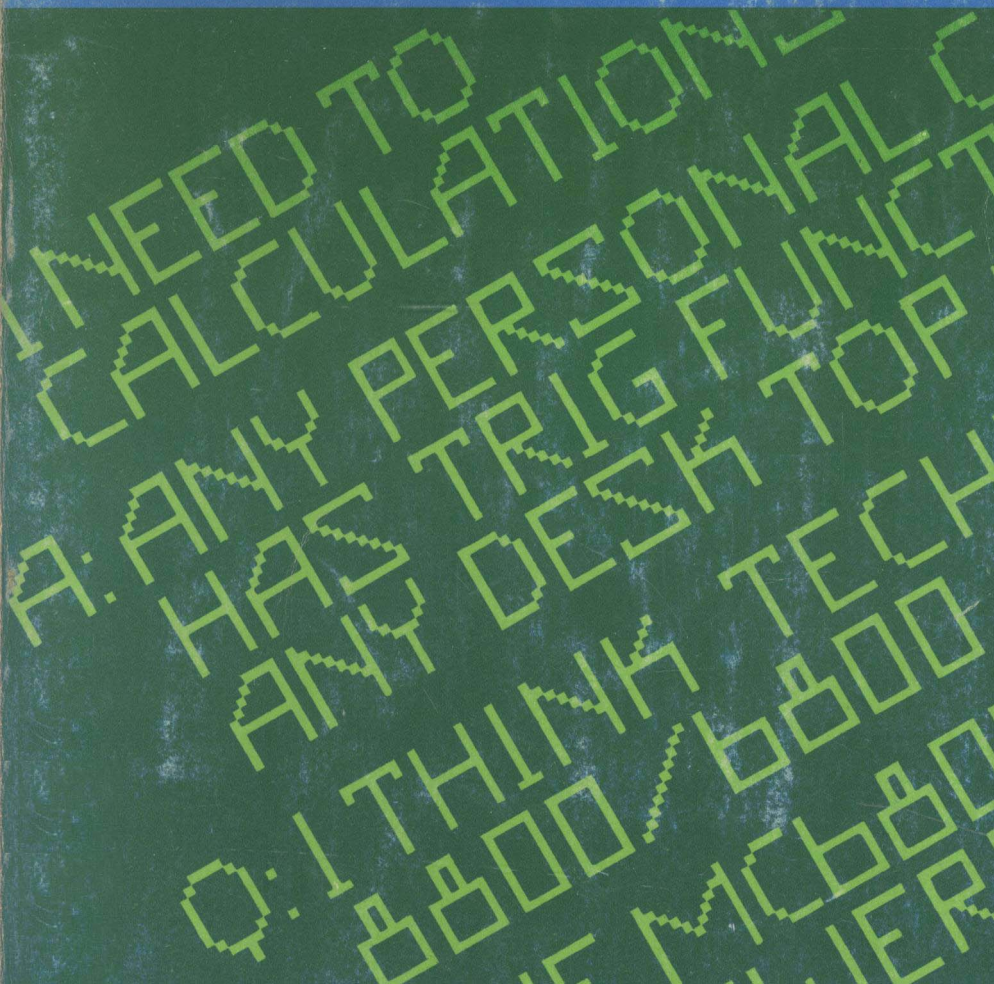
dilithium PRESS

HOME COMPUTERS: 2^{10} Questions & Answers Volume 1: Hardware

Rich Didday



HOME COMPUTERS: 2^{10} Questions & Answers Volume 1: Hardware

Rich Didday

dilithium PRESS
P.O. Box 92,
Forest Grove, Oregon 97116

The technical information, statements about specific products, and descriptions of specific computer languages in this book are based on fact and are believed to be accurate. The characters are fictional, and are not intended to portray any real persons, living or dead. Some fragments of conversations are based on real conversations.

HOME COMPUTERS: 2^{10} QUESTIONS AND ANSWERS
Volume 1: Hardware

Rich Didday

ISBN 0-918398-00-2

Library of Congress Cataloging in Publication Data

Didday, Richard L
Home computers.

