PROCESSOR ORDERING CONSISTENCY FOR A PROCESSOR PERFORMING OUT-OF-ORDER INSTRUCTION EXECUTION

BACKGROUND OF THE INVENTION

1. FIELD OF THE INVENTION:

The present invention pertains to the field of computer systems. More particularly, this invention relates to maintaining processor ordering consistency for a processor employing out of order instruction execution in a multiprocessor computer system.

2. BACKGROUND:

Inter-processor communication in a typical multiprocessor computer system is modeled as information transfer between one or more producer processor and one or more consumer processors. In a typical multiprocessor computer system, the producer processor transfers information to the consumer processors via messages stored in a shared memory subsystem.

Each processor in such a multiprocessor system usually must conform to a common processor ordering model to ensure consistent information flow to the consumer processors through the memory subsystem. A processor ordering model requires that each consumer processor observe stores from the producer processor in the same order.

For example, in a common inter-processor communication transaction, the producer processor writes message data to the memory subsystem, and then sets a message flag in the memory subsystem to indicate valid message data. Each consumer processor reads the message flag, and

then reads the message data if the message flag indicates valid message data. A processor ordering model that requires each consumer processor to observe stores from the producer processor in the same order ensures that each consumer processor observes the message data store before the message flag store. Such a processor ordering model causes each consumer processor to read valid message data if the message data store occurs before the message flag store.

Typical prior processors in a multiprocessor system implement in-order instruction execution pipelines. Such an in-order processor usually fetches an instruction stream, executes the instruction stream in a sequential program order, and dispatches loads and stores in the sequential program order. Such in-order processing of the instruction stream ensures that each consumer processor in the multiprocessor system observe stores from the producer processor in the same order because each consumer processor executes load instructions to read the message flag and the message data in the same order.

A processor may implement an out of order instruction execution pipeline to improve instruction execution performance. Such an out of order processor fetches an instruction stream and executes ready instructions in the instruction stream ahead of earlier instructions that are not ready. A ready instruction is typically an instruction having fully assembled source data and having available execution resources.

Such out of order execution improves processor performance because the instruction execution pipeline of the processor does not stall while

assembling source data for a non ready instruction. For example, a non ready instruction awaiting source data from a floating-point operation does not stall the execution of later instructions in the instruction stream that are ready to execute.

A processor that implements an out of order instruction execution pipeline generates out of order result data because the instructions in the instruction stream are executed out of order. An out of order processor may implement a reorder register file to impose the original program order on the result data after instruction execution.

Out-of-order instruction execution by processors in a multiprocessor system may cause violations of the processor ordering model. The consumer processors that execute load instructions out of order may observe stores from the producer processor in differing order.

For example, a consumer processor that executes a load instruction for the message flag before a load instruction for the message data effectively observes the producer processor stores to the message data and the message flag in a different order than a consumer processor that executes a load instruction for the message data before a load instruction for the message flag.

Such a violation of the processor ordering model may cause the consumer processors to read differing message data. One of the consumer processors may load the message data before the producer processor stores the message data, and may load the message flag after the producer processor stores the message flag. In such a case, the consumer processor loads invalid

message data and loads a message flag indicating valid message data. As a consequence, the consumer processor erroneously processes invalid message data.

SUMMARY AND OBJECTS OF THE INVENTION

One object of the present invention is to maintain processor ordering in a multiprocessor computer system for a processor having an out of order instruction execution pipeline.

Another object of the present invention is to maintain processor ordering for a processor having an out of order instruction execution pipeline, wherein each consumer processor in the multiprocessor computer system is required to observe memory stores from a producer processor in the same order.

A further object of the present invention is to maintain processor ordering in a multiprocessor computer system for a processor having an out of order instruction execution pipeline by detecting external memory store operations targeted for a memory address corresponding to an executed and unretired load memory instruction.

These and other objects of the invention are provided by a method for processor ordering in a multiprocessor computer system. A processor having an out of order instruction execution pipeline fetches an instruction stream from an external memory in a sequential program order. The instruction stream includes load memory instructions, wherein each load memory instruction specifies a load memory operation from a memory address over a multiprocessor bus of the multiprocessor computer system.

The processor assembles at least one source data value for each load memory instruction, such that the source data value specifies the memory address for the corresponding load memory instruction. The processor executes each load memory instruction after the corresponding source data value is assembled regardless of the sequential program order of the load memory instruction. Each executed load memory instruction generates a result data value.

The processor snoops the multiprocessor bus for an external store operation to the memory address of each executed load memory instruction. The processor commits the result data value of each executed load memory instruction to an architectural state in the sequential program order if the external store operation to the memory address of the executed load memory instruction is not detected.

The processor discards the result data value of each executed load memory instruction if the external store operation to the memory address of the executed load memory instruction is detected before the result data value is committed to the architectural state. The processor then reexecutes the instruction stream starting at the load memory instruction corresponding to the discarded result data value.

Other objects, features and advantages of the present invention will be apparent from the accompanying drawings, and from the detailed description that follows below.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings in which like references indicate similar elements, and in which:

Figure 1 illustrates a multiprocessor computer system comprising a set of processors and a memory subsystem;

Figure 2 is a block diagram of a processor in the multiprocessor computer system;

Figure 3 illustrates the functions of the register alias circuit which converts the logical micro-ops into corresponding physical micro-ops by mapping the logical sources and destinations into physical sources and destinations;

Figure 4 illustrates the reorder circuit which contains a reorder buffer comprising a set of ROB entries (RE0 through REn) that buffer speculative result data from the out of order speculative execution of physical micro-ops;

Figure 5 illustrates the reservation and dispatch circuit which contains a reservation dispatch table comprising a set of reservation station entries RS0 through RSx for assembling and dispatching micro-ops;

Figure 6 illustrates the real register circuit which contains a set of committed state registers that buffer committed result data values;

Figure 7 illustrates a load memory circuit which comprises an address generation circuit, a memory ordering circuit, a data translate look-aside buffer (DTLB) circuit, and a data cache circuit;

Figure 8 illustrates the memory ordering circuit which contains a load buffer comprising a set of load buffer entries LB0 through LBn;

Figure 9 illustrates the snoop detection circuitry in the memory ordering circuit which includes a snoop detect circuit corresponding to each load buffer entry LB0-LBn;

Figure 10 illustrates notification circuitry in the memory ordering circuit that generates the memory ordering restart signals;

Figure 11 illustrates processing of a load micro-op ld 0x100, EBX, EAX issued by the instruction fetch and micro-op issue circuit;

Figure 12 illustrates the dispatch and retirement of the linear load memory micro-op ld 32100, 42, lbid = 4 corresponding to the load micro-op ld 0x100, EBX, EAX.

DETAILED DESCRIPTION

Figure 1 illustrates a multiprocessor computer system 20. The multiprocessor computer system 20 comprises a set of processors 22 - 24, and a memory subsystem 26. The processors 22 - 24 and the memory subsystem 26 communicate over a multiprocessor bus 28.

Each processor 22 - 24 fetches a stream of macro instructions from the memory subsystem 26 over the multiprocessor bus 28. Each processor 22 - 24 executes the corresponding stream of macro instructions and maintains data storage in the memory subsystem 26.

Figure 2 illustrates the processor 22. The processor 22 comprises a front-end section including a bus interface circuit 30 and an instruction fetch and micro-op issue circuit 32. The processor 22 also comprises a register renaming section including a register alias circuit 34 and an allocator circuit 36. The processor 22 also comprises an out of order execution section comprising a reservation and dispatch circuit 38, an execution circuit 40, a reorder circuit 42, and a real register circuit 44.

The bus interface circuit 30 enables transfer of address, data and control information over the multiprocessor bus 28. The instruction fetch and micro-op issue circuit 32 fetches a stream of macro instructions from the memory subsystem 26 over the multiprocessor bus 28 through the bus interface circuit 30. The instruction fetch and micro-op issue circuit 32 implements speculative branch prediction to maximize macro-instruction fetch throughput.

For one embodiment the stream of macro instructions fetched over the multiprocessor bus 28 comprises a stream of Intel Architecture Microprocessor macro instructions. The Intel Architecture Microprocessor macro instructions operate on a set of architectural registers, including an EAX register, an EBX register, an ECX register, and an EDX register, etc.

The instruction fetch and micro-op issue circuit 32 converts the macro-instruction of the incoming stream of macro instructions into an in-order stream of logical micro operations, hereinafter referred to as logical micro-ops. The instruction fetch and micro-op issue circuit 32 generates one or more logical micro ops for each incoming macro instruction. The logical micro-ops corresponding to each macro instruction are reduced instruction set micro operations that perform the function of the corresponding macro instruction. The logical micro-op specify arithmetic and logical operations as well as load and store operations to the memory subsystem 26.

