# OS_Assignment11

| | |
|---|---|
| Assign | |
| tag | homework |
| 姓名 | 周鹏宇 |
| 学号 | 2019K8009929039 |

1. 一个内存空间分配器，采用伙伴算法。假设物理内存总共 64KB，

   1) 请给出第一级的一对伙伴块的起始地址

   2) 请给出第二级的二对伙伴块的起始地址

   3) 地址 0xa700，已知它位于第 7 级伙伴块中，请问该块的伙伴块的起始地址

   a. 64KB即$2^{16}$B，则第一级的一对伙伴块大小为$2^{16}/2 = 2^{15}$B，起始地址为0x0000和0x8000

   b. 第二级伙伴块大小为$\frac{2^{16}}{2^2} = 2^{14}$B，起始地址为0x0000，0x4000，0x8000，0xc000

   c. 第七级伙伴块大小为$\frac{2^{16}}{2^7} = 2^9$B,起始地址为后9位为0的数个地址，对于0xa700，只须将其末9位抹0即可，即0xa600

2. If FIFO page replacement is used with four page frames and eight pages, how many page faults will occur with the reference string 0172327103 if the four frames are initially empty? Now repeat this problem for LRU.

   - 无论是那种算法，前四次一定会发生page fault
   - 对于FIFO，在第五次访问page 3时会发生page fault，并替换掉0，之后2，7，1都正常，再访问0时会替换掉1，发生一次page fault，一共有6次page fault
   - 对于LRU，访问page3时会替换走0，之后访问page 0时会替换掉使用次数最少的page3，访问3时会替换掉近期访问次数最少的page 2，故共有7次page fault

3. Suppose that the virtual page reference stream contains repetitions of long sequences of page references followed occasionally by a random page reference. For example, the sequence: 0, 1, ... , 511, 431, 0, 1, ... , 511, 332, 0, 1, ... consists of repetitions of the sequence 0, 1, ... , 511 followed by a random reference to pages 431 and 332.

(a) Why will the standard replacement algorithms (LRU, FIFO, clock) not be effective in handling this workload for a page allocation that is less than the sequence length?
(b) If this program were allocated 500 page frames, describe a page replacement approach that would perform much better than the LRU, FIFO, or clock algorithms.

a. 对于clock和LRU算法而言，在进行一轮不包括随机数的循环时，每个页表最多被使用了一次，直接导致这两个算法等效于FIFO（假设clock在扫完一轮且未找到最优替换页后的策略是FIFO），那么每次插入页表都会导致一次page fault，若页框够大，则在每次读取一个随机页表后的下一次遍历会少一次page fault，但实际上出现的fault还是很多。

b. 单纯针对此类情况，可以用FILO算法，也即先入后出，当第一次遍历前五百个页表时，发生五百次page fault，但之后的12的页表会替换掉最后进入的488-499页表，而后在下一轮遍历和访问随机数页表时极大概率不会发生替换，直到再次遇到488-499的页表，粗略估计此后每次循环的page fault不会超过25（24+1），性能显著增加。

4. A computer has four page frames. The time of loading, time of last access, and the *R* and *M* bits for each page are as shown below (the times are in clock ticks):

| Page | Loaded | Last ref. | R | M |
|---|---|---|---|---|
| 0 | 126 | 280 | 1 | 0 |
| 1 | 230 | 265 | 0 | 1 |
| 2 | 140 | 270 | 0 | 0 |
| 3 | 110 | 285 | 1 | 1 |

(a) Which page will NRU replace?
(b) Which page will FIFO replace?
(c) Which page will LRU replace?
(d) Which page will second chance replace?

a. NRU算法会取R+M最小的页替换，故换掉page2

b. FIFO算法会取最早进入的替换，故换掉page3

c. LRU算法会取最近最少使用的替换，故换掉page1

d. 二次机会算法会选R=0中最早进入的换掉，故换掉page2

Consider the following two-dimensional array:

```
int  X[64][64];
```

Suppose that a system has four page frames and each frame is 128 words (an integer occupies one word). Programs that manipulate the $X$ array fit into exactly one page and always occupy page 0. The data are swapped in and out of the other three frames. The $X$ array is stored in row-major order (i.e., $X[0][1]$ follows $X[0][0]$ in memory). Which of the two code fragments shown below will generate the lowest number of page faults? Explain and compute the total number of page faults.

*Fragment A*

```
for (int j = 0; j < 64; j++)
   for (int i = 0; i < 64; i++) X[i][j] = 0;
```

*Fragment B*

```
for (int i = 0; i < 64; i++)
   for (int j = 0; j < 64; j++) X[i][j] = 0;
```

5.

每页存储128个int，即X数组的两行，不妨记使用FIFO算法，在A中，每两次改变X[i][j]都会发生一次page fault(X[0][0]和X[1][0]在同一页)，而后每次对j的循环都会产生page fault，故会产生$32 \times 64 = 2^{11}$次page falut

而对于B，对于j的循环不会产生page fault（除了第一次换入对应页的时候），而对i的循环每两次会产生一次page fault，故有32次page fault