

About This Book

A Warning

A Moving Target

Terminology

32-/64-bit x86 Instruction Set Architecture Specification

The Specification Is the Final Word

Book Organization

Documentation Conventions

Hexadecimal Notation

Binary Notation

Decimal Notation

Bits Versus Bytes Notation

Bit Fields (Logical Groups of Bits or Signals)

Visit Our Web Site

We Want Your Feedback

Part 1: Introduction

Chapter 1: Introduction

Two x86 ISA Architectures 1

Basic Execution Modes 2

IA-32 Sub-Modes 6

IA-32e Sub-Modes 9

Mode Switching Basics 11

Initial Switch from IA-32 to IA-32e Mode 11

Protected/Compatibility 16-/32-bit Submodes 13

CS Descriptor's D/L Bits Control Mode 14

Miscellaneous Terms and Concepts 20

Processors, Cores and Logical Processors 20

IA Instructions vs. Micro-ops 22

RISC Instructions Sets Are Simple 22

x86 Instruction Set Is Complex 22

But You Can't Leave It Behind 22

Complexity vs. Speed Dictated a Break With the Past 23

Why Not Publish a Micro-Op ISA? 23

Some Important Definitions 23

Virtual vs. Physical Memory 23

Some Other Important Terms 24

Contents

Chapter 2: A (very) Brief History

Major Evolutionary Developments 29

16-bit Mode Background 34

8086 and Real Mode 34

286 Introduced 16-bit Protected Mode 36

386 Supported Both 16- and 32-bit Protected Mode 39

The Intel Microarchitecture Families 43

A Brief Timeline 45

Chapter 3: State After Reset

State After Reset 53

Soft Reset 61

Initial Memory Reads 61

Boot Strap Processor (BSP) Selection 62

Not Part of the Instruction Set Architecture 62

Introduction 63

The BSP Selection Process 64

How the APs are Discovered and Configured 66

AP Detection and Configuration 66

Introduction 66

The BIOS's AP Discovery Procedure 67

Uni-Processor OS 68

MP OS 68

The FindAndInitAllCPUs Routine 71

Part 2: IA-32 Mode

Chapter 4: Intro to the IA-32 Ecosystem

General 77

Control Registers 78

XCR0 (XFEM) 79

CR0 80

Address Translation (Paging) Control Registers 85

CR2 85

CR3 86

CR4 (Feature Control Register) 87

EFlags Register 92

General Purpose Registers (GPRs) 96

EAX, EBX, ECX and EDX Registers 96

- EBP Register 97
- Index Registers 97
- Code, Data, and Stack Registers Define Memory Regions 98**
 - General 98
 - Segment Register Operation in Real Mode 99
 - Segment Registers in Protected Mode 101
 - So Much Stuff, Such a Small Register! 102
 - Segment Register Loads a Segment Descriptor from Memory 102
- Descriptor Table Registers 105**
 - Global Descriptor Table Register (GDTR) 105
 - Interrupt Descriptor Table Register (IDTR) 106
 - Local Descriptor Table Register (LDTR) 107
 - When a New Value Is Loaded Into a Segment Register 108
- Effective/Virtual/Linear/Physical Addresses 110**
- Introduction to Address Translation (Paging) 110**
 - RAM Is Finite and Can't Hold Everything 110
 - RAM and Mass Storage Are Managed on a Page Basis 111
 - This Requires a Series of Directories 111
 - Malloc Request 111
 - Problem: Non-Contiguous Memory Allocation 112
 - Malloc Returns a Virtual Address to the Application 112
 - IA-32 Applications Have a 4GB Virtual Address Space 113
- Instruction Pointer Register 113**
- Task Register 114**
 - A Task Data Structure Per Task 114
 - Task Register Points to Current Task's TSS 115
- x87 FPU/MMX Register Set 118**
- SSE Register Set 119**
- Debug Breakpoint Registers 120**
- Local APIC Register Set 121**
- Architecturally-Defined MSRs 123**
 - General 123
 - Determining MSR Support 123
 - Accessing the MSRs 123

Chapter 5: Instruction Set Growth

- Why a Comprehensive Instruction Set Listing Isn't Included 133
- 386 Instruction Set 133
- Instruction Set (as of March, 2009) 140

Contents

Chapter 6: 32-bit Machine Language Instruction Format

64-bit Machine Language Instruction Format 179

A Complex Instruction Set with Roots in the Past 179

Effective Operand Size 180

Introduction 180

In 16- and 32-bit Code Segments 180

In 64-bit Code Segments 181

Instruction Composition 183

Instruction Format Basics 184

Opcode (Instruction Identification) 190

In the Beginning 190

1-byte Opcodes 191

2-byte Opcodes 195

2nd-Level Opcode Map Introduced in 286 195

Instructions with 2-byte Opcodes: Four Possible Forms 195

3-byte Opcodes Use 3-Level Lookup 199

3-Level Opcode Maps Introduced in Pentium 4 Prescott 199

Currently There Are Two 3rd-Level Maps Defined 199

Instructions with 3-byte Opcodes: Three Possible Forms 199

Special Use of Prefix Bytes 200

Opcode Micro-Maps (Groups) 203

Micro-Maps Associated with 1-byte Opcodes 203

Some Opcodes Employ 2 x 8 Micro-Maps 203

Micro-Maps Associated with 2-byte Opcodes 206

3-byte Opcodes Don't Use Micro-Maps 210

x87 FP Opcodes Inhabit Opcode Mini-Maps 210

Special Opcode Fields 212

Operand Identification 217

General 217

Specifying Registers as Operands 218

Implicit Register Specification 219

Explicit Register Specification in Opcode 219

Explicit Register Specification in ModRM Byte 219

Addressing a Memory-Based Operand 222

Instruction Can Specify Only One Memory-Based Operand 222

Addressing Memory Using the ModRM Byte 224

When Effective Address Size = 16 Bits 224

When Effective Address Size = 32 Bits 225

Using the SIB Byte to Access a Data Structure 227

Near and Far Branch Addressing 230

Specifying an Immediate Value As an Operand 233

Instruction Prefixes 234

- Operand Size Override Prefix (66h) 235
 - In 32-bit Mode 235
 - In 16-bit Mode 237
- Special Usage of 66h Prefix 237
- Address Size Override Prefix (67h) 237
 - In 32-Bit Mode 238
 - In 16-Bit Mode 238
- Lock Prefix 238
 - Shared Resource Concept 238
 - Testing Availability and Gaining Ownership of Shared Resources 239
 - Race Condition Can Present Problem 240
 - Guaranteeing Atomicity of Read/Modify/Write 240
 - Use Locked RMW to Obtain and Give Up Semaphore Ownership 241
 - Instructions That Accept Lock Prefix 241
- Repeat Prefixes 242
 - Normal Usage 242
 - Special Usage 243
- Segment Override Prefix 243
 - General 243
 - Usage In String Operations 244
 - Segment Override Use With MMX and SSE1-4 Instructions 244
- Branch Hint Prefix 244

Summary of Instruction Set Formats 245

Chapter 7: Real Mode (8086 Emulation)

8086 Emulation 251

Unused Facilities 253

- Unused Protected Mode Tables and Registers 253
- Other Unused Facilities 253

Single Task Execution Environment 254

- Single-Task OS Environment Overview 254
 - Command Line Interpreter (CLI) 254
 - Program Loader 254
 - OS Services 255

Direct IO Access 256

- Application Program Memory Usage 256
- Task Initiation, Execution and Termination 256

