计算机组成原理题库分类整理

目录

1 数据的编码表示

本部分重点

- 计算机采用二进制表示。
- 原码、补码的表示，表示范围。
- 规格化浮点数的表示范围与精度取决于什么。
- IEEE754 单精度浮点格式，真值与编码转换。
- 奇偶校验。

1.1 计算机采用二进制表示

题目 1.1 冯 · 诺依曼结构计算机中的数据采用二进制编码表示，其主要原因是 ()。

I. 二进制的运算规则简单

II. 制造两个稳态的物理器件较容易

III. 便于用逻辑门电路实现算术运算

A. 仅 I、II

B. 仅 I、III

C. 仅 II、III

D. I、II 和 III

答案： D

解析： 计算机采用二进制表示数据的主要原因包括：

二进制运算规则简单

二进制只需要表示0 和1，容易用两个稳定状态的物理器件实现

二进制便于用逻辑门电路实现逻辑运算和算术运算

因此 I、II、III 均正确，答案为 D。

题目 1.2 以下是有关冯 · 诺依曼结构计算机中指令和数据表示形式的叙述，其中正确的是 ()。

A. 指令和数据可以从形式上加以区分

B. 指令以二进制形式存放，数据以十进制形式存放

C. 指令和数据都以二进制形式存放

D. 指令和数据都以十进制形式存放

答案： C

解析： 冯 · 诺依曼结构计算机采用存储程序思想，程序指令和数据都存放在存储器中，并且都采用二进制形式表示。

从存储器中存放的形式来看，指令和数据本质上都是二进制代码，不能仅凭形式直接区分，需要结合取指和执行过程来判断某个二进制代码是指令还是数据。

因此答案为 C。

1.2 原码、补码的表示与表示范围

知识点：

原码表示中，最高位为符号位，0 表示正数，1 表示负数，其余位表示数值大小。

对于 n 位原码，其表示范围为：

$$-(2^{n-1}-1) \sim +(2^{n-1}-1)$$

补码表示中，正数补码与原码相同，负数补码等于其原码除符号位外按位取反再加 1。

对于 n 位补码，其表示范围为：

$$-2^{n-1} \sim +(2^{n-1}-1)$$

例如 8 位补码的表示范围为：

$$-128 \sim +127$$

题目 1.3 若 $[X]_{补} = 1.1101010$，则 $[X]_{原} =$ ()。

A. 1.0010101

B. 1.0010110

C. 0.0010110

D. 0.1101010

答案： B

解析： 已知：

$$[X]_{补} = 1.1101010$$

符号位为 1，说明 X 为负数。

负数由补码求原码时，符号位保持 1 不变，数值位按位取反加 1。

数值位为：

$$1101010$$

先取反：

$$0010101$$

再加 1：

$$0010101 + 1 = 0010110$$

所以：

$$[X]_{原} = 1.0010110$$

因此答案为 B。

题目 1.4 在定点小数中，采用 1 位符号位，若寄存器内容为 10000000，当它分别表示为原码、补码和反码时，其对应的真值分别为_____、_____ 和_____。（均用十进制表示）

答案： -0；-1；$-\dfrac{127}{128}$

解析： 寄存器内容为：

$$10000000$$

采用 1 位符号位，说明最高位为符号位，其余 7 位为小数数值位。

若表示为原码：

$$1.0000000$$

符号位为 1，表示负数，数值位全为 0，所以真值为：

$$-0$$

若表示为补码：

对于 8 位定点小数补码，10000000 表示最小值：

$$-1$$

若表示为反码：

反码为：

$$1.0000000$$

符号位为 1，说明是负数。负数反码的数值位是其真值绝对值数值位按位取反得到的。

因此数值位取反：

$$0000000 \to 1111111$$

所以其绝对值为：

$$0.1111111_2$$

$$= \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128}$$

$$= \frac{127}{128}$$

所以真值为：

$$-\frac{127}{128}$$

因此三个真值分别为：

$$-0, \quad -1, \quad -\frac{127}{128}$$

题目 1.5 在下列有关补码和移码，设偏置常数为 2^{n-1}，关系的叙述中，错误的是 ()。

A. 零的补码和移码表示相同

B. 相同位数的补码和移码表示具有相同的表示范围

C. 同一个数的补码和移码表示，符号位相反，其余部分相同

D. 一般用移码表示浮点数的阶，而用补码表示定点整数

答案： A

解析： 移码是在真值的基础上加上一个偏置常数得到的编码。若偏置常数为 2^{n-1}，则移码和补码的关系是：

同一个数的移码与补码符号位相反，其余位相同

零的补码为：

$$000\cdots0$$

而零的移码为：

$$100\cdots0$$

二者并不相同，因此 A 错误。

相同位数的补码和移码表示范围相同，只是编码形式不同，所以 B 正确。

同一个数的补码和移码通常符号位相反，其余位相同，所以 C 正确。

在计算机中，一般用移码表示浮点数的阶码，用补码表示定点整数，所以 D 正确。

因此错误的是 A。

题目 1.6 已知 $[x]_{补} = x_0.x_1x_2\cdots x_n$，则 $[-x]_{补}$ =_____。

答案：

$$[-x]_{补} = \overline{x_0}.\overline{x_1x_2}\cdots\overline{x_n} + 2^{-n}$$

解析： 若定点小数的补码为：

$$[x]_{补} = x_0.x_1x_2\cdots x_n$$

其中 x_0 为符号位，小数部分共有 n 位。

补码求相反数的方法是：连同符号位按位取反，末位加 1。

由于小数部分最低位 x_n 的权值为：

$$2^{-n}$$

所以末位加 1 等价于加上:

$$2^{-n}$$

因此:

$$[-x]_{补} = \overline{x_0}.\overline{x_1 x_2}\cdots\overline{x_n} + 2^{-n}$$

1.3 规格化浮点数的表示范围与精度

题目 1.7 假定两种浮点数格式的位数都是 32 位，但格式 1 的阶码长、尾数短，格式 2 的阶码短、尾数长，其它所有规定都相同。则它们可表示的数的精度和范围是 ()。

A. 格式 1 可表示的数的范围更小，但精度更高

B. 格式 2 可表示的数的范围更小，但精度更高

C. 格式 1 可表示的数的范围更大，且精度更高

D. 两者可表示的数的精度和范围均相同

答案: B

解析: 浮点数中:

阶码位数决定表示范围

尾数位数决定表示精度

格式 1 的阶码长、尾数短，因此可表示的数的范围更大，但精度较低。

格式 2 的阶码短、尾数长，因此可表示的数的范围更小，但精度更高。

所以答案为 B。

23 年北科大考研题中以填空题形式出现问阶码决定什么, 位数决定什么

题目 1.8 若浮点数用移码表示阶码，补码表示尾数，则判断运算结果是否为规格化数的方法是 ()。

A. 阶符与数符相同为规格化数

B. 阶符与数符相异为规格化数

C. 数符与尾数小数点后第一位数字相同为规格化数

D. 数符与尾数小数点后第一位数字相异为规格化数

答案: D

解析: 当浮点数的尾数采用补码表示时，判断是否为规格化数，主要看尾数符号位与小数点后第一位是否相异。

若尾数为正数，则数符为 0，规格化尾数应形如：

$$0.1xxx\cdots$$

若尾数为负数，则数符为 1，规格化尾数应形如：

$$1.0xxx\cdots$$

因此，补码尾数规格化的判断方法是：

数符与尾数小数点后第一位数字相异

所以答案为 D。

在 2022 年北科大考研题中，该知识点曾以填空题形式出现。

1.4 IEEE754 单精度浮点格式：真值与编码

知识点：

IEEE754 单精度浮点数长度为 32 位，其格式为：

符号位S(1 位)　阶码E(8 位)　尾数M(23 位)

规格化数的真值为：

$$(-1)^S \times 1.M \times 2^{E-127}$$

其中 127 是单精度浮点数的阶码偏置值。

题目 1.9 `float` 型数据通常用 IEEE754 单精度格式表示。若编译器将 `float` 型变量 X 分配在一个 32 位浮点寄存器 FR1 中，且 $X = -8.25$，则 FR1 的内容是 ()。

A. C104 0000H

B. C242 0000H

C. C184 0000H

D. C1C2 0000H

答案： A

解析： 首先将 8.25 转换为二进制：

$$8.25 = 1000.01_2$$

规格化表示为：

$$1000.01_2 = 1.00001_2 \times 2^3$$

因为 $X = -8.25$，所以符号位：

$$S = 1$$

阶码为：

$$E = 3 + 127 = 130$$

130 的二进制为：

$$130 = 10000010_2$$

尾数部分取规格化后小数点后的部分：

$$M = 00001000000000000000000$$

所以 IEEE754 单精度编码为：

$$1\ 10000010\ 00001000000000000000000$$

按 4 位一组转换为十六进制：

$$1100\ 0001\ 0000\ 0100\ 0000\ 0000\ 0000\ 0000$$

即：

$$\text{C104 0000H}$$

因此答案为 A。

题目 1.10 假定某数采用 IEEE754 单精度浮点数格式表示为 $4C000000H$，求该数的十进制真值，要求写明计算过程。

答案： 33554432

解析： 先将十六进制数 $4C000000H$ 转换为二进制：

$$4C000000H = 0100\ 1100\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000$$

