수퍼스칼라 프로세서에서 명령어 패치의 해석적 모델 및 성능분석

김선모 정진하 최상방
인 하 대 학 교 전 자 공 학 과
sangbang@dragon.inha.ac.kr

Analytical Models of Instruction Fetch and Performance Analyses on Superscalar Processors

Sun-Mo Kim Sang-Bang Choi
Dept. of Electronic Engineering, Inha University

요 약

최근에 캐쉬의 성능이 전체 시스템에 미치는 영향이 커짐에 따라 캐쉬의 성능을 모델링하고 향상시키기 위한 많은 연구가 진행되고 있다. 본 논문에서는 네 가지 종류의 캐쉬모델을 가정하고 분기명령어 비율, 캐쉬미스율, 분기예측 실패율 등의 파라메터를 이용하여 수퍼스칼라 프로세서에서의 명령어 패치율을 해석적으로 모델링하였다. 시뮬레이션 결과 분기예측실패가 명령어 패치율에 미치는 영향보다는 캐쉬미스율이나 캐쉬미스 패널티의 증가로 인한 패치율의 감소가 더욱 큰 폭으로 나타났다.

1. 서론

수퍼스칼라 프로세서의 목적은 사이클 당 여러 개의 명령어를 동시에 수행시키는 것이며 이를 달성하기 위해서 프로그램 내의 병렬성을 이용한다[3]. 그러나, 명령어들이 충분한 비율로 메모리로부터 해석단에 전달되지 않는다면 병렬성은 이용될 수 없다. 또한, 프로세서와 메모리간의 속도차이가 커지면서 캐쉬의 성능이 전체 시스템에 미치는 영향이 커졌으며 이에 따라 캐쉬의 성능을 모델링하고 향상시키기 위한 연구가 활발히 진행되고 있다.

최근에 Wallece에 의해 명령어 패치율을 해석적으로 모델링하는 방법이 제안되었으나 패치율에 가장 큰 영향을 미치는 캐쉬미스나 분기예측실패 등이 고려되지 않았다[2]. 본 논문에서는 네 가지 종류의 캐쉬모델을 가정하고 분기명령어 비율, 캐쉬미스율, 분기예측 실패율등의 파라메터를 이용하여 수퍼스칼라 프로세서에서의 명령어 패치율을 해석적으로 모델링하였다.

2. 명령어 패치율의 해석적 모델링

2.1 수퍼스칼라 프로세서의 명령어 패치과정

그림 1은 수퍼스칼라 프로세서에서의 명령어 패치 과정을 나타낸 것이다. 프로세서에서 필요로 하는 명령어가 명령어 캐쉬로부터 해석단까지 전달되는 일련의 과정을 명령어 패치라고 하며 이 과정은 명령어 캐쉬, 명령어 인출기, 명령어 해석기 등의 유니트가 유기적으로 결합되어 이루어진다.

캐쉬블록의 크기가 n이고, 명령어가 분기명령어일 확률을 b라 할 때, 블록내의 연속적인 명령어들의 평균길이는 다음과 같이 구할 수 있다.

$$L(n, b) = n(1-b)^n + \sum_{i=1}^{n} i(1-b)^{i-1}b = \frac{1-(1-b)^n}{b} \quad (1)$$

이 때 캐쉬블록의 크기를 늘려서 n을 무한으로 한다면 이는 명령어 인출기가 한 사이클에 패치할 수 있는 최대 명령어의 길이를 나타내며 식 (2)와 같이 제한된다.

$$\lim_{n \to \infty} L(n, b) = \frac{1}{b} \quad (2)$$

· 본 연구는 한국과학재단 핵심전문 연구비(981-0924-125-2) 지원으로 수행되었으며 지원에 감사드립니다.

2.2 Simple 캐쉬

Simple 캐쉬는 그림 2와 같이 패치블록의 크기와 캐쉬의 라인의 크기가 같은 경우이다. 명령어 인출기는 캐쉬의 한 라인 8전체를 검색하여 명령어 해석단으로 전달할 수 있다. 하지만, 캐쉬라인 내에 분기명령어가 존재한다면, 이후의 명령어들은 무효화시키고 분기명령어까지만 해석단으로 전달한다.

