PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ^{5} : **G06F 13/32**	**A1**	**(11) International Publication Number:** **WO 92/17846** **(43) International Publication Date:** 15 October 1992 (15.10.92)

(21) International Application Number: PCT/US92/02292

(22) International Filing Date: 20 March 1992 (20.03.92)

(30) Priority data:
675,448 26 March 1991 (26.03.91) US

(71) Applicant: ECHELON CORPORATION [US/US]; 4015 Miranda Avenue, Palo Alto, CA 94304 (US).

(72) Inventors: SINKS, Rod, G. ; 10316 Denison Avenue, Cupertino, CA 95014 (US). DONNER, Robert, W. ; 2580 Bean Creek Road, Scotts Valley, CA 95066 (US).

(74) Agents: TAYLOR, Edwin, H. et al.; Blakely, Sokoloff, Taylor & Zafman, 12400 Wilshire Boulevard, 7th Floor, Los Angeles, CA 90025 (US).

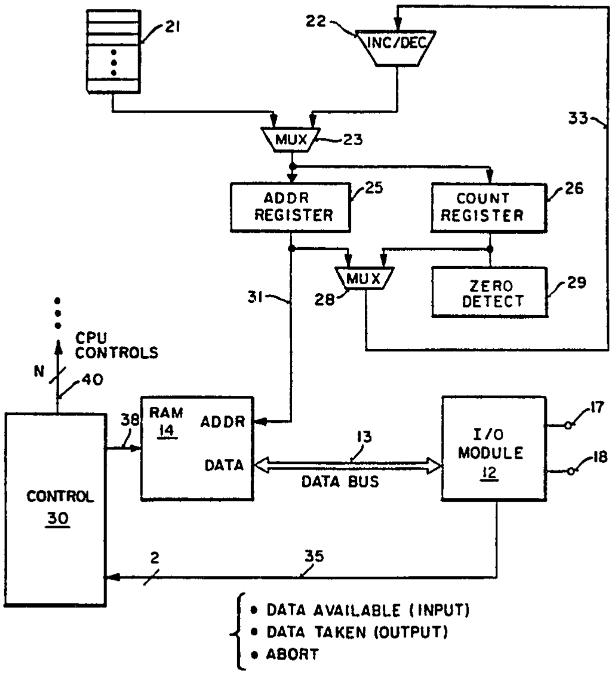
(81) Designated States: AT, AT (European patent), AU, BB, BE (European patent), BF (OAPI patent), BG, BJ (OAPI patent), BR, CA, CF (OAPI patent), CG (OAPI patent), CH, CH (European patent), CI (OAPI patent), CM (OAPI patent), DE, DE (European patent), DK, DK (European patent), ES, ES (European patent), FI, FR (European patent), GA (OAPI patent), GB, GB (European patent), GN (OAPI patent), GR (European patent), HU, IT (European patent), JP, KP, KR, LK, LU, LU (European patent), MC (European patent), MG, ML (OAPI patent), MR (OAPI patent), MW, NL, NL (European patent), NO, RO, RU, SD, SE, SE (European patent), SN (OAPI patent), TD (OAPI patent), TG (OAPI patent).

Published
With international search report.

(54) Title: APPARATUS AND METHOD FOR FAST I/O TRANSFER

(57) Abstract

A specialized apparatus and method for providing fast programmed I/O for transferring information in a multiprocessor environment which includes a CPU (11) and a memory (14) coupled to an I/O port (12) across an internal data bus (13). Multiple bytes of data are transferred in successive processing cycles to the I/O port (12) from the memory (14), or from the I/O (12) to the memory (14) by first determining the upper limit for the number of bytes that are going to be transferred. This number and the memory start address are then stored in CPU registers. The I/O module (12) is then checked by the CPU (11) to see if a data byte is available from an external device. If a data byte is available, the I/O module (12) is instructed to place the data byte on the bus (13) for storage within the memory (14) at the start address. The address is then incremented and the count is decremented. The above procedure is repeated until the count drops to zero, after which time the next instruction is fetched.



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

"Apparatus and Method for Fast I/O Transfer"

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates to the field of data transfer in a microprocessor; particularly in the case where a multi-processor is employed within an intelligent cell used in a network for sensing, communicating and controlling.

2. Background Information

There are a number of commercially available products which provide sensing, controlled and communications in a network environment. These products range from elaborate systems having a large amount of intelligence to simple systems having little intelligence. By way of example, such a system may provide control between a light switch and a light; when the light switch is operated, a digital code pattern is transmitted by one cell over power lines or free space and is received by another cell at the light. When the code is received, it is interpreted and subsequently used to control the light.

