

实验报告

课程：嵌入式系统A

第二次实验

**姓 名 凌智城**

**学 号 201806061211**

**专业班级 通信工程1803班**

**老 师 黄国兴**

**学 院 信息工程学院**

**提交日期**  **2021年5月12日**

实验6：C语言程序实验

1. 实验目的

了解使用ADS1.2编写C语言程序并进行调试。

1. 实验内容

编写一个汇编程序文件和一个C程序文件。汇编程序的功能是初始化堆栈指针和初始化C程序的运行环境，然后跳转到C程序运行，这就是一个简答的启动程序。C程序使用加法运算来计算1+2+3+…+(N-1)+N的值(N>0)。

1. 实验步骤

**步骤1：**正确连接实验箱及PC机

将Multi-ICE的一端与PC机USB口正确链接，另一端与开发板正确连接。将开发板的电源线正确连接，插上电源、串口线，先不要打开开发板的电源开关。

**步骤2：**建立工程

启动CodeWarrior for ARM Developer Suite，选择File->New命令，使用ARM Executable Image工程模板建立一个工程，工程名称为ProgramC。

**步骤3：**在工程中创建一个新文件

选择File🡪New命令，建立一个新文件Test.c和Startup.S，直接添加到项目中，输入代码并保存。

**步骤4：**设置地址和起始代码段

选择Edit🡪DebugRel Setting命令，在DebugRel Setting的对话框的左侧选择ARM Linker选项，然后再Output选项卡中设置链接地址，RO Base为0x4000000，RW Base为0x4003000；在Options选项卡中设置调试入口地址Image entry point为0x4000000；在Layout选项卡中设置位于开始位置的起始代码段，如图6-1所示。

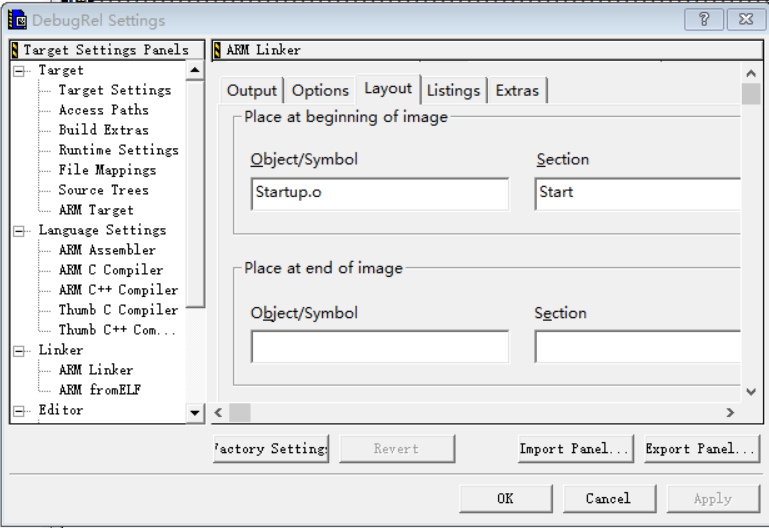


图6-1 ARM Linker选项

**步骤5：**编译工程并仿真调试

选择Project🡪Make命令，将编译、链接整个工程，代码及错误和警告对话框如图6-2所示；然后选择Project🡪Debug命令，启动AXD进行软件仿真调试。