Running Real Mode Applications Under a Protected Mode OS 257

Real Mode Applications Aren't Supported in IA-32e Mode 257

IO Operations 257

- IO Operations in IO Address Space 257

Contents

- IN and OUT Instructions 257
- Block (String) IO Operations 258
 - Block Transfer from IO Port to Memory 258
 - Block Transfer from Memory to an IO Port 259
- IO Space is Limited and Crowded 259
- Memory-Mapped IO Operations 260
 - Introduction 260
 - Know the Characteristics of Your Target 260
 - Why the Processor Must Know the Memory Type 260
 - Uncacheable (UC) Memory 261
- No IO Protection 262
- Memory Addressing 262**
 - Default Address and Operand Sizes 262
 - No Address Translation 262
 - Introduction to Real Mode Segmentation 263
 - All Segments are 64KB in Size 267
 - Accessing the Code Segment 268
 - Jumping Between Code Segments 268
 - Far Jumps and Calls 268
 - Near Jumps and Calls 269
 - IP-Relative Branches 270
 - Operations That Default to the Code Segment 270
 - Accessing the Stack Segment 271
 - Introduction 271
 - Stack Characteristics 272
 - Pushing Data Onto the Stack 273
 - Popping Data From the Stack 274
 - Processor Stack Usage 276
 - Accessing Parameters Passed on the Stack 276
 - Operations That Default To the Stack Segment 277
 - Accessing the DS Data Segment 278
 - General 278
 - Operations That Default to the DS Data Segment 278
 - Accessing the ES/FS/GS Data Segments 280
 - General 280
 - Operations That Default to the ES Data Segment 280
 - An Example 281
 - Accessing Extended Memory in Real Mode 282
 - Big Real Mode 286
 - 286 DOS Extender Programs 287
 - Hot Reset and 286 DOS Extender Programs 287
 - Alternate (Fast) Hot Reset 289

- 286 DOS Extenders on Post-286 Processors 290
- Real Mode Interrupt/Exception Handling 292**
 - Events and Event Handlers 292
 - The IDT 292
 - Definition of the IDT 292
 - IDT Initialization 299
 - Stack Initialization 300
 - Event Handling 300
 - Software Event Types 302
 - Software Exceptions 302
 - Definition of an Exception 303
 - Exception Handling 303
 - Events Are Recognized on an Instruction Boundary 304
 - Three Categories of Software Exceptions 305
 - Software Interrupts 306
 - INT nn Instruction 306
 - BOUND Instruction 307
 - INTO Instruction 307
 - INT3 (Breakpoint) Instruction 307
 - Hardware Event Types 307
 - NMI 308
 - Definition of NMI and Delivery Mechanisms 308
 - External Masking Mechanism 308
 - NMI Handling 308
 - SMI 309
 - Maskable Interrupts 310
 - Maskable Interrupts Are Originated by Hardware Devices 310
 - Enabling/Disabling Recognition of Maskable Interrupts 310
 - Selective Masking of Maskable Interrupts 310
 - Maskable Interrupt Delivery Mechanisms 310
 - IDT Entries Associated with Maskable Interrupts 311
 - Handling Maskable Interrupts 311
 - Machine Check Exception 312
- Summary of Real Mode Limitations 313**
- Transitioning to Protected Mode 313**

Chapter 8: Legacy x87 FP Support

- A Little History 315
- x87 FP Instruction Format 316
- FPU-Related CR0 Bit Fields 316
- x87 FPU Register Set 320
 - The FP Data Registers 321

Contents

x87 FPU's Native Data Operand Format	322
32-bit SP FP Numeric Format	324
Background	324
A Brief IEEE FP Primer	324
The 32-bit SP FP Format	325
Representing Special Values	326
An Example	326
Another Example	327
DP FP Number Representation	328
FCW Register	329
FSW Register	330
FTW Register	333
Instruction Pointer Register	333
Data Pointer Register	334
Fopcode Register	334
General	334
Fopcode Compatibility Mode	334
FP Error Reporting	335
Precise Error Reporting	335
Imprecise (Deferred) Error Reporting	335
Why Deferred Error Reporting Is Used	336
The WAIT/FWAIT Instruction	336
CR0[NE]	337
DOS-Compatible FP Error Reporting	337
FP Error Reporting Via Exception 16	337
Ignoring FP Errors	338

Chapter 9: Introduction to Multitasking

Concept 341

An Example—Timeslicing 342

Another Example—Awaiting an Event 342

1. Task Issues Call to OS for Disk Read 342
2. OS Initiates Disk Read 343
3. OS Suspends Task 343
4. OS Makes Entry in Event Queue 343
5. OS Starts or Resumes Another Task 343
6. Disk-Generated Interrupt Causes Jump to OS 344
7. Interrupted Task Suspended 344
8. Task Queue Checked 344
9. OS Resumes Task 344

Chapter 10: Multitasking-Related Issues

Hardware-based Task Switching Is Slow! 345

Private (Local) and Global Memory 346

Preventing Unauthorized Use of OS Code 346

With Privilege Comes Access 347

 Program Privilege Level 347

 The CPL 347

 Calling One of Your Equals 348

 Calling a Procedure to Act as Your Surrogate 348

 Data Segment Protection 348

 Data Segment Privilege Level 348

 Read-Only Data Areas 349

Some Code Segments Contain Data, Others Don't 349

IO Port Anarchy 351

No Interrupts, Please! 352

BIOS Calls 353

Chapter 11: Summary of the Protection Mechanisms

Introduction 355

Chapter 12: Protected Mode Memory Addressing

Real Mode Limitations 359

An Important Reminder: Segment Base + Offset = Virtual Address 360

Descriptor Contains Detailed Segment Description 361

Segment Register—Selects Descriptor Table and Entry 361

Introduction to the Descriptor Tables 364

 Segment Descriptors Reside in Memory 364

 Global Descriptor Table (GDT) 366

 GDT Description 366

 Setting the GDT Base Address and Size 367

 GDT Entry 0 367

 Local Descriptor Tables (LDTs) 368

 Creating and Selecting an LDT 369

 LDT May Be Selected in One of Two Ways 373

General Segment Descriptor Format 374

 Granularity Bit and the Segment Size 374

 Segment Base Address Field 375

 Default/Big Bit 375

 In a Code Segment Descriptor, D/B = "Default" Bit 375

 Override Prefixes 376

Contents

- In a Stack Segment Descriptor, D/B = “Big” Bit 377
 - Segment Type Field 378
 - Introduction to the Type Field 378
 - Non-System Segment Types 378
 - Segment Present Bit 381
 - Descriptor Privilege Level (DPL) Field 382
 - System Bit 382
 - Available Bit 383
 - Goodbye to Segmentation 384**
 - Introduction 384
 - IA-32 Flat Memory Model 384
 - No Protection? Paging Takes Care of It 387
 - A Reminder of Where We Are 388**
-

Chapter 13: Code, Calls and Privilege Checks

- Selecting the Active Code Segment 389**
 - CS Descriptor 391**
 - CS Descriptor Selector 391
 - Calculating the Descriptor’s Memory Address 391
 - Descriptor Read and Privilege Checked 392
 - CS Descriptor Format 392
 - Accessing the Code Segment 396**
 - In-Line Code Fetching 396
 - Short and Near Branches (Jumps and Calls) 396
 - General 396
 - Example Near Jump 397
 - Far Branches (Far Jumps and Calls) 397
 - General 397
 - Example Far Jump 398
 - Short/Near Jumps 400**
 - General 400
 - No Privilege Check 400
 - Unconditional Short/Near Branches 400
 - Conditional Branches 401
 - General 401
 - Loop Instructions 404
 - Unconditional Far Jumps 407**
 - The Privilege Check 407
 - Far Jump Targets 408
 - Far Jump Forms 408
 - Privilege Checking 409**
 - No Check on Near Calls or Near Jumps 409
-

Contents

- General 410
- Definitions 410
 - Definition of a Task 410
 - Definition of a Procedure 410
 - CPL Definition 410
 - CS DPL Definition 411
 - Conforming and Non-Conforming Code Segments 411
 - Definition 411
 - Examples 411
 - RPL Definition 412
 - General 412
 - RPL Usage in Privilege Check 412
 - RPL Use on RET or IRET 412
- Privilege Check on Far Call or Far Jmp 413
 - General 413
 - Example 413
- Jumping from a Higher-to-Lesser Privileged Program 414**
- Direct Procedure Calls 415**
 - Introduction 415
 - General 415
 - Near Calls/Returns 416
 - Description 416
 - Call/RET Operand Size Matching 417
 - Near Call/RET Forms 418
 - Far Calls 420
 - General 420
 - Far Call Forms 420
 - Far Call, Same Privilege Level 424
 - Far Call to a More-Privileged Procedure 424
 - Far Call to a Procedure in a Different Task 425
- Indirect Procedure Far Call Though a Call Gate 425**
 - Example Scenario Defines the Problem 425
 - The Scenario 425
 - The Problem 426
 - The Solution—Different Gateways 427
 - The Call Gate Descriptor 428
 - Call Gate Example 430
 - Execution Begins 430
 - Call Gate Descriptor Read 431
 - Call Gate Contains Target Code Segment Selector 432
 - Target Code Segment Descriptor Read 433
 - The Big Picture 435

