

**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 ： cs2208**

**学 号 ： U202215642**

**姓 名 ： 田清林**

**指导教师 ： 班鹏新**

**2024 年 4 月 30 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>

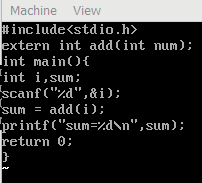
**三、实验记录及问题回答**

**（1）实验任务的实验结果记录**

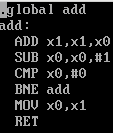
**任务1：C与汇编的混合编程：**

1. **C语言调用实现累加和求值**

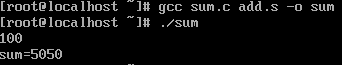
创建sum.c，如下图所示：



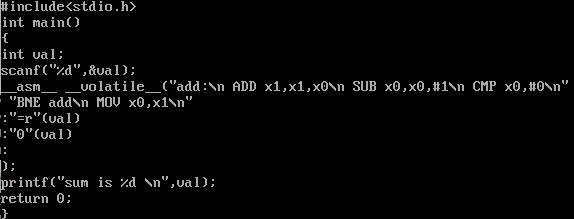
编写add.s，如下图所示



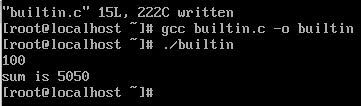
使用编译生成可执行文件sum并执行，结果如下：



1. c语言内嵌汇编。编写builtin.c，如下图所示：



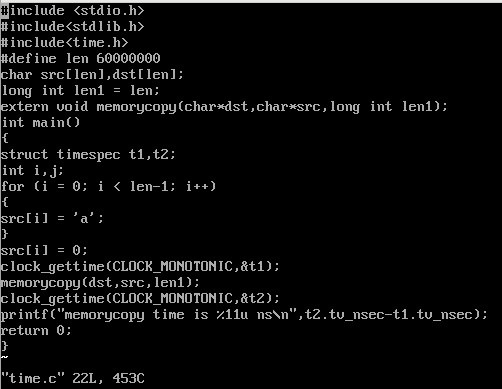
用gcc编译生成可执行文件builtin，并执行如下图所示：



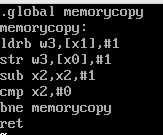
任务二：内存拷贝与优化实验

1. 基础代码

编写time.c如下图所示：



编写copy.s如下图所示：

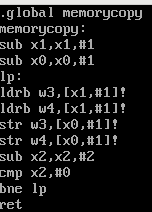


用gcc编译生成可执行文件m1，并执行，结果如下：



1. 循环展开优化

编写copy121.s如下图所示：

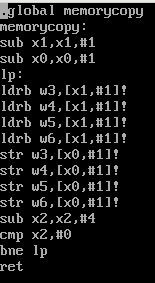


用gcc编译生成可执行文件m121，并执行，结果如下：

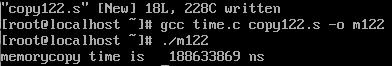


对比可以看出执行时间减少到原来的一半左右。

编写copy122.s如下图所示：

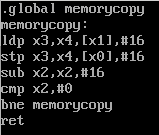


用gcc编译生成可执行文件m122，并执行，结果如下：



1. 内存突发传输方式优化

编写copy21.s如下图所示：



用gcc编译生成可执行文件m21，并执行，结果如下：



对比可以看到再次优化后，执行时间减少到最初的约五十六分之一。

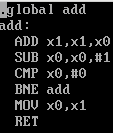
一次对16字节读写程序的执行效率明显优于单字节读写。

**（2）ARM 指令及功能说明**

**任务1：C与汇编的混合编程**

1. **C语言调用实现累加和求值**

**add.s文件内容如下所示：**

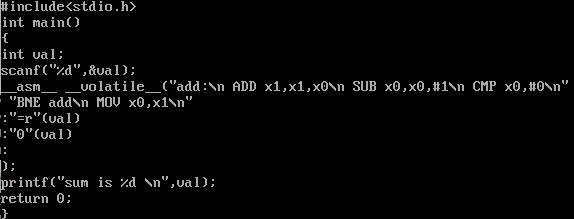


add函数具有一个参数，由x0传递，函数返回时x0为返回值。

在执行过程中，x1存储求和的中间结果，x0为循环变量。首先ADD 将x0+x1的值存入寄存器x1，之后将SUB将x0-1存入x0寄存器，比较x0和0的大小，若x0和0不相等则跳转到add，若相等则继续执行，将x1的值存入x0，作为返回值。