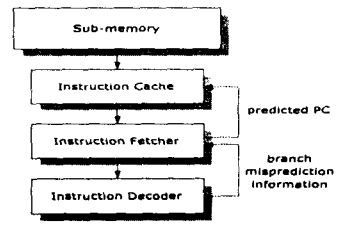


그림 1 수퍼스칼라 프로세서의 명령어 패치 과정

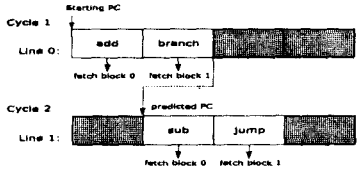


그림 2 Simple 캐쉬

캐쉬블록의 크기가 n이고 분기명령어의 확률이 b일 때 패치 시작점이 i번째 명령어일 확률 $S_i(n, b)$는 다음과 같다.

$$S_1(n, b) = 1 - \frac{n-1}{n} C(n, b)$$

$$S_i(n, b) = \frac{C(n, b)}{n}, \quad 2 \le i \le n \quad (3)$$

$C(n, b)$는 블록 전체에서 분기가 발생할 확률을 나타내며 블록내의 i번째 명령어에서 분기가 발생할 확률을 $c_i(n, b)$라 하면 다음과 같이 계산할 수 있다.

$$c_i(n, b) = \sum_{j=1}^{i} b(1-b)^{i-j} \cdot S_j(n, b) \quad (4)$$

따라서, $C(n, b)$는 블록내의 각 위치 i에서 분기가 발생할 확률의 합과 같다.

$$C(n, b) = \sum_{i=1}^{n} c_i(n, b) = \frac{n}{\frac{1}{b} + n - 1} \quad (5)$$

Simple 캐쉬에서 명령어 패치율은 패치시작점과 그 이후에 위치한 연속적인 명령어의 길이에 따라 결정되므로 패치율을 구하면 다음과 같다.

$$F_{simple}(n, b) = \sum_{i=1}^{n} S_i(n, b) \cdot L(n-i+1, b) = \frac{n}{1+b(n-1)} \quad (6)$$

2.3 Extended 캐쉬

그림 3과 같이 extended 캐쉬는 캐쉬라인의 크기(m)를 패치 블록의 크기(n)보다 크게 하고($m>n$) save 버퍼를 두어 최대 m개의 명령어를 캐쉬로부터 검색하고 마지막 $n-1$개의 명령어를 버퍼에 저장할 수 있다. 다음 사이클에서는 버퍼에 저장되어 있던 명령어와 새로 캐쉬로부터 검색된 명령어를 결합(combine)하여 명령어 해석기로 전달한다.

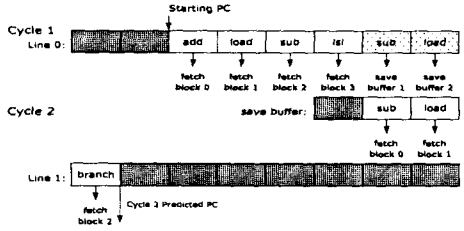


그림 3 Extended 캐쉬

명령어의 패치율은 다음과 같이 구할 수 있다.

$$F_{extended}(n, b, m) = \frac{m-n}{m} L(n, b) + \frac{n}{m} F_{simple}(n, b) = \frac{(m-n)}{m} \frac{(1-(1-b)^n)}{b} + \frac{n}{m} \frac{n}{(1+b(n-1))} \quad (7)$$

2.4 Prefetch 캐쉬

본 논문에서는 명령어 인출기 안에 prefetch 버퍼를 두어 분기예측기가 이를 참조하여 다음 사이클의 패치시작점을 결정하는 방식을 모델링하였다. 그림 4에서 보듯이 캐쉬 라인의 마지막 n번째 명령어 내에도 한 개의 분기명령어까지는 prefetch 버퍼 내를 참조하여 분기예측을 할 수 있으므로 명령어의 패치율은 다음과 같이 구할 수 있다.

$$F_{prefetch}(n, b, m) = \frac{m-n}{m} L(n, b) + \frac{n}{m} L(n, b) = L(n, b) = \frac{1-(1-b)^n}{b} \quad (8)$$

