**2025** **计算机系统基础**

目录

[第一章 2](#bookmark1)

[一、简答 2](#bookmark2)

[二、综合题 2](#bookmark3)

[第二章 7](#bookmark4)

[一、简答 7](#bookmark5)

[二、综合题 8](#bookmark6)

[第三章 18](#bookmark7)

[一、简答 18](#bookmark8)

[二、综合题 18](#bookmark9)

[第四章 28](#bookmark10)

[一、简答 28](#bookmark11)

[二、综合题 28](#bookmark12)

[第五章 31](#bookmark13)

[一、简答 31](#bookmark14)

[二、综合题 33](#bookmark15)

[第六章 40](#bookmark16)

[一、简答 40](#bookmark17)

[二、综合题 41](#bookmark18)

[第七、八章 47](#bookmark19)

[一、简答 47](#bookmark20)

[二、综合题 51](#bookmark21)

**第一章**

**一、简答**

1. 什么是“存储程序”工作方式？

答：“存储程序”方式的基本思想是:必须将事先编好的程序和原始数据送入主存 后才能执行程序，一旦程序被启动执行，计算机能在不需操作人员干预下自动完 成逐条指令取出和执行的任务。

2. 一条指令的执行过程包含哪几个阶段？

3. 计算机系统的层次结构如何划分？计算机系统的用户可分哪几类？每类用 户工作在哪个层次？

4. 请说明以下措施对缩短程序的响应时间和提高系统的吞吐率各有什么影响？

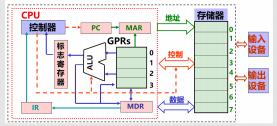
5. 程序的 CPI 与哪些因素有关？

6. 为什么说性能指标 MIPS 不能很好地反映计算机的性能？

**二、综合题**

1. 请画图说明现代计算机的基本结构，并描述各部分的功能是什么？以你所画 的系统为例，说明程序执行的基本过程。

答：



描述略

2. 请说明以下措施对缩短程序的响应时间和提高系统的吞吐率各有什么影响？ ( 1 ）使用更快的处理器。

( 2 ) 增加处理器个数， 使得不同的处理器同时执行不同的作业。 ( 3 ）优化编译生成的代码， 使得程序执行的总时钟周期数减少。

（4 ）在 CPU 和主存之间增加 cache， 减少访问指令和数据的时间。

答案：措施（ 1 ）： 因为总执行时间＝总时钟周期数 x 时钟周期。 更快的处理 器意味着时钟频率加快了 即每个时钟周期更短了 故总执行时间变短了。 因此 采用更快的处理器可缩短程序响应时间，从而使单位时间内完成的作业数增加， 系统的吞吐率也提高。

措施（ 2 ）： 处理器个数的增加，使得同时可以有多个作业在不同的处理器上 进行处理，因而系统在单位时间内可以完成更多的作业， 吞吐率会明显提高。 但每个作业的处理过程．还是一样的，所以程序的执行时间并不会缩短。但是， 因为吞吐率提高了， 所以作业在等待队列中的排队时间减少了， 因而响应时间 会在一定程度上有相应的改善。

措施（3 ）：优化编译生成的代码使总的执行时钟周期数减少， 也就缩短了程序 的时间． 因而使得单位时间内完成的作业数增加提高了系统的吞吐率。

措施（4 ）：在 CPU 和主存之间增加cache 使得程序访问存储器的时间大大缩短， 因而可以缩短程序的响应时间， 从而也就使得单位时间内完成的作业数增加， 提高了系统的吞率。

综上所述， 措施（ I ）、（3 ）和（4）是因为首先缩短了程序的执行时间而使

得系统的吞 ut 率提高；而措施（ 2 ）是因为提高了系统的吞吐率， 使得程序 等待时间缩短而缩短了程序 响应时间。程序响应时间和系统的吞吐率之间通常 是相辅相成的关系， 但在某些特定情况下， 它们有可能是对立关系。

3. 假定某程序 P 由一个 100 条指令构成的循环组成， 该循环共执行 50 次， 在 某系统 S 中执行程序 P 花了 20 000 个时钟周期， 则系统 S 在执行程序 P 时 的 CPI 是多少？

答案：

程序 P 中 100 条指令的循环被执行了 50 次， 所以在 20 000 个时钟周期中 共执行了 5000 条指令，所以，系统 S 在执行程序 P 时的 CPI=20 000/5000=4。

4. 若机器 Ml 和 M2 具有相同的指令集，其时钟频率分别为 0.8GHz 和 1.6GHz。 在指令集中有五种不同类型的指令 A-E。下表给出了在 Ml 和 M2 上每类指令 的平均时钟周期数 CPI。



请回答下列问题：

（1） Ml 和 M2 的峰值 MIPS 各是多少？

（2） 假定某程序 P 的指令序列中， 五类指令具有完全相同的指令条数， 则程序 P 在 Ml 和 M2 上运行时，哪台机器更快？快多少？在 Ml 和 M2 上执行程序 P 时的平均时钟周期 数 CPI 各是多少？

答案：

（1）计算峰值 MIPS 时应该选择 CPI 最少的指令，故在 Ml 上可以选择一段全部 由 A 类指令组成的程序，其峰值 MIPS 为 0.8GHz/l=800MIPS；在 M2 上可以选择 一段全部由 A 类和 B 类指令组成的程序，其峰值 MIPS 为 l.6GHz/2=800MIPS。

（ 2 ） 对 于 程 序 P ， 每 类 指 令 的 条 数 均 占 1/5 ， 故 Ml 的 CPI 为 CPI1=(1+2+2+3+4)/5=2.4, M2 的 CPI2 为 CPI=(2+2+4＋+5+6）／5=3.8。 当然， 不能根据以上结果 说明程序 P 在 Ml 上运行更快， 因为 Ml 和 M2 的时钟频率不 同。

假设程序 P 的指令条数为 N，则 P 在 Ml 上的执行时间为 2.4 \* N

\*1/0.8GHz=3.0N (ns); 在 M2 上的执行时间为 3.8\*N\*l/1.6GHz=2.375N ( ns）， 所以， M2 执行 P 的速度更快，每条指令平均快 0.625ns 。

从该题可以看出，虽然程序P在Ml中每条指令执行所花的时钟周期数少 ， 但是因为 M2 的时钟频率更快 ，所以时钟周期更短，使得每条指令的平均执行 时间更短。

5. 假设同一套指令集用不同的方法设计了两种机器 Ml 和 M2。机器 Ml 的时 钟周期为 0.8ns，机器 M2 的时钟周期为 l.2ns。 某个程序 P 在机器 Ml 上运行时的 CPI 为 4，在 M2 上的 CPI 为 2。对于程序 P 来说， 哪台机器 的执行速度更快？快多少？

答案：

因为 Ml 和 M2 实现的是同一套指令集， 所以程序 P 在机器 Ml 和 M2 上的指 令条数相同，假定是 N 条，则 P 在 Ml 上的执行时间为 4\*0.8ns\*N=3.2N (ns); P 在 M2 上的执行时间为 2 \*1.2ns\*N=2.4N ( ns）。 由此可知， 对于程序 P 来说， M2 的执行速度更快，平均每条指令快 0.8ns。

6. 假定某编译器对某段高级语言程序编译生成两种不同的指令序列 Sl 和 S2， 在时钟频率为 500MHz 的机器 M 上运行， 目标指令序列中用到的指令类型 有 A 、 B 、 C 和 D 四类。 四类指令在 M 上的 CPI 和两个指令序列所 用的各类指令条数如下表所示。



请问： Sl 和 S2 各有多少条指令？CPI 各为多少？所含的时钟周期数各为 多少？执行时间各为多少？

答案：

序列 Sl 的指令条数为 5+2+2+1=10, CPI 为（5\*1+2\*2+2\*3+1+4)/10=1.9， 所含的时钟周期数为 l.9\*10=19,故执行时间为 19/(500\*106)s= 38ns

序列 S2 的指令条数为 l+l+1+5=8, CPI 为(1\*l+1\*2+1\*3+5\*4)/8=3.25，

所含的时钟周期数为 3.25\*8= 26,故执行时间为 26/(500\*106)s=52ns

**第二章**

**一、简答**

1. 什么是真值？什么是机器数？二者有什么关系？

2. 为什么计算机内部采用二进制表示信息？既然计算机内部所有信息都用二 进制表示， 为什么还要用到十六进制或八进制数？

3. 常用的定点数编码方式有哪几种？通常它们各自用来表示什么？

4. 在浮点数的基数和总位数一定的情况下，浮点数的表示范围和精度分别由什 么决定？两者如何相互制约？

5. 为什么要对浮点数进行规格化？有哪两种规格化操作？

6. 为什么计算机处理汉字时会涉及不同的编码（如，输入码、内码、字模码）？ 说明这些编码中哪些用二进制编码，哪些不用二进制编码，为什么？

7. 什么是“溢出”？为什么无符号数运算时结果可能会发生 “溢出 ” ？什么 叫无符号数的 “溢出 ” ？

8. 常用的溢出判断方法有哪些？并简单描述。

9. C 语言程序中， 为什么关系表达式 “123456789=(int)(float)123456789”

的结果为 “假 ”，而关系表达式 “ 123456==(int)(float)123456” 和 “ 123456789=(int)( double) 123456789” 的结果都为 “真 ” ？

