# 第二次作业

作业发布时间：2025/05/23 星期五

本次作业要求如下：

**1.截止日期：2025/06/1 周日晚24:00**

2.**提交作业给本课程QQ群里的助教（庄养浩：QQ1473889198）**

3.**命名格式**：附件和邮件命名统一为“第三次作业+学号+姓名”， **PDF 格式**；

4.注意：

(1) 选择、判断和填空只需要写答案，大题要求有详细过程，**过程算分**。

(2) 大题的过程最好在纸上写了拍照，放到word里。

**(3) 本次作业由Part1和Part2两部分，需要全部作答**

5.本次作业遇到问题请联系课程群里的助教。

Part1 存储器层次结构+虚拟内存

**一．填空题**

1. 对于一个磁盘，其平均旋转速率是 15000 RPM，平均寻道时间是 4ms，单个磁道上平均扇区数量是 800，则这个磁盘的平均访问时间是 \_\_\_6.005ms\_\_\_\_
2. 对于一个磁盘，其有两个扇片，10000个柱面，每个磁道平均有400个扇区，而每个扇区平均有 512 个字节。那么这个磁盘的的容量为 \_\_8.192GB\_\_\_\_\_\_
3. Cache为8路的2M容量，B=64，则其Cache组的位数s= \_\_12\_\_\_\_\_
4. 在现代计算机存储层次体系中，访问速度最快的是 \_\_寄存器\_\_\_\_\_
5. 若高速缓存的块大小为 B(B>8)字节，向量 v 的元素为 int，则对 v 的步长为 1 的应用的不命中率为\_\_4/B\_\_\_
6. 某CPU使用32位虚拟地址和4KB大小的页时，需要PTE的数量是 \_\_2^20\_\_\_\_\_（不考虑多级页表情况）
7. 缓存不命中的种类有 \_\_冷不命中\_\_\_\_ 、 \_冲突不命中\_\_\_\_\_\_、 \_容量不命中\_\_\_\_\_\_\_\_\_ 。
8. 虚拟页面的状态有 \_\_未分配\_\_\_\_\_\_、已缓存、未缓存共3种。
9. Linux虚拟内存区域可以映射到普通文件和 \_\_\_匿名文件\_\_\_\_\_，这两种类型的对象中的一种。
10. 虚拟内存发生缺页时，缺页中断是由 \_\_\_\_mmu\_\_\_\_ 触发的。

**二．分析题**

1. 对于一个机器而言，有如下的假设，内存是字节寻址，并且内存访问是1字节的字。地址宽度是 13 位，其高速缓存是 2路组相联的，块大小是 4 字节，一共有 8个组。高速缓存的具体内容如图所示。



* 1. 对于该地址格式进行划分，划分出 tag 位，组索引位和块内偏移的区间。

前8位tag

中间3位index

后2位offset

* 1. 对于地址 0x0E34 ，指出其对应的tag位，组索引位和块内偏移的值，并说明高速缓存是否命中，如果命中，写出对应的字节（用16进制表示）。

tag:01110001

index:101

offset:00

命中，字节：0B

* 1. 对于地址0x0DD5，指出其对应的tag位，组索引位和块内偏移的值，并说明高速缓存是否命中，如果命中，写出对应的字节（用16进制表示）。

Tag:01101110

Index:101

Offset:01

不命中

1. 对于一个直接映射的高速缓存系统，假设其大小是 256 字节，块大小是 16字节，现在定义三个操作，L 为装载操作，S 为数据存储操作，M 为数据更改操作。L 和 S 最多引发一次缓存miss，而 M 操作可以看作是对于同一个地址先进行了 L 操作，之后进行了 S 操作。针对下面的操作序列，写出每次操作时高速缓存命中与否以及淘汰替换与否的情况。（假设高速缓存最开始是空的）

高速缓存命中 是否淘汰替换

L 10，1。 不命中 不替换

M 20，1。 不命中，命中 不替换

L 22，1。 命中 不替换

S 18，1。 命中 不替换

L 110，1。 不命中 替换

L 210，1 不命中 替换

M 12，1。 不命中，命中 替换

说明：L 10，1 表示对于地址 0x10 位置，进行了一个字节的装载操作。

1. 在一台具有块大小16字节（B=16）、整个大小为1024字节的直接映射数据缓存的机器上测量如下代码的高速缓存性能：

假设：

* sizeof(int) = 4。
* grid从内存地址0开始。
* 这个高速缓存开始时是空的。
* 唯一的内存访问是对数组grid的元素的访问，变量i、j、total\_x和total\_y存放在寄存器中。
* 数据结构定义

struct position {

int x;

int y;

};

struct position grid[16][16];

int total\_x = 0, total\_y = 0;

int i, j;

1. Test 1

for (i = 0; i < 16; i++){

for(j = 0; j < 16; j++){

total\_x += grid[i][j].x;

