US 20120210155A1

(19) **United States**

(12) **Patent Application Publication** (10) Pub. No.: **US 2012/0210155 A1**

Fleischmann et al. (43) **Pub. Date: Aug. 16, 2012**

(54) **STORING CONTEXT INFORMATION PRIOR TO NOT SUPPLYING POWER TO A PROCESSOR**

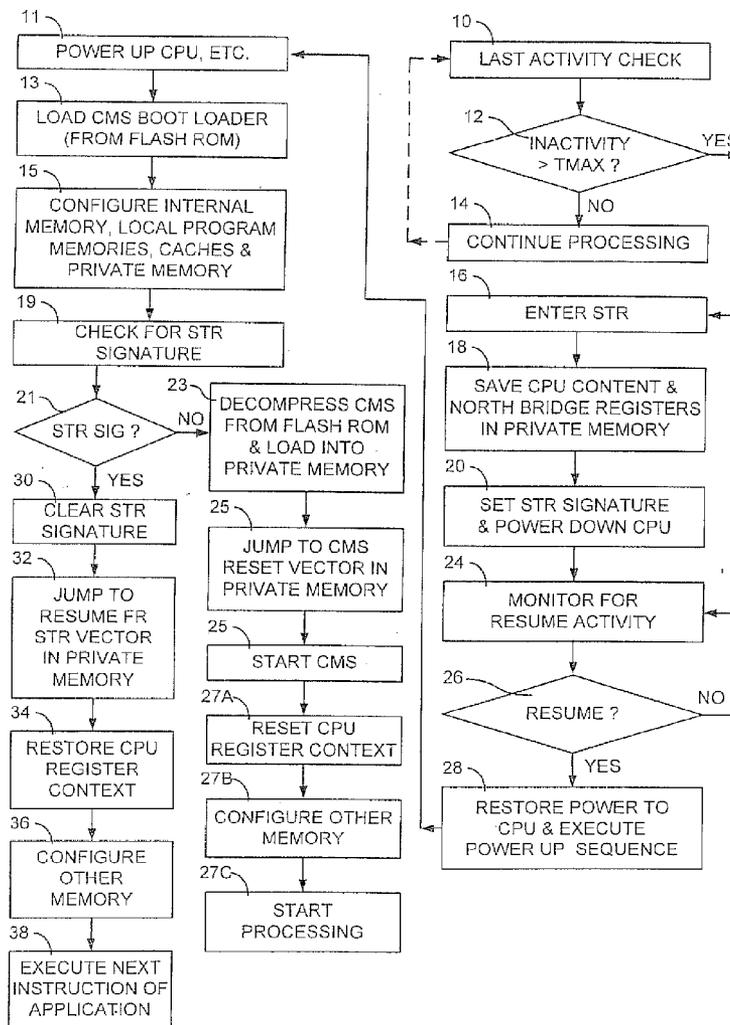(76) Inventors: **Marc Fleischmann**, Menlo Park, CA (US); **H. Peter Anvin**, San Jose, CA (US)

(57) **ABSTRACT**

A CPU (1) automatically preserves the CPU context in a computer memory (5) that remains powered-up when the CPU is powered down in sleep mode. By means of the preserved CPU context, the CPU is able to instantly and transparently resume program execution at the instruction of the program that was asserted for execution when the CPU was powered down. The CPU is permitted to power down frequently, even during execution of a program, and results in reduced average overall power consumption.

CPU

1

PRIVATE MEMORY — 5

INT. MEMORY — 6

P2

OTHER MEMORY — 3

P1   USER INPUT DEVICE — 7

P3

FIG. 1

CPU

1

PRIVATE MEMORY — 5

INT. MEM. & CACHE — 8

P2

OTHER MEMORY — 3

P1   P4   USER INPUT — 7

P3

FIG. 3

11 — POWER UP CPU, ETC.

10 — LAST ACTIVITY CHECK

13 — LOAD CMS BOOT LOADER (FROM FLASH ROM)

