1. 计算机系统概论

1.1. 计算机系统**基本组成和原理**
1.2. 计算机系统**层次结构**
1.3. 计算机系统**性能评价**

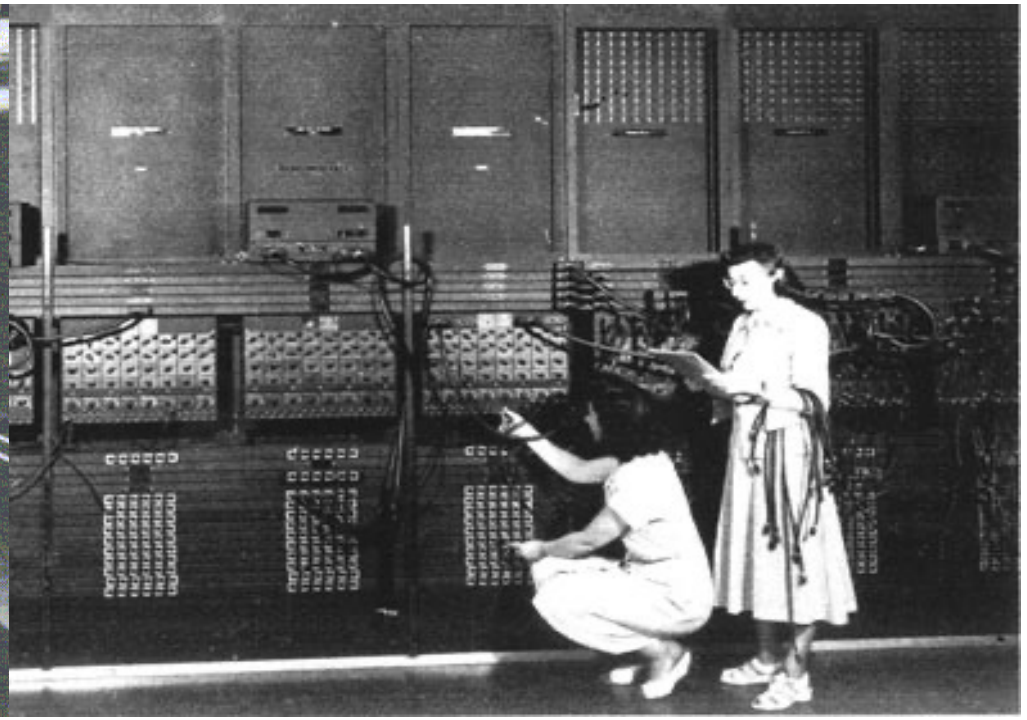
1. 计算机系统概论

第一台通用电子计算机的诞生

- 1946年，第一台通用电子计算机ENIAC诞生
  - 由电子真空管组成
  - 由美国宾夕法尼亚大学研制
  - 用于解决复杂弹道计算问题
  - 5000次加法/s
  - 平方、立方、sin、cos等
  - 用十进制表示信息和运算
  - 手动编程，通过设置开关和插拔电缆实现

Electronic Numerical Integrator And Computer
电子数字积分计算机

第一台通用电子计算机的诞生



现代计算机的原型

- ENIAC
  - 没有存储器
  - 基于十进制的表示和计算都非常麻烦
  - 制造和使用都非常困难
- EDVAC
  - 存储程序通用电子计算机方案（存储程序计算机）
  - 由普林斯顿高等研究院（IAS）研制
  - IAS计算机，1946年开始设计，1951年完成
- EDSAC
  - 1949年，**第一台存储程序计算机**
  - 由英国剑桥大学研制

冯·诺伊曼
计算机之父

现代计算机的原型（IAS计算机）

- “关于EDVAC的报告草案”，1946年
  - 计算机结构：冯·诺伊曼结构
  - 冯·诺伊曼结构的核心思想：“存储程序”工作方式

任何要计算机完成的工作都要先被编写成程序，然后将程序和原始数据送入主存并启动执行。一旦程序被启动，计算机应能在不需操作人员干预下，自动完成逐条取出指令和执行指令的任务。

  - 采用冯·诺依曼结构的计算机：冯·诺依曼计算机
    - 几乎所有现代通用计算机

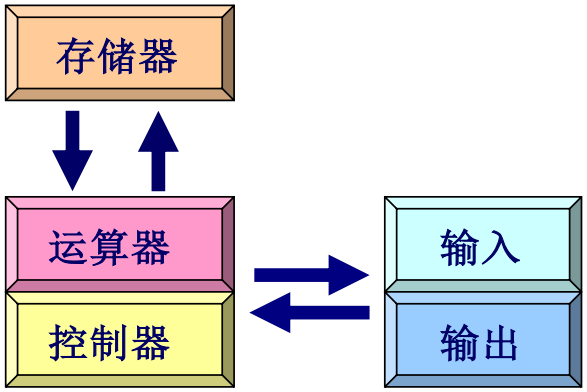
冯·诺伊曼
计算机之父

冯·诺伊曼结构的主要思想

- 计算机应由5个基本部件组成：运算器、控制器、存储器、输入设备、输出设备。
- 基本部件功能
  - 存储器不仅能存放数据，而且也能存放指令，形式上两者没有区别，但计算机应能区分数据还是指令；
  - 控制器应能自动取出指令来执行；
  - 运算器应能进行加/减/乘/除四种基本算术运算，并且也能进行一些逻辑运算和附加运算；
  - 操作人员可以通过输入设备、输出设备和主机进行通信。
- 内部以二进制表示指令和数据。每条指令由操作码和地址码两部分组成。操作码指出操作类型，地址码指出操作数的地址。由一串指令组成程序。
- 采用“存储程序”工作方式。

