

种寻址方式并不常用。一般问到扩大寻址范围时，通常指的是寄存器间接寻址。

5. 寄存器寻址

寄存器寻址是指在指令字中直接给出操作数所在的寄存器编号，即 $EA = R_i$ ，其操作数在由 R_i 所指的寄存器内，如图 4.6 所示。

寄存器寻址的优点是指令在执行阶段不访问主存，只访问寄存器，因寄存器数量较少，对应地址码长度较小，使得指令字短且因不用访存，所以执行速度快，支持向量/矩阵运算；缺点是寄存器价格昂贵，计算机中的寄存器个数有限。

6. 寄存器间接寻址

寄存器间接寻址是指在寄存器 R_i 中给出的不是一个操作数，而是操作数所在主存单元的地址，即 $EA = (R_i)$ ，如图 4.7 所示。

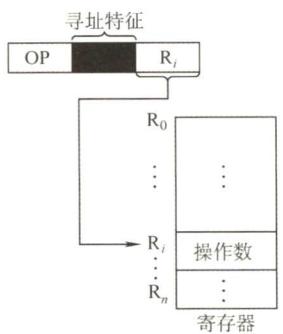


图 4.6 寄存器寻址

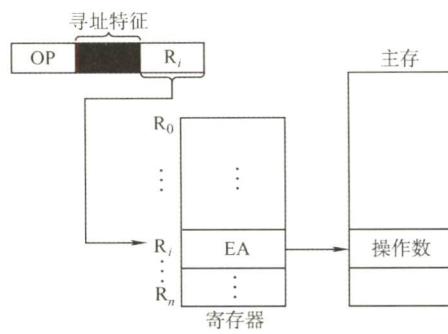


图 4.7 寄存器间接寻址

寄存器间接寻址的特点是，与一般间接寻址相比速度更快，但指令的执行阶段需要访问主存（因为操作数在主存中）。

7. 相对寻址

相对寻址是把 PC 的内容加上指令格式中的形式地址 A 而形成操作数的有效地址，即 $EA = (PC) + A$ ，其中 A 是相对于当前 PC 值的位移量，可正可负，补码表示，如图 4.8 所示。

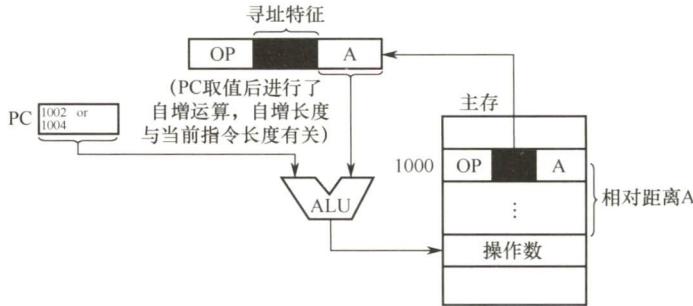


图 4.8 相对寻址

在图 4.8 中，A 的位数决定操作数的寻址范围。

相对寻址的优点是操作数的地址不是固定的，它随 PC 值的变化而变化，且与指令地址之间总是相差一个固定值，因此便于程序浮动。相对寻址广泛应用于转移指令。

注意，对于转移指令 JMP A，当 CPU 从存储器中取出一字节时，会自动执行 $(PC) + 1 \rightarrow PC$ 。

若转移指令的地址为 X，且占 2B，在取出该指令后，PC 的值会增 2，即 $(PC) = X + 2$ ，这样在执行完该指令后，会自动跳转到 $X + 2 + A$ 的地址继续执行。

8. 基址寻址

基址寻址是指将 CPU 中基址寄存器 (BR) 的内容加上指令格式中的形式地址 A 而形成操作数的有效地址，即 $EA = (BR) + A$ 。其中基址寄存器既可采用专用寄存器，又可采用通用寄存器，如图 4.9 所示。

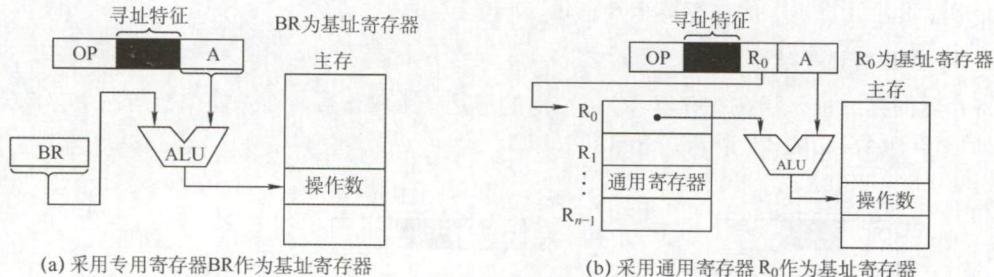


图 4.9 基址寻址

基址寄存器是面向操作系统的，其内容由操作系统或管理程序确定，主要用于解决程序逻辑空间与存储器物理空间的无关性。在程序执行过程中，基址寄存器的内容不变（作为基址），形式地址可变（作为偏移量）。采用通用寄存器作为基址寄存器时，可由用户决定哪个寄存器作为基址寄存器，但其内容仍由操作系统确定。

基址寻址的优点是可扩大寻址范围（基址寄存器的位数大于形式地址 A 的位数）；用户不必考虑自己的程序存于主存的哪个空间区域，因此有利于多道程序设计，并可用于编制浮动程序，但偏移量（形式地址 A）的位数较短。

9. 变址寻址

变址寻址是指有效地址 EA 等于指令字中的形式地址 A 与变址寄存器 IX 的内容之和，即 $EA = (IX) + A$ ，其中 IX 为变址寄存器（专用），也可用通用寄存器作为变址寄存器。图 4.10 所示为采用专用寄存器 IX 的变址寻址示意图。

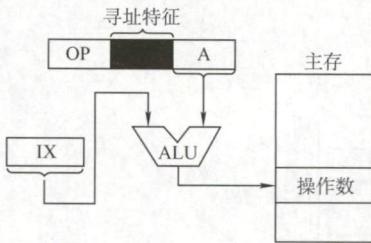


图 4.10 变址寻址

变址寄存器是面向用户的，在程序执行过程中，变址寄存器的内容可由用户改变（作为偏移量），形式地址 A 不变（作为基址）。

变址寻址的优点是可扩大寻址范围（变址寄存器的位数大于形式地址 A 的位数）；在数组处理过程中，可设定 A 为数组的首地址，不断改变变址寄存器 IX 的内容，便可很容易形成数组中任意一个数据的地址，特别适合编制循环程序。偏移量（变址寄存器 IX）的位数足以表示整个存储空间。

显然，变址寻址与基址寻址的有效地址形成过程极为相似。但从本质上讲，两者有较大区别。基址寻址面向系统，主要用于为多道程序或数据分配存储空间，因此基址寄存器的内容通常由操作系统或管理程序确定，在程序的执行过程中其值不可变，而指令字中的 A 是可变的；变址寻址立足于用户，主要用于处理数组问题，在变址寻址中，变址寄存器的内容由用户设定，在程序执行过程中其值可变，而指令字中的 A 是不可变的。

10. 堆栈寻址

堆栈是存储器（或专用寄存器组）中一块特定的、按后进先出（LIFO）原则管理的存储区，该存储区中读/写单元的地址是用一个特定的寄存器给出的，该寄存器称为堆栈指针（SP）。堆栈可分为硬堆栈与软堆栈两种。

寄存器堆栈又称硬堆栈。寄存器堆栈的成本较高，不适合做大容量的堆栈；而从主存中划出一段区域来做堆栈是最合算且最常用的方法，这种堆栈称为软堆栈。

在采用堆栈结构的计算机系统中，大部分指令表面上都表现为无操作数指令的形式，因为操作数地址都隐含使用了 SP。通常情况下，在读/写堆栈中的一个单元的前后都伴有自动完成对 SP 内容的增量或减量操作。

下面简单总结寻址方式、有效地址及访存次数（不含取本条指令的访存），见表 4.1。

表 4.1 寻址方式、有效地址及访存次数

寻址方式	有效地址	访存次数
隐含寻址	程序指定	0
立即寻址	A 即是操作数	0
直接寻址	$EA = A$	1
一次间接寻址	$EA = (A)$	2
寄存器寻址	$EA = R_i$	0
寄存器间接一次寻址	$EA = (R_i)$	1
相对寻址	$EA = (PC) + A$	1
基址寻址	$EA = (BR) + A$	1
变址寻址	$EA = (IX) + A$	1

4.2.3 本节习题精选

一、单项选择题

01. 指令系统中采用不同寻址方式的目的是（ ）。
- 提供扩展操作码的可能并降低指令译码难度
 - 可缩短指令字长，扩大寻址空间，提高编程的灵活性
 - 实现程序控制
 - 三者都正确
02. 直接寻址的无条件转移指令的功能是将指令中的地址码送入（ ）。
- 程序计数器（PC）
 - 累加器（ACC）
 - 指令寄存器（IR）
 - 地址寄存器（MAR）
03. 为了缩短指令中某个地址段的位数，有效的方法是采取（ ）。
- 立即寻址
 - 变址寻址
 - 基址寻址
 - 寄存器寻址
04. 简化地址结构的基本方法是尽量采用（ ）。
- 寄存器寻址
 - 隐地址
 - 直接寻址
 - 间接寻址
05. 在指令寻址的各种方式中，获取操作数最快的方式是（ ）。
- 直接寻址
 - 立即寻址
 - 寄存器寻址
 - 间接寻址
06. 假定指令中地址码所给出的是操作数的有效地址，则该指令采用（ ）。
- 直接寻址
 - 立即寻址
 - 寄存器寻址
 - 间接寻址
07. 设指令中的地址码为 A，变址寄存器为 X，程序计数器为 PC，则变址间址寻址方式的



- 操作数的有效地址 EA 是 ()。
- $((PC) + A)$
 - $((X) + A)$
 - $(X) + (A)$
 - $(X) + A$
08. () 便于处理数组问题。
- 间接寻址
 - 变址寻址
 - 相对寻址
 - 基址寻址
09. 堆栈寻址方式中, 设 A 为累加器, SP 为堆栈指示器, M_{sp} 为 SP 指示的栈顶单元。若进栈操作的动作是 $(A) \rightarrow M_{sp}, (SP) - 1 \rightarrow SP$, 则出栈操作的动作应为 ()。
- $(M_{sp}) \rightarrow A, (SP) + 1 \rightarrow SP$
 - $(SP) + 1 \rightarrow SP, (M_{SP}) \rightarrow A$
 - $(SP) - 1 \rightarrow SP, (M_{SP}) \rightarrow A$
 - $(M_{SP}) \rightarrow A, (SP) - 1 \rightarrow SP$
10. 相对寻址方式中, 指令所提供的相对地址实质上是一种 ()。
- 立即数
 - 内存地址
 - 以本条指令在内存中首地址为基准位置的偏移量
 - 以下条指令在内存中首地址为基准位置的偏移量
11. 在多道程序设计中, 最重要的寻址方式是 ()。
- 相对寻址
 - 间接寻址
 - 立即寻址
 - 按内容寻址
12. 指令寻址方式有顺序和跳跃两种, 采用跳跃寻址方式可以实现 ()。
- 程序浮动
 - 程序的无条件浮动和条件浮动
 - 程序的无条件转移和条件转移
 - 程序的调用
13. 某机器指令字长为 16 位, 主存按字节编址, 取指令时, 每取一字节, PC 自动加 1。当前指令地址为 2000H, 指令内容为相对寻址的无条件转移指令, 指令中的形式地址为 40H。则取指令后及指令执行后 PC 的内容为 ()。
- 2000H, 2042H
 - 2002H, 2040H
 - 2002H, 2042H
 - 2000H, 2040H
14. 对按字寻址的机器, 程序计数器和指令寄存器的位数各取决于 ()。
- 机器字长, 存储器的字数
 - 存储器的字数, 指令字长
 - 指令字长, 机器字长
 - 地址总线宽度, 存储器的字数
15. 假设寄存器 R 中的数值为 200, 主存地址为 200 和 300 的地址单元中存放的内容分别是 300 和 400, 则 () 方式下访问到的操作数为 200。
- 直接寻址 200
 - 寄存器间接寻址 (R)
 - 存储器间接寻址 (200)
 - 寄存器寻址 R
16. 假设某条指令的第一个操作数采用寄存器间接寻址方式, 指令中给出的寄存器编号为 8, 8 号寄存器的内容为 1200H, 地址为 1200H 的单元中的内容为 12FCH, 地址为 12FCH 的单元中的内容为 38D8H, 而地址为 38D8H 的单元中的内容为 88F9H, 则该操作数的有效地址为 ()。
- 1200H
 - 12FCH
 - 38D8H
 - 88F9H
17. 设相对寻址的转移指令占 3B, 第一节为操作码, 第二、三字节为相对位移量 (补码表示), 而且数据在存储器中采用以低字节为字地址的存放方式。每当 CPU 从存储器取出一字节时, 即自动完成 $(PC) + 1 \rightarrow PC$ 。若 PC 的当前值为 240 (十进制), 要求转移到 290 (十进制), 则转移指令的第二、三字节的机器代码是 (); 若 PC 的当前值为 240 (十进制), 要求转移到 200 (十进制), 则转移指令的第二、三字节的机器代码是 ()。
- 2FH, FFH
 - D5H, 00H
 - D5H, FFH
 - 2FH, 00H

关注公众号【乘龙考研】
一手更新 稳定有保障

18. 关于指令的功能及分类，下列叙述中正确的是（ ）。
- 算术与逻辑运算指令，通常完成算术运算或逻辑运算，都需要两个数据
 - 移位操作指令，通常用于把指定的两个操作数左移或右移一位
 - 转移指令、子程序调用与返回指令，用于解决数据调用次序的需求
 - 特权指令，通常仅用于实现系统软件，这类指令一般不提供给用户
19. 【2009 统考真题】某机器字长为 16 位，主存按字节编址，转移指令采用相对寻址，由 2 字节组成，第一字节为操作码字段，第二字节为相对位移量字段。假定取指令时，每取一字节 PC 自动加 1。若某转移指令所在主存地址为 2000H，相对位移量字段的内容为 06H，则该转移指令成功转移后的目标地址是（ ）。
- 2006H
 - 2007H
 - 2008H
 - 2009H
20. 【2011 统考真题】偏移寻址通过将某个寄存器的内容与一个形式地址相加来生成有效地址。下列寻址方式中，不属于偏移寻址方式的是（ ）。
- 间接寻址
 - 基址寻址
 - 相对寻址
 - 变址寻址
21. 【2011 统考真题】某机器有一个标志寄存器，其中有进位/借位标志 CF、零标志 ZF、符号标志 SF 和溢出标志 OF，条件转移指令 bgt（无符号整数比较大时转移）的转移条件是（ ）。
- $CF + OF = 1$
 - $\overline{SF} + ZF = 1$
 - $\overline{CF+ZF} = 1$
 - $\overline{CF+SF} = 1$
22. 【2013 统考真题】假设变址寄存器 R 的内容为 1000H，指令中的形式地址为 2000H；地址 1000H 中的内容为 2000H，地址 2000H 中的内容为 3000H，地址 3000H 中的内容为 4000H，则变址寻址方式下访问到的操作数是（ ）。
- 1000H
 - 2000H
 - 3000H
 - 4000H
23. 【2014 统考真题】某计算机有 16 个通用寄存器，采用 32 位定长指令字，操作码字段（含寻址方式位）为 8 位，Store 指令的源操作数和目的操作数分别采用寄存器直接寻址和基址寻址方式。若基址寄存器可使用任意一个通用寄存器，且偏移量用补码表示，则 Store 指令中偏移量的取值范围是（ ）。
- 32768～+32767
 - 32767～+32768
 - 65536～+65535
 - 65535～+65536
24. 【2016 统考真题】某指令格式如下所示。
- | | | | |
|----|---|---|---|
| OP | M | I | D |
|----|---|---|---|
- 其中 M 为寻址方式，I 为变址寄存器编号，D 为形式地址。若采用先变址后间址的寻址方式，则操作数的有效地址是（ ）。
- $I + D$
 - $(I) + D$
 - $((I) + D)$
 - $((I)) + D$
25. 【2017 统考真题】下列寻址方式中，最适合按下标顺序访问一维数组元素的是（ ）。
- 相对寻址
 - 寄存器寻址
 - 直接寻址
 - 变址寻址
26. 【2018 统考真题】按字节编址的计算机中，某 double 型数组 A 的首地址为 2000H，使用变址寻址和循环结构访问数组 A，保存数组下标的变址寄存器的初值为 0，每次循环取一个数组元素，其偏移地址为变址值乘以 sizeof (double)，取完后变址寄存器的内容自动加 1。若某次循环所取元素的地址为 2100H，则进入该次循环时变址寄存器的内容是（ ）。
- 25
 - 32
 - 64
 - 100
27. 【2019 统考真题】某计算机采用大端方式，按字节编址。某指令中操作数的机器数为 1234 FF00H，该操作数采用基址寻址方式，形式地址（用补码表示）为 FF12H，基址寄存器的

内容为 F000 0000H，则该操作数的 LSB（最低有效字节）所在的地址是（ ）。

- A. F000 FF12H B. F000 FF15H C. EFFF FF12H D. EFFF FF15H

28. 【2020 统考真题】某计算机采用 16 位定长指令字格式，操作码位数和寻址方式位数固定，指令系统有 48 条指令，支持直接、间接、立即、相对 4 种寻址方式。在单地址指令中，直接寻址方式的可寻址范围是（ ）。

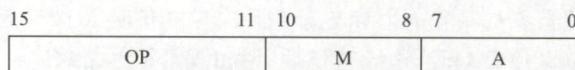
- A. 0 ~ 255 B. 0 ~ 1023 C. -128 ~ 127 D. -512 ~ 511

二、综合应用题

01. 某计算机指令系统采用定长操作码（单字和双字两种）。回答以下问题：

- 1) 采用什么寻址方式时指令码长度最短？采用什么寻址方式时指令码长度最长？
- 2) 采用什么寻址方式时执行速度最快？采用什么寻址方式时执行速度最慢？
- 3) 若指令系统采用定长指令码格式，则采用什么寻址方式时执行速度最快？

02. 某机字长为 16 位，存储器按字编址，访问内存指令格式如下：



其中，OP 为操作码，M 为寻址特征，A 为形式地址。设 PC 和 Rx 分别为程序计数器和变址寄存器，字长为 16 位，问：

- 1) 该指令能定义多少种指令？
- 2) 下表中各种寻址方式的寻址范围为多少？
- 3) 写出下表中各种寻址方式的有效地址 EA 的计算公式。

关注公众号【乘龙考研】
一手更新 稳定有保障

寻址方式	有效地址 EA 的计算公式	寻址范围
直接寻址		
间接寻址		
变址寻址		
相对寻址		

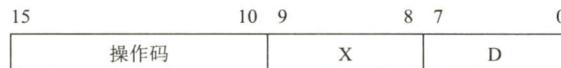
03. 一条双字长的 Load 指令存储在地址为 200 和 201 的存储位置，该指令将指定的内容装入累加器（ACC）中。指令的第一个字指定操作码和寻址方式，第二个字是地址部分。主存内容示意图如右图所示。PC 值为 200，R1 值为 400，XR 值为 100。

指令的寻址方式字段可指定任何一种寻址方式。请在下列寻址方式中，装入 ACC 的值。

- 1) 直接寻址。
- 2) 立即寻址。
- 3) 间接寻址。
- 4) 相对寻址。
- 5) 变址寻址。
- 6) 寄存器 R1 寻址。
- 7) 寄存器 R1 间接寻址。

04. 某机的机器字长为 16 位，主存按字编址，指令格式如下：

地址	主存	
	LOAD	MOD
200		
201	500	
202		
300	450	
400	700	
500	800	
600	900	
702	325	
800	300	



其中，D 为位移量；X 为寻址特征位。

X = 00：直接寻址。

X = 01：用变址寄存器 X1 进行变址。

X = 10：用变址寄存器 X2 进行变址。

X = 11：相对寻址。

设 $(PC) = 1234H$, $(X1) = 0037H$, $(X2) = 1122H$ (H 代表十六位进制数), 请确定下列指令的有效地址:

- ① 4420H ② 2244H ③ 1322H ④ 3521H ⑤ 6723H

05. 某计算机字长 16 位, 标志寄存器 FLAGS 中的 ZF、SF 和 OF 分别是零标志、符号标志和溢出标志, 采用双字节字长指令字。假定 bgt (大于零转移) 指令的第一个字节指明操作码和寻址方式, 第二个字节为偏移地址 Imm8, 用补码表示。指令功能是:

若 $(ZF + (SF \oplus OF)) = 0$, 则 $PC = PC + 2 + Imm8 \times 2$; 否则, $PC = PC + 2$ 。

请回答下列问题:

- 1) 该计算机的编址单位是多少?
- 2) bgt 指令执行的是带符号整数比较, 还是无符号整数比较?
- 3) 偏移地址 Imm8 的含义是什么? 转移目标地址的范围是什么?

06. 【2010 统考真题】某计算机字长为 16 位, 主存地址空间大小为 128KB, 按字编址, 采用单字长指令格式, 指令各字段定义如下:



转移指令采用相对寻址方式, 相对偏移量用补码表示, 寻址方式定义见下表。

Ms/Md	寻址方式	助记符	含义
000B	寄存器直接	Rn	操作数 = (Rn)
001B	寄存器间接	(Rn)	操作数 = ((Rn))
010B	寄存器间接、自增	(Rn) +	操作数 = ((Rn)), (Rn) + 1 → Rn
011B	相对	D(Rn)	转移目标地址 = (PC) + (Rn)

注: (X) 表示存储器地址 X 或寄存器 X 的内容。

回答下列问题:

- 1) 该指令系统最多可有多少条指令? 该计算机最多有多少个通用寄存器? 存储器地址寄存器 (MAR) 和存储器数据寄存器 (MDR) 至少各需要多少位?
- 2) 转移指令的目标地址范围是多少?
- 3) 若操作码 0010B 表示加法操作 (助记符为 add), 寄存器 R4 和 R5 的编号分别为 100B 和 101B, R4 的内容为 1234H, R5 的内容为 5678H, 地址 1234H 中的内容为 5678H, 5678H 中的内容为 1234H, 则汇编语句 “add (R4), (R5)+” (逗号前为源操作数, 逗号后为目的操作数) 对应的机器码是什么 (用十六进制表示)? 该指令执行后, 哪些寄存器和存储单元的内容会改变? 改变后的内容是什么?

07. 一条双字长的取数指令 (LDA) 存于存储器的 200 和 201 单元, 其中第一个字为操作码

关注公众号【乘龙考研】
一手更新稳定有保障

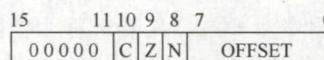
OP 和寻址特征 M，第二个字为形式地址 A。假设 PC 的当前值为 200，变址寄存器 IX 的内容为 100，基址寄存器的内容为 200，存储器相关单元的内容如下表所示：

地址	201	300	400	401	500	501	502	700
内容	300	400	700	501	600	700	900	401

下表的各列分别为寻址方式、该寻址方式下的有效地址及取数指令执行结束后累加器 (AC) 的内容, 试补全下表:

寻址方式	有效地址 (EA)	累加器 (AC) 的内容
立即寻址		
直接寻址		
间接寻址		
相对寻址		
变址寻址		
基址寻址		
先变址后间址		
先间址后变址		

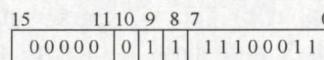
08. 【2013 统考真题】某计算机采用 16 位定长指令字格式，其 CPU 中有一个标志寄存器，其中包含进位/借位标志 CF、零标志 ZF 和符号标志 NF。假定为该机设计了条件转移指令，其格式如下：



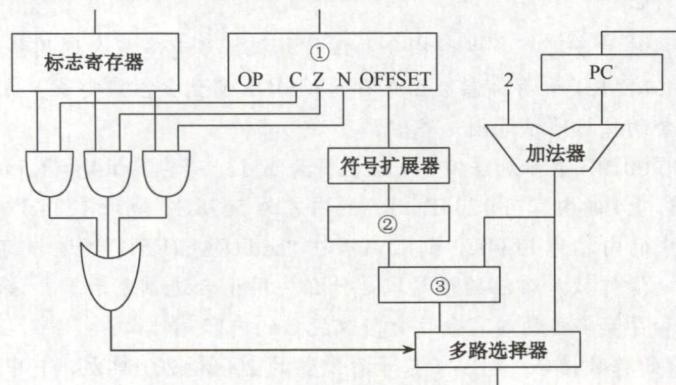
其中，00000 为操作码 OP；C、Z 和 N 分别为 CF、ZF 和 NF 的对应检测位，某检测位为 1 时表示需检测对应标志，需检测的标志位中只要有一个为 1 就转移，否则不转移。例如，若 C = 1，Z = 0，N = 1，则需检测 CF 和 NF 的值，当 CF = 1 或 NF = 1 时发生转移；OFFSET 是相对偏移量，用补码表示。转移执行时，转移目标地址为 $(PC) + 2 + 2 \times OFFSET$ ；顺序执行时，下一条指令地址为 $(PC) + 2$ 。请回答下列问题：

- 1) 该计算机存储器是按字节编址还是按字编址？该条件转移指令向后（反向）最多可跳转多少条指令？

2) 某条件转移指令的地址为 200CH，指令内容如下图所示，若该指令执行时 CF = 0，ZF = 0，NF = 1，则该指令执行后 PC 的值是多少？若该指令执行时 CF = 1，ZF = 0，NF = 0，则该指令执行后 PC 的值又是多少？请给出计算过程。



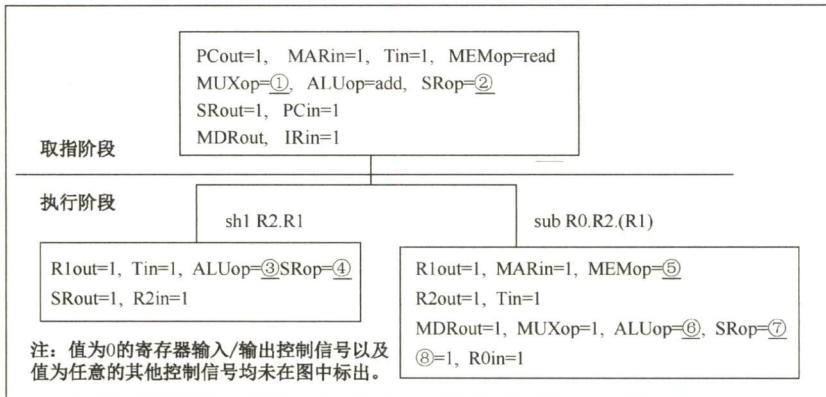
- 3) 实现“无符号数比较小于或等于时转移”功能的指令中，C、Z 和 N 应各是什么？
 4) 以下是该指令对应的数据通路示意图，要求给出图中部件①~③的名称或功能说明。



有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

关注公众号【乘龙考研】
一手更新稳定有保障

09. 【2015 统考真题】题（2015 年真题第 43 题，5.3 节综合题 8）中描述的计算机，某部分指令执行过程的控制信号如下所示。



该机指令格式如下图所示，支持寄存器直接和寄存器间接两种寻址方式，寻址方式位分别为 0 和 1，通用寄存器 R0~R3 的编号分别为 0, 1, 2 和 3。



回答下列问题：

- 1) 该机的指令系统最多可定义多少条指令？
- 2) 假定 inc、shl 和 sub 指令的操作码分别为 01H、02H 和 03H，则以下指令对应的机器代码各是什么？
 - ① inc R1 ; (R1) + 1 → R1
 - ② shl R2, R1 ; (R1) << 1 → R2
 - ③ sub R3, (R1), R2 ; ((R1)) - (R2) → R3
- 3) 假设寄存器 X 的输入和输出控制信号分别为 Xin 和 Xout，其值为 1 表示有效，为 0 表示无效（如 PCout = 1 表示 PC 内容送总线）；存储器控制信号为 MEMop，用于控制存储器的读（read）和写（write）操作。写出本题第一幅图中标号①~⑧处的控制信号或控制信号的取值。
- 4) 指令“sub R1, R3, (R2)” 和 “inc R1”的执行阶段至少各需要多少个时钟周期？

10. 【2021 统考真题】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，指令格式及说明如下：

格式	6 位	2 位	2 位	2 位	4 位	指令功能或指令类型说明
R型	000000	rs	rt	rd	op1	$R[rd] \leftarrow R[rs] op1 R[rt]$
I型	op2	rs	rt	imm		含 ALU 运算、条件转移和访存操作 3 类指令
J型	op3		target			PC 的低 10 位 \leftarrow target

其中, op1 ~ 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) ALU 的宽度是多少位? 可寻址主存空间大小为多少字节? 指令寄存器、主存地址寄存器 (MAR) 和主存数据寄存器 (MDR) 分别应有多少位?
- 2) 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?
- 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 (参考上述指令功能说明的格式进行描述)? 若 1, 2, 3 号通用寄存器当前内容分别为 B052H, 0008H, 0020H, 则分别执行指令 01B2H 和 01B3H 后, 3 号通用寄存器内容各是什么? 各自结果是否溢出?
- 4) 若采用 I 型格式的访存指令中 imm (偏移量) 为带符号整数, 则地址计算时应对 imm 进行零扩展还是符号扩展?
- 5) 无条件转移指令可以采用上述哪种指令格式?

4.2.4 答案与解析

一、单项选择题

关注公众号【乘龙考研】
一手更新 稳定有保障

01. B

采用不同寻址方式的目的是为了缩短指令字长, 扩大寻址空间, 提高编程的灵活性, 但也提高了指令译码的复杂度。程序控制是靠转移指令而非寻址方式实现的。

02. A

无条件转移指令是指程序转移到新的地址后继续执行, 因此必须给出下一条指令的执行地址, 并送入程序计数器 (PC)。

03. D

CPU 中寄存器的数量都不会太多, 用很短的编码就可以指定寄存器, 寄存器寻址需要的地址段位数为 \log_2 (通用寄存器个数), 因此能有效地缩短地址段的位数。立即寻址, 操作数直接保存在指令中, 若地址段位数太小, 则操作数表示的范围会很小; 变址寻址, EA = 变址寄存器 IX 的内容 + 形式地址 A, A 与主存寻址空间有关; 间接寻址中存放的仍然是主存地址。

04. B

隐地址不给出明显的操作数地址, 而在指令中隐含操作数的地址, 因此可以简化地址结构, 如零地址指令。

05. B

立即寻址最快, 指令直接给出操作数; 寄存器寻址次之, 只需访问一次寄存器; 直接寻址再次之, 访问一次内存; 间接寻址最慢, 要访问内存两次或以上。

注意: 寄存器间接寻址取操作数的速度接近直接寻址。

06. A

指令字中的形式地址为操作数的有效地址, 这种方式为直接寻址。

07. B

变址寻址的有效地址是 $(X) + A$, 再进行间址, 即把 $(X) + A$ 中取出的内容作为真实地址 EA, 即 $EA = ((X) + A)$ 。

寄存器中的内容和指令地址码相加得到的是操作数的地址码。

08. B

变址寻址便于处理数组问题。基址寻址与变址寻址的区别见下表。

	基址寻址	变址寻址
有效地址	$EA = (BR) + A$	$EA = (IX) + A$
访存次数	1	1
寄存器内容	由操作系统或管理程序确定	由用户设定
程序执行过程中值可变否	不可变	可变
特点	有利于多道程序设计和编制浮动程序	有利于处理数组问题和编制循环程序

09. B

进、出堆栈时对栈顶指针的操作顺序是不同的，进栈时是先压入数据($A \rightarrow M_{SP}$)，后修改指针($SP - 1 \rightarrow SP$)，说明栈指针是指向栈顶的空单元的，所以出栈时要先修改指针($SP + 1 \rightarrow SP$)，然后才能弹出数据($M_{SP} \rightarrow A$)。

10. D

相对寻址中，有效地址 $EA = (PC) + A$ (A 为形式地址)，执行本条指令时， PC 已完成加 1 操作， PC 中保存的是下一条指令的地址，因此以下一条指令的地址为基准位置的偏移量。

11. A

相对寻址编制程序时，无须指定绝对地址，只需确定程序内部的相对距离，从而可以使用浮动地址，给程序的重定位带来了方便。便于实现多道程序，因此选择选项 A。

12. C

跳跃寻址通过转移类指令（如相对寻址）来实现，可用来实现程序的条件或无条件转移。

13. C

指令字长为 16 位，2 字节，因此取指令后 PC 的内容为 $(PC) + 2 = 2002H$ ；无条件转移指令将下一条指令的地址送至 PC ，形式地址为 $40H$ ，指令执行后 $PC = 2002H + 0040H = 2042H$ 。

14. B

机器按字寻址，程序计数器（ PC ）给出下一条指令字的访存地址（指令在内存中的地址），因此取决于存储器的字数；指令寄存器（ IR ）用于接收取得的指令，因此取决于指令字长。

15. D

直接寻址 200 访问的操作数是 300，A 错误。寄存器间接寻址（ R ）的访问结果与 I 一样，B 错误。存储器间接寻址（200）表示主存地址 200 中的内容为有效地址，有效地址为 300，访问的操作数是 400，C 错误。寄存器寻址 R 表示寄存器 R 的内容为操作数，只有 D 正确。

16. A

寄存器间接寻址中操作数的有效地址 $EA = (R_i)$ ，8 号寄存器内容为 $1200H$ ，因此 $EA = 1200H$ 。

17. D, C

首先需要讲解一下补码扩充的问题。补码的扩充只需使用符号位补足即可，也就是说正数补码的扩充只要补 0，负数补码的扩充只需补 1（这是由补码的性质决定的）。理解了该性质，这道题就变成了十进制转换为十六进制的简单问题。

- 1) PC 的当前值为 240，该指令取出后 PC 的值为 243，要求转移到 290，即相对位移量为 $290 - 243 = 47$ ，转换成补码为 $2FH$ 。由于数据在存储器中采用以低字节地址为字地址的存放方式，因此该转移指令的第二字节为 $2FH$ ，由于 47 是正数，因此只需在高位补 0，所以第三字节为 $00H$ 。

2) PC 的当前值为 240, 该指令取出后 PC 的值为 243, 要求转移到 200, 即相对位移量为 $200 - 243 = -43$, 转换成补码为 D5H。由于数据在存储器中采用以低字节地址为字地址的存放方式, 因此该转移指令的第二字节为 D5H, 由于-43 是负数, 因此只需在高位补 1, 所以第三字节为 FFH。

18. D

算术与逻辑运算指令用于完成对一个(如自增、取反等)或两个数据的算术运算或逻辑运算, 因此 A 错误。移位操作用于把一个操作数左移或右移一位或多位, 因此 B 错误。转移指令、子程序调用与返回指令用于解决变动程序中指令执行次序的需求, 而不是数据调用次序的需求, 因此 C 错误。

19. C

相对寻址 $EA = (PC) + A$, 先计算取指后的 PC 值。转移指令由 2 字节组成, 每取一字节 PC 加 1, 在取指后 PC 值为 2002H, 因此 $EA = (PC) + A = 2002H + 06H = 2008H$ 。本题易误选 A 或 B, 选项 A 未考虑 PC 值的自动更新, 选项 B 虽然考虑了 PC 值的自动更新, 但未注意到该转移指令是一条 2 字节指令, PC 值应是“+2”而不是“+1”。

20. A

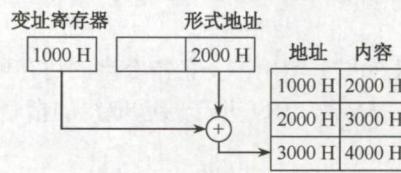
间接寻址不需要寄存器, $EA = (A)$ 。基址寻址: $EA = A + (BR)$; 相对寻址: $EA = A + (PC)$; 变址寻址: $EA = A + (IX)$ (BR 表示基址寄存器, PC 表示程序计数器, IX 表示变址寄存器)。

21. C

假设两个无符号整数 A 和 B , bgt 指令会将 A 和 B 进行比较, 也就是将 A 和 B 相减。若 $A > B$, 则 $A - B$ 肯定无进位/借位, 也不为 0(为 0 时表示两数相等), 因此 CF 和 ZF 均为 0, 选 C。其余选项中用到了符号标志 SF 和溢出标志 OF, 显然应当排除。

22. D

根据变址寻址的方法, 变址寄存器的内容(1000H)与形式地址的内容(2000H)相加, 得到操作数的实际地址(3000H), 根据实际地址访问内存, 获取操作数 4000H, 如下图所示。



关注公众号【乘龙考研】
一手更新 稳定有保障

23. A

采用 32 位定长指令字, 其中操作码为 8 位, 两个地址码共用 $32 - 8 = 24$ 位, 而 Store 指令的源操作数和目的操作数分别采用寄存器直接寻址和基址寻址, 机器中共有 16 个通用寄存器, 因此寻址一个寄存器需要 $\log_2 16 = 4$ 位, 源操作数中的寄存器直接寻址用掉 4 位, 而目的操作数采用基址寻址也要指定一个寄存器, 同样用掉 4 位, 则留给偏移量的位数为 $24 - 4 - 4 = 16$ 位, 而偏移量用补码表示, 因此 16 位补码的表示范围为-32768~+32767。

24. C

变址寻址中, 有效地址(EA)等于指令字中的形式地址 D 与变址寄存器 I 的内容之和, 即 $EA = (I) + D$ 。间接寻址是相对于直接寻址而言的, 指令的地址字段给出的形式地址不是操作数的真正地址, 而是操作数地址的地址, 即 $EA = (D)$ 。从而该操作数的有效地址是 $((I) + D)$ 。

25. D

在变址操作时, 将计算机指令中的地址与变址寄存器中的地址相加, 得到有效地址, 指令提供数组首地址, 由变址寄存器来定位数据中的各元素。所以它最适合按下标顺序访问一维数组元

素，选 D。相对寻址以 PC 为基地址，以指令中的地址为偏移量确定有效地址。寄存器寻址则在指令中指出需要使用的寄存器。直接寻址在指令的地址字段直接指出操作数的有效地址。

26. B

根据变址寻址的公式 $EA = (IX) + A$ ，有 $(IX) = 2100H - 2000H = 100H = 256$, $\text{sizeof(double)} = 8$ (双精度浮点数用 8 位字节表示)，因此数组的下标为 $256/8 = 32$ ，答案选 B。

27. D

注意，内存地址是无符号数。

操作数采用基址寻址方式， $EA = (BR) + A$ ，基址寄存器 BR 的内容为 F000 0000H，形式地址用补码表示为 FF12H 即 1111 1111 0001 0010B，因此有效地址为 F000 0000H + (-00EEH) = EFFF FF12H。计算机采用大端方式编址，所以低位字节存放在字的高地址处，机器数一共占 4 字节，该操作数的 LSB 所在的地址是 EFFF FF12H + 3 = EFFF FF15H，所以选 D。

28. A

48 条指令需要 6 位操作码字段 ($2^5 < 48 < 2^6$)，4 种寻址方式需要 2 位寻址特征位 ($4 = 2^2$)，还剩 $16 - 6 - 2 = 8$ 位作为地址码，故直接寻址范围为 0~255。注意，主存地址不能为负。

二、综合应用题

01. 【解答】

- 1) 由于通用寄存器的数量有限，可以用较少的二进制位来编码，所以采用寄存器寻址方式和寄存器间接寻址方式的指令码长度最短。因为需要在指令中表示数据和地址，所以立即寻址方式、直接寻址方式和间接寻址方式的指令码长度最长。若指令码长度太短，则无法表示范围较大的立即数和寻址到较大的内存地址空间。
- 2) 由于通用寄存器位于 CPU 内部，无须到内存读取操作数，所以寄存器寻址方式执行速度最快。立即寻址虽然无须取操作数，但因指令码长度最长，取指令访存花费的时间较多。而间接寻址方式需要读内存两次，第一次由操作数的间接地址读到操作数的地址，第二次再由操作数的地址读到操作数，所以间接寻址方式的执行速度最慢。
- 3) 若指令系统采用定长指令码格式，所有指令（包括采用立即寻址方式的指令）所包含的二进制位数均相同，则立即寻址方式执行速度最快，因为读到指令的同时，便立即取得操作数。若采用变长指令码格式，由于要表示一定范围内的立即数，包含立即数的指令通常需要较多的二进制位，取指令时，可能需要不止一次地读内存来完成取指令。因此，采用变长指令码格式时，寄存器寻址方式执行速度最快。

02. 【解答】

- 1) 因为 OP 字段长为 5 位，所以指令能定义 $2^5 = 32$ 种指令。
- 2)、3) 各种寻址方式的有效地址 EA 的计算公式、寻址范围见下表。

寻址方式	有效地址 EA 的计算公式	寻址范围
直接寻址	$EA = A$	$2^8 = 256$
间接寻址	$EA = (A)$	$2^{16} = 64K$
变址寻址	$EA = (Rx) + A$	$2^{16} = 64K$
相对寻址	$EA = (PC) + A$	$2^8 = 256$ (PC 附近 256)

03. 【解答】

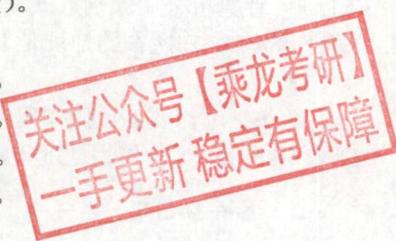
- 1) 直接寻址时，有效地址是指令中的地址码部分 500，装入 ACC 的是 800。
- 2) 立即寻址时，指令的地址码部分是操作数而不是地址，所以将 500 装入 ACC。

- 3) 间接寻址时, 操作数的有效地址存储在地址为 500 的单元中, 由此得到有效地址为 800, 操作数是 300。
- 4) 相对寻址时, 有效地址 $EA = (PC) + A = 202 + 500 = 702$, 所以装入 ACC 的操作数是 325。这是因为指令是双字长, 在该指令的执行阶段, PC 的内容已经加 2, 更新为下一条指令的地址 202。
- 5) 变址寻址时, 有效地址 $EA = (XR) + A = 100 + 500 = 600$, 所以装入 ACC 的操作数是 900。
- 6) 寄存器寻址时, R1 的内容 400 装入 ACC。
- 7) 寄存器间接寻址时, 有效地址是 R1 的内容 400, 装入 ACC 的操作数是 700。

04. 【解答】

取指后, $PC = 1235H$ (注意, 不是 1236H, 因主存按字编址)。

- ① $X = 00, D = 20H$, 有效地址 $EA = 20H$ 。
- ② $X = 10, D = 44H$, 有效地址 $EA = 1122H + 44H = 1166H$ 。
- ③ $X = 11, D = 22H$, 有效地址 $EA = 1235H + 22H = 1257H$ 。
- ④ $X = 01, D = 21H$, 有效地址 $EA = 0037H + 21H = 0058H$ 。
- ⑤ $X = 11, D = 23H$, 有效地址 $EA = 1235H + 23H = 1258H$ 。



05. 【解答】

- 1) 因为 PC 的增量是 2, 且每条指令占 2 个字节, 所以编址单位是字节。
- 2) 根据“大于”条件判断表达式, 可以看出该 bgt 指令实现的是带符号整数比较。因为无符号数比较时, 其判断表达式中没有溢出标志 OF。
- 3) 偏移地址 Imm8 为补码表示, 说明转移目标地址可能在 bgt 指令之后。计算转移目标地址时, 偏移量为 $Imm8 \times 2$, 说明 Imm8 不是相对地址, 而是相对指令数。Imm8 的范围为 -128 ~ 127, 故转移目标地址的范围是 $PC + 2 + (-128 \times 2) \sim PC + 2 + 127 \times 2$, 也即转移目标地址的范围是相对于 bgt 指令的前 127 条指令到后 128 条指令之间。

06. 【解答】

- 1) 操作码占 4 位, 则该指令系统最多可有 $2^4 = 16$ 条指令。操作数占 6 位, 其中寻址方式占 3 位、寄存器编号占 3 位, 因此该机最多有 $2^3 = 8$ 个通用寄存器。主存地址空间大小为 128KB, 按字编址, 字长为 16 位, 共有 $128KB / 2B = 2^{16}$ 个存储单元, 因此 MAR 至少为 16 位; 因为字长为 16 位, 因此 MDR 至少为 16 位。
- 2) 寄存器字长为 16 位, PC 可表示的地址范围为 $0 \sim 2^{16} - 1$, Rn 可表示的相对偏移量为 $-2^{15} \sim 2^{15} - 1$, 而主存地址空间为 2^{16} , 因此转移指令的目标地址范围为 $0000H \sim FFFFH$ ($0 \sim 2^{16} - 1$)。
- 3) 汇编语句 “add (R4), (R5)+” 对应的机器码为

字段	OP	Ms	Rs	Md	Rd
内容	0010	001	100	010	101
说明	add	寄存器间接	R4	寄存器间接、自增	R5

将对应的机器码写成十六进制形式为 $0010\ 0011\ 0001\ 0101B = 2315H$ 。

该指令的功能是将 R4 的内容所指的存储单元的数据与 R5 的内容所指的存储单元的数据相加, 并将结果送入 R5 的内容所指的存储单元中。 $(R4) = 1234H, (1234H) = 5678H; (R5) = 5678H, (5678H) = 1234H$; 执行加法操作 $5678H + 1234H = 68ACH$ 。之后 R5 自增。

该指令执行后, R5 和存储单元 5678H 的内容会改变, R5 的内容从 5678H 变为 5679H, 存储单元 5678H 中的内容变为该指令的计算结果 68ACH。

07. 【解答】

直接寻址：寄存器的内容是有效地址 EA，所以直接寻址的有效地址为 300，根据题给出的表格可知，地址 300 对应的内容为 400。

间接寻址：根据寄存器的内容寻找到的内容才是真正的有效地址，所以根据寄存器内容 300 找到的 400 才是间接寻址的有效地址，因此有效地址为 400，地址 400 对应的内容为 700。

相对寻址：寄存器的内容加上 PC 的内容为有效地址，PC 的当前值为 200，所以当取出一条指令后，变为 202，因此有效地址为 $202 + 300 = 502$ ，地址 502 对应的内容为 900。

变址寻址：变址寻址的有效地址为变址寄存器的内容加上累加器的内容，所以有效地址为 $100 + 300 = 400$ ，地址 400 对应的内容为 700。

基址寻址：基址寻址的有效地址为基址寄存器的内容加上累加器的内容，所以有效地址为 $200 + 300 = 500$ ，地址 500 对应的内容为 600。

先变址后间址：先变址，即先让变址寄存器的内容加上累加器的内容，即 400；再间址，意思就是根据地址 400 找到的内容才是有效地址，所以先变址后间址的有效地址为 700。地址 700 对应的内容为 401。

先间址后变址：先间址，即先根据累加器的内容 300 找到间址的有效地址 400；再变址，即 400 再加上变址寄存器的内容，也就是 $400 + 100 = 500$ ，地址 500 对应的内容为 600。

综上，得到下表：

寻址方式	有效地址 (EA)	累加器 (AC) 的内容
立即寻址	—	300
直接寻址	300	400
间接寻址	400	700
相对寻址	502	900
变址寻址	400	700
基址寻址	500	600
先变址后间址	700	401
先间址后变址	500	600

08. 【解答】

1) 因为指令长度为 16 位，且下一条指令地址为 $(PC) + 2$ ，因此编址单位是字节。

偏移量 OFFSET 为 8 位补码，范围为 $-128 \sim 127$ ，因此相对于当前条件转移指令，向后最多可跳转 127 条指令。

2) 指令中 $C = 0, Z = 1, N = 1$ ，因此应根据 ZF 和 NF 的值来判断是否转移。 $CF = 0, ZF = 0, NF = 1$ 时，需转移。已知指令中的偏移量为 $1110\ 0011B = E3H$ ，符号扩展后为 $FFE3H$ ，左移一位（乘 2）后为 $FFC6H$ ，因此 PC 的值（即转移目标地址）为 $200CH + 2 + FFC6H = 1FD4H$ 。 $CF = 1, ZF = 0, NF = 0$ 时不转移。PC 的值为 $200CH + 2 = 200EH$ 。

3) 指令中的 C、Z 和 N 应分别设置为 $C = Z = 1, N = 0$ 。

4) 部件①用于存放当前指令，不难得出为指令寄存器；多路选择器根据符号标志 C/Z/N 来决定下一条指令的地址是 $PC + 2$ 还是 $PC + 2 + 2 \times OFFSET$ ，因此多路选择器左边线上的结果应是 $PC + 2 + 2 \times OFFSET$ 。根据运算的先后顺序及与 $PC + 2$ 的连接，部件②用于左移一位实现乘 2，为移位寄存器。部件③用于 $PC + 2$ 和 $2 \times OFFSET$ 相加，为加法器。

部件②：移位寄存器（用于左移一位）；部件③：加法器（地址相加）。

09. 【解答】

1) 指令操作码有 7 位，因此最多可定义 $2^7 = 128$ 条指令。

2) 各条指令的机器代码如下:

- ① “inc R1”的机器码为 0000001 0 01 0 00 0 00, 即 0240H。
- ② “shl R2, R1”的机器码为 0000010 0 10 0 01 0 00, 即 0488H。
- ③ “sub R3, (R1), R2”的机器码为 0000011 0 11 1 01 0 10, 即 06EAH。

3) 各标号处的控制信号或控制信号取值如下:

- ①0; ②mov; ③mova; ④left; ⑤read; ⑥sub; ⑦mov; ⑧SRoute。

4) 指令“sub R1, R3, (R2)”的执行阶段至少包含 4 个时钟周期; 指令“inc R1”的执行阶段至少包含 2 个时钟周期。

10. 【解答】

1) ALU 的宽度为 16 位, ALU 的宽度即 ALU 运算对象的宽度, 通常与字长相同。地址线为 20 位, 按字节编址, 可寻址主存空间大小为 2^{20} 字节(或 1MB)。指令寄存器有 16 位, 和单条指令长度相同。MAR 有 20 位, 和地址线位数相同。MDR 有 8 位, 和数据线宽度相同。

2) R 型格式的操作码有 4 位, 最多有 2^4 (或 16) 种操作。I 型和 J 型格式的操作码有 6 位, 因为它们的操作码部分重叠, 所以共享这 6 位的操作码空间, 且前 6 位全 0 的编码已被 R 型格式占用, 因此 I 和 J 型格式最多有 $2^6 - 1 = 63$ 种操作。从 R 型和 I 型格式的寄存器编号部分可知, 只用 2 位对寄存器编码, 因此通用寄存器最多有 4 个。

3) 指令 01B2H = 000000 01 10 11 0010B 为一条 R 型指令, 操作码 0010 表示带符号整数减法指令, 其功能为 $R[3] \leftarrow R[1] - R[2]$ 。执行指令 01B2H 后, $R[3] = B052H - 0008H = B04AH$, 结果未溢出。指令 01B3H = 000000 01 10 11 0011B, 操作码 0011 表示带符号整数乘法指令, 执行指令 01B3H 后, $R[3] = R[1] \times R[2] = B052H \times 0008H = 8290H$, 结果溢出。

4) 在进行指令的跳转时, 可能向前跳转, 也可能向后跳转, 偏移量是一个带符号整数, 因此在地址计算时, 应对 imm 进行符号扩展。

5) 无条件转移指令可以采用 J 型格式, 将 target 部分写入 PC 的低 10 位, 完成跳转。

4.3 程序的机器级代码表示

本节的内容是 2022 年大纲新增考点, 其实在 2017 年和 2019 年就以综合题的形式考查过, 在当时看来属于超纲范畴, 很多跨考生当时对此几乎无从下手, 相信通过本节的学习后, 应能从容应对。

4.3.1 常用汇编指令介绍

1. 相关寄存器

x86 处理器中有 8 个 32 位的通用寄存器, 各寄存器及说明如图 4.11 所示。为了向后兼容, EAX、EBX、ECX 和 EDX 的高两位字节和低两位字节可以独立使用, E 为 Extended, 表示 32 位的寄存器。例如, EAX 的低两位字节称为 AX, 而 AX 的高低字节又可分别作为两个 8 位寄存器, 分别称为 AH 和 AL。

除 EBP 和 ESP 外, 其他几个寄存器的用途是比较任意的。

2. 汇编指令格式

使用不同的编程工具开发程序时, 用到的汇编程序也不同, 一般有两种不同的汇编格式:

关注公众号【乘龙考研】
一手更新 稳定有保障

AT&T 格式和 Intel 格式。它们的区别主要体现如下：

通用寄存器				16bit	32bit	说明
31	16 15	8 7	0	AX EAX	累加器 (Accumulator)	
		AH	AL	BX EBX	基址寄存器 (Base Register)	
		BH	BL	CX ECX	计数寄存器 (Count Register)	
		CH	CL	DX EDX	数据寄存器 (Data Register)	
		DH	DL	ESI	变址寄存器 (Index Register)	
	ESI			EDI	堆栈基指针 (Base Pointer)	
	EDI			EBP	堆栈顶指针 (Stack Pointer)	
	EBP			ESP	堆栈顶指针 (Stack Pointer)	
	ESP					

图 4.11 x86 处理器中的主要寄存器及说明

- ① AT&T 格式的指令只能用小写字母，而 Intel 格式的指令对大小写不敏感。
- ② 在 AT&T 格式中，第一个为源操作数，第二个为目的操作数，方向从左到右，合乎自然；在 Intel 格式中，第一个为目的操作数，第二个为源操作数，方向从右向左。
- ③ 在 AT&T 格式中，寄存器需要加前缀 “%”，立即数需要加前缀 “\$”；在 Intel 格式中，寄存器和立即数都不需要加前缀。
- ④ 在内存寻址方面，AT&T 格式使用 “(” 和 “)”，而 Intel 格式使用 “[” 和 “]”。
- ⑤ 在处理复杂寻址方式时，例如 AT&T 格式的内存操作数 “disp(base, index, scale)” 分别表示偏移量、基址寄存器、变址寄存器和比例因子，如 “8(%edx, %eax, 2)” 表示操作数为 $M[R[edx] + R[eax]*2 + 8]$ ，其对应的 Intel 格式操作数为 “[edx + eax*2 + 8]”。
- ⑥ 在指定数据长度方面，AT&T 格式指令操作码的后面紧跟一个字符，表明操作数大小，“b” 表示 byte（字节）、“w” 表示 word（字）或 “l” 表示 long（双字）。Intel 格式也有类似的语法，它在操作码后面显式地注明 byte ptr、word ptr 或 dword ptr。

注意：由于 32 或 64 位体系结构都是由 16 位扩展而来的，因此用 word（字）表示 16 位。

表 4.2 展示了两种格式的几条不同指令。其中，mov 指令用于在内存和寄存器之间或者寄存器之间移动数据；lea 指令用于将一个内存地址（而不是其所指的内容）加载到目的寄存器。

表 4.2 AT&T 格式指令和 Intel 格式指令的对比

AT&T 格式	Intel 格式	含义
mov \$100, %eax	mov eax, 100	$100 \rightarrow R[eax]$
mov %eax, %ebx	mov ebx, eax	$R[eax] \rightarrow R[ebx]$
mov %eax, (%ebx)	mov [ebx], eax	$R[eax] \rightarrow M[R[ebx]]$
mov %eax, -8(%ebp)	mov [ebp-8], eax	$R[eax] \rightarrow M[R[ebp]-8]$
lea 8(%edx, %eax, 2), %eax	lea eax, [edx+eax*2+8]	$R[edx]+R[eax]*2+8 \rightarrow R[eax]$
movl %eax, %ebx	mov dword ptr ebx, eax	长度为 4 字节的 $R[eax] \rightarrow R[ebx]$

注：R[r] 表示寄存器 r 的内容，M[addr] 表示主存单元 addr 的内容， \rightarrow 表示信息传送方向。

两种汇编指令的相互转换并不复杂。考虑到本书参考教材之一 [袁春风所著《计算机系统基础（第二版）》] 使用的是 AT&T 格式，但 2017 年和 2019 年统考综合题使用的是 Intel 格式，因此本书将混用两种格式，两种格式的汇编指令都需要理解。在本节介绍常用指令时，使用 Intel 格式；在后面介绍具体结构的机器级表示时，使用 AT&T 格式。读者在学习时可以尝试转换。

3. 常用指令

汇编指令通常可以分为数据传送指令、逻辑计算指令和控制流指令，下面以 Intel 格式为例，介绍一些重要的指令。以下用于操作数的标记分别表示寄存器、内存和常数。

- <reg>: 表示任意寄存器，若其后带有数字，则指定其位数，如<reg32>表示 32 位寄存器 (eax、ebx、ecx、edx、esi、edi、esp 或 ebp)；<reg16>表示 16 位寄存器 (ax、bx、cx 或 dx)；<reg8> 表示 8 位寄存器 (ah、al、bh、bl、ch、cl、dh、dl)。
- <mem>: 表示内存地址 (如[eax]、[var + 4]或 dword ptr [eax + ebx])。
- <con>: 表示 8 位、16 位或 32 位常数。<con8>表示 8 位常数；<con16>表示 16 位常数；<con32>表示 32 位常数。

x86 中的指令机器码长度为 1 字节，对同一指令的不同用途有多种编码方式，比如 mov 指令就有 28 种机内编码，用于不同操作数类型或用于特定寄存器，例如，

mov ax, <con16>	#机器码为 B8H
mov al, <con8>	#机器码为 B0H
mov <reg16>, <reg16>/<mem16>	#机器码为 89H
mov <reg8>/<mem8>, <reg8>	#机器码为 8AH
mov <reg16>/<mem16>, <reg16>	#机器码为 8BH

(1) 数据传送指令

- 1) **mov 指令**。将第二个操作数（寄存器的内容、内存中的内容或常数值）复制到第一个操作数（寄存器或内存）。但不能用于直接从内存复制到内存。

其语法如下：

```
mov <reg>,<reg>
mov <reg>,<mem>
mov <mem>,<reg>
mov <reg>,<con>
mov <mem>,<con>
```

关注公众号【乘龙考研】
一手更新 稳定有保障

举例：

```
mov eax, ebx          #将 ebx 值复制到 eax
mov byte ptr [var], 5 #将 5 保存到 var 值指示的内存地址的一字节中
```

- 2) **push 指令**。将操作数压入内存的栈，常用于函数调用。ESP 是栈顶，压栈前先将 ESP 值减 4（栈增长方向与内存地址增长方向相反），然后将操作数压入 ESP 指示的地址。

其语法如下：

```
push <reg32>
push <mem>
push <con32>
```

举例（注意，栈中元素固定为 32 位）：

```
push eax            #将 eax 值压栈
push [var]          #将 var 值指示的内存地址的 4 字节值压栈
```

- 3) **pop 指令**。与 push 指令相反，pop 指令执行的是出栈工作，出栈前先将 ESP 指示的地址中的内容出栈，然后将 ESP 值加 4。

其语法如下：

```
pop edi           #弹出栈顶元素送到 edi
pop [ebx]         #弹出栈顶元素送到 ebx 值指示的内存地址的 4 字节中
```

(2) 算术和逻辑运算指令

- 1) **add/sub 指令**。add 指令将两个操作数相加，相加的结果保存到第一个操作数中。sub 指令用于两个操作数相减，相减的结果保存到第一个操作数中。

它们的语法如下：

```
add <reg>,<reg> / sub <reg>,<reg>
add <reg>,<mem> / sub <reg>,<mem>
add <mem>,<reg> / sub <mem>,<reg>
add <reg>,<con> / sub <reg>,<con>
add <mem>,<con> / sub <mem>,<con>
```

