Power Performance of Instruction Pre-Fetch Unit
Young-Kyu Song, Hyeong-Cheol Oh
Department of Information Engineering, Korea University
email: (simon.hyeong)@hard.korea.ac.kr

Abstract

In this paper, we investigate the effect of adopting branch-penalty compensation schemes on the power performance of TLBs (Translation Look-aside Buffers) and instruction caches. We found that the double-buffer branch-penalty compensation scheme can reduce the power consumption of the TLBs and the instruction caches considered by up to 14-21.3%. The power consumption is estimated through simulation at the architectural level, using the Kamble/Ghose method.

I. 서론

마이크로프로세서를 설계함에 있어서 전력 소모 문제는 시스템의 속도와 연쇄 이상으로 중요한 문제가 되고 있다. 더욱이 향후 시스템 개발 주제가 효율성 시스템의 유효성을 높이기 위한 연구인 양화의 점을 고려할 때, 전력 소모 문제는 더욱 중요하게 가지고 된다. 한편, 현재의 마이크로프로세서에서 전력 소비의 많은 부분을 차지하고 있는 소자중 하나가 메모리이므로 메모리 접근 횟수를 줄이는 것은 전력 소비 측면에서 매우 중요하다.

병령어 전 인출기는 이중 병령어 빈패를 사용하여 본기 병령어에 대해 선택된 경우에 그저 많은 경우에 해당되는 병령어를 모두 전 인출함으로써 메모리 접근 횟수를 줄이는데 효과적인 방법으로 알려져 있다. 본 논문에서는 이러한 분기 손실 보상 기법의 사용이 병령어 TLB와 병령어 캐퍼의 전력 소모에 미치는 영향을 다룬다.

전력 분석을 위해 사용된 TLB는 전역 사상(Fully-associative Mapping) 방식으로 설계를 하였으며, 병령어 캐퍼의 경우 직접 사상(Direct Mapping) 방식으로 설계를 하였다. 전역 사상 방식은 다양한 크기의 페이지를 지원하기가 쉽고 허용 바이드 직렬 사상 방식보다 성능적으로 좋다. 그러나 CAM(Content Addressable Memory)을 사용 하므로 메사이클들이 더 많이 발생한다고 가장하였을 경우 메번 CAM의 모든 엔트리를 다 검색해야 하기 때문에 전력 소비면에서 직접 사상 방식보다 많아지게 된다. 본 논문에서는 본기 예측 결과에 따른 손실 보상을 위한 이중 버퍼 기법[6]이 전역 사상 방식의 병령어 TLB와 직접 사상 방식의 병령어 캐퍼의 전력 소비에 미치는 영향을 구조 수준(Architectural Level)[1]에서 고찰 하였다.


II. 병령어 전 인출기의 구조

1. RAPTOR의 구조

RAPTOR[2]는 독립된 4개의 프로세서 유닛으로

RAFORT의 파이프라인 중 전 처리단계에 해당하는 명령어 선 인출기는 명령어 캐시, MMU 블록으로 구성되었다. 명령어 선 인출기는 수행 될 명령어를 MMU로 요청하여 명령어 캐시로부터 해당 명령어들을 받아 명령어 버퍼에 저장한다. 그림 1은 RAFORT에서 사용되는 명령어 선 인출기의 블록도이다.

![Diagram 1] 명령어 선 인출기의 블록도

2. MMU의 구조

프로세서에서 발생시키는 64 비트 가장 주소를 실제 주소로 변환 해 주는 MMU의 구조와 기능은 다음과 같다.

- 명령어 TLB는 64 엔트리의 값이 전역 상하 바anges식으로 구현하였다.
- 가장 주소를 특정 프로세서로부터 받아 실제 주소로 변환한다.
- 메모리 접근 제한 및 탐색을 통하여 메모리 영역 보호(Memory Protection)와 각 프로세서의 보안 유지 기능을 담당한다.
- 메모리 탐색의 효율을 높이기 위한 다양한 계층 페이지 데이터를 지원한다.
- 페이지 오류 발생시에 효과적으로 빠른 레퍼러

위크를 지원한다.

3. 명령어 버퍼의 구조

Raptor의 명령어 캐시 유닛에서는 그림 2와 같이 이중 명령어 버퍼를 이용하여 랜덤 펑싱 기법을 구현하였다.

