IV. Some Possible Enhancements

There are several enhancements possible to the SCAMP prototype which would significantly increase the function provided. These enhancements range from rather straightforward improvements of SCAMP APL to rather significant architectural extensions.

An improvement to the prototype SCAMP APL package would be an increase in the present 6 digit accuracy. If extended precision floating point routines were used instead of the standard routines now in use, the precision could be extended to at least 9 digits and possibly 11 digits. This would require some rather extensive modifications to SCAMP APL however.

Another extension to SCAMP APL would be to supply some of the more important missing operators such as Take and Drop. These could be implemented either by supplying system subroutines written in APL, or by actually implementing the necessary 1130 software. The first choice would be the easiest to implement but would also be the slowest in terms of execution speed.

An increase in the M-space component of the APL workspace would allow for both bigger arrays and/or more variables. This change would also require modification to the existing SCAMP APL software.

Except for the 6 digit accuracy problem, all of the above modifications would not be necessary if the 1130 APL Program Product package were used instead of the type 3 1130 APL package. This would require an extensive review of the 1130 APL program product, particularly with regard to how large variable-sized workspaces are handled. In contrast, the type 3 1130 APL package employs a fixed size workspace. The program product is quite a bit larger than the type 3 package and thus would require the use of a backing store. To save space, the multi-user features of the program product should be removed. The Program Product would need modification to work with the present SCAMP 1130 emulator.

If this were not possible, the emulator could easily be modified to reflect more accurately the standard 1130 architecture.

The alpha operator is an addition to the APL operator set which serves as an architectural extension allowing for the switching of I/O devices under program control. Normally APL accepts input only from a terminal keyboard and supplies output to a terminal printer. The alpha operator allows for the possibility of accepting input from other devices such as, a TP link, process control I/O, tape cassettes, etc. in addition to the above mentioned devices output could also be supplied to special graphic devices etc. A 2741 emulator written in APL is just one example of how this capability can be used. The ability to monitor and regulate laboratory experiments via process I/O is another example. The alpha operator has been partially implemented in the APL software, although not in the other system modules which would be involved. The operator is written ∝v where v may be a scalar or vector of integers. It is envisioned that each of the I/O connections described above would be assigned a number, and the user would specify the numbers he wanted in his argument v. Alpha is an operator which does not return a value; Presently it outputs 'FN'i where i is the number found as the argument. When a vector is specified, each of the numbers is output following an 'FN'.

Acknowledgements

This report is based on our joint work with many individuals. The authors would like to gratefully acknowledge the participations of the following individuals in particular: Paul Friedl of the Palo Alto Scientific Center who contributed to many of the technical decisions as SCAMP Project Manager; Pat Lang, John Kneemeyer, Jerry Serling, and Dick Flagg in programming development; Joe George, Dennis Roberson, Ed Finnegan, Greg Toben, Chris Christopherson, and George Marenin of Los Gatos in hardware development; Roger Abernathy, Dick Pratt, Bill Tutt, and Virgil Wyatt of Boca Raton in hardware development consultation; Joe Myers of Palo Alto in demonstration programming development. Without their devoted expertise the project would not have been brought to a successful completion and resulted in the present report.

References

(1) Iverson, Kenneth E. A Programming Language. New York: John Wiley and Sons, 1962

(2) A Programming Language/1130. Program Order Number 1130-03.3.001

(3) Friedl, Paul J. Extended APL Terminal Systems for Handling Sensors, Displays, and Local Files. Palo Alto Scientific Center Report ZZ20-6411. May 1971.

(4) Smith, Patrick A., Price, Kitty S., and Isaak, James D. An APL High-Speed Telecommunications Link for Use with Intelligent Terminal Networks, Palo Alto Scientific Center Report ZZ20-6413. January 1972.

(5) IBM 1130 Functional Characteristics. Form GA26-5881.

(6) Smith, Patrick A. 1130 Emulation With a Mod 25. Palo Alto Scientific Center Report ZZ20-6400. June 1970.

(7) S/360 Assembler for 1130 and 1800 Assembly Programs. Program Order Number 360D-03.1.013

(8) FDP 5798-APB. System/7 Programmable Terminal and Data Link.

(9) Penafiel, Hugo. Monitor/7: An Interactive Program For System/7. Palo Alto Scientific Center Report ZZ20-6421. February 1973.