冯·诺伊曼结构

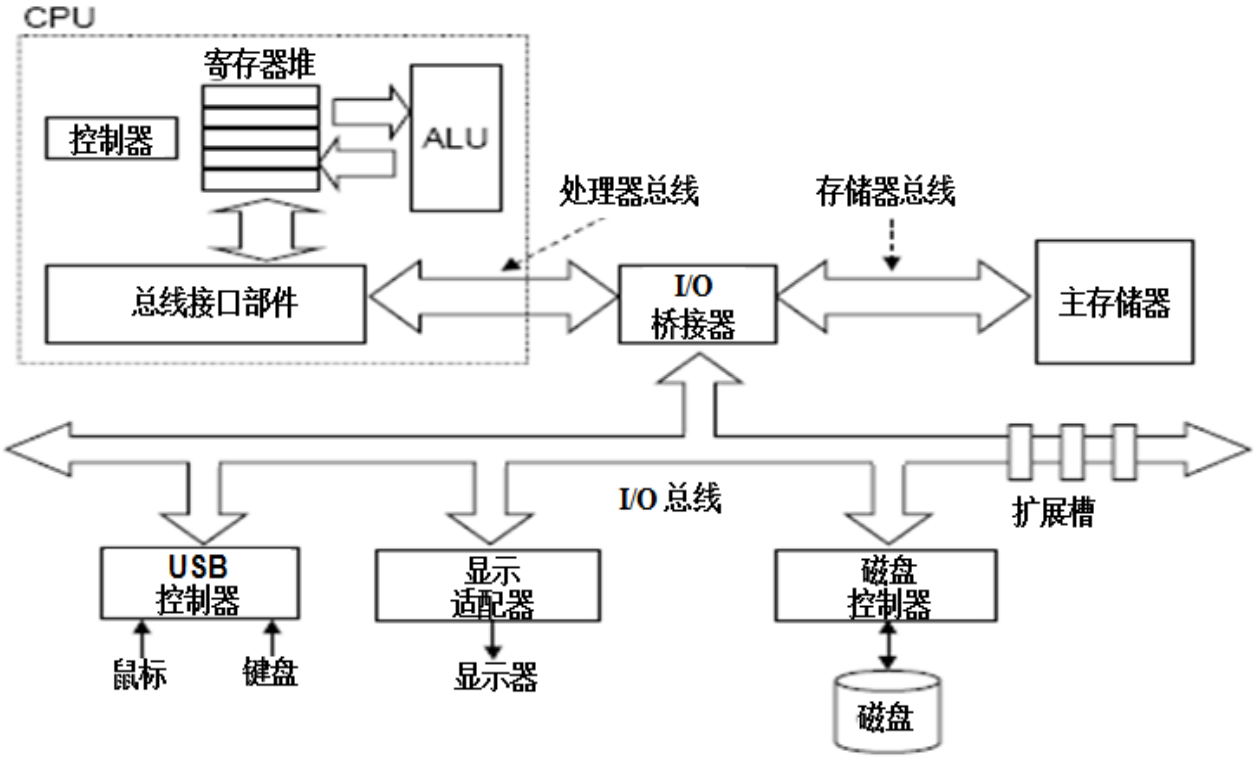
硬件



软件

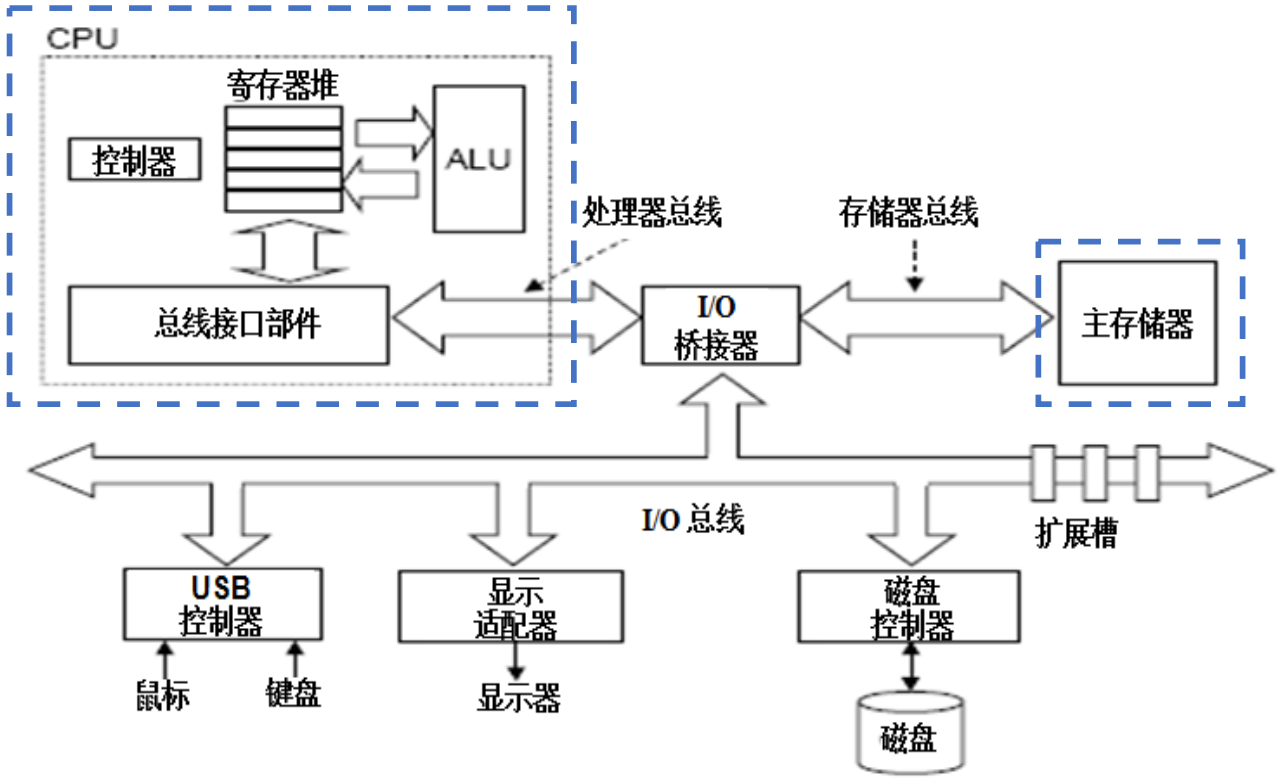
- 系统软件：操作系统，程序设计语言及其编辑、编译软件…
- 应用软件：为解决某一实际问题而编制的软件

现代计算机结构模型



以CPU为核心通过3条总线连接存储器和I/O接口，构成计算机以其为主体，配上系统软件和外设，构成**计算机系统**。

现代计算机结构模型



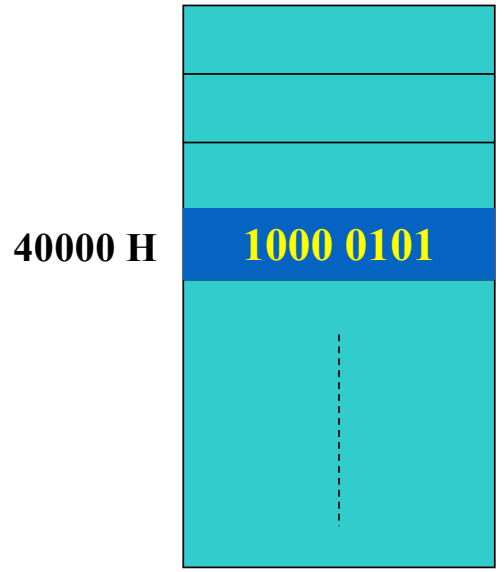
- CPU：中央处理器是超大规模集成电路，内部集成了运算器、控制器、寄存器组…
- 存储器：指系统的主存储器，简称为内存或主存。用来存放程序和数据。

现代计算机结构模型：存储器

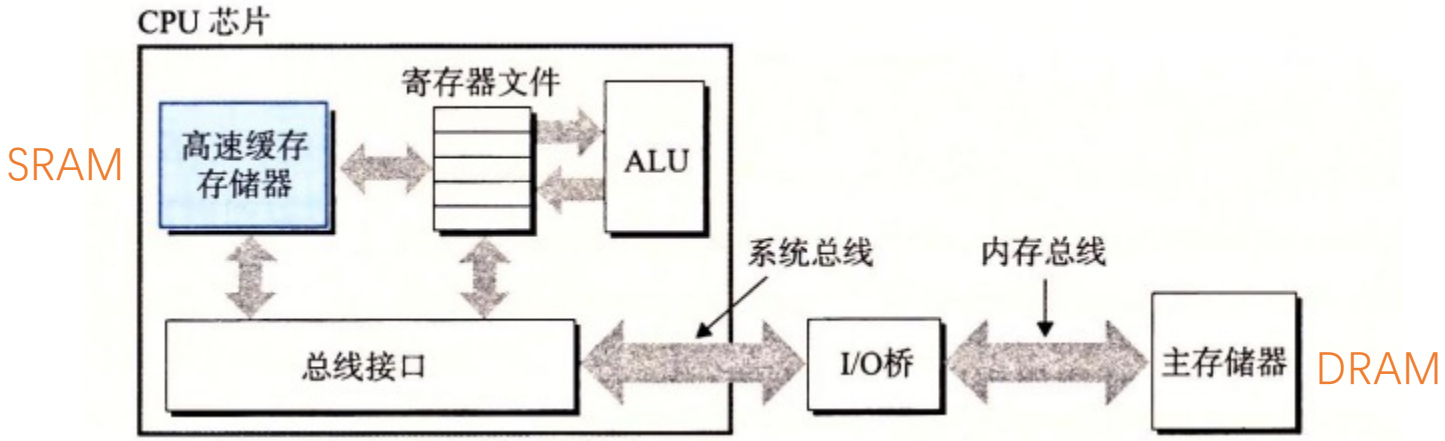
- 存储器
  - 由若干“存储单元”组成，每一单元存放一个“字节”的信息
  - 字节（byte，B）
    - 由8个二进制位（bit，b）组成，
    - 常用的存储容量计算单位

