ICS Seminar Week11 Prep

康子熙 赵廷昊 余文凯 许珈铭

2023.12.3

Rules

```
remainder <- ordinal number in WeChat Group % 4
for all questions do
    if question number % 4 == remainder then
        you should work on it
    end
end
```

Q1

14.对于虚拟存储系统，一次访存过程中，下列命中组合不可能发生的是（　　）

A. TLB 未命中，Cache 未命中，Page 未命中

B. TLB 未命中，Cache 命中，Page 命中

C. TLB 命中，Cache 未命中，Page 命中

D. TLB 命中，Cache 命中，Page 未命中

在虚拟存储系统中，MMU先访问TLB，如果TLB命中，则TLB返回一个对应页表项的地址。故若TLB命中，则一定存在一个页表项，不可能出现Page未命中的情况，D选项不可能发生。

D

Q2

16.为使虚拟内存系统有效发挥预期作用，所运行的程序应该具有的特点是
(　　)

A．该程序不应该含有过多的 I/O 操作
B．该程序的大小不应超过实际的内存容量
C．该程序应具有较好的局部性
D．该程序的指令相关不应过多

在使用虚拟内存执行程序时，虚拟内存必须经过MMU将虚拟地址转换为物理地址。由于虚拟内存开在disk上，这就导致了转换速度可能很慢，于是OS加入了页表、TLB、cache等可以加速地址转换的数据结构。和第六章以及书P565页所讲的一样，局部性保证程序趋向于在一个较小的工作集上执行，保证了这些数据结构的运行，提高了运行速度。所以局部性是最重要的因素，选C

C

Q3

15. Intel 的 IA32 体系结构采用二级页表，称第一级页表为页目录（Page Directory），第二级页表为页表（Page Table）。页面的大小为 4KB，页表项 4 字节。以下给出了页目录与若干页表中的部分内容，例如，页目录中的第 1 个项索引到的是页表 3，页表 1 中的第 3 个项索引到的是物理地址中的第 5 个页。则十六进制逻辑地址 8052CB 经过地址转换后形成的物理地址应为十进制的（）

页目录		页表 1		页表 2		页表 3	
VPN	页表号	VPN	页号	VPN	页号	VPN	页号
1	3	3	5	2	1	2	9
2	1	4	2	4	4	3	8
3	2	5	7	8	6	5	3

A. 21195
B. 29387
C. 21126
D. 47195

4KB=2^12，页内地址有12位，而页表项有4字节，故一页中的页表项数有2^10个，在有两级页表的32位地址系统中，这说明了VPN1=VPN2=10，VPO=12。于是，将0x8052CB转换为2进制，发现最高的10位为2，中间10位为5，及PPN=7，最后再对0x72CB做十进制的转换，即可得到29387。

B

Q4

13. 在 Core i7 中，关于虚拟地址和物理地址的说法，不正确的是：

A. VPO = CI + CO

B. PPN = TLBT + TLBI

C. VPN1 = VPN2 = VPN3 = VPN4

D. TLBT + TLBI = VPN

VPN=TLBT+TLBI。TLB输入一个虚拟内存地址，输出一个页表项，和PPN无关

B

Q5

8. 假定整型变量 A 的虚拟地址空间为 0x12345cf0，另一整形变量 B 的虚拟地址 0x12345d98，假定一个 page 的长度为 0x1000 byte，A 的物理地址数值和 B 的物理地址数值关系应该为：

A. A 的物理地址数值始终大于 B 的物理地址数值

B. A 的物理地址数值始终小于 B 的物理地址数值

C. A 的物理地址数值和 B 的物理地址数值大小取决于动态内存分配策略，

D. 无法判定两个物理地址值的大小

注意一个page的长度为2^12，这就代表了offset应当有12位，故A和B只有后三位不同，而对应的PPN相同，大小关系可以确定。

B

Q6

9. 虚拟内存中两层页表和单层页表相比，最主要的优势在于：

A. 更快的地址翻译速度

B. 能够提供页面更加精细的保护措施

C. 能够充分利用代码的空间局部性

D. 能够充分利用稀疏的内存使用模式

注意建立多级页表的初衷，是因为单层的页表空间占用太大了，如果内存非常稀疏，单层页表对于页表空间的利用率并不是很高，多级页表可以在一定程度上解决这一问题，所以我们使用多级页表。

