Microsoft Multitasking MS-DOS Product Specification

INTRODUCTION

Microsoft Corporation

October 30, 1984

ABSTRACT

This memo gives an introduction to the key architectural challenges facing software development for the office automation environment. It discusses Microsoft's solutions for these challenges, solutions that involve not only MS-DOS 4.0, but significant changes in the internal structures of all Microsoft software.

1. Prerequisites

This document is one of a series of related documents. They are:

- *Microsoft Multitasking MS-DOS Product Specification OVERVIEW*
- *Microsoft Multitasking MS-DOS Product Specification DEVICE DRIVERS*
- *Microsoft Multitasking MS-DOS Product Specification SYSTEM CALLS*
- *286 and 8086 Compatibility*
- *Microsoft Multitasking MS-DOS Product Specification INTRODUCTION*
- *Microsoft Multitasking MS-DOS Product Specification MEMORY MANAGEMENT*
- *Microsoft Multitasking MS-DOS Product Specification DYNAMIC LINKING*
- *Microsoft Multitasking MS-DOS Product Specification SESSION MANAGER*

Explicit and implicit reference will be made to terms and concepts introduced in that document.

2. Introduction

The next versions of MS-DOS will be more than just another MS-DOS release; they will contain powerful new features to address key issues critical to the future growth of office automation microcomputers: memory utilization, system protection, and software architecture. MS-DOS 4.0 is will be the first Microsoft Operating System to provide a full-scale implementation of Microsoft's future software architecture while simultaneously providing full "downward compatibility" for existing MS-DOS programs.

This memo provides an overview of the architectural difficulties facing the microcomputer software industry and gives a general discussion of the solutions contained in the future MS-DOS releases. Other papers provide more specific information about particular subsets of those capabilities.

The office automation marketplace demands software that is powerful, flexible and easy to use. The user will expect to do many unrelated things at the same time. He will expect to be able to combine different packages from different vendors, and have them work together. He will also expect the system to be able to do work transparently on his behalf, such as receiving mail. In order to meet these general needs, the software environment must address four distinct challenges:

1) Memory Space Limitations

   The PC architecture supports up to 640K of memory. This is not nearly enough; just the DOS, a network package, a windows package, and Lotus Symphony will consume the entire memory. A software solution must be found to this hardware problem.

2) Protection & Security

"Protection" refers to the capability to protect the system and other applications from an aberrant application. "Security" refers to the capability to protect the system and other applications from deliberate attempts to access or change data. These facilities must be provided to ensure that companies will buy networked Office Automation systems.

3) Flexible Architecture

Office Automation environments are too complex to be serviced by a single, monolithic program. A flexible architecture is necessary to allow a whole spectrum of applications and packages from different vendors to be effectively integrated.

4) 286 Upgrade Path

The 286 is the only feasible successor to the current 8086/8088. In 286 mode it can solve some of the above problems, but it will not run existing software. An architecture is needed that allows programs to take advantage of the 286 feature set while remaining 8086 compatible. Further, as it will be years before the 286 is the dominant microprocessor, it is vital to develop an architecture that provides these key features in an 8086 environment in a manner that is upwardly compatible to the 286.

3. Memory Space Limitations

The 8086 architecture limits addressable RAM to 1 megabyte. The defacto standard IBM PC architecture further limits this to 640K. This memory has to hold all applications, the DOS, and packages such as the network server.

It is certainly possible to fit the DOS, the network package, the window package, the device drivers, etc., into the 640K, but there will be little space left for applications. This restricts the software because as the DOS, applications, and other packages are improved they increase in size. Each new product, more powerful than the last, will decrease the usability of the system because of the memory restrictions.

The problem is greatly aggravated because programs cannot be swapped out. Swapping is a classic technique used by virtually every multitasking general purpose operating system. The 8086 architecture lacks the necessary hardware to allow useful swapping of arbitrary tasks. Finally, the situation is made worse by the fixed nature of 8086 programs. They cannot be moved around within the address space. 8086 model programs generally load as one contiguous chunk of memory; the combination of large chunk size, limited memory, and immovability produces major fragmentation difficulties.