    - 2字节=1“字”
    - 4字节=1“双字”

    - 1KB=1024B
    - 1MB=1024KB
    - 1GB=1024MB
    - 1TB=1024GB
    - 1PB=1024TB



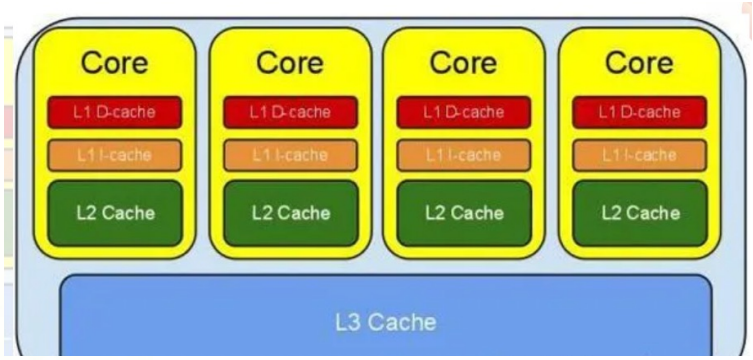
现代计算机结构模型：存储器

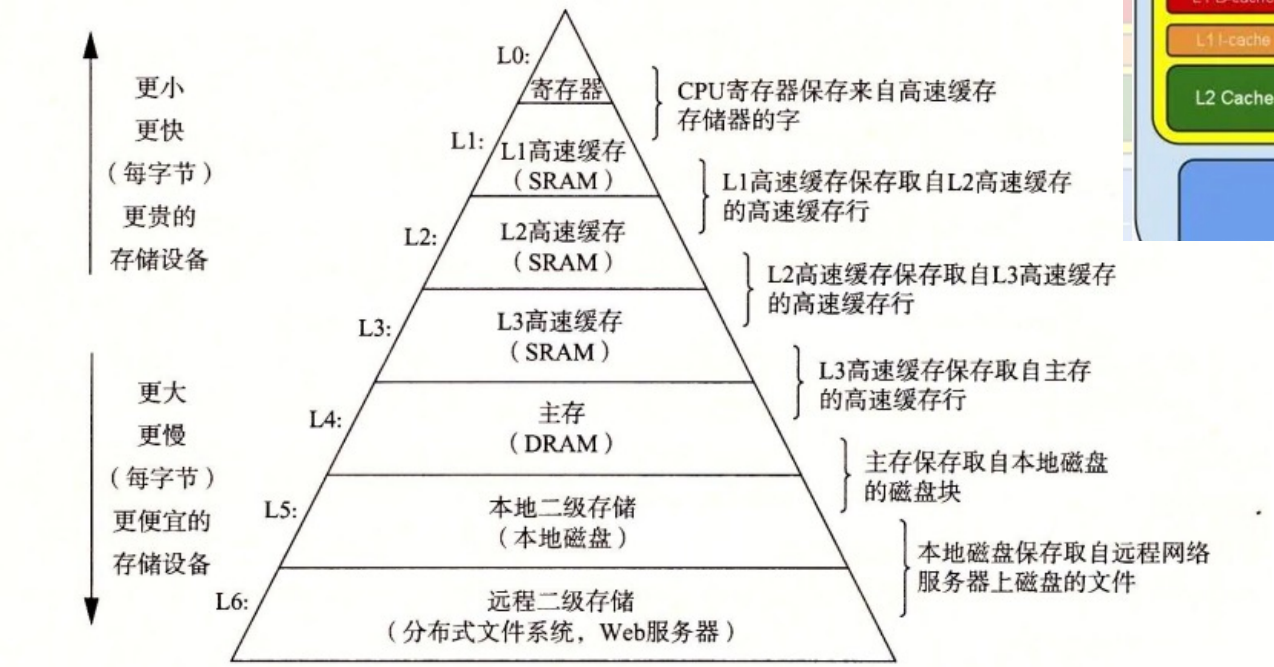


- RAM：随机存储器，习惯上称为“内存”，CPU执行指令可对其进行读、写操作。
  - SRAM（静态RAM）：基于触发器，成本高，容量小，信息稳定，读写速度快。
  - DRAM（动态RAM）：基于电容，成本低，容量大，信息存储不稳定，需要进行“刷新”操作。

