Analysis of a Shared Way Set Associative Cache

José Luis Hamkalo, Bruno Cernuschi-Frías [1]

[1] Facultad de Ingeniería
Universidad de Buenos Aires
and CONICET
Paseo Colón 850 (1063)
Buenos Aires, ARGENTINA
{jhamkal@fi.uba.ar}

Abstract—

A new cache memory organization called Shared Way Set Associative (SWSA) is analyzed in this work. The SWSA cache uses two memory banks with different sizes. A placement policy that uses bit map indexing is given for SWSA caches as well as an expression for the associativity. Different LRU related replacement policies are analyzed. The performance of SWSA caches was measured for the SPEC95 benchmarks using trace driven simulations. It was found that SWSA caches of various associativities usually perform better than two-way set associative caches of equivalent size. An analysis of the results using the D3C model is done. Also SWSA caches are compared with victim caches. An additional benefit of SWSA caches is that non power of two total cache sizes may be used. Hence, when SWSA caches are used, a very precise adjustment of the total cache size needed for a given workload is possible. For the SPEC95 benchmarks, savings of 33 percent for the total cache size are obtained with SWSA caches relative to two-way set associative caches that produce nearly the same miss rate.

Keywords— **Cache, associativity, replacement policy.**

I. Introduction

Set associative caches frequently use bit map indexing to access the sets [SMI82] [PAT95]. Thus the memory banks used for the implementation of the cache ways are constrained to power of two sizes. For a two-way set associative cache [HIL89], the constrain of banks of equal size implies a cache of power of two total size. If the restriction of banks of equal sizes is left aside, a new class of cache memory organization may be defined. In this work caches that use two memory banks with the previously mentioned asymmetry are analyzed. The asymmetry introduced makes it necessary to establish the placement policy and how the sets are conformed. Also the mechanisms for the replacement policy must be specified [BEL66]. It will be shown that for these caches, non integer associativities between 1 and 2 may be defined, and that direct mapped and two-way set associative caches may be considered as special cases of our proposed organization.

Analytical and experimental studies of our proposed organization are presented in this work. It will be shown that caches with associativities even much smaller than two may perform better than two way set associative caches of comparable size. Conventional schemes that use higher associativities (for example four way associativity) provide better miss rates statistics, but at the expense of higher complexity and access times [HIL88]. Thus, if an associativity of degree two is considered good for a given workload, the use of our organization gives an improvement over the miss rate statistics and the additional benefit of allowing a greater number of choices for the total cache size, because total cache sizes other than powers of two may be used.

II. Shared Way Set Associative (SWSA) caches

A. *Definition of Shared Way Set Associative caches*

A two-way set associative cache of size C may be thought as the result of cutting a direct mapped cache in two equal halves of size $C/2$. The sets are formed by creating a one to one association, between the lines of the two halves (see figure 1a).

For the different case in which the direct mapped cache is cut in two parts of different sizes C_1 and C_2, to build the corresponding sets an association between the lines of the two banks as shown in figure 1b could be used.

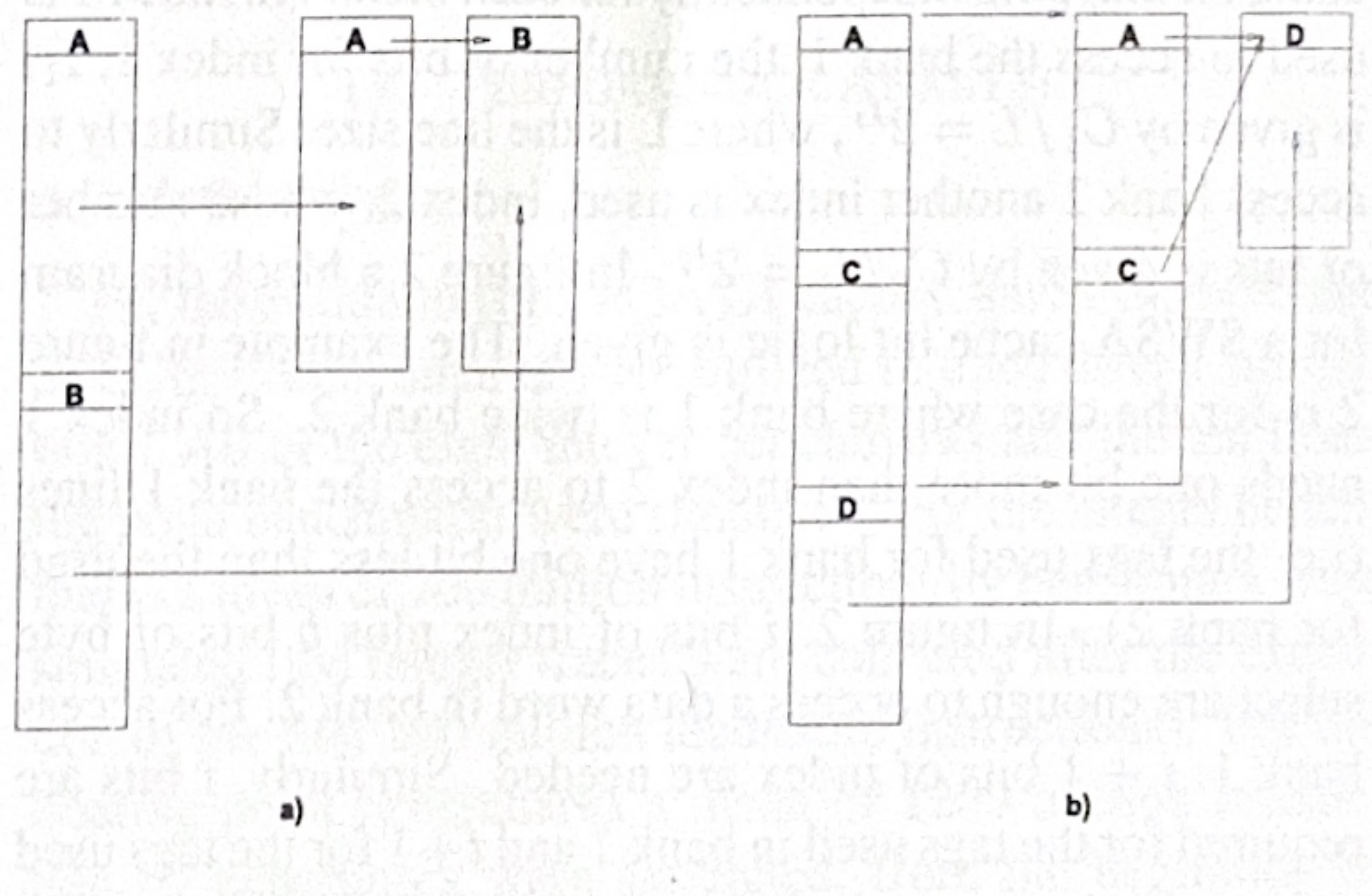

Fig. 1. Set associativity from direct mapped caches

From figure 1b, more than one line in the first bank is associated with only one line in the second bank. Thus different sets share one line in the second bank or second way. The "degree of sharing" is defined as the number of blocks in