Contents

- The Call Gate Privilege Check 436
- Automatic Stack Switch 436
 - Background 436
 - A Potential Problem 437
 - The Solution: Pre-Allocated Stacks 437
- Far Call From 32-bit CS to 16-bit CS 440**
 - General 440
 - Method 1: Far Call with Operand Size Override Prefix 440
 - Method 2: Far Call Via 16-bit Call Gate 441
 - Method 3: Call 32-bit/16-bit Interface Procedure 444
- Far Call From 16-bit CS to 32-bit CS 445**
 - Method 1: Far Call With an Operand Size Prefix 446
 - Method 2: Far Call Via a 32-bit Call Gate 447
- Far Returns 450**
 - General 450
 - Far Return Forms 451
 - Near Call/RET Forms 451

Chapter 14: Data and Stack Segments

- A Note Regarding Stack Segments 455**
- The Data Segments 455**
 - Selecting and Accessing a Data Segment 455
 - Data Segment Privilege Check 457
- Selecting and Accessing a Stack Segment 459**
 - Introduction 459
 - Expand-Up Stack 460
 - Expand-Down Stack 462
 - The Problem 462
 - Expand-Down Stack Description 463
 - An Example 463
 - Another Example 464
 - Stack Segment Privilege Check 468

Chapter 15: IA-32 Address Translation Mechanisms

- Three Generations 469**
- Demand Mode Paging Evolution 470**
- Background 471**
 - Memory and Disk: Block-Oriented Devices 471
 - Definition of a Page 472
 - Example Scenario: Block Transfer from Disk to Memory 472
- A Poor Memory Allocation Strategy 473**

Applications Are Presented With a Simplified World-View 474

- Introduction 474
- Life Without Paging Would Be Chaotic 474
- The Virtual World Is a Simply One 475
- Virtual Address Space Partitioning 481
- Example Virtual Buffer Allocation 482

Address Translation Advantages 483

- Introduction 483
- Simplifies Memory Management 484
- Efficient Memory Usage 484
 - A Wasteful Approach 484
 - A Better Approach: Load On Demand 485
- Attribute Assignment 485
- Track Access History 486
- Allows DOS Applications to Co-Exist 486
 - Problem: Running Multiple DOS Programs 486
 - Solution: Address Redirection 486

First-Generation Paging 487

- Definition of First Generation Paging 487
- Paging Logic's Interpretation of a Virtual Address 488
- First-Generation Paging Overview 489
 - The Set-Up 489
 - Virtual-to-Physical Address Translation 490
 - The Goal 490
 - The Translation 490

Two Overhead Memory Reads Take a Toll 502

- The TLBs 502
 - TLB Miss 503
 - TLB Hit 503
- TLB Maintenance 504
 - TLBs Are Cleared on Task Switch or Page Directory Change 505
 - Updating a Single Page Table Entry 505
- Global Pages 506
 - Problem 506
 - Global Page Feature 506

Enabling Paging 507

Detailed Description of PDE and PTE 509

- PDE Layout 509
- PTE Layout 511

Checking Page Access Permission 515

- The Privilege Check 515
 - Segment Privilege Check Takes Precedence Over Page Check 515

Contents

- U/S Bit in PDE and PTE Are Checked 516
- Accesses with Special Privilege 517
- The Read/Write Check 517
- Missing Page or Page Table 518**
- Introduction 518
- Page Table Not Present 518
- Page Not Present 522
- Page Faults 525
 - Page Fault Causes 525
 - Second Page Fault while in the Page Fault Handler 526
 - Page Fault During a Task Switch 526
 - Page Fault while Changing to a Different Stack 527
 - Page Fault Error Code 527
 - Additional Page Fault Information 528
- Access History 529**
- 4MB Pages 530**
- Basic Concept 530
- Enabling the PSE Feature 530
- Simplifies Housekeeping 531
- How To Set Up a 4MB Page 531
- The Address Translation 532
- Turning PAE Mode On Disables CR4[PSE] 532
- Second-Generation Paging 533**
- First-Gen Problem: 4GB Physical Memory 533
- The Solution: PAE-36 Mode 534
- Enabling PAE-36 Mode 534
- CR4[PSE] Is “Don’t Care” 534
- Application Still Limited to a 4GB Virtual Address Space 535
- Virtual Address Space Partitioning 535
 - First Generation Partitioning 535
 - Second Generation Partitioning 536
- Second Generation Uses 3-Level Lookup Mechanism 537
- CR3 Points to PDPT in Lower 4GB 538
- Enlarged Physical Address Space 539
- The Translation 540
 - Step 1: PDPT Lookup 540
 - Step 2: Page Directory Lookup 542
 - Step 2a: PDE Points to a Page Table 543
 - Step 2b: PDE Points to a 2MB Physical Page 544
 - Step 3: Page Table Lookup 546
- Page Protection Mechanisms 547
- 3-Level Lookup—Increased TLB Size 550

- Microsoft PAE Support 551
- Linux PAE Support 552
- PSE-36 Mode (PAE-36 Mode's Poor Cousin) 552**
 - PSE-36 Mode Background 553
 - Detecting PSE-36 Mode Capability 553
 - Enabling PSE-36 Mode 553
 - Per Application Virtual Memory Space = 4GB 554
 - First-Generation Lookup Mechanism 554
 - Selected PDE Can Point to 4KB Page Table or a 4MB Page 554
 - Virtual Address Maps to a 4MB Page in 64GB Space 555
 - Windows and PSE36 556
 - AMD Enhances PSE-36 to PSE-40 557
- Execute Disable Feature 557**
 - Problem: Malicious Code 557
 - The Overflow 557
 - The Exploit 559
 - The Fix: Intercept Code Fetches from Data Pages 560
 - Enabling the Execute-Disable Feature 560
 - How It Works 561
- Defining a Page's Caching Rules 565**
 - Introduction 565
 - Table Caching Rules 565
 - General 565
 - First-Generation Paging Tables 566
 - Second-Generation Paging Tables 566
 - Page Caching Rules 566
 - PAT Feature (Page Attribute Table) 567
 - What's the Problem? 567
 - Detecting PAT Support 567
 - PAT Allows More Memory Types 567
 - Default Contents of IA32_CR_PAT MSR 569
 - Memory Type When Page Definition and MTTR Disagree 570
 - General 570
 - The UC- Memory Type 570
 - Altering IA32_CR_PAT MSR 574
 - Ensuring IA32_CR_PAT and MTRR Consistency 574
 - Assigning Multiple Memory Types to a Single Physical Page 576
 - Compatibility with Earlier IA-32 Processors 577
- The Write Protect Feature 578**
 - Description 578
 - Example Usage: Unix Copy-on-Write Strategy 579
- Third Generation Paging 581**