Such a system -- comprising a network of intelligent cells in which the cells communicate, control and sense information -- is described in U.S. Patent 4,947,484, which application is assigned to the assignee of the present invention. Figure 10 of U.S. Patent No. 4,947,484 illustrates a block diagram of an intelligent cell capable of communicating, controlling and sensing information. As can be seen, such cells typically comprise a plurality of processors coupling an internal memory to an input/output (I/O) block across address and data buses. The memory portion of the intelligent cell may include a ROM for storing instructions, a RAM for buffer storage, and an EEPROM for holding configurable information or other information that needs to be retained in a non-volatile way. The I/O block includes a communications port having many functions programmed for various communications tasks. Observe that the intelligent cell of Figure 10 further includes a clocking means and associated timing control elements.

Practically all processors require some means of transferring information either from an internal memory to an external communications port, or vis-a-versa. For example, in the processing system described in the above-referenced U.S. patent, a plurality of intelligent cells are shown being distributed along a network. These cells exchange information over some communications backbone, such as a transceiver. Individual cells each contain a multi-processor composed of several microprocessors. In a distributed intelligence environment such as this, where there exists a need to control or sense things remotely, cells are serviced by a communication port that allows each of the intelligent computers to communicate with all of the other intelligent computers over a variety of media.

Within this type of distributed environment, there may be certain applications for which the processing power inside individual cells may be inadequate. For instance, if a cell were given the task of controlling a factory floor and the algorithm used for calculating the control information were very compute-intensive (e.g., there were extensive multiplication or division calculations required) then there may also be a need to communicate between the intelligent cell and a high-speed industrial computer, e.g., a main frame computer capable of floating point operations. In such a situation, data transfer and communications within the cell takes place over a general purpose I/O bus coupled to an external port of the cell.

It is important to understand that the level of performance achieved in transferring information to/from a particular cell may not be governed by how fast the individual microprocessors perform within the cell, but by how fast the address and data pipelines (i.e., buses) operate between the memory and a communications port or between memory and an I/O port.

One traditional mechanism for communicating with memory is known as programmed I/O. According to the programmed I/O scheme, information is transferred a single byte at a time. For every byte of information to be transferred, the processor must run an implicit instruction which, in the case of a data transfer out, first involves fetching a byte from a certain location within the RAM, then transferring it through the central processing unit (CPU) and then out to the communications port.

In other words, all processing is done on a byte-by-byte basis. In a given transfer (either into or out of the processing system), only a single byte is transferred for each cycle. In order to transfer multiple bytes (e.g., 256 bytes) in a programmed I/O scheme, single byte data must be repetitively moved from a location in RAM to the CPU, then from the CPU to the communications port. Each move requires execution of a separate processor instruction. Understandably, the primary drawbacks of programmed I/O are its extreme slowness and the burden it places on the CPU for transferring each byte.

In response to the relative slowness of programmed I/O, a mechanism referred to in the industry as direct-memory-access (DMA) was developed. DMA remains a popular way of transferring data between a memory and an I/O or communications port in a computer system which demands fast data transferring capability.

According to DMA, specialized controller circuitry is incorporated into both the memory and communications ports to allow each unit to directly seize control of the internal buses to access memory directly. As a result, DMA is an especially fast method of transferring data, particularly in a shared-memory architecture employing shared I/O buses.

The main problem with the DMA approach is that it involves a great deal of complexity. That is, there is a large amount of circuitry overhead associated with DMA for each of the participants. When implemented on an integrated circuit, one must realize that the superior performance of DMA comes at the expense of die area.

The reason for this is because a separate set of DMA hardware needs to be built into the communications port and each of the I/O blocks coupled to the internal data bus. DMA hardware normally comprises a state machine to seize control of the bus at precisely the right moment, and also some arbitration logic to insure that during DMA operations the microprocessor does not create a bus ownership conflict. Thus, while DMA offers a substantial increase in speed, this performance increase comes at the cost of a large hardware overhead.

The speed difference between DMA approaches and programmed I/O creates a dilemma for many intelligent cells of the type described which currently need to operate their communications and I/O ports at rates of approximately 1 Mbit/sec. In a typical cell of the type described, programmed I/O is approximately four times too slow to support the required communication rate, whereas direct memory access logic is at least an order of magnitude faster than needed. Recall that DMA circuitry generally requires a substantial increase in DMA-dedicated silicon area. (Buses and many other various signals must be brought together to accommodate DMA which requires a significant increase in routing area). In effect, DMA provides more performance than is needed at considerable additional cost.

Thus, there exists an unrequited need for an alternative to both programmed I/O and DMA, which operates at an intermediate data transfer rate.

SUMMARY AND OBJECTS OF THE INVENTION

A specialized apparatus and method for providing fast programmed I/O for transferring information in a multi-processor environment is described. The invented apparatus and method is embodied in a processor instruction which facilitates high speed data transfer.