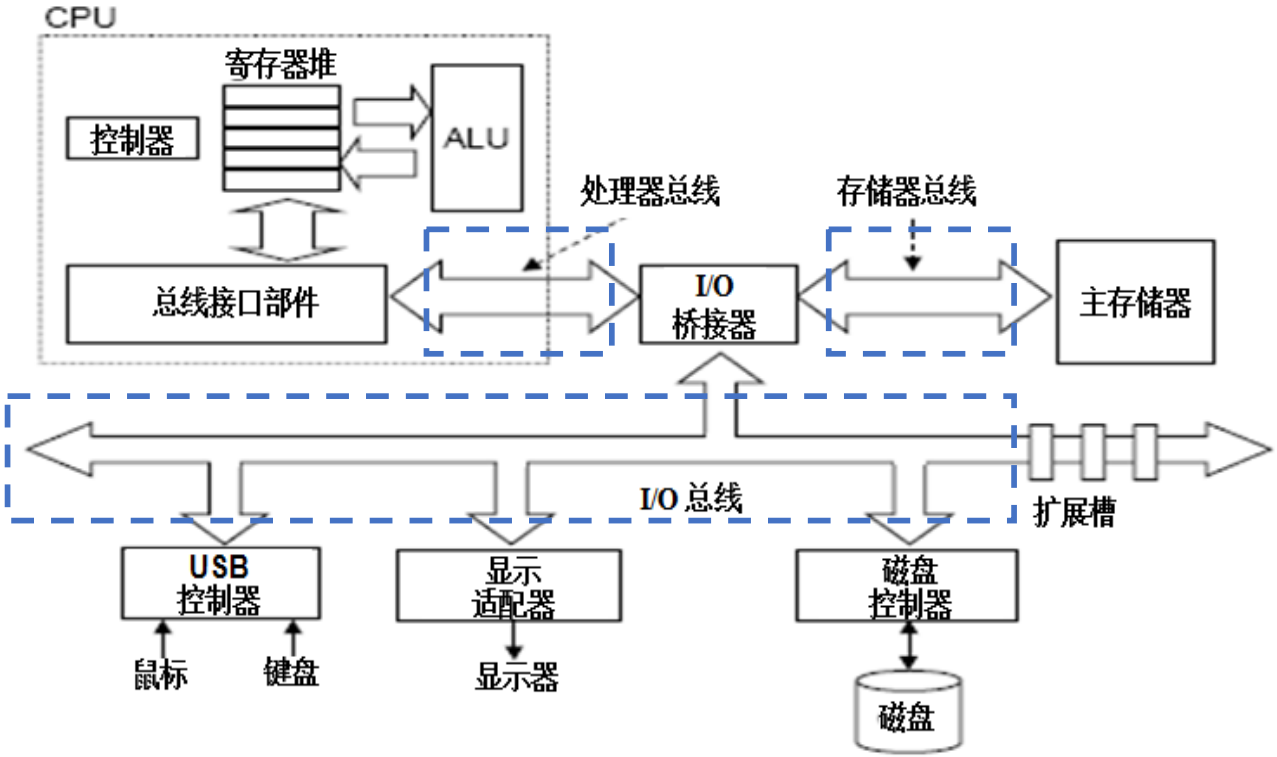
现代计算机结构模型：存储器

- 存储器分层结构



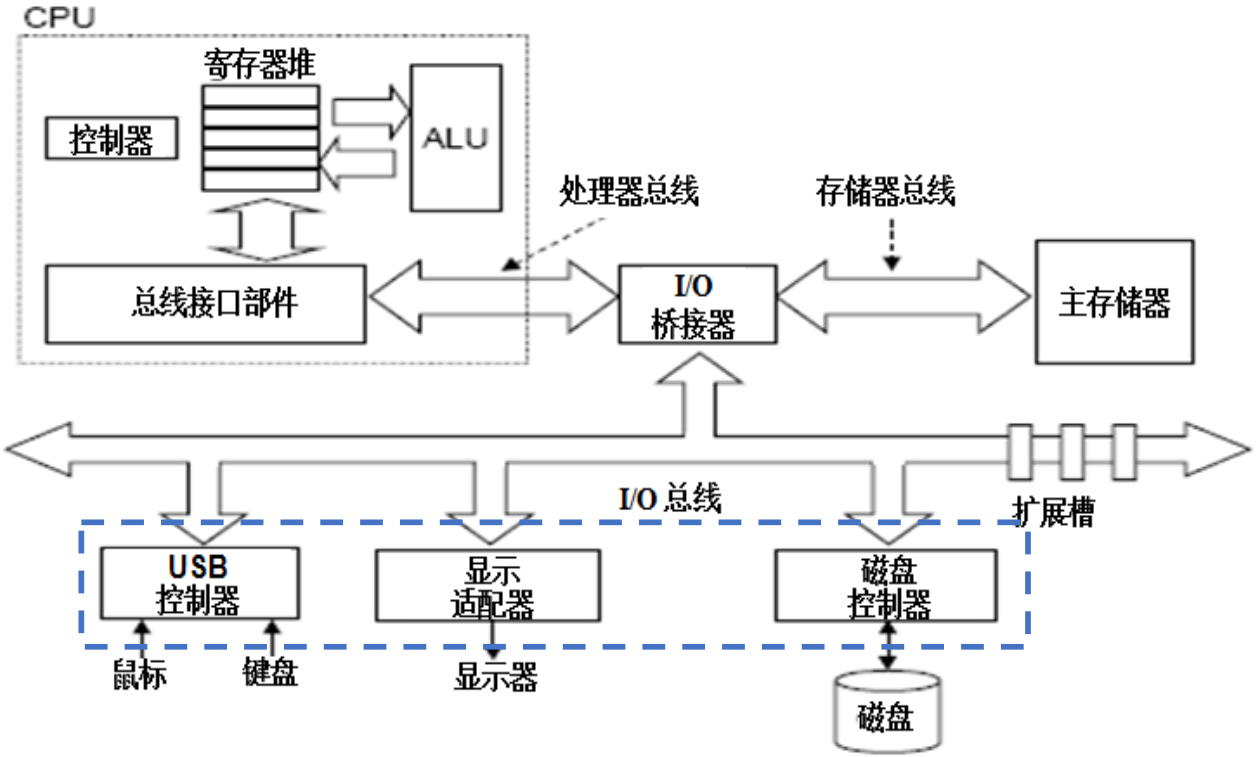


现代计算机结构模型



- 总线：CPU与存储器、I/O接口交换信息的公共通道。
  - 设计成传送定长的字节块，即字长

现代计算机结构模型



- 控制器、适配器（I/O接口）：CPU和外部设备交换信息的“中转站”
  - 外设不能直接与CPU交换信息，必须通过接口

计算机是如何工作的?

存储程序计算机:

任何要计算机完成的工作都要先被编写成程序，然后将程序和原始数据送入主存并启动执行。一旦程序被启动，计算机应能在不需操作人员干预下，自动完成逐条取出指令和执行指令的任务。

- 程序由指令组成，所有指令执行完，则程序结束。
- 程序在执行前
  - 数据和指令事先存放在存储器中
  - 每条指令和每个数据都有地址
  - 指令按序存放，由操作码、操作数组成
  - 程序起始地址IP（Instruction Pointer）

