理机	高级语言和编译、批处理监控系统	Univac LARC CDC1604 IBM7030
第三代 (1965-1974)	SSI和MSI、多层印刷电路、微程序	流水线、Cache、先行处理、系列计算机	多道程序和分时操作系统	IBM360/370CD C6600/7600、DEC PDP-8
第四代 (1974-1990)	LSI和VLSI、半导体存储器	向量处理、分布式存储器	并行与分布处理	Cray-1、IBM 3090、DEC VAX9000、Convax-1
第五代 (1991-)	高性能微处理器、大规模高密度电路	指令级并行、SMP、MPP、网络	可扩展并行与分布处理	SGI Cray T3E IBM xServer Sun E10000





# Intel为例看CPU的历史

□微处理器集中反映着最新的工艺进展，先进的设计技术和发展结构

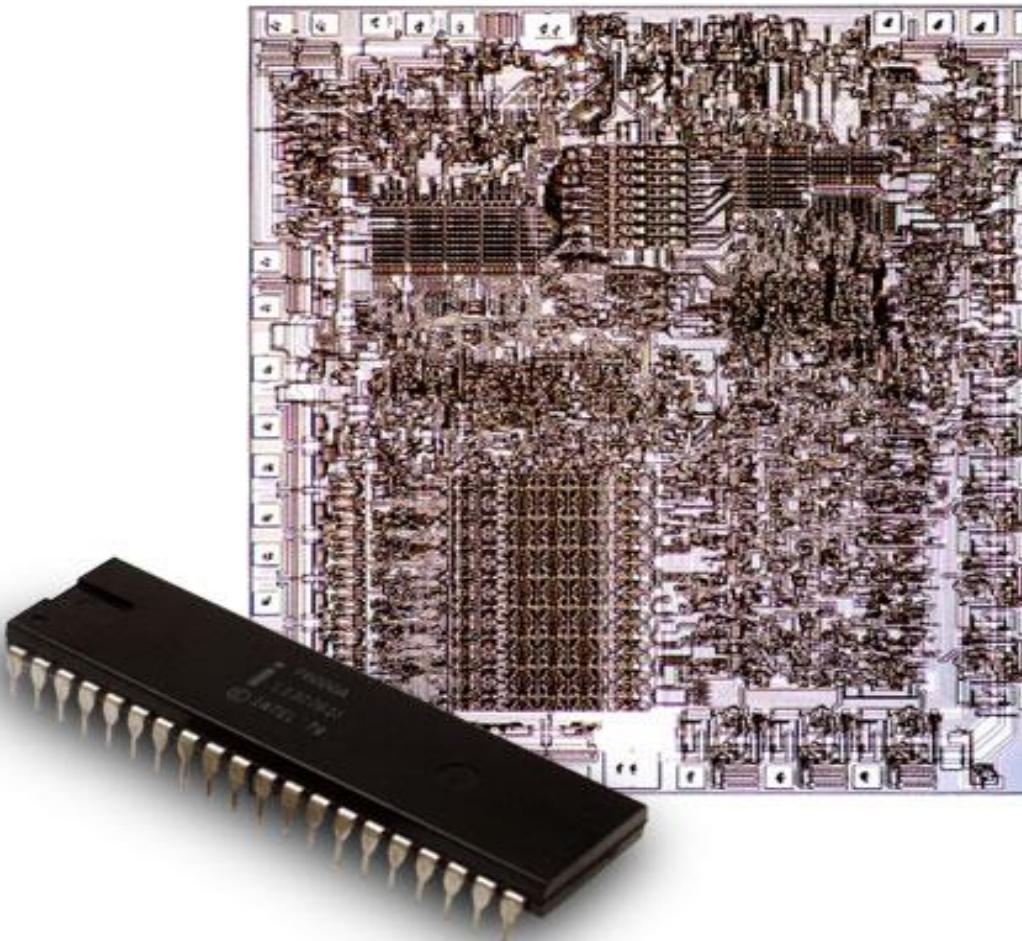


Intel 4004,  
(1971), 世  
界上第一  
个商业化的  
微处理器。

2300个晶体  
管，46条指  
令，0.75  
MHz.

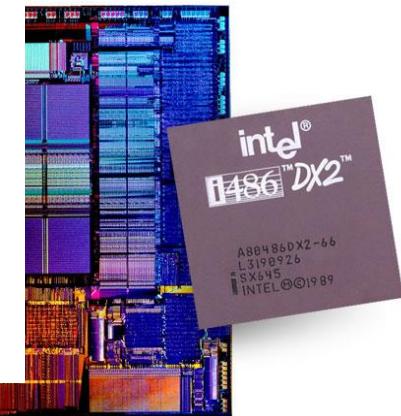


# Intel 历史从4004开始



电子计算机  
令

（大业笃工）



# 历史从这里飞速发展



中国科学技术大学  
University of Science and Technology of China

## □ 1993～1994

- ✓ 80586 : PENTIUM (P5)
- ✓ AMD K5、Cyrix 6x86
- ✓ 64位RISC
  - DEC: Alpha 21164,
  - IBM、Motorola、Ap
  - SGI MIPS: R10000(

## □ 1995

- ✓ PENTIUM Pro 200MHz,
- ✓ Pentium MMX (233MHz)

## □ 1997: PENTIUM II (

## □ 1999:

- ✓ PENTIUM III 1.2GHz,
- ✓ AMD Athlon (K7)



嵌入式系统实验室

EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC

# 历史在这里使频率不斷攀升



中国科学技术大学  
University of Science and Technology of China

## □ 2000:

- ✓ AMD Thunderbird和Duron
- ✓ VIA Cyrix III , 800MHz~
- ✓ PENTIUM IV, 0.18μm,

## □ 2001~: PENTIUM IV

- ✓ Northwood: 0.13μm, -> 万, 铜布线
- ✓ 超线程(HT), 0.13μm, 3.
- ✓ Prescott: 90nm, 31级流水, 频533MHz或800MHz, 1.





## □ 2003～：服务器用多核CPU

- ✓ Sun UltraSPARC IV. 8MB缓存/核, 2.4 GB/s内存接口, 每处理器16GB内存, 0.13 μm, 355mm<sup>2</sup>, 6,600 万, 1.2GHz.

- ✓ IBM POWER5 2.8GHz 双核心, 3200M前端总线, 3亿, 双核4线程, 1.2GHz.

## □ 2005～：

- ✓ 2005, Intel Pentium D 3.2GHz 双核4线程, 3.2GHz, 8M前端总线, 3亿, 双核4线程, 1.2GHz.
- ✓ 2006, Intel Core 2 Duo E6300 2.8GHz 双核4线程, 3.2GHz, 8M前端总线, 3亿, 双核4线程, 1.2GHz.

