Appendixes

APPENDIX 1. THE ANALOG INTERFACE

The data in a microprocessor is in digital form. This differs from the outside world where data is in analog (continuous) form. To get digital data, we need to use an *analog-to-digital* (A/D) *converter;* it will convert analog voltage or current into an equivalent digital word.

Conversely, after a CPU has processed data, it is often necessary to convert the digital answer into an analog voltage or current. This conversion requires a *digital-to-analog* (D/A) converter.

The *analog interface* is the boundary where digital and analog meet, where the microcomputer connects to the outside world. At this interface, we find either an A/D converter (input side) or a D/A converter (output side). This chapter discusses some of the hardware and software found at the analog interface.

A1-1 OP-AMP BASICS

Let us briefly review the *operational amplifier* (op amp) because this device is used with D/A and A/D converters. We will zero in on the key features that make the op amp useful at the analog interface.

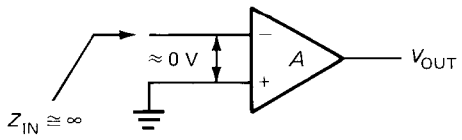


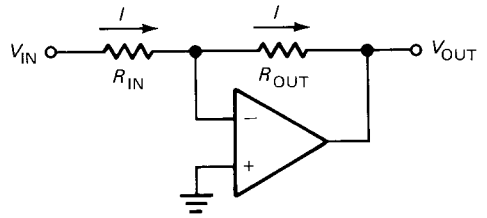
Fig. A1-1 Operational amplifier.

Virtual Ground

Figure A1-1 shows the symbol for an op amp. V_{OUT} is the output voltage with respect to ground. A is the open-loop voltage gain of the op amp, often more than 100,000. When connected as an inverter, the noninverting input (+ input) is grounded. The inverting input (− input) receives the signal voltage.

Because the voltage gain of an op amp is so large, the input voltage is in microvolts. To a first approximation, the input voltage may be treated as 0 V. Furthermore, the input impedance of the inverting input approaches infinity (sometimes FETs are used for the input stage, as in BIFET op amps). These key features, zero input voltage and infinite input impedance, make the inverting input a *virtual ground point.*

How is a virtual ground different from an ordinary ground? An ordinary ground has zero voltage while sinking any amount of current. A virtual ground, however, is a ground for voltage but not for current; it has zero voltage but can sink no current. In the discussion that follows, we will approximate the inverting input of an op amp as a virtual ground point: this means zero voltage and zero current.



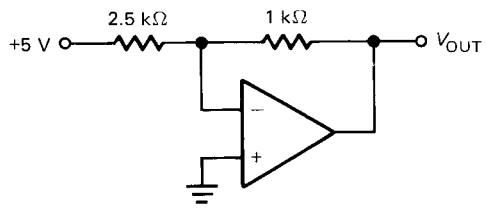


Fig. A1-2 Output current equals input current.

Output Voltage and Current

Figure A1-2*a* shows an inverting op amp with input and output resistors. V_{IN} is the input voltage with respect to ground, and V_{OUT} is the output voltage with respect to ground. Because of the high gain and input impedance, we

can approximate the inverting input as a virtual ground point. Therefore, all the input voltage appears across the input resistor, which means that the input current is

$$I = \frac{V_{IN}}{R_{IN}} \qquad \text{(A1-1)}$$

Since none of the input current can enter the virtual ground point, it must pass through the output resistor. In other words, the output current equals the input current. And the output voltage is

$$V_{OUT} = -IR_{OUT} \qquad \text{(A1-2)}$$

The minus sign indicates phase inversion. If the input voltage is positive, the output voltage is negative.

As an example of calculating input current and output voltage, look at Fig. A1-2*b*. The input current is

$$I = \frac{5\text{ V}}{2.5\text{ k}\Omega} = 2\text{ mA}$$

The output voltage is

$$V_{OUT} = -2\text{ mA} \times 1\text{ k}\Omega = -2\text{ V}$$

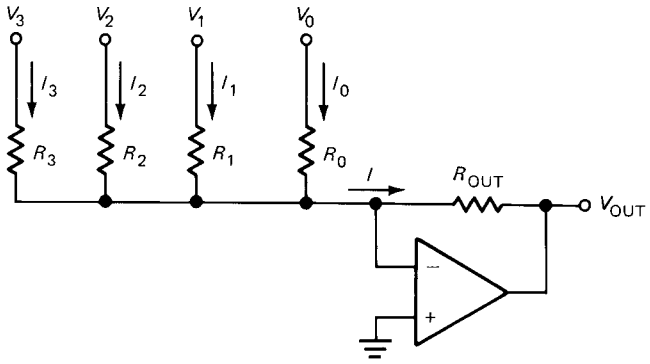


Fig. A1-3 Output current equals sum of input currents.

Summing Circuit

Figure A1-3 is an op-amp circuit whose output current is the *sum* of the input currents. Here is the proof. Because of the virtual ground point, each input voltage appears across its resistor. This means that the input currents are

$$I_3 = \frac{V_3}{R_3} \qquad I_2 = \frac{V_2}{R_2} \qquad I_1 = \frac{V_1}{R_1} \qquad I_0 = \frac{V_0}{R_0}$$

Kirchhoff's current law gives a total input current of

$$I = I_3 + I_2 + I_1 + I_0$$

Again, the virtual ground guarantees that all this input current goes through the output resistor. As before,

$$V_{OUT} = -IR_{OUT}$$

A1-2 A BASIC D/A CONVERTER

The op-amp summing circuit can be used to build a D/A converter by selecting input resistors that are weighted in binary progression. Figure A1-4 gives you the idea. V_{REF} is an accurate reference voltage, and the resistors are precision resistors to get accurate input currents. The switches can be open or closed. When all switches are open, all input currents are zero and the output current is zero.

All Bits High

When all switches are closed, the input currents are

$$I_3 = \frac{V_{REF}}{R} \qquad I_2 = \frac{V_{REF}}{2R} \qquad I_1 = \frac{V_{REF}}{4R} \qquad I_0 = \frac{V_{REF}}{8R}$$

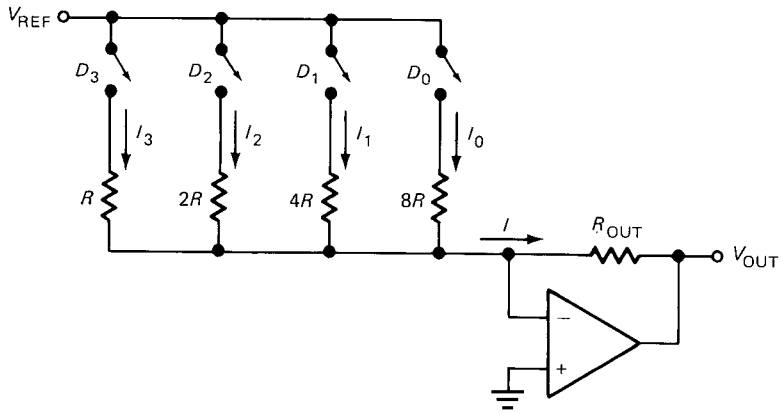


Fig. A1-4 D/A conversion with binary-weighted resistors.

The output current with all switches closed is the sum of all input currents and equals

$$I = \frac{V_{REF}}{R}(1 + 0.5 + 0.25 + 0.125) \qquad \text{(A1-3)}$$

$$I = 1.875\frac{V_{REF}}{R}$$

By opening and closing switches we can produce 16 different output currents from 0 to $1.875V_{REF}/R$.

Any Digital Input

If 0 stands for an open switch and 1 for a closed switch, we can rewrite Eq. A1-3 as

$$I = \frac{V_{REF}}{R}(D_3 + 0.5D_2 + 0.25D_1 + 0.125D_0) \qquad \text{(A1-4)}$$

In powers of 2,

$$I = \frac{V_{REF}}{R}(D_3 + 2^{-1}D_2 + 2^{-2}D_1 + 2^{-3}D_0) \qquad \text{(A1-5)}$$

This says that the output current is the sum of binary-weighted input currents. In other words, we have a D/A converter. For instance, suppose $V_{REF} = 5$ V and $R = 5$ kΩ. Then the total output current varies from 0 to 1.875 mA, as shown in Table A1-1.

TABLE A1-1. WEIGHTED D/A CONVERTER

D_3	D_2	D_1	D_0	Output current, mA	Fraction of maximum
0	0	0	0	0	0
0	0	0	1	0.125	$\frac{1}{15}$
0	0	1	0	0.25	$\frac{2}{15}$
0	0	1	1	0.375	$\frac{3}{15}$
0	1	0	0	0.5	$\frac{4}{15}$
0	1	0	1	0.625	$\frac{5}{15}$
0	1	1	0	0.75	$\frac{6}{15}$
0	1	1	1	0.875	$\frac{7}{15}$
1	0	0	0	1	$\frac{8}{15}$
1	0	0	1	1.125	$\frac{9}{15}$
1	0	1	0	1.25	$\frac{10}{15}$
1	0	1	1	1.375	$\frac{11}{15}$
1	1	0	0	1.5	$\frac{12}{15}$
1	1	0	1	1.625	$\frac{13}{15}$
1	1	1	0	1.75	$\frac{14}{15}$
1	1	1	1	1.875	$\frac{15}{15}$

Current Switches

Figure A1-5 shows how we can transistorize the switching. Data bits D_3 through D_0 drive the bases of the transistors through the current-limiting resistors. When a bit is high, it produces enough base current to saturate its transistor. When a bit is low, the transistor is cut off. Since each transistor is saturated or cut off, it acts like a closed or open switch. (Base resistance is not critical; it need only be less than collector resistance multiplied by β_{dc}.)

If the lower 4 bits of an output port are connected to D_3 to D_0, the circuit of Fig. A1-5 will convert digital data to analog current. For instance, assume port 22H has been programmed as an output port in a minimum system. If the lower 4 bits of port 22H are connected to D_3 to D_0, this program segment will operate the D/A converter:

Label	Mnemonic	Comment
	MVI A,FFH	;Initialize accumulator
LOOP:	INR A	;Count up
	OUT 22H	;Output nibble
	JMP LOOP	;Get next nibble

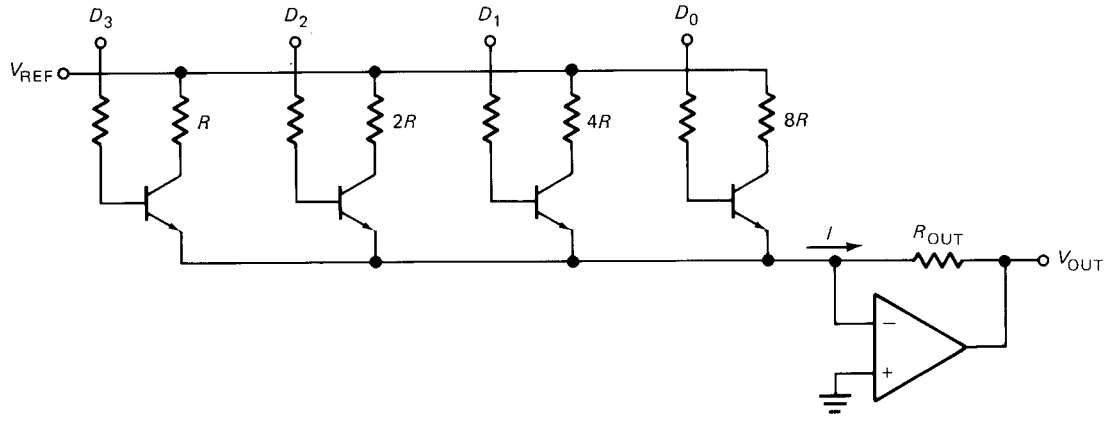


Fig. A1-5 Transistor switches for D/A converter.

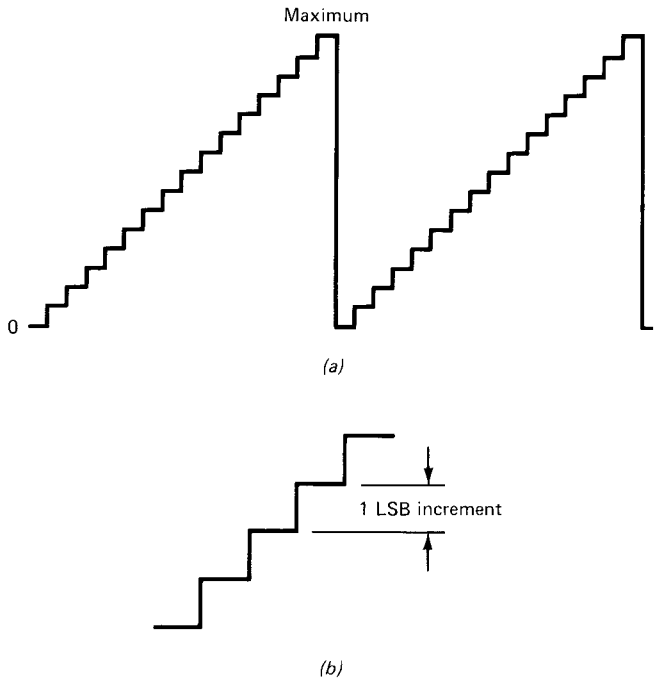


Fig. A1-6 (*a*) Staircase output current; (*b*) each step equals an LSB increment.

The first INR A produces accumulator contents of 00H. Subsequent INR executions produce 01H, 02H, . . . , 0FH, 10H, 11H, . . . , 1FH, 20H, 21H, . . . , FFH. As far as D_3 to D_0 are concerned, they see a nibble stream of 0000, 0001, 0010, 0011, . . . , 1111, 0000, 0001, and so on.

Figure A1-6*a* illustrates how the output current of the D/A converter appears. As each input nibble is latched into port 22H, the output current moves one step higher until reaching the maximum current. Then the cycle repeats. If all resistors are exact and all transistors matched, all steps are identical in size.

Resolution

In the perfect staircase of Fig. A1-6*b* a step is called an *LSB increment* because it is produced by a change in the LSB. One way to measure the quality of a D/A converter is its *resolution*, the ratio of the LSB increment to the maximum output. As a formula,

$$\text{Resolution} = \frac{1}{2^n - 1} \qquad \text{(A1-6)}$$

For instance, a 4-bit D/A converter has a resolution of

$$\text{Resolution} = \frac{1}{2^4 - 1} = \frac{1}{15}$$

This is sometimes read as 1 part in 15.

The number of different steps an n-bit converter produces is

$$\text{Steps} = 2^n - 1 \qquad \text{(A1-6}a\text{)}$$

Therefore, an alternative way to think of resolution is

$$\text{Resolution} = \frac{1}{\text{steps}} \qquad \text{(A1-6}b\text{)}$$

Percent resolution is given by

$$\text{Percent resolution} = \text{resolution} \times 100\% \qquad \text{(A1-7)}$$

If the resolution is 1 part in 15, then

$$\text{Percent resolution} = \tfrac{1}{15} \times 100\% = 6.67\%$$

The greater the number of bits, the better the resolution. With Eqs. A1-6 and A1-7 we can calculate the resolution and percent resolution for more bits. Table A1-2 is a summary of the resolution for converters with 4 to 18 bits.

Because the number of bits determines the resolution in Eq. A1-6, an indirect way to specify resolution is by stating the number of bits. For instance, an 8-bit converter has 8-bit resolution, a 10-bit converter has 10-bit resolution, and so on. This is a quick and easy way to pin down the resolution. When necessary, Eqs. A1-6, A1-6*a*, and A1-7 can give additional information.

Accuracy

In a D/A converter, *absolute accuracy* refers to how close each output current is to its ideal value. In Fig. A1-5 absolute accuracy depends on the reference voltage, resistor tolerance, transistor mismatch, and so forth. In a typical application, a trimmer adjustment is included to set the full-scale output at a preassigned value.

Relative accuracy refers to how close each output level is to its ideal fraction of full-scale output. With a 4-bit

TABLE A1-2. RESOLUTION

Bits	Resolution	Percent
4	1 part in 15	6.67
6	1 part in 63	1.59
8	1 part in 255	0.392
10	1 part in 1,023	0.0978
12	1 part in 4,095	0.0244
14	1 part in 16,383	0.0061
16	1 part in 65,535	0.00153
18	1 part in 262,143	0.000381

converter, the ideal output levels as a fraction of full-scale should be 0, $\frac{1}{15}$, $\frac{2}{15}$, $\frac{3}{15}$, and so on. Because data sheets specify relative accuracy rather than absolute accuracy, our subsequent discussions will emphasize relative accuracy.

Relative accuracy depends mainly on the tolerance of the weighted resistors in Fig. A1-5. If they are exactly R, $2R$, $4R$, and $8R$, all steps equal 1 LSB increment in Fig. A1-6*a*. When the resistors depart from ideal values, the steps may be larger or smaller than 1 LSB increment.

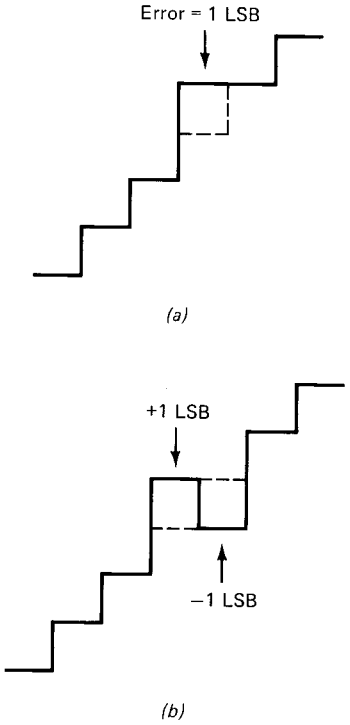


Fig. A1-7 Error specified in LSB increments.

Errors are specified in terms of LSB increments. For instance, Fig. A1-7*a* shows an error of 1 LSB; the actual output (solid line) differs from the ideal output (dashed line) by 1 LSB increment. If a negative error follows a positive error, the staircase can fall as shown in Fig. A1-7*b*. Here you see an error of +1 LSB followed by an error of −1 LSB.

