一、计算机系统知识 *00:09*

1. 本章学习建议 00:45

- **章节对应**：本章内容对应本科《计算机组成原理》课程
- **考试重点**：预计考察6道单选题，总分值约6分
- **学习难点**：知识点较多，平均正确率约50%，需系统复习
- **学习方法**：建议跟随老师讲解顺序逐步理解掌握

2. 计算机系统基础知识

1）硬件基本组成

- **冯诺依曼结构**：计算机硬件由五大部件组成（运算器、控制器、存储器、输入设备、输出设备）
- **CPU组成**：
  - 核心组件：运算器、控制器、寄存器组
  - 功能要求：需掌握各组件的具体功能和协作关系

2）数据表示与校验

- **数值表示**：
  - 四大编码：定点数、浮点数表示方法
  - 码制转换：原码、反码、补码和移码的相互转换
- **校验机制**：
  - 常见校验码：奇偶校验码、海明码等
  - 应用场景：数据传输中的错误检测与纠正

3. 计算机体系结构

1）存储系统

- **存储层次**：
  - 多级存储：寄存器→Cache→主存→磁盘
  - 工作原理：各级存储器的访问特性和协同机制
- **I/O技术**：
  - 四种方式：程序查询、中断、DMA、通道方式
  - 对比要点：原理差异、适用场景及性能特点

2）总线结构

- **分类标准**：
  - 按功能分：数据总线、地址总线、控制总线
  - 按位置分：片内总线、系统总线、通信总线
- **典型总线**：需掌握常见总线标准及其技术参数

4. 安全与可靠性

1）加密认证技术

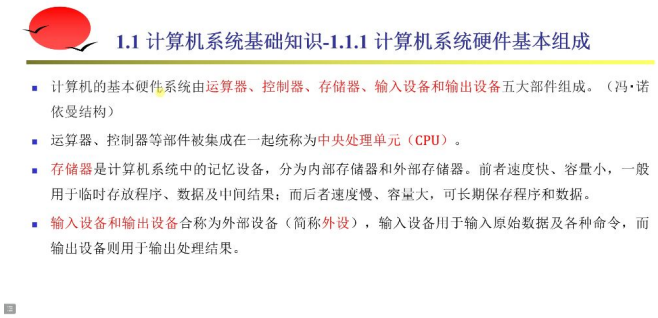
- **加密体系**：
  - 对称加密：加解密使用相同密钥
  - 非对称加密：公钥/私钥配对机制
- **认证技术**：数字签名、数字证书等验证手段

2）可靠性计算

- **系统模型**：
  - 串行系统：可靠性为各部件可靠度乘积
  - 并行系统：可靠性为1减去各部件同时失效概率
- **性能评价**：响应时间、吞吐量等关键指标分析方法

5. 计算机系统基础知识 03:11

1）计算机系统硬件基本组成 03:15



-
- **五大核心部件**：运算器、控制器、存储器、输入设备和输出设备构成计算机基本硬件系统（冯·诺依曼结构）
- **CPU集成**：运算器与控制器集成后称为中央处理单元(CPU)，是计算机的"大脑"
- **存储器分类**：
  - **内部存储器**：速度快容量小（如内存），临时存放程序、数据及中间结果
  - **外部存储器**：速度慢容量大（如硬盘），可长期保存程序和数据
- **外设定义**：输入设备（如键盘鼠标）和输出设备（如显示器）合称外部设备

2）中央处理单元 05:42

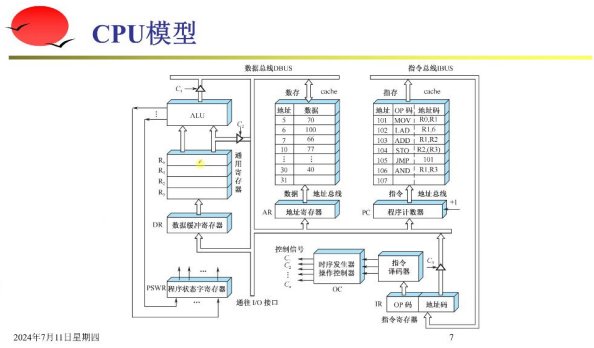
- 运算器 06:16

  -
  - **核心功能**：
    - **算术运算**：执行加减乘除等基本运算
    - **逻辑运算**：处理与、或、非等逻辑操作及逻辑测试

- **核心组件**：
  - **ALU单元**：算术逻辑单元，实际执行运算的核心部件
  - **AC寄存器**：累加器，为ALU提供工作区并暂存运算结果
  - **DR寄存器**：数据缓冲，协调CPU与内存/外设的速度差异
  - **PSW寄存器**：保存运算状态（如进位、溢出标志）
- 控制器 09:03

