PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 12/04	A1	(11) International Publication Number: WO 90/10907
		(43) International Publication Date: 20 September 1990 (20.09.90)

(21) International Application Number: PCT/US90/01405

(22) International Filing Date: 14 March 1990 (14.03.90)

(30) Priority data:
324,211 15 March 1989 (15.03.89) US

(71) Applicant: BULL MICRAL OF AMERICA [US/US]; 1970 Oakcrest Avenue, Suite 300, Roseville, MN 55113-2624 (US).

(72) Inventors: MANSFIELD, Richard, M. ; 8149 W. 103rd Street, Bloomington, MN 55438 (US). DOHSE, William, F. ; 4610 Trenton Circle, Plymouth, MN 55442 (US).

(74) Agents: MURRAY, Leslie, G. et al.; Schroeder, Davis & Orliss Inc., P.O. Box 3080, Monterey, CA 93942 (US).

(81) Designated States: CA, JP, KR.

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: CONTROLLER FOR DIRECT MEMORY ACCESS

(57) Abstract

A direct memory access controller for computer systems in which the data may be manipulated and acted upon during a transfer to and from locations in memory, or from locations in memory to and from input/output devices. For computer systems having data word widths of two or more bytes of data fewer bus cycles are required for data transfers to and from odd address locations in memory.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	ES	Spain	MG	Madagascar
AU	Australia	FI	Finland	ML	Mali
BB	Barbados	FR	France	MR	Mauritania
BE	Belgium	GA	Gabon	MW	Malawi
BF	Burkina Faso	GB	United Kingdom	NL	Netherlands
BG	Bulgaria	HU	Hungary	NO	Norway
BJ	Benin	IT	Italy	RO	Romania
BR	Brazil	JP	Japan	SD	Sudan
CA	Canada	KP	Democratic People's Republic of Korea	SE	Sweden
CF	Central African Republic			SN	Senegal
CG	Congo	KR	Republic of Korea	SU	Soviet Union
CH	Switzerland	LI	Liechtenstein	TD	Chad
CM	Cameroon	LK	Sri Lanka	TG	Togo
DE	Germany, Federal Republic of	LU	Luxembourg	US	United States of America
DK	Denmark	MC	Monaco		

DESCRIPTION

CONTROLLER FOR DIRECT MEMORY ACCESS

TECHNICAL FIELD

This invention relates to sub-systems in computers for transferring data from one memory location to another, or from or to a memory location to or from an input/output (I/O) device. In particular, the present invention relates to direct memory access (DMA) subsystems having data word widths of 2 or more bytes of data each byte comprising 8 bits, in which the data may be manipulated and acted upon during the transfer.

BACKGROUND ART

In the past, computer systems have been designed to transfer data to the central processing unit (CPU) as quickly as possible so that the CPU could perform all of the operations necessary to achieve the desired results, including large scale iterative scientific computations, on-line business transactions processing, graphics interfaces, accounting programs, and a myriad of other data manipulation intensive operations. DMA became one of the leading methods for providing faster data transfers, particularly in bus-oriented computer system architectures. Even so, DMA subsystems take finite time periods, usually referred to as "bus cycles", or simply "cycles", to complete a data transfer operation. While cycle times have become shorter as the speed of semiconductor device technology becomes faster, the fact remains that no operations on the data could be performed until the transfers were complete.

In a related consideration, early computers were configured to operate on data word widths of 8 bits. As more complex functions were computerized, 8-bit bytes quickly limited the speed of computations and therefore, the throughput of data operations. Thus, wider word widths of 16, 32, 64 bits, and more, arranged into 8-bit wide bytes, were introduced.

Wider word widths are generally encountered in large scale, main

frame systems, but may also be found in process control systems and the like which are driven by intermediate size computers also known as mini-computers. In addition to the general advancement of the level of competence of the individual user, much of the present day commercial and industrial business transactions are being implemented on personal computers (P/C) or P/C-based systems. Since it has become desirable to perform complex functions on P/C-based systems, such systems are being configured to operate on wider word widths than 8 bits to efficiently perform such complex operations. While the vast majority of installed P/C's are limited to 8-bit data word widths, microprocessors becoming commonly available for use in designing the next generation of P/C's have data widths of 16 and 32 bits. Thus, since peripheral devices and memory subsystems to which such microprocessors must interface can have data widths of 8, 16 or 32 bits, DMA subsystems must be compatible with such data widths to anticipate advances in software products and growth in the technology generally.

In the prior art, memory in an 8-bit data word system is directly addressed from the address bus. The memory is organized into 8-bit words, and each address signal decoded from the address bus points to a different and unique word (in this case, also equal to a byte) in the memory. In a system designed for 16-bit data words, the memory is organized into 16-bit words. As long as the DMA subsystem is only required to transfer data from a 16-bit memory or I/O device to or from a 16-bit I/O device or memory, each transfer comprises a word and the data is written or read to or from even numbered addresses. In the prior art, there are systems that provide both 8-bit and 16-bit memory-to-memory and memory-to-I/O device accesses.

If compatibility with 8-bit word I/O devices is desired, some provision for directing flow of data to and from the memory word locations is required. Therefore, typically, 16-bit memory words are further organized into two 8-bit bytes of data. Bits 0 - 7 and 8 - 15 are designated the low

byte and high byte, respectively. Thus, in a 16-bit data word system, each word comprises two 8-bit bytes.

The method for directing the data to and from the individual bytes locations requires additional address signals called byte enables. In a 16-bit word system, the first address line which addresses the first low byte, i.e. the A0 address line, is replaced with two byte enable lines called "BE0N" and "BE1N". When the BE0N signal line is active, the lower byte of the data word is transferred; when the BE1N signal is active, the upper byte of the data word is transferred. When both BE0N and BE1N lines are active at the same time, both bytes, i.e. the complete word, is transferred at the same time.

Similarly, in a 32-bit word system having a 32-bit DMA subsystem, each word is organized into four, 8-bit bytes. In such a system, both address lines A0 and A1 are replaced with four byte enable lines, BE0N and BE1N, BE2N and BE3N, respectively. Likewise, in a 64-bit system with a 64-bit DMA subsystem, each data word is organized into eight, 8-bit bytes per word and address lines A0, A1 and A2, are replaced with eight byte enables BE0N through BE7N. In all cases, the byte enable lines are said to "point" to the bytes of the word that are to be
transferred.

In prior art DMA subsystems, three bus cycles are required for a 16-bit data word transfer to and from an odd address in a P/C system compatible with 8-bit data word bytes. In such systems, the memory is organized into 16-bit words, (i.e. two 8-bit bytes) so that the word boundaries are on even addresses. Referring now to Figure 1A, during cycle 1 for a 16-bit odd address location memory read cycle, the address points to an even memory location and an 8-bit read of the upper byte is performed in response to the byte one enable. The data read (byte A) is stored in a register associated with bits 7-0 comprising byte A. After the first memory read cycle, the memory address is incremented by two to the next even address, and the byte pointer is activated to point to the lower byte of data to be read (byte B). The lower byte of data is read and steered into another

register during the second cycle. During the third cycle, the entire word, bits 15-8 and 7-0, i.e. bytes B and A, are assembled serially and driven onto the bus for transmission to the I/O device. Thus, it requires three cycles to transfer each word using this technique. If this technique is extrapolated for transferring words comprising more than two bytes, the number of cycles required would be (a+1)N, where a is the number of bytes per word and N is the number of words to be transferred. Thus, for a 4-byte word, 5 cycles/word would be required.

Referring to Figure 1B, a 16-bit odd memory write operation is similar to the 16-bit odd memory read operation. Again, three bus cycles per transfer are required. In the first cycle, one word, i.e. two 8-bit bytes B and A, is read from the I/O device and stored in a latch. During cycle two, lower byte A in the latch is written to the upper byte memory location in response to byte enable BE1N. The memory address is incremented, BE1N is driven inactive and the upper byte in the latch is written to the lower byte memory location in response to byte enable signal BE0N.

It should be noted that, in the prior art, the memory address may be decremented in all 8-bit accesses and 16-bit accesses from an even address. However, for a 16-bit access from an odd address, the memory address may be incremented only.

As already noted, most computer systems, whether large scale main frame computers or modern day P/C's, transfer data to registers in or near the CPU or microprocessor, respectively, before manipulations including simple arithmetic operations, exclusive-OR and barrel shifting, are performed on it. However, the advent of very large scale integration (VLSI) semiconductor technology has provided the opportunity to implement previously impractical computing system architectures. See for example "VLSI: The Challenge to Innovate", VLSI systems Design, November, 1988 at p. 6. Thus, it is now practical to design systems which do many things faster or many more things in the same time, or both, than was previously possible. In particular, it is now possible to implement a practical DMA

subsystem which transfers data faster, and which can manipulate data during the transfer, i.e. on-the-fly.

DISCLOSURE OF THE INVENTION

The DMA subsystem of the present invention includes a data manipulator which receives data during the read cycle of any DMA transfer. The data manipulator includes circuitry for rearranging the byte order of the data during transfer or optionally, transferring the bytes of data in the order received. In addition, the data manipulator includes circuitry which provides the option of rearranging bit locations of the data within the bytes of data, or of performing logical or arithmetic operations on the data, during the transfer. Finally, the data manipulator includes circuitry for driving the data, altered or unaltered, back onto the data bus during the write cycle of the DMA transfer.

The data manipulator of the present invention provides several options for transferring the data while at the same time assuring compatibility with I/O devices of different data word sizes and providing with the opportunity for overall system performance improvements. Any byte of data presented to the input of the data manipulator can be manipulated while being transferred, regardless of the word-size of the data at either the origin or destination of the transfer.

Performing a barrel-shift operation on the data allows graphics-type data manipulations to take place during DMA transfers. Arithmetic and logical operations may also be performed on the data during DMA transfers to speed up operations such as stripping columns out of files. If used in conjunction with a memory-to-memory DMA transfer, file manipulation may take place independently of the system processor, again, during the transfer.

The DMA subsystem of the present invention is designed in one byte modular increments. The system may be expanded or reduced to any data desired word width.

In addition to providing the capability of manipulating data during a

DMA transfer, the DMA subsystem of the present invention reduces the number of bus cycles required for a 16-bit data word transfer to or from an odd address. During an initial memory read cycle, the first 8 bits of data (i.e. low byte) is stored in a latch. During the second memory read cycle that first low byte is transferred into a register at the same time the next 16 bits (i.e. high byte of the first word and low byte of the second word) is loaded into the latch. During the write cycle, the second byte is steered from the latch to the upper byte location of the data bus and the first low byte is steered from the register to the lower byte location of the data bus. During subsequent cycles, the upper byte in the latch is loaded into the last-mentioned register when the latch is loaded with the next 16 bits of data. Thus, by adding a single register to the DMA subsystem and appropriately directing the flow of data, about 30% less bus cycle time is required to make DMA transfers of 16-bit data words from an odd address in the DMA subsystem of the present invention as compared with prior art DMA subsystems.

Other features and advantages of the present invention, in addition to those mentioned above, are described in the following detailed description of the preferred embodiment. The detailed description incorporates the accompanying drawings wherein similarly referenced characters refer to corresponding parts of the invention.

BRIEF DESCRIPTION OF THE DRAWING

Figures 1A and 1B illustrate a 16-bit data word transfer to and from an odd address in a prior art computer system compatible with 8-bit data word width.

Figure 2 is a block diagram of a data manipulator of a DMA controller constructed according to the principles of the present invention.

Figure 3 is a block diagram of an input latch employed in the DMA subsystem of Figure 2.

Figure 4 is a block diagram of a byte sorter employed in the DMA

subsystem of Figure 2.

Figure 5 is an ALU/barrel shifter employed in the DMA subsystem of Figure 2.

Figure 6A illustrates an 8-bit memory read transfer as performed by the DMA subsystem of Figure 2.

Figures 6B and 6C illustrate alternative 8-bit memory write transfers as performed by the DMA subsystem of Figure 2.

Figures 7A and 7B illustrate 16-bit transfers from even address locations as performed by the DMA subsystem of Figure 2.

Figures 8A and 8B illustrate a 16-bit data word transfer to and from an odd address in a computer system compatible with 8-bit data word width which incorporates the DMA subsystem of Figure 2.

Figures 9A - 9F illustrate conventions for 16-bit data word decrement transfers from an odd address in a computer system compatible with 8-bit data word width which incorporates the DMA subsystem of Figure 2.

BEST MODE FOR CARRYING OUT THE INVENTION

A DMA controller constructed according to the present invention includes data manipulator 10 as shown in Figure 2 comprising latch 20, byte sorter 22 and data pre-processor 24, all of which being operationally controlled by state machine 26. With continuing reference to Figure 2, the input of latch 20 is coupled to bi-directional data bus 15 for receiving data during a transfer operation. The output of latch 20 is coupled to the input of byte sorter 22. In similar fashion, the output of byte sorter 22 is coupled to the input of data pre-processor 24 for processing data received therefrom and for driving the data back onto bi-directional data bus 15.

The size of latch 20 is determined by the maximum data width (in bits per word) of the computer system. As shown in Figure 3 for a 64-bit computer system, latch 20 comprises standard transparent latches, such as type LACFNB manufactured by VLSI Technology, Inc., where one latch is used for each bit of data per word processed.

