Apple Lisa Computer
Technical Information

# Apple Lisa Computer: Interview with Designers (BYTE February 1983)

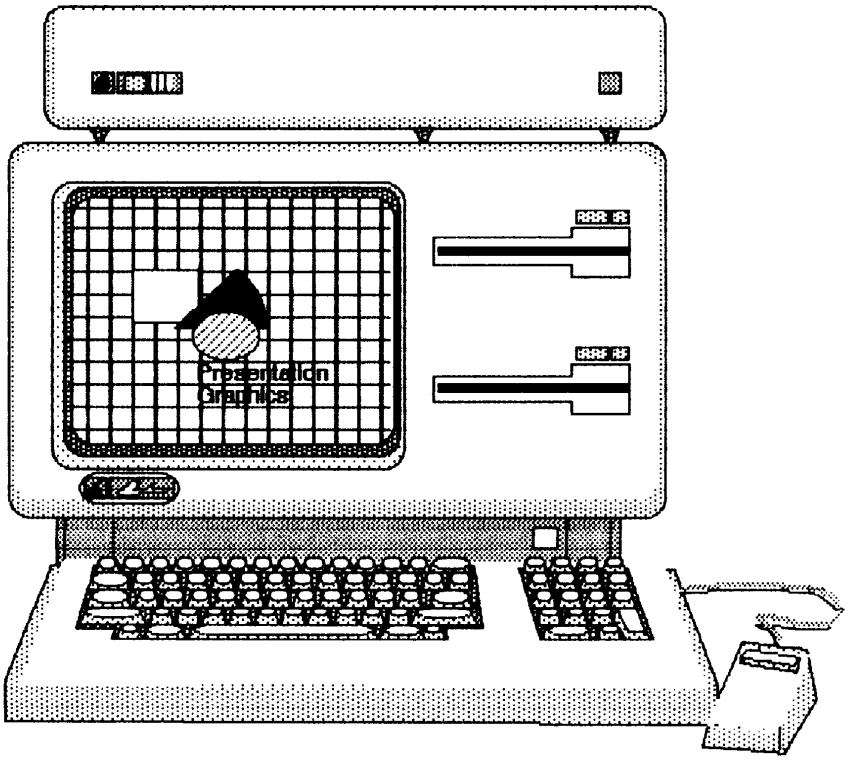
Lisa Computer:
1983 - 1985

**BYTE Interview**

# An Interview with Wayne Rosing, Bruce Daniels, and Larry Tesler

*A behind-the-scenes look at the development of Apple's Lisa.*

Chris Morgan
Gregg Williams, Senior Editor
Phil Lemmons, West Coast Editor




Apple Lisa Personal Computer 1983 - 1985

*Of the more than 90 members of the Apple engineering staff who participated in the Lisa project, Wayne Rosing, Bruce Daniels, and Larry Tesler are three of those who were most responsible for its final form. Rosing, formerly of the Digital Equipment Company, oversaw hardware development until Lisa went into pilot manufacture and then assumed responsibility for technical management of the entire Lisa project. Daniels and Tesler were responsible for Lisa's systems software and applications software, respectively. Chris Morgan, senior editor Gregg Williams, and West Coast editor Phil Lemmons interviewed the three at Apple's headquarters in Cupertino, California, last October.*

**BYTE: Tell us how you staffed the Lisa project.**

**Tesler:** In software, we drew mostly experienced people from other companies and very few people straight out of school. Even the ones we took out of school generally had lots of job experience. In fact, one time I surveyed the applications group and found an average of nine years' work experience in software. When we looked at résumés, we tried to find people with several years of experience in development. We made exceptions if someone had specialized in something we were interested in or was a top student who also had good summer experience. We wanted an experienced team because what we've been doing is a very major software effort. It's very complex, and there's such a large body of software to crank out and make reliable that it takes experienced people.

**BYTE: When did you do the hiring?**

**Tesler:** The project went through phases. There was some design and some implementation when the project first started two and a half years ago, but we hired most of our software people about two years ago. In three months, we hired most of the software staff, and then they spent several months learning about the machine and designing their particular parts of the software. The bulk of the programming started about a year and a half ago.

