第 5 章 存储系统和结构

5.1 基本内容摘要

- 存储系统的组成
  - ◆ 存储器分类
  - ◆ 存储系统层次结构

    Cache-主存存储层次；

    主存-辅存存储层次。
- 主存的组织
  - ◆ 主存的基本结构

    主存通常由存储体、地址译码驱动电路、I/O 电路和读写电路组成。
  - ◆ 主存的存储单元

    大端方案；

    小端方案。
  - ◆ 主存的主要技术指标

    存储容量；

    存取速度。
  - ◆ 数据在主存中的存放

    不浪费主存资源的存放方法；

    从一个存储字的起始位置开始的存放方法；

    边界对齐的存放方法。
- 半导体随机存取存储器和只读存储器
  - ◆ RAM 记忆单元电路
  - ◆ 动态 RAM 的刷新

    集中刷新方式；

    分散刷新方式；

    异步刷新方式。
  - ◆ RAM 芯片分析
  - ◆ 半导体只读存储器(ROM)

    掩膜式 ROM(MROM)；

    一次可编程 ROM(PROM)；

可擦除可编程 ROM(EPROM)；
闪速存储器。

- 主存的连接与控制
  - 主存容量的扩展
    位扩展法、字扩展法、字和位同时扩展法。
  - 存储芯片的地址分配和片选
  - 主存和 CPU 的连接
  - PC 系列微机的存储器接口
- 提高主存读写速度的技术
  - 主存与 CPU 速度的匹配
- 多体交叉存储技术
  - 并行访问存储器
  - 交叉访问存储器
- 高速缓冲存储器(Cache)
  - Cache 的工作原理
  - Cache 的读写操作
  - 地址映像
  - 替换算法
  - 更新策略
- 虚拟存储器
  - 虚拟存储器的基本概念
  - 页式虚拟存储器
  - 快表与慢表

5.2 重点难点梳理

1. 主存储器的编址方式

早期计算机的主存储器(简称主存)均按字编址,CPU 在访问主存时给出一个地址码,就能从主存中读出一个字长的信息或者将一个字长的信息写入主存中。这样的存储器称为按字编址的存储器。

随着计算机技术的飞速发展,存储器的字长不断增长。例如,存储器字长为 32 位,如果仍然采用按字编址方式,则每访问一次存储器只能读出和写入 32 位信息。32 位为 4 字节,如果能将访问存储器的地址指向一个字中的每个字节,就能大大增强访问存储器的灵活性,使得访问一次存储器可读出或写入任何一个字节的信息,也可读出或写入整个字的信息,这就要求主存按字节编址。

某系统中主存容量为 16K×32。如果采用按字编址方式,地址码长度应为 14 位,如果采用按字节编址,则地址码的长度应为 16 位。两种编址方式的区别,如图 5-1 所示。

图 5-1(a)和图 5-1(b) 所示的存储器容量相同,都是 16K×32,区别仅在于图 5-1(a)是按字编址,每个地址指向一个存储字;而图 5-1(b)是按字节编址,每个地址指向一个字节。显然,由于一个存储字中包含 4 字节,所以,按字节编址时,其地址码的长度要比按字编址的

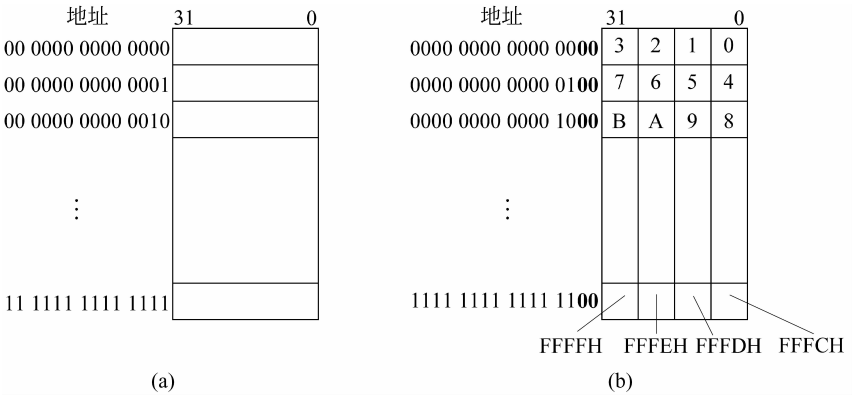

图 5-1　按字编址与按字节编址的区别

地址码长 2 位，这 2 位为 00、01、10、11，分别指向一个存储字中的 4 个字节。图 5-1(a)中的字地址码是连续的，而图 5-1(b)中的字地址码是不连续的，这是由于它所指向的是该字中最低端的那个字节。

