

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(2) is not binding precedent of the Board.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. 15

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte JAMES R. REINDERS

Appeal No. 96-1588
Application 08/036,947¹

ON BRIEF

Before THOMAS, HAIRSTON, and MARTIN, Administrative Patent Judges.

MARTIN, Administrative Patent Judge.

DECISION ON APPEAL

This is an appeal under 35 U.S.C. § 134 from the examiner's final rejection of claims 1, 2, 7, 8, 13, 14, 19, 21, and 23

¹ Application for patent filed March 25, 1993.

under 35 U.S.C. § 103. Claims 3-6, 9-12, 15-18, 20, 22, and 24 stand objected to for depending on rejected claims. We reverse.

The invention relates to a computer system that supports multiple instruction issues and, more particularly, to a method of instruction scheduling performed by compilers targeting such computer systems. In scheduling instructions, the method of the invention distinguishes between, inter alia, "squeezed" and "non-squeezed" instructions. The specification (at 4) defines a squeezed instruction as an instruction that cannot be scheduled for parallel execution with any other instructions on the targeted machine and a non-squeezed instruction as an instruction that can be scheduled for parallel execution with at least one other instruction on the targeted machine.

Appellant's Figure 8a shows an example of two four-instruction chains made up of Add, Multiply, and Divide instructions. The first chain consists of instructions "instr-1" through "instr-4" and the second chain of instructions "instr-5" through "instr-8." As explained in the paragraph bridging pages 16 and 17, the Add and Multiply instructions are non-squeezed and the Divide instruction is squeezed. The schedule size is initially selected to be as small as possible without considering the squeezed (i.e., Divide) instructions (page 17, lines 11-15).

Because the two Multiply instructions can be paired with two of the four Add instructions, the shortest possible schedule length is four time slots, labeled SLOTS 0-3 in the Resource Utilization Array of Figure 9a. The scheduler builds a candidate instruction list using the bottom-up approach, starting with the last instruction in each instruction set: instr-4 and instr-8 (Fig. 9a). Since the instruction chain ending with instr-4 contains more non-squeezed instructions (four) than does the instruction chain ending with instr-8 (two), the result is that instr-4 (Multiply) is assigned to the multiplier resource during SLOT 0 (Fig. 9a). Next the chains ending in instr-3 and instr-8 are compared (Fig. 9b). Because the instruction chain ending with instr-3 contains more non-squeezed instructions (three) than does the instruction chain ending with instr-8 (two), instr-3 (Multiply) is assigned to the next time slot available for the multiplier, i.e., SLOT 1. Next, the chains ending in instr-2 and instr-8 are compared (Fig. 9c). Because both chains contain two non-squeezed instructions, the presence of the squeezed instructions ("Divide") in the second chain are used for tie-breaking, with the result that instr-8 (Add) is assigned to the adder resource during its first available time slot, i.e., SLOT 0. Next, the chains ending in instr-2 and instr-7 are

compared (Fig. 9d). Since the chain ending in instr-2 includes two non-squeezed instructions (both Add) and the chain ending in instr-7 includes only one (Add), instr-2 (Add) is selected over instr-7 (Divide). However, because instr-2 (Add) must be executed before instr-3 (Multiply), instr-2 cannot be assigned to SLOT 1 together with instr-3; it must be assigned to the next time slot, SLOT 2. Thus, in addition to distinguishing between squeezed and non-squeezed instructions, appellant's scheduling method takes into account the availability of resources and the dependency of an instruction on another unexecuted instruction, factors which are identified as "resource constraints" and "precedence constraints" in paragraph d of the claim. The analysis proceeds in the foregoing manner (see Figs. 9e to 9h) until all of the instructions have been assigned, with the resulting schedule being shown in Figure 10a.

Claim 1 reads as follows:

1. In a computer system comprising a compiler compiling a plurality of programs targeted for a multi-issue architecture computer, a method implemented by a scheduler of said compiler for determining an execution schedule for executing a basic block of one of said programs on said targeted computer, said method comprising the steps of:

a) setting a schedule size for said execution schedule to be determined in an architectural [sic; architecturally] dependent manner;

b) generating a plurality of unassigned schedule slots based on said schedule size, the number of unassigned schedule slots generated being a function of said schedule size;

c) selecting an instruction of said basic block using a plurality of priority functions, said priority functions distinguishing squeezed instructions from non-squeezed instructions of said basic block, and factoring said distinction into their priority evaluations, said squeezed instructions being instructions that cannot be issued in parallel whereas said non-squeezed instructions are instructions that can be issued in parallel;

d) assigning said selected instruction to one of said unassigned schedule slots without violating resource constraints of said target machine^[2] and precedence constraints of said instructions of said basic block;

e) repeating said steps c) through d) until all instructions of said basic block have been scheduled.