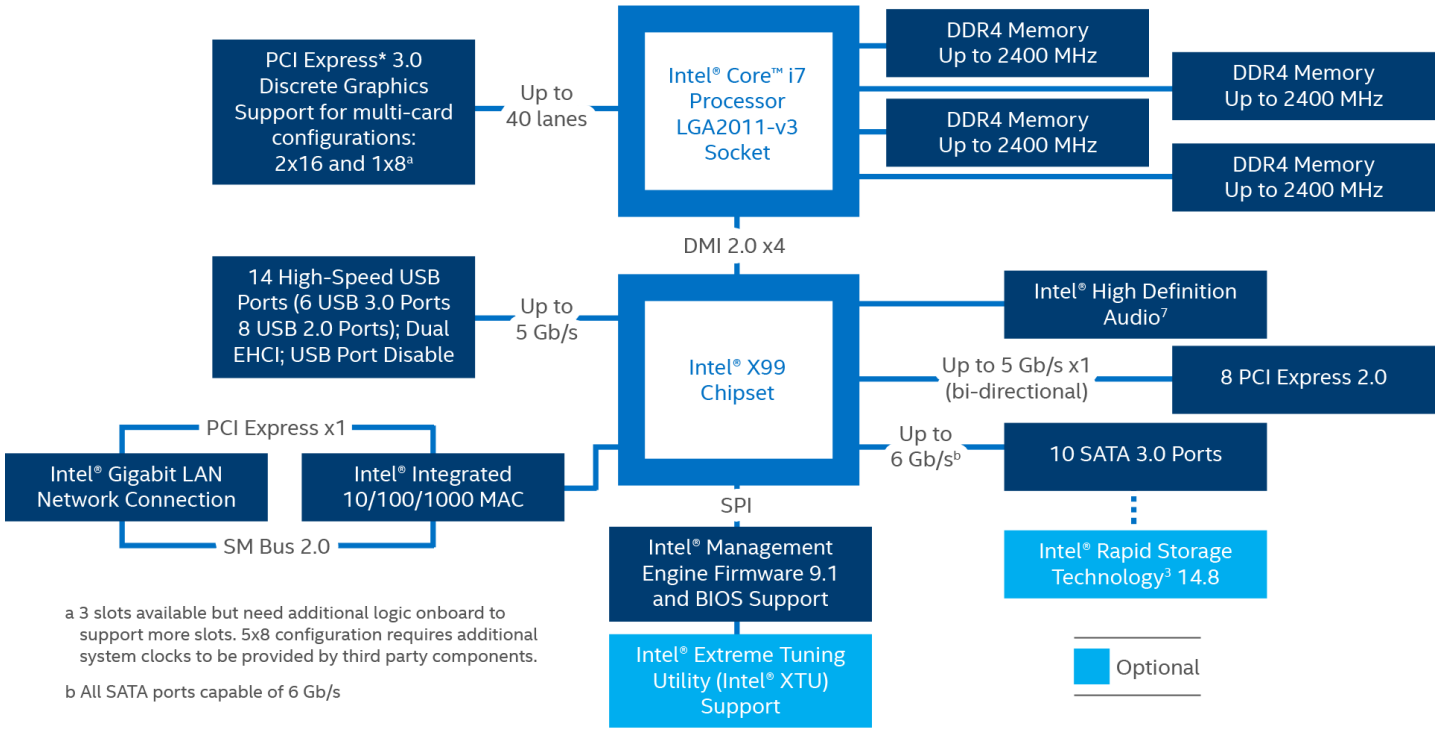
- 程序开始执行

  第一步：根据IP取指令
  第二步：指令译码
  第三步：取操作数
  第四步：指令执行
  第五步：回写结果
  第六步：修改IP的值
  继续执行下一条指令

微型计算机与Intel处理器

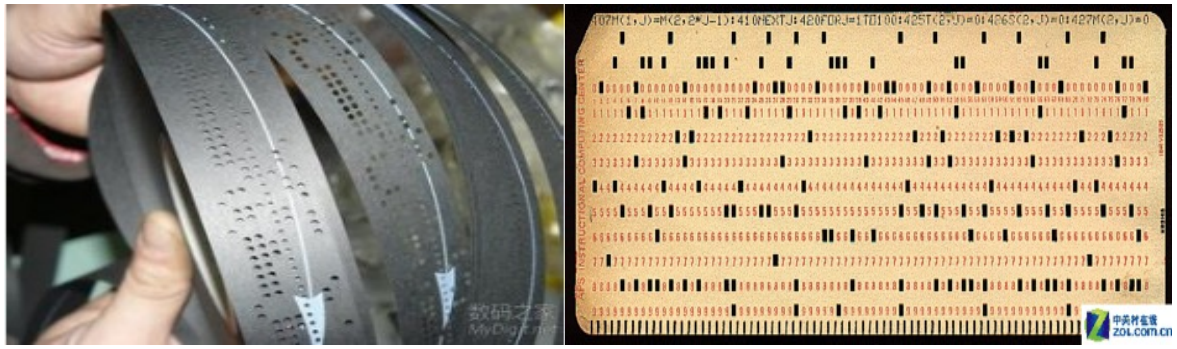
- 巨型机、大型机、中型机、小型机、微型计算机
- 微型计算机
  - 以大规模、超大规模集成电路为主要部件
  - 以集成了控制器和运算器的微处理器为核心

x86-64架构


PCI Express* 3.0 Discrete Graphics Support for multi-card configurations: 2x16 and 1x8a
Up to 40 lanes
Intel® Core™ i7 Processor LGA2011-v3 Socket
DDR4 Memory Up to 2400 MHz
DDR4 Memory Up to 2400 MHz
DDR4 Memory Up to 2400 MHz
DDR4 Memory Up to 2400 MHz
DMI 2.0 x4
14 High-Speed USB Ports (6 USB 3.0 Ports 8 USB 2.0 Ports); Dual EHCI; USB Port Disable
Up to 5 Gb/s
Intel® X99 Chipset
Intel® High Definition Audio7
Up to 5 Gb/s x1 (bi-directional)
8 PCI Express 2.0
Up to 6 Gb/sb
10 SATA 3.0 Ports
PCI Express x1
Intel® Gigabit LAN Network Connection
Intel® Integrated 10/100/1000 MAC
SM Bus 2.0
SPI
Intel® Management Engine Firmware 9.1 and BIOS Support
Intel® Extreme Tuning Utility (Intel® XTU) Support
Intel® Rapid Storage Technology3 14.8
Optional
a 3 slots available but need additional logic onboard to support more slots. 5x8 configuration requires additional system clocks to be provided by third party components.
b All SATA ports capable of 6 Gb/s

系统软件：编程语言

- 第一代程序设计语言：机器语言
  - 穿孔：0
  - 未穿孔：1



不灵活！书写阅读困难！

应用程序

指令集体系结构

计算机硬件

0010：jxx（转移指令）

0：0101 0110
1：0010 0100
2：……
3：……
4：0110 0111
5：……
6：……

系统软件：编程语言

- 第二代程序设计语言：汇编语言
  - 用助记符和标号来表示指令

无需记忆指令码，编写方便

可读性比机器语言强

需要汇编器

将汇编语言转换为机器语言


```
0:  0101 0110        sub B
1:  0010 0100        jnz L0
2:  ……               ……
3:  ……               ……
4:  0110 0111        L0:  add C
5:  ……                   ……
6:  ……               B:   ……
                     C:   ……
```

系统软件：编程语言

- 第二代程序设计语言：汇编语言
  - 源程序由汇编指令构成
  - 指令包含操作码和操作数

面向机器结构（机器级语言）

- 需要描述太多底层细节
- 程序非常长
- 难以在不同结构的机器上运行


```
sub B
jnz L0
.....
.....
L0:  add C
     .....
B:   .....
C:   .....
```

系统软件：编程语言

- 高级语言 VS 机器级语言
  - 与具体机器结构无关
  - 面向算法描述，表达能力大大增强
  - 一条语句可能对应几条、几十条、甚至几百条指令
  - 发展出“面向过程”、“面向对象”等多种编程范式
- 语言处理系统
  - 语言处理程序（编译、汇编、链接）
  - 运行时系统（标准库、调试器、程序优化）

程序处理的转换过程

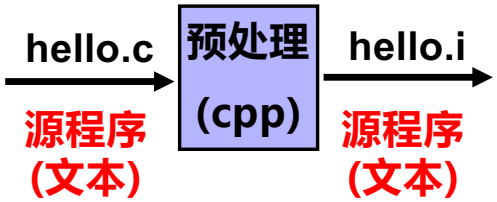
- 高级语言源程序 → ? → 机器语言程序

```
#include <stdio.h>

int main()
{
  printf("hello, world\n");
}
```

linux> ***gcc*** -o *hello hello.c* → ? → 输出“hello, world”

程序处理的转换过程



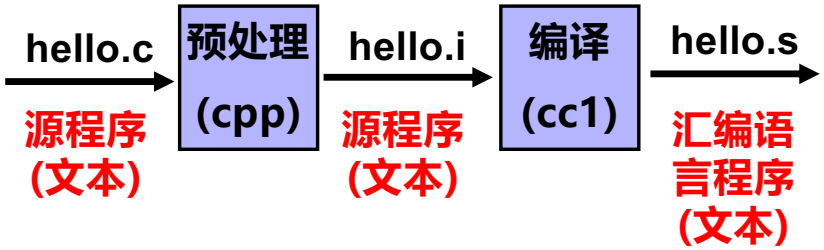
linux> ***gcc*** -E *hello.c –o hello.i*

