<<嵌入式软件基础>>

图书基本信息

书名：<<嵌入式软件基础>>

13位ISBN编号：9787040140590

10位ISBN编号：7040140594

出版时间：2004-8

出版时间：北京蓝色畅想图书发行有限公司（原高等教育出版社）

作者：刘易斯 编

页数：266

字数：340000

<<嵌入式软件基础>>

内容概要

本书是对高等院校本科二年级计算机组成原理与汇编语言程序设计的传统教材的全新替代版。
本书以实践中最常运用的方式讲解汇编语言——实现小型、快速或特殊目的的例程，这些例程由主程序（高级语言编写，如C）调用。
通过运用嵌入式软件环境，本书介绍多线程程序设计、可抢占式系统与非可抢占式系统、共享资源和调度，从而为操作系统、实时系统、计算机网络及基于多处理器的设计等后续课程提供了坚实的基础。
本书将帮助读者：理解通常为人们所忽视的二进制表示的后果和局限性问题；运用定点（而非浮点）实数实现快速实数运算；加强对于作用域、参数传递、递归和内存分配的理解；运用C语言的特性（如位操作和变量访问），这些特性在嵌入式软件中广泛应用；编写Intel x86保护模式下的汇编函数，由C程序调用；估算不同类型输入/输出程序设计的最大数据速率和等待时间；管理多线程、共享资源和临界区；开发程序设计实例，以避免优先级倒置、死锁和共享内存问题。
本书适用于高等院校工科各专业本科嵌入式计算机系统程序设计、C语言程序设计及汇编语言程序设计类课程，也可供相关技术人员学习参考。

<<嵌入式软件基础>>

书籍目录

PrefaceChapter 1 Introduction 1.1 What is an Embedded System? 1.2 What's Unique About the Design Goals for Embedded Software? 1.3 What Does "Real-Time" Mean? 1.4 What Does "Multitasking" Mean? 1.5 How Powerful Are Embedded Processors? 1.6 What Programming Languages Are Used? 1.7 What Is a "Real-Time Kernel"? 1.8 How Is Building an Embedded Application Unique? 1.9 How Big Are Typical Embedded Programs? 1.10 The Software Used in This Book ProblemsChapter 2 Data Representation 2.1 Fixed-Precision Binary Numbers 2.1.1 Positional Number Systems 2.1.2 Binary-to-Decimal Conversion 2.1.3 Decimal-to-Binary Conversion 2.1.4 Counting 2.1.5 Fixed Precision and Rollover 2.1.6 Hexadecimal Representation 2.2 Binary Representation of Integers 2.2.1 Signed Integers 2.2.2 Positive and Negative Representations of the Same Magnitude 2.2.3 Interpreting the Value of a 2's-Complement Number 2.2.4 More on Range and Overflow 2.2.5 2's Complement and Hardware Complexity 2.3 Binary Representation of Real Numbers 2.3.1 Fixed-Point Representation 2.3.2 Fixed-Point Using a Universal 16.16 Format 2.3.3 Fixed-Point Using a Universal 32.32 Format 2.3.4 Floating-Point Representation 2.4 ASCII Representation of Text 2.5 Binary-Coded Decimal (BCD) ProblemsChapter 3 Getting the Most Out of C 3.1 Integer Data Types 3.2 Mixing Data Types 3.3 Useful Typedefs and Defines 3.4 Manipulating Bits in Memory 3.4.1 Testing Bits 3.4.2 Setting, Clearing, and Inverting Bits 3.4.3 Extracting Bits 3.4.4 Inserting Bits 3.5 Manipulating Bits in I/O Ports 3.5.1 Write-Only I/O Ports 3.5.2 Ports Differentiated by Reads Versus Writes 3.5.3 Ports Differentiated by Sequential Access 3.5.4 Ports Differentiated by Bits in the Written Data 3.6 Accessing Memory-Mapped I/O Devices 3.6.1 Accessing Data Through a Pointer 3.6.2 Arrays, Pointers, and the "Address of" Operator 3.7 Structures 3.7.1 Packed Structures 3.7.2 Bit Fields 3.8 Variant Access 3.8.1 Casting the Address of an Object 3.8.2 Using Unions ProblemsChapter 4 A Programmer's View of Computer Organization 4.1 Memory 4.2 The Central Processing Unit (CPU) 4.2.1 The Arithmetic and Logic Unit (ALU) 4.2.2 Other Registers 4.2.3 The Control Unit 4.3 Input/Output (I/O) 4.4 Introduction to the Intel Architecture 4.4.1 Instruction Formats 4.4.2 Instruction Operands 4.4.3 Operand Restrictions 4.4.4 Registers 4.4.5 The Stack 4.5 The Intel Real Mode Architecture 4.5.1 Segmented Addressing 4.5.2 Addressing Modes 4.6 The Intel Protected Mode Architecture 4.6.1 Segment Registers and The Global Descriptor Table 4.6.2 The Flat Memory Model 4.6.3 Addressing Modes 4.7 Operand and Address-Size Override Prefixes 4.8 The Intel Data Manipulation Instructions 4.8.1 Data Movement, Stack, and I/O Instructions 4.8.2 Arithmetic Instructions 4.8.3 Bitwise Instructions 4.8.4 Shift Instructions ProblemsChapter 5 Mixing C and Assembly 5.1 Programming in Assembly 5.2 Register Usage Conventions 5.3 Typical Use of Addressing Options 5.3.1 Accessing Data Whose Address is a Constant 5.3.2 Accessing Data Whose Address is a Variable 5.4 Instruction Sequencing 5.4.1 Compound Conditionals 5.4.2 If-Then-Else Statements 5.4.3 Building Loops 5.4.4 Faster Loops with String Instructions 5.5 Procedure Call and Return 5.6 Parameter Passing 5.7 Retrieving Parameters 5.8 Everything is Pass by Value 5.9 Temporary Variables ProblemsChapter 6 Input/Output Programming 6.1 The Intel I/O Instructions 6.2 Synchronization, Transfer Rate, and Latency 6.3 Polled Waiting Loops 6.4 Interrupt-Driven I/O 6.4.1 The Hardware Response 6.4.2 The Interrupt Service Routine 6.4.3 Programmable Interrupt Controllers 6.4.4 Buffers and Queues 6.4.5 Writing Interrupt Service Routines in Assembly 6.4.6 Writing Interrupt Service Routines in C 6.4.7 Nonmaskable Interrupts 6.4.8 Software Interrupts 6.4.9 Exceptions 6.5 Direct Memory Access 6.5.1 Double Buffering 6.6 Comparison of Methods ProblemsChapter 7 Concurrent Software 7.1 Foreground/Background Systems 7.1.1 Thread State and Serialization 7.1.2 Managing Latency 7.1.3 Preventing Interrupt Overrun 7.1.4 Moving Work into the Background 7.2 Multithreaded Programming 7.2.1 Concurrent Execution of Independent Threads 7.2.2 Context Switching 7.2.3 Nonpreemptive (Cooperative) Multitasking 7.2.4 Preemptive Multitasking 7.3 Shared Resources and Critical Sections 7.3.1 Disabling Interrupts 7.3.2 Disabling Task Switching 7.3.3 Spin Locks 7.3.4 Mutex Objects 7.3.5 Semaphores ProblemsChapter 8 Scheduling 8.1 Thread States 8.2 Pending Threads 8.3 Context Switching 8.4 Round-Robin Scheduling 8.5 Priority-Based Scheduling 8.5.1 Priority Inversion 8.5.2 The Priority

