**mymemcpy**

**2018K8009918002 陈彦帆**

64位汇编程序 mymemcpy.S C测试程序 copy.c

注：需要AVX2支持

1. 编译指令

gcc -c -O2 copy.c

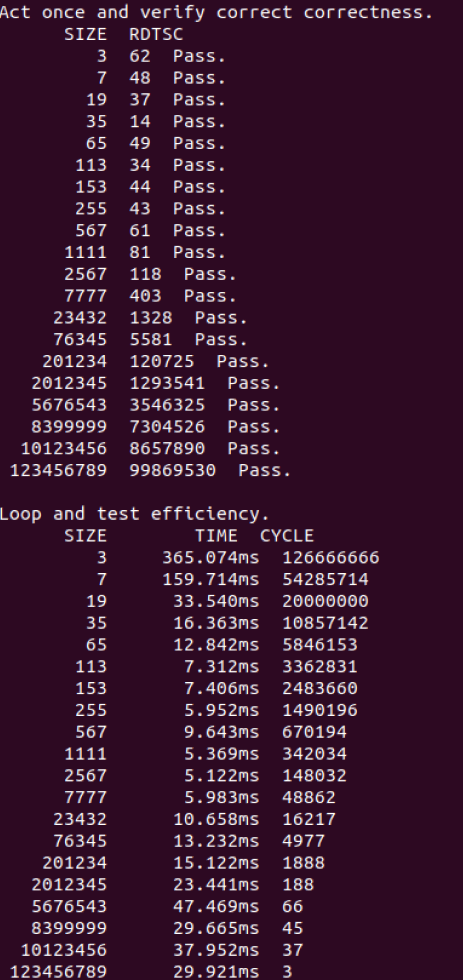
gcc -c mymemcpy.S

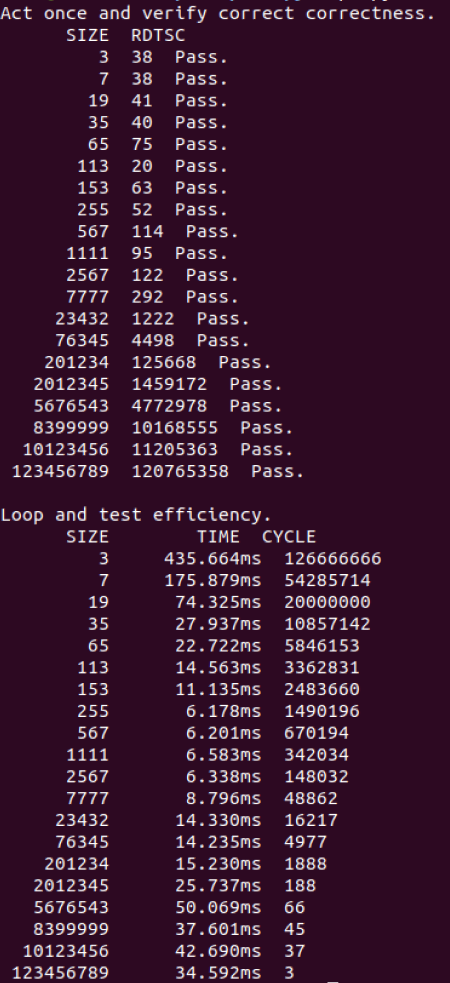
gcc mymemcpy.o copy.o -o copy

./copy

2. 测试正确性和效率

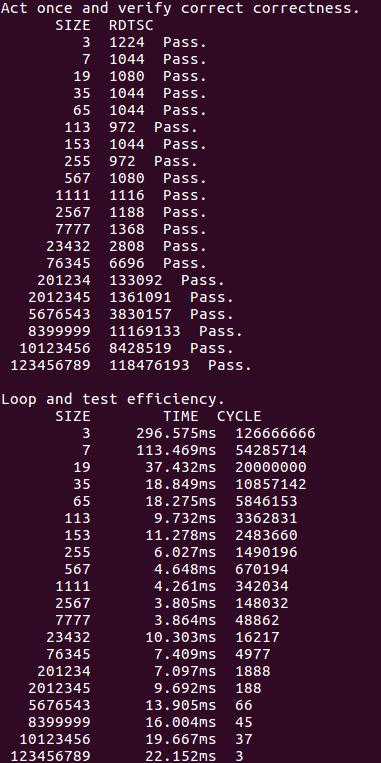
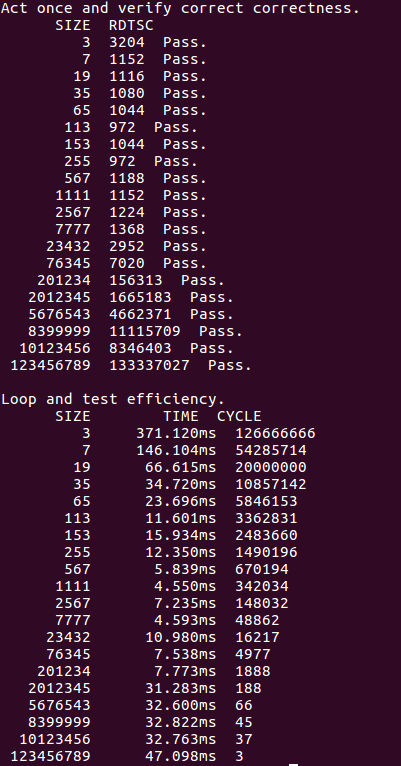
c测试程序选取20种不同大小的数组（SIZE单位为字节），分两步测试。第一步，拷贝一次，并通过for循环检查正确性。第二步，循环若干次，测试总运行时间。测试时，用最新的库函数memcpy进行对比（为此，只需把c程序中所有mymemcpy函数改为memcpy即可）。

测试1: 运行环境：Intel Core i7 8550u Ubuntu18.04

 mymemcpy 库函数

可以看出，在笔记本Intel环境下，mymemcpy的效率与库函数相当甚至稍快于库函数，由于笔记本cpu频率变化大，测试的时间可能在小范围波动。总之，mymemcpy的效率与库函数相当。

测试2: 运行环境: AMD R5 3600 Ubuntu16.04

 mymemcpy 库函数

将cpu频率固定至3500MHz，将内存频率固定至3000MHz。根据循环测试的时间可以看出，在AMD环境下，对于小尺寸和大尺寸，mymemcpy效率高于库函数。对于中等尺寸，mymemcpy效率与库函数相当。

3.算法描述

（采用AVX指令）

首先判断拷贝的大小范围，根据尺寸采取三种不同的方案。

(1)若尺寸为0-63字节。进一步确认范围至2的相邻幂次。采用两头各覆写一次的方法。比如若确认范围为4-7字节，则头尾各拷贝4字节，用movl指令。若确认范围为32-63字节，则头尾各拷贝32字节，用vmovdqu指令。

(2)若尺寸为64B-8MB。根据对齐情况在尾部拷贝64或者32字节。然后从头开始拷贝，每次用2条vmovdqu指令拷贝64字节。需拷贝n>>6次。

(3)若尺寸大于等于8MB（大于L2缓存，约等于L3缓存的尺寸），采用vmovntqu指令对齐地直写内存，每次拷贝128字节。方法是，从头拷贝32字节，尾部拷贝128字节。然后将目标地址头部32字节对齐，调整对应的读地址，每次从源地址拷贝128字节(4个vmovdqu)到%ymm寄存器，从寄存器拷贝128字节到对齐的目标地址(4个vmovntqu)。需拷贝n>>7次。