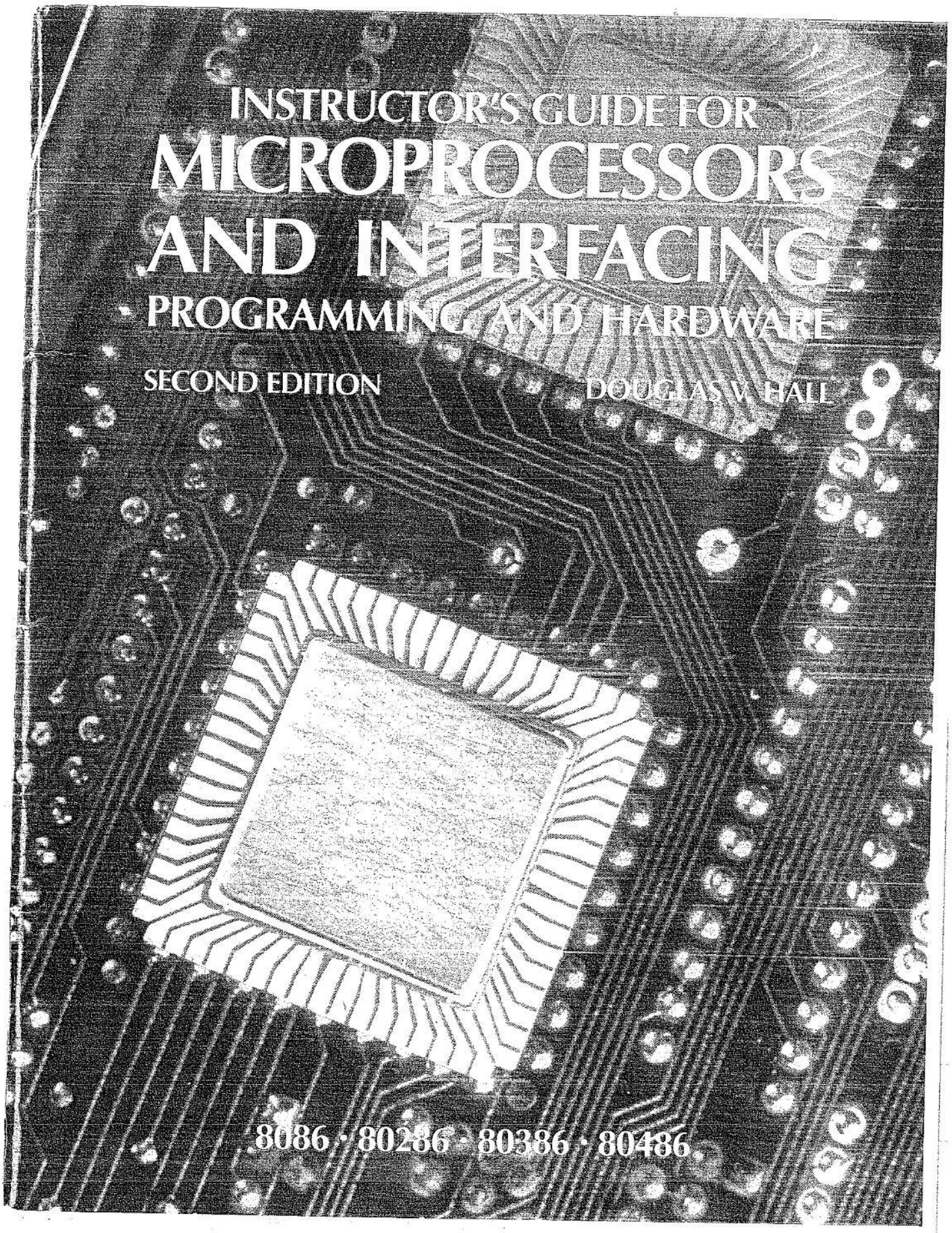

INSTRUCTOR'S GUIDE FOR
MICROPROCESSORS
AND INTERFACING
PROGRAMMING AND HARDWARE
SECOND EDITION
DOUGLAS V. HALL
8086 · 80286 · 80386 · 80486

INSTRUCTOR'S GUIDE FOR

MICROPROCESSORS AND INTERFACING

PROGRAMMING AND HARDWARE

SECOND EDITION

DOUGLAS V. HALL

GLENCOE
Macmillan/McGraw-Hill
Lake Forest, Illinois Columbus, Ohio Mission Hills, California Peoria, Illinois

Send all inquiries to:
GLENCOE DIVISION
Macmillan/McGraw-Hill
936 Eastwind Drive
Westerville, OH 43081

ISBN 0-07-025744-2

Printed in the United States of America.

1 2 3 4 5 6 7 8 9 SEM 99 98 97 96 95 94 93 92 91

CONTENTS

ANSWERS TO CHAPTER REVIEW QUESTIONS IN THE TEXT

EXPERIMENT NOTES AND ANSWERS TO SELECTED QUESTIONS IN THE EXPERIMENTS SUPPLEMENT

INTRODUCTION

In addition to the answers contained in this instructor's guide, I have prepared a set of IBM PC-compatible disks which contain the source files for:

1. All the example programs in the text.
2. All the program answers for end-of-chapter problems.
3. All the programs for the experiments, except those for a couple of open-ended exercises.
4. A program called SDKCOM1.EXE, which can be used to download binary programs to an SDK-86 board at 4800 Bd.

I think you will find that having these programs on disks is much more useful than simply having program listings, because:

1. You can assemble, link, download, and run these programs for demonstrations and/or further experimentation.
2. You can print out listings of selected programs as needed for use in class discussions.
3. You can give students part of the solution to a problem and let them fill in the missing pieces. This allows you to adapt the text problems and experiments to a wide range of student abilities. I initially give an assignment without hints; then, after students have worked on the problem for awhile, I give hints or program fragments as needed to allow the majority of students in a particular class to experience a feeling of success. This approach helps avoid the tendency of many students to wait until one of the "sharp" students figures out the solution and then just copy the solution from him or her without trying to solve the problem independently.
4. You can use these "known good" programs to test the hardware modules used in the laboratory exercises. This last use allows you to quickly deal with the common student complaint, "My program doesn't work; the hardware must be broken."

I have tried to make this instructor's guide as accurate as possible. If you find any errors, have any further questions, or have any suggestions for improving the book or the lab manual, I would appreciate hearing from you.

Notes about the Program Disks

1. The directory called TEXTXMPL contains the source code for all the program examples used in the text.
2. The directory called TEXTANS contains the source code for all the programs required as answers to problems.
3. The directory called LABANS contains the source code for all the programs for the experiments.
4. The SDKCOM1.EXE download program is in the root directory, so you can easily copy it to your hard disk. For all the systems I tried SDKCOM1.EXE works perfectly, but the older SDKDMP.EXE is included in <LABANS>. Use whichever program works better on your systems. Also, the source programs and .PRJ file for SDKCOM1 are in the LABANS directory if you want to experiment with improving the program.
5. These programs were all compiled with TASM, MASM, or the Turbo C++ Integrated Environment compiler. To the best of my knowledge, they all work. Please let me know about any problems you encounter.

Douglas V. Hall

Answers to Chapter Review Questions in the Text

CHAPTER 1 COMPUTER NUMBER SYSTEMS, CODES, AND DIGITAL DEVICES

1. $2^0 = 1$
   $2^1 = 2$
   $2^2 = 4$
   $2^3 = 8$
   $2^4 = 16$
   $2^5 = 32$
   $2^6 = 64$
   $2^7 = 128$
   $2^8 = 256$
   $2^9 = 512$
   $2^{10} = 1024$
   $2^{11} = 2048$
   $2^{12} = 4096$
   $2^{13} = 8192$
   $2^{14} = 16{,}384$
   $2^{15} = 32{,}768$
   $2^{16} = 65{,}536$
   $2^{17} = 131{,}072$
   $2^{18} = 262{,}144$
   $2^{19} = 524{,}288$
   $2^{20} = 1{,}048{,}576$

2. *a.* 22 decimal = 10110 binary
   b. 76 decimal = 1001100 binary
   c. 500 decimal = 111110100 binary

3. *a.* 1011 = 11 decimal
   b. 11010001 = 209 decimal
   c. 1110111001011001 = 61,017 decimal

4. *a.* 53 decimal = 35H
   b. 756 decimal = 2F4H
   c. 011 0110 0010 = 362H
   d. 110 0001 0111 = 617H

5. *a.* D3H = 211 decimal
   b. 3FEH = 1022 decimal
   c. 44H = 68 decimal

6. *a.* 86 decimal = 1000 0110 BCD
   b. 62 decimal = 0100 0010 BCD
   c. 33 decimal = 0011 0011 BCD

7. The L key produces the pattern 0101100 (4CH). A carriage return (ODH) would give the pattern 00001101.

8. The term parity is used to show if a data word has an even number of 1's. Even parity means an even number of 1's. Errors in transmitted data are detected as follows. The system sending a data word checks the parity of the word. If there is an odd number of 1's in the word, the system will set the parity bit (bit 7) to a 1; otherwise it will reset the parity bit. Either way, the parity of the word plus the parity bit becomes even. The data word is transmitted and the parity is checked at the other end. If the parity of the word is odd, it can be assumed that an error occurred in transmitting the data.

9. *a.* 10011 + 1011 = 11110
   b. 37 + 25 = 62

```
   0011 0111
 + 0010 0101
   0101 1100
 +      0110
   0110 0010
```

   add 6 because > 9

   c. 4AH + 77H = C1H

10.

	Decimal	8-Bit Magnitude	8-Bit Sign and Magnitude
a.	+26	0001 1010	0001 1010
b.	-7	0000 0111	1111 1001
c.	-26	0001 1010	1110 0110
d.	-125	0111 1101	1000 0011

11.

	Decimal	Pencil	2's Complement If Sign Bit = 0, Result +, Ignore Carry
a.	7	0111	0111
	- 4	- 0100	+ 1100
	3	0011	1 0011
b.	37	0010 0101	0010 0101
	- 26	- 0001 1010	+ 1110 0110
	11	0000 1011	1 0000 1011
c.	125	0111 1101	0111 1101
	- 93	- 0101 1101	+ 1010 0011
	32	0010 0000	1 0010 0000

12. *a.*
```
   1001          (9 × 3 = 27)
 × 011
   1001
  10010
  11011
```
   b.
```
   11010         (26 x 5 = 130)
 × 101
   11010
  11010
 10000010
```

13.
```
      1010       (100/10 = 10)
1010)1100100
     1010
      1010
      1010
      00000
```

14. *a.* 3AH + 94H = CEH

   b. 17AH - 4CH = 12EH

   c.
```
  0101 1001 BCD        (59 + 42 = 101)
+ 0100 0011 BCD
  1001 1011
+ 0110 0110    (add a correction of 6 6)
1 0000 0001
```
   d.
```
  0111 1001 BCD        (79 + 49 = 128)
+ 0100 1001 BCD
  1100 0010
+ 0110 0110    (add a correction of 6 6)
1 0010 1000
```

e.
```
  0101 1001 BCD      (59 - 26 = 33)
- 0010 0110 BCD
  0011 0011
```

f.
```
  0110 0111 BCD      (67 - 39 = 28)
- 0011 1001 BCD
  0010 1110
-      0110   (subtract a correction of 6)
  0010 1000
```

15. For the circuit in Figure 1-23:
- *a.* The Y output is active high.
- *b.* The C signal is active high.
- *c.* The Y output is asserted when A is high, B is low, and C is high; or when A is low, B is high, and C is low.

16. When the D latch is enabled, the Q output follows the D input. In the D flip-flop, the output only follows the D input on the rising edge of the clock signal.

17. Thirteen address lines are required to address one of the 8192 bytes in a INS8298. ($2^{13} = 8192$, 13 address lines, 0 through 12.)

18. Most ROMs and RAMs have three-state outputs (logic low, logic high, and floating high-impedance states) because usually more than one device is connected to a bus. Only one device at a time can have its outputs active on the bus; all others must be floating.

19.

		S3	S2	S1	S0	M
a.	A + B	0	0	0	1	0
b.	A - B - 1	0	1	1	0	0
c.	AB + A	1	0	0	0	0

20. *a.*
```
1010  AND  0111 = 0010
1010  OR   0111 = 1111
```
b.
```
1011  AND  1100 = 1000
1011  OR   1100 = 1111
```
c.
```
1101 0111  AND  0011 1000 = 0001 0000
1101 0111  OR   0011 1000 = 1111 1111
```
d. An 8-bit number ANDed with 1111 0000 will leave the upper 4 bits the same and zero (mask) the lower 4 bits.

CHAPTER 2 COMPUTERS, MICROCOMPUTERS, AND MICROPROCESSORS—AN INTRODUCTION

1. Advantages of a distributed processing system over a time-sharing system are as follows: If the large computer goes down, the local microcomputers can continue working until it is necessary to access the large computer; (b) the burden on the large computer is greatly reduced; and (c) the system can be designed to use a local microcomputer best suited to the task it has to do.

2. The sequence of signals is as follows:
   a. An address is sent out on the address bus.
   b. A memory-read signal is sent out on the control bus.
   c. The memory outputs the instruction byte on the data bus.
   d. The CPU reads in the instruction and decodes it.

3. The number of bits a microprocessor's ALU can work with at a time determines whether the microprocessor is considered an 8-bit, 16-bit, or 32-bit system.

4. a. The 8086 has 20 address lines.
   b. Twenty address lines allow the 8086 to access 1,048,576 memory addresses directly.
   c. Each segment contains 64K bytes within this 1M-byte range.

5. The 8088 has an 8-bit data bus, so it can read data from or write data to memory and ports only 8 bits at a time. The 8086 can read or write either 8 or 16 bits at a time.

6. a. The 8086 queue holds the next six instruction bytes from memory.
   b. The queue speeds up processing because the EU reads the next instruction from the queue. This is much faster than sending out an address to the system memory and waiting for the memory to send back the next instruction byte or bytes.

7. a. The CS register will hold 7040H.
   b. The next code byte fetched from the physical address = 70400 + 539CH = 7579CH.

8. a. 4370:561EH represents 43700H + 561EH = 48D1EH.
   b. 7A32:0028H represents 7A320H + 0028H = 7A348H.

9. The advantage of using a CPU register is that the data is already in the EU; therefore, it can be accessed much more quickly than it could in external memory.

10. SS = 3000H and SP = 8434H, TOS = 30000 + 8434H = 38434H or 3000:8434.

11. a. It is very difficult to memorize the thousands of binary instruction codes for a CPU and very easy for an error to occur when working with the long series of 1's and 0's required by machine language. Assembly language makes programming easier and less prone to error.
    b. When an 8086 executes the instruction ADD AX, BX, the contents of the BX register are added to the contents of the AX register and the result is left in the AX register. BX is left unchanged.

12. Programs which require a lot of hardware control or programs which must run as quickly as possible are usually best written in assembly language.

13. a. MOV BX, 03FFH ;Load the number 03FFH into the BX register.
    b. MOV AL, 0DBH ;Load the number DBH into the AL register.
    c. MOV DH, CL ;The contents of the CL register are copied into the DH register. CL is unchanged.
    d. MOV BX, AX ;The contents of the AX register are copied into the BX register. AX is unchanged.

14. a. MOV BP, 7986H
    b. MOV SP, BP
    c. MOV DS, AX
    d. MOV AL, 0F3H

15. EA = 14A3H and DS = 7000H, address produced by the BIU = 714A3H.

16. DS = 4000H, physical address for [234BH] = 40000 + 234BH = 4234BH.

17. MOV [4B2CH], DL

18. MOV AX, 2437H = loads the AX register with the number 2437H. MOV AX, [2437H] copies the contents of memory location DS + 2437H into AL and the contents of memory location DS + 2437H + 1 into AH (i.e., loads the 16-bit AX register from two 8-bit memory locations).

CHAPTER 3 8086 FAMILY ASSEMBLY LANGUAGE PROGRAMMING—INTRODUCTION

1. The main steps in developing an assembly language program are:
   a. Define the problem.
   b. Write down, in general terms, the algorithm for what the program has to do, using a flowchart, a sequential list, or pseudocode (i.e., represent the structure of the program in some way that is very clear to you and to anyone else who might have to work on the program).
   c. Write an initialization checklist for your program.
   d. Determine the instruction statements required to do each part of the program.
   e. Start writing the assembly language for the program.
2. The top-down design approach produces easy to understand, modular program representations that may be programmed in either high- or low-level languages.
3. A detailed algorithm enables you to break a programming problem into small modules which can easily be written, tested, and debugged. The more time you spend organizing your programs, the less time it will take you to write and debug them.
4. *a.* Sequence, Selection, and Repetition (or Sequence, IF-THEN-ELSE, and WHILE-DO).
   b. The advantage of using only these structures when writing the algorithm for a program is that the algorithm is easy to understand, implement, and debug (avoids spaghetti).
5. Pseudocode:
   Look in your kitchen for peanut brittle recipe ingredients.
   IF any of the ingredients are missing THEN go to the store and buy the missing ingredients.
   Find a teaspoon, a 1-cup measure, a 1.5-quart casserole, and a nonstick cookie sheet.
   Measure 1 cup of sugar and pour it into the casserole.
   Measure 0.5 cup of corn syrup and pour it into the casserole.
   WHILE not mixed DO stir the contents of the casserole.
   Microwave on HIGH for 4 minutes.
   WHILE microwave still cooking DO sit and read newspaper.
   Add peanuts.
   REPEAT stir the contents of casserole UNTIL mixed.
   Microwave on HIGH for 4 minutes.
   WHILE microwave still cooking DO look out of the window.
   Add 1 teaspoon of butter.
   Add 1 teaspoon of vanilla.
   REPEAT stir contents of the casserole UNTIL mixed.
   Microwave on HIGH for 2 minutes.
   WHILE microwave still cooking DO start to clean up.
   Add 1 teaspoon of baking soda.
   WHILE not light and foamy DO gently stir mixture.
   Pour mixture onto nonstick cookie sheet.
   WHILE minutes passed < 60 DO go do something else.
   IF mixture not cool THEN wait longer
   ELSE break into pieces.
   Eat peanut brittle.
6. *a.* Pseudocode: Get number from memory location and put into a register. Subtract 20H from the number in the register. IF number in register > 25H THEN output 01H to port 3AH.
   b. Flowchart: See Figure 3-1.

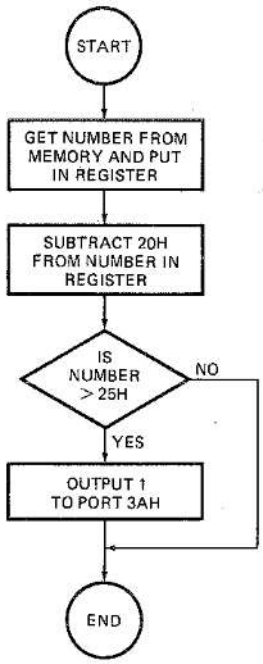


FIGURE 3-1. Flowchart for algorithm for question 6.

7. *a.* Next instruction fetched from 4000:43E8H or 443E8H.
   b. TOS address = 7000:0000 or 70000H.

8. *a.* MOV AX, BX;Copy the contents of BX to AX. AX = BX = 075AH.
   b. MOV CL, 37H;Load CL with 37H. CL = 37H.
   c. INC BX;Increment BX. BX = 075BH.
   d. MOV CX, [246BH];Copy the contents of memory at 5246BH and 5246CH into the CX register.
   e. MOV CX, 246BH;Load CX with 246BH.
   f. ADD AL, DH;Add contents of DH to AL and leave result in AL. AL = 35 + 33 = 68H, DH = 33H.
   g. MUL BX;Multiply the contents of AX with BX; the result is the high word in DX and the low word in AX. DX = 01E6H, AX = B9A2H.
   h. DEC BP;Decrement the contents of BP. BP = 2467H.
   i. DIV BL;Divide AX by BL, quotient in AL, remainder in AH. AX = BC3BH.
   j. SUB AX, BX;AX - BX; result in AX = 4235 - 075A = 3ADBH.
   k. OR CL, BL;CL is ORed with BL; result in CL = 5EH.
   l. NOT AH;Complement AH, AH = 1011 1101.
   m. ROL BX, 1;BX is rotated left once. BX was 0000 0111 0101 1010, BX now 0000 1110 1011 0100, CF = 0.
   n. AND AL, CH;AL is ANDed with CH; AL = 00.
   o. MOV DS, AX;Initialize the data segment at 4235H; copy contents of AX into DS.
   p. ROR BX, CL;Rotate the contents of the BX register right, 4 times (CL = 4). BX was 0000 0111 0101 1010, BX now 1010 0000 0111 0101, CF = 1.
   q. AND AL, OFH;AL is ANDed with OFH; AL = 05H.
   r. MOV AX, [BX];Copy the contents of memory in data segment, pointed to by BX, into the AX register. AX = contents of 5075AH and 5075BH.
   s. MOV [BX][SI], CL;Copy the contents of CL into memory in data segment at an offset of BX + SI. Memory at 5535AH will contain 04H.

9. *a.* MOV BH, AX;Register size doesn't match. Should be MOV BX, AX or MOV BH, AL or MOV BH, AH.
   b. MOV DX, CL;Register size doesn't match.
   c. ADD AL, 2073H;Number is too large for AL. Should be ADD AX, 2073H.
   d. MOV 7632H, CX;Cannot move a register's contents into an immediate number. Decide if you want to MOV CX, 7632H or if you want to copy the contents of CX to memory.
   e. IN BL, 04H;Can only transfer from a port to the AL or AX registers.

