



US005495611A

# United States Patent [19]

[11] Patent Number: **5,495,611**

Bealkowski et al.

[45] Date of Patent: **Feb. 27, 1996**

[54] **METHOD AND APPARATUS FOR DYNAMIC LOAD OF AN ABIOS DEVICE SUPPORT LAYER IN A COMPUTER SYSTEM**

5,230,052	7/1993	Dayan et al. ....	395/700
5,230,058	7/1993	Kumar .....	395/800
5,247,659	9/1993	Curran et al. ....	395/575
5,257,342	10/1993	Sideserf et al. ....	395/700
5,269,022	12/1993	Shinjo et al. ....	395/700

[75] Inventors: **Richard Bealkowski**, Delray Beach;  
**John W. Blackledge, Jr.**, Boca Raton;  
**Douglas R. Geisler**, Boca Raton;  
**Michael R. Turner**, Boca Raton, all of Fla.

*Primary Examiner*—Kevin A. Kriess  
*Attorney, Agent, or Firm*—Robert S. Babayi

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

## [57] ABSTRACT

[21] Appl. No.: **902,311**

A personal computer system is disclosed which is compatible with application programs and operating system software. The personal computer system includes a microprocessor electrically coupled to a data bus, non-volatile memory electrically coupled to the data bus, volatile memory electrically responsive to the data bus, a memory controller electrically coupled to the microprocessor, the volatile memory and the non-volatile memory, and a direct access storage device electrically responsive to the data bus. The non-volatile memory stores a first portion of operating system microcode and the volatile memory includes a volatile operating system portion intended for use by the first portion of the operating system microcode. The memory controller regulates communications between the volatile memory, the non-volatile memory and the high speed microprocessor. The direct access storage device stores a second portion of operating system microcode which is accessed by the microprocessor as needed.

[22] Filed: **Jun. 22, 1992**

[51] Int. Cl.<sup>6</sup> ..... **G06F 9/00**

[52] U.S. Cl. .... **395/700; 364/DIG. 1; 364/280.2; 364/280**

[58] Field of Search ..... **395/700, 650, 395/500**

## [56] References Cited

### U.S. PATENT DOCUMENTS

5,136,713	8/1992	Bealkowski et al. ....	395/700
5,175,831	12/1992	Kumar .....	395/425
5,187,792	2/1993	Dayan et al. ....	395/725
5,210,875	5/1993	Bealkowski et al. ....	395/700
5,220,667	6/1993	Ichieda .....	395/700

