US 20050278505A1

(19) United States

(12) Patent Application Publication

Lim et al.

(10) Pub. No.: US 2005/0278505 A1

(43) Pub. Date: Dec. 15, 2005

(54) MICROPROCESSOR ARCHITECTURE INCLUDING ZERO IMPACT PREDICTIVE DATA PRE-FETCH MECHANISM FOR PIPELINE DATA MEMORY

(76) Inventors: **Seow Chuan Lim**, Berkshire (GB); **Kar-Lik Wong**, Wokinham (GB)

Correspondence Address:
HUNTON & WILLIAMS LLP
INTELLECTUAL PROPERTY DEPARTMENT
1900 K STREET, N.W.
SUITE 1200
WASHINGTON, DC 20006-1109 (US)

(21) Appl. No.: **11/132,447**

(22) Filed: **May 19, 2005**

Related U.S. Application Data

(60) Provisional application No. 60/572,238, filed on May 19, 2004.

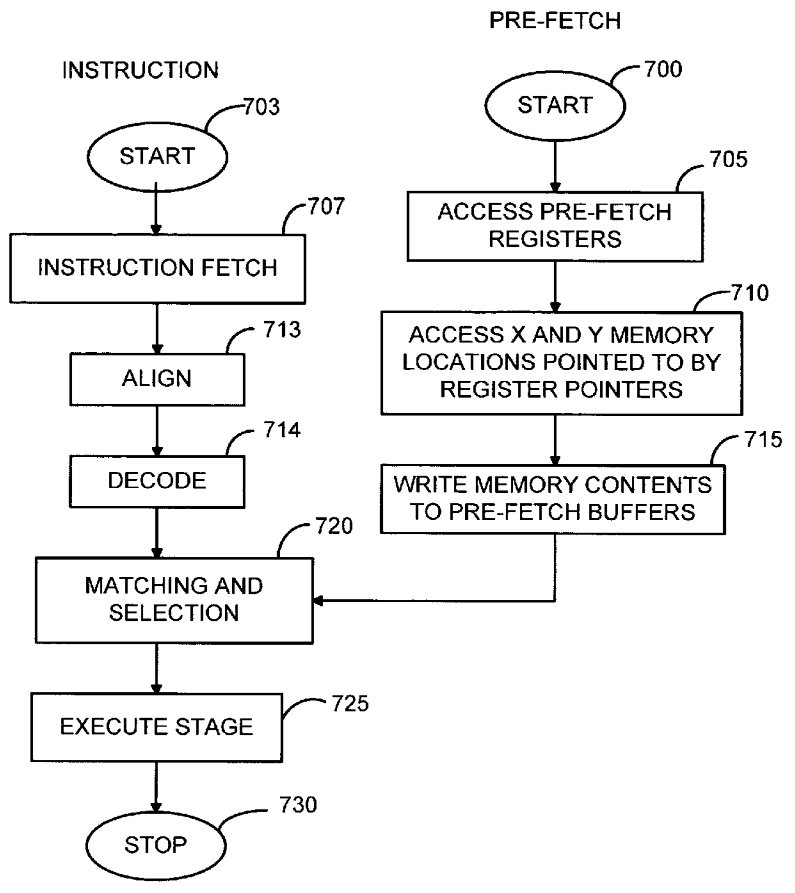
Publication Classification

(51) **Int. Cl.**^{7} **G06F 9/30**

(52) **U.S. Cl.** **712/207**

(57) **ABSTRACT**

A microprocessor architecture including a predictive pre-fetch XY memory pipeline in parallel to the processor's pipeline for processing compound instructions with enhanced processor performance through predictive prefetch techniques. Instruction operands are predictively prefetched from X and Y based on the historical use of operands in instructions that target X and Y memory. After the compound instruction is decoded in the pipeline, the pre-fetched operand pointer, address and data is reconciled with the operands contained in the actual instruction. If the actual data has been pre-fetched, it is passed to the appropriate execute unit in the execute stage of the processor pipeline. As a result, if the prediction is correct, the data to use for access can be selected and the data selected fed to the execution stage without any addition processor overhead. This pre-fetch mechanism avoids the need to slow down the clock speed of the processor or insert stalls for each compound instruction when using XY memory.


PRE-FETCH
INSTRUCTION
START
700
703
START
705
ACCESS PRE-FETCH REGISTERS
707
INSTRUCTION FETCH
710
ACCESS X AND Y MEMORY LOCATIONS POINTED TO BY REGISTER POINTERS
713
ALIGN
714
DECODE
715
WRITE MEMORY CONTENTS TO PRE-FETCH BUFFERS
720
MATCHING AND SELECTION
725
EXECUTE STAGE
730
STOP

