Th. Flik · H. Liebig

16-Bit-Microprocessor Systems

Structure, Behavior, and Programming

Translated by G. Bisiani

With 188 Figures and 27 Tables

Springer-Verlag
Berlin Heidelberg New York Tokyo

Dr.-Ing. Thomas Flik
Prof. Dr.-Ing. Hans Liebig

Institut für Technische Informatik,
Technische Universität Berlin
Franklinstrasse 28/29, 1000 Berlin 10, FRG

Gigliola Bisiani

917 Bellefonte Ave., Pittsburgh, PA 15232, USA

ISBN-13: 978-3-540-15164-7 e-ISBN-13: 978-3-642-93285-4
DOI: 10.1007/978-3-642-93285-4

Library of Congress Cataloging in Publication Data
Flik, Th., 1943-. 16-bit microprocessor systems. Translation of: 16-Bit-Mikroprozessor-systeme. Bibliography: p. Includes index. 1. Microprocessors. I. Liebig, Hans. II. Title. III. Title: Sixteen-bit microprocessor systems. QA76.5.F44613 1985 001.64 85-8056

Foreword

In the last few years, a large number of books on microprocessors have appeared on the market. Most of them originated in the context of the 4-bit and the 8-bit microprocessors and their comparatively simple structure. However, the techno-logical development from 8-bit to 16-bit microprossors led to processor components with a substantially more complex structure and with an expanded functionality and also to an increase in the system architecture's complexity.

This books takes this advancement into account. It examines 16-bit micro-processor systems and describes their structure, their behavior and their programming. The principles of computer organization are treated at the component level. This is done by means of a detailed examination of the characteristic functionality of microprocessors. Furthermore the interactions between hardware and software, that are typical of microprocessor technology, are introduced. Interfacing techniques are one of the focal points of these considerations.

This puplication is organized as a textbook and is intended as a self-teaching course on 16-bit microprocessors for students of computer science and communications, design engineers and users in a wide variety of technical and scientific fields. Basic knowledge of boolean algebra is assumed. The choice of material is based on the 16-bit microprocessors that are currently available on the market; on the other hand, the presentation is not bound to any one of these microprocessors.

The first chapter introduces the behavior, the basic structure and the assembly language programming of a simple 16-bit microprocessor system. The reader that is already familiar with the basics of computer organization and assembly language programming can skip this chapter. - The second chapter deals only with those hardware characteristics of a 16-bit microprocessor that are needed for programming and lays the foundation for the third chapter, where several important programming techniques are described. - The fourth chapter focuses on the problems of the system structure, in particular on the flow of signals between the miroprocessor and the various system components. - Chapters 5 and 6 tackle the most common interfacing techniques. Along with the principles of I/O organization and data transmission, several interfacing components and their interfacing to microprocessor systems are discussed in detail.

Chapter 7 plays the role of an appendix, where the characteristics of 16-bit microprocessors (Motorola MC 68000, Zilog Z 8000, Intel 8086 and their successors) are illustrated. This, together with the suggested references, provides microprocessor

users with an overview that is useful in their selection of a manufacturer.

We gratefully acknowlegde the assistance of Renate Kirchmann, Hildegard Klimmeck, Ingrid Kunkel, Rolf Malinowski and Dipl.-Math. Horst Seyferth during the realization of this book. In particular we would like to thank Dr. Jürgen Wazeck of Bosch Elektronik, Berlin for his cooperation; and Gigliola Bisiani, Pittsburgh who translated the book into English and carefully prepared the final version.

Berlin, July 1985 Th. Flik and H. Liebig

Table of Contents

1 Introduction to the Structure and the Programming of a Microprocessor

Microprocessor systems are general purpose programmable digital computers of small to medium capacity. Their advantages lie in the miniaturization of the components, in the low hardware costs and in the possibility to adapt the hardware to the problem at hand in a modular way. Thanks to these qualities, microprocessors are now used in new fields of application, which were precluded to conventional digital computers or had to be tackled by employing custom systems with very high design and production costs.

A starting point for the development of microprocessors is the technology, originated in 1948 with the discovery of the transistor, that allows the integration of complex logical switching circuits on semiconductor chips of a few mm^2. Thus in 1959, Fairchild, a semiconductor company, managed to place several transistors on a chip. By means of this technological advancement, the integration density could be increased and the switching times abbreviated thus increasing the capacity of the devices as well.