The instruction fetch and micro-op issue circuit 32 transfers the in-order stream of logical micro-ops to the register alias circuit 34 and the allocator circuit 36 over a logical micro-op bus 50. For one embodiment, the instruction fetch and micro-op issue circuit 32 issues up to four in-order logical micro-ops during each clock cycle of the processor 22. Alternatively, the in-order logical micro-ops may be limited to four during each clock cycle to minimize integrated circuit die area for the processor 22.

The instruction fetch and micro-op issue circuit 32 contains a micro instruction sequencer and an associated control store. The micro instruction

sequencer implements micro programs for performing a variety of functions for the processor 22, including fault recovery functions and processor ordering functions.

Each logical micro-op generated by the instruction fetch and micro-op issue circuit 32 comprises an op code, a pair of logical sources and a logical destination. Each logical source may specify a register or provide an immediate data value. The register logical sources and the logical destinations of the logical micro-ops specify architectural registers of the original macro instructions. In addition, the register logical sources and the logical destinations of the logical micro-ops specify temporary registers for microcode implemented by the micro instruction sequencer of the instruction fetch and micro-op issue circuit 32.

The register alias circuit 34 receives the in-order logical micro-ops over the logical micro-op bus 50, and generates a corresponding set of in-order physical micro-ops by renaming the logical sources and logical destinations of the logical micro-ops. The register alias circuit 34 receives the in-order logical micro-ops over the logical micro-op bus 50, maps the logical sources and the logical destination of each logical micro-op into physical sources and a physical destination, and transfers the in-order physical micro-ops over a physical micro-op bus 52.

Each physical micro-op comprises the opcode of the corresponding logical micro-op, a pair of physical sources, and a physical destination. Each physical source may specify a physical register or provide an immediate data value. The register physical sources of the physical micro-ops specify physical

sequencer implements micro programs for performing a variety of functions for the processor 22, including fault recovery functions and processor ordering functions.

Each logical micro-op generated by the instruction fetch and micro-op issue circuit 32 comprises an op code, a pair of logical sources and a logical destination. Each logical source may specify a register or provide an immediate data value. The register logical sources and the logical destinations of the logical micro-ops specify architectural registers of the original macro instructions. In addition, the register logical sources and the logical destinations of the logical micro-ops specify temporary registers for microcode implemented by the micro instruction sequencer of the instruction fetch and micro-op issue circuit 32.

The register alias circuit 34 receives the in-order logical micro-ops over the logical micro-op bus 50, and generates a corresponding set of in-order physical micro-ops by renaming the logical sources and logical destinations of the logical micro-ops. The register alias circuit 34 receives the in-order logical micro-ops over the logical micro-op bus 50, maps the logical sources and the logical destination of each logical micro-op into physical sources and a physical destination, and transfers the in-order physical micro-ops over a physical micro-op bus 52.

Each physical micro-op comprises the opcode of the corresponding logical micro-op, a pair of physical sources, and a physical destination. Each physical source may specify a physical register or provide an immediate data value. The register physical sources of the physical micro-ops specify physical

This Sheet Has Been Replaced



The allocator circuit 36 allocates the physical registers of the reorder circuit 42 to the physical micro-ops in the same order that logical micro-ops are received over the logical micro-op bus 50. The allocator circuit 36 maintains an allocation pointer for allocating physical registers of the reorder circuit 42. The allocation pointer points to a next set of consecutive physical registers in the reorder circuit 42 for each set of logical micro-ops received over the logical micro-op bus 50. The ordering of the physical registers assigned to the physical micro-ops in the reorder circuit 42 reflects the ordering of the original logical micro-ops.

The allocator circuit 36 specifies the reservation station entries for the physical micro-ops on the physical micro-ops bus 52 by transferring reservation station entry select signals to the reservation and dispatch circuit 38 over a reservation station select bus 66.

The allocator circuit 36 assigns a load buffer entries to each physical micro-ops on the physical micro-ops bus 52 that specifies a load memory opcode. The allocator circuit 36 assigns the load buffer entries by transferring load buffer identifiers to the reservation and dispatch circuit 38 over a load buffer ID bus 72.

The reservation and dispatch circuit 38 holds the physical micro-ops awaiting execution by the execution circuit 40. The reservation and dispatch circuit 38 receives the in-order physical micro-ops over the physical micro-op bus 52, assembles the source data for the physical micro-ops, and dispatches the physical micro-ops to the execution circuit 40.

The reservation and dispatch circuit 38 receives the physical micro-ops over the physical micro-op bus 50 and stores the physical micro-ops in available reservation station entries. The reservation and dispatch circuit 38 assembles source data for the physical micro-ops, and dispatches the physical micro-ops to appropriate execution units in the execution circuit 40 when the source data is assembled.

The reservation and dispatch circuit 38 receives the source data for the pending physical micro-ops from the reorder circuit 42 and the real register circuit 44 over a source data bus 58. The reservation and dispatch circuit 38 also receives source data for the pending physical micro-ops from the execution circuit 40 over a result bus 62 during a write back of speculative results from the execution circuit 40 to the reorder circuit 42.

The reservation and dispatch circuit 38 schedules the physical micro-ops having completely assembled source data for execution. The reservation and dispatch circuit 38 dispatches the ready physical micro-ops to the execution circuit 40 over a micro-op dispatch bus 60. The reservation and dispatch circuit 38 schedules execution of physical micro-ops out of order according to the availability of the source data for the physical micro-ops, and according to the availability of execution unit resources in the execution circuit 40.

The execution circuit 40 writes back the speculative results from the out of order execution of the physical micro-ops to the reorder circuit 42 over the result bus 62. The writes back of speculative results by the execution circuit 40 is out of order due to the out of order dispatching of physical micro-

ops by the reservation and dispatch circuit 38 and the differing number of processor 22 cycles required for execution of the differing types of physical micro-ops.

For one embodiment, the execution circuit 40 comprises a set of five execution units EU0-EU4. The reservation and dispatch circuit 38 dispatches up to five physical micro-ops concurrently to the execution units EU0-EU4 over the micro-op dispatch bus 60.

The execution unit EU0 performs arithmetic logic unit (ALU) functions including integer multiply and divide as well as floating-point add, subtract, multiply and divide micro-ops. The execution unit EU1 performs ALU integer functions and jump operations. The execution unit EU2 performs integer and floating-point load operations from memory as well as load linear address functions and segment register operations. The execution unit EU3 performs integer and floating-point store and segmentation register operations. The execution unit EU4 performs integer and floating-point store data operations.

The reorder circuit 42 contains the physical registers that buffer speculative results for the physical micro-ops. Each physical register in the reorder circuit 42 accommodates integer data values and floating-point data values.

The real register circuit 44 contains committed state registers that correspond to the architectural registers of the original stream of macro-

instructions. Each committed state register in the real register circuit 44 accommodates integer data values and floating-point data values.

For one embodiment, the committed state registers of the real register circuit 44 comprise the EAX, EBX, ECX, and EDX registers, etc. of the Intel Architecture Microprocessor, as well as architectural flags for the Intel Architecture Microprocessor. The real register circuit 44 also contains committed state registers for the microcode registers used by microcode executing in the instruction fetch and micro-op issue circuit 32.

The reorder circuit 42 and the real register circuit 44 receive the physical micro-ops over the physical micro-op bus 52. The physical sources of the physical micro-ops specify physical registers in the reorder circuit 42 and committed state registers in the real register file 44 that hold the source data for the physical micro-ops.

The reorder circuit 42 and the real register circuit 44 read the source data specified by the physical sources, and transfer the source data to the reservation and dispatch circuit 38 over a source data bus 58. Each physical source of the physical micro-ops includes a real register file valid (RRFV) flag that indicates whether the source data is contained in a physical register in the reorder circuit 42 or a committed state register in the real register file 44.

The physical destinations of the physical micro-ops on the physical micro-op bus 52 specify physical registers in the reorder circuit 42 for buffering the speculative results of the out of order execution of the physical micro-ops. The reorder circuit 42 receives the physical destinations of the physical micro-

ops over the physical micro-op bus 52, and clears the physical registers specified by the physical destinations.

The reorder circuit 42 receives the logical destinations corresponding to the physical micro-ops over the logical destination bus 54, and stores the logical destinations into the physical registers specified by the physical destinations of the physical micro-ops. The logical destinations in the physical registers of the reorder circuit 42 specify committed state registers in the real register circuit 44 for retirement of the physical micro-ops.