**9 Claims, 16 Drawing Sheets**

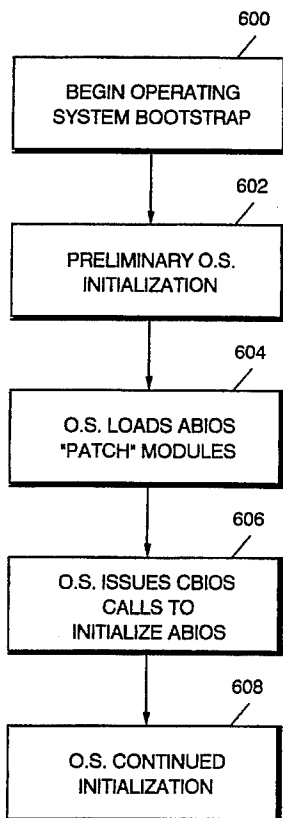


FIG. 1A

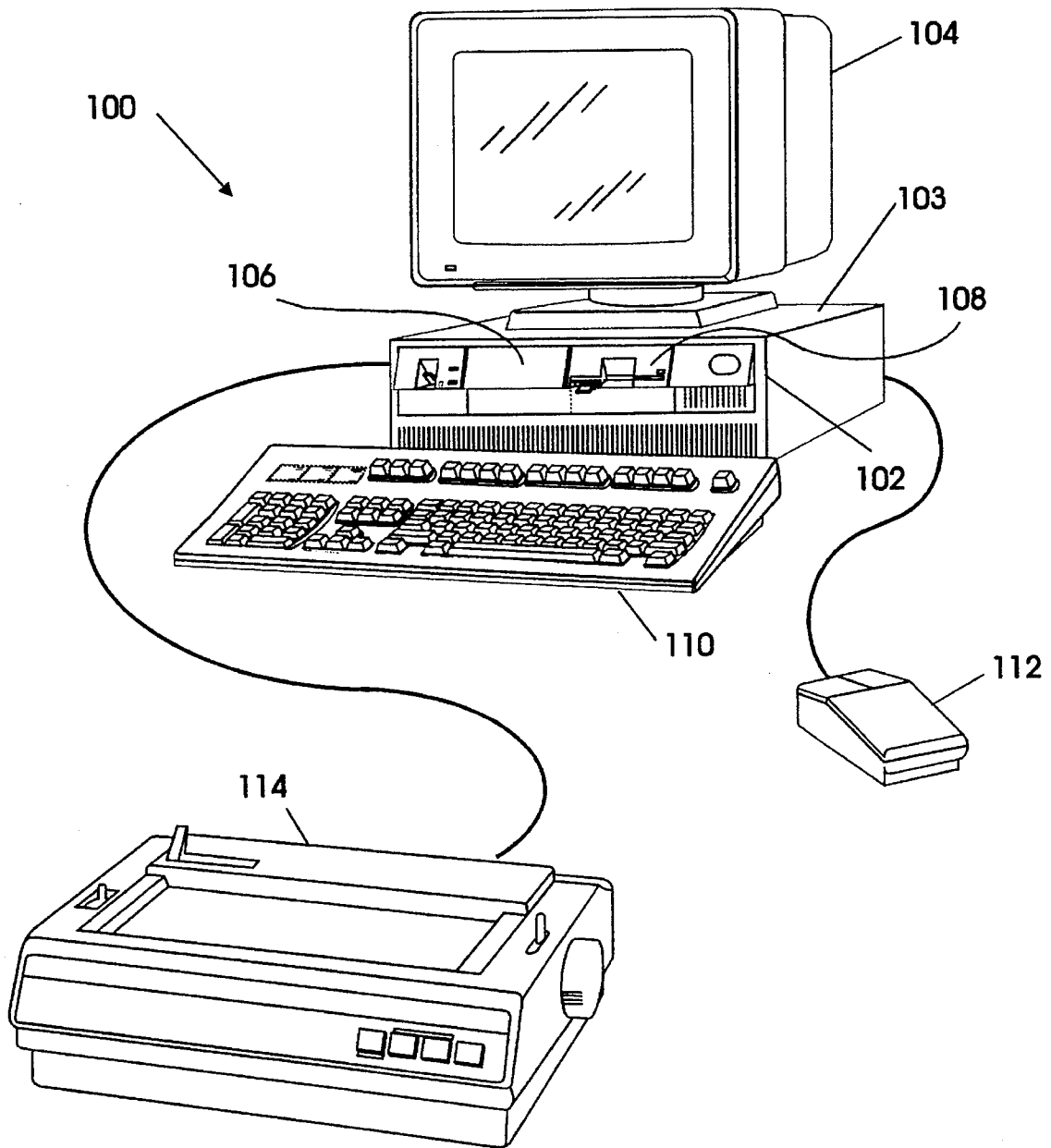


FIG. 1B

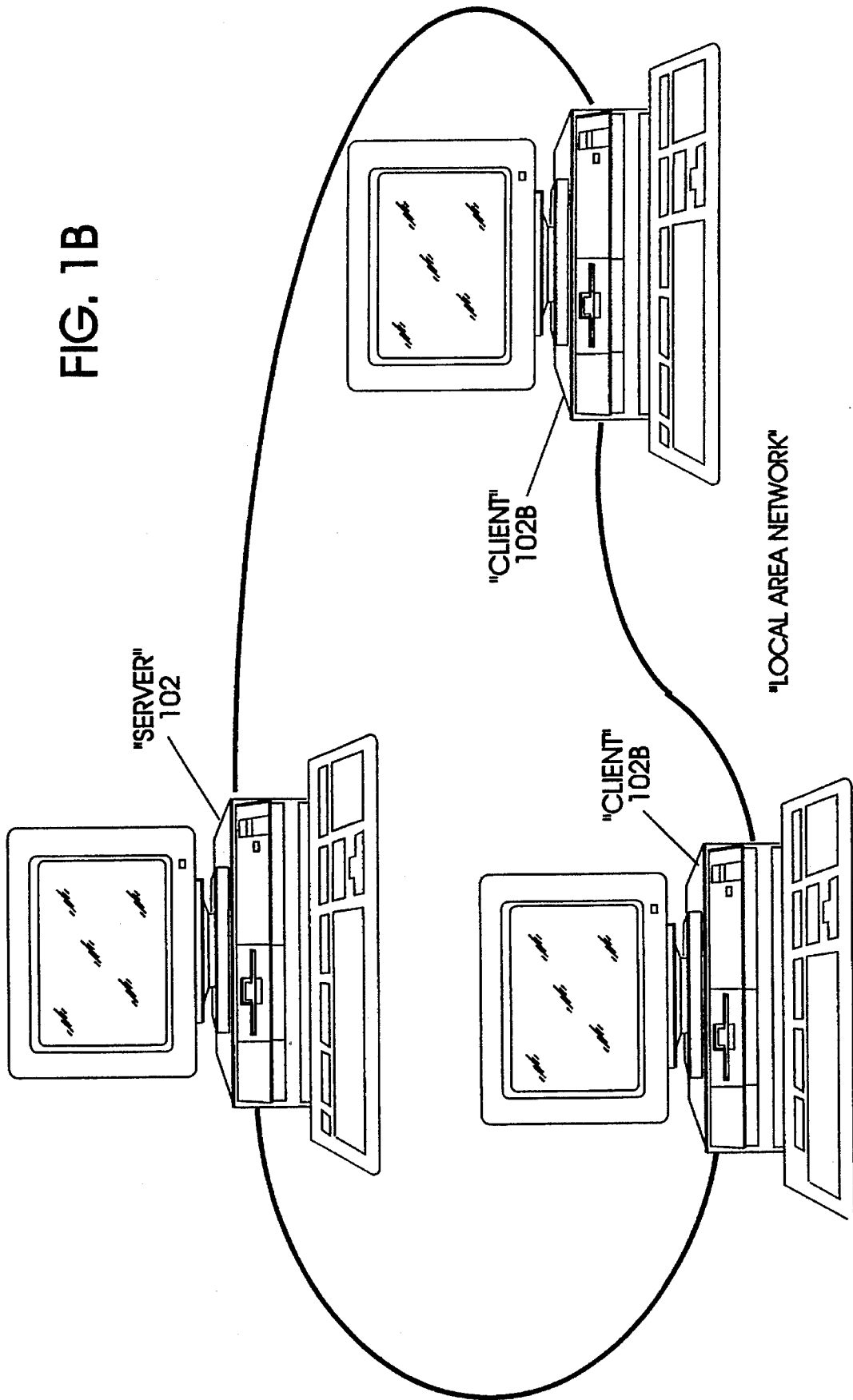
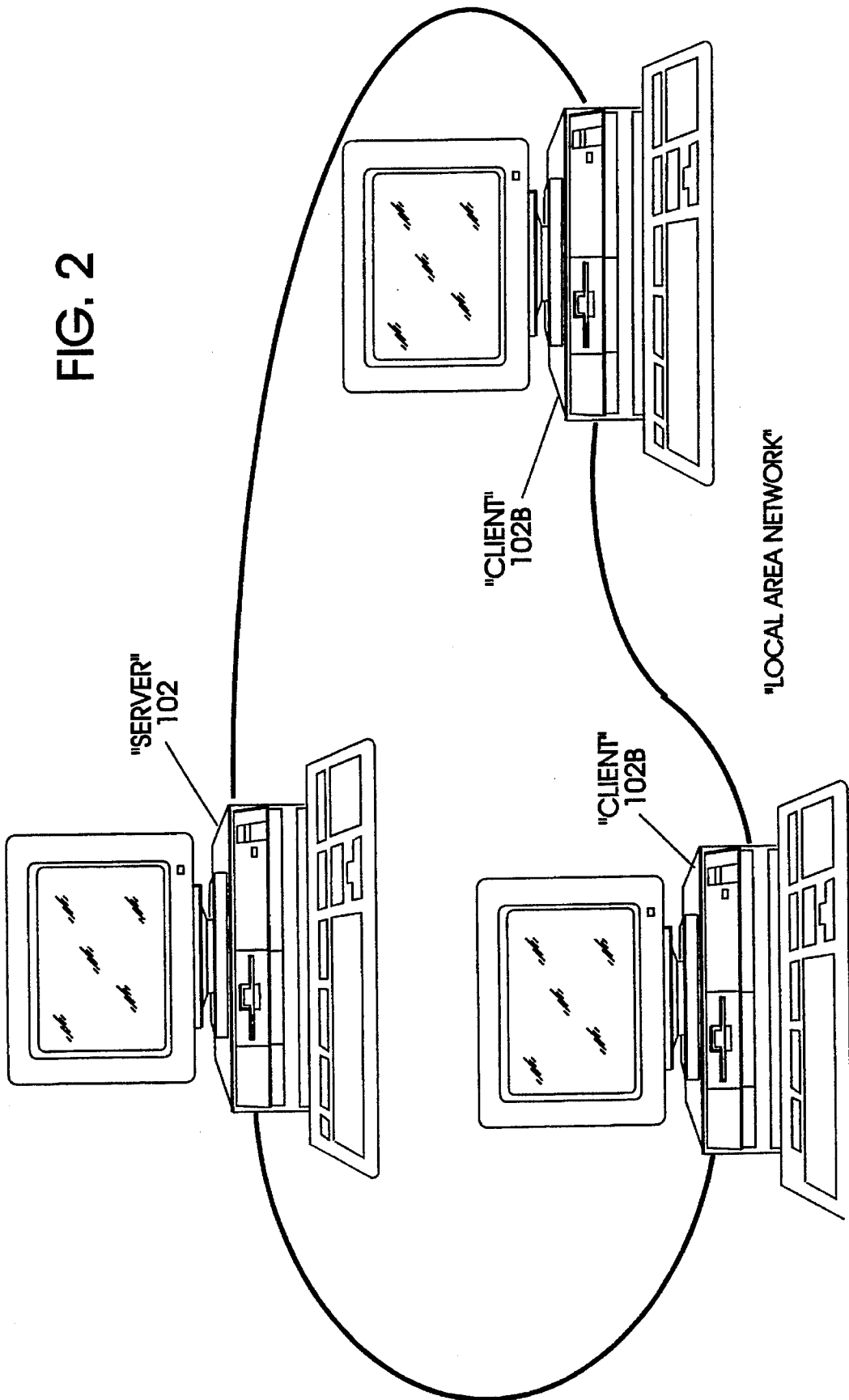
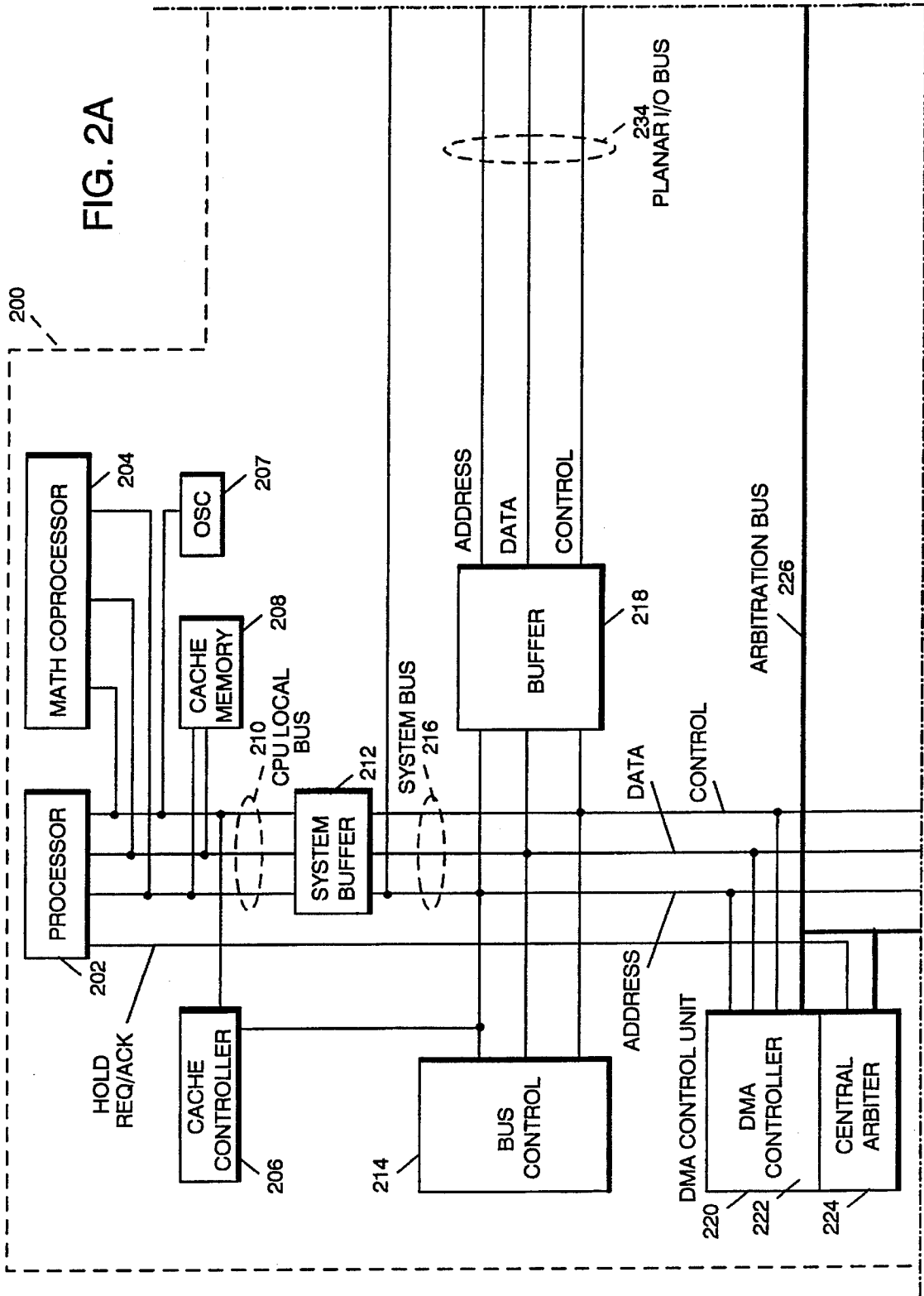


FIG. 2





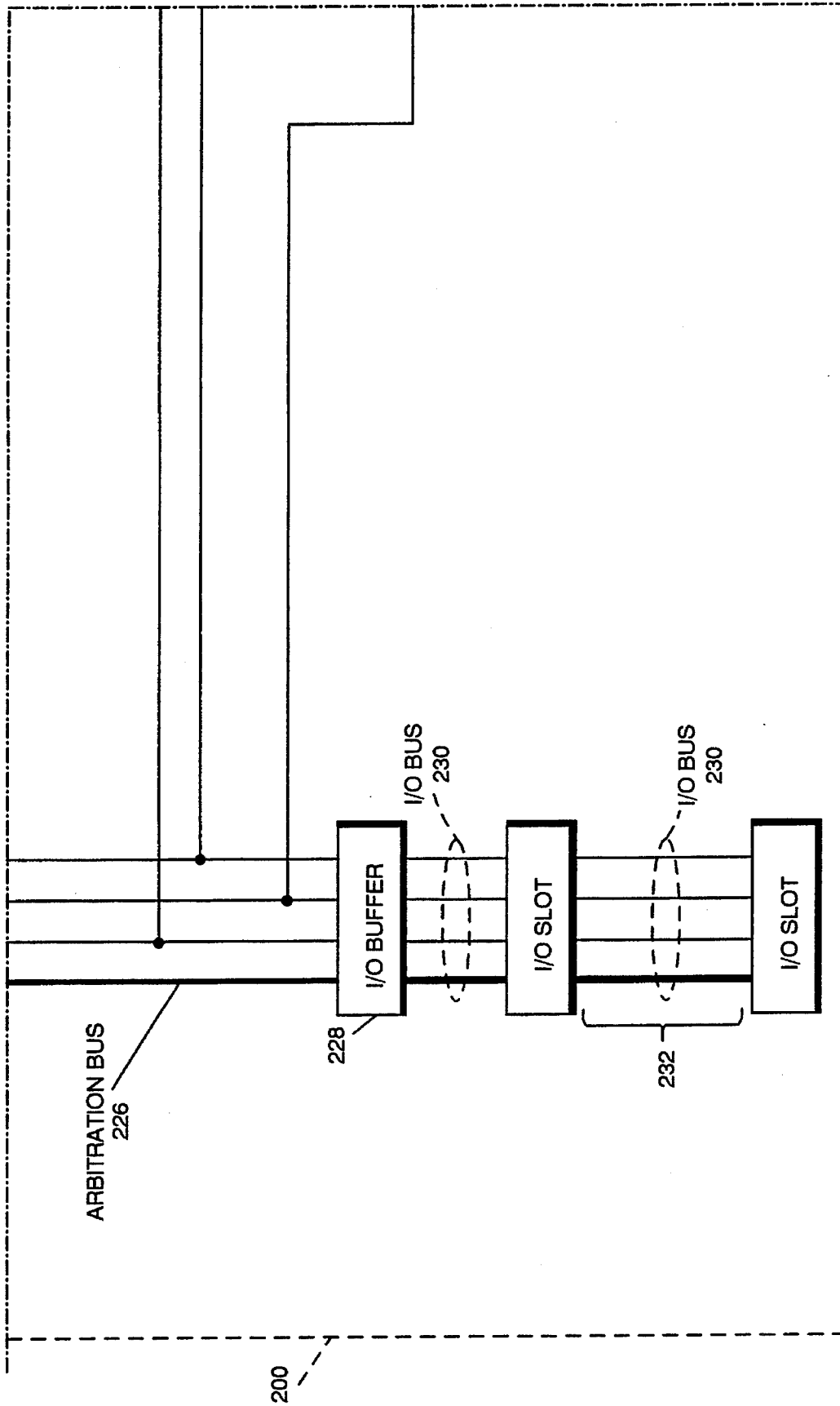
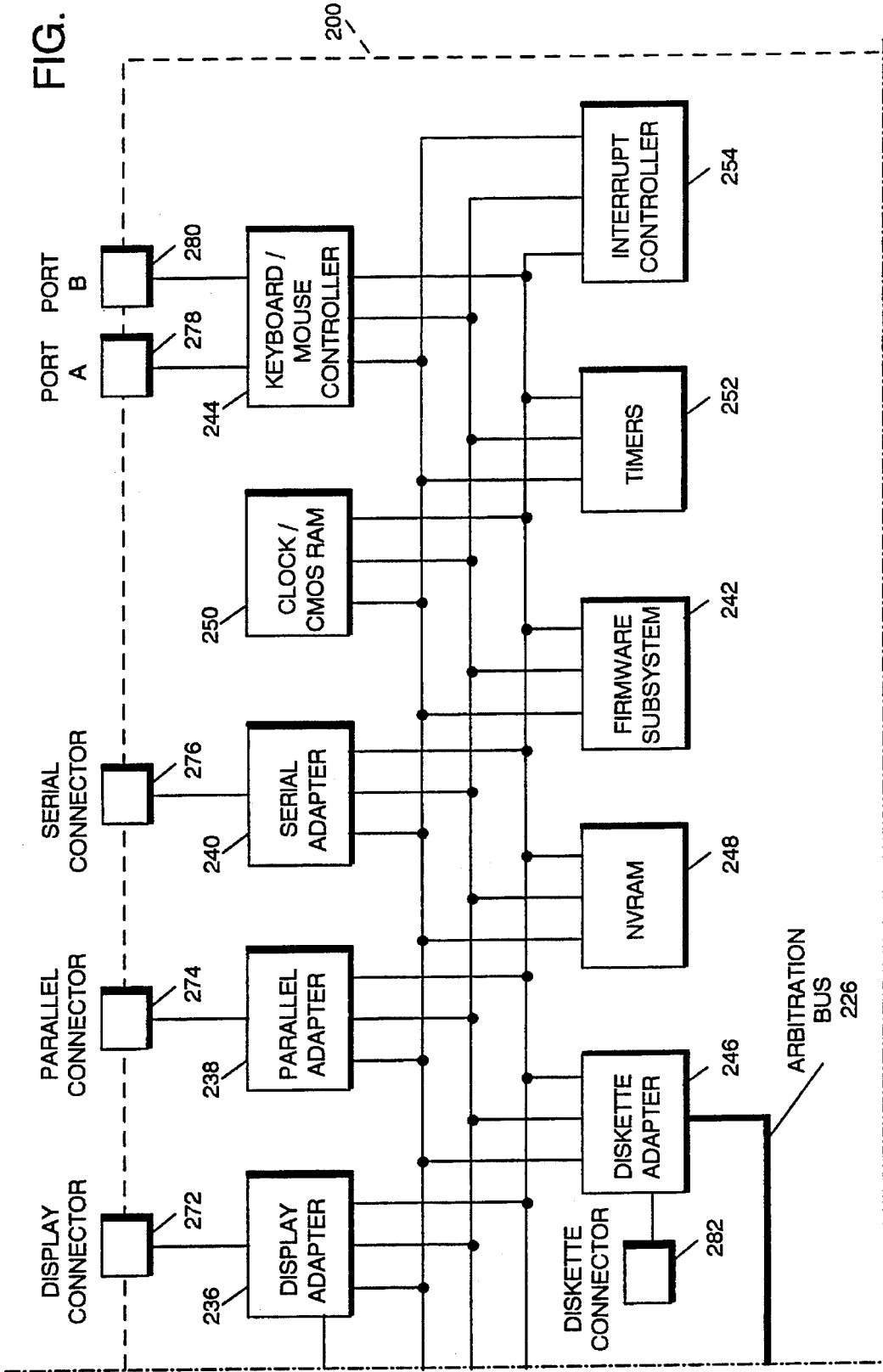