IEEE754 单精度浮点数格式为：

$$S(1\text{位}) \quad E(8\text{位}) \quad M(23\text{位})$$

因此：

$$S = 0$$

表示该数为正数。

阶码字段为：

$$E = 10011000_2 = 152$$

IEEE754 单精度浮点数的偏置常数为 127，所以阶码真值为：

$$e = 152 - 127 = 25$$

尾数字段全为 0，所以规格化尾数为：

$$1.0$$

因此该浮点数的真值为：

$$(-1)^0 \times 1.0 \times 2^{25}$$

$$= 33554432$$

所以该数的十进制真值为 33554432。

题目 1.11 浮点数 x 按 IEEE754 执行标准，则 $C136\ 0000H$ 的对应十进制数是_____。

答案： -11.375

解析： 先将十六进制数 $C136\ 0000H$ 转换为二进制：

$$C136\ 0000H = 1100\ 0001\ 0011\ 0110\ 0000\ 0000\ 0000\ 0000$$

IEEE754 单精度浮点数格式为：

$$S(1位) \quad E(8位) \quad M(23位)$$

因此符号位为：

$$S = 1$$

表示该数为负数。

阶码字段为：

$$E = 10000010_2 = 130$$

单精度浮点数的偏置常数为 127，所以阶码真值为：

$$e = 130 - 127 = 3$$

尾数字段为：

$$M = 01101100000000000000000$$

因此规格化尾数为：

$$1.M = 1.011011_2$$

将其转换为十进制：

$$1.011011_2 = 1 + \frac{1}{4} + \frac{1}{8} + \frac{1}{32} + \frac{1}{64}$$

$$= 1.421875$$

所以该浮点数真值为：

$$(-1)^1 \times 1.421875 \times 2^3$$

$$= -11.375$$

因此答案为 -11.375。

1.5 奇偶校验

知识点：

奇偶校验用于检测数据传输或存储过程中是否发生错误。

奇校验要求整个编码中 1 的个数为奇数。

偶校验要求整个编码中 1 的个数为偶数。

例如：

0110 0100

其中 1 的个数为 3，是奇数，因此满足奇校验。

题目 1.12 假定下列字符编码中含有 1 位奇偶校验位，但没有发生数据错误，那么采用奇校验的字符编码是 ()。

A. 0101 0011

B. 0110 0100

C. 1011 0100

D. 0011 0101

答案： B

解析： 奇校验要求整个编码中 1 的个数为奇数。

逐项统计：

0101 0011

其中 1 的个数为 4，为偶数，不满足奇校验。

0110 0100

其中 1 的个数为 3，为奇数，满足奇校验。

1011 0100

其中 1 的个数为 4，为偶数，不满足奇校验。

0011 0101

其中 1 的个数为 4，为偶数，不满足奇校验。
因此答案为 B。

2 指令系统

本部分重点

- 指令集体系结构 ISA、指令系统、指令集的含义。
- 定长指令，字段的构成，最大寻址空间等。
- 常见寻址方式：立即数、寄存器直接寻址、寄存器间接寻址、存储器直接寻址、存储器间接寻址、变址寻址、相对寻址等。
- 已知形式地址，如何计算操作数的有效地址。
- 综合题：指令格式与寻址方式，如 MIPS 指令系统。

2.1 指令集体系结构 ISA、指令系统与指令集

知识点：

指令集体系结构 ISA 是软件和硬件之间的接口，规定了计算机能够执行哪些指令，以及这些指令的格式、功能和使用方式。

指令系统是某台计算机所能执行的全部指令的集合。

指令集通常包括：

- 算术逻辑指令；
- 数据传送指令；
- 程序控制指令；
- 输入输出指令；
- 特权指令等。

题目 2.1 下面有关指令集体系结构的说法中，错误的是（）。

A. 指令集体系结构位于计算机软件和硬件的交界面上
B. 指令集体系结构是指低级语言程序员所看到的概念结构和功能特性
C. 指令集体系结构的英文缩写是 ISA
D. 用户可见寄存器的长度、功能与编号不属于指令集体系结构

答案： D

解析： 指令集体系结构的英文缩写是 ISA，即 Instruction Set Architecture。

ISA 是软件和硬件之间的接口，规定了低级语言程序员可见的计算机功能和结构。

ISA 通常包括：

- 指令格式；

- 指令种类;
- 寻址方式;
- 数据类型;
- 用户可见寄存器的组织;
- 存储器地址空间等。

用户可见寄存器的长度、功能和编号属于 ISA 的内容。
因此，D 说法错误。

题目 2.2 指令系统采用定长操作码，一共有 30 种不同的操作指令，则操作码至少需要_____ 位二进制数。
答案： 5
解析： 若操作码有 n 位，则最多可以表示：

$$2^n$$

种不同操作。
题目中共有 30 种不同操作指令，因此需要满足：

$$2^n \geq 30$$

因为：

$$2^4 = 16 < 30$$

$$2^5 = 32 \geq 30$$

所以操作码至少需要 5 位二进制数。
因此答案为：5。

题目 2.3 下列说法中_____ 是正确的。
A. 加法指令的执行周期一定要访存
B. 加法指令的执行周期一定不访存
C. 指令的地址码给出存储器地址的加法指令，在执行周期一定访存
D. 指令的地址码给出存储器地址的加法指令，在执行周期不一定访存
答案： C
解析： 加法指令在执行周期是否访存，取决于操作数所在的位置。
如果加法指令的操作数都在寄存器中，则执行周期不需要访问主存；如果加法指令的地址码给出的是存储器地址，则需要根据该地址访问主存，取出操作数后再进行加法运算。
因此，不能笼统地说加法指令执行周期一定访存或一定不访存。
当指令的地址码给出存储器地址时，该加法指令在执行周期一定需要访存。
因此答案为 C。

2.2 定长指令与字段构成

知识点：

定长指令是指每条指令的长度固定，例如 MIPS 指令通常为 32 位定长指令。

一条指令通常由多个字段组成，常见字段包括：

- 操作码字段：指出指令执行什么操作；
- 寄存器字段：指出源寄存器或目的寄存器；
- 地址字段：指出操作数地址或跳转目标地址；
- 立即数字段：直接给出操作数；
- 功能码字段：进一步确定具体操作。

若操作码字段有 n 位，则最多可以表示：

$$2^n$$

种不同操作。

若地址字段有 n 位，且按字节编址，则最大寻址空间为：

$$2^n \text{ Byte}$$

题目 2.4 某计算机按字节编址，指令字长固定且只有两种指令格式，其中三地址指令 29 条，二地址指令 107 条，每个地址字段为 6 位，则指令字长至少应该是 ()。

A. 24 位

B. 26 位

C. 28 位

D. 32 位

答案： A

解析： 设指令字长为 L 位。

三地址指令有 3 个地址字段，每个地址字段为 6 位，因此地址字段共占：

$$3 \times 6 = 18$$

位。

三地址指令有 29 条，所以操作码字段至少需要满足：

$$2^n \geq 29$$

因为：

$$2^4 = 16 < 29, \quad 2^5 = 32 \geq 29$$

所以三地址指令操作码至少需要 5 位。

因此指令字长至少需要：

$$18 + 5 = 23$$

位。

由于计算机按字节编址，指令字长通常应取字节的整数倍，所以至少应取：

$$24$$

位。

因此答案为 A。

2.3 常见寻址方式与有效地址计算

知识点：

寻址方式用于说明如何根据指令中的地址信息找到操作数。

常见寻址方式如下：

- 立即寻址：操作数直接在指令中给出。

$$\text{操作数} = A$$

- 寄存器直接寻址：操作数在寄存器中。

$$\text{操作数} = R_i$$

- 寄存器间接寻址：寄存器中存放操作数的有效地址。

$$EA = (R_i)$$

- 存储器直接寻址：指令地址字段直接给出有效地址。

$$EA = A$$

- 存储器间接寻址：指令地址字段给出一个存储单元地址，该单元内容才是有效地址。

$$EA = (A)$$

- 变址寻址：形式地址与变址寄存器内容相加得到有效地址。

$$EA = A + (R_x)$$

- 相对寻址：程序计数器 PC 的内容与形式地址相加得到有效地址。

$$EA = (PC) + A$$

题目 2.5 寄存器间接寻址方式中,操作数存放在_____,寄存器中存放的是_____。

答案: 主存单元中；操作数的有效地址

解析: 寄存器间接寻址不是把操作数直接放在寄存器中，而是寄存器中保存操作数所在主存单元的地址。

因此，需要根据寄存器中的地址访问主存，才能取得操作数。

寄存器间接寻址的有效地址为:

$$EA = (R_i)$$

其中，R_i 中存放的是操作数的有效地址，操作数本身存放在主存单元中。

所以答案为:主存单元中；操作数的有效地址。

题目 2.6 () 方式对实现程序浮动提供了支持。

A. 变址寻址

B. 相对寻址

C. 间接寻址

D. 寄存器间接寻址

答案: B

解析: 相对寻址方式中，操作数或转移目标的有效地址由程序计数器 PC 的内容加上位移量得到:

$$EA = (PC) + A$$

其中 A 为指令中给出的形式地址或偏移量。

由于有效地址是相对于当前 PC 计算得到的，因此程序整体装入到主存的不同位置时，只要指令之间的相对位置不变，程序仍然可以正确执行。

所以，相对寻址有利于程序浮动和位置无关代码的实现。

因此答案为 B。

题目 2.7 设指令字长等于存储字长，均为 24 位，若某指令系统可完成 108 种操作，操作码长度固定，且具有直接、间接、变址、基址、相对、立即等寻址方式，则在保证最大范围内直接寻址的前提下，指令字中操作码占_____ 位，寻址特征位占_____ 位，可直接寻址的范围是_____，一次间址的范围是_____。