We had to spend quite a long time just building a team—people who had a common view and could work together. We drew people from different companies with completely different backgrounds and tried to do something that nobody in this group had ever done. Some of us had done parts of it before. We were developing everything in parallel: the hardware, the operating system, the applications, the manuals, the details of the user interface. We did have a sort of fundamental philosophy, but having to do everything at once means you're never sure when you're going to get what you need from the person who does whatever you need next.

**Daniels:** I think communication is the key there. If you have that many things going in parallel, you spend a lot of time communicating so each of you knows what the other's doing and can depend on each other.

**Tesler:** It took a while to work out those channels. It was rough at the beginning, but it's pretty easy now. Our progress was gradual. I think I'd call it team-building. Some of the things were hard to do in an organization that's thrown together like this. But once you've got a team built, it's a valuable asset. Of course, we were doing technical work all along, but in a sense we spent a year building the team and a year building the product. Now when we build something else, we can do it without the team-building step.

**BYTE: What about project security?**

**Rosing:** We tried to be as secure as we could without creating a discouraging atmosphere for people to work in. Within the group there has always been total information transfer, and we've kept lots of machines available. People have been able to take machines home with them. There was always the risk of losing a Lisa in a burglary, but we had a rule that the floppy disk had to be kept separate from the machine. We felt it was worth risking a theft to gain the increased productivity of people working at home. We've been very fortunate; we haven't lost one machine.

**BYTE: How did you schedule the project?**

**Tesler:** People made estimates, but it was difficult. All the estimates were conditional—"If the hardware is here by a certain date and the operating system is frozen and I have the user-interface definition and I can get some assistance from people who have the right sort of experience, then I can do it in this many months." But none of the ifs were ever really possible. People were really hesitant to make a firm date because there were so many contingencies. We did come up with schedules all the time, but they were myths.

**Daniels:** Getting Lisa to market has been a dream, a goal that we all have. Although we're willing to make compromises to get Lisa out expeditiously, the dream of what we're trying to achieve is the major thing.

**Rosing:** We had this dream of what we wanted to do, and I think over time we recognized that we couldn't achieve some of the goals. We'll have to take care of them later. We've taken the attitude that Lisa is going to be good and we're not going to sacrifice the integrity of the product for scheduling. We wanted to make a very balanced set of decisions, and so everything, as I say, just started to come together. The floppy disk works well, the mouse works well, the hardware works well, the software is beginning to come, and now we're cranking to get this first release out. But we won't let it be compromised because of scheduling.

**Daniels:** Part of the difficulty was that both the user interface and the internals—the architecture—of the software are revolutionary. Getting that architecture designed and built was a big scheduling problem. Once we'd done that, we'd built the foundation. Now building the applications is much smoother and has been much easier for us to predict.

**Tesler:** We didn't know if some of the things we started would work at all, like the way the dot-matrix printer is used and even the way the letter-quality printer is used to print the graphics.

**Daniels:** No one had ever done that before.

**Tesler:** Theoretically, it ought to be possible, but it had never been done, and the manufacturer of the printer didn't believe it could be done. It had to be possible in order for this product to do what we wanted, but no one could predict how long it was going to take. When we hired the printer people we told them to do it in two months. It took them a year and a half, but they did it. And then the high-density disk drives are new technology to Apple. A lot of the concepts in there had never been tried before. That was one of the biggest risks. And Apple not only built disk drives for the first time but built *revolutionary* disk drives.

**BYTE: What makes them revolutionary?**

**Rosing:** One of the major things we did was to vary the speed of the disk as you change the track position, so the drives keep constant area density, and that gives them a greater capacity. Second, we used microstepping algorithms on the stepper motor so that if a head gets off track because of changes in humidity and temperature, the intelligent controller can hunt and find the track. So we have much better interchangeability, with much higher density, and we're getting approximately 50 to 60 percent more data on that disk by good systems engineering. Some of the competitive units have a greater capacity, but we think the error rate ultimately suffers. We wouldn't tolerate a serious error-rate problem.