（1）预处理阶段

预处理器（cpp) 根据以字符#开头的命令，修改原始的C程序。

比如#include <stdio.h>告诉预处理器读取标准库负责I/O地头文件stdio.h，把它插入到程序文本中。

程序处理的转换过程

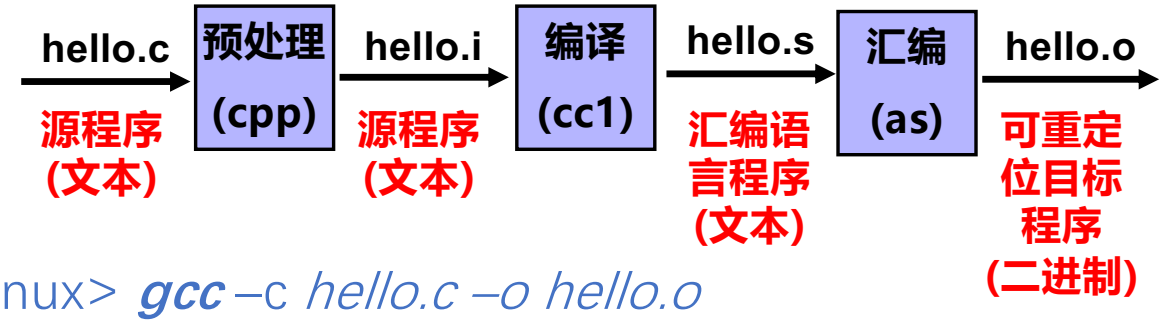


linux> ***gcc*** –S *hello.i –o hello.s*

（2）编译阶段

编译器（cc1）将预处理后的文本文件hello.i翻译成汇编语言源程序hello.s。

程序处理的转换过程

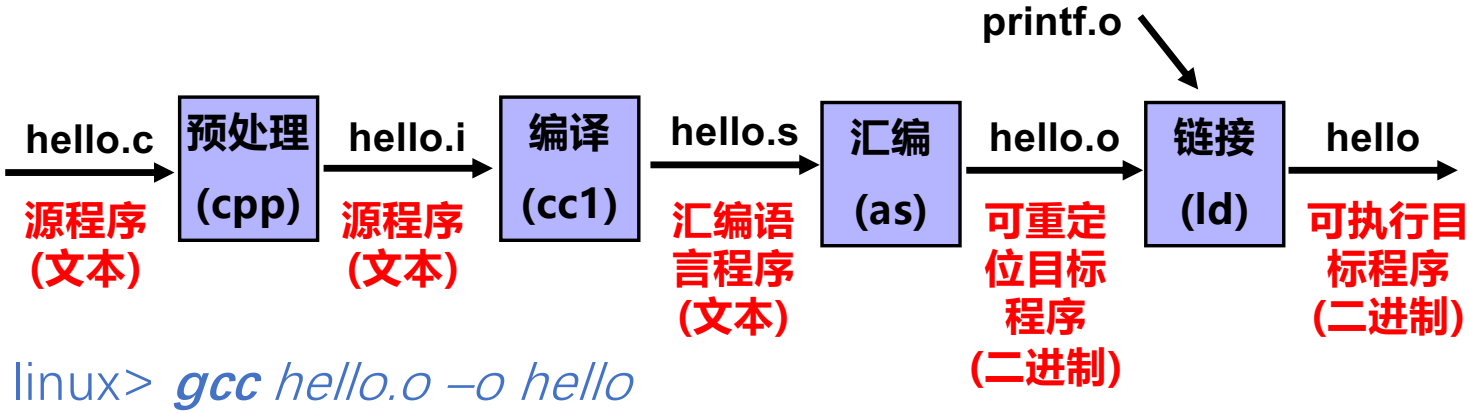


linux> ***gcc*** –c *hello.c –o hello.o*

（3）汇编阶段

汇编器（as) 将汇编程序翻译成机器语言指令，生成可重定位目标程序的二进制格式。

程序处理的转换过程



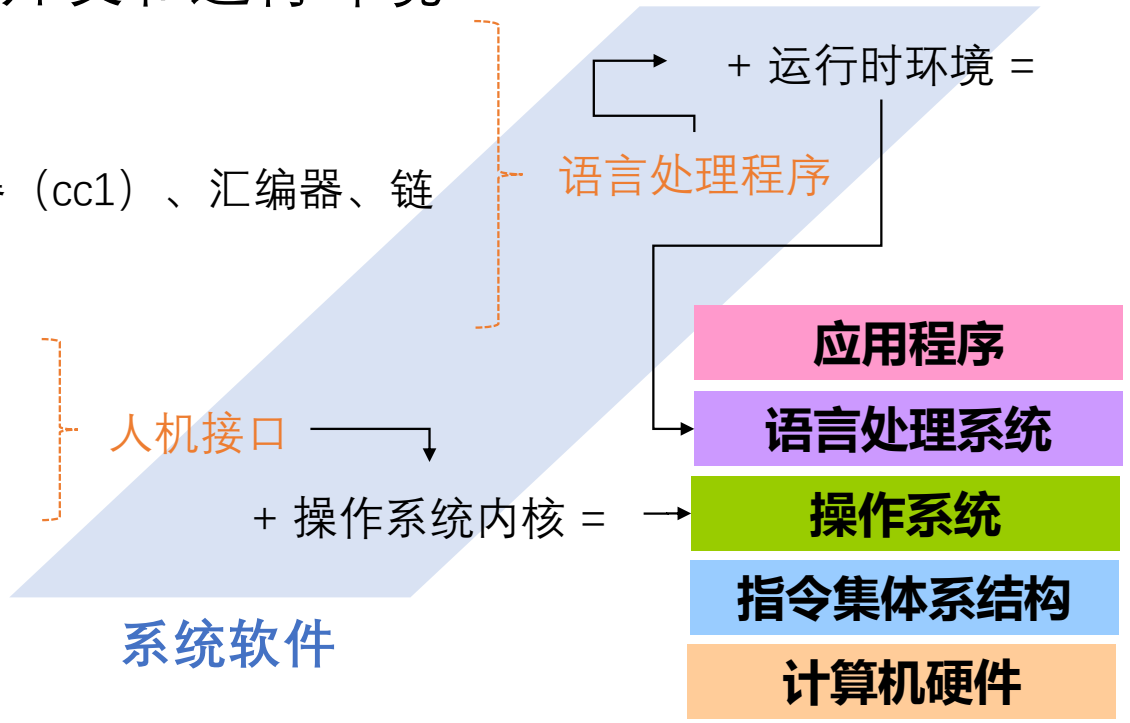
linux> ***gcc*** *hello.o –o hello*

（4）链接阶段

链接器（ld）负责处理合并printf.o和hello.o，得到可执行目标文件，它可以被加载到内存中，由系统执行。

系统软件：编程语言

- 高级语言编程需要复杂的开发和运行环境
  - 编辑器编写源码
  - 一套翻译转换软件
    - 编译器：预处理器、编译器（cc1）、汇编器、链接器、优化器…
    - 解释器
  - 执行环境的界面程序
    - GUI（图形用户界面）
    - CUI（命令行用户界面）

