

MULTIPLE EXECUTION UNIT BYPASSES FOR
EFFICIENT PROCESSOR EXECUTION

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to the field of microprocessor architecture. More specifically, the present invention relates to data bypassing within complex microprocessors to increase instruction throughput.

2. Art Background

As the computer revolution has progressed the quest of microprocessor developers has been to develop chips exhibiting more power and faster performance. Initial efforts focused essentially on increasing transistor populations on single microprocessor integrated circuits. That effort continues with today's microprocessors now housing literally millions of transistors on a single chip. Further integration has allowed processor clock speeds to be greatly increased with the increased density of transistors.

In addition to squeezing performance by overcoming physical limitations, microprocessor design has developed into an art form. Microprocessors are divided into discrete functional blocks through which instructions are propagated one stage at a time. This allows for pipelining of instructions such that when one instruction has completed the

first stage of processing and moves on to the second stage, a second instruction may begin the first stage. Thus, even where each instruction requires a number of clock cycles to complete all stages of processing, pipelining provides for the completion of instructions on every clock cycle. This single-cycle throughput of a pipelined microprocessor greatly increases the overall performance of computer systems.

Other enhancements to microprocessor design include the development of superscalar microprocessors which are capable of initiating more than one instruction at the initial stage of the pipeline per clock cycle. Likewise, in a superscalar microprocessor, frequently more than one instruction completes on each given clock cycle. Other development efforts have gone into the simplification of microprocessor instruction sets, developing reduced instruction set (RISC) microprocessors which exploit the fact that many simple instructions are more commonly executed than some complicated instructions. Eliminating the complicated instructions from the instruction set provides for a faster executing pipeline. Complicated instructions are carried out by combinations of the more simple instructions.

In order for pipelined microprocessors to operate efficiently, an instruction fetch unit at the head of the pipeline must continually provide the pipeline with a stream of instructions. However, conditional branch instructions within an instruction stream prevent an instruction fetch

unit at the head of a pipeline from fetching the correct instruction until the condition is resolved. Since the condition will not be resolved until further down the pipeline, the instruction fetch unit may not be able to fetch proper instructions.

To overcome this problem, many pipelined microprocessors use branch prediction mechanisms that predict the outcome of branches and then fetch subsequent instructions according to branch prediction. Branch prediction is achieved using a branch target buffer (BTB) to store the history of a branch instruction based upon the instruction pointer or address of that instruction. Every time a branch instruction is fetched, the branch target buffer predicts the target address of the branch using the branch history. Speculative execution is where instructions are initiated and completed before knowing if they are the correct instructions. This usually includes prediction with a BTB.

In addition to speculative execution, substantial increases in instruction throughput are achievable by implementing out-of-order dispatch of instructions to the execution units. Many experiments have confirmed that typical von Neumann code provides substantial parallelism and hence a potential performance boost by use of out-of-order execution. Out-of-order execution is possible when a given instruction does not depend on previous instructions for a result before executing. With out-of-order execution, any

number of instructions are allowed to be in execution in the execution units, up to the total number of pipeline stages for all the functional units.

In a processor using out-of-order execution, instruction
5 dispatching is stalled when there is a conflict for a
functional unit or when a dispatched instruction depends on
the result of an instruction that is not yet computed. In
order to prevent or mitigate stalls in decoding, previous
texts have described the provision of a buffer known as a
10 reservation station (RS) between the decode and execute
stages. The processor decodes instructions and places them
into the reservation station as long as there is room in the
buffer and at the same time examines instructions in the
reservation station to find those that can be dispatched to
15 the execution units (that is, instructions for which source
operands and execution units are available). Data-ready
instructions are dispatched from the reservation station with
little regard for their original program order. For further
background on the use of reservation stations and out-of-
20 order execution, see Mike Johnson, Superscalar Microprocessor
Design and Prentice-Hall, Inc., 1991, Chapters 3 and 7.