A retire logic circuit 46 imposes order on the physical micro-ops by committing the speculative results held in the physical registers of the reorder circuit 42 to an architectural state in the same order as the original logical micro-ops were received. The retire logic circuit 46 causes transfer of the speculative result data in the reorder circuit 42 to corresponding committed state registers in the real register circuit 44 over a retirement bus 64. For one embodiment, the retire logic circuit 46 retires up to four physical registers during each cycle of the processor 22. For another embodiment, the retire logic circuit 46 retires up to three physical registers during each cycle of the processor 22 to minimize integrated circuit die space.

The retire logic circuit 46 also causes the reorder circuit 42 to transfer the macro instruction pointer delta values for the retiring physical micro-ops over a macro instruction pointer offset bus 120 during retirement.

The restart circuit 48 receives macro instruction pointer delta values over the macro instruction pointer offset bus 120. The restart circuit 48

calculates a committed instruction pointer value according to the macro instruction pointer deltas for the retiring ROB entries.

The retire logic circuit 46 maintains a retirement pointer to the physical registers in the reorder circuit 42. The retirement pointer points to sets of consecutive physical registers for retirement. The retirement pointer follows the allocation pointer through the physical registers in the reorder circuit 42 as the retire logic retires the speculative results of the physical registers to the committed state. The retire logic circuit 46 retires the physical registers in order because the physical registers were allocated to the physical micro-ops in order.

The retire logic circuit 46 broadcasts the retirement physical destinations specified by the retirement pointer over a retire notification bus 70. The memory ordering buffer in the execution circuit 40 receives the retirement physical destinations, and issues a set of memory ordering restart signals 76. The memory ordering restart signals 76 indicate whether a memory load operation corresponding to one of the retiring physical destinations has caused a possible processor ordering violation. The memory ordering restart signals 76 indicate which of the retiring physical destinations has caused the possible processor ordering violation.

The memory ordering restart signals 76 are received by the restart circuit 48. If the memory ordering restart signals 76 indicate a possible processor ordering violation, the restart circuit 48 issues a reorder clear signal 78. The reorder clear signal 78 causes the reorder circuit 42 to clear the speculative result data for the unretired physical micro-ops. The reorder clear

signal 78 causes the reservation and dispatch circuit 38 to clear the pending physical micro-ops that await dispatch to the execution circuit 40. The reorder clear signal 78 also causes the allocator circuit 36 to reset the allocation pointer for allocating the physical registers in the reorder circuit 42, and causes the retire logic circuit 46 to reset the retirement pointer for retiring the physical registers.

If the memory ordering restart signals 76 indicate a possible processor ordering violation, the restart circuit 48 uses the macro instruction pointer delta values received over the macro instruction pointer offset bus 120 to calculate a restart instruction pointer value. The restart instruction pointer value specifies the macro instruction corresponding to the physical micro-op that caused the possible memory ordering violation. The restart circuit 48 transfers the restart instruction pointer value to the instruction fetch and micro-op issue circuit 32 over a restart vector bus 122.

The instruction fetch and micro-op issue circuit 32 receives the restart instruction pointer value over a restart vector bus 122. The reorder clear signal 78 causes the micro-instruction sequencer of the instruction fetch and micro-op issue circuit 32 to reissue the in order stream of logical micro-ops that were cleared from the reorder circuit 42 before retirement. The instruction fetch and micro-op issue circuit 32 reissues the logical micro-ops by fetching a macro instruction stream starting at the macro instruction address specified by the restart instruction pointer value, and by converting the macro instruction stream into logical micro-ops, and by transferring the logical micro-ops over the logical micro-op bus 50.

If the memory ordering restart signals 76 do not indicate a possible processor ordering violation, then the retirement of the physical registers specified by the retiring physical destinations proceeds. The reorder circuit 42 tests the valid flags for the retiring physical destinations. The reorder circuit 42 retires the speculative result data for each retiring physical register if the valid flag of the retiring physical register indicates valid speculative data. The reorder circuit 42 retires a physical register by causing transfer of the speculative result data to the committed state registers in the real register circuit 44 specified by the logical destinations of the physical register.

The register alias circuit 34 and the allocator circuit 36 receive the retiring physical destinations over a retire notification bus 70. The register alias circuit 34 accordingly updates the register alias table to reflect the retirement. The allocator circuit 36 marks the retired physical registers in the reorder circuit 42 as available for allocation.

Figure 3 is a diagram that illustrates the functions of the register alias circuit 34. The register alias circuit 34 receives logical micro-ops in order over the logical micro-op bus 50, converts the logical micro-ops into corresponding physical micro-ops by mapping the logical sources and destinations into physical sources and destinations, and then transfers the physical micro-ops in order over the physical micro-op bus 52.

The register alias circuit 34 implements a register alias table 80. The register alias table 80 performs logical to physical register renaming by mapping the logical sources and destinations of the logical micro-ops to the physical sources and destinations of the corresponding physical micro-ops.

The physical sources and destinations of the physical micro-ops specify physical registers in the reorder circuit 42 and committed state registers in the real register circuit 44.

The entries in the register alias table 80 correspond to the architectural registers of the original macro instruction stream. For one embodiment, the EAX, EBX, ECX, and EDX entries of the register alias table 80 correspond to the EAX, EBX, ECX, and EDX registers of the Intel Architecture Microprocessor.

Each entry in the register alias table 80 contains a reorder buffer (ROB) pointer. The ROB pointer specifies a physical register in the reorder circuit 42 that holds the speculative result data for the corresponding architectural register. Each entry in the register alias table 80 also contains a real register file valid (RRFV) flag that indicates whether the speculative result data for the corresponding architectural register has been retired to the appropriate committed state register in the real register circuit 44.

The register alias circuit 34 receives a set of in order logical micro-ops (lmop_0 through lmop_3) over the logical micro-op bus 50. Each logical micro-op comprises an op code, a pair of logical sources lsrc1 and lsrc2, a logical destination ldst, and a macro instruction pointer delta mipd. The logical sources lsrc1 and lsrc2 and the logical destination ldst each specify an architectural register of the original stream of macro-instructions.

The register alias circuit 34 also receives a set of allocated physical destinations (alloc_pdst_0 through alloc_pdst_3) from the allocator circuit 36 over the physical destination bus 56. The physical destinations alloc_pdst_0

through alloc_pdst_3 specify newly allocated physical registers in the reorder circuit 42 for the logical micro-ops lmop_0 through lmop_3. The physical registers in the reorder circuit 42 specified by the physical destinations alloc_pdst_0 through alloc_pdst_3 will hold speculative result data for the physical micro-ops corresponding to the logical micro-ops lmop_0 through lmop_3.

The register alias circuit 34 transfers a set of in order physical micro-ops (pmop_0 through pmop_3) over the physical micro-op bus 52. Each physical micro-op comprises an op code, a pair of physical sources psrc1 and psrc2 and a physical destination pdst. The physical sources psrc1 and psrc2 each specify a physical register in the reorder circuit 42 or a committed state register in the real register circuit 44. The physical destination pdst specifies a physical register in the reorder circuit 42 to hold speculative result data for the corresponding physical micro-op.

The register alias circuit 34 generates the physical micro-ops pmop_0 through pmop_3 by mapping the logical sources of the logical micro-ops lmop_0 through lmop_3 to the physical registers of the reorder circuit 42 and the committed state registers specified of the real register circuit 44 as specified by the register alias table 80. The register alias circuit 34 merges the physical destinations alloc_pdst_0 through alloc_pdst_3 into the physical micro-ops pmop_0 through pmop_3.

The opcodes of the physical micro-ops pmop_0 through pmop_3 are the same as the corresponding opcodes of the logical micro-ops lmop_0

through lmop_3. For example, the register alias circuit 34 generates pmop_0 such that the op code of pmop_0 equals the opcode of lmop_0.

For example, the register alias circuit 34 generates the physical source psrc1 for the physical micro-op pmop_0 by reading the register alias table 80 entry specified by the logical source lsrc1 of the lmop_0. If the RRFV flag of the specified register alias table 80 entry is not set, then the register alias circuit 34 transfers the ROB pointer from the specified register alias table 80 entry along with the RRFV flag over the physical micro-op bus 52 as the physical source psrc1 for the pmop_0. If the RRFV bit is set, then the register alias circuit 34 transfers a pointer to the committed state register in the real register circuit 44 that corresponds to the logical source lsrc1 along with the RRFV flag over the physical micro-op bus 52 as the physical source psrc1 for the pmop_0.

The register alias circuit 34 generates the physical source psrc2 for the physical micro-op pmop_0 by reading the register alias table 80 entry that corresponds to the logical source lsrc2 of the lmop_0. If the RRFV flag is not set, then the register alias circuit 34 transfers the ROB pointer from the specified register alias table 80 entry along with the RRFV flag over the physical micro-op bus 52 as the physical source psrc2 for the pmop_0. If the RRFV bit is set, then the register alias circuit 34 transfers a pointer to the committed state register in the real register circuit 44 that corresponds to the logical source lsrc2 along with the RRFV flag over the physical micro-op bus 52 as the physical source psrc2 for the pmop_0.