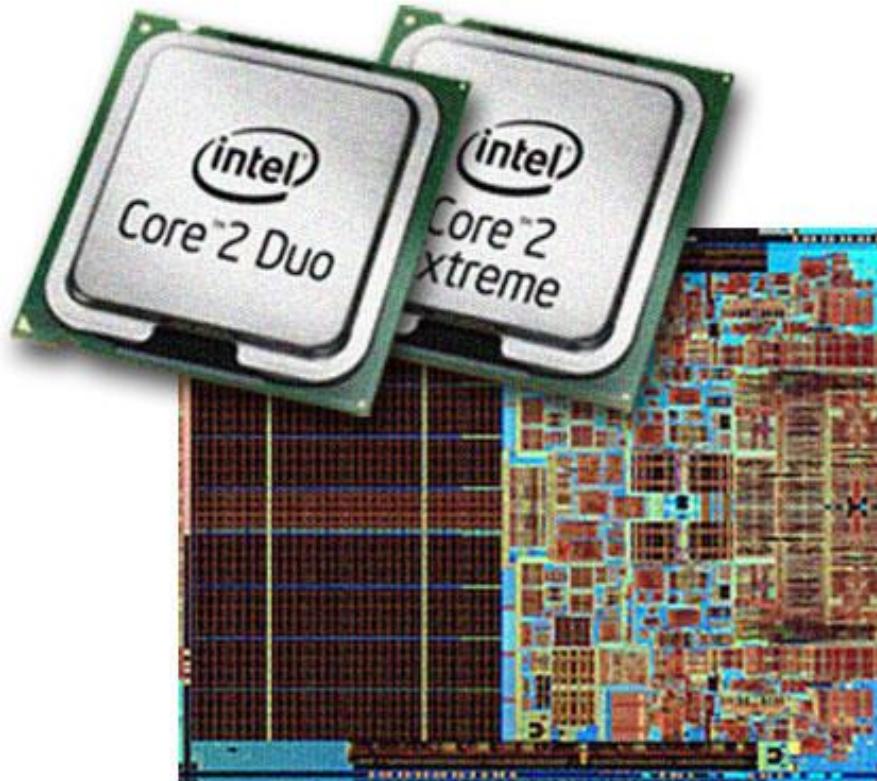




# 多核技术持续发展

□ 2005～

- ✓ AMD A
- ✓ 2007,  
技术,
- ✓ Intel  
1.33G
- ✓ Core  
前端总线



线, EM64T  
体管。

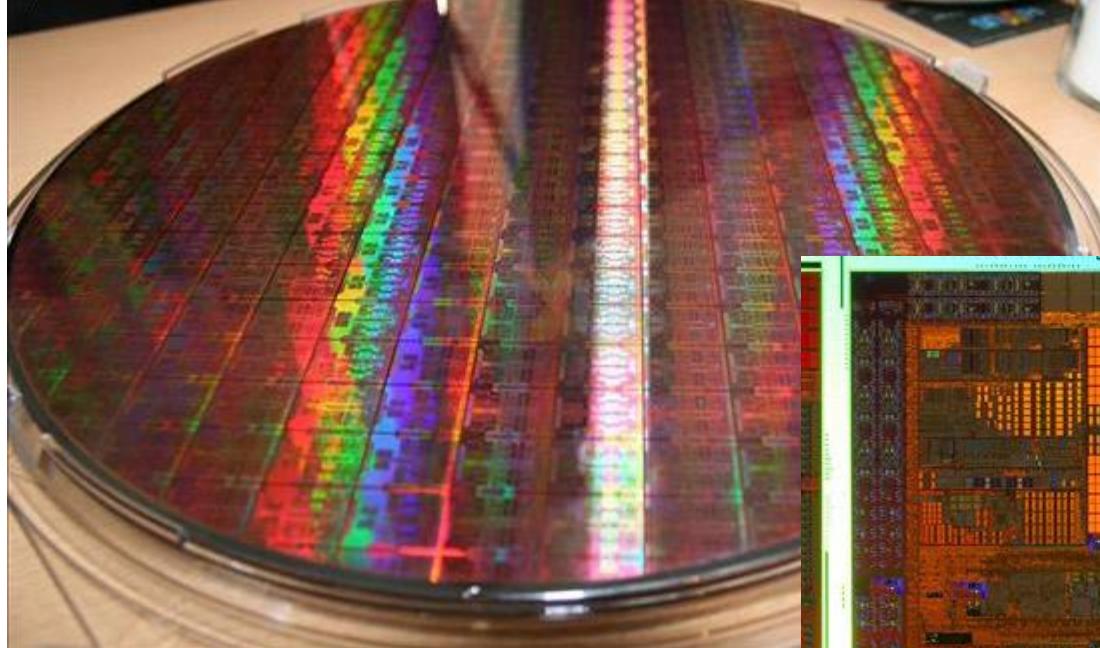
前端总线

z, 45nm,

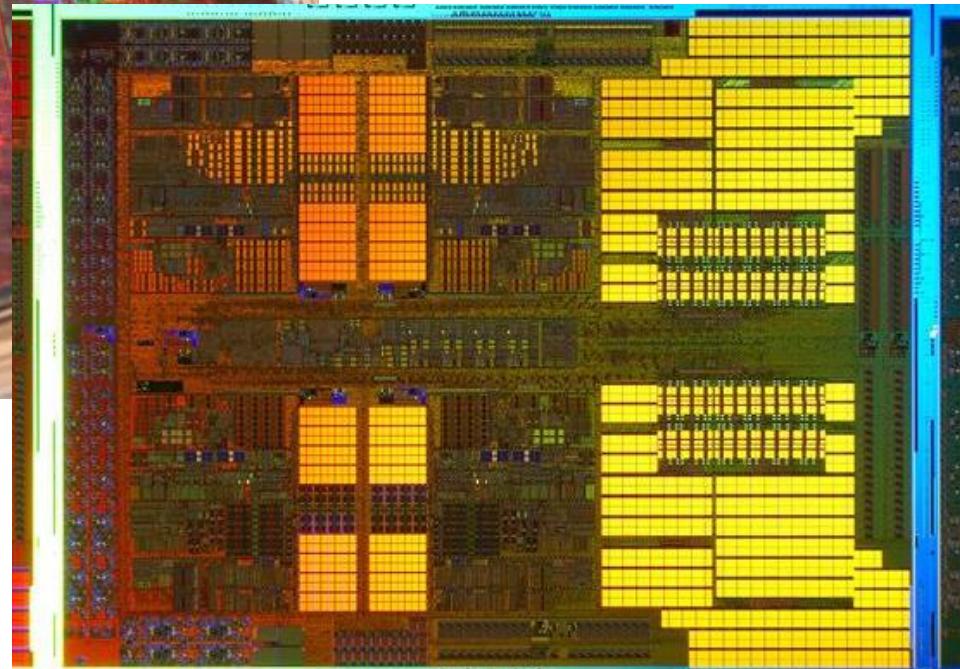




# 多核技术日趋成熟



2008, IBM/AMD,  
45nm, 原生四核





# 多核技术日趋成熟

- 2008, Intel, 45nm Yorkfield四核处理器, Atom (凌动) 处理器。25mm<sup>2</sup>, 4700万, 2.5W-4W
- AMD Phenom X3 8750, 翌龙三核
- Intel Core i7, 三通道DDR3内存控制器 (20GB/s)、QPI总线、SMT、SSE4.2, 8MB L3 (游戏、影音娱乐、工作站)。



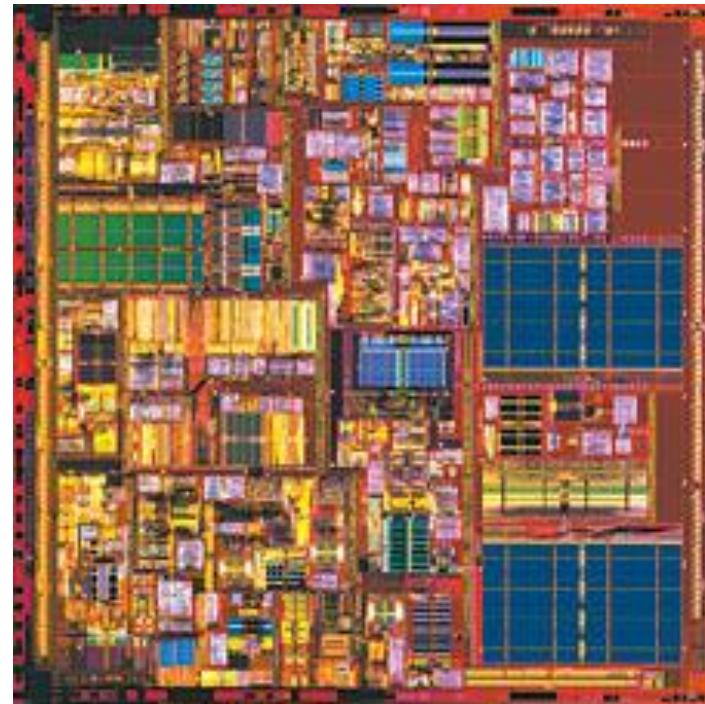


# 计算机实现技术的发展

从4004的2300个晶体管到今天拥有4亿1000万晶体管的双核微处理器，工作电压从12伏特到1.2伏特，20万倍：10倍



外形图

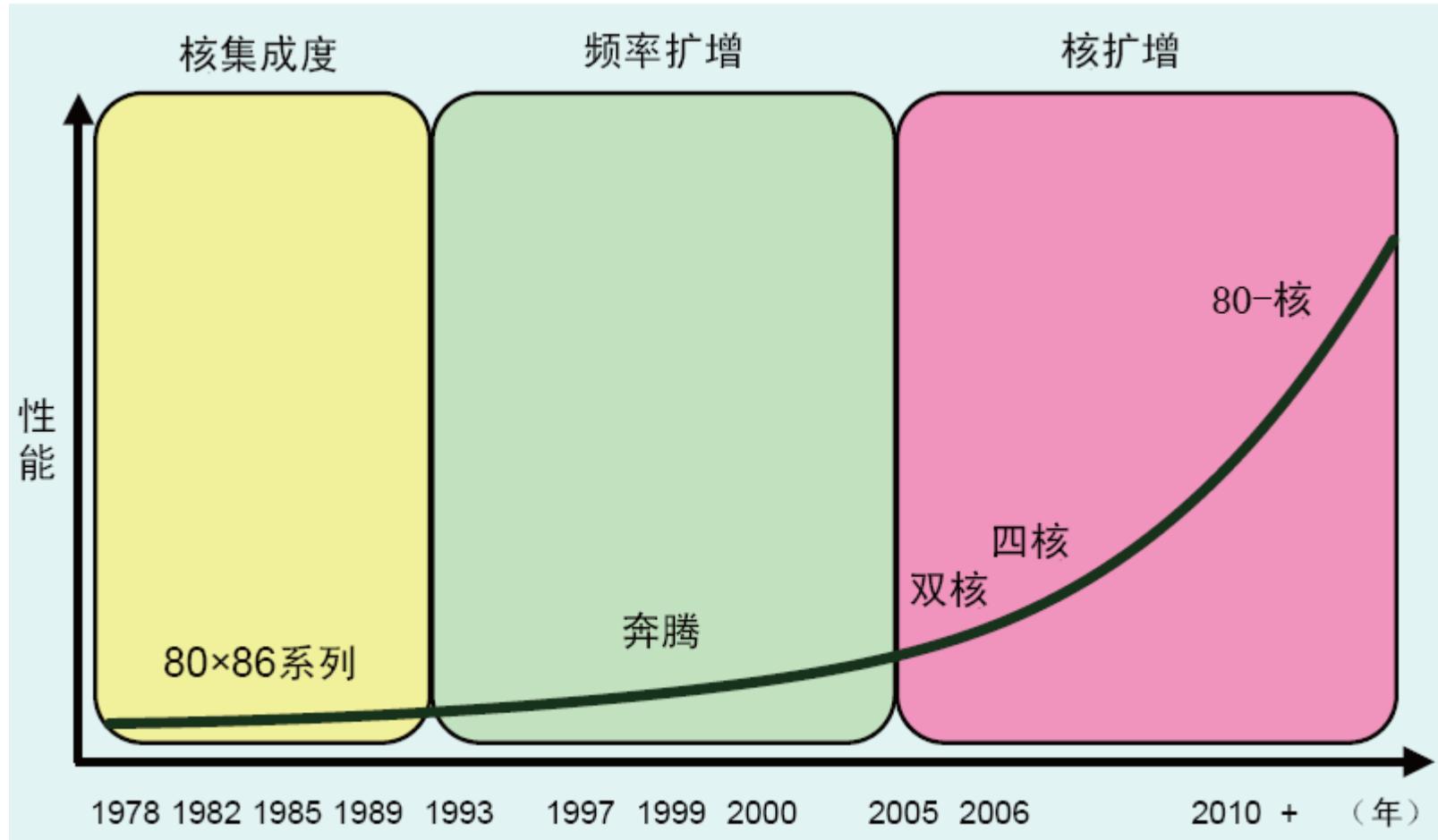


芯片图

# CPU发展趋势图



中国科学技术大学  
University of Science and Technology of China



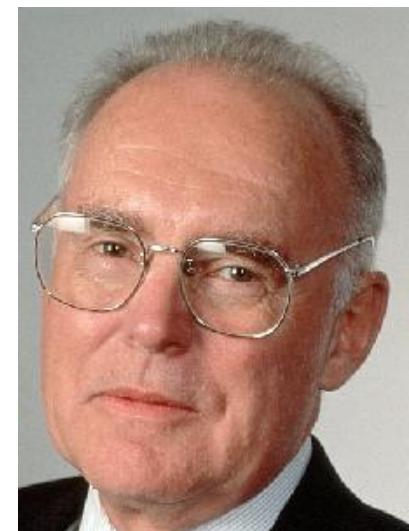
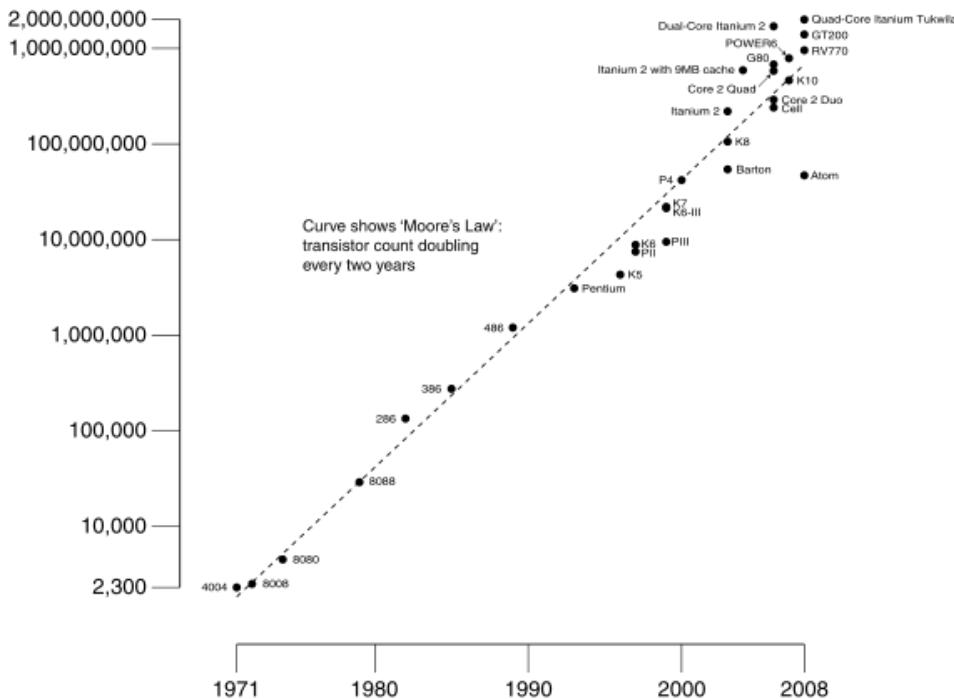
Higher Performance: Supercomputing in the Connected Era. Michael Dell 2008