Regardless of the speed and performance enhancing
designs implemented in a given processor, most processors are
generally divided into fundamental computational blocks.
25 Generally, separate functional units are provided for integer
operations, floating point operations, memory operations,

etc. Further, in some architectures it may prove efficient to duplicate particular functional unit which are heavily used, such as the integer execution unit. In an out-of-order processor, it is frequently possible that the results of an operation being carried out by one functional unit will be needed by an operation to be carried out by another functional unit. In an effort to increase system speed and performance by maximizing instruction throughput, it is desirable, and therefore an object of the present invention, to provide a comprehensive method and apparatus for providing result data from one execution unit directly to a next execution unit which may be executing a dependent instruction without requiring additional processing of the result data before scheduling of the dependent instruction.

SUMMARY OF THE INVENTION

From the foregoing, it can be appreciated that there is a need to provide a mechanism for the rapid exchange of data between various execution units implemented in complex
5 microprocessors. Therefore, it is an object of the present invention to provide an extensive bypassing method and apparatus to provide the exchange of execution unit result data efficiently between execution units in a microprocessor.

These and other objects of the present invention are
10 provided by a plurality of bypassing multiplexers implemented in a microprocessor data path incorporating a plurality of execution units. Result data from each execution unit is propagated along a write back path. Coupled to the input side of all execution units are a plurality of bypass
15 multiplexers which may take result data from the write back path and provide that data to an associated execution unit as an input data source. The enabling signals to the plurality of bypass multiplexers are controlled through scheduling logic associated with the microprocessor's execution path.

20 In the preferred and implemented embodiment of the present invention, a plurality of execution units are provided in an out-of-order executing microprocessor. Instructions to be executed are buffered in an instruction buffer referred to as a reservation station. The reservation
25 station buffers instructions until all required source data

for an instruction is available and a required execution unit is available. The reservation station dispatches instructions through any one of several dispatch ports to which the plurality of execution units are coupled. Result data from the execution unit is provided to a reorder buffer for reassembly to original program order. The result data is also provided along the write back path to the reservation station for dependent instructions that may be waiting for that result data to provide needed source data. Implemented at the input path to the execution units are a plurality of bypass multiplexers which are capable of taking result data directly from the write back paths and providing that data as input source data to an associated execution unit.

The preferred embodiment implementation of the present invention includes scheduling logic within the reservation station which knows when a given execution unit will be providing result data over the write back path. When the scheduling logic knows that a piece of result data provides needed input source data for another instruction to be executed by an execution unit, the scheduling logic of the reservation station provides an enable signal to the necessary bypass multiplexer to propagate the result data from the write back path directly to a source input of the execution unit executing the dependent instruction. This data may be bypassed to other execution units or to the same execution unit which generated the result data that is now being used as source data for another instruction. By

providing extensive bypassing capability, the present invention increases instruction throughput for a complex microprocessor.

BRIEF DESCRIPTION OF THE DRAWINGS

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

5 Figure 1 illustrates a basic computer architecture which may incorporate the present invention.

Figure 2 is a block diagram illustrating a portion of a microprocessor incorporating the present invention.

10 Figure 3 illustrates a more detailed block diagram of the out-of-order core of a microprocessor incorporating a preferred embodiment of the present invention.

Figures 4 illustrates a more detailed portion of the out-of-order core of a microprocessor implementing the present invention bypassing arrangement.

DETAILED DESCRIPTION OF THE INVENTION

The present invention provides a bypassing mechanism to increase instruction throughput for a microprocessor implementing a plurality of execution units. In the following description, numerous specific details are set forth such as particular data paths and specified couplings for a preferred embodiment implementation in order to provide a thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art that the present invention may be practiced in a more general application where such advantages provided by the present invention are desired. In other instances, well-known control structures and gate level circuits have not been shown in detail in order not to obscure important aspects of the present invention. Particularly, many functions are described to be carried out by various logic and multiplexing circuits. Once the various functions have been described, a practitioner of ordinary skill in the art will be able to implement the necessary logic and multiplexing circuits without undue experimentation.