Referring now to Figure 4, byte sorter 22 comprises multiplexers 401 - 408 coupled to multiplexers 409 to 416, respectively, and to 8-bit registers 417 - 424, respectively. Each multiplexer (MUX) 401 - 408 convert eight 8-bit bytes into one 8-bit byte; similarly, each MUX 409 - 416 convert two 8-bit bytes to one 8-bit byte, as shown. MUXes 401 - 408 and 409 - 416 are commonly constructed from components such as type MX21D1, manufactured by VLSI Technology, Inc. Registers 417 - 424 are commonly constructed from components such as type DFCTNB Flip flop, also manufactured by VLSI Technology, Inc.

For simplicity in illustrating the principles of the present invention, a 16-bit computer system, i.e. two 8-bit bytes, utilizes only the bottom two sections of the logic of byte sorter 22 for performing transfer operations.

Referring now to Figure 6A with continuing reference to Figure 4, in cycle 1 of an 8-bit memory read transfer, byte A is read from an 8-bit memory location designated 601, having an even word address value, i.e., BE0N active, BE1N inactive, and written into byte location designated 301 of latch 20. After the data has been read, the memory address counter (not shown) is incremented by one so that it points to the next even word memory address but with BE0N inactive and BE1N active in preparation for executing the next memory read operation. During cycle 2, byte A is directed to both byte locations designated 602 and 603 of the 16-bit data bus in response to control signals A0, A1 and A2 at MUX 401 and control signals B0, B1 and B2 at MUX 402 and control signals SEL A0 and SEL B0 at MUXes 409 and 410, respectively, from state machine 26. Thus, data bits 7-0 coming into byte sorter 22 on input AI are steered through MUXes 401, 402, 409 and 410, respectively, for presentation at both outputs AO (bits 7-0) and BO (bits 15-8). Since, in the case of 8-bit memory transfers, there is no need to store data in registers 417 and 418, respectively, the data is passed directly to data bus 15.

DMA subsystem 10 is controlled by state machine 26 so that addresses are driven onto the address bus while also generating byte

pointers to steer the data to the proper destination location. Thus, for example, during the second cycle of the 8-bit transfer, a byte zero enable signal, namely BE0N, is also generated as part of the address. BE0N indicates to the I/O device that it should write only the data that is designated byte zero, i.e. the lower 8 bits of the data bus, since the data is being directed to both the lower and upper byte of the data bus simultaneously.

During cycle 3, another memory read operation is executed. The data is read from the high byte memory location designated 604, having an even word address with BE0N inactive and BE1N active, and written into low byte location designated 302 of latch 20. During the fourth cycle this high byte data is directed to both byte locations 605 and 606 of the 16-bit data bus, and the address and byte pointers direct the data to the appropriate I/O device. Data continues to be transferred in the same manner during subsequent cycles as shown until the transfer operation is complete.

Referring to Figure 6B with continuing reference to Figure 4, in cycle 1 of an 8-bit memory write transfer, one byte is read from an 8-bit I/O location designated 610 having an even address value and written into byte location designated 301 of latch 20. During cycle 2, byte A is directed to both byte locations designated 611 and 612 of the 16-bit data bus in response to control signals A0, A1 and A2 at MUX 401 and control signals B0, B1 and B2 at MUX 402 and control signals SEL AO at MUX 409 and SEL BO at MUX 410 from state machine 26. Thus, data bits 7-0 coming into byte sorter 22 on input AI are steered through MUXes 401, 402, 409 and 410, respectively, for presentation at both outputs AO (bits 7-0) and BO (bits 15-8). As before for 8-bit memory transfers, there is no need to store data in registers 417 and 418, respectively, and the data is passed directly to data bus 15.

DMA subsystem 10 is controlled by state machine 26 so that addresses are driven onto the address bus while also generating byte

pointers to steer the data to the proper destination location. Thus, during the second cycle of the 8-bit transfer, a byte zero enable signal, namely BE0N, is also generated as part of the address. BE0N indicates to the memory that it should write only the data that is on designated byte zero, i.e. the lower 8 bits of the data bus, since the data is being directed to both the lower and upper byte of the data bus simultaneously.

During cycle 3, another I/O read operation is executed. The data is read from the low byte I/O location designated 613, having an even word address with BE0N active and BE1N inactive, and written into low byte location designated 301 of latch 20. During the fourth cycle this low byte data is directed to both byte locations 614 and 615 of the 16-bit data bus, and the address and byte enables BE0N and BE1N direct the data to memory location designated 614. Data continues to be transferred in the same manner during subsequent cycles as shown until the transfer operation is complete.

As shown in Figure 6C with continuing reference to Figure 4, in cycle 1 of an 8-bit memory write transfer, one byte is read from an 8-bit I/O location designated 620, having an even word address value with BE0N inactive and BE1N active, and written into byte location designated 302 of latch 20. During cycle 2, byte A is directed to both byte locations designated 621 and 622 of the 16-bit data bus in response to control signals A0, A1 and A2 at MUX 401 and control signals B0, B1 and B2 at MUX 402 and control signal SEL AO at MUX 409 and SEL BO at MUX 410 from state machine 26. Thus, data bits 7-0 coming into byte sorter 22 on input BI are steered through MUXes 401, 402, 409 and 410, respectively, for presentation at both outputs AO (bits 7-0) and BO (bits 15-8). Again, since in the case of 8-bit memory transfers, there is no need to store data in registers 417 and 418, respectively, the data is passed directly to data bus 15.

DMA subsystem 10 is controlled by state machine 26 so that addresses are driven onto the address bus while also generating byte

pointers to steer the data to the proper destination location. Thus, during the second cycle of the 8-bit transfer, a byte one enable signal, namely BE0N, is also generated as part of the address. BE0N indicates to the memory that it should write only the data that is designated byte 622, i.e. the lower 8 bytes of the data bus, since the data is being directed to both the lower and upper byte of the data bus simultaneously.

During cycle 3, another I/O read operation is executed. The data is read from byte I/O location designated 623, having even word address and BE0N inactive and BE1N active, and written into byte location designated 302 latch 20. During the fourth cycle this high byte data (hex value B) is directed to both byte locations 624 and 625 of the 16-bit data bus, and the address and byte pointers, i.e. BE0N inactive, BE1N active, direct the data to the appropriate memory location designated 624. Data continues to be transferred in the same manner during subsequent cycles as shown until the transfer operation is complete.

A 16-bit memory read operation, as shown in Figure 7A, from an even address, i.e. both byte enables BE0N and BE1N active, is straightforward. During cycle 1 of such an operation, one word (two 8-bit bytes) is read from an even memory location designated 701 and 702, and written into the lower two byte locations 301 and 302 of latch 20. The memory address counter (not shown) is incremented by two in preparation for the next memory read operation. State machine 26 controls MUXes 401 and 402 so that the bytes are passed straight through byte sorter 22 and directed onto 16-bit data bus 15 for transmission to byte locations 703 and 704 at the appropriate I/O device during the second cycle of the DMA transfer operation. Data transfers continue in the same way during subsequent cycles as shown until the transfer operation is complete. Of course, the data may be subjected to operations performed by data pre-processor during the transfer, if desired, as described elsewhere in this specification.

Referring now to Figure 7B, a 16-bit memory write operation to an

even address is similar to the corresponding 16-bit memory read operation. During cycle 1 of such a transfer, one word (two 8-bit bytes) is read from the I/O device and written into the lower two byte locations designated 301 and 302 of latch 20. During cycle 2, the data is passed directly through byte sorter 22, then directed onto data bus 15 for transmission to appropriate memory locations designated 712 and 713. Thereafter, the memory address counter (again, not shown) is incremented to the next address in preparation for executing the next transfer operation.

With reference now to Figure 8A, the first byte of a 16-bit odd memory read operation is read in the same way as described for the corresponding prior art operation. Thus, at the end of the first cycle, the first byte is stored in byte location designated 302 in latch 20, corresponding to byte location designated 801. The memory address pointer is incremented to the next even word address. However, during the second cycle, byte A, stored in location designated 302 of latch 20, is transferred through MUX 401 into register 417. Since the memory address is now at an even boundary, a 16-bit memory read operation is executed whereby both bytes B and C are written to locations designated 301 and 302 in latch 20 corresponding to byte locations 802 and 803, respectively. During cycle 3, byte B is directed to byte location designated 805 via MUXes 402 and 410. At the same time, byte A, which is stored in register 417, is directed to byte location 804 via MUX 409. The address register is incremented to point to the next even memory address in preparation for the next memory read operation. At the beginning of cycle 4, data byte C is transferred through MUX 401 into register 417. The next memory read cycle operation is also 16-bits wide, comprising byte D of word two and byte E of word three. The two data bytes are cross-transferred and stored during the write cycle of the transfer, as before.

In the present invention, only the first memory read cycle is 8 bits (1 byte) wide. After that initial cycle, read operations are always 16 bits

wide, and the bytes are cross-transferred and stored to allow the I/O device to write them in the correct order. In this way, only one extra cycle is required at the beginning of the transfer, and every 16-bit word thereafter is transferred using only two cycles to complete the transfer operation.

In the 16-bit odd address memory write operation depicted in Figure 8B, three cycles are required for the last transfer of a burst of data, but only two cycles are required for all other transfer operations. In cycle 1, one word (two 8-bit bytes A and B) is read from the I/O device and stored at locations designated 301 and 302 in latch 20 from byte locations designated 810 and 811. During cycle 2, byte A in latch 20 location designated 301 is directed to the odd byte location corresponding to BO (bits 15-8) at location designated 813 via MUXes 402 and 410. During cycle 3, the memory address counter is incremented so that it points to an even or complete word boundary. In addition, during this cycle, byte B in latch byte location 302 is stored in register 417. Since the memory address is now at an even boundary, subsequent transfers will be complete 16-bit word transfers with the individual bytes sorted and steered so that they will be written in the right order in their respective memory's locations. Also during cycle 3, the next word comprising byte C and byte D of word two is read from the I/O device and stored in latch 20 at locations designated 301 and 302, respectively. During cycle 4, byte C in latch 20 at location designated 301 is directed to byte location 817, corresponding to BO (bits 15-8). At the same time, the contents of register 417 are directed to byte location 816 corresponding to the memory location AO (bits 7-0). This sequence continues until the last full word has been written. During the memory write operation for the last full word, the memory address counter is again incremented to point to the next even address and one last cycle is executed with enable signal BE0N active and enable signal BE1N inactive. Therefore, during the last cycle, only the last byte of data, which is contained in register 417 is directed

into the data bus and the data transfer is complete.

In a P/C system compatible with 8-bit data words according to the present invention, every transfer of multi-byte words to or from a memory location requires two bus cycles to complete - one cycle for the read operation and one for the write operation. If the transfer is to or from an odd memory address location, only one additional cycle per transfer is required, irrespective of the number of words or the number of bytes comprising the words of data to be transferred. Thus, such transfers require 2N+1 cycles, where N is the number of data words to be transferred. Therefore, for a 100 word transfer using the technique of the present invention, 201 cycles are required, whereas in prior art DMA subsystems, 3N cycles, or 300 cycles, are required for the same data transfer operation.

Furthermore, the performance advantage of the present invention becomes greater as the amount of data to be transferred or the size of the memory increases. By transferring whole, multiple-byte words as soon as an even memory address boundary is reached (at the beginning of the second cycle), the speed of such transfers is substantially increased by reducing the number of cycles required, while maintaining compatibility with devices designed for computers with different data word widths and providing capability to manipulate the data on-the-fly.

As noted elsewhere in this specification, the byte address pointer may be decremented to next byte to be transferred, even from an odd address, for a 16-bit word transfer. Since no convention exists in the prior art for determining which byte segments of a 16-bit word are to be transferred from a given address to which the byte pointer has been decremented, the present invention anticipates several different conventions.

Figure 9A shows the physical layout of a memory system for purposes of reference to the alternative conventions of Figures 9B - 9G. Thus, for example, 8-bit data byte 5 is stored at physical location, address

5. In Figure 9B, transfer from even memory address 6 in decrement mode is illustrated according to the generally accepted industry convention.

Figures 9C - 9G illustrate several conventions for transfer of data from odd memory address 7 in decrement mode according to which the controller of the present invention can transfer data. Thus, for example, the first 16-bit word transferred in Figure 9D comprises bytes 6 and 7 in high byte-to-low byte order, whereas the first word transferred in Figure 9E comprises the same bytes but in reverse order.

In all conventions proposed, data transfers are implemented in analogous manner to that described for 16-bit word transfers from odd addresses in which the byte address pointer is incremented. Thus, only the desired convention is required to determine the order of control of byte enable signals, BE0N, BE1N...BEnN for implementing data transfers in decrement mode.

Data manipulation includes the sorting of data word bytes during a transfer as described in connection with byte sorter 22 as shown in Figure 4. However, by forming a data path for DMA transfer operations which includes data pre-processor 24, manipulation of data can be expanded to include operations such as barrel shifting, exclusive OR, and other arithmetic and logical operations which are otherwise typically performed by the system processor. Data pre-processor 24 includes register 51, MUX 52, ALU control logic 53 and ALU/barrel shifter module 54. ALU/barrel shifter module 54, as shown in Figure 5, is similar to the continuation of types VDP3ALU001 and VDP3BSH001, produced by VLSI Technology, Inc., and may be designed in accordance with design rules given in "VDP 10 Datapath Element Library", published by VLSI Technology, Inc.

