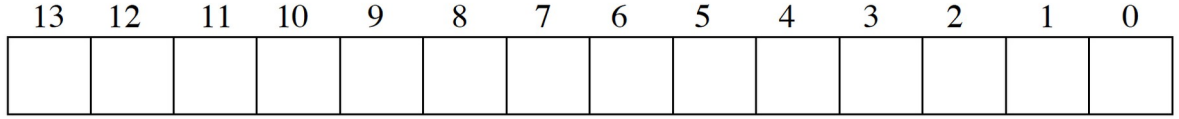
ICS2-homwork5

问题 1

- 根据 9.6.4 的示例内存程序，给定 virtual address 0x03FC，完成下列表格

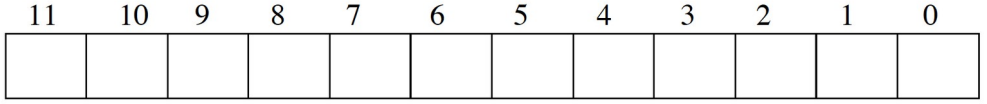
A. Virtual address format



B. Address translation

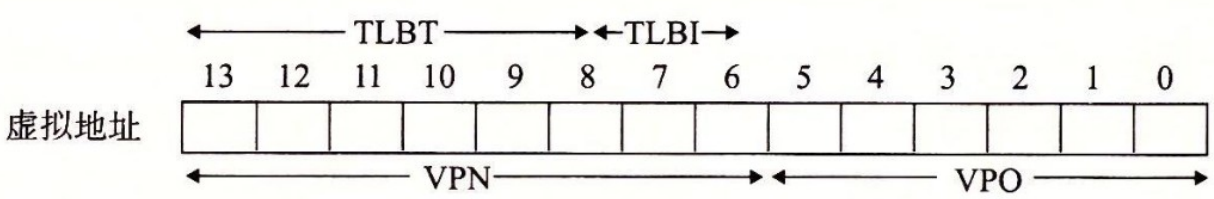
Parameter	Value
VPN	
TLB index	
TLB tag	
TLB hit?(Y/N)	
Page fault?(Y/N)	
PPN	

C. Physical address format



D. Physical memory reference

Parameter	Value
Byte offset	
Cache index	
Cache tag	
Cache hit?(Y/N)	
Cache byte returned	



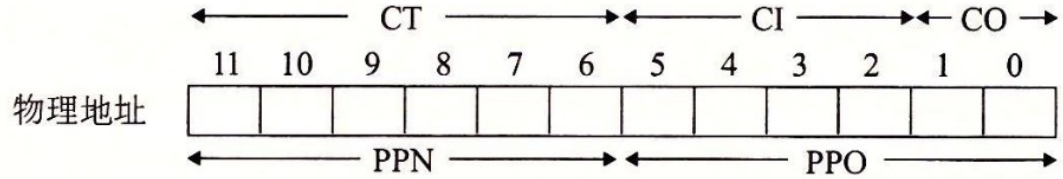
位	标记位	PPN	有效位	标记位	PPN	有效位	标记位	PPN	有效位	标记位	PPN	有效位
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

a）TLB：四组，16 个条目，四路组相联

VPN	PPN	有效位
00	28	1
01	—	0
02	33	1
03	02	1
04	—	0
05	16	1
06	—	0
07	—	0

VPN	PPN	有效位
08	13	1
09	17	1
0A	09	1
0B	–	0
0C	–	0
0D	2D	1
0E	11	1
0F	0D	1

b）页表：只展示了前 16 个 PTE



索引	标记位	有效位	块 0	块 1	块 2	块 3
0	19	1	99	11	23	11
1	15	0	—	—	—	—
2	1B	1	00	02	04	08
3	36	0	—	—	—	—
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	—	—	—	—
7	16	1	11	C2	DF	03
8	24	1	3A	00	51	89
9	2D	0	—	—	—	—
A	2D	1	93	15	DA	3B
B	0B	0	—	—	—	—
C	12	0	—	—	—	—
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	—	—	—	—

c）高速缓存：16 个组，4 字节的块，直接映射

问题 2

- 计算机有 64 位虚拟地址空间， page size 是 2048B ， page table entry 长度是 4B 。用 multi-level page table 把所有页面放在表中，需要多少 levels ？ 写出解题步骤。注意页表本身也是以 page 为单位进行空间分配和管理的，其中第一级页表是一个完整的 page 。

