http://dx.doi.org/10.7236/JIWIT.2012.12.6.175
JIWIT 2012-6-22

캐쉬 용량 효과에 대한 멀티코어 프로세서의 성능 연구

Performance Analysis of Multicore Processor Architectures Based On Cache Size Effects

이종복*

Jongbok Lee

요 약 최근에 이르러, 수퍼스칼라 프로세서의 하드웨어 복잡도와 성능 한계의 문제를 극복하기 위하여 멀티코어 프로세서가 각종 컴퓨터 시스템에 상용화되어 널리 이용되고 있다. 이 때, 멀티코어 프로세서의 성능에 큰 영향을 미치는 것은 명령어 캐쉬와 데이터 캐쉬의 구성 방법과 용량이다. 본 논문에서는 캐쉬의 구조와 용량이 멀티코어 프로세서의 성능에 미치는 영향을 분석하기 위하여, 다양한 캐쉬의 구조와 용량으로 구성되는 2 개에서 16 개까지의 멀티코어 프로세서에 대하여 SPEC 2000 벤치마크를 입력으로 하여 모의실험을 수행하였다. 모의실험 결과, 명령어 캐쉬와 데이터 캐쉬의 구조를 2 차 연관도로 구성하고 각 용량을 64 KB로 설정할 때 하드웨어의 비용 대 성능 효과가 가장 높았다.

Abstract In order to overcome the complexity and performance limit problems of superscalar processors, the multicore architecture has been prevalent recently. The configuration and the size of instruction and data caches greatly gives effect on the performance of multicore processors. Using SPEC 2000 benchmarks as input, the trace-driven simulation has been performed for the 2-core to 16-core architectures with different sizes of caches extensively. As a result, the 2-way set associative instruction and data cache with the size of 64KB brought the best cost-effective performance.

Key Words : multicore processor, instruction cache, data cache.

Ⅰ. 서 론

최근 30년간 마이크로 프로세서에 관한 연구는 싱글코어 프로세서를 더욱 빠르게 실행하도록 설계하는 것을 목표로 하였으며, 수퍼스칼라 프로세서가 그 대표적인 예이다[1]. 그러나, 이러한 수퍼스칼라 프로세서는 하드웨어가 지나치게 복잡하여 성능 향상의 한계에 봉착하였다. 이것을 해결하기 위하여 멀티코어 프로세서가 대두되었다[2-6]. 멀티코어 프로세서는 응용 프로그램을 병렬화할 수 있다는 것을 전제로 할 때, 프로세서의 성능을 높이는 유일한 해결책이다. 이러한 멀티코어 프로세서에서 명령어 캐쉬와 데이터 캐쉬가 성능에 미치는 영향이 매우 크다.

본 논문에서는 명령어 캐쉬 및 데이터 캐쉬의 구조와 용량이 멀티코어 프로세서의 성능에 끼치는 영향을 분석

*정회원, 한성대학교 정보통신공학과
접수일자 : 2012년 8월 13일, 수정완료 : 2012년 10월 29일
게재확정일자 : 2012년 12월 14일

Received: 13 August 2012 / Revised: 29 October 2012 / Accepted: 14 December 2012
**Corresponding Author: jblee@hansung.ac..kr
Dept. of Information & Communications Engineering, Hansung University, Korea

하기 위하여, 다양한 용량의 캐쉬로 구성되는 2-코어에서 16-코어의 멀티코어 프로세서에 대하여 SPEC 2000 벤치마크를 입력으로 모의실험을 수행하여 그 성능을 측정하였다. 그 결과, 멀티코어 프로세서 시스템에서 캐쉬 하드웨어의 비용 대 성능이 가장 효율적인 캐쉬의 구조와 용량을 제안하였다.

본 논문은 다음과 같이 구성된다. 2 장에서 멀티코어 프로세서, 캐쉬 및 모의실험기에 대하여 살펴보고, 3 장에서 모의실험 환경에 대하여 고찰한다. 4 장에서 모의실험 결과를 보이며, 5 장에서 결론을 맺는다.

