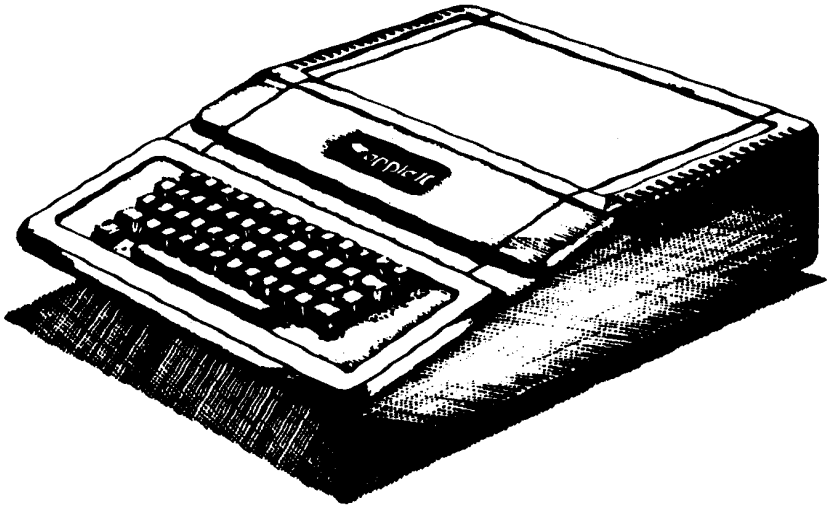
Apple 2 Computer Technical Information



Apple][Computer Family Information

Woz Interview:

The Apple Story — BYTE Dec 1984, Jan 1985

Document # 311

Ex Libris David T. Craig

"DTCA2DOC-311-00.PICT" 133 KB 2001-04-03 dpi: 300h x 300v pix: 2046h x 2850v

In this first part of a two-part interview, Steve Wozniak discusses his fondness for minimizing the number of required chips and the design of the Apple I and Apple II. (1984:12 pA67)

THE APPLE STORY

Part 1: Early History

CONDUCTED BY GREGG WILLIAMS AND ROB MOORE

An interview with Steve Wozniak

Steve Wozniak is the designer of the original Apple computer and one of the cofounders of Apple Computer Inc. Here Steve speaks at length on a variety of topics that range from the hobby activities that led to his design of the Apple I to current issues at Apple Computer. His frank comments and criticisms provide a glimpse into the workings of Apple Computer from a true insider's point of view. This is the first part of a two-part article.

BYTE: *We've heard that you designed the Apple I while you were working at Hewlett-Packard in 1975. Can you tell us what you were doing before that; what lead up to the creation of the Apple I?*
Wozniak: I have been interested in electronics and computers all my life. In my high school days, I studied TV circuits and I designed about 50 computers on paper, but I couldn't afford the parts to build them. In 1970, most people couldn't afford a monitor, so I designed video outputs that plugged into an oscilloscope and had the oscilloscope draw letters on the screen.

I had also been studying software. I wanted to know how to write compilers for languages like FORTRAN or BASIC, so I studied and kept notebooks. It was all self-taught and on paper, but I never got a chance to try it out.

I took three years of college toward a computer science degree back when only a few colleges were starting to offer it as an undergraduate curriculum. My third year had been at The University of California at Berkeley.

I originally planned to take a year off from Berkeley after my third year to earn enough money as a technician for my final year of college. But my career kept advancing. I was hired by Hewlett-Packard, made an engineer there, and I started developing better design expertise. I got into chip layout and things like that, and basically my career kept moving, my interests in life were changing, and it was too far to commute to Berkeley. I tried San José State but I didn't have the time available and it wasn't compatible with my first three years of college. It would have taken four more years to get my degree. So I never got a degree.

I was still an electronic hobbyist. I lost interest in minicomputers during the next three years, because I was doing calculator chips at HP and projects on the side at night.

I saw a Pong game in a bowling alley. So I went and designed my own. I designed the Breakout video game for Atari. I was just constantly involved in electronics as a hobby. At Hewlett-Packard we were pretty much just designing integrated circuits.

Around that time the Homebrew Computer Club got started and I happened to get directed to the first meeting by accident. I started discovering a bunch of high school kids who knew all about microprocessors and assembly language, and it was the stuff I had done up until three years before in my life. My whole life had been in minicomputers. All of a sudden, I started realizing microcomputers are the same as minicomputers, and I understood them.

BYTE: *You mentioned that you designed the Breakout game for Atari. How did that happen?*
Wozniak: Steve Jobs was working for Atari at the time. Nolan Bushnell was really annoyed because all their new games

Gregg Williams is a senior technical editor at BYTE. *Rob Moore is a hardware designer and a frequent contributor to* BYTE. *They can be reached at* POB 372, Hancock, NH 03449.

'I didn't want to pay to use somebody else's computer, so I decided to design my own.'

were coming out at 150–170 chips. He wanted low chip counts to reduce costs, and he had seen a version of Pong that I had done, that only used about 30 chips. He appreciated that. So he said if we could design a hardware Breakout in under 50 chips, we'd get 700 bucks; and if it was under 40 chips, we'd get $1000.

Atari didn't put us on a time schedule; Steve did. I had to do it in four days because Steve had to catch an airplane to Oregon. I was the designer—the engineer—and Steve was a breadboarder and test technician.

We gave them a working breadboard for it. My first design was 42 chips. By the time we got it working it was 44, but we were so tired we couldn't cut it down. So we only got 700 bucks for it.

The Apple I

BYTE: *How did you get to the point of designing the Apple I?*
Wozniak: I had worked my way up through software using a terminal on a local time-sharing system. Sometimes I'd call it from work, but I wanted to do it at home. I eventually designed and built a TV terminal and a modem so I could call this computer and play games. I was a true hacker. I started getting on ARPA-net and accessing computers all around the country. BYTE was the first magazine to get started, and I bought issue #1 at the newsstand.

I didn't want to pay to use somebody else's computer, so I decided to design my own. I wanted to have it all in one place, and I already had a terminal, so I was part-way there.

I sat down and wrote the BASIC first, and that took a lot more time than the computer design. Once it was written, I had to build the computer. I had designed a simple 6800-based computer, but the actual choice of processors was dictated by what I could afford. At the time, most microprocessors cost hundreds of dollars, but you could buy a 6501 over the counter for $20, and a 6502 was $25. So I bought a 6502, built the computer, and soldered one of my small TV terminals right onto the same board. It was a small computer with a small terminal, but it had good capabilities.

In our lab at Hewlett-Packard we had a desktop computer called the 9830 that ran BASIC. It was designed for the scientific community and cost $10,000 so it wasn't a personal computer, but you could run BASIC as soon as you sat down. That was my goal with the Apple I—you could sit down, turn it on, and start typing.