答：在 C 语言中，float 类型对应 IEEE 754 单精度浮点数格式，也即 float 型 数据的有效位数只有 24 位（相当于有 7 位十进制有效位数）；double 类型对应 IEEE 754 双精度浮点数格式，有效位数有 53 位（相当于有 1 7 位十进制有效位 数）；int 类型为 32 位整数，其有效位数为 31 位（最大数为 2147483647，相 当于 10 位十进制有效位数）。

整数 123456789 的有效位数为 9 位，转换为 float 型数据后肯定发生了有效 位数丢失，再转换为 int 型数据时， 已经不是 123456789 了，所以，关系表达 式 “ l23456789==(int) (float) 123456789 ”的结果为假。

数据改为 123456 后，有效位数 只有 6 位，转换为 float 型数据后有效位数 没有丢失，因而数据没变，再转换为 int 型数据时，还是 123456，所以，关系表

达式“123456==(int)（float) 123456”的结果为真。

整数 123456789 的有效位数为 9 位 ， 转换为 double 型数据后， 不会发 生有效位数丢失，再转换为 int 型数据时，还是 123456789，所以，关系表达式 “l23456789==(int) (double)123456789”的结果为真。

10. 什么是“word（字）”?什么是“机器字长”？ 一个 “字 ” 的宽度和一个 “机器字长 ”相同吗？为什么？

答: “字”作为信息宽度的计量单位，对于某个系列机来说，其字宽总是固定的。 例如，在 80x86 系列中，一个字的宽度为 16 位。（2 分）

“机器字长”是计算机的一个非常重要的指标。通常称 32 位机器或 64 位机器， 就是指机器的字长是 32 位或 64 位。一般情况下，机器字长定义为 CPU 中一次能 够处理的二进制整数的位数，实际上就是 CPU 中整数运算数据通路的位数。（2 分）

不相同。一个“字”的宽度可以不等于机器字长。例如，在 Intel 微处理器中， 从 80386 开始就至少都是 32 位机器了，即机器字长至少为 32 位，但其字的宽度 都定义为 16 位。（2 分，只回答是否相同给 1 分）

**二、综合题**

1. 实现下列各数的转换

(1) (25.8125 )10 = ( ? ) 2 = ( ? ) 8 = ( ? ) 16

(2) (101101.011) 2 = (?) 10 = (?) 8 = (?) 16 = （？）8421

(3) (0101 1001 0110. 0011）8421= ( ? ) 10= ( ? ) 2 = ( ? ) 16

(4) (4E. C) 16 = (?) 10 = (?)2

答案：

(1)(25.8125)10=(1l001. 1101)2=(3 l.64)s=(19 .D)16

(2)(101101.0ll)2=(45.375)10=(55.3)s=(2D.6)16=(01000101.0011Olll0101)8 421

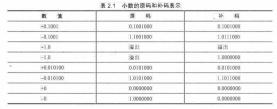
(3)(010110010110.001l)8421=(596.3)10=(1001010100.010011…)2=(254.4…)

16

(4) (4E.C)16=(78.75)10=(1001110.11)2

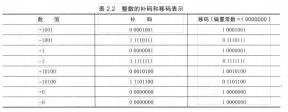
2. 假定机器数为 8 位 (1 位符号， 7 位数值）， 写出下列各二进制数的原码表 示。

+0.1001, -0.1001, +1.0, -1.0, +0.010100, -0.010100, +O, -0



3. 假定机器数为 8 位 (1 位符号， 7 位数值）， 写出下列各二进制数的原码表 示。

+0.1001, -0.1001, +1.0, -1.0, +0.010100, -0.010100, +O, -0



4. 己知［x]补， 求[x]。

(1) [x]补=11100111

(2) [x]补=10000000

(3) [x]补=01010010

(4) [x]补=11010011 答案：

(1) [x]=-00011001=-25

(2) [x]=-10000000=-128

(3) [x]= 01010010=82

(4) [x]=-00101101=-45

5. 在 32 位计算机中运行一个 C 语言程序，在该程序中出现了以下变量的初值， 请写出它们对应的机器数（用十六进制表示）。

(1)int x = - 32768

(2) short y=522

(3) unsigned z=65530

(4) char c＝ ’0 ’

(5) float a= -1.1

(6) double b=l0.5

6. 在 32 位计算机中运行一个 C 语言程序， 在该程序中出现了一些变量， 己 知这些变量在某一时刻的机器数（用十六进制表示）如下， 请写出它们对应 的真值。

(1)int x:FFFF0006H

(2)short y：DFFCH

(3)unsigned z:FFFFFFFAH

(4)char c:2AH

(5)float a:C4480000H

(6)double b:C024800000000000H

7、假定某程序中定义了三个变量 x、y 和 z, 其中 x 和 z 为 int 型，y 为 short 型。当 x=-258, y=-20 时，执行赋值语句 z=x-y 后，存放 z 的寄存器中的内容是 多少？

答案：现代计算机中的带符号整数都是用补码表示的， 因此，本题可以直接计 算 z 的值， 然后将 z 的补码形式求出来，也可以先将 x 和 y 的补码求出， 再通 过补码加法求出 z 的补码表示。显然，前一种思路效率较高。对于前一种思路， 执行赋值语句后，z=-238, 因此，问题就变成了求-238 的补码表示， 其结果为 （-000 0000 0000 0000 0000 0000 0000 1110 111O）补=11ll llll llll llll

1111 1111 0001 OO1O=FFFF FF12H 。

8、某字长为 8 位的计算机中，x 和 y 为无符号整数， 已知 x=68, y=80, x 和 y 分别存放在寄存器 A 和 B 中 。 请回答下列问题（要求最终用十六进制表示二进 制序列）。

(1) 寄存器 A 和 B 中的内容分别是什么？

(2)若 x 和 y 相加后的结果存放在寄存器 C 中， 则寄存器 C 中的内容是什么？ 运算结果是否正确？加法器最高位的进位 Cout 是什么？零标志 ZF 和进位标志 CF 各是什么？

(3)若 x 和 y 相 减后的结果存放在寄存器 D 中，则寄存器 D 中的内容 是什么？ 运算结果是否正确？加法器最高位的进位 Cout 是什么？零标志 ZF 和借位标志 CF 各是什么？

(4) 无符号整数加/减运算时，加法器最高位进位 Cout 的含义是什么？它与进/ 借位标志 CF 的关系是什么？

答案：

(1) x=68=0100 0100 B=44H; y=80=0101 0000 B=50H, 所以，寄存器 A 和 B 中 的内容分别是 44H 和 50H。

(2) x+y=O100 0100+01O1 0000=(0)100l O100=94H, 所以，寄存器 C 中的内容为 94H,对应的真值为 148, 运算结果正确。 加法器最高位的进位 Cout 为 0。 因为 结果不为 0, 所以 ZF=0，进位标志 CF=Cout=O。

(3) x-y=x+[-y]补=0100 O1OO+1Oll 0000 =(0)1111 0100 = F4H, 所以， 寄存 器 D 中的内容为 F4H,对应的真值为 244,运算结果不正确，这是由相减结果为负 数造成的。加法器最高位的进位 Cout 为 0。因为结果不为 0,所以 ZF O; 借位标 志为 CF =Cout ⊕ 1=1。

(4)在加法器中进行无符号整数加法运算时，若加法器最高位进位Cout=l, 则表 示实际结果大于最大可表示数 255, 即溢出，此时 CF =l。因此，对于无符号整 数加运算来说，CF=l 表示溢出；在加法器中进行无符号整数减法运算时，若加法 器最高位进位 Cout=l, 则表示被减数大千减数，反之被减数小于减数。 因此， 在无符号数相加 时，CF 就等于 Cout, 表示进位：在无符号数相减 时， 将最高

--------

进位 Cout 取反来作为借位标志 CF, 即 CF=Cout ⊕ 1=Cout，CF =l 表示有借位。

9、列几种情况所能表示的数的范围是什么？

(1)16 位无符号整数。

(2)16 位原码定点小数。

(3)16 位补码定点整数。

(4)下述格式的浮点数（基数为 2, 移码的偏置常数为 128, 规格化尾数，不考虑 藏位）。



答案：

(1)16 位无符号整数范围为 0—216-1, 即 0—65535。

(2)16 位原码定点小数表示的范围为-(l-2-15)—+(l-2-15)

(3)16 位补码定点整数表示的范围为-215 — +(215-1) ,即-32768—+32767。 (4) 规格化浮点数的表示范围如下。

最大正数： +O.111 1111B\*21111 1111B=+(l-2-7) \*2127

最小正数：+0.100 0000B\*20000 OOOOB=+2-1\*2-128=+2-129。

最大负数：-0.100 0000B\*20000 0000B=-2-1\*2-128=-2-129。

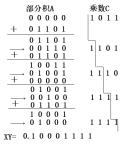
最小负数：-0.111 1111B\*21111 1111B=-(l-2-7)\*2127。

由于原码是关于原点对称的，所以，浮点数的表示范围是关于原点对称的。

对于非规格化浮点数，其最小正数和最大负数的尾数形式为+-0.0000001, 最小 正数和最大负数的值为+-2-7\*2l28=+-2-135。

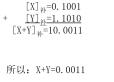
10、计算 X=0.1101 Y=0.1011 的原码一位乘法，要求写出计算过程。

