| NASA | | NASA |

USL / DBMS NASA / PC R&D

WORKING PAPER SERIES

Report Number

DBMS.NASA/PC R&D-10

The USL/DBMS NASA/PC R&D Working Paper Series contains a collection of formal and informal reports representing results of PC-based research and development activities being conducted by the Computer Science Department of the University of Southwestern Louisiana pursuant to the specifications of National Aeronautics and Space Administration Contract Number NASW-3846.

For more information, contact:

Wayne D. Dominick

Editor
USL/DBMS NASA/PC R&D Working Paper Series
Computer Science Department
University of Southwestern Louisiana
P. O. Box 44330
Lafayette, Louisiana 70504
(318) 231-6308

A PERFORMANCE EVALUATION OF THE IBM 370/XT PERSONAL COMPUTER

Spiros Triantafyllopoulos

The University of Southwestern Louisiana
Computer Science Department
Lafayette, Louisiana

October 25, 1984

A PERFORMANCE EVALUATION OF THE IBM 370/XT PERSONAL COMPUTER

ABSTRACT

This report addresses an evaluation of the IBM 370/XT personal computer. The evaluation focuses mainly on the use of the 370/XT for scientific and technical applications and applications development. A measurement of the capabilities of the 370/XT was performed by means of test programs which are presented in Appendices A and B. Also included is a review of the facilities provided by the operating system (VM/PC), along with comments on the IBM 370/XT hardware configuration.

DISCLAIMER

TABLE OF CONTENTS

A PERORMANCE EVALUATION OF THE IBM 370/XT PERSONAL COMPUTER

1. INTRODUCTION

This report covers an evaluation of the IBM 370/XT computer provided for testing to the USL NASA/RECON research team by IBM Corporation, Baton Rouge, LA. The evaluation performed was primarily targeted toward the use of the 370/XT as an engineering/scientific workstation. Therefore, most evaluation aspects will address the areas of applications most common to engineering/scientific environments, as defined traditionally, but also as defined in the modern workstation concept as well. Applications development capabilities and functionality of the 370/XT are also evaluated, with the computer performing as an applications development host.

The 370/XT is a three-mode computer. It can be used as a regular IBM PC/XT running PC-DOS 2.1, as an IBM 3277 terminal connected to a mainframe, or as a "personal" IBM 370 system running the VM/PC version of the VM/SP operating system found in many IBM mainframes. It can take the personality of any one, and even run concurrently applications in all three modes, switching back and forth by using the ESCape key. Program, as well as data transfers, are now possible within applications, through the facilities for importing/exporting files between VM/PC and PC-DOS.

In this report, the stand-alone capabilities will be the primary focus, since the two other modes (terminal and regular XT) have specific characteristics defined either by the host environment (terminal mode) that can be used only if a compatible mainframe is available as a host, or as a regular PC/XT, whose capabilities are well known in the computer world.

A workstation for engineers/scientists should include certain features at all levels of its performance spectrum, and should make these features available for use to non-computer professionals with a minimum of computer expertise, effort and maximum efficiency. As a general view of the evaluation picture, the IBM 370/XT seems to fit well into the general description and requirements posed above.

A definition of the engineers/scientists workstation must be given first in order to accurately evaluate the IBM 370/XT. Such a workstation should be able to function as a stand-alone computer, with high-performance capabilities in numeric and scientific, as well as non-numeric applications. A workstation with such capabilities should provide support for the development, as well as execution of applications traditionally used in scientific/engineering environments. Furthermore, such a workstation should provide for data communications between host and local processors, access to remote systems, file transfers and related utilities. Since the workstation would be used by professional, non-computer expert personnel, it is expected that all the abovementioned functions should be available without complicated invocation protocols and procedures, for user efficiency and effectiveness.

It is apparent from the above general description of an engineer's/scientist's workstation that the majority of the users would want to develop high-level applications and execute all applications within their own field, in a demanding research and development and/or production environment.

An area in which the 370/XT can be used effectively is applications development and tools development. Since the capabilities of the XT/370 are high for its price range, such development can become very cost-effective. The unique capability of downloading programs, compiling and executing them in the VM/PC environment without consuming such mainframe resources, can be very efficient for both the user and the mainframe/PC system as a whole.

This evaluation will initially comment on the software tools available on the XT/370. Then, the suitability of the hardware/software combination will be evaluated according to the needs for supporting the workstation concept presented above.

2. APPLICATIONS DEVELOPMENT SUPPORT

The three modes of the XT/370 create a highly functional application environment for the developer. Applications are created on either the PC-DOS level or, as an improvement to the scope and capabilities, on the VM/PC level.

The software environment of PC-DOS is well known, and will not concern this evaluation. Rather, the 370/XT will be evaluated at the VM/PC level and its facilities will be compared to other systems in order to fully evaluate its potential. There are a number of software tools on the VM/PC level suitable for applications development which were tested for performance, usability and other factors that are important to the user.

A collection of programming aids is available. As programming language support, the evaluated machine had FORTRAN/VS, COBOL/VS and PASCAL/VS. The PASCAL/VS environment is not reported on since no formal evaluation was performed on it. Due to the nature of the TXTLIB and MACLIB library module organization on the VM/PC operating system, the exact names of the library segments where the compiler resides must be known in order to compile and execute PASCAL programs. Even when this was partially achieved, the batch-oriented nature of the compiler made interactive use rather difficult.

The /VS compilers are fully compatible with the corresponding products for IBM mainframes. All compilers and run time support systems are included in appropriate libraries (TXTLIB). A detailed discussion of the languages, environments, and appropriate benchmarking results and comments, follow.

