计算机组成原理
(2014级)

计算机组成原理课程组

（刘旭东、肖利民、牛建伟、栾钟治）

Tel： 82316285

Mail：liuxd@buaa.edu.cn

liuxd@act.buaa.edu.cn

第一部分：概述

一. 计算机组成与结构简介

1. 计算机的基本组成
2. 计算机的层次结构
3. 总线结构

二. 计算机中数的表示

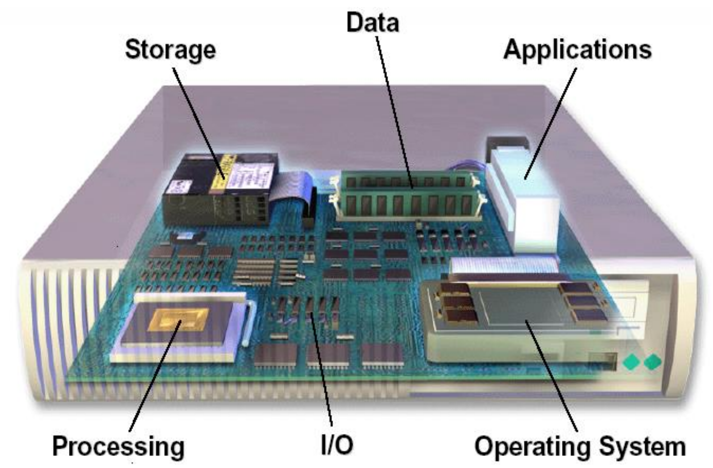
1. 无符号数和有符号数
2. 定点数、浮点数表示
3. 非数值数据的表示

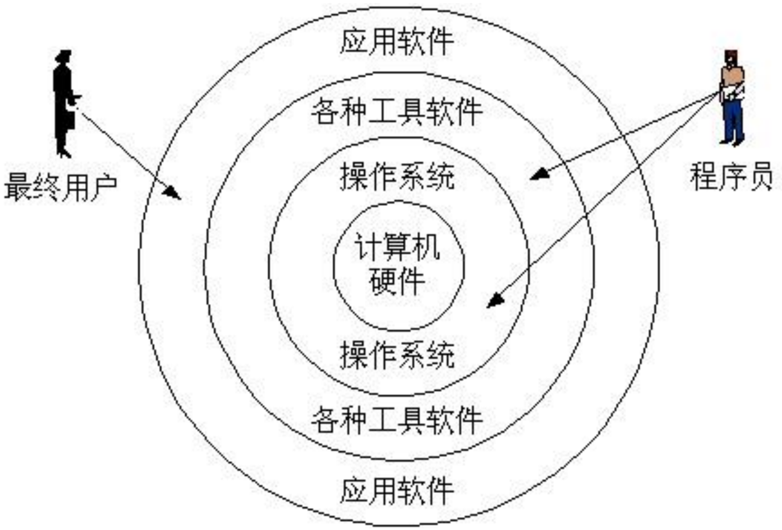
三. 计算机的基本工作过程

1. 指令的含义
2. 程序的执行

1.1 计算机的基本组成

- ❖硬件（**Hardware**）
  - ➢计算机的物理部分，可以实现计算机最基本的操作行为。
- ❖软件（**Software**）
  - ➢使计算机实现各种功能的程序集合。包括系统软件、应用软件两大类。



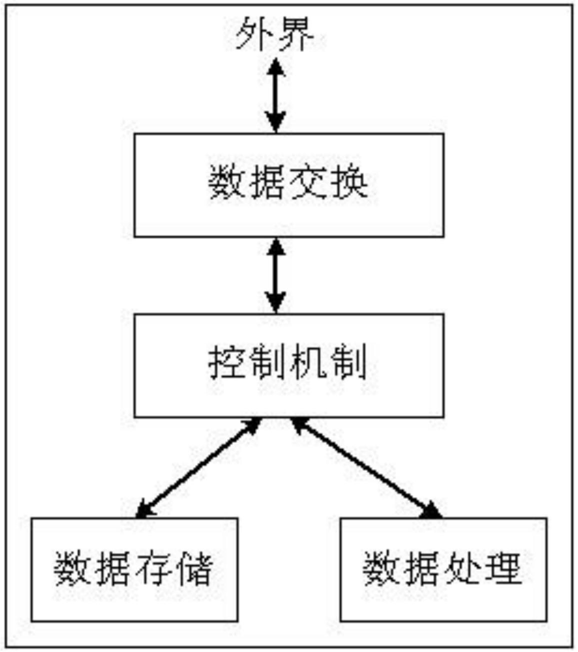


1.1 计算机的基本组成

❖计算机的功能

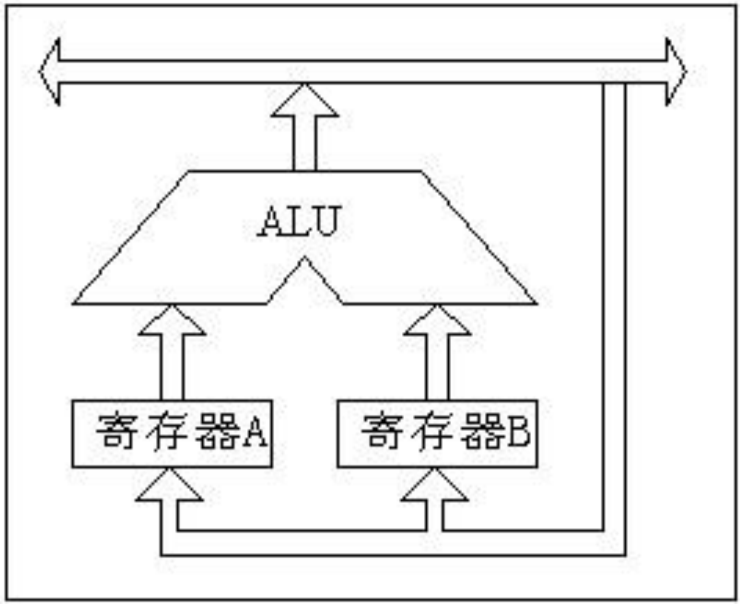
- Data Processing (数据处理)
- Data Storage （数据存储）
- Data Movement （数据移动，交换）
- Control （控制）

❖计算机的功能结构



1.1 计算机的基本组成

- 运算器：实现数据处理的部件
  - 完成最基本的算术逻辑运算
  - **ALU (Arithmetic and Logic Unit）＋ Registers**
  - 运算器与机器字长（字的概念）的关系
  - 运算器与机器性能指标：
    - **MIPS：Millions of Instructuions Per Second**
- 简单运算器结构图

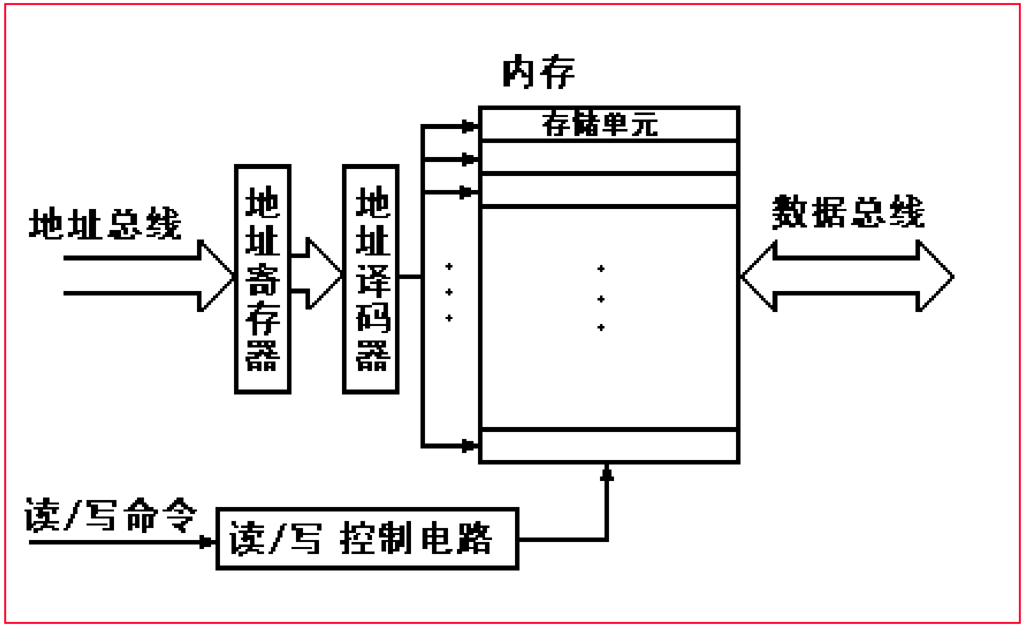


1.1 计算机的基本组成

❖存储器：实现数据存储的部件

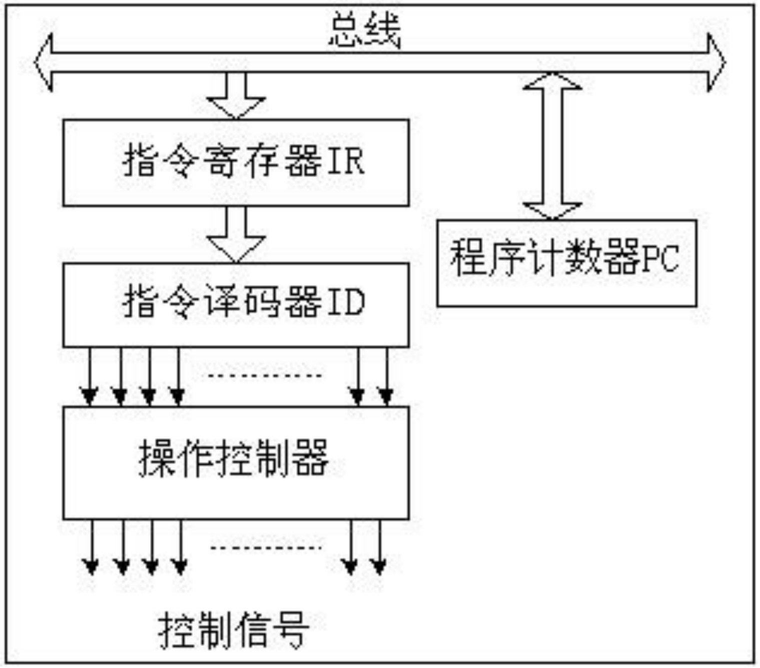
- 保存程序和数据（二进制信息）
- 存储单元：**bit, Byte, Word**
- 地址的概念：每一个字节单元拥有一个唯一的地址（索引）
- 存储器的工作方式：读、写

❖存储器结构简图



1.1 计算机的基本组成

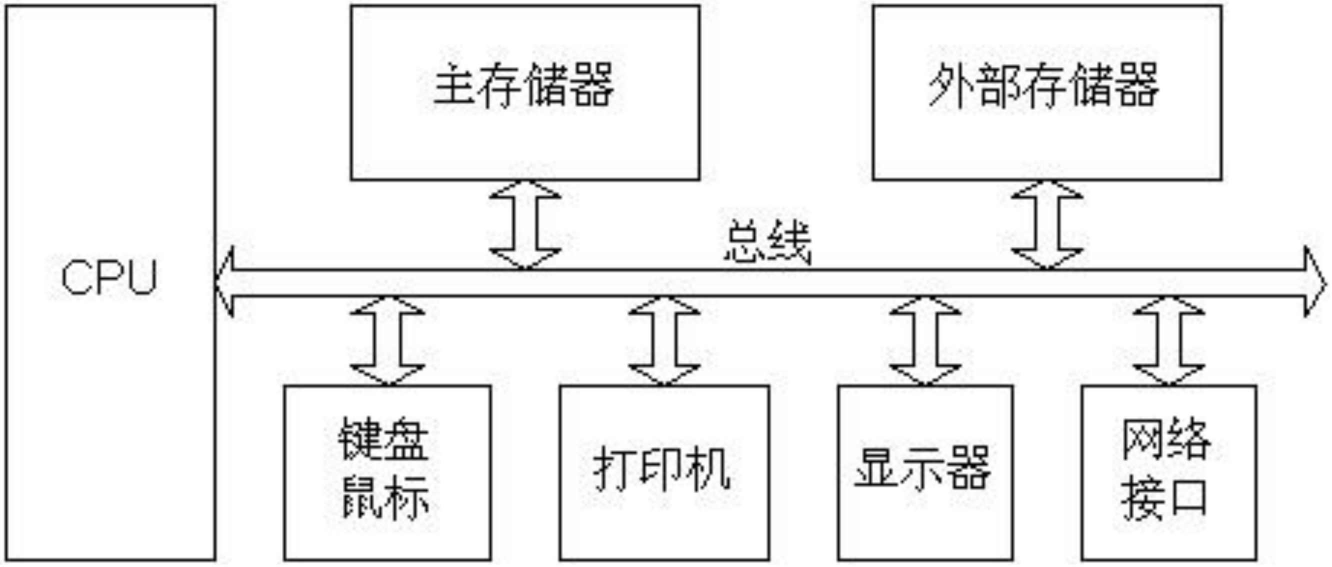
- 控制器：实现控制功能的部件
  - 提供各部件工作所需的控制信号，控制计算机其他部件协同工作
  - 指令部件（**Instruction Register， Instruction Decoder**）
  - 指令顺序控制（**Program Counter**）
  - 时序逻辑部件（**Clock，Timer， Sequencing Logic**）
  - 控制信号生成部件（**Control Signal Generator or Control Memory**）
  - **Datapath ＋Control ＝CPU（Central Process Unit）or Processor**
- 控制器结构简图



