2.1

真值：正号和负号分别用“+”和“-”表示，数据位保持二进制值不变的数据表示方法。

机器数（机器码）：数据在机器中的表示形式，是正负符号数码化后的二进制数据。

原码：正数符号用0表示，负数符号用1表示，数值位保持不变

反码：符号位与原码相同，真值为正数时，反码和原码相同；真值为负数时，反码数值位为真值数值位取反

补码： 真值为正数时，补码和原码相同；真值为负数时，对真值数据位从右到左顺序扫描，右起第一个1及其右边的0保持不变，其余各位取反

变形补码：用两个二进制位来表示数字的符号位，其余与补码相同。即“00”表示正，“11”表示负。

移码：用于定点整数的表示，通常用于表示浮点数的阶码（正数的移码符号位为1，负数的为0）

定点数：约定计算机中所有数据的小数点位置固定

浮点数：小数点位置不固定，可以浮动

溢出：在变形补码那里，符号位为01表示正溢出，10表示负溢出

强制类型转换：unsigned修饰的short、int数表示为无符号数

长度相同的整数无符号数和有符号数之间的转换：eg：short转换为unsigned short：补码长度不变，只是补码的最高位用来表示符号的现在是数据位。如果无符号数的最高位为1，则将无符号数取补码，得到的数就是有符号数。
长整数强转成短整数：eg:int转换为short：高位截断，保留低位，
短整数变长整数：eg：short转换为int：符号位扩展
无符号数短整数变无符号数长整数：最高位添0

规格化：将非规格化的数处理成规格化数的过程。规格化数规定尾数用纯小数表示，且真值表示时小数点后第一位不为0（以机器数表示时对小数点后第一位的规定与具体的机器数的形式有关）。
当阶码E全为0，尾数M不全为0时，表示非规格化小数±（0.XX...X）2×2^-126
当阶码E全为0，尾数M全为0时，表示真值±0

左规：当浮点数运算的结果为非规格化时要进行规格化处理，将尾数算数左移一位（小数点向右移动一位），阶码减一

右规：当浮点数运算的结果尾数出现溢出（双符号位为01或者10）时，将尾数算数右移一位（小数点向左移动一位，总长度不变），阶码加一

规格化浮点数的特点：
用原码表示的尾数进行规格化：正数为0.1X...X的形式，负数为1.1X...X的形式
用补码表示的尾数进行规格化：正数为0.1X...X的形式，负数为1.0X...X的形式

浮点数的加减运算：

表示十进制整数的方法：BCD码、BIC码、DPD码
BCD码（8421码）：用4位二进制数来表示1位十进制数中的0~9这10个数码，即二进制表示的十进制数。是有权编码。

二进制浮点数：不能精确表示十进制数，给运算带来很大误差

ACSII码：用于表示字符大写字母之间ACSII码连续小写字母之间ASCII码连续，但是大小写字母之间不连续

汉字机内码：计算机内部存储、处理加工和传输汉字时所用的由0和1符号组成的代码。

字形码（字型码）：汉字的输出码

汉字库：汉字字形码按区位码的顺序排列，以二进制文件形式存放在存储器中，构成汉字字模字库。

码距（海明距离）：一组编码中对应位上数字位不同的最小个数。码距=2，有检错能力，码距≥3，可能还会有纠错能力

奇偶校验：通过检测校验码中1的个数的奇/偶性是否改变来判断数据是否出错的一种数据校验方法，码距为2，不能纠错。当出错位数为偶数个时无法检验出错误
奇校验码：有效信息位和校验位中1的个数为奇数个
偶校验码：有效信息位和校验位中1的个数为偶数个

海明校验（ECC）：是一种基于多重奇校验且具有检测与纠正错误的校验方法。其基本原理是将有效信息按某种规律分成若干组，每组安排一个校验位进行奇偶测试，就能提供多位检错信息，以指出最大可能是哪位出错，从而将其纠正。

校验位k和信息位n之间的关系：2^k≥n+k+1
校验位Pi应该放在2^(i-1)的位置上

循环冗余校验（CRC）：是数据通信领域中最常用的一种具有检测与纠正错误能力差错校验码，基利用生成多项式并基于模2运算建立编码规则。

2.3

（1）为什么计算机中采用二进制进行数据表示和运算?

1、技术实现简单：计算机是由逻辑电路组成，逻辑电路通常只有两个状态，开关的接通与断开，这两种状态正好可以用“1”和“0”表示。2、抗干扰能力强，可靠性高3、运算规则简单4、适合逻辑运算5、易于与其他进制进行转换

（2）相较于奇偶校验，交叉奇偶校验的检错和纠错能力的提高需要付出哪些方面的代价?