关注公众号【乘龙考研】
一手更新 稳定有保障

举例：

sub eax, 10	#eax ← eax - 10
add byte ptr [var], 10	#10 与 var 值指示的内存地址的一字节值相加，并将结果 保存在 var 值指示的内存地址的字节中

2) **inc/dec** 指令。inc、dec 指令分别表示将操作数自加 1、自减 1。

它们的语法如下：

```
inc <reg> / dec <reg>
inc <mem> / dec <mem>
```

举例：

dec eax	#eax 值自减 1
inc dword ptr [var]	#var 值指示的内存地址的 4 字节值自加 1

3) **imul** 指令。带符号整数乘法指令，有两种格式：①两个操作数，将两个操作数相乘，将结果保存在第一个操作数中，第一个操作数必须为寄存器；②三个操作数，将第二个和第三个操作数相乘，将结果保存在第一个操作数中，第一个操作数必须为寄存器。

其语法如下：

```
imul <reg32>,<reg32>
imul <reg32>,<mem>
imul <reg32>,<reg32>,<con>
imul <reg32>,<mem>,<con>
```

举例：

imul eax, [var]	#eax ← eax * [var]
imul esi, edi, 25	#esi ← edi * 25

乘法操作结果可能溢出，则编译器置溢出标志 OF = 1，以便 CPU 调出溢出异常处理程序。

4) **idiv** 指令。带符号整数除法指令，它只有一个操作数，即除数，而被除数则为 edx:eax 中的内容（64 位整数），操作结果有两部分：商和余数，商送到 eax，余数则送到 edx。

其语法如下：

```
idiv <reg32>
idiv <mem>
```

举例：

idiv ebx	
idiv dword ptr [var]	

5) **and/or/xor** 指令。and、or、xor 指令分别是逻辑与、逻辑或、逻辑异或操作指令，用于操作数的位操作，操作结果放在第一个操作数中。

它们的语法如下：

```
and <reg>,<reg> / or <reg>,<reg> / xor <reg>,<reg>
and <reg>,<mem> / or <reg>,<mem> / xor <reg>,<mem>
and <mem>,<reg> / or <mem>,<reg> / xor <mem>,<reg>
and <reg>,<con> / or <reg>,<con> / xor <reg>,<con>
and <mem>,<con> / or <mem>,<con> / xor <mem>,<con>
```

举例：

and eax, 0FH	#将 eax 中的前 28 位全部置为 0，最后 4 位保持不变
--------------	----------------------------------

```
xor edx, edx          #置 edx 中的内容为 0
```

6) **not** 指令。位翻转指令, 将操作数中的每一位翻转, 即 $0 \rightarrow 1, 1 \rightarrow 0$ 。

其语法如下:

```
not <reg>
not <mem>
```

举例:

```
not byte ptr [var]      #将 var 值指示的内存地址的一字节的所有位翻转
```

7) **neg** 指令。取负指令。

其语法如下:

```
neg <reg>
neg <mem>
```

举例:

```
neg eax                  #eax ← -eax
```

8) **shl/shr** 指令。逻辑移位指令, shl 为逻辑左移, shr 为逻辑右移, 第一个操作数表示被操作数, 第二个操作数指示移位的位数。

它们的语法如下:

```
shl <reg>, <con8> / shr <reg>, <con8>
shl <mem>, <con8> / shr <mem>, <con8>
shl <reg>, <cl> / shr <reg>, <cl>
shl <mem>, <cl> / shr <mem>, <cl>
```

关注公众号【乘龙考研】
一手更新 稳定有保障

举例:

```
shl eax, 1              #将 eax 值左移 1 位
shr ebx, cl             #将 ebx 值右移 n 位 (n 为 cl 中的值)
```

(3) 控制流指令

x86 处理器维持着一个指示当前执行指令的指令指针 (IP), 当一条指令执行后, 此指针自动指向下一条指令。IP 寄存器不能直接操作, 但可以用控制流指令更新。通常用标签 (label) 指示程序中的指令地址, 在 x86 汇编代码中, 可在任何指令前加入标签。例如,

```
mov esi, [ebp+8]
begin: xor ecx, ecx
       mov eax, [esi]
```

这样就用 begin 指示了第二条指令, 控制流指令通过标签就可以实现程序指令的跳转。

1) **jmp** 指令。jmp 指令控制 IP 转移到 label 所指示的地址 (从 label 中取出指令执行)。

其语法如下:

```
jmp <label>
```

举例:

```
jmp begin               #转跳到 begin 标记的指令执行
```

2) **jcondition** 指令。条件转移指令, 依据 CPU 状态字中的一系列条件状态转移。CPU 状态字中包括指示最后一个算术运算结果是否为 0, 运算结果是否为负数等。

其语法如下:

```
je <label> (jump when equal)
jne <label> (jump when not equal)
jz <label> (jump when last result was zero)
jg <label> (jump when greater than)
jge <label> (jump when greater than or equal to)
jl <label> (jump when less than)
jle <label> (jump when less than or equal to)
```

举例：

```
cmp eax, ebx
jle done #如果 eax 的值小于或等于 ebx 值, 跳转到 done 指示的指令执行, 否则执行下一条指令。
```

3) **cmp/test** 指令。cmp 指令用于比较两个操作数的值, test 指令对两个操作数进行逐位与运算, 这两类指令都不保存操作结果, 仅根据运算结果设置 CPU 状态字中的条件码。

其语法如下：

```
cmp <reg>,<reg> / test <reg>,<reg>
cmp <reg>,<mem> / test <reg>,<mem>
cmp <mem>,<reg> / test <mem>,<reg>
cmp <reg>,<con> / test <reg>,<con>
```

关注公众号【乘龙考研】
一手更新 稳定有保障

cmp 和 test 指令通常和 jcondition 指令搭配使用, 举例:

```
cmp dword ptr [var], 10 #将 var 指示的主存地址的 4 字节内容, 与 10 比较
jne loop #如果相等则继续顺序执行; 否则跳转到 loop 处执行
test eax, eax #测试 eax 是否为零
jz xxxx #为零则置标志 ZF 为 1, 转跳到 xxxx 处执行
```

4) **call/ret** 指令。分别用于实现子程序（过程、函数等）的调用及返回。

其语法如下：

```
call <label>
ret
```

call 指令首先将当前执行指令地址入栈, 然后无条件转移到由标签指示的指令。与其他简单的跳转指令不同, call 指令保存调用之前的地址信息(当 call 指令结束后, 返回调用之前的地址)。ret 指令实现子程序的返回机制, ret 指令弹出栈中保存的指令地址, 然后无条件转移到保存的指令地址执行。call 和 ret 是程序(函数)调用中最关键的两条指令。

理解上述指令的语法和用途, 可以更好地帮助读者解答相关题型。读者在上机调试 C 程序代码时, 也可以尝试用编译器调试, 以便更好地帮助理解机器指令的执行。

4.3.2 过程调用的机器级表示

上面提到的 call/ret 指令主要用于过程调用, 它们都属于一种无条件转移指令。

假定过程 P(调用者)调用过程 Q(被调用者), 过程调用的执行步骤如下:

- 1) P 将入口参数(实参)放在 Q 能访问到的地方。
- 2) P 将返回地址存到特定的地方, 然后将控制转移到 Q。
- 3) Q 保存 P 的现场(通用寄存器的内容), 并为自己的非静态局部变量分配空间。
- 4) 执行过程 Q。
- 5) Q 恢复 P 的现场, 将返回结果放到 P 能访问到的地方, 并释放局部变量所占空间。
- 6) Q 取出返回地址, 将控制转移到 P。

步骤 2) 是由 call 指令实现的, 步骤 6) 通过 ret 指令返回到过程 P。在上述步骤中, 需要为入口参数、返回地址、过程 P 的现场、过程 Q 的局部变量、返回结果找到存放空间。但用户可见寄存器数量有限, 为此需要设置一个专门的存储区域来保存这些数据, 这个存储区域就是栈。寄存器 EAX、ECX 和 EDX 是调用者保存寄存器, 其保存和恢复的任务由过程 P 负责, 当 P 调用 Q 时, Q 就可以直接使用这三个寄存器。寄存器 EBX、ESI、EDI 是被调用者保存寄存器, Q 必须先将它们的值保存在栈中才能使用它们, 并在返回 P 之前先恢复它们的值。

每个过程都有自己的栈区, 称为栈帧, 因此, 一个栈由若干栈帧组成。帧指针寄存器 EBP 指示栈帧的起始位置(栈底), 栈指针寄存器 ESP 指示栈顶, 栈从高地址向低地址增长, 因此,

当前栈帧的范围在帧指针 EBP 和 ESP 指向的区域之间。

下面用一个简单的 C 语言程序来说明过程调用的机器级实现。

```
int add(int x, int y){  
    return x+y;  
}  
int caller(){  
    int templ=125;  
    int temp2=80;  
    int sum=add(templ,temp2);  
    return sum;  
}
```

关注公众号【乘龙考研】
一手更新 稳定有保障

经 GCC 编译后, caller 过程对应的代码如下 (#后面的文字是注释):

caller:	
pushl %ebp	
movl %esp,%ebp	
subl \$24,%esp	
movl \$125,-12(%ebp)	#M[R[ebp]-12]←125, 即 templ=125
movl \$80,-8(%ebp)	#M[R[ebp]-8]←80, 即 temp2=80
movl -8(%ebp),%eax	#R[eax]←M[R[ebp]-8], 即 R[eax]=temp2
mov %eax,4(%esp)	#M[R[esp]+4]←R[eax], 即 temp2 入栈
movl -12(%ebp),%eax	#R[eax]←M[R[ebp]-12], 即 R[eax]=templ
movl %eax,(%esp)	#M[R[esp]]←R[eax], 即 templ 入栈
call add	#调用 add, 将返回值保存在 EAX 中
movl %eax,-4(%ebp)	#M[R[ebp]-4]←R[eax], 即 add 返回值送 sum
movl -4(%ebp),%eax	#R[eax]←M[R[ebp]-4], 即 sum 作为 caller 返回值
leave	
ret	

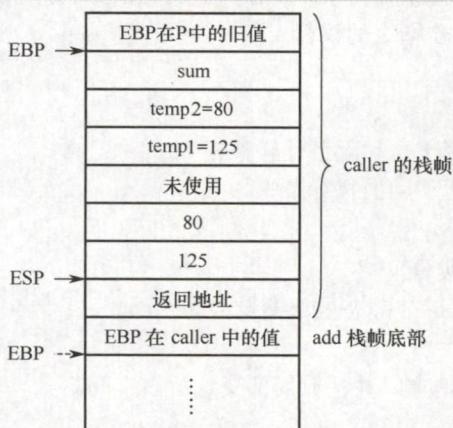


图 4.12 caller 和 add 的栈帧

图 4.12 给出了 caller 栈帧的状态, 假定 caller 被过程 P 调用。执行了第 4 行指令后 ESP 所指的位置如图中所示, 可以看出 GCC 为 caller 的参数分配了 24 字节的空间。从汇编代码中可以看出, caller 中只使用了调用者保存寄存器 EAX, 没有使用任何被调用者保存寄存器, 因而在 caller 栈帧中无须保存除 EBP 外的任何寄存器的值; caller 有三个局部变量 templ、temp2 和 sum, 皆被分配在栈帧中; 在用 call 指令调用 add 函数之前, caller 先将入口参数从右向左依次将 temp2 和 templ 的值 (即 80 和 125) 保存到栈中。在执行 call 指令时再把返回地址压入栈中。此外, 在最初进入 caller 时, 还将 EBP 的值压入了栈, 因此 caller 的栈帧中用到的空间占 $4 + 12 + 8 + 4 = 28$ 字节。但是, caller 的栈帧共有 $4 + 24 + 4 = 32$

字节, 其中浪费了 4 字节的空间 (未使用)。这是因为 GCC 为保证数据的严格对齐而规定每个函数的栈帧大小必须是 16 字节的倍数。

call 指令执行后, add 函数的返回参数放到 EAX 中, 因而在 call 指令后面的两条指令中, 指令 “movl %eax, -4(%ebp)” 将 add 的结果存入 sum 变量的存储空间, 该变量的地址为 R[ebp]-4; 指令 “movl -4(%ebp), %eax” 将 sum 变量的值作为返回值送到寄存器 EAX 中。

在执行 ret 指令之前, 应将当前栈帧释放, 并恢复旧 EBP 的值, 上述第 14 行 leave 指令实现

了这个功能，leave 指令功能相当于以下两条指令的功能：

```
movl    %ebp, %esp
popl    %ebp
```

其中，第一条指令使 ESP 指向当前 EBP 的位置，第二条指令执行后，EBP 恢复为 P 中的旧值，并使 ESP 指向返回地址。

执行完 leave 指令后，ret 指令就可从 ESP 所指处取返回地址，以返回 P 执行。当然，编译器也可通过 pop 指令和对 ESP 的内容做加法来进行退栈操作，而不一定要使用 leave 指令。

add 过程经 GCC 编译并进行链接后，对应的代码如下所示：

8048469:55	push	%ebp
804846a:89 e5	mov	%esp,%ebp
804846c:8b 45 0c	mov	0xc(%ebp),%eax
804846f:8b 55 08	mov	0x8(%ebp),%edx
8048472:8d 04 02	lea	(%edx,%eax,1),%eax
8048475:5d	pop	%ebp
8048476:c3	ret	

通常，一个过程对应的机器级代码都有三个部分：准备阶段、过程体和结束阶段。

上述第 1、2 行指令构成准备阶段的代码段，这是最简单的准备阶段代码段，它通过将当前栈指针 ESP 传递到 EBP 来完成将 EBP 指向当前栈帧底部的任务，如图 4.12 所示，EBP 指向 add 栈帧底部，从而可以方便地通过 EBP 获取入口参数。这里 add 的入口参数 x 和 y 对应的值（125 和 80）分别在地址为 R[ebp] + 8、R[ebp] + 12 的存储单元中。

上述第 3、4、5 行指令序列是过程体的代码段，过程体结束时将返回值放到 EAX 中。这里好像没有加法指令，实际上第 5 行 lea 指令执行的是加法运算 $R[\text{edx}] + R[\text{eax}] * 1 = \text{x} + \text{y}$ 。

上述第 6、7 行指令序列是结束阶段的代码段，通过将 EBP 弹出栈帧来恢复 EBP 在 caller 过程中的值，并在栈中退出 add 过程的栈帧，使得执行到 ret 指令时栈顶中已经是返回地址。这里的返回地址应该是 caller 代码中第 12 行指令“`movl %eax, -4(%ebp)`”的地址。

add 过程中没有用到任何被调用者保存寄存器，没有局部变量，此外，add 是一个被调用过程，并且不再调用其他过程，因而也没有入口参数和返回地址要保存，因此，在 add 的栈帧中除了需要保存 EBP，无须保留其他任何信息。

4.3.3 选择语句的机器级表示

常见的选择结构语句有 if-then、if-then-else、case（或 switch）等。编译器通过条件码（标志位）设置指令和各类转移指令来实现程序中的选择结构语句。

（1）条件码（标志位）

除了整数寄存器，CPU 还维护一组条件码（标志位）寄存器，它们描述了最近的算术或逻辑运算操作的属性。可以检测这些寄存器来执行条件分支指令，最常用的条件码如下：

- **CF：**进（借）位标志。最近无符号整数加（减）运算后的进（借）位情况。有进（借）位时，CF = 1；否则 CF = 0。
- **ZF：**零标志。最近的操作的运算结果是否为 0。若结果为 0，ZF = 1；否则 ZF = 0。
- **SF：**符号标志。最近的带符号数运算结果的符号。若为负，SF = 1；否则 SF = 0。
- **OF：**溢出标志。最近的带符号数运算结果是否溢出。若溢出，OF = 1；否则 OF = 0。

可见，OF 和 SF 对无符号数运算来说没有意义，而 CF 对带符号数运算来说没有意义。

常见的算术逻辑运算指令（add、sub、imul、or、and、shl、inc、dec、not、sal 等）会设置条件码。但有两类指令只设置条件码而不改变任何其他寄存器：cmp 指令和 sub 指令的行为一样，

关注公众号【乘龙考研】
一手更新 稳定有保障

test 指令与 and 指令的行为一样，但是它们只设置条件码，而不更新目的寄存器。

之前介绍的 **jcondition** 条件转跳指令，就是根据条件码 ZF 和 SF 来实现转跳的。

(2) if 语句

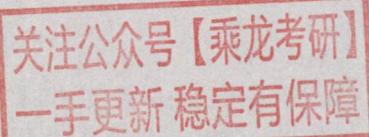
if-else 语句的通用形式如下：

```
if(test_expr)
    then_statement
else
    else_statement
```

这里的 test_expr 是一个整数表达式，它的取值为 0（假），或为非 0（真）。两个分支语句 (then_statement 或 else_statement) 中只会执行一个。

这种通用形式可以被翻译成如下所示的 goto 语句形式：

```
t=test_expr;
if(!t)
    goto false;
then_statement
goto done;
false:
else_statement
done:
```



对于下面的 C 语言函数：

```
int get_cont(int *p1, int *p2) {
    if(p1>p2)
        return *p2;
    else
        return *p1;
}
```

已知 p1 和 p2 对应的实参已被压入调用函数的栈帧，它们对应的存储地址分别为 R[ebp]+8、R[ebp]+12（EBP 指向当前栈帧底部），返回结果存放在 EAX 中。对应的汇编代码为

movl 8(%ebp), %eax	# $R[\text{eax}] \leftarrow M[R[\text{ebp}]+8]$ ，即 $R[\text{eax}] = p1$
movl 12(%ebp), %edx	# $R[\text{edx}] \leftarrow M[R[\text{ebp}]+12]$ ，即 $R[\text{edx}] = p2$
cmpl %edx, %eax	#比较 p1 和 p2，即根据 $p1 - p2$ 的结果置标志
jbe .L1	#若 $p1 \leq p2$ ，则转标记 L1 处执行
movl (%edx), %eax	# $R[\text{eax}] \leftarrow M[R[\text{edx}]]$ ，即 $R[\text{eax}] = M[p2]$
jmp .L2	#无条件跳转到标记 L2 执行
.L1:	
movl (%eax), %eax	# $R[\text{eax}] \leftarrow M[R[\text{eax}]]$ ，即 $R[\text{eax}] = M[p1]$
.L2:	

p1 和 p2 是指针型参数，故在 32 位机中的长度后缀是 l，比较指令 cmpl 的两个操作数都应来自寄存器，故应先将 p1 和 p2 对应的实参从栈中取到通用寄存器，比较指令执行后得到各个条件标志位，然后根据各条件标志值的组合选择执行不同的指令，因此需要用到条件转移指令。

4.3.4 循环语句的机器级表示

常见的循环结构语句有 while、for 和 do-while。汇编中没有相应的指令存在，可以用条件测试和转跳组合起来实现循环的效果，大多数编译器将这三种循环结构都转换为 do-while 形式来产生机器代码。在循环结构中，通常使用条件转移指令来判断循环条件的结束。

(1) do-while 循环

do-while 语句的通用形式如下：

```
do
    body_statement
    while(test_expr);
```

这种通用形式可以被翻译成如下所示的条件和 goto 语句：

```
loop:
body_statement
t=test_expr;
if(t)
    goto loop;
```

关注公众号【乘龙考研】
一手更新 稳定有保障

也就是说，每次循环，程序会执行循环体内的语句，body_statement 至少会执行一次，然后执行测试表达式。如果测试为真，就继续执行循环。

(2) while 循环

while 语句的通用形式如下：

```
while(test_expr)
    body_statement
```

与 do-while 的不同之处在于，第一次执行 body_statement 之前，就会测试 test_expr 的值，循环有可能中止。GCC 通常会将其翻译成条件分支加 do-while 循环的方式。

用如下模板来表达这种方法，将通用的 while 循环格式翻译成 do-while 循环：

```
t=test_expr;
if(!t)
    goto done;
do
    body_statement
    while(test_expr);
done:
```

相应地，进一步将它翻译成 goto 语句：

```
t=test_expr;
if(!t)
    goto done;
loop:
body_statement
t=test_expr;
if(t)
    goto loop;
done:
```

(3) for 循环

for 循环的通用形式如下：

```
for(init_expr; test_expr; update_expr)
    body_statement
```

这个 for 循环的行为与下面这段 while 循环代码的行为一样：

```
init_expr;
while(test_expr) {
    body_statement
    update_expr;
}
```

进一步把它翻译成 goto 语句：

```
init_expr;
t=test_expr;
```

```

if(!t)
    goto done;
loop:
body_statement
update_expr;
t=test_expr;
if(t)
    goto loop;
done:

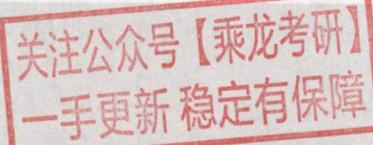
```

下面是一个用 for 循环写的自然数求和的函数：

```

int nsum_for(int n){
    int i;
    int result = 0;
    for(i=1;i<=n;i++)
        result +=i;
    return result;
}

```



这段代码中的 for 循环的不同组成部分如下：

init_expr	i=1
test_expr	i<=n
update_expr	i++
body_statement	result +=i

通过替换前面给出的模板中的相应位置，很容易将 for 循环转换为 while 或 do-while 循环。

将这个函数翻译为 goto 语句代码后，不难得出其过程体的汇编代码：

```

movl 8(%ebp),%ecx      #R[ecx]←M[R[ebp]+8], 即 R[ecx]=n
movl $0,%eax           #R[eax]←0, 即 result=0
movl $1,%edx            #R[edx]←1, 即 i=1
cmp   %ecx,%edx         #Compare R[ecx]:R[edx], 即比较 i:n
jg   .L2                #If greater, 转跳到 L2 执行
.L1:
loop:
addl %edx,%eax          #R[eax]←R[eax]+R[edx], 即 result +=i
addl $1,%edx             #R[edx]←R[edx]+1, 即 i++
cmpl %ecx,%edx           #比较 %ecx 和 %edx, 即比较 i:n
jle .L1                 #If less or equal, 转跳到 L1 执行
.L2:

```

已知 n 对应的实参已被压入调用函数的栈帧，其对应的存储地址为 R[ebp]+8，过程 nsum_for 中的局部变量 i 和 result 被分别分配到寄存器 EDX 和 EAX 中，返回参数在 EAX 中。

4.3.5 本节习题精选

一、单项选择题

01. 假设 R[ax] = FFE8H, R[bx] = 7FE6H, 执行指令 “addw %bx, %ax” 后，寄存器的内容和各标志的变化为（）。
 - A. R[ax] = 7FCEH, OF = 1, SF = 0, CF = 0, ZF = 0
 - B. R[bx] = 7FCEH, OF = 1, SF = 0, CF = 0, ZF = 0
 - C. R[ax] = 7FCEH, OF = 0, SF = 0, CF = 1, ZF = 0
 - D. R[bx] = 7FCEH, OF = 0, SF = 0, CF = 1, ZF = 0
02. 假设 R[ax] = 7FE6H, R[bx] = FFE8H, 执行指令 “sub bx, ax” 后，寄存器的内存和各标

志的变化为()。

- A. R[ax] = 8002H, OF = 0, SF = 1, CF = 1, ZF = 0
- B. R[bx] = 8002H, OF = 0, SF = 1, CF = 1, ZF = 0
- C. R[ax] = 8002H, OF = 1, SF = 1, CF = 0, ZF = 0
- D. R[bx] = 8002H, OF = 1, SF = 1, CF = 0, ZF = 0

03. 假设 P 为调用过程, Q 为被调用过程, 程序在 32 位 x86 处理器上执行, 以下是 C 语言程序中过程调用所涉及的操作:

- ① 过程 Q 保存 P 的现场, 并为非静态局部变量分配空间
- ② 过程 P 将实参存放到 Q 能访问到的地方
- ③ 过程 P 将返回地址存放到特定处, 并转跳到 Q 执行
- ④ 过程 Q 取出返回地址, 并转跳回到过程 P 执行
- ⑤ 过程 Q 恢复 P 的现场, 并释放局部变量所占空间
- ⑥ 执行过程 Q 的函数体

过程调用的正确执行步骤是()。

- A. ②→③→④→①→⑤→⑥
- B. ②→③→①→④→⑥→⑤
- C. ②→③→①→⑥→⑤→④
- D. ②→③→①→⑤→⑥→④

二、综合应用题

01. 【2017 统考真题】在按字节编址的计算机 M 上, f1 的部分源程序(阴影部分)如下。将 f1 中的 int 都改成 float, 可得到计算 f(n) 的另一个函数 f2。

```
int f1(unsigned n){
    int sum=1, power=1;
    for(unsigned i=0; i<=n-1; i++) {
        power *=2;
        sum += power;
    }
    return sum;
}
```

对应的机器级代码(包括指令的虚拟地址)如下:

	int f1(unsigned n)		
1	00401020	55	push ebp

	for(unsigned i=0; i<= n - 1; i++)		
20	0040105E	39 4D F4	cmp dword ptr [ebp-0Ch], ecx

		power *= 2;	

23	00401066	D1 E2	shl edx,1

	return sum;		

35	0040107F	C3	ret

其中, 机器级代码行包括行号、虚拟地址、机器指令和汇编指令。

1) 计算机 M 是 RISC 还是 CISC? 为什么?



- 2) f1 的机器指令代码共占多少字节？要求给出计算过程。
- 3) 第 20 条指令 cmp 通过 i 减 n-1 实现对 i 和 n-1 的比较。执行 f1(0) 的过程中，当 i=0 时，cmp 指令执行后，进位/借位标志 CF 的内容是什么？要求给出计算过程。
- 4) 第 23 条指令 shl 通过左移操作实现了 power *2 运算，在 f2 中能否用 shl 指令实现 power *2？为什么？

02. 【2019 统考真题】已知 $f(n) = n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$ ，计算 $f(n)$ 的 C 语言函数 f1 的源程序（阴影部分）及其在 32 位计算机 M 上的部分机器级代码如下：

```

int f1(int n){
    1      00401000  55          push  ebp
    ...
    if(n>1)
    11     00401018  83 7D 08 01    cmp   dword ptr [ebp+8],1
    12     0040101C  7E 17          jle   f1+35h (00401035)
        return n*f1(n-1);
    13     0040101E  8B 45 08          mov   eax, dword ptr [ebp+8]
    14     00401021  83 E8 01          sub   eax, 1
    15     00401024  50          push  eax
    16     00401025  E8 D6 FF FF FF  call  f1 ( 00401000)
    ...
    19     00401030  0F AF C1          imul  eax, ecx
    20     00401033  EB 05          jmp   f1+3Ah (0040103a)
        else return 1;
    21     00401035  B8 01 00 00 00  mov   eax,1
}
...
26     00401040  3B EC          cmp   ebp, esp
...
30     0040104A  C3          ret

```

其中，机器级代码行包括行号、虚拟地址、机器指令和汇编指令，计算机 M 按字节编址，int 型数据占 32 位。请回答下列问题：

- 1) 计算 $f(10)$ 需要调用函数 f1 多少次？执行哪条指令会递归调用 f1？
 - 2) 上述代码中，哪条指令是条件转移指令？哪几条指令一定会使程序跳转执行？
 - 3) 根据第 16 行的 call 指令，第 17 行指令的虚拟地址应是多少？已知第 16 行的 call 指令采用相对寻址方式，该指令中的偏移量应是多少（给出计算过程）？已知第 16 行的 call 指令的后 4 字节为偏移量，M 是采用大端方式还是采用小端方式？
 - 4) $f(13) = 6227020800$ ，但 f1(13) 的返回值为 1932053504，为什么两者不相等？要使 f1(13) 能返回正确的结果，应如何修改 f1 的源程序？
 - 5) 第 19 行的 imul 指令（带符号整数乘）的功能是 $R[eax] \leftarrow R[eax] \times R[ecx]$ ，当乘法器输出的高、低 32 位乘积之间满足什么条件时，溢出标志 OF = 1？要使 CPU 在发生溢出时转异常处理，编译器应在 imul 指令后加一条什么指令？
- 03. 【2019 统考真题】**对于题 2，若计算机 M 的主存地址为 32 位，采用分页存储管理方式，页大小为 4KB，则第 1 行的 push 指令和第 30 行的 ret 指令是否在同一页中（说明理由）？若指令 Cache 有 64 行，采用 4 路组相联映射方式，主存块大小为 64B，则 32 位主存地

址中，哪几位表示块内地址？哪几位表示 Cache 组号？哪几位表示标记（tag）信息？读取第 16 行的 call 指令时，只可能在指令 Cache 的哪一组中命中（说明理由）？

4.3.6 答案与解析

一、单项选择题

01. C

该指令是 AT&T 格式，add 指令的目的寄存器为 ax。add 指令的补码加法过程为 $1111\ 1111\ 1110\ 1000 + 0111\ 1111\ 1110\ 0110 = (1)0111\ 1111\ 1100\ 1110$ (7FCEH)，两个操作数的符号不同，必然不会溢出，OF = 0；结果的符号位为 0，SF = 0；有进位，CF = C \oplus Sub = 1 \oplus 0 = 1；非 0，ZF = 0。

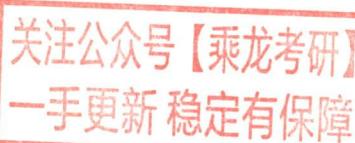
注意：不论是无符号数还是带符号数，都以二进制代码形式无差别地存放在机内，计算机并不知道哪些数是无符号数，哪些数是带符号数。即便是两个带符号数相加，也会导致 CF 的变动，只是 CF 的结果对带符号数是没有意义的。同理，两个无符号数相加，也会导致 OF 和 SF 的变动，只是 OF 和 SF 的结果仅对带符号数有意义。

02. B

该指令是 Intel 格式，sub 指令的目的寄存器为 bx。sub 减法运算用补码加法实现，被减数 + 减数逐位取反 + 1 = $1111\ 1111\ 1110\ 1000 + 1000\ 0000\ 0001\ 1001 + 1 = (1)1000\ 0000\ 0000\ 0010$ (8002H)，两个操作数的符号位都是 1，结果的符号位也是 1，无溢出，OF = 0；结果为负数，SF = 1；进位输出 $C_{out} = 1$ ，低位进位 Sub = 1，CF = $C_{out} \oplus Sub = 1 \oplus 1 = 1$ ；非 0，ZF = 0。

03. C

过程调用的具体过程已在 4.3.2 节中介绍。



二、综合应用题

01. 【解答】

- 1) M 为 CISC。M 的指令长短不一，不符合 RISC 指令系统的特点。
- 2) f1 的机器代码占 96B。因为 f1 的第一条指令“push ebp”所在的虚拟地址为 0040 1020H，最后一条指令“ret”所在的虚拟地址为 0040 107FH，所以 f1 的机器指令代码长度为 $0040\ 107FH - 0040\ 1020H + 1 = 60H = 96B$ 。
- 3) CF = 1。cmp 指令实现 i 与 n - 1 的比较功能，进行的是减法运算。在执行 f1(0)的过程中，n = 0，当 i = 0 时，i = 0000 0000H，并且 n - 1 = FFFF FFFFH。因此，执行第 20 条指令时，在补码加/减运算器中执行“0 减 FFFF FFFFH”操作，即 $0000\ 0000H + 00000000H + 1 = 0000\ 0001H$ ，此时进位输出 $C_{out} = 0$ ，低位进位 Sub = 1，CF = $C_{out} \oplus Sub = 0 \oplus 1 = 1$ 。
- 4) f2 中不能用 shl 指令实现 power *2。因为 shl 指令把一个整数的所有有效数位整体左移，而 f2 中的变量 power 是 float 型，其机器数中不包含最高有效数位，但包含了阶码部分，将其作为一个整体左移时并不能实现“乘 2”的功能，因而 f2 中不能用 shl 指令实现 power *2。浮点数运算比整型运算要复杂，耗时也较长。

02. 【解答】

- 1) 计算 f(10) 需要调用函数 f1 共 10 次，执行第 16 行的 call 指令会递归调用 f1。
- 2) 第 12 行的 jle 指令是条件转移指令，其含义为小于或等于时转移，本行代码的意义为：当 $n \leq 1$ 时，跳转至地址 0040 1035H。第 16 行的 call 指令为函数调用指令，第 20 行的 jmp 指令为无条件转移指令，第 30 行的 ret 指令为子程序的返回指令，这三条指令一定

会使程序跳转执行。

- 3) 在计算机 M 上按字节编址, 第 16 行的 call 指令的虚拟地址为 0040 1025H, 长度为 5 字节, 因此第 17 行的指令的虚拟地址为 $0040\ 1025H + 5 = 0040\ 102AH$ 。第 16 行的 call 指令采用相对寻址方式, 即目标地址 = (PC) + 偏移量, call 指令的目标地址为 0040 1000H, 所以偏移量 = 目标地址 - (PC) = $0040\ 1000H - 0040\ 102AH = FFFF\ FFD6H$ 。根据第 16 行的 call 指令的偏移量字段为 D6 FF FF FF, 可以确定 M 采用小端方式。
- 4) 因为 $f(13) = 6227020800$, 其结果超出了 32 位 int 型数据可表示的最大范围, 因此 $f(13)$ 的返回值是一个发生了溢出的错误结果。为使 $f(13)$ 能返回正确结果, 可将函数 $f1$ 的返回值类型改为 double (或 long long, 或 long double, 或 float) 类型。
- 5) 若乘积的高 33 位为非全 0 或非全 1, 则 OF = 1。编译器应在 imul 指令后加一条“溢出自陷指令”, 使得 CPU 自动查询溢出标志 OF, 当 OF = 1 时调出“溢出异常处理程序”。

03. 【解答】

因为页大小为 4KB, 所以虚拟地址的高 20 位为虚拟页号。第 1 行的 push 指令和第 30 行的 ret 指令的虚拟地址的高 20 位都是 00401H, 因此两条指令在同一页中。

指令 Cache 有 64 块, 采用 4 路组相联映射方式, 因此指令 Cache 共有 $64/4 = 16$ 组, Cache 组号共 4 位。主存块大小为 64B, 因此块内地址为低 6 位。综上所述, 在 32 位主存地址中, 低 6 位为块内地址, 中间 4 位为组号, 高 22 位为标记。

因为页大小为 4KB, 所以虚拟地址和物理地址的最低 12 位完全相同, 因而 call 指令虚拟地址 0040 1025H 中的 025H = 0000 0010 0101B 为物理地址的低 12 位, 对应的 7~10 位为组号, 因此对应的 Cache 组号为 0。

关注公众号【乘龙考研】
一手更新 稳定有保障

4.4 CISC 和 RISC 的基本概念

指令系统朝两个截然不同的方向的发展: 一是增强原有指令的功能, 设置更为复杂的新指令实现软件功能的硬化, 这类机器称为复杂指令系统计算机 (CISC), 典型的有采用 x86 架构的计算机; 二是减少指令种类和简化指令功能, 提高指令的执行速度, 这类机器称为精简指令系统计算机 (RISC), 典型的有 ARM、MIPS 架构的计算机。

4.4.1 复杂指令系统计算机 (CISC)

随着 VLSI 技术的发展, 硬件成本不断下降, 软件成本不断上升, 促使人们在指令系统中增加更多、更复杂的指令, 以适应不同的应用领域, 这样就构成了复杂指令系统计算机 (CISC)。

CISC 的主要特点如下:

- 1) 指令系统复杂庞大, 指令数目一般为 200 条以上。
- 2) 指令的长度不固定, 指令格式多, 寻址方式多。
- 3) 可以访存的指令不受限制。
- 4) 各种指令使用频度相差很大。
- 5) 各种指令执行时间相差很大, 大多数指令需多个时钟周期才能完成。
- 6) 控制器大多数采用微程序控制。有些指令非常复杂, 以至于无法采用硬连线控制。
- 7) 难以用优化编译生成高效的目标代码程序。

如此庞大的指令系统, 对指令的设计提出了极高的要求, 研制周期变得很长。后来人们发现,

一味地追求指令系统的复杂和完备程度不是提高计算机性能的唯一途径。对传统 CISC 指令系统的测试表明，各种指令的使用频率相差悬殊，大概只有 20% 的比较简单的指令被反复使用，约占整个程序的 80%；而 80% 左右的指令则很少使用，约占整个程序的 20%。从这一事实出发，人们开始了对指令系统合理性的研究，于是 RISC 随之诞生。

4.4.2 精简指令系统计算机（RISC）

精简指令系统计算机（RISC）的中心思想是要求指令系统简化，尽量使用寄存器-寄存器操作指令，指令格式力求一致。RISC 的主要特点如下：

- 1) 选取使用频率最高的一些简单指令，复杂指令的功能由简单指令的组合来实现。
- 2) 指令长度固定，指令格式种类少，寻址方式种类少。
- 3) 只有 Load/Store（取数/存数）指令访存，其余指令的操作都在寄存器之间进行。
- 4) CPU 中通用寄存器的数量相当多。
- 5) RISC 一定采用指令流水线技术，大部分指令在一个时钟周期内完成。
- 6) 以硬布线控制为主，不用或少用微程序控制。
- 7) 特别重视编译优化工作，以减少程序执行时间。

值得注意的是，从指令系统兼容性看，CISC 大多能实现软件兼容，即高档机包含了低档机的全部指令，并可加以扩充。但 RISC 简化了指令系统，指令条数少，格式也不同于老机器，因此大多数 RISC 机不能与老机器兼容。由于 RISC 具有更强的实用性，因此应该是未来处理器的发展方向。但事实上，当今时代 Intel 几乎一统江湖，且早期很多软件都是根据 CISC 设计的，单纯的 RISC 将无法兼容。此外，现代 CISC 结构的 CPU 已经融合了很多 RISC 的成分，其性能差距已经越来越小。CISC 可以提供更多的功能，这是程序设计所需要的。

4.4.3 CISC 和 RISC 的比较

和 CISC 相比，RISC 的优点主要体现在以下几点：

- 1) RISC 更能充分利用 VLSI 芯片的面积。CISC 的控制器大多采用微程序控制，其控制存储器在 CPU 芯片内所占的面积达 50% 以上，而 RISC 控制器采用组合逻辑控制，其硬布线逻辑只占 CPU 芯片面积的 10% 左右。
- 2) RISC 更能提高运算速度。RISC 的指令数、寻址方式和指令格式种类少，又设有多个通用寄存器，采用流水线技术，所以运算速度更快，大多数指令在一个时钟周期内完成。
- 3) RISC 便于设计，可降低成本，提高可靠性。RISC 指令系统简单，因此机器设计周期短；其逻辑简单，因此可靠性高。
- 4) RISC 有利于编译程序代码优化。RISC 指令类型少，寻址方式少，使编译程序容易选择更有效的指令和寻址方式，并适当地调整指令顺序，使得代码执行更高效化。

CISC 和 RISC 的对比见表 4.3。

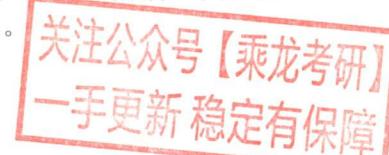


表 4.3 CISC 与 RISC 的对比

类别 对比项目	CISC	RISC
指令系统	复杂，庞大	简单，精简
指令数目	一般大于 200 条	一般小于 100 条
指令字长	不固定	定长
可访存指令	不加限制	只有 Load/Store 指令
各种指令执行时间	相差较大	绝大多数在一个周期内完成

(续表)

类别 对比项目	CISC	RISC
各种指令使用频度	相差很大	都比较常用
通用寄存器数量	较少	多
目标代码	难以用优化编译生成高效的目标代码程序	采用优化的编译程序，生成代码较为高效
控制方式	绝大多数为微程序控制	绝大多数为组合逻辑控制
指令流水线	可以通过一定方式实现	必须实现

4.4.4 本节习题精选

单项选择题

01. 以下叙述中()是正确的。
 A. RISC 机一定采用流水技术 B. 采用流水技术的机器一定是 RISC 机
 C. RISC 机的兼容性优于 CISC 机 D. CPU 配备很少的通用寄存器
02. 下列描述中，不符合 RISC 指令系统特点的是()。
 A. 指令长度固定，指令种类少
 B. 寻址方式种类尽量减少，指令功能尽可能强
 C. 增加寄存器的数目，以尽量减少访存次数
 D. 选取使用频率最高的一些简单指令，以及很有用但不复杂的指令
03. 以下有关 RISC 的描述中，正确的是()。
 A. 为了实现兼容，新设计的 RISC 是从原来 CISC 系统的指令系统中挑选一部分实现的
 B. 采用 RISC 技术后，计算机的体系结构又恢复到了早期的情况
 C. RISC 的主要目标是减少指令数，因此允许以增加每条指令的功能的方法来减少指令系统所包含的指令数
 D. 以上说法都不对
04. 【2009 统考真题】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
05. 【2011 统考真题】下列指令系统的观点中，有利于实现指令流水线的是()。
 I. 指令格式规整且长度一致 II. 指令和数据按边界对齐存放
 III. 只有 Load/Store 指令才能对操作数进行存储访问
 A. 仅 I、II B. 仅 II、III C. 仅 I、III D. I、II、III

4.4.5 答案与解析

单项选择题

01. A

RISC 必然采用流水线技术，这也是由其指令的特点决定的。而 CISC 则无此强制要求，但为了提高指令执行速度，CISC 也往往采用流水线技术，因此流水线技术并非 RISC 的专利。

02. B

A、C、D 选项都是 RISC 的特点。B 选项中，寻址方式种类尽量减少是正确的，但 RISC 是

尽量简化单条指令的功能，复杂指令的功能由简单指令的组合来实现，而增强指令的功能则是 CISC 的特点。

03. D

RISC 选择一些常用的寄存器型指令，并不是为了兼容 CISC，RISC 也不可能兼容 CISC，A 错误。RISC 只是 CPU 的结构发生变化，基本不影响整个计算机的结构，并且即使是采用 RISC 技术的 CPU，其架构也不可能像早期一样简单，B 错误。RISC 的指令功能简单，通过简单指令的组合来实现复杂指令的功能，C 错误，但 RISC 的主要目标是减少指令数是正确的。

04. A

相对于 CISC，RISC 的特点是：指令条数少；指令长度固定，指令格式和寻址种类少；只有取数/存数指令访问存储器，其余指令的操作均在寄存器之间进行；CPU 中通用寄存器多；大部分指令在一个时钟周期内完成；以硬布线逻辑为主，不用或少用微程序控制。B、C 和 D 都是 RISC 的特点。由于 RISC 的速度快，因此普遍采用硬布线控制器，A 错误。

05. D

指令长度一致、按边界对齐存放、仅 Load/Store 指令访存，这些都是 RISC 的特征，它们使取指令、取操作数的操作简化且时间长度固定，能够有效地简化流水线的复杂度。

4.5 本章小结

本章开头提出的问题的参考答案如下。



1) 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。一台计算机中所有机器指令的集合，称为这台计算机的指令系统。引入指令系统后，避免了用户与二进制代码直接接触，使得用户编写程序更为方便。另外，指令系统是表征一台计算机性能的重要因素，它的格式与功能不仅直接影响到机器的硬件结构，而且也直接影响到系统软件，影响到机器的适用范围。

2) 一般来说，指令分为哪些部分？每部分有什么用处？

一条指令通常包括操作码字段和地址码字段两部分。其中，操作码指出指令中该指令应该执行什么性质的操作和具有何种功能，它是识别指令、了解指令功能与区分操作数地址内容的组成和使用方法等的关键信息。地址码用于给出被操作的信息（指令或数据）的地址，包括参加运算的一个或多个操作数所在的地址、运算结果的保存地址、程序的转移地址、被调用子程序的入口地址等。

3) 对于一个指令系统来说，寻址方式多和少有什么影响？

寻址方式的多样化能让用户编程更为方便，但多重寻址方式会造成 CPU 结构的复杂化（详见下章），也不利于指令流水线的运行。而寻址方式太少虽然能够提高 CPU 的效率，但对于用户而言，少数几种寻址方式会使编程变得复杂，很难满足用户的需求。

4.6 常见问题和易混淆知识点

1. 简述各常见指令寻址方式的特点和适用情况。

立即寻址操作数获取便捷，通常用于给寄存器赋初值。

直接寻址相对于立即寻址，缩短了指令长度。

间接寻址扩大了寻址范围，便于编制程序，易于完成子程序返回。

寄存器寻址的指令字较短，指令执行速度较快。

寄存器间接寻址扩大了寻址范围。

基址寻址扩大了操作数寻址范围，适用于多道程序设计，常用于为程序或数据分配存储空间。

变址寻址主要用于处理数组问题，适合编制循环程序。

相对寻址用于控制程序的执行顺序、转移等。

基址寻址和变址寻址的区别：两种方式有效地址的形成都是寄存器内容 + 偏移地址，但是在基址寻址中，程序员操作的是偏移地址，基址寄存器的内容由操作系统控制，在执行过程中是动态调整的；而在变址寻址中，程序员操作的是变址寄存器，偏移地址是固定不变的。

2. 一个操作数在内存可能占多个单元，怎样在指令中给出操作数的地址？

现代计算机都采用字节编址方式，即一个内存单元只能存放一字节的信息。一个操作数（如 char、int、float、double）可能是 8 位、16 位、32 位或 64 位等，因此可能占用 1 个、2 个、4 个或 8 个内存单元。也就是说，一个操作数可能有多个内存地址对应。

有两种不同的地址指定方式：大端方式和小端方式。

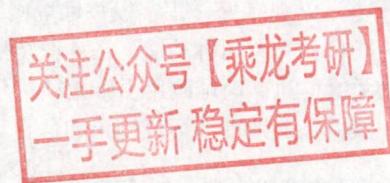
大端方式：指令中给出的地址是操作数最高有效字节（MSB）所在的地址。

小端方式：指令中给出的地址是操作数最低有效字节（LSB）所在的地址。

3. 装入/存储（Load/Store）型指令有什么特点？

装入/存储型指令是用在规整型指令系统中的一种通用寄存器型指令风格。这种指令风格在 RISC 指令系统中较为常见。为了规整指令格式，使指令具有相同的长度，规定只有 Load/Store 指令才能访问内存。而运算指令不能直接访问内存，只能从寄存器取数进行运算，运算的结果也只能送到寄存器。因为寄存器编号较短，而主存地址位数较长，通过某种方式可使运算指令和访存指令的长度一致。

这种装入/存储型风格的指令系统的最大特点是，指令格式规整，指令长度一致，一般为 32 位。由于只有 Load/Store 指令才能访问内存，程序中可能会包含许多装入指令和存储指令，与一般通用寄存器型指令风格相比，其程序长度会更长。



第 5 章 中央处理器

关注公众号【乘龙考研】
一手更新 稳定有保障

【考纲内容】

- (一) CPU 的功能和基本结构
- (二) 指令执行过程
- (三) 数据通路的功能和基本结构
- (四) 控制器的功能和工作原理
- (五) 异常和中断机制
 - 异常和中断的基本概念；异常和中断的分类；异常和中断的检测与响应
- (六) 指令流水线
 - 指令流水线的基本概念；指令流水线的基本实现
 - 结构冒险、数据冒险和控制冒险的处理；超标量和动态流水线的基本概念
- (七) 多处理器基本概念
 - SISD、SIMD、MIMD、向量处理器的基本概念；硬件多线程的基本概念
 - 多核（multi-core）处理器的基本概念；共享内存多处理器（SMP）的基本概念

扫一扫



视频讲解

【复习提示】

中央处理器是计算机的中心，也是本书的难点。其中，数据通路的分析、指令执行阶段的节拍与控制信号的安排、流水线技术与性能分析易出综合题。而关于各种寄存器的特点、指令执行的各种周期与特点、控制器的相关概念、流水线的相关概念也极易出选择题。

在学习本章时，请读者思考以下问题：

- 1) 指令和数据均存放在内存中，计算机如何从时间和空间上区分它们是指令还是数据？
- 2) 什么是指令周期、机器周期和时钟周期？它们之间有何关系？
- 3) 什么是微指令？它和第 4 章谈到的指令有什么关系？
- 4) 什么是指令流水线？指令流水线相对于传统体系结构的优势是什么？

请读者在本章的学习过程中寻找答案，本章末尾会给出参考答案。

5.1 CPU 的功能和基本结构

5.1.1 CPU 的功能

中央处理器（CPU）由运算器和控制器组成。其中，控制器的功能是负责协调并控制计算机各部件执行程序的指令序列，包括取指令、分析指令和执行指令；运算器的功能是对数据进行加工。CPU 的具体功能包括：

- 1) 指令控制。完成取指令、分析指令和执行指令的操作，即程序的顺序控制。
- 2) 操作控制。一条指令的功能往往由若干操作信号的组合来实现。CPU 管理并产生由内存取出的每条指令的操作信号，把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。对数据进行算术和逻辑运算。
- 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。

5.1.2 CPU 的基本结构

在计算机系统中，中央处理器主要由运算器和控制器两大部分组成，如图 5.1 所示。

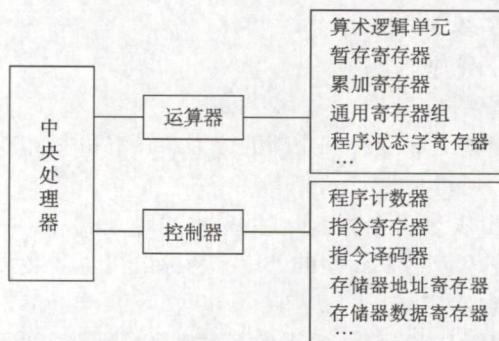


图 5.1 中央处理器的组成

1. 运算器

运算器接收从控制器送来的命令并执行相应的动作，对数据进行加工和处理。运算器是计算机对数据进行加工处理的中心，它主要由算术逻辑单元（ALU）、暂存寄存器、累加寄存器（ACC）、通用寄存器组、程序状态字寄存器（PSW）、移位器、计数器（CT）等组成。

- 1) 算术逻辑单元。主要功能是进行算术/逻辑运算。
- 2) 暂存寄存器。用于暂存从主存读来的数据，该数据不能存放在通用寄存器中，否则会破坏其原有内容。暂存寄存器对应用程序员是透明的。
- 3) 累加寄存器。它是一个通用寄存器，用于暂时存放 ALU 运算的结果信息，可以作为加法运算的一个输入端。
- 4) 通用寄存器组。如 AX、BX、CX、DX、SP 等，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。SP 是堆栈指针，用于指示栈顶的地址。
- 5) 程序状态字寄存器。保留由算术逻辑运算指令或测试指令的结果而建立的各种状态信息，如溢出标志（OF）、符号标志（SF）、零标志（ZF）、进位标志（CF）等。PSW 中的这些位参与并决定微操作的形成。
- 6) 移位器。对操作数或运算结果进行移位运算。
- 7) 计数器。控制乘除运算的操作步数。

2. 控制器

控制器是整个系统的指挥中枢，在控制器的控制下，运算器、存储器和输入/输出设备等功能部件构成一个有机的整体，根据指令的要求指挥全机协调工作。控制器的基本功能是执行指令，

每条指令的执行是由控制器发出的一组微操作实现的。

控制器有硬布线控制器和微程序控制器两种类型（见 5.4 节）。

控制器由程序计数器（PC）、指令寄存器（IR）、指令译码器、存储器地址寄存器（MAR）、存储器数据寄存器（MDR）、时序系统和微操作信号发生器等组成。

- 1) 程序计数器。用于指出欲执行指令在主存中的存放地址。CPU 根据 PC 的内容去主存中取指令。因程序中指令（通常）是顺序执行的，所以 PC 有自增功能。
- 2) 指令寄存器。用于保存当前正在执行的那条指令。
- 3) 指令译码器。仅对操作码字段进行译码，向控制器提供特定的操作信号。
- 4) 存储器地址寄存器。用于存放要访问的主存单元的地址。
- 5) 存储器数据寄存器。用于存放向主存写入的信息或从主存读出的信息。
- 6) 时序系统。用于产生各种时序信号，它们都由统一时钟（CLOCK）分频得到。
- 7) 微操作信号发生器。根据 IR 的内容（指令）、PSW 的内容（状态信息）及时序信号，产生控制整个计算机系统所需的各种控制信号，其结构有组合逻辑型和存储逻辑型两种。

控制器的工作原理是，根据指令操作码、指令的执行步骤（微命令序列）和条件信号来形成当前计算机各部件要用到的控制信号。计算机整机各硬件系统在这些控制信号的控制下协同运行，产生预期的执行结果。

注意：CPU 内部寄存器大致可分为两类：一类是用户可见的寄存器，可对这类寄存器编程，如通用寄存器组、程序状态字寄存器；另一类是用户不可见的寄存器，对用户是透明的，不可对这类寄存器编程，如存储器地址寄存器、存储器数据寄存器、指令寄存器。

5.1.3 本节习题精选

一、单项选择题

01. 下列部件不属于控制器的是（ ）。
 - A. 指令寄存器
 - B. 程序计数器
 - C. 程序状态字寄存器
 - D. 时序电路
02. 通用寄存器是（ ）。
 - A. 可存放指令的寄存器
 - B. 可存放程序状态字的寄存器
 - C. 本身具有计数逻辑与移位逻辑的寄存器
 - D. 可编程指定多种功能的寄存器
03. CPU 中保存当前正在执行指令的寄存器是（ ）。
 - A. 指令寄存器
 - B. 指令译码器
 - C. 数据寄存器
 - D. 地址寄存器
04. 在 CPU 中，跟踪后继指令地址的寄存器是（ ）。
 - A. 指令寄存器
 - B. 程序计数器
 - C. 地址寄存器
 - D. 状态寄存器
05. 条件转移指令执行时所依据的条件来自（ ）。
 - A. 指令寄存器
 - B. 标志寄存器
 - C. 程序计数器
 - D. 地址寄存器
06. 在所谓的 n 位 CPU 中， n 是指（ ）。
 - A. 地址总线线数
 - B. 数据总线线数
 - C. 控制总线线数
 - D. I/O 线数
07. 在 CPU 的寄存器中，（ ）对用户是透明的。
 - A. 程序计数器
 - B. 状态寄存器
 - C. 指令寄存器
 - D. 通用寄存器
08. 程序计数器（PC）属于（ ）。

关注公众号【乘龙考研】
一手更新 稳定有保障

- A. 运算器 B. 控制器 C. 存储器 D. ALU
09. 下面有关程序计数器 (PC) 的叙述中, 错误的是 ()。
- PC 中总是存放指令地址
 - PC 的值由 CPU 在执行指令过程中进行修改
 - 转移指令时, PC 的值总是修改为转移指令的目标地址
 - PC 的位数一般和存储器地址寄存器 (MAR) 的位数一样
10. 在一条无条件跳转指令的指令周期内, PC 的值被修改 () 次。
- 1
 - 2
 - 3
 - 无法确定
11. 程序计数器的位数取决于 ()。
- 存储器的容量
 - 机器字长
 - 指令字长
 - 都不对
12. 指令寄存器的位数取决于 ()。
- 存储器的容量
 - 机器字长
 - 指令字长
 - 存储字长
13. CPU 中通用寄存器的位数取决于 ()。
- 存储器的容量
 - 指令的长度
 - 机器字长
 - 都不对
14. CPU 中的通用寄存器, ()。
- 只能存放数据, 不能存放地址
 - 可以存放数据和地址
 - 既不能存放数据, 又不能存放地址
 - 可以存放数据和地址, 还可以替代指令寄存器
15. 在计算机系统中表征程序和机器运行状态的部件是 ()。
- 程序计数器
 - 累加寄存器
 - 中断寄存器
 - 程序状态字寄存器
16. 状态寄存器用来存放 ()。
- 算术运算结果
 - 逻辑运算结果
 - 运算类型
 - 算术、逻辑运算及测试指令的结果状态
17. 控制器的全部功能是 ()。
- 产生时序信号
 - 从主存中取出指令并完成指令操作码译码
 - 从主存中取出指令、分析指令并产生有关的操作控制信号
 - 都不对
18. 指令译码是指对 () 进行译码。
- 整条指令
 - 指令的操作码字段
 - 指令的地址码字段
 - 指令的地址
19. CPU 中不包括 ()。
- 存储器地址寄存器
 - 指令寄存器
 - 地址译码器
 - 程序计数器
20. 以下关于计算机系统的概念中, 正确的是 ()。
- CPU 不包括地址译码器
 - CPU 的程序计数器中存放的是操作数地址
 - CPU 中决定指令执行顺序的是程序计数器
 - CPU 的状态寄存器对用户是完全透明的
- I、III
 - III、IV
 - II、III、IV
 - I、III、IV

关注公众号【乘龙考研】
一手更新 稳定有保障

21. 间址周期结束时, CPU 内寄存器 MDR 中的内容为()。
 A. 指令 B. 操作数地址 C. 操作数 D. 无法确定
22. 【2010 统考真题】下列寄存器中, 汇编语言程序员可见的是()。
 A. 存储器地址寄存器 (MAR) B. 程序计数器 (PC)
 C. 存储器数据寄存器 (MDR) D. 指令寄存器 (IR)
23. 【2016 统考真题】某计算机的主存空间为 4GB, 字长为 32 位, 按字节编址, 采用 32 位字长指令字格式。若指令按字边界对齐存放, 则程序计数器 (PC) 和指令寄存器 (IR) 的位数至少分别是()。
 A. 30, 30 B. 30, 32 C. 32, 30 D. 32, 32

二、综合应用题

01. CPU 中有哪些专用寄存器?

5.1.4 答案与解析

一、单项选择题

01. C

控制器由程序计数器 PC、指令寄存器 IR、存储器地址寄存器 MAR、存储器数据寄存器 MDR、指令译码器、时序电路和微操作信号发生器组成。程序状态字 (PSW) 寄存器属于运算器的组成部分, PSW 包括两个部分: 一是状态标志, 如进位标志 C、结果为零标志 Z 等, 大多数指令的执行将会影响到这些标志位; 二是控制标志, 如中断标志、陷阱标志等。

02. D

存放指令的寄存器是指令寄存器, 因此选项 A 错。存放程序状态字的寄存器是程序状态字寄存器, 因此选项 B 错。通用寄存器本身并不一定具有计数和移位逻辑功能, 因此选项 C 错。

03. A

指令寄存器用于存放当前正在执行的指令。

04. B

程序计数器用于存放下一条指令在主存中的地址, 具有自增功能。

05. B

指令寄存器用于存放当前正在执行的指令; 程序计数器用于存放下一条指令的地址; 地址寄存器用于暂存指令或数据的地址; 程序状态字寄存器用于保存系统的运行状态。条件转移指令执行时, 需对标志寄存器的内容进行测试, 判断是否满足转移条件。

06. B

数据总线的位数与处理器的位数相同, 它表示 CPU 一次能处理的数据的位数, 即 CPU 的位数。

07. C

指令寄存器中存放当前执行的指令, 不需要用户的任何干预, 所以对用户是透明的。其他三种寄存器的内容可由程序员指定。

08. B

控制器是计算机中处理指令的部件, 包含程序计数器。

09. C

PC 中存放下一条要执行的指令的地址, 选项 A 正确。PC 的值会根据 CPU 在执行指令的过程中修改 (确切地说是在取指周期), 或自增, 或转移到程序的某处, 选项 B 正确。转移指令时, 需要判别转移是否成功, 若成功则 PC 修改为转移指令的目标地址, 否则下一条指令的地址仍然为 PC

