1 简述冯诺伊曼系结构计算机主要特点,设计思想

特点：

- 计算机硬件由…五大设备组成
- 指令和数据均用二进制表示
- 指令在存储器内按顺序存放
- 指令由操作码和地址码组成，操作码表示操作的性质，地址码用来表示操作数在存储器中的位置。
- 指令和数据以同等地位存储在存储器中，并可按地址寻访。
- 机器以运算器为中心

设计思想：

- 存储程序

2.存储器的层次结构主要体现在何处 ？各有什么特点？ 为什么要采用分层体系结构？说明每一层存储器所用的存储器所用的存储介质种类？

- Cache-主存 主存-辅存

主存：主存用来存放将要运行的程序和数据，cpu 可以直接对其进行访问，其特点是容量较小，存取速度较快，价格较高

Cache：cache 位于 cpu 和主存之间，用来存放正在执行的程序和数据，其特点是容量小，存取速度快，价格高

辅存：用来存放暂时不用的程序和数据，cpu 不能直接访问，只能与主存交换信息，其特点是容量大存取速度较慢，价格较低。

- Cache：SRAM 主存：DRAM

为了解决存储系统的大容量，高速度和低成本的三个互相制约的矛盾，cache-主存层次主要解决了 cpu 和主存速度不匹配的问题，主存-辅存主要解决了存储系统的容量问题。

3.分别说出 SRAM DRAM 的工作原理，比较它们的优缺点。

SRAM 的存储单元是用双稳态触发器来记忆信息的，因此即使信息被读出后，只要电源一直加在电路上，它就保持原有状态，不需要刷新，但当电源掉电时，信息将会丢失，它属于易失性半导体存储器。