Bibliography: p.
Includes index.
CONTENTS: v. 1. Hardware.—v. 2. Software.
1. Microcomputers—Miscellanea. I. Title.
TK7885.4.D53 001.6'4'04 77-9285
ISBN 0-918398-00-2 (v. 1)
ISBN 0-918398-01-0 (v. 2)

PRINTED IN THE UNITED STATES OF AMERICA

PREFACE

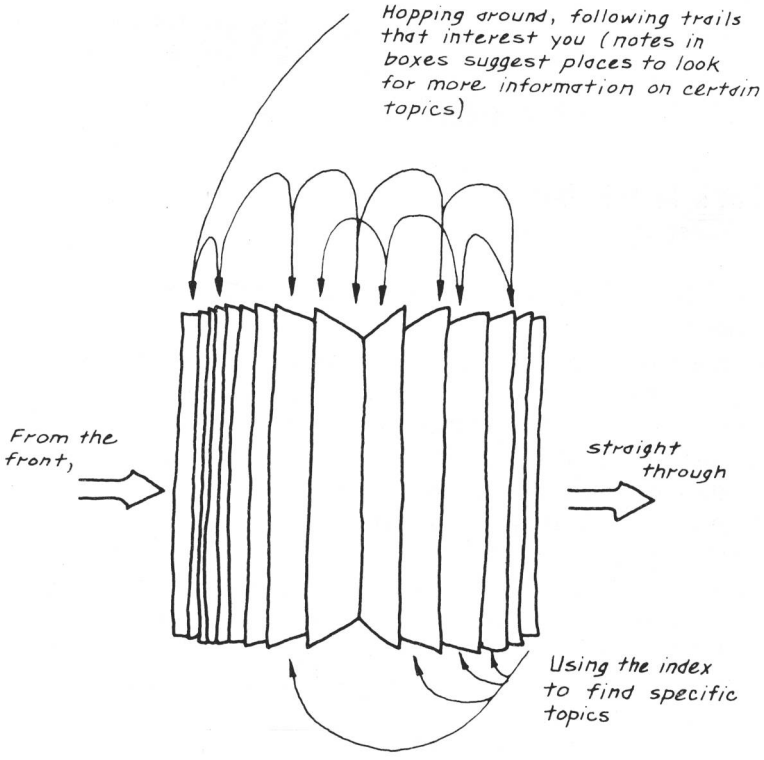
What's in this book,
2^{10} Questions and Answers

This is the first volume of the two-volume set 2^{10} Questions and Answers.

This book has two main purposes. First, it's intended to give you a real feeling for what's involved in home computing so you can make rational decisions about what way you want to go *before* you trade your hard-earned cash for equipment. Second, it's intended to give people who come with an interest, but no specialized knowledge, a general background in computing in general and microcomputers in specific. Enough of a background so that you'll have no trouble understanding articles about advanced projects in the computer hobbyist magazines, ads for home computing equipment, and people who do have specialized computing knowledge. There is *no* intent to push any specific products, nor is there an attempt to cover advanced, esoteric hardware or software techniques. The whole idea is to get you to the point where you can make your own, informed decisions about what to get and what projects to attempt.

A glance at the Table of Contents and a few minutes of flipping through the book will show you how these purposes are accomplished. The book is expressed in the form of a dialog. One participant (A) has a substantial background in computing and home computing, the other (Q) is a bright, interested newcomer. In addition, an Editor adds occasional fine points and clarifications. Diagrams, Tables, and Appendices are used to provide additional information in a compact form.

Although there is a definite structure to the whole book, so that by reading from start to finish, you will find an orderly progression of material (general overview of microcomputing including both hardware and software → number systems → digital logic notation → understanding different kinds of diagrams → assembling a computer from a kit → details of special microprocessors), the book is also designed to make it easy to skip around, covering topics of special interest to you (see Figure 0). Regardless of your specific background, I'm sure you'll find many topics of



Different Ways of Using this Book

interest, simply because computing itself is such a rich, fascinating, lively activity. (See figure 0)

And how I came to write it.

