微型原理与接口技术

Hardware Principles and Interfacing of Modern Computer

Lecture 6: Assembly Programming

陈启军，张 伟

Email: zhang_wi@mail.tongji.edu.cn

Dept. Of Automation, TongJi University

Contents

- 基本概念
  - 汇编程序,汇编器(assembler)和链接器(linker)
- 汇编开发的基本流程
- 汇编程序的语法
  - 基本的汇编程序格式
  - 伪操作
  - 结构化编程：循环与分支
  - 宏与子程序(macro & procedure)
  - 模块化编码(exten & public)
  - 库(library)
  - 数据结构与汇编语言
- 混合语言编程(Mix C/C++ and Assembly Language)
  - 嵌入式汇编(inline assembler)
  - 独立汇编模块

Objective

- 理解汇编程序开发的流程和步骤
- 熟悉宏汇编语言的常见语法，掌握其基本开发
  - 熟练掌握最基本汇编程序的开发（模板程序）
  - 在查阅手册的基础上，可以承担更复杂程序的开发
- 掌握C和汇编混合开发的技能
- 锻炼通过模块化技术把握大程序开发的能力

Reference

- 课本：内容比较分散
  - Section 4.7 Assembler Detail
  - Section 7.1
  - Chapter 8
  - Appendix A
- 沈美明，温冬婵编，IBM PC汇编语言程序设计，清华大学出版社，1994
  - Chapter 4, 5, 6, 7, 13

    注：新版本中章节号很可能有所变化，请自行决策

Motivation of ASM Developing

- 问题：为什么我们有了高级语言，还要学习汇编语言的使用和开发?
  - 硬件操纵采用汇编语言更方便，效率也更高
    - 例如很多低成本的简单单片机只支持汇编
    - 很多计算机固件也常用汇编语言开发
  - 提高系统的性能
    - 我们经常会编写一些汇编模块，代替相同功能的高级语言模块，以提升系统的性能，例如在基于DSP芯片的高速信号处理和图像处理领域中

Assembler & Linker

- 基本概念：
  - 汇编器(assembler)：
    - The assembler program converts a symbolic source file into a binary (hexadecimal) object file

      Symbolic Instructions => Binary Instructions

      (man readable) (machine readable)
  - 链接器(linker)
    - Reads the object files outputted by assembler and links them together into a single execution file. Third party binary libraries are often linked together too.

      Binary Object Files (+ Binary Libraries) = Execution File

Assembler & Linker

- 思考：为什么高级语言(如C)和低级语言(汇编)可以实现混合编程?
  - 高级语言源程序可以经过“编译”(compile)生成链接器所需要的二进制模块文件，这些模块文件和assembler生成的二进制模块文件具有相同的格式，遵循共同的标准(例如Intel的OMF标准和TI/Motorola的COFF标准?），因此它们都可以在最后环节被link程序使用，从而实现混合编程（mix-programming）。

Assembly Developing Flow

- **Step 1: 编辑(Edit)**
  - 汇编语言源程序通常保存为文本文件格式，可以用任何一个文本编辑器打开并进行修改；
- **Step 2: 汇编(Assemble)**
  - 调用assembler对源文件进行汇编，生成二进制中间代码(OBJ格式)
- **Step 3: 链接(Link)**
  - 调用linker对assembler输出的OBJ文件进行链接，生成最终的可执行文件；
- **Step 4: 调试(Debug)**
  - 启动调试程序(debugger)进行调试。如发现错误则返回step 1进行修改

Assembly Developing Flow

- 思考：如何简化上述流程并且对众多的源文件以及编译链接过程进行管理?
  - Make 程序的诞生，它可以自动识别哪些源文件更改过然后再进行自动化操作
  - 操作系统提供的批处理功能
    - DOS下的批处理文件(.bat)和Linux下面的脚本文件(script)

Programming: Basic Framework

- 思考：静态的程序和动态的进程之间的关系
- 思考：如何通过静态的程序（指令和伪指令的序列）描述内存中动态的进程？它们是如何对应在一起的？
- 一个例子：

```
title    test
;*************************************************************************************
data_seg segment                                    ; define data segment
         oper1      dw          12
         oper2      dw          230
         result     dw          ?
data_seg segment
;*************************************************************************************
data_seg2 segment                                   ; define extra segment
         ...
data_seg2 segment
;*************************************************************************************
code_seg segment                                    ; define code segment
         assume cs: code_seg, ds: data_seg, es: data_seg2
start:                                              ; starting execution address
         mov ax, data_seg                           ; data segment address
         mov ds, ax                                 ; into DS register
; main part of program goes here
         mov ax, oper1
         add ax, oper2
         jge store
         neg ax      ...
store: mov result, ax
         hlt
code_seg ends                                       ; end code segment
;*************************************************************************************
         end start                          ; end assembly
```

Programming: Assume, ...

- 概念：伪操作
  - 由汇编器(assemble)提供，帮助汇编程序的书写和开发，一般都不产生实际执行的二进制代码
  - 数据定义和存储器分配 db, dw, dd, dq, dup
  - 段定义类：segment, assume, common
  - 程序开始结束：name, title, end
  - 定位伪操作：org
  - ……

Programming: Assume, ...

