**自由教辅 信息技术重构 依据CC-BY-NC-SA许可分发 Version 0.0.1**

**第一章 计算机基础**

**一、 计算机组成**

**1. 硬件组成**

计算机的**五个基本组成部分**是：

1. **运算器**：负责执行**算术运算（如加减乘除）和逻辑运算（如与、或、非）**。它通常包含寄存器和算术逻辑单元（ALU），用于临时存储和处理数据。

--拓展1：逻辑门

1. **控制器**：负责**从存储器中取出指令，解释指令**，并**协调计算机各部分的工作**。控制器通常包括程序计数器（PC）、指令寄存器（IR）和控制单元（CU）。
2. **存储器**：用于存储数据和程序。主要分为**主存储器**（**RAM内存**、**ROM只读存储器**）和**辅助存储器**（**硬盘**、**固态硬盘**等）。**主存储器-内存存取速度快**，但**断电后数据会丢失**，而**辅助存储器用于长期存储数据**。

--拓展2：RAM、ROM、硬盘的区分

1. **输入设备**：用于将外部信息输入计算机，如**键盘、鼠标、扫描仪**等。输入设备负责将用户指令和数据转换为计算机可以识别的格式。
2. **输出设备**：用于将计算机处理后的结果输出给用户，如**显示器、打印机、扬声器**等。它们将计算机内部的二进制数据转换为人类可理解的信息。

**2. 软件组成**

软件是指使计算机硬件执行特定任务的**程序和相关文档**，主要分为两类：​

1. **系统软件**：​管理和控制计算机硬件，支持应用软件运行的程序，包括**操作系统**（如**Windows、Linux、macOS、BSD、Unix、RTOS**）和**系统工具**。​
2. **应用软件**：​为**满足用户特定需求**而开发的程序，如**文字处理软件、浏览器、媒体播放器**等。​

硬件和软件相互依赖，共同构成完整的计算机系统。​硬件提供运行平台，软件指挥硬件完成各种任务，实现计算机的功能。​

**二、进制及其转换**

**1. 进制规则**

1. 先用阿拉伯数字（0~9）表示，不够用则引入字母（A~Z）；
2. n进制则有（0,1,2,…,n-1）的数字（0也算一个数字）；
3. 相同的值，n进制表示，n越大，位数越短；
4. 不同的值，不同的进制表示，看上去可能一样；
5. 对于n进制，描述时的n是使用十进制描述的。

**2. 重要的进制法举例**

|  |  |  |  |
| --- | --- | --- | --- |
| **进制** | **定义** | **英文名** | **英文名渊源** |
| **二进制 (Binary)** | **以 2 为基数，仅使用 0 和 1** | **Binary** | **源自拉丁语 *bini*，意为“两者一组”** |
| **八进制 (Octal)** | **以 8 为基数，使用 0-7** | **Octal** | **源自拉丁语 *octo*，意为“八”** |
| **十进制 (Decimal)** | **以 10 为基数，使用 0-9** | **Decimal** | **源自拉丁语 *decem*，意为“十”** |
| **十六进制 (Hexadecimal)** | **以 16 为基数，使用 0-9 和 A-F** | **Hexadecimal** | ***Hex-* 源自希腊语 *hex*（六），*-decimal* 源自拉丁语 *decem*（十）** |

**3. 进制转换**

1. **十进制转其他进制**：首先将待转换的十进制数除以目标进制的数，记录余数，然后用所得的商继续除以目标进制，直到商为0。把**所有余数倒序排列**，得到的数就是对应进制的表示。例如，将十进制数156转换为8进制：156除以8商19余4，19除以8商2余3，2除以8商0余2，**倒序排列余数**就是2 3 4，所以156的8进制表示为234。
2. **其他进制转十进制**：从右往左给每一位乘上**目标进制的对应幂次**（最右边的数位为该进制的0次幂，依次向左依次增大），然后把所有乘积相加得到的和就是十进制数。例如，将八进制数234转换为十进制：右边数字4乘80得4，中间数字3乘81得24，最左边数字2乘82得128，将这些值相加128+24+4得156，所以八进制数234转换为十进制数为156。
3. **使用系统自带的计算器**：Windows徽标键 + R 键，输入“calc”，回车，点击“查看”，选择“程序员”模式，选择待转换的进制，输入待转换的值，再选择目标进制，结果即为所求。