At the end of the Sixties, logic components became more complex in their function, but at the same time more specific, which increasingly limited their range of application. Datapoint, an American company producing so-called intelligent terminals, developed a simple programmable processor for terminal control in 1969 and commissioned the two semiconductor companies Intel and Texas Instruments to integrate it on a single chip. Intel succeeded in producing the component; nevertheless, it could not be employed for the intended application because of its low processing speed. Thus, Intel decided to market this processor as a programmable logic device. The two available versions had processing capacities of respectively 4 and 8 bits and were called Intel 4004 and Intel 8008. The age of microprocessors had begun.

In the meantime, 1-bit, 4-bit and above all 8-bit microprocessors have become rather popular in the fields of control, feedback control and mathematical computation. This development was supported by the production of entire families of microprocessors with a large number of additional components, that considerably facilitated the design of microprocessor systems. However, the drawback of these processors rested on their low performance when executing numerical problems. The new versions of 8-bit microprocessors solve this difficulty by means of multiplication and division instructions as well as operations with 16-bit and 32-bit operands.

The first 16-bit microprocessor, Texas Instruments' TMS 9900 [1], appeared on the market in 1977, and was successfully followed by Intel 8086 [2], Zilog Z8000 [3] and Motorola MC68000 [4] over the period 1978 - 1980. 16-bit microprocessors appear to have a substantially higher performance than 8-bit processors, but their structure is also more complex. On the one hand, this allows the introduction of microprocessors in applications typical of minicomputers, whereas on the other hand the system design is more complex and leads to bigger and more expensive systems.

A trend towards microprocessors with a capacity that is higher than 16 bits emerges from the technological advancements in the production of semiconductor components. At present, 32-bit microprocessors are already available on the market; they will substitute the 16-bit digital computing devices that were built with conventional technology. Despite this development, low performance microprocessors will retain the application areas for which their capabilities are sufficient.

In this book, we will be only concerned with 16-bit microprocessors; in particular, the first chapter will provide some introductory clarification. Section 1.1 will deal with the representation of information, Section 1.2 with the structure and behavior of 16-bit microprocessor systems and Section 1.3 with machine programming and assembly language level.

1.1 Representation of Information

1.1.1 Information Units

In the case of digital systems, the information is represented in binary form. The smallest unit of information is the bit (binary digit). A bit can take on two values, that we will define 0 and 1. When implemented, they are represented, for instance, as two different voltages on a signal line or as two different magnetization states on an information carrier. Several bits are combined into bigger information units. Thus, a so-called half byte is composed of 4 bits and one byte consists of 8 bits. In the case of microprocessors, a 16-bit unit is called word, a 32-bit unit is a double word. In the representation of information units, we will enumerate the individual bits from right to left starting with zero (Figure 1.1). Accordingly, bit 0 will be defined as the Least Significant Bit (LSB) and the bit with the highest index as the Most Significant Bit (MSB).

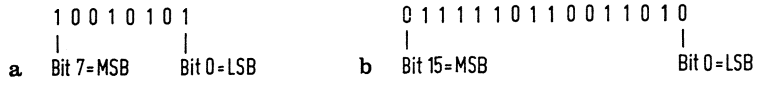

Fig. 1.1. Units of information. a) byte, b) word

1.1.2 Number Representation

In the decimal number system, the ten digits 0 through 9 are used for the representation of numbers; they are weighted by the 10th power of their position. Thus, the value of a number results from the sum of the weighted digits, for example:

$$205 = 2 \cdot 10^2 + 0 \cdot 10^1 + 5 \cdot 10^0$$

Binary Numbers. In the binary number system, only the two binary digits 0 and 1 are available for the representation of numbers, and they are weighted by the power of 2 of their position. As with decimal numbers, the value of a number results from the sum of the weighted digits. In the following example, in order to differentiate between number systems, the basis of a number is indicated by an index:

$$11001101_2 = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 205_{10}.$$

Number Conversion. The sum of the weighted digits also indicates the procedure for the conversion from binary to decimal numbers. Thus, the decimal value for the above example is 205. In order to convert a decimal number into a binary number, it has to be decomposed into its power of 2 components. A common procedure is to divide the decimal number by 2 and note the remainder; then the same procedure is to be repeated with the result until the result is zero. The remainder, which can have values 0 and 1, creates the binary number starting with the lowest bit (see also [6]). The above example can be broken up into:

205:2 = 102, remainder 1
102:2 = 51, remainder 0
51:2 = 25, remainder 1
25:2 = 12, remainder 1
12:2 = 6, remainder 0
6:2 = 3, remainder 0
3:2 = 1, remainder 1
1:2 = 0, remainder 1 → 11001101.