Referring first to Figure 1, a typical computer system which may utilize a microprocessor that implements the present invention is illustrated. As shown in Figure 1, there is computer 100 which comprises three major components. The first of these is the input/output (I/O) circuit 101

which is used to communicate information in appropriately structured form to and from other parts of the computer 100 as well as out of the computer 100. Also shown as part of the computer 100 is the central processing unit (CPU) 102 and
5 memory 103. These two latter elements are those typically found in most general purpose computers and almost all special purpose computers. In fact, the several elements contained within computer 100 are intended to be representative of this broad category of data processor.

10 Also shown in Figure 1 is an input device 105, shown in a typical embodiment as a keyboard. There is also shown as an input device a graphics tablet 107. It should be understood, however, that the input device may actually be in any other well-known input device (including, of course,
15 another computer). A mass memory device 104 is coupled to I/O circuit 101 and provides additional storage capabilities for the computer 100. The mass memory may include other programs and may take the form of magnetic or optical disc drive or any other well-known device. It will be appreciated
20 that the data retained within mass memory 104, may, in appropriate cases, be incorporated in standard fashion into computer 100 as part of memory 103.

In addition, three typical computer display devices are illustrated, the display monitor 108, the plotter 109 and a
25 laser printer 110. Each can be used to display images or documents or other data utilized by the computer 100. A

cursor control device 106, such as a mouse, trackball or stylus are also coupled to I/O circuit 101. Other pointing devices may suitably be used as appropriate.

The preferred embodiment of the present invention is intended to be implemented in a microprocessor that executes the well-known Intel Architecture Microprocessor instruction set. The present invention may, however, be implemented in any microprocessor architecture in which a reservation station is utilized for storing instructions until data and resource dependencies are resolved. A variety of microprocessors may take advantage of the present invention whether those microprocessors occupy a single chip or more than one and regardless of the materials used to fabricate the chip including silicon or gallium arsenide.

The microprocessor of the present invention, prior to executing Intel Architecture Microprocessor instructions, decodes them into a more simple, stylized sequence of "micro operations." The micro operations are then analyzed and scheduled according to resolved dependencies thus implementing out-of-order execution. The sequence below shows the mapping of the instructions of the Dhrystone trace into their constituent micro operations ("Gn" is general register n):

	Intel Architecture Instructions	micro operations
	000014e8: PUSHw BP <TOS fbfffc3c>	STw IN (G5, G4) OUT (Memfbfffc3c) SUB IN (SPO, CO) OUT (SPO)
5	000014e9: MOVw EA, R ESP EBP	MVw IN (G4) OUT (G5)
	000014eb: LEAVE <TOS fbfffc3c>	ADD IN (G5, CO) OUT (G4) LDw IN (G5, Memfbfffc3c) OUT (G5)
	000014ec: RETn	LDw IN (G4, Memfbfffc40) OUT (TO) ADD IN (SPO, CO) OUT (SPO)
10	00001625: XORw EA, R EDI EDI	JMP IN (TO) OUT () XOR IN (G7, G7) OUT (G7)
	00001625: ADDw Ib, EA ESP <IMM 18>	ADD IN (G4) OUT (G4, flags 0)

It can be seen that the PUSH of the BP register is converted into two micro operations. The first performs the store of the register and the second adjusts the stack pointer. These two micro operations are not dependent on each other. The "microcode" corresponding to the PUSH has been written explicitly to allow maximal parallelism wherever possible. It can be seen that some complex instructions translate into multiple micro operations which may or may not be related to one another, and simple instructions map directly into a unique micro operation (such as XOR or ADD).

Figure 2 is a block diagram of an embodiment of a speculative out-of-order processor in accordance with the present invention. The processor comprises an instruction fetch unit (IFU) 102 coupled to a branch target buffer (BTB) 104 and an instruction decoder (ID) 106. Based upon the instruction pointer (IP) provided by the BTB 104 to the IFU 102, the IFU 102 fetches the macro instruction found at the address in memory (not shown) that is indicated by the IP. The instruction is decoded into one or more micro operations by the instruction decoder 106. Such a decoding mechanism is

found in the Intel Microprocessor Architectures. The instruction decoder 106 transfers the stream of micro operations to a register alias table (RAT) 114 and an allocator (ALLOC) 112. In one embodiment, the instruction
5 decoder 106 issues up to three in-order micro operations during each cycle of the processor.

