

1

# METHOD AND APPARATUS FOR PROVIDING FINER MARKING GRANULARITY FOR FIELDS WITHIN OBJECTS

## Related Application

This application is related to the subject matter in a pending U.S. patent application, entitled "Supporting Space-Time Dimensional Program Execution by Selectively Versioning Memory Updates," by inventors Shailender Chaudhry and Marc Tremblay, having Ser. No. 09/313,229 and a filing date of May 17, 1999.

The subject matter of this application is also related to the subject matter in a co-pending non-provisional application by the same inventor(s) as the instant application and filed on the same day as the instant application entitled, "Using Time Stamps to Improve Efficiency in Marking Fields Within Objects," having Ser. No. 09/327,399, and filing date Jun. 7, 1999.

## BACKGROUND

### 1. Field of the Invention

The present invention relates to performance enhancements in object-oriented programming systems. More specifically, the present invention relates to a method and an apparatus that supports finer marking granularity for fields within objects defined within an object-oriented programming system.

### 2. Related Art

As increasing semiconductor integration densities allow more transistors to be integrated onto a microprocessor chip, computer designers are investigating different methods of using these transistors to increase computer system performance. Some recent computer architectures exploit "instruction level parallelism," in which a single central processing unit (CPU) issues multiple instructions in a single cycle. Given proper compiler support, instruction level parallelism has proven effective at increasing computational performance across a wide range of computational tasks. However, inter-instruction dependencies generally limit the performance gains realized from using instruction level parallelism to a factor of two or three.

Another method for increasing computational speed is "speculative execution" in which a processor executes multiple branch paths simultaneously, or predicts a branch, so that the processor can continue executing without waiting for the result of the branch operation. By reducing dependencies on branch conditions, speculative execution can increase the total number of instructions issued.

Unfortunately, conventional speculative execution typically provides a limited performance improvement because only a small number of instructions can be speculatively executed. One reason for this limitation is that conventional speculative execution is typically performed at the basic block level, and basic blocks tend to include only a small number of instructions. Another reason is that conventional hardware structures used to perform speculative execution can only accommodate a small number of speculative instructions.

What is needed is a method and apparatus that facilitates speculative execution of program instructions at a higher level of granularity so that many more instructions can be speculatively executed.

One challenge in designing a system that supports speculative execution is to detect a rollback condition. A rollback

2

condition can occur in a number of situations. For example, a rollback condition occurs when a speculative thread that is executing program instructions in advance of a head thread reads from a memory element before the head thread performs a write to the memory element. In this case, the speculative thread must "rollback" so that it can read the value stored by the head thread. A rollback condition can be detected by "marking" memory elements as they are read by the speculative thread so that the head thread can subsequently determine if the memory elements have been read by the speculative thread. Unfortunately, using a separate marking indicator for each memory element can consume a large amount of memory, which can reduce cache hit rates and thereby degrade system performance.

What is needed is a method and an apparatus for marking memory elements that does require a large amount of memory for storing marking indicators.

## SUMMARY

One embodiment of the present invention provides a system that facilitates marking of objects defined within an object-oriented programming system to keep track of accesses to fields within the objects. The system operates by receiving a reference to a field within an object, and identifying a marking bit within the object that is associated with the field. Note that each marking bit within the object is associated with a different subset of fields within the object. Next, the system sets the marking bit to indicate that at least one field within the associated subset of fields has been referenced. Finally, the system performs the reference to the field.

In one embodiment of the present invention, the object includes N marking bits numbered 0, 1, 2, . . . , N-1 and M fields numbered 0, 1, 2, . . . , M-1. In this embodiment, the system identifies the marking bit associated with the field by starting with a field number for the field, and applying a modulo N operation to the field number to produce a number for the associated marking bit. In a variation on this embodiment, N is a power of two.

In one embodiment of the present invention, the system supports space and time dimensional execution. To this end, the system supports a head thread that executes program instructions and a speculative thread that executes program instructions in advance of the head thread. The head thread accesses a primary version of the object and the speculative thread accesses a space-time dimensioned version of the object. In this embodiment, the steps of identifying the marking bit and setting the marking bit take place for a read operation by the speculative thread.

In a variation on this embodiment, there exists a separate set of marking bits for write operations. In this variation, if the reference is a write operation by the speculative thread, the steps of identifying the marking bit and setting the marking bit involve the separate set of marking bits. Upon a subsequent write operation to the field by the head thread, the head thread writes to both the primary version and the space-time dimensioned version if the marking bit is unset, and otherwise writes to the primary version.

In one embodiment of the present invention, during a subsequent write, operation to the field by the head thread, the system determines if the marking bit associated with the field has been set by executing a special bit extract instruction to examine the marking bit.

In one embodiment of the present invention, if the object is an array object with N marking bits numbered 0, 1, 2, . . . , N-1 and M array elements numbered 0, 1, 2, . . . , M-1,