12 — INACTIVITY > TMAX ?   YES

NO

15 — CONFIGURE INTERNAL MEMORY, LOCAL PROGRAM MEMORIES, CACHES & PRIVATE MEMORY

14 — CONTINUE PROCESSING

19 — CHECK FOR STR SIGNATURE

16 — ENTER STR

18 — SAVE CPU CONTENT & NORTH BRIDGE REGISTERS IN PRIVATE MEMORY

21 — STR SIG ?   NO

23 — DECOMPRESS CMS FROM FLASH ROM & LOAD INTO PRIVATE MEMORY

YES

20 — SET STR SIGNATURE & POWER DOWN CPU

30 — CLEAR STR SIGNATURE

25 — JUMP TO CMS RESET VECTOR IN PRIVATE MEMORY

24 — MONITOR FOR RESUME ACTIVITY

32 — JUMP TO RESUME FR STR VECTOR IN PRIVATE MEMORY

25 — START CMS

26 — RESUME ?   NO

34 — RESTORE CPU REGISTER CONTEXT

27A — RESET CPU REGISTER CONTEXT

YES

36 — CONFIGURE OTHER MEMORY

27B — CONFIGURE OTHER MEMORY

28 — RESTORE POWER TO CPU & EXECUTE POWER UP SEQUENCE

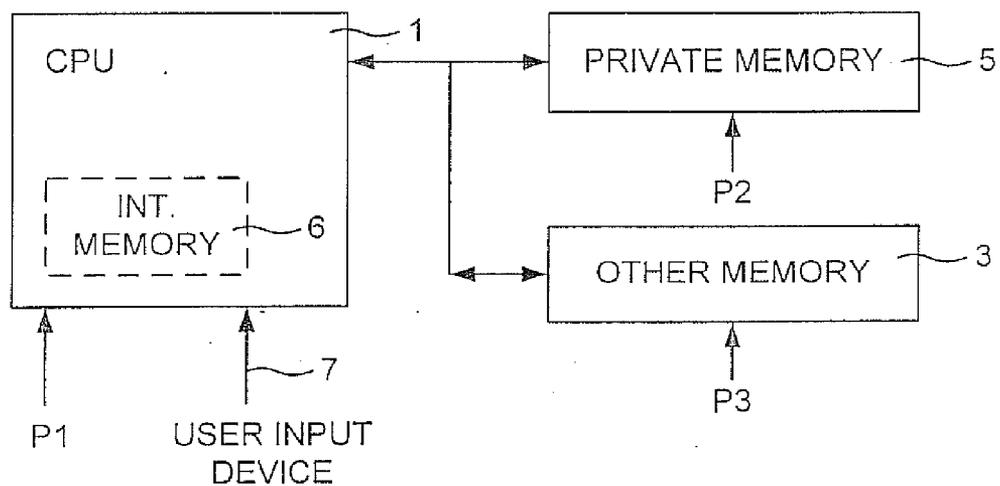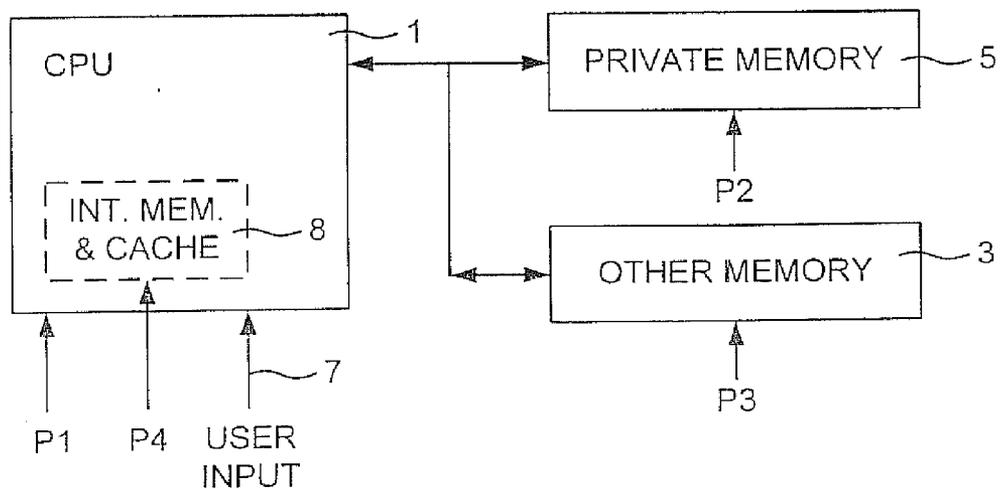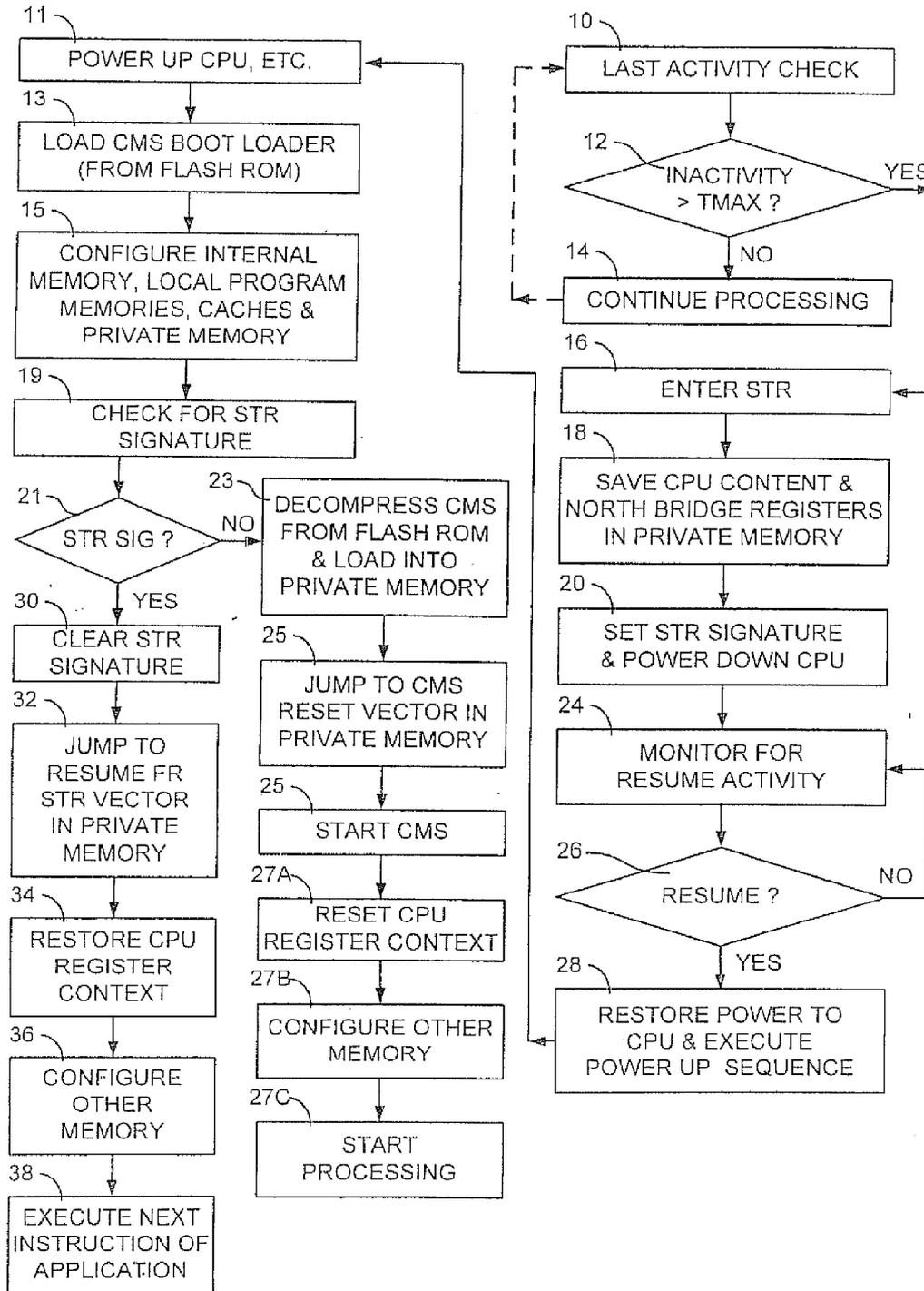38 — EXECUTE NEXT INSTRUCTION OF APPLICATION