采用按字节编址的存储器可根据给定的字节地址访问存储器中的任何一个字节，也可以根据给定的字地址访问一个存储字。

2. 操作数的存储方式

一个多字节的数据在按字节编址的主存中通常由两种排序方案——大端次序和小端次序。大端次序方案将最高有效字节存储在最小地址位置，小端次序方案将最低有效字节存储在最小地址位置。图 5-2 是 32 位的十六进制数 12345678 在存储器中的存储方式示意图。

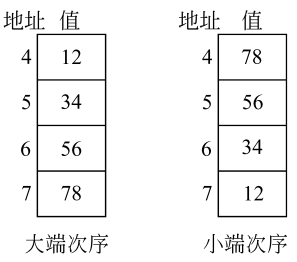

图 5-2　多字节数据的两种存储方式

Intel 80x86 是采用小端次序方案的计算机，IBM 370、Motorola 680x0 和大多数 RISC 计算机则采用大端次序方案。Power PC 是一个既支持大端次序方案又支持小端次序方案的计算机。

3. 存取时间和存取周期

存取时间(T_a)又称为访问时间或读写时间，是指从启动一次存储器操作到完成该操作所经历的时间。例如，读出时间是指从 CPU 向存储器发出有效地址和读命令开始，直到将被选单元的内容读出为止所用的时间；写入时间是指从 CPU 向存储器发出有效地址和写命令开始，直到信息写入被选中单元为止所用的时间。显然，T_a 越小，存取速度越快。

存取周期(T_m)又可称作读写周期、访存周期，是指存储器进行一次完整的读写操作所需的全部时间，即连续两次访问存储器操作的开始时间之间的最短时间。显然，一般情况下，$T_m > T_a$。这是因为对于任何一种存储器，在读写操作之后，总要有一段恢复内部状态的复原时间。对于破坏性读出的存储器，存取周期往往比存取时间要大得多，甚至可以达到 $T_m = 2T_a$，这是因为存储器中的信息读出后需要马上进行重写(再生)。

与存取周期密切相关的指标是主存带宽，它又称为数据传输率，表示主存每秒进出信息的最大数量，单位为字节每秒或位每秒。

4. 边界对齐的数据存放方式

在数据对齐存储方式下,要求一个数据字占据一个完整的存储字的位置,而不是分成两部分,各占据两个存储字位置的一部分。例如,一个32位的数据字放在32位宽度的主存储器中,若字地址为n,则在对齐方式下数据实际占据的是字节地址为n、$n+1$、$n+2$和$n+3$的存储单元,这个数据可以一次读取或者写入。如果这个数据字不按对齐方式存储,假设数据实际占据的是字节地址为$n-1$、n、$n+1$和$n+2$存储单元,这样的数据在32位的主存中需要分两次读取或者写入。在有些计算机中,规定数据的存储必须按边界对齐的方式进行。图5-3(a)是一个边界不对齐的例子,图5-3(b)是一个边界对齐的例子。

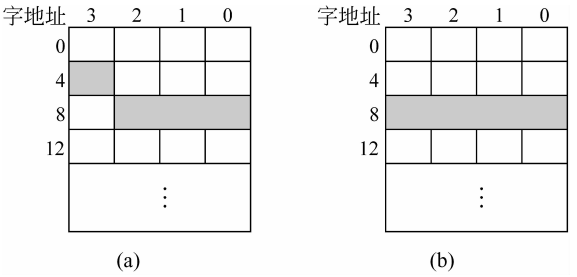

图5-3 32位的数据字在32位宽度主存中的存放

假设,某机存储字长为64位(8字节),读写的数据有4种不同长度,它们分别是字节(8位)、半字(16位)、单字(32位)和双字(64位)。

边界对齐的数据存放方式对数据的存放位置有下列要求:

- 字节数据的地址为×…×××(任意);
- 半字数据的起始地址为×…××0(2的整数倍);
- 单字数据的起始地址为×…×× 00(4的整数倍);
- 双字数据的起始地址为×…× 0 00(8的整数倍)。

5. 动态RAM的刷新

为了维持动态RAM记忆单元的存储信息,通常每隔2ms就必须对存储体中所有记忆单元的栅极电容补充一次电荷,即使许多记忆单元长期未被访问也是如此,这个过程就是刷新。

刷新和重写(再生)是两个完全不同的概念,切不可混淆。刷新仅针对动态RAM,刷新是定时进行的。动态RAM的许多记忆单元如果长期未被访问,就要及时补充电荷,否则信息就会丢失。重写则仅针对破坏性读出的存储器(如磁芯、单管动态RAM等),重写是随机进行的,某个存储单元只有在其被读取之后才需要重写。刷新通常是以存储体矩阵中的一行为单位进行的,而重写一般是以存储单元为单位进行的。

常见的刷新方式有集中式、分散式和异步式3种。

集中刷新方式的优点是读写操作时不受刷新工作的影响,因此系统的存取速度比较快。其主要缺点是在集中刷新期间必须停止读写,这一段时间称为死区,而且存储容量越大,死区就越长。

分散刷新方式没有死区,但是它有两个明显的缺点:第一是加长了系统的存取周期,降低了整机的速度;第二是刷新过于频繁,尤其是当存储容量比较小的情况下,没有充分利用

所允许的最大刷新间隔时间(2ms)。

异步刷新方式是前述两种方式的结合,它充分利用了最大刷新间隔时间,把刷新操作平均分配到整个最大刷新间隔时间内进行。这种方式虽然也有死区,但比集中刷新方式的死区小得多,死区的长度仅等于一个存储周期。

6. DRAM 的刷新中要注意的几个问题

(1) 无论是由刷新控制逻辑产生地址循环码逐行循环刷新,还是芯片内部自动刷新,都不依赖于外部的访问,刷新对 CPU 是透明的。

(2) 刷新通常是一行一行地进行的,每一行中各存储单元同时被刷新,故刷新操作时仅需要行地址,不需要列地址。

(3) 刷新操作类似于读出操作,但又有所不同。因为刷新操作仅是给栅极电容补充电荷,不需要输出信息。另外,刷新时不需要加片选信号,即整个存储器中的所有芯片同时被刷新。

(4) 因为所有芯片同时被刷新,所以在考虑刷新问题时,应当从单个芯片的存储容量着手,而不是从整个存储器的容量着手。

7. RAM 芯片结构

RAM 芯片通过地址线、数据线和控制线与外部连接。地址线是单向输入的,其数目与芯片容量有关。例如,容量为 1024×4 时,地址线有 10 根;容量为 64K×1 时,地址线有 16 根。数据线是双向的,既可输入,也可输出,其数目与数据位数有关。例如,1024×4 的芯片,数据线有 4 根;64K×1 的芯片,数据线只有 1 根。控制线主要有读写控制线和片选线两种,读写控制线用来控制芯片是进行读操作还是进行写操作,片选线用来决定该芯片是否被选中。

由于 DRAM 芯片集成度高、容量大,为了减少芯片引脚数量,DRAM 芯片把地址线分成相等的两部分,分两次从相同的引脚送入,两次输入的地址分别称为行地址和列地址。

RAM 芯片中的地址译码电路能把地址线送来的地址信号翻译成对应存储单元的选择信号。地址译码方式主要有下面两种。

(1) 单译码方式。又称字选法,它所对应的存储芯片结构是字结构的。对应任何一个地址码只有一条选择线(字线)有效,连接在这条字线上的所有存储单元同时被选中。

(2) 双译码方式。又称重合法,它所对应的存储芯片结构可以是位结构的,也可以是字段结构的。通常是把 K 位地址码分成接近相等的两段,一段用于在水平方向作 X 地址线,供 X 地址译码器译码;一段用于在垂直方向作 Y 地址线,供 Y 地址译码器译码。对应任何一个地址码只有一条 X 选择线和一条 Y 选择线有效,其交叉处的一个存储单元被选中。