  - **指令结构**：由操作码(OP)和地址码(A)两部分组成
    - 操作码指示功能（如加减法）
    - 地址码指定操作数位置
  - **核心组件**：
    - **IR寄存器**：保存当前正在执行的指令
    - **PC计数器**：具有双重功能：
      - 指向下一条指令地址
      - 执行后自动递增（增加量为指令长度）
    - **AR寄存器**：存储CPU当前访问的内存地址
    - **指令译码器**：翻译指令操作码确定操作类型
- 寄存器组 12:07

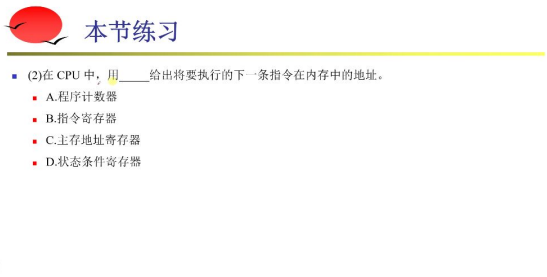
  - **分类标准**：
    - **专用寄存器**：运算器/控制器中的固定功能寄存器
    - **通用寄存器**：功能由程序员决定，主要为ALU提供工作区
  - **可见性分类**：
    - **程序员可见**：通用寄存器组、PSW、PC、AC
    - **程序员不可见**：IR、DR、AR等控制用寄存器
- CPU模型 13:32



- **取指阶段**:
  - PC提供指令地址（如103）
  - 指令通过总线存入IR
  - PC自动递增指向下条指令
- **执行阶段**:
  - 译码器解析操作码
  - AR获取操作数地址
  - DR缓冲内存数据
  - ALU执行运算并更新PSW

- 应用案例 18:21
  - 例题:CPU组成部件

  -
  - 题目解析
    - 关键区分：系统总线（外部总线）不属于CPU内部组件
    - 排除法：ALU（A）、控制单元（C）、寄存器组（D）均为CPU标准组件
    - 答案：B（系统总线）
  - 例题:给出下一条指令地址 20:09



  -
  - 题目解析
    - 核心考点：PC寄存器的双重功能
    - 干扰项分析:
      - IR存储当前指令（非地址）
      - AR存储数据地址（非指令地址）
      - PSW存储状态标志

- 答案：A（程序计数器）

3）数据表示 21:45

- 进制转换 21:52
  - 进制转换的重要性 21:58
    - **计算机存储基础**：计算机内部仅存储二进制数据（0和1），其他进制需转换为二进制才能被计算机处理
    - **理解计算机原理**: 掌握进制转换有助于理解计算机的存储和表示原理
  - 二进制与十进制转换 22:32
    - **二进制表示**: 由0、1组成，逢2进1，可表示为10111B或$(10111)_2$
    - **转换方法**: 按权展开法，从最低位开始，每位系数乘以2^n（n从0开始）
    - **示例**:$(10111)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 16 + 0 + 4 + 2 + 1 = 23$
    - **十进制类比**: 如$136=1 \times 10^2 + 3 \times 10^1 + 6 \times 10^0$，说明不同进制转换原理相同
  - 二进制与八进制转换 25:35
    - **八进制表示**: 由0-7组成，逢8进1，可表示为27Q或$(27)_8$
    - **转换示例**:$(27)_8 = 2 \times 8^1 + 7 \times 8^0 = 16 + 7 = 23$
  - 二进制与16进制转换 26:08
    - **16进制表示**:　由0-9和A-F组成（A=10,…,F=15），逢16进1，可表示为17H或$(17)_{16}$
    - **转换示例**:$(17)_{16} = 1 \times 16^1 + 7 \times 16^0 = 16 + 7 = 23$
- 单位换算 27:20