The register alias circuit 34 stores the physical destination alloc_pdst_0 into the ROB pointer field of the register alias table 80 entry specified by the

logical destination ldst of the lmop_0, and clears the corresponding RRFV bit. The clear RRFV bit indicates that the current state of the corresponding architectural register is speculatively held in the physical register of the reorder circuit 42 specified by the corresponding ROB pointer.

The register alias circuit 34 transfers a set of logical destinations ldst_0 through ldst_3 and corresponding macro instruction pointer deltas mipd_0 through mipd_3 over the logical destination bus 54. The logical destinations ldst_0 through ldst_3 are the logical destinations ldst of the logical micro-ops lmop_0 through lmop_3.

The macro instruction pointer deltas mipd_0 through mipd_3 are the macro instruction pointer deltas mipd of the logical micro-ops lmop_0 through lmop_3. The macro instruction pointer delta mipd_0 is the logical destination ldst of the lmop_0, the macro instruction pointer delta mipd_1 is the logical destination ldst of the lmop_1, etc. The macro instruction pointer deltas mipd_0 through mipd_3 identify the original macro instructions corresponding to the physical micro-ops pmop_0 through pmop_3.

Figure 4 illustrates the reorder circuit 42. The reorder circuit 42 implements a reorder buffer 82 comprising a set of ROB entries (RE0 through REn). The ROB entries RE0 through REn are physical registers that buffer speculative result data from the out of order execution of physical micro-ops. For one embodiment, the ROB entries RE0 through REn comprise a set of 64 physical registers. For another embodiment, the ROB entries RE0 through REn comprise a set of 40 physical registers.

Each ROB entry comprises a valid flag (V), a result data value, a set of flags, a flag mask, a logical destination (LDST), fault data, and an instruction pointer delta (IPDELTA).

The valid flag indicates whether the result data value for the corresponding ROB entry is valid. The reorder circuit 42 clears the valid flag for each newly allocated ROB entry to indicate an invalid result data value. The reorder circuit 42 sets the valid flag when speculative result data is written back to the ROB entry from the execution circuit 40.

The result data value is a speculative result from the out of order execution of the corresponding physical micro-op. The result data value may be either an integer data value or a floating-point data value. For one embodiment, the result data value field of each ROB entry RE0 through REn comprises 86 bits to accommodate both integer and floating-point data values.

The flags and flag mask provide speculative architectural flag information. The speculative architectural flag information is transferred to the architectural flags of the real register circuit 44 upon retirement of the corresponding ROB entry.

The logical destination LDST specifies a committed state register in the real register circuit 44. The result data value of the corresponding ROB entry is transferred to the committed state register specified by LDST during retirement of the ROB entry.

The fault data contains fault information for the fault processing microcode executing in the instruction fetch and micro-op issue circuit 32. When a fault occurs, the fault handing microcode reads the fault data to determine the cause of the fault.

The IPDELTA is a macro instruction pointer delta value that identifies the macro instruction corresponding to the physical register.

The reorder circuit 42 receives the physical micro-ops pmop_0 through pmop_3 over the physical micro-op bus 52. The reorder circuit 42 reads the source data specified by the physical micro-ops pmop_0 through pmop_3 from the reorder buffer 82. The reorder circuit 42 transfers the result data values and the valid flags from the ROB entries specified by the physical sources psrc1 and psrc2 of the physical micro-ops to the reservation and dispatch circuit 38 over the source data bus 58.

For example, the result data values and valid flags from the ROB entries specified by the physical sources psrc1 and psrc2 of the pmop_0 are transferred as source data src1/src2 data_0 over the source data bus 58. The source data src1/src2 data_0 provides the source data specified by the physical sources psrc1 and psrc2 of the pmop_0 if the corresponding valid flags indicate valid source data.

Similarly, the reorder circuit 42 transfers the result data values and valid flags from the appropriate ROB entries as the source data src1/src2 data_1 through source data src1/src2 data_3 over the source data bus 58 for the physical micro-ops pmop_1 through pmop_3.

The reorder circuit 42 clears the valid bits of the ROB entries specified by the physical destinations pdst the physical micro-ops pmop_0 through pmop_3 received over the physical micro-op bus 52. The reorder circuit 42 clears the valid bits to indicate that the corresponding result data value is not valid because the physical micro-ops pmop_0 through pmop_3 that generate the result data value are being assembled in the reservation and dispatch circuit 38.

The reorder circuit 42 receives the logical destinations ldst_0 through ldst_3 and the macro instruction pointer deltas mipd_0 through mipd_3 over the logical destination bus 54. The reorder circuit 42 stores the logical destinations ldst_0 through ldst_3 into the LDST fields of the ROB entries specified by the physical destinations pdst the physical micro-ops pmop_0 through pmop_3. The reorder circuit 42 stores the macro instruction pointer deltas mipd_0 through mipd_3 into the IPDELTA fields of the ROB entries specified by the physical destinations pdst the physical micro-ops pmop_0 through pmop_3.

For example, the reorder circuit 42 stores the ldst_0 and the mipd_0 into the LDST and the IPDELTA of the ROB entry specified by the physical destination pdst of the pmop_0. The logical destination in the LDST field of a ROB entry specifies a committed state register in the real register circuit 44 for retirement of the corresponding ROB entry. The macro instruction pointer delta in the IPDELTA field of a ROB entry specifies the original macro instruction of the corresponding ROB entry.

The reorder circuit 42 receives write back speculative result information from the execution circuit 40 over the result bus 62. The write back speculative result information from the execution units EU0 through EU4 comprises result data values, physical destinations pdst and fault data.

The reorder circuit 42 stores the write back speculative result information from the execution units EU0 through EU4 into the ROB entries specified by the physical destinations pdst on the result bus 62. For each execution unit EU0 through EU4, the reorder circuit 42 stores the result data value into the result data value field, and stores the fault data into the fault data field of the ROB entry specified by the physical destination pdst.

The result data values from the executions circuit 40 each include a valid flag. Each valid flag is stored in the valid flag field of the ROB entry specified by the physical destination pdst. The execution units EU0 through EU4 set the valid flags to indicate whether the corresponding result data values are valid.

The reorder circuit 42 receives the retirement physical destinations over the retire notification bus 70. The retirement physical destinations cause the reorder circuit 42 to commit the speculative result data values in the ROB entries RE0 through REn to architectural state by transferring the speculative result data values to the real register circuit 44 over the retirement bus 64.

The retirement bus 64 carries the speculative results for a set of retirement micro-ops rm_0 through rm_4. Each retirement micro-op rm_0

through rm_4 comprises a result data value and a logical destination ldst from one of the ROB entries RE0 through REn.

The retirement physical destinations from the retire logic circuit 46 also cause the reorder circuit 42 to transfer the macro instruction pointer deltas for the retiring ROB entries to the restart circuit 48 over the macro instruction pointer offset bus 120.

The reorder circuit 42 receives the reorder clear signal 78 from the restart circuit 48. The reorder clear signal 78 causes the reorder circuit 42 to clear all of the ROB entries.

Figure 5 illustrates the reservation and dispatch circuit 38. The reservation and dispatch circuit 38 implements a reservation dispatch table 84 comprising a set of reservation station entries RS0 through RSx. The reservation and dispatch circuit 38 receives and stores the physical micro-ops pmop_0 through pmop_3 into available reservation station entries RS0 through RSx, assembles the source data for the physical micro-ops into the reservation station entries RS0 through RSx, and dispatches the ready physical micro-ops to the execution circuit 40. A physical micro-op is ready when the source data is fully assembled in a reservation station entry RS0 through RSx.

Each reservation station entry RS0 through RSx comprises an entry valid flag, an op code, a pair of source data values (SRC1/SRC2 DATA) and corresponding valid flags (V), a pair of physical sources (PSRC1/PSRC2), a physical destination (PDST), and a load buffer identifier (LBID).

The entry valid flag indicates whether the corresponding reservation station entry RS0 through RSx holds a physical micro-op awaiting dispatch.

The op code specifies an operation of the execution unit circuit 40 for the physical micro-op in the corresponding reservation station entry RS0 through RSx.

The SRC1/SRC2 DATA fields of the reservation station entries RS0 through RSx hold the source data values for the corresponding physical micro-ops. The corresponding valid flags indicate whether the source data values are valid.

The physical sources PSRC1/PSRC2 of each reservation station entry RS0 through RSx specify the physical destinations in the reorder circuit 42 that hold the source data for the corresponding physical micro-op. The reservation and dispatch circuit 38 uses the physical sources PSRC1/PSRC2 to detect write back of pending source data from the execution circuit 40 to the reorder circuit 42.