In one embodiment, an intelligent cell or processing element includes a CPU, a memory, and an I/O port. The memory is coupled to the I/O port across an internal data bus. According to the fast I/O instruction of the present invention, multiple bytes are transferred in one processing cycle to the I/O port from the memory, or from the I/O to the memory.

In accordance with the preferred embodiment, the upper limit for the number of bytes that are going to be transferred is first determined. This number is placed in the top of an 8-byte stack register. In the next instruction cycle, the number of bytes indicated in the top of stack are transferred out without the need to go through explicit instruction fetches for each byte (which involves multiple memory operations).

One of the beneficial features of the present invention is that it takes advantage of the existing hardware within the central processing unit (CPU). This latter point is important since by avoiding the need for additional control hardware, the present invention overcomes the drawbacks characteristic of conventional prior art approaches (e.g., direct memory access mechanisms).

Thus, the invention is able to implement data transfers at a rate which is approximately five times faster than traditional programmed I/O -- yet without the hardware overhead inherent with a DMA scheme. Ultimately, this reduces the overall die size and simplifies the design.

BRIEF DESCRIPTION OF DRAWINGS

Figure 1 is a generalized block diagram of an intelligent cell for communication, control, and sensing.

Figure 2 is a block diagram of the currently preferred embodiment of the fast input/output (I/O) instruction of the present invention.

Figure 3 is a flowchart depicting the sequence of control operations executed during a fast I/O instruction, wherein information is transferred into an intelligent cell.

Figure 4 is a flowchart depicting the sequence of control operations executed during a fast I/O instruction, wherein information is transferred out of an intelligent cell.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

An apparatus and method for achieving fast programmed I/O in a computer system is described. While the present invention will be described in conjunction with a network providing communications, sensing and control through a plurality of intelligent cells, it is appreciated that the present invention may be practiced in other applications without detracting from the spirit and scope of the present invention.

In the following description, numerous specific details are set forth such as specific number of bytes, etc., in order to provide a thorough understanding of the invention. It will be obvious, however, to one skilled in the art that these details are not required to practice the invention. In other instances, well-known circuits, methods and the like are not set forth in detail in order to avoid unnecessarily obscuring the present invention.

In Figure 1, a block diagram of an intelligent cell 10 is shown. Cell 10 is one of a plurality of cells in a system comprising a network organized in a hierarchy based on communications needs. Cells are organized into working "groups" independent of the network hierarchy. Groups of cells generally are used to perform a group function by the assignment of tasks to cells within the groups. Cells communicate, control and sense information. In general, each cell has a unique identification number and performs information processing tasks such as: Bi-directional communications protocol, input/output, packet processing in analog and digital sensing and control. In general, the system comprised of the cells has the characteristic of storing network configuring information that is distributed throughout the system. The system also communicates automatically routed messages among the cells.

In Figure 1, cell 10 includes multiple central processing units (CPUs) 11 or some similar multi-processor. Also included are input/output (I/O) modules 12 which, by way of example, are shown providing a communications port 17 together with an application I/O port 18, frequently referred to simply as an I/O port. Also shown is a random-access memory (RAM) 14 coupled to I/O modules 12 across an internal data bus 13. Multi-processor 11 provides control signals to, and receives responses from, I/O modules 12 along lines 19. Likewise, multi-processor 11 provides control signals to RAM 14 along lines 15.

In the currently preferred embodiment, multi-processor 11 comprises three processors. Any of the three processors can utilize the fast I/O instruction of the present invention to transfer data to/from RAM 14 and I/O modules 12 across bus 13. RAM 14 ordinarily comprises a plurality of static memory cells, although dynamic cells can also be used. In other embodiments, the memory section of cell 10 may further include non-volatile data storage, and/or a read-only memory (ROM) for storing instruction codes. As previously mentioned, the fast I/O instruction of the present invention permits transfers of multiple bytes of data -- either into or out of cell 10 -- by invoking a single processor instruction.

The present invention is actually embodied in two separate processor instructions. First, the "Fast I/O IN" instruction automatically moves data from an I/O module into the on-chip RAM. Conversely, the "Fast I/O OUT" instruction automatically exports data from the on-chip RAM to one of the I/O modules, where it can then be ported or transferred to another cell or computer system.

Referring to Figure 2, a low level block diagram of the apparatus for implementing the instruction of the present invention is shown. In the currently preferred embodiment, the block diagram of Figure 2 comprises the use of CMOS technology wherein the present invention is incorporated into cell 10 of Figure 1, and fabricated as a single integrated circuit. Except for RAM 14, data bus 13 and I/O modules 12, each of the devices illustrated in Figure 2 comprise existing hardware elements normally integrated within CPU 11.

Before a fast I/O instruction may be executed, the CPU and the selected I/O module must first be prepared. To start, the I/O module is programmed by the CPU to initiate a data transfer to/from a device that is external to the cell. The I/O module is also prepared for data transfer to/from the CPU. Next, the RAM start address and the transfer count are pushed onto a stack 21. The transfer count represents the total number of bytes which will be transferred by a single processor instruction. In the currently preferred embodiment, up to 256 bytes may be transferred in one instruction cycle.

