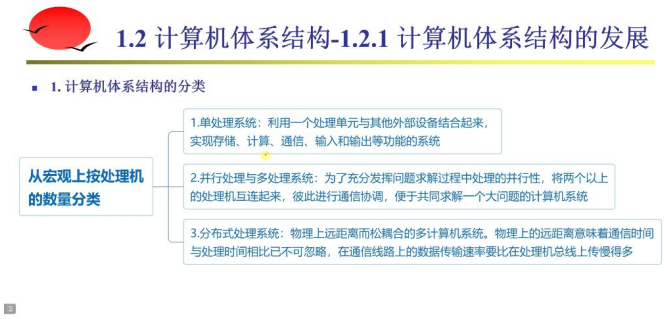
一、计算机体系结构 *00:03*

1. 计算机体系结构的分类 00:12

1）宏观分类（按处理机数量）



- **单处理系统**：利用一个处理单元与其他外部设备结合，实现存储、计算、通信、输入和输出等功能的系统。特点是所有功能由单个处理单元完成。
- **并行处理与多处理系统**：将两个以上的处理机互连，通过通信协调共同求解大问题的计算机系统。核心优势是能充分发挥问题求解过程中的处理并行性。
- **分布式处理系统**：物理上远距离且松耦合的多计算机系统。特点是通信时间不可忽略（相比处理时间），且通信线路数据传输速率显著低于处理机总线。

2）微观分类（按并行程度）

- Flynn分类法
  - **分类依据**：1966年Flynn提出，按指令流和数据流的数量进行分类。
  - **四类系统**：
    - SISD（单指令流单数据流）
    - SIMD（单指令流多数据流）
    - MISD（多指令流单数据流）
    - MIMD（多指令流多数据流）
- 马泽云分类法
  - **分类依据**：1972年美籍华人马泽云提出，按并行度对计算机系统进行结构分类。
  - **四类计算机**：
    - WSBS（字串行位串行）
    - WPBS（字并行位串行）
    - WSBP（字串行位并行）
    - WPBP（字并行位并行）
- Handler分类法

- **分类特点**：1977年德国Wolfgang　Handler提出，基于硬件并行程度和计算并行程度。
- **三个层次**：
  - 处理机级
  - 算术逻辑单元级
  - 逻辑门电路级

- Kuck分类法
  - **分类特点**：1978年David J.Kuck提出，用指令流和执行流及其多重性描述系统控制结构特征。
  - **四类系统**：
    - SISE（单指令流单执行流）
    - SIME（单指令流多执行流）
    - MISE（多指令流单执行流）
    - MIME（多指令流多执行流）

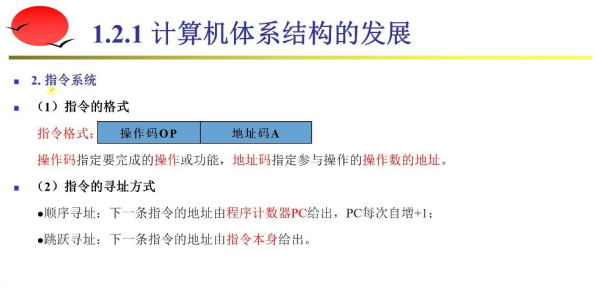
3）指令系统概述 03:20

- **系统定位**：处于软件和硬件的交界面，既不属于纯软件也不属于纯硬件。
- **执行流程**：程序被分解翻译成指令存入内存，CPU从内存中取指令并执行。
- **重要性**：作为软硬件交互的桥梁，在计算机系统中具有关键地位。