FIG. 2B

FIG. 2C



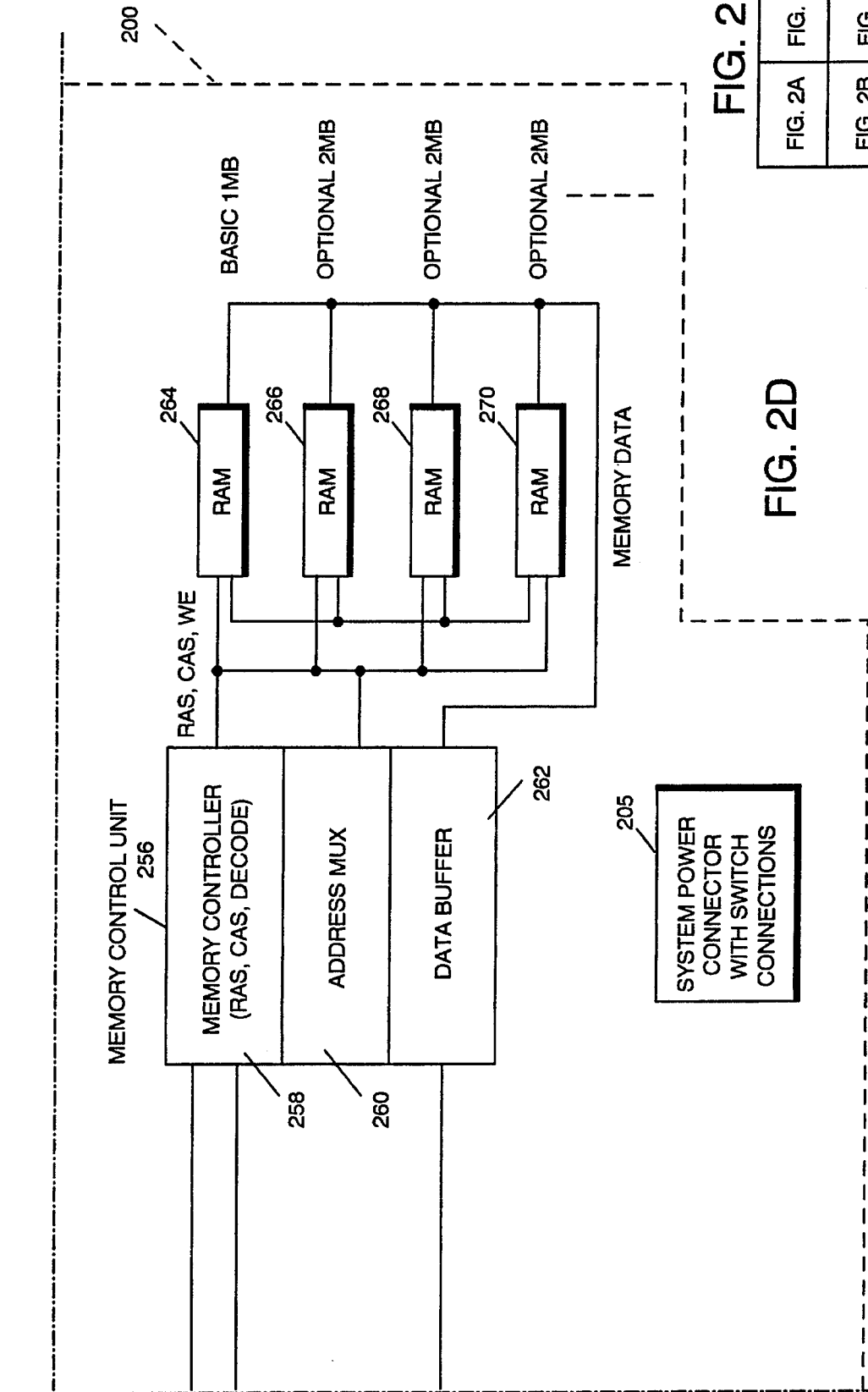
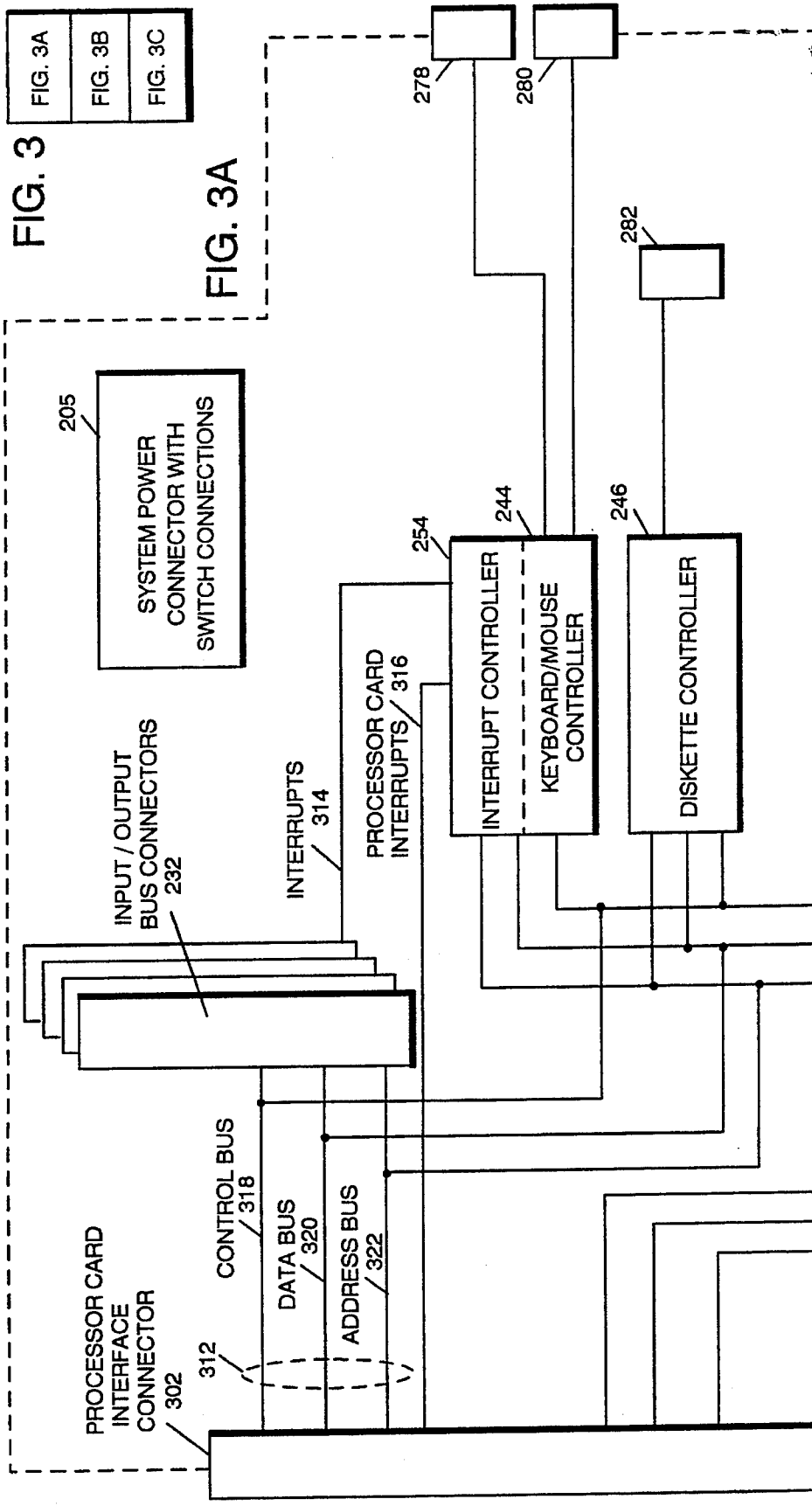