II. 멀티코어 프로세서 및 캐쉬

1. 멀티코어 프로세서의 구조

그림 1은 N 개의 코어로 구성되는 멀티코어 프로세서의 일반적인 구조를 나타낸 것이다. 각 코어는 1부터 N까지 구성되는데, 자체적으로 1 차 명령어 캐쉬와 1 차 데이터 캐쉬를 가진다. 또한 각 코어는 메인 메모리와 연결되는 공통의 2 차 통합 캐쉬를 공유한다. 각 코어에 설치된 1 차 데이터 캐쉬의 일관성(cache-coherency)을 위하여 MESI 프로토콜을 이용한다. 이하 본 논문에서 기술하는 명령어 캐쉬와 데이터 캐쉬는 모두 1 차 캐쉬를 의미한다.

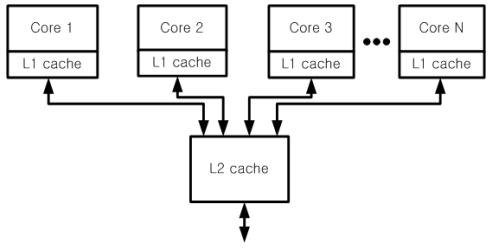


그림 1. 멀티코어 프로세서의 구조
Fig 1. The multicore processor architecture

2. 멀티코어 프로세서를 구성하는 캐쉬

기존의 RISC나 수퍼스칼라 프로세서의 명령어 캐쉬가 성능에 끼치는 영향력이 커서, 전통적으로 2 차 이상의 연관도 (2-way set associative) 및 충분한 용량으로 구성하여 캐쉬 미스에 의한 성능의 손실을 최소화하였다. 이것을 멀티코어 프로세서에서도 적용할 수 있으며, 2 차 이상의 연관도로 명령어 캐쉬를 구성하고 그 용량에 따라 충분한 캐쉬 히트율을 확보할 수 있다.

한편, RISC 및 수퍼스칼라 프로세서에서 데이터 캐쉬는 명령어 캐쉬와는 달리, 직접 캐쉬(Direct-Mapped Cache)로 구성해도 충분한 성능을 얻을 수가 있었다. 그러나, 멀티코어 프로세서 시스템에서는 여러 코어 간의 캐쉬 일관성(Cache Coherency)을 위한 MESI 프로토콜의 적용으로 인하여 캐쉬의 데이터를 무효화하는 경우가 빈번히 발생한다. 따라서, 직접 캐쉬로는 적정한 캐쉬 히트율을 확보할 수가 없어서 멀티코어 프로세서의 성능 손실이 크므로, 데이터 캐쉬 역시 2 차 이상의 연관도로 구성해야한다.

본 논문에서는 하드웨어 비용을 절감하고 멀티코어 프로세서 아키텍처 성능의 극대화를 위하여, 명령어 캐쉬와 데이터 캐쉬를 모두 2 차 연관도로 구성하였다. 그리고 각 코어의 대칭성을 위하여 명령어 캐쉬와 데이터 캐쉬의 용량을 동일하도록 멀티프로세서 시스템을 설계하였다. 또한, 비용 대 성능면에서 최적인 캐쉬의 용량을 찾기 위하여, 다양한 캐쉬의 용량에 대한 모의실험을 시행하였다.

III. 모의실험 환경

1. 멀티코어 프로세서 모의실험기

본 논문에서 명령어 자취형 (trace-driven) 멀티코어 모의실험기를 개발하였다[7]. 멀티코어 프로세서는 제 1 단계 명령어 자취의 발생, 제 2 단계 명령어 자취에 대한 멀티코어 프로세서의 실행으로 나누어진다. 제 1 단계에서 명령어 자취는 임의의 차수의 멀티코어에 적합하도록 발생되었다.

제 2 단계에서 각 코어가 단순한 RISC 프로세서로 동작하여 멀티코어 프로세서가 실행된다. 제 2 단계의 과정을 자세하게 기술하면 그림 2에 나타낸 것과 같다.

(1) 명령어 인출, 재명명 및 이슈

Initialize 함수에서 초기화 작업을 거친 후에, Grouping 함수가 Create_Window 함수를 부르고 이것은 다시 Fetch_One_Instr 함수를 호출하며, 각 코어는 매 싸이클마다 1 개의 명령어를 인출받는다. Get_Node 함수에서 인출한 명령어는 Rename 함수에서 재명명(renaming) 작업을 거치면서 명령어 종속에 의한 타임스

탬프(timestamp) 값을 설정받는다.