Analysis of a Shared Way Set Associative Cache

José Luis Hamkalo, Bruno Cernuschi-Frías [1]

[1] Facultad de Ingeniería
Universidad de Buenos Aires
and CONICET
Paseo Colón 850 (1063)
Buenos Aires, ARGENTINA
{jhamkal@fi.uba.ar}

Abstract—

A new cache memory organization called Shared Way Set Associative (SWSA) is analyzed in this work. The SWSA cache uses two memory banks with different sizes. A placement policy that uses bit map indexing is given for SWSA caches as well as an expression for the associativity. Different LRU related replacement policies are analyzed. The performance of SWSA caches was measured for the SPEC95 benchmarks using trace driven simulations. It was found that SWSA caches of various associativities usually perform better than two-way set associative caches of equivalent size. An analysis of the results using the D3C model is done. Also SWSA caches are compared with victim caches. An additional benefit of SWSA caches is that non power of two total cache sizes may be used. Hence, when SWSA caches are used, a very precise adjustment of the total cache size needed for a given workload is possible. For the SPEC95 benchmarks, savings of 33 percent for the total cache size are obtained with SWSA caches relative to two-way set associative caches that produce nearly the same miss rate.

Keywords— **Cache, associativity, replacement policy.**

I. Introduction

Set associative caches frequently use bit map indexing to access the sets [SMI82] [PAT95]. Thus the memory banks used for the implementation of the cache ways are constrained to power of two sizes. For a two-way set associative cache [HIL89], the constrain of banks of equal size implies a cache of power of two total size. If the restriction of banks of equal sizes is left aside, a new class of cache memory organization may be defined. In this work caches that use two memory banks with the previously mentioned asymmetry are analyzed. The asymmetry introduced makes it necessary to establish the placement policy and how the sets are conformed. Also the mechanisms for the replacement policy must be specified [BEL66]. It will be shown that for these caches, non integer associativities between 1 and 2 may be defined, and that direct mapped and two-way set associative caches may be considered as special cases of our proposed organization.

Analytical and experimental studies of our proposed organization are presented in this work. It will be shown that caches with associativities even much smaller than two may perform better than two way set associative caches of comparable size. Conventional schemes that use higher associativities (for example four way associativity) provide better miss rates statistics, but at the expense of higher complexity and access times [HIL88]. Thus, if an associativity of degree two is considered good for a given workload, the use of our organization gives an improvement over the miss rate statistics and the additional benefit of allowing a greater number of choices for the total cache size, because total cache sizes other than powers of two may be used.

II. Shared Way Set Associative (SWSA) caches

A. *Definition of Shared Way Set Associative caches*

A two-way set associative cache of size C may be thought as the result of cutting a direct mapped cache in two equal halves of size $C/2$. The sets are formed by creating a one to one association, between the lines of the two halves (see figure 1a).

For the different case in which the direct mapped cache is cut in two parts of different sizes C_1 and C_2, to build the corresponding sets an association between the lines of the two banks as shown in figure 1b could be used.

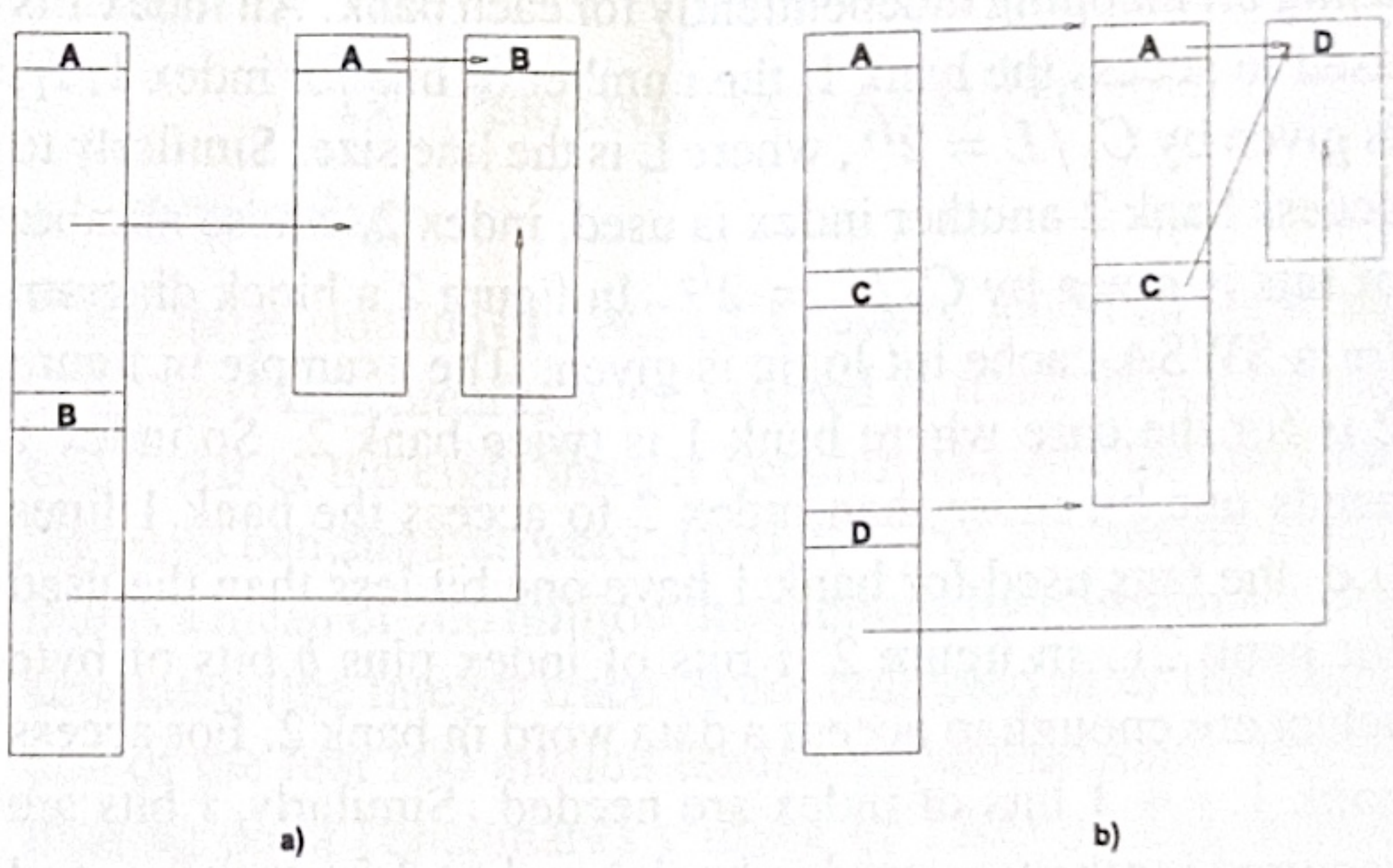

Fig. 1. Set associativity from direct mapped caches

From figure 1b, more than one line in the first bank is associated with only one line in the second bank. Thus different sets share one line in the second bank or second way. The "degree of sharing" is defined as the number of blocks in

