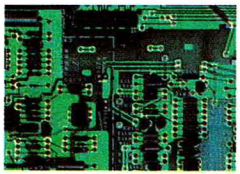
A BEGINNERS GUIDE TO COMPUTER TECHNOLOGY

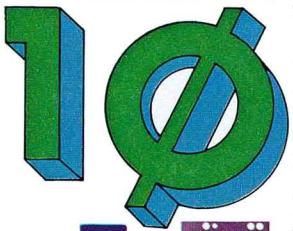
COMPUTER INTRO

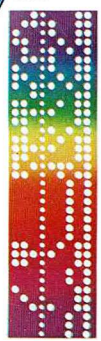
BY MAGNAVOX


ODYSSEY²

Contents







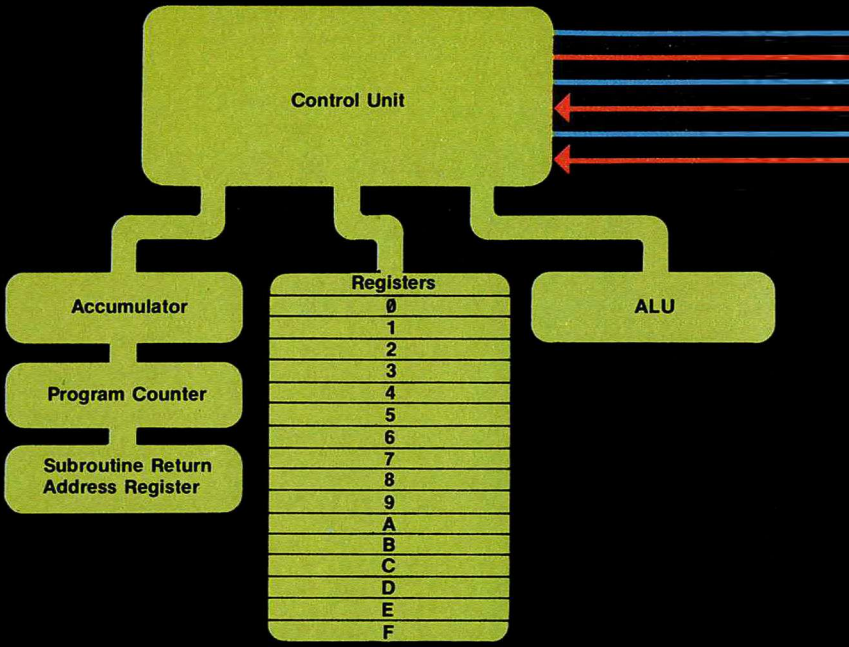
this gatefold will provide you with an electronic road map—please keep it open as you work with your Odyssey² computer

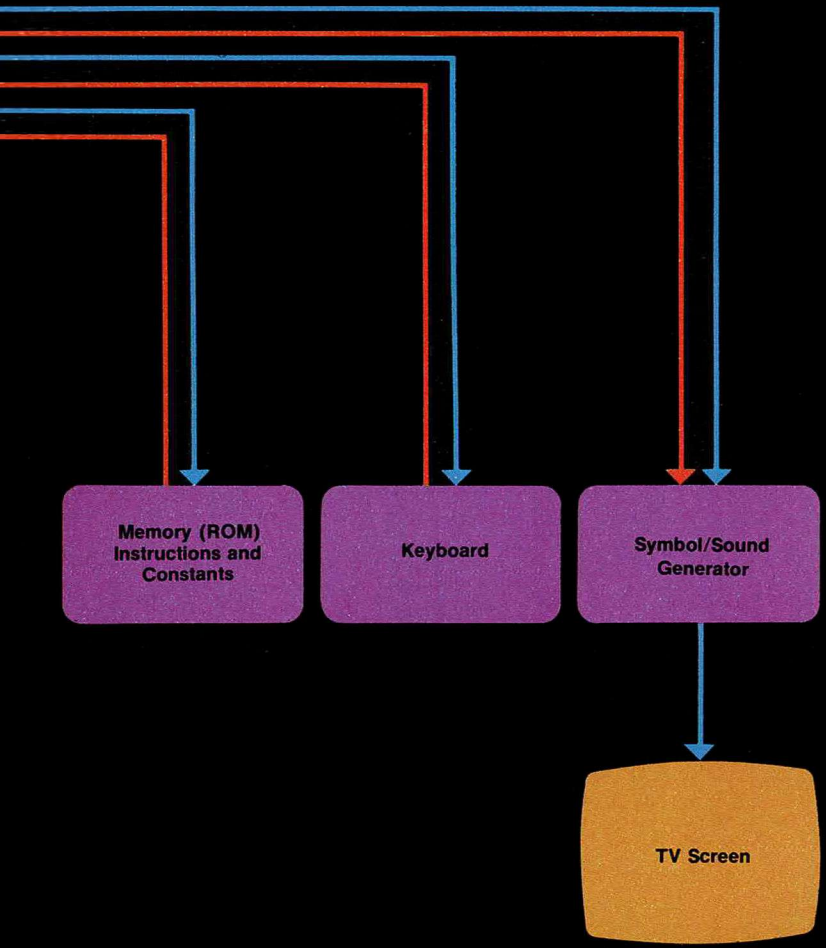
Key Codes

Key Code	Hex Code	Decimal Equivalents
0	00	00
1	01	01
2	02	02
3	03	03
4	04	04
5	05	05
6	06	06
7	07	07
8	08	08
9	09	09
A	20	32
B	25	37
C	23	35
D	1A	26
E	12	18
F	1B	27
G	1C	28
H	1D	29
I	16	22
J	1E	30
K	1F	31
L	0E	14
M	26	38
N	2D	45

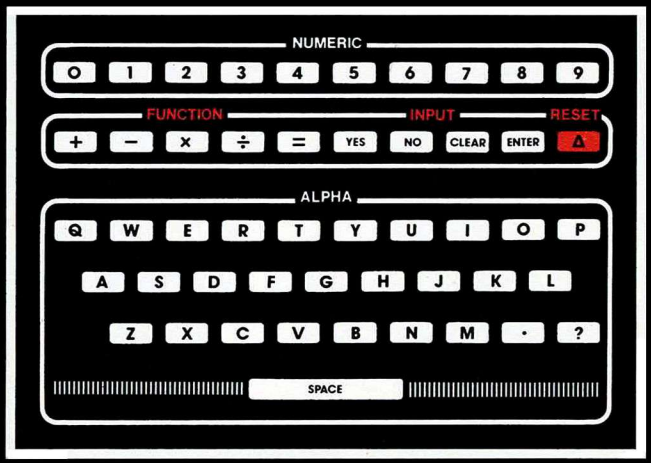
Key Code	Hex Code	Decimal Equivalents
O	17	23
P	0F	15
Q	18	24
R	13	19
S	19	25
T	14	20
U	15	21
V	24	36
W	11	17
X	22	34
Y	2C	44
Z	21	32
Blank	0C	12
:	0A	10
$	0B	11
Clear	2E	46
?	0D	13
•	27	39
+	10	16
-	28	40
X	29	41
÷	2A	42
=	2B	43
Enter	2F	47

Internal Flow




Memory (ROM)
Instructions and
Constants
Keyboard
Symbol/Sound
Generator
TV Screen

Keyboard


NUMERIC
0 1 2 3 4 5 6 7 8 9
FUNCTION
INPUT
RESET
+ − × ÷ = YES NO CLEAR ENTER Δ
ALPHA
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M . ?
SPACE

Computer Intro is not for everyone—but if you're up for a rewarding mental challenge, here is a fascinating entry point into a complex and highly technical subject.

The cartridge turns your Odyssey² into a very special kind of computer. It won't balance your checkbook or do your income tax or plot the course of a spaceship to Mars.

But it will give you some idea of how those computers do their work. You will begin to understand how a computer "thinks" and even begin to think like a computer "thinks."

The initial orientation and explanation are deliberately couched in the most simplistic of terms. They don't assume anything more on your part than a working knowledge of basic arithmetic.

After a brief explanation of how computers work and what they are made of—you will start getting "hands on" experience and will learn by doing.

You will learn how to enter a program—the first step in learning how to actually write your own.

The gatefolds in the front and back of the manual provide you with electronic road maps. Keep these references in front of you and you will clearly understand what's going on and how computers really compute.

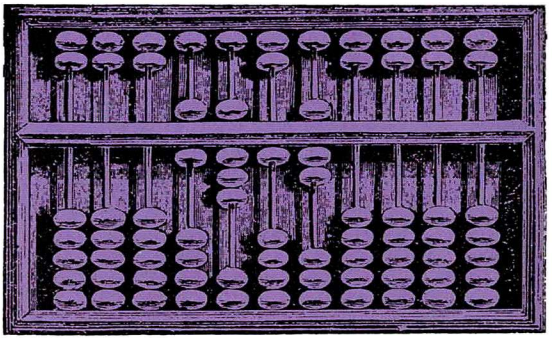
In The Beginning

In the beginning, there were ten fingers—then a prehistoric Einstein discovered his toes and man could count up to twenty.

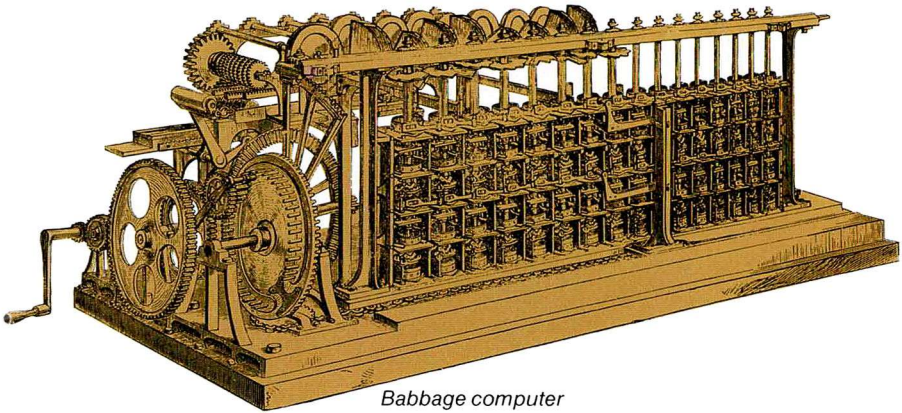
The oldest computing device we know of is the abacus. It was first used in China in the sixth century B.C.

The very first digital computer was designed by Charles Babbage in the 1830's. It was more than a calculator. Babbage designed it to be programmable, and it would have been able to perform any arithmetic or logic calculation.

It was designed to use punched cards for entering data and instructing the machine with mathematical commands.



Chinese Abacus



Babbage computer

There were two problems.

Problem one. Babbage's elaborate drawings called for a building the size of a Dickensian shoe factory to house his "analytical engine."

Problem two. Babbage died before the machine could be built.

The first *practical* programmable computer was built in a basement at Harvard University during World War II by IBM. It weighed 35 tons!

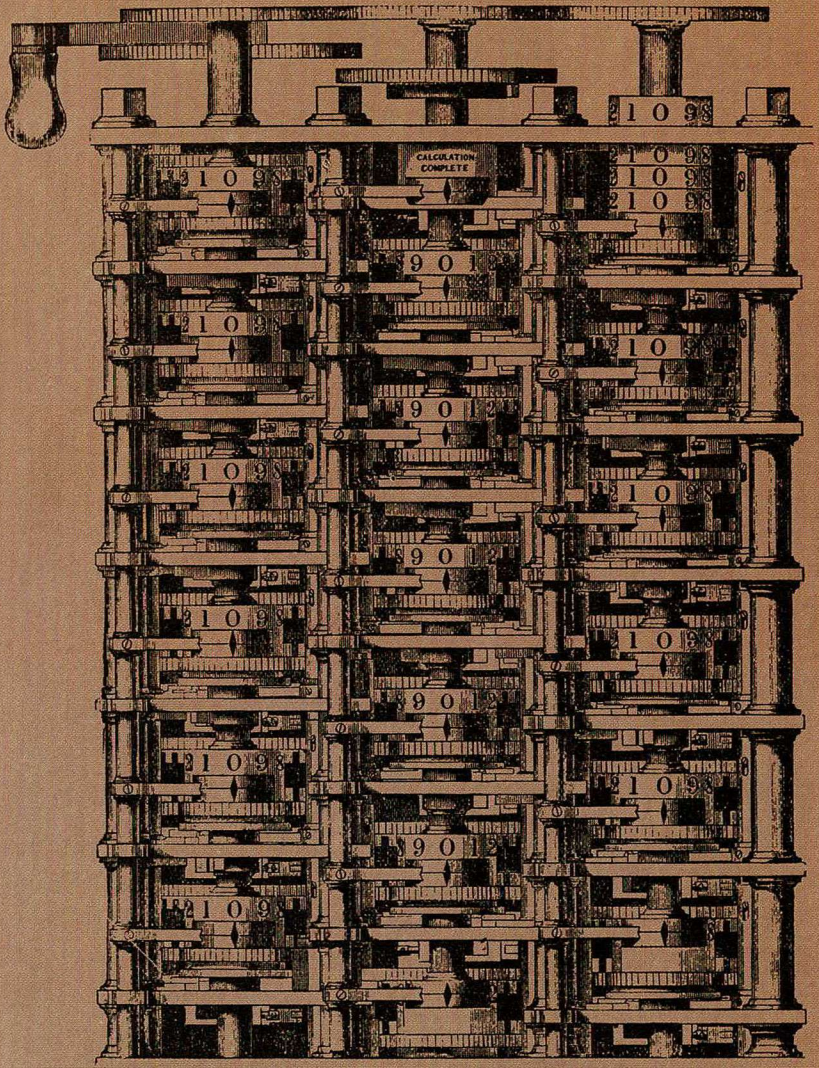
This machine used an exotic combination of electronic, electrical and mechanical gear to do its arithmetic. The instruction program was stored on a punched paper tape that unreeled automatically. Numbers were entered into the machine on a panel covered with 1,440 dials!

The first all-electronic computer was built at the University of Pennsylvania shortly after the war ended. ENIAC (Electronic Numerical Integrator and Computer) was 1,000 times faster than its predecessor.

But it filled a room 30 feet wide and 50 feet long. Its 18,000 vacuum tubes were connected by about a half-million soldering points.

Today, everything that ENIAC could do and far more is performed by a device slightly smaller in size.

It is less than one quarter of an inch square—and can make more than one million electronic decisions every second. It is called a semiconductor. It is possible for one semiconductor in one of its newest



Detail section of the Babbage computer

forms—the microprocessor—to account for more than 310 trillion separate functions.

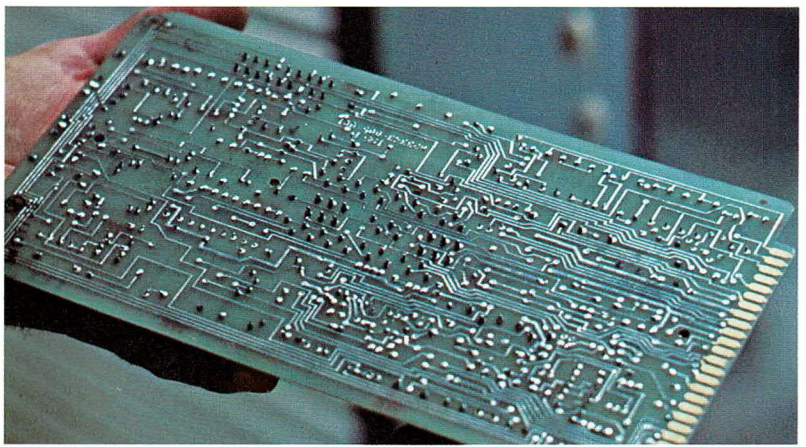
And this is only the beginning!

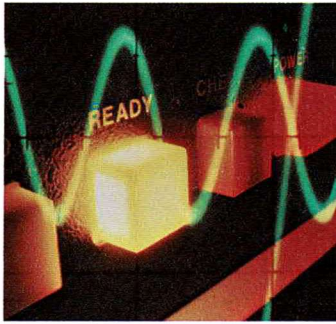
If computer technology continues to develop at its present rate, *one* of these chips will be able to store about a quarter-million bits (the smallest unit of computer information) in its memory within a few years.

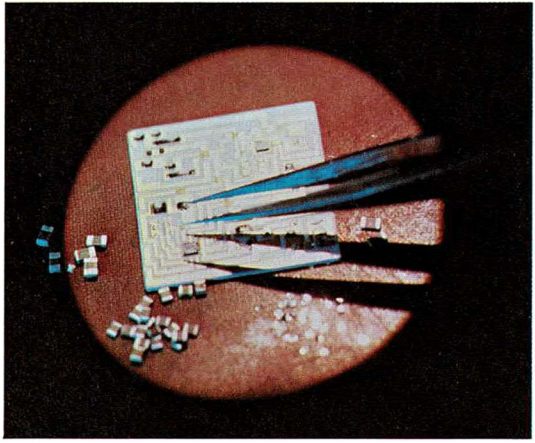
Ten years from this point, there will be chips capable of remembering a million bits of information.

By 1990, the number of logic or decision-making computer circuits on these one quarter inch chips are expected to number a quarter of a million.

Today's large computers, selling in the one million dollar range and as big as several filing cabinets, contain only about 10,000 logic circuits and a main memory capacity of a few million bits.




READY



As microprocessor technology progresses, computers may be developed that understand human speech.

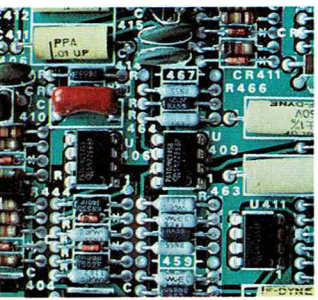
It is even possible they can be taught to read *handwriting!*

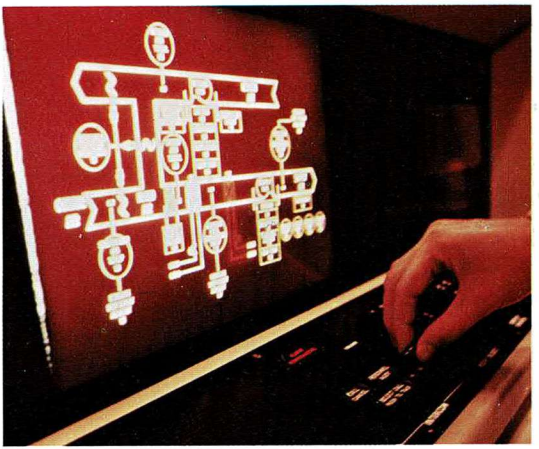
The microprocessor in your Odyssey² is infinitely more sophisticated than the mathematical marvels that were the state of the art in the forties and fifties.

