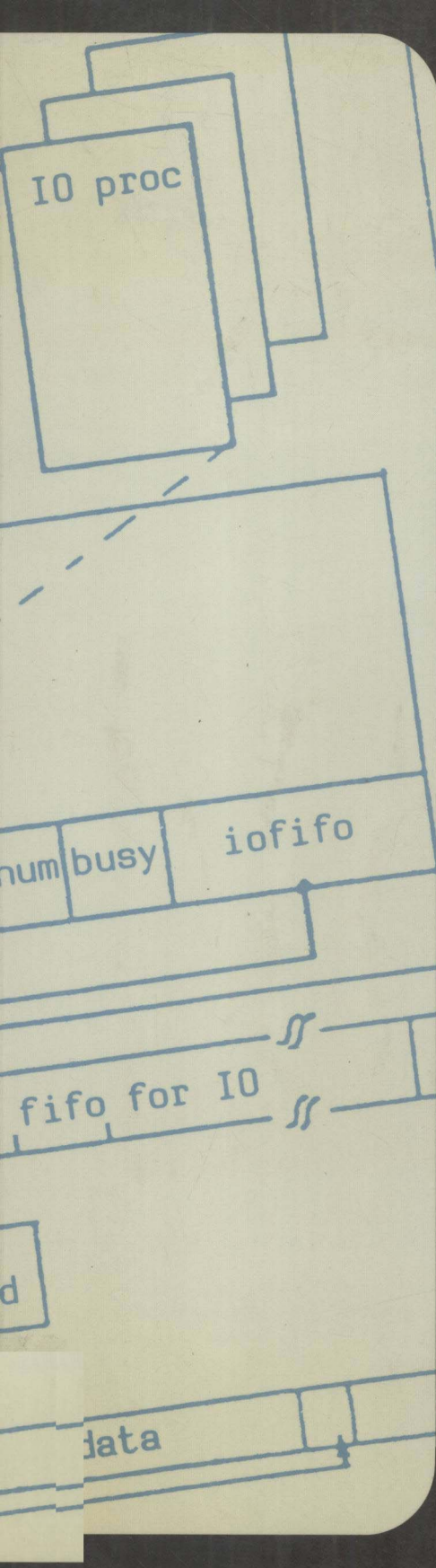




80286 and 80386 Microprocessors New PC architectures

A.B. Fontaine and
F. Barrand

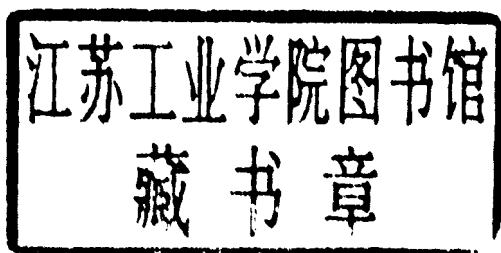
Translated by
A. Rawsthorne

80286 and 80386 Microprocessors

New PC architectures

A. B. Fontaine and F. Barrand

Translated by A. Rawsthorne
Department of Computer Science
University of Manchester



M
MACMILLAN

© Macmillan Education 1989
© Masson, Editeur, Paris, 1987

All rights reserved. No reproduction, copy or transmission of this publication may be made without written permission.

No paragraph of this publication may be reproduced, copied or transmitted save with written permission or in accordance with the provisions of the Copyright Act 1956 (as amended), or under the terms of any licence permitting limited copying issued by the Copyright Licensing Agency, 33-4 Alfred Place, London WC1E 7DP.

Any person who does any unauthorised act in relation to this publication may be liable to criminal prosecution and civil claims for damages.

Authorised English language edition of *Les Microprocesseurs* 80286/80386 by A. B. Fontaine and F. Barrand.

First published 1989

Published by
MACMILLAN EDUCATION LTD
Houndmills, Basingstoke, Hampshire RG21 2XS
and London
Companies and representatives
throughout the world

Printed and bound in Great Britain at
The Camelot Press Ltd, Southampton

ISBN 0-333-49003-7

British Library Cataloguing in Publication Data
Fontaine, A. B.