1.1 计算机的基本组成

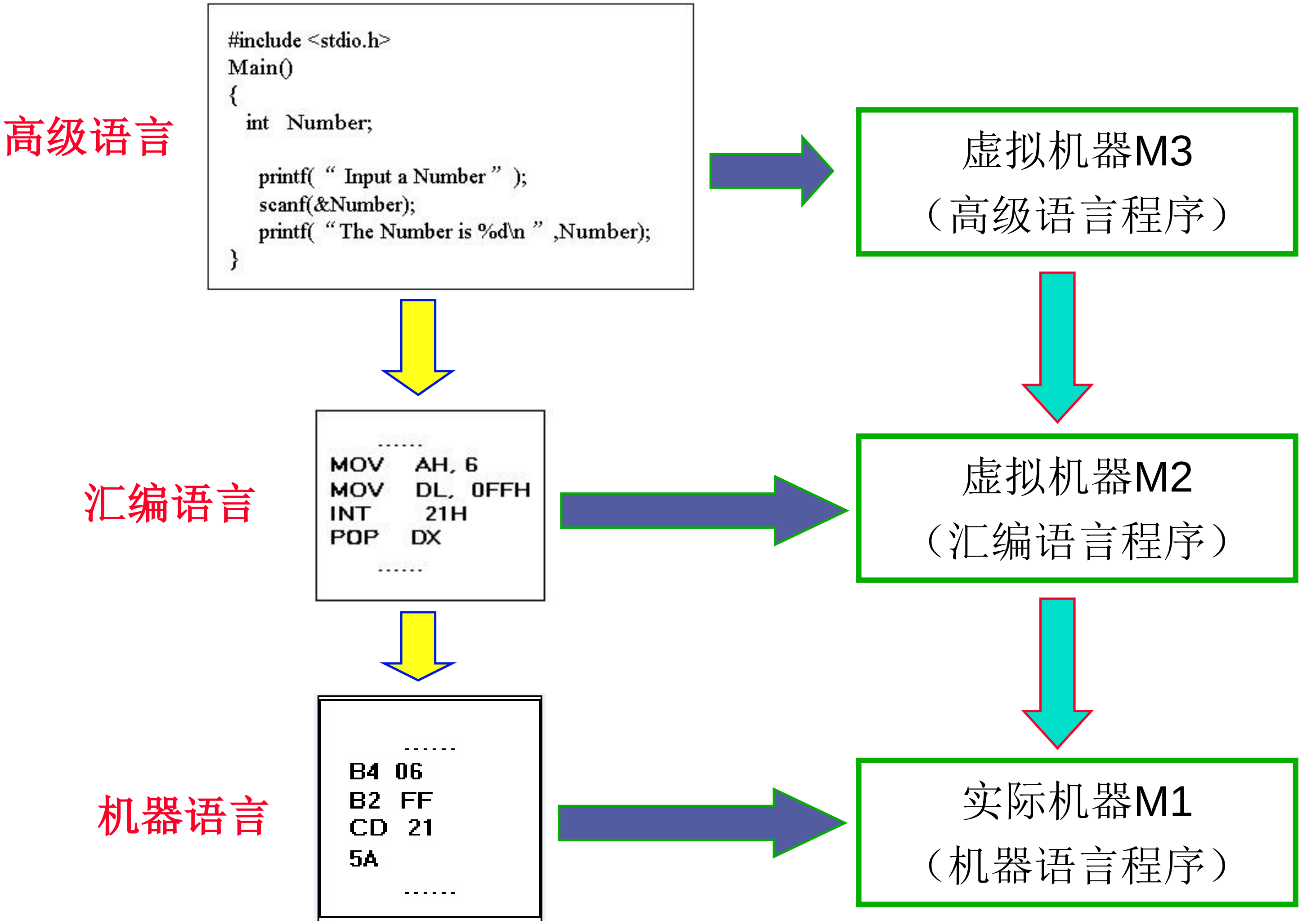
❖ 输入输出：实现数据交换的部件
- 实现计算机内部与外界（其他系统或人类）的信息交换
- 实现数据交换的设备：输入设备、输出设备
- 接口标准与接口部件

❖ 计算机整体结构简图



1.2 计算机系统层次结构

❖ 计算机的层次结构的演变



1.2 计算机系统层次结构

Higher-Level Language Program (e.g. C)

↓ *Compiler*

Assembly Language Program (e.g. MIPS)

↓ *Assembler*

Machine Language Program (MIPS)

↓ *Machine Interpretation*

Hardware Architecture Description (e.g. block diagrams)

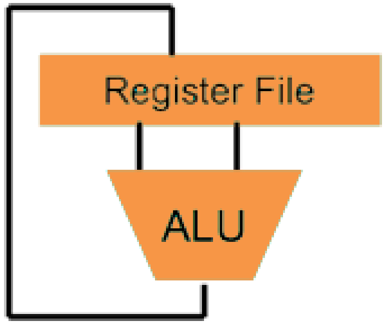
↓ *Architecture Implementation*

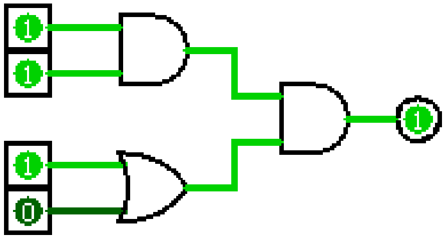
Logic Circuit Description (Circuit Schematic Diagrams)

