(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
G06F 9/38 (2006.01)

(45) 공고일자 2006년07월24일
(11) 등록번호 10-0603067
(24) 등록일자 2006년07월12일

(21) 출원번호 10-2000-7006742
(22) 출원일자 2000년06월17일
번역문 제출일자 2000년06월17일
(86) 국제출원번호 PCT/US1998/011987
국제출원일자 1998년06월10일
(65) 공개번호 10-2001-0033300
(43) 공개일자 2001년04월25일
(87) 국제공개번호 WO 1999/32966
국제공개일자 1999년07월01일

(81) 지정국 국내특허 : 일본, 대한민국,

EP 유럽특허 : 오스트리아, 벨기에, 스위스, 독일, 덴마크, 스페인, 프랑스, 영국, 그리스, 아일랜드, 이탈리아, 룩셈부르크, 모나코, 네덜란드, 포르투칼, 스웨덴, 핀란드, 사이프러스,

(30) 우선권주장 08/994,869 1997년12월19일 미국(US)

(73) 특허권자 어드밴스드 마이크로 디바이시즈, 인코포레이티드
미국 캘리포니아 94088-3453 서니베일 원 에이엠디 플레이스 메일 스톱68

(72) 발명자 무투사미카티케얀
미국텍사스78728오스틴페더크레스트드라이브2909

(74) 대리인 박장원

심사관 : 성경아

(54) 분기 예측의 타입을 분류하기 위해 복귀 선택 비트들을 이용하는 분기 예측

요약

분기 예측 유닛은 명령 캐시에 저장된 연속적인 명령 바이트들의 그룹 각각에 대응하는 분기 선택자들의 세트를 저장한다. 분기 선택자는, 분기 선택자에 대응하는 페치 어드레스가 존재하는 경우 선택될 분기 예측을 특정한다. 분기 예측 유닛은 또한 1개 이상의 분기 예측들에 대응하는 복귀 선택자들의 세트를 저장한다. 복귀 선택자들은 분기 선택의 타입을 특정한다. 예를 들어, 분기 예측들은 순차적인 분기 예측 및 분기 명령 분기 예측을 포함할 수 있다. 복귀 선택자들은, 분기 명령 분기 예측이 복귀 명령에 관련되는지, 아니면 비복귀 분기 명령에 관련되는 지를 특정할 수 있다.

대표도

도 11

색인어

분기 예측, 분기 선택자, 바이트 범위, 최소, 명령 캐시, 마이크로프로세서

명세서

기술분야

본 발명은 마이크로프로세서 분야에 관한 것으로서, 특히 마이크로프로세서 내에서의 분기 예측 메커니즘에 관한 것이다.

배경기술

EP-A-0 649 084는 슈퍼스칼라(superscalar), 슈퍼 파이프라인 마이크로프로세서에 이용하기 위한 분기 처리 유닛을 개시한다. 분기 예측 유닛은 분기들의 방향을 예측하고, 테이큰(taken)인 것으로 예측된 분기들에 대한 목표 어드레스들과, 그리고 점프들, 호출들 및 복귀들과 같은 흐름 명령들의 무조건적인 변경을 제공한다. 분기 예측 유닛은 프리페치된 어드레스를 이용하여 액세스되며, 목표 어드레스들을 프리페처에 제공한다. 방향의 분기 예측은 이력 캐시(History Cache)를 이용하고, 어드레스의 분기 예측은 복귀 비트들과 함께 목표 캐시를 이용한다. 목표 캐시는 명령 블록 내의 분기 오프셋에 대한 필드를 갖는, 4방향 캐시로서 실시된다. 8 바이트 기본 블럭 어드레스들을 이용한 액세스가 이루어진다. 최대 8개의 분기들에 대한 데이터가 명령들의 16바이트들 마다 저장될 수 있다. 목표 캐시는 테이큰 분기들로부터의 데이터를 저장하고, 이력 캐시는 낫테이큰(not-taken) 분기들로부터의 데이터를 저장한다.
US-A-5 574 871은 페치된 명령들의 블록에 대한 분기 예측을 개시한다. 분기 이력표는 128 방향 캐시로서 실시되며, 테이큰 및 낫테이큰 분기들에 대한 예측들을 포함하는바, 4개의 분기 타입들, 즉 조건적, 무조건적, 복귀 및 호출에 대한 코드를 제공하는 분기 이력표 내의 2개의 비트들을 갖는다. 4개의 엔트리들은 블럭 어드레스마다 제공되며, 테이큰인 것으로 예측된 제 1 분기에 대한 예측은 페치 어드레스 이후에 알 수 있다.
슈퍼스칼라 마이크로프로세서들은, 클럭 주기당 다수의 명령들을 실행하고, 설계에 적합한 가능한 가장 짧은 주기를 선택함으로써, 고성능을 구현한다. 본원에서 이용되는 "클럭 주기"라는 용어는 마이크로프로세서 내의 명령 처리 파이프라인의 다양한 스테이지들에 부여되는 시간 간격을 말한다. 저장 디바이스들(예를 들어, 레지스터들 및 어레이들)은 클럭 주기에 따라 그들의 값들을 포착한다. 예를 들어, 저장 디바이스는 클럭 주기를 규정하는 클럭 신호의 상승 또는 하강 에지에 따라 어떤 값을 포착할 수 있다. 그런 다음, 저장 디바이스는 클럭 신호의 이후의 상승 또는 하강 에지까지 상기 값을 각각 저장한다. 본원에서 "명령 처리 파이프라인"이란 용어는 파이프라인 방식으로 명령들을 처리하는 데에 이용되는 논리 회로들을 말한다. 비록 파이프라인은 명령 처리의 일부가 실행되는 어떠한 수의 스테이지들로 분할되기는 하지만, 명령 처리는 일반적으로 명령 페치, 명령 디코딩, 명령 실행 및 명령에 의해 특정된 목적지로의 실행 결과의 저장을 포함한다.

슈퍼스칼라 마이크로프로세서(및 슈퍼파이프라인 마이크로프로세서)의 중요한 특징은 그의 분기 예측 메커니즘이다. 이 분기 예측 메커니즘은 분기 명령에 대한 예측된 방향(테이큰 또는 낫테이큰)을 나타냄으로써, 분기 예측에 의해 나타나는 예측된 분기 명령 스트림 내에서 후속 명령의 페치가 계속될 수 있게 된다. 분기 명령은 적어도 2개의 어드레스들, 즉 분기 명령에 직접 계속되는 명령로 시작하는 명령 스트림을 특정하는 순차 어드레스와, 메모리의 임의 위치에서 시작하는 명령 스트림을 특정하는 목표 어드레스중 하나로부터 이후의 명령들이 페치될 수 있게 하는 명령이다. 무조건 분기(unconditional branch) 명령은 항상 목표 어드레스로 분기하며, 조건 분기(conditional branch) 명령은 이전 명령의 결과에 기초하여 순차 어드레스 또는 목표 어드레스중 하나를 선택할 수 있다. 예측된 명령 스트림으로부터의 명령들은 분기 명령의 실행 이전에 추론적으로 실행될 수 있으며, 그리고 어떠한 경우에도 분기 명령 실행 이전에 명령 처리 파이프라인 내에 위치된다. 예측 명령 스트림이 정확하면, 유익하게는 클럭 주기당 실행되는 명령들의 수가 증가한다. 그러나, 예측 명령 스트림이 정확하지 않다면(즉, 1개 이상의 분기 명령들이 잘못 예측(오예측)되었다면), 잘못 예측된 명령 스트림으로부터의 명령들은 명령 처리 파이프라인으로부터 버려지고, 클럭 주기당 실행되는 명령들의 수는 감소된다.

