OS实验5-模拟页表

实验目的:

1. 创建程序，将以页为基础的虚拟地址翻译成物理地址
2. 实现TLB，统计给定地址集上，使用/不使用TLB产生的性能差异
3. 实现支持缺页中断，交换的功能

基本设计:

由于我们的程序设计目标比较简单，我们可以认为我们在操作这样一台虚拟机---有64KB的外存，32KB的内存，支持最大64KB的虚拟内存空间，在运行时只加载一个进程，使用“固定分配，局部置换”的方法，将全部内存按256B粒度分割为256个物理页面(frame/block)，并为进程初始化一个最大长度为256的页表。

当程序刚开始运行，页表和物理页面被创建，但是页和物理页面间的联系还没有建立起来，物理页面的内容页没有意义。这时在虚拟机里有三个数据结构

1. 页构成的虚拟内存

page	size(byte)
0	256
1	256
....	....
255	256

2. 由frame构成的物理内存空间

frame	size(byte)	busy
0	256	false
1	256	false
....	....	false
255	256	false

3. 虚拟机中唯一进程持有的一个页表，现在页表中所有项都不可用

   由于虚拟机中只有一个进程，为了方便可以把本来属于页的属性直接绑定到页表上

page	frame	attribute
0	X	not avaliable
.....	X	not avaliable
255	X	not avaliable

随后，我们运行程序，从address.txt读取到第一个虚拟地址:16916

```
while(fin>>virtualAddr)
{
    pageNum = virtualAddr & 65280;
    pageNum = pageNum>>8;
    offset = virtualAddr & 255;
    visit(pageNum, offset);
}
```

得到pageNum: 66, offset: 20

查找页表，发现对应frame不可用，触发缺页中断，应从外存(BACKING_STORE.bin文件)读取相应frame。

```
frame ReadDisk(int start addr, int size);
newframe = ReadDisk(66*256, 256);
```

接下来检查物理内存空间，我们发现整个物理内存空间都是空闲的，使用最先找到的0号frame。

```
memory[0].context = newframe;
memory[0].busy = busy;
```

然后我们还需要修改页表

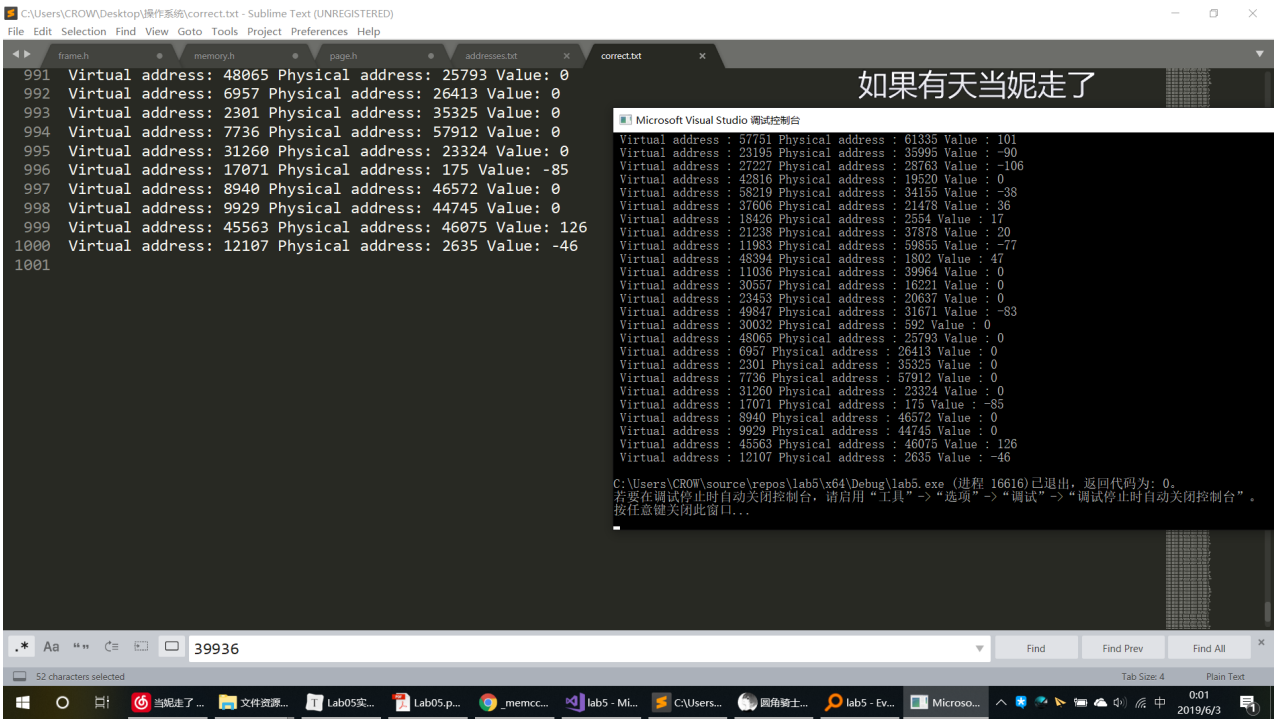
```
pageTable[66].frame = 0;
pageTable[66].attribute = avaliable;
```