- 注：
  - 堆栈的初始化(initializing the stack): TextBook Sec 4.2 讲Push/Pop指令的提到
  - 伪操作和汇编指示符的说明见TextBook Sec 4.7 Assemble Detail Table 4.21
  - 内存模式TINY, SMALL的说明见TextBook Sec 4.7 Memory Organization小节

Programming: Flow Control

- 程序的四种标准结构形式：顺序，循环，分支，子程序
  - 可以证明，通过这四种结构，可以满足一切要求，因此，人们在高级语言中设计了for/while, if等语句。这就是结构化编程的基本思想。
- 回顾：在结构化编程出现之前的汇编时代
  - Goto语句 ＋ 条件判断 + 标号 也可以实现上述四种标准结构，而且更灵活，但灵活过了度
  - Goto语句在ASM中体现为JMP/LOOP指令族，条件判断则主要通过CPU FLAG寄存器中的大量标志位实现，必要时可以结合计数器技巧(常用CX寄存器实现)

参考：Text Book Section 6.1, 6.2

Programming: Flow Control

- 思考：结构化编程和Goto编程的思想差异在何处?
  - 结构化编程在不影响语言表述能力的前提下，对编码风格和语法引入了更多的约束，更加强调系统的模块化构建，比较明确的区分了模块的接口(interface)和实现(implementation)部分
  - 基于goto的编程未能体现“接口和实现”相区分的思想，因此导致很多大型程序成为混乱的“通心面”结构，导致后续差错和维护极为困难。

Tips: 基于goto的编程是可以实现模块化编程的，但这并不是语法上的约束，而是要完全依赖程序员本身的设计。我们应在以后掌握基于goto的结构化和模块化编程技术，以更好的控制大型程序的开发。

Programming: Macro

- 概念：什么是宏(macro)? (section 7.1)
  - A macro is a group of instructions that perform one task
- Assembler是如何实现宏功能的?
  - 通过预处理器(pre-processing)
- Macro的定义和使用
  - 定义：macro name MACRO [dummy parameter list]

    ……..

    ENDM
  - 使用：可以把macro看作是一条汇编指令，从而在汇编代码中直接使用

Programming: Macro

- **Macro开发技巧**
  - 带有变元(参数)的宏
    - Macro中的变元几乎可以替代任何字符串，而不仅仅用来表达参数，因此灵活多了度，实际中易导致错误
  - 带有局部变量的宏：LOCAL伪操作修饰符
  - 宏的嵌套使用

Programming: Preprocessing

- 基于预处理技术的其它汇编特性
  - 重复伪操作
    - REPT expression
      ....
    - ENDM
  - 条件汇编
    - IF argument
    - …
    - [ELSE]
    - …
    - ENDIF

要点：预处理器用于处理原始代码文件，产生真正用于汇编的指令代码文件。所以，预处理器有效的时候指令还没有得到执行，预处理中使用到的表达式或者变量都必须预先定义，或者是可以由预处理器算出，否则就会在预处理阶段出错。

Programming: Procedure

- 概念：过程(procedure)
  - A procedure is a group of instructions that usually performs one task. (Section 6.3, P. 203)
- 过程的定义和使用
  - 定义：
```
name PROC <modifier>
        .....
        RET
name ENDP
```
  - 调用：CALL指令

Programming: Procedure

- 思考：函数的参数是如何传递的(如何输入和输出）？
  - 由此体会堆栈的作用
  - 由此理解为什么调用函数要用CALL指令，函数中要用RET指令返回。也由此理解macro与procedure在实现上的差别

Programming: Procedure

- 一类特殊的函数：中断函数ISR (interrupt service routines)
  - 特点：往往是由硬件自动调用
  - 通过INT指令进行调用，通过IRET指令返回
- 思考：硬件如何知道ISR放在何处?
  - Interrupt Table和Interrupt Vector的由来
    - Interrupt number (ref: section 6.4, P. 208)
- 思考：为什么ISR调用和返回不能借助普通子程序的CALL和RET?而要使用INT和IRET?
  - 程序状态的保存和恢复问题，例如PSW

Programming: Procedure

- 思考：如何控制中断程序执行的时机?
  - 体会：中断的整个流程
  - 中断标志位
  - 开中断指令STI和关中断指令CLI

Programming: Module

- 思想：模块化编程
  - 与模块化编程相联系的是接口与实现的思想（封装的思想），每个模块都对外提供明确的接口，其它模块通过该接口访问访问模块的功能
- 汇编语言对模块化编程的语法支持
  - extrn & public
  - extrn和public机制的实现：借助链接器(linker)
- 宏库及其使用： macro library
- 子程序库及其使用： procedure library
  - 子程序库的使用要借助链接器

Programming: Data Structure

- 汇编语言仅仅提供了最最基本的数据类型
  - 不提供任何复杂的数据类型和数据结构，如数组、链表、树、队列、图等
- 汇编程序本身必须负责数据结构的定义和使用
  - 程序员必须精心考虑数据在内存中的存放格式

Mixture Programming

- 嵌入式汇编(inline assembler)
  - 注意当前程序运行的环境：16bit还是32bit，real mode还是protected mode
  - 语法：_asm{ …}
    - 由高级语言的编译器(compiler)提供
  - 难点：如何在汇编代码和宿主程序中传递数据
- 独立式汇编模块(Independent Assembly Module)
  - 难点：如何在汇编程序和其它高级语言模块之间传递数据？必须遵从高级语言编译器的格式约定。

Ref: chapter 8

Summary

- 基本概念：assembler和linker
- 汇编程序开发的流程，理解每个步骤的用意
- 最基本的汇编程序开发模板、模式和套路
- 掌握模块化开发技术
  - 合理搭配宏、子程序、宏库、子程序库，锻炼驾驭大规模系统开发的能力
- 汇编与C混合编程的技能
  - 一般用C开发主程序，用汇编对某些关键函数进行性能优化