D

Q7

10. 根据本课程介绍的 Intel x86-64 存储系统，填写表格中某一个进程从用户态切换至内核态时，和进程切换时对 TLB 和 cache 是否必须刷新。

A. ①不必刷新 ②不必刷新 ③刷新 ④不必刷新

B. ①不必刷新 ②不必刷新 ③不必刷新 ④不必刷新

C. ①刷新 ②不必刷新 ③刷新 ④刷新

D. ①刷新 ②不必刷新 ③不必刷新 ④刷新

在改变用户态/内核态时，并没有改变进程，不会对虚拟地址与物理地址的转换以及对应的数据进行改变，不必刷新。
而在进程切换时，不会对物理内存对应的数据产生影响，但是会对虚拟地址到物理地址的转换产生影响。因为多个虚拟内存可能对应一个相同的物理地址，如果仍然使用老的TLB，那么可能会出现访问到旧的物理地址的问题，导致程序出现问题。

A

Q8

15. 为了支持 16G 的虚拟地址空间，采用 3 级页表，页大小为 1KB，页表项大小依旧为 4 字节。现在映射总大小为 1MB 的虚拟地址，其分布未知但分布的最小单位限定为 1 字节，请问实现上述映射的页表结构占用的内存至少至多分别为多大？

A. 6KB，4113KB
B. 6KB，65793KB
C. 6KB，1052688KB
D. 3KB，4113KB

16G=2^34的虚拟地址空间，有三级页表，其中每一级页表大小为2^10，页表项大小为4，也就是说一页有2^8个页表项，对应虚拟地址中的8位，所以我们可以把虚拟地址划分为VPN1=VPN2=VPN3=8 VPO=12的四块。对于1MB=2^20的虚拟地址，最佳情况应是1MB完全连续，而offset有2^10这么大，所以我们需要2^10个页表项，用4个三级页表，1个二级页表，1个一级页表即可，即6KB

最坏的情况下，每个字节都是分开的，这种情况下我们需要2^20个页表项。但是可惜的是，这个操作系统最多有1个一级页表、256个二级页表、2^16个三级页表，所以最坏情况下就是把他们都撑满，2^16+2^8+1=65793，选B

B

Q9

16. 某 64 位系统，物理内存地址长度为 52 位，虚拟内存地址长度为 43 位，已知页的大小为 8KB，采用多级页表进行地址翻译，每级页表都占一页，则需要几级页表：

A. 2 级　　B. 3 级　　C. 4 级　　D. 5 级

页的大小为8KB说明p=13，即VPO=13，而64位系统PTE为8字节，那么一个页面最多1024个页表项，也就是说VPN=10，那么需要3级页表，这与物理地址长度无关

B

Q10

17. 在页表条目中，以下哪个条目是由 MMU 在读和写时进行设置，而由软件负责清除：

A. P 位，子页或子页表是否在物理内存中

B. G 位，是否为全局页（在任务切换时，不从 TLB 中驱逐出去）

C. CD 位，能/不能缓存子页或子页表

D. D 位，修改位，是否对子页进行了修改操作

课本P579“每次对一个页进行写了之后，MMU都会设置D位，内核可以调用一条特殊的内核模式指令清除引用位或者是修改位”

D

Q11

1. 下列关于虚存和缓存的说法中，**正确**的是：
A. TLB 是基于物理地址索引的高速缓存
B. 多数系统中，SRAM 高速缓存基于虚拟地址索引
C. 在进行**线程**切换后，TLB 条目绝大部分会失效
D. 多数系统中，在进行进程切换后，SRAM 高速缓存中的内容不会失效

A、P570“TLB是一个小的，虚拟寻址的缓存”
B、P570“尽管超出讨论范围，大多数系统是物理寻址的”
C、线程共享虚拟地址空间，TLB条目不会失效
D、P570“使用物理寻址，多个进程同时在高速缓存中有存储块和共享来自相同虚拟页面的块成为很简单的事情”

D

Q12

2. 下列选项中**错误**的是
A. 在使用虚拟地址空间的的系统中，程序引用的页面总数**必须不超过**物理内存总大小。
B. **主存中**的每个**有效**字节都有至少一个选自虚拟地址空间的**虚拟地址**和一个选自物理地址空间的**物理地址**。
C. 当在程序中正常调用 malloc 函数时，操作系统会分配出相应大小（例如 k 个）的**连续虚拟页面**，并且将它们映射到物理内存中**任意位置**的 k 个**物理页面**。
D. **不同**进程的多个**虚拟**页面可以映射到**同一个**共享**物理**页面上。

A、应该是“不超过虚拟内存总大小”
B、P561“主存中每个有效字节都至少有一个选自虚拟地址空间的虚拟地址和一个选自物理地址空间的物理地址”
C、P566“当一个运行在用户进程中的程序要求额外的堆空间时，操作系统分配一个适当数字个连续的虚拟内存页面，并将他们映射到物理内存任意位置的k个任意的物理页面”
D、P566“注意，多个虚拟页面可以映射到同一个共享的物理页面”

