

Virtual Memory

13 元培数科 常欣海

Here we go! →

2024/12/4



Emphasis

Notes

■ Note Link

虚拟内存

- VM 基本概念
- VM基本思想
 - 物理寻址
 - 虚拟寻址
- VM意义
- VM作为工具
 - VM作为内存存储的工具
 - 虚拟页
 - 页表
 - 效率保证
 - VM作为内存管理的工具
 - VM作为内存保护的工具
- 地址翻译
 - 符号表
 - 宏观理解
 - 微观理解
 - 页表
 - 页表+Cache
 - 页表+TLB+Cache
 - 多级页表+TLB+Cache
- 陷阱细节
 - Cache
 - TLB
 - 页面大小
- 例子：内存系统
 - 简单例子
 - Core i7
 - 系统结构
 - 地址翻译
 - PTE结构
 - 内存管理
 - 内存映射
 - 交换空间
 - 相关函数

虚拟内存

VM 基本概念

- **PA:** *Physical Address* 物理地址, n 位
 - PPN: Physical Page Number 物理页号, $m - p$ 位
 - PPO: Physical Page Offset 物理页面偏移, p 位
- **VA:** *Virtual Address* 虚拟地址, m 位
 - VPN: Virtual Page Number 虚拟页号, $n - p$ 位
 - VPO: Virtual Page Offset 虚拟页面偏移, p 位
- **MMU:** *Memory Management Unit* 内存管理单元
- 地址空间：一个非负整数地址的有序集合 \rightarrow 一个进程可用于寻址内存的一套地址集合
 - 线性地址空间: $\{0, 1, 2, \dots\}$
 - VAS: 虚拟地址空间 $\{0, 1, 2, \dots, N - 1\}$, $N = 2^n$
 - PAS: 物理地址空间 $\{0, 1, 2, \dots, M - 1\}$, $M = 2^m$
- **VP:** *Virtual Page* 虚拟页, 大小 $P = 2^p$
- **PP:** *Physical Page* 物理页/页帧, 大小 $P = 2^p$
- **PTE:** *Page Table Entry* 页表条目, 页表将虚拟页映射到物理页
 - PTBR: *Page Table Base Register* 页表基址寄存器
 - PTEA: 页表条目地址
- **TLB:** *Translation Lookaside Buffer* 快表, 针对PTE的缓存, 存储在MMU中
 - TLBT: TLB索引, $n - p - t$ 位

Homework Review

Exercises

E1

15. 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

E1

15. 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

答案：D ↵

解析：属于中等题。考察虚拟内存和高速缓存的关系，并且和进程/线程有一定联系。 ↵

A 课本原话：A TLB is a small, virtually addressed cache where each line holds a block consisting

of a single PTE. ↵

B 课本原话：Although a detailed discussion of the trade-offs is beyond our scope here, most

systems opt for physical addressing. ↵

C 线程共享虚拟地址空间，所以 TLB 中条目不会失效 ↵

D 课本原话：With physical addressing, it is straightforward for multiple processes to have blocks
in the cache at the same time and to share blocks from the same virtual pages. ↵