2.1 FORTRAN

The FORTRAN/VS compiler available on the 370/XT has all the features of the ANSI FORTRAN-66 language specifications, supports the FORTRAN-77 extensions, and contains a number of IBM extensions for improved performance and advanced features. The compiler is invoked by the FORTVS EXEC segment, and, after compilation is complete, the user can execute the resulting TEXT object code by issuing FORTVSGO followed by the program name.

The compiler provides for a full-scale implementation of the language. To further show the compatability between the VM/PC FORTRAN language and the VM/SP mainframe version, the manual used for evaluation was the one supplied with the VM operating system on mainframes. Execution performance was measured by several standard benchmarks which were also run on USL's Honeywell Multics computer. For timing purposes, the system clock was obtained from the "QUERY TIME" command on VM/PC and the "general_ready -time" command on Multics. The results appear in the table below:

TABLE I

BENCHMARKS: IBM370/XT & FORTVS VS. H.I.S. 68/80 & FORTRAN 10.2

	370/XT LOAD TIME	370/XT COMPILE TIME	370/XT RUN TIME	MULTICS 50% COMPILE	MULTICS 50% RUN	MULTICS 90% COMPILE	MULTICS 90% RUN
BENCH1	1.31	2.03	11.20	0.15	0.10	0.20	10.40
BENCH2	1.42	2.12	2.23	0.23	0.20	0.25	1.45
BENCH3	1.25	1.35	0.28	0.10	0.10	0.20	0.45
BENCH4	1.30	1.16	1.10	0.14	0.21	0.18	5.20
BENCH5	1.20	1.54	0.29	0.20	0.10	0.27	0.23

All time units are minutes.seconds

As can be seen from the above table, the compilation process in the 370/XT is considerably slower than that of the Multics Fortran compiler. The effect of downloading a mainframe during peak usage, however, may counterbalance the speed difference between a 370/XT and a mainframe. In some of the test programs, the difference in execution speed of Multics loaded 90% and the 370/XT was not excessive. The speed difference can be seen on the BENCH2 program, in which initialization and multiplication of two 100 x 100 matrices needed 2 minutes and 23 seconds on the 370/XT, while the same task on Multics with 90% utilization used 1 minute and 45 seconds to produce the same results. The speed difference is only 38 seconds, and may be balanced by the 370/XT's speed in executing "routine" commands such as LIST, XEDIT, QUERY, etc., where it usually outperformed the 90% loaded Multics.

System load on the Multics system was provided by the command "hmu" or "how_many_users", that provides the system load in terms of active users, daemons and absentee (batch) requests.

Evaluation of the accuracy of the floating point calculations of the IBM 370/XT and Honeywell Multics was done by evaluating a Taylor series approximation of the sine function and comparing it with the built-in function of the 370/XT language. Single-precision floating point arithmetic was accurate to 9 digits in the 370/XT while double precision, 8-byte precision was accurate up to 17 digits. Multics figures appear approximately equally accurate, with double precision accurate to 16 digits [7].

File handling was a rather weak point on the 370/XT since the capabilities of the Winchester drives are limited in terms of accessing times. When excessive traffic was needed in the virtual memory (for example, for very large arrays or file I/O), execution time degraded considerably. Faster hard disks would definitely improve the performance of the system. This can be seen in the compiler loading phase; it takes almost 2 minutes to load the compiler in the virtual memory, and most of the overhead can be attributed to the hard disk speed (or lack thereof), and to the size of the compiler (1.5MB) as well. The access time of the hard disks, as given in the IBM technical specifications [4], is on the order of 80-90 milliseconds, and, in addition, all I/O and paging are done in 8-bit paths. This can explain some of the overhead associated with excessive virtual memory utilization.

As a conclusion to the evaluation of the FORTRAN compiler, it was found very powerful, yet easy to use. Error and warning messages were quite clear, and the process of using the compiler is simplified, since the EXEC segments FORTVS and FORTVSGO handle all compiling, linking and loading, compared to some microcomputer-based Fortran implementations requiring multiple commands and more complex procedures.

The library of scientific functions provided with the FORTRAN/VS compiler contains a collection of scientific subroutines and a variety of non-standard IBM extensions for string manipulation [1]. This, however, increases the size of the compiler on disk (1.5MB) and the virtual memory requirements of the compiler (2.0MB). This results in slower loading/compiling times when compared with other languages available on the 370/XT (e.g., COBOL). If faster hard disks were available (e.g., 30-40 msec access times), then the combination of FORTRAN/VS and VM/PC could be both more productive and time effective.

2.2 COBOL

The COBOL language used was the one found in the VM environment. However, its use was limited due to the lack of provided manuals describing the environment section and its associated devices and code names/catalogue names.

The compiler provided a full set of the ANSI standard COBOL language. Several options were attempted and all worked without deviation from the language standards. The COBOL compiler as a whole appears to be efficient, as judged from the benchmarks performed. File output was one major capability tested and the results, in terms of writing records to files, were very satisfactory when compared with the MULTICS COBOL file output test results, as can be seen from the results of program BENCH6 (see Table II). It appears that COBOL file output was also faster than FORTRAN file output on the IBM 370/XT, as writing 10,000 records on a disk file required 2 minutes and 10 seconds with COBOL and writing only 1,000 (or 10% of the records) required 1 minute 10 seconds (almost 50% of the time) in the FORTRAN benchmark BENCH4, in Table I.

IBM XT/370 COBOL compilation times were better than FORTRAN compilation times. This may be attributed to the smaller size of the COBOL compiler when compared to the FORTRAN compiler. According to IBM Specifications on the 370/XT Product Announcement [9], the OS/VS COBOL compiler requires 1.0 MB of disk space and 1.0 MB of virtual storage, compared with 1.5 MB of disk space and 2.0 MB of virtual storage required for the OS/VS FORTRAN compiler. Loading time was also reduced in the COBOL language. Thus, less disk/memory requirements for COBOL is one reason for the compilation and execution of COBOL programs being less time-consuming than the corresponding FORTRAN actions, on the order of 100% to 400%, as seen from Tables I and II. Average compiling time for the COBOL programs used in this testing was 46 seconds, while the corresponding average FORTRAN compiling times for equivalent complexity programs were up to 4 times as much.