}

}

for (i = 0; i < 16; i++){

for(j = 0; j < 16; j++){

total\_y += grid[i][j].y;

}

}

1. 读总数是多少？

16\*16\*2=512

1. 缓存不命中的读总数是多少？

8\*8\*2\*2=256

1. 不命中率是多少？

256/512=50%

1. Test 2

for (i = 0; i < 16; i++){

for(j = 0; j < 16; j++){

total\_x += grid[i][j].x;

total\_y += grid[i][j].y;

}

}

1. 读总数是多少？

16\*16\*2=512

1. 缓存不命中的读总数是多少？

8\*8\*2=128

1. 不命中率是多少？

128/512=25%

1. 如果高速缓存有两倍大，那么不命中率是多少？

总访问次数：512

不命中访问次数：8\*16=128

不命中率：128/512=25%

1. 对于一个地址，高速缓存通常使用地址的中间部分作为组索引，为什么不用高位地址作为组索引？

因为程序具有空间局部性，如果使用高位地址作为索引，连续地址的高位通常相同（例如数组基地址），导致所有连续块映射到同一缓存组，引发频繁的冲突不命中。

1. 下图展示了一个虚拟地址的访存过程，每个步骤都用不同的字母表示。请针对下面不同的情况，用字母序列表示不同情况下的执行流程。
2. TLB 命中，缓存物理地址命中。