Monotonicity

A *monotonic D/A converter* is one that produces an increase in output current for each successive digital input. The staircases of Fig. A1-7*a* and *b* are not monotonic because they do not produce an increase for each digital input. Figure A1-7*a* is almost monotonic, but Fig. A1-7*b* is far from monotonic. Monotonicity is the least we can expect from a D/A converter because it only makes sense; the output should increase when the input does.

For a D/A converter to be monotonic the error must be less than $\pm\frac{1}{2}$ LSB at each output level. Why? Because in the worst case, a $+\frac{1}{2}$-LSB error followed by a $-\frac{1}{2}$-LSB error produces the critical level where monotonicity is about to be lost. Figure A1-8 illustrates this critical case, an error of $+\frac{1}{2}$ LSB followed by an error of $-\frac{1}{2}$ LSB. If the error of a converter is less than $\pm\frac{1}{2}$ LSB for each output level, we are guaranteed a rising current for each successive digital input. Almost all commercially available D/A converters are monotonic because they have an accuracy of better than $\pm\frac{1}{2}$ LSB at each output level.

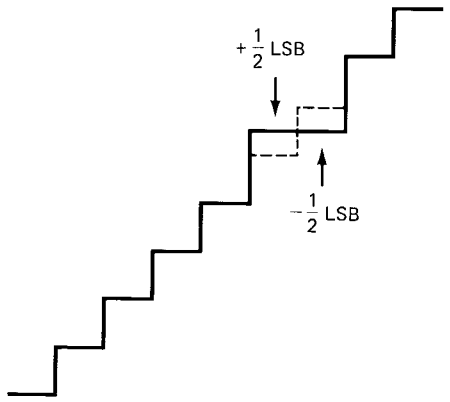


Fig. A1-8 Critical level for monotonicity.

Settling Time

After you apply a digital input, it takes a D/A converter anywhere from nanoseconds to microseconds to produce the correct output. *Settling time* is defined as the time it takes for the converter output to stabilize to within $\frac{1}{2}$ LSB of its final value. This time depends on the stray capacitance, saturation delay time, and other factors. Settling time is important because it places a limit on how fast you can change the digital inputs.

Disadvantages of Weighted Resistors

For a weighted-resistor circuit to be monotonic the tolerance of the resistors must be less than the percent resolution. For instance, if the resolution is $\frac{1}{15}$ (6.67 percent), resistors with a tolerance of less than ±6.67 percent will produce a monotonic staircase. If the resolution is $\frac{1}{255}$ (about 0.4 percent), the resistors need a tolerance of better than ±0.4 percent for a monotonic output. As you see, 4 bits are no problem, but 8 bits are.

Another difficulty arises with weighted resistors. As the number of bits increases, the range of resistance values gets awkward. For 8 bits, we need resistances of R, $2R$, $4R$, . . . , $128R$. The largest resistance is 128 times the smallest. For a 12-bit converter, the largest resistance needs to be 2,048 times the smallest. Because of the tolerance and range problems, mass production of weighted-resistor D/A converters is impractical.

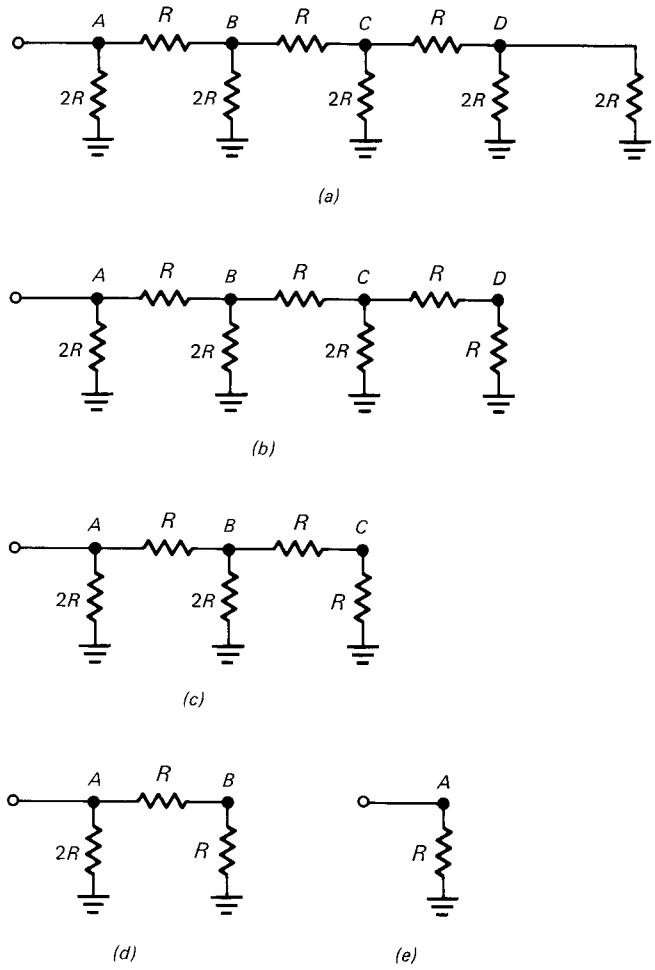


Fig. A1-9 *R-2R* ladder.

A1-3 THE LADDER METHOD

One way to get around the problems of a binary-weighted resistors is to use a *ladder* circuit. Figure A1-9*a* is an example of the *R-2R* ladder commonly used in integrated D/A converters. Only two resistance values are needed; this eliminates the range problem. Furthermore, since the resistors are on the same chip, they have almost identical characteristics; this minimizes the tolerance problem. In other words, as the number of bits increases, an integrated ladder can divide the current much more accurately than a binary-weighted circuit.

Ladder Properties

An *R-2R* ladder does something interesting to the impedance at different points in the circuit. To begin with, the two resistors at node D in Fig. A1-9*a* are in parallel and may be reduced to an equivalent resistance *R*, shown in Fig. A1-9*b*. Now, to the right of node C we have *R* in series with *R*, a total of 2*R*. Since node C has 2*R* is in parallel with 2*R*, the circuit reduces to Fig. A1-9*c*.

Looking into the left side of node B (Fig. A1-9*c*), we see 2*R* in parallel with 2*R*. Therefore, the circuit reduces to Fig. A1-9*d*. Again, 2*R* is in parallel with 2*R*, so the circuit reduces to the single *R* shown in Fig. A1-9*e*.

Figure A1-10 summarizes ladder impedances. Do you see the point? Looking into the left side of a node, we always see an equivalent resistance of *R*. Just to the right of each node, we always see a resistance of 2*R*. This impedance phenomenon is the key to analyzing modern D/A converters because they use the ladders instead of weighted resistors.

Binary Division of Current

Figure A1-11 shows how a ladder can divide the current into binary levels. The typical D/A converter has a reference current set by the user. In this example, the reference current is 2 mA. The bottom of each 2*R* resistor is grounded in either switch position. When a switch is to the right, the current through a 2*R* resistor flows to the upper ground. When a switch is to the left, the lower ground sinks the current. With all the switches to the right, as shown in Fig. A1-11, I_{OUT} is zero.

Here is how the ladder divides the 2 mA of reference current. Just to the right of node *A* we see an equivalent resistance of 2*R*. Therefore, the 2 mA of input current divides equally at node *A*. Similarly, at node *B* we see 2*R* in parallel with 2*R*; again, the current divides equally into 0.5-mA branch currents. This process continues through the ladder, so that we wind up with the upper grounds sinking 1, 0.5, 0.25, and 0.125 mA.

Other Switch Positions

When we move the switches, we do not change the way the current divides at the nodes. It still divides equally at each node. But when a switch is to the left, it steers the

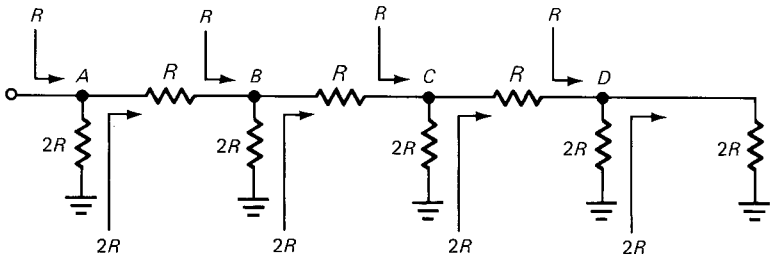


Fig. A1-10 Ladder impedances.

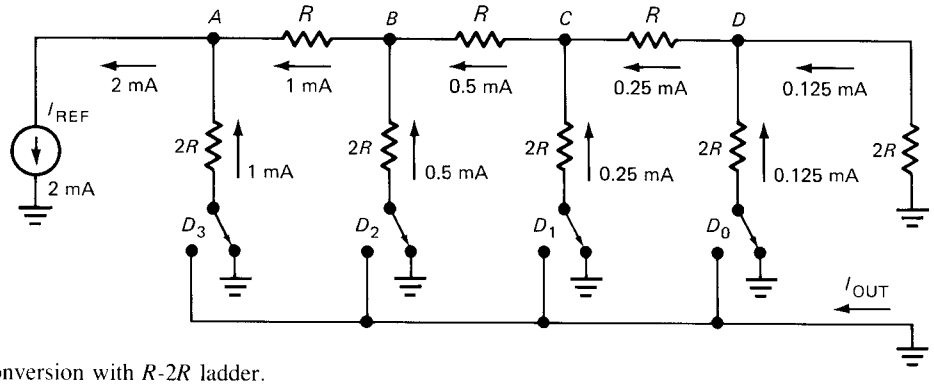


Fig. A1-11 D/A conversion with R-$2R$ ladder.

current into the lower ground. Bits D_3 to D_0 control the transistorized switches. From previous discussions, we can see that

$$I_{OUT} = (D_3 + 2^{-1}D_2 + 2^{-2}D_1 + 2^{-3}D_0)\frac{I_{REF}}{2} \quad \text{(A1-8)}$$

Therefore, the output current of a 4-bit ladder is from 0 to $\frac{15}{16}I_{REF}$.

More Bits

A similar analysis applies to longer ladders. The output current is

$$I_{OUT} = (D_{n-1} + 2^{-1}D_{n-2} + \cdots + 2^{1-n}D_0)\frac{I_{REF}}{2} \quad \text{(A1-9)}$$

For instance, an 8-bit ladder produces a maximum output current of $\frac{255}{256}I_{REF}$. The LSB increment is $\frac{1}{255}I_{REF}$.

Why Steer Current

Current steering may seem more complicated than necessary, but there is good reason for it. The currents throughout the ladder remain constant; all that changes are the ground points. Constant current implies constant voltage, which means that stray capacitance in the ladder has little effect. In other words, we do not get the usual exponential charge and discharge associated with a change in voltage. This reduces the settling time. For this reason, IC converters often use the current-steering approach shown in Fig. A1-11.

A1-4 THE COUNTER METHOD OF A/D CONVERSION

Figure A1-12 shows the simplest but least used method of A/D conversion. V_{IN} is the analog input voltage. D_7 to D_0 are the digital output. The digital output drives a D/A converter, which produces an analog output V_{OUT}. When *COUNT* is high, the counter counts upward. When *COUNT* is low, the counter stops. For convenience, an 8-bit D/A converter and 8-bit counter are used, but the idea applies to any number of bits.

Operation

The A/D conversion takes place as follows. First, the *START* pulse goes low, clearing the counter. When the

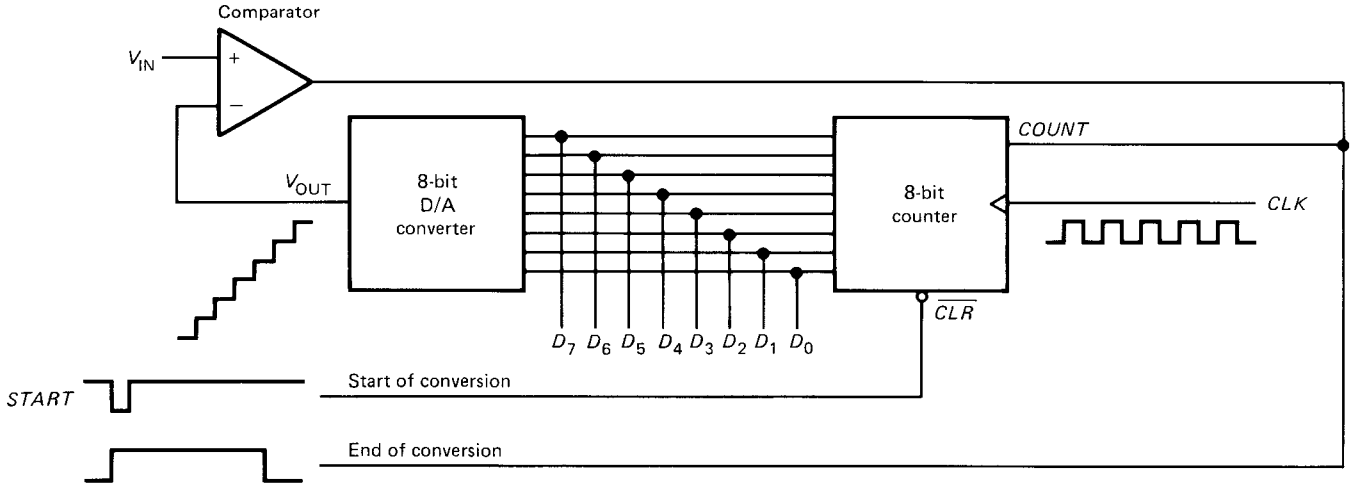


Fig. A1-12 A/D conversion with counter.

START pulse returns high, the counter is ready to go. Initially, V_{OUT} is zero; therefore, the op amp has a high output and *COUNT* is high. The counter starts counting upward from zero. Since the output of the counter drives a D/A converter, the converter output is a positive voltage staircase. As long as V_{IN} is greater than V_{OUT}, the op amp has a positive output, *COUNT* remains high, and the staircase voltage keeps rising.

At some point along the staircase, the next step makes V_{OUT} greater than V_{IN}. This forces *COUNT* to go low, and the counter stops. Now, the digital output D_7 to D_0 is the digital equivalent of the analog input. The negative-going edge of the *COUNT* signal is used as an *end-of-conversion* signal; this tells other circuits that the A/D conversion is finished.

If the analog input V_{IN} is changed, external circuits must send another *START* pulse to *start the conversion*. This clears the count and a new cycle begins. When the digital data is ready, the end-of-conversion signal has a falling edge.

Disadvantage

The main disadvantage of the *counter method* is its slow speed. In the worst case (maximum analog input) the counter has to reach the maximum count before the staircase voltage is greater than the analog input. For an 8-bit converter, this means a conversion time of 255 clock periods. For a 12-bit converter, the conversion time is 4,095 clock periods.

A1-5 SUCCESSIVE APPROXIMATION

The most widely used approach in A/D conversion is the *successive-approximation method* (see Fig. A1-13). As before, the output of a D/A converter drives the inverting input of an op-amp comparator. The difference, however, is in how the SAR register converges on the digital equivalent. (SAR stands for *successive-approximation register.*) When the conversion is finished, the digital equivalent is transferred to the output buffer register.

MSB First

When the start-of-conversion signal goes low, the SAR register is cleared and V_{OUT} drops to zero. When the start-of-conversion signal goes high, the conversion begins. Instead of counting up 1 bit at a time, the successive-approximation method starts by setting the MSB. In other words, during the first clock pulse the control circuit loads a high MSB into the SAR register, whose output then equals

1000 0000

As soon as this digital output appears, V_{OUT} jumps to $\frac{128}{255}$ times full-scale. If this is more than V_{IN}, the negative output of the comparator signals the control circuit to reset the MSB. On the other hand, if V_{OUT} is less than V_{IN}, the positive output of the comparator indicates that the MSB is to remain set. In some designs, setting and testing the MSB take place during the first clock pulse following the start of conversion. In other designs, several clock pulses may be needed to set the MSB, test it, and reset it if necessary.

Remaining Bits

Let us assume that the MSB was not reset. The SAR register contents are now 1000 0000. The next clock pulse will set

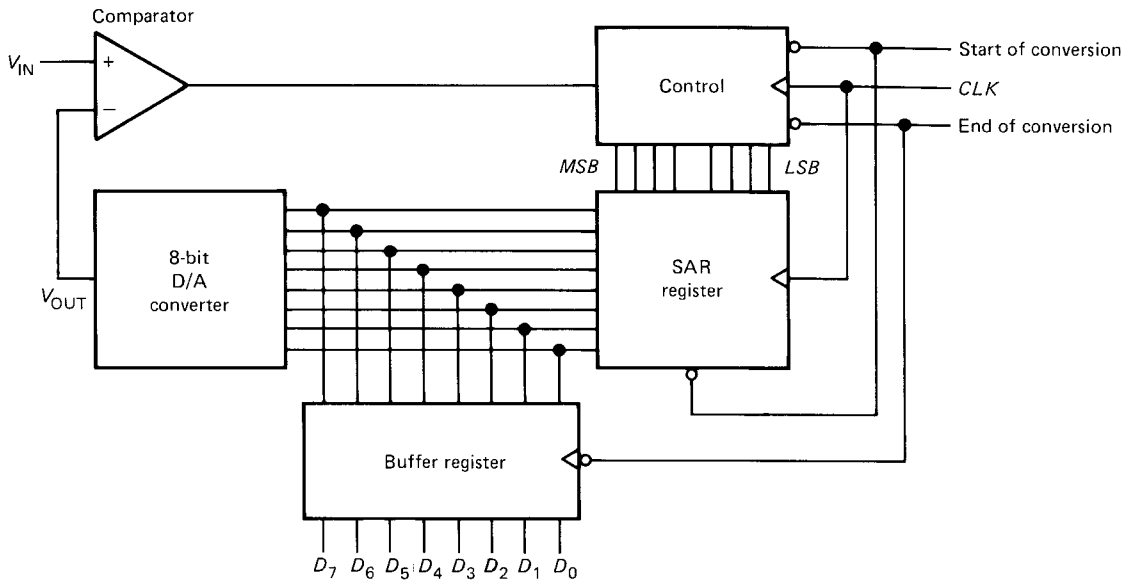


Fig. A1-13 A/D conversion by successive approximation.