关注公众号【乘龙考研】
一手更新 稳定有保障

自增后的地址，选项 C 错误。PC 与 MAR 的位数一样，选项 D 正确。

10. B

取指周期结束后，PC 值自动加 1；执行周期中，PC 值修改为要跳转到的地址，因此在这个指令周期内，PC 值被修改两次。

11. A

程序计数器的内容为指令在主存中的地址，所以程序计数器的位数与存储器地址的位数相等，而存储器地址取决于存储器的容量，可知选项 A 正确。

12. C

指令寄存器中保存当前正在执行的指令，所以其位数取决于指令字长。

13. C

通用寄存器用于存放操作数和各种地址信息等，其位数与机器字长相等，因此便于操作控制。

14. B

通用寄存器供用户自由编程，可以存放数据和地址。而指令寄存器是专门用于存放指令的专用寄存器，不能由通用寄存器代替。

15. D

程序状态字寄存器用于存放程序状态字，而程序状态字的各位表征程序和机器的运行状态，如含有进位标志 C、结果为零标志 Z 等。

16. D

程序状态字寄存器用于保留算术、逻辑运算及测试指令的结果状态。

17. C

控制器的功能是取指令、分析指令和执行指令，执行指令就是发出有关操作控制信号。

18. B

指令包括操作码字段和地址码字段，但指令译码器仅对操作码字段进行译码，借以确定指令的操作功能。

19. C

地址译码器是主存等存储器的组成部分，其作用是根据输入的地址码唯一选定一个存储单元，它不是 CPU 的组成部分。而 MAR、IR、PC 都是 CPU 的组成部分。

20. A

地址译码器位于存储器，I 正确；程序计数器中存放的是欲执行指令的地址，II 错误；程序计数器决定程序的执行顺序，III 正确；程序状态字寄存器对用户不透明，IV 错误。

21. B

间址周期的作用是取操作数的有效地址，因此间址周期结束后，MDR 中的内容为操作数地址。

22. B

汇编语言程序员可见的是程序计数器（PC），即汇编语言程序员通过汇编程序可以对某个寄存器进行访问。汇编程序员可以通过指定待执行指令的地址来设置 PC 的值，如转移指令、子程序调用指令等。而 IR、MAR、MDR 是 CPU 的内部工作寄存器，对程序员不可见。

23. B

程序计数器（PC）给出下一条指令字的访存地址（指令在内存中的地址），它取决于存储器的字数（ $4GB/32\text{ 位} = 2^{30}$ ），因此程序计数器（PC）的位数至少是 30 位；指令寄存器（IR）用于接收取得的指令，它取决于指令字长（32 位），因此指令寄存器（IR）的位数至少为 32 位。

二、综合应用题

01. 【解答】

CPU 中的专用寄存器有程序计数器 (PC)、指令寄存器 (IR)、存储器数据寄存器 (MDR)、存储器地址寄存器 (MAR) 和程序状态字寄存器 (PSW)。

5.2 指令执行过程

关注公众号【乘龙考研】
一手更新稳定有保障

5.2.1 指令周期

CPU 从主存中取出并执行一条指令的时间称为指令周期，不同指令的指令周期可能不同。指令周期常用若干机器周期来表示，一个机器周期又包含若干时钟周期（也称节拍或 T 周期，它是 CPU 操作的最基本单位）。每个指令周期内的机器周期数可以不等，每个机器周期内的节拍数也可以不等。图 5.2 反映了上述关系。图 5.2(a)为定长的机器周期，每个机器周期包含 4 个节拍 (T)；图 5.2(b)所示为不定长的机器周期，每个机器周期包含的节拍数可以为 4 个，也可以为 3 个。

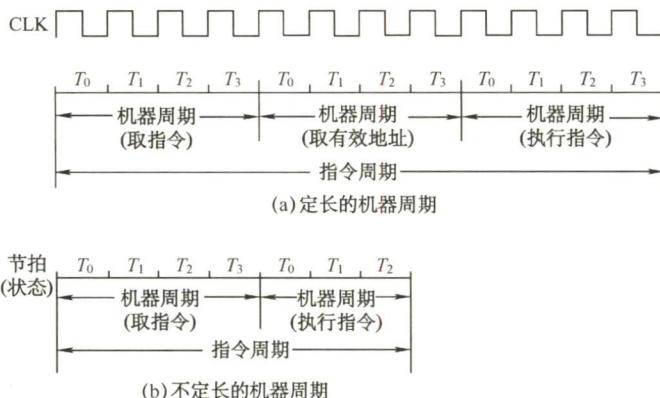


图 5.2 指令周期、机器周期、节拍和时钟周期的关系

对于无条件转移指令 JMP X，在执行时不需要访问主存，只包含取指阶段（包括取指和分析）和执行阶段，所以其指令周期仅包含取指周期和执行周期。

对于间接寻址的指令，为了取操作数，需要先访问一次主存，取出有效地址，然后访问主存，取出操作数，所以还需包括间址周期。间址周期介于取指周期和执行周期之间。

当 CPU 采用中断方式实现主机和 I/O 设备的信息交换时，CPU 在每条指令执行结束前，都要发中断查询信号，若有中断请求，则 CPU 进入中断响应阶段，又称中断周期。这样，一个完整的指令周期应包括取指、间址、执行和中断 4 个周期，如图 5.3 所示。



图 5.3 带有间址周期、中断周期的指令周期

上述 4 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，间址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。

为了区别不同的工作周期，在CPU内设置4个标志触发器FE、IND、EX和INT，它们分别对应取指、间址、执行和中断周期，并以“1”状态表示有效，分别由 $1 \rightarrow FE$ 、 $1 \rightarrow IND$ 、 $1 \rightarrow EX$ 和 $1 \rightarrow INT$ 这4个信号控制。

注意：中断周期中的进栈操作是将SP减1，这和传统意义上的进栈操作相反，原因是计算机的堆栈中都是向低地址增加，所以进栈操作是减1而不是加1。

5.2.2 指令周期的数据流

数据流是根据指令要求依次访问的数据序列。在指令执行的不同阶段，要求依次访问的数据序列是不同的。而且对于不同的指令，它们的数据流往往也是不同的。

1. 取指周期

取指周期的任务是根据PC中的内容从主存中取出指令代码并存放在IR中。

取指周期的数据流如图5.4所示。PC中存放的是指令的地址，根据此地址从内存单元中取出的是指令，并放在指令寄存器IR中，取指令的同时，PC加1。

取指周期的数据流向如下：

- 1) PC①MAR②地址总线③主存。
- 2) CU发出读命令④控制总线⑤主存。
- 3) 主存⑥数据总线⑦MDR⑧IR(存放指令)。
- 4) CU发出控制信号⑨PC内容加1。

关注公众号【乘龙考研】
一手更新 稳定有保障

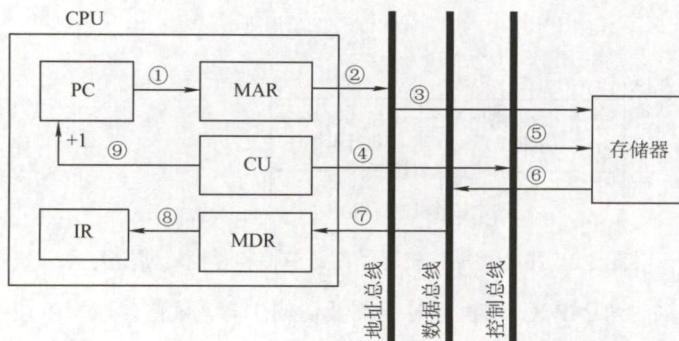


图5.4 取指周期的数据流

2. 间址周期

间址周期的任务是取操作数有效地址。以一次间址为例（见图5.5），将指令中的地址码送到MAR并送至地址总线，此后CU向存储器发读命令，以获取有效地址并存至MDR。

间址周期的数据流向如下：

- 1) Ad(IR) (或 MDR) ① MAR ② 地址总线 ③ 主存。
- 2) CU发出读命令④控制总线⑤主存。
- 3) 主存⑥数据总线⑦MDR(存放有效地址)。

其中，Ad(IR)表示取出IR中存放的指令字的地址字段。

3. 执行周期

执行周期的任务是取操作数，并根据IR中的指令字的操作码通过ALU操作产生执行结果。不同指令的执行周期操作不同，因此没有统一的数据流向。

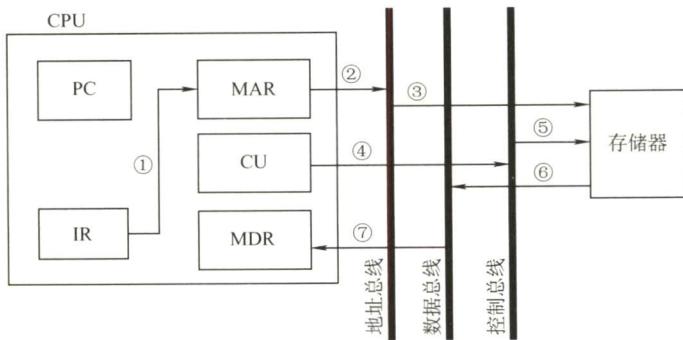


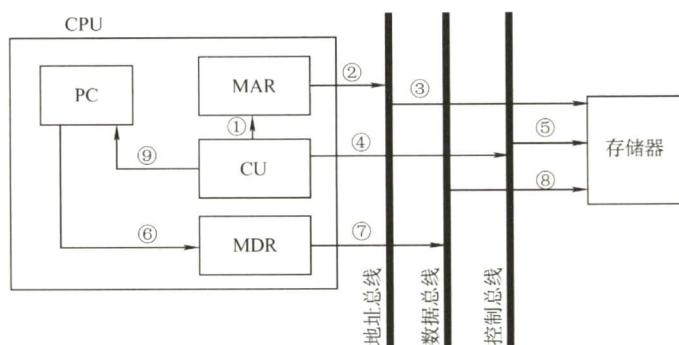
图 5.5 一次间址周期的数据流

4. 中断周期

中断周期的任务是处理中断请求。假设程序断点存入堆栈中，并用 SP 指示栈顶地址，而且进栈操作是先修改栈顶指针，后存入数据，数据流如图 5.6 所示。

中断周期的数据流向如下：

- 1) CU 控制将 SP 减 1，SP ① MAR ② 地址总线 ③ 主存。
- 2) CU 发出写命令 ④ 控制总线 ⑤ 主存。
- 3) PC ⑥ MDR ⑦ 数据总线 ⑧ 主存（程序断点存入主存）。
- 4) CU（中断服务程序的入口地址）⑨ PC。



关注公众号【乘龙考研】
一手更新 稳定有保障

图 5.6 中断周期的数据流

5.2.3 指令执行方案

一个指令周期通常要包括几个时间段（执行步骤），每个步骤完成指令的一部分功能，几个依次执行的步骤完成这条指令的全部功能。出于性能和硬件成本等考虑，可以选用 3 种不同的方案来安排指令的执行步骤。

1. 单指令周期

对所有指令都选用相同的执行时间来完成，称为单指令周期方案。此时每条指令都在一个时钟周期内完成，指令之间串行执行，即下一条指令只能在前一条指令执行结束后才能启动。因此，时钟周期取决于执行时间最长的指令的执行时间。对于那些本来可以在更短时间内完成的指令，要使用这个较长的周期来完成，会降低整个系统的运行速度。

2. 多指令周期

对不同类型的指令选用不同的执行步骤，称为多指令周期方案。指令之间串行执行，即下条

指令只能在前一指令执行完后才能启动。但可选用不同个数的时钟周期来完成不同指令的执行过程，指令需要几个周期就为其分配几个周期，不再要求所有指令占用相同的执行时间。

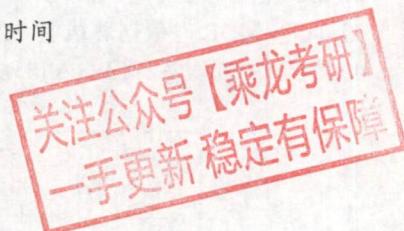
3. 流水线方案

指令之间可以并行执行的方案，称为流水线方案，其追求的目标是力争在每个时钟脉冲周期完成一条指令的执行过程（只在理想情况下才能达到该效果）。这种方案通过在每个时钟周期启动一条指令，尽量让多条指令同时运行，但各自处在不同的执行步骤中。

5.2.4 本节习题精选

一、单项选择题

01. 计算机工作的最长时间周期是（ ）。
 - A. 时钟周期
 - B. 指令周期
 - C. CPU 周期
 - D. 工作脉冲
02. 采用 DMA 方式传递数据时，每传送一个数据就要占用（ ）。
 - A. 指令周期
 - B. 时钟周期
 - C. 机器周期
 - D. 存取周期
03. 指令周期是指（ ）。
 - A. CPU 从主存取出一条指令的时间
 - B. CPU 执行一条指令的时间
 - C. CPU 从主存取出一条指令加上执行这条指令的时间
 - D. 时钟周期时间
04. 指令（ ）从主存中读出。
 - A. 总是根据程序计数器
 - B. 有时根据程序计数器，有时根据转移指令
 - C. 根据地址寄存器
 - D. 有时根据程序计数器，有时根据地址寄存器
05. 在一条无条件跳转指令的指令周期内，程序计数器（PC）的值被修改了（ ）次。
 - A. 1
 - B. 2
 - C. 3
 - D. 不能确定
06. 在取指操作后，程序计数器中存放的是（ ）。
 - A. 当前指令的地址
 - B. 程序中指令的数量
 - C. 已执行的指令数量
 - D. 下一条指令的地址
07. 以下叙述中，错误的是（ ）。
 - A. 指令周期的第一个操作是取指令
 - B. 为了进行取指操作，控制器需要得到相应的指令
 - C. 取指操作是控制器自动进行的
 - D. 指令执行时有些操作是相同或相似的
08. 指令周期由一个到几个机器周期组成，第一个机器周期是（ ）。
 - A. 从主存中取出指令字
 - B. 从主存中取出指令操作码
 - C. 从主存中取出指令地址码
 - D. 从主存中取出指令的地址
09. 由于 CPU 内部操作的速度较快，而 CPU 访问一次存储器的时间较长，因此机器周期通常由（ ）来确定。
 - A. 指令周期
 - B. 存取周期
 - C. 间址周期
 - D. 中断周期
10. 以下有关机器周期的叙述中，错误的是（ ）。
 - A. 通常把通过一次总线事务访问一次主存或 I/O 的时间定为一个机器周期



- B. 一个指令周期通常包含多个机器周期
 C. 不同的指令周期所包含的机器周期数可能不同
 D. 每个指令周期都包含一个中断响应机器周期
11. 下列说法中，合理的是（ ）。
 A. 执行各条指令的机器周期数相同，各机器周期的长度均匀
 B. 执行各条指令的机器周期数相同，各机器周期的长度可变
 C. 执行各条指令的机器周期数可变，各机器周期的长度均匀
 D. 执行各条指令的机器周期数可变，各机器周期的长度可变
12. 以下关于间址周期的描述中，正确的是（ ）。
 A. 所有指令的间址操作都是相同的
 B. 凡是存储器间接寻址的指令，它们的操作都是相同的
 C. 对于存储器间接寻址和寄存器间接寻址，它们的操作是不同的
 D. 都不对
13. CPU 响应中断的时间是（ ）。
 A. 一条指令执行结束 B. I/O 设备提出中断
 C. 取指周期结束 D. 指令周期结束
14. 以下叙述中，错误的是（ ）。
 A. 取指操作是控制器固有的功能，不需要在操作码控制下完成
 B. 所有指令的取指操作是相同的
 C. 在指令长度相同的情况下，所有指令的取指操作是相同的
 D. 中断周期是在指令执行完成后出现的
15. （ ）可区分存储单元中存放的是指令还是数据。
 A. 控制器 B. 运算器 C. 存储器 D. 数据通路
16. 下列说法中，正确的是（ ）。
 I. 指令字长等于机器字长的前提下，取指周期等于机器周期
 II. 指令字长等于存储字长的前提下，取指周期等于机器周期
 III. 指令字长和机器字长的长度没有任何关系
 IV. 为了硬件设计方便，指令字长都和存储字长一样大
 A. II、III B. II、III、IV C. I、III、IV D. I、IV
17. 【2009 统考真题】冯·诺依曼计算机中指令和数据均以二进制形式存放在存储器中，CPU 区分它们的依据是（ ）。
 A. 指令操作码的译码结果 B. 指令和数据的寻址方式
 C. 指令周期的不同阶段 D. 指令和数据所在的存储单元
18. 【2011 统考真题】假定不采用 Cache 和指令预取技术，且机器处于“开中断”状态，则在下列有关指令执行的叙述中，错误的是（ ）。
 A. 每个指令周期中 CPU 都至少访问内存一次
 B. 每个指令周期一定大于或等于一个 CPU 时钟周期
 C. 空操作指令的指令周期中任何寄存器的内容都不会被改变
 D. 当前程序在每条指令执行结束时都可能被外部中断打断



二、综合应用题

01. 指令和数据都存于存储器中，CPU 如何区分它们？

02. 中断周期的前后各是 CPU 的什么工作周期?

5.2.5 答案与解析

一、单项选择题

01. A

时钟周期是计算机操作的最小单位时间，由计算机的主频确定，是主频的倒数。工作脉冲是控制器的最小时间单位，起定时触发作用，一个时钟周期有一个工作脉冲。指令周期则可由多个 CPU 周期组成。CPU 周期，即机器周期，包含若干时钟周期。

02. D

CPU 从主存中每取出并执行一条指令所需的全部时间称为指令周期；时钟周期通常称为节拍或 T 周期，它是 CPU 操作的最基本单位；CPU 周期也称机器周期，一个机器周期包含若干时钟周期；存取周期是指存储器进行两次独立的存储器操作（连续两次读或写操作）所需的最小间隔时间。

03. C

指令周期是指 CPU 从主存取出一条指令加上执行这条指令的时间，而间址周期不是必需的。

04. A

CPU 根据程序计数器 PC 中的内容从主存取指令。读者可能会想到无条件转移指令或中断返回指令，认为不一定总是根据 PC 读出。实际上，当前指令正在执行时，PC 已经是下一条指令的地址。若遇到无条件转移指令，则只需简单地用跳转地址覆盖原 PC 的内容即可，最终的结果还是根据 PC 从主存读出。地址寄存器用来指出所取数据在主存中的地址。

05. B

首先在取指周期结束后，PC 值自动加 1；在执行周期中，PC 值修改为要跳转到的地址。综上，在一条无条件跳转指令的指令周期内，程序计数器（PC）的值被修改了 2 次。

06. D

在取指操作后，程序计数器中的内容将被修改为下一条指令的地址，而不是当前指令的地址。

07. B

取指操作是自动进行的，控制器不需要得到相应的指令。

08. A

指令周期的第一个机器周期是取指周期，即从主存中取出指令字。

09. B

存储器进行一次读或写操作所需的时间称为存储器的访问时间（或读/写时间），而连续启动两次独立的读或写操作（如连续的两次读操作）所需的最短时间称为存取周期。机器周期通常由存取周期确定。

10. D

在指令的执行周期完成后，处理器会判断是否出现中断请求，只有在出现中断请求时才会进入中断周期。

11. D

机器周期是指令执行中每步操作（如取指令、存储器读、存储器写等）所需要的时间，每个机器周期内的节拍数可以不等，因此其长度可变。因为各种指令的功能不同，所以各指令执行时所需的机器周期数是可变的。

12. C

指令的间址分为一次间址、两次间址和多次间址，因此它们的操作是不同的，A、B 错误。存储器间址通过形式地址访存，寄存器间址通过寄存器内容访存，C 正确。

13. A

中断周期用于响应中断，若有中断，则在指令的执行周期后进入中断周期。

14. B

不同长度的指令，其取指操作可能是不同的。例如，双字指令、三字指令与单字指令的取指操作是不同的。

15. A

存储器本身无法区分存储单元中存放的是指令还是数据。而在控制器的控制下，计算机在不同的阶段对存储器进行读/写操作时，取出的代码也就有不同的用处。而在取指阶段读出的二进制代码为指令，在执行阶段读出的二进制代码则可能为数据；运算器和数据通路显然不能区分。

16. A

指令字长一般都取存储字长的整数倍，若指令字长等于存储字长的2倍，则需要两次访存，取指周期等于机器周期的2倍；若指令字长等于存储字长，则取指周期等于机器周期，因此I错。根据I的分析可知，II正确。指令字长取决于操作码的长度、操作数地址的长度和操作数地址的个数，与机器字长没有必然的联系。但为了硬件设计方便，指令字长一般取字节或存储字长的整数倍，因此III正确。根据III的分析可知，指令字长一般取字节或存储字长的整数倍，而不一定都和存储字长一样大，因此IV错误。综上所述，II、III正确。

17. C

冯·诺依曼计算机根据指令周期的不同阶段来区分从存储器取出的是指令还是数据，取指周期取出的是指令，执行周期取出的是数据。

18. C

由于不采用指令预取技术，每个指令周期都需要取指令，而不采用Cache技术，因此每次取指令都至少要访问内存一次（当指令字长与存储字长相等且按边界对齐时），选项A正确。时钟周期是CPU的最短时间单位，每个指令周期一定大于或等于一个CPU时钟周期，选项B正确。即使是空操作指令，在取指操作后，PC也会自动加1，选项C错误。由于机器处于“开中断”状态，在每条指令执行结束时都可能被外部中断打断。

二、综合应用题

01. 【解答】

通常完成一条指令可分为取指阶段和执行阶段。在取指阶段通过访问存储器可将指令取出；在执行阶段通过访问存储器可以将操作数取出。因此，虽然指令和数据都以二进制代码形式存放在存储器中，但CPU可根据指令周期的不同阶段判断从存储器取出的二进制代码是指令还是数据。

02. 【解答】

中断周期之前是执行周期，之后是下一条指令的取指周期。

5.3 数据通路的功能和基本结构

5.3.1 数据通路的功能

数据在功能部件之间传送的路径称为数据通路，包括数据通路上流经的部件，如ALU、通用寄存器、状态寄存器、异常和中断处理逻辑等。数据通路描述了信息从什么地方开始，中间经过哪个寄存器或多路开关，最后传送到哪个寄存器，这些都需要加以控制。



数据通路由控制部件控制，控制部件根据每条指令功能的不同生成对数据通路的控制信号。数据通路的功能是实现 CPU 内部的运算器与寄存器及寄存器之间的数据交换。

5.3.2 数据通路的基本结构

数据通路的基本结构主要有以下几种：

- 1) CPU 内部单总线方式。将所有寄存器的输入端和输出端都连接到一条公共通路上，这种结构比较简单，但数据传输存在较多的冲突现象，性能较低。连接各部件的总线只有一条时，称为单总线结构；CPU 中有两条或更多的总线时，构成双总线结构或多总线结构。图 5.7 所示为 CPU 内部总线的数据通路和控制信号。
- 2) CPU 内部多总线方式。将所有寄存器的输入端和输出端都连接到多条公共通路上，相比之下单总线中一个时钟内只允许传一个数据，因而指令执行效率很低，因此采用多总线方式，同时在多个总线上传送不同的数据，提高效率。
- 3) 专用数据通路方式。根据指令执行过程中的数据和地址的流动方向安排连接线路，避免使用共享的总线，性能较高，但硬件量大。

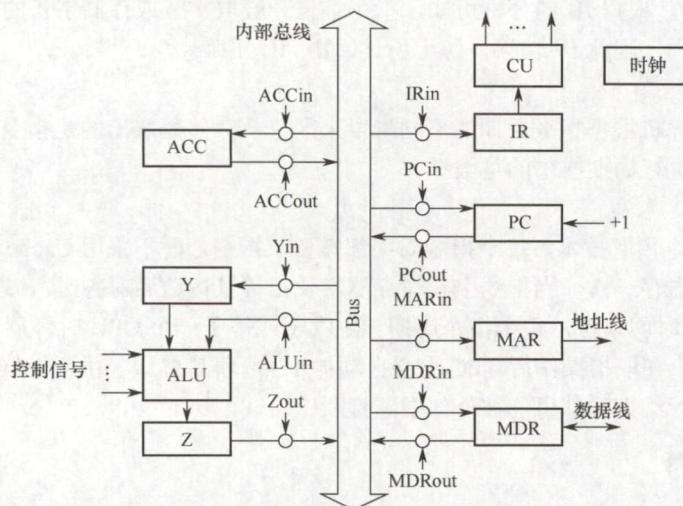


图 5.7 CPU 内部总线的数据通路和控制信号

在图 5.7 中，规定各部件用大写字母表示，字母加“in”表示该部件的允许输入控制信号；字母加“out”表示该部件的允许输出控制信号。

注意：内部总线是指同一部件，如 CPU 内部连接各寄存器及运算部件之间的总线；系统总线是指同一台计算机系统的各部件，如 CPU、内存和各类 I/O 接口间互相连接的总线。

1. 寄存器之间的数据传送

寄存器之间的数据传送可通过 CPU 内部总线完成。在图 5.7 中，某寄存器 AX 的输出和输入分别由 AXout 和 AXin 控制。现以 PC 寄存器为例，把 PC 内容送至 MAR，实现传送操作的流程及控制信号为

(PC) → MAR

PCout 和 MARin 有效，PC 内容 → MAR

2. 主存与 CPU 之间的数据传送

主存与 CPU 之间的数据传送也要借助 CPU 内部总线完成。现以 CPU 从主存读取指令为例

说明数据在数据通路中的传送过程。实现传送操作的流程及控制信号为

(PC) → MAR	PCout 和 MARin 有效, 现行指令地址 → MAR
1 → R	CU 发读命令
MEM (MAR) → MDR	MDRin 有效
(MDR) → IR	MDROUT 和 IRin 有效, 现行指令 → IR

3. 执行算术或逻辑运算

执行算术或逻辑操作时, 由于 ALU 本身是没有内部存储功能的组合电路, 因此如要执行加法运算, 相加的两个数必须在 ALU 的两个输入端同时有效。图 5.7 中的暂存器 Y 即用于该目的。先将一个操作数经 CPU 内部总线送入暂存器 Y 保存, Y 的内容在 ALU 的左输入端始终有效, 再将另一个操作数经总线直接送到 ALU 的右输入端。这样两个操作数都送入了 ALU, 运算结果暂存在暂存器 Z 中。

(MDR) → MAR	MDROUT 和 MARin 有效, 操作数有效地址 → MAR
1 → R	CU 发读命令
MEM (MAR) → MDR	操作数从存储器 → MDR
(MDR) → Y	MDROUT 和 Yin 有效, 操作数 → Y
(ACC) + (Y) → Z	ACCCout 和 ALUin 有效, CU 向 ALU 发加命令, 结果 → Z
(Z) → ACC	Zout 和 ACCin 有效, 结果 → ACC

数据通路结构直接影响 CPU 内各种信息的传送路径, 数据通路不同, 指令执行过程的微操作序列的安排也不同, 它关系着微操作信号形成部件的设计。

5.3.3 本节习题精选

一、单项选择题

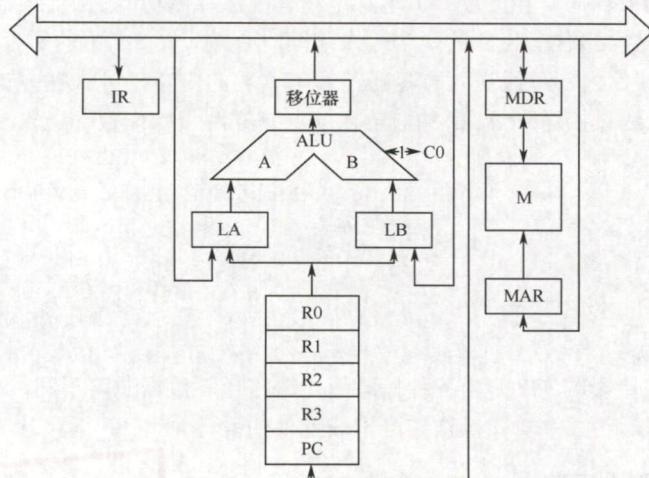
01. 下列不属于 CPU 数据通路结构的是()。
- A. 单总线结构
 - B. 多总线结构
 - C. 部件内总线结构
 - D. 专用数据通路结构
02. 在单总线的 CPU 中, ()。
- A. ALU 的两个输入端及输出端都可与总线相连
 - B. ALU 的两个输入端可与总线相连, 但输出端需通过暂存器与总线相连
 - C. ALU 的一个输入端可与总线相连, 其输出端也可与总线相连
 - D. ALU 只能有一个输入端可与总线相连, 另一输入端需通过暂存器与总线相连
03. 采用 CPU 内部总线的数据通路与不采用 CPU 内部总线的数据通路相比, ()。
- A. 前者性能较高
 - B. 后者的数据冲突问题较严重
 - C. 前者的硬件量大, 实现难度高
 - D. 以上说法都不对
04. CPU 的读/写控制信号的作用是()。
- A. 决定数据总线上的数据流方向
 - B. 控制存储器操作的读/写类型
 - C. 控制流入、流出存储器信息的方向
 - D. 以上都是
05. 【2016 统考真题】单周期处理器中所有指令的指令周期为一个时钟周期。下列关于单周期处理器的叙述中, 错误的是()。
- A. 可以采用单总线结构数据通路
 - B. 处理器时钟频率较低
 - C. 在指令执行过程中控制信号不变
 - D. 每条指令的 CPI 为 1
06. 【2021 统考真题】下列关于数据通路的叙述中, 错误的是()。
- A. 数据通路包含 ALU 等组合逻辑(操作)元件
 - B. 数据通路包含寄存器等时序逻辑(状态)元件

关注公众号【乘龙考研】
一手更新 稳定有保障

- C. 数据通路不包含用于异常事件检测及响应的电路
- D. 数据通路中的数据流动路径由控制信号进行控制

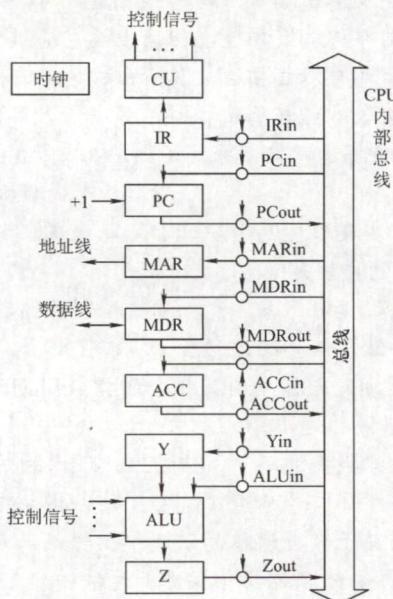
二、综合应用题

01. 某计算机的数据通路结构如下图所示，写出实现 ADD R1, (R2) 的微操作序列（取指令及确定后继指令地址）。

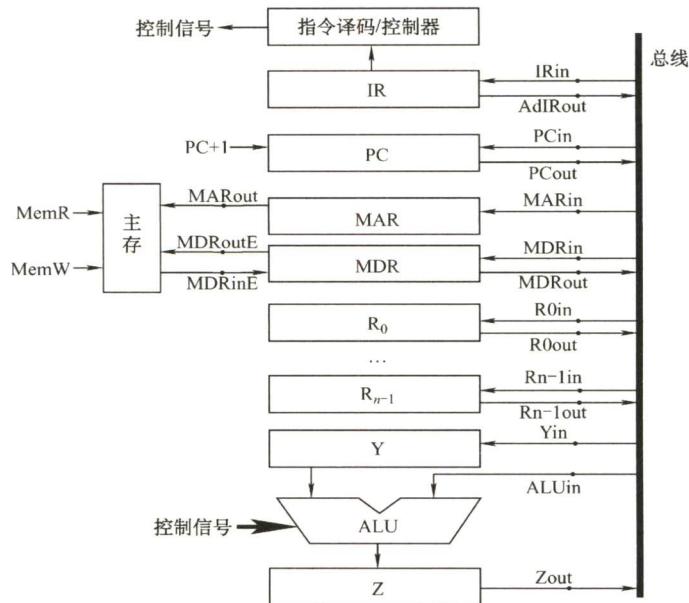


02. 设 CPU 内部结构如下图所示，此外还设有 B、C、D、E、H、L 六个寄存器（图中未画出），它们各自的输入端和输出端都与内部总线相通，并分别受控制信号控制（如 Bin 受寄存器 B 的输入控制；Bout 受寄存器 B 的输出控制），假设 ALU 的结果直接送入 Z 寄存器。要求从取指令开始，写出完成下列指令的微操作序列及所需的控制信号。

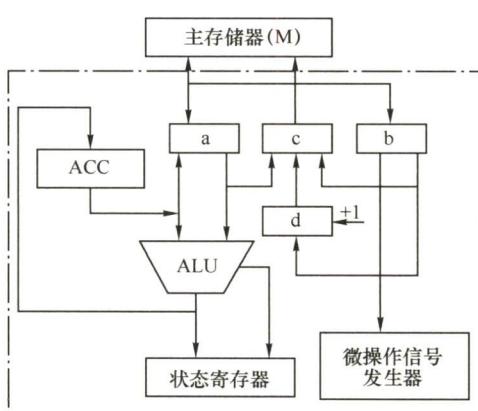
ADD B, C	$(B) + (C) \rightarrow B$
SUB ACC, H	$(ACC) - (H) \rightarrow ACC$



03. 设有如下图所示的单总线结构，分析指令 ADD (R0), R1 的指令流程和控制信号。



04. 下图是一个简化的 CPU 与主存连接结构示意图（图中省略了所有的多路选择器）。其中有一个累加寄存器 (ACC)、一个状态数据寄存器和其他 4 个寄存器：主存地址寄存器 (MAR)、主存数据寄存器 (MDR)、程序寄存器 (PC) 和指令寄存器 (IR)，各部件及其之间的连线表示数据通路，箭头表示信息传递方向。

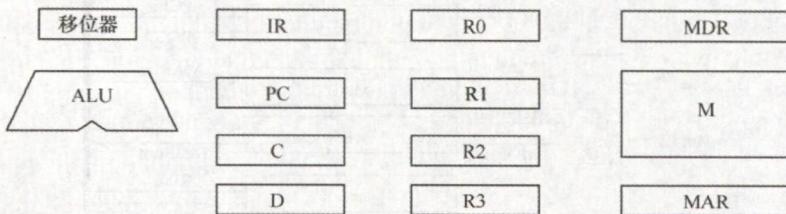


要求：

- 1) 请写出图中 a、b、c、d 四个寄存器的名称。
- 2) 简述图中取指令的数据通路。
- 3) 简述数据在运算器和主存之间进行存/取访问的数据通路（假设地址已在 MAR 中）。
- 4) 简述完成指令 LDA X 的数据通路（X 为主存地址，LDA 的功能为(X)→ACC）。
- 5) 简述完成指令 ADD Y 的数据通路（Y 为主存地址，ADD 的功能为(ACC) + (Y)→ACC）。

6) 简述完成指令 STA Z 的数据通路 (Z 为主存地址, STA 的功能为 $(ACC) \rightarrow Z$)。

05. 某机主要功能部件如下图所示, 其中 M 为主存, MDR 为主存数据寄存器, MAR 为主存地址寄存器, IR 为指令寄存器, PC 为程序计数器 (并假设当前指令地址在 PC 中), R0~R3 为通用寄存器, C、D 为暂存器。

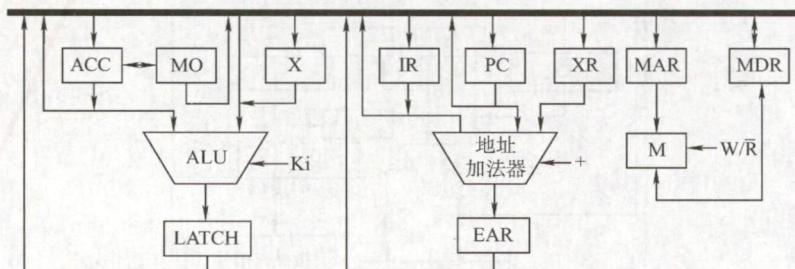


1) 请补充各部件之间的主要连接线 (总线自己画), 并注明数据流动方向。

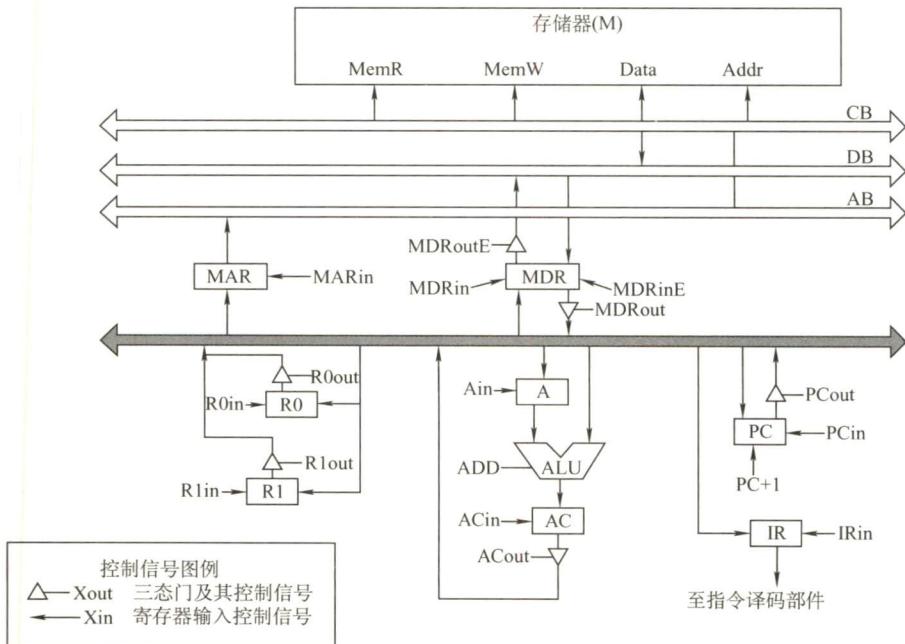
2) 画出 “ADD (R1), (R2) +” 指令周期流程图 “该指令的含义是进行求和运算, 源操作数地址在 R1 中, 目标操作数寻址方式为自增型寄存器间接寻址方式 (先取地址后加 1), 并将相加结果写回 R2 寄存器。”

06. 已知单总线计算机结构如下图所示, 其中 M 为主存, XR 为变址寄存器, EAR 为有效地址寄存器, LATCH 为暂存器。假设指令地址已存在于 PC 中, 请给出 ADD X, D 指令周期信息流程和相应的控制信号。说明:

- 1) ADD X, D 指令字中, X 为变址寄存器 XR, D 为形式地址。
- 2) 寄存器的输入/输出均采用控制信号控制, 如 PC_i 表示 PC 的输入控制信号, MDR_o 表示 MDR 的输出控制信号。
- 3) 凡需要经过总线的传送, 都需要注明, 如 $(PC) \rightarrow MAR$, 相应的控制信号为 PC_o 和 MAR_i 。



07. 【2009 统考真题】某计算机字长 16 位, 采用 16 位定长指令字结构, 部分数据通路结构如下图所示。图中所有控制信号为 1 时表示有效, 为 0 时表示无效。例如, 控制信号 MDRinE 为 1 表示允许数据从 DB 打入 MDR, MDRin 为 1 表示允许数据从总线打入 MDR。假设 MAR 的输出一直处于使能状态。加法指令 “ADD (R1), R0” 的功能为 $(R0) + ((R1)) \rightarrow (R1)$, 即将 R0 中的数据与 R1 的内容所指主存单元的数据相加, 并将结果送入 R1 的内容所指主存单元中保存。



下表给出了上述指令取指和译码阶段每个节拍（时钟周期）的功能和有效控制信号，请按表中描述方式用表格列出指令执行阶段每个节拍的功能和有效控制信号。

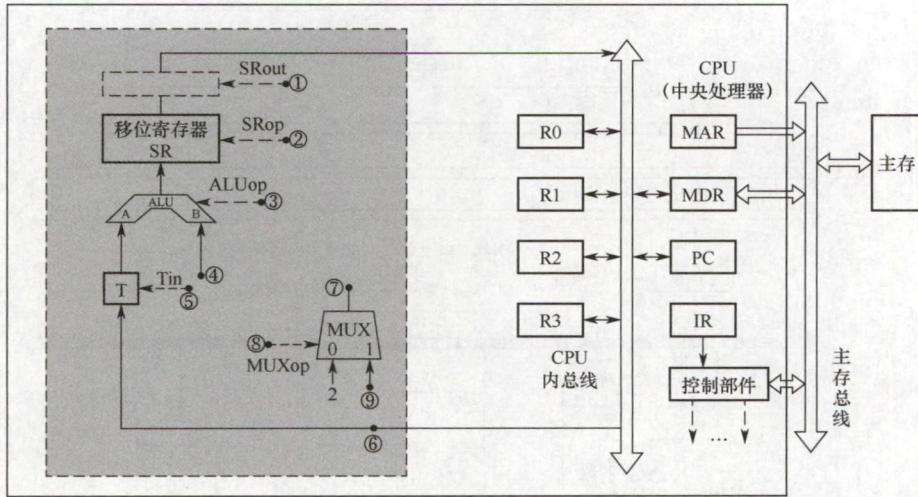
时 钟	功 能	有效控制信号
C1	MAR←(PC)	PCout, MARin
C2	MDR←M(MAR) PC←(PC) + 1	MemR, MDRinE, PC + 1
C3	IR←(MDR)	MDRout, IRin
C4	指令译码	无

08. 【2015统考真题】某16位计算机的主存按字节编码，存取单位为16位；采用16位定长指令字格式；CPU采用单总线结构，主要部分如下图所示。图中R0~R3为通用寄存器；T为暂存器；SR为移位寄存器，可实现直送（mov）、左移一位（left）和右移一位（right）三种操作，控制信号为SRop，SR的输出由信号SRout控制；ALU可实现直送A（mova）、A加B（add）、A减B（sub）、A与B（and）、A或B（or）、非A（not）、A加1（inc）七种操作，控制信号为ALUop。

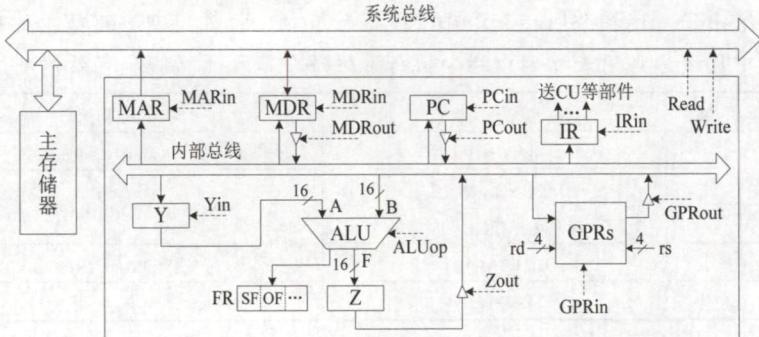
回答下列问题：

- 1) 图中哪些寄存器是程序员可见的？为何要设置暂存器 T？
- 2) 控制信号 ALUop 和 SRop 的位数至少各是多少？
- 3) 控制信号 SRout 所控制部件的名称或作用是什么？
- 4) 端点①~⑨中，哪些端点须连接到控制部件的输出端？
- 5) 为完善单总线数据通路，需要在端点①~⑨中相应的端点之间添加必要的连线。写出连线的起点和终点，以正确表示数据的流动方向。
- 6) 为什么二路选择器 MUX 的一个输入端是 2?

关注公众号【乘龙考研】
一手更新 稳定有保障



09. 【2022统考真题】某CPU中部分数据通路如下图所示，其中，GPRs为通用寄存器组；FR为标志寄存器，用于存放ALU产生的标志信息；带箭头虚线表示控制信号，如控制信号Read、Write分别表示主存读、主存写，MDRin表示内部总线上数据写入MDR，MDRout表示MDR的内容送内部总线。



请回答下列问题。

- 1) 设 ALU 的输入端 A、B 及输出端 F 的最高位分别为 A_{15} 、 B_{15} 及 F_{15} ，FR 中的符号标志和溢出标志分别为 SF 和 OF，则 SF 的逻辑表达式是什么？A 加 B、A 减 B 时 OF 的逻辑表达式分别是什么？要求逻辑表达式的输入变量为 A_{15} 、 B_{15} 及 F_{15} 。
- 2) 为什么要设置暂存器 Y 和 Z？
- 3) 若 GPRs 的输入端 rs、rd 分别为所读、写的通用寄存器的编号，则 GPRs 中最多有多少个通用寄存器？rs 和 rd 来自图中的哪个寄存器？已知 GPRs 内部有一个地址译码器和一个多路选择器，rd 应连接地址译码器还是多路选择器？
- 4) 取指令阶段（不考虑 PC 增量操作）的控制信号序列是什么？若从发出主存读命令到主存读出数据并传送到 MDR 共需 5 个时钟周期，则取指令阶段至少需要几个时钟周期？
- 5) 图中控制信号由什么部件产生？图中哪些寄存器的输出信号会连到该部件的输入端？

5.3.4 答案与解析

一、单项选择题

01. C

对 CPU 而言，数据通路的基本结构分为总线结构和专用数据通路结构，其中总线结构又分

为单总线结构、双总线结构、多总线结构。

02. D

由于 ALU 是一个组合逻辑电路，因此其运算过程中必须保持两个输入端的内容不变。又由于 CPU 内部采用单总线结构，因此为了得到两个不同的操作数，ALU 的一个输入端与总线相连，另一个输入端需通过一个寄存器与总线相连。此外，ALU 的输出端也不能直接与内部总线相连，否则其输出又会通过总线反馈到输入端，影响运算结果，因此输出端需通过一个暂存器（用来暂存结果的寄存器）与总线相连。

03. D

采用 CPU 内部总线方式的数据通路的特点：结构简单，实现容易，性能较低，存在较多的冲突现象；不采用 CPU 内部总线方式的数据通路的特点：结构复杂，硬件量大，不易实现，性能高，基本不存在数据冲突现象。

04. D

读/写控制信号线决定了是从存储器读还是向存储器写，显然 A、B、C 选项都正确。

05. A

单周期处理器是指所有指令的指令周期为一个时钟周期的处理器，选项 D 正确。因为每条指令的 CPI 为 1，要考虑比较慢的指令，所以处理器的时钟频率较低，选项 B 正确。单总线数据通路将所有寄存器的输入输出端都连接在一条公共通路上，一个时钟内只允许一次操作，无法完成指令的所有操作，选项 A 错误。控制信号是 CU 根据指令操作码发出的信号，对于单周期处理器来说，每条指令的执行只有一个时钟周期，而在一个时钟周期内控制信号并不会变化；若是多周期处理器，则指令的执行需要多个时钟周期，在每个时钟周期控制器会发出不同信号，选项 C 正确。

06. C

指令执行过程中数据所经过的路径，包括路径上的部件，称为数据通路。ALU、通用寄存器、状态寄存器、Cache、MMU、浮点运算逻辑、异常和中断处理逻辑等，都是指令执行过程中数据流经的部件，都属于数据通路的一部分。数据通路中的数据流动路径由控制部件控制，控制部件根据每条指令功能的不同，生成对数据通路的控制信号。选项 C 错误。

二、综合应用题

01. 【解答】

实现 ADD R1, (R2) 的微操作序列如下：

微操作	控制信号
(PC) → MAR	PC → BUS, BUS → MAR
M → MDR	READ
(PC) + 1 → PC	+1
(MDR) → IR	MDR → BUS, BUS → IR
(R1) → LA	R1 → LA
(R2) → MAR	R2 → BUS, BUS → MAR
M → MDR	READ
(MDR) → LB	MDR → BUS, BUS → LB
(LA) + (LB) → R1	+ , 移位器 → BUS, BUS → R1

关注公众号【乘龙考研】
一手更新 稳定有保障

02. 【解答】

两条指令的微操作序列如下。

(1) ADD B, C 指令。

微操作	控制信号
(PC) → MAR	PCout, MARin
(PC) + 1 → PC	+1
M(MAR) → MDR → IR	MDRout, IRin
(B) → Y	Bout, Yin
(Y) + (C) → Z	Cout, ALUin, “+”
(Z) → B	Zout, Bin

(2) SUB ACC, H 指令。

微操作	控制信号
(PC) → MAR	PCout, MARin
(PC) + 1 → PC	+1
M(MAR) → MDR → IR	MDRout, IRin
(ACC) → Y	ACcout, Yin
(Y) - (H) → Z	Hout, ALUin, “-”
(Z) → ACC	Zout, ACCin

注：Y 是与 ALU 的一个输入端相连接的暂存器。

03. 【解答】

指令 ADD (R0), R1 的功能是把 R0 的内容作为地址送到主存中取得一个操作数，再与 R1 中的内容相加，最后将结果送回主存，即实现 $(R0) + (R1) \rightarrow (R0)$ 。其流程和控制信号如下。

1) 取指周期：公共操作。

时序	微操作	有效控制信号	具体功能
1	(PC) → MAR	PCout, MARin	将 PC 经内部总线送至 MAR
2	M(MAR) → MDR, Read	MemR, MARout, MDRinE	主存通过数据总线将 MAR 所指单元的内容送至 MDR
3	(MDR) → IR	MDRout, IRin	将 MDR 的内容送至 IR
4	指令译码	-	操作字开始控制 CU
5	(PC) + 1 → PC	-	当 PC + 1 有效时，使 PC 内容加 1

2) 间址周期：完成取数操作，被加数在主存中，加数已经放在寄存器 R1 中。

时序	微操作	有效控制信号	具体功能
1	(R0) → MAR	R0out, MARin	将 R0 中的地址（形式地址）送至存储器地址寄存器
2	M(MAR) → MDR	MemR, MARout, MDRinE	主存通过数据总线将 MAR 所指单元的内容（有效地址）送至 MDR 中
3	(MDR) → Y	MDRout, Yin	将 MDR 中数据通过数据总线送至 Y

3) 执行周期：完成加法运算，并将结果返回主存。

时序	微操作	有效控制信号	具体功能
1	(R1) + (Y) → Z	R1out, ALUin, CU 向 ALU 发 ADD 控制信号	R1 的内容和 Y 的内容相加，结果送至寄存器 Z
2	(Z) → MDR	Zout, MDRin	将运算结果送至 MDR
3	(MDR) → M(MAR)	MemW, MDRoutE, MARout	向主存写入数据

04. 【解答】

1) b 单向连接微控制器, 由微控制器的作用可以推出 b 是 IR; a 和 c 直接连接主存, 只可能是 MDR 和 MAR, c 到主存是单向连接, a 和主存双向连接, 根据指令执行的特点, MAR 只单向给主存传送地址, 而 MDR 既存放从主存中取出的数据, 又存放将要写入主存的数据, 因此 c 为 MAR, a 为 MDR。d 具有自动加 1 的功能, 且单向连接 MAR, 为 PC。因此, a 为 MDR, b 为 IR, c 为 MAR, d 为 PC。

2) 将指令地址从 PC 送入 MAR, 在相关的控制下从主存中取出指令送至 MDR, 然后将 MDR 中的指令送至 IR, 最后流向微控制器。取指令的数据通路为

$$\text{PC} \rightarrow \text{MAR} \rightarrow \text{主存} \rightarrow \text{MDR} \rightarrow \text{IR}$$

3) 根据 MAR 中的地址去主存取数据, 将取出的数据送至 MDR, 然后将 MDR 中的数据送至 ALU 进行运算, 运算的结果送至 ACC。存储器读的数据通路为

$$\text{MAR} \text{ (先置数据地址)}, \text{ 主存 } M \rightarrow \text{MDR} \rightarrow \text{ALU} \rightarrow \text{ACC}$$

将 ACC 中的结果送至 MDR, 再将 MDR 中的数据写入主存。存储器写的数据通路为

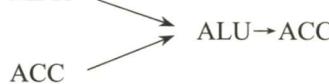
$$\text{MAR} \text{ (先置数据地址)}, \text{ ACC} \rightarrow \text{MDR} \rightarrow \text{主存 } M$$

4) 指令 LDA X 的数据通路为

$$X \rightarrow \text{MAR} \rightarrow \text{主存} \rightarrow \text{MDR} \rightarrow \text{ALU} \rightarrow \text{ACC}$$

5) 指令 ADD Y 的数据通路为

$$Y \rightarrow \text{MAR} \rightarrow \text{主存} \rightarrow \text{MDR}$$

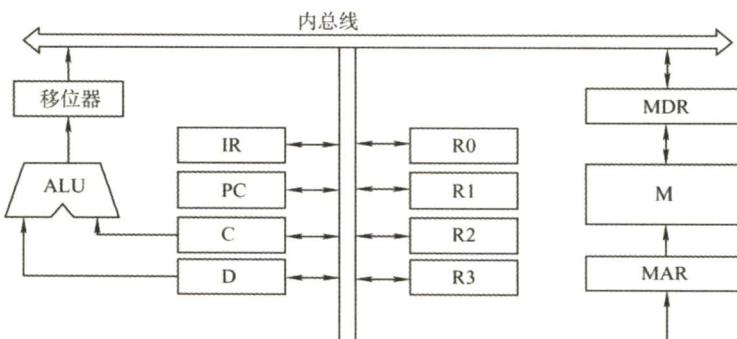


6) 指令 STA Z 的数据通路为 (ACC 中的数据需放在主存中)

$$Z \rightarrow \text{MAR}, \text{ ACC} \rightarrow \text{MDR} \rightarrow \text{主存}$$

05. 【解答】

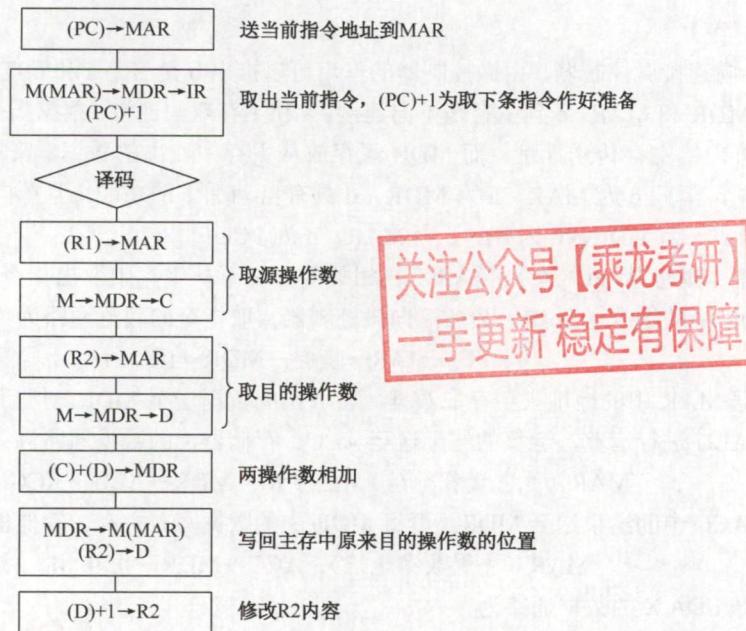
1) 各功能部件的连接关系, 以及数据通路如下图所示。



2) 分析过程如下:

- 取指令地址送到 IR 并译码。
- 取源操作数和目的操作数。
- 将源操作数和目的操作数相加送到 MAR, 随之送到以前目的操作数所在内存的地址。
- 将寄存器 R2 的内容加 1。

取指周期流程如下图所示。



06. 【解答】

ADD X, D 指令周期信息流程和相应的控制信号见下表。

周 期	微 操 作	有效控制信号
取指周期	(PC) → MAR	PC _o , MAR _i
	M(MAR) → MDR	MAR _o , R/W, MDR _i
	(PC) + 1 → PC	+1
	(MDR) → IR	MDR _o , IR _i
执行周期	(XR) + Ad(IR) → EAR	XR _o , IR _o , +, EAR _i
	(EAR) → MAR	EAR _o , MAR _i
	M(MAR) → MDR	MAR _o , R/W, MDR _i
	(MDR) → X	MDR _o , X _i
	(ACC) + (X) → LATCH	ACC _o , X _o , K _i = +, LATCH _i
	(LATCH) → ACC	LATCH _o , ACC _i

注：题目中的 D 即为 Ad(IR)。

07. 【解答】

题干已给出取值和译码阶段每个节拍的功能和有效控制信号，我们应以弄清楚取指阶段中数据通路的信息流动为突破口，读懂每个节拍的功能和有效控制信号，然后应用到解题思路中，包括划分执行步骤、确定完成的功能、需要的控制信号。

先分析题干中提供的示例（本部分解题时不做要求）：

取指令的功能是根据 PC 的内容所指的主存地址，取出指令代码，经过 MDR，最终送至 IR。这部分和后面的指令执行阶段的取操作数、存运算结果的方法是相通的。

C1: (PC) → MAR

在读写存储器前，必须先将地址（这里为(PC)）送至 MAR。

C2: M(MAR) → MDR, (PC) + 1 → PC

读写的数据必须经过 MDR，指令取出后 PC 自增 1。

C3: (MDR)→IR

然后将读到的 MDR 中的指令代码送至 IR 进行后续操作。

指令“ADD (R1), R0”的操作数一个在主存中，一个在寄存器中，运算结果在主存中。根据指令功能，要读出 R1 的内容所指的主存单元，必须先将 R1 的内容送至 MAR，即(R1)→MAR。而读出的数据必须经过 MDR，即 M(MAR)→MDR。

因此，将 R1 的内容所指的主存单元的数据读出到 MDR 的节拍安排如下：

C5: (R1)→MAR

C6: M(MAR)→MDR

ALU 一端是寄存器 A，MDR 或 R0 中必须有一个先写入 A 中，如 MDR。

C7: (MDR)→A

然后执行加法操作，并将结果送入寄存器 AC。

C8: (A) + (R0)→AC

之后将加法结果写回到 R1 的内容所指的主存单元，注意 MAR 中的内容没有改变。

C9: (AC)→MDR

C10: (MDR)→M(MAR)

有效控制信号的安排并不难，只需看数据是流入还是流出，如流入寄存器 X 就是 Xin，流出寄存器 X 就是 Xout。还需注意其他特殊控制信号，如 PC + 1、Add 等。

于是得到参考答案如下表所示。

时 钟	功 能	有效控制信号
C5	MAR←(R1)	R1out, MARin
C6	MDR←M(MAR)	MemR, MDRinE
C7	A←(MDR)	MDRout, Ain
C8	AC←(A) + (R0)	R0out, Add, ACin
C9	MDR←(AC)	ACout, MDRin
C10	M(MAR)←(MDR)	MDRoutE, MemW

本题答案不唯一，若在 C6 执行 M(MAR)→MDR 的同时，完成(R0)→A（即选择将(R0)写入 A），并不会发生总线冲突，这种方案可节省 1 个节拍，见下表。

时 钟	功 能	有效控制信号
C5	MAR←(R1)	R1out, MARin
C6	MDR←M(MAR), A←(R0)	MemR, MDRinE, R0out, Ain
C7	AC←(MDR) + (A)	MDRout, Add, ACin
C8	MDR←(AC)	ACout, MDRin
C9	M(MAR)←(MDR)	MDRoutE, MemW

08. 【解答】

- 1) 程序员可见寄存器为通用寄存器 (R0~R3) 和 PC。因为采用了单总线结构，因此若无暂存器 T，则 ALU 的 A、B 端口会同时获得两个相同的数据，使数据通路不能正常工作。
- 2) ALU 共有 7 种操作，其操作控制信号 ALUop 至少需要 3 位；移位寄存器有 3 种操作，其操作控制信号 SRop 至少需要 2 位。
- 3) 信号 SRout 所控制的部件是一个三态门，用于控制移位器与总线之间数据通路的连接与

