**黑龙江大学**

**实 验 报 告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **课程名称** | **计算机操作系统** | | | | | |
| **实验项目名称** | **分页式存储管理** | | | | | |
| **实验时间**  **（日期及节次）** | **2016年9月27日 1-2节** | | | | | |
| **专业** | **计算机** | | **学生所在学院** | | **计算机科学技术学院** | |
| **年级** | **2014级** | | **学号** | | **20123349** | |
| **姓名** | **刘伟** | | **指导教师** | | **朱敬华** | |
| **实验室名称** |  | | | | | |
| **实验成绩** | **预习情况** | **操作技术** | **实验报告** | **附加：综合创新能力** | | **实验**  **综合成绩** |
|  |  |  |  | |  |
| **教师签字** |  | | | | | |

**黑龙江大学教务处**

**实验二 分页式存储管理**

**一．实验名称：**分页式存储管理

**二．实验目的：**

（一）实现分页式存储地址转换过程，在此基础上实现请求分页的地址转换。分页式存储管理系统是内存非连续存储管理中基本的方法，可以通过把一个作业分成多个页面分配到不连续的内存块中去。实验可以通过位示图的方式来模拟内存的使用情况，为每个作业建立页表用于完成正确的地址转换。

（二）实现请求页式地址转换中出现的缺页现象中，用到的先进先出FIFO、最近最久未使用LRU、最佳OPT置换算法。熟悉文件的基本操作（读、写）。

**三．实验内容：**

用PCB表示整个进程实体，利用随机数方法或键盘控制方法模拟进程执行中产生的事件。或者利用鼠标或者键盘中断的基于图形接口方式的进程控制管理。

**四．实验代码**

var http = require('http');

var qs = require('querystring');

var url = require('url');

var items = [];

var server = http.createServer(function(req,res){

var urlObj = url.parse(req.url);

var pathname = urlObj.pathname;

var id = qs.parse(urlObj.query).id;

if(pathname == '/'){

switch(req.method){

case 'GET':

show(res);

break;

case 'POST':

convert(req,res);

break;

default:

badRequest(res);

}

}else if(pathname == '/delete'){

remove(id);

show(res);

}

});

server.listen(3000);

console.log("Web Server is listening port 3000");

var bitMap = function(){

var bitArr = [];

for(var i=0; i<=63; i++){

bitArr[i] = parseInt(Math.random()\*2);

}

return bitArr;

}

var bitmap = bitMap();

var pagetable = [];

var len = 4;

function show(res,logic,pageNo,blockNo,pageAdd,physical){

if(!logic){

logic = '未输入';

}

if(!pageNo){

pageNo = '未输入';

}

if(!blockNo){

blockNo = '未输入';

}

if(!pageAdd){

pageAdd = '未输入';

}

if(!physical){

physical = '未输入';

}

res.setHeader('Content-Type','text/html');

res.setHeader('Content-Length',Buffer.byteLength(html));

res.statusCode = 200;

res.end(html);

}

var arr = [];

var stackF = [];

var stackL = [];

var countF = 0;

var countL = 0;

var total = 0;

var convert = function(req,res)

{

var body = "";

req.setEncoding('utf8');

req.on('data',function(chunk){

body+=chunk;

});

req.on('end',function(){

var logic = qs.parse(body).logic;

var pageNo = parseInt(logic / 1024);

var pageAdd = logic % 1024;

var blockNo;

var physical;

if(pagetable[pageNo-1])

{

blockNo = pagetable[pageNo-1];

physical = (blockNo)\*1024 + pageAdd;

arr.push(pageNo);

FIFO(pageNo);

LRU(pageNo);

total++;

}else{

blockNo = bitmap.indexOf(0);

if(pageNo-1<len){

if(blockNo!=-1){

pagetable[pageNo-1] = blockNo+1;

physical = (blockNo+1)\*1024 + pageAdd;

bitmap[blockNo] = 1;

arr.push(pageNo);

FIFO(pageNo);

LRU(pageNo);

total++;

}

}else{

physical = (blockNo+1)\*1024 + pageAdd;

arr.push(pageNo);

FIFO(pageNo);

LRU(pageNo);

total++;

}

}

show(res,logic,pageNo,blockNo,pageAdd,physical);

});

