마이크로프로세서 캐쉬메모리의 적중률 개선을 위한 제안

정회원 조 용 훈*, 김 정 선**

A Proposal for Hit Ratio Improvement of a Microprocessor's Cache Memory

Yong Hoon Cho*, Jung Sun Kim** *Regular Members*

요 약

현재 사용되고 있는 개인용 컴퓨터의 중앙처리장치로서 주종을 이루고 있는 마이크로프로세서는 256KB, 혹은 512KB의 L2(Second Level) 캐쉬를 Direct Mapping, 32B 라인사이즈, 그리고 Write Allocation을 채택하지 않는 형태로 사용하고 있는데, 이러한 L2 캐쉬에서 Mapping 방식을 8-way Set Associative Mapping Procedure로 바꾸고, 라인사이즈를 늘려서 128B 이상으로 변경하고, 그리고 Write Allocation을 채택하였을 경우 그 적중률(Hit Ratio)이 약간의 하드웨어적 추가 비용만으로 2.5% 정도 개선됨을 확인하였다.

ABSTRACT

A microprocessor, which is used as a CPU for state-of-the-art personal computers, adopts 256KB or 512KB L2(Level 2) cache memory. This cache hires Direct Mapping Procedure, 32B Line Size, and no Write Allocation. In this cache architecture, we can expect about 2.5% hit ratio improvement by using 8-way Set Associative Mapping instead of Direct Mapping, 128B Line Size instead of 32B, and Write Allocation.

Ⅰ. 서 론

오늘날 사용되고 있는 IBM PC의 중앙처리장치인 프로세서는 그 CPU내에 32KB 내부캐쉬(Internal Cache)메모리를 사용하고 있으며, CPU를 두 겹(Dual-cavity)으로 만들어 한 쪽은 CPU, 다른 쪽은 외부캐쉬(External Cache)메모리인 Second Level(L2) 캐쉬로 사용하고 있다. 캐쉬메모리는 그 속도(Speed)가 최대의 관건이므로 L2 캐쉬를 CPU와 아주 가까운 곳에 집적·위치시키기 위한 배려이다. 이 L2 캐쉬는 그 용량이 256KB 혹은 512KB 두 개의 선택사항으로 되어있으며, 그 사상방식(Mapping Procedure)으로는 Direct Mapping을 쓰고 있다. 주기억장치로부터 한번에 읽어오는 정보의 크기인 라인사이즈(Line Size)는 32B로 되어있으며 Write Allocation은 쓰지 않고 있다.[7]

본 논문에서는 이러한 방식으로 운영되고 있는 펜티엄 프로세서의 L2 캐쉬의 성능을 개선시킬 수 있는 방안을 제시한다. 즉, L2 캐쉬의 사상방식을 8-way Set Associative Mapping 방식으로 바꾸고, 그 라인사이즈를 128B이상으로 대체하며, 쓰기 동작 실패(Write Miss) 시에도 해당 라인을 캐쉬로 읽어 오는 Write Allocation을 채택한다. 이렇게 함으로써 약간의 하드웨어적 추가부담이 발생하지만, VLSI 기술 발달로 인한 지속적인 하드웨어 가격 인하에 기인하여 그 부담은 매우 미미해 지며, Direct Mapping과 32B 라인사이즈를 쓰는 경우에 비해 적중률이 2.5% 정도 향상되었다.

* *대구미래대학 컴퓨터정보처리과 ** 한국항공대학교 항공전자공학과
논문번호 : 99336-0823, 접수일자 : 1999년 8월 23일

Ⅱ. 시뮬레이션 결과 분석

상기한 성능분석을 위해서 본 저자가 개발한 시뮬레이터를 사용하였으며, 시뮬레이션 수행 시 좀더 높은 객관성을 부여하기 위하여 기존의 시뮬레이션에서 사용하던 4개의 독립된 실제 응용프로그램 트레이스(Traces: lisp.000.din, pasc.000.din, forf.003.din, macr.000.din) - 미국에 소재하는 MORGAN KAUFMANN PUBLISHERS, INC.에 의해 제작되어 이러한 실험을 필요로 하는 일반에게 자기 테이프로 제공됨 - 에 추가로 9개의 트레이스를 미국 Stanford 대학으로부터 Download 받아 시뮬레이션에 사용하였다.[1]