# Moore定律 ( Moore's Law )

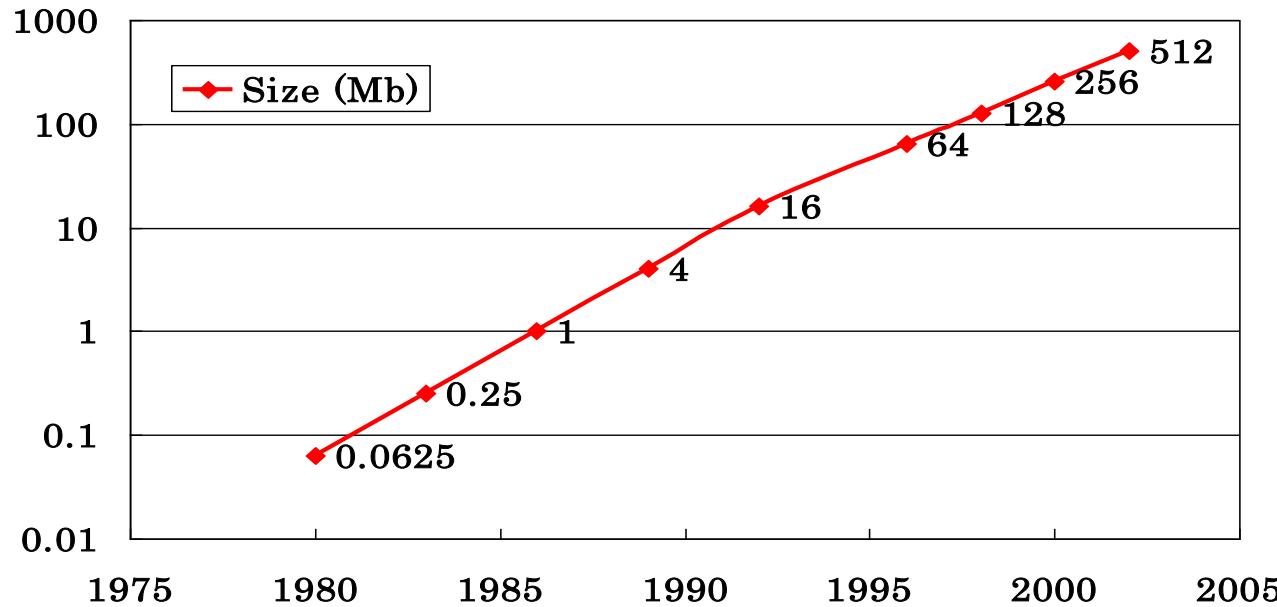


中国科学技术大学  
University of Science and Technology of China

- Original definition (Moore, 1965): Microprocessor transistor count **doubles every year**.
- Then changed to “**doubles every 2 years**”.
- Since then, other measures were used, such as processing power, complexity, etc. and doubling time settled at **18 months**.



# 计算机实现技术的发展



- 半导体DRAM(动态随机访问存储器)。芯片存储密度每年增长速率略低于60%，**平均三年增长2倍**，存储周期时间的减少比较缓慢，**大约是每10年减少1/3**。

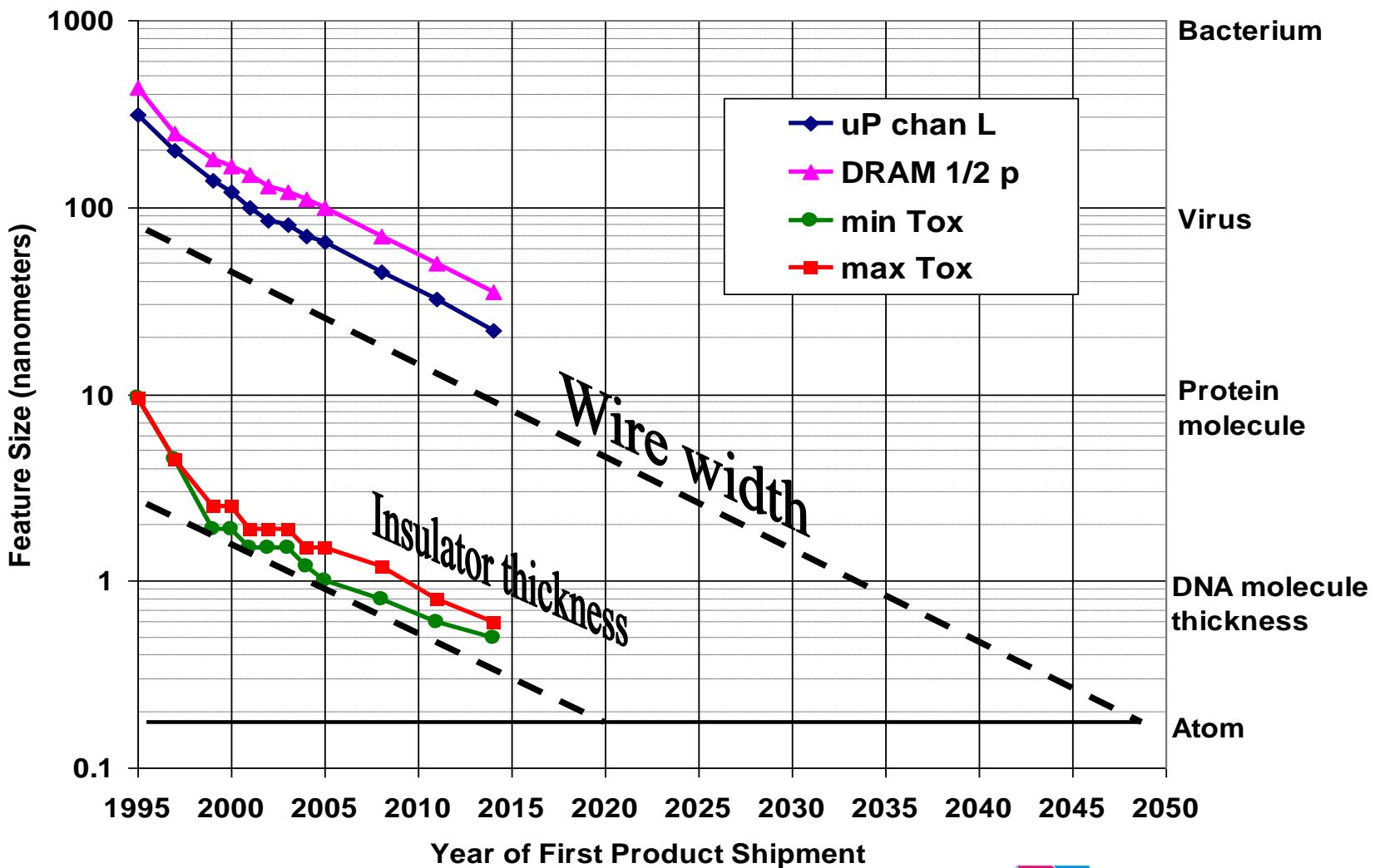


# 计算机实现技术的发展

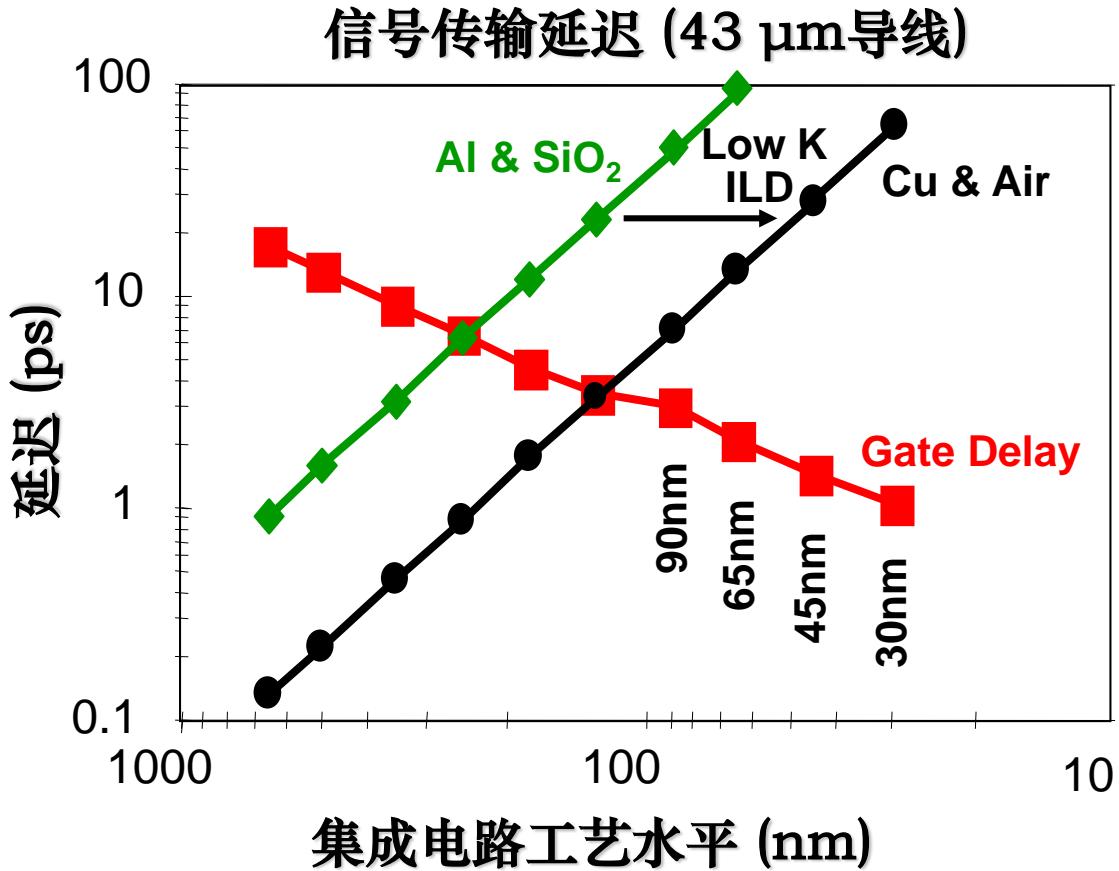
□磁盘。最近磁盘密度以每年约100%的速度增长，几乎每两年增长4倍。1990年以前磁盘密度每年增长30%，平均三年增长2倍。磁盘存取时间在过去10年内下降1/3。



# 面临的技术挑战



# 技术挑战与体系结构设计



- 导线变细，阻力变大
- 导线间隔变短，导致耦合电容，从而影响频率
- “线延迟墙”  
(Wire delay wall)



