1992-12

An investigation of memory latency reduction using an address prediction buffer

Billingsley, Arthur Brooks, Jr.

Monterey, California. Naval Postgraduate School

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION **UNCLASSIFIED**	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School / 6b. OFFICE SYMBOL (if applicable) ECE	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION / 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
An Investigation of Memory Latency Reduction Using an Address Prediction Buffer (U)

12. PERSONAL AUTHOR(S)
Arthur Brooks Billingsley, Jr.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM 05/92 TO 12/92	December 1992	39

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Memory latency, Computer Architecture, Cache Memory, Computer Performance, Latency Reduction, Cache Heirarchy

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Developing memory systems to support high-speed processors is a major challenge to computer architects. Cache memories can improve system performance but the latency of main memory remains a major penalty for a cache-miss. A novel approach to improve system performance is the use of a *memory prediction buffer*. The *memory prediction buffer* (MPB) is inserted between the cache and main memory. The MPB predicts the next cache-miss address and pre-fetches the data. The use of an MPB in a computer system is shown to decrease main-memory latency and increase system performance.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION **UNCLASSIFIED**
22a. NAME OF RESPONSIBLE INDIVIDUAL Douglas Fouts	22b. TELEPHONE (Include Area Code) (408) 646-2852 / 22c. OFFICE SYMBOL ECE/FS

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

Approved for public release; distribution is unlimited

AN INVESTIGATION OF MEMORY LATENCY REDUCTION USING AN ADDRESS PREDICTION BUFFER

by

Arthur Brooks Billingsley Jr.
Lieutenant, United States Navy
B.S.E.E, Auburn University, 1985

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1992

ABSTRACT

Developing memory systems to support high-speed processors is a major challenge to computer architects. Cache memories can improve system performance but the latency of main memory remains a major penalty for a cache-miss. A novel approach to improve system performance is the use of a *memory prediction buffer*. The *memory prediction buffer*(MPB) is inserted between the cache and main memory. The MPB predicts the next cache-miss address and pre-fetches the data. The use of an MPB in a computer system is shown to decrease main-memory latency and increase system performance.

TABLE OF CONTENTS

TABLE OF SYMBOLS

E	efficiency
I	instructions
CPI_{EFF}	effective cycles per instruction
S	speedup
$CPI_{EFF(MPB)}$	effective cycles per instruction with memory prediction buffer
T_{EA}	effective access time
T_{CS}	cache search time
C_{HR}	cache hit ratio
T_{CF}	cache fetch time
T_{MR}	memory read time
T_{MW}	memory write time
f	clock frequency in Hertz
HR_L	local cache hit rate
HR_C	first level cache hit rate
HR_{SYS}	overall system hierarchy hit rate
HR_{MPB}	MPB local hit rate
MPB	memory prediction buffer

ACKNOWLEDGEMENTS

The undertaking of any research project of this nature is not performed in isolation. It is therefore desirable to recognize the contributions of others who aided the completion of this research. Anita Borg of Digital (DEC) was extremely helpful in obtaining address traces for the simulation of design concepts. Mark Hill of the University of Wisconsin provided his cache simulators, DINEROIII and TYCHO, for the simulation of the design concept. Thanks to Richard Hamming of the Naval Postgraduate School for his guidance in the efficient progression of research and for invaluable statistical insight. A special thanks to John Powers of the Naval Postgraduate School for his professional and personal assistance to the author. The simulation for this research was accomplished using a network of Sun SPARCs and the efforts of three fine system administrators, Robert Limes, Elaine Kodres and Brad Polk. In addition, Douglas Fouts provided the initial inspiration for the development of the concept and provided support, financial and professional, to the author. This research was funded by a Naval Postgraduate School Research Initiation Grant.

I. INTRODUCTION

The technological advances in high-speed, general purpose processors have outpaced the support provided by main memory systems. In addition, software applications continue to grow in processor and memory requirements. The major factors in the design of memory systems are size of address space, bandwidth required, main-memory latency, and memory subsystem cost. Large memory subsystems use dynamic random-access memories because of their low cost per bit. Caching schemes, which employ high-cost, high-speed memories, are used to overcome main-memory latency and increase bandwidth. However, main memory latency, which is the time (in processor cycles) between the start of a memory fetch and the start of the transfer of requested data, is significant and increasing [PRZYBY90]. Further gains in memory system performance are possible through the use of different manufacturing processes (CMOS, BiCMOS, ECL and GaAs) [VAGTS92] and stringent design of the memory hierarchy. One such memory performance enhancement is the prediction of a cache-miss read address request to main memory. If the read address is predicted and the data made available, then the overall system performance is improved.

Since current RISC processors far exceed the capability of main memory systems, the focus for the computer systems architect is how to improve the performance of the memory hierarchy. Large, fully-associative caches are cost prohibitive, and direct-mapped caches offer an excellent alternative [HILL88]. Direct-mapped caches have a higher miss rate than fully-associative or set-associative caches. A disadvantage of cache memories, in general, is the miss penalty[PATHEN90],[PRZYBYZ90]. The reduction of the miss rate and subsequent miss penalty is the motivation for the memory prediction buffer (MPB).

Conceptually, the MPB is an enhancement for the data cache. The behavior of processors utilizing separate data and instruction caches is noted in this research and others[JOUPPI90],[PRZYBY90]. Examination of this behavior shows that instruction caches and data caches behave differently. Instruction caches can improve effectiveness by simply prefetching the next instruction. This approach is shown to be less effective for data caches [PATHEN90],[JOUPPI90]. If this approach is used for data cache management, it contributes to pollution of the cache and increases the number of capacity misses. Since most modern RISC

processors have separate instruction and data caches, and employ some prefetch mechanism for the instruction cache, this research will focus on improving the effectiveness of the data cache by inserting an MPB between the cache and its refill line (main memory, in most cases). Although this organization is the focus for this research, it is not the only implementation possible for the MPB[NOWICKI92].

II. MEMORY HIERARCHY AND LATENCY REDUCTION

The von Neumann architecture, used by most single-instruction-single-data[1] (SISD) and single-instruction-multiple-data (SIMD) machines, has some baseline behavioral characteristics to consider [HWANG84]. The characteristics of the memory subsystem provide the parameters for optimization of the operational behavior of the memory subsystem in conjunction with the processor and secondary storage. First, stored programs obey the principle of *locality* [PATHEN90]. This principle has two components which state that programs, while executing, favor only a portion of their address space at a given instant. The two components are:

- *Spatial Locality* - Programs tend to request data and instructions that have memory addresses near the instructions and data currently being used. The von Neumann architecture provides for the execution of sequential program instructions and programs use related data items which are likely to be adjacently stored.
- *Temporal Locality* - Programs tend to use current information and data. That is, if an item is referenced, it will probably be referenced again soon. The older the information, the less likely it is that the program will again reference it. Temporal locality is especially evident in the execution of program loops where instruction and data are used several times within a short period of time.

With reference to these principles, high-speed buffers are inserted between the main memory and the processor. These buffers are known as *caches*. The caches store portions of main memory which are currently in use by the executing program. This allows rapid access by the processor of the instructions and data needed to continue processing. Although the cache does a great job of hiding main memory latency, a disadvantage of its use is the penalty for a cache miss. The construction of the cache gives the following behavioral characteristics for a cache miss.

- *Compulsory* - cache misses that occur when a block is first accessed and the program is just starting. These are sometimes called *cold start misses* since the cache has never held the information requested.
- *Capacity* - cache misses that occur when discarded blocks are again referenced by the executing program. These misses are inevitable since the cache size is less than main memory size.
- *Conflict* - the block placement strategy dictates conflict misses. Conflict misses occur when a block is discarded because too many incoming blocks map to the same set and the

---

1. Flynn's classification (1966) is based on the multiplicity of instruction streams and data streams in a computer system [HWANG84].

discarded block is soon needed. This characteristic is evident in both set-associative mapped and direct-mapped caches.

The structure of the memory subsystem is given in Figure 1. Traversing down the hierarchy, access time increases and the storage size increases. However, bandwidth decreases significantly while traversing the hierarchy, top to bottom. Some nominal figures for size and bandwidth are also given in Figure 1. It is worthy to note that each level is a subset of the next lower level. That is, each level contains only a subset of the information contained in the next lower level. This presents a constraint of maintaining coherency (correct information) throughout the hierarchy. The MPB receives its information from the next lower level of the hierarchy. In this research, the next level of the hierarchy is the main memory. For the development of the concept of the MPB and for most of the simulation described here, the MPB is not involved in the write policy of the cache. The MPB always gets its data from the main memory which is kept up to date. Further research of the MPB will study the implementation of a write-through policy for coherency. Write-back performance will also be examined in follow-on research.

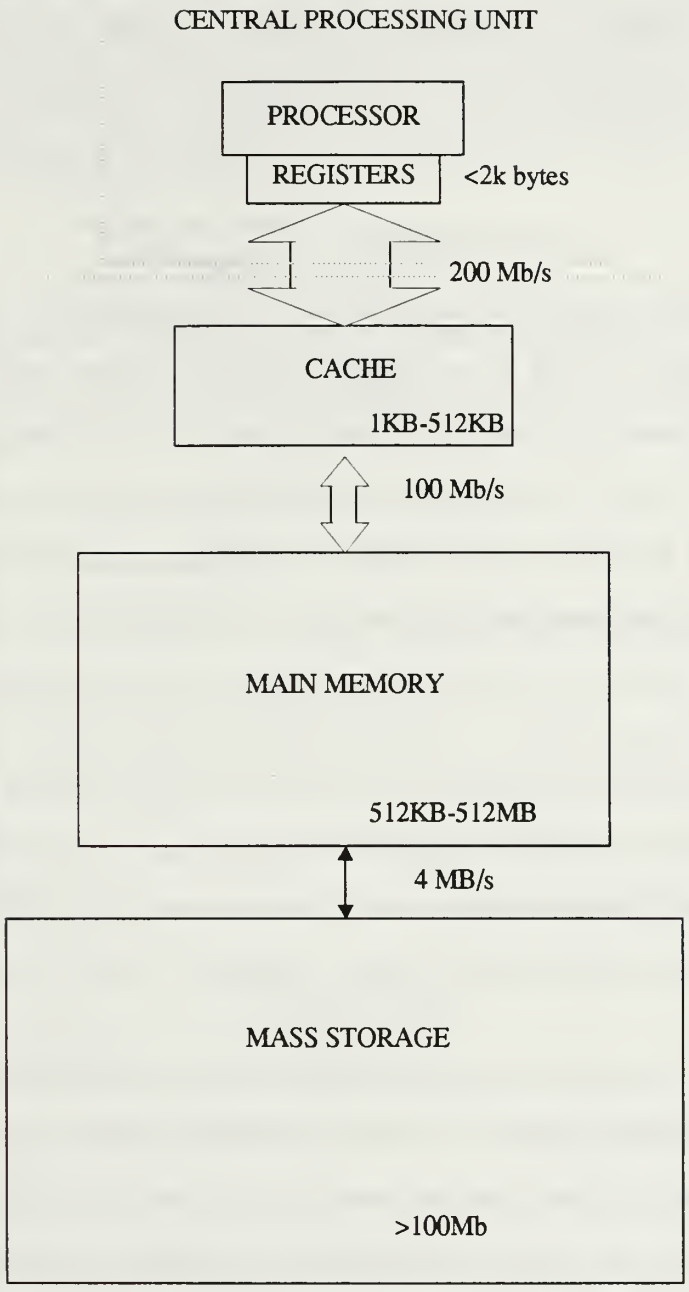


Figure 1: Memory Hierarchy

III. PERFORMANCE METRICS

In order to investigate the performance of the memory subsystem, characteristics of the memory subsystem must be developed. From the system perspective, work completed in time defines system performance. Hence, system performance can be described analytically as Equation 1.