표 1. Direct Mapping

Name of outfile = temp10.d
Write Policy used = Write_back

Trace Program1 used = forf.dat,	368212
Trace Program2 used = macr.dat,	342828
Trace Program3 used = pasc.dat,	422090
Trace Program4 used = lisp.dat,	291390
Trace Program5 used = fora.dat,	387934
Trace Program6 used = fsxzz.dat,	239334
Trace Program7 used = ivex.dat,	341968
Trace Program8 used = memxx.dat,	444849
Trace Program9 used = mul2.dat,	372104
Trace Program10 used = mul8.dat,	429432
Trace Program11 used = ue02.dat,	357810
Trace Program12 used = dec0.dat,	361982
Trace Program13 used = spic.dat,	446701

Bus Traffic Ratio

a \ b	8K	16K	32K	64K	128K	256K	512K	1M
4	20.45	15.06	11.64	9.73	8.48	7.56	7.26	6.99
8	16.71	12.21	9.25	7.56	6.44	5.64	5.35	5.09
16	27.26	19.69	14.56	11.54	9.64	8.30	7.76	7.28
32	47.18	33.96	24.82	18.71	15.25	12.74	11.72	10.71
64	87.69	61.75	44.94	32.01	25.35	20.48	18.46	16.28
128	176.70	122.73	89.48	61.60	47.73	35.92	31.63	26.01
256	464.08	262.71	193.00	125.60	94.27	67.00	57.38	42.84
512	1119.15	625.21	444.77	280.02	205.44	137.65	113.11	78.16

Hit Ratio

a \ b	8K	16K	32K	64K	128K	256K	512K	1M
4	81.55	86.38	89.34	90.94	91.97	92.70	92.93	93.16
8	85.04	89.01	91.61	93.03	93.95	95.58	94.80	95.03
16	87.98	91.27	93.49	94.75	95.52	96.04	96.24	96.45
32	89.75	92.60	95.55	95.82	96.51	97.00	97.19	97.40
64	90.60	93.37	95.13	96.49	97.15	97.62	97.80	98.03
128	90.58	93.48	95.21	96.69	97.38	97.95	98.14	98.44
256	87.66	93.09	94.90	96.70	97.49	98.14	98.37	98.75
512	85.00	91.81	94.18	96.41	97.35	98.15	98.44	98.88

※ a : Line Size(Byte) b : Cache Size(Byte)

표 2. 8-way Set Associative Mapping

Bus Traffic Ratio

a \ b	8K	16K	32K	64K	128K	256K	512K	1M
4	13.85	10.12	8.26	7.07	6.62	6.58	6.57	6.57
8	11.44	8.18	6.39	5.36	4.82	4.74	4.73	4.73
16	18.46	12.96	9.69	8.01	6.98	6.74	6.71	6.71
32	31.06	21.77	15.58	12.47	10.58	9.87	9.78	9.77
64	54.16	37.48	26.56	20.21	16.76	14.93	14.64	14.62
128	100.78	68.67	48.28	34.82	27.76	23.73	22.62	22.54
256	210.46	136.71	92.27	64.60	46.62	38.38	34.73	34.31
512	492.48	309.21	192.39	129.34	88.43	68.38	58.60	56.68

Hit Ratio

a \ b	8K	16K	32K	64K	128K	256K	512K	1M
4	87.41	90.75	92.30	93.14	93/42	93.44	93.45	93.45
8	89.66	92.59	94.11	94.89	95.23	95.27	95.28	95.28
16	91.77	94.21	95.61	96.25	96.58	96.65	96.65	96.65
32	93.18	95.21	96.53	97.14	97.45	97.55	97.56	97.57
64	94.16	95.94	97.09	97.72	98.02	98.16	98.18	98.18
128	94.66	96.35	97.41	98.08	98.40	98.56	98.59	98.60
256	94.54	96.45	97.58	98.27	98.70	98.86	98.93	98.93
512	93.77	96.06	97.54	98.31	98.79	99.01	99.11	99.12