答：



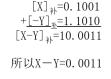
11、已知 X=0.1001,Y=－0.0110，机器字长为 5 位，用补码运算求 X+Y=？、X- Y=？，要求写出计算过程。

答：[X]补=0.1001、 [Y]补=1.1010，



已知 X=0.1001,Y=0.0110，机器字长为 5 位，用补码运算求 X-Y=？，要求写出计 算过程。

答：[X]补=0.1001、[Y]补=0.0110、[-Y]补=1.1010，



12、已知 X=0.1011，Y=-0.1101。用恢复余数法求 X÷Y。

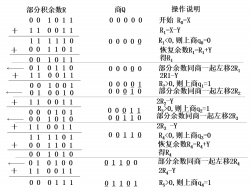
答：

• |X|=0.1011， |Y|=0.1101，

• [|X|]补=00.1011， [|Y|]补=00.1101

• [-|Y|]补=11.0011。

• 采用恢复余数-Y 可以用+[-|Y|]补来实现。采用双符号位（防止左移时部 分余数会改变符号位产生溢出）。在计算机中隐含了小数点，[恢复余数运](file:///I:/2024/2023202402计算机系统基础/期末/恢复余数法.ppt) [算过程如图](file:///I:/2024/2023202402计算机系统基础/期末/恢复余数法.ppt)



• 商的符号=0⊕1=1。所以 X/Y=－0.1101 (商)， 余数 0.0111×2

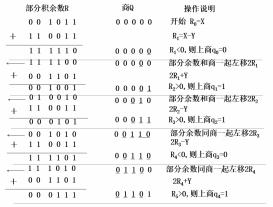
13、已知 X=0.1011，Y=-0.1101。用不恢复余数法求 X÷Y。

答：

|X|=0.1011， |Y|=0.1101， [|X|]补=0.1011， [|Y|]补=00.1101， [-|Y|]补 =11.0011。

采用不恢复余数。-Y 可以用+[-|Y|]补来实现。采用双符号位（防止左移时部分 余数会改变符号位产生溢出）。

不恢复余数运算过程



商的符号=0⊕1=1，所以 X/Y=－0.1101(商)，余数 0.0111×2-4

14、假设某字长为 8 位的计算机中， 带符号整数采用补码表示， x=-68, y=- 80, x 和 y 分别存放在寄存器 A 和 B 中。 请回答下列问题（要求最终用十六 进制表示二进制序列） 。

（1）寄存器 A 和 B 中的内容分别是什么？

（2）若 x 和 y 相加后的结果存放在寄存器 C 中， 则寄存器 C 中的内容是什么？ 运算结果是否正确？加法器最高位的进位 Cout 是什么？溢出标志 OF、符号标 志 SF 和零标志 ZF 各是什么？

（3）若 x 和 y 相减后的结果存放在寄存器 D 中， 则寄存器 D 中的内容是什 么？运算结果是否正确？此时， 加法器最高位的进位 Cout 是什么？溢出标志 OF 、符号标志 SF 和零标志 ZF 各是什么？

（4）对于带符号整数的减法运算， 能否直接根据 CF 的值对两个带符号整数的 大小进行比较？

答：

(1)[68]补 =［－1000100］补 =1011 11OOB=BCH, [-80］补 ＝[－1010000］补 ＝1011 OOOOB= BOH，所以， 寄存器 A 和 B 中的内容分别是 BCH 和 BOH。

(2) [x+y] 补＝［x］补＋［y］补 =1011 1100 B+1O11 0000 B=(1) 0110 1100B=6CH，

最高位前面的一位 l 被丢弃， 因此，寄存器 C 中的内容为 6CH，对应的真值 为＋108，结果不正确。 加法器最高位向前面的进位 Cout 为 l。槛出标志位 OF 可采用以下任意一条规则判断得到。 规则 l：若两个加数的符号位相同， 但 与结果的符号位相异， 则溢出；规则 2：若最高位上的进位和次高位上的进位不 同， 则溢出。 对于本题， 通过这两个规则都判断出结果溢出， 因此溢出标志 OF 为 1，说明寄存器 C 中的内容不是正确的结果。x 坷的正确结果应是－68+(- 80)=-148，而运算的结果为 108，两者不等。 这是因为 x+y 的值（即－148） 小于 8 位补码可表示的最小值 （即－128 ）， 也即结果发生了溢出；结果的第 一位（最高位）0 为符号标志位 SF， 即 SF=O, 表示结果为正数；因为结果不 为 0，所以零标志 ZF=O。

(3) [x-y]补＝［x］补＋［-y］补= 1011 1100 B+O 101 OOOOB=( 1) 0000 11 OOB=OCH， 最高位前 面的一位 1 被丢弃， 因此， 寄存器 D 中的内容为 OCH，对应的真 值为＋12，结果正确。 加法器最高位向前面的进位 Cout 为 l。两个加数的符 号位相异一定不会溢出， 因此溢出标志 OF=O， 说明寄存器 D 中的内容是真正 的结果；结果的第一位（最高位） 0 为符号标志位 SF，即 SF=O，表示结果为正 数；因为结果不为 0， 所以零标志 ZF=O。

( 4）对于带符号整数的 减法运算， 无法直接根据 CF 的值判断两个带符号整 数的大小。例如， 对于 x=-68, y=80, [x-y］补＝［x］补＋［－y］补 ＝1011 1100 B+ 1011 0000B=(1) 0110 1100B 得到的 Cout 为 1，因此，CF=Cout ⊕ 1=0， 表 示没有借位， 推断出被减数应该大于减数，即－68>80，显然这是不 正确的 ， 因此带符号运算中不考虑 CF 标志。

15、假定某计算机存储器按字节编址， CPU 从存储器中读出 一个 4 字节信息 D=3234 3538H,该信息的内存 地址为 0000 FOOCH， 按小端方式存放（LSB 数据 字的最低有效字节存放在小地址单元中），请回答下列问题。

（1）该信息 D 占用了几个内存单元？这几个内存单元的地址及其内容各是什么？

（2）若 D 是一个 32 位无符号数， 则其值是多少？

（3）若 D 是一个 32 位补码表示的带符号整数， 则其值是多少？

（4）若 D 是一个 IEEE 754 单精度浮点数， 则其值是多少？

（5）若 D 是一个用8421 码表示的无符号整数， 则其值是多少？

（6）若 D 是一个字符串， 每个字节的低 7 位表示对应字符的 ASCII 码 ， 则对 应字符串 是 什么？

（7）若 D 是两个汉字的国标码，则这两个汉字在 GB 2312 一 1980 字符集码表 中分别位于哪一行和哪一列？

（8）若 D 中前 3 个字节分别是一个像素的 R、 G、 B 分量的颜色值， 则其值各 是多少？

答：将 3234 3538H 展开为二进制表示为 0011 0010 0011 0100 0011 0101 0011

1OOOB。

（1）因为存储器按字节编址， 所以 4 个字节 占用 4 个内存单元 ， 其地址分 别是 0000 FOO CH 、 0000 FOODH 、 0000 FOOEH 、 0000 FOOFH。 由于采用 小端方式存放． 所以， 最低有效字节 38H 存放在 0000 FOOCH 中， 35H 存放 在 0000 FOODH 中 34H 存放在 0000 FOOEH 中，32H 存放在 0000 F00FH 中。

（2）无符号数。值为 229+228+225+221+220+218+213+212+210+28+25+24+23。

（3）补码整数。符号为 0，表示其为正数，其值与无符号数的值一样。

（4）IEEE 754 单精度浮点数。 根据 IEEE 754 单精度浮点数格式可知， 符号 位 s=O，为正数；阶码 e=O110 O1OOB=1OO，故阶为 100-127=-27 ； 尾数小数 部分 f=0.011 0100 0011 01O1 0011 1000 所以， 其值为 1.011 O100 0011 01

01 0011 lB×2-27。

（5）8421 码整数。 3234 3538H 各位表示对应十进制数 32343538，所以， 其 值为 32343538。

（6）ASCII 码字符串。 各字节的低 7 位分别为 011 0010 、 011 0100 、011

0101、 011 1000， 所以， 对应的字符串为 “2458 。

( 7 ） 汉字。 对国标码每个字节各自减 20H， 得到两个叹字的区位码， 分别 为 1214H 和 1518H， 也即， 第一个汉字在 GB 2312-1980 字符集码表中位 于 第 18 ( 12H ）行、 第 20 ( 14H ）列， 第二个汉字位于第 21 ( 15H ）行、 第 24 ( 18H）列。

( 8 ） 颜色值。 该像素的 R 、 G 、 B 分量的颜色值分别为 0011 0010B=50, 0011 01008=52, 0011 O1O1B=53。

**第三章**

**一、简答**

1. 一台计算机中的所有指令都是一样长吗？

2. 一条机器指令通常由哪些字段组成？各字段的含义分别是什么？

3. 将一个高级语言源程序转换成计算机能直接执行的机器代码通常需要哪几 个步骤？

4. 有哪些常用的数据寻址方式？并通过汇编指令举例说明。

答：数据寻址方式可以归为以下几类。①立即寻址。指令中的立即数字段既可以 作为操作数，也可以作为直接转移地址。取到 ALU 运算前，可能要对其进行扩展。

②直接寻址类。 指令中直接给出操作数所在的寄存器编号 、I/0 端口号或存储 单元地址，如直接寻址方式、 寄存器寻址方式。③间接寻址类。操作数在存储 单元中，而操作数的地址存放在寄存器或另一个存储单元中， 指令中给出操作 数的地址所在的寄存器编号或存储单元地址， 如间接寻址 方式、 寄存器间接 寻址方式。 ④偏移寻址类。 指令通过某种方式给出一个形式地址和｜一个基 地址（在某个寄存器中）， 经过相应的计算（基地址加形式地址）得到操作数所 在的存储单元地址，有变址、相对和基址寻址方式三种。