D_6, giving a digital output of

1100 0000

V_{OUT} now steps to $\frac{192}{255}$ times full-scale. If V_{OUT} is greater than V_{IN}, the negative op-amp output causes D_6 to reset. If V_{OUT} is less than V_{IN}, D_6 remains set.

During the remaining clock pulses, successive bits are set and tested. Whenever a bit causes V_{OUT} to exceed V_{IN}, the bit is reset. In this way, all bits are set, tested, and reset if necessary. With the fastest circuits, the conversion is finished after eight clock pulses, and the D/A output is the analog equivalent of the register contents. Slower designs take longer because more clock pulses are needed to set, test, and possibly reset each bit.

Output Buffer

When the conversion is finished, the control circuit sends out a low end-of-conversion signal. The falling edge of this signal loads the digital equivalent into the buffer register. In this way, the digital output will remain even though we start a new conversion cycle.

Advantage

The main advantage of the successive-approximation method is speed. At best, it takes only n clock pulses to produce n-bit resolution of the analog signal. This is a big improvement over the counter method. Even with slower designs, the successive-approximation method is still considerably better than the counter method.

APPENDIX 2. BINARY-HEXADECIMAL-DECIMAL EQUIVALENTS

Binary	Hexadecimal	UB Decimal	LB Decimal
0000 0000	00	0	0
0000 0001	01	256	1
0000 0010	02	512	2
0000 0011	03	768	3
0000 0100	04	1,024	4
0000 0101	05	1,280	5
0000 0110	06	1,536	6
0000 0111	07	1,792	7
0000 1000	08	2,048	8
0000 1001	09	2,304	9
0000 1010	0A	2,560	10
0000 1011	0B	2,816	11
0000 1100	0C	3,072	12
0000 1101	0D	3,328	13
0000 1110	0E	3,584	14
0000 1111	0F	3,840	15
0001 0000	10	4,096	16
0001 0001	11	4,352	17
0001 0010	12	4,608	18
0001 0011	13	4,864	19
0001 0100	14	5,120	20
0001 0101	15	5,376	21
0001 0110	16	5,632	22
0001 0111	17	5,888	23
0001 1000	18	6,144	24
0001 1001	19	6,400	25
0001 1010	1A	6,656	26
0001 1011	1B	6,912	27
0001 1100	1C	7,168	28
0001 1101	1D	7,424	29
0001 1110	1E	7,680	30
0001 1111	1F	7,936	31
0010 0000	20	8,192	32
0010 0001	21	8,448	33
0010 0010	22	8,704	34
0010 0011	23	8,960	35
0010 0100	24	9,216	36
0010 0101	25	9,472	37
0010 0110	26	9,728	38
0010 0111	27	9,984	39
0010 1000	28	10,240	40
0010 1001	29	10,496	41
0010 1010	2A	10,752	42
0010 1011	2B	11,008	43
0010 1100	2C	11,264	44
0010 1101	2D	11,520	45
0010 1110	2E	11,776	46
0010 1111	2F	12,032	47
0011 0000	30	12,288	48
0011 0001	31	12,544	49
0011 0010	32	12,800	50
0011 0011	33	13,056	51
0011 0100	34	13,312	52
0011 0101	35	13,568	53
0011 0110	36	13,824	54
0011 0111	37	14,080	55
0011 1000	38	14,336	56
0011 1001	39	14,592	57
0011 1010	3A	14,848	58
0011 1011	3B	15,104	59
0011 1100	3C	15,360	60
0011 1101	3D	15,616	61
0011 1110	3E	15,872	62
0011 1111	3F	16,128	63
0100 0000	40	16,384	64
0100 0001	41	16,640	65
0100 0010	42	16,896	66
0100 0011	43	17,152	67
0100 0100	44	17,408	68
0100 0101	45	17,664	69
0100 0110	46	17,920	70
0100 0111	47	18,176	71
0100 1000	48	18,432	72
0100 1001	49	18,688	73
0100 1010	4A	18,944	74
0100 1011	4B	19,200	75
0100 1100	4C	19,456	76
0100 1101	4D	19,712	77
0100 1110	4E	19,968	78
0100 1111	4F	20,224	79
0101 0000	50	20,480	80
0101 0001	51	20,736	81
0101 0010	52	20,992	82
0101 0011	53	21,248	83
0101 0100	54	21,504	84
0101 0101	55	21,760	85
0101 0110	56	22,016	86
0101 0111	57	22,272	87
0101 1000	58	22,528	88
0101 1001	59	22,784	89
0101 1010	5A	23,040	90
0101 1011	5B	23,296	91
0101 1100	5C	23,552	92
0101 1101	5D	23,808	93
0101 1110	5E	24,064	94
0101 1111	5F	24,320	95

Binary	Hexadecimal	UB Decimal	LB Decimal	Binary	Hexadecimal	UB Decimal	LB Decimal
0110 0000	60	24,576	96	1001 0010	92	37,376	146
0110 0001	61	24,832	97	1001 0011	93	37,632	147
0110 0010	62	25,088	98	1001 0100	94	37,888	148
0110 0011	63	25,344	99	1001 0101	95	38,144	149
0110 0100	64	25,600	100	1001 0110	96	38,400	150
0110 0101	65	25,856	101	1001 0111	97	38,656	151
0110 0110	66	26,112	102	1001 1000	98	38,912	152
0110 0111	67	26,368	103	1001 1001	99	39,168	153
0110 1000	68	26,624	104	1001 1010	9A	39,424	154
0110 1001	69	26,880	105	1001 1011	9B	39,680	155
0110 1010	6A	27,136	106	1001 1100	9C	39,936	156
0110 1011	6B	27,392	107	1001 1101	9D	40,192	157
0110 1100	6C	27,648	108	1001 1110	9E	40,448	158
0110 1101	6D	27,904	109	1001 1111	9F	40,704	159
0110 1110	6E	28,160	110	1010 0000	A0	40,960	160
0110 1111	6F	28,416	111	1010 0001	A1	41,216	161
0111 0000	70	28,672	112	1010 0010	A2	41,472	162
0111 0001	71	28,928	113	1010 0011	A3	41,728	163
0111 0010	72	29,184	114	1010 0100	A4	41,984	164
0111 0011	73	29,440	115	1010 0101	A5	42,240	165
0111 0100	74	29,696	116	1010 0110	A6	42,496	166
0111 0101	75	29,952	117	1010 0111	A7	42,752	167
0111 0110	76	30,208	118	1010 1000	A8	43,008	168
0111 0111	77	30,464	119	1010 1001	A9	43,264	169
0111 1000	78	30,720	120	1010 1010	AA	43,520	170
0111 1001	79	30,976	121	1010 1011	AB	43,776	171
0111 1010	7A	31,232	122	1010 1100	AC	44,032	172
0111 1011	7B	31,488	123	1010 1101	AD	44,288	173
0111 1100	7C	31,744	124	1010 1110	AE	44,544	174
0111 1101	7D	32,000	125	1010 1111	AF	44,800	175
0111 1110	7E	32,256	126	1011 0000	B0	45,056	176
0111 1111	7F	32,512	127	1011 0001	B1	45,312	177
1000 0000	80	32,768	128	1011 0010	B2	45,568	178
1000 0001	81	33,024	129	1011 0011	B3	45,824	179
1000 0010	82	33,280	130	1011 0100	B4	46,080	180
1000 0011	83	33,536	131	1011 0101	B5	46,336	181
1000 0100	84	33,792	132	1011 0110	B6	46,592	182
1000 0101	85	34,048	133	1011 0111	B7	46,848	183
1000 0110	86	34,304	134	1011 1000	B8	47,104	184
1000 0111	87	34,560	135	1011 1001	B9	47,360	185
1000 1000	88	34,816	136	1011 1010	BA	47,616	186
1000 1001	89	35,072	137	1011 1011	BB	47,872	187
1000 1010	8A	35,328	138	1011 1100	BC	48,128	188
1000 1011	8B	35,584	139	1011 1101	BD	48,384	189
1000 1100	8C	35,840	140	1011 1110	BE	48,640	190
1000 1101	8D	36,096	141	1011 1111	BF	48,896	191
1000 1110	8E	36,352	142	1100 0000	C0	49,152	192
1000 1111	8F	36,608	143	1100 0001	C1	49,408	193
1001 0000	90	36,864	144	1100 0010	C2	49,664	194
1001 0001	91	37,120	145	1100 0011	C3	49,920	195

APPENDIX 2. BINARY-HEXADECIMAL-DECIMAL EQUIVALENTS (*Continued*)

Binary	Hexadecimal	UB Decimal	LB Decimal	Binary	Hexadecimal	UB Decimal	LB Decimal
1100 0100	C4	50,176	196	1110 0010	E2	57,856	226
1100 0101	C5	50,432	197	1110 0011	E3	58,112	227
1100 0110	C6	50,688	198	1110 0100	E4	58,368	228
1100 0111	C7	50,944	199	1110 0101	E5	58,624	229
1100 1000	C8	51,200	200	1110 0110	E6	58,880	230
1100 1001	C9	51,456	201	1110 0111	E7	59,136	231
1100 1010	CA	51,712	202	1110 1000	E8	59,392	232
1100 1011	CB	51,968	203	1110 1001	E9	59,648	233
1100 1100	CC	52,224	204	1110 1010	EA	59,904	234
1100 1101	CD	52,480	205	1110 1011	EB	60,160	235
1100 1110	CE	52,736	206	1110 1100	EC	60,416	236
1100 1111	CF	52,992	207	1110 1101	ED	60,672	237
1101 0000	D0	53,248	208	1110 1110	EE	60,928	238
1101 0001	D1	53,504	209	1110 1111	EF	61,184	239
1101 0010	D2	53,760	210	1111 0000	F0	61,440	240
1101 0011	D3	54,016	211	1111 0001	F1	61,696	241
1101 0100	D4	54,272	212	1111 0010	F2	61,952	242
1101 0101	D5	54,528	213	1111 0011	F3	62,208	243
1101 0110	D6	54,784	214	1111 0100	F4	62,464	244
1101 0111	D7	55,040	215	1111 0101	F5	62,720	245
1101 1000	D8	55,296	216	1111 0110	F6	62,976	246
1101 1001	D9	55,552	217	1111 0111	F7	63,232	247
1101 1010	DA	55,808	218	1111 1000	F8	63,488	248
1101 1011	DB	56,064	219	1111 1001	F9	63,744	249
1101 1100	DC	56,320	220	1111 1010	FA	64,000	250
1101 1101	DD	56,576	221	1111 1011	FB	64,256	251
1101 1110	DE	56,832	222	1111 1100	FC	64,512	252
1101 1111	DF	57,088	223	1111 1101	FD	64,768	253
1110 0000	E0	57,344	224	1111 1110	FE	65,024	254
1110 0001	E1	57,600	225	1111 1111	FF	65,280	255

APPENDIX 3. 7400 SERIES TTL

Number	Function
7400	Quad 2-input NAND gates
7401	Quad 2-input NAND gates (open collector)
7402	Quad 2-input NOR gates
7403	Quad 2-input NOR gates (open collector)
7404	Hex inverters
7405	Hex inverters (open collector)
7406	Hex inverter buffer-driver
7407	Hex buffer-drivers
7408	Quad 2-input AND gates
7409	Quad 2-input AND gates (open collector)
7410	Triple 3-input NAND gates
7411	Triple 3-input AND gates
7412	Triple 3-input NAND gates (open collector)
7413	Dual Schmitt triggers
7414	Hex Schmitt triggers
7416	Hex inverter buffer-drivers
7417	Hex buffer-drivers
7420	Dual 4-input NAND gates
7421	Dual 4-input AND gates
7422	Dual 4-input NAND gates (open collector)
7423	Expandable dual 4-input NOR gates
7425	Dual 4-input NOR gates
7226	Quad 2-input TTL-MOS interface NAND gates
7427	Triple 3-input NOR gates
7428	Quad 2-input NOR buffer
7430	8-input NAND gate
7432	Quad 2-input OR gates
7437	Quad 2-input NAND buffers
7438	Quad 2-input NAND buffers (open collector)
7439	Quad 2-input NAND buffers (open collector)
7440	Dual 4-input NAND buffers
7441	BCD-to-decimal decoder–Nixie driver
7442	BCD-to-decimal decoder
7443	Excess 3-to-decimal decoder
7444	Excess Gray-to-decimal
7445	BCD-to-decimal decoder-driver
7446	BCD-to-seven segment decoder-drivers (30-V output)
7447	BCD-to-seven segment decoder-drivers (15-V output)
7448	BCD-to-seven segment decoder-drivers
7450	Expandable dual 2-input 2-wide AND-OR-INVERT gates
7451	Dual 2-input 2-wide AND-OR-INVERT gates
7452	Expandable 2-input 4-wide AND-OR gates
7453	Expandable 2-input 4-wide AND-OR-INVERT gates
7454	2-input 4-wide AND-OR-INVERT gates
7455	Expandable 4-input 2-wide AND-OR-INVERT gates
7459	Dual 2-3 input 2-wide AND-OR-INVERT gates
7460	Dual 4-input expanders
7461	Triple 3-input expanders
7462	2-2-3-3 input 4-wide expanders
7464	2-2-3-4 input 4-wide AND-OR-INVERT gates
7465	4-wide AND-OR-INVERT gates (open collector)
7470	Edge-triggered *JK* flip-flop
7472	*JK* master-slave flip-flop
7473	Dual *JK* master-slave flip-flop
7474	Dual *D* flip-flop
7475	Quad latch
7476	Dual *JK* master-slave flip-flop
7480	Gates full adder
7482	2-bit binary full adder
7483	4-bit binary full adder
7485	4-bit magnitude comparator
7486	Quad EXCLUSIVE-OR gate
7489	64-bit random-access read-write memory
7490	Decade counter
7491	8-bit shift register
7492	Divide-by-12 counter
7493	4-bit binary counter
7494	4-bit shift register
7495	4-bit right-shift–left-shift register
7496	5-bit parallel-in–parallel-out shift register
74100	4-bit bistable latch
74104	*JK* master-slave flip-flop
74105	*JK* master-slave flip-flop
74107	Dual *JK* master-slave flip-flop
74109	Dual *JK* positive-edge-triggered flip-flop
74116	Dual 4-bit latches with clear
74121	Monostable multivibrator
74122	Monostable multivibrator with clear
74123	Monostable multivibrator
74125	Three-state quad bus buffer
74126	Three-state quad bus buffer
74132	Quad Schmitt trigger
74136	Quad 2-input EXCLUSIVE-OR gate
74141	BCD-to-decimal decoder-driver
74142	BCD counter-latch-driver
74145	BCD-to-decimal decoder-driver
74147	10/4 priority encoder
74148	Priority encoder
74150	16-line-to-1-line multiplexer
74151	8-channel digital multiplexer
74152	8-channel data selector-multiplexer

APPENDIX 3. 7400 SERIES TTL (*Continued*)

Number	Function	Number	Function
74153	Dual 4/1 multiplexer	74190	Up-down decade counter
74154	4-line–to–16-line decoder-demultiplexer	74191	Synchronous binary up-down counter
74155	Dual 2/4 demultiplexer	74192	Binary up-down counter
74156	Dual 2/4 demultiplexer	74193	Binary up-down counter
74157	Quad 2/1 data selector	74194	4-bit directional shift register
74160	Decade counter with asynchronous clear	74195	4-bit parallel-access shift register
74161	Synchronous 4-bit counter	74196	Presettable decade counter
74162	Synchronous 4-bit counter	74197	Presettable binary counter
74163	Synchronous 4-bit counter	74198	8-bit shift register
74164	8-bit serial shift register	74199	8-bit shift register
74165	Parallel-load 8-bit serial shift register	74221	Dual one-shot Schmitt trigger
74166	8-bit shift register	74251	Three-state 8-channel multiplexer
74173	4-bit three-state register	74259	8-bit addressable latch
74174	Hex *F* flip-flop with clear	74276	Quad *JK* flip-flop
74175	Quad *D* flip-flop with clear	74279	Quad debouncer
74176	35-MHz presettable decade counter	74283	4-bit binary full adder with fast carry
74177	35-MHz presettable binary counter	74284	Three-state 4-bit multiplexer
74179	4-bit parallel-access shift register	74285	Three-state 4-bit multiplexer
74180	8-bit odd-even parity generator-checker	74365	Three-state hex buffers
74181	Arithmetic-logic unit	74366	Three-state hex buffers
74182	Look-ahead carry generator	74367	Three-state hex buffers
74184	BCD-to-binary converter	74368	Three-state hex buffers
74185	Binary-to-BCD converter	74390	Individual clocks with flip-flops
74189	Three-state 64-bit random-access memory	74393	Dual 4-bit binary counter

APPENDIX 4. PINOUTS AND FUNCTION TABLES

74LS83

The 74LS83 is a 4-bit full adder; the binary output is

$$\mathbf{S} = \mathbf{A} + \mathbf{B}$$

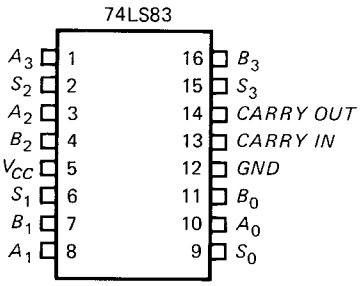


Fig. A4-1

In Fig. A4-1, pins 1, 3, 8, and 10 are the **A** input (A_3, A_2, A_1, A_0); pins 16, 4, 7, and 11 are the **B** input (B_3, B_2, B_1, B_0); and pins 15, 2, 6, and 9 are the **S** output (S_3, S_2, S_1, S_0). Pin 13 is the CARRY IN, and pin 14 is the CARRY OUT.