断开。

- 4) 端口①、②、③、⑤、⑧须连接到控制部件输出端。
- 5) 连线 1, ⑥→⑨; 连线 2, ⑦→④。
- 6) 因为每条指令的长度为 16 位, 按字节编址, 所以每条指令占用 2 个内存单元, 顺序执行时, 下条指令地址为 $(PC) + 2$ 。MUX 的一个输入端为 2, 可便于执行 $(PC) + 2$ 操作。

09. 【解析】

- 1) 符号标志 SF 表示运算结果的正负性, 因此 $SF = F_{15}$ 。

对于加法运算 $A + B \rightarrow F$, 若 A、B 为负, 且 F 为正, 则说明发生溢出; 或者, 若 A、B 为正, 且 F 为负, 也说明发生溢出。因此, 加运算时, 溢出标志 $OF = \overline{A_{15}} \cdot \overline{B_{15}} \cdot F_{15} + A_{15} \cdot B_{15} \cdot \overline{F_{15}}$ 。

对于减法运算 $A - B \rightarrow F$, 若 A 为负、B 为正, 且 F 为正, 则说明发生溢出; 或者, 若 A 为正、B 为负, 且 F 为负, 也说明发生溢出。因此, 减运算时, 溢出标志 $OF = \overline{A_{15}} \cdot B_{15} \cdot F_{15} + A_{15} \cdot \overline{B_{15}} \cdot \overline{F_{15}}$ 。

- 2) 因为在单总线结构中, 每一时刻总线上只有一个数据有效, 而 ALU 有两个输入端和一个输出端。因此, 当 ALU 运算时, 需要先用暂存器 Y 缓存其中一个输入端的数据, 再通过总线传送另一个输入端的数据。与此同时, ALU 的输出端产生运算结果, 但由于总线正被占用, 因此需要暂存器 Z, 以缓存 ALU 的输出端数据。
- 3) 由图可知, rs 和 rd 都是 4bit, 因此 GPRs 中最多有 $2^4 = 16$ 个通用寄存器; rs 和 rd 来自指令寄存器 IR; rd 表示寄存器编号, 应连接地址译码器。
- 4) 取指阶段需要根据程序计数器 PC 取出主存中的指令, 并将指令写入指令寄存器 IR 中。控制信号序列如下:

- | | |
|---------------|------------------------|
| ①PCout, MARin | // 将指令的地址写入 MAR |
| ②Read | // 读主存, 并将读出的数据写入 MDR |
| ③MDRout, IRin | // 将 MDR 的内容写入指令寄存器 IR |

步骤①需要 1 个时钟周期, 步骤②需要 5 个时钟周期, 步骤③需要 1 个时钟周期, 因此取指令阶段至少需要 7 个时钟周期。

- 5) 图中控制信号由控制部件 (CU) 产生。指令寄存器 IR 和标志寄存器 FR 的输出信号会连到控制部件的输入端。

5.4 控制器的功能和工作原理

5.4.1 控制器的结构和功能

从图 5.8 可以看到计算机硬件系统的五大功能部件及其连接关系。它们通过数据总线、地址总线和控制总线连接在一起, 其中点画线框内的是控制器部件。

现对其主要连接关系简单说明如下:

- 1) 运算器部件通过数据总线与内存储器、输入设备和输出设备传送数据。
- 2) 输入设备和输出设备通过接口电路与总线相连接。
- 3) 内存储器、输入设备和输出设备从地址总线接收地址信息, 从控制总线得到控制信号, 通过数据总线与其他部件传送数据。
- 4) 控制器部件从数据总线接收指令信息, 从运算器部件接收指令转移地址, 送出指令地址到地址总线, 还要向系统中的部件提供它们运行所需要的控制信号。

关注公众号【乘龙考研】
一手更新稳定有保障

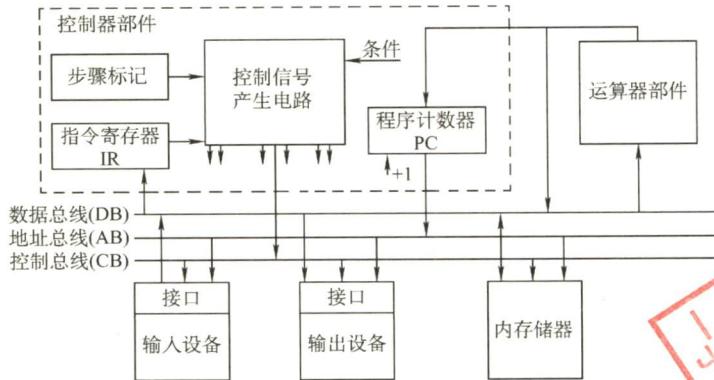


图 5.8 计算机硬件系统和控制器部件的组成

控制器是计算机系统的指挥中心，控制器的主要功能有：

- 1) 从主存中取出一条指令，并指出下一条指令在主存中的位置。
- 2) 对指令进行译码或测试，产生相应的操作控制信号，以便启动规定的动作。
- 3) 指挥并控制 CPU、主存、输入和输出设备之间的数据流动方向。

根据控制器产生微操作控制信号的方式的不同，控制器可分为硬布线控制器和微程序控制器，两类控制器中的 PC 和 IR 是相同的，但确定和表示指令执行步骤的办法以及给出控制各部件运行所需要的控制信号的方案是不同的。

5.4.2 硬布线控制器

硬布线控制器的基本原理是根据指令的要求、当前的时序及外部和内部的状态，按时间的顺序发送一系列微操作控制信号。它由复杂的组合逻辑门电路和一些触发器构成，因此又称组合逻辑控制器。

1. 硬布线控制单元图

指令的操作码是决定控制单元发出不同操作命令（控制信号）的关键。为了简化控制单元（CU）的逻辑，将指令的操作码译码和节拍发生器从 CU 分离出来，便可得到简化的控制单元图，如图 5.9 所示。

CU 的输入信号来源如下：

- 1) 经指令译码器译码产生的指令信息。现行指令的操作码决定了不同指令在执行周期所需完成的不同操作，因此指令的操作码字段是控制单元的输入信号，它与时钟配合产生不同的控制信号。
- 2) 时序系统产生的机器周期信号和节拍信号。为了使控制单元按一定的先后顺序、一定的节奏发出各个控制信号，控制单元必须受时钟控制，即一个时钟脉冲使控制单元发送一个操作命令，或发送一组需要同时执行的操作命令。
- 3) 来自执行单元的反馈信息即标志。控制单元有时需依赖 CPU 当前所处的状态产生控制信号，如 BAN 指令，控制单元要根据上条指令的结果是否为负来产生不同的控制信号。

图 5.9 中，节拍发生器产生各机器周期中的节拍信号，使不同的微操作命令 C_i （控制信号）

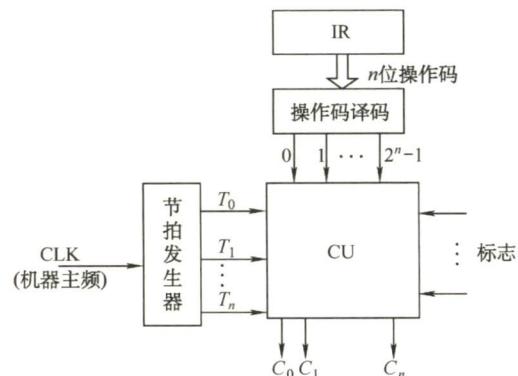


图 5.9 带指令译码器和节拍输入的控制单元图

按时间的先后发出。个别指令的操作不仅受操作码控制，还受状态标志控制，因此 CU 的输入来自操作码译码电路 ID、节拍发生器及状态标志，其输出到 CPU 内部或外部控制总线上。

注意：控制单元还接收来自系统总线（控制总线）的控制信号，如中断请求、DMA 请求。

2. 硬布线控制器的时序系统及微操作

1) 时钟周期。用时钟信号控制节拍发生器，可以产生节拍，每个节拍的宽度正好对应一个时钟周期。在每个节拍内机器可完成一个或几个需同时执行的操作。

2) 机器周期。机器周期可视为所有指令执行过程中的一个基准时间。不同指令的操作不同，指令周期也不同。访问一次存储器的时间是固定的，因此通常以存取周期作为基准时间，即内存中读取一个指令字的最短时间作为机器周期。在存储字长等于指令字长的前提下，取指周期也可视为机器周期。

在一个机器周期里可完成若干微操作，每个微操作都需一定的时间，可用时钟信号来控制产生每个微操作命令。

3) 指令周期。指令周期详见 5.2.1 节。

4) 微操作命令分析。控制单元具有发出各种操作命令（控制信号）序列的功能。这些命令与指令有关，而且必须按一定次序发出，才能使机器有序地工作。

执行程序的过程中，对于不同的指令，控制单元需发出各种不同的微操作命令。一条指令分为 3 个工作周期：取指周期、间址周期和执行周期。下面分析各个子周期的微操作命令。

① 取指周期的微操作命令。无论是什么指令，取指周期都需有下列微操作命令：

(PC) → MAR	现行指令地址 → MAR
1 → R	命令存储器读
M (MAR) → MDR	现行指令从存储器中读至 MDR
(MDR) → IR	现行指令 → IR
OP (IR) → CU	指令的操作码 → CU 译码
(PC) + 1 → PC	形成下一条指令的地址

② 间址周期的微操作命令。间址周期完成取操作数地址的任务，具体微操作命令如下：

Ad (IR) → MAR	将指令字中的地址码（形式地址）→ MAR
1 → R	命令存储器读
M (MAR) → MDR	将有效地址从存储器读至 MDR

③ 执行周期的微操作命令。执行周期的微操作命令视不同指令而定。

a. 非访存指令。

CLA	清 ACC 0 → ACC
COM	取反 $\overline{ACC} \rightarrow ACC$
SHR	算术右移 L (ACC) → R (ACC), $ACC_0 \rightarrow ACC_n$
CSL	循环左移 R (ACC) → L (ACC), $ACC_n \rightarrow ACC_0$
STP	停机指令 0 → G

b. 访存指令。

ADD X	加法指令
Ad (IR) → MAR, 1 → R	
M (MAR) → MDR	
(ACC) + (MDR) → ACC	
STA X	存数指令
Ad (IR) → MAR, 1 → W	
(ACC) → MDR	

(MDR) \rightarrow M (MAR)
 LDA X 取数指令
 Ad (IR) \rightarrow MAR, 1 \rightarrow R
 M (MAR) \rightarrow MDR
 (MDR) \rightarrow ACC

c. 转移指令。

JMP X 无条件转移	Ad (IR) \rightarrow PC
BAN X 条件转移 (负则转)	$A_0 \cdot Ad (IR) + \bar{A}_0 \cdot (PC) \rightarrow PC$

关注公众号【乘龙考研】
一手更新 稳定有保障

3. CPU 的控制方式

控制单元控制一条指令执行的过程，实质上是依次执行一个确定的微操作序列的过程。由于不同指令所对应的微操作数及复杂程度不同，因此每条指令和每个微操作所需的执行时间也不同。主要有以下3种控制方式。

- 1) 同步控制方式。所谓同步控制方式，是指系统有一个统一的时钟，所有的控制信号均来自这个统一的时钟信号。通常以最长的微操作序列和最烦琐的微操作作为标准，采取完全统一的、具有相同时间间隔和相同数目的节拍作为机器周期来运行不同的指令。
同步控制方式的优点是控制电路简单，缺点是运行速度慢。
- 2) 异步控制方式。异步控制方式不存在基准时标信号，各部件按自身固有的速度工作，通过应答方式进行联络。
异步控制方式的优点是运行速度快，缺点是控制电路比较复杂。
- 3) 联合控制方式。联合控制方式是介于同步、异步之间的一种折中。这种方式对各种不同的指令的微操作实行大部分采用同步控制、小部分采用异步控制的办法。

4. 硬布线控制单元设计步骤

硬布线控制单元设计步骤包括：

- 1) 列出微操作命令的操作时间表。先根据微操作节拍安排，列出微操作命令的操作时间表。操作时间表中包括各个机器周期、节拍下的每条指令完成的微操作控制信号。
表5.1列出了CLA、COM、SHR等10条机器指令微操作命令的操作时间表。表中FE、IND和EX为CPU工作周期标志， $T_0 \sim T_2$ 为节拍，I为间址标志，在取指周期的 T_2 时刻，若测得 $I = 1$ ，则IND触发器置“1”，标志进入间址周期；若 $I = 0$ ，则EX触发器置“1”，标志进入执行周期。同理，在间址周期的 T_2 时刻，若测得 $IND = 0$ （表示一次间接寻址），则EX触发器置“1”，进入执行周期；若测得 $IND = 1$ （表示多次间接寻址），则继续间接寻址。在执行周期的 T_2 时刻，CPU要向所有中断源发中断查询信号，若检测到有中断请求并满足响应条件，则INT触发器置“1”，标志进入中断周期。表中未列出INT触发器置“1”的操作和中断周期的微操作。表中第一行对应10条指令的操作码，代表不同的指令。若某指令有表中所列出的微操作命令，其对应的单元格内为1。
- 2) 进行微操作信号综合。在列出微操作时间表后，即可对它们进行综合分析、归类，根据微操作时间表可写出各微操作控制信号的逻辑表达式并进行适当的简化。表达式一般包括下列因素：

微操作控制信号 = 机器周期 \wedge 节拍 \wedge 脉冲 \wedge 操作码 \wedge 机器状态条件

根据表5.1便可列出每个微操作命令的初始逻辑表达式，经化简、整理可获得能用现有门电路实现的微操作命令逻辑表达式。

表 5.1 操作时间表

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	SHR	CSL	STP	ADD	STA	LDA	JMP	BAN	
FE 取指	T_0		(PC)→MAR	1	1	1	1	1	1	1	1	1	1	
			1→R	1	1	1	1	1	1	1	1	1	1	
	T_1		M(MAR)→MDR	1	1	1	1	1	1	1	1	1	1	
			(PC) + 1→PC	1	1	1	1	1	1	1	1	1	1	
FE 取指	T_2		(MDR)→IR	1	1	1	1	1	1	1	1	1	1	
			OP(IR)→ID	1	1	1	1	1	1	1	1	1	1	
			I	1→IND					1	1	1	1	1	
			\bar{I}	1→EX	1	1	1	1	1	1	1	1	1	
IND 间接寻址	T_0		Ad(IR)→MAR						1	1	1	1	1	
			1→R						1	1	1	1	1	
IND 间接寻址	T_1		M(MAR)→MDR						1	1	1	1	1	
	T_2		(MDR)→Ad(IR)						1	1	1	1	1	
			$\overline{\text{IND}}$	1→EX					1	1	1	1	1	
EX 执行	T_0		Ad(IR)→MAR						1	1	1			
			1→R						1		1			
			1→W							1				
	T_1		M(MAR)→MDR						1		1			
			(AC)→MDR							1				
	T_2		(AC) + (MDR)→AC						1					
			(MDR)→M(MAR)							1				
EX 执行	T_2		(MDR)→AC								1			
			0→AC	1										
			$\overline{AC} \rightarrow AC$		1									
			L(AC)→R(AC), AC_o 不变			1								
			$\rho^{-1}(AC)$				1							
			Ad(IR)→PC									1		
			A_0	Ad(IR)→PC									1	
			0→G					1						

例如，根据表 5.1 可写出 M(MAR)→MDR 微操作命令的逻辑表达式：

$M(\text{MAR}) \rightarrow \text{MDR}$

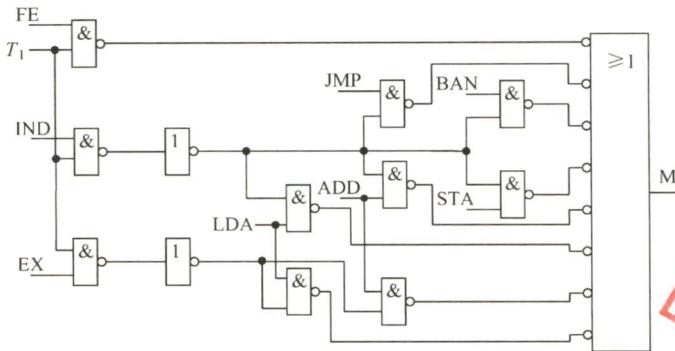
$$= FE \cdot T_1 + IND \cdot T_1(ADD + STA + LDA + JMP + BAN) + EX \cdot T_1(ADD + LDA)$$

$$= T_1 \{ FE + IND(ADD + STA + LDA + JMP + BAN) + EX(ADD + LDA) \}$$

式中，ADD、STA、LDA、JMP、BAN 均来自操作码译码器的输出。

- 3) 画出微操作命令的逻辑图。根据逻辑表达式可画出对应每个微操作信号的逻辑电路图，并用逻辑门电路实现。

例如， $M(\text{MAR}) \rightarrow \text{MDR}$ 的逻辑表达式所对应的逻辑图如图 5.10 所示，图中未考虑门的扇入系数。

图 5.10 产生 $M(MAR) \rightarrow MDR$ 命令的逻辑图

5.4.3 微程序控制器

微程序控制器采用存储逻辑实现，也就是把微操作信号代码化，使每条机器指令转化成为一段微程序并存入一个专门的存储器（控制存储器）中，微操作控制信号由微指令产生。

1. 微程序控制的基本概念

微程序设计思想就是将每条机器指令编写成一个微程序，每个微程序包含若干微指令，每条微指令对应一个或几个微操作命令。这些微程序可以存到一个控制存储器中，用寻址用户程序机器指令的办法来寻址每个微程序中的微指令。目前，大多数计算机都采用微程序设计技术。

微程序设计技术涉及的基本术语如下：

1) **微命令与微操作**。一条机器指令可以分解成一个微操作序列，这些微操作是计算机中最基本的、不可再分解的操作。在微程序控制的计算机中，将控制部件向执行部件发出的各种控制命令称为微命令，它是构成控制序列的最小单位。例如，打开或关闭某个控制门的电位信号、某个寄存器的打入脉冲等。微命令和微操作是一一对应的。微命令是微操作的控制信号，微操作是微命令的执行过程。

微命令有相容性和互斥性之分。相容性微命令是指那些可以同时产生、共同完成某一些微操作的微命令；而互斥性微命令是指在机器中不允许同时出现的微命令。相容和互斥都是相对的，一个微命令可以和一些微命令相容，和另一些微命令互斥。

注意：在组合逻辑控制器中也存在微命令与微操作这两个概念，它们并非只是微程序控制器的专有概念。

2) **微指令与微周期**。微指令是若干微命令的集合。存放微指令的控制存储器的单元地址称为微地址。一条微指令通常至少包含两大部分信息：

- ① 操作控制字段，又称微操作码字段，用于产生某一步操作所需的各种操作控制信号。
- ② 顺序控制字段，又称微地址码字段，用于控制产生下一条要执行的微指令地址。

微周期是指执行一条微指令所需的时间，通常为一个时钟周期。

3) **主存储器与控制存储器**。主存储器用于存放程序和数据，在 CPU 外部，用 RAM 实现；控制存储器（CM）用于存放微程序，在 CPU 内部，用 ROM 实现。

4) **程序与微程序**。程序是指令的有序集合，用于完成特定的功能；微程序是微指令的有序集合，一条指令的功能由一段微程序来实现。

微程序和程序是两个不同的概念。微程序是由微指令组成的，用于描述机器指令。微程序实际上是机器指令的实时解释器，是由计算机设计者事先编制好并存放在控制存储器中的，一般不提供给用户。对于程序员来说，计算机系统中微程序的结构和功能是透明的，无须知道。而程序最终由机器指令组成，是由软件设计人员事先编制好并存放在主存或辅存中的。

读者应注意区分以下寄存器：

- ① 地址寄存器 (MAR)。用于存放主存的读/写地址。
- ② 微地址寄存器 (CMAR)。用于存放控制存储器的读/写微指令的地址。
- ③ 指令寄存器 (IR)。用于存放从主存中读出的指令。
- ④ 微指令寄存器 (CMDR 或 μ IR)。用于存放从控制存储器中读出的微指令。

2. 微程序控制器组成和工作过程

(1) 微程序控制器的基本组成

图 5.11 所示为一个微程序控制器的基本结构，主要画出了微程序控制器比组合逻辑控制器多出的部件，包括：

- ① 控制存储器。它是微程序控制器的核心部件，用于存放各指令对应的微程序，控制存储器可用只读存储器 ROM 构成。
- ② 微指令寄存器。用于存放从 CM 中取出的微指令，它的位数同微指令字长相等。
- ③ 微地址形成部件。用于产生初始微地址和后继微地址，以保证微指令的连续执行。
- ④ 微地址寄存器。接收微地址形成部件送来的微地址，为在 CM 中读取微指令作准备。

(2) 微程序控制器的工作过程

微程序控制器的工作过程实际上就是在微程序控制器的控制下计算机执行机器指令的过程，这个过程可以描述如下：

- ① 执行取微指令公共操作。具体的执行是：在机器开始运行时，自动将取指微程序的入口地址送入 CMAR，并从 CM 中读出相应的微指令送入 CMDR。取指微程序的入口地址一般为 CM 的 0 号单元，当取指微程序执行完后，从主存中取出的机器指令就已存入指令寄存器中。
- ② 由机器指令的操作码字段通过微地址形成部件产生该机器指令所对应的微程序的入口地址，并送入 CMAR。
- ③ 从 CM 中逐条取出对应的微指令并执行。
- ④ 执行完对应于一条机器指令的一个微程序后，又回到取指微程序的入口地址，继续第①步，以完成取下一条机器指令的公共操作。

以上是一条机器指令的执行过程，如此周而复始，直到整个程序执行完毕。

(3) 微程序和机器指令

通常，一条机器指令对应一个微程序。由于任何机器指令的取指令操作都是相同的，因此可将取指令操作的微命令统一编成一个微程序，这个微程序只负责将指令从主存单元中取出并送至指令寄存器。此外，也可编出对应间址周期的微程序和中断周期的微程序。这样，控制存储器中

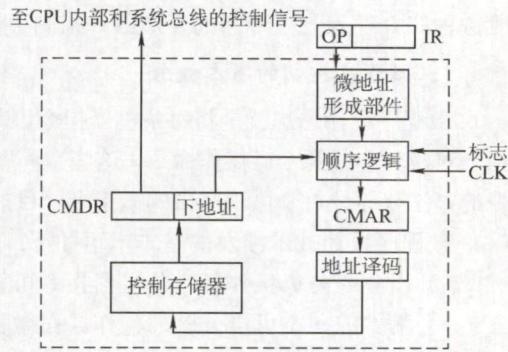


图 5.11 微程序控制器的基本结构

的微程序个数应为机器指令数再加上对应取指、间址和中断周期等公共的微程序数。

3. 微指令的编码方式

微指令的编码方式又称微指令的控制方式，是指如何对微指令的控制字段进行编码，以形成控制信号。编码的目标是在保证速度的情况下，尽量缩短微指令字长。

(1) 直接编码(直接控制)方式

微指令的直接编码方式如图 5.12 所示。直接编码法无须进行译码，微指令的微命令字段中每位都代表一个微命令。设计微指令时，选用或不选用某个微命令，只要将表示该微命令的对应位设置成 1 或 0 即可。每个微命令对应并控制数据通路中的一个微操作。

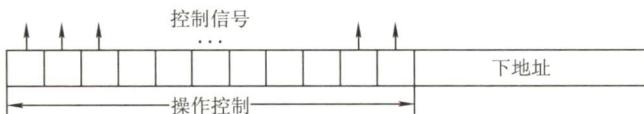


图 5.12 微指令的直接编码方式

这种编码的优点是简单、直观，执行速度快，操作并行性好；缺点是微指令字长过长， n 个微命令就要求微指令的操作字段有 n 位，造成控制存储器容量极大。

(2) 字段直接编码方式

将微指令的微命令字段分成若干小字段，把互斥性微命令组合在同一字段中，把相容性微命令组合在不同字段中，每个字段独立编码，每种编码代表一个微命令且各字段编码含义单独定义，与其他字段无关，这就是字段直接编码方式，如图 5.13 所示。

这种方式可以缩短微指令字长，但因为要通过译码电路后再发出微命令，因此比直接编码方式慢。

微命令字段分段的原则：

- ① 互斥性微命令分在同一段内，相容性微命令分在不同段内。
- ② 每个小段中包含的信息位不能太多，否则将增加译码线路的复杂性和译码时间。
- ③ 一般每个小段还要留出一个状态，表示本字段不发出任何微命令。因此，当某字段的长度为 3 位时，最多只能表示 7 个互斥的微命令，通常用 000 表示不操作。

(3) 字段间接编码方式

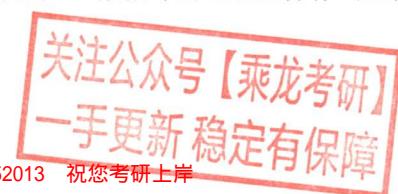
一个字段的某些微命令需由另一个字段中的某些微命令来解释，由于不是靠字段直接译码发出的微命令，因此称为字段间接编码，又称隐式编码。这种方式可进一步缩短微指令字长，但因削弱了微指令的并行控制能力，因此通常作为字段直接编码方式的一种辅助手段。

4. 微指令的地址形成方式

后继微地址的形成主要有以下两大基本类型：

- 1) 直接由微指令的下地址字段指出。微指令格式中设置一个下地址字段，由微指令的下地址字段直接指出后继微指令的地址，这种方式又称断定方式。
- 2) 根据机器指令的操作码形成。机器指令取至指令寄存器后，微指令的地址由操作码经微地址形成部件形成。

实际上，微指令序列地址的形成方式还有以下几种：



- ① 增量计数器法，即 $(CMAR) + 1 \rightarrow CMAR$ ，适用于后继微指令的地址连续的情况。
- ② 根据各种标志决定微指令分支转移的地址。
- ③ 通过测试网络形成。
- ④ 由硬件直接产生微程序入口地址。

电源加电后，第一条微指令的地址可由专门的硬件电路产生，也可由外部直接向 CMAR 输入微指令的地址，这个地址即为取指周期微程序的入口地址。

5. 微指令的格式

微指令格式与微指令的编码方式有关，通常分水平型微指令和垂直型微指令两种。

- 1) 水平型微指令。从编码方式看，直接编码、字段直接编码、字段间接编码和混合编码都属于水平型微指令。水平型微指令的基本指令格式如图 5.14 所示，指令字中的一位对应一个控制信号，有输出时为 1，否则为 0。一条水平型微指令定义并执行几种并行的基本操作。

A ₁	A ₂	...	A _{n-1}	A _n	判断测试字段	后继地址字段
操作控制					顺序控制	

图 5.14 水平型微指令格式

水平型微指令的优点是微程序短，执行速度快；缺点是微指令长，编写微程序较麻烦。

- 2) 垂直型微指令。垂直型微指令的特点是采用类似机器指令操作码的方式，在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，其基本的指令格式如图 5.15 所示。一条垂直型微指令只能定义并执行一种基本操作。

μ OP	Rd	Rs
微操作码	目的地址	源地址

图 5.15 垂直型微指令格式

垂直型微指令格式的优点是微指令短、简单、规整，便于编写微程序；缺点是微程序长，执行速度慢，工作效率低。

- 3) 混合型微指令。在垂直型的基础上增加一些不太复杂的并行操作。微指令较短，仍便于编写；微程序也不长，执行速度加快。
- 4) 水平型微指令和垂直型微指令的比较如下：
 - 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；垂直型微指令执行的时间长。
 - ③ 由水平型微指令解释指令的微程序，具有微指令字较长但微程序短的特点；垂直型微指令则与之相反，其微指令字较短而微程序长。
 - 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，相对容易掌握。

6. 微程序控制单元的设计步骤

微程序控制单元设计的主要任务是编写各条机器指令所对应的微程序。具体的设计步骤如下：

- 1) 写出对应机器指令的微操作命令及节拍安排。无论是组合逻辑设计还是微程序设计，对应相同的 CPU 结构，两种控制单元的微操作命令和节拍安排都是极相似的。如微程序控制单元在取指阶段发出的微操作命令及节拍安排如下：

T_0	$(PC) \rightarrow MAR, 1 \rightarrow R$
T_1	$M(MAR) \rightarrow MDR, (PC) + 1 \rightarrow PC$

$T_2 \quad (MDR) \rightarrow IR, OP(IR) \rightarrow$ 微地址形成部件

与硬布线控制单元相比，只在 T_2 节拍内的微操作命令不同。微程序控制单元在 T_2 节拍内要将指令的操作码送至微地址形成部件，即 $OP(IR) \rightarrow$ 微地址形成部件，以形成该条机器指令的微程序首地址。而硬布线控制单元在 T_2 节拍内要将指令的操作码送至指令译码器，以控制 CU 发出相应的微命令，即 $OP(IR) \rightarrow ID$ 。

若把一个节拍 T 内的微操作安排在一条微指令中完成，上述微操作对应 3 条微指令。但由于微程序控制的所有控制信号都来自微指令，而微指令又存在控制存储器中，因此欲完成上述这些微操作，必须先将微指令从控制存储器中读出，即必须先给出这些微指令的地址。在取指微程序中，除第一条微指令外，其余微指令的地址均由上一条微指令的下地址字段直接给出，因此上述每条微指令都需增加一个将微指令下地址字段送至 CMAR 的微操作，记为 $Ad(CMDR) \rightarrow CMAR$ 。取指微程序的最后一条微指令，其后继微指令的地址是由微地址形成部件形成的，即微地址形成部件 $\rightarrow CMAR$ 。为了反映该地址与操作码有关，因此记为 $OP(IR) \rightarrow$ 微地址形成部件 $\rightarrow CMAR$ 。

综上所述，考虑到需要形成后继微指令地址，上述分析的取指操作共需 6 条微指令完成：

T_0	$(PC) \rightarrow MAR, 1 \rightarrow R$
T_1	$Ad(CMDR) \rightarrow CMAR$
T_2	$M(MAR) \rightarrow MDR, (PC) + 1 \rightarrow PC$
T_3	$Ad(CMDR) \rightarrow CMAR$
T_4	$(MDR) \rightarrow IR$
T_5	$OP(IR) \rightarrow$ 微地址形成部件 $\rightarrow CMAR$

关注公众号【乘龙考研】
一手更新 稳定有保障

执行阶段的微操作命令及节拍安排，分配原则类似。与硬布线控制单元微操作命令的节拍安排相比，多了将下一条微指令地址送至 CMAR 的微操作命令，即 $Ad(CMDR) \rightarrow CMAR$ 。其余的微操作命令与硬布线控制单元相同。

注意：这里为了理解，应将微指令和机器指令相联系，因为每执行完一条微指令后要得到下一条微指令的地址。

2) 确定微指令格式。微指令格式包括微指令的编码方式、后继微指令地址的形成方式和微指令字长等。

根据微操作个数决定采用何种编码方式，以确定微指令的操作控制字段的位数。由微指令数确定微指令的顺序控制字段的位数。最后按操作控制字段位数和顺序控制字段位数就可确定微指令字长。

3) 编写微指令码点。根据操作控制字段每位代表的微操作命令，编写每条微指令的码点。

7. 动态微程序设计和毫微程序设计

1) 动态微程序设计。在一台微程序控制的计算机中，假如能根据用户的要求改变微程序，则这台机器就具有动态微程序设计功能。

动态微程序的设计需要可写控制寄存器的支持，否则难以改变微程序的内容。实现动态微程序设计可采用可擦除可编程只读存储器 (EPROM)。

2) 毫微程序设计。在普通的微程序计算机中，从主存取出的每条指令是由放在控制存储器中的微程序来解释执行的，通过控制线对硬件进行直接控制。

若硬件不由微程序直接控制，而是通过存放在第二级控制存储器中的毫微程序来解释的，这个第二级控制存储器就称为毫微存储器，直接控制硬件的是毫微微指令。

8. 硬布线和微程序控制器的特点

- 1) 硬布线控制器的特点。硬布线控制器的优点是由于控制器的速度取决于电路延迟，所以速度快；缺点是由于将控制部件视为专门产生固定时序控制信号的逻辑电路，所以把用最少元件和取得最高速度作为设计目标，一旦设计完成，就不可能通过其他额外修改添加新功能。
- 2) 微程序控制器的特点。微程序控制器的优点是同组合逻辑控制器相比，微程序控制器具有规整性、灵活性、可维护性等一系列优点；缺点是由于微程序控制器采用了存储程序原理，所以每条指令都要从控制存储器中取一次，影响速度。

为便于比较，下面以表格的形式对比二者的不同，见表 5.2。

表 5.2 微程序控制器与硬布线控制器的对比

类别 对比项目	微程序控制器	硬布线控制器
工作原理	微操作控制信号以微程序的形式存放在控制存储器中，执行指令时读出即可	微操作控制信号由组合逻辑电路根据当前的指令码、状态和时序，即时产生
执行速度	慢	快
规整性	较规整	烦琐、不规整
应用场合	CISC CPU	RISC CPU
易扩充性	易扩充修改	困难

5.4.4 本节习题精选

一、单项选择题

01. 取指令操作（ ）。
 - A. 受到上一条指令的操作码控制
 - B. 受到当前指令的操作码控制
 - C. 受到下一条指令的操作码控制
 - D. 是控制器固有的功能，不需要在操作码控制下进行
02. 在组合逻辑控制器中，微操作控制信号的形成主要与（ ）信号有关。
 - A. 指令操作码和地址码
 - B. 指令译码信号和时钟
 - C. 操作码和条件码
 - D. 状态信息和条件
03. 在微程序控制器中，形成微程序入口地址的是（ ）。
 - A. 机器指令的地址码字段
 - B. 微指令的微地址码字段
 - C. 机器指令的操作码字段
 - D. 微指令的微操作码字段
04. 下列不属于微指令结构设计所追求目标的是（ ）。
 - A. 提高微程序的执行速度
 - B. 提供微程序设计的灵活性
 - C. 缩短微指令的长度
 - D. 增大控制存储器的容量
05. 微程序控制器的速度比硬布线控制器慢，主要是因为（ ）。
 - A. 增加了从磁盘存储器读取微指令的时间
 - B. 增加了从主存读取微指令的时间
 - C. 增加了从指令寄存器读取微指令的时间
 - D. 增加了从控制存储器读取微指令的时间

06. 微程序控制存储器属于()的一部分。
 A. 主存 B. 外存 C. CPU D. 缓存
07. 以下说法中, 正确的是()。
 A. 采用微程序控制器是为了提高速度
 B. 控制存储器由高速RAM电路组成
 C. 微指令计数器决定指令执行顺序
 D. 一条微指令存放在控制器的一个控制存储器单元中
08. 硬布线控制器与微程序控制器相比,()。
 A. 硬布线控制器的时序系统比较简单
 B. 微程序控制器的时序系统比较简单
 C. 两者的时序系统复杂程度相同
 D. 可能是硬布线控制器的时序系统比较简单, 也可能是微程序控制器的时序系统比较简单
09. 在微程序控制器中, 控制部件向执行部件发出的某个控制信号称为()。
 A. 微程序 B. 微指令 C. 微操作 D. 微命令
10. 在微程序控制器中, 机器指令与微指令的关系是()。
 A. 每条机器指令由一条微指令来执行
 B. 每条机器指令由若干微指令组成的微程序来解释执行
 C. 若干机器指令组成的程序可由一个微程序来执行
 D. 每条机器指令由若干微程序执行
11. 水平型微指令与垂直型微指令相比,()。
 A. 前者一次只能完成一个基本操作
 B. 后者一次只能完成一个基本操作
 C. 两者都是一次只能完成一个基本操作
 D. 两者都能一次完成多个基本操作
12. 兼容性微命令指几个微命令()。
 A. 可以同时出现 B. 可以相继出现 C. 可以相互代替 D. 可以相处容错
13. 在微程序控制方式中, 以下说法正确的是()。
 I. 采用微程序控制器的处理器称为微处理器
 II. 每条机器指令由一段微程序来解释执行
 III. 在微指令的编码中, 效率最低的是直接编码方式
 IV. 水平型微指令能充分利用数据通路的并行结构
 A. I、II B. II、IV C. I、III D. III、IV
14. 下列说法中, 正确的是()。
 I. 微程序控制方式和硬布线方式相比较, 前者可以使指令的执行速度更快
 II. 若采用微程序控制方式, 则可用μPC取代PC
 III. 控制存储器可以用ROM实现
 IV. 指令周期也称CPU周期
 A. I、III B. II、III C. 只有 III D. I、III、IV
15. 通常情况下, 一个微程序的周期对应一个()。
 A. 指令周期 B. 主频周期 C. 机器周期 D. 工作周期



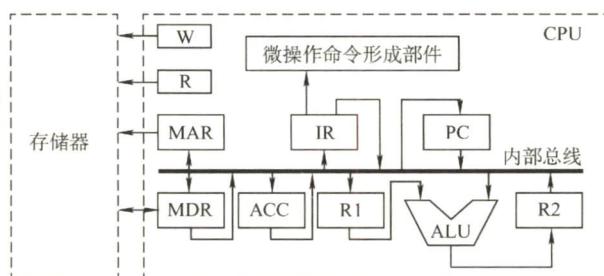
16. 下列部件中属于控制部件的是()。
 I. 指令寄存器 II. 操作控制器 III. 程序计数器 IV. 状态条件寄存器
 A. I、III、IV B. I、II、III C. I、II、IV D. I、II、III、IV
17. 下列部件中属于执行部件的是()。
 I. 控制器 II. 存储器 III. 运算器 IV. 外围设备
 A. I、III、IV B. II、III、IV C. II、IV D. I、II、III、IV
18. 【2009 统考真题】相对于微程序控制器，硬布线控制器的特点是()。
 A. 指令执行速度慢，指令功能的修改和扩展容易
 B. 指令执行速度慢，指令功能的修改和扩展难
 C. 指令执行速度快，指令功能的修改和扩展容易
 D. 指令执行速度快，指令功能的修改和扩展难
19. 【2012 统考真题】某计算机的控制器采用微程序控制方式，微指令中的操作控制字段采用字段直接编码法，共有 33 个微命令，构成 5 个互斥类，分别包含 7、3、12、5 和 6 个微命令，则操作控制字段至少有()。
 A. 5 位 B. 6 位 C. 15 位 D. 33 位
20. 【2014 统考真题】某计算机采用微程序控制器，共有 32 条指令，公共的取指令微程序包含 2 条微指令，各指令对应的微程序平均由 4 条微指令组成，采用断定法（下地址字段法）确定下一条微指令地址，则微指令中下地址字段的位数至少是()。
 A. 5 B. 6 C. 8 D. 9
21. 【2017 统考真题】下列关于主存储器 (MM) 和控制存储器 (CS) 的叙述，错误的是()。
 A. MM 在 CPU 外，CS 在 CPU 内
 B. MM 按地址访问，CS 按内容访问
 C. MM 存储指令和数据，CS 存储微指令
 D. MM 用 RAM 和 ROM 实现，CS 用 ROM 实现
22. 【2019 统考真题】下列有关处理器时钟脉冲信号的叙述中，错误的是()。
 A. 时钟脉冲信号由机器脉冲源发出的脉冲信号经整形和分频后形成
 B. 时钟脉冲信号的宽度称为时钟周期，时钟周期的倒数为机器主频
 C. 时钟周期以相邻状态单元间组合逻辑电路的最大延迟为基准确定
 D. 处理器总是在每来一个时钟脉冲信号时就开始执行一条新的指令
23. 【2019 统考真题】某指令功能为 $R[r2] \leftarrow R[r1] + M[R[r0]]$ ，其两个源操作数分别采用寄存器、寄存器间接寻址方式。对于下列给定部件，该指令在取数及执行过程中需要用到的是()。
 I. 通用寄存器组 (GPRs) II. 算术逻辑单元 (ALU)
 III. 存储器 (Memory) IV. 指令译码器 (ID)
 A. 仅 I、II B. 仅 I、II、III C. 仅 II、III、IV D. 仅 I、III、IV
24. 【2021 统考真题】下列寄存器中，汇编语言程序员可见的是()。
 I. 指令寄存器 II. 微指令寄存器
 III. 基址寄存器 IV. 标志/状态寄存器
 A. 仅 I、II B. 仅 I、IV C. 仅 II、IV D. 仅 III、IV

二、综合应用题

01. 若某机主频为 200MHz，每个指令周期平均为 2.5 个 CPU 周期，每个 CPU 周期平均包

括 2 个主频周期，问：

- 1) 该机平均指令执行速度为多少 MIPS?
 - 2) 若主频不变，但每条指令平均包括 5 个 CPU 周期，每个 CPU 周期又包含 4 个主频周期，平均指令执行速度又为多少 MIPS?
 - 3) 由此可得出什么结论?
02. 1) 若存储器容量为 $64K \times 32$ 位，指出主机中各寄存器的位数。
- 2) 写出硬布线控制器完成 STA X (X 为主存地址) 指令发出的全部微操作命令及节拍安排。
- 3) 若采用微程序控制，还需增加哪些微操作?
03. 假设某机器有 80 条指令，平均每条指令由 4 条微指令组成，其中有一条取指微指令是所有指令公用的。已知微指令长度为 32 位，请估算控制存储器 CM 容量。
04. 某微程序控制器中，采用水平型直接控制（编码）方式的微指令格式，后续微指令地址由微指令的下地址字段给出。已知机器共有 28 个微命令，6 个互斥的可判定的外部条件，控制存储器的容量为 512×40 位。试设计其微指令的格式，并说明理由。
05. 某机共有 52 个微操作控制信号，构成 5 个相斥类的微命令组，各组分别包含 5、8、2、15、22 个微命令。已知可判定的外部条件有两个，微指令字长 28 位。
 - 1) 按水平型微指令格式设计微指令，要求微指令的下地址字段直接给出后继微指令地址。
 - 2) 指出控制存储器的容量。
06. 设 CPU 中各部件及其相互连接关系如下图所示，其中 W 是写控制标志；R 是读控制标志；R1、R2 是暂存器。



- 1) 写出指令 ADD #a (#为立即寻址特征，隐含的操作数在 ACC 寄存器中) 在执行阶段所完成的微操作命令及节拍安排。
- 2) 假设要求在取指周期实现 $(PC)+1 \rightarrow PC$ ，且由 ALU 完成此操作 (ALU 能对它的一个源操作数完成加 1 运算)。以最少的节拍写出取指周期全部微操作命令及节拍安排。

5.4.5 答案与解析

一、单项选择题

01. D

取指令阶段完成的任务是将现行指令从主存中取出并送至指令寄存器，这个操作是公共的操作，是每条指令都要进行的，与具体的指令无关，所以不需要操作码的控制。

02. B

CU 的输入信号来源如下：①经指令译码器译码产生的指令信息；②时序系统产生的机器周

期信号和节拍信号；③来自执行单元的反馈信息即标志。前两者是主要因素。

03. C

执行公用的取指微程序从主存中取出机器指令后，由机器指令的操作码字段指出各个微程序的入口地址（初始微地址）。

04. D

微指令的设计目标和指令结构的设计目标类似，都是基于执行速度、灵活性和指令长度这三个主要方面考虑的。而控制存储器容量的大小与微指令的设计目标无关。

05. D

在微程序控制中，控制存储器中存放有微指令，在执行时需要从中读出相应的微指令，从而增加了时间消耗。

06. C

微程序控制存储器用来存放微程序，是微程序控制器的核心部件，属于 CPU 的一部分，而不属于主存。

07. D

硬布线控制器采用硬件电路，速度较快，但设计难度大、成本高。微程序控制器的速度较慢，但灵活性高。通常控制存储器采用 ROM 组成。微指令计数器决定的是微指令执行顺序。

08. B

硬布线控制器需要结合各微操作的节拍安排，综合分析，写出逻辑表达式，再设计成逻辑电路图，因此时序系统比较复杂；而微程序只需按照节拍的安排，顺序执行微指令，因此比较简单。

09. D

在微程序控制器中，控制部件向执行部件发出的控制信号称为微命令，微命令执行的操作称为微操作。微指令则是若干微命令的集合，若干微指令的有序集合称为微程序。

10. B

在一个 CPU 周期中，一组实现一定功能的微命令的组合构成一条微指令，有序的微指令序列构成一段微程序，微程序的作用是实现一条对应的机器指令。

11. B

一条水平型微指令能定义并执行几种并行的基本操作；一条垂直型微指令只能定义并执行一种基本操作。

12. A

兼容性微命令是指那些可以同时产生、共同完成某些微操作的微命令。

13. B

微处理器是相对于一些大型处理器而言的，与微程序控制器没有必然联系。不管是采用微程序控制器，还是采用硬布线控制器，微机的 CPU 都是微处理器，I 错误。微程序的设计思想就是将每条机器指令编写成一个微程序，每个微程序包含若干条微指令，每条微指令对应一个或几个微操作命令，II 正确。直接编码方式中每位代表一个微命令，不需要译码，因此执行效率最高，只是这种方式会使得微指令的位数大大增加，III 错误。一条水平型微指令能定义并执行几种并行的基本操作，因此能够更充分利用数据通路的并行结构，IV 正确。

14. C

微程序控制方式采用编程方式来执行指令，而硬布线方式则采用硬件方式来执行指令，因此硬布线方式速度较快，I 错误。 μ PC 无法取代 PC，因为它只在微程序中指向下一条微指令地址的寄存器。因此它也必然不可能知道这段微程序执行完毕后下一条是什么指令，II 错误。由于每条微指令执行时所发出的控制信号是事先设计好的，不需要改变，因此存放所有控制信号的存储器

应为 ROM, III 正确。指令周期是从一条指令启动到下一条指令启动的间隔时间, 而 CPU 周期是机器周期, 是指令执行中每步操作所需的时间, IV 错误。

15. A

一条微指令包含一组实现一定操作功能的微命令。许多条微指令组成的序列构成微程序, 微程序则完成对应指令的解释执行。在采用微程序控制器的 CPU 中, 一条指令对应一个微程序, 一个微程序由许多微指令构成, 一条微指令会发出很多不同的微命令。

16. B

CPU 控制器主要由三个部件组成: 指令寄存器、程序计数器和操作控制器。状态条件寄存器通常属于运算器的部件, 保存由算术指令和逻辑指令运行或测试的结果建立的各种条件码内容, 如运算结果进位标志 (C)、运算结果溢出标志 (V) 等。

17. B

一台数字计算机基本上可以划分为两大部分: 控制部件和执行部件。控制器就是控制部件, 而运算器、存储器、外围设备相对控制器来说就是执行部件。

18. D

微程序控制器采用了“存储程序”的原理, 每条机器指令对应一个微程序, 因此修改和扩充容易, 灵活性好, 但每条指令的执行都要访问控制存储器, 所以速度慢。硬布线控制器采用专门的逻辑电路实现, 其速度主要取决于逻辑电路的延迟, 因此速度快, 但修改和扩展困难, 灵活性差。

19. C

字段直接编码法将微命令字段分成若干小字段, 互斥性微命令组合在同一字段中, 相容性微命令分在不同字段中, 每个字段还要留出一个状态, 表示本字段不发出任何微命令。5 个互斥类, 分别包含 7、3、12、5 和 6 个微命令, 需要 3、2、4、3 和 3 位, 共 15 位。

20. C

计算机共有 32 条指令, 各个指令对应的微程序平均为 4 条, 则指令对应的微指令为 $32 \times 4 = 128$ 条, 而公共微指令还有 2 条, 整个系统中微指令的条数共为 $128 + 2 = 130$ 条, 所以需要 $\lceil \log_2 130 \rceil = 8$ 位才能寻址到 130 条微指令, 答案选 C。

21. B

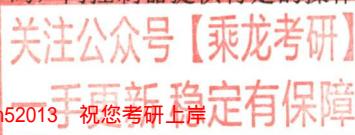
主存储器就是我们通常所说的主存, 它在 CPU 外, 用于存储指令和数据, 由 RAM 和 ROM 实现。控制存储器用来存放实现指令系统的所有微指令, 是一种只读型存储器, 机器运行时只读不写, 在 CPU 的控制器内。CS 按照微指令的地址访问, 所以 B 错误。

22. D

时钟脉冲信号的宽度称为时钟周期, 时钟周期是 CPU 工作的最短时间单位, 时钟周期的倒数为机器主频。时钟脉冲信号是由机器脉冲源发出的脉冲信号经整形和分频后形成的, 时钟周期以相邻状态单元间组合逻辑电路的最大延迟为基准确定。指令周期由若干机器周期来表示, 一个机器周期又包含若干时钟周期, 只有在理想情况下的流水线 CPU 中, 才可能实现每个时钟周期开始执行一条新指令, 但“总是”显然描述有误。

23. B

该指令的两个源操作数分别采用寄存器、寄存器间接寻址方式, 因此在取数阶段需要用到通用寄存器组 (GPRs) 和存储器 (Memory); 在执行阶段, 两个源操作数相加需要用到算术逻辑单元 (ALU)。而指令译码器 (ID) 用于对操作码字段进行译码, 向控制器提供特定的操作信号, 在取数及执行阶段用不到, 所以选 B。



24. D

汇编程序员可见的寄存器有基址寄存器（用于实现多道程序设计或者编制浮动程序）和状态/标志寄存器、程序计数器 PC 及通用寄存器组；而 MAR、MDR、IR 是 CPU 的内部工作寄存器，对汇编程序员不可见。微指令寄存器属于微程序控制器的组成部分，它是硬件设计者的任务，对汇编程序员是透明的（不可见的）。

二、综合应用题**01. 【解答】**

1) 主频为 200MHz，所以主频周期 = $1/200\text{MHz} = 0.005\mu\text{s}$ 。

每个指令周期平均为 2.5 个 CPU 周期，每个 CPU 周期平均包括 2 个主频周期，所以一条指令的执行时间 = $2 \times 2.5 \times 0.005\mu\text{s} = 0.025\mu\text{s}$ 。

该机平均指令执行速度 = $1/0.025 = 40\text{MIPS}$ 。

2) 每条指令平均包括 5 个 CPU 周期，每个 CPU 周期又包含 4 个主频周期，所以一条指令的执行时间 = $4 \times 5 \times 0.005\mu\text{s} = 0.1\mu\text{s}$ 。

该机平均指令执行速度 = $1/0.1 = 10\text{MIPS}$

3) 由此可见，指令的复杂程度会影响指令的平均执行速度。

02. 【解答】

1) 主机中各寄存器的位数如下图所示。

ACC	MQ	ALU	X	IR	MDR	PC	MAR
32	32	32	32	32	32	16	16

存储容量 = 存储单元个数×存储字长， $64\text{K} = 2^{16}$ ，因此 PC 和 MAR 为 16 位，而 MDR 为 32 位，其他寄存器的位数与 MDR 的相等。

2) 微操作命令及节拍安排如下：

T_0	$(PC) \rightarrow MAR, 1 \rightarrow R$
T_1	$M(MAR) \rightarrow MDR, (PC) + 1 \rightarrow PC$
T_2	$(MDR) \rightarrow IR, OP(IR) \rightarrow ID$
T_0	$Ad(IR) \rightarrow MAR, 1 \rightarrow W$
T_1	$(ACC) \rightarrow MDR$
T_2	$(MDR) \rightarrow M(MAR)$

3) 若采用微程序控制，还需增加下列微操作：

取指周期：

$Ad(CMDR) \rightarrow CMAR$
$OP(IR) \rightarrow CMAR$

执行周期：

$Ad(CMDR) \rightarrow CMAR$

03. 【解答】

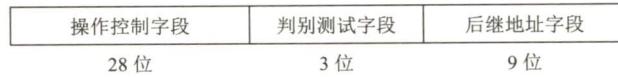
总的微指令条数 = $(4-1) \times 80 + 1 = 241$ 条，每条微指令占一个控制存储器单元，控制存储器 CM 的容量为 2 的 n 次幂，而 241 刚好小于 256，所以 CM 的容量 = 256×32 位 = 1KB。

04. 【解答】

水平型微指令由操作控制字段、判别测试字段和下地址字段三部分构成。因为微指令采用直接控制（编码）方式，所以其操作控制字段的位数等于微命令数，为 28 位。又由于后继微指令地址由下地址字段给出，因此其下地址字段的位数可根据控制存储器的容量（ 512×40 位）确定为

关注公众号【乘龙考研】
一手更新 稳定有保障

9位($512 = 2^9$)。当微程序出现分支时，后续微指令地址的形成取决于状态条件—6个互斥的可判定外部条件，因此状态位应编码成3位。非分支时的后续微指令地址由微指令的下地址字段直接给出。微指令的格式如下图所示。



05. 【解答】

1) 根据5个互斥类的微命令组，各组分别包含5、8、2、15、22个微命令，考虑到每组必须增加一种不发命令的情况，条件测试字段应包含一种不转移的情况，则5个控制字段分别需给出6、9、3、16、23种状态，对应3、4、2、4、5位(共18位)，条件测试字段取2位。根据微指令字长为28位，下地址字段取 $28 - 18 - 2 = 8$ 位，则其微指令格式如下图所示。



2) 根据后继地址字段为8位，微指令字长为28位，得出控制存储器的容量为 $2^8 \times 28$ 位。

06. 【解答】

1) 含有ACC的立即寻址，一个操作数隐藏在ACC中，立即寻址的加法指令执行周期的微操作命令及节拍安排如下：

T_0 Ad(IR) \rightarrow R1	立即数 \rightarrow R1
T_1 (R1) + (ACC) \rightarrow R2	ACC通过总线送ALU
T_2 (R2) \rightarrow ACC	结果 \rightarrow ACC

2) 由于 $(PC) + 1 \rightarrow PC$ 需由ALU完成，因此PC的值可作为ALU的一个源操作数，在ALU做加1运算得到 $(PC) + 1$ 后，结果送至与ALU输出端相连的R2，然后送至PC。

此题的关键是要考虑总线冲突的问题，因此取指周期的微操作命令及节拍安排如下：

T_0 (PC) \rightarrow MAR, 1 \rightarrow R	PC通过总线送MAR
T_1 M(MAR) \rightarrow MDR, (PC) + 1 \rightarrow R2	PC通过总线送ALU完成加1
T_2 (MDR) \rightarrow IR, OP(IR) \rightarrow 微操作命令形成部件	MDR通过总线送IR
T_3 (R2) \rightarrow PC	R2通过总线送PC

5.5 异常和中断机制

关注公众号【乘龙考研】
一手更新 稳定有保障

现代计算机中都配有完善的异常和中断处理系统，CPU的数据通路中有相应的异常检测和响应逻辑，外设接口中有相应的中断请求和控制逻辑，操作系统中有相应的中断服务程序。这些中断硬件线路和中断服务程序有机结合，共同完成异常和中断的处理过程。

5.5.1 异常和中断的基本概念

由CPU内部产生的意外事件被称为异常，有些教材中也称内中断。由来自CPU外部的设备向CPU发出的中断请求被称为中断，通常用于信息的输入和输出，有些教材中也称外中断。异常是CPU执行一条指令时，由CPU在其内部检测到的、与正在执行的指令相关的同步事件；中断

是一种典型的由外部设备触发的、与当前正在执行的指令无关的异步事件。

异常和中断处理过程的大致描述如下：当 CPU 在执行用户程序的第 i 条指令时检测到一个异常事件，或者在执行第 i 条指令后发现一个中断请求信号，则 CPU 打断当前用户程序，然后转到相应的异常或中断处理程序去执行。若异常或中断处理程序能够解决相应的问题，则在异常或中断处理程序的最后，CPU 通过执行异常或中断返回指令，回到被打断的用户程序的第 i 条指令或第 $i + 1$ 条指令继续执行；若异常或中断处理程序发现是不可恢复的致命错误，则终止用户程序。通常情况下，对异常和中断的具体处理过程由操作系统（和驱动程序）完成。

异常和中断的处理过程基本是相同的，这也是有些教材将两者统称为中断的原因。

5.5.2 异常和中断的分类

1. 异常的分类

异常是由 CPU 内部产生的意外事件，分为硬故障中断和程序性异常。硬故障中断是由硬连线出现异常引起的，如存储器校验错、总线错误等。程序性异常也称软件中断，是指在 CPU 内部因执行指令而引起的异常事件。如整除 0、溢出、断点、单步跟踪、非法指令、栈溢出、地址越界、缺页等。按异常发生原因和返回方式的不同，可分为故障、自陷和终止。

(1) 故障 (Fault)

指在引起故障的指令启动后、执行结束前被检测到的异常事件。例如，指令译码时，出现“非法操作码”；取数据时，发生“缺段”或“缺页”；执行整数除法指令时，发现“除数为 0”等。对于“缺段”“缺页”等异常事件，经处理后，可将所需的段或页面从磁盘调入主存，回到发生故障的指令继续执行，断点为当前发生故障的指令；对于“非法操作码”“除数为 0”等，因为无法通过异常处理程序恢复故障，因此不能回到原断点执行，必须终止进程的执行。

(2) 自陷 (Trap)

自陷也称陷阱或陷入，它是预先安排的一种“异常”事件，就像预先设定的“陷阱”一样。通常的做法是：事先在程序中用一条特殊指令或通过某种方式设定特殊控制标志来人为设置一个“陷阱”，当执行到被设置了“陷阱”的指令时，CPU 在执行完自陷指令后，自动根据不同“陷阱”类型进行相应的处理，然后返回到自陷指令的下一条指令执行。注意，当自陷指令是转移指令时，并不是返回到下一条指令执行，而是返回到转移目标指令执行。

在 x86 机器中，用于程序调试“断点设置”和单步跟踪的功能就是通过陷阱机制实现的。此外，系统调用指令、条件自陷指令（如 MIPS 中的 teq、teqi、tne、tnei 等）等都属于陷阱指令，执行到这些指令时，无条件或有条件地自动调出操作系统内核程序进行执行。

故障异常和自陷异常属于程序性异常（软件中断）。

(3) 终止 (Abort)

如果在执行指令的过程中发生了使计算机无法继续执行的硬件故障，如控制器出错、存储器校验错等，那么程序将无法继续执行，只能终止，此时，调出中断服务程序来重启系统。这种异常与故障和自陷不同，不是由特定指令产生的，而是随机发生的。

终止异常和外中断属于硬件中断。

2. 中断的分类

中断是指来自 CPU 外部、与 CPU 执行指令无关的事件引起的中断，包括 I/O 设备发出的 I/O 中断（如键盘输入、打印机缺纸等），或发生某种特殊事件（如用户按 Esc 键、定时器计数时间到）

等。外部 I/O 设备通过特定的中断请求信号线向 CPU 提出中断请求，CPU 每执行完一条指令就检查中断请求信号线，如果检测到中断请求，则进入中断响应周期。

中断可分为可屏蔽中断和不可屏蔽中断。

(1) 可屏蔽中断

指通过可屏蔽中断请求线 INTR 向 CPU 发出的中断请求。CPU 可以通过在中断控制器中设置相应的屏蔽字来屏蔽它或不屏蔽它，被屏蔽的中断请求将不被送到 CPU。

(2) 不可屏蔽中断

指通过专门的不可屏蔽中断请求线 NMI 向 CPU 发出的中断请求，通常是非常紧急的硬件故障，如电源掉电等。这类中断请求信号不可被屏蔽，以让 CPU 快速处理这类紧急事件。

中断和异常在本质上是一样的，但它们之间有以下两个重要的不同点：

- 1) “缺页”或“溢出”等异常事件是由特定指令在执行过程中产生的，而中断不和任何指令相关联，也不阻止任何指令的完成。
- 2) 异常的检测由 CPU 自身完成，不必通过外部的某个信号通知 CPU。对于中断，CPU 必须通过中断请求线获取中断源的信息，才能知道哪个设备发生了何种中断。