The technology of the microprocessor is unquestionably going to revolutionize the way the world works—and the way you'll live. A computer will control your car's automatic transmission and fuel injection system.

A computer will monitor fire and burglar alarm systems in your home.

The lights in your home will be computerized. So will the locks on the doors and windows.





A computer will even water the lawn.

A computer will do your shopping from the house—and pay your bills without you writing checks.

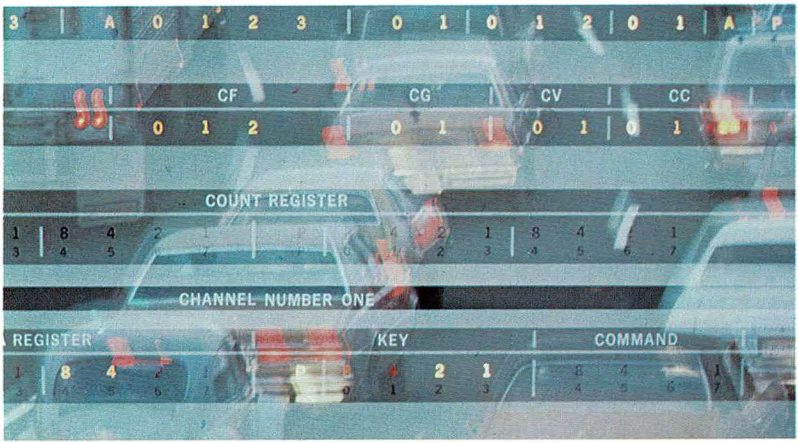
Computers will simulate three dimensional space for architects to help them mentally walk around their houses before they're built.

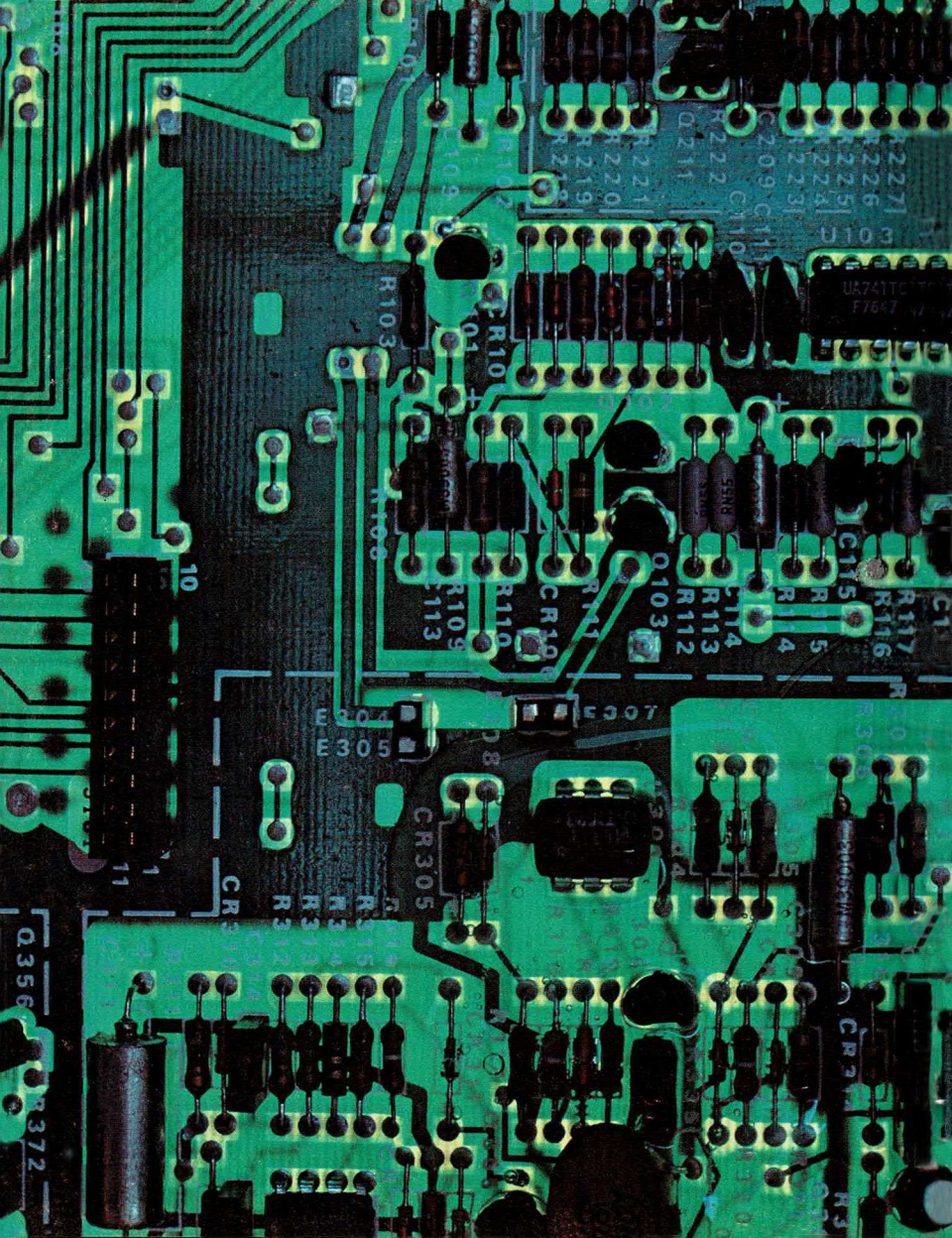
Computers will alert doctors to patient problems that would be imperceptible under today's circumstances.

Computers will help composers hear their music as they're writing it—even if it's too complicated for them to play.

Businessmen can have electronic simulations of their companies in their attaché cases.

We are really still just at the very beginning of the computer age. You have picked a very good time to get involved!




R103
CR101
R109
R113
R110
CR102
Q103
R114
R113
R112
C115
R116
R117
E305
E307
CR305
Q356
R372
U103

The World of the Computer Is Strange and Wondrous

Computers have already carried man to the moon—to Mars—and far beyond.

They lie at the heart of fearsome weapons systems.

They fly planes—monitor automobile engines—run factories—and even translate languages.

All of these with the brain power of a good screwdriver.

Congratulations! You have just completed your first lesson.

You are much smarter than any computer at the present state of the art!

For all of its awesome capabilities, the computer is nothing more than a rather simple-minded tool.

But once you learn how to use it, you'll have more power at your command than Julius Caesar ever dreamed of.

A computer has few basic talents.

It can add—and it can move numbers around.

A computer never forgets—and computers don't make mistakes. (If a computer churns out misinformation, there's a mistake in the program. Computers are utterly faithful in following instructions.)

Big deal. Computers can't multiply, divide, or even subtract the way you do. But what a computer does do, it accomplishes with absolutely astonishing speed.

Your Odyssey² can make over 100,000 electronic decisions every second—and there are computers around that are more than ten times faster than that!

To multiply, Odyssey² simply adds numbers together at an incredible speed. To subtract, it moves numbers around in a special way so that adding them together will give the correct answer. This sounds like doing it the long way—but when you're that fast at addition, the juggling act becomes worthwhile.

It's not all that hard to talk with a computer. It only understands two words. Yes—and no. Yes—means that an electrical pulse is tickling the computer's sensitivities. No—means that no electrical pulse is going through.

The symbol for "yes" in computer language is 1.

The symbol for "no" is Ø.

Once you have memorized Ø and 1, you have memorized the entire alphabet of the only language computers speak in any country of the world.

This is called a binary system because there are only two symbols involved.

It's sort of a code. Here's the key.

Binary Numbers and Their Decimal Equivalents.

1=0001	5=0101	9=1001	13=1101
2=0010	6=0110	10=1010	14=1110
3=0011	7=0111	11=1011	15=1111
4=0100	8=1000	12=1100	16=0001 0000

Letters of the Alphabet and Their Binary Code Equivalents.

Keyboard Letter	Binary	Decimal	Hexidecimal
A	0010 0000	32	20
B	0010 0101	37	25
C	0010 0011	35	23
D	0001 1010	26	1A
E	0001 0010	18	12
F	0001 1011	27	1B
G	0001 1100	28	1C
H	0001 1101	29	1D
I	0001 0110	22	16
J	0001 1110	30	1E
K	0001 1111	31	1F
L	0000 1110	14	0E
M	0010 0110	38	26
N	0010 1101	45	2D
O	0001 0111	23	17
P	0000 1111	15	0F
Q	0001 1000	24	18
R	0001 0011	19	13
S	0001 1001	25	19
T	0001 0100	20	14
U	0001 0101	21	15
V	0010 0100	36	24
W	0001 0001	17	11
X	0010 0010	34	22
Y	0010 1100	44	2C
Z	0010 0001	32	21

Working with endless daisy chains of 0's and 1's would be more than tedious for the human brain—but the computer is really good at it.

The inside of a computer is mainly a series of little electronic gates. Ø—the absence of an electrical pulse—leaves the gate open. 1—the presence of an electrical pulse—closes the gate. And remember—the electronic gates in your Odyssey² computer are opening and closing at the rate of 100,000 times every second.

You will enter programs in your Odyssey² through either the hexidecimal or assembler language (these will be explained later). The Odyssey² will then change the data and instruction sets entered into binary language (1's and Ø's) and store that information in the Memory and in the registers.

The actual computing is done by the computer's Central Processing Unit (CPU). The CPU in your Odyssey² is composed of the Accumulator (a working register which stores data temporarily); the Program Counter (a working register which locates and identifies the instruction sets and keeps them in order); the Registers (in which data, implemented by the programmer, is stored); the Sub-Routine Return Register (which is used with a certain instruction set); the Arithmetic Logic Unit (ALU); and the Control Unit.

The Control Unit directs the flood of electronic traffic traveling through the computer and controls the data flow between the different components of the computer. For example, it regulates the flow of information between the Memory and Arithmetic Logic sections and also orders processed

data to move from the Memory to the Output terminal. The Output terminal of your Odyssey² computer is the screen of your television set.

The Arithmetic Logic Unit is where the computer teaches numbers to tap dance. Its nickname is "number cruncher." It acts on the binary data fed into the computer's memory and registers—and then changes them according to the programmed instructions.

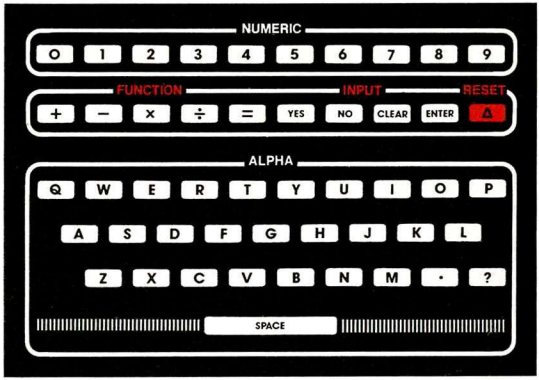
Now, you're ready to learn by doing. You're going to enter a program into your computer. Open the fold-out at the front of the book and you'll see what happens to everything along the way.

Important point. The neat thing about computers is that they will always follow your instructions with unflagging good faith. The dumb thing about computers is that they will only do what you have instructed them to do.

It's very important to make sure that you enter every step and do it right. If you make an error, the computer is going to make an error.

Be sure that the power to your Odyssey² console is turned off. Insert the COMPUTER INTRO cartridge into the console. Be sure the label side is facing the alphanumeric keyboard. Now, turn on the power.

You're going to be talking with your Odyssey² computer through the keyboard. It will talk back to you over your television screen.



Let's take a brief trip around the keyboard. It has forty-eight keys. Each key has been encoded in the computer languages we're going to be using. You'll find this code on the gatefold at the front of the book.

The keyboard also contains some surprises—four games that have been preprogrammed in the cartridge.

Press 2 on the alpha-numeric keyboard. A FLASH CARD addition game will appear on your television set. An unsolved addition problem flashes on your screen. You enter the solution through the keyboard. If the answer is less than 1Ø, preface the number with a Ø. (Important! Always use the numeral 0 at the top of the keyboard when entering a Ø.) If you give a wrong answer, an angry NO will appear on the screen. If you give a correct answer, it will appear in its proper position. To bring another problem to the screen, summon it from the computer by pressing any key.

Adding numbers is no big deal until you

see how many addition problems you can solve in one minute. If you keep trying to beat your own record or another player, you can't help but sharpen your skills.

Press RESET, then press 3 on the alpha-numeric keyboard and you're into COMPUTER TELEPATHY! It's a high-low game. The computer secretly chooses a number between ØØ and 99. The secret number won't appear on the screen—but a question mark will. You make a guess at the correct number and enter it into the computer. Your guess will appear on the screen. It will be followed by an H if it is higher than the computer's secret number—or an L if it is lower. If you guess correctly, the number will be followed by an X. Play against an opponent and see who can guess the computer's secret number in the fewest number of guesses.

Now, press RESET, then press 4 on the alpha-numeric keyboard. BETWEEN THE SHEETS appears on the screen. You'll see three sets of numbers. Example: Ø3 Ø7 ØØ.

The first two numbers are the sheets. The computer has thought of another number which it is keeping to itself. The last number is your score. If you think the computer's secret number is between Ø3 and Ø7, press YES on the alpha-numeric keyboard. If you don't think it falls between 3 and 7, press NO. If you are correct, you score a point and the number will appear between the 3 and 7. If you are wrong, the computer rewards you with a couple of BEEPS right in

the ear and your score remains the same.

Press RESET, then press 1 and a series of blocks appears on the screen. The blocks flash in random order and are accompanied by a buzzing sound. We call this spooky effect "THE CREEPY CRAWLER." You can call it anything you want.

We can re-program The Creepy Crawler to display other symbols on the screen. There is a large selection of graphics in the memory of your Odyssey^2. You'll find the complete collection in the fold-out at the back of the book.

The Creepy Crawler program is quite short and represents a good starting point. We're going to follow this and succeeding programs with a sort of road map so you can see where your input goes and what the various parts of the computer do with it.

There are two ways you can program your computer. You can speak HEXIDECIMAL code (machine language). An instruction would look like this: 6Ø.

Your computer takes 6Ø and converts it into its binary equivalent: Ø11Ø ØØØØ. Notice that this binary translation has eight digits. These are called BITS. Bits are handled by the computer in groups of eight. One group is called a BYTE. A byte is the smallest piece of information a computer can work with.

In Odyssey^2 Hexidecimal language, 6Ø means "Load a value into Register Ø." (A register is a place where a computer stores information. There are sixteen registers in your Odyssey^2. Each register has room for

one byte of information.)

The second computer language your Odyssey² understands is called ASSEMBLER. Assembler uses alpha-numeric symbols to input binary code instructions. An instruction in Assembler is more phonetic than Hex—but it is also longer.

Example: LDV.Ø.38 means—Load a value into Register Ø—and that value is 38. (Important: Be sure to enter the periods in the assembler instructions or the computer will not know what you're talking about.)

In the back of the book, you'll find a complete chart of Instruction Sets for your Odyssey².

INSTRUCTION SETS are the codes that tell your computer what you want it to do with the data you're going to give it.

Also Important: "Ø" is the equivalent of zero in the numeric section of your keyboard. "O" is the letter of the alphabet.

Now, open the gatefold at the back of the book. You'll find a flow chart that tells you how to get into the various operating modes of your Odyssey². These modes are for displaying data in the registers, entering and executing a program, rolling through a program to check data and much more.

Leave the front and back gatefolds open. You'll have your reference material right in front of you.

We are now ready to re-program the fearsome Creepy Crawler. First, we'll enter in Hex—and then we'll do a variation in Assembler.

Creepy Crawler

Press the POWER BUTTON OFF and ON to clear the computer.

Press RESET "Command" appears on your screen. Your computer is in the COMMAND mode and ready to accept instructions. It knows you want it to do something but doesn't yet know what.

Press P "Program" appears on the screen. "Aha," thinks the computer. "Somebody wants to enter a program! I wonder what language this person will speak!"

Press M "Hex Input" appears on the screen. The computer now knows you will be communicating in HEX. You're going to be using the Hexidecimal Operational Code (OP CODE).

Press I Step 00 appears on the screen and the computer is ready and waiting to accept data.

Press 60 This is Op code for Load Register 0. Register 0 is one of sixteen registers in your Odyssey² that make up part of its Random Access Memory (RAM). Each register is a small memory device that provides temporary storage for data and instructions which will eventually be needed by the ARITHMETIC LOGIC UNIT (ALU)—the place where all simple reasoning and arithmetic operations are performed. Look at the registers in the computer as sixteen in and out boxes on a desk that is shared by both you and the Arithmetic Logic Unit.

Press ENTER Program Step Ø1 appears on the screen. Register Ø has now been activated to accept your entry upon execution of the program. (The program steps you enter into the computer do not actually become functional until you press E [EXECUTE] at the completion of the program.)

Press 3A You have just selected an electronic figure from your computer's gallery of electronic art and symbols, which are stored in the symbol/sound generator. You'll find the entire gallery in the fold-out at the back of the book. We've selected the little man as an example, but you can actually use any of the figures or symbols for this program. They have been permanently stored in your computer.

There are two parts to the memory unit in your computer. The ROM (Read Only Memory) contains the instruction sets and constants to be used in programming. (The constants may be repetitive numbers needed for mathematical computation by the ALU—Arithmetic Logic Unit, or they may be letters, numbers, or symbols which you have entered into the Odyssey² in a program which will remain the same throughout the program. See the program "Message.") The other part of the computer's memory is called RAM (RANDOM ACCESS MEMORY). This memory component is like a blackboard. Programs, instruction sets and constants can be entered and later erased so that new data can be entered. You're writing this program on the RAM component of the microprocessor in your Odyssey.²

Press ENTER Program step Ø2 appears on the screen. The little man will be loaded into Register Ø.

Press 61 This is Op Code for LOAD REGISTER 1. It tells your computer you want to give input to that data storage unit.

Press ENTER Program step Ø3 appears on the screen—and the door to Register 1 will open to receive data.

Press ØC This is Op Code for a blank like the space between words in a sentence.

Press ENTER Program step Ø4 appears on your screen. The blank will be loaded into Register 1.

Press 6B This is Op Code for positioning. You are opening the door to Register B and telling it you want it to display the little man at a certain place on the screen upon execution of the program. You'll let it know where.

Press ENTER Program step Ø5 appears. The door to Register B will open.

Press ØØ This entry tells the computer you will want it to display the little man at the furthest left position on the screen.

Press ENTER Program step Ø6 appears on your screen. The positioning information will be loaded into Register B.

