绝密★启用前

2024 – 2025 年河南工业大学期末考试

计算机组成原理

注意事项：

1. 答卷前，考生务必将自己的姓名和准考证号填写在答题卡上。
2. 回答选择题时，选出每小题答案后，用铅笔把答题卡对应题目的答案标号涂黑。如需改动，用橡皮擦干净后，再选涂其它答案标号。回答非选择题时，将答案写在答题卡上。写在本试卷上无效。
3. 考试结束后，将本试卷和答题卡一并交回。请认真核对监考员在答上所粘贴的条形码上的姓名、准考证号与您本人是否相符。

一、选择题：本大题共 10 小题，每小题 1 分，共 10 分。在每小题给出的四个选项中，只有一项是符合题目要求的。

1. 冯·诺依曼机工作的基本方式的特点是

   A. 多指令流单数据流　　B. 按地址访问并顺序执行指令

   C. 堆栈操作　　D. 存储器按内容选择地址

2. 在机器数中，零的表示形式是唯一的。

   A. 原码　　B. 补码　　C. 移码　　D. 反码

3. 在定点二进制运算器中，减法运算一般通过 ____ 来实现。

   A. 原码运算的二进制减法器　　B. 补码运算的二进制减法器

   C. 原码运算的十进制加法器　　D. 补码运算的二进制加法器

4. 某计算机字长 32 位，其存储容量为 4MB，若按半字编址，它的寻址范围是

   A. 0~4MB　　B. 0~2MB　　C. 0~2M　　D. 0~1M

5. 主存贮器和 CPU 之间增加 cache 的目的是

   A. 解决 CPU 和主存之间的速度匹配问题

   B. 扩大主存贮器容量

   C. 扩大 CPU 中通用寄存器的数量

   D. 既扩大主存贮器容量，又扩大 CPU 中通用寄存器的数量

6. 单地址指令中为了完成两个数的算术运算，除地址码指明的一个操作数外，另一个常需采用

   A. 堆栈寻址方式　　B. 立即寻址方式

   C. 隐含寻址方式　　D. 间接寻址方式

7. 同步控制是

A. 只适用于 CPU 控制的方式　　B. 只适用于外围设备控制的方式

C. 由统一时序信号控制的方式　　D. 所有指令执行时间都相同的方式

8. 描述 PCI 总线中基本概念不正确的句子是

A. PCI 总线是一个与处理器无关的高速外围设备

B. PCI 总线的基本传输机制是猝发或传送

C. PCI 设备一定是主设备

D. 系统中只允许有一条 PCI 总线

9. CRT 的分辨率为 1024×1024 像素，像素的颜色数为 256，则刷新存储器的容量为

A. 512KB　　B. 1MB　　C. 256KB　　D. 2MB

10. 为了便于实现多级中断,保存现场信息最有效的办法是采用___

A. 通用寄存器　　B. 堆栈　　C. 存储器　　D. 外存

二、填空题（共 8 题，每空 1 分，共 8 分）

1. 在计算机术语中，将运算器和控制器以及_____合在一起称为_____, 而将_____和存储器合在一起称为_____。
2. 数的真值变成机器码可采用_____表示法，_____表示法和_____表示法。
3. 广泛使用的_____不如_____高速,它们都是半导体随机读写存储器。
4. 形成指令地址的方式，称为_____方式,有_____寻址和_____寻址。
5. CPU 从_____取出一条指令的命令并执行这条指令称为_____。由于各种指令操作功能不同，各种指令的指令周期是_____。
6. 微型计算机的标准总线从 16 位的_____总线，发展到 32 位的_____总线和_____总线,又进一步发展到 64 位的 PCI 总线。
7. VESA 标准是一个可扩展的标准。它除了兼容传统的_____等显示方式外,还支持_____像素光栅,每像素点_____颜色深度。
8. 中断处理过程可以_____进行。_____的设备可以中断_____的中断服务程序。

三、简答题（共 3 题，每题 5 分，共 15 分）

1. 什么是刷新存储器？其存储容量与什么因素有关?
2. 外设的 I/O 控制方式分为哪几类? 各具什么特点?
3. 什么是指令周期?什么是机器周期?什么是时钟周期?三者有什么关系?

四、综合题(共 5 题, 每题 10 分, 共 50 分)

1. 已知 $x=-0.01111, y=+0.11001$, 求 $[x]_{补}, [-x]_{补}, [y]_{补}, [-y]_{补}, x+y=?, x-y=?$

2. 假设机器字长 16 位，主存储容量为 128K 字节，指令长度为 16 位或 32 位，其有 128 条指令，设计计算机指令格式，要求有直接、立即数、相对、基值、间接、变址六种寻址方式。

3. 某机字长 32 位，常规设计的存储空间 ≤ 32M，若将存储空间扩至 256M，请提出一种可能方案.

