## Chapter 1-3

2.如何理解计算机系统的层次结构？

从计算机系统的层次结构来看，它通常可以有5个以上的层次，在每一个层次上都能进行程序设计。从下到上依次为1.微程序机器级，微指令由硬件直接执行；2.传统机器级，用微程序解释机器指令；3.操作系统级，一般用机器语言程序解释作业控制语句；4.汇编语言机器级，由汇编程序支持和执行；5.高级语言级，采用高级语言，有各种高级语言编译程序支持和执行。还可以有6.应用语言机器级，采用各种面向问题的应用语言。

4. 如何理解计算机组成和计算机体系结构？

答：计算机体系结构是指那些能够被程序员所见到的计算机系统的属性，如指令系统、数据类型、寻址技术组成及I/O机理等。计算机组成是指如何实现计算机体系结构所体现的属性，包含对程序员透明的硬件细节，如组成计算机系统的各个功能部件的结构和功能，及相互连接方法等。

5. 冯•诺依曼计算机的特点是什么？

解：冯•诺依曼计算机的特点是：P8

* 计算机由运算器、控制器、存储器、输入设备、输出设备五大部件组成；
* 指令和数据以同同等地位存放于存储器内，并可以按地址访问；
* 指令和数据均用二进制表示；
* 指令由操作码、地址码两大部分组成，操作码用来表示操作的性质，地址码用来表示操作数在存储器中的位置；
* 指令在存储器中顺序存放，通常自动顺序取出执行；
* 机器以运算器为中心（原始冯•诺依曼机）。

6、参考图1.9

主存储器作用是存放程序和数据，可以直接与CPU交换数据。

算术逻辑单元（ALU），完成算术和逻辑运算。

控制单元（CU），解释存储器中的指令，并发出各种操作来执行指令。

I/O设备，受控制单元（CU）的控制，完成输入、输出操作。

技术指标：

存储器容量：存储器的度量指标，通常用两种方式表示：

存储容量=存储单元个数\*存储字长

存储容量也可以表示为字节数

ALU的主要技术指标是机器字长，即CPU一次所能处理的数据的位数。

计算机的硬件指标还有运算速度，例如吉普森法，每秒钟百万条指令的条数（MIPS）等

7. 解释下列概念：

主机、CPU、主存、存储单元、存储元件、存储基元、存储元、存储字、存储字长、存储容量、机器字长、指令字长。

解：P9-10

 主机：是计算机硬件的主体部分，由CPU和主存储器MM合称为主机。

 CPU：中央处理器，是计算机硬件的核心部件，由运算器和控制器组成；（早期的运算器和控制器不在同一芯片上，现在的CPU内除含有运算器和控制器外还集成了CACHE）。

 主存：计算机中存放正在运行的程序和数据的存储器，为计算机的主要工作存储器，可随机存取；由存储体、各种逻辑部件及控制电路组成。

 存储单元：可存放一个机器字并具有特定存储地址的存储单位。

 存储元件：存储一位二进制信息的物理元件，是存储器中最小的存储单位，又叫存储基元或存储元，不能单独存取。

 存储字：一个存储单元所存二进制代码的逻辑单位。

 存储字长：一个存储单元所存储的二进制代码的总位数。

 存储容量：存储器中可存二进制代码的总量；（通常主、辅存容量分开描述）。

 机器字长：指CPU一次能处理的二进制数据的位数，通常与CPU的寄存器位数有关。

 指令字长：机器指令中二进制代码的总位数。

8. 解释下列英文缩写的中文含义：

CPU、PC、IR、CU、ALU、ACC、MQ、X、MAR、MDR、I/O、MIPS、CPI、FLOPS

解：全面的回答应分英文全称、中文名、功能三部分。

CPU：Central Processing Unit，中央处理机（器），是计算机硬件的核心部件，主要由运算器和控制器组成。

PC：Program Counter，程序计数器，其功能是存放当前欲执行指令的地址，并可自动计数形成下一条指令地址。

IR：Instruction Register，指令寄存器，其功能是存放当前正在执行的指令。

CU：Control Unit，控制单元（部件），为控制器的核心部件，其功能是产生微操作命令序列。

ALU：Arithmetic Logic Unit，算术逻辑运算单元，为运算器的核心部件，其功能是进行算术、逻辑运算。

ACC：Accumulator，累加器，是运算器中既能存放运算前的操作数，又能存放运算结果的寄存器。

MQ：Multiplier-Quotient Register，乘商寄存器，乘法运算时存放乘数、除法时存放商的寄存器。

X：此字母没有专指的缩写含义，可以用作任一部件名，在此表示操作数寄存器，即运算器中工作寄存器之一，用来存放操作数；

MAR：Memory Address Register，存储器地址寄存器，在主存中用来存放欲访问的存储单元的地址。

MDR：Memory Data Register，存储器数据缓冲寄存器，在主存中用来存放从某单元读出、或要写入某存储单元的数据。

I/O：Input/Output equipment，输入/输出设备，为输入设备和输出设备的总称，用于计算机内部和外界信息的转换与传送。

MIPS：Million Instruction Per Second，每秒执行百万条指令数，为计算机运算速度指标的一种计量单位。

9. 画出主机框图，分别以存数指令“STA M”和加法指令“ADD M”（M均为主存地址）为例，在图中按序标出完成该指令（包括取指令阶段）的信息流程（如→①）。假设主存容量为256M\*32位，在指令字长、存储字长、机器字长相等的条件下，指出图中各寄存器的位数。