5.5.3 异常和中断响应过程

CPU 执行指令时，如果发生了异常或中断请求，必须进行相应的处理。从 CPU 检测到异常或中断事件，到调出相应的处理程序，整个过程称为异常和中断的响应。CPU 对异常和中断响应的过程可分为：关中断、保存断点和程序状态、识别异常和中断并转到相应的处理程序。

(1) 关中断

在保存断点和程序状态期间，不能被新的中断打断，因此要禁止响应新的中断，即关中断。通常通过设置“中断允许”(IF) 触发器来实现，若 IF 置为 1，则为开中断，表示允许响应中断；若 IF 置为 0，则表示关中断，表示不允许响应中断。

(2) 保存断点和程序状态

为了能在异常和中断处理后正确返回到被中断的程序继续执行，必须将程序的断点（返回地址）送到栈或特定寄存器中。通常保存在栈中，这是为了支持异常或中断的嵌套。

异常和中断处理后可能还要回到被中断的程序继续执行，被中断时的程序状态字寄存器 PSWR 的内容也需要保存在栈或特定寄存器中，在异常和中断返回时恢复到 PSWR 中。

(3) 识别异常和中断并转到相应的处理程序

异常和中断源的识别有软件识别和硬件识别两种方式。异常和中断源的识别方式不同，异常大多采用软件识别方式，而中断可以采用软件识别方式或硬件识别方式。

软件识别方式是指 CPU 设置一个异常状态寄存器，用于记录异常原因。操作系统使用一个统一的异常或中断查询程序，按优先级顺序查询异常状态寄存器，以检测异常和中断类型，先查询到的先被处理，然后转到内核中相应的处理程序。

硬件识别方式又称向量中断，异常或中断处理程序的首地址称为中断向量，所有中断向量都存放在中断向量表中。每个异常或中断都被指定一个中断类型号。在中断向量表中，类型号和中断向量一一对应，因而可以根据类型号快速找到对应的处理程序。

整个响应过程是不可被打断的。中断响应过程结束后，CPU 就从 PC 中取出中断服务程序的第一条指令开始执行，直至中断返回，这部分任务是由 CPU 通过执行中断服务程序完成的，整个中断处理过程是由软/硬件协同实现的。

5.5.4 本节习题精选

一、单项选择题

01. 以下关于“自陷”(Trap)异常的叙述中，错误的是()。
- “自陷”是人为预先设定的一种特定处理事件
 - 可由访管指令或自陷指令的执行进入“自陷”
 - 一定是出现某种异常情况才会发生“自陷”
 - “自陷”发生后CPU将进入操作系统内核程序执行
02. 指令执行结果出现异常而引起的中断是()。
- | | |
|-----------|-----------|
| A. I/O 中断 | B. 机器校验中断 |
| C. 程序性中断 | D. 外部中断 |
03. 主存故障引起的中断是()。
- | | | | |
|---------|----------|---------|--------|
| A. 故障异常 | B. 程序性中断 | C. 硬件中断 | D. 外中断 |
|---------|----------|---------|--------|
04. 以下关于异常和中断响应的叙述中，错误的是()。
- 异常事件检测由CPU在执行每一条指令的过程中进行
 - 中断请求检测由CPU在每条指令执行结束、取下一条指令之前进行
 - CPU检测到异常事件后所做的处理和检测到中断请求后所做的处理完全相同
 - CPU在中断响应时会关中断、保存断点和程序状态并转到相应的中断服务程序
05. 以下给出的事件中，无须异常处理程序进行处理的是()。
- | | | | |
|---------|-------------|---------|---------|
| A. 缺页故障 | B. Cache 缺失 | C. 地址越界 | D. 除数为0 |
|---------|-------------|---------|---------|
06. 【2015 统考真题】内部异常(内中断)可分为故障(fault)、陷阱(trap)和终止(abort)三类。下列有关内部异常的叙述中，错误的是()。
- 内部异常的产生与当前执行指令相关
 - 内部异常的检测由CPU内部逻辑实现
 - 内部异常的响应发生在指令执行过程中
 - 内部异常处理后返回到发生异常的指令继续执行
07. 【2016 统考真题】异常是指令执行过程中在处理器内部发生的特殊事件，中断是来自处理器外部的请求事件。下列关于中断或异常情况的叙述中，错误的是()。
- | | |
|-------------------|----------------|
| A. “访存时缺页”属于中断 | B. “整数除以0”属于异常 |
| C. “DMA 传送结束”属于中断 | D. “存储保护错”属于异常 |
08. 【2020 统考真题】下列关于“自陷”(Trap，也称陷阱)的叙述中，错误的是()。
- 自陷是通过陷阱指令预先设定的一类外部中断事件
 - 自陷可用于实现程序调试时的断点设置和单步跟踪
 - 自陷发生后CPU将转去执行操作系统内核相应程序
 - 自陷处理完成后返回到陷阱指令的下一条指令执行
09. 【2021 统考真题】异常事件在当前指令执行过程中进行检测，中断请求则在当前指令执行后进行检测。下列事件中，相应处理程序执行后，必须回到当前指令重新执行的是()。
- | | | | |
|---------|--------|-------------|----------|
| A. 系统调用 | B. 页缺失 | C. DMA 传送结束 | D. 打印机缺纸 |
|---------|--------|-------------|----------|

5.5.5 答案与解析

一、单项选择题

01. C

自陷是人为设定的特殊中断机制，不是出现某些异常情况而产生的。因此选项C错误。

02. C

指令执行结果出现的异常是程序性中断（软件中断），如运算溢出等，选项 C 正确。I/O 中断属于外部中断。机器校验中断属于终止类异常。

03. C

主存故障引起的中断是终止异常，属于硬件中断。故障异常是执行指令时产生的程序性故障（软件中断）。外中断主要指由外部设备引起的中断，通常用于信息的输入输出。

04. C

CPU 检测到异常事件后所做的处理和检测到中断请求后所做的处理基本是相同的，但有些地方可能不同。例如，对于故障类异常，因为其断点为发生故障时的指令地址，所以要重新计算 PC 值，而中断的断点为下条指令地址（即 PC 值），因此无须重新计算 PC 值。

05. B

缺页、地址越界和除数为 0 都是执行某条指令时发生的故障，需要调出操作系统内核中相应的异常处理程序来处理，而 Cache 缺失由 CPU 硬件实现，无须调出异常处理程序进行处理。

06. D

内部异常是指来自 CPU 内部产生的中断，如非法指令、地址非法、校验错、页面失效、运算溢出和除数为零等，以上都是在指令的执行过程中产生的，选项 A 正确。内部异常的检测是由 CPU 自身完成的，不必通过外部的某个信号通知 CPU，选项 B 正确。内部异常不能被屏蔽，一旦出现应立即处理，选项 C 正确。对于非法指令、除数为零等异常，无法通过异常处理程序恢复故障，因此不能回到原断点执行，必须终止进程的执行，因此选项 D 错误。

07. A

中断是指来自 CPU 执行指令以外事件，如设备发出的 I/O 结束中断，表示设备输入/输出已完成，希望处理机能够向设备发出下一个输入/输出请求，同时让完成输入/输出后的程序继续运行。异常也称内中断，指源自 CPU 执行指令内部的事件。选项 A 错误。

08. A

自陷是一种内部异常，选项 A 错误。在 x86 中，用于程序调试的“断点设置”功能是通过自陷机制实现的，选项 B 正确。执行到自陷指令时，无条件或有条件地自动调出操作系统内核程序进行执行，选项 C 正确。CPU 执行陷阱指令后，会自动地根据不同陷阱类型进行相应的处理，然后返回到陷阱指令的下一条指令执行，选项 D 正确。

09. B

外部中断都是在一条指令执行完成后（中断周期）才被检测并处理的。DMA 请求只请求总线的使用权，不影响当前指令的执行，不会导致被中断指令的重新执行。而缺页中断发生在取指或间址等指令执行过程之中，并且会阻塞整个指令。当缺页中断发生后，必须回到这条指令重新执行，以便重新访存。因此选择选项 B。

关注公众号【乘龙考研】
一手更新 稳定有保障

5.6 指令流水线

前面介绍的指令都是在单周期处理机中采用串行方法执行的，同一时刻 CPU 中只有一条指令在执行，因此各功能部件的使用率不高。现代计算机普遍采用指令流水线技术，同一时刻有多条指令在 CPU 的不同功能部件中并发执行，大大提高了功能部件的并行性和程序的执行效率。

5.6.1 指令流水线的基本概念

可从两方面提高处理机的并行性：①时间上的并行技术，将一个任务分解为几个不同的子阶段，每个阶段在不同的功能部件上并行执行，以便在同一时刻能够同时执行多个任务，进而提升系统性能，这种方法被称为流水线技术。②空间上的并行技术，在一个处理机内设置多个执行相同任务的功能部件，并让这些功能部件并行工作，这样的处理机被称为超标量处理机。

1. 指令流水的定义

一条指令的执行过程可分解为若干阶段，每个阶段由相应的功能部件完成。如果将各阶段视为相应的流水段，则指令的执行过程就构成了一条指令流水线。

假设一条指令的执行过程分为如下 5 个阶段（也称功能段或流水段）^①：

- 取指 (IF): 从指令存储器或 Cache 中取指令。
- 译码/读寄存器 (ID): 操作控制器对指令进行译码，同时从寄存器堆中取操作数。
- 执行/计算地址 (EX): 执行运算操作或计算地址。
- 访存 (MEM): 对存储器进行读写操作。
- 写回 (WB): 将指令执行结果写回寄存器堆。

把 $k+1$ 条指令的取指阶段提前到第 k 条指令的译码阶段，从而将第 $k+1$ 条指令的译码阶段与第 k 条指令的执行阶段同时进行，如图 5.16 所示。

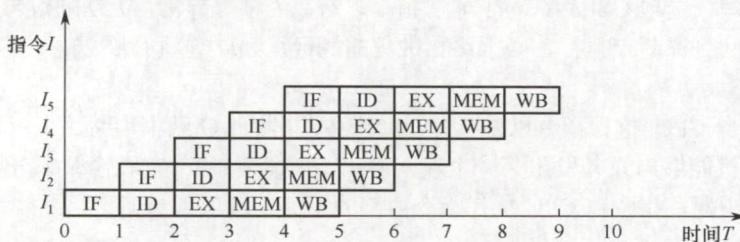


图 5.16 一个 5 段指令流水线

从图 5.16 看出，理想情况下，每个时钟周期都有一条指令进入流水线，每个时钟周期都有一条指令完成，每条指令的时钟周期数（即 CPI）都为 1。

流水线设计的原则是，指令流水段个数以最复杂指令所用的功能段个数为准；流水段的长度以最复杂的操作所花的时间为准。假设某条指令的 5 个阶段所花的时间分别如下。①取指：200ps；②译码：100ps；③执行：150ps；④访存：200ps；⑤写回：100ps，该指令的总执行时间为 750ps。按照流水线设计原则，每个流水段的长度为 200ps，所以每条指令的执行时间为 1ns，反而比串行执行时增加了 250ps。假设某程序中有 N 条指令，单周期处理机所用的时间为 $N \times 750\text{ps}$ ，而流水线处理机所用的时间为 $(N + 4) \times 200\text{ps}$ 。由此可见，流水线方式并不能缩短单条指令的执行时间，但对于整个程序来说，执行效率得到了大幅增高。

为了利于实现指令流水线，指令集应具有如下特征：

- 1) 指令长度应尽量一致，有利于简化取指令和指令译码操作。否则，取指令所花时间长短不一，使取指部件极其复杂，且也不利于指令译码。
- 2) 指令格式应尽量规整，尽量保证源寄存器的位置相同，有利于在指令未知时就可取寄存器操作数，否则须译码后才能确定指令中各寄存器编号的位置。

^① 不同的教材有不同的划分举例，本书参考了历年统考真题中的划分。

- 3) 采用 Load/Store 指令, 其他指令都不能访问存储器, 这样可把 Load/Store 指令的地址计算和运算指令的执行步骤规整在同一个周期中, 有利于减少操作步骤。
- 4) 数据和指令在存储器中“对齐”存放。这样, 有利于减少访存次数, 使所需数据在一个流水段内就能从存储器中得到。

2. 流水线的表示方法

通常用时空图来直观地描述流水线的执行情况, 如图 5.17 所示。

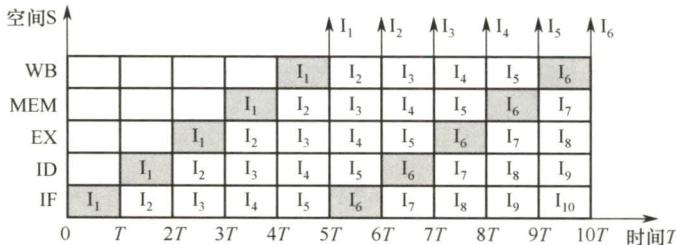


图 5.17 一个 5 段指令流水线时空图



在时空图中, 横坐标表示时间, 它被分割成长度相等的时间段 T ; 纵坐标为空间, 表示当前指令所处的功能部件。在图 5.17 中, 第一条指令 I_1 在时刻 0 进入流水线, 在时刻 $5T$ 流出流水线。第二条指令 I_2 在时刻 T 进入流水线, 在时刻 $6T$ 流出流水线。以此类推, 每隔一个时间 T 就有一条指令进入流水线, 从时刻 $5T$ 开始每隔一个时间 T 就有一条指令流出流水线。

从图中可看出, 在时刻 $10T$ 时, 流水线上便有 6 条指令流出。若采用串行方式执行, 在时刻 $10T$ 时, 只能执行 2 条指令, 可见使用流水线方式成倍地提高了计算机的速度。

只有大量连续任务不断输入流水线, 才能充分发挥流水线的性能, 而指令的执行正好是连续不断的, 非常适合采用流水线技术。对于其他部件级流水线, 如浮点运算流水线, 同样也仅适合于提升浮点运算密集型应用的性能, 对于单个运算是无法提升性能的。

5.6.2 流水线的基本实现

在单周期实现中, 这 5 个功能段是串连在一起的, 如图 5.18 所示。将程序计数器 (PC) 的值送入 IF 段取指令, 然后依次进入 ID、EX、MEM、WB 段。虽然不是所有指令都必须经历完整的 5 个阶段, 但只能以执行速度最慢的指令作为设计其时钟周期的依据, 单周期 CPU 的时钟频率取决于数据通路中的关键路径 (最长路径), 因此单周期 CPU 指令执行效率不佳。

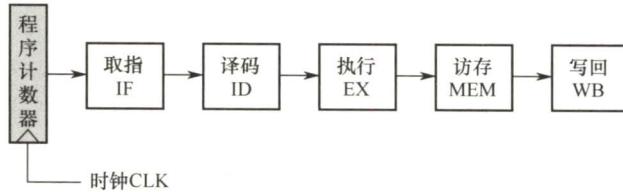


图 5.18 单周期处理机的逻辑构架图

1. 流水线的数据通路

一个 5 段流水线数据通路如图 5.19 所示。其中, IF 段包括程序计数器 (PC)、指令存储器、下条指令地址的计算逻辑; ID 段包括操作控制器、取操作数逻辑、立即数符号扩展模块; EX 段主要包括算术逻辑单元 (ALU)、分支地址计算模块; MEM 段主要包括数据存储器读写模块; WB 段主要包括寄存器写入控制模块。每个流水段后面都需要增加一个流水寄存器, 用于锁存本

段处理完成的数据和控制信号，以保证本段的执行结果能在下个时钟周期给下一流水段使用，图中增加了4个流水寄存器，并根据其所连接的功能段来命名。各种寄存器和数据存储器均采用统一时钟 CLK 进行同步，每来一个时钟，就会有一条新的指令进入流水线 IF 段；同时流水寄存器会锁存前段加工完成的数据和控制信号，为下一段的功能部件提供数据输入。

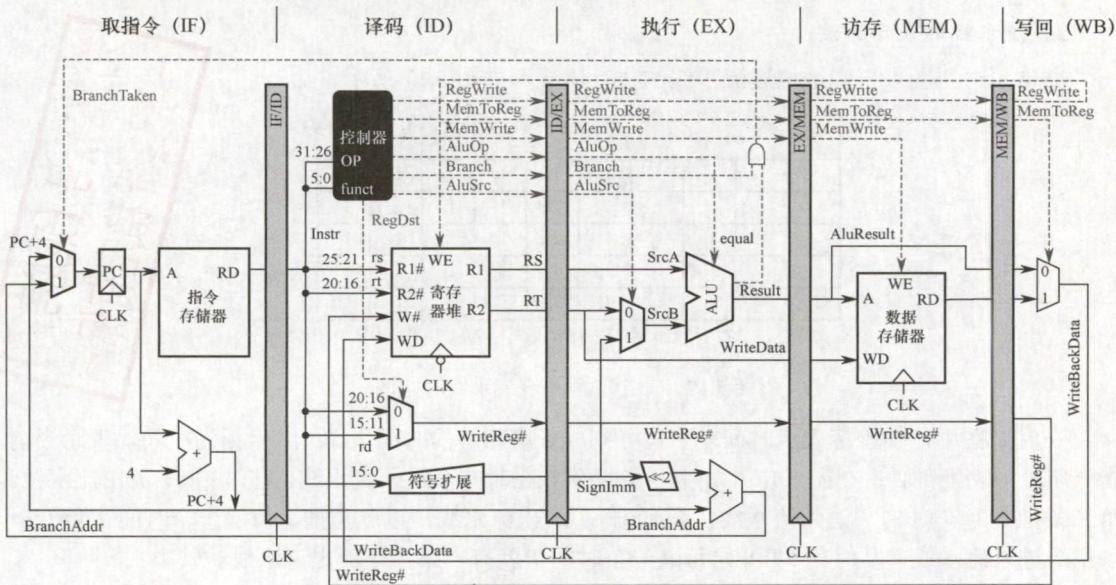


图 5.19 一个 5 段流水线数据通路（及控制信号）

不同流水寄存器锁存的数据不相同，如图 5.19 中的实线表示。IF/ID 流水寄存器需要锁存从指令存储器取出的指令字，以及 $PC + 4$ 的值；ID/EX 流水寄存器需要锁存从寄存器堆中取出的两个操作数 RS 和 RT（指令中两个操作数字段对应的寄存器值）与写寄存器编号 WriteReg#，以及立即数符号扩展的值、 $PC + 4$ 等后段可能用到的操作数；EX/MEM 流水寄存器需要锁存 ALU 运算结果、数据存储器待写入数据 WriteData、写寄存器编号 WriteReg# 等数据；MEM/WB 流水寄存器需要锁存 ALU 运算结果、数据存储器读出数据、写寄存器编号 WriteReg# 等数据。

2. 流水线的控制信号

上节描述了数据通过流水寄存器进行传递的情况。但是在某一时刻，每个流水段执行不同指令的某个阶段，每个流水段还需要正在执行指令的对应功能段的控制信号。

图 5.19 中的控制信号（虚线表示）如表 5.3 所示。控制信号的来源并不一致，如 IF 段的分支转跳信号 BranchTaken 来源于 EX 段，ID 段的 RegWrite 信号来源于 WB 段。其他控制信号通过控制器产生，由 ID 段负责译码生成控制信号，并分别在随后的各个时钟周期内使用。

表 5.3 图 5.19 中的控制信号分类

控制信号	位置	来源	功能说明
BranchTaken	IF	EX	分支跳转信号，为 1 表示跳转，由 EX 段的 Branch 信号与 equal 标志进行逻辑与生成
RegDst	ID	ID	写入目的寄存器选择，为 1 时目的寄存器为 rd 寄存器，为 0 时为 rt 寄存器
RegWrite	ID	WB	控制寄存器堆写操作，为 1 时数据需要写回寄存器堆中的指定寄存器
AluSrc	EX	EX	ALU 的第二输入选择控制，为 0 时输入寄存器 rt，为 1 时输入扩展后的立即数
AluOp	EX	EX	控制 ALU 进行不同运算，具体取值和位宽与 ALU 的设计有关
MemWrite	MEM	MEM	控制数据存储器写操作，为 0 时进行读操作，为 1 时进行写操作
MemToReg	WB	WB	为 1 时将数据存储器读出数据写回寄存器堆，否则将 ALU 运算结果写回

控制器的输入主要是 IF/ID 流水寄存器锁存的指令字中的 OP 字段，输出为 7 个控制信号，其中 RegDst 信号在 ID 段使用，其他 6 个后段使用的控制信号输出到 ID/EX 流水寄存器中并依次向后传递，以供后续各流水段使用。RegWrite 信号必须传递至 WB 段后才能反馈到 ID 段的寄存器堆的写入控制段 WE；条件分支译码信号 Branch 也需要传递到 EX 段，与 ALU 运算的标志 equal 信号进行逻辑与操作后，反馈到 IF 段控制多路选择器进行分支处理。

综上所述，每个流水寄存器中保存的信息包括：①后面流水段需要用到的所有数据信息，包括 PC + 4、指令、立即数、目的寄存器、ALU 运算结果、标志信息等，它们是前面阶段在数据通路中执行的结果；②前面传递过来的后面各流水段要用到的所有控制信号。

3. 流水线的执行过程

由于流水线的特殊结构，所有指令都需要完整经过流水线的各功能段，只不过某些指令在某些功能段内没有任何实质性的操作，只是等待一个时钟周期，这也就意味着单条指令的执行时间还是 5 个功能段时间延迟的总和。下面简单描述图 5.19 中各流水段的执行过程。

(1) 取指 (IF)

将 PC 值作为地址从指令寄存器中取出第一条指令字；并计算 PC + 4，送入 PC 输入端，以便在下一个时钟周期取下一条指令，这些功能由取指部件完成。取出的指令字通过 RD 输出端送入 IF/ID 流水寄存器，PC + 4 也要送入 IF/ID 流水寄存器，以备后续可能使用（如相对转移指令）。只要是后续功能段有可能要用到的数据和控制信号，都要向后传递。时钟到来时将更新后的 PC 值和指令字锁存到 IF/ID 流水寄存器中；本条指令 I₁ 进入 ID 段，IF 段取出下条指令 I₂。

(2) 译码/读寄存器 (ID)

由控制器根据 IF/ID 流水寄存器中的指令字生成后续各段需要的控制信号。对于 lw 访存指令，根据指令字中的 rs、rt 取出寄存器堆中的值 RS 和 RT；符号扩展单元会将指令字中的 16 位立即数符号扩展为 32 位；多路选择器根据指令字生成指令可能的写寄存器编号 WriteReg#。时钟到来时，这些数据和控制信号，连同顺序指令地址 PC + 4，都会锁存到 ID/EX 流水寄存器中；指令 I₁ 进入 EX 段，同时下条指令 I₂ 进入 ID 段，下下条指令 I₃ 进入 IF 段。

(3) 执行/计算地址 (EX)

EX 段功能由具体指令确定，不同指令经 ID 段译码后得到不同的控制信号。对于 lw 指令，EX 主要用来计算访存地址，将 ID/EX 流水寄存器中的 RS 值与符号扩展后的立即数相加得到的访存地址送入 EX/MEM 流水寄存器。EX 段可能还要计算分支地址，生成分支转跳信号 BranchTaken。RT 的值可能会在 MEM 段作为写入数据使用，所以 RT 会作为写入数据 WriteData 送入 EX/MEM 流水寄存器；ID/EX 流水寄存器中的写寄存器编号 WriteReg# 也将直接传送给 EX/MEM 流水寄存器。时钟到来后，这些数据和后段需要的控制信号都会锁存到 EX/MEM 流水寄存器中；指令 I₁ 进入 MEM 段，后续指令 I₂、I₃、I₄ 分别进入 EX、ID、IF 段。

(4) 访存 (MEM)

MEM 段的功能也由具体指令确定。对于 lw 指令，主要是根据 EX/MEM 流水寄存器中锁存的访存地址，写入数据和内存读写控制信号 MemWrite 对存储器进行读或写操作。EX/MEM 流水寄存器中的访存地址、WriteReg#、数据存储器读出的数据都会送入 MEM/WB 流水寄存器，以备后续可能使用。时钟到来后，这些数据和后段需要的控制信号都会锁存到 MEM/WB 流水寄存器中；指令 I₁ 进入 WB 段，后续指令 I₂、I₃、I₄、I₅ 分别进入 MEM、EX、ID、IF 段。

(5) 写回 (WB)

WB 段的功能也由具体指令确定。将 MEM/WB 流水寄存器中数据存储器读出的数据写回指定寄存器 WriteReg#。时钟到来时会完成数据写入寄存器，指令 I₁ 离开流水线。此时，指令 I₂ 进

入最后的 WB 段，指令 I_3 、 I_4 、 I_5 分别进入 MEM、EX、ID 段，指令 I_6 进入 IF 段。

5.6.3 流水线的冒险与处理

在指令流水线中，可能会遇到一些情况使得流水线无法正确执行后续指令而引起流水线阻塞或停顿，这种现象称为流水线冒险。根据导致冒险的原因不同主要有 3 种：结构冒险（资源冲突）、数据冒险（数据冲突）和控制冒险（控制冲突）。

1. 结构冒险

由于多条指令在同一时刻争用同一资源而形成的冲突，也称为资源冲突，即由硬件资源竞争造成的冲突，有以下两种解决办法：

- 1) 前一指令访存时，使后一条相关指令（以及其后续指令）暂停一个时钟周期。
- 2) 单独设置数据存储器和指令存储器，使取数和取指令操作各自在不同的存储器中进行。
事实上，现代计算机都引入了 Cache 机制，而 L1 Cache 通常采用数据 Cache 和指令 Cache 分离的方式，因而也就避免了资源冲突的发生。

2. 数据冒险

在一个程序中，下一条指令会用到当前指令计算出的结果，此时这两条指令发生数据冲突。当多条指令重叠处理时就会发生冲突，数据冒险可分为三类（结合综合题 3 理解）：

- 1) 写后读（Read After Write, RAW）相关：表示当前指令将数据写入寄存器后，下一条指令才能从该寄存器读取数据。否则，先读后写，读到的就是错误（旧）数据。
- 2) 读后写（Write After Read, WAR）相关：表示当前指令读出数据后，下一条指令才能写该寄存器。否则，先写后读，读到的就是错误（新）数据。
- 3) 写后写（Write After Write, WAW）相关：表示当前指令写入寄存器后，下一条指令才能写该寄存器。否则，下一条指令在当前指令之前写，将使寄存器的值不是最新值。

解决的办法有以下几种：

- 1) 把遇到数据相关的指令及其后续指令都暂停一至几个时钟周期，直到数据相关问题消失后再继续执行，可分为硬件阻塞（stall）和软件插入“NOP”指令两种方法。
- 2) 设置相关专用通路，即不等前一条指令把计算结果写回寄存器组，下一条指令也不再读寄存器组，而直接把前一条指令的 ALU 的计算结果作为自己的输入数据开始计算过程，使本来需要暂停的操作变得可以继续执行，这称为数据旁路技术。
- 3) 通过编译器对数据相关的指令编译优化的方法，调整指令顺序来解决数据相关。

3. 控制冒险

指令通常是顺序执行的，但是在遇到改变指令执行顺序的情况，例如执行转移、调用或返回等指令时，会改变 PC 值，会造成断流，从而引起控制冒险。解决的办法有以下几种：

- 1) 对转移指令进行分支预测，尽早生成转移目标地址。分支预测分为简单（静态）预测和动态预测。静态预测总是预测条件不满足，即继续执行分支指令的后续指令。动态预测根据程序执行的历史情况，进行动态预测调整，有较高的预测准确率。
- 2) 预取转移成功和不成功两个控制流方向上的目标指令。
- 3) 加快和提前形成条件码。
- 4) 提高转移方向的猜准率。

注意：Cache 缺失的处理过程也会引起流水线阻塞。在不过多增加硬件成本的情况下，如何尽可能地提高指令流水线的运行效率是选用指令流水线技术必须解决的关键问题。

5.6.4 流水线的性能指标

1. 流水线的吞吐率

流水线的吞吐率是指在单位时间内流水线所完成的任务数量，或输出结果的数量。

流水线吞吐率 (TP) 的最基本公式为

$$TP = \frac{n}{T_k}$$

式中， n 是任务数， T_k 是处理完 n 个任务所用的总时间。设 k 为流水段的段数， Δt 为时钟周期。在输入流水线中的任务连续的理想情况下，一条 k 段流水线能在 $k + n - 1$ 个时钟周期内完成 n 个任务。得出流水线的吞吐率为

$$TP = \frac{n}{(k + n - 1)\Delta t}$$

连续输入的任务数 $n \rightarrow \infty$ 时，得最大吞吐率为 $TP_{\max} = 1/\Delta t$ 。

2. 流水线的加速比

完成同样一批任务，不使用流水线与使用流水线所用的时间之比。

流水线加速比 (S) 的基本公式为

$$S = \frac{T_0}{T_k}$$

式中， T_0 表示不使用流水线的总时间； T_k 表示使用流水线的总时间。一条 k 段流水线完成 n 个任务所需的时间为 $T_k = (k + n - 1)\Delta t$ 。顺序执行 n 个任务时，所需的总时间为 $T_0 = kn\Delta t$ 。将 T_0 和 T_k 值代入上式，得出流水线的加速比为

$$S = \frac{kn\Delta t}{(k + n - 1)\Delta t} = \frac{kn}{k + n - 1}$$

连续输入的任务数 $n \rightarrow \infty$ 时，得最大加速比为 $S_{\max} = k$ 。

5.6.5 高级流水线技术

有两种增加指令级并行的策略：一种是多发射技术，它通过采用多个内部功能部件，使流水线功能段能同时处理多条指令，处理机一次可以发射多条指令进入流水线执行；另一种是超流水线技术，它通过增加流水线级数来使更多的指令同时在流水线中重叠执行。

1. 超标量流水线技术

超标量流水线技术也称动态多发射技术，每个时钟周期内可并发多条独立指令，以并行操作方式将两条或多条指令编译并执行，为此需配置多个功能部件，如图 5.20 所示。在简单的超标量 CPU 中，指令是按顺序发射执行的。为了更好地提高并行性能，多数超标量 CPU 都结合动态流水线调度技术，通过动态分支预测等手段，指令不按顺序执行，这种执行方式称为乱序执行。

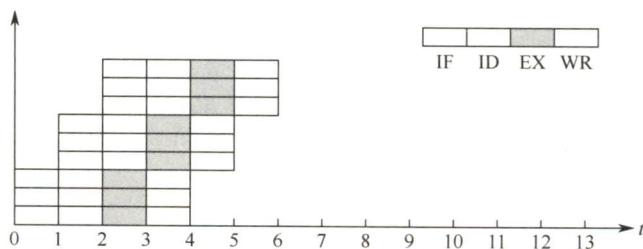


图 5.20 超标量流水线技术

2. 超长指令字技术

超长指令字技术也称静态多发射技术，由编译程序挖掘出指令间潜在的并行性，将多条能并行操作的指令组合成一条具有多个操作码字段的超长指令字（可达几百位），为此需要采用多个处理部件。

3. 超流水线技术

如图 5.21 所示，流水线功能段划分得越多，时钟周期就越短，指令吞吐率也就越高，因此超流水线技术是通过提高流水线主频的方式来提升流水线性能的。但是，流水线级数越多，用于流水寄存器的开销就越大，因而流水线级数是有限制的，并不是越多越好。

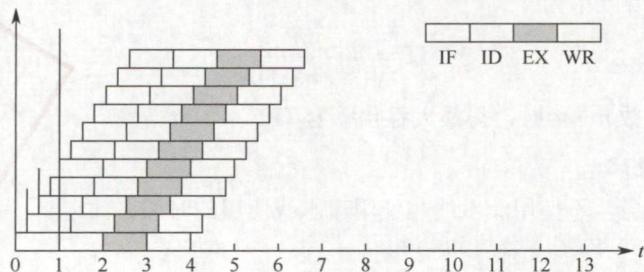


图 5.21 超流水线技术

超流水线 CPU 在流水线充满后，每个时钟周期还是执行一条指令， $CPI = 1$ ，但其主频更高；多发射流水线 CPU 每个时钟周期可以处理多条指令， $CPI < 1$ ，相对而言，多发射流水线成本更高，控制更复杂。

5.6.6 本节习题精选

一、单项选择题

01. 下列关于流水 CPU 基本概念的描述中，正确的是（ ）。
 - A. 流水 CPU 是以空间并行性为原理构造的处理器
 - B. 流水 CPU 一定是 RISC 机器
 - C. 流水 CPU 一定是多媒体 CPU
 - D. 流水 CPU 是一种非常经济而实用的时间并行技术
02. 下列关于超标量流水线的描述中，不正确的是（ ）。
 - A. 在一个时钟周期内一条流水线可执行一条以上的指令
 - B. 一条指令分为多段指令由不同电路单元完成
 - C. 超标量通过内置多条流水线来同时执行多个处理器，其实质是以空间换取时间
 - D. 超标量流水线是指运算操作并行
03. 下列关于动态流水线的描述中，正确的是（ ）。
 - A. 动态流水线是指在同一时间内，当某些段正在实现某种运算时，另一些段却正在进行另一种运算，这样对提高流水线的效率很有好处，但会使流水线控制变得很复杂
 - B. 动态流水线是指运算操作并行
 - C. 动态流水线是指指令步骤并行
 - D. 动态流水线是指程序步骤并行
04. 流水 CPU 是由一系列称为“段”的处理线路组成的。一个 m 段流水线稳定的 CPU 的

- 吞吐能力，与 m 个并行部件的 CPU 的吞吐能力相比，()。
- 具有同等水平的吞吐能力
 - 不具备同等水平的吞吐能力
 - 吞吐能力大于前者的吞吐能力
 - 吞吐能力小于前者的吞吐能力
05. 设指令由取指、分析、执行 3 个子部件完成，并且每个子部件的时间均为 Δt ，若采用常规标量单流水线处理机（即处理机的度为 1），连续执行 12 条指令，共需 ()。
- $12 \Delta t$
 - $14 \Delta t$
 - $16 \Delta t$
 - $18 \Delta t$
06. 若采用度为 4 的超标量流水线处理机，连续执行上述 20 条指令，只需 ()。
- $3 \Delta t$
 - $5 \Delta t$
 - $7 \Delta t$
 - $9 \Delta t$
07. 设指令流水线把一条指令分为取指、分析、执行 3 部分，且 3 部分的时间分别是 $t_{\text{取指}} = 2\text{ns}$ ，
 $t_{\text{分析}} = 2\text{ns}$ ， $t_{\text{执行}} = 1\text{ns}$ ，则 100 条指令全部执行完毕需 ()。
- 163ns
 - 183ns
 - 193ns
 - 203ns
08. 设指令由取指、分析、执行 3 个子部件完成，并且每个子部件的时间均为 t ，若采用常规标量单流水线处理机，连续执行 8 条指令，则该流水线的加速比为 ()。
- 3
 - 2
 - 3.4
 - 2.4
09. 指令流水线中出现数据相关时流水线将受阻，() 可解决数据相关问题。
- 增加硬件资源
 - 采用旁路技术
 - 采用分支预测技术
 - 以上都可以
10. 下面有关控制相关的描述中，错误的是 ()。
- 条件转移指令可能引起控制相关
 - 在分支指令加入若干空操作可以避免控制冒险
 - 采用转发（旁路）技术，可以解决部分控制相关
 - 通过编译器调整指令执行顺序可解决部分控制冒险
11. 关于流水线技术的说法中，错误的是 ()。
- 超标量技术需要配置多个功能部件和指令译码电路等
 - 与超标量技术和超流水线技术相比，超长指令字技术对优化编译器要求更高，而无其他硬件要求
 - 流水线按序流动时，在 RAW、WAR 和 WAW 中，只可能出现 RAW 相关
 - 超流水线技术相当于将流水线再分段，从而提高每个周期内功能部件的使用次数
12. 【2009 统考真题】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 ()。
- 90ns
 - 80ns
 - 70ns
 - 60ns
13. 【2010 统考真题】下列不会引起指令流水线阻塞的是 ()。
- 数据旁路
 - 数据相关
 - 条件转移
 - 资源冲突
14. 【2013 统考真题】某 CPU 主频为 1.03GHz，采用 4 级指令流水线，每个流水段的执行需要 1 个时钟周期。假定 CPU 执行了 100 条指令，在其执行过程中，没有发生任何流水线阻塞，此时流水线的吞吐率为 ()。
- 0.25×10^9 条指令/秒
 - 0.97×10^9 条指令/秒



C. 1.0×10^9 条指令/秒 D. 1.03×10^9 条指令/秒

15. 【2016 统考真题】在无转发机制的五段基本流水线（取指、译码/读寄存器、运算、访存、写回寄存器）中，下列指令序列存在数据冒险的指令对是（ ）。

I1: add R1, R2, R3; (R2) + (R3) → R1
 I2: add R5, R2, R4; (R2) + (R4) → R5
 I3: add R4, R5, R3; (R5) + (R3) → R4
 I4: add R5, R2, R6; (R2) + (R6) → R5

A. I1 和 I2 B. I2 和 I3 C. I2 和 I4 D. I3 和 I4

16. 【2017 统考真题】下列关于超标量流水线特性的叙述中，正确的是（ ）。

I. 能缩短流水线功能段的处理时间
 II. 能在一个时钟周期内同时发射多条指令
 III. 能结合动态调度技术提高指令执行并行性
 A. 仅 II B. 仅 I、III C. 仅 II、III D. I、II 和 III

17. 【2017 统考真题】下列关于指令流水线数据通路的叙述中，错误的是（ ）。

A. 包含生成控制信号的控制部件
 B. 包含算术逻辑运算部件（ALU）
 C. 包含通用寄存器组和取指部件
 D. 由组合逻辑电路和时序逻辑电路组合而成

18. 【2018 统考真题】若某计算机最复杂指令的执行需要完成 5 个子功能，分别由功能部件 A~E 实现，各功能部件所需时间分别为 80ps、50ps、50ps、70ps 和 50ps，采用流水线方式执行指令，流水段寄存器延时为 20ps，则 CPU 时钟周期至少为（ ）。

A. 60ps B. 70ps C. 80ps D. 100ps

19. 【2019 统考真题】在采用“取指、译码/取数、执行、访存、写回”5 段流水线的处理器中，执行如下指令序列，其中 s0、s1、s2、s3 和 t2 表示寄存器编号。

I1: add s2, s1, s0 //R[s2] ← R[s1] + R[s0]
 I2: load s3, 0(t2) //R[s3] ← M[R[t2]+0]
 I3: add s2, s2, s3 //R[s2] ← R[s2] + R[s3]
 I4: store s2, 0(t2) //M[R[t2]+0] ← R[s2]

下列指令对中，不存在数据冒险的是（ ）。

A. I1 和 I3 B. I2 和 I3 C. I2 和 I4 D. I3 和 I4

20. 【2020 统考真题】下列给出的处理器类型中，理想情况下，CPI 为 1 的是（ ）。

I. 单周期 CPU II. 多周期 CPU III. 基本流水线 CPU IV. 超标量流水线 CPU
 A. 仅 I、II B. 仅 I、III C. 仅 II、IV D. 仅 III、IV

二、综合应用题

01. 现有四级流水线，分别完成取指令、指令译码并取数、运算、回写四步操作，假设完成各部操作的时间依次为 100ns、100ns、80ns 和 50ns。试问：

1) 流水线的操作周期应设计为多少？

2) 若相邻两条指令如下，发生数据相关（假设在硬件上不采取措施），试分析第二条指令要推迟多少时间进行才不会出错。

```
ADD R1, R2, R3      # R2+R3 -> R1
SUB R4, R1, R5      # R1-R5 -> R4
```

- 3) 若在硬件设计上加以改进, 至少需要推迟多少时间?
02. 假设指令流水线分为取指 (IF)、译码 (ID)、执行 (EX)、回写 (WB) 4 个过程, 共有 10 条指令连续输入此流水线。
- 1) 画出指令周期流程图。
 - 2) 画出非流水线时空图。
 - 3) 画出流水线时空图。
 - 4) 假设时钟周期为 100ns, 求流水线的实际吞吐量 (单位时间执行完毕的指令数)。
03. 流水线中有 3 类数据相关冲突: 写后读 (RAW) 相关; 读后写 (WAR) 相关; 写后写 (WAW) 相关。判断以下 3 组指令各存在哪种类型的数据相关。

第一组 I1	ADD R1, R2, R3	$(R2 + R3) \rightarrow R1$
I2	SUB R4, R1, R5	$(R1 - R5) \rightarrow R4$
第二组 I3	STA M(x), R3	$(R3) \rightarrow M(x)$, M(x) 是存储器单元
I4	ADD R3, R4, R5	$(R4 + R5) \rightarrow R3$
第三组 I5	MUL R3, R1, R2	$(R1) \times (R2) \rightarrow R3$
I6	ADD R3, R4, R5	$(R4 + R5) \rightarrow R3$

04. 某台单流水线多操作部件处理机, 包含有取指、译码、执行 3 个功能段, 在该机上执行以下程序。取指和译码功能段各需要 1 个时钟周期, MOV 操作需要 2 个时钟周期, ADD 操作需要 3 个时钟周期, MUL 操作需要 4 个时钟周期, 每个操作都在第一个时钟周期接收数据, 在最后一个时钟周期把结果写入通用寄存器。

K:	MOV R1, R0	$(R0) \rightarrow R1$
K + 1:	MUL R0, R1, R2	$(R1) \times (R2) \rightarrow R0$
K + 2:	ADD R0, R2, R3	$(R2) + (R3) \rightarrow R0$

- 1) 画出流水线功能段结构图。
 - 2) 画出指令执行过程流水线的时空图。
05. 【2012 统考真题】某 16 位计算机中, 有符号整数用补码表示, 数据 Cache 和指令 Cache 分离。下表给出了指令系统中的部分指令格式, 其中 Rs 和 Rd 表示寄存器, mem 表示存储单元地址, (x) 表示寄存器 x 或存储单元 x 的内容。

表指令系统中部分指令格式

名 称	指令的汇编格式	指 令 功 能
加法指令	ADD Rs, Rd	$(Rs) + (Rd) \rightarrow Rd$
算术/逻辑左移	SHL Rd	$2 * (Rd) \rightarrow Rd$
算术右移	SHR Rd	$(Rd) / 2 \rightarrow Rd$
取数指令	LOAD Rd, mem	$(mem) \rightarrow Rd$
存数指令	STORE Rs, mem	$(Rs) \rightarrow mem$

该计算机采用 5 段流水方式执行指令, 各流水段分别是取指 (IF)、译码/读寄存器 (ID)、执行/计算有效地址 (EX)、访问存储器 (M) 和结果写回寄存器 (WB), 流水线采用“按序发射, 按序完成”方式, 未采用转发技术处理数据相关, 且同一寄存器的读和写操作不能在同一个时钟周期内进行。请回答下列问题:

- 1) 若 int 型变量 x 的值为 -513, 存放在寄存器 R1 中, 则执行 “SHR R1” 后, R1 中的内容是多少 (用十六进制表示)?
- 2) 若在某个时间段中, 有连续的 4 条指令进入流水线, 在其执行过程中未发生任何阻塞, 则执行这 4 条指令所需的时钟周期数为多少?

3) 若高级语言程序中某赋值语句为 $x = a + b$, x 、 a 和 b 均为 int 型变量, 它们的存储单元地址分别表示为 $[x]$ 、 $[a]$ 和 $[b]$ 。该语句对应的指令序列及其在指令流中的执行过程如下所示。

```
I1 LOAD R1, [a]
I2 LOAD R2, [b]
I3 ADD R1, R2
I4 STORE R2, [x]
```

指令	时钟 1	2	3	4	5	6	7	8	9	10	11	12	13	14
I ₁	IF	ID	EX	M	WB									
I ₂		IF	ID	EX	M	WB								
I ₃			IF				ID	EX	M	WB				
I ₄							IF				ID	EX	M	WB

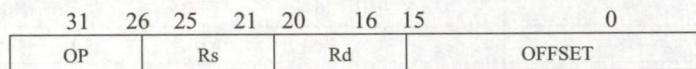
则这 4 条指令执行过程中 I₃ 的 ID 段和 I₄ 的 IF 段被阻塞的原因各是什么?

4) 若高级语言程序中某赋值语句为 $x = x * 2 + a$, x 和 a 均为 unsigned int 类型的变量, 它们的存储单元地址分别表示为 $[x]$ 、 $[a]$, 则执行这条语句至少需要多少个时钟周期? 要求模仿上图画出这条语句对应的指令序列及其在流水线中的执行过程示意图。

06.【2014 统考真题】某程序中有循环代码段 P: “for(int i = 0; i < N; i++) sum += A[i];”。假设编译时变量 sum 和 i 分别分配在寄存器 R1 和 R2 中。常量 N 在寄存器 R6 中, 数组 A 的首地址在寄存器 R3 中。程序段 P 的起始地址为 08048100H, 对应的汇编代码和机器代码如下表所示。

编 号	地 址	机器代码	汇编代码	注 释
1	08048100H	00022080H	loop: sll R4, R2, 2	(R2) << 2 → R4
2	08048104H	00083020H	add R4, R4, R3	(R4) + (R3) → R4
3	08048108H	8C850000H	load R5, 0(R4)	((R4) + 0) → R5
4	0804810CH	00250820H	add R1, R1, R5	(R1) + (R5) → R1
5	08048110H	20420001H	add R2, R2, 1	(R2) + 1 → R2
6	08048114H	1446FFFFH	bne R2, R6, loop	if(R2) != (R6) goto loop

执行上述代码的计算机 M 采用 32 位定长指令字, 其中分支指令 bne 采用如下格式:



OP 为操作码; Rs 和 Rd 为寄存器编号; OFFSET 为偏移量, 用补码表示。

请回答下列问题, 并说明理由。

- 1) M 的存储器编址单位是什么?
- 2) 已知 sll 指令实现左移功能, 数组 A 中每个元素占多少位?
- 3) 表中 bne 指令的 OFFSET 字段的值是多少? 已知 bne 指令采用相对寻址方式, 当前 PC 内容为 bne 指令地址, 通过分析表中指令地址和 bne 指令内容, 推断 bne 指令的转移目标地址计算公式。
- 4) 若 M 采用如下“按序发射、按序完成”的 5 级指令流水线: IF (取值)、ID (译码及取数)、EXE (执行)、MEM (访存)、WB (写回寄存器), 且硬件不采取任何转发措

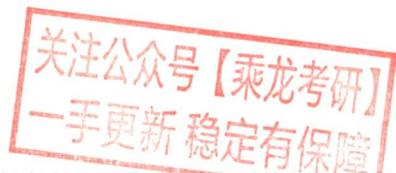
施，分支指令的执行均引起 3 个时钟周期的阻塞，则 P 中哪些指令的执行会由于数据相关而发生流水线阻塞？哪条指令的执行会发生控制冒险？为什么指令 1 的执行不会因为与指令 5 的数据相关而发生阻塞？

07. 【2014 统考真题】假设对于上题中的计算机 M 和程序 P 的机器代码，M 采用页式虚拟存储管理；P 开始执行时， $(R1) = (R2) = 0$, $(R6) = 1000$ ，其机器代码已调入主存但不在 Cache 中；数组 A 未调入主存，且所有数组元素在同一页，并存储在磁盘的同一个扇区。请回答下列问题并说明理由。

- 1) P 执行结束时，R2 的内容是多少？
- 2) M 的指令 Cache 和数据 Cache 分离。若指令 Cache 共有 16 行，Cache 和主存交换的块大小为 32B，则其数据区的容量是多少？若仅考虑程序段 P 的执行，则指令 Cache 的命中率为多少？
- 3) P 在执行过程中，哪条指令的执行可能发生溢出异常？哪条指令的执行可能产生缺页异常？对于数组 A 的访问，需要读磁盘和 TLB 至少各多少次？

5.6.7 答案与解析

一、单项选择题



01. D

空间并行即资源重复，主要指多个功能部件共同执行同一任务的不同部分，典型的如多处理器系统。时间并行即时间重叠，让多个功能部件在时间上相互错开，轮流重叠执行不同任务的相同部分，因此流水 CPU 利用的是时间并行性，因此选项 A 错误。RISC 都采用流水线技术，以提高资源利用率。但反过来并不成立，因为大部分 CISC 同样采用了流水线技术，因此选项 B 错误。流水 CPU 和多媒体 CPU 无必然联系，因此选项 C 错误。

02. D

超标量流水线是指在一个时钟周期内一条流水线可执行一条以上的指令，因此选项 A 正确。一条指令分为多段指令，由不同电路单元完成，因此选项 B 正确。超标量通过内置多条流水线来同时执行多个处理器，其实质是以空间换取时间，因此选项 C 正确。

03. A

动态流水线是相对于静态流水线而言的，静态流水线上下段连接方式固定，而动态流水线的连接方式是可变的。

04. A

吞吐能力是指单位时间内完成的指令数。 m 段流水线在第 m 个时钟周期后，每个时钟周期都可以完成一条指令；而 m 个并行部件在 m 个时钟周期后能完成全部的 m 条指令，等价于平均每个时钟周期完成一条指令。因此两者的吞吐能力等同。

05. B

单流水线处理机执行 12 条指令的时间为 $(3 + (12 - 1))\Delta t = 14\Delta t$ 。

06. C

这个超标量流水线处理机可以发送 4 条指令，所以执行指令的时间为 $(3 + (20 - 4)/4)\Delta t = 7\Delta t$ 。

07. D

每个功能段的时间设定为取指、分析和执行部分的最长时间 2ns，第一条指令在第 5ns 时执行完毕，其余的 99 条指令每隔 2ns 执行完一条，所以 100 条指令全部执行完毕所需的时间为 $(5 +$

$99 \times 2) \text{ns} = 203 \text{ns}$ 。

08. D

采用流水线时，第一条指令完成的时间是 $3t$ ，以后每经过 t 都有一条指令完成，因此共需要的时间为 $3t + (8 - 1)t = 10t$ ；而不采用流水线时，完成 8 条指令总共需要的时间为 $8 \times 3t = 24t$ ，所以流水线的加速比 $= 24t/10t = 2.4$ 。

09. B

处理数据相关问题有两种方法：一种是暂停相关指令的执行，即暂停流水线，直到能够正确读出寄存器操作数为止；另一种是采用专门的数据通路，直接把结果送到 ALU 的输入端，这种方法称为旁路技术。

10. C

采用转发（旁路）技术，可以解决的是数据相关，选项 C 错误。

11. B

要实现超标量技术，要求处理机中配置多个功能部件和指令译码电路，以及多个寄存器和总线，以便能实现同时执行多个操作，选项 A 正确；超长指令字技术对 Cache 的容量要求更大，因为需要执行的指令长度也许会很长，选项 B 错误；流水线按序流动，肯定不会出现先读后写（WAR）和写后写（WAW）相关。只可能出现没有等到上一条指令写入，当前指令就去读寄存器的错误（此时可采用旁路相关来解决），选项 C 正确。由超流水线技术的定义易知选项 D 正确。

12. A

时钟周期应以各功能段的最长执行时间为准，否则用时较长的流水段的功能将不能正确完成，因此应选 90ns。

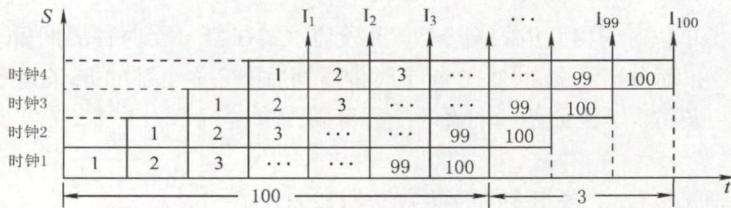
13. A

采用流水线方式，相邻或相近的两条指令可能会因为存在某种关联，后一条指令不能按照原指定的时钟周期运行，从而使流水线断流。有三种相关可能引起指令流水线阻塞：①结构相关，又称资源相关；②数据相关；③控制相关，主要由转移指令引起。

数据旁路技术的主要思想是，直接将执行结果送到其他指令所需要的地方，使流水线不发生停顿，因此不会引起流水线阻塞。

14. C

采用 4 级流水执行 100 条指令，在执行过程中共用 $4 + (100 - 1) = 103$ 个时钟周期，如下图所示。CPU 的主频是 1.03GHz，即每秒有 1.03G 个时钟周期。流水线的吞吐率为 $1.03G \times 100/103 = 1.0 \times 10^9$ 条指令/秒。



15. B

数据冒险即数据相关，指在一个程序中存在必须等前一条指令执行完才能执行后一条指令的情况，此时这两条指令即为数据相关。当多条指令重叠处理时就会发生冲突。首先这两条指令发生写后读相关，且两条指令在流水线中的执行情况（发生数据冒险）如下表所示。

时钟 指令	1	2	3	4	5	6	7
I2	取指	译码/读寄存器	运算	访存	写回		
I3		取指	译码/读寄存器	运算	访存	写回	

指令 I2 在时钟 5 时将结果写入寄存器 (R5)，但指令 I3 在时钟 3 时读寄存器 (R5)。本来指令 I2 应先写入 R5，指令 I3 后读 R5，结果变成指令 I3 先读 R5，指令 I2 后写入 R5，因而发生数据冲突。

16. C

超标量是指在 CPU 中有一条以上的流水线，并且每个时钟周期内可以完成一条以上的指令，其实质是以空间换时间。I 错误，它不影响流水线功能段的处理时间；II、III 正确。选择选项 C。

17. A

数据在功能部件之间传送的路径被称为数据通路，包括数据通路上流经的部件，如程序计数器、ALU、通用寄存器、状态寄存器、异常和中断处理逻辑等。数据通路由控制部件控制，控制部件根据每条指令功能的不同生成对数据通路的控制信号。因此，不包括控制部件。

18. D

指令流水线的每个流水段时间单位为时钟周期，题中指令流水线的指令需要用到 A~E 五个部件，所以每个流水段时间应取最大部件时间 80ps，此外还有寄存器延时 20ps，则 CPU 时钟周期至少是 100ps。答案是选项 D。

19. C

画出这四条指令在流水线中执行的过程如下图所示。

指令	1	2	3	4	5	6	7	8	9	10	11	12	13	14
add s2, s1, s0	取指	译码/取数	执行	访存	写回									
load s3, 0(t2)		取指	译码/取数	执行	访存	写回								
add s2, s2, s3			取指				译码/取数	执行	访存	写回				
store s2, 0(t2)							取指				译码/取数	执行	访存	写回

数据冒险即数据相关，指在程序中存在必须等前一条指令执行完才能执行后一条指令的情况，此时这两条指令即为数据相关。其中 I1 和 I3、I2 和 I3、I3 和 I4 均发生了写后读相关，因此必须等相关的前一条指令执行完才能执行后一条指令。只有 I2 和 I4 不存在数据冒险，答案是选项 C。

20. B

CPI 表示执行指令所需的时钟周期数。对于一个程序或一台机器来说，其 CPI 是指执行该程序或机器指令集中的所有指令所需的平均时钟周期数。对于单周期 CPU，令指令周期 = 时钟周期，CPI = 1，I 正确。对于多周期 CPU，CPU 的执行过程分成几个阶段，每个阶段用一个时钟去完成，每种指令所用的时钟数可以不同，CPI > 1，II 错误。对于基本流水线 CPU，让每个时钟周期流出一条指令，CPI = 1，III 正确。超标量流水线 CPU 在每个时钟周期内并发执行多条独立的指令，每个时钟周期流出多条指令，CPI < 1，IV 错误。

二、综合应用题

01. 【解答】

1) 流水线操作的时钟周期 T 应按四步操作中的最长时间来考虑，所以 $T = 100\text{ns}$ 。

2) 分析如下：

首先该两条指令发生写后读相关，且两条指令在流水线中的执行情况如下表所示。

关注公众号【乘龙考研】
一手更新 稳定有保障

时钟 指令 \	1	2	3	4	5	6	7
ADD	取指	指令译码并取数	运算	写回			
SUB		取指	指令译码并取数	运算	写回		

ADD 指令在时钟 4 时将结果写入寄存器堆(R1)，但 SUB 指令在时钟 3 时读寄存器堆(R1)。本来 ADD 指令应先写入 R1，SUB 指令后读 R1，结果变成 SUB 指令先读 R1，ADD 指令后写入 R1，因而发生数据冲突。若硬件上不采取措施，第二条指令 SUB 至少应推迟两个时钟周期 ($2 \times 100\text{ns}$)，即 SUB 指令中的指令译码并取数周期应在 ADD 指令的写回周期之后才能保证不会出错，如下表所示。

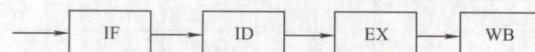
时钟 指令 \	1	2	3	4	5	6	7
ADD	取指	指令译码并取数	运算	写回			
SUB		取指			指令译码并取数	运算	写回

- 3) 若在硬件上加以改进，可以只延迟一个时钟周期 (100ns)。因为在 ADD 指令中，运算周期已得到结果。可以通过数据旁路技术在运算结果一得到时，就将结果快速地送入寄存器 R1，而不需要等到写回周期完成。流水线中的执行情况如下表所示。

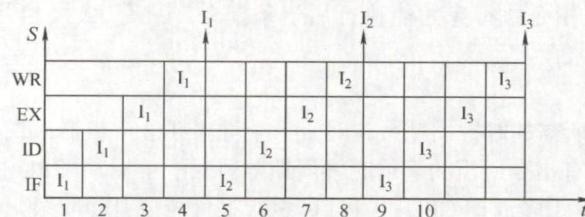
时钟 指令 \	1	2	3	4	5	6	7
ADD	取指	指令译码并 取数	运算 (并采用数据旁路技 术写入寄存器 R1)	写回			取指
SUB		取指		指令译码并取数	运算	写回	

02. 【解答】

- 1) 因为指令周期包括 IF、ID、EX、WB 四个子过程，因此其指令周期流程图如下图所示。



- 2) 假设一个时间单位为一个时钟周期，则每隔 4 个周期才有一个输出结果。非流水线的时空图如下图所示。



- 3) 第一条指令出结果需要 4 条指令周期。流水线满载时，以后每个时钟周期都可输出一个结果，即执行完一条指令，如下图所示。



4) 由上图可知，在13个时钟周期结束时，CPU执行完10条指令，因此实际吞吐率(T)为

$$T = \frac{10}{100\text{ns} \times 13} \approx 7700000 \text{条/s}$$

03. 【解答】

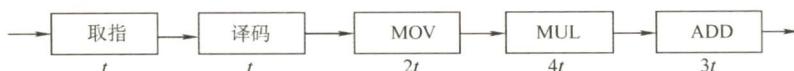
第一组指令中，I1指令运算结果应先写入R1，然后在I2指令中读出R1的内容。由于I2指令进入流水线，变成I2指令在I1指令写入R1前就读出R1的内容，发生RAW相关。

第二组指令中，I3指令应先读出R3的内容并存入存储单元M(x)，然后在I4指令中将运算结果写入R3。但由于I4指令进入流水线，变成I4指令在I3指令读出R3的内容前就写入R3，发生WAR相关。

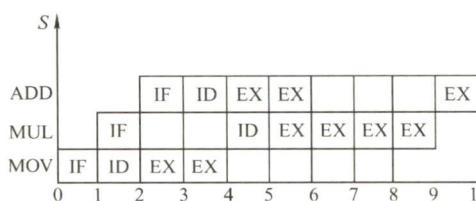
第三组指令中，若I6指令的加法运算完成时间早于I5指令的乘法运算时间，变成指令I6在指令I5写入R3前就写入R3，导致R3的内容错误，发生WAW相关。