27C — START PROCESSING

FIG. 2

STORING CONTEXT INFORMATION PRIOR TO NOT SUPPLYING POWER TO A PROCESSOR

RELATED U.S. APPLICATIONS

[0001] This application is a continuation application of application Ser. No. 12/136,679, filed Jun. 10, 2008, by Fleischmann et al., and entitled "Restoring Processor Context in Response to a Processor Power-up," which is a continuation of application Ser. No. 11/201,624, filed Aug. 10, 2005, by Fleischmann et al., and entitled "A System and Method for Saving and Restoring a Processor State Without Executing Any Instructions From a First Instruction Set," now U.S. Pat. No. 7,730,330, which in turn is a continuation application of application Ser. No. 09/595,198, filed Jun. 16, 2000, by Fleischmann et al., and entitled "System and Method for Preserving Internal Processor Context When the Processor Is Powered Down and Restoring the Internal Processor Context When Processor Is Restored," now U.S. Pat. No. 6,968,469, all of which are hereby incorporated by reference in entirety.

FIELD OF THE INVENTION

[0002] This invention relates to sleep modes for digital computers and, more particularly, to a novel method and apparatus for reducing battery power drain in portable computers during periods in which the computer is left idle for a time, even momentarily, while restoring CPU operation instantaneously once processing is to resume.

BACKGROUND

[0003] Portable digital computers rely on rechargeable DC batteries to provide the electrical power necessary for operation. When the computer is powered on for processing operation, but allowed to remain idle, the battery continues nonetheless to supply current to all the components of the computer, including the central processing unit (CPU), memory, the chipset (e.g. the Southbridge) and the display of the computer. If the user fails to turn off the computer, the battery continues to supply full current and, eventually, becomes drained of the stored electrical power. The foregoing action leads to more frequent recharging of the battery, and reduces the utility and usability of the computer system.

[0004] To reduce battery drain under such circumstances, a power management technique was previously introduced for portable computers, called the "sleep" mode. Typically, portable computers based on the INTEL X86 CPU and associated chip set, referred to as "PC's", include multiple sleep modes (e.g. states of sleep mode). The multiple sleep modes enable the portable computer, when left idle, to selectively power down the components and devices of the computer in stages, although the main power remains on. With the computer spending an increasing amount of time idling, the computer progresses through increasingly deeper and deeper stages of sleep mode (and hence, greater reductions in power consumption). One of the deepest of those modes is characterized by all of the devices, including the CPU, but excepting the main memory (RAM) and the Southbridge chip, being powered down. This latter mode is typically referred to as "Suspend to RAM" ("STR") or as "Power-on-Suspend" ("POS") or like terms. In the STR condition power consumption is dramatically reduced and offers the greatest power reduction short of that power reduction obtained by turning off every component of the computer, the latter being referred to as "suspend to disk", essentially completely shutting down the computer.

[0005] The sleep modes in the PC are defined and controlled by the operating system software, such as familiar Windows 9X, Unix, Linux and the like, in conjunction with the system BIOS of the computer. When in STR, the Southbridge portion of the chip set, which is responsible for power management of the PC, continues to monitor the keyboard and mouse (and/or resume key) of the PC for any user activity, signifying an end to the computer idle condition.