타임스탬프 방식은 명령어 자취를 이용하는 모의실험에서 데이터 종속성을 신속하고 효율적으로 부여할 수 있는 핵심적인 방법이다. 레지스터 화일의 타임스탬프 값에 의하여, 멀티코어 프로세서 명령어 간의 종속성이 유지되어 성능을 구하는데 반영된다. 이와 같이 재명명을 거친 명령어는 insert 함수에서 각 코어의 명령어 윈도우에 삽입된다.

Issue 함수로 진행하면, 싸이클이 증가함에 따라서 윈도우 내의 명령어는 자체의 타임스탬프 값이 현재 싸이클 보다 작거나 같을 때 삭제될 수 있다.

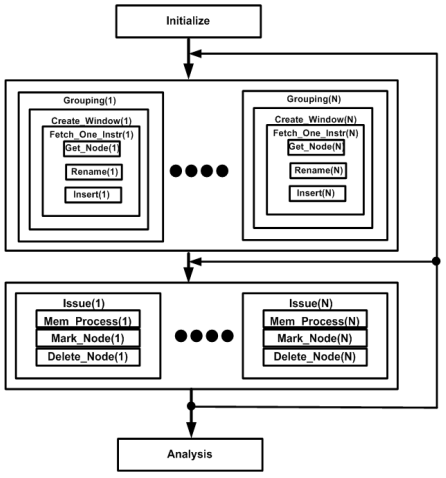


그림 2. 멀티코어 프로세서 모의실험기
Fig 2. The multicore processor architecture simulator

(2) 멀티코어 시뮬레이션

모든 N 개의 멀티코어에 대하여 해당 코어의 윈도우 공간에 Grouping 함수를 이용하여 한 개의 명령어를 인출해서 채우고, 역시 N 개의 멀티코어에 대하여 Issue 함수로 각 코어에 대하여 명령어를 실행하면서 종속성에 의하여 부여된 명령어의 타임스탬프가 충족되면 삭제한다. 이 과정은 입력으로 주어진 벤치마크 프로그램의 모든 명령어가 소진될 때까지 반복된다.

위 과정이 한번 실행될 때 마다 싸이클이 증가하므로, 매 싸이클 당 명령어의 실행 및 삭제가 가장 오래 걸리는 코어가 해당 싸이클 수를 결정한다. 모의실험에 입력으로 쓰인 명령어의 총 개수를 처리하기 위하여 소요된 총 싸이클 수로 나누어, 멀티코어 프로세서 시스템의 성능의 척도인 IPC(Instruction Per Cycle)를 계산할 수 있다.

2. 벤치마크 및 멀티코어 프로세서의 사양

표 1은 모의실험에 이용된 SPEC 2000 정수형 벤치마크 프로그램이다. Simplescalar를 통하여 MIPS IV 10억 개의 명령어 자취를 임의의 차수의 멀티코어 프로세서에 적합하도록 발생시켜서 모의실험기에 입력하였다[8].

표 2는 모의실험에 이용된 멀티코어 프로세서 아키텍쳐의 사양을 나타낸 것이다. 멀티코어의 개수는 1 개, 2 개, 4 개, 8 개 및 16 개를 대상으로 하였다. 각 코어는 RISC 방식으로 운영되므로, 매 싸이클 마다 1 개의 명령어를 인출, 이슈, 실행 및 기록한다. 각 코어의 연산유닛은 정수형 유닛, 로드 스토어 유닛, 그리고 분기명령어 유닛 각 1 개로 구성된다.