A

Q13

14、用带有 header 和 footer 的隐式空闲链表实现分配器时，如果一个应用请求一个 3 字节的块，下列说法哪一项是错误的？答：（　　）

A. 搜索空闲链表时，存储利用率为：best fit > next fit > first fit

B. 搜索空闲链表时，吞吐率为：next fit > first fit > best fit

C. 在 x86 机器上，malloc(3)实际分配的空闲块大小可能为 8 字节

D. 在 x64 机器上，malloc(3)返回的地址可能为 2147549777

A、P594“下一次适配内存利用率比首次适配配低得多”故错误

B、P594“下一次适配比首次适配运行起来明显快一些”“使用最佳适配的缺点是要求对堆进行彻底的搜索”故正确

C、malloc（a）分配a+4大小的空闲块，块大小向上舍入为接近的8的倍数，故正确

D、x64机器上，malloc的返回地址为16倍数，故错误

AD

Q14

11. 在 C 语言中实现 Mark-and-Sweep 算法时，可以基于以下哪个假设：（宿主机为 32 位机器）

A. 所有指针指向一个块的起始地址

B. 所有指针数据都是 4 字节对齐

C. 只需要扫描数据类型为指针的堆中的数据空间

D. 只需要扫描所有长度为 4 字节的堆中的数据空间

课本P608“对isptr没有任何办法判断输入参数p是不是指针”因此C、D错误，“即使我们知道p是一个指针，也没有明显的方式判断其是否指向一个已分配块的有效载荷的某个位置”A错误，B是符合事实的，B正确

B

Q15

11．关于动态内存分配，下列说法中正确的是：
A．显式分配器可以重新排列请求顺序，从而最大化内存利用率
B．显式分配器可以修改已分配的块，把内容复制到别的位置，从而消除外部碎片
C．通常显式分配器会比隐式分配器更快
D．C 语言中如果某个已分配块不再可达，那它就会被释放并返回给空闲链表

A、P590“不允许分配器为提高性能重新排列或者缓冲请求”，A错误
B、P590“不修改已分配的块”，B错误
C、正确
D、C语言没有垃圾回收器，D错误

Q16

12．在设计分配器时，下列说法中错误的是：
A．搜索空闲链表时，存储利用率为：best fit > next fit > first fit
B．带头部的隐式空闲链表，合并（内存中的）下一个空闲块可在常数时间内完成
C．如果采取立刻合并策略（immediate coalescing），会在某些请求模式中出现反复合并又分割的情况，于是会有较小的吞吐率
D．分配器使用二叉树结构，主要是为了能够更快地找到适配的空闲块

A、P594“一些研究表明，下一次适配内存利用率比首次适配低得多”A错误
B、P595“如果是，就将它的大小简单加到当前块头部的大小上，这两个块在常数时间内被合并”B正确
C、P595“但是对于某些请求模式，这种方法会产生一种形式的抖动，块会反复合并，然后马上分割”C正确
D、使用二叉树时寻找适配的空闲块的时间复杂度较低，D正确

Q17

7. 下列与虚拟内存有关的说法中哪些是不对的？

A. 操作系统为每个进程提供一个独立的页表，用于将其虚拟地址空间映射到物理地址空间。

B. MMU 使用页表进行地址翻译时，虚拟地址的虚拟页面偏移与物理地址的物理页面偏移是相同的。

C. 若某个进程的工作集大小超出了物理内存的大小，则可能出现抖动现象。

D. 动态内存分配管理，采用双向链表组织空闲块，使得首次适配的分配与释放均是空闲块数量的线性时间。

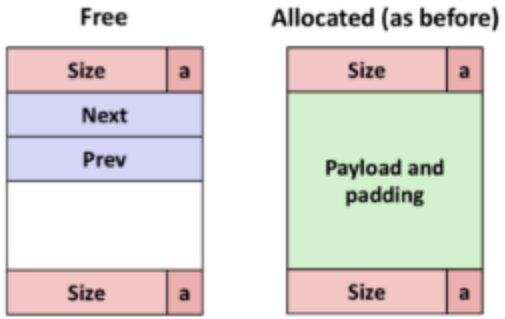
- 释放一个块的时间也可能是一个常数，后进先出策略

D

Q18

12. 现在有一个用户程序执行了如下调用序列

```
void *p1 = malloc(16);
void *p2 = malloc(32);
void *p3 = malloc(32);
void *p4 = malloc(48);
free(p2);
void *p5 = malloc(4);
free(p3);
void *p6 = malloc(56);
void *p7 = malloc(10);
```