It is appreciated that stack 21 may be substituted with various other register means in different architectures. By way of example, an ordinary register file could be substituted for stack 21, or the CPU might place the address and count information in appropriate registers before executing a fast I/O instruction. Thus there exist a variety of methods for performing the set-up portion of the fast I/O instruction. Each of these means or methods is considered well within the spirit and scope of the present invention.

Each of the "Fast I/O IN" and "Fast I/O OUT" instructions of the present invention can be divided into two separate functions: Set-up and execution. During the set-up portion of a fast I/O instruction, the RAM start address is retrieved from stack 21 and moved to a CPU address register 25 via multiplexer 23. The transfer count is also moved from stack 21 and placed in a CPU count register 26; the transfer also occurring through multiplexer 23. The set-up portion of a fast I/O instruction is now complete. (Note that in Figure 2, the respective device elements, such as stack 21, MUXs 23, 28, etc., are shown being controlled by CPU control logic block 30. The individual control lines are represented by lines 40).

The execution portion of a Fast I/O IN instruction begins with CPU control block 30 checking the status of I/O handshaking bus 35. This line indicates whether data is presently available. More generally, bus 35 indicates the status of three separate events: Data Available (for an input event), Data Taken (for an output event), or an Abort situation. If data is currently unavailable from I/O module 12, the CPU continues to check for an active "Data Available" signal on handshaking bus 35 on every clock cycle associated with the CPU that is executing the FAST I/O IN instruction. In other words, CPU enters a wait state until information arrives from its external source.

When handshaking bus 35 is finally activated, the CPU then instructs I/O module 12 to place the available data to be transferred onto data bus 13. At this point, the data is stored at the appropriate address location of RAM 14 indicated by the address currently stored within register 25. The start address location for data to be stored within RAM 14 is provided by register 25 along line 31. It should be understood that during the time that the CPU is waiting for bus 35 to be activated, the CPU will exit this wait state in the event that the I/O module "Abort" handshaking signal is activated for whatever reason.

Regardless of which condition resulted in the CPU proceeding to the next state, the present address stored within register 25 is passed through multiplexer 28 to incrementer/decrementer 22 along line 33. Device 22 increments the previous address to point to the next address location. This next address location then passes through multiplexer 23 to be stored in register 25, thereby replacing the previously stored RAM address. The present count from count register 26 is then passed through multiplexer 28 to be decremented by device 22. The decremented count is then stored back into count register 26.

Following these steps a check is performed for two conditions. If the count stored within register 26 is decremented to zero, or the I/O "Fast I/O Abort" handshake bus 35 is active, the fast I/O instruction will terminate and the next instruction will be fetched. If neither condition is true, data transfer continues and the CPU returns to its wait state until an active "Data Available" (or "Abort") signal appears on bus 35. Note that zero detect block 29 detects when the count stored within register 26 has decremented to zero. This indicates that the multiple byte data transfer has been completed.

Data transfer according to the Fast I/O OUT instruction takes place in the same manner as described above, except that data is transferred from RAM 14 to I/O module 12 along data bus 13, instead of from I/O module 12 to RAM 14. From there the data is exported to an external device. As the external device takes the data at either of ports 17 or 18, a "Data Taken" signal is sent on bus 35 to CPU control block 30. This initiates the increment/decrement process as described above. (Note that in the currently preferred embodiment, the communication port hardware is responsible for sending out bit synchronization, byte synchronization, CRC and end-of-message signals.)

Figure 3 is a flow chart describing the sequence of events which takes place during the execution of a Fast I/O IN instruction. At block 50, the RAM start address is first retrieved from stack 21. The count is then retrieved from the stack as indicated by block 51. The RAM address and count are placed in registers 25 and 26, respectively (see Figure 2).

At decision block 52 the CPU checks handshaking bus 35 to see whether data is currently available at the I/O modules. At this point if data is unavailable, the CPU enters a wait state until the "Data Available" or "Abort" signals are received.

Assuming that data becomes available, the data is moved from I/O modules 12 to RAM 14 along internal data bus 13. This occurrence is indicated by block 53. Once the data has been written into RAM 14, the RAM address location is incremented and the count is decremented. These events are shown taking place at blocks 54 and 55, respectively. The new address and count are then stored in registers 25 and 26 as previously explained.

At decision block 56 the count is checked to see whether it has decremented to zero (or an "Abort" signal is activated). If the answer is yes, the sequence moves to block 57, wherein the next instruction is fetched. However, if the count is not zero (i.e., additional bytes are yet to be transferred) the flow returns to decision block 52, wherein the CPU waits for the next byte to arrive.