16-bit microprocessors perform operations with binary numbers in byte, word or often double word format, where binary numbers with a smaller number of digits than required are adapted to a larger format by means of so-called leading zeros. Since a unit of information in the representation is limited to n bits, the value of a number is restricted to the range between 0 and $2^n - 1$ in the case of binary numbers. If this range is exceeded in an arithmetic operation, the result is still within the range limits; its value, however, is not correct, and the microprocessor signals an arithmetic overflow by setting the Carry bit (C). Figure 1.2a illustrates the occurrence of arithmetic overflow with an example of 8-bit binary numbers on the number ring.

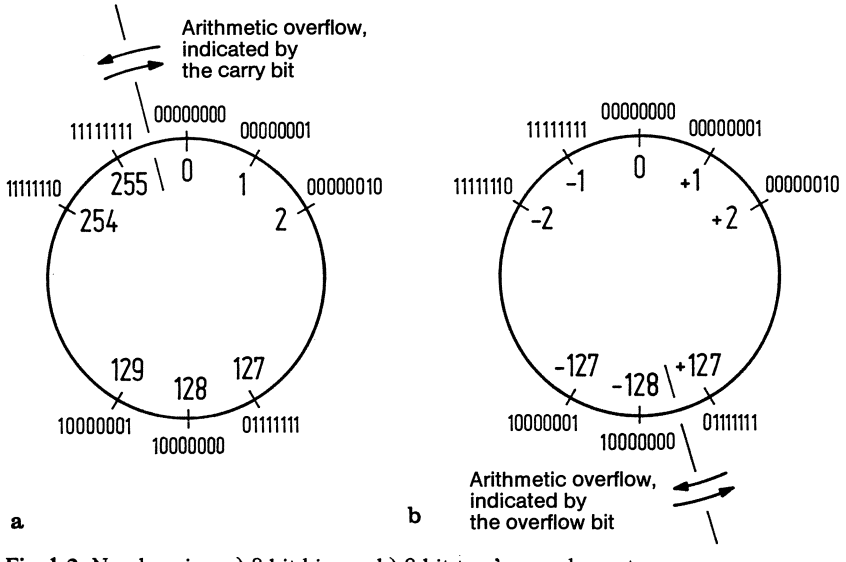

Fig. 1.2. Number ring. a) 8-bit binary, b) 8-bit two's complement

Negative Numbers. The most significant bit is used for the representation of the sign in order to distinguish between positive and negative numbers; in the case of positive numbers it is 0, for negative numbers it is 1. The values of the remaining $n-1$ bits are represented in binary code for positive numbers, whereas several representations are possible in the case of negative numbers: sign and magnitude numbers, one's complement numbers and two's complement numbers. Since 16-bit microprocessors are designed for the latter, we will only consider two's complement numbers in this book.

The sum of the positive and negative representation of a two's complement number of n digits adds to 2^n. The conversion from a positive to the corresponding negative number or viceversa (complementing) can be achieved through the bit complement of all n bits and the subsequent addition of 1; for example:

$+67_{10} = 01000011_2$

bit complement: 10111100
addition: + 00000001
complement: $10111101_2 = -67_{10}$.

The value of a two's complement number results from the sum of the weighted digits, where the sign bit is considered a negative weight. For example:

$$10111101_2 = -1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = -67_{10}.$$

The range of values extends from -2^{n-1} to $+2^{n-1}-1$ in the case of two's complement numbers, where zero is represented as a positive number. An arithmetic overflow, resulting from an arithmetic operation for instance, leads to an (incorrect) number within the range limits (Figure 1.2b). This is indicated by the overflow bit V of the microprocessor.

BCD Numbers. Another possibility of representing numbers in binary form is offered by the digital encoding of decimal numbers in binary coded decimals (BCD). In this case, the single decimal digits are substituted by 4-digit binary numbers and lined up according to Table 1.1. For example:

$205_{10} = 0010\ 0000\ 0101_{BCD}$.