Register B is the register that positions symbols or characters on the screen. It has been given eleven postions. 00 is the furthest left. 0A is the furthest right. When Register B outputs on the screen, it automatically increments or advances by one. If we output a symbol in 00 (position one), the symbol will appear in the first position and Register B will then advance automatically to position 2 (01). If Register B outputs in the last position (0A), it automatically resets itself back to the first position (00) on the next step. Whatever data was in that first position will be replaced by the new input.

Press C0 This is the Op Code that tells the computer you are going to want to bring that little man to the screen. You have given him a way to get out of Register B.

Press ENTER Program Step 07 appears on the screen.

Press C1 This is Op Code for telling the computer you want to bring that blank in Register 1 to the screen.

Press ENTER Program step 08 appears on your screen to tell you everything is going along smoothly.

Press 05 It's sound effects time. This is an Op Code that tells the computer you will want to hear a one second buzz.

Press ENTER Program step 09 appears.

09

Press Ø8 This Op Code tells the computer you want it to come with an unlimited sequence of random numbers. The computers that encipher and decipher secret messages for governments do this everyday. In the old days, crypt keys remained constant and could be broken easily. Today, they change constantly and at random. Today, it takes one computer to break another's cipher.

Press ENTER The random number instructions are entered into your computer. Program step 1Ø appears on the screen.

10

Press BB This is Op Code for UNPACK Register B. This is the Register we use to position our little man on the screen. This unpacking instruction takes the random two digit number selected by the Accumulator and places one digit in Register B (we are unpacking Register B, thus the Op Code BB) and the other digit in the Register immediately following B, which is Register C. The digit in Register C is ignored. Another example of unpacking—If we had unpacked Register Ø, the Op Code would have been BØ, and one digit of the random number would have been placed in Register Ø and the second digit in Register 1.

To look at it another way, think of the Accumulator as a suitcase and the registers as a tall dresser with 16 drawers. You unpack two items from your suitcase and put the first away in a drawer. The second item will automatically go to the drawer just below it.

At the end of the Creepy Crawler program, a variation is written explaining how to use the second digit of the random number which was loaded into Register C, along with numerous figures to execute a random display of figures on the screen.

Through this unpacking instruction, the positioning Register (Register B) is loaded from the Accumulator with a random number. It is this instruction which will cause our little man to travel to unpredictable places on the screen. (The Accumulator is a small memory device in the CPU that provides temporary storage for the Arithmetic Logic Unit. It can store the result of an ALU operation or serve as an operation source [OPERAND] for the ALU.)

Press ENTER The unpacking instruction is entered and program step 11 appears on your screen.

Press 12 This Op Code tells your computer you're going to want it to return to a previously programmed step.

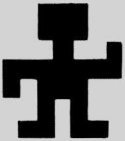
Press ENTER Program step 12 appears on the screen and the computer wonders which program steps you wish repeated.

Press 06 You have just told the computer you want it to go through the motions and always return to program step 06. This was the place you wanted the little man positioned on the screen

which was subsequently combined with a random number to change the positioning on the screen. In effect, you have now "looped" part of your program. It will do its thing endlessly with endless variations.

Press ENTER The computer salutes and will do as you ordered. Program step 13 appears.

Press RESET The program is stored and you are back in the COMMAND mode. "Command" appears on your screen.



Press E You have instructed the computer to execute your instructions. The fearsome Creepy Crawler appears on your screen. The little man or whatever symbol you have chosen will flash and buzz in different positions on the screen.

Forever.
Or, until you turn it off.
Or, change the program.
Whichever comes first.

Important! The power switch on your console is your program eraser. Turn it off and the program is cleared from the unit automatically. Turn it on—and you're ready to enter a new program.

Now that you've entered Creepy Crawler in HEX, try entering this variation in ASSEMBLER language. We will call this program Creepy Crawler with an All-Star Cast of Thousands.

First, turn the power off and on to erase any previous programming. Now, check the fold-out Operational Flow Chart to see how to get into the Assembler Input Mode.
Press RESET
Press P
Press A
Press I

You are now in the Assembler Mode at program step 00 and ready to go. One thing. Be sure to enter the periods as well as the letters and numbers.
Press L
Press D
Press V
Press .
Press 2
Press .
Press 1
Press 3
Press ENTER

We are now at Program step 02. Continue entering in Assembler until you get to Program step 13. At this step you will begin entering a variety of symbols in HEX. There is no Assembler equivalent for them—so after Program step 12, switch over to the HEX Input Mode. Check the Operational Flow Chart and then—
Press CLEAR
Press ROLL
Press CLEAR
Press M
Press I

After you finish entering the program, press RESET to store the program—then press E (EXECUTE) and watch what happens!

Creepy Crawler Version II

Step	Hex Code	Assembler Code	Byte	Remarks
00	62 13	LDV.2.13	2	Load Reg. 2 with 13
02	08	RND	1	Accum. selects a random number
03	BB	UNP.B	1	Unpack the random number into Reg. B and C
04	9C	LDA.C	1	Load the Accum. from Reg. C
05	E2	ADD.2	1	Add the contents of Reg. 2 to the Accum.
06	AC	STO.C	1	Store the contents of the Accum. in Reg. C
07	09	MOV	1	Load Accum. from a Program step
08	0B	OTA	1	Output from Accum. to screen
09	05	SIG	1	One second buzz
10	00	NOP	1	No operation (used as pause)
11	12 02	GTO.02	2	Instructs Odyssey² to go to Program step 02 and repeat program
13	32		1	Must enter Hex Mode (see page 27)
14	33		1	
15	3A		1	
16	34		1	
17	35		1	Hex codes for various symbols. These symbols with their Hex Code equivalents are shown at the front of the book.
18	37		1	
19	3D		1	
20	3E		1	
21	36		1	
22	3C		1	

The Roll Mode—Your Program Trouble Shooter

The Roll Mode is for checking a program step to be sure it contains the correct data. It is also for making a change in a program step without having to erase and re-enter an entire program.

To enter the Roll Mode, press R if you are in either the Assembler or HEX input modes. If you are in the EXECUTION mode, press RESET, then P, then M (HEX) or A (ASSEMBLER)—and then press R.

Then, press U to display the program steps upward (ØØ-99)—or press D to display the program steps downward (99-ØØ). The Roll Mode will always display its information in HEX—even if you have been entering in Assembler.

If everything checks out, roll to the last program step entered and press CLEAR to re-enter the Assembler or HEX input mode.

If you wish to make a change, press CLEAR at the program step you wish to change. The data will be cleared.

Enter M for HEX or A for Assembler, depending on the code you have been using.

Press I—the program step number you wish to change will appear on the screen.

Enter the new data.

Press RESET You will be back in the COMMAND Mode and are ready to go on with the program.

The following series of programs will be presented with a running commentary that will tell you exactly where the data is going and what is happening to it. These programs are written in HEX code. At the end of each program, you'll find a summary written in HEX as well as Assembler, so you can enter each program in either language.

Addition·Program A

This program will add two one digit numbers and display the total. Press the POWER BUTTON off and on to clear the computer.

Press RESET "Command" appears on your screen. Your computer is ready to accept instruction.

Press P "Program" appears on the screen.

Press M "Hex Input" appears on the screen. You have told the computer you will be using the Hexidecimal Operational Code (Op Code).

Press I Step 00 appears on the screen. The computer is ready to accept data.

Press 70 Press ENTER You have told the computer you are going to input the first number into Register 0.

Press 04 Press ENTER You instruct the computer to feed the second number of the addition problems into the Accumulator. Remember, the Accumulator provides temporary storage for the ALU where the numbers are going to be crunched.

Press E0 Press ENTER This instructs the computer to add the contents of Register 0 to the contents of the Accumulator and to store the sum in the Accumulator.

Press B1 Press ENTER This Op Code is an unpacking instruction. It tells the computer to unpack the sum which has been stored in the Accumulator into Register 1 and Register 2.

Press 6B Press ENTER This starts a positioning instruction.

Press 00 Press ENTER This Op Code tells the computer to display subsequent information at the 00 position on your TV screen.

Press C1 Press ENTER You tell the computer you will want to output the information in Register 1 (first digit sum).

Press C2 Press ENTER You also want to output the data in Register 2 (Second digit sum).

Press 12 Press ENTER This tells the computer you want it always to return to a certain step and repeat the program from that point.

Press 00 Press ENTER 00 is the step you want the computer to return to and repeat. Now, the computer will perform a continuous series of addition problems.

The program is now completed.

Press RESET The program is now stored in the computer's memory and you are back in the COMMAND Mode.

Press E The computer executes your program. A question mark appears on the screen. The computer is asking for two 1 digit numbers to add.

Enter the First Number. Press any digit from 0 through 9. You will hear a beep confirming entry. The number will not appear on the screen. That instruction was not included in this program.

Enter the Second Number. Press any digit 0 through 9 and the sum total of the two entered numbers will immediately appear on the screen.

To enter a new problem, press any single digit number. Then, press the other single digit number.

The previous sum will be replaced by the answer of the new addition problem as soon as you have entered the second number of the new problem.

Now, here is the addition problem you have just entered expressed in both HEX and ASSEMBLER codes. Note the following points.

Important: Remember to get into the proper input mode for the language you are using.
Press PMI for HEX.
Press PAI for Assembler.

Very Important: Look at Program step 04. It is a two Byte instruction. Remember, in Binary each byte is composed of eight bits. 6B (HEX = 0110 1011 (BINARY). 00 (HEX) = 0000 0000 (BINARY). Therefore, 6B 00 is a two byte instruction. If you are in the HEX mode, each Byte must be entered separately.
Press 6B
Press ENTER
Press 00
Press ENTER

If you are in the Assembler mode, both Bytes are fed into the computer with one entry.
Press L
Press D
Press V
Press .
Press B
Press .
Press 0
Press 0
Press ENTER

In either language, you will see Program step 06 on the screen after the data is entered. Now, push the **Power Button** to erase any previous data...choose one of the input codes...and enter the program. Be sure to press ENTER after each step.

Addition·Program A

Step	Hex Code	Assembler Code	Byte	Remarks
00	70	INP.0	1	Input Reg. 0 with 1st number
01	04	INA	1	Input Accum. with 2nd number
02	E0	ADD.0	1	Add Reg. 0 to Accum.
03	B1	UNP.1	1	Unpack Accum. into Reg. 1 and Reg. 2
04	6B 00	LDV.B.00	2	Set output position to 00
06	C1	OUT.1	1	Output Reg. 1, 1st digit sum
07	C2	OUT.2	1	Output Reg. 2, 2nd digit sum
08	12 00	GTO.00	2	Go to step 00 and repeat

Addition·Program B

This program will also add two one digit numbers. However, this time when you enter the second number, the entire problem will appear on the screen. (Example: 2+4=6). You will be entering + and = signs in this program.

First, press the POWER BUTTON off and on to clear the computer. Then—

Press RESET "Command" will appear on your screen, and your computer is all ears.

Press P "Program" appears on your screen.

Press M "Hex Input" appears on your screen. The computer knows you will be using Op Code (Hexidecimal Operation Code).

Press I Step 00 appears on the screen. You are ready to input data.

Press 70 Press ENTER You tell the computer the first number will be entered into Register 0. 01 appears on your screen.

Press Ø4 Press ENTER You tell the computer to accept the second number into the Accumulator. Ø2 appears on the screen.

Press 6 B Press ENTER This starts an output position entry.

Press ØØ Press ENTER The output position is set at ØØ at the far left of the screen.

Press CØ Press ENTER The output channel from Register Ø will open for the first number of the addition problem.

Press 63 Press ENTER The input channel to Register 3 will open.

Press 1Ø Press ENTER 1Ø will be loaded into Register 3. 1Ø is Op Code for the (+) sign.

Press C3 Press ENTER The output channel from Register 3 will open so the (+) sign can be displayed on the screen.

Press ØB Press ENTER The output channel of the Accumulator will open so that the second number can be displayed on the screen.

Press 63 Press ENTER The input channel to Register 3 will open.

Press 2B Press ENTER 2B will be loaded into Register 3. 2B is Op Code for the (=) sign.

Press C3 Press ENTER The output channel from Register 3 is opened so the (=) sign may be displayed on the screen.

Press E0 Press ENTER This instruction adds the contents of Register 0 to the contents of the Accumulator and stores the result in the Accumulator. Remember, Register 0 contained the first number of the addition problem. The Accumulator held the second number.

Press B1 Press ENTER This program step unpacks the contents of the Accumulator storing the first digit of the sum in Register 1 and the second digit of the sum in Register 2.

Press C1 Press ENTER The output channel from Register 1 will open to let the first digit sum be displayed on the screen.

Press C2 Press ENTER The output channel from Register 2 will open to let the second digit sum be displayed on the screen.

Press 12 Press ENTER This begins an instruction to return to a previous step.

Press 00 Press ENTER The computer is instructed to return to step 00 and be ready to

solve a new addition problem. The old problem will not erase from the screen until both digits of the new problem are entered.

Press RESET The program is now stored in the computer.

Press E The computer is ready to execute the program. A question mark appears on the screen asking for input.

Press any single digit number. Nothing appears on the screen.

Press a second single digit number. The entire problem and the solution appear on the screen. Both will remain there until two new digits are entered.

Addition·Program B

Step	Hex Code	Assembler Code	Byte	Remarks
00	70	INP.0	1	Input Reg. 0 with 1st number
01	04	INA	1	Input Accum. with 2nd number
02	6B 00	LDV.B.00	2	Set output position
04	C0	OUT.0	1	Output 1st number from Reg. 0
05	63 10	LDV.3.10	2	Load Reg. 3 with (+) sign
07	C3	OUT.3	1	Out Reg. 3, (/) + on screen
08	0B	OTA	1	Output 2nd number
09	63 2B	LDV.3.2B	2	Load Reg. 3 with (=) sign
11	C3	OUT.3	1	Output Reg. 3, (/) = on screen
12	E0	ADD.0	1	Add Reg. 0 to Accum.
13	B1	UNP.1	1	Unpack Accum. into Reg. 1 and Reg. 2
14	C1	OUT.1	1	Output Reg. 1, 1st digit sum
15	C2	OUT.2	1	Output Reg. 2, 2nd digit sum
16	12 00	GTO.00	2	Go to step 00 and repeat

Addition·Program C

After you enter this program a question mark on the screen asks you to press in two one digit numbers for the computer to add. The first number appears on the screen followed by (+) sign. When you press the second number, it appears on the screen followed by a (=) sign and the answer. When the first digit of the next addition problem is entered, the first problem will disappear from the screen. To begin, turn off the power to erase the previous program.

Press RESET You computer is in the "Command" Mode and ready to accept instructions.

Press P The computer enters the "Program" Mode.

Press M "Hex Input" appears on the screen. The computer is ready to accept instructions in Op Code.

Press I Step 00 appears on the screen and the computer is ready for the program.

Press 6B Press ENTER You are setting the output position of Register B.

Press 00 Press ENTER That output position is 00 at the far left of the screen.

Press 70 Press ENTER You tell the computer that the first number of the addition problem will be stored in Register 0.

Press C0 Press ENTER This tells the computer you will want it to output the contents of Register 0.

Press 63 Press ENTER You are preparing Register 3 to accept data.

Press 10 Press ENTER Register 3 will be loaded with a 10—the code for a (+) sign. You will find the complete code for all of the graphics stored in your computer on the fold-out at the back of the book.

Press C3 Press ENTER This will open the way to output the (+) sign from Register 3.

Press 04 Press ENTER This will open the Accumulator for future input.

Press 0B Press ENTER The output channel of the Accumulator is set to open.

Press 63 Press ENTER You want to load a value into Register 3.

10 Press 2B Press ENTER That value is 2B—the code for the (=) sign.

11 Press C3 Press ENTER This will open the way to output the (=) sign from Register 3.

12 Press E0 Press ENTER The computer will then add the contents of Register 0 to the Accumulator. Register 0 has already been instructed to accept the first number in the addition problem (Step 02)—and the Accumulator has been instructed to accept the second number.

13 Press B1 Press ENTER This Op Code will unpack the contents of the Accumulator (which contains the sum) into Register 1 and Register 2.

14 Press C1 Press ENTER This will open the output channel of Register 1.

15 Press C2 Press ENTER This will open the output channel of Register 2.

16 Press 70 Press ENTER This input is a pause operation. It tells the computer to leave the first problem on the screen until another problem is entered.

17 Press 6C Press ENTER Register C will be opened for loading.

Press ØB Press ENTER Register C will be loaded with ØB—the Op Code for the decimal number 11, which is exactly the number of positions available on the screen.

18

Press 67 Press ENTER Register 7 will be ready for loading.

19

Press ØC Press ENTER ØC is Op Code for blank spaces as indicated in the computer graphics section in the fold-out at the back of the book. Register 7 will be loaded with blank spaces. The computer will use these blank spaces to erase the old problem on the screen when the first digit of the new problem is entered.

20

Press 9C Press ENTER The Accumulator will be loaded with the contents of Register C—the decimal number 11.

21

Press 64 Press ENTER This will open Register 4 to accept data.

22

Press ØØ Press ENTER Register 4 will be loaded with ØØ.

23

Press Ø2 Press ENTER The amount in the Accumulator (11—the number of positions on the screen) will be decremented (subtracted) by one each time this step is reached.

24

Press C7 Press ENTER Register 1 will output its blank spaces.

25

26

Press 24 Press ENTER This is a branch instruction—sort of a fork in the electronic road. The computer is instructed to continue on to the next program step if the number in the Accumulator equals the number in Register 4 (ØØ).

27

Press 24 Press ENTER The computer is instructed to return to step 24 if the number in Register 4 is not equal to the Accumulator. Step 24 will decrement the contents of the Accumulator (which originally was 11) by 1 each time until the Accumulator is reset at ØØ to match Register 4. Program steps 16 through 27 demonstrate how a computer erases by outputting blank spaces to the screen.

28

Press 6B Press ENTER We are resetting Register B to return to its original output position once the Accumulator is equal to Register 4.

29

Press ØØ Press ENTER The output position of Register B is set at the extreme left position of the screen.

30

Press 12 Press ENTER This Op Code instructs the computer to branch to another program step when the amount in the Accumulator equals ØØ.

31

Press Ø3 Press ENTER The step the computer returns to is Ø3. Now, you are able to enter and solve repeated addition problems.

Press RESET The program is stored.

Press E The program is executed. See the beginging of the program for use instructions.