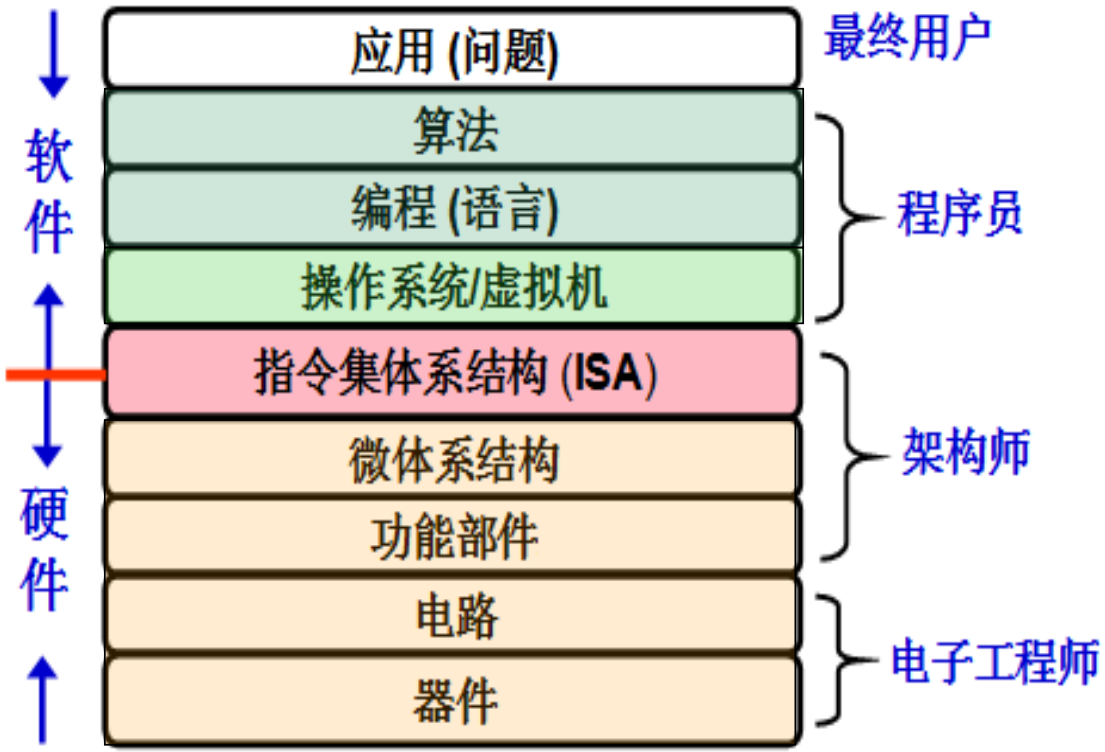


1. 计算机系统概论

1.1. 计算机系统基本组成和原理
1.2. 计算机系统层次结构
1.3. 计算机系统性能评价

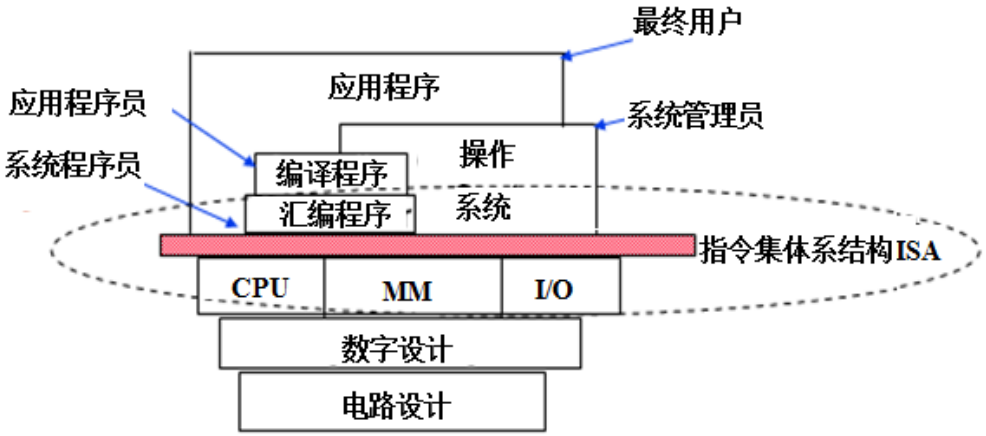
计算机系统抽象层次

- 功能转换
  - 上层是下层的抽象
  - 下层是上层的实现
- 底层为上层提供支撑环境



计算机系统的不同用户

- 最终用户工作在由应用程序提供的最上面的抽象层
- 应用程序员工作在由语言处理系统（如编译器）的抽象层
- 语言处理系统建立在操作系统之上
- 编译器的目标程序由机器级代码组成
- 系统程序员（实现系统软件）工作在ISA层次
- 系统管理员工作在由操作系统提供的抽象层
- ISA处于软件和硬件的交界面（接口）



指令集架构（ISA）

- ISA是一种规约（specification），规定了如何使用硬件
  - 可执行的指令的集合，包括指令格式、操作种类以及每种操作对应的操作数的相应规定；
  - 指令可以接受的操作数的类型；
  - 操作数所能存放的寄存器组的结构，包括每个寄存器的名称、编号、长度和用途；
  - 操作数所能存放的存储空间的大小和编址方式；
  - 操作数在存储空间存放时按照大端还是小端方式存放；
  - 指令获取操作数的方式，即寻址方式；
  - 指令执行过程的控制方式，包括程序计数器、条件码定义等。

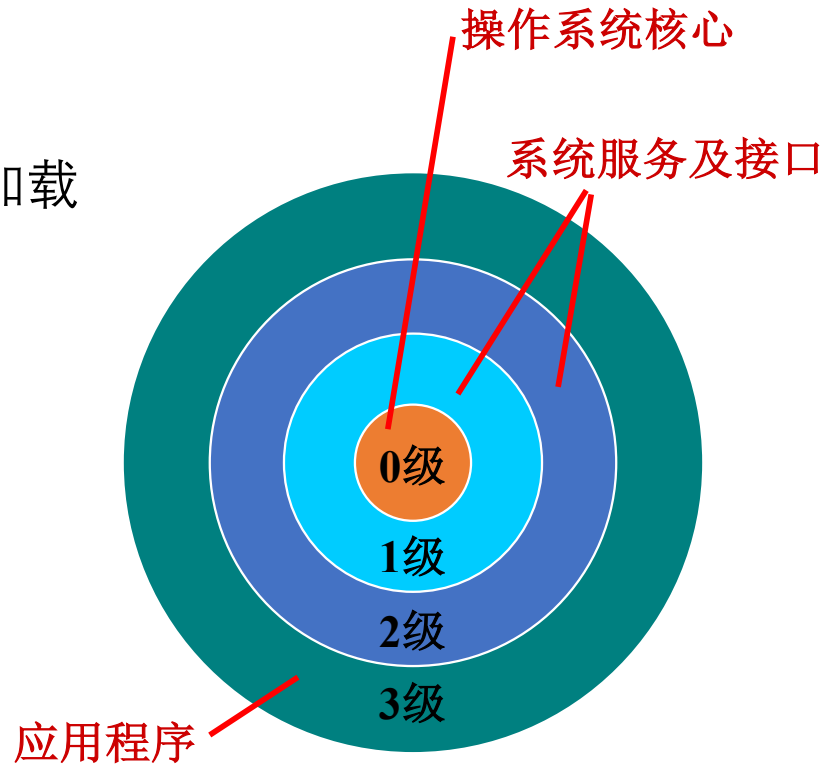
ISA及其组成（微体系结构）之间的关系

- ISA是计算机系统必不可少的抽象层
  - 没有ISA，软件无法使用硬件
  - 没有ISA，计算机无法称为通用计算机

- ISA是计算机组成的抽象
  - 计算机组成必须实现ISA规定的功能
  - 同一ISA可以有不同的计算机组成
    - 乘法指令
      - ALU：加法+移位
      - 乘法器

