第八次讨论记录

|  |  |
| --- | --- |
| 主题 | 页表及缺页中断设计 |
| 参与人员 | 计科191：郭晗  计科192：刘嘉仪、秦婧雯、郭清如 |
| 讨论时间 | 2022-03-18 |
| 讨论地点 | QQ共享屏幕 |
| 讨论时长 | 1小时 |

## 主要内容

1. 讲述基本概念以及算法的思路

1、什么是缺页中断？

进程线性地址空间里的页面不必常驻内存，在执行一条指令时，如果发现他要访问的页没有在内存中（即存在位为0），那么停止该指令的执行，并产生一个页不存在的异常，对应的故障处理程序可通过从外存加载该页的方法来排除故障，之后，原先引起的异常的指令就可以继续执行，而不再产生异常。

缺页中断与虚页调度算法是当进程访问页面不在内存时，系统发生缺页中断现象，并通过LRU页面置换算法替换相应的页面。

2、页面调度算法

将新页面调入内存时，如果内存中所有的物理页都已经分配出去，就按照某种策略（本课设使用LRU算法）来废弃整个页面，将其所占据的物理页释放出来。

3、页表生成

生成页表并将页表变化过程写入文件。

4、介绍页表项Page的结构

每一个[进程](https://baike.baidu.com/item/%E8%BF%9B%E7%A8%8B/382503)都拥有一个自己的页表。页表结构如下：



public class Page {

//页表项，共3+6+3+11=23位，需要16+16位，即4B

public int Pagenum;//页号，设置进程最大8页，3位

public int Blocknum;//物理块号，64，6位

public int State;//状态位，用于指示是否调入内存,0,未调入，1已调入，1位

public int Visit;//访问字段，用于记录本页最近已有多长时间未被访问，1位

public int Modify;//修改位，标识该页在调入内存后是否被修改，1位

public int Address[]=new int[2];//外存地址，用于指出该页在外存上的地址，供调入该页时参考，5+6=11位

二、整体思路

虚存管理部分可以设计为分配给页表为3个页框，MMU位于CPU内部，可以假想为一个进程的所需要的资源都放在虚拟地址空间里面，而CPU在取指令时，机器指令中的地址码部分为虚拟地址（线性地址），需要经过MMU转换成为内存地址，才能进行取指令，每当取指令时，先把指令的地址转化为物理地址，然后判断是否缺页中断，缺页中断这一部分可以自写一个类，函数作为CPU类内的方法。

## 讨论心得

虚存管理部分的页表设计可以参考课本上的页表项结构，设计应当有依据，不能只依靠自己的感觉。编写代码首先应当理顺自己的思路，才能顺利地仿真实现操作系统。这一部分的思路就是进程调度过程中执行指令首先要取指令，而取指令则涉及到了MMU和虚页内存管理的这一部分，需要通过逻辑地址得到物理地址，然后依次查看快表、页表、对换区与外存，并更新快表与页表等。