METHOD AND APPARATUS FOR SWITCHING CONTEXT OF STATE ELEMENTS IN A MICROPROCESSOR.

1. Field of the Invention

The present invention relates to method and apparatus for managing the memory in a computer. More specifically, the present invention relates to method and apparatus for switching the context of state elements in a very fast processor whenever there is a cache miss.

2. Art Background

It is quite common for a fast central processor unit to have a cache memory in addition to a main computer memory. The cache memory is smaller but much faster than the main computer memory, is placed between the the processor and the main memory. During the execution of a software program, the cache memory stores the most frequently utilized instructions and data. Whenever the processor needs to access information from memory, the processor examines the cache first before accessing the main computer memory. A cache miss occurs if the processor cannot find the instruction or data in the cache memory and is required to access the slower main memory. Thus, the cache memory reduces the average memory access time of a processor.

An instruction stream is a sequence of instructions executed by the processor to accomplish a given process, such as add or divide. To date, a processor does either one of the following two things when it encounters a cache miss in an instruction stream: (1) it stays idle until the instruction or data access to the main memory completes, or (2) it executes other instructions in the stream out of order. These two approaches are acceptable as long as they produce no substantial increase in the physical size (real estate) of the processor, and the penalty for cache misses does not overwhelm the average instruction cycle of the processor. Cache memory is typically 32K to 1 M byte in size, and hence does not occupy significant space on a processor chip or a processor board.

However, the faster the processors become , the heavier the penalty for cache misses. The penalty for cache misses refers to the length of time a processor takes to retrieve needed information from the main memory upon occurrence of a cache miss. In a typical 40 MHz microprocessor capable of executing 40 million instructions per second (MIPS), the penalty for every cache miss is twenty (20) clock cycles. Assuming a 1% cache miss rate for the cache memory and further assuming one clock cycle per instruction for the very fast processor, the average number of clock cycles per instruction of these processors would only be .01 (20) + .99 (1) = 1.19 instead of 1.0 because of cache miss penalty. As such, the processors merely deliver 40/1.19 = 33.6 MIPS.

It is quite common to pipeline the instruction and memory operations of fast processors. A pipeline refers to a processor's ability to perform multiple tasks concurrently in the same clock cycle. Just as cache misses slow down the raw processing speed of fast processors, they also create bottlenecks in the pipelines of the processors.

The adverse effects of the cache miss penalty and pipeline bottlenecks on the speed of very fast processors render secondary the previous concern of minimizing the physical size of a processor. The trade-off is inevitable in view of the dramatic increase in speed of processors and the popularity of pipeline processing.

As will be described more fully below, the present invention provides method and apparatus for switching the context of state elements in a very fast processor upon occurrence of a cache miss. The present invention saves the state of a first process upon a cache miss and permits the processor to begin executing a second process within one clock cycle. Should the second process encounters another cache miss, the processor may return within one clock cycle to finish executing the first process if the necessary data had been retrieved from the main memory. Otherwise, the processor may begin executing a third

process. It is understood that the number of processes who's states are duplicated may easily be a large number n.

SUMMARY OF THE INVENTION

An apparatus and method are disclosed for switching the context of state elements of a very fast processor within a clock cycle when a cache miss occurs. To date, processors either stay idle or execute instructions out of order when they encounter cache misses. As the speed of processors become faster, the penalty for a cache miss is heavier. Having multiple copies of state elements on the processor and coupling them to a multiplexer permits the processor to save the context of the current instructions and resume executing new instructions within one clock cycle. The invention disclosed is particularly useful for minimizing the average instruction cycle time for very fast processor with a main memory access time exceeding 15 processor clock cycles.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features and advantages of the method and apparatus of the present invention will be apparent from the following detailed description of the invention in which:

Figure 1A illustrates a conceptual block diagram of a processor.

Figure 1B illustrates a conceptual block diagram of the state elements and stateless elements of a processor.

Figure 2 illustrates a block diagram of the state of multiple N processes.

Figure 3 illustrates a detail description of the state elements of a very fast processor.

Figure 4 illustrates an execution unit of a very fast processor utilizing the teachings of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

An apparatus and method for switching the context of state elements in a very fast processor upon occurrence of a cache miss is disclosed. In the following description, for purposes of explanation, specific devices, signals, and data structures are disclosed in order to more thoroughly understand the present invention. However, it will be apparent to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well known circuits, devices, and data structures are not disclosed herein to avoid obscuring the present invention unnecessarily.