5. 按值传递参数和按地址传递参数两种方式有哪些不同点？

6. 为什么数据在存储器中最好按地址对齐方式存放？

**二、综合题**

1、分析如下程序，执行之后，AX，BX 中的值是多少？请写出分析过程。（6 分）

|  |  |
| --- | --- |
| MOV AX，0 | ;ax=0 |
| MOV AX，0 | ;ax=0 |
| MOV BX，1 | ;bx=1 |
| MOV CX，100 | ;cx=100 |
| A：ADD AX，BX | ;ax=1 |

INC BX ;bx=2

LOOP A 循环 100 次，跳转到 A

所以，bx 的值循环 100 次后，变成了 101，ax 的值，循环 100 次后，存的是 1+2+3+…+100=5050

执行后（BX）=（ 101 ），（AX）=（5050）

2. 设 (DS)=1000H, (ss)=2000H, (ES)=3000H, (BX)=0100H, (BP)=0120H, (SI)=0200H, (DI)=0220H, 试计算以下指令中存储器操作数的物理地址。

(1) mov al,[bx]

(2) mov cx,[bp]

(3) and ah,es:[si]

(4) push 10[bx][di]

答案：

(1) mov al,[bx] (ds)\*16+(bx)=1000h\*10h+0100h=10100h

(2) mov cx,[bp] (ss)\*16+(bp)=2000h\*10h+0120h=20120h

(3) and ah,es:[si] (es)\*16+(si)=3000h\*10h+0200h=30200h

(4) push 10[bx][di](ds)\*16+(bx+10+di)=1000h\*10h+(0100h+A+0220)=1032Ah

3、下面程序执行后，ax 中的数值为多少？写出分析过程。 assume cs:code

stack segment dw 8 dup (0)

stack ends

code segment

start: mov ax,stack

mov ss,ax mov sp,16

mov ds,ax mov ax,0

call word ptr ds:[0eH] inc ax

inc ax

inc ax

mov ax,4c00h int 21h

code ends end start

答案：

执行 call word ptr ds:[0eH] 指令之前，先将当前 ip 即将 inc ax 这条指令的 ip 放到堆栈中，此时 sp=16-2=0Eh， 正好是 call 指令要跳转到得地址，所以程 序执行 call 之后的第一条指令，即 inc ax，继续向后执行，ax+3=0+3=3

4、设(ds)=8000h, (BX)=2000H, (si)=0002H, (di)=0100H, 从 82000H 开始的内 存单元存放了 12345678H，从 82100H 开始的内存单元存放了 0a0b0c0d0e0fH, 试 说明下列各条指令执行后，ax 寄存器的内容。

(1) mov ax,bx

(2) mov ax,[bx]

(3) mov ax,[2002H]

(4) mov ax,04[bx][di]

答案：

(1) mov ax,bx （ax）=2000H

(2) mov ax,[bx] (ax)=ds:[2000H]=3412h

(3) mov ax,[2002H] (ax)=ds:[2002h]=7856h

(4)mov ax,04[bx][di] (ax)=ds:[bx+di+04]=ds:[2104]=0f0eH

5、一个有 128 个字的数据区，它的起始地址为 12ABH：00ABH，请给出这个数 据区最末一个字单元的物理地址。

答案

|  |  |
| --- | --- |
| 00AB |  |
|  |  |
|  |  |

答：128\*2=256Byte=0FFH

所以最末一个字节的偏移地址是：

00AB+00FF=01AAH

|  |  |
| --- | --- |
|  |  |
|  |  |
| 01AA |  |

所以物理地址是：

12ABH\*16+01AAH=12C5AH

所以最后一个字单元的物理地址是 12C59H

6、分析如下程序：

1000:0 mov ax,8

1000:3 jmp ax

1000:5 mov ax,0

1000:8 mov bx,ax

1000:a jmp bx

CPU 从 1000:0 处开始执行指令当执行完 1000:a 处的指令后几次修改 IP。 请写出分析过程。

答案：

1000:0 mov ax,8 执行后 IP+3=3 IP 第一次改变

1000:3 jmp ax 执行后 IP+2=5 IP 第二次改变，有转移指令，IP 再次改变，IP 第 三次改变

1000:5 mov ax,0

1000:8 mov bx,ax 执行后 IP+2=A IP 第四次改变

1000:a jmp bx 执行后 IP+2=C IP 第五次改变，有转移指令，IP 再次改变，IP 第 六次改变

7、已知 DS=2000H，BX=0100H，SI=0002H，存储单元[20100H]~[20103H]依次存 放 12 34 56 78H，[21200H]~[21203H]依次存放 2A 4C B7 65H，说明下列每条 指令执行后，AX 寄存器的内容。

（1）mov ax,1200H

（2）mov ax,bx

（3）mov ax,[bx]

（4）mov ax,[bx+si]

（5）mov ax,[bx][si+1100h]

答案：

（1）mov ax,1200H ax：1200H

（2）mov ax,bx ax：0100h

（3）mov ax,[bx] ax：3412h

（4）mov ax,[bx+si] ax：7856h

（5）mov ax,[bx][si+1100h] ax: 65b7h

8.将 datasg 段中的每个单词改为大写字母。（提示： datasg 段中每个字符串， 长度都为 16 字符，字母后面都加上了空格。）

答案：

assume cs:code,ds:datasg datasg segment

db ‘ibm ‘

db ‘dec ‘

db ‘dos ‘

db ‘vax ‘

datasg ends

code segment

start: mov ax,datasg

mov ds,ax

mov bx,0

mov cx,4

s0:mov dx,cx mov si,0

mov cx,3

s:mov al,[bx+si] and al,11011111b mov [bx+si],al

inc si

loop s

add bx,16 mov cx,dx loop s0

mov ax,4c00H int 21H

codesg ends end start

9.分析如下程序，填写指令后指定寄存器的值，并描述本段程序的功能，内存 中 2000:0000 开始的连续四个字节的数为“A0 00 03 0B”。

assume cs:code

code segment

start:mov ax,2000h AX：\_\_\_\_\_

mov ds,ax DS：\_\_\_\_\_

mov bx,0 BX：\_\_\_\_\_

s:mov cl,[bx] CL：\_\_\_\_\_

mov ch,0 CH：\_\_\_\_\_

jcxz ok

inc bx BX：\_\_\_\_\_

jmp short s

ok:mov dx,bx DX：\_\_\_\_\_

mov ax,4c00h

int 21h

code ends

end start

10. 编程实现，向内存 0:200~0:23F 依次传送数据 0~63（3FH）。

答案：

assume cs: code

code segment

mov ax,0020h

mov ds,ax

mov cx,64

mov bx,0h

s:mov ds:[bx],bl

inc bl

loop s

mov ax,4c00h

int 21h code ends

end

11、设 CS=2000H，IP=0；DS=1000H， BX=0，计算执行程序的物理地址，以及 mov ax，[BX]指令中源操作数的物理地址。（用十六进制表示）

答案：执行程序的物理地址 CS：IP=2000\*16+0000=20000H

DS：[BX]=1000\*16+0000=10000H

**12、课上提供的示例程序。**

**（1）**

assume cs:codesg,ds:datasg,ss:stacksg

datasg segment

db 'ibm '

db 'dec '

db 'dos '

db 'vax '

datasg ends

stacksg segment ;定义一个段，用来作栈段，容量为 16 个

字节

dw 0,0,0,0,0,0,0,0

stacksg ends

codesg segment

start: mov ax,stacksg

mov ss,ax

mov sp,16

mov ax,datasg

mov ds,ax

mov bx,0

mov cx,4

s0: push cx ;将外层循环的 cx 值压栈

mov si,0

mov cx,3 ;cx 设置为内层循环的次数

s: mov al,[bx+si]

and al,11011111b

mov [bx+si],al

inc si

loop s

add bx,16

pop cx ;从栈顶弹出原 cx 的值，恢复 cx

loop s0 ;外层循环的 loop 指令将 cx 中的计数值减 1

mov ax,4c00h

int 21h

codesg ends

end start

**（2）**

assume cs:code,ds:data

data segment

db 'conversation'

data ends

code segment

|  |  |  |
| --- | --- | --- |
| start: mov | ax,data |  |
| mov | ds,ax |  |
| mov | si,0 | ;ds:si 指向字符串（批量数据）所在空间的首 |
| 地址  mov | cx,12 | ;cx 存放字符串的长度 |

