SHARE SESSION REPORT

61	A002	The IBM Engineering, Scientific and Instruments Computer System	298
SHARE NO.	SESSION NO.	SESSION TITLE	ATTENDANCE
Applications Division		Robert Rosen	HDL
PROJECT		SESSION CHAIRMAN	INST. CODE

Harry Diamond Labs, 2800 Powder Mill Rd., Adelphi, MD 20783, (202) 394-2917

SESSION CHAIRMAN'S COMPANY, ADDRESS, AND PHONE NUMBER

The CS9000 Microcomputer

Marty Sandfelder
Atlanta Engineering NAD Scientific Support Center

This session will cover the new IBM microcomputer, the CS9000. What I will do today is tell you about the origins of this computing system, share with you a hardware and software overview and finish up by talking about some of the application areas we see customers becoming interested in with this system. Then, I will be glad to entertain any questions that you may have.

The IBM Instruments Computer System (IICS or CS9000) was developed, as you may have guessed, by IBM Instruments, Inc., a wholly owned subsidiary of the IBM Corporation. Their mission is to develop, manufacture, market, and service a line of analytical instrumentation in several major analytical technologies such as nuclear-magnetic resonance, electron-proton resonance, electro-chemistry, and chromotography, for example.

Recently it became clear that the trend in modern analytical instrumentation is to provide some degree of data analysis along with the instrument. Most instrumentation produces data that is spectral in nature and requires spectral analysis techniques which place increasing demands on the data processing capabilities of the analytical instrumentation computer.

In addition, this data takes up space so you need room for large data arrays. You also need improved precision of internal data representation and you need increased performace. Another requirement is the ability to connect to a wide variety of interfaces, everything from RS-232 to the blue-wire/green-wire interface that nobody has ever seen. For all of these reasons it became clear that a new kind of processor was going to be required. The IBM Instruments people went ahead and started the development program that resulted in a 68000 based system. This is the IICS.

The CS9000 tends to have a tall narrow profile, or a small footprint. The reason for this is that in the laboratory area, desk and bench top space is at a premium so you want take advantage of height rather than breadth.

Lets take a look inside this box and see how it is put together. The main processor consists of several boards. There is a flat board called the planar board, an expansion board, and additional feature boards. There is also the printer board, the case, power supply, printer, key pad and keyboard.

Lets take a closer look at the main processor unit in the area of the keyboard and keypad. The keyboard is almost identical to the IBM Personal Computer keyboard, in fact, it is electrically inter-changeable. We had an instance where a keyboard was giving us a problem, and we simply took a PC keyboard, plugged it in, and it ran fine.

The second highlighted area is a 57 keys keypad that allows you, under program control, to assign the role of those keys. You could write applications in which there is no keyboard required. A user could interact with the applications from the keypad. You can put overlays on top of the key pads and label them for whatever is appropriate for the application.

Moving back into the machine, is the printer. You can order a machine with or without a printer. The printer is a bi-directional matrix printer with a 4-color ribbon. It is also a plotter. The resolution of the printer when used as a plotter is 200 dots per inch horizontally and 336 dots per inch vertically so you can resolve anything that is on the screen.

Moving futher back, behind the power supply and the printer, is the rest of the machine. The planar board shown here is the hart of the computer. As you know, you can do very little with a microprocessor - it needs a lot of other stuff around it. A that's what makes up a total computer system.

The expansion board has 5 slots and brings the bus off the planar board up to an expansion board into which you can plug feature boards. We will talk more about feature boards later. There is a printer control board along the top.

As I said before, the heart of this machine is the planar board. One way to understand really what this machine is all about is to figuratively walk around the planar board.

First is the main processor board. We have managed to put a large amount of circuitry and components on a very, very tightly packed multi-layer board, using a Motorola 68000 microprocessor. This microprocessor has 32 bit registers with a 16 bit wide data path. It has 24 bits for memory addressing which allows us to directly address up to 16 megabytes of memory.

We run the 68000 with a 8 megahertz clock. The bus architecture is compatible with VERSAbus* which is a Motorola standard bus architecture. A few words about the VERSAbus. The feature expansion socket is a full VERSAbus socket. The expansion board discussed earlier in the presentation brings the VERSAbus sockets up off the planar board and provides a set of 5 sockets so that anybody who makes a compatible VERSAbus feature card can plug it directly into this machine. Several manufacturers now produce boards that plug directly into these sockets and provide Floating Point Hardware accelerators, array processors, communication functions, and additional storage.