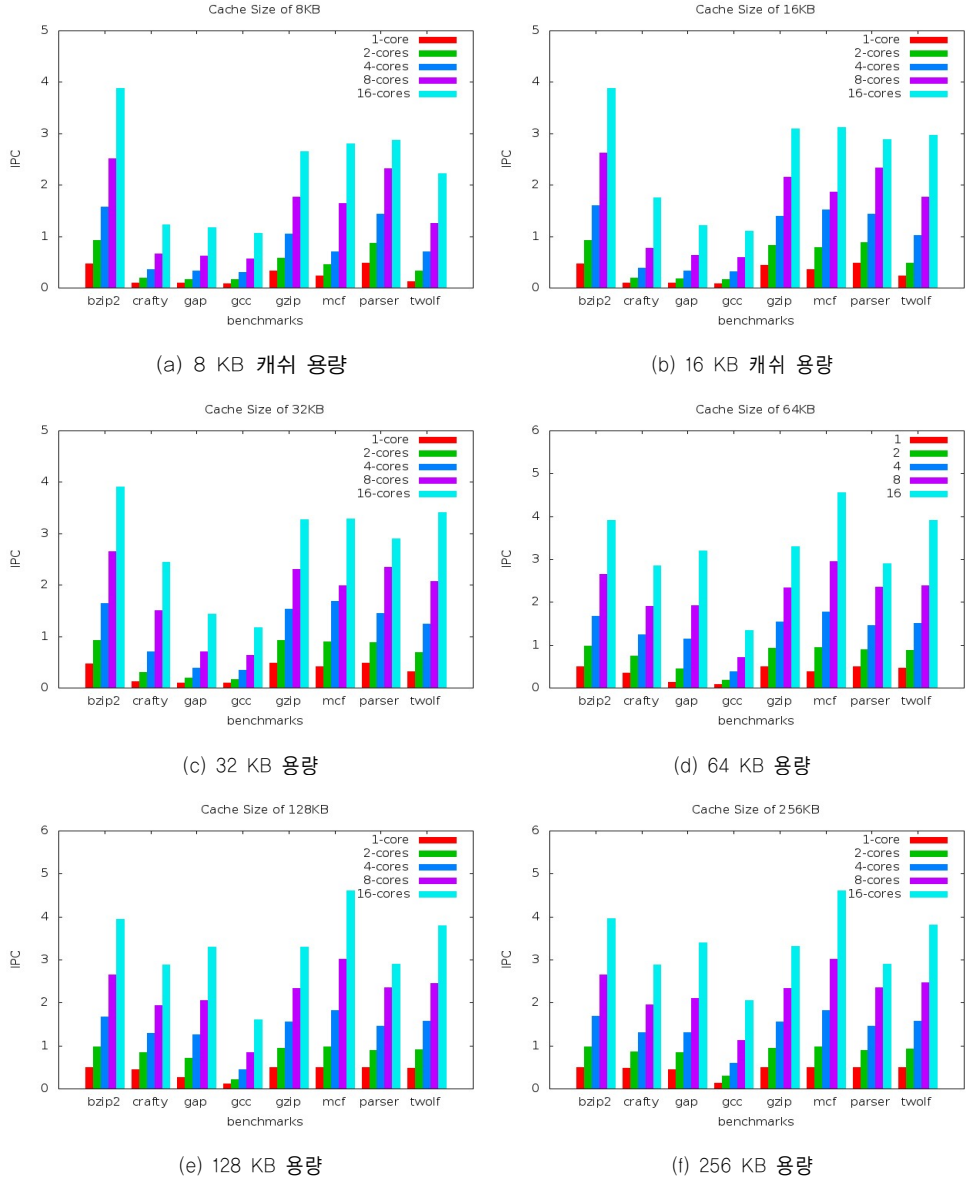
표 1. SPEC 2000 정수형 벤치마크 프로그램
Table 1. SPEC 2000 integer benchmark programs

벤치마크	설 명
bzip2	압축
crafty	체스 경기 놀이
gap	그룹 이론 해석기
gcc	C 프로그래밍 언어 컴파일러
gzip	압축
mcf	조합 최적화
paser	워드 프로세서
twolf	배선 및 배치 모의실험기

명령어 캐쉬와 데이터 캐쉬는 각 코어마다 설치되는데, 8KB에서 256 KB의 6 가지의 용량을 갖도록 설정하였으며, 2 차 연관도(2-way set associativity) 방식을 통하여 접근된다. 그러나, 모든 코어에 의하여 공유되는 2 차 캐쉬는 충분한 용량으로 인하여 100 % 히트가 난다고 가정하였다. 한편, 각 코어에 태스크를 할당하는 태스크 예측기는 2 단계 적응형 예측 방식을 적용하였으며, 태스크 어드레스 캐쉬는 2048 개의 엔트리를 갖는다.[9]

Ⅳ. 모의실험 및 결과

그림 3(a)부터 3(f)까지 8KB에서 256KB 용량의 명령어 캐쉬와 데이터 캐쉬로 구성되는 1-코어에서 16-코어 프로세서에 대하여, 8 개의 정수형 SPEC 벤치마크를 입력으로 하는 모의실험 결과를 보였다. 모의실험 결과에



(a) 8 KB 캐쉬 용량 (b) 16 KB 캐쉬 용량

(c) 32 KB 용량 (d) 64 KB 용량

(e) 128 KB 용량 (f) 256 KB 용량

그림 3. 캐쉬 용량별 멀티코어 프로세서의 성능
Fig 3. Multicore processor performance for various cache sizes.