TABLE II

BENCHMARKS: IBM370/XT & COBOLVS VS. H.I.S. 68/80 & COBOL 10.2

	370/XT	370/XT	MULTICS	MULTICS	MULTICS	MULTICS
	LOAD & COMPILE	RUN	50%	50%	90%	90%
	TIME	TIME	COMPILE	RUN	COMPILE	RUN
BENCH6	0.45	2.10	0.40	0.15	0.45	4.10
BENCH7	0.52	0.20	0.25	0.10	0.40	0.20

All units are minutes and seconds

There were two benchmarks written in COBOL. The first program, BENCH6, wrote sequentially a predetermined number of records to a user specified file. The records were written as character strings. Disk write time was efficient, judging from the figures of Tables I and II, especially when considering the low speed at which the Winchester disk operates. The same program was run on the Multics computer and the results can be seen in Table II.

The second benchmark, BENCH7, involved sequential initialization of a one-dimensional vector and performing the SEARCH command in COBOL. The compiler provided very helpful diagnostics, especially when the number of bytes to be allocated per array was exceeded. The maximum number of records that the compiler was able to allocate for a program was approximately 24,000. The execution time of this program was comparable to MULTICS, with a 50% load on MULTICS, which is good, especialy considering the virtual memory overhead.

COBOL provided many options for optimizing and changing the default compiler settings, so that the user could reconfigure the compiler environment. This was not needed in the runs for this evaluation. One feature that was a problem for the usability of the compiler was the fact that the appropriate library of object code (TXTLIB) was needed to be activated by the appropriate command before the compiler was invoked. This can be a minor nuisance for the novice user. The system provided some setup EXEC segments which improved the situation, but the names of the segments themselves are rather difficult to obtain in order to compile and/or run COBOL programs. A sample screen, with the appropriate arguments/options, activated by the COBOL command could be well suited to the unsophisticated user. Such a screen is displayed when the FORTRAN command is issued. This approach could display the COBOLVS program command to compile, and the COBOLVSGO program to load the program in memory and execute it.

The timing figures of the XT/370 COBOL compiler against the figures of the COBOL compiler running on the Honeywell MULTICS system are shown in Table II. Again, these figures were clocked with an approximate 50% and 90% system load on Multics. It can be seen that, when the load on Multics increases to the 90% region, a sharp increase in program compilation and execution time occurs.

2.3 PL/IG

The most powerful language available on the 370/XT, PL/IG, is the full G-subset of the PL/I language. The language compiler and its related libraries and run time support system were not available on the evaluation machine so benchmarking was not performed. However, the compiler requirements for the above mentioned libraries in terms of file space were large, on the order of 3.0 MB for disk space for the compiler, and 1.2 MB of disk space for the transient and resident libraries [9]. Virtual memory requirements are 1.0 MB for the compiler. According to the IBM product announcement for the VM/PC operating system [9], PL/IG is the full G subset of the PL/I language since the compiler is directly downloaded from 30X3 or 43X1 mainframes. PL/IG has been a very popular language for applications development and its ability to interface with other products (such as IMS or IDMS data base systems) should make the availability of the language on the VM/PC level very helpful and productive.

2.4 IBM 370 ASSEMBLY

One of the traditionally most used languages for system software, IBM 370 assembly language, is available on the 370/XT. The instruction sets are the same as the mainframe instruction sets so that compatibility is complete. This software product can be used with a large number of popular application programs and is very flexible.

As with PL/IG, the AS assembler was not available at evaluation time. However, it is a major software product and its availability should increase the potential capabilities of the 370/XT.

2.5 LANGUAGE CONCLUSIONS

The programming language support for application development on the IBM 370/XT was found excellent. It only requires more clarification to the user so that problems similar to those with the Pascal compiler mentioned in Section 2 are avoided. The compilers seem very good as a collection of software development tools, and the fact that they are compatible and conform to standards improve the transportability of the end products to a great extent.

The language support could be improved, however, with appropriate HELP segments or EXEC segments (similar to the one produced by FORTRAN). A last positive fact, however, is the accuracy and dependability of the software products. This is certainly a very important factor in any applications development environment.

2.6. OTHER APPLICATION DEVELOPMENT FACILITIES

Two important application facilities that the 370/XT has available for the user are the text editor XEDIT and the debug facility DEBUG. The editor is a highly versatile package, yet it is very easy to use, since it constantly displays the menu at the STATUS line (line 25) of the display. It is intented for program writing as well as general text editing.

The XEDIT editor is not very complicated. The main feature is that it allows limited menu capabilities for moving within the file(s) and selecting simple commands. The important feature is that it provides a command field for each line, so that the user can first issue the appropriate requested commands and then see the results on the entire screen. The IBM 370/XT keyboard cooperates very well with the special keys INSERT and DELETE, as well as the function keys (if set up properly). Traditional screen-type editing is also available, with the menus used to move through the files. Cut and Paste facilities, helpful features in editing, are good also. The XEDIT editor even features an "automatic new-line character" (the # character) that can be used for newlines in the text instead of <NEWLINE>. Another feature of the editor is its capability of seting defaults and symbols and allowing the user to refer to them, thus increasing user productivity by reducing keystrokes and simplifying the entry/editing process. In order to improve interaction and reduce time spent, the XEDIT editor allows execution of regular VM/PC commands so that the execution of programs and editing can be performed concurrently, or the editor can use output or provide input to programs that run on VM/PC [2, 3].