The physical destination PDST of each reservation station entry RS0 through RSx specifies a physical destination in the reorder circuit 42 to hold the speculative results for the corresponding physical micro-op.

The load buffer identifier LBID of each reservation station entry RS0 through RSx specifies a load buffer entry in the memory ordering circuit in

the execution circuit 40. The load buffer entry is valid if the corresponding reservation station entry holds a load memory physical micro-op.

The reservation and dispatch circuit 38 receives the physical micro-ops pmop_0 through pmop_3 over the physical micro-op bus 52. The reservation and dispatch circuit 38 also receives the reservation station entry select signals 66 from the allocator circuit 36. The reservation station entry select signals 66 specify the new reservation station entries.

The reservation and dispatch circuit 38 stores the opcode and physical sources psrc1 and psrc2 for each physical micro-op pmop_0 through pmop_3 into the new reservation station entries RS0 through RSx specified by the reservation station entry select signals 66. The reservation and dispatch circuit 38 sets the entry valid flag for each new reservation station entry.

The reservation and dispatch circuit 38 receives load buffer identifiers for load memory physical micro-ops over the load buffer ID bus 72 from the allocator circuit 36. The reservation and dispatch circuit 38 stores the load buffer identifiers into the appropriate LBID fields of the new reservation station entries RS0 through RSx.

The reservation and dispatch circuit 38 receives the source data values and corresponding valid flags specified by the physical sources psrc1 and psrc2 of the physical micro-ops pmop_0 through pmop_3 from the reorder circuit 42 and the real register circuit 44 over the source data bus 58. The reservation and dispatch circuit 38 transfers the source data values and valid flags into the

SRC1/SRC2 DATA fields and valid flags of the new reservation station entries corresponding to the physical micro-ops pmop_0 through pmop_3.

If the entry valid flags indicate that one or both of the source data values for a reservation station table entry RS0 through RSx is invalid, then the reservation and dispatch circuit 38 waits for the execution circuit 40 to execute previously dispatched physical micro-ops and generate the required source data values.

The reservation and dispatch circuit 38 monitors the physical destinations pdst on the result bus 62 as the execution circuit 40 writes back result data values to the reorder circuit 42. If a physical destination pdst on the result bus 62 corresponds to the physical destination of pending source data for a reservation station table entry RS0 through RSx, then the reservation and dispatch circuit 38 receives the result data value over the result bus 62 and stores the result data value into the corresponding SRC1/SRC2 DATA fields and valid flags. The reservation and dispatch circuit 38 dispatches the pending physical micro-ops to the execution circuit 40 if both source data values are valid.

Figure 6 illustrates the real register circuit 44. The real register circuit 44 implements a real register file 86. The real register file 86 comprises a set of committed state registers that hold committed result data values. The committed state registers buffer committed results for the architectural registers of the original stream of macro-instructions fetched by the instruction fetch and micro-op issue circuit 32.

The result data value in each committed state register may be either an integer data value or a floating-point data value. For one embodiment, the result data value field of each committed state register comprises 86 bits to accommodate both integer and floating-point data values.

For one embodiment, the committed state registers comprise the EAX, EBX, ECX, and EDX committed state registers, etc. that correspond to the architectural registers of the Intel Architecture Microprocessor. The real register file 86 also comprises committed state flags that correspond to the architectural flags of the Intel Architecture Microprocessor. The real register file 86 also comprises microcode registers used by microcode executing in the instruction fetch and micro-op issue circuit 32.

The real register circuit 44 receives the physical micro-ops pmop_0 through pmop_3 over the physical micro-op bus 52. The real register circuit 44 reads the result data values from the committed state registers specified by the physical sources psrc1 and psrc2 of the physical micro-ops pmop_0 through pmop_3 from the real register file 86 if the RRFV flags indicate that the physical sources are retired.

The real register circuit 44 transfers the result data values from the committed state registers specified by the physical sources psrc1 and psrc2 of the physical micro-ops to the reservation and dispatch circuit 38 over the source data bus 58 if the RRFV flags indicate that the physical sources are retired in the real register file 86. The real register circuit 44 always sets the source data valid flags while transferring source data to the reservation and

dispatch circuit 38 over the source data bus 58 because the result data in the committed state registers is always valid.

For example, the result data value from the committed state register specified by the physical source psrc1 of the pmop_0 is transferred as source data src1 data_0 over the source data bus 58 if the RRFV flag for the physical source psrc1 of the pmop_0 is set. The result data value from the committed state register specified by the physical source psrc2 of the pmop_0 is transferred as source data src2 data_0 over the source data bus 58 if the RRFV flag for the physical source psrc2 of the pmop_0 is set.

Similarly, the real register circuit 44 transfers source data src1/src2 data_1 through source data src1/src2 data_3 over the source data bus 58 to provide source data for the physical micro-ops pmop_1 through pmop_3 if the appropriate RRFV flags of the physical micro-ops pmop_1 through pmop_3 are set.

The real register circuit 44 receives the retirement micro-ops rm_0 through rm_3 from the reorder circuit 42 over the retirement bus 64. Each retirement micro-op rm_0 through rm_3 contains speculative results from one of the ROB entries RE0 through REn in the reorder buffer 82.

Each retirement micro-op rm_0 through rm_3 comprises a result data value and a logical destination ldst. The real register circuit 44 stores the result data values of the retirement micro-ops rm_0 through rm_3 into the committed state registers of the real register file 86 specified by the logical destinations ldst the retirement micro-op rm_0 through rm_3.

Figure 7 illustrates a load memory circuit in the execution circuit 40. The load memory circuit comprises an address generation circuit 100, a memory ordering circuit 102, a data translate look-aside buffer (DTLB) circuit 104, and a data cache circuit 106.

The address generation circuit 100 receives dispatched load memory physical micro-ops from the reservation and dispatch circuit 38 over the micro-op dispatch bus 60. Each dispatched load memory physical micro-op on the micro-op dispatch bus 60 comprises an opcode, a pair of source data values src1_data and src2_data, a physical destination pdst, and a load buffer identifier lbid.

The address generation circuit 100 determines a linear address for each dispatched load memory physical micro-op according to the source data values src1_data and src2_data. The linear address may also be referred to as a virtual address. For one embodiment, the address generation circuit 100 implements memory segment registers and generates the linear address according to the memory segmentation of Intel Architecture Microprocessors.

The address generation circuit 100 transfers linear load memory micro-ops to the memory ordering circuit 102 over a linear operation bus 90. Each linear load memory operation on the linear operation bus 90 corresponds to a dispatched load memory physical micro-op received over the micro-op dispatch bus 60. Each linear load memory micro-op comprises the opcode of the corresponding load memory physical micro-op, the linear address l_addr determined from the corresponding source data values src1_data and

src2_data, the corresponding physical destination pdst, and the corresponding load buffer identifier lbid.

The memory ordering circuit 102 contains a load buffer. The memory ordering circuit 102 receives the linear load memory micro-ops over the linear operation bus 90. The memory ordering circuit 102 stores the linear load memory micro-ops in the load buffer according to the corresponding load buffer identifier lbid. The memory ordering circuit 102 dispatches the linear load memory micro-ops from the load buffer to the DTLB circuit 104 over the linear operation bus 90.

The DTLB circuit 104 receives the dispatched linear load memory micro-ops from the memory ordering circuit 102 over the linear operation bus 90. The DTLB circuit 104 provides a physical address to the data cache circuit 106 over a read bus 94 for each linear load memory micro-op received from the memory ordering circuit 102.

The DTLB circuit 104 converts the corresponding linear address l_addr into a physical address for the memory subsystem 26. The DTLB circuit 104 maps the linear address l_addr of each linear load memory micro-op into a physical address according to a predetermined memory paging mechanism.

The DTLB circuit 104 transfers the mapped physical address corresponding linear address l_addr of each linear load memory micro-op to the memory ordering circuit 102 over a physical address bus 96. The memory ordering circuit 102 stores the physical addresses for each linear load memory micro-op in the corresponding load buffer entry. For one embodiment, the

memory ordering circuit 102 stores a portion of the physical addresses for each linear load memory micro-op in the corresponding load buffer entry.

The data cache circuit 106 reads the data specified by the physical address on the read bus 94. If the physical address causes a cache miss, the data cache circuit 106 fetches the required cache line from the memory subsystem 26. The data cache circuit 106 receives cache lines from the memory subsystem 26 over an interface bus 74 through the bus interface circuit 30 which is coupled to the multiprocessor bus 28.

The data cache circuit 106 transfers the read result data, a corresponding valid bit, and fault data for the read access to the reorder circuit 42 and the reservation and dispatch circuit 38 over the result bus 62. The result bus 62 also carries the physical destination from the corresponding load buffer in the memory ordering circuit 102.

