计算机组成原理

我并不是计算机专业的学生，然而，现代的自动化技术，从机器人到嵌入式系统都是以计算机为控制器，控制算法也依靠C++、Python、Matlab等高级语言进行编写。因此自动化技术和计算机的关系是极为紧密的，自动化的学生也有必要熟悉计算机的基本工作原理。

这次学习计算机组成原理将使用多媒体学习法，学习媒介包括书本、视频、网上论坛和游戏，但并不追求学习的深度，只求对计算机的底层架构能有一个最基本的理解。

·主要学习资料：

1. 计算机组成原理，Alan Clements，机械工业出版社
2. 计算机组成原理MOOC课，刘宏伟，哈尔滨工业大学
3. Multisim，National Instrument
4. Turing Complete，LevelHead
5. Minecraft，Mojang Studios，Microsoft

目录

[# 课程背景 1](#_Toc158643785)

[1.1. 不同的视角 1](#_Toc158643786)

[1.2. 相关课程 2](#_Toc158643787)

[1.3. 组成 vs 体系结构 2](#_Toc158643788)

[1. 计算机算术 2](#_Toc158643789)

[2.1. 二进制、十进制、十六进制的转换 2](#_Toc158643791)

[2.2. 整型数的表示——补码 3](#_Toc158643792)

[2.3. 整型数的四则运算 4](#_Toc158643793)

[2.4. 浮点数标准 4](#_Toc158643794)

[# 附录 5](#_Toc158643795)

[3.1. 想法与收获 5](#_Toc158643797)

[3.2. 问题与疑难 5](#_Toc158643798)

[3.3. 专业术语解释 6](#_Toc158643799)

# # 课程背景

## 不同的视角

计算机是很复杂的机器，不同的使用者基于不同的用途所看到的计算机内部结构是不相同的。从“有用无用”的角度看待，电子工程师、硬件工程师、嵌入式工程师、汇编语言程员、高级语言程序员所必须掌握的计算机原理知识也是不一样的。

## 相关课程

计算机组成原理课在国内外的大学有不同的名称和教学范围。

·国内：称为“计算机组成原理”或“计算机组成与设计”，是计算机专业核心课程，与《数据结构》、《计算机网络》、《计算机操作系统》并称为“四大天书”

·国外：称为“Computer Organization and Architecture”（计算机组成与设计）或“Computer Organization and Design”（计算机组成与体系结构）

## 组成 vs 体系结构

词汇“组成Organization”和“体系结构Architecture”经常被混用，但具体含义不同。

·“组成”：计算机具体的硬件电路实现和连接方式，是体系结构的具体实现；

·“体系结构”：抽象层面的架构设计，汇编层次的操作视角；

# 计算机算术

·关键词：进制转换、数值类型、基本运算

现代的计算机都是数字计算机，所有的数据处理和运算基于二进制0和1。为了与现实世界进行交互，我们也要关注十进制；另外，十六进制为二进制数提供了一种精简的写法。

二进制整数四则运算的实现是计算机“计算”的基础；补码为负数的处理提供了一种便于操作的表示法；而当表示对象扩展到全体实数，就要引入浮点数，这时就需要考虑进制转换和无理数带来的计算误差了。



## 二进制、十进制、十六进制的转换

**·二进制Binary<—> 十六进制Hexadecimal：**

每四位二进制数就是一位十六进制数。

|  |
| --- |
|  |

**·二进制Binary、十六进制Hexadecimal —> 十进制Decimal：**

任意n进制转换成十进制很简单，就是每一位依次乘上对应的进制基数n的幂，整数和小数部分的操作是一样的。转换是精确的。

|  |
| --- |
|  |

**·十进制Decimal —> 二进制Binary：**

十进制转换为其他进制数稍微复杂，整数部分用短除法（除以基数n取余），小数部分用乘法（乘基数n取整）。

注意，某些在十进制下可以用有限位表示的小数在二进制下无法用有限位表示，这时进制转换就引起进制误差。

下列草稿显示十进制349.71转化成二进制数的过程。

|  |
| --- |
|  |

## 整型数的表示——补码

负数并不是自然数，它是人类创造出来的表示“减损”概念的数字，因此要专门设计一种方法在计算机里表示负整数。它就是有符号整型数的补码表示法。

·源码：

把最高位MSB作为符号位，用于表示数的正负是很自然的想法，但是这种格式的有符号整型数会有两个二进制数（10000000和00000000）表示0。

·补码：

负数的二进制补码是它的源码除符号位以外取反，再加一得到的结果。这种处理方法确保了0的表示的唯一性。（此时只有00000000表示0，10000000表示-128）

|  |
| --- |
|  |

补码表示法同时赋予最高位MSB“符号位”和“数值位”的意义，一个八位的无符号数，其最高位表示，成为有符号数以后，最高位就表示

|  |
| --- |
|  |

八位二进制补码的表示范围是，其中负数比正数多一位是因为源码的“负0”分给了负数。

补码还有一个没啥用但是很有趣的性质，我把它称为“溢出周期性”。补码表示的有符号整型向上溢出时会变为可表示的最小数，向下溢出时会变为可表示的最大数，看起来就像一条衔尾蛇。以八位有符号整型为例：

