计算机系统结构

第二课：系统结构的基本概念

王韬

wangtao@pku.edu.cn

http://ceca.pku.edu.cn/wangtao

2018

第二课：系统结构的基本概念

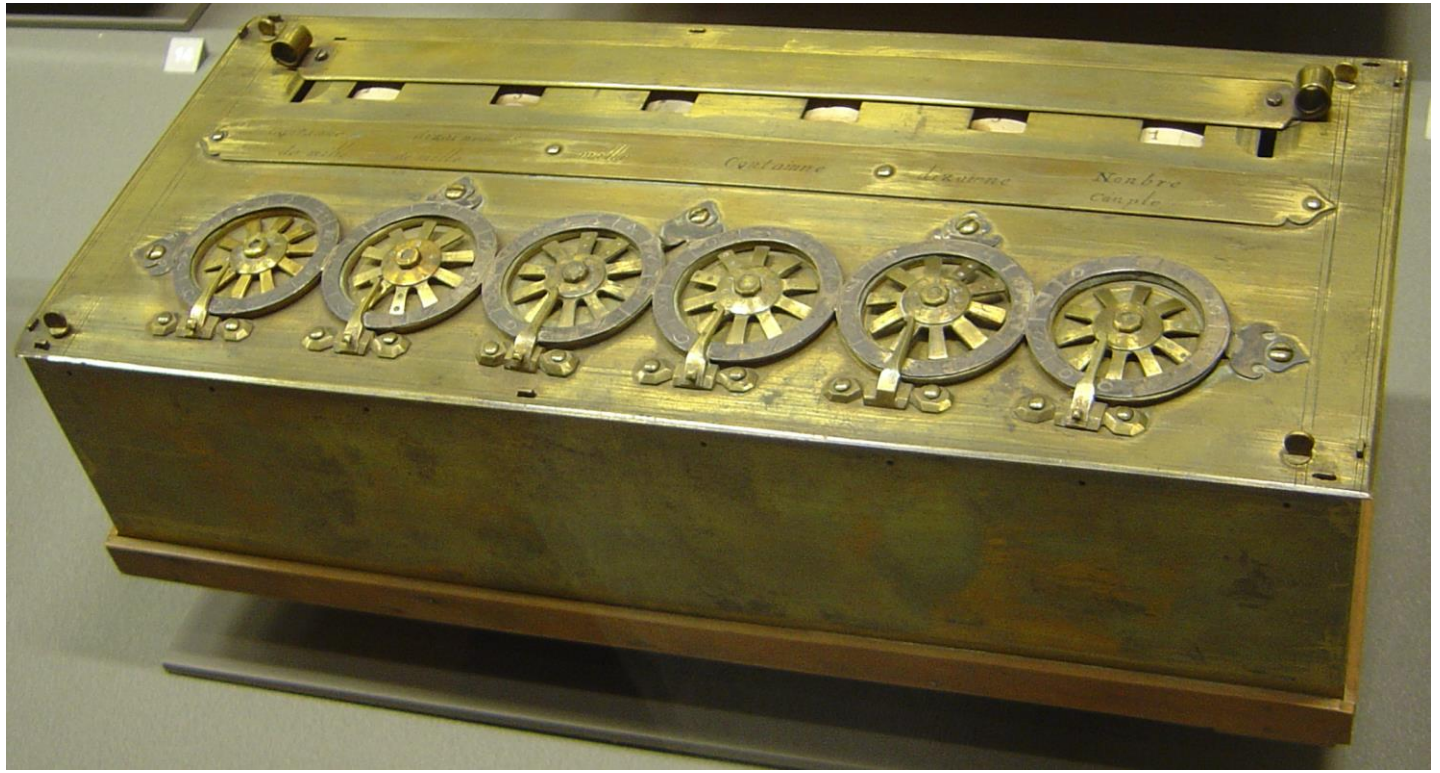
- 从计算电路到冯·诺依曼结构计算机
- 数据与公式
- 若干其它概念

第二课：系统结构的基本概念

- 从计算电路到冯·诺依曼结构计算机
- 数据与公式
- 若干其它概念

思考

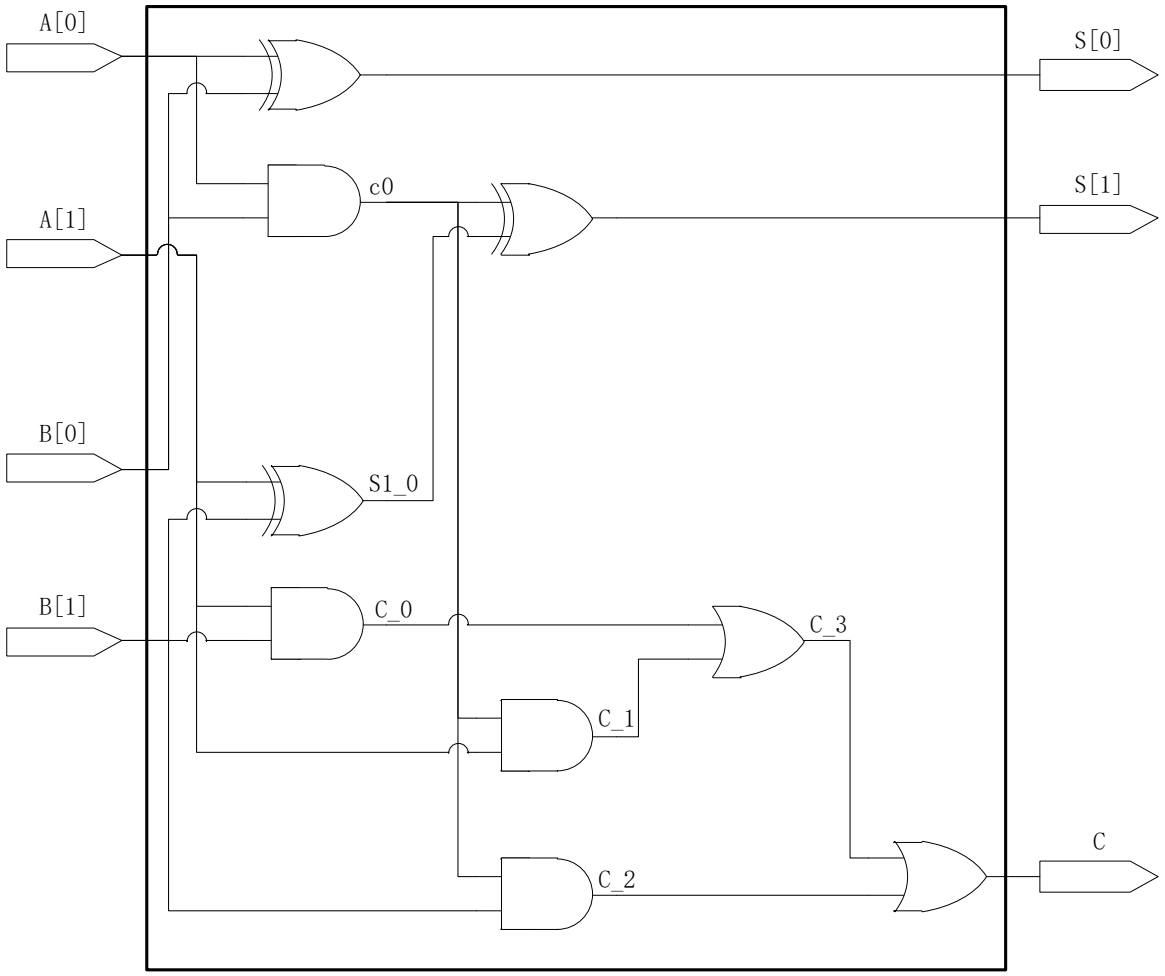
- 计算机为什么能用来做计算?