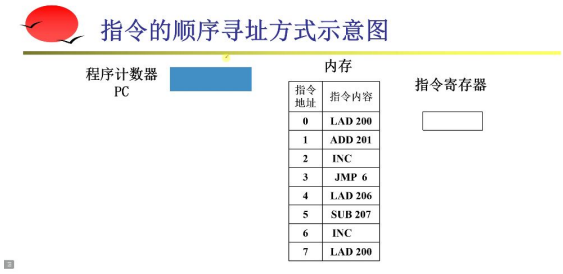
2. 计算机体系结构的发展 01:33

1）指令系统 02:46

- 指令的格式



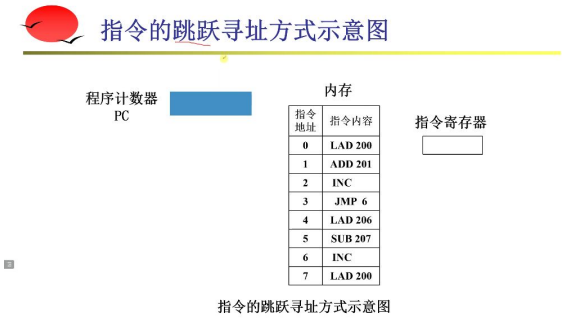
  - 23
  - **组成结构**：由操作码(OP)和地址码(A)两部分构成
  - **操作码功能**：指定指令要完成的操作类型，如加法指令对应加法操作，减法指令对应减法操作，取数指令对应数据读取操作
  - **地址码特性**：指示操作数的地址而非操作数本身，这种设计提高了指令取数据的灵活性
- 指令的寻址方式 04:29
  - 顺序寻址 07:07



指令的顺序寻址方式示意图

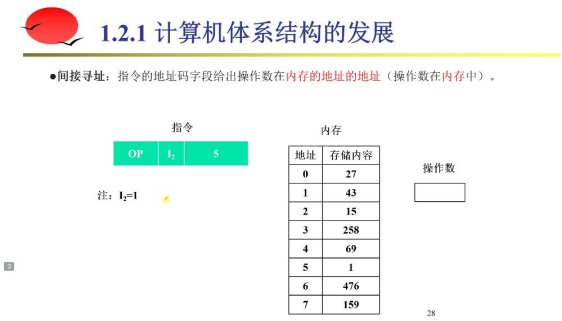
    -

        - **工作原理**：下一条指令地址由程序计数器(PC)给出，PC每次自动加1
        - **执行流程**：
            - PC初始值为0，从内存地址0取出LAD 200指令
            - 指令存入指令寄存器后PC自增为1
            - 继续从地址1取出ADD 201指令
        - **典型应用**：适用于程序顺序执行的情况
    - 跳跃寻址 08:53

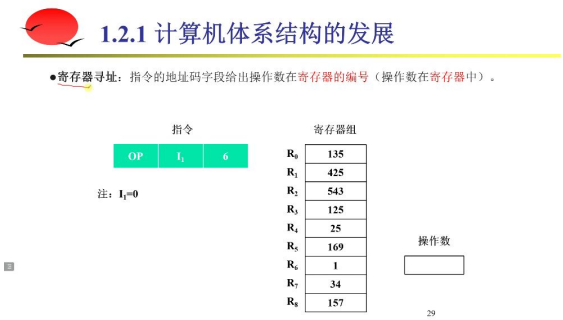


指令的跳跃寻址方式示意图

        - **工作原理**：下一条指令地址由当前指令本身给出
        - **执行示例**：
            - PC值为3时取出JMP 6指令
            - 执行后PC被修改为6而非自增后的4
            - 接着从地址6取出INC指令
        - **典型应用**：用于实现循环结构和条件分支
- 操作数的寻址方式 10:34
    - 立即寻址 10:54
        - **核心特点**：地址码字段直接存储操作数本身而非地址
        - **执行效率**：速度最快，因为无需额外访存操作
        - **指令格式**：操作码OP|操作数D
    - 直接寻址 11:41
        - **地址解析**：地址码字段给出操作数在内存的直接地址
        - **有效地址**：地址码值即为操作数的真实地址
        - **特征标识**：间址特征位为0表示直接寻址
    - 间接寻址 13:12

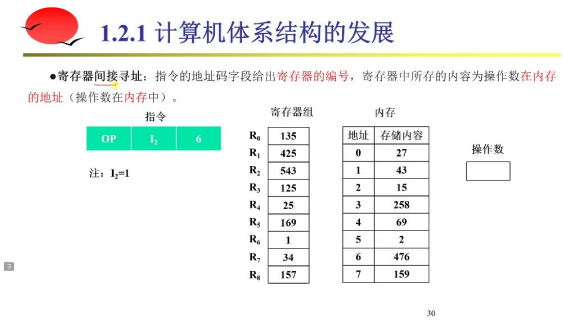


        - **地址解析**：地址码给出的是操作数地址的地址
        - **访存次数**：需要两次内存访问（先取地址，再取操作数）
        - **地址类型**：
            - 形式地址：指令中的地址码（如示例中的5）
            - 有效地址：操作数的真实地址（如示例中的1）
    - 寄存器寻址 15:05