※ a : Line Size(Byte) b : Cache Size(Byte)

표 3. WA을 채택한 8-way Set Associative Mapping

Bus Traffic Ratio

a \ b	8K	16K	32K	64K	128K	256K	512K	1M
4	13.58	9.62	7.45	5.92	5.16	5.04	5.02	5.01
8	10.21	7.04	5.17	4.04	3.32	3.16	3.14	3.14
16	15.11	10.23	7.15	5.45	4.26	3.88	3.83	3.83
32	24.08	16.34	10.90	7.92	6.00	5.03	4.88	4.86
64	41.39	27.75	18.17	12.44	9.07	6.94	6.40	6.35
128	79.01	51.42	33.55	21.71	14.72	10.53	8.79	8.56
256	177.75	107.42	66.81	42.24	26.06	17.36	12.72	11.80
512	442.94	262.10	146.22	89.30	51.69	30.69	19.90	16.69

Hit Ratio

a \ b	8K	16K	32K	64K	128K	256K	512K	1M
4	89.07	92.26	93.81	94.64	94.97	95.01	95.02	95.02
8	91.75	94.35	95.77	96.47	96.82	96.88	96.88	96.88
16	93.99	95.96	97.15	97.71	98.01	98.09	98.10	98.10
32	95.31	96.84	97.90	98.42	98.67	98.78	98.79	98.79
64	96.08	97.39	98.31	98.82	99.06	99.19	99.21	99.21
128	96.32	97.64	98.49	99.02	99.29	99.42	99.47	99.47
256	95.95	97.59	98.53	99.08	99.41	99.56	99.63	99.64
512	95.03	97.11	98.42	99.05	99.45	99.64	99.73	99.75

※ a : Line Size(Byte) b : Cache Size(Byte)

표1과 표2는 각각 Write Allocation을 채택하지 않은 Direct Mapping과 8-way Set Associative Mapping의, 그리고 표3은 Write Allocation을 채택한 8-way Set Associative Mapping의 시뮬레이션 수행 결과를 보여주고 있는데, 표1에서 보는 바와 마찬가지로 객관성 제고를 위하여 충분한 개수의(13개) 응용프로그램 트레이스를 사용하였다.(표2와 3에서는 트레이스 목록 생략) 기록방식은 펜티엄에서 사용하