That was the main thing about the Apple I. Its characteristics were largely centered on its video terminal capability. In those days, the most common input/output mechanism was the ASR-33 teletype. It had been a standard for 10 years, and the minicomputer companies had only recently started using video terminals. So I had lots of experience using teletypes, and now I could do it in video with my terminal.

In 1975 video terminals were designed with shift registers because there were no cheap RAMs. You'd set up a bunch of shift registers and keep shifting them around to send text to the TV screen. So the Apple I was slow. It could type out 60 characters a second—one character per scan of the TV screen. My motivation was totally to save chips, not to add features.

BYTE: *Was the Apple I really a full-blown computer?*
Wozniak: Yes, but its features were a little bit different than the Apple II's and other personal computers that followed. It was slow, and it was text only, but it was a lot faster than the teletypes we were used to. They could only type 10 characters per second. TV terminals were just starting to get popular in those days.

BYTE: *Weren't they still fairly expensive?*
Wozniak: Yes, but I had to be cheap because I didn't have any money. I used the oldest, cheapest surplus parts I could find. Don Lancaster had written an article called "TVT1" for *Radio-Electronics*, and it was the big hobbyist article of the day. He had a humongous design that used tons and tons of parts and gates, but it was a poor design. I was into tight, clever designs to cut chips down, so I was flexible about the video timing. I knew from my high school days that TVs are designed with a lot of slop. Even if my timing was a little bit off, it would still work with most TVs and monitors.

BYTE: *So you didn't worry too much about precise timing?*
Wozniak: No, I was in a very hobbyist realm. I wasn't designing a product, just something that would work at home on my own TV. The computer itself used a 6502 processor interfaced to the terminal through a parallel interface chip called a PIA. It could also read a keyboard, so I bought a surplus keyboard that was advertised in an electronics magazine for $60. It emulated an ASR-33 teletype and did everything I wanted it to do, so I interfaced it.

Memory was my main problem. The only surplus RAMs available at the time were 2102s—1K static RAMs. I had the computer designed and the BASIC written, so I borrowed a 4K 2102 static RAM board from a friend so I could test it out. I got my BASIC going with that, but I wanted to use dynamic memories because you could cut the chip count way down.

Steve got intrigued with all these ideas and one day he asked me, "Why don't you use these new 16-pin dynamic RAMs?" I had looked at them in my work at Hewlett-Packard, but they were new and I couldn't afford any parts that didn't come my way almost free. I'm a little bit shy, and I didn't know any of the reps, but Steve just called them up and talked them into giving us samples. I jumped on it. I thought it was a great part because you could replace 32 chips on a board with just eight. It was a little more difficult to use because you had to multiplex the row and column addresses and that cost me one or two chips. But I was very happy because it was TTL [transistor-transistor logic] compatible and I could save a lot of board space because the parts were so much smaller. My whole goal was to make it as small as I could. I now had a little computer on one board about six by eight inches that

'Showing off Breakout on the Apple II at the Homebrew Computer Club was the most satisfying day of my life.'

I could take down to the club and show off. It was only 30 or 40 chips and it could run BASIC. People would look at it. It was unexpected.

BYTE: *So how did Apple Computer actually get started?*
Wozniak: Steve and I had both been going down to the [Homebrew Computer] club and giving out schematics for the computer and the terminal, even going over to people's houses and helping them build and test the computers out. Steve said, "Look, people are interested in what you've got. Why don't we make a PC board, have it silkscreened so they know what parts to plug in, and sell it at the club?"

We had about 500 members in the club, and I thought that maybe 50 people would buy it. It would cost us about $1000 to have the board laid out, and each board would cost us about $20. So if we sold them for $40 and 50 people bought them, we'd get our $1000 back. It seemed pretty doubtful. But Steve said, "Well, yes, but at least for once in our lives we'll have a company." So Steve sold his van and I sold my HP calculator to raise the money to make the PC boards.

Right away Steve got a big order from a local computer store to supply completely built computers. They ordered something like 100 units at $500 each, to retail at $666. It was unbelievable—a $50,000 order. We were in business.

All of a sudden we needed about $20,000 to buy parts. Steve went down to the local parts suppliers and we filled out financial forms. They looked at our purchase order, made phone calls, and really looked into our customer's credit. In the end, they gave us the parts with 30 days' credit. We had everything set up to build the computers and deliver them in 10 days, and it worked out great. We delivered the computers, paid off the parts suppliers, and only had to borrow $5000 from a friend, Alan Baum, and his father.

BYTE: *How many Apple Is did you actually sell?*
Wozniak: We manufactured 200 of them and sold all but 25 over a period of nine or ten months.

BYTE: *When was that?*
Wozniak: In 1976. It was first demonstrated running BASIC in late 1975, and Steve suggested forming a company in late 1976. We formed an official partnership in March. We had a third partner at the time who had 10 percent, but he sold out for $800 because he thought we weren't going to go anywhere except into debt, and he was the only partner with money. Ron Wayne

BYTE: *Just to put four or five stories to rest, where did the name Apple Computer actually come from?*
Wozniak: It came out of Steve Jobs's head, and he's a sort of private person, so I can't say what led up to it. He came up with an inspiration. He was working from time to time in the orchards up in Oregon. I thought that it might be because there were apples in the orchard or maybe just its fructarian nature. Maybe the word just happened to occur to him. In any case, we both tried to come up with better names, but neither one of us could think of anything better after Apple was mentioned.

BYTE: *Hewlett-Packard didn't want the rights to the Apple I. You designed it while you were working there. Did you offer it to them?*
Wozniak: Yes. There were a few of us in the lab at HP who were interested in microcomputers. We had proposed it to the lab manager. We sat down and had a meeting, and worked out on paper how we could have a little $800 machine that could run BASIC and connect to a home TV. Now this guy had been the project manager of HP's 9830 desktop BASIC machine, and he had been through a lot of these issues. He knew why this couldn't be an HP product, and he was right. Hewlett-Packard just couldn't do a hobby product—they just couldn't get into an evolving market when it was too young and unforeseen. So he turned it down and I got a legal release. A funny thing happened. After we started shipping Apple Is, our calculator division at HP started a little 8-bit processor project called Capricorn. I had already done most of the stuff they were doing, but they wouldn't let me work on that project.

BYTE: *Can you summarize the characteristics of the Apple I?*
Wozniak: It used the 6502 and included 8K of RAM by the time we put it out as a product. You could load BASIC into 4K of memory and you had 4K left over for your own BASIC programs. We supplied the boards completely put together and it had a video connector, but you still had to connect a video monitor on your own. You also had to get a keyboard and wire it into a 16-pin DIP connector. We built a power supply onto the board, but you had to connect two transformers for 5 volts and 12 volts. It had no speaker, no graphics or color. It could just display text at 60 characters per second.