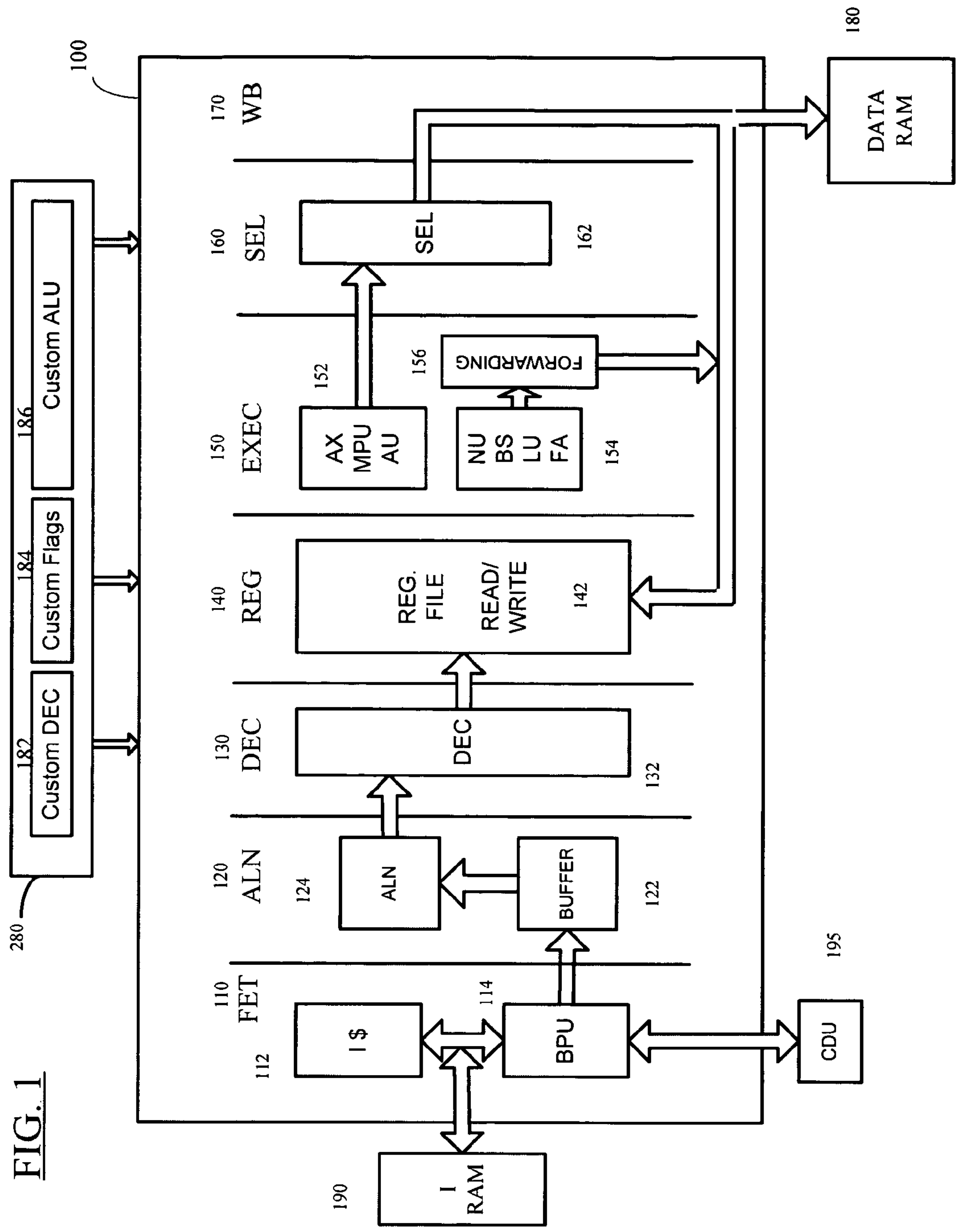
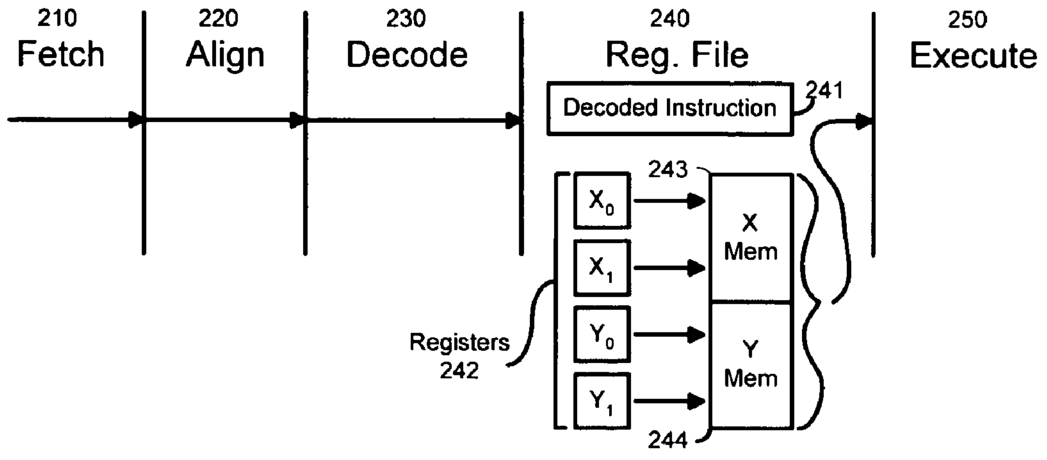

100
110 FET
120 ALN
130 DEC
140 REG
150 EXEC
160 SEL
170 WB
I $
112
114
BPU
BUFFER
122
ALN
124
DEC
132
REG. FILE READ/WRITE
142
AX MPU AU
152
NU BS LU FA
154
FORWARDING
156
SEL
162
DATA RAM
180
I RAM
190
CDU
195
Custom DEC
182
Custom Flags
184
Custom ALU
186
280

FIG. 1

FIG. 2



Decoded Instruction Format 241

Instr_name dest_ptr+update_mode, src1_ptr+update_mode, src2_ptr+update_mode

FIG. 3

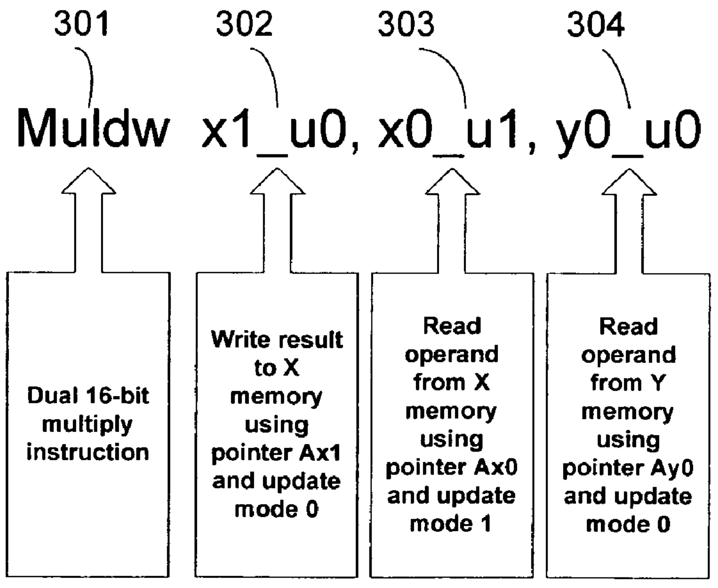

301
302
303
304
Muldw x1_u0, x0_u1, y0_u0
Dual 16-bit multiply instruction
Write result to X memory using pointer Ax1 and update mode 0
Read operand from X memory using pointer Ax0 and update mode 1
Read operand from Y memory using pointer Ay0 and update mode 0