Contents

Chapter 16: Memory Type Configuration

Characteristics of Memory Targets 583

Introduction 583

Example Problem: Caching from MMIO 583

Early Processors Implemented Primitive Mechanism 584

Solution/Problem: Chipset Memory Type Registers 584

Solution: Memory Type Register Set 585

MTRR Feature Determination 585

MTRRs Are Divided Into Four Categories 586

MTRRDefType Register 587

State of the MTRRs after Reset 587

Fixed-Range MTRRs 588

The Problem: Legacy Issues 588

Enabling the Fixed-Range MTRRs 588

Defining Memory Types in Lower 1MB 588

Variable-Range MTRRs 590

Enabling Variable-Range Register Pairs 590

How Many Variable-Range Register Pairs? 591

Variable-Range Register Pair Format 591

 MTRRPhysBasen Register 591

 MTRRPhysMaskn Register 592

Programming Variable-Range Register Pairs 592

Memory Types 592

Memory Type Defines Processor Aggressiveness 592

Five Memory Types 593

Uncacheable (UC) Memory 593

Uncacheable Write-Combining (WC) Memory 594

Cacheable Write-Protect (WP) Memory 595

Cacheable Write-Through (WT) Memory 596

Cacheable Write-Back (WB) Memory 596

Rules as Defined by MTRRs 597

Memory Type Provided in Memory Transaction 599

Paging Also Defines Memory Type 599

In an MP System, MTRRs Must Be Synchronized 600

Posted-Write Related Issues 600

General 600

Synchronizing Events 600

PMWB and WCBs Aren't Snooped 601

WCB Usage 602

An Example 602

All WCBs in Use 609

Draining the WCBs 610

Chapter 17: Task Switching

Hardware- vs. Software-Based Task Switching 611

A Condensed Conceptual Overview 612

A More Comprehensive Overview 612

The Scheduler and the Task Queue 612

Setting Up a Task 613

The Task Data Structure 613

The LDT 614

The TSS 615

The Address Translation Tables 616

The GDT and GDTR Register 617

The LDTR Register 618

The Task Register (TR) 619

Starting a Task 620

Suspend Task and Resume Scheduler Execution 621

Hardware-Based Task Switching 623

It's Slow 623

Why Didn't OSs Use It? 624

Why Wasn't It Improved? 624

Why Does It Still Exist? 624

Introduction to the Key Elements 624

The Trigger Events 628

The Descriptors 630

 TSS Descriptor 630

 Task Gate Descriptor 631

 Task Gate Selected by a Far Call/Jump 631

 Gate Selected by Hardware Interrupt or Software Exception 632

 Task Gate Selected by a Software Interrupt Instruction 632

The Task Register 634

 General 634

 TR Instruction Pair 635

 STR Instruction 635

 LTR Instruction 635

TSS Data Structure Format 637

 General 637

 Required Fields 638

 Optional Fields 638

 Register Snapshot Area 639

 LDT Selector Field 639

 Segment Register Fields 640

Contents

- General Register Fields 640
 - SS:ESP Register Pair Fields 640
 - Extended Flags (Eflags) Register Field 640
 - CS:EIP Register Pair Fields 640
 - Control Register 3 (CR3) Field 641
- Debug Trap Bit (T) 642
- IO Port Access Protection 642
 - IO Protection in Real Mode 642
 - Definition of IO Privilege Level (IOPL) 642
 - IO Permission Check in Protected Mode 643
 - IO Permission Check in VM86 Mode 644
- IO Permission Bit Map 644
 - Required or Optional? 644
 - The Bitmap Offset Field 645
 - The Permission Check 645
- Interrupt Redirection Bit Map 646
- OS-Specific Data Structures 647
 - Privilege Level 0 - 2 Stack Definition Fields 647
 - Link Field (to Old TSS Selector) 647
- Comprehensive Task Switch Description 648
- Calling Another Task 653
 - An Overview 653
 - A Comprehensive Example 654
 - LTR Instruction and the Busy Bit 659
 - When Is Busy Cleared? 659
 - Critical Error: Switching to a Busy Task 660
 - Busy Toggle Is a Locked Operation 660
 - Linkage Modification 660
- Task Switching and Address Translation 661
 - One GDT to Serve Them All 661
 - Each Task Can Have Different Virtual-to-Physical Mapping 661
 - TSS Mapping Must Remain the Same for All Tasks 662
 - Placement of a TSS Within a Page(s) 662
 - Switch from Higher-Privilege Code to Lower 663

Chapter 18: Protected Mode Interrupts and Exceptions

Handler vs. ISR 665

Real Mode Interrupt/Exception Handling 665

The IDT 667

- General 667
- Protected Mode IDT and the IDTR 668
- The Gates 671

Contents

- Introduction 671
- Interrupt Gate 674
- Trap Gate 676
- Actions Taken When Interrupt or Trap Gate Selected 677
- Actions Taken When Task Gate Selected 679
- Interrupt/Exception Event Categories 680**
- General Event Handling 682**
- State Saved on Stack (but which stack?) 687**
- Return to the Interrupted Program 692**
 - General 692
 - The IRET Instruction 693
- Maskable Hardware Interrupts 698**
 - General 698
 - Maskable Interrupt Vector Delivery 698
 - PC-Compatible Vector Assignment 699
 - Actions Performed by the Handler 704
 - Effect of CLI/STI Execution 705
 - General 705
 - Other Events That Affect Interrupt Flag Bit 707
 - Protected Mode Virtual Interrupt Feature 707
- Non-Maskable Interrupt (NMI) Requests 708**
 - PC-Compatible NMI Logic 708
 - NMI Description 708
- More Detailed Coverage of Hardware Interrupt Handling 709**
- Machine Check Exception 709**
- SMI (System Management Interrupt) 710**
- Software Interrupts 710**
 - INT nn Instruction 711
 - INTO Instruction 711
 - BOUND Instruction 712
 - INT3 (Breakpoint) Instruction 712
- Software Exceptions 712**
 - General 712
 - Faults, Traps, and Aborts 713
 - Instruction Restart After a Fault 720
 - Exception Error Codes 720
- Interrupt/Exception Priority 725**
- Detailed Description of Software Exceptions 729**
 - Divide-by-Zero Exception (0) 729
 - Processor Introduced In 729
 - Exception Class 729
 - Description 729

Contents

- Error Code 729
- Saved Instruction Pointer 729
- Processor State 729
- Debug Exception (1) 730
 - Processor Introduced In 730
 - Exception Class 730
 - Description 730
 - Error Code 730
 - Saved Instruction Pointer 731
 - Processor State 731
 - The Resume Flag Prevents Multiple Debug Exceptions 731
- NMI (2) 732
 - Processor Introduced In 732
 - Exception Class 732
 - Error Code 732
 - Saved Instruction Pointer 732
 - Processor State 732
- Breakpoint Exception (3) 732
 - Processor Introduced In 732
 - Exception Class 732
 - Description 733
 - Error Code 733
 - Saved Instruction Pointer 733
 - Processor State 733
- Overflow Exception (4) 733
 - Processor Introduced In 733
 - Exception Class 734
 - Description 734
 - Error Code 734
 - Saved Instruction Pointer 734
 - Processor State 734
- Array Bounds Check Exception (5) 734
 - Processor Introduced In 734
 - Exception Class 734
 - Description 734
 - Error Code 735
 - Saved Instruction Pointer 735
 - Processor State 735
- Invalid OpCode Exception (6) 735
 - Processor Introduced In 735
 - Exception Class 735
 - Description 735

Contents

Error Code	736
Saved Instruction Pointer	736
Processor State	736
Device Not Available (DNA) Exception (7)	737
Processor Introduced In	737
Exception Class	737
Description	737
General	737
X87 FPU Emulation	737
CR0[TS]: Task Switch, But FP/SSE Registers Not Saved	737
CR0[MP]	738
Error Code	738
Saved Instruction Pointer	738
Processor State	738
Double Fault Exception (8)	738
Processor Introduced In	738
Exception Class	738
Description	738
Shutdown Mode	741
Error Code	742
Saved Instruction Pointer	742
Processor State	742
Coprocessor Segment Overrun Exception (9)	742
Processor Introduced In	742
Exception Class	742
Description	742
Error Code	742
Saved Instruction Pointer	743
Processor State	743
Invalid TSS Exception (10)	743
Processor Introduced In	743
Exception Class	743
Description	743
Error Code	745
Saved Instruction Pointer	745
Processor State	745
Segment Not Present Exception (11)	746
Processor Introduced In	746
Exception Class	746
Description	746
Error Code	746
Saved Instruction Pointer	747

Contents

- Processor State 747
- Stack Exception (12) 747
 - Processor Introduced In 747
 - Exception Class 748
 - Description 748
 - Error Code 748
 - Saved Instruction Pointer 749
 - Processor State 749
- General Protection (GP) Exception (13) 749
 - Processor Introduced In 749
 - Exception Class 749
 - Description 749
 - Error Code 751
 - Saved Instruction Pointer 752
 - Processor State 752
- Page Fault Exception (14) 752
 - Processor Introduced In 752
 - Exception Class 752
 - Description 752
 - Error Code 753
 - CR2 754
 - Saved Instruction Pointer 754
 - Processor State 755
 - The More Common Case 755
 - Page Fault During a Task Switch 755
 - Page Fault During a Stack Switch 755
- Vector (Exception) 15 756
- FPU Exception (16) 756
 - Processor Introduced In 756
 - Exception Class 756
 - Description 756
 - Handling of Masked Errors 757
 - Handling of Unmasked Errors 758
 - Error Code 759
 - Saved Instruction Pointer 759
 - Processor State 759
- Alignment Check Exception (17) 760
 - Processor Introduced In 760
 - Background: Misaligned Transfers Affect Performance 760
 - Alignment Is Important! 760
 - Exception Class 761
 - Description 761