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

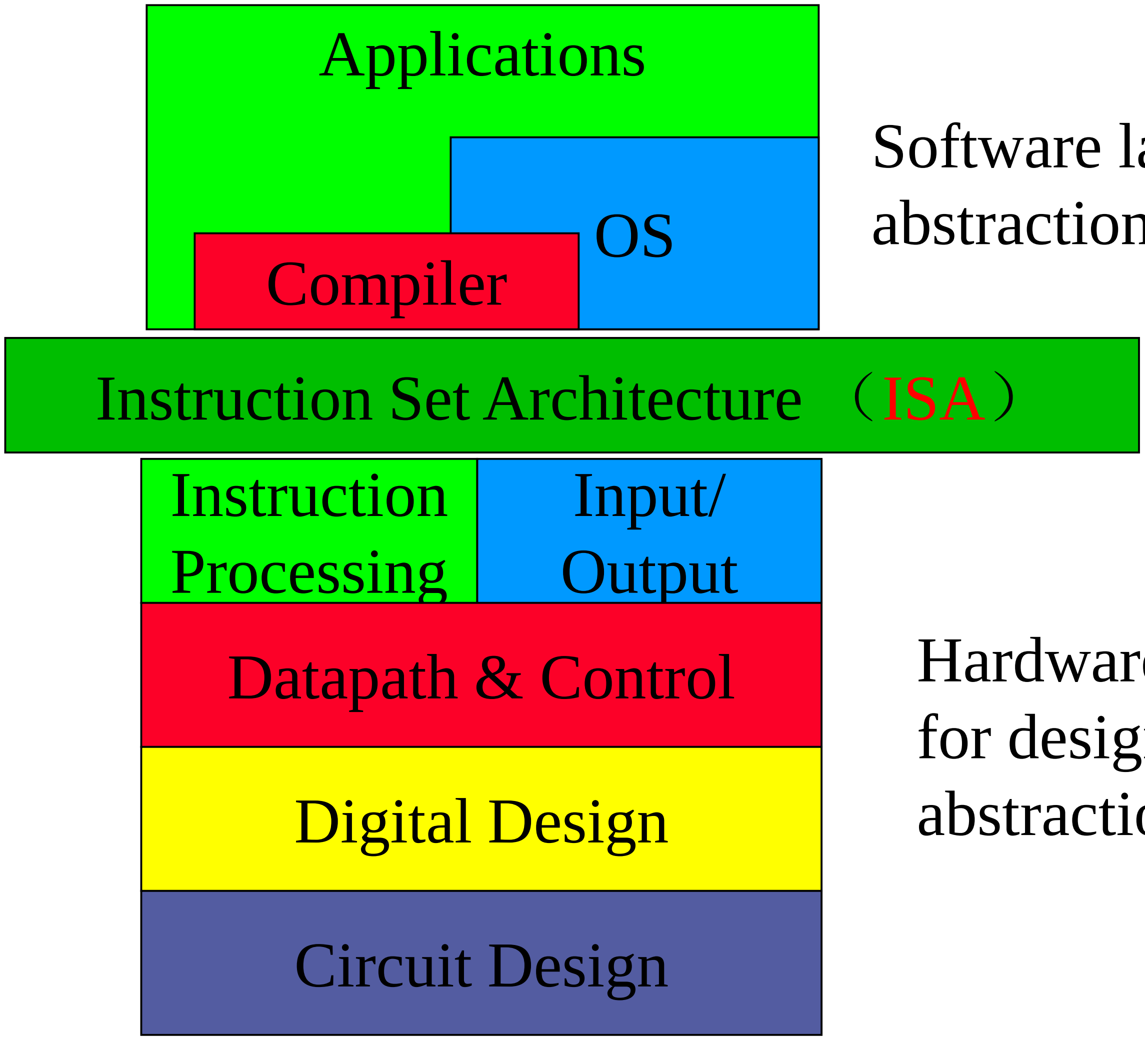
```
Lw   $t0, 0($2)
Lw   $t1, 4($2)
Sw   $t1, 0($2)
Sw   $t0, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```





1.2 计算机系统层次结构

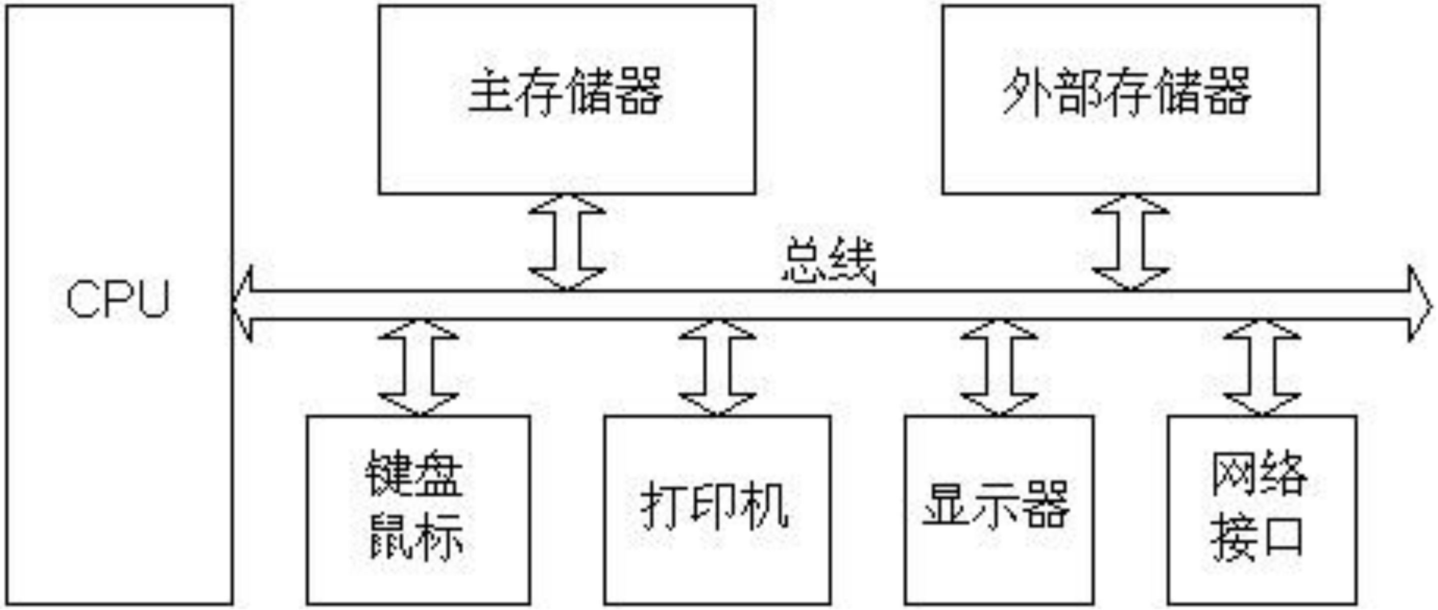


Software layers of abstraction

Hardware layers for design abstraction

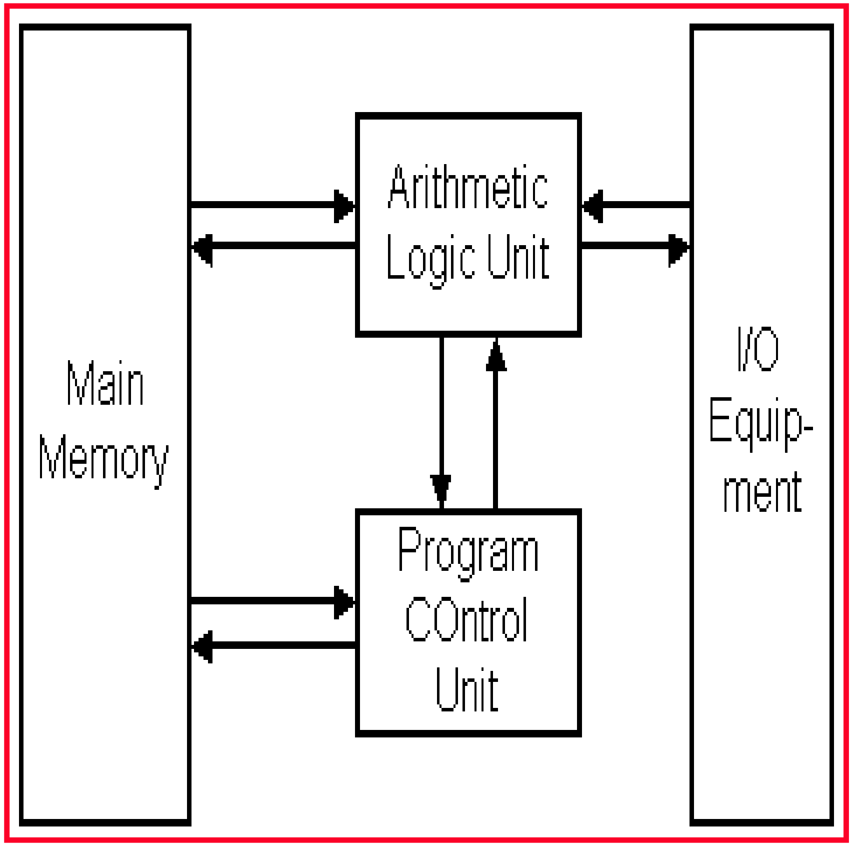
❖总线结构

- 总线：符合一定的标准的一组公共信息通道
- 单总线结构、多总线结构
- 系统总线构成：地址总线、数据总线、控制总线



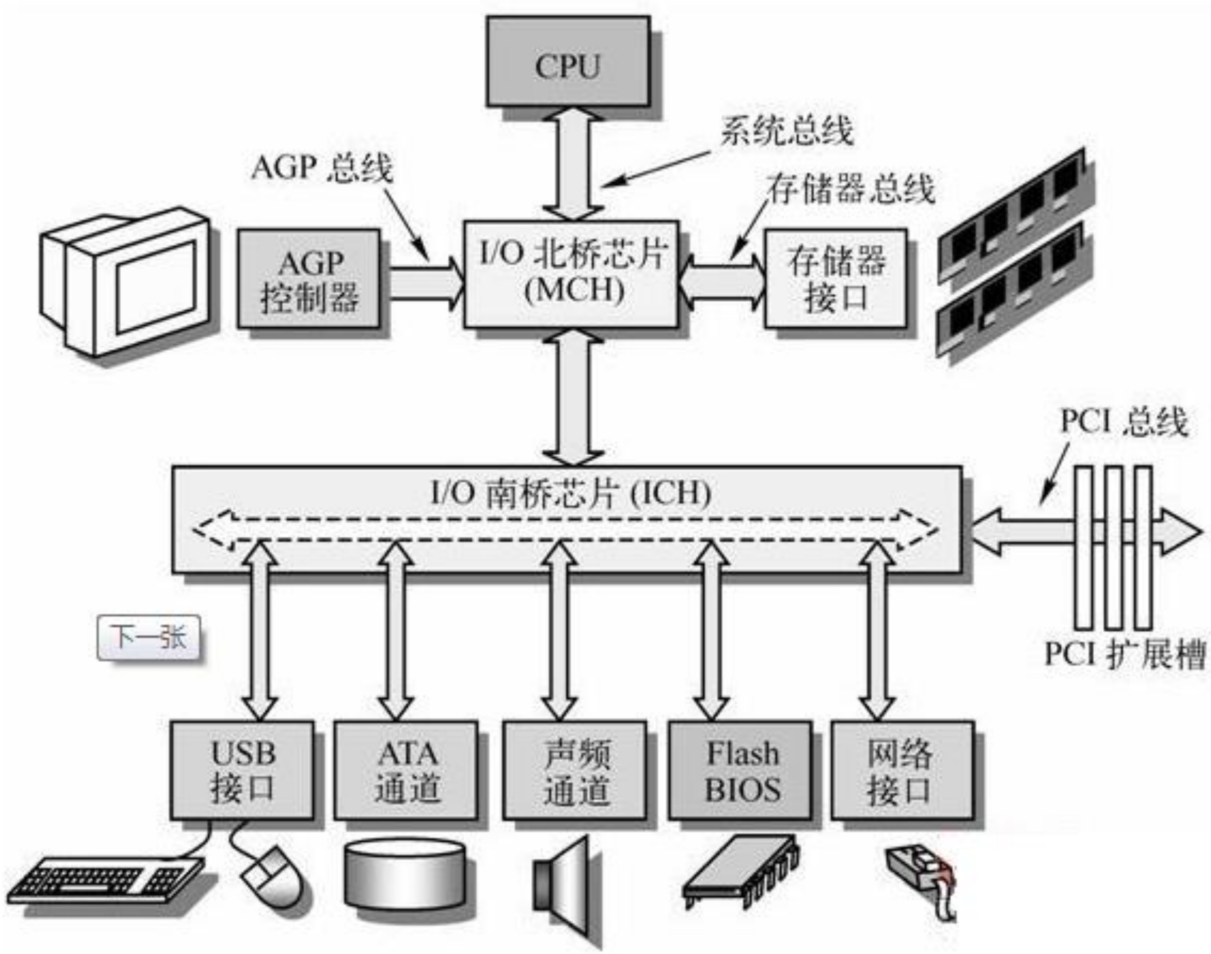
1.3 计算机总线结构

- *1946年，冯·诺依曼与同事开始研制 IAS。该机结构被公认为随后发展起来的通用计算机的原型。*

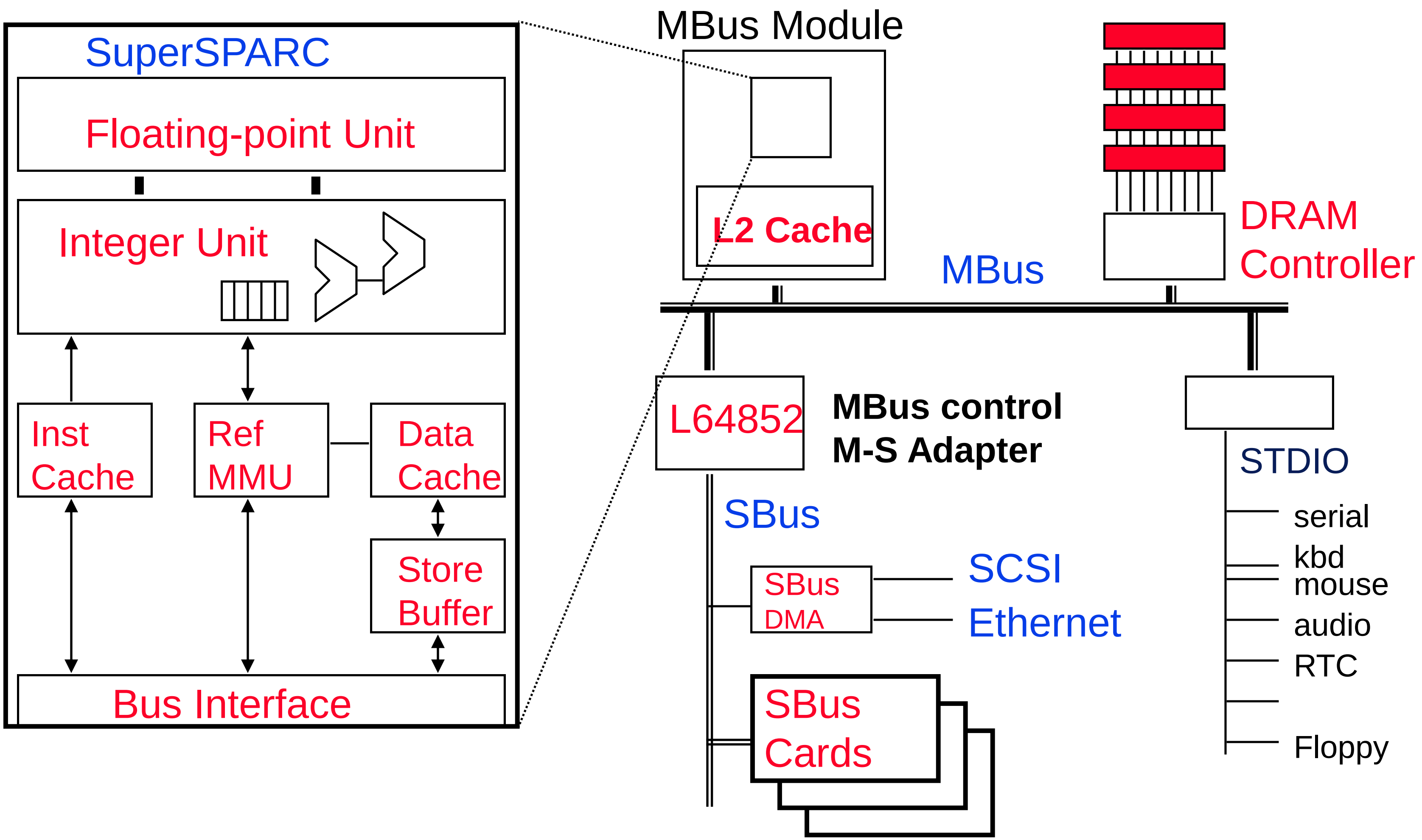


1.3 计算机总线结构

📖 普通PC（Pentium）的内部结构（多总线结构）

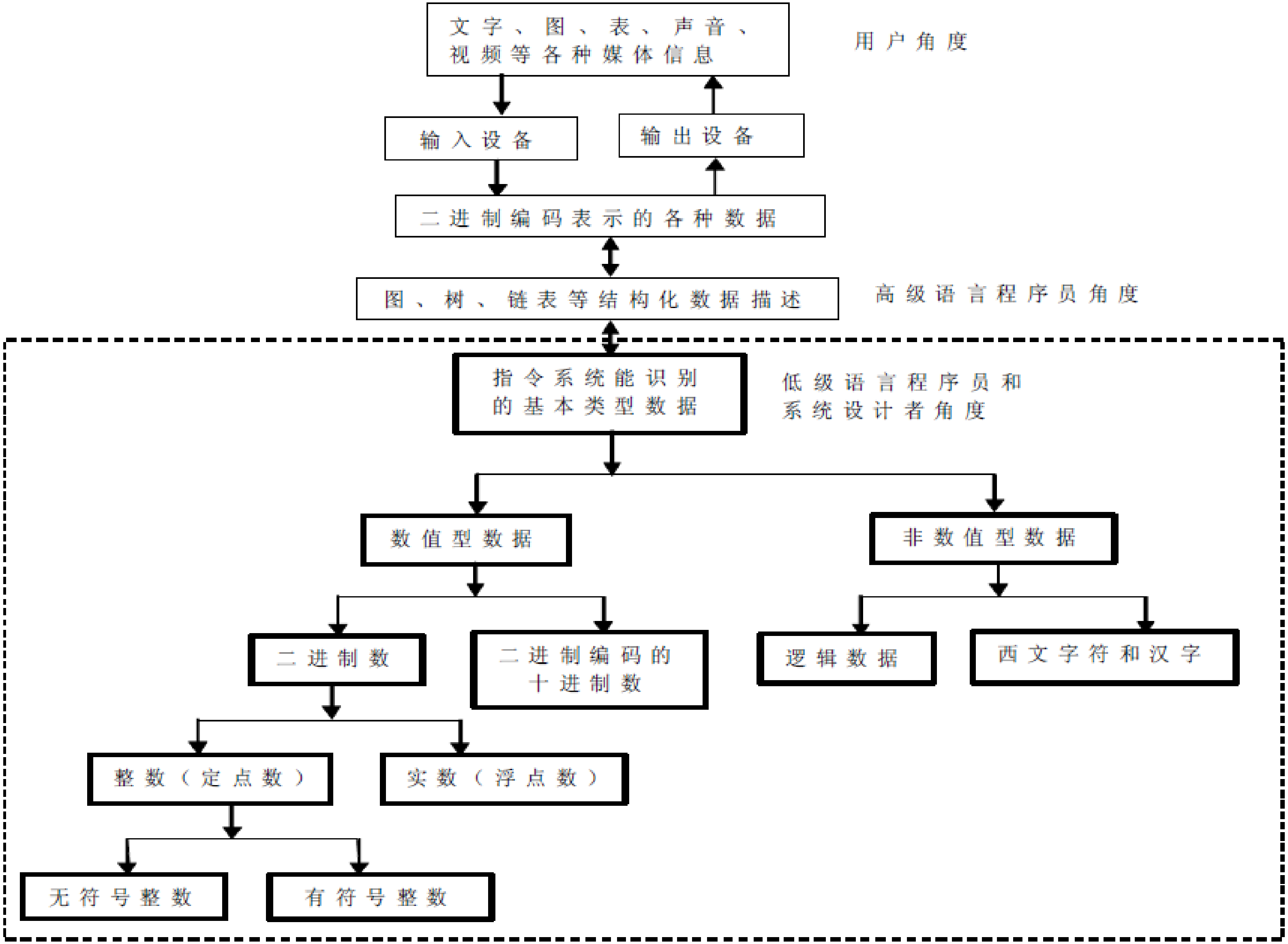


Sun SPARCstation20（RISC）多总线结构



第一部分：概述

计算机中数的表示的基本问题



计算机中数的表示的基本问题

❖计算机数值数据表示的基本问题

- 进位计数制：采用二进制，只有**1**和**0**；
- 数的符号：正数、负数、零
- 数的形态：整数、小数、小数点的性质；

01000001

字符：字母**A**（**ASCII**码）

整数：**65**

小数：**0.253906**

2.1 无符号数和有符号数

❖无符号数

- 数的编码中所有位均为数值位，没有符号位
- 只能表示 **>=0** 的正整数
- **16**为无符号数的表示范围： **0 ~ 65535**
- 一般在全部是正数运算且不出现负值结果的场合下，可使用无符号数表示，例如地址运算。

2.1 无符号数和有符号数

- 有符号数
  - 数的实例：+ 0.1010110， - 0.1101001，+ 1001.001， -1101101
- 机器数表示
  - 数的正负问题：设符号位， “0”表示“正”， “1”表示“负”，固定为编码的最高位
  - 真值0怎么办：正零，负零
  - 小数点怎么办：固定小数点 （即 定点数）
    - 定点小数：绝对值小于1
    - 定点整数：没有小数部分
  - 带有整数和小数部分的数怎么办：浮点数，按2为基的科学表示方法表示

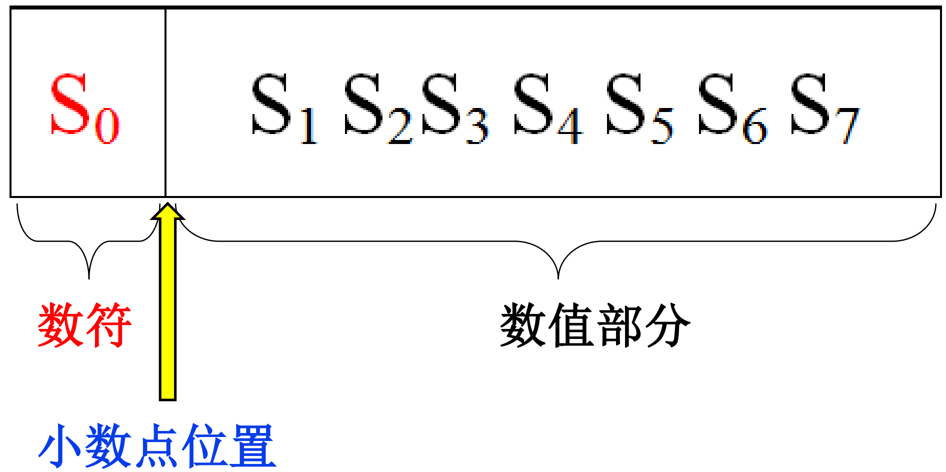
C语言中变量为什么要一定先定义类型才能使用

char、int、unsigned、float、double

➢定点小数

如：**01100000**

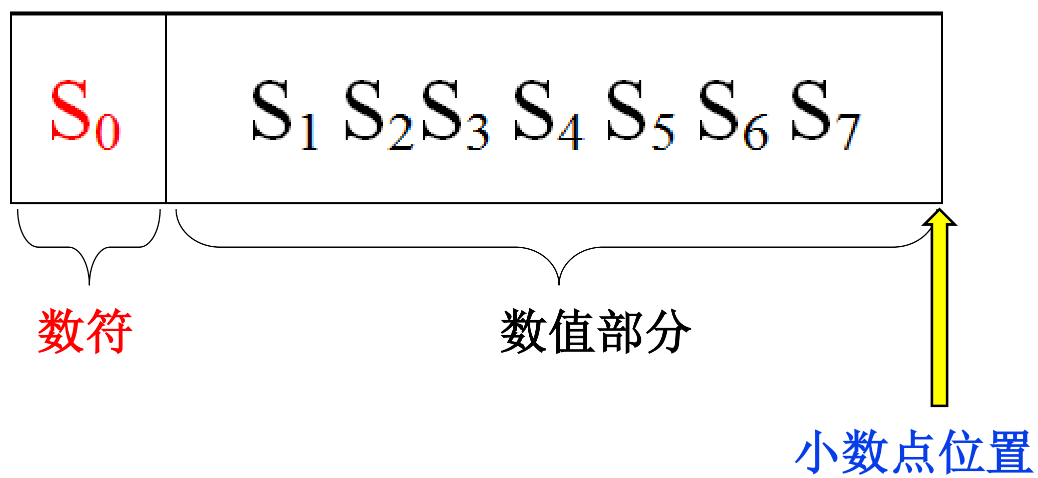
是十进制的**0.75**



➢定点整数

如：**01100000**

是十进制的**192**



2.2 定点数表示（定点整数与定点小数）

❖机器数表示及其表示范围（原码、反码、补码、移码）

$$[x]_{原}=\begin{cases}x & 0\le x\le 2^{n-1}-1\\ 2^{n-1}-x & -(2^{n-1}-1)\le x\le 0\end{cases}$$

$$[x]_{反}=\begin{cases}x & 0\le x\le 2^{n-1}-1\\ (2^{n}-1)+x & -(2^{n-1}-1)\le x\le 0\end{cases}$$

$$[x]_{补}=\begin{cases}x & 0\le x\le 2^{n-1}-1\\ 2^{n}+x & -2^{n-1}\le x\le 0\end{cases}$$

$$[x]_{移}=2^{n-1}+x \qquad -2^{n-1}\le x\le 2^{n-1}-1$$

N 位定点整数的原码、反码、补码和移码表示及其表示范围

2.2 定点数（定点整数与定点小数）

十进制数值	原码	反码	补码
0	0000	0000	0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	0101	0101
6	0110	0110	0110
7	0111	0111	0111
-0	1000	1111	0000
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001

2.2 定点数（定点整数与定点小数）

- 原码
  - 容易理解
  - “0”的表示不唯一，不利于程序员编程
  - 机器实现加、减运算的方法不统一
  - 需对符号位进行单独处理，不利于硬件设计
- 反码
  - 很少使用
- 补码
  - “0”的表示唯一
  - 机器实现加、减运算的方法统一（模运算）
  - 符号位参加运算，不需要单独处理

2.3 浮点数表示

- 浮点数的一般表示法：分为阶码和尾数两个部分
  - 阶码：采用定点整数表示
  - 尾数：采用定点小数表示