Addition-Program C

Step	Hex Code	Assembler Code	Byte	Remarks
00	6B 00	LDV.B.00	2	Set output position to 00
02	70	INP.0	1	Input 1st number to Reg. 0
03	C0	OUT.0	1	Output Reg. 0
04	63 10	LDV.3.10	2	Load Reg. with (+) sign
06	C3	OUT.3	1	Output (+) sign from Reg. 3
07	04	INA	1	Input to Accum. (second number)
08	0B	OTA	1	Output from Accum.
09	63 2B	LDV.3.2B	2	Load Reg. 3 with (=) sign
11	C3	OUT.3	1	Output (=) sign from Reg. 3
12	E0	ADD.0	1	Add Reg. 0 to Accum.
13	B1	UNP.1	1	Unpack Accum. to Reg. 1 and Reg. 2
14	C1	OUT.1	1	Output Reg. 1
15	C2	OUT.2	1	Output Reg. 2
16	70	INP.0	1	This is used as a pause operation
17	6C 0B	LDV.C.0B	2	Load Reg. C with Hex 0B (#11)
19	67 0C	LDV.7.0C	2	Load Reg. 7 with blank spaces
21	9C	LDA.C	1	Load Accum. from Reg. C
22	64 00	LDV.4.00	2	Load Reg. 4 with 00
24	02	DEC	1	Subtract 1 from Accum.
25	C7	OUT.7	1	Output Reg. 7 (blank spaces)
26	24 24	BNE.4.24	2	Branch if Accum. does not = Reg. 4
28	6B 00	LDV.B.00	2	Set output position to 00
30	12 03	GTO.03	2	Go to step 03 and repeat

One Digit Multiplication

After you enter this program, a question mark will appear on the screen asking for input. The first digit entered is the number you wish to multiply (the multiplicand). It appears on the screen with an (X) sign. The second digit entered is the multiplier. The complete problem will now appear on the screen along with the answer. Odyssey² solves multiplication problems through an addition process. If the problem is 3 X 7, the computer will arrive at the answer by adding 7+7+7. Program steps ØØ through 11 are instructions which allow the problem to be displayed on the screen. The mathematical operational sequence begins with step 12.

First, press the POWER BUTTON off and on to erase the previous program.

Press RESET You are in the "Command" Mode.

Press P You enter the "Program" Mode.

Press M "Hex Input" appears on the screen. The computer is ready for Op Code instructions.

Press I Step ØØ appears on the screen and the computer is ready for the program.

Press 6B Press ENTER The output position of Register B is ready for setting.

Press 00 Press ENTER You set the output of Register B for the far left of the screen.

Press 70 Press ENTER The multiplicand is directed to be stored in Register 0.

Press C0 Press ENTER Register 0 is given an output channel.

Press 66 Press ENTER The door will be opened to Register 6.

Press 29 Press ENTER An (X) sign will be entered into Register 6.

Press C6 Press ENTER Register 6 is given an output channel.

Press 71 Press ENTER The multiplier will be stored into Register 1.

Press C1 Press ENTER The multiplier is given a way out of Register 1.

Press 67 Press ENTER The door to Register 7 is instructed to open.

10 Press 2B Press ENTER 2B is Op Code for (=). The (=) sign will be loaded into Register 7.

11 Press C7 Press ENTER This will provide an output channel for Register 7.

12 Press 90 Press ENTER The Accumulator is instructed to be ready to accept data from Register 0. This is the Register that holds the number you want to multiply. The Accumulator will be loaded with the same value as the contents of Register 0—however, Register 0 will continue to retain its data.

13 Press E0 Press ENTER The computer is instructed to add the contents of the Accumulator to Register 0. Remember, that Odyssey² multiplies by a series of addition steps. This is the first addition step.

14 Press A2 Press ENTER The sum of the digits from the Accumulator and Register 0 will be stored in Register 2.

15 Press 91 Press ENTER The Accumulator will be loaded from Register 1 which holds the multiplier. The Accumulator will now know the number of addition steps it must perform to arrive at an answer.

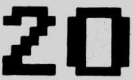
16 Press 02 Press ENTER This step will decrement the Accumulator which contains the multiplier by 1 so that the computer can keep track of how many addition steps have been made.

Press A1 Press ENTER The difference will be stored in Register 1.

Press 63 Press ENTER The door to Register 3 will open.

Press Ø1 Press ENTER Ø1 will be stored in Register 3. This will give the computer a reference point which will halt the addition process. When the Accumulator contains the contents of Register 1 (the multiplier), it is compared to the contents of Register 3 (which is Ø1). If the contents in Register 1, which is now in the Accumulator, and Register 3 coincide, the computer will stop adding. If they are not equal, the computer will loop back and continue the addition process.

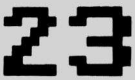
Press 33 Press ENTER This starts a branch operation.



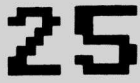
Press 25 Press ENTER The computer is instructed to branch to step 25 if the contents of the Accumulator and Register 3 are equal. In which case, the answer will be unpacked and displayed on the screen.

Press 92 Press ENTER The Accumulator will be loaded with the information in Register 2, which contains the data fed in Program Step 14.

Press 12 Press ENTER The computer is instructed to return to a previous step in the program if the conditions in program step 19 have not been met.



Press 13 Press ENTER The computer is instructed to return to step 13 and once again add the contents of Register Ø to the Accumulator. (Register Ø holds the multiplicand.)



Press 92 Press ENTER The Accumulator will be loaded with the contents of Register 2 which contains the sum of the addition operations. We are almost at the end of the tunnel. This operation is performed when the contents of the Accumulator and Register 3 are equal.

Press B4 Press ENTER This is a two digit unpacking operation, which has been explained in Addition Program A.

Press C4 Press ENTER An output channel for Register 4.

Press C5 Press ENTER An output channel for Register 5.

Press 12 Press ENTER You instruct the computer to return to a previous program step.

Press ØØ Press ENTER That step is ØØ. The computer will now be ready to accept a new multiplication problem.

Press RESET The program is stored.

Press E The program is executed.

One Digit Multiplication

Step	Hex Code	Assembler Code	Byte	Remarks
00	6B 00	LDV.B.00	2	Set output position
02	70	INP.0	1	Multiplicand stored in Reg. 0
03	C0	OUT.0	1	Output multiplicand
04	66 29	LDV.6.29	2	Symbol (×) stored in Reg. 6
06	C6	OUT.6	1	Output Reg. 6; Reg. 6 = (×)
07	71	INP.1	1	Multiplier stored in Reg. 1
08	C1	OUT.1	1	Output Multiplier
09	67 2B	LDV.7.2B	2	Symbol (=) stored in Reg. 7
11	C7	OUT.7	1	Output Reg. 7
12	90	LDA.0	1	Load Accum. from Reg. 0
13	E0	ADD.0	1	Add Accum. to Reg. 0
14	A2	STO.2	1	Store sum in Reg. 2
15	91	LDA.1	1	Load Accum. from Reg. 1
16	02	DEC	1	Decrement Accum. by 1
17	A1	STO.1	1	Store difference in Reg. 1
18	63 01	LDV.3.01	2	Load Reg. 3 with 01
20	33 25	BEQ.3.25	2	Go to step 25 if Accum. = Reg. 3
22	92	LDA.2	1	Load Accum. from Reg. 2
23	12 13	GTO.13	2	Go to step 13
25	92	LDA.2	1	Load Accum. from Reg. 2
26	B4	UNP.4	1	Unpack two digits
27	C4	OUT.4	1	Output Reg. 4
28	C5	OUT.5	1	Output Reg. 5
29	12 00	GTO.00	2	Go to step 00 and repeat

One Digit Division

A question mark will appear on the screen. The first number entered will be the dividend and it will be followed by a (÷) sign. The second number entered will be the divisor. It will appear on the screen along with an (=) sign and the answer. Your Odyssey² computer accomplishes division by a series of subtractions. It's the Odyssey² multiplication process in reverse.

Note: There are two conditions in division that must be provided. The first condition is when the divisor can be divided into the dividend equally. Example: 6 ÷ 2 = 3. The second condition is when the divisor cannot be divided into the dividend equally and there is a remainder. Example: 9 ÷ 2 = 4 + R. Your Odyssey² has been instructed to display (+R) if there is a remainder. This program provides branching instructions to satisfy both conditions.

To begin, turn the POWER BUTTON off and on to erase any previous program.

Press RESET You are in the "Command" mode.

Press P You have entered the "Program" mode.

Press M "Hex Input" appears on the screen. The computer is ready to accept instructions in Op Code.

Press I Step 00 appears on the screen, and you are ready to enter the program.

Press 63 Press ENTER The door to Register 3 will be opened.

Press 00 Press ENTER We call this "initialization"—a step to insure that Register 3 is set at absolute zero value after each problem is solved. Register 3 will contain the sum of the subtraction operations Odyssey² will perform to find the quotient.

Press 6B Press ENTER The door to Register B will be opened.

Press 00 Press ENTER Register B will be positioned to output at the far left of the screen.

Press 70 Press ENTER The number to be divided (the dividend) will go into Register 0 the dividend must always be larger than the divisor.

Press C0 Press ENTER The output channel to Register 0 is set to open.

Press 69 Press ENTER Register 9 will be opened for input.

Press 2A Press ENTER 2A is Op Code for (÷). The division sign will go into Register 9.

Press C9 Press ENTER The output channel of Register 9 is set to open.

Press 71 Press ENTER The divisor will be loaded into Register 1.

Press C1 Press ENTER The output channel of Register 1 is instructed to open.

Press 6A Press ENTER The door to Register A will be opened.

Press 2B Press ENTER 2B is Op Code for (=). The equal sign will go into Register A.

Press CA Press ENTER The Register A output channel is instructed to open.

Press 91 Press ENTER The Accumulator will be loaded with the contents of Register 1 which contains the divisor. We are going to use a sample problem so the explanation will be easier (6÷2=3). The Accumulator will be loaded with the divisor (2).

Press DØ Press ENTER The contents of the Accumulator will be subtracted from the contents of Register Ø which contains the dividend (6).

Press AØ Press ENTER The difference (4) between the dividend and the divisor (6−2=4) will be stored in Register Ø.

Press 93 Press ENTER The Accumulator will be loaded with the contents of Register 3. This is the Register that was "initialized" at 00 so that we could keep track of the number of times we subtracted.

17

Press 03 Press ENTER A 1 will then be added to the Accumulator. This is called an increment.

18

Press A3 Press ENTER The sum of the Accumulator is then stored in Register 3. This Register now contains 01.

19

Press 90 Press ENTER The Accumulator will be loaded with the contents of Register 0. Register 0 has been loaded with a value of 4, the difference between the dividend and the divisor. The value was loaded into Register 0 in Program step 16.

20

Press 13 Press ENTER This will be the start of a branch operation.

21

Press 40 Press ENTER The computer will branch to step 40 if the Accumulator equals 0. Once the Accumulator equals 0, the problem is finished and Program step 40 will unpack the answer and it will be displayed on the screen.

22

Press 91 Press ENTER If the Accumulator does not equal 0, it will go to this program step. The Accumulator will be loaded with the contents of Register 1 which contains the divisor (2).

23

Press 5Ø Press ENTER Another branch operation will be started.

Press 28 Press ENTER The computer will be instructed to branch to program step 28 if Register Ø which contains the dividend (4) is less than the Accumulator which contains the divisor (2) from Register 1.

Press 12 Press ENTER This tells the computer to return to a previous step if Register Ø is not less than the Accumulator.

Press 15 Press ENTER This completes the return instructions. The computer is programmed to return to step 15. This step begins the subtraction operations which will continue until Register Ø is loaded into the Accumulator at step 21 and, the Accumulator equals Ø in this example. When this condition is met, Odyssey² will loop to Program step 4Ø.

Press 93 Press ENTER This is the step your computer will branch to if a remainder is included. The branching instructions were given in steps 24 and 25. They programmed the computer to branch to step 28 if Register Ø which contains the dividend is less than the Accumulator which at this point, contains the divisor. We will use 9÷2=4+R as our sample problem. All program steps have been the same up to this point. The computer has looped to step 15 several times. Register Ø now contains Ø1 and the Accumulator contains Ø2. This step will load the Accumulator from Register 3 which contains the number of times (2) has been subtracted from (9) which is (4).

Press B4 Press ENTER This instruction will unpack the answer.

Press C4 Press ENTER The first digit, stored in Register 4, will be displayed on the screen.

Press C5 Press ENTER The second digit, stored in Register 5, will be displayed on the screen.

Press 66 Press ENTER Register 6 will open.

Press 10 Press ENTER 10 will be loaded into Register 6. 10 is the Op Code for (+).

Press 67 Press ENTER Register 7 will be opened.

Press 13 Press ENTER 13 will be loaded into Register 7. 13 is the Op Code for (R).

Press C6 Press ENTER The output channel for the (+) sign will be opened.

Press C7 Press ENTER The output channel for (R) will open.

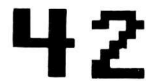
Press 12 Press ENTER The computer is instructed to return to a previous step.

Press 00 Press ENTER The computer will return to step 00 and be ready to solve a new problem.

Our program ends at this point if we have solved a problem that contains a remainder. If not, as in our first example (6÷2=3), we would have jumped from Program step 21 to Program step 40.

Press 93 Press ENTER The Accumulator will be loaded with the contents of Register 3. It will contain the number of times the divisor has been subtracted from the dividend. In step 21 the computer was programmed to branch to this step when the contents of the Accumulator equaled 0. We were using 6÷2 as our example so at this point Register 3 contains (3) - - - the number of times (2) has been subtracted from (6).

Press B4 Press ENTER This unpacking operation will convert the answer from binary into decimal.



Press C4 Press ENTER The output channel of Register 4 will be opened for the first digit.

Press C5 Press ENTER The output channel of Register 5 will be opened for the second digit.

Press 66 Press ENTER Register 6 will open.

Press 0C Press ENTER Register 6 will be loaded with 0C which is Op Code for a blank space.

Press 67 Press ENTER Register 7 will open.

Press ØC Press ENTER Register 7 will be loaded with a blank space. Registers 6 and 7 have been loaded with blanks so that (+R) will not be displayed on the screen when this branch of the program is employed by the computer.

47

Press C6 Press ENTER The output channel from Register 6 will open.

48

Press C7 Press ENTER The output channel from Register 7 will open.

49

Press 12 Press ENTER The computer is instructed to return to a previous step.

50

Press ØØ Press ENTER The computer is instructed to return to step ØØ and be ready to solve another problem.

51

Press RESET The program is stored.

Press E The program is executed.

One Digit Division

Step	Hex Code	Assembler Code	Byte	Remarks
00	63 00	LDV.3.00	2	Reg. 3 = 00 (initialization)
02	6B 00	LDV.B.00	2	Reg. B =00 (positioning)
04	70	INP.0	1	Dividend stored in Reg. 0
05	C0	OUT.0	1	Output Reg. 0
06	69 2A	LDV.9.2A	2	Symbol (÷) stored in Reg. 9
08	C9	OUT.9	1	Output Reg. 9
09	71	INP.1	1	Divisor stored in Reg. 1
10	C1	OUT.1	1	Output Reg. 1
11	6A 2B	LDV.A.2B	2	Symbol (=) stored in Reg. A
13	CA	OUT.A	1	Output Reg. A
14	91	LDA.1	1	Load Accum. from Reg. 1
15	D0	SUB.0	1	Sub. Accum. from Reg. 0
16	A0	STO.0	1	Store difference in Reg. 0
17	93	LDA.3	1	Load Accum. from Reg. 3
18	03	INC	1	Add 1 to the Accum.
19	A3	STO.3	1	Store sum in Reg. 3
20	90	LDA.0	1	Load Accum. from Reg. 0
21	13 40	BRZ.40	2	Branch to step 40 if Accum. equals 0
23	91	LDA.1	1	Load Accum. from Reg. 1
24	50 28	BLS.0.28	2	Branch to step 28 if Reg. 0 is less than Accum.
26	12 15	GTO.15	2	Go to step 15
28	93	LDA.3	1	Load Accum. from Reg. 3
29	B4	UNP.4	1	Unpack two digits
30	C4	OUT.4	1	Output 1st digit which is stored in Reg. 4
31	C5	OUT.5	1	Output 2nd digit which is stored in Reg. 5
32	66 10	LDV.6.10	2	Symbol (+) stored in Reg. 6
34	67 13	LDV.7.13	2	Symbol (R) stored in Reg. 7

Step	Hex Code	Assembler Code	Byte	Remarks
36	C6	OUT.6	1	Output (+) sign
37	C7	OUT.7	1	Output (R)
38	12 00	GTO.00	2	Go to step 00
40	93	LDA.3	1	Load Accum. from Reg. 3
41	B4	UNP.4	1	Unpack two digits
42	C4	OUT.4	1	Output 1st digit from Reg. 4
43	C5	OUT.5	1	Output 2nd digit from Reg. 5
44	66 0C	LDV.6.0C	2	A blank is stored in Reg. 6
46	67 0C	LDV.7.0C	2	A blank is stored in Reg. 7
48	C6	OUT.6	1	Output blank
49	C7	OUT.7	1	Output blank
50	12 00	GTO.00	2	Branch to step 00

Area Problems Using "Go to Subroutine" and "Return"

This program gives you an example of how and when to use the instructions "Go to Subroutine" and "Return."

A "Go to Subroutine" instruction tells the computer to branch to a specific program step which contains an operation you may wish to use several times in one program. You can use the same operation several times without having to rewrite it. When writing a program using this instruction, the next program step after the "Go to Subroutine" instruction must be reserved for returning from the Subroutine.

A program having a "Go to Subroutine" instruction must have a "Return from Subroutine" instruction as well.

After you enter this program, you will be able to find the area of a rectangle or the area of a triangle. First, enter the base measurement—then, enter the height measurement. Press 1 to find the area of a rectangle. (Base X Height = Area). Press 2 to find the area of a triangle. $\frac{\text{(Base X Height}}{2}$ **= Area)**

Before entering this program, refer to the program on pages 69 and 70 while we explain how it works.

Program steps ØØ through Ø8 will be the same for both problems.

(ØØ through Ø8) Step Ø8 programs your selection of 1) rectangle area problem, or 2) triangle area problem to be loaded into the Accumulator. First, we'll see what happens when 1 is loaded into the Accumulator.

(Ø9 and 1Ø) Steps Ø9 and 1Ø instruct the computer to branch to Program step 13 if the Accumulator (which contains Ø1) equals the contents of Register 1 (which is Ø1). If we had entered a 2, the computer would have proceeded from Program step Ø8 to Program steps 11 and 12.

(13 and 14) Program steps 13 and 14 instruct the computer to go to step 66 which starts the multiplication subroutine.(Base X Height = Area of Rectangle)

(66) Program step 66 loads the Accumulator with the contents of Register 3 which contains the base (Step Ø6). We'll use 8 as our example of the base measurement.

(67) Program step 67 adds the contents of Register 3 (which contains the base measurement of 8 in our example) to the contents of the Accumulator which now also contains 8.

(68) The contents of the Accumulator (8+8=16) are stored in Register 5.