The memory ordering circuit 102 senses or "snoops" bus cycles on the multiprocessor bus 28 through the bus interface circuit 30 over the interface bus 74. The memory ordering circuit 102 "snoops" the multiprocessor bus 28 for an external store or read for ownership operation by one of the processors 23 - 24 that may cause a processor ordering violation for one of the dispatched linear load memory micro-ops. The memory ordering circuit 102 "snoops" the multiprocessor bus 28 for an external store operation targeted for the physical address of an already dispatched linear load memory micro-op stored in the load buffer.

During retirement of each load memory physical micro-op, the memory ordering circuit 102 generates the memory ordering restart signals 76 to indicate a possible processor ordering violation according to the snoop detection.

Figure 8 illustrates the memory ordering circuit 102. The memory ordering circuit 102 implements a load buffer 88 comprising a set of load buffer entries LB0 through LBn. Each load buffer entry LB0 through LBn holds a linear load memory micro-op from the address generation circuit 100.

Each buffer entry LB0 through LBn comprises an opcode, a physical destination (PDST), a linear address, a physical address, a load status, and a snoop hit flag.

The memory ordering circuit 102 receives the linear load memory micro-ops over the micro-op dispatch bus 60. The memory ordering circuit 102 stores each linear load memory micro-op into a load buffer entry LB0 through LBn specified by the corresponding load buffer identifier lbid.

The memory ordering circuit 102 sets a "valid" status for each new linear load memory micro-op in the load buffer 88. The "valid" status indicates that the corresponding load buffer entry LB0 through LBn holds an unretired load memory micro-op.

The memory ordering circuit 102 stores the opcode, the physical destination pdst, and the linear address l_addr of each linear load memory micro-op into the corresponding fields of the load buffer entry LB0 through

LBn specified by the load buffer identifier lbid of the linear load memory micro-op.

The memory ordering circuit 102 receives the physical addresses p_addr corresponding to the linear load memory micro-ops from the DTLB circuit 104 over the physical address bus 96. The memory ordering circuit 102 stores the physical address for each linear load memory micro-op into the physical address field of the corresponding load buffer entry LB0 through LBn.

For one embodiment, the physical addresses on the physical address bus 96 comprise bits 12 through 19 of the physical address generated by the DTLB circuit 104 for the corresponding linear load memory micro-ops.

The memory ordering circuit 102 dispatches the linear load memory micro-ops from the load buffer entries LB0 through LBn over the linear operation bus 90 according to the availability of resources in the DTLB circuit 104. The memory ordering circuit 102 sets a "complete" status for each linear load memory micro-op dispatched to the DTLB circuit 104.

The memory ordering circuit 102 "snoops" the multiprocessor bus 28 for external store operations that may cause a processor ordering violation. The memory ordering circuit 102 "snoops" the multiprocessor bus 28 for external stores to one of the physical addresses specified the load buffer entries LB0 through LBn having "complete" status. The memory ordering circuit 102 senses an external physical address snoop_addr and a corresponding snoop_addr_valid signal from the multiprocessor bus 28 over the interface

bus 74. The snoop_addr_valid signal specifies a valid address for a store operation on the multiprocessor bus 28.

For one embodiment, the physical address on the multiprocessor bus comprises 40 bits (bits 0 through 39). Bits 0 through 11 of the linear address for a linear load memory micro-op equal bits 0 through 11 of the corresponding physical address. The memory ordering circuit 102 detects a processor ordering "snoop hit" by comparing bits 5 through 11 of the physical address of external store operations on the multiprocessor bus 28 with bits 5 through 11 of the linear address of the load buffer entries LB0 through LBn having "complete" status. The memory ordering circuit 102 also compares bits 12 through 19 of the physical address of external store operations on the multiprocessor bus 28 with the physical address bits 12 through 19 of the load buffer entries LB0 through LBn having "complete" status.

The memory ordering circuit 102 sets the snoop hit flag for the load buffer entries LB0 through LBn causing a processor ordering snoop hit. The memory ordering circuit 102 does not set the snoop hit flag if the load buffer entry LB0 through LBn causing a processor ordering snoop hit holds the oldest linear load memory micro-op in the load buffer 88. Snooping for the oldest linear load memory micro-op in the load buffer 88 is disabled by clearing a corresponding snoop enable flag the appropriate load buffer entry LB0 through LBn.

The memory ordering circuit 102 receives the retirement physical destinations from the retire logic circuit 46 over the retire notification bus 70. The memory ordering circuit 102 issues the memory ordering restart signals

76 to indicate a possible processor ordering violation if one of the load buffer entries LB0 through LBn specified by the retirement physical destinations has the corresponding snoop hit flag set.

Figure 9 illustrates the snoop detection circuitry in the memory ordering circuit 102. The snoop detection circuitry includes a snoop detect circuit corresponding to each load buffer entry LB0-LBn in the memory ordering circuit 102.

For example, a snoop detect circuit 200 corresponds to the load buffer entry LB0. The snoop detect circuit 200 comprises a valid register 210, complete register 214, a physical address register 216, a linear address register 218, a snoop enable register 212, and a snoop hit register 222.

The valid register 210 contains the "valid" status indicating whether the load buffer entry LB0 contains a valid load memory operation. The complete register 214 holds the "complete" status indicating whether the load memory operation for the corresponding load buffer entry LB0 has dispatched. The physical address register 216 holds the physical address bits 19-12 corresponding to the load buffer entry LB0. The linear address register 218 stores bits 11-5 of the linear address for the load memory operation corresponding to the load buffer entry LB0. The snoop enable register 212 holds a snoop enable flag that enables or disables external store snooping for the load buffer entry LB0.

The physical address register 216 receives a set of snoop address bits 230. The snoop address bits 230 comprise bits 19-12 of the snoop_addr received over the interface bus 74. The physical address register 216 asserts a physical address detect signal 236 if the physical address bits 230 equal the physical address bits 19-12 corresponding to the load buffer entry LB0.

The linear address register 218 receives a set of physical address bits 232. The physical address bits 232 comprise bits 11-5 of the snoop_addr received over the interface bus 74. The linear address register 218 generates a linear address detect signal 237 if the physical address bits 232 equal bits 11-5 of the linear address corresponding to the load buffer entry LB0.

A snoop_addr_valid signal 234 is received over the interface bus 74. The snoop_addr_valid signal 234 indicates that the snoop_addr on the interface bus 74 corresponds to a valid external store operation. The output of an AND gate 220 sets a snoop hit flag in the snoop hit register 222 by combining the physical address detect signal 236, and the linear address detect signal 237, the "complete" and the "valid" status," and the snoop enable flag.

Figure 10 illustrates notification circuitry in the memory ordering circuit 102 that generates the memory ordering restart signals 76. The memory ordering circuit 102 contains a notification circuit for each of the load buffer entries LB0-LBn.

For example, the notification circuit 250 corresponds to the load buffer entry LB0. The snoop hit register 222 contains the snoop hit flag for the load buffer entry LB0. A physical destination (PDST) register 260 holds the physical destination corresponding to the load buffer entry LB0.

The PDST register 260 receives a set of retirement physical destinations 270-272 over the retire notification bus 70 indicating the next set of retiring physical micro-ops. The PDST register 260 generates a set of control signals 300-302. The control signals 300-302 indicate whether any of the retirement physical destinations 270-272 match the physical destination in the load buffer entry LB0.

For example, the PDST register 260 generates the control signal 300 to indicate that the retirement physical destination 270 matches the physical destination in load buffer entry LB0. Similarly, the PDST register 260 generates the control signal 301 to indicate that the retirement physical destination 271 matches the physical destination in load buffer entry LB0, and the control signal 302 to indicate that the retirement physical destination 272 matches the physical destination in load buffer entry LB0.

The memory ordering restart circuit 250 receives a set of retirement physical destination valid flags 280-282 over the retire notification bus 70. The retirement physical destination valid flags 280-282 indicate whether the retirement physical destinations 270-272 are valid.

For example, the retirement physical destination valid flag 280 indicates whether the retirement physical destination 270 is valid. Similarly, the retirement physical destination valid flag 281 indicates whether the retirement physical destination 271 is valid, and the retirement physical destination valid flag 282 indicates whether the retirement physical destination 272 is valid.

The control signals 300-302 and the retirement physical destination flags 280-282 are combined with the snoop hit flag by a set of AND gates 310-312. The outputs of the AND gates 310-312 are stored in a register 262. The outputs of the register 262 are synchronized by a clock signal 350.