解：主机框图如P13图1.11所示。

（1）STA M指令：PC→MAR，MAR→MM，MM→MDR，MDR→IR，

OP(IR)→CU，Ad(IR)→MAR，ACC→MDR，MAR→MM，WR

（2）ADD M指令：PC→MAR，MAR→MM，MM→MDR，MDR→IR，

OP(IR)→CU，Ad(IR)→MAR，RD，MM→MDR，MDR→X，ADD，ALU→ACC

假设主存容量256M\*32位，在指令字长、存储字长、机器字长相等的条件下，ACC、X、IR、MDR寄存器均为32位，PC和MAR寄存器均为28位。

2. 总线如何分类？什么是系统总线？系统总线又分为几类，它们各有何作用，是单向的，还是双向的，它们与机器字长、存储字长、存储单元有何关系？

答：按照连接部件的不同，总线可以分为片内总线、系统总线和通信总线。

系统总线是连接CPU、主存、I/O各部件之间的信息传输线。

系统总线按照传输信息不同又分为地址线、数据线和控制线。地址线是单向的，其根数越多，寻址空间越大，即CPU能访问的存储单元的个数越多；数据线是双向的，其根数与存储字长相同，是机器字长的整数倍。

4. 为什么要设置总线判优控制？常见的集中式总线控制有几种？各有何特点？哪种方式响应时间最快？哪种方式对电路故障最敏感？

答：总线判优控制解决多个部件同时申请总线时的使用权分配问题；

常见的集中式总线控制有三种：链式查询、计数器定时查询、独立请求；

特点：链式查询方式连线简单，易于扩充，对电路故障最敏感；计数器定时查询方式优先级设置较灵活，对故障不敏感，连线及控制过程较复杂；独立请求方式速度最快，但硬件器件用量大，连线多，成本较高。

5. 解释下列概念：总线宽度、总线带宽、总线复用、总线的主设备（或主模块）、总线的从设备（或从模块）、总线的传输周期和总线的通信控制。

答：P46。

总线宽度：通常指数据总线的根数；

总线带宽：总线的数据传输率，指单位时间内总线上传输数据的位数；

总线复用：指同一条信号线可以分时传输不同的信号。

总线的主设备（主模块）：指一次总线传输期间，拥有总线控制权的设备（模块）；

总线的从设备（从模块）：指一次总线传输期间，配合主设备完成数据传输的设备（模块），它只能被动接受主设备发来的命令；

总线的传输周期：指总线完成一次完整而可靠的传输所需时间；

总线的通信控制：指总线传送过程中双方的时间配合方式。

6. 试比较同步通信和异步通信。

答：同步通信：指由统一时钟控制的通信，控制方式简单，灵活性差，当系统中各部件工作速度差异较大时，总线工作效率明显下降。适合于速度差别不大的场合。

异步通信：指没有统一时钟控制的通信，部件间采用应答方式进行联系，控制方式较同步复杂，灵活性高，当系统中各部件工作速度差异较大时，有利于提高总线工作效率。

15.

Tbus=4\*Tclk=4\*1/66M

数据传输率=32bit/Tbus=32bit/4\*66M=66MBps=528Mbps

提高传输率可以提高时钟频率，也可以减少时钟周期数。

16. 在异步串行传送系统中，字符格式为：1个起始位、8个数据位、1个校验位、2个终止位。若要求每秒传送120个字符，试求传送的波特率和比特率。

解：一帧包含：1+8+1+2=12位

故波特率为：（1+8+1+2）\*120=1440bps

比特率为：8\*120=960bps

## Chapter 4

1.解释概念：主存、辅存、Cache、RAM、SRAM、DRAM、ROM、PROM、EPROM、EEPROM、CDROM、Flash Memory。

答：主存：主存储器，用于存放正在执行的程序和数据。CPU可以直接进行随机读写，访问速度较高。

辅存：辅助存储器，用于存放当前暂不执行的程序和数据，以及一些需要永久保存的信息。

Cache：高速缓冲存储器，介于CPU和主存之间，用于解决CPU和主存之间速度不匹配问题。

RAM：半导体随机存取存储器，主要用作计算机中的主存。

SRAM：静态半导体随机存取存储器。

DRAM：动态半导体随机存取存储器。

ROM：掩膜式半导体只读存储器。由芯片制造商在制造时写入内容，以后只能读出而不能写入。

PROM：可编程只读存储器，由用户根据需要确定写入内容，只能写入一次。

EPROM：紫外线擦写可编程只读存储器。需要修改内容时，现将其全部内容擦除，然后再编程。擦除依靠紫外线使浮动栅极上的电荷泄露而实现。

EEPROM：电擦写可编程只读存储器。

CDROM：只读型光盘。

Flash Memory：闪速存储器。或称快擦型存储器。

4.3

存储系统层次结构主要体现在cache-主存，主存-辅存这两个层次。

cache-主存层次主要是为了解决CPU与主存速度不匹配的问题。。。。

主存-辅存层次主要是解决存储系统容量问题。。。。

cache-主存之间的数据调动是由硬件自动完成，对程序员是透明的。

主存-辅存之间的数据调动是由硬件和操作系统共同完成的。

4.8

静态RAM和动态RAM都属于随机存储器，即在程序的执行过程中，即可以读出信息又可以写入信息。但是静态存储器靠触发器原理存储信息，只要电源不掉电，信息会不会丢失；动态RAM靠电容存储器存储信息，即使电源不掉电，由于电容要放电，信息也会丢失，故需再生。