The DEBUG facility provides step-by-step execution of programs and allows many options. Its use was not explored heavily, mainly due to the lack of manuals. However, in several cases when it was used, it allowed rapid program debugging by displaying and modifying memory locations and registers. Interaction with the Program Status Word also facilitates the process [3]. DEBUG is a debugger that allows the placement of "break points" in the program so that execution can be controlled by the user, and values of storage locations can be manipulated during run time.

DEBUG can be used with any language that is available and compatible. This includes all five languages mentioned so far. DEBUG is interactive and can be used with application programs as well. Its main features are common to the interactive symbolic debuggers found in typical application development environments. For the experienced user, it is a very helpful tool, however, it still needs clear documentation and examples for the novice.

2.7 CONCLUSIONS

In this section, the application development environment of the IBM 370/XT personal computer workstation was evaluated. As an entity, the system is very coherent, running under VM/PC. The performance of several components needs improvement, though, through improvements in the supporting hardware to enhance FORTRAN/VS or through improvement in the usability of the system to enhance Pascal/VS and partially DEBUG. Such improvements can include more user-friendly interfaces, improved manuals with more examples/tutorials, an on-line HELP facility, and improved hardware in the form of faster (40-50 msec) Winchesters. The 370/XT has a high potential for stand-alone application development and it can be improved vastly by relatively minor modifications for increased performance.

3. APPLICATIONS EXECUTION

In this section, the environment provided by the 370/XT for application execution is evaluated. We define, as application execution, the process of executing various highly specialized programs without detailed knowlege of computer science topics by highly skilled, non-computer professionals such as research scientists, engineers, and so on.

Application execution requires a system to be as user-friendly as possible, but flexible and powerful at the same time as well. The IBM 370/XT was used at the VM/PC level, and the evaluation of the facilities available at that level will eventually answer the question of its usability within applications execution environments.

The VM/PC operating system uses the CMS (Conversational Monitor System) as the main interaction facility with the user. A critique on the available evaluated resources follows.

3.1 File Management.

The file handling in CMS is done primarily through the FILELIST command. The format resembles XEDIT and it can be very helpful. The user can list files and sort them according to user-selected attributes, and perform operations on any user-selected subset. FILELIST allows copying, deleting, journaling, renaming, and other operations on files. The manual has an excellent section on FILELIST.

3.2 Access Control.

A very important aspect of multiple users is handled appropriately by the VM/PC operating system. Users can be associated with different volumes, so that the users can link to other users' volumes and work with their files. This can be performed at either the terminal mode when the 370/XT serves as a 3277-type device, or at the stand-alone level. Access is given by the owner of the volume(s) to be mounted (linked). The appropriate access modes exist to protect information and ensure its proper use.

Access also is handled at the segment level, with all segments being associated with a Type. For example, user-only owned programs and other files have A1 mode assigned to their files. So a file PROG.FORTRAN will appear as "PROG FORTRAN A1" in a LIST statement. Other modes exist for shared files and object modules.

One last but very important feature of the 370/XT in the VM/PC mode is the capability of password logging in. Each user must have a user ID and a valid password. Therefore, access problems are reduced. The user can logoff by the LOGOFF command, which disables access to the Command Processor and returns the VM PC to wait for another login.

3.3 Data Exchange

Data exchange is provided between applications. With the 370/XT, a user can have up to three concurrent sessions (one 3277-type as a terminal, a stand-alone PC-DOS session and a 370/XT session). Selection can be done by the ESCape key. In order for such a configuration to be effective, data must be transfered from one application to another with little, if any, interruption to the user process. Such a procedure should also be easy, if effectiveness is desired. The VM/PC provides the IMPORT and EXPORT commands to read and write PC-DOS files from and to VM/SP files. Data transfer is accomplished with no problems whatsoever. These two commands were found to be very effective in terms of allowing the stand-alone configuration to communicate with the outside world. Standard 5 1/4 inch diskettes, formatted for PC-DOS, are used.

3.4 Synonyms and EXEC segments.

Synonym definitions allow the user to create user-specific commands or combinations of commands that can be activated with a specific synonym. For example, if, instead of LIST, the user wants to have the capability of using DIR, then, in the SYNONYMS file, the appropriate entry is used and the command can be executed by either LIST or DIR. SYNONYMS can handle combinations of commands as well. As part of the user support environment, synonyms have proven to be very useful as they allow customization of the operating system commands to fit the particular user's capabilities or task assignments. The idea of synonyms is extended by using EXEC segments, which are segments containing multiple commands with argument passing and user interaction. The EXEC segments can then execute very complicated command sequences.

The EXEC facilities on the VM/PC level are excellent. Still, there need to be some clarifications within the user's manual so that the user can understand their potential. Another problem is that no multiple (nested) EXEC segments are supported. Also, CP commands must be preceeded by the CP command before they can be used. This is a minor nuisance, however that does not create major problems as long as the CP commands are known.

3.5 Library Management.

The 370/XT provides very good library module management. Libraries of executable code (TEXT) can be created and mantained by the user, in addition to the usual libraries (TXTLIB). Such libraries are the FORTVSLIB and the VSCOBLIB for FORTRAN and COBOL. The system provides facilities for module creation and maintenance as well. Both facilities allow users to use libraries of application programs without using complicated pathnames, linkage procedures and other procedures that are not familiar to users executing applications. Macro libraries are also used, with user and system defined MACRO segments. The evaluation of the library and module facilities resulted in the need for programs that list the entire contents of these libraries. This facility must be apparent to the non-expert user since there must be a way to search and locate segments of executable application code without restoring to complicated methods. In order to provide execution of applications, the LOAD command is used. This command needs some clarification since the arguments of its usage are not clear. As a final view of the library management, the only comment is that a facility for examining detailed library contents is needed.