80286 and 80386 microprocessors: new PC architectures. — (Macmillan computer science series)

1. INTEL 80286 & 80386 microprocessor systems

I. Title II. Barrand, F.
004.165

Macmillan Computer Science Series

Consulting Editor

Professor F. H. Sumner, University of Manchester

- S. T. Allworth and R. N. Zobel, *Introduction to Real-time Software Design, second edition*
Ian O. Angell and Gareth Griffith, *High-resolution Computer Graphics Using FORTRAN 77*
Ian O. Angell and Gareth Griffith, *High-resolution Computer Graphics Using Pascal*
M. Azmoodeh, *Abstract Data Types and Algorithms*
C. Bamford and P. Curran, *Data Structures, Files and Databases*
Philip Barker, *Author Languages for CAL*
A. N. Barrett and A. L. Mackay, *Spatial Structure and the Microcomputer*
R. E. Berry, B. A. E. Meekings and M. D. Soren, *A Book on C, second edition*
G. M. Birtwistle, *Discrete Event Modelling on Simula*
B. G. Blundell and C. N. Daskalakis, *Using and Administering an Apollo Network*
B. G. Blundell, C. N. Daskalakis, N. A. E. Heyes and T. P. Hopkins, *An Introductory Guide to Silvar Lisco and HILO Simulators*
T. B. Boffey, *Graph Theory in Operations Research*
Richard Bornat, *Understanding and Writing Compilers*
Linda E. M. Brackenbury, *Design of VLSI Systems — A Practical Introduction*
G. R. Brookes and A. J. Stewart, *Introduction to occam 2 on the Transputer*
J. K. Buckle, *Software Configuration Management*
W. D. Burnham and A. R. Hall, *Prolog Programming and Applications*
P. C. Capon and P. J. Jinks, *Compiler Engineering Using Pascal*
J. C. Cluley, *Interfacing to Microprocessors*
J. C. Cluley, *Introduction to Low Level Programming for Microprocessors*
Robert Cole, *Computer Communications, second edition*
Derek Coleman, *A Structured Programming Approach to Data*
Andrew J. T. Colin, *Fundamentals of Computer Science*
Andrew J. T. Colin, *Programming and Problem-solving in Algol 68*
S. M. Deen, *Fundamentals of Data Base Systems*
S. M. Deen, *Principles and Practice of Database Systems*
C. Delannoy, *Turbo Pascal Programming*
Tim Denvir, *Introduction to Discrete Mathematics for Software Engineering*
P. M. Dew and K. R. James, *Introduction to Numerical Computation in Pascal*
M. R. M. Dunsmuir and G. J. Davies, *Programming the UNIX System*
D. England et al., *A Sun User's Guide*
A. B. Fontaine and F. Barrand, *80286 and 80386 Microprocessors*
K. C. E. Gee, *Introduction to Local Area Computers Networks*
J. B. Gosling, *Design of Arithmetic Units for Digital Computers*
M. G. Hartley, M. Healey and P. G. Depledge, *Mini and Microcomputer Systems*
Roger Hutton, *Z80 Assembly Language Programming for Students*
Roland N. Ibbett and Nigel P. Topham, *Architecture of High Performance Computers, Volume I*
Roland N. Ibbett and Nigel P. Topham, *Architecture of High Performance Computers, Volume II*

(continued overleaf)