内存分配器内部使用**显式空闲链表**实现，按照地址从低到高顺序来维护空闲链表节点顺序。分配块格式参照上图。每个分配块 16 字节对齐，头部和脚部的大小都是 4 字节。分配算法采用**首次适配**算法，将适配到的空闲块的第一部分作为分配块，剩余部分变成新的空闲块，并采用**立即合并**策略。假设初始空闲块的大小是 1KB。那么以上调用序列完成后，分配器管理的这 1KB 内存区域中，**内部碎片**的总大小是____，链表里**第一个**空闲块的大小是_____。

A. 58Byte，848Byte

B. 26Byte，32Byte

C. 74Byte，48Byte

D. 74Byte，16Byte

D

指针	P1	P2		P3	P4		
对齐前大小	16+8	32+8		32+8	48+8		
对齐后大小	32	48		48	64		
指针	**P1**	**P5**	**空**	**P3**	**P4**		
对齐前大小	16+8	4+8		32+8	48+8		
对齐后大小	32	16	32	48	64		
指针	**P1**	**P5**	**空**		**P4**		
对齐前大小	16+8	4+8			48+8		
对齐后大小	32	16	80		64		
指针	**P1**	**P5**	**P6**	**空**	**P4**	**P7**	
对齐前大小	16+8	4+8	56+8		48+8	10+8	
对齐后大小	32	16	64	16	64	32	

- 头部尾部共8字节
- 内存碎片包含头部和尾部、对齐的未使用的内存，不包含空闲块
- 第一个空闲块为16

Q19

18. 关于写时复制（copy-on-write）技术的说法，不正确的是：

A. 写时复制既可以发生在父子进程之间，也可以发生在对等线程之间

B. 写时复制既需要硬件的异常机制，也需要操作系统软件的配合

C. 写时复制既可以用于普通文件，也可以用于匿名文件

D. 写时复制既可以用于共享区域，也可以用于私有区域

写时复制不可以放在对等线程之间

A

Q20

2. 阅读下列代码并回答选项。（已知文件“input.txt”中的内容为“12”，头文件没有列出）

```
void *Mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);

int main(){
    int status;
    int fd = Open("./input.txt", O_RDWR);
    char* bufp = Mmap(NULL, 2, PROT_READ | PROT_WRITE ,
                      MAP_PRIVATE, fd, 0);
    if (Fork()>0){
       while(waitpid(-1,&status,0)>0);
       *(bufp+1) = '1';
       Write(1, bufp, 2); // 1: stdout
       bufp = Mmap(NULL, 2, PROT_READ, MAP_PRIVATE, fd, 0);
       Write(1, bufp, 2);
    }
    else{
        *bufp = '2';
        Write(1, bufp, 2);
    }
```

在 shell 中运行该程序，正常运行时的终端输出应为

A. 221112　B. 222121　C. 222112　D. 221111

A

- Mmap第四个参数为权限，Private没有修改的权限
- 父进程waitpid等待子进程运行
- 子进程缓冲区中被改为22并输出，由于没有文件修改权限，txt中仍然为12
- 子进程结束运行，父进程缓冲区被修改为11并输出
- 重新载入缓冲区，输出文件中12
- 输出为221112

Q21

第五题（12 分）

1. （2 分，每空一分，考察多级页表和单级页表的设计思想的理解）

在进行地址翻译的过程中，操作系统需要借助页表(Page Table)的帮助。考虑一个 32 位的系统，页大小是 4KB，页表项(Page Table Entry)大小是 4 字节(Byte)，如果不使用多级页表，常驻内存的页表一共需要_____页。

考虑下图已经显示的物理内存分配情况，在二级页表的情况下，已经显示的区域的页表需要占据_____页。

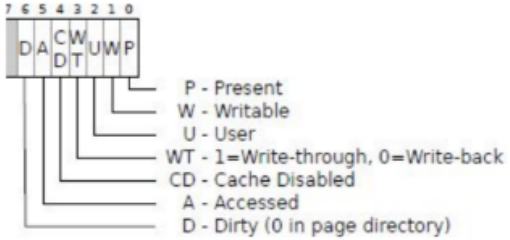
- 一个页表存储的项数为4kB/4Byte=2^10
- 常驻页表的大小为2^32/2^10=2^22，即2^22/4kB=2^10页

- 11264=1024*11，需要11个1级页表
- 已分配的1级页表会生成2级页表，0-2047需要2页，12039需要1页，12040-11263需要1页，共需要4页二级页表
- 共15页

