

# Homework1 初识模拟器

姓名：胡成成

学号：XXXXXXXXXX

## Problem

请在keil, 或segger模拟器调试运行一个C (C++) 程序：

- 列出编译调试过程的问题与解决措施
- 提交程序源码

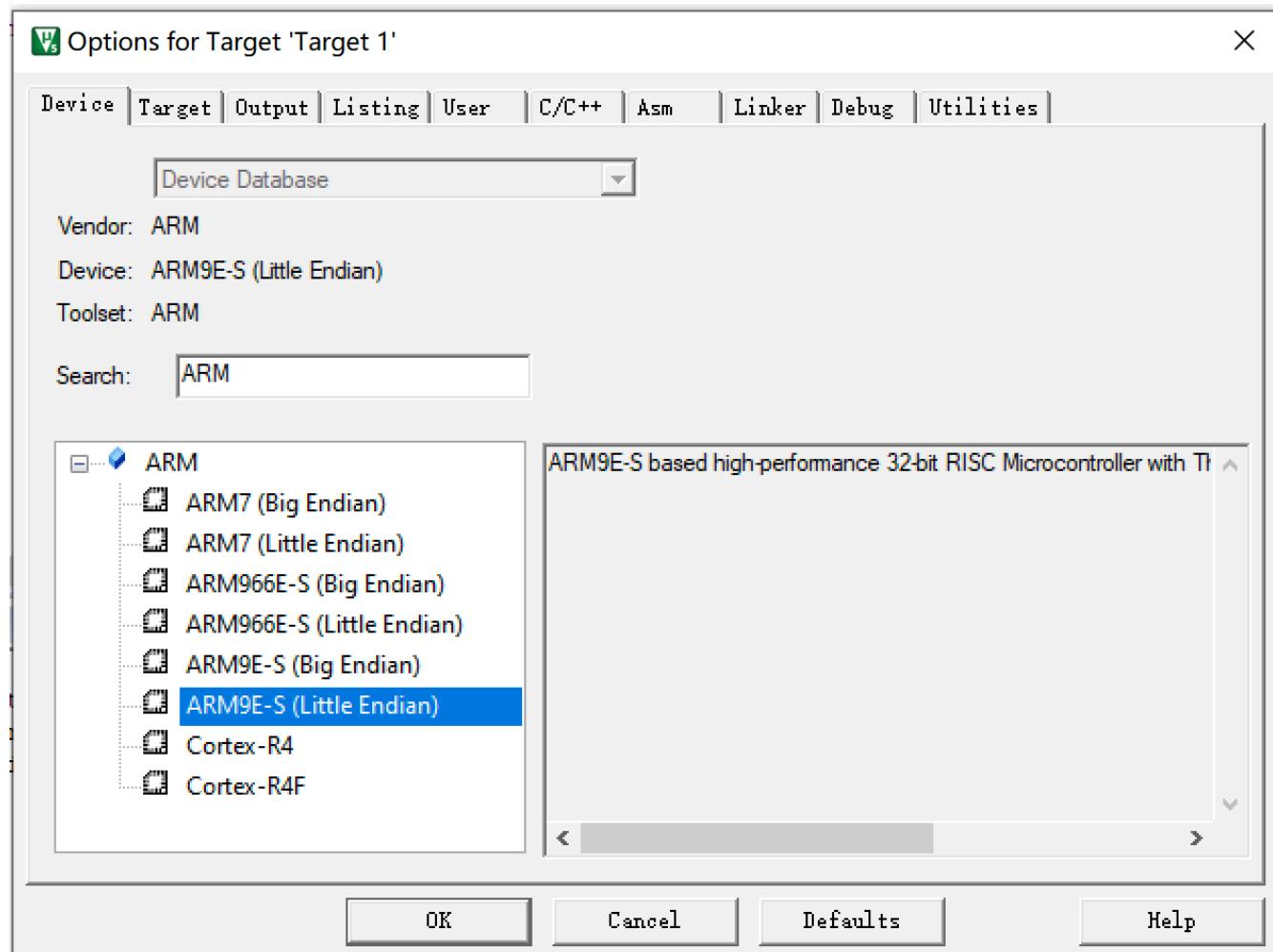