10. *a.*
```
ADD BL, Al      ;BL = 5A + 35 = 8FH
MOV [0004], BL  ;50004H = 8FH
```
   b.
```
MOV CL, 04      ;CL = 04
ROR DI, CL      ;DI = 07D0H
```
   c.
```
ADD AL, BH      ;AL = 35H + 07H = 3CH
DAA             ;AL = 42H
```
   d.
```
MOV BX, 000AH   ;BX = 000AH
MOV AL, [BX]    ;AL = contents of 5000AH
                = 7CH
SUB AL, CL      ;AL = 7C - 04 = 78H
INC BX          ;BX = 000BH
MOV [BX], AL    ;5000BH = 78H
```

11. *a.*
```
MOV BL, AL      ;Copies AL to BL
```
   b.
```
MOV CL, 43H     ;Loads CL with 43H
```
   c.
```
INC CX          ;Increments CX by one.
```
   d.
```
MOV BP, SP      ;Copies SP to BP
```
   e.
```
ADD DL, 07H     ;Adds 07 to DL
```
   f.
```
MUL BL          ;Multiples AL times BL,
                result in AX
```
   g.
```
MOV [245AH], AX ;Copies AX to a memory
                location
                ;at offset of DS:245AH
```
   h.
```
DEC SP          ;Decrements SP by one
```
   i.
```
ROL AL, 01      ;Rotates the MSB of AL
                ; into the LSB of AL
```
   j.
```
MOV [BX], DL    ;Copies DL to memory
                location
                ; pointed to by BX
```
   k.
```
AND BL, 0F0H    ;Mask lower 4 bits of BL
```
   l.
```
OR AX, 80H      ;Sets the MSB of AX to 1,
                ; doesn't affect other bits
```
   m.
```
XOR AX, OFH     ;Inverts lower 4 bits of AL
                ; but doesn't affect other
                  bits
```

12.

	Instruction	Binary Code	Hex
a.	MOV BL, AL	100010 10 11 011 000	8A D8
b.	MOV [BX], CX	100010 01 00 001 111	89 OF
c.	ADD BX, 59H [DI]	000000 11 01 011 101 59H	03 5D 59
d.	SUB [2048H], DH	001010 00 00 110 110 48H 20H	28 36 48 20
e.	XCHG CH, ES:[BX]	00100110 100001 10 00 101 111	26 86 2F
f.	ROR AX, 1	110100 01 11 001 000	D1 C8
g.	OUT DX, AL	11101110	EE
h.	AND AL, 0FH	00100100 0FH	24 0F
i.	NOP	10010000	90
j.	IN AL, DX	11101100	EC

13. DATA_HERE SEGMENT to DATA_HERE ENDS sets up a logical segment for the data called DATA_HERE.

PRESSURE DB 0 assigns the name PRESSURE to a byte type variable and initializes that variable with the value of zero. PRESSURE is a variable in the DATA_HERE segment.

PRESSURE_PORT EQU 04H and CORRECTION_FACTOR EQU 07H gives the values of 04 and 07 to the constants PRESSURE_PORT and CORRECTION_FACTOR.

CODE_HERE SEGMENT and CODE_HERE ENDS sets up a logical segment called CODE_HERE for the code in the program.

ASSUME CS:CODE_HERE, DS:DATA_HERE tells the assembler which code and data segments are to be used.

MOV AX, DATA_HERE and MOV DS, AX initializes the data segment register with the starting address of DATA_HERE.

IN AL, PRESSURE_PORT brings a byte of data into the AL register through the port addressed by PRESSURE_PORT (04H).

ADD AL, CORRECTION_FACTOR AND MOV PRESSURE, AL adds a correction of 07H to the contents of AL, and it is then copied to the variable called PRESSURE in the data segment called DATA_HERE.

The last line, END, tells the assembler there are no more instructions to assemble.

14. An assembly language program is developed and debugged using system tools in the following way:

a. Build an algorithm for your program.
b. Decide on the instructions required to build the program.
c. Build the assembly language source file in the correct format using an editor.
d. Assemble the source file by using an assembler which produces an object and list file.
e. If necessary, correct the source program using the editor.
f. Print out the assembler list file.
g. Use a linker which joins together several object files, if necessary, and produces a link file which contains the binary code for the program.
h. If necessary for your system, use a locator program to assign the specific addresses in memory for the object code.
i. If your program requires no external hardware, or requires only hardware accessible directly from your system, use a debugger to run and debug your program.
j. If your program is intended to work with external hardware, use an emulator to run and debug your program.

15. Pseudocode for flowchart:

```
REPEAT
    REPEAT
    Create source file with editor
    Assemble source file
    UNTIL no assembly errors
Link
Locate
IF external system THEN
    Load emulator
    Load program
    Run and test program
    IF errors THEN
        use emulator tools to find errors
    ELSE do nothing
ELSE Load debugger
    Load program
    Run and test program
    IF errors THEN
        use debugger tools to find errors
    ELSE do nothing
UNTIL no errors
STOP
```

CHAPTER 4 IMPLEMENTING STANDARD PROGRAM STRUCTURES IN 8086 ASSEMBLY LANGUAGE

1. *a.* ROL AX, CL; The accumulator is rotated left twice (CL = 02), which leaves AX = 901EH.

b. IN AL, DX; A byte is copied to AL from port DX (FFFAH).

c. MOV CX, [BX]; The contents of memory in the data segment at addresses 324B3H and 324B4H are moved into CX.

d. ADD AX, [BX][SI]; The contents in the data segment of memory at address 366B3H (BX + SI = 24B3 + 4200 = 66B3H, which is then added to 30000 to give the physical address) and address 366B4H are added to the AX register. The result is left in the AX register.

e. JMP 023AH; The contents of the instruction pointer are replaced with the value 023AH, and the program execution jumps to address 2023AH.

f. JMP BX; The contents of the instruction pointer are replaced with the contents of the BX register, and the program execution jumps to address 224B3H.

2.

	Instruction	Binary Code	Hex
a.	ROL AX, CL	110100 11 11 000 000	D3C0
b.	IN AL, DX	11101100	EC
c.	MOV CX, [BX]	100010 11 00 001 111	8B0F
d.	ADD AX, [BX][SI]	000000 11 00 000 000	0300
e.	JMP 023AH	11101001 0011 1010 0000 0010	E93A02
f.	JMP BX	111111 11 11 100 011	FFE3

3.

	Instruction	Flag Contents					
		OF	SF	ZF	AF	PF	CF
a.	MOV AL, AH: MOV 07, A4	0	0	0	0	0	0
b.	AND BL, CL: AND B3, 02	0	0	0	?	0	0
c.	ADD CL, DH: ADD 02, FF	0	0	0	1	0	1
d.	OR CX, BX: OR 0002, 24B3	0	0	0	?	0	0

4. *a.* Each time the loop is executed, BL is loaded with 72H. Therefore, BL never reaches zero and execution never leaves the loop.

b. Cannot mix bytes and words in instructions. Use either CX, AX or CL, AL, or CH, AL.

c. JMP needs a word, or for an indirect jump address, BL is only a byte.

d. JNZ can only go to a label in the range +127 to -128 from current IP; cannot use indirect addressing modes.

5. *a, b.* See the program A04-05B.ASM.

c. To add 2 BCD bytes, add the instruction DAA after the line ADD AL, NUM2.

6. See program A04-06.ASM.

7. See program A04-07.ASM.

8. See program A04-08.ASM.

9. See program A04-09.ASM.

10. See program A04-10.ASM.

11. See program A04-11.ASM.

12. See program A04-12.ASM.

13. See program A04-13.ASM.

14. *a.* Time for one clock cycle = 1/5MHz = 0.2 μs
Number of clock cycles required = 500μsec/0.2μsec = 2500
The following code is used for the delay loop:

```
MOV CX, N      ;          ;4 clock cycles
KILL_TIME:NOP  ;          ; 3 clock cycles
LOOP KILL_TIME ;          ; 17 or 5 clock cycles
```

$N = (C_T - C_o + D)/C_l$
C_o = overhead cycles = 4
C_l = number of cycles in loop = 20
C_T = total number of cycles wait = 2500
D = difference between LOOP executing or not = 17 - 5 = 12.

Therefore:
$N = (2500 - 4 + 12)/20 = 125.4 = 07DH$

b. See program A04-14.ASM.

CHAPTER 5 STRINGS, PROCEDURES, AND MACROS

1. *a.* See program A05-01A.ASM.
   b. See program A05-01B.ASM.

2. See program A05-02.ASM.

3. *a.*
```
MOV AX, 4000H
MOV SS, AX
MOV SP, 8000H
```
   b.
```
CALL FIXIT    ;Call a near procedure, FIXIT.
```
   c.
```
PUSH BX       ;Save BX and BP
PUSH BP       ;at start of procedure
POP BP        ;At end of procedure,
POP BX        ;notice order of POPs
```
   d.
```
RET 8         ;Return from a procedure and
               increment SP by 8.
```

4. *a.* See Figure 5-1.
   b. If the POPF instruction was accidentally left out of the procedure, the RET instruction would place the contents of the saved flag registers into the IP register. This means that execution would continue, not just after the place where the procedure was called but at some other place. One way to track down this problem is to keep a map of the stack. If you make a mistake with the stack map, you can run the program to the end of the procedure and then examine the registers to see if the RET instruction occurred at the correct SP. Another way to debug this type of problem is to use the single-step facility of the debugger, although this takes much longer.

5. *a.* The binary code for the instruction which will call a procedure which is 97 addresses higher in memory than the call instruction is 11101000 01011111.
   b. RET 4; 11000010 00000100 00000000

6. *a.* Three methods of passing parameters to a procedure include:
   i. Using general memory and named memory locations. One advantage of this method is that it makes programs easier to understand. The disadvantage is that you cannot use this method for procedures that call themselves or from procedures that will be called from a higher-level language.
   ii. Using registers to pass values of variables. The advantage of this method is speed—the parameters are immediately available in the procedure. The disadvantage is that you cannot pass a large number of parameters unless you use a register to just hold a pointer to, for example, an array of data in memory. Note: Recursive and reentrant procedures can use registers, because the register contents will be pushed on the stack during each call. Also note that some higher-level languages for specific machines do allow a few parameters to be passed in registers.
   iii. Using the stack to pass values of variables. One advantage of this method is that it does not depend on the architecture of the specific processor, so it helps make a program "portable." Also, this method can be used for reentrant procedures. The disadvantages are the time overhead involved and the complexity of using a pointer and an offset to access the passed parameters.

```
4000    SP          SP initialized by MOV SP, 4000H
3FFF    AH
3FFE    AL          SP - 2 then PUSH AX. AL at SP, AH at SP+1
3FFD    IP HIGH
3FFC    IP LOW      SP - 2 then PUSH IP for CALL MULTO
3FFB    F HIGH
3FFA    F LOW       SP - 2 then PUSH F
3FF9    BH
3FF8    BL          SP - 2 then PUSH BX

3FF9    POP BX      BX    =  contents of 3FF8 & 3FF9
3FFB    POPF        Flags =  contents of 3FFB & 3FFA
3FFD    RET         IP    =  contents of 3FFD & 3FFC
3FFF    POP AX      AX    =  contents of 3FFF & 3FFE
```

FIGURE 5-1. Stack map for question 4.

b. The term reentrant means a procedure can be interrupted by another procedure at any time, used by the interrupting procedure, and then pick up where it left off in its first execution. You must use the stack or registers to pass parameters to a reentrant procedure.

7. See program A05-07.ASM.

8. See program A05-08.ASM.

9. For the algorithm, see Figure 5-28 on page 130 of the text. See program A05-09.ASM.

10. See program A05-10.ASM.

11. *a.* To tell the assembler to make the label BINADD available to other assembly modules, the statement PUBLIC BINADD would have to be added to the part of the program calling the label.

    b. To tell the assembler to look for a byte type data item named CONVERSION_FACTOR in some other assembly module, you would have to add the statement EXTRN CONVERSION_FACTOR:BYTE, surrounded by the name of the data segment it was declared in. For instance, if it was declared in the data segment DATA_HERE, you would have to add the lines:

```
DATA_HERE SEGMENT PUBLIC
   EXTRN CONVERSION_FACTOR : BYTE
DATA_HERE ENDS
```

12. *a.*

```
POP_ALL MACRO
POP SS
POP ES
POP DS
 POP DI
 POP SI
 POP BP
 POP DX
 POP CX
 POP BX
 POP AX
 POPF
ENDM
```

b. POP_ALL

THERE ARE NO QUESTIONS FOR CHAPTER 6.

CHAPTER 7 8086 SYSTEM CONNECTIONS, TIMING, AND TROUBLESHOOTING

1. The start of an 8086 state is measured from the 50 percent point of the falling edge of the clock waveform.

2. Latches hold the address information so that the 8086 can float the address lines and then use them for data input and output.

3. The ALE high signal is used to enable the latches so that the address information is passed through to the latch outputs. When ALE goes low, the address information is held on the latch outputs.

4. The sequence of events on the 8086 data/address bus and the ALE, M/$\overline{IO}$ and $\overline{RD}$ lines as the 8086 fetches an instruction word is:
   a. M/$\overline{IO}$ goes high so that memory is accessed.
   b. ALE goes high to enable latches.
   c. An address is sent out on the data/address bus.
   d. ALE goes low, the address is latched on the latch outputs, the latches are disabled, and the data/address bus is floated.
   e. $\overline{RD}$ goes low, causing the addressed device to put data on the bus.
   f. Data comes in on the data/address bus.
   g. $\overline{RD}$ goes high.
   h. The word or byte is read, depending on $\overline{BHE}$ and AO.

5. When writing to memory, $\overline{WR}$ = 0, $\overline{RD}$ = 1, and M/$\overline{IO}$ = 1. When reading from a port, $\overline{WR}$ = 1, $\overline{RD}$ = 0, and M/$\overline{IO}$ = 0.

6. Minimum mode is used when there is only one microprocessor on the system buses. Maximum mode is used when two or more microprocessors share the system buses.

7. When RESET = 1, DS, SS, ES, IP, and the flag registers = 0, and CS = FFFFH.

8. Buffers are often needed on the address/data lines and control buses in a microprocessor system because as more devices are added to the system, each device input or output acts like a capacitance of a few picofarads connected to ground. To change the logic states up/down, all of the added capacitance has to be charged/discharged. The 8086 cannot supply enough current to charge or discharge the circuit capacitance rapidly, so high-current drive buffers are added to do the job.

9. a. If the READY input is made low, the 8086 will enter a WAIT state.
   b. An 8086 enters a WAIT state after the falling edge of T_3.
   c. Information on the buses remains the same during a WAIT state as it was at the start of the WAIT state.
   d. The 8086 stays in the WAIT state as long as READY = 0.
   e. The 8086 can have any number of WAIT states.
   f. WAIT states are needed if the system contains devices which cannot respond fast enough for the processor.

10. The DT/$\overline{R}$ signal is used to specify the direction in which the buffers are enabled. DT/$\overline{R}$ high + $\overline{DEN}$ low = transmit data to memory/ports. DT/$\overline{R}$ low + $\overline{DEN}$ low = receive data from memory/ports.

    The $\overline{DEN}$ signal is used to enable the bidirectional buffers on the data bus.

11. The transition at the tail of the arrow causes the transition at the tip.

12. See Chapter 7 in the text and Figure 7-4 on text page 169. The clock function tells the analyzer how often to take samples. The trigger tells the analyzer when to display the samples stored in its internal RAM.

13. For detailed timing measurements, use the logic analyzer's internal clock.

14. a. Use the falling edge of the ALE signal and trigger on address FFFF0H to see the sequence of addresses output after a RESET.
    b. Use the rising edge of $\overline{RD}$ as a clock signal and trigger on instruction 0EAH to see the sequence of instructions read in after a RESET.
    c. Use the clock signal as clock (falling edge) to see both the addresses sent out and the words read in.
    d. Clock on the rising edge of $\overline{RD}$, clock qualifier input connect to M/$\overline{IO}$. Set the analyzer to accept clocks when M/$\overline{IO}$ is low.

15. It is possible for a logic analyzer to display data that occurred before the trigger because whenever it receives a clock, it stores samples in its internal RAM. The trigger position specification is used to tell the analyzer to display samples pre-trigger, post-trigger, or center-trigger.

16. Wire-wrap jumpers are shown as, for instance, W27.

17. The /8 means that this one line represents 8 lines.

18. Address decoders in a microprocessor system are used to:
    a. Produce a signal which selects the device to be used when you send out an address in the range assigned for that device.
    b. Make sure that only one device at a time is outputting onto the data bus.

19. A memory device with 15 address lines and 8 data outputs connected to it will store 32,768 bytes or 8-bit words.

20. 8255As give the SDK-86 programmable parallel ports which can be used individually to input/output parallel bytes or together to input/output words (high byte to P1A, low byte to P2A). The 8251A is a UART used for serial data communication. The 8279 is a keyboard/display interface device.

21. 2ZB3 indicates that the lines come from zone B3 on sheet 2 of the schematics.

22. J identifies a jack which a connector plugs into. P identifies a plug which plugs into another connector.

23. The small capacitors are used to bypass high-frequency noise on the power supply lines (produced by devices switching from one logic level to another). The large capacitors filter out the low-frequency noise on the power lines.

24. To enable the 74S138 you need $\overline{G2A}$ (A15) low, $\overline{G2B}$ ($\overline{RD}$) low, and G1 (+5V) high. The SELECT inputs are A12, A13, and A14. Figure 7-1 shows the address decoder worksheet.

    $\overline{RD}$ is used as one of the enable inputs of the ROM decoder to prevent writing to ROM. The decoder is disabled if $\overline{RD}$ is high.

25. The memory map for ROM in Problem 24 is:

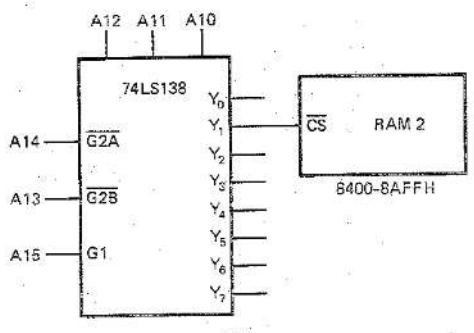
    7000 - 7FFFH
    6000 - 6FFFH
    5000 - 5FFFH
    4000 - 4FFFH
    3000 - 3FFFH
    2000 - 2FFFH

Address line numbers																Selected
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Blocks
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000 - 0FFFH
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000 - 1FFFH
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000 - 2FFFH
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	3000 - 3FFFH
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000 - 4FFFH
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5000 - 5FFFH
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6000 - 6FFFH
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	7000 - 7FFFH
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFF

FIGURE 7-1. Address decoder worksheets for question 24.

Address line numbers																Selected
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Blocks
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000 - 83FFH
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8400 - 87FFH
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	8800 - 8BFFH
1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	8C00 - 8FFFH
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	9000 - 93FFH
1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	9400 - 97FFH
1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	9800 - 9BFFH
1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	9C00 - 9FFFH

(a)



(b)

FIGURE 7-2. Address decoder worksheet and circuit for question 26.

1000 - 2FFFH
0000 - 0FFFH

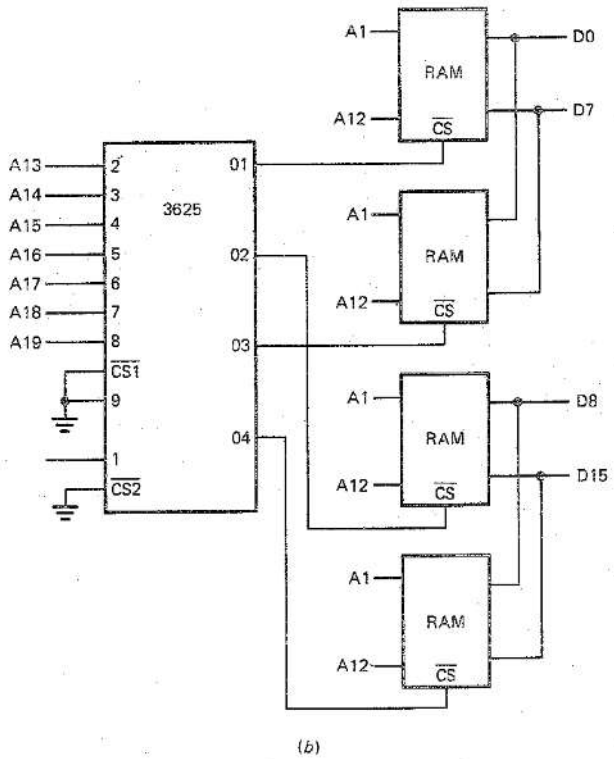
26. See Figure 7-2 for the address decoder worksheet and the circuit.

27. Many addresses will select one of the port devices connected to the port decoder in Figure 7-12 *a* because A6–A11 are not connected to the port device or the decoder. They have no effect on selecting a port, so they can have any value.

28. Memory-mapped I/O is done by using a decoder which translates memory addresses to chip select signals for port devices. The advantage of this is that any instruction which references memory can be used for I/O to ports. The disadvantage is that some system memory address space is used up for ports and is therefore not available for memory. Direct I/O uses a separate address space for I/O. It has the advantage of not using the system memory address space and the disadvantage that only IN and OUT instructions can be used for I/O of data.

A19–A15	A14	A13	BHE	A0	O4	O3	O2	O1	Address Block Bytes select
0	0	0	0	0	1	1	0	0	
0	0	0	0	1	1	1	0	1	0 - 1FFF
0	0	0	1	0	1	1	1	0	
0	0	1	0	0	0	0	1	1	
0	0	1	0	1	0	1	1	1	2000 - 3FFF
0	0	1	1	0	1	0	1	1	

(*a*)



(*b*)

FIGURE 7-3. Truth table and circuit for question 32.