call capital

mov ax,4c00h

int 21h

capital:and byte ptr [si],11011111b ;将字符转化为大写

inc si

loop capital

ret

code ends

end start

**（3）**

计算 data 段中第一组数据的 3 次方，结果保存在后面一组 dword 单元 中。

assume cs:code

data segment

dw 1,2,3,4,5,6,7,8

dd 0,0,0,0,0,0,0,0

data ends

code segment

start:mov ax,data

mov ds,ax

mov si,0 ;ds:si 指向第一组 word 单元

mov di,16 ;ds:di 指向第二组 dword 单元

mov cx,8

s:mov bx,[si] call cube

mov [di],ax

mov [di].2,dx

add si,2 ;ds:si 指向下一个 word 单元

add di,4 ;ds:di 指向下一个 dword 单元

loop s

mov

int cube:mov mul

mul

ret code ends end start

ax,4c00h 21h

ax,bx bx

bx

**（4）**

assume cs:code

code segment

start:mov ax,2000h mov ds,ax

mov bx,0

s:mov cl,[bx] mov ch,0

jcxz ok

inc bx

jmp short s

ok:mov dx,bx

mov ax,4c00h

int 21h code ends

end start

**（5）**

assume cs:codesg

codesg segment

mov ax,4c00h

int 21h

start:mov ax,0

s:nop

nop

mov di,offset s

mov si,offset s2 mov ax,cs:[si]

mov cs:[di],ax

|  |  |  |
| --- | --- | --- |
| s0:jmp  s1:mov | short  ax,0 | s |
| int | 21h |  |
| mov | ax,0 |  |
| s2:jmp  nop | short | s1 |

codesg ends end start

**第四章**

**一、简答**

1、 如果一个程序仅有单个模块，是否无须链接即可直接生成可执行目标文件？

答：不是。 如果一个程序仅有单个模块，也需要进行链接，因为每个模块分别 进行预处理、 编译和汇编而得到可重定位目标文件， 所以在链接之前无法知道 某个模块是否需要和其他模块进行合井 ， 也不知道将要与哪个模块进行合并。 因此 ， 链接之前的所有模块都采用统一的可重定位目标文件格式， 它与最终 的可执行目标文件格式有」些差别， 即使是单个模块， 也不能将可重定位目标 文件直接变成可执行目标文件。 而且， 单个模块也可能会调用库 函数（如数 学函数库、 标准 1/0 函数库等），因此，必须通过链接才能把库函数中的代码和 数据等合并到程序中，以生成可执行目标文件。

2、 如何将多个 C 语言源程序模块组合起来生成一个可执行目标文件？简述从源 程序到可执行机器代码的转换过程？

3、 哪些节组合成只读代码段？哪些节组合成可读写数据段？

答： 可执行目标文件中描述了两种可装入段： 只读代码段和可读写数据段。 只 读代码段对应可执行目标文件中所有代码和只读数据所在的区域， 通常包括 ELF 头、 程序头表以及 .init、 .text 和 .rodata 节。 可读写数据段对应可 执行目标文件中所有可读可写数据所在的区域， 通常包括 .data 和 .bss 节。

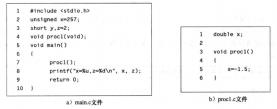
4、 可重定位目标文件和可执行目标文件的主要差别是什么？

5、 静态链接方式下， 静态链接器主要完成哪两方面的工作？

6、 静态链接和动态链接的差别是什么？

**二、综合题**

1、假设一个 C 语言程序有两个源文件：main.c 和 procl.c, 它们的内容如图 4. 所示。



(l)在上述两个文件中出现的符号哪些是强符号？哪些是弱符号？各变量的存储 空间分配在哪个节中？各占几个字节？

(2)程序执行后打印的结果是什么？请分别画出执行第 7 行的 procl()函数调用 前后，在地址&x 和&z 中存放的内容。

(3)若 main.c 的第 3 行改为 “short y= l, z= 2;", 结果又会怎样？

(4 )修改文件 procl, 使得 main.c 能输出正确的结果（即 x= 257,z= 2)。 要求 修改时不能改变任何变量的数据类型和名字。

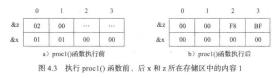
答案：

(1) main.c 中强符号有 x、z 、main, 弱符号有 y 和 procl ; proc l.c 中的强 符号有 procl,弱符号有 x。根据多重定义符号处理规则 2(若一个符号被说明为 一次强符号定义和多次弱符号定义，则按强符号定义为准），符号 x 的定义以 main.c 中的强符号 x 为准，即在main.o 的 data 节中分配 x, 占 4 个字节，随后 是另一个强符号 z 占两个字节，x 和 z 都属于.data 节， 随后是 bss 节，其中 只有一个变量 y, 按 4 字节对齐，因此，在 z 后面有两个字节空闲，再后面是变 量 y 的空间。

(2)程序执行时，在调用procl()函数之前，&x 中存放的是x 的机器数：00000101 H,随后两个字节（地址为&z) 存放 z, 即 000 2H, 再后面两个字节空闲， 如图 4.3a 所示。

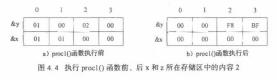
在调用 procl()函数以后， 因为 procl()中的符号 x 是弱符号， 所以， x 的定义以 main 中的强符号 x 为准， 执行 x=-1.5 后， 便将 "-1. 5" 的机器数 BFF80000 00000000H 存放到了&x 开始的 8 个字节中 。 即&x 中为其低 32 位的 00000000H, &z 中为高 32 位的 BFF80000H 中的低 16 位 OOOOH, z 后面的两个空

闲字节中为高 16 位 BFF8H, 如图 4. 3b 所示。



因此， 最终打印的结果为 x=O,z=O 。

( 3 ）若 main.c 的第 3 行改为 “ short y= l,z=2;”则 x、y、z 都是强符号， 都被分配在.data 节中， 因此， x 占 4 个字节 ， 随后是 y 占两个字节， z 占 两个字节， procl 函数执行前、 后的存储内容如图 4.4 所示， 由此可见， x 的机器数为全 0, z 的机器数为BFF8H，因此，最终打印的结果为 x=O,z=-16392。



( 4）只要将文件 procl.c 中的第 l 行修改为 “ static double x” ， 就可以 使得 procl 中的 x 设定为本地变量， 从而在 procl.o 的.d ata 节中专门分配 存放 x 的 8 个字节空间， 而不会和 main 中的 x 共用同一个存储地址， 因此也 就不会破坏 main 中 x 和 z 的值。

**第五章**

**一、简答**

1、 CPU 的基本组成和基本功能各是什么？

2、 一条指令的执行过程中要做哪些事情？

答:一条指令的执行过程包括取指令、指令译码、计算源操作数地址、取操作数、 运算送结果。其中取指令和指令译码是每条指令都必须进行的操作。有些指令需 要到存储单元取操作数，因此需要在取数之前计算操作数所在的存储单元地址。 取操作数和送结果这两个步骤，对于不同的指令，其取和送的地方可能不同，有 些指令要求在寄存器读取/保存数据有些是在内存单元中读取/保存数据，还有些 是对 10 端口进行读取或保存数据。因此，一条指令的执行阶段(不包括取指令 阶段)，可能只有 CPU 参与，也可能要通过总线去访问主存，也可能要通过总线 去访问 I/0 端口。

3、 如何控制一条指令执行结束后能够接着另一条指令执行？

4、 通常一条指令的执行要经过哪些步骤？每条指令的执行步骤都一样吗？

5、 取指令部件的功能是什么？控制器的功能是什么？

6、 怎样保证 CPU 能按程序规定的顺序执行指令呢?

答:计算机的工作过程就是连续执行指令的过程，指令在主存中连续存放。一般 情况下指令被顺序执行，只有遇到转移指令(如无条件转移、条件分支、调用和 返回等指令)才改变指令执行的顺序。当执行到非转移指令时，CPU 中的指令译 码器通过对指令译码，知道正在执行的是一种顺序执行的指令，所以就直接通过 对 PC 加“1”(这里的“1”是指一条指令的长度)来使其指向下一条顺序执行的 指令;当执行到转移指令时，指令译码器知道正在执行的是一种转移指令，因而， 控制运算器根据指令执行的结果进行相应的地址运算，把运算得到的转移目标地 址送到 PC 中，使得执行的下一条指令为转移到的目标指令，

由此可以看出，指令在主存中的存放顺序是静态的，而其执行顺序是动态的。

CPU 能根据指令执行的结果动态改变程序的执行流程。

7、 为什么是 CISC？什么是 RISC？请简单描述各自特点。

8、 数据通路是如何进行数据处理、数据传送、数据存储的?

答:数据通路中的功能部件包括两种不同的逻辑单元，即进行数据处理的组合逻 辑单元称为操作元件)和进行数据存储的时序逻辑单元(称为状态单元)。组合逻 辑单元的输出只取决于当前的输人，也就是说，输人一旦改变，在一定的线路延 迟后，输出就跟着变化，因而它只能完成一定的数据处理功能，不能存储数据， 只有将处理后的新数据送到一个状态单元才能保存下来。所以， 所有数据处理单 元都必须将处理后的输出结果写到状态单元中，而在处理前又必须从状态单元接 收输入。因此， 数据流动的起点是状态单元，经过一些组合逻辑部件，最终又流 到状态单元保存。功能部件之间的输人/出信息通过连线进行传送。

9、 如何确定 CPU 的时钟周期?