14. 某8位微型机地址码为18位，若使用4K×4位的RAM芯片组成模块板结构的存储器，试问：

（1）该机所允许的最大主存空间是多少？

（2）若每个模块板为32K×8位，共需几个模块板？

（3）每个模块板内共有几片RAM芯片？

（4）共有多少片RAM？

（5）CPU如何选择各模块板？

解：（1）该机所允许的最大主存空间是：218 × 8位 = 256K×8位 = 256KB

（2）模块板总数 = 256K×8 / 32K×8 = 8块

（3）板内片数 = 32K×8位 / 4K×4位 = 8×2 = 16片

（4）总片数 = 16片×8 = 128片

（5）CPU通过最高3位地址译码输出选择模板，次高3位地址译码输出选择芯片。地址格式分配如下：



15. 设CPU共有16根地址线，8根数据线，并用（低电平有效）作访存控制信号，作读写命令信号（高电平为读，低电平为写）。现有下列存储芯片：ROM（2K×8位，4K×4位，8K×8位），RAM（1K×4位，2K×8位，4K×8位），及74138译码器和其他门电路（门电路自定）。试从上述规格中选用合适芯片，画出CPU和存储芯片的连接图。要求：

（1）最小4K地址为系统程序区，4096~16383地址范围为用户程序区。

（2）指出选用的存储芯片类型及数量。

（3）详细画出片选逻辑。

解：（1）地址空间分配图：

系统程序区（ROM共4KB）：0000H-0FFFH

用户程序区（RAM共12KB）：1000H-3FFFH

  （2）选片：ROM：选择4K×4位芯片2片，位并联

              RAM：选择4K×8位芯片3片，字串联(RAM1地址范围为:1000H-1FFFH,RAM2地址范围为2000H-2FFFH, RAM3地址范围为:3000H-3FFFH)

  （3）各芯片二进制地址分配如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| ROM1,2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| RAM1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| RAM2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| RAM3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

CPU和存储器连接逻辑图及片选逻辑如下图(3)所示：



28. 设主存容量为256K字，Cache容量为2K字，块长为4。

（1）设计Cache地址格式，Cache中可装入多少块数据？

（2）在直接映射方式下，设计主存地址格式。

（3）在四路组相联映射方式下，设计主存地址格式。

（4）在全相联映射方式下，设计主存地址格式。

（5）若存储字长为32位，存储器按字节寻址，写出上述三种映射方式下主存的地址格式。

解：（1）Cache容量为2K字，块长为4，Cache共有2K/4=211/22=29=512块，

Cache字地址9位，字块内地址为2位

因此，Cache地址格式设计如下：

|  |  |
| --- | --- |
| Cache字块地址（9位） | 字块内地址（2位） |

（2）主存容量为256K字=218字，主存地址共18位，共分256K/4=216块，

主存字块标记为18-9-2=7位。

直接映射方式下主存地址格式如下：

|  |  |  |
| --- | --- | --- |
| 主存字块标记（7位） | Cache字块地址（9位） | 字块内地址（2位） |

（3）根据四路组相联的条件，一组内共有4块，得Cache共分为512/4=128=27组，

主存字块标记为18-7-2=9位，主存地址格式设计如下：

|  |  |  |
| --- | --- | --- |
| 主存字块标记（9位） | 组地址（7位） | 字块内地址（2位） |

（4）在全相联映射方式下，主存字块标记为18-2=16位，其地址格式如下：

|  |  |
| --- | --- |
| 主存字块标记（16位） | 字块内地址（2位） |

（5）若存储字长为32位，存储器按字节寻址，则主存容量为256K\*32/8=220B，

Cache容量为2K\*32/8=213B，块长为4\*32/8=16B=24B，字块内地址为4位，

在直接映射方式下，主存字块标记为20-9-4=7位，主存地址格式为：

|  |  |  |
| --- | --- | --- |
| 主存字块标记（7位） | Cache字块地址（9位） | 字块内地址（4位） |

在四路组相联映射方式下，主存字块标记为20-7-4=9位，主存地址格式为：

|  |  |  |
| --- | --- | --- |
| 主存字块标记（9位） | 组地址（7位） | 字块内地址（4位） |

在全相联映射方式下，主存字块标记为20-4=16位，主存地址格式为：

|  |  |
| --- | --- |
| 主存字块标记（16位） | 字块内地址（4位） |

32. 设某机主存容量为4MB，Cache容量为16KB，每字块有8个字，每字32位，设计一个四路组相联映射（即Cache每组内共有4个字块）的Cache组织。

（1）画出主存地址字段中各段的位数。

（2）设Cache的初态为空，CPU依次从主存第0，1，2，…，89号单元读出90个字（主存一次读出一个字），并重复按此次序读8次，问命中率是多少？

（3）若Cache的速度是主存的6倍，试问有Cache和无Cache相比，速度约提高多少倍？

解：（1）根据每字块有8个字，每字32位（4字节），得出主存地址字段中字块内地址为3+2=5位。

根据Cache容量为16KB=214B，字块大小为8\*32/8=32=25B，得Cache地址共14位，Cache共有214-5=29块。

根据四路组相联映射，Cache共分为29/22=27组。

根据主存容量为4MB=222B，得主存地址共22位，主存字块标记为22-7-5=10位，故主存地址格式为：

|  |  |  |
| --- | --- | --- |
| 主存字块标记（10位） | 组地址（7位） | 字块内地址（5位） |

（2）由于每个字块中有8个字，而且初态为空，因此CPU读第0号单元时，未命中，必须访问主存，同时将该字所在的主存块调入Cache第0组中的任一块内，接着CPU读第1~7号单元时均命中。同理，CPU读第8，16，…，88号时均未命中。可见，CPU在连续读90个字中共有12次未命中，而后8次循环读90个字全部命中，命中率为：



（3）设Cache的周期为t，则主存周期为6t，没有Cache的访问时间为6t\*90\*8，有Cache的访问时间为t（90\*8-12）+6t\*12，则有Cache和无Cache相比，速度提高的倍数为：



39. 某磁盘存储器转速为3000转/分，共有4个记录盘面，每毫米5道，每道记录信息12 288字节，最小磁道直径为230mm，共有275道，求：

（1）磁盘存储器的存储容量。

（2）最高位密度（最小磁道的位密度）和最低位密度。

（3）磁盘数据传输率。

（4）平均等待时间。

解：（1）存储容量 = 275道×12 288B/道×4面 = 13 516 800B

（2）最高位密度 = 12 288B/（π×230）= 17B/mm = 136位/mm（向下取整）

   最大磁道直径=230mm+2×275道/(5道/mm) = 230mm + 110mm = 340mm

  最低位密度 = 12 288B /(π×340)= 11B/mm = 92位 / mm （向下取整）

（3）磁盘数据传输率= 12 288B × 3000转/分=12 288B × 50转/秒=614 400B/s

（4）平均等待时间 = 1s/50 / 2 = 10ms

42.