(69) The Accumulator is loaded with the contents of Register 4 which contains the height. We'll use 3 as our example. This data was entered into Register 4 in Program step Ø7.

(7Ø) In Program step 7Ø, the Accumulator is decremented by 1. (Remember, in multiplying, the multiplier is decremented by 1 with each addition operation until the multiplier equals Ø1).

(71) The contents of the Accumulator (now 2) are loaded into Register 4.

(72 through 73) These instructions tell the computer to branch to Program step 77 if the contents of the Accumulator are equal to the contents of Register 1. (Register 1=Ø1). This was accomplished in Program step Ø2.

(74) If the Accumulator does not equal Ø1 at Program step 72, the Accumulator will move to Program step 75 and be loaded from Register 6 which contains 16.

(75 and 76) Go to step 67. The addition process is repeated until the amount in the Accumulator is equal to the value of Register 1. When this condition is achieved, the computer will branch to Program step 77 as instructed by Program steps 72 and 73.

(77) The Accumulator is loaded from Register 5. (Register 5 now equals 24 which is the answer.)

(78) Program step 78 instructs the computer to return from subroutine.

This returns the computer to program step 15 which unpacks the contents of Register 5.

Program steps 16 and 17 output the answer to the screen.

Program step 18 tells the computer to go to the blanking operation at program step 58.

Program steps 58 through 63 output blank spaces that erase the old problem and make room for a new one.

Program step 64 tells the computer to return to step 00 and to get ready to solve a new problem.

Now, we'll see how this program computes the area of a triangle $\frac{\text{Base X Height}}{2}$ = Area of Triangle) For our example, we'll use 6 as the base and 2 as the height.

This time we will choose 2 (triangle routine) at Program step 08 to load into the Accumulator. Since Program steps 09 and 10 do not apply, the computer will jump to Program step 11.

Program step 11 instructs Odyssey² to branch to program step 20 for the triangle routine.

Program step 20 instructs Odyssey² to branch to step 66 for the multiply routine.

Program steps 66 through 75 perform the addition operations and continue to loop until the Accumulator equals Register 1 (01). Then the Odyssey² branches to step 77.

Program step 77 loads the Accumulator with Register 5, which holds the answer for B X H or 6 X 2 = 12. We must now divide this answer by 2 to find the area of a triangle.

Program step 78 instructs Odyssey² to return to the program step immediately following the subroutine from which it branched originally.

Program step 22—a pause is implemented.

Program step 23 stores Accumulator (which contains 12) in Register 3. This now becomes the dividend.

Program steps 24 and 25 load Register 4 with 02; this becomes the divisor.

Program steps 26 and 27 load Register 7 with 00. This is the initialization operation, since this Register will hold the sum of the subtraction operations. ("Initialization" was first introduced at the beginning of the One Digit Division Program.)

Program step 28 loads Accumulator from Register 4 (which contains the divisor, 2).

Program step 29 subtracts Accumulator from Register 3 (which contains the dividend, 12).

Program step 30 stores the difference (12 - 2 = 10) in Register 3; Register 3 = 10.

Program step 31 loads Accumulator from Register 7; Register 7 = 00.

Program step 32 adds one to the Accumulator. Remember, this is done to keep track of the number of times we subtract the divisor from the dividend.

Program step 33 stores sum in Register 7; Register 7 = 01.

Program step 34 loads Accumulator from Register 3; Register 3 = 10, dividend.

Program steps 35 and 36 order a branch to step 54 if Accumulator equals 00.

Program step 37 loads Accumulator from Register 4; Register 4 = 2, divisor.

Program steps 38 and 39 branch to step 42 if Register 3 is less than the Accumulator.

Program step 40. If the Odyssey² has not branched at this point to another step number, this instruction loops the Odyssey² back to Program step 29, so that additional subtraction operations can be performed.

At Program step 35, the computer, after completing the subtraction operations so that the Accumulator and Register 3 (the dividend) equal 00, branches to step 54.* At Program step 54, the Accumulator is loaded from Register 7 (which contains the number of times we subtracted the answer). Program step 55 then unpacks this answer and Program steps 56 and 57 output the answer to the screen. Blanks are outputed, since in this example there is no remainder, and at step 64, the Odyssey² is instructed to return to Program step 00 in preparation for a new problem.

Now, you're ready to enter the program into your Odyssey.² Be sure to turn the power off and on to erase any previous data.

*Note: If there had been a remainder, the computer would have branched at Program step 38 to step 42 and when the answer was unpacked and displayed on the screen, a (+R) would also have been displayed.

"Go to Subroutine" and "Return"

Step	Hex Code	Assembler Code	Byte	Remarks
00	6B 00	LDV.B.00	2	Reg. B = 00 (Positioning)
02	61 01	LDV.1.01	2	Area of rectangle—select (1)
04	62 02	LDV.2.02	2	Area of triangle—select (2)
06	73	INP.3	1	Input value (base) to Reg. 3
07	74	INP.4	1	Input value (height) to Reg. 4
08	04	INA	1	Select 1 or 2
09	31 13	BEQ.1.13	2	Go to rectangle routine
11	32 20	BEQ.2.20	2	Go to triangle routine
13	14 66	GTS.66	2	Go to multiply subroutine
15	B5	UNP.5	1	Unpack Reg. 5 and 6
16	C5	OUT.5	1	Output 1st digit
17	C6	OUT.6	1	Output 2nd digit
18	12 58	GTO.58	2	Go to blanking operation
20	14 66	GTS.66	2	Go to multiply subroutine
22	00	NOP	1	No operation (pause)
23	A3	STO.3	1	Store Accum. in Reg. 3
24	64 02	LDV.4.02	2	Load Reg. 4 with 02
26	67 00	LDV.7.00	2	Load Reg. 7 with 00
28	94	LDA.4	1	Load Accum. from Reg. 4
29	D3	SUB.3	1	Subtract Accum. from Reg. 3
30	A3	STO.3	1	Store difference in Reg. 3
31	97	LDA.7	1	Load Accum. from Reg. 7
32	03	INC	1	Add one to Accum.
33	A7	STO.7	1	Store sum in Reg. 7
34	93	LDA.3	1	Load Accum. from Reg. 3
35	13 54	BRZ.54	2	Branch to step 54 if A = 0
37	94	LDA.4	1	Load Accum. from Reg. 4
38	53 42	BLS.3.42	2	Branch to step 42 if Reg. 3 is less than Accum.
40	12 29	GTO.29	2	Go to step 29

Step	Hex Code	Assembler Code	Byte	Remarks
42	97	LDA.7	1	Load Accum. from Reg. 7
43	B8	UNP.8	1	Unpack Reg. 8 and Reg. 9
44	C8	OUT.8	1	Output 1st digit
45	C9	OUT.9	1	Output 2nd digit
46	6E 10	LDV.E.10	2	Load Reg. E with Symbol (+)
48	6F 13	LDV.F.13	2	Load Reg. F with Symbol (R)
50	CE	OUT.E	1	Output (+)
51	CF	OUT.F	1	Output (R)
52	12 58	GTO.58	2	Go to step 58
54	97	LDA.7	1	Load Accum. from Reg. 7
55	B8	UNP.8	1	Unpack Reg. 8 and Reg. 9
56	C8	OUT.8	1	Output 1st digit
57	C9	OUT.9	1	Output 2nd digit
58	6E 0C	LDV.E.0C	2	Load Reg. E with a blank
60	6F 0C	LDV.F.0C	2	Load Reg. F with a blank
62	CE	OUT.E	1	Output blank
63	CF	OUT.F	1	Output blank
64	12 00	GTO.00	2	Go to step 00
66	93	LDA.3	1	Load Accum. from Reg. 3
67	E3	ADD.3	1	Add Reg. 3 to Accum.
68	A5	STO.5	1	Store Accum. in Reg. 5
69	94	LDA.4	1	Load Accum. from Reg. 4
70	02	DEC	1	Decrement Accum. by 1
71	A4	STO.4	1	Store Accum. in Reg. 4
72	31 77	BEQ.1.77	2	If Accum. = Reg. 1, branch to step 77
74	95	LDA.5	1	Load Accum. from Reg. 5
75	12 67	GTO.67	2	Go to step 67
77	95	LDA.5	1	Load Accum. from Reg. 5
78	07	RET	1	Return to subroutine

One Digit Addition Flash Card

When you enter this program, a Flash Card addition game will appear on your television set. An unsolved addition problem flashes on the screen. You enter the solution through the keyboard. If the answer is less than 10, preface the number with a 0.

This program is different from the Flash Card game which is already programmed in the computer. In this program, the old problem is erased automatically and a new problem is displayed on the screen. There is also a rather interesting reward for entering the correct answer. Program steps 45 through 61 input a computerized musical comedy production number. Program steps 70 through 78 are the rests (pauses) in the melody.

Program step 26 is a packing operation. The data from Register 3 and Register 4 is combined and loaded into the Accumulator.

Program steps 62 through 69 reset Register B to 00 so that it's ready for the next problem.

Program steps 62 through 69 reset Register 3 to 00 so that it's ready for the next problem.

One Digit Addition Flash Card

Step	Hex Code	Assembler Code	Byte	Remarks
00	6A 0C	LDV.A.0C	2	Load a blank into Reg. A
02	68 10	LDV.8.10	2	Load a (+) sign into Reg. 8
04	69 2B	LDV.9.2B	2	Load an (=) sign into Reg. 9
06	6C 2D	LDV.C.2D	2	Load (N) into Reg. C
08	6D 17	LDV.D.17	2	Load (O) into Reg. D
10	08	RND	1	Load Accum. with random number
11	B0	UNP.0	1	Separate digits
12	6B 00	LDV.B.00	2	Set output position
14	C0	OUT.0	1	Output first digit
15	C8	OUT.8	1	Output (+) sign
16	C1	OUT.1	1	Output second digit
17	00	NOP	1	A no operation instruction must follow every third output instruction in a row
18	C9	OUT.9	1	Output (=) sign
19	90	LDA.0	1	Load Accum. from Reg. 0
20	E1	ADD.1	1	Add Reg. to the Accum.
21	A2	STO.2	1	Store sum in Reg. 2
22	73	INP.3	1	Input first digit guess
23	C3	OUT.3	1	Output first digit guess
24	74	INP.4	1	Input second digit guess
25	C4	OUT.4	1	Output second digit guess
26	83	PAK.3	1	Combine digits. This is a packing operation.
27	CA	OUT.A	1	Output blank
28	32 45	BEQ.2.45	2	If correct guess...buzz
30	CC	OUT.C	1	Output (N)
31	CD	OUT.D	1	Output (O)
32	6B 04	LDV.B.04	2	Set output position to 04
34	73	INP.3	1	Input first number of second guess
35	C3	OUT.3	1	Output first number
36	CA	OUT.A	1	Output blank

Step	Hex Code	Assembler Code	Byte	Remarks
37	CA	OUT.A	1	Output blank
38	00	NOP	1	No operation
39	CA	OUT.A	1	Output blank
40	CA	OUT.A	1	Output blank
41	6B 05	LDV.B.05	2	Set output position to 05
43	12 24	GTO.24	2	Go to step 24
45	05	SIG	1	Buzz
46	14 70	GTS.70	2	No sound
48	05	SIG	1	Buzz
49	05	SIG	1	Buzz
50	05	SIG	1	Buzz
51	14 70	GTS.70	2	No sound
53	05	SIG	1	Buzz
54	14 70	GTS.70	2	No sound
56	14 70	GTS.70	2	No sound
58	05	SIG	1	Buzz
59	14 70	GTS.70	2	No sound
61	05	SIG	1	Buzz
62	6B 00	LDV.B.00	2	Set position to 00
64	CA	OUT.A	1	Output blank
65	9B	LDA.B	1	Load Accum. from Reg. B
66	13 10	BRZ.10	2	If Accum. = 0, the computer branches to step 10
68	12 64	GTO.64	2	Go to step 64
70	67 00	LDV.7.00	2	Load Reg. 7 with 00
72	6E 75	LDV.E.75	2	Load Reg. E with 75
74	9E	LDA.E	1	Load Accum. from Reg. E
75	00	NOP	1	No operation
76	02	DEC	1	Subtract 1 from Accum.
77	27 75	BNE.7.75	2	Branch if Accum. is not = to Reg. 7 which contains 00
79	07	RET	1	Return from subroutine

Three Ways to Enter and Output a Letter

These three sample programs are presented to show you the three different instructions which can be used to input and output a letter on the screen.

For the first example, we have chosen to input and display the letter "H" or 1D in HEX Code. With this type of program, whatever is loaded into the register and is outputed to the screen will remain on the screen. You cannot change it. With this program, you could enter a complete message and have it remain on the screen.

The second example uses the instructions, "Input to a Register" and "Output from a Register," but does not designate any particular value. Thus, once the program is entered, any value can be entered and it will be displayed on the screen.

The third example is similar to the second in that any value may be entered, but it is inputed to the Accumulator rather than to a register.

You will note, in all three examples, the last instruction was "Input to a Register" which was used as a pause since no output instruction was indicated, thus only one keyboard depression could be made. Now, try this—using example two or three, program the appropriate instruction sets in order to create a loop so that all 11 positions on the screen may be used. (Hint! Refer back to the addition programs. Check to see how they let you keep entering one problem after another by returning to a previous program step.)

Three Ways to Enter and Output a Letter (For this example, use "H")

A

Step	Hex Code	Assembler Code	Byte	Remarks
00	6B 00	LDV.B.00	2	Positioning
02	60 1D	LDV.0.1D	2	Load Reg. 0 with H
04	C0	OUT.0	1	Output Reg. 0 = H
05	71	INP.1	1	Input to Reg. 1 (used as pause)

B

Step	Hex Code	Assembler Code	Byte	Remarks
00	6B 00	LDV.B.00	2	Positioning
02	70	INP.0	1	Input Reg. 0
03	C0	OUT.0	1	Output Reg. 0
04	71	INP.1	1	Input Reg. 1 (pause)

C

Step	Hex Code	Assembler Code	Byte	Remarks
00	6B 00	LDV.B.00	2	Positioning
02	04	INA	1	Input Accum.
03	0B	OTA	1	Output Accum.
04	71	INP.1	1	Input Reg. 1 (pause)

Six Letter Guess

After being entered, this program allows you to enter a six letter word into the Odyssey.² Six dots appear on the screen and your opponent enters a letter. If it is used in the word, it appears on the screen in the correct position. If the letter does not appear in the word, nothing happens.

Let's look at some of the program steps in detail:

Program step ØØ used as a flag or reference position. Ø1 is loaded into Register 7. Ø1 was chosen rather than ØØ because Ø1 can only mean the decimal number 1 and nothing else, while ØØ can be a number or the instruction "No Operation."

Program steps Ø4, Ø5, and Ø6 input 1st letter into Register 9, load a dot into Register 1, output Register 1 to screen. This is an initialization process and steps Ø7 through 27 are the same. This is done so that the six dots appear on the screen when the word is first inputed. Note the Register Use column.

Program steps 28 through 37 position Odyssey² to ØØ each time a guess is taken and output to the screen either the correct letter guessed or a dot.

Program steps 38 and 39 instruct Odyssey² to return to ØØ if Accumulator equals the contents of Register 7. The computer is now ready for a new game. (Note: This is a flag or reference point.)

Program step 4Ø inputs a guess to the Accumulator. It is compared to each register in Program steps 41 through 52.

Program steps 53 and 54 instruct Odyssey² to go to Program step 71 if a letter in the word is missing.

Program steps 71 and 72 load Register 8 with a dot.

Program step 73 loads the Accumulator from Register 8.

Program steps 74 through 85 instruct Odyssey² to branch to Program step 28 if any register (1 through 6) is equal to the Accumulator (in other words, if the register still remains a dot.)

Program steps 86 and 87 load Register 7 with a 2B (=). This is a flag.*

Program step 88 loads the Accumulator from Register 7.

Program steps 89 and 90 sound the buzz which indicates the word has been displayed correctly.

Program steps 91 and 92 instruct Odyssey² to go to step 28 for positioning.

Program steps 28 through 37 display word on screen.

Program steps 38 and 39 instruct Odyssey² to return to ØØ if Accumulator = Register 7.

Program step ØØ loads Register with Ø1 and game continues.

*Note: The 2B(=) sign was used as a flag in this instance: however, any sign could have been used instead.

Six Letter Guess

Step	Hex Code	Assembler Code	Byte	Remarks	Register/Use
00	67 01	LDV.7.01	2	Reg. 7 is loaded with reference position (flag)	01
02	6B 00	LDV.B.00	2	Positioning	1—1st dot
04	79	INP.9	1	Input 1st letter	2—2nd dot
05	61 27	LDV.1.27	2	Read 1st dot	3—3rd dot
07	C1	OUT.1	1	1st dot on screen	4—4th dot
08	7A	INP.A	1	Input 2nd letter	5—5th dot
09	62 27	LDV.2.27	2	Read 2nd dot	6—6th dot
11	C2	OUT.2	1	2nd dot on screen	7—01 (flag)
12	7C	INP.C	1	Input 3rd letter	8—7th dot
13	63 27	LDV.3.27	2	Read 3rd dot	9—1st letter
15	C3	OUT.3	1	3rd dot on screen	A—2nd letter
16	7D	INP.D	1	Input 4th letter	B—Positioning
17	64 27	LDV.4.27	2	Read 4th dot	C—3rd letter
19	C4	OUT.4	1	4th dot on screen	D—4th letter
20	7E	INP.E	1	Input 5th letter	E—5th letter
21	65 27	LDV.5.27	2	Read 5th dot	F—6th letter
23	C5	OUT.5	1	5th dot on screen	
24	7F	INP.F	1	Input 6th letter	
25	66 27	LDV.6.27	2	Read 6th dot	
27	C6	OUT.6	1	6th dot on screen	
28	6B 00	LDV.B.00	2	Position on screen	
30	C1	OUT.1	1	Put dots on screen	
31	C2	OUT.2	1	Put dots on screen	
32	C3	OUT.3	1	Put dots on screen	
33	00	NOP	1	No operation	
34	C4	OUT.4	1	Put dots on screen	
35	C5	OUT.5	1	Put dots on screen	
36	C6	OUT.6	1	Put dots on screen	
37	00	NOP	1	No operation	

Step	Hex Code	Assembler Code	Byte	Remarks	Register/Use
38	37 00	BEQ.7.00	2	Reset	
40	04	INA	1	Input guess to Accum.	
41	39 55	BEQ.9.55	2	Letter in word	
43	3A 58	BEQ.A.58	2	Letter in word	
45	3C 61	BEQ.C.61	2	Letter in word	
47	3D 64	BEQ.D.64	2	Letter in word	
49	3E 67	BEQ.E.67	2	Letter in word	
51	3F 70	BEQ.F.70	2	Letter in word	
53	12 71	GTO.71	2	Wrong guess	
55	A1	STO.1	1	1st letter correct	
56	12 43	GTO.43	2	Check next position	
58	A2	STO.2	1	2nd letter correct	
59	12 45	GTO.45	2	Check next position	
61	A3	STO.3	1	3rd letter correct	
62	12 47	GTO.47	2	Check next position	
64	A4	STO.4	1	4th letter correct	
65	12 49	GTO.49	2	Check next position	
67	A5	STO.5	1	5th letter correct	
68	12 51	GTO.51	2	Check next position	
70	A6	STO.6	1	6th letter correct	
71	68 27	LDV.8.27	2	Load Reg. 8 with dot	
73	98	LDA.8	1	Accum. is loaded from Reg. 8 which contains a dot	
74	31 28	BEQ.1.28	2	Position (Step 28)	
76	32 28	BEQ.2.28	2	Position (Step 28)	
78	33 28	BEQ.3.28	2	Position (Step 28)	
80	34 28	BEQ.4.28	2	Position (Step 28)	
82	35 28	BEQ.5.28	2	Position (Step 28)	
84	36 28	BEQ.6.28	2	Position (Step 28)	
86	67 2B	LDV.7.2B	2	Set flag to (=)	
88	97	LDA.7	1	Accum. loaded from Reg. 7 which contains 01	
89	05	SIG	1		
90	05	SIG	1		
91	12 28	GTO.28	2	Positioning	

Message

After being entered, this program allows you to press any number between 1 and 6 to call a programmed message to the screen. In the program as it is written, we have entered six messages. After studying the program, you can enter your own messages.

