中国科学技术大学
University of Science and Technology of China
信息科学技术学院

第7章　ARM程序设计

本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
- 7.4 ARM汇编语言与C/C++的混合编程

ARM程序设计

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 目前基于ARM处理器的程序大多采用C语言开发
  - 无操作系统(while语句无限循环体内功能实现+中断)
  - 有操作系统（μC/OS、Linux、Android 进行任务调度）

- 特定场合下必须使用汇编语言（系统启动程序）

汇编指令可以直接对ARM处理器中的寄存器进行操作，掌握必要的汇编程序设计知识，就能更全面、更深入地理解ARM处理器的工作原理

本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
  - 7.1.1 常用ARM程序开发环境
  - 7.1.2 MDK开发环境简介
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
- 7.4 ARM汇编语言与C/C++的混合编程

ARM程序开发环境

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 用户选用ARM处理器开发嵌入式系统时，选择合适的开发工具可以加快开发进度，节省开发成本。

- 一套含有编辑软件、编译软件、汇编软件、链接软件、调试软件、工程管理及函数库的集成开发环境（IDE）一般来说是必不可少的。

- 嵌入式实时操作系统、评估板等其他开发工具则可以根据应用软件规模和开发计划选用。

本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
  - 7.1.1 常用ARM程序开发环境
  - 7.1.2 MDK开发环境简介
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
- 7.4 ARM汇编语言与C/C++的混合编程

7.1.1 常用ARM程序开发环境

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

ARM程序开发环境主要分为基于Windows平台和基于Linux平台的两大类。

- 基于Windows平台的ARM程序开发环境主要有SDT、ADS、RVDS、RealView MDK、ARM Development Studio 5等。
- 基于Linux平台的ARM程序开发环境主要是ARM-Linux-GCC。

目前使用最多的是基于Windows平台的开发工具。

7.1.1 常用ARM程序开发环境

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- SDT
  - ARM Software Development Kit是ARM公司最早推出的开发工具
  - 目前已停止更新
- ADS
  - ARM Developer Suite，ARM公司大约在1999年推出，用来代替SDT。
  - 目前已停止更新

7.1.1 常用ARM程序开发环境

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- RVDS
  - RealView Developer Suite，是ARM公司继ADS之后推出的集成开发工具，RVDS为从事SoC、FPGA、和ASIC设计的工程师开发复杂的嵌入式应用和平台接口而设计的。
  - 目前已停止更新

7.1.1 常用ARM程序开发环境

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- MDK
  - MicroController Development Kit
  - 由Keil公司推出。Keil公司是一家业界领先的微控制器（MCU）软件开发工具的独立供应商，最流行的单片机开发工具Keil C51就是Keil公司出品的。MDK主要特点：

（1）支持内核： ARM7，ARM9，Cortex-M4/M3/M1，Cortex-R0/R3/R4等ARM微控制器内核，后续可能变化；

（2）IDE： μVision IDE；

（3）编译器：ARM C/C++ 编译器(armcc)；

（4）仿真器： μVision CPU & Peripheral Simulation；

（5）硬件调试单元：ULink /JLink。

7.1.1常用ARM程序开发环境

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- MDK

  2005年Keil公司被ARM公司收购之后，ARM公司的开发工具从此分为两大分支：MDK系列和RVDS系列。MDK系列是ARM 公司推荐的针对微控制器，或者基于单核ARMTDMI、Cortex-M或者Cortex-R处理器的开发工具链，基于Keil公司一直使用的μVision集成开发环境；而RVDS系列（后升级为DS-5）包含全部功能，支持所有ARM内核。

7.1.1 常用ARM程序开发环境

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- ARM-Linux-GCC
  - GNU Compiler Collection（GCC）是一套由GNU开发的编译器集，不仅支持C语言编译，还支持C++、Ada、Object C等许多语言。GCC还支持多种处理器架构，包括X86、ARM、和MIPS等处理器架构，是在Linux平台下被广泛使用的软件开发工具。

GNU：是“GNU is Not Unix”的递归缩写，是一个自由软件工程项目。这些软件在GNU通用公共许可的保护下允许任何人免费使用和传播（但必须同时提供源程序），GNU软件许可相当宽松，有很多公司利用GNU软件进行商业活动。

7.1.1 常用ARM程序开发环境

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- ARM-Linux-GCC
  - ARM-Linux-GCC是基于ARM目标机的交叉编译软件，所谓交叉编译简单来说就是在一个平台上生成另一个平台上的可执行代码。

所谓的平台，实际上包含两个概念：体系结构（Architecture）和操作系统（Operating System）

同一个体系结构可以运行不同的操作系统；同样，同一个操作系统也可以在不同的体系结构上运行。

一个常见的例子是：嵌入式软件开发人员通常在个人计算机上为运行在基于ARM、PowerPC或MIPS的目标机编译软件。

7.1.1 常用ARM程序开发环境

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- ARM Development Studio 5（DS-5）
  - DS-5是一款支持开发所有ARM内核芯片的集成开发环境，主要特点如下：

（1）支持内核： 全部；

（2）定制的Eclipse IDE；

（3）编译器：ARM Compiler 6、ARM Compiler 5、GCC (Linaro GNU GCC Compiler for Linux)；

（4）调试器：DS-5调试器支持ETM 指令和数据跟踪、PTM程序跟踪；

（5）仿真器：DS-5支持ULINK2、ULINKPRO和DSTREAM仿真器。

7.1.1 常用ARM程序开发环境

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- ARM Development Studio 5（DS-5）
  - MDK是用于满足开发者基于ARM7/9，ARM Cortex-M处理器的开发需求，包括它自带的RTX实时操作系统和中间库，都是属于MCU应用领域的。
  - DS-5是用于创建Linux/Andriod的复杂嵌入式系统应用和系统平台驱动接口，DS-5支持设备添加，包括多核调试和支持，主要针对复杂的多核调试、片上系统开发而推出的。

如果你要做MCU应用，推荐用KEIL MDK；如果你要做片上系统、Linux/Android驱动和应用开发，推荐使用DS-5+DSTREAM。用户可以根据自己的功能需求、使用习惯（比如很多从单片机开发转到嵌入式开发的开发者更习惯使用MDK）、开发用途等选择不同的开发环境。

本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
  - 7.1.1 常用ARM程序开发环境
  - 7.1.2 MDK开发环境简介
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
- 7.4 ARM汇编语言与C/C++的混合编程

7.1.2 MDK开发环境简介

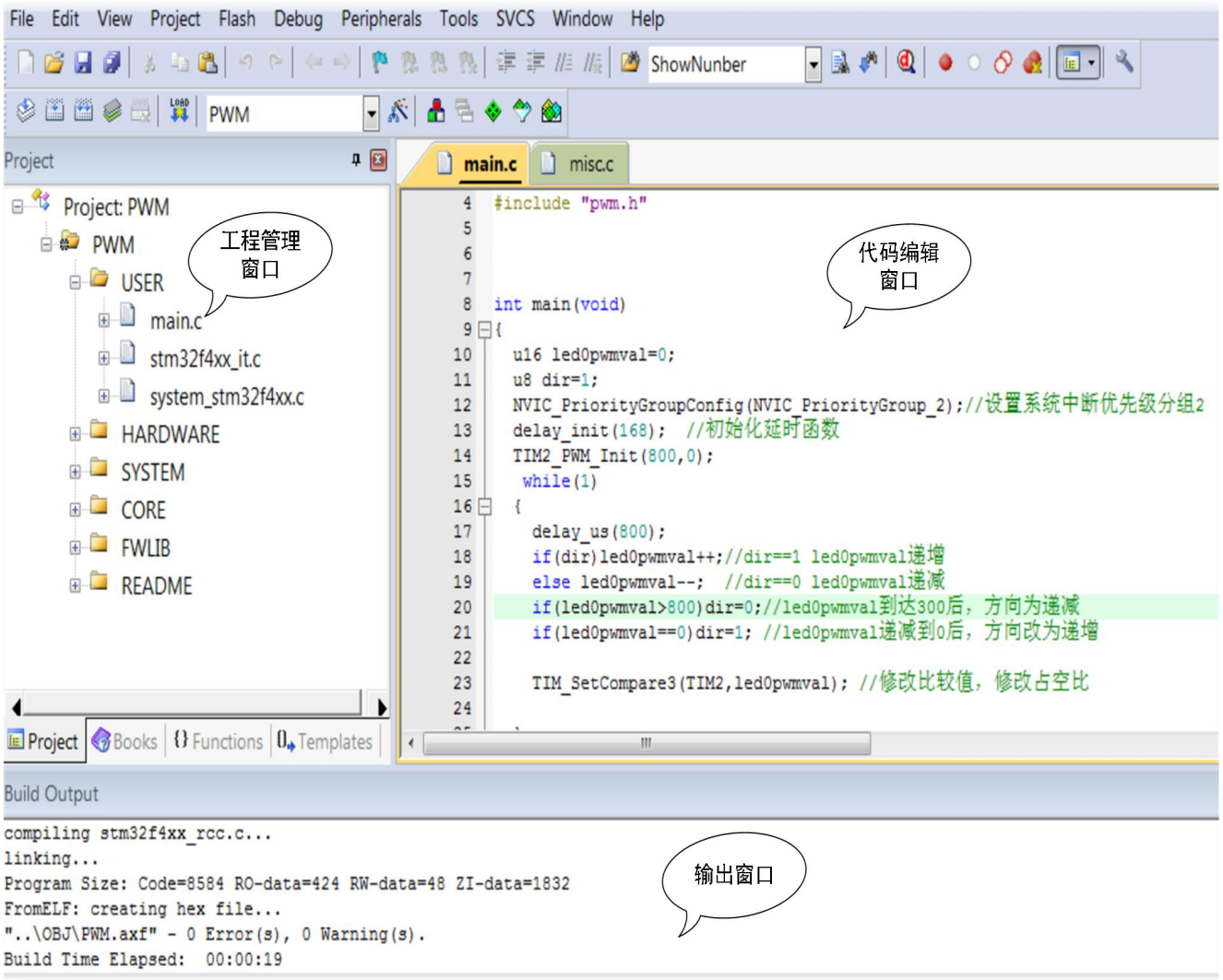
中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- ARM-MDK
  - MDK是德国知名软件公司KEIL（现已并入ARM 公司）开发的微控制器软件开发平台，是目前基于ARM内核开发的主流工具之一。
  - KEIL提供了包括C编译器、宏汇编、连接器、库管理和一个功能强大的仿真调试器在内的完整开发方案，通过一个集成开发环境（μVision）将这些功能组合在一起，它的界面和常用的微软VC++的界面相似，界面友好，在调试程序、软件仿真方面也有很强大的功能。

