임베디드 프로세서를 위한 에너지 효율의 명령어 캐쉬 계층 구조

강진구° 이인환

한양대학교 전자통신컴퓨터공학과

jkkang@csl.hanyang.ac.kr° ihlee@hanyang.ac.kr

Energy-Efficient Instruction Cache Hierarchy for Embedded Processors

Jinku Kang° Inhwan Lee

Dept. of Electronics and Computer Engineering, Hanyang University

요 약

계층적 메모리 구조는 성능 향상 이외에도 하위 캐쉬로의 접근을 줄임으로서 전체적인 소비 전력 효율을 높이는 방법으로 사용될 수 있다. 본 논문에서는 임베디드 프로세서의 대표적인 StrongARM의 단일 계층 구조를 대상으로 프로세서에 근접한 명령어 캐쉬를 새로 추가하여 첫 번째와 두 번째 계층의 명령어 캐쉬 크기에 따라 변화하는 소비 전력을 모의실험을 통해 측정하고 두 계층의 명령어 캐쉬 크기에 따른 상호 관계에 대해 알아본다. 직접 사상과 32B의 블록 크기를 갖는 L0 명령어 캐쉬를 삽입하여 에너지 효율이 가장 높은 크기를 찾아보고 효율적 크기에서 소비전력을 측정한 결과 온 칩 구조로 가정한 프로세서 전체의 소비 전력이 최대 약 65%로 감소됨을 볼 수 있으며, L1 명령어 캐쉬가 두 배씩 증가함에 따라 에너지 효율적인 L0 명령어 캐쉬의 크기 또한 두 배씩 증가함을 알 수 있다.

1. 서 론

최근 임베디드 시스템을 설계하는데 있어서 충분한 성능을 얻기 위해 고성능 프로세서를 사용하는 경우가 증가하고 있다. 또한, 고성능 프로세서를 뒷받침하기 위해 평균 데이터 접근 시간을 줄이는 방법으로 계층적 메모리 시스템을 사용한다. 계층적 메모리 시스템은 주 메모리와 프로세서 사이에 캐쉬 메모리를 삽입하여, 하위 계층에 위치하는 메모리로의 접근을 줄임으로서 성능을 높이는 방법으로 임베디드 프로세서들 중 StrongARM[2]의 경우 단일 계층 캐쉬를 채용하고 있다.

임베디드 시스템에 있어서 충분한 성능뿐만 아니라 소비 전력을 최소화 하는 것 또한 중요한 논의 중 하나이다. 임베디드 프로세서중 대표적인 StrongARM의 경우 캐쉬 메모리가 칩 내에 위치한 형태로 캐쉬 메모리가 소비하는 전력은 전체의 약 43%를 차지하며, 특히 명령어 캐쉬의 경우 27%로, 전체 캐쉬 대비 63%에 해당하는 전력을 소비한다[2]. 최근 캐쉬 소비 전력을 줄이는 연구들이 진행 되었는데, 이중 상대적으로 소비전력이 많은 명령어 캐쉬를 대상으로 계층을 하나 더 추가 하는 방법으로 전체적인 소비 전력을 줄이는 연구가 있다.

본 논문에서는 모의실험을 통해 StrongARM을 대상으로 단일 계층 캐쉬 구조에 크기가 작은 명령어 캐쉬를 추가하여 이중 계층 캐쉬 구조로 전환함으로서 얻을 수 있는 에너지 효율에 대해 알아보고, 가장 좋은 효율을 갖는 두 계층 간의 관계를 살펴보고자 한다.

