

REGISTERS FOR 2-D MATRIX PROCESSING

BACKGROUND

1. FIELD

The present invention relates generally to computer systems and, more specifically, to processor architectures.

2. DESCRIPTION

Some processors are designed to provide extensions to their instruction set architecture (ISA) for multimedia operations. For example, the MMX™ instructions supported by Pentium® II, Pentium® III, and Celeron™ processors, commercially available from Intel Corporation of Santa Clara, California, implement various functions useful for multimedia applications, such as digital signal processing, and audio and video processing. These instructions support "single instruction multiple data" (SIMD) operations on multimedia and communications data types. Although the use of these instructions provide an improvement over combinations of pre-existing instructions to perform a given function, and individual MMX™ instructions are efficient for some types of processing, various impediments to faster multimedia processing still remain. For example, many implementations of block-based image and video processing algorithms (such as joint photographic experts group (JPEG) and moving picture expert group (MPEG) schemes) result in the data, stored in a set of registers accessible as operands for the MMX™ instructions, being transposed during matrix mathematical operations. The transposition of data among registers incurs significant overhead, however, thereby slowing overall processor throughput for multimedia processing. Therefore, any techniques for avoiding or minimizing these delays would be a valuable advance in the processor art.

SUMMARY

An embodiment of the present invention is a processor having a first set of registers, the first set storing a matrix of data, and a second set of registers coupled to the first set, the second set storing a transposed copy of the matrix of data.

Another embodiment of the present invention is a method of using two sets of registers for matrix processing by a processor. The method includes storing a matrix of data into a first set of registers, the first set of registers having a first number of registers, each register comprising a first number of storage units, each storage unit storing an element of the matrix, and transposing the matrix of data into a second set of registers, the second set of registers having a second number of registers, each register comprising a second number of storage units. The method also includes referencing one of the first set of registers to operate on a row of the matrix of data, and referencing one of the second set of registers to operate on a column of the matrix of data.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

Figure 1 is a diagram of a set of MMX™ registers according to the prior art;

Figure 2 is a diagram of the set of MMX™ registers storing an 8 pixel by 8 pixel block of image data;

Figure 3 is a diagram of the set of MMX™ registers storing the transposed 8 pixel by 8 pixel block of image data;

Figure 4 is a diagram of a virtual MMX™ register set linked to the MMX™ register set according to an embodiment of the present invention;

Figure 5 is a diagram of the set of virtual MMX™ registers storing the transposed 8 pixel by 8 pixel block of image data; and

Figure 6 is a diagram of a system having a processor with a MMX™ register set and a virtual MMX™ register set according to an embodiment of the present invention.

DETAILED DESCRIPTION

An embodiment of the present invention comprises a method and apparatus for extending MMX™ registers to be more efficient for two dimensional (2-D) matrix operations.

Reference in the specification to "one embodiment" or "an embodiment" of the present invention means that a particular feature, structure or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, the appearances of the phrase "in one embodiment" appearing in various places throughout the specification are not necessarily all referring to the same embodiment.

When executing an instruction, a processor typically references one or more register operands. For MMX™ instructions, the register operands may be one or more of a special set of registers called MMX™ registers. Figure 1 is a diagram of a set of MMX™ registers according to the prior art. In the register set 10 shown in Figure 1, there are eight MMX™ registers labeled mm0 12 through mm7 14. In other embodiments, the number of registers may be higher or lower than eight. Each register includes a

plurality of units of data, ordered from a low unit 16 to a high unit 18 as shown. In one embodiment, a unit comprises a byte. In other embodiments, a unit may comprise a word, a double word, or other storage unit. In at least one known system, the number of units (e.g., bytes) per MMX™ register is eight, although in other systems other numbers of units may be employed. To efficiently implement SIMD multimedia processing using the MMX™ registers, data to be processed should be aligned in such a way that multiple related data items are arranged in a single MMX™ register. For example, suppose an 8 pixel by 8 pixel block of image data is laid out in the MMX™ registers as shown in Figure 2, with each pixel value $P(i, j)$ being represented in one unit and the overall set of registers representing a matrix. The 8 pixel by 8 pixel block may be a portion of a larger image. In this example, the first row of image data of the block is stored in the first MMX™ register, mm0 12, with the first column of the first row being stored in the low unit of mm0 and the last column of the first row being stored in the high unit of mm0, the second row of image data is stored in the second MMX™ register, mm1 20, with the first column of the second row being stored in the low unit of mm1 and the last column of the second row being stored in the high unit of mm1, and so on.

Once the data is stored in the MMX™ registers as shown, the processor may execute instructions to efficiently operate on the 8x8 matrix one row at a time. This type of processing is commonly used in block-based imaging applications, for example, as well as other applications. For example, all of the data of row 0 may be added to the data of row 3 using a single MMX™ instruction as shown below.

PADDB MM0, MM3 ; add row 0 to row 3 and store results in row 0.