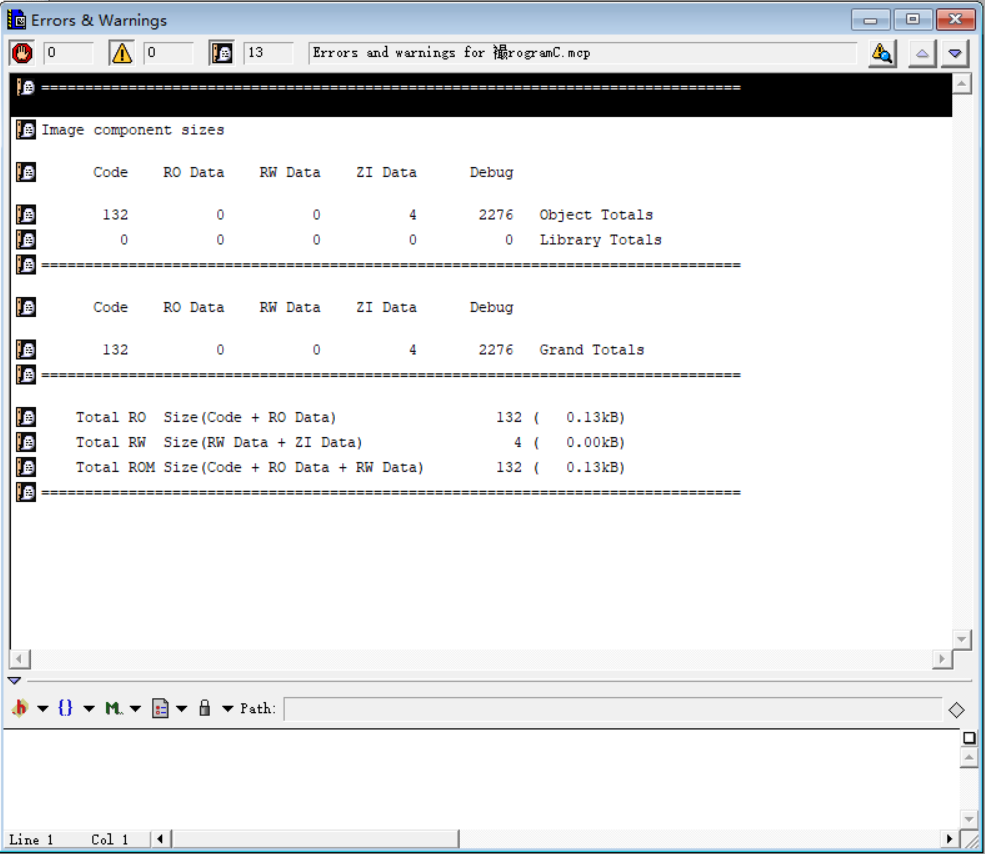


图6-2 映像文件数据

**步骤6：**设置断点并运行程序

在Startup.S的B Main处设置断点，然后元素运行程序，如图6-3所示。

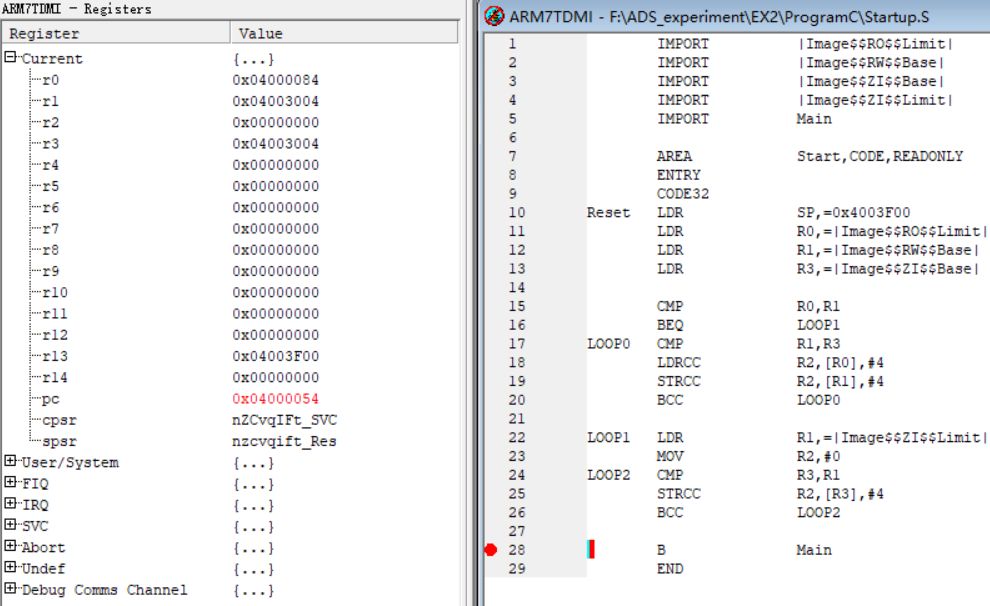


图6-3 程序全速运行图

**步骤7：**单步运行程序

程序在断点处停止。单步运行程序，判断程序是否跳转到C程序运行，如图6-4所示。

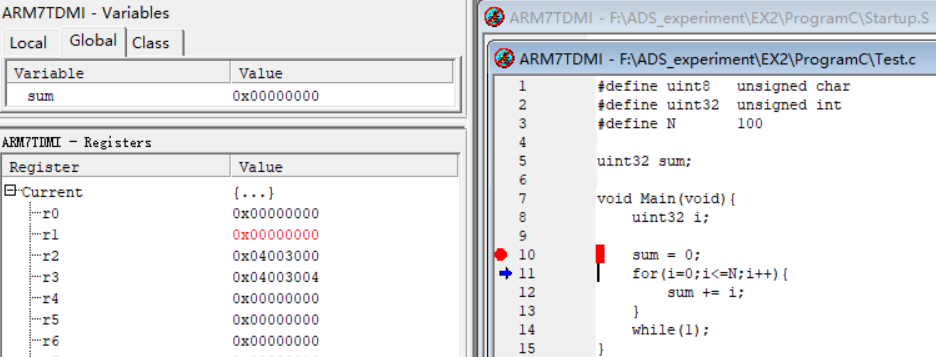


图6-4 程序单步运行图

**步骤8：**观察全局变量的值，判断程序的运算结果是否正确

选择Processor View🡪Variables命令，打开变量观察窗口，观察全局变量的值，单步或全速运行程序，判断程序的运行结果是否正确（也可以选择在while(1)处设置断点后再全速运行），如图6-5所示。

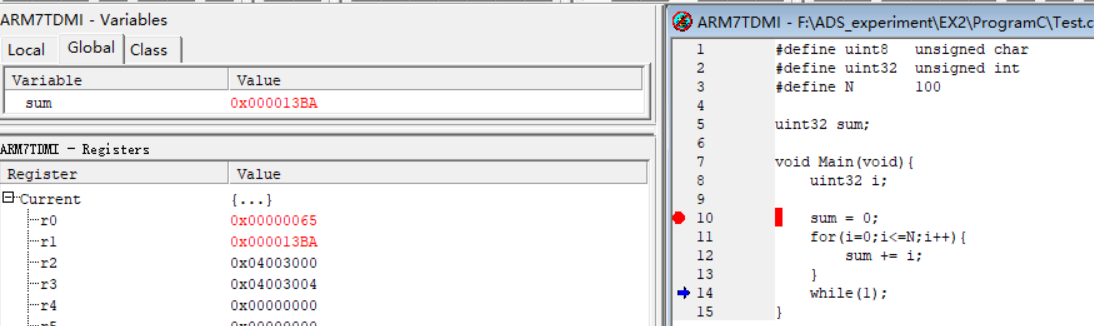


图6-5 程序变量运行图（0x000013BA=5050）

1. 心得与体会
2. RO（readonly）是程序中的指令和常量，RW（read/write）是程序中的已初始化变量，ZI（zero）是程序中的未初始化的变量。
3. Total ROM Size（Code + RO Data + RW Data）
4. |Image$$RO$$Base| = RO Base