The allocator 112 assigns each incoming micro operation to a location in the reorder buffer (ROB) 40, thereby mapping the logical destination address (LDST) of the micro operation
10 to a corresponding physical destination address (PDST) in the ROB. A register alias table (RAT) 114 maintains this mapping.

The contents of a ROB register are retired to a location in a real register file (RRF) 110. The RAT 114 thus also
15 stores a real register file valid bit (RRFV) that indicates whether the value indicated by the logical address is to be found at the physical address in the ROB or in the RRF after retirement. Based upon this mapping, the RAT 114 also associates every logical source address to a corresponding
20 location in the ROB or the RRF (the source operand of one instruction generally must have been the destination of a previous instruction).

Each incoming micro operation is also assigned and written into an entry in the reservation station 20 by the
25 allocator 112. The reservation station (RS) 20 assembles the micro operations awaiting execution by an appropriate

execution unit, such as integer execution unit (IEU) 34 or memory execution unit (MEM EU) 117.

Figure 3 illustrates a more detailed portion of the microprocessor incorporating the present invention. The microprocessor implementing the preferred embodiment of the present invention utilizes out-of-order execution of instructions. The portion of the microprocessor illustrated in Figure 3 can be considered the out-of-order core of the microprocessor. The first element of the out-of-order core illustrated in Figure 3 is the reservation station 20. The reservation station 20 is a buffer in front of the functional units that holds micro operations that are waiting for needed resources in order to execute. These needed resources include the source data operands (that previous micro operations may not yet have created) and a functional unit with which to execute the micro operation.

The reservation station performs several functions. Its data fields serve as a temporary buffer for instructions and the source data that is or will be available. The reservation station maintains waiting instructions and is "watching" all the result/write back buses from all the execution interfaces "looking" for source data that it needs for its micro operations. When the watched-for data arrives on the result bus, the reservation station writes it into the appropriate source data field.

Once all the source operands for a micro operation are in the reservation station, the reservation station determines when an appropriate execution unit is available and schedules the data ready micro operation for dispatch.

- 5 It is the scheduling of micro operation dispatch which concerns the present invention.

The reservation station 20 of Figure 3 is illustrated having five dispatch ports. Primarily, micro operations in the embodiment illustrated will be dispatched over either
10 Dispatch Port 0 or Dispatch Port 1. In the preferred embodiment microprocessor, there are five execution units coupled to Dispatch Port 0 from reservation station 20. These are the floating point add unit 31, the multiply unit 32 for performing integer and floating point multiplies,
15 integer and floating point divide unit 33, integer execution unit 34, and a hybrid execution unit 35. As was described, the complex Intel Microprocessor instruction set is reduced to a simpler stream of micro operations. These micro operations, when they are dispatched from the reservation
20 station, are operated on by the various execution units. Most applications heavily utilize the integer execution unit. Integer execution unit 34 is designed to be able to process data-ready micro operations in one clock cycle. It is therefore referred to as a single-cycle functional unit. The
25 integer execution unit 34 receives data having a width of 32 bits. The floating point execution units 31, 32 and 33 carry out more complicated tasks when they receive data-ready micro

operations. These functional units receive input streams having a width for 86 bits of source data.

In the preferred embodiment of the present invention microprocessor, a floating point add requires three clock
5 cycles to execute, and a floating point multiply requires five clock cycles to execute. Floating point divides and square roots, which are both carried out by the floating point divide unit 33, have variable length latencies depending upon the precision required for their result. The
10 wire unit 35 is a hybrid execution unit implemented in the preferred embodiment for various other instructions. It is similar to a floating point unit in that its input stream takes sources that are 86 bits wide, but it is more like an integer execution unit in that it performs its operations in
15 a single clock cycle.

Dispatch port 1 from reservation station 20 also has two execution units coupled to it. There is an integer execution unit 36 and a jump unit 37. The integer execution unit 36 may be identical to the execution unit 34, or it may have
20 additional or different functionality. Two integer execution units are provided because of the high volume of integer micro operations common in most code, thereby enhancing performance of the overall system. Of course, alternative microprocessors may incorporate more or fewer execution units
25 depending upon design objectives.