1. **C语言内嵌汇编**

bulitin.c文件内容如下所示：



在C语言中内嵌汇编代码，需要使用\_\_asm\_\_关键字，\_\_volatile\_\_关键字的作用是告诉编译器不要优化内嵌的汇编语句，这样可以保证程序达到预期的执行效果。

第一部分是汇编代码，汇编代码之间用\n隔开，可以放在多个引号内，这里的代码与先前add.s内容相同，不再解释。

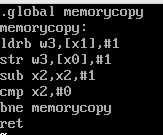
第二部分是输出操作数列表，引号内的r表示存放在某个通用寄存器中，即用一个寄存器代替（）部分中定义的val，=表示只写，即在汇编代码里只能改变C变量的值，而不能取它的值。

第三部分是输入操作数列表，引号内的0代表与第一个输出参数共用一个寄存器。

任务二：内存拷贝与优化实验

1. 基础代码

**copy.s文件内容如下：**

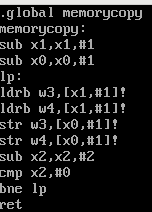


**该段汇编代码定义了一个函数memorycopy（char\*dst，char\* src，int len），功能是实现将尺寸为len的src字符数组的内容拷贝到同样尺寸的dst字符数组中。三个参数分别依次由x0，x1，x2传递。**

**ldrb从源字符串地址中读取一个字节，str向目的字符串地址中写入（此处ldrb和str都使用后索引方式，以ldrb为例子，将x1中一个字节读取并存在w3内之后将x1加1），x2内存储循环变量，初始值为len，每次循环经sub指令减1，之后cmp指令比较x2是否为0，不为0则跳转到memorycopy重新循环，为0则继续执行，函数返回。**

1. 循环展开优化

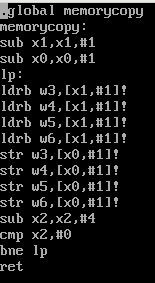
copy121.s如下图所示：



copy121.s是对copy.s进行二倍展开优化后的结果，即每次循环读和写两个字节。

在函数开头先使用两个sub语句将x1和x0分别减1，供接下来的自动索引方式使用。之后每次使用两次ldrb和str来读取和写入**（此处ldrb和str都使用自动索引方式，以ldrb为例子，读取x1+1处的一个字节并存在w3内，之后将x1加1），且x每次循环减2，之后cmp指令比较x2是否为0，若x0不为0则跳转到lp重新循环，为0则继续执行，函数返回（注意程序有漏洞，只有在len为偶数时才能正常工作）。**

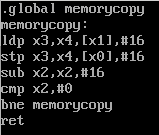
编写copy122.s如下图所示：



**copy122.s是对**copy.s进行四倍展开优化后的结果，即每次循环读和写四个字节。原理与copy121.s相同，不再赘述。

1. 内存突发传输方式优化

编写copy21.s如下图所示：



由于内存在连续读/写多个数据时，其性能要优于非连续读/写的方式，因此可以通过一次对多个字节进行读写来对copy.s进行优化。

函数使用ldp指令和stp指令，这两条指令可以一次访问16个字节的内存数据，以ldp为例子，从x1处连续读16个字节，放置在x3，x4中，再将x1加16（后索引寻址方式）。stp指令同理，将x3，x4写入x0处的连续16个字节，再将x2加16。之后将x2即len减16，比较x2是否为0，若不为0则跳转到memorycopy重新循环，否则继续执行，函数返回。

**四、体会**

**本次实验是对ARM架构系统的探索。先前实验中使用的系统都是x86架构的，本次实验需要我们编写汇编语言程序，ARM和x86的汇编代码相似也有许多不同点，比如ARM指令是定长指令，这是由不同的架构导致的设计不同。本次实验通过编写ARM汇编代码，增进了本人对ARM汇编语言的了解，**

**本次实验教会了我们如何在c语言中内嵌汇编代码，c语言作为高级语言中相对底层的语言，内嵌汇编代码方便编程者在指令级对计算机进行操纵，有利于更深层的优化。**

**在内存拷贝与优化实验中，本人通过将逐字节拷贝的代码优化为每次循环2字节、4字节拷贝，直到最后每次连续进行16字节的拷贝，将程序的拷贝效率提高了约56倍，这符合课堂上讲到的关于存储和缓存的知识点，加深了我对上述知识的理解。**