An early Pascaline on display at the Musée des Arts et Métiers, Paris

计算电路

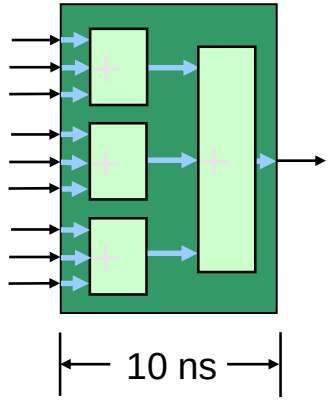
- 2-bit full adder:
  S = A + B

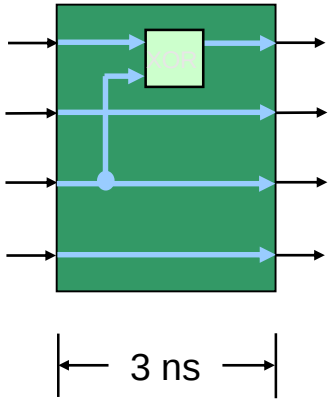


组合逻辑

- 输出是输入的函数
  - 输入如果在任何时刻发生变化，一段特定时间后，输出（可能）发生相应变化，且变化后的值是输入的函数；这段时间叫做延迟（latency）
- 问题：计算机是否是组合逻辑？
- 能否用组合逻辑来实现累加器？

$$S \leftarrow S + x$$

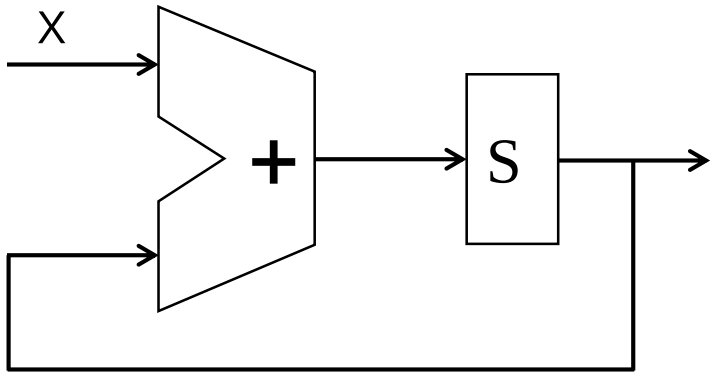




时序逻辑

- 为何需要时序逻辑?
- 如何来实现累加器?

$$S \leftarrow S + x$$



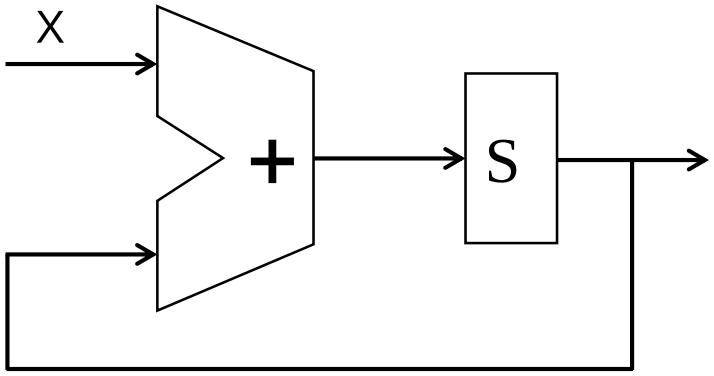
“一步步地计算”

$S \leftarrow S + x$

- 在计算机中自然的计算模型

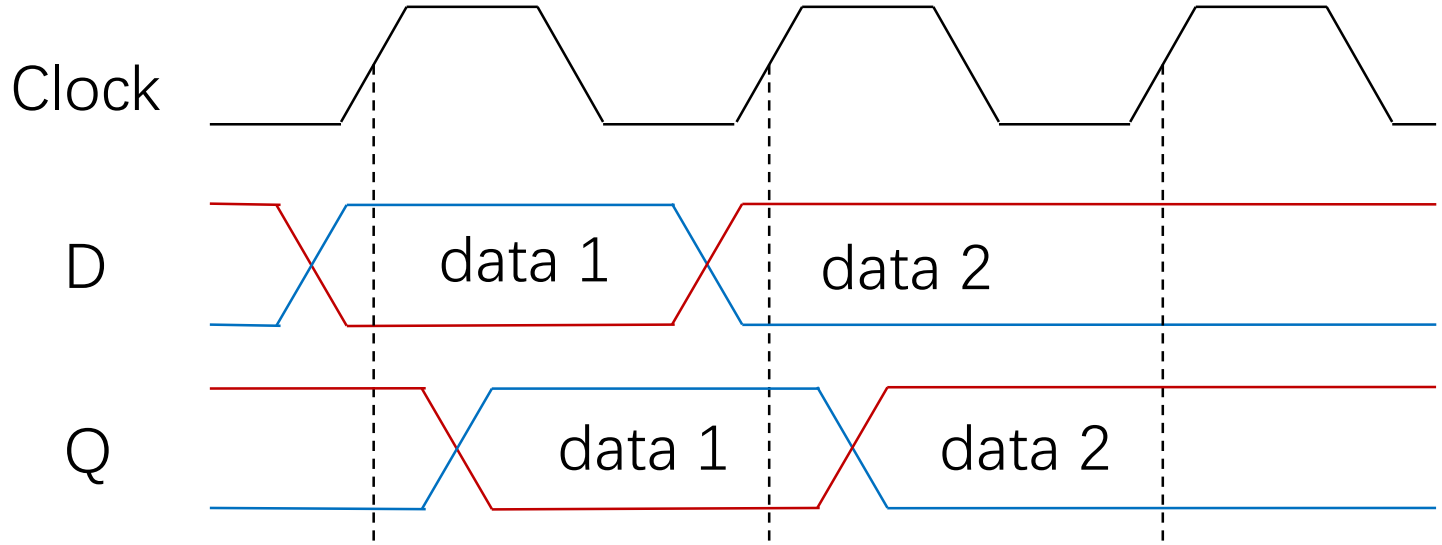
$$
\begin{aligned}
S_0 &= 0; \\
S_1 &= S_0 + x_0 \\
S_2 &= S_1 + x_1 \\
S_3 &= S_2 + x_2 \\
&\cdots \\
S_{n+1} &= S_n + x_n
\end{aligned}
$$