交叉奇偶校验原理：将带编码的原始数据信息构造成行列矩阵式结构，同时进行行和列两个方向的奇偶校验。缺点：复杂，占用的内存变大

（3）为什么计算机中采用补码表示带符号的整数?

使用补码,可以将符号位和数值域统一处理;同时,加法和减法也可以统一处理。

（4）浮点数的表示范围和精度分别由什么决定?

表示范围由阶码E的位数决定，精度由尾数的位数决定。

（5）汉字输入码、机内码和字形码在汉字处理过程中各有何作用?
1、输入码：将汉字输入到计算机中。常用的输入码有拼音码、五笔字型码、自然码、表形码、认知码、区位码和电报码等
2、机内码：输入的汉字外码到机器内部都要转换成机内码，才能被存储和进行各种处理。
3、字形码：字形码是汉字的输出码，输出汉字时都采用图形方式，无论汉字的笔画多少，每个汉字都可以写在同样大小的方块中。通常用16×16点阵来显示汉字。

（6）在汉字机内码中如何区分ASCII码和汉字字符?

ASCII码的首位为0，而汉字编码的首位为1。

（7）为什么现代处理器中又开始支持十进制浮点运算?

十进制浮点数可以解决二级制浮点数最大的问题——不能精确表示十进制数。

（8）如何识别浮点数的正负？浮点数能表示的数值范围和数值的精度取决于什么?

尾数的符号位也就是浮点数的符号位S，为1时表示为负数，为0时表示为正数。

（9）浮点数有两个0会带来什么问题?

（10）简述CRC校验码的检错原理，CRC能纠错吗?

1、原理：先在要发送的帧后面附加一个数（即用来校验的校验码，但要注意，这里的数也是二进制序列的，下同），生成一个新帧发送给接收端。当然，这个附加的数不是随意的，它要使所生成的新帧能与发送端和接收端共同选定的某个特定数整除（注意，这里不是直接采用二进制除法，而是采用一种称之为“模2除法”）。到达接收端后，再把接收到的新帧除以（同样采用“模2除法”）这个选定的除数。因为在发送端发送数据帧之前就已通过附加一个数，做了“去余”处理（也就已经能整除了），所以结果应该是没有余数。如果有余数，则表明该帧在传输过程中出现了差错。

2、CRC能纠错。

3.1

全加器：三个输入两个输出，输入端分别为相加数Xi、Yi，低位进位输入Ci。输出端分别是和数Si、高位进位输出Ci+1

半加器：相较于全加器，半加器没有进位输入，其内部逻辑只有一个异或门，用于产生和数一个与门，用于产生进位输出

算术移位：分为算术左移和算术右移。其中算数左移n位相当于乘上2n,执行方法是把原来的数中每一位都向左移动n个位置，左面移出的高位丢弃不要，右面低位空出的位置上全部补0，当符号位发生改变时表明发生了溢出。算术右移时，符号位保持不变，其余各位依次右移，最右边一位移出，将符号位拷贝到左边空出的位，一次移位相当于除2。

逻辑移位：逻辑左移n位的执行方法，是把原来的数中每一位都向左移动n个位置，左面移出的高位丢弃不要，右面低位空出的位置上全部补"0"。 逻辑右移n位的执行方法是把原来数中的每一位都向右移动n个位置，右面移出的低位丢弃不要，左面高位空出的位置上全部补0。

阵列乘法：采用类似手工乘法运算的方法，用大量与门产生手工乘法中的各乘积项，同时将大量一位全加器按照手工乘法算式中需要进行加运算的各相关项的排列方式组成加法器阵列。

恢复余数除法：比较被除数（余数）与除数的大小是用减法实现的。对原码除法而言，由于操作数以绝对值的形式参与运算，因此，相减结果为正（余数的符号位为0）说明够减，商上1；相减结果为负（余数的符号位为1）说明不够减，商上0。

不恢复余数除法：又称加减交替法，是对恢复余数法的改进。不恢复余数法的特点是不够减时不再恢复余数，而根据余数的符号作相应处理就可继续往下运算，因此运算步数固定，控制简单，提高了运算速度。

阵列除法：类似于阵列乘法器的思想，为了加快除法的执行速度，也可以采用阵列除法器来实现除法。为简化运算及阵列除法器的结构，对参加运算的数据进行适当的处理，使其以正数的形式参加运算。