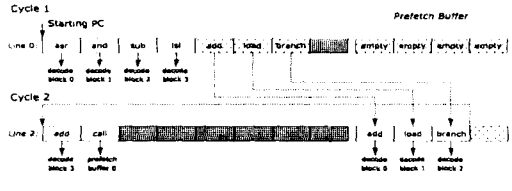


그림 4 Prefetch 캐쉬

2.5 Interleaved 캐쉬

그림 5와 같이 캐쉬에 메모리 뱅크의 개념을 도입한 것이 interleaved 캐쉬이다[1]. 명령어 인출기는 다중분기 예측기를 사용하여 현재의 PC로 다음 두 번째의 명령어까지를 예측한다. 이후에 각각 독립적인 두 개의 캐쉬뱅크를 검색하여 명령어를 패치한다.

그림 5 Interleaved 캐쉬

각각의 캐쉬뱅크를 독립적인 simple 캐쉬로 본다면 명령어 패치율은 다음과 같다.

$$F_{interleaved}(n, b) = 2F_{simple}(n, b) \quad (\le n) \quad (9)$$

3. 캐쉬미스, 분기예측실패를 고려한 모델링

3.1 Simple 캐쉬와 extended 캐쉬의 경우

전 장에서 정리한 명령어 패치율은 인출단에서 요구한 모든 명령어가 캐쉬에 존재하며 분기예측도 100% 정확하다는 가정하에 구한 값이다. 그러나, 캐쉬미스와 분기예측실패를 고려한다면 이로 인해 캐쉬가 서비스를 하지 못하는 사이클이 발생하게 되며 이를 패치율의 계산에 적용하여야 한다. 앞으로의 모델링을 위해서 B는 분기명령어의 빈도, R_C는 캐쉬미스율, R_B는 분기예측실패율, P_C는 캐쉬미스 패널티, P_B는 분기예측실패 패널티, 그리고 C_i는 case(i)에 대하여 명령어당 추가로 소요되는 사이클을 각각 나타내는 변수로 정의한다.

표 1 패치과정에 따른 고려사항

연속적인 명령어 접근		분기명령에 의한 비연속적인 명령어 접근					
		캐쉬히트			캐쉬미스		
캐쉬 히트	캐쉬 미스	분기 예측 성공	분기예측실패 캐쉬 히트	분기예측실패 캐쉬 미스	분기 예측 성공	분기예측실패 캐쉬 히트	분기예측실패 캐쉬 미스
-	P_C 고려	-	P_B 고려	P_C, P_B 고려	P_C 고려	P_C, P_B 고려	P_C, P_B 고려
case(1)	case(2)	case(3)	case(4)	case(5)	case(6)	case(7)	case(8)

표 1은 패치과정에 따라 추가적으로 소요되는 사이클을 계산하기 위해 고려해야 할 사항들을 캐쉬접근 방식과 캐쉬미스발생여부, 그리고 분기예측성공여부에 따라 명령어의 종류를 구분한 것이다.

예를 들어 case(2)에 해당하는 명령어의 경우를 보면, 전 사이클에서 분기하지 않은 연속적인 명령어의 흐름이지만 요구된 명령어가 캐쉬에 존재하지 않는 경우이므로 메모리로부터 명령어가 읽혀지는 동안 캐쉬는 더 이상 서비스를 하지 못하고 기다려야 한다. 따라서, 이 경우에 해당되는 명령어들에 대하여 평균적으로 추가 소요되는 사이클의 계산은 다음과 같다.

$$C_2 = (1-B) \cdot R_C \cdot P_C \quad (10)$$

같은 방법으로 각각의 case(i)에 대해 추가적으로 소요되는 사이클은 다음과 같이 계산할 수 있다.

$$C_4 = B \cdot (1-R_C)^2 \cdot R_B \cdot P_B$$

$$C_5 = B \cdot (1-R_C) \cdot R_B \cdot R_C \cdot (P_C + P_B)$$

$$\cdots$$

$$C_{total} = \sum_{i=1}^{8} C_i = C_2 + C_4 + C_5 + C_6 + C_7 + C_8 \quad (11)$$

식 (11)은 전체 명령어 수행 시 캐쉬미스와 분기예측실패로 인해 추가적으로 소요되는 총 사이클의 합이므로, 이를 전장의 명령어 패치율에 적용한다면 식 (6)과 (7)은 다음과 같이 수정할 수 있다.

