960744-5 2100

A COMPUTER SYSTEM HAVING SOFTWARE INTERRUPT (INTn) INSTRUCTIONS SELECTIVELY OPERATING IN A VIRTUAL MODE

This Sheet Has Been Replaced



FIELD OF THE INVENTION

The present invention relates generally to a computer system including a given microprocessor specifically designed to operate in a virtual operating mode that allows a software program previously written for an earlier designed single programmed microprocessor to execute in a protected, paged, multi-tasking environment under a particularly designed host operating software program. The present invention relates more particularly to an improvement in the way in which the given microprocessor and its host operating software program handle software interrupt instructions (INTn instructions) using emulation software forming part of the host program in order to emulate the way in which those instructions would have been executed by the earlier microprocessor.

BACKGROUND OF THE INVENTION

In the discussion immediately above, reference was made to a given microprocessor, to a previously written software program, to an earlier designed single program microprocessor, and to a particularly designed host operating software program. In the actual practice of the present invention, the given microprocessor specifically refers to an improvement to Intel Corporation's 80386 and 80486™ microprocessor, hereinafter merely referred to as the 386 microprocessor since the invention is equally applicable to both. The earlier designed single programmed microprocessor refers to Intel Corporation's earlier designed single program 8086 microprocessor. The previously written software program refers to any of a number of different programs including specifically DOS programs that were previously written for the 8086 microprocessor. Finally, the host operating software program refers to any suitable host or central control

operating system such as WINDOWS™ software by Microsoft or UNIX™ system software by AT&T, both written specifically for the given microprocessor so that the latter can operate in the virtual operating mode described above. While the present invention will be described hereinafter in relation to the Intel 386 (and 486™) and 8086 microprocessors, the previously written DOS programs, and a WINDOWS or UNIX operating system, it is to be understood that the present invention is not limited to those particular microprocessors, to those previously written software programs, or to those particular central operating systems. With that understanding in mind, a brief history of the 8086 and the 386 microprocessors will immediately follow. A more detailed discussion of those microprocessors as they relate specifically to the present invention will be provided in the DETAILED DESCRIPTION.

It can be stated, without reservation, that Intel Corporation's earlier 8086 microprocessor was so highly successful that there were many software programs written for it. However, given that this microprocessor was to be the first of a series of subsequently improved microprocessors, it did have its limitations. One in particular, although it was not necessarily considered a limitation at the time, was the inability of the microprocessor to act on more than one software program at a time. This, of course, meant that the program itself did not have to execute in a protected, paged, multi-tasking environment. However, as the technology evolved, Intel Corporation eventually developed its 386 microprocessor which, as stated previously, is capable of running a number of programs using a more sophisticated central operating system such as WINDOWS or UNIX. At the same time, the 386 microprocessor was designed to operate in a virtual 8086 operating mode that allowed multiple software programs previously written for the earlier 8086 microprocessor to be used and specifically to execute in a protected, paged, multi-tasking environment under the more sophisticated central operating system, even though those earlier software programs were not intended to execute in that manner. There were just too many earlier

programs to allow them to become obsolete. Therefore, whenever changes are made to successors of the 8086 microprocessor, compatibility with earlier 8086 software is always a consideration.

While compatibility between Intel's present microprocessors and the earlier written software is, indeed, important, there are instances in which this objective compromises certain other operational aspects of the overall computer system. Such is the case in the execution of software interrupt instructions (INTn instructions). As will be discussed in more detail hereinafter, the present 386 microprocessor is designed to execute all INTn instructions using emulation software forming part of the host program in order to emulate the way in which these instructions would have executed by the earlier 8086 microprocessor. While the number of executions of these particular instructions is a relatively small percentage of the total instructions executed by means of emulation, they account for a relatively large amount of execution time. One particular solution to this problem which has been proposed heretofore is to allow all of the INTn instructions to be executed by the software program itself in conjunction with the microprocessor and its host operating program without using the emulation software, that is, without trapping the INTn instructions in the host operating program's emulation software or emulator as it may be called. Applicants have found that this previous approach does save computing time generally (for the whole system) and emulation time in particular where the particular INTn instruction is simple, for example asking for the time, or even somewhat more complex, for example printing a character. However, Applicants have also discovered that for the more highly complex INTn instructions, for example reading a file, it is more efficient to allow those particular instructions to be trapped in and executed by the emulation software.

SUMMARY OF THE INVENTION

In view of the foregoing, it is a primary object of the present invention to actually reduce the time required to execute INTn instructions by a computer system of the type described above.

As will be seen hereinafter, a computer system is disclosed herein including a given microprocessor, for example the 386 microprocessor, which is specifically designed to operate in a virtual operating mode that allows a software program, for example a DOS program previously written for an earlier designed single program microprocessor, for example the 8086 microprocessor, to execute in a protected, paged, multi-tasking environment under a particularly designed host operating software program, for example WINDOWS or UNIX. This system also includes means forming part of the given microprocessor and the host operating software program for executing INTn instructions using emulation software forming part of the host program in order to emulate the way in which the instructions would have been executed by the earlier microprocessor.

