



G00002

ARRAY SEARCHING OPERATIONS

BACKGROUND

5 This invention relates to array searching operations for a computer.

Many conventional programmable processors, such as digital signal processors (DSP), support a rich instruction set that includes numerous instructions for manipulating arrays of data. These operations are typically computationally intensive and can require
10 significant computing time, depending upon the number of execution units, such as multiply-accumulate units (MACs), within the processor.

SUMMARY OF THE INVENTION

15 According to the invention in a first aspect, there is provided a method comprising: receiving N/M machine instructions directing a processor to search an array of data elements, where N and M are integers greater than one; and executing a first machine instruction by: retrieving M data elements in a single fetch cycle ; concurrently comparing the M data elements to corresponding current extreme values; and updating a
20 set of references based on the comparing wherein the set of M references comprise pointer registers to store addresses of extreme data quantities in the array of N data elements.

According to the invention in a second aspect, there is provided a method for searching an array of N data elements for an extreme value, the method comprising: issuing N/M machine instructions to a processor, wherein the processor is adapted to process M data elements in parallel; executing a first machine instruction by: retrieving M data
5 elements; concurrently comparing the M data elements to corresponding M current extreme values, and updating accumulators and pointers associated with the M current extreme values based on said comparing, the pointers including one or more pointer registers to store addresses of extreme values in the array of N data elements; and analyzing results of the machine instructions to identify the extreme value in the array.

10

According to the invention in a third aspect, there is provided a method comprising: retrieving a pair of data elements from an array of elements in a single fetch operation, wherein the pair of data elements includes an even data element and an odd data element; substantially comparing the even element of the pair and the odd element of
15 the pair; maintaining a first pointer register to store an address for an extreme value of the even data elements and maintaining a second pointer register to store an address for an extreme value of the odd data elements; and substantially fetching and comparing the remaining pairs of data elements of the array until all of the data elements of the array have been processed.

20

According to the invention in a fourth aspect, there is provided an apparatus comprising: an execution pipeline adapted to process M data elements in parallel; and a control unit adapted to direct the execution pipeline to search an array of N data elements for an extreme value in response to N/M machine instructions, the execution pipeline being

configured to: retrieve M data elements from the array of N data elements in a single fetch cycle; concurrently compare the retrieved M data elements to corresponding M current extreme values, and update accumulators and pointers associated with the M current extreme values based on said comparing, the pointers including one or more
5 pointer registers to store information indicative of addresses of extreme values in the array of N data elements.

According to the invention in a fifth aspect, there is provided an article comprising a medium having computer- executable instructions stored thereon for compiling a
10 software program, wherein the computer-executable instructions are adapted to generate N/M machine instructions to search an array of N data elements to find an extreme value, each machine instruction causing a programmable processor to: retrieve M data elements from an array of N elements in a single fetch operation; concurrently compare the retrieved M data elements to M corresponding current extreme values; and update
15 accumulators and pointers associated with the M current extreme values based on said comparing, the pointers including one or more pointer registers to store information indicative of extreme values in the array on N elements.

According to the invention in a sixth aspect, there is provided a system comprising: a
20 memory device ; and a processor coupled to the memory device, wherein the processor includes a pipeline configured to process M data elements in parallel and a control unit configured to direct the pipeline to search an array of N data elements for an extreme value in response to N/M machine instructions, wherein in response to each of the machine instructions, the pipeline being configured to: retrieve M data elements from

the array of N data elements in a single fetch cycle; concurrently compare the retrieved M data elements to a corresponding M current extreme values, and update accumulators and pointers associated with the M current extreme values based on said comparing, the pointers including one or more pointer registers to store information indicative of

5 addresses of extreme values in the array of N data elements.

DESCRIPTION OF DRAWINGS

Figure 1 is a block diagram illustrating an example of a pipelined programmable processor according to the invention.

5

Figure 2 is a block diagram illustrating an example execution pipeline for the programmable processor.

Figure 3 is a flowchart for implementing an example array manipulation machine

10 instruction according to the invention.

Figure 4 is a flowchart of an example routine for invoking the machine instruction.

Figure 5 shows a search instruction; and

Figure 6 shows N/M search instruction.