I've been interested in computing for a long time (I wrote my first programs in 1964, soldered my first digital logic chips together in 1965), and when affordable computers became available, I jumped at the opportunity to have my own system, free from the constraints imposed by the companies and universities whose computers I had been using. As I went around talking to people, dropping into newly opened computer stores, going to conventions, I was struck by the high proportion of people I met who had substantial backgrounds in electronics and computer hardware — there seemed to be relatively few newcomers. "Why is that?", I wondered. Further investigation revealed what

should have been obvious. Although dealing with computers is inherently no more difficult than working on your car, and although programming in a language like Basic is easily and commonly taught to children, the mystique and special jargon built up around computers serves as a barrier. People can't be expected to take a deep interest in computing if they can't really understand what's being talked about and if they have no way of knowing what they're in for.

My first plan was to produce a short book of answers to questions like "What is a buffer?", "What is an interface?", "What is the difference between static and dynamic memory?", "What is assembly language?", etc. That failed because the terms, although each one is simple in itself, are densely interrelated and make little sense out of context. The solution was to write a more extensive book consisting of coherent conversations involving the terms to be explained. But now another problem arose. If enough material was included to make the book of real use to people with wide differences in background, the cost to the purchaser would be objectionably high. Fortunately, the material lent itself quite naturally to being divided in two, so the person who is interested mainly in hardware aspects of home computing as well as the person who is interested mainly in programming can each find the material they want in a moderately priced book (the first five days and the last five days, respectively).

Overall, it turned out to be much more work than I'd bargained for, but I feel it will be well worth the effort if it succeeds in helping people over the initial barriers, enabling them to discover the joys and excitements of computing.

Thanks to . . .

Many, many thanks to "Nick" Nichparenko, Dan Ross, Dennie Van Tassel, John Craig, the Byte Shop of Santa Cruz, Merl Miller, Rob Walker of Intel, Margaret Kinstler, Raymond Langsford, the late Walter Orvedahl, Rex Page, the lady in Albuquerque, and William Makepeace Thackeray.

CONTENTS

DAY 1: OVERVIEW

What are we doing here?

Q . . . recorder is finally working, so I guess we can get started.

A OK, ask me a question. Anything you want.

Q Before we get going, there's something I've been wondering about ever since you called to ask if I'd do this.

A Well?

Q You said you wanted to do a book of conversations about home computers, and you wanted to call it (if I remember right) 124 or 256 or 1028 Questions and Answers About Home Computers.

A Right . . . I want to call it 128 or 256 or 512 or 1024 Questions and Answers, depending on how many we wind up with.

Q 1 But every other book like this is called 101 or 1001 Questions and Answers About Aardvarks, or whatever.

A Ah. But I want the number we use to be a power of two. See, 128 is two to the seventh, 256 is 2^8, 512 is 2^9, 1024 is 2^{10} . . .

Q 2 Oh. Because computers work with binary numbers. Is that the idea?

A Right. I thought it would tie in with computers better. Make sort of a snazzy title. People who work with computers are always coming across powers of two because digital computers are made up to two-state elements, and . . .

Q 3 Wait — I have lots of questions about what you just said. Like, why *two*-state elements, and what does **state** mean, for that matter, and do I have to learn trivia like the powers of two to get anything out of computers, and . . .

A Hold It! Let's not get too far ahead of ourselves. No, you don't have to memorize the powers of two if you don't want to. There are so many aspects of computing that you *can't* learn everything right at the start. Designing, building, using, and thinking about computers can be done separately or in various combinations, and each phase requires different skills. You can start with the areas that seem most interesting to you, and not bother with things that seem boring. If you find that you need to know something you skipped over earlier, when you go back