효과적으로 하기 위해, 예측된 명령 스트림이 가능한한 자주 정확하도록, 분기 예측 메커니즘은 매우 정확해야 한다. 전형적으로, 분기 예측 메커니즘의 정확도의 증가는 분기 예측 메커니즘의 복잡성을 증가시킴으로써 달성된다. 예를 들어, 캐시 라인(cache-line) 기반 분기 예측 설계가 이용될 수 있는바, 여기서 분기 예측은 명령 캐시 내의 명령 바이트들의 특정한 캐시 라인에 의해 저장된다. 캐시 라인은 명령 캐시 내의 저장 공간의 할당(allocation)과 할당 해제(deallocation)를 위한 유닛으로서 처리되는 다수의 연속 바이트들이다. 캐시 라인이 페치되면, 대응하는 분기 예측들이 또한 페치된다. 또한, 특정 캐시 라인이 버려지면, 대응하는 분기 예측들 또한 버려진다. 캐시 라인은 메모리 내에 정렬된다. 캐시 라인 기반 분기 예측 설계는 각 캐시 라인에 대해 보다 많은 수의 분기 예측들을 저장함으로써 보다 정확해질 수 있다. 소정의 캐시 라인은 다수의 분기 명령들을 포함할 수 있는바, 각 분기 명령은 다른 분기 예측에 의해 표현된다. 따라서, 하나의 캐시 라인에 할당된 보다 많은 분기 예측들은 분기 예측 메커니즘에 의해 보다 많은 분기 명령들이 표현 및 예측될 수 있게 한다. 분기 예측 메커니즘 내에 표현될 수 없는 분기 명령은 예측되지 않으며, 이후 분기가 테이큰인 것으로 밝혀지면, "오예측"이 검출될 수 있다. 그러나, 부가적인 분기 예측들 간에서 선택되야 하는 요구에 의해, 예측 메커니즘의 복잡성이 증가한다.
본원에서 이용되는 "분기 예측"은, 분기 예측 메커니즘에 의해 분기 명령이 테이큰인지 낫테이큰인지의 여부의 예측으로서

해석될 수 있는 값이다. 또한, 분기 예측은 목표 어드레스를 포함할 수 있다. 캐시 라인 기반 분기 예측 메커니즘에 있어서, 페치되고 있는 캐시 라인에 대한 순차 라인의 예측은, 캐시 라인으로부터 페치되고 있는 명령들 내에서 어떠한 분기 명령들도 없을 때의 분기 예측이다.

분기 예측 메커니즘에 관련된 문제는 많은 종류의 분기 명령들이 충돌할 수 있다는 것이다. 이러한 종류의 분기 명령들은 서로 다른 종류의 분기 예측들 및 서로 다른 목표 어드레스들을 필요로 할 수 있다. 예를 들어, 복귀 명령은 항상 테이큰인 것으로 예측될 수 있으며, 목표 어드레스는 복귀 어드레스 스택으로부터 검색된다. 대조적으로, 조건 분기 명령은 테이큰 또는 낫테이큰인 것으로 예측될 수 있으며, 목표 어드레스는 목표 어드레스 어레이로부터 얻을 수 있다. 서로 다른 종류의 명령들에 대처하기 위해, 각 캐시 라인에 대한 분기 예측들이 서로 다른 종류의 분기 명령들 간에 할당된다. 예를 들어, 1개의 분기 예측이 순차 라인에 할당될 수 있고, 1개의 분기 예측이 복귀 명령에 할당될 수 있으며, 그리고 2개의 분기 예측들이 비복귀 분기 명령들에 할당될 수 있다. 불행히도, 상기 방식은 2개의 복귀 명령들 또는 3개의 비복귀 분기 명령들을 갖는 캐시 라인들에는 대처하지 못한다.

발명의 상세한 설명

상기 설명한 문제들은 본 발명에 따른 분기 예측 장치 및 방법에 의해 해결된다. 본 발명의 분기 예측 장치는, 특정 분기 예측이 복귀 명령과 관련되는지, 아니면 비복귀 분기 명령과 관련되는 지를 특정하는 1개 이상의 복귀 선택자 비트들을 저장한다. 이러한 방식으로, 각 분기 예측은 복귀 명령들 또는 비복귀 명령들에 대처할 수 있다. 따라서, 최소의 부가적인 저장 공간 및 회로를 이용하여, 복귀 및 분기 명령들의 보다 많은 조합들에 대처할 수 있게 된다.

대체적으로 말하여, 본 발명은 분기 예측 저장부 및 선택 메커니즘을 포함하는 분기 예측 장치를 제시한다. 분기 예측 저장부는 명령 캐시로부터 페치되는 연속적인 명령 바이트들의 그룹에 대응하는 페치 어드레스를 수신하도록 결합된다. 분기 예측 저장부는 다수의 분기 선택자들 및 복귀 선택자를 저장하도록 구성된다. 다수의 분기 선택자들중 적어도 1개는 연속적인 명령 바이트들의 그룹 내의 제 1 명령에 대응하고, 복귀 선택자는 분기 선택자들중 적어도 1개에 대응한다. 다수의 분기 선택자들중 1개와 복귀 선택자는, 제 1 명령이 페치되는 경우 선택될 특정 분기 예측을 특정한다. 선택 메커니즘은 다수의 분기 선택자들 및 복귀 선택자를 수신하도록 분기 예측 저장부에 결합된다. 선택 메커니즘은, 연속적인 명령 바이트들의 그룹을 페치하는 데에 이용되는 페치 어드레스의 다수의 최하위 비트들에 응답하여 다수의 분기 선택자들중 특정한 1개를 선택하고, 특정 분기 선택자 및 복귀 선택자에 응답하여 특정 분기 예측을 선택하도록 구성된다.

본 발명은 다수의 명령들을 포함하는 연속적인 명령 바이트들의 그룹에 대응하는 분기 예측을 선택하는 방법을 제시하는 바, 이 방법은 분기 선택 저장부에 다수의 분기 선택자들을 저장하는 단계와, 여기서 상기 다수의 분기 선택자들중 적어도 1개는 다수의 명령들중 제 1 명령에 대응하며; 상기 분기 예측 저장부에 복귀 선택자를 저장하는 단계와, 여기서 상기 복귀 선택자는 다수의 분기 선택자들중 적어도 1개에 대응하고, 상기 다수의 분기 선택자들중 적어도 1개와 복귀 선택자는 상기 다수의 명령들중 제 1 명령이 페치되는 경우 선택될 특정 분기 명령을 특정하며; 연속적인 명령 바이트들의 그룹을 페치함과 동시에, 상기 연속적인 명령 바이트들의 그룹을 특정하는 페치 어드레스를 이용하여 다수의 분기 선택자들 및 복귀 선택자를 페치하는 단계와; 상기 페치 어드레스에 응답하여 상기 다수의 분기 선택자들중 1개를 선택하는 단계와; 그리고 상기 다수의 분기 선택자들중 1개와 상기 복귀 선택자에 응답하여 분기 예측을 선택하는 단계를 포함한다.

본 발명은 또한 명령 캐시 및 분기 예측 유닛을 포함하는 마이크로프로세서를 제시한다. 상기 명령 캐시는, 명령 바이트들의 다수의 캐시 라인들을 저장하고, 그리고 페치 어드레스 수신시 상기 명령 바이트들의 그룹을 상기 마이크로프로세서의 명령 처리 파이프라인에 제공하도록 구성된다. 상기 분기 예측 유닛은 상기 명령 캐시에 결합되고, 상기 명령 캐시와 동시에 상기 페치 어드레스를 수신하도록 결합된다. 상기 분기 예측 유닛은, 상기 명령 바이트들의 그룹에 대해 다수의 분기 선택자들 및 복귀 선택자를 저장하도록 구성되며, 상기 페치 어드레스에 응답하여 상기 다수의 분기 선택자들중 1개를 선택하도록 구성된다. 다수의 분기 선택자들중 1개와 상기 복귀 선택자는 명령 캐시에 의해 순차적인 페치 어드레스로서 이용되는 분기 예측을 특정한다.

도면의 간단한 설명

본 발명의 다른 목적들 및 장점들은 첨부 도면을 참조하여 설명되는 하기의 상세한 설명으로부터 보다 명확해질 것이다.

도 1은 수퍼스칼라 마이크로프로세서 일 실시예의 블록선도이다.

