

may be fetched from main memory **106** and loaded into the data array **238** of the RAMset. A hit in the RAMset logic preferably takes precedence over the normal cache logic. The standard logic of the two-way cache generates a miss when the RAMset logic generates a hit. Information can reside in both the RAMset and the two-way cache without causing any misbehavior; the duplicated cache entry in the 2-way cache will eventually be evicted by the replacement mechanism of the two-way cache because such data will not be used. However, in the preferred embodiment the data mapped onto a RAMset is first removed from the cache to avoid a data coherency problem. When configured as a RAMset, data array **238a, b, c** can be configured as a local RAM or as a cached segment depending on the setting of a suitable configuration bit (e.g., LR/C bit **231**). However, even when configured as a local RAM, individual valid bits may be updated but misses do not generate accesses to the external memory.

**[0055]** To configure a RAMset for operation, the Full\_set\_tag register **232** preferably is loaded with a start address (set\_start\_addr) and the RAM\_fill\_mode bit **224** is configured to a desired fill mode. The circuitry for filling the cache can be the same as that used to fill lines of the set associative cache. At least one fill mode may be implemented and is referred to as a “line-by-line” fill mode as described below. Other fill modes may be implemented if desired such as the “set fill” mode described in at least one of the documents incorporated by reference.

**[0056]** For the line-by-line fill (RAM\_fill\_mode=0), the global valid bit **34** is set to “1” and each of the valid entry bits **237** is set to “0” when the Full\_set\_tag register **232** is loaded with the starting address. At this point, the data array **238** is empty (it is assumed that the Cache\_Enable bit **226** is set to “1” to allow operation of the data storage **122**). Upon receiving an address from the core **120**, a valid entry bit **237** is selected based on the low order bits of the address. As provided above, if the RAMset is 16 Kbytes in size, organized as an array of 1K×16 bytes, where 16 bytes is equivalent to a block line in the associated 2-way cache, the Full\_set\_TAG register **232** may store 18 bits [31:14] of the starting address. The address indexing each entry of the RAMset (ADDR[L]) may include 10 bits [13:4] while the data address used to access one data value in the line may include 4 bits [3:0] (assuming data accesses are 1 byte). In Java, local variables comprise four byte entities but, as explained previously, the RAMset may be shared between local variables and other, possibly critical, data. A line of the data array **238** (at ADDR[L]) is loaded from main memory **106** each time that a miss situation occurs because the comparator **240** determines a match between ADDR[H] and the content of Full\_set\_TAG, the Global valid bit **34** is set to “1” and the valid bit **237** associated with the line at ADDR[L] is “0”. The state of the RAMset in this mode of operation is also referred to as the cache policy “CP” state. This situation indicates that the selected line is mapped to the RAMset, but has not yet been loaded into the RAMset’s data array **238**. When the line is loaded into the data array **238** from main memory **106**, the valid bit **237** corresponding to the line is set to “1”.

**[0057]** This loading procedure (resulting in the valid bit being set to indicate the presence of valid data) has the same time penalty as a normal cache line load, but the entry will remain locked in the RAMset (i.e., the valid bit will remain

set) unless the content of the Full\_Set\_Tag is changed and, therefore, the processing device will not be penalized on a subsequent access. As such, the lines used by a completed method remain valid so that re-using the lines by subsequent methods does not necessitate accesses to main memory **106**. Further, freeing the local variable space for a completed method generally only involves disregarding the relevant base pointer. Further still, there is no need to copy back local variables upon to main memory **106** upon completion of a method because such extinct local variables are not used any more.

**[0058]** In some situations, the capacity of the D-RAMset **126** may not be sufficient to hold all desired local variables. In accordance with at least one embodiment, excess local variables may be stored in the non-D-RAMset data arrays **238**. In accordance with other embodiments, a larger block of local variables (i.e., larger than just the excess local variables) may be mapped to the non-D-RAMset cache ways. During the “invoke” bytecodes, that initiates a method call, the local variable size of the called method is known by the JVM **108**. The JVM also knows the total RAMset size (via a readable configuration register) and the RAMset size already utilized. Therefore, based on this information, the JVM may or may not decide to map the new local variable area onto the RAMset. A method may have a large chunk of local variables and not use them on each call. Therefore, mapping those local variables onto the RAMset may force unnecessary RAMset management of the base pointer and saving/restoring of local variables of calling methods or may cause more frequent overflow of a subsequently called method. Instead, the JVM **108** may map the methods with larger chunks of local variables onto the non-RAMset data cache and thus preserve more space in the RAMset for methods with a smaller number of local variables. In some embodiments, many methods may have less than 10 local variables and almost all methods have less than about 40 local variables, but, of course, these numerical characterizations are application dependent. For methods with many local variables, the system may map those local variables outside the RAMset avoiding penalizing other methods. This technique is generally transparent for the return mechanism because of the management of the PTR\_LV of the calling method. Upon completion of a method, the lines containing that method’s local variables may remain marked as valid. As noted above, maintaining such lines marked as valid avoids generating misses in calls of new methods.

**[0059]** In accordance with some embodiments, more than one contiguous block of external memory **106** may be mapped onto the D-RAMset’s data array **238**. As illustrated in **FIG. 9**, for example, two contiguous blocks **600** and **602** of external memory **106** may be mapped onto the D-RAMset **126**. Block **600** comprises 16K of contiguous bytes from the address range of 0×0000 to 0×3FFF. Similarly, block **602** comprises 16K of contiguous bytes from the address range of 0×8000 to 0×BFFF. One block **600, 602** at a time may be mapped onto the D-RAMset **126** by reprogramming the D-RAMset’s Full\_set\_tag register **232** as explained previously.

**[0060]** A plurality of commands may be implemented in connection with the data storage **122**. Such commands may include, without limitation, D-RAMset-Clean, D-RAMset-Flush, and D-RAMset-policy-set. In addition to valid bits **237** for each line, a dirty bit also may be provided to indicate