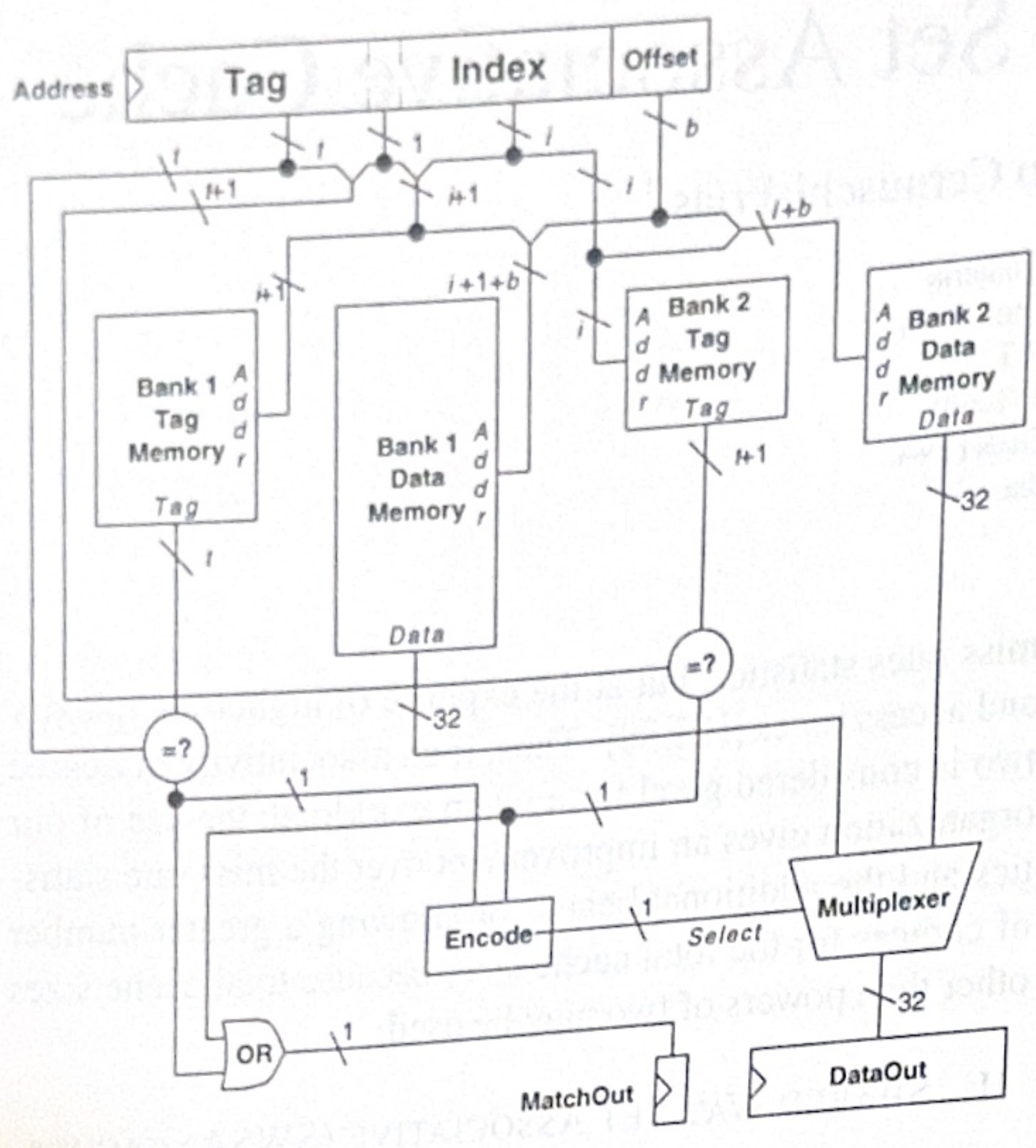

Fig. 2. Block diagram for a SWSA cache hit logic

the first bank that share the same block in the second bank. Hence the degree of sharing may be calculated as the ratio between the number of blocks in the first and second banks. The previously described organization is called Shared Way Set Associative cache (SWSA).

For a SWSA cache the access of the sets could be done by using bit mapping independently for each bank. An index 1 is used to access the bank 1, the number of bits for index 1, I_1, is given by $C_1/L = 2^{I_1}$, where L is the line size. Similarly to access bank 2 another index is used, index 2, whose number of bits is given by $C_2/L = 2^{I_2}$. In figure 2 a block diagram for a SWSA cache hit logic is given. The example in figure 2 is for the case where bank 1 is twice bank 2. So index 1 needs one bit more than index 2 to access the bank 1 lines (i.e. the tags used for bank 1 have one bit less than the used for bank 2). In figure 2, i bits of index plus b bits of byte select are enough to access a data word in bank 2. For access bank 1, $i+1$ bits of index are needed. Similarly, t bits are required for the tags used in bank 1 and $t+1$ for the tags used in bank 2. A tag match (hit) in bank 1 or bank 2 will assert the MatchOut bit and the proper data word will be chosen with the multiplexer and placed in DataOut.

B. Degree of associativity for SWSA caches

A quantitative expression for the associativity of SWSA caches is given below.

Definition: the associativity n for a SWSA cache is give by,

$$n = 1 + C_2/C_1 \quad (1)$$

where C_1 and C_2 are the sizes of the banks 1 and 2 respectively. For example if bank 1 has 16Kbytes and bank 2 has 8kbytes, then, the SWSA cache has 24Kbytes and from (1) the associativity of the cache is $n = 1.5$. It should be noted that the two-way set associative and direct mapped caches are special cases of SWSA caches. In a two-way set associative cache, the sizes of the banks C_1 and C_2 are equal, and the the ways in the second bank are not shared, but exclusive to each line in the bank 1 and from (1) it is $n = 2$. For a direct mapped cache, the bank 2 does not exist and this could be written as $C_2 = 0$, so the relation C_2/C_1 is 0 and from (1) is $n = 1$.