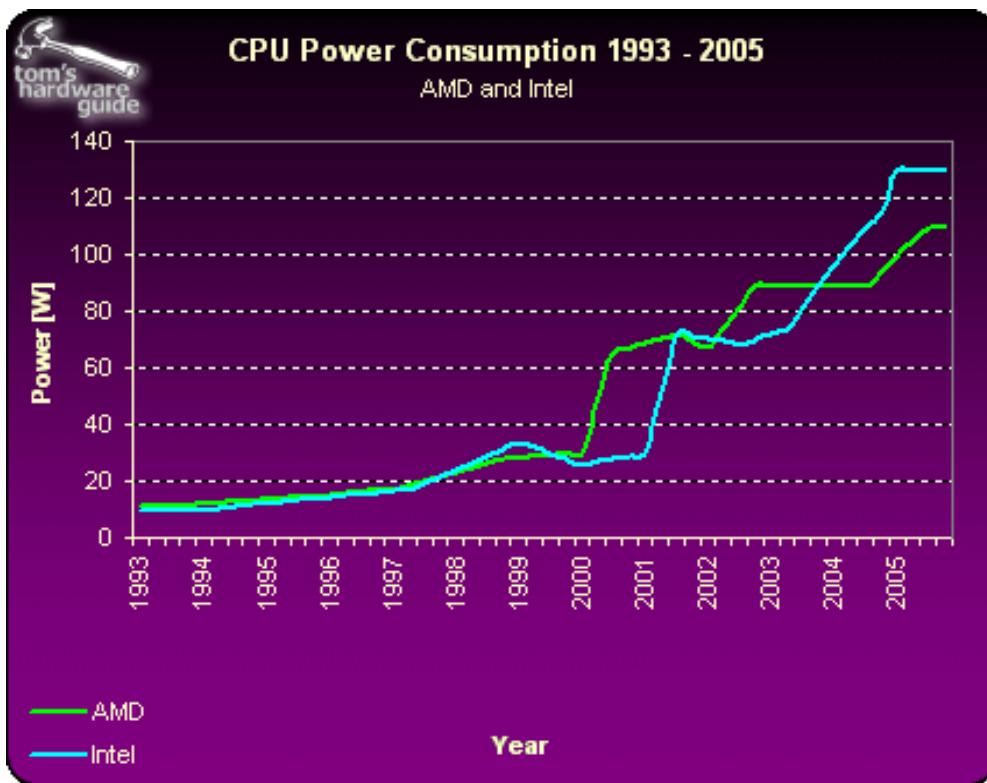


# 技术挑战与体系结构设计

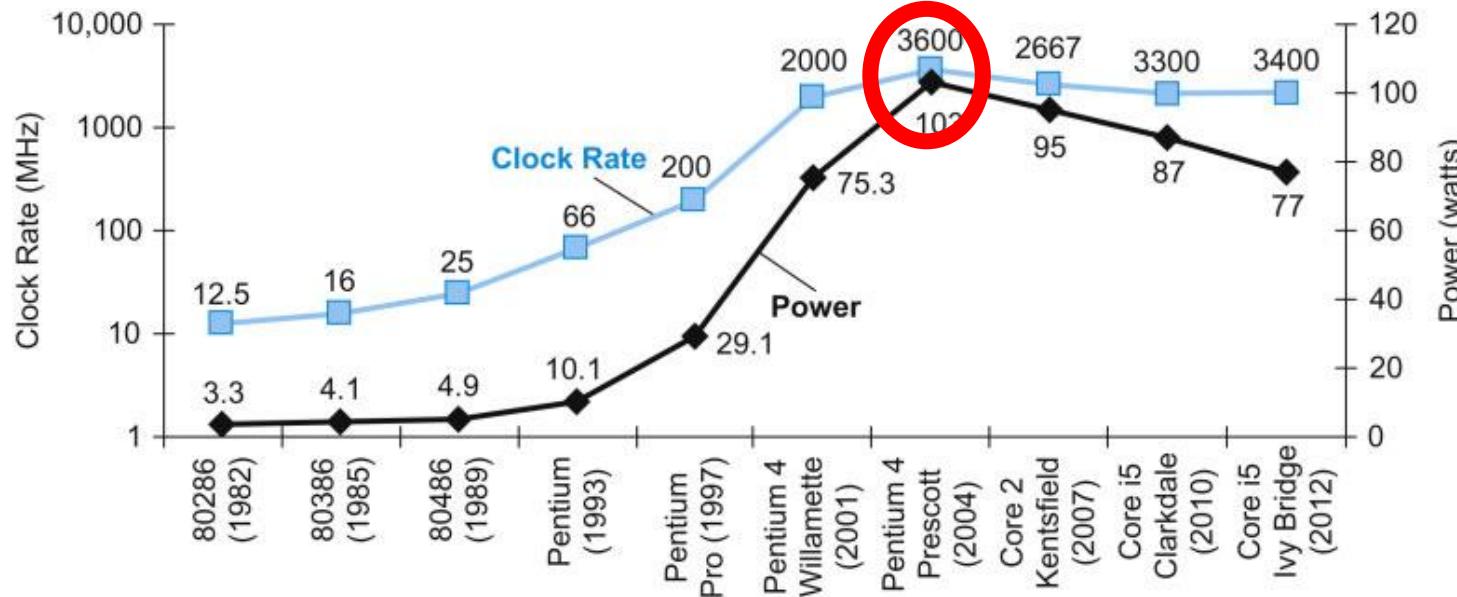
## □ 登纳德缩放定律 (Dennard Scaling) :

- ✓ 晶体管缩小，开启和关闭时所需的电压和电流也降低。
- ✓ 直到65nm，晶体管开始“泄漏”电子

## □ 功耗墙 (Power Wall) 频率墙 (Frequency Wall)



# 功耗墙



占统治地位的集成电路技术是 CMOS（互补型金属氧化半导体），其主要的功耗来源是动态功耗，即在晶体管开关过程中产生的功耗。动态功耗取决于每个晶体管的负载电容、工作电压和晶体管的开关频率：

$$\text{功耗} = \text{负载电容} \times \text{电压}^2 \times \text{开关频率}$$

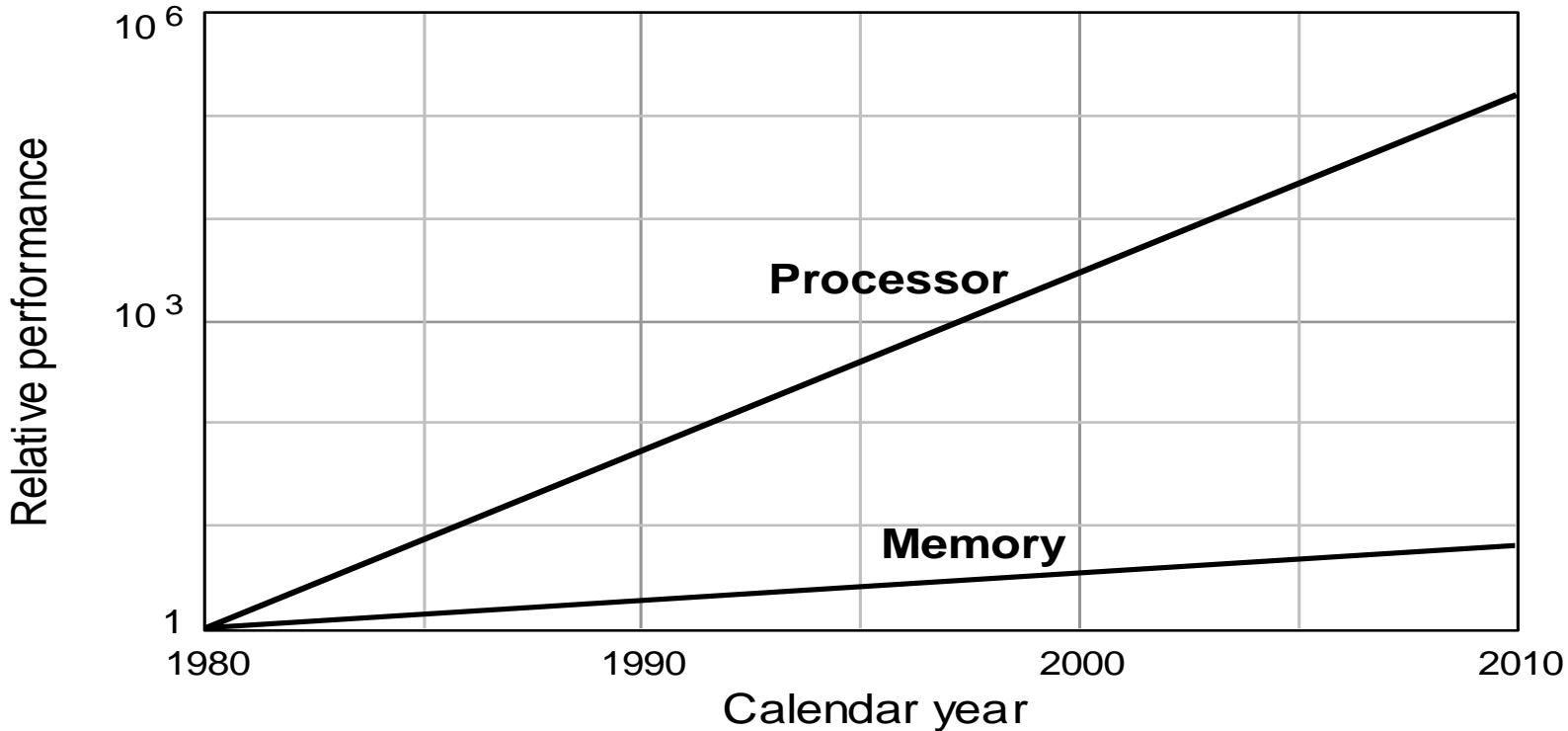
开关频率是时钟频率的函数，负载电容是连接到输出上的晶体管数量（称为扇出）和工艺的函数，该函数决定了导线和晶体管的电容。

为什么时钟频率增长为 1000 倍，而功耗只增长为 30 倍呢？因为功耗是电压平方的函数，能够通过降低电压来大幅减少，每次工艺更新换代时都会这样做。一般来说，每代的电压降低大约 15%。20 多年来，电压从 5 V 降到了 1 V。这就是功耗只增长为 30 倍的原因所在。

# 技术挑战与体系结构设计



中国科学技术大学  
University of Science and Technology of China

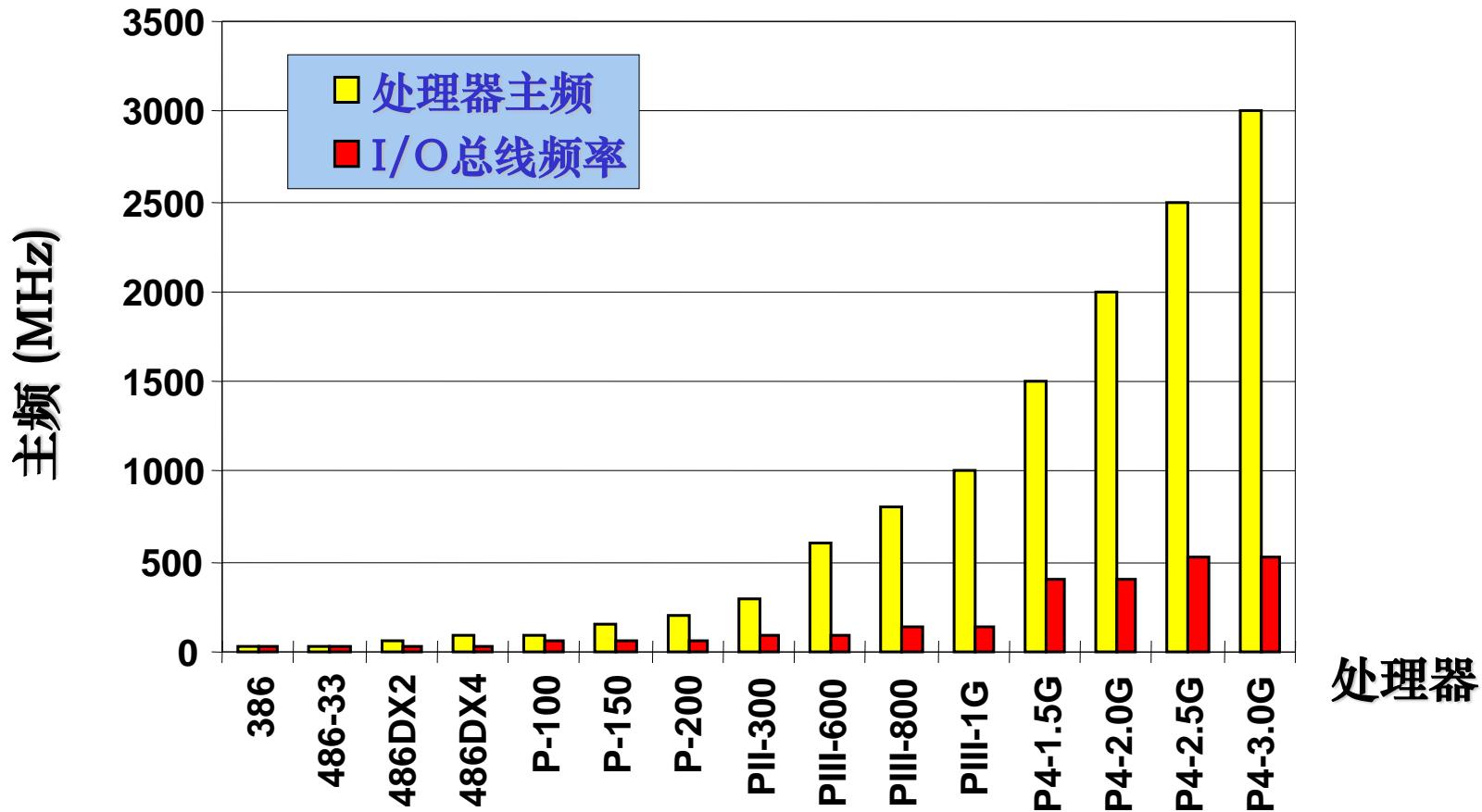


处理器性能与存储系统性能之间存在巨大差距  
(存储墙: Memory Wall)

