Introducing Microprocessors

A new series on understanding microprocessors and how they work.

By Mike Tooley

The general learning objectives for part one of Introducing Microprocessors are that readers should be able to:

(a) understand the terminology used to describe microcomputer and microprocessor based systems

(b) identify the major logic families and scale of integration employed within integrated circuits

(c) draw a diagram showing the architecture of a representative microcomputer system and state the function of the principal internal elements

(d) understand binary and hexadecimal number systems and convert from/to decimal

The specific objectives for this part are as follows:

1.1 Terminology

1.1.1 State the meaning of any of the terms listed in the glossary 1.1.2 Distinguish between the terms; computer, microprocessor, microcomputer, and single-chip microcomputer.

1.2 Integrated Circuit Terminology and Logic Families

1.2.1 Define the terms SSI, MSI, LSI and VLSI as applied to integrated circuits.

1.2.2 State the characteristics of CMOS and TTL semiconductor technologies.

1.2.3 State the basic properties of CMOS and TTL logic families.

1.3 System Architecture

1.3.1 Draw a block diagram showing the internal architecture of a representative microcomputer system.

1.3.2 State the function of each of the principal internal elements of a representative microcomputer system.

1.3.3 Explain the bus system used to link the internal elements of a microcomputer system.

1.3.4 Distinguish between the following types of bus; address, data and control.

Related Theory

Explain the binary and hexadecimal number systems.

Convert binary, hexadecimal and decimal numbers over the range 0 to 65535 (decimal).

Explain how negative numbers are represented in microprocessor systems.

Perform addition and subtraction of binary and hexadecimal numbers over the range 0 to 255 (decimal).

Note: Rather than publish a complete glossary of terms, we shall be producing a mini-glossary for each part; introducing new terms as we progress through the series.

Integrated Circuits

The vast majority of today's electronic systems rely on the use of integrated circuits in which hundreds of thousands of components are fabricated on a single chip of silicon. A relative measure of the number of individual semiconductor devices within the chip is given by referring to its "scale of integration", as shown below:

Scale of integration	Abbreviation	Logic gate equivalent
Small	SSI	1 to10
Medium	MSI	10 to 100
Large	LSI	100 to 1000
Very large	VLSI	1000 to 10000
Super large	SLSI	10000 to 100000

The "logic gate equivalent" referred to in the table provides us with a rough measure of the complexity of integrated circuit and simply gives the equivalent number of standard logic gates. A logic gate is a basic circuit element capable of performing a logical function (such as AND, OR, NAND or NOR). A basic logic gate (e.g. a standard TTL two-input AND) would typically employ the equivalent of six transistors, three diodes and six resistors. At this stage, readers need not worry too much about the function of a logic gate as we shall be returning to this later on. Readers need only be aware that such devices form the basis of circuits which can perform logical decisions.

Integrated circuits are encapsulated in a variety of packages but the most popular type (and that with which most readers will already be familiar) is the plastic dual in-line (DIL) type. These are available with a differing number of pins depending upon the complexity of the integrated circuit in question and, in particular, the need to provide external connections to the device.

Conventional logic gates, for example, are often supplied in 14- pin or 16-pin DIL packages while microprocessors (and their more com-

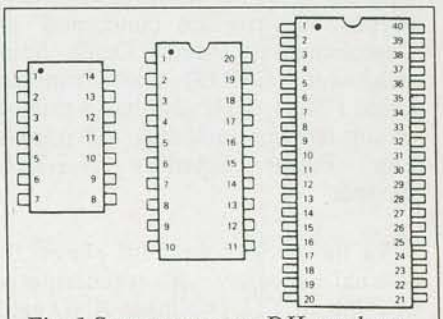


Fig. 1 Some common DIL package outlines (top view).

plex support devices) often require 40-pins or more. Fig. 1 shows some common DIL package outlines together with pin numbering. It should be noted that these are TOP views of the devices, i.e. they show how the device appears when viewed from the component side of the PCB, NOT from the underside.

In each case, the pins of the IC are numbered sequentially (starting at the indentation) moving in an anti-clockwise direction. Thus, in the case of a 14-pin DIL package viewed from the top, pins one and 14 appear respectively on the left and-right-hand side of the indentation.

Problem 1.1

Fig. 2 shows the outline of an 8-pin DIL device viewed from above. Identify the pin numbers for this device.

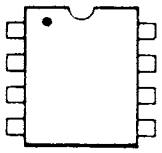


Fig. 2. An 8-pin DIL package.

Logic Families

The integrated circuit devices on which modern digital circuitry depends belong to one of the other of several "logic families". Readers should note this term refers to the type of semiconductor technology employed in the fabrication of the integrated circuit.

The semiconductor technology employed in the fabrication of an integrated circuit is instrumental in determining its characteristics. This encompasses such important criteria as supply voltage, power dissipation, switching speed, and immunity to noise.

The most popular logic families, at least as far as the more basic general purpose devices are concerned, are Complementary Metal Oxide Semiconductor (CMOS) and Transistor Logic (TTL), TTL also has a number of sub-families including the popular Low Power Schottky (LS-TTL) variants.