Q21

2. （6 分，每错一空扣一分，扣完为止。考察地址翻译过程）

IA32 体系采用小端法和二级页表。其中两级页表大小相同，页大小均为 4KB，结构也相同。TLB 采用直接映射。TLB 和页表每一项的后 7 位含义如下图所示。为简便起见，假设 TLB 和页表每一项的后 8~12 位都是 0 且不会被改变。注意后 7 位值为“27”则表示可读写。



当系统运行到某一时刻时，TLB 内容如下：

索引	TLB 标记	内容	有效位
0	0x04013	0x3312D027	1
1	0x01000	0x24833020	0
2	0x005AE	0x00055004	1
3	0x00402	0x24AEE020	0
4	0x0AA00	0x0005505C	0
5	0x0000A	0x29DEE000	1
6	0x1AE82	0x00A23027	1
7	0x28DFC	0x00023000	0

一级页表的基地址为 0x0C23B00，物理内存中的部分内容如下：

地址	内容	地址	内容	地址	内容	地址	内容
00023000	E0	00023001	BE	00023002	EF	00023003	BE
00023120	83	00023121	C8	00023122	FD	00023123	12
00023200	23	00023201	FD	00023202	BC	00023203	DE
00023320	33	00023321	29	00023322	E5	00023323	D2
0005545C	97	0005545D	C2	0005545E	7B	0005545F	45
00055464	97	00055465	D2	00055466	7B	00055467	45
0C23B020	27	0C23B021	EB	0C23B022	AE	0C23B023	24
0C23B040	27	0C23B041	40	0C23B042	DE	0C23B043	29
0C23B080	05	0C23B081	5D	0C23B082	05	0C23B083	00
2314D200	23	2314D201	12	2314D202	DC	2314D203	0F
2314D220	A9	2314D221	45	2314D222	13	2314D223	D2
29DE404C	27	29DE404D	42	29DE404E	BA	29DE404F	00
29DE4400	D0	29DE4401	5C	29DE4402	B4	29DE4403	2A

此刻，系统先后试图对两个已经缓存在 cache 中的内存地址进行写操作，请分析**完成写之后**系统的状态（写的地址和上面的内存地址无交集），完成下面的填空。若不需要某次访问或者缺少所需信息，请填“\”。

第一次向地址 0xD7416560 写入内容，TLB 索引为：_____，完成写之后该项 TLB 内容为：_____，

二级页表页表项地址为：_____，物理地址为：_____。

第二次向地址 0x0401369B 写入内容，

TLB 索引为：_____，完成写之后该项 TLB 内容为：_____

二级页表页表项地址为：_____，物理地址为：_____。

- 页大小4kB，12位offset（可记忆32位为10+10+12）
- 索引0-7可推得索引位3位，标志位17位
- D7416560偏移量为560，索引位为110，标志位为1101 0111 0100 0001 0即1AE82，页命中，内容为0x00A23067，故不需要查找二级页表，只需要将后12位替换为offset就得到物理地址00A23560
- 0401369B偏移量为69B，索引位为011，标志位为0000 0100 0000 0001 0即00802，索引位3对应的有效位为0，触发缺页异常；
- 一级页表基址0C23B00，加上VA的前十位0000 0100 00乘4，得到0C23B40，查表对应的32位地址为29 DE 40 27（注意小端法），忽略后12位，得到二级页表的基址29DE4000
- 二级页表基址29DE4000，加上VA的中间十位00 0001 0011乘4，得到29DE404C，查表对应的32位地址为00 BA 42 27，忽略后12位，得到PPN为00BA4，加上offset得到物理地址00BA469B
- TLB内容会替换，PPN即00BA4，后12位按照含义应该为067（第7位D含义为被写过，被写过则为1），故最终替换为00BA4067

Q21

3. （2 分，考察对于虚拟内存独立地址空间的理解）

本学期的 fork bomb 作业中，大家曾用 fork 逼近系统的进程数量上限。下面有一个类似的程序，请仔细阅读程序并填空。

```
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
#define N 4
int main() {
        volatile int pid, cnt = 1;
        for (int i = 0; i < N; i++) {
                if ((pid = fork()) > 0) {
                        cnt++;
        }
        while (wait(NULL) > 0);
        return 0;
}
```

整个过程中，变量 cnt 最大值是_____。假设所有的数据都已经存在于内存中，pid 和 cnt 在同一个物理页中。从第一个进程开始执行 for 语句开始，此过程对于 cnt 的操作至少会导致页表中_____次虚拟页对应的物理页被修改。