Figure 1A is a conceptual block diagram of a processor. A register file **1** is coupled to a combinational logic circuit **3** and registers **5**. The combinational logic circuit **3** and the registers **5** are responsible for decoding instruction from the register file **1**. Next, the decoded instructions are transferred to combinational logic circuit **7** and registers **9** for execution. Thereafter, the results of executed instructions are written on an communication or instruction bus **8**.

Figure 1B is a conceptual block diagram of the state elements and stateless elements of a processor. State elements **2** are coupled to a pipeline **6** over an instruction bus **4**. By state elements, the present invention refers to flip-flops and registers on the the processor chip which store binary information indicative of the state of a particular process or processes. Such state elements typically comprise of register files, status bits, condition codes, and pre-fetched instructions. On the other hand, the pipeline **6** is stateless, i.e. does not store information indicative of the state of process that needs to be preserved during execution of a process. Most fast processors have at least a five-stage pipeline: they include fetch instruction, decode instruction, queue instruction for execution, execute instruction, and write result of instruction. For further reference, please see Mano, M. Morris, Computer System Architecture, (Prentice-Hall: 2nd. ed.), Chapter 22.

Figure 2 illustrates a typical block diagram of the state of two processes feeding the pipeline 6. It will be appreciated that a plurality n processes can feed one pipeline, and the processes can feed each other. It is an objective of the present invention to maximize the speed of the processor by utilizing the pipeline to the fullest extent. As such, when a cache miss occurs, the pipeline shall switch from one process to another. In the present example, if the pipeline 6 is executing process 8 when a cache miss occurs, the pipeline 6 is switched within one clock cycle to execute process 10 while saving the contents or states of process 8 prior to the cache miss. Should process 10 encounters a cache miss before the process 8 retrieves the necessary data from the main memory, the pipeline 6 switches within one clock cycle to another process. Through this construction, the context of a plurality process 11 may be saved.

Figure 3 shows a detailed description of the various state elements of a processor 13 and a sketch of its execution unit. A pre-fetched queue 12 prefetches and decodes instructions from memory. For simple instructions, pre-fetched queue transmits execution microinstructions to an execution unit 14 over a microinstruction bus 16. For more complex instructions, the pre-fetched queue 12 starts the microprogram flow by sending signals to a microprogram sequencer 18 over the bus 16. An address translation unit 22 implements efficient on-chip memory management by performing virtual to physical address translation for all memory translation. When a cache miss occurs, the pre-fetched queue 12 sends a microinstruction to the address translation unit 22 over the bus 16 to fetch the address of the next process from memory. Finally, a bus control logic 26 manages the bus protocol and recognizes events such as interrupts and initialization.

Referring again to Figure 3, at the lower left hand corner of the illustration is a detailed sketch of a typical execution unit of a very fast processor. A write-read buffer 30 is coupled to a register file 32. An arithmetic logic shift unit (ALSU) 36 is further coupled to the register file 32 over a plurality

of multiplexers 34. The output of (ALSU) 36 is registered with condition codes and other registers 38 before it is transmitted on a result bus 40.

According to the principles of the present invention, state flip-flops of each of state elements of a processor is duplicated. A multiplexer is coupled to all the duplicate flip-flops. As such, Figure 4 shows the present invention as applied to the execution unit 14 of the processor 13. A multiplexer 29 is coupled to a plurality n number of write-read buffer 30 - 30". The plurality of write-read buffer is in turn coupled to a plurality of register files 32 - 32" over a multiplexer 31. Before data flows from the plurality of register files 32 - 32" to the ALSU 36, they are channeled over multiplexers 33, 33", and 34. The ALSU 36 is again coupled to a multiplexer 37 before the data are stored in a plurality of condition codes and other registers 38 - 38". Note that the multiplexers in front of the plurality of state flip-flops function to preserve the context of a first process upon occurrence of a cache miss. Within one clock cycle, the pipeline can execute another process without additional idling or delay. Should the second process encounters another cache miss, the processor may return within one clock cycle to finish executing the first process if the necessary data had been retrieved from the main memory. Otherwise, the processor may begin executing a third process. It is understood that a plurality n number of processes who's states are duplicated may easily be saved.