Appendix A - Programming Tools Used

The success of any programming project depends to a large extent upon which programming tools are used to help in the development of new software. The selection of these tools is one of the more crucial 'design' decisions required. This selection process is guided by the computer environment that is available and by which tools already exist and can be adapted to the task at hand. The Palo Alto Scientific Center has a S/370 Model 145, which is used to run VM/370, and a System 7 with DI/DO capability. Also made available for this project was an 1130 with a 32K halfword memory, 1442 card reader, and 1132 printer. A large part of this project's success can be attributed to the excellent properties of VM/370 for doing program development.

1) PALM Programming Tools

Figure 8 shows a diagram of the programming tools used to produce PALM text tapes or to run an 1130 PALM Simulator. The PALM source file is created from a terminal and then assembled using the macro facilities of the S/370 Assembler. Each PALM instruction is defined as a macro so there was no need to write a PALM assembler per se.

The System 7 was used as a mechanism for generating PALM text tapes from text files generated in the S/370. To provide cassette tape input to SCAMP, a monitor program called M7 (9) is invoked and the text information is transmitted over a teleprocessing link to a System 7 containing the 'Programmable Terminal and Data Link' FDP (8). This FDP enables a System 7 to communicate with a host in the same manner as a 2741 would. It gives the System 7 the ability to write cassette tapes containing information supplied by the host machine.

A PALM simulator was obtained which runs on an 1130. This simulator has several options that were quite useful in debugging PALM code. The simulator accepts card input, which is generated from the assembly text file by a small

conversion program called CNV.

2) 1130 APL Programming Tools

Figure 9 presents a block diagram of the programming tools used to produce APL system input tapes to SCAMP, or to the 1130 Simulator that runs on S/370.

The 1130 APL Source Files are updated from the terminal, then assembled using a modified version of the S/360 Assembler for 1130 and 1800 (7). A SCAMP APL input tape is made in the manner described for PALM text tapes.

An 1130 Simulator which runs under VM/370 was written for this project. It provides the ability to interactively debug the 1130 APL SCAMP code from a terminal. Among the options that are available on the 1130 Simulator are the ability to perform traces, address stops, analyze which sections of the code are used most frequently, what percentage of the time the various 1130 instructions are used, etc.

3) SCAMP Debug Supervisor

The SCAMP Debug Supervisor was written as an interactive development aid for this project. Its purpose is to aid in the debugging of new PALM and APL code on SCAMP. When in use, the supervisor occupies about 2K bytes of PALM memory. Space for the supervisor is not shown in Figure 4 since it will not be present in the final product. The keyboard is used for command input and the display is used to provide messages to the operator. Commands provide the ability to: a) Modify SCAMP memory; b) Display SCAMP memory and registers; c) Pass control to a program at a specified address; d) Execute PALM code from a specified address to another specified address; e) Trace 1130 instructions during execution; f) Stop on a specified 1130 Instruction address; g) Dump memory to the printer.