After an execution unit executes a micro operation, it writes back the result to the reorder buffer 40 over write back bus 120. The reorder buffer 40 serves as the place where the results of out-of-order operations are collected so that they can later be committed to machine state in strict von Neumann order; the committed state registers being in the RRF 110. Thus, the reorder buffer is where the out-of-order core reassembles the instruction stream into its program order, before commitment to state. At the same time that execution units write back to the reorder buffer 40, they may simultaneously write back to the reservation station 20. It may be that the result of an execution unit's operation provides a source operand needed for the execution of another micro operation waiting in the reservation station. Similarly, a result from one execution unit may be immediately bypassed to another (or the same) execution unit if a dependent instruction has already been dispatched.

In the preferred embodiment implementation of the out-of-order microprocessor, the reservation station 20 is responsible for scheduling the dispatch of instructions to the various execution units described above. The scheduling logic of the reservation station 20 upon dispatching an instruction to an execution unit knows when the result data for a given instruction is expected to be written back over the write back buses. Further, the reservation station operates like a content address memory in which CAM matching techniques may be implemented for determining that write back

data will supply needed source data for a dependent instruction waiting for dispatch in the reservation station.

Referring now to **Figure 4**, a more detailed portion of the out-of-order core of a microprocessor implementing the present invention is shown. In **Figure 4**, only those execution units which are coupled to dispatch ports 0 and 1 are shown. In the discussion which follows, it should be understood that the same principles and implementation may be applied to the functional units which are supplied instructions through dispatch ports 2, 3 and 4 of the preferred embodiment microprocessor.

As was described above, IEU[0] and a floating point execution unit cluster receive instructions through dispatch port 0 of the reservation station 20. IEU[1] and the Jump Unit 37 receive instructions through dispatch port 1. The functional units coupled to dispatch port 0 write back result data over write back bus 120 which is coupled to the reservation station 20 through write back port 0 and to the ROB 40. IEU[1] and the Jump Unit 37 which receive instructions from dispatch port 1 write back result data over write back bus 121 which is coupled to the reservation station 20 through write back port 1. In the preferred embodiment implementation, the microprocessor includes a third write back bus 122 which the memory functional units use to write back result data. Write back bus 122 is coupled to the reservation station 20 through write back port 2.

It can be seen from Figure 4 that the out-of-order core of the preferred embodiment microprocessor is provided with a number of bypassing multiplexers used to implement the present invention. For example, bypassing multiplexers 51 and 52 are coupled within the input path of IEU[0]. Similarly, bypassing multiplexers 53 and 54 are coupled to the input path of IEU[1]. Finally, as shown in Figure 4, bypass multiplexers 55 and 56 are coupled to the input path of all the floating point units which receive instructions through dispatch port 0.

The bypass multiplexers shown in Figure 4 are each coupled to receive source 1 and source 2 data from the dispatch port to which they are coupled as well as from each of the write back buses. It should be understood that with respect to Figure 4, only the supplying of source 1 and source 2 operand data to the execution units is illustrated. It should be understood that when a reservation station dispatches an instruction to an execution unit, it generally supplies an instruction opcode as well as the source 1 and source 2 operands. In addition, other information such as flag and fault information may be exchanged between units within the microprocessor, however, the conveyance of that data is not within the scope of the present invention.

It can be seen in Figure 4 that the bypassing multiplexers receive select line enable signals labeled "Bypass Control". The bypass control signals are provided by

the reservation station 20 for controlling the enabling of the bypass multiplexers. The reservation station logic determines when an operand for a dependent instruction is going to be written back over one of the write back buses as
5 result data from one of the functional units. If an instruction is pending in the reservation station and has all of its needed components except for one source operand, and that source operand is going to be the result of a previously dispatched instruction, the reservation station may schedule
10 the dispatch of the waiting instruction in advance of the receipt of the needed source operand. The reservation station then dispatches the waiting instruction and the source operand that was already stored in the reservation station to the designated functional unit for that
15 instruction. Simultaneously with the dispatch of the instruction from the reservation station to the execution unit, the reservation station provides a control signal to the appropriate bypassing multiplexer for the receiving execution unit, directing the result data from the write back
20 bus to be provided to the execution unit as the needed source operand input.