For the curious we have shown the internal circuitry of representative CMOS and TTL two-input AND gates in Fig. 3. Despite the obvious dissimilarity of these two arrangements, it must be stressed that they have IDENTICAL logical functions.

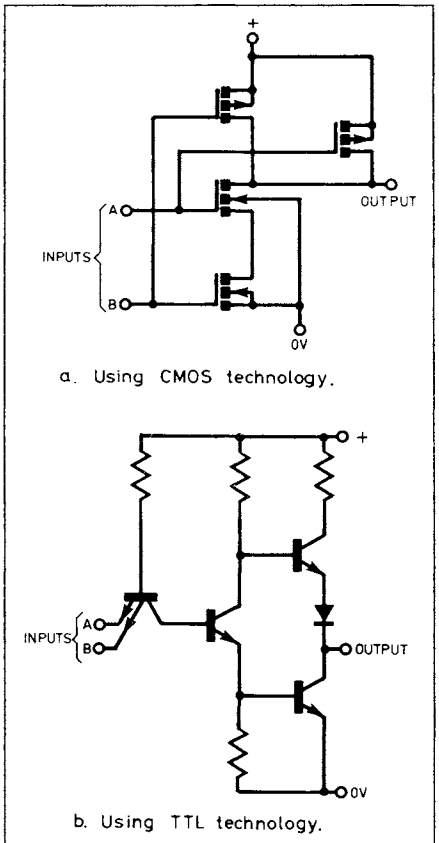


Fig. 3. Representative circuitry of two-input AND gates.

TTL logic

The most common range of conventional TTL logic devices is known as the "74" series. These devices are, not surprisingly, distinguished by the prefix number 74 in their coding. Thus devices coded with the numbers 7400, 7408, 7432, and 74121 are all members of this family which is often referred to as "Standard TTL". Other versions (or "subfamilies") of standard TTL exist and these are distinguished by appropriate letters placed after the "74" coding. Where "Low Power Schottky" technology is used in the manufacture of a TTL logic gate, for example, device coding include the letters "LS". Thus a 74LS00 device is a Low Power Schottky version of a standard 7400 device.

CMOS Logic

The most popular CMOS family is the "4000" series and devices have an initial prefix of 4. Thus 4001, 4174, 4501 and 4574 are all CMOS devices. CMOS devices are sometimes also given a suffix letter; A to denote the "original" (now obsolete) unbuffered series, and B to denote the improved (buffered) series. A UB suffix denotes an unbuffered B-series device.

Problem 1.2

To which logic families do each of the following devices belong?

(a) 7407
(b) 74LS74
(c) 4052

Power supplies

Most TTL and CMOS logic systems are designed to operate from a single supply voltage rail of nominally +5V. With TTL devices, it is important for this voltage to be very closely regulated. Typical TTL IC specifications call for regulation of better than 5% (i.e. the supply voltage should not fall outside the range 4.75V to 5.25V).

CMOS logic devices are fortunately very much more tolerant of their supply voltage than their TTL counterparts. Most CMOS devices will operate from a supply rail of anything between +3V and + 15V. This, coupled with a minimal requirement for supply current (a CMOS gate typically requires a supply current of only a few microamps in the quiescent state) makes them eminently suited to portable battery powered equipment.

TTL devices require considerably more supply current than their CMOS equivalents. A typical TTL logic gate requires a supply current of around 8mA; approximately 1000 times that of its CMOS counterparts when operating at a typical switching speed of 10kHz.

CMOS versus TTL

Having spent some time in discussing the merits of CMOS and TTL devices it is now worth summarizing three of the important differences between these logic families in tabular form:

Problem 1.3

Which logic family would be most suited for use in a piece of equipment which has to operate over long periods from dry batteries?

Microprocessors

Having dealt with the basic building blocks of digital circuits we have at last

	CMOS	TTL
Speed of operation	Relatively slow (power consumption increases as switching speed increases)	Fast (power consumption substantially constant)
Power consumption	Negligible (but see above)	Appreciable
Supply voltage	Operates over a wide voltage range (typically 3V to 15V)	Must be closely regulated at, or near 5V

arrived at the point at which we can introduce the microprocessor.

Microprocessors are VLSI and SLSI integrated circuit devices which are capable of accepting, decoding and executing instructions presented to them in binary coded form. Microprocessors thus form the heart of all microcomputer systems in which they act as the central processing unit (CPU). Despite this, the majority of microprocessors are not capable of providing the complete range of facilities expected of a computer (i.e. input, processing, memory and output). Microprocessors require a certain amount of external hardware and other support devices, not the least important of which are those which provide a memory for the sequences of software instructions (i.e. programs) and transient information (i.e. data) used during processing.

Some specialized microprocessors incorporate their own internal memory for data and program storage as well as input/output ports for the connection of external devices such as keyboards and displays. These devices are aptly known as "single-chip microcomputers" and they are ideal for use in low-cost stand- alone control systems.

Microprocessors are often divided into categories depending upon the size of the binary number on which they fundamentally perform operations. Most modern microprocessors perform operations on groups of either eight or 16 binary digits (bits). Whilst 16-bit microprocessors tend to be more powerful than their 8-bit counterparts this is unimportant in many simple applications.