**4. 进制转换技巧**

1. 待转换值的**最右边的数位的转换结果**与最终结果的**最右边的数位**一致；
2. 二进制与八进制或十六进制之间转换时，可先将二进制分成3位（23，对于二进制/八进制）或4位一组（24，对于二进制/十六进制），再分别对应转换后合并。

**三、汇编语言基础**

**1. 汇编语言基本概念**

1. 寄存器（Registers）：CPU 内部的小型存储单元，如 AX、BX（8086）；
2. 指令（Instructions）：CPU 可执行的基本操作，如 MOV（数据传输）、ADD（加法）；
3. 内存寻址：汇编通过地址操作数据，如 MOV AX, [1234H] 读取内存地址 0x1234 的数据；
4. 中断：计算机系统中用于处理异步事件的一种机制。当 CPU 在执行程序时，某些事件（如 I/O 设备请求、异常情况或外部信号）可能需要 CPU 立即响应，这时 CPU 会暂停当前任务，转去执行相应的中断处理程序，然后再恢复原任务。

**2. 基本寄存器**

|  |  |  |  |
| --- | --- | --- | --- |
| 寄存器 | 名称 | 作用 | 示例用途 |
| **AX** | 累加寄存器（Accumulator） | 主要用于算术运算、I/O 操作 | MOV AX, BX（数据传输） |
| **BX** | 基址寄存器（Base） | 存放内存地址或数据 | MOV [BX], AL（访问内存） |
| **CX** | 计数寄存器（Counter） | 主要用于循环计数 | LOOP start（循环控制） |
| **DX** | 数据寄存器（Data） | I/O 端口访问、扩展乘除法 | OUT DX, AL（I/O 操作） |
| **SP** | 堆栈指针（Stack Pointer） | 指向栈顶（相对于 SS 段） | PUSH AX（压栈） |
| **BP** | 基址指针（Base Pointer） | 访问栈中的局部变量 | MOV AX, [BP-2] |

**3. 汇编语言基本指令**

|  |  |  |  |
| --- | --- | --- | --- |
| **指令** | **语法** | **作用** | **示例** |
| **MOV** | MOV 目标, 源 | 数据传输 | MOV AX, BX （将 BX 的值复制到 AX） |
| **ADD** | ADD 目标, 源 | 加法 | ADD AX, 5 （AX += 5） |
| **SUB** | SUB 目标, 源 | 减法 | SUB BX, AX （BX -= AX） |
| **MUL** | MUL 源 | 无符号乘法 | MUL CX （AX = AX \* CX） |
| **IMUL** | IMUL 源 | 有符号乘法 | IMUL DX （AX = AX \* DX） |
| **DIV** | DIV 源 | 无符号除法 | DIV CX （AX / CX，余数存 DX） |
| **IDIV** | IDIV 源 | 有符号除法 | IDIV DX （AX / DX，余数存 DX） |
| **INC** | INC 目标 | 加 1 | INC AX （AX += 1） |
| **DEC** | DEC 目标 | 减 1 | DEC BX （BX -= 1） |
| **CMP** | CMP 目标, 源 | 比较 | CMP AX, BX （AX - BX，不改变寄存器值） |
| **JMP** | JMP 目标地址 | 无条件跳转 | JMP start |
| **JE** | JE 目标地址 | 等于时跳转 | JE loop （ZF=1 时跳转） |
| **JNE** | JNE 目标地址 | 不等时跳转 | JNE retry （ZF=0 时跳转） |
| **JG** | JG 目标地址 | 大于时跳转（有符号） | JG end |
| **JL** | JL 目标地址 | 小于时跳转（有符号） | JL retry |
| **AND** | AND 目标, 源 | 按位与 | AND AL, 0x0F （AL &= 0x0F） |
| **OR** | OR 目标, 源 | 按位或 | OR BX, AX （BX |
| **XOR** | XOR 目标, 源 | 按位异或 | XOR CX, CX （清零 CX） |
| **NOT** | NOT 目标 | 按位取反 | NOT DX |
| **SHL** | SHL 目标, 位数 | 左移 | SHL AX, 1 （AX <<= 1） |
| **SHR** | SHR 目标, 位数 | 右移 | SHR BX, 2 （BX >>= 2） |
| **PUSH** | PUSH 目标 | 压栈 | PUSH AX |
| **POP** | POP 目标 | 出栈 | POP BX |
| **CALL** | CALL 地址 | 调用子程序 | CALL my\_func |
| **RET** | RET | 返回 | RET |
| **INT** | INT 中断号 | 调用中断 | INT 0x21（调用 DOS 21h 号中断） |
| **SYSCALL** | SYSCALL | 调用 Linux 系统调用 | SYSCALL |