表示RO输出段运行时起始地址，也可以说是程序代码存放的起始地址。

1. |Image$$RO$$Limit| = RO Base + Total RO Size（Code + RO Data）

表示RO输出段运行时存储区域界限。

1. |Image$$RW$$Base| = RW Base

表示RW输出段运行时起始地址，而不一定是加载时的存放地址，因为RW输出段在加载时可能是在ROM中并紧跟着RO输出段存放的，当程序运行时才移动到RAM起始地址为|Image$$RW$$Base|的区域，由RW Base这个参数指定；未指定的话，默认紧跟RO输出段，那么|Image$$RW$$Base|＝|Image$$RO$$Limit|。

1. |Image$$RW$$Limit| = RW Base + Total RW Size（RW Data + ZI Data）

表示RW输出段运行时存储区域界限。

1. |Image$$ZI$$Base| = |Image$$RW$$Base| + RW Data

表示ZI输出段运行时起始地址。

1. |Image$$ZI$$Limit| = |Image$$ZI$$Base| + ZI Data

表示ZI输出段运行时存储区域界限。

1. 汇编程序调用C程序需要先IMPORT [函数名]
2. 附录

Startup.S

        IMPORT          |Image$$RO$$Limit|      *; 引入其他源文件的定义，区分大小写*

        IMPORT          |Image$$RW$$Base|

        IMPORT          |Image$$RW$$Limit|

        IMPORT          |Image$$ZI$$Base|

        IMPORT          |Image$$ZI$$Limit|

        IMPORT          Main             *; 引入C程序中Main函数的定义*

        AREA            Start,CODE,READONLY     *; 声明一段只读代码，名字为Start*

        ENTRY                                   *; 指定程序的入口点*

        CODE32 *; 32 位的 ARM 指令*

Reset   LDR             SP,=0x4003F00           *; 将#0x04003F00加载到SP*

        LDR             R0,=|Image$$RO$$Limit|  *; 将#0x04000084加载到R0*

        LDR             R1,=|Image$$RW$$Base|   *; 将#0x04003000加载到R1*

        LDR             R3,=|Image$$ZI$$Base|   *; 将#0x04003000加载到R3*

        CMP             R0,R1                   *; 减法比较R0和R1的值,R0-R1为负数*

        BEQ             LOOP1                   *; 上条指令N=1,Z=0,此处不跳转*

LOOP0   CMP             R1,R3                   *; 减法比较R1和R3的值,R1-R3=0*

        LDRCC           R2,[R0],#4              *; 上条指令Z=1,N=0,C=1,CC条件是C=0不执行*

        STRCC           R2,[R1],#4              *; 保证Image$$RW$$Base≥Image$$ZI$$Base*

        BCC             LOOP0                   *; 同上条*

LOOP1   LDR             R1,=|Image$$ZI$$Limit|  *; 将#0x04003004加载到R3*

        MOV             R2,#0                   *; 0->R2*

LOOP2   CMP             R3,R1                   *; 减法比较R3和R1的值,R3-R1<0，产生借位,N=1,Z=0,C=0*

        STRCC           R2,[R3],#4              *; CC条件为C=0即R3比R1小，则R3+4写回R3*

        BCC             LOOP2                   *; 满足CC条件，再来一遍LOOP2知道R3=R1*

        B               Main                    *; 跳转到C程序的Main函数继续执行*

        END                             *; 指示本源程序结束*

Test.c

#define uint8   unsigned char   *// 宏定义*

#define uint32  unsigned int    *// 宏定义*

#define N       100             *// 宏定义*

uint32 sum;                     *// 声明全局变量*

void Main(void){

    uint32 i;                   *// 声明无符int型i*

    sum = 0;

    for(i=0;i<=N;i++){          *// for循环实现累加*

        sum += i;

    }

    while(1);                   *// 原地等待*

}

调用C语言实现两数相减

1. 实验目的

掌握在C语言程序中条用汇编程序，了解ATPCS基本规则

1. 实验内容

在C程序中调用汇编子程序，实现两个整数的减法运算。汇编子程序的原型为uint32 Sub(uint32 x, uint32 y)。

其中，uint32已定义为unsigned int。

1. 实验步骤

**步骤1：**正确连接实验箱及PC机

将Multi-ICE的一端与PC机USB口正确链接，另一端与开发板正确连接。将开发板的电源线正确连接，插上电源、串口线，先不要打开开发板的电源开关。

**步骤2：**建立工程

启动CodeWarrior for ARM Developer Suite，选择File->New命令，使用ARM Executable Image工程模板建立一个工程，工程名称为ProgramC2。

**步骤3：**在工程中创建一个新文件

选择File🡪New命令，建立一个新文件Startup.S、Sub.S和Test.c，设置直接添加到项目中，输入代码并保存，其中Startup.S程序同上一实验。

**步骤4：**设置地址和起始代码段

选择Edit🡪DebugRel Setting命令，在DebugRel Setting的对话框的左侧选择ARM Linker选项，然后再Output选项卡中设置链接地址，RO Base为0x4000000，RW Base为0x4003000；在Options选项卡中设置调试入口地址Image entry point为0x4000000；在Layout选项卡中设置位于开始位置的起始代码段设置为Startup.o的Start段。