The first generation of 8-bit microprocessors appeared a little over 14 years ago in the shape of an Intel device, the 8008. At the time, this was something of a minor miracle - a device which could replace countless other chips and which could address a staggering 16K of memory! By modern standards, the 8008 is extremely crude but it was not long before Intel introduced another device, the 8080. This time NMOS technology was employed instead of the PMOS technology which was used in the 8008. The 8080 had 16 address lines (thus being able to address 64K of memory) and 78 software instructions for the programmer to use. The 8080 was an instant success and led the way to enhanced devices such as the 8085 and the Z80.

Other manufacturers were also developing microprocessor chips hard on the heels of Intel. These included Motorola (with the 6800) and MOS Technology (with the 6502). In subsequent years industry has not been content to stand still and much effort has been devoted into huge advances into 16 and 32-bit technology. Despite this, all of these simple 8-bit microprocessors (and their various enhancements and derivatives) are still in common use today.

Costs have fallen very significantly and it is eminently possible to put together a microprocessor system (comprising CPU and a handful of support chips) at a very moderate cost. Therefore, if one had the task of designing, for example, a simple control system, one would almost certainly use a microprocessor (or single-chip microcomputer) to form the basis of the controller.

Such a system would not only be capable of fulfilling all of the functions of its conventional counterpart but it would also provide a far more sophisticated means of processing our data coupled with the ability to store it and examine it at a later date or even transmit it to a remote supervisory computer installation. The vast saving in hardware development can usefully be devoted to the software aspects of a project and future modifications can simply involve the substitution of "firmware" (ROM based software).

Bits, Bytes and Buses

An 8-bit microprocessor fetches and outputs data in groups of 8- bits (known as "bytes"). This data is moved around on eight separate lines (labelled D0 to D7) which collectively form a "data bus". For the curious, the word "bus" is a contraction of the Greek work "omnibus" which simply means "to all", thus aptly describing the concept of a system of wiring which links together all of the components of a microprocessor system using a common set of shared lines.

Microprocessors determine the source of data (when it is being "read") and the destination of data (when it is being "written") by outputting the location of the data in the form of a unique "address". This process involves placing a binary pattern on an "address bus". In the case of 8-bit microprocessors, the address bus invariably comprises 16 separate lines, labelled A0 to A15.

The address at which the data is to be placed or from which it is to be fetched can either constitute part of the "memory" of the system (i.e. RAM or ROM) or can be considered to be "input/output" (I/O). The allocation of the 64K memory address range of an 8-bit microprocessor can usefully be described using a "memory map". We shall discuss this in greater detail later in the series.

A further bus is used for a variety of general housekeeping functions such as determining the direction of data movement (i.e. whether a "read" or "write" operation is being performed), and placing the machine in an orderly state of power-up. This bus is known as the "control bus" and often has between five and 13 lines depending upon the microprocessor.

System Architecture

We have already identified the principal elements within a microprocessor system as CPU (i.e. the microprocessor), RAM, ROM, and I/O. Before showing how they are interconnected within a typical microprocessor system,

it is worth briefly restating the function of each:

Central processing unit (CPU)

The central processing unit in a microcomputer is a single VLSI device, the microprocessor. The device accepts instructions and data for processing. It also provides control and synchronizing information for the rest of the system. We shall look more closely at the function of the microprocessor in Part Two.

Random Access Memory (RAM)

All microprocessors require access to read/write memory and, while single-chip microcomputers contain their own low-capacity area of read/write memory, read/write memory is invariably provided by a number of semiconductor random access memories (RAM).

Read-only Memory (ROM)

Microprocessors generally also require more permanent storage for their control programs and, where appropriate operating systems and high-level language interpreters. This facility is invariably provided by means of semiconductor read-only memories (ROM).

Input/Output Devices (I/O)

To fulfil any useful function, the microprocessor needs to have links with the outside world. These are usually supplied by means of one, or more, VLSI devices which may be configured under software control and are thus said to be "programmable". I/O devices fall into two general categories; "parallel" (a byte is transferred at a time) or "serial" (one byte is transferred after another along a single line).

The basic configuration of a microprocessor system is shown in Fig. 4, the principal elements; microprocessor, ROM, RAM, and I/O have been shown. Note that the three bus systems; address, data, and control are used to link the elements and thus an essential requirement of a support device is that it should have "tri- state" outputs. It can thus be disconnected from the bus when it is not required.

Support devices (such as ROM, RAM, etc.) are fitted with select or enable inputs. These lines are usually driven by address decoding logic (not shown in Fig. 4). The inputs of the address decoding logic are derived from one, or more, of the address bus lines. The address decoder effectively divides the available memory into blocks which each correspond to a particular support device. Therefore, where the processor is reading and writing to RAM, for example, the address decoding logic will ensure that only the RAM is selected and the internal buffers in the ROM and I/O chips are kept in the tri-state output condition.