FIG. 4

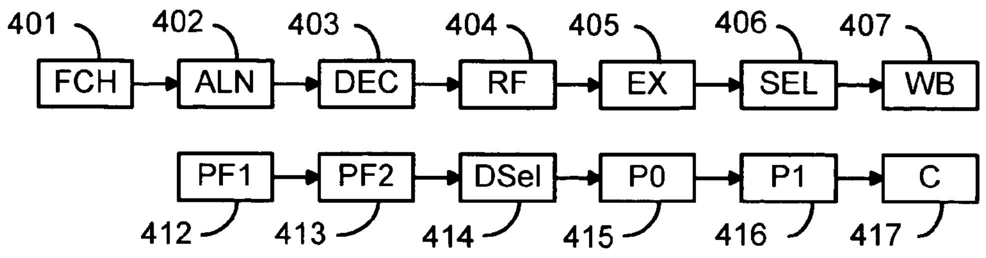

401
FCH
402
ALN
403
DEC
404
RF
405
EX
406
SEL
407
WB
PF1
412
PF2
413
DSel
414
P0
415
P1
416
C
417

FIG. 5

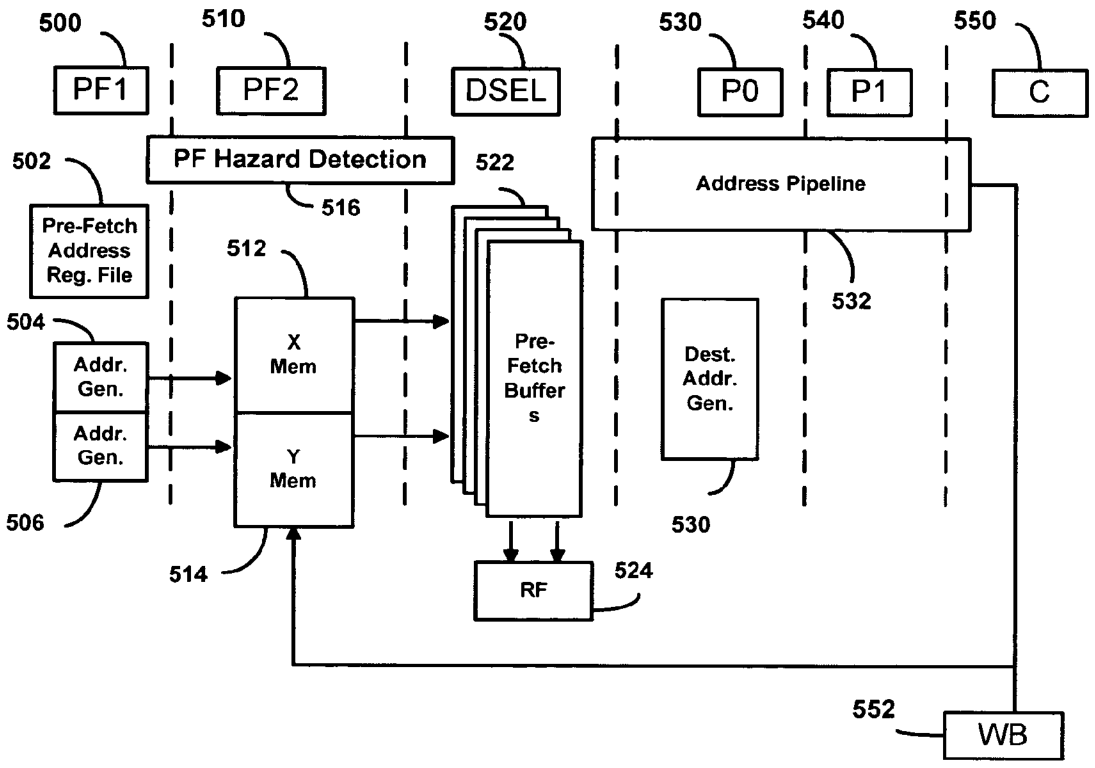

500
PF1
510
PF2
520
DSEL
530
P0
540
P1
550
C
502
Pre-Fetch Address Reg. File
PF Hazard Detection
516
522
Address Pipeline
532
512
504
Addr. Gen.
Addr. Gen.
506
X Mem
Y Mem
514
Pre-Fetch Buffer s
Dest. Addr. Gen.
530
RF
524
552
WB

FIG. 6

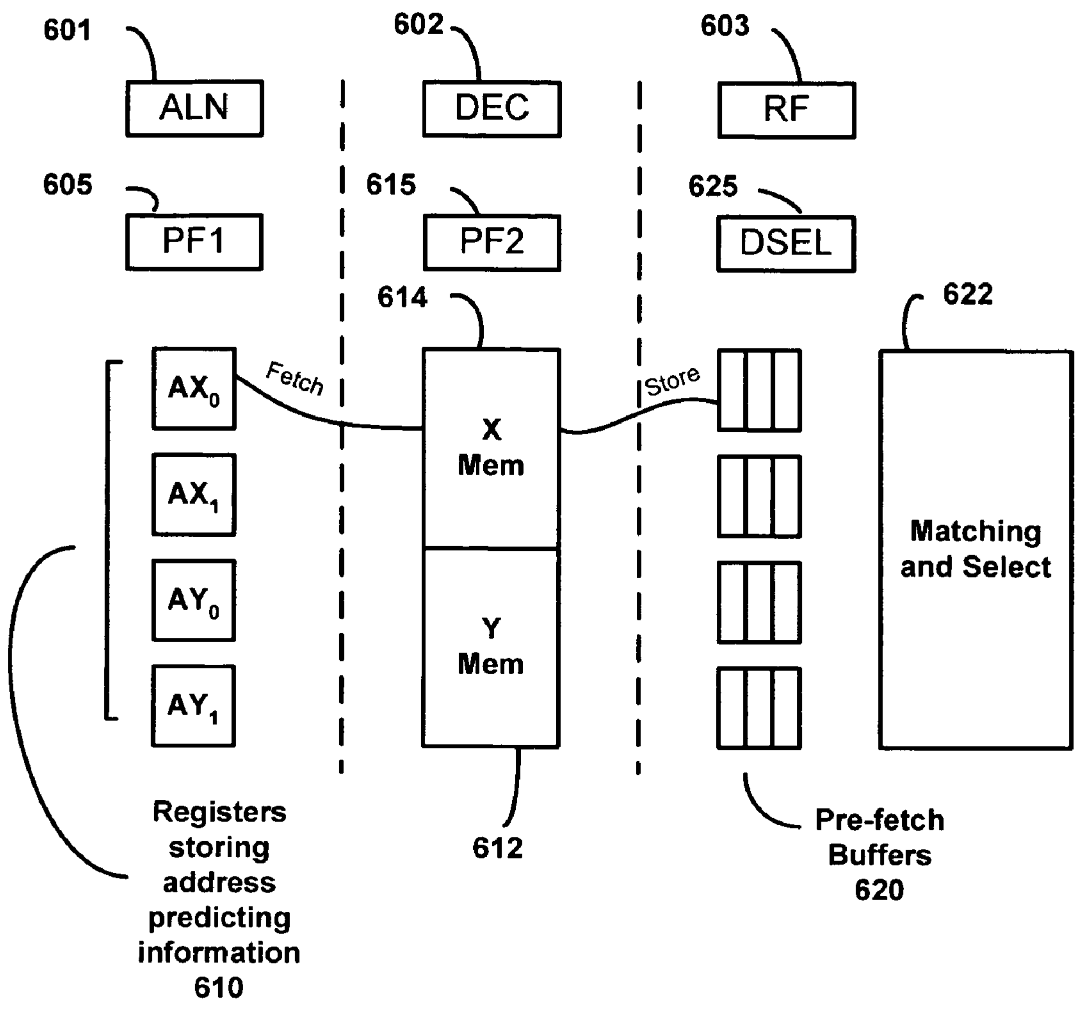
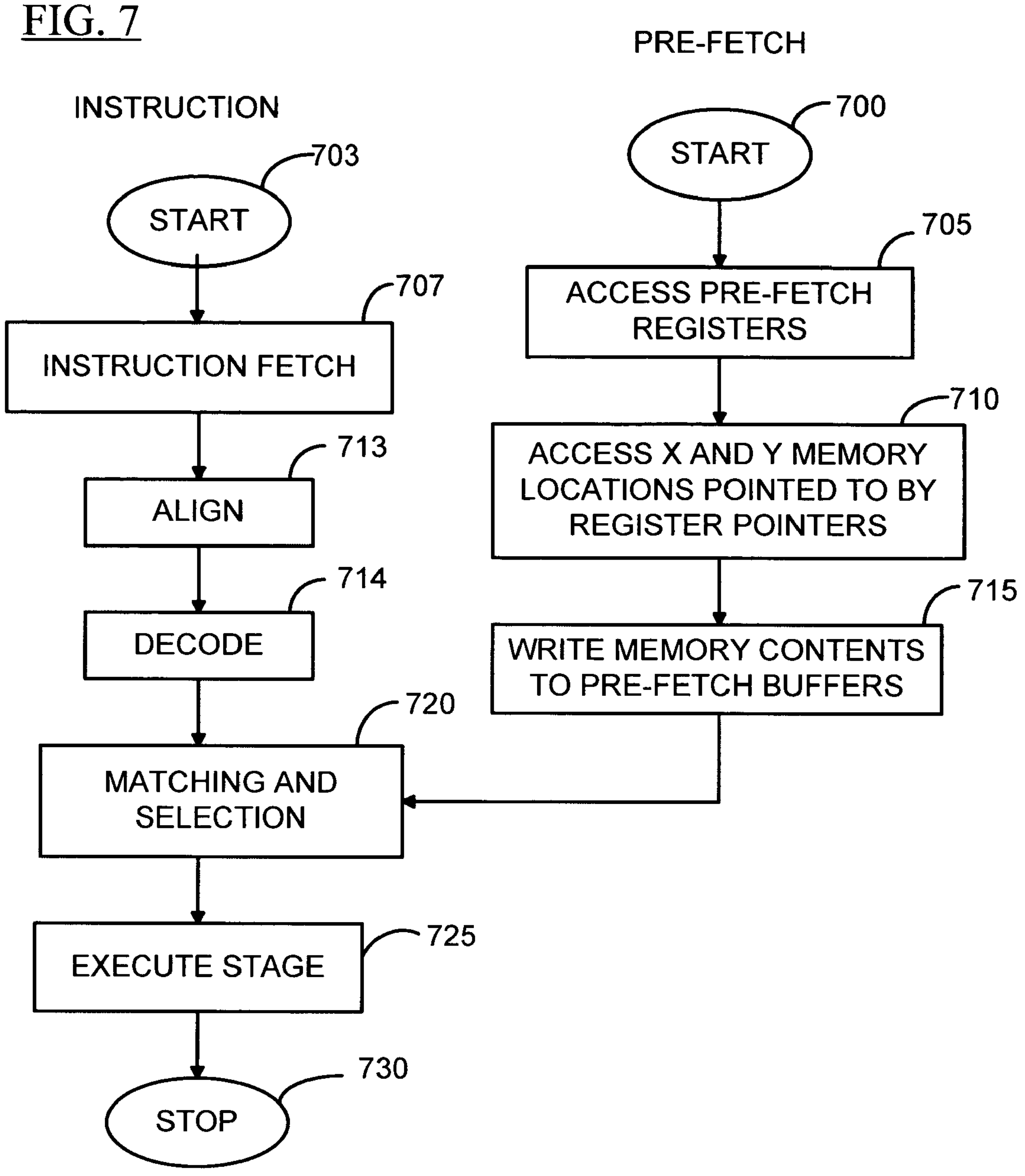