答案: 7；3；2^{14} 个存储字；2^{24} 个存储字

解析: 该指令系统可完成 108 种操作，且操作码长度固定。设操作码占 n 位，则应满足:

$$2^n \geq 108$$

因为:

$$2^6 = 64 < 108, \quad 2^7 = 128 \geq 108$$

所以操作码至少需要:

位。

题目中共有直接、间接、变址、基址、相对、立即 6 种寻址方式。设寻址特征位占 m 位，则应满足：

$$2^m \geq 6$$

因为：

$$2^2 = 4 < 6, \quad 2^3 = 8 \geq 6$$

所以寻址特征位至少需要：

$$3$$

位。

指令字长为 24 位，在保证最大范围内直接寻址的前提下，剩余位数都作为地址码字段，因此地址码字段位数为：

$$24 - 7 - 3 = 14$$

所以直接寻址范围为：

$$2^{14}$$

个存储字。

由于存储字长为 24 位，一次间接寻址时，地址字段先指出一个存储单元，该存储单元中的 24 位内容可作为有效地址，因此一次间址的范围为：

$$2^{24}$$

个存储字。

因此答案为：

$$7, \quad 3, \quad 2^{14} \text{ 个存储字}, \quad 2^{24} \text{ 个存储字}$$

题目 2.8 假定采用相对寻址方式的转移指令占两个字节，第一字节是操作码，第二字节是相对位移量，位移量用补码表示。取指令时，每次 CPU 从存储器取出一个字节，自动完成 $(PC) + 1 \rightarrow PC$。假设执行到某转移指令时，取指令前 PC 的内容是 $200CH$，该指令的转移目标地址为 $1FFEH$，求该转移指令第二字节的内容，要求用十六进制形式表示，并写明计算过程。

答案： $F0H$

解析： 该转移指令占两个字节，取指令前：

$$(PC) = 200CH$$

取第一字节后：

$$PC = 200DH$$

取第二字节后：

$$PC = 200EH$$

相对寻址方式中，转移目标地址由取完指令后的 PC 值加上相对位移量得到：

$$目标地址 = (PC) + 位移量$$

已知目标地址为：

$$1FFEH$$

所以位移量为：

$$1FFEH - 200EH = -10H$$

第二字节位移量用补码表示，$10H$ 的 8 位二进制为：

$$0001\ 0000$$

因此 $-10H$ 的 8 位补码为：

$$1111\ 0000$$

转换为十六进制：

$$1111\ 0000 = F0H$$

所以该转移指令第二字节的内容为：

$$F0H$$

题目 2.9 指令系统采用不同寻址方式的目的是 ()。

A. 实现存贮程序和程序控制

B. 缩短指令长度，扩大寻址空间，提高编程灵活性

C. 可直接访问外存

D. 提供扩展操作码的可能并降低指令译码的难度

答案： B

解析： 寻址方式用于说明如何根据指令中的地址信息找到操作数。

指令系统采用多种寻址方式，可以根据不同场景灵活表示操作数地址，在缩短指令地址字段长度的同时扩大寻址能力，并提高程序设计的灵活性。

因此答案为 B。

题目 2.10 假设某指令的一个操作数采用变址寻址方式，变址寄存器中的值为 126，指令中给出的形式地址为 C000H，地址 C000H 中的内容为 B000H，则该操作数的有效地址为 ()。

A. B126H

B. C126H

C. B07EH

D. C07EH

答案： D

解析： 变址寻址方式中，有效地址由形式地址与变址寄存器内容相加得到：

$$EA = A + (R_x)$$

题中形式地址为：

$$A = C000H$$

变址寄存器中的值为：

$$126_{10} = 7EH$$

因此有效地址为：

$$EA = C000H + 007EH = C07EH$$

题中地址 C000H 中的内容 B000H 是存储器间接寻址时才会用到的信息，本题采用的是变址寻址，因此不需要使用 B000H。

所以答案为 D。

2.4 MIPS 指令格式与寻址方式综合题

知识点：

MIPS 指令系统常见三种指令格式：

R 型指令：

$$op \mid rs \mid rt \mid rd \mid shamt \mid funct$$

常用于寄存器之间的算术逻辑运算，例如：

$$add\ \$t0,\ \$t1,\ \$t2$$

I 型指令：

$$op \mid rs \mid rt \mid immediate$$

常用于立即数运算、访存指令和条件分支指令，例如：

$$addi,\ lw,\ sw,\ beq$$

J 型指令：

$$op \mid address$$

常用于无条件跳转，例如：

$$j,\ jal$$

题目 2.11 指令长度 16 位，操作码 $OP[15:10]$，寄存器编号 $R[9:8]$，寻址方式 $MOD[7:6]$，形式地址 $A[5:0]$。假定要执行的指令为加法指令，存放在 $1000H$ 单元中，PC 中取指阶段后自增，形式地址 A 的编码为 $02H$，主存按字编址。其中加法指令两个操作数，一个来自形式地址 A 或者主存，另一个来自目的寄存器 R_0，运算结果放 R_0。

MOD	寻址方式	A
00	立即寻址	A 为立即数
01	变址寻址	A 为偏移量
10	相对寻址	A 为偏移量

存储单元地址	存储单元内容	寄存器内容
$1000H$	$1000H$	$R_x = 2000H$
$1002H$	$1100H$	
$\cdots$	$\cdots$	$R_0 = 0100H$
$2001H$	$2000H$	
$2002H$	$3000H$	

求：

（1）$MOD = 00$，$(R_0) =$_____

（2）$MOD = 01$，$(R_0) =$_____

（3）$MOD = 10$，$(R_0) =$_____，$(PC) =$_____

答案： （1）$0102H$；（2）$3100H$；（3）$1200H$，$1002H$

解析： 已知目的寄存器初值为：

$$(R_0) = 0100H$$

形式地址为：

$$A = 02H$$

（1）当 $MOD = 00$ 时，为立即寻址，A 本身就是立即数，因此源操作数为：

$$02H$$

执行加法：

$$(R_0) \leftarrow (R_0) + 02H = 0100H + 02H = 0102H$$

所以：

$$(R_0) = 0102H$$

（2）当 $MOD = 01$ 时，为变址寻址，A 为偏移量，变址寄存器内容为：

$$(R_x) = 2000H$$

有效地址为：

$$EA = (R_x) + A = 2000H + 02H = 2002H$$

由表可知：

$$M[2002H] = 3000H$$

执行加法：

$$(R_0) \leftarrow (R_0) + M[2002H] = 0100H + 3000H = 3100H$$

所以：

$$(R_0) = 3100H$$

（3）当 $MOD = 10$ 时，为相对寻址，A 为偏移量。指令存放在 $1000H$ 单元中，取指后 PC 自增，故：

$$(PC) = 1002H$$

根据题表给出的相对偏移地址，可访问：

$$1000H + 02H = 1002H$$

由表可知：

$$M[1002H] = 1100H$$

执行加法：

$$(R_0) \leftarrow (R_0) + M[1002H] = 0100H + 1100H = 1200H$$

所以：

$$(R_0) = 1200H, \quad (PC) = 1002H$$

题目 2.12

(2026 北科大期末考试题) 假定计算机 M 字长为 16 位，按字节编址，连接 CPU 和主存的系统总线中地址线为 20 位，数据线为 8 位，采用 16 位定长指令字，指令格式及其说明如下：

格式	6 位	2 位	2 位	2 位	4 位
R 型	000000	rs	rt	rd	$op1$

其功能为：

$$R[rd] \leftarrow R[rs]\ op1\ R[rt]$$

格式	6 位	2 位	2 位	6 位
I 型	$op2$	rs	rt	imm

I 型指令包括 ALU 运算、条件转移和访存操作三种指令。

格式	6 位	10 位
J 型	$op3$	$target$

J 型指令功能为:

$$PC \leftarrow pc + 2 + signext(target << 1)$$

其中，$op1 \sim op3$ 为操作码，rs、rt 和 rd 为通用寄存器编号，$R[r]$ 表示寄存器 r 的内容，imm 为立即数，$target$ 为转移目标的形式地址。请回答下列问题:

1. R 型格式最多可定义多少种操作? I 型和 J 型格式总共最多可定义多少种操作? 通用寄存器最多有多少个?
2. J 形指令的基地址是什么,target 的范围是什么,target 给出的是偏移的指令条数还是偏移的物理块数.
3. 假定 $op1$ 为 0010 和 0011 时，分别表示带符号整数减法和带符号整数乘法指令，则指令 $01B2H$ 的功能是什么?

