国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第16卷第3期 1994年9月 Vol. 16 No. 3

高速缓冲存储器教学演示系统设计与实现*

史 扬 张晨曦 张春元

（国防科技大学计算机系 长沙 410073）

摘 要 利用形象生动的动画演示来讲解教学内容是当今CAI发展的一个重要方向。本文论述了我们设计实现的有关高速缓冲存储器工作原理的教学动画演示系统。该系统由Cache工作过程、地址映象、LRU替换算法、LRU算法硬件实现等四个演示模块组成。文中介绍了这几个模块的设计思想、界面以及实现技术等。

关键词 高速缓冲存储器，动画，界面

分类号 TP391.7

在现代教育体系中，计算机辅助教学是一门新兴的学科、作为辅助教学工具的微机，具有快速运算能力、大存储容量和高质量的图形功能，能够有效地在教学的各个环节，如授课、复习、辅导、测验中发挥作用，大大提高学习效率。

计算机辅助教学效果的好坏取决于课件的好坏，一个好的课件应能充分利用微机的各种功能，通过文字、声音、动画形象逼真地演示教学内容，给学习者留下深刻印象。

本文所论及的是COMPUTER－MOVIE教学软件的一部分，即与高速缓冲存储器（Cache）有关的工作原理、概念、算法等演示模块的设计思想和实现技术。具体有Cache工作过程演示模块地址映象演示模块、LRU替换算法演示模块、LRU算法硬件实现演示模块。该软件为计算机原理及系统结构的教学提供了一个很好的工具，其实现技术、设计思想对其它课件的研制也具有参考价值。

1 LRU替换算法演示模块

1.1 设计思想

LRU算法的具体含义是近期最少使用，它在计算机体系结构中是一种重要的概念和算法，常用此算法进行Cache——主存页面、虚拟存储器的替换。其原理是选择处理机近期最少访问的页面作为被替换的页。根据程序局部性原理：当前最少使用的页很可能就是未来最少被访问的页。故采用此算法是比较合理的，但是实现真正的LRU算法是比较困难的，所以实际应用中是把近期最久未被访问过的页先替换掉，为简便起见，我们仍称之为LRU算法。

虚拟存储器的速度和Cache——主存机构的效率都与替换算法有关，通过模拟测得：采用LRU算法优于另外几种算法（如RAND算法、FIFO算法），其特点是随着主存容量（虚拟存储器）、Cache容量（Cache——主存机构）的增加而使命中率提高。这种特点的算法又称为堆栈型算法。对于LRU算法，把Cache中刚被访问过的页置于栈顶，而最久未被访问过的页被置于栈底。这是一个“提压”过程，

* 1993年12月6日收稿

在此过程中，处于栈顶的始终是最近刚被访问过的页，而处于栈底的始终是当前时刻最久未被访问的页，它将在下一次替换页面中被替换掉。“提压”过程的具体动作是：每当某一页面被命中，则将此页面所在的块从堆栈中抽出并移至栈顶。当未命中时将栈中所有块顺序下压一个位置，将新的请求页面置于栈顶。