두 개의 명령어 버퍼를 사용함으로써 한 개의 조건 본기 명령어에 대해서 예측된 경로와 그렇지 않은 경로에 두 명령어 버퍼를 할당하고, 각 경로에 해당하는 명령어들을 선 인출 해두므로써 본기 예측이 실패하였을 경우 예측되지 않았던 경로에 해당된 명령어들을 수행하고, 다음 명령어들을 요청함으로써 본기 손실을 줄일 수 있다. 한 개의 조건 본기 명령어에 대해서만 명령어 버퍼를 할당하여 처리하므로, 연속된 조건 본기 명령어의 경우 두 번째 명령어의 정보는 저장하였다가 첫 번째 본기 명령어가 처리된 후 그 결과에 따라 두 번째 본기 명령어를 수행하도록 한다. 명령어 버퍼의 수를 증가함으로써 처리 가능 조건 본기 명령어의 수를 늘리거나,램프 내의 처리 명령어 수를 추가시켜 기존 유 효 명령어의 수를 늘리므로써 본기 손실의 감소 효과가 컷 될 수 있다.

![Diagram 2] 이중 명령어 버퍼 사용한 본기 펑싱

명령어 캐시는 수행될 명령어들을 명령어 캐시로부터 프리복제하여 명령어 버퍼에 저장하여 두며, 명령어를 프리디코딩 한다. 프로그램 수행의 순서를 바꾸는 명령어들(예: 본기 명령어 등)을 구별하여 해당 타겟 명령어들을 프리복제하여 인속적인 명령어 수행이 이루어지게 한다. 두 개의 명령어 버퍼를 두어 순차적 명령어 불록인 경우 한 개의 명령어 버퍼를 사용하여 프리복제
치 해 두며, 본기 명령어의 타겟 주소에 해당되는 명령어는 또 다른 한 바위에 프리페치하여, 예측된 경로에 해당되는 명령어들을 수행하며 본 기 명령어의 처리 결과에 따라 예측이 틀렸을 경우 다른 비파의 명령어를 수행함으로써 본기 손실을 줄인다.

III. 전력 평가 방법 및 실험

1. 전력 평가 방법


CMOS 물질에서 소모되는 에너지는 다음 식(1)과 같이 부하 전적용량 소자들에 대한 중간량 전의에 대한 식으로 표현할 수 있다.

\[ E = 0.5 \cdot C \cdot V^2 \] (1)

여기서 \( C \) 는 부하 전적용량이고 \( V \) 는 소자에 공급되는 전압이다. 그리고 캐서 전체에 소모되는 전력 소비는 다음 식(2)과 같이 구할 수 있다.

\[ E_{\text{total}} = E_{P} + E_{Q} + E_{\text{extra}} + E_{\text{lost}} \] (2)

각각의 요소에 대한 설명은 다음과 같다.

- \( E_{P} \) (Energy Dissipated in Bit-Lines): 비트 라인(Bit-Line)에서 소모되는 에너지로서 프리페치, 일기/쓰기 동안에 소모되는 에너지이다.

\[ E_{P} = 0.5 \cdot V^2 \cdot \left[ N_{\text{write}} \cdot C_{\text{C0}} + N_{\text{read}} \cdot C_{\text{C1}} + N_{\text{delete}} \cdot C_{\text{C2}} + m \cdot (8 \cdot L + T \cdot S + 1) \cdot C_{\text{data}} + C_{\text{extra}} \cdot C_{\text{extra}} \right] \] (3)

- \( E_{\text{Q}} \) (Energy Dissipated in Word-Lines): Word-Line 에서 소모되는 에너지로서 하나의 열이 선택될 때 필요한 에너지이다.

\[ E_{\text{Q}} = V^2 \cdot C_{\text{row}} \cdot (L \cdot 8 + T \cdot S + 1) \cdot (2 \cdot C_{\text{Q}} + C_{\text{extra}}) \] (4)

- \( E_{\text{extra}} \) (Energy Dissipated Driving External Buses): 데이터를 비스에 송/수신 할 때마다 미스 시 주소를 하위 메모리로 보낼 때 소모되는 에너지이다.

\[ E_{\text{extra}} = E_{\text{extra}} + E_{\text{extra}} + E_{\text{extra}} + E_{\text{extra}} \] (5)

\[ E_{\text{extra}} = 0.5 \cdot V^2 \cdot \left[ N_{\text{write}} \cdot C_{\text{extra}} + N_{\text{delete}} \cdot C_{\text{extra}} \right] \] (6)

\[ E_{\text{extra}} = 0.5 \cdot V^2 \cdot \left[ N_{\text{write}} \cdot C_{\text{extra}} + N_{\text{delete}} \cdot C_{\text{extra}} \right] \] (7)

- \( E_{\text{lost}} \) (Energy dissipated in the address input lines):

\[ E_{\text{lost}} = 0.5 \cdot V^2 \cdot N_{\text{map}} \cdot l(m+1) \cdot 2 \cdot C_{\text{extra}} + C_{\text{extra}} \] (8)

여기서 TLB의 경우 전력 사상 방식의 구조이므로 주소 입력 라인에서 소모되는 에너지 \( E_{\text{lost}} \) 는 없게 됐다. 사용된 문자와 의미는 다음과 같다.