7.1.2 MDK开发环境简介

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

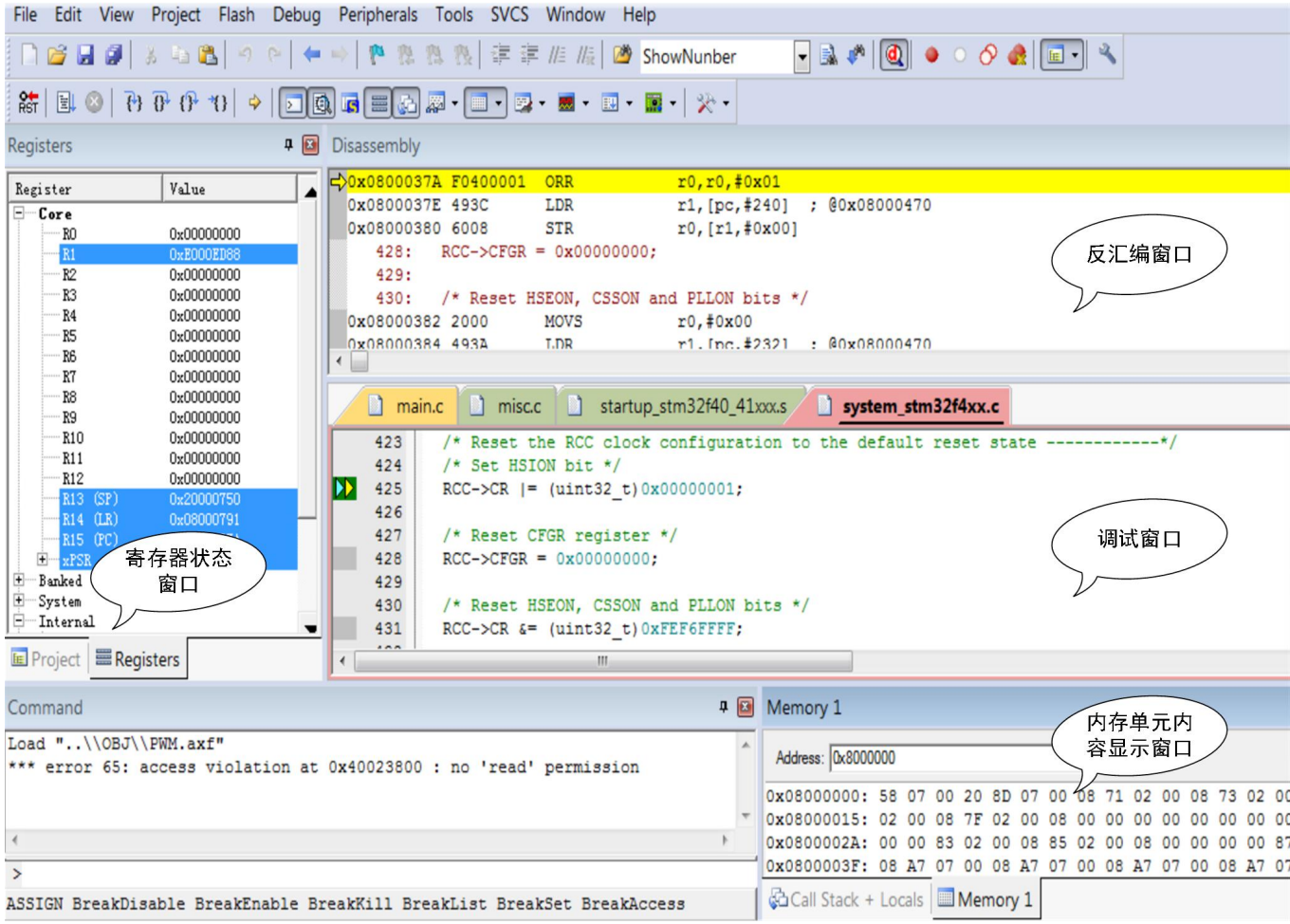
- μVision
  - μVision IDE是一款集编辑，编译和项目管理于一身的基于窗口的软件开发环境。μVision集成了C语言编译器，宏编译，链接/定位，以及HEX文件产生器。



7.1.2 MDK开发环境简介

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

一个MDK工程在调试状态时的典型窗口界面。



7.1.2 MDK开发环境简介

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- ARM仿真器
  - 仿真器可以替代目标系统中的MCU，仿真其运行。它运行起来和实际的目标处理器一样，但是增加了其它功能，使用户能够通过计算机或其它调试界面来观察MCU中的程序和数据，并控制MCU的运行，它是调试嵌入式软件的一个经济、有效的手段。具有以下优点：

(1) 不使用目标系统或CPU资源；

(2) 硬件断点、跟踪功能(TRACE)；

(3) 条件触发；

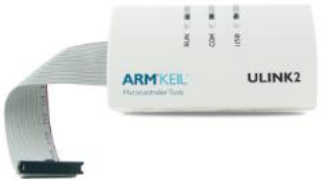
(4) 实时显示存储器和I/O口内容；

(5) 硬件性能分析；

7.1.2 MDK开发环境简介

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- ARM仿真器
  - ARM官方仿真器有以下几款：DSTREAM、RVI & RVT2、ULINKPRO、ULINK2、ULINK-ME、ULINK等，用于低端嵌入式微控制器的话，用ULINK2、ULINKPpro即可。



ULINK2

ULINKPRO

7.1.2 MDK开发环境简介

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- ARM仿真器

用于高端的嵌入式微控制器如ARM9、ARM11、Cortex-A系列等，就需要用到DSTREAM仿真器。

7.1.2 MDK开发环境简介

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- ARM仿真器
  - JTAG仿真器也称为JTAG调试器，是通过ARM芯片的JTAG边界扫描口进行调试的设备。JTAG仿真器比较便宜，连接比较方便，而且JTAG调试的目标程序是在目标板上执行，仿真更接近于目标硬件。
  - JLink是SEGGER公司为支持仿真ARM内核芯片推出的JTAG仿真器。支持所有ARM7/ARM9/ARM11，Cortex M0/M1/M3/M4，Cortex A5/A8/A9等内核芯片的仿真，与Keil MDK等编译环境无缝连接，据说功能也是众多仿真器里最强悍的。

本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
- 7.4 ARM汇编语言与C/C++的混合编程

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 在ARM汇编语言程序里，有一些特殊指令助记符，这些助记符与指令系统的助记符不同，没有相对应的操作码，通常称这些特殊指令助记符为伪指令，他们所完成的操作称为伪操作。

- 伪指令不像机器指令那样在处理器运行期间由机器执行，而是汇编程序对源程序汇编期间由汇编程序处理，包括：定义变量、分配数据存储空间、控制汇编过程、定义程序入口等，这些伪指令仅在汇编过程中起作用，一旦汇编结束，伪指令的使命就完成了。

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 符号定义(Symbol Definition)伪指令

符号定义伪指令用于定义ARM汇编程序中的变量、对变量赋值以及定义寄存器的别名等操作。常见的符号定义伪指令有如下几种：

- 用于定义全局变量的GBLA、GBLL和GBLS
- 用于定义局部变量的LCLA、LCLL和LCLS
- 用于对变量赋值的SETA、SETL、SETS
- 为通用寄存器列表定义名称的RLIST

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 符号定义(Symbol Definition)伪指令

1. GBLA、GBLL和GBLS

语法格式：

GBLA（GBLL或GBLS） 全局变量名;