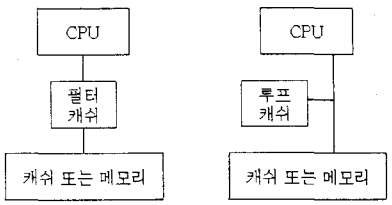

그림 1 필터 및 루프 캐쉬

2. 계층 구조를 통한 소비 전력 감소

계층 명령어 캐쉬 구조를 통해 전력을 줄이는 연구로 프로세서와 1차 명령어 캐쉬 사이에 작은 크기의 명령어 캐쉬를 삽입하여 주 캐쉬로의 접근 빈도를 줄이는 연구가 있었다[3]. 이 연구에서는 그림1에서 보는 바와 같이 직접 사상의 전통적인 캐쉬를 이용한 방법과 태그를 제거한 루프 캐쉬(Loop Cache)의 두 가지 형태를 접근하였다. 그림1의 우측과 같은 루프 캐쉬의 경우 태그를 제거하여 추가적인 명령어 'sbb'와 컴파일러의 지원을 통해 동작하게 된다. 루프 캐쉬는 전통적인 캐쉬 구조에서 태그를 제거함에 따라 하드웨어로 캐쉬 적중과 실패를 판별 할 수 없다. 따라서 새로 추가된 명령어 'sbb'를 이용한다. 먼저 컴파일러에서 작은 루프를 검색하여 루프의 시작점과 끝나는 지점에 'sbb' 명령어를 추가함으로서 루프 시작점과 끝점을 프로세서에 알려주게 된다. 그렇게 되면, 수행 시간에 프로세서는 시작 'sbb'와 다음의 'sbb' 명령어 사이에 위치한 루프를 캐쉬로 미리 읽은 후 루프 캐쉬로만 접근하여 루프를 수행하게 된다. 따라서 루프를 수행하는 동안 하위 계층에 위치한 캐쉬 접근을 줄여 명령어 캐쉬 전체의 소비 전력을 줄

인다. 하지만, 루프는 컴파일러에 의해 검색되며, 수행 시간에 새로운 명령어를 통해 루프 캐쉬로 미리 읽혀지게 되는 동작상 문제(Fill Problem)가 있다. 또한 'sbb' 명령어와 시작과 끝을 판별하기 위한 상태 장치들로 인하여 추가적인 하드웨어가 필요하고, 미리 읽는 과정에서 프로세서가 동작하지 않게 되어(CPU Stall) 성능 또한 감소하게 된다. 필터 캐쉬(Filter Cache)는 크기가 작은 직접 사상의 전통적인 구조의 캐쉬를 프로세서와 인접한 위치에 삽입하여, 상대적으로 소비전력이 큰 하위 계층 캐쉬로의 접근을 줄임으로서 전체적인 전력을 줄이는 방법이다. 루프 캐쉬에 비하여 캐쉬를 제외한 추가적인 하드웨어는 불필요 하나 필터 캐쉬의 적중률에 따라 성능이 감소하게 된다.

위 두 연구 모두 명령어 캐쉬의 소비 전력을 줄이는데 기여를 한 방법들이다. 특히 이 중 필터 캐쉬의 경우 전통적인 메모리 계층 구조를 응용한 방법으로 최근 몇몇 연구 논문에서는 프로세서에 가장 근접한 계층의 캐쉬로 L0 명령어 캐쉬라 명명하여 인용하기도 하는데, L0 명령어 캐쉬의 연관 사상이나 크기에 따라 소비 전력이 차이를 보인다[4].

3. 추상적 캐쉬 에너지 모델

C. Su와 A. Despain의 1995년 논문에서 캐쉬 소비 전력을 계산하기 위하여 추상적 캐쉬 에너지 모델(Abstract Cache Energy Model)과 간략한 식을 제시하였다[1].

$$E_{Cache} = E_{Decordingpath} + E_{Cellarray} + E_{I/Opath} \quad (1)$$

$$E_{Decodingpath} = \alpha \times Addr\_bus\_bs \quad (2)$$

$$E_{Cellarray} = \beta \times Word\_line\_size \times Bit\_line\_size \times Bit\_line\_bs(Static\ \log ic) \quad (3)$$

$$E_{Cellarray} = \beta \times Word\_line\_size \times Bit\_line\_size (Dynamic\ \log ic) \quad (4)$$