# 技术挑战与体系结构设计



中国科学技术大学  
University of Science and Technology of China



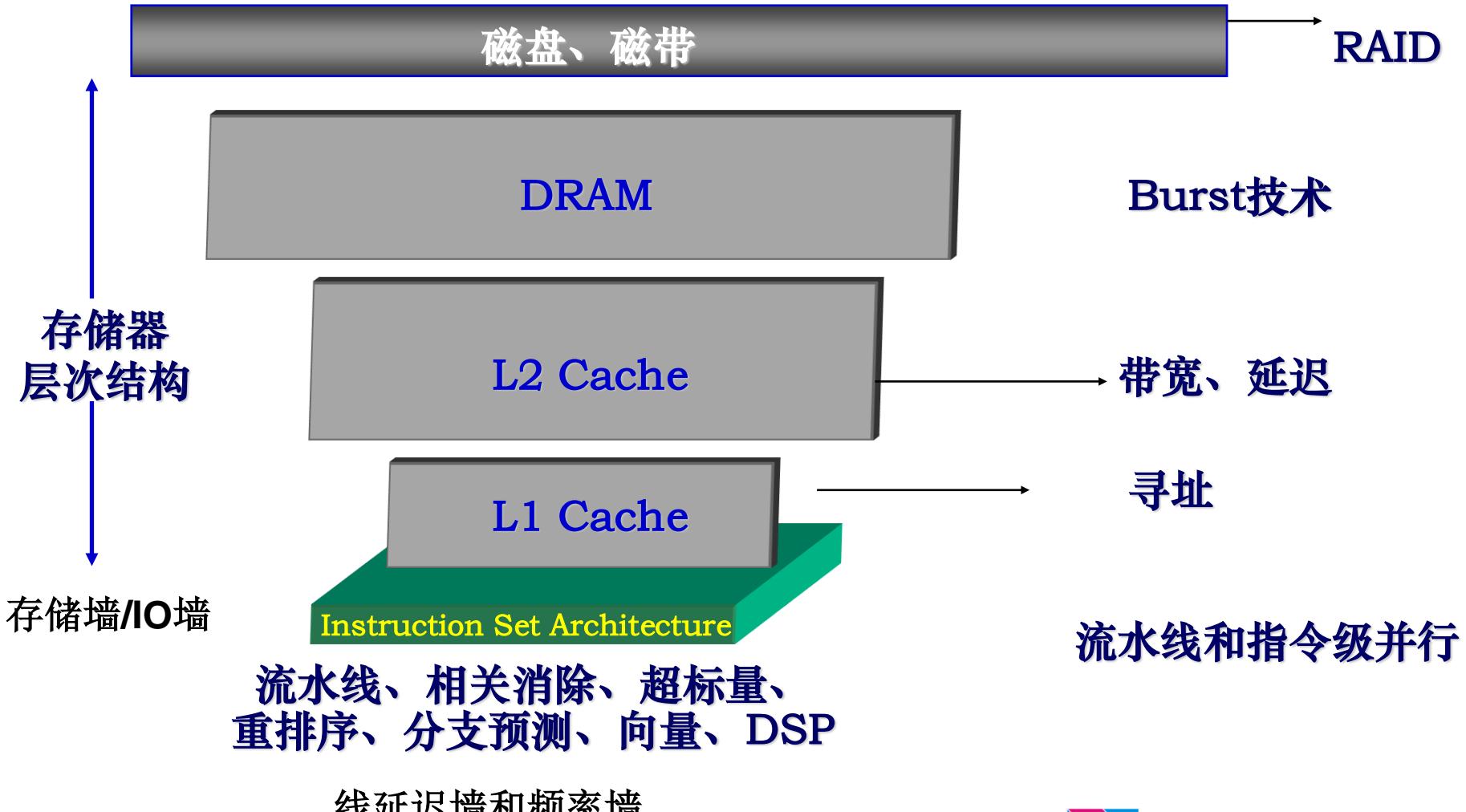
处理器的主频与I/O总线时钟频率之间存在巨大差距  
(**I/O墙: I/O wall**)



嵌入式系统实验室  
EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC



# 如何消除这些“墙”



# Amdahl 定律



中国科学技术大学  
University of Science and Technology of China

## 口加速比：

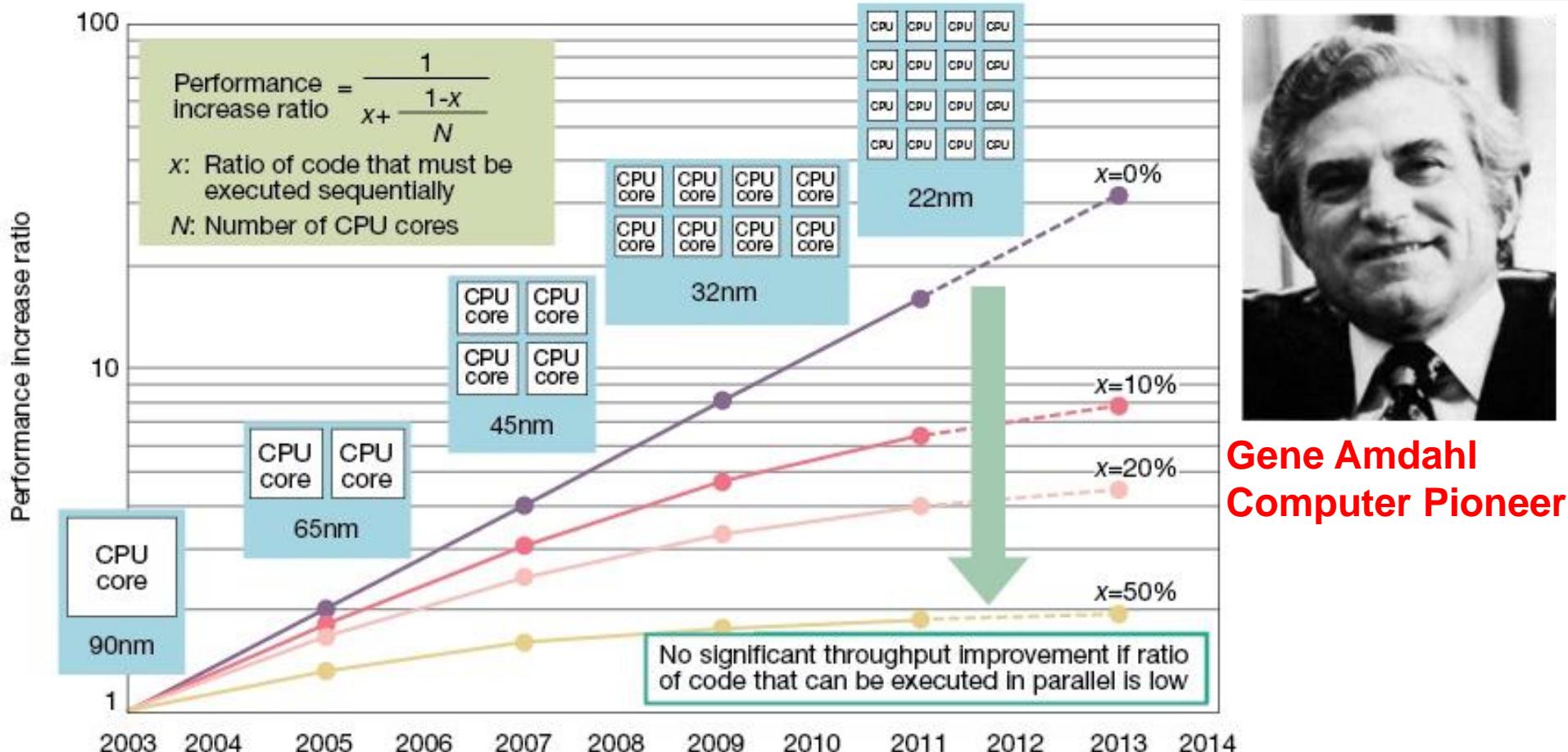
$$\text{系统加速比} = \frac{\text{系统性能}_{\text{改进后}}}{\text{系统性能}_{\text{改进前}}} = \frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}}$$



# 并行加速比: Amdahl's Law 1967



中国科学技术大学  
University of Science and Technology of China



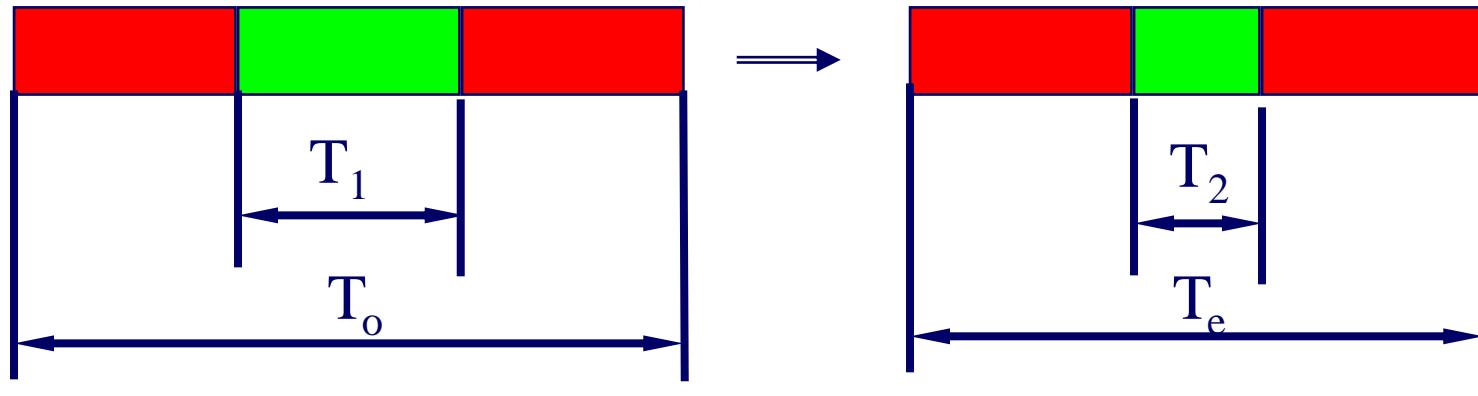
Gene Amdahl  
Computer Pioneer

**Fig 3 Amdahl's Law an Obstacle to Improved Performance** Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor performance. Diagram assumes no overhead in parallel processing. Years shown for design rules based on Intel planned and actual technology. Core count assumed to double for each rule generation.