도 2는 도 1에 도시된 한 쌍의 디코드 유닛들의 일 실시예의 블록선도이다.

도 3은 연속적인 명령 바이트들의 그룹과, 대응하는 분기 선택자들 및 복귀 선택자들의 세트를 도시한다.

도 4는 도 1에 도시한 마이크로프로세서에 대한 분기 예측 메커니즘의 일 실시예에 따른 것으로, 연속적인 명령 바이트들의 그룹과, 대응하는 분기 선택자들 및 복귀 선택자들의 세트를 도시한다.

도 5는 연속적인 명령 바이트들의 세트 내에서의 바이트 범위의 일 실시예를 도시한 표이다.

도 6은 제 1 명령 세트에 대한 분기 선택자들의 제 1 예이다.

도 7은 제 2 명령 세트에 대한 분기 선택자들의 제 2 예이다.

도 8은 제 3 명령 세트에 대한 전형적인 분기 예측기들의 제 3 예이다.

도 9는 제 4 명령 세트에 대한 전형적인 분기 예측기들의 제 4 예이다.

도 10A 내지 10C는 본 발명의 일 실시예들에 따른 분기 선택자들과 복귀 선택자들의 세트들의 예들이다.

도 11은 본 발명 일 실시예에 따른 분기 예측 저장 엔트리의 일 실시예의 블록선도이다.

도 12는 연속적인 명령 바이트들의 그룹에 대응하는 분기 예측 저장 엔트리의 블록선도이다.

도 13은 도 1에 도시된 분기 예측 유닛의 일 실시예에 대한 예시적인 분기 선택자 및 복귀 선택자 엔코딩을 도시한 표이다.

도 14는 도 1에 도시된 마이크로프로세서의 일 실시예에 따른 마스크 생성을 도시한 흐름도이다.

도 15는 분기 선택자 갱신의 일 실시예를 도시한 흐름도이다.

도 16은 도 1에 도시된 마이크로프로세서에 따른 컴퓨터 시스템 일 실시예의 블록선도이다.

본 발명은 다양한 변형들 및 대안적인 형태들을 갖지만, 본원에서는 특정한 실시예들에 대해 예시적으로 설명할 것이다. 하지만, 주목할 사항으로서, 도면들 및 그에 대한 상세한 설명은 본 발명을 개시된 특정 형태로만 한정하지 않는다. 본 발명은 첨부된 청구항들에 의해 정의되는 본 발명의 원리 및 범위 내에 있는 모든 변형들, 등가들 및 대안들을 포괄한다.

실시예

도 1은 마이크로프로세서(10)의 일 실시예의 블록선도이다. 마이크로프로세서(10)는 프리페치/프리디코드 유닛(12), 분기 예측 유닛(14), 명령 캐시(16), 명령 정렬 유닛(18), 다수의 디코드 유닛들(20A-20C), 다수의 예약 스테이션(22A-22C), 다수의 기능 유닛들(24A-24C), 로드/저장 유닛(26), 데이터 캐시(28), 레지스터 파일(30), 재배열 버퍼(reorder buffer)(32) 및 MROM 유닛(34)을 포함한다. 본원에서 특정한 참조 번호와 그 옆에 표기된 문자를 이용하여 설명하는 요소들은 그 참조 번호 만을 이용하여 집합적으로 불려진다. 예를 들어, 디코드 유닛들(20A-20C)은 디코드 유닛들(20)로서 집합적으로 불려질 것이다.

프리페치/프리디코드 유닛(12)은 주 메모리 서브시스템(미도시)으로부터 명령들을 받도록 결합되며, 또한 명령 캐시(16) 및 분기 예측 유닛(14)에 결합된다. 유사하게, 분기 예측 유닛(14)은 명령 캐시(16)에 결합된다. 또한, 분기 예측 유닛(14)은 디코드 유닛들(20) 및 기능 유닛들(24)에 결합된다. 명령 캐시(16)는 또한 MROM 유닛(34) 및 명령 정렬 유닛(18)에 결합된다. 명령 정렬 유닛(18)은 또한 디코드 유닛들(20)에 결합된다. 각 디코드 유닛(20A-20C)은 로드/저장 유닛(26) 및 각각의 예약 스테이션들(22A-22C)에 결합된다. 예약 스테이션들(22A-22C)은 각각의 기능 유닛들(24A-24C)에 결합된다. 또한, 디코드 유닛들(20) 및 예약 스테이션들(22)은 레지스터 파일(30) 및 재배열 버퍼(32)에 결합된다. 기능 유닛들(24)은 로드/저장 유닛(26), 레지스터 파일(30) 및 재배열 버퍼(32)에 또한 결합된다. 데이터 캐시(28)는 로드/저장 유닛(26) 및 주 메모리 서브시스템에 결합된다. 마지막으로, MROM 유닛(34)은 디코드 유닛들(20)에 결합된다.

일반적으로 말해서, 분기 예측 유닛(14)은 캐시 라인 기반 분기 예측 메커니즘을 이용하여 분기 명령들을 예측한다. 다수의 분기 예측들은 각 캐시 라인에 저장될 수 있다. 또한, 분기 선택자는 캐시 라인 내의 각 바이트에 대해 저장된다. 특정 바

이트에 대한 분기 선택자는, 캐시 라인에 대해 저장될 수 있는 분기 예측들중 어떤 것이, 특정 바이트를 페치하는 명령 페치 어드레스에 대해 적절한 분기 예측인지를 나타낸다. 또한, 1개 이상의 복귀 선택자 비트들이 각 분기 선택자에 대해 저장될 수 있다. 복귀 선택자 비트들은, 분기 선택자가 복귀 명령 분기 예측과 관련되는지, 아니면 비복귀 명령 분기 예측과 관련되는 지를 나타낸다. 적절한 분기 예측은, 특정 바이트에 후속하는 캐시 라인 내의 테이큰인 것으로 예측되는 제 1 분기 명령에 대한 분기 예측이다. 본원에서 이용되는 "후속" 및 "이전"이라는 용어들은 캐시 라인 내에서의 바이트들의 순서를 말한다. 제 2 바이트가 저장된 메모리 어드레스보다 수치적으로 작은 메모리 어드레스에 저장된 바이트는 제 2 바이트보다 선행한다. 반대로, 제 2 바이트가 저장된 메모리 어드레스 보다 수치적으로 큰 메모리 어드레스에 저장된 바이트는 제 2 바이트 보다 후속한다. 유사하게, 프로그램을 형성하는 명령들의 시퀀스를 통해 한 번에 하나씩 진행될 때, 제 1 명령이 제 2 명령 이전에 있으면, 제 1 명령은 제 2 명령에 선행한다.

일 실시예에서, 마이크로프로세서(10)는 명령 세트가 가변 바이트 길이 명령 세트인 마이크로프로세서 구조(예를 들어, x86 마이크로프로세서 구조)를 이용한다. 가변 바이트 길이 명령 세트가 이용될 때에는, 캐시 라인 내의 어떠한 바이트가, 소정의 페치 어드레스에 의해 페치될 제 1 바이트로서 특정될 수 있다. 예를 들어, 분기 명령은 캐시 라인 내의 바이트 위치 2에 목표 어드레스를 가질 수 있다. 이러한 경우, 바이트 위치 0 및 1의 바이트들은 현재의 캐시 액세스 동안 페치되지 않는다. 또한, 제 1 바이트에 후속하는 테이큰인 것으로 예측된 분기에 후속하는 바이트들은 현재의 캐시 액세스 동안 페치되지 않는다. 분기 선택자들은 각 바이트에 대해 저장되기 때문에, 테이큰인 것으로 예측된 분기에 대한 분기 예측은, 캐시 라인으로부터 페치되는 제 1 바이트의 분기 선택자를 선택함으로써 찾아낼 수 있다. 분기 선택자를 이용하여 적절한 분기 예측을 선택한 다음, 이를 명령 캐시(16)의 명령 페치 논리에 제공한다. 계속되는 클럭 주기 동안, 분기 예측은 페치 어드레스로서 이용된다. 유익하게는, 페치되고 있는 제 1 바이트의 바이트 위치와 테이큰인 것으로 예측된 분기 명령들의 바이트 위치들을 비교하는 과정은, 페치 어드레스에 응답하는 분기 예측의 발생으로부터 생략된다. 이에 따라, 분기 예측을 형성하는 데에 필요한 시간이 단축되어, 분기 예측 메커니즘은 여전히 단일 클럭 분기 예측을 제공하면서 보다 높은 클럭 주파수들(즉, 보다 짧은 클럭 주기)에서 실행될 수 있게 된다.