RAM prices will continue to drop, RAM density will continue to rise, but neither will alleviate this fundamental problem. Although the user may easily *afford* megabytes of RAM, the fundamental architecture of the system limits usable memory to 640K.

4. Protection & Security

The term "protection" refers to facilities that protect programs and data from accidental damage; "security" refers to protection against deliberate attempts to read or modify programs and data.

The current unprotected systems operate fairly well in a single-tasking mode. If a program occasionally changes a random memory location, that location is usually past the end of the program's segment in an unused memory area. Programs doing enough damage to be noticed are correctly blamed, because they are the only programs running at the time of the damage or crash.

Protection is a much greater concern in a multitasking environment. If four tasks are being run, then there is four times the risk that damage will be done. If a program oversteps its bounds, it is likely to damage some other program that will then "blow up" or cause further damage. Often the damage is not noticed for quite some time; perhaps bad results are generated or perhaps the damaged program crashes hours after the actual malefactor has terminated.

The greatly increased likelihood of damage and the great difficulties in detecting the damage and identifying the cause make this a nasty problem. Customers will undoubtedly blame the hardware manufacturer and the OS developer for the fact that their machine has crashed twice in one week because there is no apparent correlation between the running of the faulty application and the subsequent damages. A crash is mild compared to an undetected data modification that could cause untold grief.

The current unsecure systems are acceptable to users because they sit on people's desks and are typically dedicated to one person's use. They are as secure as a desk-top notepad, via the same mechanisms (e.g., a lock on the office door). As soon as networking becomes common, however, security will become a critical issue. Few companies will be willing to automate their office with a system that allows anyone to access surreptitiously and/or modify any data or communication. It is no comfort to point out that the average office worker doesn't have the knowledge to "hack the network"; it's just a matter of time before "break and enter" programs are available in the advertisements in the back of "PC" magazine.

5. Flexible Architecture

The issue of software architecture has a tremendous bearing on the current and future utility of the personal computer. A powerful architecture will allow the rapid development of extremely sophisticated systems; a weak architecture will cripple future software development. A good software architecture should provide:

1) Portability

The IBM product line includes the PC, the XT, PC-*jr*, and the AT. Other manufacturers have their own multi-machine product lines. Undoubtedly more will come along. The system architecture should maximize the portability of applications among these systems. This should not be accomplished via a "common denominator" approach, but in a manner that allows the features of each environment to be used to the fullest.

2) Upgradability

A typical customer has a significant investment in hardware and software. A good architecture will allow him to upgrade his system piecemeal, both hardware and software.

3) Configurability

A good architecture allows extensive software and hardware configurability. It is impossible to produce a system that has one of everything—it is too expensive. A good architecture allows a non-expert user to piece together the kind of system he needs.

4) Flexibility

This is perhaps the most important point: the office automation environment is too varied and the industry is too active to be addressed by a single fixed product. The current DOS serves primarily as a file system package and a program loader; programs must be effectively self-contained. If a program is to be able to generate graphical output to any of a dozen devices then it must contain code to drive each of them. The end result is a program that is large, unwieldy, and inflexible; its ability to deal with a dozen devices does not allow it to deal with a thirteenth. The DOS does allow installable device drivers that alleviate many of the problems of device independence, but this mechanism is not strong enough to serve the general need.

6. 286 Upgrade Path

As has been mentioned, the Intel 286 processor contains hardware provisions to solve some key architectural problems. Unfortunately, the 286 (running in 286 mode) is sufficiently incompatible with the 8086, and no existing 8086 programs will run in the protected mode. This prevents Microsoft from offering a "protected mode only" MS-DOS for three reasons:

1) Millions of 8086-based Machines

   It is estimated that 8086/8088-based machines will outsell 286-based machines until sometime in 1986. Even then, a significant portion of machines sold will be 8086-based and the "installed base," millions of machines, will be predominantly 8086. A product that ignores 8086 systems could not be successful for several years.