04. 【解答】

1) 流水线功能段结构图如下图所示。



2) 三条指令存在数据相关，采用后推法得到指令执行过程流水线的时空图如下图所示(IF取指、ID译码、EX执行)。



注：① 图中 MUL 的 ID 阶段推迟，因为 ID 阶段要将 R1 取至 MDR，且 MUL 与前面的 MOV 存在数据相关。②

第 5~6 段时间中 ADD 的 EX 与 MUL 的 EX 不会冲突，因其不在同一个执行部件，而 ADD 的最后一个 EX 需要等 MUL 执行完后才能执行。

05. 【解答】

1) x 的机器码为 $[x]_b = 1111\ 1101\ 1111\ 1111B$ ，即指令执行前($R1 = FDFFH$)，右移 1 位后为 $1111\ 1110\ 1111\ 1111B$ ，即指令执行后($R1 = FEFFH$)。

2) 每个时钟周期只能有一条指令进入流水线，从第 5 个时钟周期开始，每个时钟周期都会有一条指令执行完毕，因此至少需要 $4 + (5 - 1) = 8$ 个时钟周期。

3) I_3 的 ID 段被阻塞的原因：因为 I_3 与 I_1 和 I_2 都存在数据相关，需等到 I_1 和 I_2 将结果写回寄存器后， I_3 才能读寄存器内容，所以 I_3 的 ID 段被阻塞。 I_4 的 IF 段被阻塞的原因：因为 I_4 的前一条指令 I_3 在 ID 段被阻塞，所以 I_4 的 IF 段被阻塞。

注意：要求“按序发射，按序完成”，因此第 2 小问中下一条指令的 IF 必须和上一条指令的 ID 并行，以免因上一条指令发生冲突而导致下一条指令先执行完。

4) 因 $2*x$ 操作有左移和加法两种实现方法，因此 $x = x*2 + a$ 对应的指令序列为

I1	LOAD	R1, [x]
I2	LOAD	R2, [a]
I3	SHL	R1 //或者 ADD R1, R2

I4 ADD	R1, R2
I5 STORE	R2, [x]

这 5 条指令在流水线中执行过程如下图所示。

指令	时间单元																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
I1	IF	ID	EX	M	WB												
I2		IF	ID	EX	M	WB											
I3			IF			ID	EX	M	WB								
I4						IF				ID	EX	M	WB				
I5									IF					ID	EX	M	WB

因此执行 $x = x * 2 + a$ 语句最少需要 17 个时钟周期。

06. 【解答】

该题为计算机组成原理的综合题型，难度较大。该题涉及指令系统、存储管理和 CPU 三部分内容，特别是五流水段流水线考生应高度重视这部分知识。

整个指令执行过程中各流水段的时间是相同的，受统一的时钟控制。各流水段在 5.6.1 节中介绍过，这里讨论流水段发生阻塞的情况：

- ① 若上一条指令的 WB 写回的寄存器与本指令对应的寄存器相同，会发生资源冲突。这时有 3 个时钟周期的阻塞，使本指令 ID 应在上一条 WB 后，如下图所示。

指令	时间单元																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14			
I ₁	IF	ID	EX	M	WB												
I ₂		IF	ID	EX	M	WB											
I ₃			IF				ID	EX	M	WB							
I ₄						IF					ID	EX	M	WB			

第三条指令阻塞了 3 个时钟周期。

- ② 跳转指令 (JMP) (bra)：由于流水线默认直接提取下一条指令，若指令为 JMP 或 JC（根据情况预判跳转结果），在没有分支预测的情况下，默认有 3 个时钟周期的阻塞使本指令 ID 应在上一条 WB 后。`bne` 表示条件跳转 (branch not equal) 指令。

在了解上面的基础知识后，我们再看这道大题。

- 1) 已知计算机 M 采用 32 位定长指令字，即一条指令占 4B，观察表中各指令的地址可知，每条指令的地址差为 4 个地址单位，即 4 个地址单位代表 4B，一个地址单位就代表了 1B，所以该计算机是按字节编址的。
- 2) 在二进制中某数左移两位相当于乘以 4，由该条件可知，数组间的数据间隔为 4 个地址单位，而计算机按字节编址，所以数组 A 中的每个元素占 4B。
- 3) 由表可知，`bne` 指令的机器代码为 1446FFFFAH，根据题目给出的指令格式，后 2B 的内容为 OFFSET 字段，所以该指令的 OFFSET 字段为 FFFAH，用补码表示，值为 -6。系统执行到 `bne` 指令时，PC 自动加 4，PC 的内容为 08048118H，而跳转的目标是 08048100H，两者相差了 18H，即 24 个单位的地址间隔，所以偏移地址的一位即是真实跳转地址的 $-24/-6 = 4$ 位。可知 `bne` 指令的转移目标地址计算公式为 $(PC) + 4 + \text{OFFSET} \times 4$ 。

4) 由于数据相关而发生阻塞的指令为第 2、3、4、6 条，因为第 2、3、4、6 条指令都与各自前一条指令发生数据相关。第 6 条指令会发生控制冒险。

当前循环的第 5 条指令与下次循环的第 1 条指令虽然有数据相关，但由于第 6 条指令后有 3 个时钟周期的阻塞，因而消除了该数据相关。

07. 【解答】

该题继承了上题中的相关信息，统考中首次引入此种设置，具体考查程序的运行结果、Cache 的大小和命中率的计算，以及磁盘和 TLB 的相关计算，是一道比较综合的题型。2015 年同样出现了 23 分大题的设定，希望读者对其足够重视。

1) R2 中装的是 i 的值，循环条件是 $i < N(1000)$ ，即当 i 自增到不满足这个条件时跳出循环，程序结束，所以此时 i 的值为 1000。

2) Cache 共有 16 行，每块 32 字节，所以 Cache 数据区的容量为 $16 \times 32B = 512B$ 。

P 共有 6 条指令，占 24B，小于主存块大小 32B，其起始地址为 0804 8100H，对应一块的开始位置，由此可知所有指令都在一个主存块内。读取第一条指令时会发生 Cache 缺失，因此将 P 所在的主存块调入 Cache 的某一行，以后每次读取指令时，都能在指令 Cache 中命中。因此在 1000 次循环中，只会发生 1 次指令访问缺失，所以指令 Cache 的命中率为 $(1000 \times 6 - 1) / (1000 \times 6) = 99.98\%$ 。

3) 指令 4 为加法指令，即对应 $sum += A[i]$ ，当数组 A 中元素的值过大时，会导致这条加法指令发生溢出异常；而指令 2、5 虽然都是加法指令，但它们分别为数组地址的计算指令和存储变量 i 的寄存器进行自增的指令，而 i 最大到达 1000，所以它们都不会产生溢出异常。

只有访存指令可能产生缺页异常，即指令 3 可能产生缺页异常。因为数组 A 在磁盘的一页上，而一开始数组并不在主存中，第一次访问数组时会导致访盘，把 A 调入内存，而以后数组 A 的元素都在内存中，不会导致访盘，所以该程序共访盘一次。

每访问一次内存数据就会查一次 TLB，共访问数组 1000 次，所以此时又访问 1000 次 TLB，还要考虑到第一次访问数组 A，即访问 A[0] 时，会多访问一次 TLB（第一次访问 A[0] 时会先查一次 TLB，然后产生缺页，处理完缺页中断后，会重新访问 A[0]，此时又查 TLB），所以访问 TLB 的次数一共是 1001 次。

5.7 多处理器的基本概念

5.7.1 SISD、SIMD、MIMD 的基本概念

基于指令流的数量和数据流的数量，将计算机体系结构分为 SISD、SIMD、MISD 和 MIMD 四类。常规的单处理器属于 SISD，而常规的多处理器属于 MIMD。

1. 单指令流单数据流 (SISD) 结构

SISD 是传统的串行计算机结构，这种计算机通常仅包含一个处理器和一个存储器，处理器在一段时间内仅执行一条指令，按指令流规定的顺序串行执行指令流中的若干条指令。为了提高速度，有些 SISD 计算机采用流水线的方式，因此，SISD 处理器有时会设置多个功能部件，并且采用多模块交叉方式组织存储器。本书前面介绍的内容多属于 SISD 结构。

2. 单指令流多数据流 (SIMD) 结构

SIMD 是指一个指令流同时对多个数据流进行处理，一般称为数据级并行技术。这种结构的



计算机通常由一个指令控制部件、多个处理单元组成。每个处理单元虽然执行的都是同一条指令，但是每个单元都有自己的地址寄存器，这样每个单元就都有不同的数据地址，因此，不同处理单元执行的同一条指令所处理的数据是不同的。一个顺序应用程序被编译后，可能按 SISD 组织并运行于串行硬件上，也可能按 SIMD 组织并运行于并行硬件上。

SIMD 在使用 for 循环处理数组时最有效，比如，一条分别对 16 对数据进行运算的 SIMD 指令如果在 16 个 ALU 中同时运算，则只需要一次运算时间就能完成运算。SIMD 在使用 case 或 switch 语句时效率最低，此时每个执行单元必须根据不同的数据执行不同的操作。

3. 多指令流单数据流 (MISD) 结构

MISD 是指同时执行多条指令，处理同一个数据，实际上不存在这样的计算机。

4. 多指令流多数据流 (MIMD) 结构

MIMD 是指同时执行多条指令分别处理多个不同的数据，MIMD 分为多计算机系统和多处理器系统。多计算机系统中的每个计算机节点都具有各自的私有存储器，并且具有独立的主存地址空间，不能通过存取指令来访问不同节点的私有存储器，而要通过消息传递进行数据传送，也称消息传递 MIMD。多处理器系统是共享存储多处理器 (SMP) 系统的简称，它具有共享的单一地址空间，通过存取指令来访问系统中的所有存储器，也称共享存储 MIMD。

向量处理器是 SIMD 的变体，是一种实现了直接操作一维数组（向量）指令集的 CPU，而串行处理器只能处理单一数据集。其基本理念是将从存储器中收集的一组数据按顺序放到一组向量寄存器中，然后以流水化的方式对它们依次操作，最后将结果写回寄存器。向量处理器在特定工作环境中极大地提升了性能，尤其是在数值模拟或者相似的领域中。

SIMD 和 MIMD 是两种并行计算模式，其中 SIMD 是一种数据级并行模式，而 MIMD 是一种并行度更高的线程级并行或线程级以上并行计算模式。

5.7.2 硬件多线程的基本概念

在传统 CPU 中，线程的切换包含一系列开销，频繁地切换会极大影响系统的性能，为了减少线程切换过程中的开销，便诞生了硬件多线程。在支持硬件多线程的 CPU 中，必须为每个线程提供单独的通用寄存器组、单独的程序计数器等，线程的切换只需激活选中的寄存器，从而省略了与存储器数据交换的环节，大大减少了线程切换的开销。

硬件多线程有 3 种实现方式：细粒度多线程、粗粒度多线程和同时多线程 (SMT)。

1. 细粒度多线程

多个线程之间轮流交叉执行指令，多个线程之间的指令是不相关的，可以乱序并行执行。在这种方式下，处理器能在每个时钟周期切换线程。例如，在时钟周期 i，将线程 A 中的多条指令发射执行；在时钟周期 i+1，将线程 B 中的多条指令发射执行。

2. 粗粒度多线程

仅在一个线程出现了较大开销的阻塞时，才切换线程，如 Cache 缺失。在这种方式下，当发生流水线阻塞时，必须清除被阻塞的流水线，新线程的指令开始执行前需要重载流水线，因此，线程切换的开销比细粒度多线程更大。

3. 同时多线程

同时多线程 (SMT) 是上述两种多线程技术的变体。它在实现指令级并行的同时，实现线程级并行，也就是说，它在同一个时钟周期中，发射多个不同线程中的多条指令执行。

图 5.22 分别是三种硬件多线程实现方式的调度示例。

时钟	CPU
i	发射线程 A 的指令j、j+1
i+1	发射线程 B 的指令k、k+1
i+2	发射线程 A 的指令j+2、j+3
i+3	发射线程 B 的指令k+2、k+3

(a) 细粒度多线程示例

时钟	CPU
i	发射线程 A 的指令j、j+1, 线程 B 的指令k、k+1
i+1	发射线程 A 的指令j+2, 线程 B 的指令k+2, 线程 C 的指令m
i+2	发射线程 A 的指令j+3, 线程 C 的指令m+1、m+2

(b) 粗粒度多线程示例

时钟	CPU
i	发射线程 A 的指令j、j+1, 线程 B 的指令k、k+1
i+1	发射线程 A 的指令j+2, 线程 B 的指令k+2, 线程 C 的指令m
i+2	发射线程 A 的指令j+3, 线程 C 的指令m+1、m+2

(c) 同时多线程示例

图 5.22 三种硬件多线程方式的调度示例

Intel 处理器中的超线程 (Hyper-threading) 就是同时多线程 SMT, 即在一个单处理器或单个核中设置了两套线程状态部件, 共享高速缓存和功能部件。

5.7.3 多核处理器的基本概念

多核处理器是指将多个处理单元集成到单个 CPU 中, 每个处理单元称为一个核 (core)。每个核可以有自己的 Cache, 也可以共享同一个 Cache。所有核一般都是对称的, 并且共享主存储器, 因此多核属于共享存储的对称多处理器。图 5.23 是一个不共享 Cache 的双核 CPU 结构。

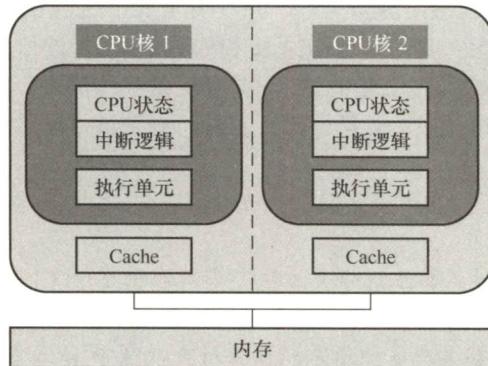


图 5.23 不共享 Cache 的双核 CPU 结构



在多核计算机系统中, 如要充分发挥硬件的性能, 必须采用多线程 (或多进程) 执行, 使得每个核在同一时刻都有线程在执行。与单核上的多线程不同, 多核上的多个线程是在物理上并行执行的, 是真正意义上的并行执行, 在同一时刻有多个线程在并行执行。而单核上的多线程是一种多线程交错执行, 实际上在同一时刻只有一个线程在执行。

下面通过一个例子来理解相关的概念。假设要将四颗圆石头滚到马路对面, 滚动每颗石头平均需花费 1 分钟。串行处理器会逐一滚动每颗石头, 花费 4 分钟。拥有两个核的多核处理器让两个人去滚石头, 即每人滚两颗, 花费 2 分钟。向量处理器找到一根长木板, 放在四颗石头后面, 推动木板即可同时滚动四块石头, 理论上只要力量够大, 就只需要 1 分钟。多核处理器相当于拥有多名工人, 而向量处理器拥有一种方法, 可以同时对多件事进行相同的操作。

5.7.4 共享内存多处理器的基本概念

具有共享的单一物理地址空间的多处理器被称为共享内存多处理器（SMP）。处理器通过存储器中的共享变量互相通信，所有处理器都能通过存取指令访问任何存储器的位置。注意，即使这些系统共享同一个物理地址空间，它们仍然可在自己的虚拟地址空间中单独地运行程序。

单一地址空间的多处理器有两种类型。第一类，每个处理器对所有存储单元的访问时间是大致相同的，即访问时间与哪个处理器提出访存请求及访问哪个字无关，这类机器被称为统一存储访问（UMA）多处理器。第二类，某些访存请求要比其他的快，具体取决于哪个处理器提出了访问请求以及访问哪个字，这是由于主存被分割并分配给了同一机器上的不同处理器或内存控制器，这类机器被称为非统一存储访问（NUMA）多处理器。

- 统一存储访问（UMA）多处理器：根据处理器与共享存储器之间的连接方式，分为基于总线、基于交叉开关网络和基于多级交换网络连接等几种处理器。
- 非统一存储访问（NUMA）多处理器：处理器中不带高速缓存时，被称为 NC-NUMA；处理器中带有一致性高速缓存时，被称为 CC-NUMA。

早期的计算机，内存控制器没有整合进 CPU，访存操作需要经过北桥芯片（集成了内存控制器，并与内存相连），CPU 通过前端总线和北桥芯片相连，这就是统一存储访问（UMA）构架。随着 CPU 性能提升由提高主频转到增加 CPU 数量（多核、多 CPU），越来越多的 CPU 对前端总线的争用使得前端总线成为瓶颈。为了消除 UMA 架构的瓶颈，非统一存储访问（NUMA）构架诞生，内存控制器被集成到 CPU 内部，每个 CPU 都有独立的内存控制器。每个 CPU 都独立连接到一部分内存，CPU 直连的这部分内存被称为本地内存。CPU 之间通过 QPI 总线相连。CPU 可以通过 QPI 总线访问其他 CPU 的远程内存。与 UMA 架构不同的是，在 NUMA 架构下，内存的访问出现了本地和远程的区别，访问本地内存明显要快于访问远程内存。

由于可能会出现多个处理器同时访问同一共享变量的情况，在操作共享变量时需要进行同步，否则，一个处理器可能会在其他处理器尚未完成对共享变量的修改时，就开始使用该变量。常用方法是通过对共享变量加锁的方式来控制对共享变量互斥访问。在一个时刻只能有一个处理器获得锁，其他要操作该共享变量的处理器必须等待，直到该处理器解锁该变量为止。

5.7.5 本节习题精选

单项选择题

01. 按照 Flynn 提出的计算机系统分类方法，多处理机属于（ ）。
 - A. SISD
 - B. SIMD
 - C. MISD
 - D. MIMD
02. 具有一个控制部件和多个处理单元的计算机系统属于（ ）结构。
 - A. SISD
 - B. SIMD
 - C. MISD
 - D. MIMD
03. 下列关于超线程（HT）技术的描述中，正确的是（ ）。
 - A. 超线程技术可以令四核的 Intel Core i7 处理器变成八核
 - B. 超线程是一项硬件技术，能使系统性能大幅提升，与操作系统和应用软件无关
 - C. 含有超线程技术的 CPU 需要芯片组的支持才能发挥技术优势
 - D. 超线程模拟出的每个 CPU 核都具有独立的资源，各自工作互不干扰
04. 双核 CPU 和超线程 CPU 的共同点是（ ）。
 - A. 都有两个内核
 - B. 都能同时执行两个运算
 - C. 都包含两个 CPU
 - D. 都不会出现争抢资源的现象
05. 下列关于双核技术的叙述中，正确的是（ ）。

- A. 双核是指主板上有两个 CPU
 B. 双核是利用超线程技术实现的
 C. 双核是指在 CPU 上集成两个运算核心
 D. 双核 CPU 是时间并行的并行计算
06. 下列有关多核 CPU 和单核 CPU 的描述中，错误的是（ ）。
 A. 双核的频率为 2.4GHZ，那么其中每个核心的频率也是 2.4GHZ
 B. 采用双核 CPU 可以降低计算机系统的功耗和体积
 C. 多核 CPU 共用一组内存，数据共享
 D. 所有程序在多核 CPU 上运行速度都快
07. 下列关于多核 CPU 的描述中，正确的是（ ）。
 A. 各核心完全对称，拥有各自的 Cache
 B. 任何程序都可以同时在多个核心上运行
 C. 一颗 CPU 中集成了多个完整的执行内核，可同时进行多个运算
 D. 只有使用了多核 CPU 的计算机，才支持多任务操作系统
08. 下列关于多处理器的说法中，正确的是（ ）。
 I. 一般采用偶数路 CPU，如 2 路、4 路、6 路等
 II. NUMA 构架比 UMA 构架的运算扩展性要强
 III. UMA 构架需要解决的重要问题是 Cache 一致性
 A. I B. I 和 II C. I 和 III D. I、II 和 III
09. 【2022 统考真题】下列关于并行处理技术的叙述中，不正确的是（ ）。
 A. 多核处理器属于 MIMD 结构 B. 向量处理器属于 SIMD 结构
 C. 硬件多线程技术只可用于多核处理器
 D. SMP 中所有处理器共享单一物理地址空间

5.7.6 答案与解析

单项选择题

01. D

Flynn 分类法将计算机体系结构分为 SISD、SIMD、MISD 和 MIMD 四类。常规的单处理器属于 SISD，常规的多处理器属于 MIMD。

02. B

单指令流多数据流（SIMD）结构的计算机通常由一个指令控制部件、多个处理单元组成，不同处理单元执行的同一条指令所处理的数据可以不同。

03. C

超线程技术是在一个 CPU 中，提供两套线程处理单元，让单个处理器实现线程级并行。虽然采用超线程技术能够同时执行两个线程，但是当两个线程同时需要某个资源时，其中一个线程必须暂时挂起，直到这些资源空闲后才能继续运行。因此，超线程的性能并不等于两个 CPU 的性能。而且，超线程技术的 CPU 需要芯片组、操作系统（如 Windows 98 不支持超线程技术）和应用软件的支持，才能发挥该项技术的优势。双核技术是指将两个一样的 CPU 集成到一个封装内（或者直接将两个 CPU 做成一个芯片），而超线程技术在 CPU 内部仅复制必要的线程资源来让两个线程同时运行，能并行执行两个线程，模拟实体双核心。仅选项 C 正确。

04. B

超线程技术在 CPU 内部仅复制必要的线程资源，共享 CPU 的高速缓存和功能部件，让两个线程可以并行执行，模拟双核心 CPU，选项 A、C 错误。当两个线程同时需要某个共享资源时，其中一个线程必须暂时挂起，直到这些资源空闲后才能继续运行，选项 D 错误。选项 B 正确。

05. C

双核是指将两个 CPU 核心集成到一个封装中，核心又称内核，是 CPU 最重要的组成部分，选项 C 正确。主板上有两个 CPU 属于多处理器，选项 A 错误。超线程技术是模拟实体双核，并不能算作真正意义上的双核，选项 B 错误。时间并行是指流水线技术，空间并行则是指硬件资源的重复，空间并行导致了两类并行机的产生，按 Flynn 分类法分为 SIMD 和 MIMD，选项 D 错误。

06. D

多核 CPU 的核心通常都是对称的，因此 2.4GHz 双核 CPU 中两个核的主频也是 2.4GHz，选项 A 正确。早期 CPU 性能提升主要靠提高主频，导致功耗增大，发热量大，而且当主频提高到一定程度后，CPU 性能的提升不再明显，后来转到增加 CPU 核心的方向，将 2 个核心集成到一个芯片内，提供等同双 CPU 的性能，这显然也降低了 CPU 的体积，选项 B 正确。选项 C 显然正确。在多核 CPU 上运行一个不支持多线程的程序，显然不能发挥多核 CPU 的优势，选项 D 错误。

07. C

多核 CPU 的各核心可以有独自的 Cache，也可以共享同一个 Cache，选项 A 错误。只有支持多线程的并行处理程序才能同时在多个核心上运行，发挥多核的优势，选项 B 错误。选项 C 正确。多任务系统又称多道程序系统，可以运行在单核 CPU 上，宏观上并行，微观上串行，选项 D 错误。

08. D

SMP 也称对称多处理器，一般采用偶数路 CPU，I 正确。UMA 构架由于所有 CPU 共享相同的内存，增加 CPU 路数会加大访存冲突，通常 2 或 4 路的性能最好，而 NUMA 理论上支持无限扩展，II 正确。UMA 构架中所有 CPU 共享同一内存空间，每个 CPU 的 Cache 中都是共享内存中的一部分副本，因此各 CPU 的 Cache 一致性是需要解决的重要问题，III 正确。

注意，第 3 章讨论的一致性是指 Cache 和主存之间的数据一致性。在多核系统中，每个 CPU 的 Cache 中都是它们共享的内存中的一部分副本，因此多核系统的 Cache 一致性既包括 Cache 和内存之间的一致性，还包括各 CPU 的 Cache 之间的一致性，也就是说，对内存同一位置的数据，不同 CPU 的 Cache 不应该有不一致的内容。

09. C

MIMD 结构分为多计算机系统和多处理器系统，选项 A 正确。向量处理器是 SIMD 的变体，属于 SIMD 结构，B 正确。硬件多线程技术在一个核中处理多个线程，可用于单核处理器，选项 C 错误。共享内存多处理器（SMP）具有共享的单一物理地址空间，所有核都可通过存取指令来访问同一片主存地址空间，选项 D 正确。

5.8 本章小结

本章开头提出的问题的参考答案如下。

1) 指令和数据均存放在内存中，计算机如何从时间和空间上区分它们是指令还是数据？

从时间上讲，取指令事件发生在“取指周期”，取数据事件发生在“执行周期”。从空间上讲，

从内存读出的指令流流向控制器（指令寄存器），从内存读出的数据流流向运算器（通用寄存器）。

2) 什么是指令周期、机器周期和时钟周期？它们之间有何关系？

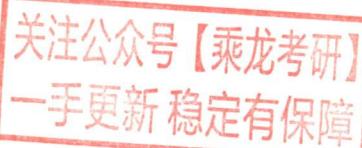
CPU 每取出并执行一条指令所需的全部时间称为指令周期；机器周期是在同步控制的机器中，执行指令周期中一步相对完整的操作（指令步）所需的时间，通常安排机器周期长度 = 主存周期；时钟周期是指计算机主时钟的周期时间，它是计算机运行时最基本的时序单位，对应完成一个微操作所需的时间，通常时钟周期 = 计算机主频的倒数。

3) 什么是微指令？它和上一章谈到的指令有什么关系？

控制部件通过控制线向执行部件发出各种控制命令，通常把这种控制命令称为微命令，而一组实现一定操作功能的微命令的组合，构成一条微指令。许多条微指令组成的序列构成微程序，微程序完成对指令的解释执行。指令，即指机器指令。每条指令可以完成一个独立的算术运算或逻辑运算操作。在采用微程序控制器的 CPU 中，一条指令对应一个微程序，一个微程序由许多微指令构成，一条微指令会发出很多不同的微命令。

4) 什么是指令流水线？指令流水线相对于传统体系结构的优势是什么？

指令流水线是把指令分解为若干子过程，通过将每个子过程与其他子过程并行执行，来提高计算机的吞吐率的技术。采用流水线技术只需增加少量硬件就能把计算机的运算速度提高几倍，因此成为计算机中普遍使用的一种并行处理技术，通过在同一个时间段使用各功能部件，使得利用率明显提高。



5.9 常见问题和易混淆知识点

1. 流水线越多，并行度就越高。是否流水段越多，指令执行越快？

错误，原因如下：

- 1) 流水段缓冲之间的额外开销增大。每个流水段有一些额外开销用于缓冲间传送数据、进行各种准备和发送等功能，这些开销加长了一条指令的整个执行时间，当指令间逻辑上相互依赖时，开销更大。
- 2) 流水段间控制逻辑变多、变复杂。用于流水线优化和存储器（或寄存器）冲突处理的控制逻辑将随流水段的增加而大增，这可能导致用于流水段之间控制的逻辑比段本身的控制逻辑更复杂。

2. 有关指令相关、数据相关的几个概念

- 1) 两条连续的指令读取相同的寄存器时，会产生读后读（Read After Read, RAR）相关，这种相关不会影响流水线。
- 2) 某条指令要读取上一条指令所写入的寄存器时，会产生写后读（Read After Write, RAW）相关，它称数据相关或真相关，影响流水线。按序流动的流水线只可能出现 RAW 相关。
- 3) 某条指令的上条指令要读/写该指令的输出寄存器时，会产生读后写（Write After Read, WAR）和写后写（Write After Write, WAW）相关。在非按序流动的流水线中，既可能发生 RAW 相关，又可能发生 WAR 相关和 WAW 相关。

对流水线影响最严重的指令相关是数据相关。

3. 组合逻辑电路和时序逻辑电路有什么区别？

组合逻辑电路是具有一组输出和一组输入的非记忆性逻辑电路，它的基本特点是任何时刻的输出信号状态仅取决于该时刻各个输入信号状态的组合，而与电路在输入信号作用前的状态无关。组合电路不含存储信号的记忆单元，输出与输入之间无反馈通路，信号是单向传输的。

时序逻辑电路中任意时刻的输出信号不仅和当时的输入信号有关，而且与电路原来的状态有关，这是时序逻辑电路在逻辑功能上的特点。因而时序逻辑电路必然包含存储记忆单元。

此外，组合逻辑电路没有统一的时钟控制，而时序逻辑电路则必须在时钟节拍下工作。

关注公众号【乘龙考研】
一手更新 稳定有保障

第 6 章 总线

关注公众号【乘龙考研】
一手更新 稳定有保障

【考纲内容】

- 总线的基本概念
- 总线的组成及性能指标
- 总线事务和定时

扫一扫



视频讲解

【复习提示】

本章的知识点较少，通常以选择题的形式出现，特别是总线的特点、猝发传输方式、性能指标、定时方式及常见的总线标准等。总线带宽的计算也可能结合其他章节出综合题。

在学习本章时，请读者思考以下问题：

- 1) 引入总线结构有什么好处？
- 2) 引入总线结构会导致什么问题？如何解决？

请读者在学习本章的过程中寻找答案，本章末尾会给出参考答案。

6.1 总线概述

随着 I/O 设备的种类和数量越来越多，为了更好地解决 I/O 设备和主机之间连接的灵活性，计算机的结构从分散连接发展为总线连接。为了进一步简化设计，又提出了各类总线标准。

6.1.1 总线基本概念

1. 总线的定义

总线是一组能为多个部件分时共享的公共信息传送线路。分时和共享是总线的两个特点。分时是指同一时刻只允许有一个部件向总线发送信息，若系统中有多个部件，则它们只能分时地向总线发送信息。共享是指总线上可以挂接多个部件，各个部件之间互相交换的信息都可通过这组线路分时共享，多个部件可同时从总线上接收相同的信息。

2. 总线设备

总线上所连接的设备，按其对总线有无控制功能可分为 **主设备** 和 **从设备** 两种。

主设备：指获得总线控制权的设备。

从设备：指被主设备访问的设备，它只能响应从主设备发来的各种总线命令。

3. 总线特性

总线特性是指机械特性（尺寸、形状）、电气特性（传输方向和有效的电平范围）、功能特性（每根传输线的功能）和时间特性（信号和时序的关系）。

6.1.2 总线的分类

计算机系统中的总线，按功能划分为以下 4 类。

1. 片内总线

片内总线是芯片内部的总线，它是 CPU 芯片内部寄存器与寄存器之间、寄存器与 ALU 之间的公共连接线。

2. 系统总线

系统总线是计算机系统内各功能部件（CPU、主存、I/O 接口）之间相互连接的总线。按系统总线传输信息内容的不同，又可分为 3 类：数据总线、地址总线和控制总线。

- 1) 数据总线用来传输各功能部件之间的数据信息，它是双向传输总线，其位数与机器字长、存储字长有关。
- 2) 地址总线用来指出数据总线上的源数据或目的数据所在的主存单元或 I/O 端口的地址，它是单向传输总线，地址总线的位数与主存地址空间的大小有关。
- 3) 控制总线传输的是控制信息，包括 CPU 送出的控制命令和主存（或外设）返回 CPU 的反馈信号。

注意区分数据通路和数据总线：各个功能部件通过数据总线连接形成的数据传输路径称为数据通路。数据通路表示的是数据流经的路径，而数据总线是承载的媒介。

3. I/O 总线

I/O 总线主要用于连接中低速的 I/O 设备，通过 I/O 接口与系统总线相连接，目的是将低速设备与高速总线分离，以提升总线的系统性能，常见的有 USB、PCI 总线。

4. 通信总线

通信总线是在计算机系统之间或计算机系统与其他系统（如远程通信设备、测试设备）之间传送信息的总线，通信总线也称外部总线。

此外，按时序控制方式可将总线划分为同步总线和异步总线，还可按数据传输格式将总线划分为并行总线和串行总线。

6.1.3 系统总线的结构

1. 单总线结构

单总线结构将 CPU、主存、I/O 设备（通过 I/O 接口）都挂接在一组总线上，允许 I/O 设备之间、I/O 设备与主存之间直接交换信息，如图 6.1 所示。CPU 与主存、CPU 与外设之间可直接进行信息交换，而无须经过中间设备的干预。

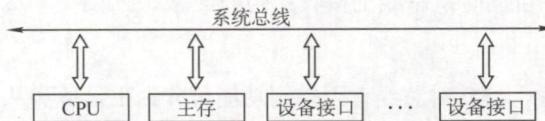


图 6.1 单总线结构

注意，单总线并不是指只有一根信号线，系统总线按传送信息的不同可细分为地址总线、数据总线和控制总线。

优点：结构简单，成本低，易于接入新的设备。

关注公众号【乘龙考研】
一手更新 稳定有保障

缺点：带宽低、负载重，多个部件只能争用唯一的总线，且不支持并发传送操作。

2. 双总线结构

双总线结构有两条总线：一条是主存总线，用于在 CPU、主存和通道之间传送数据；另一条是 I/O 总线，用于在多个外部设备与通道之间传送数据，如图 6.2 所示。

优点：将低速 I/O 设备从单总线上分离出来，实现了存储器总线和 I/O 总线分离。

缺点：需要增加通道等硬件设备。

3. 三总线结构

三总线结构是在计算机系统各部件之间采用 3 条各自独立的总线来构成信息通路，这 3 条总线分别为主存总线、I/O 总线和直接内存访问（DMA）总线，如图 6.3 所示。

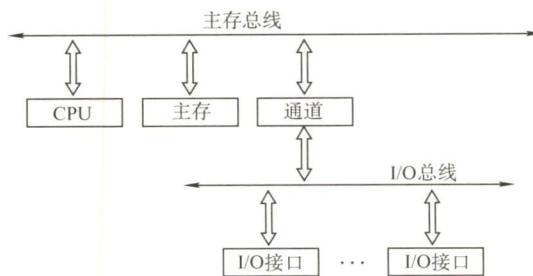


图 6.2 双总线结构

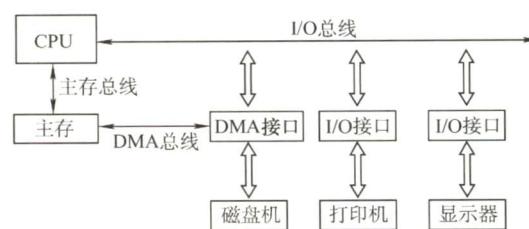


图 6.3 三总线结构

主存总线用于在 CPU 和内存之间传送地址、数据和控制信息。I/O 总线用于在 CPU 和各类外设之间通信。DMA 总线用于在内存和高速外设之间直接传送数据。

优点：提高了 I/O 设备的性能，使其更快地响应命令，提高系统吞吐量。

缺点：系统工作效率较低。

6.1.4 常见的总线标准

总线标准是国际上公布的互连各个模块的标准，是把各种不同的模块组成计算机系统时必须遵守的规范。典型的总线标准有 ISA、EISA、VESA、PCI、AGP、PCI-Express、USB 等。它们的主要区别是总线宽度、带宽、时钟频率、寻址能力、是否支持突发传送等。

- 1) ISA, Industry Standard Architecture, 工业标准体系结构。是最早出现的微型计算机的系统总线，应用在 IBM 的 AT 机上。
- 2) EISA, Extended Industry Standard Architecture, 扩展的 ISA。是为配合 32 位 CPU 而设计的扩展总线，EISA 对 ISA 完全兼容。
- 3) VESA, Video Electronics Standards Association, 视频电子标准协会。是一个 32 位的局部总线，是针对多媒体 PC 要求高速传送活动图像的大量数据而推出的。
- 4) PCI, Peripheral Component Interconnect, 外部设备互连。是高性能的 32 位或 64 位总线，是专为高度集成的外围部件、扩充插板和处理器/存储器系统设计的互连机制。目前常用的 PCI 适配器有显卡、声卡、网卡等。PCI 总线支持即插即用。PCI 总线是一个与处理器时钟频率无关的高速外围总线，属于局部总线。
- 5) AGP, Accelerated Graphics Port, 加速图形接口。是一种视频接口标准，专用于连接主存和图形存储器，用于传输视频和三维图形数据，属于局部总线。
- 6) PCI-E, PCI-Express。是最新的总线接口标准，它将全面取代现行的 PCI 和 AGP。

- 7) RS-232C。是由美国电子工业协会（EIA）推荐的一种串行通信总线，是应用于串行二进制交换的数据终端设备（DTE）和数据通信设备（DCE）之间的标准接口。
- 8) USB, Universal Serial Bus, 通用串行总线。是一种连接外部设备的 I/O 总线，属于设备总线。具有即插即用、热插拔等优点，有很强的连接能力。
- 9) PCMCIA, Personal Computer Memory Card International Association。广泛应用于笔记本电脑的一种接口标准，是一个用于扩展功能的小型插槽。具有即插即用功能。
- 10) IDE, Integrated Drive Electronics, 集成设备电路。更准确地称为 ATA，是一种 IDE 接口磁盘驱动器接口类型，硬盘和光驱通过 IDE 接口与主板连接。
- 11) SCSI, Small Computer System Interface, 小型计算机系统接口。是一种用于计算机和智能设备之间（硬盘、软驱）系统级接口的独立处理器标准。
- 12) SATA, Serial Advanced Technology Attachment, 串行高级技术附件。是一种基于行业标准的串行硬件驱动器接口，是由 Intel、IBM、Dell 等公司共同提出的硬盘接口规范。

6.1.5 总线的性能指标

- 1) 总线传输周期。指一次总线操作所需的时间，包括申请阶段、寻址阶段、传输阶段和结束阶段。总线传输周期通常由若干总线时钟周期构成。
- 2) 总线时钟周期。即机器的时钟周期。计算机有一个统一的时钟，以控制整个计算机的各个部件，总线也要受此时钟的控制。
- 3) 总线工作频率。总线上各种操作的频率，为总线周期的倒数。实际上指 1 秒内传送几次数据。若总线周期 = N 个时钟周期，则总线的工作频率 = 时钟频率/ N 。
- 4) 总线时钟频率。即机器的时钟频率，它为时钟周期的倒数。
- 5) 总线宽度。又称总线位宽，它是总线上同时能够传输的数据位数，通常指数据总线的根数，如 32 根称为 32 位总线。
- 6) 总线带宽。可理解为总线的最大数据传输率，即单位时间内总线上最多可传输数据的位数，通常用每秒传送信息的字节数来衡量，单位可用字节/秒 (B/s) 表示。总线带宽 = 总线工作频率 × (总线宽度/8)。

注意：总线带宽和总线宽度应加以区别。

- 7) 总线复用。总线复用是指一种信号线在不同的时间传输不同的信息，因此可以使用较少的线传输更多的信息，从而节省空间和成本。
- 8) 信号线数。地址总线、数据总线和控制总线 3 种总线数的总和称为信号线数。其中，总线的主要性能指标为总线宽度、总线（工作）频率、总线带宽，总线带宽是指总线本身所能达到的最高传输速率，它是衡量总线性能的重要指标。

三者关系：总线带宽 = 总线宽度 × 总线频率。

例如，总线工作频率为 22MHz，总线宽度为 16 位，则总线带宽 = $22 \times (16/8) = 44\text{MB/s}$ 。

6.1.6 本节习题精选

一、单项选择题

01. 挂接在总线上的多个部件（ ）。
- A. 只能分时向总线发送数据，并只能分时从总线接收数据
 - B. 只能分时向总线发送数据，但可同时从总线接收数据
 - C. 可同时向总线发送数据，并同时从总线接收数据

- D. 可同时向总线发送数据，但只能分时从总线接收数据
02. 在总线上，同一时刻（ ）。
- 只能有一个主设备控制总线传输操作
 - 只能有一个从设备控制总线传输操作
 - 只能有一个主设备和一个从设备控制总线传输操作
 - 可以有多个主设备控制总线传输操作
03. 在计算机系统中，多个系统部件之间信息传递的公共通路称为总线，就其所传送的信息的性质而言，下列（ ）不是在公共通路上传送的信息。
- 数据信息
 - 地址信息
 - 系统信息
 - 控制信息
04. 系统总线用来连接（ ）。
- 寄存器和运算器部件
 - 运算器和控制器部件
 - CPU、主存和外设部件
 - 接口和外部设备
05. 计算机使用总线结构便于增减外设，同时（ ）。
- 减少信息传输量
 - 提高信息的传输速度
 - 减少信息传输线的条数
 - 提高信息传输的并行性
06. 间址寻址第一次访问内存所得到的信息经系统总线的（ ）传送到CPU。
- 数据总线
 - 地址总线
 - 控制总线
 - 总线控制器
07. 系统总线中地址线的功能是（ ）。
- 选择主存单元地址
 - 选择进行信息传输的设备
 - 选择外存地址
 - 指定主存和I/O设备接口电路的地址
08. 在单机系统中，三总线结构计算机的总线系统组成是（ ）。
- 片内总线、系统总线和通信总线
 - 数据总线、地址总线和控制总线
 - DMA总线、主存总线和I/O总线
 - ISA总线、VESA总线和PCI总线
09. 不同信号在同一条信号线上分时传输的方式称为（ ）。
- 总线复用方式
 - 并串行传输方式
 - 并行传输方式
 - 串行传输方式
10. 主存通过（ ）来识别信息是地址还是数据。
- 总线的类型
 - 存储器数据寄存器（MDR）
 - 存储器地址寄存器（MAR）
 - 控制单元（CU）
11. 在32位总线系统中，若时钟频率为500MHz，传送一个32位字需要5个时钟周期，则该总线的数据传输率是（ ）。
- 200MB/s
 - 400MB/s
 - 600MB/s
 - 800MB/s
12. 传输一幅分辨率为640×480像素、颜色数量为65536的照片（采用无压缩方式），设有有效数据传输率为56kb/s，大约需要的时间是（ ）。
- 34.82s
 - 43.86s
 - 85.71s
 - 87.77s
13. 某总线有104根信号线，其中数据线（DB）为32根，若总线工作频率为33MHz，则其理论最大传输率为（ ）。
- 33MB/s
 - 64MB/s
 - 132MB/s
 - 164MB/s
14. 在一个16位的总线系统中，若时钟频率为100MHz，总线周期为5个时钟周期传输一个字，则总线带宽是（ ）。
- 4MB/s
 - 40MB/s
 - 16MB/s
 - 64MB/s

关注公众号【乘龙考研】
一手更新稳定有保障

15. 微机中控制总线上完整传输的信号有()。
I. 存储器和 I/O 设备的地址码
II. 所有存储器和 I/O 设备的时序信号与控制信号
III. 来自 I/O 设备和存储器的响应信号
A. 仅 I B. II 和 III C. 仅 II D. I、II、III
16. 下列总线标准中属于串行总线的是()。
A. PCI B. USB C. EISA D. ISA
17. 在现代微机主板上，采用局部总线技术的作用是()。
A. 节省系统的总带宽 B. 提高抗干扰能力
C. 抑制总线终端反射 D. 构成紧耦合系统
18. 下列不属于计算机局部总线的是()。
A. VESA B. PCI C. AGP D. ISA
19. 【2009 统考真题】假设某系统总线在一个总线周期中并行传输 4 字节信息，一个总线周期占用 2 个时钟周期，总线时钟频率为 10MHz，则总线带宽是()。
A. 10MB/s B. 20MB/s C. 40MB/s D. 80MB/s
20. 【2010 统考真题】下列选项中的英文缩写均为总线标准的是()。
A. PCI、CRT、USB、EISA B. ISA、CPI、VESA、EISA
C. ISA、SCSI、RAM、MIPS D. ISA、EISA、PCI、PCI-Express
21. 【2011 统考真题】在系统总线的数据线上，不可能传输的是()。
A. 指令 B. 操作数 C. 握手（应答）信号 D. 中断类型号
22. 【2012 统考真题】某同步总线的时钟频率为 100MHz，宽度为 32 位，地址/数据线复用，每传输一个地址或数据占用一个时钟周期。若该总线支持突发（猝发）传输方式，则一次“主存写”总线事务传输 128 位数据所需要的时间至少是()。
A. 20ns B. 40ns C. 50ns D. 80ns
23. 【2012 统考真题】下列关于 USB 总线特性的描述中，错误的是()。
A. 可实现外设的即插即用和热拔插 B. 可通过级联方式连接多台外设
C. 是一种通信总线，连接不同外设 D. 同时可传输 2 位数据，数据传输率高
24. 【2013 统考真题】下列选项中，用于设备和设备控制器(I/O 接口)之间互连的接口标准是()。
A. PCI B. USB C. AGP D. PCI-Express
25. 【2014 统考真题】某同步总线采用数据线和地址线复用方式，其中地址/数据线有 32 根，总线时钟频率为 66MHz，每个时钟周期传送两次数据（上升沿和下降沿各传送一次数据），该总线的最大数据传输率（总线带宽）是()。
A. 132MB/s B. 264MB/s C. 528MB/s D. 1056MB/s
26. 【2014 统考真题】一次总线事务中，主设备只需给出一个首地址，从设备就能从首地址开始的若干连续单元读出或写入多个数据。这种总线事务方式称为()。
A. 并行传输 B. 串行传输 C. 突发传输 D. 同步传输
27. 【2015 统考真题】下列有关总线定时的叙述中，错误的是()。
A. 异步通信方式中，全互锁协议最慢
B. 异步通信方式中，非互锁协议的可靠性最差
C. 同步通信方式中，同步时钟信号可由各设备提供

- D. 半同步通信方式中，握手信号的采样由同步时钟控制
28. 【2016 统考真题】下列关于总线设计的叙述中，错误的是（ ）。
- 并行总线传输比串行总线传输速度快
 - 采用信号线复用技术可减少信号线数量
 - 采用突发传输方式可提高总线数据传输率
 - 采用分离事务通信方式可提高总线利用率
29. 【2017 统考真题】下列关于多总线结构的叙述中，错误的是（ ）。
- 靠近 CPU 的总线速度较快
 - 存储器总线可支持突发传送方式
 - 总线之间须通过桥接器相连
 - PCI-Express×16 采用并行传输方式
30. 【2018 统考真题】下列选项中，可提高同步总线数据传输率的是（ ）。
- 增加总线宽度
 - 提高总线工作频率
 - 支持突发传输
 - 采用地址/数据线复用
- 仅 I、II
 - 仅 I、II、III
 - 仅 III、IV
 - I、II、III 和 IV
31. 【2019 统考真题】假定一台计算机采用 3 通道存储器总线，配套的内存条型号为 DDR3-1333，即内存条所接插的存储器总线的工作频率为 1333MHz，总线宽度为 64 位，则存储器总线的总带宽大约是（ ）。
- 10.66GB/s
 - 32GB/s
 - 64GB/s
 - 96GB/s
32. 【2020 统考真题】QPI 总线是一种点对点全双工同步串行总线，总线上的设备可同时接收和发送信息，每个方向可同时传输 20 位信息（16 位数据+4 位校验位），每个 QPI 数据包有 80 位信息，分 2 个时钟周期传送，每个时钟周期传递 2 次。因此，QPI 总线带宽为：每秒传送次数 \times 2B \times 2。若 QPI 时钟频率为 2.4GHz，则总线带宽为（ ）。
- 4.8GB/s
 - 9.6GB/s
 - 19.2GB/s
 - 38.4GB/s

关注公众号【乘龙考研】
一手更新 稳定有保障

二、综合应用题

01. 某总线的时钟频率为 66MHz，在一个 64 位总线中，总线数据传输的周期是 7 个时钟周期传输 6 个字的数据块。
- 总线的数据传输率是多少？
 - 若不改变数据块的大小，而将时钟频率减半，这时总线的数据传输率是多少？
02. 某总线支持二级 Cache 块传输方式，若每块 6 个字，每个字长 4 字节，时钟频率为 100MHz。
- 读操作时，第一个时钟周期接收地址，第二、三个为延时周期，另用 4 个周期传送一个块。读操作的总线传输速率是多少？
 - 写操作时，第一个时钟周期接收地址，第二个为延时周期，另用 4 个周期传送一个块，写操作的总线传输速率是多少？
 - 设在全部的传输中，70%用于读，30%用于写，该总线在本次传输中的平均传输速率是多少？

6.1.7 答案与解析

一、单项选择题

01. B

为了使总线上的数据不发生“冲突”，挂在总线上的多个设备只能分时地向总线发送数据，即某个时刻只能有一个设备向总线传送数据，而从总线接收数据的设备可以有多个，因为接收数据的设备不会对总线产生“干扰”。

02. A

只有主设备才能获得总线控制权，总线上的信息传输由主设备启动，一条总线上可以有多个设备作为主设备，但在同一时刻只能有一个主设备控制总线的传输操作。

03. C

总线包括数据线、地址线和控制线，传送的信息分别为数据信息、地址信息和控制信息。

04. C

系统总线用于连接计算机中的各个功能部件（如 CPU、主存和 I/O 设备）。

05. C

计算机使用总线结构便于增减外设，同时减少信息传输线的条数。但相对于专线结构，实际上也降低了信息传输的并行性及信息的传输速度。

06. A

间址寻址首次访问内存所得到的信息是操作数的有效地址，该地址作为数据通过数据总线传送至 CPU，地址总线是用于 CPU 选择主存单元地址和 I/O 端口地址的单向总线，不能回传。

地址总线由单向的多根信号线组成，可用于 CPU 向主存、外设传送地址信息；数据总线由双向的多根信号线组成，CPU 可以沿着这些线从主存或外设读入数据，也可以发送数据；控制总线上传输控制信息，包括控制命令和反馈信号等。

07. D

地址总线上的代码用来指明 CPU 欲访问的存储单元或 I/O 端口的地址。

08. C

选项 A 是总线按功能层次的划分，单机系统可不需要通信总线。选项 B 都属于系统总线。选项 D 则是三种不同的总线标准。只有选项 C 组成了三总线结构系统。

09. A

串行传输是指数据的传输在一条线路上按位进行，并行传输是指每个数据位有一条单独的传输线，所有数据位同时传输。不同信号在同一条信号线上分时传输的方式，称为总线复用。

10. A

地址和数据在不同的总线上传输，根据总线传输信息的内容进行区分，地址在地址总线上传输，数据在数据总线上传输。

11. B

总线带宽 = 总线宽度×总线频率，本题中的总线宽度为 32 位，即 4B，总线频率为 $500\text{MHz}/5 = 100\text{MHz}$ ，因此总线的数据传输率为 $4\text{B} \times (500\text{MHz}/5) = 400\text{MB/s}$ 。

12. D

$65536 = 2^{16}$ 色，因此颜色深度为 16 位，占据的存储空间为 $640 \times 480 \times 16 = 4915200$ 位。有效传输时间 = $4915200/(56 \times 10^3)\text{s} \approx 87.77\text{s}$ 。

13. C

数据总线 32 根，因此每次传输 32 位，即 4B 数据，总线工作频率为 33MHz，因此理论最大传输速率为 $33 \times 4 = 132\text{MB/s}$ 。

14. B

时钟频率为 100MHz，因此时钟周期 = $1/100\text{MHz} = 0.01\mu\text{s}$ ，总线周期 = 5 个时钟周期 = $5 \times 0.01\mu\text{s} = 0.05\mu\text{s}$ ，总线工作频率 = $1/0.05 = 20\text{MHz}$ ，因总线是 16 位的，即 2B，因此总线带宽 = $20 \times (16/8) = 40\text{MB/s}$ 。

15. B

CPU 的控制总线提供的控制信号包括时序信号、I/O 设备和存储器的响应信号等。

16. B

PCI、EISA、ISA 均是并行总线，USB 是通用串行总线。

17. A

高速设备采用局部总线连接，可以节省系统的总带宽。

18. D

ISA 是系统总线而非局部总线。

19. B

总线带宽是指单位时间内总线上传输数据的位数，通常用每秒传送信息的字节数来衡量，单位为 B/s。由题意可知，在 1 个总线周期（= 2 个时钟周期）内传输了 4 字节信息，时钟周期 = $1/10\text{MHz} = 0.1\mu\text{s}$ ，因此总线带宽为 $4\text{B}/(2 \times 0.1\mu\text{s}) = 4\text{B}/(0.2 \times 10^{-6}\text{s}) = 20\text{MB/s}$ 。

20. D

典型的总线标准有 ISA、EISA、VESA、PCI、PCI-Express、AGP、USB、RS-232C 等。A 中的 CRT 是纯平显示器；B 中的 CPI 是每条指令的时钟周期数；C 中的 RAM 是半导体随机存储器、MIPS 是每秒执行多少百万条指令数。

21. C

取指令时，指令便是在数据线上传输的。操作数显然在数据线上传输。中断类型号用以指出中断向量的地址，CPU 响应中断请求后，将中断应答信号（INTR）发回数据总线，CPU 从数据总线上读取中断类型号后，查找中断向量表，找到相应的中断处理程序入口。而握手（应答）信号属于总线定时的控制信号，应在控制总线上传输。

22. C

由于总线频率为 100MHz，因此时钟周期为 10ns。总线位宽与存储字长都是 32 位，因此每个时钟周期可传送一个 32 位存储字。猝发式发送可以连续传送地址连续的数据，因此总传送时间为：传送地址 10ns，传送 128 位数据 40ns，共需 50ns。

23. D

USB（通用串行总线）的特点有：①即插即用；②热插拔；③有很强的连接能力，采用菊花链形式将众多外设连接起来；④有很好的可扩充性，一个 USB 控制器可扩充高达 127 个外部 USB 设备；⑤高速传输，速率可达 480Mb/s。所以选项 A、B、C 都符合 USB 总线的特点。对于选项 D，USB 是串行总线，不能同时传输 2 位数据。

24. B

USB 是一种连接外部设备的 I/O 总线标准，属于设备总线，是设备和设备控制器之间的接口。而 PCI、AGP、PCI-E 作为计算机系统的局部总线标准，通常用来连接主存、网卡、视频卡等。

25. C

数据线有 32 根，也就是一次可以传送 $32b/8 = 4B$ 的数据，66MHz 意味着有 66M 个时钟周期，而每个时钟周期传送两次数据，可知总线每秒传送的最大数据量为 $66M \times 2 \times 4B = 528\text{MB}$ ，所以总线的最大数据传输率为 528MB/s。

26. C

猝发（突发）传输是在一个总线周期中，可以传输多个存储地址连续的数据，即一次传输一个地址和一批地址连续的数据，并行传输是在传输中有多个数据位同时在设备之间进行的传输，串行传输是指数据的二进制代码在一条物理信道上以位为单位按时间顺序逐位传输的方式，同步

关注公众号【乘龙考研】
一手更新 稳定有保障

传输是指传输过程由统一的时钟控制。

27. C

在同步通信方式中，系统采用一个统一的时钟信号，而不由各设备提供，否则无法实现统一的时钟。

28. A

初看可能会觉得选项 A 正确，并行总线传输通常比串行总线传输速率快，但这不是绝对的。在实际时钟频率较低的情况下，并行总线因为可以同时传输若干比特，速率确实比串行总线快。但是，随着技术的发展，时钟频率越来越高，并行总线之间的相互干扰越来越严重，当时钟频率提高到一定程度时，传输的数据已无法恢复。而串行总线因为导线少，线间干扰容易控制，反而可通过不断提高时钟频率来提高传输速率，选项 A 错误。总线复用是指一种信号线在不同的时间传输不同的信息，它可使用较少的线路传输更多的信息，从而节省空间和成本，因此选项 B 正确。突发（猝发）传输是指在一个总线周期中，可以传输多个存储地址连续的数据，即一次传输一个地址和一批地址连续的数据，选项 C 正确。分离事务通信是总线复用的一种，相比单一的传输线路可以提高总线的利用率，选项 D 正确。

29. D

多总线结构用速率高的总线连接高速设备，用速率低的总线连接低速设备。一般来说，CPU 是计算机的核心，是计算机中速度最快的设备之一，所以选项 A 正确。突发传送方式把多个数据单元作为一个独立传输处理，从而最大化设备的吞吐量。现实中一般用支持突发传送方式的总线来提高存储器的读写效率，选项 B 正确。各总线通过桥接器相连，后者起流量交换作用。PCI-Express 总线都采用串行数据包传输数据，所以选择选项 D。

30. B

总线数据传输率 = 总线工作频率×(总线宽度/8)，所以 I 和 II 会影响总线数据传输率。采用突发（猝发）传输方式，可在总线周期内传输存储地址连续的多个数据字，因此能提高传输效率。采用地址/数据线复用只是减少了线的数量，节省了成本，并不能提高传输率。

31. B

由题目可知，计算机采用 3 通道存储器总线，存储器总线的工作频率为 1333MHz，即 1 秒内传送 1333M 次数据，总线宽度为 64 位即单条总线工作一次可传输 8 字节 (Byte)，因此存储器总线的总带宽为 $3 \times 8 \times 1333\text{MB/s}$ ，约为 32GB/s，所以选择选项 B。

32. C

每个时钟周期传送 2 次，故每秒传送的次数 = 时钟频率×2 = 2.4G×2/s。

总线带宽 = 每秒传送次数×2B×2 = 2.4G×2×2B×2/s = 19.2GB/s。

题中已给出总线带宽公式，降低了难度。公式中的“×2”是因为每次传输 16 位数据，“×2”是因为采用点对点全双工总线，两个方向可同时传输信息。

二、综合应用题

01. 【解答】

1) 总线周期为 7 个时钟周期，总线频率为 66/7MHz。

总线在一个完整的操作周期中传输了一个数据块，总线在一个周期内传输的数据量为 $64\text{bit}/8 \times 6 = 48\text{B}$ ，所以总线的宽度为 48B，传输率为 $48\text{B} \times 66/7\text{MHz} = 452.6\text{MB/s}$ 。

2) 时钟频率减半时的总线频率为 $(66/7)/2\text{MHz}$ ，因数据块大小不变，因此总线宽度仍为 48B，传输率为 $48\text{B} \times 33/7\text{MHz} = 226.3\text{MB/s}$ 。

注意总线周期和时钟周期的联系与区别，总线周期通常由多个时钟周期组成。

02. 【解答】

1) 读操作的时钟周期数： $1 + 2 + 4 = 7$

对应的频率：100MHz/7

总线宽度： $6 \times 4B = 24B$

所以数据传输率 = 总线宽度/读取时间 = $24 \times (100\text{MHz}/7) = 343\text{MB/s}$ 。

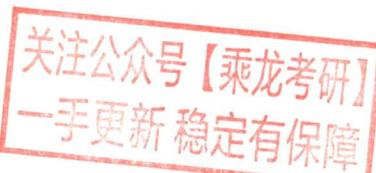
2) 写操作的时钟周期数： $1 + 1 + 4 = 6$

对应的频率：100MHz/6

总线宽度： $6 \times 4B = 24B$

所以数据传输率 = 总线宽度/写操作时间 = $24 \times (100\text{MHz}/6) = 400\text{MB/s}$ 。

3) 平均传输速率 = $1/(0.7/343 + 0.3/400) = 358\text{MB/s}$ 。



6.2 总线事务和定时

总线定时是指总线在双方交换数据的过程中需要时间上配合关系的控制，这种控制称为总线定时，其实质是一种协议或规则，主要有同步和异步两种基本定时方式。

6.2.1 总线事务

从请求总线到完成总线使用的操作序列称为总线事务，它是在一个总线周期中发生的一系列活动。典型的总线事务包括请求操作、仲裁操作、地址传输、数据传输和总线释放。

- 1) 请求阶段。主设备（CPU 或 DMA）发出总线传输请求，并且获得总线控制权。
- 2) 仲裁阶段。总线仲裁机构决定将下一个传输周期的总线使用权授予某个申请者。
- 3) 寻址阶段。主设备通过总线给出要访问的从设备地址及有关命令，启动从模块。
- 4) 传输阶段。主模块和从模块进行数据交换，可单向或双向进行数据传送。
- 5) 释放阶段。主模块的有关信息均从系统总线上撤除，让出总线使用权。