는 Write-Back 방식을 채택하였으며, 버스의 폭(Width of Memory Bus)은 PCI(Peripheral Component Interconnect)의 그것인 64bit(8Bytes)를 사용하였으며[7], 캐쉬용량 8KB부터 1MB까지 각각의 라인사이즈 4B부터 512B까지 실험하였다. 또한 Direct Mapping의 경우에는 필요하지 않으나 Set Associative Mapping에서는 반드시 필요한 교환방식(Replacement Algorithm)으로는 LRU(Least Recently Used) 방식을 채택하였으며, 펜티엄에서는 Write Allocation을 사용하지 않으므로 표1과 2의 경우 기록동작 시 실패(Miss)가 발생하면 기록할 데이터를 주기억장치에 직접 기록하도록 하였다. 그리고 현재의 펜티엄에서 Tag RAM의 용량을 반으로 감소시키기 위해 사용한 Sector의 개념은 본 논문에서 도입하지 아니하였다. 표에서 Bus Traffic Ratio(BTR)란, 임의의 응용프로그램 수행 시 L2 캐쉬를 사용하지 않을 경우 캐쉬와 주기억장치간에 주고받는 총 데이터 량에 대한, L2 캐쉬를 사용할 경우 발생되는 총 데이터 량의 비율이다. 일반적으로 라인사이즈가 커지면 Miss Ratio(MR)는 작아지나 BTR은 커진다.[1][5][6] 그러나 고정된 구조의 캐쉬에서는 접근 실패(Miss)가 많이 발생하거나 주기억장치 기록동작이 많이 필요할 경우 이 비율은 높아지며, 크면 클수록 버스전송에 있어 지연(Latency)이 발생한다는 것을 의미하므로 단일 프로세서 시스템에서는 BTR이 100% 이하인 라인사이즈를 채택하는 것이 요망된다. 즉, 표1에서처럼 현재의 펜티엄에서 라인사이즈 256B를 선택하면, 버스의 전송지연 없이 256KB 캐쉬는 98.14%, 512KB 캐쉬는 98.37%의 적중률을 기대할 수 있으나 라인사이즈 512B를 선택하면 BTR이 모두 100%이상으로 되어, 결국 캐쉬 사용으로 인해 데이터 전송 시 지연을 초래하게 되어 그 효율성을 인정받기 어려워진다. 따라서, 실험결과 100% 이상의 BTR이 발생하는 경우는 캐쉬 메모리 사용으로 인하여 버스의 지연이 초래되는 것을 의미하므로[6] 그러한 구조의 캐쉬는 분석대상에서 제외하였다.

Direct Mapping의 경우, 표1에서 보는 바와 같이 Direct Mapping과 32B 라인사이즈를 사용하는 한, 캐쉬사이즈를 256KB에서 512KB로 확장하는 것은 약 0.2% 정도의 적중률 향상뿐이므로 별 의미가 없다. 그러나 라인사이즈 32B를 128B로 4배 늘릴 경우 버스의 지연 없이 0.95%의 향상이 있음을 알 수 있다. 라인사이즈와 캐쉬사이즈를 동시에 늘릴 경우는 약 1.14%의 적중률 개선이 가능함을 보여주고 있다.

표1과 표2를 비교하여보면 Direct Mapping을 8-way Set Associative Mapping으로 바꾼 것만으로 0.55% 이득이 있다. 그러나 라인사이즈와 캐쉬 사이즈를 각각 32B에서 128B로, 256KB에서 512KB로 각각 바꿀 경우 1.59%의 적중률 개선이 있음을 알 수 있다.

표3에서 보는 바와 같이 Write Allocation을 채택하게 되면 상황은 또 달라지게 되어 라인사이즈 128B와 캐쉬사이즈 512KB 사용 시 적중률이 99.47%가 되어 2.47% 향상된다. 이것은 Von Neumann형 컴퓨터 프로그램의 국부성(Locality)을 잘 설명해 주고 있다.

이러한 각 캐쉬구조의 적중률 비교를 위하여 그림1에 Direct Mapping 256KB, 512KB, 8-way 256KB, 512KB, 그리고 Write Allocation을 채택한 8-way 256KB, 512KB 캐쉬구조의 라인사이즈의 변화에 따른 적중률 변화를 나타내었다. 위에서 설명한 바와 같은 캐쉬 사용으로 인한 전송 지연 때문에 그림1의 그래프에서 Direct Mapping의 라인사이즈 512B 부분은 나타내지 아니하였다. 이러한 캐쉬의 경우 way수를 더 늘리는 것은 별 효과가 기대되지 않는다.

Ⅲ. 문제점(Overhead)과 그 해결방안

여기서 세 가지 문제점이 발생된다. 첫째, Direct Mapping의 경우는 각 주기억장치 라인이 캐쉬에 기억될 때 그 기억될 라인이 이미 정해져 있고 태그(Tag)가 하나밖에 없으므로 적중여부의 판단이 간단하여 결정속도가 빠르나, 8-way의 경우는 8개의 태그를 순차적으로 비교해야 하므로 속도가 느려지게 된다. 이를 해결하기 위해서는 8개의 태그를 동시에 병렬 비교하는 약간의 논리회로(병렬 비교기)나 소 용량의 연관기억장치 (Associative Memory)가 필요하다.