Patrick Jaulent, *The 68000 — Hardware and Software*
 P. Jaulent, L. Baticle and P. Pillot, *68020–30 Microprocessors and their Coprocessors*
 J. M. King and J. P. Pardoe, *Program Design Using JSP — A Practical Introduction*
 E. V. Krishnamurthy, *Introductory Theory of Computer Science*
 V. P. Lane, *Security of Computer Based Information Systems*
 Graham Lee, *From Hardware to Software — an introduction to computers*
 A. M. Lister and R. D. Eager, *Fundamentals of Operating Systems, fourth edition*
 Tom Manns and Michael Coleman, *Software Quality Assurance*
 Brian Meek, *Fortran, PL/I and the Algols*
 A. Mével and T. Guéguen, *Smalltalk-80*
 Y. Nishinuma and R. Espesser, *UNIX — First Contact*
 Pim Oets, *MS-DOS and PC-DOS — A Practical Guide, second edition*
 A. I. Pilavakis, *UNIX Workshop*
 Christian Queinnec, *LISP*
 E. J. Redfern, *Introduction to Pascal for Computational Mathematics*
 Gordon Reece, *Microcomputer Modelling by Finite Differences*
 W. P. Salman, O. Tisserand and B. Toulout, *FORTH*
 L. E. Scales, *Introduction to Non-Linear Optimization*
 Peter S. Sell, *Expert Systems — A Practical Introduction*
 A. G. Sutcliffe, *Human-Computer Interface Design*
 Colin J. Theaker and Graham R. Brookes, *A Practical Course on Operating Systems*
 M. R. Tolhurst *et al.*, *Open Systems Interconnection*
 J-M. Trio, *8086–8088 Architecture and Programming*
 M. J. Usher, *Information Theory for Information Technologists*
 B. S. Walker, *Understanding Microprocessors*
 Peter J. L. Wallis, *Portable Programming*
 Colin Walls, *Programming Dedicated Microprocessors*
 I. R. Wilson and A. M. Addyman, *A Practical Introduction to Pascal — with BS6192, second edition*

Non-series

Roy Anderson, *Management, Information Systems and Computers*
 I. O. Angell, *Advanced Graphics with the IBM Personal Computer*
 J. E. Bingham and G. W. P. Davies, *Planning for Data Communications*
 B. V. Cordingley and D. Chamund, *Advanced BASIC Scientific Subroutines*
 N. Frude, *A Guide to SPSS/PC+*
 Barry Thomas, *A PostScript Cookbook*

Acknowledgements

The translator wishes to thank Mr K Gallup, of Rapid Silicon, for making available facilities to compile the example programs in appendix C.

PC is a trademark of IBM, Unix is a trademark of AT&T, MS/DOS, Xenix, and Windows/386 are trademarks of Microsoft; Macintosh is a trademark of Apple Computer, Inc., Lotus 1-2-3 is a trademark of Lotus Development Corp., and Atari is a trademark of Atari Corp.

Contents

1	Introduction	1
1.1	8086 and 80286 Family Standard Architectures	3
1.2	Operating Systems	4
1.3	Windowing Environments	5
1.4	Networking	6
2	Hardware Structure of the 80286	8
2.1	General Characteristics	8
2.2	Principal Family Components	9
2.2.1	82288 Bus Controller	10
2.2.2	82289 Multibus Arbiter Circuit	11
2.2.3	82284 Clock Generator	11
2.2.4	80287 Numeric Coprocessor	13
2.2.5	Other Circuits	13
2.3	System Architectures	13
2.4	80286 Processor Internal Structure	19
2.5	Bus Interface	22
2.5.1	Physical Memory Organisation	22
2.5.2	Bus Cycle Operation	24
2.5.3	Interrupt Structure	26
2.5.4	Use of the 80287 Numeric Coprocessor	28
2.6	Memory Architecture of the 80286	31
2.6.1	Segmentation	31
2.6.2	80286 Data Types	35
2.6.3	CPU Registers	36
2.6.4	Addressing Modes	39
2.6.5	Introduction to Protection	41

