**Chapter 4 exercises**

16281262 麻锦涛

1. Name two differences between logical and physical addresses.

回答： 地址用于唯一标识CPU内存中某些内容的位置。这些地址分为两种主要类型，第一种是逻辑地址，另一种是物理地址。两者都具有不同但有些相似的功能。逻辑地址用作可由用户程序查看的虚拟地址。然而，物理地址不能由用户程序直接查看，并且逻辑地址被用作借助指针访问物理地址的资源。当正在执行程序时，CPU也生成逻辑地址，而物理地址是在存储器单元内找到的实际位置。当逻辑地址被映射到其对应的物理地址时，它成为CPU和承载存储器的总线之间的存储器管理单元，因为当它到达地址转换层和CPU时执行的任务是相似的。定义所创建的这种层的最佳方式是数据链路层，其充当整个计算机网络的硬件和软件之间的连接器。逻辑地址和物理地址之间的**关键区别**在于[CPU](https://www.differencebetween.com/difference-between-alu-and-cpu/#CPU)**在程序执行期间生成逻辑地址，而物理地址是存储器单元中的位置。**

简单来说，CPU生成逻辑地址或虚拟地址。从正在运行的程序的角度来看，项似乎位于逻辑地址提供的地址。内存单元观察物理地址。此外，它允许通过数据总线访问主存储器中的特定存储器单元。

|  |  |
| --- | --- |
| **逻辑地址与物理地址** | |
| 逻辑地址是项目从执行的应用程序的角度看起来驻留的地址。 | 物理地址是在地址总线电路上以二进制数的形式表示的存储器地址，以便使数据总线能够访问主存储器的*特定*存储单元或存储器映射的I / O设备的寄存器。 |
| **能见度** | |
| 用户可以查看程序的逻辑地址。 | 用户无法查看程序的物理地址。 |
| **生成方法** | |
| CPU生成逻辑地址。 | MMU计算物理地址。 |
| **无障碍** | |
| 用户可以使用逻辑地址来访问物理地址。 | 用户无法直接访问物理地址。 |

1. Consider a system in which a program can be separated into two parts: code and data. The CPU knows whether it wants an instruction (instruction fetch) or data (data fetch or store). Therefore, two base – limit register pairs are provided: one for instructions and one for data. The instruction base –limit register pair is automatically read-only, so programs can be shared among different users. Discuss the advantages and disadvantages of this scheme.（考虑一个系统，其中程序可以分为两部分：代码和数据。 CPU知道它是否需要指令（指令获取）或数据（数据获取或存储）。因此，提供了两个基本限制寄存器对：一个用于指令，一个用于数据。指令库限制寄存器对是自动只读的，因此程序可以在不同用户之间共享。讨论这个方案的优缺点。）

回答：该方案的主要优点是它是一种有效的代码和数据共享机制。例如，只需要将编辑器或编译器的一个副本保存在内存中，并且可以由需要访问编辑器或编译器代码的所有进程共享此代码。另一个优点是保护代码免受错误修改。缺点是代码和数据必须分开，这通常会贯穿在编译器生成的代码中。

1. Why are page sizes always powers of 2?

回答：通过将地址分解为页面和偏移号来实现分页。将地址分成X页位和Y偏移位是最有效的，而不是对地址执行算术来计算页码和偏移量。因为每个位位置表示2的幂，所以在位之间分割地址，产生页面大小为2的幂。

1. Consider a logical address space of 64 pages of 1,024 words each, mapped onto a physical memory of 32 frames.（考虑64页的1,024个字的逻辑地址空间，映射到32块的物理存储器上。）
2. How many bits are there in the logical address?（逻辑地址中有多少位？）
3. How many bits are there in the physical address? （物理地址中有多少位？）

回答：a.逻辑地址：64 \*1024=2 ^5\*2^10=2^15

（逻辑地址的后15位为“页内地址”，又叫“页内偏移量”，或“页内位移”及有效位）

b.物理地址：32 \*1024=2 ^5\*2^10=2^16

所以，逻辑地址和物理地址的有效位分别是15和16.

1. What is the effect of allowing two entries in a page table to point to the same page frame in memory? Explain how this effect could be used to decrease the amount of time needed to copy a large amount of memory from one place to another. What effect would updating some byte on the one page have on the other page?