If, for example, an operand already stored in the reservation station 20 is the source 1 operand for an instruction to be dispatched to IEU[0], the reservation
25 station will dispatch the opcode and that source operand to the execution unit. The reservation station will also provide a bypass control signal to the source 1 bypass

multiplexer 51 instructing that multiplexer to propagate the source 1 operand received from the reservation station dispatch port 0 to the IEU[0] source 1 operand input. Further, if the second source operand is result data from a previously executing instruction that is being written back from execution unit IEU[1], the reservation station propagates the appropriate bypass control signal to the source 2 bypass multiplexer 52 directing it to route the result data from write back bus 121 to the source 2 operand input of IEU[0]. This allows the pending instruction in the reservation station to be dispatched earlier than if it were necessary to wait for the result data from IEU[1] to be written into the reservation station and then scheduled for dispatch to IEU[0].

15 In the preferred embodiment implementation of the present invention bypassing arrangement, two sets of source 1 and source 2 bypass multiplexers are provided for the functional units receiving instructions through dispatch port 0 of the reservation station 20. The IEU[0] functional unit has its own set of source bypass multiplexers 51 and 52 while the floating point execution cluster shares the source bypass multiplexers 55 and 56. Each of these multiplexers as well as the bypass multiplexers supplied for the other functional blocks are capable of receiving data dispatched from the reservation station as well as data bypassed from any one of the write back buses 120, 121 or 122. Separate bypass multiplexers are dedicated to the IEU[0] functional unit

because this results in less loading on the bypass.
multiplexer and thus a higher clock speed can be obtained.
The floating point execution units are more latency tolerant,
so the sharing of common bypass multiplexers 55 and 56 may be
5 more efficient than having separate multiplexers for each
unit. Alternatively, all of the execution units receiving
instructions from dispatch port 0 could share one set of
bypass multiplexers where speed considerations are less
important. Likewise, each of the execution units receiving
10 instructions from dispatch port 0 could have dedicated bypass
multiplexers supplied. In addition to facilitating a
critical speed path, having discrete bypass multiplexers 51
and 52 for the IEU[0] allows these multiplexers to include
special functionality for data manipulations that may be
15 required in propagating data to the IEU[0].

Although Figure 4 only illustrates the functional units
and bypass multiplexers coupled through dispatch port 0 and
dispatch port 1, it should be understood that equivalent
bypassing circuitry is applied to the inputs of the
20 functional units receiving instructions from dispatch ports
2, 3 and 4. In a preferred embodiment implementation, only
the Store Data functional unit receives instructions through
dispatch port 4 of the reservation station 20. Because of
this, an instruction dispatched through dispatch port 4 only
25 requires a single source operand, and thus only one bypass
multiplexer need be provided there. Of course, to maximizing
bypassing efficiency, any bypass multiplexers coupled to

execution units receiving their instruction through dispatch ports 2, 3 or 4 should be equipped for receiving dispatches not only from the reservation station 20, but also bypassed data from each of the write back buses 120, 121 and 122.

5 An additional feature worth noting is that when a source operand is bypassed directly to a functional unit, it is also written into the ROB 40 and offered to the reservation station 20. This is done because the result data on the write back bus may be required for more than one pending
10 instruction in the reservation station. Further, if for example an execution unit is receiving two bypassed operands, one from write back bus 121 as the output from IEU[1] and the other the result of a memory load data instruction being written back on write back bus 122, it is possible that the
15 result of the load data instruction will not be available due to a cache miss. In that event, the result data from IEU[1] should be buffered in the reservation station with the pending instruction and allow the receiving functional unit to carry on with other instructions until the memory
20 operation is complete. By not latching the pending data in the execution unit, and instead in the reservation station, the benefits of out-of-order execution are preserved wherein the execution unit may continue performing completely independent operations.