答:在一个边沿触发的同步数字系统中，一个状态量的改变总是在时钟的上升沿 或下降沿发生，通常称上升沿或下降沿的到来为时钟有效信号到达。一个状态量 的改变必须满足以下三个条件:①新写人的数据已经生成并稳定在状态单元的输 人端;②写使能信号有效③时钟有效信号到达。在前面两个条件满足的情况下， 一旦时钟有效信号到达，则输入数据开始写人状态单元，经过一定的延迟后，状 态单元的输出变为新输入的数据。所以要能使电路正确工作， 必须使新写人的数 据在时钟有效信号到达前稳定在输入端，也即时钟周期必须足够长，使得在状态 单元之间进行数据处理的组合逻辑电路有足够的时间来得到新的数据因此，应将 所有相邻状态单元之间的组合逻辑电路中最长的延时作为基准来确定时钟周期 以保证在一个时钟周期内所有组合电路能完成必要的数据处理工作，在下个时钟 到来后，数据能存储在状态单元中。

10、 采用流水线方式能使一条指令的执行时间更短吗?

答:不能。采用流水线方式使得指令吞吐率提高了，即在给定的时间内完成指令 执行的条数增加了，但每条指令的执行过程没有减少，因此不会缩短一条指令的 执行时间，相反，流水线方式还会延长一条指令的执行时间。

11、 指令流水线的简单原理是什么？

**二、综合题**

1、某计算机主频为 800MHz，其 CPU 采用三级时序（机器周期－节拍－脉冲）进 行定时， 为单脉冲节拍方式，每个机器周期的基本宽度为 4 个节拍。该计算机 每个指令周期平均有 5 个机器周期，并平均访问 2 次主存，没有设置 cache。 请回答下列问题：

(1）若采用异步方式访问内存，每个“存储器读 ”机器周期平均需 8 个节拍， 每 个“存储器写”机器周期平均需 6 个节拍，则执行一条指令的平均时间为多少？ MIPS 数为多少？平均 CPI 为多少？

(2）若采用同步方式访问内存，每个“存储器读 ”机器周期需在基本宽度的基础 上再插入个4“等待 ” 状态， 每个 “存储器写 ” 机器周期无需 “等待 ” 状 态， 则执行一条指令的平均时间为多少？MIPS 数为多少？平均 CPI 为多少？ （提示： 一 等待状态 ” 即是一个节拍）

答案：

早期的计算机中没有 cache, CPU 访存时，由于主存速度慢，无法像访问寄 存器或 cache 那样能在一个节拍内存取好数据，因此，需要 CPU 等待若干个节 拍才能完成存储访问。 因而“存储器读 ”和“存储器写 ”机器周期比基本的无 访存操作的机器周期长。

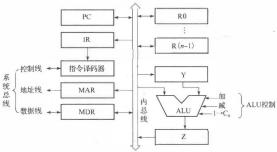
(1) CPU 和主存之间采用异步方式通信时，CPU 每次发送读或写命令后，便在随 后的每个节拍内采样主存送来的“存储操作完成 ”（MFC）信号， 以便在主存数 据准备好时， 到存储器数据寄存器（MDR）中取数据。由题意可知， 在异步方式 下，每条指令的平均时钟周期（节拍）数为（5-2）×4+2×（（8+6)/2)=26，所以 CPl=26。 执行一条指令的平均时间为 26×1/(800×106Hz)=32.5ns。 MIPS 数约 为 800/26=3l。

(2) CPU 和主存之间采用同步方式通信时，CPU 每次发送读或写命令后，便按照 同步通信协议，等待一个固定长度的时间（若干个节拍）后，到存储器数据寄存 器（MDR）中取数据。由题意可知，在同步方式下，每条指令的平均时钟周期（ 节 拍）数为（5-2) × 4+2 × ((4+4+4)/2)=24，所以 CPI=24。执行一条指令的平均 时间为 24×l/(800×l06Hz)=30ns。MIPS 数约为 800/24=33。

2、假定在如图所示的单总线数据通路中，总线传输延迟和 ALU 运算时间分别是 20ps 和 200ps，寄存器建立时间为 1Ops， 寄存器保持时间为 5ps， 寄存器的锁 存延迟（ Clk-to-Q time）为 4ps，控制信号的生成延迟（ Clk-to-signal time） 为 7ps， 三态门接通时间为 3ps,则从当前时钟到达开始算起， 完成以下操作的 最短时间是多少？各需要几个时钟周期？

(1）将数据从一个寄存器传送到另一个寄存器。

(2）将程序计数器 PC 加 1。



答案：

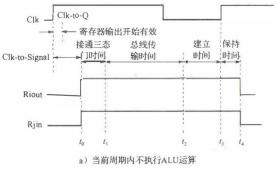
如图所示的数据通路中，所有与内部总线相连的寄存器都有相应的 Rin 和 Rout 控制信号，以控制总线和寄存器之间的数据传送。总线和 ALU 输入端之间、 Y 寄存器与 ALU 输入端之间都无需控制信号。ALU 输出与 Z 寄存器之间可以有控 制信号 Zin，也可以没有。若没有 Zin，则每来一个时钟，ALU 的输出总是被写入 Z 寄存器。以下说明中， 为了明显表示 ALU 结果送至 Z 寄存器，假定有控制信号 Zin。

图给出了单总线数据通路中主要路径的定时。时钟边沿到达后，经过 Clk- to-Q time 的延时， 寄存器中的内容被读出， 同时， 在指令译码器中的控制 逻辑生成当前时钟周期内所需的控制信号， 其延时为 Clk-to-signal time。 随 后，由生成的控制信号 Riout 接通三态门并使寄存器数据在总线上传输。若当前 时钟周期内不做 ALU 运算而是直接在寄存器之间传送， 则总线上的数据在 Rjin

的控制下直接被送到目的寄存器时的输入端，在下一个时钟边沿到来之前，总线 上的数据必须继续稳定一段寄存器建立时间，以使寄存器 Rj 的输入在这段建立 时间内保持不变，如图 5.2a 所示；若当前时钟周期内有 ALU 运算， 则还需 ALU 电路延时， 最后 ALU 结果直接送 Z 寄存器，在下一个时钟边沿到来之前，ALU 的 输出必须继续稳定一段寄存器建立时间，以使寄存器 Z 的输入在这段建立时间内 保持不变，如图 b 所示。

（1）由图 a 可知，寄存器之间进行传送的时间延迟至少为 7ps+3ps+20ps+ 1Ops=40ps。在这个寄存器数据传送过程中，只需要在一个寄存器中保存信息， 因此只需要一个时钟周期就可完成该操作。

（2）将 PC 中的内容加 l 送 PC，被分解成以下两个过程：PC 加 l 送 Z、 Z 送 PC。对于第一个过程，由图 2b 可知，其延迟至少为 7ps+3ps+20ps+200ps+1 Ops=240ps；第二个过程实现的是寄存器之间的传送，因此延迟至少为 40ps。因 为在该操作过程中保存了两次信息，所以需要两个时钟周期才能完成该操作。





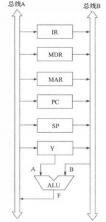
3、图给出了某 CPU 内部结构的一部分，MAR 和 MDR 直接连到存储器总线（图中 省略）。在两个总线之间的所有数据传送都需经过算术逻辑部件 ALU。ALU 的部分 功能及其控制信号如下。

MOVa:F=A; MOVb:F=B;

a+1:F=A+1; b+l:F=B+1

a+1:F=A-1; b-1:F=B-1

其中，A 和 B 是 ALU 的输入，F 是 ALU 的输出。假定调用指令 CALL 占两个 字，第一个字是操作码，第二个字给出过程（或子程序 ）的起始地址，返回地 址保存在主存的钱中， 用 SP(栈顶指针寄存器）指向栈顶处的空元素，栈从高地 址向低地址方向增长（自顶向下），按字编址，每次按同步方式从主存读取一个 字。 请写出读取并执行 CALL 指令所要求的控制信号序列，并说明至少需要多少 个时钟周期。（提示：当前指令地址在 PC 中 ）



答案：

因为采用同步方式读写内存，所以在 read 和 write 信号后不需加等待信号 WMFC。CALL 指令有两个字，按字编址，每次从主存读取一个字， 因此，CALL 指 令需要读两次主存，一次是读取指令中的操作码，另一次是读取指令中给出的子 程序首地址。其指令周期分为以下三个阶段。

（1）读取指令操作码：将 PC 的内容作为地址访问 存储器，取出指令的操作码， 送指令寄存器 IR，同时 PC +l 送 PC，以指向指令的第二个字，至少需要 3 个时 钟周期（节拍）。

PCout,MOVb,MARin

Read,b+1,PCin

MDRout,MOVb,IRin

（2）取子程序首地址：将 PC 的内容作为地址，取出指令的第二个字（即子程

序人口地址）送 PC，以使下一个指令周期从子程序的第一条指令开始执行。同

时，计算 PC＋1 以得到返回地址，送 Y 寄存器，至少需要 3 个时钟周期。

Cout,MOVb,MARin

Read,b+1,Yin

MDRout,MOVb,PCin

（3）保存返回地址至栈中：将临时保存在 Y 寄存器的返回地址送到战顶保存， 并自动调整栈顶指针，因为是“ 自顶向下 ”栈，因此栈顶指针需要进行－1操 作。 这个过程至少需要 3 个时钟周期。