  - 例： $(178.125)_{10}=(10110010.001)_2$

$$=0.10110010001\times 2^{01000}$$

阶码：**01000**

尾数：**0.10110010001**

J_0	$J_1 J_2 J_3 J_4 \ldots J_m$	S_0	$S_1 S_2 S_3 S_4 \ldots S_n$
阶符	阶码的数值部分	数符	尾数的数值部分

2.3 浮点数表示（IEEE 754）

- IEEE 754：符号（**S**ign）、阶码（**E**xponent）和尾数（**M**antissa）。
- IEEE 754标准：单精度浮点数32位，双精度浮点数64位
  - 数符 S： 1位，0表示正数，1表示负数
  - 阶码 E：用移码表示，n 位阶码偏移量为 $2^{n-1}-1$。如8位阶码偏移量为 7FH（即127），11位阶码偏移量3FFH（即1023）
  - 尾数 M： 尾数必须规格化成小数点左侧一定为1，并且小数点前面这个1作为隐含位被省略。这样单精度浮点数尾数实际上为24位。
  - 规格化数（尾数）形式：M=1.m

单精度浮点数32位	数符 S：1位	阶 E：8位	尾数m：23位

双精度浮点数64位	数符 S：1位	阶 E：11位	尾数m：52位

2.3 浮点数表示（IEEE 754）

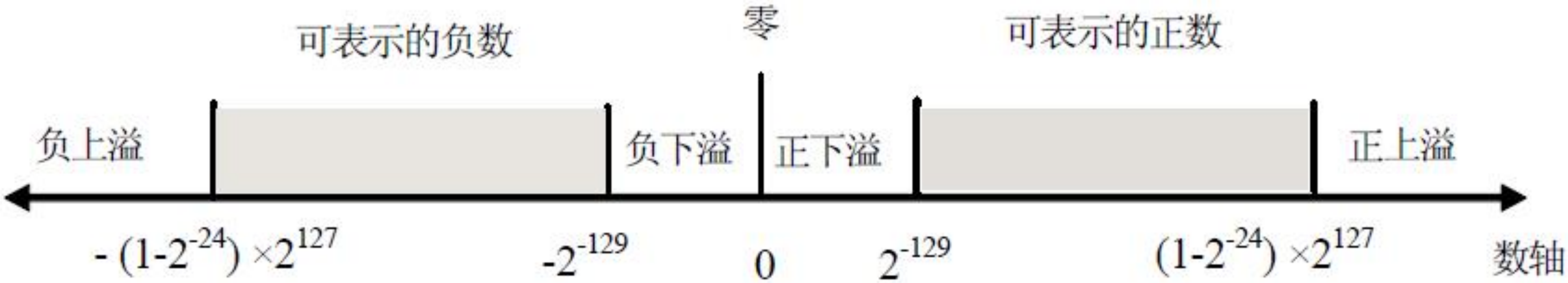
❖浮点数的表示范围（单精度为例）

0	1　　　8	9　　　　　　　　31
S	阶码E	尾数M

第0位为数符S；第1～8位为8位移码表示的阶码E（偏置常数为128）；第9～31位为24位二进制原码小数表示的尾数M。规格化尾数的第一位总是1，故规定第一位默认的“1”不明显表示出来。这样可用23个数位表示24位尾数。

最大正数： $0.11…1 \times 2^{11…1} = (1-2^{-24}) \times 2^{127}$　　**最小正数：** $0.10…0 \times 2^{00…0} = (1/2) \times 2^{-128}$

因为原码和移码都是对称的，所以其表示范围是关于原点对称的。



机器0：阶码为0 或 落在下溢区中的数

2.3 浮点数表示（IEEE 754标准）

❖ 浮点数精度

- 单精度浮点数表示公式：$(-1)^S \times 1.m \times 2^{(E-127)}$
- 双精度浮点数表示公式：$(-1)^S \times 1.m \times 2^{(E-1023)}$

❖ IEEE 754关于浮点数表示的约定（单精度为例）

E	M	浮点数 N
$1 \le E \le 254$	$M \ne 0$	表示规范浮点数 $N = (-1)^s \times 1.m \times 2^{(E-127)}$
$E = 0$	$M = 0$	表示 $N = 0$
$E = 0$	$M \ne 0$	表示非规范浮点数 $N = (-1)^s \times 0.m \times 2^{-126}$
$E = 255$	$M = 0$	表示无穷大，由符号位 S 确定是正无穷大还是负无穷大
$E = 255$	$M \ne 0$	NaN（Not a Number）不是一个数

非规格化浮点数尾数部分不必规格化成小数点左侧为1，而是0。

2.3 浮点数表示（IEEE 754标准）

❖单精度浮点数表示范围

Range Name	Sign (s) 1 [31]	Exponent (E) 8 [30-23]	Mantissa (m) 23 [22-0]	Hexadecimal Range	Range
-NaN	1	11..11	11..1 ~ 00..01	FFFFFFF ~ FF800001	
-Infinity (Negative Overflow)	1	11..11	00..00	FF800000	$< -(2-2^{-23}) \times 2^{127}$
Negative Normalized $-1.m \times 2^{(E-127)}$	1	11..10 ~ 00..01	11..11 ~ 00..00	FF7FFFFF ~ 80800000	$-(2-2^{-23}) \times 2^{127} \sim -2^{-126}$
Negative Denormalized $-0.m \times 2^{(-126)}$	1	00..00	11..11 ~ 00..01	807FFFFF ~ 80000001	$-(1-2^{-23}) \times 2^{-126} \sim -2^{-149}$
-0	1	00..00	00..00	80000000	-0
+0	0	00..00	00..00	00000000	0
Positive Denormalized $0.m \times 2^{(-126)}$	0	00..00	00..01 ~ 11..11	00000001 ~ 007FFFFF	$2^{-149} \sim (1-2^{-23}) \times 2^{-126}$
Positive Normalized $1.m \times 2^{(E-127)}$	0	00..01 ~ 11..10	00..00 ~ 11..11	00800000 ~ 7F7FFFFF	$2^{-126} \sim (2-2^{-23}) \times 2^{127}$
+Infinity (Positive Overflow)	0	11..11	00..00	7F800000	$> (2-2^{-23}) \times 2^{127}$