DSAM 是靠电容存储电荷的原理存储信息的，由于 DSAM 电容上的电荷一般只

能维持 1-2ms，因此即使电源不断电，信息也会自动消失，为此，每隔一段时间必须刷新，常用的刷新方式有，集中刷新，分散刷新，异步刷新。

SRAM 的特点是存取速度快，但集成度低，功耗大，且价格较贵，所以一般用来组成高速缓冲存储器。

相对于 SRAM 来说，DRAM 存取速度相对较慢，但是 DRAM 容易集成，价位低，容量大和功耗低等优点，所以一般用来组成大容量的主存。

4 cache 有哪几种地址映射方法？各有什么特点？

直接映射，全相联映射，组相联映射

直接映射：每一个主存块只与一个缓冲块相对应。其优点是实现简单，缺点是不够灵活，缓存空间地址得不到充分利用，空间利用率低，命中率低。

全相联映射：允许主存的每一个字块映射到 cache 中任何一块位置上，优点是灵活，空间利用率高，命中率高，缺点是地址变换速度慢，实现成本高。

组相联映射：是对直接映射和全相联映射的一种折中方式，兼顾了两者的优点，尽量避免了两者的缺点。

5 请说明指令周期，机器周期，时钟周期之间的关系。

指令周期：取出并执行一条指令的时间

机器周期：也称 cpu 周期，通常用内存中存取一个指令是最短时间来规定 cpu 周期，

时钟周期：也称节拍脉冲，是处理操作的基本单元。

一般来说，指令周期常用·若干个机器周期来表示，一个机器周期包含若干个时钟周期

6 程序查询方式和程序中断方式的异同，优缺点，适用场合

程序查询方式：数据传送完全由计算机程序控制，优点是 cpu 操作能与外设同步。硬件结构简单 缺点是 cpu 时间浪费严重 主要用于简单系统控制。

程序中断方式是外设主动通知 cpu 进行信息传送的，优点是节约 cpu 的时间。缺点是硬件设备相当复杂，主要用于处理随机事件

7 cpu 中的控制器的主要有那两种设计方法

微程序控制器和硬布线控制器，前者具有规整性，灵活性和可维护性优点，后者主要是执行速度快。

8 什么是 RISC ?RISC 的特点？

RISC 是精简指令系统计算机。

1 选取使用频率高的一些简单指令，复杂指令的功能由简单指令的组合来实现

2 指令字长固定，指令格式种类少，寻址方式种类少

3 Cpu 中通用寄存器的数量相当多

4 一定采用流水线控制

5 以硬布线为主，不用或少用微程序控制

6 只有 load/store（取数，存数）指令指令访存

7 特别注重编译优化

9. 若某计算机采用微程序控制方式实现指令，请论述一下计算机是如何自动运行我们所编写的程序的。

高级语言通过编译生成汇编语言，汇编语言再生成机器语言指令，计算机自动执行机器语言指令……4 分

执行机器指令时，先把第一条机器指令所在地址装入 PC，通过 PC 自动加入和执行转移指令时重新装载 PC,来自动运行下一条机器指令…….3 分

通过机器指令的译码，获得该指令所对应的微程序的入口地址，通过每条微指令中携带下一条微指令的地址来自动运行该机器指令所对应的微程序….3 分

10 什么叫指令，什么叫指令系统？

指令是计算机执行某种操作的命令，也就是常说的机器指令。一台机器中所有机器指令的集合，称这台计算机的指令系统。

11 程序中断大概分为几个阶段

中断请求（1 分）中断判优（1 分）中断响应（1 分）中断服务（1 分）中断返回（1 分）

12 一个较完整的指令系统包括哪几类？

包括：数据传送指令、算术运算指令、逻辑运算指令、程序控制指令、输入输出指令、堆栈指令、字符串指令、特权指令等

13 简要论述计算机是如何去执行高级语言编写的一条语句的？

高级语言编译为汇编语言、汇编语言生成机器语言（机器指令）、采用微指令

执行机器指令

13 比较水平微指令和垂直微指令的优缺点？

（1）水平型微指令并行操作能力强、效率高、灵活性强，垂直型微指令则较差。

（2）水平型微指令执行一条指令的时间短，垂直型微指令执行时间长。

（3）由水平型微指令解释指令的微程序，具有微指令字比较长，但微程序短的特点，而垂直型微指令正好相反。

（4）水平型微指令用户难以掌握，而垂直型微指令与指令比较相似，相对来说比较容易掌握

14 简述 DMAc 操作过程及其优点？

（1）外设发出 DMA 请求；

（2）CPU 响应请求，DMA 控制器从 CPU 接管总线的控制；

（3）由 DMA 控制器执行数据传送操作；

（4）向 CPU 报告 DMA 操作结束。

主要优点是数据数据速度快

15 在计算机系统中，采用总线结构有什么样的优点，在总线的集中式仲裁中，主要有链式查询方式、计数器定时查询方式和独立请求方式，简单说明这三种仲裁方式的优缺点。

引入总线方便计算机系统的设计简单，外部设备便于接入，数据的传输，系统的扩充与更新。

链式查询方式：优点：只用很少几根线就能按一定优先次序实现总线控制，并且这种链式结构很容易扩充设备。缺点：是对询问链的电路故障很敏感，优先级固定；（2 分）

计数器的初值也可用程序来设置，这可以方便地改变优先次序，但这种灵活性是以增加线数为代价的。可方便的改变优先级（2 分）；

独立请求方式：优点是响应时间快，即确定优先响应的设备所花费的时间少。对优先次序的控制也是相当灵活的（2 分）

16.利用你所学知识，简要地论述一下目前计算机是如何提高其执行程序的速度的。

（1）流水线技术；（2）Cache 技术；（3）虚拟存贮器技术；（4）硬件乘法除法器；（5）多核技术；（6）提高 CPU 主频；（7）双端口存贮器；等等。

17.分析并比较 Cache 存储器三种典型的地址映像方式各自的优缺点

全相联映像，优点是访问，冲突率低，只有 Cache 满时才会出现冲突；缺点是地址变换复杂，速度相对慢。

直接映射，优点是地址变换简单；缺点是每块相互对应，不够灵活，利用率不高。

组相联映射，融合了直接映像和全相联映像两种方式，结合了两者的优缺点，具体实现容易，命中率与全相联映射相近。

18.机器进行浮点加法的主要步骤

1. 0 操作数检查
2. 比较阶码大小完成对阶。规定小阶对大阶，为了尽量少损失精度。
3. 尾数进行加法运算，尾数加法运算采用补码运算。
4. 结果规格化并进行舍入操作，当尾数使用二进制补码表示时，应使得数值位最高位与符号位不同
5. 判断溢出，一般使用双符号位判断溢出，当双符号位出现不同时，则为溢出，其最高位为正确的符号位。

19.计算机系统的层次结构

计算机系统的层次结构主要包括 5 级,第 1 级是微程序设计级或逻辑电路级，第 2 级是机器语言级，第 3 级是操作系统级，第 4 级是汇编语言级，第 5 级是高级语言级。