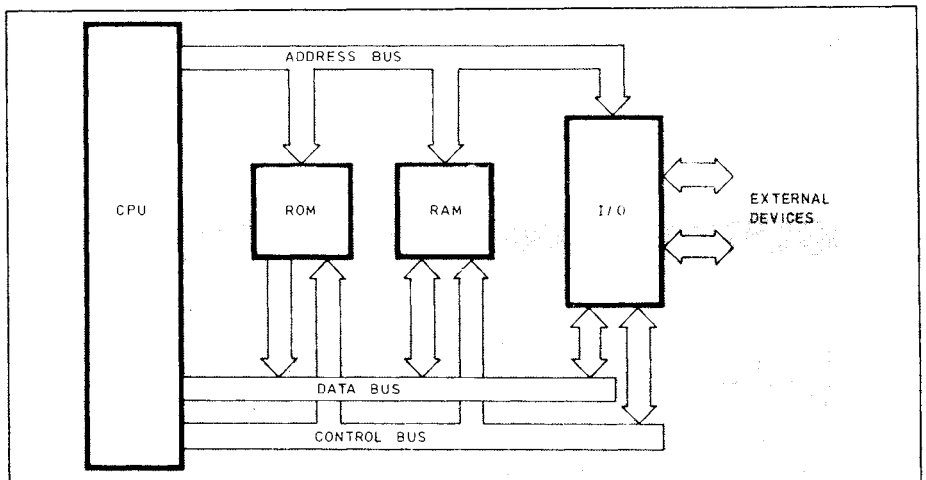


Fig. 4. Principal elements of a microprocessor system.

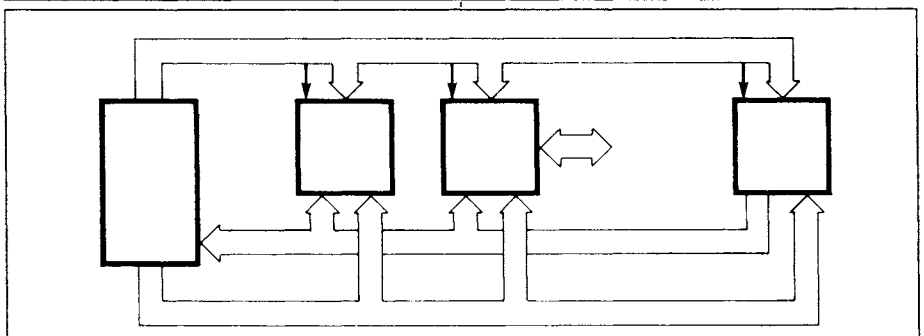


Fig. 5. Microprocessor system (see problem 1.4).

Table 1 Powers of two

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768

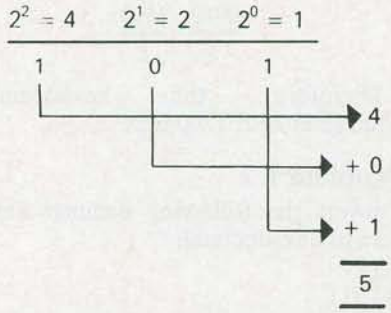
Problem 1.4

The diagram shown in Fig. 5 represents a microprocessor system. Compare this diagram with that shown in Fig. 4 and hence label the diagram fully.

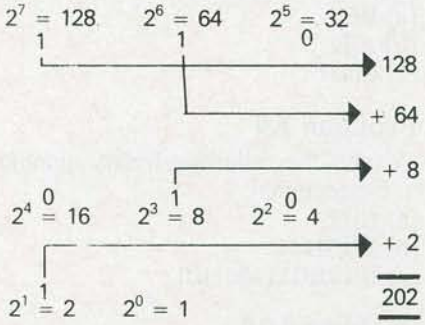
Number Systems

Unfortunately, the decimal or decimal number system with which we are all very familiar, is not at all appropriate to microcomputers. The reason is simply that electronic logic circuits are used on devices which have only two states (variously known as "on/off" "high/low", and "true/false"). The number system most appropriate is therefore binary in which only two digits (0 and 1) are used. Powers of two are shown in Table 1.

The process of converting binary to decimal and decimal to binary is quite straightforward. The binary number 101, for example, can be converted to decimal as follows:



Now, taking a somewhat more difficult example, let's convert the binary number 11001010 to decimal. We shall again write the number using column headings but this time we shall simply ignore the zeros in our addition:

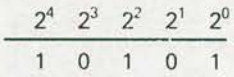


There are two basic methods for converting decimal to binary. One involves taking the decimal number and subtracting from it successively smaller numbers which are themselves powers of two. Where a power of two can be subtracted (to leave zero or a remainder), a 1 is placed in the appropriate column of the binary number. This all sounds very much more difficult than it really is, so here is a simple example to illustrate the process.