In order to reduce the time it takes for the computer system generally and the emulation software in particular to execute INTn instructions during the virtual operating mode of the computer system, means are provided for causing certain ones of the INTn instructions to be executed by means of the emulation software while allowing certain others of the INTn instructions to be executed by means of the previously written program, but not by means of the emulation software. In the specific embodiment disclosed herein, the overall system is provided with a bit map including a series of bits, one for each of the INTn instructions. Each of these bits is placed in either an emulating first state or a non-emulating second state. Means are also provided for causing all of the INTn instructions associated with the emulating first state of the bit map to be executed by means of the emulation software while allowing all of the INTn instructions associated with the non-emulating second state of the bit map to be trapped in and executed by the previously written program in cooperation with the given software in the host

operating software program, but not means of the emulation software. In other words, the present invention provides for taking some but not all of the INTn instructions out of the hands of the emulator. In that way, those INTn instructions to be emulated and those that are not to be emulated can be preselected depending upon the complexity of the particular instructions so as to minimize the overall time it takes to execute all of the instructions. This is to be contrasted first with computer system including the existing 386 microprocessor which executes all INTn instructions by means of emulation and second with the previously proposed modification to the 386 system which executes none of the INTn instructions by means of emulation. The present invention provides the best of both of these approaches as regards the execution of INTn instructions.

This Sheet Has Been Replaced



BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be described in more detail hereinafter in conjunction with the drawings, wherein:

FIGURE 1 diagrammatically illustrates certain aspects of a prior art computer system including the earlier 8086 microprocessor;

FIGURE 2 is a similar view of a computer system including the earlier 386 microprocessor, specifically illustrating the way in which it executes INTn instructions by means of emulation;

FIGURE 3 is diagrammatic illustration of a computer system which executes INTn instructions in accordance with the present invention; and

FIGURE 4 is a flow diagram depicting the way in which the computer system illustrated in Figure 3 executes INTn instructions in accordance with the present invention.

This Sheet Has Been Replaced



DETAILED DESCRIPTION

Turning now to the drawings, attention is first directed to Figure 1 which, as indicated above, is a diagrammatic illustration of certain aspects of a prior art computer system including the earlier 8086 microprocessor. The overall computer system is generally indicated by the reference numeral 10. While this particular system includes a number of components not illustrated, it is specifically shown including an 8086 microprocessor 12 having an INTn register 14, a single DOS software program 16 and a number of I/O peripherals 18, 20 and 22, for example a monitor, a disk drive and a keyboard, respectively, although the present invention is not limited to these particular I/O devices. The INTn register 14 actually communicates with an address table (not shown) containing different execution addresses depending upon the particular INTn instruction in question. In system 10 there are 256 possible INTn instructions 0-255, each being characterized by a specific operand (n) for indexing into the table. Examples of DOS interrupt vectors (n) include:

MS-DOS terminate process,
MS-DOS function dispatcher,
MS-DOS terminate address
MS-DOS Ctrl-C handler address,
MS-DOS critical-error handler address,
MS-DOS absolute disk read,
MS-DOS absolute disk write,
MS-DOS terminate and stay resident,
MS-DOS idle interrupt,
MS-DOS reserved,
MS-DOS network redirector
MS-DOS reserved
MS-DOS multiplex interrupt
MS-DOS reserved

This Sheet Has Been Replaced



Having described computer system 10 thus far, attention is now directed to the way in which this system handles INTn instructions from software program 16. Assume, for example, that program 16 initiates INTn instruction INTn 21. The operand 21 is placed into the INTn register 14 which is used to index the address table which contains the beginning

address of a particular execution routine associated with that instruction. This is done through the microprocessor hardware in conjunction with single DOS program. 16

Turning now to Figure 2, attention is directed to a more advanced computer system which is generally indicated by the reference numeral 24. Among other differences between this system and system 10, system 24 includes the 386 microprocessor indicated at 26 and is capable of operating simultaneously on more than one software program including programs previously written for the 8086 microprocessor, although only one such program, for example, the DOS program 16, is shown. As a result of this ability to operate on multiple software programs simultaneously, and the desire to use 8086 software, the 386 microprocessor was designed to operate in a particular virtual operating mode, as stated previously, that is in an operating mode that allows 8086 programs, for example, DOS program 16, to execute in a protected, paged, multi-tasking environment under a host operating system. As also stated previously, one such host or central operating system is the previously recited WINDOWS or UNIX program which is diagrammatically depicted at 32 in Figure 2. This program is written to include 8086 emulation software, hereinafter sometimes referred to merely as an emulator, for executing certain sensitive interrupt related instructions including specifically software interrupt instructions (INTn instructions) from the previously written program by means of the emulation software in order to emulate the way in which the instructions would have been executed by the earlier 8086 microprocessor. In this regard, system 24 includes an INTn register 34 corresponding in function to INTn register 14. While not shown, system 24 would also include I/O peripherals such as those illustrated in Figure 1.