You will note the first Program steps, ØØ and Ø1, are load a value into Register Ø and the value is 9Ø. You will note that Program step 9Ø is the "No Operation" instruction after the last message, and that program steps 91 through 96 are a relocation table. The Hex Code at each of the program steps is the first program step number of each of the messges. It is this first instruction, "load a value into Register Ø and the value is 9Ø' which allows you to select any number between 1 and 6 to call a message to the screen. Let's look at a few of the other instructions in the program.

Program steps Ø2 and Ø3 load Register 1 with ØC (blank). This blank will be used as the space between words in messages which have more than one word.

Program step Ø4 inputs to the Accumulator; you may select 1,2,3,4,5,6, on the keyboard in order to call to the screen any one of six messages and whichever you choose will be inputed to the Accumulator.

Program step Ø5—add Register Ø to Accumulator. In other words, if we had chosen number 2, the contents of Register Ø (which are 9Ø) are added to the Accumulator (which is 2), thus 92 is now in the Accumulator.

Program step Ø6—store Accumulator in Register C; Register C now equals 92.

Program step Ø7—Register C moves the program counter to Program step 92, and the contents at Program step 92 (which are 36) are loaded into the Accumulator. This is the "Move" instruction or "Load Accumulator from a program step." Register C is always used with this instruction.

Program step Ø8—store Accumulator (36) in Register C; C now equals 36.

Program steps Ø9 and 1Ø load Register B (positioning) with the value ØØ (the furthest left position).

Program steps 11 and 12 load Register 2 with the number 11 (the number of positions on the screen).

Program steps 13 and 14 load Register 3 with ØØ to be used as a reference.

Program step 15—load the Accumulator from Register 1; Register 1 equals a blank. This begins the loop which erases an old message from the screen in preparation for a new message. You will note program steps 15 through 21, load the Accumulator with a blank, output the blank, load the Accumulator from Register 2 (11), decrement the Accumulator by 1, store the result in Register 2, and the Odyssey² branches to step 15 if the Accumulator is not equal to Register 3 (ØØ). Remember, when erasing, each of the 11 positions must be filled with a blank.

Program steps 22 and 23 load Register B with ØØ (furthest left position). This is used to position Register B in preparation for a new message.

Program step 24 takes the contents of Register C (36), moves to that program step (36) and loads the contents at that program step (14) into the Accumulator.

Program steps 25 and 26: If the Accumulator equals ØØ at this point, the Odyssey² would branch to Program step Ø4, and prepare itself for a new message. If the Accumulator contains a value (as in this example, it contains 14), then the Odyssey² steps to Program step 27.

Program step 27 outputs the contents of the Accumulator to the screen; a "T" appears. Refer to your Hex Code chart.

Program steps 28 and 29 instruct Odyssey² to go to step 24 and loop through the previous instructions to display message.* When the message is completed (note at the end of each message, there is a no operation instruction, ØØ), and the Odyssey² steps to Program step 25, the Accumulator will be equal to Register 3 (ØØ), and the Odyssey² will branch to Program step Ø4 in preparation for a new message.

*Note: When repeating the loop at Program step 24, the contents of Register C remain the same (36); however, the program counter increments by one each time so that the appropriate program step is reached.

Message

Step	Hex Code	Assembler Code	Byte	Remarks	Register/Use
00	60 90	LDV.0.90	2	Location table	0—90
02	61 0C	LDV.1.0C	2	A blank is loaded into Reg. 1	1—0C (blank)
04	04	INA	1	Press 1,2,3,4,5, or 6	2—0B (11)
05	E0	ADD.0	1	Add Reg. 0 to Accum.	3—00
06	AC	STO.C	1	Contents of Accum. are stored in Reg. C	4
07	09	MOV	1	Accum. is loaded with contents of Reg. C which is a program step number	5
08	AC	STO.C	1	Contents of Accum. is stored in Reg. C	6
09	6B 00	LDV.B.11	2	Load Reg. B with 00 (positioning)	7
11	62 11	LDV.2.0B	2	Load Reg. 2 with the number 11 (positions on screen)	8
13	63 00	LDV.3.00	2	Load Reg. 3 with 00 (used as a reference)	9
15	91	LDA.1	1	Load Accum. from Reg. 1	A
16	0B	OTA	1	Output blank from Accum.	B
17	92	LDA.2	1	Load Accum. from Reg. 2	C
18	02	DEC	1	Decrement Accum. by 1	D
19	A2	STO.2	1	Store contents of Accum in Reg. 2	E
20	23 15	BNE.3.15	2	Loop to program step 15 and output more blanks if Accum. is not equal to Reg. 3	F
22	6B 00	LDV.B.00	2	Positioning of Reg. B	
24	09	MOV	1	Reg. C moves to program step and loads contents at the program step into the Accum.	
25	33 04	BEQ.3.04	2	Restart—if Accum. equals 00, return to step 04	
27	0B	OTA	1	Output contents of Accum.	

Step	Hex Code	Assembler Code	Byte	Remarks	Register/Use
28	12 24	GTO.24	2	Go to step 24 to display message	
30	1D	*	1	Output (H)	
31	12	*	1	Output (E)	
32	0E	*	1	Output (L)	
33	0E	*	1	Output (L)	
34	17	*	1	Output (O)	
35	00	*	1	End Mess. 1	
36	14	*	1	Output (T)	
37	20	*	1	Output (A)	
38	1F	*	1	Output (K)	
39	12	*	1	Output (E)	
40	0C	*	1	Blank	
41	20	*	1	Output (A)	
42	0C	*	1	Blank	
43	0E	*	1	Output (L)	
44	17	*	1	Output (O)	
45	17	*	1	Output (O)	
46	1F	*	1	Output (K)	
47	00	*	1	End Mess. 2	
48	13	*	1	Output (R)	
49	12	*	1	Output (E)	
50	26	*	1	Output (M)	
51	20	*	1	Output (A)	
52	13	*	1	Output (R)	
53	1F	*	1	Output (K)	
54	20	*	1	Output (A)	
55	25	*	1	Output (B)	
56	0E	*	1	Output (L)	
57	12	*	1	Output (E)	
58	00	*	1	End Mess. 3	
59	2D	*	1	Output (N)	
60	12	*	1	Output (E)	
61	11	*	1	Output (W)	
62	0C	*	1	Blank	

Step	Hex Code	Assembler Code	Byte	Remarks	Register/Use
63	1B	*	1	Output (F)	
64	17	*	1	Output (O)	
65	13	*	1	Output (R)	
66	0C	*	1	Blank	
67	07	*	1	Output (7)	
68	08	*	1	Output (8)	
69	00	*	1	End Mess. 4	
70	18	*	1	Output (Q)	
71	15	*	1	Output (U)	
72	12	*	1	Output (E)	
73	19	*	1	Output (S)	
74	14	*	1	Output (T)	
75	16	*	1	Output (I)	
76	17	*	1	Output (O)	
77	2D	*	1	Output (N)	
78	19	*	1	Output (S)	
79	0D	*	1	Output (?)	
80	00	*	1	End Mess. 5	
81	23	*	1	Output (C)	
82	17	*	1	Output (O)	
83	26	*	1	Output (M)	
84	12	*	1	Output (E)	
85	0C	*	1	Blank	
86	25	*	1	Output (B)	
87	20	*	1	Output (A)	
88	23	*	1	Output (C)	
89	1F	*	1	Output (K)	
90	00	*	1	End Mess. 6	
91	30	*	1	Location table	
92	36	*	1	Location table	
93	48	*	1	Location table	
94	59	*	1	Location table	
95	70	*	1	Location table	
96	81	*	1	Location table	

*When entering single letters and numbers, the Hex Code only is used, since there is no Assembler Code for them.

You have now had some experience in entering programs and should have some insight into how a computer "thinks."

You can get a deeper understanding of your Odyssey²'s logic processes by trying to enter these programs without using the references.

First, go back and review one of the shorter programs. Enter it into the computer. Use your reference at this stage—but study it for sequence of events, logic patterns and flow. Then, go to one of the blank program sheets in the back of this book and try writing the program from scratch. No peeking.

Here are some things you'll want to keep in mind.

Odyssey² has 16 registers. Each register has room for 8 bits of information.

Your computer has a capacity of 99 program steps. There are 11 positions available on the screen. The contents of Register B position data on the screen.

You can only output data to the screen from a register or the Accumulator.

Keep the fold-outs open so you can refer to the instruction sets and check your data traffic patterns.

As you begin to write these programs by yourself, you'll get used to thinking along a computer's logic lines. You may want to try writing variations of the programs in this book. You may even want to try creating *original* programs. If something doesn't work, try it another way. You're playing with electronic building blocks that have endless combinations of possibilities.

Once you feel you have a good understanding of how Odyssey² does its computing, consider yourself graduated with high honors. If there's a computer store in your area, drop in and visit. You'll find friendly professionals who are always ready to talk computers and share knowledge.

And you now know far more than most people who walk in the door.

Operating Mode Review

There are eight operating modes in the Odyssey^2. They are:

Command	Assembler
Execution	Hex Input
Display	Input
Program	Roll

Let's look at each in detail and refer to the operation diagram on the fold-out as you read about each mode.

Command Mode

To enter the Command Mode, you may press "Reset" or "Clear" if you are in any of the following modes:

Assembler
Display
Hex Input

If you are in the Execution Mode, to enter the Command Mode, press "Reset."

Once in the Command Mode, you may enter the following modes:

Execution
Display
Program

by pressing:
E—To enter the Execution Mode. Program execution will begin with step ØØ. You are ready to play your game, write your message, solve your problem, etc. if you have already entered your program.

or by pressing:
C—To enter the Continue Mode. This mode is used to locate a problem within a register. For example, let's assume we have a 40 step program that is not working correctly. A branch decision was made at some lower step number and we would like to see if the correct branch was taken to the step number we had indicated, say Program step 14. Using the Roll Mode, we would move to Program step 14 and replace the Op Code at that step number with a halt statement (Op Code FF). To examine the contents of the program counter (which would contain the Program step 14, thus informing us of the correct branch), we would press (D) to enter the Display Mode. We would then press (P) to display the contents of the program counter. If the correct result is displayed, we would press "Clear" which returns us to the Command Mode. We now press (P) to enter the Program Mode, then press (M) to enter the Hex Input Mode, and then press (R) to enter the Roll Mode. We now must roll up (U) from step 00 to step 14 where the FF statement is located and replace it with the original Op Code. We may now place an FF statement at some other step to check another part of the program. Please note that only one FF statement at a time can be present in the program.

or by pressing:
D—To enter the Display Mode (to be explained in detail later).

or by pressing:
P—To enter the Program Mode (to be explained in detail later).

Display Mode

To enter the Display Mode, press (D) from the Command Mode. In this mode, you may display on the screen any register you wish to review. This mode is often used to troubleshoot problems, since you can check the contents of each register.

To check the registers, you press:

Ø To display the contents of Register Ø
1 To display the contents of Register 1
2 To display the contents of Register 2
3 To display the contents of Register 3
4 To display the contents of Register 4
5 To display the contents of Register 5
6 To display the contents of Register 6
7 To display the contents of Register 7
8 To display the contents of Register 8
9 To display the contents of Register 9
A To display the contents of Register A
B To display the contents of Register B
C To display the contents of Register C
D To display the contents of Register D
E To display the contents of Register E
F To display the contents of Register F
P To display the contents of the Program Counter
S To display the contents of Subroutine Counter
X To display the contents of the Accumulator

To leave the Display Mode, press "Clear" or "Reset" to enter the Command Mode.

Program Mode

To enter the Program Mode, press (P) from the Command Mode, or press "Clear" if you are in the Roll Mode.

The Program Mode sets the Odyssey^2 to accept a program. From this mode, press (A) to enter Assembler Language or press (M) to enter Hex Language (Op Code).

To leave the Program Mode, press "Reset" and you will enter the Command Mode, or you may leave the Program Mode by pressing (A) to enter the Assembler Mode or by pressing (M) to enter the Hex Input Mode.

Assembler Mode

To enter the Assembler Mode, press (A) if you are in the Program Mode, or press "Clear" if you are already in the Input Mode.

Once you have pressed (A), you are in the Assembler Mode and you may now press (I) to enter the Input Mode, or press (R) to enter the Roll Mode.

To leave the Assembler Mode, press "Clear" or "Reset" to enter the Command Mode, or press any valid Assembler Mode command (i.e., [I] or [R]). (Be sure to refer to the fold-out operational diagram.)

Hex Input Mode

To enter the Hex Input Mode, press (M) if you are in the Program Mode or press "Clear" if you are in the Input Mode.

Once in the Hex Input Mode, press (I) to enter the Input Mode, or press (R) to enter the Roll Mode.

To leave Hex Input Mode, press "Clear" or "Reset" to enter Command Mode or press

any valid Hex Input Mode command (i.e., [I] or [R]).

Input Mode

To enter the Input Mode, press (I) if you are in either Assembler or Hex Input Mode.

Once you are in the Input Mode, you may enter any assembler language instruction if you have entered from the Assembler Mode, or you may enter any HEX language instruction (Op Code) if you have entered from the Hex Input Mode. The Input Mode is the mode in which you will enter your program.

To leave the Input Mode, you may press "Reset" to enter the Command Mode, or press "Clear" to enter the Assembler or Hex Input Mode.

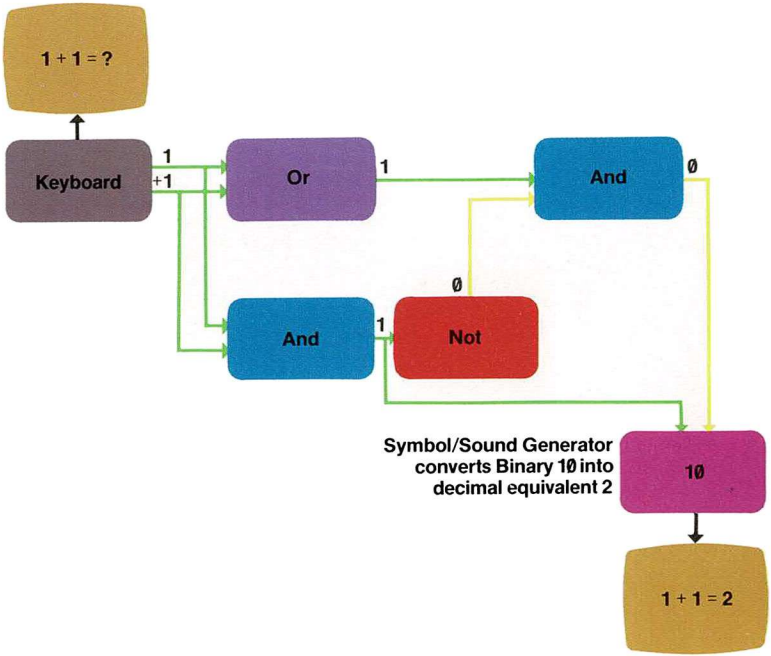
Roll Mode

To enter the Roll Mode, press (R) if you are in either Assembler or Hex Input Mode.

Once you are in the Roll Mode, you may press (U) to display the program steps from ØØ to 99, or you may press (D) to display the program steps from 99 to ØØ. This mode is often used to check a program step to be sure it contains the correct data.

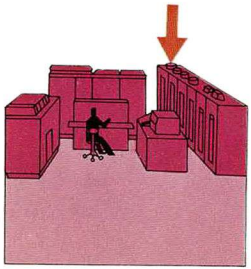
To leave the Roll Mode, press "Clear" to enter the Assembler or Hex Input Mode.

How a Computer Adds 1 + 1



Glossary

Glossary of Frequently Used Computer Terms

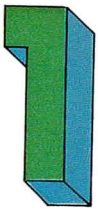


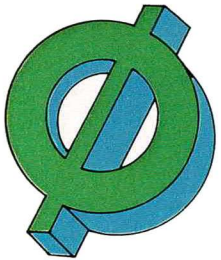
Accumulator. A working register within a computer. It is a small memory device that provides temporary data storage and/or instruction storage for the ALU. It can also store the result of the ALU's operation and may be used as an operand source for the ALU.

Applications. A job given to the computer to do.

Application Software. Programs written for the computer.

Arithmetic Logic Unit (ALU). A part of the Central Processing Unit (CPU). The ALU accepts data from different sources, acts upon it, then outputs one result. It is in the ALU that all arithmetic and logic operations are performed. It is also known as the "number cruncher" since it's here that all binary data is acted upon.





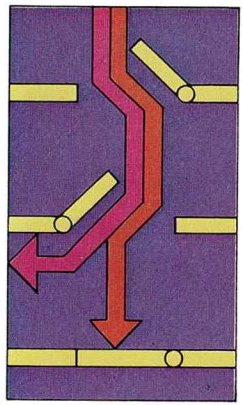
Binary Numbers. A number based on two digits. 1 and Ø. With enough 1's and Ø's, any number can be expressed. The inside of a computer is basically a series of on-off switches that turn on an electrical charge to express 1. They turn off the electrical charge to express Ø. A computer performs binary calculations by sending these on-off signals through logic gates which pass on

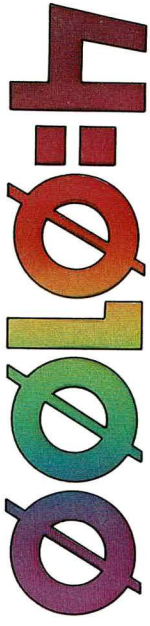
information according to the rules built into them.