+NaN	0	11..11	00..01 ~ 1..11	7F800001 ~ 7FFFFFFF	

❖单精度浮点数示例:178.125, -0.0449219

$$
\begin{aligned}
(178.125)_{10} &= (10110010.001)_2 \\
&= 1.0110010001 \times 2^{111}
\end{aligned}
$$

$$S = 0$$

$$
\begin{aligned}
E &= 00000111 + 01111111 \\
&= 10000110
\end{aligned}
$$

$$m = 01100100010000000000000$$

$$
\begin{aligned}
(-0.0449219)_{10} &= (-0.0000101110)_2 \\
&= -1.01110 \times 2^{-101}
\end{aligned}
$$

$$S = 1$$

$$
\begin{aligned}
E &= -00000101 + 01111111 \\
&= 01111010
\end{aligned}
$$

$$m = 01110000000000000000000$$

2.4 非数值数据的表示

- 逻辑数据编码
  - 一位二进制编码表示：真（1）、假（0）

- 西文字符编码（ASCII码，7位）
  - 数字字符：0/1/2/…/9
  - 英文字母（大小写）：A/B/C/…/Z/a/b/c/…/z
  - 专用符号：+/-/%/*/&/…
  - 控制字符（不可打印或显示字符）

- 汉字编码
  - 输入码：用于汉字的输入，如拼音码、五笔字型码；
  - 国标码：1981年我国颁布了《信息交换用汉字编码字符集·基本集》(GB2312—80)。该标准规定了6763个常用汉字的标准代码。
  - 内　码：用于汉字存储、查找、传送等，基于国标码，占2个字节；
  - 点阵码或汉字向量描述：用于汉字的显示和打印。

2.4 非数值数据的表示

❖国际多字符集

- 国际标准**ISO/IEC 10646**提出了一种包括全世界现代书面语言文字所使用的所有字符的标准编码，每个字符用**4**个字节编码（**UCS-4**）或**2**字节编码（**UCS-2**）。
- 我国（包括港台地区）与日本、韩国联合制订了一个统一的汉字字符集（**CJK**编码），共收集了上述不同国家和地区的共约**2**万多汉字及符号，采用**2**字节编码（**UCS-2**），现已成为国标（**GB13000** ）。
- 微软**Windows**中采用中西文统一编码，收集了中、日、韩三国常用的约**2**万汉字，称为“**Unicode**”，采用**2**字节编码，与**UCS-2**一致。

2.4 非数值数据的表示

❖数据的检错/纠错

- 数据在存取和传送时，由于元器件故障或噪音干扰等原因会出现数据差错，因此产生了检查差错、纠正差错的需求；
- 采用“冗余校验”的思想，在原数据编码之外，增加若干位校验码，实现检错或纠错功能。
- 常用校验码
  - 奇偶校验码
  - 海明校验码
  - 循环冗余校验码

第一部分：概述

3.1 计算机的工作过程

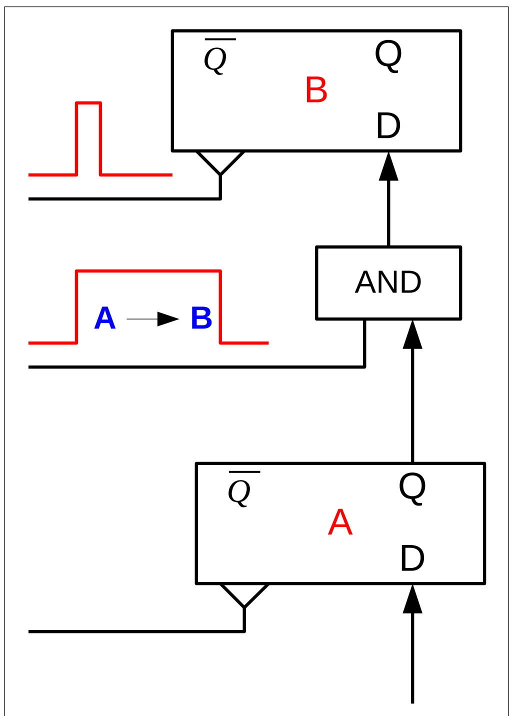
- 机器指令：计算机硬件可以执行的表示一种基本操作的二进制代码。
  - 指令格式：操作码 ＋ 操作数（操作数地址）
  - 操作码：指明指令的操作性质
  - 操作数（地址）：指令操作数的位置（或操作数本身）

操作码	操作数地址
11010101	10000100 01010001 10100000

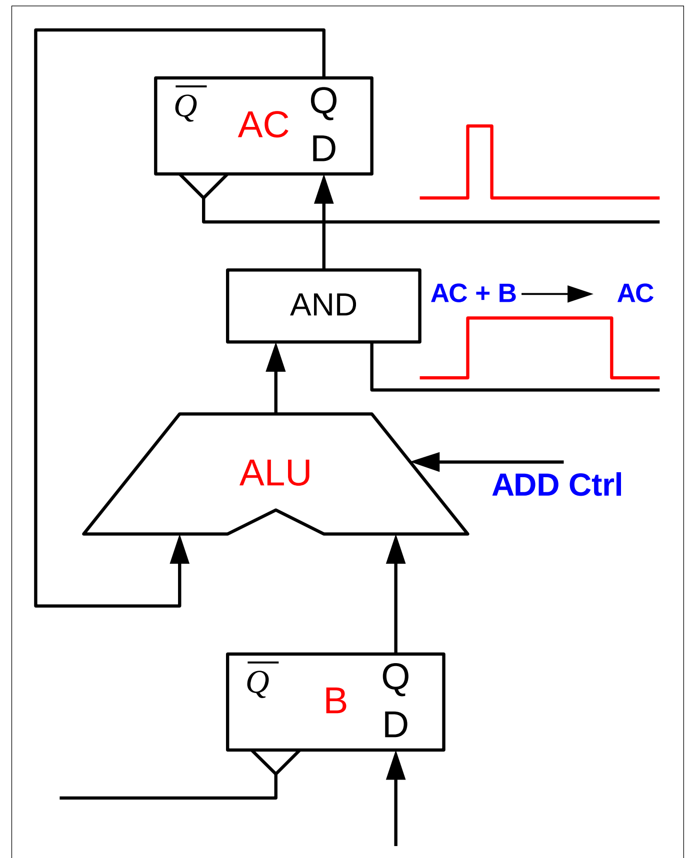
- 程序：在此特指一段机器指令序列。
  - 完成一定的功能，采用某种算法，具备一定的流程；
  - 计算机按照程序所规定的流程和指令顺序，一条一条地执行指令，达到完成程序所规定的功能的目的。
  - 计算机采用程序计算器（Program Counter）来决定指令执行的顺序。

3.2 指令的执行过程

- 微操作：计算机可以完成的最基本的操作，一条机器指令的执行可以解释为一系列的微操作的执行
  - 操作性质：对数据进行某种处理
  - 操作对象
  - 操作的时间与条件



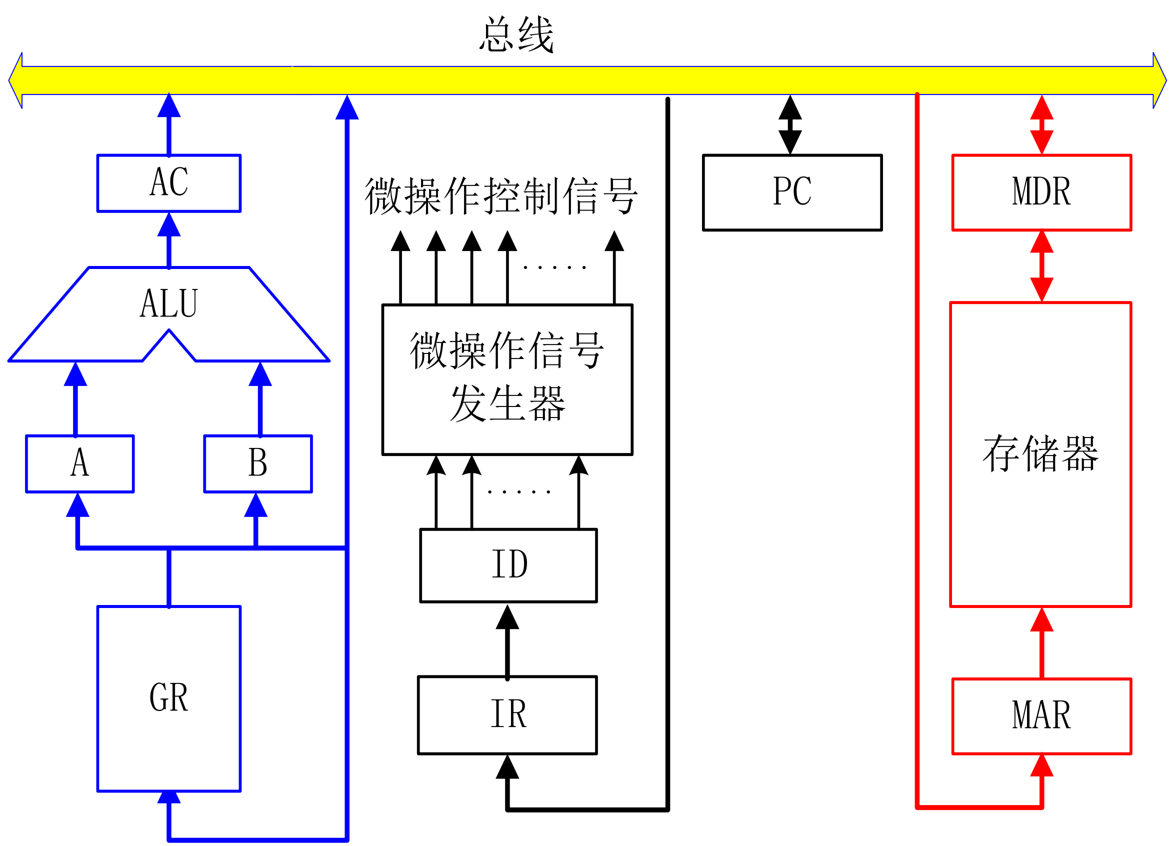
微操作：RegA→RegB



微操作：AC +RegB→ AC

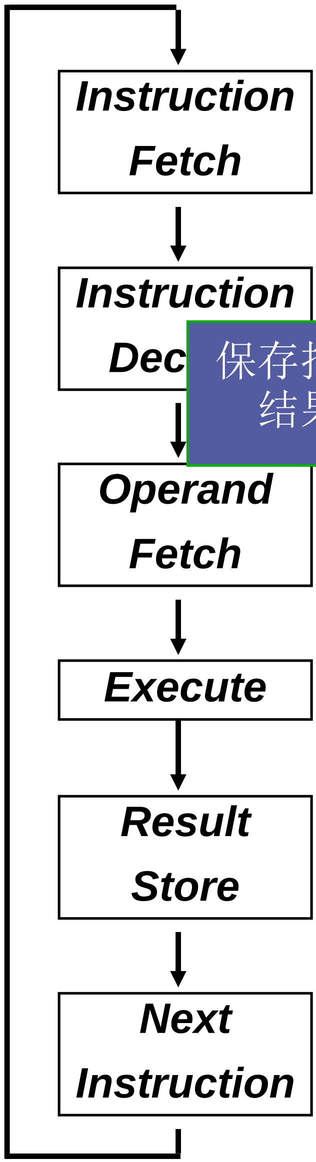
3.2 指令的执行过程

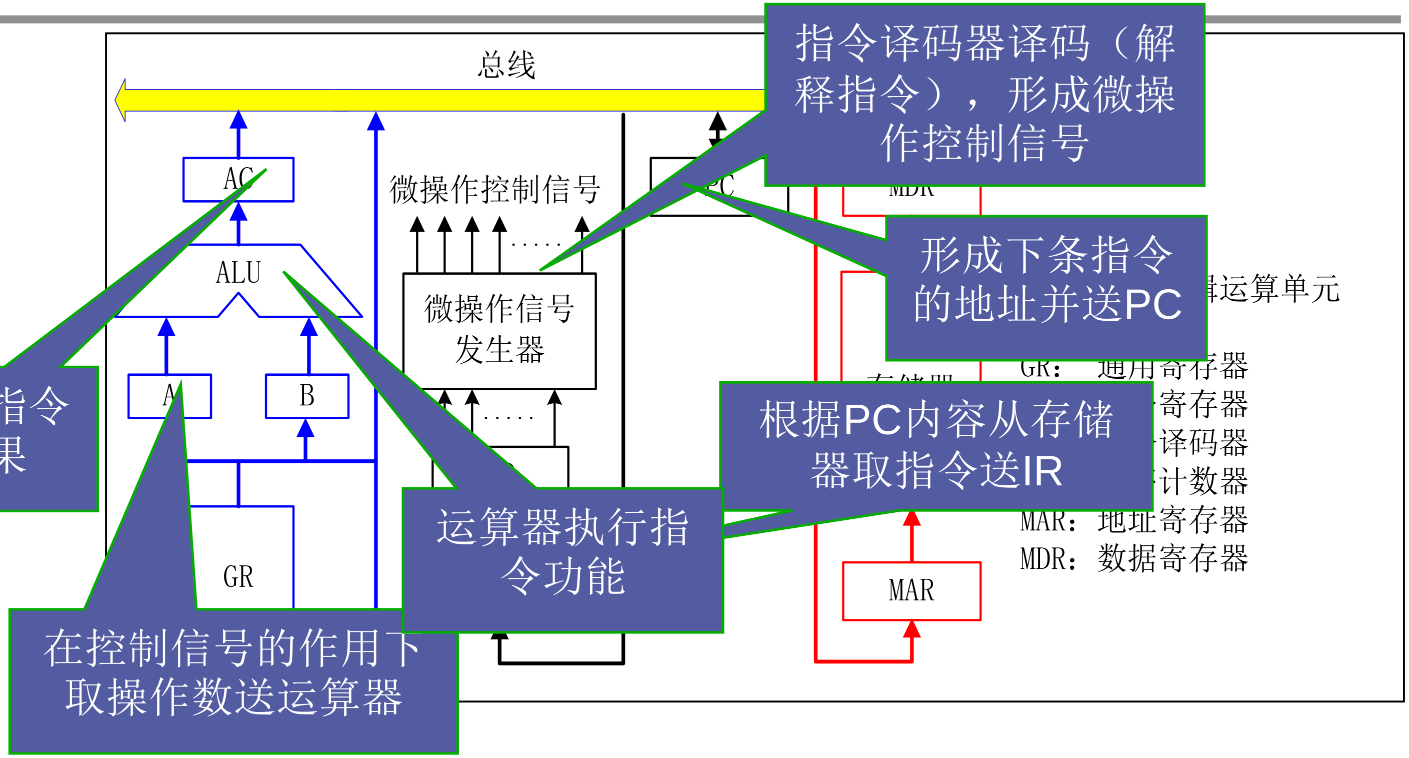
机器结构简化图



AC：　累加器
ALU：算术逻辑运算单元
A，B：缓冲器
GR：　通用寄存器
IR：　指令寄存器
ID：　指令译码器
PC：　程序计数器
MAR：地址寄存器
MDR：数据寄存器

3.2 指令的执行过程





1. 根据PC内容从存储器取指令送IR；
2. 指令译码器译码（解释指令），形成微操作控制信号；
3. 在控制信号的作用下取操作数送运算器；
4. 运算器运行；
5. 保存结果；
6. 形成下一条指令地址送PC。

Example

$Y=ax^2+bx-c$ 假定a,b,c,x均为已知数，且已存放在内存，求y。

假定指令系统：16位指令系统

Opcode	Address
8	8

指令	操作码	说明
ADD	00H	AC ← (AC)＋Mem(Add)