The sequence of events for a Fast I/O OUT instruction is essentially the same as that for a Fast I/O IN instruction. These events are depicted by the flow diagram of Figure 4. Basically, blocks 60 through 67 are the same as corresponding blocks 50 through 57 of Figure 3. The only difference is at block 63, wherein data is moved from the RAM to the I/O modules, rather than from the I/O modules to RAM 14.

Whereas many alternations and modifications of the present invention will no doubt become apparent to a person of ordinary skill in the art after having read the foregoing description, it is to be understood that the particular embodiment shown and described by way of illustration is in no way intended to be considered limiting. For example, although this disclosure has shown a particular way of implementing a fast I/O instruction in the context of a plurality of intelligent cells, other implementations and/or applications are possible.

Every computer or microprocessor-based device must deal with the problem of performing I/O operations. Therefore, reference to the details of the preferred embodiment are not intended to limit the scope of the claims which themselves recite only those features regarded as essential for the invention.

CLAIMS

We claimed:

1. In a computer system which comprises at least one central processing unit (CPU) and a memory coupled to at least one input/output (I/O) module along an internal bus, an apparatus for transferring multiple bytes of data to said memory from said module across said bus comprising:

register means for storing a current address and the current count of the data to be transferred to said memory, said count representing the remaining number of bytes to be transferred across said bus;

CPU control means for successively instructing said I/O module to transfer an available byte of data onto said bus for storage into said memory at said current address, said control means issuing a control signal following each said transfer;

means responsive to said control signal for incrementing said address and decrementing said count, wherein the incremented address and decremented count are restored in said register means as said current address and said current count, respectively, prior to the transfer of the next available byte of data;

means coupled to said control means for terminating said transfer whenever said count equals zero.

2. The apparatus of Claim 1 wherein said CPU control means further comprises handshaking means for communicating the availability of data bytes from said I/O module to said CPU.

3. The apparatus of Claim 2 wherein said handshaking means further communicates in abort condition wherein in said transfer is immediately terminated and the next instruction is fetched.

4. The apparatus of Claim 1 wherein said register means comprises an address register for storing said current address and a count register for storing said current count.

5. The apparatus of Claim 4 wherein said memory comprises a random-access memory (RAM).

6. In a computer network having a multi-processor and a random-access memory (RAM) coupled to an input/output (I/O) module across an internal bus, a method of transferring multiple bytes of data to said memory from said module across said bus, said method being executable by a single processor instruction within said multi-processor and comprising the steps of:

a) programming said I/O module to initiate a data transfer from an external device;

b) storing a current RAM address into a first register and a current count, representing the remaining number of bytes to be transferred across said bus, into a second register;

c) checking said I/O module to see if a data byte is available from said external device, if said data byte is available, said multi-processor instructing said I/O module to place said data byte on said bus for storage within said RAM at said RAM address, if said data byte is unavailable said multi-processor waiting until said data byte is available;

d) incrementing said current address and storing the incremented address back in said first register;

e) decrementing said current count and storing the decremented count back in said second register;

f) checking whether said current count equals zero, transfer of data terminating whenever said count equals zero; otherwise,

g) repeating steps (c-f).

7. The method of Claim 6 wherein step (b) comprises the steps of:

pushing the start address of said RAM in the transfer of count onto a stack, said transfer count representing the total number of bytes to be transferred across said bus; and

retrieving said start address and said transfer count from said RAM for storage within said first and second registers, respectively.

8. The method of Claim 6 further comprising the step of:

halting said data transfer in the event an abort signal is received from said I/O module by said multi-processor; and

fetching the next instruction.

