icrocontrollers

HD44795 MC68HC11 MCS-51 MCS-96 80960CA Architecture, Implementation, & Programming

KENNETH HINTZ DANIEL TABAK

Microcontrollers

Architecture, Implementation, and Programming

Kenneth J. Hintz **Daniel Tabak**





McGraw-Hill, Inc.

New York St. Louis San Francisco Auckland Bogotá Caracas Lisbon London Madrid Mexico Milan Montreal New Delhi Paris San Juan São Paulo Singapore Sydney Tokyo Toronto

Library of Congress Cataloging-in-Publication Data

Hintz, Kenneth.

Microcontrollers: architecture, implementation, and programming / Kenneth Hintz,

Daniel Tabak p. cm.

Includes bibliographical references and index.

ISBN 0-07-028977-8:

1. Programmable controllers. 2. Microprocessors. I. Tabak,

Daniel, date. II. Title.

TJ223.P76H55 1992 629.8'9516—dc20

91-37333

CIP

Copyright © 1992 by McGraw-Hill, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

1 2 3 4 5 6 7 8 9 0 DOC/DOC 9 7 6 5 4 3 2 1

ISBN 0-07-028977-8

The sponsoring editor for this book was Daniel A. Gonneau, the editing supervisor was Fred Dahl, and the production supervisor was Suzanne W. Babeuf. It was set in Century Schoolbook by Inkwell Publishing Services.

Printed and bound by R. R. Donnelley & Sons Company.

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The author and publisher have exercised care in preparing this book and the programs contained in it. They make no representation, however, that the programs are error-free or suitable for every application to which a reader may attempt to apply them. The author and publisher make no warranty of any kind, expressed or implied, including the warranties of merchantability or fitness for a particular purpose, with regard to these programs or the documentation or theory contained in this book, all of which are provided "as is." The author and publisher shall not be liable for damages in an amount greater than the purchase price of this book, or in any event for incidental or consequential damages in connection with, or arising out of the furnishing, performance, or use of these programs or the associated descriptions or discussions.

Readers should test any program on their own systems and compare results with those presented in this book. They should then construct their own test programs to verify that they fully understand the requisite calling conventions and data formats for each of the programs. Then they should test the specific application thoroughly.

Microcontrollers

Other Books of Interest

ANTOGNETTI • Semiconductor Device Modeling with SPICE

CLEMENTS • 68000 Sourcebook

DEWAR, SMOSNA . Microprocessors

FORSYTHE, GOODALL . Digital Control

DI GIACOMO • VLSI Handbook

DI GIACOMO · Digital Bus Handbook

ELLIOTT • Integrated Circuit Fabrication Technology

PERRY • VHDL

SHERMAN • CD-ROM Handbook

SIEGEL • Interconnection Networks for Large-Scale Parallel Processing

TABAK · Advanced Microprocessors

TRONTELJ, TRONTELJ, SHENTON • Analog/Digital ASIC Design

TSUI • LSI/VLSI Testability Design

VAN ZANT . Microchip Fabrication

VIRK • Digital Computer Control Systems

Preface

Microcontrollers are digital computers particularly designed to supervise. manage, monitor, and control various processes in industry, business, defense, aerospace, and other areas of application. With the advent of VLSI technology, microcontrollers are becoming essentially single-chip microcomputers. As the technology advances, microcontroller chips attain higher density, and more and more equivalent transistors can be placed on a single chip—over a million at the beginning of the nineties and over 50 million around 2000, as promised by some chip manufacturers. A microcontroller chip, being a self-sustained microcomputer, contains in addition to a central processing unit (CPU), cache, main memory (more memory is placed on a chip as technology develops), input/output (I/O) interfaces, direct memory access (DMA) controllers, interrupt handlers, timers, and other subsystems necessary to implement an efficient microcontroller.

Microcontrollers, being a particular case of digital computers, have to be programmed so that they can perform their assigned tasks. In the control of relatively simple processes, such as a single traffic intersection, one can probably get by with a relatively short and simple assembly language program. However, many processes are very complicated and require lengthy and sophisticated software. In such cases, assembly language programming is too time-consuming and impractical. Thus, high level language (HLL) programming (such as in C, ADA, or PASCAL) becomes an indispensable necessity. Naturally, one has to develop efficient approaches for generating useful and reliable microcontroller software, be it in assembly, HLL, or a combination of the above.