DESCRIPTION

Figure 1 is a block diagram illustrating a programmable
5 processor 2 having an execution pipeline 4 and a control
unit 6. Processor 2, as explained in detail below, reduces
the computational time required by array manipulation
operations. In particular, processor 2 may support a
machine instruction, referred to herein as the SEARCH
10 instruction, that reduces the computational time to search
an array of numbers in a pipelined processing environment.

Pipeline 4 has a number of stages for processing
instructions. Each stage processes concurrently with the
other stages and passes results to the next stage in
15 pipeline 4 at each clock cycle. The final results of each
instruction emerge at the end of the pipeline in rapid
succession.

Control unit 6 controls the flow of instructions and
data through the various stages of pipeline 4. During the
20 processing of an instruction, for example, control unit 6
directs the various components of the pipelined to fetch and
decode the instruction, perform the corresponding operation
and write the results back to memory or local registers.

Figure 2 illustrates an example pipeline 4 configured
25 according to the invention. Pipeline 4, for example, has
five stages: instruction fetch (IF), decode (DEC), address

calculation (AC), execute (EX) and write back (WB).

Instructions are fetched from memory, or from an instruction cache, during the IF stage by fetch unit 21 and decoded within address registers 22 during the DEC stage. At the
5 next clock cycle, the results pass to the AC stage, where data address generators 23 calculate any memory addresses that are necessary to perform the operation.

During the EX stage, execution units 25A through 25M perform the specified operation such as, for example, adding
10 or multiplying numbers, in parallel. Execution units 25 may contain specialized hardware for performing the operations including, for example, one or more arithmetic logic units (ALU's), floating-point units (FPU) and barrel shifters. A variety of data can be applied to execution units 25 such as
15 the addresses generated by data address generator 23, data retrieved from data memory 18 or data retrieved from data registers 24. During the final stage (WB), the results are written back to data memory or to data registers 24.

The SEARCH instruction supported by processor 2, may
20 allow software applications to search an array of N data elements by issuing N/M search instructions, where M is the number of data elements that can be processed in parallel by execution units 25 of pipeline 4. Note, however, that a single execution unit may be capable of executing two or
25 more operations in parallel. For example, an execution unit

may include a 32-bit ALU capable of concurrently comparing two 16-bit numbers.

Generally, the sequence of SEARCH instructions allow the processor to process M sets of elements in parallel to identify an ``extreme value'', such as a maximum or a minimum, for each set. During the execution of the search instructions, processor 2 stores references to the location of the extreme value of each of the M sets of elements. Upon completion of the N/M instructions, as described in detail below, the software application analyzes the references to the extreme values for each set to quickly identify an extreme value for the array. For example, the instruction allows the software applications to quickly identify either the first or last occurrence of a maximum or minimum value. Furthermore, as explained in detail below, processor 2 implements the operation in a fashion suitable for vectorizing in a pipelined processor across the M execution units 25.

As described above, a software application searches an array of data by issuing N/M SEARCH machine instructions to processor 2. Figure 3 is a flowchart illustrating an example mode of operation 20 for processor 2 when it receives a single SEARCH machine instruction. Process 20 is described with reference to identifying the last occurrence of a minimum value within the array of elements; however, process 20 can be easily modified to perform other functions

such as identifying the first occurrence of a minimum value, the first occurrence of a maximum value or a last occurrence of a maximum value.

For exemplary purposes, process 20 is described in
5 assuming M equals 2, i.e., processor 2 concurrently processes two sets of elements, each set having $N/2$ elements. However, the process is not limited as such and is readily extensible to concurrently process more than two sets of elements. In general, process 20 facilitates
10 vectorization of the search process by fetching pairs of elements as a single data quantity and processing the element pairs through pipeline 4 in parallel, thereby reducing the total number of clock cycles necessary to identify the minimum value within the array. Although
15 applicable to other architectures, process 20 is well suited for a pipelined processor 2 having multiple execution units in the EX stage. For each set of elements, process 20 maintains two pointer registers, P_{Even} and P_{Odd} , that store locations for the current extreme value within the
20 corresponding set. In addition, process 20 maintains two accumulators, $A0$ and $A1$, that hold the current extreme values for the sets. The pointer registers and the accumulators, however, may readily be implemented as general-purpose data registers without departing from process 30.

25 Referring to Figure 3, in response to each SEARCH instruction, processor 2 fetches a pair of elements in one

clock cycle as a single data quantity (21). For example, processor 2 may fetch two adjacent 16-bit values as one 32-bit quantity. Next, processor 2 compares the even element of the pair to a current minimum value for the even elements (22) and the odd element of the pair to a current minimum value for the odd elements (24).