# Amdahl 定律

- 系统加速比依赖于两个因素：
- “**可改进比例**”：可改进部分在原系统计算时间中所占的比例，它总是小于等于1的。
- “**部件加速比**” 可改进部分改进以后的性能提高，一般情况下它是大于1的。



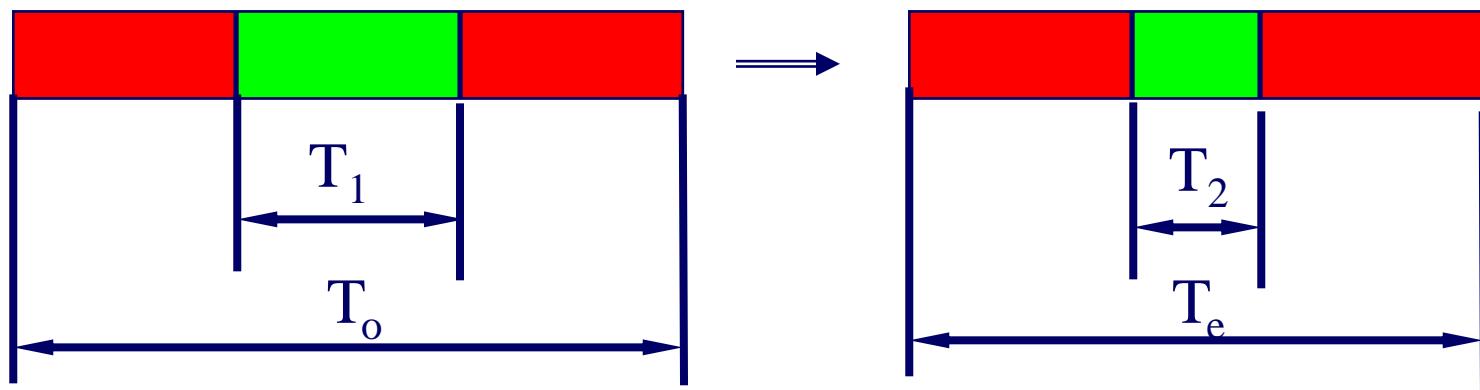


# Amdahl 定律

$$f_e = \frac{T_1}{T_o} \quad S_e = \frac{T_1}{T_2}$$

$$T_e = T_o \left[ (1 - f_e) + \frac{f_e}{S_e} \right]$$

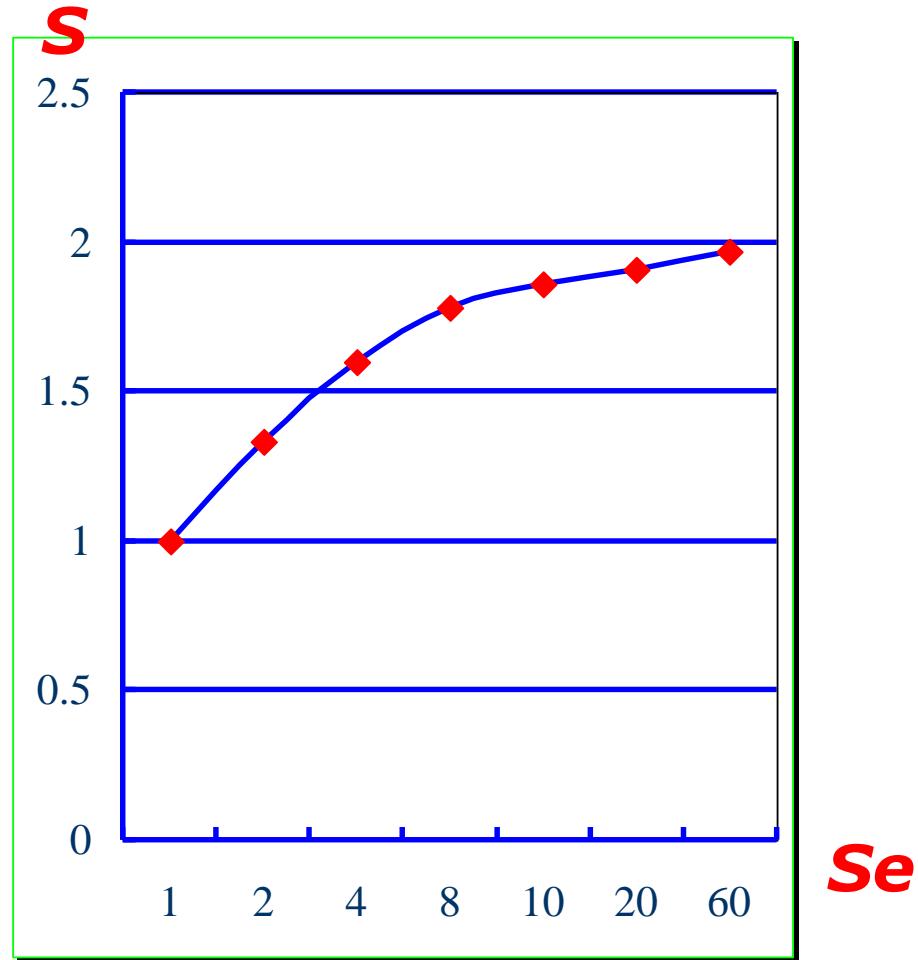
$$S = \frac{1}{(1 - f_e) + \frac{f_e}{S_e}}$$





□ 如果仅仅对计算机中的一部分做性能改进，则改进越多，系统获得的效果越小。

□ **推论：**如果只针对整个任务的一部分进行优化，那么所获得的加速比不大于 $1/(1-f_e)$ 。





例1：假设在某程序的执行过程中，浮点操作时间占整个执行时间的10%，现希望对浮点操作加速。

- ✓ 设对浮点操作的加速比为 $S_f$ 。请画出程序总的加速比 $S$ 和 $S_f$ 之间的关系曲线；
- ✓ 请问程序的最大加速比可达多少？

# Amdahl 定律练习



中国科学技术大学  
University of Science and Technology of China



$$S_{\max} = \lim_{S_f \rightarrow \infty} \frac{1}{0.9 + \frac{0.1}{S_f}} = 10/9$$

$$\begin{aligned} S &= \frac{1}{(1 - f_f) + \frac{f_f}{S_f}} \\ &= \frac{1}{(1 - 10\%) + \frac{10\%}{S_f}} \\ &= \frac{1}{0.9 + \frac{0.1}{S_f}} \end{aligned}$$



嵌入式系统实验室  
EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC

# Amdahl 定律练习



中国科学技术大学  
University of Science and Technology of China

例2：求平方根和浮点乘是图形应用中两种常用的转换。假设浮点操作在某机器的一个基准程序中占总执行时间的 $50\%$ ，求平方根操作在总执行时间的 $20\%$ ，现通过两种方法加速操作：

- ①增加专门的软件加速库处理求平方根，使其执行速度为原来的 $10$ 倍；
- ②通过调整流水线把浮点速度提高为原来的 $1.6$ 倍。

问：分别采用两种方法增强后此基准程序加速比各是多少？

# Amdahl 定律练习



中国科学技术大学  
University of Science and Technology of China

□ (1)

$$S = \frac{1}{(1 - f_e) + \frac{f_e}{S_e}}$$

$$= \frac{1}{(1 - 0.2) + \frac{0.2}{10}}$$

$$= 1.22$$

□ (2)

$$S = \frac{f_e}{(1 - f_e) + \frac{f_e}{S_e}}$$

$$= \frac{1}{(1 - 0.5) + \frac{0.5}{1.6}}$$

$$= 1.23$$



嵌入式系统实验室  
EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC



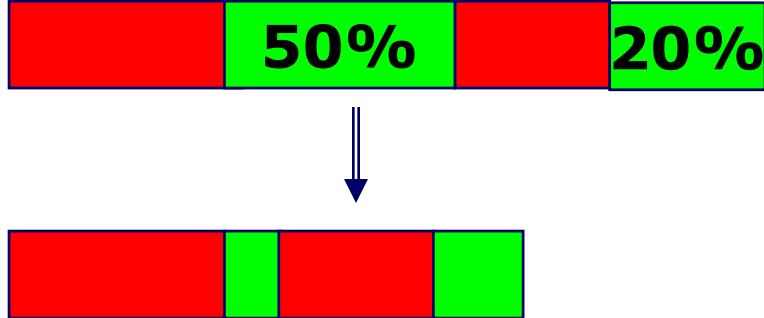
**例3：**求平方根和浮点乘是图形应用中两种常用的转换，假设求平方根操作在某机器的一个基准程序中占总执行时间的20%，浮点乘操作在该程序中占总执行时间50%，现通过两种方法加速两操作：①增加专门的软件函数加速库求平方根，使其执行速度为原来的10倍；②通过调整流水线把浮点乘速度提高为原来的1.6倍。

问：同时采用两种方法增强后此基准程序加速比是多少？

# Amdahl 定律练习



中国科学技术大学  
University of Science and Technology of China



加速比 = 增强前时间 / 增强后时间

$$= 1 / ((1 - 0.5 - 0.2) + 0.2 / 10 + 0.5 / 1.6)$$

$$= 1 / (0.3 + 0.2 / 10 + 0.5 / 1.6))$$

$$= 1 / 0.6325$$

$$= 1.58$$

# Amdahl 定律练习



中国科学技术大学  
University of Science and Technology of China

例4：求平方根、浮点乘和定点乘是图形应用中三种常用的转换。假设浮点操作在某机器的一个基准程序中占总执行时间的30%，求平方根操作在总执行时间的20%，定点乘15%。现通过三种方法加速操作：

- ①增加专门的硬件处理求平方根，使其执行速度为原来的10倍；
- ②通过调整流水线把浮点速度提高为原来的1.6倍。
- ③通过调整流水线把定点速度提高为原来的3倍。

问：分别采用三种方法增强后此基准程序加速比各是多少？

同时采用三种方法增强后此基准程序加速比各是多少？

# Amdahl 定律练习



中国科学技术大学  
University of Science and Technology of China

□ (1) 平方根,

$f_e = 0.2, S_e = 10$

$$S = \frac{1}{(1 - f_e) + \frac{f_e}{S_e}}$$

$$= \frac{1}{(1 - 0.2) + \frac{0.2}{10}}$$

$$= 1.22$$

□ (2) 浮点,  $f_e = 0.3, S_e = 1.6$

$$S = \frac{1}{(1 - f_e) + \frac{f_e}{S_e}}$$

$$= \frac{1}{(1 - 0.3) + \frac{0.3}{1.6}}$$

$$= 1.13$$



嵌入式系统实验室  
EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC

# Amdahl 定律练习



中国科学技术大学  
University of Science and Technology of China

## □ (4) 三种方法同时使用

$$S = \frac{1}{(1-f_e) + \frac{f_e}{S_e}}$$
$$= \frac{1}{(1-0.15) + \frac{0.15}{3}}$$
$$= 1.11$$

$$S = \frac{1}{(1-f_e) + \frac{f_e}{S_e}}$$
$$= \frac{1}{(1-0.3-0.2-0.15) + \frac{0.3}{1.6} + \frac{0.2}{10} + \frac{0.15}{3}}$$
$$= 1.65$$

