# 实验2 虚拟存储器 实验报告

信息安全1501 沈鑫楠

3150604028

### 1.实验题目

#### 第一题：

模拟分页式存储管理中硬件的地址转换和产生缺页中断。

#### 第三题：

用近少用（LRU）页面调度算法处理缺页中断。

### 2.实验目的与要求

在计算机系统中，为了提高主存利用率，往往把辅助存储器（如磁盘）作为主存储器的 扩充，使多道运行的作业的全部逻辑地址空间总和可以超出主存的绝对地址空间。用这种办 法扩充的主存储器称为虚拟存储器。通过本实验帮助同学理解在分页式存储管理中怎样实现 虚拟存储器。

### 3.程序中使用的数据结构及符号说明

#### 第一题

结构体page表示作业的页表，存储页号、标志、主存块号和在磁盘上的位置

结构体order表示指令序列，存储操作、页号和单元号。

struct page

{

int num;//页号

int sign;//标志

int block;//主存块号

int place;//在磁盘上的位置

};

struct order

{

string operate;//操作

int page\_num;//页号

int unit\_num;//单元号

};

#### 第三题

结构体page表示作业的页表，存储页号、标志、主存块号、修改标志和在磁盘上的位置

结构体order表示指令序列，存储操作、页号和单元号。

struct page

{

int num;//页号

int sign;//标志

int block;//主存块号

int modifysign;//修改标志

int place;//在磁盘上的位置

};

struct order

{

string operate;//操作

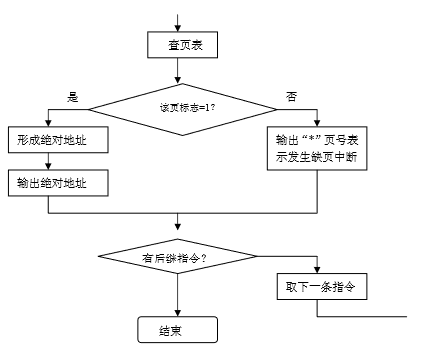
int page\_num;//页号

int unit\_num;//单元号

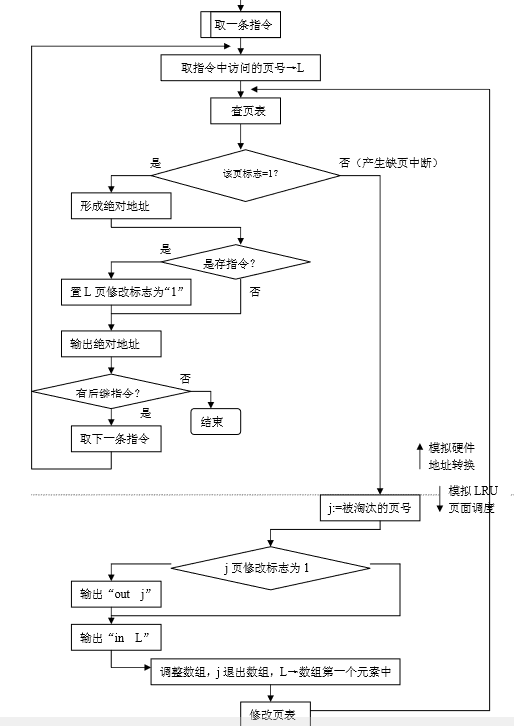
};

### 4.流程图

#### 第一题

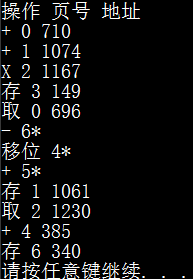


#### 第三题



### 5.实验测试结果与结果分析

#### 第一题



输出正确，符合题目要求

#### 第三题



输出正确，符合题目要求

### 6.原程序清单

#### 第一题

#include <iostream>

#include <string>

#include <stdlib.h>

#include <time.h>

#define Length 128

#define Unrefered -1

using namespace std;

struct page

{

int num;//页号

int sign;//标志

int block;//主存块号

int place;//在磁盘上的位置

};

struct order

{

string operate;//操作

int page\_num;//页号

int unit\_num;//单元号

};

struct page assignment[7] = { { 0, 1, 5, 11 }, { 1, 1, 8, 12 }, { 2, 1, 9, 13 }, { 3, 1, 1, 21 }, { 4, 0, Unrefered, 22 }, { 5, 0, Unrefered, 23 }, { 6, 0, Unrefered, 121 } };//定义初始作业的页表

bool exist(int num,int n)//判断主存块号是否已存在

{

for (int i = 0; i < 7;i++)

{

if (i==n)

{

continue;

}

if (assignment[i].block==num)

{

return true;

}

}

return false;

}

int main()