All of the appealed claims stand rejected under 35 U.S.C. § 102 as unpatentable for obviousness over the following reference:

Rasbold et al. (Rasbold) 5,202,975 April 13, 1993

Because appellant treats all of the appealed claims as standing or falling together (Brief at 4), we will address only claim 1.

Rasbold discloses a software compiler which rearranges the order of a basic block of instructions in order to reduce the

² We are construing "said target machine" as a reference to the "targeted computer" recited in the preamble.

overall execution time of those instructions (col. 1, lines 25-33). Referring to Figure 1b, Rasbold's sequencing method begins with all of the instructions being classified in the basic block (10)m, from which they will be moved to the Leader Set (12) and then to the Ready Set (16), unless they are Instructions With Interlocks (18). Figure 2 shows an example of a series of source equations 20 and corresponding intermediate language statements 21 that approximate assembly language instructions (col. 9, lines 11-16). Membership in the Leader Set is determined by constructing a direct acyclic graph (DAG) 22, which depicts the dependency of the instructions in the basic block (col. 9, lines 16-29). Each instruction is assigned a "cost" representing the time consequence of not issuing the instruction; an instruction from which many others depend has a higher cost than an instruction from which few others depend (col. 9, lines 30-36). With the aid of the DAG, each instruction in the basic block that has not yet been scheduled is placed in the Leader Set if it is not dependent on any unissued instructions (col. 9, lines 53-56; col. 11, lines 13-19). This function is represented as step 40 in the flow chart of Figure 4. In step 60, the desired issue time (DIT) is calculated for each instruction in the Leader Set, after which the instructions with a DIT less than the current

simulation time are moved to the Ready Set (step 62) (col. 11, lines 19-29). If the Ready Set is empty (step 44) and the Leader Set is empty (step 54), all of the instructions have been scheduled and the process ends; if the Ready Set is not empty (step 44), the instruction in the Ready Set having the highest cost is scheduled (step 46) and its node and outward edges are removed from the DAG (step 48) (col. 11, lines 33-39). The simulation time is then advanced to the point in time at which the just issued instruction would issue (step 64) (col. 11, lines 40-42). The machine resources such as registers and the functional unit are assigned to the scheduled instruction at step 50, and any new interlocks caused by the assignment of machine resources are checked to see if instructions in the Ready Set need to be moved back into the Leader Set (step 52) (col. 11, lines 42-48). The process then returns to the beginning, to schedule the next instruction (col. 11, lines 48-49).

The examiner concedes that Rasbold does not expressly characterize his instructions as squeezed and non-squeezed, but argues that "it would have been obvious to a person of ordinary skill in the art that the claimed squeezed/(not squeezed) instructions are not more than the dependent/independent

Appeal No. 96-1588
Application 08/036,947

instructions, to the extent claimed" (Answer at 3, para. 11). We agree with appellant that the examiner's position ignores the fact that terms "squeezed" and "non-squeezed" are defined in paragraph c of the claim and at page 4 of the specification in a way that clearly distinguishes them from the dependent/independent concept and that the claim must be construed in accordance with those definitions. See In re Morris, 127 F.3d 1048, 1054, 44 USPQ2d 1023, 1027 (Fed. Cir. 1997):

[T]he PTO applies to the verbiage of the proposed claims the broadest reasonable meaning of the words in their ordinary usage as they would be understood by one of ordinary skill in the art, taking into account whatever enlightenment by way of definitions or otherwise that may be afforded by the written description contained in the applicant's specification.

Nothing in Rasbold suggests selecting instructions based on the claimed squeezed/non-squeezed distinction. The DAG diagram in Rasbold's Figure 2 and the associated "cost" for each instruction node clearly concern dependence versus independence. Rasbold's teaching of avoiding interlocks caused by the assignment of machine resources (col. 11, lines 42-48) corresponds to appellant's step of selecting instructions without violating "resource constraints" (claim para. d); it is not a squeezed/non-squeezed distinction. Nor is Rasbold's calculation of the Desired Issue Time (DIT) for each instruction.

Appeal No. 96-1588
Application 08/036,947

For the foregoing reasons, the rejection of claim 1 under § 103 as unpatentable over Rasbold is reversed, as is the rejection of the claims that stand or fall with claim 1, i.e., claims 2, 7, 8, 13, 14, 19, 21, and 23.

REVERSED

| | | |
|-----------------------------|---|-----------------|
| |) | |
| JAMES D. THOMAS |) | |
| Administrative Patent Judge |) | |
| |) | |
| |) | |
| |) | BOARD OF PATENT |
| KENNETH W. HAIRSTON |) | |
| Administrative Patent Judge |) | APPEALS AND |
| |) | |
| |) | INTERFERENCES |
| |) | |
| JOHN C. MARTIN |) | |
| Administrative Patent Judge |) | |

Appeal No. 96-1588
Application 08/036,947

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 Wilshire Boulevard, 7th Floor
Los Angeles, CA 90025