As we move around the board, we next have dynamic RAM and DMA (Direct Memory Access). On the planar board itself we provide 128,000 bytes of Random Access Memory. This is implemented in 64,000 byte chips. The access time for this

*VERSAbus is a trademark of the Motorola Company.

memory is 250 nanoseconds and we have implemented RAM with no wait states. This provides for very quick memory data transfer.

There are 4 channels of Direct Memory Access implemented on this board. The DMA transfer rate is 1 megahertz per second. In addition, using feature boards, you can expand dynamic RAM in 265K increments up to 1 megabyte per board. You can put 5 boards in so you could add 5 megabytes of additional RAM on this machine.

The next area discussed is the CRT and graphics control section. The CRT is a high resolution monochrome display, 12 inch rasterscan screen, with 80 by 30 lines in character mode. The character matrix is 7 by 14 dots in a 9 by 16 frame. An important feature of our monochrome display is the 768 by 480 pixels, or pels, or resolutions, fully bit mapped. We have storage in this area for the acreen which is separate from user storage so it does not detract from the 128K available to the user. In addition, enough storage is provided here for two screens so you can write one screen and switch between the screens.

The I/O capability of the planar board consists of several ports. In particular, we have 3-RS232C ports. The baud rate goes up to 19.2KB, asynchronous ASCII coded characters. A nice feature for debugging programs is to put a short cable from one of these jacks to the other. You can write to one port and read in from another port and provide a very nice debugging tool for developing telecommunication program. There is a parallel I/O section here which provides an 8 bit group of parallel I/O with handshaking. Finally, you can see that there is also an IEEE488 instrumentation port that is native to the planar board. We support controller, listener, and talker. It uses direct memory access with a data transfer rate of 1 megahertz per second. With software overhead that's degraded to roughly half a megahertz per second.

The next area I will cover on the planar board is the diskette control area. We support on this machine floppy diskettes and hard disks. You can intermix up to 4 units. Either the 5¼ or 8 inch diskettes, again using DMA, with a transfer rate of about 500,000 bits per second. We also support up to 4, 5¼ inch hard disks with either 5 or 10 megabytes of formatted capacity. We use the SA1000 and ST506 Interface. This requires a separate control card.

Let's take another look at the expansion board area. As I said before, there are 5 VERSAbus sockets and we provide some expansion features of our own. The expansion feature card is used for adding memory, as I mentioned before and you can add up to 1 megabyte of storage in 256K byte increments so you could plug five memory cards into the system.

The next feature provided for by the expansion board is the sensor I/O card which provides analog sensor I/O. You can have up to 4 analog inputs, at 30 samples per second sampling rate. It has a dynamic range of 10^6. The A to D converter is an integrating converter with 12 bit resolution, polarity and over range indicator, and auto zero for high accuracy and low drift.

In addition on the sensor I/O card, you can have the digital input/output feature which includes 8 general purpose debounced switch inputs, 8 general purpose output drivers, if you want to light up LED's for example, 4, 8-bit ports for input and output, full handshake available on all the ports, fully buffered input and output bits and the digitals outputs will drive 5-volt relays. There are also 2, 16-bit timer counters on the analog sensor I/O board.

They run at a 2 megahertz maximum counting rate with 500 nanoseconds timing resolution. Under software control you can have either pulse, 1 shot, or square wave output.

The last item on the Sensor I/O card is serial communications in which we provide 4 additiona RS-232 serial ports which again are either terminal or modem configuration with data rates from 50 to 19,200 baud. An on board baud-rate generator is provided.

Well, so much for hardware, the nuts and bolts, let's move onto the operating system and software. The operating system on this computer is the Computing System Operating System called CSOS. This is an operating system, optimized for multi-tasking real time operations. It is a full multi-tasking operating system which supports several high level languages, in particular, BASIC, FORTRAN AND PASCAL and we'll talk more about them in a moment. We have made direction statements for communications, both 3101 emulation, 3270 emulation, a mathematical and statistics library and what I think is a pretty nice full-screen editor. In addition there are operating system extensions which include a text editor which is more of a line editor. We had the line editor before we had the full-screen editor. A macro assembler, a linkage editor, and disk utilities.