**四、拓展**

**拓展1. 逻辑门**

逻辑门是计算机电路里最基本的“开关”，它们根据输入信号决定输出结果，就像做数学题一样。下面是 12 种常见的逻辑门：

|  |  |  |  |
| --- | --- | --- | --- |
| **逻辑门** | **符号** | **功能** | **学生能理解的例子** |
| **与门（AND）** | ∧ | 只有所有输入都是 1，输出才是 1 | 只有 **你完成作业 并且 老师批准**，才能出去玩 |
| **或门（OR）** | ∨ | 只要有一个输入是 1，输出就是 1 | 你可以去春游，只要 **交了钱 或者 被选中** |
| **非门（NOT）** | ¬ | 取反，1 变 0，0 变 1 | 老师说“没交作业不能下课”，NOT 让“没交”变成“交了” |
| **与非门（NAND）** | ↑ | 与门的反向，只有全 1 时输出 0，否则 1 | **你和朋友都作弊，才会被扣分**（但 NAND 反转成“不给分”） |
| **或非门（NOR）** | ↓ | 或门的反向，只有全 0 才输出 1 | 老师规定“**只要一个人说话，全班不能下课**” |
| **异或门（XOR）** | ⊕ | 输入不同时输出 1，相同时输出 0 | **你和朋友不能点一样的菜，否则就没得吃** |
| **同或门（XNOR）** | ⊙ | 输入相同时输出 1，不同时输出 0 | **你和朋友答案一样，才能得分** |
| **缓冲器（BUFFER）** | = | 输出和输入相同，但增强信号 | **老师用麦克风讲课，声音变大但内容不变** |
| **三态门（Tri-state）** | Z | 允许或阻断信号（高阻态） | **老师决定谁可以发言，谁要保持安静** |
| **施密特触发器（Schmitt Trigger）** | - | 过滤噪音，保持信号稳定 | **班上噪声很小，老师不会管，但超过一定程度就会喊“安静！”** |
| **传输门（Transmission Gate）** | 开关 | 选择信号是否通过 | **食堂只有你饭卡有钱，才会允许你刷卡吃饭** |
| **门控反相器（Controlled Inverter）** | 受控 NOT | 只有控制信号允许，才会取反 | **作业只有老师批准后，才能被改成对的答案** |

**拓展2. RAM、ROM、硬盘的区分**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | **RAM（随机存取存储器）** | **ROM**（**只读存储器）** | **硬盘（HDD/SSD）** |
| 定义 | 计算机的**运行内存**，用于**临时存储数据** | 预先写入数据的存储器，主要用于**存储固件** | 计算机的**长期存储设备**，用于**存储操作系统、应用程序和用户数据** |
| 读写方式 | **可读可写，速度快** | **只能读取**，某些类型支持写入 | **可读可写，速度比 RAM 慢但比 ROM 快（SSD > HDD）** |
| 数据持久性 | **断电后数据会丢失** | **断电后数据仍然保留** | **断电后数据仍然保留** |
| 用途 | **运行程序、缓存数据，提高计算机响应速度** | 存储 BIOS/UEFI **固件**、嵌入式系统程序 | 存储**操作系统、软件、文件**等**长期数据** |
| 典型容量 | 4GB - 128GB | 几 MB - 几 GB  EEPROM可能几KB | 256GB - 数 TB |
| 示例 | **DDR4、DDR5 内存** | **BIOS 芯片、微控制器固件** | **HDD（机械硬盘）、SSD（固态硬盘）** |