## Answer

### Keil

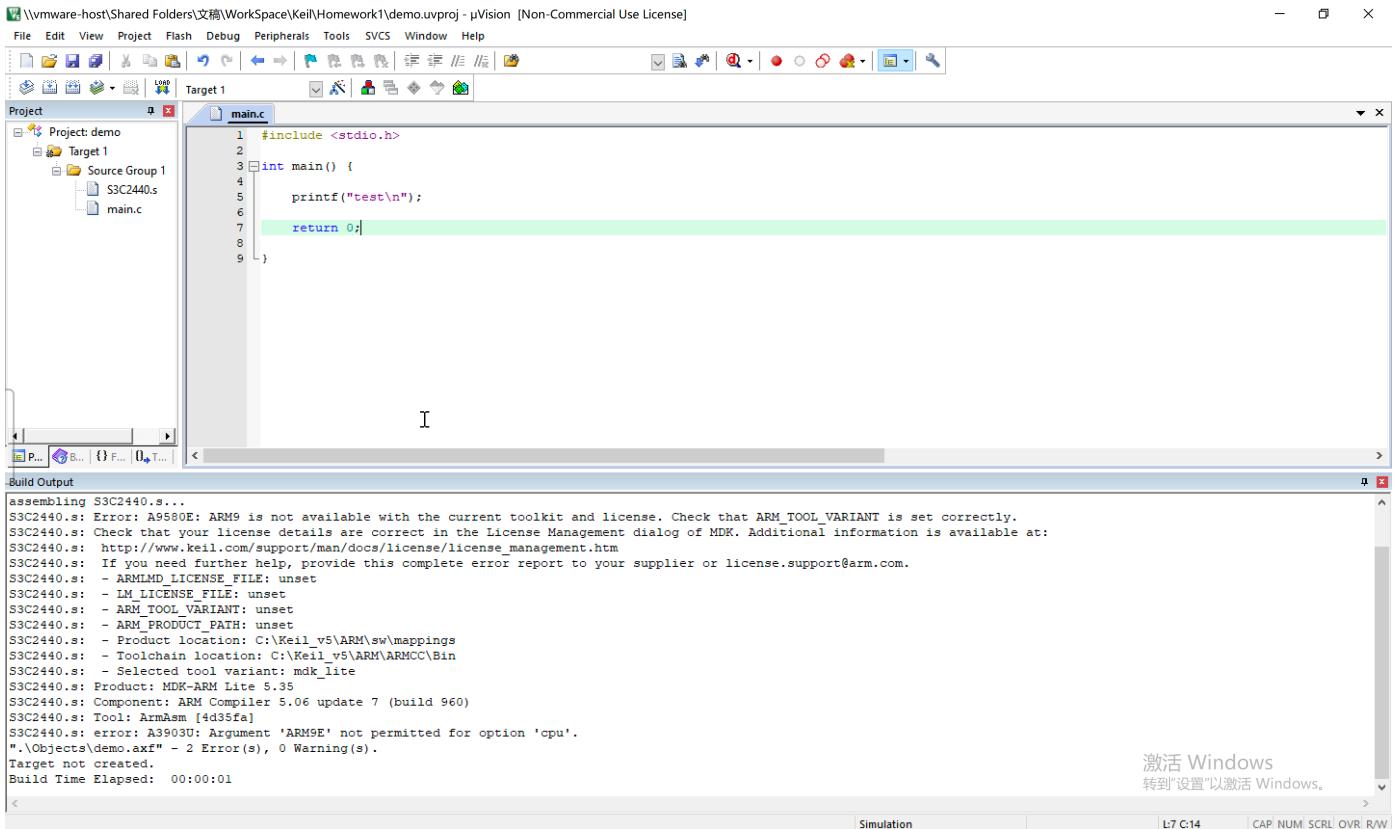
针对该问题，首先下载了keil进行测试，从keil官网 (<https://www.keil.com/>) 上下载安装MDK-ARM版的Keil。

