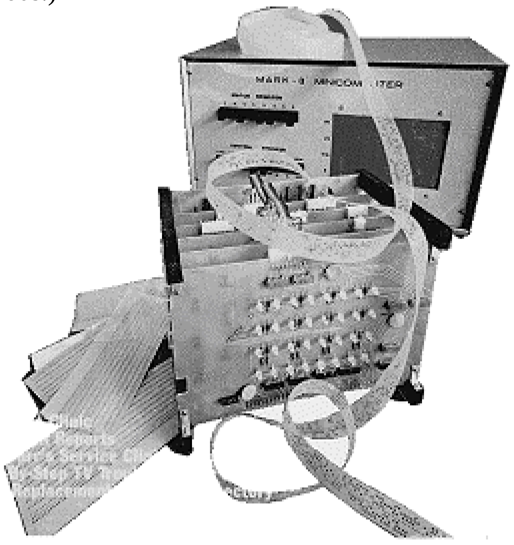
The Mark-8 Minicomputer

Part 1: The Beginning

...being some memories of the machine from its designer, Jon Titus
Last revision: March 21, 1999, Copyright © 1999 Jon Titus
Jon Titus
This shrine to the Mark 8 Minicomputer discusses the machine with its designer, Jon Titus. Jon was gracious enough to write up most of the data you see here, and also provide some of the diagrams and pictures. Jon is happy to discuss the machine with you, but please look for your answer here, first!
Jon Titus
Milford, MA 01757-1362 USA
jontitus@cahners.com Work
harrowsmith@mediaone.net Home

The Beginning

I had been fascinated with computers and electronics for some time, but I'll tell you about the immediate events leading up to the Mark-8. (I still have some high-school-years "computer" projects in the basement from the early 60s.)



I was a graduate student at Virginia Polytechnic Institute and State University (Virginia Tech) in Blacksburg, VA, working toward a Ph.D. in chemistry. My research involved using minicomputers such as the Digital Equipment Corporation (Maynard, MA) PDP-8/L. The people in my research group were using minicomputers to control chemical instruments and experiments, and the computers also acquired the experimental information and processed it. At the time, acquiring 1 Msamples/sec (6- or 8-bits/sample) was considered high speed! My project involved designing a high-speed interface that would acquire 6-bit binary values, stack two values to form a 12-bit word and then jam these words into the PDP-8/L's memory using direct-memory access (DMA). The PDP-8 family operated on 12 bits at a time.

I enjoyed using the PDP-8/L computer for research and I had fun fooling around with it on my own. I taught myself assembly-language programming and I enjoyed making the computer do "neat" things. The neat things were controlling an X-Y display, controlling an X-Y plotter, dialling a phone line so we could get access to a mainframe, and so on. I often wondered how I could get my own computer. I realized that buying a PDP-8/L was out of the question. The computer alone cost about $5000 (US) in the mid 70s, and I would have needed a teletypewriter, too, for an extra $1000! At the time, a group called the Amateur Computer Society had quite a few members, and as I recall, some members were trying to clone a PDP-8. I can't recall whether or not they had any success. In those days, although some functions were available on small-scale integrated circuits, the biggest roadblock was the memory.

The minicomputers of the time used core memory, which took quite a bit of external circuitry to properly drive the individual magnetic cores which actually stored information. Thus, memory was out of reach for almost all amateur computer enthusiasts, and so were computers themselves.

Intel

In 1971, Intel introduced the 4004 microprocessor and some support chips for it. This 4-bit device seemed revolutionary, but with only four bits, it also seemed limited to calculator-type applications. I studied the 4004 architecture, the instruction set and the peripheral chips that provided I/O and memory functions. I quickly decided against using a 4004 CPU as the basis for my own computer. Intel's first 8-bit processor, the 8008 changed my mind. It provided a nice instruction set, an interrupt, a small internal stack, multiple internal registers, and it could address a whopping 16 kbytes of memory. To put this in context, a basic PDP-8/L minicomputer could directly address only 4 kwords. We were ecstatic when our basic PDP-8/L got an upgrade to 8 kwords, but the extra memory took up about the same space as the PDP-8/L computer itself.