The register 262 stores the memory ordering restart flags for the load buffer entry LB0. The outputs of the AND gates 320-322 control a set of pull down transistors Q1, Q2 and Q3. The pull down transistors Q1, Q2 and Q3 are coupled to a set of memory ordering restart signal lines 290-292. The memory ordering restart signal lines 290-292 are also coupled to a set of pull up transistors Q4, Q5 and Q6 which are synchronized by the clock signal 350.

If the control signal 300 indicates that the retirement physical destination 270 matches the physical destination in load buffer entry LB0, and if the retirement physical destination valid flag 280 indicates that the retirement physical destination 270 is valid, and if the snoop hit flag for the load buffer entry LB0 is set, then the output of the AND

gate 320 switches on the transistor Q1. The transistor Q1 pulls down the voltage on the memory ordering restart signal line 290 to indicate that the physical micro-op specified by the retirement physical destination 270 has caused a possible processor ordering violation.

Similarly, the memory ordering restart signal line 291 indicates that the physical micro-op specified by the retirement physical destination 271 has caused a possible processor ordering violation, and the memory ordering restart signal line 292 indicates that the physical micro-op specified by the retirement physical destination 272 has caused a possible processor ordering violation.

Figure 11 illustrates a load micro-op issue by the instruction fetch and micro-op issue circuit 32. The logical micro-op (ld 0x100, EBX, EAX) is transferred by the instruction fetch and micro-op issue circuit 32 over the logical micro-op bus 50. The logical micro-op ld 0x100, EBX, EAX specifies a load memory operation to the architectural register EAX from the memory subsystem 26. The address is specified by the contents of the architectural register EBX and offset 100 hex.

The allocator circuit 36 receives the logical micro-op ld 0x100, EBX, EAX over the logical micro-op bus 50, and generates a physical destination pdst equal to 42. The allocator circuit 36 transfers the pdst 42 to the register alias circuit 34 over the physical destination bus 56.

The register alias circuit 34 receives the physical destination pdst 42, and translates the logical micro-op ld 0x100, EBX, EAX into a physical micro-

op ld 100, 35, 42. The argument 100 specifies that psrc1 is a constant data value equal to 100 hex. The argument 35 specifies that psrc2 is the RE35 entry in the reorder buffer 82 according to the ROB pointer and the RRFV flag for the EBX entry in the register alias table 80.

The register alias circuit 34 transfers the physical micro-op ld 100, 35, 42 to the reservation and dispatch circuit 38, the reorder circuit 42, and the real register circuit 44 over the physical micro-op bus 52.

The register alias circuit 34 stores the allocated pdst 42 for the physical micro-op ld 100, 35, 42 into the ROB pointer of the EAX entry in the register alias table 80. The register alias circuit 34 also clears the RRFV bit for the EAX entry in the register alias table 80 to indicate that the logical register EAX is mapped to the reorder buffer 82 in a speculative state.

The reorder circuit 42 and the real register circuit 44 receive the physical micro-op ld 100, 35, 42 over the physical micro-op bus 52. The reorder circuit 42 reads source data for the physical source psrc2 35 by reading ROB entry RE35 of the reorder buffer 82. The ROB entry RE35 of the reorder buffer 82 contains a result data value equal to 2000 and a valid bit set for the current speculative state of the EBX architectural register.

The reorder circuit 42 transfers the result data value 2000 and the constant data value 100 along with corresponding valid bits to the reservation and dispatch circuit 38 over the source data bus 58 as a source data pair src1/src2 data.

The reorder circuit 42 receives the logical destination ldst EAX for the physical micro-op ld 100, 35, 42 over the logical destination bus 54. The reorder circuit 42 stores the logical destination ldst EAX into the LDST of the entry RE42 of the reorder buffer 82. The reorder circuit 42 clears the valid flag of the entry RE42 of the reorder buffer 82 to indicate that the corresponding result data is not valid.

The reservation and dispatch circuit 38 receives the physical micro-op physical micro-op ld 100, 35, 42 over the physical micro-op bus 52. The reservation and dispatch circuit 38 stores the opcode ld into the opcode field of the entry RS0 of the reservation station table 84 as specified by the allocator circuit 36. The reservation and dispatch circuit 38 stores the physical destination pdst 42 into the PDST of the reservation station table 84 entry RS0. The reservation and dispatch circuit 38 stores the physical sources psrc1 xxx and psrc2 35 into the PSRC1/PSRC2 of the reservation station table 84 entry RS0. The reservation and dispatch circuit 38 also sets the entry valid flag of the reservation station table 84 entry RS0.

The reservation and dispatch circuit 38 receives the source data values src1/src2 data 100 and 2000 and corresponding valid flags over the source data bus 58. The reservation and dispatch circuit 38 stores the source data values src1/src2 data 100 and 2000 and corresponding valid flags into the SRC1/SRC2 and V fields of the reservation station table 84 entry RS0.

The reservation and dispatch circuit 38 receives a load buffer identifier lbid = 4 for the physical micro-op ld 100, 35, 42 from the allocator circuit 36 over the load buffer ID bus 72. The reservation and dispatch circuit 38 stores

the load buffer identifier lbid = 4 into the LBID field of the reservation station table 84 entry RS0.

The reservation and dispatch circuit 38 dispatches the load memory physical micro-op ld 100, 2000, 42, lbid = 4 to the address generation circuit 100 over the micro-op dispatch bus 60. The address generation circuit 100 converts the source data values 100, 2000 into a linear address 32100 according to segment register values. The address generation circuit 100 then transfers a corresponding linear load memory micro-op ld 32100, 42, lbid = 4 to the memory ordering circuit 102 over the linear operation bus 90.

The memory ordering circuit 102 receives the linear load memory micro-op ld 32100, 42, lbid = 4 over the linear operation bus 90. The memory ordering circuit 102 stores the linear load memory micro-op ld 32100, 42, lbid = 4 into entry LB4 of the load buffer 88 as specified by the corresponding load buffer identifier lbid = 4. The memory ordering circuit 102 sets "valid" load status for load buffer entry LB4.

The memory ordering circuit 102 also contains an older linear load memory micro-op ld 31000, 41 in entry LB3 of the load buffer 88 having a "complete" status. The "complete" status indicates that the linear load memory micro-op ld 31000, 41 has been dispatched to the DTLB circuit 104 for execution. The load buffer entry LB3 contains physical address bits 6 - 19 equal to 1040 hex, which corresponds to a physical address equal to 41000 generated by the DTLB circuit 104 for the linear address 31000.

Figure 12 illustrates the dispatch and retirement of the linear load memory micro-op ld 32100, 42, lbid = 4. The memory ordering circuit 102 dispatches the linear load memory micro-op ld 32100, 42 from the load buffer entry LB4 to the DTLB circuit 104 over the linear operation bus 90. The memory ordering circuit 102 then sets "complete" status for the load buffer entry LB4.

The DTLB circuit 104 generates a physical address equal to 42100 for the linear address 32100 of the linear load memory micro-op ld 32100, 42. The DTLB circuit 104 performs a read access of the data cache circuit 106 at physical address 42100 over the read bus 94.

The memory ordering circuit 102 receives the physical address bits 6 - 19 equal to 1084 corresponding to the linear load memory micro-op ld 32100, 42 over the physical address bus 96. The memory ordering circuit 102 stores the physical address bits 6 - 19 equal to 1084 into the physical address field of the load buffer entry LB4.

The data cache circuit 106 transfers the result data value equal to 225 for the read to physical address 42100, a corresponding valid bit, and fault data for the read access to the reorder circuit 42 and the reservation and dispatch circuit 38 over the result bus 62. The result bus 62 also carries the physical destination 42 corresponding to the result data for the linear load memory micro-op ld 32100, 42.

The reorder circuit 42 stores the result data equal to 225 and corresponding valid bit into entry RE42 of the reorder buffer 82 as specified by the physical destination 42 on the result bus 62.

The memory ordering circuit 102 senses a snoop hit on the multiprocessor bus 28 for an external store having the physical address bits 6 - 19 equal to 1084. The physical address bits 6 - 19 equal to 1084 correspond to entry LB4 of the load buffer 88. The memory ordering circuit 102 sets the snoop hit flag for entry LB4 to indicate the processor ordering snoop hit.

The memory ordering circuit 102 also senses a snoop hit on the multiprocessor bus 28 for an external store having the physical address bits 6 - 19 equal to 1040. The memory ordering circuit 102 does not set the snoop hit flag for entry LB3 because the linear load memory micro-op ld 31000, 41 is the oldest linear load memory micro-op in the load buffer 88.

The memory ordering circuit 102 then receives a set of retirement physical destinations 40, 41, 42 from the retire logic circuit 46 over the retire notification bus 70. In response, the memory ordering circuit 102 issues the memory ordering restart signals 76 to indicate a possible processor ordering violation for the linear load memory micro-op ld 32100, 42.