The design of the data manipulator of the present invention is modular, in single 8-bit byte increments. The word width can be expanded or contracted to any desired data word width. In addition, the duration of time for data manipulation between read and write operations of a DMA transfer also may be expanded to provide for any number and

types of manipulations of the data being transferred consistent with the computer system operating parameters.

A detailed specification of a DMA controller constructed according to the preferred embodiment of the present invention is given in Exhibit A hereto, which is incorporated by reference herein. While preferred forms and arrangements have been shown illustrating the invention, it is to be understood that various changes in detail and arrangement may be made without departing from the spirit and scope of this disclosure or the claimed invention.

EXHIBIT A

OVERVIEW

The GC183 provides eight channels of Direct Memory Access (DMA). The system microprocessor programs the DMA registers for the various modes of operation, transfer addresses and transfer counts. The DMA Controller may be programmed in one of two modes. The first programming mode emulates the operation of two Intel 8237 DMA Controllers. The other programming mode uses extended addressing. In the DMA transfer mode, I/O devices may transfer data directly to and from memory in single transfer, burst or read verification mode. The data transfer is initiated when an I/O device requests and is granted control of the bus by the bus arbitration logic and the DMA has been programmed to service the DMA request.

Several additional functions were included in the device to reduce the need for separate circuits in the system. These include:

- DRAM refresh logic
- Numeric coprocessor interface
- Central Arbitration Point
- Floppy Disk Arbitration

SYSTEM CONFIGURATION

The DMA Controller is connected to the private bus to receive instructions from the CPU and to provide the bus control signals during DMA transfers. It is also connected to the Micro Channel to control the bus arbitration process. When the CPU programs the DMA Controller, the CPU treats it like an 8 bit I/O device on the Micro Channel. The DMA Controller uses addresses in the range 0000H - 00DFH. In the 8237 mode, individual registers are represented by separate locations in the I/O address map. In the extended mode, one address is used to store a code which indicates the

register being addressed and the function to be performed. A second address is used for the data to be read or written.

OPERATING MODES

The DMA controller operates in the two basic states of Idle and Command. In the Idle mode, the DMA controller is not actively controlling the bus and may be programmed by the system board processor. Conversely, in the command mode the DMA Controller drives Micro Channel bus control signal in order to generate DMA transfers.

REGISTERS

The following registers are contained with the DMA:

Register	Length	Registers
Memory	24	8
Memory Address (Current)	24	8
I/O Address	16	8
Transfer Count (Base)	16	8
Transfer Count (Current)	6	8
Mode	8	8
Status	8	2
Arbus	4	2
Mask	4	2
Temp Holding	16	1
Function	8	1
Arbitration	8	1
Refresh	10	1

CHANNEL REGISTERS

Each of the eight channels uses a set of four registers: Memory Address, I/O Address, Transfer Count, and Mode.

Memory Address Register: Each channel uses a set of Memory Address registers. The base address register is programmed by the CPU with the starting address for the DMA transfer. The base address is copied into the current address register. The contents of the base register do not change during DMA transfers. The current address register points to the memory

address to be used in the next transfer and is updated (incremented or decremented) after each DMA transfer cycle. The CPU can read the base and current registers. When the autoinitialization option is selected, the base address is copied into the current address register upon completion of the transfer.

I/O Address Register: This register identifies the address for the I/O device being used for the DMA transfer. The contents of this register do not change during DMA transfers.

Transfer Count Register: Each channels uses a set of transfer count registers. The Base Count register is programmed by the CPU with the count of transfers to be performed for the DMA transfer. The count is copied to the current count register. The contents of the Base register do not change during DMA transfers. The current count register is decremented with each transfer that is completed. When this register is decremented from 0000 to FFFF, the terminal count signal is activated. Because of this method of indicating the terminal count, this register should be loaded at setup with the value one less than the number of transfers required. When the autoinitialization option is selected, the base count is copied into the current count register upon completion of the transfer.

Mode Register: The Mode register configures its associated channel for the type of operation to perform during DMA transfers. The functions programmed are the same but the methods for programming this register are different for the 8237 mode and the extended mode. This register may be read only in the extended mode. The bit assignments for the two operating modes are as follows:

Mode Register (8237 Compatible)

Bit	Function
7,6	Reserved = 0
5	0 = Increment Memory Address
	1 = Decrement Memory Address
4	0 = Autoinitialization Disabled
	1 = Autoinitialization Disabled
3,2	00 = Verify Transfer

		01 = Write Transfer
		10 = Read Transfer
		11 = Reserved
	1,0	00 = Select Channel 0 or 4
5		01 = Select Channel 1 or 5
		10 = Select Channel 2 or 6
		11 = Select Channel 3 or 7

Mode Register (Extended Mode)

10

Bit	Function
7	Reserved = 0
6	0 = 8 Bit Transfer
	1 = 16 Bit Transfer
15 5	Reserved = 0
4	0 = Memory Address Increment
	1 = Memory Address Increment
3	0 = Memory Read Transfer
	1 = Memory Write Transfer
20 2	0 = Verify Transfer
	1 = Data Transfer
1	0 = Autoinitialization Disabled
	1 = Autoinitialization Enabled
0	0 = I/O Address Forced to 0000H
25	1 = I/O Address Taken from I/O Address Register

COMMON REGISTERS

Several registers are shared among the channels or are used for other 30 functions. These are the Status Register, Arbus Register, Mask Registers, Byte Pointer, Temporary Holding Register, and Refresh Register.

Status Register: Two 8-bit Status registers are provided. One for channels 0 thru 3 and the other for channels 4 thru 7. The information in these registers tells which channels have reached a terminal count and which 35 channels have requested the bus since the last time the status register was read (bits are cleared after each time the register is read). This register can be read using either 8237 mode or extended mode.

In the extended mode, the entire byte containing the information for the selected channel is returned. The information is retrieved in conjunction with the Byte Pointer. The channel number in the Extended Function Register (address 0018H) is ignored. The value returned corresponds to the group selected by the Byte Pointer. When the Byte Pointer is cleared, it selects the status register for channels 0-3. Reading the status register in the extended mode toggles the Byte Pointer.

Status Register Bit Assignments

Bit	Function
7	Channel 3 or 7 Request
6	Channel 2 or 6 Request
5	Channel 1 or 5 Request
4	Channel 0 or 4 Request
3	Terminal count for Channel 3 or 7
2	Terminal count for Channel 2 or 6
1	Terminal count for Channel 1 or 5
0	Terminal count for Channel 0 or 4

Arbus Register: The DMA Controller can work with any of the arbitration levels from 0 to 14, with eight possible channels. Channels 1-3 and 5-7 are assigned to the corresponding arbitration levels. Two 4-bit registers are provided, one register is for channel 0 and the other for channel 4. These registers provide virtual DMA operation by allowing the system microprocessor to assign the arbitration level and allow channels 0 and 4 to service devices at any arbitration level. When channel 0 or channel 4 is assigned to one of the pre-assigned levels (1-3 or 5-7), the mask bits are used to select which of the channels (or none) is to be used.

Arbus Register Bit Assignments

Bit	Function
7-4	Reserved
3-0	Arbitration level

Mask Registers: If a device requests DMA service by winning the arbitration for a channel which has the mask bit set, the DMA will not execute any DMA transfer cycles. This method may be used for a bus master to gain access to the Micro Channel, or to select among several DMA channels assigned to the same arbitration level. If the arbitrating device does not execute the transfer itself, a timeout will occur which will generate a nonmaskable interrupt.

In the 8237 mode, two 4-bit Mask Registers are provided, one register for channels 7-4 and one for channels 3-0. In the extended mode, the individual bits are set and cleared by selecting the desired channel and using the appropriate command.

Individual mask bits may be set or cleared in either 8237 mode or extended mode. All mask bits are set by a system reset or by a DMA Controller master clear. All mask bits in the four-register group are cleared by a Clear Mask Register command in the 8237 mode.

Mask Register Bit Assignments

Single Mask Register (0000A, 00D4)

Bit	Function
7-3	Reserved=0
2	0 = Clear Mask Reg 1 = Set Mask Reg
1,0	00 = Select Channel 0 or 4 01 - Select Channel 1 or 5 10 = Select Channel 2 or 6 11 = Select Channel 3 or 7

All Mask Register (000F, 00DE)

Bit	Function
7-4	Reserve=0
3	Channel 3 or 7 0 = Clear Mask 1 = Set Mask Reg
2	Channel 2 or 6 0 = Clear Mask Reg 1 = Set Mask Reg
1	Clear Channel 1 or 5 0 = Clear Mask Reg 1 = Set Mask Reg
0	Channel 0 or 4 0 = Clear Mask Reg 1 = Set Mask Reg

Arbitration Register: This register is used to program several options for the arbitration function and to monitor the status of key parameters of the

arbitration cycle. This register is located at 0090H on the I/O address map. The bit representations are different for writing and reading the register:

Arbitration Register Bit Assignments

Bit	Rd/WR	Function	Initial Value
7	R	1 = Enable CPU cycles during arbitration cycles 0 = Disable CPU cycles during arbitration cycles	
	W	1 = Enable CPU cycles during arbitration cycles 0 = Disable CPU cycles during arbitration cycles	
6	R	1 = An NMI has occurred and has masked arbitration 0 = Normal operation	
	W	1 = Force arbitration when CPU controls the channel 0 = Normal arbitration	
5	R	1 = A bus timeout has occurred 0 = Normal operation	
	W	1 = Arbitration time is 600 ns. 0 = Arbitration time is 300 ns.	
4	R/W	1 = Enable Preempt for interrupt request 0 = Disable Preempt for interrupt request	
3-0	R	Value of Arbitration Bus during the previous Grant state	
	W	Reserved = 0	

Refresh Register: One 10-bit Refresh register is provided. This register provides memory address for the refresh operation. The system microprocessor does not have access to the refresh register.

PROGRAMMING

This section describes how to configure the DMA controller to perform transfers.

8237 MODE

The DMA Controller contains logic to model two Intel 8237 DMA Controllers. When this mode is used for programming, the registers are accessed by selecting individual addresses and reading from- or writing to them. The DMA Controller provides full 16-bit address decoding for the I/O bus. The functions and their addresses are as follows:

DMA Controller Registers

Address Pointer		Function	Byte
0000	W	Base and Current Memory Address Write Channel 0	Yes
	R	Current Memory Address Read Channel 0	Yes
0001	W	Base and Current Transfer Count Write Channel 0	Yes
	R	Current Transfer Count Read Channel 0	Yes
0002	W	Base and Current Memory Address Write Channel 1	Yes
	R	Current Memory Address Read Channel 1	Yes
0003	W	Base and Current Transfer Count Write Channel 1	Yes
	R	Current Transfer Count Read Channel 1	Yes
0004	W	Base and Current Memory Address Write Channel 2	Yes
	R	Current Memory Address Read Channel 2	Yes
0005	W	Base and Current Transfer Count Write Channel 2	Yes
	R	Current Transfer Count Read Channel 2	Yes
0006	W	Base and Current Memory Address Write Channel 3	Yes
	R	Current Memory Address Read Channel 3	Yes
0007	W	Base and Current Transfer Count Write Channel 3	Yes
	R	Current Memory Address Read Channel 3	Yes
0008	W	Reserved	
	R	Channel 0-3 Status Register Read	
0009	R/W	Reserved	
000A	W	Channel 0-3 Single Mask Register Write	
	R	Reserved	
000B	W	Channel 0-3 Mode Register Write	
	R	Reserved	
000C	W	Clear Byte Pointer	
	R	Reserved	
000D	W	MasterClear	
	R	Reserved	
000E	W	Channel 0-3 Mask Register Clear	
	R	Reserved	
000F	W	Channel 0-3 All Mask Register Bits Write	
	R	Reserved	
0018		See Extended Mode Operation	
01A		See Extended Mode Operation	
0080	W/R	Not Implemented	
0081	W/R	Channel 2, Page Table Address Register	
0082	W/R	Channel 3, Page Table Address Register	
0083	W/R	Channel 1, Page Table Address Register	
0084-0088		Not Implemented	
0087	W/R	Channel 0, Page Table Address Register	
0088	W/R	Not Implemented	
0089	W/R	Channel 6, Page Table Address Register	
008A	W/R	Channel 7, Page Table Address Register	

	008B	W/R	Channel 5, Page Table Address Register	
	008C-008E		Not Implemented	
	008F	W/R	Channel 4, Page Table Address Register	
	0090	W/R	Arbitration Register Write/Read	
	00C0	W	Base and Current Memory Address Write Channel 4	Yes
		R	Current Transfer Count Read Channel 4	Yes
	00C1	W/R	Reserved	
	00C2	W	Base and Current Transfer Count Write Channel 4	Yes
		R	Current Transfer Count Read Channel 4	Yes
	00C3	W/R	Reserved	
	00C4	W	Base and Current Memory Address Write Channel 5	Yes
		R	Current Transfer Count Read Channel 5	Yes
	00C5	W/R	Reserved	
	00C6	W	Base and Current Transfer Count Write Channel 5	Yes
		R	Current Transfer Count Read Channel 5	Yes
	00C7	W/R	Reserved	
	00C8	W	Base and Current Memory Address Write Channel 6	Yes
		R	Current Transfer Count Read Channel 6	Yes
	00C9	W/R	Reserved	
	00CA	W	Base and Current Transfer Count Write Channel 6	Yes
		R	Current Transfer Count Read Channel 6	Yes
	00CB	W/R	Reserved	
	00CC	W	Base and Current Memory Address Write Channel 7	Yes
		R	Current Transfer Count Read Channel 7	Yes
	00CD	W/R	Reserved	
	00CE	W	Base and Current Transfer Count Write Channel 7	Yes
		R	Current Transfer Count Read Channel 7	Yes
	00CF	W/R	Reserved	
	00D0	W	Reserved	
		R	Channel 4-7 Status Register Read	
	00D1	W/R	Reserved	
	00D2	W/R	Reserved	
	00D3	W/R	Reserved	
	00D4	W	Channel 4-7 Single Mask Register Write	
		R	Reserved	
	00D5	W/R	Reserved	
	00D6	W	Channel 4-7 Mode Register Write	
		R	Reserved	
	00D7	W/R	Reserved	
	00D8	W	Clear Byte Pointer	
		R	Reserved	
	00D9	W/R	Reserved	
	00DA	W	Master Clear	
		R	Reserved	
	00DB	W/R	Reserved	
	00DC	W	Channel 4-7 Mask Register Clear	

00DD	W/R	Reserved
00DE	W	Channel 4-7 All Mask Register Bits Clear
	R	Reserved
00DF	W/R	Reserved

Byte Pointer: The memory addressing range requires 24 bits of address data. The Transfer Counter and I/O addresses use 16 bits. To move data on an 8-bit bus, the DMA Controller uses a Byte Pointer. Each access of a part of the system using the Byte Pointer causes it to toggle to the next state. A write to 000CH or 00D8H in 8237 mode or a master reset command in extended mode will reset the Byte Pointer. In addition, any write to the extended mode Function register (0018H) will reset the Byte Pointer. When reset, the Byte Pointer will select the least significant byte of the word accessed. The next access will be to the next more significant byte. Once the most significant byte is accessed, the Byte Pointer will again toggle to the least significant byte. The Byte Pointer is also used with the Status register to select between the status bits for channels 0-3 and for channels 4-7.