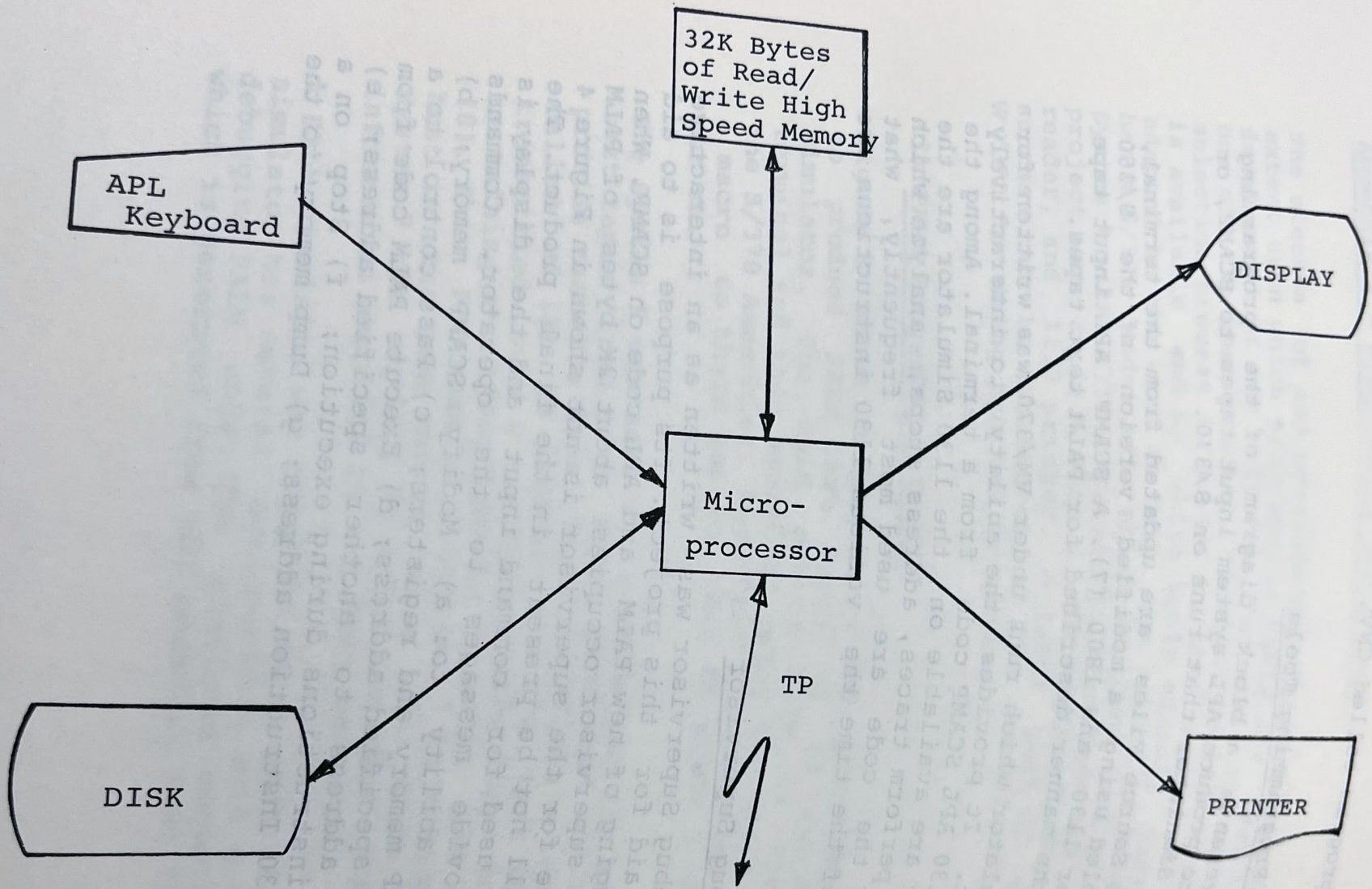


FIGURE 1. INITIAL SCAMP SYSTEM HARDWARE BLOCK DIAGRAM

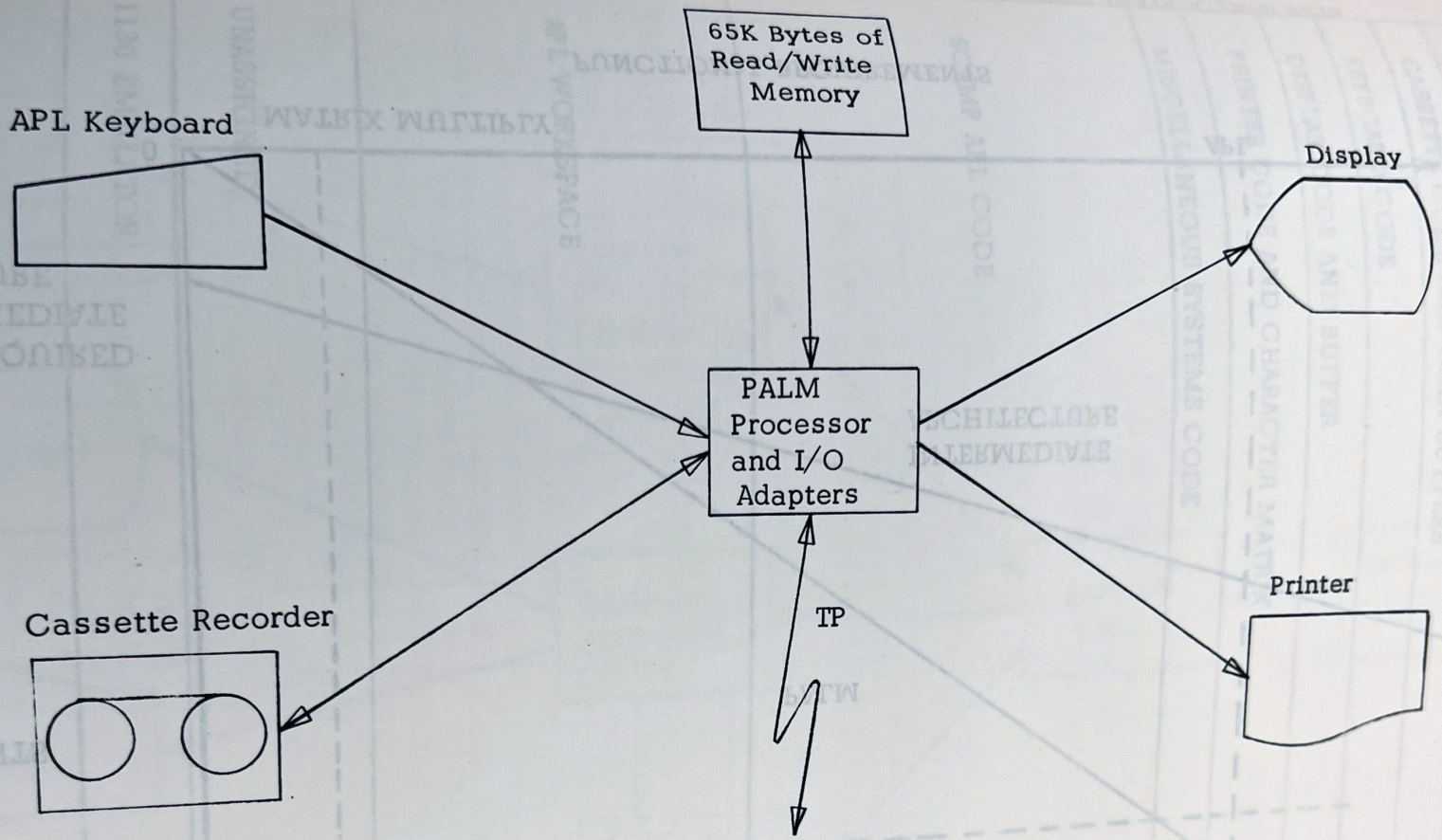


FIG. 2. SCAMP PROTOTYPE HARDWARE BLOCK DIAGRAM

IBM CONFIDENTIAL until March 1984,
thereafter, IBM INTERNAL USE ONLY.