本模块的设计就是采用这种“提压”方法，用动画的形式模拟算法的过程，同时计算并显示在某一页面地址流下堆栈内容的变化情况、命中次数和命中率等。

1.2 界面及使用说明

LRU 替换算法演示模块界面如图 1 所示。

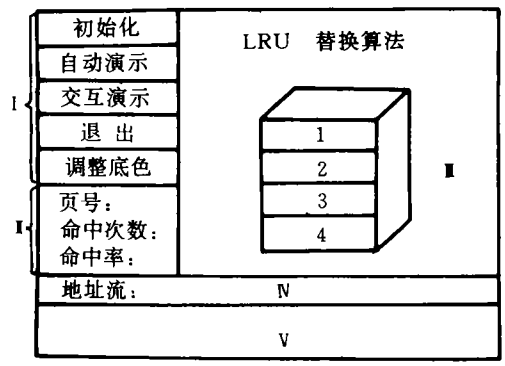

图 1 LRU 替换算法界面简图

界面如图分成五个部分：

Ⅰ为菜单项。当某一项在闪烁时表明该项处于被选状态，用上下键可以选择菜单项，键入回车则执行相应的功能，键入退出项回到上一级菜单。初始化项为初始化参数，用于设置堆栈的块数与地址流的长度等，选择此项后，屏幕的Ⅲ区弹出一菜单提示用户输入参数。自动演示项为自动演示 LRU 替换算法的过程，在过程中将自动给出中间结果，如命中次数、命中率和每一步堆栈的内容。交互演示项提供了与用户交互工作的机制，用户可随机输入页地址流，观察 LRU 的替换过程。退出项退出本模块。调整底色项的改变屏幕的背景色。

Ⅱ是结果显示区，显示替换过程中每一步的结果，如命中次数、命中率等。

Ⅲ是 LRU 算法的演示区，演示 LRU 算法的提压过程。

Ⅳ是主存页地址流显示区。

Ⅴ是 Cache 内容显示区，显示在每一页地址下 Cache 中内容的变换情况。

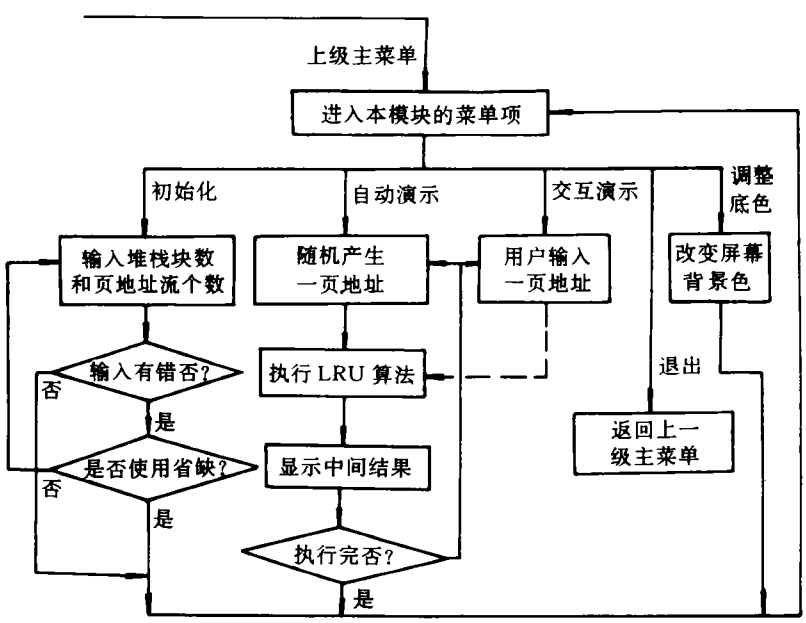

图 2 LRU 算法流程图

1.3 源程序流程图

本模块的 C 语言源代码约一千行，采用模块代程序设计，其流程图如图 2 所示。

1.4 实现

本模块中的主要技术难点是堆栈块的移动。在实现堆栈块的移动过程中，采用了如下两种技术。

(1) 画线抹线技术。其主要思想是在块的移动方向上用与块相同的颜色不断画线，同时在背离块的移动方向上用背景色画线，以保持块原有的大小和形状。这是一个循环的过程，在整体上给人以块移动的感觉。这是一种较难实现的方法，因为采用这种技术要考虑许多细节问题，尤其是在移动不规则图形时，所耗费的代价是巨大的，但是用这种方法实现的移动画面连续平稳，颇具动画效果。

(2) 剪贴技术。这种技术的实现使用了 C 语言的两个图形函数：函数 getimage (xleft, ytop, xright, ybottom, picture)，用于将屏幕上的某一区域的图形以点阵信息方式取到内存中；函数 putimage (xleft, ytop, picture, copy_put)，是将内存中存放的图形点阵信息恢复到屏幕的某一位置，这一位置由左上顶点坐标 (xleft, ytop) 来确定。利用这两个函数使每次画面做一细小的移动，用循环控制实现画面持续稳定的移动。

