内存管理

一、 选择题

1 D 2 C 3 C 4 B 5 C 6 A 7 D 8 B 9 C 10 C

11 B 12 C 13 B 14 C 15 D 16 B 17 C 18 D 19 A 20 C

21 C 22 C 23 B

二、计算题（选择）

24 -26 C A A
27-29 B D B
30-31 C C
32-33
34-36 B D A
37-39 B D B
41-46 B C C D A B A

三、计算题(填空)

1 .

(1) 00DBFFADH。由 DS=000FH 知，GDT/LDT 位为 1，应查询 LDT，段号为 1，段基址为 00000001H， 段限为 1GB，ESI 不大于段限，故线性地址为 00000001H+00DBFFACH=00DBFFADH。

(2) 越界。由 ES=0008H 知，GDT/LDT 位为 0，应查询 GDT，段号为 1，段限为 4MB，EDI 大于段限， 故越界。

(3) 16K。页面大小为 4KB，页表项 4 字节，故一页有 1024 个页表项，3072 个页，页表需要 3 页，加上 页目录表，共 4 页，16K。

(4)01535180H。高 20 位为页号，即 03FFFH，查表知，页框号为 01535H，有效位为 1，用页框号替换。

(5) 缺页。页号为 02FF3，查表知，有效位为 0，故缺页。

2.

(1) 4KB

(2) 4MB

(3) LA>>22&3FFH

(4) LA>>12&3FFH

注：1(3)(unsigned int)LA>>22 1(4)(unsigned int)LA>>12&0x3FF 也可以

(5) 16KB

(6) 00200020H

(7) 00200024H

(8) 00900H

(9) 00901H

(10) 00901000H_

四、 计算题(简答)

1.

(1) 根据页式管理的工作原理,应先考虑页面大小,以便将页号和页内位移分解出来。页面大小为 4KB,
即 212 ,则得到页内位移占虚地址的低 12 位,页号占剩余高位。可得三个虚地址的页号 P 如下 (十六进制的一位数字转换成 4 位二进制,因此,十六进制的低三位正好为页内位移,最高位为页 号):

2362H:P=2,访问快表 10ns,因初始为空,访问页表 100ns 得到页框号,合成物理地址后访问 主存 100ns,共计 10ns+100ns+100ns=210ns。

1565H:P=1,访问快表 10ns,落空,访问页表 100ns 落空,进行缺页中断处理 108 ns,访问快 表 10ns,合成物理地址后访问主存 100ns,共计 10ns+100ns+108 ns+10ns+100ns=100 000 220ns。

25A5H:P=2,访问快表,因第一次访问已将该页号放入快表,因此花费 10ns 便可合成物理地 址,访问主存 100ns,共计 10ns+100ns=110ns。

(2) 当访问虚地址 1565H 时,产生缺页中断,合法驻留集为 2,必须从页表中淘汰一个页面,根据题 目的置换算法,应淘汰 0 号页面,因此 1565H 的对应页框号为 101H。由此可得 1565H 的物理 地址为 101565H。

2.

1) 应淘汰 Page 3，因为其访问计数器为 00100，为最小值。
2) 老化算法只有有限位的存储，记录最近若干次页面使用情况，更早的使用情况

会丢失。另外，老 化算法没有记录在一个时间周期内页面的使用频度和时间等，所以与 LRU 相比只是一个近似实现。

3.

(1)17CAH -> 000101**1111001010**，页号为 5

(2)LRU，替换 0 页，得到页框号 7，物理地址 000111**1111001010 ->**3FCAH

4.

(1)页面长 **4KB**，说明页内偏移地址占 **12** 位，虚拟地址空间 **64** 位，说明页号总长为 **64-12=52** 位。页面长 **4KB**，页表项 **4B**，故每张页表不超过 **4K/4=1K=2****10** 项，即每级 页表地址长度不应该超过 **10** 位。页号总长 **52** 位/每页表最长 **10** 位**=5.2**，向上取整为 **6**。 即采用六层分页策略。

(2)如采用倒排页表，因为物理地址空间 **4GB**，故倒排页表应该有 **4GB/4KB=1M** 个页 表项，每页表项大小为 **4B**，倒排页表大小为 **1M*4B=4MB**。
系统维护一张倒排页表。
可以使用 **Hash** 散列，解决倒排页表不便于逻辑地址向物理地址转换的问题。

五、简答题

1. 答：分段式内存管理解决了分页式内存管理中划分页时仅根据大小划分，这样可能将无关的 内容分到一页中，此页不便设置权限与保护，也不利于共享。也有可能把密切相关的内容分 到不同页中，当页面置换算法设置不当时，内存紧张时容易形成“抖动”现象。 分段式内存管理带来的问题是段往往过大，多次分配释放后可能形成大量“外碎片”，利 用内存效率不高。