Inheritance Protocol 8.5.3 The Priority Ceiling Protocol 8.6 Assigning Priorities 8.6.1 Deadline-Driven Scheduling 8.6.2 Rate-Monotonic Scheduling 8.7 Deadlock 8.8 Watchdog Timers ProblemsChapter 9 Memory Management 9.1 Objects in C 9.2 Scope 9.2.1 Refining Local Scope 9.2.2 Refining Global Scope 9.3 Lifetime 9.4 Automatic Allocation 9.4.1 Storage Class "Register" 9.5 Static Allocation 9.6 Three Programs to Distinguish Static from Automatic 9.6.1 Object Creation 9.6.2 Object Initialization 9.6.3 Object Destruction 9.7 Dynamic Allocation 9.7.1 Fragmentation 9.7.2 Memory Allocation Pools 9.8 Automatic Allocation with Variable Size (alloca) 9.8.1 Variable-Size Arrays 9.9 Recursive Functions and Memory Allocation ProblemsChapter 10 Shared Memory 10.1 Recognizing Shared Objects 10.1.1 Shared Global Data 10.1.2 Shared Private Data 10.1.3 Shared Functions 10.2 Reentrant Functions 10.3 Read-Only Data 10.3.1 Type Qualifier "const" 10.4 Coding Practices to Avoid 10.4.1 Functions That Keep Internal State in Local Static Objects 10.4.2 Functions That Return the Address of a Local Static Object 10.5 Accessing Shared Memory 10.5.1 The Effect of Processor Word Size 10.5.2 Read-Only and Write-Only Access 10.5.3 Type Qualifier "volatile" ProblemsChapter 11 System Initialization 11.1 Memory Layout 11.2 The CPU 11.2.1 Setting Up a Flat Memory Model 11.2.2 Switching into Protected Mode 11.3 C Run-Time Environment 11.3.1 Copying from ROM to RAM 11.3.2 Zeroing Uninitialized Statics 11.3.3 Setting Up a Heap 11.4 System Timer 11.4.1 Timer 0: Timer Tick 11.4.2 Timer 1: Memory Refresh 11.4.3 Timer 2: Speaker Frequency 11.5 Interrupt System 11.5.1 Initializing the IDT 11.5.2 Initializing the 8259 PICs 11.5.3 Installing a New Interrupt Service RoutineAppendix A: Contents of the CD-ROMAppendix B: The DJGPP C/C++ Compiler Installation Compilation On-Line Documentation (Info)Appendix C: The NASM Assembler Installation Running NASMAppendix D: Programming Projects Files Required from the CD-ROM for All Applications Files Required for Nonpreemptive Multithreaded Applications Files Required for Preemptive Multithreaded Applications Compiling and Assembling Your Embedded Application Linking Your Embedded Application Preparing the Boot Diskette Running Your Embedded ApplicationAppendix E: The LIBEPC Library Memory Layout and Initialization Display Functions (display.c) Window Functions (window.c) Keyboard Functions (keyboard.c) Speaker Functions (speaker.c) Timer Functions (timer.c, cycles.asm) Interrupt Vector Access Functions (init-idt.c) Dynamic Memory Allocation Functions (heap.c) Fixed Point (fixedpt.asm) Interfunction Jumps (setjmp.asm) Miscellaneous Functions (init-crt.c)Appendix F: The Boot Loader Index

<<嵌入式软件基础>>

版权说明