When a new minimum value for the even elements is detected, processor 2 updates accumulator A0 to hold the new minimum value and updates a pointer register P_{Even} to hold a pointer to corresponding data quantity within the array (23). Similarly, when a new minimum value for the odd elements has been detected, processor 2 updates accumulator A1 and a pointer register P_{Odd} (25). In this example, each pointer register P_{Even} and P_{Odd} points to the data quantity and not the individual elements, although the process is not limited as such. Processor 2 repeats the process until all of the elements within the array have been processed (26). Because processor 2 is pipelined, element pairs may be fetched until the array is processed.

The following illustrates exemplary syntax for invoking the machine instruction:

$$(P_{\text{Odd}}, P_{\text{Even}}) = \text{SEARCH } R_{\text{Data}} \text{ LE, } R_{\text{Data}} = [P_{\text{fetch_addr}}++]$$

Data register R_{Data} is used as a scratch register to store each newly fetched data element pair, with the least significant word of R_{Data} holding the odd element and the

most significant word of R_{data} holding the even element. Two accumulators, A0 and A1, are implicitly used to store the actual values of the results. An additional register, P_{fetch_addr} , is incremented when the SEARCH instruction is
5 issued and is used as a pointer to iterate over the $N/2$ data quantities within the array. The defined condition, such as ``less than or equal'' (LE) in the above example, controls which comparison is executed and when the pointer registers P_{Even} and P_{Odd} , as well as the accumulators A0 and A1, are
10 updated. The ``LE'', for example, directs processor 2 to identify the last occurrence of the minimum value.

In a typical application, a programmer develops a software application or subroutine that issues the N/M search instructions, probably from within a loop construct.
15 The programmer may write the software application in assembly language or in a high-level software language. A compiler is typically invoked to process the high-level software application and generate the appropriate machine instructions for processor 2, including the SEARCH machine
20 instructions for searching the array of data.

Figure 4 is a flowchart of an example software routine
30 for invoking the example machine instructions illustrated above. First, the software routine 30 initializes the registers including initializing A0 and A1 and pointing P_{Even}
25 and P_{Odd} to the first data quantity within the array (31). In one embodiment, software routine 30 initializes a loop

count register with the number of SEARCH instructions to issue (N/M). Next, routine 30 issues the SEARCH machine instruction N/M times. This can be accomplished a number of ways, such as by invoking a hardware loop construct supported by processor 2. Often, however, a compiler may unroll a software loop into a sequence of identical SEARCH instructions (32).

After issuing N/M search instructions, A0 and A1 hold the last occurrence of the minimum even value and the last occurrence of the minimum odd value, respectively. Furthermore, P_{Even} and P_{Odd} store the locations of the two data quantities that hold the last occurrence of the minimum even value and the last occurrence of the minimum odd value.

Next, in order to identify the last occurrence of the minimum value for the entire array, routine 30 first increments P_{Odd} by a single element, such that P_{Odd} points directly at the minimum odd element (33). Routine 30 compares the accumulators A0 and A1 to determine whether the accumulators contain the same value, i.e., whether the minimum of the odd elements equals the minimum of the even elements (34). If so, the routine 30 compares the pointers to determine whether P_{Odd} is less than P_{Even} and, therefore, P_{Odd} and P_{Even} whether the minimum even value occurred earlier in the array (35). Based on the comparison, the routine determines whether to copy P_{Odd} into P_{Even} (37).

When the accumulators A0 and A1 are not the same, the routine compares A0 to A1 in order to determine which holds the minimum value (36). If A1 is less than A0 then routine 30 sets P_{Even} equal to P_{Odd} , thereby copying the pointer to the
5 minimum value from P_{Odd} into P_{Even} (37).

At this point, P_{Even} points to the last occurrence of the minimum value for the entire array. Next, routine 30 adjusts P_{Even} to compensate for errors introduced to the pipelined architecture of processor 2 (38). For example,
10 the comparisons described above are typically performed in the EX stage of pipeline 4 while incrementing the pointer register P_{fetch_addr} typically occurs during the AC stage, thereby causing the P_{Odd} and P_{Even} to be incorrect by a known quantity. After adjusting P_{Even} , routine 30 returns P_{Even} as a
15 pointer to the last occurrence of the minimum value within the array(39).