回答： 通过允许页表中的两个条目指向内存中的相同页面框架，用户可以共享代码和数据。如果代码是可重入的，则可以通过共享使用大型程序（如文本编辑器，编译器和数据库系统）来节省大量内存空间。通过使不同的页表指向相同的存储位置，可以实现复制大量存储器。

但是，共享不繁琐的代码或数据意味着任何有权访问代码的用户都可以修改它，并且这些修改将在其他用户的“副本”中得到完善。

1. Describe a mechanism by which one segment could belong to the address space of two different processes.（描述一个段可以属于两个不同进程的地址空间的机制。）

回答：由于段表是base-limit寄存器的集合，因此当两个不同作业的段表中的条目指向同一物理位置时，可以共享段。两个段表必须具有相同的基本指针，并且共享段号在两个进程中必须相同。

1. Sharing segments among processes without requiring that they have the same segment number is possible in a dynamically linked segmentation system.
2. Define a system that allows static linking and sharing of segments without requiring that the segment numbers be the same.
3. Describe a paging scheme that allows pages to be shared without requiring that the page numbers be the same.

（在动态链接的分段系统中，可以在进程之间共享分段而不要求它们具有相同的分段号。

a.定义一个允许静态链接和共享段的系统，而不需要段号相同。

b.描述允许共享页面而不要求页码相同的分页方案。）

回答： 这两个问题都减少了能够在不知道与地址相关联的段或页码的情况下能够引用其自己的代码及其数据的程序。 MULTICS通过将四个寄存器与每个进程相关联来解决此问题。一个寄存器具有当前程序段的地址，另一个具有堆栈的基址，另一个具有全局数据的基址，依此类推。我的想法是所有引用都必须通过映射到当前段或页码的寄存器间接进行。通过更改这些寄存器，相同的代码可以针对不同的进程执行，而无需相同的页面或段号。解释内部和外部碎片之间的区别。

1. Given six memory partitions of 300 KB , 600 KB , 350 KB , 200 KB , 750 KB , and 125 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB , 500 KB , 358 KB , 200 KB , and 375 KB (in order)? Rank the algorithms in terms of how efficiently they use memory.（鉴于六个内存分区分别为300 KB，600 KB，350 KB，200 KB，750 KB和125 KB（按顺序），首次匹配，最佳适应和最差适应算法如何放置大小为115 KB的进程，500 KB，358 KB，200 KB和375 KB（按顺序）？根据算法使用内存的效率对算法进行排名。）