GBLA、GBLL和GBLS伪指令用于定义一个ARM程序中的全局变量，并将其初始化。其中：

GBLA伪指令用于定义一个全局的数字变量，并初始化为0；

GBLL伪指令用于定义一个全局的逻辑变量，并初始化为F（假）；

GBLS伪指令用于定义一个全局的字符串变量，并初始化为空；

由于以上三条伪指令用于定义全局变量，因此在整个程序范围内变量名必须唯一。

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 符号定义(Symbol Definition)伪指令

1. GBLA、GBLL和GBLS

［例］

GBLA Test1 ;定义一个全局的数字变量，变量名为Test1

Test1 SETA 0xaa ;将该变量赋值为0xaa

GBLL Test2 ;定义一个全局的逻辑变量，变量名为Test2

Test2 SETL {TRUE} ;将该变量赋值为真

GBLS Test3 ;定义一个全局的字符串变量，变量名为Test3

Test3 SETS “Testing” ;将该变量赋值为“Testing”

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 符号定义(Symbol Definition)伪指令

2. LCLA、LCLL和LCLS

语法格式：

LCLA（LCLL或LCLS） 局部变量名;

LCLA、LCLL和LCLS伪指令用于定义一个ARM程序中的局部变量，并将其初始化。其中：

LCLA伪指令用于定义一个局部的数字变量，并初始化为0；

LCLL伪指令用于定义一个局部的逻辑变量，并初始化为F（假）；

LCLS伪指令用于定义一个局部的字符串变量，并初始化为空；

以上三条伪指令用于声明局部变量，在其作用范围内变量名必须唯一。

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 符号定义(Symbol Definition)伪指令

2. LCLA、LCLL和LCLS

[例]

```
LCLA Test4 ;声明一个局部的数字变量，变量名为Test4
Test3 SETA 0xaa ;将该变量赋值为0xaa
LCLL Test5 ;声明一个局部的逻辑变量，变量名为Test5
Test4 SETL {TRUE} ;将该变量赋值为真
LCLS Test6 ;定义一个局部的字符串变量，变量名为Test6
Test6 SETS “Testing” ;将该变量赋值为“Testing”
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 符号定义(Symbol Definition)伪指令

3. SETA、SETL和SETS

语法格式：

变量名 SETA（SETL或SETS） 表达式；

伪指令SETA、SETL、SETS用于给一个已经定义的全局变量或局部变量赋值。

SETA伪指令用于给一个数学变量赋值；

SETL伪指令用于给一个逻辑变量赋值；

SETS伪指令用于给一个字符串变量赋值；

其中，变量名为已经定义过的全局变量或局部变量，表达式为将要赋给变量的值。

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 符号定义(Symbol Definition)伪指令

3. SETA、SETL和SETS

[例]

```
LCLA Test3 ;声明一个局部的数字变量，变量名为Test3
Test3 SETA 0xaa ;将该变量赋值为0xaa
LCLL Test4 ;声明一个局部的逻辑变量，变量名为Test4
Test4 SETL {TRUE} ;将该变量赋值为真
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 符号定义(Symbol Definition)伪指令

4. RLIST

语法格式:

名称 RLIST {寄存器列表};

RLIST伪指令可用于对一个通用寄存器列表定义名称，使用该伪指令定义的名称可在ARM指令LDM/STM中使用。在LDM/STM指令中，列表中的寄存器访问次序为根据寄存器的编号由低到高，而与列表中的寄存器排列次序无关。

[例]

RegList RLIST {R0-R5, R8, R10} ;将寄存器列表名称定义为RegList，可在ARM指令LDM/STM中通过该名称访问寄存器列表

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

数据定义(Data Definition)伪指令

数据定义伪指令一般用于为特定的数据分配存储单元，同时可完成已分配存储单元的初始化。常见的数据定义伪指令有如下几种：

- DCB 用于分配一段连续的字节存储单元并用指定的数据初始化
- DCW（DCWU）用于分配一段连续的半字存储单元并用指定的数据初始化
- DCD（DCDU）用于分配一段连续的字存储单元并用指定的数据初始化
- DCFD（DCFDU）用于为双精度的浮点数分配一段连续的字存储单元并用指定的数据初始化

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 数据定义(Data Definition)伪指令
  - DCFS （DCFSU）用于为单精度的浮点数分配一段连续的字存储单元并用指定的数据初始化
  - DCQ （DCQU）用于分配一段以8字节为单位的连续的存储单元并用指定的数据初始化
  - SPACE 用于分配一段连续的存储单元
  - MAP 用于定义一个结构化的内存表首地址
  - FIELD 用于定义一个结构化的内存表的数据域

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 数据定义(Data Definition)伪指令

1. DCB

语法格式:

标号 DCB 表达式;

DCB伪指令用于分配一段连续的字节（8位）存储单元并用伪指令中指定的表达式初始化。其中，表达式可以为0—255的数字或字符串。DCB也可用“=”代替。

[例]

Str DCB “This is a test!”;分配一段连续的字节存储单元并初始化

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 数据定义(Data Definition)伪指令

2. DCW （或DCWU）

语法格式：

标号 DCW （或DCWU） 表达式；

DCW （或DCWU）伪指令用于分配一段连续的半字（16位）存储单元并用伪指令中指定的表达式初始化。其中，表达式可以为程序标号或数字表达式。

用DCW分配的字存储单元是半字对齐的，而用DCWU分配的字存储单元并不严格半字对齐

[例]

DataTest DCW 1,2,3 ;分配3个连续的半字存储单元并用1，2，3初始化

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 数据定义(Data Definition)伪指令

3. DCD（或DCDU）

语法格式：

标号 DCD（或DCDU） 表达式；

DCD（或DCDU）伪指令用于分配一段连续的字（32位）存储单元并用伪指令中指定的表达式初始化。其中，表达式可以为程序标号或数字表达式，DCD也可用“&”代替。

用DCD分配的字存储单元是字对齐的，而用DCDU分配的字存储单元并不严格字对齐。

[例]

DataTest DCD 4,5,6 ;分配3个连续的字存储单元并用4，5，6初始化

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 数据定义(Data Definition)伪指令

4. DCFD （或DCFDU）

语法格式：

标号 DCFD （或DCFDU） 表达式；

DCFD （或DCFDU）伪指令用于为双精度的浮点数分配一段连续的字存储单元并用伪指令中指定的表达式初始化，每个双精度的浮点数占据两个字单元。

用DCFD分配的字存储单元是字对齐的，而用DCFDU分配的字存储单元并不严格字对齐。

[例]

FDataTest DCFD 0.1,0.2,0.3 ;分配3个连续的双字存储单元并初始化为0.1,0.2,0.3的双精度数表达

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 数据定义(Data Definition)伪指令

5. DCFS （或DCFSU）

语法格式：

标号 DCFS （或DCFSU） 表达式;

DCFS （或DCFSU）伪指令用于为单精度的浮点数分配一段连续的字存储单元并用伪指令中指定的表达式初始化，每个单精度的浮点数占据一个字单元。

用DCFS分配的字存储单元是字对齐的，而用DCFSU分配的字存储单元并不严格字对齐。

[例]

FDataTest DCFS -0.1,-0.2,-0.3 ;分配3个连续的字存储单元并初始化为-0.1，-0.2，-0.3的单精度数表达

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 数据定义(Data Definition)伪指令

6. DCQ(或DCQU)

语法格式:

标号 DCQ (或DCQU) 表达式;

DCQ (或DCQU) 伪指令用于分配一段以8个字节为单位的连续存储区域并用伪指令中指定的表达式初始化。

用DCQ分配的存储单元是字对齐的，而用DCQU分配的存储单元并不严格字对齐。

[例]

DataTest DCQ 100,101,102 ;分配3个连续的双字内存单元，并用100D，101D，102D的16进制数据进行初始化

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 数据定义(Data Definition)伪指令

7. SPACE

语法格式:

标号 SPACE 表达式;

SPACE伪指令用于分配一片连续的存储区域并初始化为0。其中，表达式为要分配的字节数。SPACE也可用“%”代替。

[例]

DataSpace SPACE 100 ;分配连续100字节的存储单元并初始化为0

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 数据定义(Data Definition)伪指令

8. MAP和FIELD

伪指令MAP和FIELD经常结合在一起使用。MAP用于定义一个结构化的内存表的首地址，可以用“^”代替。

语法格式：

MAP 表达式{，基址寄存器}；

表达式可以为程序中的标号或数学表达式，基址寄存器为可选项，当基址寄存器选项不存在时，表达式的值即为内存表的首地址，当该选项存在时，内存表的首地址为表达式的值与基址寄存器的和。

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 数据定义(Data Definition)伪指令

[例]

MAP 0x100, R0 ;定义结构化内存表首地址的值为0x100＋R0。

FIELD伪指令常与MAP伪指令配合使用来定义结构化的内存表。MAP伪指令定义内存表的首地址，FIELD伪指令定义内存表中的各个数据域，并可以为每个数据域指定一个标号供其他的指令引用。