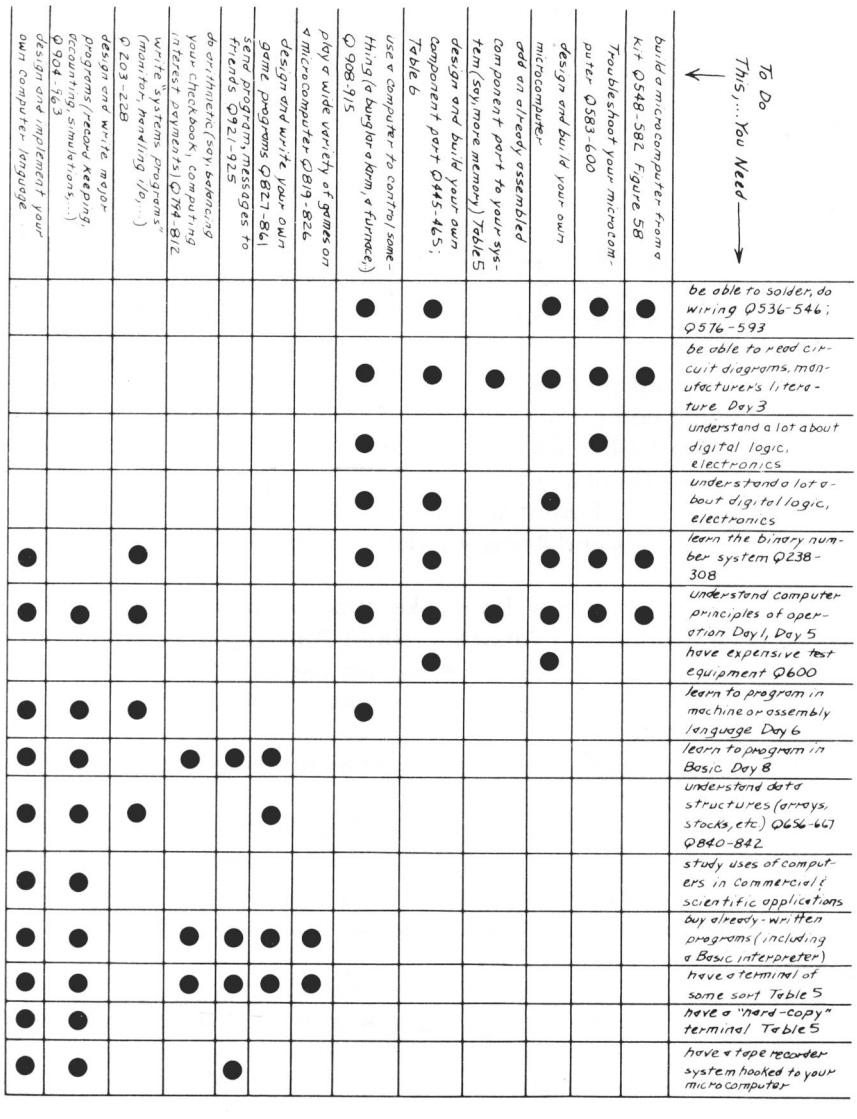


Table 1

over it, it won't seem boring any more.

Q 4 I presume that we're going to go into all the things mentioned in Table 1?*

A If you ask all the right questions, I suppose we will.

Q In detail?

A Well, I don't think we'll have time for complete detail on every aspect of home computers. My idea was that since you've just developed an interest in computers, and you seem to have bought every piece of literature you could find in the local computer store . . . Good grid! What all do you have in that bag?

Q 5 Here. Have a look.

A Hmmm. An issue of Kilobaud, an issue of Popular Electronics, Byte, 73, Interface, Radio-Electronics, Microtrek, Computer, a bunch of sales literature, the People's Computer Company Newsletter, Dr. Dobb's Journal, Personal Computing, Computer Notes**,. . .*Swank?*

Q 6 Ooops! How did that get in there?

A Quite a pile.

Anyway, my idea was that I could provide some background, cover the main ideas, help you over some of the hurdles to understanding what they're talking about in the articles, help you understand how to read the ads. I thought I could tell you what's involved in putting a computer kit together, and what it really feels like to program. Generally what I thought I could do is give you a feeling for what it's all about, so you'll have some basis for making a decision before you go rushing off to buy a lot of equipment, or start tackling some big project.

Q 7 I have a feeling already that I'm going to have to ask you questions to figure out what *you're* talking about — there are a lot of terms in Table 1 that I'm not sure about.

A Well, there's no way around that. There are a lot of inter-related concepts in computing. We'll just have to start in somewhere, and then go back and keep filling in the parts you don't understand. Today, let's try to stay pretty general so you get an overview of all aspects of home computers. Then we can take a day or so going into detail on each of the major topics.

Want to go back now to your question about **state**?

Q 8 All right. Well, before that, why don't you tell me why you're calling these things **home computers**. I've

* Table 1 shows what different sorts of skills and knowledge are needed for a range of different activities.