Intel provided a nice databook for the 8008, and I devoured and lived with it for several months. While driving to Canada for a vacation in 1973, I resolved to adapt a demonstration circuit that Intel published in its book and use it as the basis of my own computer. I wrote to Intel and asked for some sample devices because the 8008 chips sold for $125 each from distributors who carried the chips. (Not many distributors carried them, because they didn't know what sort of market there was for them.)

Although I used the basic Intel circuit, a made some modifications. The Intel circuit provided about 1 Kbyte of read-write memory (Intel 2102 chips) and about 2 Kbytes of PROM (programmable read-only memory, in 1702 or 1702A devices, each of which held 256 bytes of data or instructions). I expected to add more memory, so I built in decoding for all 14 address bits. In effect, I could expand the memory up to the entire 16 Kbytes that the 8008 could directly address. At the time, that seemed like a tremendous amount of memory.

Because the PROMs sold for about $35 each and I had no way to either program them or erase them, I decided that my computer--like a PDP-8/L minicomputer--would provide a set of front-panel controls and indicators. By using the switches I could load in short programs in binary using the switches. The programs I had in mind would be short "loaders" or "monitors" that would then let me use a keyboard or a display device, such as Don Lancaster's popular TV Typewriter, to display information.

Success and Failure

In retrospect, the front-panel design was the most innovative contribution I made. The design was simple and elegant, and it used programmable counters to temporarily store an address during front-panel use. The counters served double duty as latches during normal operation. The design relies on simple debounced momentary-action switches to increment the counters, load the counters from the switch register, and so on.

If you look at the front-panel circuit on later computers such as the Altair, you can see what a kludge the designer made of the circuitry. The Altair used the next-generation 8-bit chip, the 8080, which provided for much simpler control of the device. The front panel circuit, though, hardly takes advantage of the chip's capabilities and instead relies on all sorts of timing circuits to accomplish front-panel control. I don't mean to be mean about the Altair's design, but it was a bit of a kludge.

I also redesigned the clock circuit. Intel's design relied on four monostables to create a two-phase clock needed by the 8008. I used a crystal oscillator, figuring that I might have difficulty finding a dual-trace scope to use during prototyping and debugging. The monostable clock required a user to adjust four trimmer resistors to get the timing right. I built a breadboard version of the monostable clock and found that it didn't always start when power was applied. So, this experiment provided another reason to scrap the monostable clock circuit.

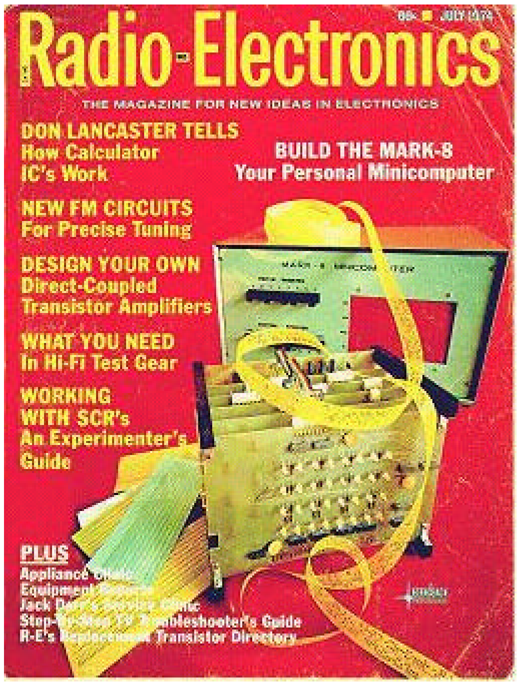
The biggest failing during the design was the input-port circuit. I simply took what Intel provided and duplicated it. I could have used an three-state or open-collector bus for input, but it didn't cross my mind to design the input ports this way. The Signetics chips used to multiplex the input data proved a bit difficult to find, so I suspect some people who built a Mark-8 modified the input ports so they could have more of them and so they wouldn't use the Signetics chips.

Prototype

My prototype of the entire computer was wired point to point using Veroboard. Experimenters from the 70s may remember Veroboard because many people favored it for experiments and projects. The board provided a grid of small holes on 1/10th-inch centers, which were just right for integrated circuits and other components that used, or could accommodate a 0.10-in. spacing between leads. One side of the Vero board contained narrow strips of copper that ran down the board from hole to hole in one long strip. If you wanted to break a run of the copper, you used a small hand tool which looked like a small countersink bit. The tool cut through the copper to isolate a specific run. The tool cut through only the copper and a bit of the underlying phenolic board material. The idea was to cut through the conductor, not all the way through the board.