Let me tell you more about the multi tasking operating system. CSOS supports multiple tasks so you can have up to 8 tasks running. Since the system task is a task, that only leaves 7 the way the system is currently designed. You can set priorities from 1 to 127. The system task runs at priority level 64 so you could theoretically set a task to have a higher priority level than the system. However, it you do that you had better be careful because the system may never get control back if you don't ever let it go. CSOS is interrupt driven. We have implemented seven levels of priority on the interrupts. Four levels of interrupt are expanded to 32 programmable interrupts so you have a rich amount of interrupt capability. This is a multi-tasking system so each task gets a shot at the processor of 50 miliseconds and then its rolled out for the next one that is ready. The highest priority task on the ready-queue will get control and if it's waiting on I/O it will be rolled out and other task that's on the queue will come in. There are resident CSOS commands in ROM and there are transient commands out on the diskette.

The resident system commands are SPOOLER so you cn SPOOL a file out to the printer and go off and do something else. a SUBMIT which allows tasks to start up from a diskette text file, a SUSPEND which does just what its name implies - it stops a task for a designated time, a TASKS command which displays the current task status, and the TIME command for time of date.

The transient commands that are provided with the operating system are utilities such as COPY, DISKCOPY, DISKUTIL - which is nice for looking at sectors on diskette, a FORMAT and a HELP which gives you HELP messages on all of the commands that are available to you.

One of the more attractive software features we think will find a lot of use by applications developers is a common access path from any of the three application environments BASIC, FORTRAN, OR PASCAL to the system facilities. Object code created from ASSEMBLY LANGUAGE, PASCAL, or FORTRAN can be compiled, linked and stored starting at a particular address in the system. You could write a program in BASIC for example and take a USER EXIT to an address at which

this OBJECT CODE starts. This code could then be executed and you would then return to a USER RETURN location and continue executing your BASIC programs. This again is very important for applications. For example, if you are doing spectral analysis, and you have some new powerful fast FOURIER TRANSFORM ROUTINE that you want to use you can write your application is BASIC and just call your FFT from BASIC, invoke it and then return.

The system resources available to application languages use a common technique called SYSTEM CALLS AND FUNCTION PACKETS. I need to spend a little time on FUNCTION PACKETS because we find that this is extremely useful technique for getting to any of the system resources.

Each of the system resources or services, whatever they are, Disks, RS-232C, etc. utilize FUNCTION PACKETS. A FUNCTION PACKET is nothing more than an integer array. Each element in the array has a specific meaning depending upon the I/O driver being accessed. For example, if we are accessing the RS232 I/O driver, elements 20 and 21 in that particular function packet array would be the ones that you would use to set the BAUD rate.

The three application languages supported by this system are BASIC, FORTRAN, and PASCAL. CSOS BASIC is an enhanced BASIC that supports psuedo-compliation. There is no compiler on this machine for BASIC. Nor is there an interpreter. We perform what we call pseudo-compilation. When you key in a program, transparent to the user, we create a separate run-time file in which we do 4 things:

(1) We do syntax checking at entry time, so that if you enter some syntax which is snytaxically incorrect, you will get an immediate error message.

(2) We strip out all dead code, for example, remarks.

(3) We resolve all the symbol table pointers.

(4) We resolve and remove all line numbers.

Now this run-time file is what is executed when you run your BASIC program. It is not a compiled-basic because we don't create the file in native 68000 instructions. But it is not a pure interpretive file either.

We support a 64 bit representation of floating point, with an exponent range from -308 to +308 and precision is approximately 15 places. Integer type ranges from negative 32768 to positive 32767 and we support numeric string data type up to 56 characters. If you create a numeric 56 character string, you can do some numeric operations on it. We provide a compare, subtract, round, multiply, divide, and addition, so if you need to do string manipulation with 56 numeric characters you cn do that with this system.

Since this machine is designed for the engineering-scientific environment, it is important for it to support matrices. We support two dimensional arrays and we have built in primitives in the BASIC language, or extended the basic language to do matrix handling directly. This includes the ability to do dimensions, matrix reads and prints, take an inverse and transpose, zero, matrix addition and multiplication and in this case scalar and/or matrix multiplication. And also to do determinants. Now we also have increased the power of BASIC by adding to the ability to handle Virtual Arrays. Virtual Arrays are nothing more

than a BASIC object that is dimensioned larger than the working area that you have available. And it is stored out on either hard disk or diskette. At the current time, Virtual Arrays on diskette are useful but when we start shipping the hard disk then of course the speed improvement you get with hard disk over floppy diskettes will significantly improve the usability of virtual arrays. We also support substring searching in BASIC and chaining which is a common procedure these days for chaining together application programs in BASIC.