$$\text{System Performance} = \frac{\text{Instructions Completed}}{\text{Elapsed Time}} \tag{1}$$

This definition of system performance does derive the ubiquitous MIPS units. This unit of measurement should not be used in comparison of different systems performing the same task [PATHEN90]. However, for characterization of a specific system performing the same task, this unit of measure is useful. This measure of performance can be focused in terms of processor cycles. Efficiency is a product of the number of instructions executed, the number of clock cycles per instruction and the clock speed (Equation 2).

$$E = I \cdot CPI \cdot f \tag{2}$$

Expanding this model, the number of cycles per instruction executed is the metric that is directed influenced by the memory subsystem. Statistically, a more stable metric is the effective CPI. The effective CPI is the statistical average of several measurements. The effective CPI is

$$CPI_{EFF} = \sum_i \frac{CPI_i}{i} \tag{3}$$

The number of cycles per instructions is largely determined by processor architecture and register/cache structure(effectiveness). With a focus toward the memory structure, the effective access time of the memory subsystem is the best metric to indicate memory subsystem performance. This parameter depends on the cache access time and the main memory access time. By decreasing the number of cycles per instruction, the system performance is improved. The speedup in system performance is modelled by Equation 4.

$$S = \frac{CPI_{EFF} - CPI_{EFF(MPB)}}{CPI_{EFF}} = 1 - \frac{CPI_{EFF(MPB)}}{CPI_{EFF}} \tag{4}$$

The nominal figures for the number of cycles per instruction in high performance processors is 1.2-2.0 CPI. If we assume that the processor can execute instructions at the bandwidth of the memory subsystem, the speedup becomes a function of the effective access time of the memory subsystem. Equation 5 determines the speedup of a given system by reference to the effective access time with the MPB and without the MPB.

$$S = 1 - \frac{T_{EA\,(MPB)}}{T_{EA}} \tag{5}$$

The effective access time measures the memory hierarchy performance. The effective access time is therefore, a function of the cache performance and main memory performance as noted in Equation 6.

$$T_{EA} = T_{CS} + C_{HR} \cdot T_{CF} + (1 - C_{HR}) \cdot (T_{CS} + T_{MR} + T_{CF}) \tag{6}$$

This relationship can be simplified by noting the time for a cache tag search T_{CS} is very small. In addition, the cache tag search and cache fetch are much smaller than the time to read/fetch data from main memory, T_{MR}. The effective access time can then be approximated as in Equation 7.

$$T_{EA} \approx C_{HR} \cdot T_{CF} + (1 - C_{HR}) \cdot (T_{MR}) \tag{7}$$

This approximation can be used only for comparison between simulation models. The description given by Equation 6 must be used for evaluation of the simulation model with respect to implementation performance.

IV. MEMORY PREDICTION BUFFER

The memory prediction buffer(MPB) was conceived to predict the next cache-miss address and prefetch the data before the request is made by the processor. The MPB can be inserted between the cache and its refill line as depicted in Figure 2. Another possible configuration could be the use

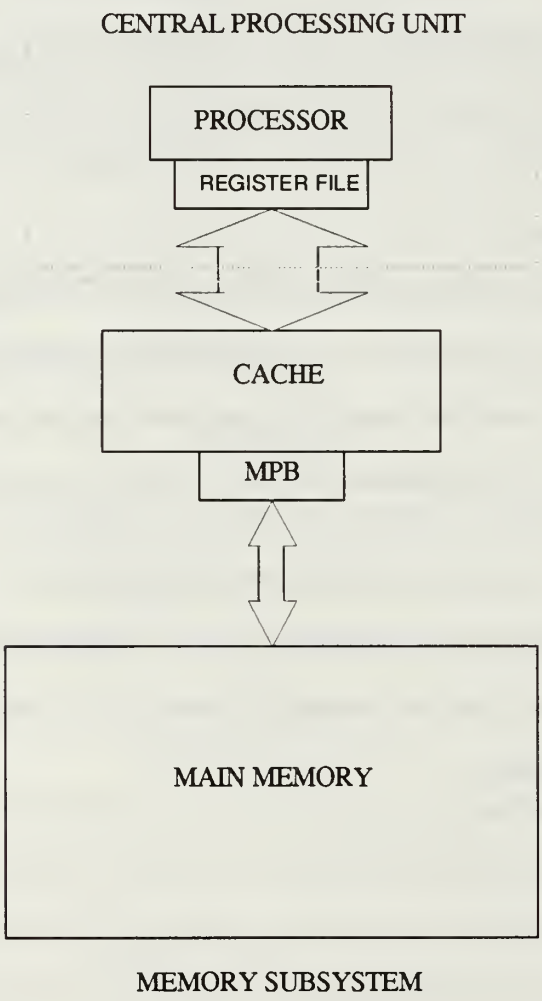


Figure 2: MPB With Cache Implementation

of smaller MPBs attached to individual memory chips (DRAMs). This implementation is realized in recent work by Nowicki[NOWICK92]. A block diagram of this approach is given in Figure 3. In

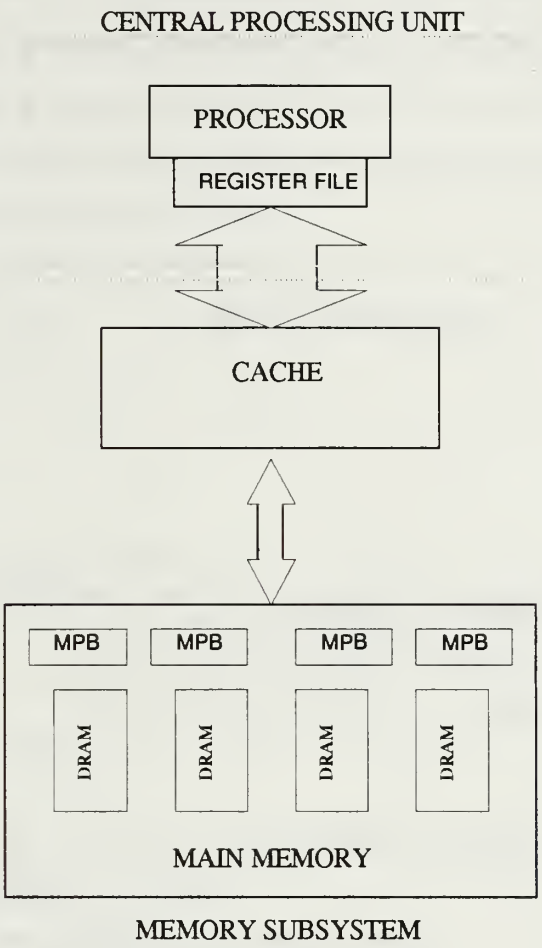


Figure 3: MPB With Main Memory Implementation

the early research of this idea, efforts turned instinctively toward statistical methods for prediction. The area of digital signal processing was explored for possible solutions to the prediction requirement[HAMMIN83],[THERRI92]. Kalman filters, Wiener filters and other adaptive techniques for prediction were proposed and investigated. However, further characterization of the problem provided more specifications for possible solutions.

Cache simulation was achieved using Mark Hill's DINEROIII cache simulator. The model cache is a direct-mapped, 8K data, 8K instruction with a 32 byte line size. Using various ATUM traces[GRIMSR92] and DEC traces[BORG90], cache miss addresses were investigated[AGARWL86]. Review of the traces show that spatial locality and temporal locality are valid for all processes. Since no curves are noted in the traces, prediction should employ linear methods. The physical construction of the memory prediction buffer is given in Figure 4. The

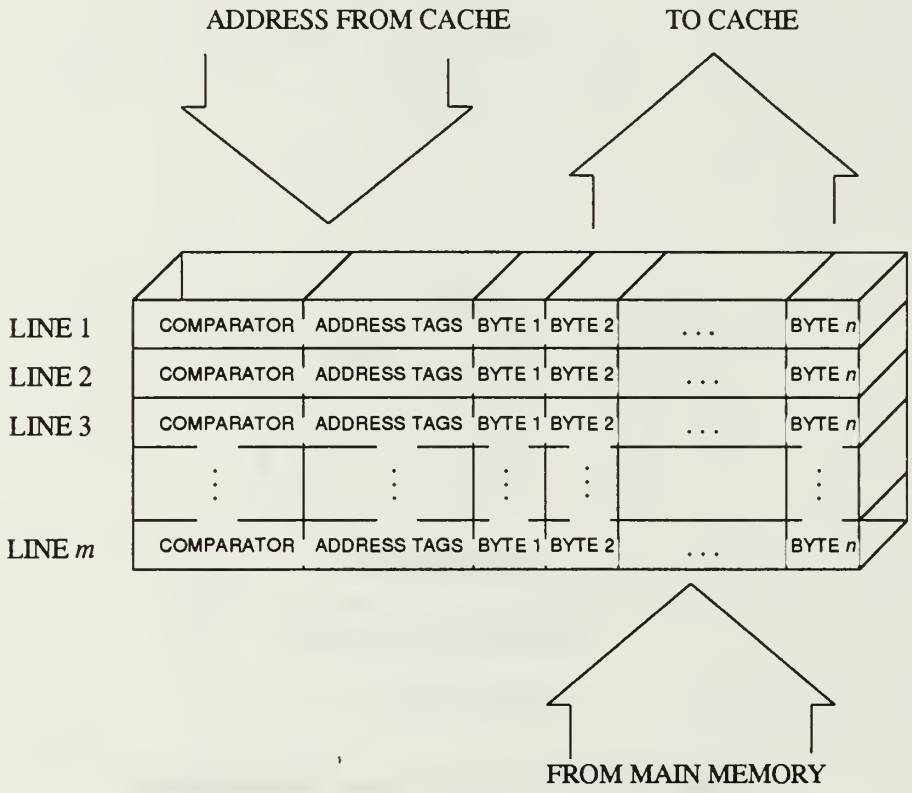


Figure 4: Memory Prediction Buffer

simulation was configured to give the number of cache hits before a miss is encountered. The average of these miss events give the constraint of time available to predict and prefetch a miss address. Since the average of cache-hits before a cache-miss is 4-6, it is possible that some 6-10 cycles are available for prediction and prefetch. In addition, the system bus bandwidth must be considered for prefetch solution. These constraints were responsible for the development of a

simpler prediction algorithm. The prediction algorithm yields a bias for the ensuing prefetch. The algorithm is implemented in C for simulation.