20 Cache 替换策略中的近期最少使用算法；

近期最少使用算法：将近期内长久未被访问过的行换出，为此，每行设置一个计数器，但当 Cache 每命中一次，命中行计数器清零，其他各行计数器增 1，当需要换出时，比较各行计数器的值，将计数器最大的行换出。

21.操作数寻址方式中的堆栈寻址方式；

堆栈寻址方式有寄存器堆栈和存贮器堆栈两种形式，它们都是以先进后出的原理存贮数据，数据的存取都与栈顶地址打交道，为此需要一个隐式或显式的堆栈指示器。

22.解释程序、指令、微程序和微指令的关系

程序是完成逻辑功能的指令集合，在微程序控制方式中，一条指令对应一段微程序，一个微程序是由若干条微指令组成。

23.中断向量地址

中断向量地址：当 CPU 响应中断时，由硬件直接产生一个固定的地址，即向量地址，由向量地址指出每个中断源设备的中断服务程序的入口。

24 微指令三种编码方式的优缺点

- 直接表示法：简单直观，缺点：微指令字较长，控存容量大
- 编码表示法：缩短指令字长度，缺点：增加译码电路
- 混合表示法：上述方法混合使用，综合考虑字长、灵活性等方面要求。

25.简述 CPU 内部的主要寄存器有哪些，至少列举出四种并说明其主要功能

- 通用寄存器组（R）：可用于传输和暂存数据，也可参与算术逻辑运算，并保存运算结果。
- 地址寄存器（AR）：用来保存当前 CPU 所访问的内存单元的地址
- 指令寄存器（IR）：用来保存当前正在执行的一条指令。当执行一条指令时，先把它从内存取到数据寄存器（DR）中，然后再传送至 IR。
- 程序计数器（PC）：用于存放下一条指令所在的单元的地址。

26 简述机器指令和微指令的关系

- 一条机器指令对应一个微程序，这个微程序是有若干条微指令构成的。因此，一条机器指令的功能是若干条微指令组成的序列来实现的。
- 从一般指令的微程序执行流程图可以看出。每个 CPU 周期就对于一条微指令。这就告诉我们怎么设计微程序，也将使得我们进一步体验到机器指令与微指令的关系。
- 机器指令通常存在在主存中，微程序储存在控制存储器中。

27.什么是指令?什么是指令系统?，为什么要引入指令系统?

指令就是要计算机执行某种操作的命令，指令系统是一台计算机中所有机器指令的集合，

名词解释

- 高速缓冲存储器：解决 cpu 与主存速度不匹配，由硬件实现，对所有软件人员透明
- 虚拟存储器：解决容量问题，由硬件和操作系统实现，对应用软件人员透明
- 存取周期：两次访问存储器所允许的最小时间间隔
- 读写时间：从给出读写命令到完成该读写操作所需要的时间
- 程序计数器：在程序执行过程中用来确定下一条指令地址的器件。
- 微程序：由许多条微指令组成用来执行一条机器指令功能的序列。
- 资源相关：指多条指令进入流水线后在同一机器时钟周期内争用一个功能部件所发生的冲突。
- 垂直型微指令：在指令中设置微操作码字段，采用微操作码编译法，由微操作码规定微指令的功能。
- DMA 方式：直接依靠硬件实现主存与外设之间的数据直接传输，传输过程本身不需 CPU 程序干预。

