### 辽师

### 《操作系统实验四课程报告》

——模拟页式虚拟存储管理中硬件地址变换和缺页中断，并用LRU算法处理缺页中断

### 

**学 院：计算机与信息技术学院**

**专 业：计算机科学与技术（师范）**

**班级序号：01班**

**学 号：201621012299**

**学生姓名：周惠馨**

**完成时间：2018.11.23**

1. **实验目的**
   1. 通过本实验的模拟算法理解和加深页式存储管理的地址变换过程。
   2. 通过模拟LRU算法加强理解和掌握OS中实现虚拟存储器的关键技术。
2. **实验环境**

Linux下的C，Turbo C或Visual C

1. **实验内容**
   1. 数据结构

（1）假定主存的每块长度1KB，现有一个共7页的作业，其副本在磁盘上，系统为该作业分配了4个主存块，且该作业的第0页至第3页已经装入主存，其余3页尚未装入主存，该作业的页表如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 页号 | 标志 | 主存块号 | 修改标志 |
| 0  1  2  3  4  5  6 | 1  1  1  1 | 5  8  9  12 | 0  0  0  0 |

（2）该作业执行的指令序列如下：

操作 页号 页内地址

+ 0 70

+ 1 50

\* 2 15

S（存） 3 21

L（取） 0 56

— 6 40

M（移位）4 53

+ 5 23

S 1 37

L 2 78

+ 4 10

S 6 84

（3）假定作业被选中，把开始的m个页面装入内存则数组元素可选定m个，设m=4，初始时假定p[0]=0，p[1]=1，p[2]=2，p[3]=3,在上述指令前提下页面的变化为

最新 3 0 6 4 5 1 2 4 6

2 3 0 6 4 5 1 2 4

1 2 3 0 6 4 5 1 2

最老 0 1 2 3 0 6 4 5 1

采用LRU置换算法，产生缺页中断后，OS总是淘汰由最后一个元素所指示的页，再把访问页装入淘汰页所占主存块中，页号登记到数组的第一个元素中，重新启动刚才那条指令执行。即数组中第一个元素总是当前刚访问的页号，最久没被子访问过总是由最后一个元素指出。

2．流程图

输出OUTj

Y

N

输出绝对地址

置L页修改标志=1

调整数组内容

形成绝对地址

L页标志=1?

查页表

开始

取一条指令

取指令中访问页号，→L

是”存”指令?

Y

有后继指令吗?

结束

取指令

N

Y

产生缺页中断

j页修改

标志=1?

N

j**:**=被淘汰页

输出INL

调整数组内容

修改页表

j=>退出数组

L=>数组中第

一元素

3.要求能打印出每条指令执行完之后的绝对地址以及P数组变化。

1. **实验代码**

#include <stdio.h>

#include <string.h>

#define N 12

typedef struct

{

int page; //页号

int flag; //标志

int block; //主存块号

int mark; //修改标志

}table; //页表

typedef struct

{

char operation; //操作

int page; //页号

int address; //页内地址

}list; //指令序列

table table1[7]={{0,1,5,0},{1,1,8,0},{2,1,9,0},{3,1,12,0},{4,0,0,0},{5,0,0,0},{6,0,0,0}};

list list1[N]={{'+',0,70},{'+',1,50},{'\*',2,15},{'S',3,21},{'L',0,56},{'-',6,40},{'M',4,53},{'+',5,23},{'S',1,37},{'L',2,78},{'+',4,10},{'S',6,48}};

int vec[5]={0};

int flg[5]={0};

int main()

{

int n,i,j,k=0,pos,page,flag,mark,memaddress;

for(i=0;i<N;i++)

{

mark=0;

page=list1[i].page;

flag=table1[page].flag;

if(i<4)

{

for(j=i;j>=0;j--)

vec[j]=vec[j-1];

vec[0]=page;

flag=1;

if(list1[i].operation=='S')

table1[page].mark=1;

}

else

{

for(j=0;j<4;j++)

{

if(page==vec[j])

{

mark=1;

pos=j;

}

}

if(mark)

{

for(j=pos+1;j>=0;j--)

vec[j]=vec[j-1];

vec[0]=page;

}

else

{

for(j=4;j>=0;j--)

vec[j]=vec[j-1];

pos=vec[4];

table1[page].block=table1[pos].block;

table1[page].flag=table1[pos].flag;

table1[pos].flag=0;

table1[pos].mark=0;

table1[pos].block=0;

vec[0]=page;

if(list1[i].operation=='S')

table1[page].mark=1;

}

for(n=0;n<4;n++)

printf("%d\t",vec[n]);

printf("");

}

if((list1[i].page<7)&&i>3)

{

memaddress=table1[page].block\*1024+list1[i].address;//物理地址

printf("页号为%d的操作是%c，它的物理地址为：%ld\n",table1[page].page,list1[page].operation,memaddress);

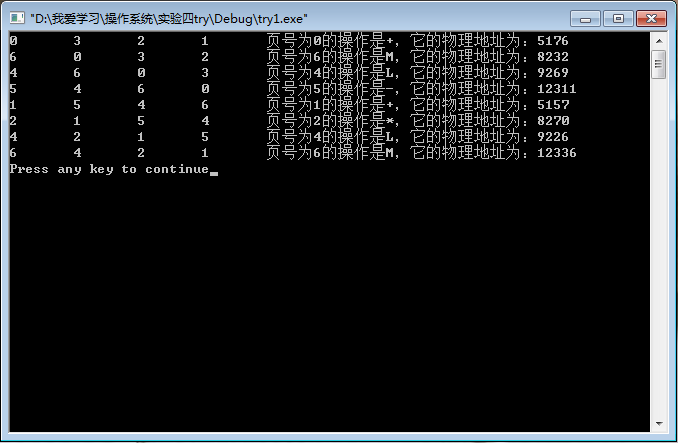
}

}

return 0;

}

1. **实验结果**



1. **思考分析**

最近最久未使用（LRU）置换算法原理就是：当需要淘汰某页面时，选择当前一段时间内最久未使用过的页先淘汰，即淘汰距当前最远的上次使用的页。

LRU算法提高内存利用率，提供了内外存进程对换机制，内存空间的分配和回收均以页为单位进行，一个进程只需将其一部分调入内存便可运行，还支持请求调页的存储管理方式。

使用LRU算啊时要注意对修改标志mark和标志flag的随时更改。