The goal of this book is to present to the reader a unified document containing a variety of aspects involved in the design and implementation of microcontrollers. The book covers microcontroller architecture, organization, structure, and assembly and HLL (with a stress on the C language) programming. In addition, the text presents a number of examples of actual microcontrollers of different capabilities, currently available on the market. Design examples based on existing commercial microcontrollers are also included.

The book is not intended for beginners in computers. It is assumed that the reader has had a basic course in computer organization and in programming of at least one assembly and one HLL computer programming language. The book can be used as a reference text by practicing engineers, computer scientists, and other professionals involved in the design and implementation of microcontrollers. It can also be used as a textbook in special senior level or first-year graduate courses on microcontrollers. Such a course (ECE 447) is indeed being taught at the authors' school, the George Mason University (GMU). It is a senior course, and a significant part of the material in the text was successfully presented in it. Parts of the book were also taught in other GMU courses, such as a senior course on computer design and a graduate course on advanced microprocessors. Since microcontrollers are applied in practically all industrial areas, the text can be used in any engineering curriculum. It can also be used by computer science students interested in practical applications.

The book is subdivided into two major parts. The first part (Chapters 1 to 4) presents a unified, generic coverage of the principles of microcontrollers, covering their architecture, organization, and software. The second part (Chapters 5 to 7) presents a selection of modern leading microcontrollers, including practical design examples involving some of them. The chapters covering the examples are subdivided according to the basic microcontroller word (or data bus) size: Chapter 5 deals with 4- and 8-bit microcontrollers, Chapter 6 with 16-bit, and Chapter 7 with 32-bit systems. Concluding comments are given in Chapter 8. Although every attempt has been made to include the most current manufacturers data, current detailed design information should be obtained from the individual manufacturers before designing a production microcontroller system.

Some of the examples of actual systems were reviewed by professionals within the particular manufacturing company. The authors would particularly like to thank Mr. Lindsay Wallace of Intel, Mr. Rob Tobias of LSI Logic, Mr. Mike Johnson of AMD (Advanced Micro Devices), Mr. Edward Goldberg of NEC Electronics, Inc., and Mr. Reuven Marko of National Semiconductor for their valuable comments and material made available to the authors. The authors would like to express particular appreciation to Prof. Jack Lipovski (University of Texas, Austin) for reviewing the text for McGraw-Hill and for many helpful suggestions. The authors acknowledge with thanks the editorship of Mr. Daniel Gonneau of McGraw-Hill. This work was partially supported by the Virginia Center for Innovative Technology (CIT) through the Center of Excellence in Command, Control, Communications and Intelligence (C³I) at George Mason University. Last but not least, the authors would like to express their appreciation to their wives Sue Hintz and Pnina Tabak for their patience and moral support.

Abbreviations

\$xxxx:	Hexadecimal number	HSI:	High-speed input
%xxxx:	Binary number	HSIO:	High-speed I/O
A/D:	Analog-to-digital	HSO:	High-speed output
ALU:	Arithmetic logic unit	Hz:	Hertz, cycles/second
AMD:	Advanced Micro Devices	ICE:	In-circuit emulator
BIU:	Bus interface unit	ICU:	Interrupt control unit
CC:	Condition code	IMB:	Intermodule bus
CFG:	Configuration register	I/O:	Input/output
CFM:	Control flow machine	IOP:	I/O processor
CISC:	Complex instruction set	IR:	Instruction register
	computer	IRQ:	Interrupt request
CMOS:	Complementary metal oxide	ISR:	Interrupt service routine
	silicon	k:	1000
COP:	Computer operating properly,	K:	1024
	control-oriented processor	LCD:	Liquid crystal dispaly
CPN:	Colored Petri net	LED:	Light emitting diode
CPU:	Central processing unit	LRC:	Local register cache
CSG:	Control signal generator	LRU:	Least recently used
DFM:	Data flow machine	LSB:	Least significant bit
D/A:	Digital-to-analog	MAR:	Memory address register
DMA:	Direct memory access	MCU:	Microcontroller unit,
DOS:	Disc operating system		single-chip microcontroller
DRAM:	Dynamic RAM	MDR:	Memory data register
DSP:	Digital signal processing	MIPS:	Microprocessor without
EBCDIC:	Extended Binary Coded		interlocked pipeline stages
	Decimal Interchange Code	MMU:	Memory management unit
EBI:	External bus interface	MODEM:	Modulator/demodulator
EOC:	End-of-conversion (A/D)	MSB:	Most significant bit
FAM: FAX:	Facsimile accelerator module Facsimile	N	Petri net with no specific initial marking
FIFO:	First-in, first-out	(N,M_0) :	Petri net with a specific
FIP:	Fluorescent indicator panel	(14,1110).	initial marking
FIR:	Finite impulse response (filter)	NC:	Normally closed (switch)
FP:	Frame pointer	NMI:	Nonmaskable interrupt
FPAL:	Floating-point arithmetic	NO:	Normally open (switch)
	library	NRZ:	Nonreturn to zero (signaling)
FPU:	Floating-point unit	OEM:	Original equipment
FSR:	Floating-point status register		manufacturer
HCMOS:	High-density CMOS	Op-code:	Operation code
HLL:	High-level language	OS:	Operating system
	-		