1.1.3 数据表示

1. 进制转换（复习）

二进制：由0、1表示的数，逢2进1。如：10111B或者$(10111)_2$

八进制：由0.....7表示的数，逢8进1。如：27Q或者$(27)_8$

十六进制：由0.....9和A.....F表示的数，逢16进1。如：17H或者$(17)_{16}$

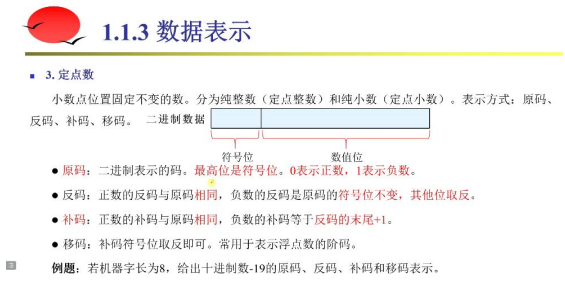
2. 单位换算（复习）

单位	表示	换算
位	bit，b	1b=二进制的一位
字节	Byte，B	1B=8bit
千字节	KB	$1KB=2^{10}B$
兆字节	MB	$1MB=2^{10}KB$
吉字节	GB	$1GB=2^{10}MB$
太字节	TB	$1TB=2^{10}GB$

10

  - **基本单位**:
    - **位(bit)**: 二进制的一位，最小单位
    - **字节(Byte)**: 1B=8bit
  - **扩展单位**:
    - $1KB=2^{10}B=1024B$
    - $1MB=2^{10}KB=1024KB$
    - $1GB=2^{10}MB=1024MB$
    - $1TB=2^{10}GB=1024GB$
  - **换算规律**: 相邻单位间都是乘以$2^{10}(1024)$
- 定点数 28:48
  - **定义**: 小数点位置固定不变的数，包括定点整数和定点小数
  - **表示形式**: 原码、反码、补码、移码
  - **结构组成**:
    - **符号位**: 最高位，0表示正，1表示负
    - **数值位**: 其余位表示数值大小
  - 编码转换方法
    - **原码**: 直接二进制表示，符号位+绝对值

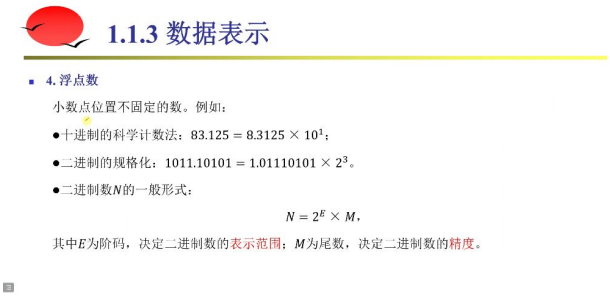
        - 示例：+19=$(00010011)_2$，-19=$(10010011)_2$
    - **反码**:
        - 正数：与原码相同
        - 负数：符号位不变，其他位取反
        - 示例：-19的反码=$(11101100)_2$
    - **补码**:
        - 正数：与原码相同
        - 负数：反码末位+1
        - 快速求法：从最低位开始找到第一个1，该位及右边不变，左边取反
        - 示例：-19的补码=$(11101101)_2$
    - **移码**: 补码符号位取反，主要用于浮点数阶码
        - 示例：-19的移码=$(01101101)_2$
- 例题:十进制数负19的原码、反码、补码和移码表示 32:30



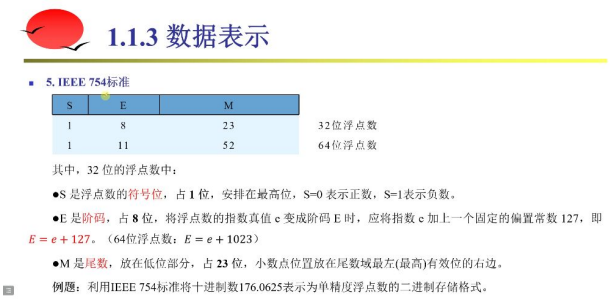
    - **题目解析**:
        - **机器字长**: 8位
        - **转换步骤**:
            - 求19的二进制：$(10011)_2$（通过除2取余法）
            - 补全8位：$(00010011)_2$
            - 原码：符号位1+数值位→$(10010011)_2$
            - 反码：符号位不变，数值位取反→$(11101100)_2$
            - 补码：反码末位+1→$(11101101)_2$
            - 移码：补码符号位取反→$(01101101)_2$
    - **计算机应用**: 计算机内部使用补码进行运算，因其能统一加减法处理
- 定点数的加减运算 40:00

- 原码的加减运算特点 40:25
    - **零表示缺陷**：存在+0（0000）和-0（1000）两种表示形式，实际运算中应视为同一数值但编码不同
    - **真值转换优势**：直接对应二进制真值，如十进制5转换为0101（最高位为符号位）
    - **运算复杂性**:

        - 异号数相加需先比较绝对值大小确定结果符号
        - 实际运算需转换为减法操作（如3+(-2)需计算3-2）
        - 溢出处理机制复杂，需单独判断符号位和数值位溢出情况
    - 反码的加减运算特点 42:01
        - **零表示问题**：仍保留+0（0000）和-0（1111）两种形式
        - **运算改进**：
            - 符号位可直接参与运算（如1010+1100可直接相加）
            - 减法可转换为加法（A-B = A+(-B)的反码）
        - **溢出处理**：运算产生进位时需将溢出位循环加到最低位（称为"循环进位"）
    - 补码的加减运算特点 42:44
        - **零表示唯一性**：0000表示唯一零值，解决原码/反码的双零问题
        - **运算简化机制**：
            - 统一加减法：所有减法$A-B$可转换为$A+(-B)$的补码加法
            - 符号位融合：符号位与数值位同等参与运算（如1011+1101直接相加）
            - 溢出处理：高位溢出直接舍弃（如8位补码相加产生第9位则丢弃）
        - **硬件优势**：只需加法器即可完成所有算术运算，极大简化CPU设计
- 浮点数 43:51

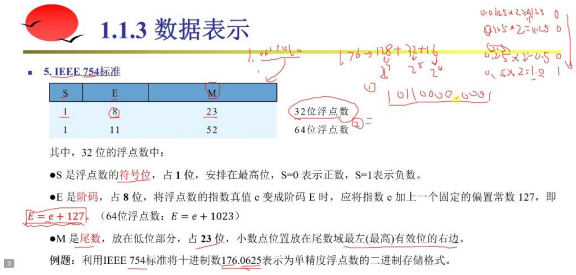


    - **核心特征**：小数点位置动态浮动，类似科学计数法的二进制实现
    - **规格化表示**：
        - 十进制示例：$83.125 = 8.3125 \times 10^1$
        - 二进制示例：$1011.10101 = 1.01110101 \times 2^3$
    - **通用形式**：$N = 2^E \times M$（基数固定为2）
        - **阶码(E)**：决定数值范围，采用移码表示（如8位阶码可表示2^{-128}~2^{127}）
        - **尾数(M)**：决定精度，采用规格化形式（1≤M<2），隐含最高位1
    - **组件分工**：
        - 阶码控制"量级"（类似显微镜的放大倍数）
        - 尾数存储"有效数字"（类似显微镜下的观测细节）
- IEEE754标准 46:42



    - **存储结构**：
        - 32位浮点数：1位符号位(S) + 8位阶码(E) + 23位尾数(M)

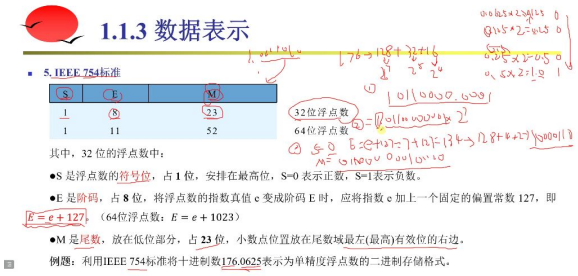
        - 64位浮点数：1位符号位(S) + 11位阶码(E) + 52位尾数(M)
    - **符号位(S)**：
        - 最高位，占1位
        - $S = 0$表示正数，$S = 1$表示负数
    - **阶码(E)**：
        - 32位：$E = e + 127$（指数真值e加固定偏置常数127）
        - 64位：$E = e + 1023$
        - 优势：保证阶码始终为非负数；使数值分布更对称美观
    - **尾数(M)**：
        - 23位（32位）或52位（64位）
        - 小数点位置在最高有效位右侧
        - 存储规格化后的小数部分（即1.xxx中的xxx）
    - 例题：十进制数转浮点数二进制 49:44