29. *a.* The memory is set up as 2-byte-wide banks so that it is possible to read/write either words or individual bytes.

b. When writing a byte to 04274H, $\overline{BHE}$ = 1 and A0 = 0. When writing a word to 04274H, $\overline{BHE}$ = 0 and A0 = 0.

c. Two machine cycles required to write a word to address 04373H are as follows:
   i. During the first machine cycle, the 8086 will output the address 04373, make $\overline{BHE}$ = 0, and A0 = 1. A byte will be written to 04373, $\overline{WR}$ = 0 and M/$\overline{IO}$ = 1.
   ii. During the second machine cycle, the 8086 outputs the address 04374, makes $\overline{BHE}$ = 1, A0 = 0. The second byte will be written to 04374, $\overline{WR}$ = 0, and M/$\overline{IO}$ = 1.

30. You cannot accidentally write a byte or word to ROM because $\overline{RD}$ = 0 enables the 3625 PROM decoder. If $\overline{RD}$ = 1, the decoder is disabled and ROM cannot be selected.

31. Some ROM is put at the top of the address space because the 8086, when reset, goes to address FFFF0H to get its first instruction.

32. *a.* For the truth table and circuit, see Figure 7-3.

33. Reading a word from ports FFF8H and FFF9H requires the logic levels shown on the following signals:

$\overline{BHE}$ = M/$\overline{IO}$ = $\overline{RD}$ = 0 $\quad$ $\overline{WR}$ = 1

A0–A2 = 0

A3–A15 = 1

A16–A19 = 0

Ports are direct I/O.

The instructions to do the read operation are:

```
MOV DX, 0FFF8H
IN AX, DX
```

34. *a.* The $\overline{OFF\ BOARD}$ signal will be asserted low if the 8086 is doing a memory operation and the address sent is not in one of the ranges decoded for the on-board RAM/ROM.

b. The purpose of the $\overline{OFF\ BOARD}$ signal is to enable buffers for devices added to the expansion area of the SDK-86 board.

35. There is only one memory bank in the 8088 system. Because the 8088 has only one memory bank which contains both odd and even addresses, the $\overline{BHE}$ signal is not necessary.

36. *a.* Maximum clock frequency is 8 MHz.
b. TCLRL = 100 ns.
c. TCLDX (Data Hold Time) = 10 ns.
d. TLLAX = TCHCL - 10 = 44 - 10 = 34 ns.

37. For the 27128-25:

t_{ACC} = 250 ns; t_{CE} = 250 ns; t_{OE} = 100 ns

Address latches propagation delay = T_p = 12 ns

Decoder propagation delay = T_d = 30 ns

For the 8086-2:

$T_1 = T_2 = T_3$ = state time = 125 ns

TCLAV = 60 ns

TDVCL = 20 ns

TCLRL = 100 ns

Time available for t_{ACC} or t_{CE}:

$T_1 + T_2 + T_3$ - (TCLAV - T_p - TDVCL)

= 375 - 92 = 283 ns.

This time is greater than t_{ACC} or t_{CE} required by the 27128-25.

Time available for t_{OE}:

$T_1 + T_2$ - (TCLRL - T_d + TDVCL)

= 250 - 150 = 100 ns.

This time equals t_{OE}, so it is not considered an adequate available time for memory access for the 27128-25. The device may not work correctly in the system. A WAIT state can be inserted to ensure proper operation.

38. To troubleshoot the microcomputer system:

a. Identify the symptoms.

b. Make a careful visual and tactile inspection.

c. Check the power supply.

d. Check the control signals such as $\overline{RD}$, $\overline{WR}$, ALE, RDY, and RESET.

e. Use a working system to systematically test the ICs from the nonworking system.

39. An oscilloscope checks the power supply for excessive noise or ripple.

40. Mark each IC from the good system with a wide-tip, water-soluble marking pen.

41. See Figure 7-4*a* for a routine to test the system RAM at addresses 0200–07FFH.

42. See Figure 7-4*b* for a routine to output alternating patterns of all 1's and all 0's to port FFF9H over and over.

43. *a.* If pin 8 of A15 (2ZD5) is stuck low, the processor is in a NOT READY state and will keep inserting WAIT states. The signals on the buses will all be at constant logic levels.

b. If the RESET key is stuck on, DS, ES, SS, IP, and the flag registers will be zero and CS will be FFFFH. There will be no activity on the buses.

c. If none of the outputs of A29 (6ZD7) ever go low, nothing will be read from or written to RAM. RAM is never selected.

d. If pin 6 of A3 (5ZA5) is stuck low, off-board devices will never be accessed because $\overline{OFF\ BOARD}$ is always high.

FIGURE 7-4. Routines to test system RAM.
(a) For question 41.
(b) For question 42.

```
;A "one" is written to the RAM at a specific address and the
;contents of the RAM are then read back to see if it is still
;one. If 0 is detected, the routine ends and leaves an error
;flag set.

DSEG SEGMENT   WORD PUBLIC
     ERROR_FLAG      DB   00
     MEM_LOCS        DB   600H DUP(?)
DSEG ENDS

CSEG SEGMENT   WORD PUBLIC
     ASSUME CS:CSEG, DS:DSEG
START:MOV  AX, DSEG                ; Initialize data segment
      MOV  DS, AX
      MOV  BX, OFFSET MEM_LOCS     ; Start at beginning
      MOV  CX, 600H                ; Size of memory to test
NEXT: MOV  BYTE PTR[BX], OFFH      ; Put in 1's in location
      CMP  BYTE PTR[BX], OFFH      ; Is it still 1's?
      JNE  ERROR
      INC  Bx
      LOOP NEXT                    ; Continue until done
      JMP  FIN
ERROR:MOV  ERROR_FLAG, OFFH        ; Leave a error message
      FIN:
CSEG  ENDS
      END
```

(a)

```
;A test routine to output alternating all ones and all
;zeros to port FFF9H over and over

CSEG SEGMENT   WORD PUBLIC
       ASSUME CS : CSEG
START: MOV  DX, OFFF9H
       MOV  AL, 00
AGAIN: OUT  DX, AL
       NOT  AL          ; flip all bits
       JMP  AGAIN       ; keep going for ever
CSEG   ENDS
       END
```

(b)

CHAPTER 8 8086 INTERRUPTS AND INTERRUPT APPLICATIONS

1. The 8086 will push the flag register on the stack, disable the single-step function and the INTR input, and do an indirect far call to the start of the interrupt procedure that was written to respond to the interrupt.

2. The 8086 interrupt-vector table stores the starting addresses of the interrupt-service procedures.

3. Type 2 interrupt-vector IP at 00008H, CS at 0000AH.

4. Type 4 interrupt-service procedure starting address is 0010:0082. Put 0010 at address 00012H and 0082 at address 00010H.

5. *a.* The locations correspond to a type 32 interrupt.

   b. The starting address is 0040:4A24 (04E24H). See Figure 8-2 on page 208 of the text.

6. A type 0 interrupt is caused when the quotient from a DIV or IDIV operation is too large to fit in the result register. Dividing by zero is a special case of this.

   A type 1, single-step interrupt is enabled when the TRAP flag is set.

   A type 2, nonmaskable interrupt results when the 8086 receives a low to high transition on its NMI input pin.

   A type 3, breakpoint interrupt is caused by the execution of the INT 3 instruction.

   A type 4, overflow interrupt is caused when an INTO instruction executes after an overflow error sets the overflow flag. If the overflow flag is not set, INTO acts as a NOP.

7. PUSH and POP all the registers used in the procedure so that the interrupted program will resume with its registers the same as they were before the interrupt.

8. You should use an IRET instruction because, as part of its interrupt response, the 8086 pushes the flag register on the stack and IRET pops the flag register and the return address off the stack. RET pops only the return address.

9. The assembler directive and instructions are shown in Figure 8-1.

10. *a.* The main use of the 8086 type 1 interrupt is to implement single stepping in a debug program.

    b.
```
PUSHF                   ;Push flags on stack.
MOV BP, SP              ;Copy SP to BP for use as
OR [BP + 0], 0100H      ;an index. Set the TF bit.
POPF                    ;Restore the flag register.
```

11. The mainline program is in A08-11A.ASM; the procedure is in A08-11B.ASM.

12. *a.* The 8086 INTR input is disabled to allow you to initialize ports, timers, registers, and interrupt controllers without being disturbed by interrupts.

    b. The INTR input is enabled by setting the IF with the STI instruction.

```
INT_PROC_HERE   SEGMENT WORD    PUBLIC
   EXTRN  DIV_0_ERROR:FAR  ; Let assembler know procedures
   EXTRN  POWER_FAIL :FAR  ; DIV_0_ERROR and POWER_FAIL are in
INT_PROC_HERE   ENDS       ; another assembly module

CODE_HERE       SEGMENT
        ASSUME CS:CODE_HERE
        MOV AX, 0000        ; Clear the ES register
        MOV ES, AX

;Store address for DIV_0_ERROR procedure at addresses 0000:0000
;Addresses 00000 to 00003 are where type 0 interrupt gets interrupt
;service procedure address. CS at 00002 & 00003, IP at 00000 & 00001

        MOV  WORD PTR ES:0002H, SEG DIV_0_ERROR
        MOV  WORD PTR ES:0000H, OFFSET DIV_0_ERROR

;Store address for POWER_FAIL procedure starting at address 0000:0008
        MOV  WORD PTR ES:000AH, SEG POWER_FAIL
        MOV  WORD PTR ES:0008H, OFFSET POWER_FAIL

        ; Rest of code here for program goes in here
CODE_HERE ENDS
        END
```

FIGURE 8-1. Assembler directives and instructions to initialize an interrupt pointer table for a type 0 and a type 2 interrupt.

c. The CLI (Clear Interrupt) instruction will disable the INTR input.

d. Disabling INTR ensures that the INTR input signal does not cause the 8086 to continuously interrupt itself.

e. The INTR is automatically reenabled by the IRET instruction, which restores the flags to the condition they were in before the procedure.

13. The 8086 will push the flags on the stack, clear TF and IF, push the return address on the stack, and go to the start of the divide error (type 0) service procedure. The 8086 will then respond to the NMI (type 2) interrupt. When the 8086 finishes the NMI procedure, it will return to the divide error procedure, finish executing that procedure, and then return to the mainline program.

14. The algorithm and mainline program are in A08-14A.ASM; the procedure is in A08-14B.ASM.

15. The algorithm and mainline program are in A08-15A.ASM; the procedure is in A08-15B.ASM.

16. The algorithm and program are in A08-16.ASM.

17. Break at:
    a. HERE:JMP HERE to see if the vector table is initialized correctly.
    b. the start of the interrupt procedure to see if execution gets there.
    c. after AND AL, 7FH to see the character loaded into the buffer.
    d. after POP AX to see if the return address is correct.

18. a. The base address for the added 8254 will be FF09H.
    b. The eight data lines should be connected to the upper half of the 8086 data bus (A0 = 1).
    c. Counter 0—address FF09H; counter 1—address FF0BH; counter 2—address FF0DH; control word—address FF0FH.
    d. The control word is 01 11 011 1.
    e.

```
MOV AL, 77H       ;Control word 01110111
MOV DX, 0FF0FH    ;Point at control register
OUT DX, AL        ;Send control word
MOV DX, 0FF0B     ;Point at counter 1 register
MOV AL, 56H       ;Load LSB
OUT DX, AL        ;Send LSB to counter 1
MOV AL, 03H       ;Load MSB
OUT DX, AL        ;Send MSB to counter 1
```

    f. Assuming the GATE input is high, the count is loaded on the next falling edge of the CLK input after $\overline{WR}$ goes high. The counter will start counting down in MODE 3 on the next clock pulse after that.

    g. The waveform will be a symmetrical square wave (the number is even) with a frequency of 2 kHz (712000/356 Hz), a period of 0.5 ms (1/2000 secs), and a 50 percent duty cycle.

    h. This control word will set counter 2, read/write LSB only, MODE 0 (interrupt on terminal count), with a binary count.

19. The calculation to find the value to be written to the counter is:

```
Clock signal        = 2.456 MHz
                    = 2,456,000 Hz
Frequency Required  = 1/0.0012 Hz
                    = 833.333 Hz
Count Required      = 2,456,000/833.33 = 2947
```

MODE 1 (one shot), write LSB and MSB, count in BCD.

```
Control word        = 10 11 001 1
MOV AL, 0B3H      ;Load control word
MOV DX, 0FF07H    ;Point at control register
OUT DX, AL        ;Send control word
MOV DX, 0FF05H    ;Point at counter 2 register
MOV AL, 47H       ;Load LSB of count
OUT DX, AL        ;and send to counter 2
MOV AL, 29H       ;Load MSB of count
OUT DX, AL        ;and send to counter 2
```

20. Assuming the 8254 was programmed for read/write the LSB and MSB when initialized:

```
MOV AL, 01000000B ;Counter 1 latch command
MOV DX, 0FF07H    ;Point at control register
OUT DX, AL        ;Send latch command
MOV DX, 0FF03H    ;Point at counter 1 address
IN AL, DX         ;Read LSB of latched count
MOV BL, AL        ;and save in BL
IN AL, DX         ;Read MSB of latched count
MOV BH, AL        ;Count now in BX
```

21. When an 8259A receives an interrupt signal on its IR2 input, (assuming IR2 is unmasked in the 8259A and the 8086 INTR input has been enabled with an STI instruction), the priority resolver checks the bits in the IRR (interrupt request register) to see if the interrupt is unmasked. As IR2 is unmasked, the priority resolver checks the ISR (in the service register) to see if a higher-priority input is being serviced. IR2 has the highest priority (all other interrupts are masked), so the priority resolver sets bit 2 in the ISR and activates the circuitry, which sends an INT signal to interrupt the 8086. When the 8086 sends out an $\overline{INTA}$ pulse in response to this interrupt, the 8259A responds with the type number for the IR2 service procedure. The 8086 uses this type number to find and execute the IR2 service procedure.

22. The cascade pins (CAS0, CAS1, CAS2) from the master 8259A are connected to the corresponding pins of an 8259A slave. For the master these pins function as outputs, and for the slave they function as inputs. When the master receives the first $\overline{INTA}$ pulse from the 8086, it outputs a 3-bit slave identification number on the CAS lines to enable the slave that has been given that ID number with an ICW3 word.

23. Assuming a fixed priority for the IR inputs, the 8259A will service an interrupt on its IR3 input before it services an interrupt on its IR5 input even if they are received at the same time. If two interrupt signals occur at the same time, the 8259A will service the one with the higher priority first, assuming both inputs are unmasked. If the 8259A is servicing an IR5 interrupt and an IR3 interrupt occurs, the priority resolver will check to see if the IR3 is unmasked. It then checks to see if a higher-priority interrupt is being serviced. As the new interrupt has higher priority, the priority resolver sets the appropriate bit in the ISR and activates the circuitry, which sends a new INT signal to the 8086. If the 8086 INTR input was reenabled with an STI instruction at the start of the IR5 service routine, the IR3 will interrupt the IR5 service subroutine. When the IR3 subroutine is over, control is returned to the IR5 routine.

24. An EOI command resets the in-service register so that the higher-priority interrupts may be serviced.

25. See Figure 8-2 for the sequence of command words and instructions.

26. The advantage is that you don't need to worry about the absolute address where the procedure actually resides or about trying to link the procedure into your program. All you have to know is the interrupt type for the procedure and how to pass parameters to the procedure.

```
Base address FF10H
edge triggered                          only one 8259A
an 8086 system                          normal EOI
non-buffered mode                       IR1 and IR3 unmasked
interrupt type 40 corresponds           to IR0 input
not special fully nested mode

ICW1      000   1 0 0 1 1
          |||   | | | | |_____ ICW4 needed, 8088 system
          |||   | | | |_______ not cascaded, single 8259A
          |||   | | |_________ don't care, so set at zero
          |||   | |___________ edge triggered
          |||   |_____________ this bit always 1
          |||_________________ not applicable to 8088

ICW2      0010 1000                     40 = lowest type interrupt
                                        for a IR0 interrupt type

ICW3                                    NONE, no slaves

ICW4      000   0 0 0 0 1
          |||   | | | | |_____ 8086 mode
          |||   | | | |_______ normal EOI
          |||   | | |_________ non buffered
          |||   | |___________ not special fully nested mode
          |||   |_____________ no choice, 0's
          |||_________________

OCW1      1111 0101                     unmask IR1 and IR3 with 0's

The base address is FF10H, so the two system addresses are
FF10H and FF12H.

; instructions for initializing 8259A for conditions above
     MOV AL, 00010011B          ; ICW1 command word
     MOV DX, 0FF10H             ; Point at 8259 control reg
     OUT DX, AL                 ; send ICW1
     MOV AL, 00101000B          ; Type 40 is IR0 type
     MOV DX, 0FF12H             ; point at ICW2 address
     OUT DX, AL                 ; send ICW2
     MOV AL, 00000001B          ; ICW4 command word for 8086
     OUT DX, AL                 ; send ICW4
     MOV AL, 11110101B          ; OCW1 to unmask IR1 and IR3
     OUT DX, AL                 ; send OCW1
```

FIGURE 8-2. Command words and instructions to initialize an 8259A.

CHAPTER 9 DIGITAL INTERFACING

1. Data is sent on a handshake basis to prevent the sending system from sending data bytes faster than the printer can read them.

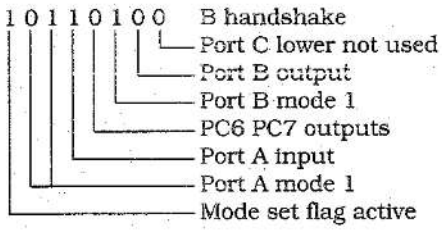
2. *a.* The sending device asserts its STB signal; the receiving device asserts its ACK signal.

   b. The sending device asserts its STB signal low to determine if the receiving device is ready. The receiving device indicates it is ready by raising its ACK signal. The sending device then sends a byte of data and raises its STB high to indicate that valid data is on its way. After the receiving device has read in the data, it drops its ACK line low to tell the sending system that it has the data and is ready for more.

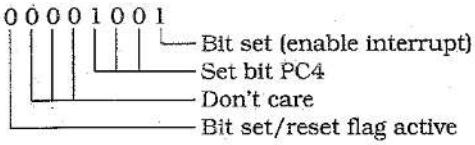
3. Port lines are inputs on reset to prevent ports from outputting into the outputs of a device connected to the port and possibly destroying one or both outputs.

4. Port A—FFF9H; port B—FFFBH; port C—FFFDH; control—FFFFH

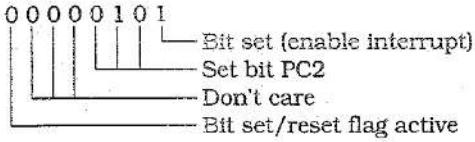
5. *a.* Mode-set control word: Port A—handshake input; port B—handshake output; port C—PC6 and PC7 outputs



b. Input control signal definition: INTR A controlled by bit set/reset of PC4



Output control signal definition: INTR B controlled by bit set/reset of PC2



c.

```
MOV DX, 0FFFFH      ;Point to 8255A control
                     register
MOV AL, 10110100B   ;Ports control mode-set word
OUT DX, AL
MOV AL, 00001001B   ;Set/reset word for INTR A
OUT DX, AL
MOV AL, 00000101B   ;Set/reset word for INTR B
OUT DX, AL
```

d. i. Set up an interrupt jump table for the interrupt procedure.
   ii. Initialize the 8259A, edge triggered, ADI = X, single.
   iii. ICW4 = 8086 mode.
   iv. Unmask the IR3 input.
   v. Enable 8086 INTR input with STI instruction.

e.

```
MOV DX, 0FFFFH      ;Point DX at control register
MOV AL, 00001101B   ;Set/reset word (set PC6)
OUT DX, AL          ;Send set/reset word
```

6. The tape reader sends a byte of data to port A on its 8 data line. It then asserts its $\overline{\text{STB}}$ signal low to tell the 8255A that a new byte of data has been sent. In response, the 8255A raises its IBF signal on PC5 high to tell the tape reader it is ready for the data. When the tape reader sees the IBF is high, it raises its $\overline{\text{STB}}$ signal high again. The rising edge of this signal causes the data to be latched on the input latches of the 8255A and, if the interrupt signal output has been enabled, causes the 8255A to output an interrupt request signal to the 8086 on bit PC3. The tape reader can then remove the data byte in preparation for the next data byte. When the 8086 reads a byte of data from the 8255, the falling edge of $\overline{\text{RD}}$ causes the 8255 to reset its INTR output. The rising edge of $\overline{\text{RD}}$ causes the 8255 to reset the IBF signal, telling the tape reader to send the next byte.

7. When connecting peripheral devices to a computer, connecting the logic and chassis grounds together only at the computer prevents large noise currents from flowing in the logic ground wire.

8. *a.* $\overline{\text{STROBE}}$ computer -> printer; character available.

   b. $\overline{\text{ACKNLG}}$ printer -> computer; data accepted, ready for the next character.

   c. BUSY printer -> computer; not ready to receive character (e.g., out of paper).

   d. $\overline{\text{INIT}}$ computer -> printer; perform printer internal initialization sequence.

9. For the algorithm, see Figure 9-15*a* on page 256

of the text. Programs A09-09A.ASM and A09-09B.ASM show the printer driver mainline and printer driver procedure.

10. Yes. There has to be a 0.5 μs wait before the $\overline{\text{STROBE}}$ is asserted low. The instructions to assert the $\overline{\text{STROBE}}$ low take 16 clock cycles. (MOV DX, 0FFFH = 4, MOV AL 000000110 = 4, and OUT DX, AL = 8 clock cycles)
The time for 1 clock cycle with an 8-MHz 8086 = 0.125 μs.
The time for 16 clock cycles = 2 μs, which is greater than the minimum time required (0.5 μs) before asserting the $\overline{\text{STROBE}}$ low.

