슈퍼스칼라 프로세서에서 정적 및 동적 분류를 사용한 혼합형 결과 값 예측기

(A Hybrid Value Predictor using Static and Dynamic Classification in Superscalar Processors)

김 주 익 † 박 홍 준 †† 조 영 일 †††
(Ju-Ick Kim) (Hong-Joon Park) (Young-Il Cho)

요 약 데이타 종속성은 명령어 수준 병렬성을 향상시키는데 중요한 장애요소가 되고 있으며, 최근 여러 논문에서 데이타 종속을 제거하기 위하여 결과 값을 예상하는 방법이 연구되고 있다. 혼합형 결과 값 예측기는 여러 예측기의 장점을 이용하여 높은 예상 정확도를 얻을 수 있지만, 동일한 명령어가 여러 개의 예측기 테이블에 중복 엔트리를 갖게되어 높은 하드웨어의 비용을 필요로 한다는 단점이 있다.

본 논문에서는 정적 및 동적 분류 정보를 이용하여 높은 성능을 얻을 수 있는 새로운 혼합형 결과 값 예측기를 제안한다. 제안된 예측기는 반입 단계 동안 정적 분류 정보를 사용하여 적절한 예측기에 할당함으로써 테이블 크기를 효과적으로 감소시켰고 예상정확도를 향상시켰다. 또한 제안된 예측기는 동적 분류를 사용하여 "Unknown" 유형의 명령어에 가장 적절한 예측방법을 선택하도록 하여 예상 정확도를 더욱 향상시켰다.

SimpleScalar/PISA 툴셋과 SPECint95 벤치마크 프로그램에서 시뮬레이션 한 결과, 정적 분류 정보를 사용하였을 경우 평균 예상 정확도가 85.1%, 정적 및 동적 분류 정보를 모두 사용하였을 경우 87.6%의 평균 예상 정확도를 얻을 수 있었다.

키워드 : 슈퍼스칼라 프로세서, 데이타 종속성, 명령어 수준 병렬성, 동적분류, 정적분류, 혼합형 결과 값 예측기

Abstract Data dependencies are one of major hurdles to limit ILP(Instruction Level Parallelism), so several related works have suggested that the limit imposed by data dependencies can be overcome to some extent with use of the data value prediction. Hybrid value predictor can obtain the high prediction accuracy using advantages of various predictors, but it has a defect that same instruction has overlapping entries in all predictor.

In this paper, we propose a new hybrid value predictor which achieves high performance by using the information of static and dynamic classification. The proposed predictor can enhance the prediction accuracy and efficiently decrease the prediction table size of predictor, because it allocates each instruction into single best-suited predictor during the fetch stage by using the information of static classification. Also, it can enhance the prediction accuracy because it selects a best-suited prediction method for the "Unknown" pattern instructions by using the dynamic classification mechanism.

Simulation results based on the SimpleScalar/PISA tool set and the SPECint95 benchmarks show the average correct prediction rate of 85.1% by using the static classification mechanism. Also, we achieve the average correction prediction rate of 87.6% by using static and dynamic classification mechanism.

Key words : Superscalar Processor, Data Dependency, Instruction-Level Parallelism, Dynamic Classification, Static Classification, Hybrid Value Prediction

† 비 회 원 : 수원대학교 컴퓨터학과
mkby12@hanmail.net
†† 비 회 원 : 극동정보대학 전산정보처리과 교수
juny923@nownuri.net
††† 정 회 원 : 수원대학교 정보공학대학 컴퓨터학과 교수
yicho@mail.suwon.ac.kr
논문접수 : 2003년 2월 26일
심사완료 : 2003년 7월 29일

1. 서 론

최근 명령어 수준 병렬성(ILP : Instruction Level Parallelism)을 향상시키기 위해 실 데이타 종속(True Data Dependency)에 의한 장애를 제거하는 결과 값 예

상기법에 대한 연구가 진행되고 있다[1-12]. 결과 값 예상 기법은 데이타 종속적인 명령어가 선행 명령어의 결과 값이 나올 때까지 기다리는 것이 아니라 선행 명령어의 결과를 미리 예상하여 모험적으로 수행함으로써 데이타 종속 관계를 제거하는 하드웨어 방법이다.

결과 값 예상 기법 중 최근 결과 값 예측기(LVP : Last Value Predictor)는 프로그램 수행동안 결과 값이 일정한 값을 유지한다는 사실을 이용하여 바로 이전에 수행된 결과로 예상하는 방법이다[1]. 스트라이드 결과 값 예측기(SVP : Stride Value Predictor)는 명령어의 결과 값이 일정한 값만큼 변한다는 사실을 이용하여 다음 결과를 예상하는 방법이다[2,3]. 2-단계 결과 값 예측기(TVP : Two-level Value Predictor)는 프로그램 수행동안 60% 이상의 명령어가 4개의 값 범위 내에서 사용한다는 사실을 이용하여 이전 수행된 4개의 결과 값 중 하나를 다음 값으로 예상하는 방법이다[3]. 기존의 결과 값 예측기는 다양한 특성을 갖는 모든 명령어를 예상할 수 없기 때문에 예상 정확도가 낮다는 단점이 있다. 이러한 단점을 극복하기 위해 여러 예측기를 혼합해서 사용하는 혼합형 결과 값 예측기(HVP : Hybrid Value Predictor)가 제안되었다[3,4,5]. 혼합형 결과 값 예측기는 예상정확도는 높지만 하나의 명령어가 여러 개의 예측기의 엔트리에 중복 할당됨으로써 하드웨어 비용이 높다는 단점이 있다. 하드웨어 비용을 줄이기 위해서는 가장 적합한 예측기에만 명령어를 할당하는 분류 방법이 필요하다. 분류 방법에는 컴파일 시간에 이루어지는 정적 분류방법[6]과 수행시간에 이루어지는 동적 분류 방법[7]이 있다.

본 논문에서는 기존 연구된 여러 개의 예측기들을 하나로 결합하고, 정적 및 동적으로 결과 값들의 실행 유형을 분류하여, 임의의 명령어에 가장 적합한 예측기에 할당하여 예상하는 분류 능력을 갖는 새로운 혼합형 결과 값 예측기를 제안한다.