74LS157

This chip is a word multiplexer. Two words of 4 bits each are the inputs; one word of 4 bits is the output. The two input words are designated **L** (left) and **R** (right); the output word is **Y.** In Fig. A4-2, pin 1 (SELECT) and pin 15 (STROBE) are control inputs. The **L** word goes to pins 14, 11, 5, 2 (L_3, L_2, L_1, L_0), and the **R** word goes to pins 13, 10, 6, and 3 (R_3, R_2, R_1, R_0).

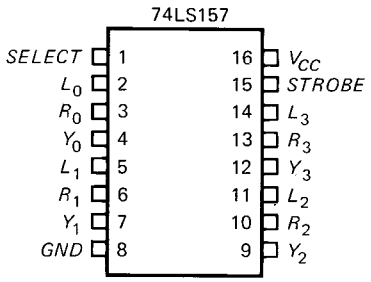


Fig. A4-2

TABLE A4-1. FUNCTION TABLE

STROBE	*SELECT*	Y	Comment
1	X	0	Output goes low
0	0	**L**	Output equals left word
0	1	**R**	Output equals right word

As indicated in Table A4-1, a high *STROBE* input produces a low output, no matter what the input words. When *STROBE* is low, the *SELECT* input controls the operation. A low *SELECT* will send the **L** word to the output; a high *SELECT* sends the **R** word to the output.

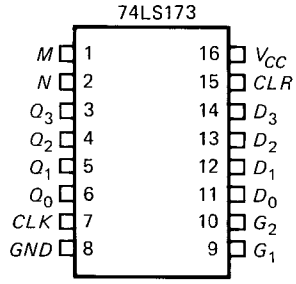


Fig. A4-3

74LS173

The 74LS173 is a 4-bit buffer register with three-state outputs. In Fig. A4-3, pins 14, 13, 12, and 11 are the data inputs (D_3, D_2, D_1, D_0). Pins 3, 4, 5, and 6 are the data outputs (Q_3, Q_2, Q_1, Q_0). Pins 9 and 10 (G_1 and G_2) are the input control. Pins 1 and 2 (M and N) are the output control.

As shown in Table A4-2, both *M* and *N* must be low to get a *Q* output. If either *M* or *N* (or both) is high, the output is three-stated (floating or high impedance).

When *M* and *N* are both low, Table A4-3 applies. As indicated, a high *CLEAR* will clear all *Q* bits to 0. When *CLEAR* is low, G_1 and G_2 control input loading. If either G_1 or G_2 (or both) are high, no change takes place in the *Q* bits. When both G_1 and G_2 are low, the next positive clock edge loads the input data.

TABLE A4-2. OUTPUT CONTROL

M	*N*	Output
0	0	Connected
0	1	Hi-Z
1	0	Hi-Z
1	1	Hi-Z

TABLE A4-3. FUNCTION TABLE FOR *M* = 0 AND *N* = 0

CLEAR	*CLOCK*	G_1	G_2	D_n	Q_n	Comment
1	X	X	X	X	0	Clear output
0	0	X	X	X	NC	No change
0	↑	1	X	X	NC	No change
0	↑	X	1	X	NC	No change
0	↑	0	0	0	0	Reset bit *n*
0	↑	0	0	1	1	Set bit *n*

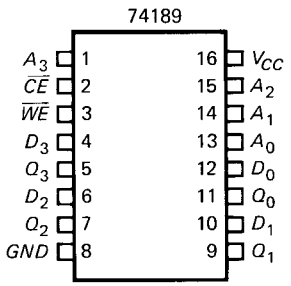


Fig. A4-4

74189

The 74189 is a 64-bit RAM organized as 16 words of 4 bits each. In Fig. A4-4 pins 1, 15, 14, and 13 are the address inputs (A_3, A_2, A_1, A_0). Pins 4, 6, 10, and 12 are the data inputs (D_3, D_2, D_1, D_0). Pins 5, 7, 9, and 11 are the data outputs (Q_3, Q_2, Q_1, Q_0).

TABLE A4-4. FUNCTION TABLE

$\overline{CE}$	$\overline{WE}$	Output	Comment
1	X	Hi-Z	Do nothing
0	0	Hi-Z	Write complement
0	1	Stored word	Read

Table A4-4 summarizes the operation of this read-write memory. When $\overline{CE}$ is high, the output is three-stated (high impedance). When $\overline{CE}$ is low and $\overline{WE}$ is low, the complement of the input data word is stored at the addressed memory location; during this write operation, the output is three-stated. When $\overline{CE}$ is low and $\overline{WE}$ is high, the stored word appears at the output.

APPENDIX 5. SAP-1 PARTS LIST

Chips

C1: 74LS107, dual *JK* master-slave flip-flop
C2: 74LS107
C3: 74LS126, quad three-state normally open switches
C4: 74LS173, buffer register, three-state outputs, 4 bits
C5: 74LS157, 2-to-1 nibble multiplexer
C6: 74189, 64-bit (16 × 4) static RAM, three-state outputs
C7: 74189
C8: 74LS173
C9: 74LS173
C10: 74LS173
C11: 74LS173
C12: 74LS126
C13: 74LS126
C14: 74LS86, quad 2-input EXCLUSIVE-OR gates
C15: 74LS86
C16: 74LS83, quad full adders
C17: 74LS83
C18: 74LS126
C19: 74LS126
C20: 74LS173
C21: 74LS173
C22: 74LS173
C23: 74LS173
C24: 7400, quad 2-input NAND gates
C25: 74LS10, triple 3-input NAND gates
C26: 74LS00
C27: 7404, hex inverter
C28: NE555, timer
C29: 74LS107
C30: LM340T-5, voltage regulator, 5 V
C31: 74LS04, hex inverter
C32: 74LS20, dual 4-input NAND gates
C33: 74LS20
C34: 74LS20
C35: 74LS04
C36: 74LS107
C37: 74LS107
C38: 74LS107
C39: 74LS00
C40: 74LS00
C41: 74LS00
C42: 74LS00
C43: 74LS00
C44: 74LS20
C45: 74LS10
C46: 74LS00
C47: 74LS04
C48: 74LS04

Diodes

D1: 1N4001, rectifier diode, 50 PIV, 1 A
D2: 1N4001
D3: 1N4001
D4: 1N4001

Switches

S1: SPST DIP switch, 4 bits
S2: DPST on-off
S3: SPST DIP, 8 bits
S4: SPST push button, momentary, normally open
S5: SPDT push button, momentary
S6: SPDT push button, momentary
S7: SPDT on-on switch

Miscellaneous

Resistors: eight 1-kΩ, fourteen 10-kΩ, one 18-kΩ, one 36-kΩ
Capacitors: 0.01-μF, 0.1-μF, 1000-μF (50 V)
Transformer: F-25X = 115 V primary, 12.6 V secondary CT, 1.5 A
Fuse: $\frac{3}{8}$-A slow blow

Totals

1N4001–4	74LS20–4
LM340T-5–1	74LS83–2
NE555–1	74LS86–2
7400–1	74LS107–6
74LS00–7	74LS126–5
7404–1	74LS157–1
74LS04–4	74LS173–9
74LS10–2	74189–2

Instruction	Op Code	*T* states	Flags	Main Effect
ACI byte	CE	7	All	A ← A + CY + byte
ADC A	8F	4	All	A ← A + A + CY
ADC B	88	4	All	A ← A + B + CY
ADC C	89	4	All	A ← A + C + CY
ADC D	8A	4	All	A ← A + D + CY
ADC E	8B	4	All	A ← A + E + CY
ADC H	8C	4	All	A ← A + H + CY
ADC L	8D	4	All	A ← A + L + CY
ADC M	8E	7	All	A ← A + M_{HL} + CY
ADD A	87	4	All	A ← A + A
ADD B	80	4	All	A ← A + B
ADD C	81	4	All	A ← A + C
ADD D	82	4	All	A ← A + D
ADD E	83	4	All	A ← A + E
ADD H	84	4	All	A ← A + H
ADD L	85	4	All	A ← A + L
ADD M	86	7	All	A ← A + M_{HL}
ADI byte	C6	7	All	A ← A + byte
ANA A	A7	4	All	A ← A AND A
ANA B	A0	4	All	A ← A AND B
ANA C	A1	4	All	A ← A AND C
ANA D	A2	4	All	A ← A AND D
ANA E	A3	4	All	A ← A AND E
ANA H	A4	4	All	A ← A AND H
ANA L	A5	4	All	A ← A AND L
ANA M	A6	7	All	A ← A AND M_{HL}
ANI byte	E6	7	All	A ← A AND byte
CALL address	CD	18	None	PC ← address
CC address	DC	18/9	None	PC ← address if $CY = 1$
CM address	FC	18/9	None	PC ← address if $S = 1$
CMA	2F	4	None	A ← $\overline{A}$
CMC	3F	4	CY	CY ← $\overline{CY}$
CMP A	BF	4	All	Z ← 1 if A = A
CMP B	B8	4	All	Z ← 1 if A = B
CMP C	B9	4	All	Z ← 1 if A = C
CMP D	BA	4	All	Z ← 1 if A = D
CMP E	BB	4	All	Z ← 1 if A = E
CMP H	BC	4	All	Z ← 1 if A = H
CMP L	BD	4	All	Z ← 1 if A = L
CMP M	BE	7	All	Z ← 1 if A = M_{HL}
CNC address	D4	18/9	None	PC ← address if $CY = 0$
CNZ address	C4	18/9	None	PC ← address if $Z = 0$
CP address	F4	18/9	None	PC ← address if $S = 0$
CPE address	EC	18/9	None	PC ← address if $P = 1$
CPI byte	FE	7	All	Z ← 1 if A = byte
CPO address	E4	18/9	None	PC ← address if $P = 0$
CZ address	CC	18/9	None	PC ← address if $Z = 1$
DAA	27	4	All	A ← BCD number
DAD B	09	10	CY	HL ← HL + BC
DAD D	19	10	CY	HL ← HL + DE
DAD H	29	10	CY	HL ← HL + HL

Instruction	Op Code	*T* states	Flags	Main Effect
DAD SP	39	10	CY	HL ← HL + SP
DCR A	3D	4	All but CY	A ← A − 1
DCR B	05	4	All but CY	B ← B − 1
DCR C	0D	4	All but CY	C ← C − 1
DCR D	15	4	All but CY	D ← D − 1
DCR E	1D	4	All but CY	E ← E − 1
DCR H	25	4	All but CY	H ← H − 1
DCR L	2D	4	All but CY	L ← L − 1
DCR M	35	10	All but CY	$M_{HL} \leftarrow M_{HL} - 1$
DCX B	0B	6	None	BC ← BC − 1
DCX D	1B	6	None	DE ← DE − 1
DCX H	2B	6	None	HL ← HL − 1
DCX SP	3B	6	None	SP ← SP − 1
DI	F3	4	None	Disable interrupts
EI	FB	4	None	Enable interrupts
HLT	76	5	None	Stop processing
IN byte	DB	10	None	A ← byte
INR A	3C	4	All but CY	A ← A + 1
INR B	04	4	All but CY	B ← B + 1
INR C	0C	4	All but CY	C ← C + 1
INR D	14	4	All but CY	D ← D + 1
INR E	1C	4	All but CY	E ← E + 1
INR H	24	4	All but CY	H ← H + 1
INR L	2C	4	All but CY	L ← L + 1
INR M	34	10	All but CY	$M_{HL} \leftarrow M_{HL} + 1$
INX B	03	6	None	BC ← BC + 1
INX D	13	6	None	DE ← DE + 1
INX H	23	6	None	HL ← HL + 1
INX SP	33	6	None	SP ← SP + 1
JC address	DA	10/7	None	PC ← address if $CY = 1$
JM address	FA	10/7	None	PC ← address if $S = 1$
JMP address	C3	10	None	PC ← address
JNC address	D2	10/7	None	PC ← address if $CY = 0$
JNZ address	C2	10/7	None	PC ← address if $Z = 0$
JP address	F2	10/7	None	PC ← address if $S = 0$
JPE address	EA	10/7	None	PC ← address if $P = 1$
JPO address	E2	10/7	None	PC ← address if $P = 0$
JZ address	CA	10/7	None	PC ← address if $Z = 1$
LDA address	3A	13	None	$A \leftarrow M_{adr}$
LDAX B	0A	7	None	$A \leftarrow M_{BC}$
LDAX D	1A	7	None	$A \leftarrow M_{DE}$
LHLD address	2A	16	None	$H \leftarrow M_{adr}$
LXI B, dble	01	10	None	BC ← dble
LXI D, dble	11	10	None	DE ← dble
LXI H, dble	21	10	None	HL ← dble
LXI SP, dble	31	10	None	SP ← dble
MOV A,A	7F	4	None	A ← A
MOV A,B	78	4	None	A ← B
MOV A,C	79	4	None	A ← C
MOV A,D	7A	4	None	A ← D
MOV A,E	7B	4	None	A ← E
MOV A,H	7C	4	None	A ← H

Instruction	Op Code	*T* states	Flags	Main Effect
MOV A,L	7D	4	None	A ← L
MOV A,M	7E	7	None	A ← M_{HL}
MOV B,A	47	4	None	B ← A
MOV B,B	40	4	None	B ← B
MOV B,C	41	4	None	B ← C
MOV B,D	42	4	None	B ← D
MOV B,E	43	4	None	B ← E
MOV B,H	44	4	None	B ← H
MOV B,L	45	4	None	B ← L
MOV B,M	46	7	None	B ← M_{HL}
MOV C,A	4F	4	None	C ← A
MOV C,B	48	4	None	C ← B
MOV C,C	49	4	None	C ← C
MOV C,D	4A	4	None	C ← D
MOV C,E	4B	4	None	C ← E
MOV C,H	4C	4	None	C ← H
MOV C,L	4D	4	None	C ← L
MOV C,M	4E	7	None	C ← M_{HL}
MOV D,A	57	4	None	D ← A
MOV D,B	50	4	None	D ← B
MOV D,C	51	4	None	D ← C
MOV D,D	52	4	None	D ← D
MOV D,E	53	4	None	D ← E
MOV D,H	54	4	None	D ← H
MOV D,L	55	4	None	D ← L
MOV D,M	56	7	None	D ← M_{HL}
MOV E,A	5F	4	None	E ← A
MOV E,B	58	4	None	E ← B
MOV E,C	59	4	None	E ← C
MOV E,D	5A	4	None	E ← D
MOV E,E	5B	4	None	E ← E
MOV E,H	5C	4	None	E ← H
MOV E,L	5D	4	None	E ← L
MOV E,M	5E	7	None	E ← M_{HL}
MOV H,A	67	4	None	H ← A
MOV H,B	60	4	None	H ← B
MOV H,C	61	4	None	H ← C
MOV H,D	62	4	None	H ← D
MOV H,E	63	4	None	H ← E
MOV H,H	64	4	None	H ← H
MOV H,L	65	4	None	H ← L
MOV H,M	66	7	None	H ← M_{HL}
MOV L,A	6F	4	None	L ← A
MOV L,B	68	4	None	L ← B
MOV L,C	69	4	None	L ← C
MOV L,D	6A	4	None	L ← D
MOV L,E	6B	4	None	L ← E
MOV L,H	6C	4	None	L ← H
MOV L,L	6D	4	None	L ← L
MOV L,M	6E	7	None	L ← M_{HL}
MOV M,A	77	7	None	M_{HL} ← A

Instruction	Op Code	*T* states	Flags	Main Effect
MOV M,B	70	7	None	$M_{HL} \leftarrow B$
MOV M,C	71	7	None	$M_{HL} \leftarrow C$
MOV M,D	72	7	None	$M_{HL} \leftarrow D$
MOV M,E	73	7	None	$M_{HL} \leftarrow E$
MOV M,H	74	7	None	$M_{HL} \leftarrow H$
MOV M,L	75	7	None	$M_{HL} \leftarrow L$
MVI A,byte	3E	7	None	A ← byte
MVI B,byte	06	7	None	B ← byte
MVI C,byte	0E	7	None	C ← byte
MVI D,byte	16	7	None	D ← byte
MVI E,byte	1E	7	None	E ← byte
MVI H,byte	26	7	None	H ← byte
MVI L,byte	2E	7	None	L ← byte
MVI M,byte	36	10	None	$M_{HL} \leftarrow$ byte
NOP	00	4	None	Delay
ORA A	B7	4	All	A ← A OR A
ORA B	B0	4	All	A ← A OR B
ORA C	B1	4	All	A ← A OR C
ORA D	B2	4	All	A ← A OR D
ORA E	B3	4	All	A ← A OR E
ORA H	B4	4	All	A ← A OR H
ORA L	B5	4	All	A ← A OR L
ORA M	B6	7	All	A ← A OR M_{HL}
ORI byte	F6	7	All	A ← A OR byte
OUT byte	D3	10	None	Port byte ← A
PCHL	E9	6	None	PC ← HL
POP B	C1	10	None	B ← M_{stk}
POP D	D1	10	None	D ← M_{stk}
POP H	E1	10	None	H ← M_{stk}
POP PSW	F1	10	None	F ← M_{stk}, A ← $M_{stk} - 1$
PUSH B	C5	12	None	$M_{stk} - 1 \leftarrow B$, $M_{stk} - 2 \leftarrow C$
PUSH D	D5	12	None	$M_{stk} - 1 \leftarrow D$, $M_{stk} - 2 \leftarrow E$
PUSH H	E5	12	None	$M_{stk} - 1 \leftarrow H$, $M_{stk} - 2 \leftarrow L$
PUSH PSW	F5	12	None	$M_{stk} - 1 \leftarrow A$, $M_{stk} - 2 \leftarrow F$
RAL	17	4	CY	Rotate all left
RAR	1F	4	CY	Rotate all right
RC	D8	12/6	None	PC ← return address if $CY = 1$
RET	C9	10	None	PC ← return address
RIM	20	4	None	A ← I
RLC	07	4	CY	Rotate left with carry
RM	F8	12/6	None	PC ← return address if $S = 1$
RNC	D0	12/6	None	PC ← return address if $CY = 0$
RNZ	C0	12/6	None	PC ← return address if $Z = 0$
RP	F0	12/6	None	PC ← return address if $S = 0$
RPE	E8	12/6	None	PC ← return address if $P = 1$
RPO	E0	12/6	None	PC ← return address if $P = 0$
RRC	0F	4	CY	Rotate right with carry
RST 0	C7	12	None	PC ← 0000H
RST 1	CF	12	None	PC ← 0008H
RST 2	D7	12	None	PC ← 0010H
RST 3	DF	12	None	PC ← 0018H
RST 4	E7	12	None	PC ← 0020H
RST 5	EF	12	None	PC ← 0028H