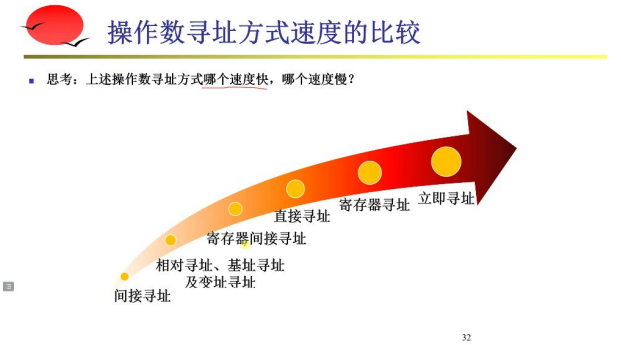
- ■
- ■ **地址解析**：地址码给出寄存器编号，操作数直接存储在寄存器中
- ■ **执行效率**：比内存寻址更快，因为寄存器访问速度高于内存
- ■ **典型示例**：地址码为6时，直接从R6寄存器读取操作数1

- ○ 寄存器间接寻址 16:15



- ■
- ■ **地址解析**：
  - ● 地址码给出寄存器编号
  - ● 寄存器存储操作数的内存地址
  - ● 根据该地址访问内存获取操作数
- ■ **性能比较**：比纯间接寻址快，因为减少了内存访问次数
- ■ **特征标识**：间址特征位为1表示间接寻址

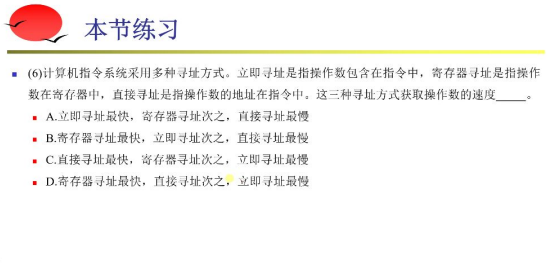
- ● 操作数寻址方式速度的比较 22:01



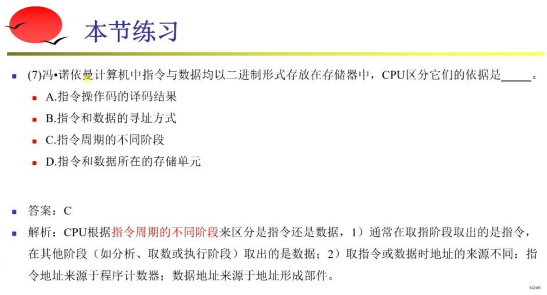
- ○
- ○ 速度排序原理
  - ■ **访问层级差异**：由快到慢依次为：立即寻址 > 寄存器寻址 > 直接寻址 > 寄存器间接寻址 > 相对/基址/变址寻址 > 间接寻址
  - ■ **最快原因**：立即寻址直接获取指令中的操作数，无需额外访问存储单元
  - ■ **最慢原因**：间接寻址需两次内存访问（先取地址再取数据）
  - ■ **关键考点**：考试常给出三种寻址方式要求排序，需掌握每种方式的数据获取路径
- ○ 操作数位置分类
  - ■ **指令本身**：立即寻址（操作数直接编码在指令中）
  - ■ **寄存器**：寄存器寻址、寄存器间接寻址
  - ■ **内存**：直接寻址、相对/基址/变址寻址、间接寻址

