

THIS OPINION WAS NOT WRITTEN FOR PUBLICATION

The opinion in support of the decision being entered today (1) was not written for publication in a law journal and (2) is not binding precedent of the Board.

Paper No. 24

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte RICHARD L. SITES
and RICHARD T. WITEK

Appeal No. 95-1351
Application 07/547,630¹

ON BRIEF

Before HARKCOM, Vice Chief Administrative Patent Judge, and JERRY SMITH and BARRETT,
Administrative Patent Judges.

BARRETT, Administrative Patent Judge.

¹ Application for patent filed June 29, 1990, entitled "Prefetch Instruction For Improving Performance In Reduced Instruction Set Processor."

DECISION ON APPEAL

This is a decision on appeal under 35 U.S.C. § 134 from the final rejection of claims 1-4, 6, 8-11, 13, and 15-20, all the claims pending in the application. The amendment after final rejection received March 22, 1994 (Paper No. 17) has been entered.

The invention is directed to a processor system and a method of operating a processor system having a cache and using load and store instructions for accessing a given location in memory. A "FETCH" instruction prefetches a block of data including the given location in memory referred to in the load or store instruction to a faster-access cache in the memory hierarchy before the data block is to be used. This is described in the specification at page 9, lines 8-16, page 41, lines 24-25, and page 42, lines 6-24. "The FETCH instruction is intended to help software bury memory latencies on the order of 100-cycles" (specification, page 42, lines 21-22).

Claim 1 is reproduced below.²

1. A method of operating a processor system of the type having a CPU and a hierarchical memory, the hierarchical memory having a faster-access part and a slower-access part, wherein said faster-access part of said memory is a cache memory, the CPU having a register set including a plurality of registers, comprising the steps of:

executing a sequence of instructions by said CPU, said sequence including a load or store instruction for accessing a given location of said memory and for transferring information between a selected one of said registers and said given location in said memory, the step of executing said load or store instruction including sending an address from said CPU to said memory on a bus;

executing in said sequence a prefetch instruction to move a block of data including said given location from said slower-access part of said memory to said

² It is noted that claim 19 in the claims on appeal depends on cancelled claim 7.

faster-access part, the step of executing said prefetch instruction including sending an address from said CPU to said memory on a bus, said prefetch instruction being executed a number of cycles prior to said load or store instruction, said step of executing said prefetch instruction not altering the content of any of said registers of said register set;

wherein said cache stores multi-word lines of data, and wherein said prefetch instruction moves a block of data larger than one of said multi-word lines.

The examiner relies on the following references:³

Sachs et al. (Sachs) 4,884,197 November 28, 1989

Kane, MIPS R2000 RISC Architecture (Prentice Hall 1987), pages 1-1 to 4-11, A-1 to A-9.

(Motorola) MC68030 Enhanced 32-Bit Microprocessor User's Manual Second Edition (Prentice Hall 1989), pages 3-111 to 3-115, 3-120 to 3-122, 6-1 to 6-17.

Claims 1, 3, 4, 6, 8, 10, 11, 13, and 15-20⁴ stand rejected under 35 U.S.C. § 103 as being unpatentable over Kane and Motorola. Claims 2 and 9 stand rejected under 35 U.S.C. § 103 as being unpatentable over Kane and Motorola as applied in the rejection of claims 1 and 8 further in view of Sachs.

OPINION

We reverse.

³ The examiner cites the paper by Chi et al., Compiler-Driven Cache Policy, Purdue University, June 1987, pages 1-54, in the Examiner's Answer, page 3, but does not mention it further or apply it in any ground of rejection. Accordingly, the paper will not be further discussed.

⁴ The Examiner's Answer lists claims 15, 17, 19, and 20 instead of claims 15-20. However, since claims 16 and 18 are discussed (Examiner's Answer, page 5) it is assumed that the rejection was meant to include claims 16 and 18.

Appellants argue that claims 1 and 8 "are distinguished from Kane by reciting that (1) that a separate prefetch instruction (different from the load or store instruction) is executed before a load or store in an instruction stream, and (2) the prefetch instruction does not alter the content of a register" (Brief, page 5). We agree that both limitations are recited in claims 1 and 8 and that neither limitation is found in Kane or Motorola. It is sufficient to just examine the prefetch limitation.

The examiner finds that "[a]lthough Kane did not specifically detail (claims 1,8) that a prefetch which [sic] was executed one or more cycles ahead of time, Kane taught (e.g., see p. A5) that there was a latency of one instruction" (Examiner's Answer, page 3). The examiner concludes that "it would have been obvious to one of ordinary skill in the DP art at the time of the invention that the instruction which was prefetched must have been prefetched at least one or more cycles ahead of when it was used or executed" (Examiner's Answer, page 4). The examiner's reasoning does not address the actual claim language.