The functions of these logic gates are called AND, OR and NOT.

This diagram demonstrates how a computer's logic circuit adds 1 and 1.

1. Two electrical currents enter the logic circuit. (Remember—the presence of the electricity signals 1. The absence signals Ø.)
2. The incoming currents flow to the OR gate at the top and to the first AND gate at the bottom. The OR gate lets the current through because at least one of the circuits is carrying an electrical charge. The AND gate lets the current through because both circuits are carrying a charge.
3. The current at the top heads to the second AND gate. The bottom stream is divided with one part flowing to the exit and the other part heading toward the NOT gate.
4. A NOT gate is a **"reverser."** It changes 1's to Ø's and Ø's to 1's. The NOT gate turns off the current flowing through it.
5. The second AND gate senses the absence of current from the NOT gate and the presence of current coming to it from the OR gate.
6. Since one of the streams entering the second AND gate is charged and the other is not, the second AND gate will not allow any current to pass to the exit.
7. The original two electrical currents representing 1 and 1 come out of the logic circuit as only one stream of electrical current.





This is expressed as 1 and Ø—the binary equivalent of 2.

Computers operate only on binary numbers. A computer can add, subtract, multiply (by adding the same number repeatedly), and divide (by using subtraction and addition). When a computer adds and subtracts binary numbers, there are several rules which it follows:

Binary Addition:

Ø	Ø	1	1
Ø	+1	+Ø	+1
Ø	1	1	Ø with a carry of 1

A carry bit is produced from the addition of 1+1. Binary carries are treated in the same way as decimal carries; they are carried to the left.

Example:

					(1)	(1)	(1)	(1)	
9	=	Ø	Ø	Ø	Ø	1	Ø	Ø	1
+7	=	+Ø	Ø	Ø	Ø	Ø	1	1	1
16	=	Ø	Ø	Ø	1	Ø	Ø	Ø	Ø

Binary Subtraction:

1. All ones in the subtrahend are changed to zeros and all zeros are changed to ones.
2. A one is then added to the least significant bit of the new subtrahend.
3. Add the result to the minuend and ignore the carry bit, if there is one.

Example: 1Ø = ØØØØ 1Ø1Ø minuend
−5 = ØØØØ Ø1Ø1 subtrahend
5 =

Rule 1: 0000 0101 becomes 1111 1010

Rule 2:
```
 1111 1010
+        1
 1111 1011
```

Rule 3:
```
 0000 1010
+1111 1011
 0000 0101
```

Listed below are some binary numbers and their decimal equivalents.

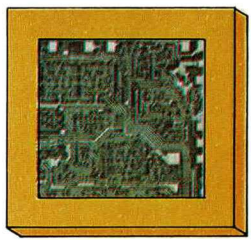
Try adding and subtracting in binary. It really works.

1=0001	5=0101	9=1001	13=1101
2=0010	6=0110	10=1010	14=1110
3=0011	7=0111	11=1011	15=1111
4=0100	8=1000	12=1100	16=0001 0000

Basic. Beginners All-Purpose Symbolic Instruction Code. This is one of the simplest programming languages and is in general use on many computers.

Bit. The basic unit of information used by a computer. A bit is a binary number. 1 or 0. A simple pocket calculator may store less than a hundred bits of information. A large computer may have somewhere between a billion and a trillion bits in storage.

Byte. A byte is eight bits on smaller computers, such as your Odyssey.[2] A byte can be twelve, sixteen or more bits on larger computers.



Chip. Nickname for Integrated Circuit. (IC)

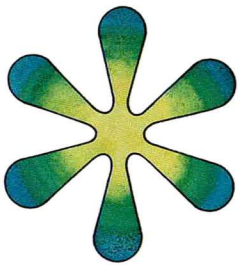
Cobol. Common Business-Oriented Language. A computer programming language used widely in the business world.

Computer System. The computer and all of its related hardware. Your Odyssey² system consists of the console, the computer cartridge and your TV—the video terminal.

Control Unit. The internal part of a computer that directs the binary traffic.

Core Memory. The internal memory of a computer. Today, this memory consists of Integrated Circuits (IC's). In the old days, the core memory was stored on magnetic cores made from tiny rings of magnetic material strung on a grid of fine wire.

Central Processing Unit (CPU). This is the brain of the computer. The CPU of your Odyssey² is contained on a microprocessor which can make over 1,000,000 electronic decisions every second.



Cursor. A visual indicator on the video display terminal of some computers that signals where a character must be corrected or the position where data should be entered. It is usually a star or an asterisk.

Data. Refers to all letters, numbers, symbols and facts which can be processed or used by a computer.

External Memory. That memory used by the computer which is not stored within the computer itself. External memory is usually

stored on cassette tape, floppy or hard disk, or paper tape.

Firmware. A combination of hardware and software. This term is generally used in reference to software stored permanently on Read Only Memory chips (ROM).

Floppy Disk. It's thin, flexible and looks like a square record—a storage device that typically holds more than 250,000 eight bit bytes of information.

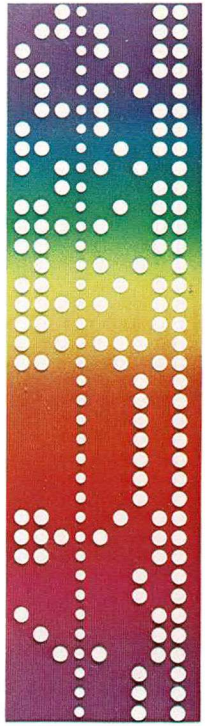
Fortran. FORmula TRANslator. The most widely used scientific computer programming language.

Hardware. The physical devices that form a computer system—including all mechanical, magnetic, electronic, electromechanical and electric components. "Software" contains the instructions that tell all this what to do.

High-Level Language. Software that makes it possible to program a computer in more or less plain English. *Basic, Cobol* and *Fortran* are all examples.

Input. The data entered into computers.

Integrated Circuit (IC). Many hundreds of electronic circuits on a single chip of silicon. This circuitry can be contained in less than a quarter inch square and can make more than one million electronic decisions per second. There are no moving parts. It is theoretically possible for an integrated circuit to function for thousands of years.



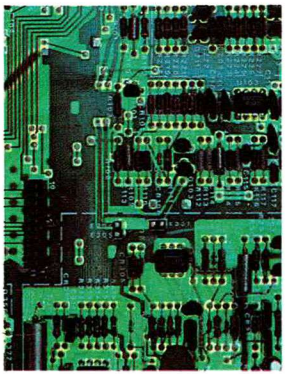
Internal Memory. Memory located within the computer itself.

Interface. Refers to the connection of one computer component to another. When your TV is connected to your Odyssey,[2] it is interfaced.

Machine Language (Binary). The instructions that a computer actually follows. High-level language instructions in Basic, Cobol and Fortran are broken down into machine language by the computer before they are executed.

Maxicomputers. The giant computers used by governments, insurance companies and businesses.

Microcomputers. Small computers with Central Processing Units contained on one single integrated circuit called a microprocessor.



Microprocessor. An integrated circuit that contains the complete central processing unit for a computer. Today, microprocessors are designed by computer technology on a much larger than life scale. Then, the circuits are transferred photographically to a chip of silicon less than one quarter of an inch square.

Nonvolatile Memory. That part of a computer's memory that is not lost when the computer is turned off.

Output. Information that comes out of a computer and is displayed on a screen (as

with Odyssey²) or on a tape, floppy disc, print out, etc.

Programmable Read Only Memory (PROM). The nonvolatile memory stored on an integrated circuit from which the computer can read information. The computer cannot store information on a PROM. The PROM is programmable because the information stored on it can be changed one or more times depending on the type of PROM.

Read Only Memory (ROM). Permanent, nonvolatile memory stored on an integrated circuit (IC). The computer cannot store information on a ROM but can read the information on it whenever necessary. The ROM is not re-programmable and the information on it cannot be modified or changed.

Random Access Memory (RAM). This is the *volatile* memory of a computer which is stored on integrated circuits. The computer can read the data stored on the chips—and new data can be entered onto the chips. This information disappears when the current is turned off. Think of the ROM as the computer's dictionary and encyclopedia and the RAM as the computer's scratch pad. One is printed and kept forever. The other is used as a worksheet and thrown away.

Software. All of the instructions used by the computer to perform its functions. This includes languages, operating systems and programs.

```
1.ØC
LDV.B.Ø
63 ØØ Ø9
6B ØØ
LDV.3.ØØ
6Ø 9Ø
LDV.Ø.9Ø
ADD.Ø
```

Instruction Sets

Instruction	Hex Code	Assembler Code	Byte
Input			
Input to Register	7R	INP.R	1
Input to Accumulator	04	INA	1
Output			
Output from Register	CR	OUT.R	1
Output from Accumulator	0B	OTA	1
One second Buzz	05	SIG	1
Change Accumulator Contents			
Set to 0	01	CLR	1
Subtract 1	02	DEC	1
Add 1	03	INC	1
Load with Random No.	08	RND	1
Load from Program Step	09	MOV	1
Combine 2 digits	8R	PAK.R	1
Separate 2 digits	BR	UNP.R	1
Load from Register	9R	LDA.R	1
Subtract from Register	DR	SUB.R	1
Add Register	ER	ADD.R	1
Change Register Contents			
Store Accumulator	AR	STO.R	1
Load A Value	6RNN	LDV.R.NN	2

Function
To store a value from the keyboard (symbol or numeral) in a specified register.
To input data from the keyboard (symbol or numeral) into the Accumulator.
To display the contents of a specified register on the television screen. If you have a series of output instructions, one right after another, you must place a "No Operation" instruction after every third output instruction.
To display data stored in the Accumulator on the television screen.
To implement a one second buzz.
To clear Accumulator and set its contents to 0.
To decrement the contents of the Accumulator by one.
To increment the contents of the Accumulator by one.
To load Accumulator with a random number from 00 to 99.
To load Accumulator with the contents (two-digit value) contained in the program step specified by Register C. When using the MOV instruction in a program, Register C must remain empty. In other words, you should not program any value in Register C.
To combine two digits from two specified registers in the Accumulator. This instruction is used when working with numbers. Since the microprocessor reads the numbers in Binary, we must instruct it to combine two digits in order to produce and display a base 10 number. The first register must always contain a number less than ten.
To separate two digits in the Accumulator and store them in two specified Registers.
To load the Accumulator with the contents of a specified register.
To subtract the contents of the Accumulator from a specified register and store the results in the Accumulator.
To add the contents of a specified Register (R) to the contents of the Accumulator and to store the result in the Accumulator. If the result is larger than two digits, only the lowest two digits will be kept.
To store contents of the Accumulator in a specified register.
To load a value (NN) into a specified register (R). Registers may be 0 through 9 or A through F.

Instruction	Hex Code	Assembler Code	Byte
Control Execution Order			
No Operation	00	NOP	1
Halt	FF	HLT	1
Go to Subroutine	14NN	GTS.NN	2
Return from Subroutine	07	RET	1
Branching Decision			
Branch on Decimal Borrow	10NN	BDB.NN	2
Branch on Decimal Carry	11NN	BDC.NN	2
Branch Unconditionally	12NN	GTO.NN	2
Branch if Accum. is 0	13NN	BRZ.NN	2
Branch if Reg. not = Accum.	2RNN	BVNE.R.NN	2
Branch if Reg. = Accum.	3RNN	BEQ.R.NN	2
Branch if Reg. is greater than Accum.	4RNN	BGT.R.NN	2
Branch if Reg. is less than Accum.	5RNN	BLS.R.NN	2

Function
To implement a delay in execution of the program. Can be used when writing a program to utilize several program steps, so that when checking the program, if an extra instruction step is needed, several will be vacant.
To halt execution of program in order to enter a different operational mode to check registers. Used for trouble-shooting. The halt instruction is entered after your program is entered. In other words, you would enter your complete program, then, using the Roll Mode, you would enter a halt instruction (FF) in place of an instruction already programmed. After entering the Display mode and checking the registers for errors, you would return to the program step containing FF, clear it, and re-enter the program step you had removed. (See Continue Mode description in the Operating Mode Review.)
To instruct microprocessor to branch to a specified program step (NN) which contains an operation which you may wish to use several times in one program. This instruction set allows you to use the same operation several times without having to rewrite it. The next sequential step number is saved for returning from the subroutine. (See sample program "Area Problems Using Subroutine and Return.") You must have a "Return from Subroutine" when you have a "Go to Subroutine."
To instruct the microprocessor to return to a specified program step. This would be the step immediately following the instruction set "Go to Subroutine." (See sample program "Addition Flash Cards.") Your must have a "Go to Subroutine" in order to have a "Return from Subroutine."
To instruct the microprocessor to branch to the specified program step (NN) if the high order bits of the Accumulator equal a (9).*
To instruct the microprocessor to branch to the specified program step (NN) if the high order bits of the Accumulator do not equal a (Ø).*
To instruct the microprocessor to branch to a specified program step (NN). (See sample program "Message.")
To instruct the microprocessor to move to another program step if conditions are satisfied. Most often used in arithmetic problems. (See 1 Digit Division.)
To instruct microprocessor to branch to a specified program step (NN) if Accum. is not equal to a specified register (R). (See sample program "Message.")
To instruct microprocessor to branch to a specified program step (NN) if the contents of the Accum. are equal to the contents of the specified register (R). (See sample programs "One Digit Multiplication" and "Six Letter Guess.")
To instruct the microprocessor branch to a specified program step (NN) if the specified register (R) is greater than the Accum.
To instruct the microprocessor to branch to a specified program step (NN) if the specified register (R) is less than the Accum. (See sample program "One Digit Division.")

*The Accumulator contains eight bits of data. The first four are the high order bits and the last four are the low order bits.

Program Sheets

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

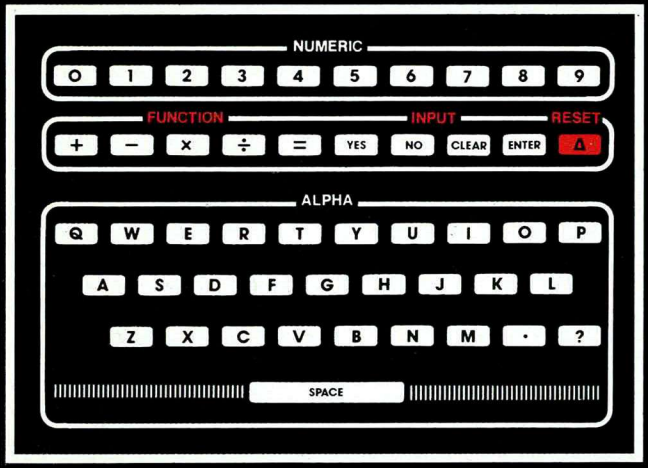
Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

Program Name _____ Date _____ Page _____

Step	Hex Code	Assembler Code	Byte	Remarks	Register Use
					0
					1
					2
					3
					4
					5
					6
					7
					8
					9
					A
					B
					C
					D
					E
					F

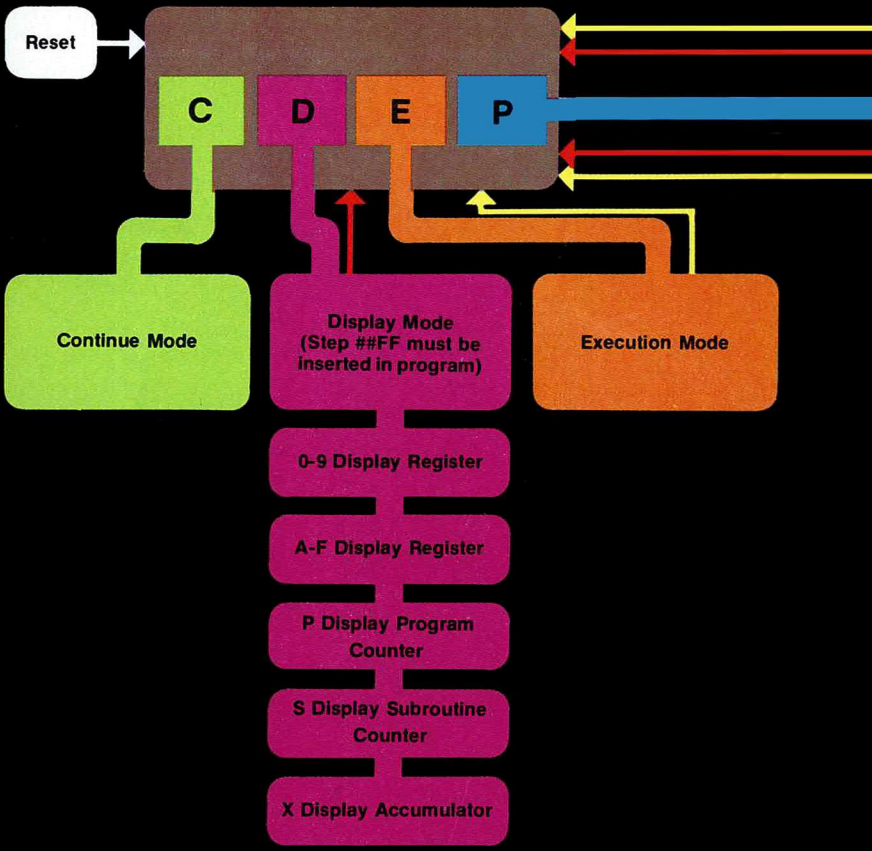
this gatefold will provide you with an electronic road map—please keep it open as you work with your Odyssey² computer

Keyboard


NUMERIC
0 1 2 3 4 5 6 7 8 9
FUNCTION
INPUT
RESET
+ − × ÷ = YES NO CLEAR ENTER
ALPHA
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M . ?
SPACE

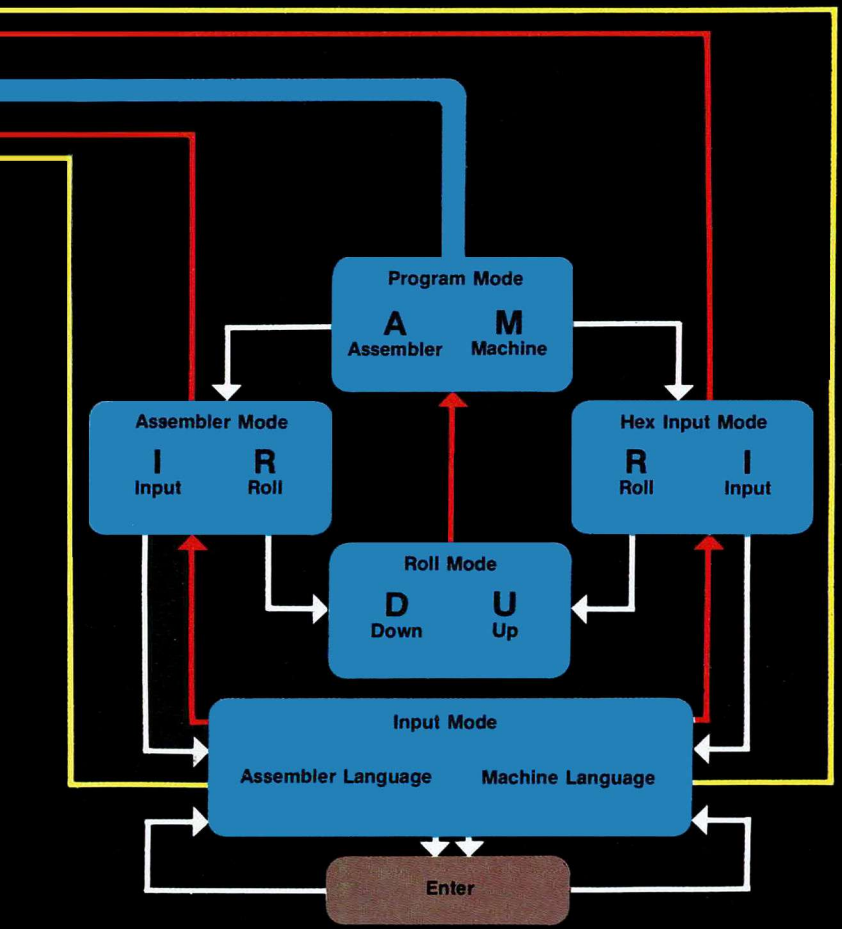
Computer Symbols

External Flow



Legend

Reset ■ Clear ■


Program Mode
A
Assembler
M
Machine
Assembler Mode
I
Input
R
Roll
Hex Input Mode
R
Roll
I
Input
Roll Mode
D
Down
U
Up
Input Mode
Assembler Language
Machine Language
Enter

IB3389-1 AJ9406

THE ULTIMATE COMPUTER VIDEO GAME SYSTEM...BY MAGNAVOX

COMPUTER INTRO!

