**数字逻辑与数字系统**

**第一次作业**

## 软件工程专业为什么要学习数字逻辑与数字系统？

根据《西安交通大学软件工程专业培养方案》，数字逻辑与数字系统是软件工程专业的专业大类必修课，是计算机组成与结构、计算机网络等必修课的先修课程。数字逻辑与数字系统主要满足软件工程专业毕业要求中的以下几项：

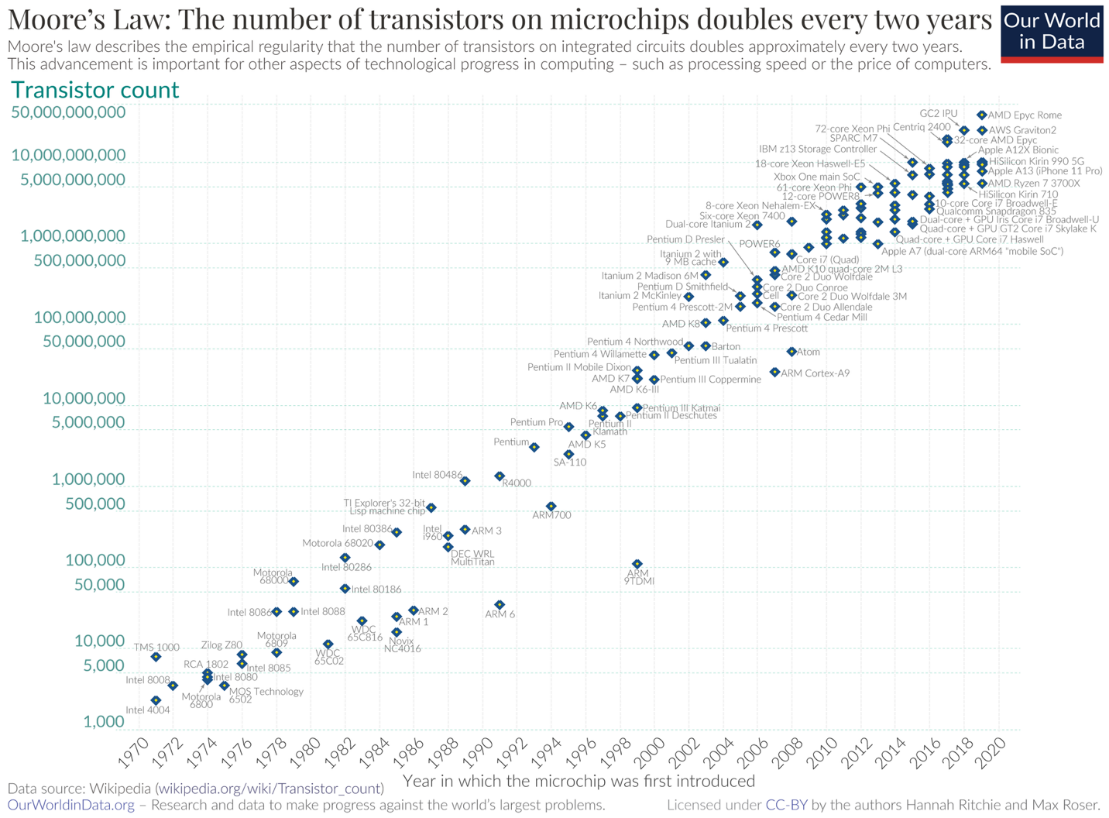
1. 自主学习。数字逻辑与数字系统这门课可以使我们认识和了解数字电路，从而理解计算机底层原理。由此，我们能够在未来软件工程及相关领域技术和观念飞速发展、变化时，更好地学习和适应。
2. 问题分析。了解数字逻辑与数字系统相关的知识，有助于我们在日后的软件项目开发中，分析关键环节和影响因素，并探寻其中的影响机制或规律。同时，学习这门课也可以培养我们的逻辑推理能力，提高解决问题的能力。
3. 开发解决方案。针对软件系统或计算机系统设计、实现或测试中的工程问题，我们可以利用相关知识，改进和设计创新性的解决方案。

参考文献：西安交通大学软件工程专业培养方案.http://se.xjtu.edu.cn/info/1010/2528.htm

## 用实例说明摩尔定律。

目前流传较广的摩尔定律为：集成电路上可容纳的晶体管数目，约每隔18至24个月便增加一倍。

1965年，英特尔公司创始人之一戈登·摩尔在《电子学》杂志上的一篇文章中预言，半导体芯片上集成的晶体管和电阻数量将每年增加一倍。十年后，摩尔对自己的预测进行了修正，认为1980年之前半导体复杂度每年还会持续以两倍速度提升，但此后会逐步缩减到每两年两倍。摩尔否认自己曾说过“18个月”，“18个月”的来源似乎是英特尔首席执行官大卫·豪斯，他曾预言，每18个月，芯片的性能将提高一倍。

摩尔定律作为一个经验猜想，在数十年的半导体芯片发展历程中均得到验证。一张来自Our World in Data的图表非常直观地验证了摩尔定律（如下图），可见晶体管数量的对数与年份基本成线性关系。

摩尔定律反映了信息技术进步的速度和趋势，对于半导体行业和计算机领域有着重要的指导意义。

作为一个推测，摩尔定律无法被严格证明。2010年以来，有人认为摩尔定律已经开始放缓，这是因为晶体管尺寸已经接近极限而无法继续缩小。更小的晶体管尺寸可能面临量子隧穿效应的影响，且功耗、散热、可靠性等问题也越来越严重。许多专家学者预言，摩尔定律将在2025年前后彻底失效。当然，随着科技创新不断突破物理限制和技术难题，摩尔定律也仍可能持续有效。例如，陈杨等在谷电子学与微纳光子学交叉领域的一项最新研究成果，为突破摩尔定律的瓶颈提供了新思路。