**BYTE: How does the error rate compare with double-sided double-density disks?**

**Rosing:** As for hard-error rates, we're talking about $10^{-12}$, and that occurs after so many bits that it's hard to measure. But we're quite delighted that the measurements are impossible to take. Basically that means the errors are low.

**BYTE: Did you work more than 40-hour weeks?**

**Tesler:** Each engineer set his or her own schedule. Some engineers work something like Monday through Friday from nine to five. Others work all day at the office, then go home and work all night there. And what an individual engineer does may vary from time to time.

**Daniels:** These people have pride. They set their own milestones and they want to meet them, so they'll put in extra work to do that.

**Tesler:** We decided a long time ago that since the project would obviously go on for more than a few months—a couple of years—we couldn't have this constant pressure on everybody, because people would just crack.

**BYTE: As individual designers, do you feel that your signature is on that machine?**

**Tesler:** I think that's true of everybody in the group. Even people who have been with us for only a few months have something in the Lisa that they can look at and say, "That was my idea; that's my code." It's really a group effort. Even marketing got involved in the design effort in various ways, particularly in user-interface issues, product design, packaging, and the style of the manuals. The whole division really got involved.

**BYTE: When did you decide to incorporate all the fundamental applications into the system software?**

**Daniels:** At the very beginning. Some applications weren't decided until later, but the integration, the way it all fit together, was a goal from the very beginning.

**Rosing:** As a matter of fact, we cut out a few more things because we just didn't feel we could manage a project that large. Then we added a couple things back in as we became more comfortable with the development cycle. But we've basically been operating on the same goal for the past two years, with very little change of direction.

**BYTE: What was the sequence in the early days? Did you decide what the project had to look like to the end user, and then what software was required, and then. . .**

**Daniels:** Then hardware. In fact, we spent the first six months hammering out the user-interface docket. We had that completely specified before we really started the applications. I think the key to success here is to know where you're going before you start.

**Tesler:** The hardware, the operating system, and the applications were all developed somewhat in parallel, but there was a definite cause and effect. The people who designed the hardware had to make decisions, for example, about whether the disk drive should have a door that you flip open or a button to push, that kind of thing. The designers focused on that aspect of the user interface even before the rest of the user interface. They didn't want the user to be able to accidentally pull out a disk when it was being written on or something. So some decisions were made even before the hardware was designed. There have also been hardware revisions. The first Lisa hardware was here when I came, over two years ago. It's gone through several. . . how many revisions since then?

**Rosing:** About four. Each one's been an iteration. We discovered a few things in the early hardware that wouldn't work well. We just took them out because we couldn't do them properly. The rest has mostly been a matter of fine-tuning Lisa so that it's very manufacturable and very reliable.

**Tesler:** Each time they go through a cycle, the people working on user interface get another crack at it—"Since you're going to revise the hardware anyway, why don't you. . . ?" Or the people doing the operating system say, "The memory-management unit needs to be more general, and since you're redesigning the hardware anyway. . . " So we were able to get in some hardware revisions. Also, that keyboard you saw yesterday is not the final one. After user testing, and

because of needing to support the European market, we determined that we really needed a couple more keys on the keyboard, so we made a major change in the keyboard layout.

**Rosing:** One of the things about this project that's different is that, more than any other I've been associated with, there's a continuous loop for dealing with user issues. We've gone to the software and that has implied a hardware change. We synthesized a lot of different disciplines. The power-off button used to be a traditional button on the back of the machine, but we didn't want to encourage users to turn off their machines that way because if they left a document open, they would lose it.

**BYTE: Do you expect to find a little initial resistance to the fact that the machine doesn't actually turn off when you push a button? Do you think people are going to say, "Well, I know I can leave it alone now, but I want to make sure it turns off"?**

**Rosing:** Right. It does feel a little funny at first, but after a few times you begin to have confidence that the thing does turn itself off.

**BYTE: When you finally got the user-interface specified, did you have a brief description of it that everybody knew by heart?**

**Daniels:** It was about a 35-page document.

**BYTE: Thirty-five pages of specifications?**

**Tesler:** We have something called the User-Interface Standard, and it consisted of those things which would be common to all applications. Also, the year after that document was published some revisions and some changes were made, and as we built applications we found that they had even more in common than we envisioned. Then we would adopt those things as part of the standard.