Implicit Privilege Level 0 Accesses	762
Storing GDTR, LDTR, IDTR or TR	762
FP/MMX/SSE Save and Restore Accesses	763
MOVUPS and MOVUPD Accesses	763
FSAVE and FRSTOR Accesses	763
Error Code	763
Saved Instruction Pointer	763
Processor State	764
Machine Check Exception (18)	764
Processor Introduced In	764
Exception Class	764
Description	764
Error Code	764
Saved Instruction Pointer	765
Processor State	765
SIMD Floating-Point Exception (19)	765
Processor Introduced In	765
Exception Class	765
Description	765
Exception Error Code	767
Saved Instruction Pointer	768
Processor State	768
Legacy Problem: 2-Step SS:ESP Update	768
Problem Description	768
The Solution	768

Chapter 19: Virtual 8086 Mode

A Special Note	769
Real Mode Applications Are Dangerous	769
Solution: a Watchdog	770
Real Mode Applications Run at Privilege Level 3	772
Switching Between Protected Mode and VM86 Mode	772
Eflags[VM] = 1 Switches Processor into VM86 Mode	772
But Software Cannot Directly Access Eflags[VM]	773
Scheduler Activates VM86 Mode	773
Exiting VM86 Mode	774
Determining Interrupted Task Is a Real Mode Task	774
Returning to VM86 Mode from VMM	775
VMM Passes Control to Real Mode Interrupt/Exception Handler	775
Real Mode Application's World View	775
The DOS World	775
Memory Address Formation in VM86 Mode	776

Contents

- Multiple DOS Domains Partitioned in Separate 1MB Areas 778
- VMM Should Not Reside in the HMA 780
- Dealing with Segment Wraparound 781
 - 8088/8086 Processor 781
 - 286 and Later Processors 781
 - Solutions 782
- Using the Address Size Override Prefix 782
- Sensitive Instructions 782**
 - Problematic Instructions 782
 - CLI (Clear Interrupt Enable) Instruction 783
 - STI (Set Interrupt Enable) Instruction 784
 - PUSHF (Push Flags) Instruction 785
 - POPF (Pop Flags) Instruction 785
 - INT nn (Software Interrupt) Instruction 785
 - IRET (Interrupt Return) Instruction 785
 - Solution: IOPL Sensitive Instructions 786
- Handling Direct IO 786**
 - The Problem 786
 - IO-Mapped IO 787
 - IO Permission in Protected Mode 787
 - IO Permission in VM86 Mode 788
 - Memory-Mapped IO 789
 - To Permit an Access 790
 - To Deny an Access 790
 - For Finer Control 790
 - Handling Display Frame Buffer Updates 790
- Handling Exceptions 791**
 - Processor Actions 791
 - Option 1: Protected Mode Handler Services Exception 793
 - Option 2: Handler Passes Exception to VMM for Servicing 793
 - Option 3: Exception Handled by Another Task 795
- Hardware Interrupt Handling 796**
 - NMI, SMI, and Maskable Interrupts 796
 - Real Mode Application's Unreal Reality 798
 - VM86 Task Executes CLI When VME = 0 799
 - CLI Handling 799
 - Subsequent High-Priority Interrupt Detected 800
 - Servicing of Lower-Priority Interrupt Deferred 800
 - STI/POPF/PUSHF/IRET Handling When VME = 0 803
 - Attempted Execution of STI Instruction (VME = 0) 803
 - Attempted Execution of PUSHF Instruction (VME = 0) 804
 - Attempted Execution of POPF Instruction (VME = 0) 804

- Attempted Execution of IRET Instruction (VME = 0) 805
- CLI/STI/POPF/PUSHF Handling When VME = 1 805
 - VM86 Extensions 805
 - Background 806
 - When VME = 1 and IOPL = 3, Task Can Control Eflags[IF] 806
 - When VME = 1 and IOPL < 3, Task Controls VIF, Not IF 807
 - Eflags[VIP] Is Controlled by the VMM 807
 - Software Cannot Directly Access Eflags[VIP] 807
 - CLI Followed by a Maskable Interrupt 808
 - Subsequent STI Effect Depends on Eflags[VIP] 809
 - A Special Case 812
 - POPF/PUSHF Handling 812
- Software Interrupt Instruction Handling 813**
 - Software Interrupt Handling in Protected Mode 813
 - Software Interrupt Handling in VM86 Mode 813
 - INT3 Is Special 813
 - VMM Passes Control To Real Mode Handler 817
- Halt Instruction in VM86 Mode 821**
- Protected Mode Virtual Interrupt Feature 821**
 - General 821
 - 1. Task executes CLI, Clears VIF 822
 - 2. Maskable Interrupt Occurs and Is deferred 822
 - 3. Task Executes STI 822
- Registers Accessible in Real/VM86 Mode 823**
- Instructions Usable in Real/VM86 Mode 823**

Chapter 20: The MMX Facilities

- Introduction 825
- Detecting MMX Capability 826
- The Basic Problem 826
 - Assumptions 826
 - The Operation 827
 - Example: Processing One Pixel Per Iteration 827
 - Example: Processing Four Pixels Per Iteration 828
- MMX SIMD Solution 830
- Dealing with Unpacked Data 831
- Dealing with Math Underflows and Overflows 832
- Elimination of Conditional Branches 833
 - Introduction 833
 - Non-MMX Chroma-Key/Blue Screen Compositing Example 833
 - MMX Chroma-Keying/Blue Screen Compositing Example 835
 - Changes To the Programming Environment 837

Contents

Handling a Task Switch 838
MMX Instruction Set Syntax 838

Chapter 21: The SSE Facilities

Chapter Objectives 841

SSE: MMX on Steroids 841

Streaming SIMD Extensions (SSE) 846

- The Motivation Behind SSE 846
- Detecting SSE Support 846
- The SSE Elements 847
- SSE Data Types 848
- The MXCSR 849
 - MXCSR Description 849
 - Loading and Storing the MXCSR 852
- SIMD (Packed) Operations 852
- Scalar Operations 853
- Cache-Related Instructions 854
 - Overlapping Data Prefetch with Program Execution 854
 - Streaming Store Instructions 857
 - Introduction 857
 - The MOVNTPS Instruction 859
 - MOVNTQ Instruction 860
 - MASKMOVQ Instruction 861
- Ensuring Delivery of Writes Before Proceeding 862
 - An Example Scenario 862
 - SFENCE Instruction 863
- Elimination of Mispredicted Branches 866
 - Background 866
 - SSE Misprediction Enhancements 866
 - Comparisons and Bit Masks 866
 - Min/Max Determination 867
 - The Masked Move Operation 867
- Reciprocal and Reciprocal Square Root Operations 867
- MPEG-2 Motion Compensation 868
- Optimizing 3D Rasterization Performance 869
- Optimizing Motion-Estimation Performance 869
- Accuracy vs. Fast Real-Time 3D Processing (FTZ) 870
- SSE Alignment Checking 870
- The SIMD FP Exception 870
- Saving and Restoring x87/MMX/SSE Registers 871
 - General 871
 - MXCSR Mask Field 871

- OS Support for SSE 873
 - General 873
 - Enable SSE Instruction Sets and Register Set Save/Restore 873
 - Enable the SSE SIMD FP Exception 874
- SSE Setup 874
- Summary of the SSE Instruction Set 875
- The SSE2 Instruction Set 876**
 - General 876
 - DP FP Number Representation 877
 - Packed and Scalar DP FP Instructions 877
 - SSE2 64-Bit and 128-Bit SIMD Integer Instructions 878
 - SSE2 128-Bit SIMD Integer Instruction Extensions 878
 - Your Choice: Accuracy or Speed (DAZ) 879
 - The Cache Line Flush Instruction 880
 - Fence Instructions 881
 - MFENCE Instruction 881
 - LFENCE Instruction 882
 - General 882
 - LFENCE Ordering Rules 883
 - SFENCE Instruction 884
 - Non-Temporal Store Instructions 884
 - Introduction 884
 - MOVNTDQ Instruction 884
 - MOVNTPD Instruction 885
 - MOVNTI Instruction 886
 - MASKMOVDQU Instruction 887
 - General 887
 - When a Mask of All Zeros Is Used 887
 - PAUSE Instruction 888
 - Thread Synchronization 888
 - The Problem 889
 - The Fix 889
 - When A Thread Is Idle 890
 - Spin-Lock Optimization 891
 - Branch Hints 891
- SSE3 Instruction Set 891**
 - Introduction 891
 - Improved x87 FP-to-Integer Conversion Instruction 892
 - The Problem 892
 - The Solution 893
 - New Complex Arithmetic Instructions 893
 - Improved Motion Estimation Performance 894

