

Contents

Preface	vii
PART I: Introductory Topics	1
1 Introduction	3
1.1 A User's View of Computer Systems	4
1.2 What Is Assembly Language?	6
1.3 Advantages of High-Level Languages	7
1.4 Why Program in Assembly Language?	8
1.5 Typical Applications	10
1.6 Why Learn Assembly Language?	11
1.7 Performance: C Versus Assembly Language	11
1.8 Summary	14
1.9 Exercises	15
1.10 Programming Exercises	16
1.11 Program Listings	17
2 Basic Computer Organization	21
2.1 Basic Components of a Computer System	22
2.2 The Processor	23
2.2.1 The Pentium Processor	24
2.2.2 The Pentium Registers	25
2.2.3 The System Clock	30
2.2.4 The Intel 80X86 Processor Family	31
2.3 Memory	32
2.3.1 Two Basic Memory Operations	33
2.3.2 Types of Memory	34
2.3.3 Storing Multibyte Data	36

2.4	Pentium Memory Architecture	37
2.4.1	Real Mode Memory Architecture	38
2.4.2	Protected Mode Memory Architecture	43
2.4.3	Segment Registers	44
2.4.4	Segment Descriptors	45
2.4.5	Segment Descriptor Tables	47
2.4.6	Segmentation Models	48
2.4.7	Mixed Mode Operation	48
2.4.8	Which Segment Register to Use?	49
2.5	Input/Output	50
2.5.1	Accessing I/O Devices	52
2.6	Performance: Effect of Data Alignment	52
2.7	Summary	55
2.8	Exercises	56
2.9	Programming Exercises	57
3	Overview of Assembly Language	59
3.1	Assembly Language Statements	60
3.2	Data Allocation	62
3.3	Where Are the Operands?	71
3.3.1	Register Addressing Mode	72
3.3.2	Immediate Addressing Mode	72
3.3.3	Direct Addressing Mode	73
3.3.4	Indirect Addressing Mode	74
3.4	Data Transfer Instructions	75
3.4.1	The mov Instruction	75
3.4.2	Ambiguous Moves: PTR Directive	77
3.4.3	The xchg Instruction	77
3.4.4	The xlat Instruction	78
3.5	Overview of Assembly Language Instructions	78
3.5.1	Simple Arithmetic Instructions	79
3.5.2	Conditional Execution	82
3.5.3	Iteration Instruction	85
3.5.4	Logical Instructions	86
3.5.5	Shift Instructions	89
3.5.6	Rotate Instructions	90
3.6	Defining Constants	93
3.6.1	The EQU Directive	93
3.6.2	The = Directive	94

3.7	Illustrative Examples	95
3.8	Performance: When to Use the xlat Instruction	106
3.9	Summary	108
3.10	Exercises	109
3.11	Programming Exercises	112

PART II: Basic Topics **115**

4 Procedures and the Stack **117**

4.1	What Is a Stack?	118
4.2	Pentium Implementation of the Stack	119
4.3	Stack Operations	122
4.3.1	Basic Instructions	122
4.3.2	Additional Instructions	123
4.4	Uses of the Stack	123
4.4.1	Temporary Storage of Data	124
4.4.2	Transfer of Control	125
4.4.3	Parameter Passing	125
4.5	Procedures	125
4.6	Assembler Directives for Procedures	128
4.7	Pentium Instructions for Procedures	129
4.7.1	How Is Program Control Transferred?	129
4.7.2	The ret Instruction	130
4.8	Parameter Passing	131
4.8.1	Register Method	131
4.8.2	Stack Method	135
4.8.3	Preserving Calling Procedure State	139
4.8.4	Which Registers Should Be Saved?	140
4.8.5	Illustrative Examples	142
4.9	Handling a Variable Number of Parameters	149
4.10	Local Variables	153
4.11	Multiple Source Program Modules	159
4.11.1	PUBLIC Directive	160
4.11.2	EXTRN Directive	160
4.12	Performance: Procedure Overheads	163
4.12.1	Stack Versus Registers	164
4.12.2	Comparison of C and Assembly Language Versions	165
4.12.3	Local Variable Overhead	167
4.13	Summary	168