为了支持ARM9，于是查阅网上资料找到这篇[博客](#)，根据该内容安装了ARM9的Device。

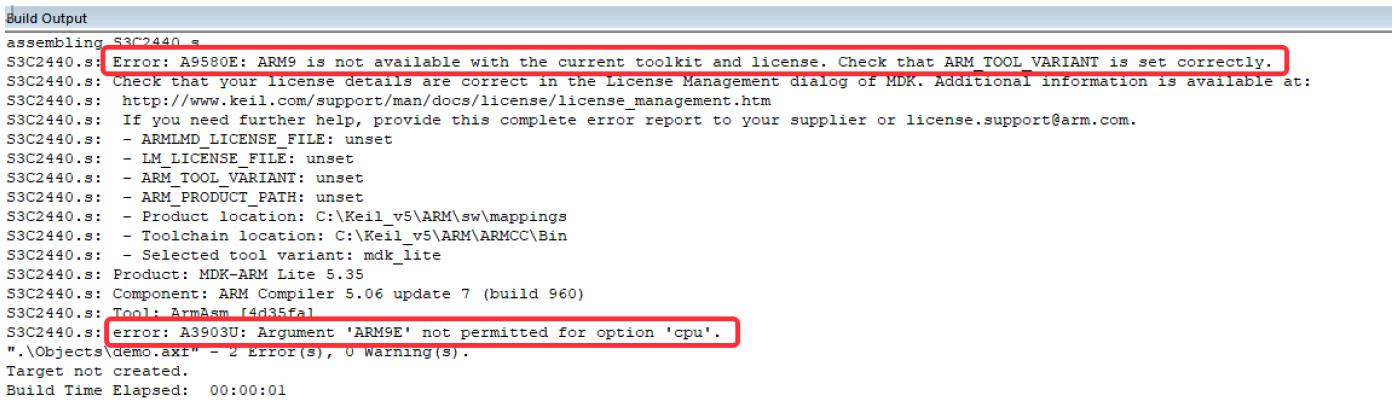
于是新建项目选择Arm9的Device进行最基础的代码测试：



运行一个最基础的printf函数：



发现报错信息，并且尝试根据报错信息去搜索资料，并没有查到资料解决此问题，Keil软件的不熟悉让我难以下手，遇到该问题暂时无法解决：



## Segger

于是，选择试一试老师提供的另一个工具[Segger Embedded Studio](#)，在官网下载安装，同时观察到官网提供了软件的文档[Document](#)

根据文档中的Getting Start部分进行简单学习：

# Getting Started

You will need to install a CPU support package:

Choose **Tools > Package Manager**

Choose the CPU support packages you wish to install and complete the dialog.

You will need to create a project:

Choose **File > New Project**

Select the appropriate Executable project type

Specify a location for the project

Complete the dialog selecting the appropriate **Target Processor** value

You will need to build the project:

Choose **Build | Build 'Project'**

To debug on the simulator

Choose **Project | Options...** to show the project options dialog

In the **Search Options** type in **Simulator**

Choose **Simulator** for the **Target Connection** option

To debug on hardware

Choose **Project | Options...** to show the project options dialog

In the **Search Options** type in **J-Link**

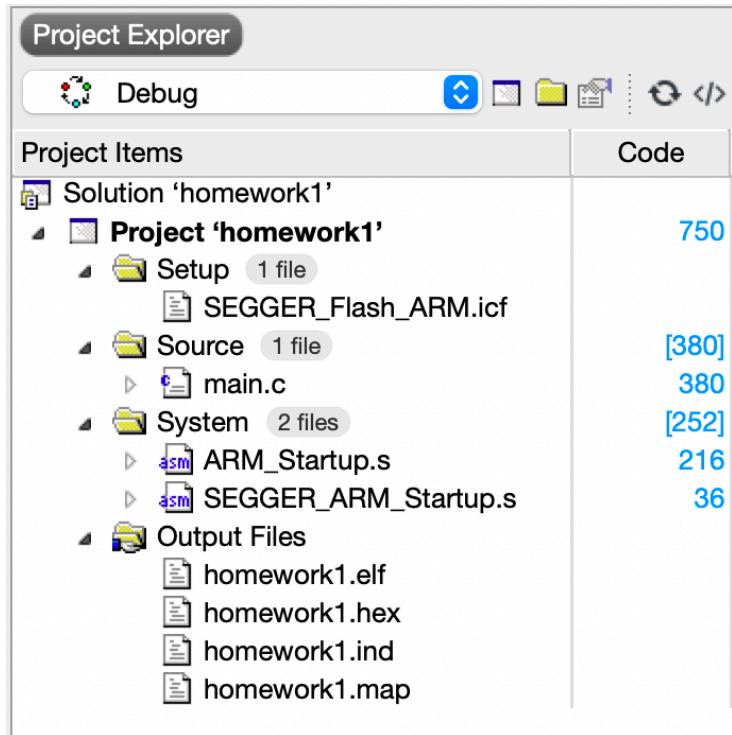
Choose **J-Link** for the **Target Connection** option