Temporary Holding Register: One 16-bit Temporary Holding register is provided. During a DMA transfer, data that have been read are held in this register while waiting to be written to their destination. The system microprocessor does not have access to this register.

Function Register: One 8-bit Function Register is provided. This receives the extended mode I/O commands and is the register which identifies the operation to be performed. The system microprocessor executes functions by first writing to the function register (address 0018) the function is to be performed and the channel to use. The selected function is then executed by writing or reading port address 001A.

I/O Address	Command
0018	Function Register
001A	Execute Function Register

The Function Register bits are assigned as follows:

Bit	Function
7-4	Command
3	Reserved = 0
2-0	Channel

Extended Commands Registers

Command	Register	Bits	Byte Pointer
0 W/R	I/O Address Register	15-00	Yes
1 W/R	Reserved		
2 W	Base and Current Memory Address Write	23-00	Yes
2 R	Base Memory Address Read	23-00	Yes
3 W	Reserved		
3 R	Current Address Memory Read	23-00	Yes
4 W	Base and Current Transfer Count Write	15-00	Yes
4 R	Base Transfer Count Read	15-00	Yes
5 W	Reserved		
5 R	Current Transfer Count Read	15-00	Yes
6 W	Reserved		
6 R	Status Register Read	7-00	Yes
7 W/R	Extended Mode Register Write/Read		
8 W/R	Arbus Register Write/Read	4-00	
9	Mask Register Set Single Bit (Direction from Function Register)		
A	Mask Register Clear Single Bit (Direct from Function Register)		
B W/R	Reserved		
C W/R	Reserved		
D	Master Clear (Direct from Function Register)		
E W/R	Reserved		
F W/R	Reserved		

EXTENDED MODE

In the extended mode, two addresses are used to access all registers and bits. The first address (0018), known as the Extended Function Register, serves as a pointer to the desired channel register or bit and identifies the function to be performed. The second address (001A) is used to pass data, if required. Refer to the Function Register description above for details on the function codes.

AUTOINITIALIZATION

The DMA Controller autoinitialization feature allows the programmer to reduce the overhead associated with the setup for DMA transfers by automatically initializing the current Memory Address register and the Transfer Count register with the values in their corresponding base registers after completion of a DMA transfer. To do this, the autoinitialization bit is set in the Mode register.

ARBITRATION

Arbitration of the contention for access to the Micro Channel bus consists of allowing all master or slave devices requesting access to issue their requests in parallel and then selecting the device with the highest priority to perform its transfer. The sequence is:

1. A device requests access by pulling the PREEMPTIN line low.

2. The DMA Controller begins the arbitration cycle by pulling/GRANT high.

3. The DMA Controller requests access to the Private Bus from the CPU by pulling the HOLDR line high.

4a. The CPU completes its current cycle and releases the bus, signaling the DMA controller by pulling the HOLDA line high.

4b. At the same time, all devices requesting service drive the Arbitration bus with their priority code. They adjust until the code on the bus matches the code of the device with the highest priority (lowest number).

5. After allowing time for the Arbitration bus to stabilize, the DMA Controller pulls the /GRANT signal low to lock in the code of the device to be serviced. This signals the winning device that DMA transfer may begin.

FLOPPY DISK ARBITRATION

The system board floppy disk controller notifies the DMA controller chip that it is ready to perform a DMA transfer by asserting FDREQ. The

DMA controller then sets /FDPRMT to request access to the Micro Channel bus. During the ensuing arbitration cycle the DMA controller completes for the bus by driving FDARB0 and FDARB23 if the mask bit for channel 2 is not set (i.e. DMA is enable for that channel). Arbitration attempts continue until channel 2 wins an arbitration cycle. Then the DMA controller asserts /FDACK and proceeds to do a non-bursting transfer for the floppy controller.

DRAM REFRESH LOGIC

DRAM refresh is controlled by the DMA Controller. This is performed as if it were a request from a peripheral device for DMA service. The sequence is:

1. The refresh timer generates a request for refresh every 15 microseconds.

2. The DMA Controller generates a PREEMPT and waits for the end of transfer.

3. The DMA Controller generates an arbitration cycle. The refresh request automatically wins access. The Grant mode is not entered.

4. The refresh address is transferred to the address bus and /REFRESH, MEM, STATUS and READ signals are issued.

5. The DMA controller completes the refresh cycle and returns control of the system either to the CPU or to a device on the channel if a preempt was active.

NUMERIC COPROCESSOR INTERFACE

The 80387 and 80287 Numeric Coprocessors require some support logic to operate in the system. The DMA Controller has this circuitry included.

Address Decoding and Bus Timing: The Numeric Coprocessor is located at I/O address 00FXH. The Chip Select (/NCPCS) and Busy (/CPUBUSY) signals are generated to handshake with the Bus Controller.

Error Processing: The DMA Controller uses the Numeric Coprocessor error signal (/NPERROR) signal to generate the coprocessor error interrupt (/RQ13).

Reset: The DMA Controller uses the system reset signal (/RESET) or the I/O write to 00F0 (software reset) to provide the Numeric Coprocessor reset signal (NPRESET).

Pin Symbol	Pin Numbers	Pin Type	Description
CPU AND BUS CONTROLLER INTERFACE			
A20-A23	73-76	O	Address lines 1 and 16 through 23. These are active high, three state outputs used to address the memory and I/O. Note that these addresses are not used as inputs to the DMA Controller.
A16-A19	68-71	O	
A12-A15	63-66	IO	
A8-A10	55-57	IO	
A5-A7	50-53	IO	
A2-A4	47-48	IO	
A1	44	IO	
/BE0	45	O	Byte Enable 0. A three state, active low signal by the DMA controller to address I/O and memory.
/BE1	46	O	Byte Enable 1.A three state, active low signal used to the DMA Controller to address I/O and memory.

/BE0 and /BE1 identify the location of the bytes of data on the bus as follows:

/BE0	/BE1	Bits	Locations
0	1	8	0-7
1	0	8	8-15

	0	0	16	0-15
	1	1	not used	
/BUSY	32	O	DMA Busy. An active low output which tells the Bus Controller that a DMA transfer is in progress.	
/DMAADL	37	I	DMA Address Latch. An active low input used to qualify data on the address bus.	
/CB0E	114	I	Channel Byte 0 Enable. An active low input from the channel, similar to BE0N above. This signal is used to address the DMA Controller in the program mode.	
/CWLE	113	I	Channel Word Low Enable. An active low input used to steer bytes of data. This signal is used to address the DMA Controller in the program mode.	
/CCMD	122	I	Channel Command. An active low input used to qualify data on the bus.	
CMEM	118	I	Channel Memory/-I/O. An active high input used to differentiate between Memory (high) and I/O (low) operations.	
/CS0	117	I	Channel Status Bit 0. An active low input used to differentiate between I/O and Memory writes and reads.	
/CS1	116	I	Channel Status Bit 1. An active low input used to differentiate between I/O and Memory writes and reads.	

/CS0, /CS1, and CMEM are interpreted as follows:

CMEM	/CS0	CS1	Operation
0	0	1	I/O Write
0	1	0	I/O Read
1	0	1	Memory Write
1	1	0	Memory Read
1	0	0	Interrupt Acknowledge

/S0OUT	81	O	DMA cycle status bit 0. Indicates the type of cycle required by the DMA. The	

				interpretation of the combination of this bit, /S1OUT and PMEM is identical to the table above for /CSO, /CS1 and CMEM
	/S1OUT\	79	O	DMA cycle status bit 1.
	/DMACS	97	O	DMA Chip Select. An active low output which indicates that the DMA Controller is responding to a command directed to it. This signal is used by the Bus Controller to control the bus drivers to properly direct the data bus signals
	/GCSENA	78	I	Global chip select enable. Active low signal which enables the internal address decode for access to internal registers. Normally tied to the system processor HLDA.
	HOLDA	31	I	Hold Acknowledge. An active high input driven by the CPU to indicate that the processor has relinquished the bus.
	HOLDR	30	O	Hold Request. An active high output used to request that the CPU enter its hold condition.
	D0	128	IO	Data Bus. Bi-directional, three-state.
	D1-D3	129-131	IO	
	D4-D7	133-136	IO	
	D8	140	IO	
	D9-D12	144-147	IO	
	D13-D15	149-151	IO	
	/NMI	34	IO	Non-maskable Interrupt. A bi-directional, three-state signal used in an open collector configuration. This signal is generated by the DMA Controller at the end of an arbitration bus time-out to indicate an error on the bus. As an input, this signal forces

			the central arbitration control point into the arbitration mode.
/PADS	110	IO	Processor Address Strobe. This three-state, active low signal is used in the address decoding sequence to control latching of the address and control signals. It is provided by the DMA Controller during DMA transfers.
PMEM	111	IO	Processor Memory/-I/O. This three-state signal indicates whether the current processor cycle is a Memory (high) or I/O (low) operation. It is provided by the DMA Controller during DMA transfers.
/READY	36	I	Ready. An active low input. It is driven high by the Bus Controller to indicate the end of the current cycle.

MICRO CHANNEL INTERFACE

ARB0-3	94-91	I	Arbitration Bus. These input lines are driven high with the priority level of the arbitrating bus participants.
FDARB0	21	O	Floppy disk arbitration bit 0. The chip drives this line when the floppy disk has a request pending and it is arbitrating for the floppy. This line must be buffered externally to drive the arbitration bus line.
FLARB23	124	O	Floppy disk arbitration bit 22 and 3. The chip drives this line when the floppy disk has a request pending and it is arbitrating for the floppy. This line must be buffered externally through two separate drivers to drive the arbitration bus lines.
FDREQ	99	I	Floppy disk request. Request for DMA service from the system board floppy disk controller.

/FDACK	100	O	Floppy disk acknowledge. Indicates a grant of the DMA service request to the system board floppy disk controller.
/BURST	89	I	Burst. This active low input indicates that the winner of the arbitration is requesting operation in the burst mode.
/BUSTO	6	O	Bus Time-out. An active low output which indicates an arbitration time-out has occurred.
/GRANT	88	O	Grant. An active low output indicating the end of an arbitration cycle and authorizing the winning DMA device to begin its DMA cycle.
/PREEMPTIN	126	I	Preempt Input. This active low input indicating a bus arbitration request generated within the DMA Controller. The source of this signal is a refresh request.
/PREEMPT	87	O	Preempt. An active low output indicating a bus arbitration request generated within the DMA Controller. The source of this signal is a refresh request.
/FDPRE	22	O	Floppy disk preempt request. Generated when the system board floppy disk controller has a DMA request pending and the bus is in use. Must be buffered externally and tied to the Micro Channel PREEMPT line.
/TC	85	O	Terminal Count. An active low output indicating that the transfer count has reached zero.
RDY387	121	O	Ready signal from an installed 80387 to provide a means to insert wait states if required.
/BUSY386	159	O	Provides busy indication to the 80386 processor if required.

NUMERIC COPROCESSOR INTERFACE

/CPUBUSY	157	O	CPU busy. An active low output from the DMA Controller indicating that the Numeric Coprocessor is busy.
/IRQ13	153	O	Interrupt Request 13. An active low output indicating an error in the Numeric Coprocessor.
/NCPBUSY	155	I	Numeric Coprocessor Busy. An active low input signal from the Numeric Coprocessor that it is busy with a current operation.
/NCPCS	96	O	Numeric Coprocessor Chip Select. An active low output which indicates that the Numeric Coprocessor has responded to a command addressed to it. This signal is used by the Bus Controller to Control the bus drivers to properly direct the bus signals and to avoid bus contentions.
/NCPERR	156	I	Numeric Coprocessor Error. An active low input signalling an error condition in the Numeric Coprocessor.
NCPRST	154	O	Numeric Coprocessor Reset. An active low output which forces an 80287 Numeric Coprocessor to reset. (Note that different reset timing is required for an 80387).
RST387	1	O	Numeric Coprocessor Reset. An active low output which forces an 80387 Numeric Coprocessor to reset. (Note that different reset timing is required for an 80287).
STEN387	119	O	Status enable for an 80387. Enables the status output pins on the numeric coprocessor.