[0006] When the user later returns to perform computing and observes the computer is in a sleep mode, the user operates a "resume" key (or any key of the keyboard) or the like. That action initiates a chain of events in the computer transparent to the user, that restores full power to the CPU; and the computer recovers quickly. Return from the upper stages of the sleep mode recovers more quickly than recovery from the STR stage, the deepest stage after the Suspend to Disk stage, the latter recovery procedure being referred to as a "resume from STR".

[0007] Of particular convenience, the user may immediately resume computing at the precise location in any application program that was active in the computer at the time the computer entered the sleep mode. To reach that point from the STR stage of sleep mode, the CPU processes a number of steps of the "boot-up" routine for the computer; steps that typically occur in a manner transparent to the user. The computer is able to resume where it left off, because, prior to entering STR, the computer preserved the complete state of all software applications and of all components and devices, including the CPU, in a memory that remained powered up during the "sleep".

[0008] For the power management technique of sleep mode, the CPU and the external memory (DRAM) are independently supplied with power, that is, are located in separate power domains. In the deepest sleep mode, STR, power is removed from the CPU (and other electronic components of the computer, such as the display), while maintaining the DRAM memory and the Southbridge chip under power. The application programs and the state of those application programs (e.g. the CPU "context") is preserved by transferring the state information to the DRAM.

[0009] In processing operation, the CPU executes application programs by continuously modifying both its internal state and memory contents according to the instructions of the program. The internal CPU memory of the X86 system resides in the same power domain as the CPU. Thus, whenever the CPU is powered down, such as for an STR procedure, the internal memory is also powered down, and normally results in the loss of that CPU context. In order for the CPU of the X86 system to resume processing of an application program on Resume from STR, the processor must at that time at least "know" the state of the program on entering STR. Before entering STR, the CPU executes an instruction (of the power management software) that saves the CPU context at a well defined location in external memory, such as the DRAM memory. That context information subsumes the state of the operating system and the state of the application program. By maintaining power to the DRAM during STR, the state information of the program is preserved, and is available for use later upon a Resume from STR.

[0010] Once the resume button is pressed and is detected by the Southbridge chip, power is reapplied to the CPU, which

commences its start-up routines. The CPU processes the normal boot-up routine stored in the ROM of the BIOS chip. That boot up procedure initializes the internal registers of the CPU and flushes its caches, thereby establishing a baseline state for the CPU. The process takes a noticeable time in which to complete. However, prior to loading the operating system, such as Windows 9X, the routine checks to determine if the boot-up procedure is a "power up reset" as occurs upon initially powering up the computer, or instead is a Resume from STR. When the routine detects the latter condition, the computer "knows" that the state of the operating system software, any application program, and the corresponding CPU context already resides in the external memory (DRAM). The CPU then completes the boot-up procedure by restoring the device states, and, with a special instruction, finally restores the CPU context from the external memory. Thereafter, the CPU is able to simply proceed with executing the next application program instruction exactly where the CPU left off when entering STR.

[0011] In a stage of sleep mode that lies one stage above the STR stage, the penultimate stage (e.g. the pre-STR stage) referred to as "deep sleep", existing operating systems issue an instruction to remove the system clock from the CPU, but to maintain the CPU powered up, continuing to consume battery power. The removal of the system clock reduces power consumption also, but that is not as great a reduction as when power is removed from the CPU, such as during STR. Without clock signals being applied, the CPU is no longer able to process (as would consume additional current), but maintains system context in the associated internal registers of the CPU. That context is not lost and is not required to be saved to external memory as is the case in entering the STR stage. As an advantage, the invention powers down the CPU in all sleep modes and preserves the CPU context, saving additional power.

[0012] Accordingly, an object of the invention is to reduce the power consumption of a computer during periods in which the computer is idle, providing a more effective sleep mode.

[0013] Another object, of the invention is to promote the pre-STR stage of sleep mode in existing power management systems to the STR stage, creating an "Instant STR", and reduce the time required by the computer system to return from that stage, ideally providing a Resume from STR that appears instantaneous.

[0014] And, a related object of the invention is to replace on-the-fly a CPU context maintaining sleep mode of existing computer systems that is governed by the operating system with a substitute sleep made that affords a lower power consumption and remains transparent to the software.

SUMMARY OF THE INVENTION

[0015] In accordance with the foregoing objects and advantages, the present invention provides for saving the internal context of the CPU (which includes the state of the application software) and powering down the CPU not only for an STR condition but for any pre-STR stage of sleep mode, and for restoring the state, including such internal context, on the Resume from STR condition, whereby the processor system is quickly restored for operation ready to execute the next instruction of the application program. With the invention, any stage of sleep mode can be promoted to an STR condition to completely power down the CPU in a manner that is transparent to the operating system.

[0016] When applied to directly to existing X86 applications and operating systems, the present invention intercepts (or converts) an instruction that calls for an X86 type computer system to enter a CPU context-maintaining sleep mode, a pre-STR stage of sleep mode, and "promotes" or converts that instruction into an instruction to enter the STR stage, and thereafter perform all of the actions described for entering STR. Whereas the X86 operating system of the CPU may "think" that the context of the application program has been saved in one way, the present system actually saves that program context and the CPU context in a different manner, transparent to the X86 operating system. When Resuming from STR, the context is restored, and the X86 operating system is none the wiser.