둘째, Direct Mapping의 경우는 상기한 이유로 인하여 태그(Tag)가 하나밖에 없으므로 교환 알고리즘(Replacement Algorithm)이 필요치 않으나, 8-way Set Associative Mapping의 경우는 각 캐쉬 라인 번호마다 8개의 주기억장치 라인이 들어와 있으므로 이들 중 어떤 라인을 포기할 것인지를 결정하는 알고리즘이 필요하다. 본 논문에서는 8 개 라인 중 가장 오래 전에 접근된 경험을 갖는 라인을 골라 포기하는 LRU(Least Recently Used) 방식을 사용

하였으며, 이를 구현하기 위해서는 각 캐쉬 라인번호의 각 라인(8개)마다 하나씩의 카운터(Counter)가 필요하다.

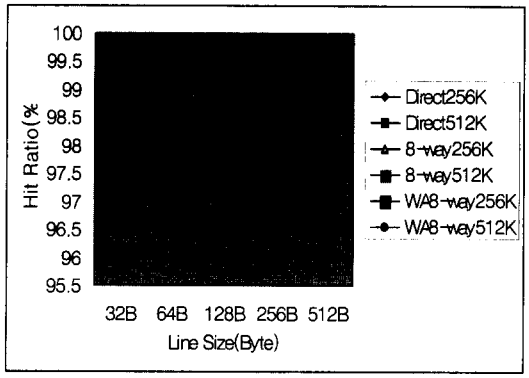

그림 1. Line Size의 변화에 따른 각 Cache 구조의 Hit Ratios

셋째, 현재 펜티엄의 경우 라인사이즈 32B를 인출하기 위해서는 주기억장치를 4회(32Byte/64bit) 접근하여야 한다. 그러나 라인사이즈 128B의 경우에는 16회(128Byte/64bit)의 접근이 필요하므로 많은 전송지연이 발생하게 된다. 따라서 본 논문에서 제시된 2.5%의 적중률 향상을 위해서는 버스 폭을 현재의 64비트에서 256비트로 4배 확장하여야만 한다. 그런데 적중률이 2.5% 향상된다는 것은 실패할 확률이 3%에서 0.53%로 5.6배 줄어든다는 것을 의미하므로 최소한 버스 폭을 전혀 확장하지 아니하여도 어느 정도의 성능 개선을 기대할 수 있는 것이며, 약간의 하드웨어를 추가하여 그 버스 폭을 256비트로 확장해 주면 2.5% 정도의 성능 개선은 무난히 달성할 수 있는 것이다.

위에서 설명한 바와 같이 약간의 하드웨어적인 추가 부담이 발생하나, 오늘날 VLSI 기술의 발전으로 인하여 하드웨어 가격이 점차 하락하고 있으므로 적은 비용으로 이러한 회로를 구현하는 것은 충분히 가능한 일이다.

이상과 같은 간단한 하드웨어 추가로 Direct Mapping의 경우와 거의 비슷한 시간 내에 적중여부를 결정하고, 포기할 라인을 결정할 수 있게 된다.

Ⅳ. 결 론

이상에서 설명한 바와 같이 약간의 하드웨어 비용의 추가부담으로 Direct Mapping을 8-way Set Associative Mapping으로, 그리고 라인사이즈를 32B에서 128B로 변경하고, Write Allocation을 채택해 줌으로써 L2 캐쉬의 적중률(97%)을 99.47%로 향상시킬 수 있다. 상기한 바와 같이 오늘날 VLSI 기술의 눈부신 발전으로 인하여 간단한 병렬 비교기나 카운터의 추가적인 제작, 또는 캐쉬와 주기억장치간 버스 폭의 확장 등에는 그 비용이 별로 많이 들지 아니한다.