Claim 1 does not recite that an instruction was prefetched ahead of the time it was executed as stated in the examiner's rejection. We agree that instructions in the cache are prefetched in advance of their execution by the processor. Claim 1 recites "a prefetch instruction to move a block of data including said given location [for access by a load or store instruction] from said slower-access part of said memory to said faster-access part . . . said prefetch instruction being executed a number of cycles prior to said load or store instruction." That is, claim 1 requires a separate prefetch instruction, not just the prefetching of an instruction that occurs during normal program execution. Furthermore, what is prefetched is a block of data including the memory location

for accessing by a load or store instruction, not just the load or store instruction itself. Kane does not expressly or impliedly disclose a separate prefetch instruction to move a block of data including the location of memory to be accessed by a load or store instruction.

It is possible that the examiner misapprehends what is meant by latency. Kane describes that the instruction latency is because "the loaded operand is not immediately available to subsequent instructions in a processor with an instruction pipeline" (page 1-7) and "[t]he technique used in many RISC designs to handle this data dependency is to recognize and make visible to compilers the fact that all load instructions inherently have a latency or load delay" (page 1-8). The latency is a delay between a load instruction (which loads data from memory into a register) and the time the next instruction can use the value out of the register. The latency occurs during execution of the instruction; thus, even if the load instruction is prefetched into cache there will still be a latency during execution (which time is normally filled by the compiler inserting instructions which do not depend on the data to be loaded). The claimed prefetch instruction overcomes the latency problem because the memory address to be accessed by the load instruction has been previously put into the cache by the prefetch instruction.

To aid in understanding the invention we attach a copy of pages 402-404 from Computer Architecture A Quantitative Approach by D.A. Patterson and J.L. Hennessy (Morgan Kaufmann Publishers, Inc., 2d ed. 1996), which discusses compiler-controlled prefetching. This reference is not prior art. We note that prefetching is not discussed in the first edition.

The examiner further finds that Kane does not expressly disclose executing a prefetch instruction that does not alter the contents of the registers (Examiner's Answer, page 4). The examiner further relies on the teachings in Motorola. The examiner finds that Motorola teaches a MOVE to CCR instruction (which the examiner states has a correct name of MOVEC) that alters a cache control register which causes data to be fetched to the cache from main memory (burst mode filling) (Examiner's Answer, page 4). The examiner seems to rely on the MOVEC instruction in Motorola as being the prefetch instruction (Examiner's Answer, page 8). We conclude that Motorola also fails to teach a prefetch instruction, as claimed. Therefore, Motorola cannot make obvious the limitation of "said step of executing said prefetch instruction not altering the content of any of said registers of said register set."

We agree with appellants' arguments that the examiner has misinterpreted Motorola. Motorola states (page 6-1): "When appropriate, the bus controller requests a burst mode operation to replace an entire cache line. The cache control register (CACR) is accessible by supervisor programs to control the operation of both [instruction and data] caches." Therefore, the cache fill is under the control of the bus controller whereas claim 1 requires executing the sequence of instructions which includes the prefetch instruction on the CPU. In addition, the burst mode operation occurs after a cache miss in an ordinary read cycle, not prior to any particular instruction. Therefore, the burst mode does not operate as a prefetch instruction executed before (i.e., pre-) a load or store instruction. Even if burst mode filling is considered a form of prefetch, it does not prefetch a "block of data including said given location [for access by a load or store instruction]"

(claim 1), but only prefetches the next sequential instructions. Lastly, appellants are correct that the examiner errs in finding that the burst mode is enabled by a bit in the cache control register being set by the MOVE to CCR instruction. CCR stands for the condition code register, which holds condition codes such as overflow, zero, not-equal-zero, etc. The cache control register is the CACR. It is true that the CACR can be written or read by the MOVEC instruction (page 6-16), but the MOVE to CCR is not the same as MOVEC. The MOVEC would have to be an instruction in the supervisor program (pages 6-1, 6-16). It has not been shown that there is any mechanism for executing a MOVEC instruction "a number of cycles prior to the load or store instruction" (claim 1). In any case, as explained above, the burst mode does not cause a block transfer of data to cache until after there has been a cache miss, so the MOVEC instruction cannot be a prefetch instruction as claimed.

Appeal No. 95-1351
Application 07/547,630

For the reasons stated above, the rejection of independent claims 1 and 8 is reversed. Since dependent claims 2-4, 6, 9-11, 13, and 15-20 incorporate the limitation of claims 1 and 8, the rejection of these claims is also reversed.

REVERSED

GARY V. HARKCOM)	
Vice Chief Administrative Patent Judge)	
)	
)	
)	
)	BOARD OF PATENT
JERRY SMITH)	APPEALS
Administrative Patent Judge)	AND
)	INTERFERENCES
)	
)	
)	
LEE E. BARRETT)	
Administrative Patent Judge)	

Appeal No. 95-1351
Application 07/547,630

John G. Graham
ARNOLD, WHITE & DURKEE
P.O. Box 4433
Houston, TX 44210