4. 如图 1, 有两条独立的总线和两个独立的存贮器。已知指令存贮器 IM 最大容量为 16384 字（字长 18 位），数据存器 DM 最大容量是 65536 字（字长 16 位）。各寄存器均有“打入”（R_{in}）“送出”（R_{out}）控制命令，但图中未标出

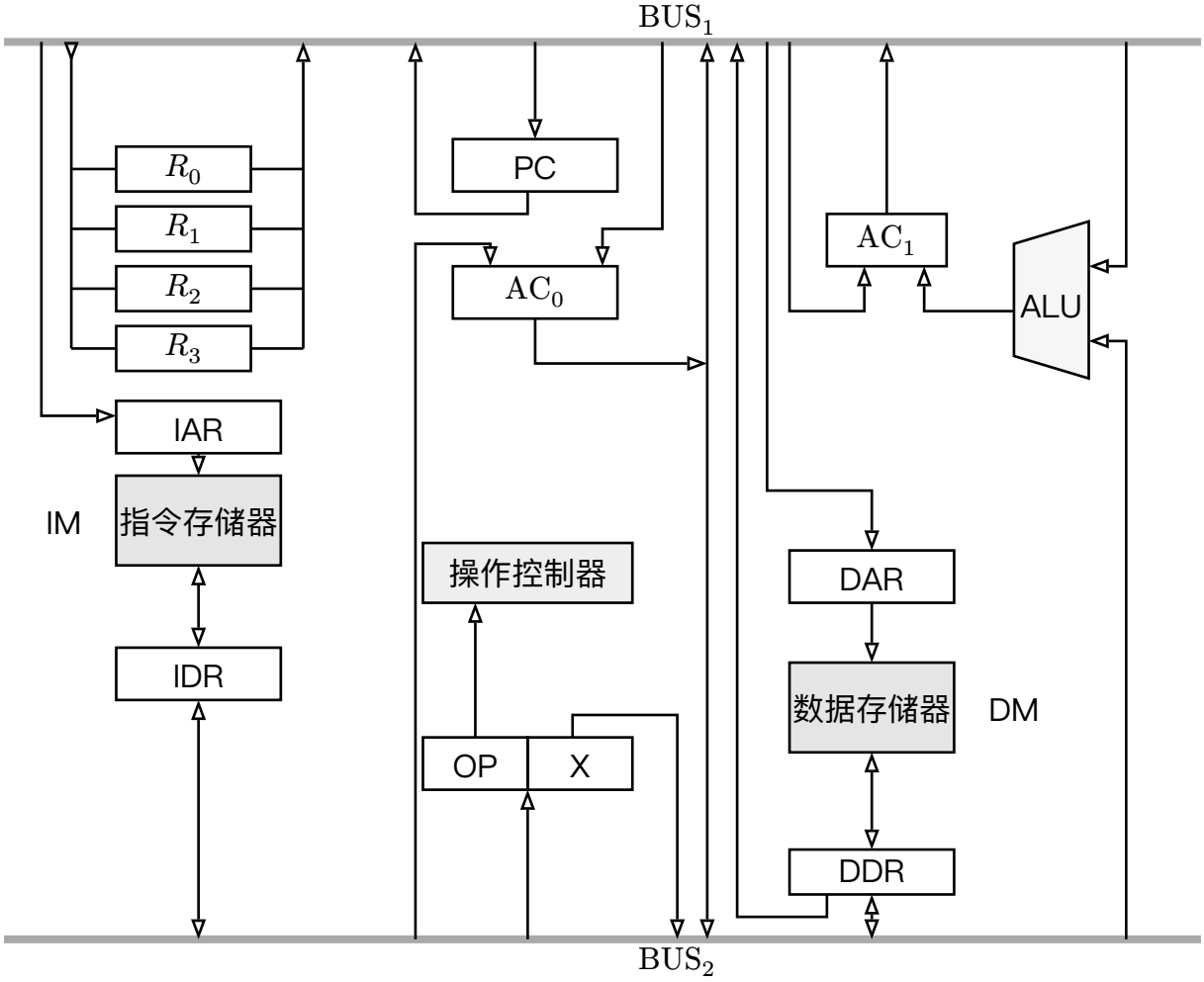
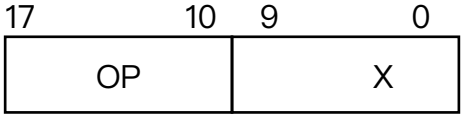


图 1

假设处理机格式为:



加法指令可写为“ADD $X(R_i)$”。其功能是 $(\text{AC}_o + ((R_i) + X) \rightarrow \text{AC}_I)$，其中 $((R_i) + X)$ 部分通过寻址方式指向数据存贮器，现取 R 为R_I。试画出 ADD 指令从取指令开始到执行结束的操作序列图，写明基本操作步骤和相应的微操作控制信号。

1. 假设某磁盘，每面有 220 道：已知磁盘转速 =3000 装/分。数据传输率为 175000B/s。求该磁盘总容量.

