# 内存管理项目 —请求调页存储管理方式模拟 设计文档

1953608 吴英豪



## 一、算法说明

### 1.思路分析

- 一共有4个内存块，当需要运行某一页的指令时，首先判断指令所在页是否在内存中，若已经在内存中，则直接在对应的内存块中运行即可。
- 当指令所在页不在内存中，判断内存中所有内存块是否已经用完，若有可用内存块，直接将指令所在页面载入内存。
- 若内存块被占满，则采用FIFO算法或LRU算法从内存中调出一个页，并将当前指令所在的页调入内存中。
- 在所有的320条指令中前35条指令与最后35条指令为顺序向后执行，中间的250条指令为随机跳转指令；程序开始时从320条指令中随机选择一条运行；当指令运行总数达到阈值（300）条时，从头开始顺序向后执行所有未被执行到的指令。

### 2.FIFO算法

- 采用队列的数据结构，当内存被占满时，将队首的页面（最先进入队列的页面）调出内存，将新页面加入队尾。

### 3. LRU算法

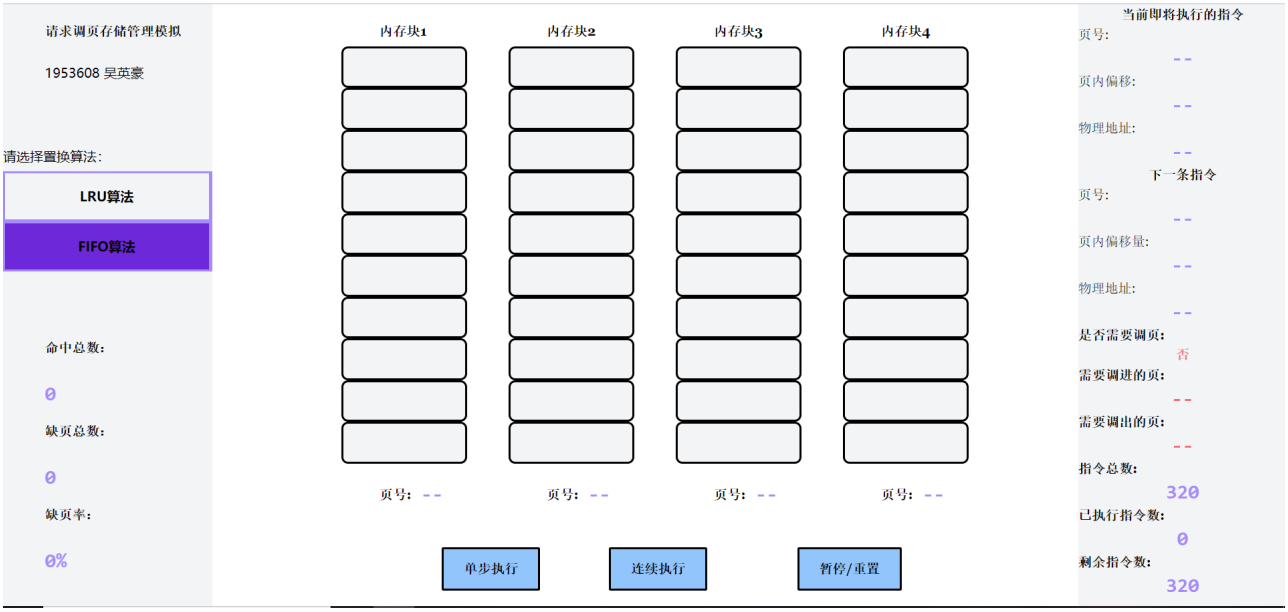
- 采用四个指针 pointerOne、pointerTwo、pointerThree、pointerFour来监管四个内存块，每次调出最久未使用的页面（pointerFour）,并调进新的页面(pointerOne),并根据具体的情况修改其他指针的指向。
- 需要额外考虑的内容有：

  a.当前指令所在页已在内存中时，不需要调入调出页面，但需要修改pointerOne的指向（指向当前指令所在的页），并修改其他指针指向

  b.内存块有剩余时，不需要调出页面，但是需要修改pointerOne的指向，指向最新加入的页面，并修改其他指针指向。

## 二、使用说明

### 1. 界面及使用介绍




- 左边栏将动态显示命中总数、缺页总数以及缺页率，用户可人为选择LRU算法运行还是FIFO算法执行
- 右边栏为信息显示栏，主要的信息有：运行指令数、当前指令以及下一条指令的页号、偏移量、物理地址，是否需要调页等
- 中间为运行按钮，“单步执行”表示运行一步，“连续执行”则会持续运行下去直至结束。在连续运行时按下“暂停/重置”按钮可以暂停运行，当项目暂停时（或单步执行之后）按下“暂停/重置”按钮将会重置所有指令的执行，回到原始情况。


## 2.使用说明

- 本项目为vue-cli项目，可以用命令行打开项目源码文件夹，并输入以下指令运行项目

```
npm install
npm run serve
```

  注意：这种运行方式需要安装***nodejs***、***vue-cli***等环境

- 点击项目运行示例目录下的*index.html*即可查看项目示例

  注意：这种运行方式需要保证运行示例目录下的***favicon.icon***、***index.html***、***js文件夹***、***css文件夹***均在同一个目录下

- **(推荐!!)**此项目已经部署在了github上，可以访问网址https://hollywyh.github.io/Operating-System/ 直接查看项目结果，也可以访问GitHub https://github.com/HOLLYwyh/Operating-System 进行查看

## 三、开发环境

- 开发环境：Windows10
- 开发软件：WebStorm
- 开发语言：Vue、JavaScript、css、html