제안된 혼합형 결과 값 예측기는 컴파일 시간에 프로파일링(Profiling)으로 얻어진 정적 분류 정보를 이용하여 명령어 반입 시 명령어를 적절한 예측기에 할당하고, 예상이 어려운 명령어들은 실행 시 동적으로 분류하여 가장 적절한 예측 방법을 선택함으로써 예상 정확도를 향상시킬 수 있다. 또한 동적 분류를 별도의 분류 테이블 없이 수정된 2-단계 결과 값 예측기를 사용하여 하드웨어 비용을 효율적으로 감소시킬 수 있다.

2. 관련 연구

2.1 결과 값 예측기(Value Predictor)

최근 결과 값 예측기는 프로그램의 수행동안 명령어의 결과 값의 변화가 적음을 이용한 방법이다. 명령어가 가장 최근에 생성한 결과 값을 예상 테이블에 저장하여, 다음에 동일 명령어를 반입할 때, 바로 이전의 수행 시 저장된 결과 값을 예상 값으로 사용하는 방법이다. 적은 하드웨어를 사용하고 구현이 쉽지만, 결과 값이 수행 중 변하지 않는 값을 생성하는 명령어만을 예상할 수 있다는 단점이 있다.

스트라이드 결과 값 예측기는 결과 값들이 일정한 간격으로 변한다는 사실을 이용한 방법이다. 마지막으로 수행된 명령어의 결과 값과 마지막 두 번의 수행된 결과 값의 차이 값(stride)을 저장하여 이후 동일 명령어의 반입 시 마지막 수행 결과 값과 차이 값의 합으로 결과 값을 예상한다. 상수 값과 일정하게 증감하는 유형을 모두 예상할 수 있어서 최근 결과 값 예측기보다 다양한 명령어를 예상할 수 있지만 반복형 비 스트라이드 시퀀스 유형 같이 규칙적인 패턴을 결과 값으로 생성하는 명령어들을 예측하지 못한다는 단점이 있다.

2-단계 결과 값 예측기는 명령어가 수행한 서로 다른 마지막 4개의 결과 값을 저장하여 이를 바탕으로 결과 값을 예상하는 방법이다. 높은 예상 정확도를 갖지만 여러 개의 결과 값을 저장하기 때문에 하드웨어 비용이 높다는 단점이 있다.

2.2 혼합형 결과 값 예측기(HVP : Hybrid Value Predictor)

명령어에 의해 생성되는 결과 값의 시퀀스는 다양한 형태로 나타난다. 각각의 예측기는 각 예측기에 적합한 특정 유형(pattern)에 대해서는 좋은 결과를 보이지만 다른 유형에 대해서는 예측율이 떨어지는 문제점이 있다. 따라서 이들 예측기의 장점을 결합한 혼합형 결과 값 예측기들이 제안되었다.

Wang 등은 스트라이드 결과 값 예측기와 2-단계 결과 값 예측기를 혼합한 혼합형 결과 값 예측기를 제안하였다[3]. 이 예측기는 신뢰성 카운터(Confidence Counter)에 의하여 두 예측기 중 한 예측기로부터 예상 값을 사용하는 방법이다. 명령어가 두 개의 예측기에 모두 엔트리를 갖고 있다면, 카운터의 값이 높은 예측기의 예상 값을 선택한다. 높은 예상 정확도를 얻을 수 있었지만 한 예측기만으로 예상될 수 있는 명령어도 두 개의 예측기 테이블에 중복 할당되어 많은 하드웨어 비용을 필요로 한다는 단점이 있다.

Rychlik 등이 제안한 혼합형 결과 값 예측기는 최근 결과 값 예측기, 스트라이드 결과 값 예측기, FCM을 혼합한 혼합형 결과 값 예측기이다[7]. 분류 테이블을 이용하여 실행시간 동안 명령어들의 유형을 분류함으로써 가장 잘 예상 할 수 있는 예측기에 할당하는 동적 분류 방법을 사용하였다. 새로운 명령어를 만나면 바로 예측기를 지정하는 것이 아니라 그림 1과 같이 분류 테

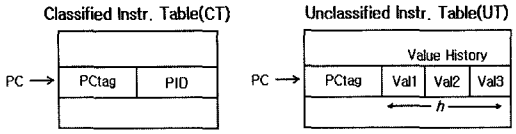

그림 1 Rychlik의 혼합형 예측기의 분류 테이블

이블에서 일정한 준비기간을 갖는다.

CT(Classified instruction Table)는 명령어 주소에 의해 인덱스되고 명령어에 대해 분류된 예측 정보가 PID(Prediction IDentification)에 저장된다. PID는 "Don't Predict", "Use Popular Last Value", "Use Stride+", "Use FCM" 중 하나의 값을 갖는다.

UT(Unclassified instruction Table)는 각 명령어에 대하여 학습 기간동안 3개의 최근 결과 값(Val1, Val2, Val3)을 저장하며, 각 명령어에 대해 적합한 예측기를 선택하기 위하여 사용된다. UT에 있는 결과 값들 사이의 차(strides)가 0이면 "Popular Last Value", 모든 차가 같으면 "Stride+", 이외의 경우에는 "FCM"으로 지정된다. 예상 정확도는 높지만 별도의 분류 테이블이 필요하여 하드웨어 비용이 높고 일정한 준비기간을 걸쳐야하는 단점이 있다.

앞에서 보듯이 명령어의 결과 값을 예상하기 위한 여러 가지 예상방법이 제안되었으나 아직 만족할 만한 예상정확도를 갖지 못하고 있다. 또한 예상 값이 틀릴 경우 회복(recovery)을 위한 패널티가 크므로 예상이 어려운 명령어는 예상하지 않는 것이 오히려 성능 향상에 도움이 된다.

3. 제안된 혼합형 결과 값 예측기

제안된 혼합형 결과 값 예측기는 최근 결과 값 예측기, 스트라이드 결과 값 예측기, 2-단계 결과 값 예측기를 혼합한 구조를 갖는다. 프로파일링(Profiling)으로 얻어진 정적 분류 정보인 예상 타입(Prediction Type)에 따라 반입된 명령어가 적절한 예측기에만 할당되도록 하였으며, 예상이 어려운 명령어들에 대해서는 동적으로 재분류하여 적절한 예측 방법을 선택하도록 구성하였다.

이 장에서는 제안된 HVP의 구조와 정적 분류, 동적 분류 방법에 대하여 살펴보겠다.