xii Abbreviations

PC:	Personal computer, program	ROM:	Read only memory
	counter (register)	RPM:	Revolutions per minute
PC:	Programmable controller	RTOP:	Real time output ports
PCB:	Process control block	SA:	Successive approximation
PFP:	Previous frame pointer		(A/D)
PGA:	Pin grid array	SCI:	Serial communications
PLA:	Programmable logic array		interface
PLL:	Phase-lock loop	SFR:	Special function register
PMU:	Peripheral management units	S/H:	Sample and hold
PN:	Petri net	SIM:	System integration module
PW:	Pulse width	SIO:	Serial input/output
PWM:	Pulse width modulation	SP:	Stack pointer
PSR:	Processor status register	SPARC:	Scalable processor
PSW:	Program status word		architecture
PTS:	Peripheral transaction server	SPI:	Serial peripheral interface
QSM:	Queued serial module	SRAM:	Static RAM
RALU:	Register ALU	TDM:	Time division multiplexing
RAM:	Random access memory	TLB:	Translation lookaside buffer
	(read/write memory)	TPU:	Time processing unit
RIP:	Return instruction pointer	UART:	Universal asynchronous
RISC:	Reduced instruction set		receiver transmitter
100000	computer	WSR:	Window select register

Contents

Preface ix Abbreviations xi

		Computer, Microprocessor,	
and M	icroco	introller Architectures	1
1.1	The E	ssential Elements of a Computer	4
*	1.1.1	이 마을에 가는 것 같은 그렇게 가는 사람들이 되었다. 그렇게 되었다. 그렇게 되었다면 하는 사람들이 되었다면 하는 것이 없는 것이 없는 것이 없는 것이 없다면 하는 것이 없다.	5
	1.1.2	나이 어린 아이들의 이렇게 하는 것이 맛있다면 하는데 하는데 아이들이 되었다. 그렇게 되었다면 하는데 하는데 하는데 하는데 되었다.	8
	1.1.3	이 마음하게 하면 살아보고 하다면 하는데 하면 하는데	13
	1.1.4	The Control Unit Element	16
1.2	Micro	processor: Most of a Computer on a Chip	27
	1.2.1	맛있다면서 어린 생각을 하다면 보면 가게 되었다면서 하는 사람들이 되었다. 그는 사람들이 하는 사람들이 하는 사람들이 하는 사람들이 하는 사람들이 되었다.	27
	1.2.2	강에는 (24대명) 등에 이렇게 하면 계획하면 보고 됐었다면 되었다. 그리고 있다면 보고 말라는 말라는 말라고 있다면 하는데 그리고 말라고 있다. (1) 그리고 있다는데 그리고 있다. 나는 나는	29
1.3	Single	e-Chip Microcomputers	29
1.4		controller: I/O-Oriented Single-Chip Microcomputer	31
	1.4.1	병생해야 하게 되었다면 하는 그들은 그는 이름을 하는 것이 없는 그들은 그래요? 하게 하게 하게 하는 것이 되었다면 하는 것이 되었다면 하는 것이 없어 없었다.	32
	1.4.2	Interrupts	32
	1.4.3	ALU	34
	1.4.4	Timers	34
	1.4.5	Parallel and Serial I/O	35
	1.4.6	External Devices	35
	1.4.7	Configurations of Microprocessors and Microcontrollers	40
Chapt	er 2. (Computer, Microprocessor, and Microcontroller	
Instru			43
2.1	Comp	outer Instruction Sets	43
	2.1.1	Desirable Characteristics of Instruction Sets	45
	2.1.2	Instruction Formats	45
	2.1.3	Addressing Modes	48
	2.1.4	SISC, RISC, and CISC	49
2.2	Task-	Oriented Instructions	51
	2.2.1	Instructions for Business, Text Processing,	
		and Data Manipulation	51