E2

16. 阅读下列代码。(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);
    }
    close(fd);
    return 0;
}
```

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

E2

答案：A ←

解析：属于中等题。主要考察对私有对象 COW 机制及 `mmap` 函数的理解。←

程序打开文件描述符 `fd` 后，使用 `mmap` 函数创建了一块映射到文本 “`./input.txt`” 所在区域的私有虚拟内存区域，其地址存储在指针 `bufp` 中。父进程创建完子进程后，首先等待子进程结束。子进程在试图对`*bufp`写入‘2’时，触发了一次 COW。它在物理内存中创建一个“`input.txt`”文本所在页面的副本，并在新的页面里完成写入操作，写入完成后子进程 `bufp` 所在地址的第一个字节值为新写入的‘2’，第二个字节值为原始值‘2’。子进程结束后，其缓冲区内“21”的值正常输出。父进程在试图对`*(bufp+1)`写入‘1’，同样触发了一次 COW，`bufp` 所在地址的第一个字节为原始值‘1’，第二个字节为新写入的‘1’，故父进程输出“11”。最后父进程重新创建一块映射到“`input.txt`”所在页面的虚拟内存并输出。由于上述写入都是在新的物理页面下完成的，“`input.txt`”所在页面没有发生修改，故最后一步父进程的输出为“12”。综上，最终在终端的输出为“221112”。←

E3

14. 假设在 64 位系统下页大小被设置为 16KB，那么采用类似 Core i7 的地址翻译过程，四级页表最多可以索引多少位地址空间？
- A. 64
 - B. 58
 - C. 54
 - D. 52

E3

14. 假设在 64 位系统下页大小被设置为 16KB，那么采用类似 Core i7 的地址翻译过程，四级页表最多可以索引多少位地址空间？
- A. 64
 - B. 58
 - C. 54
 - D. 52
14. B。根据 Core i7 的用法，每张页表恰好有一页，每个页表项 8 字节，所以每个 VPN 都是 $14 - 3 = 11$ 位，于是四级 VPN 和 VPO 一共 58 位。

E4

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

E4

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
15. B. 第二句话表明题目中的内存是按字节寻址的。首先计算 VPO 为 10 位，每个 VPN 都是 8 位，恰好是 34 位虚拟地址。最好情况是 1 MB 虚拟地址映射到连续的页，即占有 1024 页，所以需要 1024 个三级页表条目（占有 4 页），1 个二级页表条目（占 1 页）和 1 个一级页表条目（占 1 页），即需要 6 KB。最坏情况是每字节都恰好映射到不同的页面，也就是占有 1024×1024 个不同的页面（可以实现）。这些页面可以用尽全部 256×256 张三级页表（比如每 16 页用一个三级页表），所以需要 $256 \times 256 + 256 + 1 = 65793$ 页，即 65 973 KB。

E5

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

E5

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
16. B。和物理地址无关，VPO 为 13 位，VPN 均为 10 位，所以需要三级页表。

E6

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

E6

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

17. D。根据题目的描述就可以推断是 A 位和 D 位。

E7

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

E7

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
18. A。未解之谜。如果将共享区域理解为 MAP_SHARED 则 D 错误；否则共享库也是一种共享区域，会发生 COW。如果参照 Wikipedia 上的解说，COW 是一种技术，用户可以自行实现，那么对等线程之间也可能发生 COW（例如 `string y=x` 之后修改 `y`，C++ 中用 COW 实现）。A 可能是更好的选择。

E8

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