OTHER

REFRESH	138	O	Refresh. An active low output signal indicating that a refresh operations is in progress.
CLK119M	83	I	1.19 MHz Clock. Used for Refresh Timing.
CLKPRO	102	I	CPU Clock.
CLKSYS	107	I	System Clock.
/FASTREF	82	I	Fast Refresh. An active low input used to select the fast refresh rate.
INTR	35		Interrupt Request. This active high input signal is used to indicate the presence of any interrupt. This signal is the OR of all interrupt requests.
MASK2	84	O	Mask Register bit 2. This active high output indicates that the mask bit for channel 2 is set. This signal may be used by external logic to disable the arbitration for this channel when the mask bit is set.
/RESET	29	I	System Reset. Active low.
/REQRST	125	I	Request reset. This is a composite signal from the system board for requesting a reset of the system board processor.
/RSTPRO	104	O	Reset processor. This signal is caused by the presence of a /RSTPRO input and the fact that a DMA cycle is not in progress. It assures that partial DMA cycles (and therefore unrecoverable data error conditions) do not occur during a processor reset sequence.
PRORST	58	I	Processor reset. Inhibits initiation of a DMA cycle is a processor reset sequence is in progress.

DIAGNOSTICS/TEST

/GBURST	90	O	A test point which shows the burst timing within the chip.
/MBURST	105	O	The test point which shows the implied burst timing between a read and write cycle pair which are required to accomplish a DMA cycle.
DOREF	3	O	Test pin that indicates that a refresh is in progress for arbitration test purposes.
/REFREQ	4	I	Test point to request refresh from external source.
DISI00ARB	123	I	Disables the 100 nanosecond arbitration period option if pulled high.
ENREFRSH	143	I	Enables the internal refresh arbitration circuitry. Provided as a system diagnostic function.
ENGBURST	142	I	Enable the internal burst mode operation of the microchannel. Provided as a system diagnostic feature.
ENCAP	42	I	Enable central arbitration point. Test and diagnostic input which enables the central arbitration function in the chip. Must be tied high.
ENEOT	38	I	Enable End of Transfer. Enables the EOT output pin if in the high state. This pin is a test and diagnostic aid and must be tied high in the system.
EOT	5	O	Indicates and end of transfer condition which allows arbitration to proceed.
DIS78BTO	109	I	Disable 7.8 microsecond bus timeout. Disables the specified bus timeout for test and diagnostic purposes. Must be tied low in the system.

/DLYGNT	59	I	Delay grant. If low the grant for a bus access will be delayed during the arbitration sequence. Must be tied high in the system.
/TEST	24	I	Chip test signal. Not used in the system. Must be left unconnected.
T0,T1,T3,T4,T5	27-25,39,41	I	Test inputs. Not used in the system. Must be left unconnected.
DIAG0-7	19-16,14-11	O	Diagnostic outputs. Must be left unconnected.
DIAGIN0-3	9-7	I	Diagnostic test inputs. Must be left unconnected.
PAROUT	28	O	Parametric test output. Only for use in manufacturing. Must be unconnected.

MISCELLANEOUS

VDD	23,40,49,62,72 80,95,120	I	Power
VSS	2,10,15,20,33,43 54,60,67,77,86,98, 101,103,106,108, 115	I	Ground

CLAIMS

1. In a computer system having data storage means and input/output devices coupled thereto, a direct memory access system comprising:

transfer means for transferring data to and from locations in said data storage means and for transferring data from locations in said data storage means to and from said input/output devices via a data bus having data bus cycles; and,

manipulator means coupled to the transfer means for manipulating the data during said transfers.

2. A direct memory access system as in Claim 1 wherein said manipulator controls the number of data bus cycles required for transfer of data.

3. A direct memory access system as in Claim 1 wherein said manipulator means further includes sorter means for controlling the sequence of data being transferred.

4. A direct memory access system as in Claim 1 wherein said manipulator means further includes latch means, coupled to said transfer means and to said sorter means for receiving and transmitting data being transferred.

5. A direct memory access system as in Claim 1 wherein said manipulator means further includes data pre-processor means, coupled to said sorter means and to said transfer means for performing arithmetic and logical operations on data being transferred.

6. A direct memory access system as in Claim 2 wherein said sorter means performs operations on the data being transferred during said transfers.

7. A direct memory access system as in Claim 6 wherein said operations include controlling the amount of data being transferred.

8 A direct memory access system as in Claim 2 wherein:

said data includes words of data, each word comprising a plurality of bytes and having a word address;

said locations have odd and even byte addresses for storing bytes of data thereat; and,

said manipulator means controls transfer of data to and from locations having odd word addresses in 2N+1 bus cycles, where N is the number of words of data being transferred.

9. A direct memory access system as in Claim 8 wherein said manipulator means includes:

first selector means for selecting the word address of the data being transferred;

second selector means for selecting the byte address of the data being transferred;

said first selector means being effective for decrementing and incrementing the word address of the data being transferred.

10. A direct memory access system as in Claim 2 wherein:

said data is organized into words, each word having a plurality of bytes and having a word address; and

each transfer of data words comprises one bus cycle for a read operation and one bus cycle for a write operation.

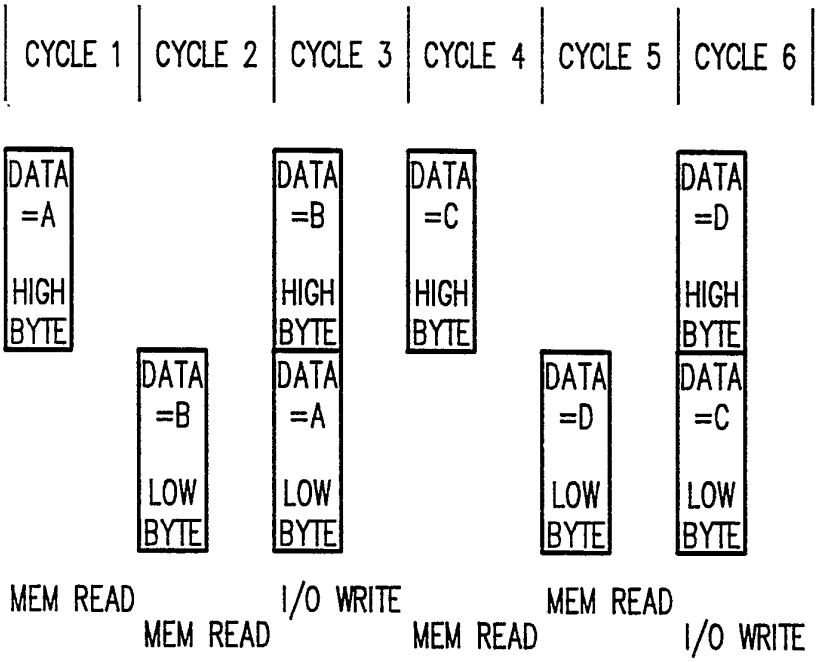
11. A direct memory access system as in Claim 10 wherein:

said locations having odd and even byte and word address; and

each transfer of one or more data words from an odd word address requires one additional bus cycle.

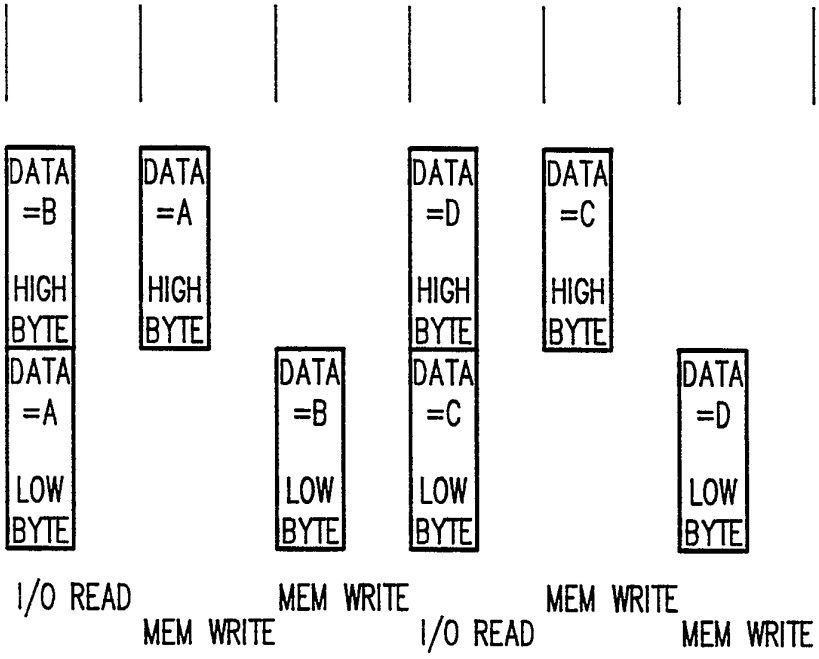
12. A direct memory access system as in Claim 10 wherein said manipulator means manipulates data being transferred between read and write operations.

13. A direct memory access system as in Claim 10 wherein the read operation for a data word being transferred occurs concurrently with the write operation for the data word previously read.



PRIOR ART

FIG 1A



PRIOR ART

FIG. 1B

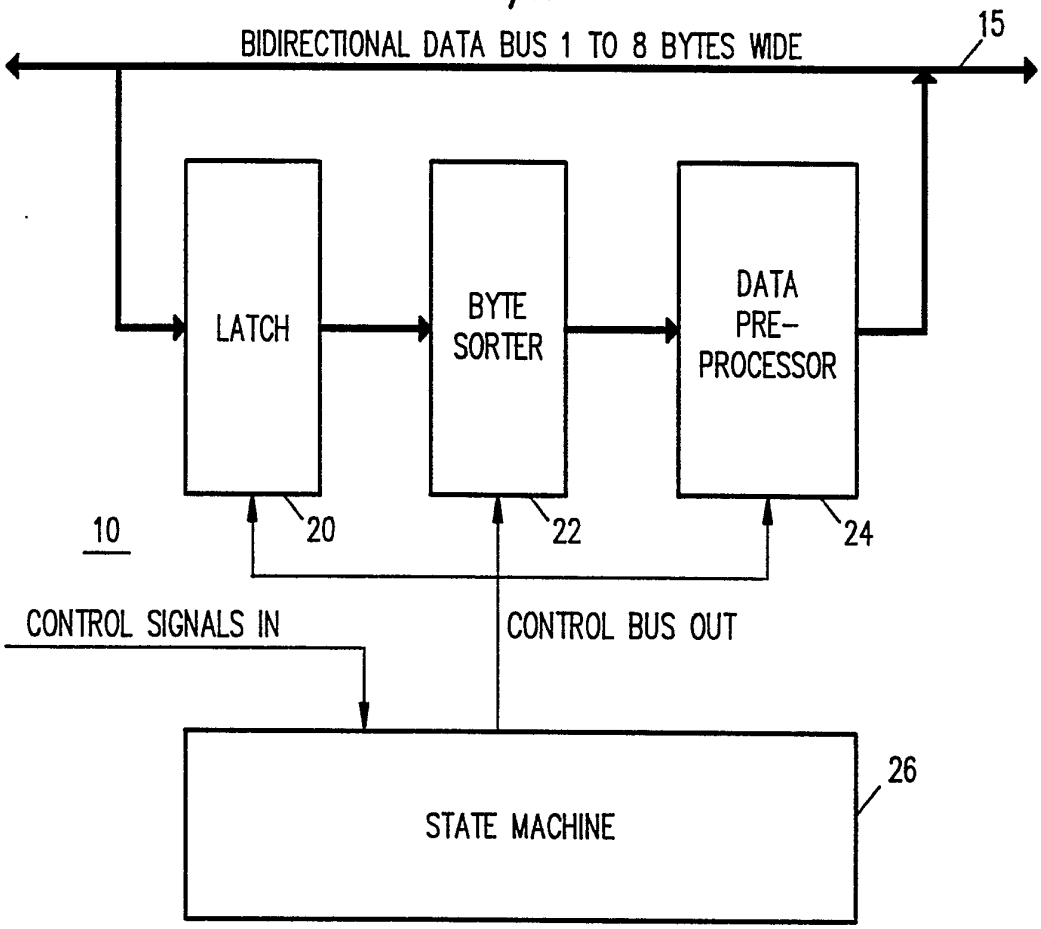

BIDIRECTIONAL DATA BUS 1 TO 8 BYTES WIDE
15
LATCH
BYTE SORTER
DATA PRE-PROCESSOR
10
20
22
24
CONTROL SIGNALS IN
CONTROL BUS OUT
STATE MACHINE
26