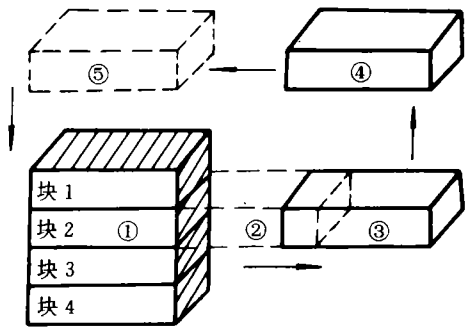

图 3 块 2 的移动轨迹

这两种技术根据堆栈块在画面中的不同位置而分别采用，因为 getimage 和 putimage 只能取下和恢复矩形区域的图形，而块的移动是长方体形状的移动，因此不能只采用剪贴技术。图 3 说明了在哪个阶段采用画线抹线技术，在哪个阶段使用剪贴技术。在图中假定块 2 抽出放到栈顶，箭头标明块的运动轨迹。图中块 2 从①—②—③采用了画线抹线技术，同时块 2 在位置②时块 1 的下落也采用了画线抹线技术。从③—④—⑤采用了剪贴技术。从位置⑤到块 1 的位置又采用了画线抹线技术。

2 Cache 工作过程演示模块

2.1 设计思想

Cache (高速缓冲存储器) 是介于处理机和主存之间的高速小容量的存储器，使得从 CPU 看，主存如同既具有 Cache 的高速度，又具有主存大容量的优点。它的内容是主存的部分拷贝。引入 Cache 的目的是为了降低处理机访问主存的频率，以提高处理速度。

为了便于管理和地址映象变换，Cache 和主存都以相同的尺寸划分成块。Cache ——主存机构的一般工作过程如下：首先处理机给出数据或指令的主存地址，通过 Cache ——主存地址映象变换机构判定访问单元所在块是否在 Cache 中。若在 (命中)，则经地址映象变换机构将主存地址变换成 Cache 地址，从 Cache 中读出单元内容送处理机；若不在 (不命中) 则需从主存中将单元所在块调入 Cache，同时单元内容送往处理机。在向 Cache 调主存块时，还要根据 Cache 的状态分两种情况处理：(1) Cache 中还未装满主存块的内容，这时可直接在 Cache 中选择一块装入；(2) Cache 中已装满主存块的内容，这时需根据 Cache 替换策略将 Cache 中某块替换到主存中，再将单元所在的主存块调到此 Cache 块中。综上所述，可将 Cache 的工作过程归纳成三种情况：(1) 命中；(2) 未命中，可装入；(3) 未命中，不可装入。

在本模块中，对应每种情况均给出了相应的动态工作流程。在每种过程中动态地显示了地址流，数据流以及主存、Cache、地址映象变换机构，及 Cache 替换策略机构的工作情况。

2.2 界面及使用说明

本模块的界面与 LRU 替换算法演示模块相似，这里不再叙述。

2.3 演示内容说明及程序流程图

下面就这三种情况详细描述演示的全过程。

1. 命中。首先从处理机来一主存地址，在画面上表现为一条动态延伸绿色指到主存地址中，并使

之变为绿色，表明地址已打入主存地址寄存器。接着从主存地址寄存器出来一条延伸线到Cache——主存地址映象交换机构，并使之闪烁，表明其正在工作，从地址映象变换机构引出一红色虚线并标明“命中”。随后从主存地址、地址映象变换机构出来两条同步延伸绿线指到Cache地址，并使之变为绿色，表明已形成Cache地址。接着从Cache地址画一延伸线到Cache并使之闪烁表明在Cache中取出单元内容。最后从Cache中出来一红色延伸线到处理机。

2. 未命中，不需替换。演示前半部分与1相同，只是在地址映象变换时显示“未命中”，而后从主存出来一绿色延伸线到主存并使之闪烁表示要在主存中取数据，接着从主存出来一蓝色的延伸线分别到Cache和处理机，并使它们同时闪烁表示主存数据送处理机和向Cache调主存块的操作同时进行。