주목할 사항으로서, "캐시 라인"이라는 용어가 상기에서 이용되었지만, 명령 캐시(16)의 일부 실시예들은 소정의 클럭 주기 동안 그 출력에 전체 캐시 라인을 제공하지 않을 수도 있다. 예를 들어, 일 실시예에서, 명령 캐시(16)는 32 바이트 캐시 라인들로 구성된다. 그러나, 소정의 클럭 주기 동안에는 단지 16 바이트(캐시 라인의 상위 절반 또는 하위 절반) 만이 페치된다. 분기 예측 저장 위치들 및 분기 선택자들이, 페치되고 있는 캐시 라인의 일부에 할당된다. 본원에서 이용되는 "연속적인 명령 바이트들의 그룹"이란 용어는, 페치 어드레스에 응답하여 특정 클럭 주기 동안 명령 캐시에 의해 제공되는 명령 바이트들을 말한다. 많은 실시예들에 따르면, 연속적인 명령 바이트들의 그룹은 캐시 라인의 일부 또는 전체 캐시 라인이 될 수 있다. 연속적인 명령 바이트들의 그룹이 캐시 라인의 일부일 때, 이는 여전히 캐시 라인의 정렬된 일부이다. 예를 들어, 연속적인 명령 바이트들의 그룹이 캐시 라인의 절반인 경우, 이는 캐시 라인의 상부 절반 또는 하부 절반이다. 다수의 분기 예측 저장 위치들은 연속적인 명령 바이트들의 각 그룹에 할당되며, 그리고 분기 선택자들은 그 그룹에 관련된 분기 예측 저장 위치들중 하나를 나타낸다. 또한, 분기 선택자들은 복귀 스택 구조로부터 복귀 스택 어드레스를 나타내거나, 또는 연속적인 명령 바이트들의 그룹 내의 대응하는 바이트 및 마지막 바이트 사이에 어떠한 분기 명령도 없는 경우에는 순차 어드레스를 나타낸다.

명령 캐시(16)는 명령들을 저장하기 위한 고속 캐시 메모리이다. 명령들은 명령 캐시(16)로부터 페치되고, 디코드 유닛들(20)로 디스페치한다. 일 실시예에서, 명령 캐시(16)는 32 바이트 라인들(1 바이트는 8개의 이진 비트로 구성됨)을 갖는 4 방향 세트 어소시에이티브 구조(4 way set associative structure)로 최대 32 킬로바이트의 명령들을 저장하도록 구성된다. 명령 캐시(16)는 또한 명령 캐시로의 액세스 시간을 단축하기 위해 방향(way) 예측 방식을 이용할 수 있다. 명령들의 각 라인들을 특정하는 태그들을 액세스하고, 방향을 선택하기 위해 이 태그들을 페치 어드레스와 비교하는 대신, 명령 캐시(16)는 액세스되는 방향을 예측한다. 이러한 방식으로, 방향은 명령 저장부를 액세스하기 전에 선택된다. 명령 캐시(16)의 액세스 시간은 직접 맵핑된(direct-mapped) 캐시와 유사하다. 태그 비교가 실행되고, 방향 예측이 부정확하면, 정확한 명령들이 페치되고 부정확한 명령들은 버려진다. 주목할 사항으로서, 명령 캐시(16)는 완전 어소시에이티브(fully associative), 세트 어소시에이티브 또는 직접 맵핑된 구성으로서 실시될 수 있다.

명령들은 주 메모리로부터 페치되어, 프리페치/프리디코드 유닛(12)에 의해 명령 캐시(16)에 저장된다. 명령들은, 프리페치 방식에 따라 명령 캐시(16)로부터의 명령 요구 이전에 프리페치될 수 있다. 다양한 프리페치 방식들이 프리페치/프리디코드 유닛(12)에 의해 이용될 수 있다. 프리페치/프리디코드 유닛(12)이 명령들을 주 메모리로부터 명령 캐시(16)로 전달할 때, 프리페치/프리디코드 유닛(12)은 명령들의 각 바이트에 대해 3개의 프리디코드 비트들, 즉 시작 비트, 끝 비트 및 기능 비트를 발생시킨다. 프리디코드 비트들은 각 명령의 경계들을 표시하는 태그들을 형성한다. 프리디코드 태그들은 또한, 주어진 명령이 디코드 유닛들(20)에서 직접 디코드될 수 있는 지에 대한 정보, 또는 MROM 유닛(34)에 의해 제어되는 마

이크로코드 절차를 호출함으로써, 명령이 실행되는 지에 대한 정보와 같은 부가적인 정보를 전달할 수 있다. 이에 대해서는 하기에서 보다 상세히 설명된다. 또한, 프리페치/프리디코드 유닛(12)은, 분기 명령들을 검출하고, 분기 명령들에 대응하는 분기 예측 정보들을 분기 예측 유닛(14)에 저장하도록 구성된다.

다음으로, x86 명령 세트를 사용하는 마이크로프로세서(10) 일 실시예에 대한 프리디코드 태그들의 엔코딩의 일례를 설명한다. 주어진 바이트가 명령의 제 1바이트이면, 그 바이트에 대한 시작 비트가 세트된다. 바이트가 명령의 마지막 바이트이면, 그 바이트에 대한 끝 비트가 세트된다. 디코드 유닛(20)에 의해 직접 디코드될 수 있는 명령들은 "빠른 경로(fast path)" 명령들로서 명명된다. 일 실시예에 따르면, 나머지 x86 명령들은 MROM 명령로서 명명된다. 빠른 경로 명령들에 있어서, 명령에 포함된 각 접두 바이트(prefix byte)에 대해 기능 비트가 세트되는바, 이는 다른 바이트들에 대해서는 클리어된다. 대안적으로, MROM 명령들에 있어서, 기능 비트는 각 접두 바이트에 대해 클리어되며, 다른 바이트들에 대해 세트된다. 명령의 종류는 종료 바이트에 대응하는 기능 비트를 검사함으로써 결정된다. 기능 비트가 클리어되는 경우, 그 명령은 빠른 경로 명령이다. 반대로, 기능 비트가 세트되면, 그 명령은 MROM 명령이다. 이로써, 명령의 연산코드(opcode)는, 디코드 유닛(20)에 의해 직접 디코드될 수 있는 명령 내에서, 명령의 제 1 클리어 기능 비트와 관련된 바이트로서 위치될 수 있다. 예를 들어, 2개의 접두 바이트들, 즉 Mod R/M 바이트 및 SIB 바이트를 포함하는 빠른 경로 명령은 다음과 같은 시작 비트, 끝 비트 및 기능 비트를 갖는다.

시작 비트들 10000

끝 비트들 00001

기능 비트들 11000

MROM 명령들은 디코드 유닛들(20)에 의해 디코드하기에 너무 복잡한 것으로 판단되는 명령들이다. MROM 명령들은 MROM 유닛(34)을 호출함으로써 실행된다. 보다 구체적으로, MROM 명령을 접하면, MROM 유닛(34)은 상기 명령을 분석한 다음, 이를 규정된 빠른 경로 명령들의 서브셋(subset)으로 발행하여, 원하는 동작을 실시한다. MROM 유닛(34)은 빠른 경로 명령들의 서브셋을 디코드 유닛들(20)로 디스페치한다. 빠른 경로 명령들로서 분류된 예시적인 x86 명령들의 리스트가 이하 제공될 것이다.