I must have cut thousands of these "pits" in two large pieces of Veroboard. Luckily I caught on right away and used a drill bit in my Unimat (set up as a drill press) to cut the pits. The entire surface of both Veroboards were set up for 14-pin and 16-pin IC sockets. The exception was the 8008 chip, which required an 18-pin socket. Because IC sockets were expensive, I used Molex strips to create the sockets. The Molex strips provided individual, uninsulated IC socket pins in long strips on a roll. I cut the strips into 7-pin and 8-pin segments and soldered them in place to form inexpensive IC sockets. I figured that if the computer circuit didn't work, at least I could salvage the chips.

After making up the sockets, I plugged in all the chips except for the 8008. Having the chips in place helped me make the proper wiring connections. Without quick reference to what chips were where, I would have often lost my place.


JULY 1974
Radio-Electronics
THE MAGAZINE FOR NEW IDEAS IN ELECTRONICS
DON LANCASTER TELLS How Calculator IC's Work
BUILD THE MARK-8 Your Personal Minicomputer
NEW FM CIRCUITS For Precise Tuning
DESIGN YOUR OWN Direct-Coupled Transistor Amplifiers
WHAT YOU NEED In Hi-Fi Test Gear
WORKING WITH SCR's An Experimenter's Guide
PLUS
R-E's Replacement Transistor Directory

The prototype allowed two contacts in the Veroboard per IC pin, so I had to carefully plan my wiring. With only two contact points, all the signals had to run in a daisy-chain fashion, from one IC to the next.

After I completed wiring the prototype, I tested the various sections of the computer circuitry to be sure that these subsections worked properly. Without exception, they operated as designed. After thoroughly testing these circuits I carefully plugged in one of the three 8008 chips that Intel had sent me and kept my fingers crossed. I plugged in the power supply and the computer worked the first time. I couldn't believe it. I had built a lot of hobbyist circuits and kits, and almost nothing ever worked the first time. I was amazed that such a complex circuit worked. My first "programs" were simple ones that put the computer into some simple loops--jumping from one location to another, and back. I used the single-step function to make sure the operations were really running.

I did have one bent pin on an IC that causes a problem, but I quickly found the pin, which was rubbing against an adjacent pin and causing a short. The prototype was up and running about 2:30 in the morning and it was all I could do to contain my enthusiasm. I didn't sleep a wink the rest of the night. I thought about waking my wife, but decided against it. She had no interest in computers and would have had no interest in the fact that I had built a working computer.

Finally I had my own computer. The point-to-point wiring was neat, but the integrated circuits were so close together that it was difficult to use the computer for more than testing the basic design. That prototype contained three banks of 256-byte memory, so it let me experiment with less than a kbyte of memory.

Radio Electronics

After thoroughly testing the prototype, I talked with Larry Steckler at Radio-Electronics magazine about publishing information about the Mark-8 (it now had a name) as a construction project. I also approached Popular-Electronics magazine, but they showed no interest. Larry was a bit skeptical, so I designed circuit boards, got prototype boards, made a few modifications, and put the boards in a ready-made metal box to give it a professional look. Larry visited Blacksburg one day in late winter or early spring in 1974 to see that the computer actually worked. We had it set up with Don Lancaster's TV Typewriter, a keyboard, a digital-to-analog converter (DAC), and an oscilloscope. The DAC operated based on simple commands from the keyboard and it would change the shape of the waveform on the oscilloscope accordingly. I think my brother Chris and I had set up the DAC to produce a triangle wave, sawtooth waves, and a square wave. Generating a sine wave was a bit beyond our capabilities. We didn't want to load in by hand a sine look-up table, nor did we want to write a sine-calculator algorithm in assembly language for the 8008.

Based on his experience with us in Blacksburg, Larry agreed to publish the Mark-8 article, so I set to work writing it and I also wrote some extra experiments that R-E published in a separate booklet. The experiments covered how to use the computer with various I/O devices so that the builders--if there ever were any--could see how they could use the computer to do useful things.