3. 未命中，需替换。大部分过程与2相同，只多了替换页面工作。若块内容作过修改，则需先将此页写回主存，否则直接替换此块。

本模块程序流程图如图4所示：

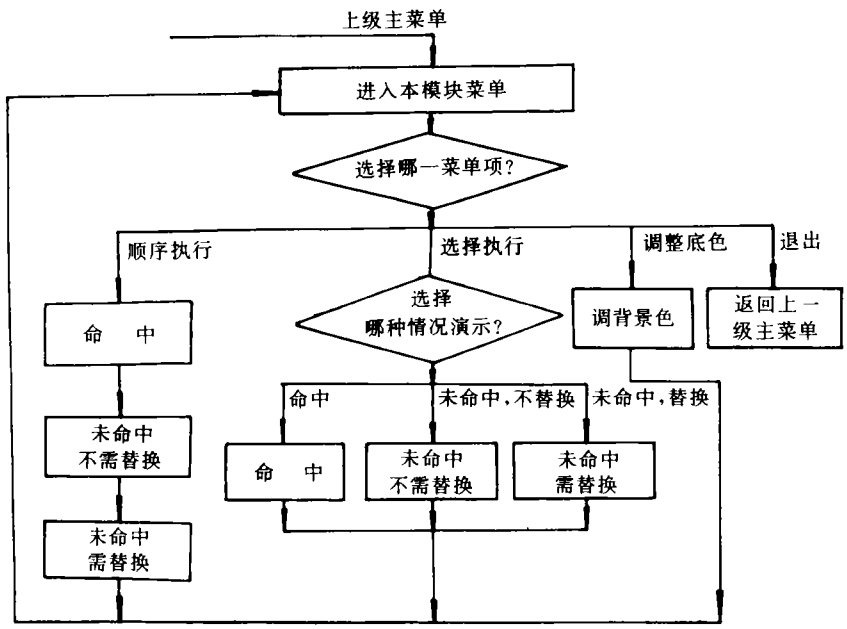

图 4 Cache 工作过程演示模块流程图

3 Cache 地址映象演示模块

3.1 设计思想

由于在Cache——主存机构中，主存容量一般远大于Cache的容量，因此存在着如何将主存中的块对应到Cache中块的地址映象问题。较常用的地址映象规则有：直接映象、全相联映象、组相联映象。

(1)直接映象。直接映象是最简简单的映象方式。在这种方式下，主存中的每一块只能映象到Cache中的某一特定块中去。其映象规则为：

$$j=i\ (\mathrm{mod})\ 2^{cb}$$

其中i为主存块号，j为Cache块号，Cb是Cache地址中块地址部分的位数。这种映象方式特点是硬件简单、速度快但是块冲突率高。

(2)全相联映象。主存中任何一块都可以映象到Cache中任何一块。它的特点是块冲突率小，Cache

利用率高，但是硬件实现复杂代价高。

(3) 组相联映象。这是介于以上两种方式之间的折衷方案，具有两者的优点。在这种方式下，Cache和主存均分成组，组间采用直接映象，组内采用全相联映象。

本模块采用动画方法演示这三种映象方式。

3.2 源程序流程图

当本模块演示时，程序自动地根据当前的参数值分析采用的是哪一种映象方式，从而进入相应的演示程序。例如：Cache 块为 4，主存块为 8，每组块数为 4 时，表示整个 Cache 为一组，因此此时采用的是全相联映象。每组块数为 4 则为直接相联映象。每组块数为 2，Cache 被分为两组，这时采用的是组相联映象方式。图 5 为本演示模块之流程图。