601
ALN
602
DEC
603
RF
605
PF1
615
PF2
625
DSEL
614
622
AX_0
AX_1
AY_0
AY_1
Fetch
X Mem
Y Mem
Store
Matching and Select
Registers storing address predicting information 610
612
Pre-fetch Buffers 620

FIG. 7


PRE-FETCH
INSTRUCTION
START
700
703
ACCESS PRE-FETCH REGISTERS
705
INSTRUCTION FETCH
707
ACCESS X AND Y MEMORY LOCATIONS POINTED TO BY REGISTER POINTERS
710
ALIGN
713
DECODE
714
WRITE MEMORY CONTENTS TO PRE-FETCH BUFFERS
715
MATCHING AND SELECTION
720
EXECUTE STAGE
725
STOP
730

MICROPROCESSOR ARCHITECTURE INCLUDING ZERO IMPACT PREDICTIVE DATA PRE-FETCH MECHANISM FOR PIPELINE DATA MEMORY

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to provisional application No. 60/572,238 filed May 19, 2004, entitled "Microprocessor Architecture," hereby incorporated by reference in its entirety.

FIELD OF THE INVENTION

[0002] This invention relates generally to microprocessor architecture and more specifically to systems and methods for achieving improved performance through a predictive data pre-fetch mechanism for a pipeline data memory, including specifically XY-type data memory.

BACKGROUND OF THE INVENTION

[0003] Multistage pipeline microprocessor architecture is known in the art. A typical microprocessor pipeline consists of several stages of instruction handling hardware, wherein each rising pulse of a clock signal propagates instructions one stage further in the pipeline. Although the clock speed dictates the number of pipeline propagations per second, the effective operational speed of the processor is dependent partially upon the rate that instructions and operands are transferred between memory and the processor. For this reason, processors typically employ one or more relatively small cache memories built directly into the processor. Cache memory typically is an on-chip random access memory (RAM) used to store a copy of memory data in anticipation of future use by the processor. Typically, the cache is positioned between the processor and the main memory to intercept calls from the processor to the main memory. Access to cache memory is generally much faster than off-chip RAM. When data is needed that has previously been accessed, it can be retrieved directly from the cache rather than from the relatively slower off-chip RAM.

[0004] Generally, the microprocessor pipeline advances instructions on each clock signal pulse to subsequent pipeline stages. However, effective pipeline performance can be slower than that implied by the processor speed. Therefore, simply increasing microprocessor clock speed does not usually provide a corresponding increase in system performance. Accordingly, there is a need for a microprocessor architecture that enhances effective system performance through methods in addition to increased clock speed.

[0005] One method of doing this has been to employ X and Y memory structures in parallel to the microprocessor pipeline. The ARCtangent-A4™ and ARCtangent-A5™ line of embedded microprocessors designed and licensed by ARC International, Inc. of Hertfordshire, UK, (ARC) employ such an XY memory structure. XY memory was designed to facilitate executing compound instructions on a RISC architecture processor without interrupting the pipeline. XY memory is typically located in parallel to the main processor pipeline, after the instruction decode stage, but prior to the execute stage. After decoding an instruction, source data is fetched from XY memory using address pointers. This source data is then fed to the execution stage. In the exemplary ARC XY architecture the two X and Y memory structures source two operands and receive results in the same cycle. Data in the XY memory is indexed via pointers from address generators and supplied to the ARC CPU pipeline for processing by any ARC instruction. The memories are software-programmable to provide 32-bit, 16-bit, or dual 16-bit data to the pipeline.

[0006] It should be appreciated that the description herein of various advantages and disadvantages associated with known apparatus, methods, and materials is not intended to limit the scope of the invention to their exclusion. Indeed, various embodiments of the invention may include one or more of the known apparatus, methods, and materials without suffering from their disadvantages.