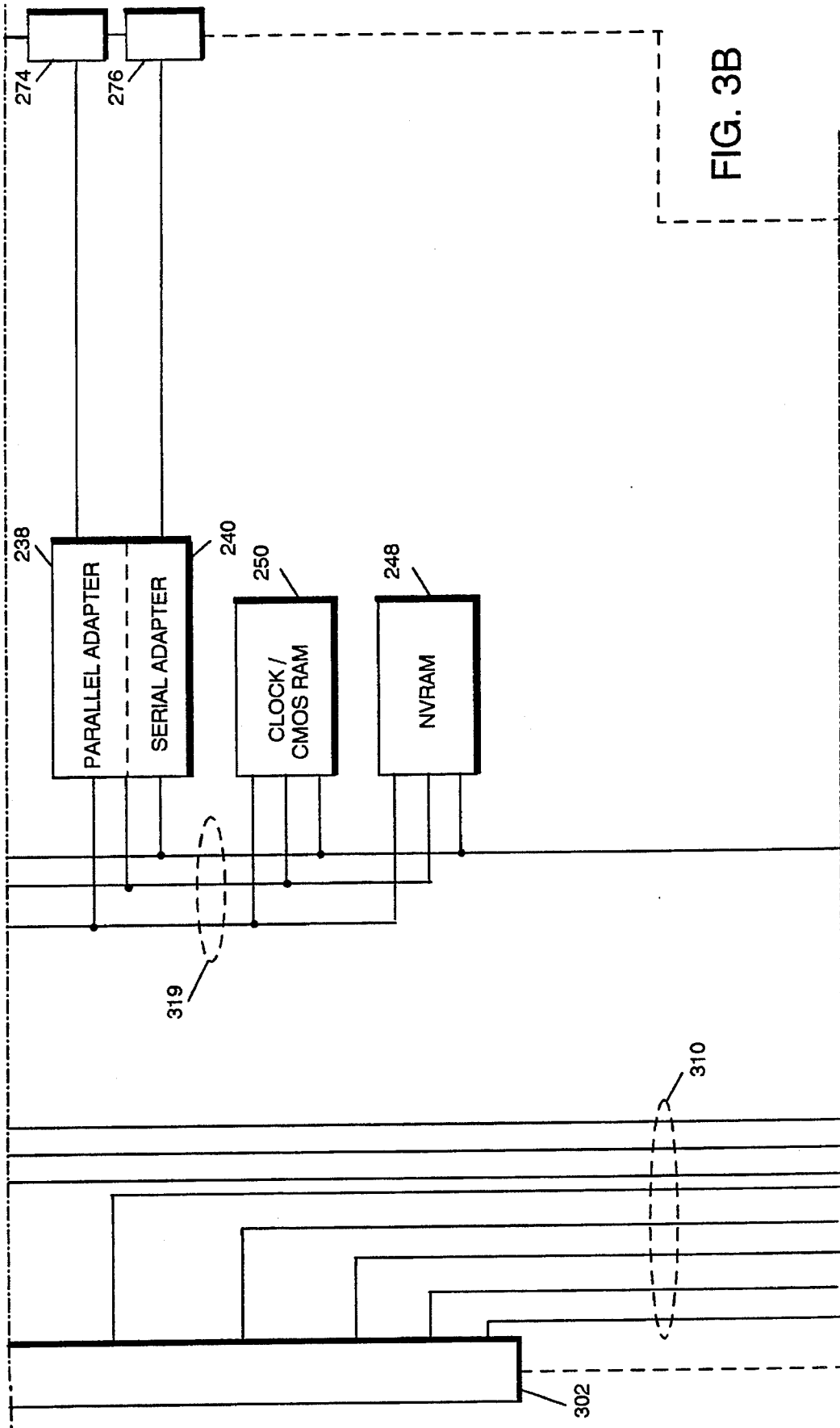
FIG. 2

FIG. 2A	FIG. 2C
FIG. 2B	FIG. 2D

FIG. 2D







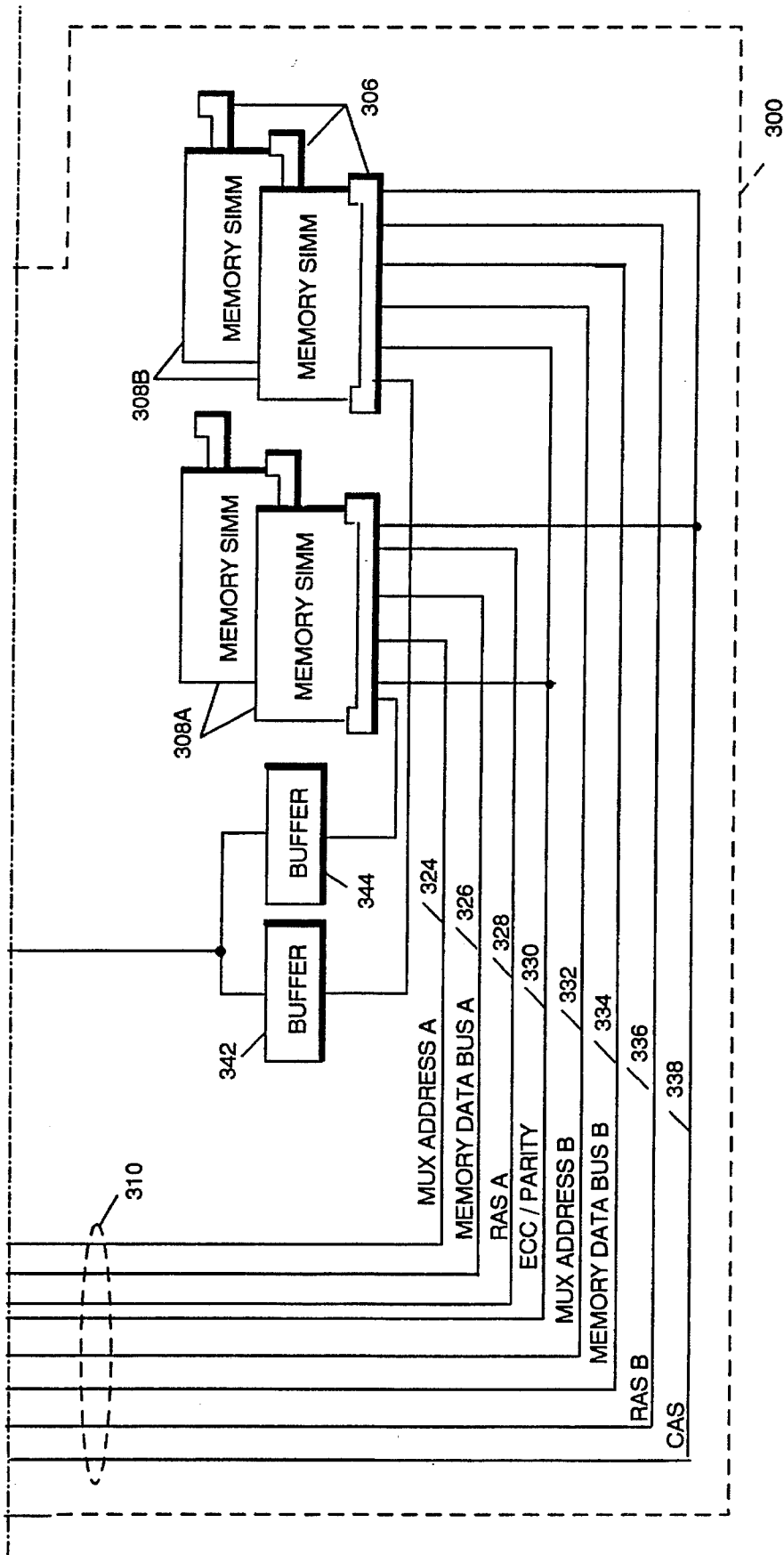


FIG. 3C

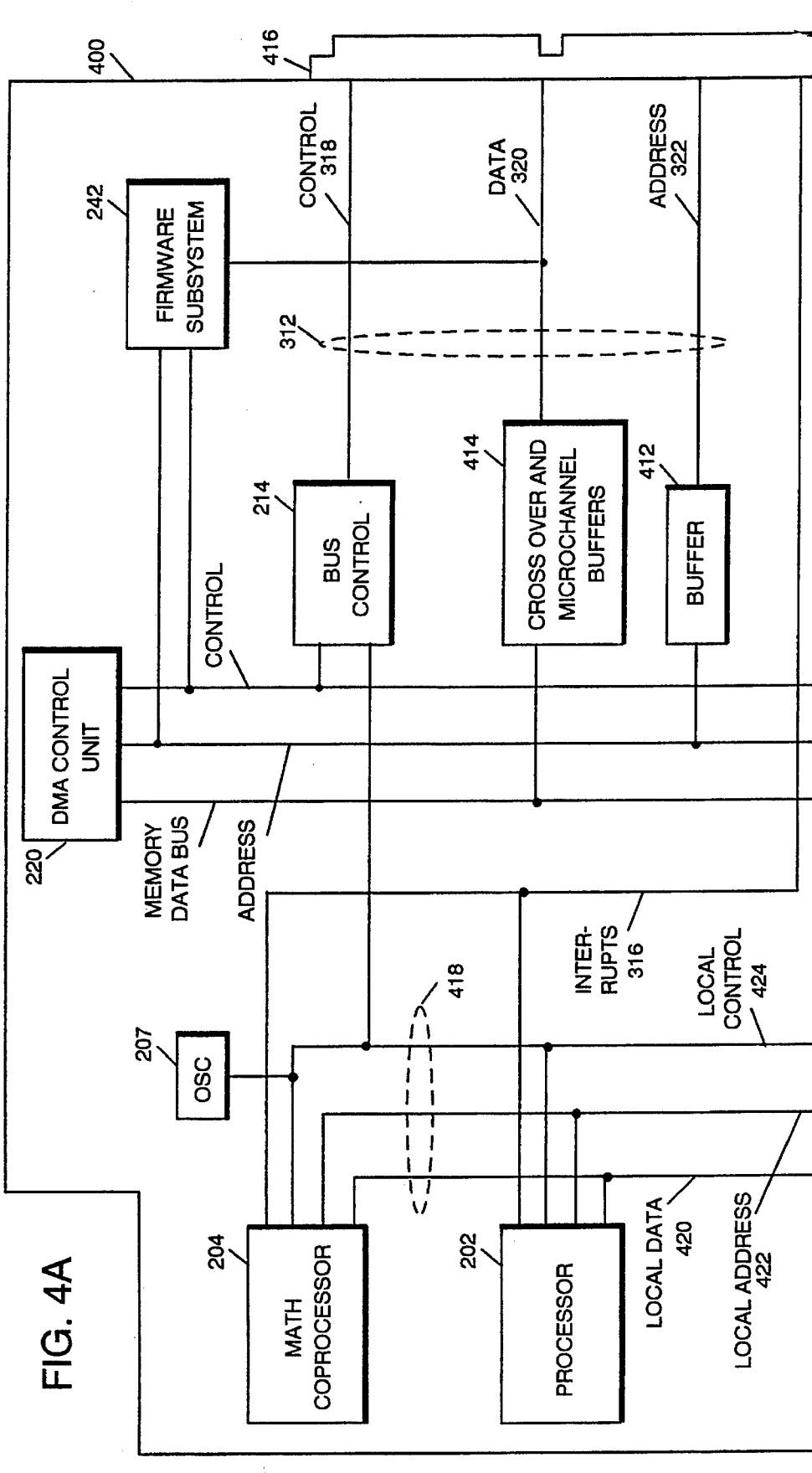


FIG. 4A

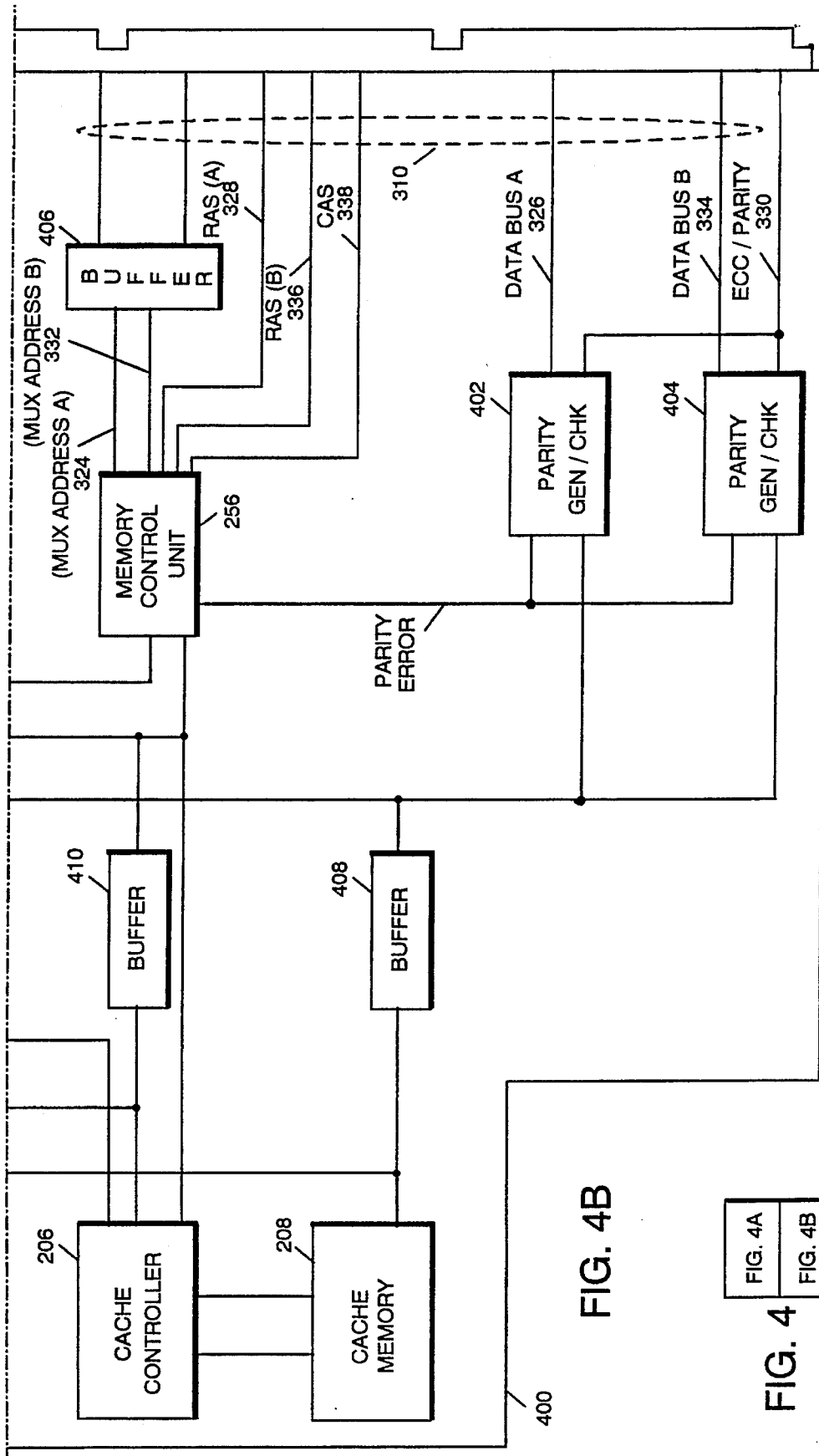


FIG. 4B

FIG. 4A
FIG. 4B

FIG. 4

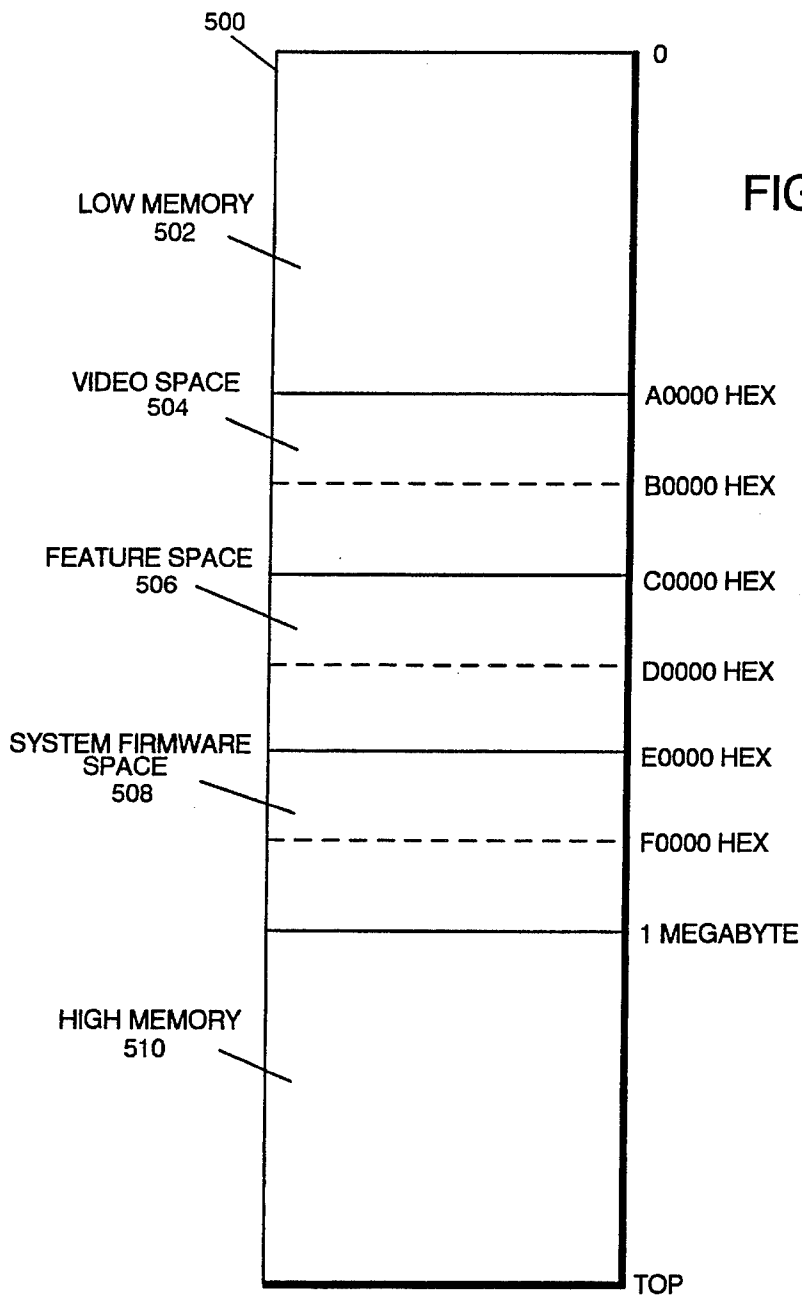


FIG. 5A

FIG. 5B

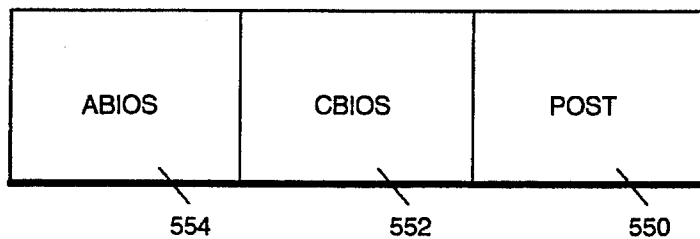


FIG. 6

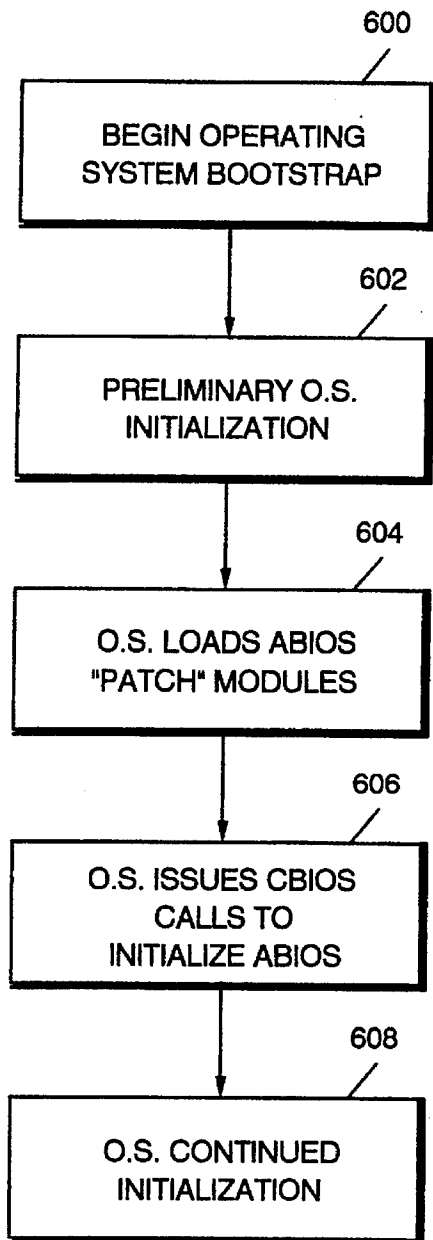


FIG. 8

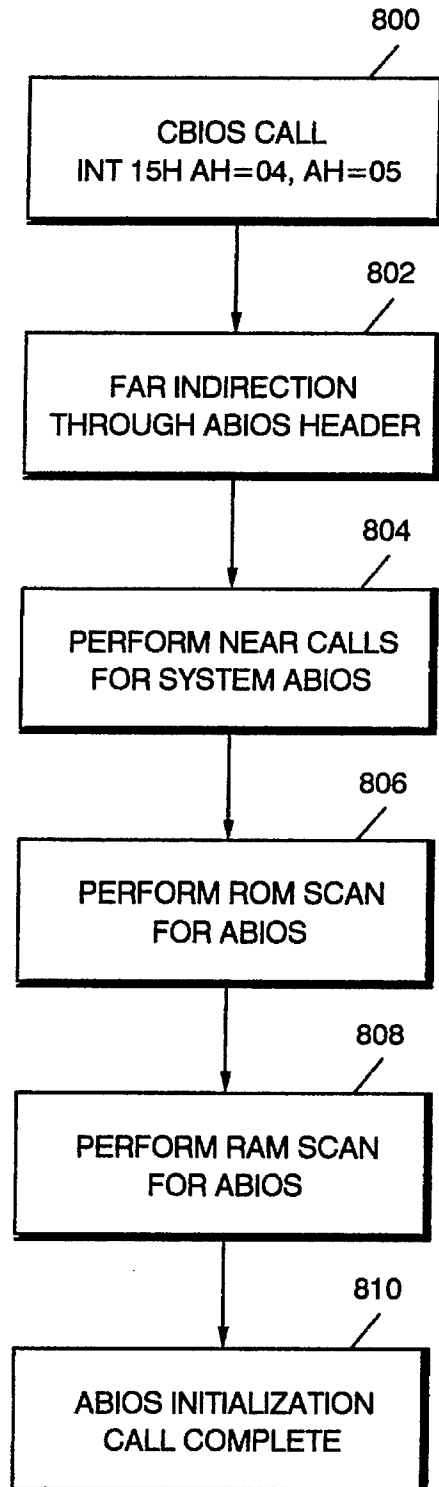