M(x)=1001 000

模2除G(x)=1011

|  |  |
| --- | --- |
|  | 1 010 |
| 1011 | 1001 000 |
|  | 1011 |
|  | 0010 |
|  | 0100 |
|  | 1000 |
|  | 1011 |
|  | 0110 |
|  | 110 |

所以R(x)=110

CRC(x)=1001 110

## Chapter 5

2. 简要说明CPU与I/O之间传递信息可采用哪几种联络方式？它们分别用于什么场合？

答： CPU与I/O之间传递信息常采用三种联络方式：直接控制（立即响应）、 同步、异步。 适用场合分别为：

    直接控制适用于结构极简单、速度极慢的I/O设备，CPU直接控制外设处于某种状态而无须联络信号。

    同步方式采用统一的时标进行联络，适用于CPU与I/O速度差不大，近距离传送的场合。

    异步方式采用应答机制进行联络，适用于CPU与I/O速度差较大、远距离传送的场合。

3.（1）程序查询方式。其特点是主机与I/O串行工作。CPU启动I/O后，时刻查询I/O是否准备好，若设备准备就绪，CPU便转入处理I/O与主机间传送信息的程序。若设备没有准备就绪，则CPU反复查询，“踏步”等待直到I/O准备就绪为止。可见这种方式CPU的效率很低。

(2)程序中断方式。其特点是主机与I/O并行工作。CPU启动I/O后，不必时刻查询I/O是否准备好，而是继续执行程序。当I/O准备就绪时，I/O外设向CPU发终端请求信号，CPU在适当的时候响应I/O的中断请求，暂停现行程序的I/O服务。这种方式消除了“踏步”现象，提高了CPU效率。

(3)DMA方式。其特点是主机与I/O并行工作，主存与I/O之间有一条直接的数据通路。CPU启动I/O后，不必查询I/O是否准备好，当I/O准备就绪后，发出DMA请求，此时CPU不直接参与I/O与主存之间的信息交换，只是把系统总线的使用权交给DMA，仍然可以完成自身内部的操作（如加法、移位等），故不必中断现行程序，只需暂停一个存取周期访存（即周期挪用），CPU的效率更高。

(4)通道方式。通道是一个具有特殊功能的处理器，CPU把部分权力下放给通道，由它实现对外围设备的统一管理和外围设备与主存之间的数据交换，大大提高CPU的效率，但是它以花费更多的硬件为代价。

(5)I/O处理机方式。它是通道方式的进一步发展，CPU将I/O操作及外围设备的管理权全部交给I/O处理机，其实质是多级系统，因而效率有更大的提高。

8.某计算机的I/O设备采用异步串行传送方式传送字符信息。字符信息的格式为1位起始位、7位数据位、1位校验位和1位停止位。若要求每秒钟传送480个字符，那么该设备的数据传送速率为多少？  
解：480×10=4800位/秒=4800波特

波特——是数据传送速率波特率的单位。

14.在什么条件下，I/O设备可以向CPU提出中断请求？

I/O设备向CPU提出中断请求的条件是：I/O接口中的设备工作完成状态为1（D=1），中断屏蔽码为0 （MASK=0），且CPU查询中断时，中断请求触发器状态为1（INTR=1）。

5.18 键盘是一种常见的输入设备，它需要识别。见图5.15

5.29DMA接口电路如图5.47所示。DMA完整的工作过程包括预处理、数据传送和后处理3各阶段，这里结合接口电路图，以输入为例，说明传送过程如下：

(1)当设备准备好一个字时，发出选通信号，将该字读写到DMA的数据缓冲寄存器（BR）中，表示数据缓冲寄存器为满。

(2)与此同时设备向DMA接口发请求（DREQ）。

（3）DMA接口向CPU申请总线控制权（HRQ）

(4)CPU发出HLDA信号，表示允许将总线控制权交给DMA接口。

(5)将DMA主存地址寄存器中的主存地址送地址总线，并命令存储器写。

(6)通知设备已被授予一个DMA周期（DACK），并为交换下一个字做准备。

(7)将DMA数据缓冲寄存器的内容送数据总线。

(8)主存将数据总线上的信息写至地址总线指定的存储单元中。

(9)修改主存地址和字计数值。

（10）判断数据块是否传送结束，若未接收，则继续传送，若已结束，则向CPU发送中断请求，标志数据块传送结束。

32. 设磁盘存储器转速为3000转/分，分8个扇区，每扇区存储1K字节，主存与磁盘存储器数据传送的宽度为16位（即每次传送16位）。假设一条指令最长执行时间是25μs，是否可采用一条指令执行结束时响应DMA请求的方案，为什么？若不行，应采取什么方案？

解：先算出磁盘传送速度，然后和指令执行速度进行比较得出结论。

道容量= 1K ×8 ×8 位= 8KB = 4K字

数传率=4K字×3000转/分=4K字×50转/秒 =200K字/秒

一个字的传送时间=1/200K秒≈ 5μs （注：在此1K=1024，来自数据块单位缩写。）

因为5 μs<<25μs，所以不能采用一条指令执行结束响应DMA请求的方案，应采取每个CPU机器周期末查询及响应DMA请求的方案（通常安排CPU机器周期=MM存取周期）。