$$E_{I/Opath} = \gamma \times (Addr\_pad\_bs + Data\_pad\_bs) \quad (5)$$

식(2)에서 (5)의 α, β 및 γ는 반도체 제조 공정에 따른 상수 값이다. 식(1)은 캐쉬 에너지 소비에 대한 것이며, 크게 세 개의 항목으로 분류된다. 첫 번째 항목은 (2)식에 해당하며 상수 값 α와 명령당 평균적인 주소선의 비트 변화의 곱으로 산출 된다. 두 번째 항목은 태그 정보과 자료를 저장하는 메모리에 대한 에너지에 대한 식이며, 정적 혹은 동적 논리 회로인지 여부에 따라 식(3)과 (4)로 구분된다. 이 역시 입출력 선 크기와 접근 빈도에 의존한다. 마지막으로 세 번째 항목은 식(5)로 자료 입출력 경로에서 소비되는 전력을 나타내며, 이 또한 주소선과 자료 입출력경로에서의 비트 변화에 의존한다. 각각의 항목은 모두 데이터 입출력과 주소선의 비트 변화에 의존적임을 알 수 있으며, 이는 캐쉬 접근 빈도에 따라 증가한다는 점이다. 특히 두 번째 항목은 태그 정보 및 자료를 저장하는 메모리에 대한 소비 전력에 대한 것으로 접근 빈도 이외에 크기에도 의존적임을 알 수 있다.

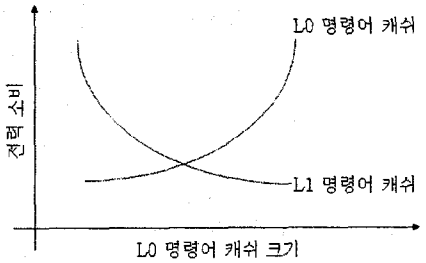

그림 2 L0 명령어 캐쉬 크기에 따른 L0, L1 소비 전력

4. 에너지 효율적 L0 명령어 캐쉬 크기

추상적 캐쉬 에너지 모델을 이중 캐쉬 계층 구조에 적용하여 생각해 볼 경우, 그림2에서 보는 바와 같다. L0 명령어 캐쉬에 있어서 크기에 상관없이 접근 횟수는 일정하므로, 전력은 크기에 의존한다. 그러나 L1 명령어 캐쉬의 경우 L0 명령어 캐쉬의 적중률이 높아짐에 따라 L1 명령어 캐쉬 접근 횟수가 줄어들게 되어, L1 명령어 캐쉬의 전력은 L0 명령어 캐쉬 적중률에 의존적이다. 또한 L1 명령어 캐쉬 크기가 커짐에 따라 증가하게 된다. 즉, L0 명령어 캐쉬 크기가 클수록 소비전력은 커지나 그에 반하여 적중률은 향상된다. 그에 따라 하위 계층인 L1 명령어 캐쉬로의 접근 빈도는 줄어들게 되며, 소비 전력 또한 감소하게 된다. 그러나, L0 명령어 캐쉬의 크기가 지나치게 클 경우 오히려 L1 명령어 캐쉬에서의 소비 전력보다 L0 명령어 캐쉬에서 소비 되는 전력이 급격히 증가하게 되어 전체적인 소비 전력은 증가하게 된다. 따라서 임의의 L1 명령어 캐쉬 크기에 대한 L0 명령어 캐쉬 크기 변화에 따른 전체 소비 전력은 두 캐쉬의 소비 전력 합으로 최소가 되는 지점이 존재하게 된다. 본 논문에서는 최적의 에너지 효율을 갖는 L0 명령어 캐쉬 크기를 찾는 실험을 통해 두 계층 간의 관계를 알아보고자 한다.

5. 소비 전력 실험 및 결과

본 장에서는 소비 전력을 측정하기 위한 방법을 제시하고 필요한 환경과 도구를 나열한다. 그 다음으로 L0, L1 명령어 캐쉬 구조를 변경하면서 소비 전력을 측정하고, 결과를 정리한다. 결과는 각 세부 항목을 살펴보고 마지막으로 프로세서 전체에 대해 살펴본다.