3 运算器

本部分重点

- 补码运算简化运算，简化运算器的设计。
- 如何实现补码加减法与溢出判断，计算溢出标志。
- 零扩展、符号扩展。
- 多功能 ALU 的组成。

3.1 补码运算简化运算器设计

知识点：

补码的主要优点是可以将减法转换为加法，从而简化运算器的设计。

在补码系统中：

$$A - B = A + (-B)$$

因此，加法和减法可以统一由加法器完成，不需要单独设计减法器。

补码还可以统一符号位和数值位的处理，使符号位也能参与运算。

题目 3.1 为运算器构造的简单性，运算方法中常采用_____ 加减法、_____ 乘除法。

答案： 补码加减法；补码乘除法

解析： 补码运算可以把减法转化为加法：

$$A - B = A + (-B)$$

因此加法和减法可以统一用加法器实现，不需要单独设计减法器，从而简化运算器结构。

同时，补码表示中符号位和数值位可以统一参加运算，便于处理带符号数。

乘除法中也常采用补码相关算法，以便统一处理正数和负数。

所以答案为：补码加减法；补码乘除法。

3.2 补码加减法与溢出判断

知识点：

补码加法直接按二进制相加，最高位产生的进位丢弃。

补码减法可以转换为补码加法：

$$A - B = A + [-B]_{补}$$

溢出只可能发生在两个同号数相加时：

$$正数 + 正数 = 负数 \quad 溢出$$

$$负数 + 负数 = 正数 \quad 溢出$$

异号数相加不会发生溢出。

也可以根据符号位进位判断溢出：

$$OF = C_n \oplus C_{n-1}$$

其中 C_n 是符号位产生的进位，C_{n-1} 是进入符号位的进位。二者不同，则发生溢出。

题目 3.2 某计算机字长为 8 位，其 CPU 中有一个 8 位加法器。x 和 y 是两个带符号整数变量，用补码表示，$x = 68$，$y = 36$。现要在该加法器中完成 $x - y$ 的运算。

1. 该加法器输入的两个加数和低位进位分别是什么？用二进制形式表示。
2. 若加法器中实现了溢出判断，则 $x - y$ 的机器数及相应的溢出标志 OF 分别是什么？用二进制形式表示。

答案：

1. 两个加数分别为 01000100 和 11011011，低位进位为 1。
2. $x - y$ 的机器数为 00100000，溢出标志 $OF = 0$。

解析： 因为 $x = 68$，$y = 36$，字长为 8 位，所以：

$$[x]_{补} = 01000100$$

$$[y]_{补} = 00100100$$

要计算：

$$x - y$$

可以转化为：

$$x - y = x + (-y)$$

在加法器中实现减法时，通常将减数 y 按位取反，并使低位进位 $C_0 = 1$，即：

$$-y = \overline{y} + 1$$

所以加法器的两个输入加数为：

$$01000100$$

$$\overline{00100100} = 11011011$$

低位进位为：

$$C_0 = 1$$

于是：

$$\begin{array}{r} 01000100 \\ +11011011 \\ +00000001 \\ \hline 00100000 \end{array}$$

最高位产生的进位舍去，结果为：

$$00100000$$

即十进制的 32。
由于 $x = 68$，$y = 36$，实际结果为：

$$68 - 36 = 32$$

32 在 8 位补码表示范围 $[-128, 127]$ 内，因此没有溢出。
所以溢出标志为：

$$OF = 0$$

最终结果为：

$$x - y = 00100000, \quad OF = 0$$

题目 3.3 假定有符号整数采用补码表示,若 `int` 型变量 x 和 y 的机器数分别是 $FFFF\ FFDFH$ 和 $0000\ 0041H$，则 x、y 的值及 $x-y$ 的机器数分别是 ()。

A. $x=-65$, $y=41$，$x-y$ 的机器数溢出

B. $x=-33$, $y=65$，$x-y$ 的机器数为 $FFFF\ FF9DH$

C. $x=-33$, $y=65$，$x-y$ 的机器数为 $FFFF\ FF9EH$

D. $x=-65$, $y=41$，$x-y$ 的机器数为 $FFFF\ FF96H$

答案： C

解析： `int` 型变量通常为 32 位，有符号整数采用补码表示。

首先分析 x:

$$x = FFFF\ FFDFH$$

最高位为 1，说明 x 是负数。求其真值时，对补码取反加 1:

$$FFFF\ FFDFH \xrightarrow{取反} 0000\ 0020H$$

$$0000\ 0020H + 1 = 0000\ 0021H = 33$$

所以:

$$x = -33$$

再分析 y:

$$y = 0000\ 0041H$$

最高位为 0，说明 y 是正数:

$$0000\ 0041H = 65$$

所以:

$$y = 65$$

因此:

$$x - y = -33 - 65 = -98$$

将 98 写成 32 位十六进制:

$$98 = 0000\ 0062H$$

求 -98 的补码:

$$0000\ 0062H \xrightarrow{\text{取反}} FFFF\ FF9DH$$

$$FFFF\ FF9DH + 1 = FFFF\ FF9EH$$

所以：

$$x - y = FFFF\ FF9EH$$

-98 在 32 位补码表示范围内，没有溢出。

因此答案为 C。

题目 3.4 定点补码加减法运算溢出判断的三种方法是什么？分别写出逻辑表达式并加以说明。

答案： 符号位判断法、双符号位判断法、进位判断法。

解析： 设被加数、加数和结果的符号位分别为 x_s、y_s、s_s。

(1) 符号位判断法

两个同号数相加，若结果变号，则发生溢出。

正数加正数，结果为负数时溢出：

$$\overline{x_s}\ \overline{y_s}\ s_s$$

负数加负数，结果为正数时溢出：

$$x_s y_s \overline{s_s}$$

所以加法溢出逻辑表达式为：

$$OF = \overline{x_s}\ \overline{y_s}\ s_s + x_s y_s \overline{s_s}$$

减法可以看作加上相反数，即：

$$x - y = x + (-y)$$

因此减法溢出逻辑表达式可写为：

$$OF = \overline{x_s} y_s s_s + x_s \overline{y_s s_s}$$

即被减数和减数异号，且结果符号与被减数符号不同，则发生溢出。

(2) 双符号位判断法

双符号位判断法采用两个符号位表示数的符号。

运算结果的双符号位为：

00

或

$$11$$

表示没有溢出。
若运算结果的双符号位为:

$$01$$

或

$$10$$

则表示发生溢出。
其中:

$$01$$

通常表示正溢出,

$$10$$

通常表示负溢出。

(3) 进位判断法

设符号位产生的进位为 C_s，最高数值位向符号位产生的进位为 C_{s-1}。
若二者相同，则没有溢出；若二者不同，则发生溢出。
因此溢出标志为:

$$OF = C_s \oplus C_{s-1}$$

其中 $\oplus$ 表示异或运算。

题目 3.5 某 8 位计算机中，假定 x 和 y 是两个带符号整数变量，用补码表示，$x = +63$，$y = 31$，则 $x - y$ 的机器数及其相应的溢出标志 OF 分别是 ()。

A. 0010 0000, $OF = 1$

B. 0010 0000, $OF = 0$

C. 1010 0000, $OF = 1$

D. 1010 0000, $OF = 0$

答案: B

解析: 因为 $x = +63$，$y = 31$，字长为 8 位，采用补码表示。

首先写出二进制补码:

$$[x]_{\text{补}} = 0011\ 1111$$

$$[y]_{补} = 0001\ 1111$$

计算：

$$x - y = 63 - 31 = 32$$

32 的 8 位补码为：

$$0010\ 0000$$

由于 8 位补码带符号整数的表示范围为：

$$-128 \sim 127$$

结果 32 在表示范围内，没有溢出，因此：

$$OF = 0$$

所以答案为 B。

题目 3.6 一个 C 语言程序在一台 32 位机器上运行。程序中定义了三个变量 x, y, z，其中 x 和 z 是 int 型，y 为 short 型。当 $x = 126$，$y = -9$ 时，执行赋值语句 $z = x + y$ 后，x, y, z 的值分别是_____。

A. $x = 0000007EH,\ y = FFF9H,\ z = 00000075H$

B. $x = 0000007EH,\ y = FFF9H,\ z = FFFF0075H$

C. $x = 0000007EH,\ y = FFF7H,\ z = FFFF0075H$

D. $x = 0000007EH,\ y = FFF7H,\ z = 00000075H$

答案： D

解析： 在 32 位机器中，int 型通常为 32 位，short 型通常为 16 位。

首先：

$$x = 126 = 7EH$$

由于 x 是 int 型，所以用 32 位十六进制表示为：

$$x = 0000007EH$$

其次：

$$y = -9$$

9 的 16 位二进制表示为：

$$0000\ 0000\ 0000\ 1001$$

求 -9 的补码，按位取反加 1：

$$1111\ 1111\ 1111\ 0110 + 1 = 1111\ 1111\ 1111\ 0111$$

即：

$$y = FFF7H$$

执行：

$$z = x + y = 126 + (-9) = 117$$

而：

$$117 = 75H$$

由于 z 是 int 型，所以用 32 位十六进制表示为：

$$z = 00000075H$$

因此：

$$x = 0000007EH, \quad y = FFF7H, \quad z = 00000075H$$

所以答案为 D。

3.3 零扩展与符号扩展

知识点：

零扩展用于无符号数扩展，高位补 0。

例如：

$$1010 \rightarrow 00001010$$

符号扩展用于有符号补码扩展，高位补符号位。

正数符号位为 0，高位补 0：

$$0101 \rightarrow 00000101$$

负数符号位为 1，高位补 1：

$$1011 \rightarrow 11111011$$

补码符号扩展后，数值大小不变。

题目 3.7 8 位补码定点整数 1010 0101，扩展至 16 位后的值用十六进制表示为 ()。

A. $00A5H$

B. $A500H$

C. $A5FFH$

D. $FFA5H$

答案： D

解析： 补码定点整数扩展位数时采用符号扩展，即用原数的符号位填充高位。

原 8 位补码为：

$$1010\ 0101$$

最高位为 1，表示负数，因此扩展到 16 位时，高 8 位应全部补 1：

$$1111\ 1111\ 1010\ 0101$$

将其转换为十六进制：

$$1111\ 1111\ 1010\ 0101 = FFA5H$$

因此答案为 D。

3.4 多功能 ALU 的组成

知识点：

ALU 是算术逻辑单元，主要完成算术运算和逻辑运算。

多功能 ALU 通常包括：

- 加法器：完成加法、减法等算术运算。
- 逻辑运算部件：完成与、或、非、异或等逻辑运算。
- 移位器：完成逻辑左移、逻辑右移、算术右移等操作。
- 选择器：根据控制信号选择输出哪一种运算结果。
- 标志生成电路：产生零标志、溢出标志、进位标志、符号标志等。