[0017] In accordance with a more specific aspect, the present invention provides a private memory within the processor system, a memory external to the CPU, that is accessible only to the CPU and is powered independently of the CPU to store the CPU context when the CPU is powered down in a sleep mode. The private memory may constitute a logical portion of the external memory, DRAM, in the X86 system, or may be physically separate therefrom or may constitute a logical partition of the CPU internal memory, such as a cache or subset of such cache. Like the existing DRAM, a key characteristic of the foregoing memory is that such memory is independently powered from the CPU core. In the Transmeta processing system, hereafter noted, that memory may be commandeered to serve as private memory for code morphing software (e.g. "CMS memory"). The power down routine for sleep mode includes reading the internal CPU context from the internal memory of the CPU, caches and registers, and writing that context within the private memory. When powered up, either initially or on return from SIR, the processor determines whether to perform a normal power up operation, called a power on reset, or a context-restoring operation.

[0018] On entering into the Instant STR condition, before concomitant powering down of the CPU of the computer, the CPU preserves the entire CPU context in the independently powered private external memory of the CPU. So long as the private memory remains powered up, the CPU context is preserved despite the fact that the CPU is powered down. Upon Resume from Instant STR to resume processing, power is restored to the CPU. During the boot-up process, the CPU detects that the condition is a resume from Instant STR and, in response, restores its entire internal state from the preserved CPU context located in the private memory. Thereafter the CPU may perform an internal resume to the instruction of the application program that follows the last completed instruction of that program.

[0019] As an advantage, the CPU is able to automatically and transparently recover from a sleep mode that powers down the CPU and continue execution of any arbitrary computer program after lapses of arbitrary duration in the sleep mode without any assistance being required of the operating system, BIOS or the like.

[0020] The present invention is particularly adaptable to incorporation within and enhances a newly disclosed processor system that is found to already contain a private memory that is powered separately from the processor, but is used to store, among other things, only a part of the internal context of the CPU. That processor system is the very long instruction word (VLIW) processor described, for one, in U.S. Pat. No. 6,031,992, granted Feb. 29, 2000, entitled Combining Hard-

ware and Software to provide an Improved Microprocessor, assigned to Transmeta Corporation, (referred to as the 992 Transmeta patent), which is incorporated herein by reference.

[0021] The processor of the 992 Transmeta patent is formed by a combination of a hardware processing portion (called a "morph host"), and an emulating software portion, referred to therein as code morphing software. Among other things, the code morphing software carries out a significant portion of the functions of processors in software, reducing the hardware required for processing and the greater electrical power that such hardware requires. The Transmeta processor serves as the host processor system capable of executing software programs (the "target application") designed with an instruction set intended to run on a processor system of different design, the target processor, one that contains an instruction set unique to the target processor, but foreign to the host system, such as the X86 type processor.

[0022] That processor translates instructions that comprise a target software program "on-the-fly" into instructions of the host instruction set (the host instructions); and then executes the latter instructions. In so doing, the morph host processor executes the code morphing software which, functioning as either a software "interpreter" or "translator" of target application programs into host processor instructions, which are able to accomplish the purpose of the original application software. For translation, the code morphing software defines a translator, which converts the target instructions to host instructions, and a translation process that translates the target application programs dynamically and optimizes the latter host instructions to provide an optimized translation.

[0023] The foregoing Transmeta processing system, hereafter referred to as the Transmeta processor, necessarily keeps track of its state or CPU context during processing operation. It is found that the Transmeta processor places a considerable portion of its CPU context on memory external of the CPU, such as DRAM. Conceptually, that external memory is herein referred to as "private" CPU memory, because that memory is not visible to (and cannot be accessed by) external application programs; and only the code morphing software of the CPU is able to access the private memory. Further the CPU private memory resides in a different power domain than the CPU, and, therefore, may be powered up or clown independently of the CPU. Consequently, the private memory is able to maintain a portion of the internal state of the CPU when the CPU is powered clown.

[0024] Other portions of the CPU context are saved in memory (caches) internal to the CPU, apart from the private memory. Normally that portion of the CPU context (information) would disappear whenever the Transmeta processor is powered down. In accordance with the present invention, the entire CPU context of the Transmeta processor is stored in the private memory in the Instant STR process. Accordingly, a further object of the invention is to provide a portable computer constructed in accordance with the Transmeta processing system with an improved power saving sleep mode.

[0025] When powered up, either initially or on return from Instant STR, the processor determines whether it is to perform a normal power up operation (a power on reset) or a context-restoring operation. Traditional processors typically employ external circuitry to make that decision. The Transmeta processor deposits the corresponding persistent information in the private memory. The presence of that information in the private memory is called an "STR signature".

[0026] As an advantage, the Transmeta CPU is able to automatically and transparently recover from a sleep mode in which the CPU is powered down and continue execution of any arbitrary application program, including target applications, after time lapses of arbitrary duration in the sleep mode without any assistance required of the application program. Upon resume from such a sleep mode, the Transmeta processor is able to continue with execution of a target application program. The Transmeta processor is capable of suspending arbitrary X86 computer programs to private memory (e.g. CMS memory) and automatically and transparently resume executing the next instruction of the program without any assistance of the target application.