回答：首次适应是每次从头开始找，直到找到第一个比当前要放置的内存大小要大的内存空间时，放置该内存。

最佳适应是每次遍历内存空间一次，找大于当前要放置的内存块大小要大的中间的最小者，放置该内存。

最差适应则相反，是取大于当前内存大小中的最大者。

下面给出三种存储分配方式的最终分配结果：

**首次适应：**

对于115kB的进程，选择第一个大于它大小的内存空间，为300kB，并分配给它相应大小的空间，该部分剩余大小300-115=185KB

对于500kB的进程，选择第一个大于它的大小的内存空间，为600kB，分配给它相应大小的空间，该部分剩余大小为600-500=100KB

对于358kB的进程，选择第一个大于它大小的内存空间，为750kB，分配给它相应大小的空间，该部分剩余大小750-358=392KB

对于200kB的进程，选择第一个大于它大小的内存空间，为350kB，分配给它相应大小的空间，该部分剩余大小350-200=150KB

对于375kB的进程，选择第一个大于它大小的内存空间，为392kB，分配给它相应大小的空间，该部分剩余大小392-375=17KB。

内存利用率为 （115+500+358+200+375）/（300+600+350+200+750+125） \*100%=66.6%

**最佳适应：**

对于115kB的进程，选择大于它大小的内存空间中的最小者的内存空间，为125kB，并分配给它相应大小的空间，该部分剩余大小125-115=10KB

对于500kB的进程，选择大于它大小的内存空间中的最小者的内存空间，为600kB，分配给它相应大小的空间，该部分剩余大小为600-500=100KB

对于358kB的进程，选择大于它大小的内存空间中的最小者的内存空间，为750kB，分配给它相应大小的空间，该部分剩余大小750-358=392KB

对于200kB的进程，选择大于它大小的内存空间中的最小者的内存空间，为300kB，分配给它相应大小的空间，该部分剩余大小300-200=100KB

对于375kB的进程，选择大于它大小的内存空间中的最小者的内存空间，为392kB，分配给它相应大小的空间，该部分剩余大小392-375=17KB。

内存利用率为 （115+500+358+200+375）/（300+600+350+200+750+125） \*100%=66.6%

**最差适应：**

对于115kB的进程，选择大于它大小的内存空间中的最大者的内存空间，为750kB，并分配给它相应大小的空间，该部分剩余大小750-115=635KB

对于500kB的进程，选择大于它大小的内存空间中的最大者的内存空间，为635kB，分配给它相应大小的空间，该部分剩余大小为635-500=135KB

对于358kB的进程，选择大于它大小的内存空间中的最大者的内存空间，为600kB，分配给它相应大小的空间，该部分剩余大小600-358=242KB

对于200kB的进程，选择大于它大小的内存空间中的最大者的内存空间，为300kB，分配给它相应大小的空间，该部分剩余大小300-200=100KB

对于375kB的进程，找不到比他大的内存空间，无法分配，只能等待其他进程释放空间才能为它分配空间。

内存利用率为（115+500+112+200）/（300+600+350+200+750+125） \*100%=46.2%



综上：内存利用率最高的是首次适应和最佳适应算法。

1. Most systems allow a program to allocate more memory to its address space during execution. Allocation of data in the heap segments of programs is an example of such allocated memory. What is required to support dynamic memory allocation in the following schemes?

a. Contiguous memory allocation

1. Pure segmentation
2. Pure paging

回答：a. 连续内存分配：可能需要重定位整个程序，因为没有足够的空间让程序增加其分配的内存空间。

b.纯分段：可能还需要重新定位需要扩展的段，因为没有足够的空间让段增加其分配的内存空间。

c.纯分页：在此方案中可以增量分配新页面，而无需重新定位程序的地址空间。

11. Compare the memory organization schemes of contiguous memory allocation, pure segmentation, and pure paging with respect to the following issues:

a. External fragmentation

1. Internal fragmentation
2. Ability to share code across processes

回答：连续的内存分配方案受到外部碎片的影响，因为地址空间是连续分配的，并且随着旧进程的死亡和新进程的启动而产生漏洞。它也不允许进程共享代码，因为进程的虚拟内存段不会分解为不连续的细粒度段。纯粹的细分也会受到外部碎片的影响，因为一个过程的一部分在物理内存中连续布局，并且当死过程的一部分被新过程的一部分取代时会发生碎片化。但是，分段使进程能够共享代码;例如，两个不同的进程可以共享代码段但具有不同的数据段。纯分页不会受到外部碎片的影响，而是会受到内部碎片的影响。进程以页面粒度分配，如果页面未被完全利用，则会导致内部碎片和相应的空间浪费。分页还使进程能够以页面粒度共享代码。

1. On a system with paging, a process cannot access memory that it does not own. Why? How could the operating system allow access to other memory? Why should it or should it not?( 在具有分页的系统上，进程无法访问它不拥有的内存。为什么？操作系统如何允许访问其他内存？为什么要或不应该？)

回答：寻呼系统上的地址是逻辑页号和偏移量。通过基于逻辑页码搜索表来找到物理页面以产生物理页码。由于操作系统控制此表的内容，因此可以将进程限制为仅访问分配给该进程的那些物理页。进程无法引用它不拥有的页面，因为页面不在页面表中。为了允许这种访问，操作系统只需要允许将非进程内存的条目添加到进程的页表中。当两个或多个进程需要交换数据时，这很有用 - 它们只是读取和写入相同的物理地址（可能位于不同的逻辑地址）。这使得非常有效的进程间通信成为可能。

1. Compare paging with segmentation with respect to how much memory the address translation structures require to convert virtual addresses to physical addresses.（将分页与分段进行比较，关于地址转换结构将虚拟地址转换为物理地址所需的内存量。）
2. 回答：物理地址=块号+页内地址