然后我们重新进行visit操作，这次在页表中查到66号页表对应0号frame，能获取有效的物理内存，并能读取到数据。

```
physicAddr = 0*256+20 = 20;
```

实验结果

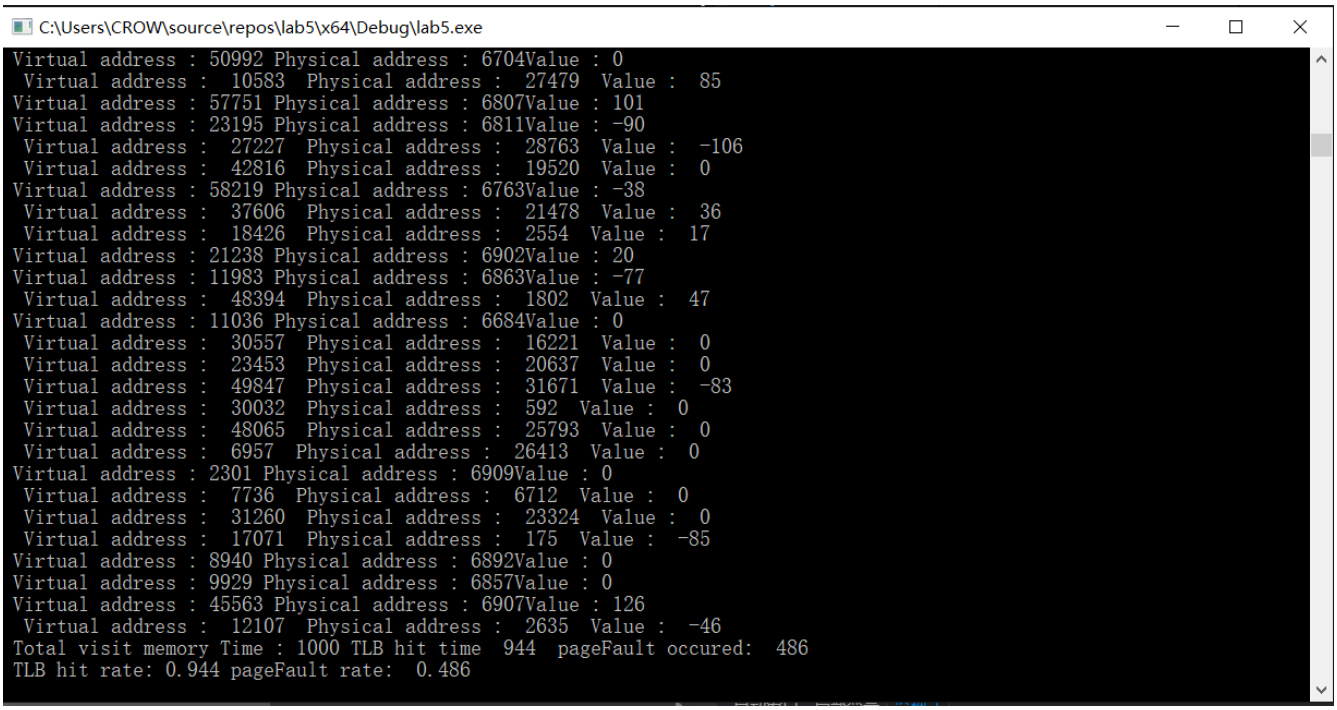
实验目标一(无TLB，物理内存足够，不发生页置换)对应的源代码是"page_adress_translation_1.cpp"文件，其输出与样例"correct.txt"一致。



实验目标二对应"page_adress_translation_2.cpp"。这个版本的代码添加了TLB，并将物理内存缩小到128个frame。在实验二中，页面的置换使用随机数选择，TLB的更新使用FIFO。在输出时可以发现，当读取超过128个页面时，由于页置换，物理内存地址和correct.txt不再相符，但value仍然相符。

统计结果

TLB hit rate	page fault rate
94.4%	48.6%



实验目标3和实验目标源码极度相似，由于实验目标一的代码中已经实现了页的置换(使用随机算法换出)，所以只需要更改物理内存大小和添加少量用于统计运算状态的代码。实验目标三对应的是"page_adress_translation_3.cpp"。

统计结果

page fault rate
53.2%