[0027] The foregoing and additional objects and advantages of the invention together with the structure characteristic thereof, which was only briefly summarized in the foregoing passages, will become more apparent to those skilled in the art upon reading the detailed description of a preferred embodiment of the invention, which follows in this specification, taken together with the illustrations thereof presented in the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0028] In the drawings:
[0029] FIG. 1 is a high-level block diagram of a digital processor which incorporates the invention: and
[0030] FIG. 2 is a flow chart defining the operation of the processor of FIG. 1.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0031] The invention is preferably implemented within the improved 992 Transmeta processor system earlier described, which implements software-enhanced execution (eg. dynamic translation), and is best explained in connection with that processor. The principal elements of that processing system as modified to incorporate the invention are symbolically illustrated in FIG. 1, to which reference is made. The processing system contains a computer processing unit ("CPU") 1, a memory 3 for the X86 program, the target machine code that the processor is to translate, its data, and a memory 5 with the private run time support for the computer, the latter memory herein referred to as private memory. Memory 3 may encompass the entire memory hierarchy, including disk, RAM, 2nd level cache and 1st level cache. ROM memory, which is used by the processor, is considered as being separate from the processor, and for purposes of this description is not considered as part of the memory hierarchy of the processor. The foregoing illustration also excludes the memory that is regarded as internal to the CPU ("internal" memory).

[0032] The foregoing architecture is recognized as different from the traditional processor systems, such as the X86 system, which does not contain memory 5. As represented by the power arrows P1, the private memory 5 resides in a separate power domain than CPU 1 (and memory 3). Hence, electrical power to each component of the system may be separately and independently controlled, more specifically, CPU 1 may be powered down, while private memory 5 remains powered.

[0033] During processing operation, the Transmeta CPU 1 keeps track of its state or context. Since the Transmeta computer "virtualizes" the target processor, such as the X86 processor system, the computer must also keep track of the state

4

of the X86 processor system, possibly including part of the chipset and the target (X86) application program for that processor, which is being translated and executed by the code morphing software. Hence, the context of the Transmeta processor constitutes a super-set of and subsumes the state of the X86 processor system.

[0034] The context of the Transmeta processing system is reflected in the contents of the general purpose registers, including core special registers ("CSR's"), peripheral special registers ("PSR's"), and the Northbridge registers, the D-cache, the I-cache, local program memory ("LPM"), and local data memory ("LDM"). Some of that context is stored in private memory **5**. However, the internal context of CPU **1** is stored in internal memory, represented in the clotted block **6** within CPU **1**.

[0035] The Transmeta computer includes a start up procedure to initialize and configure ("set up") the computer and begin processing, a procedure defined by software or in firmware, such as ROM code, which is included. That start-up procedure ("boot-up") is initiated once power to the computer is switched on to apply power (P1, P2, and P3) to both the CPU **1** and memories **3** and **5**, a "boot-up" procedure referred to as "power-on reset".

[0036] The processor commences the start up procedure, as represented by operation **11** in FIG. **2** to which reference is made. Both the code morphing software and the CMS boot loader software of the Transmeta processing system are stored in compressed form in flash ROM (not illustrated) of the computer.

[0037] The CPU locates the CMS boot loader routine in the flash ROM and installs that routine in internal memory **6**, as represented at block **13**. As represented by operation **15**, the CMS boot loader continues with configuring the cache, the private memory **5**, and, possibly, the other memory.

[0038] At this juncture, and in accordance with the invention, the boot loader looks for an "STR signature", operation **19**, which is later described herein in greater detail in connection with a Resume from STR condition. However, since the initial power on reset of the computer is presently being described, the STR signature is not present, and the decision at decisional block **21** is negative. A discussion of the operation that occurs when the STR signature is found is reserved to a later point in this description. The normal power on reset process continues, represented at operation **23**, with the boot loader program requiring CPU **1** to locate the compressed CMS in the flash ROM, decompress the CMS and load that software in the private memory **5** where stored for use in processing. Depending upon the design of the computer, the program may also set up a debugging environment for the processor, prior to decompressing the code morphing software.

[0039] Once stored in private memory **5**, the boot loader routine "jumps" into the code morphing software at its "reset" vector in private memory **5**, as represented at block **25** and begins executing (e.g. starts) the CMS. On starting, the CMS resets the CPU registers and establishes the CPU context, as represented by block **27A**, configures other memory, block **27B**, and then commences normal processing, the details of which are not relevant or material to the present invention.

[0040] Continuing with right side of FIG. **2**, during normal processor operation, which in accordance with the present invention includes the sleep mode, the power management sub-routine of the BIOS or operating system periodically checks for activity on the processor keyboard and other user input devices, and, as represented by block **10**, checks to determine if instructions of an application program are being executed by the processor. The routine also records and stores the time of the last such activity and compares that time to the time of the present check to determine if the interval is greater than a predetermined time, Tmax, represented by decisional block **12**. If not, the determination is negative and normal processing activity of the processor continues, as represented at block **14**, including periodically repeating the activity check operation indicated by the dotted line to block **10**.

[0041] Should the period of inactivity continue until the comparison determination results in an interval of greater than Tmax, a flag is set, represented by an affirmative decision at block **12**. On finding that flag, the CPU program branches from the main routine to retrieve and execute the Instant STR routine, as represented at block **16**

[0042] An initial step in the Instant STR routine is to retrieve the part of the CPU context and Northbridge registers that is stored in the CPU's internal memory, and save that context in private memory **5**. As represented at block **18**, the foregoing state information is stored in the private memory along with the normal X86 execution state information, which the Transmeta processor normally saves in that memory.