I had to scavenge the prototype to obtain the switches and integrated circuits for the final version, so the hand-wired boards eventually went into the trash. People have asked why I didn't save them. Remember, I was a graduate student who was earning very little money at the time.

In the spring of 1974 I lugged the computer, including the external power supply to New York City so that R-E could have a photographer take several shots for the cover. I'm not sure of the exact dates, but it was just before the weekend in '74 when Virginia Tech's basketball team won the National Invitational Tournament.

Why Octal?

That's an interesting question, and one I hear every once in a while. First, the PDP-8 minicomputers broke down instructions into four 3-bit words, using an octal digit to represent each 3-bit group. The minicomputer had eight basic types of instructions, so the octal system served well to represent codes and addresses.
Second, although you can use almost any system you want to represent binary codes, why remember 16 codes in hexadecimal (0-9 and A-F) when the digits 0 through 7 work just as well.
Finally, the instructions for the 8008 microprocessor chip broke down nicely into 2- and 3-bit groups; XXYYYZZZ.
Here's an example that shows the beauty of using octal notation with the 8008. The processor has seven general-purpose registers, A, B, C, D, H, and L. A served as the accumulator. Registers H and L operated as general-purpose registers, but they also served as memory-pointer registers, thus the L, for the low-order eight bits, and H for the high-order address bits. These registers were coded in binary as follow:

- A 000
- B 001
- C 010

- D 011
- E 100
- H 101
- L 110

Note that at this point there's no register associated with code 111.
In the 8008, there's an instruction that lets you move the contents of any register to any other register. It's coded in binary as:
`1 1 D D D S S S`
The three D bits represent the destination register for the data, and the three S bits represent the source register that holds the byte you want to move. So, to move a byte from the C register to the E register, you need an op code of:
`1 1 1 0 0 0 1 0`
If you break this into octal groups you get:
`11  100  010`
or
`342`
. If you want to reverse the order of the transfer, the bits look like this:
`1 1 0 1 0 1 0 0`
or
`11  010  100`
which in octal comes to:
`324`
.
You can see how you simply reverse the digits in the octal number to reverse the flow of information. After a short while, a programmer understands that the 300-series instructions are all used to move information to and from registers.
Compare these two movements, register C to register E, and then the reverse, as noted in hexadecimal: E2 and D4
No pattern there!
Most of the other 8008 instructions broke down nicely into octal groups. It became easy to "converse" in octal codes and to quickly look at a listing of octal codes and immediately see what the instructions were supposed to do.
Interestingly, Intel continued with a similar format in the 8080 family, slthough the op codes changed and there were more of them. In 1976, we made up some nice cardboard slide rules that listed the codes for the 8080 and 8085 microprocessors. By sliding the cardboard insert you could quickly come up with the proper op code for any operation. We produced the cards in both octal and hexadecimal versions. In the 8080, the various conditional-jump instructions in octal are 3X2, where X varies from 0 to 7. The octal codes are easy to remember. In hexadecimal, the same codes are: C2, CA, D2, DA, E2, EA, F2, and FA. They don't exactly show an easy-to remember pattern.
I guess the computer scientists at Intel got hold of the 8008 and 8080 op codes and decided to force them into hexadecimal notation. One of the early Intel manuals used decimal coding, of all things, but the company quickly switched to hex. Too bad for the users. My colleagues and I stuck with octal for as long as we programmed with Intel and Zilog processors. And we taught using it, too. As far as I can remember, the students caught on quickly and were remembering op codes in no time. I can still remember octal op codes for the 8080 and when I fool around with assembly-language programming for the 8085 or Z-80, I automatically revert to octal. It's a snap.

Early Reactions

Well, the computer caused quite a stir in the hobbyist-experimenter circles. Remember that at that time there were no--or almost no--hobbyists who had their own computer. Here comes one that they can build for about $350. Experimenters aren't sure what to do with the Mark-8 at first, but lots of people want one. Keep in mind that 4-function 8-digit calculators cost about $100, and the first scientific calculator, the Hewlett Packard HP-35 costs $325 to $350.
I liken this to inventing the egg beater when no one knows you can eat eggs, or that eggs are good to eat. It can take years of missionary work to convince people to eat eggs and then the egg beater catches on. The Mark-8 came along at just the right time.
Several "user groups" started in the US, and one published the Mark-8 Newsletter, later changed to the Micro-8 Newsletter. One of the founders, John Craig, went on to become the first editor of InfoWorld. My copies of the Mark-8 Newsletter went to the Smithsonian years ago.