- 每次fork父进程cnt++，共调用四次fork，最大为5
- 父子进程对cnt操作，由于操作了同一个变量，是写时复制，每次cnt++都会导致虚拟页与物理页的对应关系发生修改，总共发生了15次

Q21

1024, 5

6, 0x00A23067, \, 0x00A23560

3, 0x00BA4067, 0x29DE404C, 0x00BA469B

5, 15

Q22

第五题（12 分）

1. （2 分，每空一分，考察多级页表和单级页表的设计思想的理解）

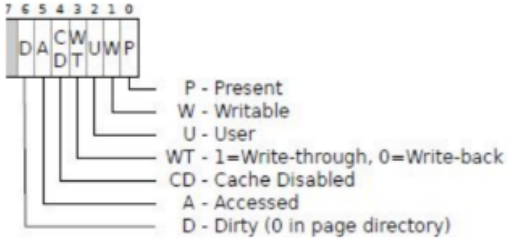
在进行地址翻译的过程中，操作系统需要借助页表(Page Table)的帮助。考虑一个 32 位的系统，页大小是 4KB，页表项(Page Table Entry)大小是 4 字节(Byte)，如果不使用多级页表，常驻内存的页表一共需要_____页。

考虑下图已经显示的物理内存分配情况，在二级页表的情况下，已经显示的区域的页表需要占据_____页。

Q22

2. （6 分，每错一空扣一分，扣完为止。考察地址翻译过程）

IA32 体系采用小端法和二级页表。其中两级页表大小相同，页大小均为 4KB，结构也相同。TLB 采用直接映射。TLB 和页表每一项的后 7 位含义如下图所示。为简便起见，假设 TLB 和页表每一项的后 8~12 位都是 0 且不会被改变。注意后 7 位值为“27”则表示可读写。



当系统运行到某一时刻时，TLB 内容如下：

索引	TLB 标记	内容	有效位
0	0x04013	0x3312D027	1
1	0x01000	0x24833020	0
2	0x005AE	0x00055004	1
3	0x00402	0x24AEE020	0
4	0x0AA00	0x0005505C	0
5	0x0000A	0x29DEE000	1
6	0x1AE82	0x00A23027	1
7	0x28DFC	0x00023000	0

一级页表的基地址为 0x0C23B00，物理内存中的部分内容如下：

地址	内容	地址	内容	地址	内容	地址	内容
00023000	E0	00023001	BE	00023002	EF	00023003	BE
00023120	83	00023121	C8	00023122	FD	00023123	12
00023200	23	00023201	FD	00023202	BC	00023203	DE
00023320	33	00023321	29	00023322	E5	00023323	D2
0005545C	97	0005545D	C2	0005545E	7B	0005545F	45
00055464	97	00055465	D2	00055466	7B	00055467	45
0C23B020	27	0C23B021	EB	0C23B022	AE	0C23B023	24
0C23B040	27	0C23B041	40	0C23B042	DE	0C23B043	29
0C23B080	05	0C23B081	5D	0C23B082	05	0C23B083	00
2314D200	23	2314D201	12	2314D202	DC	2314D203	0F
2314D220	A9	2314D221	45	2314D222	13	2314D223	D2
29DE404C	27	29DE404D	42	29DE404E	BA	29DE404F	00
29DE4400	D0	29DE4401	5C	29DE4402	B4	29DE4403	2A

此刻，系统先后试图对两个已经缓存在 cache 中的内存地址进行写操作，请分析**完成写之后**系统的状态（写的地址和上面的内存地址无交集），完成下面的填空。若不需要某次访问或者缺少所需信息，请填“\”。

第一次向地址 0xD7416560 写入内容，TLB 索引为：_____，完成写之后该项 TLB 内容为：_____，

二级页表页表项地址为：_____，物理地址为：_____。

第二次向地址 0x0401369B 写入内容，

TLB 索引为：_____，完成写之后该项 TLB 内容为：_____

二级页表页表项地址为：_____，物理地址为：_____。

Q22

3. (2 分，考察对于虚拟内存独立地址空间的理解)

本学期的 fork bomb 作业中，大家曾用 fork 逼近系统的进程数量上限。下面有一个类似的程序，请仔细阅读程序并填空。

```
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
#define N 4
int main() {
        volatile int pid, cnt = 1;
        for (int i = 0; i < N; i++) {
                if ((pid = fork()) > 0) {
                        cnt++;
        }
        while (wait(NULL) > 0);
        return 0;
}
```

整个过程中，变量 cnt 最大值是_____。假设所有的数据都已经存在于内存中，pid 和 cnt 在同一个物理页中。从第一个进程开始执行 for 语句开始，此过程对于 cnt 的操作至少会导致页表中_____次虚拟页对应的物理页被修改。