Instruction	Op Code	*T* states	Flags	Main Effect
RST 6	F7	12	None	PC ← 0030H
RST 7	FF	12	None	PC ← 0038H
RZ	C8	12/6	None	PC ← return address if $Z = 1$
SBB A	9F	4	All	A ← A − A − CY
SBB B	98	4	All	A ← A − B − CY
SBB C	99	4	All	A ← A − C − CY
SBB D	9A	4	All	A ← A − D − CY
SBB E	9B	4	All	A ← A − E − CY
SBB H	9C	4	All	A ← A − H − CY
SBB L	9D	4	All	A ← A − L − CY
SBB M	9E	7	All	A ← A − M − CY
SBI byte	DE	7	All	A ← A − byte − CY
SHLD address	22	16	None	M_{adr+1} ← H, M_{adr} ← L
SIM	30	4	None	I ← A
SPHL	F9	6	None	SP ← HL
STA address	32	13	None	M_{adr} ← A
STAX B	02	7	None	M_{BC} ← A
STAX D	12	7	None	M_{DE} ← A
STC	37	4	CY	CY ← 1
SUB A	97	4	All	A ← A − A
SUB B	90	4	All	A ← A − B
SUB C	91	4	All	A ← A − C
SUB D	92	4	All	A ← A − D
SUB E	93	4	All	A ← A − E
SUB H	94	4	All	A ← A − H
SUB L	95	4	All	A ← A − L
SUB M	96	7	All	A ← A − M
SUI byte	D6	7	All	A ← A − byte
XCHG	EB	4	None	HL ↔ DE
XRA A	AF	4	All	A ← A XOR A
XRA B	A8	4	All	A ← A XOR B
XRA C	A9	4	All	A ← A XOR C
XRA D	AA	4	All	A ← A XOR D
XRA E	AB	4	All	A ← A XOR E
XRA H	AC	4	All	A ← A XOR H
XRA L	AD	4	All	A ← A XOR L
XRA M	AE	7	All	A ← A XOR M
XRI byte	EE	7	All	A ← A XOR byte
XTHL	E3	16	None	HL ↔ stack

APPENDIX 7. MEMORY LOCATIONS: POWERS OF 2

Address Bits	Hexadecimal	Decimal	Power of 2
0000 0000 0000 0001	0001H	1	0
0000 0000 0000 0010	0002H	2	1
0000 0000 0000 0100	0004H	4	2
0000 0000 0000 1000	0008H	8	3
0000 0000 0001 0000	0010H	16	4
0000 0000 0010 0000	0020H	32	5
0000 0000 0100 0000	0040H	64	6
0000 0000 1000 0000	0080H	128	7
0000 0001 0000 0000	0100H	256	8
0000 0010 0000 0000	0200H	512	9
0000 0100 0000 0000	0400H	1,024	10
0000 1000 0000 0000	0800H	2,048	11
0001 0000 0000 0000	1000H	4,096	12
0010 0000 0000 0000	2000H	8,192	13
0100 0000 0000 0000	4000H	16,384	14
1000 0000 0000 0000	8000H	32,768	15

APPENDIX 8. MEMORY LOCATIONS: 16K AND 8K INTERVALS

Address Bits	Hexadecimal	Decimal	Zone
Zone bits = $A_{15}A_{14}$			
0000 0000 0000 0000	0000H	0	0
0011 1111 1111 1111	3FFFH	16,383	
0100 0000 0000 0000	4000H	16,384	1
0111 1111 1111 1111	7FFFH	32,767	
1000 0000 0000 0000	8000H	32,768	2
1011 1111 1111 1111	BFFFH	49,151	
1100 0000 0000 0000	C000H	49,152	3
1111 1111 1111 1111	FFFFH	65,535	
Zone bits = $A_{15}A_{14}A_{13}$			
0000 0000 0000 0000	0000H	0	0
0001 1111 1111 1111	1FFFH	8,191	
0010 0000 0000 0000	2000H	8,192	1
0011 1111 1111 1111	3FFFH	16,383	
0100 0000 0000 0000	4000H	16,384	2
0101 1111 1111 1111	5FFFH	24,575	
0110 0000 0000 0000	6000H	24,576	3
0111 1111 1111 1111	7FFFH	32,767	
1000 0000 0000 0000	8000H	32,768	4
1001 1111 1111 1111	9FFFH	40,959	
1010 0000 0000 0000	A000H	40,960	5
1011 1111 1111 1111	BFFFH	49,151	
1100 0000 0000 0000	C000H	49,152	6
1101 1111 1111 1111	DFFFH	57,343	
1110 0000 0000 0000	E000H	57,344	7
1111 1111 1111 1111	FFFFH	65,535	

APPENDIX 9. MEMORY LOCATIONS: 4K INTERVALS

Address Bits	Hexadecimal	Decimal	Zone
Zone bits = $A_{15}A_{14}A_{13}A_{12}$			
0000 0000 0000 0000	0000H	0	0
0000 1111 1111 1111	0FFFH	4,095	
0001 0000 0000 0000	1000H	4,096	1
0001 1111 1111 1111	1FFFH	8,191	
0010 0000 0000 0000	2000H	8,192	2
0010 1111 1111 1111	2FFFH	12,287	
0011 0000 0000 0000	3000H	12,288	3
0011 1111 1111 1111	3FFFH	16,383	
0100 0000 0000 0000	4000H	16,384	4
0100 1111 1111 1111	4FFFH	20,479	
0101 0000 0000 0000	5000H	20,480	5
0101 1111 1111 1111	5FFFH	24,575	
0110 0000 0000 0000	6000H	24,576	6
0110 1111 1111 1111	6FFFH	28,671	
0111 0000 0000 0000	7000H	28,672	7
0111 1111 1111 1111	7FFFH	32,767	
1000 0000 0000 0000	8000H	32,768	8
1000 1111 1111 1111	8FFFH	36,863	
1001 0000 0000 0000	9000H	36,864	9
1001 1111 1111 1111	9FFFH	40,959	
1010 0000 0000 0000	A000H	40,960	10
1010 1111 1111 1111	AFFFH	45,055	
1011 0000 0000 0000	B000H	45,056	11
1011 1111 1111 1111	BFFFH	49,151	
1100 0000 0000 0000	C000H	49,152	12
1100 1111 1111 1111	CFFFH	53,247	
1101 0000 0000 0000	D000H	53,248	13
1101 1111 1111 1111	DFFFH	57,343	
1110 0000 0000 0000	E000H	57,344	14
1110 1111 1111 1111	EFFFH	61,439	
1111 0000 0000 0000	F000H	61,440	15
1111 1111 1111 1111	FFFFH	65,535	

APPENDIX 10. MEMORY LOCATIONS: 2K INTERVALS

Address Bits	Hexadecimal	Decimal	Zone	Address Bits	Hexadecimal	Decimal	Zone
Zone bits = $A_{15}A_{14}A_{13}A_{12}A_{11}$							
0000 0000 0000 0000	0000H	0	0	1000 0000 0000 0000	8000H	32,768	16
0000 0111 1111 1111	07FFH	2,047		1000 0111 1111 1111	87FFH	34,815	
0000 1000 0000 0000	0800H	2,048	1	1000 1000 0000 0000	8800H	34,816	17
0000 1111 1111 1111	0FFFH	4,095		1000 1111 1111 1111	8FFFH	36,863	
0001 0000 0000 0000	1000H	4,096	2	1001 0000 0000 0000	9000H	36,864	18
0001 0111 1111 1111	17FFH	6,143		1001 0111 1111 1111	97FFH	38,911	
0001 1000 0000 0000	1800H	6,144	3	1001 1000 0000 0000	9800H	38,912	19
0001 1111 1111 1111	1FFFH	8,191		1001 1111 1111 1111	9FFFH	40,959	
0010 0000 0000 0000	2000H	8,192	4	1010 0000 0000 0000	A000H	40,960	20
0010 0111 1111 1111	27FFH	10,239		1010 0111 1111 1111	A7FFH	43,007	
0010 1000 0000 0000	2800H	10,240	5	1010 1000 0000 0000	A800H	43,008	21
0010 1111 1111 1111	2FFFH	12,287		1010 1111 1111 1111	AFFFH	45,055	
0011 0000 0000 0000	3000H	12,288	6	1011 0000 0000 0000	B000H	45,056	22
0011 0111 1111 1111	37FFH	14,335		1011 0111 1111 1111	B7FFH	47,103	
0011 1000 0000 0000	3800H	14,336	7	1011 1000 0000 0000	B800H	47,104	23
0011 1111 1111 1111	3FFFH	16,383		1011 1111 1111 1111	BFFFH	49,151	
0100 0000 0000 0000	4000H	16,384	8	1100 0000 0000 0000	C000H	49,152	24
0100 0111 1111 1111	47FFH	18,431		1100 0111 1111 1111	C7FFH	51,199	
0100 1000 0000 0000	4800H	18,432	9	1100 1000 0000 0000	C800H	51,200	25
0100 1111 1111 1111	4FFFH	20,479		1100 1111 1111 1111	CFFFH	53,247	
0101 0000 0000 0000	5000H	20,480	10	1101 0000 0000 0000	D000H	53,248	26
0101 0111 1111 1111	57FFH	22,527		1101 0111 1111 1111	D7FFH	55,295	
0101 1000 0000 0000	5800H	22,538	11	1101 1000 0000 0000	D800H	55,296	27
0101 1111 1111 1111	5FFFH	24,575		1101 1111 1111 1111	DFFFH	57,343	
0110 0000 0000 0000	6000H	24,576	12	1110 0000 0000 0000	E000H	57,344	28
0110 0111 1111 1111	67FFH	26,623		1110 0111 1111 1111	E7FFH	59,391	
0110 1000 0000 0000	6800H	26,624	13	1110 1000 0000 0000	E800H	59,392	29
0110 1111 1111 1111	6FFFH	28,671		1110 1111 1111 1111	EFFFH	61,439	
0111 0000 0000 0000	7000H	28,672	14	1111 0000 0000 0000	F000H	61,440	30
0111 0111 1111 1111	77FFH	30,719		1111 0111 1111 1111	F7FFH	63,487	
0111 1000 0000 0000	7800H	30,720	15	1111 1000 0000 0000	F800H	63,488	31
0111 1111 1111 1111	7FFFH	32,767		1111 1111 1111 1111	FFFFH	65,535	

APPENDIX 11. MEMORY LOCATIONS: 1K INTERVALS

Address Bits	Hexadecimal	Decimal	Zone	Address Bits	Hexadecimal	Decimal	Zone
Zone bits = $A_{15}A_{14}A_{13}A_{12}A_{11}A_{10}$							
0000 0000 0000 0000	0000H	0	0	0101 0000 0000 0000	5000H	20,480	20
0000 0011 1111 1111	03FFH	1,023		0101 0011 1111 1111	53FFH	21,503	
0000 0100 0000 0000	0400H	1,024	1	0101 0100 0000 0000	5400H	21,504	21
0000 0111 1111 1111	07FFH	2,047		0101 0111 1111 1111	57FFH	22,527	
0000 1000 0000 0000	0800H	2,048	2	0101 1000 0000 0000	5800H	22,528	22
0000 1011 1111 1111	0BFFH	3,071		0101 1011 1111 1111	5BFFH	23,551	
0000 1100 0000 0000	0C00H	3,072	3	0101 1100 0000 0000	5C00H	23,552	23
0000 1111 1111 1111	0FFFH	4,095		0101 1111 1111 1111	5FFFH	24,575	
0001 0000 0000 0000	1000H	4,096	4	0110 0000 0000 0000	6000H	24,576	24
0001 0011 1111 1111	13FFH	5,119		0110 0011 1111 1111	63FFH	25,599	
0001 0100 0000 0000	1400H	5,120	5	0110 0100 0000 0000	6400H	25,600	25
0001 0111 1111 1111	17FFH	6,143		0110 0111 1111 1111	67FFH	26,623	
0001 1000 0000 0000	1800H	6,144	6	0110 1000 0000 0000	6800H	26,624	26
0001 1011 1111 1111	1BFFH	7,167		0110 1011 1111 1111	6BFFH	27,647	
0001 1100 0000 0000	1C00H	7,168	7	0110 1100 0000 0000	6C00H	27,648	27
0001 1111 1111 1111	1FFFH	8,191		0110 1111 1111 1111	6FFFH	28,671	
0010 0000 0000 0000	2000H	8,192	8	0111 0000 0000 0000	7000H	28,672	28
0010 0011 1111 1111	23FFH	9,215		0111 0011 1111 1111	73FFH	29,695	
0010 0100 0000 0000	2400H	9,216	9	0111 0100 0000 0000	7400H	29,696	29
0010 0111 1111 1111	27FFH	10,239		0111 0111 1111 1111	77FFH	30,719	
0010 1000 0000 0000	2800H	10,240	10	0111 1000 0000 0000	7800H	30,720	30
0010 1011 1111 1111	2BFFH	11,263		0111 1011 1111 1111	7BFFH	31,743	
0010 1100 0000 0000	2C00H	11,264	11	0111 1100 0000 0000	7C00H	31,744	31
0010 1111 1111 1111	2FFFH	12,287		0111 1111 1111 1111	7FFFH	32,767	
0011 0000 0000 0000	3000H	12,288	12	1000 0000 0000 0000	8000H	32,768	32
0011 0011 1111 1111	33FFH	13,311		1000 0011 1111 1111	83FFH	33,791	
0011 0100 0000 0000	3400H	13,312	13	1000 0100 0000 0000	8400H	33,792	33
0011 0111 1111 1111	37FFH	14,335		1000 0111 1111 1111	87FFH	34,815	
0011 1000 0000 0000	3800H	14,336	14	1000 1000 0000 0000	8800H	34,816	34
0011 1011 1111 1111	3BFFH	15,359		1000 1011 1111 1111	8BFFH	35,839	
0011 1100 0000 0000	3C00H	15,360	15	1000 1100 0000 0000	8C00H	35,840	35
0011 1111 1111 1111	3FFFH	16,383		1000 1111 1111 1111	8FFFH	36,863	
0100 0000 0000 0000	4000H	16,384	16	1001 0000 0000 0000	9000H	36,864	36
0100 0011 1111 1111	43FFH	17,407		1001 0011 1111 1111	93FFH	37,887	
0100 0100 0000 0000	4400H	17,408	17	1001 0100 0000 0000	9400H	37,888	37
0100 0111 1111 1111	47FFH	18,431		1001 0111 1111 1111	97FFH	38,911	
0100 1000 0000 0000	4800H	18,432	18	1001 1000 0000 0000	9800H	38,912	38
0100 1011 1111 1111	4BFFH	19,455		1001 1011 1111 1111	9BFFH	39,935	
0100 1100 0000 0000	4C00H	19,456	19	1001 1100 0000 0000	9C00H	39,936	39
0100 1111 1111 1111	4FFFH	20,479		1001 1111 1111 1111	9FFFH	40,959	

APPENDIX 11. MEMORY LOCATIONS: 1K INTERVALS (*Continued*)

Address Bits	Hexadecimal	Decimal	Zone	Address Bits	Hexadecimal	Decimal	Zone
Zone bits = $A_{15}A_{14}A_{13}A_{12}A_{11}A_{10}$							
1010 0000 0000 0000	A000H	40,960	40	1101 0000 0000 0000	D000H	53,248	52
1010 0011 1111 1111	A3FFH	41,983		1101 0011 1111 1111	D3FFH	54,271	
1010 0100 0000 0000	A400H	41,984	41	1101 0100 0000 0000	D400H	54,272	53
1010 0111 1111 1111	A7FFH	43,007		1101 0111 1111 1111	D7FFH	55,295	
1010 1000 0000 0000	A800H	43,008	42	1101 1000 0000 0000	D800H	55,296	54
1010 1011 1111 1111	ABFFH	44,031		1101 1011 1111 1111	DBFFH	56,319	
1010 1100 0000 0000	AC00H	44,032	43	1101 1100 0000 0000	DC00H	56,320	55
1010 1111 1111 1111	AFFFH	45,055		1101 1111 1111 1111	DFFFH	57,343	
1011 0000 0000 0000	B000H	45,056	44	1110 0000 0000 0000	E000H	57,344	56
1011 0011 1111 1111	B3FFH	46,079		1110 0011 1111 1111	E3FFH	58,367	
1011 0100 0000 0000	B400H	46,080	45	1110 0100 0000 0000	E400H	58,368	57
1011 0111 1111 1111	B7FFH	47,103		1110 0111 1111 1111	E7FFH	59,391	
1011 1000 0000 0000	B800H	47,104	46	1110 1000 0000 0000	E800H	59,392	58
1011 1011 1111 1111	BBFFH	48,127		1110 1011 1111 1111	EBFFH	60,415	
1011 1100 0000 0000	BC00H	48,128	47	1110 1100 0000 0000	EC00H	60,416	59
1011 1111 1111 1111	BFFFH	49,151		1110 1111 1111 1111	EFFFH	61,439	
1100 0000 0000 0000	C000H	49,152	48	1111 0000 0000 0000	F000H	61,440	60
1100 0011 1111 1111	C3FFH	50,175		1111 0011 1111 1111	F3FFH	62,463	
1100 0100 0000 0000	C400H	50,176	49	1111 0100 0000 0000	F400H	62,464	61
1100 0111 1111 1111	C7FFH	51,199		1111 0111 1111 1111	F7FFH	63,487	
1100 1000 0000 0000	C800H	51,200	50	1111 1000 0000 0000	F800H	63,488	62
1100 1011 1111 1111	CBFFH	52,223		1111 1011 1111 1111	FBFFH	64,511	
1100 1100 0000 0000	CC00H	52,224	51	1111 1100 0000 0000	FC00H	64,512	63
1100 1111 1111 1111	CFFFH	53,247		1111 1111 1111 1111	FFFFH	65,535	

APPENDIX 12. PROGRAMMING MODELS

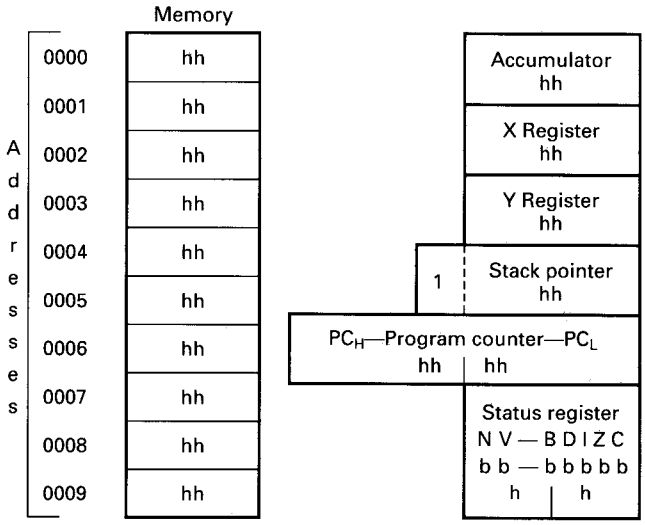


Fig. A12-1 6502 programming model.