I can't recall any specific reviews of the Mark-8, because after all it was not a kit, but a collection of circuit-board layouts and information on what to do with them. Radio-Electronics magazine sold several thousand of the $5 supplemental booklets that they offered for sale in the magazine article. I had contracted with a circuit-board company (Techniques) in New Jersey to sell sets of circuit boards for about $50. As best I can recall, they sold a couple of hundred sets of boards. I got a royalty of the booklets and on the circuit boards, and I got paid several hundred dollars for the magazine article. The extra money was nice--I bought an IBM Correcting Selectric typewriter with some of the money. We didn't have word processors yet!

We didn't have email either, but I corresponded and talked with many people who needed a bit of help, or who had ideas for projects, improvements, and so on. One of the nice things about the Mark-8 was its robust electronic design. The computer worked flawlessly every time. It never received any criticism for a flawed design, circuit problems, and so on. Other later computers, in particular the MITS Altair (and the S-100 bus), had a reputation for severe electrical and electronic problems. I know of several people who never got their Altairs to run properly. Anyone who knows the details of the S-100 bus can tell you what a mess it was. I have never since seem a poorer bus design. I never received one complaint that the Mark-8 wouldn't run.

By the way, a mail-order electronics-parts company in Texas offered a kit of the hard-to-get components. I never received any royalty from them, but I didn't expect to. Anyone could put together a bag of parts and sell it, as far as I was concerned.

Plans For Improvement

I had no plans to improve it. I just wanted to show people that they could build their own computer. It was an educational project as far as I was concerned. After R-E published the design, I figured it was up to others to improve it, add to it, modify it, and so on.

One fellow in Syracuse, NY (Maury Goldberg?) asked me to send him the printed-circuit tape ups. Those are the layouts that eventually get photographed and reduced to form the basis of the copper traces on the printed-circuit board. He wanted to add edge connector "fingers" to the bottom of the boards so that people could plug them into a real bus. He was going to lay out a simple bus, too.

The original boards got connected by wires threaded from board to board along one edge. Soldering in the boards made it difficult to make modifications. But the board-to-board wiring proved very reliable and it was a lot cheaper that using gold-plated fingers on the cards and edge connectors on a bus. The Mark-8, after all, was a low-budget approach to getting your own computer.

Maury modified the tape ups to add the fingers, but for whatever reason, he never actually made any boards, nor did he devise the required bus. Later he returned the tape ups to me, but over time they deteriorated and I threw them out.

I'm Rich!

One thing you didn't ask that others often do is, "Why didn't you get rich?" First, there was no market for small computers, so no one was going to sell a lot of them.

Second, I did get rich, but not in terms of money. I'm rich because I enjoyed helping people start using small computers and I made them aware that they could get their hands in them and get the computers to do interesting things.

I'm rich because I met and talked with a lot of interesting people.

I'm rich because the Mark-8 helped launch me into a new career of developing educational hardware and software, technical writing, teaching, and editing. Overall life has been a lot of fun. Every once in a while I run into someone who remembers some of the early work my colleagues and I did the area of educating them about digital electronics and computers. Hearing them say that we helped them get started in a rewarding career makes me feel really good.

I wrote this in one sitting and spent little time editing it or polishing it. By the way, I'm not an electrical engineer. My college degrees are all in chemistry, but I have always been involved with electronics in some way.

People are welcome to drop me a note. I can't guarantee a reply, but will try.

Cheers,

Jon Titus
Milford, MA 01757-1362 USA
jontitus@cahners.com Work
harrowsmith@mediaone.net Home

Q & A

Part 4: Jon answers some Questions!

Last revision: April 5th, 1999, Copyright © 1999 Jon Titus

Most of the questions here were asked of Jon by Doug Salot. Some answers refer to Doug's fine website Blinkenlights.