To start debugging

Choose **Debug | Go**

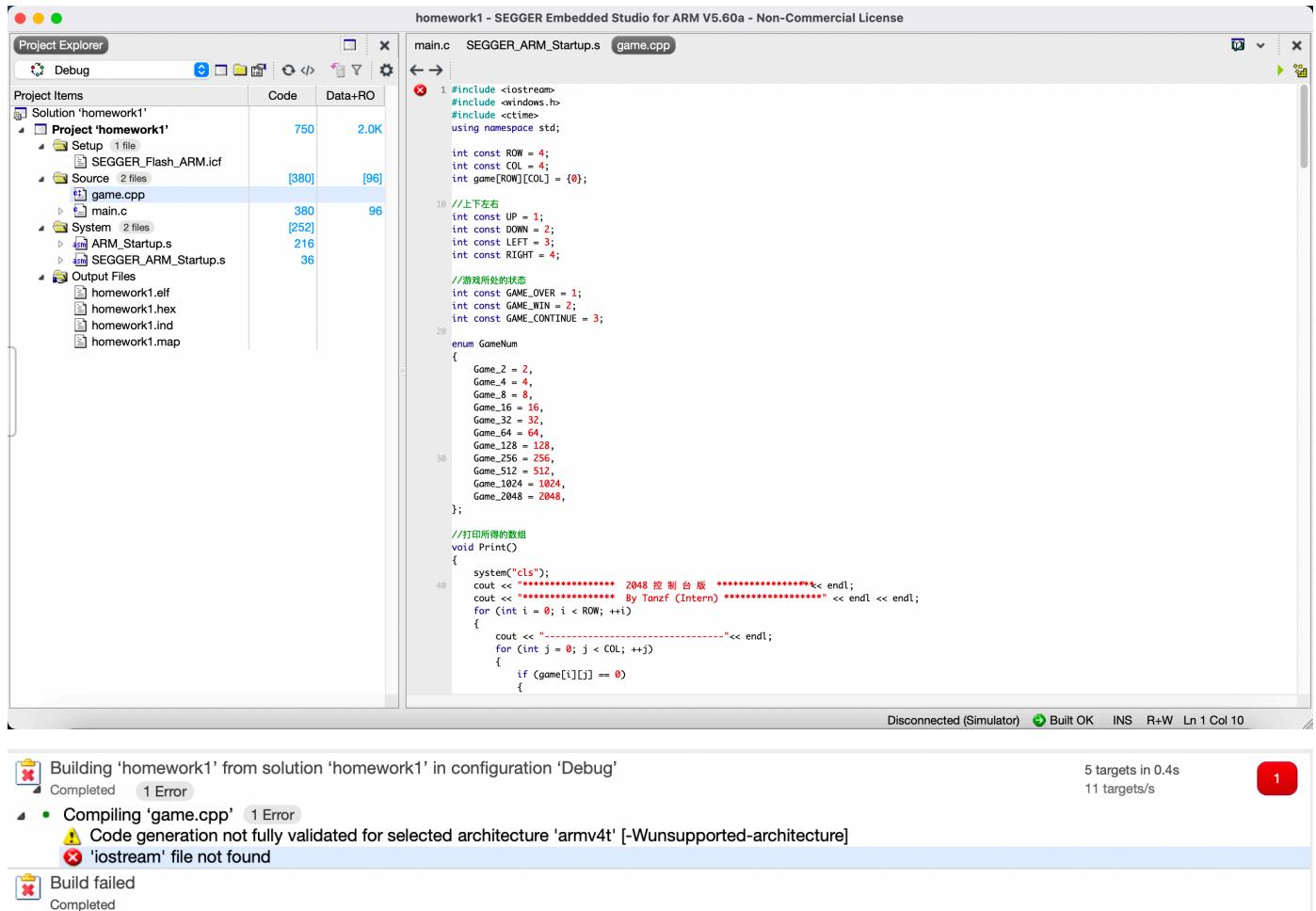
The debugger will stop the program at the main, you can now debug the application.

