# 内存管理模拟系统设计文档

## 一、引言

### 1.1 项目目的

本项目旨在通过模拟动态分区分配和请求分页存储管理方式，深入理解内存管理的核心原理与实现过程。具体目标如下：

1. 掌握动态分区分配中的数据结构设计（如空闲分区链）和分配算法（首次适应、最佳适应）。
2. 理解请求分页系统中的页表机制、地址转换过程及页面置换算法（FIFO、LRU）。
3. 通过代码实现提升对内存管理算法的编程实践能力。

### 1.2 项目背景

本设计文档依据内存管理项目PPT的需求，结合 Java 代码实现，详细描述系统的设计与实现方案。该系统分为动态分区分配和请求分页存储管理两个模块，支持算法模拟、状态可视化及结果统计。

## 二、系统设计

### 2.1 整体架构

系统采用**模块化设计**，分为两大核心模块：动态分区分配模块和请求分页存储管理模块，通过图形用户界面（GUI）实现交互。架构图如下：

图示

AI 生成的内容可能不正确。

### 2.2 模块划分

|  |  |  |
| --- | --- | --- |
| 模块名称 | 功能描述 | 核心类 |
| 动态分区分配 | 模拟内存动态分区的分配与回收，支持首次适应和最佳适应算法 | PartitionManager、MemoryBlock、PartitionPanel |
| 请求分页管理 | 模拟请求调页过程，实现 FIFO/LRU 置换算法，生成符合规则的指令序列 | PageReplacementManager、Page、PageReplacementPanel |

## 三、详细设计

### 3.1 动态分区分配模块

#### 3.1.1 数据模型设计

1. **MemoryBlock 类**：表示内存分区
   1. startAddress：起始地址（单位：K）
   2. size：分区大小（单位：K）
   3. allocated：是否已分配
   4. processName：占用分区的进程名（未分配时为 null）
   5. 方法：getEndAddress()计算结束地址，toString()格式化输出分区状态
2. **PartitionManager 类**：管理内存分区
   1. **属性**：memoryBlocks（内存块列表），初始化为 1 个 640K 的空闲块
   2. **核心方法**：
      1. allocateFirstFit()：首次适应算法，遍历空闲块找到首个足够大的分区进行分配或分割。
      2. allocateBestFit()：最佳适应算法，遍历所有空闲块，找到与请求大小最接近的分区进行分配或分割。
      3. deallocate()：释放指定进程的内存分区，并合并相邻空闲块（mergeAdjacentBlocks()方法实现合并逻辑）。
      4. getFragmentation()：计算内存碎片率，公式为1 - 最大空闲块大小 / 总空闲内存。

#### 3.1.2 界面设计（PartitionPanel）

1. **控制面板**：
   1. 输入框：进程名、分配大小、释放进程名。
   2. 下拉框：选择分配算法（首次适应 / 最佳适应）。
   3. 按钮：分配内存、释放内存、重置系统。
2. **内存状态表格**：显示分区的起始地址、结束地址、大小、状态（已分配 / 空闲）及进程名。
3. **操作日志**：记录分配、释放操作结果及系统提示信息。
4. **状态面板**：显示内存碎片率。

### 3.2 请求分页存储管理模块

#### 3.2.1 数据模型设计

1. **Page 类**：表示逻辑页面
   1. pageNumber：页面号（0-31）
   2. inMemory：是否在内存中
   3. physicalFrame：物理块号（-1 表示未装入）
   4. lastAccessTime：最后访问时间（用于 LRU 算法）
   5. loadTime：装入时间（用于 FIFO 算法）
2. **PageReplacementManager 类**：管理页面置换
   1. **属性**：
      1. pages：页表数组（32 页）
      2. physicalMemory：物理内存块（4 块，存储页面号，-1 表示空闲）
      3. pageFaults：缺页次数
      4. currentTime：当前时间戳（用于算法计时）
   2. **核心方法**：
      1. accessInstruction()：模拟访问指令，计算页号，判断是否缺页。若缺页，优先分配空闲块，无空闲块时根据算法置换页面（selectPageFIFO()/selectPageLRU()）。
      2. generateInstructionSequence()：生成符合规则的指令序列：
         1. 50% 顺序执行：当前指令 + 1。
         2. 25% 前地址随机：在 [0, currentPC-1] 范围内随机跳转。
         3. 25% 后地址随机：在 [currentPC+1, 319] 范围内随机跳转。
      3. simulateExecution()：执行指令序列，统计缺页率。

#### 3.2.2 界面设计（PageReplacementPanel）

1. **控制面板**：
   1. 下拉框：选择置换算法（FIFO/LRU）。
   2. 按钮：生成指令序列、开始模拟、单步执行、重置系统。
2. **状态表格**：
   1. 页表：显示页号、是否在内存中、物理块号。
   2. 物理内存：显示内存块号、页号、装入时间、最后访问时间。
3. **执行日志**：记录每条指令的访问结果（命中 / 缺页）、置换操作及统计信息。
4. **进度条**：显示模拟执行进度（总 320 条指令）。

## 四、代码结构

### 4.1 核心类列表

|  |  |  |
| --- | --- | --- |
| 类名 | 职责 | 依赖关系 |
| MemorySimulator | 主程序入口 | MainPanel |
| MainPanel | 主界面容器 | PartitionPanel、PageReplacementPanel |
| MemoryBlock | 内存块数据模型 | - |
| PartitionManager | 动态分区逻辑 | - |
| PartitionPanel | 动态分区界面 | PartitionManager、MemoryBlock |
| Page | 页面数据模型 | - |
| PageReplacementManager | 请求分页逻辑 | Page |
| PageReplacementPanel | 请求分页界面 | PageReplacementManager、Page |

### 4.2 关键交互流程

1. **动态分区分配流程**：

用户输入分配参数 → PartitionPanel触发分配按钮 → 调用PartitionManager的分配算法 → 更新内存块列表 → 刷新表格和日志。

1. **请求分页模拟流程**：

生成指令序列 → PageReplacementManager执行模拟 → 遍历指令调用accessInstruction() → 处理缺页/置换 → 更新页表和物理内存 → 统计缺页率并显示日志。

## 五、测试方案

### 5.1 动态分区模块测试用例

|  |  |  |
| --- | --- | --- |
| 测试场景 | 输入参数 | 预期结果 |
| 首次适应算法分配 | 进程 A，100K | 分配起始地址 0K 的空闲块，剩余 540K 空闲块。 |
| 最佳适应算法分配 | 进程 B，200K | 分配最小足够块（假设剩余 540K），分割为 200K（已分配）和 340K（空闲）。 |
| 释放内存并合并 | 释放进程 A | 合并相邻空闲块，恢复 640K 空闲块。 |
| 分配失败 | 进程 C，1000K | 提示 “没有足够的空闲空间”。 |

图形用户界面, 应用程序

AI 生成的内容可能不正确。

图形用户界面, 应用程序

AI 生成的内容可能不正确。

### 5.2 请求分页模块测试用例

|  |  |  |
| --- | --- | --- |
| 测试场景 | 输入参数 | 预期结果 |
| 指令序列生成 | 无 | 序列中约 50% 连续递增，25%<当前 PC，25%> 当前 PC。 |
| FIFO 置换算法 | 缺页次数统计 | 根据局部性原理，缺页率应低于随机算法。 |
| LRU 置换算法 | 访问历史相关页面 | 优先置换最久未使用的页面。 |

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

图形用户界面, 应用程序

AI 生成的内容可能不正确。

图形用户界面, 应用程序

AI 生成的内容可能不正确。