서 알 수 있듯이, 각 벤치마크 프로그램 별로 코어의 개수가 증가할 수록, 캐쉬의 용량이 증가할 수록성능이 증가함을 알 수 있다. 항목별 성능 결과에 대하여 기하평균 값을 구하면, 코어의 개수가 2 배가 될 때 최저 1.6배, 최고 2.0 배 성능이 향상되었으며, 캐쉬의 용량이 2 배가 될 때 최저 1.1 배에서 최고 1.3 배의 성능이 향상되었다.

캐쉬의 용량이 64 KB일 때, 각 코어의 차수별 기하평균을 구하면 단일코어의 경우 0.31 IPC를 기록하였으며, 단일코어 대비 2-코어에서 0.66 IP로 성능이 2.2 배가 되었다. 같은 기준으로 4-코어에서는 4.3 배인 1.31 IPC, 8-코어에서 8.2 배인 2.5 IPC를 나타냈다. 마지막으로 16-코어의 경우 4.06 IPC로 단일코어 대비 13.3 배의 성능을 가져왔다. 위 결과는 4-코어 때 약 4 배, 8-코어 때 약 8 배, 16-코어 때 10 배~15 배의 성능향상을 기록하는 기존

의 멀티코어 프로세서 모의실험기를 통하여 얻은 결과와 일치하였다[10][11].

표 2. 모의실험에 이용된 멀티코어 프로세서 아키텍쳐 하드웨어의 사양
Table 2. The architecture specification of each core

항목	값
멀티코어 수	1, 2, 4, 8, 16
명령어 캐쉬 및 데이터 캐쉬의 공통 사항	8 KB~256 KB, 2 차 연관, 16 B 미스 페날티 10 싸이클
연산유닛 사양	산술논리(1), 분기(1), 로드(1), 스토어(1)
태스크 어드레스 캐쉬	2 K 엔트리
태스크 예측기	2 단계 14 비트 전역 히스토리 방식 미스 페날티 6 싸이클
이슈 지연 싸이클	산술논리(1), 분기(1), 로드(1), 스토어(1)
결과 지연 싸이클	산술논리(1), 분기(1), 로드(1), 스토어(1)

그림 4는 명령어 캐쉬와 데이터 캐쉬의 용량을 8 KB에서 256 KB까지 2 배씩 증가시켰을 때, 각 벤치마크 프로그램으로 부터 측정한 멀티코어 프로세서 성능의 기하평균을 그래프를 통하여 일목요연하게 나타낸 것이다. 이 그래프에서 알 수 있듯이, 멀티코어 프로세서의 성능은 캐쉬의 용량이 가장 작은 8 KB일 때 최저를, 가장 큰 256 KB일 때 최고를 기록하였다.

그러나 멀티코어 프로세서 성능의 향상률을 구간별로 분석하면, 캐쉬의 용량이 32 KB에서 64 KB로 증가했을 때, 1-코어에서 16-코어에 걸쳐 평균 26 %로 가장 높은 값을 기록하였다. 특히, 그 중에서 2-코어일 때 31 %로 최고 상승률을 기록하였다. 그러나, 캐쉬의 크기를 128 KB나 256 KB로 증가시킬 경우 성능의 향상률은 6 % 미만으로, 투입된 캐쉬 용량 증대에 비하여 성능 향상에 미치는 효과가 미미함을 알 수 있다.

따라서, SPEC2000 벤치마크를 입력으로 하는 최고 16개의 멀티코어 프로세서에서, 명령어 캐쉬와 데이터 캐쉬를 2 차 연관도로 구성하고, 용량을 각각 64 KB로 설정하는 것이 가장 효율적이다.