创建**Solution: Homework1**和**Project: Homework1**:



在main.c中 **build and debug** 默认给的hello world循环100次的程序，发现可以完美运行，总算是实现了第一步。

于是，找了大学C++程序设计课程的[小游戏代码](#)，直接报错如下：



可以看出，找不到这个库，于是在[官网文档](#) 中找到**C++ Library User Guide**

# C++ Library User Guide

SEGGER Embedded Studio for ARM provides a limited C++ library suitable for use in an embedded application.

## Standard library

The following C++ standard header files are provided in \$(StudioDir)/include:

File	Description
<cassert>	C++ wrapper on <a href="#">assert.h</a> .
<cctype>	C++ wrapper on <a href="#">ctype.h</a> .
<cerrno>	C++ wrapper on <a href="#">errno.h</a> .
<cfloat>	C++ wrapper on <a href="#">float.h</a> .
<ciso646>	C++ wrapper on <a href="#">iso646.h</a> .
<climits>	C++ wrapper on <a href="#">limits.h</a> .
<locale>	C++ wrapper on <a href="#">locale.h</a> .
<cmath>	C++ wrapper on <a href="#">math.h</a> .
<csetjmp>	C++ wrapper on <a href="#">setjmp.h</a> .
<cstdarg>	C++ wrapper on <a href="#">stdarg.h</a> .
<cstddef>	C++ wrapper on <a href="#">stddef.h</a> .
<cstdint>	C++ wrapper on <a href="#">stdint.h</a> .
<cstdio>	C++ wrapper on <a href="#">stdio.h</a> .
<cstdlib>	C++ wrapper on <a href="#">stdlib.h</a> .

825

发现并没有像在Windows上运行的C++命令窗口那样的程序，分析了一下应该是嵌入式的程序是烧录到板子里的，不支持windows.h等库

并且iostream标准输入输出需要使用cstdlib去代替。

就此问题，于是还是现场写一段C的简单代码来试一试吧。（该软件的换行缩进很不好用，体验很差。。。）

于是，写了个简单的递归求解阶乘和二分插入排序算法来感受一下模拟器如何调试程序，具体代码如下：

```
#include <stdio.h>
#include <stdlib.h>

/**
 calc_n()

 input: n
 output: n!

 */
int calc_n(int n){
    if(n==1){
        return 1;
    }
}
```

```

        else{
            return calc_n(n-1)*n;
        }
    }

/***
 * sort
 */
//exchange the 2 items a and b
void swap(int a, int b){
    int tmp = a;
    a = b;
    b = tmp;
}

void BinaryInsertSort(int *arr, int len){
    for (int i = 1; i < len; ++i){
        int tmp = arr[i];
        int iHigh = i - 1;
        int iLow = 0;
        int iMid = 0;
        while (iLow <= iHigh){
            iMid = (iLow + iHigh) / 2;
            if (tmp > arr[iMid] ){
                iLow = iMid + 1;
            }
            else{
                iHigh = iMid - 1;
            }
        }
        for (int j = i; j > iMid; --j){
            arr[j] = arr[j - 1];
        }
        arr[iLow] = tmp;
    }
}

*****
*
*      main()
*
*  Function description
*  Application entry point.
*/
int main(void) {