在总线事务的传输阶段，主、从设备之间一般只能传输一个字长的数据。

突发（猝发）传送方式能够进行连续成组数据的传送，其寻址阶段发送的是连续数据单元的首地址，在传输阶段传送多个连续单元的数据，每个时钟周期可以传送一个字长的信息，但是不释放总线，直到一组数据全部传送完毕后，再释放总线。

6.2.2 同步定时方式

所谓同步定时方式，是指系统采用一个统一的时钟信号来协调发送和接收双方的传送定时关系。时钟产生相等的时间间隔，每个间隔构成一个总线周期。在一个总线周期中，发送方和接收方可以进行一次数据传送。因为采用统一的时钟，每个部件或设备发送或接收信息都在固定的总线传送周期中，一个总线的传送周期结束，下一个总线的传送周期开始。

优点：传送速度快，具有较高的传输速率；总线控制逻辑简单。

缺点：主从设备属于强制性同步；不能及时进行数据通信的有效性检验，可靠性较差。

同步通信适用于总线长度较短及总线所接部件的存取时间比较接近的系统。

6.2.3 异步定时方式

在异步定时方式中，没有统一的时钟，也没有固定的时间间隔，完全依靠传送双方相互制约

的“握手”信号来实现定时控制。通常，把交换信息的两个部件或设备分为主设备和从设备，主设备提出交换信息的“请求”信号，经接口传送到从设备；从设备接到主设备的请求后，通过接口向主设备发出“回答”信号。

优点：总线周期长度可变，能保证两个工作速度相差很大的部件或设备之间可靠地进行信息交换，自动适应时间的配合。

缺点：比同步控制方式稍复杂一些，速度比同步定时方式慢。

根据“请求”和“回答”信号的撤销是否互锁，异步定时方式又分为以下3种类型。

- 1) 不互锁方式。主设备发出“请求”信号后，不必等到接到从设备的“回答”信号，而是经过一段时间便撤销“请求”信号。而从设备在接到“请求”信号后，发出“回答”信号，并经过一段时间后自动撤销“回答”信号。双方不存在互锁关系，如图6.4(a)所示。
- 2) 半互锁方式。主设备发出“请求”信号后，必须在接到从设备的“回答”信号后，才撤销“请求”信号，有互锁的关系。而从设备在接到“请求”信号后，发出“回答”信号，但不必等待获知主设备的“请求”信号已经撤销，而是隔一段时间后自动撤销“回答”信号，不存在互锁关系。半互锁方式如图6.4(b)所示。
- 3) 全互锁方式。主设备发出“请求”信号后，必须在从设备“回答”后才撤销“请求”信号；从设备发出“回答”信号后，必须在获知主设备“请求”信号已撤销后，再撤销其“回答”信号。双方存在互锁关系，如图6.4(c)所示。

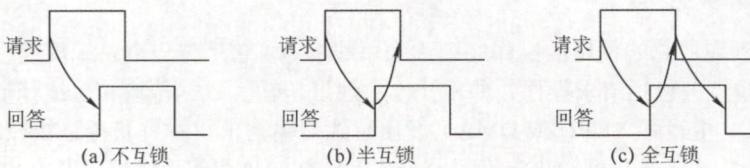


图6.4 请求和回答信号的互锁

6.2.4 本节习题精选

一、单项选择题

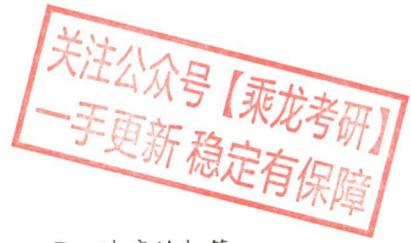
01. 在不同速度的设备之间传送数据，()。
 - A. 必须采用同步控制方式
 - B. 必须采用异步控制方式
 - C. 可以选用同步控制方式，也可选用异步控制方式
 - D. 必须采用应答方式
02. 某机器I/O设备采用异步串行传送方式传送字符信息，字符信息格式为1位起始位、7位数据位、1位校验位和1位停止位。若要求每秒传送480个字符，则该设备的数据传输率为()。

A. 380b/s	B. 4800B/s	C. 480B/s	D. 4800b/s
-----------	------------	-----------	------------
03. 同步控制方式是()。

A. 只适用于CPU控制的方式	B. 只适用于外部设备控制的方式
C. 由统一的时序信号控制的方式	D. 所有指令执行时间都相同的方式
04. 同步通信之所以比异步通信具有较高的传输速率，是因为()。

A. 同步通信不需要应答信号且总线长度较短	B. 同步通信的时钟频率较高
-----------------------	----------------

- B. 同步通信用一个公共的时钟信号进行同步
 C. 同步通信中，各部件的存取时间较接近
 D. 以上各项因素的综合结果
05. 以下各项中，()是同步传输的特点。
 A. 需要应答信号 B. 各部件的存取时间比较接近
 C. 总线长度较长 D. 总线周期长度可变
06. 在异步总线中，传送操作()。
 A. 由设备控制器控制 B. 由CPU控制
 C. 由统一时序信号控制 D. 按需分配时间
07. 总线的异步通信方式是()。
 A. 既不采用时钟信号，又不采用“握手”信号
 B. 只采用时钟信号，不采用“握手”信号
 C. 不采用时钟信号，只采用“握手”信号
 D. 既采用时钟信号，又采用“握手”信号
08. 在各种异步通信方式中，()的速度最快。
 A. 全互锁 B. 半互锁 C. 不互锁 D. 速度均相等
09. 在手术过程中，医生将手伸出，等护士将手术刀递上，待医生握紧后，护士才松手。若把医生和护士视为两个通信模块，上述动作相当于()。
 A. 同步通信 B. 异步通信的全互锁方式
 C. 异步通信的半互锁方式 D. 异步通信的不互锁方式
10. 【2021统考真题】下列关于总线的叙述中，错误的是()。
 A. 总线是在两个或多个部件之间进行数据交换的传输介质
 B. 同步总线由时钟信号定时，时钟频率不一定等于工作频率
 C. 异步总线由握手信号定时，一次握手过程完成一位数据交换
 D. 突发(Burst)传送总线事务可以在总线上连续传送多个数据



二、综合应用题

01. 在异步串行传输方式下，起始位为1位，数据位为7位，偶校验位为1位，停止位为1位，假设每秒传输1200比特，试问有效数据传输率为多少？

6.2.5 答案与解析

一、单项选择题

01. C

在不同速度的设备之间传送数据时，可采用同步方式，也可采用异步方式。异步方式主要用于在不同的设备间进行通信，两种速度不同的设备使用同一时钟进行控制时，采用同步控制方式同样可以进行数据的传送，但不能发挥快速设备的高速性能。

02. D

一个字符占用 $1 + 7 + 1 + 1 = 10$ 位，因此数据传输率为 $10 \times 480 = 4800$ 位/秒。

03. C

同步控制是指由统一时序控制的通信方式，同步通信采用公共时钟，有统一的时钟周期。同步控制既可用于CPU控制，又可用于高速的外部设备控制。

04. D

同步通信采用统一的时钟，每个部件发送或接收信息都在固定的总线传送周期中，一个总线传送周期结束，开始下一个总线传送周期。它适用于总线长度较短且各部件的存取时间较接近的情况，因此具有较高的传输速率。选项 A、B、C 都是正确原因，因此选择选项 D。

05. B

各部件的存取时间比较接近时，最适合采用同步传输，以发挥其优势。

06. D

异步总线即采用异步通信方式的总线。在异步方式下，没有公用的时钟，完全依靠传送双方相互制约的“握手”信号来实现定时控制。传送操作是由双方按需求分配时间的。

07. C

异步通信方式也称应答方式，没有公用的时钟信号，也没有固定的时间间隔，完全依靠传送双方相互制约的“握手”信号来实现定时控制。

08. C

在全互锁、半互锁和不互锁 3 种“握手”方式中，只有不互锁方式的请求信号和回答信号没有相互的制约关系，主设备发出请求信号后，不必等待回答信号的到来，便自己撤销了请求信号，所以速度最快。

09. B

由题意可知，医生是主模块，护士是从模块。医生伸出手后（即主模块发出请求信号），等待护士将手术刀递上（主模块等待从模块的回答信号），护士也必须等待医生握紧后才松开手（从模块等待主模块的回答信号），以上整个流程就是异步通信的全互锁方式。

10. C

总线是在两个或多个设备之间进行通信的传输介质，选项 A 正确。同步总线是指总线通信的双方采用同一个时钟信号，但是一次总线事务不一定在一个时钟周期内完成，即时钟频率不一定等于工作频率，选项 B 正确。异步总线采用握手的方式进行通信，每次握手的过程完成一次通信，但是一次通信往往交换多位而非一位数据，选项 C 错误。突发传送总线事务是指发送方在传输完地址后，连续进行若干次数据的发送，选项 D 正确。

二、综合应用题

01. 【解答】

在这样的一个数据帧中，有效数据位是 7 位，传输过程中发送的代码位共有 $1+7+1+1=10$ 位，所以有效数据传输率为 $1200 \times 7 / (1+7+1+1) = 840 \text{ b/s}$ 。

6.3 本章小结

本章开头提出的问题的参考答案如下。

1) 引入总线结构有什么好处？

引入总线结构主要有以下优点：

① 简化了系统结构，便于系统设计制造。

② 大大减少了连线数目，便于布线，减小体积，提高系统的可靠性。

- ③便于接口设计，所有与总线连接的设备均采用类似的接口。
- ④便于系统的扩充、更新与灵活配置，易于实现系统的模块化。
- ⑤便于设备的软件设计，所有接口的软件对不同的接口地址进行操作。
- ⑥便于故障诊断和维修，同时也能降低成本。

2) 引入总线会导致什么问题？如何解决？

引入总线后，总线上的各个设备分时共享同一总线，当总线上多个设备同时要求使用总线时就会导致总线的冲突。为解决多个主设备同时竞争总线控制权的问题，应当采用总线仲裁部件，以某种方式选择一个主设备优先获得总线控制权，只有获得了总线控制权的设备才能开始数据传送。

6.4 常见问题和易混淆知识点

1. 同一个总线不能既采用同步方式又采用异步方式通信吗？

半同步通信总线可以。这类总线既保留了同步通信的特点，又能采用异步应答方式连接速度相差较大的设备。通过在异步总线中引入时钟信号，其就绪和应答等信号都在时钟的上升沿或下降沿有效，而不受其他时间的信号干扰。

例如，某个采用半同步方式的总线总是从某个时钟开始，在每个时钟到来时，采样 Wait 信号，若无效，则说明数据未准备好，下个时钟到来时，再采样 Wait 信号，直到检测到有效，再去数据线上取数据。PCI 总线也是一种半同步总线，它的所有事件都在时钟下降沿同步，总线设备在时钟开始的上升沿采样总线信号。

2. 一个总线在某一时刻可以有多对主从设备进行通信吗？

不可以。在某个总线周期内，总线上只有一个主设备控制总线，选择一个从设备与之进行通信（即一对一的关系），或对所有设备进行广播通信（即一对多的关系）。所以一个总线在某一时刻不能有多对主从设备进行通信，否则会发生数据冲突。



第 7 章 输入/输出系统

关注公众号【乘龙考研】
一手更新 稳定有保障

【考纲内容】

(一) I/O 接口 (I/O 控制器)

I/O 接口的功能和基本结构; I/O 端口及其编址

(二) I/O 方式

程序查询方式

程序中断方式: 中断的基本概念、中断响应过程、中断处理过程、多重中断和中断屏蔽的概念

DMA 方式: DMA 控制器的组成, DMA 传送过程

扫一扫



视频讲解

【复习提示】

I/O 方式是本章的重点和难点, 每年不仅会以选择题的形式考查基本概念和原理, 而且可能会以综合题的形式考查, 特别是各种 I/O 方式效率的相关计算, 中断方式的各种原理、特点、处理过程、中断屏蔽, DMA 方式的特点、传输过程、与中断方式的区别等。

在学习本章时, 请读者思考以下问题:

- 1) I/O 设备有哪些编址方式? 各有何特点?
- 2) CPU 响应中断应具备哪些条件?

请读者在学习本章的过程中寻找答案, 本章末尾会给出参考答案。

*7.1 I/O 系统基本概念^①

*7.1.1 输入/输出系统

输入/输出是以主机为中心而言的, 将信息从外部设备传送到主机称为输入, 反之称为输出。输入/输出系统解决的主要问题是对于各种形式的信息进行输入和输出的控制。

I/O 系统中的几个基本概念如下:

- 1) 外部设备。包括输入/输出设备及通过输入/输出接口才能访问的外存储设备。
- 2) 接口。在各个外设与主机之间传输数据时进行各种协调工作的逻辑部件。协调包括传输过程中速度的匹配、电平和格式转换等。
- 3) 输入设备。用于向计算机系统输入命令和文本、数据等信息的部件。键盘和鼠标是最基本的输入设备。
- 4) 输出设备。用于将计算机系统中的信息输出到计算机外部进行显示、交换等的部件。显

^① 加“*”的内容表示新大纲中已删除, 仅供学习参考。

示器和打印机是最基本的输出设备。

5) 外存设备。指除计算机内存及 CPU 缓存等外的存储器。如，硬磁盘、光盘等。

一般来说，I/O 系统由 I/O 软件和 I/O 硬件两部分构成：

- 1) I/O 软件。包括驱动程序、用户程序、管理程序、升级补丁等。通常采用 I/O 指令和通道指令实现 CPU 与 I/O 设备的信息交换。
- 2) I/O 硬件。包括外部设备、设备控制器和接口、I/O 总线等。通过设备控制器来控制 I/O 设备的具体动作；通过 I/O 接口与主机（总线）相连。

*7.1.2 I/O 控制方式

在输入/输出系统中，经常需要进行大量的数据传输，而传输过程中有各种不同的 I/O 控制方式，基本的控制方式主要有以下 4 种：

- 1) 程序查询方式。由 CPU 通过程序不断查询 I/O 设备是否已做好准备，从而控制 I/O 设备与主机交换信息。
- 2) 程序中断方式。只在 I/O 设备准备就绪并向 CPU 发出中断请求时才予以响应。
- 3) DMA 方式。主存和 I/O 设备之间有一条直接数据通路，当主存和 I/O 设备交换信息时，无须调用中断服务程序。
- 4) 通道方式。在系统中设有通道控制部件，每个通道都挂接若干外设，主机在执行 I/O 命令时，只需启动有关通道，通道将执行通道程序，从而完成 I/O 操作。

其中，方式 1) 和方式 2) 主要用于数据传输率较低的外部设备，方式 3) 和方式 4) 主要用于数据传输率较高的设备。

*7.1.3 外部设备

最基本的外部设备主要有键盘、鼠标、显示器、打印机、磁盘存储器和光盘存储器等。

1. 输入设备

(1) 键盘

键盘是最常用的输入设备，通过它可发出命令或输入数据。

(2) 鼠标

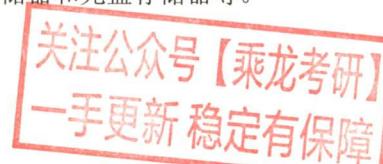
鼠标是常用的定位输入设备，它把用户的操作与计算机屏幕上的位置信息相联系。

2. 输出设备

(1) 显示器

按所用的显示器件分类，有阴极射线管（CRT）显示器、液晶显示器（LCD）、发光二极管（LED）显示器等。显示器属于用点阵方式运行的设备，有以下主要参数。

- 1) 屏幕大小：以对角线长度表示，常用的有 12~29 英寸等。
- 2) 分辨率：能表示的像素个数，屏幕上的每个光点就是一个像素，以宽和高的像素数的乘积表示，如 800×600、1024×768 和 1280×1024 等。
- 3) 灰度级：指黑白显示器中所显示的像素点的亮暗差别，在彩色显示器中则表现为颜色的不同，灰度级越多，图像层次越清楚、逼真，典型的有 8 位（256 级）、16 位等。
- 4) 刷新：光点只能保持极短的时间便会消失，为此必须在光点消失之前再重新扫描显示一遍，这个过程称为刷新。
- 5) 刷新频率：指单位时间内扫描整个屏幕内容的次数。按照人的视觉生理，刷新频率大于 30Hz 时才不会感到闪烁，通常显示器的刷新频率为 60~120Hz。



6) 显示存储器 (VRAM): 也称刷新存储器, 为了不断提高刷新图像的信号, 必须把一帧图像信息存储在刷新存储器中。其存储容量由图像分辨率和灰度级决定, 分辨率越高, 灰度级越多, 刷新存储器容量越大。

$$\text{VRAM 容量} = \text{分辨率} \times \text{灰度级位数}$$

$$\text{VRAM 带宽} = \text{分辨率} \times \text{灰度级位数} \times \text{帧频}$$

(2) 打印机

用于将计算机的处理结果打印在相关介质上。按工作方式, 打印机可分为点阵打印机、针式打印机、喷墨式打印机、激光打印机等。

- 1) 针式打印机。针式打印机擅长“多层复写打印”, 实现各种票据或蜡纸等的打印。其工作原理简单, 造价低廉, 耗材(色带)便宜, 但打印分辨率和打印速度不够高。
- 2) 喷墨式打印机。彩色喷墨打印机基于三基色原理, 即分别喷射3种颜色的墨滴, 按一定的比例混合出所要求的颜色。喷墨式打印机可实现高质量彩色打印。
- 3) 激光打印机。计算机输出的二进制信息, 经过调制后的激光束扫描, 在感光鼓上形成潜像, 再经过显影、转印和定影, 在纸上得到所需的字符或图像。激光打印机打印质量高、速度快、处理能力强, 它是将激光技术和电子显像技术相结合的产物。

3. 外部存储器(辅存)

(1) 磁表面存储器

所谓“磁表面存储”, 是指把某些磁性材料薄薄地涂在金属铝或塑料表面上作为载磁体来存储信息。磁盘存储器、磁带存储器和磁鼓存储器均属于磁表面存储器。

(2) 固态硬盘(SSD)

微小型高档笔记本电脑采用高性能Flash存储器作为硬盘来记录数据, 这种“硬盘”称为固态硬盘(SSD)。固态硬盘除需要Flash存储器外, 还需要其他硬件和软件的支持。

(3) 光盘存储器

光盘存储器是利用光学原理读/写信息的存储装置, 它采用聚焦激光束对盘式介质以非接触方式记录信息。完整的光盘存储系统由光盘片、光盘驱动器、光盘控制器等组成。

7.1.4 本节习题精选

单项选择题

01. 在微型机系统中, I/O设备通过()与主板的系统总线相连接。
A. DMA控制器 B. 设备控制器 C. 中断控制器 D. I/O端口
02. 下列关于I/O指令的说法中, 错误的是()。
A. I/O指令是CPU系统指令的一部分
B. I/O指令是机器指令的一类
C. I/O指令反映CPU和I/O设备交换信息的特点
D. I/O指令的格式和通用指令格式相同
03. 以下关于通道程序的叙述中, 正确的是()。
A. 通道程序存放在主存中 B. 通道程序存放在通道中
C. 通道程序是由CPU执行的 D. 通道程序可以在任何环境下执行I/O操作
04. 下列关于I/O设备的说法中, 正确的是()。
I. 键盘、鼠标、显示器、打印机属于人机交互设备

- II. 在微型计算机中, VGA 代表的是视频传输标准
 III. 打印机从打字原理的角度来区分, 可分为点阵式打印机和活字式打印机
 IV. 鼠标适合于用中断方式来实现输入操作
 A. II、III、IV B. I、II、IV C. I、II、III D. I、II、III、IV
05. 下列说法中, 正确的是()。
 A. 计算机中一个汉字内码在主存中占用 4B
 B. 输出的字型码 16×16 点阵在缓冲存储区中占用 32B
 C. 输出的字型码 16×16 点阵在缓冲存储区中占用 16B
 D. 以上说法都不对
06. 显示汉字采用点阵字库, 若每个汉字用 16×16 的点阵表示, 7500 个汉字的字库容量是()。
 A. 16KB B. 240KB C. 320KB D. 1MB
07. CRT 的分辨率为 1024×1024 像素, 像素的颜色数为 256, 则刷新存储器的每单元字长为(), 总容量为()。
 A. 8B, 256MB B. 8bit, 1MB C. 8bit, 256KB D. 8B, 32MB
08. 【2010 统考真题】假定一台计算机的显示存储器用 DRAM 芯片实现, 若要求显示分辨率为 1600×1200, 颜色深度为 24 位, 帧频为 85Hz, 显存总带宽的 50% 用来刷新屏幕, 则需要的显存总带宽至少约为()。
 A. 245Mb/s B. 979Mb/s C. 1958Mb/s D. 7834Mb/s

7.1.5 答案与解析

单项选择题

01. B

I/O 设备不可能直接与主板总线相连, 它总是通过设备控制器来相连的。

02. D

I/O 指令是指令系统的一部分, 是机器指令的一类, 但其为了反映与 I/O 设备交互的特点, 格式和其他通用指令相比有所不同。

03. A

通道程序存放在主存而非通道中, 由通道从主存中取出并执行。通道程序由通道执行, 且只能在具有通道的 I/O 系统中执行。

04. B

键盘、鼠标、显示器、打印机等都属于机器与人交互的媒介(用户使用键盘、鼠标来控制计算机, 计算机使用显示器和打印机来向用户传递信息), 因此 I 正确; VGA 是一个用于显示的视频传输标准, 因此 II 正确; 打印机从打字原理的角度来分, 可分为击打式和非击打式两种, 按照能否打出汉字来分, 可分为点阵式打印机和活字式打印机, 因此 III 错误; 键盘、鼠标等输入设备一般都采用中断方式来实现, 原因在于 CPU 需要及时响应这些操作, 否则容易造成输入的丢失, 因此 IV 正确。

05. B

计算机中一个汉字内码在主存中占用 2B, 输出的字型码 16×16 点阵在缓冲存储区中占用 $16 \times 16 / 8 = 32B$ 。

06. B

每个汉字用 16×16 点阵表示, 占用 $16 \times 16 / 8 = 32B$, 汉字库容量 = $7500 \times 32B = 240000B \approx$

关注公众号【乘龙考研】
 一手更新 稳定有保障

240KB。

07. B

刷新存储器中存储单元的字长取决于显示的颜色数，颜色数为 m ，字长为 n ，二者的关系为 $2^n = m$ 。本题中的颜色数为 $256 = 2^8$ ，因此刷新存储器单元字长为 8 位。刷新存储器的容量是每个像素点的位数和像素点个数的乘积，因此刷新存储器的容量为 $1024 \times 1024 \times 8\text{bit} = 1\text{MB}$ 。

08. D

刷新所需带宽 = 分辨率×色深×帧频 = $1600 \times 1200 \times 24\text{bit} \times 85\text{Hz} = 3916.8\text{Mb/s}$ ，显存总带宽的 50% 来刷新屏幕，于是需要的显存总带宽至少为 $3916.8 / 0.5 = 7833.6\text{Mb/s} \approx 7834\text{Mb/s}$ 。

7.2 I/O 接口

I/O 接口（I/O 控制器）是主机和外设之间的交接界面，通过接口可以实现主机和外设之间的信息交换。主机和外设具有各自的工作特点，它们在信息形式和工作速度上具有很大的差异，接口正是为了解决这些差异而设置的。

7.2.1 I/O 接口的功能

I/O 接口的主要功能如下：

- 1) 进行地址译码和设备选择。CPU 送来选择外设的地址码后，接口必须对地址进行译码以产生设备选择信息，使主机能和指定外设交换信息。
- 2) 实现主机和外设的通信联络控制。解决主机与外设时序配合问题，协调不同工作速度的外设和主机之间交换信息，以保证整个计算机系统能统一、协调地工作。
- 3) 实现数据缓冲。CPU 与外设之间的速度往往不匹配，为消除速度差异，接口必须设置数据缓冲寄存器，用于数据的暂存，以避免因速度不一致而丢失数据。
- 4) 信号格式的转换。外设与主机两者的电平、数据格式都可能存在差异，接口应提供计算机与外设的信号格式的转换功能，如电平转换、并/串或串/并转换、模/数或数/模转换等。
- 5) 传送控制命令和状态信息。CPU 要启动某一外设时，通过接口中的命令寄存器向外设发出启动命令；外设准备就绪时，则将“准备好”状态信息送回接口中的状态寄存器，并反馈给 CPU。外设向 CPU 提出中断请求时，CPU 也应有相应的响应信号反馈给外设。

7.2.2 I/O 接口的基本结构

如图 7.1 所示，I/O 接口在主机侧通过 I/O 总线与内存、CPU 相连。通过数据总线，在数据缓冲寄存器与内存或 CPU 的寄存器之间进行数据传送。同时接口和设备的状态信息被记录在状态寄存器中，通过数据线将状态信息送到 CPU。CPU 对外设的控制命令也通过数据线传送，一般将其送到 I/O 接口的控制寄存器。状态寄存器和控制寄存器在传送方向上是相反的。

接口中的地址线用于给出要访问的 I/O 接口中的寄存器的地址，它和读/写控制信号一起被送到 I/O 接口的控制逻辑部件，通过控制线传送来的读/写信号确认是读寄存器还是写寄存器，此外控制线还会传送一些仲裁信号和握手信号。

接口中的 I/O 控制逻辑还要能对控制寄存器中的命令字进行译码，并将译码得到的控制信号通过外设界面控制逻辑送到外设，同时将数据缓冲寄存器的数据发送到外设或从外设接收数据到数据缓冲寄存器。另外，它还要具有收集外设状态到状态寄存器的功能。

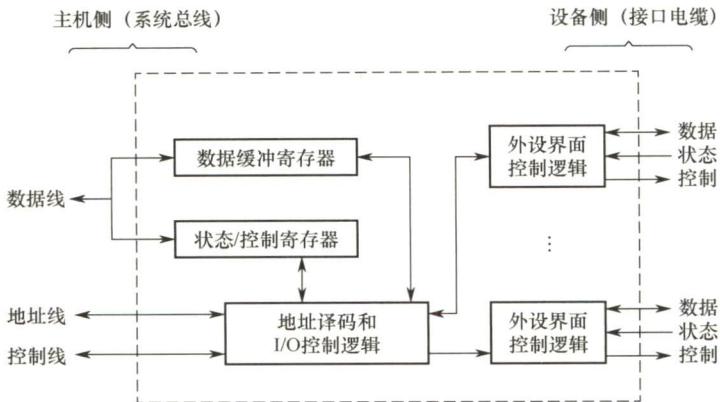


图 7.1 I/O 接口的基本结构

对数据缓冲寄存器、状态/控制寄存器的访问操作是通过相应的指令来完成的，通常称这类指令为 I/O 指令，I/O 指令只能在操作系统内核的底层 I/O 软件中使用，它们是一种特权指令。

注意：接口和端口是两个不同的概念。端口是指接口电路中可以进行读/写的寄存器，若干端口加上相应的控制逻辑才可以组成接口。

7.2.3 I/O 接口的类型

从不同的角度看，I/O 接口可以分为不同的类型。

关注公众号【乘龙考研】
一手更新 稳定有保障

- 1) 按数据传送方式可分为并行接口（一字节或一个字的所有位同时传送）和串行接口（一位一位地传送），接口要完成数据格式的转换。

注意：这里所说的数据传送方式指的是外设和接口一侧的传送方式。

- 2) 按主机访问 I/O 设备的控制方式可分为程序查询接口、中断接口和 DMA 接口等。
- 3) 按功能选择的灵活性可分为可编程接口和不可编程接口。

7.2.4 I/O 端口及其编址

I/O 端口是指接口电路中可被 CPU 直接访问的寄存器，主要有数据端口、状态端口和控制端口，若干端口加上相应的控制逻辑组成接口。通常，CPU 能对数据端口执行读写操作，但对状态端口只能执行读操作，对控制端口只能执行写操作。

I/O 端口要想能够被 CPU 访问，就必须要对各个端口进行编号，每个端口对应一个端口地址。而对 I/O 端口的编址方式有与存储器统一编址和独立编址两种。

- 1) 统一编址，又称存储器映射方式，是指把 I/O 端口当作存储器的单元进行地址分配，这种方式 CPU 不需要设置专门的 I/O 指令，用统一的访存指令就可以访问 I/O 端口。
优点：不需要专门的输入/输出指令，可使 CPU 访问 I/O 的操作更灵活、更方便，还可使端口有较大的编址空间。**缺点：**端口占用存储器地址，使内存容量变小，而且利用存储器编址的 I/O 设备进行数据输入/输出操作，执行速度较慢。
- 2) 独立编址，又称 I/O 映射方式，I/O 端口的地址空间与主存地址空间是两个独立的地址空间，因而无法从地址码的形式上区分，需要设置专门的 I/O 指令来访问 I/O 端口。
优点：输入/输出指令与存储器指令有明显区别，程序编制清晰，便于理解。**缺点：**输入/输出指令少，一般只能对端口进行传送操作，尤其需要 CPU 提供存储器读/写、I/O 设备读/

写两组控制信号，增加了控制的复杂性。

7.2.5 本节习题精选

单项选择题

01. 在统一编址的方式下，区分存储单元和 I/O 设备是靠（ ）。
 - A. 不同的地址码
 - B. 不同的地址线
 - C. 不同的控制线
 - D. 不同的数据线
02. 下列功能中，属于 I/O 接口的功能的是（ ）。
 - I. 数据格式的转换
 - II. I/O 过程中错误与状态检测
 - III. I/O 操作的控制与定时
 - IV. 与主机和外设通信
 - A. I、IV
 - B. I、III、IV
 - C. I、II、IV
 - D. I、II、III、IV
03. 下列关于 I/O 端口和接口的说法中，正确的是（ ）。
 - A. 按照不同的数据传送格式，可将接口分为同步传送接口和异步传送接口
 - B. 在统一编址方式下，存储单元和 I/O 设备是靠不同的地址线来区分的
 - C. 在独立编址方式下，存储单元和 I/O 设备是靠不同的地址线来区分的
 - D. 在独立编址方式下，CPU 需要设置专门的输入/输出指令访问端口
04. I/O 的编址方式采用统一编址方式时，进行输入/输出的操作的指令是（ ）。
 - A. 控制指令
 - B. 访存指令
 - C. 输入/输出指令
 - D. 都不对
05. 下列叙述中，正确的是（ ）。
 - A. 只有 I/O 指令可以访问 I/O 设备
 - B. 在统一编址下，不能直接访问 I/O 设备
 - C. 访问存储器的指令一定不能访问 I/O 设备
 - D. 只有在具有专门 I/O 指令的计算机中，I/O 设备才可以单独编址
06. 在统一编址的情况下，就 I/O 设备而言，其对应的 I/O 地址不可取的是（ ）。
 - A. 要求固定在地址高端
 - B. 要求固定在地址低端
 - C. 要求相对固定在地址的某部分
 - D. 可以随意在地址的任何地方
07. 磁盘驱动器向盘片磁道记录数据时采用（ ）方式写入。
 - A. 并行
 - B. 串行
 - C. 并行-串行
 - D. 串行-并行
08. 程序员进行系统调用访问设备使用的是（ ）。
 - A. 逻辑地址
 - B. 物理地址
 - C. 主设备地址
 - D. 从设备地址
09. 【2012 统考真题】下列选项中，在 I/O 总线的数据线上传输的信息包括（ ）。
 - I. I/O 接口中的命令字
 - II. I/O 接口中的状态字
 - III. 中断类型号
 - A. 仅 I、II
 - B. 仅 I、III
 - C. 仅 II、III
 - D. I、II、III
10. 【2014 统考真题】下列有关 I/O 接口的叙述中，错误的是（ ）。
 - A. 状态端口和控制端口可以公用同一个寄存器
 - B. I/O 接口中 CPU 可访问的寄存器称为 I/O 端口
 - C. 采用独立编址方式时，I/O 端口地址和主存地址可能相同
 - D. 采用统一编址方式时，CPU 不能用访存指令访问 I/O 端口
11. 【2017 统考真题】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

12. 【2021 统考真题】下列选项中，不属于 I/O 接口的是（ ）。

- A. 磁盘驱动器 B. 打印机适配器 C. 网络控制器 D. 可编程中断控制器

7.2.6 答案与解析

单项选择题

01. A

在统一编址的情况下，没有专门的 I/O 指令，因此用访存指令来实现 I/O 操作，区分存储单元和 I/O 设备是靠它们各自不同的地址码。

02. D

I/O 接口的功能有：①选址功能、②传送命令功能、③传送数据功能、④反映 I/O 设备工作状态的功能。选项 I 可参考唐朔飞的《计算机组成原理》教材，为设置接口的原因之一，也是接口应具有的功能；选项 II 属于④；选项 III 属于②；选项 IV 属于③。

03. D

选项 D 显然正确。按照不同的数据传送格式，可将接口分为并行接口和串行接口，因此选项 A 错；在统一编址方式下，存储单元和 I/O 设备是靠不同的地址码而非地址线来区分的，因此选项 B 错；在独立编址方式下，存储单元和 I/O 设备是靠不同的指令来区分的，因此选项 C 错。

04. B

统一编址时，直接使用指令系统中的访存指令来完成输入/输出操作；独立编址时，则需要使用专门的输入/输出指令来完成输入/输出操作。

05. D

在统一编址的情况下，访存指令也可访问 I/O 设备，因此选项 A、B、C 错误。在独立编址的方式下，访问 I/O 地址空间必须通过专门的 I/O 指令，因此选项 D 正确。

06. D

在统一编址方式下，指令靠地址码区分内存和 I/O 设备，若随意在地址的任何地方编址，将给编程造成极大的混乱，因此选项 D 错误。选项 A、B、C 的做法都是可取的。

07. B

磁盘驱动器向盘片磁道记录数据时采用串行方式写入。

08. A

物理地址是外部连接使用的，且是唯一的，它与“地址总线相对应”；而逻辑地址是内部和编程使用的，并不唯一。在内存中的实际地址就是所谓的“物理地址”，而逻辑地址就是用于逻辑段管理内存的，因此程序员使用逻辑地址访问设备。

09. D

I/O 总线分为三类：数据线、控制线和地址线。数据缓冲寄存器和命令/状态寄存器的内容都是通过数据线来传送的；地址线用以传送与 CPU 交换数据的端口地址；而控制线用以给 I/O 端口发送读/写信号，只是用来对端口进行读/写控制的。因此 I、II 和 III 均正确。

10. D

采用统一编址时，CPU 访存和访问 I/O 端口用的是一样的指令，所以访存指令可访问 I/O 端口，D 选项错误。其他三个选项均为正确陈述。

11. D

I/O 端口是指 I/O 接口中用于缓冲信息的寄存器，由于主机和 I/O 设备的工作方式和工作速度有很大差异，I/O 端口应运而生。在执行一条指令时，CPU 使用地址总线选择所请求的 I/O 端口，

关注公众号【乘龙考研】

一手更新 稳定有保障

使用数据总线在 CPU 寄存器和端口之间传输数据，所以选择选项 D。

12. A

I/O 接口即 I/O 控制器，其功能是接收主机发送的 I/O 控制信号，并实现主机和外部设备之间的信息交换。磁盘驱动器是由磁头、磁盘和读写电路等组成的，也就是我们平常所说的磁盘本身，A 错误。B、C 和 D 均为 I/O 控制器。

7.3 I/O 方式

输入/输出系统实现主机与 I/O 设备之间的数据传送，可以采用不同的控制方式，各种方式在代价、性能、解决问题的着重点等方面各不相同，常用的 I/O 方式有程序查询、程序中断、DMA 和通道等，其中前两种方式更依赖于 CPU 中程序指令的执行。

7.3.1 程序查询方式

信息交换的控制完全由 CPU 执行程序实现，程序查询方式接口中设置一个数据缓冲寄存器（数据端口）和一个设备状态寄存器（状态端口）。主机进行 I/O 操作时，先发出询问信号，读取设备的状态并根据设备状态决定下一步操作究竟是进行数据传送还是等待。

程序查询方式的工作流程如下（见图 7.2）：

- ① CPU 执行初始化程序，并预置传送参数。
- ② 向 I/O 接口发出命令字，启动 I/O 设备。
- ③ 从外设接口读取其状态信息。
- ④ CPU 不断查询 I/O 设备状态，直到外设准备就绪。
- ⑤ 传送一次数据。
- ⑥ 修改地址和计数器参数。
- ⑦ 判断传送是否结束，若未结束转第③步，直到计数器为 0。

在这种控制方式下，CPU 一旦启动 I/O，就必须停止现行程序的运行，并在现行程序中插入一段程序。程序查询方式的主要特点是 CPU 有“踏步”等待现象，CPU 与 I/O 串行工作。这种方式的接口设计简单、设备量少，但 CPU 在信息传送过程中要花费很多时间来查询和等待，而且在一段时间内只能和一台外设交换信息，效率大大降低。

7.3.2 程序中断方式

1. 程序中断的基本概念

程序中断是指在计算机执行现行程序的过程中，出现某些急需处理的异常情况或特殊请求，CPU 暂时中止现行程序，而转去对这些异常情况或特殊请求进行处理，处理完毕后再返回到现行程序的断点处，继续执行原程序。早期的中断技术是为了处理数据传送。

随着计算机的发展，中断技术不断被赋予新的功能，主要功能有：

- ① 实现 CPU 与 I/O 设备的并行工作。
- ② 处理硬件故障和软件错误。

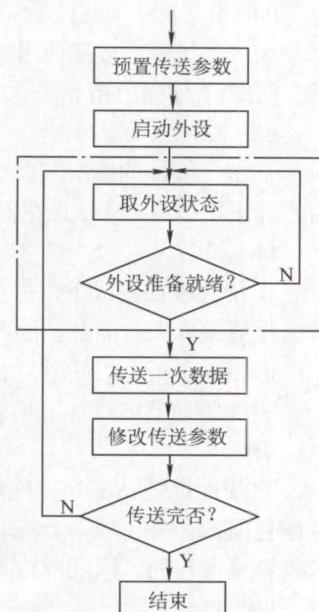


图 7.2 程序查询方式流程图

- ③ 实现人机交互，用户干预机器需要用到中断系统。
- ④ 实现多道程序、分时操作，多道程序的切换需借助于中断系统。
- ⑤ 实时处理需要借助中断系统来实现快速响应。
- ⑥ 实现应用程序和操作系统（管态程序）的切换，称为“软中断”。
- ⑦ 多处理器系统中各处理器之间的信息交流和任务切换。

程序中断方式的思想：CPU 在程序中安排好在某个时机启动某台外设，然后 CPU 继续执行当前的程序，不需要像查询方式那样一直等待外设准备就绪。一旦外设完成数据传送的准备工作，就主动向 CPU 发出中断请求，请求 CPU 为自己服务。在可以响应中断的条件下，CPU 暂时中止正在执行的程序，转去执行中断服务程序为外设服务，在中断服务程序中完成一次主机与外设之间的数据传送，传送完成后，CPU 返回原来的程序，如图 7.3 所示。

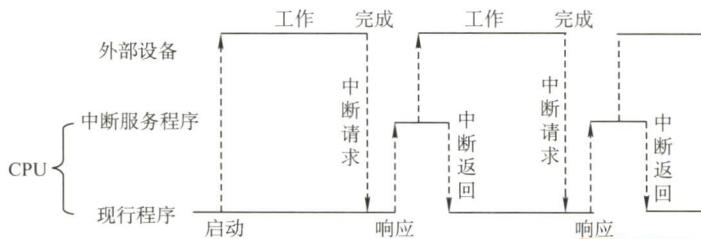


图 7.3 程序中断方式示意图

关注公众号【乘龙考研】
一手更新稳定有保障

2. 程序中断的工作流程

(1) 中断请求

中断源是请求 CPU 中断的设备或事件，一台计算机允许有多个中断源。每个中断源向 CPU 发出中断请求的时间是随机的。为记录中断事件并区分不同的中断源，中断系统需对每个中断源设置中断请求标记触发器，当其状态为“1”时，表示中断源有请求。这些触发器可组成中断请求标记寄存器，该寄存器可集中在 CPU 中，也可分散在各个中断源中。

通过 INTR 线发出的是可屏蔽中断，通过 NMI 线发出的是不可屏蔽中断。可屏蔽中断的优先级最低，在关中断模式下不会被响应。不可屏蔽中断用于处理紧急和重要的事件，如时钟中断、电源掉电等，其优先级最高，其次是内部异常，即使在关中断模式下也会被响应。

(2) 中断响应判优

中断响应优先级是指 CPU 响应中断请求的先后顺序。由于许多中断源提出中断请求的时间都是随机的，因此当多个中断源同时提出请求时，需通过中断判优逻辑来确定响应哪个中断源的请求，中断响应的判优通常是通过硬件排队器实现的。

一般来说，①不可屏蔽中断 > 内部异常 > 可屏蔽中断；②内部异常中，硬件故障 > 软件中断；③DMA 中断请求优先于 I/O 设备传送的中断请求；④在 I/O 传送类中断请求中，高速设备优先于低速设备，输入设备优先于输出设备，实时设备优先于普通设备。

注意：中断优先级包括响应优先级和处理优先级，响应优先级在硬件线路上是固定的，不便改动。处理优先级可利用中断屏蔽技术动态调整，以实现多重中断，具体在后文中介绍。

(3) CPU 响应中断的条件

CPU 在满足一定的条件下响应中断源发出的中断请求，并经过一些特定的操作，转去执行中断服务程序。CPU 响应中断必须满足以下 3 个条件：

- ① 中断源有中断请求。
- ② CPU 允许中断及开中断（异常和不可屏蔽中断不受此限制）。
- ③ 一条指令执行完毕（异常不受此限制），且没有更紧迫的任务。

注意：I/O 设备的就绪时间是随机的，而 CPU 在统一的时刻即每条指令执行阶段结束前向接口发出中断查询信号，以获取 I/O 的中断请求，也就是说，CPU 响应中断的时间是在每条指令执行阶段的结束时刻。这里说的中断仅指 I/O 中断，内部异常不属于此类情况。

(4) 中断响应过程

CPU 响应中断后，经过某些操作，转去执行中断服务程序。这些操作是由硬件直接实现的，我们将它称为中断隐指令。中断隐指令并不是指令系统中的一条真正的指令，只是一种虚拟的说法，本质上是硬件的一系列自动操作。它所完成的操作如下：

- ① 关中断。CPU 响应中断后，首先要保护程序的断点和现场信息，在保护断点和现场的过程中，CPU 不能响应更高级中断源的中断请求。否则，若断点或现场保存不完整，在中断服务程序结束后，就不能正确地恢复并继续执行现行程序。
- ② 保存断点。为保证在中断服务程序执行完后能正确地返回到原来的程序，必须将原程序的断点（指令无法直接读取的 PC 和 PSW 的内容）保存在栈或特定寄存器中^①。
- 注意异常和中断的差异：异常指令通常并没有执行成功，异常处理后要重新执行，所以其断点是当前指令的地址。中断的断点则是下一条指令的地址。
- ③ 引出中断服务程序。识别中断源，将对应的服务程序入口地址送入程序计数器 PC。有两种方法识别中断源：硬件向量法和软件查询法。本节主要讨论比较常用的向量中断。

(5) 中断向量

中断识别分为向量中断和非向量中断两种。非向量中断即软件查询法，第 5 章已介绍。

每个中断都有一个唯一的类型号，每个中断类型号都对应一个中断服务程序，每个中断服务程序都有一个入口地址，CPU 必须找到入口地址，即中断向量。把系统中的全部中断向量集中存放到存储器的某个区域内，这个存放中断向量的存储区就称为中断向量表。

CPU 响应中断后，通过识别中断源获得中断类型号，然后据此计算出对应中断向量的地址；再根据该地址从中断向量表中取出中断服务程序的入口地址，并送入程序计数器 PC，以转而执行中断服务程序，这种方法被称为中断向量法，采用中断向量法的中断被称为向量中断。

(6) 中断处理过程

不同计算机的中断处理过程各具特色，就其多数而言，中断处理流程如图 7.4 所示。

中断处理流程如下：

- ① 关中断。
- ② 保存断点。
- ③ 中断服务程序寻址。

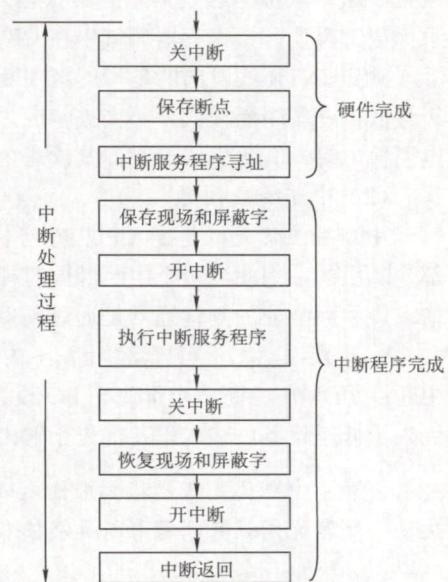


图 7.4 可嵌套中断的处理流程

^① x86 机器保存 PC 和 PSW 到内存栈中；MIPS 机器没有 PSW，只保存 PC 到特定寄存器中。

④ 保存现场和屏蔽字。进入中断服务程序后首先要保存现场和中断屏蔽字，现场信息是指用户可见的工作寄存器的内容，它存放着程序执行到断点处的现行值。

注意：现场和断点，这两类信息都不能被中断服务程序破坏。现场信息因为用指令可直接访问，所以通常在中断服务程序中通过指令把它们保存到栈中，即由软件实现；而断点信息由CPU在中断响应时自动保存到栈或指定的寄存器中，即由硬件实现。

⑤ 开中断。允许更高级中断请求得到响应，实现中断嵌套。

⑥ 执行中断服务程序。这是中断请求的目的。

⑦ 关中断。保证在恢复现场和屏蔽字时不被中断。

⑧ 恢复现场和屏蔽字。将现场和屏蔽字恢复到原来的状态。

⑨ 开中断、中断返回。中断服务程序的最后一条指令通常是一条中断返回指令，使其返回到原程序的断点处，以便继续执行原程序。

其中，①~③由中断隐指令（硬件自动）完成；④~⑨由中断服务程序完成。

注意：恢复现场是指在中断返回前，必须将寄存器的内容恢复到中断处理前的状态，这部分工作由中断服务程序完成。中断返回由中断服务程序的最后一条中断返回指令完成。

3. 多重中断和中断屏蔽技术

若CPU在执行中断服务程序的过程中，又出现了新的更高优先级的中断请求，则这种中断称为单重重断，如图7.5(a)所示。若CPU暂停现行的中断服务程序，转去处理新的中断请求，则这种中断称为多重中断，又称中断嵌套，如图7.5(b)所示。

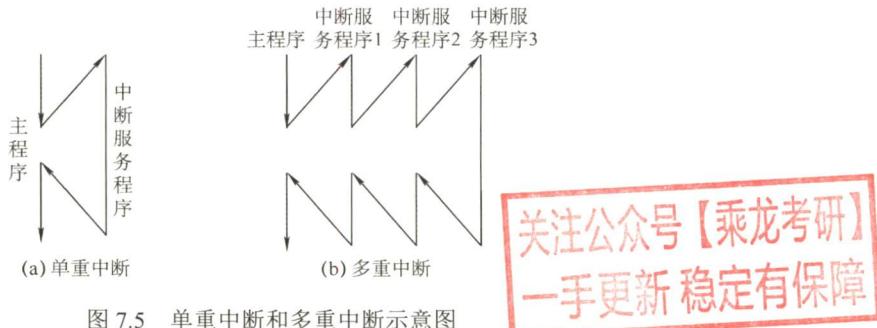


图7.5 单重重断和多重中断示意图

CPU要具备多重中断的功能，必须满足下列条件：

- ① 在中断服务程序中提前设置开中断指令。
- ② 优先级别高的中断源有权中断优先级别低的中断源。

中断处理优先级是指多重中断的实际优先级处理次序，可以利用中断屏蔽技术动态调整，从而可以灵活地调整中断服务程序的优先级，使中断处理更加灵活。如果不使用中断屏蔽技术，则处理优先级和响应优先级相同。现代计算机一般使用中断屏蔽技术，每个中断源都有一个屏蔽触发器，1表示屏蔽该中断源的请求，0表示可以正常申请，所有屏蔽触发器组合在一起便构成一个屏蔽字寄存器，屏蔽字寄存器的内容称为屏蔽字。

关于中断屏蔽字的设置及多重中断程序执行的轨迹，下面通过实例说明。

【例7.1】设某机有4个中断源A、B、C、D，其硬件排队优先次序为A>B>C>D，现要求将中断处理次序改为D>A>C>B。

- 1) 写出每个中断源对应的屏蔽字。
- 2) 按图 7.6 所示的时间轴给出的 4 个中断源的请求时刻，画出 CPU 执行程序的轨迹。设每个中断源的中断服务程序时间为 $20\mu\text{s}$ 。

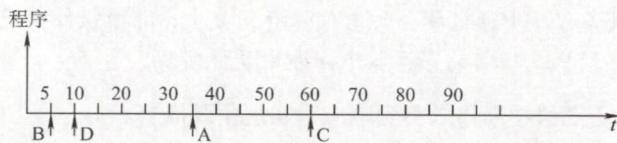


图 7.6 中断源的请求时刻

解：

- 1) 在中断处理次序改为 $D > A > C > B$ 后，D 具有最高优先级，可以屏蔽其他所有中断，且不能中断自身，因此 D 对应的屏蔽字为 1111；A 具有次高优先级，只能被 D 中断，因此 A 对应的屏蔽字为 1110，以此类推，得到 4 个中断源的屏蔽字，见表 7.1。

表 7.1 中断源对应的中断屏蔽字

中断源	屏蔽字			
	A	B	C	D
A	1	1	1	0
B	0	1	0	0
C	0	1	1	0
D	1	1	1	1

- 2) 根据处理次序，在时刻 5，B 发中断请求，获得 CPU；在时刻 10，D 发中断请求，此时 B 虽还未执行完毕，但 D 的优先级高于 B，于是 D 中断 B 而获得 CPU；在时刻 30，D 执行完毕，B 继续获得 CPU；在时刻 35，A 发中断请求，此时 B 虽还未执行完毕，但 A 的优先级高于 B，于是 A 中断 B 而获得 CPU，如此继续下去，执行轨迹如图 7.7 所示。

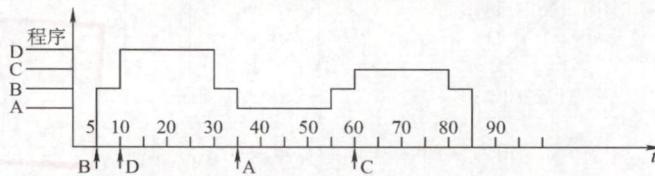


图 7.7 CPU 执行程序的轨迹

7.3.3 DMA 方式

DMA 方式是一种完全由硬件进行成组信息传送的控制方式，它具有程序中断方式的优点，即在数据准备阶段，CPU 与外设并行工作。DMA 方式在外设与内存之间开辟一条“直接数据通道”，信息传送不再经过 CPU，降低了 CPU 在传送数据时的开销，因此称为直接存储器存取方式。由于数据传送不经过 CPU，也就不需要保护、恢复 CPU 现场等繁琐操作。

这种方式适用于磁盘、显卡、声卡、网卡等高速设备大批量数据的传送，它的硬件开销比较大。在 DMA 方式中，中断的作用仅限于故障和正常传送结束时的处理。

1. DMA 方式的特点

主存和 DMA 接口之间有一条直接数据通路。由于 DMA 方式传送数据不需要经过 CPU，因

此不必中断现行程序，I/O 与主机并行工作，程序和传送并行工作。

DMA 方式具有下列特点：

- ① 它使主存与 CPU 的固定联系脱钩，主存既可被 CPU 访问，又可被外设访问。
- ② 在数据块传送时，主存地址的确定、传送数据的计数等都由硬件电路直接实现。
- ③ 主存中要开辟专用缓冲区，及时供给和接收外设的数据。
- ④ DMA 传送速度快，CPU 和外设并行工作，提高了系统效率。
- ⑤ DMA 在传送开始前要通过程序进行预处理，结束后要通过中断方式进行后处理。

2. DMA 控制器的组成

在 DMA 方式中，对数据传送过程进行控制的硬件称为 DMA 控制器（DMA 接口）。当 I/O 设备需要进行数据传送时，通过 DMA 控制器向 CPU 提出 DMA 传送请求，CPU 响应之后将让出系统总线，由 DMA 控制器接管总线进行数据传送。其主要功能如下：

- 1) 接受外设发出的 DMA 请求，并向 CPU 发出总线请求。
- 2) CPU 响应并发出总线响应信号，DMA 接管总线控制权，进入 DMA 操作周期。
- 3) 确定传送数据的主存单元地址及长度，并自动修改主存地址计数和传送长度计数。
- 4) 规定数据在主存和外设间的传送方向，发出读写等控制信号，执行数据传送操作。
- 5) 向 CPU 报告 DMA 操作结束。

图 7.8 给出了一个简单的 DMA 控制器。

- 主存地址计数器：存放要交换数据的主存地址。
- 传送长度计数器：记录传送数据的长度，计数溢出时，数据即传送完毕，自动发中断请求信号。
- 数据缓冲寄存器：暂存每次传送的数据。
- DMA 请求触发器：每当 I/O 设备准备好数据后，给出一个控制信号，使 DMA 请求触发器置位。
- “控制/状态”逻辑：由控制和时序电路及状态标志组成，用于指定传送方向，修改传送参数，并对 DMA 请求信号、CPU 响应信号进行协调和同步。
- 中断机构：当一个数据块传送完毕后触发中断机构，向 CPU 提出中断请求。

关注公众号【乘龙考研】
一手更新 稳定有保障

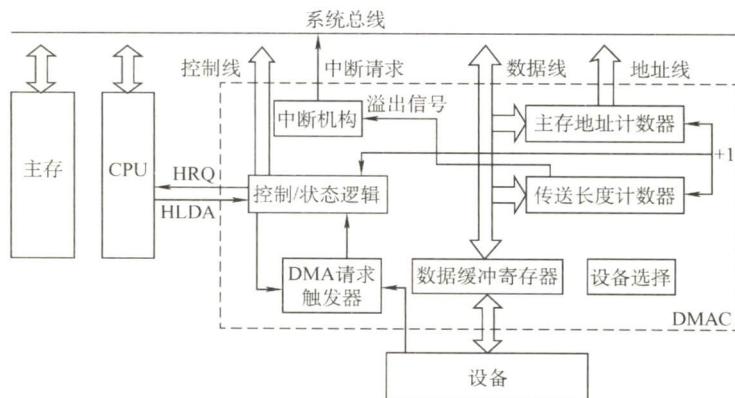


图 7.8 简单的 DMA 控制器

在 DMA 传送过程中，DMA 控制器将接管 CPU 的地址总线、数据总线和控制总线，CPU 的主存控制信号被禁止使用。而当 DMA 传送结束后，将恢复 CPU 的一切权利并开始执行其操作。

由此可见，DMA 控制器必须具有控制系统总线的能力。

3. DMA 的传送方式

主存和 I/O 设备之间交换信息时，不通过 CPU。但当 I/O 设备和 CPU 同时访问主存时，可能发生冲突，为了有效地使用主存，DMA 控制器与 CPU 通常采用以下 3 种方式使用主存：

- 1) 停止 CPU 访存。当 I/O 设备有 DMA 请求时，由 DMA 控制器向 CPU 发送一个停止信号，使 CPU 脱离总线，停止访问主存，直到 DMA 传送一块数据结束。数据传送结束后，DMA 控制器通知 CPU 可以使用主存，并把总线控制权交还给 CPU。
- 2) 周期挪用（或周期窃取）。当 I/O 设备有 DMA 请求时，会遇到 3 种情况：①是此时 CPU 不在访存（如 CPU 正在执行乘法指令），因此 I/O 的访存请求与 CPU 未发生冲突；②是 CPU 正在访存，此时必须待存取周期结束后，CPU 再将总线占有权让出；③是 I/O 和 CPU 同时请求访存，出现访存冲突，此时 CPU 要暂时放弃总线占有权。I/O 访存优先级高于 CPU 访存，因为 I/O 不立即访存就可能丢失数据，此时由 I/O 设备挪用一个或几个存取周期，传送完一个数据后立即释放总线，是一种单字传送方式。
- 3) DMA 与 CPU 交替访存。这种方式适用于 CPU 的工作周期比主存存取周期长的情况。例如，若 CPU 的工作周期是 $1.2\mu s$ ，主存的存取周期小于 $0.6\mu s$ ，则可将一个 CPU 周期分为 C_1 和 C_2 两个周期，其中 C_1 专供 DMA 访存， C_2 专供 CPU 访存。这种方式不需要总线使用权的申请、建立和归还过程，总线使用权是通过 C_1 和 C_2 分时控制的。

4. DMA 的传送过程

DMA 的数据传送过程分为预处理、数据传送和后处理 3 个阶段：

- 1) 预处理。由 CPU 完成一些必要的准备工作。首先，CPU 执行几条 I/O 指令，用以测试 I/O 设备状态，初始化 DMA 控制器中的有关寄存器、设置传送方向、启动该设备等。然后，CPU 继续执行原来的程序，直到 I/O 设备准备好发送的数据（输入情况）或接收的数据（输出情况）时，I/O 设备向 DMA 控制器发送 DMA 请求，再由 DMA 控制器向 CPU 发送总线请求（有时将这两个过程统称为 DMA 请求），用以传输数据。
- 2) 数据传送。DMA 的数据传输可以以单字节（或字）为基本单位，也可以以数据块为基本单位。对于以数据块为单位的传送（如硬盘），DMA 占用总线后的数据输入和输出操作都是通过循环来实现的。需要指出的是，这一循环也是由 DMA 控制器（而非通过 CPU 执行程序）实现的，即数据传送阶段完全由 DMA（硬件）控制。
- 3) 后处理。DMA 控制器向 CPU 发送中断请求，CPU 执行中断服务程序做 DMA 结束处理，包括校验送入主存的数据是否正确、测试传送过程中是否出错（错误则转诊断程序）及决定是否继续使用 DMA 传送其他数据等。DMA 的传送流程如图 7.9 所示。

5. DMA 方式和中断方式的区别

DMA 方式和中断方式的重要区别如下：

- ① 中断方式是程序的切换，需要保护和恢复现场；而 DMA 方式不中断现行程序，无需保护现场，除了预处理和后处理，其他时候不占用任何 CPU 资源。
- ② 对中断请求的响应只能发生在每条指令执行结束时（执行周期后）；而对 DMA 请求的响应可以发生在任意一个机器周期结束时（取指、间址、执行周期后均可）。
- ③ 中断传送过程需要 CPU 的干预；而 DMA 传送过程不需要 CPU 的干预，因此数据传输率非常高，适合于高速外设的成组数据传送。