Q22

1024, 5

6, 0x00A23067, \, 0x00A23560

3, 0x00BA4067, 0x29DE404C, 0x00BA469B

5, 15

Q23

第五题（12 分）

1. （2 分，每空一分，考察多级页表和单级页表的设计思想的理解）

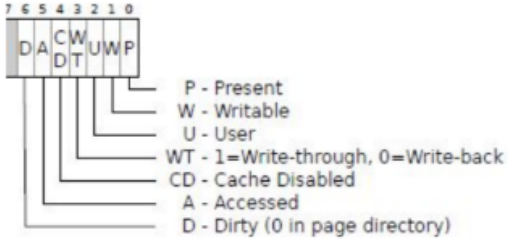
在进行地址翻译的过程中，操作系统需要借助页表(Page Table)的帮助。考虑一个 32 位的系统，页大小是 4KB，页表项(Page Table Entry)大小是 4 字节(Byte)，如果不使用多级页表，常驻内存的页表一共需要_____页。

考虑下图已经显示的物理内存分配情况，在二级页表的情况下，已经显示的区域的页表需要占据_____页。

Q23

2. （6 分，每错一空扣一分，扣完为止。考察地址翻译过程）

IA32 体系采用小端法和二级页表。其中两级页表大小相同，页大小均为 4KB，结构也相同。TLB 采用直接映射。TLB 和页表每一项的后 7 位含义如下图所示。为简便起见，假设 TLB 和页表每一项的后 8~12 位都是 0 且不会被改变。注意后 7 位值为“27”则表示可读写。



当系统运行到某一时刻时，TLB 内容如下：

索引	TLB 标记	内容	有效位
0	0x04013	0x3312D027	1
1	0x01000	0x24833020	0
2	0x005AE	0x00055004	1
3	0x00402	0x24AEE020	0
4	0x0AA00	0x0005505C	0
5	0x0000A	0x29DEE000	1
6	0x1AE82	0x00A23027	1
7	0x28DFC	0x00023000	0

一级页表的基地址为 0x0C23B00，物理内存中的部分内容如下：

地址	内容	地址	内容	地址	内容	地址	内容
00023000	E0	00023001	BE	00023002	EF	00023003	BE
00023120	83	00023121	C8	00023122	FD	00023123	12
00023200	23	00023201	FD	00023202	BC	00023203	DE
00023320	33	00023321	29	00023322	E5	00023323	D2
0005545C	97	0005545D	C2	0005545E	7B	0005545F	45
00055464	97	00055465	D2	00055466	7B	00055467	45
0C23B020	27	0C23B021	EB	0C23B022	AE	0C23B023	24
0C23B040	27	0C23B041	40	0C23B042	DE	0C23B043	29
0C23B080	05	0C23B081	5D	0C23B082	05	0C23B083	00
2314D200	23	2314D201	12	2314D202	DC	2314D203	0F
2314D220	A9	2314D221	45	2314D222	13	2314D223	D2
29DE404C	27	29DE404D	42	29DE404E	BA	29DE404F	00
29DE4400	D0	29DE4401	5C	29DE4402	B4	29DE4403	2A

此刻，系统先后试图对两个已经缓存在 cache 中的内存地址进行写操作，请分析**完成写之后**系统的状态（写的地址和上面的内存地址无交集），完成下面的填空。若不需要某次访问或者缺少所需信息，请填“\”。

第一次向地址 0xD7416560 写入内容，TLB 索引为：_____，完成写之后该项 TLB 内容为：_____，

二级页表页表项地址为：_____，物理地址为：_____。

第二次向地址 0x0401369B 写入内容，

TLB 索引为：_____，完成写之后该项 TLB 内容为：_____

二级页表页表项地址为：_____，物理地址为：_____。

Q23

3. （2 分，考察对于虚拟内存独立地址空间的理解）

本学期的 fork bomb 作业中，大家曾用 fork 逼近系统的进程数量上限。下面有一个类似的程序，请仔细阅读程序并填空。

```
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
#define N 4
int main() {
        volatile int pid, cnt = 1;
        for (int i = 0; i < N; i++) {
                if ((pid = fork()) > 0) {
                        cnt++;
        }
        while (wait(NULL) > 0);
        return 0;
}
```

整个过程中，变量 cnt 最大值是_____。假设所有的数据都已经存在于内存中，pid 和 cnt 在同一个物理页中。从第一个进程开始执行 for 语句开始，此过程对于 cnt 的操作至少会导致页表中_____次虚拟页对应的物理页被修改。