Suppose that we wish to convert the decimal number 21 to binary. The highest power of two which can be taken away from 21 is 2^4 (=16). This leaves a remainder of 21 - 16 = 5. 2^3 (=8) cannot be subtracted from the remainder but 2^2 can to leave a remainder of 5-4 = 1. 2^0 (=1) can now be subtracted from the second remainder to leave zero. Another way of putting this is that;

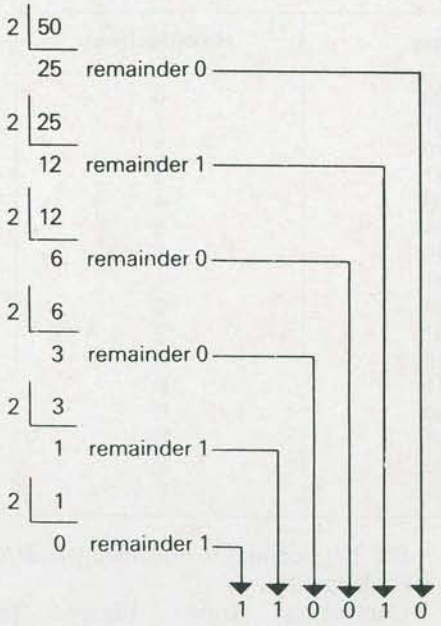
$$21 = 16 + 4 + 1 = 2^4 + 2^2 + 2^0$$

We can write this as a binary number by placing 1s and 0s in columns, as follows:



The second method involves successively diving the decimal number by two and noting down the remainder each stage. The binary number is then formed by reading the remainders as shown in the example below (note that the least significant remainder becomes the most significant bit of the binary number!).

Suppose we wish to convert the decimal number 50 to binary.



Therefore 50 decimal is equal to 110010 binary.

Which of the two methods you employ is entirely a matter of preference. If you can easily spot the powers of two present in a decimal number it is probably best to use the first method. If, on the other hand, you are uncertain as to which powers of two make up a number it is best to use the longer method. Remember that, in both cases, you can always convert back to decimal in order to check your results!

Problem 1.5

Convert the following binary numbers to decimal:
(a) 1011
(b) 100110
(c) 11100111

Problem 1.6

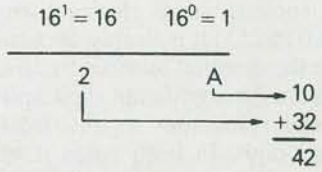
Convert the following decimal numbers to binary:
(a) 33
(b) 100
(c) 213

Hexadecimal

The binary number system is somewhat tedious for human use since numbers of any appreciable size are rather difficult to handle. For this reason we settle on the hexadecimal (base 16) number system. This has the advantage that it is relatively simple to convert from hexadecimal to binary and the hexadecimal numbers which can appear on an 8-bit data bus can be represented using just two digits (rather than the eight binary digits which would otherwise be necessary).

Since the hexadecimal numbering system employs more than 10 digits we have to make use of the first six letters of the alphabet to represent those equivalent to the decimal numbers 10 to 15. Hexadecimal digits thus range from 0 to F as shown below left.

Numbers in excess of fifteen will, of course, require more than one hexadecimal digit. Suppose, for example, we wish to convert the hexadecimal number 2A to decimal. Using a similar technique to that which we employed when converting from binary to decimal we would arrive at:



Now, suppose that we wish to convert 2A to binary rather than decimal. This is an easy process (if is NOT necessary to convert to decimal as an intermediate step!) as shown below:

Hexadecimal	2	A
Binary	0010	1010

Therefore hexadecimal 2A is equivalent to binary 00101010. Note that the leading zeros are optional (00101010 is exactly the same number as 101010). When dealing with numbers present on an 8-bit bus it is, however, a good idea to get into the habit of showing leading zeros in bi-

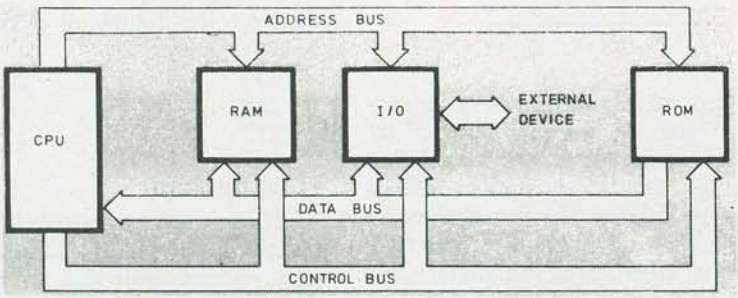


Fig. 6 Answer to problem 1.4

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

nary numbers so that all 8-bits are clearly shown.

To convert from decimal (in the range 0 to 255) it is simply necessary to divide the decimal number by 16 to obtain the most significant digit and then take the remainder as the least significant digit. In both cases it will be necessary to convert a digit answer greater than nine to its corresponding hexadecimal digit (i.e. A to F). Suppose, for example, we had the task of converting decimal 75 to hexadecimal:

16 | 75
4 Remainder 11
= B
4B

Therefore, decimal 75 is equivalent to 4B hexadecimal.

Taking another example, suppose we had to convert 250 decimal to hexadecimal:

16 | 250
15 Remainder 10
= F = A
FA

So, 250 decimal is equivalent to FA hexadecimal.

Converting from binary to hexadecimal over the range 0 to 255 is extremely simple. All we need to do is to separate the binary number into two groups of four bits (known as "nibbles") and then convert each of these groups to its corresponding hexadecimal character.

As an example suppose we need to convert the binary number 10001001 to hexadecimal:

1000 1001
8 9

The binary number 10001001 is equivalent to hexadecimal 89.