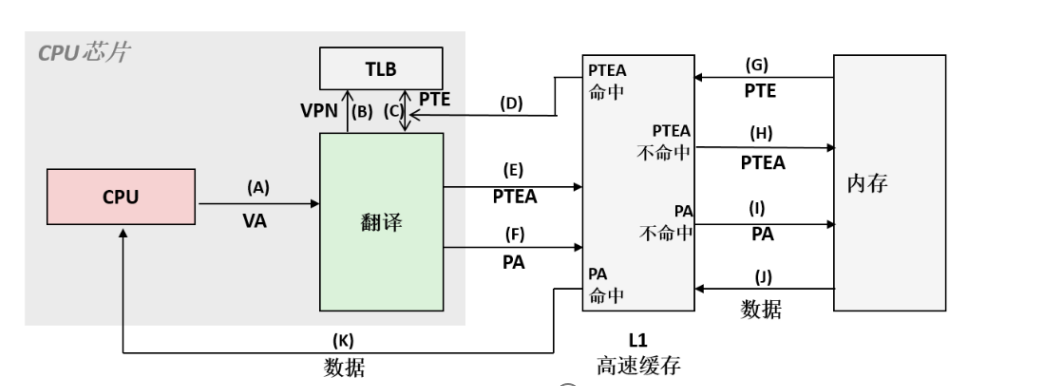
ABCFK

1. TLB 不命中，缓存页表命中，缓存物理地址命中。

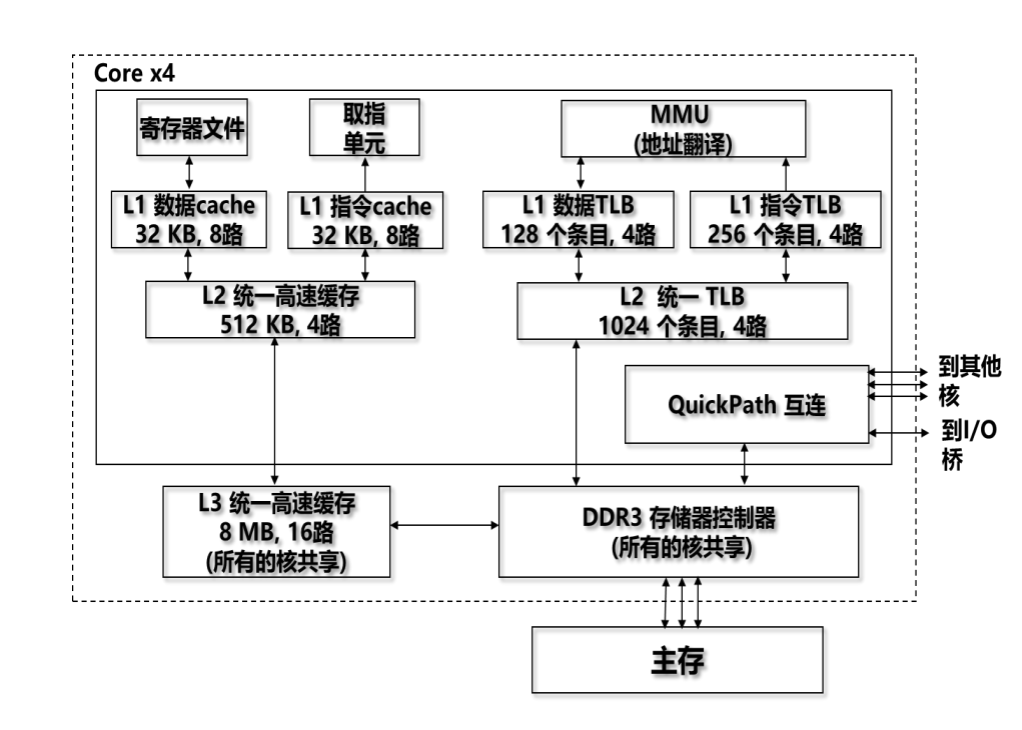
ABEDFK

1. TLB 不命中，缓存页表不命中，缓存物理地址不命中。

ABEHGDFIJK



1. Intel I7 处理器的虚拟地址为 48 位。虚拟内存的页大小是 4KB，物理地址为 52 位，Cache 块大小为 64B。物理内存按照字节寻址。其内部结构如下图所示，依据这个结构，回答问题。



1. 虚拟地址的 VPN 占多少位？一级页表占多少项？L1数据TLB的组索引位数 TLBI 占多少位？

36

2^9

5

1. L1 数据 Cache 有多少组，相应的 Tag 位，组索引位和块内偏移位分别是多少？

2^6

40

6

6

1. 对于某指令，其访问的虚拟地址为 0x804849B，则该地址对应的 VPO 为多少？对应的L1 TLBI 位为多少？（用 16进制表示）

0x49B

0x08

1. 对于某指令，其访问的物理地址为 0x804849B，则该地址访问 L1 Cache 时，CT 位为多少？CO 位为多少？（用16进制表示）

0x0000008048

0x1B

Part2 链接器+异常处理+I/O

**一．选择题**

1. 链接时两个文件同名的弱符号，以（ C ）为基准
2. 连接时先出现的
3. 连接时后出现的
4. 任一个
5. 链接报错
6. 链接时两个同名的强符号，以哪种方式处理？（ D ）
7. 链接时先出现的符号为准
8. 链接时后出现的符号为准
9. 任一个符号为准
10. 链接报错
11. 以下关于程序中链接“符号”的陈述，错误的是（ B ）
12. 赋初值的非静态全局变量是全局强符号
13. 赋初值的静态全局变量是全局强符号
14. 未赋初值的非静态全局变量是全局弱符号
15. 未赋初值的静态全局变量是本地符号
16. C 源文件 m1.c 和 m2.c 的代码分别如下所示，编译链接生成可执行文件后

执行，结果最可能为 (D )

$ gcc –o a.out m2.c m1.c ; ./a.out

0x1083020 ； ；

A. 0x1083018, 0x108301c B. 0x1083028, 0x1083024

C. 0x1083024, 0x1083028 D. 0x108301c, 0x1083018

|  |  |
| --- | --- |
| // m1.c  #include <stdio.h>  int a1 ;  int a2 = 2 ;  extern int a4 ;  void hello()  {  printf("%p;", &a1);  printf("%p;", &a2);  printf("%p\n", &a4);  } | //m2.c  int a4 = 10 ;  int main()  {  extern void hello() ;  hello() ;  return 0 ;  } |

1. 对于以下一段代码，可能的输出为：A

|  |
| --- |
| int count = 0;  int pid = fork();  if (pid == 0){  printf("count = %d\n",--count);  }  else{  printf("count = %d\n",++count);  }  printf("count = %d\n",++count); |

A.1 2 -1 0 B.0 0 -1 1

C.1 -1 0 0 D.0 -1 1 2

1. Linux进程终止的原因可能是( D )

A.收到一个信号 B.从主程序返回 C.执行exit函数 D.以上都是

**二．填空题**

1. C 语句中的全局变量，在\_\_链接\_\_\_\_\_\_阶段被定位到一个确定的内存地址。
2. 子程序运行结束会向父进程发送\_\_\_SIGCHLD\_\_\_\_\_信号。
3. 向指定进程发送信号的linux命令是\_\_/bin/kill\_\_\_\_\_\_。

**三．分析题**

请阅读以下程序，然后回答问题（假设程序中的函数调用都可以正确执行）：

|  |
| --- |
| int main() {  printf("A\n");  if (fork() == 0) {  printf("B\n");  }else {  printf("C\n");  A  }  printf(“D\n"); exit(0);  } |

1. 如果程序中的 A 位置的代码为空，列出所有可能的输出结果：

一：

A

B

D

C

D

二：

A

C

D

B

D

三：

A

B

C

D

D

四：

A

C

B

D

D

1. 如果程序中的 A 位置的代码为：

waitpid(-1, NULL, 0);

列出所有可能的输出结果：

一：

A

B

D

C

D

二：

A

B

C

D

D

三：

A

C

B

D

D

1. 如果程序中的 A 位置的代码为：

printf(“E\n”);

列出所有可能的输出结果：

ABDCED

ACBDED

ACEBDD

ACEDBD

ABCDED

ABCEDD

ACBEDD