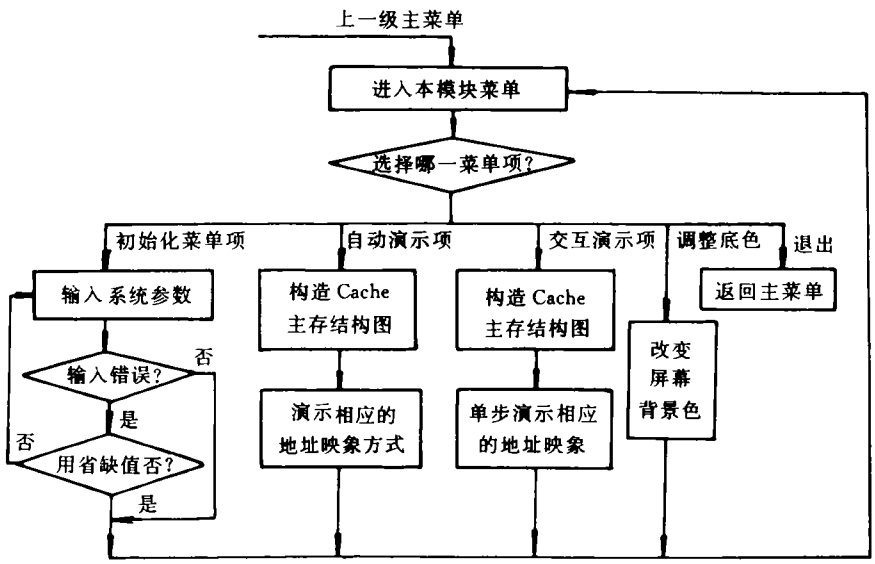

图 5 Cache 地址映象演示模块流程图

4 LRU 算法硬件实现演示模块

4.1 设计思想

LRU 算法是一种常用的页替换算法，其实现的好坏直接影响 Cache ——主存机构的效率。常见的硬件实现有以下两种：

(1) 堆栈法。利用堆栈，使栈底恒为近期最久未被访问的块号。因此可组成 R_0 (Cache 块数) 顶堆栈，把被访问的块号与堆栈中各项比较，相符则移至栈顶，否则把此块压入栈中，原栈中所有项下移一位置。当栈满时栈底项就是下次被替换的项。本模块中用 D 触发器实现。

(2) 比较对法。让各块成对组合，用触发器状态表示每个比较对内的访问次序，从而找出被替换的块。

本模块中使用动画的手法分别演示这两种硬件实现的原理及工作过程。

4.2 演示画面与说明

(1) LRU 算法的堆栈法实现。堆栈法实现的画面中有八个 Cache 块号，分别对应 Cache 的八个块，如图 6 所示。用户输入一个二进制（三位）数，代表要访问的 Cache 块，此时内容与该地址相同的单元的左边所有单元向右顺移一个位置，最左边单元接收新的块号。用 "CP" 闪烁和 D 触发器闪烁表示 CP 正跳变以及打入 D 触发器，用动态生长线表示信息的传输。红线代表 "1"，白线代表 "0"。最左栈单元

表示栈顶，最右元表示栈底，用户输入要访问的块号，自动演示信号的控制过程以及栈中各单元的变化情况。图中 NA，…，NG 是 LRU 堆栈中对应块的控制信号。如果刚被访问的块号不是 A，则 NA 为 1.