3	The 80286 in Protected Mode	43
3.1	Memory Management	43
3.1.1	Protected Virtual Addressing	44
3.1.2	Descriptors and Descriptor Tables	46
3.1.3	Memory Management Registers	52
3.2	Protection Mechanisms	54
3.2.1	Types of Protection	54
3.2.2	Implementation of Protection	55
3.2.3	Protection and Memory Management	56
3.2.4	Protection and Privilege Levels	56
3.2.5	Conformant Segments	59
3.2.6	Gates	59
3.3	Management of Tasks	63
3.3.1	Task State Segment	63
3.3.2	TSS Descriptor	65
3.3.3	Changing the Context of a Task	66
3.3.4	Using a Task Gate	67
3.3.5	System Access Control	67
3.4	Interrupts	68
3.4.1	Interrupts in Real Mode	68
3.4.2	Interrupts in Protected Virtual Mode	73
3.5	Virtual Memory	80
3.5.1	Strategy for Managing Segments	81
3.5.2	Input/Output Management	83
4	Programming the 80286	85
4.1	80286 Programs	85
4.1.1	Applications Code	85
4.1.2	System Code	85
4.2	Assembly Language	86
4.2.1	SEGMENT Directive - Defining a Segment	88
4.2.2	Definition of Data Items	89
4.2.3	Initialising Segment Registers	90
4.2.4	END Directive	91
4.2.5	Structure of Procedures	91
4.2.6	Structures, Records and Macro Instructions	91
4.3	Modular Programming in Assembly Language	92
4.3.1	Segment Sharing	93

4.3.2	PUBLIC Data and Procedures	94
4.3.3	Invoking the Assembler	95
4.4	Development Tools	96
4.4.1	Builder	96
4.4.2	Binder	100
4.4.3	Mapper	101
4.4.4	LIB286 Librarian	101
4.4.5	OVL286 Overlay Manager	101
4.5	PLM/286	102
4.5.1	PLM/286: A High Level System Language	102
4.5.2	Variables and Constants	103
4.5.3	Program Structure	104
4.5.4	Arithmetic and Logical Operators	104
4.5.5	Control Constructs	106
4.5.6	Subprograms	107
4.5.7	BASED Variables	109
4.5.8	Modular Programming	110
4.5.9	System Programming	111
4.5.10	System Management	112
4.5.11	Compilation Modes	113
4.5.12	Invoking the Compiler	113
5	Operating Systems	115
5.1	Real-time Operating Systems	115
5.1.1	Memory Management	115
5.1.2	Object Management	116
5.1.3	Descriptor Management	116
5.1.4	Exceptions	116
5.1.5	Extensions to Operating System Primitives	116
5.2	XENIX/286: A Development Environment	117
5.3	Operating System Design	117
5.3.1	Tasks	119
5.3.2	Levels of Privilege	120
5.3.3	Structures of Programs	122
5.3.4	Configuring Software Systems	122
5.4	ETR_286: A Real-Time Toolbox	123
5.4.1	Specification	124
5.4.2	Task Scheduling	125

5.4.3	Task Descriptor	126
5.4.4	Timer Manager	127
5.4.5	Message Queue	129
5.4.6	Semaphores	129
5.4.7	Events	130
5.4.8	Memory Allocation	130
5.4.9	Input/Output Management	131
5.4.10	Extensions	133
5.4.11	Exceptions	133
5.4.12	Utilities	134
5.4.13	Creating an Absolute File	134
6	Input/Output and Networking	135
6.1	Physical Management of Input/output	135
6.1.1	Hardware Aspects	135
6.1.2	Software Aspects	136
6.1.3	Local Area Networks	136
6.2	High Level Protocols	139
6.3	Open Net	139
6.3.1	Open Net Under iRMX286	140
6.3.2	Usage Under XENIX/286	142
6.3.3	PC/NETWORK	142
7	The 80386 Microprocessor	143
7.1	General Characteristics of the 80386	143
7.1.1	Software Aspects	143
7.1.2	Hardware aspects	144
7.2	80386 Structure - the Applications View	145
7.2.1	General Registers and Flags	145
7.2.2	Numeric Coprocessors	148
7.2.3	Memory and Addresses	148
7.3	System Aspects of the 80386 Structure	154
7.3.1	System and Task Registers	154
7.3.2	Memory Management	158
7.3.3	Virtual Memory and Protection	161
7.3.4	Changes of Sequence	162
7.4	Cache Memory	162
7.4.1	Definition of a Cache	163
7.4.2	Cache Design Criteria	163
7.4.3	Cache Structures	164
7.5	Using an 80386	167