With computer system 24 described thus far, attention is now directed to the way in which the system handles interrupts in its virtual operating mode. Specifically, when a particular INTn instruction by one of the software programs being operated upon, for example, program 16, it is delivered to

This Sheet Has Been Replaced



register 34. From there, the software interrupt instruction is trapped in the emulator of host operating program 32 and executed by means of the emulation software in order to emulate the way in which the instructions would have been executed by the earlier 8086 microprocessor 12. All INTn instructions are executed in this way. This has been found to take a disproportionate amount of the emulator's execution time considering the rather small number of INTn instructions being executed. An approach which has previously been proposed to solve this problem has been to eliminate all trappings of INTn instructions in the emulator. Thus, according to that proposal, all 256 INTn instructions are handled outside the emulator, that is, by DOS program 16 itself in conjunction with microprocessor 26 and the host operating program.

As indicated previously, while it is quite time consuming to trap and emulate all INTn instructions, it could be even more time consuming to allow software program 16 and the other corresponding programs to execute all of the INTn instructions outside the emulator. Again, this is because some of the INTn instructions may be so complex as to require the software programs in conjunction with the microprocessor and host operating program to carry out additional subroutines not required by the emulator. As will be seen immediately below, the present invention eliminates the problem.

Turning now to Figure 3, attention is directed to a further computer system 36 and specifically one which is designed in accordance with the present invention. System 36 includes a modified 386 microprocessor 26', depicted in Figure 3 as a 386' microprocessor. This microprocessor, like the 386 microprocessor shown in Figure 2, is specifically designed to operate in a virtual operating mode that allows multiple previously written 8086 software programs including, for example, DOS program 16 to execute in a protected, paged, multi-tasking environment under a particularly designed host operating software program 32' having its own 8086 emulation software in the same manner as system 24. System 36 is also shown including corresponding INTn register 34 and, while not shown, includes I/O

This Sheet Has Been Replaced



This Sheet Has Been Replaced

peripherals in the same manner as system 10. In addition, system 36 includes a bit map 38 containing a series of 256 bits, one for each of the 256 INTn instructions. Each bit is in either an emulating first state, designated for example by a one, or a non-emulating second state designated for example by a zero. Microprocessor 26' and host operating software program 32' are designed to cause all of the INTn instructions associated with the emulating first state to be executed by means of the emulation software while allowing all of the INTn instructions associated with the non-emulating second state to be executed by means of the previously written DOS program 16 in cooperation with the microprocessor and host operating software program, but not by means of the emulation software. Each of the other software programs corresponding to DOS program 16 may include its own bit map.

Whether a particular bit within bit map 38 is in its emulating first state or its non-emulating second state is determined by the programmer and incorporated into the host operating software program. This determination is based on the complexity of each of the INTn instructions. As a general rule, those instructions that are so complex that they would take longer to execute outside the emulator, for example disk I/0, are allowed to trap within the emulator in the same manner as system 24. On the other hand, the simpler INTn instructions can be executed outside the emulator faster than they can by means of emulation and therefore are executed by means of the previously written software system in cooperation with the microprocessor and host operating program. While this is generally the criteria for setting the various bits in bit map 38, it should be apparent that the programmer could take into account other criteria. The point of the present invention is that the programmer is given this flexibility.

Having described computer system 36 and the way in which it differs from system 24 and the previous proposal for modifying system 24, it is to be understood that the present invention is not limited to Intel Corporation's 386 microprocessor and its associated components, its 486™ microprocessor, or any related microprocessor, although the present invention is especially

This Sheet Has Been Replaced

applicable to such microprocessors. Moreover, it should be apparent from the foregoing that the present invention relates specifically to the handling of INTn instructions and that only those features have been described. Obviously, each of the systems illustrated includes other components not pertinent to the present invention. Those components and any component not shown but necessary to the way in which the systems operate to handle INTn instructions would be obvious to one with ordinary skill in the art.

Having described the way in which computer system 36 handles INTn instructions and its advantages over the prior art and the previously proposed system, attention is directed to Figure 4. This figure depicts a flow chart of the way in which system 36 selectively handles INTn instructions in the manner described previously. In view of this particular flow chart and the teachings herein, one with ordinary skill in the art could readily practice the invention.

Turning now to Figure 4, a flow chart is illustrated there, depicting the way in which computer system 36 shown in Figure 3 handles INTn instructions in accordance with the present invention, while the flow chard should be self-explanatory, a brief description will be provided here. As seen there, the first decision to be made for a particular INTn instruction is to determine from the bit map if "n" has been assigned a 1 or a 0. If "n" is a 1, then the instruction is executed by means of emulation and the process ends. If "n" is a 0, then the instruction is executed by the microprocessor (including its microcode) as in the 8086 processor and the process ends.