How did you come up with the Mark-8 name? Was it in homage to the Harvard Mark I and its descendants?

I came up with the name on the spur of the moment. I was putting the decals on the front panel and realized that the computer should have a name. Up until then (late 1973) it existed just as the "8008 computer," and it had no formal name. Perhaps the "Mark" came from my experience in the military, or I may have simply picked it out of the blue. The "8" related to the number of bits the computer used.

What was the first software written for the Mark-8?

The absolute first software was a simple jump test, that if I recall correctly, went something like this (in octal):

```
Address    Instruction    Mnemonic
000 000    104            JMP
000 001    000            000
000 002    000            000
```

This simply caused the computer to go into a three step endless loop, which I did the first time I tried it. As I recall, the second test also used a JMP instruction and it went something like this:

```
Address    Instruction    Mnemonic
000 000    104            JMP     /Jump to
000 001    100            100     /this low address
000 002    000            000     /and this hi address

000 100    104            JMP     /Jump to
000 101    000            000     /starting address
000 102    000            000
```

Then I recall running a simple test that incremented a register in the 8008 and put out the binary bit pattern to an output port. I purposely placed an output port right at the front panel both for diagnostic use and for general-purpose output of binary information.

The LEDs in the prototype were tiny--about the size of a pencil point. I think they were MV-10s or something similar from Monsanto. They were very difficult to see, so I turned out the lights on my workbench and used the single-step controls on the prototype to step through the instructions, one at a time. Doing so confirmed that the computer worked properly.

What was the most interesting software written for the Mark-8?

That's difficult to say. My brother, Chris, and I programmed some interesting math routines that produced very pretty shapes on an X-Y display. We used two digital-to-analog converters to drive the plotter.

Any third-party hardware?

None that I can recall, although I did start work on a punched paper-tape reader, which I got to a prototype stage, but then abandoned as too complicated and too expensive.

Were the front panel and box plans included in the plans that Radio-Electronics magazine sold for $5? Or was the project just the boards?

The project was just for the boards. I thought that most people would build their own cases to suit their needs. The original Mark-8 metal box was one I had purchased for another project that never got built. I made the cut out for the

LED displays and found a piece of clear plexiglass that would fit into the space. I used transparent red glass dye, which came as a liquid to color the plastic. I placed a "dam" of masking tape around the edges and then poured the dye onto the plastic and let it level itself and then dry. It made a nice dark red filter for the larger LEDs used in the final version of the Mark-8.

Hal Chamberland (Chamerblain?) estimated the number of Mark-8 computers as around 1000 in 1975. Does that number sound right to you?

It sounds high. Of the top of my head I'd say several hundred sets of boards were sold.
I found my old tax forms and looked at the royalty amounts I received from Radio-Electronics and from Techniques, the company that produced the printed-circuit boards. It looks like Radio-Electronics sold about 7500 of the $5 booklets, which is far more that I would have remembered. About 400 sets of boards were sold.

Are you familiar with the work of Nat Wadsworth, who built a commercial 8008-based micro in 1973 and marketed it from his company, Scelbi?

I was familiar with Nat's published work. He produced some interesting books about programming the 8008. I got a lot out of those books. I think my original copies went to the Smithsonian with a lot of other computer publications. I didn't know about the Scelbi computer until right about the time that the Mark-8 came out. I recall seeing some small ads for the computer, but I never saw one of the actual computers and I don't know anyone who had one, though.
Another small "computer" came out at about the same time. I think it was called the Kenbak-1. It used one of the Texas Instruments arithmetic-logic unit (ALU) integrated circuits (SN74181?) to perform rudimentary processing operations. The computer stored a few instructions, which users set by pressing buttons on the front panel. I investigated the possibility of using Kenbak computers in a course taught at Virginia Polytechnic Institute (Blacksburg, VA). My colleagues and I decided that the Kenbak was interesting, but we couldn't use it as a real computer. Also, as I recall, the Kenbak had no provisions for any form of digital I/O, which we needed.
Intel produced a number of small "computers" at about this time, too, although they required a teletypewriter for I/O. The first one was a SIM-4 board that included a 4004 4-bit processor. Then Intel produced a SIM-8 board for the 8008 processor. They also sold a PROM-programmer board as a companion for it, but the set, including a nice blue box that provided connectors and some lights, sold for over $1000. I still have a set of those boards and one of the boxes, although I can't recall the model number of the box. I know that it required external power supplies. And it required on-board PROMs that contained a monitor program that controlled the teletypewriter. Very basic. The lights and switches weren't used to load program steps into the computer. I think they could generate an interrupt or jam an instruction into the CPU.
I used the basic SIM-8 circuit as the basis for the Mark-8, but with many modifications so that the computer could accomodate a real front panel that would give users access to the memory and let them control the computer.