4.14 Exercises	168
4.15 Programming Exercises	170
5 Addressing Modes	173
5.1 Simple Addressing Modes	174
5.1.1 Register Addressing Mode	174
5.1.2 Immediate Addressing Mode	175
5.2 Memory Addressing Modes	176
5.2.1 Direct Addressing	178
5.2.2 Register Indirect Addressing	179
5.2.3 Based Addressing	181
5.2.4 Indexed Addressing	182
5.2.5 Based-Indexed Addressing	183
5.3 Illustrative Examples	184
5.4 Arrays	191
5.4.1 One-Dimensional Arrays	192
5.4.2 Multidimensional Arrays	193
5.4.3 Examples of Arrays	195
5.5 Performance: Usefulness of Addressing Modes	198
5.6 Summary	201
5.7 Exercises	202
5.8 Programming Exercises	203
6 Arithmetic Flags and Instructions	207
6.1 Status Flags	208
6.1.1 The Zero Flag	209
6.1.2 The Carry Flag	211
6.1.3 The Overflow Flag	214
6.1.4 The Sign Flag	217
6.1.5 The Auxiliary Flag	218
6.1.6 The Parity Flag	220
6.1.7 Flag Examples	221
6.2 Arithmetic Instructions	223
6.2.1 Addition Instructions	223
6.2.2 Subtraction Instructions	225
6.2.3 Multiplication Instructions	227
6.2.4 Division Instructions	231
6.3 Application Examples	234
6.3.1 PutInt8 Procedure	235

6.3.2	GetInt8 Procedure	237
6.4	Multiword Arithmetic	241
6.4.1	Addition and Subtraction	241
6.4.2	Multiplication	242
6.4.3	Division	247
6.5	Performance: Multiword Multiplication	250
6.6	Summary	251
6.7	Exercises	252
6.8	Programming Exercises	254
7	Selection and Iteration	257
7.1	Unconditional Jump	258
7.2	Compare Instruction	262
7.3	Conditional Jumps	263
7.3.1	Jumps Based on Single Flags	263
7.3.2	Jumps Based on Unsigned Comparisons	265
7.3.3	Jumps Based on Signed Comparisons	267
7.4	Looping Instructions	269
7.5	Implementing High-Level Language Decision Structures	272
7.5.1	Selective Structures	272
7.5.2	Iterative Structures	275
7.6	Illustrative Examples	278
7.7	Indirect Jumps	284
7.7.1	Multiway Conditional Statements	286
7.8	Evaluation of logical expressions	289
7.8.1	Full Evaluation	289
7.8.2	Partial Evaluation	289
7.9	Performance: Logical Expression Evaluation	291
7.10	Summary	293
7.11	Exercises	293
7.12	Programming Exercises	295
8	Logical and Bit Operations	299
8.1	Logical Instructions	300
8.1.1	The and Instruction	300
8.1.2	The or Instruction	304
8.1.3	The xor Instruction	306
8.1.4	The not Instruction	310
8.1.5	The test Instruction	310

8.2	Shift Instructions	311
8.2.1	Logical Shift Instructions	312
8.2.2	Arithmetic Shift Instructions	315
8.2.3	Why Use Shifts for Multiplication and Division?	318
8.2.4	Double Shift Instructions	318
8.3	Rotate Instructions	319
8.3.1	Rotate Without Carry	319
8.3.2	Rotate Through Carry	321
8.4	Logical Expressions in High-Level Languages	323
8.4.1	Representation of Boolean Data	323
8.4.2	Logical Expressions	323
8.4.3	Bit Manipulation	325
8.5	Bit Instructions	325
8.5.1	Bit Test and Modify Instructions	325
8.5.2	Bit Scan Instructions	326
8.6	Illustrative Examples	327
8.7	Performance: Shift Versus Multiplication	333
8.8	Summary	334
8.9	Exercises	335
8.10	Programming Exercises	338
PART III: Advanced Topics		341
9	String Processing	343
9.1	String Representation	344
9.1.1	Explicitly Storing String Length	344
9.1.2	Using a Sentinel Character	345
9.2	String Instructions	345
9.2.1	Repetition Prefixes	346
9.2.2	Direction Flag	348
9.2.3	String Move Instructions	349
9.2.4	String Compare Instruction	352
9.2.5	Scanning a String	354
9.3	Illustrative Examples	355
9.4	Testing String Procedures	368
9.5	Performance: Advantage of String Instructions	370
9.6	Summary	372
9.7	Exercises	473
9.8	Programming Exercises	474