对阶：使阶码相等的过程，对阶的时一般采取小的阶码向大阶码看齐的方式。

保留附加位：对阶过程中尾数右移时通常将暂时保留的最低位移出位

规格化：就是使浮点数的运算结果中，将尾数从非规格化数变成规格化数的过程。根据尾数形式的不同，规格化可分为左移规格化和右移规格化。

3.3

（1）为什么采用并行进位能提高加法器的运算速度?
由于并行进位电路能很快产生各位的进位信号，使得加法器的速度大大提高。

（2）如何判断浮点数运算结果是否发生溢出?
浮点数的溢出是通过接码的是否溢出为判断标志。对于采用双符号位的阶码而言，当双符号位不同时表示浮点数发生溢出，否则则未发生溢出。

（3）如何判断浮点数运算结果是否为规格化数?
如果不是规格化数，如何进行规格化？当尾数采用补码表示时，若运算结果不是11.0××„×或00.1××„×的形式时，结果就不是规格化数。
规格化处理：
当尾数符号为01或10时,需要向右规格化，且只需将尾数右移一位，同时将结果的阶码值加1。 •当尾数运算结果为11.1××„×或00.0××„×时需要左移规格化，而且左移次数不固定，与运算结果的形式有关。
左规的方法是尾数连同符号位一起左移位、和的阶码减1，直到尾数部分出现11.0或00.1的形式为止。

（4）为什么阵列除法器中能用CAS的进位/借位控制端作为上商的控制信号?
阵列除法器利用不恢复余数的除法，当商上1的时候，会产生进位，当商上0时，不产生进位，进位信号与上商信号是相同的，所以可以用CAS的进位/借位控制作为上商的控制信号。

（5）移位运算和乘法及除法运算有何关系?
移位运算是乘除法中的基本运算。

4.1

存取时间：又称为存储器的访问时间，是指启动一次存储器的操作(读或写分别对应存与取)到该操作完成所经历的时间。

存取周期：连续启动两次访问操作之间的最短时间间隔。

存储器带宽：单位时间内存储器所能传输的信息量，常用的单位包括位/秒或字节/秒。

存储单元：保存数据的基本内存单元。根据保存内容的大小，一般可分位存储单元，字存储单元等。存储单元一般应具有存储数据和读写数据的功能，每个单元有一个地址，并按地址访问。

大端存储：数据的高字节存储在低地址中，数据的低字节存储在高地址中（即往往将数字按照原本顺序存储）

小端存储：数据的高字节存储在高地址中，数据的低字节存储在低地址中（即往往将数字反向存储）

Compare大端存储&小端存储：eg：12345678
大端存储：12 34 56 78 小端存储：78 56 34 12

静态存储器(SRAM)：不需要刷新，运行速度快，储存成本高的SRAM，常用作Cache

动态存储器(DRAM)：需要刷新，运行速度较慢，储存成本低的DRAM，常用作主存

刷新：动态存储单元中，为使所存信息能长期保存，在电容电荷泄露完之前定时地补充电荷的过程。

刷新周期：需要多久刷新一次，一般为2ms

字扩展：用多位满足一定要求的存储芯片构成容量更大的存储器。

位扩展：用多片存储芯片构成位数更多的存储器。

多体交叉存储器：由多个存储容量相同，读写速度相同或相近的多个存储模块构成容量更大的存储器，根据存储模块的组织方式不同，又可分为低位交叉和高位交叉两种组织方式。

高速缓冲存储器（Cache）：为缓解快速的CPU与慢速主存之间的速度差异，在CPU和主存之间插入的一至多级速度较快、容量较小的SRAM，起到缓冲作用；使CPU既可以以较快速度存取SRAM中的数据，又不使系统成本上升过高。

相联存储器(CMA)：在计算机系统中，相联存储器主要用于虚拟存储器中存放段表、页表和快表以及高速缓冲存储器中的查找。

时间局部性：指当程序访问一个存储位置时，有很大的可能性程序会在不久的将来再次访问同一位置，程序的循环结构和过程调用就很好地体现了时间局部性。

地址映射：指把主存地址空间按照某种规则映射到Cache的地址空间。

全相联映射：主存数据块可以映射到Cache的任意行中；地址变换时，需查找所有的Cache行。

组相联映射：地址映射时，主存数据块只能映射到索引字段所指向的Cache特定组（其中的行可任选）；地址变换时，需查找的范围也只是索引字段所指向的特定Cache组的所有行。

