

**课 程 实 验 报 告**

**课程名称： 计算机系统基础实验**

**实验名称：** **ARM指令系统的理解**

**院 系 ：计算机科学与技术**

**专业班级 ： CS2205**

**学 号 ： U202215510**

**姓 名 ： 徐新飏**

**指导教师 ： 李专**

**2024 年 5 月 11 日**

**一、实验目的与要求**

通过在ARM虚拟环境下调试执行程序，了解 ARM的指令系统。

实验环境：ARM 虚拟实验环境 QEMU

工具：gcc, gdb 等

**二、实验内容**

**任务1、C与汇编的混合编程**

**任务2、内存拷贝及优化实验**

程序及操作方法 见 <ARM实验任务.pdf>

**三、实验记录及问题回答**

**（1）实验任务的实验结果记录**

任务1：C与汇编的混合编程

C语言调用汇编实现累加和求值：

使用vim编写程序sum.c，其中输入输出由c语言直接实现，add函数由汇编语言实现。

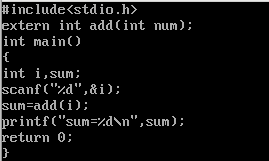


图1.1

使用vim编写程序add.s，实现从1到i的累加求和。

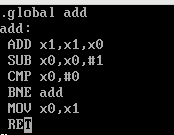


图1.2

使用gcc编译生成可执行文件，运行程序，输入100，输出5050。

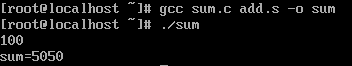


图1.3

C语言内嵌汇编：

使用vim编写程序builtin.c，其中使用\_\_asm\_\_ \_\_volatile\_\_来实现内嵌汇编。

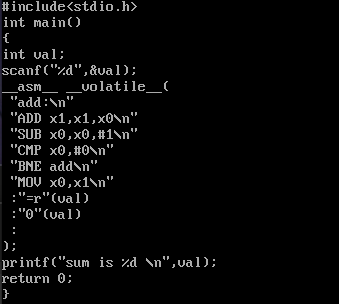


图1.4

使用gcc进行编译然后运行，输入100，返回累加和5050。



图1.5

任务2：内存拷贝及优化实验

使用vim编写c语言计时程序time.c。

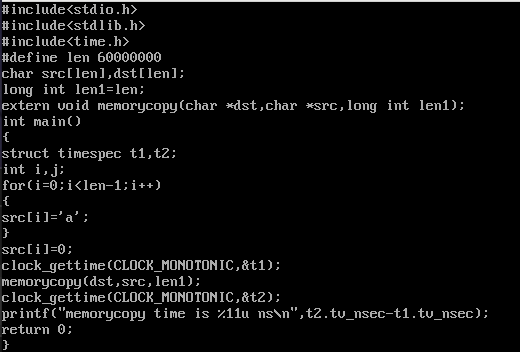


图2.1

使用vim创建copy.s文件，编写优化前的原始汇编代码。

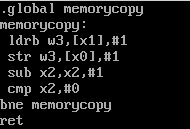


图2.2

使用gcc完成编译并执行。



图2.3

循环展开优化：

进行循环展开，将copy.s展开两倍，一次循环读写两个字节，优化后的copy121.s如下。

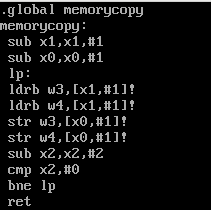


图2.4

编译并运行执行文件，memorycopy函数的执行时间变为了183071127ns，实现了优化。



图2.5

创建4倍展开优化copy122.s文件

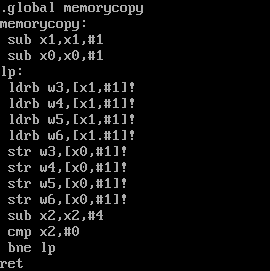


图2.6

编译并运行执行文件，memorycopy函数的执行时间变为了169093395ns，实现了进一步优化。



图2.7

内存突发传输方式优化：

之前每次内存读写都是以一个字节为单位进行的，这样的效率很低，由于内存在连续读写多个数据时其性能要优于非连续读写数据的方式，因此使用ldp和stp指令来一次读写多个字节，实现优化，其一次可以访问16个字节的内存数据，大大提高了内存读写效率。

创建内存突发传输优化copy21.s文件

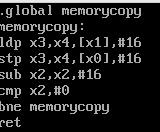


图2.8

编译并运行执行文件



图2.9

可见一次对16字节读写程序执行效率明显优于单字节读写。

**（2）ARM 指令及功能说明**

ARM64提供了31个64位通用寄存器，其中ARM指令可以由如下模板描述，即：

MNEMONIC{S}{condition} {Rd}, Operand1, Operand2

模板中各字段的作用如下所示：

MNEMONIC：指令的助记符如ADD。

{S}：可选的扩展位。

{condition}：执行条件，如果没有指定，默认为AL(无条件执行)。

{Rd}：目的寄存器，存储指令计算结果。

Operand1：第一个操作数，可以是一个寄存器或一个立即数。

Operand2：第二个(可变)操作数，可以是一个立即数或寄存器甚至带移位操作的寄存器。

本次实验用到了

ADD数据相加：rd=op1+op2

SUB数据相减：rd=op1-op2

MOV数据移动:将op2的值赋给op1

CMP数据比较：op1与op2比较，做op1-op2的操作

BNE数据跳转：标志寄存器中Z标志位不等于0时，跳转到bne后标签处

LDRB字节数据加载；例如LDRB x0,[x1,#8];将存储器地址为x1＋8的字节数据读入寄存器x0，并将x0的高24位清零。

STR存储数据：例如STR x0,[x1,#8];意思是将x0中的字数据写入x1+8为地址的存储器中。

LDP一次访问16个字节，从内存中加载数据，将其储存到两个寄存器中。

STP一次访问16个字节，从两个寄存器中将数据存储到内存中。

RET函数返回。

**四、体会**

此次实验通过在ARM虚拟环境（QEMU）下调试执行程序，了解了ARM的指令系统。通过使用gcc，gdb和vim等工具，这一实验帮助我掌握了C与汇编的混合编程，也更深入地了解了内存操作及其优化方法。

在QEMU虚拟环境中执行和调试程序，深入体会到了ARM指令集以其简洁和高效著称的特点， 将C语言和汇编语言结合起来进行编程，我学会了如何在C语言中嵌入汇编代码，掌握了内联汇编的使用方法，也理解了其重要的作用，在关键代码段使用汇编可以显著优化性能，而在其他部分使用C语言则可以提高代码的可读性和维护性。

在内存拷贝及其优化实验中，通过不同的优化方法（如循环展开、使用更高效的指令等），程序的性能得到了显著提升。这让我理解了在实际编程中，如何通过算法优化和合理的指令选择来提高程序效率。

此次ARM虚拟环境下的实验，使我对ARM指令系统有了深入的了解，这些经验对于今后的学习和工作都非常宝贵，实验中遇到的问题和解决方法也让我明白了理论与实践结合的重要性，为我今后继续深入研究ARM指令等方向打下了坚实的基础。