

**课 程 实 验 报 告**

**课程名称： 计算机系统基础**

**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 ： CS2305**

**学 号 ： U202315677**

**姓 名 ： 岳皓**

**指导教师 ： 班鹏新**

**2025 年 4 月 25 日**

# 实验六 ARM指令系统的理解

**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）实验过程和结果记录**

进行实验前，先进行如下操作：

1）将给出的压缩包释放到任意硬盘中，比如D:中，最后形成的目录为D:\QEMUTEST\QEMU

2）在windows系统高级设置中，修改windows环境变量中的系统变量，增加一个path=D:\QEMUTEST\QEMU，重启动计算机。具体操作可参见windows操作说明或"ARM虚拟环境安装说明"中相关内容

3）以系统管理员身份运行“CMD"

4）进入到D:\QEMUTEST目录下

方法：C:>D:

D:>CD QEMUTEST

5)每次启动时，将下列命令串复制到命令行

qemu-system-aarch64 -m 4096 -cpu cortex-a57 -smp 4 -M virt -bios edk2-aarch64-code.fd -hda openEuler-20.03-LTS.aarch64.qcow2 -serial vc:800x600

6）在显示的QEMU窗口中，在菜单“View”下选择Serial0。

用户名：root

密码：openEuler12#$

**任务一**

首先输入命令vi sum.c，进入到创建的sum.c文件中，按“A”键进入编辑模式进行代码的编写。编写好的累加和代码如图1，编写完成后，按“ESC”键进入命令模式，再输入“:wq”后按回车键进行保存。接着，按照相同的步骤，输入命令vi add.s来编写所调用的汇编代码，如图2。

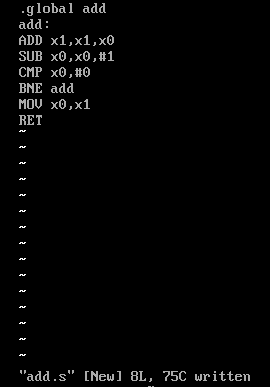
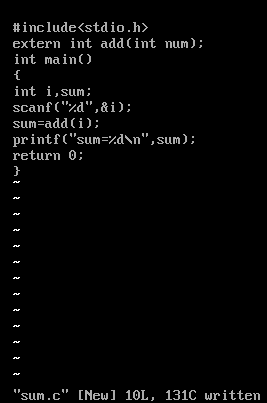


图 1 sum.c代码 图 2 add.s汇编代码

编写完成后，需要输入gcc命令来生成可执行文件，按照语法，输入gcc sum.c add.s -o sum来让系统执行生成名字为sum的可执行文件的操作，然后输入./sum来运行该可执行文件，输入100，由图3可知，输出为累加和5050，结果正确。

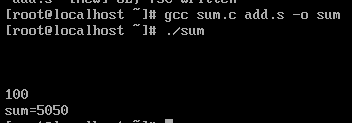


图 3 运行结果

随后进行下一个实验，首先输入命令vi builtin.c，进入到创建的builtin.c文件中，按“A”键进入编辑模式进行代码的编写。利用内嵌汇编代码编写好的累加和代码如图4所示，编写完成后，按“ESC”键进入命令模式，再输入“:wq”后按回车键进行保存。其中，\_\_asm\_\_用于声明内嵌汇编代码表达式，关键字volatile用于告诉编译器不用优化内嵌的汇编语句。

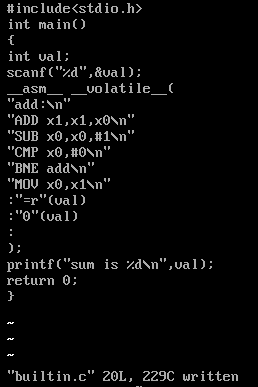


图 4 builtin.c代码

编写完成后，需要输入gcc命令来生成可执行文件，按照语法，输入gcc builtin.c -o sum来让系统执行生成名字为sum的可执行文件的操作，然后输入./builtin来运行该可执行文件，输入100，由图5可知，输出为累加和5050，结果正确。