FIG. 2

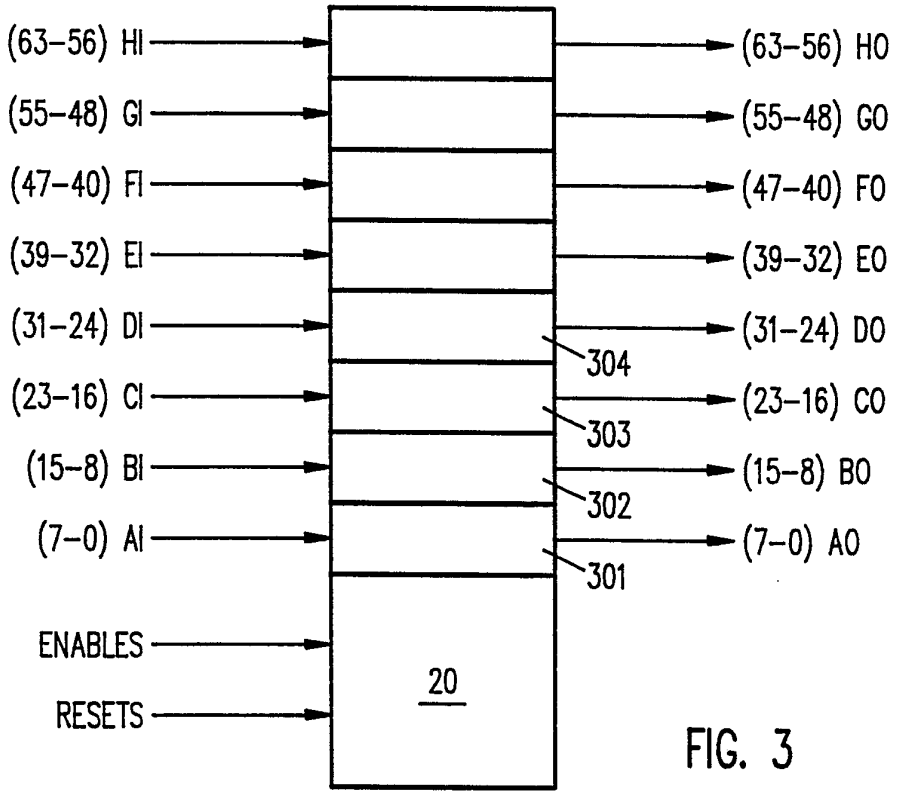

(63-56) HI
(55-48) GI
(47-40) FI
(39-32) EI
(31-24) DI
(23-16) CI
(15-8) BI
(7-0) AI
ENABLES
RESETS
20
(63-56) HO
(55-48) GO
(47-40) FO
(39-32) EO
(31-24) DO
(23-16) CO
(15-8) BO
(7-0) AO
304
303
302
301

FIG. 3

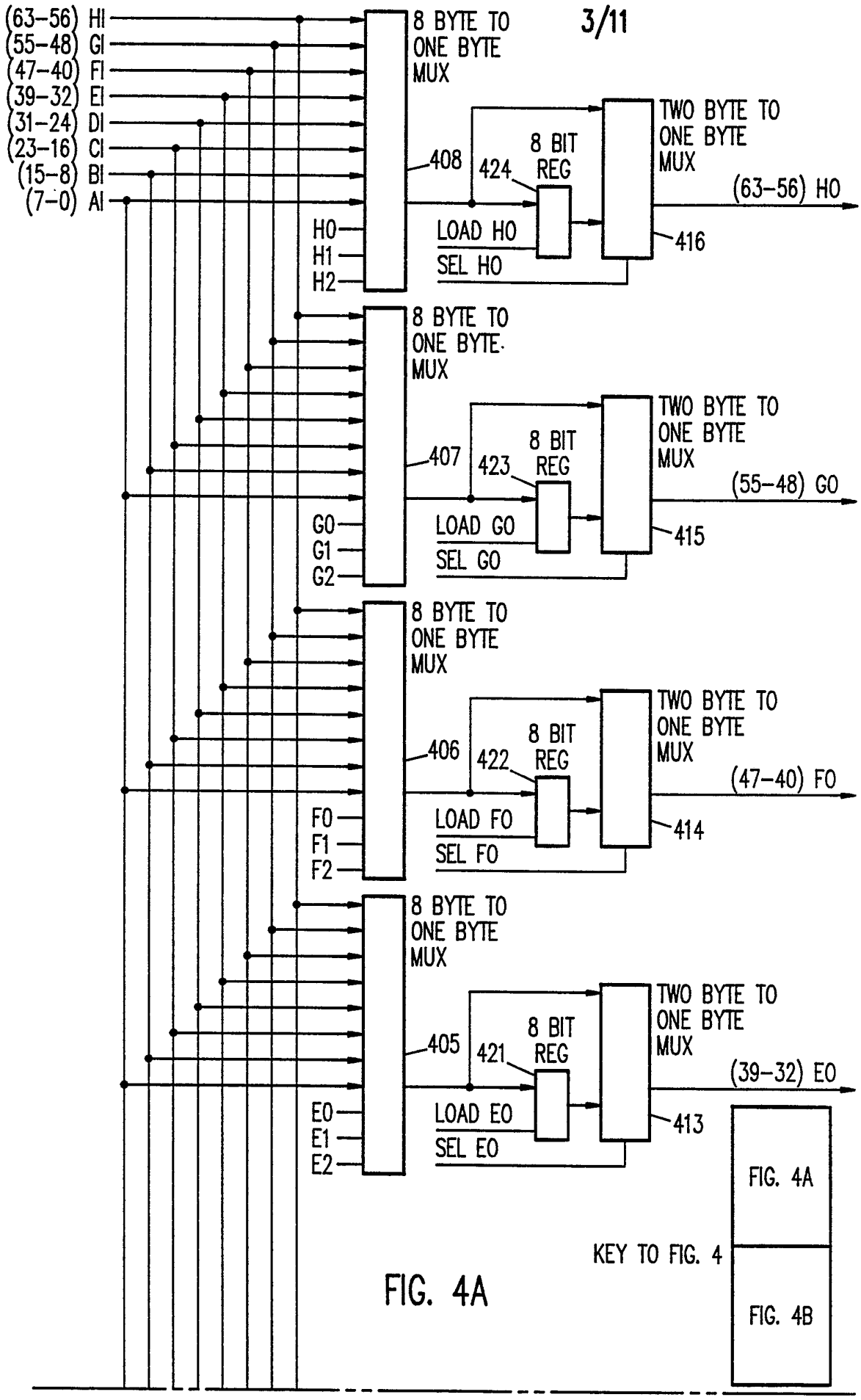

(63-56) HI
(55-48) GI
(47-40) FI
(39-32) EI
(31-24) DI
(23-16) CI
(15-8) BI
(7-0) AI
8 BYTE TO ONE BYTE MUX
H0
H1
H2
408
424
8 BIT REG
TWO BYTE TO ONE BYTE MUX
(63-56) HO
LOAD HO
SEL HO
416
8 BYTE TO ONE BYTE MUX
G0
G1
G2
407
423
8 BIT REG
TWO BYTE TO ONE BYTE MUX
(55-48) GO
LOAD GO
SEL GO
415
8 BYTE TO ONE BYTE MUX
F0
F1
F2
406
422
8 BIT REG
TWO BYTE TO ONE BYTE MUX
(47-40) FO
LOAD FO
SEL FO
414
8 BYTE TO ONE BYTE MUX
E0
E1
E2
405
421
8 BIT REG
TWO BYTE TO ONE BYTE MUX
(39-32) EO
LOAD EO
SEL EO
413
FIG. 4A
KEY TO FIG. 4
FIG. 4B

FIG. 4A

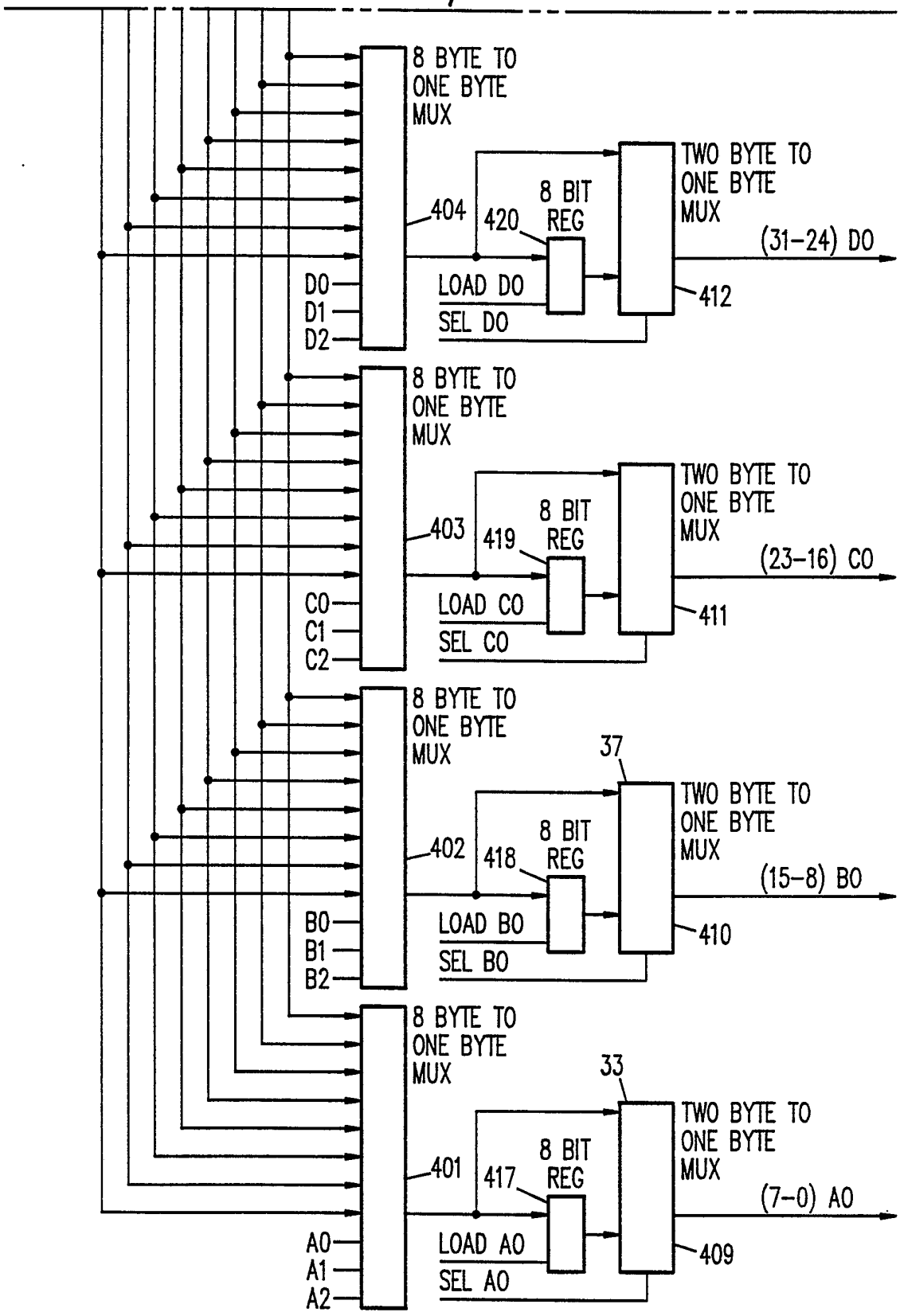

8 BYTE TO ONE BYTE MUX
404
420
8 BIT REG
TWO BYTE TO ONE BYTE MUX
(31-24) DO
412
D0
D1
D2
LOAD DO
SEL DO
8 BYTE TO ONE BYTE MUX
403
419
8 BIT REG
TWO BYTE TO ONE BYTE MUX
(23-16) CO
411
C0
C1
C2
LOAD CO
SEL CO
8 BYTE TO ONE BYTE MUX
37
402
418
8 BIT REG
TWO BYTE TO ONE BYTE MUX
(15-8) BO
410
B0
B1
B2
LOAD BO
SEL BO
8 BYTE TO ONE BYTE MUX
33
401
417
8 BIT REG
TWO BYTE TO ONE BYTE MUX
(7-0) AO
409
A0
A1
A2
LOAD AO
SEL AO