3.1 명령어의 정적 분류(Static Classification)

기존의 혼합형 결과 값 예측기는 높은 예상 정확도를 가지나 여러 예측기에 예상할 명령어를 중복하여 할당함으로써 하드웨어 경비가 증가한다는 문제점이 있다. 반입되는 명령어의 실행 유형을 분석할 수 있다면 반입되는 명령어의 실행 유형에 가장 적절한 예측기를 선택하여 할당함으로써 이러한 문제점을 해결할 수 있다. 본 논문에서는 명령어들의 실행 유형 조사하기 위하여 프로파일링(profiling)을 수행함으로써 전체 명령어에 대해 실행된 결과 값들을 분석하였다. 실행 유형을 조사하기 위하여 명령어에 대해 실행된 결과 값을 저장하는 구조는 그림 2와 같으며, 처리 과정은 다음과 같다.

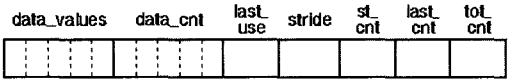

그림 2 정적 분류를 위한 자료구조

실행되는 각 명령어에 의해 생성되는 결과 값을 5개의 결과 값 필드(data_values) 중의 하나에 기록하며, 그 결과 값이 생성된 횟수를 기록하기 위해 대응되는 data_cnt 필드의 값을 '1' 증가시킨다. 바로 이전에 생성된 결과 값과의 차이를 계산하여 차이(stride)를 계산하여 값이 '0'이라면 최근 결과 값 카운터(last_cnt)를 증가시키고, 차이 값이 스트라이드(stride)필드에 저장된 이전 차이 값과 동일하다면 스트라이드 카운터(stride_cnt)를 '1' 증가시킨다. tot_cnt 필드에는 각 명령어가 실행된 횟수를 기록하며, last_use 필드에는 5개의 결과 값 필드에서 마지막으로 기록된 결과 값의 필드 위치를 지정하기 위하여 사용하였다.

모든 명령어들의 실행이 완료된 후 각 명령어에 대해 생성된 결과 값 중 최근 결과 값 실행 유형과 스트라이드 결과 값 실행 유형이 나타나는 비율을 계산하여 그 분포를 조사하였다.

그림 3은 SPECint95 벤치마크 프로그램에 대해 최근 결과 값과 스트라이드 결과 값을 사용하는 명령어들의 실행 유형에 대한 분포를 분석한 결과이다. 최근 결과 값 또는 스트라이드 결과 값의 실행 유형이 나타나는 비율을 x축에 나타내었고, y축은 실행 명령어 중 최근 결과 값 또는 스트라이드 결과 실행 유형의 비율로 실행되는 명령어의 비율을 나타내었다. 예를 들어 vortex 프로그램의 경우, 실행 결과 값 중 50%에 해당하는 결과 값이 최근 결과 값 실행 유형(vortex_last)으로 생성되는 명령어는 전체 명령어 중 4.8%를 차지하고 스트라이드 실행 유형(vortex_st)으로 생성되는 명령어는 전체 명령어 중 0.39%를 나타낸다.

위의 실험 결과에 따라 본 논문에서는 프로파일링을 통해 반복해서 수행하는 명령어들이 생성하는 전체 실행 결과 값 중 50% 이상의 최근 결과 값 실행 유형을 결과 값으로 생성하는 명령어들은 "Last"로, 50% 이상의 스트라이드 결과 값 실행 유형으로 결과 값을 생성하는 명령어들은 "Stride"로 분류하는 정적 분류 정보를 사용한다. 단 한 번의 수행으로 예상하지 않을 명령어 즉, 예측기의 엔트리를 할당하지 않을 명령어는 "Not"으로, 그 외의 명령어들은 혼합된 패턴을 갖고 있어 예상