常见标志位包括：

ZF 零标志，结果为0 时置1

OF 溢出标志，补码运算溢出时置1

CF 进位标志，无符号运算产生进位或借位时置1

SF 符号标志，结果最高位为1 时置1

题目 3.8 ALU 的核心部件是 ()。

A. 多路选择器

B. 移位器

C. 加法器

D. 寄存器

答案： C

解析： ALU 是算术逻辑单元，主要完成算术运算和逻辑运算。

在 ALU 中，加法器是实现加法、减法、比较以及许多逻辑扩展操作的核心基础部件。减法可以通过补码加法实现，比较操作也常以减法或加法器运算结果为基础。

因此，ALU 的核心部件是加法器。

因此答案为 C。

题目 3.9 运算器的 ALU 输入端结构和寄存器组结构的选择会影响运算器速度，下面四个选择方案中，速度最慢的是 ()。

A. ALU 输入端采用锁存器向 ALU 传送操作数，寄存器组采用高速小存储器结构

B. ALU 输入端采用锁存器向 ALU 传送操作数，寄存器组采用独立寄存器结构

C. ALU 输入端采用多路选择器向 ALU 传送操作数，寄存器组采用独立寄存器结构

D. ALU 输入端采用多路选择器向 ALU 传送操作数，寄存器组采用高速小存储器结构

答案： A

解析： 运算器速度受 ALU 输入端结构和寄存器组结构影响。

ALU 输入端若采用锁存器传送操作数，通常需要先将操作数送入锁存器，再由锁存器送入 ALU，数据传送环节较多，速度较慢；若采用多路选择器，则可以根据控制信号直接选择操作数送入 ALU，速度相对较快。

寄存器组若采用高速小存储器结构，本质上类似用存储器方式组织寄存器，访问速度通常比独立寄存器结构慢；独立寄存器结构可以直接选通寄存器，速度较快。

因此，速度最慢的方案是：ALU 输入端采用锁存器向 ALU 传送操作数，寄存器组采用高速小存储器结构。

所以答案为 A。

4 控制器

本部分重点

- CPU 的组成，数据通路的组成、功能。
- CPU 中某些部件如 PC、控制器等的功能。
- 指令周期。
- 转移指令、条件转移指令的操作，如 beq、jal。
- 已知数据通路图，给出某指令的取指和执行阶段的全部传送操作及节拍安排。
- 了解 IR、MDR、MAR 等部件功能。

4.1 CPU 的组成与数据通路的组成、功能

知识点：

CPU 通常由运算器、控制器和寄存器组组成。

数据通路是指指令执行过程中，数据在各个部件之间传送所经过的路径。

数据通路通常包括：

- PC：程序计数器；
- IR：指令寄存器；
- 寄存器组；
- ALU：算术逻辑单元；
- MAR：存储器地址寄存器；
- MDR：存储器数据寄存器；
- 多路选择器；
- 存储器；
- 内部总线等。

数据通路的功能是完成取指、译码、执行、访存和写回等操作中的数据传送与处理。

题目 4.1 CPU 的四个主要功能是_____、_____、_____ 和时序控制。
答案： 指令控制；操作控制；数据加工
解析： CPU 的基本功能包括：

- 指令控制：控制程序中指令的执行顺序；

- 操作控制：产生完成各类操作所需的控制信号；
- 数据加工：完成算术运算和逻辑运算；
- 时序控制：协调各部件按照规定的时间顺序工作。

因此答案为：

指令控制；操作控制；数据加工；时序控制

题目 4.2 下列有关数据通路的叙述中，错误的是（）。

A. 数据通路由若干操作元件和状态元件连接而成

B. 数据通路的功能由控制部件送出的控制信号决定

C. 通用寄存器属于状态元件，但不包含在数据通路中

D. ALU 属于操作元件，用于执行各类算术和逻辑运算

答案： C

解析： 数据通路是指指令执行过程中，数据在各个部件之间传送和处理所经过的路径，通常由组合逻辑操作元件和寄存器等状态元件组成。

ALU 属于操作元件，主要完成算术运算和逻辑运算；通用寄存器属于状态元件，用于暂存操作数、中间结果或运算结果，也是数据通路的重要组成部分。

因此，“通用寄存器属于状态元件，但不包含在数据通路中”这一说法错误。

所以答案为 C。

题目 4.3 CPU 内若设置一组通用寄存器，那么通用寄存器的位数一般取决于（）。

A. 指令字的长度

B. 地址寄存器的位数

C. 机器字长

D. 主存容量

答案： C

解析： 通用寄存器主要用于暂存操作数、中间结果和运算结果，其位数通常应与 CPU 一次能够处理的数据位数相一致。

机器字长是指计算机一次能直接处理的二进制数据位数，通常与运算器、通用寄存器的位数有关。因此，通用寄存器的位数一般取决于机器字长。

因此答案为 C。

4.2 CPU 中常见部件的功能

知识点：

PC：程序计数器

PC 用于存放下一条将要执行的指令地址。通常顺序执行时：

$$PC \leftarrow PC + 4$$

遇到转移或跳转指令时：

$$PC \leftarrow \text{目标地址}$$

控制器

控制器根据指令操作码、时序信号和状态标志产生控制信号，控制数据通路中各部件协调工作。

IR：指令寄存器

IR 用于存放当前正在执行的指令。

MAR：存储器地址寄存器

MAR 用于存放当前要访问的存储器地址。

MDR：存储器数据寄存器

MDR 用于暂存从存储器读出的数据，或暂存将要写入存储器的数据。

题目 4.4 以下有关 CPU 中部分部件功能的描述中，错误的是 ()。

A. IR 称为指令寄存器，用来存放当前指令的操作码

B. PC 用于存放将要执行的指令的地址

C. 通过将 PC 按当前指令长度增量，可实现指令的按序执行

D. 控制单元用于对指令操作码译码并生成控制信号

答案： A

解析： IR 是指令寄存器，用来存放当前正在执行的整条指令，而不只是操作码。

一条指令通常包括操作码字段和地址码字段等，操作码只是指令的一部分。

PC 用于存放将要执行的指令地址，顺序执行时，通常通过：

$$PC \leftarrow PC + \text{当前指令长度}$$

使 PC 指向下一条指令。

控制单元用于对指令操作码译码，并产生相应的控制信号。

因此错误的是 A。

题目 4.5 执行完当前指令后，PC 中存放的是后继指令的地址，因此 PC 的位数和 () 的位数相同。

A. 指令寄存器 IR

B. 程序状态字寄存器 PSW

C. 主存数据寄存器 MDR

D. 主存地址寄存器 MAR

答案： D

解析： PC 用于存放指令地址，即下一条将要执行的指令在主存中的地址。

MAR 是主存地址寄存器，用于存放访问主存时的地址。

由于 PC 和 MAR 都需要表示主存地址空间中的地址，因此二者的位数通常相同。

IR 用于存放指令，MDR 用于存放主存读写数据，PSW 用于存放程序状态信息，它们的位数不一定与 PC 相同。

因此答案为 D。

题目 4.6 假设控制信号用 C_n 来表示，指令译码器输出用 I_m 表示，节拍信号用 M_k 表示，状态反馈信息用 B_j 表示，则控制器的基本原理可表示为 ()。