	2.2.2 Science-Oriented Instructions	52
	2.2.3 Control-Oriented Instructions	52
2.3	MCU Instruction Sets	53
	2.3.1 Capability Determined by Word Size	53
	2.3.2 A Comparison of Four MCU Instruction Sets	60
	2.3.3 I/O Instructions	61
	2.3.4 Arithmetic Instructions	65
	2.3.5 Bit Manipulation Instructions	65
	2.3.6 Program Flow Control Instructions	66
2.4	Programmable Controller Instructions	68
	2.4.1 Relay Logic	69
	2.4.2 Arithmetic, Data Manipulation, Data Transfer, and Program Control	71
2.5	Summary	71
Chante	er 3. Controller Software Design	77
•		79
3.1	Finite State Machine Model 3.1.1 The State Transition Function	80
	3.1.2 The Output Function	81
	3.1.3 FSM Table or FSM Diagram	81
	3.1.4 Control Flow and the FSM	84
	3.1.5 A Limitation of the FSM Model	86
3.2		92
5.2	3.2.1 Formal Definition of a Petri Net	93
	3.2.2 An FSM Coin Counter in Petri Net Notation	95
	3.2.3 The Coin Counter as a Colored Petri Net	96
	3.2.4 Interrupts, Flags, and Semaphores as Tokens	96
	3.2.5 Petri Tables as a Software Design Tool	98
3.3	Integrated Software Design Model	103
	3.3.1 Table Summarizing Design Steps for Petri Tables	104
	3.3.2 Example Problem: UAV Controller/Autopilot (UAV CAP)	106
	3.3.3 Petri Table for UAV Controller/Autopilot (UAV CAP)	109
	3.3.4 Example Problem: Infrared Sensor Target Tracker	115
3.4.	Summary	121
	and a service of the consensus part of the consensus of t	
Chapt	er 4. Microcontroller Software Implementation	127
4.1	Software Development Process	127
	4.1.1 Software Development	128
	4.1.2 Include and Header Files	132
	4.1.3 HLL/Assembly Language Program Development	136
4.2		137
	4.2.1 High-Level Languages	140
	4.2.2 Macro Expansion and Functions	143
	4.2.3 Assembly Language Programming	146
	4.2.4 C Language Programming	147

	Solitorius -	
	4.2.5 Comparison of C and Assembly Language	153
4.3	Conversion from Petri Table to Software	156
	4.3.1 UAV Petri Table Implementation in C	156
	4.3.2 Magnetometer Assembly Language Example	158
4.4	Interfacing C and Assembly Language	159
	4.4.1 Example Linker Command File	163
	4.4.2 Smart Compass Example of C and Assembly Language Linking	165
	4.4.3 Interrupt Handling in the IR Tracker	165
	4.4.4 Miscellaneous C/Assembly Language Interactions	167
4.5	Summary	169
Chapt	er 5. 4-bit and 8-bit Microcontrollers	173
5.1	4-bit Microcontrollers	174
5.2	Texas Instruments TMS1000 Family Members	175
	5.2.1 TMS1000 Family Members	175
	5.2.2 TMS1000 Architecture	176
	5.2.3 Design and Application Tools	182
5.3	NEC μPD7500 Family of 4-Bit MCUs	182
	5.3.1 μPD7500 Family Members	183
	5.3.2 μPD7500 Architecture	184
	5.3.3 Design and Application Tools	191
	5.3.4 μPD75 Family Members	192
	5.3.5 μPD75x Architecture	192
5.4	National Semiconductor COP400	198
	5.4.1 National Semiconductor COP400 Family Members	198
	5.4.2 National Semiconductor COP400 Architecture	198
	5.4.3 Design and Application Tools	208
5.5	Other 4-Bit MCUs	208
5.6	8-Bit Microcontrollers	208
5.7	Motorola M6801 Family	209
	5.7.1 Motorola M6801 Family Members	209
	5.7.2 M6801 Architecture	210
	5.7.3 Design and Application Tools for the M6801	231
5.8	Motorola M6805 Family	231
	5.8.1 M6805 Family Members	231
	5.8.2 M6805 Architecture	232
5.9	Motorola MC68HC11	235
	5.9.1 Motorola MC68HC11 Family Members	237
	5.9.2 Architecture	237
5.10	Intel MCS-51 Family	249
	5.10.1 Intel MCS-51 Family Members	249
	5.10.2 MCS-51 Architecture	251
5.11	Texas Instruments TMS370 Family	257
	5.11.1 TMS370 Architecture	257
5.12	Summary	262