CLAIMS

1. In a computer system including a given microprocessor which is specifically designed to operate in a virtual operating mode that allows a software program previously written for an earlier designed single program microprocessor to execute in a protected, paged, multi-tasking environment under a particularly designed host operating software program, said system further including means forming part of said given microprocessor and said host operating software program for executing software interrupt instructions (INTn instructions) using emulation software forming part of said host program in order to emulate the way in which said instructions would have been executed by said earlier microprocessor, the improvement comprising:

(a) means for causing certain ones of the INTn instructions to be executed by means of said previously written program in cooperation with said given microprocessor and said host operating software program but not means of said emulation software.

2. In a computer system including a given microprocessor which is specifically designed to operate in a virtual operating mode that allows a software program previously written for an earlier designed single program microprocessor to execute in a protected, paged, multi-tasking environment under a particularly designed host operating software program, said system further including means forming part of said given microprocessor and said host operating software program for executing software interrupt instructions (INTn instructions) using emulation software forming part of said host program in order to emulate the way in which said instructions would have been executed by said earlier microprocessor, the improvement comprising:

(a) a bit map including a series of bits, one for each of said INTn instructions, each of said bits being placed in either an emulating first state or a non-emulating second state; and

This Sheet Has Been Replaced

(b) means for causing certain ones of the INTn instructions to be executed by means of said previously written program in cooperation with said given microprocessor and said host operating software program but not means of said emulation software.

3. In a method of operating a computer system including a given microprocessor which is specifically designed to operate in a virtual operating mode that allows a software program previously written for an earlier designed single program microprocessor to execute in a protected, paged, multi-tasking environment under a particularly designed host operating software program, said system further including means forming part of said given microprocessor and said host operating software program for executing software interrupt instructions (INTn instructions) using emulation software forming part of said host program in order to emulate the way in which said instructions would have been executed by said earlier microprocessor, the improvement comprising the step of:

(a) executing certain ones of said INTn instructions by means of said emulation software and executing certain others of the INTn instructions by means of said previously written program in cooperation with said given microprocessor and said host operating software program but not by means of said emulation software.

4. In a method of operating a computer system including a given microprocessor which is specifically designed to operate in a virtual operating mode that allows a software program previously written for an earlier designed single program microprocessor to execute in a protected, paged, multi-tasking environment under a particularly designed host operating software program, said system further including means forming part of said given microprocessor and said host operating software program for executing software interrupt instructions (INTn instructions) using emulation software forming part of said host program in order to emulate the way in which said instructions would have been executed by said earlier microprocessor, the improvement comprising the steps of :

This Sheet Has Been Replaced

(a) providing a bit map including a series of bits, one for each of said INTn instructions, each of said bits being placed in either an emulating first state or a non-emulating second state; and

(b) executing all of the INTn instructions associated with said emulating first state by means of said emulation software and executing all of the INTn instructions associated with said non-emulating second state by means of said previously written program in cooperation with said given microprocessor and said host operating software program but not by means of said emulation software.