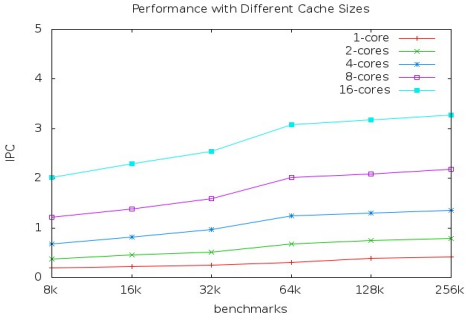


그림 4. 캐쉬 용량에 따른 멀티코어 프로세서의 평균 성능
Fig 4. Average performance of multicore processors with different cache sizes.

V. 결 론

본 논문에서는 다양한 용량의 명령어 및 데이터 캐쉬로 구성되는, 2 개부터 16 개까지의 멀티코어 프로세서 아키텍처에 대하여 SPEC 벤치마크를 입력으로 하여 모의실험을 통하여 성능을 측정하고 분석하였다. 그 결과, 명령어 캐쉬와 데이터 캐쉬의 크기를 모두 64 KB의 2 차 연관도로 구성할 때 가격대 성능 면에서 가장 효율적이라는 것을 알 수 있었다.

추후로, 동질 코어(homogeneous core)가 아닌 비동질 코어(heterogeneous core)를 채택하는 비대칭 칩 멀티프로세서(asymmetric chip multiprocessor) 구조의 캐쉬에 대한 연구가 필요하다. 비대칭 칩 멀티프로세서에서는 코어와 같이 캐쉬의 구조 및 용량도 비대칭적이므로, 이 때 최적의 캐쉬 구조 및 용량에 대한 연구 및 모의실험이 필요하다. 더 나아가, 최신 경향인 코어 수 64개 이상의 매니코어 (many-core) 아키텍쳐에 대한 캐쉬의 연구 및 모의실험도 수행할 예정이다.

참 고 문 헌

[1] P. K. Dubey, G. B. Adams III, and M. J. Flynn, "Instruction Window Size Trade-Offs and Characterization of Program Parallelism," IEEE Transactions on Computers, vol. 43, pp 431-442, Apr. 1994.

[2] D. E. Culler and J. P. Singh, "Parallel Computer

Architecture," Morgan Kauffmann Publishers, Inc. Aug. 1998.

[3] S. W. Keckler, K. Olukotun, and H. P. Hofsee, "Multicore Processors and Systems," Springer. 2009.

[4] T. Ungerer, B. Robic, and J. Silk, "Multithreaded Processors," The Computer Journal, Vol. 45, No. 3, 2002

[5] D. Pham et. al, "The Design and Implementation of a First-Generation CELL processor," ISSCC 2005.

[6] D. Genbrugge and L. Eeckhout, "Chip Multiprocessor Design Space Exploration through Statistical Simulation," IEEE Transactions on Computers 58(12), pp.1668-1681, Dec. 2009.

[7] A. Rico, A. Duran. F. Cabarcas, Y. Etsion, A. Ramirex, and M. Valero, "Trace-driven Simulation of Multithreaded Applications," ISPASS, 2011.

[8] T. Austin, E. Larson, and D. Ernest, "SimpleScalar : An Infrastructure for Computer System Modeling," Computer, vol. 35, no. 2, pp. 59-67, Feb. 2002.

[9] T-Y. Yeh and Y. N. Patt, "Alternative Implementations of Two-Level Adaptive Branch Prediction," in Proceedings of the 19th International Symposium on Computer Architecture, pp.124-134, May. 1992,

[10] S. Biswas, et. al, "Multi-Execution : Multicore Caching for Data-Similar Executions," Proceedings of the 36th Annual International Symposium on Computer Architecture, pp.164-173.

[11] M. Monchiero, et. al, "How to Simulate 1000 Cores," ACM SIGARCH Computer Architecture News archive, Vol. 37, Issue 2, pp. 10-19, May 2009.

저자 소개

이 종 복(정회원)

- 1964년 8월 20일생.
- 1988년 : 서울대 컴퓨터공학과 졸업.
- 1998년 : 동 대학 전기공학부 졸업 (공박).
- 1998년~2000년 : LG반도체 선임연구원.
- 2000년~현재 : 한성대 정보통신공학과 교수
- Tel : 02-760-4497
- Fax : 02-760-4435
- E-mail : jblee@hansung.ac.kr

※ 본 연구는 한성대학교 교내연구장려금 지원과제 임.