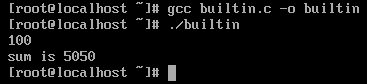


图 5 运行结果

**任务二**

首先输入命令vi time.c，进入到创建的time.c文件中，按“A”键进入编辑模式进行代码的编写。编写好的计算内存拷贝函数运行时间的代码如图6所示，其中，内存拷贝函数memorycopy()的功能是实现将尺寸为len(这里设置为60000000)的src字符数组的内容拷贝到同样尺寸的dst字符数组中。memorycopy(函数用AArch64汇编代码实现。 在本例中，需要传递的参数有三个: 第一个参数是目标字符串的首地址，用寄存器x0来传递； 第二个参数是源字符串的首地址，用寄存器x1来传递；第三个参数是传输的字节数目，用寄存器x2来传递,调用clock\_gettime()来分别记录memorycopy()执行前后的系统时间，然后通过相减的结果得到以纳秒为单位的精确结果。

编写完成后，按“ESC”键进入命令模式，再输入“:wq”后按回车键进行保存。接着，按照相同的步骤，输入命令vi copy.s来编写所调用的原始汇编代码，如图7。

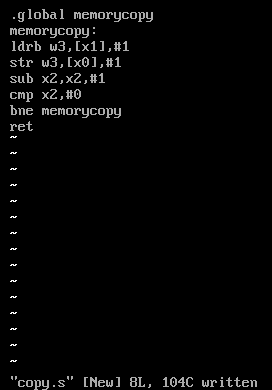
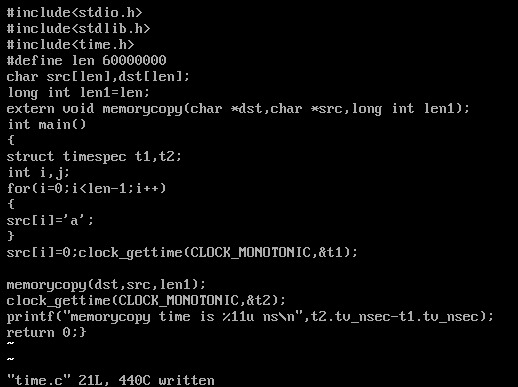


图 6 time.c代码 图 7 copy.s汇编代码

编写完成后，需要输入gcc命令来生成可执行文件，按照语法，输入gcc time.c copy.s -o m1来让系统执行生成名字为m1的可执行文件的操作，然后输入./m1来运行该可执行文件，由图8可知，memorycopy()函数具体执行时间为3543128800ns。

QQ20250425-174649

图 8 运行结果

首先输入命令vi copy121.s，进入到创建的copy121.s文件中，按“A”键进入编辑模式进行代码的编写。编写的copy.s展开两倍的代码如图9所示，编写完成后，按“ESC”键进入命令模式，再输入“:wq”后按回车键进行保存。

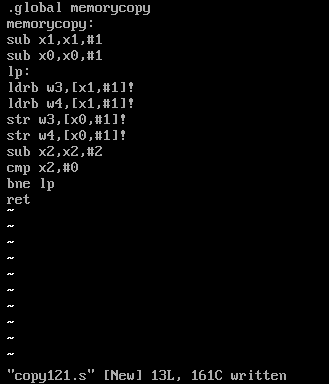


图 9 copy121.s代码

编写完成后，需要输入gcc命令来生成可执行文件，按照语法，输入gcc time.c copy121.s -o m121来让系统执行生成名字为m121的可执行文件的操作，然后输入./m121来运行该可执行文件，由图10可知，memorycopy()函数具体执行时间为3508783296ns。

QQ20250424-195257

图 10 运行结果

首先输入命令vi copy122.s，进入到创建的copy122.s文件中，按“A”键进入编辑模式进行代码的编写。编写的copy.s展开四倍的代码如图11所示，编写完成后，按“ESC”键进入命令模式，再输入“:wq”后按回车键进行保存。

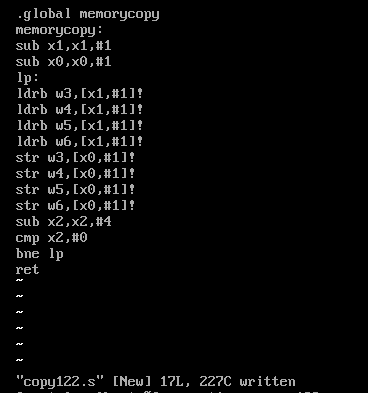


图 11 copy122.s代码

编写完成后，需要输入gcc命令来生成可执行文件，按照语法，输入gcc time.c copy122.s -o m122来让系统执行生成名字为m122 的可执行文件的操作，然后输入./m122来运行该可执行文件，由图12可知，memorycopy()函数具体执行时间为240019104ns。