LD	01H	AC ← Mem(Add)
SUB	02H	AC ← (AC) — Mem(Add)
MUL	03H	AC ← (AC)×Mem(Add)
ST	04H	Mem(Add) ← (AC)

内存	地址
	00H
	02H
	04H
	06H
	08H
	0AH
	0CH
	0EH
结果y将存放在此	10H
值a	12H
值b	14H
值c	16H
值x	18H

Example

$Y=ax^2+bx-c$ 假定a,b,c,x均为已知数，且存放在内存中，求y。

指令	操作码	说明
ADD	00H	AC ← (AC)＋Mem(Add)
LD	01H	AC ← Mem(Add)
SUB	02H	AC ← (AC) — Mem(Add)
MUL	03H	AC ← (AC)×Mem(Add)
ST	04H	Mem(Add) ← (AC)

程序如下

指令	代码	
LD a	0112H	AC ← a
MUL x	0318H	AC ← ax
ADD b	0014H	AC ← ax + b
MUL x	0318H	AC ← ax^2+bx
SUB c	0216H	AC ← ax^2 + bx - c
ST y	0410H	Mem ← (AC)

内存

内存	地址
	00H
	02H
	04H
	06H
	08H
	0AH
	0CH
	0EH
结果y将存放在此	10H
值a	12H
值b	14H
值c	16H
值x	18H

Example

AC $\boxed{ax^2+bx-c}$

		内存	地址
开始PC	00H	0112H	00H
PC	02H	0318H	02H
PC	04H	0014H	04H
PC	06H	0318H	06H
PC	08H	0216H	08H
PC	0AH	0410H	0AH
PC	0CH		0CH
			0EH
		ax^2+bx-c	10H
		值a	12H
		值b	14H
		值c	16H
		值x	18H

程序如下

指令	代码	
LD a	0112H	AC ← a
MUL x	0318H	AC ← ax
ADD b	0014H	AC ← ax + b
MUL x	0318H	AC ←ax^2+bx
SUB c	0216H	AC ←$ax^2 + bx - c$
ST y	0410H	Mem ← (AC)

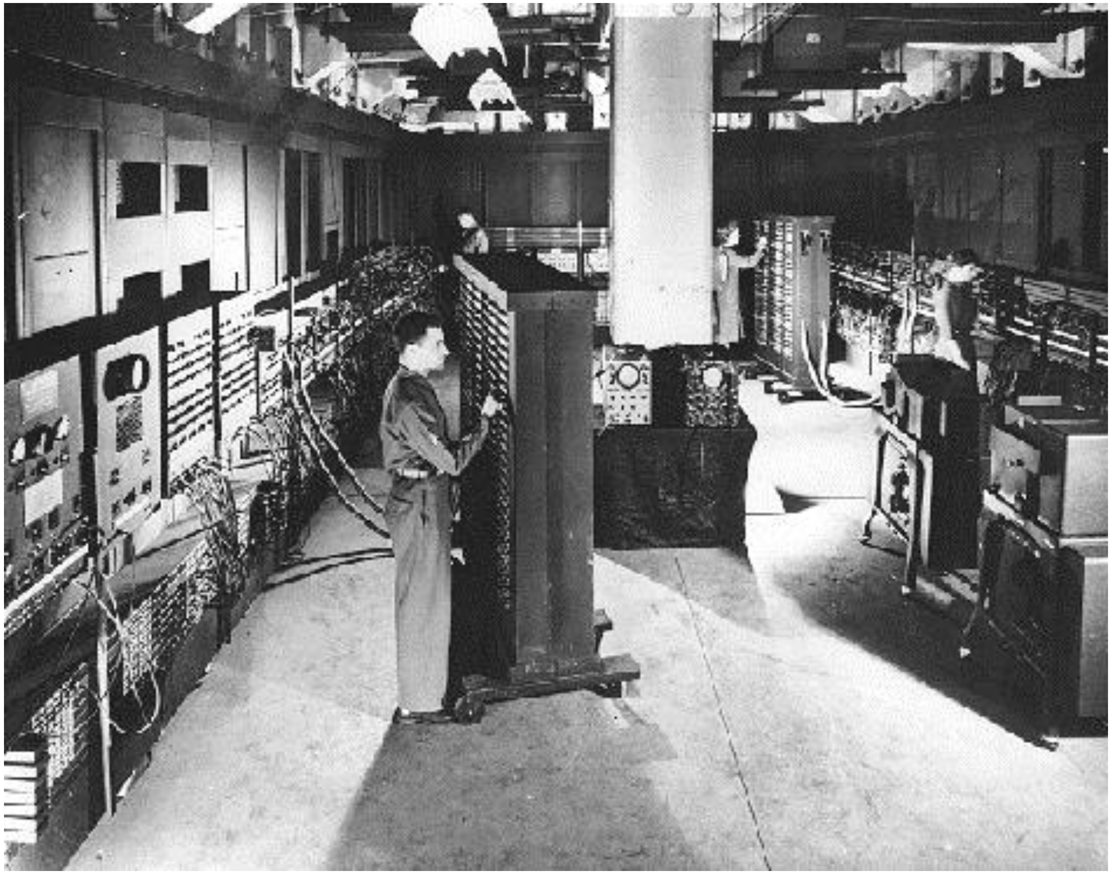
冯．诺依曼计算机结构的特点

❖主要特点

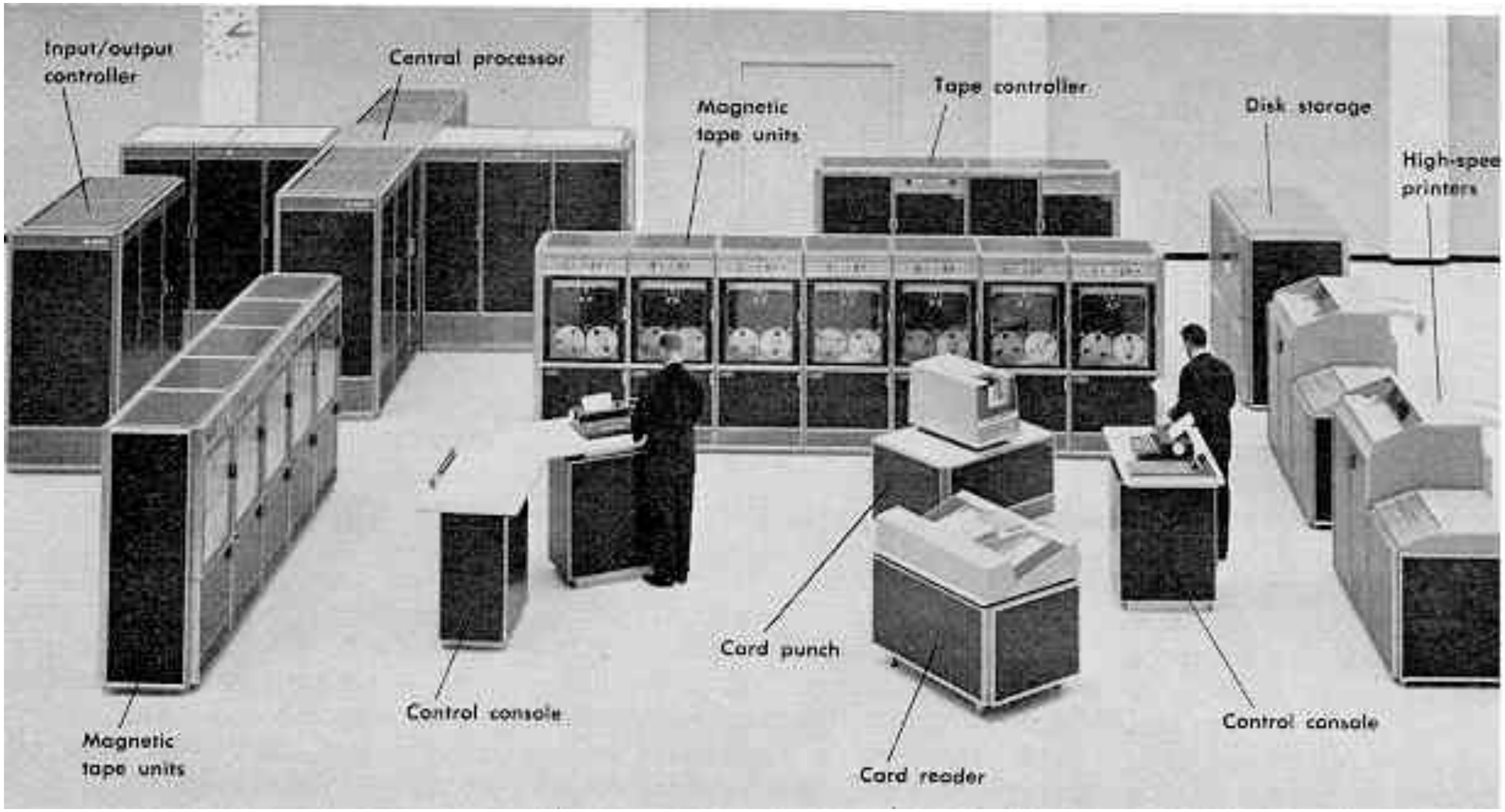
- 计算机由运算器、存储器、控制器和输入输出部分组成
- 指令和数据用二进制表示，两者在形式上没有差别
- 指令和数据存放在存储器中，按地址访问
- 指令有操作码和地址码两个部分组成，操作码指定操作性质，地址码指定操作数位置
- 采用“存储程序”方式进行工作

ENIAC（1946）

- ENIAC：十进制（而非二进制）计算机，用十个真空管（一个ON，其余OFF）表示一位十进制数，算术运算按十进制的方式完成。
- 占地170平方米，重30吨，耗电140千瓦，共用18000个真空管，每秒可进行5000次加减法运算。



Mainframe Era: 1950s - 1960s



Enabling Tech: Computers

Big Players: “Big Iron” (IBM, UNIVAC)

Cost: $1M, Target: Businesses

Using: COBOL, Fortran, timesharing OS

Minicomputer Era: 1970s

Enabling Tech: Integrated circuits

Big Players: Digital, HP

Cost: $10k, Target: Labs & universities

Using: C, UNIX OS

PC Era: Mid 1980s - Mid 2000s

Enabling Tech: Microprocessors

Big Players: Apple, IBM

Cost: $1k, Target: Consumers (1/person)

Using: Basic, Java, Windows OS

Post-PC Era: Late 2000s - ???

Personal Mobile Devices (PMD):

Enabling Tech: Wireless networking, smartphones

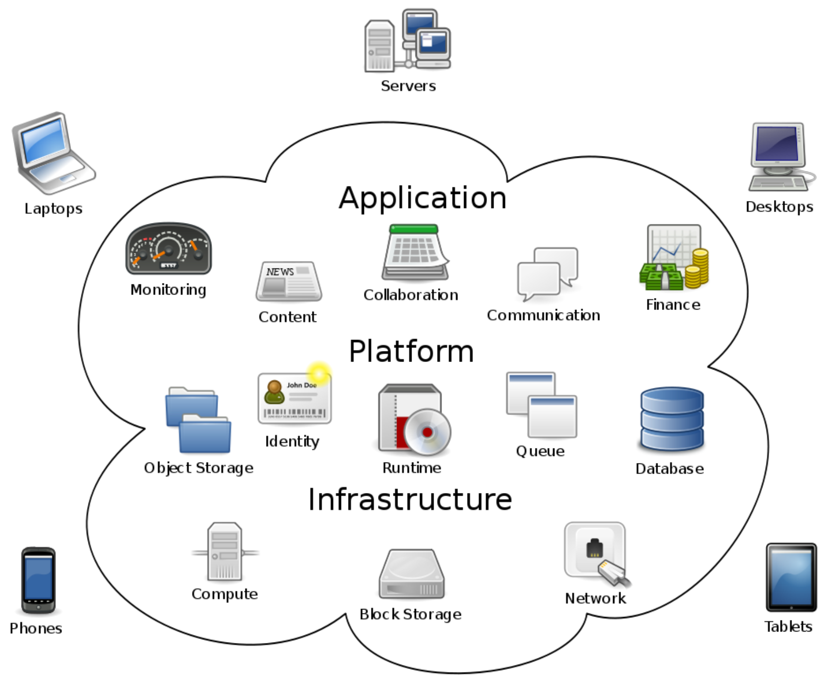
Big Players: Apple, Nokia, …

Cost: $500, Target: Consumers on the go

Using: Objective C, Android OS

Post-PC Era: Late 2000s - ???

Cloud Computing:



Enabling Tech: Local Area Networks, broadband Internet

Big Players: Amazon, Google, …

Target: Transient users or users who cannot afford high-end equipment

Moore's Law

The experts look ahead

Cramming more components onto integrated circuits