3.6 Various Utilities.

The stand alone 370/XT provides many utilities for the application user. Such routines PRINT files to the print spooler, SORT files, QUERY the system for user information, etc.. The main question in the utilities is that the PRINT command, by use of its spooler, does not allow use of the full 4MB of Virtual memory. If 4MB was to be allocated, then no space for spooling results and no files can be printed. This seems to be a problem since, according to the IBM VM/PC Reference Manual [3,5], the main memory (virtual) can be up to 4MB. With the spooler attached, the requirements of which are only 0.5MB, the system has to be configured for 2MB only. This limitation could be overcome, however.

Other utilities include SPOOLing files to the host, LINK files for virtual machine use and other utilities. In general, the utilities provided are the same ones found at the CMS level, plus a collection of PC-related utilities to do data conversion, exchange, etc. Most of the utilities that were evaluated were found to be easy to use and documentation appears adequate.

3.7 Conclusions.

The 370/XT running the VM/PC operating system can provide very good support to applications. Such support can be seen from the number of facilities that VM/PC provides for access control, process simplification, data exchange, or simply from the industry-compatible VM operating system environment support.

The only problem is, however, the space requirements to run VM/PC, which are estimated at close to 1.5MB. While the performance of the system is satisfactory for the expenses allocated, the space requirement can be a serious problem if not enough memory is allocated for applications software. A 10MB configuration may easily be used for limited applications execution, but, as the user demands increase, a second 10MB hard disk is the only answer. IBM suggests the use of floppy disk drives for CMS programs [5]. The access time, however is degraded and so is program execution time if disk I/O intensive programs (typical information storage and retrieval applications, for example) are run with the hard disk. Also, it is rather easy to exceed even the 20MB capacity if a large number of software applications and/or data files are used. In any case, however, the software performance and support for both applications development and applications execution was found satisfactory.

The next chapter will present some evaluation notes and comments on the hardware performance of the 370/XT.

4. HARDWARE CONSIDERATIONS

In this section, a brief overview and discussion of the 370/XT's technical characteristics will be given. No formal evaluation was performed, due to lack of appropriate equipment, e.g., hardware monitors. However, the components of the system and their functionality will be discussed at the level possible.

Three microprocessors cooperate in order to make the 370/XT work; two MOTOROLA MC68000 and an Intel 8087 function in addition to the 8088 on the motherboard of the XT. The first MC68000 fetches, decodes and emulates most of the fixed point 370 processor instructions; it also uses the MC68000 registers to emulate the 370's GPR (General Purpose Registers) and Program Counter/Status Word Registers. The second MC68000 handles memory management and exception handling, all supervisor mode calls and diagnostics. Naturally, the 8087 handles the floating point instruction set and emulates the 4 FPRs (Floating Point Registers).

All three microprocessors work in high speed, efficient modes. Their use is to emulate the 370 hardware, and according to independent figures [6] as well as our own benchmarks, seem to attain their goal.

There is a feature of the 370/XT, however, that drastically limits the performance of the machine, namely, the use of PC-DOS to handle all I/O and interrupts. According to IBM [5,9], the PC-DOS software handles all I/O from and to the keyboard/display, file I/O, and, most inefficiently, disk paging I/O. This results in 32-bit processing, while I/O is done at PC-DOS speed or less (due to synchronization problems). The situation could definitely be improved by having a separate I/O processor to handle all I/O. However, this is only a thought that needs to be explored and not a technical suggestion. One has to consider facts such as page size, paging algorithms, memory management schemes, and other system-specific details before attempting serious technical suggestions. An I/O processor could entail the risk of making the PC/XT unit incompatible with the 370/XT unit, also increasing the cost (two I/O configurations: 8088 for PC/XT and special I/O processing for the 370/XT).

The need for more efficient I/O is apparent from the benchmarks that multiply two 2-dimensioned vectors, implemented in VS/FORTRAN. In one case, multiplication of two 200 x 200 matrices was aborted after 45 minutes because of the large number of page faults generated and the Winchester disk's inability to handle the process efficiently. In contrast, similar operations on 100 x 100 matrices took around 2 minutes on the average. The problem does not appear to be with the processor itself, but rather with the

I/O devices that handle paging. The same was true in the COBOL file writing benchmark BENCH6, when 15-byte records were written to the Winchester (BENCH6), although on a smaller scale.

Another option which would be welcome on the 370/XT would be an extended RAM, on the order of 1M to 2M. This would minimize the amount of page traffic, and would definitely improve the performance without adding large amounts in terms of cost. The extra memory could be used by PC-DOS as well, to simulate a RAM disk and improve data handling by decreasing the access times. This modification is possible, although PC-DOS can not address more than 640K bytes and with a minimum of 1.5M or 2M only a portion of it is usable from PC-DOS. Since the MC68000 processors have 32-bit addressing, it would not be difficult to handle the additional memory so that paging would be kept to a minimum when operating at the VM/PC environment.

As mentioned earlier, faster disk devices may be the key to the improvement in speed. The newly announced PC-AT has a 40 msec disk access time compared to the 80-90 msec times of the older Winchester disks found on the XT. This alone could improve the speed of page fault handling by a significant amount, and improve disk I/O as well.

The last factor that this report covers is the 370/XT's capabilities for networking. IBM does not provide facilities for networking at the VM/PC level, but rather at the PC-DOS level. The networking would allow for improved cost effectiveness and resource sharing, while providing (or denying) access to critical data on the shared disks. The VM/PC can be extended up to the features of the VM/SP that allow multi-user capabilities to further improve its performance/cost factor. Then, a machine of this size would be able to provide a multi-user, multi-tasking environment for a number of users with a reduced cost per user.