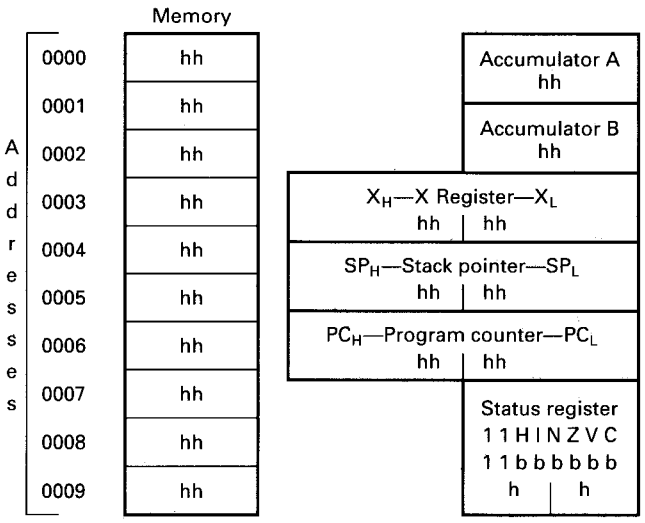


Fig. A12-2 6800/6808 programming model.

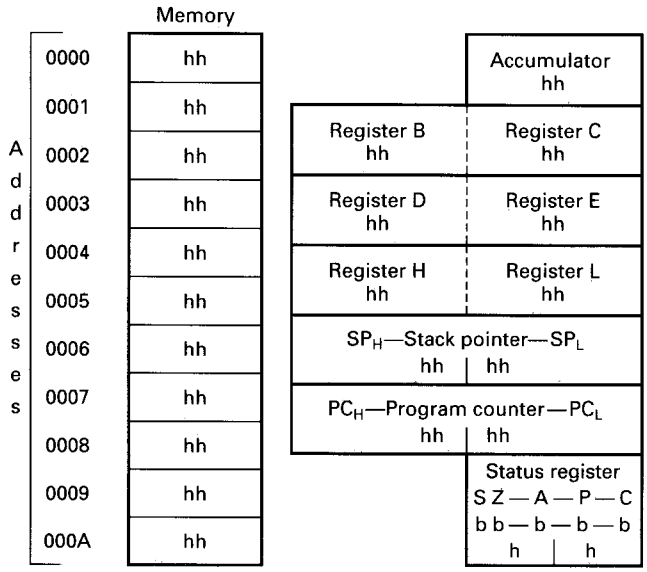


Fig. A12-3 8085/Z80 (8085/8080 subset) programming model.

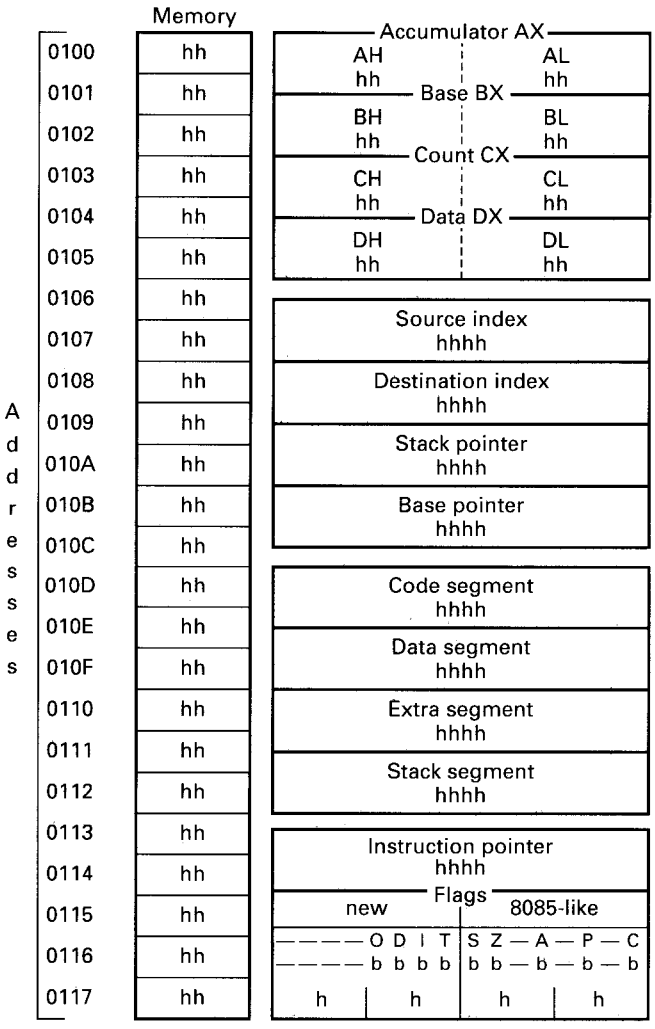


Fig. A12-4 8088/8086 programming model.

Answers to Odd-Numbered Problems

CHAP. 1. 1-1. a. 1 b. 2 c. 2½ 1-3. a. 10 b. 2 c. 5 d. 16 1-5. 1,024, 4,096, 8K 1-7. 1010 1100, 172 1-9. 201 1-11. 1100 0111, 199 1-13. 111000 1-15. 1001 0110 1-17. F52B, F52C, F52D, F52E, F52F, F530 1-19. a. 1111 1111 b. 1010 1011 1100 c. 1100 1101 0100 0010 d. 1111 0011 0010 1001 1-21. 0011 1110, 0000 1110, 1101 0011, 0010 0000, 0111 0110 1-23. a. 4,095 b. 16,383 c. 32,740 d. 46,040 1-25. 16,384, 16K 1-27. 0000, FFFF 1-29. a. EE b. 1D7B c. 3BFF d. B8B5 1-31. a. 87 b. 9,043 c. 597,266 1-33. 100 1100, 100 1001, 101 0011, 101 0100

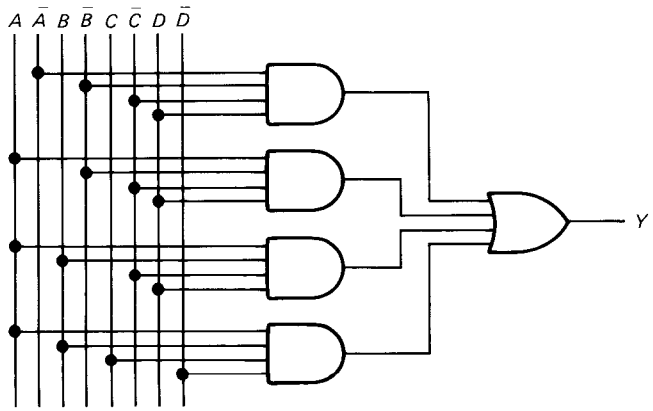
CHAP. 2. 2-1. One or more, one 2-3. Noninverter 2-5. 64, 000000 2-7. 3, 9, C, F 2-9. 128, 1111111 2-11. 0, 59 2-13. $Y = \overline{A + B}$, low 2-15. 8 2-17. 0, $Y = \overline{A} + \overline{B} + \overline{C}$, 000 to 110, 111 2-19. $Y = \overline{ABC}$, 0 2-21. $Y = \overline{AB + CD}$, 16, 0000, 0001, 0010, 0100, 0101, 0110, 1000, 1001, 1010 2-23. a. 0000 b. 0001 c. *JIM* d. *OPR* 2-25. a. Positive b. Negative c. Positive d. Negative

CHAP. 3. 3-1. High; low; inverter 3-3. None, Z_5, Z_6 3-5. Q is 1, $\overline{Q}$ is 0 3-7. Change the output NOR gate of Fig. 3-28*a* to a bubbled AND gate; all bubbles cancel leaving the simplified circuit of Fig. 3-28*b*. 3-9. 0, 1 3-11. 512 3-13. 16; 0, 1, 1, 0 3-15. 1, 0, inverter 3-17. a. None b. Z_7 c. Z_2 d. X_2 and Y_2 3-19. 0, 1 3-21. 512 3-23. Low, high 3-25. a. 0 b. 1 c. 1 d. 1 3-27. a. 11010b. 01001 c. 11111 d. 10010 3-29. Remove the inverter 3-31. a. *CARRY* = 0, *SUM* = 0 b. 0, 1 c. 0, 1 d. 1, 0 3-33. a. 0011 1100 b. 0101 0000 1100 c. 0001 1110 0101 1100 d. 1111 0000 1101 0010

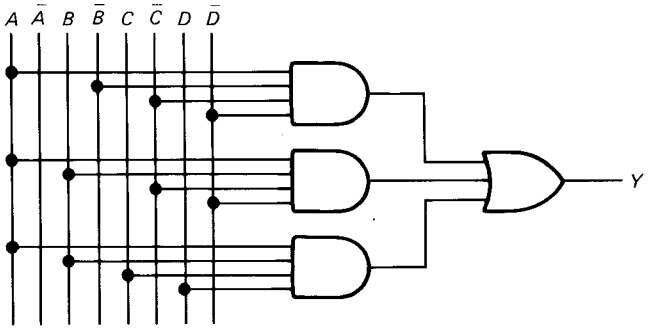
CHAP. 4. 4-1. 1.075 mA, 1.387 mA 4-3. 5 4-5. All; b, c, f, g

CHAP. 5. 5-1. $\overline{A}BC\overline{D}$, $AB\overline{C}D$, $ABC\overline{D}$

5-3.



5-5.

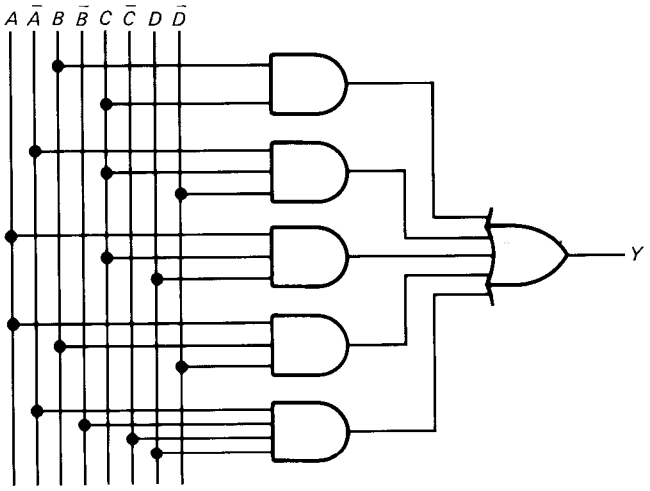


5-7.

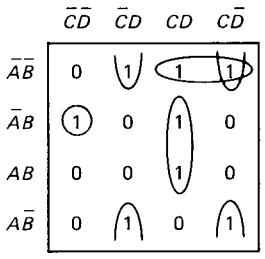
	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	0	0	0	0
AB	1	1	1	1
$A\bar{B}$	1	1	1	1

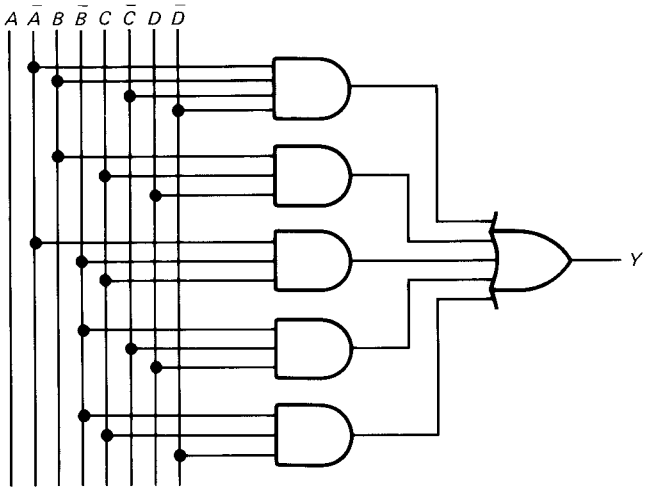
5-9.

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	0	1
$\bar{A}B$	0	0	1	1
AB	1	0	1	1
$A\bar{B}$	0	0	1	0

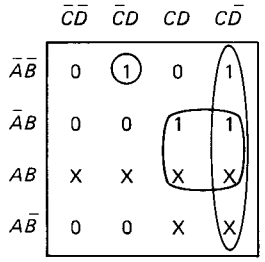


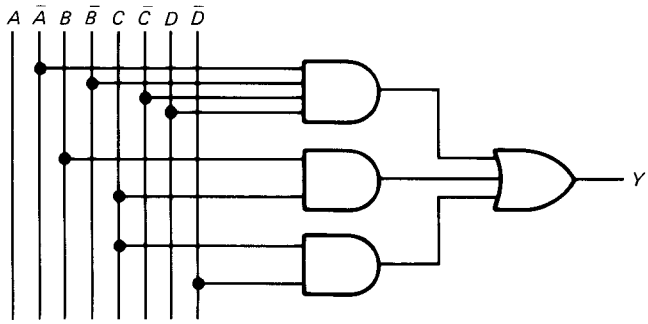
5-11.



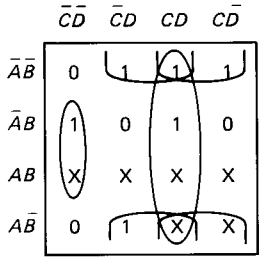


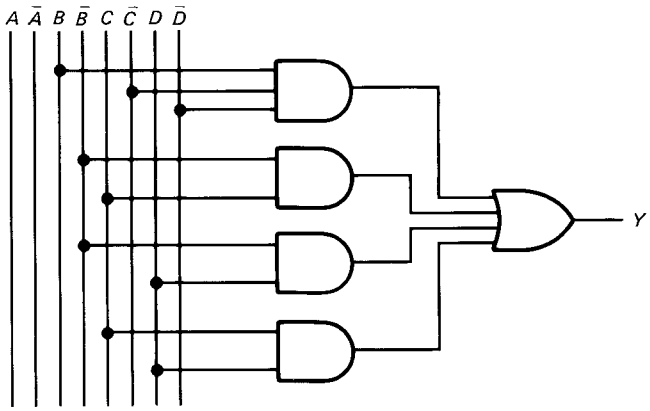
5-13.





5-15.





CHAP. 6. 6-1. a. 0001 1000, 18H b. 0010 0100, 24H c. 0010 1010, 2AH d. 0110 0011, 63H 6-3. a. 7BH b. 78H c. A8H d. D1H 6-5. a. +30 b. −7 c. −28 d. +49 6-7. a. F9H b. 01H c. 03H d. 1FH 6-9. a. 1110 1101, EDH b. 1101 0000, D0H c. 0010 0101, 25H d. 1101 1111, DFH 6-11. 9BH, DDH

CHAP. 7. 7-1. a. C b. G 7-3. a. 0000 b. 1001 7-5. 3 MHz; the output frequency is half the input frequency 7-7. $Q = 0, Y = 1$; $Q = 1, Y = CLK$

CHAP. 8. 8-1. a. 0001 0111 b. 1000 1101 8-3. 385 Ω 8-5. 4 μs 8-7. 6.4 μs 8-9. 65,535 8-11. 1 μs, 6 μs 8-13. 1.6 μs, 0.2 μs 8-15. Two answers: 7490 (divide by 10) and 7492 (divide by 6), or 7490 (divide by 5) and 7492 (divide by 12) 8-17. 136 8-19. a. 0, 1 b. 1, 1 c. 0

CHAP. 9. 9-1. 16,384 9-3. 12

9-5.