FIG. 1

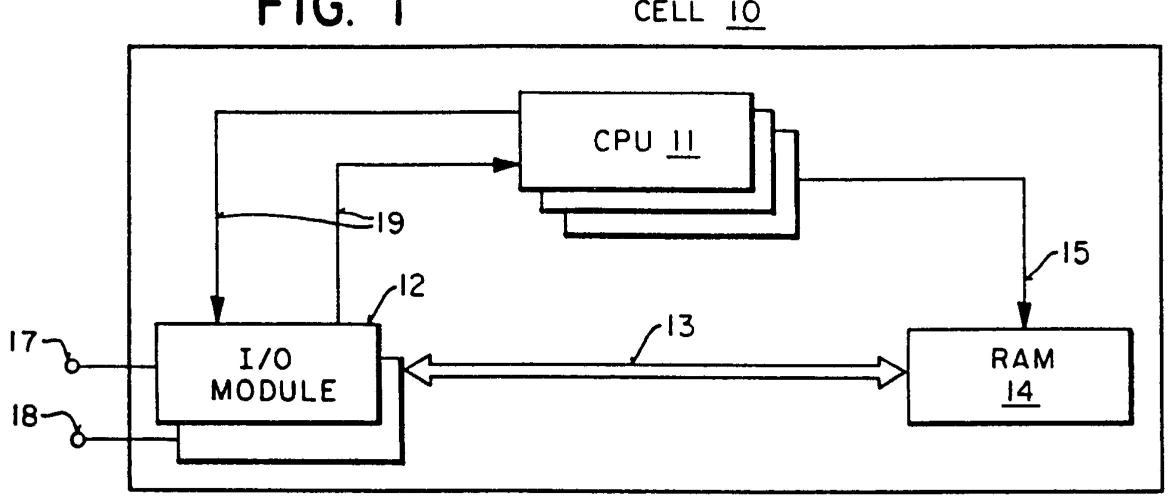

CELL 10
CPU 11
19
15
12
17
18
I/O MODULE
13
RAM 14

FIG. 2

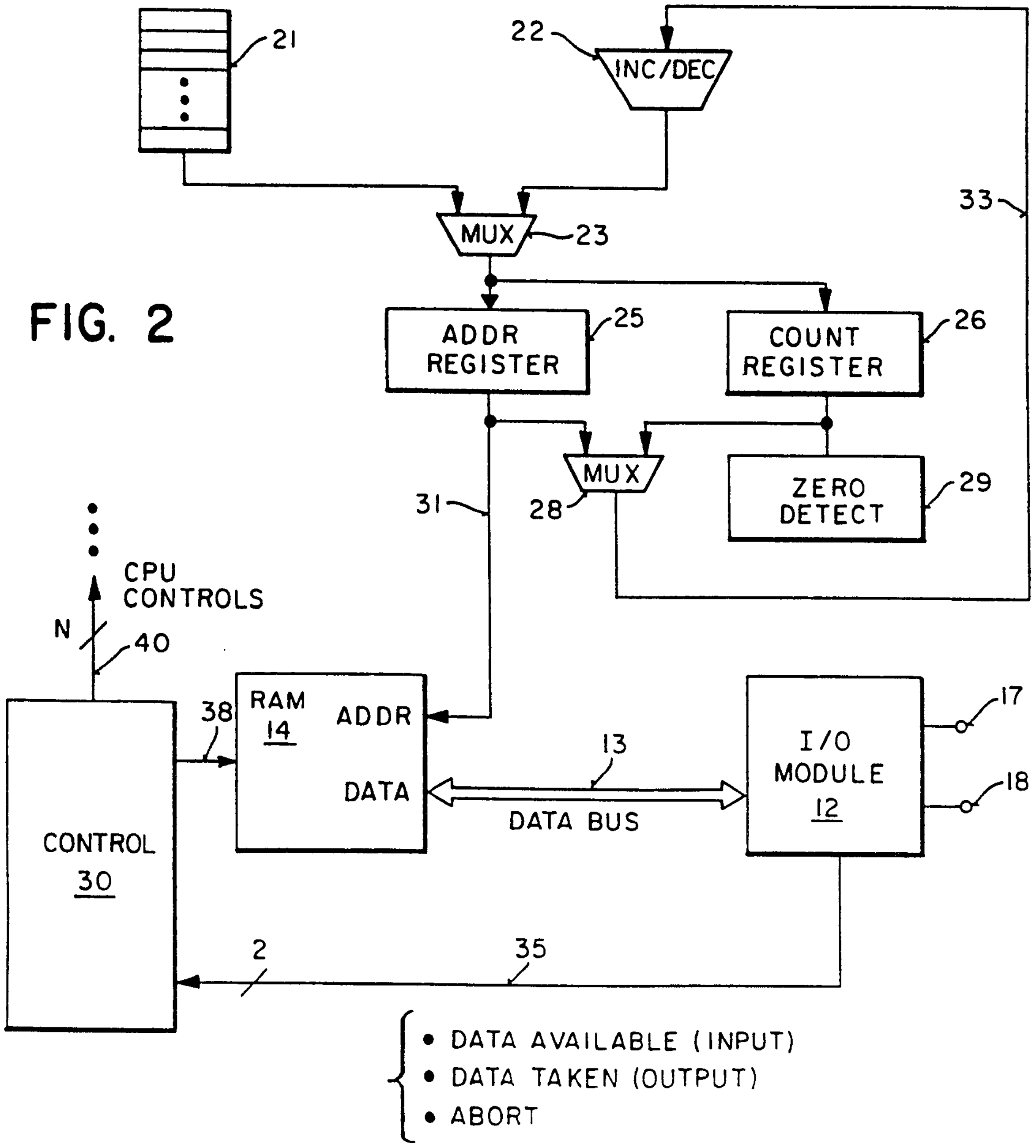

21
22
INC/DEC
33
MUX
23
ADDR REGISTER
25
COUNT REGISTER
26
MUX
28
31
ZERO DETECT
29
CPU CONTROLS
N
40
38
RAM 14
ADDR
DATA
13
DATA BUS
I/O MODULE 12
17
18
CONTROL 30
2
35
• DATA AVAILABLE (INPUT)
• DATA TAKEN (OUTPUT)
• ABORT