The memory ordering restart signals 76 cause the restart circuit 48 to issue the reorder clear signal 78. The reorder clear signal 78 causes the reorder circuit 42 to clear the speculative result data for the unretired physical micro-ops in the reorder buffer 82, and causes the reservation and dispatch circuit 38 to clear the pending physical micro-ops that await dispatch to the execution

circuit 40. The reorder clear signal 78 also causes the allocator circuit 36 to reset the allocation pointer for allocating the physical registers in the reorder circuit 42, and causes the retire logic circuit 46 to reset the retirement pointer for retiring the physical registers.

The restart circuit 48 uses the macro instruction pointer delta values received over the macro instruction pointer offset bus 120 to calculate a restart instruction pointer value. The restart circuit 48 transfers the restart instruction pointer value to the instruction fetch and micro-op issue circuit 32 over the restart vector bus 122.

The reorder clear signal 78 causes the micro-instruction sequencer of the instruction fetch and micro-op issue circuit 32 to reissue the in order stream of logical micro-ops that were cleared from the reorder circuit 42 before retirement.

In the foregoing specification the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are accordingly to be regarded as illustrative rather than a restrictive sense.

CLAIMS



1. A method for processor ordering in a multiprocessor computer system, comprising the steps of:

fetching a load memory instruction from an external memory in a sequential program order, the load memory instruction specifying a load memory operation from a memory address over a multiprocessor bus;

executing the load memory instruction and snooping the multiprocessor bus for a processor ordering conflict at the memory address;

committing the load memory instruction to an architectural state in the sequential program order if the external store operation to the memory address is not detected;

reexecuting the load memory instruction if the external store operation to the memory address is detected.

2. The method of claim 1, wherein the step of fetching a load memory instruction comprises the step of fetching an instruction stream from the external memory in the sequential program order, the instruction stream comprising the load memory instruction, the load memory instruction specifying a load memory operation from the memory address over the multiprocessor bus.

3. The method of claim 2, wherein the step of executing the load memory instruction comprises the steps of:



assembling at least one source data value for the load memory instruction, such that the source data value specifies the memory address for the load memory instruction;

executing the load memory instruction after the source data value for the load memory instruction is assembled, the executed load memory instruction generating a result data value.

4. The method of claim 3, wherein the step of snooping the multiprocessor bus for a processor ordering conflict at the memory address comprises the step of snooping the multiprocessor bus for an external store operation to the memory address of the executed load memory instruction.

5. The method of claim 3, wherein the step of snooping the multiprocessor bus for a processor ordering conflict at the memory address comprises the step of snooping the multiprocessor bus for an external read for ownership operation to the memory address of the executed load memory instruction.

6. The method of claim 4, wherein the step of committing the load memory instruction to an architectural state comprises the step of committing the result data value to the architectural state in the sequential program order if the external store operation to the memory address of the executed load memory instruction is not detected.

7. The method of claim 6, wherein the step of reexecuting the load memory instruction if the external store operation to the memory address is detected comprises the steps of discarding the result data value, then



reexecuting the instruction stream starting with the load memory instruction corresponding to the discarded result data value if the external store operation to the memory address of the executed load memory instruction is detected before the result data value is committed to the architectural state.

8. The method of claim 7, wherein the step of executing the load memory instruction after the source data value for the load memory instruction is assembled comprises the steps of:

determining a linear memory address for the load memory instruction from the source data value for the load memory instruction;

storing the load memory instruction and the linear memory address in an available load buffer entry of a load buffer;

converting the linear address of the load memory instruction into a physical address, and storing the physical address into the load buffer entry;

performing a load memory operation from the physical address, and setting a complete status for the load buffer entry, the load memory operation generating the result data value.

9. The method of claim 8, wherein the step of snooping the multiprocessor bus for an external store operation to the memory address of the executed load memory instruction comprises the steps of:

snooping the multiprocessor bus for an external store operation to the physical address stored in the load buffer entry if the load buffer entry contains the complete status;

setting a snoop hit flag in the load buffer entry if the external store operation to the physical address stored in the load buffer entry is detected.



10. The method of claim 9, wherein the step of committing the result data value to an architectural state in the sequential program order comprises the steps of:

generating a retirement pointer in the sequential program order, such that the retirement pointer specifies the result data value stored in a physical register of a reorder buffer, and specifies the load buffer entry in the load buffer;

committing the result data value in the physical register to the architectural state if the snoop hit flag of the load buffer entry is not set.

11. The method of claim 9, wherein the step of discarding the result data value comprises the steps of:

generating a retirement pointer in the sequential program order, such that the retirement pointer specifies the result data value stored in a physical register of a reorder buffer, and specifies the load buffer entry in the load buffer;

clearing the result data value in the physical register if the snoop hit flag of the load buffer entry is set.

12. An apparatus for processor ordering in a multiprocessor computer system, comprising:

means for fetching a load memory instruction from an external memory in a sequential program order, the load memory instruction specifying a load memory operation from a memory address over a multiprocessor bus;

means for executing the load memory instruction and snooping the multiprocessor bus for a processor ordering conflict at the memory address;



means for committing the load memory instruction to an architectural state in the sequential program order if the external store operation to the memory address is not detected;

means for reexecuting the load memory instruction if the external store operation to the memory address is detected.

13. The apparatus of claim 12, wherein the means for fetching a load memory instruction comprises means for fetching an instruction stream from the external memory in the sequential program order, the instruction stream comprising the load memory instruction, the load memory instruction specifying a load memory operation from the memory address over the multiprocessor bus.

14. The apparatus of claim 13, wherein the means for executing the load memory instruction comprises:

means for assembling at least one source data value for the load memory instruction, such that the source data value specifies the memory address for the load memory instruction;

means for executing the load memory instruction after the source data value for the load memory instruction is assembled, the executed load memory instruction generating a result data value.

15. The apparatus of claim 14, wherein the means for snooping the multiprocessor bus for a processor ordering conflict at the memory address comprises means for snooping the multiprocessor bus for an external store operation to the memory address of the executed load memory instruction.



16. The apparatus of claim 14, wherein the means for snooping the multiprocessor bus for a processor ordering conflict at the memory address comprises means for snooping the multiprocessor bus for an external read for ownership operation to the memory address of the executed load memory instruction.

17. The apparatus of claim 15, wherein the means for committing the load memory instruction to an architectural state comprises means for committing the result data value to the architectural state in the sequential program order if the external store operation to the memory address of the executed load memory instruction is not detected.

18. The apparatus of claim 17, wherein the means for reexecuting the load memory instruction if the external store operation to the memory address is detected comprises means for discarding the result data value, and means for reexecuting the instruction stream starting with the load memory instruction corresponding to the discarded result data value if the external store operation to the memory address of the executed load memory instruction is detected before the result data value is committed to the architectural state.

19. The apparatus of claim 18, wherein the means for executing the load memory instruction after the source data value for the load memory instruction is assembled comprises:

means for determining a linear memory address for the load memory instruction from the source data value for the load memory instruction;

means for storing the load memory instruction and the linear memory address in an available load buffer entry of a load buffer;



means for converting the linear address of the load memory instruction into a physical address, and storing the physical address into the load buffer entry;

means for performing a load memory operation from the physical address, and setting a complete status for the load buffer entry, the load memory operation generating the result data value.

20. The apparatus of claim 19, wherein the means for snooping the multiprocessor bus for an external store operation to the memory address of the executed load memory instruction comprises:

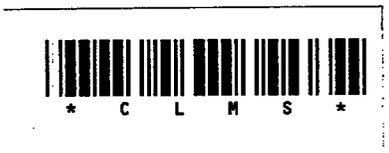
means for snooping the multiprocessor bus for an external store operation to the physical address stored in the load buffer entry if the load buffer entry contains the complete status;

means for setting a snoop hit flag in the load buffer entry if the external store operation to the physical address stored in the load buffer entry is detected.

21. The apparatus of claim 20, wherein the means for committing the result data value to an architectural state in the sequential program order comprises:

means for generating a retirement pointer in the sequential program order, such that the retirement pointer specifies the result data value stored in a physical register of a reorder buffer, and specifies the load buffer entry in the load buffer;

means for committing the result data value in the physical register to the architectural state if the snoop hit flag of the load buffer entry is not set.



22. The apparatus of claim 20, wherein the means for discarding the result data value comprises:

means for generating a retirement pointer in the sequential program order, such that the retirement pointer specifies the result data value stored in a physical register of a reorder buffer, and specifies the load buffer entry in the load buffer;

means for clearing the result data value in the physical register if the snoop hit flag of the load buffer entry is set.

23. A method for processor ordering in a multiprocessor computer system, substantially as hereinbefore described.

24. An apparatus for processor ordering in a multiprocessor computer system, substantially as hereinbefore described, with reference to the accompanying drawings.