BYTE: *The display wasn't memory-mapped?*
Wozniak: No, it was just a computer with a built-in video terminal. You could hand-wire certain addresses to read the keyboard, write to the video terminal, and select addresses for 4K memory bank 1 and bank 2. It had one external card connector, and we quickly filled that with a cassette interface card. It also had a side connector where we could theoretically add something later to expand it. The idea was just very small and simple, and it was all on one board.

BYTE: *Was there a monitor in* ROM?
Wozniak: Not really. In those days PROMs were very expensive, so I just used two small 256 by 4 PROMs to give me 256 bytes of ROM. I wrote a little program that would let you type in hexadecimal codes, examine a range of memory, or run a program at a particular address. Those three functions fit in 256 bytes.

THE APPLE II

BYTE: *How did you make the transition from the Apple I to the Apple II?*
Wozniak: We were selling Apple Is and we were just having fun and getting known. It was the most incredible thing we had ever done in our lives. I was still working at Hewlett-Packard and just moonlighting at Apple to test out boards, write more programs, and design a cassette interface so you could load BASIC in just a few minutes. [*Before the cassette interface, you had to type in the entire* 4K BASIC *interpreter by hand, in hexadecimal.* . . . R.M.] I had taken the machine down to the Homebrew Computer Club for its official introduction and pointed out its features. I got asked a lot of questions—people wanted to know if it could do other things.

At the time, Cromemco had just come out with a board called the Color Dazzler that did color graphics on S-100 systems. We had also had a demo, at the club, of a minicomputer running a display of a clock on the screen in color. During one of my times at Atari with Steve, I had designed a little seven-chip circuit to do color on a TV screen and it worked. So I started working on things that I wanted to add to the Apple I. I was thinking about clever color circuits and how to cut the chips down.

Remember, I had a computer with its own memory and a terminal with the display memory in shift registers. So I started thinking how I could combine parts into one—to somehow get them both to use the same memory. I finally worked out a design where a small part of the main memory was used as the terminal's video memory.

Eventually I got the whole machine done. It did everything that the Apple I did, except the display was in main memory so you could change any location on the screen instantly. I had built-in color, and EAROMs [electrically alterable read-only memories] were getting more common, so it had software built in to make it operate like a terminal. So it had color, it was very fast, and it was still cheap. In the end it was about half as many chips as the Apple I and it was many times better.

BYTE: *Was the designed-in color low resolution or high resolution?*
Wozniak: At first it was low-resolution color. Basically it could output characters in two different ways, either as text or as colors. I actually had to modify our horizontal video rates to be a little different from the NTSC [National Television Standards Committee] standards, because NTSC was not designed for digital video.

Once the computer was done, I started writing routines for the ROMs. I wrote terminal software so the characters would display in rows, from left to right and move down to the next line for a carriage return. I expanded my monitor routines to do more than just load hex, display memory, and run programs. I added the cassette routines because I knew they were going to be important. Eventually I started adding fancy features like a disassembler and the ability to split the screen into different windows. I would just go down to the club every two weeks and show off the latest routines I had done, because it was impressive. It wound up being a 2K monitor with graphics routines and everything included.

All my thinking from the Apple I days was carried over into the Apple II. Most people could not afford a color monitor, so it had to work with a home TV. A home TV could only display 40 characters across the screen, and I had to do what a TV could display. It was interesting because the technology defined the product rather than the market defining the product.

BYTE: *What caused the odd mapping of the graphics addresses?*
Wozniak: I had horizontal and vertical counters in the video circuits, and I had to map them into the display memory addresses. If you think about it, it's simple to have a perfectly linear address space. You take the vertical count times 40 and add it to the horizontal count. But it took three 7483 adder chips to do that, and I was looking for ways to save chips in the design. I figured out a couple of tricks that let me do it with just one adder, so I was able to save two chips; that helped others at the Homebrew Computer Club.

Also, I assumed that the user would just send characters to the screen through the terminal routines in the monitor. He would never see the odd addressing. I just wanted to play games and run BASIC. I didn't perceive that anyone would want to address the screen directly. Fortunately, only the vertical addresses were out of order, and there are a lot of easy ways to handle that so it doesn't slow things down.

BYTE: *How did you come up with the scheme where each byte only displays 7 bits, and when did the high-resolution graphics arrive?*
Wozniak: I had used the cheapest parts available, and a 5 by 7 character generator was the only cheap one in 1975. So my characters were five dots across and they fit into a 7 by 8 matrix. From my experiments with color and my experience with TVs, I knew how much time was available to display data during a TV scan. I also knew that my master clock had to be a multiple of the color-burst frequency, so I used 14.31818 MHz, four times the U.S. color subcarrier. So with that clock, I just counted how many dots per character would get 40 characters on the screen. I could have used eight dots per character and displayed 32 characters per line, but if I tried to display 40 characters of eight dots each it would have been too wide to fit on the TV screen.

One day I mentioned to Steve that I had noticed something interesting in the video addressing. I could make a little change by adding two chips, and then I could just shift each byte out onto the screen and we'd have hi-res graphics. I wasn't sure that it was worth the two chips because I was very chip-conscious. But Steve was pushing for all the features we could get, so eventually we put it in. At the time we had no idea that people were going to be able to write games with animation and little characters bouncing all around the screen. It was a neat feature, so we put it in there. Well, now I had a computer that had color in roughly the size of the bricks in the Breakout game I had designed for Atari. I had written a bunch of assembly-language routines to draw spirals and zig-zags of color, and I would take it into Hewlett-Packard to show the engineers. Sometimes they would sit down and say, "This is the most incredible product I've ever seen in my life." Many people were saying that when they saw the colors, but it was not in BASIC yet.

I wanted to write Breakout. I knew I could do it in assembly language, but I hadn't put any graphics commands in the BASIC yet. I knew how to add commands, since I had written the BASIC, so I put in a command to plot simple color

squares and soon got to a point where I could draw a brick wall. Then I did a ball and wrote some routines to make it bounce around. Now I needed a game paddle.

I wasn't sure how to sense the value of a paddle. I didn't want to put in 20 chips or anything expensive. So I found some little timer chips—NE555 timers. They were very cheap, but they could sense the value of a pot [potentiometer] by controlling their timeouts. There was also a larger one that was four 555 timers on one chip. So it was trivial. With a few resistors and capacitors I could read four paddles. As I got further into the game I needed sound, so I put a speaker in also.

Basically, all the game features were put in just so I could show off the game I was familiar with—Breakout—at the Homebrew Computer Club. It was the most satisfying day of my life when I went down there. I got a couple of high school kids to help me set it up, and I demonstrated Breakout—totally written in BASIC. It seemed like a huge step to me. After designing hardware arcade games, I knew that being able to program them in BASIC was going to change the world. All of that stuff is the essence of the Apple II.