III. THE REPLACEMENT POLICIES

A. The swap replacement policy

We first analyze a replacement policy called "swap" [JOU90]. In [JOU90], the swap replacement policy was used with victim caches. Here the swap replacement policy keeps the most recently referenced line in the set, in the bank 1. For example a reference that produces a hit in the bank 1 produces no changes in the cache. If the hit is in the second bank, the block in bank 2 (indexed with I_2 bits) is swapped with the block checked in bank 1 (indexed with I_1 bits). For the case of a miss, the indexed block in bank 1 is copied to bank 2 (i.e., the corresponding block in bank 2 is taken out from the cache), and the incoming block is placed in bank 1. A useful property of the swap replacement policy is that it has the property of inclusion. For example all the references that hit in a direct mapped cache (DM) of size $C_{DM} = C_1$ also hit in a SWSA cache of size $C_{SWSA} = C_1 + C_2$ that uses the swap replacement policy. This is true because the swap replacement policy guarantees that at any given time the DM cache and the bank 1 in the SWSA cache have exactly the same content, and so the second bank in the SWSA cache gives the possibility of additional hits for the SWSA cache.

A similar analysis could be made when two SWSA caches of different associativities (sizes) are compared. Consider two SWSA caches, SWSA and SWSA', of size $C_{SWSA} = C_1 + C_2$ and $C'_{SWSA} = C_1 + C'_2$ respectively, with $C'_2 > C_2$. Similarly to the previous example, the banks 1 for the two caches have exactly the same content. The references that could not be contained in the banks 1 are directed to the banks 2. From the banks 2 point of view, they are direct mapped caches of sizes C_2 and C'_2 that are receiving the same pattern of memory requests. Because direct mapped caches have the property of inclusion, all the hits in the bank 2 of size C_2 are included in the SWSA' bank 2 of size C'_2

$(C_2' > C_2)$. So all the references that hit in the SWSA cache also hit in the SWSA' cache, and the property of inclusion is maintained. With the property of inclusion, SWSA caches with successively greater banks 2 (from a DM cache of size C_1 with $C_2 = 0$, to a two-way set associative cache of size $2C_1$ with $C_2 = C_1$) have always lower or equal miss rates.

B. *Other replacement policies*

Here the utilization of three alternative replacement policies for SWSA caches are analyzed: a LRU replacement policy, a LRU replacement policy with reallocation, and a simple 1 bit LRU replacement policy.

B.1 A LRU replacement policy

When a SWSA cache is checked for a hit or a miss the two address tags actually cached in bank 1 and bank 2 are checked against the address requested by the processor.

In the swap replacement policy, the candidate for replacement always is the block in bank 2, even if it was more recently referenced than the corresponding block in bank 1. In general, a LRU replacement policy for SWSA caches must maintain the temporal information of block usage for the groups that conform all the sets that share the same block in bank 2. These groups are called the replacement sets.

A possible LRU replacement policy for SWSA caches could be:

hit: the requested item hits in bank 1 or bank 2, only the replacement set LRU information is updated.

miss: the block replaced is the least recently used of the two blocks that are checked in bank 1 and bank 2. The replacement set LRU information is updated.

B.2 A LRU replacement policy with reallocation

The previously described LRU replacement policy limits the utilization of the replacement set temporal information of blocks usage, only to the two blocks that are checked with index 1 and index 2. A LRU replacement policy with reallocation not only updates the temporal information in the replacement set, but also moves blocks between the cache banks when this is possible. This data moving is done with the objective of replacing the least recently used block in the replacement set that could be replaced. This will be possible when the block checked in bank 2 does not map in the same place that the block that is checked in bank 1. The last may happen because index 2 uses one or more bits less than index 1 ($C_1 > C_2$, i.e. $I_1 > I_2$) and these bits could be different for these two blocks. This fact creates the need of including a third block as a candidate at replacement time if a more effective LRU replacement policy is wanted. For example such three blocks may be blocks A, C and D in figure 1b.

A possible LRU replacement policy with reallocation for SWSA caches is given below:

hit: the requested item hits in bank 1 or bank 2, only the replacement set LRU information is updated.

miss:

a) the block checked in bank 2 maps in the same place that the block checked in bank 1. The least recently used of this two blocks is replaced.

b) the block checked in bank 2 maps in a different block that the one checked in bank 1.

b1) this third block is the least recently used and it is replaced by the block checked in bank 2 (reallocation). The block that caused the miss is placed in bank 2.

b2) this third block is not the least recently used. The replace is done as in case a).

Also for cases a) and b) the replacement set LRU information is updated.

B.3 A 1 bit LRU replacement policy

Here a simple LRU related replacement policy that uses one bit associated with each block in the bank 2 is analyzed. This policy works as follows:

Hit: if the hit is in bank 1 the corresponding LRU bit is set to 0. If the hit is in bank 2 the corresponding LRU bit is set to 1.

Miss: if the LRU bit is checked to be 0, the block in bank 2 is replaced and the LRU bit is set to 1, else the block in bank 1 is replaced and the LRU bit is set to 0.

It should be noted that the four replacement policies considered in this work, for the particular case of two-way set associative caches, coincide with the standard LRU replacement policy.

IV. Performance Analysis

A. *Methodology*

For the evaluation of SWSA L1 caches, the SPECint95 and SPECfp95 benchmarks were utilized in trace driven simulations. All of the eight integer benchmarks and the ten floating point benchmarks were simulated. For the integer benchmarks a mean of 700 million instructions by benchmark were simulated (the integer traces were collected after the execution of the first 500 million load/store instructions). For the floating point benchmarks a mean of 1200 million instruction by benchmark were simulated, from the beginning of the benchmarks. For the collection of the traces the ATOM tool was used on a Alpha Station 5000 (Alpha processor instruction set architecture) [SRI94] . For the caches simulated, the line size was 32 bytes and the swap replacement policy was used. In section V a 1 bit LRU replacement policy is considered too.

The mean miss rates obtained for instruction and data SWSA L1 caches (SW) are shown in figure 3. The horizontal