** descriptions of these and other relevant publications appear in the Bibliography

skimmed through some of the magazines, and they use terms like **microcomputers, microprocessors, personal computers,** . . .

A OK. Calling them **home computers** may just be my own little idiosyncrasy. A **microprocessor** is one of the key parts of a **microcomputer.** A **microcomputer** is called that because it's much smaller in physical size (but not necessarily in capabilities) than older computers. **Home computers** or **personal computers** are computers (almost always microcomputers) that people use in their home, with friends, or in small businesses, that is, that aren't used in industry or universities for scientific or engineering problems or for large commercial data processing jobs. I use the term **home computers** just because **personal computer** makes me think of a pocket calculator, and **microcomputer** makes them sound insignificant. No big deal.

Q 9 Are we starting at the right place? I'd really just like to ask "what's a home computer, what can you do with it, and how do you do it?", but I realize that's a pretty broad question.

A Well, hopefully, you'll know all that after all our conversations. Some of this will probably be familiar at first, because it's all assumed in the magazines, but . . .

Q 10 That'll be all right, I think. Sometimes when I look at them it just seems like a jumble of unrelated things.

A Ask me how computers work.

Q 11 All right, how do computers work?

A By the hour, with no time off for lunch, and no old-age pension plans.

Q Very funny.

A OK. Let's start at the most general level.

The general organization of computer systems

There are two major categories of things associated with all computers, namely **hardware** and **software. Hardware** refers to the parts of a computer system that you can touch. Actual physical devices. The chips containing the electronics, the wires, the switches, lights, power supply, terminal, plugs, sockets, tape recorders, the boxes the parts are mounted in, etc.

Software refers to things that exist as patterns, namely programs, data, stored values.

There are three major sub-parts of a computer's hardware, which I'll call the **controller** (or **processor**), the **memory,**

and **input/output devices.** (Incidentally, **input/output** is always abbreviated as **i/o.**)

When you use a computer, you enter a **program** (which is a piece of software), and cause it to be carried out (by the hardware). A **program** causes the computer to go through some sort of symbol manipulation process, possibly to accept data from you, and certainly to output information in some form that you can use. The **memory** stores the program, data, and partial results. The **controller** interprets and carries out the instructions in a program. (That's what a **program** is — a sequence of instructions to the controller.) The **i/o devices** permit communication between the controller and the "outside world".

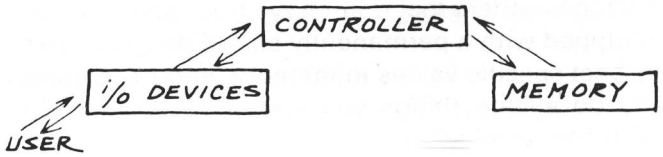


Figure 1 The Conceptual Computer

To give you a feeling for how all these parts interact, let me talk through what happens when you write a program and run it.

Q 12 All right. But I'm going to start writing questions down to ask later.

Where do you start?

A Let's see. First, you need a computer.

Q I think I could have figured that part out.

A Then we would write a program. That involves having some task or problem in mind, figuring out (in detail) how to solve it, and then translating our solution into some language the computer is equipped to accept.

Q 13 That depends on what brand the computer is?

A Not necessarily. There are some languages, called **higher-level languages*** that are accepted by a wide range of different computers. Anyway, once we have our program written, we enter it into the machine through some sort of **i/o device.**

Q 14 For instance?

A The most commonly used i/o devices on home computers are the **switches** on the front panel of the microcomputer itself, a **keyboard** used in conjunction with a TV set, and the ever-popular **Teletype.**

*higher level languages Q 185-189

Q 15 But not all computers have switches on the front. I've seen ads for several that don't. Why not? Is that bad?

A We're getting a little off the track, but I'll try to answer you. You can use a microcomputer which has panel switches without having to buy any kind of terminal, because you can enter programs and inspect values in memory by flipping the appropriate switches and looking at the lights. If the computer doesn't have front panel switches, you've got to have some kind of terminal to do anything with it (besides use it as a paper weight).

Q 16 So it *is* bad if it doesn't have switches.