[0007] As background to the techniques discussed herein, the following references are incorporated herein by reference: U.S. Pat. No. 6,862,563 issued Mar. 1, 2005 entitled "Method And Apparatus For Managing The Configuration And Functionality Of A Semiconductor Design" (Hakewill et al.); U.S. Ser. No. 10/423,745 filed Apr. 25, 2003, entitled "Apparatus and Method for Managing Integrated Circuit Designs"; and U.S. Ser. No. 10/651,560 filed Aug. 29, 2003, entitled "Improved Computerized Extension Apparatus and Methods", all assigned to the assignee of the present invention.

SUMMARY OF THE INVENTION

[0008] Various embodiments of the invention may ameliorate or overcome one or more of the shortcomings of conventional microprocessor architecture through a predictively fetched XY memory scheme. In various embodiments, an XY memory structure is located in parallel to the instruction pipeline. In various embodiments, a speculative pre-fetching scheme is spread over several sections of the pipeline in order to maintain high processor clock speed. In order to prevent impact on clock speed, operands are speculatively pre-fetched from X and Y memory before the current instruction has even been decoded. In various exemplary embodiments, the speculative pre-fetching occurs in an alignment stage of the instruction pipeline. In various embodiments, speculative address calculation of operands also occurs in the alignment stage of the instruction pipeline. In various embodiments, the XY memory is accessed in the instruction decode stage based on the speculative address calculation of the pipeline, and the resolution of the predictive pre-fetching occurs in the register file stage of the pipeline. Because the actual decoded instruction is not available in the pipeline until after the decode stage, all pre-fetching is done without explicit knowledge of what the current instruction is while this instruction is being pushed out of the decode stage into the register file stage. Thus, in various embodiments, a comparison is made in the register file stage between the operands specified by the actual instruction and those predictively pre-fetched. The pre-fetched values that match are selected to be passed to the execute stage of the instruction pipeline. Therefore, in a microprocessor architecture employing such a scheme, data memory fetches, arithmetic operation and result write back can be performed using a single instruction without slowing down the instruction pipeline clock speed or stalling the pipeline, even at high processor clock frequencies.

[0009] At least one exemplary embodiment of the invention may provide a predictive pre-fetch XY memory pipeline for a microprocessor pipeline. The predictive pre-fetch XY memory pipeline according to this embodiment may comprise a first pre-fetch stage comprising a pre-fetch pointer address register file and X and Y address generators, a second pre-fetch stage comprising X and Y memory structures accessed using address pointers generated in the first pre-fetch stage, and third data select stage comprising at least one pre-fetch buffer in which speculative operand data and address information are stored.

[0010] At least one additional exemplary embodiment may provide a method of predictively pre-fetching operand address and data information for a instruction pipeline of a microprocessor. The method of predictively pre-fetching operand address and data information for a instruction pipeline of a microprocessor according to this embodiment may comprise, prior to decoding a current instruction in the pipeline, accessing a set of registers containing pointers to specific locations in pre-fetch memory structures, fetching operand data information from the specific locations in the pre-fetch memory structures, and storing the pointer and operand data information in at least one pre-fetch buffer.

[0011] Yet another exemplary embodiment of this invention may provide a microprocessor architecture. The microprocessor architecture according to this embodiment may comprise a multi-stage microprocessor pipeline, and a multi-stage pre-fetch memory pipeline in parallel to at least a portion of the instruction pipeline, wherein the pre-fetch pipeline comprises a first stage having a set of registers serving as pointers to specific pre-fetch memory locations, a second stage, having pre-fetch memory structures for storing predicted operand address information corresponding to operands in an un-decoded instruction in the microprocessor pipeline, and a third stage comprising at least one pre-fetch buffers, wherein said first, second and third stage respectively are parallel to, simultaneous to and in isolation of corresponding stages of the microprocessor pipeline.

[0012] Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] **FIG. 1** is a block diagram illustrating a processor core in accordance with at least one exemplary embodiment of this invention;

[0014] **FIG. 2** is a block diagram illustration a portion of an instruction pipeline of a microprocessor core architecture employing an XY memory structure and a typical multi-operand instruction processed by such an instruction pipeline in accordance with a conventional non-speculative XY memory;

[0015] **FIG. 3** is an exemplary instruction format for performing a multiply instruction on 2 operands and a memory write back with a single instruction in accordance with at least one embodiment of this invention;

[0016] **FIG. 4** is a block diagram illustrating a microprocessor instruction pipeline architecture including a parallel predictive pre-fetch XY memory pipeline in accordance with at least one embodiment of this invention;

[0017] **FIG. 5** is a block diagram, illustrating in greater detail the structure and operation of a predictively pre-fetching XY memory pipeline in accordance with at least one embodiment of this invention;

[0018] **FIG. 6** is a block diagram illustrating the specific pre-fetch operations in an XY memory structure in accordance with at least one embodiment of this invention; and

[0019] **FIG. 7** is a flow chart detailing the steps of a method for predictively pre-fetching instruction operand addresses in accordance with at least one embodiment of this invention.

DETAILED DESCRIPTION OF THE DISCLOSURE

[0020] The following description is intended to convey a thorough understanding of the invention by providing specific embodiments and details involving various aspects of a new and useful microprocessor architecture. It is understood, however, that the invention is not limited to these specific embodiments and details, which are exemplary only. It further is understood that one possessing ordinary skill in the art, in light of known systems and methods, would appreciate the use of the invention for its intended purposes and benefits in any number of alternative embodiments, depending upon specific design and other needs.

[0021] Discussion of the invention will now made by way of example in reference to the various drawing figures. **FIG. 1** illustrates in block diagram form, an architecture for a microprocessor core **100** and peripheral hardware structure in accordance with at least one exemplary embodiment of this invention. Several novel features will be apparent from **FIG. 1** which distinguish the illustrated microprocessor architecture from that of a conventional microprocessor architecture. Firstly, the exemplary microprocessor architecture of **FIG. 1** features a processor core **100** having a seven stage instruction pipeline. However, it should be appreciated that additional pipeline stages may also be present. An align stage **120** is shown in **FIG. 1** following the fetch stage **110**. Because the microprocessor core **100** shown in **FIG. 1** is operable to work with a variable bit-length instruction set, namely, 16-bits, 32-bits, 48-bits or 64-bits, the align stage **120** formats the words coming from the fetch stage **110** into the appropriate instructions. In various exemplary embodiments, instructions are fetched from memory in 32-bit words. Thus, when the fetch stage **110** fetches a 32-bit word at a specified fetch address, the entry at that fetch address may contain an aligned 16-bit or 32-bit instruction, an unaligned 16 bit instruction preceded by a portion of a previous instruction, or an unaligned portion of a larger instruction preceded by a portion of a previous instruction based on the actual instruction address. For example, a fetched word may have an instruction fetch address of Ox**4**, but an actual instruction address of Ox**6**. In various exemplary embodiments, the 32-bit word fetched from memory is passed to the align stage **120** where it is aligned into an complete instruction. In various exemplary embodiments, this alignment may include discarding superfluous 16-bit instructions or assembling unaligned 32-bit or larger instructions into a single instructions. After completely assembling the instruction, the N-bit instruction is forwarded to the decoder **130**.