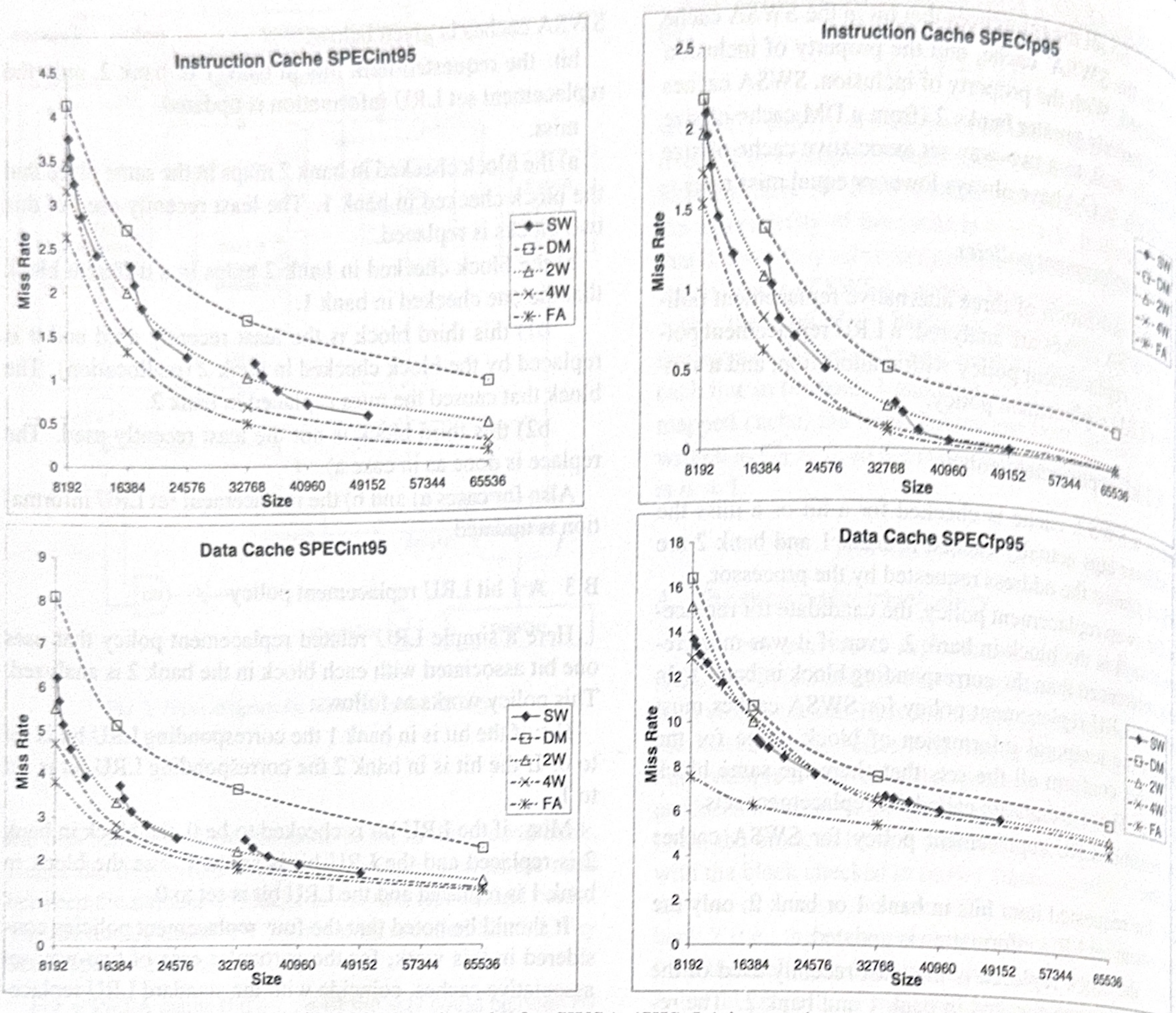

Fig. 3: Mean miss rate for the SPEC95 benchmarks for SWSA (SW) L1 instruction and data caches. Direct mapped (DM), two-way set associative (2W), four-way set associative (4W), and fully-associative (FA) cache curves are plotted for comparison purposes. Each contiguous curve for the SW curve has five points corresponding to associativities $1\frac{1}{32}, 1\frac{1}{16}, 1\frac{1}{8}, 1\frac{1}{4}$ and $1\frac{1}{2}$.

axes in the figures represent the total cache size. Each cache size corresponds to a given non integer associativity for the SWSA curves. The plotted points have the sequence of associativities $1\frac{1}{32}, 1\frac{1}{16}, 1\frac{1}{8}, 1\frac{1}{4}$ and $1\frac{1}{2}$ which form the contiguous curves corresponding to SW curves shown in fig. 3. For example, the first point at the right of a power of two point has an associativity of $1\frac{1}{32}$, the next of $1\frac{1}{16}$, and so on. Direct mapped (DM), two-way set associative (2W), four-way set associative (4W) and fully-associative (FA) cache curves are plotted too for comparisons purposes. Linear scales are used for the four figures in order to give a precise visual image of the relative cache sizes related to their performance.

B. Analysis of the results

In this section the miss rates obtained from the simulations shown in figure 3 are analyzed. A direct mapped cache consists of only one logical bank of size C_1, as it was described in section II. The addition of a small second bank of size C_2 (for example the points in figure 3 with associativities $1\frac{1}{32}$), gives significant reductions in the miss rate. As the size of the second bank C_2 increases, the diminution in the miss rates persists, with a very sharp tendency. It may be observed that the curve for the SWSA caches intersects the 2W tendency curve in the proximity of the points corresponding to associativities $1\frac{1}{16}$ and $1\frac{1}{8}$. This indicates that SWSA roughly attains the same level of performance than two way set associative caches of comparable size, with associativities between $1\frac{1}{16}$ and $1\frac{1}{8}$. For these caches the associativity is very low, the degree of sharing is 16 and 8 respectively. The points with associativities of $1\frac{1}{4}$ and $1\frac{1}{2}$, are usually under the two-way tendency curve. It may be observed that for large cache size, a SWSA cache of associativity $1\frac{1}{2}$ attains a very similar miss rate as the first two-way set associative cache of greater size. For example in figure 3, the caches of sizes 48K ($n = 1.5$) and 64k ($n = 2$), present very similar miss rates, but the two-way cache is 33 per cent greater. This may be explained noting that SWSA caches can capture the application working set with the best total cache size needed. In the next section a deeper analysis for these cases is given.

C. Analysis using the D3C model

In this section the performance of SWSA caches are analyzed using the D3C model [HAM99]. In the D3C model a non compulsory miss is of conflict type if the LRU distance for the reference that produces the miss is less or equal than the number of blocks in the cache. Otherwise the reference is a capacity miss i.e., the LRU distance for that reference is greater than the number of blocks in the cache.

Capacity and conflict misses for SWSA (SW) caches are plotted in figure 4. Compulsory misses are not plotted in figure 4 because they are the same for all the configurations and close to zero. Direct mapped (DM) and two-way set associative (2W) caches are considered too for comparison purposes. Each of the eight graphs in figure 4, correspond to a unique combination of cache type (instruction or data, as shown in each graph title), benchmark type (integer or floating point, also indicated in the graphs titles as SPECint95 or SPECfp95), and miss type (capacity or conflict, indicated in each graph vertical axis label).

It may be seen from figure 4 that capacity misses (the left graphs) are equal for the SW, DM and 2W curves for each graph, except for the instruction cache under SPECfp95, were the relative performance for SW and 2W caches alternate.