        - **典型考题**：可能要求判断特定寻址方式的操作数存储位置
- 计算机指令执行过程 23:09
    - 执行流程
        - **取指令阶段**：
            - PC寄存器输出指令地址到地址总线
            - CPU通过内存读取指令内容并存入IR（指令寄存器）
        - **分析阶段**：
            - 指令译码器解析指令功能
            - 识别需要的操作数类型和数量
        - **执行阶段**：
            - 获取操作数（根据寻址方式）
            - 生成控制信号完成运算/传输等操作
        - **关键特征**：该过程是冯·诺依曼体系结构的核心体现，具有顺序性、循环性
    - 硬件协作
        - **PC寄存器**：始终保存下条指令地址
        - **地址总线**：传输指令/数据的内存地址
        - **IR寄存器**：暂存当前执行的指令内容
        - **译码器**：将机器指令转换为控制信号
        - **注意点**：取指令阶段必然访问内存，而操作数获取可能不需要（如立即/寄存器寻址）
- CPU如何区分指令和数据 24:03
    - 冯诺依曼体系结构
        - **存储方式**： 在冯诺依曼型计算机中，指令和数据是混合存放在内存中的，不进行物理区分。
        - **对比结构**： 哈佛结构采用指令和数据分开存储的方式，与冯诺依曼结构形成鲜明对比。
        - **本质特征**: 无论指令还是数据，在计算机中都以二进制形式(0和1)存在。
    - 区分机制
        - **核心原理**: CPU通过指令周期的不同阶段来区分指令和数据。
        - **阶段划分**:
            - **取指令阶段**: 程序计数器(PC)将指令地址送入地址总线，CPU从内存取出指令存入指令寄存器(IR)
            - **分析阶段**: 指令译码器解析操作码
            - **执行阶段**: 获取源操作数并执行指令
        - **动态区分**: 在指令周期的不同阶段，CPU会根据当前需要自动识别处理的是指令还是数据。
    - 执行过程详解
        - **取指令**:
            - PC提供指令地址
            - 通过地址总线访问内存
            - 指令内容存入IR
        - **分析指令**:
            - 指令译码器解析操作码
            - 确定指令类型和操作数
        - **执行指令**:
            - 获取源操作数
            - 执行相应操作
            - 更新PC指向下一条指令
- 应用案例 25:27

- 例题:指令寻址方式



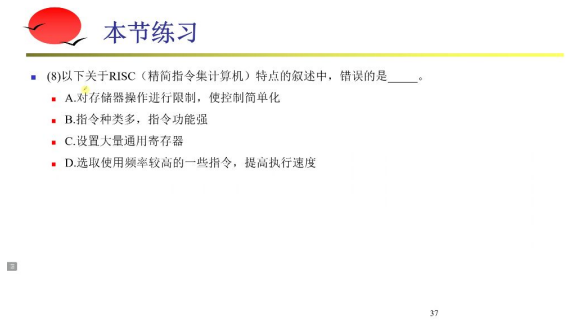
  - 题目解析
    - **速度比较**：立即寻址最快（操作数直接包含在指令中），寄存器寻址次之（操作数在寄存器中），直接寻址最慢（需要访问内存获取操作数）
    - **原理分析**：
      - 立即寻址：直接获取操作数，无需额外访问
      - 寄存器寻址：只需访问寄存器，速度仅次于立即寻址
      - 直接寻址：需要访问内存，速度最慢
    - **答案确认**：正确答案为A选项（立即寻址>寄存器寻址>直接寻址）
- 例题:指令区分阶段 27:05



  - 题目解析
    - **区分原理**：CPU通过指令周期的不同阶段来区分指令和数据
    - **具体机制**：
      - 取指阶段取出的是指令
      - 分析/取数/执行阶段取出的是数据
    - **地址来源差异**：
      - 指令地址来源于程序计数器(PC)
      - 数据地址来源于地址形成部件
    - **答案确认**：正确答案为C选项（指令周期的不同阶段）
- 指令集的分类 27:34
  - CISC与RISC对比

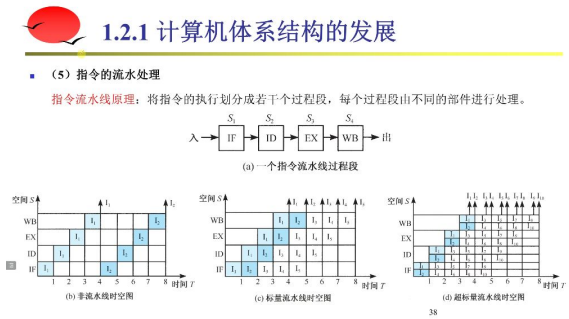
  - **基本定义**:
    - CISC(复杂指令集): 指令数量多，各条指令按顺序串行执行