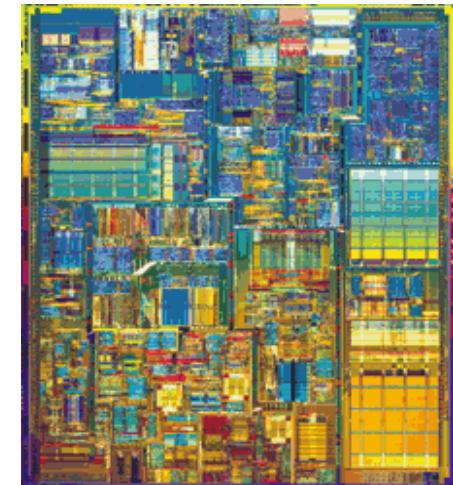
# 集成电路的生产过程



中国科学技术大学  
University of Science and Technology of China

## □ 芯片制作：

- ✓ 版图设计
- ✓ 备片
- ✓ 光刻成像
- ✓ 掺杂、金属化、钝化
- ✓ 切割
- ✓ 测试
- ✓ 封装



# 集成电路的生产过程



中国科学技术大学  
University of Science and Technology of China

## 备片



打磨后的硅片



在硅锭上切割出  
硅片



嵌入式系统实验室

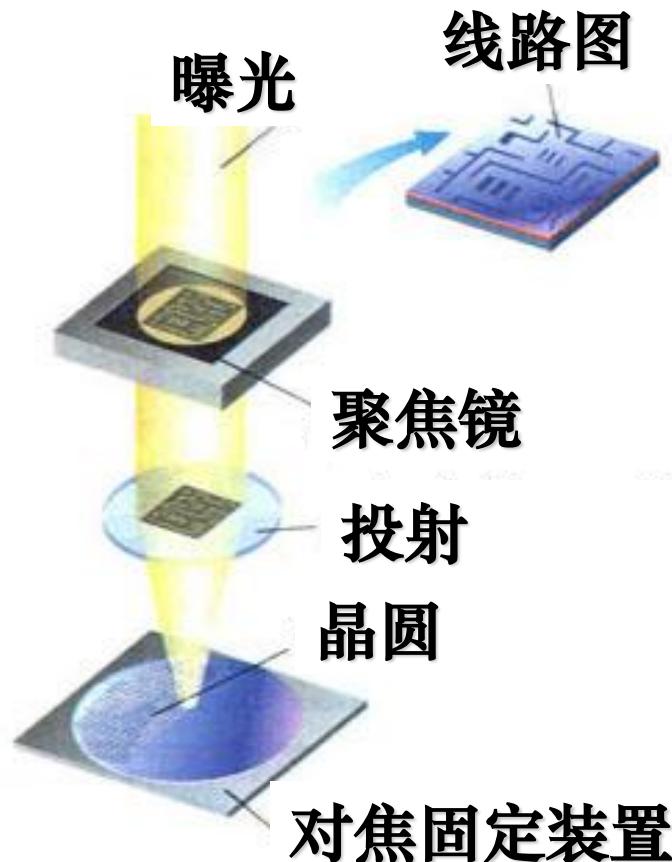
EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC



# 集成电路的生产过程

## 口光刻成像

- ✓ 首先用光刻胶涂抹器在硅片的表面涂上一层光刻胶，然后用烘烤/蒸发器使光刻胶变硬。一旦完成上述过程，用硅片曝光器将光刻胶曝光，然后用显影器移走被曝光的部分，使得曝光部分的表面变低。

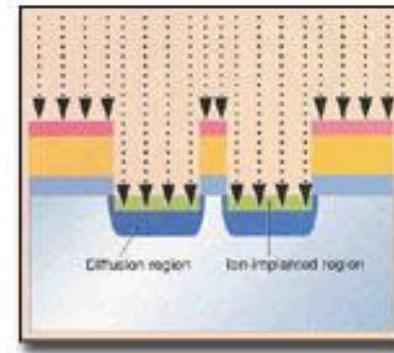




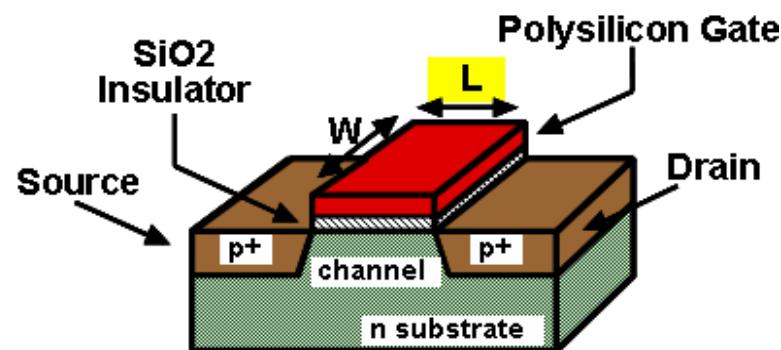
# 集成电路的生产过程

## 口离子注入和扩散（掺杂）

- ✓ 根据集成电路设计的结构在曝光的部分注入具有特定物理特征的离子，改变硅片各个区域的电气特性。形成相应的晶体管、电阻、电容等。



## 口绝缘体沉积和金属化、钝化

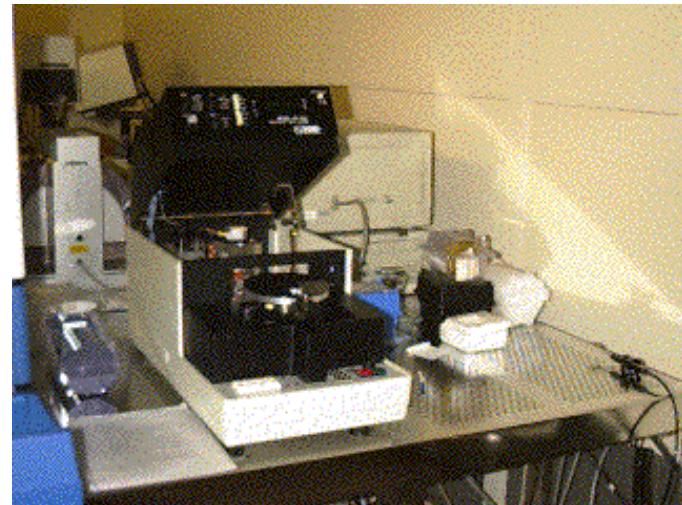
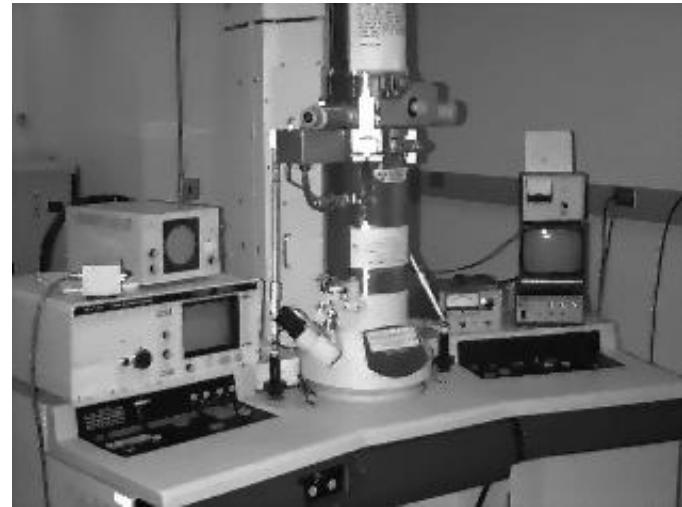
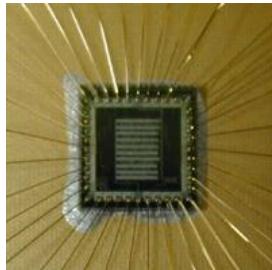
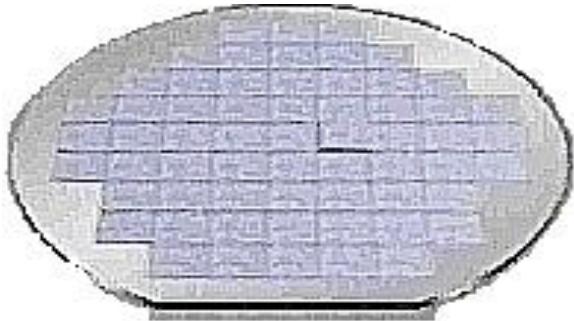


# 集成电路的生产过程



中国科学技术大学  
University of Science and Technology of China

- 测试
- 切割
- 封装



嵌入式系统实验室  
EMBEDDED SYSTEM LABORATORY  
SUZHOU INSTITUTE FOR ADVANCED STUDY OF USTC

# AMD芯片的生产过程



中国科学技术大学  
University of Science and Technology of China



**Silicon, Wafer, Mask, Transistor, Semi-Conductor,  
Lattice, Integrated Circuits, Copper-wire**

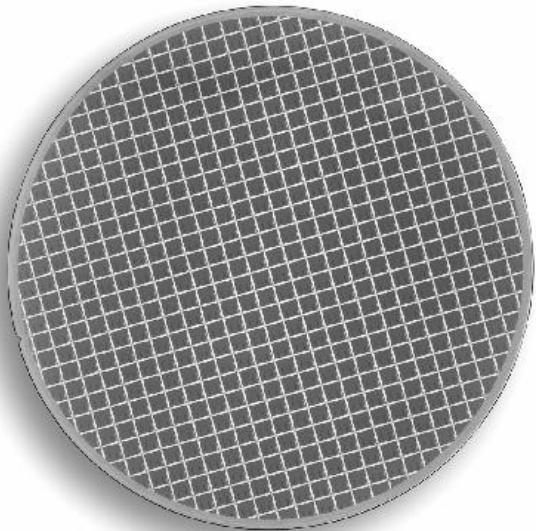


从沙子到芯片



# 集成电路的成本

$$\text{集成电路成本} = \frac{\text{晶片成本} + \text{晶片测试成本} + \text{封装成本}}{\text{最终成品率}}$$



晶片成本 =

$$\frac{\text{晶圆成本}}{\text{每块晶圆上的晶片数} \times \text{晶片成品率}}$$

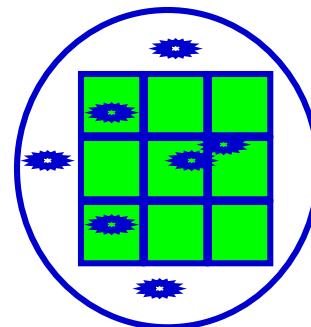
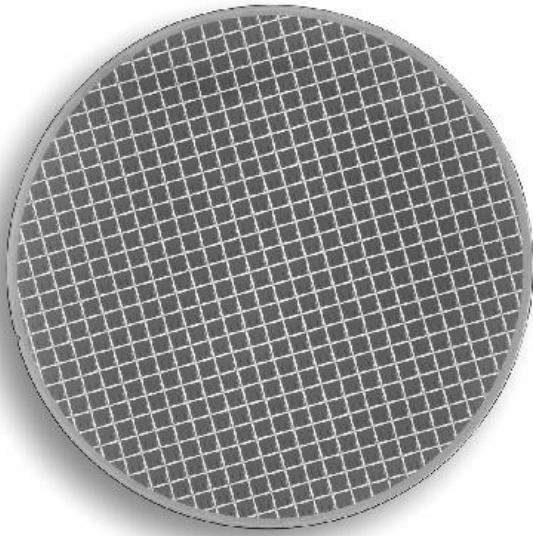
$$\text{每块晶圆上的晶片数} = \frac{\text{晶圆面积}}{\text{晶片面积}} - \frac{\pi \times \text{晶圆直径}}{\sqrt{2} \times \text{晶片面积}}$$



