实验二 请求式分页存储管理算法设计与实现

请求式分页存储管理算法设计与实现

实验的目的是模拟请求页式存储管理系统中的地址转换和缺页中断处理机制，通过使用先进先出（FIFO）页面置换算法来管理内存中的页面。此外，实验还包括考虑页面是否被修改过，以决定是否需要将页面写回磁盘。

实验内容

目标

- 模拟请求页式存储管理中硬件的地址转换过程。
- 处理缺页中断，并用FIFO页面调度算法进行页面置换。
- 考虑页面在执行过程中是否被修改过，从而确定是否需要将页面写回磁盘。

具体任务

- 设计并实现一个地址转换程序。
- 编制一个FIFO页面调度程序。
- 按照给定指令序列调试程序，输出绝对地址或调出、调入的页号。

要求及方法

① 程序输入：页面大小ps; 常驻集大小m；进程大小s；指令集；指令格式如下：

如果该作业依次执行的指令序列如下表所示：

操作	页号	页内地址	操作	页号	页内地址
+	0	070	移位	4	053
+	1	050	+	5	023
×	2	015	存	1	037
存	3	021	取	2	078
取	0	056	+	4	001
−	6	040	存	6	084

依次执行上述指令调试你所设计的程序（仅模拟指令的执行，不考虑序列中具体操作的执行）。

2 根据输入信息，随机初始化页表，页表格式参考：

页表大小由进程大小和页面大小决定；若进程大小为7页，其副本已在磁盘上。若常驻集大小为4，该作业的第0页至第3页已装入内存，其余3页未装入主存，该作业的页表如下：

页号	标志	页架号	修改标志	在磁盘上位置
0	1	5	0	011
1	1	8	0	012
2	1	9	0	013
3	1	1	0	021
4	0		0	022
5	0		0	023
6	0		0	121

3 设计一个地址转换程序来模拟硬件的地址转换和缺页中断。当访问的页在主存时则形成绝对地址，但不去模拟指令的执行，可以输出转换后的绝对地址来表示一条指令已执行完成。当访问的页不在主存中时，则输出“*页号”来表示硬件产生了一次缺页中断。模拟地址转换流程见图1。

4 编制一个FIFO页面调度程序；FIFO页面调度算法总是先调出作业中最先进入主存中的哪一页。因此可以用一个数组来表示（或构成）页号队列。数组中每个元素是该作业已在主存中的页面号，假定分配给作业的页架数为m，且该作业开始的m页已装入主存，则数组可由m个元素构成。

P[0]，P[1]，P[2]，…，P[m−1]

它们的初值为P[0]：=0，P[1]：=1，P[2]：=2，…，P[m−1]：=m−1

用一指针K指示当要调入新页时应调出的页在数组中的位置，K的初值为“0”，当产生缺页中断后，操作系统总是选择P[K]所指出的页面调出，然后执行：

P[K]：=要装入的新页页号

K：=(k+1)mod m

在实验中不必实际地启动磁盘执行调出一页和装入一页的工作，而用输出“OUT调出的页号”和“IN要装入的新页页号”来模拟一次调出和装入过程，模拟程序的流程图见附图1。

按流程控制过程如下：

提示：输入指令的页号和页内偏移和是否存指令，若d为−1则结束，否则进入流程控制过程，得P^1和d，查表在主存时，绝对地址=P^1×1024+d

算法流程

1. **初始化**
   - 初始化页表，设置每一页的标志位、页架号、修改标志以及在磁盘上的位置。
   - 设置常驻集大小 `m` （即内存中可以容纳的最大页面数），并根据这个值初始化FIFO队列。
2. **指令处理**
   - 对于给定的每个指令，检查要访问的页是否已经在内存中：
     - 如果该页在内存中，则进行地址转换，并输出绝对地址。
     - 如果该页不在内存中，则触发缺页中断，输出“*页号”，并进入页面置换逻辑。
3. **页面置换逻辑**
   - 使用FIFO算法选择最老的页面从内存中移除。
   - 如果被替换的页面被标记为已修改，则模拟将其写回磁盘的操作（打印“OUT 调出的页号”）。
   - 将新页面装入内存，并更新页表和FIFO队列（打印“IN 要装入的新页页号”）。

4. **结束**
    - 所有指令执行完毕后，实验结束。

数据结构及说明

- **页表**：记录每个页的状态，包括：
    - **页号(Page ID)**：页面的唯一标识符。
    - **标志(Present Flag)**：指示页面是否在内存中（1表示在内存，0表示不在）。
    - **页架号(Frame ID)**：如果页面在内存中，此字段给出它所在的物理页架号；否则为空。
    - **修改标志(Dirty Bit)**：指示页面是否被修改过（1表示修改过，0表示未修改）。
    - **磁盘位置(Disk Location)**：页面不在内存时，在磁盘上的位置。
- **FIFO队列**：用于记录当前在内存中的页面顺序。通过数组 `P[]` 实现，其中每个元素对应一个页面号。同时维护一个指针 `K`，指向下一个将被替换的页面的位置。

程序清单及描述

1. 程序功能说明

- **页面调度模拟**：通过C语言实现了一个简单的页面调度模拟器，使用FIFO算法来管理内存中的页面。
- **指令执行**：定义了一组指令，每条指令对应一个操作（如读取、写入），并且指定了该操作涉及的页号和页内地址。
- **页表管理**：实现了页表的初始化、查看和更新功能，以反映页面状态的变化。
- **缺页处理**：当需要访问的页面不在内存中时，会触发缺页中断，程序按照FIFO策略选择一个旧页进行替换。

2. 代码结构解析

- **数据结构**：
    - `struct instruct`：用于表示指令的操作符、页号及页内地址。
    - `struct page`：表示页表项，包含页号、标志位、页架号、修改标志和磁盘位置等信息。
- **全局变量**：
    - 定义了指令数组 `instruct[]` 和页表数组 `page[]`，以及页架号数组 `p[]`。
- **函数列表**：

- `init()`：初始化页表和指令集。
- `print1()` 和 `print2()`：打印页表信息和指令信息。
- `transform()`：根据指令集执行相应的操作，并处理任何发生的缺页中断。
- `main()`：程序入口点，调用以上函数完成整个模拟过程。