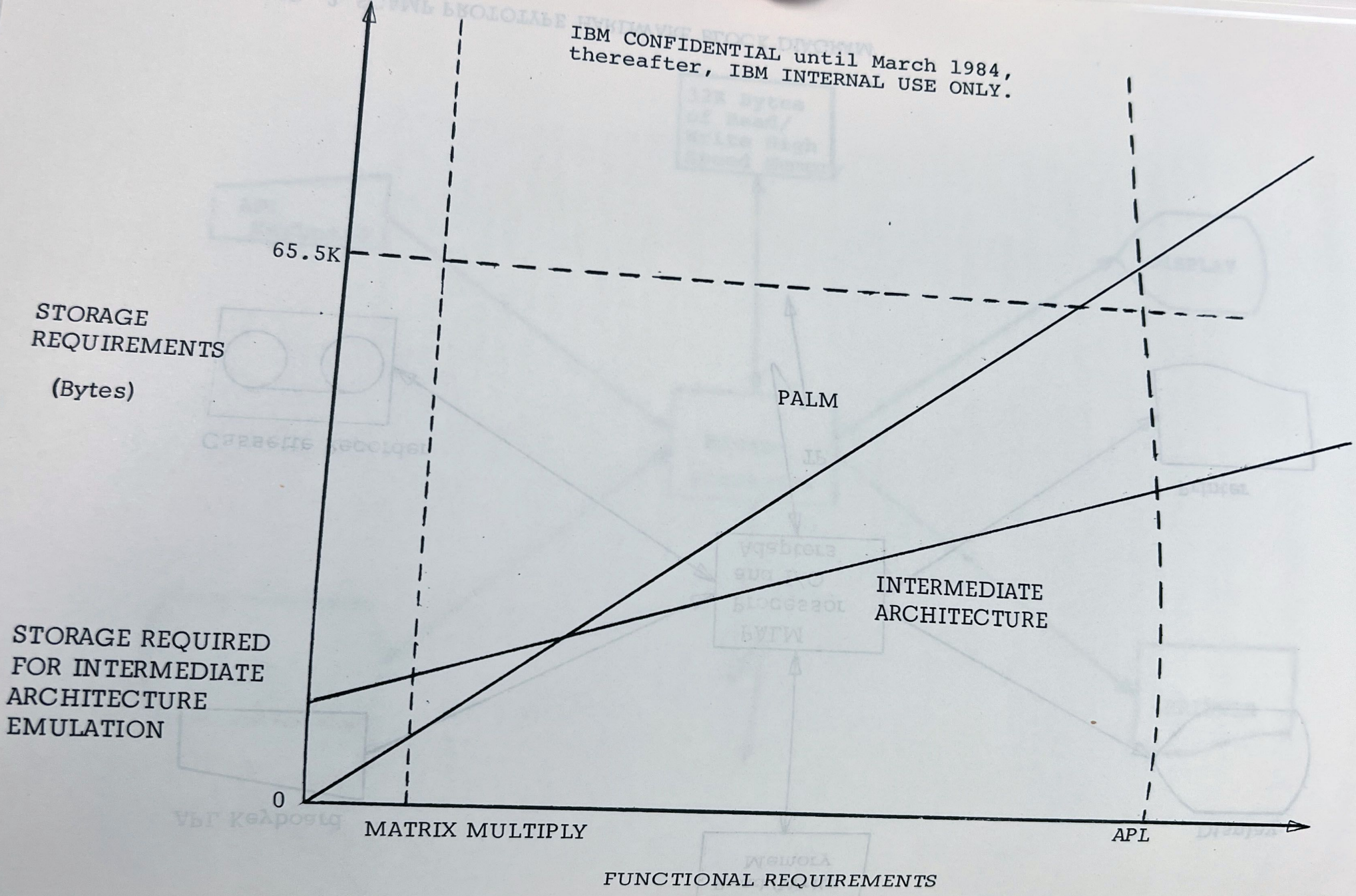


FIG. 3 FUNCTIONAL COMPARISON OF PALM AND INTERMEDIATE ARCHITECTURE INSTRUCTION SETS

DIRECTLY ADDRESSABLE MEMORY	.5K
CASSETTE CODE AND DATA BUFFERS	1.5K
KEYBOARD CODE	.7K
DISPLAY CODE AND BUFFER	1.2K
PRINTER CODE AND CHARACTER MATRIX	2 .2K
MISCELLANEOUS SYSTEMS CODE	.8K
SCAMP APL CODE	26 .0K
APL WORKSPACE	22 .2K
UNASSIGNED	6 .9K
1130 EMULATOR	3 .5K