A. $C_n = f(I_m)$

B. $C_n = f(I_m, B_j)$

C. $C_n = f(M_k, B_j)$

D. $C_n = f(I_m, M_k, B_j)$

答案： D

解析： 控制器的作用是根据当前指令、当前执行节拍以及状态反馈信息产生相应的控制信号。

其中：

$$I_m$$

表示指令译码器输出，用于说明当前正在执行哪一类指令；

$$M_k$$

表示节拍信号，用于说明当前处于指令执行过程中的哪一个时刻或阶段；

$$B_j$$

表示状态反馈信息，例如条件码、标志位、中断请求等。

因此控制信号不仅与指令译码结果有关，还与节拍信号和状态反馈信息有关。

所以控制器的基本原理可表示为：

$$C_n = f(I_m, M_k, B_j)$$

因此答案为 D。

题目 4.7 CPU 响应中断时需要保存当前现场，这里现场指的是_____ 寄存器和_____ 寄存器的内容，它们被保存到_____ 中。

答案： 程序计数器 PC；程序状态字 PSW；堆栈

解析： CPU 响应中断时，需要保存当前程序的现场，以便中断服务程序执行完毕后能够正确返回原程序继续执行。

现场通常包括当前程序的断点地址和程序状态信息。其中，断点地址保存在程序计数器 PC 中，程序状态信息保存在程序状态字寄存器 PSW 中。

这些内容一般被保存到堆栈中，中断返回时再从堆栈中恢复。

因此答案为：程序计数器 PC；程序状态字 PSW；堆栈。

题目 4.8 堆栈寻址需在 CPU 内设一个专用的寄存器,称为_____,其内容是_____

答案： 堆栈指针寄存器 SP；栈顶地址

解析： 堆栈是一种后进先出的存储结构，常用于保存返回地址、现场信息和临时数据等。

为了支持堆栈操作，CPU 内通常设置一个专用寄存器，称为堆栈指针寄存器，记作 SP。

SP 中保存当前栈顶单元的地址，用于指示压栈和出栈操作的位置。

因此答案为：堆栈指针寄存器 SP；栈顶地址。

题目 4.9 状态寄存器中的各个状态标志位是依据_____ 来置位的。

A. 算术逻辑部件上次的运算结果

B. CPU 将要执行的指令

C. CPU 已执行的指令

D. 累加器中的数据

答案： A

解析： 状态寄存器用于保存 CPU 运算后的状态信息，如零标志、进位标志、溢出标志、符号标志等。

这些状态标志位通常由 ALU 的运算结果决定。例如，若运算结果为 0，则零标志位被置位；若补码运算发生溢出，则溢出标志位被置位；若结果最高位为 1，则符号标志位被置位。

因此，状态寄存器中的各个状态标志位是依据算术逻辑部件上次的运算结果来置位的。

所以答案为 A。

题目 4.10 在计算机系统中，描述系统运行状态的部件是 ()。

A. 程序计数器

B. 累加器

C. 通用寄存器

D. PSW 寄存器

答案： D

解析： PSW 是程序状态字寄存器，用来保存程序或 CPU 运行过程中的状态信息。

常见状态信息包括进位标志、零标志、溢出标志、符号标志、中断允许状态等。

程序计数器 PC 用于存放下一条将要执行的指令地址；累加器和通用寄存器主要用于暂存操作数或运算结果。

因此，描述系统运行状态的部件是 PSW 寄存器，答案为 D。

4.3 指令周期

知识点：

指令周期是指 CPU 从取出一条指令到执行完该指令所经历的全过程。

一条指令的执行过程通常包括：

取指 → 译码 → 执行 → 访存 → 写回

不同类型的指令不一定都需要所有阶段。

例如：

R 型运算指令：

取指 → 译码 → 执行 → 写回

访存读指令：

取指 → 译码 → 地址计算 → 访存 → 写回

存储指令：

取指 → 译码 → 地址计算 → 访存

题目 4.11 CPU 从主存取出一条指令并执行该指令的时间叫_____。

答案： 指令周期

解析： 指令周期是指 CPU 从主存中取出一条指令，并完成该指令执行所需要的全部时间。

一条指令的执行过程通常包括取指、译码、执行等阶段，不同类型的指令可能还包括访存、写回等阶段。

因此答案为：指令周期。

题目 4.12 若指令流水线把一条指令分为取指、分析和执行三部分，且三部分的时间分别是 $t_{取指}=2ns$，$t_{分析}=2ns$，$t_{执行}=1ns$。则 100 指令全部执行完毕需 () ns。

A. 163

B. 183

C. 193

D. 203

答案： D

解析： 该流水线共有 3 个阶段：取指、分析、执行。

流水线的时钟周期由最长阶段决定：

$$T=\max(2,2,1)=2ns$$

第一条指令完成需要经历三个阶段，所用时间为：

$$2+2+1=5ns$$

从第二条指令开始，在流水线稳定后，每隔一个时钟周期完成一条指令，即每隔 $2ns$ 完成一条指令。

因此 100 条指令全部完成所需时间为：

$$5 + (100 - 1) \times 2 = 5 + 198 = 203ns$$

因此答案为 D。

题目 4.13 假定执行最复杂的指令需要完成 6 个子功能，分别由对应的功能部件 A–F 完成，部件 A–F 所花的时间分别为 $80ps, 40ps, 50ps, 70ps, 40ps, 30ps$，流水段寄存器延时为 $20ps$。现把最后两个功能部件 F 和 E 合并，以产生一个 5 段流水线。该 5 段流水线的时钟周期至少是 () ps。

A. 100

B. 90

C. 80

D. 70

答案： A

解析： 流水线的时钟周期应由最慢流水段的延迟决定，并且还要加上流水段寄存器的延迟。

将功能部件 E 和 F 合并后，该流水段的延迟为：

$$40 + 30 = 70ps$$

因此 5 个流水段的功能部件延迟分别为：

$$80ps,\ 40ps,\ 50ps,\ 70ps,\ 70ps$$

其中最大延迟为：

$$\max(80, 40, 50, 70, 70) = 80ps$$

再加上流水段寄存器延迟 $20ps$，所以流水线时钟周期至少为：

$$80 + 20 = 100ps$$

因此答案为 A。

题目 4.14 某计算机的指令流水线由 4 个功能段组成，指令流经各功能段的时间（忽略各功能段之间的缓存时间）分别为 $90ns$、$80ns$、$70ns$ 和 $60ns$，则该计算机的 CPU 周期至少是 ()。

A. $90ns$

B. $80ns$

C. $70ns$

D. $60ns$

答案： A

解析： 流水线的 CPU 周期，也称流水线时钟周期，应由最慢的流水段决定。

题目中 4 个功能段的时间分别为：

$$90ns,\ 80ns,\ 70ns,\ 60ns$$

其中最大值为：

$$\max(90, 80, 70, 60) = 90ns$$

又因为题目说明忽略各功能段之间的缓存时间，所以不需要再加流水段寄存器延迟。

因此该计算机的 CPU 周期至少为：

$$90ns$$

所以答案为 A。

4.4 转移指令与条件转移指令

知识点：

转移指令用于改变程序执行顺序。

条件转移指令 beq

$$beq\ rs,\ rt,\ offset$$

若：

$$R[rs] = R[rt]$$

则发生转移：

$$PC \leftarrow PC + 4 + (offset \times 4)$$

否则：

$$PC \leftarrow PC + 4$$

跳转并链接指令 jal

$$jal\ target$$

执行过程为：

$$R[31] \leftarrow PC + 4$$

$$PC \leftarrow target$$

其中 $R[31]$ 通常为返回地址寄存器。

4.5 数据通路传送操作与节拍安排

知识点：

已知数据通路图时，常要求写出某条指令在取指阶段和执行阶段的全部传送操作及节拍安排。

例如，普通取指过程可以写为：

$$T_0：\ MAR \leftarrow PC$$

$$T_1：\ MDR \leftarrow M[MAR],\quad PC \leftarrow PC + 4$$

$$T_2：\ IR \leftarrow MDR$$

不同指令在执行阶段的传送操作不同。

例如加法指令可能包括：

$$A \leftarrow R[rs],\quad B \leftarrow R[rt]$$

$$ALUOut \leftarrow A + B$$

$$R[rd] \leftarrow ALUOut$$

访存指令可能包括：

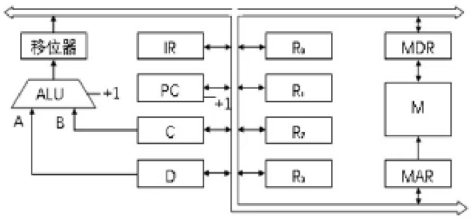
$$MAR \leftarrow R[base] + offset$$

$$MDR \leftarrow M[MAR]$$

$$R[rt] \leftarrow MDR$$

题目 4.15 某计算机的数据通路如下图所示，其中 M 为主存，MDR 为主存数据寄存器，MAR 为主存地址寄存器，$R_0 \sim R_3$ 为通用寄存器，IR 为指令寄存器，PC 为程序计数器，C、D 为暂存寄存器，ALU 为算术逻辑单元，移位器不移位，控制信号可双向传送。求指令

$$ADD\ R_2, (R_1)$$



周期	操作
C_0	
C_1	
C_2	
...	...