33. 试从下面七个方面比较程序查询、程序中断和DMA三种方式的综合性能。

（1）数据传送依赖软件还是硬件。

（2）传送数据的基本单位。

（3）并行性。

（4）主动性。

（5）传输速度。

（6）经济性。

（7）应用对象。

解：比较如下：

（1）程序查询、程序中断方式的数据传送主要依赖软件，DMA主要依赖硬件。 （注意：这里指主要的趋势）

（2）程序查询、程序中断传送数据的基本单位为字或字节，DMA为数据块。

（3）程序查询方式传送时，CPU与I/O设备串行工作；程序中断方式时，CPU与I/O设备并行工作，现行程序与I/O传送串行进行；DMA方式时，CPU与I/O设备并行工作，现行程序与I/O传送并行进行。

（4）程序查询方式时，CPU主动查询I/O设备状态；程序中断及DMA方式时，CPU被动接受I/O中断请求或DMA请求。

（5）程序中断方式由于软件额外开销时间比较大，因此传输速度最慢；程序查询方式软件额外开销时间基本没有，因此传输速度比中断快；DMA方式基本由硬件实现传送，因此速度最快；

注意：程序中断方式虽然CPU运行效率比程序查询高，但传输速度却比程序查询慢。

（6）程序查询接口硬件结构最简单，因此最经济；程序中断接口硬件结构稍微复杂一些，因此较经济；DMA控制器硬件结构最复杂，因此成本最高；

（7）程序中断方式适用于中、低速设备的I/O交换；程序查询方式适用于中、低速实时处理过程；DMA方式适用于高速设备的I/O交换；

讨论：

问题1：这里的传送速度指I/O设备与主存间，还是I/O与CPU之间？

答：视具体传送方式而定，程序查询、程序中断为I/O与CPU之间交换，DMA为I/O与主存间交换。

问题2：主动性应以CPU的操作方式看，而不是以I/O的操作方式看。

## Chapter 6

6.12作业

(1)均为原码

51/128=110011/128=0.0110011

浮点数：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶码 | 尾符 | 尾数 |
| 1 | 0001 | 0 | 1100110000 |

-27/1024=-11011/1024=-0.0000011011

浮点数：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶码 | 尾符 | 尾数 |
| 1 | 0101 | 1 | 1101100000 |

7.375=111.011

浮点数：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶码 | 尾符 | 尾数 |
| 0 | 0011 | 0 | 1110110000 |

-86.5=-1010110.1

浮点数：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶码 | 尾符 | 尾数 |
| 0 | 0111 | 1 | 1010110100 |

（2）均为补码

51/128=110011/128=0.0110011

浮点数：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶码 | 尾符 | 尾数 |
| 1 | 1111 | 0 | 1100110000 |

-27/1024=-11011/1024=-0.0000011011

浮点数：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶码 | 尾符 | 尾数 |
| 1 | 1011 | 1 | 0010100000 |

7.375=111.011

浮点数：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶码 | 尾符 | 尾数 |
| 0 | 0011 | 0 | 1110110000 |

-86.5=-1010110.1

浮点数：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶码 | 尾符 | 尾数 |
| 0 | 0111 | 1 | 0101001100 |

（3）阶码移码，尾数补码

51/128=110011/128=0.0110011

浮点数：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶码 | 尾符 | 尾数 |
| 0 | 1111 | 0 | 1100110000 |

-27/1024=-11011/1024=-0.0000011011

浮点数：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶码 | 尾符 | 尾数 |
| 0 | 1011 | 1 | 0010100000 |

7.375=111.011

浮点数：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶码 | 尾符 | 尾数 |
| 1 | 0011 | 0 | 1110110000 |

-86.5=-1010110.1

浮点数：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶码 | 尾符 | 尾数 |
| 1 | 0111 | 1 | 0101001100 |

6.16

（1）0-65535

（2）-（1-2-15）~ +（1-2-15）

（3）-1~ +（1-2-15）

（4）-32768~+32767

（5）-32767~+32767

（6）原码

最大正数，（1-2-10）\*2^（25-1）

最小正数，2-10\*2^（-（25-1））

最大负数，-2-10\*2^（-（25-1））

最小负数，-（1-2-10）\*2^（25-1）

（7）补码

最大正数，（1-2-10）\*2^（25-1）

最小正数，2-1\*2^（-（25））

最大负数，-2-1\*2^（-（25））

最小负数，-（1）\*2^（25-1）

6.20

（1）x=0.110111，y=-0.101110

原码一位乘，符号位单独运算，1⊕0=1

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0.000000 | 101110 |  |
|  | 0.000000 | **0**10111 | 末位0，加0，然后右移 |
| **+** | 0.110111 |  | 末位1，加x |
|  | 0.110111  0.011011 | **10**1011 | 联合右移 |
| **+** | 0.110111 |  | 末位1，加x |
|  | 1.010010  0.101001 | **010**101 | 联合右移 |
| **+** | 0.110111 |  | 末位1，加x |
|  | 1.100000  0.110000 | **0010**10 | 联合右移 |
|  | 0.011000 | **00010**1 | 末位0，加0然后联合右移 |
| **+** | 0.110111 |  | 末位1，加x |
|  | 1.001111  **0.100111** | **100010** | 联合右移 |