FIG. 7

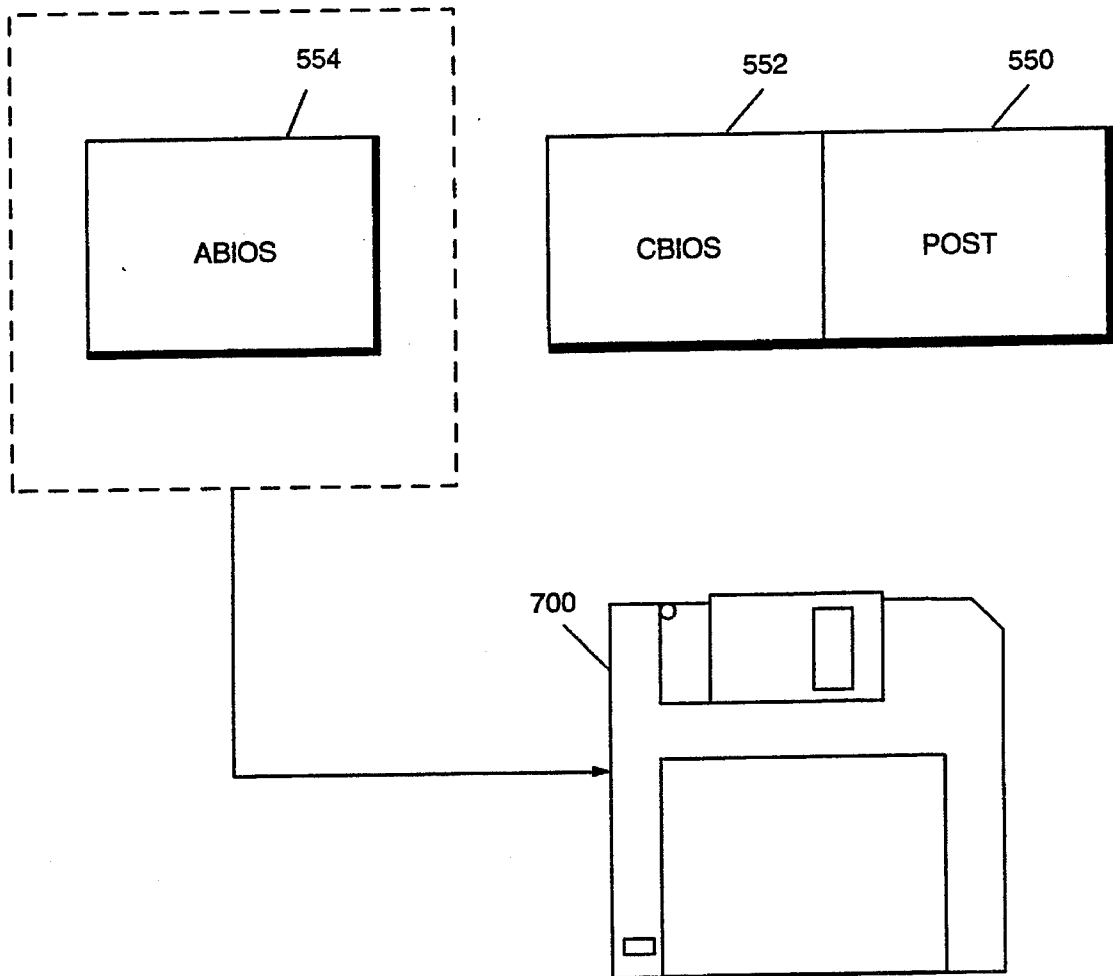




FIG. 9

Field	Offset	Length
Signature = AA55H (Word Value)	+00H	2
Length in 512-Byte Blocks	+02H	1
Model Byte	+03H	1
Submodel Byte	+04H	1
ROM Revision Level	+05H	1
Device ID	+06H	2
Number of Initialization Table Entries	+08H	1
Build Initialization Table Entry Point	+09H	3
Secondary Device ID	+0CH	1
Revision	+0DH	1
Extension Header Length in Bytes	+0EH	2
Reserved	+10H	2
Reserved	+12H	2
Initialization Routine Offset	+14H	2