마이크로프로세서(10)는 조건 분기 명령들에 후속하는 명령들을 추론적으로 페치하기 위해 분기 예측을 이용한다. 분기 예측 유닛(14)은 분기 예측 동작들을 수행하기 위해 포함된다. 일 실시예에서, 명령 캐시(16)의 각 캐시 라인의 각 16 바이트 부분에 관련하여, 최대 3개의 분기 목표 어드레스들이 저장된다. 프리페치/프리디코드 유닛(12)은 특정 라인이 프리디코드될 때 초기 분기 목표들을 결정한다. 캐시 라인 내에서의 명령들의 실행에 의해, 캐시 라인에 대응하는 분기 목표들에 대한 후속 갱신이 이루어진다. 명령 캐시(16)는 페치되는 명령 어드레스의 표시를 제공하며, 이에 따라 분기 예측 유닛(14)은 분기 예측을 형성하기 위해 어떤 분기 목표 어드레스를 선택할 것인 지를 결정할 수 있다. 디코드 유닛들(20) 및 기능 유닛들(24)은 분기 예측 유닛(14)에 갱신 정보를 제공한다. 분기 예측 유닛(14)이 캐시 라인의 16 바이트 부분 마다 3개의 목표들을 저장하기 때문에, 라인 내의 일부 분기 명령들은 분기 예측 유닛(14)에 저장되지 못할 수 있다. 디코드 유닛들(20)은 분기 예측 유닛(14)에 의해 예측되지 못했던 분기 명령들을 검출한다. 기능 유닛들(24)은 분기 명령을 실행하고, 예측된 분기 방향이 부정확한 지를 결정한다. 분기 방향은 "테이큰"인 것으로 될 수 있는바, 이 경우 후속 명령들은 분기 명령의 목표 어드레스로부터 페치된다. 반대로, 분기 방향은 "낫테이큰"인 것으로 될 수 있는바, 이 경우 후속 명령들은 분기 명령에 연속하는 메모리 위치들로부터 페치된다. 오예측된 분기 명령이 검출되면, 이 오예측된 분기에 후속하는 명령들은 마이크로프로세서(10)의 많은 유닛들로부터 버려진다. 다수의 적절한 분기 예측 알고리즘들이 분기 예측 유닛(14)으로서 이용될 수 있다.

명령 캐시(16)로부터 페치되는 명령들은 명령 정렬 유닛(18)으로 전달된다. 명령들이 명령 캐시(16)로부터 페치되면, 대응하는 프리디코드 데이터가 주사(scan)되어, 페치되고 있는 명령에 대한 정보를 명령 정렬 유닛(18)(및 MROM 유닛(34))에 제공한다. 명령 알고리즘 유닛(18)은 주사 데이터를 이용하여, 명령을 디코드 유닛들(20)들 각각에 정렬시킨다. 일 실시예에서, 명령 정렬 유닛(18)은 8개의 명령 바이트들의 3개의 세트들로부터의 명령을 디코드 유닛들(20)에 정렬시킨다. 명령들은 8의 명령 바이트들의 각 세트로부터 독립적으로 선택되어, 예비 발행 위치(preliminary issue position)들에 놓여진다. 이후, 예비 발행 위치들은 디코드 유닛들(20)에 대응하는 정렬된 발행 위치들의 세트에 합쳐지며, 이에 따라 정렬된 발행 위치들은 프로그램 순서로 예비 발행 위치 내의 다른 명령에 선행하는 3개의 명령을 포함한다. 디코드 유닛(20A)은 (프로그램 순서로) 디코드 유닛들(20B 및 20C)에 의해 동시에 수신된 명령들에 선행하는 명령을 수신한다. 유사하게, 디코드 유닛(20B)은 프로그램 순서로 디코드 유닛(20C)에 동시에 수신된 명령에 선행하는 명령을 수신한다.

디코드 유닛들(20)은 명령 정렬 유닛(18)으로부터 수신한 명령들을 디코드하도록 구성된다. 레지스터 오퍼랜드 정보가 검출되어, 레지스터 파일(30) 및 재배열 버퍼(32)로 전송된다. 또한, 명령들이 1개 이상의 메모리 연산들의 수행을 요구하는 경우, 디코드 유닛들(20)은 메모리 연산들을 로드/저장 유닛(26)으로 디스페치한다. 각 명령은 기능 유닛들(24)을 위한 제어값들의 세트로 디코드되며, 이러한 제어값들은 오퍼랜드 어드레스 정보, 및 그 명령에 포함될 수도 있는 변위 또는 즉시 데이터와 함께 예약 스테이션들(22)로 디스페치된다.

마이크로프로세서(10)는 순서를 벗어난(out of order) 실행을 지원하며, 이에 따라 재배열 버퍼(32)를 이용하여, 레지스터 읽기 및 쓰기 동작들을 위한 최초 프로그램 순서의 트랙을 유지하고, 레지스터 재명명(renaming)을 실시하며, 추론적인 명령 실행 및 분기 오예측 복구를 가능하게 하고, 그리고 정밀한 예외를 용이하게 한다. 재배열 버퍼(32) 내의 임시 저장 위치는 레지스터의 갱신을 포함하는 명령의 디코드시 지정되어, 추론적인 레지스터의 상태를 저장한다. 분기 예측이 정확하지 않으면, 오예측된 경로를 따라 추론적으로 실행된 명령들의 결과는, 이들이 레지스터 파일(30)에 쓰여지기 전에 버퍼 내에서 무효화될 수 있다. 유사하게, 특정 명령이 예외를 야기시키면, 특정 명령에 후속하는 명령들은 버려질 수 있다. 이런 방식으로, 예외들은 "정확하다"(즉, 예외를 야기시키는 특정 명령에 후속하는 명령들의 완료가 예외에 선행하지 않는다). 주목할 사항으로서, 특정 명령은 프로그램 순서로 이 특정 명령을 선행하는 명령들 이전에 실행되는 경우, 추론적으로 실행된다. 선행하는 명령들은 분기 명령 또는 예외 야기 명령이 될 수 있는바, 이 경우 추론적인 결과들은 재배열 버퍼(32)에 의해 버려질 수 있다.

디코드 유닛들(20)의 출력에 제공되는 명령 제어값들 및 즉시 또는 변위 데이터는 각각의 예약 스테이션들(22)로 직접 전달된다. 일 실시예에서, 각 예약 스테이션(22)은 대응하는 기능 유닛으로의 발행을 기다리는 최대 3개의 계류중인(pending) 명령들에 대해, 명령 정보(즉, 오퍼랜드값들, 오퍼랜드 태그들 그리고/또는 즉시 데이터 뿐 아니라 명령 제어값들)을 보유할 수 있다. 주목할 사항으로서, 도 1의 실시예에서, 각 예약 스테이션(22)은 전용 기능 유닛(24)에 결합된다. 따라서, 3개의 전용 "발행 위치들"은 예약 스테이션들(22) 및 기능 유닛들(24)에 의해 형성된다. 다시 말해, 발행 위치 0은 예약 스테이션(22A) 및 기능 유닛(24A)에 의해 형성된다. 예약 스테이션(22A)으로 정렬되어 디스페치되는 명령들은 기능 유닛(24A)에 의해 실행된다. 유사하게, 발행 위치 1은 예약 스테이션(22B) 및 기능 유닛(24B)에 의해 형성되고, 발행 위치 2는 예약 스테이션(22C) 및 기능 유닛(24C)에 의해 형성된다.