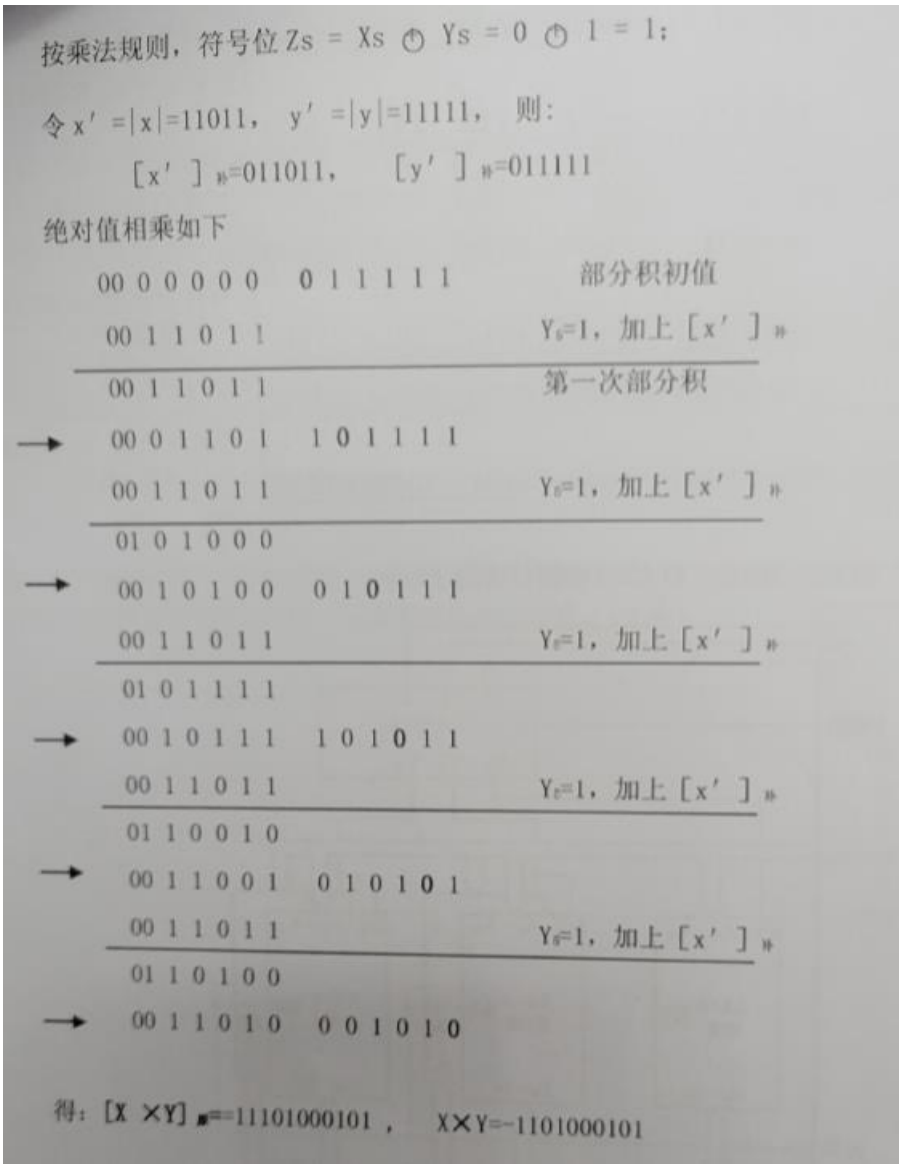
已知 x=101001，y=111，用不恢复余数法求解 x÷y 的值。

x=101001 y=111（10 分）

[x]补=0101001 [y]补＝0111 [-y]补＝1001 (3 分)

	0101001			
+[-y]补	1001			
	1110001	<0	q4 = 0	.....2 分
+[y]补→	00111			
	0001101	>0	q3 = 1	.....2 分
+[-y]补→	111001			
	1111111	<0	q2 =0	
+[y]补 →	0000111			
	0000110	>0	q1=1	.....2 分

