实验5 IntelSIMD指令实验

余北辰 519030910245

Exercise 1: 熟悉SIMD intrinsics函数

找出能完成以下操作的128-位intrinsics函数：(one for each):

- Four floating point divisions in single precision (i.e.float)（4个并行的单精度浮点数除法）

__m128 _mm_div_ps (__m128 a, __m128 b)

- Sixteen max operations over unsigned 8-bit integers (i.e.char)（16个并行求8位无符号整数的最大值）

__m128i _mm_max_epu8 (__m128i a, __m128i b)

- Arithmetic shift right of eight signed 16-bit integers (i.e.short)（8个并行的16位带符号短整数的算术右移）

__m128i _mm_srai_epi16 (__m128 a, int imm)

Exercise 2： 阅读SIMD 代码

观察sseTest.s文件的内容，哪些指令是执行 SIMD 操作的？

```
movapd  xmm2, XMMWORD PTR 48[rsp]
mov rax, QWORD PTR .LC3[rip]
movapd  xmm7, XMMWORD PTR .LC5[rip]
mulpd   xmm7, xmm2
movsd   QWORD PTR 64[rsp], xmm1
mov QWORD PTR 72[rsp], rax
```

```
    mov eax, 6
    movsd   QWORD PTR 40[rsp], xmm8
    mulpd   xmm2, xmm3
    movapd  xmm6, XMMWORD PTR 64[rsp]
    addpd   xmm7, XMMWORD PTR 80[rsp]
    movapd  xmm4, xmm6
    addpd   xmm2, XMMWORD PTR 96[rsp]
    mulpd   xmm4, xmm3
    addpd   xmm6, xmm6
```

mulpd、movapd、addpd都是执行 SIMD 操作的。

# Exercise 3: 书写SIMD 代码

```
static int sum_vectorized(int n, int *a)
{
    // WRITE YOUR VECTORIZED CODE HERE
    __m128i sum = _mm_setzero_si128();
    for (int i = 0; i < n / 4 * 4; i += 4)
    {
        __m128i temp = _mm_loadu_si128((__m128i *)(a + i));
        sum = _mm_add_epi32(temp, sum);
    }

    int A[4] = {0, 0, 0, 0};

    _mm_storeu_si128((__m128i *)A, sum);

    int ans = 0;
    ans += A[0] + A[1] + A[2] + A[3];
    for (int i = n / 4 * 4; i < n; i++)
        ans += a[i];
    return ans;
}
```

性能提升，输出结果为：

naive: 3.81 microseconds

unrolled: 2.81 microseconds

vectorized: 1.12 microseconds

vectorized unrolled: ERROR!

# Exercise 4: Loop Unrolling循环展开

```
static int sum_vectorized_unrolled(int n, int *a)
{
    // UNROLL YOUR VECTORIZED CODE HERE
    __m128i sum = _mm_setzero_si128();
    for (int i = 0; i < n / 16 * 16; i += 16)
    {
        __m128i temp = _mm_loadu_si128((__m128i *)(a + i));
        sum = _mm_add_epi32(temp, sum);

        temp = _mm_loadu_si128((__m128i *)(a + i + 4));
        sum = _mm_add_epi32(temp, sum);

        temp = _mm_loadu_si128((__m128i *)(a + i + 8));
        sum = _mm_add_epi32(temp, sum);

        temp = _mm_loadu_si128((__m128i *)(a + i + 12));
        sum = _mm_add_epi32(temp, sum);
    }

    int A[4] = {0, 0, 0, 0};

    _mm_storeu_si128((__m128i *)A, sum);

    int ans = 0;
    ans += A[0] + A[1] + A[2] + A[3];
    for (int i = n / 16 * 16; i < n; i++)
        ans += a[i];
    return ans;
}
```

性能提升，输出结果为：

```
naive: 2.90 microseconds
unrolled: 2.17 microseconds
vectorized: 1.01 microseconds
vectorized unrolled: 0.58 microseconds
```