11.

```
MOV   DX, 0FFFDH      ;Point at port C
IN AL, DX             ;Read status word from C
AND   AL, 00111111B   ;Mask o/p lines of status
                       word
```

12. The three major tasks are detect, debounce, and encode.

13. The compare method works as follows. The keypress codes are placed in a table in the order of the hex codes they represent. Once a code is obtained for the keypress row and column, that code is compared with each value in the table until a match is found. A counter is used to keep track of how far down the table you have to go to find a match for a particular input code. When a match is found, the counter will contain the hex code for the key pressed.

14. Error trapping is necessary in real code-conversion programs so that, if no match is found when all code comparisons have been made, the program gives an error message rather than continuing to compare after it has reached the end of the code table. In the program in Figure 9-20 in the text, if BX is decremented below zero, the sign flag will be set (BX = FFFFH). The program will fall through the JNS instruction and the error code loaded in AH (AH = 01—invalid code, AH = 00—valid code). AX is returned to the calling program, which checks to see if the code in AL is valid by checking the contents of AH.

15. See program A09-15.ASM. XLAT is more efficient because it requires the execution of only two loads and one XLAT instruction. The compare method would have to scan the table until it found a matching entry.

16. *a.* The answer is 77Ω. The calculation is as follows:
I = 40 mA per segment
V_d = voltage drop across LED when lit = 1.5 V
V_l = output low voltage for 7447 is max of 0.4 V @ 40 mA
$V_R = V_{cc} - V_l - V_d = 5.0 - 1.5 - 0.4 = 3.1$ V
R = required resistance = V_R/I = 77.5Ω (approx. 77Ω)

b. The answer is approximately 100 mA. The calculation is as follows:
$V_d = 1.5$ $V_{pnp} = 0.8$ $V_{npn} = 0.4$
V_R limiting resistor, R = 22Ω
$V_R = V_{CC} - V_d - V_{pnp} - V_{npn}$
$= 5 - 1.5 - 0.8 - 0.4 = 2.3$ V
$I = V_R/R$
$= 2.3$ V/22Ω = 100 mA approx.

17. *a.* See program A09-17A.ASM.

b. See programs A09-17A.ASM and A09-17B.ASM. The procedure is called every 2 ms by an interrupt signal to IR4 of an 8259A.

18. See programs A09-18A.ASM and A09-18B.ASM.

19. *a.* Look at the output line from the ULN2003A (A34) to see if it changes. Also, try another LED; that segment could be burnt out.

b. A low may never be sent to the digit-driver transistor on line 07 from the 7445. Put a scope on the digit-8 line out of the 7445 (A32) to see if it ever goes low. If a signal is present on 07, check to see if the collector of the PNP driver transistor goes to approximately +4.2 V, indicating that the transistor is on.

c. All the digits must be lit at the same time. Check the outputs of the 7445 to see if any are 0's. If they are all 0's, all digits are being driven. An alternative possibility is that the blanking code of FFH is not being sent. This would produce a ghosting effect on the LEDs.

20. See Figure 9-1 on the next page.

21. See program A09-21.ASM.

22. LCD displays deteriorate rapidly if the backplane and segment-line signals are not pulsed.

23. The circuit in Figure 9-36 on page 279 of the text can be attached to an 8255A port B pin to drive a 1-A solenoid valve from a +12 V supply if a 10-k(Ω) resistor is connected from the input base to +12 V and the transistor is mounted on a heatsink.

24. This placement of reverse-biased diodes is necessary to stop the induced voltage (inductive kick) caused by the collapsing magnetic field, which will usually be large enough to break down the transistor.

25. *a.* MOSFET = metal oxide semiconductor field effect transistor; IGBT = isolated gate bipolar transistor.

b. These devices have high-input impedances, so they require only an input voltage to turn them on. Darlington bipolar transistors require an input current.

26. *a.* Disadvantages of mechanical relays include arcing between contacts and switching on/off at any point in the ac cycle.

b. Solid-state relays produce less EMI, have no mechanical contacts to arc, and are easily driven from microprocessor ports.

27. *a.* Electrical isolation is achieved optically.
b. The zero-crossing detector turns on the current at the zero voltage point of the ac cycle.
c. A snubber circuit prevents dV/dT turning on the triac.

28. See program A09-28.ASM.

29. *a.* When reading the shaft position, gray code reduces the size of the largest possible error to the value of the least significant bit.
b. The angular resolution is $360/2^6 = 360/64 = 5.625°$.

30. *a.* The waveforms represent clockwise rotation.
b. See program A09-30B.ASM.
c. See program A09-30C.ASM.

a.

```
Keyboard/display mode set command word  =  000 DD KKK
16-character display, left entry          DD  =  01
Encoded scan keyboard, N-key rollover     KKK =  010

Program clock command word      =  001 PPPPP
1 MHz clock divided by 10   PPPPP =  01010
Clear Control Word              =  110 CCC 011
Blanking character of 11      CCC =  000
Data address                    =  80H
Control address                 =  82H

MOV AL, 00001010B     ;Load Keyboard/Display command word
OUT 82H, AL           ; and send to 8279 control
MOV AL, 00101010B     ;Load program clock command word
OUT 82H, AL           ; and send it to 8279 control
MOV AL, 11000000B     ;Load blanking command word
OUT 82H, AL           ; and send it
```

b.

```
;Write 99H to first location in display RAM
;and autoincrement display RAM pointer.
MOV AL, 1001000B      ;Load write display RAM control
OUT 82H, AL           ;word and send to 8279 control
MOV AL, 99H           ;Load 7-segment code for 99 and
OUT 80H, AL           ;send to display RAM data address
```

c.

```
;Read first byte from 8279 FIFO RAM.
MOV AL, 01000000B     ;Load read FIFO RAM control word
OUT 82H, AL           ;and send to 8279 control
IN AL, 80H            ;Read FIFO RAM
```

d.

```
Segments to be lit: H = gfecb, E = gfeda, L = fed, P = gfeba
Code on data                .gfe dcba
lines for          H  =  0111  0110  =  76H
                   E  =  0111  1001  =  79H
                   L  =  0011  1000  =  38H
                   P  =  0111  0011  =  73H
```

e.

```
;Clear control word = 110
;CdCdCd = blanking code = 100
;Cf  = Clear FIFO = 1
;Ca  = Clear all = 1
MOV AL, 11010011B     ; Clear mode word
OUT 82H, AL           ; Send to 8279 control address
```

FIGURE 9-1. Answer to question 20.

CHAPTER 10 ANALOG INTERFACING AND INDUSTRIAL CONTROL

1. *a.* The comparator will output (+15 − 1)V = 14 V.
   b. Closed-loop voltage gain = A_{VCL} = (R1 + R2)/R1 = 20.
   $V_{OUT} = V_{IN} \times A_{VCL}$ = 0.03 V × 20 = 0.6 V.
   V_{IN} = Voltage on inverting input = 0.03 V.
   If R2 = 0Ω, then A_{VCL} = (10k + 0)/10k = 1.
   c. $A_{VCL} = R_f/R1$ = 75k/15k = 5.
   $V_{OUT} = -V_{IN} \times A_{VCL}$ = −0.73 × 5 = −3.65 V.
   Zero volts is always measured on the inverting input of this amplifier.
   d. V_{OUT} = (V2 − V1)R_f/R1 = (5.1 − 4.9)1M/100k = 2 V.
   e. The instrumentation amplifier provides better rejection of the common-mode signal.
   f. The closed-loop bandwidth is f_c = 1 MHz/20 = 50 kHz. In part 2 of question 10-1*b*, f_c = 1 MHz/1.

2. The circuit is shown in Figure 10-1.

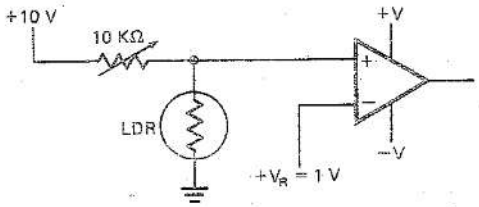


FIGURE 10-1. Circuit for question 2.

3. Voltage on inverting input = −2 V. An FET input amplifier should be used because the photodiode leakage current is very small and the FET input amplifier does not require an input-bias current. The electrons will flow from the output of the amplifier through the resistor and then through the photodiode to ground. This current is the leakage current for the diode.

4. A temperature-dependent current device would be used for applications where long wires are necessary for connecting the device to the rest of the circuit. A temperature-dependent voltage device would not suit these applications because of the voltage drop along the wires.

5. A second junction provides a reference, and that reference is held at a constant temperature. The nonlinearity can be corrected with analog circuitry which changes the gain of an amplifier according to the value of the signal or by using a lookup table in ROM.

6. Strain gages are usually connected in a bridge configuration so that temperature-caused changes in the unstressed gage compensate for temperature-caused changes in the stressed gage. You use a differential amplifier because you want to amplify the difference in voltage between the reference strain gage and the measuring strain gage.

7. The full-scale output voltage is:

$$\begin{aligned} V_{OUT} &= V_{REF}(1/R1 + 1/R2 + 1/R3 + 1/R4) \times R_F \\ &= 5\text{ V}(1/100\text{k} + 1/50\text{k} + 1/25\text{k} + 1/12.5\text{k}) \times 10\text{k} \\ &= 5(0.01 + 0.02 + 0.04 + 0.08) \times 10\text{ V} \\ &= 5 \times 0.15 \times 10\text{ V} \\ &= 7.5\text{ V} \end{aligned}$$

8. The resolution is 1 part in 2^{13} = 0.012 percent. The size of each step is 10.000/8192 = 1.22 mV. The actual maximum output voltage = (10.000 − 1.22 mV) = 9.9988 V. The converter should have an accuracy of (100 × 0.5 × 0.00122/10) = 0.01 percent.

9. Latches on D/A inputs prevent incorrect output values caused by the time between the two writes required to get all of the information to the D/A.

10. See the section *Parallel Comparator A/D Converter* on page 304 of the text. The main advantage is speed of conversion. The main disadvantage is that this type of converter has many parts and so is relatively expensive.

11. The number of counts displayed is directly proportional to the input voltage. An input of 1 V = 1000 counts; therefore, an input of 2.372 V = 2372 counts. The resolution of a 4.5-digit, slope-type A/D converter is 1 part in 19,999 or between 14 and 15 bits.

12. The conversion requires 12 clock cycles. Note: Some converters use one additional clock pulse for internal housekeeping.

    The number of clock cycles required for a 12-bit, dual-slope type has different values in different systems, but is fixed for a given system.
    Assume 5.12 V full scale:
    Clock cycles = full-scale count × $V_{IN}/V_{FULLSCALE}$
    For 0.1 V, clock cycles = 4096 × 0.1 /5.12 V
    = 80 counts
    For 5 V, clock cycles = 4096 × 5.0 /5.12 V
    = 4000 counts

13. *a.* See program A10-13A.ASM.
    b. See program A10-13B.ASM. The successive-approximation algorithm is much faster, because it only takes 8 cycles through the conversion loop to do an 8-bit conversion.

14. The algorithm is as follows:
Poll digit 1 (MSD) strobe until high
Read in MSD
Rotate BCD nibble to high nibble
Mask low nibble
Move to BH
Poll digit 2 strobe until high
Read in digit 2
Mask upper nibble
OR with digit 1 in BH
Poll digit 3 strobe until high
Read in digit 3
Rotate BCD nibble to upper nibble
Mask lower nibble
Move to BL
Poll digit 4 strobe until high
Read in digit 4
Mask upper nibble
OR with digit 3 nibble in BL
RESULT in BX

15. 01011011 will represent −1.44 V.
Output code = 64 + 16 + 8 + 3 = 91
0.0391 V = 1 least significant bit, so V = − 5.000V + (91× 0.0391) = −1.44 V
You could convert to 2's complement representation by inverting the MSB of the output code.

16. See the program A10-16.ASM.

17. The voltage measured on the inverting input of the LM308 should be 6.9 V. With no load, the two input voltages should be +5 V; the output voltage should be 0 V.

18. See program A10-18.ASM.

19. See the section *Overview of Industrial Process Control* on page 317 of the text.

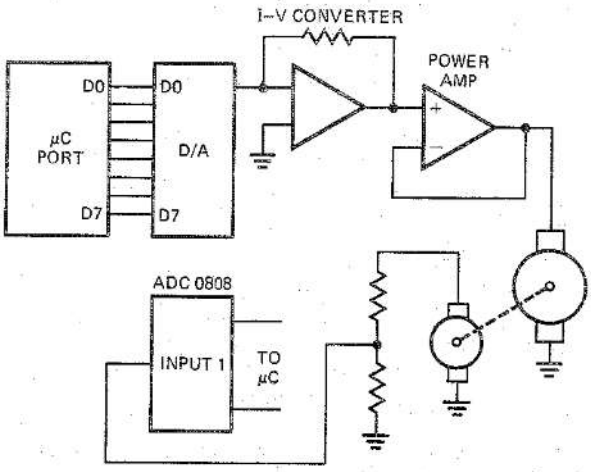
20. Integral feedback helps eliminate residual error, but the effect of integral feed is slow. Derivative feedback is added to give a quick response to an error signal.

21. The major advantage of a microcomputer-controlled loop is versatility. Process variables, the rate at which control loops are serviced, and the feedback algorithms can be adjusted by simply changing the program.

22. *a.* See Figure 10-2*a*.

b. See Figure 10-2*b*.

c. A lookup table could be used to determine the feedback value by subtracting the actual speed from the setpoint and using the difference as a pointer into a table to determine the new value to be sent to the D/A.



(a)

```
REPEAT
  Do A/D conversion for Tachometer input of A/D
  Compare tach output value to setpoint
  IF lower THEN
       increment control byte
       output to D/A
  IF higher THEN
       decrement control byte
       output to D/A
  Wait for response
UNTIL Tachometer output = setpoint
```

(b)

FIGURE 10-2. Question 22 (a) Circuit. (b) Algorithm.

23. *a, b.* See the section *Time-domain and Frequency-domain View of a Square Wave* on page 337 of the text.

24. *a.* See the section *Digital Filters* on page 338 of the text.
    b. The output value of an IIR filter is saved to be used in computing feedback terms. In the FIR filter, the samples from the A/D converter are directly saved to be used in later computations.

25. *a.* See the section *Digital Filter Hardware* on page 339 of the text.
    b. Features of a digital signal processing microprocessor include sizable amounts of on-chip registers, ROM and RAM; separate buses for code words and for data words; an optimized multiplier; a 32-bit barrel shifter; a 16-bit or 32-bit CPU, an instruction set optimized for DSP applications; and, sometimes, a floating-point processor.

26. *a.* The sampling theorem says 2 samples/cycle, but the more samples per cycle, the easier it is to reconstruct the sine wave.
    b. An analog low-pass filter prevents aliasing by filtering out signals that have a frequency greater than half the sampling rate.
    c. The sample-and-hold circuit is used to keep the value on the input of the A/D constant during the conversion cycle.

27. In writing the program for a digital filter, determine the coefficients that the terms in the equation will be multiplied by to implement the desired filter. Write a program which reads in values from the A/D, computes an output value, and sends the computed value to the D/A converter at the right time.

CHAPTER 11 DMA, DRAMs, CACHE MEMORIES, COPROCESSORS, AND EDA TOOLS

1. Peripheral expansion slots let you easily customize the system based on your application and financial state. Then you won't have to pay for features you won't use.

2. In an 8086 operating in maximum mode, the control bus signals are sent out in coded form on the status line $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$. An external controller, such as the Intel 8288, decodes these lines to produce the required control bus signals.

3. DMA data transfer is faster because the data does not have to go through the microprocessor; it is transferred directly between the peripheral and memory.

4. The series of actions performed by a DMA controller is as follows:
   a. If that channel of the DMA is unmasked, the DMA controller sends a hold request (HRQ) to the microprocessor HOLD input.
   b. The microprocessor floats its buses and sends out a hold acknowledge (HLDA) signal to the DMA controller.
   c. The controller outputs on the address bus the memory address for the data to be transferred.
   d. The controller then sends a DMA acknowledge signal (DACK0) to the peripheral device to tell it to get ready to output a byte.
   e. The controller asserts $\overline{MEMW}$ to enable addressed memory for writing data to it.
   f. The controller then asserts $\overline{IOR}$ to enable the peripheral device to output data. The data is then transferred.

5. The 20-bit address is produced as follows:
   a. The DMA controller disables U1 so that address lines A7–A0 no longer come from the 8086 bus. The controller directly outputs these bits of the memory address.
   b. The controller then outputs bits A15–A8 of the memory address on its data bus pins and latches this address on the outputs of U2.
   c. Bits A19–A16 are produced either by hard wiring the inputs of U3 to give fixed values for these addresses or by specifying the bits under program control by connecting the inputs of U3 to an output port.

6. The devices U5 and U6 act as a switch which can route data to/from the disk controller from/to either odd or even addresses in memory. A0 determines which half of the data bus is connected to the eight disk controller data pins.

7. The sequence of signals is shown in Figure 11-8 on page 354 of the text.

8. The major tasks to be done when using dynamic RAM in a microprocessor system are:
   a. To refresh each RAM location at the proper time interval.
   b. To funnel the two halves of the address into each device with the appropriate $\overline{RAS}$ and $\overline{CAS}$ strobes.
   c. To ensure that a read or write operation and a refresh operation do not take place at the same time.
   d. To provide a read/write control signal to enable data into/out of the devices.

9. The dynamic RAM controller will hold $\overline{AACK}$ high if the 8086 tries to access memory. This causes the 8086 to insert a WAIT state which gives the controller enough time to finish its refresh cycle.

10. a. The time parameter is the read-cycle time, $t_{w(RH)}$.
    b. If successive data words are read from locations in the same page (row), no precharge time is required. Also, if successive data words are read from the same page, the row address is the same, so a new row address does not have to be sent out and strobed in with another $\overline{RAS}$ signal. With the proper controller, these two factors make it possible to read data from a page without WAIT states.
    c. In page mode operation, as long as the microprocessor continues to access memory locations in the same page, the controller will hold $\overline{RAS}$ low, send out the column part of the addresses to the DRAMs, and pulse $\overline{CAS}$ low for each new column address. Static column mode operation operates like page mode for the first memory access. If the next address is in the same row, the DRAM controller will hold both $\overline{RAS}$ and $\overline{CAS}$ low and send out a sequence of column addresses.

11. a. See the section *Cache Mode DRAM Systems* on page 358 of the text.

b. A cache controller uses a cache tag RAM (cache directory) to keep track of which blocks from the main memory are present in the cache.

c. Each entry in the cache tag RAM contains the upper bits or base address of a page in the main memory. In the case of the Intel 82385 cache controller, each directory entry also contains a tag valid bit and eight line valid bits for the eight lines represented by that entry.

d. The two-way set associative cache approach sets up two separate caches and two separate directories so that the same lines from two different pages can be cached at the same time.

12. In microcomputers such as the IBM PC, the DRAM memory banks are 9 bits wide. A parity generator/checker circuit determines the parity of an 8-bit data word as it is being written to a memory location. A parity bit is generated such that the overall parity of the 8 data bits plus the parity bit is always, for example, odd. When the data byte and parity bit are read from memory, the parity of the 9 bits is checked. If the parity of these bits is not as it should be, an error has been introduced somewhere.

    The major shortcoming of this method of error detection is that two errors in a data word can cancel each other. Also, you know only that an error has occurred, not which bits are in error.

13. Seven encoding bits are required to detect and correct a single-bit error in a 64-bit data word.

14. The 8088 is configured in maximum mode because the MN/$\overline{\text{MX}}$ pin is tied to ground.

15. You would look on schematic sheet 2 to see how the AEN signal is produced.

16. Coprocessors are specialized microprocessors. A standard microprocessor instruction set is general purpose. A standard microprocessor can operate alone, whereas a coprocessor needs a standard processor as a host.

17. a. Binary: 2435.5625 = 100110000011.1001

    Normalized binary: 2435.5625 = 1.00110000011 1001 E11 (0BH)

    Long-real format: 2435.5625 = 40A3 0720 0000 0000 (Exponent = 3FF + 0BH = 40AH; magnitude = 011 0000 0111 0010 + 36 zeros)

    Temporary-real format: 2435.5625 = 400A 9839 0000 0000 0000 (Exponent bits = 0BH + 3FFFH = 400AH; number = 1001 1000 0011 1001 + 48 zeros. Note: MSB = 1)

    b. Most floating-point numbers are approximations because you need an infinite number of digits to represent some of the numbers you work with.

18. a. Register 0 is TOS after the 8087 is reset.

    b. Register 7 will be ST after one data item is read.

    c. The contents of the ST(3) location are added to the contents of the ST(2) location. The result is left in the ST(2) location.

    d. The stack will be popped once after the instruction. The result will be in ST(1).

19. a. FLD TAX_RATE: Decrements pointer and pushes the contents of memory called TAX_RATE on stack at new ST(0).

    b. FMUL INFLATION_FACTOR: ST is multiplied by INFLATION_FACTOR, and the result is placed in ST.

    c. FSQRT: Replaces contents of ST with square root of ST.

    d. FLDPI: Decrements pointer and pushes the value of π on the stack.

    e. FSTSW CHECK_ANSWER: Copies 8087 status word to CHECK_ANSWER and increments pointer.

    f. FPTAN. Produces tangent ratio (Y/X) for angle in ST. Y value replaces the angle, X is pushed on the stack. Therefore, X ends up at ST(0) and Y at ST(1).

20. The assembler inserts 9BH to allow the 8087 to finish the instruction before allowing the 8086 to continue with its next instruction.

