Project 4 Virtual Memory 设计文档

中国科学院大学
郑旭舟
2021 年 1 月 28 日

1. 内存管理设计

我实现的是 S-Core，因此不涉及 TLB Miss 和 Page Fault，仅完成：①页表项初始化②根据页表项填充 TLB MMU.

1.1. 页表项初始化

由 S-Core 的要求，采用一级页表，使用的数据结构是 pte_t 组成的线性表。

其中，表项的类型定义为：

```
typedef struct pte {
  uint32_t flag_eo, // even or odd
      valid;
  uint64_t entryhi, entrylo;
} pte_t; // 4*2+8*2=24B
```

因为一对相邻的奇偶 pte_t 对应 TLB 中的一个表项（一对奇偶页），因此只要 PT_SIZE 大于或等于 TLB 表项数×2 就能满足 S-Core 的运行要求。

该页表放在一个 unmapped 区域，并且经过计算不会与用户栈/内核栈冲突：

```
#define PTE_BASE_ADDR 0xffffffffa0ffd000
```

它自身占据的大小是$24\text{B} \times 128 = 3\text{KB}$，索引的物理空间为$128 \times 4\text{KB} = 0.5\text{MB}$.

1.2. TLB 表项初始化

如果 TLB 表项在 initial 阶段通过 TLBWI 指令而非 TLB Refill 填充，则只需要在每次 TLBWI 之前向协处理器寄存器写入响应的值即可。

2. 关键函数功能

2.1. init_page_table

```
void init_page_table() {
  uint64_t i;
  for (i = 0; i < 64; i++) {
   // 根据设计，计算填充页表项操作过程中的参数值
    uint64_t enhi = (i << 13); // |(asid&0xff) 忽略进程
    uint64_t dpfn = (i << 1) + 0x8000;
    uint64_t enl0 =
        (dpfn << 6) | (2 << 3) | (1 << 2) | (1 << 1) | (1); // d-pfn/c/d/v/g
    uint64_t enl1 = enl0 | (1 << 6);                        // d-pfn/c/d/v/g
   // 预备当前页表项
```

```
    pte_t *pte_even, *pte_odd;
    pte_even = (pte_t *)(PTE_BASE_ADDR + (i * 2) * sizeof(pte_t));
    pte_odd = (pte_t *)(PTE_BASE_ADDR + (i * 2 + 1) * sizeof(pte_t));
    // 奇偶
    pte_even->flag_eo = 0;
    pte_odd->flag_eo = 1;
    // valid 位
    pte_even->valid = pte_odd->valid = 1;
    // entryhi/entrylo*寄存器
   pte_even->entryhi = pte_odd->entryhi = enhi;
    pte_even->entrylo = enl0;
    pte_odd->entrylo = enl1;
  }
}
```

2.2. init_tlb_entry

```
void init_tlb_entry(void) {
  for (uint64_t i = 0; i < 64; i++) {
   // 取出当前页表项
    pte_t *pte_even, *pte_odd;
    pte_even = (pte_t *)(PTE_BASE_ADDR + (i * 2) * sizeof(pte_t));
    pte_odd = (pte_t *)(PTE_BASE_ADDR + (i * 2 + 1) * sizeof(pte_t));

    // 为 TLBWI 操作设置参数，存于 CP0 寄存器
    set_cp0_entryhi(pte_even->entryhi);
    set_cp0_entrylo0(pte_even->entrylo);
    set_cp0_entrylo1(pte_odd->entrylo);
    set_cp0_pagemask(0);
    set_cp0_index(i);
    // TLBWI
    tlbwi_operation();
  }
}
```