25 There has thus been described a method and apparatus for implementing extensive data bypassing in a complex

microprocessor. It should be understood that additional features and functionality may be appreciated by those of ordinary skill in the art. Accordingly, the present invention should only be judged in terms of the scope of the
5 claims which follow.

CLAIMS

1. An apparatus for efficient processing of instruction data in a microprocessor comprising:

an instruction buffer for buffering instructions, said instruction buffer for scheduling the dispatch of an instruction when said instruction has first and second source operands available;

a first plurality of execution units in communication with said instruction buffer, each of said execution units for receiving an instruction with first and second source operands for executing said instruction to produce result data;

a first write back path coupled to said first plurality of execution units for conveying said result data from said execution units;

at least a first and second pair of first and second source operand bypass multiplexers, each coupled between at least one of said first plurality of execution units and said instruction buffer, each bypass multiplexer responsive to control signals generated by said instruction buffer, and each coupled to said first write back path, said instruction



buffer controlling said bypass multiplexers to provide for writing result data from said write back path directly to a selected one of said plurality of execution units.

2. The apparatus of claim 1 wherein said instruction buffer comprises a first dispatch port for dispatching instructions and source operands, said first plurality of execution units including a first integer execution unit and a first plurality of floating point execution units, said first integer execution unit and said first plurality of floating point execution units each for receiving instructions dispatched through said first dispatch port and each for propagating result data over said first write back path.

3. The apparatus of claim 2 wherein said pairs of first and second source operand bypass multiplexers comprises said first pair of first and second source operand bypass multiplexers coupled to the source operand inputs of said integer execution unit and coupled to receive data from said instruction buffer and said first write back path, said first pair of first and second source operand bypass multiplexers selectively routing result data from said write back path to be source operand input data to said first integer execution



unit in response to a control signal generated by said instruction buffer..

4. The apparatus of claim 3 wherein said pairs of first and second operand bypass multiplexers further comprise said second pair of first and second source operand bypass multiplexers coupled to the source operand inputs of said plurality of floating point execution units and coupled to receive data from said instruction buffer and said first write back path, said second pair of first and second source operand bypass multiplexers selectively routing result data to be source operand input data to a selected one of said first plurality of floating point execution units in response to a control signal generated by said instruction buffer.

5. The apparatus of claim 4 wherein said instruction buffer further comprises a second dispatch port for dispatching instructions and source operands, said microprocessor further comprising a second write back path and a second integer execution unit for receiving instructions and source operands from said instruction buffer through said second dispatch port, said second integer execution unit coupled to said second write back path for conveying result data from said second integer execution unit.



6. The apparatus of claim 5 further comprising a third pair of first and second source operand bypass multiplexers coupled to the source operand inputs of said second integer execution unit and coupled to receive data from said instruction buffer and said second write back path, said third pair of first and second source operand bypass multiplexers selectively routing result data to be source operand input data to said second integer execution unit in response to a control signal generated by said instruction buffer.

7. The apparatus of claim 6 wherein said first, second and third pairs of first and second source operand bypass multiplexers are coupled to said first write back path and to said second write back path for providing for the bypassing of result data from either said first or second write back path through said first, second or third pairs of source operand bypass multiplexers.

8. The apparatus of claim 7 wherein said instruction buffer further comprises a third dispatch port for dispatching instructions and source operands, said microprocessor further comprises a memory execution unit for



receiving instructions through said third dispatch port of said instruction buffer, said microprocessor further comprising a third write back path coupled to receive result data from said memory execution unit.

9. The apparatus of claim 8 further comprising a fourth pair of first and second source operand bypass multiplexers coupled to the source operand inputs of said memory execution unit and coupled to receive data from said instruction buffer and said third write back path, said fourth pair of first and second source operand bypass multiplexers selectively routing result data to be source operand input data to said memory execution unit in response to control signals generated by said instruction buffer.

10. The apparatus of claim 9 wherein each of said pairs of first and second source operand bypass multiplexers is coupled to said instruction buffer, said first, second and third write back paths providing a bypass mechanism for result data to be routed as source operand input data by any of said pairs of first and second source operand bypass multiplexers in response to control signals generated by said instruction buffer.