Taking one further example, suppose we have to find the hexadecimal equivalent of the binary number 101100 (note that this number consists of six rather than eight bits):
101100 = 00101100 (inserting two leading zeros to form an eight bit binary number)

0010 1100
2 C

Therefore, the hexadecimal equivalent of 101200 is 2C.

Problem 1.7

Convert the following decimal numbers to hexadecimal:
(a) 27
(b) 511
(c) 4100
(Hint: Refer to Table 3 for powers of 16)

Problem 1.8

Convert the following hexadecimal numbers to decimal:
(a) BE
(b) 10B
(c) C000

Problem 1.9

Convert the following binary numbers to hexadecimal:
(a) 1011
(b) 10101111
(c) 1010101111001101

Problem 1.10

Convert the following hexadecimal numbers to binary:
(a) 3F
(b) 23A
(c) 812C

Addition

Binary and hexadecimal addition follows the same general rules as employed with decimal numbers. The rule for addition of two numbers (which MUST be expressed in the same base) is that digits in the corresponding position in each number are added, starting from the right most (least significant) and a carry into the next most significant position occurs if the sum of two digits equals or exceeds the value of the base used.

As an example, suppose we have to add the binary numbers 10010010 and 01010000.

	1	0	0	1	0	0	1	0
+	0	1	0	1	0	0	0	0
Carry	0	0	1	0	0	0	0	0
	1	1	1	0	0	0	1	0

(Note that a carry is generated when the two 1s are added and the carry is added into the next more significant column.)

To make the process clear it is worth taking another example in which we shall add the binary numbers 10001101 and 01001111:

	1	0	0	1	1	1	0	1
+	0	1	1	0	1	1	1	1
Carry	1	1	1	1	1	1	1	0
	1	0	0	0	0	1	1	0	0

Addition of hexadecimal numbers follows the same pattern. Suppose we have to find the sum of the hexadecimal numbers 14 and 23.

	1	4
+	2	3
Carry	0	0
	3	7

Therefore, the sum of hexadecimal numbers 14 and 23 is (not surprisingly!) equal to 37.

Now let's try something a little more difficult by finding the sum of 1F and 2C:

	1	F
+	2	C
Carry	1	0
	4	B

(Note that adding F to C gives 1B)

Therefore, adding 2C to 1F gives 4B hexadecimal.

Problem 1.11

Add:
(a) the binary number 01001111 to the binary number 10000101
(b) the hexadecimal number 5C to the hexadecimal number 71.

Subtraction

Subtraction of binary and hexadecimal numbers also follows the same general rules as employed with decimal numbers. The rule of subtraction of two numbers (which again MUST be expressed in the same base) is that digits of the number to be subtracted (the subtrahend) are subtracted from digits in the corresponding position of the minuend. If the difference obtained is less than zero (i.e. negative) a value equals to the base employed is BORROWED FROM the next most significant position. The value of the borrow (i.e. 1) must be ADDED TO the next most significant position of the subtrahend. Again all this sounds much more complicated than it really is so we shall again use several examples to illustrate the techniques.

Consider the problems of subtracting the binary number 0100 from the binary number 1101:

	1	1	0	1	(minuend)
−	0	1	0	0	(subtrahend)
Borrow	0	0	0	0	
	1	0	0	1	

(In this simple example we have not needed to borrow)

Now consider the slightly more difficult problem of subtracting the binary number 0110 from 1101.

	1	1	0	1	(minuend)
−	0	1	1	0	(subtrahend)
Borrow	1	1	0	0	
	0	1	1	1	

(Note that, if you are unsure of an answer, it is always permissible to convert the numbers to decimal in order to check the validity of a result. In the foregoing example we have the equivalent decimal calculation; 13 − 6 = 7)

Subtraction of hexadecimal numbers again follows the same pattern. Suppose we have to find the difference of the hexadecimal numbers 2F and 1A:

	2	F
−	1	A
Borrow	0	0
	1	5

(Note that subtracting A from F leaves 5)

So, the result of subtracting 1A hexadecimal from 2F hexadecimal is 15 hexadecimal.

Now let's try something a little more difficult by subtracting hexadecimal 14 from hexadecimal 23:

	2	3
−	1	4
Borrow	1	0
	0	F

Therefore, the result of subtracting the hexadecimal number 14 from the hexadecimal number 23 is 0F.

Problem 1.12

Subtract;
(a) the binary number 01001101 from the binary number 110000101
(b) the hexadecimal number 3C from the hexadecimal number F0

Negative Numbers

Thus far we have confined ourselves to positive numbers and many of you will be wondering how we go about representing negative numbers within a computer. One rather obvious method is that of using the first (most significant) bit of a number solely to indicate its sign (i.e. whether it is positive or negative). The convention used is that a 0 in the most significant bit (MSB) position indicates that the number is positive. An MSB of 1, on the other hand, indicates that the number is negative. The magnitude of the number is then given by the remaining bits. We refer to such a number as being "signed".

As an example the signed eight-bit binary number 01111111 represents the decimal number 127 (it is positive as the MSB is 0) whereas the signed binary number 11111111 represents the decimal number − 127 (the MSB is 1 and therefore the number is negative).