This Sheet Has Been Replaced

FIG_1

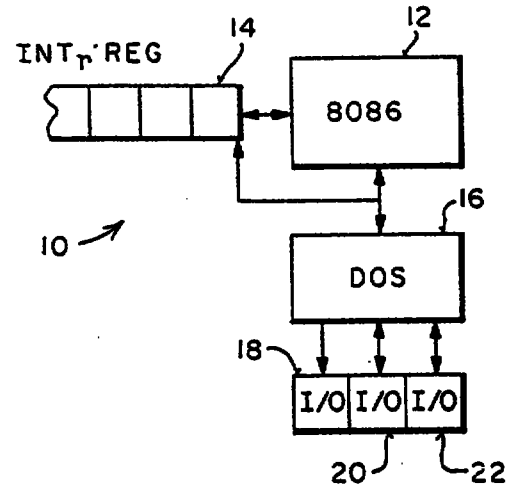


FIG_2

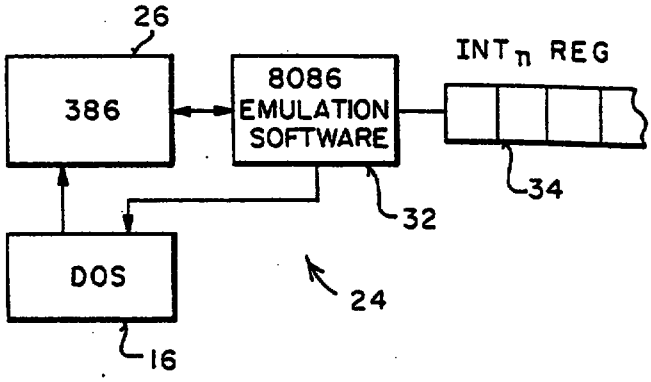


FIG_3

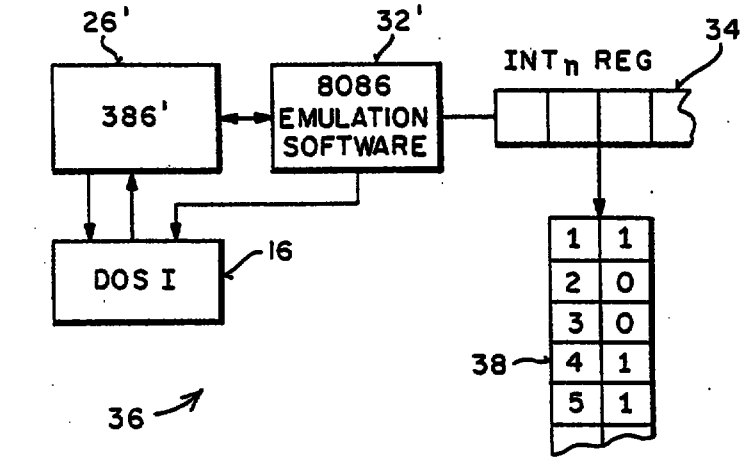
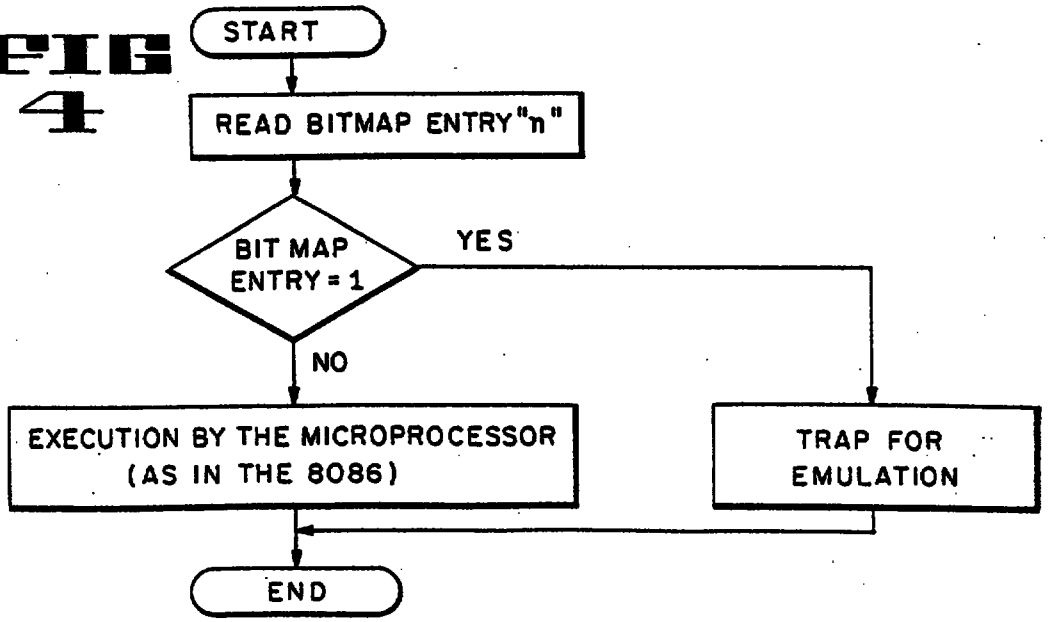


FIG 4



ABSTRACT

A COMPUTER SYSTEM HAVING SOFTWARE INTERRUPT (INTn) INSTRUCTIONS SELECTIVELY OPERATING IN A VIRTUAL MODE. (Figure 3)

A computer system is disclosed herein including a given microprocessor specifically designed to operate in a virtual operating mode that allows a software program previously written for an earlier designed single program microprocessor to execute in a protected, paged, multi-tasking environment under a particularly designed host operating software program. The system also includes means for executing software interrupt (INTn) instructions, using emulation software forming part of the host program in order to emulate the way in which these instructions would have been executed by the earlier microprocessor. As a unique improvement to this overall computer system, certain ones of the INTn instructions are executed by means of emulation software while others are executed by means of the previously written program in cooperation with the given microprocessor and its host operating software program.

This Sheet Has Been Replaced



2/17/99

A ~~COMPUTER SYSTEM HAVING SOFTWARE INTERRUPT (INTn) INSTRUCTIONS SELECTIVELY OPERATING IN A VIRTUAL MODE~~ A COMPUTER SYSTEM AND METHOD FOR EXECUTING INTERRUPT INSTRUCTIONS IN TWO OPERATING MODES

FIELD OF THE INVENTION

The present invention relates generally to a computer system including a given microprocessor specifically designed to operate in a virtual operating mode that allows a software program previously written for an earlier designed single-~~programmed~~ program microprocessor to execute in a protected, paged, multi-tasking environment under a particularly designed host operating software program. The present invention relates more particularly to an improvement in the way in which the given microprocessor and its host operating software program handle software interrupt instructions (INTn instructions) using emulation software forming part of the host program in order to emulate the way in which those instructions would have been executed by the earlier microprocessor.