{

//打印初始页表

printf("初始页表：\n");

for (int i = 0; i < 7;i++)

{

printf("%d %d %d %d\n", assignment[i].num, assignment[i].sign, assignment[i].block, assignment[i].place);

}

struct order sequence[12];

//输入需要执行的指令序列

printf("请输入作业的指令序列:\n");

printf("操作 页号 单元号\n");

for (int i = 0; i < 12;i++)

{

cin >> sequence[i].operate >> sequence[i].page\_num >> sequence[i].unit\_num;

}

//输出每条指令执行时访问的地址

printf("操作 页号 地址\n");

for (int i = 0; i < 12;i++)

{

if (assignment[sequence[i].page\_num].sign==1)//标志为1，在内存中

{

int addr = assignment[sequence[i].page\_num].block\*Length + sequence[i].unit\_num;//计算出物理地址

cout << sequence[i].operate << " " << sequence[i].page\_num << " " << addr << endl;//输出

}

else//标志为0，缺页中断

{

cout<<sequence[i].operate<<" "<<sequence[i].page\_num<<"\*"<<endl;//输出

//中断处理，把页面调入主存

assignment[sequence[i].page\_num].sign = 1;

assignment[sequence[i].page\_num].block = 0;

do

{

assignment[sequence[i].page\_num].block++;

} while (exist(assignment[sequence[i].page\_num].block, sequence[i].page\_num));//分配主存块号

}

}

return 0;

}

#### 第三题

#include <iostream>

#include <string>

#include <stdlib.h>

#include <time.h>

#include <vector>

#define Length 128

#define Unrefered -1

using namespace std;

struct page

{

int num;//页号

int sign;//标志

int block;//主存块号

int modifysign;//修改标志

int place;//在磁盘上的位置

};

struct order

{

string operate;//操作

int page\_num;//页号

int unit\_num;//单元号

};

struct page assignment[7] = { { 0, 1, 5, 0, 11 }, { 1, 1, 8, 0, 12 }, { 2, 1, 9, 0, 13 }, { 3, 1, 1, 0, 21 }, { 4, 0, Unrefered, 0, 22 }, { 5, 0, Unrefered, 0, 23 }, { 6, 0, Unrefered, 0, 121 } };//定义初始页表

bool exist(int num, int n)//判断主存块号是否已存在

{

for (int i = 0; i < 7; i++)

{

if (i == n)

{

continue;

}

if (assignment[i].block == num)

{

return true;

}

}

return false;

}

int main()

{

vector<int> vec;//定义向量表示空闲块

vec.push\_back(3);

vec.push\_back(2);

vec.push\_back(1);

vec.push\_back(0);

//初始化

vector<int>::iterator it;

printf("初始页面:");

for (it = vec.begin(); it != vec.end(); ++it)//遍历输出初始页面

{

printf("%d\t", \*it);

}

printf("\n");

fflush(stdin);

fflush(stdout);

struct order sequence[12];

printf("请输入作业的指令序列:\n");//输入要执行的指令序列

printf("操作 页号 单元号\n");

for (int i = 0; i < 12; i++)

{

cin >> sequence[i].operate >> sequence[i].page\_num >> sequence[i].unit\_num;

}

for (int i = 0; i < 12; i++)

{

int L = sequence[i].page\_num;//要访问的页面

if (assignment[L].sign == 1)//在主存中

{

if (sequence[i].operate == "存" || sequence[i].operate == "cun")//被修改

{

assignment[L].modifysign = 1;//修改标志置为1

}

int addr = assignment[L].block\*Length + sequence[i].unit\_num;

int j = 0;

for (it = vec.begin(); it != vec.end(); ++it, ++j)

{

if (\*it == L)//改变这一页在数组中的位置

{

vec.erase(it);

vec.insert(vec.begin(), L);

break;

}

}

}

else//缺页

{

int J = vec.back();//数组中最久未被访问的页

if (assignment[J].modifysign==1)//被修改过

{

cout << "OUT " << J << endl;//调出

}

cout << "IN " << L << endl;//调入新的页

vec.pop\_back();

vec.insert(vec.begin(),L);//插入到数组中

//旧的页被调出，结构体中的内容改变

assignment[J].sign = 0;

assignment[J].block = Unrefered;

//新的页被调入，结构体中相关内容改变

assignment[L].sign = 1;

assignment[L].block = 0;

do

{

assignment[L].block++;

} while (exist(assignment[L].block,L));

}

//输出执行这条指令后主存中页面的状态

cout << "当前状态：";

for (it = vec.begin(); it != vec.end(); ++it)

{

printf("%d\t", \*it);

}

cout << endl;

}

return 0;

}