BYTE: *We've heard that someone else helped you with the design of the I/O slots on the Apple II. Is that true?*
Wozniak: Yes. Many of the computers of the day had I/O card connectors, but they all required a bunch of address-decoding circuitry on each card. It was costly and required extra chips. I was into low-chip-count designs, and I wanted to have eight slots. So I was thinking of dividing part of the address space up between the eight slots and just using one decoder on my board to decode the eight different addresses. In that way, each I/O board didn't need three chips' worth of decoders on it.

Alan Baum helped a lot. I already had some address decoders on the board that decoded every 16th and every 256th I/O address. Alan realized very clearly that each card could have its own little 256-byte PROM on it and how they could all share a bank-switched 2K space. So I could just send two decoding signals to each card. Each board had 16 addresses for its I/O and 256 bytes of space for its PROM. Alan was the most constructive person in terms of realizing very clearly just how well the hardware-software interaction would work.

I saw one of Steve Ciarcia's articles in a recent issue of BYTE. He implemented a new voice chip design on an Apple II, and he said it was "because you don't have to decode it on the board." That was exactly the purpose behind the I/O decoding Alan and I designed. Oddly enough, he and his father were the ones who lent us the $5000 we needed in the Apple I days.

BYTE: *How did you raise the money to build the Apple II?*
Wozniak: By the time we did the Apple II, we had to build 1000 boards that cost $250 each. We needed $250,000 and we had nothing. Mike Markkula helped us write a business plan, and then he started perceiving that maybe this home computer market was really going to happen. He thought that home computers would hook up to your oven and refrigerator. Obviously this didn't happen, but he decided to join us as a third partner. ■

Through the years, Microsoft has steadily built on MS-DOS: version 1.25 added support for double-sided disks; version 2.0 provided for Unix-like hierarchical file structure and hard disks; versions 3.0, 3.1, and 3.2 added support for 1.2-megabyte floppy disks, higher-capacity hard disks, Microsoft Networks, and 3 1/2-inch floppies. With each enhancement, compatibility with preceding generations was an important—and restricting—consideration; it kept Microsoft from adding significant new functions to the operating system. What's more, the 16-bit 8086 is becoming obsolete in a world increasingly dominated by 32-bit microprocessors. Microsoft is attempting to overcome these limitations with a new operating system that harnesses the power of Intel's 80286: OS/2 — Eva White and Richard Grehan, BYTE Authors (1987:6 p117)

Powerful microcomputers are now relatively inexpensive. In designing OS/2, we attempt to maximize the machine's response and utility to its user — Gordon Letwin, Chief OS/2 Architect at Microsoft (1987:6 p118)

CONDUCTED BY GREGG WILLIAMS AND ROB MOORE

In this second part of a two-part interview, Steve Wozniak discusses Apple's first disk drive, the Apple III, his plane crash, and other topics. (1985:1 p167)

THE APPLE STORY

PART 2: MORE HISTORY AND THE APPLE III

An interview with Steve Wozniak

Last month, Steve talked about his background, the evolution of the Apple I and II, and the early days of the company. In this part, the conversation switches to various aspects of the Apple II design, later personal history, and Steve's thoughts about the personal computing industry.

SWEET-16

BYTE: *One of the more interesting things in the Apple II ROM was your 16-bit pseudo-machine called "Sweet-16." How did you come up with that?*

WOZNIAK: While I was writing my BASIC, I had been thinking about ways to save code. There were several places where I had to handle 16-bit pointers with an 8-bit processor, and that was pretty awkward.

So I decided to write a little emulator and implement a 16-bit machine that could interpret pseudo-codes and implement registers 0 to 15 in the 6502 base page. It ran about 30 times as slow as 6502 assembly language, but it saved tons of code every time I used it in a program.

BYTE: *Did you actually use it in your Integer* BASIC?

WOZNIAK: No, I never had the time to reimplement the BASIC to use it. But I did use it in later years to write things like BASIC renumbering routines totally in Sweet-16. It was easy to mix Sweet-16 code with assembly language.

BYTE: *Isn't Sweet-16 still used in Apple* DOS *and ProDOS editor/assemblers?*

WOZNIAK: Yes, it's used in EDASM [the Apple Tool Kit 6502 editor/assembler], mostly in the editor portion. Randy Wigginton wrote EDASM. He's worked here since before we even had a company. Lately he's written the Macintosh word processor—MacWrite. He's done a lot for the company, and he's used Sweet-16 in several things he's done.

THE DISK DRIVE

BYTE: *Can you tell us a little about how you came up with the Apple II disk drive and how you ended up picking your form of group-coded recording?*

WOZNIAK: The disk design was my most incredible experience at Apple and the finest job I did. I never really knew what a disk controller was or what it had to do. But at Hewlett-Packard I had looked through a Shugart manual to see what signals were used and what they did. There were signals to make the head step in and out and signals to cause magnetic flux changes. It was similar to audio recording, and I knew about that. It was like a signal on a tape where you write it and then you read it back. So I figured out a simple little circuit to write signals at changing rates and read them back. I didn't know how disk controllers worked, so I assumed that I was doing something totally different. Maybe it wasn't as efficient, but at least I could write some data and read it back.

Well, Mike Markkula was annoyed because the cassette tape was too slow. He had a favorite checkbook program, and it took two minutes to read in the program and another two

Gregg Williams is a senior technical editor at BYTE. *Rob Moore is a hardware designer and frequent contributor to* BYTE. *They can be contacted at* POB 372, *Hancock*, NH 03449.

PHOTOGRAPH COURTESY OF APPLE COMPUTER INC.

'Hobbyists are a tiny part of our market, but they're faithful to the company.'

minutes to read in his check files. He was complaining about this at a staff meeting, and I mentioned that I had this clever little five-chip circuit that could read and write a floppy disk. At the time, all the existing floppy-disk controllers were 40 or 50 chips, so I knew there must be something important that I wasn't doing.

I went off and tried to figure out what it was that I wasn't doing. One of our technicians had a North Star system, so I looked through their manuals. I read their schematics and figured out what every chip did. And I looked through their listings until I understood exactly what they were doing.

I was doing a lot more. I didn't even have to look at the sector holes, so I could use any disk drive, any floppy disk in the world. It was then that I new I had a really clever design.

The next week was Christmas vacation. Randy Wigginton and I spent the entire week, including the holidays, trying to get this disk reading and writing with a very simple operating system. We did the bottom levels of an operating system in that week. You could type R (for "read") followed by a program name like STARTREK, and it would load STARTREK into memory.

We were highly motivated because, at the end of the week, a show called the CES [Consumer Electronics Show] was starting in Las Vegas, and we wanted to go.