For conflict misses (the right graphs in figure 4) the following observations were done: high degrees of sharing such as 32 or 16 (the first two point in each contiguous curve for SW) result in very low associativities. Figure 4 shows that the instruction caches for both the integer and floating point benchmarks, and the data cache for the integer benchmarks, could not resolve the conflict misses as well as a two-way set associative cache of comparable size. However degrees of sharing of 4 or 2 perform better relative to the two-way cache. This behavior is most pronounced in instruction caches and less in data caches. For the data caches under the floating point benchmarks, small SWSA caches have less conflict misses than the two-way set associative caches, and roughly the same as the cache size increases. An interesting observation is that for instruction and data caches a 1.5 SWSA cache often gives the same conflict miss rate than the next size cache, i.e. a two set associative cache, which is 33 percent greater. This happens for example for all of the 48 Kbytes caches (instruction or data, integer or floating point) and some 24 Kbytes caches. These observations mean that the completion of the bank 2 from a 1.5 to a two-way set associative cache (of greater size), usually does not result in a better solving of conflicts, and so this increment in the cache size only benefits in a reduction in capacity misses. So, when all the capacity misses are removed, or when they remain constant as the cache increases (i.e. nearly all these misses have LRU distances greater than the cache size range considered), 1,5 SWSA caches give the same miss rate as a two-way set associative cache that is 33 percent greater.

D. Comparison of SWSA caches with victim caches.

In this section a comparison between SWSA caches and victim caches [JOU90] is made. When in a SWSA cache the bank 2 is LRU fully-associative and the swap replacement policy is used, a victim cache is obtained. Because of the implementation constrains of a fully-associative bank, victim caches were defined using a small number of entries in the fully-associative buffer. So the comparison between SWSA caches and victim caches is done for SWSA caches with degree of sharing of 32 and 16. For example for a 8Kbytes bank 1, a degree of sharing of 32 gives a bank 2 of 8 entries and a degree of sharing of 16 gives a bank 2 of 16 entries. As the cache doubles its size for the same degree of sharing the number of entries in the bank 2 doubles too.

Mean miss rates for the SPEC95 integer and floating point benchmarks for SWSA caches and victim caches are given in figure 5 (for the victim cache, we account as a miss a reference that was not found in the direct mapped bank or in the fully associative buffer). In figure 5 fully associative banks 2 of large number of entries are plotted too for comparison with SWSA caches with small degrees of sharing. For simplicity the term victim cache is used for all the equivalent degrees of sharing. From figure 5, for the integer benchmarks the victim instruction cache performs slightly better than a SWSA instruction cache for degrees of sharing of 32 and 16. For the floating point benchmarks the SWSA instruction cache with a degree of sharing of 32 performs equal than a victim cache and for a degree of sharing of 16 the SWSA cache performs better. For the data cache under the integer benchmarks victim caches perform slightly better than a $1\frac{1}{32}$ SWSA cache and better for a $1\frac{1}{16}$ SWSA cache. For the data caches under the floating benchmarks victim caches perform much better than SWSA caches, even for small degrees of sharing. The last shows a very high degree of interference between references with very short LRU distances in some of the floating point benchmarks. As the cache size increases the relative difference between SWSA and victim caches is smaller.

An interesting observation from figure 5 is that victim caches usually give the miss ratio of a fully-associative cache of the same total cache size when the ratios of the direct mapped bank 1 and the fully-associative bank 2 is 2. So the miss rate of a LRU fully-associative cache could be obtained

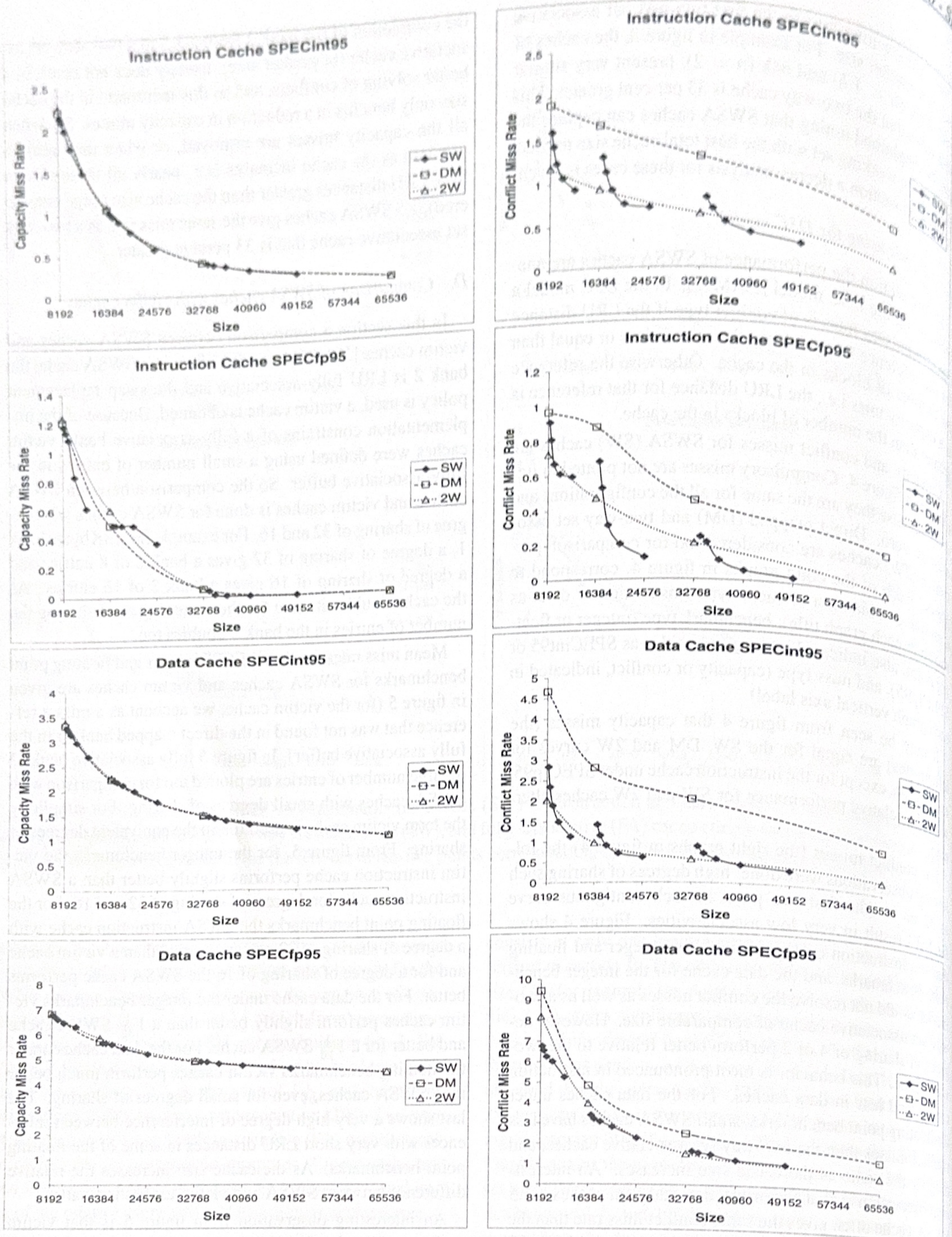

Fig. 4: Mean capacity and conflict miss rates for the SPEC95 benchmarks for Shared Way Set Associative (SW) L1 instruction and data caches. Direct mapped (DM) and two-way set associative (2W) cache curves are plotted for comparison purposes. Each contiguous curve for the SW curve has five points corresponding to associativities $1\frac{1}{32}, 1\frac{1}{16}, 1\frac{1}{8}, 1\frac{1}{4}$ and $1\frac{1}{2}$.