It's not for everyone—but if you're up to a rewarding mental challenge, here is a fascinating entry point into a complex and highly technical subject. This cartridge turns your Odyssey² into a very special kind of computer. It won't balance your checkbook or plot the course of a spaceship to Mars—but it will give you some idea of how those computers do their work. You will learn how computers "think." You will learn how to enter a program—the first step in learning how to actually write one. Electronic road maps graphically show you where each byte of data goes—and what happens to it. Then, you will actually run the program and see the exciting results on your TV. Shut off the power and the old program is automatically erased so you can enter a new one immediately!

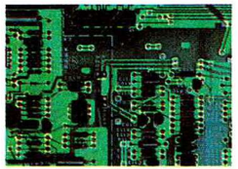
Notes about the PDF transfer:

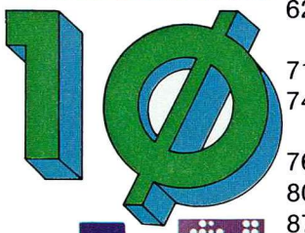
The Computer Intro! manual was bound with double wire rings to make it easy to flip open. The original page size is 5" wide x 6.5" tall. The paper is glossy.

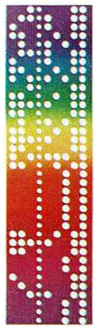
The original manual has three gatefold pages. One just inside the front and back covers and one at the start of the glossary. They are folded such that they are 4 3/4" wide. This shortening makes it easy to flip to the glossary.

This PDF version is modified to eliminate the gatefolds to make it more printer and display friendly. The original gatefolds are included hereafter for archival purposes. You will need a ledger size printer to print them out at their original size.

Contents





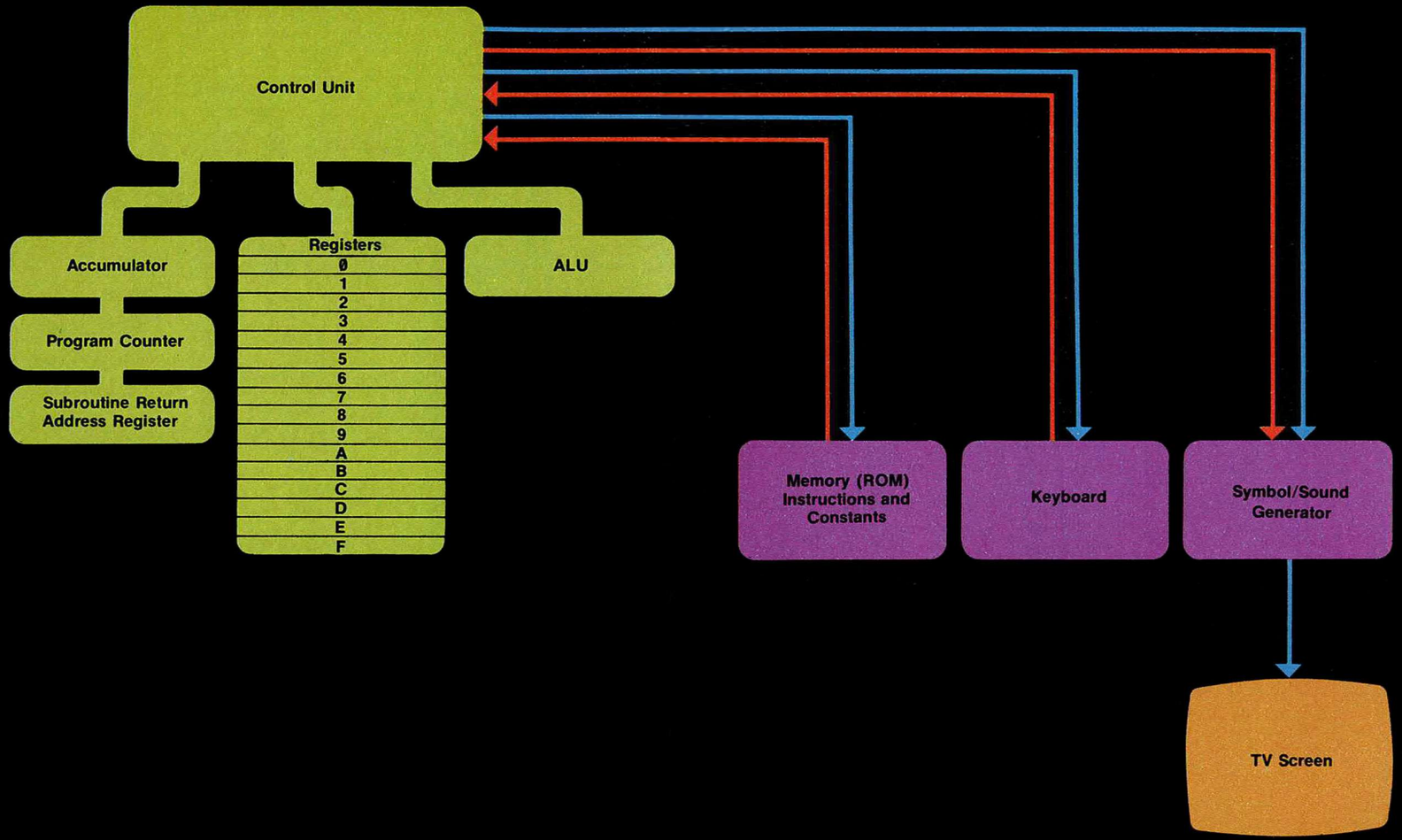


this gatefold will provide you with an electronic road map—please keep it open as you work with your Odyssey² computer

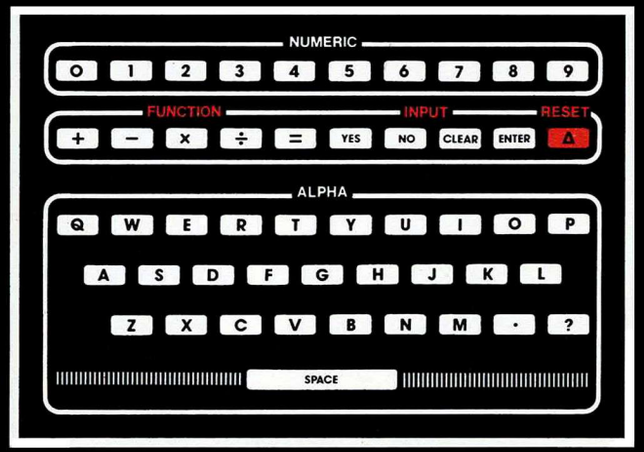
Key Codes

Key Code	Hex Code	Decimal Equivalents	Key Code	Hex Code	Decimal Equivalents
0	00	00	O	17	23
1	01	01	P	0F	15
2	02	02	Q	18	24
3	03	03	R	13	19
4	04	04	S	19	25
5	05	05	T	14	20
6	06	06	U	15	21
7	07	07	V	24	36
8	08	08	W	11	17
9	09	09	X	22	34
A	20	32	Y	2C	44
B	25	37	Z	21	32
C	23	35	Blank	0C	12
D	1A	26	:	0A	10
E	12	18	$	0B	11
F	1B	27	Clear	2E	46
G	1C	28	?	0D	13
H	1D	29	•	27	39
I	16	22	+	10	16
J	1E	30	-	28	40
K	1F	31	X	29	41
L	0E	14	÷	2A	42
M	26	38	=	2B	43
N	2D	45	Enter	2F	47

Internal Flow



Keyboard



any valid Hex Input Mode command (i.e., [I] or [R]).

Input Mode

To enter the Input Mode, press (I) if you are in either Assembler or Hex Input Mode.

Once you are in the Input Mode, you may enter any assembler language instruction if you have entered from the Assembler Mode, or you may enter any HEX language instruction (Op Code) if you have entered from the Hex Input Mode. The Input Mode is the mode in which you will enter your program.

To leave the Input Mode, you may press "Reset" to enter the Command Mode, or press "Clear" to enter the Assembler or Hex Input Mode.

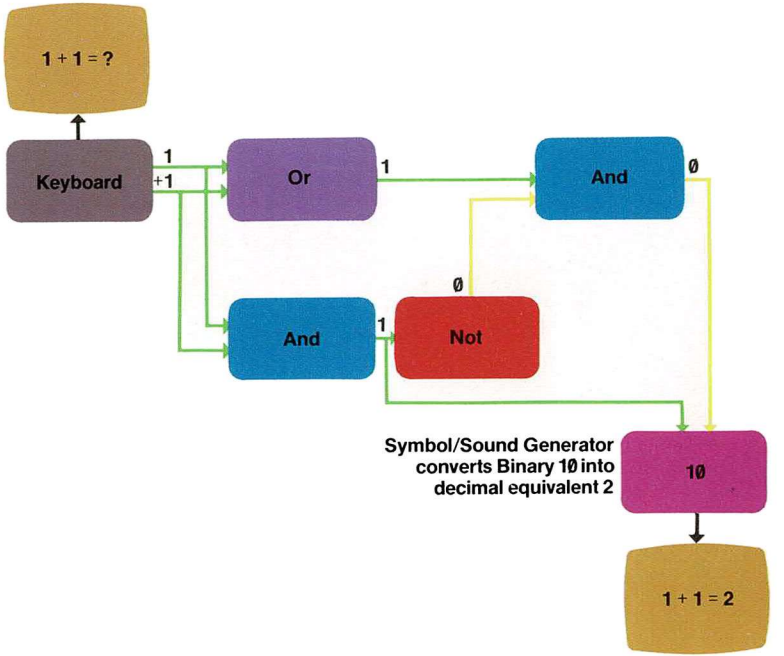
Roll Mode

To enter the Roll Mode, press (R) if you are in either Assembler or Hex Input Mode.

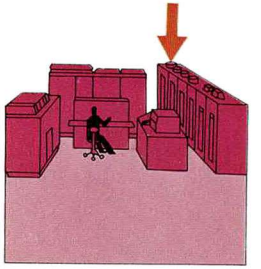
Once you are in the Roll Mode, you may press (U) to display the program steps from ØØ to 99, or you may press (D) to display the program steps from 99 to ØØ. This mode is often used to check a program step to be sure it contains the correct data.

To leave the Roll Mode, press "Clear" to enter the Assembler or Hex Input Mode.

How a Computer Adds 1 + 1



Glossary of Frequently Used Computer Terms

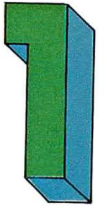


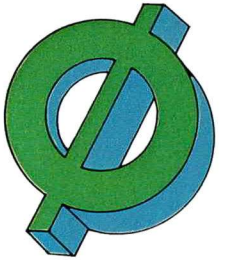
Accumulator. A working register within a computer. It is a small memory device that provides temporary data storage and/or instruction storage for the ALU. It can also store the result of the ALU's operation and may be used as an operand source for the ALU.

Applications. A job given to the computer to do.

Application Software. Programs written for the computer.

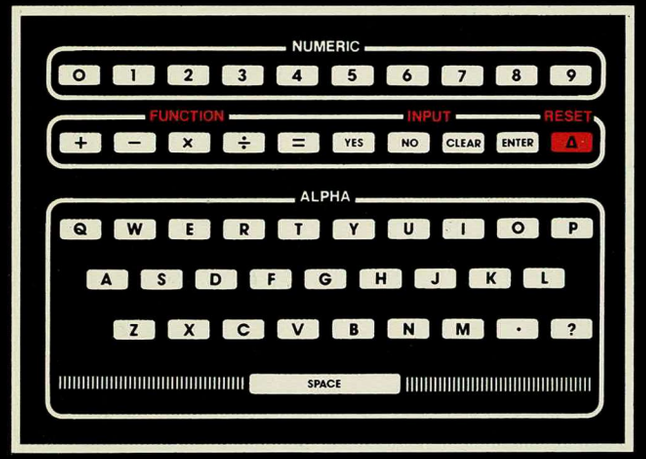
Arithmetic Logic Unit (ALU). A part of the Central Processing Unit (CPU). The ALU accepts data from different sources, acts upon it, then outputs one result. It is in the ALU that all arithmetic and logic operations are performed. It is also known as the "number cruncher" since it's here that all binary data is acted upon.



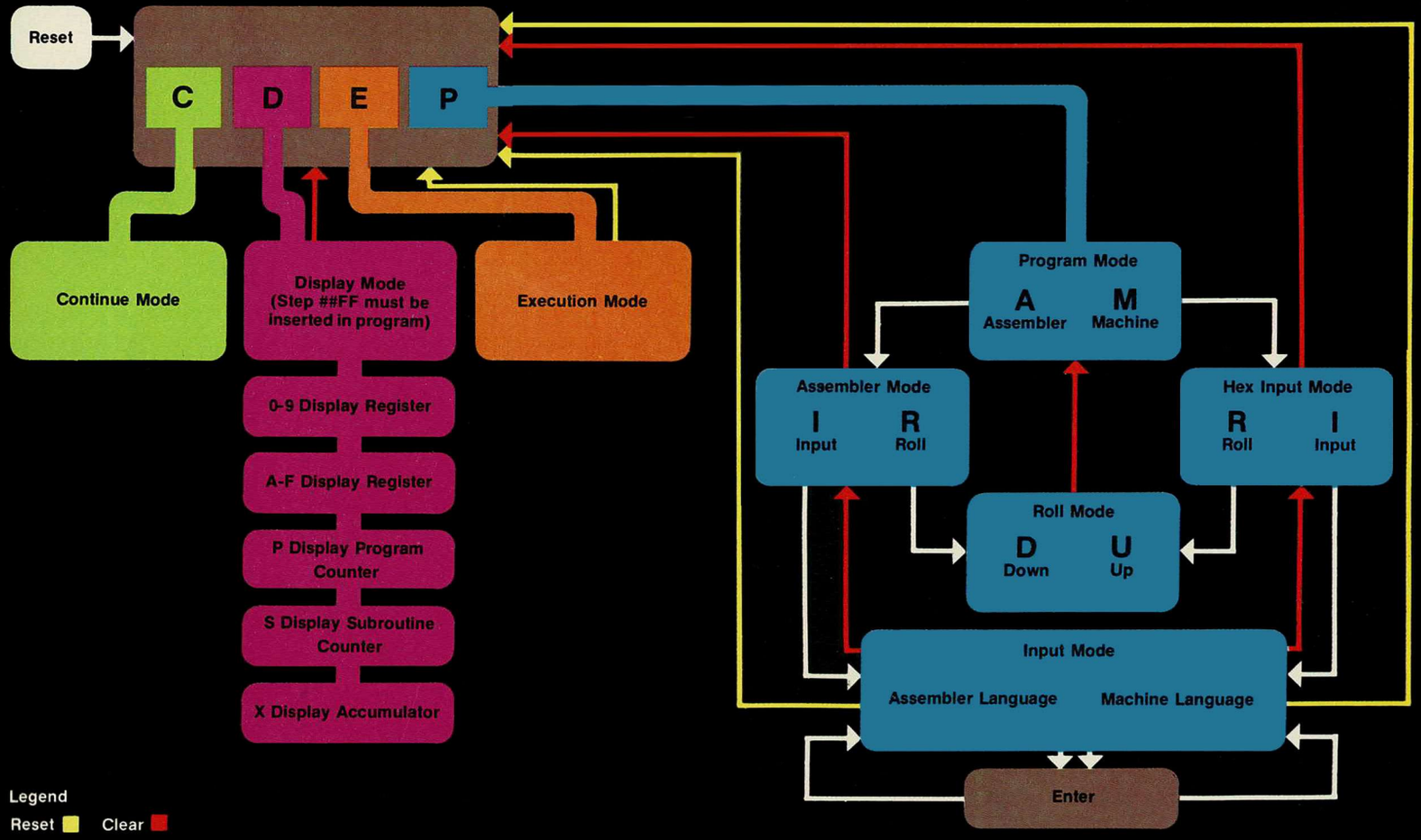


Binary Numbers. A number based on two digits. 1 and Ø. With enough 1's and Ø's, any number can be expressed. The inside of a computer is basically a series of on-off switches that turn on an electrical charge to express 1. They turn off the electrical charge to express Ø. A computer performs binary calculations by sending these on-off signals through logic gates which pass on

Keyboard



External Flow



Computer Symbols

this gatefold will provide you with an electronic road map—please keep it open as you work with your Odyssey² computer