Lab 2

思考题

Thinking 2.1

均为虚拟地址

Thinking 2.2

用宏来实现链表，可以实现原生C语言中不支持的泛型效果，可以支持任意元素类型的链表操作。

	头部插入/删除	尾部插入/删除	指定节点前插入/指定节点删除	指定节点后插入
单向链表	$O(1)$	$O(n)$	$O(n)$	$O(1)$
循环链表	$O(1)$	$O(1)$	$O(1)$	$O(1)$
双向链表	$O(1)$	$O(n)$	$O(1)$	$O(1)$

Thinking 2.3

C

Thinking 2.4

- ASID 可用来唯一标识进程，并为进程提供地址空间保护。ASID 允许 TLB 同时包含多个进程的条目，如果 TLB 不支持独立的 ASID，每次选择一个页表时，TLB 就必须被冲刷（flushed）或删除，以确保下一个进程不会使用错误的地址转换。所以有 ASID 就不用每次切换进程都要 flush 所有 TLB。
- R3000 中 ASID 为 6 位，因此可容纳 $2^6 = 64$ 个不同地址空间。

Thinking 2.5

- tlb_invalidate 调用 tlb_out 。
- 使特定虚拟地址在TLB中的旧表项失效。
- 逐行解释如下：

```
LEAF(tlb_out)
.set noreorder
    mfc0    t0, CP0_ENTRYHI /* 保存 EntryHi 寄存器原数值 */
    mtc0    a0, CP0_ENTRYHI /* 将函数参数传入 EntryHi */
    nop
    /* Step 1: Use 'tlbp' to probe TLB entry */
```

```
    /* Exercise 2.8: Your code here. (1/2) */
    tlbp /* 根据 EntryHi 中的 Key 查找对应的旧表项，将表项的索引存入 Index */
    nop
    /* Step 2: Fetch the probe result from CP0.Index */
    mfc0    t1, CP0_INDEX /* 读取 Index 中的查询结果 */
.set reorder
    bltz    t1, NO_SUCH_ENTRY /* 如果未查到，Index最高位被置1，结果小于 0，跳转到结尾*/
.set noreorder
    mtc0    zero, CP0_ENTRYHI /* EntryHi 清零 */
    mtc0    zero, CP0_ENTRYLO0 /* EntryLo 清零 */
    nop
    /* Step 3: Use 'tlbwi' to write CP0.EntryHi/Lo into TLB at CP0.Index  */
    /* Exercise 2.8: Your code here. (2/2) */
    tlbwi /* 将已经清零的 EntryHi 和 EntryLo 中的值(也就是 0)写入索引指定的表项 */

.set reorder

NO_SUCH_ENTRY:
    mtc0    t0, CP0_ENTRYHI /* 恢复原 EntryHi */
    j       ra /* return */
END(tlb_out)
```

Thinking A.1

- 三级页表页目录的基地址：$PT_{base}|(PT_{base} >> 9)|(PT_{base} >> 18)$
- 映射到页目录自身的页目录项（自映射）：
  $PT_{base}|(PT_{base} >> 9)|(PT_{base} >> 18)|(PT_{base} >> 27)$

Thinking 2.6

- x86架构的内存管理机制分为两部分：**分段机制和分页机制**。分段机制为程序提供彼此隔离的代码区域、数据区域、栈区域，从而避免了同一个处理器上运行的多个程序互相影响。分页机制实现了传统的按需分页、虚拟内存机制，可以将程序的执行环境按需映射到物理内存。此外，分页机制还可以用于提供多任务的隔离。

  分段机制将内存划分成以基地址（Base）和长度（Limit）描述的块。段可以与程序最基本的元素联系起来，程序可以简单地划分为代码、数据和栈，段机制就有相应的代码段、数据段和栈段。

  分段机制由 4 个基本部分构成：逻辑地址、段选择寄存器、段描述符和段描述符表。其核心思想是：使用段描述符描述段的基地址、长度以及各种属性。当程序使用逻辑地址访问内存的某个部分时，CPU 通过逻辑地址中的段选择符索引段描述符表以得到该内存对应的段描述符。并检验程序的访问是否合法，如合法，根据段描述符中的基地址将逻辑地址转换为线性地址。

  分段机制的目的是将内存中的线性地址空间划分成以基地址和长度描述的多个段进行管理，程序对应的逻辑地址以基地址和偏移量来描述，实现逻辑地址到线性地址空间的映射。而分页机制是使用单位“页”来管理线性地址空间和物理地址空间的映射关系。同时，分页机制允许一个页面存放在物理内存中或磁盘的交换区域中，程序可以使用比机器物理内存更大的内存区域，从而实现现代操作系统中虚拟内存机制。

  分页机制的核心思想是通过页表将线性地址转换为物理地址，并配合旁路转换缓冲区（Translation Lookaside Buffer，后面简称为TLB）来加速地址转换的过程。分页机制主要由**页表**、**CR3寄存器**和**TLB**三个部件构成。

- 存在一种额外的地址——**逻辑地址。**

  逻辑地址是程序直接使用的地址，由一个16位的段选择符和一个32位的偏移量（32位平台）构成。

难点分析

宏部分仍为简单的 c 语言知识，较为轻松

难点在于对页表的理解、如何正确的读取访问页表、如何将虚拟地址与物理地址相转化，以及 TLB 相关理论不熟悉。

心得体会

因为对页表访问具体方法的不了解，exam 花了 1.5h 才完成，extra 思路已理清但剩下一个函数没写完。

在课下还是需要尽可能掌握任务内容的细节与原理，而不是过了测试就抛下不看。