1 参考答案

1.1 选择题

1. 正确答案：A

    1. B. **按地址访问并顺序执行指令**
        - 存储程序并按地址访 2w1 问
        - 指令和数据以同等地位存放在存储器中
        - 指令按序执行
        - **正确。**
    2. A. **多指令流单数据流**
        - 这是 Flynn 分类法中的 SISD 计算机特征，不是冯·诺依曼机的基本特点
        - **错误。**
    3. C. **堆栈操作**
        - 堆栈操作是一种**数据结构**的操作方式，不是冯·诺依曼机的基本特点
        - **错误。**
    4. D. **存储器按内容选择地址**
        - 这是相联存储器的特点，不是冯·诺依曼机的基本特点
        - **错误。**

2. 正确答案：A

    1. A. **原码**
        - 原码表示法是唯一的零表示形式
        - **正确。**
    2. B. **补码**
        - 补码表示法有两个零表示形式
        - **错误。**
    3. C. **移码**
        - 移码表示法有两个零表示形式
        - **错误。**
    4. D. **反码**
        - 反码表示法有两个零表示形式
        - **错误。**

3. 正确答案：D

    1. A. **原码运算的二进制减法器**
        - 原码运算的二进制减法器不能实现补码运算
        - **错误。**
    2. B. **补码运算的二进制减法器**
        - 补码运算的二进制减法器可以实现补码运算
        - **错误。**
    3. C. **原码运算的十进制加法器**
        - 原码运算的十进制加法器不能实现补码运算
        - **错误。**
    4. D. **补码运算的二进制加法器**
        - 补码运算的二进制加法器可以实现补码运算
        - **正确。**

4. 正确答案：B

> 📌 **已知条件**
>
> | 项目 | 数值 |
> | --- | --- |
> | 字长 | 32 位（= 4 字节） |
> | 存储容量 | 4 MB = 4 × 1024 × 1024 = 4,194,304 字节 |
> | 编址单位 | 半字（即 2 字节） |
>
> ---
>
> “字（word）”是计算机内部数据处理的最基本单位，它的长度（word length）指的是一“字”有多少 位（bit）。32 位 = 4 字节（byte），**1 字节 = 8 位。**
>
> ---
>
> ❓ **半字编址释义**
> - 半字长度为 2 字节。
> - 字节编址：每个内存地址对应 1 个字节单元。

- 半字编址：每个内存地址对应 2 个字节单元。

🔍 **寻址范围计算** 寻址范围指可唯一标识的地址单元总数。

$$\begin{aligned}\text{地址单元总数} &= \frac{\text{总存储容量（字节）}}{\text{编址单位大小（字节/地址）}} \\ &= 4,194,304\frac{\text{字节}}{2\ \text{字节/地址}} \\ &= 2,097,152\ \text{个地址}\end{aligned}$$

✅ **结论** 该计算机的寻址范围为 0 至 $2,097,151$，共计 $2,097,152$ 个地址。$2,097,152 = 2 \times 1024 \times \frac{1024}{1024} = 2 \times 2^{10} = 2^{1} \times 2^{10} = 2^{11}$ $2,097,152 = 2 \times 1,048,576 = 2\mathrm{M}$ (其中 $1\mathrm{M} = 2^{20}$) 因此，寻址范围为 $2\mathrm{M}$ 个地址。

选项分析：

- A. 0 4MB：超出实际可寻址范围，错误
- B. 0 2MB：符合计算结果，正确
- C. 0 2M：单位不对，错误
- D. 0 1M：范围太小，错误

5. 正确答案：A

1. A. **解决 CPU 和主存之间的速度匹配问题**
   - Cache 的主要作用是提高 CPU 和主存之间的数据传输速度
   - **正确。**
2. B. **扩大主存贮器容量**
   - Cache 并不直接扩大主存容量
   - **错误。**
3. C. **扩大 CPU 中通用寄存器的数量**
   - Cache 与通用寄存器无关
   - **错误。**
4. D. **既扩大主存贮器容量，又扩大 CPU 中通用寄存器的数量**
   - Cache 并不直接扩大主存容量或寄存器数量
   - **错误。**

6. 正确答案：C

- 在 **单地址指令** 中，一个操作数通常是由地址字段给出的，而另一个操作数必须来自于某个固定的地方。这个“固定的地方”通常是 累加器（AC）。

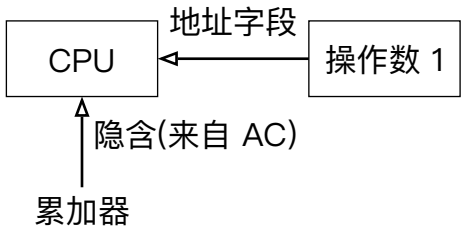


图 3　单地址指令的操作数图示

1. A. **堆栈寻址方式**
   - 堆栈寻址方式是隐含寻址方式的一种
   - **错误。**
2. B. **立即寻址方式**
   - 立即寻址方式不适用于单地址指令
   - **错误。**
3. C. **隐含寻址方式**
   - 隐含寻址方式是单地址指令中常用的寻址方式
   - **正确。**
4. D. **间接寻址方式**
   - 间接寻址方式不适用于单地址指令
   - **错误。**

寻址方式	适用场景
堆栈寻址	‣ 用于堆栈操作指令（PUSH, POP） ‣ 操作数位于栈顶 ‣ 地址隐含在栈顶指针中
立即寻址	‣ 操作数直接包含在指令中 ‣ 常用于装入常数 ‣ 不需要访存
隐含寻址	‣ 操作数在固定寄存器（如累加器）中 ‣ 地址不需要显式指定 ‣ 节省指令长度
间接寻址	‣ 指令地址字段给出操作数的地址 ‣ 需要多次访存 ‣ 可访问更大地址空间