If the current address is larger than the past address, then the bias is positive (negative otherwise). The algorithm for the MPB is given in Figure 5. The determination and application of

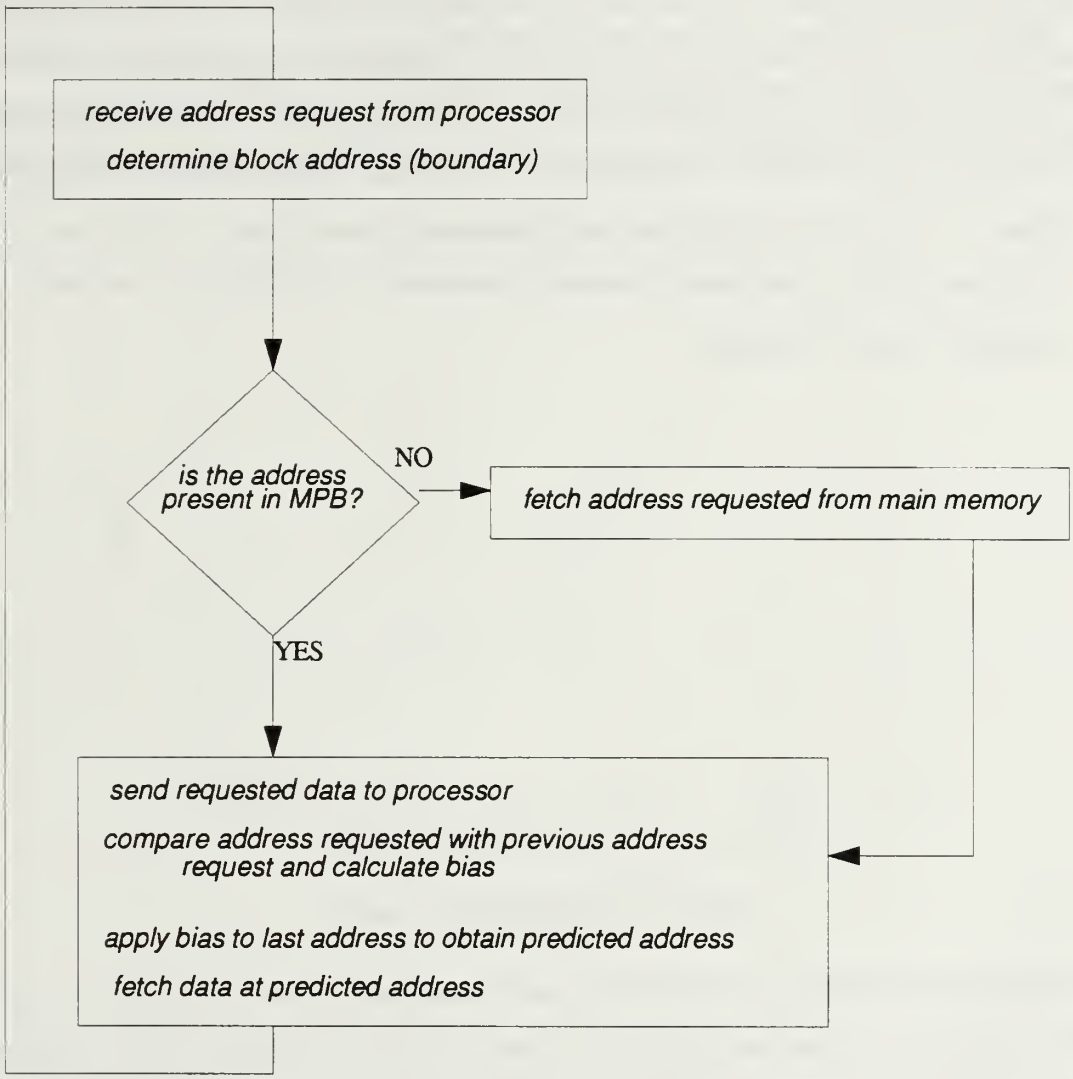


Figure 5: Memory Prediction Buffer Algorithm

the bias is central to the algorithm. The bias is simply the difference in address boundaries (if word aligned) of the previous address and the current address. If the address requested is greater than 32K away, another address stream bias is established. The corresponding address stream bias is used to predict the next requested address. The bias may be positive or negative, that is, ascending or

descending in memory. The correct address stream bias is determined using a simple but fast binary search. The search time can be reduced further using a fully associative algorithm.

The structure of the memory prediction buffer is similar to a conventional fully-associative cache. The MPB is composed of *m* lines of *n* byte blocks. For the cache used in this research, the MPB has 16-256 lines of 32 byte blocks. The blocks are aligned on the same address(word) boundaries as the first level cache. The block size is dependent on the block size of the first level cache. The optimal size of the MPB is 64-256 lines. This size is due to the fan-out requirements (and costs) for the construction of a fully associative cache and the number of lines (sets) needed to allow effective use of the replacement policy used (random replacement vice LRU, FIFO, etc.). If a LRU replacement policy is used instead of random replacement, a smaller MPB can be used to give the same performance improvement.

V. MEMORY PREDICTION BUFFER PERFORMANCE

A. MPB THEORECTICAL PERFORMANCE

The memory prediction buffer determines the future cache miss address using previous cache miss addresses. For this analysis, only the data cache is given a MPB. The instruction cache is set to prefetch instructions. Given a model cache with a hit ratio of 93.2%, if the MBP is found to be correct on 33% of its predictions, an increase of 2.1% is realized for the cache hit rate. The effective cache hit ratio is improved to 93.2% from 95.3%. The graph of Figure6 gives the effective cache

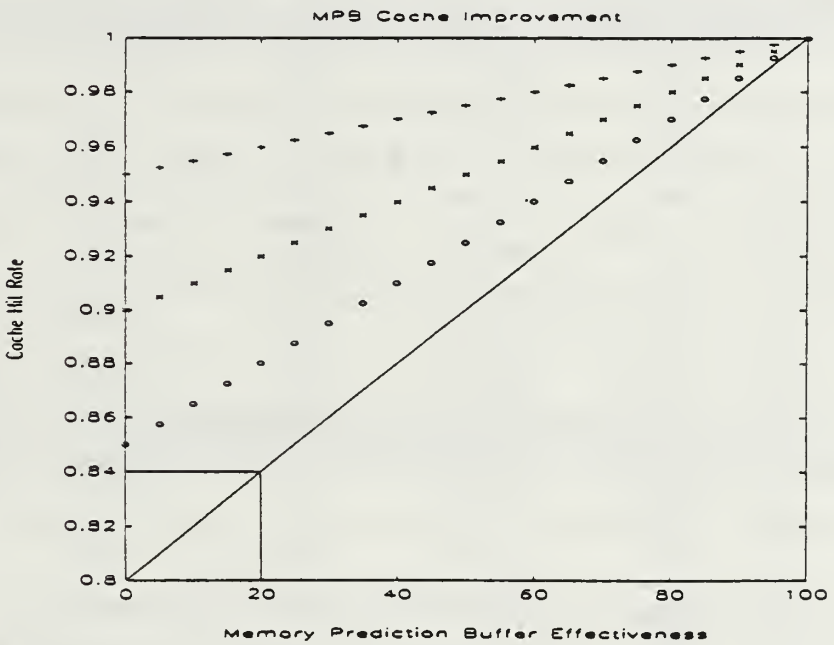


Figure 6: MPB Performance Graph

hit rate as a function of MFP effectiveness. There are four cache models that are compared. One model has an 80% initial hit rate, another model has an 85% hit rate and so on. A sample reading is shown for a base cache hit ratio of 80% with an MPB effectiveness rating of 20%. The resulting effective cache hit ratio for this sample is 84%. This is an increase of 4% in the effective cache hit ratio. The resulting system performance achieves a speedup of 9%.

The model system for this investigation has 10ns cache memory and 80ns main memory. This model memory hierarchy is used by the simulation study also. The cycle time of the main memory is not considered but would add to the effectiveness of the MPB.

B. BASELINE SYSTEM PERFORMANCE

In order to compare the performance of the MPB to existing latency reduction strategies, several measurements of the baseline system had to be collected and examined. This baseline system was constructed using the cache simulator, DINEROIII. The system simulates separate 8K direct-mapped data and 8K direct-mapped instruction caches.