FIG. 4B

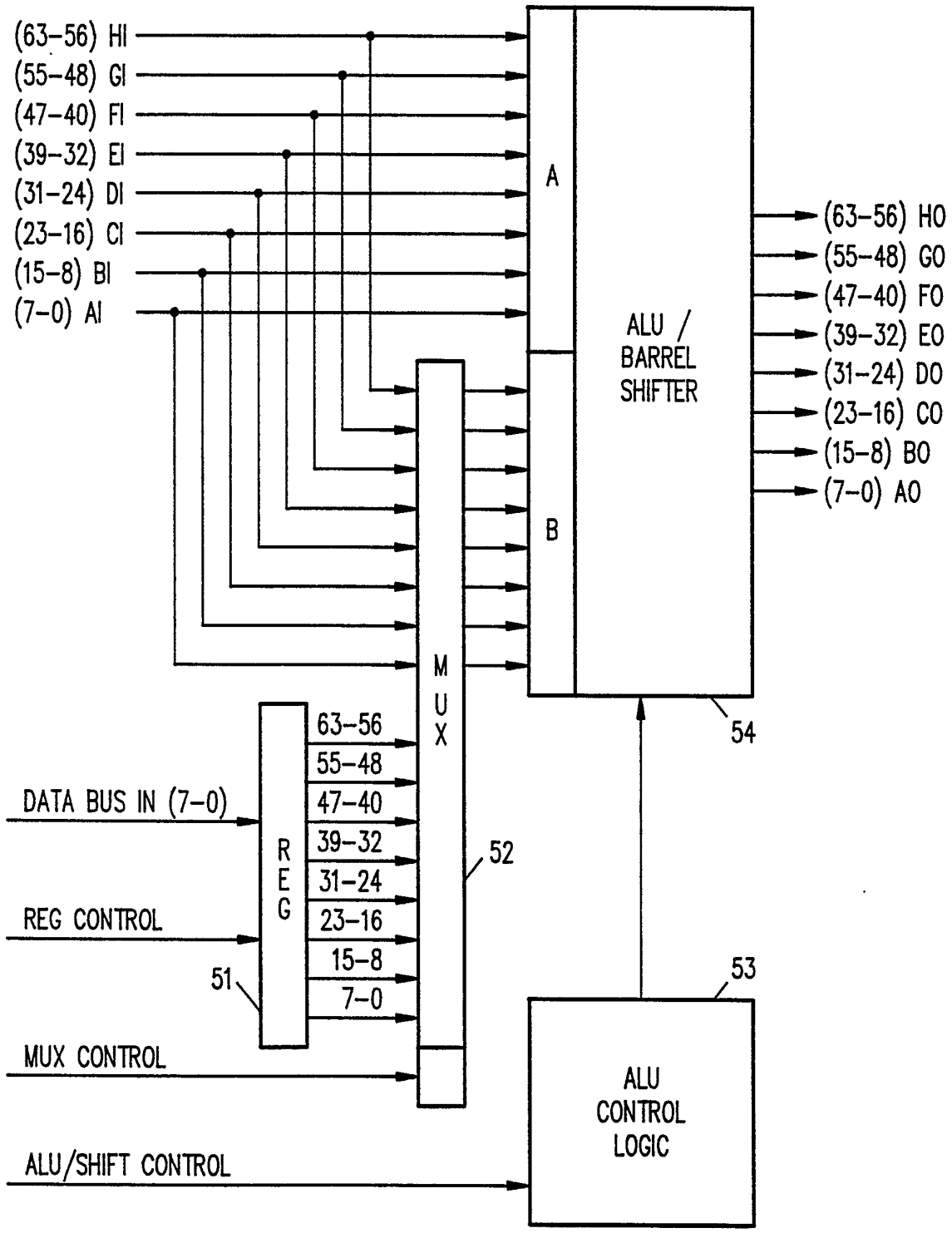

(63–56) HI
(55–48) GI
(47–40) FI
(39–32) EI
(31–24) DI
(23–16) CI
(15–8) BI
(7–0) AI
A
B
ALU / BARREL SHIFTER
(63–56) HO
(55–48) GO
(47–40) FO
(39–32) EO
(31–24) DO
(23–16) CO
(15–8) BO
(7–0) AO
54
M U X
52
63–56
55–48
47–40
39–32
31–24
23–16
15–8
7–0
DATA BUS IN (7–0)
REG CONTROL
REG
51
MUX CONTROL
ALU/SHIFT CONTROL
53
ALU CONTROL LOGIC

FIG. 5

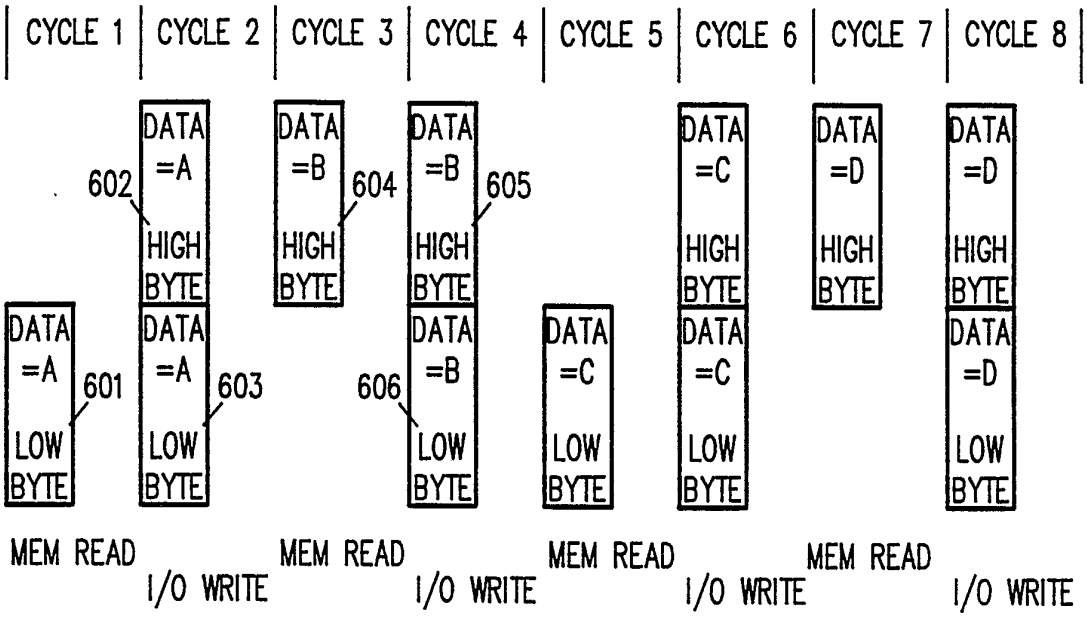


FIG. 6A

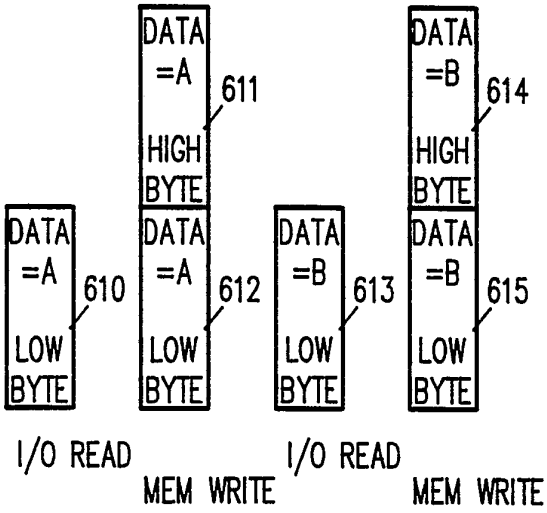


FIG. 6B

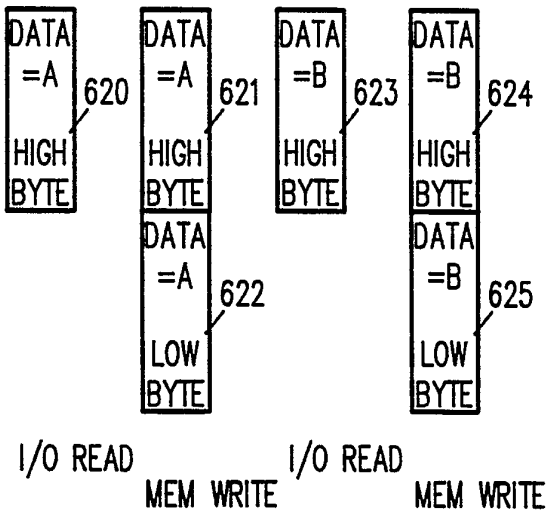


FIG. 6C

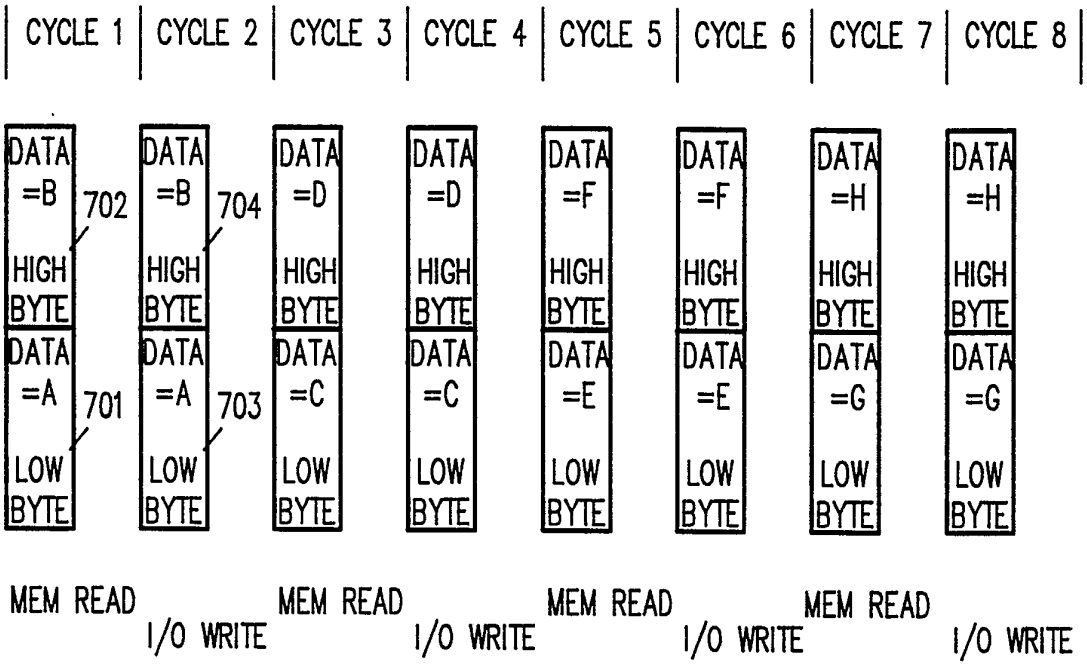

CYCLE 1
CYCLE 2
CYCLE 3
CYCLE 4
CYCLE 5
CYCLE 6
CYCLE 7
CYCLE 8
DATA =B HIGH BYTE
DATA =A LOW BYTE
702
701
DATA =B HIGH BYTE
DATA =A LOW BYTE
704
703
DATA =D HIGH BYTE
DATA =C LOW BYTE
DATA =D HIGH BYTE
DATA =C LOW BYTE
DATA =F HIGH BYTE
DATA =E LOW BYTE
DATA =F HIGH BYTE
DATA =E LOW BYTE
DATA =H HIGH BYTE
DATA =G LOW BYTE
DATA =H HIGH BYTE
DATA =G LOW BYTE
MEM READ
I/O WRITE
MEM READ
I/O WRITE
MEM READ
I/O WRITE
MEM READ
I/O WRITE

FIG. 7A

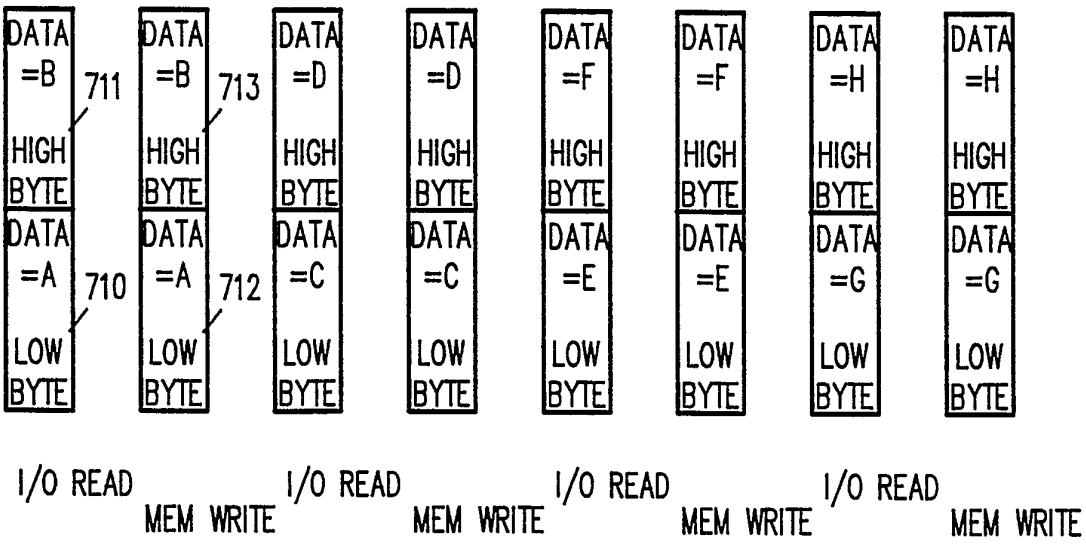

DATA =B HIGH BYTE
DATA =A LOW BYTE
711
710
DATA =B HIGH BYTE
DATA =A LOW BYTE
713
712
DATA =D HIGH BYTE
DATA =C LOW BYTE
DATA =D HIGH BYTE
DATA =C LOW BYTE
DATA =F HIGH BYTE
DATA =E LOW BYTE
DATA =F HIGH BYTE
DATA =E LOW BYTE
DATA =H HIGH BYTE
DATA =G LOW BYTE
DATA =H HIGH BYTE
DATA =G LOW BYTE
I/O READ
MEM WRITE
I/O READ
MEM WRITE
I/O READ
MEM WRITE
I/O READ
MEM WRITE