```

```

int i;

//calculate 10!
int nn = calc_n(3);

printf("calculate n!: %d \n", nn);

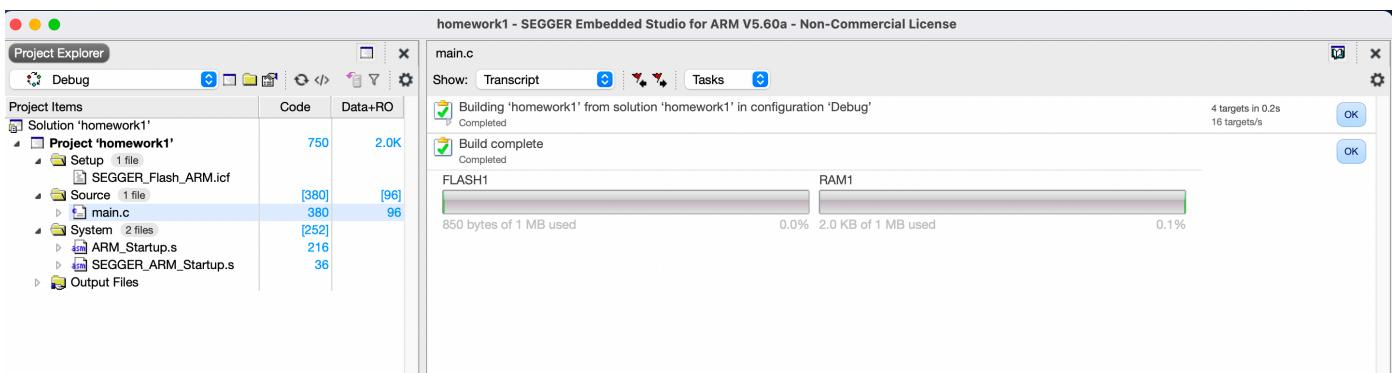
//test sort
int buf[10] = { 15, 45, 4, 7, 10, 235, 31, 52, 988, 15 };
int m = sizeof(buf);

printf("before sort: \n ");
for(i = 0; i <sizeof(buf) / sizeof(int); i++){
    printf("%d ", buf[i]);
}

BinaryInsertSort(buf, sizeof(buf) / sizeof(int));
printf("\n after sort: \n ");
for(i = 0; i <sizeof(buf) / sizeof(int); i++){
    printf("%d ", buf[i]);
}
}

```

点击build, 成功build:



其次, 进行debug, 进入debug页面:

homework1 - SEGGER Embedded Studio for ARM V5.60a - Non-Commercial License (Stopped)

**Disassembly**

```

main.c
    int main()
        arr[iLow] = tmp;
    }

    *****
    *      main()
    *      Function description
    *      Application entry point.
    */
75 int main(void) {
    int i;

    //calculate 10!
    int nn = calc_n(3);

    B580 push {r4-r5, r7, lr}
    B090 sub sp, sp, #0x40
    2000 movs r0, #0
    9001 str r0, [sp, #0x34]
    900f str r0, [sp, #0x3C]

    // main.c -- 76
    int i;
    //calculate 10!
    int nn = calc_n(3);

    2003 movs r0, #3
    000048 b1 0x00000254 <calc_n>
    900d str r0, [sp, #0x34]
    // main.c -- 80
    printf("calculate ni: %d \n", nn);
    9900 ldr r1, [sp, #0x34]
    481e add r0, sp, #12
    F000F845 b1 0x000000E0 <printf_TT_veneer_TI>

    // main.c -- 82
    //test sort
    int buf[10] = { 15, 45, 4, 7, 10, 235, 31, 52, 988, 15 };
    int m = sizeof(buf);

    printf("before sort: \n ");
    for(i = 0; i <sizeof(buf) / sizeof(int); i++){
        printf("%d ", buf[i]);
    }

```

**Locals**

Expression	Value
buf	Restricted memory ra
i	Restricted memory ra
m	Restricted memory ra
nn	Restricted memory ra