**Daniels:** Another thing we've done is user tests—taking our ideas and bringing in naive users and sitting them down and seeing what their impressions are. That has caused some changes, and I think that's all shown in the quality.

**BYTE: Where did you get your naive users?**

**Tesler:** Various places—the bulk of them were new Apple employees. We had a screening process. New Apple employees go through an orientation the first Monday morning they're here. We handed out a questionnaire to the new employees about their previous experience with computers, word processors, video games, and that sort of thing, and then what kind of work they did. Someone in our training department screened all those vitae. I'd go in and say I needed three user test subjects this week who have no word-processing experience but who are secretaries or accounting people to test out our Lisa Calc. She'd go through and pick out some candidates and I'd pick the ones I wanted, based on their experience for whatever test I was trying to run. We had about 50 tests this year in engineering to test out the software.

**BYTE: The fact that you responded to the tests speaks well for the end product. The changes in the keyboard, for instance. How recently did you decide to change the keyboard for the final time?**

**Tesler:** There were several changes. Those from the user tests had to do with changing the numeric pad so it had the arrow keys on it so you could move around in the Lisa Calc table. Those tests were run around January [1982], I think.

**Rosing:** January, and in March we decided to make the change.

**Tesler:** That was just key-cap legends that had changed. The other change has to do with the number of keys on the keyboard and was primarily for the benefit of international sales, although it did improve the user interface in terms of the positioning of the Enter key and the Extended Character option key, which gives you extended character sets. Those were all done around the same time.

**Rosing:** The interesting thing is that we were at the stage in the program where the decision to make even what sounds like a simple change takes six months to percolate through because it's not a simple engineering change—it's manufacturing, tooling, documentation.

**Daniels:** We made one legend change in June or July—the Apple key. When was that?

**Tesler:** July, and it's just now showing up.

**BYTE: A legend change?**

**Tesler:** You saw two keys that said Command on them. The new version has only one, and instead of saying Command it has a picture of an apple on it. The reason is that the key's used as a shortcut to choose a menu command. If you look at a menu, on the right you'll see this little apple symbol and a letter. If you hold down the Apple key and the letter, you get the command. We couldn't find any way to symbolize the Command key that would fit nicely in a menu and be recognizable to people. We tried and tried. Finally we decided that the apple looked nice and had a nice sound to it—"Apple X," "Apple R"—and it keeps Apple in the mind of the user instead of "control" or something else. It's a symbol that everybody using this machine will recognize instantly, so we decided to put it on the key as well as on the screen. To finish the artwork in time to get the machines to test users in time to get responses, and so on, the change had to be in by a certain date. The decision was made only hours before the deadline.

**BYTE: Are there going to be two Command keys without legends on them?**

**Tesler:** No, only one. We studied IBM and DEC and other keyboards and found that they all have just a single Command or Control key on the left-hand side. We also really wanted to put an Enter key on the main keyboard because we would like to be able to offer a configuration in which an alphabetic keyboard and a numeric keyboard are independent—for, say, a company that

does only word processing. Word processors don't need the Clear function, but they do need the Enter function, so we wanted to be able to have the Enter key on the main keyboard; that way, even people without a numeric keypad can hit Enter. Again, on IBM and DEC keyboards the Enter key is standard; on many of those keyboards, that's the standard position for the Enter key. So we decided to be more like other companies. The Enter key also gives us the option of removing the numeric keypad without losing an important function. And then the option keys were put on the side of those, and there we decided we did need two option keys, left and right, because they're used very much like shift keys for typing, and in Europe it would be very important to be able to touch-type foreign alphabets for international correspondence, mathematical symbols, and other special characters. So there were some trade-offs. We didn't want to just keep jamming two of every key on the keyboard, so we decided what the priorities were and ended up being fairly close to the industry standard. We have one Apple key, one Enter key, and two Option keys.