BACKGROUND OF THE INVENTION

In the discussion immediately above, reference was made to a given microprocessor, to a previously written software program, to an earlier designed single program microprocessor, and to a particularly designed host operating software program. In the actual practice of the present invention, the given microprocessor specifically refers to an improvement to Intel Corporation's 80386 and 80486™ microprocessor, hereinafter merely referred to as the 386 microprocessor since the invention is equally applicable to both. The earlier designed single-~~programmed~~ program microprocessor refers to Intel Corporation's earlier designed single program 8086 microprocessor. The previously written software program refers to any of a number of different programs including specifically DOS programs that were previously written for the 8086 microprocessor. Finally, the host operating software program refers to any suitable host or central control

operating software program, but not by means of the emulation software. In other words, the present invention provides for taking some but not all of the INTn instructions out of the hands of the emulator. In that way, those INTn instructions to be emulated and those that are not to be emulated can be preselected depending upon the complexity of the particular instructions so as to minimize the overall time it takes to execute all of the instructions. This is to be contrasted first with the computer system including the existing 386 microprocessor which executes all INTn instructions by means of emulation and second with the previously proposed modification to the 386 system which executes none of the INTn instructions by means of emulation. The present invention provides the best of both of these approaches as regards the execution of INTn instructions.

REPLACEMENT SHEET
22 JUL 1998

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be described in more detail hereinafter in conjunction with the drawings, wherein:

FIGURE 1 diagrammatically illustrates certain aspects of a prior art computer system including the earlier 8086 microprocessor;

FIGURE 2 is a similar view of a computer system including the earlier 386 microprocessor, specifically illustrating the way in which it executes INTn instructions by means of emulation;

FIGURE 3 is a diagrammatic illustration of a computer system which executes INTn instructions in accordance with the present invention; and

FIGURE 4 is a flow diagram depicting the way in which the computer system illustrated in Figure 3 executes INTn instructions in accordance with the present invention.

REPLACEMENT SHEET
22 JUL 1998

DETAILED DESCRIPTION

Turning now to the drawings, attention is first directed to Figure 1 which, as indicated above, is a diagrammatic illustration of certain aspects of a prior art computer system including the earlier 8086 microprocessor. The overall computer system is generally indicated by the reference numeral 10. While this particular system includes a number of components not illustrated, it is specifically shown including an 8086 microprocessor 12 having an INTn register 14, a single DOS software program 16 and a number of I/O peripherals 18, 20 and 22, for example a monitor, a disk drive and a keyboard, respectively, although the present invention is not limited to these particular I/O devices. The INTn register 14 actually communicates with an address table (not shown) containing different execution addresses depending upon the particular INTn instruction in question. In system 10 there are 256 possible INTn instructions 0-255, each being characterized by a specific operand (n) for indexing into the table. Examples of DOS interrupt vectors (n) include:

MS-DOS terminate process,
MS-DOS function dispatcher,
MS-DOS terminate address
MS-DOS Ctrl-C handler address,
MS-DOS critical-error handler address,
MS-DOS absolute disk read,
MS-DOS absolute disk write,
MS-DOS terminate and stay resident,
MS-DOS idle interrupt,
MS-DOS reserved,
MS-DOS network redirector
MS-DOS reserved
MS-DOS multiplex interrupt
MS-DOS reserved

REPLACEMENT SHEET
22 JUL 1998

Having described computer system 10 thus far, attention is now directed to the way in which this system handles INTn instructions from software program 16. Assume, for example, that program 16 initiates INTn instruction INTn 21. The operand "21" is placed into the INTn register 14 which is used to index the address table which contains the beginning

REPLACEMENT SHEET
22 JUL 1998

address of a particular execution routine associated with that instruction. This is done through the microprocessor hardware in conjunction with the single DOS program 16.

Turning now to Figure 2, attention is directed to a more advanced computer system which is generally indicated by the reference numeral 24. Among other differences between this system and system 10, system 24 includes the 386 microprocessor indicated at 26 and is capable of operating simultaneously on more than one software program including programs previously written for the 8086 microprocessor, although only one such program, for example, the DOS program 16, is shown. As a result of this ability to operate on multiple software programs simultaneously, and the desire to use 8086 software, the 386 microprocessor was designed to operate in a particular virtual operating mode, as stated previously, that is an operating mode that allows 8086 programs, for example, DOS program 16, to execute in a protected, paged, multi-tasking environment under a host operating system. As also stated previously, one such host or central operating system is the previously recited WINDOWS or UNIX program which is diagrammatically depicted at 32 in Figure 2. This program is written to include 8086 emulation software 33, hereinafter sometimes referred to merely as an emulator 33, for executing certain sensitive interrupt related instructions including specifically software interrupt instructions (INTn instructions) from the previously written program 16 by means of the emulation software 33 in order to emulate the way in which the instructions would have been executed by the earlier 8086 microprocessor. In this regard, system 24 includes an INTn register 34 corresponding in function to INTn register 14. While not shown, system 24 would also include I/O peripherals such as those illustrated in Figure 1.