5.1 전력 측정 방법

다양한 크기와 구조를 갖는 캐쉬의 소비 전력을 산출하기 위해 모의실험을 선택했다. 모의실험 도구로는 인자 값으로 캐쉬를 가변적으로 설정할 수 있는 도구인 SimpleScalar/ARM power analyzer [6,7]을 사용하였다. SimpleScalar는 수행 기반 실험 도구(Execution-driven Simulator)로 PISA를 시작으로 현재 Alpha, ARM, PowerPC 및 x86등 다양한 아키텍쳐

를 지원하며 현재 네 번째 개정 판 까지 개발이 진행되고 있다. 또한, 전력 모델링 프로젝트(SimpleScalar Power Modeling Project[7])를 통해 전력 산출 기능을 지원한다. SimpleScalar/ARM은 기본적으로 이중 계층 캐쉬 구조를 지원하는데, 인자 값만으로 단일 계층 캐쉬 구조로 설정 할 수 없는 한계점이 있다. 따라서 단일 계층 캐쉬 구조를 실험하기 위해서는 소스 코드를 수정하여 L2 캐쉬 제거가 필요 하였다. 또한 변수 누락(Undefined Reference to 'errno')과 마이크로코드 버그(LFM-SFM Micro-Code Bug)의 문제점이 존재하여, SimpleScalar/ARM Patch[8]를 통해 문제를 해결하는 과정이 필요하다.

전력 산출을 위한 인자 값들로는 분석 도구 기본 값인 0.18um 기술을 가정하였으며, StrongARM과 동일한 단일 계층 캐쉬 구조를 기반으로 32바이트의 블록 크기와 LRU 교환 정책 및 완전 연관 구조로 구성하였다. 또한 실험을 위해 프로세서와 L1 명령어 캐쉬 사이에 직접 사상 구조로 L0 명령어 캐쉬에서 소비 되는 전력의 영향을 보다 줄이기 위하여[1] FIFO 교환 정책을 갖는 캐쉬를 삽입한 모의실험 도구를 준비하였다.

입력 자료로는 MiBench[9]를 사용하였다. MiBench는 임베디드 시스템을 위한 성능 평가 프로그램으로서 Automotive, Consumer, Network, Office, Security와 Telecom의 여섯 개의 항목으로 분류되어 있다. 또한 각각의 항목은 자주 사용하는 임베디드용 프로그램들로 총 21개로 구성되어 있다. MiBench 홈페이지에 미리 컴파일된 ARM용 이진 파일이 등록되어 있어 별도의 컴파일 과정은 생략하였다.

5.2 결과 및 분석

먼저 L1 명령어 캐쉬 크기를 16KB로 고정하여 L0 명령어 캐쉬가 없는 상태에서 L1 명령어 캐쉬에서 소비되는 전력을 산출한 뒤, L0 명령어 캐쉬를 추가하고 크기를 32B부터 4KB까지 변경하여 L0와 L1 명령어 캐쉬 소비 전력의 합을 앞서 산출한 L1 명령어 캐쉬만의 소비 전력과의 비율로 산출하였다. 산출한 결과에 대해 각 항목별로 평균을 내고, 전 항목의 평균을 정리하여 그림3과 같이 정리를 하였다.

그림3의 그래프에서 볼 수 있듯이 항목에 따라 에너지 효율이 가장 좋은 L0 명령어 캐쉬 크기에는 약간의 차이가 존재한다. L0 크기에 상관없이 대체로 Consumer 항목이 가장 좋은 효율을 보였다. 이는 각 항목을 구성하는 프로그램의 지역성 차이로 인한 것으로 작은 루프가 상대적으로 많은 Consumer 항목에서 높은 효율을 보이는 것으로 생각된다. 최적의 크기는 항목에 따라 약간의 차이가 존재하나 대체로 L0 크기가 512B인 지점을 중심으로 점차 감소하였다가 증가하는 모습을 보이고 있다. 또한 전체 평균을 보면, L0 명령어 캐쉬가 512B일 때 에너지 효율이 단일 계층 캐쉬

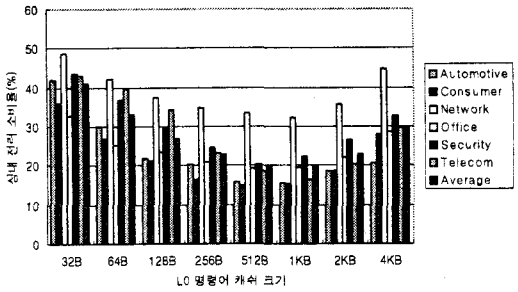

그림 3 L1 16KB일 때 L0 크기에 따른 상대적인 소비전력

대비 20%로 가장 좋았으며, L0 명령어 캐쉬가 작을수록 낮은 적중률로 인하여 L1 명령어 캐쉬에서 큰 소비 전력을 보였다. 또한 L0 명령어 캐쉬 크기가 1KB 을 넘어 가면서부터 L0 명령어 캐쉬에서 소비 되는 전력이 증가함에 따라 전체 명령어 캐쉬의 소비 전력이 증가하는 모습을 보였다.