|  |
| --- |
|  |

## 整型数的四则运算

无论是二进制数还是十进制数，四则运算的基本原理是一样的，不过计算机和人的计算步骤不太一样。

**·加减法**

减去一个数等价于加上这个数的相反数；用补码表示负整数的一个好处是，加减法可以用相同的硬件完成。

|  |
| --- |
|  |

加减法的运算结果可能会溢出一位。

**·移位**

移位不是基本四则运算，但从算术原理上，它是实现乘法的前提；二进制数左移一位数值上变为原来的两倍，右移一位则变为原来的一半。

|  |
| --- |
|  |

**·乘法**

计算机执行乘法操作的性能花销远大于加减法，一次乘法操作就相当于多次的加法和移位操作。

## 浮点数标准

为了让计算机能表示整数以外的实数，以及很小、很大的数，引入了浮点数Floating-point Value的概念；浮点数是一种基于科学计数法的二进制标准化实数表示法。

·二进制浮点数的规格化（科学计数法）：

|  |
| --- |
|  |

**IEEE 754**标准定义了常用的32位单精度浮点数float，64位双精度浮点数double，以及其他较为少用的浮点数格式，如128位四精度、16位半精度。

标准格式下的浮点数由三部分组成，符号位Sign、偏置指数Exponent、尾数（小数部分）Fraction。

|  |
| --- |
|  |

·浮点数格式的特殊处理：

1. 偏置指数：偏置指数所表示的并不是实际的指数，而是实际指数加上一个偏置值的数，这个设计主要是为了能用全0浮点数表示数字0.0。
2. 不保存整数开始位：在二进制的科学计数法中，除了表示数字0.0以外，尾数的整数部分都是1，因此为了节省储存空间，浮点数不会保存整数位的1；
3. 负数用单独符号位而非补码表示：浮点数运算不是精确的，对于截断或舍入处理后为0的运算结果，符号位可以用于判断数的正负，因此浮点数反而没有必要使用补码规避“正0”和“负0”的重复；

# # 附录



## 想法与收获

·只读存储器：早期的任天堂红白机卡带是一种只读存储器ROM，现在的Switch卡带则使用闪存技术。

·指针和寻址：在寻址取数的过程中，地址比数值更重要，计算机并不关心具体的数值。就像是一个刚开始学做饭的孩子，他可以只记住某道菜需要的调味料是第几个架子上的第几瓶，不需要知道具体是什么调味料。P28

·计算机的复杂性：计算机的原理是非常简单的，现代计算机复杂的一个原因是包含大量为提高性能、扩展功能而引入的设计细节。P38

·保存地址的重要性：无论是高级语言的指针还是CPU的指令寄存器，都体现了储存地址的重要性——数据看似被规整地保存在存储器中，实则是投入了数据的茫茫大海中；如果丢失它的存储地址，就难以找回了。

## 问题与疑难

·主存和内存有什么不同？

·寄存器的作用是什么？为什么指令的类型要分为从寄存器读取和从储存器读取？

·浮点数怎么表示0.0？不会和1.0混淆吗？

注意，由于浮点数的指数是偏置指数，1.0的指数在单精度格式下实际上表示为127而不是0，尾数则为全零；浮点数表示的0.0实际上是四种特殊情况之一，此时偏置指数和尾数都取0。

所以浮点数格式下的0.0和1.0区别在于偏置指数，前者是127，后者是0，不会混淆的。

·冯·诺依曼结构的计算机怎么区分存储器中存放的指令和数据？

实际上并不需要刻意从存储格式上进行区分，CPU中的指令寄存器IR会记录即将执行的指令在存储器中的地址。

·程序计数器PC是怎么知道将要执行的指令的存放地址的？

没错，实际上PC并不知道指令存放在哪，它只是一个存放执行中指令地址的寄存器。

如果多条连续执行的指令在存储器中的地址是连续的，PC通过自带的加法计数器可以自动定位下一条指令，但如果指令是任意存放的，就需要依靠汇编语言常见的跳转Jump或分支Branch指令来告诉PC下一条要执行的指令在哪。

·为什么计算机的位数都是8位、16位、32位、64位？没有27位、9位、21位？

## 专业术语解释

对专业术语的不熟悉是学习过程的一大障碍，一般分为两种情况：一、完全不知道专业术语的含义，造成阅读障碍；二、专业术语的指代过于具体，相似术语之间容易混淆。

|  |  |  |
| --- | --- | --- |
| 中文 | 英文 | 解释 |
| 指令集体系架构 | ISA, Instruction Set Architecture | 程序员看到的计算机视图， |
| 微体系结构 | Microarchitecture |  |
| 微处理器 | Microprocessor |  |
| 系统体系结构 | System Architecture | 一台完整计算机的结构，包括CPU、存储器、总线、接口和所有输入输出外设，关注它们的排布 |
| 主存 | Main Memory |  |
| 冯·诺依曼瓶颈 | The von Neumann bottleneck | CPU与存储器之间的传输制约计算机的性能，这是因为冯·诺依曼架构的计算机把指令和数据同时存放在存储器中 |
| 储存墙 | Memory Wall | 储存性能的落后制约计算机技术的发展，因为储存技术的发展速度远低于处理器的发展速度 |
| 最高/低位 | Most/Least Significant Bit | 一个二进制数的最高/最低位（比特），英文直译“最重要/最次要比特” |