问题 3

- 一个计算机虚拟内存空间大小为 1MB ，物理内存空间大小为 8MB 。虚存空间依次分为 4 个 segment （ code, data, heap, stack ），采用 segmentation 内存管理，用 Va 最高两位标记 segment ，有寄存器记录下列信息。请根据给定程序回答问题。

segment	base	size	positive
code	4MB	4KB	1
data	2MB	2KB	1
heap	7MB	8KB	1
stack	1MB	256KB	0

问题 3

1. 程序涉及到的数据结构被存放在了哪一段？请画图标注。
2. 程序输出了 s=0xC1300，求 s 对应的 PA。如果 VA=258K/514K，对应的 PA 是多少？写出求解过程。

```c
#include <unistd.h>
#include <sys/mman.h>
#include <stdio.h>
#define ARRAY_LEN (128*1024)
#define SIZE   100
int a = 0;
int b = 1;

int main(void) {
    int idx;

    char s[ARRAY_LEN];
    printf("s=%#X\n", &s);

    int fd = open("test.txt", O_RDWR);
    read(fd, s, ARRAY_LEN);

    close(fd);
    return 0;
}
```

问题 4

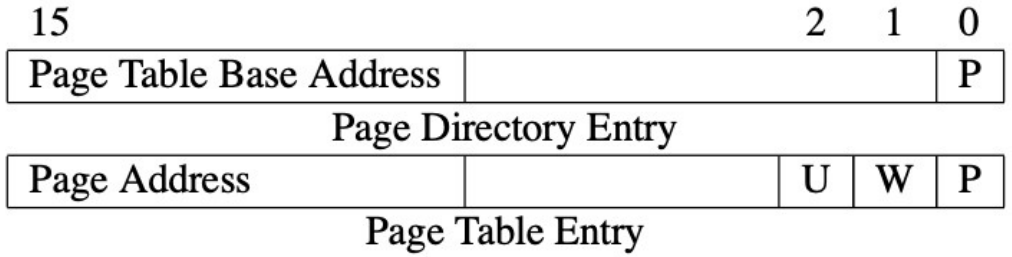
- The following problem concerns virtual memory and the way virtual addresses are translated into physical addresses. Below are the specifications of the system on which the translation occurs.
- The system is a 16-bit machine - words are 2 bytes.
- Memory is byte addressable.
- The maximum size of a virtual address space is 64KB.
- The system is configured with 16KB of physical memory.
- The page size is 64 bytes.
- The system uses a two-level page tables. Tables at both levels are 64 bytes (1 page) and entries in both tables are 2 bytes as shown below.

问题 4

In this problem, you are given parts of a memory dump of this system running 2 processes. In each part of this question, one of the processes will issue a single memory operation (read or write of one byte) to a single virtual address (as indicated in each part). Your job is to figure out which physical addresses are accessed by the process if any, or determine if an error is encountered.

Entries in the first and second level tables have in their low-order bits flags denoting various access permissions.

- P=1⇒Present
- W = 1 ⇒ Writable (applies both in kernel and user mode)
- U = 1 ⇒ User-mode



The contents of relevant sections of memory is shown on the next page. All numbers are given in **hexadecimal**.

For the purposes of this problem, omitted entries have contents = 0.

Address	Contents
0118	2381
0130	2101
0160	2281
018E	1581
019C	1201
01B8	1A01
120A	2701
1214	27C1
1228	2741
158A	25C1
1594	2541
15A8	2501
1A0A	2041
1A14	20C1
1A28	2081
2106	3FC7
210C	3A47
2118	3587
2286	3107
228C	3447
2298	3007
2386	33C7
238C	3887
2398	3247

问题 4

- Process 1 is a process in **user** mode (e.g. executing part of `main()`) and has page directory base address `0x0100`.
  Process 2 is a process in **kernel** mode (e.g. executing a `read()` system call) and has page directory base address `0x0180`.
- For each of the following memory accesses, first calculate and fill in the address of the page directory entry and the page table entry. Then, if the lookup is successful, give the physical address accessed. Otherwise, circle the reason for the failure and give the address of the table entry causing the failure. You may use the 16-bit breakdown table if you wish, but you are not required to fill it in.

问题 4

1. Process 1 writes to `0xC1B2`.
   Scratch space:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

(a) Address of PDE: 0x

(b) Address of PTE: 0x

(c) The result of the address translation is (write NONE if the translation does not result in a valid address): 0x

(d) The result of the access is (circle EXACTLY one):

success / page not present / page not writable / illegal non-supervisor access

问题 4

2. Process 2 writes to `0x728F`. Scratch space:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

(a) Address of PDE: 0x

(b) Address of PTE: 0x

(c) The result of the address translation is (write NONE if the translation does not result in a valid address): 0x

(d) The result of the access is (circle EXACTLY one):

success / page not present / page not writable / illegal non-supervisor access