With unit cost falling as the number of components per circuit rises, by 1975 economics may dictate squeezing as many as 65,000 components on a single silicon chip

By Gordon E. Moore

Director, Research and Development Laboratories, Fairchild Semiconductor division of Fairchild Camera and Instrument Corp.

The future of integrated electronics is the future of electronics itself. The advantages of integration will bring about a proliferation of electronics, pushing this science into many new areas.

Integrated circuits will lead to such wonders as home computers—or at least terminals connected to a central computer—automatic controls for automobiles, and personal portable communications equipment. The electronic wristwatch needs only a display to be feasible today.

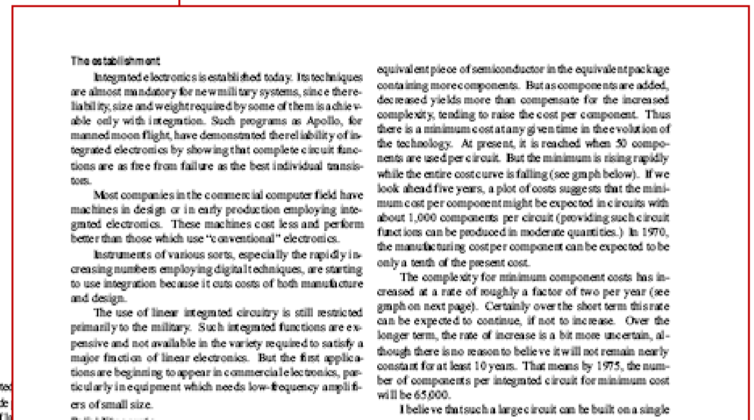
But the biggest potential lies in the production of large systems. In telephone communications, integrated circuits in digital filters will separate channels on multiplex equipment. Integrated circuits will also switch telephone circuits and perform data processing.

Computers will be more powerful, and will be organized in completely different ways. For example, memories built of integrated electronics may be distributed throughout the

The author

Dr. Gordon E. Moore is one of the new breed of electronic engineers, schooled in the physical sciences rather than in electronics. He earned a B.S. degree in chemistry from the University of California and a Ph.D. degree in physical chemistry from the California Institute of Technology. He was one of the founders of Fairchild Semiconductor and has been director of the research and development laboratories since 1959.

Electronics, Volume 38, Number 8, April 19, 1965



The establishment

Integrated electronics is established today. Its techniques are almost mandatory for new military systems, since the reliability, size and weight required by some of them is achievable only with integration. Such programs as Apollo, for manned moon flight, have demonstrated the reliability of integrated electronics by showing that complete circuit functions are as free from failure as the best individual transistors.

Most companies in the commercial computer field have machines in design or in early production employing integrated electronics. These machines cost less and perform better than those which use "conventional" electronics.