**步骤5：**编译工程并仿真调试

选择Project🡪Make命令，将编译、链接整个工程，代码及错误和警告对话框如图1所示；然后选择Project🡪Debug命令，启动AXD进行软件仿真调试。

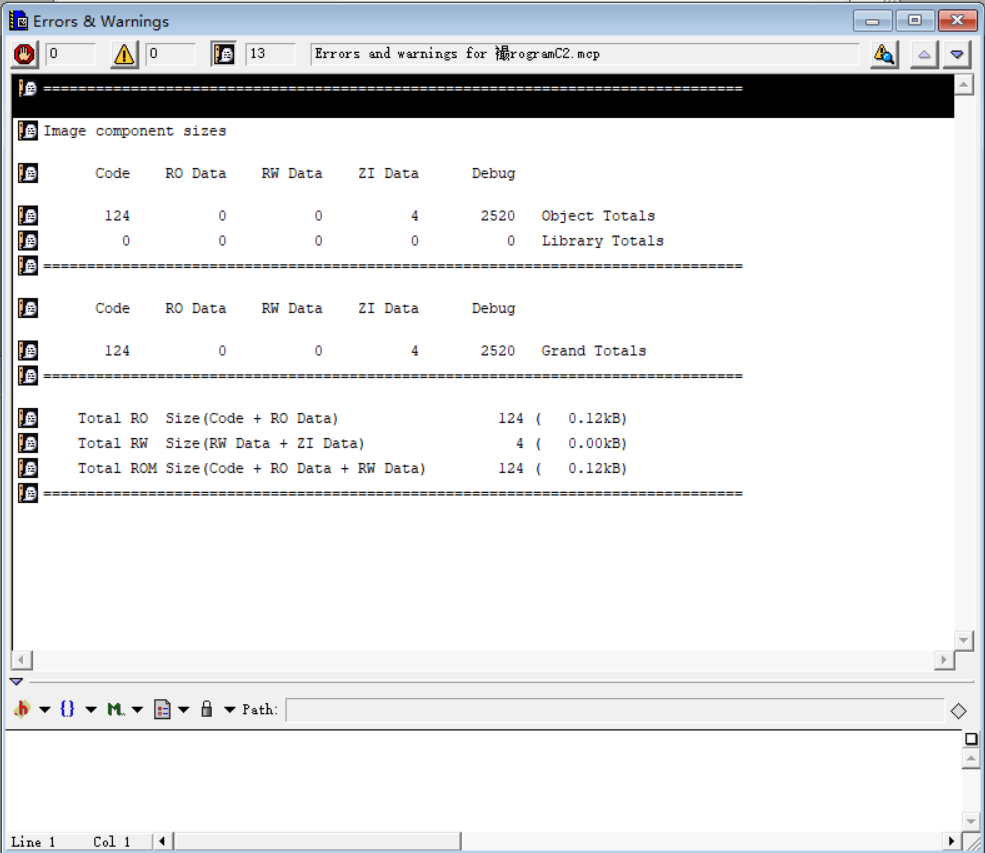


图1 映像文件数据

**步骤6：**设置断点并运行程序

在Startup.o的B Main处设置断点，然后全速运行程序。单步运行跳至Test.c文件，运行至sum=Sub()处，再单步运行，观察程序是否跳转到汇编程序Sub.S，如图2所示。