}

function FIFO(num) {

flag = true;

var lab = true;

if(stackF.length>=len){

for(var j = 0; j < stackF.length ;j++){

if(num == stackF[j]){

flag = !flag;

break;

}

}

if(flag){

lab = stackF.shift()

stackF.push(num);

countF++;

}

}else{

for(var j = 0; j < stackF.length ;j++){

if(num == stackF[j]){

flag = false;

break;

}

}

if(flag){

stackF.push(num);

countF++;

}

}

return lab;

}

function LRU(num) {

flag = true;

var lab = true;;

if(stackL.length>=len){

for(var j = 0; j < stackL.length ;j++){

if(num == stackL[j]){

stackL.splice(j,1);

stackL.push(num);

flag = !flag;

break;

}

}

if(flag){

lab = stackL.shift()

stackL.push(num);

countL++;

}

}else{

for(var j = 0; j < stackL.length ;j++){

if(num == stackL[j]){

stackL.splice(j,1);

stackL.push(num);

flag = !flag;

break;

}

}

if(flag){

stackL.push(num);

countL++;

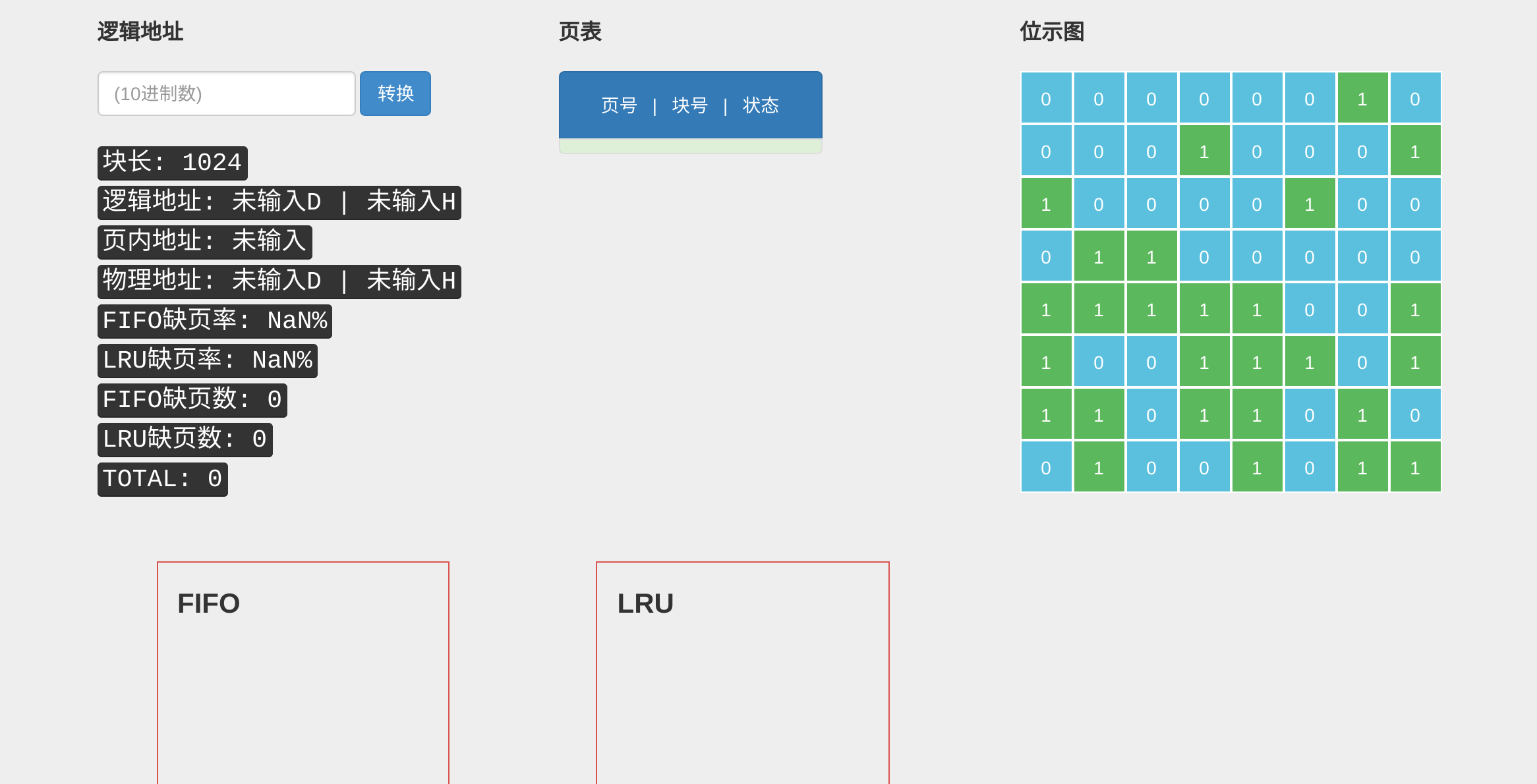
}

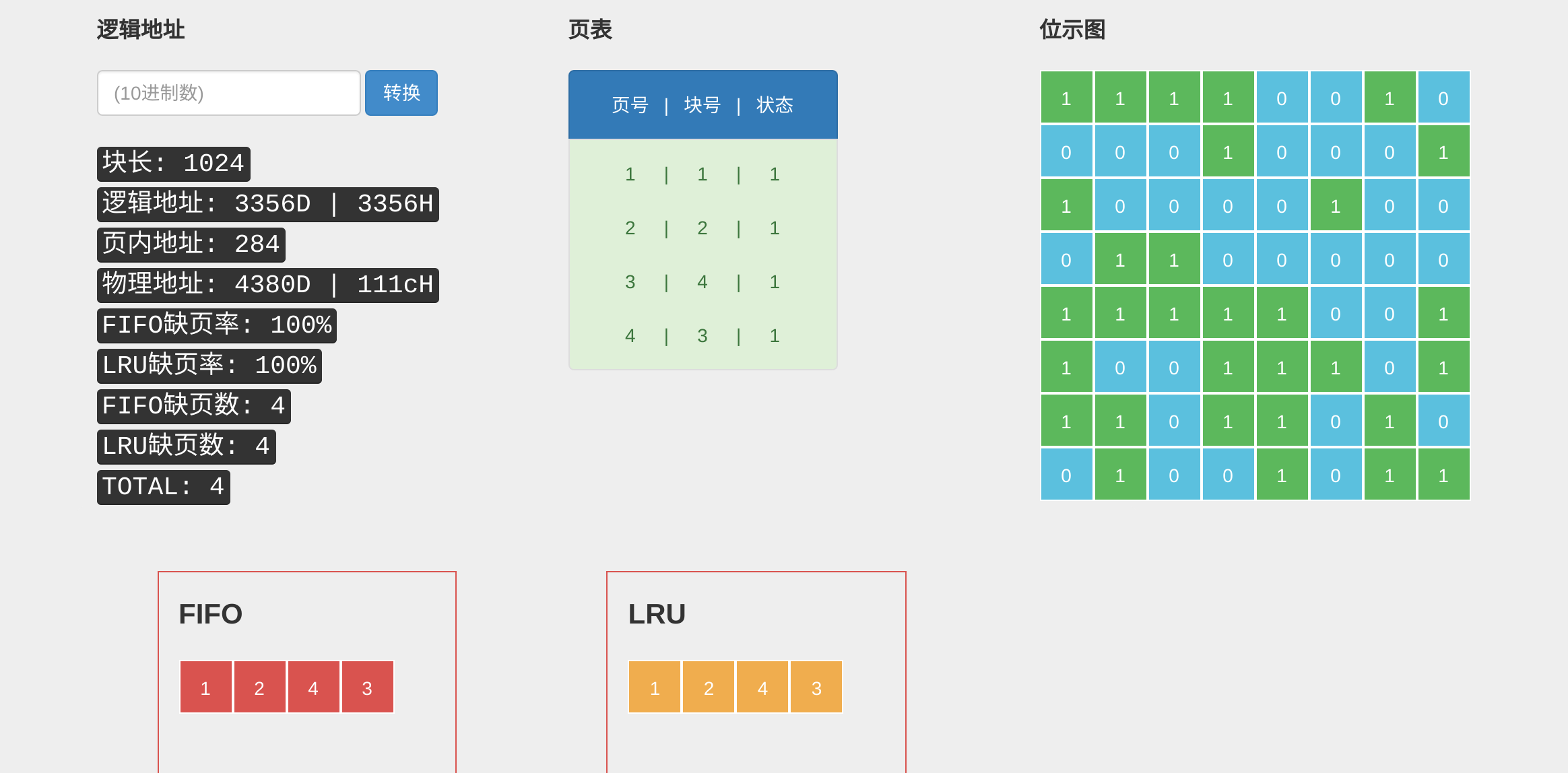
}

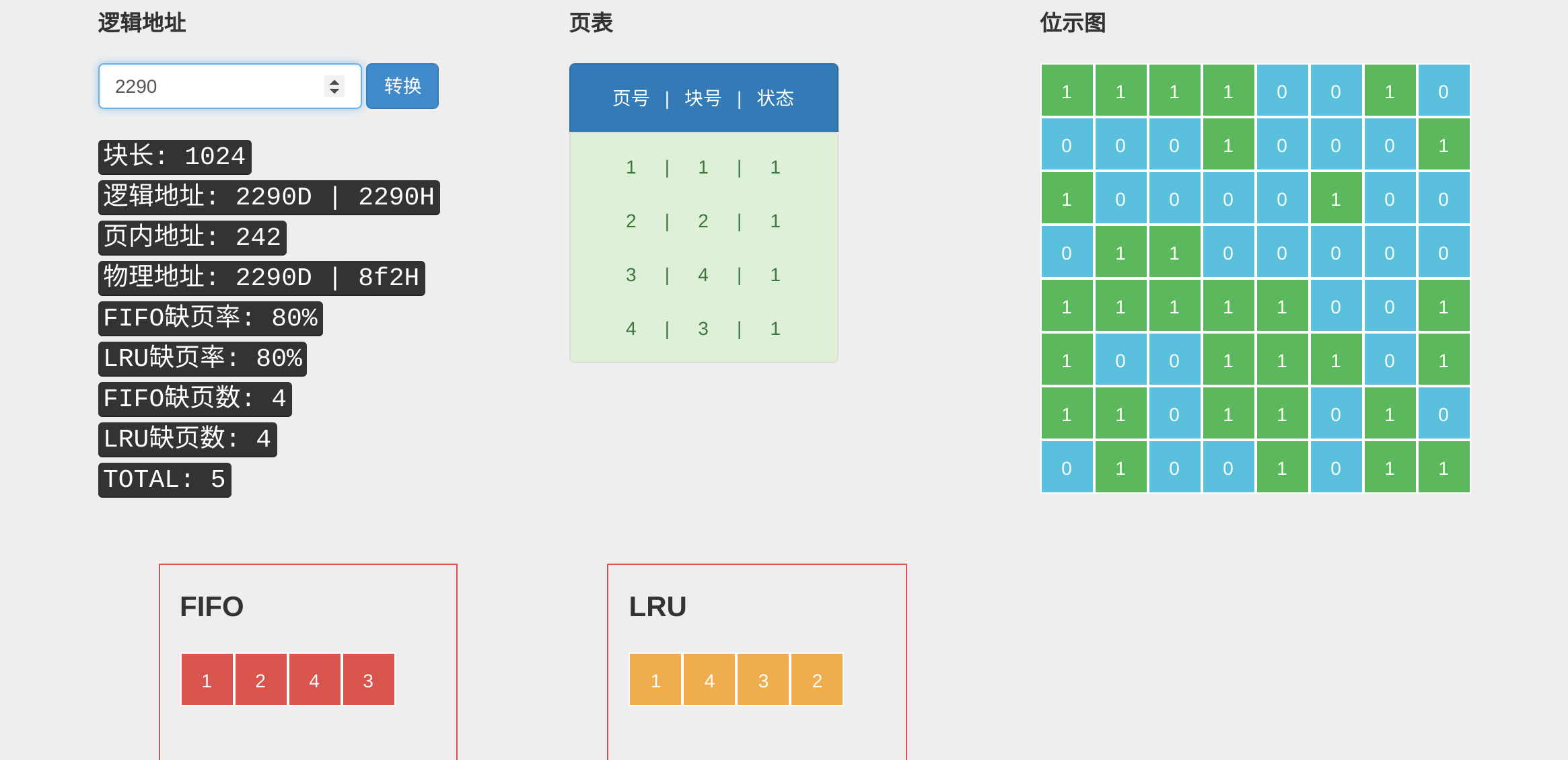
return lab;