         逻辑地址=页号+页内地址

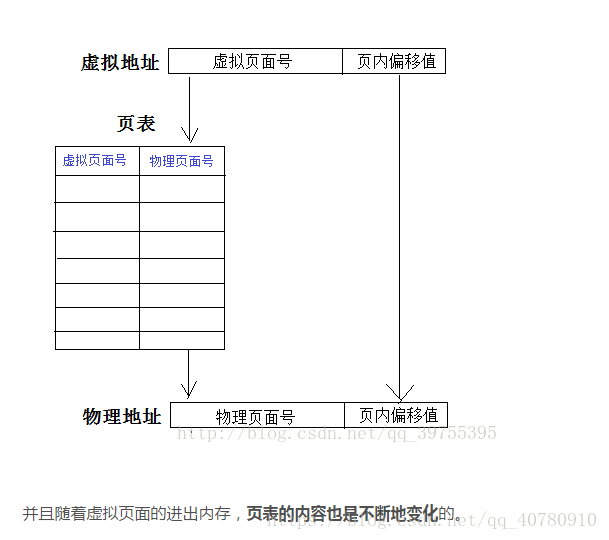
所以物理地址和逻辑地址后面部分是相同的，即都为页内地址。

（1）分页内存管理机制将虚拟内存和物理内存都分成大小一样大的部分，我们称为页，然后按页进行内存分配，分页的最大作用就在于：使得进程的物理地址空间可以是非连续的。

一般页的大小有4KB、8KB、16KB。在该管理机制下内存分配单位化而且不需要空闲连续在一起即可使用。理解了上面几点知识就可以尝试理解如下图表：（了解页表是什么即可。）

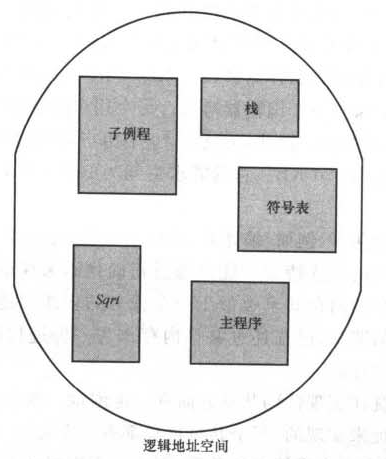
虚拟地址的页号表示和物理地址页号表示是从0开始的。

（比如虚拟地址的第1页前面还有第0页，这个对理解计算非常重要）。



通过页表我们就可以发现，每个虚拟页面号都对应着一个物理页面号，比如虚拟页面号是1，查询该表，我们就可以知晓虚拟页面号1对应的物理页面号在哪里。之所以有这个表存在，就是因为我们使用的物理内存是不连续的

（2）采用分页内存管理有一个不可避免的问题：用户视角的内存和实际内存的分离。设想一段main函数代码，里面包含Sqrt函数的调用。按照编写者的理解，这段代码运行时，操作系统应该分配内存给：符号表(编译时使用)，栈(存放局部变量与函数参数值)，Sqrt代码段，主函数代码段等。这样，编写者就可以方便地指出："函数sqrt内存模块的第五条指令"，来定位一个元素。而实际上，由于采用Paging的管理方式，所有的一切都只是散落在物理内存中的各个帧上，并不是以编写者的理解来划分模块。



Segmentation的内存管理方式可以支持这种思路。逻辑地址空间由一组段组成。每个段都有名字和长度。地址指定了段名称和段内偏移。因此用户通过两个量来指定地址：段名称和偏移。段是编号的，通过段号而非段名称来引用。因此逻辑地址由有序对构成：

 <segment-number,offset>(<段号s, 段内偏移d>)

段偏移d因该在0和段界限之间，如果合法，那么就与基地址相加而得到所需字节在物理内存中的地址。因此段表是一组基地址和界限寄存器对。

1. Assuming a 1- KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers): （假设页面大小为1 KB，以下地址引用的页号和页内偏移是什么（以十进制法解决）

a. 3085

1. 42095
2. 215201
3. 650000
4. 2000001

回答：此处页号为a/b/c/d/e除以1kb（1024）的商，而页内偏移为余数，即：

a. page = 3; offset = 13

b. page = 41; offset = 111

c. page = 210; offset = 161

d. page = 634; offset = 784

e. page = 1953; offset = 129

第二种解题方法 .（用二进制法解决）

l.转换逻辑地址：十进制→二进制

2. 将二进制地址拆分为2个部分（页号，页内偏移）

3. 转换页号和页内偏移 : 十进制→二进制



15. The BTV operating system has a 21-bit virtual address, yet on certain embedded devices, it has only a 16-bit physical address. It also has a 2- KB page size. How many entries are there in each of the following?（BTV操作系统具有21位虚拟地址，但在某些嵌入式设备上，它只有一个16位的物理地址。它还有2 KB的页面大小。以下各项中有多少页表项数目？）