E8

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
10. A。高速缓存是物理寻址的，所以一定不用刷新。用户态和内核态的转换不会改变内存映射，而上下文切换会改变虚拟内存到物理地址的对应关系，所以 TLB 在上下文切换时要刷新。

E9

11. 已知某系统页面长 2KB，页表项 8 字节，采用多层分页策略映射 48 位虚拟地址空间。若限定最高层页表占 1 页，则它可以采用多少层的分页策略？
- A. 3 层
 - B. 4 层
 - C. 5 层
 - D. 6 层

E9

11. 已知某系统页面长 2KB，页表项 8 字节，采用多层分页策略映射 48 位虚拟地址空间。若限定最高层页表占 1 页，则它可以采用多少层的分页策略？
- A. 3 层
 - B. 4 层
 - C. 5 层
 - D. 6 层
11. C。VPO 为 11 位，页表均占一页，则 VPN 都为 8 位，所以映射满 48 位地址空间至少需要 5 级页表。

E10

13. 在一个具有 TLB 和高速缓存的系统中，假设地址翻译使用四级页表来进行，且不发生缺页异常，那么在 CPU 访问某个虚拟内存地址的过程中，至少会访问（ ）次物理内存，至多会访问（ ）次物理内存。
- A. 0, 4
 - B. 0, 5
 - C. 1, 4
 - D. 1, 5

E10

13. 在一个具有 TLB 和高速缓存的系统中，假设地址翻译使用四级页表来进行，且不发生缺页异常，那么在 CPU 访问某个虚拟内存地址的过程中，至少会访问（ ）次物理内存，至多会访问（ ）次物理内存。
- A. 0, 4
 - B. 0, 5
 - C. 1, 4
 - D. 1, 5
13. B。最好情况是高速缓存和 TLB 都命中，不需要访问内存；最坏情况是都不命中，需要访问四级页表项以及主存中的数据，一共 5 次。

E11

第五题. 请结合教材第九章“虚拟内存”的有关知识回答问题(15分)

IA32 体系采用**小端法**、32位虚拟地址和两级页表。两级页表大小相同，页大小都是 4 KB = 2^{12} Byte，结构也相同。TLB 采用**直接映射**，4位组索引。TLB 和页表每一项格式如图所示：

31	12	11	9	8	7	6	5	4	3	2	1	0
Address of 4KB page frame	Ignored	G	A	D	A	C	W	/	/	P	D	T

部分位的含义如下：

0 (P): 1 表示存在, 0 表示不存在

1 (R/W): 1 表示可写, 0 表示只读

2 (U/S): 1 表示内核模式, 0 表示用户模式

当系统运行到某一时刻, TLB 有效位为 1 的条目如下(未列出部分都是无效的)：

索引(十进制)	TLB 标记	内容
0	0x0400	0x0ec91313
3	0x02ff	0x5d2bac01
5	0xd551	0x019fa42d
11	0x55a6	0xfd3c66b
13	0x5515	0xb591926b

一级页表地址为 0x00e66000, 物理内存中的部分内容如下(均为十六进制)：

地址	内容	地址	内容	地址	内容
00e15000	21	00e15001	2d	00e15002	ee
00e15003	c0	00e154d0	ff	00e154d1	a0
00e66001	a1	00e66002	a4	00e66003	67
00e66004	21	00e66005	57	00e66006	61
00e66007	00	21e7e000	42	21e7e001	67
21e7e002	9a	21e7e003	7c	c0ee2000	6f
c0ee2001	d5	c0ee2002	7e	c0ee24d0	48
c0ee24d1	83	c0ee24d2	ec	c0ee2d21	11
c0ee2d22	6b	c0ee2d23	82	c0ee2d24	8a

1. 将 cache 清空, 访问一个在主存中的虚拟地址, TLB 命中, 没有触发缺页异常, 这一过程中, 需要访问物理内存 ____ 次(3分)。具体来说, 如果该地址为 $y = 0xd5151213$, y 地址所具有的**实质权限**是 _____ (多选题, 选对才得分 2 分)。

①可写 ②只读 ③用户模式权限 ④内核模式权限

2. 不考虑第一小问, 将 cache 清空, 访问一个在主存中的虚拟地址, TLB 不命中, 没有触发缺页异常, 这一过程中, 需要访问物理内存 ____ 次(3分)。具体来说, 如果该地址为 $x = 0x004004d0$, 那么 x 对应的二级页表起始地址是 _____ (填写 16 进制, 例如 0x00123000, 2 分), x 地址上单字节的内容是 _____ (填写 16 进制, 例如 0x00, 1 分)。

3. 考虑下面计算矩阵和向量乘法代码:

```
1 int *mat_vec_mul(int **A, int *x, int n)
2 {
3     int i, j;
4     int *y = (int *)malloc(n * sizeof(int));
5     for (i = 0; i < n; i++)
6         for (j = 0; j < n; j++)
7             y[i] += A[i][j] * x[j];
8     return y;
9 }
```

(1) 在 64 位 Linux 机器中运行该代码, 输入同一组合法的参数后, 每次运行返回的向量的值都不一样, 修复这一错误有一种简单的方法, 将第 ____ 行改为 ____。(第二空填写 C 代码, 每空 1 分, 共 2 分)

可能用到的函数:

```
void *memcpy(void *dest, const void *src, size_t n);
void *memset(void *s, int c, size_t n);
void *calloc(size_t nelem, size_t elemsize);
void *realloc(void *ptr, size_t size);
```

(2) 在进行前一问的测试之前, 还在 64 位 Linux 机器上进行过如下用户代码测试(输入 mat_vec_mul 的参数都非零):

```
int *y = mat_vec_mul(A, x, n);
int z = y[0];
```

结果发生了段错误。通过 gdb 调试发现 mat_vec_mul 函数内并没有发生段错误, 但是在初始化变量 z 时发生了段错误, 后来发现是参数的输入有问题。写出出现这种错误的充分必要条件 _____ (1 分)

4. Double fault: Intel 处理器中有一种特殊的异常, 被称为 double fault。此异常发生表明调用某个故障(fault) A 的处理程序后又触发了另一个故障 B。正常情况下, 故障 B 会有相应异常处理程序来处理。因此两个故障 B 和 A 可以被顺序解决。但是如果处理器无法正常处理故障 B, 或是处理了之后依然无法处理故障 A, 就会产生 double fault, 并终止(abort)。假设除了缺页异常处理程序外, 其他异常处理程序都不会产生新的故障。如果在某次缺页故障时产生了 double fault, 其原因可能是 _____ (不定项选择, 都选对才得分, 1 分)

- ① 运行缺页故障处理程序时, CPU 上的权限位是内核态, 但所执行代码段 U/S=0
- ② 运行缺页故障处理程序时, CPU 接收到了键盘发送的 Ctrl + C 信号
- ③ 缺页故障处理程序没有加载到主存中

E11

答案:

1. ②④
2. 3 0x00615000 0x48
3. (1) 4 int *y = (int *)calloc(n, sizeof(int)); 或
int *y = (int *)Calloc(n, sizeof(int));
其中 calloc/Calloc 的两个参数只要乘起来等于 $n * \text{sizeof}(\text{int})$ 即算对
(2) $n < 0$ (或其他等价表达)
4. ③

←

←

解析:

1. 考察 TLB 的位划分与表项含义，属于简单题。

题目告知 TLB 命中且无缺页，因此只需要访问物理页，即访问一次物理内存

根据 y 的值可以得到 TLBT=0xd551, TLBI=5, 查表知 y 在 TLB 条目的内容为 0x019fa42d, R/W=0, U/S=1, 因此这一页的实质权限为只读、内核模式。

参见课本第 9.6 和 9.7 节 (P567-582)

←

2. 考察虚拟内存地址翻译，属于简单+中等题。

题目告知 TLB 未命中且无缺页，因此需要访问页目录、二级页表和物理页，共访问三次物理内存。

计算得 x 的 VPN1=1, VPN2=0, PPO=0x4d0。页目录地址在 0x00e66000，因此相应页目录中所在条目地址为 $0x00e66000 + 1 * 4 = 0x00e66004$ ，从表中按小端法读出 0x00615721，因此二级页表存在，首地址为 0x00615000，这恰好也是相应二级页表中所在条目的地址，从表中按小端法读出 0xc0ee2d21，因此物理页存在，首地址为 0xc0ee2000，因此 x 的物理地址为 $0xc0ee2000 + 0x4d0 = 0xc0ee24d0$ 。从表中读出一个字节，为 0x48。

参见课本第 9.6 和 9.7 节 (P567-582)

3. 本题考查内存安全。

(1) 考察 malloc 函数的性质，属于中等题。

malloc 函数并不会清零所分配内存区域，因此同样的输入可能会有不同的输出，但 mat_vec_mul 函数别的行都不可能产生这一效果，因此只需要将第 4 行换为

`int *y = (int *)Calloc(n, sizeof(int));`

Calloc 是一个带错误检查的、会将所分配区域置零的函数。

参见课本第 9.11.2 节正文例子 (P610)。

←

(2) 内存安全的综合考察，属于难题。

假设 malloc 函数分配成功，那么 y 不是空指针。y[0] 不可能在外部访问出现段错误，因此这种情况不成立，malloc 一定出现了问题，导致 y 是空指针。但是此时 mat_vec_mul 并没有触发段错误，说明函数内部一次对 y 的访问都没有，这只有可能循环都没有进入，因此必须有 $n < 0$ ，根据题设，参数是非零的，所以 $n < 0$ 。以上推导出了 $n < 0$ 是必要条件。下面推导 $n < 0$ 是充分条件，即任何 $n < 0$ 都会导致这样的错误。为此，只需要说明 malloc 函数一定分配失败。首先，这是 Linux 64 位系统，因此地址空间为 64 位，`size_t` 为 64 位，用户虚拟地址空间为 48 位。注意 $n * \text{sizeof}(\text{int})$ 会将 n 的类型转为 long，再转为 `unsigned long`，而 n 只有 32 位，所以符号扩展之后最高位至少有 33 个 1，乘 `sizeof(int)`，即 4 之后，最高位还有至少 29 个 1，因此 $n * \text{sizeof}(\text{int}) > 2^{63} > 2^{48}$ ，超过了用户虚拟地址空间的大小，因此无论如何，malloc 函数都会分配失败。

参见课本第 9.8 节 (P576)、9.9.1 节 (P588)、2.2.8 节 (P58-59)。

←

4. 本题考查对缺页故障、异常处理的综合理解，属于难题。

(1) 如果 CPU 权限位是内核态，它自然可以运行用户级代码，不会触发异常。

(2) 如果缺页故障处理程序运行时，收到外部中断，这不属于故障，因此不会触发。

(3) 如果缺页故障处理程序不在主存，那么调用它会触发缺页故障，于是这个故障又会导致调用缺页故障处理程序，但它仍然不在主存，故这一故障无法被解决，于是触发 double fault。

相关内容参见课本第 8.1 节 (P502-507)、9.7.2 节 (P581-582) 和 Intel 64 and IA-32 Architectures Software Developer's Manual - Volume 3: System Programming Guide 的第 6.15 节。

E12

第五题 (10 分)

某 32 位机器有 24 位地址空间，采用二级页表，一级页表中有 64 个 PTE，二级页表中有 1024 个 PTE，一级页表 4KB 对齐，PTE 最高一位是有效位，没有 TLB 和 Cache。惠楚思程序员在该机器上执行了如下代码。