语法格式:

标号 FIELD 表达式;

表达式的值为当前数据域在内存表中所占的字节数。

注意MAP和FIELD伪指令仅用于定义数据结构，并不实际分配存储单元。

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 数据定义(Data Definition)伪指令

[例]

```
MAP 0x100 ;定义结构化内存表首地址的值为0x100。
A FIELD 16 ;定义A的长度为16字节，起始位置为0x100
B FIELD 32 ;定义B的长度为32字节，起始位置为0x110
S FIELD 256 ;定义S的长度为256字节，起始位置为0x130
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

汇编控制(Assembly Control)伪指令

汇编控制伪指令用于控制汇编程序的执行流程，常用的汇编控制伪指令包括以下几条：

- IF、ELSE、ENDIF
- WHILE、WEND
- MACRO、MEND
- MEXIT

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 汇编控制(Assembly Control)伪指令

1.IF、ELSE、ENDIF

语法格式:

IF 逻辑表达式

指令序列1

ELSE

指令序列2

ENDIF

IF、ELSE、ENDIF伪指令能根据条件的成立与否决定是否执行某个指令序列，当IF后面的逻辑表达式为真，则执行指令序列1，否则执行指令序列2。其中，ELSE及指令序列2可以没有，此时，当IF后面的逻辑表达式为真，则执行指令序列1，否则继续执行后面的指令。

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 汇编控制(Assembly Control)伪指令

1.IF、ELSE、ENDIF

[例]

```
GBLL Test ;声明一个全局的逻辑变量，变量名为Test
……
IF Test = TRUE
指令序列1
ELSE
指令序列2
ENDIF
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 汇编控制(Assembly Control)伪指令

2.WHILE、WEND

语法格式:

WHILE 逻辑表达式

指令序列

WEND

WHILE、WEND伪指令能根据条件的成立与否决定是否循环执行某个指令序列。当WHILE后面的逻辑表达式为真，则执行指令序列，该指令序列执行完毕后，再判断逻辑表达式的值，若为真则继续执行，一直到逻辑表达式的值为假。

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 汇编控制(Assembly Control)伪指令

2.WHILE、WEND

[例]

```
GBLA Counter ;声明一个全局的数学变量，变量名为Counter
Counter SETA 3 ;由变量Counter控制循环次数
……
WHILE Counter < 10
指令序列
WEND
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 汇编控制(Assembly Control)伪指令

3.MACRO、MEND

语法格式:

MACRO

{$标号} 宏名 {$参数1,$参数2,……}

指令序列

MEND

MACRO、MEND伪指令可以将一段代码定义为一个整体，这两条指令称为宏指令，需要时可以在程序中通过宏指令多次调用这段代码。其中，$标号在宏指令被展开时会被替换为用户定义的符号。宏指令可以使用一个或多个参数，当宏指令被展开时，这些参数被相应的值替换。

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 汇编控制(Assembly Control)伪指令

3.MACRO、MEND

[例]

```
 MACRO ;宏定义开始
$label test $p1,$p2,$p3 ;宏的名称为test，有三个参数p1,p2,p3
;宏的标号$label可用于构造宏定义体内的其他标号名称
 CMP $p1,$p2 ;比较参数p1,p2大小
 BHI $label.save ;无符号比较后若p1>p2，跳转到$labe.save标号处，$label.save
;为宏定义体的内部标号
 MOV $p3,$p2
 B $label.end
$label.save MOV $p3,$p1
$label.end ;宏定义功能即将参数p1,p2无符号比较后的大值存入参数p3
 MEND ;宏定义结束
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 汇编控制(Assembly Control)伪指令

3.MACRO、MEND

调用上述宏可以采用下面的方式:

```
abc  test  R0,R1,R2  ;通过宏的名称test调用宏，宏的标号为abc
;三个参数为寄存器R0,R1,R2
```

汇编处理宏时会将宏展开还原为一段代码，结果如下:

```
CMP  R0,R1
BHI  abc.save
MOV  R2,R1
B  abc.end
abc.save  MOV  R2,R0
abc.end
.....
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 其他常用的伪指令

还有一些其他的伪指令，在汇编程序中经常会被使用，包括以下几条：

- AREA
- ENTRY
- EXPORT （或GLOBAL）
- IMPORT
- END
- 略

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 其他常用的伪指令

1. AREA

语法格式:

AREA 段名 属性1，属性2，……;

AREA伪指令用于定义一个代码段或数据段，段名若以数字开头，则该段名需用“|”括起来，如|1_test|。

[例]

```
AREA RESET,CODE,READONLY
.....
;该伪指令定义了一个代码段，段名为RESET，属性为只读
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 其他常用的伪指令

2. ENTRY

语法格式：

ENTRY;

ENTRY伪指令用于指定汇编程序的入口点。在一个完整的汇编程序中至少要有一个ENTRY（也可以有多个，当有多个ENTRY时，程序的真正入口点由链接器指定），但在一个源文件里最多只能有一个ENTRY（可以没有）。

[例]

```
AREA  RESET,CODE,READONLY
ENTRY  ;指定应用程序的入口点
……
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 其他常用的伪指令

3. EXPORT

语法格式:

EXPORT 标号 {[WEAK]};

EXPORT伪指令用于在程序中声明一个全局的标号，该标号可在其他的文件中引用。

[例]

```
AREA RESET,CODE,READONLY
EXPORT Stest ;声明一个可全局引用的标号Stest
……
END
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 其他常用的伪指令

4. IMPORT

语法格式:

IMPORT 标号 {[WEAK]};

IMPORT伪指令用于通知编译器要使用的标号在其他的源文件中定义。

[例]

```
AREA   RESET,CODE,READONLY
IMPORT MAIN ;通知编译器当前文件要引用标号MAIN, 但MAIN在其他源文件中定义
……
END
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 其他常用的伪指令

5. END

语法格式:

END;

END伪指令用于通知编译器已经到了源程序的结尾。

[例]

```
AREA  RESET, CODE, READONLY
……
END  ;指定应用程序的结尾
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

汇编语言中常用的符号

在汇编语言程序设计中，经常使用各种符号代替地址、变量和常量等，以增加程序的可读性。尽管符号的命名由编程者决定，但并不是任意的，必须遵循以下的约定：

（1）符号区分大小写，同名的大、小写符号会被编译器认为是两个不同的符号。

（2）符号在其作用范围内必须唯一。

（3）自定义的符号名不能与系统的保留字相同。

（4）符号名不应与指令或伪指令同名。

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 汇编语言中常用的符号
  - 程序中的变量

1. 程序中的变量是指其值在程序的运行过程中可以改变的量。ARM（Thumb）汇编程序所支持的变量有数字变量、逻辑变量和字符串变量。

2. 数字变量用于在程序的运行中保存数字值，但注意数字值的大小不应超出数字变量所能表示的范围。

3. 逻辑变量用于在程序的运行中保存逻辑值，逻辑值只有两种取值情况：真或假。

4. 字符串变量用于在程序的运行中保存一个字符串，但注意字符串的长度不应超出字符串变量所能表示的范围。

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 汇编语言中常用的符号
  - 程序中的变量

[例]

```
LCLS  S1  ;
LCLS  S2  ;定义局部字符串变量S1和S2
S1  SETS  “Test!”  ;字符串变量S1的值为“Test!”
S2  SETS  “Hello!”  ;字符串变量S2的值为“Hello!”
```

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 汇编语言中常用的符号
  - 程序中的常量

程序中的常量是指其值在程序的运行过程中不能被改变的量。ARM（Thumb）汇编程序所支持的常量有数字常量、逻辑常量和字符串常量。

数字常量一般为32位的整数，当作为无符号数时，其取值范围为 $0\sim2^{32}-1$，当作为有符号数时，其取值范围为 $-2^{31}\sim2^{31}-1$。

逻辑常量只有两种取值情况：真或假。

ARM汇编程序中的伪指令

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 汇编语言中常用的符号
  - 程序中的常量

[例]

```
NUM EQU 64
abcd EQU 2   ;定义abcd符号的值为2
abcd EQU label+16 ;定义abcd符号的值为(label+16)
abcd EQU 0x1c, CODE32 ;定义abcd符号的值为绝对地址值0x1c，而且此处为ARM32指令
```

本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
  - 7.3.1 ARM汇编语言的语句格式
  - 7.3.2 ARM汇编语言程序结构
  - 7.3.3 ARM汇编程序设计实例
- 7.4 ARM汇编语言与C/C++的混合编程

ARM汇编语言程序设计

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 由于C语言便于理解，有大量的支持库，所以它是当前ARM程序设计所使用的主要编程语言。
- 但是对硬件系统的初始化、CPU状态设定、中断使能、主频设定以及RAM控制参数初始化等C程序力所不能及的底层操作，还是要由汇编语言程序来完成。
- ARM程序在完成上述功能时通常是C语言和汇编语言混合编程。

本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
  - 7.3.1 ARM汇编语言的语句格式
  - 7.3.2 ARM汇编语言程序结构
  - 7.3.3 ARM汇编程序设计实例
- 7.4 ARM汇编语言与C/C++的混合编程