Problem 1.13

Convert the following signed binary numbers to decimal:
(a) 01000000
(b) 11000000
(c) 10001001

Problem 1.14

Convert the following decimal numbers to signed 8-bit binary:
(a) +5
(b) -99
(c) +99

One's and Two's Complement

The one's complement of a binary number is found by simply changing all of the 0s present to 1s and changing all of the 1s present to 0s. This process is called "inversion". The one's complement of the binary number 1010 is thus 0101. Note that the result of adding a number to its one's complement will produce a succession of 1s, as shown in the following example:

Binary number	1 0 1 0 1 1 0 0
One's complement	0 1 0 1 0 0 1 1
Sum	1 1 1 1 1 1 1 1

The two's complement of a binary number is found by simply adding 1 to its one's complement. Therefore, the two's complement of 1010 is 0110 (i.e. 0101 + 0001).

The result of adding a binary number to its two's complement is not a succession of 1s but a succession of 0s with a leading 1 in the next more significant bit position. This is demonstrated by the following example:

Binary number	1 0 0 0 1 0 1 1
One's complement	0 1 1 1 0 1 0 0
Two's complement	0 1 1 1 0 1 0 1
	added 1

Original number	1 0 0 0 1 0 1 1
Two's complement	+ 0 1 1 1 0 1 0 1
Sum	1 0 0 0 0 0 0 0 0

Problem 1.15

Find the one's and two's complements of each of the following:
(a) 1001
(b) 10110011
(c) 10001110

Readers can be excused for wondering what all this is leading up to. It actually yields a very neat method for performing subtraction. We simply form the two's complement of the number to be subtracted (the subtrahend) and ADD it to the first number (the minuend). Any leading 1 generated is ignored. To show how this works consider the following example which subtracts the binary number 00110110 from the binary number 10010101:

Subtrahend	0 0 1 1 0 1 1 0
One's complement	1 1 0 0 1 0 0 1
Two's complement	1 1 0 0 1 0 1 0

Minuend	1 0 0 1 0 1 0 1
Two's complement of subtrahend	+ 1 1 0 0 1 0 1 0
	1 0 1 0 1 1 1 1 1

Ignoring the leading 1 gives a result of 01011111.

It is usually much easier to use two's complements to perform subtraction than to use conventional subtraction (using borrow). Readers can easily check the validity of the statement by performing the foregoing example using conventional techniques!

Problem 1.16

Use the two's complement to subtract the binary number 00011001 from the binary number 11010100.

Representing Number Bases

So far we have been quite explicit in stating (rather long-windedly) the number base which we have been working in. Methods commonly used to indicate the base include:
(a) using a suffix to indicate the base
e.g. 101000_2 is a number having base 2 (i.e. binary)[2]
127_{10} is a number having base 10 (i.e. decimal)
76_{16} is a number having base 16 (i.e. Hexadecimal)
(b) adding a trailing H to indicate that a number is hexadecimal
e.g. 9FH, C9H and FFH are all hexadecimal
(c) adding a leading $ to indicate that a number is hexadecimal
e.g. $2A is the same as 2AH which is the same as 2A

Problem 1.17

Determine the decimal equivalent of each of the following:
(a) 10_2
(b) 10^2
(c) $\$10^{10}$

Glossary for Part One

Address
A number which indicates the position of a word in memory. Addresses typically comprise 16 bits and therefore can range from 0 to 65535

Address bus
A set of lines (usually 16) used to transmit addresses, usually from the microprocessor to a memory or I/O device.

Address decoding
The process of selecting a specific address or range of addresses to enable unique support devices.

Binary
A system of numbers using base 2. Binary numbers thus use only two characters; 0 and 1.

Bit
A contraction of binary digit. A single digit in a binary number.

Bus
A path of signals having some common function. Most microprocessor systems have three buses; an address bus, data bus and control bus.

Byte
A group of eight bits.

Central Processing Unit
The part of a computer that decodes instructions and controls the other hardware elements of the system. The CPU comprises a control unit, arithmetic/logic unit and internal storage.

Chip
The term commonly used to describe an integrated circuit.

Complement
The process of changing a 1 to a 0 and vice versa.

Control Bus
The set of control signal lines in a computer system. The control bus provides

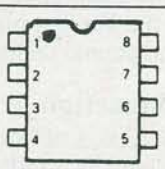


Fig. 6.

ANSWERS TO PROBLEMS

1.1 See Fig. 6.
1.2 (a) standard TTL (b) low-power Schottky (c) CMOS
1.3 CMOS (reasons are wide supply voltage range and very low current requirements)
1.4 See Fig. 7.

1.5 (a) 11	(b) 38	(c) 231
1.6 (a) 100001	(b) 1100100	(c) 11010101
1.7 (a) 1B	(b) 1FF	(c) 1004
1.8 (a) 190	(b) 267	(c) 49152
1.9 (a) B	(b) AF	(c) ABCD
1.10 (a) 111111	(b) 1000111010	(c) 1000000100101100
1.11 (a) 11010100	(b) CD	
1.12 (a) 1111000	(b) B4	
1.13 (a) +64	(b) −64	(c) −9
1.14 (a) 10000101	(b) 11100011	(c) 01100011
1.15 (a) 0110 and 0111	(b) 01001100 and 01001101	(c) 01110001 and 01110010
1.16 10111011		
1.17 (a) 2	(b) 10	(c) 16

the synchronization and control information to operate the system.