参考文献：

[1]摩尔定律-维基百科. <https://zh.wikipedia.org/wiki/摩尔定律/>

[2]Moore's law: The number of transistors per microprocessor - Our World in Data. <https://ourworldindata.org/grapher/transistors-per-microprocessor>

[3]你理解的“摩尔定律”可能是错的：回顾走过半个世纪的摩尔定律-黄烨锋. <https://www.eet-china.com/news/202210130923.html>

[4]科学家另辟蹊径，为摩尔定律“续命”-孙明源.科技日报,2022-11-22(006)

## 说明计算机中二进制浮点数的表示格式和进行四则运算的方法。

如今，计算机中二进制浮点数的表示格式一般遵循IEEE 754标准。IEEE 754标准将浮点数在内存中的存储划分为sign、exponent和fraction三个区域，分别为符号位、阶码和尾数。下面具体说明这三个内容：

将一个二进制浮点数转换为类似科学计数法的形式，即形如-1.01101\*211。用符号位表示浮点数的正负，此例中为1；阶码是次方数加上该长度浮点数的偏移值（这是为了表示负的次方数），在此例中若偏移值为127，则阶码为10000011即130；尾数为小数点后面的数值即01101。

根据浮点数所占用内存空间的不同，可以分为不同的表示格式，常见的几种有双精度、单精度和半精度，它们的区别如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| **浮点数类型** | **双精度** | **单精度** | **半精度** |
| **占用位数** | 64 | 32 | 16 |
| **符号位** | 1 | 1 | 1 |
| **指数位（阶码）** | 11 | 8 | 5 |
| **尾数位** | 52 | 23 | 10 |
| **指数偏移值** | 1023 | 127 | 15 |
| **用途** | 精度最高，适用于需要较大数、较小数或所需的精度极高的研究场合 | 精度适中，可以满足绝大多数的运算 | 精度较低，一般用于一些内存空间有限、需要节省内存空间的场合，如嵌入式系统等 |

IEEE 754浮点数的加减法可以通过以下步骤来完成：

1. 对阶。由于两个浮点数的指数位很有可能不同，在完成加减法计算前应计算二者的阶差，并依据“小阶对大阶”的原则将二者的阶统一。这个过程中应对浮点数的尾码进行变化。
2. 计算尾数。此时的尾数是同阶的，计算加减的结果即可。
3. 结果规格化、舍入。如果计算得到的结果不符合存储格式，如101.1101\*2111，需要将其变换为一般形式即1.011101\*21001。在这个过程中如果末尾存在舍去，依据“0舍1入”进行舍入。

IEEE 754浮点数的乘除法可以通过以下步骤来完成：

1. 0操作数检查。若参与运算的浮点数中有0，则乘法结果为0，除法结果为0或不存在。
2. 阶码加减。与科学计数法的乘除类似，将2的幂相乘除，即将阶数相加减。
3. 尾码相乘除。利用加减、移位的方式进行。例如，1.011\*1.1=1.011+.1011=10.0001。
4. 结果规格化、舍入。与加减法相同。这里，计算结果超出的位数可能较多，可以选择直接截断或者进行特定舍入。

参考文献：

[1]IEEE 754-维基百科.https://zh.wikipedia.org/wiki/IEEE\_754/

[2]浮点数运算. https://blog.csdn.net/weixin\_45699237/article/details/120750768

## 课后习题

##### 1.1 完成下列数制转换：

1. （1101011）2 =（6B）16
2. （101111.0111）2 =（57.34）8
3. （67.24）8 =（110111.0101）2
4. （15C.38）16 =（101011100.00111）2
5. （5436.15）O =（101100011110.001101）B =（B1E.38）H
6. （552273）O =（101101010010111011）B =（2D4BB）H

##### 完成下列二进制加减法：

1. （110101）2 +（11001）2 =（1001110）2
2. （101110）2 +（100101）2 =（1010011）2
3. （11011101）2 -（1100011）2 =（1111010）2
4. （1110010）2 -（1101101）2 =（101）2

##### 1.8 分别写出下列十进制数的8位二进制原码、反码和补码。

|  |  |  |  |
| --- | --- | --- | --- |
| 十进制数 | 原码 | 反码 | 补码 |
| +18 | 00010010 | 00010010 | 00010010 |
| +115 | 01110011 | 01110011 | 01110011 |
| +79 | 01001111 | 01001111 | 01001111 |
| -49 | 10110001 | 11001110 | 11001111 |
| -3 | 10000011 | 11111100 | 11111101 |
| -100 | 11100100 | 10011011 | 10011100 |

##### 1.16 已知下列机器数，写出它们所对应的真值。

|  |  |
| --- | --- |
| 机器数 | 真值 |
| [x1]原=11011 | -11 |
| [x2]反=11011 | -4 |
| [x3]补=11011 | -5 |
| [x4]原=00000 | 0 |
| [x5]反=01111 | 15 |
| [x6]补=01000 | 8 |

##### 1.23 完成下列数制转换。

1. （1010111）BCD =（57）10
2. （100000111001.01110101）BCD =（839.75）10
3. （1011001111001001）余3 =（1000000010010110）BCD
4. （752.18）10 =（11101010010.00011）BCD