ARM汇编语言的语句格式

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- ARM汇编中语句中所有标号必须在一行的顶格书写，其后面不要添加“:”，而所有指令均不能顶格书写。ARM汇编器对标识符大小写敏感，书写标号及指令时字母大小写要一致，在ARM汇编程序中，一个ARM指令、伪指令、寄存器名可以全部为大写字母，也可以全部为小写字母，但不要大小写混合使用。

- 注释使用“;”，注释内容由“;”开始到此行结束，注释可以在一行的顶格书写。ARM汇编语句的格式如下：

  [LABEL] OPERATION [OPERAND] [;COMMENT]

ARM汇编语言的语句格式

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

1.标号域（LABLE）

（1）标号域用来表示指令的地址、变量、过程名、数据的地址和常量。

（2）标号是可以自己起名的标识符，语句标号可以是大小写字母混合，通常以字母开头，由字母、数字、下划线等组成。

（3）语句标号不能与寄存器名、指令助记符、伪指令(操作)助记符、变量名同名。

（4）ARM汇编规范规定：语句标号必须在一行的开头书写，不能留空格；指令则在行开头必须要留出空格。

ARM汇编语言的语句格式

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

2.操作助记符域(OPERATION)

（1）操作助记符域可以为指令、伪操作、宏指令或伪指令的助记符。

（2）ARM汇编器对大小写敏感，在汇编语言程序设计中，每一条指令的助记符可以全部用大写、或全部用小写，但不允许在一条指令中大、小写混用。

（3）所有的指令都不能在行的开头书写，必须在指令的前面有空格，然后再书写指令。

（4）指令助记符和后面的操作数或操作寄存器之间必须有空格，不可以在这之间使用逗号。

ARM汇编语言的语句格式

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

3. 操作数域(OPERAND)

操作数域表示操作的对象，操作数可以是常量、变量、标号、寄存器名或表达式，不同对象之间必须用逗号“,”分开。

本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
  - 7.3.1 ARM汇编语言的语句格式
  - 7.3.2 ARM汇编语言程序结构
  - 7.3.3 ARM汇编程序设计实例
- 7.4 ARM汇编语言与C/C++的混合编程

ARM汇编语言程序结构

中国科学技术大学
University of Science and Technology of China
信息科学技术学院

- 在ARM（Thumb）汇编语言程序中，通常以段为单位来组织代码段是具有特定名称且功能相对独立的指令或数据序列根据段的内容，分为代码段和数据段。

- 一个汇编程序至少应该有一个代码段，当程序较长时，可以分割为多个代码段和数据段。

ARM汇编语言程序结构

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

一个汇编语言程序段的基本结构如下所示：

```
AREA Buf, DATA, READWRITE;定义一个可读写属性的数据段
Num    DCD    0x11
Nums   DCD    0x22    ;分配一片连续字存储单元并初始化
;本节及下节所有汇编代码程序都是利用MDK IDE编写，Device设置为STM32F407ZG，代码段放在IROM中起始地址为0x08000000，数据段放在IRAM中起始地址为0x20000000。
AREA RESET, CODE, READONLY ;只读的代码段，RESET为段名
ENTRY                      ;程序入口点
```

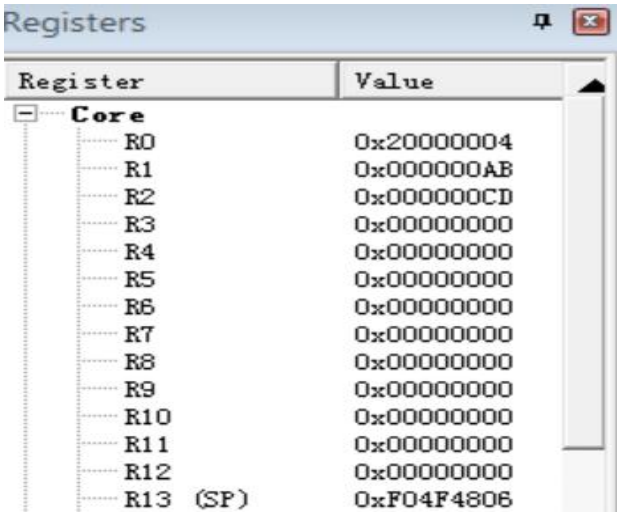
ARM汇编语言程序结构

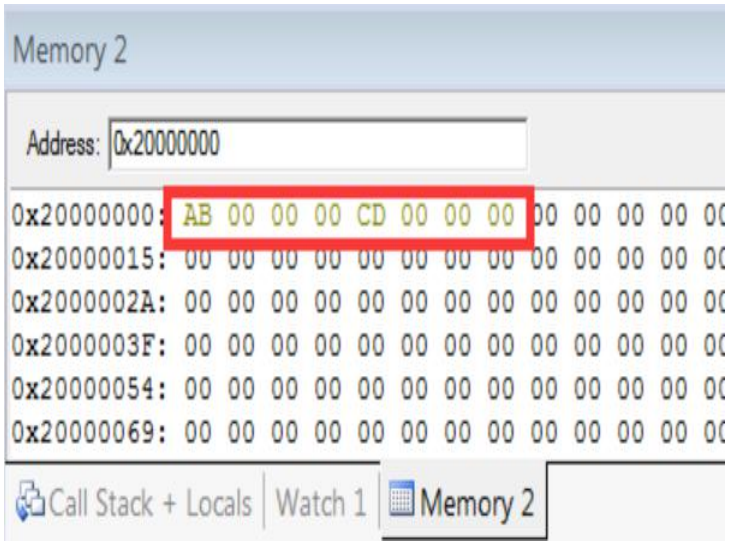
中国科学技术大学 University of Science and Technology of China 信息科学技术学院

```
START   LDR R0, = Num ;    取Num地址赋给R0
 LDR R1, [R0]  ;    取Num中内容赋给R1
 ADD R1, #0x9A ;    R1=R1+0x9A
 STR R1, [R0]  ;    R1内容赋给Num单元
 LDR R0, = Nums;    ......
 LDR R2, [R0]
 ADD R2, #0xAB
 STR R2, [R0]
LOOP   B   LOOP    ;       无限循环反复执行
END                ;       段结束
```

ARM汇编语言程序结构

中国科学技术大学 University of Science and Technology of China 信息科学技术学院





本例定义了两个段，先定义名为Buf的数据段，属性为可读写(数据段也可以不用单独的段结构而直接在代码段内定义)后又定义代码段，属性为只读。伪指令ENTRY标识了程序的入口，即该程序段被执行的第一条指令，接下来为程序主体。

本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
  - 7.3.1 ARM汇编语言的语句格式
  - 7.3.2 ARM汇编语言程序结构
  - 7.3.3 ARM汇编程序设计实例
- 7.4 ARM汇编语言与C/C++的混合编程

ARM汇编程序设计实例

中国科学技术大学
University of Science and Technology of China
信息科学技术学院

- 本节主要通过汇编程序设计实例重点介绍顺序结构、分支结构、循环结构等3大程序控制结构以及子程序调用等实例。

- 通过学习这些实例，掌握ARM汇编程序设计的基本方法，为设计更复杂的ARM程序奠定基础。

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

1.顺序结构

顺序结构是一种最简单的程序结构，这种程序按指令排列的先后顺序逐条执行。

[例]对数据段中数据进行寻址操作

```
AREA   BUF, DATA, READWRITE     ;定义数据段Buf
Array DCB   0x11,0x22,0x33,0x44
DCB 0x55,0x66,0x77,0x88
DCB 0x00,0x00,0x00,0x00    ;定义12个字节的数组Array
AREA   RESET, CODE, READONLY
ENTRY
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

```
LDR R0,=Array   ;取得数组Array的首地址
LDR R2,[R0]   ;从数组第1字节取32位数据给R2 即R2=0x44332211
MOV R1,#1     ; R1=1
LDR R3,[R0,R1,LSL#2]   ;将存储器地址为R0＋R1×4的32位数据读入寄存器R3,
;R3=0x88776655
LOOP      B     LOOP
END
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

2.分支结构

一般情况下，程序按指令的先后顺序逐条执行，但经常要求程序根据不同条件选择不同的处理方法，利用条件指令或条件转移指令根据当前CPSR中的状态标志值选择路径，使用带有条件码的指令实现的分支程序段。

条件码	助记符	含义	标志位取值
0000	EQ	等于（Equal）	Z == 1
0001	NE	不等于（Not equal）	Z == 0
0010	CS	产生了进位（Carry set）	C == 1
0011	CC	进位为零（Carry clear）	C == 0
0100	MI	负数（Minus, negative）	N == 1
0101	PL	整数或零（Plus, positive or zero）	N == 0
0110	VS	溢出（Overflow）	V == 1
0111	VC	没有溢出（No overflow）	V == 0
……		……	