특정 명령을 디코드할 때, 요구되는 오퍼랜드가 레지스터 위치이면, 레지스터 어드레스 정보는 재배열 버퍼(32) 및 레지스터 파일(30)로 동시에 전달된다. 당업자라면, x86 레지스터 파일이 8개의 32 비트 리얼 레지스터들(real register)(즉, 전형적으로 EAX, EBX, ECX, EDX, EBP, ESI 및 ESP라 일컬어짐)을 포함한다는 것을 알 수 있을 것이다. x86 마이크로프로세서 구조를 이용하는 마이크로프로세서(10)의 일 실시예에서, 레지스터 파일(30)은 32 비트 리얼 레지스터들 각각에 대한 저장 위치들을 포함한다. 레지스터 파일(30) 내에는 MROM 유닛(34)에 의해 이용하기 위한 부가적인 저장 위치들이 포함될 수 있다. 재배열 버퍼(32)는 이러한 레지스터들의 내용을 변경하는 결과에 대한 임시 저장 위치들을 포함하며, 이에 의해 순서를 벗어난 실행이 가능해진다. 재배열 버퍼(32)의 임시 저장 위치는, 디코드시 리얼 레지스터들중 1개의 내용을 변경하는 것으로 판단되는 각 명령에 대해 지정된다. 이에 따라, 특정 프로그램을 실행하는 동안 많은 지점들에서, 재배열 버퍼(32)는 주어진 레지스터의 추론적으로 실행된 내용을 포함하는 1개 이상의 위치들을 가질 수 있다. 소정 명령의 디코드에 이어서, 재배열 버퍼(32)가 그 소정 명령의 오퍼랜드로서 이용되는 레지스터에 할당된 이전 위치 또는 위치들을 갖는 것으로 판단되면, 재배열 버퍼(32)는 대응하는 예약 스테이션에, 1) 가장 최근에 할당된 위치의 값, 또는 2) 이전 명령을 궁극적으로 실행하게 될 기능 유닛이 아직 상기 값을 생성하지 않은 경우에는, 가장 최근에 할당된 위치에 대한 태그를 전달한다. 재배열 버퍼(32)가 소정의 레지스터에 지정된 위치를 갖는 경우, 오퍼랜드값(또는 재배열 버퍼 태그)은 레지스터 파일(30)이 아닌 재배열 버퍼(32)로부터 제공된다. 재배열 버퍼(32)에 요구되는 레지스터에 대해 어떠한 위치도 지정되어 있지 않은 경우, 상기 값은 레지스터 파일(30)로부터 직접 제공된다. 오퍼랜드가 메모리에 대응하는 경우, 오퍼랜드값은 로드/저장 유닛(26)을 통해 예약 스테이션에 제공된다.

일 특정 실시예에서, 재배열 버퍼(32)는 동시에 디코드된 명령들을 한 단위로 저장하여 조작하도록 구성된다. 이러한 구성은 "라인 지향(line-oriented)"이라 불려진다. 몇 개의 명령들을 함께 조작함으로써, 재배열 버퍼(32) 내에서 이용되는 하드웨어를 단순하게 할 수 있다. 예를 들어, 본 실시예에 포함되는 라인 지향 재배열 버퍼(32)는, 디코드 유닛들(20)이 1개 이상의 명령을 디스페치할 때 마다, 3개의 명령들(각 디코드 유닛(20)에서 1개씩)에 관련된 명령 정보에 대해 충분한 저장부를 할당한다. 대조적으로, 종래의 재배열 버퍼들에서는, 실제로 디스페치되는 명령들의 수에 따라, 가변량의 저장부가 할당되었다. 가변량의 저장부를 할당하기 위해서는, 비교적 많은 수의 논리 게이트들이 필요하다. 동시에 디코드되는 명령들 각각이 실행될 때, 명령의 결과들은 레지스터 파일(30)에 동시에 저장된다. 이후, 이 저장부는 동시에 디코드되는 명령들의 다른 세트에 할당하기 위해 비워진다. 부가적으로, 제어 논리는 몇 개의 동시에 디코드되는 명령들 전체에 공용되기 때문에, 명령 마다 이용되는 제어 논리 회로의 양은 감소된다. 특정 명령을 특정하는 재배열 버퍼 태그는 2개의 필드들, 즉 라인 태그 및 오프셋 태그로 분할될 수 있다. 라인 태그는 특정 명령을 포함하는 동시 디코드된 명령들의 세트를 특정하고,

오프셋 태그는 세트 내의 어떤 명령이 특정 명령에 대응하는 지를 특정한다. 주목할 사항으로서, 레지스터 파일(30)에 명령 결과를 저장하고 대응하는 저장부를 비우는 것은 명령의 "폐기(retiring)"로 일컬어진다. 또한, 주목할 사항으로서, 마이크로프로세서(10)의 다양한 실시예들은 어떠한 재배열 버퍼라도 이용할 수 있다.

상기 설명한 바와 같이, 예약 스테이션들(22)은 해당하는 기능 유닛(24)에 의해 명령들이 실행되기 전까지 명령들을 저장한다. 명령들은, (i) 명령의 오퍼랜드들이 제공되고, (ii) 동일한 예약 스테이션들(22A-22C) 내에 있으며 프로그램 순서에 있어서 상기 명령에 선행하는 명령들에 대한 오퍼랜드들이 아직 제공되지 않은 경우, 실행을 위해 선택된다. 주목할 사항으로서, 기능 유닛들(24)중 하나에 의해 명령이 실행되면, 명령의 결과는 재배열 버퍼(32)를 갱신하기 위해 전송됨과 동시에, 그 결과를 기다리는 어떠한 예약 스테이션들(22)로 직접 전송된다(이 기술은 일반적으로 "결과 전송(result forwarding)"이라 불려진다). 관련된 결과가 전송되는 클럭 주기 동안, 실행하기 위한 명령이 선택되어 기능 유닛들(24A-24C)로 전송될 수 있다. 이 경우, 예약 스테이션들(22)은 전송된 결과를 기능 유닛(24)으로 보낸다.

일 실시예에서, 상기 각 기능 유닛들(24)은 정수의 덧셈 및 뺄셈 산술 연산 뿐 아니라, 시프트, 로테이트, 논리 연산들 및 분기 연산들을 수행하도록 구성된다. 이러한 연산들은 디코드 유닛들(20)에 의해 특정 명령에 대해 디코드된 제어값들에 응답하여 수행된다. 주목할 사항으로서 부동 소수점 유닛(미도시)을 이용하여 부동 소수점 연산에 대처할 수 있다. 부동 소수점 유닛은 코프로세서(coprocessor)로서 동작할 수 있는바, 이는 MROM 유닛(34)에서 명령들을 받은 다음, 이 명령들을 완성하기 위해 재배열 버퍼(32)와 통신한다. 또한, 기능 유닛들(24)은 로드/저장 유닛(26)에 의해 실행되는 메모리 로드 및 저장 연산들에 대한 어드레스 생성을 실행하도록 구성될 수 있다.

기능 유닛들(24) 각각은 또한 조건 분기 명령들의 실행에 대한 정보를 분기 예측 유닛(14)에 제공한다. 분기 예측이 부정확했을 경우, 분기 예측 유닛(14)은 명령 처리 파이프라인으로 들어왔던 오예측 분기에 후속하는 명령들을 방출하고, 명령 캐시(16) 또는 주 메모리로부터 필요한 명령이 페치되게 한다. 주목할 사항으로서, 이러한 상황들에서, 추론적으로 실행되고 로드/저장 유닛(26) 및 재배열 버퍼(32)에 임시적으로 저장된 것들을 포함하여, 오예측된 분기 명령 다음에 발생하는 최초 프로그램 순서의 명령 결과는 버려진다.

기능 유닛들(24)에 의해 생성되는 결과들은, 레지스터 값이 갱신되는 경우에는 재배열 버퍼(32)로 보내지고, 메모리 위치의 내용들이 변하는 경우에는 로드/저장 유닛(26)으로 보내진다. 결과가 레지스터에 저장되는 경우, 재배열 버퍼(32)는 명령이 디코드되었을 때 레지스터값에 대해 지정된 위치에 그 결과를 저장한다. 기능 유닛들(24) 및 로드/저장 유닛(26)으로부터의 결과들을 전송하기 위한 다수의 결과 버스들(38)이 포함된다. 결과 버스들(38)은 생성된 결과 뿐 아니라, 실행되는 명령을 특정하는 재배열 버퍼 태그를 전달한다.