直接映射：地址映射时，主存数据块只能映射到索引字段所指向的Cache行中保存；地址变换时，需查找的范围也只涉及索引字段所指向的特定Cache行。

命中率：指CPU访问存储系统时，命中Cache的次数占总访问次数的比率。设NC为某程序运行期间命中Cache的次数，Nm为从主存中访问信息的次数，则命中率H定义为：
虚拟存储器：是一种解决主存容量不足的存储管理机制，处于存储系统层次结构中“主存-辅存”存储层次。

页表(慢表)：是一张保存虚拟页号和物理页号(也称实页号)之间对应关系的表格。

TLB(快表)：为了降低虚拟存储器地址转换的开销，根据局部性原理，将页表的一部分装入Cache中，从而减少虚拟地址与物理地址之间转换时访问内存的次数。

页表项：页表的表项，每一个表项由有效位和物理页号两部分构成，用于实现虚拟地址与物理地址之间的转换。

LRU：近期最少使用算法

LFU：最不经常使用算法

Cache一致性：指保存在cache中的数据与保存在主存相关单元的数据相同。
写回法：当CPU对Cache写命中时，只修改Cache的内容不立即写入主存，只当Cache行被替换时才将Cache中的数据写回主存。

写穿法：当Cache写命中时，同时对Cache和主存中同一数据块进行修改；当cache写未命中时，直接向主存写入新的信息，但此时是否将修改过的主存块调入Cache，写穿法有两种选择。一种是把主存中的块调入Cache，在Cache中修改，称为写分配法。另一种是只写入主存，不调入Cache，称为非写分配法。

4.3

（1）算机系统中采用层次化存储体系结构的目的是什么?

采用层次化存储体系的目的包括两方面:其一是解决快速的CPU和慢速的主存之间的速度差异;其二是解决主存容量不够大的问题。

（2）为什么在存储器芯片中设置片选输入端?

由于存储芯片的容量及字长与目标存储器的容量及字长之间可能存在差异，应用存储芯片组织一定容量与字长的存储器时，一般可采用位扩展、字扩展、字位同时扩展等方法来组织。这样就会使用多个存储芯片，从而要设置片选输入端来选择正确的存储芯片来进行操作。

（3）动态MOS存储器为什么要刷新？如何刷新？
动态存储单元中，信息以电荷形式存储在T1或T2管的栅极电容中。由于电容容量小，所存电荷会在一段时间后逐渐泄漏(一般为ms级)，为使所存信息能长期保存，需要在电容电荷泄露完之前定时地补充电荷，这一过程称为刷新。
刷新的方法： ①刷新方式：集中刷新、分散刷新和异步刷新。前者存在CPU死时间；分散刷新由于刷新次数过多，降低了存储器的速度；异步刷新是前两者的折中。 ②刷新按行进行,因此设计刷新电路时需要知道动态存储器的内部行、列结构。 ③刷新地址由刷新地址计数提供。

（4）试述多体交叉存储器的设计思想和实现方法。

多体交叉存储器由多个存储模块构成，这些模块的容量和存取速度相同，具有各自独立的地址寄存器、地址译码器、驱动电路和读写控制电路。根据对多各模块编址方式的不同，又可分为高位多体交叉和低位多体交叉两种方式。
(1)高位交叉：按存储器地址的高位地址划分模块，同一存储体内的地址是连续的。当多个目标同时访问

存储器时(如CPU和DMA设备同时访问存储器)，如果访问的地方范围处于不同的存储芯片，则提供并行访问。

(2)低位交叉：按存储器地址的低位地址划分模块，同一存储体内的地址不相邻，相邻地址处在不同存储体中。CPU可同时启动多个存储体，并进行并行访问。

（5）为什么说Cache对程序员是透明的?

因为在程序员看来，数据是在内存和辅存之间进行交换的，程序员感觉不到中间层cache 的存在。

（6）直接映射方式下为什么不需要使用替换算法?

因为在直接映射方式中，一个内存块只能固定的映射到cache中的特定行，当有新的主存块调入时，cache特定行中的内容必须调出,因此不需要替换算法去选择替换掉哪一块。

（7）为什么要考虑cache的一致性?