We worked all night the day before we had to show it [the disk drive] at CES. At about six in the morning it was ready to demonstrate. Randy thought we ought to back it up, so we copied the disk, track by track. When we were all done, he looked down at them in his hands and said, "Oh, no! I wrote on the wrong one!" We managed to recover it and actually demonstrated it at CES.

BYTE: *Had you been exposed to group-code recording before, or did you invent yours independently?*
WOZNIAK: The first version of the floppy-disk routines did not have group coding. I had followed the Shugart manual, which showed that you had alternate clock bits and data bits, so every other bit was wasted. I couldn't understand why it was necessary, but I started that way.

Then I came up with this idea to use coded recording. I knew the technical rule was that you could only have one or two zeros in a row. You could have either 4 or 8 microseconds between flux transitions. I didn't really know what group-coded recording was; I just knew that I could fit 13 sectors on a disk instead of 10. I had to write a program to take bytes off the disk, convert them to 5-bit chunks, and reassemble them into 8-bit data bytes.

It was a difficult routine to write. It was about a 20-hour job, and I'd work through the day for 10 or 12 hours and I wouldn't quite get there. The next day I'd come back and find out that I was starting exactly where I had the day before. This went on for almost a month. I was not quite getting the routines, and we were getting within a month of shipping the disk drives. Finally I stayed up all night until I got all five routines that had to work together done. So we were able to ship it the first time with the group-coded recording in place. Later, we changed the encoding method and stepped up to 16 sectors. That was DOS 3.3.

BYTE: *The Macintosh uses a custom chip called the* IWM—*Integrated Woz Machine—that does the same sort of recording. Can you tell us anything about that?*
WOZNIAK: My design was basically a little sequencer, or state machine. It used a PROM and a latch and cycled through various states depending on the input data coming off the floppy disk. The IWM takes that design and adds other features like the ability to go twice as fast—it can also do IBM format, double-density recordings.

BYTE: *It sounds like a fascinating part. Do you think we'll see Apple II owners benefit from it in any way?*
WOZNIAK: Well, it's our standard disk controller now, and it's cheaper than the older design. It's used in the Apple IIc.

BYTE: *Could you use it to get higher-density recording on an Apple II 5¼-inch disk?*
WOZNIAK: No, because the disk drives themselves aren't certified for double-density recording. You need heads with the proper gap, and they're more expensive.

BYTE: *You're a former hobbyist. Could a hobbyist buy one of these chips and a spec sheet and start playing with it?*
WOZNIAK: I don't think Apple would give out the spec sheet. I totally disagree with that policy because I'm very respectful to the hobbyists. They're a tiny part of our market, but they're loyal supporters and faithful people to the company. If they had a spec sheet, they could start playing with it and figure out a lot more incredible things that we never planned it to be used for—even using it as a communications channel from Apple to Apple, Macintosh to Macintosh. There are a lot of great tricks you could do with that little part. It's a beautiful random I/O device that has too many things that have not been taken anywhere.

PERSONAL DETAILS

BYTE: *In 1981 you were in a plane crash and you left Apple for a while shortly after that. How long did it take you to recover from the accident?*
WOZNIAK: That was in February 1981. For about five weeks I had a type of amnesia that prevents you from forming any new long-term memories. After I recovered, people would show me pictures of myself in the hospital, playing games with my computer with my face all battered up. They would tell me stories of how I tried to sneak out of the hospital to visit my wife, Candy, or how I went to parties and rode my motorcycle. I didn't remem-

ber any of that. I had all of my old memories, but I'd forget new things from one day to the next. Finally I came out of it one night, but I never got those memories back.

BYTE: *Why did you leave Apple?*
WOZNIAK: We had a hundred engineers at that point, and I was no longer really important to the company. I didn't want to be a manager; I was just an engineer, and I wasn't really needed there. But I didn't feel comfortable going to Steve Jobs or Mike Markkula and saying I wanted to take off. The plane crash was a good excuse. After five weeks of amnesia, I simply didn't go back. I decided that if I was going to take a year off I might as well finish college. It was the hardest year of my life.

BYTE: *We've heard that you went to* UC–*Berkeley and had some run-ins with your instructors. Could you tell us about that?*

WOZNIAK: I was going under an assumed name—Rocky Clark—so they didn't know who I was. I took computer science courses, economics, statistics, and a few other courses.

My computer science courses were interesting, but I have to criticize them a little because they taught only specific problems with specific solutions. You spent your time memorizing standard problems and solutions and then tried to recognize variations of them in the tests. You weren't supposed to explore new avenues or try things that nobody else was doing. You were only supposed to learn the proper answer. They thought that you could be trained to know all the problems and the standard solutions. Once you learned them all, you could solve them. It was wrong because they weren't really teaching you to solve problems—they taught you to identify them.

My economics course was interesting also. We had a socialist TA [teaching assistant] who taught us that companies made money by cheating the consumer. All the kids in the class thought that companies would make a lot of profit if they could figure out a way to cut the costs of a product down, to make it cheap and screw the consumer.

I contrast that with the way we did things at Apple. Every product design decision was based on what consumers wanted, what would compete the best, what they would buy. We tried to do what customers wanted, in our best judgment, and give them high-quality products.

So I would stand up in class and argue about what the TA was saying. After a while he started telling me to shut up, or that he would kick me out if I interrupted him again. Apple was the greatest business success in history, but I couldn't tell him who I was.

BYTE: *So you came back to Apple after about a year. What would you say is the most important thing that you've worked on since you came back?*
WOZNIAK: There isn't too much. When I came back I started getting a little bit involved with this division's management, but it was unofficial. Officially I took the title of Engineer. Mostly, I've stayed involved with the Apple II because that's where I've got the most to offer.

Because I'm a founder at Apple, I could take almost any role I want, but I've tried to avoid the newest, most far-reaching projects because there are other capable people to do them. I try to stick to small projects where I can sit down and handle them myself.

Hard disks are likely to remain too expensive to become standard equipment outside the office. Prices have plunged in the last three years, but hard-disk systems still cost at least $1500. More often the prices are closer to $2500... The mechanics required to rotate the disk at very high speed while the magnetic head floats microns above the disk are not simple and the manufacturing process cannot get much less expensive — Phill Lemmons, West Coast Editor (1983:3 p8)

CURRENT WORK

BYTE: *Do you have anything interesting going on now?*
WOZNIAK: There are not many individual projects in this whole business. The floppy disk may have been the last one for Apple. I have got a few projects that I'm working on now, but they're not all going at once.