21. See program A11-21.ASM.

22. a. The standard microprocessor and the coprocessor share status, QS1–QS0, address, and data lines, so they have to be connected together. The fact that the status and queue status lines are connected allow the 8087 to track the 8086 queue.

    b. The 8087 coprocessor gets its instructions from memory as they are fetched by the 8088 and puts them in its own queue.

    c. The main processor distinguishes its instructions from those for the 8087 by the fact that the 8087 instruction codes have 11011 as the most significant bits of their first code byte.

    d. To load a long-real data item from memory to the 8087 ST, the 8086 outputs the address of the first data word on the address bus and outputs the appropriate control signals. The 8087 reads a data word from the data bus and the 20-bit physical address output by the 8086. The 8087 then takes over the bus from the 8086 to transfer the remaining data words it needs from memory directly to the 8087.

    e. The 8087 sends out a low-going pulse on its $\overline{\text{RQ}}$/$\overline{\text{GT0}}$ pin to signal the 8088 that it needs to use the buses.

    f. You can prevent the 8088 from going on with its next instruction before the 8087 has

completed an instruction by putting an FWAIT instruction before the 8087 instruction in your program. The FWAIT instruction causes the 8088 to sit in a loop until its TEST input is asserted low when the 8087 unasserts its BUSY signal.

23. *a.* See the section *Initial Design and Schematic Generation* on page 379 of the text.

b. A circuit diagram can be easily and quickly modified and resimulated. Also, a design file containing the logical and timing parameters for the devices is created.

24. *a.* Software breadboarding means using a simulator to test the operation of a circuit.

b. Simulation allows you to change the design and resimulate the circuit in a shorter time. Also, you can pick out marginal timing errors, don't have to wait for parts, and can generate complex test signals easily.

c. A simulation model for a device contains a software description of the characteristics of the device. These include a picture file with the schematic symbol, the logical characteristics (truth table), and the timing parameters.

d. See the section *A Microcomputer Simulation Example* on page 383 of the text.

e. Simulation tells you if the circuit connections and the timing of the circuit are correct.

25. See the section *Design for Test* beginning on page 384 of the text.

CHAPTER 12 C, A HIGH-LEVEL LANGUAGE FOR SYSTEM PROGRAMING

1. a. The index value of the first element is zero.
   b. During the second execution of the loop index = 1, so the second element in cost will be accessed.
   c. The #include<stdio.h> directive tells the compiler that the declaration for any predefined functions used are in the file stdio.h.
   d. The word printf, followed by parentheses, calls the predefined printf function to display the information passed to the function in parentheses on the screen.

2. a. The integrated environment saves the time required to switch among the different tools and displays error messages on the screen with the program you are working on.
   b. The IDE compiler displays error messages in a window at the bottom of the screen. It highlights one error message at a time and highlights the program line which caused that error.
   c. When you put a watch on a variable in the tc environment, the value of that variable is displayed in the watch window as you step through the program.

3. Range actually depends somewhat on the compiler and machine used, but most common values are:
   a. Char -128-> + 127
   b. Int -32,768 -> + 32,767
   c. Unsigned int 0 -> 65,535
   d. Long -2,147,483,648 -> +2,147,483,647
   e. Float 3.4E -38 -> 3.4E + 38

4. a. int total_boards;
   b. char no = 'n';
   c. float body_temp = 98.6;
   d. int scores[5];
   e. int scores[6] = {95, 89, 84, 93, 92};
   f. int *ptr = scores;
   g. char screen[25][80];
   h. char screen_buffer [4][25][80];
   i. unsigned int monitor_start = 0xFE00;
   j. char *answer;
   k. int *ptr = &setpoint;
   l. float *wptr = net_weights;

5. a. Multiply 4 by 7, divide result by 9, subtract quotient from 5.
   b. Add 4 to a, multiply sum by 17, subtract quotient of B/2, add 6 to result.
   c. Add y to x, increment y by one.
   d. Decrement y, subtract decremented value of y from x.
   e. Count = count +4. add 4 to value of count.
   f. AND value of strobe_val with 0001H.
   g. Rotate the a value 4 bit positions to the right and assign the result to y.
   h. Remainder produced by dividing 39 by 4 is assigned to b. The value of b then is 3.
   i. The value of ch is compared with the ASCII codes for Y and y. If ch is equal to either code, execution will go the statement at the label start. If ch is not equal to either code, execution will simply go on to the next program statement.

6. a. printf("%d",count);
   b. printf("Please enter your weight in Kg, and press enter.")
   c. printf("%10.4f",conversion_factor);
   d. printf("%e",average_lunar_distance);

7. See program A12-07.C on the disk.

```
\* Program to average 4 numbers and print result */
#include<stdio.h>
int index, sum = 0, av;
int nums[] = {45, 65, 38, 72};

main()

{
        for(index = 0; index < 4; index++)
        sum = sum + nums[index];
        av = sum/4;
        printf("The average of 45, 65, 38, and 72 is
        %d.\n",av);
}
```

8. See program A12-08.C on the disk.

```
/* Program to read and average 5

test scores */

int scores[6];
int *spntr = scores;
int i, sum = 0;

main()
{
printf("Enter 5 scores. After each score press space or
enter.\n
for (i=0; i < 5; i++");
  {
        scanf("%d", spntr);
        spntr++;
  }
spntr = scores; /* reset pointer to start of array */
for (i=0; i < 5; i++)
  {
```

```
        sum = sum + *spntr;
        spntr++;
 }
*spntr = sum/5;
spntr = scores; /* reset pointer to start of array */
printf("For scores of %d, %d, %d, %d, and %d the
      average" " is %d.\n",*spntr, *(spntr+1),
      *(spntr+2), *(spntr+3), *(spntr+4), *(spntr+5));
}
```

9. See program A12-09.C and alternate A12-09B.C on the disk.

10. *a.* See program A12-10A.C on the disk.
    b. See program A12-10B.C on the disk.

```
/* program section to read up to 1000 characters and
        put in array */
#include<stdio.h>
char buffer[1000];
char ch = 0x00;
int index = 0;
main()
{
printf("Enter up to 10 characters. Enter ^Z to
exit.\n"); for(index=0; index < 10; index++)
 {
        ch=getchar( );
        if(ch==EOF)
        {
        printf("Goodbye.\n");
        exit( );
        }
        buffer[index] = ch;
        ch=getchar( ); /* clear carriage return from
          buffer */
 }
        printf("Buffer full.\n");
        for(index = 0; index < 10; index++)
        printf("%c", buffer[index]);
}
```

11. See program A12-11.C on the disk.

12. See program A12-12.C on the disk.

13. See program A12-13.C on the disk.

14. Formal arguments are declared in the function definition header. They identify the variable that will receive the values passed during the function call. Actual arguments are the values passed to the function during the function call.

15. See program A12-15.C on the disk.

16. See program A12-16.C on the disk.

17. See program A12-17.C on the disk.

18. See program A12-18.C on the disk.

19. *a.* program, global
    b. program, global
    c. program, global
    d. program, this source module only
    e. program, this function only
    f. this function only, this function only
    g. this function only, this function only

20. See program A12-20.C on the disk.

21. The main advantage of using predefined C library functions is that these functions have already been written, tested, and debugged. All you have to do is call them and make sure the libraries containing the functions are available to the linker. The main disadvantage is that the predefined functions often do not perform the exact task you need done. The solution to this problem is to obtain a copy of the source listing for the desired function and generate a modified version that meets your needs.

22. See program A12-22.C on the disk.

23. Identify the function as extern in the C module, maintain the segment naming structure in the assembly language module, declare the assembly language procedure public in the assembly language module, access parameters on stack, return value in AX.

24. *a.* Tiny, small, medium, compact, large, and huge.
    b. The distinguishing features are the number of segments and the size of the pointers used for code and data references.
    c. Small model: One 64-Kbyte code segment, one 64-Kbyte data segment shared by data segment, stack segment, and extra segment.
    d.
```
DOSSEG
.MODEL SMALL
.CODE
.DATA
```

25. See the discussion on page 432 of the text.

26. See programs A12-26.C and A12-26.ASM on the disk.

CHAPTER 13 MICROCOMPUTER SYSTEM PERIPHERALS

NOTE: The end-of-chapter questions for this chapter relate mostly to factual information from the text. The experiments manual contains more complex programming problems based on the concepts and examples in the chapter.

1. *a.* Scanf and getche are not always suitable because they read codes only in the range 00H-7FH.
   b. See program A13-01B.C on the disk.

2. A noninterlaced raster is produced on a CRT as follows. An electron gun at the rear of a vacuum tube produces a beam of electrons directed toward the front of the tube. The inside surface of the front of the tube is coated with a phosphor substance which gives off light when struck by electrons. The beam is swept back and forth from the top to the bottom of the screen, as shown in Figure 13-4*b* on page 439 of the text. To produce an image, the electron beam is turned on or off as it is swept across the screen. For a more detailed explanation, see pages 439–440 of the text.

3. Refer to Figure 13-5 on page 440 of the text. As the beam sweeps across the first scan line of the character row containing the X, it is turned on to produce the dots at the upper corners of the X. On the next sweep of the beam, the beam is turned on to produce the two dots for that scan line of the X. The process continues until all the scan lines of the X have been swept out. When all of the rows of the characters have been swept through, the beam is retraced to the top of the screen to start over.

4. Refer to Figure 13-6 on page 441 of the text when reading the answers to this problem.
   a. The purpose of the RAM in this circuit is to hold the codes for the characters to be displayed.
   b. The address inputs of the RAM are changed with each move from one character to the next along a scan line, each horizontal retrace, each increment to the next row of characters, and each vertical retrace.
   c. The R0–R3 address inputs of the character generator ROM are changed when the dot-row counter changes.
   d. The shift register on the output of the character generator ROM is used to convert the parallel data from the ROM into serial form.
   e. Horizontal sync pulses are produced to cause the beam to sweep back to the left of the screen, that is, retrace at the end of a scan line.
   f. Vertical-sync pulses are produced when the beam reaches the bottom of the screen and needs to be retraced back to the top.

5. NOTE: This problem uses 3 character times for horizontal overscan, but a more reasonable figure is 20 character times.
   a. 80 + 3 overscan = 83 character times per row.
   b. (12 lines/character × 25 rows) + 120 overscan = 420 scan lines per frame.
   c. 60 frames/second × 420 lines/frame = 25,200 lines per second.
   d. 25,200 lines/second × 83 characters/line × 9 dots/character = 18,824,400 dots/second.
   e. 9.5–10 MHz minimum bandwidth.
   f. 25,200 lines/second × 83 characters/line = 2,091,600 characters/second = 1 character/478 ns.

6. *a.* (14,000,000 dots/second)/15,750 lines/second = 889 number of dot times per scan line.
   b. Active display = 80 characters × 8 dots/character = 640; 889 total dots/line − 640 dots/line active = 249 dot times left for horizontal overscan.
   c. (15,750 lines/second)/(60 frames/second) = 262 scan lines per frame + overscan.
   d. Active scan lines = 25 rows × 8 lines/row = 200; 262 lines − 200 active = 62 left for vertical overscan.

7. If the 6845 CRT controller and the microprocessor want to access the display RAM at the same time, the CRT display system in Figure 13-8 of the text arbitrates the dispute by interleaving the 6845 accesses and CPU accesses. The character clock signal going to the 6845 and the multiplexers allows the CPU to access the RAM only during one half of the clock signal and the 6845 only during the other half of the clock signal. If the CPU tries to access the display RAM during the controller's half of the character clock cycle, a not ready signal from the CRT controller will cause the processor to insert WAIT states until the half of the character clock signal when it can access the display refresh RAM.

8. See the program A13-8.ASM for the algorithm and program.

9. The memory required to store the pel data for a bit-mapped display of 640 × 480 is (640 × 480)/8 = 38,400 bytes, or about 40 Kbytes.

10. Three electron beams are used to produce all possible colors on a color CRT screen by using different intensity ratios for each beam. See Figure 13-10 on page 444 and Figure 13-11 on page 445 of the text.

11. *a.* Eight bits are required to specify one of 256 colors for a pixel.
    b. 1024 × 768 = 786,432 pixels. 786,432 pixels × 4 bits/pixel = 3,145,728 bits = 393,216 bytes, or about 400 Kbytes.
    c. See Figure 13-12 on page 446 of the text. In packed pixel storage, all data for the pixel is in one nibble of a memory location. In planar pixel storage, data for the pixel comes from four bytes in four different areas of memory (planes).

12. See Figure 13-17 on page 449 of the text. Four data bits for a pixel select one of 16 locations in the palette register. The palette register contains a 6-bit code for 6 color outputs. Only 16 of 64 possible colors are in the palette register at one time, so only 16 colors are produced.

13. *a.* Eighteen bits are required for one of 256K colors.
    b. There is too much memory, and the memory access rate is too high.
    c. See Figure 13-18 on page 450 of the text. Eight-bit pixel data selects one of 256 color registers. Color registers contain 18-bit codes for 256 out of 256K possible colors.
    d. A 6-bit D/A converter is used to convert each group of 6 bits to an equivalent analog value which is used to drive a color gun.

14. See program A13-14.ASM on the disk.

15. *a.* Initialize the graphics adapter board for 640 × 480 × 16 color mode.
    b. Call the windowsize function to determine the number of bytes required to save pixel data for desired window.
    c. Save a copy of the pixel data for the window area in a memory buffer named window_buffer, which was created with malloc.
    d. Restore the original display for the area covered by the window.
    e. Release the memory allocated to window_buffer by malloc so that it can be reallocated to some other use.

16. *a.* Complex graphics images require so much computation that if the main processor were used, it would not have time to do anything else.
    b. Manipulation of complex graphics images requires many floating-point computations, so a math coprocessor is included to handle these computations efficiently.

17. The major advantages of LCD displays are that they require very low power, are lightweight, and are flat. Disadvantages are higher cost and the need for back lighting.

18. See programs A13-18A.ASM and A13-18B.ASM on the disk.

19. To make an image sensor, several hundred CCD shift registers are built in parallel on the same chip. A photodiode is doped in under every gate. When all of the gates with photodiodes under them are made positive, potential wells are created. A camera is used to focus an image on the surface of the chip, and light shining on the photodiodes causes a charge proportional to the light intensity to be put in each well that has a diode. These charges can be shifted out to produce the dot-by-dot values for the scan lines of a picture. The video information must then be passed through an A/D converter to convert it to digital information so that it can be stored in memory.

20. *a.* The read/write head for a disk drive is moved into position over a specified track using a stepping motor. See also Figure 13-32 on page 466 of the text.
    b. The rigid medium allows tracks to be closer together so that more data can be stored. It also allows the disk to be rotated faster, permitting the data to be read off the disk faster.
    c. The actual rate at which data is coming off the disk varies with the position of the head on the disk and changes in disk speed. A phase-locked loop circuit is required to produce a clocking signal which is synchronized with the data being read.
    d. RLL 2,7 encoding allows more data to be stored in a given track than MFM does.

21. *a.* The main improvement of an ESDI interface is a greater data rate and the ability to interface with larger hard disks.
    b. See Figure 13-39*b* on page 472 of the text for a separate I/O bus which can be used for disks and other peripherals.

22. *a.* Use to check whether there are any errors in the data read from the block. If an error is found, another attempt to read the block can be made.
    b. Put a disk in, for example, the A: drive, type FORMAT A:, and press the Enter key.
    c. i. Use DEBUG to do the low-level format.
    ii. Use the DOS FDISK command to partition the disk into logical drives such as C:, D:, E:, etc.
    iii. Use the DOS FORMAT command to format each partition.

23. a. The file allocation table keeps track of which clusters are allocated to which file and which clusters are available.
    b. The first FAT entry for a file contains the number of the cluster which contains the next section of the file; the FAT entry for the second section of the file contains the number of the cluster which holds the next section of the file; etc.
    c. Types of information include Filename, extension, file attribute (read, hidden, system, volume), time, date, starting cluster, and size in bytes.

24. See program A13-24.ASM on the disk.

25. See program A13-25.ASM on the disk.

26. a. FILE *fp; declares fp as a special type of pointer called a file pointer, which points to a data structure which duplicates the file control block.
    b. fopen (filename,"wt") opens the specified file for write operations. If the function call returns a 0 to fp, an error occurred during the attempt to open the file. In this case, the predefined function perror is called to write an error message to the screen.
    c. Marks the file represented by file pointer fp closed and closes the stream associated with fp.
    d. While not end of file reads a character from the file stream with the fgetc() function and sends the returned character to the screen (sdtout) with the fputc() function.

27. a. A RAM disk program configures a section of RAM as a virtual disk drive so that data can be read from it and written to it using the same procedures used for magnetic disks. A RAM disk is much faster than a magnetic disk drive because it does not have the mechanical access time.
    b. A disk cache functions similarly to the way an SRAM cache does for a DRAM system. When a program requests, for example, the first few blocks of a file, the cache program reads a large part of the file into the cache. Then, when the program needs the next part of the file, it will already be present in the cache and a new read from the hard disk will not be needed. A RAM disk is referred to as D:, etc. A disk cache is transparent to the user.

28. a. With an optical disk, data is read by bouncing a tiny laser beam off the track as the disk rotates. One advantage of this method is that there is no chance of the head "crashing" on the disk and destroying it.
    b. Advantages of optical disk storage include a greater amount of storage capacity per system, removability, and imperviousness to magnetic fields and humidity.
    c. Data bits are recorded in magneto-optic read/write optical disk systems using a disk coated with an exotic metal alloy which has the required magnetic properties. The read/write head has a laser diode and a coil of wire. A current is passed through the coil to produce a magnetic field perpendicular to the disk. To record a 1 at a spot in a data track, a pulse of light from the laser diode is used to heat up that spot. Heating the spot makes it possible for the applied magnetic field to flip the magnetic domains around at that spot and create a tiny vertical magnet. When heated with no field present, the magnetism of the spot will flip around in line with the horizontal field on the disk.

29. A human brain can store about 10^{10} bits of data and has an access time of about 1 second. Optical disksystems such as the Maxtor Tahiti store 10^9 bits with an access time of about 30 ms and transfer data at a maximum rate of about 10 Mbits/second.

30. For a description of the print mechanisms and their advantages/disadvantages, see pages 479-481 of the text.

31. Figure 13-45 on page 481 of the text shows a block diagram of a waveform-modification type of speech synthesizer. LPC synthesizers use a digital filter to modify the signals from a pulse and a white-noise source. Formant synthesizers use several resonant filters to massage the signals from a variable-frequency signal source and a white-noise source. In phoneme synthesizers, a 6-bit code controls the characteristics of some formant filters. The major difference between an LPC and a formant synthesizer is the type of filter used to modify the signal.

32. In a direct digitization speech synthesizer, an A/D converter is used to take samples of actual speech signals from a microphone. These samples are stored in memory. To reproduce speech, the samples from memory are applied to the inputs of a D/A converter. The advantage of this method is very accurate reproduction. The disadvantage is the large amount of data that must be stored to record even a small amount of speech.

33. a. The main storage technique is data compression. For both audio and video information, only the changes from previous values are stored, so much less data storage is required.
    b. The system uses signals from controls you operate to choose "out the window" view and instrument displays. The system can be programmed to insert emergency conditions that you must respond to.

CHAPTER 14 DATA COMMUNICATION AND NETWORKS

1. The bit format used for asynchronous serial data is shown in Figure 14-1 on page 488 of the text.

2. *a.* If a terminal is transmitting asynchronous serial data at 1200 Bd, the bit time is 0.833 ms (1/1200).
   b. Assuming 7 data bits, 1 parity bit, 1 start bit, and 1 stop bit, it takes (0.833 × (7 + 1 + 1 + 1) or 8.33 ms to transmit one character.

3. The main difference between a UART and a USART is that a UART can only do asynchronous communication, while a USART can be programmed to do either asynchronous or synchronous communication.

4. A modem, which stands for modulator-demodulator, is a device used to convert digital signals to audio frequency tones (in the frequency range which phone lines can transmit) and to convert transmitted tones back to digital information. A modem is needed to send digital data over standard switched phone lines because these phone lines have a bandwidth of only about 300–3000 Hz. This is not enough bandwidth to transmit digital signals directly.

5. After the terminal power is turned on and the terminal runs any self-checks, it asserts the $\overline{\text{DTR}}$ signal to tell the modem it is ready. When it is powered up and ready to transmit or receive data, the modem asserts the $\overline{\text{DTR}}$ signal to the terminal. When the terminal has a character to send, it asserts an $\overline{\text{RTS}}$ signal to the modem. The modem then asserts its CD to the terminal to indicate that it has made contact with the computer. When the modem is ready to transmit data, it asserts the $\overline{\text{CTS}}$ signal back to the terminal. The terminal then sends serial data characters to the modem. When the terminal has sent all the characters it needs to, it makes $\overline{\text{RTS}}$ high, which causes the modem to unassert its $\overline{\text{CTS}}$ signal and stop transmitting. See also Figure 14-2 on page 489 of the text.

6. In order for an 8251A to transmit data at 4800 Bd with a baud rate factor of 16, the transmit clock frequency must be (4800 × 16), or 76,800 Hz.

7. *a.* The bit patterns for the mode and command words for the 8251A are shown in Figure 14-1.
   b. The sequence of instructions required to initialize an 8251A at addresses 80H and 81H are shown in Figure 14-5 on page 492 of the text.
   c. See Figure 14-6*b* on page 493 of the text for the sequence of instructions that can be used to poll this 8251A to determine when the receiver buffer has a character to be read.
   d. You can determine whether a character received by an 8251A contains a parity error by checking the D3 bit in the status word. A 1 in this bit indicates a parity error.
   e. The frequency transmit and receive clock required to send data at 2400 Bd using this 8251A with a baud rate factor of 64 is 153,600 Hz.
   f. Characters can be sent to and read from the 8251A on an interrupt basis. To send characters on an interrupt basis, the TxRDY pin of the 8251A is connected to an interrupt input on the processor or an IR input on an 8259A priority interrupt controller.