- 能够在一个电路中，让信号一步步地传播么？



保存“上一步”的状态

- D触发器（D flip-flop）：保存住上个周期末时输入的数据

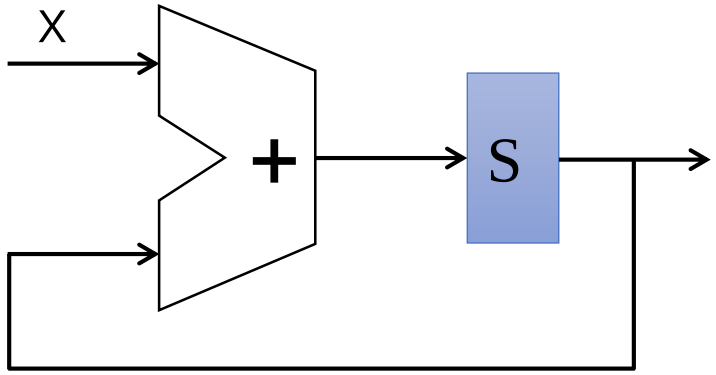


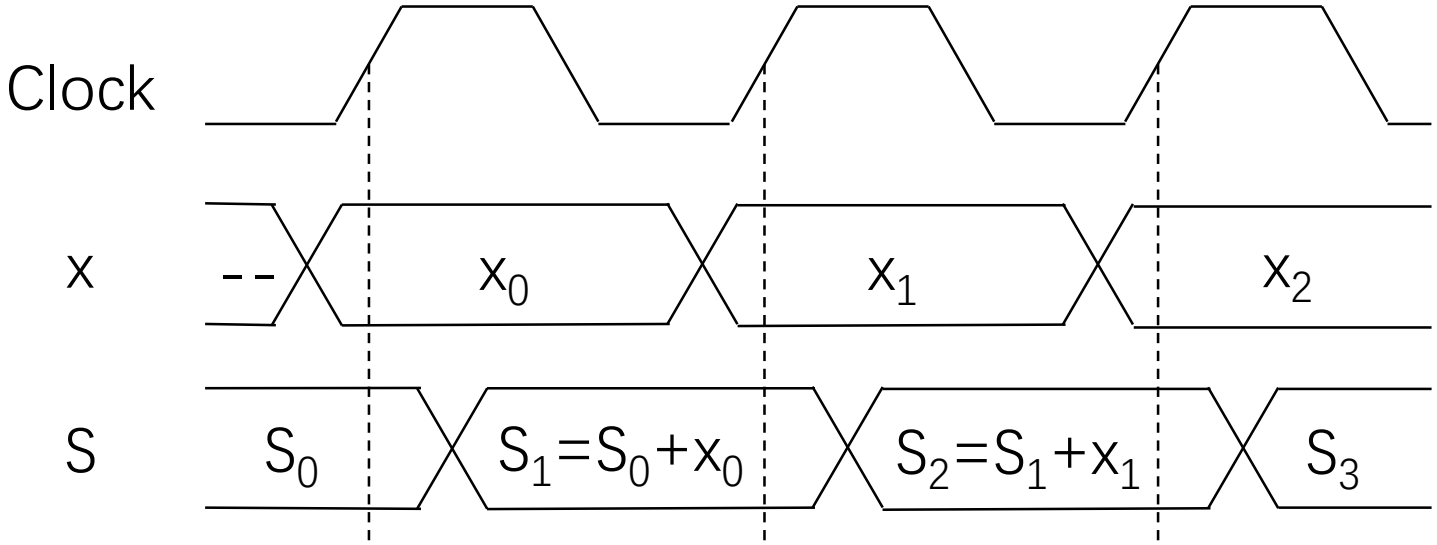
- 寄存器：多个D触发器并联起来，可以同时存住多位数据

时序逻辑构造累加器：加法器 + 寄存器

$$S \leftarrow S + x$$

$S_0 = 0;$
$S_1 = S_0 + x_0$
$S_2 = S_1 + x_1$
$S_3 = S_2 + x_2$
...
$S_{n+1} = S_n + x_n$

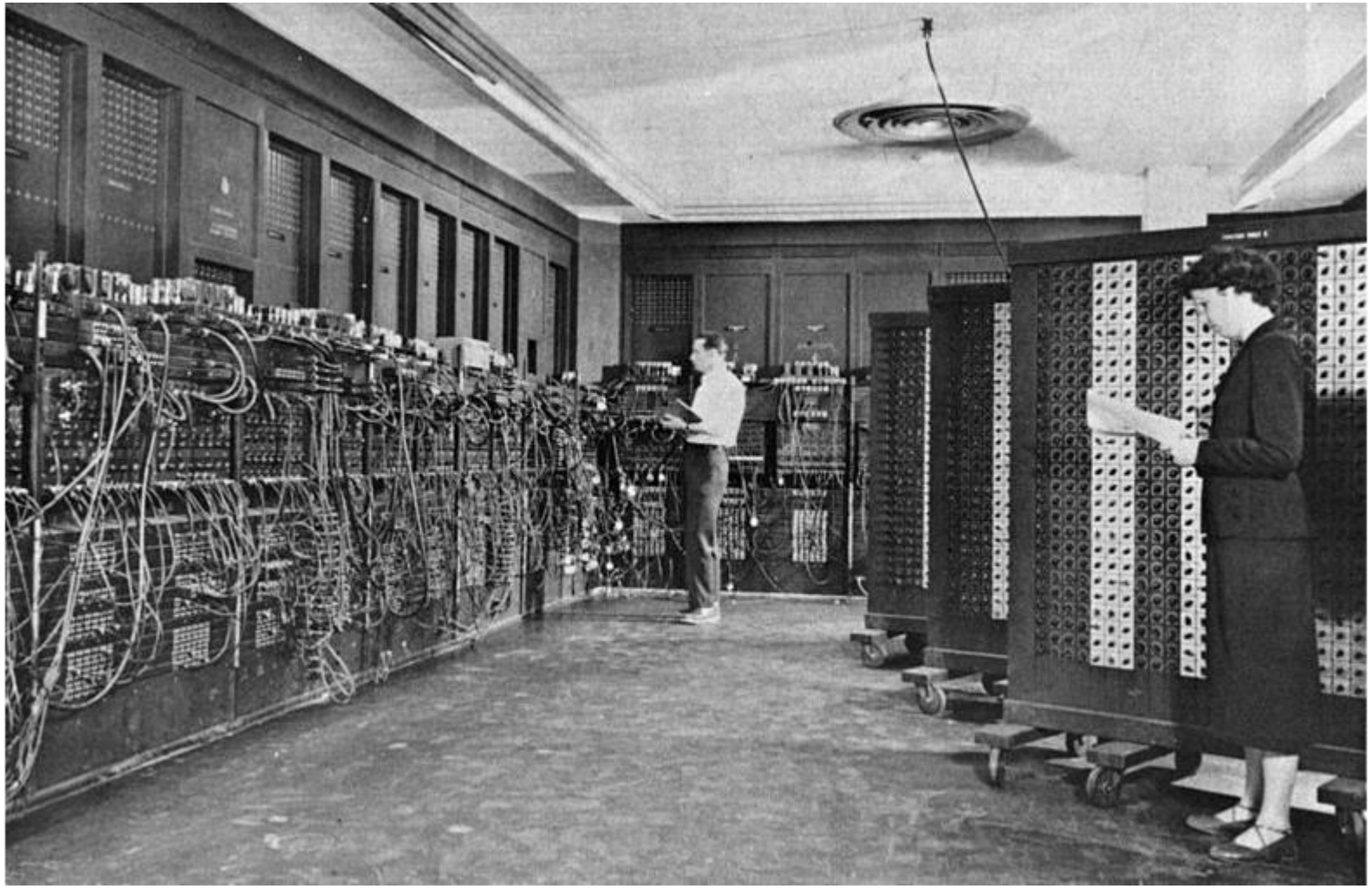




时序逻辑与计算机

- 问题：计算机是否是时序逻辑?
- 普通的时序逻辑 v.s. 计算机?
  - 最初的ENIAC
  - 冯·诺依曼体系结构
- 如何让计算机“编程”更加方便?