小容量的 RAM 宜采用单译码方式,它对选择线的负载能力要求不高,但选择线数量多;大容量的 RAM 宜采用双译码方式,它的选择线比较少,但每条选择线的负载较重。

8. 用若干芯片构成主存

主存是整个存储系统的核心,通常分为随机存储器(RAM)和只读存储器(ROM)两大部分。RAM 和 ROM 在主存中是统一编址的。

存储芯片的容量是有限的,主存往往要由一定数量的芯片构成。根据主存所要求的容量和选定的存储芯片的容量,就可以计算出总的芯片数,即

$$\text{总片数} = \frac{\text{总容量}}{\text{存储芯片容量}}$$

1) 位扩展

当主存的字数与单个存储芯片的字数相同而位数(字长)不同时,可采用位扩展方式来组织多个芯片构成主存。

位扩展仅在位数方向扩展(加大字长),位扩展的连接方式是将各存储芯片的地址线、片选线和读写线相应地并联起来,而将各芯片的数据线单独列出。

2) 字扩展

当主存的字长与单个存储芯片的字长相同而字数不同时,可采用字扩展方式来组织多个芯片构成主存。

字扩展仅在字数方向扩展,而位数不变。字扩展将芯片的地址线、数据线、读写线并联,由片选信号来区分各个芯片。字扩展时有一个地址分配问题,地址线的高位部分通过译码器产生若干个片选信号 CS_i,分别选中若干个芯片中的一个。

3) 字和位同时扩展

实际中更多出现的是存储芯片的字数和字长均不能满足主存总容量要求的情况,这时要采用字和位同时扩展的方式来构成主存。

9. CPU 的访存地址

CPU 访问主存时需要给出地址码,其长度取决于 CPU 可直接访问的最大存储空间,一般要将其地址码分成片内地址和选片地址两部分。

片内地址由低位的地址码构成,其长度取决于所选存储芯片的字数。例如,芯片容量为 8K×4 和 8K×1,它们的片内地址相同,均为 13 位($2^{13}=8K$),片内地址用于从选中的芯片中选择出相应的存储单元,以进行数据的存取,故又称为字选。

选片地址由高位的地址码构成,用于选择存储芯片,故又称为片选地址。大多数情况下由选片地址通过译码后产生存储芯片的选片信号($\overline{CS}$)。

10. 选片地址的全译码方式

所谓全译码方式是指所有的选片地址全部参加译码。有两种情况必须采用全译码方式。

(1) 实际使用的存储空间与 CPU 可访问的最大存储空间相同。

例如,CPU 给出的访存地址长 16 位($A_{15}\sim A_0$),即可访问的最大存储空间为 64KB,选用的存储芯片容量为 16K×4,共 8 片,构成 4 个小组,这时片内地址为 14 位,选片地址为 2 位,2 位选片地址必须全部参加译码,才能产生 4 个选片信号,分别用作 4 个小组的选片信号。

(2) 实际使用的存储空间小于 CPU 可访问的最大存储空间,而对实际空间的地址分配有严格的要求。

例如,CPU 给出的访存地址长为 16 位($A_{15}\sim A_0$),即可访问的最大存储空间为 64KB,而系统中实际使用的存储空间只有 8KB,且选用存储容量为 4K×2 的芯片共 8 片,并要求其地址范围必须是 4000H~5FFFH,其地址译码方式如图 5-4 所示。