        - RISC(精简指令集): 减少指令总数，采用优化编译、硬布线、重叠寄存器窗口等技术
    - **编程特性**:
        - CISC: 指令多，编程灵活
        - RISC: 编程量更大，采用较多通用寄存器
    - **实现方式**:
        - CISC: 采用微程序控制技术(用软件设计硬件的技术)
        - RISC: 采用硬布线逻辑控制优化编译程序，使用流水线技术加快执行速度
    - **指令特征**:
        - 指令数目: CISC多，RISC少
        - 指令长度: CISC为可变长指令，RISC大部分为等长指令(如64位)
        - 控制器复杂性: CISC复杂，RISC简单
        - 寻址方式: CISC较丰富(提高编程灵活性)，RISC较少(提高效率)
    - **典型示例**:
        - 乘法运算: CISC可直接提供乘法指令，RISC需将乘法转换为加法运算实现
- 例题:RISC特点 30:31



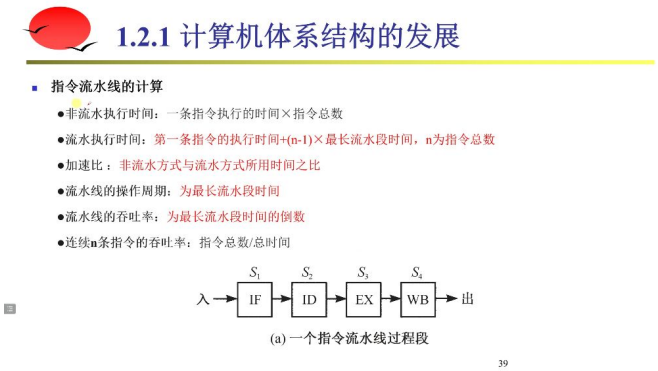
    - 题目解析
        - **题目分析**: 要求识别关于RISC特点的错误叙述
        - **选项解析**:
            - A: 正确，RISC确实对存储器操作进行限制以简化控制
            - B: 错误，这是CISC的特点而非RISC
            - C: 正确，RISC会设置大量通用寄存器
            - D: 正确，RISC会选取使用频率高的指令提高执行速度
        - **正确答案**: B
        - **考点提示**: 需准确区分RISC与CISC的核心特征差异
- 指令的流水处理 31:57
    - **基本原理**：将指令执行划分为多个过程段（如取指IF、译码ID、执行EX、写回WB），每个过程段由专用部件处理，形成类似工厂流水线的作业方式
    - **典型过程段**:
        - 取指(IF)：由取指部件完成
        - 译码(ID)：由译码部件完成
        - 执行(EX)：由执行部件完成
        - 写回(WB)：由写回部件完成
    - **部件特点**：不同部件执行速度可能不同，例如取指需$2\Delta t$，译码需$3\Delta t$，执行需$4\Delta t$，写回需$5\Delta t$
    - 非流水线时空图 34:06
        - **执行方式**：串行执行，前一条指令完全执行完毕后才开始下一条指令
        - **时间计算**:
            - 单条指令时间：各段执行时间之和，如$T_{单} = 2 + 3 + 4 + 5 = 14\Delta t$

        - n条指令总时间：$T_{总} = n \times T_{单} = 14n\Delta t$
    - **效率问题**：任一时刻只有1个部件工作，其他部件空闲（如译码时取指部件闲置）
- 标量流水线时空图 37:00
    - **执行方式**：各部件并行工作，前一条指令进入下一段时，当前段立即处理下一条指令
    - **时间计算**：
        - 第一条指令时间：$T_{首}$ = 各段时间和 = $14\Delta t$
        - 后续指令时间：受最慢段限制（瓶颈段），如写回需$5\Delta t$，则每$5\Delta t$完成1条指令
        - n条指令总时间：$T_{总} = T_{首} + (n-1) \times T_{瓶颈} = 14 + (n-1) \times 5$
    - **效率优势**：相比非流水线，时间从$14n\Delta t$缩短至$14 + 5(n-1)\Delta t$
    - **瓶颈问题**：流水线速度取决于最慢的段，如写回段$5\Delta t$决定整体吞吐率
- 超标量流水线



    - **核心思想**：通过增加硬件资源（如双取指部件、双译码部件）实现多条指令并行
    - **执行方式**：相当于多条流水线同时工作，用空间换时间
    - **应用场景**：适用于需要更高指令吞吐率的处理器设计