Table 1.1. BCD code

Decimal code	BCD code
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Table 1.2. ASCII code (USASCII [27])

Bit position 7	6	5	4	3	2	1									
			0	0	0	0	NUL	DLE	SP	0	@*	P	'**	p
			0	0	0	1	SOH	DC1	!	1	A	Q	a	q
			0	0	1	0	STX	DC2	" *	2	B	R	b	r
			0	0	1	1	ETX	DC3	#*	3	C	S	c	s
			0	1	0	0	EOT	DC4	$	4	D	T	d	t
			0	1	0	1	ENQ	NAK	%	5	E	U	e	u
			0	1	1	0	ACK	SYN	&	6	F	V	f	v
			0	1	1	1	BEL	ETB	' *	7	G	W	g	w
			1	0	0	0	BS	CAN	(	8	H	X	h	x
			1	0	0	1	HT	EM	)	9	I	Y	i	y
			1	0	1	0	LF	SUB	*	:	J	Z	j	z
			1	0	1	1	VT	ESC	+	;	K	[*	k	{ *
			1	1	0	0	FF	FS	, *	<	L	*	l	¦ *
			1	1	0	1	CR	GS	-	=	M	]*	m	} *
			1	1	1	0	SO	RS	.	>	N	^*	n	~ *
			1	1	1	1	SI	US	/	?	O	_	o	DEL

1.1.3 Symbol Representation

The ASCII code (American Standard Code of Information Interchange) is most common means of representing symbols such as letters, digits and special symbols of an alphabet in microprocessor systems. Each symbol is represented by a 7-bit code (Table 1.2). If the code for a symbol is completed with an eighth bit, for example a zero at the beginning or an extra bit for code check, the symbol becomes as long as a byte.

1.1.4 Hexadecimal and Octal Representation

Reading binary coded information, like numbers, symbols or other bit combinations, is usually difficult for humans and, because of multiple digit numbers, hard to grasp. For this reason, binary information is usually represented in a condensed form when outside a microprocessor system, e.g.

on a printer output. Most of the time, we will adopt the hexadecimal representation, where each bit combination is given as a number in the basis 16 number system. The decimal digits 0 through 9 and the letters A through F are used as hexadecimal digits with values 0 through 15. When the bit combination is to be translated into hexadecimal representation, bits are divided from right to left in a sequence of 4-bit units, and each unit is associated with the hexadecimal digit that corresponds to the 4-bit binary number. For example:

1100 1010 1111 0101 = CAF5

Table 1.3 shows the correspondence of hexadecimal digits to 4-bit binary numbers.

Table 1.3. Hexadecimal code

4-bit binary code	Hexadecimal code
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	A
1 0 1 1	B
1 1 0 0	C
1 1 0 1	D
1 1 1 0	E
1 1 1 1	F

Another common compact representation is the octal code. In this case, each bit combination is represented by a digit in the basis 8 number system. The decimal digits 0 through 7 are used for the representation of digits, and 3-bit units are combined. For example:

1 100 101 011 110 101 = 145365.

For an in-depth study of information representations see for example [5-7].

1.2 Introduction to the Hardware Structure

1.2.1 Hardware Components of a Microprocessor System

The basic structure of a microprocessor system consists of three hardware components: the actual microprocessor for the processing of mathematical quantities (operands, results) controlled by a program, the main memory (working memory) for the storage of mathematical quantities and of the program, and the I/O unit for the input and output of data (programs and mathematical quantities) (Figure 1.3).

The microprocessor itself includes a control unit and a data processing unit. The control unit manages the microprocessor as well as the main memory and the I/O unit. It reads instructions from the main memory, interprets them and controls their execution. The operands that have to be processed are read from the main memory or by means of an I/O unit into the data processing unit. The results are made available to the main memory or to an I/O unit. Thus, the data processing unit plays the role of a translator between operands and results by executing logical and arithmetic operations on the operands.

The main memory includes a large number of single memory cells organized in byte or word format; it is useful to think of these cells as arranged in sequential order. For selecting a given cell, memory locations are numbered; i.e., each memory location can be unequivocally identified by a number. This number is called address. Addresses as well can be represented as 0/1 combinations; with 16 bits one can, for example, encode a maximum of $2^{16} = 65536 = 64$ K addresses. The limits of the available address space of the main memory, and therefore its memory cell capacity, are thus indicated by the address length (K stands for "one thousand": $1\text{ K} = 2^{10} = 1024$).