CMOS
Complementary metal oxide semiconductor. A family of integrated circuit devices based on unipolar field effect devices.

Data
General term used to describe numbers, letters and symbols present within a computer during processing.

Data Bus
The set of electrical conductors which carries data between the different elements of a computer system.

Digital
A system which only allows discrete states. Most digital logic uses only two states (on/off, low/high or 0/1).

Firmware
A program (software) stored in read-only memory.

Hardware
The physical components of a computer system.

Hexadecimal
A number system with base 16. The first six letters of the alphabet (A to F) are used to represent the hexadecimal equivalents of the decimal numbers 10 to 16.

Input/output
Lines of devices used to transfer information outside the computer system.

Input port
A circuit that connects signals from external devices as inputs to a microcomputer system.

Integrated circuit
An electronic circuit fabricated on a single wafer (chip) and packaged as a single component.

Memory
The part of a computer system into which information can be placed and held for future use. Storage and memory are interchangeable terms. Digital memories accept and hold binary numbers only. Common types of memory are magnetic disc, magnetic tape, and semiconductor (which includes RAM and ROM)

Microcomputer
A small computer based upon a microprocessor CPU. Backing storage is usually provided by magnetic disc or tape; input is provided by means of a keyboard and output by a VDU.

Microprocessor
A central processing unit fabricated on one or two chips. The processor contains an arithmetic logic unit (ALU), control block and registers.

One's complement
The inverse of a binary number which is formed by changing all 0s to 1s and vice versa.

Output port
A circuit that allows a microprocessor system to output signals to other devices.

Peripheral
Any device that is connected to a computer whose activity is under the control of the central processing unit.

Program
A procedure of solving a problem coded into a form suitable for use by a computer and frequently described as software.

RAM (random access memory)
Usually used to mean semiconductor read/write memory. Strictly speaking, ROM devices are also random access!

Random access
An access method in which each word can be retrieved in the same amount of time (i.e. the storage locations can be accessed in any desired order).

Read
The process of transferring information from memory or I/O into a register in the central processor.

Register
A single word of memory. The CPU contains a number of registers for temporary storage of data.

ROM (read-only memory)
A permanently programmed memory. Mask-programmed ROMs are programmed by the chip manufacturer. PROMS (programmable ROM devices) can be programmed by the user. EPROMs (erasable PROM devices) can be erased under ultraviolet light before reprogramming.

TTL (transistor transistor logic)
Transistor transistor logic. A family of integrated circuit devices based on conventional bipolar transistors.

Two's complement numbers
A number system used to represent both positive and negative numbers. The positive numbers in two's complement representation are identical to the positive numbers in standard binary, however the two's complement representation of a negative number is the complement of the absolute binary value plus 1. Note that the most significant bit (usually the eight) indicates the sign; 0 = positive, 1 = negative).

Visual display unit (VDU)
A VDU is an output device (usually based on a cathode ray tube) on which text and/or graphics can be displayed. A VDU is normally used in conjunction with an integral keyboard when it is sometimes referred to as a console.

Word
A set of characters that occupies one storage location in memory and is treated by the computer as a unit. Program instructions and program data both have the same word length (equal to 8-bits or 1-byte in the case of 8-bit microprocessors)

Write
The process of transferring information from a register within the central processor to memory or I/O.

Sources For Microprocessor Training Materials.

Active, Division of Future Electronics
Carries the full Edukit line including the "Digital Designer" in teacher's version or a home-study version, and the Motorola MC68000 kit.
Locations across Canada or phone 1-800-361-5884 or1-800-361-9046 (Quebec and Ottawa Valley)

Radio Shack
Stocks a variety of laboratory-type kits dealing with electronics and microprocessors.
Radio Shack stores across Canada

ElectroSonic Inc.
They list a number of RCA and Motorola microprocessor evaluation kits in varying forms and prices.
1100 Gordon Baker Road, Willowdale, Ontario M2H 3B3 (416) 494-1555.

Tapto Corp.
Stocks a variety of microprocessor-oriented kits from Velleman of Belgium.
P.O. Box 44247, Denver, CO 80202-4247. (303) 296-5544

Labvolt
Distributes the Labvolt series of comprehensive microprocessor/logic training systems starting at $885.
Ontario - Harold Burrows, 200 Consumers Rd., Suite 200, Willowdale, Ontario M2J 4R4 Tel: (416) 491-8611
Quebec & Other - 4555 Metropolitain E., Suite 102, Montreal, Quebec H1R 1Z4 (514) 376-2120

Heath Company
Carries a wide range of microprocessor trainers. Available from Heath/Zenith Computers and Electronic Centres across Canada. Mail order from 1020 Islington Ave., Toronto, Ontario M8Z 5Z3.

Mastertech Laboratories
Microlab 1 Digital Trainer system. Contact Mastertech at: 302 Royal Trust Building, 612 View Street, Victoria, B.C. V8W 1T5 (604) 388-6631.

■