로드/저장 유닛(26)은 기능 유닛들(24)과 데이터 캐시(28) 간의 인터페이스를 제공한다. 일 실시예에서, 로드/저장 유닛(26)은 데이터를 위한 8개의 저장 장소들, 및 절박한 로드 또는 저장을 위한 어드레스 정보를 갖는 로드/저장 버퍼를 갖추도록 구성된다. 디코드 유닛들(20)은 로드/저장 유닛(26)에 대한 액세스를 조정한다. 버퍼가 가득차면, 디코드 유닛은 로드/저장 유닛(26)이 절박한 로드 또는 저장 요구 정보를 위한 빈 곳을 가질 때 까지 기다려야 한다. 로드/저장 유닛(26)은 또한 절박한 저장 메모리 연산들에 대한 로드 메모리 연산들의 종속성 검사를 실행하여, 데이터의 코히런시가 유지될 수 있게 한다. 메모리 연산은 마이크로프로세서(10)와 주 메모리 서브시스템 간의 데이터 전송이다. 메모리 연산들은 메모리에 저장된 오퍼랜드를 이용하는 명령의 결과, 또는 다른 어떠한 연산도 일으키지 않으면서 데이터 전송만을 일으키는 로드/저장 명령의 결과가 될 수 있다. 또한, 로드/저장 유닛(26)은 x86 마이크로프로세서 구조에 의해 정의되는 어드레스 변환 메커니즘에 관련된 세그먼트 레지스터들 및 다른 레지스터들과 같은 특별한 레지스터들을 위한 특별한 레지스터 저장부를 포함할 수 있다.

일 실시예에서, 로드/저장 유닛(26)은 로드 메모리 연산들을 추론적으로 실행하도록 구성된다. 저장 메모리 연산들은 프로그램 순서로 실행되지만, 예측된 방향(way)으로 추론적으로 저장될 수 있다. 예측된 방향이 정확하지 않은 경우, 저장 메모리 연산에 선행하는 데이터는 이후 예측된 방향으로 복구되고, 저장 메모리 연산은 정확한 방향으로 수행된다. 다른 실시예에서는, 저장들 또한 추론적으로 실행될 수 있다. 추론적으로 실행되는 저장들은 갱신 이전의 캐시 라인의 카피와 함께 저장 버퍼 내에 놓여진다. 추론적으로 실행되는 저장이 분기 오예측 또는 예외로 인해 이후 버려지는 경우, 캐시 라인은 버퍼에 저장된 값으로 복구될 수 있다. 주목할 사항으로서, 로드/저장 유닛(26)은, 어떠한 추론적인 실행도 포함하지 않는, 어떠한 양의 추론적인 실행을 수행하도록 구성될 수 있다.

데이터 캐시(28)는 로드/저장 유닛(26)과 주 메모리 서브시스템 간에 전송되는 임시 저장 데이터를 제공하는 고속 캐시 메모리이다. 일 실시예에서, 데이터 캐시(28)는 8 방향 세트 어소시에이티브 구조로 최대 16 킬로바이트의 데이터를 저장하는 용량을 갖는다. 명령 캐시(16)와 유사하게, 데이터 캐시(28)는 방향 예측 메커니즘을 이용할 수 있다. 데이터 캐시(28)는, 세트 어소시에이티브 구성을 포함하는, 다수의 특정한 메모리 구성들로 실시될 수 있다.

x86 마이크로프로세서 구조를 이용하는 마이크로프로세서(10)의 일 특정 실시예에서, 명령 캐시(16) 및 데이터 캐시(28)는 선형으로(linearly) 어드레스된다. 이러한 선형 어드레스는 명령에 의해 특정되는 오프셋, 및 x86 어드레스 변환 메커니즘의 세그먼트 부분에 의해 특정되는 베이스 어드레스에 의해 형성된다. 선형 어드레스들은 주 메모리를 액세스하기 위한 물리적인 어드레스들로 선택적으로 변환될 수 있다. 선형 어드레스들의 물리적인 어드레스들로의 변환은 x86 어드레스 변환 메커니즘의 페이징 부분에 의해 지정된다. 주목할 사항으로서, 선형으로 어드레스되는 캐시는 선형 어드레스 태그들을 저장한다. 물리적 태그들의 세트(미도시)는, 선형 어드레스들을 물리적인 어드레스들로 맵핑하고, 앨리어스(alias)들을 검출하는 데에 이용될 수 있다. 또한, 물리적인 태그 블럭이 선형 어드레스의 물리적인 어드레스로의 변환을 수행할 수 있다.

다음으로, 도 2는 디코드 유닛들(20B 및 20C)의 일 실시예의 블록선도를 도시한다. 각 디코드 유닛(20)은 명령 정렬 유닛(18)으로부터 명령을 받는다. 또한, MROM 유닛(34)은 특정한 MROM 명령에 대응하는 빠른 경로 명령들을 디스페치하기 위해 각 디코드 유닛(20)에 결합된다. 디코드 유닛(20B)은 초기 디코드 유닛(40B), 멀티플렉서(42B) 및 연산코드 디코드 유닛(44B)을 포함한다. 유사하게, 디코드 유닛(20C)은 초기 디코드 유닛(40C), 멀티플렉서(42C) 및 연산코드 디코드 유닛(44C)을 포함한다.

x86 명령 세트의 특정 명령들은 상당히 복잡하며 빈번하게 이용된다. 마이크로프로세서(10)의 일 실시예에서, 이러한 명령들은 특정한 기능 유닛(24A-24C) 내에 포함된 하드웨어가 수행하도록 구성된 것 보다 훨씬 복잡한 연산들을 포함한다. 이러한 명령들은 "이중 디스페치" 명령로서 일컬어지는 특별한 형태의 MROM 명령로 분류된다. 이러한 명령들은 한 쌍의 연산코드 디코드 유닛들(44)로 디스페치된다. 주목할 사항으로서, 연산코드 디코드 유닛들(44)은 각각의 예약 스테이션들(22)에 결합된다. 연산코드 디코드 유닛들(44A-44C) 각각은 대응하는 예약 스테이션(22A-22C) 및 기능 유닛(24A-24C)과 함께 발행 위치를 형성한다. 명령들은 연산코드 디코드 유닛(44)으로부터 대응하는 예약 스테이션(22)으로 전달되고, 또한 대응하는 기능 유닛(24)으로 전달된다.

멀티플렉서(42B)는 MROM 유닛(34)에 의해 제공되는 명령과 초기 디코드 유닛(40B)에 의해 제공되는 명령중 어떠한 것을 선택하기 위해 포함된다. MROM 유닛(34)이 명령들을 디스페치하는 동안, 멀티플렉서(42B)는 MROM 유닛(34)에 의해 제공되는 명령들을 선택한다. 다른 시간들 동안, 멀티플렉서(42B)는 초기 디코드 유닛(40B)에 의해 제공되는 명령들을 선택한다. 유사하게, 멀티플렉서(42C)는 MROM유닛(34)에 의해 제공되는 명령과 초기 디코드 유닛(40B)에 의해 제공되는 명령과, 그리고 초기 디코드 유닛(40C)에 의해 제공되는 명령중 어떠한 것을 선택한다. MROM 유닛(34)이 명령들을 디스페치하는 동안, MROM 유닛(34)으로부터의 명령들이 선택된다. 디코드 유닛(20A)(미도시) 내의 초기 디코드 유닛이 이중 디스페치 명령을 검출하는 동안, 초기 디코드 유닛(40B)으로부터의 명령이 멀티플렉서(42C)에 의해 선택된다. 그 외의 경우들에는, 초기 디코드 유닛(40C)으로부터의 명령이 선택된다. 초기 디코드 유닛(40B)으로부터 연산코드 디코드 유닛(44C)으로 명령을 선택하게 되면, 디코드 유닛(20B)에 의해 디코드되는 빠른 경로 명령들이, 디코드 유닛(20A)에 의해 디코드된 이중 디스페치 명령과 동시에 디스페치될 수 있다.

x86 명령 세트를 이용하는 일 실시예에 따르면, 초기 디코드 유닛들(40)은 다음 연산을 수행한다.