1. A conventional, single-level page table（传统的单级页表）
2. An inverted page table（倒排页表）

回答：a.首先， 21位虚拟地址可表示的进程大小应该是2^21B，页面的数量=进程的大小/页面的大小=逻辑地址的大小/页面的大小=2^21/2KB=2^20，因此页表项数目为2^20

b. 首先， 16位物理地址可表示的进程大小应该是2^16B，对于倒排序表（反置页表）而言，页面的数量=物理地址的大小/页面的大小=2^16/2KB=2^15，因此页表项数目为2^15

16. Consider a logical address space of 256 pages with a 4-KB page size, mapped onto a physical memory of 64 frames.（考虑256页的逻辑地址空间，页面大小为4 KB，映射到64帧的物理内存。）

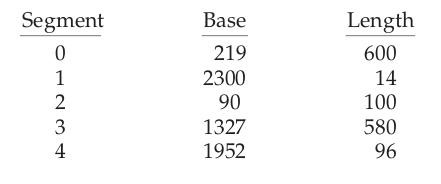
1. How many bits are required in the logical address?
2. How many bits are required in the physical address?

回答：逻辑地址=页号+页面；物理地址=页框号×页块大小

a. 12 + 8 = 20 bits.

b. 12 + 6 = 18 bits.

17. Consider the following segment table:



What are the physical addresses for the following logical addresses?

a. 0,430

1. 1,10
2. 2,500
3. 3,400
4. 4,112

回答：A 0，430 先查询段号为0的基地址为219，然后判断是否在段内，由于430<600，所以在段内，对应物理地址为219+430=649

B 1，10，首先查找段号为1的基地址2300，然后判断是否在段内，由于10<14,所以在段内，对应物理地址为2300+10=2310

C 2,500,首先查找段号为2的基地址90，然后判断是否在段内，由于500>100，所以不在段内，访问非法

D 3，400，首先查找段号为3的基地址1327，然后判断是否在段内，由于400<580,所以在段内，对应物理地址为1327+400=1727

E 4,112，首先查找段号为4的基地址1952，然后判断是否在段内，由于112>96，所以不在段内，访问非法

1. What is the purpose of paging the page tables?

回答：在某些情况下，页表可能变得足够大，以至于通过分页页表，可以简化内存分配问题，并且还可以交换当前未使用的页表部分。备注：可以参考参阅书第8. 5节开始部分中的示例（使用32位逻辑地址集的系统）

1. Under what circumstances do page faults occur? Describe the actions taken by the operating system when a page fault occurs.( 在什么情况下会出现页面错误？描述发生页面错误时操作系统所采取的操作。)

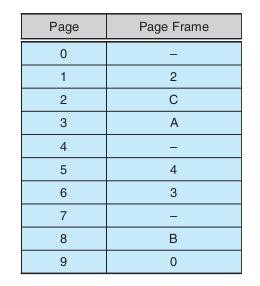
回答：当访问未进入主存储器的页面时，会发生页面错误。操作系统验证内存访问，如果程序无效则中止程序。如果它是有效的，则定位一个空闲帧，并请求I / O将所需页面读入空闲帧。完成I / O后，将更新进程表和页表，并重新启动指令。

1. Assume that you have a page-reference string for a process with m frames (initially all empty). The page-reference string has length p, and n distinct page numbers occur in it. Answer these questions for any page-replacement algorithms:（假设您有一个包含m帧（最初全部为空）的进程的页面引用字符串。页面引用字符串的长度为p，并且其中出现n个不同的页码。回答任何页面替换算法的这些问题：）
2. What is a lower bound on the number of page faults?（页面错误数量的下限是多少？）
3. What is an upper bound on the number of page faults?（页面错误数量的上限是多少？）