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

2.分支结构

分支结构中经常使用的条件码助记后缀：HI 无符号数大于，LS 无符号数小于或等于；GT带符号数大于，LE 带符号数小于或等于。

例如寄存器中R0和R1分别保存两个数，如果R0小于R1，将R1值传给R0，实现代码如下：

```
CMP R0, R1;          比较R0和R1
MOVLT R0, R1;   MOVLT = MOV+ LT LT是有符号数比较小于
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

2.分支结构

分支结构的几种实现方法：

（1）利用条件码可以很方便地实现IF　ELSE分支结构的程序

ARM汇编语言实现分支结构代码片段：

```
MOV R0, #76 ;初始化R0的值
MOV R1, #88 ;初始化R1的值
CMP R0, R1 ;判断R0>R1 ?
MOVHI R2, #100 ;R0>R1时, R2=100
MOVLS R2, #50 ;R0<=R1时, R2=50
.....
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

2.分支结构

（2）B（Branch）条件转移及衍生指令实现分支结构

1）B指令

格式：B{条件} + 目标地址

用法：B指令是最简单的跳转指令，一旦遇到一个B指令，ARM处理器将立即跳转到给定的目标地址，从那里继续执行。

[例]

```
B   Label  ;程序无条件跳转到标号Label处执行
```

[例]

```
CMP R1,#0
BEQ Label  ;当CPSR寄存器中Z条件码置位时，程序跳转到标号Label处执行.....
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

2.分支结构

2）BL指令

格式：BL{条件} + 目标地址

用法：BL是一个跳转指令，但跳转之前，会在寄存器R14中保存PC当前的内容。因此，可以通过将R14的内容重新加载到PC中，来返回到跳转指令之后的那个指令处执行。这个指令是实现子函数调用的一个基本且常用的手段。

BL Label ;让程序无条件跳转到标号是Label处执行，同时将当前的PC值保存到R14（LR）中。

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

2.分支结构

BL实现子函数调用时完成如下的三个操作：

（i）将子程序的返回地址（当前PC）保存在R14（LR）中。

（ii）将PC指向子程序的入口即跳转（也即BL后面的目标指令）。

（iii）子程序执行完毕之后需要返回时，只需将R14中的LR赋给PC。

使用BL调用子程序后，通常在子程序的尾部添加MOV PC，LR来返回。

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

3.循环结构

循环结构可以减少源程序重复书写的工作量，用来描述重复执行某段算法的问题，这是程序设计中最能发挥计算机特长的程序结构。

（1）for循环结构实现

假设利用C语言实现如下循环结构：

```
for(i = 0; i < 10; i++)
x++;
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

3.循环结构

[例]已知32位有符号数X存放在数据段Arr单元中，要求实现：Y=1（X>0）或 Y=0（X=0）或Y=-1（X<0）。

```
AREA   RESET, CODE, READONLY
ENTRY
LDR    R1, =Arr
LDR    R2, [R1]
CMP    R2, #0
BEQ    ZERO
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

3.循环结构

```
   BGT PLUS
   MOV R0, #-1
   B   FINISH
PLUS  MOV R0, #1
   B   FINISH
ZERO  MOV R0, #0
FINISH STR R0, [R1]
END
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

3.循环结构

ARM汇编语言实现同样循环结构代码片段：R0为x，R2为i，均为无符号整数

```
MOV R0, #0 ; 初始化R0=0
MOV R2, #0 ; 初始化R2=0
LOOP CMP R2, #10 ; 判断R2<10?
BCS FOR_E ; 若条件失败（即R>=10），退出循环
ADD R0, R0, #1 ; 执行循环体，R0=R0+1，即x++ ，ADD R0, #1也可
ADD R2, R2, #1 ; R2=R2+1，即i++
B LOOP
FOR_E ......
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

3.循环结构

（2）While循环结构实现

假设利用C语言实现如下循环结构：

```
while(x<=y)
x*=2;
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

3.循环结构

ARM汇编代码片段实现：x为R0，y为R1，均为无符号整数

```
MOV R0,   #1  ；初始化R0=1
MOV R1,   #20  ；初始化R1=20

W1   CMP R0, R1  ；判断R0<=R1，即x<=y
MOVLS R0, R0, LSL#1  ；循环体，R0*=2
BLS W1  ；若R0<=R1，继续循环体
W_END ......
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

3.循环结构

[例]实现1+2+3+...+N的累加求和。

```
    AREA RESET, CODE, READONLY
    ENTRY
    MOV R0, #100  ; 循环数，即累加个数N
    MOV R1, #0        ;初始化数据
LOOP ADD R1, R1, R0  ; 将数据进行相加，获得最后的数据
    SUBS R0, R0, #1 ; 循环数据R0减去1
    CMP R0, #0        ; 将R0与0比较看循环是否结束
    BNE LOOP          ; 判断循环是否结束，接受则进行下面的步骤
    LDR R2, =RESULT    ; RESULT为数据段存储结果单元，将RESULT地址赋给R2
    STR R1, [R2]
    END
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

4.子程序调用与返回

在ARM汇编语言中，子程序的调用一般是通过BL指令来完成的。BL指令的语法格式如下：

BL　　SUB

SUB是被调用的子程序的名称。

BL指令完成2个操作，即将子程序的返回地址放在LR寄存器中，同时将PC寄存器指向子程序的入口点，当子程序执行完毕需要返回主程序时，只需将存放在LR中的返回地址重新赋给指令指针寄存器PC即可。

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

4.子程序调用与返回

BL调用子程序的经典用法如下：

```
    BL    NEXT   ;跳转到NEXT
     ……
NEXT
     ……
    MOV     PC，LR    ；从子程序返回。
```

当子程序需要使用的寄存器与主程序使用的寄存器发生冲突（即子程序与主程序都要使用同一组寄存器时），为了防止主程序这些寄存器中的有用数据丢失，在子程序的开始应该把这些寄存器数据压入堆栈以保护现场，在子程序返回之前需要把保护到堆栈的数据自堆栈中弹出以恢复现场。

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

4.子程序调用与返回

保存和恢复数据可以用PUSH/POP指令实现，PUSH {R4，LR}将寄存器R4入栈，LR也入栈。POP {R4，PC}将堆栈中的数据弹出到寄存器R4及PC中。

如果需要保存数据较多即需要入栈和出栈多个寄存器时，可以用PUSH {R0-R7，LR}

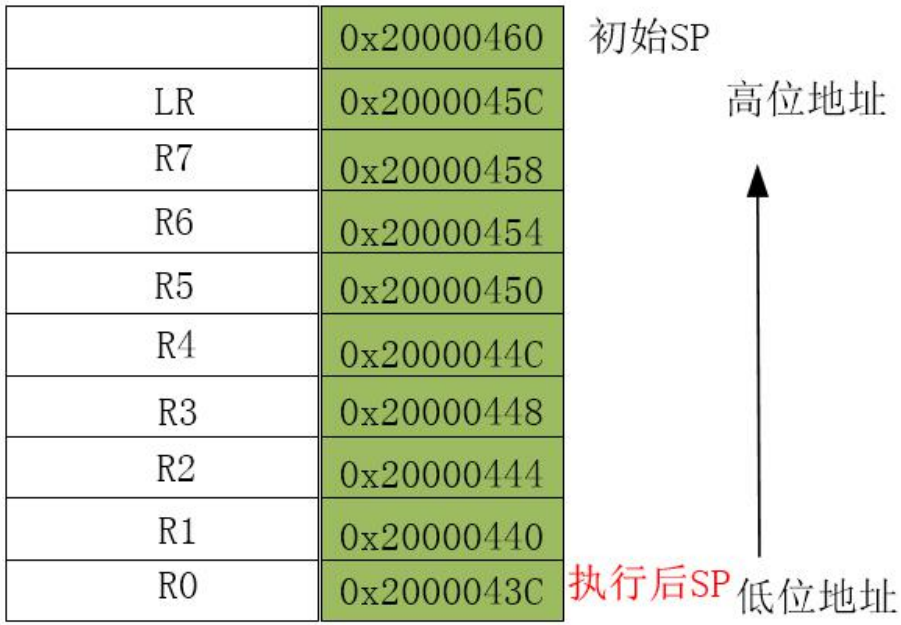
将寄存器R0-R7全部入栈，LR也入栈；POP {R0-R7，PC} 将堆栈中的数据弹出到寄存器R0-R7及PC中。

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

4.子程序调用与返回

ARM汇编指令中还有与PUSH/POP功能类似的压栈和出栈指令：STMFD SP!，{R0-R7，LR}；功能是满递减入栈，将寄存器R0-R7、LR压栈，SP不断减4，执行后SP=SP-9*4 [SP]=R0，[SP+4]=R1，.....，[SP+4*8]=LR。



ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

4.子程序调用与返回

LDMFD SP!, {R0-R7, PC}; 满递减出栈，给寄存器R0-R9出栈，并使程序跳转回函数的调用点，SP不断增4；同理，LDMFD是STMFD的逆操作，[SP]->R0，[SP+4]->R1，...，[SP+4*8]->PC SP=SP+4*9