viii Contents

Chapter 6. 16-bit Microcontrollers	265
6.1 Intel MCS-96 Microcontroller Family 6.2 Motorola MC68332 Microcontroller Appendix A: Design Example	265 288 297
Chapter 7. 32-Bit Microcontrollers	303
 7.1 Intel 80960CA Superscalar Embedded Processor 7.1.1 Superscalar Organization 7.1.2 80960CA Architecture 7.1.3 The 80960 Family and Applications 7.2 LSI Logic LR33000 Embedded Processor 7.3 AMD 29050 Embedded Processor 7.4 National Semiconductor Embedded Processors 7.5 Comparison and Evaluation of 32-bit Microcontrollers Appendix A: 80960CA Pin Description Appendix B: Design Examples 	303 305 313 336 353 372 385 400 409
Chapter 8. Concluding Comments	453
Appendices	455
Appendix I. Smart Compass	455
Appendix II. UAV Code	464
Appendix III. Linker	468
Appendix IV. IRTrack	470

Index 477

Computer, Microprocessor and Microcontroller Architectures

There is a great distinction between computer architecture and organization; the following discussion focuses primarily on computer architecture. Computer architecture is a description (definition) of the attributes of a computing system as seen by a machine language programmer or a compiler writer. Writable control stores for modifying microcode during computer operation are not considered available to the normal machine language programmer. Computer organization pertains to the various methods that can be used to implement a specific computer architecture. The demarcation between these two components of microcontroller implementation is becoming less distinct with the advent of reconfigurable designs. This will become more evident as this book progresses from the earliest to the most recent designs.

The categorization of computers was once easy because there was a clear difference in cost, amount of memory, type of peripherals, and physical operating speed. While there are still supercomputers, specialized array and pipelined processors, and expensive multiuser computers, the computational capabilities of the most powerful machines of a few years ago have trickled down to the level of personal computers (PCs). An inspection of computer specifications shows that the capabilities of computers in terms of complex arithmetic operations, memory addressing capabilities (virtual address space), as well as amount of connected memory (physical address

TABLE 1.1 MATLAB Benchmarks of Various Computers Showing the Favorable Performance of Microprocessor-Based Machines Relative to Larger Computers

Computer ·	Figure of Merit Relative to PC/XT @ 4.7 MHZ	KFlop/Second	
MAC (8 MHz 68000)	0.2233	3	
PC/AT (6 MHz/80286/EGA)	1.3570	15	
PC/XT (4.7 MHz/8088/CGA)	1.0000	17	
AT&T 6300 (8 MHz/8086)	1.8760	29	
Mac II (68020/68881)	6.2358	85	
Apollo DN3000 (16 MHz)	5.4287	72	
Sun-3/50 (15 MHz with 68881)	5.1628	89	
Apollo DN4000 (25 MHz)	9.0300	140	
MicroVAX II (VMS/D_floating)	4.1810	140	
Mac IIcx (68030/68882)	9.6776	168	
803886/80387 (20 MHz, 386-MATLAB)	14.3466	232	
Sun-386i (25 MHz)	10.7024	198	
VAXStation 3100 (VMS/D_floating)	18.9047	365	
Sun-3/260 (25 MHz with FPA)	19.4267	490	
Sun-4/110	35.4397	730	
Sun SparcStation	65.5196	1196	
Ardent Titan	61.6884	3614	

space), have inexorably migrated from the larger to the smaller machines. Even PCs that operate in a single-user mode have comparable computational capabilities to those of much larger machines as is shown by the comparison of MATLAB benchmarks in Table 1.1. The seven standard benchmarks that were run are