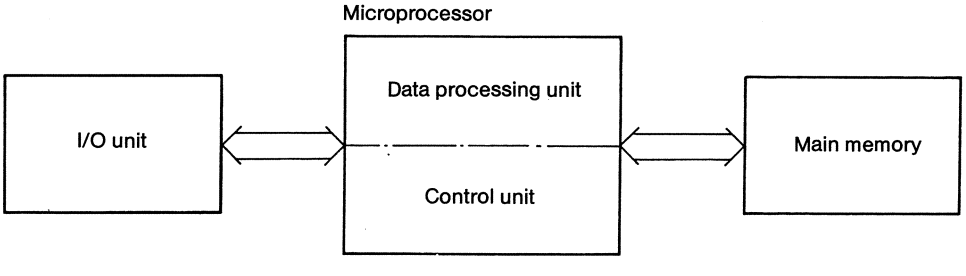

Fig. 1.3. Components of a microprocessor system

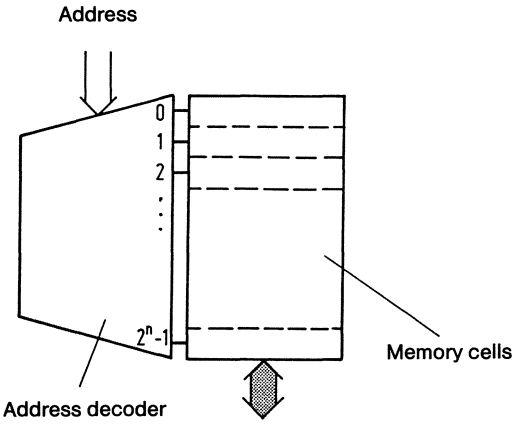

Fig. 1.4. Random access memory (RAM)

Figure 1.4 introduces the symbolic representation of a memory. The rectangular field defines the writeable and readable memory cells, and the trapezoid stands for the address decoder, which selects just one memory cell when an address is applied. These memories, whose cells can be all accessed in the same amount of time, are also called Random Access Memories (RAM's).

The microprocessor itself only possesses very few memory cells when compared to the main memory. Some of these cells are designated to particular purposes within the processor and are not directly available for machine and assembly programming. Other processor memory cells can be accessed like main memory cells. Such single memory cells, which can also be found in other system components, are often called registers; several registers linked together to form a small memory are called register files. Although registers and memory cells do not differ in their functionality, we would like to keep the conceptual distinction between registers as single memory cells and big memories.

The I/O unit represents the interface between the microprocessor system and the peripherals (for example an I/O device). It is thus called interface unit or just interface. Its simplest form includes an addressable register for the temporary storage of the data to be transported. More complex interfaces take over control functions for the data transmission. For example, they allow the synchronization between an often much slower peripheral device and the fast working microprocessor hardware.

1.2.2 Basic System Structure

In Figure 1.3 there are connection paths, that are independent from each other and connect the single components of a microprocessor system; these paths generally indicate the transfer of

information. In reality, we distinguish between three types of information for which separate connection paths are usually implemented:

1. data, which includes instructions, operands and results;

2. addresses for the selection of memory cells and registers; and

3. signals for the control of the information exchange between the individual components.

The cost for the connecting paths would greatly increase with the number of components in a situation similar to what portrayed in Figure 1.3. Thus, data, address and signal control paths are built only once and are made available to all components for information transmission. These connecting paths are also called buses, or respectively data bus, address bus and control bus. If combined they are called system bus. A bus is a collection of functionally related signal lines that connect at least two components of a digital system for the exchange of information. These components can be single registers, or complete functional units, such as microprocessors, memories and I/O units.

Figure 1.5 shows a bus-oriented microprocessor system. The data bus is represented by a shaded double line, the address bus by a clear double line. The control bus is indicated by two single thick lines, where each line represents several control circuits. In the future we will retain this convention and we will omit additional characterization of buses in the figures.

Unlike Figure 1.3, Figure 1.5 presents several memory units and several I/O units, which indicate the modularity of a bus-oriented microprocessor system. In this rather simple, but basic system