[0022] Still referring to **FIG. 1**, an instruction extension interface **180** is also shown which permits interface of customized processor instructions that are used to complement the standard instruction set architecture of the microprocessor. Interfacing of these customized instructions occurs through a timing registered interface to the various stages of the microprocessor pipeline **100** in order to minimize the effect of critical path loading when attaching customized logic to a pre-existing processor pipeline. Specifically, a custom opcode slot is defined in the extensions instruction interface for the specific custom instruction in order for the microprocessor to correctly acknowledge the presence of a custom instruction **182** as well as the extraction of the source operand addresses that are used to index the register file **142**. The custom instruction flag interface **184** is used to allow the addition of custom instruction flags that are used by the microprocessor for conditional evaluation using either the standard condition code evaluators or custom extension condition code evaluators **184** in order to determine whether the instruction is executed or not based upon the condition evaluation result of the execute stage (EXEC) **150**. A custom ALU interface **186** permits user defined arithmetic and logical extension instructions the result of which are selected in the result select stage (SEL) **160**.

[**0023**] Referring now to **FIG. 2**, a block diagram illustrating a portion of an instruction pipeline of a microprocessor core architecture employing an XY memory structure and a typical multi-operand instruction processed by such an instruction pipeline in accordance with a conventional non-speculative XY memory is illustrated. XY-type data memory is known in the art. Typically, in a RISC processor, only one memory load or store can be effected per pipelined instruction. However, in some cases, in order to accelerate pipeline efficiency, i.e., the number of operations executed per clock, it is desirable to have a single instruction perform multiple operations. For example, a single instruction could perform a memory read, an arithmetic operation and a memory write operation. The ability to decode and execute these kind of compound instructions is particularly important for achieving high performance in Digital Signal Processor (DSP) operations. DSP operations typically involve repetitive calculations on large data sets, thus, high memory bandwidth is required. By using an XY-memory structure, up to 2×32-bits of source data memory read access , and 1×32-bits of destination data memory write access per clock cycle are possible, resulting in a very high data memory bandwidth. (For example, a 4.8 Gbytes/s memory bandwidth can be achieved based on 3 32-bit accesses, 2 read and 1 write, per instruction in a 400 MHz processor or 3*32 bits*400 MHz/sec=38.4 Gbit/s or 4.8 Gbyte/s.)

[**0024**] In typical XY memory implementation, data used for XY memory is fetched from memory using addresses that are selected using register numbers decoded from the instruction in the decode stage. This data is then fed back to the execution units in the processor pipeline. **FIG. 2** illustrates such an XY memory implementation. In **FIG. 2**, an instruction is fetched from memory in the fetch stage **210** and, in the next clock cycle is passed to the align stage **220**. In the align stage **220**, the instruction is formatted into proper form. For example, if in the fetch stage **210** a 32-bit word is fetched from memory with the fetch address **0**x**4**, but the actual instruction address is for the 16-bit word having instruction address **0**x**6**, the first 16 bits of 32-bit word are discarded. This properly formatted instruction is then passed to the decode stage **230**, where it is decoded into an actual instruction, for example, the decoded instruction **241** shown in **FIG. 2**. This decoded instruction is then passed to the register file stage **240**.

[**0025**] **FIG. 2** illustrates the format of such a decoded instruction **241**. The instruction is comprised of a name (any arbitrary name used to reference the instruction), the destination address pointer and update mode, the first source address pointer and update mode, and the second source address pointer and update mode. In the register file stage **240**, from the decoded instruction **241**, the address of the source and destination operands are selected using the register numbers (windowing registers) as pointers to a set of address registers **242**. The source addresses are then used to access X memory **243** and Y memory **244**. Thus, between the decode stage **230** and the execute stage **250**, the address to use for access needs to be selected, the memory access performed, and the data selected fed to the execution stage **250**. As microprocessor clock speeds increases, it becomes difficult, if not impossible, to perform all these steps in a single clock cycle. As a result, either a decrease in the processor clock frequency must occur to accommodate these extra steps, or multiple clock cycles for each instruction using XY memory must be used, both of which negate or at least reduce the benefits of using XY memory in the first place.

[**0026**] One method of solving this problem is extending the processor pipeline to add more pipeline stages between the decode and the execution stage. However, extra processor stages are undesirable for several reasons. Firstly, they complicate the architecture of the processor. Secondly, any penalties from incorrect predictions in the branch prediction stage will be increased. Finally, because XY memory functions may only be needed when certain applications are being run on the processor, extra pipeline stages will necessarily be present even when these applications are not being used.

[**0027**] An alternative approach is to move the XY memory to an earlier stage of the instruction pipeline, ahead of the register file stage, to allow for more cycle time for the data selection. However, doing so may result in the complication that, when XY memory is moved into the decode stage, the windowing register number is not yet decoded before accessing memory.

[**0028**] Therefore, in accordance with at least one embodiment of this invention, to overcome these problems, the source data is predictively pre-fetched and stored for use in data buffers. When the source data from X or Y memory is required, just before the execution stage, a comparison may be made to check if the desired data was already pre-fetched, and if so, the data is simply taken from the pre-fetched data buffer and used. If it has not been pre-fetched, then the instruction is stalled and the required data is fetched. In order to reduce the number of instructions that are stalled, it is essential to ensure that data is pre-fetched correctly most of the time. Two schemes may be used to assist in this function. Firstly, a predictable way of using windowing registers may be employed. For example, if the same set of N windowing registers are used most of the time, and each pointer address is incremented in a regular way (sequentially as selected by the windowing registers), then the next few data for each of these N windowing registers can be pre-fetched fairly accurately. This reduces the number of prediction failures.

[0029] Secondly, by having more prediction data buffers, more predictive fetches can be made in advance, reducing the chance of a prediction miss. Because compound instructions also include updating addresses, these addresses must also be predictively updated. In general, address updates are predictable as long as the user uses the same modifiers along with its associated non-modify mode in a sequence of code and the user sticks to a set of N pointers for an implementation with N pre-fetch data buffers. Since the data is pre-fetched, the pre-fetched data can become outdated due to write-backs to XY memory. In cases such as this, the specific pre-fetch buffer can be flushed and the out-of-date data re-fetched, or, alternatively, data forwarding can be performed to update these buffers.

[0030] **FIG. 3** illustrates the format of a compound instruction, such as an instruction that might be used in a DSP application that would require extendible processing functions including XY memory in accordance with various embodiments of this invention. The compound instruction **300** consists of four sub-components, the name of the instruction **301**, the destination pointer **302**, the first operand pointer **303** and the second operand pointer **304**. In the instruction **300** shown in **FIG. 3**, the instruction, Muldw, is a dual 16-bit multiply instruction. The destination pointer **302** specifies that the result of the calculation instruction is to be written to X memory using the pointer address AX1. The label u**0** specifies the update mode. This is a user defined address update mode and must be specified before calling the extendible function. The source operand pointers **303** and **304**, specify that the first operand is to be read from X memory using the pointer address AX**0** and updated using update mode u**1** and the second operand is to be read from Y memory using the pointer address AY**0** and the update mode u**0**.