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

4.子程序调用与返回

另外有一点需要注意：BL　SUB ... MOV PC，LR这种调用子程序结构在一些情况下是会出现问题的，例如当出现子程序嵌套调用时，LR寄存器中内容在第二次子程序调用时会被覆盖，如果仍使用常用调用方式会出现无法返回现场的问题，这个时候使用STMFD/LDMFD指令可以有效避免LR被覆盖而出现错误。

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

4.子程序调用与返回

子程序嵌套调用时，STMFD/LDMFD指令的用法

```
AREA   RESET, CODE, READONLY
ENTRY
START      LDR    SP, =0x20000460
           MOV    R0, #0x03
           MOV    R1, #0x04
MOV    R7, #0x07
           BL   POW
LOOP       B    LOOP
```

ARM汇编程序设计实例

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

```
POW       STMFD   SP!, {R0-R7, LR}
          MOVS    R2, R1
          MOVEQ   R0, #1
          BEQ     POW_END
          MOV     R1, R0
          SUB     R2, R2, #1
POW_L1    BL      DO_MUL      ; 子程序调用嵌套，这个时候如果不用
STMFD/LDMFD指令会出现LR覆盖，影响子程序调用的正常返回
          SUBS    R2, R2, #1
          BNE     POW_L1
POW_END         LDMFD   SP!, {R0-R7, PC}
DO_MUL          MUL     R0, R1, R0
          MOV     PC, LR
END
```

本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
- 7.4 ARM汇编语言与C/C++的混合编程
  - 7.4.1 C语言与汇编语言之间的函数调用
  - 7.4.2 C/C++语言与汇编语言的混合编程
  - 7.4.3 C编程中访问特殊寄存器的指令

ARM汇编语言与C/C++的混合编程

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 嵌入式软件开发过程中，通常会使用包括ARM汇编语言和C/C++语言在内的多种语言。一般情况下，一个ARM工程(Project)应该由多个文件组成，其中有可能包括扩展名为.s的汇编语言源文件、扩展名为.c的C语言源文件、扩展名为cpp的C++源文件，以及扩展名为.h的头文件等。
- 通常程序会使用汇编完成处理器启动阶段的初始化等工作，有些对处理器运行效率较高的底层算法也会采用汇编语言进行编写和手工优化，而在开发主程序时一般会采用C/C++语言，因此嵌入式软件开发人员必须掌握ARM汇编语言与C/C++语言混合编程技能。

本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
- 7.4 ARM汇编语言与C/C++的混合编程
  - 7.4.1 C语言与汇编语言之间的函数调用
  - 7.4.2 C/C++语言与汇编语言的混合编程
  - 7.4.3 C编程中访问特殊寄存器的指令

C语言与汇编语言之间的函数调用

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

1. ATPCS

- 为了使单独编译的C语言程序和汇编程序之间能够相互调用，必须为子程序之间的调用制定一定的规则。 ATPCS就是ARM程序和Thumb程序中子程序调用的基本规则。
- ATPCS（ARM-Thumb Procedure Call Standard，基于ARM指令集和Thumb指令集过程调用规则）规定了一些不同语言撰写的函数之间相互调用（mix calls）的基本规则，这些基本规则包括子程序调用过程中寄存器的使用规则、数据栈的使用规则、以及参数的传递规则。

C语言与汇编语言之间的函数调用

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

ATPCS规定了在子程序调用时的一些基本规则，主要包括以下3方面的内容：

- 各寄存器的使用规则及其相应的名字；
- 数据栈的使用规则（FD）；
- 参数传递的规则；

参数传递规则：

当参数个数不超过4个时，可以使用寄存器R0～R3来传递参数，如果参数多于4个，则将剩余的字数据通过堆栈传递，入栈的顺序与参数传递顺序相反，即最后一个字数据先入栈，第一个字数据最后入栈。

C语言与汇编语言之间的函数调用

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

2. C程序调用汇编函数实例

- ARM编译器使用的函数调用规则就是 ATPCS标准，也是设计可被C程序调用的汇编函数的编写规则。为了保证程序调用时参数传递正确，C程序调用的汇编函数时必须严格按照ATPCS规则。

- 如果汇编函数和调用函数的C程序不在同一个文件中，则需要在汇编语言中用EXPORT声明汇编语言起始处的标号为外部可引用符号，该标号应该为C语言中所调用函数的名称。

C语言与汇编语言之间的函数调用

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

[例]调用汇编函数，实现把字符串srcstr复制到字符串dststr中。

```
//main.c
extern void strcopy (char * d , char * s ) ; //需要调用的汇编函数原型并加extern关键字
int main ()
{
    char * srcstr = "0123456" ;
    char  dststr [ ] = "abcdefg" ;
    strcopy(dststr,srcstr);
    return 0 ;
}
```

C语言与汇编语言之间的函数调用

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

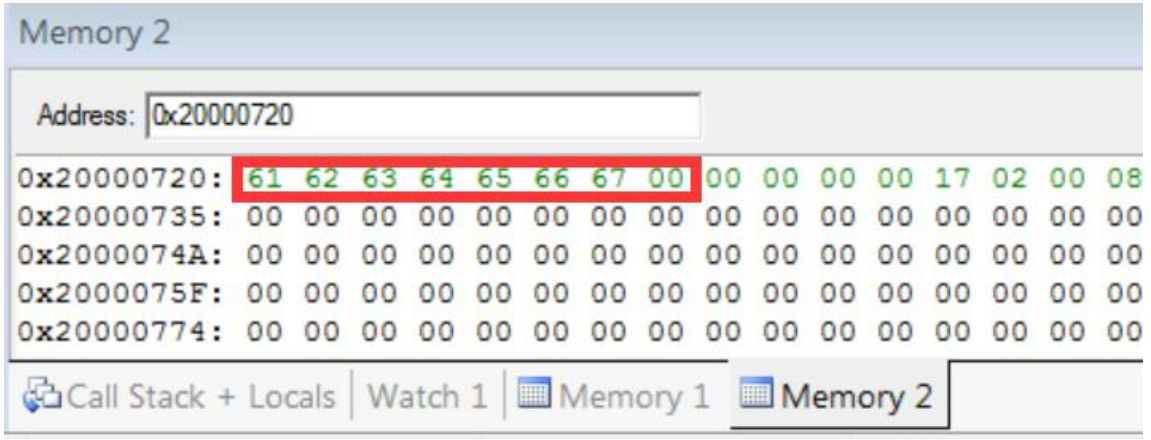
汇编语言源程序Scopy.s，汇编文件和*.c文件在同一工程中

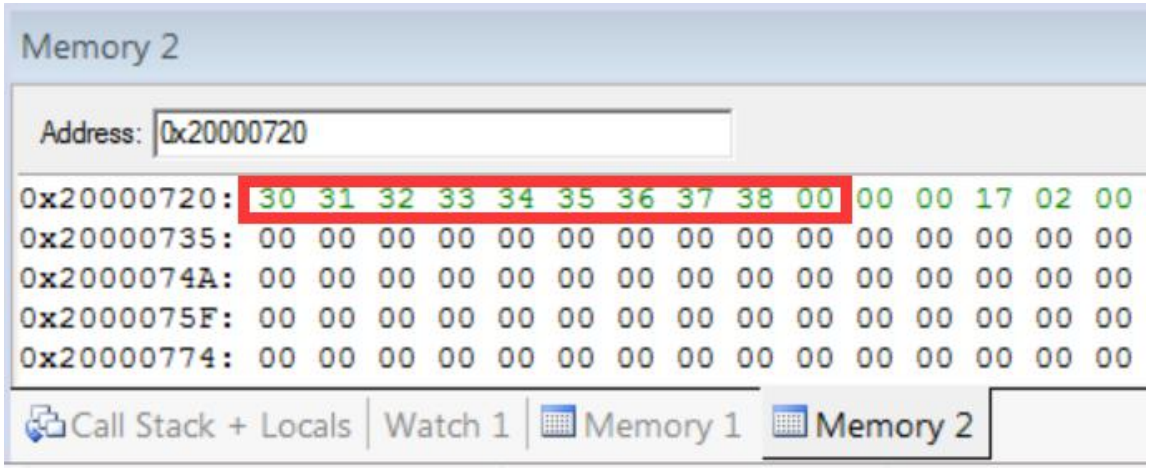
```
AREA Scopy,CODE,READONLY
 EXPORT strcopy
strcopy   ;必须与EXPORT后面标号一致
LOOP    LDRB R2, [R1],#1 ;R1指向源地址
        STRB R2, [R0],#1 ;R0指向目标地址
        CMP  R2, #0
         BNE  LOOP ;先执行后判断，源字符串的终止符‘\0’
                   ;也复制到目的字符串
        MOV  PC, LR
END
```

C语言与汇编语言之间的函数调用

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

StrCopy执行前后目的字符串中内容变化





C语言与汇编语言之间的函数调用

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

3.汇编程序调用C函数实例

- 在汇编程序中调用C语言函数，需要在汇编程序中利用IMPORT说明对应的C函数名，按照ATPCS的规则保存参数。完成各项准备工作后利用跳转指令跳转到C函数入口处开始执行。跳转指令后所跟标号为C函数的函数名。