L0와 L1 명령어 캐쉬 크기에 따른 상관관계를 알아보기 위해 L1 명령어 캐쉬를 8KB, 32KB 및 64KB로 변경하면서, 위 실험을 반복하여 모든 항목의 평균을 각 L1 크기별로 정리하여 그림4에 정리하였다. L1 명령어 캐쉬 크기를 변경할 때 마다 단일 계층으로 명령어 캐쉬가 소비하는 전력을 먼저 산출하여 그에 따른 감소율을 계산 하였다.

L1 명령어 캐쉬 크기가 8KB일 경우 에너지 효율이 가장 좋은 L0 명령어 캐쉬 크기는 512B 였으나, 크기를 고려하여 감소율에 큰 차이가 나지 않는 256B가 가장 좋은 선택이 될 수 있다. L1 크기가 두 배인 16KB 일 때, 에너지 효율이 좋은 L0 크기는 512B로 약 20%의 효율을 나타내었다. 마찬가지로 L1 크기를 32KB, 64KB로 변경하여 측정해본 결과 에너지 효율이 가장 좋은 크기는 각각 1KB와 2KB로 나타났으며, 에너지 효율은 16%, 12%로 더 증가함을 볼 수 있다.

이상의 결과를 살펴 볼 때 L1 명령어 캐쉬 크기에 따라 가장 좋은 에너지 효율을 나타내는 L0 캐쉬 크기가 각각 다름을 알 수 있었으며, L1 명령어 캐쉬 크기가 커질수록 L0를 삽입함으로서 나타나는 전력 감소 효과가 더 크다는 것을 볼 수 있다. 주목할 만 한 점은

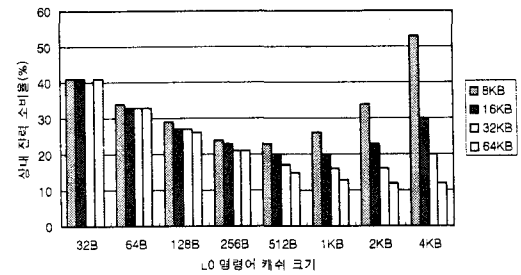

그림 4 L0, L1 크기에 따른 명령어 캐쉬 평균 소비전력 감소율

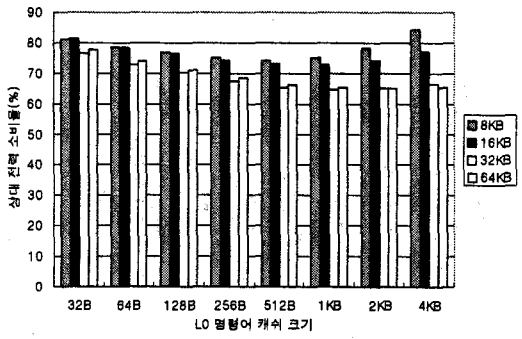

그림 5 L0, L1 크기에 따른 프로세서 평균 소비전력 감소율

대체로 L1 명령어 캐쉬가 두 배씩 증가함에 따라 에너지 효율이 좋은 L0 명령어 캐쉬의 크기 또한 두 배씩 커진다는 점이다. 이번 실험에서 L0와 L1 명령어 캐쉬 크기의 관계는 32배로 볼 수 있다. 만약 L1의 크기를 128KB로 설정 했을 경우 에너지 효율이 좋은 L0의 크기는 4KB로 예측해 볼 수 있을 것이다.