最终结果0.101111 100010

原码两位乘

X\*=00.110111 y\*=00.101110, [2X\*]补=01.101110,[ -X\*]补=11.001001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00.000000 | 00 101110 | C=0 |  |
| **+** | 01.101110 |  |  | 末两位10，加[2X\*]补,欠位位置0 |
|  | 01.101110  00.011011 | **10** 001011 | 0 | 联合右移2位 |
| **+** | 11.001001 |  |  | 末两位11，加[ -X\*]补，欠位位置1 |
|  | 11.100100  11.111001 | **0010** 0010 | 1 | 联合右移2位 |
| **+** | 11.001001 |  |  | 末两位10，C=1，加[ -X\*]补，欠位位置1 |
|  | 11.000010  11.110000 | **100010** 00 | 1 | 联合右移2位 |
| **+** | 00.110111 |  |  | 最后一次末位00，C=1，加x\*，欠位位置0 |
|  | **00.100111** | **100010** 00 |  | 不再移位 |

补码一位乘（BOOTH）

[x]补=0.110111,[y]补=1.010010，[-x]补=11.001001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00.000000 | 1.010010 | Yn+1=0 |  |
|  | 00.000000 | **0**1．01001 | 0 | Yn+1-yn=0，右移1位 |
| **+** | 11.001001 |  |  | Yn+1-yn=-1，加[-x]补 |
|  | 11.001001  11.100100 | **10**1．0100 | 1 | 联合右移1位 |
| **+** | 00.110111 |  |  | Yn+1-yn=1，加[x]补 |
|  | 00.011011  00.001101 | **110**1.010 | 0 | 联合右移1位 |
|  | 00.000110 | **1110**1.01 | 0 | Yn+1-yn=0，右移1位 |
| **+** | 11.001001 |  |  | Yn+1-yn=-1，加[-x]补 |
|  | 11.001111  11.100111 | **11110**1.0 | 1 | 联合右移1位 |
| **+** | 00.110111 |  |  | Yn+1-yn=1，加[x]补 |
|  | 00.011110  00.001111 | **0111101.** | 0 | 联合右移1位 |
| + | 11.001001 |  |  | Yn+1-yn=-1，加[-x]补 |
|  | **11.011000** | **011110** |  | 最后一次不移位 |

最终的结果11.011000 011110，注意这是补码

补码两位乘，高位部分积三符号位补码，低位部分积双符号位

[x]补=0.110111,[y]补=1.010010，[-x]补=11.001001 [-2x]补=10.010010

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 000.000000 | 11.010010 | Yn+1=0 |  |
| **+** | 10.010010 |  |  | Yn+1+yn-2\*yn-1=-2，加[-2x]补 |
|  | 10.010010  11.100100 | **10** 11.0100 | 1 | 联合右移2位 |
| **+** | 00.110111 |  |  | Yn+1+yn-2\*yn-1=1，加[x]补 |
|  | 00.011011  00.000110 | **1110**11.01 | 0 | 联合右移2位 |
| **+** | 00.110111 |  |  | Yn+1+yn-2\*yn-1=1，加[x]补 |
|  | 00.111101  00.001111 | **011110**.11 | 0 | 联合右移2位 |
|  | 00.000110 | **1110**1.01 | 0 | Yn+1+yn-2\*yn-1=-1，加[-x]补 |
| **+** | 11.001001 |  |  | 不再移位 |
|  | 11.001111  11.100111 | **11110**1.0 | 1 | 联合右移1位 |
| **+** | 00.110111 |  |  | Yn+1-yn=1，加[x]补 |
|  | 00.011110  00.001111 | **0111101.** | 0 | 联合右移1位 |
| + | 11.001001 |  |  | Yn+1-yn=-1，加[-x]补 |
|  | **11.011000** | **011110** |  | 最后一次不移位 |

（2）x=-0.010111,y=-0.010101

原码一位乘，[x\*y]S=xS ⊕yS = 1⊕1=0

X\*=0.010111, y\*=0.010101

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0.000000 | 010101 |  |
| **+** | 0. 010111 |  | 末位1，加x |
|  | 0.010111  0.001011 | **1 01010** | 联合右移1位 |
|  | 0.000101 | **11** 0101 | 末位0，加0，然后联合右移 |
| **+** | 0. 010111 |  | 末位1，加x |
|  | 0.011100  0.001110 | **011** 010 | 联合右移1位 |
|  | 0.000111 | **0011** 01 | 末位0，加0，然后联合右移 |
| **+** | 0.010111 |  | 末位1，加x |
|  | 0.011110  0.001111 | **00011** 0 | 联合右移1位 |
| **+** | 0.000111 | **100011** | 末位0，加0，然后联合右移 |

最终结果，+0.000111 10011

原码两位乘

X\*=00.010111 y\*=00.010101, [2X\*]补=00.101110,[ -X\*]补=11.101001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00.000000 | 00.010101 | C=0 |  |
| **+** | 00.010111 |  |  | 末两位01，加[X\*]补,欠位位置0 |
|  | 00.010111  00.000101 | **11** 00.0101 | 0 | 联合右移2位 |
| **+** | 00.010111 |  |  | 末两位01，C=0，加[ X\*]补，欠位位置0 |
|  | 00.011100  00.000111 | **0011** 00.01 | 0 | 联合右移2位 |
| **+** | 00.010111 |  |  | 末两位01，C=0，加[ X\*]补，欠位位置0 |
|  | 00.011110  00.000111 | **100011** 00. | 0 | 联合右移2位 |
| **+** | **00. 000111** | **100011** |  | 最后一次末位00，C=0，加0，不再移位 |