**BYTE: The user-interface design seems to have been difficult.**

**Tesler:** That was the hard thing that affected the most people. A lot of software and hardware engineering issues were very difficult, but they affected only a few people. Interface issues affected half the division because Training, Publications, Marketing, and the software person implementing the application all had an opinion. People like us who were overviewing all the applications had opinions, in-between managers had opinions, kibitzers on the side had opinions, too. Not everybody can talk about what gate to use in some circuit or what routine to use in some program, but everybody can talk about the user interface. So we had to accommodate all of these things. And it turned out that good ideas and good criticisms came from everywhere. We had to come up with some objective way to decide. That's why we established the methodology which involved user testing. We had a procedure for proposing changes, reviewing the changes, narrowing it down to a few choices, with certain criteria like consistency and parsimony. And then we actually implemented two or three of the various ways and tested them on users, and that's how we made the decisions. Sometimes we found that everybody was wrong. We had a couple of real beauties where the users couldn't use any of the versions that were given to them and they would immediately say, "Why don't you just do it this way?" and that was obviously the way to do it. So sometimes we got the ideas from our user tests, and as soon as we heard the idea we all thought, "Why didn't we think of that?" Then we did it that way.

**BYTE: Bruce, could you say something about the software architecture?**

**Daniels:** There's an operating system underneath that we built ourselves because we felt that the ones that were out there didn't quite meet our needs.

**BYTE: What does yours do that others don't?**

**Daniels:** It's not just what it does, but what it doesn't do. Some other operating systems are basically timesharing systems like Unix that have a lot of features that we don't need, and why take up extra space for that? We wanted a system that the user didn't have to be experienced to understand, and it had to be very reliable. It had to maintain the user's data and keep it there. It also had to support things like graphics, the windows that we have on the screen, the mouse, and so forth. We didn't really find an operating system that met our needs, so we felt we had to go build our own. We built the other features on top of this—the support for the windows, the support for graphics, the support for multiple fonts, the support for printing. It's really quite a rich architecture. At least half of the software is in this foundation software.

**BYTE: How large is that in bytes? How much code is in that foundation software?**

**Daniels:** Well, source code is something like 10 megabytes.

**Tesler:** Object code is about half a megabyte.

**BYTE: That's what's there before you put the application programs in—half a megabyte?**

**Daniels:** Yes.

**BYTE: After you specified the user interface, what list of hardware requirements did you come up with?**

**Rosing:** Well, the main list that was specifically user interface would be the bit-mapped graphics display and the resolution of approximately 700 pixels across in the horizontal dimension, the mouse, and the doorless disk drives with the eject button rather than an eject handle. They determined a lot of the hardware design. We had other user-interface considerations, though. We wanted to make the system very easy for its users to service—I presume you've seen it break apart. Servicing really is simple. It took a moderate amount of extra product cost to get that feature in there. And that's a part of the even more global user interface, how people perceive the whole system.

**BYTE: Why did you choose the 68000 microprocessor and what alternatives did you consider?**

**Daniels:** We thought its architecture was very broad and strong and would take us through the '80s, and we wanted that. We wanted something to support the graphics, and we thought that processor gave us what we needed then. The 68000 was a bit of a gamble because it was very young when we got on it. We were getting one sample at a time from the local Motorola engineer here.

**BYTE: Do you think the 68000 will be the dominant processor in the next few years? Is it going to overcome the 8088, the 8086?**

**Rosing:** I would speculate that for high-end applications with very computer-intensive, graphics-intensive needs, the 68000 will become dominant.

**Daniels:** But the 8086 has such an installed base going already, I think that alone would carry it. . .

**Tesler:** You mean numbers of actual units with the 68000 in it, or the number of different products?

**BYTE: Both of those questions.**

**Tesler:** Well, we're putting 68000s in the units we'll sell, so that will mean more units with 68000s. We expect to sell a lot of machines.

**BYTE: You've got a 68000 machine with a lot of memory in there, and not too much special-purpose hardware. Why did you decide to do it that way instead of using some versatile hardware chips, like the NEC 7220, for video display?**

**Daniels:** We're very much boosters of bit-mapped graphics, and in fact hardware support for bit-mapped graphics is pretty small. All you need is sort of a shift register. We thought the flexibility that would give us in graphics and the things we could do in user interface with bit-mapped graphics was well worth the price.