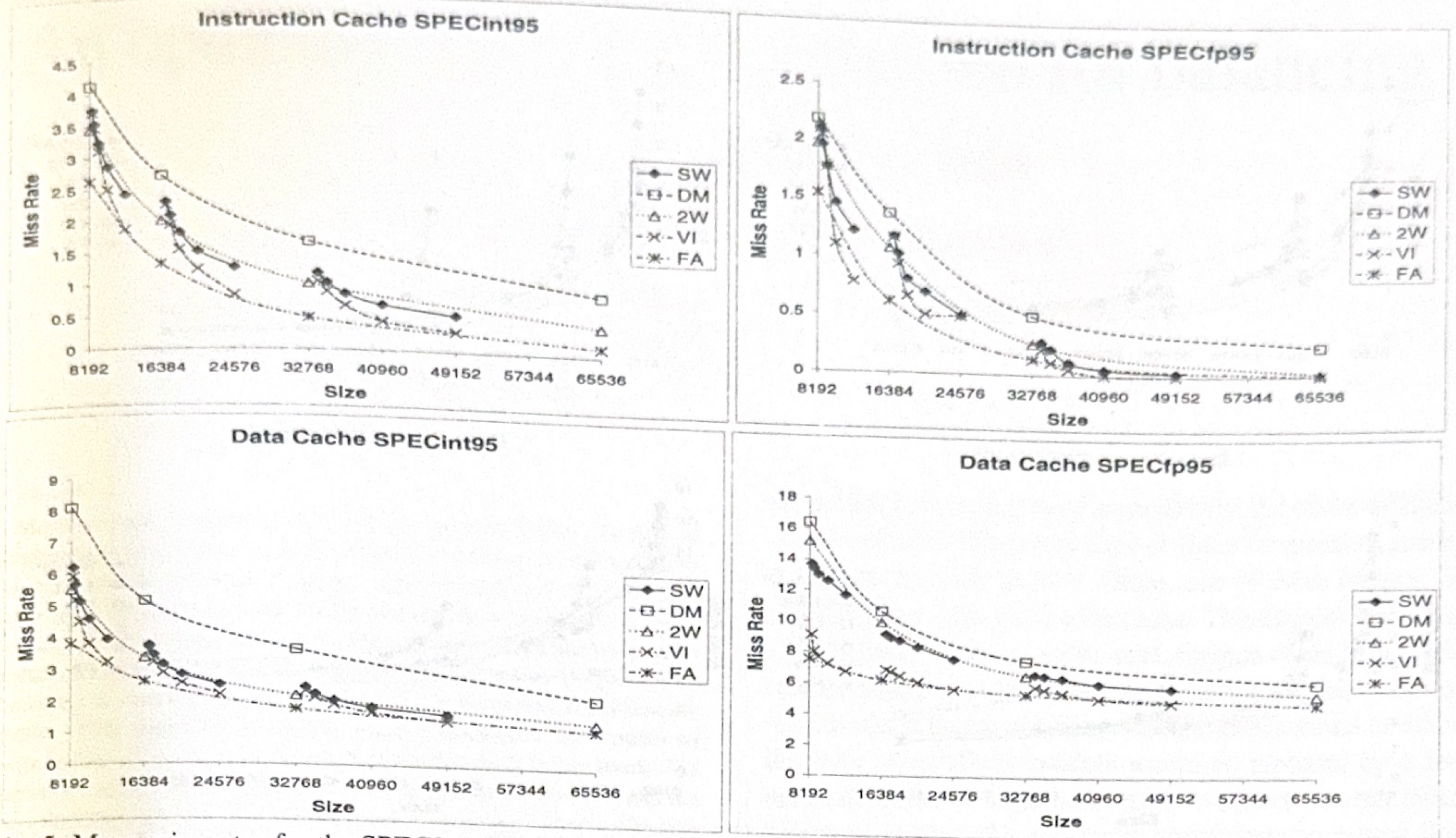

Fig. 5: Mean miss rates for the SPEC95 benchmarks for Shared Way Set Associative (SW) L1 instruction and data caches and the case when the bank 2 is fully-associative (VI). Direct mapped (DM), two-way set associative (2W) and fully-associative (FA) cache curves are plotted for comparison purposes.

with a cache where only 1/3 of the cache is fully associative and the rest is direct mapped.

V. Performance analysis for Shared Way Set Associative Caches using a 1 bit LRU replacement policy.

In this section we consider a 1 bit LRU replacement policy, as explained in section III. This policy has not the property of inclusion for general SWSA caches. For example, the addition of a second bank of size $C2$ to a direct mapped cache, could increase the number of misses for certain memory reference sequences giving an anomalous cache behavior [BEL69]. Results obtained for the 1 bit LRU replacement policy are shown in figure 6 (1 Bit curve). SWSA caches with the swap replacement policy are plotted for comparisons purposes (SWAP curve) as well as direct mapped, two-way set associative, four-way set associative, and fully-associative LRU caches (DM, 2W,4W and FA curves). It may be seen from figure 6, that an anomalous cache behavior usually happens for very high degrees of sharing, such as 32 or 16. In general high degrees of sharing and the use of the 1 bit LRU replacement policy cause that the references that are placed in the second bank become quickly LRU references due to the first hit in the other members of the replacement set. This effect becomes less probable for smaller degrees of sharing, and for associativities of 1.5 this destructive interference in the LRU information is nearly negligible, as it may be observed from figure 6.

It should be noted that as was mentioned in section III, the 1 bit LRU replacement policy coincides with a standard LRU or SWAP replacement policy when the degree of sharing is one (i.e. the cache is two-way set associative). The experimental results have shown that the 1 bit LRU replacement policy gives the same miss rate statistics than the SWAP replacement policy for associativities of 1.5. This observation is consistent as well as for instruction or data caches and for the integer and floating point benchmarks. This makes the 1.5 SWSA caches with 1 bit LRU replacement policy very interesting due to the combination of very good statistics and simplicity of implementation.