10	Macros and Conditional Assembly	377
10.1	What Are Macros?	378
10.2	Macros with Parameters	380
10.3	Macros Versus Procedures	381
10.4	Labels in Macros	385
10.5	Comments in Macros	386
10.6	Macro Operators	388
10.7	List Control Directives	392
10.8	Repeat Block Directives	394
10.8.1	REPT Directive	394
10.8.2	WHILE Directive	396
10.8.3	IRP and IRPC Directives	397
10.9	Conditional Assembly	400
10.9.1	IF and IFE Directives	401
10.9.2	IFDEF and IFNDEF Directives	403
10.9.3	IFB and IFNB Directives	405
10.9.4	IFIDN and IFDIF Directives	406
10.10	Nested Macros	408
10.11	Performance: Macros Versus Procedures	409
10.12	Summary	413
10.13	Exercises	414
10.14	Programming Exercises	415
11	ASCII and BCD Arithmetic	417
11.1	ASCII and BCD Representations of Numbers	418
11.1.1	ASCII Representation	418
11.1.2	BCD Representation	419
11.2	Processing in ASCII Representation	420
11.2.1	ASCII Addition	420
11.2.2	ASCII Subtraction	422
11.2.3	ASCII Multiplication	423
11.2.4	ASCII Division	423
11.2.5	Example: Multidigit ASCII Addition	424
11.3	Processing Packed BCD Numbers	426
11.3.1	Packed BCD Addition	426
11.3.2	Packed BCD Subtraction	427
11.3.3	Example: Multibyte Packed BCD Addition	428
11.4	Performance: Decimal Versus Binary Arithmetic	431
11.5	Summary	435

11.6 Exercises	435
11.7 Programming Exercises	437
12 Interrupts and Input/Output	439
12.1 Introduction	440
12.2 A Taxonomy of Interrupts	441
12.3 Interrupt Processing	443
12.3.1 Interrupt Processing in Protected Mode	443
12.3.2 Interrupt Processing in Real Mode	444
12.4 Software Interrupts	446
12.5 Keyboard Services	447
12.5.1 Keyboard Description	447
12.5.2 DOS Keyboard Services	447
12.5.3 Extended Keyboard Keys	451
12.5.4 BIOS Keyboard Services	454
12.6 Text Output to Display Screen	460
12.7 Printer Support	461
12.7.1 DOS Printer Services	462
12.7.2 BIOS Printer Support	462
12.8 Exceptions	464
12.9 Hardware Interrupts	469
12.10 Direct Control of I/O Devices	470
12.10.1 Accessing I/O Ports	471
12.11 Peripheral Support Chips	472
12.11.1 8259 Programmable Interrupt Controller	472
12.11.2 8255 Programmable Peripheral Interface Chip	475
12.12A Hardware Interrupt Example	477
12.13 Performance: Polling Versus Interrupts	482
12.14 Summary	484
12.15 Exercises	485
12.16 Programming Exercises	486
13 High-Level Language Interface	489
13.1 Why Program in Mixed-Mode?	490
13.2 Overview	491
13.3 Calling Assembly Procedures from C	493
13.3.1 Parameter Passing	493
13.3.2 Returning Values	495
13.3.3 Preserving Registers	496

13.3.4	Publics and Externals	497
13.3.5	Illustrative Examples	497
13.4	Calling C Functions from Assembly	502
13.5	Simplified Calling Mechanisms	505
13.5.1	The ARG Directive	505
13.5.2	Extended CALL Instruction	507
13.6	Inline Assembly Code	509
13.7	Summary	511
13.8	Exercises	511
13.9	Progammng Exercises	513
Appendices		517
A	Internal Data Representation	519
A.1	Positional Number Systems	520
A.1.1	Notation	521
A.2	Number Systems Conversion	523
A.2.1	Conversion to Decimal	523
A.2.2	Conversion from Decimal	525
A.2.3	Conversion among Binary, Octal, and Hexadecimal	527
A.3	Unsigned Integer Representation	530
A.3.1	Arithmetic on Unsigned Integers	531
A.4	Signed Integer Representation	538
A.4.1	Signed-Magnitude Representation	538
A.4.2	Excess-M Representation	539
A.4.3	1's Complement Representation	540
A.4.4	2's Complement Representation	543
A.5	Floating-Point Representation	545
A.5.1	Fractions	545
A.5.2	Representing Floating-Point Numbers	549
A.5.3	Floating-Point Representation	550
A.6	Character Representation	554
A.7	Summary	556
A.8	Exercises	557
A.9	Progammng Exercises	559
B	Assembling and Linking Assembly Language Programs	561
B.1	Structure of Assembly Language Programs	562
B.2	Input/Output Routines	565

B.3	Assembling and Linking	570
B.3.1	The Assembly Process	570
B.3.2	Linking Object Files	579
B.4	Summary	580
B.5	Exercises	581
B.6	Programming Exercises	581
C	Debugging Assembly Language Programs	583
C.1	Strategies to Debug Assembly Language Programs	584
C.2	DEBUG	586
C.2.1	Miscellaneous Group	591
C.2.2	An Example	591
C.3	Turbo Debugger TD	595
C.4	CodeView	601
C.5	Summary	602
C.6	Exercises	603
C.7	Programming Exercises	604
D	Pentium Instruction Set	605
D.1	Pentium Instruction Format	605
D.1.1	Instruction Prefixes	606
D.1.2	General Instruction Format	607
D.2	Selected Pentium Instructions	609
E	ASCII Character Set	633
	Index	637