Figure 5 illustrates the operation for a single SEARCH instruction as generalized to the case where processor 2 is capable of processing M elements of the array in parallel,
20 such as when processor 2 includes M execution units. The SEARCH instruction causes processor 2 to fetch M elements in a single fetch cycle (51). Furthermore, in this example, processor 2 maintains M pointer registers to store addresses (locations) of a corresponding extreme value for each of the
25 M sets of elements. After fetching the M elements, processor 2 concurrently compares the M elements to a

current extreme value for the respective element set, as stored in M accumulators (52). Based on the comparisons, processor 2 updates the M accumulators and the M pointer registers (53).

5 Figure 6 illustrates the general case where a software application issues N/M SEARCH instructions and, upon completion of the instructions, determines the extreme value for the entire array. First, the software application initializes a loop counter, the M accumulators used to store
10 the current extreme values for the M element sets and the M pointers used to store the locations of the extreme values (61). Next, the software application issues N/M SEARCH instructions (62). After completion of the instructions, the software application may adjust the M pointer registers
15 to correctly reference its respective extreme value, instead of the data quantity holding the extreme value (63). After adjusting the pointer registers, the software application compares the M extreme values for the M element sets to identify an extreme value for the entire array, i.e., a
20 maximum value or a minimum value (64). Then, the software application may use the pointer registers to determine whether more than one of the element sets have an extreme value equal to the array extreme value and, if so, determine which extreme value occurred first, or last, depending upon
25 the desired search function (65).

Various embodiments of the invention have been described. For example, a single machine instruction has been described that searches an array of data in a manner that facilitates vectorization of the search process within
5 a pipelined processor. The processor may be implemented in a variety of systems including general purpose computing systems, digital processing systems, laptop computers, personal digital assistants (PDA's) and cellular phones. For example, cellular phones often maintain an array of
10 values representing signal strength for services available 360° around the phone. In this context, the process discussed above can be readily used upon initialization of the cellular phone to scan the available services and quickly select the best service. In such a system, the
15 processor may be coupled to a memory device, such as a FLASH memory device or a static random access memory (SRAM), that stores an operating system and other software applications. These and other embodiments are within the scope of the following claims.

CLAIMS

1. A method comprising: receiving N/M machine instructions directing a processor to
5 search an array of data elements, where N and M are integers greater than one; and
executing a first machine instruction by:
retrieving M data elements in a single fetch cycle ;
concurrently comparing the M data elements to corresponding current extreme values;
and updating a set of references based on the comparing wherein the set of M references
10 comprise pointer registers to store addresses of extreme data quantities in the array of N
data elements.
2. The method of claim 1, wherein retrieving the M data elements comprises retrieving
the M data elements as a single data quantity containing the M data elements.
- 15 3. The method of claim 1, further comprising determining an address of an extreme
value based on a value in a pointer register and based on a correction factor to
compensate for one or more errors..
- 20 4. The method of claim 1, wherein $M = 2$ and N is greater than two.
5. The method of claim 1, wherein executing the first machine instruction further
includes: storing the current M extreme values in M accumulators; and copying the M
data elements to the accumulators based on the comparing.

6. The method of claim 4, wherein concurrently comparing the M data elements comprises processing a first data element with a first execution unit of a pipelined processor and processing a second data element with a second execution unit of the
5 pipelined processor.

7. The method of claim 4, wherein concurrently comparing the M data elements comprises concurrently processing a first data element and a second data element within a single execution unit of a pipelined processor.
10

8. The method of claim 1, wherein concurrently comparing the M data elements to M corresponding current extreme values includes determining whether each of the data elements is less than the corresponding current extreme value.

15 9. The method of claim 1, wherein concurrently comparing the M data elements to M corresponding current extreme values includes determining whether each of the data elements is greater than the corresponding current extreme value.

10. A method for searching an array of N data elements for an extreme value, the
20 method comprising:
issuing N/M machine instructions to a processor, wherein the processor is adapted to process M data elements in parallel;
executing a first machine instruction by:
retrieving M data elements;

concurrently comparing the M data elements to corresponding M current extreme values, and

updating accumulators and pointers associated with the M current extreme values based on said comparing, the pointers including one or more pointer registers to store

5 addresses of extreme values in the array of N data elements; and

analyzing results of the machine instructions to identify the extreme value in the array.