FIGURE 4 SCAMP 1 MEMORY ALLOCATION

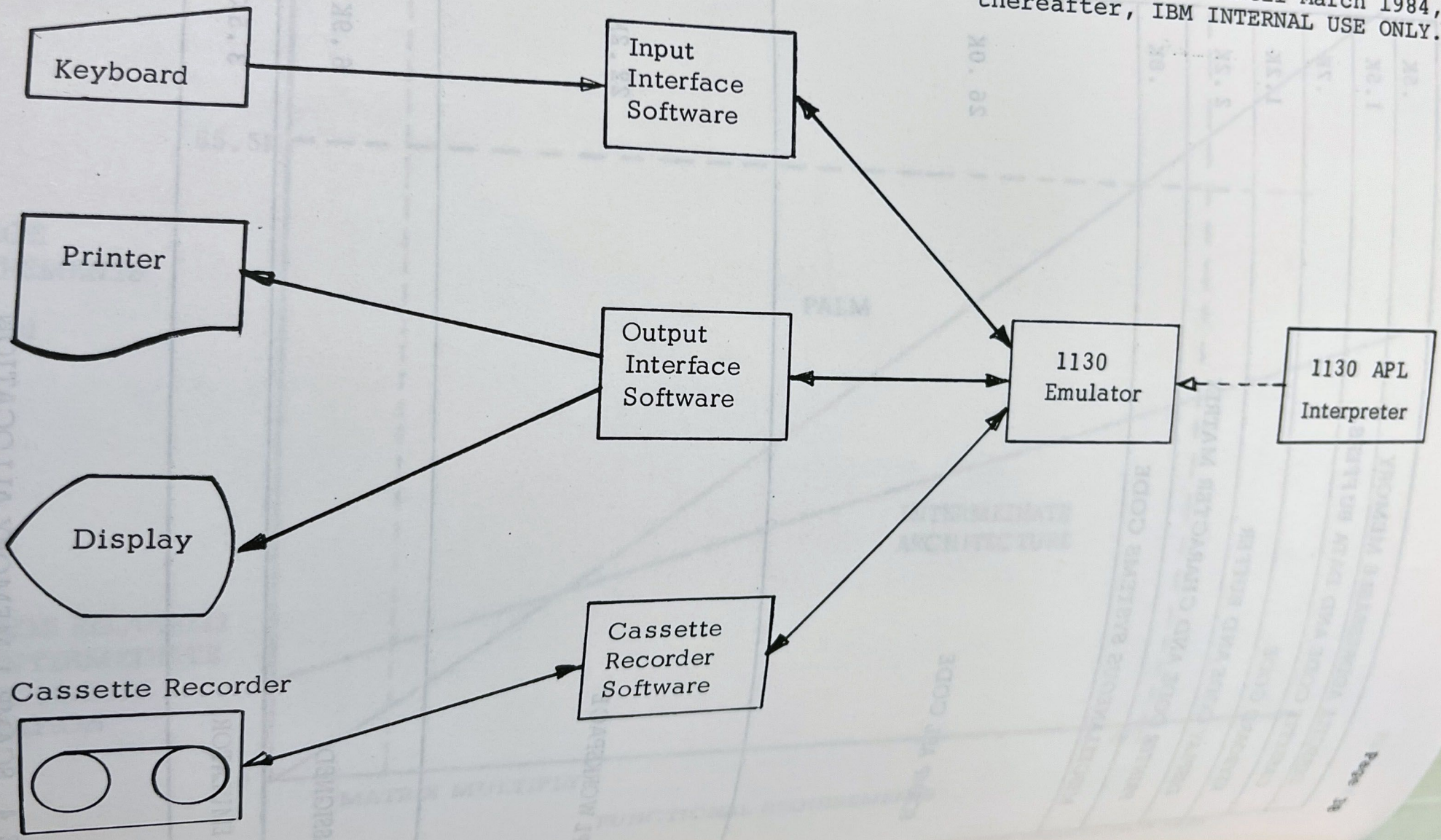
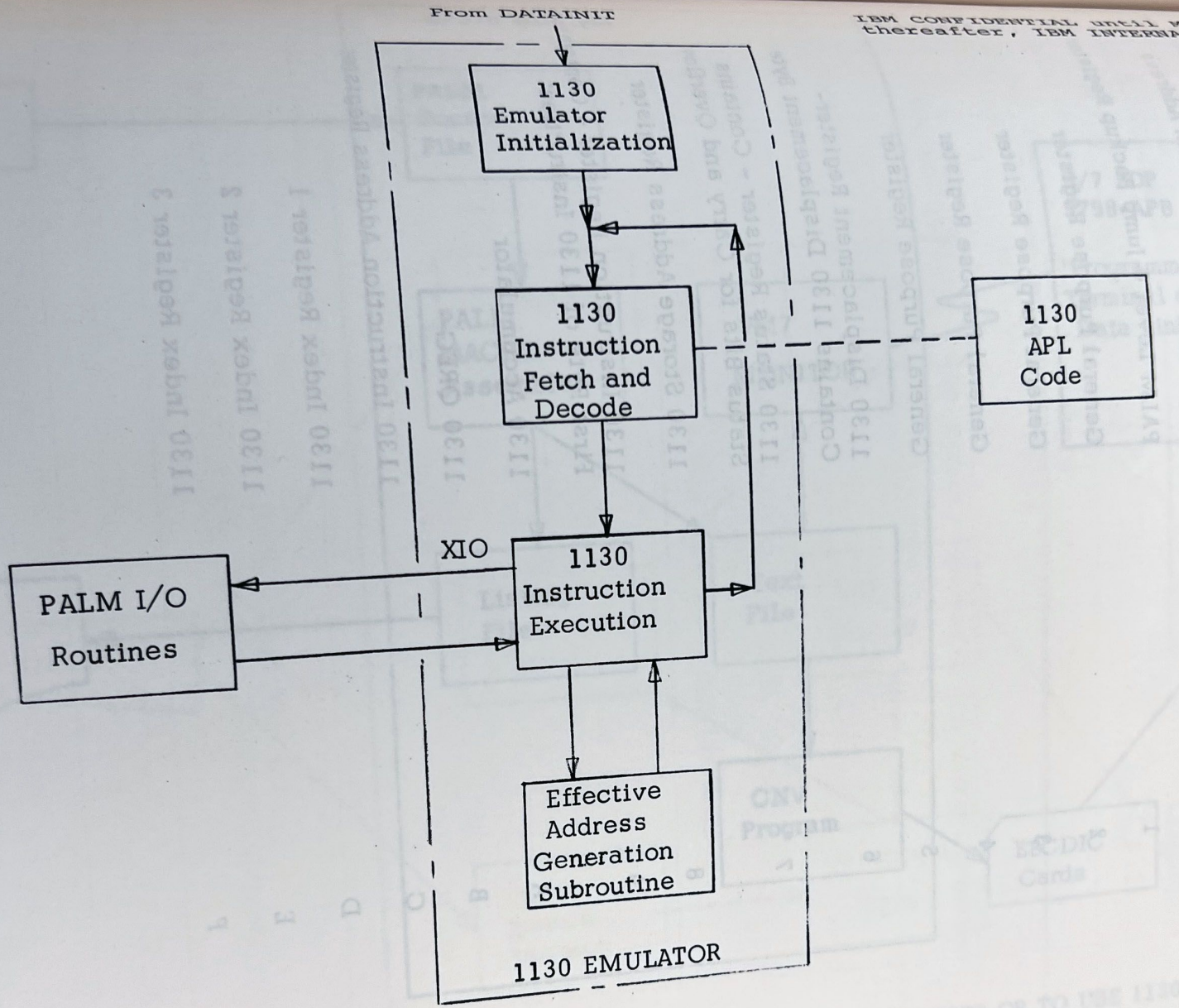