**METHOD AND APPARATUS FOR DYNAMIC  
LOAD OF AN ABIOS DEVICE SUPPORT  
LAYER IN A COMPUTER SYSTEM**

**RELATED APPLICATIONS**

The following United States patent applications, which are filed on even date herewith, are incorporated by reference:

Application Ser. No. 07/902,330, Filed Jun. 22, 1992 Entitled "A METHOD AND APPARATUS FOR PROVIDING A MODULAR ABIOS DEVICE SUPPORT LAYER IN A COMPUTER SYSTEM"

Application Ser. No. 07/902,134, Filed Jun. 22, 1992 Entitled "A METHOD AND APPARATUS FOR CONFIGURING AND INSTALLING A LOADABLE ABIOS DEVICE SUPPORT LAYER IN A COMPUTER SYSTEM"

Application Ser. No. 07/902,315, Filed Jun. 22, 1992 Entitled "A METHOD AND APPARATUS FOR AN AUTOMATED DYNAMIC LOAD OF AN ABIOS DEVICE SUPPORT LAYER IN A COMPUTER SYSTEM"

**1. Field of the Invention**

This invention relates to personal computer systems and, more particularly, to a method and apparatus for loading and initializing firmware.

**2. Background of the Invention**

Personal computer systems in general, and IBM personal computers in particular, have attained widespread use for providing computer power to many segments of today's society. A personal computer system can usually be defined as a desk top, floor standing, or portable computer that includes a system unit having a system processor, a display monitor, a keyboard, one or more diskette drives, a fixed disk storage, an optional pointing device such as a "mouse," and an optional printer. These systems are designed primarily to give independent computing power to a single user or small group of users and are inexpensively priced for purchase by individuals or businesses. Examples of such personal computer systems are sold under the trademarks: IBM's PERSONAL COMPUTER, PERSONAL COMPUTER XT, PERSONAL COMPUTER AT and IBM's PERSONAL SYSTEM/2 Models 25, 30, 50, 55, 56, 57, 60, 65, 70, 80, 90 and 95 (hereinafter referred to as the IBM PC, XT, AT, and PS/2, respectively).

These systems can be classified into two general families. The first family, usually referred to as Family 1 Models, uses a bus architecture exemplified by the AT computer and other "IBM compatible" machines. The second family, referred to as Family 2 Models, uses IBM's MICRO CHANNEL bus architecture exemplified by IBM's PS/2 Models 50 through 95. The bus architectures used in Family 1 and Family 2 models are well known in the art.

Beginning with the earliest personal computer system of the Family 1 models, the IBM PC, and through the current Family 2 models, the system processor of the personal computer is from the Intel 86 Family of microprocessors. The Intel 86 Family of processors includes the 8088, 8086, 80286, 80386, and 80486 processors commercially available from Intel Corporation. The architecture of the Intel 86 Family of processors provides an upwardly compatible instruction set which assists in preserving software investments from previous processors in the 86 Family of processors. This upward compatibility preserves the software application base of the personal computers which use this

family of processors. A variety of commonly available and well known software operating systems, such as a DOS or an OS/2 operating system, operate on various members of the Intel 86 Family of processors.

The PC and XT computers use the Intel 8088 processor. The AT computers use the Intel 80286 processor. The PS/2 line spans several of the Intel processors. More specifically, a PS/2 Model 30, which is similar to the IBM PC and XT, uses an Intel 8086 processor. The PS/2 Models 50 and 60 both use the Intel 80286 processors. The Intel 80386 processor is used in the IBM PS/2 Model 80 and certain versions of the IBM PS/2 Model 70. Other versions of the IBM PS/2 Model 70, as well as the PS/2 Models 90 XP 486 and 95 XP 486, use the Intel 80486 processor.

The processors in the Intel 86 Family support a variety of operating modes. Real mode, which supports a one megabyte system address space, is the only operating mode of the 8088 and 8086 processors. The 80286 supports both a real and a protected operating mode. Protected mode provides a mode of operation which prevents an application from interfering with the operation of other applications or the operating system. The 80286 provides extended addressing capabilities, allowing up to sixteen megabytes of memory to be addressed directly. To maintain downward compatibility, the 80286 can be operated in real mode to emulate the real mode of the 8088 or 8086. The 80386 and 80486 can address up to four gigabytes of physical memory. The 80386 and 80486 also support a virtual 86 mode of operation. The virtual 86 mode supports the operational characteristics of the real mode within the overall confines of the protected mode environment.

With personal computers, software and hardware compatibility is of great importance. To provide software and hardware compatibility, an insulation layer of system resident code, also referred to as microcode, was established between the hardware and the software. This code provided an operational interface between a user's application program or operating system and the hardware device to relieve the user of the concern about the characteristics of hardware devices. Eventually, the code developed into a basic input/output system (BIOS), for allowing new hardware devices to be added to the system, while insulating the application program/operating system from the peculiarities of the hardware devices. The importance of BIOS was immediately evident because it freed a device driver from depending on specific hardware device characteristics while providing the device driver with an intermediate interface to the hardware device. Because BIOS was an integral part of the computer system and controlled the movement of data in and out of the system processor, it was resident on a system planar board of the system unit and was shipped to the user in either a read-only memory (ROM) or an erasable programmable read-only memory (EPROM). BIOS in the original IBM PC occupied 8K bytes (a kilobyte or "K byte" refers to a quantity of 1024 bytes) of ROM resident on the planar board. The ROM also contained a power-on self test (POST) program which was used to test and initialize the computer system. The accumulation of code resident in the computer system ROM became known as the "system firmware," or simply "firmware." Thus, the firmware included a POST portion and a BIOS portion. Sometimes, BIOS was defined to include the POST program.

As new models of the personal computer family were introduced, the firmware was updated and expanded to support new hardware devices such as new input/output (I/O) devices. As could be expected, the firmware started to increase in memory size. For example, with the introduction

of the IBM AT, the firmware required 32K bytes of ROM. With the introduction of the PS/2 line, a significantly new BIOS, known as Advanced BIOS, or ABIOS, was developed. However, to maintain software compatibility, BIOS from the Family 1 models had to be included in the Family 2 models. The Family 1 BIOS became known as Compatibility BIOS or CBIOS. Thus, BIOS evolved to include more than one type of BIOS. Present architectural definitions for personal computer systems allow for up to 128K bytes of system firmware address space.

Personal computer systems may be linked to form a network of computers (e.g., a Local Area Network (LAN)) so that users can exchange information, share I/O devices, and utilize a particular direct access storage device (DASD) such as a particular hardfile or diskette. Typically, the LAN includes a client and a server. A server is a computer system which includes a DASD for supplying the storage for one or more clients of the local area network. A client or server may require modifications, updates, extensions or maintenances of the system firmware.

Arrangements for storing, loading and initializing firmware are known. See, for example, commonly owned: U.S. patent application Ser. No. 07/521,050 entitled "Method and Apparatus for Selectively Reclaiming a Portion of RAM in a Personal Computer System," U.S. patent application Ser. No. 07/398,865, entitled "Initial BIOS Load for a Personal Computer System," U.S. patent application Ser. No. 07/777,844, entitled "Programmable Firmware Store for a Personal Computer System," U.S. patent application Ser. No. 07/799,486, entitled "Automated Programmable Firmware Store for a Personal Computer System," and U.S. patent application Ser. No. 07/590,749, entitled "Apparatus and Method for Loading BIOS into a Computer System from a Remote Storage Location," which are all incorporated herein by reference.

### SUMMARY OF THE INVENTION

It has been discovered that by linking a first portion of microcode, e.g., a CBIOS portion, to a second portion of microcode, e.g., an ABIOS portion, it is possible to remove the ABIOS portion of microcode from the address space which is reserved for microcode.

A principal object of the present invention is removing the ABIOS program from the "ROM" address spaces of a personal computer system.

Another object of the present invention is to provide more "free" space in the system and feature ROM address space.

Yet another object of the present invention is loading an ABIOS device support software layer in an "on demand" mode.

Yet a further object of the present invention is to continue to provide ABIOS support in a compatible fashion on both disk-based and medialess computer systems.

### BRIEF DESCRIPTION OF THE DRAWING

Further and still other objects of the present invention will become more readily apparent in light of the following description when taken in conjunction with the accompanying drawing, in which:

FIG. 1A is a perspective view of a typical personal computer system;

FIG. 1B is a diagram of a typical local area network;

FIG. 2 is a block schematic diagram of a unified planar board for the computer system of FIG. 1A;

FIG. 3 is a block schematic diagram of an alternative planar board for the computer system of FIG. 1A;

FIG. 4 is a block schematic diagram of a processor card for use with the alternative planar board of FIG. 3;

FIG. 5A is a diagram of the system address space for the computer system of FIG. 1A;

FIG. 5B is a diagram of a typical firmware memory map;

FIG. 6 is a flow diagram of a system initialization sequence as it pertains to ABIOS;

FIG. 7 is a diagram of an enhanced delivery mechanism for ABIOS;

FIG. 8 is a flow diagram of the internals of ABIOS initialization and

FIG. 9 is an ABIOS program module header.

### DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

Referring to FIG. 1A, there is shown a personal computer system 100 which employs the present invention. The personal computer system 100 includes a system unit 102 having a suitable enclosure or casing 103, output device or monitor 104 (such as a conventional video display), input devices such as a keyboard 110, an optional mouse 112, and an optional output device such as a printer 114. Finally, the system unit 102 may include one or more mass storage devices such as a diskette drive 108 (operable with a diskette—not shown) and a hard disk drive (hardfile) 106.

Referring to FIG. 1B, the system unit 102 may be connected electronically in a well-known manner with other system units 102B to form a local area network (LAN). In the LAN, system unit 102 functions as the server and system units 102B function as the clients. System units 102B are identical to the unit 102, except that units 102B include no drives 106, 108 and are thus referred to as medialess clients. Other conventional I/O devices may be connected to the system units 102, 102B for interaction therewith.

Referring to FIG. 2, there is shown a block diagram of a unified planar 200 of the system unit 102. The planar 200 includes a printed circuit board (PCB) 201 upon which are mounted or connected a number of input/output bus connectors 232 having I/O slots, a processor 202 which is connected by a high speed CPU local bus 210 under control of a bus control unit 214 to a memory control unit 256. The unit 256 is further connected to a main memory such as volatile random access memory (RAM) 264. Any appropriate processor 202 can be used such as an Intel 80386, Intel 80486 or the like. A system power connector 205 is mounted on the PCB 201 for connection to a power unit (not shown) that supplies the necessary power for the system 100.

The CPU local bus 210 (comprising address, data and control components) provides for the interconnection of the processor 202, an optional math coprocessor 204, an optional cache controller 206, and an optional cache memory 208. Also coupled onto the CPU local bus 210 is a system buffer 212. The system buffer 212 is itself connected to a system bus 216 which comprises address, data and control components. The system bus 216 extends between the system buffer 212 and an I/O buffer 228. The system bus 216 is further connected to the bus control unit 214 and to a direct memory access (DMA) control unit 220. The DMA control unit 220 includes a central arbiter 224 and a DMA controller 222. The I/O buffer 228 provides an interface between the system bus 216 and an I/O bus 230. An oscillator 207 is connected as shown for providing suitable

clock signals to the computer system 100. Those skilled in the art will recognize that while the preferred embodiment is implemented on the MICRO CHANNEL bus of an IBM PS/2 computer system, which is well known in the art, alternative bus architectures could also be used to employ the invention.