最终结果00. 000111 100011

补码一位乘（BOOTH）

[x]补=1.101001,[y]补=1.101011，[-x]补=00.010111

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00.000000 | 1.101011 | Yn+1=0 |  |
| **+** | 00.010111 |  |  | Yn+1-yn=-1，加[-x]补 |
|  | 00.010111  00.001011 | **1**1．10101 | 1 | 联合右移1位 |
| **+** | 00.000101 | **11**1.1010 | 1 | Yn+1-yn=0，直接右移 |
| **+** | 11.101001 |  |  | Yn+1-yn=1，加[x]补 |
|  | 11.101110  11.110111 | **011**1.101 | 0 | 联合右移1位 |
| **+** | 00.010111 |  |  | Yn+1-yn=-1，加[-x]补 |
|  | 00.001110  00.000111 | **0011**1.10 | 1 | 联合右移1位 |
| **+** | 11.101001 |  |  | Yn+1-yn=1，加[x]补 |
|  | 11.110000  11.111000 | **00011**1.1 | 0 | 联合右移1位 |
| + | 00.010111 |  |  | Yn+1-yn=-1，加[-x]补 |
|  | 00.001111  00.000111 | **100011**1. | 1 | 联合右移1位 |
|  | **00.000111** | **100011** |  | Yn+1-yn=0，加0，最后一次不移位 |

最终的结果00.000111 100011，

补码两位乘，高位部分积三符号位补码，低位部分积双符号位

[x]补=1.101001,[y]补=11.101011，[-x]补=00.010111 [-2x]补=00.101110

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 000.000000 | 11.101011 | Yn+1=0 |  |
| **+** | 000.010111 |  |  | Yn+1+yn-2\*yn-1=-1，加[-x]补 |
|  | 000.010111  000.000101 | **11** 11.1010 | 1 | 联合右移2位 |
| **+** | 000.010111 |  |  | Yn+1+yn-2\*yn-1=-1，加[-x]补 |
|  | 000.011100  000.000111 | **0011** 11.10 | 1 | 联合右移2位 |
| **+** | 000.010111 |  |  | Yn+1+yn-2\*yn-1=-1，加[-x]补 |
|  | 000.011110  000.000111 | **100011**.11 | 1 | 联合右移2位 |
|  | 000.000111 | **100011** | 0 | Yn+1+yn-2\*yn-1=0，加0，不再移位 |

最终结果000.000111 100011

6.21

（1）x=0.100111，y=0.101011

原码加减交替，符号位单独处理，0⊕0=0

X\*=x=00.100111,y\*=y=00.101011，[-y\*]补=11.010101

|  |  |  |  |
| --- | --- | --- | --- |
|  | 00.100111 | 0000000 |  |
| **+** | 11.010101 |  | 上来就减y，等于加[-y\*]补 |
|  | 11.111100 | **0000000** | 结果为负，上商0 |
|  | 11.111000 | **00000 0.** | 联合左移1位 |
| **+** | 00.101011 |  | 这一次加上y\* |
| **+** | 00.100011 | **00000 0.1** | 结果为正，上商1 |
|  | 01.000110 | **0000 0.1** | 左移一位 |
| **+** | 11.010101 |  | 这一次加[-y\*]补 |
|  | 00.011011  00.110110 | **0000 0.11**  **000 0.11** | 结果为正，上商1  联合左移1位 |
| **+** | 11.010101 |  | 这一次加[-y\*]补 |
|  | 00.001011  00.010110 | **000 0.111**  **00 0.111** | 结果为正，上商1  联合左移1位 |
| **+** | 11.010101 |  | 这一次加[-y\*]补 |
|  | 11.101011  11.010110 | 00 **0.1110**  0 0.1110 | 结果为负上商0  联合左移1位 |
| **+** | 00.101011 |  | 这一次加上y\* |
|  | 00.000001  00.000010 | 0 0.11101  0.11101 | 结果为正，上商1  联合左移1位 |
| **+** | 11.010101 |  | 这一次加[-y\*]补 |
|  | 11.010111 | 0.111010 | 结果为负上商0 |
| **+** | 00.101011 |  | 最后一次，加回来 |
|  | 00.000010 | 0.111010 | 最终结果 |

补码加减交替法

[x]补=00.100111,[y]补=00.101011，[-y\*]补=11.010101

|  |  |  |  |
| --- | --- | --- | --- |
|  | 00.100111 | 0000000 |  |
| **+** | 11.010101 |  | 余数与除数异号，上来就减y，等于加[-y\*]补 |
|  | 11.111100 | **0000000** | 除余异号，上商0 |
|  | 11.111000 | **00000 0.** | 联合左移1位 |
| **+** | 00.101011 |  | 这一次加上y\* |
| **+** | 00.100011 | **00000 0.1** | 除余同号，上商1 |
|  | 01.000110 | **0000 0.1** | 左移一位 |
| **+** | 11.010101 |  | 这一次加[-y\*]补 |
|  | 00.011011  00.110110 | **0000 0.11**  **000 0.11** | 除余同号，上商1  联合左移1位 |
| **+** | 11.010101 |  | 这一次加[-y\*]补 |
|  | 00.001011  00.010110 | **000 0.111**  **00 0.111** | 除余同号，上商1  联合左移1位 |
| **+** | 11.010101 |  | 这一次加[-y\*]补 |
|  | 11.101011  11.010110 | 00 **0.1110**  0 0.1110 | 除余异号，上商0  联合左移1位 |
| **+** | 00.101011 |  | 这一次加上y\* |
|  | 00.000001  00.000010 | 0 0.11101  0.11101 | 除余同号，上商1  联合左移1位 |
| **+** | 11.010101 |  | 这一次加[-y\*]补 |
|  | 11.010111 | 0.111010 | 除余异号，上商0 |
| **+** | 00.101011 |  | 最后一次，加回来 |
|  | 00.000010 | 0.111010 | 最终结果 |