**BYTE: But doesn't the 7220 have bit-mapped graphics itself?**

**Rosing:** Well, there were a couple of practical considerations. The NEC 7220 didn't exist when we designed Lisa, although we knew it was planned. The second consideration was that the 7220 cost more than the TTL [transistor-transistor logic] hardware needed to implement the equivalent functions. And the third consideration was this: because we were able to interleave the memory and display cycles, we were able to essentially get data out of the memory at very little penalty. Using a 7220 would actually cost considerably more in terms of system performance. And there was one more consideration: with the 7220, you can't access the display memory bank when the chip is refreshing the CRT, and that limits the time you can access it to about 10 percent of what we have, which would drastically affect performance. We can access memory any time. For equivalent performance, we would have to use two 7220s, and that would push the cost and the "real estate" beyond what we have.

**BYTE: On the other hand, software doesn't get written overnight. . . there's a certain cost to that. You know, this is very software-intensive.**

**Rosing:** Most of the software that supports the graphics took three years to write, but no hardware in the world can duplicate what that software does.

**BYTE: Really? The software is faster than the hardware?**

**Rosing:** No, not always faster.

**BYTE: Its functionality is greater?**

**Tesler:** Yes. The graphics package lets us draw circles, rectangles, ovals, and rectangles with rounded corners. It also automatically handles clipping on nonrectangular boundaries. If you have one object over another, you can draw the one behind without splashing the pixels on top of the one that's in front. That's a . . .

**BYTE: A software revolution?**

**Tesler:** A very unusual capability, which no one else has in that general form. The other implementations are all either very, very expensive hardware—the $100,000 class—or in software, which isn't really that general and performs much much worse. There's nothing in the same class as our software as far as capability and speed. Of course, there is graphics software that's faster and hardware that's faster, but it doesn't have anywhere near this capability.

**BYTE: Do you have a Xerox Star here that you work with?**

**Tesler:** No, we didn't have one here. We went to the NCC when the Star was announced and looked at it. And in fact it did have an immediate impact. A few months after looking at it we made some changes to our user interface based on ideas that we got from it. For example, the desktop manager we had before was completely different; it didn't use icons at all, and we never liked it very much. We decided to change ours to the icon base. That was probably the only thing we got from the Star, I think. Most of our Xerox inspiration was Smalltalk rather than Star.

**BYTE: What does Lisa have that the Star doesn't have?**

**Tesler:** We're talking about graphics capability. You originally asked why we didn't use graphics hardware. Our graphics primitives in software are more general than the Star's, so they perform better. We have a faster and more general ability to draw on the screen a picture of multiple graphical objects in different shapes, to have one window that uncovers another, and to repaint just the parts that are uncovered.

**Daniels:** Look at the desktop managers of the Star and Lisa. With the Star, you can only put them at fixed places on the screen so you know they don't ever overlap. On ours, you can put them any place you want. It's that generality that allows us to have arbitrarily shaped things and covering each other up and . . .

**BYTE: Documents or forms, shapes, or anything. . .**

**Daniels:** Yes.

**Tesler:** Right. We have curves in it. Everything in the Star, you'll notice, is really rectangular, and our things can have curved edges and that sort of thing.

**BYTE: Another hardware question: How many microprocessors are in the machine, what are they, and what do they do?**

**Rosing:** Let's see. One to scan the keyboard, in the keyboard housing proper; a second one that receives the keyboard commands and keys up mouse events; the 6504 that controls two floppy disks; a Z8 microprocessor in the hard-disk controller—it's an intelligent controller; and then, of course, the 68000. That's five.

**Tesler:** Almost every major chip manufacturer except for one.

**Rosing:** And with only one exception all our I/O (input/output) cards have microprocessors.

**BYTE: You say that the magnetic read/write head in the disk drive is microprocessor-controlled in order to let it be more sensitive to variations in the alignment. Is that the 6504?**