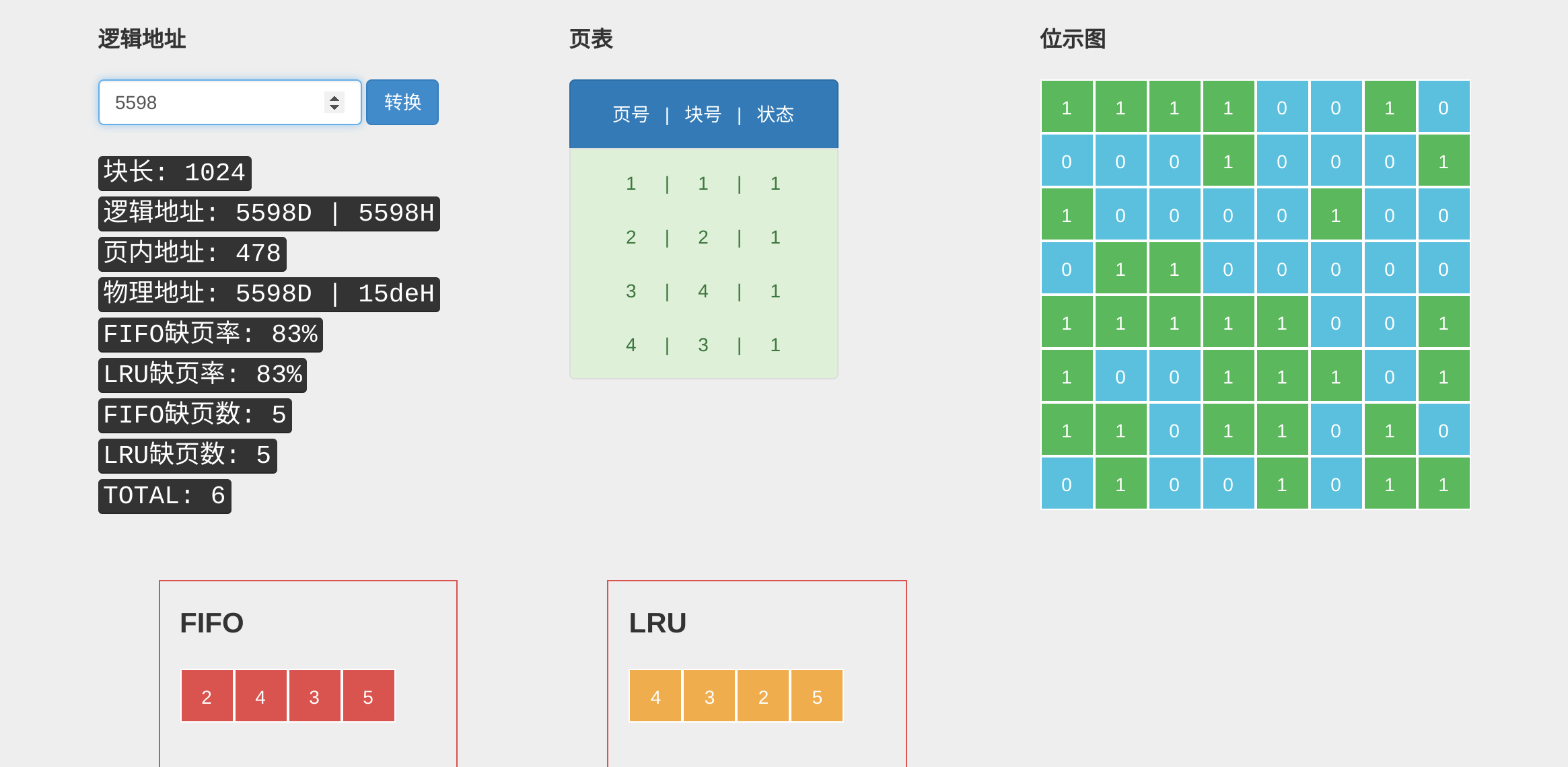
}

**五．实验截图：**









**六．实验心得：**

通过本次实验，了解了分页式存储管理的基本概念和实现方式。了解了逻辑地址和物理地址转换的基本原理。了解了请求式分页式存储管理的基本概念和两中不同的页面置换算法的基本原理和是实现方式。