表 1　各种寻址方式的使用场景比较

7. 正确答案：C

同步控制（Synchronous Control）是指：控制单元按照 统一的时钟信号（clock） 来驱动各个部件进行协调工作。

1. A. **只适用于 CPU 控制的方式**
   - 同步控制不仅适用于 CPU，也适用于外围设备
   - **错误。**
2. B. **只适用于外围设备控制的方式**
   - 同步控制不仅适用于外围设备，也适用于 CPU
   - **错误。**
3. C. **由统一时序信号控制的方式**
   - 同步控制是由统一时序信号控制的
   - **正确。**
4. D. **所有指令执行时间都相同的方式**
   - 同步控制不一定要求所有指令执行时间相同
   - **错误。**

8. 正确答案：C

1. A. **PCI 总线(Peripheral Component Interconnect)是一个与处理器无关的高速外围设备**（platform–independent）
   - **正确。**
2. B. **PCI 总线的基本传输机制是猝发或传送**
   - PCI 总线的基本传输机制是突发或传送，一旦握手完成，可以连续传输多个数据周期
   - **正确。**
3. C. **PCI 设备一定是主设备**
   - PCI 设备不一定是主设备,它可以是：
     - 主设备（master）：主动发起传输（如显卡）
     - 从设备（slave）：被动响应 CPU 的访问（如网卡）
   - **错误。**
4. D. **系统中只允许有一条 PCI 总线**
   - 根据百度百科的相关信息，PCI 总线并非只能有一条。
   - **错误。**但是答案上说只可以有一条🥲，大家以老师的为准

9. **正确答案：B**

解析：

1. **已知条件**
   - 分辨率：1024 × 1024 像素
   - 每像素颜色数：256 种（表示为 8 位 = 1 字节）
   - 求：刷新存储器容量（显存大小）
2. **计算过程**
   1. 每个像素所需位数： $256 = 2^8$，需要 8 位（1 字节）存储颜色信息
   2. 总像素数： 1024 × 1024 = 1,048,576 像素
   3. 显存大小： 1,048,576 像素 × 1 字节/像素 = 1,048,576 字节 = 1 MB

10. **正确答案：B**

解析：

❓ **问题背景** 在计算机系统中，中断允许外部事件（如 I/O 完成）或内部异常（如除零错误）打断当前正在执行的程序，转而去执行特定的中断服务程序（ISR）。处理完中断后，系统需要能够精确地恢复到被打断前的状态，继续执行原程序。这个被打断前的状态信息集合，称为“现场”（Context），通常包括程序计数器（PC）、状态寄存器（PSW）、通用寄存器等。

🎯 **核心挑战：多级中断** 当一个中断服务程序正在执行时，如果允许被更高优先级的中断再次打断，就形成了多级中断（或中断嵌套）。这种情况下，每次中断发生时都需要保存当前现场，并在中断返回时按正确的顺序恢复现场。

💡 **解决方案对比**

保存方式	是否适合多级中断	原因说明
通用寄存器	✗ 不适合	数量有限，新中断保存现场时会覆盖旧中断的现场信息，导致无法正确返回。
堆栈 (Stack)	✓ 非常适合	堆栈具有“后进先出”（LIFO）的特性。每次中断发生时，将现场信息压入栈顶；中断返回时，从栈顶弹出信息恢复现场。完美支持嵌套，后发生的中断先处理完并恢复，不影响之前的中断现场。
固定内存区域	✗ 不适合	如果为每个中断预留固定内存区域，管理复杂且浪费空间；如果所有中断共用一个固定区域，同样存在覆盖问题，无法支持嵌套。
外存	✗ 极不适合	外存访问速度太慢，中断处理要求快速响应，将现场保存到外存效率极低，不现实。

🧠 **堆栈工作流程示例（多级中断）**

1. 主程序 P 正在执行。
2. 中断 A 发生：保存 P 的现场到栈顶，跳转执行 ISR A。
3. ISR A 执行中，更高优先级的中断 B 发生：保存 ISR A 的现场到栈顶（现在栈顶是 ISR A 的现场，下面是 P 的现场），跳转执行 ISR B。
4. ISR B 执行完毕：从栈顶弹出 ISR A 的现场，恢复并返回 ISR A 继续执行。
5. ISR A 执行完毕：从栈顶弹出 P 的现场，恢复并返回主程序 P 继续执行。

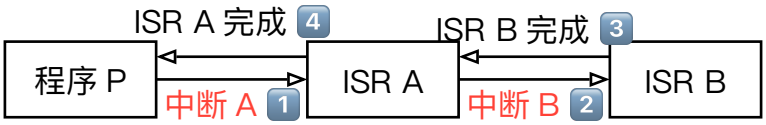


图 4　多级中断的执行流程和堆栈变化

✅ **结论** 为了有效、可靠地实现多级中断，使用 **堆栈** 来保存和恢复现场信息是最普遍且最高效的方法。它利用 LIFO 原则自然地处理了中断嵌套的现场管理问题。