VI. Conclusions

A new cache memory organization was analyzed. We call this organization Shared Way Set Associative Cache (SWSA). Results obtained with simulations using the SPEC95 benchmarks, show that SWSA caches with the swap replacement policy usually perform better than LRU two-way set associative caches of equivalent size. A simple 1

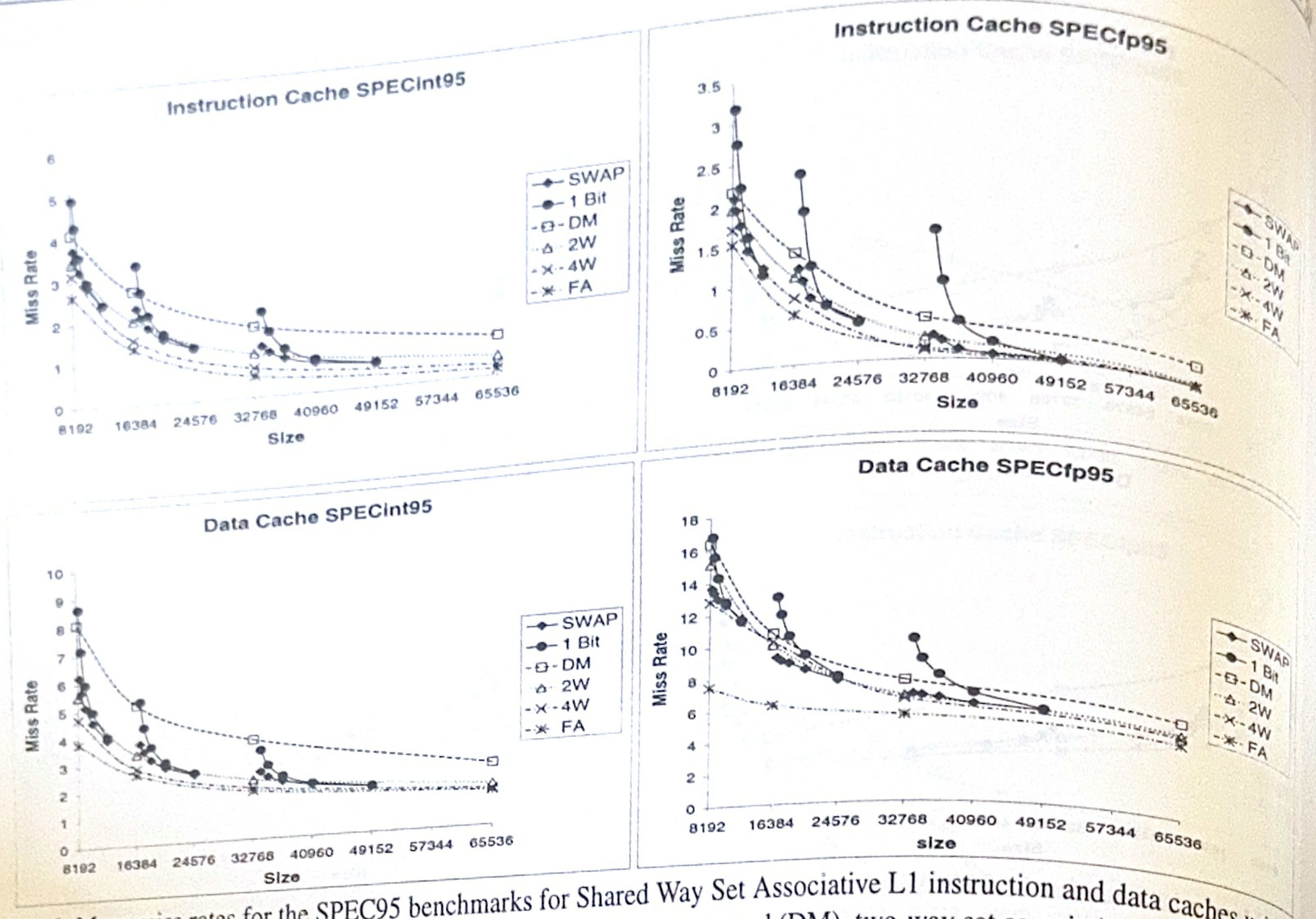

Fig. 6: Mean miss rates for the SPEC95 benchmarks for Shared Way Set Associative L1 instruction and data caches using th SWAP (SWAP) and 1 bit LRU (1 Bit) replacement policies. Direct mapped (DM), two-way set associative (2W), four-way se associative (4W), and fully-associative (FA) cache curves are plotted for comparison purposes.

bit LRU replacement policy gives nearly the same miss rate than the swap replacement policy for 1.5 SWSA caches. This makes the SWSA caches an interesting option of simple implementation. Because SWSA caches are not constrained to powers of two total cache size, SWSA caches may be better adjusted to the needed total cache size. Thus, working sets may be captured with an effective use of cache space. This might be specially applicable for the important case of specific purpose microprocessors, where the cache can be designed for specific applications. Simulations have shown that for large cache sizes, SWSA caches have similar miss rates than two-way set associative caches which are 33 percent larger. This is very important for on chip caches, because of the savings in processor chip area and power consumption.

ACKNOWLEDGMENTS

We gratefully acknowledge Professor Daniel Etiemble from LRI, Université de Paris-Sud, and Professor Alberto Dams from the Universidad de Buenos Aires. We also would like to thank the anonymous referees for their helpful comments. This work was partially financed by the Universidad de Buenos Aires, grant TI-09, and the Consejo Nacional de Investigaciones Científicas y Técnicas , CONICET, grant PIP-4030.

REFERENCES

[BEL66] Belady, L. A.. A Study of Replacement Algorithms for a Virtua Storage Computer. *IBM Systems Journal*, , Vol 5, N2, 1966.

[BEL69] Belady, L. A.; Nelson R. A.; Shidler, G. S.. An Anomaly in Spa Time Characteristics of Certain Programs Running in a Paging Enviro ment. *Communications of the ACM*, p. 349-353, December 1969.

[HAM99] Hamkalo, J. L.; Cernuschi-Frías, B.. A Taxonomy for Cach Memory Misses.. In: Proceedings of the 11th Symposium on Compute Architecture and High Performance Computing. Natal, Brazil, 1999. p 67-73.

[HIL88] Hill, M. D.. A Case for Direct-Mapped Caches.. *IEEE Computer*, Vol 21, n.12, p. 25-40, 1998.

[HIL89] Hill, M. D.; Smith, A. J.. Evaluating Associativity in CPU Caches *IEEE Trans. on Computer*, C-38, n.12, p. 1612-1630, 1995.

[JOU90] Jouppi, P. N.. Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers. In: Proceedings of the ACM INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE, 1990. p. 364-373.

[PAT95] Patterson, D. A.; Hennessy, J. L. **Computer Architecture a Quantitative Approach**, San Mateo, California: Morgan Kaufmann Publishers, 1995.

[SMI82] Smith, A. J. Cache Memories. *ACM Computing Surveys*, p. 473-530. Sept. 1982.

[SRI94] Srivastava, A.; Eustace, A. ATOM: A System for Building Customized Program Analysis Tools. In: Proceedings of the ACM CONFERENCE ON PROGRAMMING LANGUAGE DESIGN AND IMPLEMENTATION, 1994. p. 196-205.