Applying the present invention to a typical processor that delivers 40 MIPS and assuming a 1% cache miss ratio and one clock cycle per instruction, the penalty for every cache miss shall be as low as one (1) clock cycle. The average number of clock cycles per instruction of the processor would be .01(2) + .99(1) = 1.01. As such, the processor deliver 40/1.01 = 39.6 MIPS. It follows that the present invention permits users to harness almost all the speed a fast processor can deliver.

The present invention is practical when the memory cycle time to the system exceeds 15 processor cycles. Although the present invention has been

described with reference to **Figures 1-4**, it will be appreciated that the figures are for illustration only and that a variety of fast processors may be designed to incorporate the teachings of the present invention.

CLAIMS

1. In a processing unit comprising state elements of a plurality of processes, a pipeline, an instruction and data bus, cache memory, and main memory, a method for minimizing the time penalty when a cache miss occurs, said method comprising the steps of:

duplicating said state elements of said processor unit, said state elements being each coupled to multiplexers, said multiplexers being coupled to said pipeline of said processor unit;

storing the context of said state elements of said processor unit;

suspending the execution of a first process in said pipeline upon occurrence of a cache miss;

switching the execution of instruction to one of a plurality of processes in said pipeline.

2. The method according to claim 1 wherein said execution of said first process is suspended within a predetermined number of instruction cycles of said processor unit, said context of said first process being stored in said duplicate pair of state elements.

3. The method according to claim 2 wherein said execution of said one of a plurality of processes is switched within a predetermined number of instruction cycle upon occurrence of a cache miss, said execution being carried out in said pipeline of said processor.

4. The method according to claim 3 wherein said execution of said first process shall resume upon the occurrence of a next cache miss and upon the retrieval of necessary data for said first process from said main memory.

5. The method according to claim 4, wherein upon the occurrence of said next cache miss, if said necessary data for said first process has not been retrieved from said main memory, said method further comprising the steps of:

storing the context of said state elements of said processor unit;

suspending the execution of said one of said plurality of processes in said pipeline;

switching the execution of instruction to another one of said plurality of said processes in said pipeline, said another one of said plurality of said processes being different from said one of said plurality of said processes.

6. The method according to claim 5, wherein said main memory access time exceeds 15 processor clock cycles.

7. The method according to claim 6 wherein said predetermined number of instruction cycles is one.

8. An apparatus including state elements of a plurality of processes, a pipeline, an instruction and data bus, a cache memory, and main memory, for minimizing the time penalty of a processor when a cache miss occurs, comprising:

duplicating means comprising means for storing the contents of a plurality of processes in said pipeline; said duplicating means being formed of a plurality of state elements of said processor unit;

suspending means for halting the execution of instruction of a first process in said pipeline upon the occurrence of a cache miss;

switching means for switching the execution of instruction from said first process in said pipeline to one of a plurality of processes, said switching means being formed of a plurality of multiplexers coupled to said duplicating means.

9. The system as defined in claim 8 wherein said execution of instruction of said first process is suspended within one or more instruction cycle of said processor unit.

10. The system as defined in claim 9 wherein said execution of instruction of said one of a plurality of processes is switched within a predetermined number of instruction cycle upon occurrence of a cache miss, said execution being carried out in said pipeline of said processor unit.

11. The system as defined in claim 10 wherein said execution of instruction of said first process shall resume upon the occurrence of a next cache miss and upon the retrieval of necessary data for said first process from said main memory.

12. The system as defined in claim 11 wherein upon the occurrence of said next cache miss, if said necessary data for said first process has not been retrieved from said main memory:

said duplicating means storing the context of said state elements of said processor unit;

said suspending means suspending the execution of said one of said plurality of said processes in said pipeline;

said switching means switching the execution of

instruction to another one of said plurality of said processes in said pipeline, said another one of said plurality of said processes being different from said one of said plurality of said processes.

13. The system as defined in claim 12 wherein said processor unit executes an instruction cycle within twenty (20) nano seconds.

14. The system as defined in claim 13 wherein said predetermined number of instruction cycles is one.

15. A processing unit comprising a state elements of a plurality of processes, a pipeline, an instruction and data bus, cache memory, and main memory, a method for minimizing the time penalty when a cache miss occurs, substantially as hereinbefore described with reference to the accompanying drawings.

16. An apparatus including state elements of a plurality of processes, a pipeline, an instruction and data bus, a cache memory, and main memory, for minimizing the time penalty of a processor when a cache miss occurs, substantially as hereinbefore described with reference to the accompanying drawings.