**Rosing:** Yes.

**BYTE: What is the microprocessor that handles the keyboard and the mouse?**
**Rosing:** That's a National COPS. We tried to pick the processor that we felt was best for each particular job.
**BYTE: The memory is 64K-byte chips?**
**Rosing:** Yes, 64K chips.
**Tesler:** On the memory we have parity and. . .
**BYTE: What part of the memory is video memory?**
**Daniels:** Some area in the main memory can be the video.
**Tesler:** Any area at all. In fact, if you noticed yesterday in the demonstration, when we're developing software, we need debugging information to be displayed for the programmer, but we don't want it to come out on the same screen that the user is seeing, so we had this magic toggle we were hitting that flipped between two screens. There are really two different areas of memory with a bit map in each. The software can switch between the two to display each in turn.
**BYTE: But they're within the main memory?**
**Tesler:** Yes, absolutely. Anywhere in memory. Take any number of consecutive bytes and say that's the bit map.
**BYTE: Is anything else in main memory, or is the rest of it all available to the user? Is anything else mapped to the memory?**
**Tesler:** Oh, I see what you're saying—the shared memory. Shared memory with I/O is not main memory. The I/O memory is in the I/O cards.
**Rosing:** It's not in the memory, but it's accessed like main memory, from the 68000 bus.
**Tesler:** It's in the address space, but it's not in those 64K chips.
**BYTE: A certain address is really an I/O port, is that right?**
**Rosing:** Yes; it's the top physical address of the 68000.
**BYTE: Did you consider voice as part of the user interface?**
**Rosing:** Yes. We looked at it pretty hard and at one time in the early system we actually had a CVSD-based voice subsystem in the computer, and we took it out because we didn't feel it achieved the quality we wanted to have associated with this system.
**BYTE: What does CVSD mean?**
**Rosing:** Continuously Variable Slope Delta modulation. It's much easier to say alphabet soup. We've thought about voice; it's part of our network architecture and will appear in the future, but only when we feel the technology's right so we can be proud of what we offer.
**BYTE: That's both input and output?**
**Rosing:** Right. We look at voice as being three problems. There's store and forward, which is just moving voice messages around, like a glorified answering machine. Second is text to voice; and third, of course, is voice recognition, or voice to text. The last one's the hardest of all, but we look at voice technology as something we have to approach in a unified way.
**BYTE: What about the programmable serial ports? What chip is used there?**
**Rosing:** They use the Zilog SIO. That was one of the last major changes we made in the hardware design. We did it because we had two high-speed ports with less board space, and the Zilog SIO chip supports asynchronous as well as byte-sync and bit-sync protocols. We felt that made a heck of a lot more sense for the customer as the world evolves toward X25-type packet transmission. We didn't want to make the customer buy an I/O card to upgrade from async to bit-sync. We have only three I/O slots, so we're careful not to waste them on things we can put in the main machine.
**BYTE: Both serial ports can be bisynchronous?**
**Rosing:** Yes; they can be programmed any way.
**BYTE: And can this SIO function as a UART?**
**Rosing:** Yes. A UART/USART combination.
**BYTE: When did you know that you were going to have half a megabyte as standard memory? When did you know how much you were going to need?**
**Daniels:** It's always been a backward sort of thing. We had the capability for a full megabyte in the machine, and it was more a case of how much memory we needed to achieve our goal.
**Tesler:** The sales force wanted it to be 128K; the programmers wanted a megabyte. We negotiated.
**Rosing:** Since we were writing the code we got the megabyte.
**Tesler:** So the hardware people made it as big as they could in the address space, and then after some testing of the system we determined that half a megabyte was a reasonable compromise of cost and performance.
**BYTE: Do you expect the standard memory on other manufacturers' machines to jump dramatically after the appearance of Lisa?**
**Tesler:** Well, apart from its impact on cost, I don't think the amount of memory is a critical factor in deciding what machine you want to buy. If you're an end user, you should be buying a machine based on what it does for you, how easy it is for you to learn, how easy it is to use. Whether it has 12 bytes or it has 12 megabytes doesn't really matter to the end user, which is our marketplace. We're not selling the machine primarily to programmers who might care about that. End users have no idea which systems have more memory or less memory or one megabyte or one hundred thousand bytes. If other manufacturers are trying to match Apple, they should try to match us on ease of use and functionality and things like that. If they can do it in a small amount of memory, more power to them.