1.2 填空题

1. 在计算机术语中，将运算器和控制器以及寄存器组合在一起称为中央处理器，而将外部设备和存储器合在一起称为计算机主机。

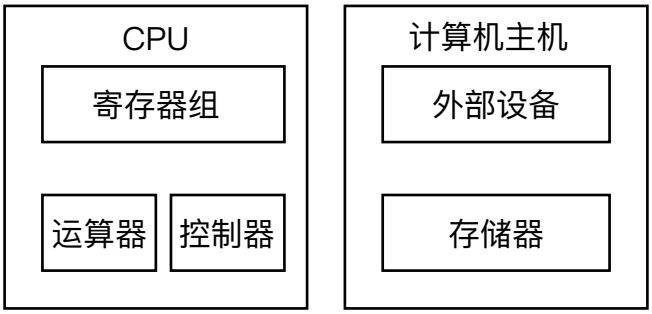


图 5　计算机主机组成示意图

2. 数的真值变成机器码可采用 补码 表示法， 原码 表示法和 反码 表示法。

   解析：

数值	原码	反码	补码
+5	0101	0101	0101
−5	1101	1010	1011
+0	0000	0000	0000
−0	1000	1111	N/A

表 2　4 位二进制数的不同表示法示例

3. 广泛使用的 动态随机存取存储器 不如 静态随机存取存储器 高速,它们都是半导体随机读写存储器。(推荐写英文缩写)

   解析：

- **DRAM (Dynamic RAM)**：利用电容存储电荷来表示数据（0 或 1）。电容会漏电，需要定期刷新（充电）以维持数据，因此称为“动态”。结构简单，集成度高，成本较低，容量大，但速度相对较慢，主要用作计算机的主存（内存）。
- **SRAM (Static RAM)**：利用触发器（通常由多个晶体管组成）来存储数据。只要供电，数据就能保持不变，无需刷新，因此称为“静态”。结构复杂，集成度低，成本高，容量小，但速度快，主要用作 CPU 的高速缓存（Cache）。

4. 形式指令地址的方式，称为 立即 方式,有 直接 寻址和 间接 寻址。

   解析：寻址方式是指指令中如何给出操作数或指令的地址。它们的对比**详见 表 5**

5. CPU 从 主存 取出一条指令的命令并执行这条指令称为 指令周期 。由于各种指令操作功能不同，各种指令的指令周期是 不相同 。

   解析：指令周期是 CPU 执行一条指令所花费的全部时间。它通常包含若干个机器周期（CPU 周期），而每个机器周期又包含若干个时钟周期（T 周期）。 一个典型的指令周期包括：

   1. **取指周期 (Fetch Cycle)**：从主存获取指令。
   2. **间址周期 (Indirect Cycle)**：如果指令是间接寻址，需要访问内存获取有效地址。
   3. **执行周期 (Execute Cycle)**：执行指令指定的操作。
   4. **中断周期 (Interrupt Cycle)**：如果允许中断且有中断请求，则响应中断。

   由于不同指令的操作复杂度和寻址方式不同（例如，有的指令需要访存取操作数，有的不需要；有的需要多次访存），完成这些阶段所需的时间也不同，因此指令周期通常是可变的，即不相同。

6. 微型计算机的标准总线从 16 位的 ISA 总线，发展到 32 位的 EISA 总线和 VESA 总线,又进一步发展到 64 位的 PCI 总线。

   解析：

总线标准	位宽	频率	最大带宽	特点	首次发布
ISA(Industry Standard Architecture)	16 位	8MHz	8MB/s	IBM PC/AT 标准总线	1984 年 IBM
EISA(Extended ISA)	32 位	8.33MHz	33MB/s	ISA 扩展，向下兼容	1988 年 PC 厂商联盟
VESA(Video Electronics Standards Association)	32 位	33MHz	132MB/s	局部总线，主要用于显卡	1992 年 VESA 组织
PCI(Peripheral Component Interconnect)	32/64 位	33/66MHz	533MB/s	高性能，广泛应用	1993 年 Intel

表 3　计算机总线技术发展历程

7. VESA 标准是一个可扩展的标准。它除了兼容传统的 VGA 等显示方式外,还支持 1280 × 1024 像素光栅,每像素点 24 颜色深度。

   解析：
   - **VGA (Video Graphics Array)**：IBM 于 1987 年推出的标准，分辨率为 640x480，16 色；或 320x200，256 色。
   - **SVGA (Super VGA)**：VGA 的扩展，没有统一标准，通常指比 VGA 更高分辨率和更多颜色的显示模式，如 800x600, 1024x768 等。
   - **VESA (Video Electronics Standards Association)**：一个制定视频标准的组织。它制定了 VESA Local Bus (VLB) 总线标准，以及 **VESA BIOS Extensions (VBE)**，允许软件以标准方式访问 SVGA 显卡的高分辨率和多颜色模式。
   - **颜色深度 (Color Depth)**：指每个像素点能表示的颜色数量，通常用位数表示。16 位颜色深度（High Color）可以表示 2^16 = 65536 种颜色。题目中的“16 颜色深度”可能指 16 位颜色深度，而非仅 16 种颜色。VBE 标准支持多种颜色深度，包括 8 位（256 色）、15/16 位（高彩）、24 位（真彩）等。