THE SPREADSHEET

In late 1982, A.P.P.L.E., a national Apple users group, offered an astounding value to its members. For $22.50, A.P.P.L.E. members could purchase a fully functional VisiCalc-like spreadsheet program called "THE Spreadsheet." At the time, various versions of VisiCalc and other popular spreadsheet programs were selling for between $200 and $300. THE Spreadsheet offered most of the features found in VisiCalc and added a few of its own. What's more, it was coauthored by the now legendary Steve Wozniak. Mysteriously, THE Spreadsheet was available for only one month, after which it was permanently discontinued. Since that time, original copies of THE Spreadsheet have become one of the few legitimate microcomputer collector's items. In the following segment, Steve describes how THE Spreadsheet came about.

WOZNIAK: It started because Mike Scott was having difficulty negotiating with Personal Software [now VisiCorp] about VisiCalc. They just wouldn't do what he felt was right as far as giving Apple breaks on price or doing new enhancements to it. So he came by one day and he said to me, "Would you like to do a VisiCalc?" It wasn't even known as a spreadsheet then.

I was really scared. VisiCalc was the only spreadsheet for a long while, and none of us had really been in business before. Was it legitimate to go out and do your own spreadsheet? Or was it equivalent to just copying and ripping somebody off? I didn't want to get close to this whole thing because I didn't want it to look like I was copying somebody else.

Mike got Randy Wigginton and Randy said, "Sure. I'll do it." I said I'd do the arithmetic routines because they were general—something that could be used by any program—and I had some good algorithms that I had picked up at Hewlett-Packard that I wanted to implement.

So we started working on it and, a little ways in, Randy finished it. He wrote 4K of code in a couple of weeks, and I was shocked that he came in so quick; I hadn't even started mine. So, I was getting to work on my arithmetic routines and my first demo was almost ready but still a few days off.

At the time, everyone was trying to get the Apple III into production. Mike Scott was camped out in the Apple III building, forcing the things that had to get done to get it finished and delivered on schedule.

He was in a bad mood. Everywhere anyone ran into him in the company, he had a sour face—no laughs, no jokes. And because Randy finished his first part of the spreadsheet and I hadn't finished my first part, he'd always come to me. The way he was talking to me, I was afraid I was going to get fired. He was in a *bad* mood.

It was the scariest time of my life at Apple. I was really getting badly addressed just for not being on time with something. I had three more days to go, and I didn't dare run into him once more because I was already way overdue. So I said, "I've got to get him in a good mood for three days."

Well, he's a Star Wars fan. He gets the T-shirts the cast gets before they get them because he knows who prints them. So I had a friend of mine call his secretary saying, "This is George Lucas. Is Mike Scott in?" Of course he wouldn't leave a phone number, but he said, "I'll call back." It actually worked. Mike was in a good mood for a few days, and I got my routines done.

Anyway, we finished the spreadsheet, and Apple started looking it over. Originally it had been defined to be as accurate as VisiCalc, and we were so much faster than VisiCalc it was ridiculous. The exponential routines that I really wanted to write for so long, because I had a good algorithm, were 30 times faster than VisiCalc's.

All of a sudden we started hiring experts on precision and things like that, and they started redefining the project from what Mike Scott had originally defined. They added a lot of features that Randy had to implement. This made life very difficult here at Apple, so we finally decided, "No, it won't be a Special Delivery Software product [a line of software offered by Apple a few years ago]."

After a long, long time, Randy went to *Call-A.P.P.L.E.* [the magazine put out by A.P.P.L.E.]. One night we were out to dinner and he said to Steve, "Well, I'm going to sell that to *Call-A.P.P.L.E.*" or "I'm going to give this spreadsheet to *Call-A.P.P.L.E.*" Steve said, "Well, that's good" or something like that. It was not official or formal. Randy went ahead and hit on it and gave it to *Call-A.P.P.L.E.* It was sold for $22.50—really cheap. We didn't want any money out of it. It was just the old Homebrew Computer club spirit of "Give to help others."

It was a good product, and it had been delayed well over a year. If Apple didn't want it, they should have just gotten rid of it, and it would have been out a year sooner. As soon as it popped up from *Call-A.P.P.L.E.* and was advertised, Apple got in a real huff about it. They forced *Call-A.P.P.L.E.* to only sell it to members, and only one copy per member. So Apple basically put it out of business—only allowed it to be sold for one more month.

It's really funny, because how did Apple get started? I designed the Apple while I was at Hewlett-Packard. Hewlett-Packard has a policy of deciding about these things very quickly. They would check with their legal department and other divisions and decide very quickly. So they gave me a formal release on it quickly because they didn't want it. But here was Apple, which didn't want it but wasn't going to release it.

If you use much Apple II software, you'll be surprised when you take the spreadsheet that Randy and I wrote and boot it in. It boots in so fast, you can't believe what's going on.

BYTE: *Is that because there's no copy-protection on it?*

WOZNIAK: No. Believe it or not, I used those routines that I did for Apple II Pascal.

BYTE: *You, Randy Wigginton, and Guil Banks were listed as the authors. Who is Guil Banks?*

WOZNIAK: Guil is right here in this building. I did the boot stuff, and he had written a lot of the other fast disk routines in there. He did all the routines that managed the disk and read and wrote DOS files appropriately without using DOS.

This was basically an all-volunteer project that we did all on our own time. And it was really neat, so we decided that we'd give it to the rest of the world.

The language Fifth is one of them, but I haven't written it yet. I'd like to combine the best features of BASIC, FORTH, and Pascal, and leave out the worst ones. You could have the formal structures of Pascal, the immediateness of BASIC, and the extensibility of FORTH. In FORTH or Logo, new words become part of the language, and you can use them immediately. It's really helpful for debugging. In Macintosh Pascal, you can define a procedure and run it immediately. I also want some level of globalness, like BASIC. I don't want to always have to declare a variable before I can use it. I like variables with scope, but I'd like undeclared variables to be totally global.

Another project is an operating system, like the one on the Macintosh, only a bit different and a lot more relational.

I also have a hardware project that I'd like to do, a personal computer based around consumer video sources like TVs, videotapes, and videodiscs. It would switch them around, synchronize them, and mix them, sort of like a little home editing studio. I think it's possible because memories are getting so cheap. You could hold a frame from each of your video sources in memory and let the software accommodate the sync variations. There are a lot of new chips available that do NTSC modulation and demodulation, so there ought to be a minimal chip solution.

My main interest is still the Apple II—the home computer we started with. I'd like to see Apple do more with speech. There are some really inexpensive speech chips now, and that's the way the rest of the personal computer world is heading. I think we've been deficient in that area.

SPECULATIONS

BYTE: *Are you thinking about using the new 65816 processor for anything?*

WOZNIAK: We're thinking about it and doing some R&D with it, but I don't know if we'll use it. Anything we do has to be compatible with the Apple II. If we found out that the 65816 wasn't, it would be a serious question. It's too new a part right now.