从图 5-4 可以看出,其地址分配如表 5-1 所示。

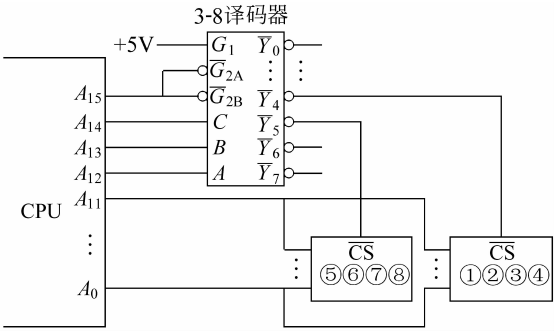

图 5-4　地址码采用全译码方式

表 5-1　全译码方式的地址分配

所选芯片号	选片地址				片内地址					译码输出	地址分配
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	…	A_0		
	0	0	0	0	0	0	0	…	0	$\overline{Y}_0=0$	0000H～0FFFH
	0	0	0	0	1	1	1	…	1		
	0	0	0	1	0	0	0	…	0	$\overline{Y}_1=0$	1000H～1FFFH
	0	0	0	1	1	1	1	…	1		
	0	0	1	0	0	0	0	…	0	$\overline{Y}_2=0$	2000H～2FFFH
	0	0	1	0	1	1	1	…	1		
	0	0	1	1	0	0	0	…	0	$\overline{Y}_3=0$	3000H～3FFFH
	0	0	1	1	1	1	1	…	1		
①②③④	0	1	0	0	0	0	0	…	0	$\overline{Y}_4=0$	4000H～4FFFH
	0	1	0	0	1	1	1	…	1		
⑤⑥⑦⑧	0	1	0	1	0	0	0	…	0	$\overline{Y}_5=0$	5000H～5FFFH
	0	1	0	1	1	1	1	…	1		
	0	1	1	0	0	0	0	…	0	$\overline{Y}_6=0$	6000H～6FFFH
	0	1	1	0	1	1	1	…	1		
	0	1	1	1	0	0	0	…	0	$\overline{Y}_7=0$	7000H～7FFFH
	0	1	1	1	1	1	1	…	1		

从表 5-1 中可以看出，按照这种译码方式，当前使用的存储空间的地址范围被严格地定义为 4000H～5FFFH，最大可扩充到 32KB 的存储空间。如果要将主存容量扩充到 64KB，只需将译码电路稍作修改即可。

全译码方式的共同特点是所使用的存储芯片的地址范围是唯一的。

11. 选片地址的部分译码方式

当实际使用的存储空间比 CPU 可访问的最大存储空间小，而且对其地址分配没有严格要求时，可采用部分译码方式。

例如，CPU 可提供的地址为 16 位（A_{15}～A_0），而实际使用的存储空间为 16KB，拟采用 4K×4 的存储芯片共 8 片，则可采用的部分译码方式如图 5-5 所示。

由于采用部分译码方式，使得各组芯片的地址不再是唯一的。各组芯片的地址分配如下：

第1组：0000H～0FFFH，4000H～4FFFH，8000H～8FFFH，C000H～CFFFH；

第2组：1000H～1FFFH，5000H～5FFFH，9000H～9FFFH，D000H～DFFFH；

第3组：2000H～2FFFH，6000H～6FFFH，A000H～AFFFH，E000H～EFFFH；

第4组：3000H～3FFFH，7000H～7FFFH，B000H～BFFFH，F000H～FFFFH。

可以看出，采用部分译码方式时，各组芯片出现了重叠的地址范围，其地址重叠区的个数取决于没有参加译码的地址码的位数。由于有2位地址码(A_{15}、A_{14})没有参加译码，所以每组芯片都出现4个地址重叠区。