FIG. 3

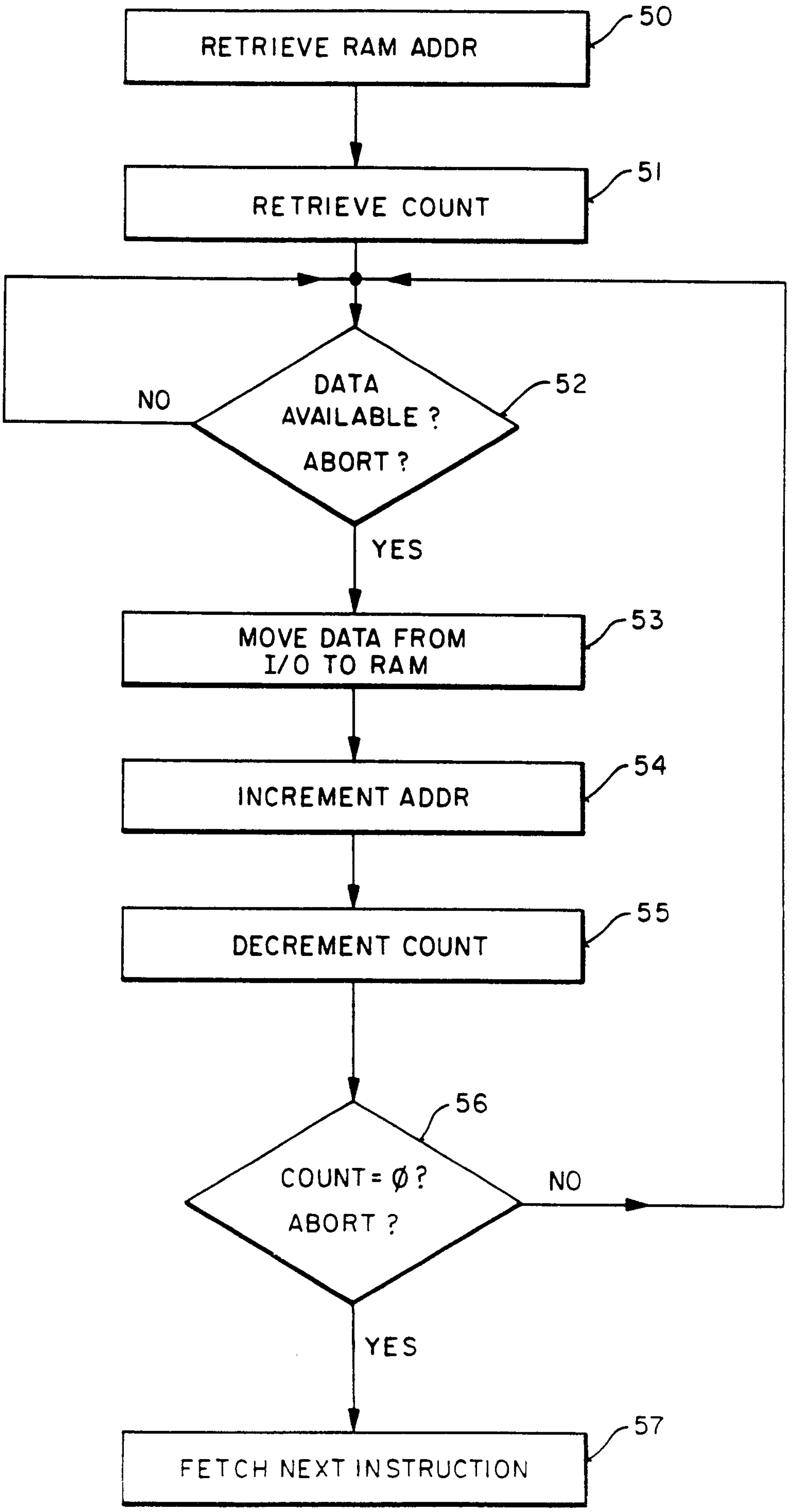
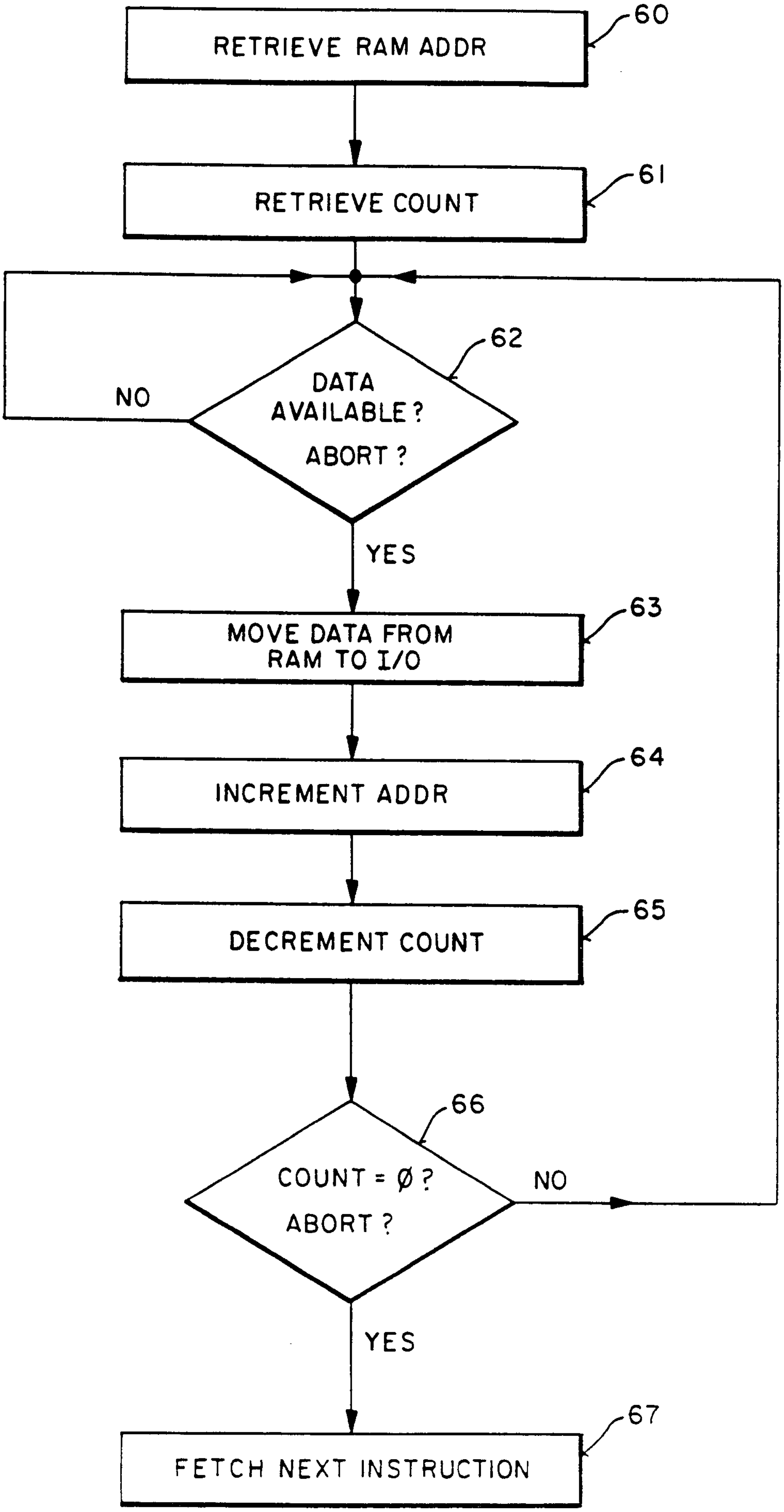