图 1: Enter Caption

的取指和执行过程，其中指令功能是：

$$R_2 \leftarrow R_2 + M[(R_1)]$$

该指令长度占一个编址单位。

答案： 微操作序列如下：

周期	微操作
C_0	$PC \rightarrow MAR, \quad Read$
C_1	$M[MAR] \rightarrow MDR, \quad PC + 1 \rightarrow PC$
C_2	$MDR \rightarrow IR$
C_3	$R_1 \rightarrow MAR, \quad Read$
C_4	$M[MAR] \rightarrow MDR$
C_5	$R_2 \rightarrow C$
C_6	$MDR \rightarrow D$
C_7	$C + D \rightarrow R_2$

解析： 该指令为：

$$ADD\ R_2, (R_1)$$

其中 (R_1) 表示寄存器间接寻址，即 R_1 中保存的是主存地址，真正的源操作数在主存单元 $M[(R_1)]$ 中。

指令功能为：

$$R_2 \leftarrow R_2 + M[(R_1)]$$

由于该指令长度占一个编址单位，因此取指阶段中：

$$PC + 1 \rightarrow PC$$

取指过程为：

$$C_0: \ PC \rightarrow MAR, \quad Read$$

表示把当前指令地址送入 MAR，并发出读主存信号。

$$C_1: \ M[MAR] \rightarrow MDR, \quad PC + 1 \rightarrow PC$$

表示从主存中读出指令，同时 PC 指向下一条指令。

$$C_2: \ MDR \rightarrow IR$$

表示将取出的指令送入指令寄存器 IR。
执行阶段中，需要先根据 R_1 的内容访问主存，取出源操作数：

$$C_3: \ R_1 \rightarrow MAR, \quad Read$$

$$C_4: \ M[MAR] \rightarrow MDR$$

然后将 R_2 和主存中取出的操作数分别送入暂存寄存器：

$$C_5: \ R_2 \rightarrow C$$

$$C_6: \ MDR \rightarrow D$$

最后由 ALU 完成加法，并写回 R_2：

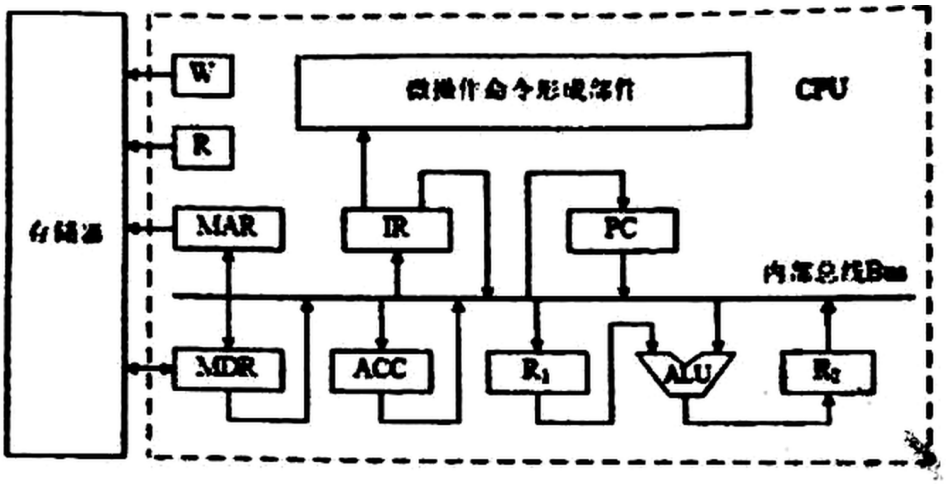
$$C_7: \ C + D \rightarrow R_2$$

因此完整微操作序列为：

周期	微操作
C_0	$PC \rightarrow MAR, \quad Read$
C_1	$M[MAR] \rightarrow MDR, \quad PC + 1 \rightarrow PC$
C_2	$MDR \rightarrow IR$
C_3	$R_1 \rightarrow MAR, \quad Read$
C_4	$M[MAR] \rightarrow MDR$
C_5	$R_2 \rightarrow C$
C_6	$MDR \rightarrow D$
C_7	$C + D \rightarrow R_2$

题目 4.16

(北科大 2026 期末考试) 设 CPU 中各部件及其相互连接关系如下图所示。图中 W 是写控制标志，R 是读控制标志，R_1 和 R_2 是暂存器。



1. 假设要求在取指周期中由 ALU 完成 $(PC)+1 \rightarrow PC$ 的操作，即 ALU 可以对它的一个源操作数完成加 1 的运算。要求以最少的节拍写出取指周期全部微操作命令及节拍安排。
2. 写出指令 ADD $\#a$ 在执行阶段所需的微操作命令及节拍安排。其中 $\#$ 为立即寻址特征，隐含的操作数在 ACC 中。

答案： （1）取指周期的微操作及节拍安排为：

节拍	微操作
T_0	$PC \rightarrow MAR, \quad R$
T_1	$M[MAR] \rightarrow MDR, \quad PC+1 \rightarrow PC$
T_2	$MDR \rightarrow IR$

（2）指令 ADD $\#a$ 的执行阶段微操作及节拍安排为：

节拍	微操作
T_0	$ACC \rightarrow R_1$
T_1	IR(地址码字段) $\rightarrow R_2$
T_2	$R_1 + R_2 \rightarrow ACC$

解析： （1）取指周期的基本任务是根据 PC 中的地址从主存取出指令，并送入指令寄存器 IR。

首先将 PC 中的指令地址送入 MAR，同时发出读命令：

$$PC \rightarrow MAR, \quad R$$

然后主存根据 MAR 中的地址读出指令，送入 MDR。题目要求取指周期中由 ALU 完成：

$$(PC) + 1 \rightarrow PC$$

因此可以在读存储器的同时完成 PC 加 1 操作：

$$M[MAR] \rightarrow MDR, \quad PC + 1 \rightarrow PC$$

最后将 MDR 中的指令送入 IR：

$$MDR \rightarrow IR$$

所以取指周期至少需要 3 个节拍。

（2）指令 ADD $\#a$ 表示立即寻址，操作数 a 直接包含在指令中，另一个隐含操作数在 ACC 中，指令功能为：

$$(ACC) + a \rightarrow ACC$$

执行时，先将 ACC 中的内容送入暂存器 R_1，再将指令中给出的立即数 a 送入暂存器 R_2，最后由 ALU 完成加法运算，并将结果送回 ACC：

$$ACC \rightarrow R_1$$

$$IR(\text{地址码字段}) \rightarrow R_2$$

$$R_1 + R_2 \rightarrow ACC$$

因此执行阶段可按 3 个节拍完成。

4.6 IR、MDR、MAR 等部件

知识点：

IR：指令寄存器，用于存放正在执行的指令。

MDR：存储器数据寄存器，用于暂存从存储器读出的数据或将要写入存储器的数据。

MAR：存储器地址寄存器，用于存放当前访问存储器的地址。

这三个寄存器常出现在取指、访存和写回相关的数据通路题中。

5 存储器

本部分重点

- 层次化存储体系，速度、容量的关系。
- 存储器容量与地址。

5.1 层次化存储体系，速度、容量的关系

知识点：

存储器采用分级体系结构，通常按距离 CPU 从近到远可分为：

$$\text{寄存器} \rightarrow \text{Cache} \rightarrow \text{主存} \rightarrow \text{辅存}$$

一般规律为：

- 越靠近 CPU，速度越快，容量越小，价格越高；
- 越远离 CPU，速度越慢，容量越大，价格越低。

因此，存储器分级的主要目的是在速度、容量和价格之间取得折中，解决三者之间的矛盾。

题目 5.1 存储器分层体系结构中，存储器从速度最快到最慢的排列顺序是 ()。

A. 寄存器–主存–Cache–辅存
B. 寄存器–主存–辅存–Cache
C. 寄存器–Cache–辅存–主存
D. 寄存器–Cache–主存–辅存

答案： D

解析： 存储器层次结构中，越靠近 CPU，速度越快，容量越小，价格越高；越远离 CPU，速度越慢，容量越大，价格越低。

速度从快到慢的顺序为：

$$\text{寄存器} \rightarrow \text{Cache} \rightarrow \text{主存} \rightarrow \text{辅存}$$

因此答案为 D。

题目 5.2 计算机的存储器采用分级方式是为了 ()。

A. 减少主机箱的体积
B. 存储大量数据方便
C. 解决容量、速度、价格三者之间的矛盾

D. 操作方便

答案： C

解析： 高速存储器速度快但容量小、价格高；低速存储器容量大、价格低但速度慢。采用存储器分级体系，可以在速度、容量和价格之间取得较好的折中。

因此答案为 C。

5.2 存储器容量与地址

知识点：

存储器容量与地址范围的计算是常考内容。

若存储器容量为 N 个地址单元，首地址为 A，则末地址为：

$$A+N-1$$

若地址线有 n 根，且按字节编址，则最大寻址空间为：

$$2^n \text{ Byte}$$

常见单位换算：

$$1KB = 2^{10}B = 1024B$$

$$1MB = 2^{20}B$$

$$1GB = 2^{30}B$$

注意：Cache 是主存内容的副本，用于提高访问速度，通常不增加存储系统的有效容量。

题目 5.3 内存储器容量为 $6K$ 时，若首地址为 $00000H$，那么末地址的十六进制表示为_____。

答案： $017FFH$

解析： $1K = 1024$，所以：

$$6K = 6 \times 1024 = 6144$$

十进制的 6144 转换为十六进制为：

$$6144 = 1800H$$

因为首地址为 $00000H$，末地址应为：

$$00000H + 1800H - 1 = 017FFH$$

所以末地址为 $017FFH$。

题目 5.4 由容量为 $16KB$ 的缓存和容量为 $16MB$ 的主存构成的存储系统的总容量为_____。

答案： $16MB$

解析： Cache 是主存内容的副本，用于提高访问速度，并不增加存储系统的有效容量。

因此由 $16KB$ Cache 和 $16MB$ 主存构成的存储系统，其总容量通常按主存容量计算，为：

$$16MB$$

所以答案为 $16MB$。

题目 5.5 存储容量为 $64KB$，按字节编址，$0000H$–$1FFFH$ 为 ROM 区，其余为 RAM 区，至少需要_____ 片 $8K \times 4$ 位的 RAM。

答案： 14

解析： 存储器总容量为：

$$64KB$$

地址范围 $0000H$–$1FFFH$ 的容量为：

$$1FFFH - 0000H + 1 = 2000H = 8KB$$

该区域为 ROM 区，因此 RAM 区容量为：

$$64KB - 8KB = 56KB$$

题目要求按字节编址，因此每个存储单元为 8 位。

一片 $8K \times 4$ 位 RAM 的容量为：

$$8K \times 4bit$$

要组成 $8K \times 8$ 位，需要 2 片 $8K \times 4$ 位 RAM。

而 $56KB$ 可以分成：

$$\frac{56KB}{8KB} = 7$$

组 $8K \times 8$ 位存储区。

所以总共需要：

$$7 \times 2 = 14$$

片 RAM。

因此答案为：14。

6 杂例

本部分主要记录考试范围内比较模糊的一些题目, 比如流水线是否会包含 (虽然可以包含进入指令周期, 但是这种背诵题目应该还是很痛苦的, 放到这里缓解一下压力), 以及一些非常规题目等

题目 6.1 为什么流水线方式会延长一条指令的执行时间?