Contents

- The Problem 894
- The Solution 894
- The Downside 895
- Instructions to Improve Processing of a Vertex Database 895
- MONITOR/MWAIT Instruction Pair 897
 - Background 897
 - Monitor Instruction 898
 - Mwait Instruction 898
 - Example Code Usage 899
 - The Wake Up Call 899

Part 3: IA-32e OS Environment

Chapter 22: IA-32e OS Environment

- The Big Picture 903**
- Mode Switching Overview 905**
 - Booting Into Protected Mode 905
 - Initial Switch from IA-32 to IA-32e Mode 906
 - CS D and L Bits Control IA-32e Sub-Mode Selection 908
- Old and New Applications Running Under a 64-bit OS 911**
- Things You Lose In IA-32e Mode (hint: not much) 912**
- Old Applications Live in an Expanded Universe 912**
 - Old Legacy Universe = 4GB or 64GB 912
 - IA-32e Universe Is At Least 16 Times Larger 913
- Virtual Memory Addressing in IA-32e Mode 914**
 - Virtual Address in Compatibility Mode 914
 - Virtual Address in 64-bit Mode 914
- In Compatibility Mode, Segmentation Is Operative 915**
- In 64-bit Mode, Hardware-Enforced Flat Model 916**
 - General 916
 - New Segment Selector Causes Descriptor Read 916
 - Segment Register Usage in 64-bit Mode 916
 - Ignored Descriptor Fields and Assumed Segment Characteristics 924
 - 64-bit Mode: No Limit Checking = No Limits? 925
 - Table Limit Checks Are Performed 926
 - Stack Management 926
 - Stack Management in Compatibility Mode 926
 - Stack Management in 64-bit Mode 927
 - Push/Pop Size is 64-bits 927
 - Address Translation Replaces Limit Checking 927
 - Segment Override Prefixes Other Than FS/GS Are Ignored 928

Protection Provided by Paging 928
Segment Registers Preserved On Mode Switch 928

64-bit Instruction Pointer 929

Instruction Fetching 929
RIP-Relative Data Accesses 930

Changes To Kernel-Related Registers and Structures 930

Address Translation Mechanism 930
 Basic Description 930
 Top-Level Directory Placement 931
 Detailed Description 931
GDT/LDT Descriptor Changes 931
GDT and GDTR Changes 939
 GDT Descriptor Types 939
 Executing LGDT in 64-bit Mode 942
 Unaligned Accesses to GDT or LDT 943
LDT and LDTR Changes 944
 LDT Descriptor Types 944
 LDTR Contents in IA-32e Mode 945
 Unaligned Accesses to LDT 945
IDT/IDTR and Interrupt/Exception Changes 947
 IDT Descriptor Types 947
 Interrupt/Trap Gate Operational Changes 950
 General 950
 Interrupt/Exception Stack Switch 951
 Motivation for the IST 952
 IRET Behavior 952
 Executing LIDT in 64-bit Mode 955
 All Accesses to IDT Are Properly Aligned 956
IA-32e Call Gate Operation 956
 General 956
 IA-32e Call Gate Detailed Operation 956
 IA-32e Call Gate Stack Switch 957
TR and TSS Changes 959
 General 959
 Illegal For Jump or Call To Select a TSS Descriptor 964
 Executing LTR in Compatibility Mode 964
 Executing LTR in 64-bit Mode 965
 Revised TSS Structure 965
 TSS Usage 967
 General 967
 Call Gate Stack Switch 968
 Interrupt/Exception Stack Switch 968

Contents

- Register Set Expansion (in 64-bit Mode) 968**
 - Scheduler's Software-Based Task Switching Mechanism 969**
 - Switching to a 64-bit Task 969
 - Switching to a Legacy Task 971
-

Chapter 23: IA-32e Address Translation

- Theoretical Address Space Sizes 975**
 - Limitation Imposed by Current Implementation 975**
 - Four-Level Lookup Mechanism 976**
 - Address Space Partitioning 976
 - The Address Translation 978
 - Initializing CR3 978
 - Step 1: PML4 Lookup 978
 - Step 2: PDPT Lookup 980
 - Step 3: Page Directory Lookup 982
 - Step 3a: PDE Points to a Page Table 984
 - Step 3b: PDE Points to a 2MB Physical Page 984
 - Step 4: Page Table Lookup 987
 - Page Protection Mechanisms in IA-32e Mode 989
 - Page Protection in Compatibility Mode 989
 - Page Protection in 64-bit Mode 989
 - Don't Forget the Execute Disable Feature! 990
 - TLBs Are More Important Than Ever 993**
 - No 4MB Page Support 994**
-

Part 4: Compatibility Mode

Chapter 24: Compatibility Mode

- Initial Entry to Compatibility Mode 997**
 - Switching Between Compatibility Mode and 64-bit Mode 997**
 - Differences Between IA-2 Mode and Compatibility Mode 998**
 - IA-32 Background 998
 - Unsupported IA-32 Features 998
 - Changes to the OS Environment 998
 - Memory Addressing 1000**
 - Segmentation 1000
 - FS/GS Segments 1001
 - Virtual Address 1001
 - Address Translation 1001
 - Register Set 1002**
 - Visible Registers 1002
-

No Access to Additional or Extended Registers	1003
Control Register Accesses	1003
Debug Register Accesses	1003
Register Preservation Across Mode Switches	1003
Exception and Interrupt Handling	1004
OS Kernel Calls	1004
Call Gates	1004
Kernel Call Instruction Usage	1005
SysEnter Instruction	1005
SysCall instruction	1005
Odds and Ends	1005
IRET Changes	1005
Segment Load Instructions	1006

Part 5: 64-bit Mode

Chapter 25: 64-bit Register Overview

Overview of 64-bit Register Set	1009
EFER (Extended Features Enable) Register	1010
Sixteen 64-bit Control Registers	1012
64-bit Rflags Register	1017
Sixteen 64-bit GPRs	1018
Kernel Data Structure Registers in 64-bit Mode	1021
SSE Register Set Expanded in 64-bit Mode	1022
Debug Breakpoint Registers	1023
Local APIC Register Set	1024
x87 FPU/MMX Register Set	1025
Architecturally-Defined MSRs	1025

Chapter 26: 64-bit Operands and Addressing

Helpful Background	1027
Switching to 64-bit Mode	1027
The Defaults	1027
The REX Prefix	1028
Problem 1: Addressing New Registers	1028
Problem 2: Using 16- and 64-bit Operands	1030
Solution: The Rex Prefix	1030
Making Room for REX	1030
REX Prefix Positioning	1031
When You Need REX...	1032
...and when you don't	1032

Contents

Anatomy of a REX Prefix	1034
General	1034
The Width Bit	1036
The Register Bit	1036
Description	1036
An Example	1037
The IndeX and Base Bits	1039
Description	1039
An Example	1040
Addressing Registers Using REX[B] + Opcode[Reg]	1042
Addressing Registers Using REX[B] + ModRM[RM]	1042
Byte-Register Addressing Limitations	1043
Sometimes, REX Fields Have No Effect	1043
Addressing Memory in 64-bit Mode	1043
64-bit Mode Uses a Hardware-Enforced Flat Model	1043
CS, DS, ES, and SS Segments Start at Virtual Address 0	1043
CS/DS/ES/SS Segment Override Prefixes Ignored	1044
FS and GS Segments Can Start at Non-Zero Base Addresses	1044
FS/GS Segment Override Prefixes Matter	1044
Default Virtual Address Size (and overriding it)	1045
Actual Address Size Support: Theory vs. Practice	1046
Canonical Address	1047
General	1047
32- (and 16-) bit Addressing Limited to Lower 4GB	1048
32-bit Address Treatment in 64-bit Mode	1048
Address Treatment in Compatibility Mode	1048
Memory-based Operand Address Computation	1049
RIP-relative Data Addressing	1053
Near and Far Branch Addressing	1054
Immediate Data Values in 64-bit Mode	1056
Displacements in 64-bit Mode	1057