Address	Data
DDDD	UDDD UDDU
DDDU	DUUU UUDD
DDUD	DDUU DUUD
DDUU	DDUD DDUU
DUDD	DDDU DUUU
DUDU	DUDU UUUU
DUUD	UUUD UUDU
DUUU	UUUU UDDD

9-7. 63 9-9. BFFFH; 49,151 9-11. a. 47, 212, 207, 110, 83, 122 b. 36,357

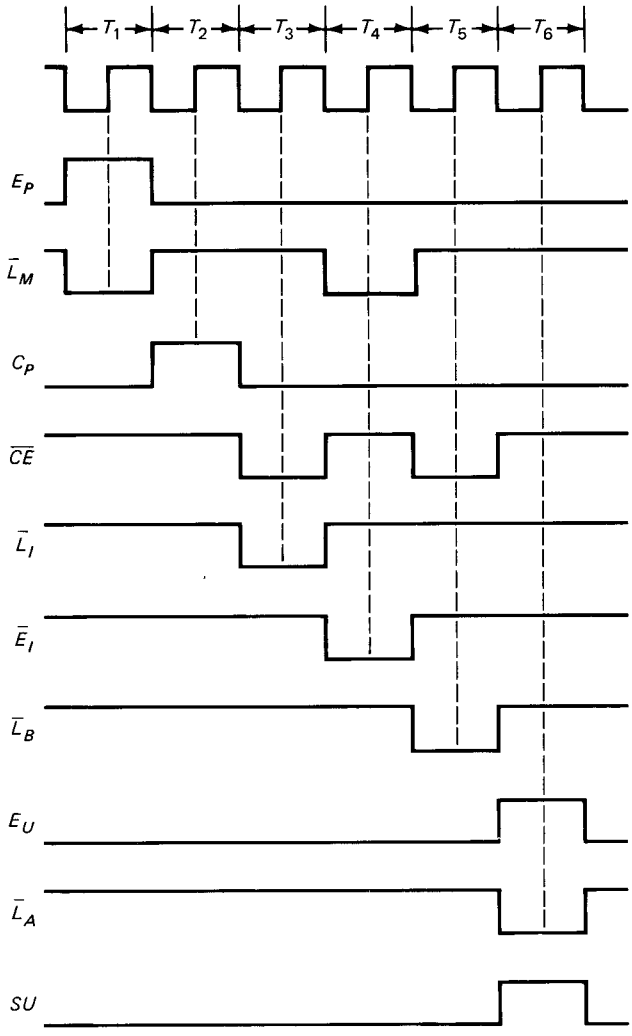
CHAP. 10. 10-1.

Address	Mnemonic
0H	LDA DH
1H	ADD EH
2H	SUB FH
3H	OUT
4H	HLT
DH	05H
EH	04H
FH	06H

10-3.

Address	Mnemonic
0H	LDA BH
1H	ADD CH
2H	SUB DH
3H	ADD EH
4H	SUB FH
5H	HLT
BH	08H
CH	04H
DH	03H
EH	05H
FH	02H

10-5.



10-7. LDA: 1A3H or 0001 1010 0011, 2C3H or 0010 1100 0011, 3E3H or 0011 1110 0011; SUB: 1A3H or 0001 1010 0011, 2E1H or 0010 1110 0001, 3CFH or 0011 1100 1111 10-9. a. Negative edge; *CLK* is on its rising edge b. High c. Low d. High 10-11. a. Low b. Low c. High

CHAP. 11. 11-1.

```
Mnemonic
MVI A,64H
MVI B,96H
MVI C,C8H
HLT
```

11-3.

```
Mnemonic
MVI A,32H
STA 4000H
MVI A,33H
STA 4001H
MVI A,34H
STA 4002H
HLT
```

11-5.

```
Mnemonic
MVI A,44H
MVI B,22H
ADD B
STA 5000H
HLT
```

11-7. a. 120 b. 119 c. Change the first instruction to MVI C,D2H

11-9.

```
Mnemonic
MVI A,00H
MVI B,19H
MVI C,07H
CALL F006H
STA 2000H
HLT
```

11-11.

```
Label   Mnemonic
        IN 01H
        ANI 01H
        JNZ ODD
        MVI A,45H
        JMP DONE
ODD:    MVI A,4FH
DONE:   MVI C,08H
AGAIN:  OUT 04H
        RAR
        DCR C
        JNZ AGAIN
        HLT
```

11-13.

Address	Contents
2000H	DBH
2001H	02H
2002H	E6H
2003H	01H
2004H	CAH
2005H	00H
2006H	20H
2007H	DBH
2008H	01H
2009H	32H
200AH	00H
200BH	40H
200CH	76H

11-15.

Address	Contents
2000H	0EH
2001H	23H
2002H	0DH
2003H	C2H
2004H	02H
2005H	20H
2006H	C9H

11-17.

Label	Mnemonic
	MVI A,05H
LOOP:	CALL F020H
	DCR A
	JNZ LOOP
	RET

Address	Contents
E100H	3EH
E101H	05H
E102H	CDH
E103H	20H
E104H	F0H
E105H	3DH
E106H	C2H
E107H	02H
E108H	E1H
E109H	C9H

11-19.

Address	Contents
F080H	3EH
F081H	06H
F082H	32H
F083H	93H
F084H	F0H
F085H	CDH
F086H	60H
F087H	F0H
F088H	3AH
F089H	93H
F08AH	F0H
F08BH	3DH
F08CH	32H
F08DH	93H
F08EH	F0H
F08FH	C2H
F090H	85H
F091H	F0H
F092H	C9H

11-21.

Address	Contents
2000H	D3H
2001H	04H
2002H	0EH
2003H	42H
2004H	0DH
2005H	C2H
2006H	04H
2007H	20H
2008H	2FH
2009H	00H
200AH	C3H
200BH	00H
200CH	20H

CHAP. 12. 12-1.

Mnemonic
MVI A,00H
MVI B,01H
MVI C,59H
MVI D,02H
MVI E,F1H
ADD C
ADD E
MOV L,A
MVI A,00H
ADC B
ADD D
MOV H,A
HLT

An alternative solution is

Mnemonic
MVI A,F1H
ADI 59H
MOV L,A
MVI A,02H
ACI 01H
MOV H,A
HLT

12-3.

Label	Mnemonic
	LXI H,4FFFH
LOOP:	INX H
	MOV B,M
	MOV A,H

```
        ADI 40H
        MOV H,A
        MOV M,B
        SUI 40H
        MOV H,A
        CPI 53H
        JNZ LOOP
        MOV A,L
        CPI FFH
        JNZ LOOP
        HLT
```

12-5.

Label	Mnemonic
	LXI SP,E000H
	MVI A,00H
	MVI B,FFH
LOOP:	INR A
	OUT 22H
	CALL F010H
	DCR B
	JNZ LOOP
	HLT

12-7.

Label	Mnemonic
	LXI SP,E000H
	LXI H,5FFFH
LOOP:	INX H
	MOV A,M
	OUT 22H
	CALL F020H
	MOV A,H
	CPI 61H
	JNZ LOOP
	MOV A,L
	CPI FFH
	JNZ LOOP
	HLT

12-9.

Label	Mnemonic
	LXI SP,E000H
	LXI H,4FFFH
LOOP:	INX H
	MOV A,M
	MOV B,08H
AGAIN:	OUT 22H
	CALL F010H
	RAR
	DCR B
	JNZ AGAIN
	MOV A,L
	CPI FFH
	JNZ LOOP
	HLT

CHAP. 14. 14-1. How you would accomplish your task without a computer. 14-3. Branch. 14-5. The subroutine (part of the program) needs to be written only once but can then be used many times. 14-7. Formula translation. 14-9. Creating a language which would encourage programmers to write by using what are considered "correct" programming practices.

CHAP. 15. 15-1. By its address. 15-3. 1,048,576. 15-5. The accumulator. 15-7. Registers are faster. 15-9. The status register (or condition code register or flag register). 15-11. The carry flag. 15-13. No. 15-15. DE. 15-17. C581. 15-19. 8 bits. 15-21. 256 bytes. 15-23. 16 bits. 15-25. Nothing. They are always set. 15-27. None. 15-29. It is named AX and is 16 bits wide with an 8-bit upper half (called AH) and an 8-bit lower half (called AL). 15-31. The instruction pointer. 15-33. 65,536 bytes.

CHAP. 16. 16-1. Nothing. 16-3. The original number in the accumulator is still there. 16-5. 00. 16-7. It copies the contents of the Y register to the accumulator. 16-9. STY. 16-11. 01. 16-13. 16. 16-15. **C**lea**R** accumulator **A**.

16-17.

Addr	Obj	Assembler	Comment
0000	C6	LDAB #$89	Load the number immediately following the LDAB op code (C6) into accumulator B (89)
0001	89		
0002	17	TBA	Transfer (copy) the contents of B to A
0003	3E	WAI	Stop

16-19. 76. 16-21. It copies the contents of register C to register B. 16-23. STA aaaa *[LD (aaaa),A]*. 16-25. DEBUG. 16-27. Register or memory. 16-29. DL. 16-31. The contents of memory location 4456_{16}. 16-33. It stands for **a**ssemble and it translates 8088/8086 mnemonics into machine code. 16-35. It executes one instruction and then displays the current values of all registers and stops.

16-37.

```
-a
9522:0100 mov BL,89
9522:0102 mov CL,BL
9522:0104

-u 100 103
9522:0100 B389        MOV  BL,89
9522:0102 88D9        MOV  CL,BL

-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=ADDE  BP=0000  SI=0000  DI=0000
DS=9522  ES=9522  SS=9522  CS=9522  IP=0100   NV UP EI PL NZ NA PO NC
9522:0100 B389          MOV     BL,89

-t
AX=0000  BX=0089  CX=0000  DX=0000  SP=ADDE  BP=0000  SI=0000  DI=0000
DS=9522  ES=9522  SS=9522  CS=9522  IP=0102   NV UP EI PL NZ NA PO NC
9522:0102 88D9          MOV     CL,BL

-t
AX=0000  BX=0089  CX=0089  DX=0000  SP=ADDE  BP=0000  SI=0000  DI=0000
DS=9522  ES=9522  SS=9522  CS=9522  IP=0104   NV UP EI PL NZ NA PO NC
```

Note: Answers to Chapters 18 to 23 are in the teacher's manual.

Index

Note: For entries marked with (#), refer also to specific families listed under "Microprocessor families."

Absolute accuracy, 488
Absolute addressing, 265, 333–334
Access time, 132–133
Accumulator, 142, 158, 174, 176, 184, 226, 230, 232, 233, 235 (*See also* ALU)
Accumulator addressing, 264–268
Accuracy, 488–489
Active low state, 98
ADD instruction, 143, 148–150, 178, 197–198
Adder-subtracter, 85–87, 142, 158
Addition, 79–87, 199, 271–272, 281–282, 284–287, 290–292, 294, 298–300
Addition-with-carry, 274, 276–277
Address, 12, 131, 133, 135–137, 330 (*See also* Addressing mode)
Address bus, 225
Address field, 145
Address line, 131
Address mapping, 183
Address state, 147 (*See also T* state)
#Addressing mode, 224–226
  absolute, 265, 333–334
  base plus index, 340
  base relative plus index, 340–341
  direct, 187, 264–268
  extended, 266
  immediate, 187, 244, 247, 264–268
  implied, 188, 264–267
  indexed, 332–336
  indexed indirect, 335
  indirect, 205, 331, 333, 336, 338–340
  indirect indexed, 334–335
  paging, 263–264
  program direct, 268
  program indirect, 340
  program relative, 337–338
  range, 225
  register (accumulator), 188, 264–268
  register indirect, 336, 338–340
  register relative, 337
  relative, 330, 332–333, 335, 337–338
  zero page, 333–334
Alphanumerics, 14
ALU, 7, 79, 175
American Standard Code for Information Exchange, 14–15, 271
ANA instruction, 184
Analog interface, 485
Analog-to-digital (A/D) converter, 485, 491–493
AND gate, 22–23, 33–34, 49, 54
AND instruction, 305–306, 308–310, 312–314
AND operations, 65–66
AND sign, 24–25
AND-OR gate, 55
AND-OR-INVERT gate, 55–57
ANI instruction, 184
#Architecture, 224–226
  of SAP-1, 140–142
  of SAP-2, 173–176
  of SAP-3, 195–196
#Arithmetic instructions, 271–276
Arithmetic-logic unit, 7, 79, 175
ASCII code, 14–15, 271
Assembler, 181, 222, 354–355, 357, 358 (*See also* Machine language)
Assembly language, 145, 221–222, 337
Associative law, 64
Asynchronous operation, 142 (*See also* Clocking)

B register, 142, 158, 175
Base, 6–7
Base plus index addressing, 340
Base register, 340
Base relative plus index addressing, 340–341
BASIC, 221
BCD number, 13–14, 270–271
BCD-to-decimal conversion, 13–14
Bidirectional register, 173
Binary adder, 82–83
Binary adder-subtracter, 85–87, 142, 158
Binary addition, 79–87 (*See also* Addition)
Binary code, 2–3
Binary digit, 4
Binary number, 2–3, 6–15, 270, 271, 274
Binary odometer, 1–2, 84
Binary programming (*see* Machine language)
Binary subtraction, 80–81, 85–87 (*See also* Subtraction)
Binary weight, 6
Binary word, 20
Binary-coded-decimal number, 13–14, 270–271
Binary-to-decimal conversion, 3, 6–7
Binary-to-decimal decoder, 27
Binary-to-hexadecimal conversion, 10–11, 12
Bipolar families, 48
Bit, 4
Bit comparison, 42
BIT instruction, 309–310, 311
Bit position, 271
Bit-serial form (*see* Serial data stream; Serial loading)
Boldface notation, 42
Boolean algebra, 19, 23–27, 64–70
Boolean function generator, 58–60
Borrow, 196, 275–276, 281
#Branch instruction, 179–180, 219, 342–343
Branch-back instruction (*see* Return instruction)
Breakpoint, 294
Broadside loading, 110
Bubble memory, 135
Bubbled AND gate, 33–34
Bubbled OR gate, 36
Buffer, 54 (*See also* Buffer register)
Buffer register, 54, 106–107, 110, 122
Bus, 69, 122
Bus transient, 152
Bus-organized computer, 121, 122–125, 152
Byte, 6, 189–193
  defined, 345, 348, 351

C language, 221
C register, 175
CALL instruction, 180, 182, 210–211
Carry flag, 196–197, 272, 274–277, 281, 312
Cell, 134
Central processing unit (*see* CPU)
Chip, 4, 49
Chip enable, 134
Chunking, 11
Clear, 97
Clear-start debouncer, 158–159
Clock, 93, 158
Clock generator, 102–103
Clocking:
  edge-triggered, 96–100
  level, 93–97, 102
  master-slave, 100–103
  positive and negative, 94
CMA instruction, 184
CMOS, 48
COBOL, 221
Code, binary, 2–3
Code segment register, 268
Comment, 181–182
Commutative law, 64
#Compare and test instruction, 343
Compatibility, 51–52
Complement, 19
Complement instruction, 311, 314
Complementary MOSFETs, 48
Computer, 7
  architecture, 224–226
  bus-organized, 121, 122–125, 152
  (*See also* Microprocessor)
CON (*see* Control unit)
Condition code register, 227–228, 232–233
#Conditional jump (branching), 179, 180, 187, 342–343
Contact bounce, 92–93
Content, 131, 224–225
Control input, 90
Control matrix, 36–37, 161
Control ROM, 163
Control routine, 148–152
Control unit, 7, 146–152
Controlled buffer register, 106–107
Controlled inverter, 41–42
Controlled shift register, 108–110
Controller-sequencer, 141–142, 161, 174
Conversion, 331
  analog-to-digital, 485, 491–493
  BCD-to-decimal, 13–14
  binary-to-decimal, 6–7
  binary-to-hexadecimal, 10–11, 494–496
  decimal-to-binary, 8
  decimal-to-hexadecimal, 13
  digital-to-analog, 485, 486, 489
  hexadecimal-to-binary, 10–11, 270
  hexadecimal-to-decimal, 11–13
Core RAM, 133

Counter:
down, 118
mod-10, 116–118
presettable, 118–120, 162
program, 113, 140, 147, 153, 173, 227, 230–232, 234, 330
programmable modulus, 120
ring, 114–116, 146–147, 159–161
ripple, 110–113
software, 181
synchronous, 113–114
TTL, 120
up-down, 118
Counter method of A/D conversion, 491–492
#CPU, 7, 213 (*See also* ALU; Control unit)
CPU register, 195–196
Current sink, 52
Current steering, 491

D flip-flop, 96–98
D latch, 95–96
DAD instruction, 204–205
Data, 3
Data bus, 225
Data processor, 3
Data segment, 338
Data selector, 58–59
Data settling (*see* Bus transient; Settling time)
#Data transfer instructions, 241–260
Date pointer, 205
De Morgan's theorem, 33–37, 66
Debouncer, 92–93, 158–159
DEBUG, 253, 255–260, 293–302, 337–340
Decade counter, 118, 120
Decimal addition, 284–285, 290–292, 298–300
Decimal adjust, 280, 284–285, 290, 298
Decimal flag, 279–281
Decimal number, 84–85
Decimal odometer, 1
Decimal weight, 6
Decimal-to-binary conversion, 8, 21–22
Decimal-to-hexadecimal conversion, 13
Decision-making element, 25
Decoder:
binary-to-decimal, 27
binary-to-hexadecimal, 54
decimal-to-BCD, 54
instruction, 125, 158–159
seven-segment, 54
#Decrement instruction, 178, 180–181, 200, 205, 343
Define byte, 345, 348, 351
Delay, 189–190
Digit, 1
Digital-to-analog (D/A) converter, 485, 486–489
Diode ROM, 130–131
Diode-transistor logic, 48
Direct addressing, 187, 264–268
Direct reset, 97
Direct set, 97
Disassembler, 222
Distributive law, 65
Division, 276, 302
Don't care condition, 75–77, 95
Do-nothing state (*see* NOP instruction)
Double-byte addition, 199
Double-byte subtraction, 202
Double-dabble, 8
Double inversion, 34, 66
Double-precision number, 274
Down counter, 118
Driver, 54
DTL, 48
Duality theorem, 66–67
Dynamic RAM, 133–134