2) "Catch 22"

   Ignoring existing and new 8086-based machines, there is a "catch-22" situation: customers might be willing to buy 286-only MS-DOS for their new 286 boxes, but only if they can purchase applications such as Lotus 123. Lotus might be willing to produce a 286-only version, but only if the 286-only MS-DOS sells well.

3) Multiple Versions

   ISVs are highly resistant to multiple versions of their products. They take time to produce, they confuse the customer, and they complicate ordering and stocking for everyone from the manufacturer through the distributor up to the retail store manager, who has limited shelf space.

   If faced with the choice of writing their program in 8086 mode, that would run on both 8086's and 286's in compatibility mode, versus producing a second version that runs in protected 286 mode, ISVs will choose the simpler one-version approach.

In summary, the operating system architecture must allow programs to be written so that a single binary copy can run in protected 286 mode on a 286-based machine, and still run on existing 8086 machines, including 8086 machines running earlier versions of MS-DOS. If this cannot be arranged, then the 286 will never happen; 286 chips will be run exclusively in 8086 mode "forever." The solution to these problems will have to wait until the 386 microprocessor is available (that can run 8086 programs while providing protection), and this means a delay of many years.

7. A Segmented Software Architecture

Microsoft has developed a segment oriented software architecture that addresses the discussed problems in a powerful and effective way. In addition, the architecture runs well on both 8088 and 286 machines, allowing programs to take advantage of the 286's powerful features while remaining 8086 compatible. Finally, Microsoft's architecture anticipates the 386, providing a uniform and compatible environment across all three processor classes.

This section briefly describes Microsoft's segment architecture, and demonstrates how this architecture resolves the above needs.

The fundamental concept in the new architecture is the segment. Although the 8086 is widely described as a segment-based machine, it is actually a linear machine that uses base registers to allow its 65K addressing capability to extend to 1 megabyte. 8086 addresses are often described as "segment:offset" but they are in fact "base:offset"; each byte in memory can be addressed by 4K different "base:offset" pairs. When MS-DOS 2.0 or MS-DOS 3.0 loads a 100K program into memory it uses 100K of contiguous memory and sets the base registers to its beginning. It is the responsibility of the program to adjust the base registers so that it can reach the remainder of its memory.

A segment-based architecture, on the other hand, views a program as a collection of segments, none larger than 65K bytes. The system provides a variety of services on a per-segment basis:

- segment relocation
- segment swapping
- virtual segments
- automatic sharing of pure segments
- dynamic linking (both loadtime and runtime)
- load on demand
- efficient (runs on the PC as well as the AT)
- compatible w/ MS-DOS, XENIX and 286 protected environment
- maximum transparency to applications

Note that the 286 (in protected mode) is a true segment-based machine with built-in hardware to provide the above capabilities in an efficient manner. Unfortunately, most existing applications assume the "base:offset" address form instead of the "segment:offset" form and thus cannot run in 286 protected mode. Microsoft's architecture uses software to provide the above services to both "base:offset" and "segment:offset" programs on the 8086; on a 286 machine "base:offset" programs can run only in 8086 mode, and "segment:offset" programs can be run in both 8086 and 286 modes. This frees them from the 640K limit and gives them a secure and protected environment.

8. Memory Space Limitations

Microsoft's segment architecture cures the 640K problem by allowing segments to be moved and swapped. Often the ability to coalesce free space by moving segments will yield the needed memory; if not, segments can be swapped to an optional swap device. Note that the segments are "virtual"; some of a program's segments can be swapped while the program continues to run in other segments. When the program references or returns to a swapped segment, that segment is automatically retrieved from the swap device.

In addition to allowing better use of the existing memory, segment architecture actually reduces the effective size of applications in two ways: demand loading and library sharing. *Demand loading* is an old virtual memory trick. Some, or all, of the segments are not loaded into memory until they are actually referenced. A typical 120K program might contain 60K of infrequently used code and data, exception handling, and rarely used services. By putting these items in separate segments, they will only be loaded if they are referenced.

Program size can also be reduced by using shared standard libraries. For example, Microsoft's architecture allows the creation of a C-language run-time library. All running C programs will then share a single copy of the run-time library, reducing each program by (typically) 10K.