- \( m \): 연산도
- \( L \): 라인의 크기
- \( T \): 테그의 크기
- \( D \): 전체 용량
- \( CA \): 총 캐서 접근 횟수
- \( St \): 불록 당 대책 비트의 수
- \( C_{\text{extra}} \): 부하 전적용량
- \( N_{\text{map}} \): 전적 횟수 (Number of Transition)


2. 실험 결과

단일 비파 기법과 본 논문에서 제안하는 이중 비파 기법을 사용하여 전력 성능에 대한 실험은 TLB 엔트리의 크기에 따른 전력 성능 비교와 병 분배의 개체에 대한 전력 성능 비교를 수행하고 있다.

전력 분석을 위해 사용된 명령어 TLB의 병 분 배 개체 그리고 명령어 비파의 사상은 다음과 같다.

- 32, 64, 128 엔트리를 갖는 병 분배(Fully-Associative) 방식의 명령어 TLB
- 32bytes 까지 단 하나 크기의 16Kbytes 의 직접 사상(Direct Mapping) 방식의 명령어 개체
- 64 비파의 엔트리를 갖는 병 분배 비파

현재 SPEC CPU95 벤치마크 프로그램 중 데이터베이스 프로그램인 Vortex 와 C 언어로 작성된 작은 규모의 응용 프로그램인 Quicksort을 대상으로 실험을 수행하였으며, Quicksort 프로그램은
2회 이상 반복 수행되었다. 그 실험 결과는 아래 표 1과 표 2에 보였다.

<table>
<thead>
<tr>
<th>Single Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>벤치마크</td>
</tr>
<tr>
<td>Vortex</td>
</tr>
<tr>
<td>Quicksort</td>
</tr>
</tbody>
</table>

(a) Single Buffer

<table>
<thead>
<tr>
<th>Double Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>벤치마크</td>
</tr>
<tr>
<td>Vortex</td>
</tr>
<tr>
<td>Quicksort</td>
</tr>
</tbody>
</table>

(b) Double Buffer

표 1) 벤치마크의 빈도성능

[표 2] 벤치마크 캐시의 빈도성능

표에서 볼 수 있는 바와 같이, TBL의 정확도는 다음과 테스트: 벤치마크의 단일 벤치마크 기법 브로더 Vortex의 경우 15.3%, Quicksort는 21.3%의 정확도를 제공하였고 Mလ래의 경우도 Vortex는 7%, Quicksort는 10%의 정확도를 제공하였다. 이는 TBL과 캐시의 전체적인 성능비교는 Vortex의 경우 6%, Quicksort의 경우 14%의 정확도를 보였다.

실험 결과에 대한 명령어의 벤치마크의 경우 32 엔트리를 갖는 TBL의 전체 성능은 이중 명령어 벤치마크의 전체 성능과 비교했을 때 거의 차이가 없으며 TBL의 엔트리 수가 증가함수록 성능이 더 많이 상향선을 준다. 이는 TBL의 참조 사양들의 실제 실험결과에 따른 성능이 캐시에서 주소를 발행시킬 메인메모리의 모든 엔트리를 다 접근해야 하므로 캐시 수가 증가 할수록 CAM의 비트라인(Bit Line)에서 소모되는 전력이 증가하게 된다.

V. 결론 및 향후 계획

벤치마크 TB의 명령어의 캐시에 대한 이중 명령어 벤치마크의 사례로는 비교적 흔히 캐시 기법의 전체 성능의 비교 분석 결과에 먼저 명령어 벤치마크의 전체적으로 약 10%의 전력 소모를 절약할 수가 있다. 전력 성능 면이 외에도 수행 시간 비교 결과 약 15%정도의 성능 향상이 있다. 이와 같은 결과는 본기 예측에서 향후의 미리 인증해 놓은 다른 경우의 명령어를 수행함으로써 단일 캐시에서 발생할 수 있는 본기 예측 결과에 따른 손실을 줄일 수가 있기 때문이다.

본 실험에서 사용한 Kambale/Hash 모델의 경력 성능 모델의 빈도와 점유는 2%내외로 알려져있기 때문에[3], 캐시 매개 변수의 경우 TBL과 캐시의 전체적인 성능비교는 Vortex의 경우 6%, Quicksort의 경우 14%의 정확도를 보였다.

만약 실험 결과에 대한 명령어의 벤치마크의 경우 32 엔트리를 갖는 TBL의 전체 성능은 이중 명령어 벤치마크의 전체 성능과 비교했을 때 거의 차이가 없으며 TBL의 엔트리 수가 증가함수록 성능이 더 많이 상향선을 준다. 이는 TBL의 참조 사양들의 실제 실험결과에 따른 성능이 캐시에서 주소를 발행시킬 메인메모리의 모든 엔트리를 다 접근해야 하므로 캐시 수가 증가 할수록 CAM의 비트라인(Bit Line)에서 소모되는 전력이 증가하게 된다.

참고 문헌