8. 中断处理过程可以 嵌套 进行。 优先级高 的设备可以中断 优先级低 的中断服务程序。

   解析：中断是计算机处理外部或内部紧急事件的一种机制。当中断发生时，CPU 暂停当前任务，转去执行相应的中断服务程序 (ISR)。

   中断嵌套 (Interrupt Nesting)：指在一个中断服务程序执行期间，如果发生了一个优先级更高的中断请求，CPU 会暂停当前正在执行的低优先级中断服务程序，转而去处理高优先级的中断。待高优先级中断处理完毕后，再返回继续执行被中断的低优先级中断服务程序。这需要 CPU 在进入中断服务程序时保存现场（寄存器状态等），并在中断返回时恢复现场。

实现中断嵌套的关键在于中断优先级管理和现场保护机制。并非所有系统都允许或支持无限层嵌套。通常，在进入 ISR 时会暂时屏蔽同级或更低优先级的中断，只允许更高优先级的中断请求打断当前 ISR 的执行。

1.3 简答题

1. 什么是刷新存储器？其存储容量与什么因素有关?

   解析： 刷新存储器定义：是存储像素信息用于显示器显示图像的存储器，主要用于帧缓存。

   存储容量相关因素：与显示分辨率和颜色深度有关。分辨率越高，像素点越多；颜色深度越大，每个像素存储信息所需位数越多，都会使存储容量增大

2. 外设的 I/O 控制方式分为哪几类? 各具什么特点?

   解析：

控制方式	特点
程序控制方式	• CPU 直接控制外设, 不需要其他硬件支持 • CPU 一直等待，利用率低, 适用于低速设备
中断驱动方式	• 外设就绪时向 CPU 发中断请求, CPU 可执行其他任务 • 提高 CPU 利用率, 适用于中速设备
直接存储器(DMA)控制方式	• 外设直接与主存交换数据 • 不需要 CPU 干预, 效率最高, 适用于高速设备
通道方式	• 多个设备共享总线, 灵活性高 • 设备间可直接通信, 适用于复杂系统 • 但是硬件和软件复杂,成本高

表 4　外设 I/O 控制方式

3. 什么是指令周期? 什么是机器周期? 什么是时钟周期? 三者有什么关系?

   解析：
   - 指令周期：CPU 取指令并执行的时间总和。
   - 机器周期：CPU 从内存读取一个指令字的最短时间，一个指令周期含若干机器周期。
   - 时钟周期：计算机最基本时间单位，由主频决定。

   三者关系: 一个指令周期含一个或多个机器周期，一个机器周期含若干时钟周期

1.4 综合题

1. 已知 $x=-0.01111, y=+0.11001$, 求 $[x]_{补}, [-x]_{补}, [y]_{补}, [-y]_{补}, x+y=?, x-y=?$

解析：

- 补码表示法：
  1. $x = -0.01111$ 的补码表示为 $[x]_{\{补\}} = 1.10001$
  2. $[-x]_{\{补\}} = 0.01111$
  3. $y = +0.11001$ 的补码表示为 $[y]_{\{补\}} = 0.11001$
  4. $[-y]_{\{补\}} = 1.00111$
- 运算：
  1. $x + y = -0.01111 + 0.11001 = 0.10010$
  2. $x - y = -0.01111 - 0.11001 = -0.10110$
- 补码运算：
  1. $x + y = 0.10010$
  2. $x - y = 1.01010$
- 结果：
  1. $[x]_{\{补\}} = 1.10001$
  2. $[-x]_{\{补\}} = 0.01111$
  3. $[y]_{\{补\}} = 0.11001$
  4. $[-y]_{\{补\}} = 1.00111$
  5. $x + y = 0.10010$
  6. $x - y = 1.01010$

2. 假设机器字长 16 位，主存储容量为 128K 字节，指令长度为 16 位或 32 位，其有 128 条指令，设计计算机指令格式，要求有直接、立即数、相对、基值、间接、变址六种寻址方式。

解析：

1. **引言：指令设计的核心挑战**

   计算机通过执行一系列二进制的机器指令来完成任务。这些指令涵盖了各种基本操作，如算术运算（加法、减法）、数据传输（加载、存储）以及控制流（跳转、分支）。然而，计算机硬件在处理指令时存在固有的限制：
   - 指令必须以二进制形式表示。
   - 处理器一次通常只能读取固定长度的指令（例如，16 位或 32 位）。

   因此，指令设计的核心挑战在于：如何在有限的位数内，高效且清晰地编码所有必要的信息？

   一条典型的机器指令需要传达以下关键信息：
   1. **操作类型**：需要执行什么操作（如：加法、移动数据）？
   2. **操作数来源/目的地**：操作涉及的数据在哪里（如：在寄存器中、在内存的特定地址、或直接包含在指令中）？
   3. **具体地址或数值**：如果涉及内存地址或立即数，其具体值是多少？地址需要多少位来表示？

本解析旨在探讨一个假设的指令系统（基于 128 条指令、128K 字节内存、6 种寻址方式、8 个通用寄存器）的设计过程，阐明其字段分配的逻辑依据，特别是操作码、寻址方式和地址字段的设计。