Connected to the I/O bus 230 is a plurality of I/O bus connectors having slots 232 for receiving adapter cards (not shown) which may be further connected to I/O devices or memory (e.g., hardfile 106). Two I/O connectors 232 are shown for convenience, but additional I/O connectors may easily be added to suit the needs of a particular system. An arbitration bus 226 couples the DMA controller 222 and the central arbiter 224 to the I/O connectors 232 and a diskette adapter 246. Also connected to the system bus 216 is the memory control unit 256 which includes a memory controller 258, an address multiplexer 260, and a data buffer 262. The memory control unit 256 is further connected to the main memory such as the random access memory as represented by the RAM module 264. The memory control unit 256 includes logic for mapping addresses to and from the processor 202 to and from particular areas of the RAM 264. While the system 100 is shown with a basic one megabyte RAM module 264, it is understood that additional memory can be interconnected as represented in FIG. 2 by optional memory modules 266, 268, 270.

A buffer 218 is coupled between the system bus 216 and a planar I/O bus 234. The planar I/O bus 234 includes address, data, and control components. Coupled along the planar I/O bus 234 are a variety of I/O adapters and other peripheral components such as a display adapter 236 (which is used to drive the optional display 104), a clock/CMOS RAM 250, a nonvolatile RAM 248 (hereinafter referred to as NVRAM), a serial adapter 240 (other common terms used for "serial" are "asynchronous" and "RS232"), a parallel adapter 238, a plurality of timers 252, the diskette adapter 246, a keyboard/mouse controller 244, an interrupt controller 254, and a firmware subsystem 242. The firmware subsystem typically includes a nonvolatile program store (e.g., ROM) which contains the POST and BIOS programs.

The clock/CMOS RAM 250 is used for time of day calculations. The NVRAM 248 is used to store system configuration data. That is, the NVRAM 248 will contain values which describe the present configuration of the system 100. The NVRAM 248 contains information which describes, for example, adapter card initialization data, capacity of a fixed disk or a diskette, the amount of main memory, etc. Furthermore, these data are stored in NVRAM 248 whenever a configuration program is executed. This configuration program can be a conventional Set Configuration program provided on a system Reference Diskette included with IBM PS/2 computer systems. The Reference Diskette is sometimes referred to as a diagnostic, maintenance or service diskette. The purpose of the configuration program is to store values characterizing the configuration of this system 100 to NVRAM 248 which are saved when power is removed from the system. The NVRAM can be a low power CMOS memory with a battery backup.

Connected to the keyboard/mouse controller 244 are a port A 278 and a port B 280. These ports A,B are used to connect the keyboard 110 and the mouse 112 to the personal computer system 100. Coupled to the serial adapter 240 is a serial connector 276. An optional device such as a modem (not shown) can be coupled to the system through this connector 276. Coupled to the parallel adapter 238 is a parallel connector 274 to which a device such as the printer 114 can be connected. Connected to the diskette adapter 246

is a diskette connector 282 used to attach one or more diskette drives 108.

Referring to FIGS. 3 and 4, system unit 102 may also use a planar board 300 and a processor card 400 (FIGS. 3 and 4). The processor card 400 is removably mounted on and is electrically connected to the planar board 300. Identical element numbers of FIG. 2 correspond to identical elements in FIGS. 3 and 4.

Referring now to FIG. 3, the planar board 300 comprises a printed circuit board (PCB) 301 upon which are mounted (e.g., surface mounted) various components that are interconnected by wiring or circuits in the PCB. Such components include a suitable commercially available electrical connector 302 into which an edge 416 of the processor card 400 is plugged for removably mounting and electrically connecting the processor card 400 to the planar board 300. A plurality of single in-line memory module (SIMM) connectors 306 is also mounted on the PCB 301 for connecting to memory banks 308A, 308B forming the system main memory or RAM. One or more I/O bus or expansion connectors 232 are also mounted on the PCB 301 for connection to different expansion adapters and options that may be added or incorporated into the personal computer system 100. For example, the fixed disk drive 106 may be connected to an adapter card (not shown) having a disk controller which is connected to a connector 232. Preferably, each connector 232 is a commercially available connector of the type conforming to the above-mentioned MICRO CHANNEL architecture.

Also mounted on the planar board 300 are an interrupt controller 254 and a keyboard/mouse controller 244 which are connected to keyboard and mouse connectors 278, 280, a diskette controller or adapter 246 connected to a diskette connector 282, and serial and parallel adapters 240, 238 connected to serial and parallel connectors 276, 274 which allow the various I/O devices to be connected into the system. A system power connector 205 is mounted on the PCB 301 for connection to a power unit (not shown) that supplies the necessary power for the system. A nonvolatile memory (NVRAM) 248 and a time-of-day clock/CMOS RAM 250 are also mounted on the PCB 301. The PCB 301 also has mounted thereon various oscillators (not shown) to provide timing signals, and buffers 342, 344 (not all shown) to isolate sections of the circuitry in a manner well known.

The wiring of PCB 301 interconnects the various components as shown in the drawing and is grouped into three groupings, a memory bus 310 (including lines 324-338), a channel bus 312 (including an address bus 322, a data bus 320 and a control bus 318), and miscellaneous signal lines including interrupt lines 314, 316, all of which are connected to counterpart wiring on the PCB 401 through the connectors 302, 416. Tapped off the bus 312 is a planar function bus 319.

Referring to FIG. 4, there is shown the processor card 400 for removably mounting on the planar board 300. The processor card 400 comprises a printed circuit board (PCB) 401 having mounted (e.g., surface mounted) thereon a plurality of commercially available components including a processor 202, an optional math coprocessor 204, an optional cache controller 206, an optional cache memory 208, a direct memory access (DMA) control unit 220, a bus control unit 214, a memory control unit 256, a firmware subsystem 242, and parity checking units 402, 404. The processor 202 preferably is a high performance type, such as an Intel 80486, having thirty-two bit data paths and providing thirty-two bit addressing capability. Of course, Intel

80386 and the like processors can be used. The remaining components are selected in conventional fashion for their compatibility with such processor. A plurality of buffers **406**, **408**, **410**, **412**, **414** is connected as shown. The buffers provide selective isolation or connection between the circuits allowing different portions to be used concurrently, for example, to move data between the processor **202** and the cache memory **208** while other data is being transferred between an I/O unit and the main memory **308A**, **308B**. All of the above components are electrically connected to each other as appropriate by printed wiring circuits in PCB **401** which terminate at the edge connector **416**. The edge connector **416** is pluggable into the edge connector **302** on the planar board **300** shown in FIG. 3 so that the planar board **300** and the processor card **400** are electrically and mechanically interconnectable.

The wiring circuits of the PCB **401** include a local bus **418** including data, address and control lines **420**, **422**, **424**, respectively, which interconnect the processor **202** with an optional math coprocessor **204**, an optional cache controller **206** and an optional cache memory **208**, as shown in FIG. 4. The remaining circuit lines generally include interrupt lines **316**, channel bus lines **312** and memory bus lines **310**. The channel bus lines **312** include control, data and address bus lines **318**, **320**, **322**, respectively. Memory bus lines **310** include multiplexed memory address lines **324**, **332**, row address strobe (RAS) lines **328**, **336** for memory banks **308A**, **308B**, column address strobe (CAS) line **338**, data bus A and B lines **326** and **334**, and a line **330** for use in error checking via parity check or ECC checking. An oscillator **207** is connected as shown for providing suitable clock signals to the computer system **100**. For simplicity, certain miscellaneous lines, such as reset, grounds, power-on, etc. have been omitted from FIGS. 2, 3 and 4.

During operation of a personal computer system **100** having a board **300** and a card **400**, the card **400** is electrically and mechanically connected to the board **300** and typically lies in a plane perpendicular to the board **400**.

#### ABIOS Load and Access

System firmware includes the Power-On Self Test program (POST) and the Basic Input Output System program (BIOS). BIOS further includes the compatibility BIOS or CBIOS and the advanced BIOS or ABIOS. POST is the set of instructions which execute when the system is first powered-on to initialize the personal computer system **100**. BIOS is the set of instructions which facilitates the transfer of data and control instructions between the processor **202** and I/O devices.

In the medialess environment, a medialess system unit (e.g., **102B**) includes a suitable network adapter or card (not shown) for providing a Remote Initial Program Load (RIPL) facility within the unit **102B**. The card is connected, for example, to one of the connectors **232**. The RIPL program permits booting an operating system from a network server **102** rather than from a local storage device such as the fixed disk **106** or the diskette **108**. RIPL is also referred to as simply Remote Program Load or RPL, and the terms are used interchangeably. RPL is well understood in the art.

POST contains a bootstrap program which attempts to locate a boot device and load a boot record. Typically, the boot device is hardfile **106** or diskette drive **108**. Diskette drive **108** requires a boot or operating system diskette to operate. If POST successfully loads a boot record from a boot device, then POST transfers control to the boot record,

completing the operation of the POST bootstrap program. If a boot record was unable to be loaded and a RPL adapter is present, then POST transfers control to an RPL program. If no RPL program is present, then POST prompts the user indicating that a boot source is required. CBIOS is essential to the bootstrap operation of the computer. CBIOS provides a number of services including access to the hardfile **106** and diskette drive **108**. ABIOS is demand initialized and normally not a required part of the POST bootstrap process.

Referring now to FIG. 5A, there is shown a memory map of the system address space **500** for a client **102B** or server **102** (FIG. 1B). The system address space **500** includes a plurality of memory address regions **502**, **504**, **506**, **508**, **510** which are addressable by the processor **202**. The low memory region **502** is the traditional compatibility space in which real mode programs operate. For example, both DOS and CBIOS are real mode programs. The video region **504** occupies a 128K byte address space beginning at a physical location **A0000** hex up through a physical location **BFFFF** hex or the **A000** hex and **B000** hex segments. (A segment is a real mode term used to describe a 64K byte region which is aligned on a 16 byte boundary, i.e., a paragraph boundary.) The video region **504** provides video regeneration buffers; data which are stored in these buffers define what a user sees on the computer display **104**. The feature region **506** occupies a 128K byte address space beginning at a physical location **C0000** hex up through a physical location **DFFFF** hex or the **C000** hex and **D000** hex segments. The feature region **506** is used to store adapter firmware; this region may also provide buffer space. For example, a small computer system interface (SCSI) disk controller adapter includes an adapter firmware ROM containing POST, CBIOS and ABIOS programs which during operation is stored in feature region **506**. The system firmware region **508** occupies a 128K byte address space beginning at a physical location **E0000** hex up through a physical location **FFFFF** hex or the **E000** hex and **F000** hex segments. The system firmware region **508** stores the system firmware including POST and BIOS. The region above a one megabyte boundary is a high or extended memory region **510**. High memory **510** is utilized by operating systems such as the OS/2 operating system.

Extended memory refers to memory above the one megabyte address location. Expanded memory refers to memory which is bank switched into an area below the one megabyte address location. Expanded memory bank switching provides real mode applications with the ability to address more physical memory than the real mode address space directly allows. Expanded memory operation requires an available address space or window to be present in the real mode address space. Expanded memory operation also requires an expanded memory program to manage the bank switching. A common location for the expanded memory window is in the feature space **506** as well as the system firmware space **508**. Maximizing the amount of available space for expanded memory windows is a competitive requirement for personal computer systems.

Referring to FIG. 5B, a firmware image includes an ABIOS portion **554**, a CBIOS portion **552**, and a POST portion **550**. For an adapter card firmware such as the IBM SCSI adapter card, the size of the ABIOS portion is approximately 11K bytes. For the system firmware such as the IBM PS/2 Model 95 XP 486, the size of the ABIOS portion is approximately 32K bytes. With both the adapter card firmware and the system firmware, ABIOS uses substantial portion of the overall address space. In the system firmware case, an ABIOS size of approximately 32K bytes occupies

25% of the system firmware space 508. For a computer system with eight feature slots, an equal distribution of feature firmware space results in 16K bytes of feature space 506 per slot or adapter. An ABIOS size of 11K bytes uses approximately 70% of the 16K byte range. Additionally, some adapters exceed the 16K byte range.