Were you familiar with any of the earlier hobbyist groups, such as the ACS, organized around machines like the PDP-8?

Yes, and I was a member of the ACS for several years. I've covered some of this history elsewhere, so I won't get repetitious here. I did use PDP-8/L minicomputers in my reserch work at Virginia Tech, so I had a natural interest in the possibility of duplicating a PDP-8/L for my own use. The catch always seemed to be getting core memory. Many companies sold arrays of core planes, but few came with electronics, so they were difficult to control. Those few that came with electronics usually had only part of the electronics--the drivers, and not the control electronics. I spent quite a bit of time investigating core memory, but never got further than driving and sensing a single core--one bit!
I don't know if any ACS members ever did get a PDP-8 "clone" working.

I don't suppose you were influenced at all by the work of Edmund Berkeley were you?

No. I just looked him up on your web site (Blinkenlights) and found some interesting facts about him, but prior to that I had not heard of him. I am pretty much self taught in electronics. I got my start in "computers" by teaching myself about Boolean logic and numbering systems in high school. I built a 4-bit binary adder from my own design and I still have it in the basement. I also have a punched-card reader I built. It took 3-inch X 5-inch punched cards

and used a 7-bit code. That was in 1962-1963. I later improved the design by adding a better stepping motor to move the cards through the reader.
A friend of mine and I built some relay-based "learning machines" in 1962 and 1963 when we were seniors in high school, but they could only react to a few stimuli at a time, so after we had fun with them we put them aside and never tried to do anything practical with them. In relay circuits, power is the limiting factor. We needed many large 24-volt power supplies to drive our creations, and we just couldn't figure out how to build as many as we needed.
When I was in junior high school, or perhaps in early high-school grades, my Dad bought a Geniac "computer" for about $20. It provided a cumbersome switching arrangement of brass clips and shorting strips that had to be positioned properly to get the circuits in the right configuration to "compute." I can't remember what the circuits actually did, but I found it almost useless because it was so darn hard to wire and get working.
By the way, on your site you show a Heathkit analog computer. That was the second analog computer from Heath. Previous to that model they had a much larger one with a sloped front panel and many, many banana jacks and lots of potentiometers. Cost was about $1000, so they didn't exactly appeal to hobbyists. You had to know a lot of math and calculus to get much out of a computer like that. I think the computer could drive an oscilloscope and a strip-chart recorder so you could "see" the results of a computation which you wired up with discrete components.
A magazine in the UK, Wireless World, published a series of articles in the mid 60s. It required builders to use discrete transistors to build up operational amplifiers. I thought about building some of the circuits, but getting the special transistors and other components from the UK proved too expensive. My employer, Reed Elsevier, now owns Wireless Worls, and if you would like, I can do a bit of digging to see what more I can find about this series of articles.
Also, we need to give credit to Don Lancaster, who published a lot of digital IC projects in Popular Electronics in the late 60s. To a great extent it was Don's articles that convinced a lot of us that we could actually use these new-fangled things called integrated circuits. Many of Don's projects relied on DTL ICs, and later on TTL ICs. I can't recall building any of Don't projects, but I built parts of them--power supplies, clock circuits, lamp drivers, counters, and so on that I neede to construct some of my own projects. SouthWest Technical Products in Texas produced a lot of kits for Don's projects. Don lives in Thatcher, AZ, so you might want to contact him for his perspective.
Any who can forget Don't TV Typweriter, a device that could use a TV set to display alphanumeric information. I used one with the Mark-8 to display otal and hex codes for testing and debugging.
There was also a company in Colorado called Environmental Products that sold a lot of digital IC kits and parts. I may have a contact who could tell you more about the company, but it has been quite a while since I talked with him.