A No, it all depends on what you want to do. You'd go bananas doing anything but really small programs through the switches, so you need a terminal anyway. Also, the microcomputers that don't have front panels usually come equipped with a permanently stored program that lets you inspect specific values in memory, enter programs, generally do all the things you can do using switches, right from the terminal.

Q 17 So now you're saying it's a waste of money to have the switches?

A No, I'm saying that it's a matter of taste. Some people feel more comfortable with switches and lights on the front panel, and some people feel it's more convenient to be able to do everything form the terminal and don't care if they have the switches at all.

Q 18 What do you think I'd like better?

A Don't know. Go to your local computer store and play around with one of each type. [Editor's note: The trend seems to be away from front panels.] Let's see. Where were we?

Q I think we were entering the imaginary program into the computer.

A Oh, right. OK. So we have the program written, and then we'd enter it, one character at a time . . .

Q 19 Do you mean anything special by that?

A No, I mean that just as you type one letter at a time on a typewriter, you enter one character at a time into the computer. As each character enters, it goes to the controller (processor) and then gets stored in memory, one by one. Of course, it's an electrical signal that we're talking about, a coded form of the characters. We'll . . .

Q I know. We'll get to that later.

A OK. So once we're through entering our program, and it's neatly stored away in memory, we cause the controller to start **running** or **executing** it.

Q 20 How do you do that?

A Depends on the details of the particular system. Either you type in one last thing, or you flip some switches.

Now comes the key part to understand. A program, as I said before, is a sequence of commands or instructions to the controller. To **run** a program, the controller brings one instruction at a time from memory, and then carries it out. Then it fetches the next instruction from memory, carries it out, and so on.

Q 21 Wait a minute . . . it doesn't sound like you're really telling me very much. What are these instructions like?

A OK. Here's what some typical instructions do.

Get a value from a particular place in memory and bring it into the controller.

Store a value in a particular place in memory.

Stop.

Form the sum of two values.

Accept a value from a specific i/o device.

Test a value to see if it's zero, and if it is, take the next instruction from some specific place in memory.

Q 22 Those don't sound very exciting.

A Maybe not, but by putting them together in the right order, you can create programs that do myriad and marvelous things. That is, you can do that once you understand two things. First, you need to understand what each type of instruction actually does, that is, how the various instructions make the parts of the computer system (**controller, memory,** and **i/o**) perform and interact. Second, you need to learn how to make correspondences between parts of the problem you're trying to solve and things going on in the computer.

Q 23 Is it hard to learn to program?

A Depends on you and how deeply you want to get into it. Most people I know think it's really fun. I know I do.

Q 24 Now that I think about it, I'm not sure I understand what your second thing was. The thing about correspondences between your problem and the computer. If the controller carries out instructions, it must know what the correspondences are. How else would it know what to do?

A Ooosh. I guess I've been too sloppy in explaining this so far. There's a really important idea to get here, so you don't go around thinking computers are magic. See, when I say things like "the controller fetches an instruction, figures out what it is, and carries it out", that's just a colorful way of speaking. It's not as if there's a tiny person in there who thinks about what to do. Hmmm . . .

Q Don't be ridiculous! Of *course* I don't think there's a tiny . . .

A Here. Here's the crux of the thing. You know those signs in radio stations that light up and say "ON THE AIR"?

Q I've seen them in movies, I know what you mean.

A OK. When a sign like that lights up, and we look at it, we say that it "means" something, right? But we know that what it means isn't in the sign itself, right? The sign itself is either on or off, that's all. We interpret the sign to mean that the station is operating, and that we should be quiet, or whatever.

Q Sure, sure.

A OK. When a computer prints out some sequence of letters, say, uh, "YES, I UNDERSTAND.", it can look meaningful to us, and it may well *be* meaningful to us, but the meaning doesn't arise mysteriously from within the computer circuitry, it comes from the way the designer of the computer and the programmer assigned **symbols** (in this case **letters,** in other cases, **numbers**) to processes going on in the computer.

Q 25 But wait a minute. Computers *can* make decisions, right? One of the instructions you listed a while back decided whether or not some number was zero, right?

A Yes and no. Computers "decide" to do something in the same sense that a light "decides" to go on depending on whether or not the switch is thrown or not. Here. Let me draw out a little example that shows how meanings get assigned to events in electrical circuits.