[0043] Thereafter, the processor initiates the chipset to shut down power to CPU **1** (and other components of the processor), excepting the private memory **5**, after a defined short time interval and sets the STR signature, as represented by operation **20**. The chipset withdraws the power P1 to the CPU, which ceases operation at this point. Any information in the internal memories of the CPU is disappears. Since private memory **5** remains powered up, that is, P2 (FIG. **1**) continues, the entire context information is preserved. At this stage the CMS of the Transmeta processor is retained private memory **5**.

[0044] While shutting down (i.e. before removing the power), CMS detects the Instant STR event and sets a corresponding flag (e.g. the "STR signature") in non-volatile memory. This enables the code morphing software to detect the Instant STR condition upon a power-on-reset, as later herein described. The foregoing flag may be implemented by writing the STR signature into private memory **5** Concurrently additional hardware, such as the Southbridge chip, serves to monitor the user input device **7** (FIG. **1**) represented at block **24**. If a negative determination is made, the checking continues as indicated at decisional block **26**. If, however, any user activity is detected, such as the user's manipulation of any user input device **7**, such as keyboard or mouse or the like, then an affirmative determination is made at decisional operation **26** and the computer initiates the operation, represented at block **28** to restore power to the CPU and execute the power up sequence. The latter operation returns the flow to operational block **11**, which was earlier described.

[0045] The sequence of operation proceeds through the operations represented by operational blocks **13**, **15** and **19**, earlier described and not here repeated. When the boot loader checks for an STR signature this time as represented at block **19**, an affirmative determination is made at decisional block **21**. The processor clears the STR signature, **30**, skips decompressing CMS as the software resides in the private memory, and, as represented in block **32** directly jumps into the CMS "resume from STR" routine vector in private memory **5**. The CMS commences by restoring the internal CPU context and Northbridge registers from the private memory into the

appropriate registers of the CPU as represented at operation **34** and configures other memory **36**. With internal is CPU context restored, all memories configured, and program state present in the private memory, the processor is restored to the state existing at the time of initiation of the sleep mode procedure at which juncture the processor may then retrieve the next instruction of the X86 target application for execution, as represented at block **38**. As one appreciates, the foregoing method completes rapidly.

[0046] An alternative embodiment of the sleep mode invention would employ faster memory still, but would require modification to the structure of the Transmeta computer. Specifically, where it is practicable to modify the integrated circuit chip that contains the processor of the Transmeta computer so as to permit the on-chip caches to be powered up or down separately from the remaining elements of the processor, then it is possible to retain the internal context of the CPU within the processor during the sleep mode, avoiding the necessity for transferring the internal context to the private memory and back again during resume from the Instant STR procedure. Such a configuration is illustrated in FIG. **3**, in which the internal memory and cache **8** are powered by power domain P**4**, separate from the power domain P**1** of the other CPU elements. In as much as the memory employed for cache application is typically the fastest memory available, and the location is physically closer to the principal elements of the processor, propagation time is reduced. Further, the processing time to transfer the information is eliminated. In this embodiment, the memory serves both the function of cache and of the private memory found in the initial embodiment, earlier described.

[0047] The foregoing invention may be incorporated within target processing systems that currently employ a sleep mode, such as the X86 system, but requires some modification to the target system's operating software. Since a memory of the X86 system is already powered separately from the CPU of the processing system and for sleep mode operation that memory is already used to store the state of the operating system and the application program, an extra portion of that external memory may be used to also store the internal context of the X86 CPU.

[0048] Assume two levels of "sleep" for the X86 type computer: A first level in which the central processing unit (CPU) of the processor maintains the CPU context and a second level, more deep than the first, in which the context of the CPU is not maintained. In the first level, the Southbridge circuit of the processor shuts off (sic "kills") the input clocks to the CPU, but maintains power to the CPU. The BIOS and operating system ("OS") of the computer typically uses the first level or state a good deal of the time, because the state is low-overhead. In the second level, the Southbridge circuit kills the CPU input clocks, as before, and then also removes electrical power to the CPU and other devices of the processor. As a result of the removal of electrical power, the context information of the CPU state is lost. The BIOS and OS of the computer uses this second state less frequently than the first because the state is high overhead.

[0049] The present invention permits transparent power cycling of the CPU, turning the CPU off and then turning the CPU on, repeatedly at will, with low overhead (e.g. virtually maintaining the CPU context). Two embodiments of the invention are thus possible. The invention may replace the STR process found in the X86 processor system. In such an embodiment, the resume process restores the processor more

quickly than the existing STR procedure although no savings is made in power consumption.

[0050] In a preferred embodiment, the invention replaces the CPU context-maintaining sleep state of the X86 system (e.g. "deep sleep" which is the deepest CPU context maintaining sleep mode) and the invention "tricks" the BIOS and/or OS to "think" that the CPU is placed in the sleep mode, deep sleep, when, in fact, once the BIOS and OS initiates the "deep sleep" action, all electrical power is removed from the CPU and the CPU enters the Instant STR condition. The advantage is that less power is consumed over time than when the CPU is placed in sleep mode. And with the benefit of the very rapid resume from STR action available with the invention the computer user never knows that the CPU had been turned off.