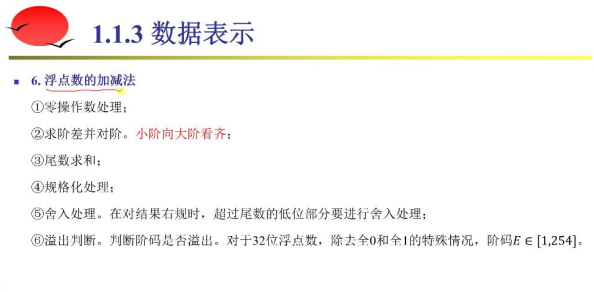
        -
        - 题目解析：
            - **整数部分转换**：
                - 176 = 128(2^7) + 32(2^5) + 16(2^4)
                - 二进制：10110000
            - **小数部分转换**：
                - 0.0625 × 2 = 0.125 → 取0
                - 0.125 × 2 = 0.25 → 取0
                - 0.25 × 2 = 0.5 → 取0
                - 0.5 × 2 = 1.0 → 取1
                - 二进制：0001
            - **合并结果**：
                - 完整二进制：10110000.0001
            - **规格化处理**：
                - 移动小数点：1.01100000001 ×2^7
            - **提取三部分**：
                - S=0（正数）
                - E=7+127=134 → 二进制：10000110
                - M=01100000001（补零至23位）
- 规格化 53:06
    - **定义**：将浮点数表示为$1.xxx \times 2^E$的标准形式
    - **十进制示例**：
        - $83.125 = 8.3125 \times 10^1$
    - **二进制示例**：
        - $1011.1010101 = 1.01110101 \times 2^3$
    - **通用形式**：
        - $N = 2^E \times M$（E决定范围，M决定精度）
    - 例题：小数点规格化移动

- 关键步骤：
  - 确定小数点移动位数（例中移动7位）
  - 保证移动后整数部分为1
  - 指数部分记录移动方向（左移为正，右移为负）
- **数值还原公式**：
  - 32位：$x = (-1)^S \times 1.M \times 2^{E-127}$
  - 64位：$x = (-1)^S \times 1.M \times 2^{E-1023}$
- **特殊说明**：
  - 尾数M非零时，最高有效位必为1（隐含前导1）
  - 实际存储时省略前导1（节省1位存储空间）

- IEEE 754浮点数表示 55:54
  - 32位浮点数结构



    - 14
    - **符号位(S)**: 占1位最高位，$S = 0$表示正数，$S = 1$表示负数
    - **阶码(E)**: 占8位，存储时需将指数真值e加上偏置常数127，即$E = e + 127$（64位浮点数偏置为1023）
    - **尾数(M)**: 占23位低位，采用隐含最高位1的表示法，实际值为$1.M$
  - 数值转换公式
    - **32位转换**:$x = (-1)^S \times 1.M \times 2^{E-127}$
    - **64位转换**:$x = (-1)^S \times 1.M \times 2^{E-1023}$
    - **隐含位处理**: 尾数域最高有效位固定为1但不存储，计算时必须补回该位
  - 例题：浮点数二进制转十进制
    - 题目解析
      - 将16进制$(43301000)_{16}$转为32位二进制
      - 分段解析：符号位0→正数；阶码10000110→十进制134→指数7；尾数01100000000000000000000
      - 计算过程：$1.011000... \times 2^7 = 10110000.0001_2$
      - 结果：$176 + 0.0625 = 176.0625_{10}$
- 浮点数的加减法 01:01:00

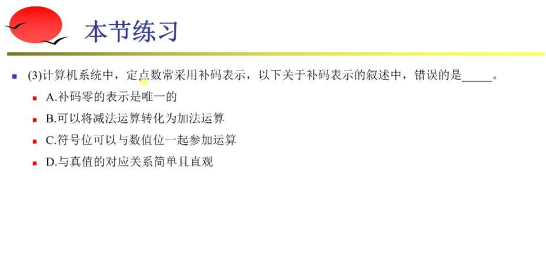


  - 16
  - **操作流程**:
    - 零操作数检查

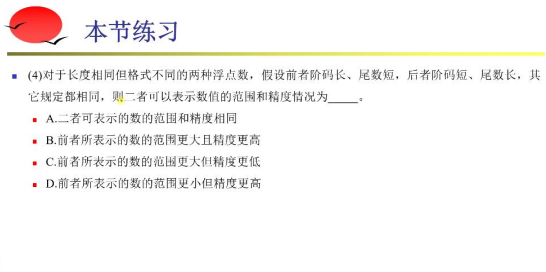
- 求阶差并对阶（小阶向大阶对齐）
- 尾数求和
- 规格化处理（调整为$1.M \times 2^E$形式）
- 舍入处理（超过23位部分按规则舍入）
- 溢出判断（阶码$E \in [1,254]$为有效范围）