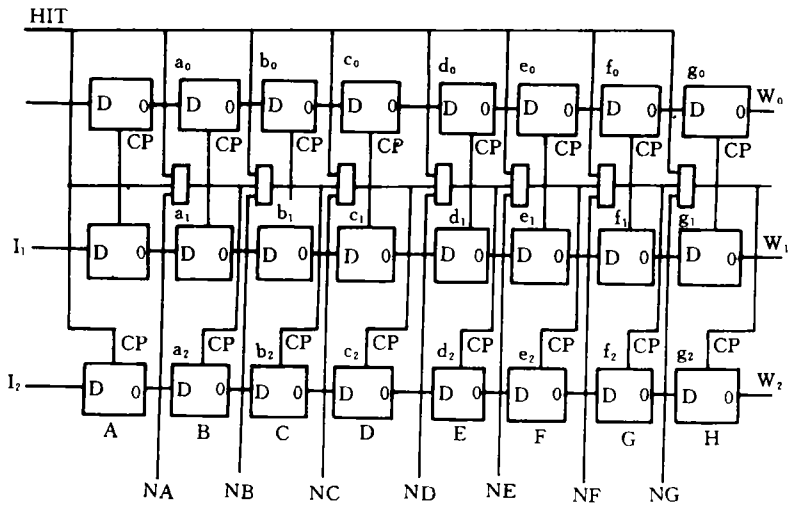

图 6　堆栈法实现演示画面

图中：$NA=(a_0+I_0)\ \vee\ (a_1+I_1)\ \vee\ (a_2+I_2)$

$NB=(b_0+I_0)\ \vee\ (b_1+I_1)\ \vee\ (b_2+I_2)$

$NC=(c_0+I_0)\ \vee\ (c_1+I_1)\ \vee\ (c_2+I_2)$

$ND=(d_0+I_0)\ \vee\ (d_1+I_1)\ \vee\ (d_2+I_2)$

$NE=(e_0+I_0)\ \vee\ (e_1+I_1)\ \vee\ (e_2+I_2)$

$NF=(f_0+I_0)\ \vee\ (f_1+I_1)\ \vee\ (f_2+I_2)$

$NG=(g_0+I_0)\ \vee\ (g_1+I_1)\ \vee\ (g_2+I_2)$

(2) LRU 算法实现。LRU 算法比较对法硬件实现演示画面如图 7 所示。在这里演示四个 Cache 块的比较对法的实现，共需 6 个 T 触发器。图 7 中下面一行六个块代表 T 触发器，上面的一排四个块为与门。用户每次输入所访问的 Cache 块号，该程序演示硬件的动作情况，用红线表示信号“1”，白线表示“0”．动态延伸表示信号的传输过程。

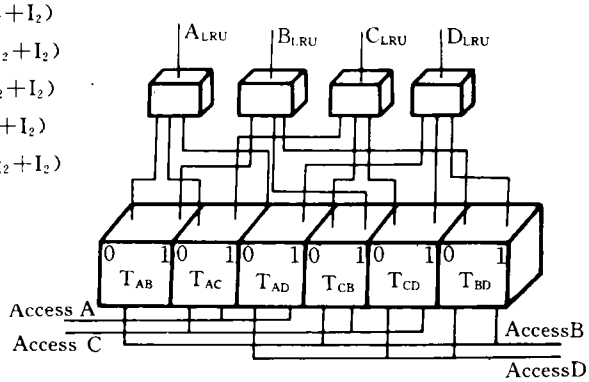

图 7　比较对法实现的演示画面

图中：$A_{LRU}=\overline{T_{AB}}\cdot\overline{T_{AC}}\cdot\overline{T_{AD}}$　　$B_{LRU}=\overline{T_{BD}}\cdot T_{AB}\cdot T_{CB}$

$C_{LRU}=T_{AC}\cdot\overline{T_{CB}}\cdot\overline{T_{CD}}$　　$D_{LRU}=T_{AD}\cdot T_{CD}\cdot T_{BD}$

如果为 1，则表示对应的 Cache 块最久未被访问过，下次将被替换。

4.3　程序流程图

程序流程比较直观简单，不再赘述。

5　结束语

本文介绍了 Cache 工作过程、地址映象、LRU 替换算法、LRU 算法的硬件实现等四个演示程序的

设计思想与实现技术。该系统对于计算机体系结构教学具有重要的实用价值。本文介绍的思想和技术对于设计其它教学软件也有参考价值。

参 考 文 献

1 李勇，刘恩林．计算机体系结构．长沙：国防科技大学出版社，1988

2 张晨曦等．COMPUTER MOVIT 的设计与实现技术．国防科技大学校庆四十周年论文集第七卷，1993

Design and Implementation of Cache Working Principle Teaching System

Shi yang Zhang Chenxi Zhang Chunyuan

(Department of Computer Science)

Abstract

A very important trend in CAI development is to take advantage of vivid animation to illustrate the content of the course. In this paper, we discuss an animate-based courseware we designed and implemented, the Cache working principle teaching system made up of four parts: Cache working process module; address mapping module; LRU replacing algorithm module; and LRU hardware implementation. The details of design methods, human-machine interface and implementation technology are presented in this paper.

Key words cache, animation, interface

(上接 67 页)

Well-Designed Multi-Variable Function Generator

Du Tieta Liu Mingliang

(Department of Computer Science)

Abstract

The techniques of the optimized multi-variable function generator (MVFG) are introduced in this paper, including quick function interpolation, elimination of redundant search of break point, and the strategy of quich search of break point. Practice shows that the MVFG is much more efficient than the other common MVFG. The time spent on multi-variable function generation is reduced by 50%.

Key words function generation, break point search, function interpolation, system simulation