```
1. int* a=malloc(10000, sizeof(int));
2. int* b = a+5000;
3. fork();
4. *b=0x80C3F110;
```

(1) 假设编译后 b 的值保存在寄存器中，请问该机器页面大小为①____字节，VPN1 长度为②____，VPN2 长度为③____。

(2) 由于该执行结果不符合预期，惠楚思需要对程序执行过程进行还原。程序执行过程中硬件记录了部分内存访问记录，但访问记录并不完整。目前已知在父进程执行第四行的时候，进行了若干次物理内存的读写操作，其中第一次读内存操作从地址 0x67F0E8 读出 0x80AA32C4，另外有两次将 0x80C3F110 写入内存的操作，写入的地址分别是 0xC3F50D 和 0xAA3AD0，但写入顺序并不清楚。程序结束后变量 b 中保存的值是④_____。在解析 “*b”的过程中，一级页表的起始地址为⑤_____；二级页表的起始地址为⑥_____；物理页面的起始地址为⑦_____。

(地址均用 16 进制表示)

E12

第五题（10分）

某 32 位机器有 24 位地址空间，采用二级页表，一级页表中有 64 个 PTE，二级页表中有 1024 个 PTE，一级页表 4KB 对齐，PTE 最高一位是有效位，没有 TLB 和 Cache。惠楚思程序员在该机器上执行了如下代码。