However, to operate on the 8x8 matrix one column at a time presents problems because each column's data is spread out among the eight MMX™ registers. For example, the first column's data is distributed among the low units of mm0 12 through mm7 14, and the last column's data is distributed among the high units of mm0 through mm7, respectively. To continue to gain the benefits of using MMX™ SIMD processing, it becomes necessary to transpose the 8x8 matrix as shown in Figure 3. Matrix transposition is well known in mathematics. After transposition, the first MMX™ register mm0 12 stores the original first column of data, with the first row of the original first column being stored in the low unit 16 and the last row of the original first column being stored in the high unit 18 as shown. Similarly, the other MMX™ registers store columns of the 8x8 matrix as shown.

Although an 8x8 matrix transposition may be performed using known pack and unpack instructions, this processing is very inefficient and incurs significant processing overhead. Generally, transposition processing for an 8x8 matrix may be implemented by the performance of four 4x4 matrix transpositions, with each transposition requiring at least 12 processing cycles on a Pentium®III processor. Thus, at least 64 cycles are used merely to reposition the data so that an MMX™ instruction may be used to operate on the eight elements of a given column.

Embodiments of the present invention overcome the need to transpose the data in the MMX™ registers through a series of pack and unpack instructions by providing an additional set of registers in the processor to assist in multimedia processing. In embodiments of the present invention, an equivalent "mirror" set of the MMX™ registers is designed into the processor architecture. This register set may be called the virtual MMX register set, or VMX set. Figure 4 is a diagram of the virtual MMX register set linked to the MMX™ register set according to an embodiment of the present invention. In one embodiment, the number of MMX™ registers in MMX™ register set 10 is the same as the number of

VMX registers provided in VMX register set 22. In addition, in one embodiment, the number of VMX registers is greater than or equal to the number of units in each MMX™ register (with a unit being a byte, word, double word, or other storage unit, depending on the implementation). VMX register set 22 stores the transposed matrix data from the MMX™ register set and may be automatically updated by register update logic 23 whenever any unit of any register in the MMX™ set is modified. Hence, a load of one row of MMX™ register set 10 automatically results in a load of one column of VMX register set 22. For example, the data from the first MMX™ register mm0 12 may be automatically stored in the low units of the VMX registers vm0 through vm7, the data from the second MMX™ register mm1 20 may be automatically stored in the second lowest units of the VMX registers, and so on.

Referring back to Figure 2, if the MMX™ registers are loaded with an 8x8 matrix designated P0,0 to P7,7 as shown, then the VMX registers may be automatically loaded by register update logic with the transposed matrix as shown in Figure 5. To operate on row elements of the matrix, a program may simply reference one or more of MMX™ registers mm0 12 through mm7 14. However, to operate on column elements of the matrix, a program references the VMX registers vm0 24 through vm7 26, instead. Since the MMX™ registers are mirrored in the processor hardware with the VMX registers, there are no coherency issues. All MMX™ instructions may be operated with either of the register sets by using references to the appropriate registers as needed. No changes to the instruction set of the processor are needed, only operand references may be changed in a program.

One advantage of embodiments of the present invention over existing processor architectures is that the invention provides greater parallelism for matrix operations. This is achieved by avoiding costly transposes for

column operations through the execution of multiple pack and unpack instructions.

One example of how the present invention may be employed is in the Discrete Cosine Transform (DCT) algorithm used for processing 8 pixel by 8 pixel blocks in many video processing schemes. Currently, to perform an 8x8 DCT, processing includes performing a 1x8 column transform first, transposing the 8x8 matrix, performing another 1x8 column transform, and then transposing the results again to get the DCT coefficients. The most optimized DCT currently running on Pentium® class processors takes about 300 cycles of processing. Of this amount, about 100 cycles are used to perform the transpose operations for an 8x8 matrix. Thus, implementation of the present invention yields an approximately 30% improvement for the DCT processing. Similar performance gains may be achieved for inverse DCTs. Although the example of a DCT is discussed herein, embodiments of the present invention may be useful for any matrix operations, including those used in various image and video compressions algorithms.

Although the present invention has been discussed above in the context of a two dimensional (2-D) matrix, the concept may be adapted to three or more dimensions. For example, a third register set may be included in the design of the processor to store other transpositions of the matrix data.

Figure 6 is a diagram of a system having a processor with a MMX™ register set and a virtual MMX™ register set according to an embodiment of the present invention. System 50 may include processor 52 coupled to memory 54. Processor 52 comprises various components well-known in the art, many of which have been omitted from Figure 6 for clarity. Instruction memory 56 stores instructions that may reference one or more of MMX™ registers 10 or one or more of VMX registers 22. Register update logic 23 coordinates the automatic updating of the VMX registers 22 when the MMX™ registers change. Multiplexor (MUX) 58 selects data from

either MMX™ registers or VMX registers for input to arithmetic logic unit (ALU) 60. The ALU produces data for data memory 52.