Instruments of various sorts, especially the rapidly increasing numbers employing digital techniques, are starting to use integration because it cuts costs of both manufacture and design.

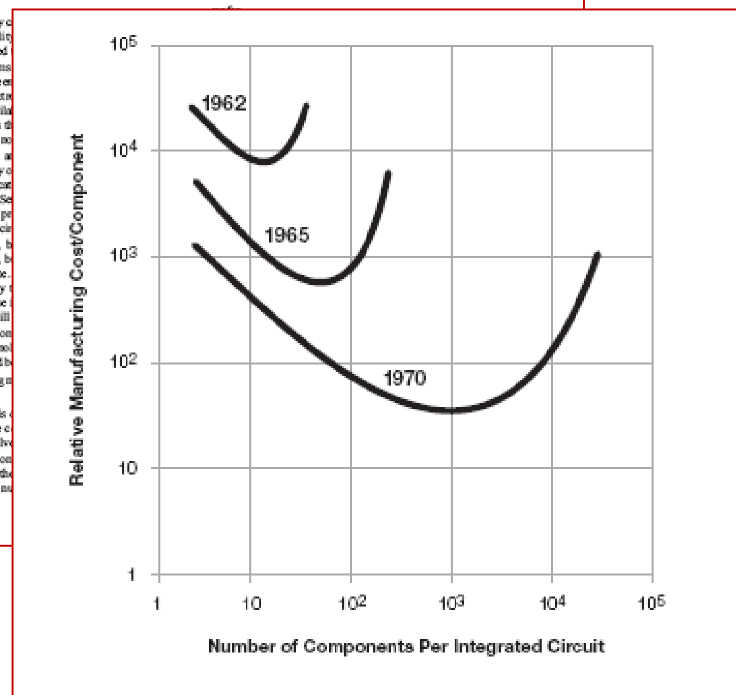
The use of linear integrated circuitry is still restricted primarily to the military. Such integrated functions are expensive and not available in the variety required to satisfy a major fraction of linear electronics. But the first applications are beginning to appear in commercial electronics, particularly in equipment which needs low-frequency amplifiers of small size.

Reliability counts

equivalent piece of semiconductor in the equivalent package containing more components. But as components are added, decreased yields more than compensate for the increased complexity, tending to raise the cost per component. Thus there is a minimum cost at any given time in the evolution of the technology. At present, it is reached when 50 components are used per circuit. But the minimum is rising rapidly while the entire cost curve is falling (see graph below). If we look ahead five years, a plot of costs suggests that the minimum cost per component might be expected in circuits with about 1,000 components per circuit (providing such circuit functions can be produced in moderate quantities.) In 1970, the manufacturing cost per component can be expected to be only a tenth of the present cost.

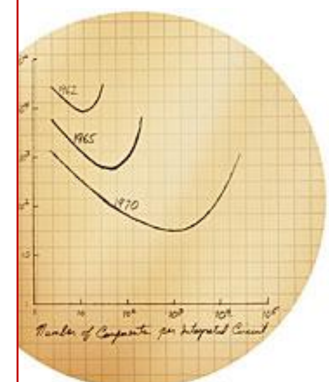
The complexity for minimum component costs has increased at a rate of roughly a factor of two per year (see graph on next page). Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000.

I believe that such a large circuit can be built on a single



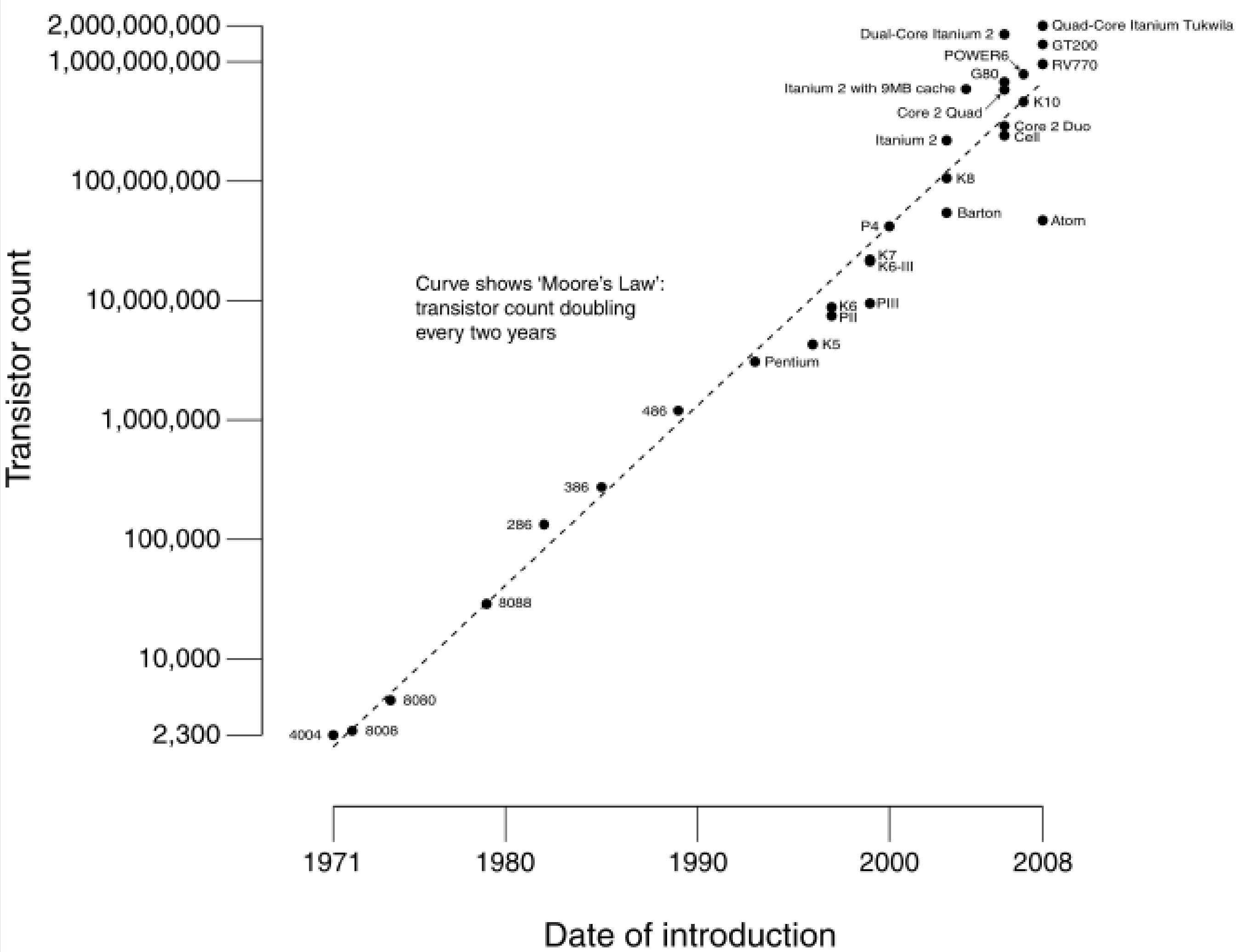
Electronics, Volume 38, Number 8, April 19, 1965

Gordon E. Moore



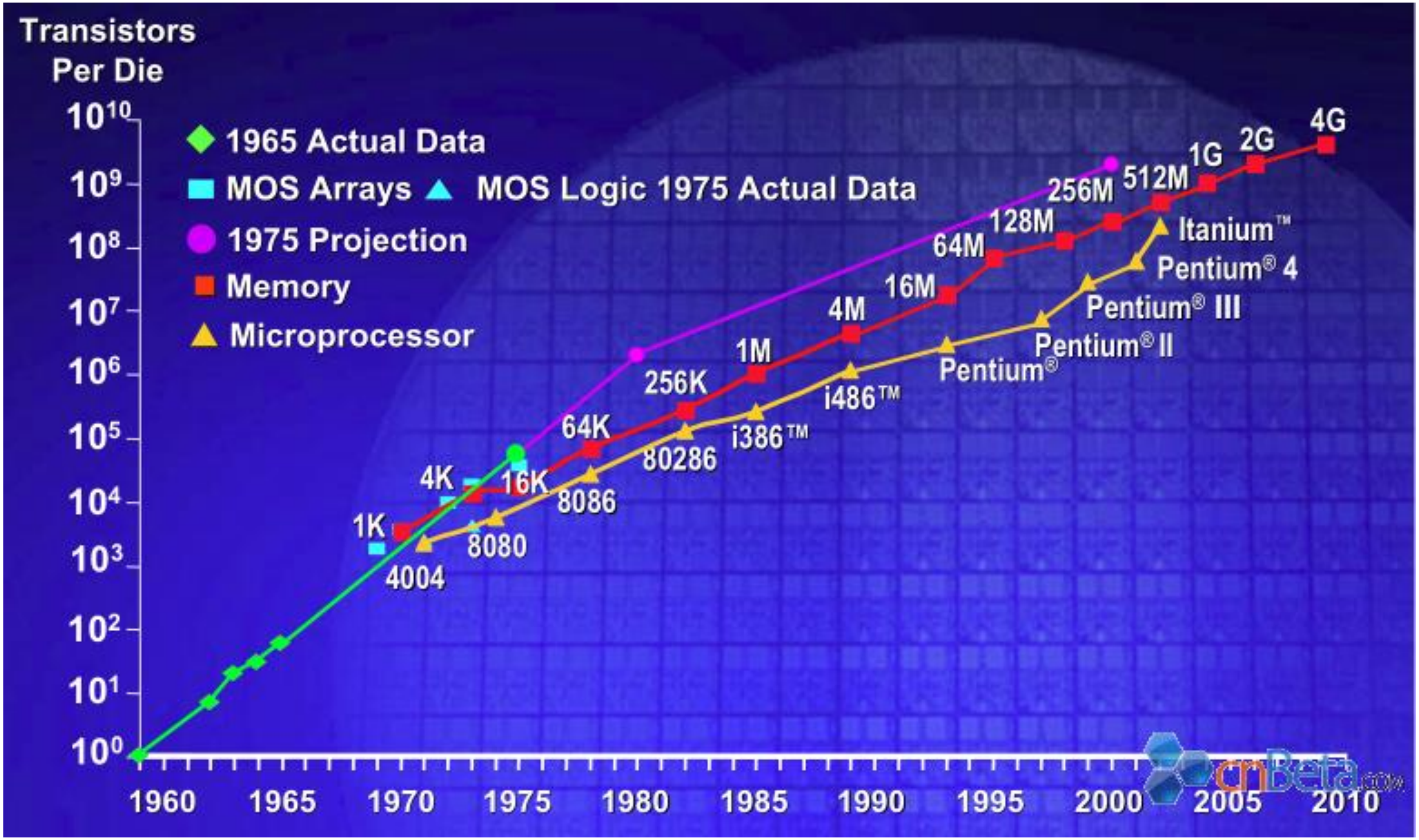
Moore’s Law

CPU Transistor Counts 1971-2008 & Moore’s Law



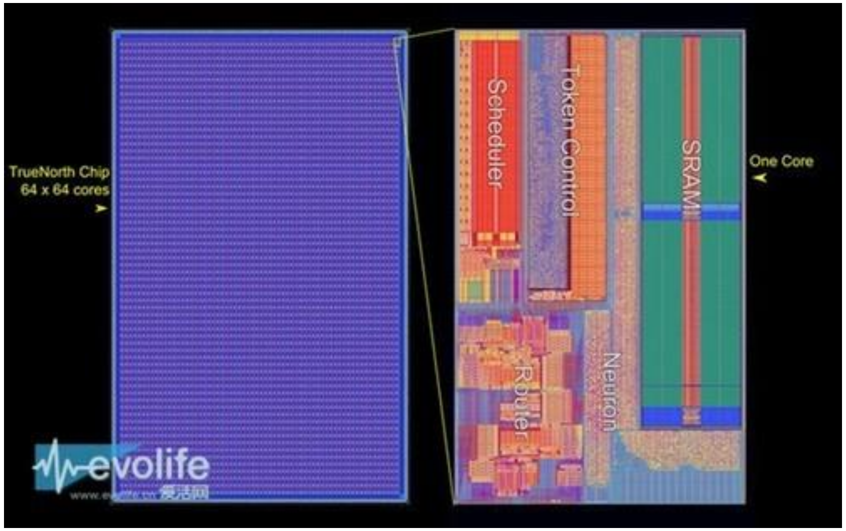
Moore's Law

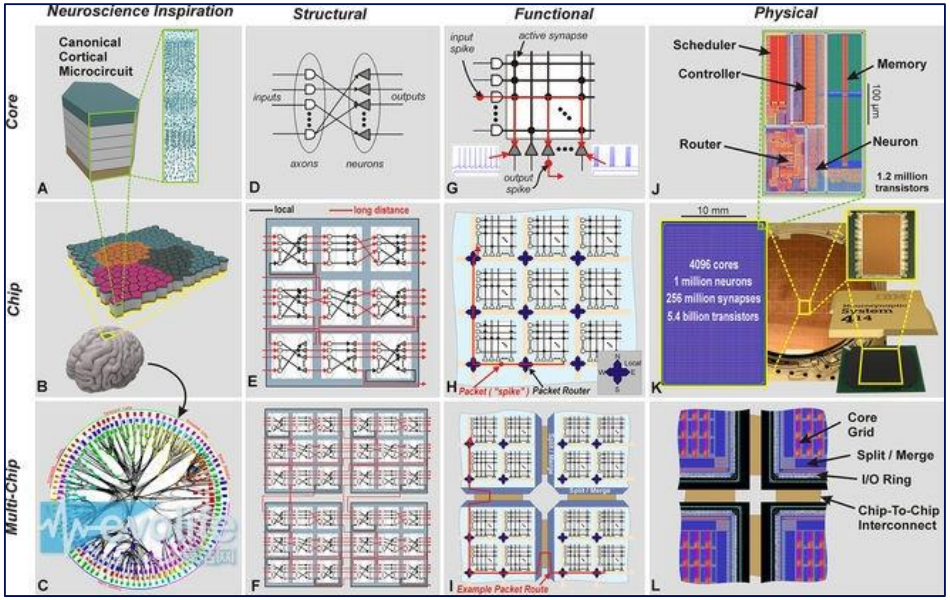
iSuppli公司的首席分析师Len Jelinek称：**摩尔定律将可能在2014年到达极限**，当芯片工艺从20纳米向18纳米进军时，半导体的工艺技术将达到芯片制造的极限。



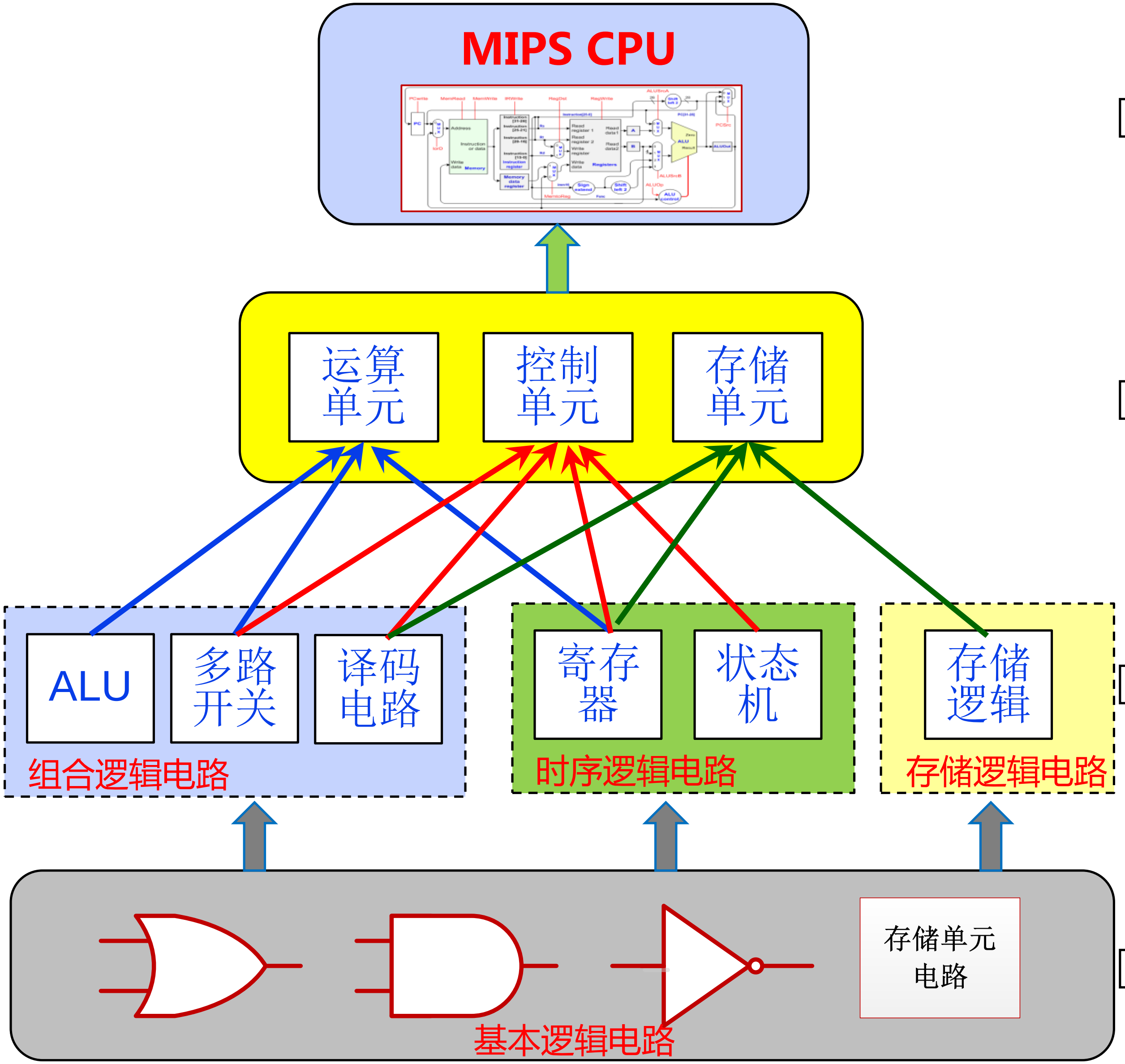
黑科技：IBM超级神经元芯片TrueNorth（仿人脑芯片）

- 邮票大小、重量几克
- 54 亿个硅晶体管
- 4096 个内核
- 100 万个 “神经元”
- 2.56 亿个 “突触”
- 相当于一台超级计算机
- 功耗 65 毫瓦





课程教学内容视图



MIPS 指令系统
单周期MIPS CPU
多周期MIPS CPU
流水线MIPS CPU

运算单元与方法
存储系统(CACHE)
控制单元设计

存储器结构
组合逻辑电路
时序逻辑电路

存储单元电路
基本逻辑电路
布尔逻辑（代数）