答案: 因为流水线会引入流水段寄存器开销，且时钟周期由最慢流水段决定。

解析: 流水线技术提高的是多条指令连续执行时的吞吐率，并不一定缩短单条指令的执行时间。

在非流水方式下，一条指令可以按照各阶段实际需要的时间顺序完成；而采用流水线后，每条指令必须经过所有流水段。

同时，流水段之间需要加入流水段寄存器，这会带来锁存、传输等额外时间开销。

此外，流水线的时钟周期由最慢的流水段决定。若各流水段执行时间不均衡，则较快的流水段也必须等待最慢流水段完成。

因此，一条指令在流水线中的完成时间可能比非流水方式更长。

题目 6.2 IBM370 的短浮点数格式中，总位数为 32 位，左边第一位为数符，随后 7 位为阶码，用移码表示，偏置常数为 64；右边 24 位为 6 位十六进制原码小数表示的尾数。

规格化尾数形式为:

$$0.x_1x_2x_3x_4x_5x_6$$

其中 $x_1 \sim x_6$ 为十六进制表示，最高位 x_1 为非 0 数，底为 16。

若用该浮点数格式表示十六进制真值 $-1234H$，则对应的机器数是什么?

答案: $C4123400H$

解析: 该题不是 IEEE754 浮点数格式，而是 IBM370 短浮点数格式。

其机器数格式为:

数符1 位　阶码7 位　尾数24 位

首先将真值规格化为 IBM370 的十六进制浮点形式:

$$-1234H = -0.123400H \times 16^4$$

因此数符位为:

1

阶码真值为:

4

由于阶码采用移码表示，偏置常数为 64，所以移码阶码为:

$$64 + 4 = 68$$

$$68 = 44H = 1000100_2$$

尾数为：

$$123400H$$

将数符、阶码和尾数组合：

$$1\ 1000100\ 00010010001101000000000$$

按 4 位一组写成十六进制：

$$1100\ 0100\ 0001\ 0010\ 0011\ 0100\ 0000\ 0000$$

即：

$$C4123400H$$

所以对应的机器数为：

$$C4123400H$$

题目 6.3 挂接在总线上的多个部件 ()。

A. 只能分时向总线发送数据，并且只能分时从总线接收数据

B. 可同时向总线发送数据，但只能分时从总线接收数据

C. 可同时向总线发送数据，并同时从总线接收数据

D. 只能分时向总线发送数据，但可同时从总线接收数据

答案： D

解析： 总线是多个部件共享的一组信息传输线路。

由于总线是共享资源，同一时刻只能允许一个部件向总线发送数据，否则会发生总线冲突。因此，多个部件只能分时向总线发送数据。

但是，总线上传输的数据可以同时被多个部件接收。也就是说，一个部件发送数据时，多个接收部件可以同时从总线上接收该数据。

所以，挂接在总线上的多个部件只能分时向总线发送数据，但可同时从总线接收数据。

因此答案为 D。

题目 6.4 I/O 指令实现的数据传送通常发生在 ()。

A. I/O 设备和 I/O 端口之间

B. 通用寄存器和 I/O 设备之间

C. I/O 端口和 I/O 端口之间

D. 通用寄存器和 I/O 端口之间

答案： D

解析： I/O 指令用于实现 CPU 与外部设备之间的数据交换。

在程序控制方式下，CPU 通常不能直接与 I/O 设备本体交换数据，而是通过 I/O 端口进行访问。执行输入指令时，数据一般从 I/O 端口送入 CPU 的通用寄存器；执行输出指令时，数据一般从 CPU 的通用寄存器送到 I/O 端口。

因此，I/O 指令实现的数据传送通常发生在通用寄存器和 I/O 端口之间。

所以答案为 D。

题目 6.5 下列关于 RISC 的说法中，错误的是 ()。

A. RISC 普遍采用微程序控制器

B. RISC 大多数指令在一个时钟周期内完成

C. RISC 的内部通用寄存器数量相对 CISC 多

D. RISC 的指令数、寻址方式和指令格式种类相对 CISC 少

答案： A

解析： RISC 是精简指令集计算机，其主要特点包括：指令种类少、指令格式规整、寻址方式少、通用寄存器数量较多，并且多数指令可在一个时钟周期内完成。

RISC 为了提高指令执行速度，通常采用硬布线控制器，而不是微程序控制器。微程序控制器更多见于 CISC 结构中，因为 CISC 指令复杂，适合用微程序方式实现控制。

因此，“RISC 普遍采用微程序控制器” 这一说法错误。

所以答案为 A。

题目 6.6 某浮点数字长 16 位，其中阶码 4 位（含 1 位阶符），以 2 为底，移码表示；尾数含 1 位数符，共 12 位，补码表示，规格化。求：

（1）真值 $(-2^5 \times 0.375)_{10}$ 的浮点数代码（16 进制表示编码）。

（2）浮点数编码为 1010 0110 10000000 的真值。

答案： （1）$CA00H$；（2）3.25

解析： 该浮点数格式为：

$$\text{阶码 4 位} + \text{尾数 12 位}$$

阶码采用移码表示，偏置值为：

$$2^{4-1} = 8$$

尾数采用补码表示，且要求规格化。

（1）真值为：

$$-2^5 \times 0.375 = -0.375 \times 2^5$$

由于尾数采用补码规格化表示，负数规格化尾数应满足：

$$1.0xxx\cdots$$

将其规格化为：

$$-0.375 \times 2^5 = -0.75 \times 2^4$$

因此阶码真值为：

$$E = 4$$

阶码采用移码表示：

$$E_{移} = 4 + 8 = 12 = 1100_2$$

尾数为：

$$-0.75$$

+0.75 的 12 位定点小数原码为：

$$0.11000000000$$

所以 −0.75 的补码为：

$$1.01000000000$$

即尾数编码为：

$$101000000000$$

将阶码和尾数组合：

$$1100\ 1010\ 0000\ 0000$$

转换为十六进制：

$$1100\ 1010\ 0000\ 0000 = CA00H$$

所以浮点数代码为：

$$CA00H$$

（2）已知浮点数编码为：

$$1010\ 0110\ 10000000$$

其中阶码为：

$$1010_2 = 10$$

阶码采用移码表示，偏置值为 8，所以阶码真值为：

$$E = 10 - 8 = 2$$

尾数为：

$$0110\ 10000000$$

即：

$$0.11010000000_2$$

其十进制值为：

$$0.1101_2 = \frac{1}{2} + \frac{1}{4} + \frac{1}{16} = 0.8125$$

所以该浮点数的真值为：

$$0.8125 \times 2^2 = 3.25$$

因此答案为：

$$3.25$$

题目 6.7 寄存器中的值有时是地址，有时是数据，在形式上没有差别，只有通过 () 才能识别它是数据还是地址。

A. 寄存器编号

B. 判断程序

C. 指令操作码或寻址方式位

D. 时序信号

答案： C

解析： 寄存器中存放的内容本质上都是二进制位串，仅从形式上无法判断它表示数据还是地址。

在指令执行过程中，寄存器内容的含义需要由当前指令的操作码和寻址方式来解释。

例如，某些指令中寄存器内容可作为操作数，而在寄存器间接寻址中，寄存器内容则表示操作数所在主存单元的有效地址。

因此答案为 C。

题目 6.8 已知 $X = 0.1010$，$Y = -0.1011$，用原码一位乘法计算 $X \times Y$，其中寄存器、加法器的宽度均为 4 位，写出结果。

答案：

$$[X \times Y]_{原} = 1.01101110$$

即：

$$X \times Y = -0.01101110$$

解析： 原码乘法中，乘积的符号位由两个操作数的符号位异或得到。

因为 X 为正数，Y 为负数，所以乘积为负数。

数值部分相乘：

$$0.1010_2 \times 0.1011_2$$

其中：

$$1010_2 = 10, \qquad 1011_2 = 11$$

由于两个数的小数点后均为 4 位，因此：

$$0.1010_2 = \frac{10}{16}, \qquad 0.1011_2 = \frac{11}{16}$$

所以数值部分乘积为：

$$\frac{10}{16} \times \frac{11}{16} = \frac{110}{256} = 0.01101110_2$$

乘积符号为负，因此：

$$X \times Y = -0.01101110_2$$

写成原码形式为：

$$[X \times Y]_{原} = 1.01101110$$