第一台电子计算机



Glen Beck (background) and Betty Snyder (foreground) program ENIAC in BRL building 328. (U.S. Army photo)

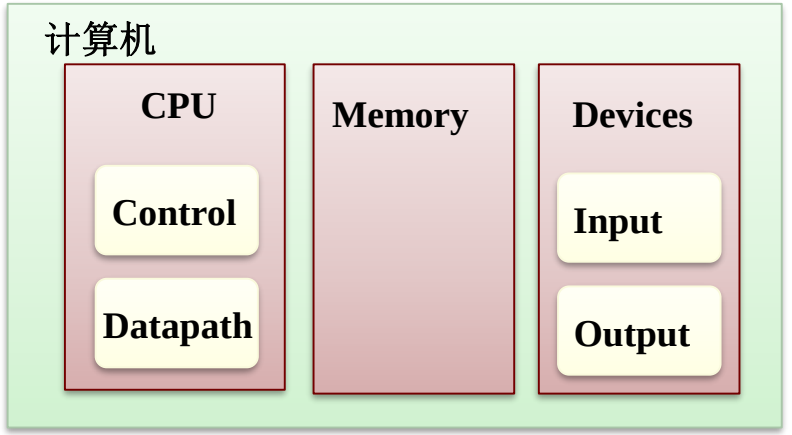
冯·诺依曼体系结构 1/2

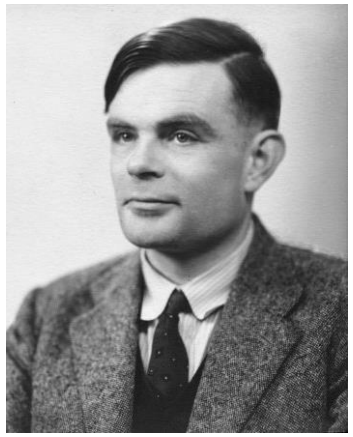
- 一种较为流行的观点：冯·诺依曼体系结构，也叫普林斯顿体系结构，描述了含有如下部件的电子数字计算机
  - 一个处理单元，包含了数学逻辑单元和处理器寄存器
  - 一个控制单元，包含了一个指令寄存器和一个PC
  - 一个内存来同时存储数据和指令
  - 外部大容量存储
  - 输入输出机制
- 这个概念演化为
  - 一种存储程序的计算机，在这样的计算机上，由于指令的读取和数据的操作共享同一总线，两者不能同时进行

From Wikipedia: http://en.wikipedia.org/wiki/Von_neumann_architecture

冯·诺依曼体系结构 2/2

- 冯·诺依曼体系结构有哪些优势、哪些劣势？
- 能不能想出一些非冯·诺依曼体系结构的计算系统？





Alan Turing
On Computable Numbers, with an Application to the Entscheidungsproblem, 1936

J. Presper Eckert
John William Mauchly
Progress report for the first period of the ENIAC's development, 1943

John Von Neumann
First Draft of a Report on the EDVAC, 1945

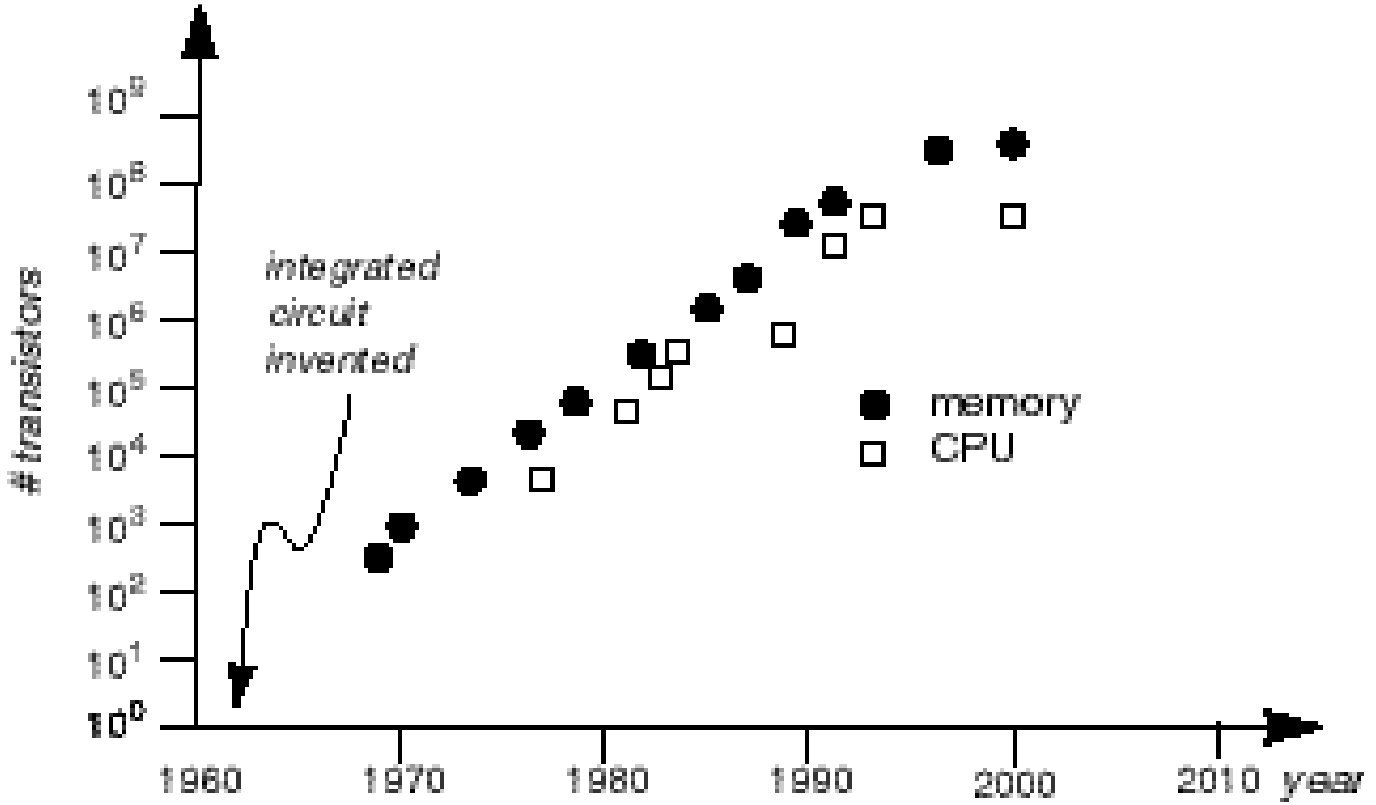
EDSAC, May 1949 (UK)
The first practical stored-program electronic computer