ECL, 48
Edge triggering, 96–100
Effective address, 330
8080/8085/Z80 family, 214, 417–422, 502–506
addressing, 266–267, 336, 409
architecture, 233–235, 329
arithmetic instructions, 286–287, 292–293, 391–395, 411–412, 416
conditional jump (branch) instructions, 351–352, 402, 413–414, 417
CPU control instructions, 381, 410, 415
data transfer instructions, 249–253, 381–390, 410–411, 415–416
flag instructions, 287–292, 390–391, 408–409, 411, 416
increment and decrement instructions, 398–400, 413, 416–417
input-output instructions, 408, 415, 417
interrupt instructions, 407–408, 415, 417
logical instructions, 395–398, 412, 416
programming, 511
rotate and shift instructions, 323–324, 398, 412–413, 416
stack instructions, 406–407, 415, 417
subroutine instructions, 370–373, 402–406, 414–415, 417
test and compare instructions, 352, 401, 413, 417
unconditional jump instructions, 350–351, 400, 413, 417
8086/8088 family, 214, 469–470
addressing, 267–269, 336–341
architecture, 235–237, 329
arithmetic instructions, 293–294, 300–302, 447–450, 466
conditional jump (branch) instructions, 357–358, 456–459, 467
CPU control instructions, 445, 465
data transfer instructions, 253–260, 445–446, 466
flag instructions, 294–299, 446–447, 466
increment and decrement instructions, 455, 467
input-output instructions, 462–463, 468
interrupt instructions, 461–462, 468
logical instructions, 314–317, 450–451, 466
loop instructions, 464–465, 468
programming, 511
rotate and shift instructions, 324–327, 451–455, 467
stack instructions, 460–461, 468
string instructions, 463–464, 468
subroutine instructions, 373–377, 459–460, 468
test and compare instructions, 358, 456, 467
unconditional jump instructions, 355–357, 455, 467
Emitter-coupled logic, 48
ENABLE input, 23
Encoder, 21–22, 54
End-of-conversion signal, 492
Erasable PROM (EPROM), 132, 224
Even parity, 39, 234
EXCLUSIVE-NOR gate, 42
EXCLUSIVE-OR gate, 37–42, 307–309
Execution cycle, 148–152
Expandable gate, 56–57
Expander gate, 56–57
Extended addressing, 266
Extended register, 204–205

Factoring, 69, 70
Fanout, 52–53
Fetch cycle, 148, 150, 151, 227
Fetch microroutine, 152, 161
Firmware, 243, 247, 251
First-in-last-out (FILO) structure, 228, 363
#Flag instructions, 175, 175, 179, 180–181, 187, 227–228, 272–276, 310
Flip-flop, 90–103
Floating TTL input, 50–51
Flowchart, 217, 218–220
FORTH, 221
FORTRAN, 221
Full adder, 81–82
Function tables, 499–500
Fundamental product, 67

Gate:
AND, 22–23, 33–34, 49, 54
AND-OR, 55
AND-OR-INVERT, 55–57
expandable, 56–57
NAND, 34–36, 49, 53–55, 118–120
NOR, 32–34, 49, 53–54
NOT, 19–20
OR, 20–22, 36, 54
standard TTL, 49
XNOR, 42
XOR, 37–42, 49
General-purpose register, 227, 230, 232–236

Half-adder, 81
Half-carry flag, 272
Halt instruction, 143, 151, 185, 241
Hand-assembly, 178, 183, 244, 248, 251
Handshaking, 176, 186
Hardware, 3–4, 213
Hardwired control, 161
Hex inverter, 20
Hexadecimal address, 133, 136–137
Hexadecimal number, 9–13, 14, 270
Hexadecimal-to-binary conversion, 10–11, 270
Hexadecimal-to-decimal conversion, 11–13
Hex-dabble, 13
High-level language, 221
High-speed TTL, 50
Hold time, 98

Immediate addressing, 187, 244, 247, 264–268
Immediate instruction, 176, 184, 201–202, 204, 206
Implied addressing, 188, 264–267
IN instruction, 185
Inactive state, 90
INCLUSIVE OR (*see* OR gate)
#Increment instruction, 147, 178, 180–181, 199–200, 205, 343
Index register, 227, 231, 232, 234, 236, 332, 340
Indexed addressing, 332, 333–336
Indexed indirect addressing, 335
Indirect addressing, 205, 331, 333, 336, 340
Indirect indexed addressing, 334–335
Indirect instruction, 205–207
Inherent addressing, 264–267
Input gate lead, 69
Input-output unit, 7
Input register, 173
Input unit, 7
Instruction cycle, 151 (*See also* Machine cycle)
Instruction decoder, 125, 158–159
Instruction field, 145
Instruction pointer, 205, 236, 330
Instruction register, 125, 141, 153, 174

Instruction set, 142–144, 240
Integrated circuit, 4, 48
Interface circuit (*see* Analog interface)
Inversion:
  bubble, 19–20
  double, 34, 66
  sign, 19, 23–24
  symbol, 19–20
Inverter, 19–20, 41–42
I/O unit, 7
Italic notation, 25

JK flip-flop, 99–103
JK master-slave flip-flop, 100–103
Jump flag, 187
#Jump instruction, 173, 179–180, 182, 183, 202–204, 342–343

K- (kilo-), 7
K input, 99–100
Karnaugh maps, 70–77

Label, 181–182
Ladder, 490–491
Large-scale integration, 48
Latch, 90–96
LDA instruction, 142, 148, 149, 176
LDA microroutine, 161–162
LED display, 3
Level clocking, 93–97, 102
Light-emitting diode, 3
Load the accumulator instruction, 142, 148, 149, 176, 242–248, 252–253
Loading:
  parallel, 110
  serial, 108–110
  TTL device, 52–53
Logic circuit, 19, 68
#Logical instructions, 305–308
Loop, 181, 218–219, 342–344
Loop counter, 181
Low-level language, 221
Low-power Schottky TTL, 50, 52–53
Low-power TTL, 50
LSB (least significant byte), 274, 488
LSI, 48

Machine cycle:
  definition, 151
  fixed, 161–162, 163
  variable, 163–164
Machine language, 145, 146, 220, 221, 337
Machine phase (*see T* state)
Macroinstruction, 152–153
Magnetic core, 5
Magnetic tape, 5
Manual assembly, 221
Manual-auto debouncer, 158–159
Mapping (*see* Address mapping)
MAR, 140, 153, 174
Mask, 131, 186, 306–308
Master-slave flip-flop, 100–103
Medium-scale integration, 48
Memory, 5–7, 130–137, 224, 268
Memory address register, 140, 153, 174
Memory data register, 174
Memory element, 90–103
Memory enable (*see* Chip enable; Write enable)
Memory-intensive architecture, 329
Memory location, 10–12, 331, 507–510
Memory-reference instruction, 143–144, 176–177
Memory register (*see* Memory location)
Memory state, 147
Microcode (*see* Microprogram)
Microcomputer, 7
Microcontroller, 161–164
Microinstruction, 152
Microprocessor, 7, 213–216, 226–237, 270–271
Microprocessor families (*see* 8080/8085/Z80 family; 8086/8088 family; 6502 family; 6800/6808 family)
Microprogram, 152–153, 161–164
Microroutine (*see* Microprogram)
Mnemonic, 143, 221
Modulus, 116–120
Monitor, 174, 241
  assembly, 222
Monotonic D/A converter, 489
MOS families, 48
Move instruction, 177–178, 195–196, 199, 206
MRI, 143–144, 176
MSB (most significant bit), 200, 273, 274, 492
MSI, 48
Multiplexer, 58–60, 153
Multiplication, 182, 183, 276, 300–302
MVI, 189, 195–196, 199

NAND gate, 34–36, 49, 53–55, 118–120
NAND latch, 92–95
Natural modulus, 120
n-channel MOSFETs, 48
NEG instruction, 308, 311–312, 316–317
Negative (sign) flag, 275, 277–278, 282–283
Negative clocking, 94
Negative logic, 25
Negative toggle, 118
Nesting, 343–344
  loop, 343–344
  subroutine, 189–190, 364, 367, 369–371, 373–374
Nibble, 13–14
NMOS, 48
No operation instruction, 241, 242, 245, 249
Noise margin, 52
Noninverter, 20
Nonsaturated circuit, 4–5
Nonvolatile memory, 133
NOP instruction, 148, 185, 241, 242
NOR gate, 32–34, 49, 53–54
NOR latch, 91, 92
NOT gate, 19–20
NOT instruction, 308, 315–316
Notation:
  boldface, 42
  italic, 25
  positional, 11–12
  roman, 25
Number:
  binary, 2, 3, 6–15, 270, 271, 274
  binary-coded-decimal, 13–14, 270–271
  decimal, 1, 84–85
  hexadecimal, 9–13, 14, 270
  (*See also* Conversion)

Object code, 221
Object program, 145
Octet, 72, 73
Odd parity, 39, 234
Odd-parity generator, 40
Odd-parity tester, 39
Odometer, 330
  binary, 1–2, 84
  decimal, 1
  hexadecimal, 9
Offset, 332
On-chip decoding, 131, 132
1's complement, 41–42, 312
Open-collector gate, 58
Operand, 145, 176
Operation code, 144, 176–177, 241
Operational amplifier (op amp), 485–486
OR gate, 20–22, 36, 54
OR instruction, 65, 66, 184, 306–307, 309, 310, 313, 314–315
OR sign, 24
OUT instruction, 143, 150–151, 185
Output buffer, 493
Output register, 7, 106–107, 110, 142, 158, 176
Overflow, 87, 196, 272–274, 279, 284, 288–289, 296–297
Overlapping, 74

Paging, 263–264
Pair, 72
Parallel loading, 110
Parameter passing, 183
Parity, 39, 234
Parity flag, 203, 288–289, 296
Parity generator, 39–40
Pascal, 221
PC, 113, 140, 147, 153
p-channel MOSFETs, 48
Phase (*see T* state)
Pinouts, 499–500
PMOS, 48
Pointer, 140, 205, 227
POP instruction, 209–210
Port instruction, 185–186
Positional notation, 11–12
Positive clocking, 94
Positive logic, 25
Positive toggle, 118
Power dissipation, 49
Power of 2, 7
Power supply, 158
Preset, 97
Presettable counter, 118–120, 162
Prime memory (*see* Dynamic RAM; Static RAM)
Program, 3, 216
Program counter, 113, 140, 147, 153, 173, 227, 230–232, 234, 330
Program direct addressing, 268
Program indirect addressing, 340
Program relative addressing, 337–338
Program status word, 208
Programmable modulus, 120
Programmable ROM (PROM), 131–132, 224
Programmed multiplication, 182, 183
#Programming, 135–136, 216–222
  data transfer instructions, 241–260
  models, 511
PROM, 131–132, 224
Propagation delay time, 49, 98
Punched cards, 5
PUSH instruction, 208–209 (*See also* Stack)
Pushing and popping registers, 366, 367–368, 370, 371, 374

Quad, 72–73

Race condition, 91, 94, 95, 100
Radix, 6–7
RAL instruction, 185, 200, 201
Random-access memory (RAM), 133–137, 153, 224
RAR instruction, 185, 200, 201
Read-only memory (ROM), 130–133, 161–164, 224

Redundant Karnaugh group, 74–75
Refresh, 133–134
Register, 4, 217
  bidirectional, 173
  buffer, 54, 106–107, 110, 122
  controlled, 106–110
  CPU, 195–196
  8-bit, 229–230
  input, 173
  output, 7, 106–107, 110, 142, 158, 176
  pair, 204
  shift, 108–110
  shift-left, 108, 109
  shift-right, 108, 109
  16-bit, 230
  three-state, 121–122
  transfers, 122–123
  width of, 229–230
  (*See also specific types of register*)
Register addressing, 188, 264–268
Register indirect addressing, 336, 338–340
Register-intensive architecture, 329
Register parameter passing, 183
Register relative addressing, 337
Relative accuracy, 488–489
Relative addressing, 330, 332–333, 335, 337–338
Reset-and-carry, 1
Resolution, 488
Return instruction, 180, 210–211, 364–366
Ring counter, 114–116, 146–147, 159–161
Ripple counter, 110–113
Rolling, Karnaugh map, 74
ROM (*see* Read-only memory)
Roman notation, 25
#Rotate instruction, 185, 200, 319–321
RS latch, 90–94

SAP-1, 140–164
  counters, 106, 107, 113, 116, 117
  parts list, 501
  RAM, 115–116
SAP-2, 144, 151, 173–193
SAP-3, 144, 195–212
Saturated circuit, 4
Saturation delay time, 4, 50
Schmitt trigger, 54–55
Schottky TTL, 50, 52–53
Segment register, 236
Serial data stream, 191–193
Serial loading, 108–110
Settling time, 489
Setup time, 98
Seven-segment decoder, 54
#Shift instruction, 319, 320
Shift register, 108–110
SHL control, 108–110
Sign bit, 83
Sign flag, 175, 179, 180–181, 287, 294–296
Signed binary number, 83, 272, 284, 289
Sign-magnitude number, 83
Single-precision number, 274
Single-step debouncer, 158–159
Sink, 52
6502 family, 214, 481–483
  addressing, 265, 332–335, 476, 477
  architecture, 230–231, 329
  arithmetic instructions, 276–277, 472, 478, 480–481
  conditional jump (branch) instructions, 345–346, 475, 479–480, 481
  CPU control instructions, 242, 471, 478, 480
  data transfer instructions, 242–245, 471–472, 478, 480
6502 family *(Cont.)*:
  flag instructions, 277–281, 472, 476–478, 480
  increment and decrement instructions, 473–474, 479, 481
  input-output instructions, 476, 480, 481
  interrupt instructions, 476, 480, 481
  logical instructions, 308–310, 472–473, 478–479, 481
  programming, 511
  rotate and shift instructions, 321–322, 473, 479, 481
  stack instructions, 475–476, 480, 481
  subroutine instructions, 366–369, 475, 480, 481
  test and compare instructions, 346, 474, 479, 481
  unconditional jump instructions, 344, 474, 479, 481
6800/6808 family, 214, 434–437, 443, 444
  addressing, 265–266, 335–336, 433
  architecture, 329, 632–633
  arithmetic instructions, 281–282, 285–286, 424–425, 438, 441
  conditional jump (branch) instructions, 348–349, 429–431, 440, 442
  CPU control instructions, 422, 437, 441
  data transfer instructions, 245–249, 423, 437, 441
  flag instructions, 282–285, 423–424, 433, 437–438, 441
  increment and decrement instructions, 428, 439, 442
  input-output instructions, 432, 441, 442
  interrupt instructions, 432, 441, 442
  logical instructions, 310–314, 425–426, 438–439, 441–442
  programming, 511
  rotate and shift instructions, 322–323, 426–427, 439, 442
  stack instructions, 431–432, 440–441, 442
  subroutine instructions, 369–370, 431, 440, 442
  test and compare instructions, 349, 428–429, 439, 442
  unconditional jump instructions, 347–348, 428, 439, 442
Small-scale integration, 48
Software, 3–4, 218
Software emulation program, 215
Source, 52
Source code, 221
Source program, 145
SSI, 48
#Stack, 195, 207–211, 228–229, 231, 233, 234, 236, 363–364
Stack pointer, 195, 207–208, 228–229, 231, 233, 234, 236, 363–364, 366–367, 369, 371, 373
Stack segment, 338
Standard TTL, 49–52
State diagram, 117
Static RAM, 133–134
Status register, 227–228, 231–234, 236
Store the accumulator, 176
Straight-line program, 218
String, 1
#Subroutine, 180, 219, 363–377
  branching vs., 364
  nested, 189–190, 364, 367, 369–371, 373–374
  pushing and popping registers, 366, 367–368, 370, 371, 374
#Subroutine *(Cont.)*:
  return instruction, 180, 210–211, 364–366
  stack and stack pointer, 363–364, 366–367, 369, 371, 373
Subtract instruction, 143, 150, 178, 198–199
Subtraction, 80–81, 86–87, 202, 275, 285–286, 292–293, 300
Subtraction-with-carry (borrow), 196, 275–276, 281
Successive-approximation method, 492–493
Sum-of-products circuit, 67–68
Switch, current, 487–488
Switch debouncer, 92–93
Synchronous counter, 113–114

T state, 146–151, 187
Temporary register, 175
Three-state RAM, 134
Three-state register, 121–122
Three-state switch, 121
Time delay, 189–190
Timing diagram, 91, 92, 94, 95
Timing signal, 36, 116
Timing state, 146–151
Toggle, 99–100, 102, 118
Totem-pole output, 49
Trace command, 293
Traffic light, 190–191
Trainer, microprocessor, 215
Transistor, 4
  inverter, 19
  latch, 90–91
  register, 4
Transistor-transistor logic, 48–63
Transparent latch, 95
Triple-precision number, 274
Tristate switch, 111–112
Truth table, 20, 21
  deriving logic circuit from, 68
  JK master-slave, 102
  Karnaugh maps from, 70–77
  transistor latch, 90–91, 94
TTL, 48–63, 120, 135–136, 497–498
2's complement, 83–87, 312, 331
Two-state design, 4–6

#Unconditional jump, 179, 180, 342
Universal logic circuit, 60
Unsigned binary number, 272, 284, 289–290
Up-down counter, 118

Virtual ground point, 485
Volatile RAM, 134

Weight:
  binary, 6
  decimal, 6
  hexadecimal, 11–12
Weighted resistors, 489
Word, 20, 208
Word comparator, 42–43
Word multiplexer, 60
Worksheet, 222
Worst-case TTL characteristics, 50–51
Write enable, 134

XNOR gate, 42
XOR gate, 37–42, 49
XOR instruction, 313, 315
XRA instruction, 184
XRI instruction, 184

Zero flag, 175, 179, 180–181, 275, 278–279, 283–284, 287–289, 296, 332
Zero page addressing, 333–334