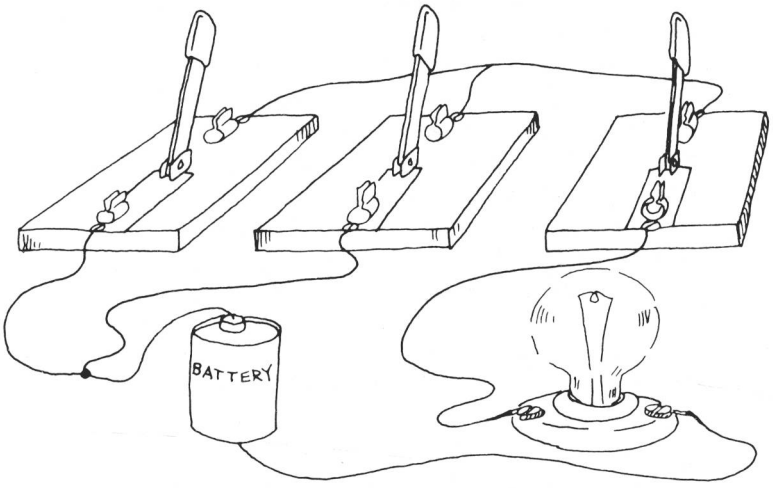


Figure 2 The basic circuit

Tell me what this circuit does.

Q 26 Let's see. The light bulb lights if the switch on the right is closed, and at least one of the two switches to the left is closed.

A Right. No magic, no "electronic brain", just a simple electrical circuit.

Now suppose I take the exact same circuit and lean some signs up against the switches and in front of the light bulb.

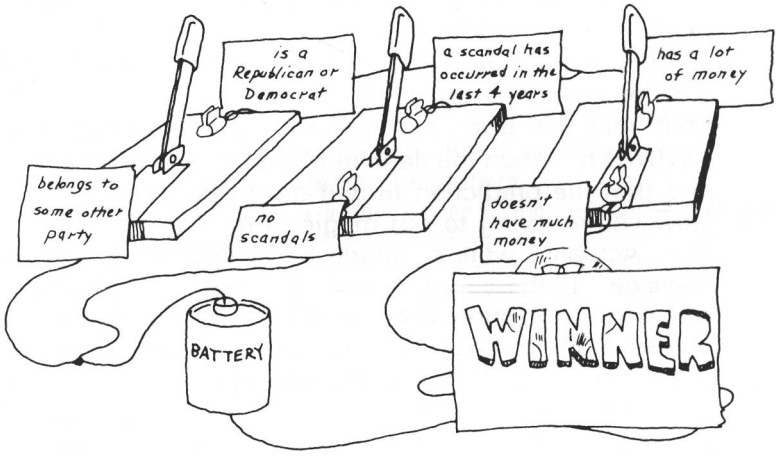


Figure 3 The Prognosticator begins to take form

In fact, just to get carried away, why don't we put the circuit in a nice looking box, and let the switches stick out of it, and put a fancy name on the front.

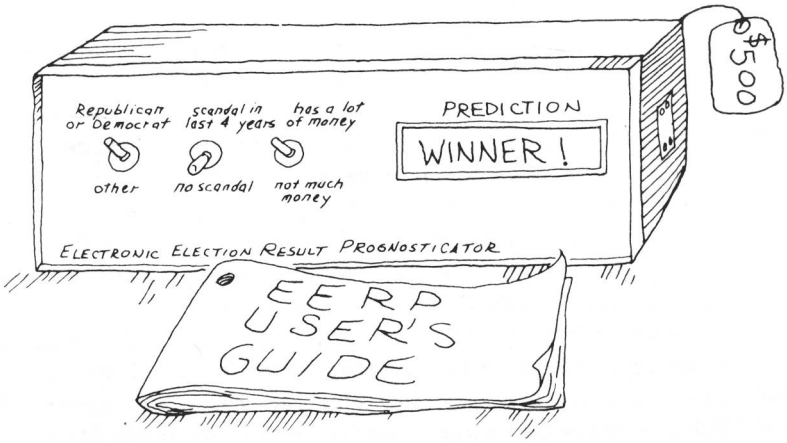


Figure 4 The EERP