단일 계층에 L0 명령어 캐쉬를 추가함으로서 명령어 캐쉬 전체의 소비 전력을 알아 본 결과 최대 약 12%의 감소율을 보였다. L0 명령어 캐쉬를 추가함에 따라 온 칩 캐쉬 구조의 프로세서에서 소비되는 전력 감소율 변화를 알아보기 위해 프로세서의 소비 전력을 별도로 정리해 보았다. 그림5는 앞서 실험한 L0와 L1 크기에 따른 전력을 온 칩 구조로 가정하여 프로세서 전체에서 소비되는 전력을 측정한 것으로 각각은 단일 계층에서 명령어 캐쉬 크기를 기준으로 하여 상대적인 값으로 측정하였다. 마찬가지로 L1 명령어 캐쉬가 큰 경우 L0 명령어 캐쉬 추가로 인한 에너지 효율 증가폭이 더 컸으며, 각각의 L1 명령어 캐쉬 크기가 두 배씩 증가함에 따라 에너지 효율이 높은 L0 명령어 캐쉬 크기 역시 약 두 배씩 증가함을 볼 수 있다.

6. 결론

현재 StrongARM의 단일 계층 구조 보다 이중 계층 구조가 저 전력 관점에서 훨씬 더 이득이며, 본 논문의 실험에서 두 계층의 캐쉬 크기는 32배의 관계를 보이고 있다. 간단하게 L0 명령어 캐쉬 추가만으로도 15% 이상의 전력 절감 효과를 볼 수 있었으며, 최적의 크기를 찾아 적용하게 되면 약 25% 이상, 최대 35%에 이르는 전력을 줄일 수 있다.

본 논문에서는 L0 명령어 캐쉬의 구조를 소비 전력이 최소가 될 수 있는 형태로 선정 하였다. 그에 반하여, 크기에 따라 낮은 적중률로, 에너지 효율이 가장 높은 L0 크기가 다소 큰 편으로 측정되었다. 모의실험을 하는데 상당한 시간이 소요되어 보다 다양한 구조를 갖는 L0 명령어 캐쉬에 대한 실험을 하지 못한 관계로 연관 사상이나 블록 크기에 따른 두 캐쉬 계층의 관계를 알아보지 못하였다.

L0 및 L1 명령어 캐쉬 크기에 대한 관계를 찾아보았다. 이는 다시 말해 L0 명령어 캐쉬의 에너지 효율과 성능에 관여된 문제라 생각해 볼 수 있다. 상당히 작은 크기의 캐쉬로 하위 계층에 위치한 캐쉬나 메모리로의 접근을 줄이는 것이 중요한 논의이다. 따라서 앞으로 크기가 작은 캐쉬에 대해 높은 에너지 효율과 고성능에 대한 연구를 진행 하고자 한다.

참고 문헌

[1] C. Su and A. Despain, "Cache Design Tradeoffs for Power and Performance Optimization: A Case Study," Proc. of International Symposium on Low Power Design, 1995.

[2] J. Montanaro and e. al., "A 160MHz 32b 0.5W CMOS RISC Microprocessor," Proc. Of International Solid-state Circuits Conference, 1996.

[3] A. Gordon-Ross, S. Cotterell, and F. Vahid. "Tiny instruction caches for low power embedded systems." Trans. on Embedded Computing Sys. 2003.

[4] J. Kin, M. Gupta, and W. H. Mangione-Smith. "The filter cache : An energy efficient memory structure." In Proceedings of the 1997 International Symposium on Microarchitecture, pages 184.193, 1997.

[5] Lee. L.H., Moyer, B., and Arends, J. "Instruction fetch energy reduction using loop caches for embedded applications with small tight loops." International Symposium On Low Power Electronics and Design, 1999.

[6] SimpleScalar microarchitecture simulator. http://www.simplescalar.com

[7] SimpleScalar-ARM Power Modeling Project. http://www.eecs.umich.edu/~panalyer.

[8] SimpleScalar-ARM bug fix. http://www.jwhitham.org.uk/simplescalar/patches/

[9] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. "MiBench: A free, commercially representative embedded benchmark suite." IEEE 4th Annual Workshop on Workload Characterization, December 2001. http://www.eecs.umich.edu/mibench