C语言与汇编语言之间的函数调用

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

3.汇编程序调用C函数实例

[例]汇编程序中调用C函数实现求5个整数相加的和

;test.s，工程设置为基于汇编的工程，不需要从main函数启动，代码段名称必须设为RESET

```
PRESERVE8
AREA RESET,CODE,READONLY
ENTRY
IMPORT CAL ;
LDR    SP, =0x20000460 ;设置堆栈指针
MOV  R0, #1; R0=1
ADD  R1, R0, R0; R1=2
```

C语言与汇编语言之间的函数调用

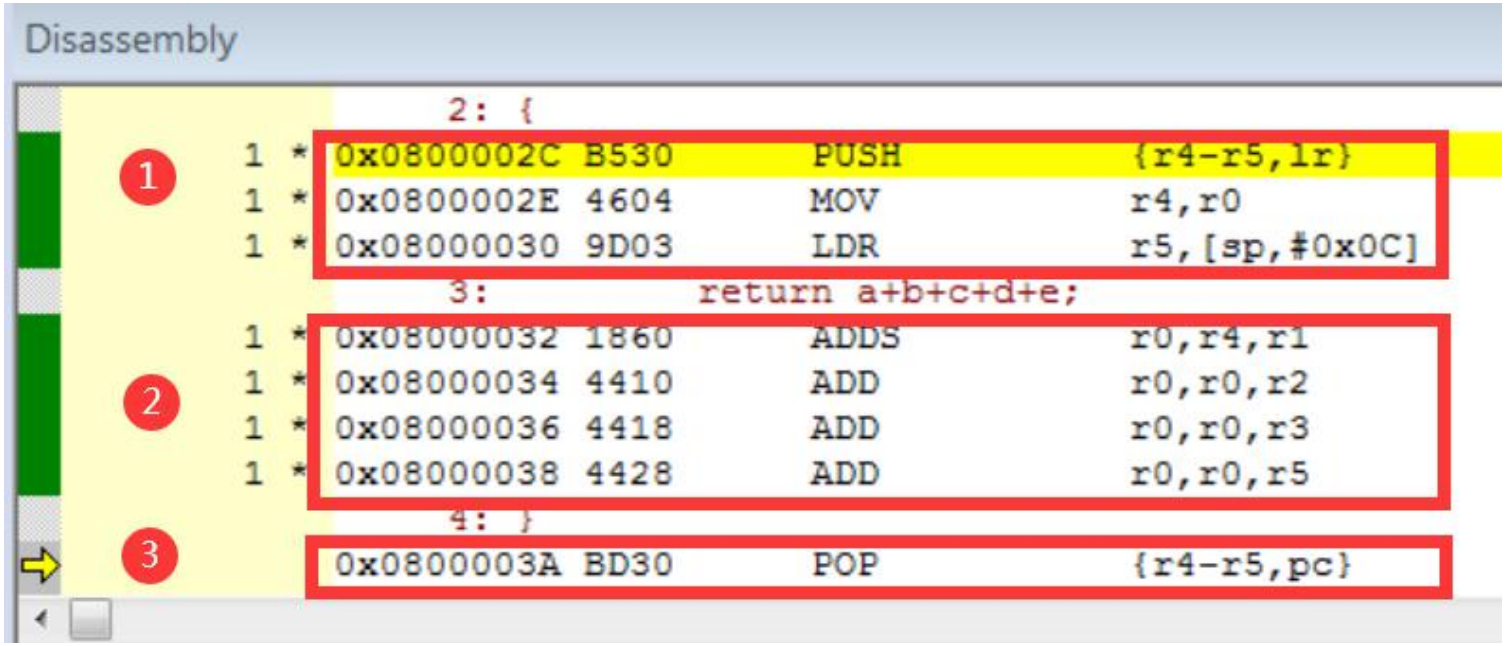
中国科学技术大学 University of Science and Technology of China 信息科学技术学院

```
ADD   R2, R1, R0; R2=3
ADD   R3, R0, R2; R3=4
ADD   R4, R0, R3; R4=5
STR   R4, [SP, #-4]!
BL    CAL
LDR R4, =0x20000000 ;结果R0存入内存单元
STR R0, [R4]
END
```

C语言与汇编语言之间的函数调用

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

```
//C源程序example.c，和汇编文件在同一工程中
int CAL(int a, int b, int c, int d, int e)
{
    return (a+b+c+d+e);//子程序反汇编后如图
}
```



本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
- 7.4 ARM汇编语言与C/C++的混合编程
  - 7.4.1 C语言与汇编语言之间的函数调用
  - 7.4.2 C/C++语言与汇编语言的混合编程
  - 7.4.3 C编程中访问特殊寄存器的指令

C/C++语言与汇编语言的混合编程

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 嵌入式系统开发中，目前使用的主要编程语言是C和汇编，C++已经有相应的编译器，但是现在使用还是比较少的。在稍大规模的嵌入式软件中，大部分的代码都是用C编写的。

- C语言中使用内嵌汇编代码，可以在C/C++程序中实现C/C++不能完成的一些操作，同时程序的代码效率也会更高。

C/C++语言与汇编语言的混合编程

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

1.在C语言程序中嵌入汇编指令

如果要在C程式中嵌入汇编可以有两种方法：内联汇编和内嵌汇编。嵌入汇编使用的标记是 __asm或者asm关键字，用法如下：

```
__asm
{
instruction [; instruction]
…
[instruction]
}
asm(“instruction [; instruction]”);
```

C/C++语言与汇编语言的混合编程

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

1.在C语言程序中嵌入汇编指令

（1）内联汇编的示例代码如下:

```
int Add(int  i)
{
   int R0;
   __asm
  {
     ADD  R0,i,1
     EOR  i,  R0,i
  }
   return i;
}
```

C/C++语言与汇编语言的混合编程

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

1.在C语言程序中嵌入汇编指令

（2）内嵌汇编示例代码如下:

```
#include <stdio.h>
__asm void my_strcpy(const char *src, char *dst)
{
  LOOP LDRB R2, [R0], #1 ;R0保存第一个参数
  STRB R2, [R1], #1 ;R1保存第二个参数
  CMP  R2, #0
  BNE  LOOP
  BLX   LR ; 返回指令须要手动加入
}
```

C/C++语言与汇编语言的混合编程

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

1.在C语言程序中嵌入汇编指令

```
int main(void)//主程序
{
        const char *a = "Hello world!";
        char b[20];
        my_strcpy (a, b);//调用内嵌汇编程序
        return 0;
}
```

由两种方法的示例可以看出：内联汇编可以直接嵌入C代码使用，而内嵌汇编更像是一个函数。由于内联式汇编只能在ARM状态中进行，而Cortex-M3/M4只支持Thumb-2，所以Cortex-M3/M4只能使用内嵌汇编的方式，也就是第二种方式。

C/C++语言与汇编语言的混合编程

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

1.在C语言程序中嵌入汇编指令

在C语言中内嵌汇编指令与汇编程序中的指令有些不同，存在一些限制，主要有下面几个方面：

（1）不能直接向PC寄存器赋值，程序跳转要使用B或者BL指令。

（2）在使用物理寄存器时，不要使用过于复杂的C表达式，避免物理寄存器冲突。

（3）R12和R13可能被编译器用来存放中间编译结果，计算表达式值时可能将R0到R3、R12及R14用于子程序调用，因此要避免直接使用这些物理寄存器。

（4）一般不要直接指定物理寄存器，而让编译器进行分配。

C/C++语言与汇编语言的混合编程

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

【例】在C程序中使用内联汇编代码实现字符串复制。

```
#include <stdio.h>
void my_strcpy(const char *src,  char *dest)
{
char ch;
__asm/ /注意是双下划线
  {
    LOOP:
    LDRB ch, [src], #1
    STRB ch, [dest], #1
    CMP ch, #0
    BNE LOOP
  }
}
```

C/C++语言与汇编语言的混合编程

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

```
int main()
{
   char *a = "ok!";
   char b[64];
   my_strcpy(a, b);
   return 0;
}
```

C/C++语言与汇编语言的混合编程

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

2.在汇编中调用C语言定义的函数和全局变量

- 使用内联或者内嵌汇编不用单独编辑汇编语言文件使用起来比较方便，但是有诸多限制。
- 当汇编文件较多的时候就需要使用专门的汇编文件编写汇编程序，在C语言和汇编语言进行数据传递的最简单的形式是使用全局变量。

本章内容

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

- 7.1 ARM程序开发环境
- 7.2 ARM汇编程序中的伪指令
- 7.3 ARM汇编语言程序设计
- 7.4 ARM汇编语言与C/C++的混合编程
  - 7.4.1 C语言与汇编语言之间的函数调用
  - 7.4.2 C/C++语言与汇编语言的混合编程
  - 7.4.3 C编程中访问特殊寄存器的指令（略）

习题

中国科学技术大学 University of Science and Technology of China 信息科学技术学院

① 7.3
② 7.4
③ 7.5
④ 7.6
⑤ 7.7
⑥ 7.8
⑦ 7.9
⑧ 7.10
⑨ 7.11
⑩ 7.12