[0031] **FIG. 4** is a block diagram illustrating a microprocessor instruction pipeline architecture including a parallel predictive pre-fetch XY memory pipeline in accordance with at least one embodiment of this invention. In the example illustrated in **FIG. 4**, the instruction pipeline is comprised of seven stages, FCH **401**, ALN **402**, DEC **403**, RF **04**, EX **405**, SEL **406** and WB **407**. As stated above, each rising pulse of the clock cycle propagates an instruction to the next stage of the instruction pipeline. In parallel to the instruction pipeline is the predictive pre-fetch XY memory pipeline comprised of 6 stages including PF**1412**, PF**2413**, DSEL **414**, P**0415**, P**1416** and C **417**. It should be appreciated that various embodiments may utilize more or less pipeline stages. In various exemplary embodiments, speculative pre-fetching may begin in stage PF**1412**. However, in various exemplary embodiments, pre-fetching does not have to begin at the same time as the fetch instruction **401**. Pre-fetching can happen much earlier, for example, when a pointer is first set-up, or was already fetched because it was recently used. Pre-fetching can also happen later if the pre-fetched instruction was predicted incorrectly. The two previous stages PF**1412** and PF**2413**, prior to the register file stage **404**, allow sufficient time for the access address to be selected, the memory access performed, and the data selected to be fed to the execution stage **405**.

[0032] **FIG. 5**, is a block diagram, illustrating in greater detail the structure and operation of a predictively prefetching XY memory pipeline in accordance with at least one embodiment of this invention. In **FIG. 5**, **6** pipeline stages of the predictive pre-fetch XY memory pipeline are illustrated. As noted here, it should be appreciated that in various embodiments, more or less stages may be employed. As stated above in the context of **FIG. 4**, these stages may include the PF**1500**, PF**2510**, DSEL (data select) **520**, P**0530**, P**1540** and C **550**. Stage PF**1500**, which occurs simultaneous to the align stage of the instruction pipeline, includes the pre-fetch shadow pointer address register file **502** and the X and Y address generators (used to update the pointer address) **504** and **506**. Next, stage PF2, includes access to X memory unit **512** and Y memory unit **514**, using the pointers **504** and **506** in stage PF**1500**. In stage DSEL **520**, the data accessed from X memory **512** and Y memory **514** in stage PF**2510** are written to one of multiple pre-fetch buffers **522**. For purposes of example only, four pre-fetch buffers **522** are illustrated in **FIG. 5**. In various embodiments, multiple queue-like pre-fetch buffers will be used. It should be noted that typically each queue is associated to any pointer, but each pointer associated with at most one queue. In the DSEL stage **520**, the pre-fetched data is reconciled with the pointer of the operands contained in the actual instruction forwarded from the decode stage. If the actual data have been pre-fetched, they are passed to the appropriate execute unit in the execute stage.

[0033] P**0530**, P**1540** and C **550** stages are used to continue to pass down the source address and destination address (destination address is selected in DSEL stage) so that when they reach the C **550** stage, they update the actual pointer address registers, and the destination address is also used for writing the results of execution (if required, as specified by the instruction) back to XY memory. The address registers in PF**1500** stage are only shadowing address registers which are predictively updated when required. These values only become committed at the C stage **550**. Pre-fetch hazard detection performs the task of matching the addresses used in PF**1500** and PF**2510** stages to the destination addresses in DSEL **520**, P**0530**, P**1540**, and C **550** stage, so that if there is a write to a location in memory that is to be pre-fetched, the pre-fetch is stalled until, or restarted when, this Read after Write hazard has disappeared. A pre-fetch hazard can also occur when there is a write to a location in memory that has already been prefetched and stored in the buffer in DSEL stage. In this case, the item in the buffer is flushed and refetched when the write operation is complete

[0034] **FIG. 6** is a block diagram illustrating the specific structure of the pre-fetch logic in an XY memory structure in accordance with at least one embodiment of this invention. In various exemplary embodiments, in the PF1 stage **605**, speculative pre-fetch is performed by accessing a set of registers **610** that serve as pointers pointing to specific locations in the X and Y memories **614** and **612**. In the PF2 stage **602**, the data is fetched from the XY memory and then on the next clock pulse, the speculative operand data and address information is stored in pre-fetch buffers **620**. While still in the DSEL stage which also corresponds with the processor's Register File stage **603**, matching and select block **622** checks for the pre-fetched addresses. If the required operand addresses from the decoded instruction are in the pre-fetch buffers, they are selected and registered for use in the execution stage. In various exemplary embodiments, the pre-fetch buffers may be one, two, three or more deep such that a first in, first out storing scheme is used. When a data item is read out of one of the pre-fetch buffers

620, it no longer resides in the buffer. The next data in the FIFO buffer automatically moves to the front of the queue.

[**0035**] Referring now to **FIG. 7**, a flow chart detailing the steps of a method for predictively pre-fetching instruction operand addresses in accordance with at least one embodiment of this invention is depicted. In **FIG. 7**, the steps of a pre-fetch method as well as the steps of a typical instruction pipeline are illustrated in parallel. The individual steps of the pre-fetch method may occur at the same time as the various steps or even before.

[**0036**] Any correspondence between steps of the pre-fetch process and the instruction pipeline process implied by the figure are merely for ease of illustration. It should be appreciated that the steps of the pre-fetch method occur in isolation of the steps of the instruction pipeline method until matching and selection.

[**0037**] With continued reference to **FIG. 7**, operation of the pre-fetch method begins in step **700** and proceeds to step **705** where a set of registers are accessed that serve as pointers pointing to specific locations in the X and Y memory structures. In various embodiments, step **705** may occur simultaneous to a compound instruction entering the fetch stage of the microprocessor's instruction pipeline. However, as noted herein, in various other embodiments, because the actual compound instruction has not yet been decoded, and therefore, the pre-fetch process is not based on any information in the instruction this may occur before, an instruction is fetched in step **707**. Alternatively, step **705** may occur after a compound instruction is pre-fetched but prior to decoding.

[**0038**] As used herein, a compound instruction is one that performs multiple steps, such as, for example, a memory read, an arithmetic operation and a memory write.

[**0039**] With continued reference to the method of **FIG. 7**, in step **710**, the X and Y memory structures are accessed at locations specified by the pointers in the pre-fetch registers.

[**0040**] Operation of the method then goes to step **715** where the data read from the X and Y memory locations are written to pre-fetch buffers.

[**0041**] Next, in step **720**, the results of the pre-fetch method are matched with the actual decoded instruction in the matching and selection step. Matching and selection is performed to reconcile the addresses of the operands contained in the actual instruction forwarded from the decode stage of the instruction pipeline with the pre-fetched data in the pre-fetch buffers. If the pre-fetched data is correct, operation continues to the appropriate execute unit of the execute pipeline in step **725** depending upon the nature of the instruction, i.e., shift, add, etc. It should be appreciated that if the pre-fetched operand addresses are not correct, a pipeline flush will occur while actual operands are fetched and injected into pipeline. Operation of the pre-fetch method terminates after matching and selection. It should be appreciated that if necessary, that is, if the instruction requires a write operation to X Y memory, the results of execution are written back to XY memory. Furthermore, it should be appreciated that because steps **700**-**715** are performed in parallel and isolation to the processor pipeline operations **703**-**720** that they do not effect or otherwise delay the processor pipeline operations of fetching, aligning, decoding, register file or execution.