2. 答：Intel IA32 的保护模式下内存管理方法不是段页式内存管理方法。因为段页式内存管理是 分段时段内再分页，整个是一件事。而保护模式是通过分段将逻辑地址转换成线性地址，第二步通过分页将线性地址转成物理地址，这是两件事。

3. LRU 需求记录所有页面长期运行中被使用的时间和次数，需要大量的快速存储空间，而且比较 pthread*kill()：向线程发送一个信号 算法非常复杂，不容易实现。z 同步函数，用于 mutex 和条件变量替代方案可以采用老化（aging）算法，每个页面有一个长度有限的记数器，记录每个 tick 内* *pthread*mutex*init() 初始化互斥锁 面使用情况，记录信息的权重逐次递减。这种算法与 LRU 相比，不能记录每个 tick 中内存使用情况，* *pthread*mutex_destroy() 删除互斥锁 而且记录的位数有限，但是实现方案较易实现。

4. 分页式：逻辑地址空间划分只简单依靠页面大小，缺乏内在逻辑性，导致一方面相关内容被分散 到多页上，页面置换不当时容易造成内存抖动，另一方面不同性质的内容被分到同一页中，使得页面 权限保护设置困难。 分段式：段体积大，在内存中无法不连续存储，易形成内存外碎片，降低内存利用率。 段页式：先分段再分页，以段为单位调入调出，以页为单位在内存中不连续存储，既保证了相关内容 同时进出内存，便于设置权限保护，又可以充分利用内存空间。 段页式结构复杂，实现起来效率低，所以没有被广泛采用。

5. 答：最差适应分配最大空间的分区给进程使用，以期剩余外碎片空间较大，再次利用的可能性较大。固定分区无外碎片，故不应采用这种算法。
交换技术交换的单位是进程，请求式分段技术交换的单位是段。
请求式分段是操作系统进行段调入调出，对程序员透明，而覆盖技术需要程序员自己完成调入调出。

6. 答：页面置换应该在内存空余空间小于一个固定的下限阈值时开始，并在达到另一个上限阈值时停止。 OPT 最理想但不可能实现。 LRU 要求比较最近最少使用的页面，条件多，要存储的数据量大，比较的时间长，很难实现。 Belady 异常指的时当增加页框时缺页中断发生的数量反而升高的现象。 FIFO 存在 Belady 异常。

7. 答：进程设置一个虚拟时钟，执行一个时钟周期就加 1，不执行就不增加。（1 分）每个页面被访问时，记录最后访问的虚拟时间，R 位置 1。R 位定期清除。（1 分）如果 R=1,则保留，将当前时间记录下来。（1 分）如果 R=0 对比当前虚拟时间与页面最后访问时间差 age 与阈值 τ，如 age>τ 则淘汰（1 分）。如 age <= τ，则记录其访问时间，必要时淘汰其中最旧的。（1 分）

8. 答：分段比分页更有逻辑性，将同类的或相关的内容放在一个段内，这样不会由于页面置换算法选择不当而形成“抖动”现象。（1 分）同类内容划分在一个段内，可以实现段的保护，如代码段设置为只读，数据段设置为读写。（1 分）公共代码段可以通过映射共享到多个进程。（1 分）段页式既按照相关性划分段，继承了分段的优势（1 分），又有分页管理可以不连续存储，能够充分利用空间的好处。（1 分）

9. **答：缓存主要用于解决 CPU 和内存之间存在的速度差。一般来说，CPU 中寄存器的速度要远快于内存，将 CPU 要用到的数据预先从内存中读到缓存，这样 CPU 使用时就可以快速得到数据，写回内存的过程也类似。**
TLB 就是使用缓存的一个典型例子。

10. 在 64 位系统中，由于虚拟地址太大，普通页表会非常大，无法存储。另一方面，实际内存 相对较小，所以建立一张从物理地址索引得到相对地址的倒排

页表。 最大的问题的难于从相对地址查找到绝对地址。可以采用 hash 表□高查找效率，并使用 TLB 加速查找。

11. 答：相同点都是为了在内存不足的情况下装入更多的进程，都是会产生外碎片。不同点为交换 技术交换的对象是整个进程而请求式分段交换的进程中的一个段。

12. 答：老化算法与 LRU 相比，主要有两点区别：（1）老化算法记录使用情况的寄存器只有有限位， 比如 8 位，无法记录所有使用情况。（2）同一时间间隔内只使用 0/1 区分页面使用情况，无法详 细区别间隔内的具体时间