Table 1: BASELINE SYSTEM PERFORMANCE

Process	Cache Size	HR_L	HR_C	HR_{SYS}	Speedup
8K FIRST LEVEL CACHE BASE-SYSTEM PERFORMANCE					
SPICE	8192	96.51	96.51	96.51	- 0 -
Pascal	8192	91.57	91.57	91.57	- 0 -
LISP	8192	92.44	92.44	92.44	- 0 -
FORTRAN	8192	93.88	93.88	93.88	- 0 -
Tree	8192	98.66	98.66	98.66	- 0 -
SOR	8192	90.50	90.50	90.50	- 0 -
12K FIRST LEVEL CACHE PERFORMANCE					
SPICE	12288	97.16	97.16	97.16	3.66
Pascal	12288	94.40	94.40	94.40	12.46
LISP	12288	96.32	96.32	96.32	17.76
FORTRAN	12288	95.11	95.11	95.11	6.03
Tree	12288	97.43	97.43	97.43	(-7.87)
SOR	12288	91.16	91.16	91.16	2.77
8K FIRST LEVEL CACHE (DM) WITH 4K SECOND LEVEL CACHE (FA)					
SPICE	4096	24.46	96.51	97.37	4.84
Pascal	4096	36.91	91.57	94.68	13.69
LISP	4096	75.59	92.44	98.16	26.18
FORTRAN	4096	32.58	93.88	95.81	9.46
Tree	4096	68.32	98.56	99.44	4.99

Table 1: BASELINE SYSTEM PERFORMANCE

Process	Cache Size	HR_L	HR_C	HR_{SYS}	Speedup
SOR	4096	23.84	90.50	92.77	9.54

C. MPB SIMULATION PERFORMANCE

The theoretical study of the MPB was realized when implemented using trace-driven simulation (TDS)[GRIMSR92] with the DINEROIII cache simulator (provided by Mark Hill). As with any TDS research, address traces and their accuracy are critical to proper simulation. For this research, ATUM traces[AGARWL86] and DEC Titan[BORG90] traces were used. Some behavioral characteristics of the simulation are graphically illustrated in the appendix. Table 2 gives

Table 2: MEMORY PREDICTION BUFFER PERFORMANCE(DEC)

Process	MPB Lines	Blocks per line	HR_{MPB}	HR_C	HR_{SYS}	Speedup
TREE 1	128	32	69.89	97.87	99.37	9.14
TREE 2	128	32	59.57	98.01	99.20	7.31
SOR 1	128	32	12.77	90.51	91.79	5.38
SOR 2	128	32	10.20	90.29	91.35	4.42

a summary of MPB performance for two processes and two runs of each. SOR is Renato Deleones' successive over-relaxation algorithm that uses sparse matrices. TREE is Joel Bartletts' program which builds a tree data structure and searches for the largest element in the tree. His program is a variant of LISP. Both of these process traces were provided by DEC WRL. The model system is a RISC processor with separate 8K instruction and 8K data caches. There are 32-byte blocks in the cache and in the MPB. The cache is direct-mapped for reasons given by [HILL88]. The initial cache hit rate CHR was before the insertion of the MPB. The local hit rate for the MPB is given under MHR. The overall hit rate for the cache and MPB combined is listed under NHR. The speedup is listed for the overall system. For these examples, each line of the MPB consists of 32-byte lines(blocks) and 128 lines. Each line is boundary aligned in the same way as the cache. That is, just as the cache may use word aligned blocks, so does the MPB. This MPB simulation used a random

replacement policy for the removal of lines. Toward the end of this research effort, a MPB was simulated using a least-recently used (LRU) replacement policy. Several simulations using this replacement policy showed that the number of lines in the MPB could be reduced while maintaining the effectiveness of the MPB. In particular, 64 lines were shown to perform nearly as well as 128 lines. For the simulation results of Table 2, the speedup numbers are modest but, the cost of this implementation is minimal when compared to a 256K next level cache[PATHEN90].

In addition to the simulations using the DEC traces, simulations were also done using ATUM traces. Table 3 list results of simulation using ATUM traces. The model system is the same as used

Table 3: MEMORY PREDICTION BUFFER PERFORMANCE (ATUM)

Process	MPB Lines	Blocks per line	HR_{MPB}	HR_C	HR_{SYS}	Speedup
Spice	128	32	33.50	93.22	95.27	6.75
Pascal	128	32	47.35	95.62	97.45	9.80
LISP	128	32	69.75	92.68	97.72	23.33
FORTRAN	128	32	40.11	94.22	96.90	13.36

in the DEC trace simulation. These simulation results can be used to motivate further research. ATUM traces are relatively short for cache modelling and behavior analysis. Each trace is approximately 400,000 addresses. This number of addresses is marginally adequate for a 32K cache simulation and larger cache-size simulation would require a larger number of addresses for proper and accurate simulation.

For the preceding research, a random-replacement policy was used by the MPB. An early implementation of the MPB using a least-recently-used (LRU) policy shows improved performance over the random-replacement algorithm. . Table 4 lists the results of this research using the process

Table 4: MEMORY PREDICTION BUFFER PERFORMANCE (LRU)

Process	MPB Lines	Blocks per line	HR_{MPB}	HR_C	HR_{SYS}	Speedup
TREE	128	32	79.11	97.91	99.98	12.64

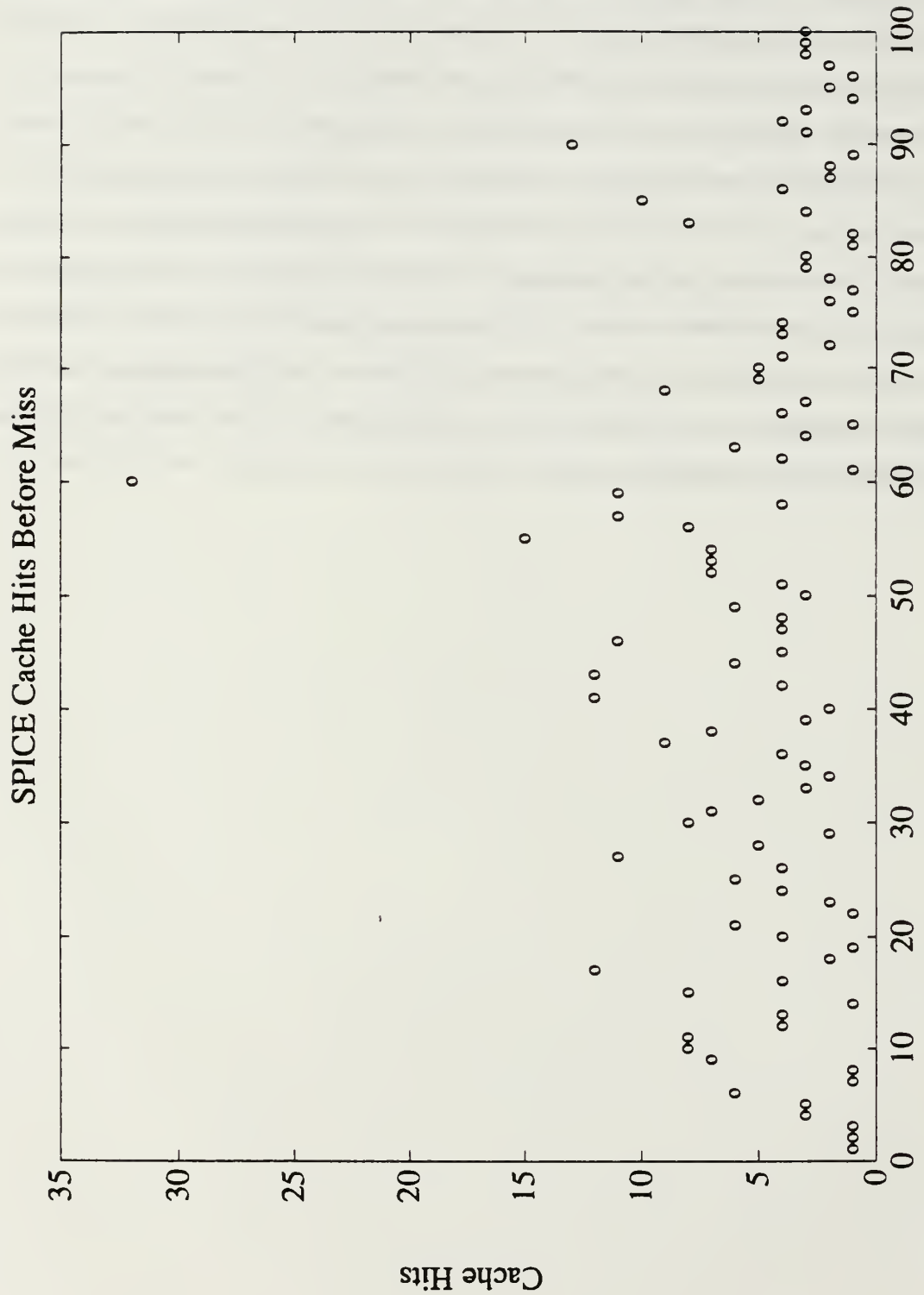
"tree". Results of this implementation using other processes were not yet accomplished at the time of the report. As evidenced by all these simulation studies, the MPB is shown to be a favorable architectural concept for consideration in systems where the highest possible performance is desired and systems costs are constrained.