**BYTE: Doesn't it matter when you're doing something like dictionary software or when you want to read a dictionary into memory fast and proofread a document very fast?**

**Tesler:** Yes, there are certain functions where it definitely makes a difference. We have that in our Lisa Calc. In order to do rapid recalculation, the whole matrix really should be in resident memory, so we spent a lot of time coming up with a data structure that packed that data as tight as possible so that it would get as many cells as possible into memory, no matter what size memory there was.

**BYTE: Your version of BASIC will use more than 64K?**

**Daniels:** Oh, yes. We could have put less memory in it, but the performance would have been unacceptable. Unfortunately, some companies advertise machines that have less memory than anyone would ever reasonably buy. We haven't tried to do that here.

**BYTE: You didn't use less memory and fewer disk drives than would really be effective, and so on?**

**Daniels:** Yes, and I think when you look at the typical configurations that people buy of other machines, the cost is really not that different from the kind of costs we're talking about for Lisa. If the other machines get loaded up with disks and memory and the other kinds of things you want to run, then their prices will be comparable.

**BYTE: When you decided you had to have hard-copy graphic output that accurately represented the quality of the screen graphics, what choices did you consider before you did this amazing adaptation of a $600-$700 printer?**

**Rosing:** A wide range of options were being discussed, all the way from thermal printers to laser printers. We tried to identify what's critical in the marketplace. We thought there were two printers of first priority: a personal printer and one with letter quality. At the same time our sister division, the Apple II-III division, was evaluating the same two sets of printers. So we teamed up and did a survey of virtually all the printers that were available from every manufacturer who would have the volume capability to serve our needs. We did an extensive test and put about eight dot-matrix printers through their paces with really tough software. Quite a few of them just fell right off the table—it was clear that the quality wasn't there. Certain vendors were also much more responsive to fixing problems. So it really boiled down to two printers. Then, as we developed our printer software, the one we're using now—the C. Itoh—just far and away stood out as having the best mechanical design. You could put the dots where you wanted them repeatedly, and that's what we needed more than anything else in the world—good mechanical design.

**Rosing:** And a good price. Same for the letter-quality printer.

**BYTE: The printer you are using is from C. Itoh, but it's your own ROM and your own systems software that drives the printer through the ROM.**

**Rosing:** Correct.

**BYTE: What else can you tell us about the printer, especially the dot-matrix?**

**Daniels:** Mechanically it's just a raster device.

**Tesler:** A character generator is built into it; it has some capabilities. It has a single type style that can be stretched horizontally and vertically as it's printed, and it has what they call a graphics mode. They thought that would be used lightly, but it's what we use almost exclusively. And even within the graphics mode, there are two resolutions, low and high. High resolution is a lot slower. We wanted to offer the user all these choices.

**BYTE: So this is a custom design for you. . . custom changes?**

**Tesler:** Custom changes I would say, yes.

**BYTE: Did you say it sometimes prints out in character mode? I thought all of its printing when you were controlling it was using the highest resolution.**

**Daniels:** I think all the stuff you saw was done at high resolution.

**BYTE: For speed you can go to a different mode?**

**Tesler:** Yes; we're planning to offer the customer a way to get a quick draft using the character generator. Characters won't look quite the way they will in the final version, but you can get output in a hurry.

**Rosing:** The printer will have three different speeds and three different quality levels.

**BYTE: Do you have an idea where you're going next?**

**Rosing:** We have what feels like ten years' worth of backlog. We have a pretty good idea what we're going to do for the next few years.

**BYTE: What's that?**

**Rosing:** The thrust is to expand the level of integration within the applications and to add facilities to make it easier for more applications to be written outside of Apple.

**BYTE: Those facilities are the development toolkit?**

**Rosing:** Yes. The development toolkit is a key thing. And for a large part of the marketplace, adding network applications and data communications is very important. Last but not least is adding really serious database functionality to the system. If you add all that up, it's as big a task or bigger than what we've just done.

**Daniels:** In fact, almost as important as the team building that we've gone through is building up this foundation that we've used to create the six applications we've now built. The foundation is an amazing application machine. We and others outside Apple can build applications that are just amazing now, because no one has to rebuild the foundation. It's already there, in place, and we really hope to leverage off that in the future.■