```
1. int* a=calloc(10000, sizeof(int));
2. int* b = a+5000;
3. fork();
4. *b=0x80C3F110;
```

(1) 假设编译后 b 的值保存在寄存器中，请问该机器页面大小为①__字节，VPN1 长度为②__，VPN2 长度为③__。

(2) 由于该执行结果不符合预期，惠楚思需要对程序执行过程进行还原。程序执行过程中硬件记录了部分内存访问记录，但访问记录并不完整。目前已知在父进程执行第四行的时候，进行了若干次物理内存的读写操作，其中第一次读内存操作从地址 0x67F0E8 读出 0x80AA32C4，另外有两次将 0x80C3F110 写入内存的操作，写入的地址分别是 0xC3F50D 和 0xAA3AD0，但写入顺序并不清楚。程序结束后变量 b 中保存的值是①_____。在解析 “*b” 的过程中，一级页表的起始地址为②_____；二级页表的起始地址为③_____；物理页面的起始地址为④_____。
(地址均用 16 进制表示)

解答题四 此题我们课上讲过了，原题有几个数据错误，条件也不够。摘取正确题目的分析如下：

1. 4096; 6; 10。送分题。
2. 0xEAB450D, 0x67F000, 0xAA3000, 0xC3F000。

考虑父进程执行 *b 的写时的读写问题。在解析 *b 时，第一步当然是找出一级页表项，因此从地址 0x67F0E8 读出 0x80AA32C4 就是从一级页表中，物理地址 0x67F0E8 的位置，读出一级页表项的内容 0x80AA32C4。根据题目对 PTE 结构的描述，我们知道 0xAA3 是二级页表页的页号，即 0xAA3000 是二级页表的起始物理地址。由于一级页表是 4KB 对齐的，所以一级页表项的物理地址就是 0x67F000。

为什么会写两次？显然，我们知道 COW 机制在起作用，这段内存首先要被复制。0x80C3F110 这个数字当然是精心设计的，必然暗藏玄机。首先，一次写肯定是写 *b，那么另一次写呢？注意这次写也是父进程，因此和子进程无关。由于 COW 后内存映射需要修改，结合后面也有类似于 0xC3F... 的物理地址，我们可以推定这个实际上就是 COW 后，b 对应的二级页表项。这样我们马上知道，其二级页表项的物理地址在 0xAA3AD0，而 b 指向的真实物理地址是 0xC3F50D（页号 0xC3F）。这时我们就可以计算出 b 的值（即虚拟地址）了。一级页表项的偏移是 $(0x67F0E8 - 0x67F000) / 4 = 0x3A$ ，二级页表项的偏移是 $(0xAA3AD0 - 0xAA3000) / 4 = 0x2B4$ ，VPO=0x50D，所以虚拟地址是它们的拼接。注意不是直接将 16 进制拼接，虚拟页号为 11 1010 10 1011 0100=0xEAB4，所以 b=0xEAB450D。此题页表项中的最低 12 位是何含义，属于未解之谜。

E1

19. 垃圾收集器的根节点不包括以下哪个节里的数据：

- A. text
- B. data
- C. bss
- D. stack

E1

19. 垃圾收集器的根节点不包括以下哪个节里的数据：

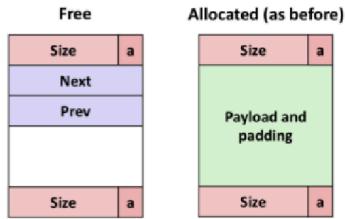
- A. text
- B. data
- C. bss
- D. stack

19. A。其他段中都可能有指针，其引用的内存需要管理。代码段中则没有这种问题。

E2

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);
```



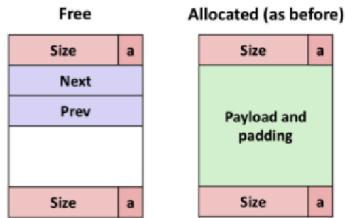
内存分配器内部使用**显式空闲链表**实现，按照地址从低到高顺序来维护空闲链表节点顺序。分配块格式参照上图。每个分配块 16 字节对齐，头部和脚部的大小都是 4 字节。分配算法采用**首次适配**算法，将适配到的空闲块的第一部分作为分配块，剩余部分变成新的空闲块，并采用**立即合并**策略。假设初始空闲块的大小是 1KB。那么以上调用序列完成后，分配器管理的这 1KB 内存区域中，**内部碎片**的总大小是____，链表里**第一个空闲块**的大小是____。

- A. 58Byte, 848Byte
- B. 26Byte, 32Byte
- C. 74Byte, 48Byte
- D. 74Byte, 16Byte

E2

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

12. D。直接模拟就可以。

Notices

THANKS

Made by WalkerCH

changxinhai@stu.pku.edu.cn

Reference: [Weicheng Lin]'s presentation.

Reference: [Arthals]'s templates and content.