Referring now to FIG. 6, a flow diagram of an operating system bootstrap and initialization process is shown. After the computer system is powered on and the POST completes, the operating system begins to bootstrap, step 600. The operating system then begins some of its preliminary initialization, step 602. The type and scope of operating system preliminary initialization is well known. The operating system then loads the ABIOS patch modules or files, step 604. These patch files, normally resident on the operating system boot device, provide functional corrections and enhancements to existing resident ABIOS program code. The ABIOS program modules can be listed in a control file such as an ABIOS.SYS. This ABIOS.SYS list can be a text file containing the names of the ABIOS modules such as MODULE1.BIO, MODULE2.BIO, etc. The naming convention chosen is a matter of design choice. After the operating system has loaded the ABIOS patch module or modules, step 604, the operating system can then initialize ABIOS, step 606. ABIOS is initialized by issuing CBIOS function calls. Once ABIOS is initialized, step 606, the operating system can continue further initialization, step 608.

CBIOS provides two function calls to support the initialization of ABIOS. These calls are accessed through a CBIOS system services interface defined as software interrupt 15 hex (INT 15H). The first function call is a build system parameter table function which is indicated by the value of 04 being placed in the AH register (a shorthand notation for this function call is AH=04). The entry requirements of the AH=04 call are a pointer to a memory buffer where the caller wants the system parameters table to be built (this pointer is indicated by the notation ES:DI) and a segment with an assumed offset of zero which indicates the ABIOS extension area (this segment is indicated by the notation DS and the corresponding pointer would be DS:O). The second function call is a build initialization table function which is indicated by the value of 05 being placed in the AH register (AH=05). The entry requirements for the AH=05 call are a pointer to a memory buffer where the caller wants the initialization table to be built (ES:DI) and a segment with an assumed offset of zero which indicates the ABIOS extension area (DS). ABIOS initialization is more clearly defined in the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference Manual*.

Referring to FIG. 7, in an enhanced delivery mechanism for ABIOS, the ABIOS 554 is physically separated from the remaining firmware, and packaged as a file on diskette 700 or other media. By being physically separated from the remaining firmware, the space occupied by ABIOS in firmware address space 506, 508 is freed. Moving ABIOS 554 applies to both the system firmware ABIOS as well as feature space ABIOS. Accordingly, the space normally occupied by ABIOS in system firmware space 506 is freed and the space normally occupied by ABIOS in feature space 508 is freed.

Referring to FIG. 8, a flow diagram of ABIOS initialization is shown. The INT 15H ABIOS initialization calls AH=04 and AH=05 are to be done sequentially, AH=04 then AH=05. An operating system, or other software program, accesses ABIOS initialization through the CBIOS system services calls AH=04 and AH=05, step 800. The AH=04 and

AH=05 calls have similar control sequences. Step 802 is the redirection of control of the present invention. The CBIOS system services call must redirect control to the ABIOS initialization program now present in the RAM extension area. Once CBIOS system services to begin ABIOS initialization has been invoked, step 800, the CBIOS system services code resident in the system firmware first must locate and transfer control to the now RAM resident ABIOS initialization program. This is done through the previously described "DS" register which specifies the segment address as to where the RAM ABIOS area begins. The CBIOS system services program, present in the system firmware, obtains the system services call from step 800 and must redirect this call to the ABIOS initialization code now in RAM. This redirection is performed through an indirect call through an entry in the ABIOS header which begins at offset zero of the segment specified by the "DS" register. This ABIOS header is described in more detail in conjunction with FIG. 9. A far call is performed to the address specified by DS: [ABIOSINIT], step 802, thus transferring the ABIOS initialization operation to the ABIOS program now in system RAM. ABIOSINIT is discussed in greater detail in conjunction with FIG. 9. A far jump can be used in place of the far call and the selection of call or jump is a matter of well known design choice. ABIOS initialization then performs a set of near call operations designed to initialize ABIOS, step 804. ABIOS initialization then performs a ROM scan operation designed to locate and initialize ABIOS present in the adapter firmware address space, step 806. A ROM scan is performed by searching for a specific header pattern in the feature space or adapter firmware space (see FIG. 5A). ABIOS initialization then performs a RAM scan designed to locate and initialize ABIOS present as a RAM loaded extension or patch, step 808. The current ABIOS module, the one containing the ABIOS initialization program, need not be searched for in RAM scan thus RAM scan begins after the current module. Once RAM scan has completed, step 808, the system services call to initialize ABIOS is complete, step 810 and control returns to the caller.

Referring now to FIG. 9, an ABIOS module header is shown. The field at offset +14H (an offset of 14 hex bytes or 20 decimal bytes), which prior to the present invention was a reserved field, stores an offset of the ABIOS initialization programs. The table entry at +14H can be termed ABIOSINIT. These ABIOS initialization programs are accessed through the CBIOS system services functions AH=04 and AH=05. Of course, other means to extend a header and provide an entry point field are possible and matters of well known design choice. Other table entries are set forth in the ABIOS section of the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference Manual*.

Providing an ABIOS which is loadable and separate provides a computer system with a plurality of advantages. A loadable ABIOS increases the amount of available address space in both the system firmware space 508 and the feature space 506. Increasing the amount of available space in the system firmware space 508 and the feature space 506 allows and enhances the operation of expanded memory manager programs. Also, increasing the amount of available space in the system firmware space 508 and feature space 506 allows for larger POST and CBIOS programs to be resident in the respective spaces. Additionally, development, test, distribution and maintenance of ABIOS becomes simplified since ABIOS is a single entity rather than part of a POST CBIOS ABIOS bundle.

While a preferred embodiment of the present invention was described above, it will be understood by those skilled

11

in the art that various changes in detail may be made without departing from the spirit, scope, and teaching of the invention. For example, while the preferred embodiment uses Intel processors and an IBM PS/2 MICRO CHANNEL bus for illustrative purposes, this invention can be implemented on other processors and/or bus types. Likewise, those skilled in the art will recognize that many elements of the invention can be implemented either in hardware or software. Accordingly, the invention should be limited only as specified by the appended claims.

What is claimed is:

1. A personal computer system for making efficient use of a logical memory address space containing plural bounded regions, each region containing multiple address locations; said regions comprising at least a low region having discrete upper and lower boundaries, a high region having a discrete lower boundary, and an intermediate region situated between the upper boundary of said low region and the lower boundary of said high region; said intermediate region traditionally being reserved for storing firmware information required for controlling said computer system, said firmware information including microcode consisting of discretely separate first and second portions, said first microcode portion being required by said system for completing a preliminary initialization enabling said system to handle application programs incapable of addressing said high region; said second microcode portion being useful by said system only after completion of said preliminary initialization and enabling said system to handle applications requiring access to any of said regions; said personal computer system comprising:

a data bus;

a microprocessor electrically coupled to said data bus; said microprocessor operating in first and second modes characterized in that said first mode restricts said microprocessor to address only said low and intermediate regions while said second mode permits said microprocessor to address any of said regions;

non-volatile memory electrically coupled to said data bus, said non-volatile memory being accessible to said microprocessor via said data bus, and storing said first portion of microcode for use by said microprocessor, said non-volatile memory containing physical address locations mapped into said intermediate region of said address space;

volatile memory electrically coupled to the data bus and being accessible to said microprocessor via said data bus, said volatile memory containing physical storage address locations being mappable into said low, high and intermediate regions of said memory address space; said volatile memory address locations mapped into said high region being used to store a linking code for use by said microprocessor in initializing said second portion of microcode, when said second portion is required by said microprocessor for operating said system;

a memory controller electrically coupled to said microprocessor, said volatile memory and said non-volatile memory, said memory controller regulating communications between said volatile memory, said non-volatile memory and said microprocessor; said memory controller operating to translate logical addresses defined by programs used in said system into physical addresses associated with physical storage locations in said volatile and non-volatile memories; and,

a direct access storage device electrically coupled to the data bus, said direct access storage device storing said

12

second portion of operating system microcode; said second portion of microcode being conditionally loaded from said direct access storage device to said volatile memory only when first needed by said system after completion of said preliminary initialization to be processed to an initial state and thereafter accessed by said microprocessor.

2. The computer system of claim 1 wherein

said first portion of microcode includes compatibility operating system microcode used by said microprocessor during said preliminary initialization, and only when said microprocessor is operating in said first mode; and

said second portion of operating system microcode includes advanced operating system microcode used by said microprocessor only after completion of said preliminary initialization.

3. The computer system of claim 1 wherein

said direct access storage device is remote from said microprocessor, and is accessed via a network.

4. An apparatus for loading an operational interface used in the operation of a personal computer system containing a system processor, a non-volatile memory, a volatile memory, and a direct access storage device; said operational interface comprising discretely separate first and second interface portions having different memory addressing capabilities; said first interface portion being required for completing a preliminary initialization of said system, and both said first and second interface portions being useful for enabling said system to perform applications defined by application programs; said non-volatile memory storing said first interface portion of said operational interface and a request indicator for indicating whether said second interface portion of the operational interface is required after completion of said preliminary initialization of said system; said direct access storage device storing said second interface portion of the operational interface; said apparatus comprising:

means formed by said system processor and said first interface portion of said operational interface, during performance of said preliminary initialization of said system, for storing an image of said first interface portion of the operational interface in a predetermined part of said volatile memory;

means formed by said system processor and said first interface portion, at completion of said preliminary initialization, for determining whether said computer system requires the second interface portion of the operational interface for continued operation of said system;

determination means for determining whether said second interface portion of the operational interface is required; means responsive to said determination means for loading said second portion of the operational interface from said direct access storage device into said volatile memory if said second interface portion is required and for permitting continued operation of said system without loading said second portion if said second interface portion is not required.

5. The computer system of claim 4 wherein said system processor is operable in different first and second modes, said modes characterized in that said system processor has a first memory addressing range when operating in said first mode, and a second memory addressing range greater than said first range when operating in said second mode, and wherein

the first interface portion of the operational interface includes compatibility operating system microcode

## 13

required by said system processor when operating in said first mode, and

the second interface portion of the operational interface includes advanced operating system microcode that can be used by said system processor when it is operating in either of said first and second modes.

6. The computer system of claim 4 wherein

said direct access storage device is remote from said system processor and is accessed via a network.

7. A personal computer system compatible with application programs and operating system software, the personal computer system comprising:

a microprocessor electrically coupled to a data bus; said microprocessor operating in different first and second modes; said microprocessor having access to first and second logical memory address spaces when operating respectively in said first and second modes; wherein said second logical address space contains and is larger than said first logical address space; said microprocessor operating initially in said first mode when it is powered up;

non-volatile memory electrically coupled to the data bus, said non-volatile memory storing a first portion of operating system microcode used by said microprocessor only when it is operating in said first mode;

a direct access storage device electrically coupled to the data bus for storing a second portion of operating system microcode; said second microcode enabling said system to handle applications requiring access to any of said regions;

volatile memory electrically coupled to the data bus, said volatile memory storing a linking microcode for use by said microprocessor in effecting an initial transition from said first mode to said second mode;

a memory controller electrically coupled to said microprocessor, said volatile memory and said non-volatile memory, said memory controller regulating communi-

## 14

cations between said volatile memory, said non-volatile memory and said microprocessor; said memory controller translating logical addresses associated with said logical address spaces into physical addresses associated with physical storage locations in said volatile and non-volatile memories; and

linking means for linking said microprocessor to said second portion of operating system microcode during the initial transition of said microprocessor between said first and second modes, said linking means providing a logical address pointer indicating a starting location in said second logical address space; said pointer being used by said memory controller for locating said second portion of the operating system microcode in physical address locations of said volatile and non-volatile memories; said linking means enabling said microprocessor to access said second portion of the operating system microcode when first needed by said microprocessor to effect a transition from said first mode to said second mode.

8. The computer system of claim 7 wherein

said first portion of operating system microcode includes compatibility operating system microcode, used by said microprocessor only when said microprocessor is operating in said first mode, for maintaining compatibility between said microprocessor and application programs written to be executed only by a microprocessor operating in said first mode, and

said second portion of operating system microcode includes advanced operating system microcode used by said microprocessor when operating in said second mode.

9. The computer system of claim 7 wherein

said direct access storage device is remote from said microprocessor and is accessed by said microprocessor via a network.

\* \* \* \* \*