# 集成电路的成本

$$\text{晶片成本} = \frac{\text{晶圆成本}}{\text{每块晶圆上的晶片数} \times \text{晶片成品率}}$$

$$\text{晶片成品率} = \text{晶圆成品率} \times \left(1 + \frac{\text{疵点密度} \times \text{晶片面积}}{\alpha}\right)^{-\alpha}$$



$$\alpha = 3 \sim 4$$



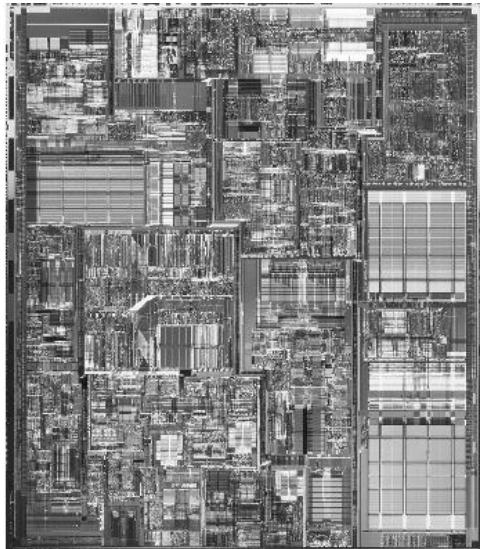


# 集成电路的成本

晶片成本 =

晶圆成本

$$\frac{\left( \frac{\text{晶圆面积}}{\text{晶片面积}} - \frac{\pi \times \text{晶圆直径}}{\sqrt{2} \times \text{晶片面积}} \right) \times \text{晶圆成品率} \times \left( 1 + \frac{\text{疵点密度} \times \text{晶片面积}}{\alpha} \right)^{-\alpha}}$$



$$\text{晶片成本} = f(\text{晶片面积}^{\alpha+1})$$



# 芯片的成本

## □ 单片集成电路成本

### □ NRE 费用 (Non-Recurring Engineering cost)

- ✓ 一次性投片费用

## □ 总成本

- ✓ NRE 费用+单片成本\*芯片数

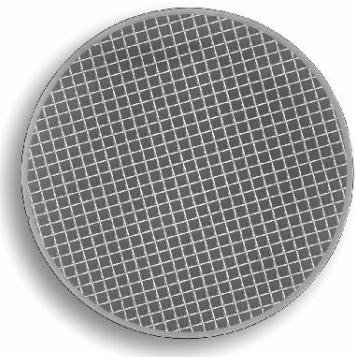
## □ 每个芯片的成本

- ✓ 总成本/芯片数= (NRE费用/芯片数) + 单片成本

- ✓ NRE=\$20000, 单片集成电路成本=\$100

- ✓ 10片

- 总成本 = \$20000 + 10\*\$100 = \$21000
- 每个芯片成本 = \$20000/10 + \$100 = \$2100





口假设晶圆的成品率为100%，晶片为长为1.5cm的正方形。（1）直径为30cm的晶圆上有多少晶片？（2）假设单位面积的疵点密度为0.4/cm<sup>2</sup>，且 $\alpha=4.0$ ，求晶片的成品率？

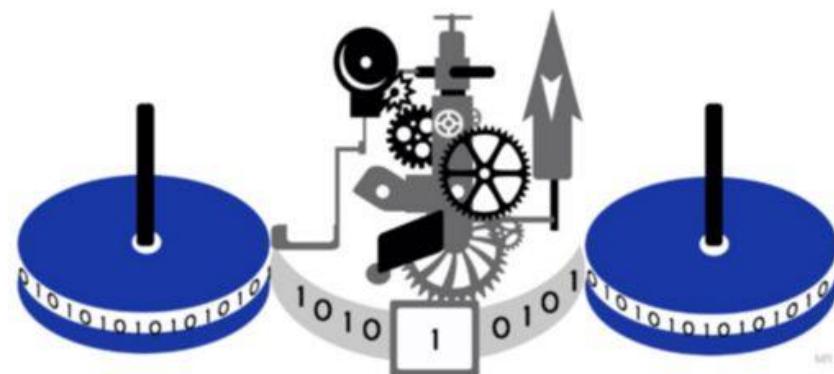
$$\text{每块晶圆上的晶片数} = \frac{\text{晶圆面积}}{\text{晶片面积}} - \frac{\pi \times \text{晶圆直径}}{\sqrt{2} \times \text{晶片面积}}$$

$$\text{晶片成品率} = \text{晶圆成品率} \times \left(1 + \frac{\text{疵点密度} \times \text{晶片面积}}{\alpha}\right)^{-\alpha}$$



# 小结

- 计算机系统由软件和硬件组成
- 计算机系统的层次化结构
- 计算机组装 vs. 计算机体系结构
- Von Neumann机
- 指令的执行过程
- 硬件系统技术指标
- 性能评测方法



# 作业

□ COD4: 1.14, 1.16

□ 下周五交

## Exercise 1.14

Section 1.8 cites as a pitfall the utilization of a subset of the performance equation as a performance metric. To illustrate this, consider the following data for the execution of given instruction sequence of  $10^6$  instructions in different processors.

Processor	Clock rate	CPI
P1	4 GHz	1.25
P2	3 GHz	0.75

**1.14.1** [5] <1.8> One usual fallacy is to consider the computer with the largest clock rate as having the large performance. Check if this is true for P1 and P2.

**1.14.2** [10] <1.8> Another fallacy is to consider that the processor executing the largest number of instruction will need a larger CPU time. Considering that processor P1 is executing a sequence of  $10^6$  instructions and that the CPI of processors P1 and P2 do not change, determine the number of instructions that P2 can execute in the same time that P1 needs to execute  $10^6$  instructions.

**1.14.3** [10] <1.8> A common fallacy is to use MIPS (millions of instructions per second) to compare the performance of two different processors, and consider that the processor with the largest MIPS has the largest performance. Check if this is true for P1 and P2.

Another common performance figure is MFLOPS (million of floating-point operations per second), defined as

$$\text{MFLOPS} = \text{No. FP operations}/\text{execution time} \times 10^6$$

but this figure has the same problems as MIPS. Consider the programs in the following table, running on a processor with clock rate = 3 GHz.

	Instr. count	L/S Instr.	FP Instr.	Branch Instr.	CPI(L/S)	CPI(FP)	CPI(Branch)
a.	$10^6$	50%	40%	10%	0.75	1	1.5
b.	$3 \times 10^6$	40%	40%	20%	1.25	0.70	1.25

**1.14.4** [10] <1.8> Find the MFLOPS figures for the programs.

**1.14.5** [10] <1.8> Find the MIPS figures for the programs.

**1.14.6** [10] <1.8> Find the performance for the programs and compare with MIPS and MFLOPS.



## Exercise 1.16

Another pitfall, relating to the execution of programs in multiprocessor systems, is expecting improvement in performance by improving only the execution time of part of the routines. The following table shows the execution time of five routines of a program running on different numbers of processors.

# Processors	Routine A (ms)	Routine B (ms)	Routine C (ms)	Routine D (ms)	Routine E (ms)
a.	20	80	10	70	5
b.	4	14	2	12	2

**1.16.1** [10] <1.8> Find the total execution time and by how much it is reduced if the time of routines A, C, and E is improved by 15%.

**1.16.2** [10] <1.8> By how much is the total time reduced if routine B is improved by 10%?

**1.16.3** [10] <1.8> By how much is the total time reduced if routine D is improved by 10%?

Execution time in a multiprocessor system can be split into computing time for the routines plus routing time spent sending data from one processor to another. Consider the execution time and routing time given in the following table. In this case, the routing time is an important component of the total time.

# Processors	Routine A (ms)	Routine B (ms)	Routine C (ms)	Routine D (ms)	Routine E (ms)	Routing (ms)
2	20	78	9	65	4	11
4	12	44	4	34	2	13
8	1	23	3	19	3	17
16	4	13	1	10	2	22
32	2	5	1	5	1	23
64	1	3	0.5	1	1	26

**1.16.4** [10] <1.8> For each doubling of the number of processors, determine the ratio of new to old computing time and the ratio of new to old routing time.

**1.16.5** [5] <1.8> Using the geometric means of the ratios, extrapolate to find the computing time and routing time in a 128-processor system.

**1.16.6** [10] <1.8> Find the computing time and routing time for a system with one processor.

# 作业



## □C0D4: 1.14, 1.16

### 习题 1.14

1.8 节引证了一个用性能公式的一个子集去计算性能的陷阱。为了说明它，下表是在不同的处理器中执行 $10^6$  条指令序列的有关数据。

处理器	时钟频率	CPI
P1	4 GHz	1.25
P2	3 GHz	0.75

**1.14.1** [5] <1.8>一个常见的错误是，认为时钟频率最高的计算机具有最高的性能。这种说法正确吗？请用 P1 和 P2 来验证这一说法。

**1.14.2** [10] <1.8>另一个错误是，认为执行指令最多的处理器需要更多的 CPU 时间。考虑 P1 执行  $10^6$  条指令序列所需的时间，P1 和 P2 的 CPI 不变，计算一下 P2 用同样的时间可以执行多少条指令？

**1.14.3** [10] <1.8>一个常见的错误是用 MIPS（每秒百万条指令）来比较 2 台不同的处理器的性能，并认为 MIPS 最大的处理器具有最高的性能。这种说法正确吗？请用 P1 和 P2 验证这一说法。

另一个常见的性能标志是 MFLOPS（每秒百万条浮点指令），其定义为  $MFLOPS = \text{浮点操作的个数} / \text{执行时间} \times 10^6$ 。它与 MIPS 有同样的问题。考虑下表所示的程序，在时钟频率为 3 GHz 的处理器上运行。

	指令数	读/写	浮点	分支	CPI (读/写)	CPI (浮点)	CPI (分支)
程序 a	$10^6$	50%	40%	10%	0.75	1	1.5
程序 b	$3 \times 10^6$	40%	40%	20%	1.25	0.70	1.25

**1.14.4** [10] <1.8>求程序的 MFLOPS 值。

**1.14.5** [10] <1.8>求程序的 MIPS 值。

**1.14.6** [10] <1.8>求程序的性能，并与 MFLOPS 和 MIPS 值作比较。

# 作业



## □C0D4: 1.16

### 习题 1.16

还有一个易犯的错误是有关在多处理器系统中运行，希望只改进一部分例行程序来改进整体性能。下表表示某个程序的 5 个例程在不同数目处理器中的执行时间。

	处理器数	例程 A (ms)	例程 B (ms)	例程 C (ms)	例程 D (ms)	例程 E (ms)
a.	2	20	80	10	70	5
b.	16	4	14	2	12	2

1.16.1 [10] <1.8> 求总的执行时间。如果例程 A, C 和 E 的时间改进 15%，总的执行时间能减少多少？

1.16.2 [10] <1.8> 如果例程 B 的时间改进 10%，总的执行时间能减少多少？

1.16.3 [10] <1.8> 如果例程 D 的时间改进 10%，总的执行时间能减少多少？

多处理器系统中的执行时间可分成例程计算时间加处理器之间的通信时间。下表给出了例程计算时间和通信时间。在这种情况下，通信时间是总时间的重要组成部分。

处理器数	例程 A (ms)	例程 B (ms)	例程 C (ms)	例程 D (ms)	例程 E (ms)	通信 (ms)
2	20	78	9	65	4	11
4	12	44	4	34	2	13
8	1	23	3	19	3	17
16	4	13	1	10	2	22
32	2	5	1	5	1	23
64	1	3	0.5	1	1	26

1.16.4 [10] <1.8> 每当处理器数量加倍时，求新的计算时间与旧的计算时间之比和新与旧的通信时间之比。

1.16.5 [5] <1.8> 用比值的几何平均值，推算在 128 台处理器的系统中的计算时间和通信时间。

1.16.6 [10] <1.8> 求在 1 个处理器系统中的计算时间和通信时间。

# 作业



中国科学技术大学  
University of Science and Technology of China

## 口思考（不交）

- ✓ 指令和数据都存储于存储器中，计算机如何区分它们？
- ✓ 综述计算机技术的发展历程及热点问题
- ✓ 最新Intel处理器的性能指标？
- ✓ 计算机的开机过程？
- ✓ C语言计算机模型？
- ✓ getchar() 的实现过程？
- ✓ PC系统活动与性能分析
  - 跑一个Benchmark，给出结果？
    - 每次执行同一个程序，结果都一样？
  - 主频与计算性能的关系？



# 联系方式

课程主页：<http://staff.ustc.edu.cn/~cswang>

王超

中国科学技术大学计算机学院  
嵌入式系统实验室（西活科住物业对面）  
高效能智能计算实验室

主要研究方向：

- 基于分布式系统, **GPU**, **FPGA**的神经网络、图计算加速
- 人工智能和深度学习（寒武纪）芯片及智能计算机

课程问题、研究方向问题、实验室问题 欢迎大家邮件

[cswang@ustc.edu.cn](mailto:cswang@ustc.edu.cn)

<http://staff.ustc.edu.cn/~cswang>

Thank you