正常情况下,cache中的数据是主存数据的副本,当两者不一致时可能导致程序结果不正确,因此,必须考虑并设法保证Cache的一致性。

（8）替换算法有哪几种？他们各有何优缺点?
① 先进先出算法(FIFO) 基本思想：按照数据块进入Cache的先后决定替换的顺序，即在需要进行替换时，选择最先被调入Cache中的块作为替换块。这种方法要求为每块记录它们进入Cache的先后次序。
优点：FIFO算法系统开销较小。
缺点：是不考虑程序访问的局部性，可能会把一些需要经常使用的块（如循环程序块）也作为最早进入Cache的块而替换掉，因此，可能导致Cache的命中率不高。
② 近期最少使用(LRU)算法 基本思想：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，cache每命中一次，命中行计数器清零，其它各行计数器增1，因此它是未访问次数计数器。当需要替换时，比较各特定行的计数值，将计数值最大的行换出。
优点：这种算法显然保护了刚调入Cache的新数据，符合cache工作原理，因而使cache有较高的命中率。LRU算法硬件实现简单
③ 最不经常使用(LFU)算法 基本思想：将一段时间内被访次数最少的那行数据换出。为此，每行设置一个计数器，新调入行的数据从0开始计数，每访问一次被访行的计数器增1。当需要替换时，对这些特定行的计数值进行比较，将计数值最小的行换出。
缺点：一段期间访问情况不能严格反映近期访问情况。例如特定行中的A、B两行，A行在期间的前期多次被访问而后期未被访问，但累积计数值很大，B行是前期不常用而后期却正被频繁访问，但可能由于累积计数小于A行而被LFU算法换出了。
④ 随机替换算法基本思想：需要进行替换时，从特定的行位置中随机地选取一行进行替换。 优点：硬件实现最容易，而且速度也比前几种策略快。
缺点：随意换出的数据很可能马上又要用，从而降低命中率和cache工作效率。但这个负面效应随着cache容量增大会减少，模拟研究表明随机替换策略的功效只是稍逊于LFU和LRU。

5.1

指令：控制计算机执行某种操作(如加、减、传送、转移等操作)的命令。

指令系统：一台计算机中所有指令的集合。

操作码：指令中用于控制指令操作性质的字段。

扩展地址码：将指令的操作码字段向不用的地址码字段扩展，从而在指令长度不变的情况下支持更多的指令。

地址码：指令中用于参与指令操作的操作数的地址或偏移量地址的字段。

寻址方式：寻找指令或操作数有效地址的方法。

立即寻址：形式地址A就是操作数本身（其特征是带有 # 符号）
优点：指令执行阶段不访问主存，指令执行时间最短

直接寻址：形式地址A就是操作数的真是我地址EA
优点：简单，指令执行阶段仅访问一次主存，不需专门计算机操作数的地址

间接寻址：指令的地址字段给出的形式地址不是操作数的真正地址，而是操作数有效地址所在的存储单元地址，也就是操作数地址的地址，即EA=（A）
优点：可扩大寻址范围（有效地址EA的位大于形式地址A的位数）。便于编制程序（用间接寻址可以方便的完成子程序返回）。

寄存器寻址：在指令字中直接给出操作数所在的寄存器编号，即EA=Ri，其操作数在由Ri所指的寄存器内。
优点：指令在执行阶段不访问主存，只访问寄存器；指令字短且执行速度快，支持向量、矩阵运算。

寄存器间接寻址：寄存器Ri中给出的不是一个操作数，而是操作数所在的主存单元的地址，即EA==（Ri）
特点：与一般间接寻址相比速度更快，但令的执行阶段需要访问主存（因为操作数在主存中）。

相对寻址：把程序计时器PC的内容加上指令格式找那个的形式地址A而形成操作数的有效地址，即EA=（PC）+A，其中A相当于是对于当前指令地址的位移量，可正可负，补码表示。
优点：操作数的地址不是固定的，他随着PC值的变化而变化，并且与指令地址之间总是相差一个固定值，一次便于程序浮动。
相对寻址广泛用于转移指令。

堆栈寻址：操作数存放在堆栈中，隐含使用堆栈指针（SP）作为操作数地址。堆栈是存储器（货专用寄存器组）中一块特定的按"后进先出"原则管理的存储区，改存储区中被读、写单元的地址用一个特定的寄存器给出的，该寄存器称为堆栈指针（SP）
硬堆栈：由寄存器实现
软堆栈：由主存实现