- **关键原则**: 对阶时小阶向大阶看齐可避免精度损失

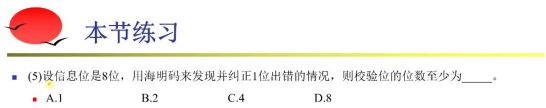
- 应用案例 01:03:03
  - 例题：补码特点



    - 题目解析
      - A选项正确（补码零唯一表示）
      - B选项正确（减法转加法特性）
      - C选项正确（符号位参与运算）
      - D选项错误（原码对应关系更直观）
      - 答案：D
  - 例题：浮点数格式比较



    - 题目解析
      - 阶码长度决定表示范围，尾数长度决定精度
      - 前者阶码长→范围大；尾数短→精度低
      - 答案：C（范围更大但精度更低）
  - 例题：海明码校验位计算



    - 题目解析
      - 应用公式$n + k + 1 \leq 2^k$，$n = 8$时最小$k = 4$
      - 答案：C（至少需要4位校验位）

- 校验码
  - 奇偶校验码


1.1.4 校验码

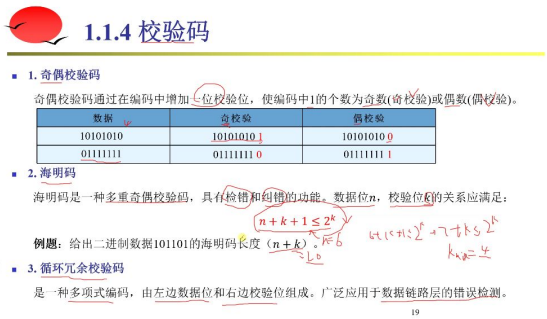
1. 奇偶校验码

奇偶校验码通过在编码中增加一位校验位，使编码中1的个数为奇数(奇校验)或偶数(偶校验)。

数据	奇校验	偶校验
10101010		
01111111		

19

    - **原理**: 增加1位使1的个数为奇数（奇校验）或偶数（偶校验）
    - **示例**:
      - 数据10101010→奇校验101010101，偶校验101010100
      - 数据01111111→奇校验011111110，偶校验011111111
  - 海明码


1.1.4 校验码

1. 奇偶校验码

奇偶校验码通过在编码中增加一位校验位，使编码中1的个数为奇数(奇校验)或偶数(偶校验)。

数据	奇校验	偶校验
10101010	10101010 1	10101010 0
01111111	01111111 0	01111111 1

2. 海明码

海明码是一种多重奇偶校验码，具有检错和纠错的功能。数据位n，校验位k的关系应满足：

$n+k+1 \le 2^k$

例题：给出二进制数据101101的海明码长度（$n+k$）

3. 循环冗余校验码

是一种多项式编码，由左边数据位和右边校验位组成。广泛应用于数据链路层的错误检测。

19

    - **特性**: 多重奇偶校验，具备检错纠错功能
    - **位数关系**: 满足$n + k + 1 \le 2^k$（数据位n，校验位k）
    - **示例**: 数据101101（6位）→$k = 4$→海明码长度10位
  - 循环冗余校验码
    - **组成**: 数据位+校验位构成的多元式编码
    - **应用**: 主要应用于数据链路层错误检测

二、知识小结

知识点	核心内容	考试重点/易混淆点	难度系数
计算机系统硬件组成	冯诺依曼型五大部件（运算器、控制器、存储器、输入设备、输出设备）	**CPU核心部件（运算器+控制器）**，存储器分类（内存 vs 外存）	★★
CPU结构与功能	运算器（ALU、累加寄存器、数据缓冲寄存器、状态条件寄存器）、控制器（指令寄存器、程序计数器、地址寄存器、指令译码器）	**程序计数器（PC）功能**（自增+指向下条指令地址）	★★★

数据表示（定点数）	原码、反码、补码、移码的转换规则	**负数补码快速计算法**（从低位到首个1不变，其余取反），**计算机存储采用补码原因**（加减运算统一性）	★★★★
浮点数表示（IEEE 754）	32位单精度格式（1符号位+8阶码+23尾数），规格化处理（1.xxx×2^e）	**阶码=指数+127**，**隐含最高位1**（尾数解析时需补回）	★★★★
校验码	奇偶校验（奇/偶校验位添加规则）、海明码（数据位n与校验位k关系：n+k+1≤2^k）	**海明码校验位最小位数计算**（如8位数据需4位校验）	★★
浮点数加减运算流程	1. 对阶（小阶向大阶对齐） 2. 尾数求和 3. 规格化 4. 舍入处理 5. 溢出判断	**对阶原则**（避免精度损失），**阶码溢出范围**（1~~254~~ ~~对应指数-126~~127）	★★★