5. CONCLUSIONS

The VM/PC operating system was found to be a very improved environment for applications development and applications execution over the traditional PC-DOS or CP/M environments common to workstations today. Software support and verified programs, compilers, utilities, etc., are available for this level. However, the hardware that executes the software appears to have difficulty coping with the demand placed by the software. Increased memory size, improved, faster disks, and better I/O processors can be the deciding factors between two products that can run the same software. Speed of execution seems crucial, especially within the application areas of many of the potential users, namely scientists and engineers. With improved speed would come improved marketability and demand, factors that justify even more improvement. If the hardware can perform at the level that the software does, then the IBM 370/XT can certainly be a prime contender within the scientific/engineering workstation environment.

REFERENCES

[1] IBM FORTRAN/VS Reference Manual and Library Manual. Manual Number GC26-3986 and GC26-3985, April, 1983.

[2] IBM Virtual Machine/Personal Computer Conversational Monitor System (CMS) Primer. Manual Number SC24-5236, January, 1984.

[3] IBM Virtual Machine/Personal Computer Conversational Monitor System (CMS) User's Guide. Manual Number SC24-5254, January 1984.

[4] IBM Personal Computer XT Technical Reference Manual. Manual Number 6322-508, August, 1984.

[5] IBM Product Announcement for the IBM 370/XT Models 588 and 568, National Marketing Division, October 18, 1983.

[6] Byte Magazine, Special IBM Issue, Volume 9, Number 9, October 1984, pp.210-218.

[7] Honeywell Multics Fortran 10.2 Reference Manual. Manual Number AT58-03B, December 1983.

[8] Honeywell Multics COBOL 5.2 Reference Manual. Manual Number AS44-02, December 1976.

[9] IBM Product Announcement for the IBM Virtual Machine Personal Computer (VM/PC) Licenced Program. National Marketing Division, October 18, 1983.

APPENDIX A

FORTRAN BENCHMARKS

```
      PROGRAM BENCH1
C ************************************************************
C *                                                          *
C *   THIS PROGRAM CALCULATES ALL THE PRIME NUMBERS BETWEEN  *
C *   5 AND 1000000. A SEQUENTIAL GENERATION ALGORITHM       *
C *   IS USED, SO THAT THE NUMERIC PROCESSING CAPABILITIES   *
C *   CAN BE FULLY EVALUATED.                                *
C *     INPUT:  NONE REQUIRED BY USER.                       *
C *     OUTPUT: THE NUMBER OF PRIMES BETWEEN 1 AND 100000    *
C *             AND ALSO THE LAST PRIME OF THE SEQUENCE.     *
C *                                                          *
C ************************************************************
C
      IMPLICIT INTEGER*4 (A-Z)
C
      PRIMES = 1000000
      NUMPRI = 2
      DO 5 I = 5, 1000000, 2
11       K = SQRT(REAL(I))
         DO 4 J = 3, K, 2
            IF (MOD(I, J).EQ.0) GOTO 5
4        CONTINUE
         NUMPRI = NUMPRI + 1
         IF (PRIMES.EQ.NUMPRI) GOTO 9
5     CONTINUE
9     WRITE(6,6) NUMPRI, I
6     FORMAT(1X, 'THE ', i8,' PRIME NUMBER IS ', I8)
C
      STOP
      END
      PROGRAM BENCH2
C *************************************************************
C *                                                           *
C *   THIS PROGRAM WILL INITIALIZE AND MULTIPLY TWO LARGE     *
C *   (UP TO 200 BY 200) ARRAYS. THE TYPICAL ALGORITHM        *
C *   IS USED TO MULTIPLY THE ARRAYS. SEQUENTIAL INITIALI-    *
C *   ZATION IS PERFORMED. THIS PROGRAM HAS BEEN INCLUDED     *
C *   TO TEST ARRAY HANDLING CAPABILITIES OF THE MACHINES     *
C *     INPUT:  NONE REQUIRED BY USER.                        *
C *     OUTPUT: NONE PRODUCED.                                *
C *                                                           *
C *************************************************************
      INTEGER M1(200,200), M2(200,200), MR(200, 200), K,L,M
      INTEGER I,J,N,VALUE,S
      VALUE = 0
      K = 185