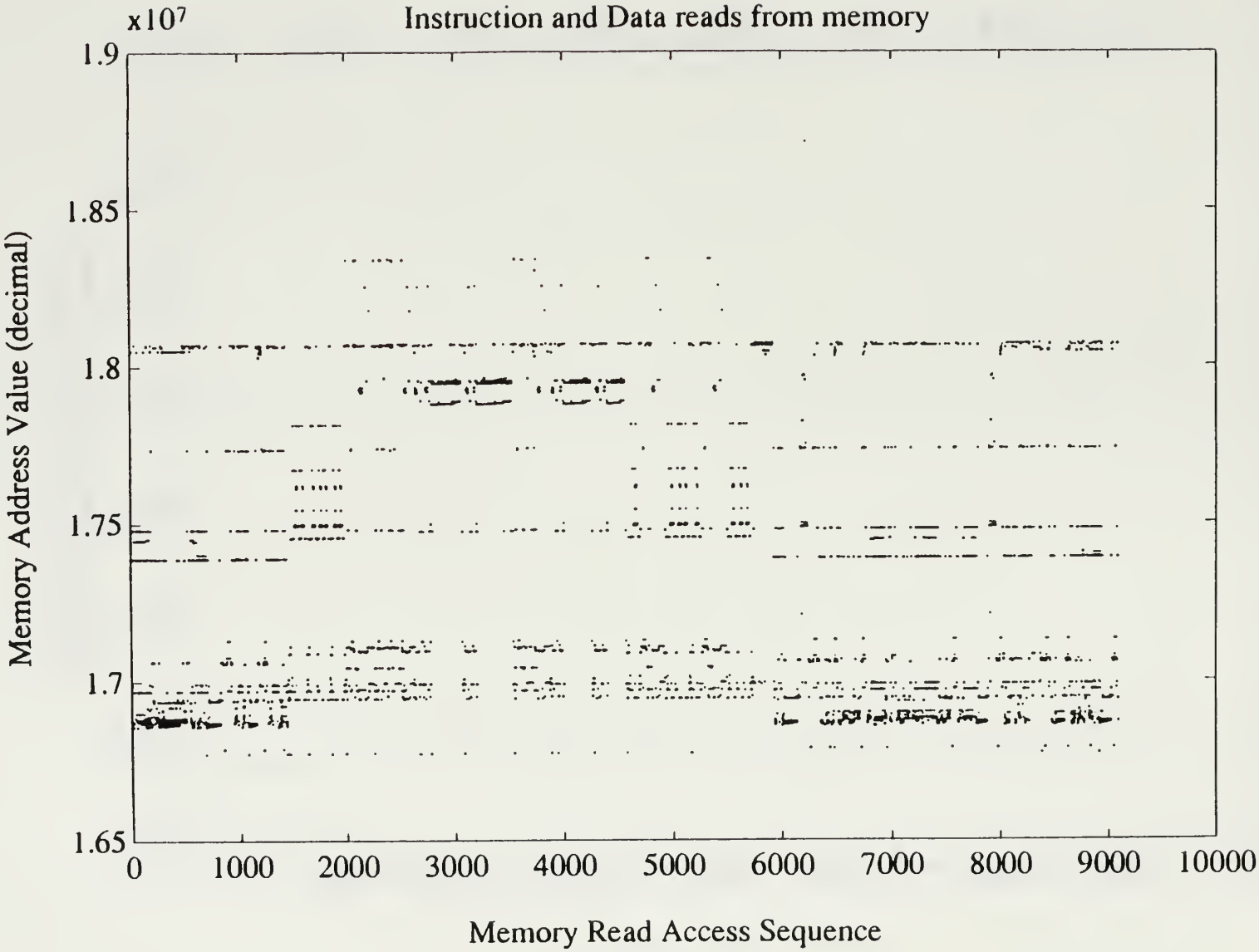
VI. CONCLUSIONS

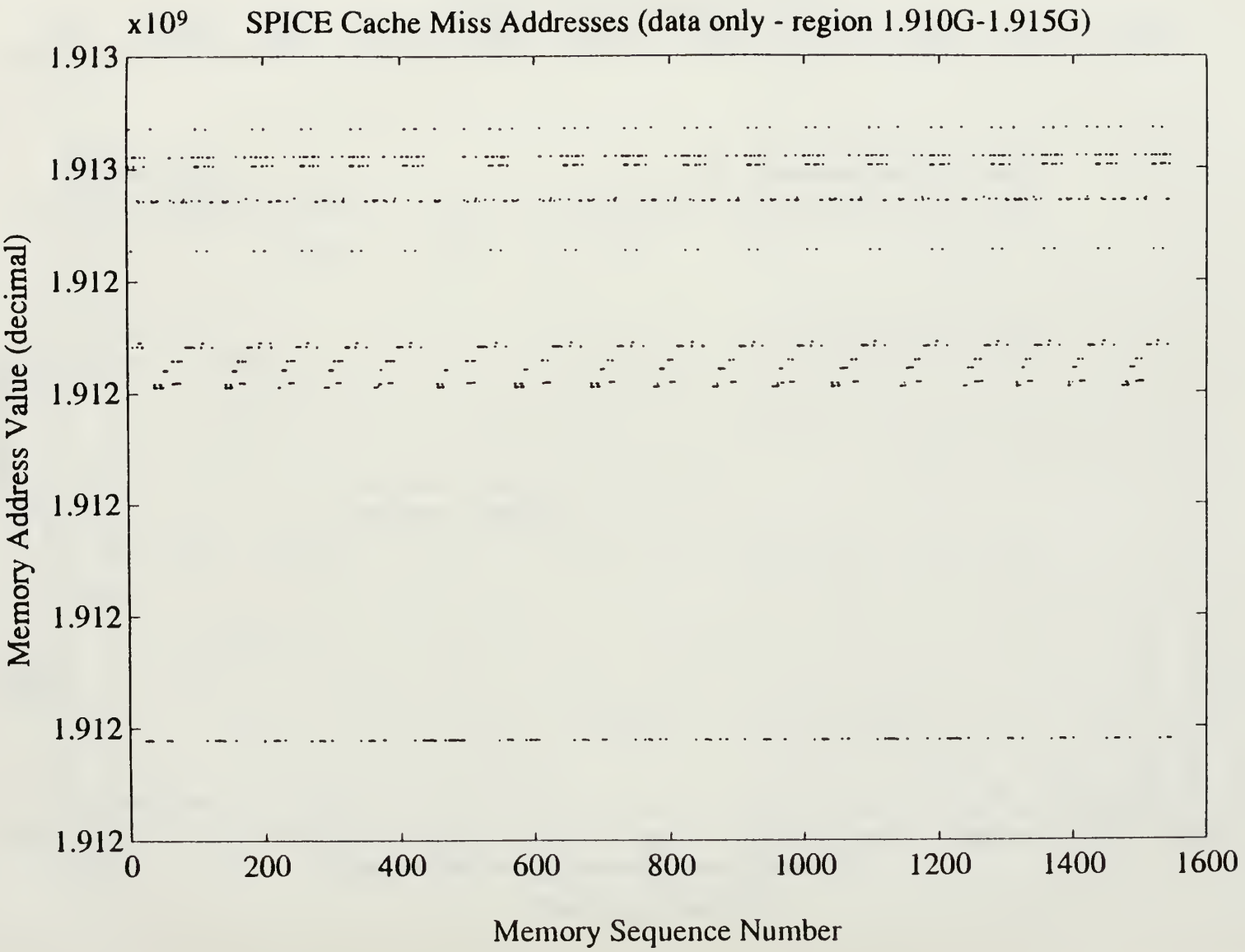
The memory prediction buffer is proposed as a component for high performance computer systems. The widening gap between processor speed and memory subsystems require the investigation of alternative architectures for reducing main memory latency while restraining costs. The MPB outperforms prefetch always strategies by allowing addressing in the up and down direction. In addition, the MPB does not contribute to pollution of the cache. Effective memory latency reduction must be addressed at the time of system design. In addition, as the requirements for a larger address space grows, memory heirarchy design and implementation will continue to increase in complexity. The implementation of a MPB is less expensive than a next-level cache and delivers a comparable performance enhancement. In addition, the algorithm used can be tailored to the proposed system environment to provide a more effective latency reduction structure. The MPB is shown to improve overall system performance and provide reasonable gains in speedup.