2. **指令基本构成要素分析**

2.1. **问题一：操作码 (Opcode) 位数确定**

需求：系统需支持 128 条不同的指令。为了唯一标识每条指令，需要为其分配一个独一无二的二进制编码，即操作码（Opcode, OP）。

计算：确定表示 128 种不同状态所需的最小二进制位数。

$$2^n \geq 128$$

当 $n = 7$ 时，$2^7 = 128$。

结论：因此，操作码字段至少需要 **7 位** 才能覆盖所有 128 条指令。

2.2. **问题二：内存地址表示位数确定**

需求：系统内存容量为 128K 字节。需要确定能够寻址到每个字节单元所需的地址位数。

计算：

1. 将内存容量转换为字节数：

$$128 \text{ K 字节} = 128 \times 1024 \text{ 字节} = 131072 \text{ 字节}$$

2. 确定表示 131072 个不同地址所需的最小二进制位数：

$$2^m \geq 131072$$

当 $m = 17$ 时，$2^{17} = 131072$。

结论：因此，需要 **17 位** 地址才能访问整个 128K 字节的内存空间。

2.3. **面临的挑战：指令长度限制**

根据上述分析，仅操作码和内存地址就需要：

$$\text{OP 位数} + \text{地址位数} = 7 + 17 = 24\text{位}$$

这产生了一个明显的问题：如果指令长度被限制为 16 位，则无法在单条指令中同时容纳 7 位的操作码和 17 位的完整内存地址。

3. **<u>解决方案：可变指令长度</u>**

为了解决上述空间限制问题，同时兼顾效率和功能性，通常采用可变指令长度的设计策略。在此假设系统中，指令长度可以为 16 位或 32 位。

- **16 位短指令**：适用于操作相对简单、不涉及完整内存地址或仅需小范围偏移的操作。例如，寄存器间操作、使用立即数的操作、或基于程序计数器的短距离跳转。

- **32 位长指令**：适用于需要访问内存地址或采用更复杂寻址方式的操作。其额外的位数可以容纳完整的地址信息或其他必要的字段。

这种设计允许常用、简单的指令使用较短的编码，提高代码密度和执行效率，而复杂操作则利用更长的格式来提供足够的表达能力。

4. **关键字段设计**

4.1. **问题三：寻址方式 (Addressing Mode, AM) 字段**

需求：系统支持 6 种不同的寻址方式，用以指示操作数的来源或计算有效地址的方法。这些方式可能包括：

1. 立即数寻址 (Immediate)	操作数直接包含在指令中
2. 直接寻址 (Direct)	指令中包含操作数的完整内存地址
3. 间接寻址 (Indirect)	指令中的地址指向另一个包含实际操作数地址的内存位置
4. 相对寻址 (Relative)	以程序计数器为基准，通过偏移量计算操作数地址
5. 变址寻址 (Indexed)	基础地址加上变址寄存器的内容得到操作数地址
6. 基址寻址 (Based)	基址寄存器的内容加上指令中的偏移量得到操作数地址

表 5　寻址方式列表

计算：确定表示 6 种不同寻址方式所需的最小二进制位数。

$$2^k \geq 6$$

当 $k = 3$ 时，$2^3 = 8$，足以表示 6 种方式（还有 2 种编码可留作扩展或他用）。

结论：因此，设计一个 **3 位** 的寻址方式（AM）字段是合适的。

4.2. **问题四：寄存器 (Register, R) 字段**

需求：某些寻址方式（如基址寻址、变址寻址）以及许多操作本身（如寄存器间算术运算）需要指定一个或多个通用寄存器。假设系统拥有 8 个通用寄存器。

计算：确定表示 8 个不同寄存器所需的最小二进制位数。

$$2^r \geq 8$$

当 $r = 3$ 时，$2^3 = 8$。

结论：因此，需要一个 **3 位** 的寄存器（R）字段来指定参与操作或寻址的寄存器。

5. **最终指令格式设计**

基于以上分析，可以提出如下的 16 位和 32 位指令格式：

16 位短指令格式

这种格式适用于不直接涉及内存地址或使用隐含地址/短偏移量的指令。

字段	位数	用途说明
OPCODE	7	指定执行的操作（128 种之一）
AM	3	指定寻址方式（6 种之一）
剩余字段	6	根据具体指令和寻址方式，可用于存放短立即数、寄存器编号、短偏移量等。 $7+3+6=16$ 位

表 6　16 位指令格式示例

32 位长指令格式

这种格式主要用于需要访问内存、使用较长偏移量或涉及寄存器参与地址计算的指令。

字段	位数	用途说明
OPCODE	7	指定执行的操作（128 种之一）
AM	3	指定寻址方式（6 种之一）
R	3	指定参与寻址或操作的通用寄存器（8 个之一）
地址/偏移量/立即数	19	提供内存地址、基址/变址偏移量或较长的立即数。 $7+3+3+19=32$ 位。注意：19 位足以表示 17 位地址，并有余量。