第二课：系统结构的基本概念

- 从计算电路到冯·诺依曼结构计算机
- 数据与公式
- 若干其它概念

摩尔定律（Moore's Law）1/2

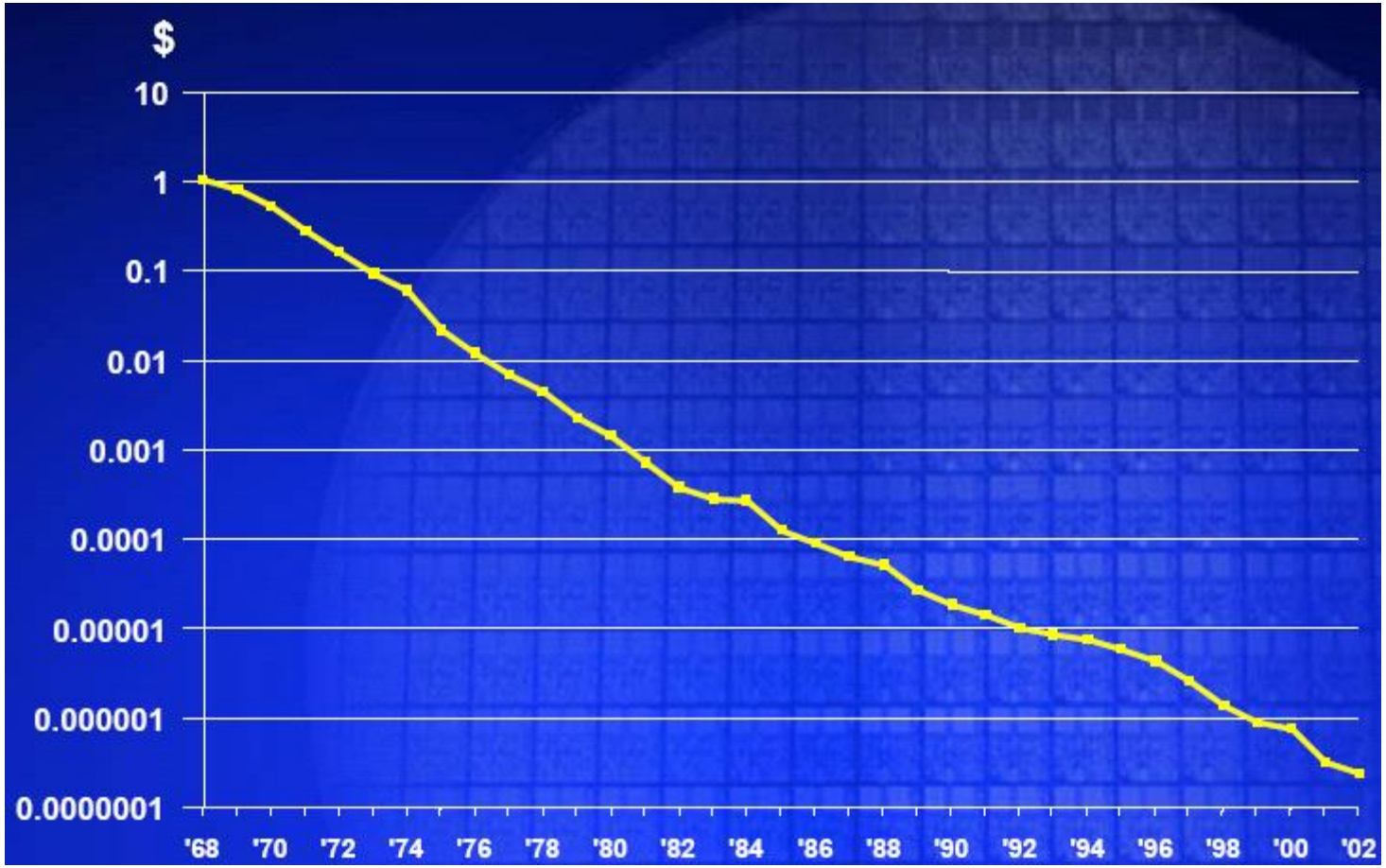
- 1965年，Gordon Moore预期，芯片上晶体管的数目每18个月翻一番



摩尔定律（Moore's Law）2/2

- 40余年，（处理器）芯片上晶体管数目从2300突破了1800亿
  - 1971，2300晶体管，Intel 4004（1 MHz），10 μm
  - 1979，68000晶体管，Motorola 68000，4 μm
  - 1989，100万晶体管，Intel 80486，1 μm
  - 1999，950万晶体管，Intel Pentium III，250nm
  - 2003，1亿晶体管， AMD K8，130 nm
  - 2010，10亿晶体管，SUN 16-Core SPARC T3，40nm
  - 2012，50亿晶体管，Intel 62-Core Xeon Phi，22nm
  - 2014，556亿晶体管，Intel 18-core Xeon Haswell-E5，22nm
  - 2015， 1000亿晶体管， Oracle SPARC M7, 20nm
  - 2017，1800亿晶体管，Qualcomm Centriq 2400， 10nm