程序计数器PC：程序计数器是用于存放下一条指令所在单元的地址的寄存器。

有效地址：表示操作数所在主存单元的物理地址。

存储器堆栈：以先进后出的方式存储数据，在内存空间开辟堆栈区，该类堆栈容量大，速度慢，栈顶移动而堆栈中的数据不动。

寄存器堆栈：以先进后出的方式存储数据，利用寄存器开辟的堆栈区，该类堆栈容量小，速度块，栈顶不动，出栈和入栈操作设计栈内所有数据的移动。

基址寄存器：基址寻址方式下用于存放基地址的寄存器。

变址寄存器：变址寻址方式下，用于存放变化的地址的寄存器。

转子指令：子程序调用指令。

CISC：CISC是复杂指令系统计算机的简称，这类计算机指令系统复杂，寻址方式种类较多，指令执行效率低。

RISC：RISC是精简指令集计算机的简称，这类计算机指令系统简单，寻址方式种类少，指令执行效率高。

5.3

(1)什么叫指令？什么叫指令系统?
指令是指控制计算机执行某种操作(如加、减、传送、转移等操作)的命令，而一台计算机中所有指令的集合称为该计算机的指令系统。

(2)计算机中为什么要设置多种操作数寻址方式?
这是为了在效率和方便性以及寻址空间大小保持平衡。
用于快速访问的寻址方式：立即数寻址、寄存器寻址等
扩大寻址范围的寻址方式：间接寻址、寄存器间接寻址、基址寻址等
便于程序设计灵活性的寻址方式：变址寻址、相对寻址、直接寻址等
既扩大寻址范围，又由利于指令执行速度提高的寻址方式：寄存器间接寻址
另外，多种复合寻址寻址方式使得寻址更加灵活。

(3)操作数寻址方式在指令中如何表示?
由于不同指令可能采用不同的寻址方式获得操作数,因此,一般情况下，指令的格式会进一步细分出寻址方式字段。下图所示的为包含寻址方式字段的单地址指令结构。

其中，OP为操作码，I为寻址方式特征码。D为形式地址，或称偏移量。寻址过程就是把I和D的不同组合变换成有效地址的过程。I与操作数寻址方式相关。

(4)基址寻址和变址寻址的作用是什么?分析它们的异同点.
基址寻址面向系统，主要用于程序的重定位和扩展寻址空间。变址寻址是面向用户的，主要解决程序循环问题。
相同点：在形式上以及计算操作数的有效地址的方法上，变址寻址和基址寻址中是相似的，都是把个寄存器的内容加上指令字中的形式地址而形成操作数有有效地址。
不同点：两者有着不同的用途。
首先，在采用了基址寻址的计算机系统中，基址是不变的，程序中的所有地址都是相对于基地址来变化的。而对于变址寻址来说则相反，指令中的地址字段的形式地址给出的是一个存储器地址基准，变址寄存器X中存放的是相对于该基准地址的偏移量。不同的变址寄存器给出的不同的单元。
第二，在基址寻址中，偏移量位数较短，而在变址寻址中，偏移量位数足以表示整个存储空间。
第三，基址寻址主要是解决程序逻辑空间与存储器物理空间的无关性，而变址寻址主要是为了可以编写出高效访问一片存储空间的程序。

(5)RISC处理器有何特点?
RISC具有如下特点：使用等长指令、寻址方式少且简单、只有取数和存数指令访问存储器、指令数量和指令格式少于、指令功能简单、CPU内部设置了大量的寄存器、控制器多采用硬布线方式、大多数指令可在一个时钟周期内完成、支持指令流水并强调指令流水的优化使用。

(6)比较定长指令与变长指令的优缺点。
定长指令的优点：定长指令具有结构规整，有利于简化硬件，尤其是指令译码部件的设计。
定长指令的缺点：定长指令平均长度长、容易出现冗余码点和指令不易扩展等不足。
变长指令的优点：变字长指令结构灵活，能充分利用指令中的每一位，所以指令码点冗于少，指令的平均长度短，易于扩展。
变长指令的缺点：变长指令的格式不规整，不同指令的取指时间可能不同，导致控制复杂。

(7)指令的地址码与指令中的地址码含义有何不同?
指令的地址码通常指定参与操作的操作数的地址。指令中的地址码字段的作用随指令类型和寻址方式的不同而不同，它可能作为一个操作数、也可能是操作数的地址（包括操作数所在的主存地址、寄存器编号或外部设备端口地址）、也可能是一个用于计算地址的偏移量。

6.1

指令周期：取指令并执行一条指令所需要的时间，一般由若干个机器周期组成，包括从取指令、分析指令到执行完所需的全部时间。一般情况下，一条指令所需的最短时间为两个机器周期：取指周期和执行周期
指令周期流程：

取指周期：根据PC中的内容取出指令代码并存放在IR中
间址周期：根据IR中指令地址码取操作数有效地址
执行周期：根据指令字的操作码和操作数进行相应的操作
中断周期：保存断点，送中断向量，处理中断请求
四个工作周期都有CPU访存操作，只是访存的目的不同。取指周期是为了取指令，间指周期是为了取有效地址，执行周期是为了取操作数，中断周期是为了保存程序断点。
MAR：主存地址寄存器
MDR：主存数据寄存器
IR：指令寄存器