[**0042**] As stated above, when performing repetitive functions, such as DSP extension functions where data is repeatedly read from and written to XY memory, predictive pre-fetching is an effective means of taking advantage of the benefits of XY memory without impacting the instruction pipeline. Processor clock frequency may be maintained at high speeds despite the use of XY memory. Also, when applications being run on the microprocessor do not require XY memory, the XY memory functionality is completely transparent to the applications. Normal instruction pipeline flow and branch prediction are completely unaffected by this XY memory functionality both when it is invoked and when it is not used. The auxiliary unit of the execute branch provides an interface for applications to select this extendible functionality. Therefore, as a result of the above-described microprocessor architecture, with careful use of pointers and their associated update modes, operands can be predictively pre-fetched with sufficient accuracy to outweigh the overhead associated with mispredictions and without any impact on the processor pipeline.

[**0043**] It should be appreciated that, while the descriptors "X" and "Y" have been used throughout the specification that theses terms are purely descriptive to the extent that they do not imply any specific structural. That is to say that any two dimensional pre-fetch memory structure can be considered "X Y memory."

[**0044**] While the foregoing description includes many details and specificities, it is to be understood that these have been included for purposes of explanation only. The embodiments of the present invention are not to be limited in scope by the specific embodiments described herein. For example, although many of the embodiments disclosed herein have been described with reference to particular embodiments, the principles herein are equally applicable to microprocessors in general. Indeed, various modifications of the embodiments of the present inventions, in addition to those described herein, will be apparent to those of ordinary skill in the art from the foregoing description and accompanying drawings. Thus, such modifications are intended to fall within the scope of the following appended claims. Further, although the embodiments of the present inventions have been described herein in the context of a particular implementation in a particular environment for a particular purpose, those of ordinary skill in the art will recognize that its usefulness is not limited thereto and that the embodiments of the present inventions can be beneficially implemented in any number of environments for any number of purposes. Accordingly, the claims set forth below should be construed in view of the full breadth and spirit of the embodiments of the present inventions as disclosed herein.

1. A microprocessor comprising:

a multistage instruction pipeline; and

a predictive pre-fetch memory pipeline comprising:

a first pre-fetch stage comprising a pre-fetch pointer address register file and memory address generators;

a second pre-fetch stage comprising pre-fetch memory structures accessed using address pointers generated in the first pre-fetch stage; and a data select stage comprising at least one pre-fetch buffer in which predictive operand address and data information from the pre-fetch memory structures are stored.

2. The microprocessor of claim 1, wherein the pre-fetch memory structures comprise X and Y memory structures storing operand address data.

3. The microprocessor of claim 1, wherein the first and second pre-fetch stages and the data select stage occur in parallel to stages preceding an execute stage of the instruction pipeline.

4. The microprocessor of claim 1, wherein the instruction pipeline comprises align, decode and register file stages, and the first and second and pre-fetch stages and the data select stage occur in parallel to the align, decode and register file stages, respectively.

5. The microprocessor of claim 1, wherein the predictive pre-fetch memory pipeline further comprises hardware logic in the data select stage adapted to reconcile actual operand address information contained in an actual decoded instruction with the predictive operand address information the predictive operand address information.

6. The microprocessor of claim 5, wherein the predictive pre-fetch memory pipeline further comprises hardware logic adapted to pass the predictive operand address information from the pre-fetch buffer to an execute stage of the instruction pipeline if the actual operand address information matches the predictive operand address information.

7. The microprocessor of claim 1, wherein the predictive pre-fetch memory pipeline further comprises a write back structure invoked after the execute stage and being adapted to write the results of execution back to XY memory if the instruction requires a write to at least one of the pre-fetch memory structures.

8. A method of predictively pre-fetching operand address and data information for an instruction pipeline of a microprocessor, the method comprising:

prior to decoding a current instruction in the instruction pipeline, accessing at least one register containing pointers to specific locations in pre-fetch memory structures;

fetching predictive operand data from the specific locations in the pre-fetch memory structures; and

storing the pointer and predictive operand data in at least one pre-fetch buffer.

9. The method according to claim 8, wherein accessing, fetching and storing occur in parallel to, simultaneous to and in isolation of the instruction pipeline.

10. The method according to claim 9, wherein accessing, fetching and storing occur, respectively, in parallel to align, decode and register file stages of the instruction pipeline.

11. The method according to claim 8, further comprising, after decoding the current instruction, reconciling actual operand data contained in the decoded current instruction with the predictive operand data.

12. The method according to claim 8, further comprising decoding the current instruction and passing the pre-fetched predictive operand data to an execute unit of the microprocessor pipeline if the pre-fetched predictive operand data matches actual operand data contained in the current instruction.

13. The method according to claim 8, wherein accessing, fetching and storing are performed on successive clock pulses of the microprocessor.

14. The method according to claim 8 further comprising, performing pre-fetch hazard detection.

15. The method according to claim 14, wherein performing pre-fetch hazard detection comprises at least one operation selected from the group consisting of: stalling pre-fetch operation or restarting pre-fetch operation when the read after write hazard has disappeared, if it is determined that there is a read after write hazard characterized by a memory write to a location in memory that is to be pre-fetched; and clearing the pre-fetch buffers if there is a read from a memory location previously pre-fetched.

16. A microprocessor comprising:

a multistage microprocessor pipeline; and

a multistage pre-fetch memory pipeline in parallel to at least a portion of the microprocessor pipeline, wherein the pre-fetch memory pipeline comprises:

a first stage having at least one register serving as pointers to specific pre-fetch memory locations;

a second stage, having pre-fetch memory structures for storing predicted operand address information corresponding to operands in a pre-decoded instruction in the microprocessor pipeline; and

a third stage comprising at least one pre-fetch buffer;

wherein said first, second and third stages, respectively, are parallel to, simultaneous to and in isolation of corresponding stages of the microprocessor pipeline.

17. The microprocessor according to claim 16, wherein the microprocessor pipeline comprises align, decode, and register file stages, and the first, second and third stages of the pre-fetch memory pipeline, respectively, are parallel to the align, decode and register file stages.

18. The microprocessor according to claim 16, further comprising hardware logic in the third stage adapted to reconcile operand address information contained in an actual instruction forwarded from a decode stage of the microprocessor pipeline with the predicted operand address information.

19. The microprocessor according to claim 16, further comprising circuitry adapted to passing the predicted operand address information from the pre-fetch buffer to an execute stage of the microprocessor pipeline if the operand pointer in the actual instruction matches the predicted operand address information.

20. The microprocessor according to claim 16, further comprising post-execute stage hardware logic adapted to write the results of execution back to pre-fetch memory if a decoded instruction specifies a write back to at least one pre-fetch memory structure.

* * * * *