Chapter 27: 64-bit Odds and Ends

New Instructions	1059
General	1059
SwapGS Instruction	1059
The Problem	1059
The SwapGS Solution	1060
MOVSXD Instruction: Stretch It Out	1061
Enhanced Instructions	1061
Invalid Instructions	1062
Reassigned Instructions	1064

LAHF/SAHF Instruction Support	1064
Instructions That Default to a 64-bit Operand Size	1065
Stack Operations	1065
Near Branches	1065
Branching in 64-bit Mode	1066
Short/Near Branches Default to 64-bit Operand Size	1066
Unconditional Jumps in 64-bit Mode	1067
Calls/Ret/Iret in 64-bit Mode	1069
Instruction Forms in 64-bit Mode	1069
Example Call/Return Operations	1072
64-bit Near Call/Return	1072
32-bit Level 3 Code Calls 64-bit Level 3 Procedure	1073
32-bit Level 3 Code Calls 64-bit Level 2 Procedure	1074
Previous Example Plus Call to Level 0 Procedure	1076
Conditional Branches in 64-bit Mode	1076
NOP Instruction	1080
FXSAVE/FXRSTOR	1080
General	1080
Fast FxSave/Restore Feature (AMD-only)	1080
The Nested Task Bit (Rflags[NT])	1084
SMM Save Area	1085
IA-32 Processor SM Save Area	1085
Intel 64 Processor SM Save Area	1089

Part 6: Mode Switching Detail

Chapter 28: Transitioning to Protected Mode

Real Mode Peculiarities That Affect the OS Boot Process	1097
Typical OS Characteristics	1098
Flat Model With Paging	1098
Software-Based Task Switching	1099
Protected Mode Transition Primer	1099
GDT Must Be In Place Before Switch to Protected Mode	1099
No Interrupts or Exceptions During Mode Switch	1103
Creation of Protected Mode IDT	1104
Other Protected Mode Structures	1105
TSS	1105
Address Translation Mechanism	1106
Protected Mode Is a Prerequisite	1106
Identity Mapping	1106
Which Translation Mechanism?	1107

Contents

Optional Structure: LDT	1107
Enable A20 Gate	1108
Load Initial Code and Handlers Into Memory	1109
The Switch to Protected Mode	1109
Loading Segment Registers With GDT Descriptors	1109
Load TSS Descriptor Into TR	1111
Enable Interrupts	1111
Load Application Into Memory	1111
Create Task's Address Translation Tables	1112
Switching From OS Scheduler to First Task	1112
Example: Linux Startup	1112
1. Bootsect	1112
2. Setup	1113
3a. Startup_32 in boot/compressed/head.s	1115
3b. Startup_32 in kernel/head.s	1115

Chapter 29: Transitioning to IA-32e Mode

No Need to Linger in Protected Mode	1123
Entering Compatibility Mode	1123
Switch to 64-bit Mode	1125

Part 7: Other Topics

Chapter 30: Introduction to Virtualization Technology

Just an Introduction?	1129
Detailed Coverage of Virtualization	1129
OS: I Am the God of All Things!	1130
Virtualization Supervisor: Sure You Are (:<)	1131
Root versus Non-Root Mode	1131
Detecting VMX Capability	1132
Entering/Exiting VMX Mode	1133
Entering VMX Mode	1133
Exiting VMX Mode	1133
Virtualization Elements/Terminology	1133
Introduction to the VT Instructions	1134
Introduction to the VMCS Data Structure	1136
Preparing to Launch a Guest OS	1140
Launching a Guest OS	1141
Guest OS Suspension	1142
Resuming a Guest OS	1142
Some Warnings Regarding VMCS Accesses	1143

Chapter 31: System Management Mode (SMM)

What Falls Under the Heading of System Management? 1146

The Genesis of SMM 1146

SMM Has Its Own Private Memory Space 1147

The Basic Elements of SMM 1147

A Very Simple Example Scenario 1148

How the Processor Knows the SM Memory Start Address 1149

Normal Operation, (Including Paging) Is Disabled 1149

The Organization of SM RAM 1149

General 1149

IA-32 Processor SM State Save Area 1150

Intel 64 Processor SM Save Area 1155

Entering SMM 1159

The SMI Interrupt Is Generated 1159

No Interruptions Please 1159

General 1159

Exceptions and Software Interrupts Permitted but Not Recommended 1160

Servicing Maskable Interrupts While in the Handler 1160

Single-Stepping through the SM Handler 1160

If Interrupts/Exceptions Permitted, Build an IDT 1161

SMM Uses Real Mode Address Formation 1161

NMI Handling While in SMM 1162

Default NMI Handling 1162

How to Re-Enable NMI Recognition in the SM Handler 1162

If an SMI Occurs within the NMI Handler 1163

Informing the Chipset SM Mode Has Been Entered 1164

General 1164

A Note Concerning Memory-Mapped IO Ports 1164

The Context Save 1164

General 1164

Although Saved, Some Register Images Are Forbidden Territory 1164

Special Actions Required on a Request for Power Down 1165

The Register Settings on Initiation of the SM Handler 1165

The SMM Revision ID 1167

The Body of the Handler 1168

Exiting SMM 1168

The Resume Instruction 1168

Informing the Chipset That SMM Has Been Exited 1169

The Auto Halt Restart Feature 1169

Executing the HLT Instruction in the SM Handler 1170

The IO Instruction Restart Feature 1171

Contents

- Introduction 1171
- An Example Scenario 1171
- The Detail 1171
- Back-to-Back SMIs During IO Instruction Restart 1172
- Multiple Core/Processor System Presents a Problem 1172
- Caching from SM Memory 1174**
 - Background 1174
 - The Physical Mapping of SM RAM Accesses 1175
 - FLUSH# and SMI# 1179
- Setting Up the SMI Handler in SM Memory 1179**
- Relocating the SM RAM Base Address 1180**
 - Description 1180
 - In an MP System, Each Processor Must Have a Separate State Save Area 1180
 - Accessing SM RAM Above the First MB 1181
- SMM in an MP System 1181**
- SM Mode and Virtualization 1182**

Chapter 32: Machine Check Architecture (MCA)

- Why This Subject Is Included in This Book 1183**
- MCA = Hardware Error Logging Capability 1183**
- The MCA Elements 1184**
 - The Machine Check Exception 1184
 - The MCA Register Set 1185
- The Global Registers 1186**
 - Introduction 1186
 - The Global Count and Present Register 1187
 - The Global Status Register 1188
 - The Global Control Register 1188
 - The Extended MC State MSRs 1189
- The Composition of a Register Bank 1192**
 - Overview 1192
 - The Bank Control Register 1192
 - General 1192
 - P6 and Core Processors 1193
 - The Bank Status Register 1193
 - General 1193
 - Error Valid Bit 1194
 - Overflow Bit 1195
 - Uncorrectable Error Bit 1195
 - Error Enabled Bit 1195
 - Miscellaneous Register Valid Bit 1195
 - Address Register Valid Bit 1196

- Processor Context Corrupt Bit 1196
- MCA Error Code and Model Specific Error Code 1196
- Other Information 1196
- The Bank Address Register 1196
- The Bank Miscellaneous Register 1196
- The Error Code 1197**
 - The Error Code Fields 1197
 - Simple MCA Error Codes 1197
 - Compound MCA Error Codes 1198
 - General 1198
 - Correction Report Filtering Bit 1199
 - Example External Interface Error Interpretation 1203
- Cache Error Reporting 1206**
 - Green/Yellow Cache Health Indicator 1206
 - Background 1206
 - TES (Threshold Error Status) Feature 1206
 - Interrupt On Soft Error Threshold Match 1207
 - Before CMCI, Soft Error Logging Required Periodic Scan 1207
 - CMCI Eliminates MC Register Scan 1208
 - Determining Processor's CMCI Support 1209
 - Determining a Bank's CMCI Support 1209
 - CMCI Interrupt Is Separate and Distinct From MC Exception 1209
 - CMC Interrupt May Affect Multiple Cores/Logical Processors 1209
 - CMC Interrupt Should Only Be Serviced Once 1209
- MC Exception Is Generally Not Recoverable 1210**
- Machine Check and BINIT# 1211**
- Additional Error Logging Notes 1211**
 - Error Buffering Capability 1211
 - Additional Information for Each Log Entry 1211