2）指令流水线的计算 44:41



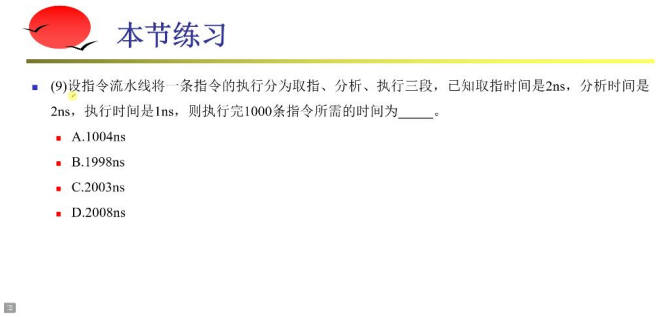
- 执行时间计算
    - **非流水执行时间**: 一条指令执行的时间乘以指令总数，公式为$T_{非流水} = t_{单指令} \times n$
    - **流水执行时间**: 由两部分组成，第一条指令完整执行时间加上剩余指令的最长段执行时间，公式为$T_{流水} = t_{第一条} + (n-1) \times t_{最长段}$
- 性能指标
    - **加速比**: 非流水方式与流水方式所用时间之比，即$S = \frac{T_{非流水}}{T_{流水}}$
    - **操作周期**: 等于流水线中最长段的时间，决定了流水线的工作节奏
    - **基础吞吐率**: 操作周期的倒数，表示单位时间内完成的指令数，公式为$TP = \frac{1}{t_{最长段}}$

- **连续吞吐率**: 对于n条指令，计算公式为$TP_n = \frac{n}{T_{流水}}$，反映实际连续执行时的效率

- 流水线结构
  - **典型五段式**: 包含取指(IF)、译码(ID)、执行(EX)、访存(MEM)、写回(WB)五个阶段
  - **关键路径**: 各段中执行时间最长的段将决定整个流水线的性能瓶颈

二、应用案例 *45:54*

1. 例题:指令流水线计算



1）题目解析

- **流水线分段**：指令执行分为取指（2ns）、分析（2ns）、执行（1ns）三个过程段
- **计算原理**:
  - 第一条指令时间 = 各段耗时总和 =2 + 2 + 1 = 5ns
  - 剩余$n-1$条指令时间 =$(n-1)$ × 最长流水段时间=999 × 2 = 1998ns

总时间公式：$T = kt + (n-1)max(t_i)$

- （k为段数，t为各段时间和，n为指令数，$max(t_i)$为最长段耗时）
- **具体计算**：5 + 999 × 2 = 2003ns
- **正确答案**：C选项（2003ns）
- **关键点**：识别最长流水段时间（本例中为2ns）是计算剩余指令时间的关键

三、知识小结

知识点	核心内容	考试重点/易混淆点	难度系数
计算机体系结构分类	宏观/微观分类法：单处理系统、并行多处理系统、分布式处理系统	**微观分类标准**（处理机数量）vs宏观分类标准	★★☆☆☆
FLYNN分类法	按指令流和数据流数量分类：SISD/SIMD/MISD/MIMD	**MISD实际不存在**，SIMD与向量处理关系	★★★☆☆
指令系统特性	软件硬件交界面、操作码+地址码结构	地址码存储的是**操作数地址**而非数据本身	★★☆☆☆
寻址方式对比	8种寻址方式：立即/直接/间接/寄存器/基址/变址/相对寻址	**速度排序**：立即>寄存器>直接>间接	★★★★☆

		（访问内存次数差异）	
指令执行流程	取指→译码→取数→执行四阶段流水线	**PC自增机制**与跳跃寻址的区别	★★★☆☆
CISC vs RISC	CISC：复杂指令/变长/微程序控制；RISC：精简指令/等长/硬布线	**关键区别**：指令数量与控制器复杂度	★★★★☆
流水线技术	标量流水线时间公式：kΔt+(n-1)*max(Δt)	**瓶颈段决定吞吐率**（最长阶段时间）	★★★★☆
超标量流水线	空间换时间：多套功能部件并行	与标量流水线的**硬件资源差异**	★★★☆☆