$$F_{modified} = \frac{F_{original}}{1 + C_{total}} \quad (12)$$

3.2 Prefetch 캐쉬의 경우

Prefetch 캐쉬에서는 표 1의 각 case에 따라 다른 방식의 패치율을 적용하여야 한다. Case(1)과 case(3)에 해당되는 명령어의 경우는 이미 정의한 식 (8)과 같이 prefetch 캐쉬의 패치율을 그대로 적용할 수 있다.

$$\begin{aligned} F_{case(1)+case(3)} &= [(1-B)\cdot(1-R_C) + B\cdot(1-R_C) \\ &\quad \cdot(1-R_B)]\cdot F_{prefetch} \\ &= W\cdot F_{prefetch} \end{aligned} \quad (13)$$

그러나 나머지 경우 즉, 분기예측이 실패한 경우나 캐쉬미스가 발생한 경우는 prefetching의 기법을 사용하더라도 서브메모리로부터 명령어가 서비스되는 동안에는 prefetch 버퍼 내에 있는 명령어만이 해석기로 전달되므로 이는 extended 캐쉬의 save 버퍼를 적용할 때와 같은 방식으로 명령어가 패치된다.

$$F_{case(2)+\sum_{i=4}^{13} case(i)} = (1-W)\cdot\frac{F_{extended}}{1+C_{total}} \quad (14)$$

따라서, 모든 경우를 고려하여 패치율을 계산하면 다음과 같다.

$$F_{modified} = W\cdot F_{prefetch} + (1-W)\cdot\frac{F_{extended}}{1+C_{total}} \quad (15)$$

3.3 Interleaved 캐쉬의 경우

표 2는 표 1과 같은 방식으로 interleaved 캐쉬에 대한 명령어의 구분을 나타낸 것이다. Case (8)·(10)·(13)의 경우 더욱 세분하게 구분하여야 하지만, 더 이상의 구분은 의미가 없다고 가정한다.

Interleaved 캐쉬에서 뱅크의 개념을 도입하였다 하더라도 경우에 따라 단순히 simple 캐쉬의 패치율을 적용해야 하는 경우가 있다. 예를 들어, case (2)에 해당하는 명령어는 두 뱅크에서 모두 히트가 발생하지만 뱅크 1의 분기예측이 실패하여 뱅크 0에서만 명령어가 인출되는 경우이다. 이런 경우를 고려하여 크게 네 가지 그룹으로 분류하면 다음과 같다. 가중치 W의 첨자는 해당하는 case를 나타낸다.

$$\begin{aligned} W_{2,3,7,9} &= W_1 = (1-B)(1-R_C)^2R_B + (1-B)(1-R_C)R_C \\ &\quad + B(1-R_C)^2(1-R_B)R_B + B(1-R_C)R_C(1-R_B) \\ W_{5,12} &= W_2 = (1-B)R_C\cdot R_B + B\cdot R_C(1-R_B)R_B \\ W_{1,6} &= W_3 = (1-B)(1-R_C)^2(1-R_B) + B(1-R_C)^2(1-R_B)^2 \\ W_{4,8,10,11,13} &= W_4 = 1 - \sum_{i=1}^{3} W_i \end{aligned} \quad (16)$$

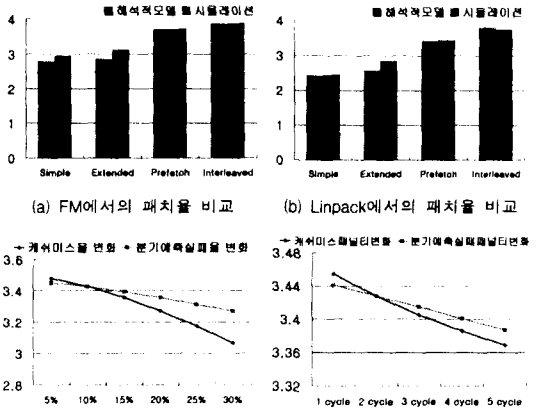
W_1은 simple 캐쉬의 패치율을 그대로 적용하는 경우, W_2는 이에 추가적으로 소요되는 사이클을 고려하는 경우, W_3은 interleaved 캐쉬 패치율을 적용하는 경우, 그리고 W_4는 이에 추가적으로 소요되는 사이클을 고려하는 경우이다. 기존의 패치율을 수정하면 다음과 같다.