FIGURE 5 SCAMP SOFTWARE OVERVIEW



SCAMP 1130 EMULATOR BLOCK DIAGRAM

FIGURE 6

INTERRUPT LEVEL 0 PALM REGISTERS	MEANINGS ASSIGNED TO DESIGNATED REGISTER
0	PALM Level 0 Instruction Address Register
1	PALM Level 0 Jump Backup Register
2	General Purpose Register
3	General Purpose Register
4	General Purpose Register
5	General Purpose Register
6	1130 Displacement Register- Contains 1130 Displacement Byte
7	1130 Status Register - Contains Status Bits for Carry and Overflow
8	1130 Storage Address Register
9	1130 Instruction Register - Contains First Byte of 1130 Instruction'
A	1130 Accumulator
B	1130 QREG
C	1130 Instruction Address Register
D	1130 Index Register 1
E	1130 Index Register 2
F	1130 Index Register 3

FIGURE 7 1130 EMULATOR USAGE OF PALM LEVEL 0 REGISTERS

SYSTEM DIAGRAM OF TOOLS USED TO PRODUCE SCAMP SYSTEM TAPES OR TO USE 1130 PALM EMULATOR

FIGURE 8

SYSTEM DIAGRAM OF TOOLS USED TO PRODUCE SCAMP APL INPUT TAPE OR TO USE THE 1130 SIMULATOR

FIGURE 9

CONFIDENTIAL REPORT INDEXING INFORMATION

1. AUTHOR(S):
Patrick Smith and Kitty Price

2. TITLE: An Architectural and Design Overview of SCAMP

3. ORIGINATING DEPARTMENT:
Palo Alto Scientific Center

4. REPORT NUMBER:
ZZ20-6426

5a. NO. OF PAGES
42

5b. NO. OF REFERENCES
9

9. INDEX TERMS FOR THE IBM SUBJECT INDEX:
Emulation
Microprogramming
APL
03 - Communications
06 - Computer Peripheral
07 - Computers
21 - Programming

6a. DATE COMPLETED
March 1974

6b. DATE OF INITIAL PRINTING
May 1974

6c. DATE OF LAST PRINTING

7. ABSTRACT:

This report describes architectural considerations for the prototype design of an APL machine called SCAMP. Also covered are key software design decisions plus some possible I/O extensions to the existing SCAMP architecture.

This report will have limited distribution upon approval of need-to-know by Dr. H. P. Flatt, Palo Alto Scientific Center.

8. REMARKS:
IBM CONFIDENTIAL until March 1984
thereafter IBM INTERNAL USE ONLY.