Finally, multiple copies of pure segments are combined into a single copy that is then shared among all the users. This is done transparently and automatically; no program or user action is required. If a given application has 100K of code and 30K of data, a second copy can be started using an additional 30K bytes, instead of 130K.

It is important to note that programs need not reject the "base:offset" addressing mode to use these features, but they should follow some protocol to take full advantage of the segment facility. Programs can continue to run "as is," fixed into a contiguous chunk of memory, or they can be partially conforming and take partial advantage of the features. In all cases, full compatibility is maintained; the system supports a mixture of non-conforming and conforming tasks.

9. Protection & Security

Unfortunately, no software can provide protection or security in the 8086 environment. In the 286 environment, however, programs that adhere to "segment:offset" addresses can be run in 286 mode and are therefore fully protected and secure.

Note: If the user is running a combination of 8086 and 286 mode processes the 8086-mode processes can still do as they please, even to the extent of damaging 286-mode processes.

Partial protection is still valuable; users can run untested programs in 286 mode without fear. A user can fully secure his system by using only 286-compatible applications.

10. Software Architecture

The Microsoft segment architecture provides:

1) Portability

The dynamic linking facility provides a powerful portability tool. Just as device drivers allow portability across devices, dynamically linked procedures provide an equivalent service at the procedural level. As an example, C programs are binary-compatible with MS-DOS and XENIX under this mechanism, since they do all their I/O and OS interfacing via the C run-time library. A call to **printf()** is a call to **printf()**, regardless of how the implementation of **printf()**'s may differ on each system.

As has been stressed, applications that use the segment architecture's services run under MS-DOS 2.0, MS-DOS 3.0, MS-DOS 4.0, and the 286. Under MS-DOS 4.0 they automatically take advantage of swapping and virtual memory; on the 286 under MS-DOS 4.0 they automatically take advantage of the 286's memory management hardware in 286 mode.

2) Upgradability

The great flexibility and portability of the architecture allows the user to upgrade piecemeal, and at will. Note that the upgrading is not restricted to hardware: a user might replace his plotting library package with a specialized or higher performance one; his applications will automatically start using the new library package.

3) Configurability

Similarly, such a system is highly configurable, both in hardware and software. Dynamic loading and shared libraries allow the effective use of devices, such as laser printers, that may need a more powerful interface to the using program than that provided by device drivers.

Software is configurable. The user can buy the packages he wants and just "plug them in." As an example, a user of a commercial network service might install a routing clearing-house package provided by the network vendor. If the user later changed vendors the new vendor's clearing-house system could be installed. Some programs may be sold in sections: a base-level product and some options packages. The user can buy just the components needed and add others as desired.

4) Flexibility

From a long term viewpoint, this is probably the most important issue discussed in this memo, since the flexibility of the system as a whole will dictate the rate at which the products will improve and new technologies will be introduced.

Simply, the segmented architecture allows Microsoft to move from a restrictive, monolithic, self-contained architecture to a dynamic, flexible, segmented, and virtual architecture. Programs will no longer need to be self-sufficient. They can call upon a variety of services from other packages and routines. Further, they can do so in a manner that is extremely powerful and flexible, as well as being upwardly and downwardly compatible across the industry's entire MS-DOS-compatible computer offerings.

11. Getting There From Here

The popularity of third-party software products produces a classic catch-22 situation: customers won't buy a new machine or system until software is widespread; ISVs won't convert their software until they see that the machine/system sells well.

The segmented architecture described above avoids this problem via its compatibility. 8086 customers will buy MS-DOS 4.0 immediately because of its multitasking capability; 286 customers will buy it for the multitasking and its superior memory utilization. ISVs will rapidly convert their offerings to the segment architecture because:

1) The new version is downward compatible; they won't have to market parallel versions.

2) Their product will then have a strong competitive edge over their competitors since their product runs as if it had more memory while actually consuming less memory.

3) If they can easily convert their product to 286-style addresses they will do so to gain another competitive edge: a protected and secure program when running on 286-based hardware. ISVs are aware that the introduction of networking will make protection and security key issues, and they will be anxious to make their offerings compatible with such an environment.