With computer system 24 described thus far, attention is now directed to the way in which the system handles interrupts in its virtual operating mode. Specifically, when a particular INTn instruction in one of the software programs is being operated upon, for example, program 16, its operand is delivered to

REPLACEMENT SHEET
22 JUL 1998

register 34. From there, the software interrupt instruction is trapped in the emulator 33 of the ~~emulator of~~ host operating program 32 and executed by means of the emulation software 33 in order to emulate the way in which the instructions would have been executed by the earlier 8086 microprocessor 12. All INTn instructions are executed in this way. This has been found to take a disproportionate amount of the emulator's execution time considering the rather small number of INTn instructions being executed. An approach which has previously been proposed to solve this problem has been to eliminate all trappings of INTn instructions in the emulator 33. Thus, according to that proposal, all 256 INTn instructions are handled outside the emulator 33, that is, by DOS program 16 itself in conjunction with the microprocessor 26 and the host operating program 32.

As indicated previously, while it is quite time consuming to trap and emulate all INTn instructions, it could be even more time consuming to allow software program 16 and the other corresponding programs to execute all of the INTn instructions outside the emulator 33. Again, this is because some of the INTn instructions may be so complex as to require the software programs in conjunction with the microprocessor 26 and the ~~microprocessor and~~ host operating program 32 to carry out additional subroutines not required by the emulator 33. As will be seen immediately below, the present invention eliminates the problem.

Turning now to Figure 3, attention is directed to a further computer system 36 and specifically one which is designed in accordance with the present invention. System 36 includes a modified 386 microprocessor 26', depicted in Figure 3 as a 386' microprocessor. This microprocessor, like the 386 microprocessor shown in Figure 2, is specifically designed to operate in a virtual operating mode that allows multiple previously written 8086 software programs including, for example, DOS program 16 to execute in a protected, paged, multi-tasking environment under a particularly designed host operating software program 32' having its own 8086 emulation software 33 in the same manner as system 24. System 36 is also shown including corresponding INTn register 34 and, while not shown, includes I/O

REPLACEMENT SHEET
22 JUL 1998

peripherals in the same manner as system 10. In addition, system 36 includes a bit map 38 containing a series of 256 bits, one for each of the 256 INTn instructions. Each bit is in either an emulating first state, designated for example by a one, or a non-emulating second state designated for example by a zero. Microprocessor 26' and host operating software program 32' are designed to cause all of the INTn instructions associated with the emulating first state to be executed by means of the emulation software 33 while allowing all of the INTn instructions associated with the non-emulating second state to be executed by means of the previously written DOS program 16 in cooperation with the microprocessor 26 and host operating software program 32 but not by means of the emulation software 33. ~~Each of the other software programs corresponding to DOS program 16 may include its own bit map~~. The bitmap 38 is in the host operating sortware program.

Whether a particular bit within bit map 38 is in its emulating first state or its non-emulating second state is determined by the programmer and incorporated into the host operating software program 32. This determination is based on the complexity of each of the INTn instructions. As a general rule, those instructions that are so complex that they would take longer to execute outside the emulator 33, for example disk I/0, are allowed to trap within the emulator 33 in the same manner as system 24. On the other hand, the simpler INTn instructions can be executed outside the emulator 33 faster than they can by means of emulation and therefore are executed by means of the previously written software system in cooperation with the microprocessor 26 and host operating program 32. While this is generally the criteria for setting the various bits in bit map 38, it should be apparent that the programmer could take into account other criteria. The point of the present invention is that the programmer is given this flexibility.

Having described computer system 36 and the way in which it differs from system 24 and the previous proposal for modifying system 24, it is to be understood that the present invention is not limited to Intel Corporation's 386 microprocessor and its associated components, its 1486 ~~486™~~ microprocessor, or any related microprocessor, although the present invention is especially

applicable to such microprocessors. Moreover, it should be apparent from the foregoing that the present invention relates specifically to the handling of INTn instructions and that only those features have been described. Obviously, each of the systems illustrated includes other components not pertinent to the present invention. Those components and any component not shown but necessary to the way in which the systems operate to handle INTn instructions would be obvious to one with ordinary skill in the art.