SPout,MOVb,MARin

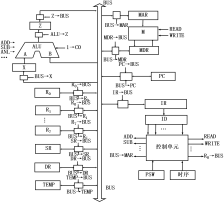
Yout, MOVb,MDRin

write,SPout,b-1,Spin

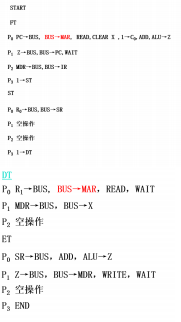
显然， 上述每个节拍中执行的操作所需要时间不等，其中，存储访问 （read/write）时间最长，时钟周期以最长的存储访问 时间为准，CALL 指令的 指令周期至少有 9 个时钟周期（节拍）。

如果将第一次PC+l 的结果送到 Y 寄存器，第二阶段以 Y 的内容作为地址访 问主存，并继续对 Y 寄存器加 1，结果送 PC，则也能实现 CALL 指令的功能。这 种方式下，也是 9 个时钟周期。

4、以下图计算机系统模型为例，说明实现 ADD [R1]，R0(计算[R1]与R0 相 加，结果送回[R1]的微操作序列。



答：



**第六章**

**一、简答**

1、计算机内部为何要采用层次结构存储体系？层次结构存储体系如何构成？

2、SRAM 芯片和 DRAM 芯片各有哪些特点？各自用在哪些场合？

3、为什么在 CPU 和主存之间引人 cache 能提高 CPU 的访存效率？

4、什么是物理地址？什么是逻辑地址？地址转换由硬件还是软件实现？为什么？

5、在存储器层次化结构中，“cache—主存 ”、“ 主存—磁盘 ”这两个层次有哪 些不同？

6、分段和分页存储管理有何区别？

答案：

（1）页是信息的物理单位，分页是为了实现离散分配方式，以消减内存的外部 零头，提高内存利用率。段则是信息的逻辑单位，它含有一组相对完整的信息。

（2）页的大小固定且由系统决定，由系统把逻辑地址划分为页号和页内地址两 部分，是由机械硬件实现的,因而在系统中只能有一种大小的的页面 ;而段的长度 却不固定,决定于用户所编写的程序,通常由编译程序在对原程序进行编译时,根 据信息的性质来划分。

（3）分页的作业地址空间是一维的,而分段作业地址空间则是二维的。

7、CPU 执行指令进行一次存储访问操作要存取几次内存?

答:在具有 cache 并采用动态重定位存储管理的系统中，其大致过程如下:①根据 虚页号查快表，若快表中有对应页的页表项，则取出页框号形成物理地址，转若 快表中不存在对应页的页表项，则发生 TLB 缺失，转③。

②判断物理地址中的标记是否和 cache 中标记相等并且有效位是否为 1，若为 1， 则 cache 命中，从 cache 取数或写数据到 cache(直写方式同时也写主存);若为

0，则发生 cache 缺失，转④。

③当 TLB 缺失时，根据页表基址寄存器的值和虚页号找到主存中的页表项，判断

有效位是否为 1，若为 1，则说明该页在主存中，把该页的页表项装人 TLB 中， 并取出页框号形成物理地址，转②;若不是，则说明该页不在主存中，发生了缺 页异常。此时， 需要调出操作系统中的缺页异常处理程序，实现从磁盘读入一个 页面的功能。缺页处理结束后， 重新执行当前指令。此时， 一定能在主存中找到 需访问的信息。

④当 cache 缺失时，CPU 根据物理地址到主存读一块信息到 cache，然后再取到 CPU 或 CPU 写信息到 cache。

从上述过程来看，CPU 进行一次存储访问操作，最好的情况下(TLB 命中、cache 命中)，无需访问内存 ;最坏的情况下(缺页)，不仅要多次访问内存，还要读写盘 数据。

8、存储器分层结构中，各层次上存储器的速度如何?

答:在计算机系统中，存储器采用的是一种分层结构，包括寄存器-cache-主存一 磁盘。若用一个 CPU 时钟周期来表示它们之间的相对速度，则 1 个时钟周期不到 就能从寄存器访问到信息;1~10 个时钟周期能够在 cache 中访问到信息;50~100 个时钟周期能在主存中访问到信息 ;如果要从磁盘读信息的话，则大约需要几十 万到几百万个时钟周期。因此， 程序员必须能够充分理解存储器的分层结构。随 着技术的进步，各类存储器的速度可能会发生变化，但它们之间的差异总是存在 的。

**二、综合题**

1、假定某计算机的主存地址空间大小为 512MB，按字节编址。若每次读写操作最 多可以存取 32 位，则存储器地址寄存器 MAR 和存储器数据寄存器 MDR 的位数至 少分别为多少？

答案：

主存地址空间大小为 512MB，按字节编址，说明每个存储单元有 8 位，共有 512M=1029 个存储单元。所以，地址位数至少应有 29 位，故存放主存地址的存储 器地址寄存器 MAR 至少应有 29 位。每次读写最多存取 32 位，因此，用来作为读 /写数据缓冲的存储器数据寄存器 MDR 的位数至少应有 32 位。

2、假定有一个磁盘存储器，磁盘片外径为 355.6mm，有 20 个记录面，每面有 51mm 区域用于记录信息，道密度为 3.92TPM（道/mm ），位密度为 90BPM ( bit/mm）， 转速为 2400RPM,道间移动时间为 0.2ms。请问：

（1）磁盘容量约为多少？如果道密度和位密度同时扩大 100 倍，则容量约为多 少？

（2）平均存取时间和平均数据传输率各为多少？

（3）如果在道密度和位密度同时扩大 100 倍的同时转速扩大 3 倍，则平均存取 时间和平均数据传输率各为多少？

答案：

（1）每面磁道数为 51mm×3.92TPM=200 道，给出的位密度是指最内圈上的位密 度。所以最内圈周长为 3.14×（355.6-2×51) = 796.3mm，故每道信息量为 796.3mmx90BPM = 71664bit. 因此， 磁盘容量为 20 x 200 x 71664bit=286656 OOObit=273Mbit(lM=220)。若道密度和位密度同时扩大 100 倍 则磁道数扩大 100 倍，每道容量扩大 100 倍，所以整个盘组容量扩大 1O000 倍。磁盘容量大约为 273M x 104bit = 2666Gbit = 333GB(lG=230）。

（ 2 ）平均寻道时间为［（ 200-1 ） ×0.2+0]/2=19.9 ；转 一 圈的时间为 60×l000/2400RPM= 25ms,平均（旋转）等待时间为（25ms+0)/2=12.5ms。平均 存取时间为平均寻道时间与平均寻道时间之和，即 l9.9ms+ 12.5ms=32.4ms。平 均数据传输率为 R=71664bit x 103/25=2.87Mbit/s ( 1M=l06）。

（3）道密度扩大 100 倍，则平均寻道时间约为（19999×0.002+0)/2=20ms；转 速扩大 3 倍转一 圈的时间缩短为 25ms/3=8.33ms ，平均等待 时 间缩短为

8.33ms/2=4.2ms，故平均存取时间为 20ms+4.2ms=24.2ms。位密度扩大 100 倍， 则每道容量扩大 100 倍，转速扩大 3 倍，则转一圈的时间缩短为 1/3，因此，平 均数据传输率共扩大 100×3=300 倍。即平均数据传输率为 300×2.87Mitb/s=86 lMbit/s。

4、什么是段式管理?其与页式管理的有何区别（请至少举例说明三点）? （段式 管理即分段存储管理，页式管理即分页存储管理）

答案：段式管理就是将程序按照内容或过程(函数)关系分成段，每段拥有自己的

名字。一个用户作业或进程所包含的段对应于一个二维线性虚拟空间，也就是一 个二维虚拟存储器。段式管理程序以段为单位分配内存， 然后通过地址映射机构 把段式虚拟地址转换成实际的内存物理地址。同页式管理类似， 段式管理也采用 只把那些经常访问的段驻留内存，而把那些在将来—段时间内不被访问的段放入 外存，待需要时自动调入相关段的方法实现二维虚拟存储器。

段式管理和页式管理的主要区别有：

(1)页式管理中源程序进行编译链接时是将主程序、子程序、数据区等按照 线性空间的—维地址顺序排列起来。段式管理则是将程序按照内容或过程(函数) 关系分成段，每段拥有自己的名字。一个用户作业或进程所包含的段对应于一个 二维线性虚拟空间，也就是一个二维虚拟存储器。

(2)当实现虚拟存储功能时，段式管理与页式管理有所不同。段式虚存每次 交换的是一段有意义的信息；而不是像页式虚存管理那样只交换固定大小的页， 需要多次的缺页中断才能把所需信息完整地调入内存。