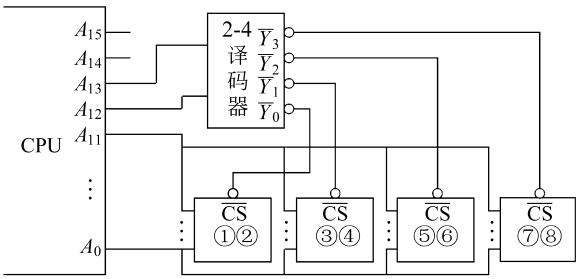

图5-5　地址码采用部分译码方式

12. CPU对主存的基本操作

CPU对主存进行读写操作时，首先在地址总线上给出地址信号，然后发出相应的读或写命令，并在数据总线上交换信息。读写的基本操作如下。

1) 读

读操作是指从CPU送来的地址所指定的存储单元中取出信息，再送给CPU，其操作过程如下：

(1) 地址→MAR→AB　　CPU将地址信号送至地址总线；

(2) Read　　CPU发读命令；

(3) Wait for MFC　　等待存储器工作完成信号；

(4) M (MAR)→DB→MDR　　读出信息经数据总线送至CPU。

2) 写

写操作是指将要写入的信息存入CPU所指定的存储单元中，其操作过程如下：

(1) 地址→MAR→AB　　CPU将地址信号送至地址总线；

(2) 数据→MDR→DB　　CPU将要写入的数据送至数据总线；

(3) Write　　CPU发写命令；

(4) Wait for MFC　　等待存储器工作完成信号。

13. 双口RAM

双口RAM是指同一个存储器具有两组相互独立的读写控制电路，是一种高速工作的存储器。它有两个独立的端口，分别具有各自的地址线、数据线和控制线，可以对存储器中任何位置上的数据进行独立的存取操作。

双口RAM的核心部分是用于数据存储的存储器阵列，可为左、右两个端口所共用。当两个端口的地址不相同时，在两个端口上进行读写操作，一定不会发生冲突。当任一端口被选中时，就可对整个存储器进行存取，每一个端口都有自己的片选控制和输出驱动控制。

当两个端口同时存取主存的同一存储单元时，就会因数据冲突造成数据存储或读取错误。两个端口对同一主存操作有4种情况：

(1) 两个端口不同时对同一地址单元存取数据；

(2) 两个端口同时对同一地址单元读出数据；

(3) 两个端口同时对同一地址单元写入数据；

(4) 两个端口同时对同一地址单元进行操作，一个写入数据，另一个读出数据。

在第(1)、(2)种情况时，两个端口的存取不会出现错误，第(3)种情况会出现写入错误，第(4)种情况会出现读出错误。为避免第(3)、(4)种错误情况的出现，双口RAM设计了硬件$\overline{\text{BUSY}}$功能输出，其工作原理如下：当左、右端口不对同一地址单元存取数据时，$\overline{\text{BUSY}_R}$=H，$\overline{\text{BUSY}_L}$=H，可正常存储。当左、右端口对同一地址单元存取数据时，有一个端口的$\overline{\text{BUSY}}$=L，禁止数据的存取。此时，两个端口中，哪个存取请求信号出现在前，则其对应的$\overline{\text{BUSY}}$=H，允许其存取数据；哪个存取请求信号出现在后，则其对应的$\overline{\text{BUSY}}$=L，禁止其写入数据。需要注意的是，两端口间的存取请求信号出现时间要相差5ns以上，否则仲裁逻辑无法判定哪个端口的存取请求信号在前；在无法判定哪个端口先出现存取请求信号时，控制线$\overline{\text{BUSY}_L}$和$\overline{\text{BUSY}_R}$只有一个为低电平，不会同时为低电平，这样就能保证对应于$\overline{\text{BUSY}}$=H的端口能进行正常存取，对应于$\overline{\text{BUSY}}$=L的端口不存取，从而避免双端口存取出现错误。

14. 多模块存储器

多模块存储器的每个模块具有相同的容量和存取速度，各模块都有独立的地址寄存器、数据寄存器、地址译码电路、驱动电路和读写电路，它们既能并行工作，又能交叉工作。

多模块交叉存储器是线性编址的，地址在各模块中有两种安排方式，分别是高位交叉编址(顺序方式)和低位交叉编址(交叉方式)。

高位交叉编址的多模块存储器用地址码的高位区分存储模块，用地址码的低位选择存储单元。低位交叉编址的多模块存储器用地址码的低位区分存储模块，用地址码的高位选择存储单元。

15. 程序访问的局部性原理

程序访问的局部性有两个方面的含义：时间局部性和空间局部性。时间局部性是指如果一个存储单元被访问，则可能该存储单元会很快被再次访问，这是因为程序存在着循环。空间局部性是指如果一个存储单元被访问，则该存储单元邻近的存储单元也可能很快被访问，这是因为程序中大部分指令是顺序存储、顺序执行的，数据一般也是以向量、数组、树、表等形式簇聚地存储在一起的。