数据通路：数据在功能部件之间传送的路径。
数据通路的基本结结构：
1、CPU内部单总线方式
2、CPU内部多总线方式
3、专用数据通路方式

机器周期：一般将一个指令周期划分为若干机器周期，每个机器周期完成一个基本操作，如取指周期、取数周期、执行周期、中断周期等。一个机器周期可由若干节拍组成，一个节拍可以安排一个或几个工作脉冲。

节拍（时钟周期）：是计算机操作的最小时间单位。一个节拍对应一个电位信号，控制一个或几个微操作的执行

同步控制：选取部件中最长的操作时间作为统一的时间间隔标准，使所有部件都在这个时间间隔内启动并完成操作。

异步控制：系统不设立统一的时间间隔标准(基准时钟除外)，各部件按各自的时钟工作，分别实现各自的时序控制，时间衔接通过应答通讯方式(又称握手方式)实现。

单周期处理器：所有指令在一个时钟周期内完成的处理器。

多周期处理器：每条指令的执行分成多个阶段，每个时钟周期完成一个阶段的工作。
时序发生器：

硬布线控制器：又称为组合逻辑控制器，指令执行所需要的控制信号直接由逻辑门电路和触发器等构成的电路产生，与微程序控制器相比，具有结构复杂但速度快的特点。

微命令：即控制部件通过控制线向执行部件发出各种控制命令 。在微指令的控制字段中，每一位代表一个微命令。

微指令：由微指令产生的一组实现一定微操作功能的微命令的组合。

微程序：实现一条指令功能的若干条微指令的集合。

微操作：执行部件收到微命令后所进行的操作。

公操作：在一条指令执行完毕后，CPU所开始进行的一些操作，这些操作主要是CPU对外围设备请求的处理，如中断处理、通道处理等。

相容性微命令：能同时并行执行的微命令。

互斥性微命令：不能并行执行的微操作。

取指微程序：取指令阶段属于公操作。取指令的公操作通常由一个取指微程序来完成。机器开始运行时，自动将“取指微程序”的入口地址送入微地址寄存器。根据入口地址来从控存中读出微指令送到微指令寄存器。自此取指微程序开始执行。当取指微程序执行完毕后，机器指令已被送入指令寄存器（IR）中。

微程序控制器：采用微程序设计方法设计的控制器。指令执行过程中所需要的所有控制信号以微指令的方式存在在控制存储器中，指令执行时，逐条读出微指令，以产生执行执行过程中所需要的控制信号。

微指令周期：指在串行方式的微程序控制器中，微指令周期等于读出微指令的时间加上执行该条微指令的时间。

控制存储器：微程序控制器中用于存放解释所有指令微程序的存储器。

水平型微指令：一次能定义并执行多个并行操作微命令的微指令。

垂直型微指令：微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。在一条垂直型微指令中，一般只能完成一个操作，控制了、一两个信息传送通路，因此微指令的并行操作能力低，效率低。

中断响应微程序：

6.3

1、中央处理器的基本功能是什么?从实现其功能的角度分析，它应由哪些部件组成?
1、指令控制：完成取指令、分析指令和执行指令的操作，即嘘的顺序控制。
2、操作控制：一条指令的功能往往是由若干操作信号的组合来实现的。CPU管理并产生由内存取出的每条指令的操作信号，吧各种操作信号送往相应的部件，从而控制这些部件按指令的要求进行动作。
3、时间控制：对各种操作加以时间上的控制。时间控制要为每条指令按时间顺序提供应有的控制信号
4、数据加工：对数据进行算数和逻辑运算。
5、中断处理：对计算机运行过程中出现的异常情况和特殊请求进行处理。
组成：中央处理器主要由控制器和运算器两部分构成。控制器的主要功能包括：取指令、计算下一条指令的地址、对指令译码、产生相应的操作控制信号、控制指令执行的步骤和数据流动的方向。运算器是执行部件，由算术逻辑单元和各种寄存器组成。