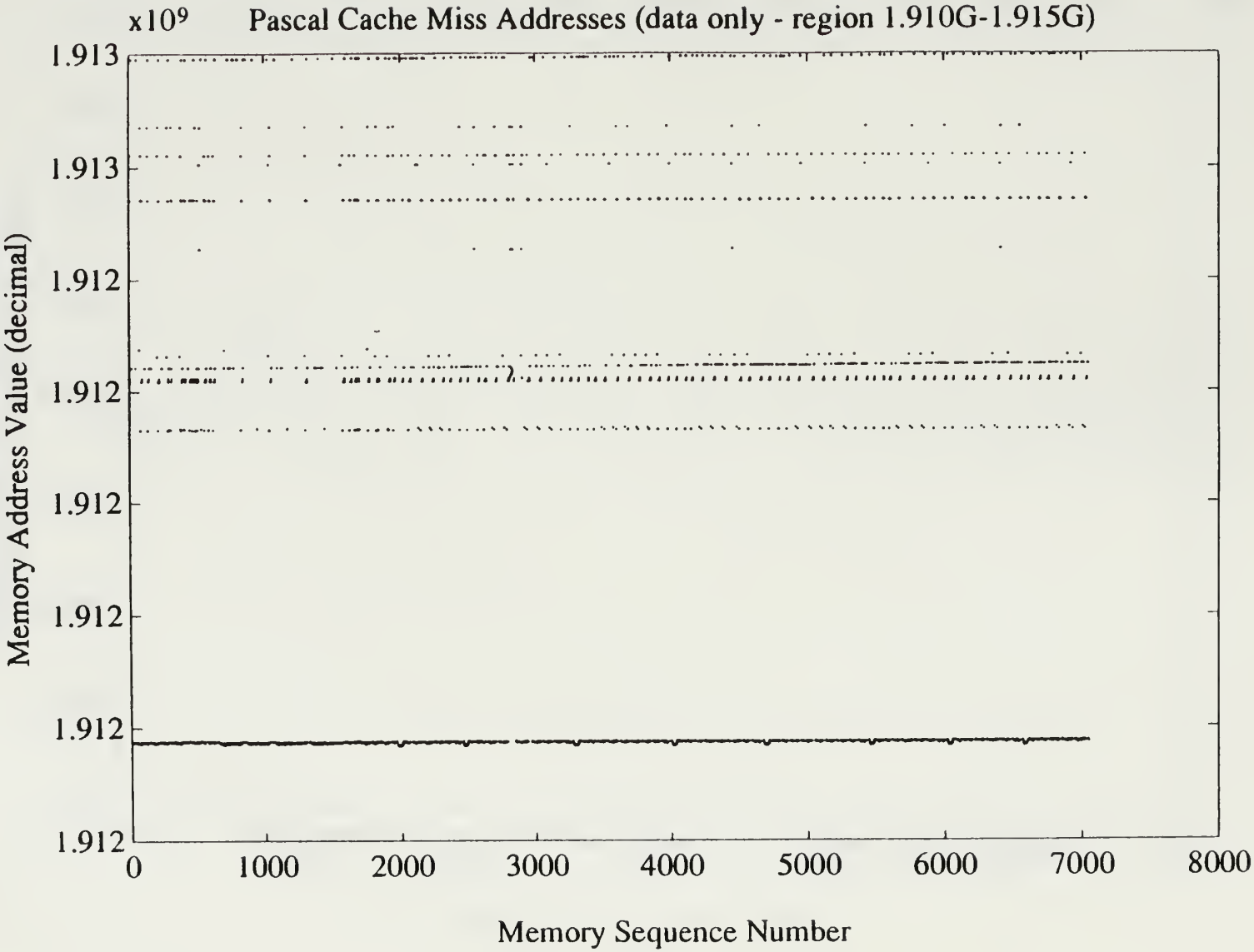
VII. RECOMMENDATIONS FOR FUTURE RESEARCH

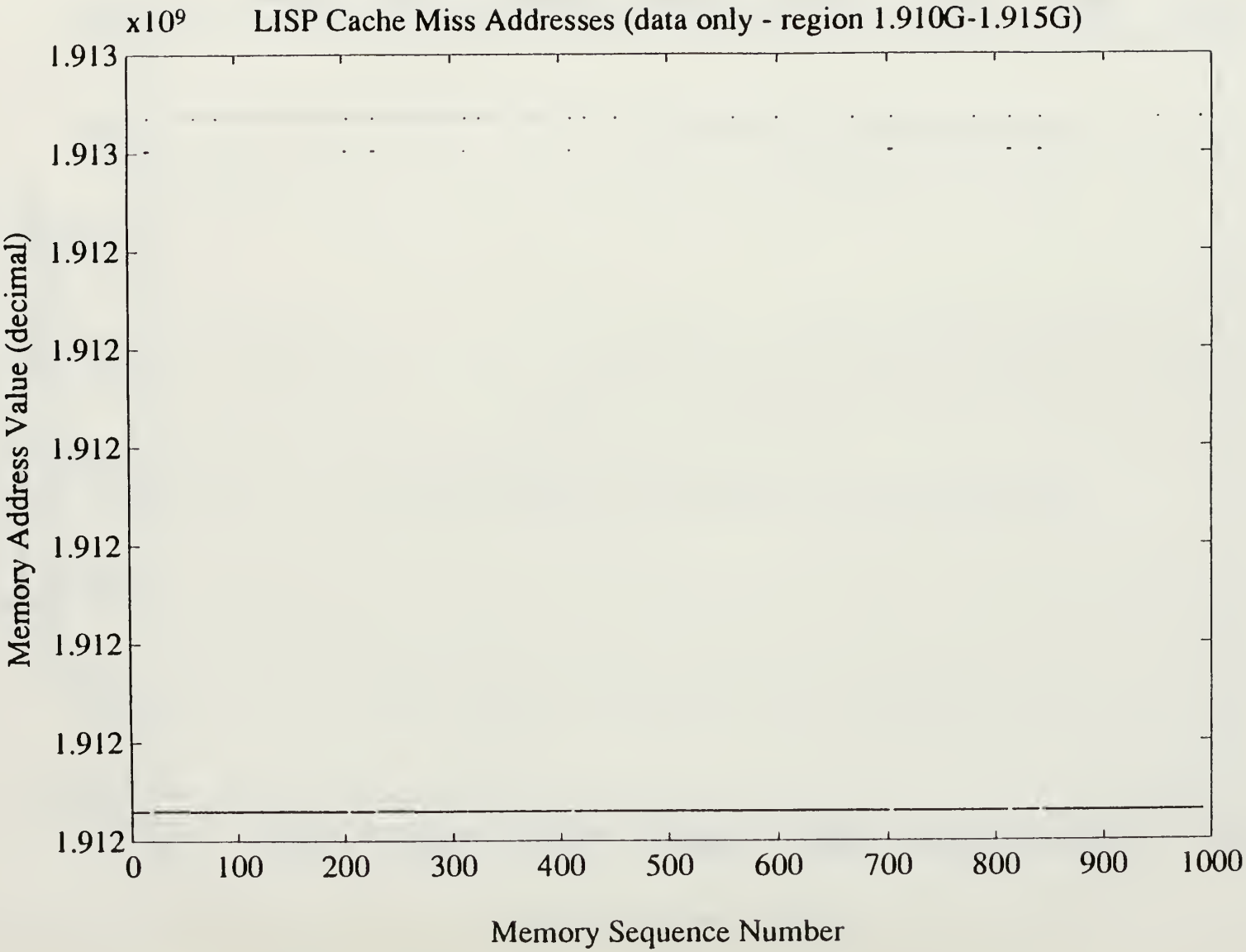
The memory prediction buffer is studied and simulated for enhancement of the data cache of a uniprocessor. Its use or enhancement in a multiprocessor environment is not yet known. In addition, the question of whether the MPB can be used to significantly enhance the performance of the instruction cache has not fully been explored. The algorithm for the MPB of this research focused on a random replacement policy for discarding lines. The LRU replacement policy showed an improvement over random however, the effect of other replacement policies is available for discussion. Simulation and study of the memory bandwidth required to support an architecture with a MPB and without a MPB is needed. A comparison of the amount of bandwidth required by the base architecture (cache and processor) with the bandwidth required by the architecture with a MPB installed, is useful. The cache write-back policy and its effect on systems performance with and without an MPB is an area open for study.

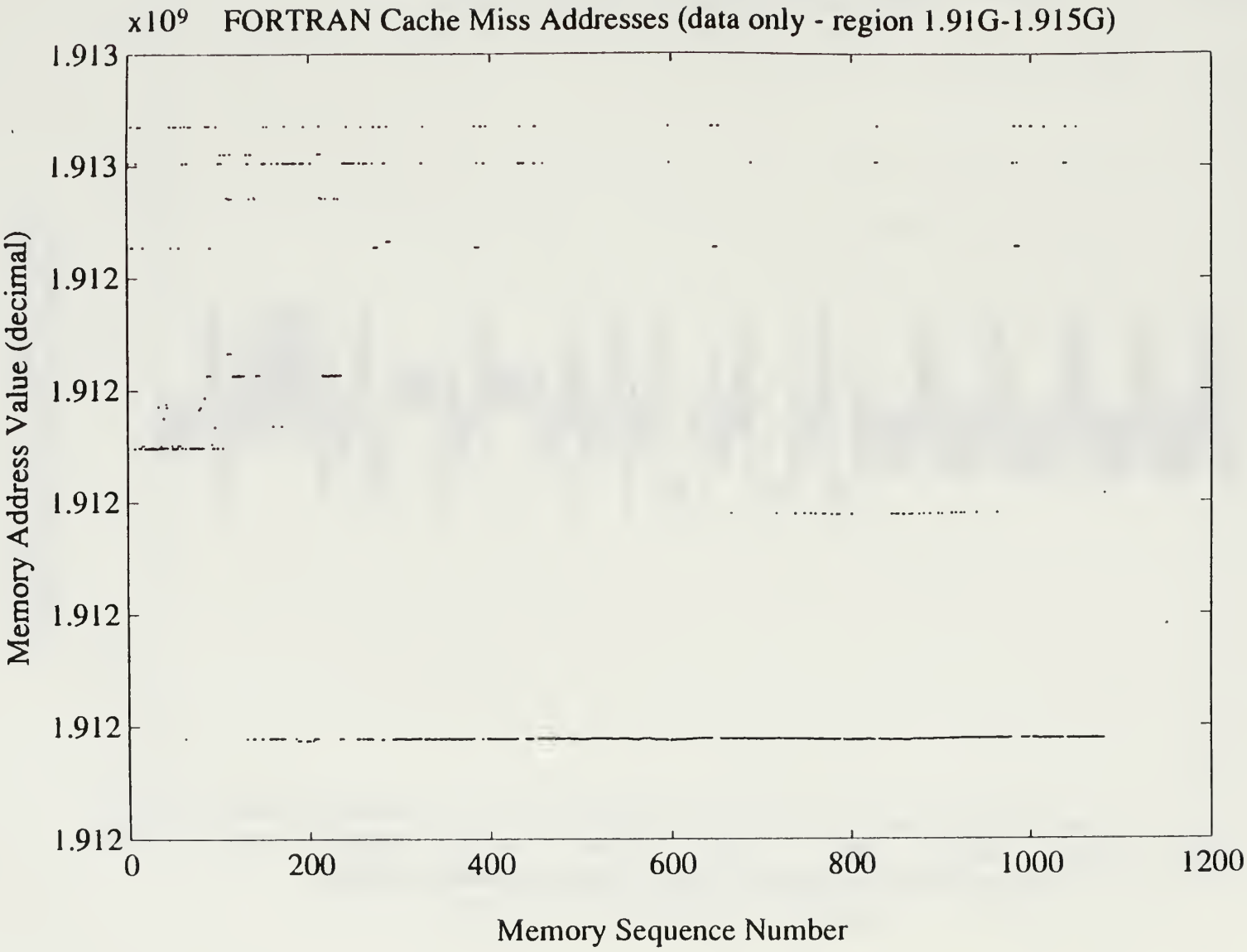

SPICE Cache Hits Before Miss
Cache Hits
Sequence Number
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60
70
80
90
100