也就是说，最近用过的、未来要用的指令和数据大多局限于正在使用的指令和数据，或存放在与这些指令和数据的存储位置上邻近的存储单元中。这样，就可以把目前常用或将要用到的信息预先放在存取速度最快的存储器M_1中，从而使CPU的访问速度大大提高。

CPU访存时的基本原则是由近到远：首先访问M_1；若在M_1中找不到所要的数据，就访问M_2，将包含所需数据的块或页面调入M_1；若在M_2中还找不到，就要访问M_3；依此类推。

16. Cache的基本工作原理

利用程序的局部性原理，把程序中正在使用的部分存放在一个高速的容量较小的

Cache中,使CPU的访存操作大多数针对Cache进行,从而使程序的执行速度大大提高。

Cache和主存都被分成若干个大小相等的块,每块由若干字节组成。由于Cache的容量远小于主存的容量,所以Cache中的块数要远少于主存中的块数,它保存的信息只是主存中最急需执行的若干块的副本。当CPU发出主存地址后,首先判断该存储字是否在Cache中,若是(称为命中),则直接访问Cache;若不是(称为不命中),则访问主存并将该字所在的主存块装入Cache。

命中率H定义为CPU产生的逻辑地址能在Cache中访问到的概率。在一个程序执行期间,设N_1为访问Cache的命中次数,N_2为访问主存的次数,则

$$H=\frac{N_1}{N_1+N_2}$$

Cache-主存存储层次的等效访问时间T_A与主存访问的启动时间有关:

- 若Cache访问和主存访问是同时启动的,$T_A=H\times T_{A_1}+(1-H)\times T_{A_2}$。
- 若Cache不命中时才启动主存,$T_A=T_{A_1}+(1-H)\times T_{A_2}$。

Cache-主存存储层次的访问效率为$e=\frac{T_{A_1}}{T_A}$。

17. Cache和主存之间的映射方式

由于Cache的容量小,因此Cache中的内容会经常地被新的主存块替换掉,它们之间就有地址映射问题。常见的地址映射的方式有3种:全相联映射、直接映射和组相联映射。

全相联映射就是让主存中任何一个块均可以映射(装入)到Cache中任何一个块的位置上。全相联映射方式比较灵活,Cache的块冲突概率最低,空间利用率最高,但是地址变换速度慢,而且成本高,实现起来比较困难。

直接映射是指主存中的每一个块只能被放置到Cache中唯一的一个指定位置,若这个位置已有内容,则产生块冲突,原来的块将无条件地被替换。直接映射方式是最简单的地址映射方式,成本低,易实现,地址变换速度快,但这种方式不够灵活,Cache的块冲突概率最高,空间利用率最低。

组相联映射实际上是全相联映射和直接映射的折中方案,所以其优点和缺点介于全相联和直接映射方式之间。当组数等于1(不再分组)时,组相联映射就变成全相联映射;当组数等于Cache中块的数目时,组相联映射就变成直接映射。

通常将组内2块的组相联映射称为二路组相联映射,组内4块的组相联映射称为四路组相联映射。关于组相联映射方式的具体实现方案,目前在不同的教材中有两种不同的说法,以二路组相联为例,图5-6(a)所示的方案称为方案一,图5-6(b)所示的方案称为方案二。

例如,主存的第9块,按方案一,将映射到Cache的第1组中,放在Cache的第2块或第3块的位置上;而按方案二,将映射到Cache的第0组中,放在Cache的第0块或第1块的位置上。比较这两个方案可以发现,两者的区别在于主存是否要按照Cache的大小再分区。这两种方案对应的主存地址是有区别的,如图5-7所示。

图5-7(a)是方案一对应的主存地址,分为3个字段。图5-7(b)是方案二对应的主存地址,分为4个字段,其中组内块号字段指出组相联映射中的一个Cache组(行)中块的数量,也就是组相联映射中的路数。相比之下,方案一的实现比较简单。