[0051] The foregoing second embodiment is possible of accomplishment in two different species. In the first specie of the preferred embodiment the code morphing software of the Transmeta processor intercepts the command issued by the operating system, such as Windows 9X, to shut down the CPU clock input to the CPU. Then the code morphing software issues a command to power down the CPU completely, that is, disable the CPU clock input and also remove electrical power from the CPU. Thus while the operating system "thinks" that the CPU remains powered, in fact, the CPU is completely powered down. In that sense the action is transparent to the X86 OS software and to the BIOS. In existing computers the context of the CPU is lost in this power saving mode. In the present invention the CPU context is saved in private memory **5**.

[0052] The second species of the preferred embodiment requires the BIOS and/or the OS software to be modified to explicitly invoke an Instant STR operation. It may not be possible to accomplish this in computers that use a "closed" operating system, such as WINDOWS 9X, that can only be modified by the copyright owner, but can be readily accomplished in "open" operating systems, such as the familiar LINUX operating system. In new generations of BIOS and OS software, such a command may be easily incorporated, if timely suggested to the designers of such software.

[0053] By powering down the CPU additional power savings may be achieved when the computer is left idle. In the case of present CPU's operating at 600 MHz, the is power saving is about sixty milliwatts. That savings would increase for computers in the future that contain future versions of CPU's containing process technology advances that typically allow higher operating frequencies, but result in greater power consumption in sleep mode.

[0054] It is believed that the foregoing description of the preferred embodiments of the invention is sufficient in detail to enable one skilled in the art to make and use the invention. However, it is expressly understood that the detail of the elements presented for the foregoing purpose is not intended to limit the scope of the invention, in as much as equivalents to those elements and other modifications thereof, all of which come within the scope of the invention, will become apparent to those skilled in the art upon reading this specification. Thus, the invention is to be broadly construed within the full scope of the appended claims.

1-14. (canceled)

15. A computing system, comprising:

a processing device;

a first memory configured to have data stored in the first memory accessible by the processing device and con-

figured to not have the data stored in the first memory accessible by other devices included in the computing system; and

a second memory configured to be accessible to the processing device and the other devices included in the computing system;

wherein the computing system is configured to allow power to be supplied to the first memory during a time at which power is not supplied to the processing device and the second memory and wherein in response to the processing device detecting a command to inactivate a clock signal supplied to the processing device, the processing device is operable to store context information in the first memory and the computing system is configured to not supply the power to the processing device after the context information is stored in the first memory and not supply the power to the second memory.

16. The computing system of claim 15, configured to restore the context information from the first memory in response to the computing system resupplying the power to the processing device.

17. The computing system of claim 16, wherein the context information is restored from the first memory to the processing device's internal memory.

18. The computing system of claim 17, wherein the processing device's internal memory and the processing device are in different power domains.

19. The computing system of claim 15, wherein the first memory and the processing device are in different power domains.

20. The computing system of claim 15, configured to set a flag in response to not supplying the power to the processing device to indicate the context information is stored in the first memory, wherein the flag is checked in response to the computing system resupplying the power to the processing device.

21. The computing system of claim 15, wherein the command is in response to detecting a prescribed period of inactivity.

22. The computing system of claim 15, configured to monitor a user input device for activity and to resupply the power to the processing device in response to detecting the activity.

23. The computing system of claim 15, operable for executing a program comprising a first instruction and a next instruction in an instruction set; wherein operations performed by the processing device and the computing system after detecting the command are performed after the first instruction is executed and before the next instruction is executed, wherein the next instruction is executed after the power is resupplied to the processing device.

24. A method, comprising:

supplying power to a processing device included in a computing system;

supplying power to a first memory, wherein the first memory is configured to have data stored in the first memory accessible by the processing device and con-

figured to not have the data stored in the first memory accessible by other devices included in the computing system;

supplying power to a second memory, wherein the second memory is configured to be accessible to the processing device and the other devices included in the computing system;

detecting a command to inactivate a clock signal supplied to the processing device;

in response to the detecting the command, storing context information in the first memory using the processing device;

stopping the supplying of the power to the processing device using the computing system in response to completion of the storing the context information in the first memory; and

stopping the supplying of the power to the second memory.

25. The method of claim 24, further comprising restoring the context information from the first memory in response to the computing system resupplying the power to the processing device.

26. The method of claim 25, wherein the context information is restored from the first memory to the processing device's internal memory.

27. The method of claim 26, wherein the processing device's internal memory and the processing device are in different power domains.

28. The method of claim 24, wherein the first memory and the processing device are in different power domains.

29. The method of claim 24, further comprising:

setting a flag in response to not supplying the power to the processing device to indicate the context information is stored in the first memory; and

checking the flag in response to the computing system resupplying the power to the processing device.

30. The method of claim 24, wherein the command is in response to detecting a prescribed period of inactivity.

31. The method of claim 24, further comprising:

monitoring a user input device for activity; and

resupplying the power to the processing device in response to detecting the activity.

32. The method of claim 24, further comprising:

executing a program comprising a first instruction and a next instruction in an instruction set;

after executing the first instruction and before executing the next instruction, detecting the command, storing the context information, and stopping the supplying of the power to the processing device and to the second memory;

restoring the context information from the first memory in response to resupplying the power to the processing device and before executing the next instruction; and

executing the next instruction after the power is resupplied to the processing device.

* * * * *