# Lab 2 实验报告

## Thinking 2.1

> *请根据上述说明，回答问题：在编写的 C 程序中，指针变量中存储的地址 是虚拟地址，还是物理地址？MIPS 汇编程序中 lw 和 sw 使用的是虚拟地址，还是物理地址？*

都是虚拟地址

## Thinking 2.2

> *请思考下述两个问题：*
>
> - *从可重用性的角度，阐述用宏来实现链表的好处。*
> - *查看实验环境中的 /usr/include/sys/queue.h，了解其中单向链表与循环链表的实 现，比较它们与本实验中使用的双向链表，分析三者在插入与删除操作上的性能差 异。*

使用宏实现函数，可以应用于不同的数据类型，减少冗余代码的出现

在插入、删除操作上，三者都是 $O(1)$ 复杂度

## Thinking 2.3

C

## Thinking 2.4

> *请思考下面两个问题：*
>
> - *请阅读上面有关 R3000-TLB 的描述，从虚拟内存的实现角度，阐述 ASID 的必要性。*

> - *请阅读《IDT R30xx Family Software Reference Manual》的 Chapter 6，结合 ASID 段的位数，说明 R3000 中可容纳不同的地址空间的最大数量。*

ASID 是为了解决多进程共享统一物理内存时，由于虚拟地址相同而导致的 TLB 冲突问题。ASID 可以为每一个进程分配唯一的标识符，这样在进程切换时，只需要切换 ASID 即可，不需要清空 TLB 中全部条目，从而提高了命中率与系统性能。

ASID 段为 6 位，说明可容纳不同地址空间的最大数量为 6

## Thinking 2.5

> *请回答下述三个问题：*
>
> - *tlb_invalidate 和 tlb_out 的调用关系？*
> - *请用一句话概括 tlb_invalidate 的作用。*
> - *逐行解释 tlb_out 中的汇编代码。*

tlb_invalidate 调用 tlb_out 以清除 TLB 中指定的条目。这些指令在改变页表时使用，以确保 TLB 中不会包含过时的映射

tlb_invalidate 是一个函数，用于使 TLB 失效。

```
1  LEAF(tlb_out)
2  .set noreorder
3      /* save EntryHi */
4      mfc0    t0, CP0_ENTRYHI
5      /* load a0 to EntryHi */
6      mtc0    a0, CP0_ENTRYHI
7      nop
8      /* Step 1: Use 'tlbp' to probe TLB entry */
9      tlbp
10     nop
11     /* Step 2: Fetch the probe result from CP0.Index */
12     mfc0    t1, CP0_INDEX
13     /* 设置指令重排 */
14 .set reorder
15     /* Exception if Index < 0 */
16     bltz    t1, NO_SUCH_ENTRY
17     /* 关闭指令重排 */
```

```
.set noreorder
    /* 清零 */
    mtc0    zero, CP0_ENTRYHI
    mtc0    zero, CP0_ENTRYLO0
    nop
    /* Step 3: Use 'tlbwi' to write CP0.EntryHi/Lo into TLB at CP0.Index  */
    tlbwi
    nop
```

## Thinking 2.6

> *任选下述二者之一回答：*
>
> - *简单了解并叙述X86 体系结构中的内存管理机制，比较X86 和MIPS 在内存管理上 的区别。（√）*
> - *简单了解并叙述RISC-V 中的内存管理机制，比较RISC-V 与MIPS 在内存管理上 的区别。*

X86 的内存管理机制通过分段和分页来实现。分段机制将内存分为多个段，每个段都有自己的基地址和长度，CPU使用段寄存器指定当前使用的段，并使用偏移量访问该段中的内存；分页机制用细粒度的单位页来管理线性地址空间和物理地址空间，线性地址被分为固定大小的页，物理内存被分为相同大小的页，每个页有一个唯一的物理地址，CPU使用页表将线性地址转换为物理地址。虚拟地址通过分段机制转换为线性地址，再通过分页机制转为物理地址。

MIPS 只使用分页式内存存储，可能无法实现分段管理的内存保护机制。