X/Y=0101 余数为 110



填空题：

1 计算机系统由　硬件　和　软件　两大部分组成

2 cpu 是　控制器　和　运算器　的总称

3 总线一般由　数据总线 和　地址总线　和 控制总线 ，单总线结构的系统总线由　，　，　。

4 通过 转移　类指令,可设定 pc 的内容

5 访问存储器是指 按地址对存储器进行读或写的操作

6 从设计者的角度看 指令系统　是硬件和软件之间的界面

7 运算器的基本功能是进行　逻辑　运算和 算术 运算

8 浮点数的表示范围取决于　阶码的位数 相对精度取决于尾数　的位数

9 数据总线的数据通路带宽是指 一次能并行传送的数据位数　而把单位时间能传送的数据量　定义为总线的数据传输率

10 对存储器最基本的要求是　　，　　，

11 虚拟存储器是由　主存　和　辅存 组成　对于 应用程序员　是透明的。相联存储器是一种按　内容　　寻址的存储器

12 一般指令由　操作码和 地址码　两部分构成

13 浮点数运算的指令是由　硬件 实现的　浮点数运算时，阶码进行小阶向大阶对齐

14 串行进位的并行加法器的求和时间 随位数的增加而增加

15 我国国标码规定每个汉字使用 2 个字节来表示

16 随机存储器的含义有两点　可按地址随机访问任一单元　访问任何单元时间相同

17 DMA 方式可以实现 cpu 和 I/O 并行工作

18 组合逻辑控制器中，输入信号是指令信号，时序信号，状态信号，输出信号是微命令序列

19 存储器是用来存放数据和程序的　辅助存储器的容量取决于 cpu 的地址宽度

20 控制器可分为硬布线控制器和微程序控制器

21 RISC 的主要特点是 cpu 指令集大大简化

22 相联存储器是按　内容 访问的存储器，用于与主存交换信息的场合，其功能全部由硬件实现

23 DMA 与主存交换数据可采用三种方法：停止 cpu 访问主存，DMA 与 cpu 交替访问主存、周期挪用。

24 利用访存指令与设备交换信息，称为统一编址

25 指令系统中采用不同寻址方式的目的是 缩短指令字长 扩大寻址空间 提高编程灵活性

26 为了缩短指令中某个地址的位数，可采用间接寻址

27 虚拟存储：页式虚拟存储，段式虚拟存储，段页式虚拟存储

28：指令寻址有两种方式，一种是顺序寻址方式，其指令由 pc 给出，列外一种是跳跃寻址方式，其指令地址由指令本身给出

30 指令字长有 半字长 单字长 双字长

31 数据传输方式可采用 串行传输 并行传输 复用传输

32 通道是一个特殊功能的 处理器，它有自己的 程序和数据 专门负责输入输出的控制

通道程序(指令) 主存 中，由通道从主存中取出并执行

33 cpu 的功能 指令控制 操作控制 时间控制 数据加工 中断处理

34 主存储器的性能指标主要是 存储容量 存取时间 存储周期 存储器带宽

35 cpu 通过执行 I/O 指令启动通道

36 一个完善的指令系统应满足四个方面的要求，它们是 完备性、有效性、规整性、兼容性。

37 计算机的软件和硬件在逻辑功能上具有等价性

38 集中式总线仲裁的三种方式是：链式查询方式、计时器定时查询和独立请求。

39 单地址指令中为了实现两个数的算术运算，除地址码指明的一个操作数外，另一个常采用隐含寻址方式

40 在微程序控制器中，确定下一条微指令地址的方式通常有两种方式，分别是计数器方式和多路转移方式

快排

```
void quicksort(int array[], int max, int low, int high)
{
    int i, j;

    if(low < high)
    {
        //int v = array[low];     //中轴元素 v
        i = low + 1;    // array[low]作为基准数(一般第一个)，从 array[begin+1]开始与基准数比较
        j = high;               // array[high]是数组的最后一位
        while(i < j)
        {
            if(array[i] > array[low])  // 如果比较的数组元素大于基准数，则交换位置。
            {
                swap(&array[i], &array[j]); // 交换两个数
                j--;
            }
            else
            {
                i++;                    // 将数组向后移一位，继续与基准数比较。
            }
        }

/* 跳出 while 循环后，i = j。
 * 然后就是分割操作了
 * 这个时候将数组 array 分成两个部分，再将 array[i]与 array[low]进行比较，决定 array[i]的位置。
 * 最后将 array[i]与 array[high]交换，进行两个分割部分的排序！退出条件不满足 i = j
 */

        if(array[i] >= array[low])  // 取等>=
        {
            i--;
        }

        swap(&array[low], &array[i]);

        quicksort(array, max, low, i);
        quicksort(array, max, j, high);
    }
}
```

堆排

```
void adjuctHeap(int i, int *arr, int count) {
    int tmp;
    int max;

    while(i <= count / 2 - 1) {                    // 条件判断检测是否到了叶子节点
        tmp = 2 * i + 2 >= count ? 0 : arr[2 * i + 2];
//总数为偶数，最后一个父节点没有右孩子
        max = arr[2 * i + 1] >= tmp ? 2 * i + 1 : 2 * i + 2;
//max 值：左右孩子中更大的那个孩子节点的下标
        if(arr[max] > arr[i]) {
            tmp = arr[max];
            arr[max] = arr[i];
            arr[i] = tmp;
            i = max;
        }
        else
              break;
    }
}

void heapSort(int *arr, int count) {
    int i;
    int tmp;

    for(i = count / 2 - 1; i > 0; i--) {
     // 停留在构造成功大根堆的前一步。 count / 2 - 1 表示倒数第一个非叶子节点。
        adjuctHeap(i, arr, count);
    }

    while(count > 1) {
        adjuctHeap(0, arr, count);                  // 构造大根堆
        tmp = arr[0];
        arr[0] = arr[count - 1];                        // 交换
        arr[count - 1] = tmp;
        --count;                                  // 末尾往前移动
    }
}
```