이러한 Set Associative Mapping에 의한 적중률 향상은 미래의 다중 프로세서 PC 시대를 대비하기 위해서도 필수적이다. 즉 이중 프로세서(Dual Processor) PC의 경우 하나의 버스를 두 개의 프로세서가 공유하여야 하는데, 표1에 의하면 Direct Mapping에서 적중률 향상을 위하여 그 라인사이즈를 증가시켜 256B나 512B를 쓰면 그 BTR이 50%를 초과하여 67%와 137.65%가 되므로 버스 전송지연이 발생하게 된다. 따라서 128B를 사용하게 되면 그 적중률은 97.95% 밖에 되지 않는다. 그러나 표2의 Set Associative Mapping에서는 라인사이즈 256B까지 증가시켜도 BTR이 50%를 초과하지 않게 되며, 그때의 적중률은 98.86%, 즉 현재의 L2 캐쉬에 비해 2개의 프로세서를 지원하면서도 버스 전송지연 없이 적중률은 1.86% 개선되게 된다. 표3의 경우처럼 Write Allocation을 채택하는 경우에는 사정이 더욱 호전된다. 버스 폭 확장 등과 같은 하드웨어 추가부담에도 불구하고 그 성능을 확장시켜야 할 필요성이 대두될 경우, 이중 프로세서임에도 불구하고 라인사이즈를 512B 이상으로 증가시킬 수 있는 것은 물론이며, 나아가서 이러한 경우에는 채택할 수 있는 프로세서의 수가 한층 더 늘어나는 장점까지 누릴 수가 있게 된다.

참 고 문 헌

[1] Y. H. Cho, J. S. Kim, "The Effects of Cache Memory on the System Bus Traffic", 한국통신학회논문지 제21권 제1호 , Jan. 1996

[2] J. L. Hennessy, D. A. Patterson, "Computer Architecture, a Quantitative Approach", Morgan Kaufmann Publishers, Inc., 1990

[3] D. J. Lilja, "Cache Coherence in Large-Scale Shared-Memory Multiprocessors: Issues and Comparisons", ACM Computing Surveys, Vol. 25, No. 3, Sept. 1993, pp303 - 312

[4] M. D. Hill, A. J. Smith, "Evaluating Associativity in CPU Caches," IEEE Trans. on Computer,

Dec. 1989
[5] J. R. Goodman, "Cache Memory Optimization t o Reduce Processor/Memory Traffic," Journal o f VLSI and Computer Systems, Vol.2, No.1-2, 1987, pp61-86
[6] 조용훈, 김정선, "n-way Set Associative Cache 와 Fully Associative Cache의 성능분석",한국 정보처리학회논문지 제4권 제3호, 1997
[7] Intel Corporation, "82434LX/82434NX PCI, CA CHE AND MEMORY CONTROLLER(PCM C)", URL;http//developer.intel.com/design/pciset s/datashts/290479.htm, Dec. 1994

조 용 훈(Yong Hoon Cho) 정회원

1979년 2월 : 한국항공대학교 항공통신공학과(공학사)
1978년 12월~1982년 2월: (주) 대한항공(선임항공기술사)
1985년 8월 :미국 Purdue Univ. Dept. of Electrical Engineering (공학석사, 병렬처리 전공)
1992년 12월~1994년 2월 : 미국 Ball State Univ. Dept. of Computer Science(교육부 교환교수)
1992년 9월~현재 : 한국항공대학교 대학원 전자공학과 박사과정
1986년 3월~현재 : 대구미래대학 컴퓨터정보처리과 부교수
<주관심 분야> 컴퓨터구조학, 병렬처리

김 정 선(Jeong Sun Kim) 정회원
1965년 2월 : 한국항공대학교 항공전자공학과(공학사)
1972년 2월 : 한양대학교 전자공학과(공학석사)
1983년 2월 : 경희대학교 전자공학과(공학박사)
1994년 6월~현재 : 전국대학정보전산기관협의회 회장
1976년 3월~현재 : 한국항공대학교 교수
2000년 1월~현재 : 한국통신학회 감사
<주관심 분야> 컴퓨터구조학