(3)在段式管理中，段长可根据需要动态增长。这对那些需要不断增加或改 变新数据或子程序的段来说，将是非常有好处的。

(4)段式管理便于对具有完整逻辑功能的信息段进行共享。

(5)段式管理便于进行动态链接，而页式管理进行动态链接的过程非常复杂。

**存储扩展的并联、串联、混联三种方法**

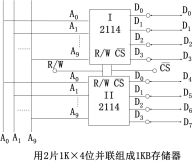
5、用 2114（1K×4 位）芯片并联构成 1KBRAM

答：

（1）计算所需要 2114 的芯片数

所需芯片数=（RAM 的容量）/单个芯片的容量=1KB/（1K×4）=2

（2）画出连线图。图用 2 片 1K×4 位并联组成 1KB 存储器



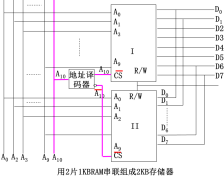
6、用 1KBRAM 芯片串联构成 2KBRAM

答：

（1）计算所需的芯片数

芯片数=（RAM 的容量）/单个芯片的容量=2KB/1K=2；用 2 片 （2）画连线图。

用 2 片 1KBRAM 串联组成 2KB 存储器。



7、用 2114（1K×4）芯片构成 2KBRAM 存储器 解：

（1）计算所需的芯片数

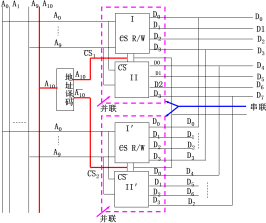
芯片数=（RAM 的容量）/单个芯片的容量=（2K×8）/（1K×4）=4

（2）分组

由于每个芯片只有 1K 空间，而 2K 的地址空间需要 2 组，每组需要 2 个芯片。

（3）画连线图。

先将芯片两两并联分成两组，每组存储容量都是 1KB（方法同并联）。再将两个 1KB 的组按照串联的方法扩展到 2KB。



9 、 在 一 个 请 求 分 页 存 储 管 理 系 统 中 ， 一 个 作 业 的 页 面 走 向 为 4,3,2,1,4,3,5,4,3,2,1,5，当分配给作业的物理块数分别为 3 和 4 时，试计算 采用下述页面淘汰算法时的缺页率(假设开始执行时主存中没有页面)，并比较结 果。

1)最佳置换算法;

2)先进先出置换算法 ;

3)最近最久未使用算法。

解：



**第七、八章**

**一、简答**

1、引起异常控制流的事件主要有哪几类？

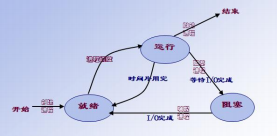
2、什么是程序？什么是进程？二者有什么区别？

3、进程有几种状态？并说明进程在不同状态之间转换的过程和原因。

答案：基本为三态（运行、就绪、阻塞） 或五态（运行、就绪、阻塞、创建、结 束）



三态图



五态图

4、什么是线程?试述线程与进程的区别。

答案：线程是在进程内用于调度和占有处理机的基本单位，它由线程控制表、存 储线程上下文的用户栈以及核心栈组成。线程可分为用户级线程、核心级线程以

及用户／核心混合型线程等类型。其中用户级线程在用户态下执行， CPU 调度算 法和各线程优先级都由用户设置，与操作系统内核无关。核心级线程的调度算法 及线程优先级的控制权在操作系统内核。混合型线程的控制权则在用户和操作系 统内核二者。线程与进程的主要区别有：

(1)进程是资源管理的基本单位，它拥有自己的地址空间和各种资源，例如内存 空间、外部设备等； 线程只是处理机调度的基本单位，它只和其他线程一起共享 进程资源，但自己没有任何资源。

(2)以进程为单位进行处理机切换和调度时，由于涉及到资源转移以及现场保护 等问题，将导致处理机切换时间变长，资源利用率降低。以线程为单位进行处理 机切换和调度时，由于不发生资源变化，特别是地址空间的变化，处理机切换的 时间较短，从而处理机效率也较高。

(3)对用户来说，多线程可减少用户的等待时间。提高系统的响应速度。例如， 当一个进程需要对两个不同的服务器进行远程过程凋用时，对于无线程系统的操 作系统来说需要顺序等待两个不同调用返回结果后才能继续执行，且在等待中容 易发生进程调度。对于多线程系统而言， 则可以在同一进程中使用不同的线程同 时进行远程过程调用，从而缩短进程的等待时间。

(4)线程和进程一样，都有自己的状态．也有相应的同步机制，不过，由于线程 没有单独的数据和程序空间，因此，线程不能像进程的数据与程序那样，交换到 外存存储空间。从而线程没有挂起状态。

(5)进程的调度、同步等控制大多由操作系统内核完成，而线程的控制既可以由 操作系统内核进行，也可以由用户控制进行。

5、 试比较 FCFS 和 SPF 两种进程调度算法。

答案：

相同点：两种调度算法都可以用于作业调度和进程调度。

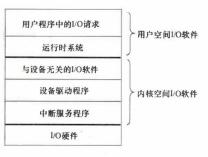
不同点：FCFS 调度算法每次都从后备队列中选择一个或多个最先进入该队列的 作业，将它们调入内存、分配资源、创建进程、插入到就绪队列。该算法有利于 长作业/进程，不利于短作业/进程。 SPF 算法每次调度都从后备队列中选择一 个或若干个估计运行时间最短的作业，调入内存中运行。该算法有利于短作业/ 进程，不利于长作业/进程。

6、简述异常和中断事件形成异常控制流的过程。

7、I/O 子系统的层次结构是怎样的？

答案：

I/O 子系统包含 I/O 软件和 I/0 硬件两大部运行时系统分。I/0 软件包括最 上层提出 I/0 请求的用户空间与设备无关的 I/0 软件（称为用户 I/0 软件）和在 底层操作系统中对 I/0 进行具体管理和控制的内核空间 I/0 软件（称为系统 I/0 软件），系统 I/0 软件又分三个层次，分别是与设备无关的 I/0 软件层、设备驱 动程序层和中断服务程序层。I/0 硬件在操作系统的控制下完成具体的I/0 操作。



8、机器字长、编址单位、存取宽度、传输宽度、指令字长各指什么?它们之间有 何关系?

答:在计算机内部，有指令和数据两大类信息。指令和数据都以二进制形式存放 在存储器中，运行程序时，需要把指令和数据从存储器读出，通过总线传输到 CPU， 然后 CPU 再通过执行指令来对操作数进行相应的运算，最后把结果数据送到寄存 器或存储器中。所以， 在设计或使用计算机过程中，要涉及很多问题，例如，指 令和数据在存储器中按什么长度存放 ;写人或读出时按什么长度存取 ;在总线上 传输时同时传送多少位 ;数据和指令送到 CPU 后，在 CPU 的寄存器中按多少位存 放;在运算器中按多少位来运算，等等。因而，出现了一系列有关信息单位和信 息宽度的概念，这些概念非常重要，但比较容易混淆，需要将很多知识和概念联 系在一起，才能很好地理解这些概念及其相互之间的关系上述概念的定义和关系

说明如下。

(1)机器字长是计算机一个非常重要的指标。通常称 32 位机器或 64 位机器，就 是指机器的字长是 32 位或 64 位。一般情况下，机器字长定义为 CPU 中在同一时 间内一次能够处理的二进制数的位数，实际上就是 CPU 中定点运算数据通路的位 数。在计算机中， “字 ”的概念经常出现。一个字的宽度并不等于机器字长。字 作为机器中所有信息宽度的计量单位对于某个系列机来说，其字宽总是固定的。 例如，在 80x86 系列中，一个字的宽度为 16 位，因此 32 位是双字，64 位是四

字。在 IBM303X 系列中，一个字的宽度为 32 位，所以 16 位 为半字，32 位为单字，64 位为双字。

(2)编址单位就是存储单元的宽度，指存储器中具有相同地址的若干个存储元件 (或称存储元、存储基元、记忆单元)构成的一个二进制代码的宽度，可以是 8 位、 16 位、32 位等现代大多数计算机按字节编址，即编址单位为 8 位，每一个字节 有一个地址。由此可见， 一个数据(如 32 位整数、32 位浮点数或 64 位浮点数等) 可能占多个存储单元，CPU 要求一次从存储器读出或写人的信息也可能有多个存 储单元。

(3)存取宽度是指一次从一个由多个 DRAM 芯片构成的存储模块中同时读写的信 息的宽度，例如，假定某个存储模块由 8 个 4096x4096x8 位的 DRAM 芯片按交叉 编址方式构成，则该存储模块的存取宽度是 64 位，也即 8 个芯片可同时读写， 每个芯片同时读 8 位，因而最多可以同时存取 64 位信息。

(4)传输宽度就是总线宽度，也就是一次最多能在总线上传输的数据位数。对于 存储器总线来说，总线上传输的信息宽度应该等于存储器的存取宽度。因此， 在 设计系统时，应考虑传输宽度和存取宽度的匹配，并且每个设备中的总线接口部 件也要与这些宽度匹配

(5)指令字长是指指令的位数。有定长指令字机器和不定长指令字机器。定长指 令字机器的所有指令的位数是相同的，目前定长指令字大多是 32 位指令字 ;不定 长指令字机器的指令有长有短，但每条指令的长度一般都是 8 的倍数。因此， 一 个指令字在存储器中存放时，可能占用多个存储单元 ;从存储器读出并通过总线 传输时，可能分多次进行，也可能一次读多条指令。

**二、综合题**

1、举例说明程序并发执行时为什么会失去封闭性和可再现性?

答案：因为程序并发执行时，多个程序共享系统中的各种资源，资源状态需要多 个程序来改变，即存在资源共享性使程序失去封闭性；而失去了封闭性导致程序 失去可再现性。例如：