There are graphic primitives that do line, fill, points, and ellipses. If you are wondering where the circle is, I would just like to remind you that the circle is a special case of an ellipse. And we have the system call capability which we spoke of before.

We have also implemented PASCAL on CSOS. This is a PASCAL compiler that follows very closely the proposed ISO standard with some minor exceptions. We do support such things as a separate compilation, an include capability, strings,and structured programming. We have an otherwise clause on case statements, hex constants, pointer operations, and built in procedures functions, and built in data types. We support 8, 16 and 32 bit integer, 32 and 64 bit reals, the IEEE floating point compatability, boolean character strings and up to 4096 elements in sets.

This system as you may remember, was developed for engineers and scientists and, therefore we have to have a FORTRAN on it and FORTRAN 77 is the FORTRAN we have implemented. It is designed to the specifications of ANSI X.3.9 1978 full level. We just refer to it as FORTRAN 77. It includes complex data types, again important for engineering applictions, list directed IO, structured programming facilities, 1, 2, and 4 byte integers, 4 and 8 byte reals, Boolean 1, 2 or 4 and character data. The compiler produces native 68000 object code. FORTRAN is the fastest language on the machine. The ratios are that FORTRAN and PASCAL are comparable and run 10 to 11 times the speed of the pseudo-compiled BASIC.

No system is completed without a full screen editor and we are now shipping the full screen editor with the FORTRAN. It is I think, a very useful, nice screen editor. It is very similar to the personal computer editor. You can do block move and/or deletes. You can do global locates on context, changes, and function keys. You can do horizontal scrolling so that you can have up to 133 character lines if you wish. You can copy material from other files. There is paging up and down, insert and deletes. The editor divides the screen into two areas. The dark area at the bottom is the command line and when you are doing commands you use the escape key to toggle back and forth and that puts you into the command line where you would enter a command like locate/text/. The numbers directly on the right side of the control line stand for the current line number and the column number for the system into the program. At the lower left hand corner is the file name.

There are further extensions to the CSOS which include a macro assembler, a cross reference utility, a linker, locator, and library manager. These are necessary for people who are going to be programming in assemble language. For those people who are doing that, since assembler language programs generally also require a debug capability, there is a computing system debug which provides an environment in high storage to run a user program. To utilize debug you bring up the debug utility. It brings it into its own environment and then from within that environment you can run the program, look at registers, storage addresses, patch storage, set break points and do all kinds of thinks necessary to assemble the language programs.

We have gone over the IBM Instruments Engineerings Scientific Computing System and I would like to finish with a coup of words about where we see this product being used. In general most people are looking at it for at least three areas. One, as a relatively low cost intelligent controller usually with some data acquisition component to it. The IICS is a candidate for process modeling and simulation in control situations. A second application area is as an Engineering Scientific work station. This is a very powerful processor at roughly 10,000 floating point instructions per second with the current level of the operating system and that is without an optimizing compiler for FORTRAN or Floating Point hardware. Along that line with up to 5 megabytes of memory and the speed of the FORTRAN it obviously is very attractive at a FORTRAN application engine and we see a signficant amount of interest as a FORTRAN work station.

3/C/ear/7

SESSION REPORT

A Field Study of Human Factors Involved in Debugging FORTRAN Programs

61	A053		55
SHARE NO.	SESSION NO.	SESSION TITLE	ATTENDANCE

Human Factors	Jim Lipkis	NYU
PROJECT	SESSION CHAIRMAN	INST. CODE

Courant Institute, 251 Mercer St., New York, NY 10012 212-460-7166

SESSION CHAIRMAN'S COMPANY, ADDRESS, and PHONE NUMBER

A FIELD STUDY OF HUMAN FACTORS INVOLVED IN DEBUGGING FORTRAN PROGRAMS

Richard Halstead-Nussloch and Vance Sutton
IBM Corporation
Department C42/701-S
P.O. Box 390
Poughkeepsie, NY 12602

Phone: 914-463-6196

SHARE Installation Code: IBM

SHARE Human Factors Project

Session A053
Thursday 25 August 1983 10:15 A.M.