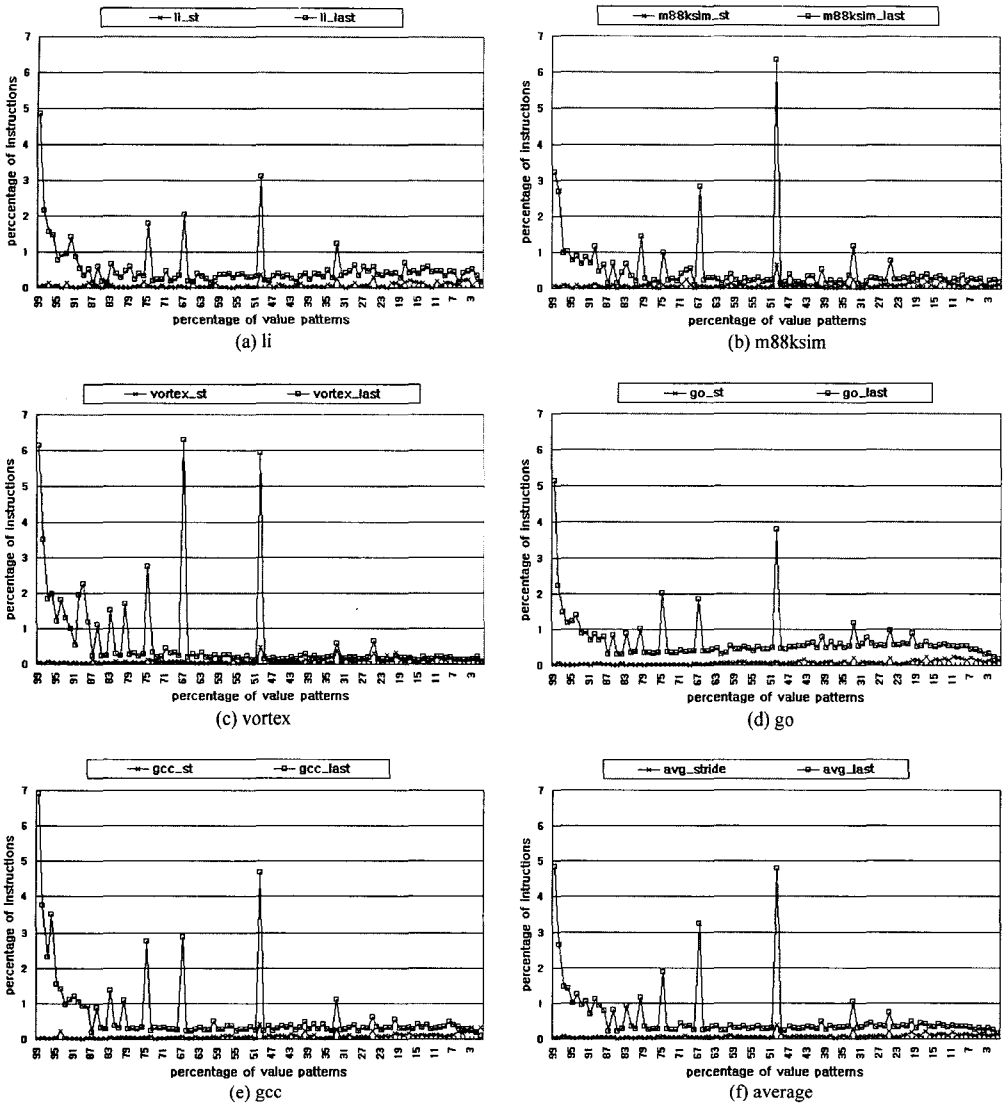

그림 3 벤치마크 프로그램에서 명령어 실행 유형의 분포

이 어려우므로 "Unknown"으로 분류하였으며, 표 1과 같이 명령어의 실행 유형에 따라 예측기를 선택하도록 사용한다.

정적 분류는 프로파일링 단계에서 각 명령어에 대한 실행 유형에 대한 정보를 수집하여, 높은 예상이 가능한 명령어와 예상이 불가능한 명령어를 검출할 수 있다. 실행 유형에 따라 명령어를 분류하는 것은 예상이 불필요한 명령어들이 값 예상의 후보가 되는 것을 피하도록 하여 예상이 가능한 명령어들만 예상 테이블에 할당하여 예상 테이블을 효과적으로 이용할 수 있고, 잘못 예상되는 명령어의 수를 감소시켜 잘못 된 예상에 의한 페널티를 줄일 수 있다. 또한 명령어가 HVP의 여러 예측기에 중복 할당되는 것을 방지하여 하드웨어 경비를 줄여 준다.

정적분류는 프로그램 수행 시 명령어를 예상하기까지 필요한 준비기간을 감소시켜 준다. "Last"로 분류되어 LVP에 할당된 명령어의 경우, 바로 다음 수행부터 예상을 할 수 있고, "Stride"로 분류되어 SVP에 할당된 경우, 다음 수행에서 스트라이드 값이 구해지고, 3번째 수행부터 예상을 할 수 있다.

표 1 명령어의 실행 유형 정보와 선택 예측기

결과 값의 유형		결과 값 유형 예	선택 예측기
Not		한 번 수행으로 예상하지 않을 명령어	예상 않음
Last		5 5 5 5 5 ...	LVP
		결과 값의 증감 폭이 모두 0임	
Stride		1 2 3 4 5 ...	SVP
		증감 폭이 모두 +1임	
Un-known	Non Stride	28 12 -13 99 456 ...	TVP
		증감 폭이 일정하지 않음	
	Repeat Stride	1 2 3 1 2 3 ...	
		내부 루프의 반복 스트라이드값(1,2,3)	
	Repeat Non Stride	1 34 -3 76 1 34 -3 76 ...	
		내부 루프의 반복 비스트라이드 값 (1 34 -3 76)	

3.2 명령어의 동적 분류(Dynamic Classification)

동적 분류방법은 각 명령어의 유형을 분석하여 가장 예상 정확도가 높은 예측기를 실행시간에 분류하는 방법이다.

본 논문에서는 정적 분류 때 다양한 실행 유형으로 예상이 어려운 명령어("Unknown" 실행 유형)로 분류된 명령어들에 대해서 처음 3개의 수행 결과 값이 나타내는 유형에 따라 적합한 예측 방법을 선택하는 동적 분류 방법을 사용한다. 특히 본 논문의 동적 분류방법은 별도의 분류 테이블을 사용하지 않고, 그림 4(a)와 같이 기존의 2 단계 결과 값 예측기의 VHT(Value History Table) 엔트리에 PI(Prediction Identification)를 추가하여 동적 분류를 수행한다. 이 동적 분류방법의 상태 전이도를 그림 4(b)에 나타내었다.

정적 분류에 의해 "Unknown"으로 분류된 명령어가 최초로 수행 시 먼저 TVP에 할당하고 수행 결과 값을 할당된 엔트리의 결과 값 필드에 저장하며, PI 필드의 값을 0으로 설정한다.

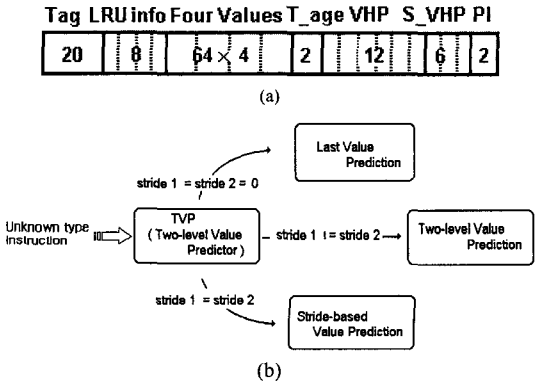

그림 4 (a) TVP 엔트리의 구조, (b) 동적 분류의 상태 전이도

TVP 엔트리의 PI 필드가 초기에 0을 갖는 경우에는 분류기로 동작하게 된다. 처음 3개의 실행 결과 값이 동일한 최근 결과 값 실행 유형(stride1=stride2=0)으로 나타나면 PI 필드의 값을 1로 설정하고, 3개의 실행 결과 값의 차가 동일한 스트라이드 결과 값 실행 유형(stride1=stride2)으로 나타나면 PI 필드의 값을 2로 설정하며, 그 외의 경우에는 PI 필드의 값을 3으로 설정한다.

3.3 제안된 혼합형 결과 값 예측기의 구조 및 동작

3.3.1 제안된 혼합형 결과 값 예측기의 구조

본 논문에서는 앞 절에서 분석한 결과들을 바탕으로, 정적 및 동적 분류를 사용하여 명령어를 가장 적절한 예측기에 할당함으로써 성능을 향상시킬 수 있는 혼합형 결과 값 예측기를 제안한다. 제안된 혼합형 결과 값 예측기는 프로파일링으로 얻어진 정적 분류 정보를 사용하여 명령어 반입 시 명령어를 적절한 예측기에 할당하고, 예상이 어려운 명령어들은 동적으로 재분류하여 적절한 예측 방법을 선택하도록 하였다.