BYTE: *How is its performance compared to the 68000?*

WOZNIAK: It should be available soon in an 8-MHz version that will beat the pants off a 68000 in most applications, and in graphics applications it comes pretty close. Some of the Macintosh people might disagree with me, but there are ways around most of the problems they see. An 8-MHz 65816 is about equivalent to a 16-MHz 68000 in speed, and a 16-MHz 68000 doesn't exist.

BYTE: *The Apple II family has been a great success, and many innovations have come along to extend its life. What things do you think were responsible for its success?*

WOZNIAK: For three years we were one of the biggest business successes in history. We had three years of fantastic business success, and lately we've had three years of sort of dismal business. We've grown, but any additional revenues have just replaced the stock dilution, and the price remains about the same.

During that three years there were two main factors that led to our success—our floppy disk and VisiCalc. Out of the original home computers, which included the TRS-80 and the Commodore PET, ours was the only one that had enough memory to run VisiCalc. VisiCalc and the floppy disk sent this company into the number-one position.

We were also very faithful to our users—we tried to support everybody. When we changed over to floppy disks, we still supported cassettes heavily. When we moved up to the Apple II+ with floating-point BASIC built in, we still supported the original Apple II.

Lately our strategies seem to be changing. When we come up with a new enhancement, we start moving away from the prior version much more quickly. This could be harmful to our good relationships with a lot of our faithful users.

The Apple IIe is a good example. Most new software that really uses the features of the Apple IIe won't run properly on an Apple II or II+. A lot of good software for the Apple II+ won't run on the Apple IIe. It's an incompatible world.

BYTE: *How do you think the Apple II family will be extended and improved in the future?*

WOZNIAK: There are obvious areas. We're always trying to come up with better combinations of features and still reduce the cost. We're looking into improved processors like the 65816 we discussed before. Video resolution is always improving. We're trying to increase speed and the amount of memory in the machine because it's critical to certain applications.

The IBM PC is very successful, and it had no competition from Apple for the last three years because we made sure that it didn't. We would not allow the Apple II to compete in that market for three years.

PCSD is the most difficult division in the company. [Steve is referring to the Personal Computer Systems Division, responsible for the Apple II and Apple III products.] We're hamstrung by the need to be compatible; Macintosh isn't.

So we will continue to make improvements and produce new machines, but they'll always be compatible. With the Apple IIe we went to 128K, 80-column display, and double-resolution graphics. We came out with the IIc portable.

APPLE VERSUS IBM

BYTE: *What did you mean when you said that the Apple II was not allowed to compete in the IBM market?*

WOZNIAK: Apple has never really supported the Apple II in the business market. If you walk around a trade show and look at the software running on the IBM PC, you'll see that most of it is a step above what's possible on today's Apple II. They have more RAM that's easily addressed and better access to hard-disk drives. Programs like 1-2-3 cannot be easily implemented on a 128K Apple II, but IBM has a capable machine for that level of software. Our machine has to be able to address more memory and

'I have positive and negative feelings about things Apple has done.'

handle larger disk drives before we can really start to compete with equivalent software because IBM beats us in capability today. We need better screen resolution, more memory, and better speed. A 16-bit processor would help, although "16-bit" really doesn't mean that much.

Whatever we do will be compatible, because we don't want to alienate our existing customer base. If we can come up with the right machine, then we can start to talk about some really good software and compete well with IBM. Even if we do, the new features won't really be used right away.

BYTE: *Have you said all that you meant to say about where Apple is vis-à-vis IBM and where you personally feel that Apple has made its mistakes?*

WOZNIAK: I have both positive and negative feelings about things Apple has done, but I'm always honest. There is one real mistake that Apple made, in my opinion, and this is very subjective. It is symbolic of what could happen to IBM with its PCjr.

We had become a huge business success in 1979. We had really made it with our floppy disks and VisiCalc, and it looked like we were going a long way. So we decided it was time to start putting together a real company, a big company. We needed to start staffing up and hire a lot more engineers. So we set the Apple III project in motion.

The executive staff felt it understood the Apple II market. After VisiCalc, it was perceived that 90 percent of all Apple IIs sold were going to small businesses. Only 10 percent were going into this home hobby market that we originally thought was going to grow to be billions. Originally we were a home hobby computer. Now, suddenly, small businesses were buying Apple IIs, and they wanted more features—an 80-column display, lowercase characters, maybe more graphics modes and colors, and more memory. These were all the things that one product, VisiCalc, led to.

According to any research we could dig up, many people were buying the Apple II for small business because it had a disk drive and it could run VisiCalc. These weren't the people I was closest to, so I kept my mouth shut because I was only one out of the staff of 15. So we started staffing up and building an organization and a management structure based around doing this new product—the Apple III.

THE APPLE III

We had some problems getting things done on time with the Apple III because of (a) our lack of experience as a group working together, and (b) not being able to predict project lengths well enough.

We started hiring intermediate levels of managers. Sometimes the managers were getting hired at a rate that added a completion date to the project faster than it was getting completed.

Around this time we started developing a perception of market separations—good strong separations between products so they don't overlap. You don't want to design a product that competes heavily with your own existing product. I claim that's untrue. What you really don't want to do is design a product that doesn't offer any more than your existing product.

So we started setting up strong boundaries. The Apple III would be our 80-column business machine and have 90 percent of our market. The Apple II would be our 40-column home/school machine and have 10 percent of our market. The entire executive staff was sure that once the Apple III was out, the Apple II would stop selling in six months. I felt really down, because this 10 percent were my friends—the hobbyists and home users.

The Apple III hurt Apple in many ways, but it was a very well conceived product. And because we were so successful with the Apple II, we decided to build in an Apple II emulation mode to take advantage of all the software that was out there. The emulation mode did get built in, but, because of our concept of market separations, it was a very limited emulation. While our Apple II customers were adding 80-column cards and 16K RAM cards to their machines, we actually added chips to the Apple III to prevent access to many of its features during emulation mode. In emulation mode you only had access to 48K of memory—you couldn't use the 80-column display, the extra graphics modes, or the extra memory. The emulation mode wouldn't even run much of the existing Apple II business software, and there wasn't much Apple III software available.

Originally, we planned to deliver four applications with the Apple III—word processing, a spreadsheet, business graphics, and a database program. Steve Jobs's thinking at the time was, "People don't really want to buy a computer. They don't want to know about microprocessors or cards or buses. They want to buy VisiCalc—they want to buy a solution." So we were going to provide the four major solutions. But because we were having problems managing the Apple III project while we were building our management structure, we were only able to deliver our operating system—SOS—and VisiCalc, which was done by Personal Software [which later became VisiCorp].

The Apple III shipped very late and had 100 percent hardware failures. This is very subjective, and some people might disagree with me, but I think we were trying to be too pure. We wanted to do it on one PC board, not two, and it didn't fit on one PC board. So we got a company that could put three traces between two IC pins, had them do the PC board, and 100 percent of the Apple IIIs failed.

