

**102.** The base pointer of the currently active method preferably is stored in register **R5** as noted previously. In general, the base pointer for the active method may be computed by the JVM **108** while executing the invoke bytecode of the active method.

[**0040**] Sequence **510** depicts the state of the D-RAMset **126** when method A calls method B. In accordance with the preferred embodiments of the invention, the local variables (LVB) associated with method B are stacked in storage space **512** generally adjacent LVA (“on top of” LVA when viewed as in **FIG. 5**). Following arrow **505**, the base pointer for LVA (PTR LVA) preferably is also stored in the D-RAMset **126** adjacent (e.g., below) the LVB data at location **504A**. Thus, the two local variable sets LVA and LVB may be separated by the base pointer (PTR LVA) for LVA and possibly other data. Once the base pointer **504** for LVA is stored adjacent (below) the reserved space for the LVB data set **502**, register **R5** is updated (i.e., loaded) with a base pointer **514** for use with the LVB data set.

[**0041**] Following arrow **507** to time sequence **520**, when method C is invoked (called by method B), the base pointer for method B (PTR LVB) is stored in location **514A** which may be on top of LVB and below PTR LVC as shown and register **R5** is loaded with the base pointer **524** (PTR LVC) to the base of the LVC data set **522**. Method C’s local variables (LVC) are allocated to storage space **522** which generally is adjacent (on top of) LVB **512** and PTR LVB **514A** as shown. The PTR LVB value is stored in location **514A** according to a similar calculation as that described above.

[**0042**] **FIG. 6** illustrates the return process as each method (Methods C and then B) completes and returns to its calling method (methods B and then A). Beginning with time sequence **530** in which the local variable frame comprises LVA, LVB, and LVC along with pointers PTR LVA and PTR LVB for LVA and LVB, method C completes. Control returns to method B and LVB’s base pointer is loaded from location **514A** into register **R5** as shown by arrow **519** at time sequence **532** by accessing PTR LVB through a load instruction that include a fixed offset from PTR LVC as a target address. Then, when method B completes, LVA’s pointer (PTR LVA) is loaded into register **R5** from location **504A** as illustrated by arrow **521** at time sequence **534**. The base pointers may be retrieved from their locations in data cache **126** by loading the value located at the location pointed by the currently active method’s base pointer minus an offset (e.g., 1).

[**0043**] In accordance with preferred embodiments of the invention, the D-RAMset **126** is configured to provide any one or more or all of the following properties. The implementation of the D-RAMset **126** to provide these properties is explained in detail below. The local variables and pointers stored in the D-RAMset **126** preferably are “locked” in place meaning that, although the D-RAMset **126** is implemented as cache memory, eviction of the local variables generally can be prevented in a controlled manner. The locking nature of the D-RAMset **126** may be beneficial while a method executes to ensure that no cache miss penalty is incurred. Additionally, write back of valid, dirty local variables to main memory **106** is avoided in at least some situations (specified below). Further, mechanisms can be employed in the event that the D-RAMset **126** has insufficient capacity to

accommodate all desired local variables. Further still, once a method has completed, the portion of the D-RAMset allocated for the completed method’s local variables remains marked as “valid.” In this way, if and when such methods or any new methods are executed and re-use the RAMset space (such as that described in one or more of the copending applications mentioned above), such methods’ associated local variables will be mapped to the same portion of the D-RAMset. If the RAMset lines are already marked as valid, access to those new local variables may not generate any misses. Retrieval of data from memory in this situation is unnecessary because the local variables only have significance while a method executes and a newly executing method first initializes all of its local variables before using them. Not generating misses and thus avoiding fetching lines from external memory reduces latency and power consumption. After a relatively short period of time following the start of a Java program execution, all relevant lines of the RAMset are marked as valid and accesses to local variables of newly called methods do not generate misses, thereby providing superior performance of a “0-wait state memory.” Furthermore, the cache properties of RAMset allow discarding or saving of the data in main memory whenever required.

[**0044**] In accordance with a preferred embodiment of the invention, the local variables (LVA-LVC) and associated pointers (PTR LVA-PTR LVC) may be stored in D-RAMset **126**. The D-RAMset **126** may be implemented in accordance with the preferred embodiment described below and in copending applications entitled “Cache with multiple fill modes,” filed Jun. 9, 2000, Ser. No. 09/591,656; “Smart cache,” filed Jun. 9, 2000, Ser. No. 09/591,537; and publication no. 2002/0065990, all of which are incorporated herein by reference.

[**0045**] As described in greater detail below, in the preferred embodiment, the data storage **122** (**FIG. 3**) preferably comprises a 3-way cache with at least one cache way comprising D-RAMset **126**. The D-RAMset (“RAMset”) cache **126** may be used to cache a contiguous block of memory (e.g., local variables and pointers as described above) starting from a main memory address location. The other two cache ways **124** may be configured as RAMset cache memories, or use another architecture as desired. For example, the data storage **122** may be configured as one RAMset cache **126** and a 2-way set associative cache **124**. As such, the data storage **122** generally comprises one or more forms of cache memory. The instruction storage **130** may be similarly configured if desired.

[**0046**] In operation, the processor’s core **102** may access main memory **106** (**FIG. 1**) within a given address space. If the information at a requested address in main memory **106** is also stored in the data storage **122**, the data is retrieved from the data cache **124**, **126**. If the requested information is not stored in data cache, the data may be retrieved from the main memory **106** and the data cache **124**, **126** may be updated with the retrieved data.

[**0047**] **FIG. 7** illustrates a more detailed block diagram of the data storage **122** in accordance with a preferred embodiment with a RAMset cache and a two-way set associative cache. A cache controller **222** may control operation of the data storage **122**. The controller **222** may be communicatively coupled to the data storage **122** and to other compo-