（2）x=-0.10101，y=0.11011

原码加减交替，符号位单独处理，1⊕0=0

X\*=x=00.10101,y\*=y=00.11011，[-y\*]补=11.00101

|  |  |  |  |
| --- | --- | --- | --- |
|  | 00.10101 | 000000 |  |
| **+** | 11.00101 |  | 上来就减y，等于加[-y\*]补 |
|  | 11.11010 | **000000** | 结果为负，上商0 |
|  | 11.10100 | **0000 0.** | 联合左移1位 |
| **+** | 00.11011 |  | 这一次加上y\* |
|  | 00.01111 | **0000 0.1** | 结果为正，上商1 |
|  | 00.11110 | **0000 0.1** | 左移一位 |
| **+** | 11.00101 |  | 这一次加[-y\*]补 |
|  | 00.00011  00.00110 | **0000 0.11**  **000 0.11** | 结果为正，上商1  联合左移1位 |
| **+** | 11.00101 |  | 这一次加[-y\*]补 |
|  | 11.01011  11. 10110 | **000 0.110**  **00 0.110** | 结果为正，上商0  联合左移1位 |
| **+** | 00.11011 |  | 这一次加[y\*]补 |
|  | 00.10001  01.00010 | 00 **0.1101**  0 0.1101 | 结果为负上商1  联合左移1位 |
| **+** | 11.00101 |  | 这一次加[-y\*]补 |
|  | 00.00111  00.01110 | 0 0.11011  0.11011 | 结果为正，上商1  联合左移1位 |

26. 先将x、y转换成机器数形式：

**（1）**x=2-011× 0.101 100，y=2-010×（-0.011 100）

[x]补=1，101；0.101 100, [y]补=1，110；1.100 100

[Ex]补=1,101, [y]补=1,110, [Mx]补=0.101 100, [My]补=1.100 100

    1）对阶：

[ΔE]补=[Ex]补+[-Ey]补 = 11,101+ 00,010=11,111 < 0，

应Ex向Ey对齐，则：[Ex]补+1=11，101+00，001=11，110 = [Ey]补

[x]补=1，110；0.010 110

2）尾数运算：

  [Mx]补+[My]补= 0.010 110 + 11.100 100=11.111010

[Mx]补+[-My]补=0.010 110 + 00.011100= 00.110 010

3）结果规格化：

  [x+y]补=11，110；11.111 010 = 11，011；11.010 000 （尾数左规3次，阶码减3）

  [x-y]补=11，110；00.110 010, 已是规格化数。

4）舍入：无

5）溢出：无

则：x+y=2-101×（-0.110 000）

    x-y =2-010×0.110 010

**（2）**x=2-011×（-0.100010）,y=2-010×（-0.011111）

    [x]补=1，101；1.011 110, [y]补=1，110；1.100 001

1. 对阶：过程同(1)的1），则

[x]补=1，110；1.101 111

  2）尾数运算：

    [Mx]补+[My]补= 11.101111 + 11. 100001 = 11.010000

    [Mx]补+[-My]补= 11.101111 + 00.011111 = 00.001110

3）结果规格化：

[x+y]补=11，110；11.010 000,已是规格化数

[x-y]补=11，110；00.001 110 =11，100；00.111000 （尾数左规2次，阶码减2）

4）舍入：无

5）溢出：无

则：x+y=2-010×（-0.110 000）

    x-y =2-100×0.111 000

**（3）**x=2101×（-0.100 101）,y=2100×（-0.001 111）

    [x]补=0，101；1.011 011, [y]补=0，100；1.110 001

1）对阶：

[ΔE]补=00，101+11，100=00，001 >0，应Ey向Ex对齐，则：

[Ey]补+1=00，100+00，001=00，101=[Ex]补

[y]补=0，101；1.111 000（1）

2）尾数运算：

  [Mx]补+[My]补= 11.011011+ 11.111000（1）= 11.010011（1）

  [Mx]补+[-My]补= 11.011011+ 00.000111（1）= 11.100010（1）

3）结果规格化：

  [x+y]补=00，101；11.010 011（1），已是规格化数

  [x-y]补=00，101；11.100 010（1）=00，100；11.000 101 （尾数左规1次，阶码减1）

4）舍入：

[x+y]补=00，101；11.010 011（舍）

[x-y]补 不变

5）溢出：无

则：x+y=2101×（-0.101 101）

x-y =2100×（-0.111 011）

32. 设机器字长为16位，分别按4、4、4、4和5、5、3、3分组后，

  （1）画出按两种分组方案的单重分组并行进位链框图，并比较哪种方案运算速度快。

  （2）画出按两种分组方案的双重分组并行进位链框图，并对这两种方案进行比较。

  （3）用74181和74182画出单重和双重分组的并行进位链框图。

解：（1）4—4—4—4分组的16位单重分组并行进位链框图见教材286页图6.22。

    5—5—3—3分组的16位单重分组并行进位链框图如下：

（2）4—4—4—4分组的16位双重分组并行进位链框图见教材289页图6.26。

    5—5—3—3分组的16位双重分组并行进位链框图如下：

   5—5—3—3分组的进位时间=2.5ty×3=7.5ty；

  4—4—4—4分组的进位时间=2.5ty×3=7.5ty；

可见，两种分组方案最长加法时间相同。

结论：双重分组并行进位的最长进位时间只与组数和级数有关，与组内位数无关。

（3）单重分组16位并行加法器逻辑图如下（正逻辑）：

**注意：** 1）74181芯片正、负逻辑的引脚表示方法；

  2）为强调可比性，5-5-3-3分组时不考虑扇入影响；

  3）181芯片只有最高、最低两个进位输入/输出端，组内进位无引脚；

  4）181为4位片，无法5-5-3-3分组，只能4-4-4-4分组；

  5）单重分组跳跃进位只用到181，使用182的一定是双重以上分组跳跃进位；

  6）单重分组跳跃进位是并行进位和串行进位技术的结合；双重分组跳跃进位是二级并行进位技术；特别注意在位数较少时，双重分组跳跃进位可以采用全先行进位技术实现；位数较多时，可采用双重分组跳跃进位和串行进位技术结合实现。