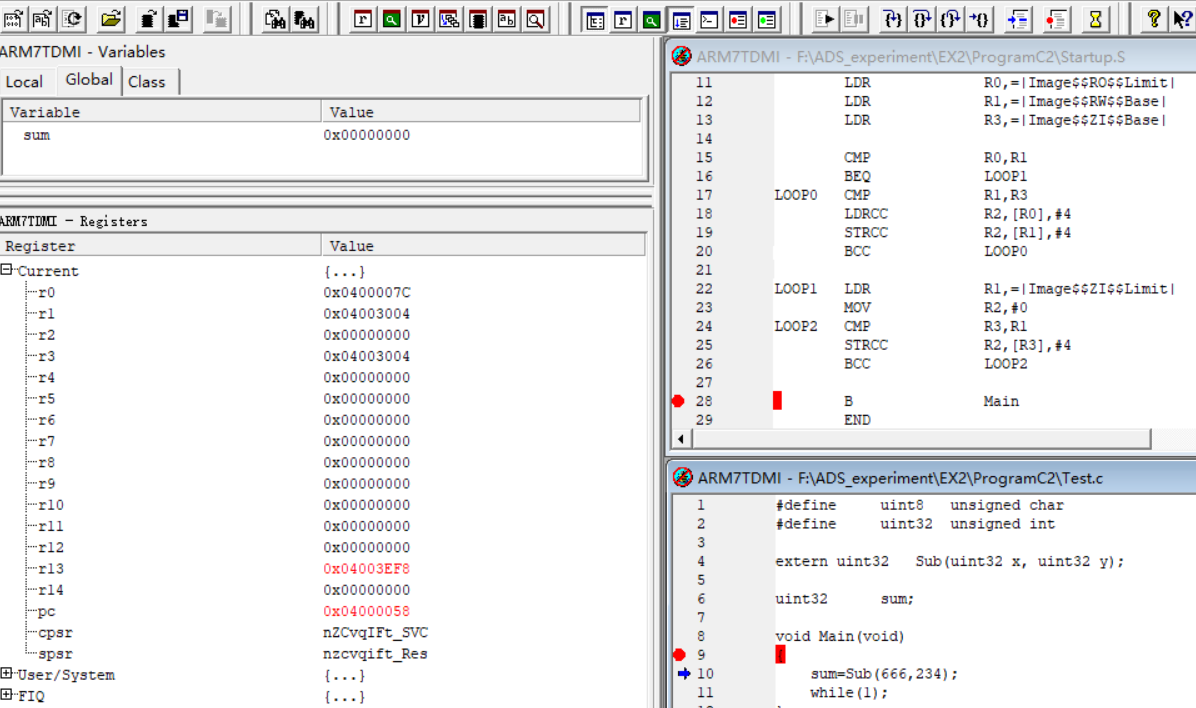


图2 程序运行图

**步骤7：**观察全局变量的值，判断程序的运算结果是否正确

选择Processor View🡪Variables命令，打开变量观察窗口，观察全局变量的值，单步运行程序，判断程序的运算结果是否正确，如图3、图4所示。

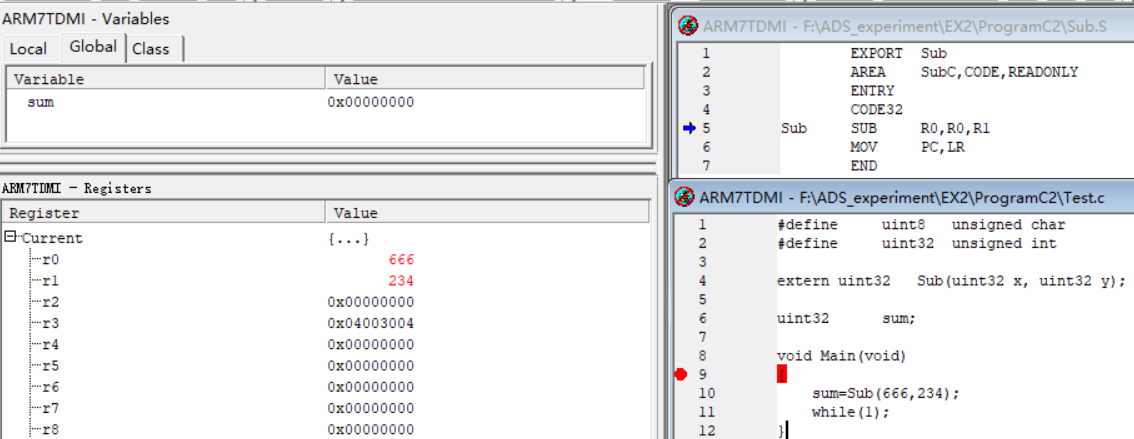


图3 程序变量运行图1

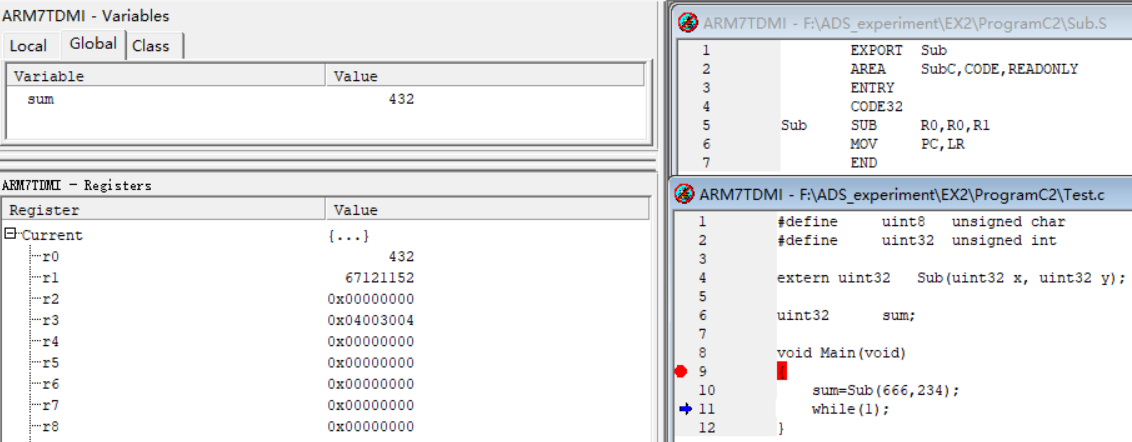


图4 程序变量运行图2

1. 心得与体会

在从C语言程序跳转到汇编语言程序时，会自动将C语言程序的下一条指令地址存放在在R14（LR），在结束汇编语言程序的时候记得要将LR传送给PC，以保证程序能正确返回。测试数据666-234=432正确。

通过本次实验熟悉和掌握ARM汇编指令中的常用指令及ADS1.2开发环境的使用方法，对ARM汇编语言和C语言混合编程的方式等知识点有了更为直观和充分的了解，对使用ADS1.2进行ARM软件编程开发有了更为深入的了解 。