QQ20250424-195926

图 12 运行结果

首先输入命令vi copy21.s，进入到创建的copy21.s文件中，按“A”键进入编辑模式进行代码的编写。将传输优化的代码如图13所示，编写完成后，按“ESC”键进入命令模式，再输入“:wq”后按回车键进行保存。

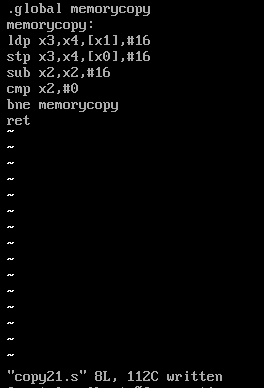


图 13 copy21.s代码

编写完成后，需要输入gcc命令来生成可执行文件，按照语法，输入gcc time.c copy21.s -o m21来让系统执行生成名字为m21 的可执行文件的操作，然后输入./m21来运行该可执行文件，由图14可知，memorycopy()函数具体执行时间为57676304ns。

QQ20250424-200501

图 14 运行结果

1. **ARM 指令功能介绍和特点分析**

1.mov   
功能：比较两个寄存器或立即数的值，并根据得到的值更新条件标志。

特点：不修改源操作数，仅通过标志位反映比较结果，常用于条件分支前的判断。

2.ldp

功能：从内存连续加载两个64位数据到两个寄存器。

特点：高效的双寄存器加载，减少内存访问次数，支持前/后变址寻址。

3.stp

功能：将两个寄存器的值连续存储到内存。

特点：与ldp对称，优化批量数据存储。 同样支持变址寻址模式。

4.sub

功能：执行减法运算。

特点：可设置标志位,用于后续条件判断，支持灵活的操作数(寄存器、移位后的寄存器或立即数)。

5.bne

功能：若条件标志Z=0，则跳转目标地址。

特点：典型的条件转移指令，依赖CHP或sUBS等指令设置的标志位，跳转范围受限于相对偏移量编码。

6.ret

功能：从函数返回，通常跳转到链接寄存器LR(X30)保存的地址。

特点：可指定返回地址寄存器(如RET x3),默认为x30，用于函数调用栈的恢复。

7.str

功能：将寄存器值存储到内存。

特点：支持多种数据宽度(如STRB字节、STRH半字)，寻址模式灵活(基址+偏移、前/后变址)。

8.ldrb

功能：从内存加载一个字节到寄存器，高位零扩展。

特点：用于处理8位数据，避免符号污染，类似指令LDRSB会进行符号扩展。

9.bl

功能：跳转到目标地址，同时将返回地址保存到LR(x30)。

特点：用于函数调用，实现子程序跳转，结合RET完成调用栈管理。

10.mov

功能：在寄存器间或立即数间移动数据。

特点：支持多种操作数。

**四、体会**

通过本次实验，我深入理解了ARM指令系统的基本操作和优化方法，收获如下：ARM指令功能掌握:熟悉了mov、ldp、stp、sub、bne等核心指令的功能与特点，例如LDP/STP通过批量加载/存储提升效率，BNE依赖标志位实现条件跳转。混合编程实践:在任务一中，通过C调用汇编函数(add.s)实现累加和计算，理解了跨语言调用的参数传递规则(如寄存器x0-x7用于参数传递)。 内嵌汇编应用:任务二中，使用\_\_asm\_\_ \_\_volatile\_\_内嵌汇编直接优化C代码，避免了函数调用开销，加深了对编译器优化的认识·性能优化技巧:在内存拷贝实验(任务三至六)中，通过展开循环(2倍、4倍)和使用LDP/STP指令，使拷贝时间显著减少,显著提升了效率。

本次实验不仅帮助我巩固了ARM指令的理论知识，更通过实践掌握了性能分析和优化的关键方法。未来在嵌入式开发或系统编程中，这些经验将帮助我编写更高效的底层代码。