8. RS-232C logic high-voltage range = −3 to −15 V under load
   RS-232C logic low-voltage range = 3 to 15 V under load

9. *a.* When you attempt to connect together two RS-232C devices that are both configured as DTE, the TxD pin on one device will drive data into the TxD pin on the other device. Likewise, the RxD pins and the handshake signal pins will be mismatched.
   b. The problem can be solved by making an adapter called a null modem as shown in Figure 14-10 on page 496 of the text.

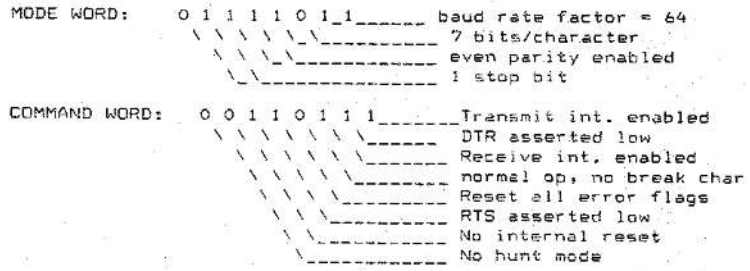


FIGURE 14-1. Bit format for the mode and command words for question 14-7a.

10. *a.* The two ground pins on an RS-232C connector are connected together only at the power supply in the terminal or the computer to prevent large ac-induced ground currents on the signal ground.
    b. If the wire connected to pin 5 of an RS-232C terminal is broken, the clear-to-send signal will not be received. Most terminals will send no data unless they receive the $\overline{\text{CTS}}$ signal.

11. The higher transmission rate of the RS-422A is possible because the differential lines are terminated by resistors, so they act as proper transmission lines instead of simply as open wires. The RS-423 also uses a transmission line, so it too can transmit at higher rates.

12. *a.* An FSK modem uses one tone to represent a 0 and another tone to represent a 1 in the signal it sends out on a phone line.
    b. Full-duplex communication over standard phone lines can be achieved by using one pair of tones for communication in one direction and a different pair of tones for communication in the other direction.
    c. The maximum FSK data rate is about 1200 Baud.

13. *a.* Figure 14-2 shows the waveform of a signal that a simple PSK modem will send out to represent the binary data 011010100.

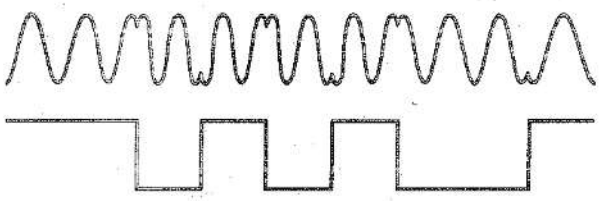


FIGURE 14-2. Waveform of a signal that a simple PSK modem will send out for the binary data 011010100. (Answer to question 14-13a.)

    b. Use four different phase shifts to represent the four possible combinations of two data bits as shown in Figure 14-15*a* on page 499 of the text.
    c. Eight different phase angles and two amplitude values are used to represent 16 possible combinations of a group of 4 data bits. See Figure 14-16 on text page 500.

14. *a.* Digital signals have much better noise rejection than analog signals and can more easily be regenerated as needed along the line.
    b. Codec stands for coder-decoder. The coder is an A/D converter that converts the analog signal into digital form. The decoder is a D/A converter that converts the digital signal back to analog.
    c. Codecs are designed with a nonlinear response (small steps for small signals, large steps for large signals) to reduce the dynamic range of the signals and provide greater accuracy for small signals with a minimum number of converter bits.
    d. Telephone companies use time division multiplexing or frequency division multiplexing to transmit many signals on the same wire, fiber, or radio channel.

15. *a.* The ISDN converts analog voice signals to digital form at the user's site instead of in the local branch office. A basic service supplies two 64 Kbit/s data channels and a 16 Kbits/second control channel.
    b. The significance of the ISDN for computer communication is that the ISDN allows computers to communicate directly in digital form instead of using modems with audio tones. This provides higher speed and better error rates.

16. *a.* See Figure 14-22*e* on page 506 of the text.
    b. LEDs and ILDs (infrared injection laser diodes) are two types of devices used to produce the light beam for a fiber optic cable. Darlington photodetectors (i.e., the MF0D73) or avalanche photodiodes (APDs) are used to detect the light at the receiving end of the fiber.
    c. You should never look into the end of a fiber optic cable because the light beam may be strong enough to permanently damage your eye.
    d. A fiber with a diameter large enough to allow beams with several different entry angles to propagate through it is a multimode fiber. Single-mode fibers only allow beams very nearly parallel to the axis of the fiber to be transmitted through them. The multimode fibers are easy to make but have problems at high data rates. The single-mode fibers can transmit at very high rates, but it is difficult to make low loss connections with the tiny fibers. See page 506 of the text.
    e. The major advantages of fiber optic cables over metallic conductors are that they are smaller, are immune to electrical noise, send longer distances without repeaters, and can accommodate a large number of fibers.

17. See program A14-17.ASM on the disk.

18. See program A14-18.ASM for the answer to this problem. This would be inserted in the program of Figure 14-25 after the INT 16H, just below the RDKY: label.

19. A circular buffer is a special type of queue that has a tail pointer which is used to keep track of where the next byte is to be written to the buffer and a head pointer to keep track of where the next character to be read from the buffer is located. The buffer is circular because when the tail pointer reaches the highest location in the memory space set aside for it, it is wrapped around to the beginning of the buffer again. The

buffer-empty condition is indicated when the head pointer is equal to the tail pointer. The buffer-full condition is indicated when the (tail pointer + 1) is equal to the head pointer. See Figure 14-29 on page 517 of the text.

20. It is necessary to disable the UART interrupt input of the 8259A during part of the CHK_N_DISPLAY procedure so that an interrupt from the UART cannot call the SERIAL_IN procedure while CHK_N_DISPLAY is using the head and tail pointers of the circular buffer.

21. *a.* When changing a bit in a control word or interrupt mask, you should not change the other bits because it would change (mask or unmask) other interrupts.

b. The assembly language instructions to unmask IR5 of an 8259A at base address 80H without changing the interrupt status of any other bits are as follows:

```
IN AL,81H              ;Read 8259A mask register
AND AL, 11011111B      ;Unmask IR5
OUT 81H,AL             ;Send new mask register
                       ; to 8259A
```

22. No start or stop bits are needed for individual data characters in synchronous serial data communication because the receiver automatically knows that every 8 bits received after synchronization represents a data character. This is more efficient than asynchronous serial transmission, which requires a total of 10 bits to be sent for each 8-bit data character, hence wasting 20 percent of the transmission time.

23. *a.* If an 8251A is being used in synchronous mode for a BISYNC data link, the additional initialization words to send to the device are the desired sync character(s)—depending on whether the mode specified one or two sync characters—to the control address.

b. The 8251A detects the start of a message by detecting a sync character, SOH, STX, ENQ, ETX, and BCC bytes.

c. The 8251A asserts its SYNDET output pin and sets bit 6, the SYNDET bit, in its status register.

d. The receiving station in a BISYNC link indicates that it found an error in the received data by sending out an NAK message. See Figure 14-30 on page 518 of the text.

24. *a.* The start of a message frame is indicated in a bit-oriented protocol such as HDLC by detecting a specific pattern, 01111110, called a *flag*. See Figure 14-32 on text page 521.

b. A special circuit stuffs zeros in any data bytes that contain more than 5 1's in a sequence. The extra zeros are removed when the data is received.

c. The receiver in an HDLC system tells the transmitter that an error was found in a transmitted frame by sending an S frame containing the number of the last information frame that was correctly received.

25. *a.* See Figure 14-33 on page 522 of the text for a diagram of the network topologies.

b. The commercial systems that use the topologies are as follows:

Star: PABX phone system
Loop: GPIB
Ring: Apollo/HP Domain
Tree: Wangnet
Common-bus: Ethernet

26. Baseband transmission transmits one digital data signal directly. Broadband transmission allows voice, data, and video signals to be transmitted at the same time throughout the network by modulating a carrier frequency signal. See Figure 14-33 on page 522 of the text.

27. *a.* See Figure 14-34 on text page 523. The seven layers of the ISO open systems model are application, presentation, session, transport, network, data link, and physical.

b. The data-link layer of the protocol is responsible for assembling messages into frames or packets.

c. The transport layer of the protocol is responsible for ensuring that a message is transmitted and received correctly.

28. *a.* Ethernet is implemented in a common-bus topology with a single 50-ohm coaxial cable using baseband transmission at 10 Mbits/second.

b. The method used by a unit on Ethernet to gain access to the network for transmitting a message is called CSMA/CD (carrier sense-multiple access-collision detection). The unit looks at the coax to see if a carrier is present. If there is no carrier on the line, the unit starts transmitting. If a carrier is present, the unit waits for a time and tries again.

c. If a transmitting station finds that another station starts transmitting after it starts, it will stop transmitting and try again after a random period of time.

d. The condition that occurs when two units transmit at the same time is called a collision.

29. *a.* The method used by a unit on a token-passing ring to take control of the network for transmitting a message frame is as follows: It withdraws the not-busy token and changes it to a busy token, which it sends on around the ring. The transmitting unit then sends a frame of data around the ring to the intended

receiver.

b. The advantage of this scheme over the method used in Ethernet is that because signals travel in only one direction around the ring, it is ideally suited for fiber optic transmission.

c. If a token is lost while being passed around the ring (i.e., a token in the data stream is not detected in a specific period of time), the token ring network assumes the token was lost and recovers by clearing any leftover data from the ring and sending out a new not-busy signal.

30. *a.* See Figure 14-39*a* on text page 528. The interceptor determines that the file is in the workstation, so the request is passed on to DOS to read the file into memory.

b. Figure 14-39*b*. The interceptor part of the user system determines that the requested file is in the file server. The request is formatted and sent by netbios to the file server over the network. If the requesting user has access rights to the file, the file server will convert the file to packets and send the packets back to the user over the network.

31. *a.* The GPIB was designed to interface smart test instruments with a computer. See page 529.

b. The three types of devices which the GPIB defines are listeners, talkers, and controllers. See page 530.

c. The three signal groups of the GPIB are data bus, data byte transfer control, and general interface management. The data bus transfers data, addresses, commands, and status bytes among instruments. The data byte transfer control group consists of three handshake lines that coordinate the transfer of data bytes on the data bus. The general interface management lines are used to reset all units, request use of the bus, etc. See Figure 14-40 on page 530 of the text.

d. The sequence of handshake signals that take place when a talker on a GPIB transfers data to several listeners is as follows. When all listeners have released the NRFD line, the talker asserts the DAV line low to indicate that a valid data byte is on the bus. All the addressed listeners then pull NRFD low and start accepting the data. When the slowest listener has accepted the data, the NDAC line will be released high. The talker senses NDAC becoming high and unasserts its DAV signal. All the listeners pull NDAC low again, and the sequence is repeated until the talker has sent all of the data bytes it has to send. This handshake scheme allows talkers and listeners with very different data rates to operate correctly because the rate of data transfer is determined by the rate at which the slowest listener can accept the data. See Figure 14-40 on text page 530.

CHAPTER 15 THE 80286, 80386, AND 80486 MICROPROCESSORS

1. *a.* When a TSR program is loaded into memory and run, it remains resident in memory when it terminates. In other words, the memory allocated to the program when it was loaded is not deallocated when the program is terminated. This means the program can be run again by simply pressing some "hot key" sequence which vectors to the program through the keyboard interrupt handler. Figure 15-1*b* on page 535 of the text shows a memory map of how a TSR is loaded into the DOS system.
   b. Figure 15-2*a* on text page 536 shows how a passive TSR gets executed. Basically a passive TSR intercepts the keyboard interrupt, calls the keyboard read procedure, and interrupts the key code passed back from the keyboard procedure. If the keycode is a "standard" code, execution is simply passed back to the program interrupted by a keypress. If the code is a "special" code, the TSR carries out the action specified by that code and then returns execution to the interrupted program.

2. The two types of scheduling commonly used in multiuser/multitasking operating systems are time-slice and preemptive priority-based scheduling. In time-slice scheduling, the CPU executes one task for perhaps 20 ms, then switches to the next task. After all the tasks have had their turn, execution returns to the first. In the preemptive priority-based scheduling method, an executing task can be interrupted by a higher-priority task.

3. If two users in a time-share computer system each want to print out a file, a flag in memory is used to indicate if the printer is in use. If the printer is in use, the printing task becomes blocked and the users will have to line up to use the printer, i.e., join the queue of tasks.

4. Deadlock occurs when two tasks each control a resource that the other task needs to complete some action. For example, one task controls the printer and another task controls the disk drive, and each needs both resources to print a file. One way to prevent deadlock is to link resources together under one semaphore so that the two resources are accessed with a single action. Another way to prevent deadlock is to set up a hierarchy among tasks so that if deadlock occurs, the higher-priority task can gain access to all of the resources it needs.

5. A critical region is a piece of code for an operation that must be protected from access by other tasks until the operation is completed. A technique called mutual exclusion is used to prevent two tasks from accessing a critical region at the same time. Figure 15-3 on page 538 of the text shows assembly language instructions which use a semaphore to protect a critical region.

6. An overlay scheme is used to run programs, such as compilers, which are too large to be loaded into physical memory all at once. The compiler is written in modules, and its executive module is initially loaded into memory. An additional memory space, the overlay area, is then loaded with the module that the compiler requires at a particular time. When the compiler needs another module, that module is loaded into the overlay area.

7. *a.* An output port is used to enable one of several banks of physical memory at a time (see Figure 15-5 on text page 540). Each block maps into the same system address space when enabled. Thus, many blocks can be mapped into the system address space, one block at a time.
   b. LIM 4.0 allows 16-Kbyte blocks of memory to be bank switched into address spaces within the basic 1-Mbyte addressing range of the 8086 and 8088. The primary address windows used are the four pages between system address C800H and D800H. This effectively increases the amount of memory available for large programs, because megabytes of memory can be switched, in four pages at a time, as needed.
   c. Expanded memory involves switching blocks of memory into windows within the basic 1-Mbyte addressing range of the 8086 as described in part b. Extended memory means directly addressing memory in the address space above the basic 1-Mbyte space. An 80286 operating in protected mode activates 24 address lines, so it can directly address up to 16 Mbytes of memory. An 80386 operating in protected mode activates 32 address lines, so it can directly address up to 4 Gbytes of memory.

8. *a.* Virtual memory that can be logically referred to in programs, but is not all present in physical memory at one time. Segments or pages of virtual memory are loaded from disk

into the actual physical memory as needed to execute a program. Figure 15-8 in the text shows how a logical address is converted to a physical address by a memory management unit using a descriptor table. The selector in the logical address is used to access a descriptor in a table of descriptors in memory. The descriptor contains the physical base address, the privilege level, and some control bits for the segment. This base address is then added to the segment offset to produce the physical address.

b. If the MMU finds that a requested segment or page is not present in physical memory, it sends an interrupt to the CPU, which will then read the desired code or data segment from a disk or other secondary storage and load it into the physical memory. The MMU then computes and outputs the physical address.

c. Two other major advantages provided by indirect addressing are privilege levels and program protection.

9. The four main processing units in the 80286 are the bus unit, the instruction unit, the execution unit, and the address unit. The bus unit performs all memory reads and writes. The instruction unit fully decodes up to three prefetched instructions and holds them in a queue. The execution unit sequentially executes instructions it receives from the instruction unit. The address unit computes physical addresses in the real address mode; in the PVAM mode, it acts as an MMU. See Figure 15-8 on text page 542.

10. When an 80286 is operated in its real address mode, physical memory addresses are produced directly by adding an offset to a segment base, just as the 8086 does. In this mode, the 8086 acts like a "souped-up" 8086. In protected mode, the 80286 uses the selector part of the logical address to access a descriptor for a segment in a descriptor table. The descriptor contains the physical base address for the segment and the privilege level for the requested segment. If the user program has a high enough privilege level and the segment is present in physical memory, the address unit in the 80286 will add the segment base from the descriptor to the offset from the original instruction to produce the physical address.

11. An interrupt is caused by some external condition applied to INTR or NMI input. An exception is caused by an error condition during execution of an instruction. A fault is detected and signaled before an instruction executes (e.g., segment not present exception). A trap is detected and signaled after an instruction executes (e.g., divide by zero exception).

12. An 80286 may be switched from real address mode to PVAM operation by setting the protection enable bit of the machine status word. The MSW is transferred from a register or memory location to the 80286 with the LMSW instruction. The only way the 80286 may be switched back to real address mode operation is by resetting the system.

13. *a.* Fourteen bits in the selector part of the logical address specify one of 16,384 possible segment descriptors. The offset part of the logical address contains 16 bits, so each segment can be up to 65,536 bytes in length. 16,384 segments × 65,536 bytes/segment = 1 Gbyte virtual memory.

b. In real mode only 16 address lines are active, so only 1 Mbyte of memory can be accessed. In PVAM 24 address lines are active, so up to 16 Mbytes of physical memory can be addressed. Of course, the actual memory addressable is limited by the amount of physical memory present in a given system.

14. *a.* Three major advantages of the 80386 over the 80286 are: (1) it executes instructions faster due to a faster clock and more extensive pipelining; (2) for the DX version, it can address about 4 Gbytes of physical memory and about 64 Tbytes of virtual memory; and (3) it can access virtual memory on a demand paged basis as well as on a segment basis.

b. The 386DX has a 32-bit address bus and a 32-bit data bus, so it can directly access 4 Gbytes of physical memory and directly read/write 32-bit data words. The 386SX is a lower-cost version which has the same instruction set and basic internal architecture as the DX, but has a 16-bit data bus and a 24-bit address bus. The reduced buses mean it can directly access only 16 Mbytes of physical memory and can read/write at most 16- rather than 32-bit words.

c. The BE0–BE3 signals are the bank enable signals. The memory for a 386DX system is set up as 4 byte-wide banks, and these signals are used to enable individual banks so the processor can read bytes, words, or double words.

15. *a.* The main difference is that the ISA has 24 address lines and 16 data lines, whereas the EISA has 32 data lines and 32 address lines.

b. MCA boards are smaller and use different edge connectors than those on an EISA board. See Figure 15-18*a* on text page 550 and Figure 15-19 on text page 552.

c. With the EISA, a master asserts an individual MREQ line if it needs to use the buses. If it is the highest master requesting service, the bus controller will respond with a MACK signal to tell that master it can use the buses.

With the MCA, any master can assert a common PREEMP line low to request use of the buses. The central control circuitry asserts the ARB/GNT line, and then each master attempts to put its preassigned arbitration code on the common ARB0–ARB4 lines. The master with the lowest arbitration code will be granted control of the buses.

16. *a.* Fourteen bits in the selector part of a virtual address specify one of 16,384 segments. The offset part of the virtual address is 32 bits, so each segment can be as large as 4 Gbytes. 16,384 segments × 4 Gbytes/segment = about 64 Tbytes of virtual memory.
    b. In real mode only the lower 20 address lines are active, so a 386DX or 386SX can access only 1 Mbyte of physical memory. In protected mode all 32 address lines of a 386DX are active, so about 4 Gbytes of physical memory can be accessed. A 386SX in protected mode activates all 24 address lines, so up to 16 Mbytes of physical memory can be accessed.

17. *a.* The two parts of an 80386 virtual address are the selector and the offset.
    b. The 80386 automatically multiplies the index value in the upper 13 bits of the selector by 8 and adds the results to a descriptor table base address. If the TI bit in the selector is a 0, the result will be added to the global descriptor table base contained in the global descriptor table register. If the TI bit in the selector is a 1, the result will be added to the local descriptor table base address contained in the local descriptor table register. The 80386 then fetches the indexed segment descriptor from the specified descriptor table and loads it into the hidden part of the segment register. The 32-bit segment base address, loaded in as part of the descriptor, is added directly to the 32-bit offset part of the virtual address to produce the physical address of the desired byte or word.
    c. If the memory location is in the global memory area, the 80386 will use the same overall method of accessing it, except that the index value multiplied by 8 will be added to the global descriptor table base address in the GDT register instead of to the base address in the LDT register.
    d. The 386 holds the base address for the global descriptor table in the global descriptor table register. The base address for the currently used local descriptor table is held in the local descriptor table register.
    e. The length of a segment is contained in the segment descriptor, so the 386 can generate an exception if a program instruction attempts to access a location outside the segment.

18. User tasks are protected from one another in an 80386 system by the fact that a task cannot directly access descriptors in the local descriptor table that belongs to another task.

19. In an 80386 protected-mode operating system, kernel procedures are put at the highest privilege level. Application programs operate at a lower privilege level. A task operating at a lower privilege level cannot directly access a procedure at a higher privilege level. Access to a higher privilege level can be done only through some controlled mechanism such as a call gate.

20. The mechanism used to allow a task at a level 2 privilege to call a procedure at a higher privilege level is referred to as a *gate*. A gate is simply a special type of descriptor. When a call is made to a procedure at another privilege level, the selector is loaded into the CS register and a call gate descriptor is loaded into the hidden part of the CS register. If the call is legal, the selector for the procedure (contained in the call gate descriptor) is loaded into the CS register and the actual descriptor for the segment containing the procedure is loaded into the hidden part of the CS register. The procedure is then accessed. This indirect call through the gate descriptor allows the access to be limited to tasks having a higher priority level than a privilege level specified in the gate descriptor.