- 1. N = 50 real matrix multiply
- 2. N = 50 real matrix inverse
- 3. N = 25 real eigenvalues
- 4. 4096-point complex FFT
- 5. LINPACK benchmark
- 6. 1000 iteration FOR loop
- 7. N = 25 3-D mesh plot

The geometric mean of the execution times was computed to determine a figure of merit relative to a PC/XT machine. Table 1.1 also lists the KFlop/s throughputs of the various machines. Initially microcontrollers were thought to be simpler implementations of computers with a limited instruction set and enhanced I/O capabilities that could be mass-produced inexpensively. With the advent of increasingly complex microcontrollers, including single-chip microcontrollers with complex computational capabilities and 32-bit architectures, microcontrollers have taken on a new set of problems such as digital signal processing, computer peripheral control, and real-time detection and modulation/demodulation of FAX signals as well as complex non-linear control problems. One can no longer look at just simple parameters such as number of instructions, quantity of virtual and physical memories, and processing speed to separate supercomputers from microcontrollers. The functional use of the computer determines whether it is a microcontroller, computer, or supercomputer. It is not too difficult to conceive of a massively parallel computer that uses all of its computational resources to maintain tracks of aircraft and direct the air traffic control (ATC) in a specific geographic area. There is a lot of computation performed in correlating radar returns, merging and splitting tracks, and predicting trajectories of aircraft for collision avoidance and routing. However, the computations are incidental to the primary function of the computer, which is air traffic control. There is, however, a clear distinction on closer inspection, when one studies the capabilities of an instruction set relative to a particular problem. Continuing with this example, if there are included in the instruction set single instruction operations for performing the discrete Fourier transform (DFT) as well as matrix operations, the computations necessary for target tracking can be performed more efficiently. If there are many I/O instructions that can interact directly with the contents of memory, then the computer can be interfaced more easily with radar inputs and control outputs. If there is a complex set of graphical primitive operators, then the computer can more effectively display the information to the human operators. When it comes to control, the application, more than the descriptive title of the computer or integrated circuit, must be used in selecting the correct hardware. However, microcontrollers will be considered to have all of the elements of a computer on a single integrated circuit (IC) and be able to operate with few or no external components.

Control and microcontrollers mean different things to different people. The ATC system is an example of control, but so is the control of motors in industrial settings, the control of microwave ovens, the control of videocassette recorders (VCRs), the control of drug administration to hospital patients, the replacement or bypassing of human nervous systems that have been damaged, the control of bank (automatic) teller machines (ATMs), the control of remotely piloted or autonomous vehicles for operations in hazardous environments and the control of automotive engines for

4 Chapter One

pollution reduction and improved efficiency. It is not possible for one type of controller to meet all of the diverse requirements of these various applications; this prompts the impartial study of the various architectures and organizations of microcontrollers. No single computer or microcontroller is the best for all applications, and only a thorough understanding of the alternative implementations and their impact on design will allow one to select the correct hardware and software configuration for a particular application. This is the source of our egalitarian approach to computers, which implies that there is no hierarchy among computers, only the correct computer as determined by the particular application. There is no hierarchy between software and hardware, only the right application of each capability to solve the task at hand.

1.1 THE ESSENTIAL ELEMENTS OF A COMPUTER

There are five essential elements of a computer: the arithmetic logic unit (ALU), the input section, the output section, the control unit (CU), and the memory. These are shown in the block diagram of Fig. 1.1. The input and output sections are commonly referred to by their collective name of input/output (I/O), and depending on the convenience of the situation, they may be treated collectively or individually. All of these elements are necessary to form a complete computer. If there is no input, the computer cannot respond to its environment. If there is no output, the computer cannot effect changes in its environment. If there is no ALU, the computer cannot perform alterations of its input but only move it from one storage location to another and/or pass it from inputs to outputs. If there is no memory, the system is no more than a finite-state machine. And while most microcontroller applications can be represented by and behave as finite-state machine as well as compute any computable number. For an expansion of

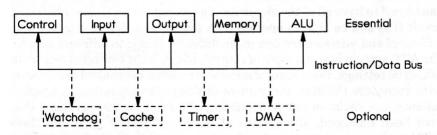


Figure 1.1 Required and typical optional elements of a single-chip computer.