FIG. 7B

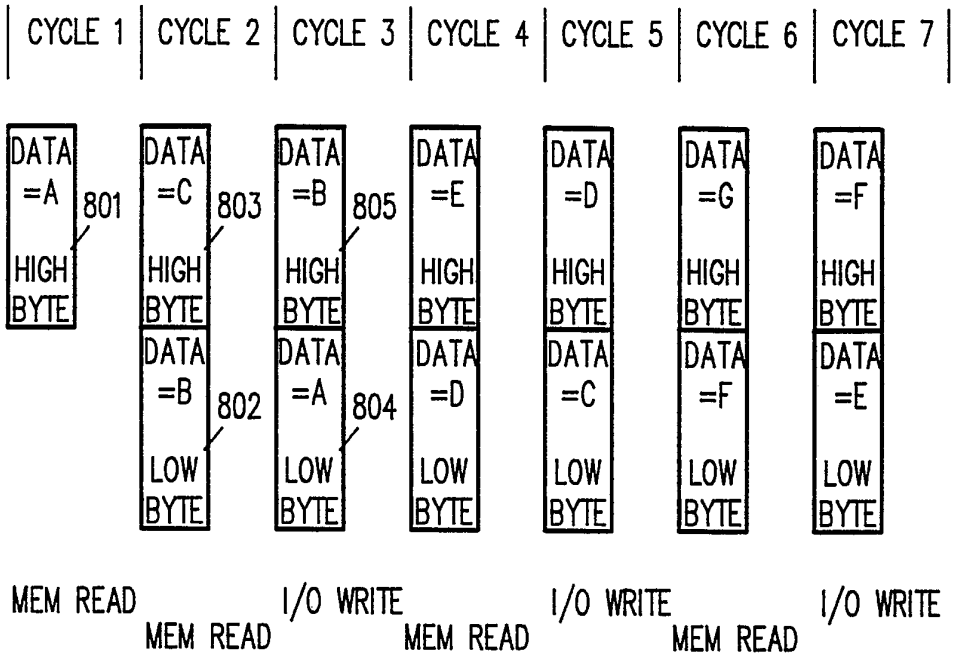

CYCLE 1
CYCLE 2
CYCLE 3
CYCLE 4
CYCLE 5
CYCLE 6
CYCLE 7
DATA =A
HIGH BYTE
801
DATA =C
HIGH BYTE
803
DATA =B
LOW BYTE
802
DATA =B
HIGH BYTE
805
DATA =A
LOW BYTE
804
DATA =E
HIGH BYTE
DATA =D
LOW BYTE
DATA =D
HIGH BYTE
DATA =C
LOW BYTE
DATA =G
HIGH BYTE
DATA =F
LOW BYTE
DATA =F
HIGH BYTE
DATA =E
LOW BYTE
MEM READ
MEM READ
I/O WRITE
MEM READ
I/O WRITE
MEM READ
I/O WRITE

FIG. 8A

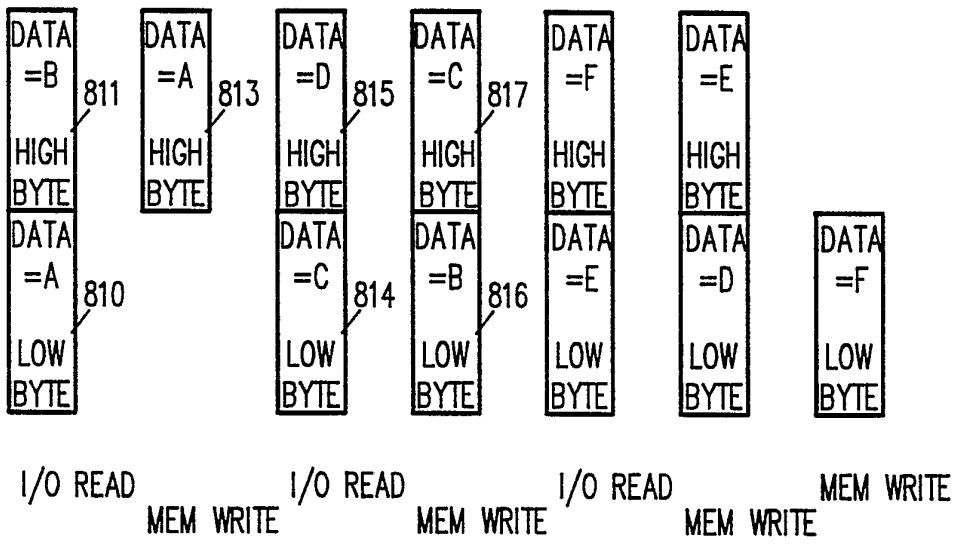

DATA =B
HIGH BYTE
811
DATA =A
LOW BYTE
810
DATA =A
HIGH BYTE
813
DATA =D
HIGH BYTE
815
DATA =C
LOW BYTE
814
DATA =C
HIGH BYTE
817
DATA =B
LOW BYTE
816
DATA =F
HIGH BYTE
DATA =E
LOW BYTE
DATA =E
HIGH BYTE
DATA =D
LOW BYTE
DATA =F
LOW BYTE
I/O READ
MEM WRITE
I/O READ
MEM WRITE
I/O READ
MEM WRITE
MEM WRITE

FIG. 8B

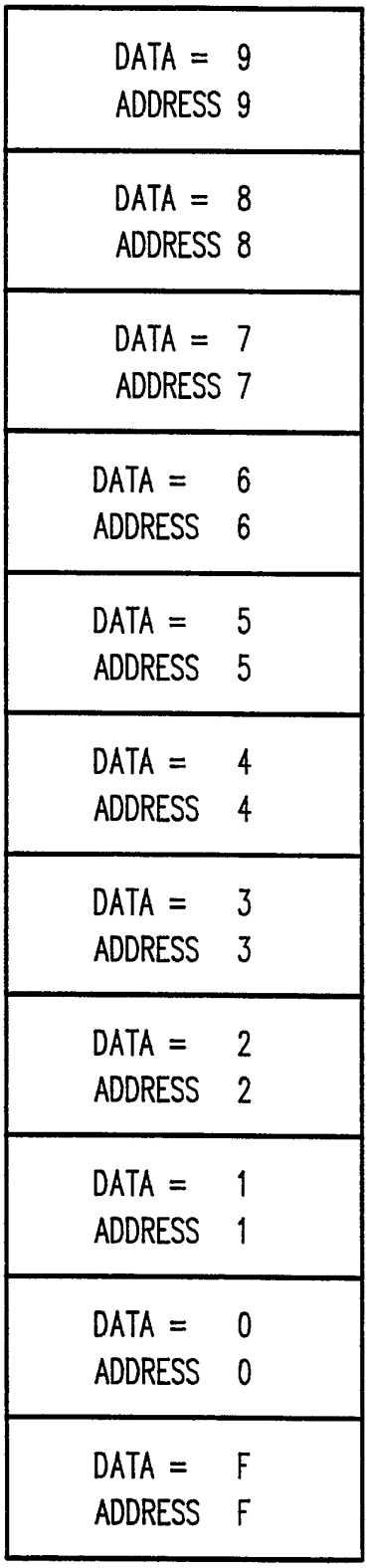
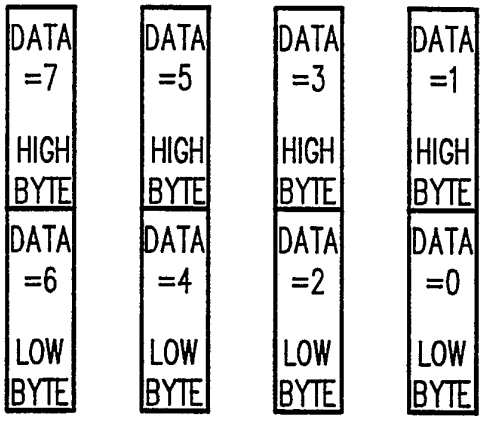

DATA = 9
ADDRESS 9
DATA = 8
ADDRESS 8
DATA = 7
ADDRESS 7
DATA = 6
ADDRESS 6
DATA = 5
ADDRESS 5
DATA = 4
ADDRESS 4
DATA = 3
ADDRESS 3
DATA = 2
ADDRESS 2
DATA = 1
ADDRESS 1
DATA = 0
ADDRESS 0
DATA = F
ADDRESS F

FIG. 9A



PRIOR ART

FIG. 9B

16 BIT EVEN DECREMENT TRANSFER FROM MEMORY ADDRESS 6

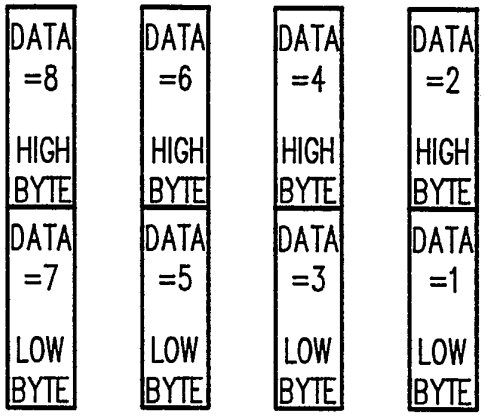


FIG. 9C

16 BIT ODD DECREMENT TRANSFER FROM MEMORY ADDRESS 7

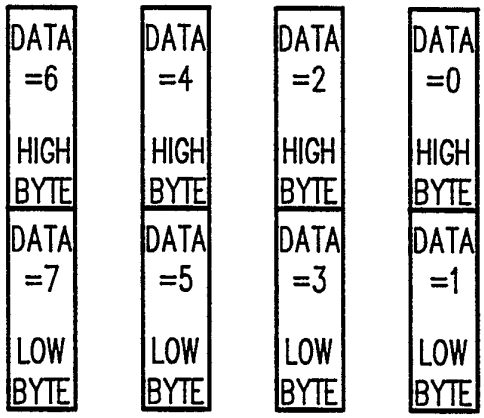


FIG. 9D

16 BIT ODD DECREMENT TRANSFER FROM MEMORY ADDRESS 7

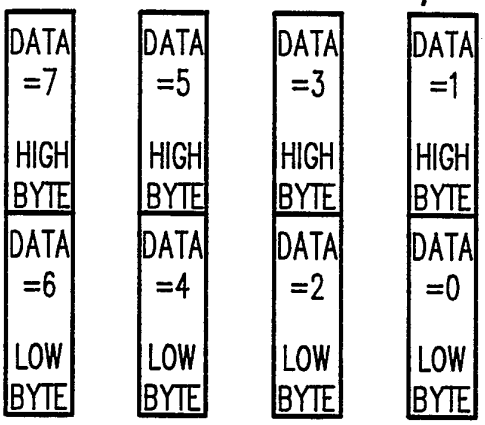


FIG. 9E

16 BIT ODD DECREMENT TRANSFER FROM MEMORY ADDRESS 7

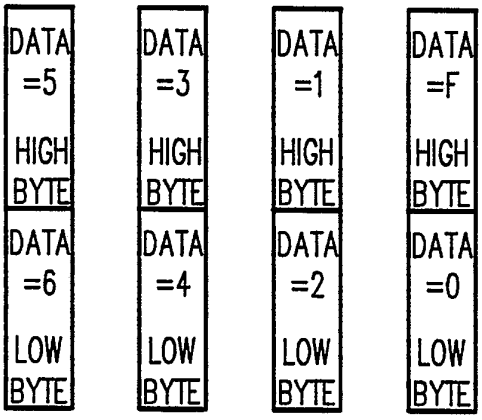


FIG. 9F

16 BIT ODD DECREMENT TRANSFER FROM MEMORY ADDRESS 7

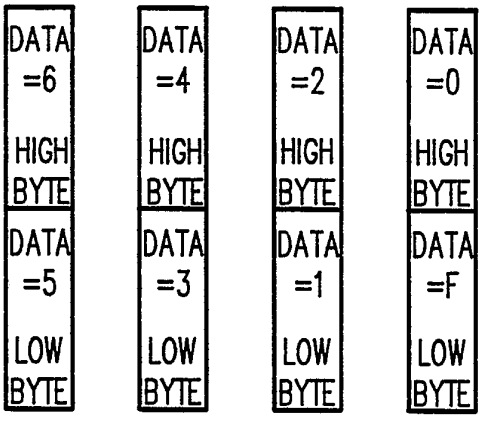


FIG. 9G

16 BIT ODD DECREMENT TRANSFER FROM MEMORY ADDRESS 7

INTERNATIONAL SEARCH REPORT

International Application No. PCT/US90/01405

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [6]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC(5): G06F 12/04
U.S. CL.: 364/200, 900

II. FIELDS SEARCHED

Minimum Documentation Searched [7]

Classification System	Classification Symbols
U.S.	364/900

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched [8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [9]

Category *	Citation of Document, [11] with indication, where appropriate, of the relevant passages [12]	Relevant to Claim No. [13]
X P	US, A, 4,845,664 (AICHELMANN, JR. ET AL.) 04 JULY 1989, see the entire document.	1-4,6-7
X	US, A, 4,408,275 (KUBO ET AL.) 04 October 1983, Figs. 1 and 2, Column 3 - column 5.	1,2,6-13
X	US, A, 4,722,050 (LEE ET AL.) 26 January 1988, Fig 4, column 5, line 35 - colomn 6, line 9.	1,5
A	US, A, 4,447,878 (KINNIE ET AL.) 08 May 1984, see the entire document.	1-13
A	US, A, 4,633,434 (SCHEUNEMAN) 30 December 1986, see the entire document.	1-13
A	US, A, 4,439,827 (WILKES) 27 MARCH 1984, see the entire document.	1-13

* Special categories of cited documents: [10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
29 MAY 1990	18 JUL 1990
International Searching Authority	**Signature of Authorized Officer**
ISA/US	REBECCA L. RUDOLPH