1. 附录

Startup.S

        IMPORT          |Image$$RO$$Limit|      *; 引入其他源文件的定义，区分大小写*

        IMPORT          |Image$$RW$$Base|

        IMPORT          |Image$$RW$$Limit|

        IMPORT          |Image$$ZI$$Base|

        IMPORT          |Image$$ZI$$Limit|

        IMPORT          Main             *; 引入C程序中Main函数的定义*

        AREA            Start,CODE,READONLY     *; 声明一段只读代码，名字为Start*

        ENTRY                                   *; 指定程序的入口点*

        CODE32 *; 32 位的 ARM 指令*

Reset   LDR             SP,=0x4003F00           *; 将#0x04003F00加载到SP*

        LDR             R0,=|Image$$RO$$Limit|  *; 将#0x04000080加载到R0*

        LDR             R1,=|Image$$RW$$Base|   *; 将#0x04003000加载到R1*

        LDR             R3,=|Image$$ZI$$Base|   *; 将#0x04003000加载到R3*

        CMP             R0,R1                   *; 减法比较R0和R1的值,R0-R1为负数*

        BEQ             LOOP1                   *; 上条指令N=1,Z=0,此处不跳转*

LOOP0   CMP             R1,R3                   *; 减法比较R1和R3的值,R1-R3=0*

        LDRCC           R2,[R0],#4              *; 上条指令Z=1,N=0,C=1,CC条件是C=0不执行*

        STRCC           R2,[R1],#4              *; 保证Image$$RW$$Base≥Image$$ZI$$Base*

        BCC             LOOP0                   *; 同上条*

LOOP1   LDR             R1,=|Image$$ZI$$Limit|  *; 将#0x04003004加载到R3*

        MOV             R2,#0                   *; 0->R2*

LOOP2   CMP             R3,R1                   *; 减法比较R3和R1的值,R3-R1<0，产生借位,N=1,Z=0,C=0*

        STRCC           R2,[R3],#4              *; CC条件为C=0即R3比R1小，则R3+4写回R3*

        BCC             LOOP2                   *; 满足CC条件，再来一遍LOOP2知道R3=R1*

        B               Main                    *; 跳转到C程序的Main函数继续执行*

        END                             *; 指示本源程序结束*

Test.c

#define     uint8   unsigned char   *// 宏定义*

#define     uint32  unsigned int    *// 宏定义*

extern uint32   Sub(uint32 x, uint32 y);    *// 申明uint32型汇编程序Add*

uint32      sum;                    *// 声明全局变量sum*

void Main(void)

{

    sum=Sub(666,234);               *// 调用Sub汇编程序计算两数之差*

    while(1);                       *// 原地等待*

}

Sub.S

EXPORT  Sub                     *; 声明Add可以被其他文件引用，相当于全局*

        AREA    SubC,CODE,READONLY      *; 声明一段只读代码，名字为AddC*

        ENTRY                           *; 指定程序的入口点*

        CODE32                          *; 32 位的 ARM 指令*

Sub     SUB     R0,R0,R1                *; R0=R0-R1*

        MOV     PC,LR                   *; LR -> PC 用来返回程序*

        END                             *; 指示本源程序结束*

实验7：C语言程序实验

1. 实验目的

掌握在C语言程序中条用汇编程序，了解ATPCS基本规则

1. 实验内容

在C程序中调用汇编子程序，实现两个整数的加法运算。汇编子程序的原型为uint32 Add(uint32 x, uint32 y)。

其中，uint32已定义为unsigned int。

1. 实验步骤

**步骤1：**正确连接实验箱及PC机

将Multi-ICE的一端与PC机USB口正确链接，另一端与开发板正确连接。将开发板的电源线正确连接，插上电源、串口线，先不要打开开发板的电源开关。

**步骤2：**建立工程

启动CodeWarrior for ARM Developer Suite，选择File->New命令，使用ARM Executable Image工程模板建立一个工程，工程名称为ProgramC1。

**步骤3：**在工程中创建一个新文件

选择File🡪New命令，建立一个新文件Startup.S、Add.S和Test.c，设置直接添加到项目中，输入代码并保存，其中Startup.S程序同上一实验。

**步骤4：**设置地址和起始代码段

选择Edit🡪DebugRel Setting命令，在DebugRel Setting的对话框的左侧选择ARM Linker选项，然后再Output选项卡中设置链接地址，RO Base为0x4000000，RW Base为0x4003000；在Options选项卡中设置调试入口地址Image entry point为0x4000000；在Layout选项卡中设置位于开始位置的起始代码段设置为Startup.o的Start段。

**步骤5：**编译工程并仿真调试

选择Project🡪Make命令，将编译、链接整个工程，代码及错误和警告对话框如图7-1所示；然后选择Project🡪Debug命令，启动AXD进行软件仿真调试。