11. The method of claim 10, further comprising: setting up registers for accumulators and pointers.

10

12. A method comprising:

retrieving a pair of data elements from an array of elements in a single fetch operation, wherein the pair of data elements includes an even data element and an odd data element;

15 substantially comparing the even element of the pair and the odd element of the pair; maintaining a first pointer register to store an address for an extreme value of the even data elements and maintaining a second pointer register to store an address for an extreme value of the odd data elements; and

substantially fetching and comparing the remaining pairs of data elements of the array

20 until all of the data elements of the array have been processed.

13. The method of claim 12, wherein substantially comparing the pair of data elements includes setting an even minimum value as function of the even element of the element

pair and setting an odd minimum value as a function of the odd element of the element pair.

14. The method of claim 12, wherein substantially comparing the pair of data elements
5 includes maintaining a first accumulator to store a minimum value for the even elements and a second accumulator to store a minimum value for the odd elements.

15. The method of claim 12, wherein the extreme value of the even data elements is a minimum value, and wherein the extreme value of the odd data elements is a minimum
10 value.

16. The method of claim 12, further including adjusting at least one of the pointer registers after processing all of the pairs of data elements to account for a number of stages in the pipeline.

15

17. The method of claim 12, wherein the method is invoked by issuing N/M machine instructions to a programmable processor, wherein N equals the number of elements in the array, and M equals the number of data elements that the processor can concurrently compare.

20

18. An apparatus comprising: an execution pipeline adapted to process M data elements in parallel; and

a control unit adapted to direct the execution pipeline to search an array of N data elements for an extreme value in response to N/M machine instructions, the execution pipeline being configured to:

retrieve M data elements from the array of N data elements in a single fetch cycle;

5 concurrently compare the retrieved M data elements to corresponding M current extreme values, and

update accumulators and pointers associated with the M current extreme values based on said comparing, the pointers including one or more pointer registers to store information indicative of addresses of extreme values in the array of N data elements.

10

19. The apparatus of claim 18, wherein in response to the machine instructions, the control unit directs the pipeline to set up registers for accumulators and pointers.

20. The apparatus of claim 18, wherein the pipeline includes M registers adapted to
15 store accumulators and pointers associated with the extreme values.

21. The apparatus of claim 20, wherein the registers include first and second pointer registers to store information indicative of addresses of first and second extreme values of the array.

20

22. The apparatus of claim 20, wherein the registers are general-purpose data registers.

23. The apparatus of claim 18, wherein the pipeline includes M accumulators to store M current extreme values.

24. The apparatus of claim 18, wherein the pipeline includes M general-purpose registers to store M current extreme values.

- 5 25. An article comprising a medium having computer- executable instructions stored thereon for compiling a software program, wherein the computer-executable instructions are adapted to generate N/M machine instructions to search an array of N data elements to find an extreme value, each machine instruction causing a programmable processor to:
- 10 retrieve M data elements from an array of N elements in a single fetch operation; concurrently compare the retrieved M data elements to M corresponding current extreme values; and
- update accumulators and pointers associated with the M current extreme values based on said comparing, the pointers including one or more pointer registers to store
- 15 information indicative of extreme values in the array on N elements.

26. The article of claim 25, wherein the first machine instruction causes the processor to concurrently process a first data element and a second data element within a single execution unit of a pipelined processor.

20

27. A system comprising: a memory device ; and a processor coupled to the memory device, wherein the processor includes a pipeline configured to process M data elements in parallel and a control unit configured to direct the pipeline to search an array of N

data elements for an extreme value in response to N/M machine instructions, wherein in response to each of the machine instructions, the pipeline being configured to:

retrieve M data elements from the array of N data elements in a single fetch cycle;

concurrently compare the retrieved M data elements to a corresponding M current

5 extreme values, and

update accumulators and pointers associated with the M current extreme values based on said comparing, the pointers including one or more pointer registers to store information indicative of addresses of extreme values in the array of N data elements.

10 28. The system of claim 27, wherein the pipeline includes M registers configured to store references to the extreme values.

29. The system of claim 28, wherein the registers include first and second pointer registers to store information indicative of addresses of first and second extreme values
15 of the array.

30. The system of claim 28, wherein the registers are general-purpose data registers.

31. The system of claim 27, wherein the memory device comprises static random access
20 memory.

32. The system of claim 27, wherein the memory device comprises FLASH memory.