Chapter 33: The Local and IO APICs

- APIC and the IA-32 Architecture 1213**
- Hardware Context Is Essential 1213**
- A Short History of the APIC's Evolution 1214**
 - APIC Introduction 1214
 - Pentium Pro APIC Enhancements 1214
 - The Pentium II and Pentium III 1215
 - Pentium 4 APIC Enhancements: xAPIC 1215
 - The x2APIC Architecture 1216
- Before the APIC 1216**
- MP Systems Need a Better Interrupt Distribution Mechanism 1219**
 - Legacy Interrupt Delivery System Is Inefficient 1219

Contents

The APIC Interrupt Distribution Mechanism	1221
Introduction	1221
Message Types	1222
Inter-Processor Interrupt (IPI) Messages	1222
NMI, SMI and Init Messages	1222
Legacy Interrupt Message	1222
Message Transfer Mechanism Prior to the Pentium 4	1223
Message Transfer Mechanism Starting with the Pentium 4	1223
Message Transfer Mechanism in QPI-based Systems	1224
Processors Reside in Clusters	1224
Each Core/Logical Processor Has a Dedicated Local APIC	1225
Introduction to the Message Addressing Modes	1225
Detecting Presence/Version/Capabilities of Local APIC	1226
Presence	1226
Version	1227
x2APIC Capability Verification	1227
Local APIC's Initial State	1227
Enabling/Disabling the Local APIC	1228
General	1228
Disabling Local APIC for Remainder of Power-Up Session	1229
Dynamically Enabling/Disabling Local APIC	1230
Mode Selection	1231
The Local APIC Register Set	1232
Register Access in xAPIC Mode: MMIO	1232
General	1232
Local and IO APIC xAPIC Register Areas Are Uncacheable	1232
xAPIC Register Access Alignment	1232
Register Access in x2APIC Mode: MSR	1233
Introduction to the Local APIC's Register Set	1234
Local APIC ID Assignments and Addressing	1248
ID Assignment in xAPIC Mode	1248
Introduction	1248
Cluster ID Assignment	1248
Physical/Logical Processor and Local APIC ID Assignment	1249
Example Xeon MP System with Hyper-Threading Disabled	1249
Example Xeon MP System with Hyper-Threading Enabled	1250
Dual Processor System with Hyper-Threading Enabled	1250
A Single-Processor System with Hyper-Threading Enabled	1251
xAPIC ID Register	1252
BIOS/OS Reassignment of xAPIC ID	1253
Logical xAPIC Address Assignment	1253
Maximum Number of xAPICs	1254

Contents

ID Assignment in x2APIC Mode	1254
Two Hardware-Assigned Local APIC IDs	1254
x2APIC ID (Physical Local APIC ID)	1255
General	1255
Some Interesting Questions	1255
CPLID Provides the Answers	1255
x2APIC ID and Physical Destination Mode	1256
Obtaining the x2APIC ID	1256
Logical x2APIC ID	1256
xAPIC Logical Addressing Background	1256
Logical x2APIC ID Formation	1257
Logical x2APIC ID Usage	1258
Local APIC Addressing	1259
Physical Addressing: Single Target	1259
Logical Addressing: Multiple Targets	1259
Introduction	1259
x2APIC Cluster Model	1259
Flat Model	1260
Flat Cluster Model	1261
Hierarchical Cluster Model	1262
Message Addressing Summary	1263
Lowest-Priority Delivery Mode	1265
General	1266
Warnings Related to Lowest-Priority Delivery Mode	1266
Chipset-Assisted Lowest-Priority Delivery	1267
Local APIC IDs Are Stored in the MP and ACPI Tables	1268
Accessing the Local APIC ID	1268
An Introduction to the Interrupt Sources	1269
Local Interrupts	1269
Remote Interrupt Sources	1270
Introduction to Interrupt Priority	1271
General	1271
Definition of a User-Defined Interrupt	1272
User-Defined Interrupt Priority	1273
Definition of Fixed Interrupts	1276
Masking User-Defined Interrupts	1276
Task and Processor Priority	1276
Introduction	1276
The Task Priority Register (TPR)	1277
The Processor Priority Register (PPR)	1278
The User-Defined Interrupt Eligibility Test	1278
CR8 (Alternative TPR)	1279

Contents

IO/Local APICs Cooperate on Interrupt Handling 1280

- The Purpose of the IO APIC 1280
- Overview of Edge-Triggered Interrupt Handling 1283
 - Assumptions 1283
 - Description 1283
- Overview of Level-Sensitive Interrupt Handling 1288
 - Assumptions 1288
 - Description 1289
- Higher-Priority Fixed Interrupt Preempts Handler 1293
- IO APIC Register Set 1298
 - IO APIC Register Set Base Address 1298
 - IO APIC Register Set Description 1299
 - IRQ Pin Assertion Register 1302
 - IO APIC EOI Register and Shared Interrupts 1302
 - Non-Shareable IRQ Lines 1302
 - Shareable IRQ Lines 1303
 - Linked List of Interrupt Handlers 1303
 - How It Works 1303
 - Broadcast Versus Directed EOI 1304
 - IO APIC ID Register 1307
 - IO APIC Version Register 1307
 - IO APIC Redirection Table (RT) Register Set 1308
- IO APIC Interrupt Delivery Order Is Rotational 1311

Message Signaled Interrupts (MSI) 1311

- General 1311
- Using the IO APIC as a Surrogate Message Sender 1312
- Direct-Delivery of an MSI 1313
- Memory Already Sync'd When Interrupt Handler Entered 1317
 - The Problem 1317
 - Old Solution 1317
 - How MSI Solves the Problem 1317

Interrupt Delivery from Legacy 8259a Interrupt Controller 1318

- Virtual Wire Mode A 1318
- Virtual Wire Mode B 1319

SW-Initiated Interrupt Message Transmission 1321

- Introduction 1321
- Sending a Message From the Local APIC 1322
 - ICR in XAPIC Mode 1322
 - ICR in X2APIC Mode 1323

x2APIC Mode's Self IPI Feature 1329

Locally Generated Interrupts 1330

- Introduction 1330

- The Local Vector Table 1331
 - The Pentium Family's LVT 1331
 - The P6 Family's LVT 1331
 - The Pentium 4 Family's LVT 1332
 - Core Processor's LVT 1332
- Local Interrupt 0 (LINT0) 1332
 - Introduction 1332
 - The Mask Bit 1332
 - The Trigger Mode and the Input Pin Polarity 1333
 - The Delivery Mode 1333
 - The Vector Field 1335
 - The Remote IRR Bit 1335
 - The Delivery Status 1335
- Local Interrupt 1 (LINT1) 1335
- The Local APIC Timer 1336
 - General 1336
 - The Divide Configuration Register 1337
 - One Shot Mode 1337
 - Periodic Mode 1337
- The Performance Counter Overflow Interrupt 1338
- The Thermal Sensor Interrupt 1340
- Correctable Machine Check (CMC) Interrupt 1342
- The Local APIC's Error Interrupt 1343
 - Local APIC Error LVT Register 1343
 - Error Status Register (ESR) Operation in xAPIC Mode 1343
 - Error Status Register Operation in x2APIC Mode 1343
- The Spurious Interrupt Vector 1346**
 - The Problem 1346
 - The Solution 1346
 - Additional Spurious Vector Register Features 1347
- Boot Strap Processor (BSP) Selection 1348**
 - Introduction 1348
 - The BSP Selection Process 1348
- How the APs are Discovered and Configured 1351**
 - AP Detection and Configuration 1351
 - Introduction 1352
 - BIOS AP Discovery Procedure 1352
 - Uni-Processor OS and the APs 1354
 - MP OS and the APs 1354
 - The FindAndInitAllCPUs Routine 1357

Glossary 1361

Contents