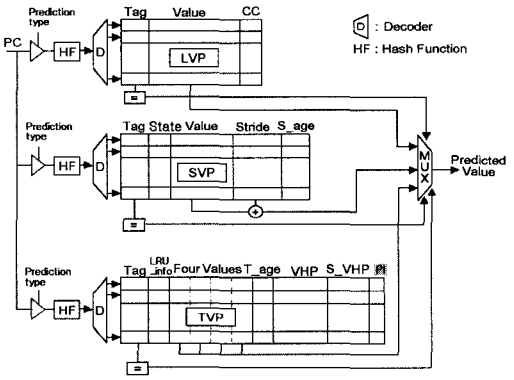

그림 5 제안된 혼합형 결과 값 예측기의 구조

혼합형 결과 값 예측기는 그림 5와 같이 최근 결과 값 예측기, 스트라이드 결과 값 예측기, 2-단계 결과 값 예측기로 구성된 혼합형 예측기로, 분류 테이블과 같은 별도의 하드웨어 추가 없이 동적 분류를 수행하기 위해 2-단계 결과 값 예측기의 엔트리에 PI 필드를 추가하였다. LVP의 신뢰성 카운터(CC : Confidence Counter)는 3비트로 구성되었으며, 임계치는 5로 설정되었다.

SVP의 S_age, TVP의 T_age와 S_VHP는 [5]에서와 같이 모험적 갱신과 부적절한 데이타의 사용으로 인한 예상 실패율을 감소시키는데 사용된다.

3.3.2 제안된 혼합형 결과 값 예측기의 동작

명령어가 반입되었을 경우 제안된 혼합형 결과 값 예측기의 동작은 다음과 같다.

1) 반입된 명령어가 LVP에 엔트리를 갖고 있을 경우

LVP의 신뢰성 카운터 값은 5로 초기화되고, 명령어 반입 시 신뢰성 카운터 값이 임계치 이상이면 결과 값 필드에 있는 값으로 예상하고 임계치(5) 이하이면 예상하지 않는다. 명령어 수행 완료 후 예상 결과가 맞으면 카운터 값을 증가시키고, 예상 결과가 틀렸을 경우 수행 결과를 결과 값 필드에 저장하고 카운터를 감소시킨다.

2) 반입된 명령어가 SVP에 엔트리를 갖고 있을 경우

명령어 반입 시 SVP 엔트리의 상태 필드가 예상 가능한 상태(Steady state)이면 결과 값 필드와 스트라이드 필드의 합으로 결과 값을 예상한다. 명령어 수행 완료 후 예상이 맞았을 경우, 결과 값 필드를 갱신한다. 예상이 틀렸을 경우에는 상태 필드를 과도기 상태(Transient state)로 전이시키고 결과 값 필드와 스트라이드 필드를 갱신한다.

3) 반입된 명령어가 TVP에 엔트리를 갖고 있을 경우

본 논문에서 제안한 예측기에서 TVP는 결과를 예상하는 예측기의 역할은 물론 결과 값들의 실행 유형을 조사하여 적합한 예측 방법을 선택하는 분류기의 역할로도 사용된다.

TVP 엔트리의 PI 필드가 초기에 0을 갖는 경우에는 3.2에서 설명하였듯이 분류기로 동작하게 된다. 동적 분류 과정이 끝나면 TVP는 예측기로 동작하게 된다. 예측기로서의 역할을 보면, PI 필드의 값에 따라 적절한 예측 방법을 사용하여 결과 값을 예상한다.

(1) PI 필드의 값이 1인 경우

최근 결과 값 실행 유형으로 결과 값을 예상한다. 따라서 최후에 저장된 결과 값을 예상 값으로 사용한다.

(2) PI 필드의 값이 2인 경우

스트라이드 결과 값 실행 유형으로 결과 값을 예상한다. 최후에 저장된 결과 값과 스트라이드의 합으로 결과 값을 예상한다.

(3) PI 필드의 값이 3인 경우

기존의 2-단계 결과 값 예측기와 동일한 방법으로 결과 값을 예상한다. 명령어 반입 시 TVP 엔트리의 VHP 필드를 사용하여 PHT 엔트리를 인덱스하고, PHT 엔트리에서 선택된 카운터 값이 임계치 이상이면 해당 카운터에 대응하는 결과 값 필드의 값으로 예상한다.

4. 실험 환경

이 장에서는 본 논문에서 제안한 정적 및 동적 분류를 사용하는 HVP의 성능을 평가 분석하기 위하여, 실험에서 사용된 실험 모델의 기본 설정 값들을 소개한다. 성능을 비교 평가하기 위하여 표 2와 같은, 16 이슈(Issue) 크기에 대한 기본 구조를 사용한다.

RUU는 비순차적 이슈를 위한 명령 윈도우와 리오더 버퍼(Reorder Buffer)를 결합한 것으로 256 엔트리의 큐 구조로 구성되며, LSQ(Load Store Queue)는 비순차적 이슈 머신에서 메모리 참조 명령어들의 수행된 순서를 유지하기 위해 사용되는 큐로 256엔트리로 구성하였다. 분기 예측을 위해 1K 엔트리를 갖는 2-단계 예측기를 사용하였다.

표 2 기본 시스템 구조의 구성

	인수	값	비고
Processor Core	RUU size	256	Instruction window
	LSQ size	256	Load-Store queue
	Fetch width	16 instr./cycle	
	Decode width	16 instr./cycle	
	Issue width	16 instr./cycle	
	Commit width	16 instr./cycle	
	Functional units	10/20 int alu(1) 4/8 fp add(2) 2/4 int mult(3) div(12) 2/4 fp mult(4) div(12)	( ) is latency
Branch Predictor	2lev	1K entry 8 history size	Two level predictor
Cache	L1 Inst-cache	512K, direct ma-p, 32byte blocks	Instruction cache (1 cycle hit latency)
	L1 Data-cache	128K, 4-way, 32 byte blocks	Data cache (1 cycle hit latency)
	L2 Unified-cache	1024K, 4-way, 64 cache-blocks	Secondary cache (6cycle hit latency)

표 3 실험에 사용된 HVP의 구성

결과 값 예측기		테이블 구성
기본 HVP	LVP	8K VHT
	SVP	8K VHT
	TVP	8K VHT, 4K PHT, 4 history values
제안한 HVP	LVP	8K/4K VHT
	SVP	8K/4K/2K/512/256 VHT
	TVP	8K/4K/2K/1K VHT, 4K PHT, 4 history values

표 3은 실험에서 사용된 HVP들의 구성을 나타내고 있다. 기본 HVP는 LVP, SVP, TVP로 구성되어 있다.