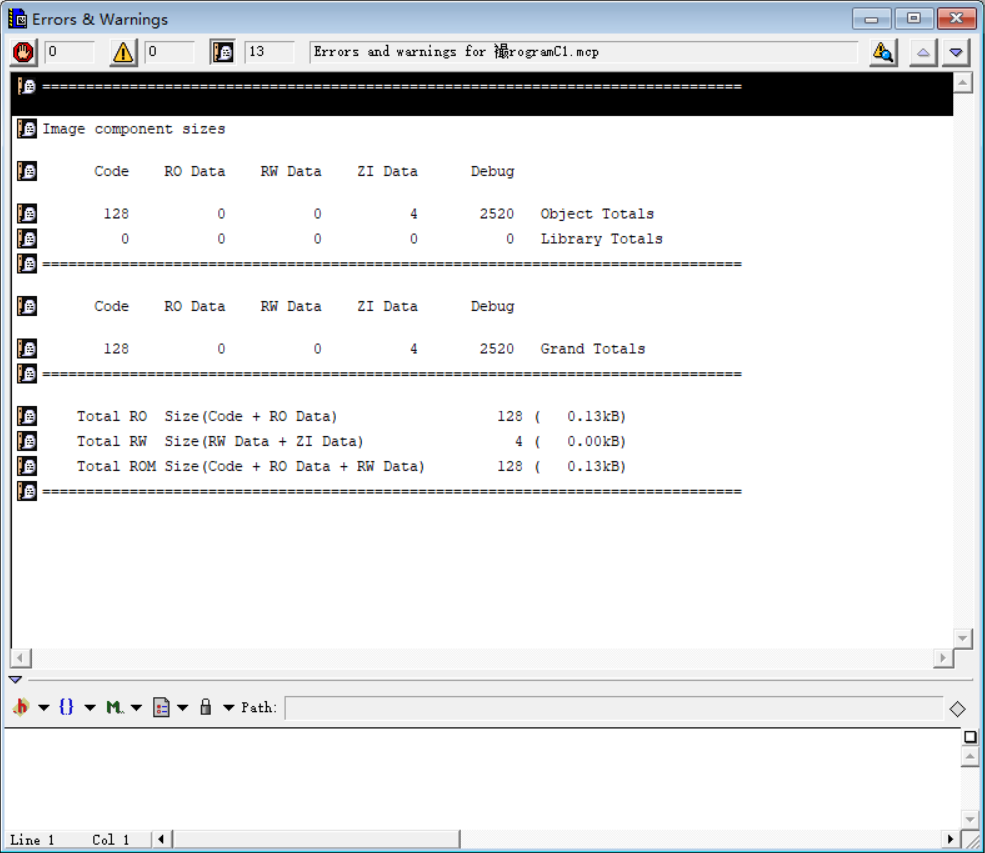


图7-1 映像文件数据

**步骤6：**设置断点并运行程序

在Startup.o的B Main处设置断点，然后全速运行程序。单步运行跳至Test.c文件，运行至sum=Add()处，再单步运行，观察程序是否跳转到汇编程序Add.S，如图7-2所示。

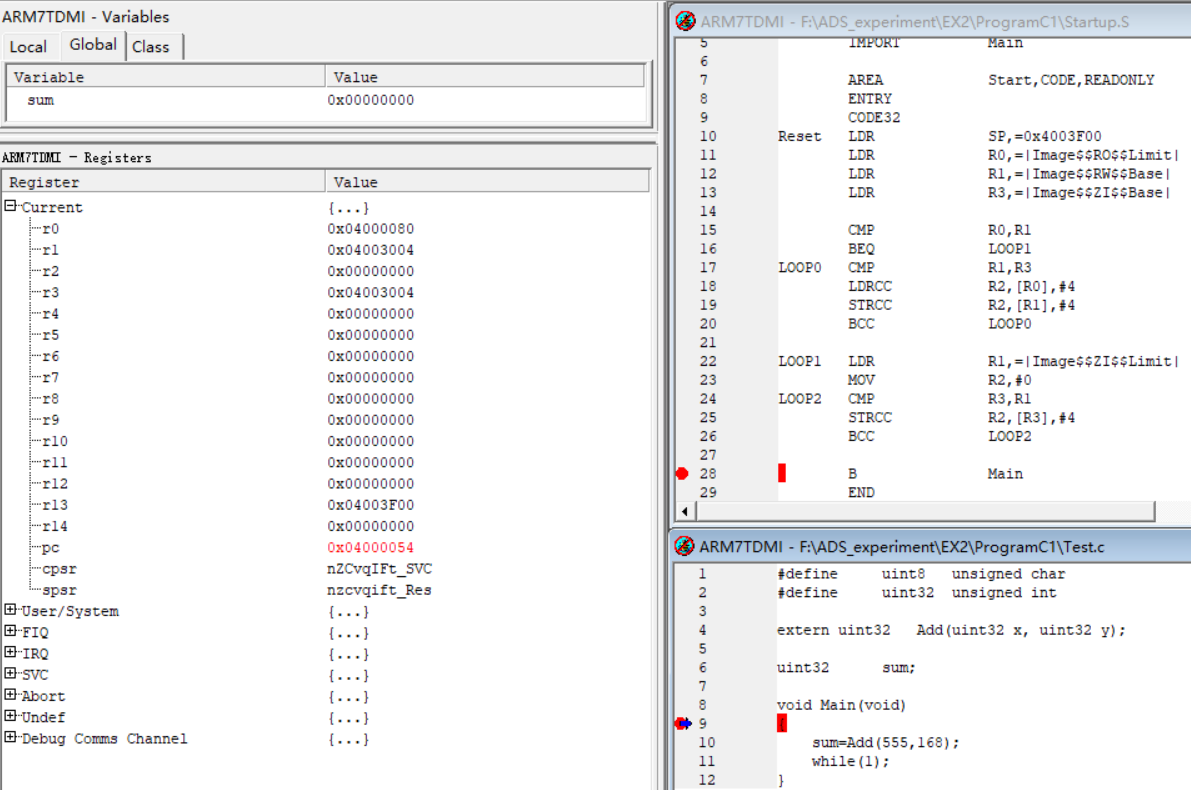


图7-2 程序运行图

**步骤7：**观察全局变量的值，判断程序的运算结果是否正确

选择Processor View🡪Variables命令，打开变量观察窗口，观察全局变量的值，单步运行程序，判断程序的运算结果是否正确，如图7-3、图7-4所示。