回答：a.页面错误数量的下限是n，

b.页面错误数量的上限是p

21. Consider the page table shown in the following figure for a system with 12-bit virtual and physical addresses and with 256-byte pages. The list of free page frames is D, E, F (that is, D is at the head of the list, E is second, and F is last).（对于具有12位虚拟和物理地址以及256字节页面的系统，请考虑下图所示的页表。可用页面框的列表是D，E，F（即，D位于列表的头部，E是第二个，F是最后一个））



Convert the following virtual addresses to their equivalent physical addresses in hexadecimal. All numbers are given in hexadecimal. (A dash for a page frame indicates that the page is not in memory.)（将以下虚拟地址转换为十六进制的等效物理地址。所有数字均以十六进制表示。 （页面框的短划线表示页面不在内存中。））

* 9EF
* 111
* 700
* 0FF

回答：

• 9EF - 0EF

• 111 - 211

• 700 - D00

• 0F F - EFF

22. Consider the following page-replacement algorithms. Rank these algorithms on a five-point scale from “bad” to “perfect” according to their page-fault rate. Separate those algorithms that suffer from Belady’s anomaly from those that do not. （请考虑以下页面替换算法。根据页面错误率，将这些算法按照从“坏”到“完美”的五个等级进行排序。将那些遭受Belady异常困扰的算法与那些没有Belady异常的算法分开。）

a. LRU 置换（[最久最近未使用算法](http://www.baidu.com/link?url=sXvFjo85UtpIGxvmtQ-WHkhwy-tZNWlKImK4vb3ElDfqFISwEfxPdYbTJLm1mQyPjukrafgUiTvPQBYIqLWlS0HeSw4Txr3CPQDOKfXNdnVDiQf8TIK3pUgiaCw-NYBfD0MyQflBCqrhsLEuhENXVDZ-qJ1L963aEGFAE0j35OoEXrXAY5IW637ZmuAdJG55TD5GCE5iSoHHTjqivyJwe_)）

1. FIFO 置换（先进先出页面置换算法）
2. Optimal（最佳置换算法）
3. Second-chance 置换（二次机会置换）

回答：

算法排序如下：

1 Optimal （最佳页面置换算法）no

2 LRU （[最久最近未使用页面置换算法](http://www.baidu.com/link?url=sXvFjo85UtpIGxvmtQ-WHkhwy-tZNWlKImK4vb3ElDfqFISwEfxPdYbTJLm1mQyPjukrafgUiTvPQBYIqLWlS0HeSw4Txr3CPQDOKfXNdnVDiQf8TIK3pUgiaCw-NYBfD0MyQflBCqrhsLEuhENXVDZ-qJ1L963aEGFAE0j35OoEXrXAY5IW637ZmuAdJG55TD5GCE5iSoHHTjqivyJwe_)）no

3 Second-chance （二次机会页面置换算法）yes

4 FIFO （先进先出页面置换算法）yes

发生Belady异常的算法有：

d.Second-chance （二次机会页面置换算法）和b.FIFO （先进先出页面置换算法）

没有发生Belady异常的算法有

c.Optimal （最佳页面置换算法）和 a.LRU （[最久最近未使用页面置换算法](http://www.baidu.com/link?url=sXvFjo85UtpIGxvmtQ-WHkhwy-tZNWlKImK4vb3ElDfqFISwEfxPdYbTJLm1mQyPjukrafgUiTvPQBYIqLWlS0HeSw4Txr3CPQDOKfXNdnVDiQf8TIK3pUgiaCw-NYBfD0MyQflBCqrhsLEuhENXVDZ-qJ1L963aEGFAE0j35OoEXrXAY5IW637ZmuAdJG55TD5GCE5iSoHHTjqivyJwe_)）

1. Consider the two-dimensional array A:

int A[][] = new int[100][100];

where A[0][0] is at location 200 in a paged memory system with pages of size 200. A small process that manipulates the matrix resides in page 0 (locations 0 to 199). Thus, every instruction fetch will be from page 0. For three page frames, how many page faults are generated by the following array-initialization loops? Use LRU replacement, and assume that page frame 1 contains the process and the other two are initially empty.( 其中A [0] [0]位于具有大小为200的页面的分页存储器系统中的位置200处。操作矩阵的小过程驻留在页面0（位置0到199）中。因此，每次取指令都来自第0页。对于三个页面帧，下面的数组初始化循环会产生多少页面错误？使用LRU替换，并假设页面框架1包含进程，而其他两个最初为空。)

* 1. for (int j = 0; j < 100; j++) for (int i = 0; i < 100; i++)

A[i][j] = 0;

* 1. for (int i = 0; i < 100; i++) for (int j = 0; j < 100; j++) A[i][j] = 0;

回答：

a. 5,000

b. 50

1. Consider the following page reference string:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, and seven frames? Remember that all frames are initially empty, so your first unique pages will cost one fault each.