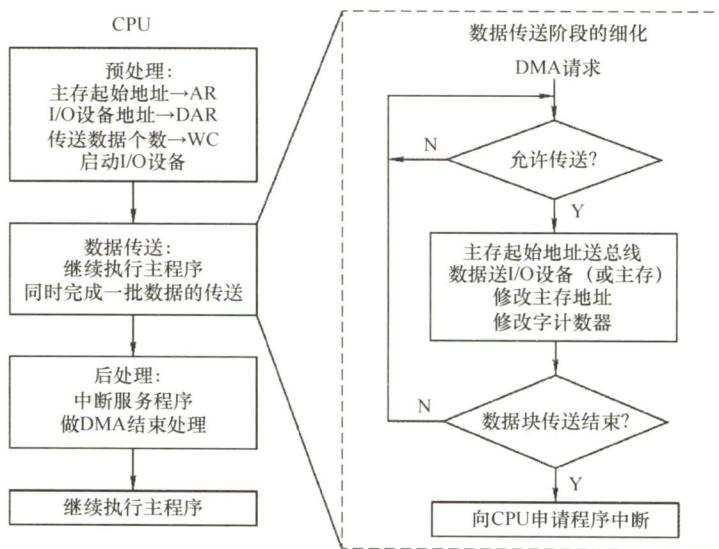


图 7.9 DMA 的传送流程

- ④ DMA 请求的优先级高于中断请求。
- ⑤ 中断方式具有处理异常事件的能力，而 DMA 方式仅局限于大批数据的传送。
- ⑥ 从数据传送来看，中断方式靠程序传送，DMA 方式靠硬件传送。

7.3.4 本节习题精选

一、单项选择题

01. 设置中断排队判优逻辑的目的是（ ）。
- 产生中断源编码
 - 使同时提出的请求中的优先级别最高者得到及时响应
 - 使 CPU 能方便地转入中断服务子程序
 - 提高中断响应速度
02. 以下说法中，错误的是（ ）。
- 中断服务程序一般是操作系统模块
 - 中断向量方法可提高中断源的识别速度
 - 中断向量地址是中断服务程序的入口地址
 - 重叠处理中断的现象称为中断嵌套
03. 当有中断源发出请求时，CPU 可执行相应的中断服务程序，可以提出中断的有（ ）。
- 外部事件
 - Cache
 - 虚拟存储器失效
 - 浮点数运算下溢
 - 浮点数运算上溢
- I、III 和 IV
 - I 和 V
 - I、II 和 V
 - I、III 和 V
04. 关于程序中断方式和 DMA 方式的叙述，错误的是（ ）。
- DMA 的优先级比程序中断的优先级要高
 - 程序中断方式需要保护现场，DMA 方式不需要保护现场
 - 程序中断方式的中断请求是为了报告 CPU 数据的传输结束，而 DMA 方式的中断请求完全是为了传送数据

关注公众号【乘龙考研】
一手更新 稳定有保障

- A. 仅 II B. II、III C. 仅 III D. I、III
05. 下列说法中，错误的是（ ）。
 - I. 程序中断过程是由硬件和中断服务程序共同完成的
 - II. 在每条指令的执行过程中，每个总线周期要检查一次有无中断请求
 - III. 检测有无 DMA 请求，一般安排在一条指令执行过程的末尾
 - IV. 中断服务程序的最后指令是无条件转移指令
 A. III、IV B. II、III、IV C. II、IV D. I、II、III、IV
06. 能产生 DMA 请求的总线部件是（ ）。
 - I. 高速外设
 - II. 需要与主机批量交换数据的外设
 - III. 具有 DMA 接口的设备
 A. 仅 I B. 仅 III C. I、III D. II、III
07. 中断响应由高到低的优先次序宜用（ ）。
 - A. 访管→程序性→机器故障
 - B. 访管→程序性→重新启动
 - C. 外部→访管→程序性
 - D. 程序性→I/O→访管
08. 在具有中断向量表的计算机中，中断向量地址是（ ）。
 - A. 子程序入口地址
 - B. 中断服务程序的入口地址
 - C. 中断服务程序入口地址的地址
 - D. 中断程序断点
09. 中断响应是在（ ）。
 - A. 一条指令执行开始
 - B. 一条指令执行中间
 - C. 一条指令执行之末
 - D. 一条指令执行的任何时刻
10. 在下列情况下，可能不发生中断请求的是（ ）。
 - A. DMA 操作结束
 - B. 一条指令执行完毕
 - C. 机器出现故障
 - D. 执行“软中断”指令
11. 在配有通道的计算机系统中，用户程序需要输入/输出时，引起的中断是（ ）。
 - A. 访管中断
 - B. I/O 中断
 - C. 程序性中断
 - D. 外中断
12. 某计算机有 4 级中断，优先级从高到低为 1→2→3→4。若将优先级顺序修改，改后 1 级中断的屏蔽字为 1101，2 级中断的屏蔽字为 0100，3 级中断的屏蔽字为 1111，4 级中断的屏蔽字为 0101，则修改后的优先顺序从高到低为（ ）。
 - A. 1→2→3→4
 - B. 3→1→4→2
 - C. 1→3→4→2
 - D. 2→1→3→4
13. 下列不属于程序控制指令的是（ ）。
 - A. 无条件转移指令
 - B. 有条件转移指令
 - C. 中断隐指令
 - D. 循环指令
14. 在中断响应周期中，CPU 主要完成的工作是（ ）。
 - A. 关中断，保护断点，发中断响应信号并形成向量地址
 - B. 开中断，保护断点，发中断响应信号并形成向量地址
 - C. 关中断，执行中断服务程序
 - D. 开中断，执行中断服务程序
15. 在中断响应周期中，由（ ）将允许中断触发器置 0。
 - A. 关中断指令
 - B. 中断隐指令
 - C. 开中断指令
 - D. 中断服务程序
16. CPU 响应中断时最先完成的步骤是（ ）。
 - A. 开中断
 - B. 保存断点
 - C. 关中断
 - D. 转入中断服务程序

17. 设置中断屏蔽标志可以改变()。
A. 多个中断源的中断请求优先级 B. CPU对多个中断请求响应的优先次序
C. 多个中断服务程序开始执行的顺序 D. 多个中断服务程序执行完的次序
18. 在CPU响应中断时,保护两个关键的硬件状态是()。
A. PC和IR B. PC和PSW C. AR和IR D. AR和PSW
19. 在各种I/O方式中,中断方式的特点是(),DMA方式的特点是()。
A. CPU与外设串行工作,传送与主程序串行工作
B. CPU与外设并行工作,传送与主程序串行工作
C. CPU与外设串行工作,传送与主程序并行工作
D. CPU与外设并行工作,传送与主程序并行工作
20. 在DMA传送方式中,由()发出DMA请求,在传送期间总线控制权由()掌握。
A. 外部设备、CPU B. DMA控制器、DMA控制器
C. 外部设备、DMA控制器 D. DMA控制器、内存
21. 下列叙述中,()是正确的。
A. 程序中断方式和DMA方式中实现数据传送都需要中断请求
B. 程序中断方式中有中断请求,DMA方式中没有中断请求
C. 程序中断方式和DMA方式中都有中断请求,但目的不同
D. DMA要等指令周期结束时才可以进行周期窃取
22. 以下关于DMA方式进行I/O的描述中,正确的是()。
A. 一个完整的DMA过程,部分由DMA控制器控制,部分由CPU控制
B. 一个完整的DMA过程,完全由CPU控制
C. 一个完整的DMA过程,完全由DMA控制器控制,CPU不介入任何控制
D. 一个完整的DMA过程,完全由CPU采用周期挪用法控制
23. CPU响应DMA请求的条件是当前()执行完。
A. 机器周期 B. 总线周期
C. 机器周期和总线周期 D. 指令周期
24. 关于外中断(故障除外)和DMA,下列说法中正确的是()。
A. DMA请求和中断请求同时发生时,响应DMA请求
B. DMA请求、非屏蔽中断、可屏蔽中断都要在当前指令结束之后才能被响应
C. 非屏蔽中断请求优先级最高,可屏蔽中断请求优先级最低
D. 若不开中断,所有中断请求就不能响应
25. 以下有关DMA方式的叙述中,错误的是()。
A. 在DMA方式下,DMA控制器向CPU请求的是总线使用权
B. DMA方式可用于键盘和鼠标的数据输入
C. 在数据传输阶段,不需要CPU介入,完全由DMA控制器控制
D. DMA方式要用到中断处理
26. 在主机和外设的信息传送中,()不是一种程序控制方式。
A. 直接程序传送 B. 程序中断
C. 直接存储器存取(DMA) D. 通道控制
27. 中断发生时,程序计数器内容的保护和更新是由()完成的。
A. 硬件自动 B. 进栈指令和转移指令

关注公众号【乘龙考研】
一手更新 稳定有保障

- C. 访存指令 D. 中断服务程序

28. 在 DMA 方式传送数据的过程中, 由于没有破坏()的内容, 所以 CPU 可以正常工作(访存除外)。
A. 程序计数器 B. 程序计数器和寄存器
C. 指令寄存器 D. 堆栈寄存器

29. 在 DMA 方式下, 数据从内存传递到外设经过的路径是()。
A. 内存→数据总线→数据通路→外设 B. 内存→数据总线→DMAC→外设
C. 内存→数据通路→数据总线→外设 D. 内存→CPU→外设

30. 【2009 统考真题】下列选项中, 能引起外部中断的事件是()。
A. 键盘输入 B. 除数为 0 C. 浮点运算下溢 D. 访存缺页

31. 【2010 统考真题】单级中断系统中, 中断服务程序内的执行顺序是()。
I. 保护现场 II. 开中断 III. 关中断 IV. 保存断点 V. 中断事件处理
VI. 恢复现场 VII. 中断返回
A. I→V→VI→II→VII B. III→I→V→VII
C. III→IV→V→VI→VII D. IV→I→V→VI→VII

32. 【2011 统考真题】某计算机有五级中断 $L_4 \sim L_0$, 中断屏蔽字为 $M_4M_3M_2M_1M_0$, $M_i = 1 (0 \leq i \leq 4)$ 表示对 L_i 级中断进行屏蔽。若中断响应优先级从高到低的顺序是 $L_4 \rightarrow L_0 \rightarrow L_2 \rightarrow L_1 \rightarrow L_3$, 则 L_1 的中断处理程序中设置的中断屏蔽字是()。
A. 11110 B. 01101 C. 00011 D. 01010

33. 【2011 统考真题】某计算机处理器主频为 50MHz, 采用定时查询方式控制设备 A 的 I/O, 查询程序运行一次所用的时钟周期数至少为 500。在设备 A 工作期间, 为保证数据不丢失, 每秒需对其查询至少 200 次, 则 CPU 用于设备 A 的 I/O 的时间占整个 CPU 时间的百分比至少是()。
A. 0.02% B. 0.05% C. 0.20% D. 0.50%

34. 【2012 统考真题】响应外部中断的过程中, 中断隐指令完成的操作, 除保护断点外, 还包括()。
I. 关中断
II. 保存通用寄存器的内容
III. 形成中断服务程序入口地址并送 PC
A. 仅 I、II B. 仅 I、III C. 仅 II、III D. I、II、III

35. 【2013 统考真题】下列关于中断 I/O 方式和 DMA 方式的叙述中, 错误的是()。
A. 中断 I/O 方式请求的是 CPU 处理时间, DMA 方式请求的是总线使用权
B. 中断响应发生在一条指令执行结束后, DMA 响应发生在一个总线事务完成后
C. 中断 I/O 方式下数据传送通过软件完成, DMA 方式下数据传送由硬件完成
D. 中断 I/O 方式适用于所有外部设备, DMA 方式仅适用于快速外部设备

36. 【2014 统考真题】若某设备中断请求的响应和处理时间为 100ns, 每 400ns 发出一次中断请求, 中断响应所允许的最长延迟时间为 50ns, 则在该设备持续工作过程中, CPU 用于该设备的 I/O 时间占整个 CPU 时间的百分比至少是()。
A. 12.5% B. 25% C. 37.5% D. 50%

37. 【2015 统考真题】在采用中断 I/O 方式控制打印输出的情况下, CPU 和打印控制接口中的 I/O 端口之间交换的信息不可能是()。

- A. 打印字符 B. 主存地址 C. 设备状态 D. 控制命令
38. 【2017 统考真题】下列关于多重中断系统的叙述中，错误的是（ ）。
- A. 在一条指令执行结束时响应中断
 - B. 中断处理期间 CPU 处于关中断状态
 - C. 中断请求的产生与当前指令的执行无关
 - D. CPU 通过采样中断请求信号检测中断请求
39. 【2018 统考真题】下列关于外部 I/O 中断的叙述中，正确的是（ ）。
- A. 中断控制器按所接收中断请求的先后次序进行中断优先级排队
 - B. CPU 响应中断时，通过执行中断隐指令完成通用寄存器的保护
 - C. CPU 只有在处于中断允许状态时，才能响应外部设备的中断请求
 - D. 有中断请求时，CPU 立即暂停当前指令执行，转去执行中断服务程序
40. 【2019 统考真题】某设备以中断方式与 CPU 进行数据交换，CPU 主频为 1GHz，设备接口中的数据缓冲寄存器为 32 位，设备的数据传输率为 50kB/s。若每次中断开销（包括中断响应和中断处理）为 1000 个时钟周期，则 CPU 用于该设备输入/输出的时间占整个 CPU 时间的百分比最多是（ ）。
- A. 1.25% B. 2.5% C. 5% D. 12.5%
41. 【2019 统考真题】下列关于 DMA 方式的叙述中，正确的是（ ）。
- I. DMA 传送前由设备驱动程序设置传送参数
 - II. 数据传送前由 DMA 控制器请求总线使用权
 - III. 数据传送由 DMA 控制器直接控制总线完成
 - IV. DMA 传送结束后的处理由中断服务程序完成
- A. 仅 I、II B. 仅 I、III、IV C. 仅 II、III、IV D. I、II、III、IV
42. 【2020 统考真题】下列事件中，属于外部中断事件的是（ ）。
- I. 访存时缺页 II. 定时器到时 III. 网络数据包到达
- A. 仅 I、II B. 仅 I、III C. 仅 II、III D. I、II 和 III
43. 【2020 统考真题】外部中断包括不可屏蔽中断（NMI）和可屏蔽中断，下列关于外部中断的叙述中，错误的是（ ）。
- A. CPU 处于关中断状态时，也能响应 NMI 请求
 - B. 一旦可屏蔽中断请求信号有效，CPU 将立即响应
 - C. 不可屏蔽中断的优先级比可屏蔽中断的优先级高
 - D. 可通过中断屏蔽字改变可屏蔽中断的处理优先级
44. 【2020 统考真题】若设备采用周期挪用 DMA 方式进行输入和输出，每次 DMA 传送的数据块大小为 512 字节，相应的 I/O 接口中有一个 32 位数据缓冲寄存器。对于数据输入过程，下列叙述中，错误的是（ ）。
- A. 每准备好 32 位数据，DMA 控制器就发出一次总线请求
 - B. 相对于 CPU，DMA 控制器的总线使用权的优先级更高
 - C. 在整个数据块的传送过程中，CPU 不可以访问主存储器
 - D. 数据块传送结束时，会产生“DMA 传送结束”中断请求
45. 【2021 统考真题】下列是关于多重中断系统中 CPU 响应中断的叙述，错误的是（ ）。
- A. 仅在用户态（执行用户程序）下，CPU 才能检测和响应中断
 - B. CPU 只有在检测到中断请求信号后，才会进入中断响应周期
- 关注公众号【乘龙考研】
一手更新 稳定有保障

- C. 进入中断响应周期时, CPU 一定处于中断允许(开中断)状态
D. 若 CPU 检测到中断请求信号, 则一定存在未被屏蔽的中断源请求信号
46. 【2022 统考真题】下列关于中断 I/O 方式的叙述中, 不正确的是()。
A. 适用于键盘、针式打印机等字符型设备
B. 外设和主机之间的数据传送通过软件完成
C. 外设准备数据的时间应小于中断处理时间
D. 外设为某进程准备数据时 CPU 可运行其他进程

二、综合应用题

01. 在 DMA 方式下, 主存和 I/O 设备之间有一条物理通路相连吗?
02. 回答下列问题:
1) 一个完整的指令周期包括哪些 CPU 工作周期?
2) 中断周期前和中断周期后各是 CPU 的什么工作周期?
3) DMA 周期前和 DMA 周期后各是 CPU 的什么工作周期?
03. 假定某 I/O 设备向 CPU 传送信息的最高频率为 4 万次/秒, 而相应中断处理程序的执行时间为 $40\mu s$, 则该 I/O 设备是否可采用中断方式工作? 为什么?
04. 在程序查询方式的输入/输出系统中, 假设不考虑处理时间, 每个查询操作需要 100 个时钟周期, CPU 的时钟频率为 50MHz。现有鼠标和硬盘两个设备, 而且 CPU 必须每秒对鼠标进行 30 次查询, 硬盘以 32 位字长为单位传输数据, 即每 32 位被 CPU 查询一次, 传输率为 $2 \times 2^{20} B/s$ 。求 CPU 对这两个设备查询所花费的时间比率, 由此可得出什么结论?
05. 设某计算机有 4 个中断源 1、2、3、4, 其硬件排队优先次序按 $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ 降序排列, 各中断源的服务程序中所对应的屏蔽字如下表所示。

中断源	屏蔽字			
	1	2	3	4
1	1	1	0	1
2	0	1	0	0
3	1	1	1	1
4	0	1	0	1

- 1) 给出上述 4 个中断源的中断处理次序。
2) 若 4 个中断源同时有中断请求, 画出 CPU 执行程序的轨迹。
06. 一个 DMA 接口可采用周期窃取方式把字符传送到存储器, 它支持的最大批量为 400B。若存取周期为 $0.2\mu s$, 每处理一次中断需 $5\mu s$, 现有的字符设备的传输率为 $9600B/s$ 。假设字符之间的传输是无间隙的, 试问 DMA 方式每秒因数据传输占用处理器多少时间? 若完全采用中断方式, 又需占处理器多少时间(忽略预处理所需时间)?
07. 假设磁盘传输数据以 32 位的字为单位, 传输速率为 $1MB/s$, CPU 的时钟频率为 $50MHz$ 。
回答以下问题:
1) 采取程序查询方式, 假设查询操作需要 100 个时钟周期, 求 CPU 为 I/O 查询所花费的时间比率(假设进行足够的查询以避免数据丢失)。
2) 采用中断方式进行控制, 每次传输的开销(包括中断处理)为 80 个时钟周期。求 CPU 为传输硬盘所花费的时间比率。
3) 采用 DMA 的方式, 假定 DMA 的启动需要 1000 个时钟周期, DMA 完成时后处理需

要 500 个时钟周期。若平均传输的数据长度为 4KB (此处 K = 1000), 试问硬盘工作时 CPU 将用多少时间比率进行输入/输出操作? 忽略 DMA 申请总线的影响。

08. 【2009 统考真题】某计算机的 CPU 主频为 500MHz, CPI 为 5 (即执行每条指令平均需 5 个时钟周期)。假定某外设的数据传输率为 0.5MB/s, 采用中断方式与主机进行数据传送, 以 32 位为传输单位, 对应的中断服务程序包含 18 条指令, 中断服务的其他开销相当于 2 条指令的执行时间。回答下列问题, 要求给出计算过程。

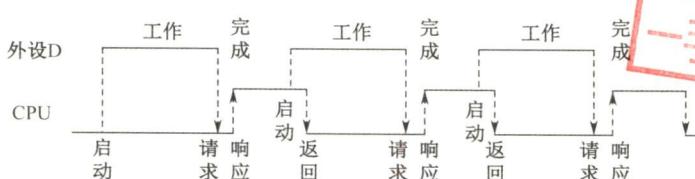
- 1) 在中断方式下, CPU 用于该外设 I/O 的时间占整个 CPU 时间的百分比是多少?
- 2) 当该外设的数据传输率达到 5MB/s 时, 改用 DMA 方式传送数据。假定每次 DMA 传送块大小为 5000B, 且 DMA 预处理和后处理的总开销为 500 个时钟周期, 则 CPU 用于该外设 I/O 的时间占整个 CPU 时间的百分比是多少 (假设 DMA 与 CPU 之间没有访存冲突)?

09. 【2012 统考真题】假定某计算机的 CPU 主频为 80MHz, CPI 为 4, 平均每条指令访存 1.5 次, 主存与 Cache 之间交换的块大小为 16B, Cache 的命中率为 99%, 存储器总线宽带为 32 位。回答下列问题。

- 1) 该计算机的 MIPS 数是多少? 平均每秒 Cache 缺失的次数是多少? 在不考虑 DMA 传送的情况下, 主存带宽至少达到多少才能满足 CPU 的访存要求?
- 2) 假定在 Cache 缺失的情况下访问主存时, 存在 0.0005% 的缺页率, 则 CPU 平均每秒产生多少次缺页异常? 若页面大小为 4KB, 每次缺页都需要访问磁盘, 访问磁盘时 DMA 传送采用周期挪用方式, 磁盘 I/O 接口的数据缓冲寄存器为 32 位, 则磁盘 I/O 接口平均每秒发出的 DMA 请求次数至少是多少?
- 3) CPU 和 DMA 控制器同时要求使用存储器总线时, 哪个优先级更高? 为什么?
- 4) 为了提高性能, 主存采用 4 体低位交叉存储模式, 工作时每 1/4 个存储周期启动一个体。若每个体的存储周期为 50ns, 则该主存能提供的最大带宽是多少?

10. 【2016 统考真题】假定 CPU 主频为 50MHz, CPI 为 4。设备 D 采用异步串行通信方式向主机传送 7 位 ASCII 码字符, 通信规程中有 1 位奇校验位和 1 位停止位, 从 D 接收启动命令到字符送入 I/O 端口需要 0.5ms。回答下列问题, 要求说明理由。

- 1) 每传送一个字符, 在异步串行通信线上共需传输多少位? 在设备 D 持续工作过程中, 每秒最多可向 I/O 端口送入多少个字符?
- 2) 设备 D 采用中断方式进行输入/输出, 示意图如下所示:



I/O 端口每收到一个字符申请一次中断, 中断响应需 10 个时钟周期, 中断服务程序共有 20 条指令, 其中第 15 条指令启动 D 工作。若 CPU 需从 D 读取 1000 个字符, 则完成这一任务所需时间大约是多少个时钟周期? CPU 用于完成这一任务的时间大约是多少个时钟周期? 在中断响应阶段 CPU 进行了哪些操作?

11. 【2018 统考真题】假定计算机的主频为 500MHz, CPI 为 4。现有设备 A 和 B, 其数据传输率分别为 2MB/s 和 40MB/s, 对应 I/O 接口中各有一个 32 位数据缓冲寄存器。回答下列问题, 要求给出计算过程。

关注公众号【乘龙考研】
一手更新稳定有保障

- 1) 若设备 A 采用定时查询 I/O 方式, 每次输入/输出都至少执行 10 条指令。设备 A 最多间隔多长时间查询一次才能不丢失数据? CPU 用于设备 A 输入/输出的时间占 CPU 总时间的百分比至少是多少?
- 2) 在中断 I/O 方式下, 若每次中断响应和中断处理的总时钟周期数至少为 400, 则设备 B 能否采用中断 I/O 方式? 为什么?
- 3) 若设备 B 采用 DMA 方式, 每次 DMA 传送的数据块大小为 1000B, CPU 用于 DMA 预处理和后处理的总时钟周期数为 500, 则 CPU 用于设备 B 输入/输出的时间占 CPU 总时间的百分比最多是多少?
12. 【2022 统考真题】假设某磁盘驱动器中有 4 个双面盘片, 每个盘面有 20000 个磁道, 每个磁道有 500 个扇区, 每个扇区可记录 512 字节的数据, 盘片转速为 7200rpm (转/分), 平均寻道时间为 5ms。请回答下列问题。
- 1) 每个扇区包含数据及其地址信息, 地址信息分为 3 个字段。这 3 个字段的名称各是什么? 对于该磁盘, 各字段至少占多少位?
 - 2) 一个扇区的平均访问时间约为多少?
 - 3) 若采用周期挪用 DMA 方式进行磁盘与主机之间的数据传送, 磁盘控制器中的数据缓冲区大小为 64 位, 则在一个扇区读写过程中, DMA 控制器向 CPU 发送了多少次总线请求? 若 CPU 检测到 DMA 控制器的总线请求信号时也需要访问主存, 则 DMA 控制器是否可以获得总线使用权? 为什么?

7.3.5 答案与解析

一、单项选择题

01. B

当有多个中断请求同时出现时, 中断服务系统必须能从中选出当前最需要给予响应的且最重要的中断请求, 这就需要预先对所有的中断进行优先级排队, 这个工作可由中断判优逻辑来完成, 排队的规则可由软件通过对中断屏蔽寄存器进行设置来确定。

02. C

中断服务程序是处理器处理的紧急事件, 可理解为一种服务, 是通过执行事先编好的某个特定的程序来完成的, 一般属于操作系统的模块, 以供调用执行, 因此选项 A 正确。中断向量由向量地址形成部件, 即由硬件产生, 并且不同的中断源对应不同的中断服务程序, 因此通过该方法, 可以较快速地识别中断源, 因此选项 B 正确。中断向量是中断服务程序的入口地址, 中断向量地址是内存中存放中断向量的地址, 即中断服务程序入口地址的地址, 因此选项 C 错误。重叠处理中断的现象称为中断嵌套, 因此选项 D 正确。

03. D

外部事件如按 Esc 键以退出运行的程序等, 属于外中断, I 正确。Cache 完全是由硬件实现的, 不会涉及中断层面, II 错误。虚拟存储器失效如缺页等, 会发出缺页中断, 属于内中断, III 正确。浮点数运算下溢, 直接当作机器零处理, 而不会引发中断, IV 错误。浮点数上溢, 表示超过了浮点数的表示范围, 属于内中断, V 正确。因此选择选项 D。

04. C

DMA 方式不需要 CPU 干预传送操作, 仅在开始和结尾借用 CPU 一点时间, 其余不占用 CPU 任何资源; 中断方式是程序切换, 每次操作需要保护和恢复现场, 所以 DMA 优先级高于中断请求, 从而可以加快处理效率, I 正确。从 I 的分析可知, 程序中断需要中断现行程序, 因此需保护

关注公众号【乘龙考研】
一手更新 稳定有保障

现场，以便中断执行完后还能回到原来的点去继续没有完成的工作；DMA 方式不需要中断现行程序，CPU 仅仅做一些辅助性工作，因为主存和 DMA 接口之间有一条数据通路，所以无须使用 CPU 内部寄存器，也就无须保护现场，II 正确。III 的说法正好相反。

注意：程序中断的保护现场是由中断服务子程序完成的，不同中断源对应的中断子程序是不同的，可以理解为因 DMA 方式无须使用 CPU 内部寄存器，所以其对应的中断服务子程序也无须保存 CPU 现场。此外，“DMA 方式无须保护现场”是唐溯飞老师教材的原话。

05. B

程序中断过程是由硬件执行的中断隐指令和中断服务程序共同完成的，因此 I 正确。每条指令周期结束后，CPU 会统一扫描各个中断源，然后进行判优来决定响应哪个中断源，而不是在每条指令的执行过程中这样做，因此 II 错误。CPU 会在每个存储周期结束后检查是否有 DMA 请求，而不是在指令执行过程的末尾这样做，因此 III 错误。中断服务程序的最后指令通常是中断返回指令，与无条件转移指令不同的是，它不仅要修改 PC 值，而且要将 CPU 中的所有寄存器都恢复到中断前的状态，因此 IV 错误。

06. B

只有具有 DMA 接口的设备才能产生 DMA 请求，即使当前设备是高速设备或需要与主机批量交换数据，若没有 DMA 接口的话，也不能产生 DMA 请求。

07. B

中断优先级由高至低为访管→程序性→重新启动。重新启动应当等待其他任务完成后再进行，优先级最低，访管指令最紧迫，优先级最高。硬件故障优先级最高，访管指令优先级要高于外部中断。

08. C

中断向量地址是中断向量表的地址，由于中断向量表保存着中断服务程序的入口地址，所以中断向量地址是中断服务程序入口地址的地址。

09. C

CPU 响应中断必须满足下列 3 个条件：①CPU 接收到中断请求信号。首先中断源要发出中断请求，同时 CPU 还要收到这个中断请求信号。②CPU 允许中断，即开中断。③一条指令执行完毕。因此中断响应是在指令执行末尾，选项 C 正确。

10. B

DMA 操作结束、机器出现故障、执行“软中断”指令时都会产生中断请求。一条指令执行完毕可能响应中断请求，但它本身不会引起中断请求。

11. A

用户程序需要输入/输出时，需要调用操作系统提供的接口（请求操作系统服务），此时会引起访管中断，系统由用户态转为核心态。

12. B

屏蔽字“1”表示不可被中断，“0”表示可被中断。由 3 级中断的屏蔽字可知，它屏蔽所有中断，优先级最高；再由 1 级中断的屏蔽字可知，它屏蔽除 3 外的所有中断，优先级次之；以此类推，可知选 B。

【另解】“1”越多表示优先级越高，因此屏蔽其他中断源就越多。

13. C

中断隐指令并不是一条由程序员安排的真正的指令，因此不可能把它预先编入程序中，只能在

响应中断时由硬件直接控制执行。中断隐指令不在指令系统中，因此不属于程序控制指令。

14. A

在中断响应周期，CPU 主要完成关中断、保护断点、发中断响应信号并形成中断向量地址的工作，即执行中断隐指令。

15. B

允许中断触发器置 0 表示关中断，由中断隐指令完成，即由硬件自动完成。

16. C

只有先关中断，才可以保护断点。若先不保护断点，则可能会丢失当前程序的断点。同理，在恢复现场前也要关中断。这个过程和操作系统中的信号量 PV 操作类似，都是将内部过程变为不可打断的原子操作。

17. D

中断屏蔽标志的一种作用是实现中断升级，即改变中断处理的次序（注意分清中断响应次序和中断处理次序，中断响应次序由硬件排队电路决定），因此其可以改变多个中断服务程序执行完的次序。

18. B

PC 的内容是被中断程序尚未执行的第一条指令地址，PSW 寄存器保存各种状态信息。CPU 响应中断后，需要保护中断的 CPU 现场，将 PC 和 PSW 压入堆栈，这样等到中断结束后，就可以将压入堆栈的原 PC 和 PSW 的内容恢复到相应的寄存器，原程序从断点开始继续执行。

19. B、D

在程序查询方式中，CPU 与外设串行工作，传送与主程序串行工作。在中断方式中，CPU 与外设并行工作，当数据准备好时仍需中断主程序以执行数据传送，因此传送与主程序仍是串行的。在 DMA 方式中，CPU 与外设、传送与主程序都是并行的。

20. C

在 DMA 传送方式中，由外部设备向 DMA 控制器发出 DMA 请求信号，然后由 DMA 控制器向 CPU 发出总线请求信号。在 DMA 方式中，DMA 控制器在传送期间有总线控制权，这时 CPU 不能响应 I/O 中断。

21. C

程序中断方式在数据传输时，首先要发出中断请求，此时 CPU 中断正在进行的操作，转而进行数据传输，直到数据传送结束，CPU 才返回中断前执行的操作。DMA 方式只是在 DMA 的前处理和后处理过程中需要用中断的方式请求 CPU 操作，但在数据传送过程中，并不需要中断请求，因此选项 A 错误。DMA 方式和程序中断方式都有中断请求，但目的不同，程序中断方式的中断请求是为了进行数据传送，而 DMA 方式中的中断请求只是为了获得总线控制权或交回总线控制权，因此选项 B 错误、C 正确。CPU 对 DMA 的响应可在指令执行过程中的任何两个存取周期之间，因此选项 D 错误。

22. A

一个完整的 DMA 过程主要由 DMA 控制器控制，但也需要 CPU 参与控制，只是 CPU 干预比较少，只需在数据传输开始和结束时干预，从而提高了 CPU 的效率。

23. A

每个机器周期结束后，CPU 就可以响应 DMA 请求。注意区别：DMA 在与主存交互数据时通过周期窃取方式，窃取的是存取周期。

24. A

DMA 连接的是高速设备，其优先级高于中断请求，以防止高速设备数据丢失，选项 A 正确。DMA 请求的响应时间可以发生在每个机器周期结束时，只要 CPU 不占用总线；中断请求的响应时间只能发生在每条指令执行完毕，选项 B 错误。DMA 的优先级要比外中断（非屏蔽中断、可屏蔽中断）高，选项 C 错误。如果不开中断，内中断和非屏蔽中断仍可响应，选项 D 错误。

25. B

DMA 方式只能用于数据传输，它不具有对异常事件的处理能力，不能中断现行程序，而键盘和鼠标均要求 CPU 立即响应，因此无法采用 DMA 方式。

26. C

只有 DMA 方式是靠硬件电路实现的，三种基本的程序控制方式即直接程序传送、程序中断、通道控制都需要程序的干预。

27. A

中断发生时，程序计数器内容的保护和更新是由硬件自动完成的，即由中断隐指令完成。

28. B

DMA 方式传送数据时，挪用周期不会改变 CPU 现场，因此无须占用 CPU 的程序计数器和寄存器。

29. B

DMA 方式的数据传送不经过 CPU，但需要经过 DMA 控制器中的数据缓冲寄存器。输入时，数据由外设（如磁盘）先送往 DMA 的数据缓冲寄存器，再通过数据总线送到主存。反之，输出时，数据由主存通过数据总线送到 DMA 的数据缓冲寄存器，然后送到外设。

30. A

外部中断是指 CPU 执行指令以外的事件产生的中断，通常指来自 CPU 与内存以外的中断。选项 A 中键盘输入属于外部事件，每次键盘输入 CPU 都需要执行中断以读入输入数据，所以能引起外部中断。选项 B 中除数为 0 属于异常，也就是内中断，发生在 CPU 内部。选项 C 中浮点运算下溢将按机器零处理，不会产生中断。而选项 D 中访存缺页属于 CPU 执行指令时产生的中断，也不属于外部中断。所以能产生外部中断的只能是输入设备键盘。

31. A

在单级（或单重）中断系统中，不允许中断嵌套。中断处理过程为：①关中断；②保存断点；③识别中断源；④保存现场；⑤中断事件处理；⑥恢复现场；⑦开中断；⑧中断返回。其中①～③由硬件完成，④～⑧由中断服务程序完成，因此选择选项 A。

32. D

高优先级置 0 表示可被中断，比该中断优先级低（相等）的置 1 表示不可被中断。从中断响应优先级看， L_1 只能屏蔽 L_3 和其自身，中断屏蔽字 $M_4M_3M_2M_1M_0 = 01010$ ，因此选择选项 D。

33. C

每秒进行 200 次查询，每次 500 个时钟周期，则每秒最少占用 $200 \times 500 = 100000$ 个时钟周期，因此占 CPU 时间的百分比为 $100000 / 50M = 0.20\%$ 。

34. B

在响应外部中断的过程中，中断隐指令完成的操作包括：①关中断；②保存断点；③引出中断服务程序（形成中断服务程序入口地址并送 PC），所以只有 I、III 正确。II 中保存通用寄存器的内容是在进入中断服务程序后首先进行的操作。

35. D

中断处理方式：在 I/O 设备输入每个数据的过程中，由于无须 CPU 干预，因而可使 CPU 与

I/O 设备并行工作。仅当输完一个数据时，才需 CPU 花费极短的时间去做一些中断处理。因此中断申请使用的是 CPU 处理时间，发生的时间是在一条指令执行结束之后，数据在软件的控制下完成传送。而 DMA 方式与之不同。DMA 方式：数据传输的基本单位是数据块，即在 CPU 与 I/O 设备之间，每次传送至少一个数据块；DMA 方式每次申请的是总线的使用权，所传送的数据是从设备直接送入内存的，或者相反；仅在传送一个或多个数据块的开始和结束时，才需要 CPU 干预，整块数据的传送是在控制器的控制下完成的。

36. B

每 400ns 发出一次中断请求，而响应和处理时间为 100ns，其中容许的延迟为干扰信息，因为在 50ns 内，无论怎么延迟，每 400ns 仍要花费 100ns 处理中断，所以该设备的 I/O 时间占整个 CPU 时间的百分比为 $100\text{ns}/400\text{ns} = 25\%$ 。

37. B

在程序中断 I/O 方式中，CPU 和打印机直接交换，打印字符直接传输到打印机的 I/O 端口，不会涉及主存地址。而 CPU 和打印机通过 I/O 端口中的状态口和控制口来实现交互。

38. B

多重中断在保护被中断进程现场时关中断，执行中断处理程序时开中断，选项 B 错误。CPU 一般在一条指令执行结束的阶段检测中断请求信号，查看是否存在中断请求，然后决定是否响应中断，选项 A、D 正确。中断是指来自 CPU 执行指令以外的事件，选项 C 正确。

39. C

中断优先级由屏蔽字而非请求的先后次序决定，A 错误。中断隐指令完成的工作有：①关中断；②保存断点；③引出中断服务程序，通用寄存器的保护由中断服务程序完成，选项 B 错误。中断允许状态即开中断后，才能响应中断请求，选项 C 正确。有中断请求时，如果是关中断的状态，或新中断请求的优先级较低，则不能响应新的中断请求，选项 D 错误。

40. A

设备接口中的数据缓冲寄存器为 32 位，即一次中断可以传输 4B 数据，设备数据传输率为 50kB/s，共需要 12.5k 次中断，每次中断开销为 1000 个时钟周期，CPU 主频为 1GHz，则 CPU 用于该设备输入/输出的时间占整个 CPU 时间的百分比最多是 $(12.5\text{k} \times 1000)/1\text{G} = 1.25\%$ 。

41. D

每类设备都配置一个设备驱动程序，设备驱动程序向上层用户程序提供一组标准接口，负责实现对设备发出各种具体操作指令，用户程序不能直接和 DMA 打交道。DMA 的数据传送过程分为预处理、数据传送和后处理 3 个阶段。预处理阶段由 CPU 完成必要的准备工作，数据传送前由 DMA 控制器请求总线使用权；数据传送由 DMA 控制器直接控制总线完成；传送结束后，DMA 控制器向 CPU 发送中断请求，CPU 执行中断服务程序做 DMA 结束处理。

42. C

访存时缺页属于内部异常，I 错误；定时器到时描述的是时钟中断，属于外部中断，II 正确；网络数据包到达描述的是 CPU 执行指令以外的事件，属于外部中断，III 正确。

43. B

由 CPU 内部产生的异常称为内中断，内中断是不可屏蔽中断。通过中断请求线 INTR 和 NMI，从 CPU 外部发出的中断请求称为外中断，通过 INTR 信号线发出的外中断是可屏蔽中断，而通过 NMI 信号线发出的是不可屏蔽中断。不可屏蔽中断即使在关中断（IF = 0）情况下也会被响应，A 正确。不可屏蔽中断的优先级最高，任何时候只要发生不可屏蔽中断，都要中止现行程序的执行，转到不可

屏蔽中断处理程序执行，C 正确。CPU 响应中断需要满足 3 个条件：①中断源有中断请求；②CPU 允许中断及开中断；③一条指令执行完毕，且没有更紧迫的任务。故选项 B 错误。

44. C

周期挪用法由 DMA 控制器挪用一个或几个主存周期来访问主存，传送完一个数据字后立即释放总线，是一种单字传送方式，每个字传送完后 CPU 可以访问主存，选项 C 错误。停止 CPU 访存法则是指在整个数据块的传送过程中，使 CPU 脱离总线，停止访问主存。

45. A

中断服务程序在内核态下执行，若只能在用户态下检测和响应中断，则显然无法实现多重中断（中断嵌套），A 错误。在多重中断中，CPU 只有在检测到中断请求信号后（中断处理优先级更低的中断请求信号是检测不到的），才会进入中断响应周期。进入中断响应周期时，说明此时 CPU 一定处于中断允许状态，否则无法响应该中断。如果所有中断源都被屏蔽（说明该中断的处理优先级最高），则 CPU 不会检测到任何中断请求信号。

46. C

中断 I/O 方式适用于字符型设备，此类设备的特点是数据传输速率慢，以字符或字为单位进行传输，选项 A 正确。若采用中断 I/O 方式，当外设准备好数据后，向 CPU 发出中断请求，CPU 暂时中止现行程序，转去运行中断服务程序，由中断服务程序完成数据传送，选项 B 正确。若外设准备数据的时间小于中断处理时间，则可能导致数据丢失，以输入设备为例，设备为进程准备的数据会先写入设备控制器的缓冲区（缓冲区大小有限），缓冲区每写满一次，就会向 CPU 发出一次中断请求，CPU 响应并处理中断的过程，就是将缓冲区中的数据“取走”的过程，因此若外设准备数据的时间小于中断处理时间，则可能导致外设往缓冲区写入数据的速度快于 CPU 从缓冲区取走数据的速度，从而导致缓冲区的数据被覆盖，进而导致数据丢失。选项 C 错误。若采用中断 I/O 方式，则外设为某进程准备数据时，可令该进程阻塞，CPU 运行其他进程，选项 D 正确。

二、综合应用题

01. 【解答】

没有。通常所说的 DMA 方式在主存和 I/O 设备之间建立一条“直接的数据通路”，使得数据在主存和 I/O 设备之间直接进行传送，其含义并不是在主存和 I/O 之间建立一条物理直接通路，而是主存和 I/O 设备通过 I/O 设备接口、系统总线及总线桥接部件等相连，建立一个信息可以相互通达的通路，这在逻辑上可视为直接相连的。其“直接”是相对于要通过 CPU 才能和主存相连这种方式而言的。

02. 【解答】

- 1) 一个完整的指令周期包括取指周期、间址周期、执行周期和中断周期。其中取指周期和执行周期是每条指令均有的。
- 2) 中断周期前是执行周期，中断周期后是下一条指令的取指周期。
- 3) DMA 周期前可以是取指周期、间址周期、执行周期或中断周期，DMA 周期后也可以是取指周期、间址周期、执行周期或中断周期。总之，DMA 周期前后都是机器周期。

03. 【解答】

I/O 设备传送一个数据的时间为 $1/(4 \times 10^4)s = 25\mu s$ ，所以请求中断的周期为 $25\mu s$ ，而相应中断处理程序的执行时间为 $40\mu s$ ，大于请求中断的周期，会丢失数据（单位时间内 I/O 请求数量比中断处理的多，自然会丢失数据），所以不能采用中断方式。

04. 【解答】

- 1) CPU 每秒对鼠标进行 30 次查询，所需的时钟周期数为 $100 \times 30 = 3000$ 。CPU 的时钟频率为 50MHz，即每秒 50×10^6 个时钟周期，因此对鼠标的查询占用 CPU 的时间比率为
- $$[3000/(50 \times 10^6)] \times 100\% = 0.006\%$$

可见，对鼠标的查询基本不影响 CPU 的性能。

- 2) 对于硬盘，每 32 位 (4B) 被 CPU 查询一次，因此每秒查询次数为 $2 \times 2^{20} B / 4B = 512K$ ；则每秒查询的时钟周期数为

$$100 \times 512 \times 1024 = 52.4 \times 10^6$$

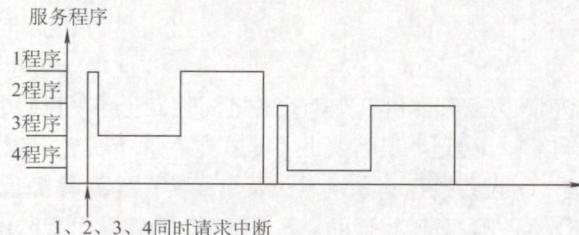
因此对硬盘的查询占用 CPU 的时间比率为

$$[52.4 \times 10^6 / (50 \times 10^6)] \times 100\% = 105\%$$

可见，即使 CPU 将全部时间都用于对硬盘的查询，也不能满足磁盘传输的要求，因此 CPU 一般不采用程序查询方式与磁盘交换信息。

05. 【解答】

- 1) 中断屏蔽字 “1” 表示不可被中断，“0” 表示可被中断。根据表中 “1”的个数的降序排列可知，4 个中断源的处理次序是 3→1→4→2。
- 2) 当 4 个中断源同时有中断请求时，由于硬件排队的优先次序是 1→2→3→4，因此 CPU 先响应 1 的请求，执行 1 的服务程序。该程序中设置了屏蔽字 1101，因此开中断指令后转去执行 3 服务程序，且 3 服务程序执行结束后又回到了 1 服务程序。1 服务程序结束后，CPU 还有 2、4 两个中断源请求未响应。由于 2 的响应优先级高于 4，因此 CPU 先响应 2 的请求，执行 2 服务程序。在 2 服务程序中由于设置了屏蔽字 0100，意味着 1、3、4 可中断 2 服务程序。而 1、3 的请求已经结束，因此在开中断指令后转去执行 4 服务程序，4 服务程序执行结束后又回到 2 服务程序的断点处，继续执行 2 服务程序，直至该程序执行结束。CPU 执行程序的轨迹如下图所示。



06. 【解答】

根据字符设备的传输率为 9600b/s，得每秒能传输

$$9600/8 = 1200B，即 1200 个字符（本题中字符、字节不加以区分）$$

- 1) 若采用 DMA 方式，传输 1200 个字符共需 1200 个存取周期，考虑到每传 400 个字符需中断处理一次，因此 DMA 方式每秒因数据传输占用处理器的时间是

$$0.2\mu s \times 1200 + 5\mu s \times (1200/400) = 255\mu s$$

- 2) 若采用中断方式，每秒因数据传输占用处理器的时间是

$$5\mu s \times 1200 = 6000\mu s$$

07. 【解答】

- 1) 采用程序查询方式，硬盘传输速率为 1MB/s，一个字为 32bit = 4B，每秒查询的次数为 $1MB/4B = 2.5 \times 10^5$ ，每秒查询所需的总时钟周期数为 $2.5 \times 10^5 \times 100 = 2.5 \times 10^7$ 。CPU 的时钟频率为 50MHz。

因此, I/O 查询所花费的时间比率为 $2.5 \times 10^7 / 50M = 2.5 \times 10^7 / (5 \times 10^7) = 50\%$ 。

- 2) 采用中断方式时, 每传输一个字便进行一次中断处理。

每传输一个字的时间为 $32bit / (1MB/s) = 4\mu s$ 。

CPU 的时钟周期为 $1s / (50MHz) = 0.02s / (1M) = 0.02\mu s$ 。

因此花费的时间比率为 $(80 \times 0.02\mu s) / 4\mu s = 40\%$ 。

- 3) 采用 DMA 方式时, CPU 所花时间仅为启动时间和后处理时间。

每传输一次数据 CPU 所花的时间为 $1000 + 500 = 1500$ 个时钟周期。

DMA 平均传送长度为 $4000B$, 每秒产生的 DMA 次数为 $(1MB/s) / (4 \times 10^3 B) = 250$ 。

故 CPU 为 DMA 所花费时间的比率为 $(1500 \times 250) / 50M = 0.75\%$ 。
【乘龙考研】
更新稳定有保障

08. 【解答】

- 1) 按题意, 外设每秒传送 $0.5MB$, 中断时每次传送 $32bit = 4B$ 。由于 CPI 为 5, 在中断方式下, CPU 每次用于数据传送的时钟周期为 $5 \times 18 + 5 \times 2 = 100$ (中断服务程序 + 其他开销)。为达到外设 $0.5MB/s$ 的数据传输率, 外设每秒申请的中断次数为 $0.5MB / 4B = 125000$ 。

1 秒内用于中断的开销为 $100 \times 125000 = 12500000 = 12.5M$ 个时钟周期。

CPU 用于外设 I/O 的时间占整个 CPU 时间的百分比为 $12.5M / 500M = 2.5\%$ 。

- 2) 当外设数据传输率提高到 $5MB/s$ 时改用 DMA 方式传送, 每次 DMA 传送一个数据块,

大小为 $5000B$, 则 1 秒内需产生的 DMA 次数为 $5MB / 5000B = 1000$ 。

CPU 用于 DMA 处理的总开销为 $1000 \times 500 = 500000 = 0.5M$ 个时钟周期。

CPU 用于外设 I/O 的时间占整个 CPU 时间的百分比为 $0.5M / 500M = 0.1\%$ 。

09. 【解答】

本题涉及多个考点: 计算机的性能指标、存储器的性能指标、DMA 的性能分析、DMA 方式的特点及多体交叉存储器的性能分析。

- 1) 平均每秒 CPU 执行的指令数为 $80M / 4 = 20M$, 因此 MIPS 数为 20。平均每条指令访存 1.5 次, 因此平均每秒 Cache 缺失的次数 = $20M \times 1.5 \times (1 - 99\%) = 300K$ 。当 Cache 缺失时, CPU 访问主存, 主存与 Cache 之间以块为传送单位, 此时主存带宽为 $16B \times 300k/s = 4.8MB/s$ 。在不考虑 DMA 传送的情况下, 主存带宽至少达到 $4.8MB/s$ 才能满足 CPU 的访存要求。
- 2) 题中假定在 Cache 缺失的情况下访问主存, 平均每秒产生缺页中断 $300000 \times 0.0005\% = 1.5$ 次。因为存储器总线宽度为 32 位, 所以每传送 32 位数据, 磁盘控制器发出一次 DMA 请求, 因此平均每秒磁盘 DMA 请求的次数至少为 $1.5 \times 4KB / 4B = 1.5K = 1536$ 。
- 3) CPU 和 DMA 控制器同时要求使用存储器总线时, DMA 请求的优先级更高。因为 DMA 请求得不到及时响应, I/O 传输数据可能会丢失。
- 4) 4 体交叉存储模式能提供的最大带宽为 $4 \times 4B / 50ns = 320MB/s$ 。

10. 【解答】

- 1) 每传送一个 ASCII 码字符, 需要传输的位数有 1 位起始位、7 位数据位 (ASCII 码字符占 7 位)、1 位奇校验位和 1 位停止位, 因此总位数为 $1 + 7 + 1 + 1 = 10$ 。
- I/O 端口每秒最多可接收 $1000 / 0.5 = 2000$ 个字符。
- 2) 一个字符传送时间包括: 设备 D 将字符送 I/O 端口的时间、中断响应时间和中断服务程序前 15 条指令的执行时间。时钟周期为 $1 / (50MHz) = 20ns$, 设备 D 将字符送 I/O 端口的时间为 $0.5ms / 20ns = 2.5 \times 10^4$ 个时钟周期。一个字符的传送时间约为 $2.5 \times 10^4 + 10 + 15 \times 4 =$

25070 个时钟周期。完成 1000 个字符传送所需的时间约为 $1000 \times 25070 = 25070000$ 个时钟周期。

CPU 用于该任务的时间约为 $1000 \times (10 + 20 \times 4) = 9 \times 10^4$ 个时钟周期。

在中断响应阶段，CPU 主要进行以下操作：关中断、保护断点和程序状态、识别中断源。

11. 【解答】

- 1) 程序定时向缓存端口查询数据，由于缓存端口大小有限，必须在传输完端口大小的数据时访问端口，以防止部分数据未被及时读取而丢失。设备 A 准备 32 位数据所用的时间为 $4B/2MB = 2\mu s$ ，所以最多每隔 $2\mu s$ 必须查询一次，每秒的查询次数至少是 $1s/2\mu s = 5 \times 10^5$ ，每秒 CPU 用于设备 A 输入/输出的时间至少为 $5 \times 10^5 \times 10 \times 4 = 2 \times 10^7$ 个时钟周期，占整个 CPU 时间的百分比至少是 $2 \times 10^7 / 500M = 4\%$ 。
- 2) 中断响应和中断处理的时间为 $400 \times (1/500M) = 0.8\mu s$ ，这时只需判断设备 B 准备 32 位数据要多久，若准备数据的时间小于中断响应和中断处理的时间，则数据会被刷新，造成丢失。经过计算，设备 B 准备 32 位数据所用的时间为 $4B/40MB = 0.1\mu s$ ，因此设备 B 不适合采用中断 I/O 方式。
- 3) 在 DMA 方式中，只有预处理和后处理需要 CPU 处理。设备 B 每秒的 DMA 次数最多为 $(40MB/s)/1000B = 40000$ ，CPU 用于设备 B 输入/输出的时间最多为 $40000 \times 500 = 2 \times 10^7$ 个时钟周期，占 CPU 总时间的百分比最多为 $2 \times 10^7 / 500M = 4\%$ 。

12. 【解析】

- 1) 3 个字段的名称为柱面号（或磁道号）、磁头号（或盘面号）、扇区号。由于每个盘面有 20000 个磁道，因此该磁盘共有 20000 个柱面，柱面号字段至少占 $\lceil \log_2 20000 \rceil = 15$ 位；由于该磁盘共有 4 个盘片，每个盘片有 2 个盘面，因此磁头号字段至少占 $\log_2(4 \times 2) = 3$ 位；由于每个磁道有 500 个扇区，因此扇区号字段至少占 $\lceil \log_2 500 \rceil = 9$ 位。
- 2) 一个扇区的访问时间由寻道时间、延迟时间、传输时间三部分组成。平均寻道时间为 5ms，平均延迟时间等于磁盘转半圈所需要的时间，平均传输时间等于一个扇区划过磁头下方所需要的时间。而该磁盘转一圈的时间为 $60 \times 10^3 / 7200 \approx 8.33$ ms，因此一个扇区的平均访问时间约为 $5 + 8.33/2 + 8.33/500 \approx 9.18$ ms。
- 3) 磁盘控制器中的数据缓冲区每充满一次，DMA 控制器就需要发出一次总线请求，将这 64bit 数据通过总线传送到主存，因此，在一个扇区读写过程中，DMA 控制器向 CPU 发送了 $512B/64bit = 64$ 次总线请求。由于采用周期挪用 DMA 方式，因此当 CPU 和 DMA 控制器都需要访问主存时，DMA 控制器可以优先获得总线使用权。因为一旦磁盘开始读写，就必须按时完成数据传送，否则数据缓冲区中的数据会发生丢失。

7.4 本章小结

本章开头提出的问题的参考答案如下。

关注公众号【乘龙考研】
一手更新 稳定有保障

1) I/O 设备有哪些编址方式？各有何特点？

统一编址和独立编址。统一编址是在主存地址中划出一定的范围作为 I/O 地址，以便通过访存指令即可实现对 I/O 的访问，但主存的容量相应减少。独立编址是指 I/O 地址和主存是分开的，I/O 地址不占主存空间，但访存需专门的 I/O 指令。

2) CPU 响应中断应具备哪些条件?

- ① 在 CPU 内部设置的中断屏蔽触发器必须是开放的。
 - ② 外设有中断请求时, 中断请求触发器必须处于“1”状态, 保持中断请求信号。
 - ③ 外设(接口)中断允许触发器必须为“1”, 这样才能把外设中断请求送至 CPU。
- 具备上述三个条件时, CPU 在现行指令结束的最后一个状态周期响应中断。

7.5 常见问题和易混淆知识点

关注公众号【乘龙考研】
一手更新 稳定有保障

1. 中断响应优先级和中断处理优先级分别指什么?

中断响应优先级是由硬件排队线路或中断查询程序的查询顺序决定的, 不可动态改变; 而中断处理优先级可以由中断屏蔽字来改变, 反映的是正在处理的中断是否比新发生的中断的处理优先级低(屏蔽位为“0”, 对新中断开放), 若是, 则中止正在处理的中断, 转到新中断去处理, 处理完后再回到刚才被中止的中断继续处理。

2. 向量中断、中断向量、向量地址三个概念是什么关系?

中断向量: 每个中断源都有对应的处理程序, 这个处理程序称为中断服务程序, 其入口地址称为中断向量。所有中断的中断服务程序入口地址构成一个表, 称为中断向量表; 也有的机器把中断服务程序入口的跳转指令构成一张表, 称为中断向量跳转表。

向量地址: 中断向量表或中断向量跳转表中每个表项所在的内存地址或表项的索引值, 称为向量地址或中断类型号。

向量中断: 指一种识别中断源的技术或方式。识别中断源的目的是找到中断源对应的中断服务程序的入口地址的地址, 即获得向量地址。

3. 程序中断和调用子程序有何区别?

两者的根本区别主要表现在服务时间和服务对象上不一样。

- 1) 调用子程序过程发生的时间是已知的和固定的, 即在主程序中的调用指令(CALL)执行时发生主程序调用子程序过程, 调用指令所在位置是已知的和固定的。而中断过程发生的时间一般是随机的, CPU 在执行某个主程序时收到中断源提出的中断申请, 就发生中断过程, 而中断申请一般由硬件电路产生, 申请提出时间是随机的。也可以说, 调用子程序是程序设计者事先安排的, 而执行中断服务程序是由系统工作环境随机决定的。
- 2) 子程序完全为主程序服务, 两者属于主从关系。主程序需要子程序时就去调用子程序, 并把调用结果带回主程序继续执行。而中断服务程序与主程序二者一般是无关的, 不存在谁为谁服务的问题, 两者是平行关系。
- 3) 主程序调用子程序的过程完全属于软件处理过程, 不需要专门的硬件电路; 而中断处理系统是一个软/硬件结合的系统, 需要专门的硬件电路才能完成中断处理的过程。
- 4) 子程序嵌套可实现若干级, 嵌套的最多级数受计算机内存开辟的堆栈大小限制; 而中断嵌套级数主要由中断优先级来决定, 一般优先级数不会很大。

从宏观上看, 虽然程序中断方式克服了程序查询方式中的 CPU “踏步”现象, 实现了 CPU 与 I/O 并行工作, 提高了 CPU 的资源利用率, 但从微观操作分析, CPU 在处理中断服务程序时, 仍需暂停原程序的正常运行, 尤其是当高速 I/O 设备或辅助存储器频繁地、成批地与主存交换信息时, 需要不断打断 CPU 执行现行程序, 而执行中断服务程序。

参 考 文 献

- [1] 袁春风. 计算机系统基础[M]. 北京: 机械工业出版社, 2018.
- [2] 袁春风. 计算机组装与系统结构[M]. 北京: 清华大学出版社, 2015.
- [3] 袁春风. 计算机系统基础: 习题解答与教学指导. 北京: 机械工业出版社, 2019.
- [4] 唐朔飞. 计算机组装原理[M]. 北京: 高等教育出版社, 2008.
- [5] 唐朔飞. 计算机组装原理: 学习指导与习题解答. 北京: 高等教育出版社, 2012.
- [6] 兰德尔 E. 布莱恩德等. 深入理解计算机系统[M]. 北京: 机械工业出版社, 2016.
- [7] 李春葆等. 计算机组装原理联考辅导教程[M]. 北京: 清华大学出版社, 2010.
- [8] 本书编写组. 计算机专业基础综合考试大纲解析[M]. 北京: 高等教育出版社, 2009.
- [9] 徐爱萍. 计算机组装原理考研指导[M]. 北京: 清华大学出版社, 2003.
- [10] 谭志虎. 计算机组装原理: 微课版. 北京: 人民邮电出版社, 2021.

王道论坛&网易慕课考研

经久才是王道

十五年考研磨练

2024年王道计算机考研课程

- ◆ 阶段一：计算机考研零基础入门
- ◆ 阶段二：大纲考点、经典习题精讲
- ◆ 阶段三：暑期强化提升训练
- ◆ 阶段四：真题、模拟题、考前冲刺
- ◆ 阶段五：复试机试课程



抄底价/299元起

加客服领券购课

- 2024年数据结构考研复习指导
- 2024年计算机组成原理考研复习指导
- 2024年操作系统考研复习指导
- 2024年计算机网络考研复习指导
- 2024年计算机专业基础综合考试历年真题解析
- 2024年计算机专业基础综合考试冲刺模拟题
- 计算机考研——机试指南（第2版）



责任编辑：谭海平
封面设计：张昱

ISBN 978-7-121-44474-6

9 787121 444746 >

定价：69.00 元