Appendix A	80286 Instruction Set	168
Appendix B	80386 Instruction Set	191
B.1	Double-word Instructions	191
B.2	Additional Instructions	192
Appendix C	ETR_286 Real-time Monitor	195
C.1	etr.h	196
C.2	etrstr.h	199
C.3	etrprc.h	203
C.4	etrto.pl6	205
C.5	etrtr1.pl6	206
C.6	etrtrf.pl6	207
C.7	etrsched.s86	208
C.8	etrto.pl6	210
C.9	etrford.pl6	213
C.10	etrevms.pl6	217
C.11	etrfifo.pl6	221
C.12	etrmemx.pl6	223
C.13	etrmemu.pl6	224
C.14	etrbud.pl6	225
C.15	etrcnf.pl6	229
C.16	etrdrv.pl6	230
C.17	etrexpl6	235
C.18	etrut.pl6	238
C.19	etrmain.pl6	242
C.20	etrinit.pl6	246
C.21	etres.pl6	249
C.22	etr286.bld	252
Bibliography		256
Index		258

List of Figures

2.1.1 The 80286	8
2.2.1 82288 Bus controller	10
2.2.2 82289 Bus arbiter	11
2.2.3 82284 Clock Generator	12
2.2.4 80287 Pinout	12
2.3.1 Multiprocessor system	14
2.3.2 Local and system buses	15
2.3.3 Multibus wiring	17
2.3.4 Using the 80287 numeric coprocessor	18
2.3.5 A complex system architecture	19
2.4.1 80286 Internal structure	20
2.4.2 Pipelining	21
2.5.1 Memory organisation	23
2.5.2 Transferring a misaligned word	24
2.5.3 80286 Bus read cycle	25
2.5.4 Table of interrupt descriptors	26
2.5.5 Processing simultaneous interrupts	27
2.5.6 Using the 8259A interrupt controller	27
2.5.7 Internal structure of the 80287	29
2.5.8 Numeric coprocessor register set	30
2.6.1 Segmented memory	31
2.6.2 Displacements or offsets	32
2.6.3 Address calculation in real mode	33
2.6.4 Reserved memory in real address mode	34
2.6.5 Format of a segment selector	35
2.6.6 Physical implementation of data	36
2.6.7 General purpose registers	37
2.6.8 Addressing in protected virtual mode	39

