计算机系统概论（2024 秋）作业 4

本周作业只有两道超级大题。

1) 虚拟内存映射

假设你获得了一个卡片计算机，它的计算能力略强于算盘，但略弱于紫荆宿舍的洗衣机。它的内存系统参数如下：

- 每个字节 8-bit，物理内存大小 4MiB = $4 \times 1024 \times 1024$ B
- 虚拟内存地址长度 32-bit
- 页大小 4KiB = 4×1024 B

1.1) 静态参数的设置

请填写以下参数（使用十进制数字即可）：

1. 如果它的物理地址正好可以编址整个物理内存，那么物理地址的长度是 ____22 bit，物理地址页内偏移（PPO）的长度是 ____12 bit，物理页号（PPN）的长度是 ____10 bit。
2. 因此，为了和物理地址的参数匹配，使地址翻译能够正常执行，虚拟地址页内偏移（VPO）的长度应该是 ____12 bit，虚拟页号（VPN）的长度是 ____20 bit。

1.2) 地址翻译

卡片计算机上的某程序访问 `0xDEADBEEF`，需要经过地址翻译，请回答（数字请填写 16 进制）：

1. 查询的 VPN 是 `0x`_____。 `0xDEADB`
2. 如果页表中存在一项匹配了查询的 VPN，页表项中的 PPN 是 `0x233`，那么翻译结束后访问的物理地址是 `0x`_____。 `0x233EEF`
3. 如果页表中不存在匹配的 VPN，那么 ____： C

- **A**. 不会发生异常，地址将不会被翻译，直接用于访问物理内存
- **B**. 不会发生异常，硬件将填充 TLB。
- **C**. 会发生缺页异常（Page fault）
- **D**. 会发生 TLB 缺失异常

1.3) 快表（TLB）

卡片计算机神奇地拥有快表。假设其快表组织形式为 2-way，最多存储 16 条目。请回答以下问题，可以使用 10 进制（无前缀，如 `13`），16 进制（以 `0x` 前缀，如 `0xD`）或者二进制（以 `0b` 前缀，如 `0b1101`）：

1. TLB Index（TLBI）长度为 ____3 bit，TLB Tag（TLBT）长度为 ____17 bit。
2. 上述对 `0xDEADBEEF` 的访问中，TLBI 为 _____`0b011`，TLBT 为 _____`0b11011110101011011`。
3. 为了使 TLB 访问命中， ___E 的 TLBT 应该与虚拟地址中的 TLBT 匹配。
   - **A**. TLB 中任意一个有效条目
   - **B**. TLB 中所有有效条目
   - **C**. TLB 中任意 Set 中的所有有效条目
   - **D**. TLB 中所有 Set 中的均有任意有效条目
   - **E**. TLB 中由 TLBI 索引的 Set 中的任意有效条目
   - **F**. TLB 中由 TLBI 索引的 Set 中的所有有效条目

3. 如果 TLB 没有命中，那么 ____： B

- **A.** 不会发生异常，地址将不会被翻译，直接用于访问物理内存
- **B.** 不会发生异常，硬件将填充 TLB。
- **C.** 会发生缺页异常（Page fault）
- **D.** 会发生 TLB 缺失异常

1.4) 懒惰文件映射

Linux （及类 Unix 系统）提供了 `mmap` 系统调用，用于将一个文件的内容直接映射到一块内存中。这一文件映射是懒惰的，即只有在第一次真正访问某个被映射的文件页时才会将对应的文件页的内容读入内存。假设某程序依次进行了如下操作：

- 调用 `mmap` 将文件 `foo.txt` 映射到虚拟地址 `0x10000000` 处。文件长度 1MiB。
- 读取 `0x10000008` 处的字节。
- 读取 `0x10000016` 处的字节。
- 读取 `0x10002016` 处的字节。
- 调用 `munmap` 将文件映射解除。

在这个过程中，发生了 ____2 次 Page fault，操作系统总共修改了 ____3 次页表（同时修改多项算作一次）。`foo.txt` 文件总共被读取了 ____2 次，共读取了 ____8KiB 字节。

1.5) 硬件查询页表过程，及走向多级页表

卡片计算机硬件上页表也储存在物理内存中，处理页表查询的方法通常是根据一个基地址，并将 VPN 作为索引，如下**伪代码**：

```
// PTE = Page Table Entry 页表项
PTE page_table[1 << VPN_LEN];

uint32_t get_ppn(uint32_t vpn) {
  return page_table[vpn].ppn;
}
```

与助教商议后，确定这儿仅包括该指令本身的访存次数~

卡片计算机每从内存读取一个页表项需要一次内存访问，那么每个内存访问指令被执行时，**不考虑读取指令字，不考虑数据缓存**，至多需要访问 ____2 次内存，至少访问 ____1 次内存。页表的基地址应该储存为 _____A：

- **A.** 物理地址
- **B.** 虚拟地址

如果我们假设 PTE 的大小为 4-byte (32-bit)，那么在卡片计算机中，页表的大小将会占用内存大小为 _____4MiB。这太大了！为了节省内存，真实硬件引入了一种多级页表机制，将页表存储为树形结构。感兴趣的同学可以参考课本 CSAPP 第三版 9.6.3 一节。

2) 链接和重定位

考虑以下 C 程序，在 C11 标准下编译，编译选项 `-O0`，输出了 `foo.o` 文件，后续和其他文件链接得到可执行文件 `bar`：

```c
// foo.c
extern void (*test1)();

static void test2() {}

void test3() {}

extern void test4();

void test_call(void (*test5)()) {
    test1();
    test2();
    test3();
    test4();
    test5();
}
```

你可以在实验集群上尝试这一过程，编译命令为：

```
gcc -c -o foo.o -O0 foo.c
```

2.1) 请列出 `test_call` 中的五个函数调用，在链接期间需要全局重定位参与的调用有哪些，并简短解释原因。

（即生成的 foo.o 文件里这些调用中，需要重定位的符号有哪些）

`test1`，`test3`，`test4`。参考 6.1-与 c 语言-5.pdf 课件中 P25。

`test2` 是 static 的，`test5` 是局部变量。这两个调用在编译时即可确定，test1 是全局变量，test3，test4 是全局函数，需要在链接时才能确定。

如果回答没有 `test3` 但是言之有理也可：`test3` 可能被内联。

2.2) 请列出 `foo.o` 文件中 `.symtab` 中必须以全局符号存储的符号名，并简短解释原因。

（即生成的 foo.o 文件中，所有可能参与后续链接的全局函数及变量符号。你只需要列出上述源代码中出现的变量及函数即可）

你可以使用 `readelf -a foo.o` 查看符号表，注意输出中 `.symtab` 部分。

`test1`, `test3`, `test4`, `test_call`

2.3) 最终可执行文件链接外部变量、函数所处的对象文件时，可以使用静态链接也可以使用动态链接。在运行时谁更有性能优势？请简要解释原因。

静态链接库有优势，因为访问这些变量函数时无需穿过 GOT，减少一次内存访问。