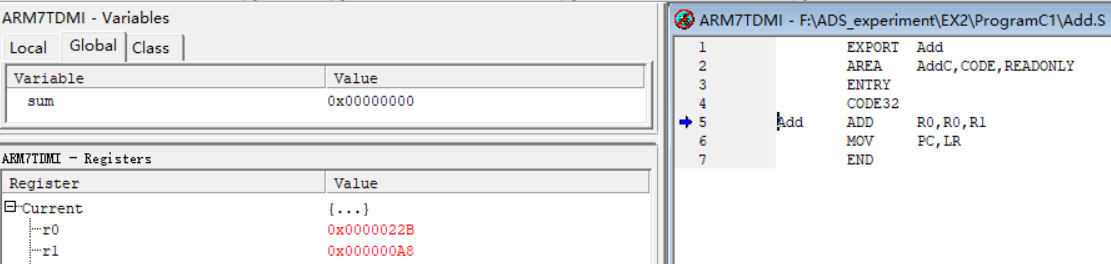


图7-3 程序变量运行图1

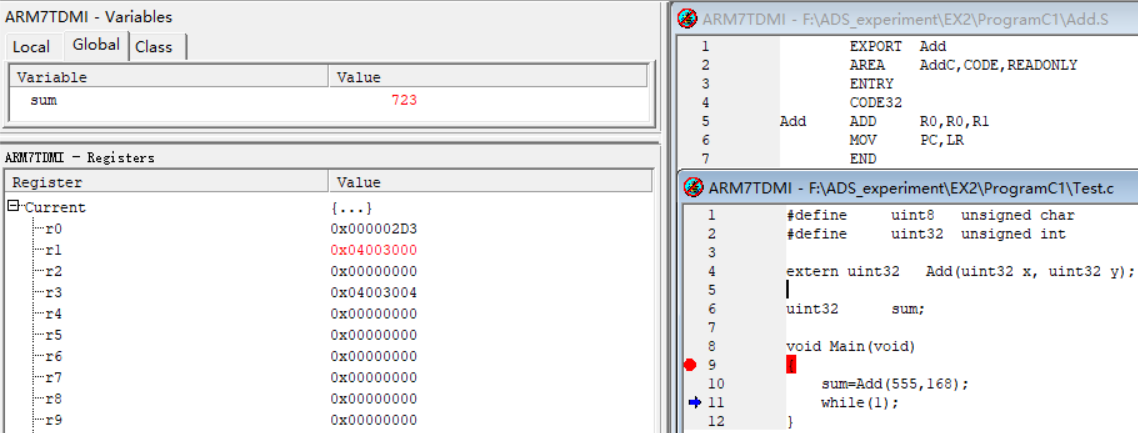


图7-4 程序变量运行图2

1. 心得与体会

在从C语言程序跳转到汇编语言程序时，会自动将C语言程序的下一条指令地址存放在在R14（LR），在结束汇编语言程序的时候记得要将LR传送给PC，以保证程序能正确返回。通过本次实验熟悉和掌握ARM汇编指令中的常用指令及ADS1.2开发环境的使用方法，对ARM汇编语言和C语言混合编程的方式等知识点有了更为直观和充分的了解，对使用ADS1.2进行ARM软件编程开发有了更为深入的了解 。

1. 附录

Startup.S

        IMPORT          |Image$$RO$$Limit|      *; 引入其他源文件的定义，区分大小写*

        IMPORT          |Image$$RW$$Base|

        IMPORT          |Image$$RW$$Limit|

        IMPORT          |Image$$ZI$$Base|

        IMPORT          |Image$$ZI$$Limit|

        IMPORT          Main             *; 引入C程序中Main函数的定义*

        AREA            Start,CODE,READONLY     *; 声明一段只读代码，名字为Start*

        ENTRY                                   *; 指定程序的入口点*

        CODE32 *; 32 位的 ARM 指令*

Reset   LDR             SP,=0x4003F00           *; 将#0x04003F00加载到SP*

        LDR             R0,=|Image$$RO$$Limit|  *; 将#0x04000080加载到R0*

        LDR             R1,=|Image$$RW$$Base|   *; 将#0x04003000加载到R1*

        LDR             R3,=|Image$$ZI$$Base|   *; 将#0x04003000加载到R3*

        CMP             R0,R1                   *; 减法比较R0和R1的值,R0-R1为负数*

        BEQ             LOOP1                   *; 上条指令N=1,Z=0,此处不跳转*

LOOP0   CMP             R1,R3                   *; 减法比较R1和R3的值,R1-R3=0*

        LDRCC           R2,[R0],#4              *; 上条指令Z=1,N=0,C=1,CC条件是C=0不执行*

        STRCC           R2,[R1],#4              *; 保证Image$$RW$$Base≥Image$$ZI$$Base*

        BCC             LOOP0                   *; 同上条*

LOOP1   LDR             R1,=|Image$$ZI$$Limit|  *; 将#0x04003004加载到R3*

        MOV             R2,#0                   *; 0->R2*

LOOP2   CMP             R3,R1                   *; 减法比较R3和R1的值,R3-R1<0，产生借位,N=1,Z=0,C=0*

        STRCC           R2,[R3],#4              *; CC条件为C=0即R3比R1小，则R3+4写回R3*

        BCC             LOOP2                   *; 满足CC条件，再来一遍LOOP2知道R3=R1*

        B               Main                    *; 跳转到C程序的Main函数继续执行*

        END                             *; 指示本源程序结束*

Test.c

#define     uint8   unsigned char   *// 宏定义*

#define     uint32  unsigned int    *// 宏定义*

extern uint32   Add(uint32 x, uint32 y);    *// 申明uint32型汇编程序Add*

uint32      sum;                    *// 声明全局变量sum*

void Main(void)

{

    sum=Add(555,168);               *// 调用Add汇编程序计算两数之和*

    while(1);                       *// 原地等待*

}

Add.S

EXPORT  Add                     *; 声明Add可以被其他文件引用，相当于全局*

        AREA    AddC,CODE,READONLY      *; 声明一段只读代码，名字为AddC*

        ENTRY                           *; 指定程序的入口点*

        CODE32                          *; 32 位的 ARM 指令*

Add     ADD     R0,R0,R1                *; R0=R0+R1*

        MOV     PC,LR                   *; LR -> PC 用来返回程序*

        END                             *; 指示本源程序结束*