실험은 8K 엔트리 VHT, 4K 엔트리 VHT를 갖는 HVP에 제안한 개념들을 적용하며 각각을 실험하여 예상정확도를 분석하였다. 또한 LVP의 VHT를 4K 엔트리로 고정하고 SVP와 TVP의 VHT 크기를 정적분류 정보에 따라 다양하게 구성하여 하드웨어 비용에 대한 영향을 실험하였다.

본 논문의 실험을 위해 SPECint95 벤치마크 중 8개를 실험하였다. 표 4는 실험에서 사용한 벤치마크프로그램과 입력데이타를 보여주고 있다.

각 벤치마크 프로그램은 -O2 -g3의 최적화 옵션을

표 4 벤치마크 프로그램과 입력 데이타

벤치마크	입력 데이타	프로그램 특성
go	50 9 2stone9.in	Game playing
m88ksim	tiny.in	A simulator for the 88100 processor
vortex	vortex.tiny.in	A single user object oriented database transaction benchmark
li	queen6.lsp	LISP interpreter
gcc	jump.i	A C compiler based on GNU C compiler version 2.7.2
perl	scrabbl.pl scrabbl.in	Anagram search program
ijpeg	tinyrose.ppm	JPEG encoder
compress	test.in	A data compression program using adaptive Lempel-Ziv coding

가진 GNU gcc 2.7.2를 사용하여 컴파일했으며 실험은 각 벤치마크 프로그램의 수행된 명령어 수를 최대 100M까지 제한적으로 수행하였다.

본 연구에서 사용된 시뮬레이터는 결과 값 예측기의 최적 설계 파라미터의 추출과 검증을 위해서 Simple-Scalar 툴셋[13]을 사용하였다. MIPS-4 명령어 집합을 기반으로 하는 SimpleScalar 아키텍처는 비 순서적(Out-of-Order) 슈퍼스칼라 프로세서 시뮬레이터를 수행할 수 있고, 비 블록킹 캐시(non-blocking cache)의 수행을 지원한다.

5. 실험 결과

이 장에서는 정적 및 동적 분류를 사용하는 제안한 혼합형 결과 값 예측기의 예상 정확도에 대해서 살펴보겠다.

5.1 정적 분류 정보 적용

프로파일링으로 얻어진 정적 분류 정보를 사용하여 명령어 반입 시 HVP가 명령어에 적절한 예측기를 선택하도록 구성하였을 경우의 성능 향상을 살펴보도록 한다.

그림 6은 정적 분류를 사용하지 않는 HVP와 정적 분류 정보를 적용시켰을 경우의 예상 정확도를 보여주고 있다.

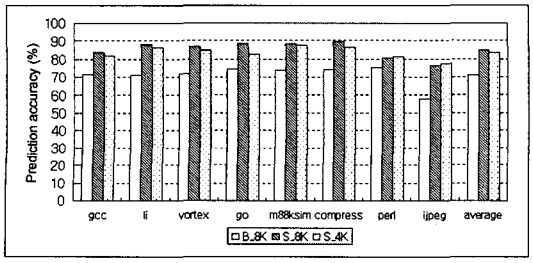

그림 6 정적 분류 정보를 사용한 예상 정확도

그림 6에서 B_8K는 분류를 사용하지 않는 8K 엔트리의 VHT를 갖는 기본 HVP를 나타내고, S_8K, S_4K는 정적 분류를 사용하는 8K 엔트리, 4K 엔트리의 VHT를 갖는 HVP를 나타낸다. 정적 분류를 사용하는 S_8K는 예상정확도가 분류를 사용하지 않는 기존의 B_8K의 평균 71.2%에 비해 평균 85.2%로 크게 향상되었다. 또한 각 예측기의 VHT를 4K 엔트리로 축소한 S_4K도 평균 83.7%로 예상 정확도가 약간 감소하였다.

이 결과는 정적 분류 정보를 사용하여 명령어 반입 시 한번의 수행 등으로 예상이 필요없는 명령어는 예측기에 할당하지 않기 때문에 예상을 수행하지 않으며, 예상 가능한 명령어를 적절한 예측기에만 할당함으로써 중복 할당을 피할 수 있어, 예측 테이블을 효과적으로 이용함으로써 예상정확도가 향상되었음을 보여주고 있다.

5.2 정적 및 동적 분류 적용

정적 분류 정보를 사용한 HVP에서 예상이 어려운 "Unknown" 유형의 명령어에 대해 동적분류 방법을 적용했을 때 성능향상을 살펴보도록 한다.

그림 7은 제안된 HVP에 정적 및 동적 분류를 모두 적용시켰을 경우의 예상정확도를 보여주고 있다.

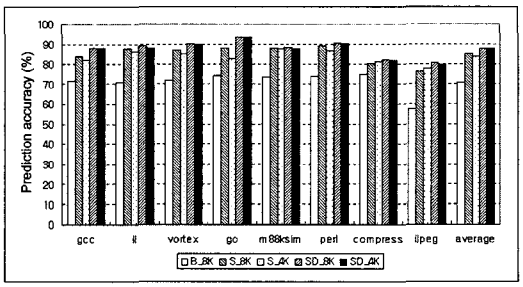

그림 7 정적 및 동적 분류 정보를 사용한 예상 정확도

그림 7에서 SD_8K, SD_4K는 정적 및 동적 분류를 모두 사용하는 8K 엔트리, 4K 엔트리의 VHT를 갖는 HVP를 나타낸다. HVP에 정적 및 동적 분류를 모두 사용한 SD_8K의 예상정확도는 분류를 사용하지 않는 B_8K의 71.2%와 정적 분류를 사용하는 S_8K의 평균 85.2%에 비해 평균 87.9%로 향상되었다. VHT 크기를 4K 엔트리로 축소한 SD_4K도 "Table miss"의 증가로 예상 정확도가 87.5%로 낮아지나, S_8K 예측기보다는 예상 정확도가 높았다.

이 결과는 예상이 어려운 "Unknown" 유형의 명령어에 대하여 동적으로 실행 유형을 분석하여 가장 적절한 예측 방법을 선택하여 예상을 수행함으로써 보다 향상된 성능을 얻을 수 있음을 보여주고 있다.