The present invention allows MMX™ registers to be manipulated in a more intuitive manner. With the addition of the mirror register set, implementation of any block-based algorithms may be simplified and their performance increased. Some examples of applications which may benefit from the present invention include discrete cosine transforms (DCTs) used in video compression algorithms, matrix transforms in three dimensional (3-D) graphics algorithms, and others.

While this invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the inventions pertain are deemed to lie within the spirit, and scope of the invention.



CLAIMS

What is claimed is:

1. A processor comprising:
a first set of registers, the first set storing a matrix of data; and
a second set of registers coupled to the first set, the second set storing a transposed copy of the matrix of data.
2. The processor of claim 1, further comprising logic to automatically update the transposed copy of the matrix of data in the second set of registers when any portion of the matrix in the first set of registers is modified.
3. The processor of claim 2, wherein a modification of a row of the matrix in the first set of registers automatically results in a corresponding modification of a column of the transposed copy of the matrix in the second set of registers.
4. The processor of claim 1, wherein the first set of registers comprises a first number of registers, each register comprising a first number of storage units, the second set of registers comprises a second number of registers, each register comprising a second number of storage units, and the second number of storage units is greater than or equal to the first number of registers.
5. The processor of claim 4, wherein the first number of registers is equal to the second number of registers.



6. The processor of claim 5, wherein the first set of registers comprise MMX™ registers, the first number of registers is eight, and the matrix of data comprises image data.

7. The processor of claim 1, wherein the processor executes an instruction referencing one of the first set of registers to operate on a row of the matrix of data and executes an instruction referencing one of the second set of registers to operate on a column of the matrix of data.

8. A method of using two sets of registers for matrix processing by a processor comprising:

storing a matrix of data into a first set of registers, the first set of registers comprising a first number of registers, each register comprising a first number of storage units, each storage unit storing an element of the matrix;

transposing the matrix of data into a second set of registers, the second set of registers comprising a second number of registers, each register comprising a second number of storage units;

referencing one of the first set of registers to operate on a row of the matrix of data; and

referencing one of the second set of registers to operate on a column of the matrix of data.

9. The method of claim 8, further comprising automatically updating the transposed matrix of data in the second set of registers when any portion of the matrix in the first set of registers is modified.

10. The method of claim 9, further comprising modifying a row of the matrix in the first set of registers and automatically modifying a corresponding column of the transposed matrix in the second set of registers



11. The method of claim 8, further comprising performing a transform operation on column data stored in one of the second set of registers.

12. The method of claim 11, wherein performing the transform operation comprises performing a discrete cosine transform operation on column data stored in one of the second set of registers.

13. The method of claim 8, wherein the second number of storage units is greater than or equal to the first number of registers.

14. The method of claim 8, wherein the first number of registers is equal to the second number of registers.

15. The method of claim 8, wherein the first set of registers comprise MMX™ registers, the first number of registers is eight, and the matrix of data comprises image data.

16. A system comprising:

a memory;

a processor coupled to the memory, the processor comprising:

a first set of registers, the first set storing a matrix of data; and

a second set of registers coupled to the first set, the second set storing a transposed copy of the matrix of data.

17. The system of claim 16, wherein the processor further comprises logic to automatically update the transposed copy of the matrix of data in the second set of registers when any portion of the matrix in the first set of registers is modified.



18. The system of claim 17, wherein a modification of a row of the matrix in the first set of registers automatically results in a corresponding modification of a column of the transposed copy of the matrix in the second set of registers.

19. The system of claim 16, wherein the first set of registers comprises a first number of registers, each register comprising a first number of storage units, the second set of registers comprises a second number of registers, each register comprising a second number of storage units, and the second number of storage units is greater than or equal to the first number of registers.

20. The system of claim 19, wherein the first number of registers is equal to the second number of registers.

21. The system of claim 20, wherein the first set of registers comprise MMX™ registers, the first number of registers is eight, and the matrix of data comprises image data.

22. The system of claim 16, wherein the processor executes an instruction referencing one of the first set of registers to operate on a row of the matrix of data and executes an instruction referencing one of the second set of registers to operate on a column of the matrix of data.

23. A method of using two sets of registers for discrete cosine transform (DCT) processing of a matrix of image data by a processor comprising:

storing the matrix into a first set of registers, the first set of registers comprising a first number of registers, each register comprising a first number of storage units, each storage unit storing an element of the matrix;



transposing the matrix into a second set of registers, the second set of registers comprising a second number of registers, each register comprising a second number of storage units;

performing DCT processing, at least in part, by referencing one of the second set of registers to operate on a column of the matrix.

24. The method of claim 23, further comprising automatically updating the transposed matrix of data in the second set of registers when any portion of the matrix in the first set of registers is modified.

25. The method of claim 23, further comprising modifying a row of the matrix in the first set of registers and automatically modifying a corresponding column of the transposed matrix in the second set of registers