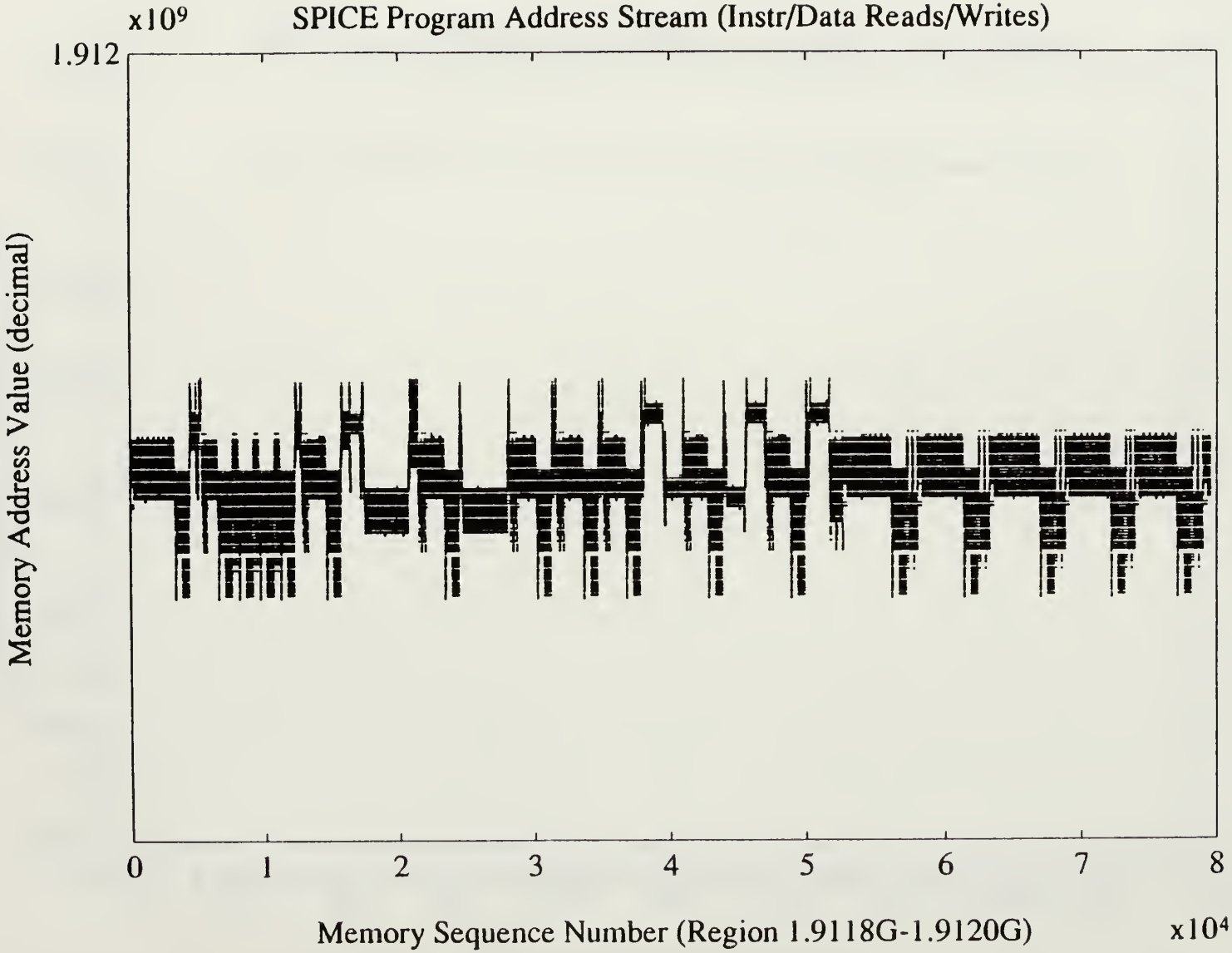

Instruction and Data reads from memory
x10^7
1.9
1.85
1.8
1.75
1.7
1.65
Memory Address Value (decimal)
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
Memory Read Access Sequence

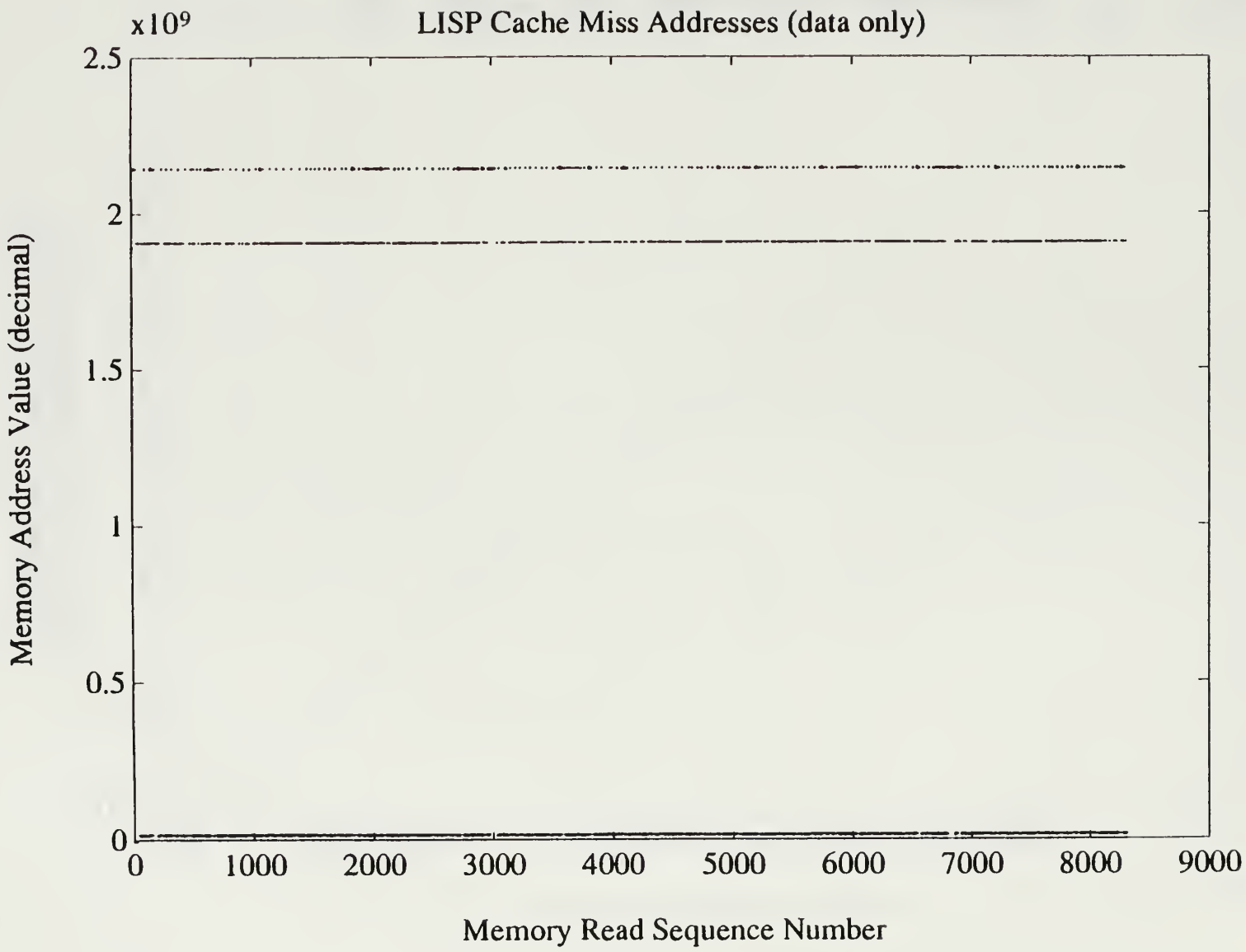

x10^9
SPICE Cache Miss Addresses (data only - region 1.910G-1.915G)
1.913
1.913
1.912
1.912
1.912
1.912
1.912
Memory Address Value (decimal)
0
200
400
600
800
1000
1200
1400
1600
Memory Sequence Number

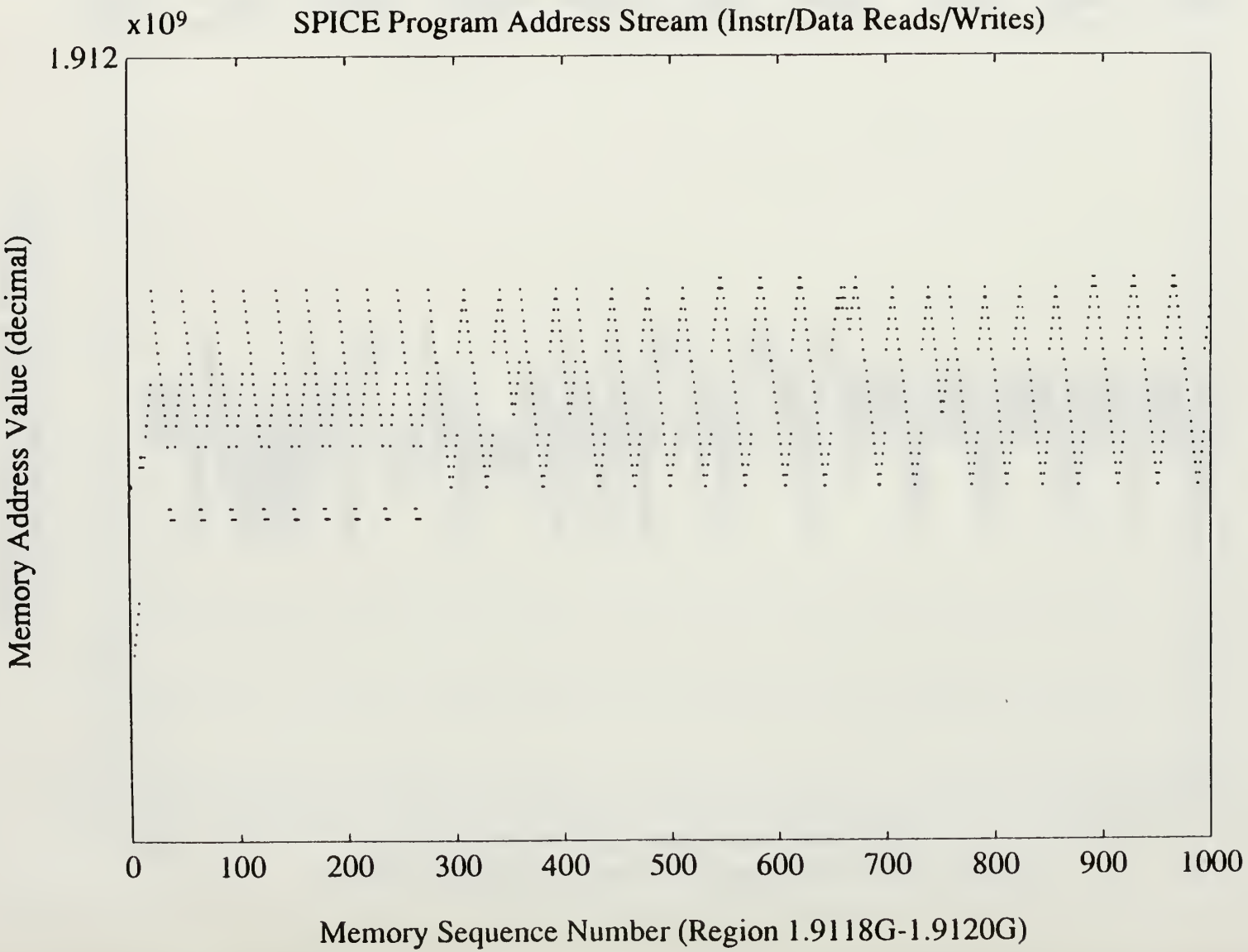

x10^9
Pascal Cache Miss Addresses (data only - region 1.910G-1.915G)
1.913
1.913
1.912
1.912
1.912
1.912
1.912
1.912
Memory Address Value (decimal)
0
1000
2000
3000
4000
5000
6000
7000
8000
Memory Sequence Number