실험을 통해 얻어진 결과를 살펴보면 기존 HVP에 본 논문에서 제안한 정적 및 동적 분류방법 등을 단계적으로 추가하여 적용하였을 경우 예상 정확도가 단계적으로 향상되고 있음을 확인할 수 있다. 이는 반입되는 각 명령어에 정적 및 동적 분류방법으로 가장 적절한 예상 방법을 찾아 정확한 예상을 수행하고 있음을 보여주는 것이다.

5.3 예측기 테이블의 크기 변화 시 예상 정확도

본 절에서는 정적 및 동적 분류 정보를 사용하는 HVP에 대해 각 예측기의 VHT 크기를 변화시키면서 예상 정확도의 변화와 하드웨어의 비용 감소 효과를 살펴보도록 한다. 프로파일링을 통해 얻어진 명령어의 실행 유형 분포 비율에 따라 가장 많은 비중을 나타내는 LVP의 VHT 엔트리 수를 4K로 고정하고 다른 예측기의 VHT 엔트리 수를 다양하게 구성하여 하드웨어 크기에 따른 예상 정확도를 측정하였다. 표 5는 HVP의 엔트리 수로 하드웨어 크기를 비교하였다.

SD_422K 예측기는 SD_4K 예측기와 비교하여 전체 테이블 크기를 50%로 축소할 수 있고, SD_8K와 비교하면 전체 예측기 테이블의 크기를 25%로 축소하는 효과를 얻을 수 있다. SD_451K은 SD_4K에 비해 전체 테이블 크기를 36%로 축소할 수 있다.

그림 8은 제안한 정적 및 동적 분류를 사용한 혼합형 예측기의 VHT 크기를 변화시키며 측정한 예상 정확도를 보여준다.

표 5 실험 예측기의 하드웨어 크기 비교

예측기	엔트리 구성			테이블 크기 (bit)	비 고
	LVP	SVP	TVP		
SD_8K	8192	8192	8192	2,376K	각 8K 엔트리의 VHT
SD_4K	4096	4096	4096	1,188K	각 4K 엔트리의 VHT
SD_42K	4096	2048	2048	704K	LVP 4K, SVP 2K, TVP 2K 엔트리의 VHT
SD_451K	4096	512	1024	430K	LVP 4K, SVP 512, TVP 1K 엔트리의 VHT
SD_422K	4096	256	2048	592K	LVP 4K, SVP 256, TVP 2K 엔트리의 VHT

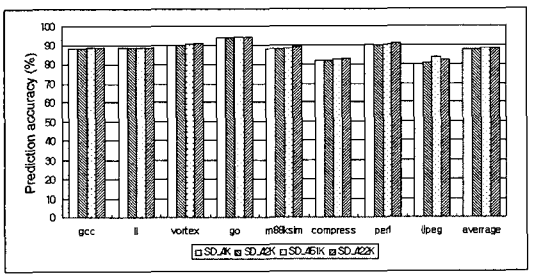

그림 8 예측기 테이블의 크기 변화 시 예상 정확도

각 4K VHT 엔트리를 갖는 SD_4K 예측기의 예상정확도가 각 8K VHT 엔트리를 갖는 SD_8K 예측기의 예상 정확도 87.9%(그림 7)에 비해 87.5%로 약간의 감소를 보이고 있으나, 하드웨어의 크기를 약 절반 크기로 축소하는 효과를 얻을 수 있었다. 또한, LVP의 VHT 크기를 4K 엔트리로 고정하고 SVP와 TVP의 VHT 크기를 변화시키며 측정한 예상 정확도도 각각 87.6% (SD_42K), 88.26%(SD_451K), 88.28 (SD_422K)로 나타나 비슷한 결과를 보이고 있다. 이와 같이 SVP와 TVP의 엔트리 수가 크게 줄어들었음에도 예상 정확도가 비슷하게 나타나는 것은 SVP와 TVP는 예상이 적중할 가능성이 높은 명령어만을 예상하는 것이 유리하기 때문이다. 테이블 크기가 크면 예상 정확도가 높지 않은 명령어도 예상하기 때문에 오히려 예상정확도가 떨어질 수도 있다.

5.4 성능 향상(Speed Up)

결과 값 예측기를 사용하지 않는 기본 구조의 예측기와 제안한 다양한 혼합형 결과 값 예측기에 대해 IPC(Instructions Per Cycle)를 측정하여 성능 향상에 미치는 영향을 분석한다. IPC는 한 사이클 당 명령들을 공급할 수 있는 능력으로써 한 사이클 당 많은 명령을 공급하게 되면 시스템의 성능을 향상시킬 수 있다.

그림 9는 실험에 사용된 기본 구조에 비해 벤치마크별로 제안된 예측기들의 IPC 증가율을 보여주고 있다.

실험 결과 제안한 예측기들은 기본 구조보다 평균 최소 4.4%에서 최대 6.9%까지 IPC가 증가함을 보여주고 있다. 8K 엔트리의 VHT를 갖는 예측기들에 대해서 IPC 증가율을 살펴보면 기존의 혼합형 결과 값 예측기(B_8K)가 4.4%, 정적 분류를 사용한 예측기(S_8K)가 6.4%, 정적 및 동적 분류 모두를 사용하는 예측기(SD_8K)가 6.7% 증가한 것으로 나타나고 있다. 각 4K 엔트리의 VHT를 갖고 정적 분류를 사용하는 예측기(S_4K)는 6.2%, 정적 및 동적 분류를 사용하는 예측기(SD_4K)도 기본 구조보다 6.5% 증가된 IPC를 보이고 있다. 또한 하드웨어 크기를 최소로 구성한 예측기(SD_451K)가 가장 큰 6.9%의 증가를 보이고 있다. 태

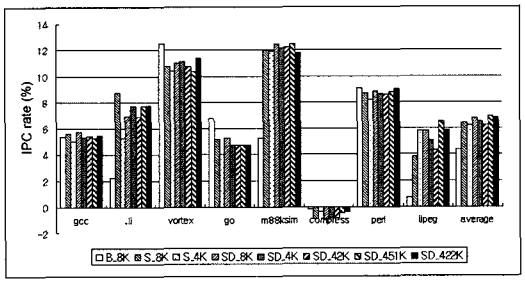

그림 9 각 예측기의 IPC 비교

이블 크기가 크게 되면 준비기간이 증가하므로 IPC가 감소되어 성능이 감소된다.