表 7　表 2：32 位指令格式示例

注意：19 位的地址/偏移量字段足够容纳所需的 17 位内存地址，剩余的位数可以用于其他目的或保持未使用，或者允许更大的偏移量范围。

6. **示例说明**

    1. **示例 1：`ADD R1, 5`**
        - **含义**：将寄存器 R1 的内容与立即数 5 相加，结果存回 R1。
        - **分析**：这是一个典型的立即数寻址操作，不涉及内存访问。
        - **适用格式**：很可能使用 **16 位短指令** 格式。
            - OPCODE: `ADD` 指令的 7 位编码。
            - AM: 立即数寻址方式的 3 位编码。
            - 剩余 6 位：可能一部分用于指定目标寄存器 R1（如果寄存器字段不固定在此处），另一部分用于表示立即数 5（如果 6 位足够）。具体分配取决于详细设计。

    1. **示例 2：`MOV R2, [0x2F1A]`**
        - **含义**：将内存地址 `0x2F1A` 处的数据加载到寄存器 R2 中。

- **分析**：这是一个直接寻址操作，需要指定一个内存地址。地址 `0x2F1A` (十六进制) 转换为二进制需要 14 位 ($2^{\{13\}} < 0\text{x}2\text{F}1\text{A} < 2^{\{14\}}$)，在 17 位地址空间范围内。
- **适用格式**：由于需要包含内存地址，必须使用 **32 位长指令** 格式。
  - OPCODE: `MOV` (内存到寄存器) 指令的 7 位编码。
  - AM: 直接寻址方式的 3 位编码。
  - R: 目标寄存器 R2 的 3 位编码。
  - 地址/偏移量: 19 位字段用于存放 17 位的地址 `0x02F1A` (高位补零)。

7. **So**

   计算机指令格式的设计是一个在功能需求、性能效率和硬件限制之间进行权衡的过程。通过分析操作码、内存地址、寻址方式和寄存器等要素所需的位数，并采用可变长度指令（如 16 位和 32 位）的策略，可以设计出既能满足复杂操作需求，又能高效执行简单常用操作的指令集体系结构。上述格式划分清晰地展示了如何在有限的比特位中，结构化地编码指令执行所需的各种信息。

---

3. 某机字长 32 位，常规设计的存储空间 ≤ 32M，若将存储空间扩至 256M，请提出一种可能方案.

   解析： 可以采用多体交叉存取方案, 即将主存分为 8 个相互独立,容量相同的模块 $M_0, M_1, M_2, ...M_7$,每个模块 32M × 32 位. 它各自具有一套寄存器,数据缓冲器, 各自以同等的方式与 CPU 传递信息, 其组成结构如图 6.

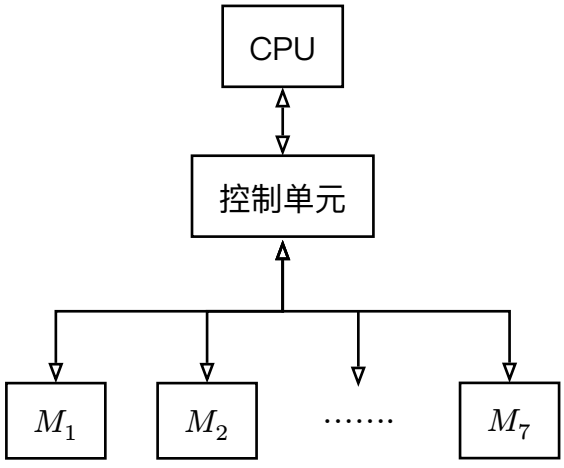


图 6　多体交叉存取存储器示意图

   CPU 访问 8 个存贮模块，可采用两种方式：一种是在一个存取周期内，同时访问 8 个存贮模块，由存贮器控制它们分时使用总线进行信息传递。另一种方式是：在存取周期内分时访问每个体，即经过 1/8 存取周期就访问一个模块。这样，对每个模块而言，从 CPU 给出访存操作命令直到读出信息，仍然是一个存取周期时间。而对 CPU 来说，它可以在一个存取周期内连续访问 8 个存贮体，各体的读写过程将重叠进行。

4. (1)

指令存储器 IM = 18 位	数据存储器 DM = 16 位
PC = 14 位	I R = 18 位
AC_0 = AC_1 = 16 位	R_0 ~ R_3 = 16 位
IAR = 14 位	I D R = 18 位
DAR = 16 位	DDR = 16 位

(2) 加法指令“ADD X(R_i)”是一条隐含指令，其中一个操作数来自 AC_0。另一个操作数在数

据存储器中，其地址由通用寄存器的内容(R_i)加上指令格式中的 X 量值决定，可认为这是一种变址寻址。指令周期的操作流程如图 10–4–3 所示。

---

1. 假设某磁盘，每面有 220 道：已知磁盘转速 r=3000 转/分。数据传输率为 175000B/s。求该磁盘总容量.

   解析：

   - 磁盘转速 = 3000 转/分 = 50 转/秒
   - 所以转说 50 圈传了 175000B 的数据, 那么一圈的数据量为 $\frac{175000B}{50} = 3500B$
   - 每面有 220 道, 所以每道的数据量为 $3500B \times 220 = 1540000B$