11. A computer system comprising:

an input/output (I/O) means for providing a communication's interface;

a memory means coupled to said I/O means for storing instructions and computer data;

data input means coupled to said I/O means for providing data input and data output to interface with a computer user; and

microprocessor means coupled to said I/O means for executing instructions and processing computer data, said microprocessor means comprising:

an instruction buffer for buffering instructions, said instruction buffer for scheduling the dispatch of an instruction when said instruction has first and second source operands available;

a first plurality of execution units in communication with said instruction buffer, each of said execution units for receiving an instruction with first and second source operands for executing said instruction to produce result data;

a first write back path coupled to said first plurality of execution units for conveying said result data from said execution units;



at least a first and second pair of first and second source operand bypass multiplexers, each coupled between at least one of said first plurality of execution units and said instruction buffer, each bypass multiplexer responsive to control signals generated by said instruction buffer, and each coupled to said first write back path, said instruction buffer controlling said bypass multiplexers to provide for writing result data from said write back path directly to a selected one of said plurality of execution units.

12. The computer system of claim 11 wherein said instruction buffer comprises a first dispatch port for dispatching instructions and source operands, said first plurality of execution units including a first integer execution unit and a first plurality of floating point execution units, said first integer execution unit and said first plurality of floating point execution units each for receiving instructions dispatched through said first dispatch port and each for propagating result data over said first write back path.

13. The computer system of claim 12 wherein said pairs of first and second source operand bypass multiplexers comprises said first pair of first and second source operand



bypass multiplexers coupled to the source operand inputs of said integer execution unit and coupled to receive data from said instruction buffer and said first write back path, said first pair of first and second source operand bypass multiplexers selectively routing result data from said write back path to be source operand input data to said first integer execution unit in response to a control signal generated by said instruction buffer.

14. The computer system of claim 13 wherein said pairs of first and second operand bypass multiplexers further comprises said second pair of first and second source operand bypass multiplexers coupled to the source operand inputs of said plurality of floating point execution units and coupled to receive data from said instruction buffer and said first write back path, said second pair of first and second source operand bypass multiplexers selectively routing result data to be source operand input data to a selected one of said first plurality of floating point execution units in response to a control signal generated by said instruction buffer.

15. The computer system of claim 14 wherein said instruction buffer further comprises a second dispatch port for dispatching instructions and source operands, said microprocessor further comprising a second write back path



and a second integer execution unit for receiving instructions and source operands from said instruction buffer through said second dispatch port, said second integer execution unit coupled to said second write back path for conveying result data from said second integer execution unit.

16. The computer system of claim 15 further comprising a third pair of first and second source operand bypass multiplexers coupled to the source operand inputs of said second integer execution unit and coupled to receive data from said instruction buffer and said second write back path, said third pair of first and second source operand bypass multiplexers selectively routing result data to be source operand input data to said second integer execution unit in response to a control signal generated by said instruction buffer.

17. The computer system of claim 16 wherein said first, second and third pairs of first and second source operand bypass multiplexers are coupled to said first write back path and to said second write back path for providing for the bypassing of result data from either said first or second write back path through said first, second or third pairs of source operand bypass multiplexers.



18. The computer system of claim 17 wherein said instruction buffer further comprises a third dispatch port for dispatching instructions and source operands, said microprocessor further comprises a memory execution unit for receiving instructions through said third dispatch port of said instruction buffer, said microprocessor further comprising a third write back path coupled to receive result data from said memory execution unit.

19. The computer system of claim 18 further comprising a fourth pair of first and second source operand bypass multiplexers coupled to the source operand inputs of said memory execution unit and coupled to receive data from said instruction buffer and said third write back path, said fourth pair of first and second source operand bypass multiplexers selectively routing result data to be source operand input data to said memory execution unit in response to a control signal generated by said instruction buffer.

20. The apparatus of claim 19 wherein each of said pairs of first and second source operand bypass multiplexers is coupled to said instruction buffer, said first, second and third write back paths for providing a bypass mechanism for



result data to be routed as source operand input data by any of said pairs of first and second source operand bypass multiplexers in response to a control signal generated by said instruction buffer.