每年每晶体管的平均价格

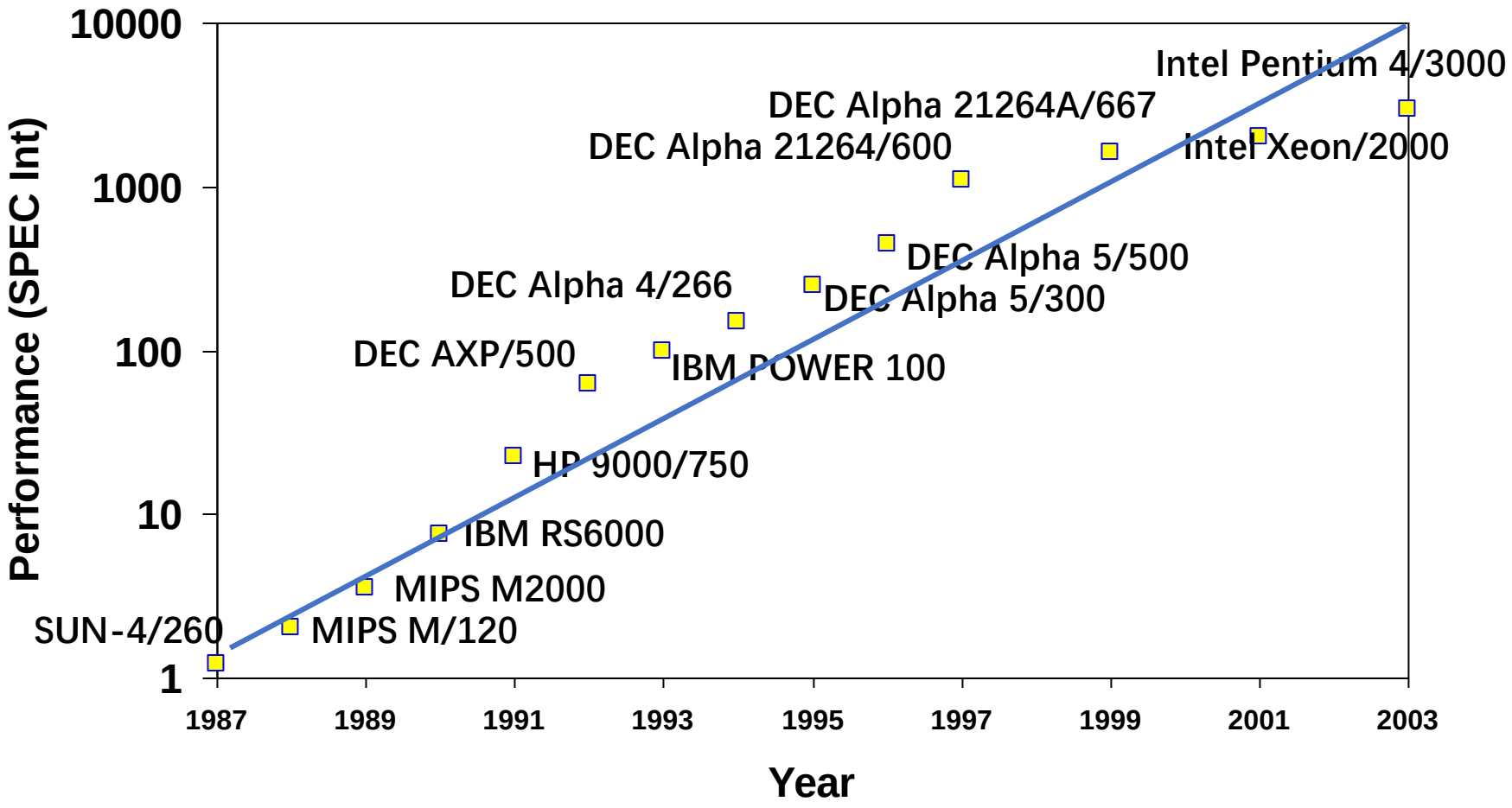


Source: G.Moore, ISSCC 2003

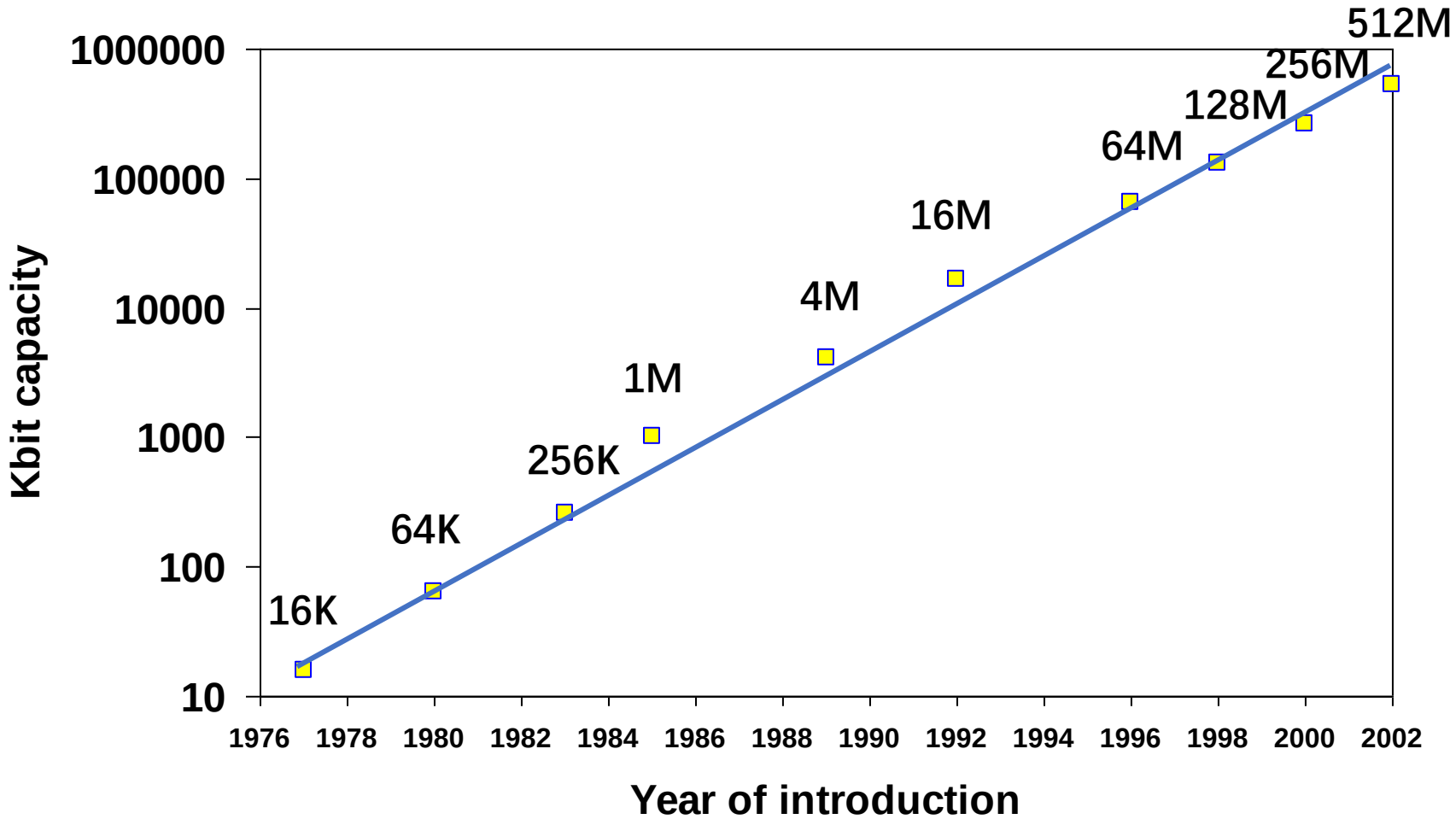
计算机最底层技术的发展

Year	Technology	Relative Pref/Unit Cost
1951	Vacuum Tube	1
1965	Transistor	35
1975	Integrated Circuit (IC)	900
1995	Very Large Scale IC (VLSI)	2,400,000
2005	Ultra VLSI	6,200,000,000

处理器性能的不断提升



DRAM容量的变化



新技术的影响

- 处理器
  - 逻辑容量：每年增长30%左右
  - 性能：每1.5年提高一倍
- 内存
  - DRAM容量：原本每三年4倍；现在每两年2倍
  - 内存速度：每十年1.5倍
  - 每bit价格：大致每年减少25%
- 硬盘
  - 容量：每年提高60%

CPU性能公式

- 程序/进程运行的CPU时间
  = 指令数 × CPI × 时钟周期时间
  = 指令数 × CPI ÷ 时钟频率

- CPI：平均执行每条指令所需的时钟周期数

单处理器与超级计算机性能的发展

- MIPS：每秒百万次整数运算
  - Intel 4004，0.092 MIPS at 740 KHz（1971）
  - Motorola 68000，1.4 MIPS at 8 MHz（1979）
  - Intel 80486DX2，25.6 MIPS at 66 MHz（1992）
  - Intel Pentium III，2,054 MIPS at 600 MHz（1999）
  - Intel Core i7 2600K（4-Core），128,300 MIPS at 3.4 GHz（2011）
  - Intel Core i7 5960X（ 8-Core），298,190 MIPS at 3.0 GHz（2014）
  - Intel Core i7 6950X（ 10-Core ），317,900 MIPS at 3.0 GHz（2016）
- MFLOPS：每秒百万次浮点运算，通常用于超级计算机评测
  - 2017.11月数据：http://www.top500.org
  - 当前世界最快的超级计算机神威-太湖之光
    - Linpack速度93.0146 Pflops（93,014,600,000 MFLOPS）
  - 世界第二：天河二号，Linpack速度33.8627 Pflops（33,862,700,000 MFLOPS）

数据中心

Hiding in Plain Sight, Google Seeks More Power

Melanie Conner for The New York Times

Google is building two computing centers, top and left, each the size of a football field, in The Dalles, Ore.

By JOHN MARKOFF and SAUL HANSELL
Published: June 14, 2006

THE DALLES, Ore., June 8 — On the banks of the windswept Columbia River, Google is working on a secret weapon in its quest to dominate the next generation of Internet computing. But it is hard to keep a secret when it is a computing center as big as two football fields, with twin cooling plants protruding four stories into the sky.

计算的开销

- 硬件购制
- 电费
- 散热
- 空间
- …

现代计算机发展的趋势

- 向大发展、向小发展
  - 大 – 并行计算机、超级计算机、云计算平台
  - 小 – 手机、可穿戴设备
- 网络化
  - 从单机，到局域网，到Internet
  - 从有线到无线
- 能耗的考虑
  - 大规模数据中心：节约能源
  - 移动设备/装电池的设备：延长续航
- 程序设计与人机交互
  - 从汇编，到C语言，到Java，甚至到Python
  - 从命令行，到图形化界面，到触摸输入、语音输入

Amdahl定律

- 一段程序，由W_S和W_P两部分组成（$W_S + W_P = 1$），W_S部分是串行运行的，W_P部分是并行运行的（或可以加速的），假设通过某种方法可以将W_P部分加速N倍，最后会使程序快多少倍？
  - 设原运行时间为T_S，加速后的时间T_T为
    $T_T = (T_S * W_S) + (T_S * W_P)/N = T_S * (W_S + W_P/N)$
  - 加速比Speedup为$T_S / T_T = \dfrac{1}{W_S + \dfrac{W_P}{N}}$