$$\begin{aligned} F_{modified} &= W_1\cdot F_{simple} + W_2\cdot\frac{F_{simple}}{1+C_5+C_{12}} + W_3\cdot F_{interleaved} \\ &\quad + W_4\cdot\frac{F_{interleaved}}{1+C_4+C_8+C_{10}+C_{11}+C_{13}} \end{aligned} \quad (17)$$

4. 시뮬레이션 및 분석

본 논문에서 제안한 해석적 모델을 시뮬레이션하기 위해 필요한 트레이스 파일은 Sun SPARC-20 워크스테이션 기종에서 GNU C/C++의 표준옵션(-O0)을 적용하여 컴파일 한 후, SPA [4] 프로그램으로 생성하였다. 캐쉬 모델은 64K의 크기를 가지며 직접매핑 방식(direct-mapped)의 n=4인 모델을 가정하였다.

그림 6의 (a)와 (b)는 FM(b=3.40%)과 Linpack(b=8.64%) 벤치마크 프로그램들에 대한 결과이다. y축은 명령어 패치율을 의미하며 사용된 파라메터는 R_C=10%, R_B=5%, P_C=3cycles, 그리고 P_B=1cycle로 각각 가정되었다. 해석적 모델과 시뮬레이션의 결과를 비교해 볼 때 거의 일치하고 있다. (c)는 Linpack 벤치마크 프로그램에서 P_C와 P_B를 고정하고 명령어 패치율에 미치는 캐쉬미스율, 분기예측실패율의 영향을 분석한 것이며 (d)는 R_C와 R_B를 고정하고 캐쉬미스 패널티, 분기예측실패 패널티의 영향을 분석한 것이다. 분기예측실패가 명령어 패치율에 미치는 영향보다는 캐쉬미스율이나 캐쉬미스 패널티의 증가로 인한 패치율의 감소가 더욱 큰 폭으로 나타났다.



(a) FM에서의 패치율 비교 (b) Linpack에서의 패치율 비교

(c) 캐쉬미스율, 분기예측 실패율에 따른 패치율변화 (d) 캐쉬미스 패널티, 분기예측 실패 패널티에 따른 패치율변화

그림 6 시뮬레이션 결과

참 고 문 헌

[1] T.M. Conte, K.N. Menezes, P.M. Mills, and B.A. Patel, "Optimization of Instruction Fetch Mechanisms for High Issue Rates", Proceedings of the 22nd Annual International Symposium on Computer Architecture, Jun. 1995.

[2] S. Wallace, and N. Bagherzadeh, "Modeled and Measured Instruction Fetching Performance for Super- scalar Microprocessors", IEEE Trans. on Parallel and Distributed Systems, June 1998.

[3] M. Johnson. Superscalar Microprocessor Design. Englewood Cliffs, N.J: Prentice Hall, 1991.

[4] G. Irlam, Spa Personal Communication 1995.

표 2 Interleaved 캐쉬에서 패치과정에 따른 고려사항

연속적인 명령어 접근					분기명령에 의한 비연속적인 명령어 접근							
두 뱅크 모두 히트		뱅크1 미스	뱅크0 미스		두 뱅크 모두 히트			뱅크1 미스		뱅크0 미스		
뱅크1 분기예측성공	뱅크1 분기예측실패	-	뱅크1 분기예측성공	뱅크1 분기예측실패	두 뱅크 분기예측성공	뱅크1 분기예측실패	뱅크0 분기예측실패	뱅크0 분기예측성공	뱅크0 분기예측실패	두 뱅크 분기예측성공	뱅크1 분기예측실패	뱅크0 분기예측실패
-	-		P_C 고려	P_C 고려	-	-	P_C, P_B 고려	-	P_C, P_B 고려	P_C 고려	P_C 고려	P_C, P_B 고려
case(1)	case(2)	case(3)	case(4)	case(5)	case(6)	case(7)	case(8)	case(9)	case(10)	case(11)	case(12)	case(13)