      L = 200
      M = 195
      DO 1 I = 1,K
         DO 2 J = 1,L
            M1(I,J) = VALUE
            VALUE = VALUE + 1
2        CONTINUE
1     CONTINUE
      VALUE = 0
      DO 3 I = 1,L
         DO 4 J = 1,M
            M2(I,J) = VALUE
            VALUE = VALUE + 1
4        CONTINUE
3     CONTINUE
C -------------------------------------
      I = 0
10    I = I + 1
      J = 0
20    J = J + 1
      S = 0
      DO 25 N = 1, L
         S = S + M1(I, N) * M2(N, J)
25    CONTINUE
      MR(I, J) = S
      IF (J.LT.M) GOTO 20
      IF (I.LT.K) GOTO 10
C ---------------------------
      WRITE (6, 93)
93    FORMAT(1X, 'END')
      STOP
      END
      PROGRAM BENCH3
C ***************************************************************
C *                                                             *
C *   THIS PROGRAM CALCULATES FIBBONACCI SEQUENCE    BETWEEN *
C *   1 AND A USER INPUT. A SEQUENTIAL CALCULATION ALGORITHM*
C *   IS USED, SINCE FORTRAN CAN NOT HANDLE RECURSION.        *
C *   THIS PROGRAM WAS USED TO EVALUATE SPEED AND PRECISION *
C *   OF INTEGER ARITHMETIC.                                  *
C *     INPUT:  N, FOR THE N-TH TERM.                          *
C *     OUTPUT: THE FIBBONACCI NUMBER THAT CORRESPONDS TO    *
C *             THE SEQUENCE 1..N.                            *
C *                                                             *
C ***************************************************************
      INTEGER A, B, C, I, N
      A = 0
      B = 1
      I = 2
      WRITE(*, 1)
1     FORMAT(1X, 'PLEASE INPUT N')
      READ(*,2) N
2     FORMAT(I5)
3     IF (I.GE.N) GOTO 4
         I = I + 1
         C = A + B
         A = B
         B = C
      GOTO 3
4     WRITE(*,5) N, C
5     FORMAT(1X, 'FIBONACHI OF ', I5, ' IS ', I8)
      STOP
      END
      PROGRAM BENCH4
C ************************************************************
C *                                                          *
C *   THIS PROGRAM WRITES 20-BYTE RECORDS IN A FILE. SEQUE-  *
C *   NTIAL FILE IS USED. THE NUMBER OF RECORDS IS 100,000.  *
C *   THIS PROGRAM TESTED THE FILE I/O CAPABILITIES OF THE   *
C *   FORTRAN LANGUAGE COMPILER                              *
C *     INPUT:  NONE REQUIRED BY USER.                       *
C *     OUTPUT: THE FILE THAT CONTAINS THE RECORDS THAT      *
C *             WERE CREATED.                                *
C *                                                          *
C ************************************************************
      IMPLICIT INTEGER (A-Z)
      DO 10 I = 1, 100000
         WRITE(2,20) I
20       FORMAT('THE  RECORD IS ',I5)
10    CONTINUE
      STOP
      END
      PROGRAM BENCH5
C **************************************************************
C *                                                            *
C *   THIS PROGRAM CALCULATES THE SIN TRIGONOMETRIC FUNCT-     *
C *   ION FOR A GIVEN X. A TAYLOR SERIES ALGORITHM WAS         *
C *   USED, SO THAT THE FLOATING  PROCESSING  CAPABILITIES     *
C *   CAN BE EVALUATED.                                        *
C *     INPUT:  THE VALUE FOR X, THE INPUT VARIABLE.           *
C *     OUTPUT: THE SIN(X) AS CALCULATED BY THE PROGRAM        *
C *             AND ALSO AS GIVEN BY THE FORTRAN LIBRARY.      *
C *             ALSO THE DELTA EPSILON VALUE FOR ERROR.        *
C *                                                            *
C **************************************************************
      IMPLICIT DOUBLE PRECISION (A-Z)
      INTEGER I
      X = 1.23456
      ERROR = 10E-12
      TERM = X
      TSIN = X
      I = 1
      X2 = X * X
1     IF (ABS(TERM).LT.ERROR) GOTO 2
         I = I + 2
         TERM = -TERM * X2 / REAL((I * (I - 1)))
         TSIN = TSIN + TERM
      GOTO 1
2     MYSIN = TSIN
      WRITE(6,3) ERROR, X, MYSIN, X, SIN(X)
3     FORMAT(1X, 'THE ERROR IS ', F22.20,/,
     &       1X, 'THE CALCULATED  SIN(',F5.3,') IS ',F22.20,/,
     &       1X, 'THE MULTICS     SIN(',F5.3,') IS ',F22.20)
      STOP
      END
```