* LRU replacement
* FIFO replacement
* Optimal replacement

回答：

帧数 LRU FIFO Optimal

1 20 20 20

2 18 18 15

3 15 16 11

4 10 14 8

5 8 10 7

6 7 10 7

7 7 7 7

25. Consider a demand-paged computer system where the degree of mul-tiprogramming is currently fixed at four. The system was recently measured to determine utilization of the CPU and the paging disk. Three alternative results are shown below. For each case, what is happening? Can the degree of multi-programming be increased to increase the CPU utilization? Is the paging helping?（考虑一个需求分页计算机系统，其中多程序编程目前固定为四。最近测量了该系统以确定CPU和寻呼磁盘的利用率。三种替代结果如下所示。对于每个案例，发生了什么？可以增加多程序的程度以增加CPU利用率吗？分页有帮助吗？）

1. CPU utilization 13 percent; disk utilization 97 percent
2. CPU utilization 87 percent; disk utilization 3 percent
3. CPU utilization 13 percent; disk utilization 3 percent

回答：

a.正在发生颠簸（操作系统抖动）。

b.单独使用CPU的利用率非常高，并且增加了多道程序的运行程度。

c. 增加多道程序的运行程度。

26. A simplified view of thread states is Ready, Running, and Blocked, where a thread is either ready and waiting to be scheduled, is running on the processor, or is blocked (for example, waiting for I/O ). Assuming a thread is in the Running state, answer the following questions, and explain your answer:

1. Will the thread change state if it incurs a page fault? If so, to what state will it change?
2. Will the thread change state if it generates a TLB miss that is resolved in the page table? If so, to what state will it change?
3. Will the thread change state if an address reference is resolved in the page table? If so, to what state will it change?

回答：

•在页面错误时，线程状态设置为阻塞，因为需要I / O操作才能启动新的页面进入内存。

•在TLB失败时，如果在页表中解析了地址，则线程将继续运行。

•如果在页表中解析了地址，则线程将继续运行。

1. A certain computer provides its users with a virtual memory space of 2^32 bytes. The computer has 2^22 bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4,096 bytes. A user process generates the virtual address 11123456. Explain how the system establishes the corresponding physical location. Distinguish between software and hardware operations.

回答：二进制形式的虚拟地址是

0001 0001 0001 0010 0011 0100 0101 0110

由于页面大小为212，因此页面大小为220.因此，低位12位“0100 0101 0110”用作页面中的位移，而剩余的20位“0001 0001 0001 0010 0011”用作页表中的位移。

4096 bytes = 212 bytes.

Address = | p: 20bits | d: 12 bits|

对于逻辑地址1112 3456h 来说， 1 1123h 为页号， 456h为页内地址偏移量

1. Assume that you are monitoring the rate at which the pointer in the clock algorithm moves. (The pointer indicates the candidate page for replacement.) What can you say about the system if you notice the following behavior:

a. Pointer is moving fast.

b. Pointer is moving slow.

回答：如果指针快速移动，则程序同时访问大量页面。最有可能的是，在清除对应于页面的位的点与再次检查之间的时段期间，再次访问该页面，因此不能替换该页面。这会导致在找到受害者页面之前对页面进行更多扫描。如果指针移动缓慢，则虚拟存储器系统非常有效地找到用于替换的候选页面，显示许多驻留页面未被访问。

29. What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?（颠簸的原因是什么？系统如何检测颠簸？一旦检测到颠簸，系统可以做些什么来消除这个问题？）

回答：颠簸是由于进程所需的最小页数分配不足导致其不断页面错误。与多道程序化水平相比，系统可以通过评估CPU使用率水平来检测颠簸。它可以通过降低多道程序的水平来消除这个问题。