21. *a.* Descriptors for each task state segment are kept in the global descriptor table. The task register in the 80386 holds the selector and the descriptor for the task which is currently executing. During a task switch, the task register is automatically loaded with the selector and the descriptor for the new task.
    b. When the 386 does a far jump or call to switch tasks, the privilege level of the call is checked. The RPL of the CALL selector and the CPL of the current task must both be less than the DPL of the desired segment or a privilege level exception will be generated. If the privilege level is high enough, the 386 will store all the register values for the executing task in its task state segment. It will then load the task register with the selector and the descriptor for the new task. Then it will copy the register values for the new task to the appropriate registers. Execution will continue with the segment and EIP values loaded as part of this operation.

22. *a.* Register CR3 in the 386 holds the 32-bit base address for a page directory in memory. The upper 10 bits of the 32-bit linear address from the segmentation unit are used to access one of the entries in the page directory. This 32-bit entry in the page directory is the base address of a page table in memory. The middle 10 bits of the linear address from the segmentation unit are used

to select one of the entries in this page table. The selected entry in the page table contains the 32-bit starting address for the desired page. The lowest 12 bits of the linear address from the segmentation unit select the desired byte word, or double word, in the selected page.

b. Pages are usually much smaller than segments, so they can be swapped in and out of physical memory much faster than segments can.

c. In the simple flat addressing model, all 386 segments start at address 0 and extend through the full 32-bit address range of the 80386. All code and data words are accessed with 32-bit offsets from the same segment base address, 0. This mode effectively eliminates segmentation.

In the paged flat memory model, all the segments are initialized to start at absolute address 0 and extend through the full 32-bit address space of the 386, thus effectively turning off segmentation. However, paging is enabled to provide virtual memory capability and some degree of protection.

23. a. If the FM bit in the EFLAGS register for the new task is set when the 386 does a task switch, the 386 will switch to virtual 8086 mode to execute the new task.

b. The INT 21H instruction will cause a task switch to the virtual machine monitor, which operates in protected mode. The VMM monitor will call the equivalent protected-mode operating system function and then return execution to the virtual 8086 task.

24. See program A15-24.ASM on the disk.
Algorithm: (Uses 80386 shld instruction)

Copy operands into two 32-bit registers.
Shift 8 bits of upper double word left into a third register.
Shift 8 bits from lower double word into upper double word.
Rotate 8 bits saved in third register 24 bits left.
Shift 8 bits from third register left into lower word.

25. The 80486 has an on-board, 8-Kbyte data and instruction cache, a built-in floating-point math coprocessor, and more extensive pipelining, which allows several instructions to be in various stages of decoding and execution at the same time.

26. Features of a RISC-based computer include the following:

a. The instruction set is limited to simple types, so instruction decoding is simpler and faster.

b. It uses extensive pipelining, so several instructions can be fetched, decoded, and executed at the same time.

c. Next instructions are fetched from both the sequence after the jump instruction and the jump destination instruction. This eliminates the usual delay when the jump is taken.

d. It contains a large number of on-board registers because register-register operations are faster than register-memory and memory-memory operations.

27. The major advantage of parallel processor configurations is that different processors can work on different parts of a program at the same time rather than on the parts in series. A disadvantage of parallel processing computers is that writing programs to run on them is currently somewhat more difficult.

28. a. If the sum of the signals on the inputs is greater than a specified threshold value, the neuron fires and outputs a signal to other neurons.

b. A neural network is trained to produce the correct response rather than programmed. A set of input conditions is applied to the inputs, and the network "learns" by adapting the weights of the interconnections until the output response is correct. If several different sets of input conditions are applied in sequence, the network will generalize its response so that it responds correctly for new but similar input signals.

c. Advantages of neural network–type computers are that they can be taught rather than programmed; they can learn; input data does not have to be precise, because the network averages out the effect of any one signal value; and information is stored in a distributed manner, which is highly fault tolerant.

d. Neural networks are ideally suited to applications such as weather forecasting, stock market analysis, and speech recognition where neither the data nor the rules for processing the data are precisely known.

29. a. A traditional digital logic control system can work only with the fixed values 0 and 1. In a fuzzy logic system, the variables can have values other than 0 and 1, so the controller can recognize many combinations of the input variables which represent a given state. See Figure 15-35 on page 574 of the text.

b. Advantages of fuzzy logic control systems are that they can use imprecise values such as hot, warm, light, and dark and can provide very smooth control of mechanical systems.

Experiment Notes and Answers to Selected Questions in the Experiments Supplement

Experiment 1 Introduction to the SDK-86 Microprocessor Development Board

1.	Device	Intel SDK-86	URDA SDK-86
a.	CPU	1 — 8086	1 — 8086
b.	ROMs	4 — 2716s or 2616s	1 — 2732As
c.	RAMS	4 or 8 — 2142 2K or 4K	1 —16K 6264
d.	I/O ports	2 — 8255As	2 — 8255As
e.	Keypad/display	1 — 8279	1 — 8279

2.	Intel SDK-86	URDA SDK-86
a.	2K ending at 007FFH	16K ending at 003FFFH
b.	00100H first user RAM	00100H first user Ram
c.	006FFH bytes of RAM	03EFFH bytes of RAM
d.	8K bytes of ROM	8K bytes of ROM

Program on disk:
P1.ASM Incrementing Hex Count

Experiment 2 Debugging Programs with the SDK-86 Keypad Monitor

Using the Single-Step Command

5. The problem with single-stepping all the way through this program is that you have to press the "," key about 195,000 times to get past the loop containing the PAUSE label (65,000 × 3).

Modifying the Operation of a Program

1. The number in the SI register should be 0 at offset 003AH.
4. The EW command can be used to change the word loaded into the SI register.
5. The address in memory of the FFFFH part of the MOV SI, OFFFFH instruction is 0000:134H.
8. The new value will make the display change more quickly.

Experiment 3 A Simple 8086 Arithmetic Program

1. The algorithm is as follows:
   Load AH and AL with required numbers
   Add AH and AL, leave result in AH
   Put the result into the BL register
   Move the carry flag into the LSB of BH
   Zero all but the LSB of BH
   STOP

2. a. You would want to use the ADD instruction, because you don't want to add the status of the carry flag to the result.
   b. You can use the instruction MOV BL, AH
   c. You can use the instruction RCL BH, 01.
   d. You can use the instruction AND BH, 01.

4. ADD AH, AL ;AH = AH + AL

5. 02 E0

6.
```
02 E0      ADD AH, AL     ;AH = AH + AL
8 A DC     MOV BL, AH     ;BL = AH
D0 D7      RCL BH, 01     ;Save carry flag
80 E7 01   AND BH, 01     ;Zero all but LSB
CC         INT 3          ;INT 3 break point
```

10. AH = 42H, AL = 35H

    Results: AH = 77H, AL = 35H, BH = 0, BL = 77H

14. AH = 9AH, AL = 78H
    Results: AH = 12H, AL = 78H, BH = 1, BL = 12H

Experiment 4 Writing a Program Which Accesses Data in Memory

1. Offset
   0000 — NUMBER 1
   0001 — NUMBER 2
   0002 — SUM
   0003 — CARRY

2. The algorithm is as follows:
   Add NUMBER 1 to NUMBER 2
   Adjust for BCD
   Put the result into SUM
   Put the contents of the carry flag into CARRY
   STOP

3. *a.* The following instructions are needed to initialize the DS register so that program instructions can access the data locations in memory:
   MOV AX, DATA_HERE
   MOV DS, AX

4. 104H is the offset of the first instruction of the program.

Listing of Program

```
0100 46                  NUMBER_1    DB 46 H
0101 38                  NUMBER_2    DB 38H
0102 00                  SUM         DB 00
0103 00                  CARRY       DB 00
0104 B8 1000             MOV AX, DATA_HERE
0107 8E D8               MOV DS, AX
0109 A0 00 00            MOV AL, NUMBER_1
010C 02 06 01 00         ADD AL, NUMBER_2
0110 27                  DAA
0111 A2 02 00            MOV SUM, AL
0114 D0 16 03 00         RCL CARRY, 01
0118 CC                  INT 3
```

9. NUMBER_1 = 89H, NUMBER_2 = 93H, SUM = 82H, CARRY = 01H

3. *continued*
   b. The ADD instruction is more reasonable to use in this program because you don't want the value of the carry flag added to the result.
   c. The DAA instruction must be used to adjust the sum to the correct BCD form.
   d. The following instructions can be used to copy the carry flag into the least significant bit of a byte and save zeros in the upper 7 bits of the byte:

```
MOV CARRY, 0
RCL CARRY, 01
```

   e. The INT 3 instruction can be used at the end of the program to prevent accidentally executing the random contents in RAM after the program.

```
;result = 84
;carry = 0
;Initialize the
; DS register
;AH = NUMBER 1
;AH = SUM
;Adjust for BCD
;Save the result
;Save carry flag
```

10. *a.* To do the subtraction, you can use the SUB AL, NUMBER_2 instruction.

b. The DAS instruction is needed to adjust the result of the subtraction to BCD form.

Listing of Subtraction Program

```
0100 81                 NUMBER_1    DB 81H
0101 67                 NUMBER_2    DB 67H
102 00                  RESULT      DB 00           ;result = 14
0103 00                 CARRY       DB 00           ;carry = 0
0104 B8 1000            MOV AX, DATA_HERE           ;Initialize the
0107 8E D8              MOV DS, AX                  ;DS register
0109 A0 00 00           MOV AL, NUMBER_1            ;AH = NUMBER 1
010C 2A 06 01 00        SUB AL, NUMBER_2            ;AH = SUM
0110 2F                 DAS                         ;Adjust for BCD
0111 A2 02 00           MOV RESULT, AL              ;Save the result
0114 D0 16 03 00        RCL CARRY, 01               ;Save carry flag
0118 CC                 INT 3
```

Experiment 5 Using a Computer to Develop Assembly Language Programs

Program on disk:
P5.ASM Averaging Two Temperatures

Experiment 6 Using DEBUG to Execute and Debug Programs

11. The new value of AV_TEMP is 38H.

14.
```
MOV     AX, DATA_HERE
MOV     DS, AX
MOV     AL, HI_TEMP     ;AL = 92H
ADD     AL, LO_TEMP     ;AL = E4H
MOV     AH, 00H         ;AH = 00H
ADC     AH, 00H         ;AH = 00H
MOV     BL, 02H         ;BL = 02H
DIV     BL              ;BL = 02H, AL = 72H, AH = 00H
MOV     AV_TEMP, AL
```

Program on Disk:
P6.ASM Averaging Two Temperatures

Experiment 7 Using TURBO DEBUGGER to Run and Debug Programs

There are no questions in this experiment.

Experiment 8 Downloading Programs to the SDK-86 for Execution and Debugging

14. With HI_TEMP equal to 20H and LO_TEMP equal to 40H, AV_TEMP will equal 30H.

Experiment 9 Reading Characters from a Keyboard on a Polled Basis

2. *a.* It is necessary to poll the keypressed strobe until it is unasserted before looping back to look for a high so that you don't keep reading in the same key code while the strobe is still high.
   b. A programming structure to use to represent polling the keypressed strobe until it is asserted or unasserted is REPEAT_UNTIL.
   c. A way to indicate in the algorithm that you want to input the codes for ten pressed keys and then stop is to use a REPEAT_UNTIL or a FOR_LOOP structure.

3. *c.* You can use the following instructions to determine if the strobe signal in the LSB position is asserted and loop until it is asserted:

```
LOOK_STROBE_HI:  IN    AL, DX
                 AND   AL, 01
                 JZ    LOOK_STROBE_HI
```

   d. You can use the following instructions to determine if the strobe signal in the LSB position is unasserted and loop until it is unasserted:

```
LOOK_STROBE_LO:  IN    AL, DX
                 AND   AL, 01
                 JNZ   LOOK_STROBE_LO
```

   e. You can use the following instructions to make the program read in ten key codes before going on:

```
                 MOV   BX,  OFFSET TABLE
                 MOV   CX,  10
LOOK_STROBE_HI:  IN    AL,  DX
                 AND   AL,  01
                 JZ    LOOK_STROBE_HI
                 MOV   DX,  OFFF8H
                 IN    AL,  DX
                 AND   AL,  7FH
                 MOV   [BX], AL
                 INC   BX
                 MOV   DX,  OFFFAH
LOOK_STROBE_LO:  IN    AL,  DX
                 AND   AL,  01
                 JNZ   LOOK_STROBE_LO
                 LOOP  LOOK_STROBE_HI
```

Program on disk:
P9.ASM Input Characters from Keyboard

Experiment 10 ASCII-to-Hex Conversion

Algorithm for Lab 10:
IF number < 30H THEN error
ELSE
    If number < 3AH THEN subtract 30H (it's a number 0-9)
  ELSE (number is >39H)
    IF number < 41H THEN error (number in range 3AH–40H)
    ELSE
      IF number < 47H THEN subtract 37H (it's a letter 41–47)
      ELSE error

Program on disk:
P10.ASM ASCII-to-Hex Conversion

Experiment 11 Generating Digital Waveforms on an Output Port

2. The waveform is high for 71 microseconds and low for 71 microseconds.
3. The program algorithm is as follows:
    Initialize ports
    Point to output port
    Initialize AL low
  REPEAT
    Output AL
    Wait time delay for one half cycle
    Change value of LSB of AL
  UNTIL FOREVER
4. The assembly language program is as follows:

```
CODE_HERE       SEGMENT
                ASSUME CS:CODE_HERE
START:  MOV     DX,  0FFFFH     ;Point DX to port control register
        MOV     AL,  99H        ;Control word to set up P1B as output
        OUT     DX,  AL         ;P1A, P1C inputs, send control word
        MOV     DX,  OFFFBH     ;Point to output port
        MOV     AL,  0          ;Initialize AL low
;Start outputting waveform
AGAIN:  INC     AL              ;Change value of LSB of AL
        OUT     DX,  AL         ;Output it to port
        MOV     CX,  07H        ;Set up counter for wait
DELAY1: NOP                     ;Now wait
        LOOP    DELAY1          ;
        JMP     AGAIN           ;Send next half cycle
CODE_HERE       ENDS
                END     START
```

5. The computer's period =
$1/(2.45 \times 10^6 \text{Hz}) = 0.408\ \mu\text{s}$.

6. The number of CPU clock cycles for each half-cycle of the waveform is 142 microseconds/(2 × 0.408 microseconds) = 174 clock cycles (CT).

7.
```
AGAIN:   INC AL          3 clock cycles
         OUT DX, AL      8 clock cycles
         MOV CX, 07H     4 clock cycles
DELAY1:  NOP             3 clock cycles
         LOOP DELAY1     17 or 5 clock cycles
         JMP AGAIN       15 clock cycles
```

8.
```
AGAIN:   INC AL          3 clock cycles
         OUT DX, AL      8 clock cycles
         MOV CX, 07H     4 clock cycles
         JMP AGAIN       15 clock cycles
                         C_O = 30 clock cycles
```

9.
```
DELAY1:  NOP             3 clock cycles
         LOOP DELAY1     17 or 5 clock cycles
                         C_L = 20 or 8 clock cycles
```

10. $N = (C_T - C_O + D)/C_L$
$N = (174 - 30 + 12)/20)$
$N = 7.8$

17. The frequency on D1 is one-half the frequency on D0.

18. The program produces all these different frequencies because the port lines simulate an 8-bit binary counter.

Program on disk:
P11.ASM 7040-Hz Square Wave

Experiment 12 Working with Strings

1. Algorithm for part 1:

```
Set up address pointers to each string
Set up character count
REPEAT
        Move character of source string to
        destination string
        Increment pointers
        Decrement character count
UNTIL character count = 0
```

2. See program P12-1.ASM on the disk.

4. Algorithm for part 2:

```
Set up pointer to start of character string
Set up counter with length of string
REPEAT
        Compare the character with a carriage
        return character
        Decrement count
UNTIL count = 0 OR carriage return is found
IF carriage return is found THEN
        put string length in AL
ELSE put 50H in AL
```

5. See program P12-2.ASM on the disk.

7. Algorithm for part 3:

```
Set up counter
Set up pointer to name string
REPEAT
        Compare the character in source string
        with a period
        Decrement counter
UNTIL counter = 0 OR character = period
IF period not found THEN error condition, do no
   more
Increment pointer to next character
Compare next character with space character
IF space not found THEN error condition, do no
   more
Increment pointer to next character (start of
   TUNA)
Decrement counter
Save the state of this counter
Set up pointer for destination string
REPEAT
        Move character to destination string
        Decrement count
UNTIL counter = 0
Move a comma character to the destination
   string
Increment the destination pointer
Set up pointer to beginning of source string
Set up counter = number of characters in name -
   length of second name (saved above)
Decrement counter (for space not moved)
REPEAT
        Move character to destination string
        Decrement character count
UNTIL counter = 0
```

8. See program P12-3.ASM on the disk.

11. Algorithm for part 4 (optional):

```
Initialize pointer to end of source string
Initialize pointer to end of destination string =
end of source string + 4
Put number of characters in source string into
counter
Make pointers decrement when move is being
done
```

```
REPEAT
        Move character from source string to
          destination decrement character count
UNTIL character count = 0
```

Programs on disk:

P12-1.ASM	Move a String
P12-2.ASM	Search a String
P12-3.ASM	Move and Change a String
P12-4.ASM	Move a String with Overlap

Experiment 13 Procedures—Nested Loops and Microcomputer Music

Part 1

2. Algorithm for part 1:

```
Initialize output port
Point to output port
REPEAT
        Output the 2.5-Hz signal low
        Wait one half-period
        Output the 2.5-Hz signal high
Wait one half-period
UNTIL FOREVER
```

3. See program P13-1.ASM on the disk.

4. See program P13-1.ASM on the disk.

5. Time required for 1 clock cycle using 2.45-MHz. PCLK = 0.408 μs.

6. 0.2 μs per half-cycle/0.408 μs per clock = 490,196 clock cycles.

8. See program P13-1.ASM on disk.

Part 2

1. Algorithm for part 2:

```
Initialize the stack segment and stack pointer
Set up an output port
Point to output port
REPEAT
        output 7040 Hz signal for 0.2 sec.
        output constant low for 0.2 sec.
UNTIL FOREVER
```

2. See program P13-2.ASM on the disk.

5. *a.* To change the time for which a note plays, increase the outer loop constant.

   b. To change the frequency of the note, change the inner loop constant.

   c. The delay constant for the frequency of the note affects the constant required for the time the note plays.

Part 3 (Optional)

See program P13-3.ASM on the disk.

Programs on disk:

P13-1.ASM	2.5-Hz Square Wave
P13-2.ASM	Pulsed 7040-Hz Square Wave
P13-3.ASM	Tune

Experiment 14 Procedures—Measuring Reaction Time with a Microcomputer

1. The algorithm is as follows:

```
Initialize data segment
Initialize stack segment
Initialize input and output ports
Call WAIT-2S
Initialize counter
Set R_TIME to 0
Point to output port
Light LED
Point to input port
REPEAT
        Wait 1MS
        Increment counter
        Check switch
UNTIL switch high
Turn off LED
```

2. See program P14-1.ASM on the disk.

Programs on disk:
P14-1.ASM Measuring Reaction Time
P14-2.ASM Test Program

Experiment 15 Microcomputer Circuits and Bus Signals

1.

Function	Part Number	
	Intel SDK-86	**URDA SDK-86**
CPU	8086	8086
ROM	2—2616	2—2732A
	2—2716	
RAM	4 or 8—2142	1—6264
Parallel ports	8255A-5	8255A
Address bus latches	74S373	74S373
Data bus transceivers	8286	8286
Clock generator	8284	8284

2. *b.* CLK = 4.9152 MHz; PCLK = 2.4576 MHz

3. Jumper on W40, PCLK = 2.45 MHz
Jumper on W41, PCLK = 4.92 MHz

8. *a.* 0100 for first ALE
0102 for second ALE
0104 for third ALE
b. The 8086 outputs these three addresses during the instruction execution to fetch the instructions for the queue.

9. *b.* See Figure 15-1.
c. A14 is the address latch which demultiplexes the address bit from the D1 line.
d. AD1 enters A14(7) and A1 exits A14(6).
g. See Figure 15-1.

10. *b.* See Figure 15-1.
c. The RD pulses are interleaved between the ALE pulses because you have to enable the address latches with ALE before you can read from an address.

12. *a.* FEEB 9090 5504.
b. The three words represent the data fetched during each memory read. If you write the words as bytes (LSB first), you will see the instructions for the program. Note the random information between the read and the next instruction.

14. The W27-W34 jumper pins specify how many WAIT states are inserted.

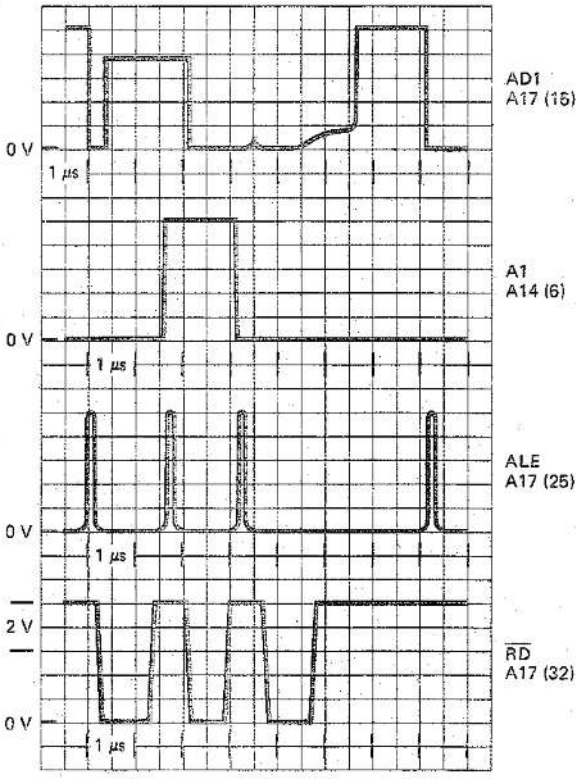


FIGURE 15-1.