Q23

1024, 5

6, 0x00A23067, \, 0x00A23560

3, 0x00BA4067, 0x29DE404C, 0x00BA469B

5, 15

Q24

第五题（12 分）

1. （2 分，每空一分，考察多级页表和单级页表的设计思想的理解）

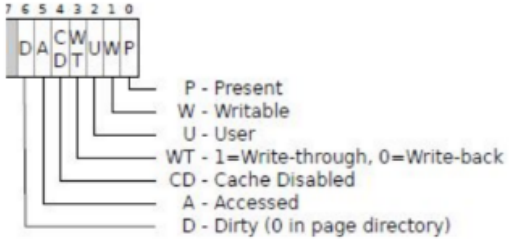
在进行地址翻译的过程中，操作系统需要借助页表(Page Table)的帮助。考虑一个 32 位的系统，页大小是 4KB，页表项(Page Table Entry)大小是 4 字节(Byte)，如果不使用多级页表，常驻内存的页表一共需要_____页。

考虑下图已经显示的物理内存分配情况，在二级页表的情况下，已经显示的区域的页表需要占据_____页。

Q24

2. （6 分，每错一空扣一分，扣完为止。考察地址翻译过程）

IA32 体系采用小端法和二级页表。其中两级页表大小相同，页大小均为 4KB，结构也相同。TLB 采用直接映射。TLB 和页表每一项的后 7 位含义如下图所示。为简便起见，假设 TLB 和页表每一项的后 8~12 位都是 0 且不会被改变。注意后 7 位值为“27”则表示可读写。



当系统运行到某一时刻时，TLB 内容如下：

索引	TLB 标记	内容	有效位
0	0x04013	0x3312D027	1
1	0x01000	0x24833020	0
2	0x005AE	0x00055004	1
3	0x00402	0x24AEE020	0
4	0x0AA00	0x0005505C	0
5	0x0000A	0x29DEE000	1
6	0x1AE82	0x00A23027	1
7	0x28DFC	0x00023000	0

一级页表的基地址为 0x0C23B00，物理内存中的部分内容如下：

地址	内容	地址	内容	地址	内容	地址	内容
00023000	E0	00023001	BE	00023002	EF	00023003	BE
00023120	83	00023121	C8	00023122	FD	00023123	12
00023200	23	00023201	FD	00023202	BC	00023203	DE
00023320	33	00023321	29	00023322	E5	00023323	D2
0005545C	97	0005545D	C2	0005545E	7B	0005545F	45
00055464	97	00055465	D2	00055466	7B	00055467	45
0C23B020	27	0C23B021	EB	0C23B022	AE	0C23B023	24
0C23B040	27	0C23B041	40	0C23B042	DE	0C23B043	29
0C23B080	05	0C23B081	5D	0C23B082	05	0C23B083	00
2314D200	23	2314D201	12	2314D202	DC	2314D203	0F
2314D220	A9	2314D221	45	2314D222	13	2314D223	D2
29DE404C	27	29DE404D	42	29DE404E	BA	29DE404F	00
29DE4400	D0	29DE4401	5C	29DE4402	B4	29DE4403	2A

此刻，系统先后试图对两个已经缓存在 cache 中的内存地址进行写操作，请分析**完成写之后**系统的状态（写的地址和上面的内存地址无交集），完成下面的填空。若不需要某次访问或者缺少所需信息，请填“\”。

第一次向地址 0xD7416560 写入内容，TLB 索引为：_____，完成写之后该项 TLB 内容为：_____，

二级页表页表项地址为：_____，物理地址为：_____。

第二次向地址 0x0401369B 写入内容，

TLB 索引为：_____，完成写之后该项 TLB 内容为：_____

二级页表页表项地址为：_____，物理地址为：_____。

Q24

3. （2 分，考察对于虚拟内存独立地址空间的理解）

本学期的 fork bomb 作业中，大家曾用 fork 逼近系统的进程数量上限。下面有一个类似的程序，请仔细阅读程序并填空。

```
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
#define N 4
int main() {
        volatile int pid, cnt = 1;
        for (int i = 0; i < N; i++) {
                if ((pid = fork()) > 0) {
                        cnt++;
        }
        while (wait(NULL) > 0);
        return 0;
}
```

整个过程中，变量 cnt 最大值是_____。假设所有的数据都已经存在于内存中，pid 和 cnt 在同一个物理页中。从第一个进程开始执行 for 语句开始，此过程对于 cnt 的操作至少会导致页表中_____次虚拟页对应的物理页被修改。

Q24

1024, 5

6, 0x00A23067, \, 0x00A23560

3, 0x00BA4067, 0x29DE404C, 0x00BA469B

5, 15