- 如果N=5，则Wp = 90%时会怎样？Wp = 10%时会怎样？
- 如果N=∞呢？

第二课：系统结构的基本概念

- 从计算电路到冯·诺依曼结构计算机
- 数据与公式
- 若干其它概念

计算机系统结构、组成、实现

- 计算机系统结构
  - 也称计算机体系结构，是软硬件的交界面，是程序员所看到的计算机的属性，即概念性结构与功能特性
  - 例：指令系统中是否含有乘法指令？
- 计算机组成
  - 计算机系统结构的逻辑实现，包括机器级内的数据通路和控制的组成及逻辑设计等
  - 例：乘法指令是通过加法器实现的，还是通过专门的高速乘法器（例如booth乘法器）实现的？
- 计算机实现
  - 计算机组成的物理实现，如器件、微组装技术等
  - 例：加法器、高速乘法器的具体实现是什么？

指令集体系结构（ISA）

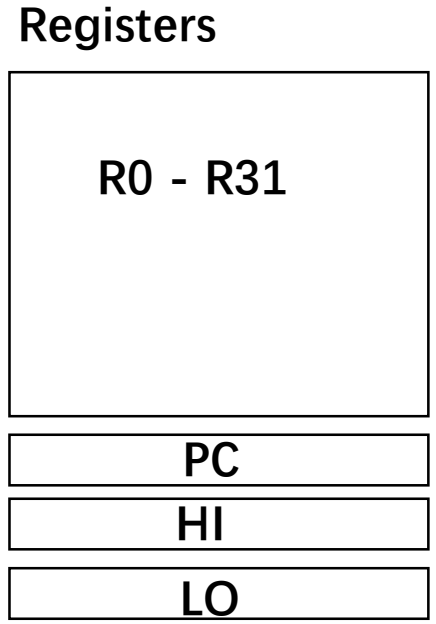
- 一台机器硬件与最底层软件之间的抽象界面，包含了编写机器语言程序所必需的所有信息，包括指令集、寄存器、存储访问方法、I/O方法等

  “... the attributes of a [computing] system as seen by the programmer, i.e., the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls, the logic design, and the physical implementation.”
  – Amdahl, Blaauw, and Brooks, 1964

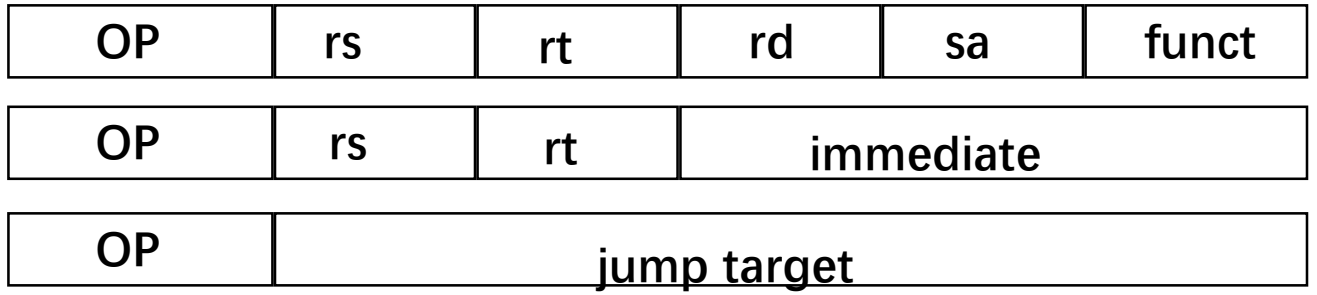
- 可以使相同软件运行在同一指令集体系结构的不同型号计算机/处理器上（它们具有不同的价格和性能）

MIPS R3000指令集体系结构

- 指令类型
  - Load/Store
  - Computational
  - Jump and Branch
  - Floating Point
    - coprocessor
  - Memory Management
  - Special