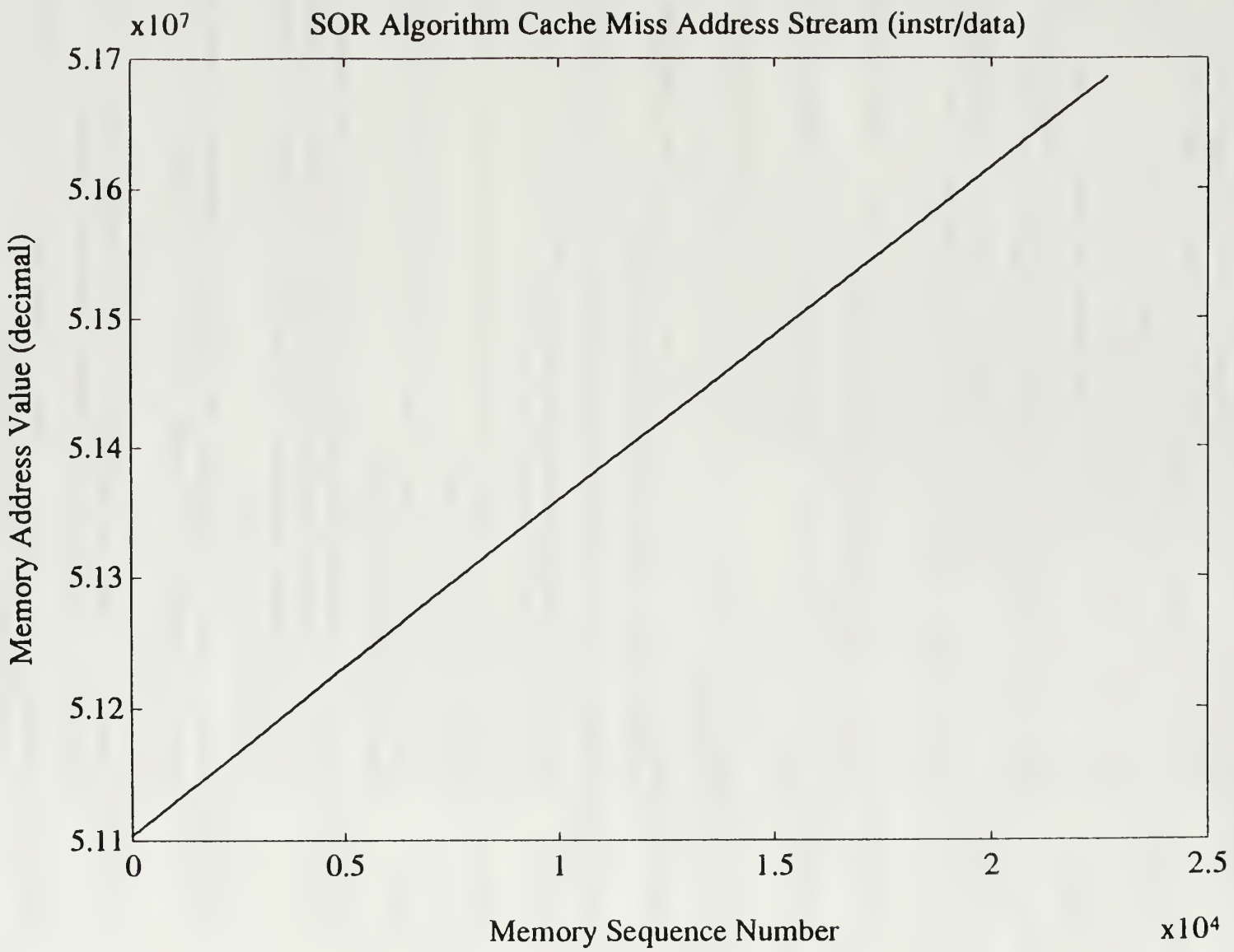

x10^9
LISP Cache Miss Addresses (data only - region 1.910G-1.915G)
1.913
1.913
1.912
1.912
1.912
1.912
1.912
1.912
Memory Address Value (decimal)
0
100
200
300
400
500
600
700
800
900
1000
Memory Sequence Number


x10^9 FORTRAN Cache Miss Addresses (data only - region 1.91G-1.915G)
1.913
1.913
1.912
1.912
1.912
1.912
1.912
1.912
Memory Address Value (decimal)
0
200
400
600
800
1000
1200
Memory Sequence Number


x10^9
SPICE Program Address Stream (Instr/Data Reads/Writes)
1.912
Memory Address Value (decimal)
0
1
2
3
4
5
6
7
8
Memory Sequence Number (Region 1.9118G-1.9120G)
x10^4


x10^9
LISP Cache Miss Addresses (data only)
2.5
2
1.5
1
0.5
0
Memory Address Value (decimal)
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
Memory Read Sequence Number


x10^9
SPICE Program Address Stream (Instr/Data Reads/Writes)
1.912
Memory Address Value (decimal)
0
100
200
300
400
500
600
700
800
900
1000
Memory Sequence Number (Region 1.9118G-1.9120G)


SOR Algorithm Cache Miss Address Stream (instr/data)
x10^7
5.17
5.16
5.15
5.14
5.13
5.12
5.11
Memory Address Value (decimal)
0
0.5
1
1.5
2
2.5
Memory Sequence Number
x10^4

LIST OF REFERENCES

[AGARWL86] Agarwal, A., et al., "ATUM: A New Technique for Capturing Address Traces Using Microcode", *The 13th Annual International Symposium on Computer Architecture*, IEEE Computer Society Press, Los Alamitos, California (vol 14, no3), 1986.

[AZIMI92] Azimi, M. et al, "Two Level Cache Architectures", *COMPCON '92*, IEEE Computer Society Press, Los Alamitos, California, 1992 pg 344-349.

[BORG90] Borg, A., Kessler, R.E., Wall, D.W., "Generation and Analysis of Very Long Address Traces", *The 17th Annual International Symposium on Computer Architecture*, IEEE Computer Society Press, Los Alamitos, California (vol 18, no2), 1990.

[BUGGE90] Bugge, H.O. et al, "Trace Driven Simulations for a Two-Level Cache Design in Open Bus Systems", *IEEE Computer Society Press*, Los Alamitos, California, (vol 18 no2), 1990.

[BURSKY92] Bursky, D., "Combination DRAM-SRAM", *Electronic Design*, Penton Publishing, Cleveland, Ohio, January 1992, (vol 40, no. 2), pg 39.

[CLEMEN91] Clements, A., *Microprocessor Support Chips Sourcebook*, McGraw-Hill Inc., London, England, 1991.

[GAJSKI87] Gajski, D.D. et al, *Computer Architecture*, IEEE Computer Society Press, Washington, D.C., 1987.

[GRIMSR92] Grimsrud, K. et al., "Estimation of Simulation Error Due to Trace Inaccuracies", Brigham Young University, November 1992, unpublished.

[HAMMIN83] Hamming, R.W., *Digital Filters*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[HILL88] Hill, M.D., "A Case for Direct-Mapped Caches", *IEEE Computer*, IEEE Computer Society, Los Alamitos, California, December 1988.

[HWANG84] Hwang, K., Briggs, F., *Computer Architecture and Parallel Processing*, McGraw-Hill, New York, New York, 1984.

[JAIN91] Jain, Raj., *The Art of Computer Systems Performance Analysis*, John Wiley and Sons, New York, New York, 1991.

[JOUPPI90] Jouppi, N.P., "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers", *The 17th Annual International Symposium on Computer Architecture*, IEEE Computer Society Press, Los Alamitos, California (vol 18, no2), 1990.

[KURIAN91] Kurian, L. et al, "Classification and Performance Evaluation of Instruction Buffering Techniques", *IEEE Computer Society Press*, Los Alamitos, California, (vol 19 no 3), 1991.

[NOWICK92] Nowicki, G., "Design and Implementation of a Read Prediction Buffer", Master's Thesis, Naval Postgraduate School, Monterey, California, December 1992.

[PATHEN90] Patterson, D.A. & Hennessy J.L., *Computer Architecture-A Quantitative Approach*, Morgan Kauffman Publishers, San Mateo, California, 1990.

[POHM83] Pohm, A.V., *High-Speed Memory Systems*, Reston Publishing Company, Reston, Virginia, 1983.

[POLLAR90] Pollard, L.H., *Computer Architecture and Design*, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[PRZYBY90] Przybylski, S. A., *Cache and Memory Hierarchy Design: A Performance-Directed Approach*, Morgan Kaufmann Publishers, San Mateo, California, 1990.

[PRZYBY88] Przybylski, S. et al, "Performance Trade-offs in Cache Design", *IEEE Computer Society Press*, Los Alamitos, California, (vol 16 no 2), 1988.

[PRZYBY90] Przybylski, S. A., "The Performance Impact of Block Sizes and Fetch Stategies", *IEEE Computer Society* Press, Los Alamitos, California, (vol 18 no 2), 1990.

[SHORT88] Short, R.T. and Levy, H.M., "A Simulation Study of Two-Level Caches", *The 17th Annual International Symposium on Computer Architecture*, IEEE Computer Society Press, Los Alamitos, California (vol 16, no2), 1988.

[SMITH85] Smith, A.J.,"Cache Evaluation and the Impact of Workload Choice", *IEEE Computer Society Press*, Los Alamitos, California, (vol 18 issue 3), 1985.

[SMITH82] Smith, A.J., "Cache Memories", *ACM Computing Surveys*, New York, New York, 1982, (vol 14, no 3 September).

[THIEBT92] Thiebaut, D., Wolf, J.L., Stone S.S., "Synthetic Traces for Trace-Driven Simulation of Cache Memories", *IEEE Transactions on Computers*, VOL 41 NO. 4, April 1992.

[THERRI92] Therrien, C.W., *Discrete Random Signals and Statistical Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

[VAGTS92] Vagts, C., "A Single Transistor Cell For GaAs Dynamic RAM", Master's Thesis, Naval Postgraduate School, Monterey, California, 1992.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virgnia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5000	2
3.	Chairman, Code EC Department of Electrical and Conputer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
4.	Prof. Douglas J. Fouts, Code EC/FS Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	2
5.	Prof. Richard W. Hamming, Code CS/HG Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	1
6.	Arthur Billingsley, LT, USN Space and Naval Warfare Systems Command Department of the Navy SPAWAR (PMW-156-1) UHF SATCOMM Washington, D.C. 20363-5100	2


DEMCO