**Registers**

Name	Value
r0	0x000001cc
r1	0xcddcdcd
r2	0xcddcdcd
r3	0xcddcdcd
r4	0x00000354
r5	0xcddcdcd
r6	0xcddcdcd
r7	0xcddcdcd
r8	0xcddcdcd
r9	0xcddcdcd
r10	0xcddcdcd
r11	0xcddcdcd
r12	0xcddcdcd
sp(r13)	0x20100000
lr(r14)	0x000001d0

**Output**

Preparing target for dow  
Completed

Downloading 'homewor' 0.8 KB in 0.3s  
Download successful

**Call Stack**

Function	Call Address
int main()	0x0000003C
_SEGGER_init_done()	0x0000001CC

ARM920T on Simulator 35 Instructions Built OK INS R+W Ln 75 Col 1

点击Step Over (F10) 按钮进行debug，测试3的阶乘，结果如下：

homework1 - SEGGER Embedded Studio for ARM V5.60a - Non-Commercial License (Stopped)

**Disassembly**

```

main + 0x1a
    int main()
        arr[iLow] = tmp;
    }

    *****
    *      main()
    *      Function description
    *      Application entry point.
    */
75 int main(void) {
    int i;

    //calculate 10!
    int nn = calc_n(3);

    B580 push {r4-r5, r7, lr}
    B090 sub sp, sp, #0x40
    2000 movs r0, #0
    9001 str r0, [sp, #0x34]
    900f str r0, [sp, #0x3C]

    // main.c -- 76
    int i;
    //calculate 10!
    int nn = calc_n(3);

    2003 movs r0, #3
    000048 b1 0x00000254 <calc_n>
    900d str r0, [sp, #0x34]
    // main.c -- 80
    printf("calculate ni: %d \n", nn);
    9900 ldr r1, [sp, #0x34]
    481e add r0, sp, #12
    F000F845 b1 0x000000E0 <printf_TT_veneer_TI>

    // main.c -- 82
    //test sort
    int buf[10] = { 15, 45, 4, 7, 10, 235, 31, 52, 988, 15 };
    int m = sizeof(buf);

    printf("before sort: \n ");
    for(i = 0; i <sizeof(buf) / sizeof(int); i++){
        printf("%d ", buf[i]);
    }

```

**Locals**

Expression	Value
buf	<array>
i	0xcddcdcd
m	0xcddcdcd
nn	0x00000006

**Registers**

Name	Value
r0	0x00000011
r1	0x200ffff8
r2	0x00000316
r3	0x200ffff94
r4	0xcddcdcd
r5	0xcddcdcd
r6	0xcddcdcd
r7	0xcddcdcd
r8	0xcddcdcd
r9	0xcddcdcd
r10	0xcddcdcd
r11	0xcddcdcd
r12	0x000001e4
sp(r13)	0x200ffffb0
lr(r14)	0x00000057

**Debug Terminal**

calculate ni: 6

**Call Stack**

Function	Call Address
int main()	0x00000056
_SEGGER_init_done()	0x0000001CC

ARM920T on Simulator 132 Instructions Built OK INS R+W Ln 84 Col 1

继续debug，测试排序如下：

The screenshot shows the SEGGER Embedded Studio interface with the following components:

- Disassembly:** Shows assembly code for main.c, highlighting the main loop and sort logic.
- Code Editor:** Displays the C source code for `main.c`, including the `binaryInsertSort` function.
- Registers:** Lists CPU registers (r0-r14) with their current values.
- Locals:** Shows variable values: `buf` (array), `i` (0x00000009), `m` (0x00000028), and `nn` (0x00000006).
- Call Stack:** Shows the call stack with `int main()` at the top.
- Debug Terminal:** Displays the output of the program, showing the array before and after sorting.

程序成功运行，和预期效果一致。

## 心得

在测试过程中发现debug过程软件提供了编译的汇编代码窗口，寄存器窗口，能够实时看到Debug过程的细节，还是很不错的，在这个过程中可以体会到程序对CPU寄存器的实际操作流程与运行原理。

This screenshot is similar to the previous one, but the code editor window is explicitly highlighted with a red border. The rest of the interface (disassembly, registers, locals, call stack, debug terminal) remains visible.