Having described the way in which computer system 36 handles INTn instructions and its advantages over the prior art and the previously proposed system, attention is directed to Figure 4. This figure depicts a flow chart of the way in which system 36 selectively handles INTn instructions in the manner described previously. In view of this particular flow chart and the teachings herein, one with ordinary skill in the art could readily practice the invention.

Turning now to Figure 4, a flow chart is illustrated there, depicting the way in which computer system 36 shown in Figure 3 handles INTn instructions in accordance with the present invention, while the flow chart should be self-explanatory, a brief description will be provided here. As seen there, the first decision to be made for a particular INTn instruction is to determine from the bit map if "n" has been assigned a 1 or a 0. If "n" is a 1, then the instruction is executed by means of emulation and the process ends. If "n" is a 0, then the instruction is executed by the microprocessor (including its microcode) as in the 8086 processor and the process ends.

REPLACEMENT SHEET
22 JUL 1998

Claims

1. In a method of operating a computer system including a microprocessor which operates in a virtual operating mode that allows an existing program previously written for an earlier microprocessor to execute in a protected, paged, multi-tasking environment under a host operating software program, said host operating software program having emulation software for emulating the way interrupt (INTn) instructions in said existing program would have been executed by said earlier microprocessor, an improvement comprising the steps of:

(a) providing a user programmable mode bit map resident in said host operating software program, said mode bit map including a series of mode bits corresponding to said INTn instructions, said mode bits designating at least one of an emulating first state and a non-emulating second state;

(b) if said INTn instruction designates said emulating first state, then executing said INTn instruction using said emulation software; and

(c) if said INTn instruction designates said non-emulating second state, then executing said INTn instruction without emulation.

2. A computer system comprising of:

REPLACEMENT SHEET
22 JUL 1998

a microprocessor of a first type, operable in a first mode and a second mode, said first mode capable of executing a set of instructions executable on a microprocessor of a second type; and

a user programmable bit map including a series of mode bits corresponding to said instructions, said mode bits designating whether a respective instruction of said set of instructions is emulated when executed on said microprocessor of said first type, or non-emulated when executed on said microprocessor of said first type.

3. The computer system of claim 2, wherein said set of instructions consist of interrupt instructions.

4. The computer system of claim 3, wherein said microprocessor of said second type is an earlier designed microprocessor with respect to said microprocessor of said first type.

5. The computer system of claim 4, further including a host operating software having an emulator for emulating execution of said set of

REPLACEMENT SHEET
22 JUL 1998

instructions when executed on said microprocessor of said first type, as said instructions would be executed on said microprocessor of said second type.

6. The computer system of claim 5, wherein said computer system is capable of executing in a multitasking environment.

7. The computer system of claim 6, wherein said first mode is a virtual mode, and said second mode is a protected mode.

8. A computer system comprising of:

a first type of microprocessor, operable in a first mode and a second mode, said first mode capable of executing a set of instructions written for a second type of microprocessor; and

a user programmable bit map including a series of mode bits corresponding to said instructions, said mode bits designating whether a respective instruction of said set of instructions is to be emulated when executed on said first type of microprocessor in said first mode, as said respective instruction would have been executed on said second type of microprocessor, or

REPLACEMENT SHEET
22 JUL 1998

non-emulated when executed on said first type of microprocessor in said first mode.

9. A computer system comprising of:

a first type of microprocessor, operable in a first mode and a second mode, said first mode capable of executing a set of instructions written for a second type of microprocessor;

an emulator for emulating execution of said set of instructions when executed on said first type of microprocessor, as said instructions would be executed on said second type of microprocessor; and

a user programmable bit map including a series of mode bits, each of said mode bits corresponding to at least one instruction of said set of instructions, said mode bits designating whether a corresponding instruction of said set of instructions is to be emulated when executed on said first type of microprocessor in said first mode, or executed without emulation when executed on said first type of microprocessor in said first mode.

REPLACEMENT SHEET

22 JUL 1998

ABSTRACT

~~A COMPUTER SYSTEM HAVING SOFTWARE INTERRUPT (INTn) INSTRUCTIONS SELECTIVELY OPERATING IN A VIRTUAL MODE.~~

A COMPUTER SYSTEM AND METHOD FOR EXECUTING INTERRUPT INSTRUCTIONS IN TWO OPERATING MODES

A computer system ~~is disclosed herein~~ including a given microprocessor specifically designed to operate in a virtual operating mode ~~that~~ allows a software program previously written for an earlier designed single program microprocessor to execute in a protected, paged, multi-tasking environment under a particularly designed host operating software program. The system also includes means for executing software interrupt (INTn) instructions, using emulation software forming part of the host program in order to emulate the way in which these instructions would have been executed by the earlier microprocessor. As a unique improvement to this overall computer system, certain ones of the INTn instructions are executed by means of emulation software while others are executed by means of the previously written program in cooperation with the given microprocessor and its host operating software program.

(Figure 3)

REPLACEMENT SHEET

22 JUL 1998