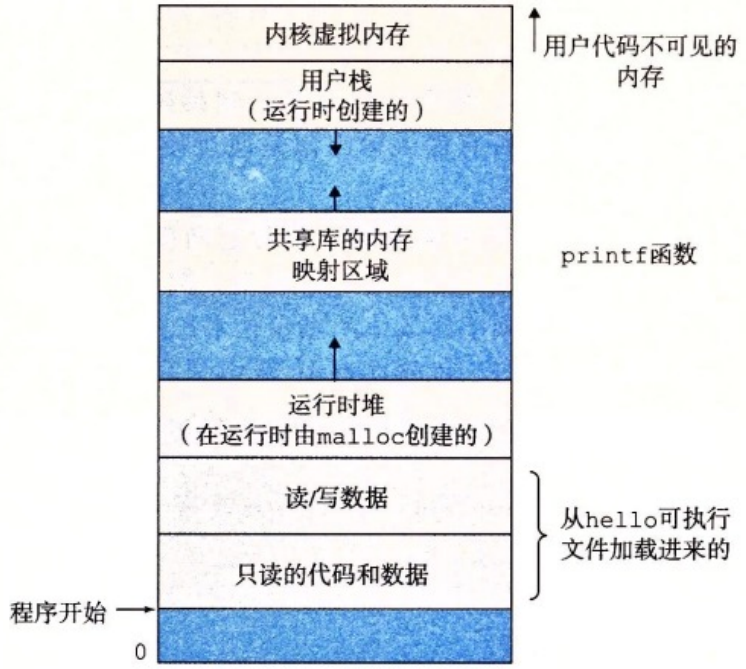
x86-64处理器的工作模式

- 实地址模式（实模式）
  - 程序直接访问物理内存，寻址空间小
  - 用于硬件自检、操作系统引导、内存加载
  - 兼容8086
- 保护虚拟地址模式（保护模式）
  - 引入虚拟内存和分页机制
  - 支持多任务操作系统
  - 采用保护机制



保护虚拟地址模式

- 虚拟存储器地址空间
  - 虚拟存储器：存储管理部件把主存（物理存储器）和辅存（磁盘）看作是一个整体
  - 虚拟地址（逻辑地址）：寻址空间大大超过物理空间
  - 当程序运行要调用的程序和要访问的数据不在物理存储器时，操作系统将其调入物理存储器。



1. 计算机系统概论

1.1. 计算机系统基本组成和原理
1.2. 计算机系统层次结构
1.3. 计算机系统性能评价

计算机性能的评价

- 完成任务所需的时间
  - 响应时间　事务处理系统（存/取款速度快）
  - 执行时间
  - 等待时间（时延）
- 单位时间能完成的任务数
  - 吞吐量　多媒体应用（音/视频播放流畅）
  - 带宽

文件服务器
Web服务器等

计算机性能的基本评价指标

- CPU的执行时间
  - 机器X的速度（性能）是Y的n倍
    - 相对性能用执行时间的倒数表示

$$\frac{\text{Performance(X)}}{\text{Performance(Y)}} = n = \frac{\text{ExeTime(Y)}}{\text{ExeTime(X)}}$$

= 机器X比Y快n-1倍

常用术语

- 字长（数据宽度）
  - 处理器一次可以直接处理的二进制数码的位数
  - 通常取决于微处理器内部通用寄存器的位数和数据总线的宽度
- 时钟周期
  - 一个脉冲信号所需要的时间
  - CPU最小时间单位
- 主频（时钟频率）
  - 单位时间能产生多少脉冲信号
  - 用来表示微处理器运行速度
  - 主频越高，处理器速度越快

CPU执行时间的计算

- CPI（Cycles Per Instruction）：一条指令所需的时钟周期数
- CPU执行时间=CPU时钟周期数×时钟周期
  =CPU时钟周期数÷时钟频率
  =指令条数×CPI÷时钟频率
- CPI是多方面综合的衡量结果
  - ISA
  - 计算机组成
  - 程序（编译、算法）

CPU执行时间的计算

程序P在机器A上运行需10s，机器A的时钟频率为400MHz。现在要设计一台机器B，希望该程序在B上运行只需6s。

已知机器B时钟频率的提高导致其CPI增加，使得程序P在机器B上的时钟周期数是机器A上的1.2倍。机器B的时钟频率达到A的多少倍才能达到6s的运行时间？

解：

主频是2倍，不代表速度是2倍！

$$\text{CPU运行时间A}=10\text{s}=\frac{\text{时钟周期数}_\text{A}}{\text{时钟频率}_\text{A}}=\frac{\text{时钟周期数}_\text{A}}{\text{时钟频率}_\text{A}}$$

$$\text{CPU运行时间B}=6\text{s}=\frac{\text{时钟周期数}_\text{B}}{\text{时钟频率}_\text{B}}=\frac{1.2\times\text{时钟周期数}_\text{A}}{n\times\text{时钟频率}_\text{A}}=\frac{1.2}{n}\times 10$$

$n=2$

指令执行速度

- CPU执行时间=指令条数×CPI÷时钟频率
- MIPS（Million Instructions Per Second）：指令执行速度

  =指令条数IC÷(CPU执行时间×10^6)

  =指令条数IC÷(指令条数IC×CPI÷时钟频率×10^6)

  =时钟频率÷(CPI×10^6)
- MFLOPS（Million Floating-point Operations Per Second）：浮点操作速度

  =浮点操作数÷(时间×10^6)
  - GFLOPS、TFLOPS、PFLOPS、EFLOPS

浮点操作速度的计算

Intel core i7-14700K有8个性能核，8个能效核，性能核最大睿频5.6GHz，能效核最大睿频4.2GHz，每个核心每个时钟周期能执行16次单精度浮点运算，求该CPU的TFLOPS。

解：

$$8\times5.6\times10^{9}\times16+8\times4.2\times10^{9}\times16=1.2544\times10^{12}=1.2544\ \text{TFLOPS}$$

全球超级计算机500强

- 125.44 PFlop/s
  $=10^5 \times 1.2544$ TFLOPS

序号	系统	所属国家	内核	运算性能 (PFlop/s)	峰值性能 (PFlop/s)	功率 (千瓦)
1	Frontier	美国	8730112	1102.00	1685.65	21100
2	Supercomputer Fugaku	日本	7630848	442.01	537.21	29899
3	LUMI	芬兰	2220288	309.10	428.70	6016
4	Leonardo	意大利	1463616	174.70	255.75	5610
5	Summit	美国	2414592	148.60	200.79	10096
6	Sierra	美国	1572480	94.64	125.71	7438
7	神威太湖之光	中国	10649600	93.01	125.44	15371
8	Perlmutter	美国	761856	70.87	93.75	2589
9	Selene	美国	555520	63.46	79.22	2646
10	天河-2A	中国	4981760	61.44	100.68	18482

2022年数据

Amdahl定律

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

- 加快某部件执行速度所能获得的系统性能加速比，受限于该部件的执行时间占系统中总执行时间的百分比。

$$T_{new} = (1-\alpha)T_{old} + \frac{\alpha T_{old}}{k} = T_{old}(1-\alpha+\frac{\alpha}{k})$$

  - 可改进比例α
  - 部件加速比k

$$S = \frac{T_{old}}{T_{new}} = \frac{1}{1-\alpha+\frac{\alpha}{k}}$$

Amdahl定律

例：将计算机系统中某一功能的处理速度加快15倍，但该功能的处理时间仅占整个系统运行时间的40%，则采用此增强功能方法后，能使整个系统的性能提高多少？

解：

$$\alpha = 0.4, \quad k = 15$$

$$S = \frac{1}{1-\alpha+\frac{\alpha}{k}} = \frac{1}{1-0.4+\frac{0.4}{15}} \approx 1.6$$

提高到原来的1.6倍

Amdahl定律

$$S = \frac{1}{1 - \alpha + \frac{\alpha}{k}}$$

- 一种性能改进的递减规则
  - 如果仅仅对计算任务中的一部分做性能改进，则改进得越多，所得到的总体性能的提升就越有限。

- 重要推论
  - 如果只针对整个任务的一部分进行改进和优化，那么所获得的加速比不超过

$$S = \frac{1}{1 - \alpha}$$

1. 计算机系统概论

1.1. 计算机系统**基本组成和原理**
1.2. 计算机系统**层次结构**
1.3. 计算机系统**性能评价**