본 논문에서 제안한 정적 및 동적 분류 정보를 사용하는 혼합형 결과 값 예측기는 예상 정확도가 높을 뿐만 아니라, 명령어의 정적 분류 및 동적 분류 정보에 따라 적절한 예측 방법을 선택하도록 하고 한번 수행만으로 예상할 필요가 없는 명령어는 예상을 수행하지 않도록 함으로써 잘못 예상으로 인한 페널티를 감소시켜 기존의 혼합형 결과 값 예측기보다 전체적인 시스템의 성능 향상을 보이고 있다.

6. 결 론

본 논문에서는 정적 및 동적 분류를 사용하는 혼합형 결과 값 예측기를 제안하였다. 제안한 예측기는 프로파일링으로 얻어진 정적 분류 정보를 사용함으로써 명령어 반입 시 명령어를 적절한 예측기에 할당하여 동일 명령어가 여러 예측기의 엔트리에 중복 할당되는 문제점을 해결할 수 있었다. 또한 예상이 어려운 실행 유형을 갖는 명령어에 대해서는 동적으로 적절한 예측 방법을 선택하도록 하여 예상 실패로 인한 페널티를 축소하고, 예상 정확도를 더욱 향상시키도록 하였다.

정적 분류 정보만을 사용 시 각 예측기의 VHT를 8K 엔트리로 했을 때 평균 예상 정확도가 85.2%로 정적 분류 정보를 사용하지 않은 예측기보다 14% 향상되었다. 또한, 예상이 어려운 명령어에 대해서도 동적으로 적절한 예측 방법을 선택하도록 하여 정적 및 동적 분류를 모두 사용함으로써 예상 정확도가 평균 87.9%로 향상되었다.

각 4K VHT 엔트리를 갖는 SD_4K의 예상 정확도는 각 8K 엔트리를 갖는 SD_8K의 87.9%와 비교하여 87.5%로 약간 감소하나 하드웨어 크기를 절반으로 줄일 수 있었다. LVP 4K, SVP 256, TVP 2K 엔트리의 VHT로 구성되는 SD_422K는 SD_8K와 비교하여 하드웨어 크기를 25% 정도 줄일 수 있었고 예상 정확도는 비슷하게 나타났다.

향후 과제로 명령어에 적합한 예측기를 선택할 수 있는 컴파일러 기반 분류 알고리즘을 개발에 대한 연구가 필요하다. 또한, 예상이 어려운 혼합된 실행 유형을 갖는 명령어들을 보다 정확히 분석할 수 있는 동적 분류 방법에 대한 연구가 필요하다고 생각된다.

참 고 문 헌

[1] M. H. Lipasti and J. P. Shen, "Exceeding the Dataflow Limit via Value Prediction," MICRO-29, pp. 226-237, December 1996.

[2] J. Gonzalez, and A. Gonzalez, "The Potential of Data Value Speculation to Boot ILP," In 12th International Conference on Supercomputing, 1998.

[3] K. Wang and M. Franklin, "Highly Accurate Data Value Prediction Using Hybrid Predictors," MICRO-30, pp. 281-290, December 1997.

[4] 박홍준, 신영호, 조영일, "슈퍼스칼라 프로세서에서 모험적 갱신을 사용한 하이브리드 결과 값 예측기", 한국정보과학회 논문지(시스템 및 이론), 제28권 제11호, pp. 592-600, December 2001.

[5] 박홍준, 조영일, "슈퍼스칼라 프로세서에서 예상 테이블의 모험적 갱신과 명령어 실행 유형의 정적 분류를 이용한 혼합형 결과 값 예측기", 한국정보과학회 논문지(컴퓨팅의 실제), 제8권 제1호, pp. 107-115, February, 2002.

[6] Q. Zhao. D. J. Lilja, "Compiler-Directed Static Classification of Value Locality Behavior," Laboratory for Advanced Research in Computing Technology and Compilers Technical Report No. ARCTiC 00-07, July, 2000.

[7] B. Rychlik, J. W. Faistl, B. P. Krug, A. Y. Kurland, J. J. Sung, M. N. Velev, J. P. Shen, "Efficient and Accurate Value Prediction Using Dynamic Classification," Tech. Rep, CMUART, April 1998.

[8] B. Calder, G. Reinman, D. M. Tullsen, "Selective Value Prediction," ISCA-26, May 1999.

[9] F. Gabbay, and A. Mendelson, "The Effect of Instruction Fetch Bandwidth on Value Prediction," ISCA-25, pp. 272-281, 1998.

[10] Sang-Jung Lee, Yuan Wang and Pen-Chung Yew, "Decoupled Value Prediction on Trace Processors," HPCA-6, pp. 231-240, January 2000.

[11] Sang-Jung Lee, Pen-Chung Yew, "On Some Implementation Issue for Value Prediction on Wide-Issue ILP Processors," PACT 2000, October 2000.

[12] G. Reinman and B. Calder, "Predictive Techniques for Aggressive Load Speculation," MICRO-31, pp. 127-137, December 1998.

[13] D. C. Burger and T. M. Austin, "The Simple-Scalar Tool Set, Version 2.0," Technical Report CS-TR-97-1342, University of Wisconsin, Madison, June 1997.

김 주 익

1996년 수원대학교 전자계산학과 이학사 1998년 수원대학교 전자계산학과 이학석사. 2003년 수원대학교 컴퓨터학과 이학박사 수료. 관심분야는 슈퍼스칼라 프로세서에서 결과 예상 방법, ILP 프로세서의 정적 및 동적 명령어 스케줄링 등

박 홍 준

1984년 아주대학교 전자계산학과 공학사 1987년 한양대학교 전자계산학과 공학석사. 2001년 수원대학교 전자계산학과 이학박사. 1994년 3월~현재 극동정보대학 전산정보처리과 부교수. 관심분야는 ILP 프로세서의 분기예상 메커니즘 및 결과값 예상 메커니즘, ILP 프로세서의 정적 및 동적 명령어 스케줄링 등

조 영 일

1980년 한양대학교 전자공학과 공학사 1982년 한양대학교 전자공학과 공학석사 1985년 한양대학교 전자공학과 공학박사 1986년 3월~현재 수원대학교 정보공학대학 컴퓨터학과 교수. 관심분야는 ILP 프로세서, 멀티미디어 통신, 최적화 컴파일러 등