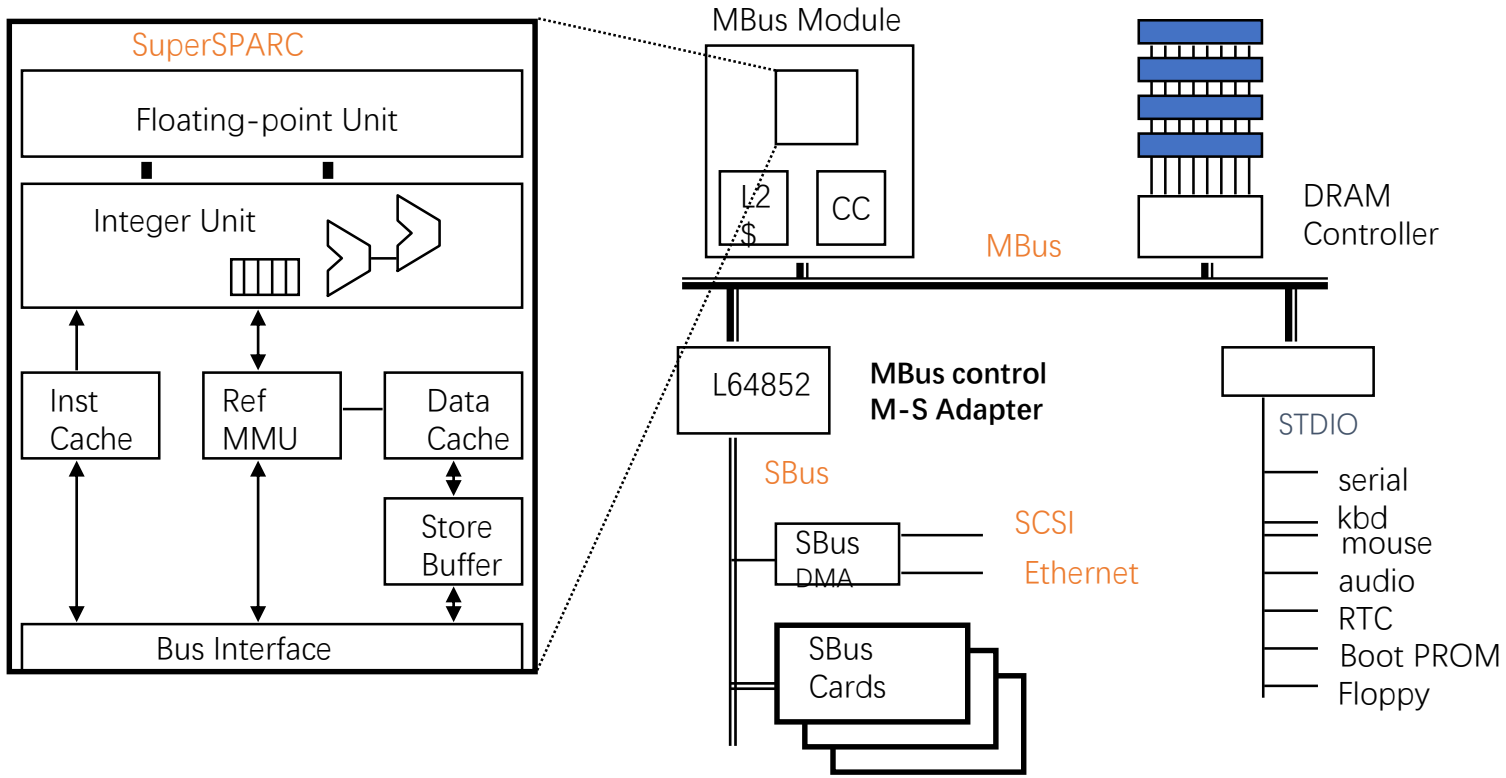


3 Instruction Formats: all 32 bits wide

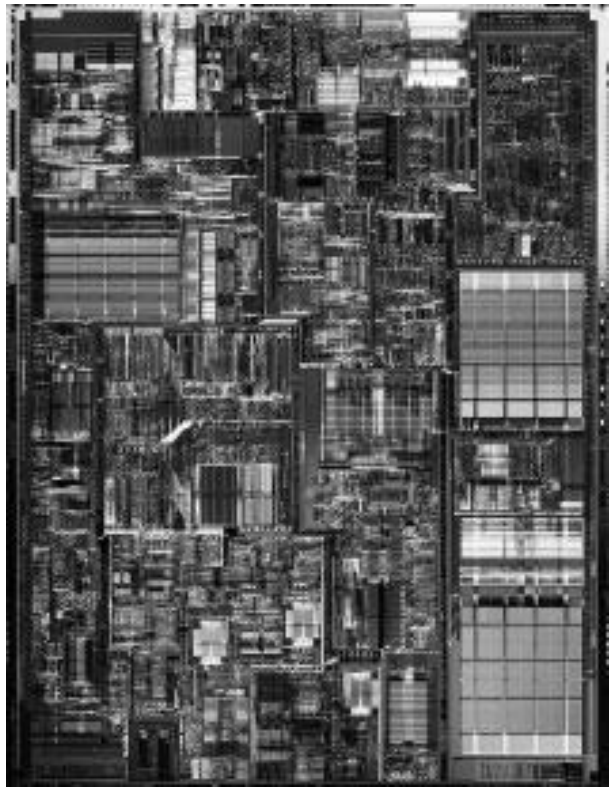


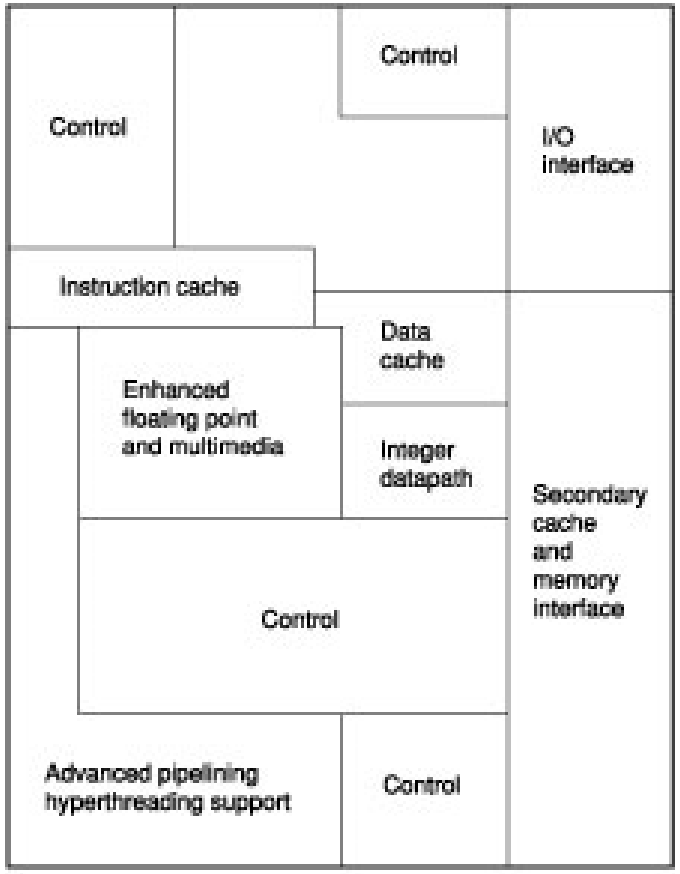
计算机组成示例

- 在Sun SPARCstation20中的TI SuperSPARC^{tm} TMS390Z50



Pentium 4处理器布局





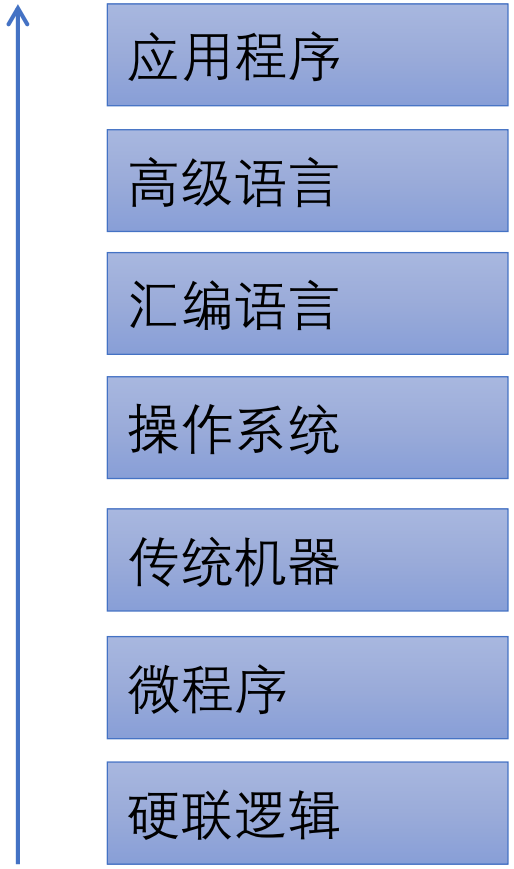
结构 v.s. 实现

By the *architecture* of a system, I mean the complete and detailed specification of the user interface. … As Blaauw has said, "Where architecture tells *what* happens, implementation tells *how* it is made to happen."

The Mythical Man-Month, Brooks, pg 45

计算机系统层次结构

- 从使用语言的角度，按照功能分级（某种分法）



计算机系统结构在哪一层？

计算机组织在哪一层？

系统结构分类的一种方法

- 可以按指令流和数据流分别能够同时处理的数目将系统结构分为四类：
  - SISD：单指令单数据流
  - SIMD：单指令多数据流
  - MISD：多指令单数据流
  - MIMD：多指令多数据流

透明性

- 透明指的是（客观存在的事物或属性从某个角度看不到），它带来的好处是（简化某级的设计），带来的不利是（无法控制）
- 注意，透明性是分层级的
- 透明性举例：对于程序员来讲，其程序所在的计算机的高速缓存就是透明的

- 还有什么透明性的例子？

局部性原理

- 时间局部性
  - 近期被访问的位置，很可能会再次被访问
- 空间局部性
  - 近期被访问的位置的“邻居位置”，也很有可能会被访问
- 是高速缓存、页面替换等机制的基础
  - 如何应用?

数据类型、数据表示和数据结构

- 数据类型：图、表、树、队列、栈…
- 数据表示：研究这些数据类型怎么能够“表示”为可以被计算机指令系统识别、调用的形式
- 数据结构：面对这些数据类型，研究它们内部逻辑结构和物理结构（程序中的数据组织方法）之间的关系，并给出相应的方法来操作
- 设计计算机系统时，确定数据表示时考虑的因素
  - 程序运行时间、CPU与内存的通信量、数据表示的通用性和利用率

总结

- 从计算电路到冯·诺依曼结构计算机
- 数据与公式
- 若干其它概念

- 下一课：指令系统

思考题

1. 如果有一个经解释实现的计算机，可以按功能划分为4级。每一级为了执行一条指令需要下一级的N条指令解释。若执行第一级的一条指令需要K ns的时间，那么执行第2、3、4级的一条指令各需要用多少时间？
2. 如果某一计算任务用向量方式求解比用标量方式求解要快20倍，称可用向量方式求解部分所花费时间占总时间的百分比为可向量化百分比。请画出加速比与可向量化比例两者关系的曲线。
3. 假设高速缓存cache工作速度为主存的5倍，且cache被访问命中的概率为90%，则采用cache后能使整个存储系统获得多高的加速比Sp？
4. 数据类型、数据表示和数据结构之间的关系。