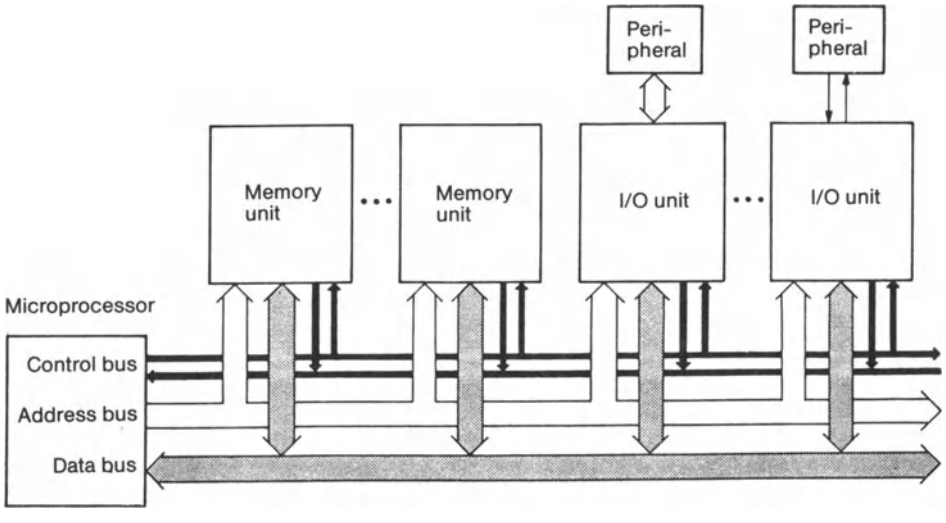

Fig. 1.5. Bus-oriented microprocessor system

configuration, the microprocessor is the only active component, i.e. it is the only device that can control the bus system. The memory and I/O units behave passively. Only two of the system components (a sender and a receiver) are simultaneously connected to the common data bus, so as to avoid conflicts during data transfer.

As shown in Figure 1.5, the data bus allows data transfers in both directions; it can be switched in the direction of the transfer. Such a bus is known as a bidirectional bus. On the contrary, the address bus allows transfers only in one direction; such a bus is called unidirectional. Most of the signal lines of the control bus are unidirectional, some of them are bidirectional. According to their function, the unidirectional control lines permit the signal flow to either go towards or away from the microprocessor.

The advantages of a bus-oriented microprocessor system lie in the low cost and easy extensibility of the general purpose connection paths. The single components of the microprocessor system will be briefly discussed below.

1.2.3 The Microprocessor

The behavior of a microprocessor system is specified by a sequence of instructions stored as a program in the main memory and processed by the microprocessor. The possible processor operations are determined by its instruction set. The most important instructions are those for data transfer, for arithmetic and logical operations, and instructions for so-called program branches.

Instruction Formats. For a dyadic operation that combines two operands and achieves one result, we need four specifications:

1. Type of operation (op code),
2. Address of the first operand (first source address),
3. Address of the second operand (second source address),
4. Address of the results (destination address).

If these four specifications are combined in one instruction, we have a three-address instruction format corresponding to Figure 1.6.

To quote an example, 8-bits for the op code and 16 bits for each address result in a 56-bit long instruction. In relation to a memory word length of 16 bits, a similar instruction would use four memory cells when storing a program, where 8 bits would go unused in one cell.

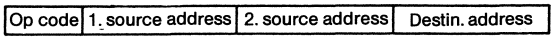

Fig. 1.6. Format of a three-address instruction

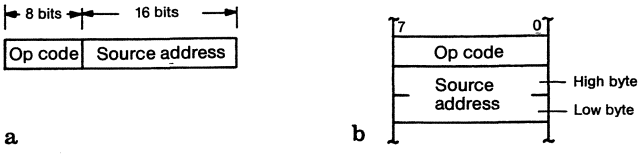

Fig. 1.7. One-address instruction of an 8-bit microprocessor: a) general purpose format, b) byte-oriented memory representation

In order to shorten the size of an instruction, the number of addresses in the instruction, for instance, can be reduced. Moreover, there are two possibilities, implicit and hidden addressing, that can be often used in combination and can be found in both 8-bit and 16-bit microprocessors.

One-Address Instructions. 8-bit microprocessors are equipped with a special register, the accumulator, that is addressed as the source of one of the two operands whenever there is a dyadic operation. At the same time, this register is used as the destination for the result, so that the operand that was originally stored in the accumulator is lost. The address of the accumulator is not given explicitly in the instruction, but it is implicitly contained in the op code (implicit addressing); in addition, two addresses coincide thanks to the double function of the accumulator (hidden addressing). Besides the op code, the source address of the second operand is also indicated in the instruction as a memory address. Special load and store instructions allow the operand transfer between the accumulator and other registers or memory cells. Figure 1.7a shows such a one-address instruction format for an 8-bit microprocessor. In a byte-organized memory typical of an 8-bit microprocessor, this instruction uses three consecutive memory cells (Figure 1.7b).

One of the drawbacks of one-address instructions, when carrying out a two-operand operation, is that they need three instructions altogether:

1. load the accumulator with the content of a memory cell (first operand),
2. combine the content of the accumulator with the content of a memory cell (second operand),
3. store the content of the accumulator (result) in a memory cell.

Two-Address Instructions. Instead of the typical one-address instruction accumulator, 16-bit microprocessors have a set of 8 or 16 general purpose processor registers. Each one of them can, among other things, assume the function of the accumulator. Moreover, a register address is