Experiment 16 Address Decoders, I/O Operations, and WAIT States

3. The 01 output, A26(14), enables the keypad monitor ROMs.

4. ROM decoder (3625) — A26
   Keypad monitor ROMs — A27 and A30
   ALE — A17(25)
   ROM decoder output — A26(14)

5. See Figure 16-1.

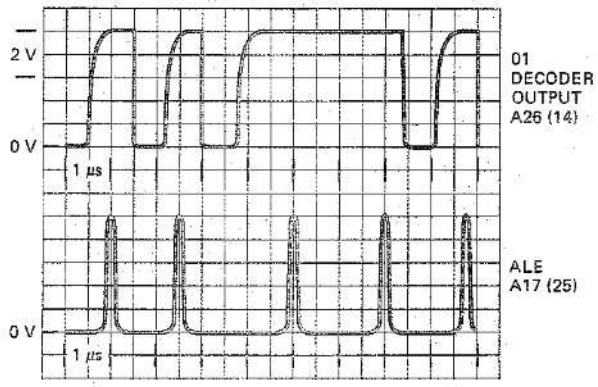


FIGURE 16-1.

6. The 02 output, A26(13), (which enables the serial monitor ROMs A36, A37), is constantly high (not asserted) because the keypad monitor program is running.

7. When the serial monitor program is running, there will be pulses present on the address decoder output, A26(13).

8. *a.* See Figure 16-2.
   b. A38 and A41 contain the byte at address 00100H.
   c. A43 and A45 contain the byte at address 00101H.
   d. The RAM address decoder PROM outputs 01 and 03 are connected to the even byte devices.
   e. The RAM address decoder PROM outputs 02 and 04 are connected to the odd byte devices.

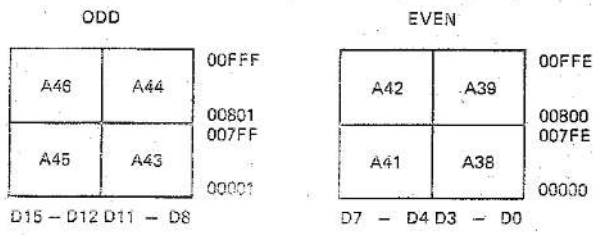


FIGURE 16-2.

10. *a.* See Figure 16-3.
    b. Both decoder outputs are enabled when words are read from memory.

11. The 04, A22(11), enables port P1A.

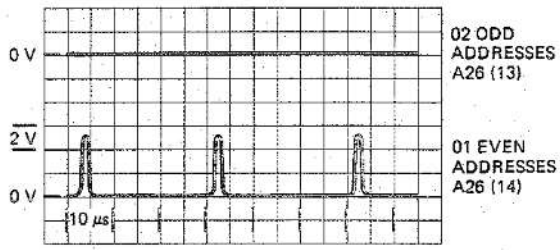


FIGURE 16-3.

12. The codes for the instructions are BA FB FF EC EB FA.

13. See Figure 16-4.

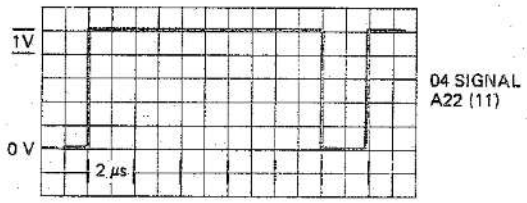


FIGURE 16-4.

14. The W27–W34 jumper pins specify how many WAIT states are inserted.

17. *a.* See Figure 16-5.
    b. Twenty-four clock cycles are required for one loop execution

19. See Figure 7-1b in the text and add seven WAIT states after T3. Now 31 clock cycles are required for one loop execution. The extra cycles are inserted after RD goes low to allow the input port more time to present data on its data lines.

20. Twenty-six clock cycles are now required for one loop execution.

21. Thirty-two clock cycles are now required for one loop execution.

22. The WAIT state generator is now adding WAIT states for every read and write operation as well as for every I/O operation.

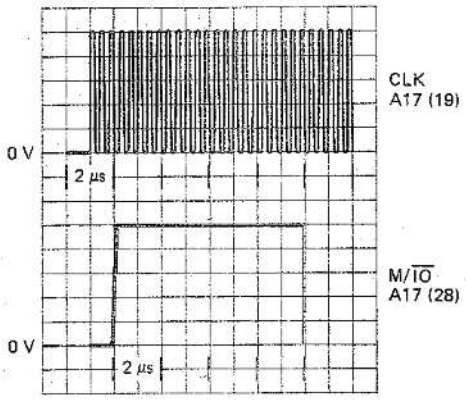


FIGURE 16-5.

Experiment 17 Looking at Microcomputer Signals with a Logic Analyzer

12. Set analyzer to clock on falling edge of ALE, because address is valid on data bus at that time.

13. One sequence of addresses should be 0100, 0102, and 0104.

16. Clock the analyzer on the rising edge of the RD signal, because data from memory is valid on the data bus at that time.

20. One sequence of data words should be FEEB, 9090, and 5504.

21. Clock the analyzer on the rising edge of the WR signal, because valid data from the 8086 is present on the data bus at that time.

23. The M/IO signal indicates whether the 8086 is doing a memory or port operation. If this signal is low, a write to port is occurring.

27. The correct sequence of data bytes written to the port is 00, 01, 02, 03, 04, 05, 06, 07, 08, and 09.

28. The first 10 data bytes written to memory should all be 55.

Experiment 18 Further Practice with a Logic Analyzer

1. Use CLK, A17(19) as the external clock signal (falling edge).
   Trigger word = FFF0.

7.

FFF0	9CEA
FFF2	0000
FFF4	FFFF
FFF6	0001
F09C	2EFA
F09E	168E
F0A0	0098
F0A2	50BC
F0A4	8600
F0A6	2EEC

8. The first instruction that is being fetched from memory, starting at address FFFF0, is EA 9C 00 00 FF. This is an intersegment jump to FF09CH.

9. The 8086 outputs address FFF06 and fetches a data word from that address as part of its prefetch operation.

10. The destination address for the JMP instruction is FF09CH. The data word at FF09H is 2EFAH. The low byte, FAH, is the CLI instruction.

15. You would expect to see 10 or more samples of the address FFF0H. If the logic analyzer clock pulses take 40 ns each and a processor clock cycle takes 408 ns, then 10 (408/40) samples are taken of the bus during one processor clock cycle. If the trigger word is present for 10 logic analyzer clock cycles, which represents 10 × 40 ns = 0.4 μs, the resolution is 40 ns.

16. 9CEA is on the bus for 27 logic analyzer cycles. This represents 27 × 40 ns = 1.08 μs.

19. There are 21 logic analyzer clock cycles between $\overline{RD}$ going low and valid data appearing on the outputs of the monitor ROMs (i.e., $\overline{RD}$ going high again). This represents 21 × 40 ns = 840 ns. The resolution of this measurement is 40 ns.

20. The greater resolution in this mode gives a more accurate value.

Experiment 19 Microcomputer Troubleshooting

5. a. The 8086 RDY is continuously low; check 8284, 74LS164, 74LS04, 74LS10.
   b. The 8086 RST is continuously high; check 8284, 74LS04, 74LS164.
   c. Check address latches 74S373 and processor ALE.
   d. No enable on monitor ROMs; check 3625.
   e. LEDs all 8's; check 7445.

7. See program P19.ASM for algorithm and program.

Program on disk:
P19.ASM Memory Testing

Experiment 20 Using 8086 Interrupts— A Real-time Clock

Hardware

2. If you connect the 1-Hz signal to the NMI before you are ready to test your program, an interrupt will occur before the interrupt procedure address is in place. Program execution will jump to whatever CS and IP addresses are in the locations 0000AH and 00008H.

Software

3. Interrupt service procedure algorithm for clock (update clock):

```
Increment seconds count

IF seconds count = 60 THEN
    seconds = 0
    increment minutes
    IF minutes count = 60 THEN
            minutes = 0
            increment hours count
            IF hours = 24 THEN hours = 0
Update clock display on LEDs
```

Programs on disk:
P20A.ASM Real-time Clock Main Module
P20B.ASM Clock Procedure Module
DISPLY.ASM Display Procedure Module

Experiment 21 Using an 8254 to Generate Real-time Clock Interrupts

Answers to questions can be found in program P21B.ASM.

Programs on disk:

P21A.ASM Real-time Clock Main Module with 8254 Init
P21B.ASM 1 KHz Input Clock Procedure Module
DISPLY.ASM Display Procedure Module

Experiment 22 Initializing and Using an 8259A Priority Interrupt Controller

Answers to questions can be found in program P21B.ASM.

Programs on disk:

P22A.ASM Real-time Clock Main Module with 8259A Init
P22B.ASM 1 KHz Input Clock Procedure Module with EOI
DISPLY.ASM Display Procedure Module

Experiment 23 Interfacing an Unencoded Keyboard to a Microcomputer

14. *a.* SRC = 00 000 001 for shift + B
    b. The shift + B code will be the second in the table.
    c. The ASCII code for B (42H) should be second in the table. See the program for how the codes are placed in the table.
    d. SRC = 01 000 000 if you press the A key (no shift). The ASCII code for "a" (61H) should be placed at an offset of 40H (the 65th position) in the table.

19. *a.* Using the control key would double the size of the table.
    b. Put EBCDIC codes in the table instead of ASCII codes.

Programs on disk:

P23A.ASM Unencoded Keyboard Main Module
P23B.ASM Detect, Debounce, Encode Procedure Module
DISPLY.ASM Display Procedure Module

Experiment 24 Speech Synthesis and Handshake Data Output

8. *a.* To determine if execution ever gets to the interrupt-service procedure, put a break point just inside the procedure.

   b. To determine if the 8255A ever sends an interrupt signal to the 8259A, look for signal on IR0 with a scope.

   c. To determine if an interrupt signal gets to the 8086, look for a signal on the INTR pin.

   d. To determine if the 8086 responds to the interrupt from the 8259A, put a scope on the INTA pin of the 8086 to see if it pulses. If it pulses, it is acknowledging the interrupt.

   e. Look at the data inputs of the Digitalker to see if the codes being output from the 8255A get there.

   Programs on disk:

P24-1.ASM	Digitalker Using Only SSR1 and SSR2
P24-2.ASM	Digitalker Using All Four SSR ROMs

Experiment 25 Lighting an LED Matrix—Timed Output and Multiplexing

1. *a.* A logic high is required to turn on a row.

   b. A logic high is required to turn on a column.

   c. Output 01H to port A and 0FFH to port B to light the entire top row of LEDs.

   d. Output 0FFH to port A and 80H to port B to light the entire leftmost column of LEDs.

Programs on disk:

P25-1.ASM	Light Top Corner LED
P25-2.ASM	Light Each Row in Turn
P25-3.ASM	Light X Pattern
P25-4A.ASM	Use Interrupts to Light X Pattern (Mainline)
P25-4B.ASM	Display Update Module
P25-5A.ASM	Displaying Characters on LEDs (Mainline)
P25-5B.ASM	Display Scroll Module

Experiment 26 Analyzing the SDK-86 Keyboard and Display Circuitry Operation

1. See program P26.ASM on the disk.
4. A low turns on the digit driver transistors.
5. See Figure 26-1 on the next page.
6. See Figure 26-2 on the next page.
   a. The rightmost digit gets turned on first in the refresh sequence.

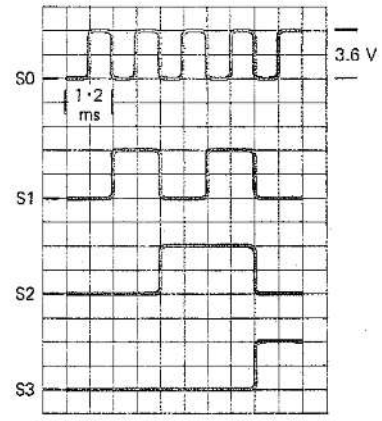


FIGURE 26-1.

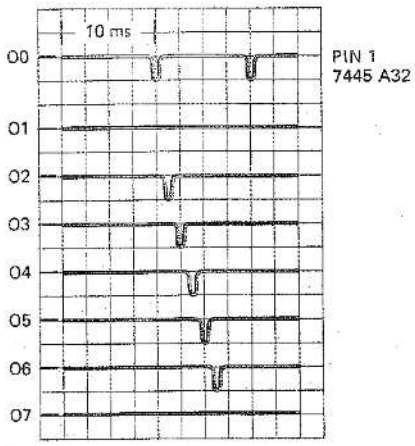


FIGURE 26-2.

b. No two digits are ever turned on at the same time.

c. Each digit is turned on once every 10 ms. Note that the 7445 on the SDK-86 board is connected to all four SL outputs of the 8279, so it must cycle through all 16 values before 00 goes low again.

d. The multiplex rate is 1/10 ms = about 100 Hz.

e. Each digit is turned on for 0.64 ms during a refresh cycle.

7. See Figure 26-3. The low blanking codes are present between the actual seven-segment codes. The reason the lows on the seven-segment codes appear larger than the highs is that the low blanking codes on each side of them look like part of the segment code.

8. The 2Y0 output cannot go high because there is no pull-up resistor connected to these outputs until a key is pressed. (Pull-up resistors are built into the 8279). Pressing the EB key connects a pull-up resistor to the 2Y0 output. The waveform shows a low-going pulse every 5.1 ms, so this is the rate at which keypresses are being checked for.

Program on disk:
P26.ASM Test Program for Lab.

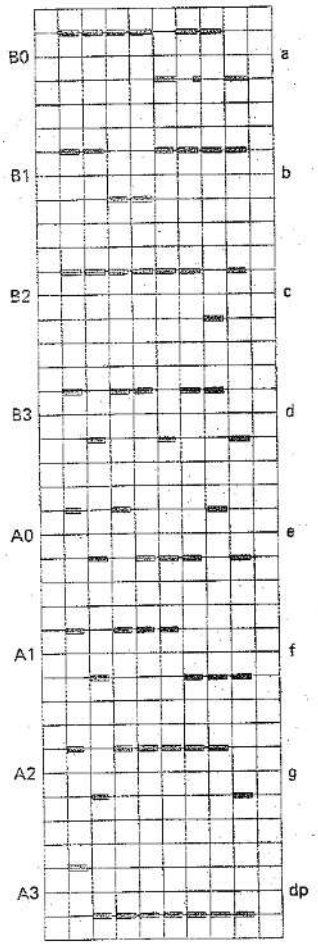


FIGURE 26-3.

Experiment 27 Stepper Motors

1. See program P27-1.ASM on the disk.

Programs on disk:

P27-1.ASM	Stepper Motor Driver (Delay Loop)
P27-2A.ASM	Stepper Motor (Interrupt Mainline)
P27-2B.ASM	Stepper Interrupt Service Procedure

Experiment 28 Using a D/A Converter with a Microcomputer

3. See program P28-1.ASM on the disk.
4. Voltage on op amp output (pin 2) = -5 V.
6. *a.* Resolution of 8-bit D/A converter = 1/256, or 0.39 percent.
   b. Resolution of D/A in volts = 10/256 = 39 mV.
   c. Full-scale output is always 1 LSB less than the named value of the converter = 10 - 1/256 - 5 = 4.961 V.
   d. For 0 V output 80H (128).
7. See program P28-2.ASM on the disk.
8. The waveform should be sawtooth ramping from -5 V to +4.96 V. The factors that limit the frequency of the waveform are the processor frequency, the number of instructions in the loop outputting the incrementing count, and the number of bits in the D/A.
9. Incrementing the count by 2 almost doubled the frequency.
10. Decrementing the count produces a reverse sawtooth ramp, from +5 V to -5 V.
11. See program P28-3.ASM on the disk.
13. See program P28-4.ASM on the disk.

Programs on disk:

P28-1.ASM	Output Zeros to D/A (Port P1C)
P28-2.ASM	Output Sawtooth Waveform
P28-3.ASM	Output Triangular Waveform
P28-4.ASM	Output Sine Wave

Experiment 29 Analog-to-Digital Conversion with a Microcomputer Counter-Type A/D Converter

2. See program P29-1.ASM on the disk
4. Comparator output is low if V from D/A > input V.
5. See program P29-2.ASM on the disk.
6. See program P29-2.ASM on the disk.
7. Expected values are 2 V - 33H.
8. The resolution is 10 V/ 256 = 0.039 V.
9. The time required for each conversion with V_{IN} = 10.0 V is 7.2 ms. The number of conversions per second is 1/7.2 ms, or 138.
10. With V_{IN} at 5 V, the conversion time is about 3.6 ms.

Successive-Approximation-Type A/D Converter

2. See program P29-3.ASM on the disk.
3. See program P29-3.ASM on the disk.
4. Expected values are 2 V - 33H.
5. The resolution is 10V/256 = 0.039 V.
7. Eight steps are required to complete a conversion. About 2000 conversions occur per second for 10 V input. See Figure 29-1.

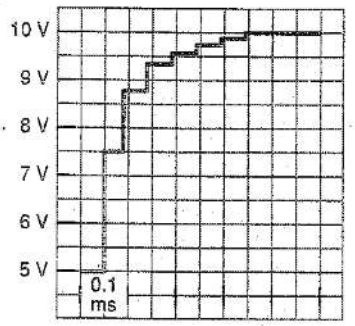


FIGURE 29-1.

8. See Figure 29-2. Eight steps are required to complete a conversion; same as with V_{IN} of 10 V, 2000 conversions per second.

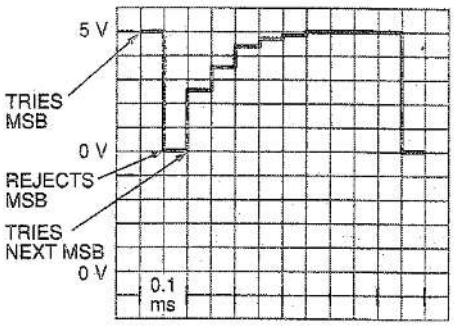


FIGURE 29-2.

9. Counter-type 10 V = 7.2 ms per conversion. Successive-approximation-type 10 V = 0.45 ms per conversion.
10. Advantage—doesn't require SAR; disadvantage—ties up the microprocessor.

Programs on disk:

P29-1.ASM	A/D Calibrate
P29-2.ASM	A/D Using Incrementing Count
P29-3.ASM	A/D Using Successive Approximation
DISPLY.ASM	SDK-86 Display Procedure

Experiment 30 8087 Programming

4. See program listing.
10. Volume = $4/3 \times \pi \times 2^3 = 33.510321$.
12. The major advantage of using an 8087 to do numerical computations in a program is that it increases the speed of the program's execution.

Program on disk:
P30-1.ASM Find Volume of Sphere

Experiment 31 Introduction to the TURBO C++ Integrated Development Environment

There are no questions in this experiment.

Experiment 32 C Programming Practice

Index Method

1. See program P32-1.C.
2. See program P32-1.C.
4. See program P32-1.C.

Pointer Method

1. See program P32-2.C.
2. See program P32-2.C.

Programs on disk:
P32-1.C Solution #1 for Experiment 32
P32-2.C Solution #2 for Experiment 32

Experiment 33 Displaying Characters on an IBM PC Screen

There are no questions in this experiment.

Programs on disk:
P33-1.ASM Fill Screen with Diamonds (Just for Fun)
P33-2.ASM Display Name in Center of Screen
P33-2.ASM Display Name Using BIOS Calls

Experiment 34 Assembly Language Color Graphics

3. Pixel data byte for four blue dots = 55H.
   Fill command = F0 200 55.
5. Debug command to point DS at odd scan line region of display RAM:
   RDS<CR>BA00<CR>.
   Fill command for magenta dots: F0 200 AA<CR>.
7. Fill command to draw single blue line: F1400 144F 55.

Programs on disk:
P34-1.ASM Draw Rectangle on Screen
P34-2.ASM Use Cursor Keys to Draw Lines
P34-3.ASM Changing Screen Colors
P34-4.ASM Expanding Colored Rectangle (Just for Fun)

Experiment 35 C Graphics Programming

There are no questions in this experiment.

Experiment 36 Vector Graphics with D/A Converters

6. Square = 0,0; 0,64H; 64H, 64H; 64H,0.

```
REPEAT
        Increment y count to 64H
        Increment x count to 64H
        Decrement y count to 0
        Decrement x count to 0
FOREVER
```

11. As the sides of the square increase, more counts have to be output from the D/A to produce that side.

Programs on disk:
P36-1.ASM Displays Square on Oscilloscope
P36-2.ASM Displays Enlarging Square

Experiment 37 A Text Editor—Using DOS Function Calls to Manipulate Disk Files

There are no questions in this experiment.

Programs on disk:
P37-1.ASM Write Text to a File
P37-2.ASM Open and Read a File
P37-3.ASM Open and Write File to Printer
P37-4.ASM Text Editor

Experiment 38 A Text Editor Using C Functions

There are no questions in this experiment.

Program on disk:
P38.C Simple Editor with Backspace Erase

Experiment 39 SDK-386 Programming

There are no questions in this experiment.

Programs on disk:

P39-1.ASM	Incrementing Count on SDK-386 Using Delay Loop
P39-2.ASM	Counting and Displaying the Number of Interrupts

Experiment 40 Putting It All Together—386 Programming on a PC

There are no questions in this experiment. You're on your own. Good luck!