The Apple III is really very good, but we spent three solid years keeping the Apple II down, and now it's finally being allowed to grow in that direction.

We've come out with ProDOS, which is a major improvement, and the ProFile [hard disk] is available for the Apple II now. It's a good start. I think they are going to find out that allowing the Apple II to get there will improve the whole Apple image.

The III will do very well in its established vertical markets forever, but it really won't make the huge success we thought it would. It may have the best chance it's ever had. ProDOS is the best way to get someone closer to SOS. We did SOS three years ago, and the rest of the world hasn't caught up or come close to it. Macintosh went in a different direction, so I can't compare the two systems.

'We did SOS 3 years ago, and the rest of the world hasn't caught up.'

BYTE: Is SOS *really that good?*
WOZNIAK: I think it's the finest operating system on any microcomputer ever. It's the greatest thing in the world, but I wish we gave out listings of it.

BYTE: *Wasn't it the first commercial system that actually had installable device drivers?*
WOZNIAK: Yes, the Apple II uses device drivers in ROM, on the I/O cards, but they won't work on the III because they depend on specific memory locations. On the III, you load a device driver into RAM for any device you add to a system, so it's infinitely flexible. You can always change it, correct bugs, and it's clean. But it's more difficult for outside manufacturers, because they have to supply a disk and instructions on how to update your system. It would have been the best of both worlds if you could plug in a card with a ROM on it and if the first byte in the ROM were a 12, for example, the system would recognize that the ROM holds a device driver and link to it automatically. That would have made it much easier for the card manufacturer. It would have been really easy to allow both techniques, but the Apple III engineering group didn't want to do anything the way it was done on the II. Marketing just didn't have enough leadership and control.

Anyway, having the Apple III as a failure didn't really hurt the company much. The Apple II was still very healthy, but for the next three years the Apple III hurt the company tremendously because everyone in Apple knows how great a machine the Apple III really is. It's a very clean machine, it's easy to use, and it's really been organized right with the operating system.

Unfortunately, we made it very difficult for anyone to get access to the insides of the machine. We had hired some very bright people who figured, "This is the right way it should be done. So we'll give out enough information to do this and we won't give them any more, because they might try to do something they're not supposed to do." The right way for one person is not the right way for another. We closed that machine up to where somebody could have a very difficult time finding out how to add their own I/O drivers. We did not make it easy for the outside world. We thought we wanted all of the markets for ourselves.

You have to let the end users develop their own standards. You've got to give them the freedom to discover how they're going to use an operating system, what sort of things they're going to buy. And if you're really right and have provided a good solution, that's where they're going to settle. The thinking on the III was very much like a religion in that it could only be done one way—our way. We made it very difficult for outside developers, instead of providing all the information as we did with the Apple II.

BYTE: *Has that attitude changed now?*
WOZNIAK: No. It's still the most negative thing in our whole company, and it will be for years.

I think that when a new market evolves, like personal computers did, there's a period of time when you've got to let the world go in all random directions, and eventually it will subside because it wants standardization. Then, once it's obvious what the standards are, they should be heavily supported by the manufacturer. You can't try to dictate a standard.

THE APPLE II AND THE APPLE III

When we came out with the Apple III, the engineering staff canceled every Apple II engineering program that was ongoing, in expectation of the Apple III's success. Every single one was canceled. We really perceived that the Apple II would not last six months. So the company was almost all Apple III people, and we worked for years after that to try and tell the world how good the Apple III was, because we knew.

There is a lot to somebody's perception or image of a machine and how good it is. How many of my friends have them? How many people in the world have them? The Apple III was a failure the first year as a product—it had a bad image. When you give a bad first impression, you can go for five years trying to overcome it.

If you looked at our advertising and R&D dollars, everything we did here was done first on the III, if it was business related. Then maybe we'd consider doing a sub-version on the II. To make sure there was a good boundary between the two machines, anything done on the II had to be done on a lower level than on the III. Only now are we discovering that good solutions can be implemented on the II.

The Apple II was kept out of the business market to keep the III going, to give our users only one choice. We wanted to make the III the success it hadn't become and that it deserved.

Unfortunately, we made sure that the Apple II was nowhere close to the market that the [IBM] PC took. We made sure the Apple II was not allowed to have a hard disk or more than 128K of memory. At a time when outside companies had very usable

schemes for adding up to a megabyte of memory, we came up with a method of adding 64K to an Apple IIe, which was more difficult to use and somewhat limited. We refused to acknowledge any of the good 80-column cards that were in the outside world—only ours, which had a lot of problems.

At one point during the Apple III development, I wrote some fast disk routines for our Pascal system on the II. And I got a lot of flak from Apple III engineers who felt that they shouldn't go on the Apple II. Nothing on the II should be allowed to run as fast or faster than the Apple III. That was the thinking that stuck with the company for three solid years.

It was unfortunate the way things worked out, because we probably put $100 million in advertising, promotion, and research and development into a product that was 3 percent of our revenues. In that same time frame, think what we could have done to improve the Apple II, or how much could have been done by Apple to give us products in IBM's market.

BYTE: *Are you putting more resources to work in that direction now?*

WOZNIAK: Yes, but things don't change in six months. In the Apple III we had a beautiful machine, and we spent a lot of money to try and emphasize that. We were trying to force the world to finally accept the machine because we knew just how good it was.

The PCjr had a poor initial reception like the Apple III did. It came out in the wrong month—the month when Macintosh was going to be perceived as the leader. The PCjr was perceived as an uninteresting product.

They may try for three years to overcome its bad first impression. They may put a lot of their corporate resources into trying to promote the PCjr and lose sight of the PC. The PC will keep selling because it's been accepted and there are a lot of aftermarket companies out there selling software and hardware for it. But if IBM neglects it enough, we may have a chance to turn it over on them in the next few years if we have a better product. ■

Both RISC and its predecessor, CISC, are commonly credited to IBM. The first CISC machine was probably the IBM 360 mainframe, which was created in 1964. The 360 made extensive use of microprogramming, building instructions out of series of microinstructions that were in turn stored in ROM within the CPU.... RISC began in 1975 at IBM. John Cocke, an IBM Fellow, was working with a team to make a very large telephone switching system.... The outgrowth of their efforts was the 801 minicomputer built in 1979 — Phillip Robinson, Contributing Editor (1987:4 p143)

Two of the most widely used methods for designing CPU control sections in microprocessors, minicomputers, and mainframes are microcode and hard-wired logic — Phil Koopman, BYTE Author (1987:1 p235)

The [PS/2] Model 80—IBM's first 80386-based system—will compete head-to-head with Apple's powerful 32-bit Macintosh II — BYTE Editorial Staff (1987:6 p102)