2、CPU内部有哪些寄存器?它们的功能分别是什么?哪些是程序员可见的？哪些是必须的?
(1) 指令寄存器(IR)：IR用于保存指令。从主存储器取出的指令存放在IR中，直到新的指令从主存中取出为止。
(2) 程序计数器(PC) ： PC保存将要执行的指令地址，故又称指令地址寄存器。CPU取指令时，将PC的内容送到主存地址寄存器，然后修改PC的值形成下一条将要执行的指令地址
(3) 地址寄存器(AR)：AR用来保存当前CPU所要访问的主存单元地址，无论CPU是取指令还是存取数据，都必须先将要访问的主存单元地址送AR，直到读/写操作完成。
(4) 通用寄存器组(GR)：通用的含义是指寄存器的功能有多种用途，GR可作为ALU的累加器、变址寄存器、地址指针、指令计数器、数据缓冲器，用于存放操作数(包括源操作数、目的操作数及中间结果)和各种地址信息等。
(5) 数据缓冲寄存器(DR):DR作为CPU和主存之间的数据缓冲寄存器用于存放操作数、运算结果或中间结果，以减少访问主存的次数；也可存放从主存中读出的数据，或准备写入主存的数据。
(6) 程序状态字寄存器(PSW)
PSW用于保存由算术运算指令、逻辑运算指令、测试结果等建立的各种条件标志。常见的状态信息包括进位标志©、溢出标志(V)、结果为负数标志(S)及结果为零标志(Z)等。
可见寄存器：通用寄存器组，程序状态字寄存器，程序计数器pc

3、什么是取指周期？取指周期内应完成哪些操作?
取指周期就是从开始取指令到取指令完成所需要的时间。在取指周期内，CPU根据PC中的内容取出指令代码并存放在IR中。

4、计算机为什么要设置时序系统?
指令执行过程中的所有操作必须按照一定的次序完成，而且这些操作持续的时间也有严格的限制，因此，在计算机系统中需要设置时序系统，对指令执行过程中的所有控制信号进行时间控制，以保证指令功能的正确实现。

5、简述传统三级时序和现代时序的差异。

6、比较单周期MIPS处理器与多周期MIPS处理器的差异。

7、组合逻辑控制器与微程序控制器各有什么特点?
硬布线控制器又称为组合逻辑控制器，这种控制器中的控制信号直接由各种类型的逻辑门电路和触发器等构成，与微程序控制器相比，具有结构复杂但速度快的特点。
微程序控制器的设计采用了存储技术和程序设计技术，使复杂的控制逻辑得到简化。通过过读出存放在微程序控制器中微指令产生指令执行过程中所需要的控制信号，所以，与硬布线控制器相比，微程序控制器的速度较慢。

8、说明程序与微程序，指令与微指令的异同
程序则是为了完成某一应用功能所编写的指令（包括机器语言指令或高级语言指令）集合，属于高级语言级别，对用户的透明性好，运行时存放在计算机的主存中。微程序是多条微指令系列的集合,用于实现指令的功能,属于机器指令级别，对用于的透明性不强，存放在CPU内的控制存储器中；
指令是指挥计算机执行某种功能的命令，是构成程序的基本单位，由操作码和地址字段构成；而微指令则用于微程序控制器中产生指令执行过程中所需要的微命令，是构成微程序的基本单位，由操作控制字段、判别测试字段和下地址字段等组成。

9、命令有哪几种编码方法?它们是如何实现的?
微指令的微命令有三种编码方法，分别是直接表示方法、字段直接译码法和混合控制法。
直接表示法的基本思想是：将微指令操作控制字段的每个二进制位定义为一个微命令，用“1”或“0"表示相应的微命令的“有”或“无”。
字段直接译码法的基本思想是：将微指令格式中的操作控制字段分成若干组，每组中包含若干个互斥性微命令，将相容性的微命令安排在不同组。
混合控制法：将直接表示法与字段直接译码法混合使用，以便在微指令字长、并行性及执行速度和灵活性等方面进行折衷，发挥它们的共同优点。

10、简述微程序控制器和硬布线控制器的设计方法?
1、微程序控制器设计方法：
1)分析指令执行的数据通路，列出每条指令在所有寻址方式下的执行操作流程和每一步所需要的控制信号；
2)对指令的操作流程进行细化，将每条指令的每个微操作分配到具体的机器周期的各个时间节拍信号上；
(3)设计微指令格式、微命令编码方法和程序组织方式；
(4)编制每条指令的微程序；并按照所设计的微程序组织方式存放到控存中；
(5)对微命令进行同步控制，并送数据通路的相关控制点。
2、硬布线控制器设计方法：
1)分析指令执行的数据通路，列出每条指令在所有寻址方式下的执行操作流程和每一步所需要的控制信号；
2)对指令的操作流程进行细化，将每条指令的每个微操作分配到具体的机器周期的各个时间节拍信号上，即对操作控制信号进行同步控制。
3)对每一个控制信号进行逻辑综合，得到每个控制信号的逻辑表达式。
4)最后采用逻辑门或PLA或ROM实现逻辑表达式的功能，各控制信号送数据通路的相关控制点。

11、简述CPU内部异常和外部中断的区别。

12、简述异常与中断处理的一般流程。