2.6.9 Status and control words 40

2.6.10 Addressing modes 41

2.6.11 Segment descriptor format 42

3.1.1 Access to segmented data 43

3.1.2 Segment selector format 45

3.1.3 Descriptor table selection by the TI bit 46

3.1.4 Address spaces and isolation between tasks 47

3.1.5 Segment descriptor 48

3.1.6 System descriptor 48

3.1.7 Segment types 49

3.1.8 Virtual addressing mechanism 50

3.1.9 Translating virtual into physical addresses 51

3.1.10 Non global shared data 52

3.1.11 Segment registers 53

3.2.1 Gate descriptor format 59

3.2.2 Changing privilege levels using a Call Gate 61

3.2.3 Examples of calls by Call Gate 62

3.3.1 Task State Segment 64

3.3.2 Segment structures for two tasks 65

3.3.3 TSS descriptor 66

3.3.4 Task Gate 67

3.3.5 Task switch using a Task Gate 68

3.4.1 8259A Interrupt controller causing INTR 69

3.4.2 Interrupt acknowledge cycle 70

3.4.3 Sources of interrupts 71

3.4.4 Stack following an interrupt 72

3.4.5 Interrupt Descriptor Table 74

3.4.6 IDTR and other CPU Task Registers 75

3.4.7 Accessing the interrupt descriptor 75

3.4.8 Interrupt and Trap Gate descriptor format 77

3.4.9 Stack following an error exception 78

3.4.10 Relationship between two tasks 80

3.5.1 Virtual memory 82

4.2.1 A simple program 88

4.3.1 Small model 93

4.3.2 Large model 93

4.4.1	Using Builder	97
4.4.2	Using Binder	100
4.5.1	Block structure in a PLM/286 program	105
4.5.2	General layout of a subprogram	108
4.5.3	BASED variables	110
4.5.4	Accessing variables and procedures from other modules	111
5.1.1	Adding iRMX286 primitives	117
5.2.1	Xenix/286 organisation	118
5.3.1	Separation of data between tasks	120
5.3.2	Sharing information between tasks	121
5.3.3	System with four privilege levels	122
5.3.4	Constructing a static system	123
5.3.5	Constructing a dynamic system	124
5.4.1	Principal ETR_286 data structures	127
5.4.2	Data structures for timeouts	129
5.4.3	Message queue	130
5.4.4	ETR_286 Input/Output structures	133
6.1.1	Network subsystem using 82588	137
6.1.2	Data structures used by the 82588	138
6.2.1	OSI and CCITT protocols	140
6.3.1	iNA960 under iRMX286	141
6.3.2	Open Net file systems	142
7.2.1	General registers	146
7.2.2	80386 flags register	147
7.2.3	Translating logical addresses	150
7.2.4	Segment registers	151
7.2.5	Address mode summary	152
7.3.1	Descriptor fields	154
7.3.2	System segment descriptors	155
7.3.3	System segment descriptor registers	156
7.3.4	Principal TSS fields	157
7.3.5	Control registers	158
7.3.6	Address translation with paging	159
7.3.7	Paging mechanism	160
7.4.1	Direct-mapped 64 kilobyte cache	164
7.4.2	Direct-mapped cache hardware	165
7.4.3	Two-way set-associative cache	166

1 *Introduction*

The importance of sixteen-bit microprocessors has grown very significantly in the ten years since their introduction, and today only a small niche remains for the eight-bit processors. Apart from the Zilog Z80, which seems to go on and on, and excepting special-purpose sequencers and microcontrollers, the microcomputer market is dominated by 16- and 32-bit architectures. For a number of years now, the systems architect has had a number of suppliers at his disposal, and has been able to choose between numerous architectures; most current systems have, however, been based on one of two microcomputer families.

The two most significant families of microprocessor components in common usage today are the Intel 8086 and the Motorola 68000 architectures, although a number of less popular alternatives exist. The competition between these families is intense, and is based on the related bus and system architectures as well as on the capabilities of the semiconductor components themselves. Intel have developed the Multibus I and II architectures, while Motorola promote the VME bus, although it is possible to mix processors and bus architectures. Two significant influences on the adoption of these two families were the use of the Intel processor in the IBM PC and the appearance of the Motorola chip in the Apple Macintosh.

The financial rewards from a successful architecture are vast, since the investment in software for one architecture must be preserved, so semiconductor manufacturers need to hold on to their customers. One source inside Intel estimates that to date U.S. \$6 billion have been spent on writing 8086 software!

In this book, we have chosen to describe in detail the iAPX 286, a successor to the 8086, with significant enhancements to help its users exploit its power. We also describe the significant enhancements brought to the Intel family by the introduction of the 32-bit iAPX 386.

The speed of progress in this area can be judged from a few figures: the 8086 processor, conceived in 1976 and introduced in 1978, has a raw performance of about 0.2 MIPS (million instructions per second) at 8 MHz. It can address 1 megabyte of memory. At the time of its introduction, this processor offered significant advantages over then-current 8-bit devices. Today, however, it has been largely superseded, for a number of reasons.

The first reason is the size of computer programs. Software sometimes seems

like an ideal gas filling all available memory. For example, the IBM PC, with a maximum of 640 kilobytes, is too small for a number of applications. This restriction has encouraged Intel and Lotus (of 1-2-3 spreadsheet fame) to develop a standard means of extending the 8086 address space to 8 megabytes, at the cost of some complexity in programs that need to manage this space explicitly.

The second reason is that of speed. Simple computer applications need only a limited computer power. Evolving applications tend to be much more sophisticated, with an optimised man-machine interface, with intensive use of graphics, in multiple windows, and extensive use of network communications. It is just not possible to provide these facilities on a less powerful machine. (The reader may test this assertion by comparing the usability of the MS-Windows system on a PC/XT and on a PC/AT.)

