实验四 请求调页存储管理方式的的模拟

实验目的

通过对页面、页表、地址转换和页面转换过程的模拟，加深对请求调页系统的原理和实现过程的理解。

实验内容

（1） 用C语言模拟一作业的执行过程，该作业共有320条指令。假设每个页面中可以存放10条指令，即该作业的逻辑地址空间为32页。分配给这个作业的内存块数为4个，目前它的所有页都还没有调入内存。

（2） 模拟如下请求分页存储管理方式：

● 如果所访问的指令已在内存中，则显示其物理地址，并转下一条指令。

● 如果所访问的指令还未装入内存，则发生缺页，此时须记录缺页的次数，并将相应页调入内存；如果调页时所有内存均已装入，则需要按照置换算法进行页面置换。最后显示其物理地址，并转下一条指令。

（3） 作业中指令的执行次序按如下原则随机生成：

● 50%的概率顺序执行下一条指令。

● 25%的概率跳转执行前面的指令。

● 25%的概率跳转执行后面的指令。

具体实现办法：

1） 在[0, 319] 间随机选取一条指令开始启动执行，设其序号为 m；

2） 顺序执行下一条指令，即序号为 m + 1 的指令；

3） 通过随机数跳转到[0, m]中的某条指令执行，设其序号为 m1；

4） 顺序执行下一条指令，即序号为 m1 + 1 的指令；

5） 通过随机数跳转到[m1 + 2, 319]中的某条指令处，设其序号为 m2；

6） 顺序执行下一条指令，即序号为 m2 + 1 的指令；

7） 重复执行 2）-6）步，直到执行完 320 条指令为止。

（4） 编写程序分别使用 OPT、FIFO、LRU 置换算法模拟上述流程，计算缺页率。

思考

（1） 如果增加分配给作业的内存块数，将会对作业运行过程中的缺页率产生什么影响？

（2） OPT、FIFO、LRU 哪种置换算法的性能好？为什么一般情况下 LRU 具有比 FIFO 更好的性能？