Multics ec Segment
****

```
lts -nlb
& --- Start log session to record execution
gr -time
& --- Obtain the time from the system clock
fortran &1 -fold -la
& --- Compile with the large array option if needed,
& --- translating all input to lower case.
gr -time
& --- Obtain system time for compilation time calculation
&1
& --- Execute the program that was compiled before.
gr -time
& --- Obtain time needed to time the run of the program
sts &1.[time]
& --- Record execution of program in a segment.
```

IBM 370/XT EXEC Segment
****

```
CP  QUERY  TIME
# --- Obtain system time for compiler load timing.
FORTVS BENCH1
# --- Compile the program; the compiler provides
# --- Appropriate measure for Compile time only.
CP  QUERY  TIME
# --- Obtain compilation & load time measure
GLOBAL TXTLIB VFORTLIB
# --- Initialize library modules needed to be activated.
CP  QUERY  TIME
# --- Start timer for execution of program timing.
LOAD BENCH1 (START NOMAP)
# --- Load program in core memory and execute it.
CP  QUERY  TIME
# --- Final time reading of execution process.
```

APPENDIX B

COBOL BENCHMARKS

```
IDENTIFICATION DIVISION.
*
* THIS PROGRAM WILL CREATE A SEQUENTIAL OUTPUT FILE
* AND WILL WRITE A PREDETERMINED NUMBER OF 20-BYTE
* RECORDS TO IT. THE FILE NAME IS "OUTPUT PRINTER A1".
* THE PROGRAM WAS USED TO EVALUATE THE FILE CAPABILITIES
* OF THE COBOL COMPILERS USED.
*     INPUT:  NONE REQUIRED BY THE USER.
*     OUTPUT: THE FILE IN THE CURRENT DIRECTORY OR DEVICE.
*
PROGRAM-ID. BENCH6.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.             XT-370.
OBJECT-COMPUTER.             XT-370.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT OUTFILE ASSIGN TO PRINTER
    CATALOGUE-NAME IS "OUTPUT".
DATA DIVISION.
FILE SECTION.
FD  OUTFILE
    LABEL RECORDS ARE OMITTED
    DATA RECORD IS OUT-RECORD.
01  OUT-RECORD.
    02 JUNK-REC PIC X(15).
    02 JUNK-NUM PIC 999999.
WORKING-STORAGE SECTION.
77  REC-COUNTER PIC 999999 VALUE 0.
77  NUMBER-REC  PIC 999999 VALUE 0.
PROCEDURE DIVISION.
MAIN-LOGIC.
    OPEN OUTPUT OUTFILE.
    MOVE 050000 TO NUMBER-REC.
    MOVE "THE RECORD IS   " TO JUNK-REC.
    PERFORM WRITE-A-RECORD VARYING REC-COUNTER
        FROM 1 BY 1 UNTIL REC-COUNTER = NUMBER-REC.
    CLOSE OUTFILE.
    STOP RUN.
WRITE-A-RECORD.
    MOVE REC-COUNTER TO JUNK-NUM.
    WRITE OUTFILE FROM OUT-RECORD.
```

```
 IDENTIFICATION DIVISION.
*
* THIS PROGRAM WILL INITIALIZE A ONE DIMENSIONAL
* INTEGER ARRAY. THEN IT WILL PERFORM SEARCH ON A GIVEN NUMBER.
* THE SEARCH RESULTS ARE PUT IN "OUTFILE PRINTER A1".
* THE PROGRAM WAS USED TO EVALUATE THE ARRAY CAPABILITIES
* OF THE COBOL COMPILERS USED.
*    INPUT:  NONE REQUIRED BY THE USER.
*    OUTPUT: THE FILE IN THE CURRENT DIRECTORY OR DEVICE.
*            CONTAINING THE RESULTS OF THE SEARCH.
*
 PROGRAM-ID. BENCH7.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER.             XT-370.
 OBJECT-COMPUTER.             XT-370.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT OUTFILE ASSIGN TO PRINTER.
 DATA DIVISION.
 FILE SECTION.
 FD  OUTFILE
     LABEL RECORDS ARE OMITTED
     DATA RECORD IS OUT-RECORD.
 01  OUT-RECORD.
     02 OUT-REC PIC X(15).
 WORKING-STORAGE SECTION.
 01  DATA-TABLE.
     02 NUMBERS OCCURS 20000 TIMES
        INDEXED BY MY-INDEX
        PIC 99999.
 77  IND-1 PIC 99999 VALUE 0.
 77  TITLE-OUT PIC X(15).
 77  LOOK-FOR PIC 99999 VALUE 0.
 77  REC-COUNTER PIC 99999 VALUE 0.
 PROCEDURE DIVISION.
     SET MY-INDEX TO 1.
     OPEN OUTPUT OUTFILE.
     MOVE 19999 TO LOOK-FOR.
     MOVE 'UNSORTED  TABLE' TO OUT-REC.
     WRITE OUT-RECORD AFTER ADVANCING 2 LINES.
     PERFORM INIT-ARRAY VARYING IND-1
         FROM 1 BY 1 UNTIL IND-1 = 20000.
     SEARCH NUMBERS
         AT END PERFORM REQUEST-NOT-FOUND
         WHEN NUMBERS(MY-INDEX) = LOOK-FOR
             PERFORM REQUEST-FOUND.
     CLOSE OUTFILE.
```

```
    STOP RUN.
INIT-ARRAY.
    ADD 1 TO REC-COUNTER.
    MOVE REC-COUNTER TO NUMBERS(IND-1).
REQUEST-FOUND.
    MOVE 'RECORD IS FOUND' TO OUT-REC.
    WRITE OUT-RECORD AFTER ADVANCING 2 LINES.
REQUEST-NOT-FOUND.
    MOVE 'RECORD NO FOUND' TO OUT-REC.
    WRITE OUT-RECORD AFTER ADVANCING 2 LINES.
```

Multics ec Segment
****

```
lts -nlb
& --- Start recording the execution phase of the benchmark.
gr -time
& --- Obtain system clock value for compilation timing
cobol &1 -fold
& --- Compile the program, translating all input to lowercase
gr -time
& --- Obtain timer reading for compilation process
an  &1 [lowercase &1]
& --- Translate executable segment name to lowercase
& --- (needed to execute program in Multics)
[lowercase &1]
& --- Execute the program just compiled.
gr -time
& --- Obtain system time for timing of run
sts &1.[time]
& --- Save generated segment in a file.
```

IBM 370/XT EXEC Segment
****

```
CP  QUERY  TIME
# --- Obtain system clock value for timing of run
COBOL BENCH6
# --- Compile the COBOL program
CP  QUERY  TIME
# --- Obtain system time for compilation timing.
GLOBAL TXTLIB COBLIBVS
# --- Initialize library modules in memory
CP  QUERY  TIME
# --- Obtain system time for execution timing
LOAD BENCH6 (START NOMAP)
# --- Load and execute object program
CP QUERY TIME
# --- Obtain final time reading
```

5.10

1. Report No. IN-32	2. Government Accession No. ~~147389~~ 183580	3. Recipient's Catalog No.
4. Title and Subtitle 49P. USL/NGT-19-010-900: A PERFORMANCE EVALUATION OF THE IBM 370/XT PERSONAL COMPUTER		5. Report Date October 5, 1984 DATE OVERRIDE
		6. Performing Organization Code
7. Author(s) SPIROS TRIANTAFYLLOPOULOS		8. Performing Organization Report No.
9. Performing Organization Name and Address University of Southwestern Louisiana The Center for Advanced Computer Studies P.O. Box 44330 Lafayette, LA 70504-4330		10. Work Unit No.
		11. Contract or Grant No. NGT-19-010-900
12. Sponsoring Agency Name and Address		13. Type of Report and Period Covered FINAL; 07/01/85 - 12/31/87
		14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This Working Paper Series entry addresses an evaluation of the IBM 370/XT personal computer. The evaluation focuses primarily on the use of the 370/XT for scientific and technical applications and applications development. A measurement of the capabilities of the 370/XT was performed by means of test programs which are presented in appendices to the report. Also included is a review of the facilities provided by the operating system (VM/PC), along with comments on the IBM 370/XT hardware configuration.

This report represents one of the 72 attachment reports to the University of Southwestern Louisiana's Final Report on NASA Grant NGT-19-010-900. Accordingly, appropriate care should be taken in using this report out of the context of the full Final Report.

17. Key Words (Suggested by Author(s)) PC Performance Evaluation, PC-Based Research and Development	18. Distribution Statement

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 47	22. Price*

*For sale by the National Technical Information Service, Springfield, Virginia 22161