The third reason concerns the needs of protection. In order to use a multitasking or multi-user operating system, it is necessary to provide protection between processes. Protection facilities were introduced to the Intel family in the 80286 processor.

These considerations all played their part in Intel's development of the 80286, which also, of course, represents their attempt to keep a dominant position in the microprocessor market place.

The 80286 can be seen simply as an enhancement of the 8086. In real mode, or compatibility mode, it behaves as an 8086, but 4 or 5 times faster. It approaches a performance of 1 MIP, about the raw performance of a DEC VAX 11/780, for about \$50. Used in this way, it offers a significant performance with no software modifications.

The two other problems mentioned above are addressed when the chip is used in protected mode, and need operating system modifications to exploit them. In this mode, the processor offers the same raw performance, but the real addressing capacity is increased to 16 megabytes, and an individual process can address a virtual space of 1 gigabyte. In this mode, the processor supports protection, which permits applications to run without corrupting others and, by catching erroneous accesses, eases debugging during program development. So far as compatibility is concerned, investment in existing 8086 software is preserved in that while applications needing to exploit new features will need rewriting, most current applications need very few, if any, changes to run on the 286. Even assembly-language programs can be ported to the 286 with minimal changes. Only an operating system that wishes to use the protected mode will need significant changes when ported from the 8086 to the 80286.

The story continues from here, since the 80286 now has a successor, the 80386, which retains the protected mode of the 80286, but extends its power: it can address 4 gigabytes of real memory, and has a raw performance of between 3 and 4 MIPS. A number of microcomputer manufacturers (including IBM) have introduced products based upon this processor: one of the first to be introduced was a portable computer by Compaq, offering a performance of about 3 MIPS, and

supporting 140 megabytes of disk, a streaming tape drive, and up to 6 megabytes of main memory. This machine showed a speed of about 3 times an IBM PC/AT, and was first offered in 1986 at a price of about U.S. \$8000.

Software for the 80386 is also available now: the multitasking OS/2 is available, as are versions of Xenix and Unix System V. Some suppliers are offering concurrent UNIX and MS/DOS facilities, and one can predict the availability of “hypervisors” in the style of the VM/370 system for IBM mainframes.

1.1 8086 and 80286 Family Standard Architectures

The single most significant difference between the 8086 and the Motorola 68000 families is Intel’s segmented memory addressing scheme. In order to address larger memory spaces, an address of 20 or more bits is necessary. If used everywhere, these addresses would lengthen the instructions in a program and slow its execution speed. Both architectures, therefore, allow the use of base registers, which can be loaded once with a full address, and thereafter permit data to be accessed using a short 8- or 16-bit displacement. The 68000 provides both of these and an absolute address form.

Intel, however, do not encourage the use of absolute addresses, and require the programmer to specify a Base Register and a displacement for every memory access. They call the base register a “segment register”, and describe all memory accessible from that base as a “segment”.

Segment sizes are limited, to 64 kilobytes in the case of the 8086 and the 80286 and to 4 megabytes in the 80386. This can be a significant constraint for an assembly-language programmer, but should be handled automatically by a high-level language compiler.

Segmentation can have advantages for the construction of large software applications, in that individual segments can be characterised by their access requirements: the 80286, for instance, monitors each access to a segment. If it is marked “read-only”, no software can write to it: if it is not marked “executable” it is impossible to execute code from it. Forbidden accesses are trapped by the processor, enhancing the security of the software that is running.

A related topic to segmentation is providing virtual memory by “paging”. This scheme allows an operating system to use memory and processor time efficiently by allocating memory in fixed-size “pages”, which are usually between 1 and 4 kilobytes in size. Address translation hardware is needed to permit programs and their data to be loaded into non-contiguous real memory. Segments, however, are of variable size and may be much larger than an optimum page size. The choice of providing either segmentation or paging in an architecture is resolved in the 80386 by providing both.