FAST I/O IN
RETRIEVE RAM ADDR
50
RETRIEVE COUNT
51
NO
DATA AVAILABLE ?
ABORT ?
52
YES
MOVE DATA FROM I/O TO RAM
53
INCREMENT ADDR
54
DECREMENT COUNT
55
56
COUNT = Ø ?
ABORT ?
NO
YES
FETCH NEXT INSTRUCTION
57

FIG. 4


FAST I/O OUT
RETRIEVE RAM ADDR
60
RETRIEVE COUNT
61
DATA AVAILABLE?
ABORT ?
62
NO
YES
MOVE DATA FROM RAM TO I/O
63
INCREMENT ADDR
64
DECREMENT COUNT
65
COUNT = Ø ?
ABORT ?
66
NO
YES
FETCH NEXT INSTRUCTION
67

INTERNATIONAL SEARCH REPORT

International Application No. PCT/US92/02292

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 13/32

II. FIELDS SEARCHED

Minimum Documentation Searched [7]

Classification System	Classification Symbols
US	395/325, 275

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	US, A 4,837,677 (Burrus, Jr. et al.) 06 June 1989 See columns 6-7.	1-8
A	US, A 4,204,250 (Getson, Jr. et al.) 20 May 1980 See entire document.	1-8
A	US, A 4,792,891 (Baba) 20 December 1988 See entire document.	1-8
A	US, A 4,792,898 (McCarthy et al.) 20 December 1988 See entire document.	1-8

* 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 11 June 1992	Date of Mailing of this International Search Report 30 JUN 1992
International Searching Authority ISA/US	Signature of Authorized Officer Robert S. Hauser