(i) 명령의 접두 바이트들을 엔코드된 접두 바이트로 병합한다.

(ii) 분기 예측동안 검출되지 않았던 무조건 분기 명령들(이는 무조건 점프, CALL 및 복귀를 포함할 수 있다)을 디코드한다.

(iii) 소스 및 목적지 플래그들을 디코드한다.

(iv) 레지스터 오퍼랜드들인 소스 및 목적지 오퍼랜드들을 디코드하고, 오퍼랜드 크기 정보를 발생시킨다.

(v) 변위 및 즉시 데이터가 연산코드 디코드 유닛으로 전송될 수 있도록, 변위 그리고/또는 즉시 크기를 결정한다.

연산코드 디코드 유닛들(44)은 명령의 연산코드를 디코드하여, 기능 유닛(24)을 위한 제어값들을 생성하도록 구성된다. 변위 및 즉시 데이터는 제어값과 함께 예약 스테이션(22)으로 전송된다.

초기 디코드 유닛들(40)이 오퍼랜드들을 검출하기 때문에, 멀티플렉서들(42)의 출력들은 레지스터 파일(30) 및 재배열 버퍼(32)로 전송된다. 이에 의해, 오퍼랜드 값들 또는 태그들이 예약 스테이션들(22)로 전송될 수 있게 된다. 또한, 메모리 오퍼랜드들은 초기 디코드 유닛들(40)에 의해 검출된다. 따라서, 멀티플렉서들(42)의 출력들은 로드/저장 유닛(26)으로 전송된다. 메모리 오퍼랜드들을 갖는 명령에 대응하는 메모리 연산들은 로드/저장 유닛(26)에 의해 저장된다.

다음으로, 도 3은 예시적인 연속적인 명령 바이트들의 그룹(50)과, 대응하는 분기 선택자들(52) 및 복귀 선택자들(53)의 세트의 블록선도를 도시한다. 도 3에서, 명령 내의 각 바이트는 짧은 수직선(예를 들어, 54)으로 도시된다. 또한, 그룹(50)에서 명령들을 분리하는 수직선들은 바이트들의 경계를 나타낸다(예를 들어, 56). 도 3에 도시된 명령들은 가변 길이를 가지며, 이에 따라 도 3에 도시된 명령들을 포함하는 명령 세트는 가변 바이트 길이 명령 세트이다. 다시 말해, 가변 바이트 길이 명령 세트의 제 1 명령은, 명령 세트 내의 제 2 명령에 의해 점유되는 제 2 수의 바이트들과 다른 제 1 수의 바이트들을 점유할 수 있다. 다른 명령 세트들은 고정 길이를 가질 수 있으며, 이에 따라 명령 세트 내의 각 명령은 다른 각 명령과 동일한 수의 바이트들을 점유한다.

도 3에 도시된 바와 같이, 그룹(50)은 비분기 명령들(IN0-IN5)을 포함한다. 명령들(IN0, IN3, IN4 및 IN5)은 2바이트 명령들이다. 명령(IN1)은 1바이트 명령이고, 명령(IN2)은 3바이트 명령이다. 테이큰인 것으로 예측된 2개의 분기 명령들(PB0 및 PB1)이 또한 도시되는바, 이들은 각각 2바이트를 점유하는 것으로 도시되어 있다. 주목할 사항으로서, 비분기 명령들 및 분기 명령들은 많은 수의 바이트들을 점유할 수 있다.

테이큰인 것으로 예측된 분기들(PB0 및 PB1) 각각의 마지막 바이트는 그룹(50)을 3개의 영역들, 즉 제 1 영역(58), 제 2 영역(60) 및 제 3 영역(62)으로 분할한다. 그룹(50)을 특정하는 페치 어드레스가 존재하고, 그룹 내의 페치 어드레스의 오프셋이 제 1 영역(58) 내의 바이트 위치를 특정하는 경우, 테이큰인 것으로 예측된 최초의 분기 명령은 PB0이고, 이에 따라 PB0에 대한 분기 예측은 분기 예측 메커니즘에 의해 선택된다. 유사하게, 페치 어드레스의 오프셋이 제 2 영역(60) 내의 바이트를 특정하는 경우, 적절한 분기 예측은 PB1에 대한 분기 예측이다. 마지막으로, 페치 어드레스의 오프셋이 제 3 영역(62) 내의 바이트를 특정하는 경우, 명령 바이트들의 그룹 내의 특정된 바이트에 후속해서는 테이큰인 것으로 예측되는 어떠한 분기 명령도 없다. 따라서, 제 3 영역(62)에 대한 분기 예측은 순차적이다. 순차 어드레스는 주 메모리 내의 그룹(50) 바로 다음에 오는 명령 바이트들의 그룹을 특정한다.

본원에서 이용되는 어드레스의 오프셋은 그 어드레스의 다수의 최하위 비트들을 포함한다. 비트들의 수는 오프셋이 관련된 바이트들의 그룹 내의 각 바이트에 대한 비트들의 서로 다른 엔코딩을 제공하기에 충분하다. 예를 들어, 그룹(50)은 16 바이트이다. 따라서, 그룹 내의 어드레스의 4개의 최하위 비트들은그 어드레스의 오프셋을 형성한다. 어드레스의 나머지 비트들은 주 메모리 내의 명령 바이트들의 다른 그룹들로부터 그룹(50)을 특정한다. 또한, 나머지 비트들의 다수의 최하위 비트들은 그룹(50)을 저장하는 데에 적합한 저장 위치들의 열을 선택하기 위해 명령 캐시(16)에 의해 이용되는 인덱스를 형성한다.

세트(52)는 그룹(50)을 위한 분기 선택자들의 세트의 예이다. 그룹(50) 내의 각 바이트에 대해 하나의 분기 선택자가 포함된다. 세트(52) 내의 분기 선택자들은 하기의도 13에 도시된 엔코딩을 이용한다. 본 예에서, PB0을 위한 분기 예측은 그룹(50)과 관련된 3개의 분기 예측들중 3번째의 것으로 저장된다(분기 선택자 값이 "3"으로 표시됨). 따라서, 제 1 영역(58) 내의 각 바이트에 대한 분기 선택자는 "3"으로 세트된다. 유사하게, PB1에 대한 분기 예측은 분기 예측들중 2번째의 것으로서 저장된다(분기 선택자 값이 "2"로 표시됨). 따라서, 제 2 영역(60) 내의 각 바이트에 대한 분기 선택자는 "2"로 세트된다. 마지막으로, 순차적인 분기 예측은 제 3 영역(62)에 대해 분기 선택자에 의해 분기 선택자 엔코딩 "0"으로 표시된다.

주목할 사항으로서, x86 명령 세트의 가변 바이트 길이 특성으로 인해, 분기 명령은 연속적인 명령 바이트들의 한 그룹 내에서 시작하고, 연속적인 명령 바이트들의 2번째 그룹 내에서 끝난다. 이러한 경우, 분기 명령에 대한 분기 예측은 연속적인 명령 바이트들의 2번째 그룹과 함께 저장된다. 특히, 연속적인 명령 바이트들의 2번째 그룹 내에 저장된 분기 명령의 바이트들은 페치되어 디스페치될 필요가 있다. 연속적인 명령 바이트들의 1번째 그룹에 분기 예측을 형성하게 되면, 명령 바이트들의 2번째 그룹 내에 있는 분기 명령의 바이트들의 페치되지 않게 된다.

세트(53)는 그룹(50)에 대한 복귀 선택자들의 세트의 예이다. 그룹(53) 내의 복귀 선택자들은 하기 도 13에 도시된 엔코딩들을 이용한다. 본 예에서, 복귀 선택자들은 분기 선택자가 복귀 명령을 특정하는지, 아니면 비복귀 분기 명령을 특정하는지를 나타낸다. 도시된 실시에에서, 1개의 복귀 선택자 비트는 각각의 비연속적인 분기 선택자들과 관련된다(즉, 분기 선택자 값들은 "1", "2" 및 "3"이다). 복귀 선택자 값은 복귀 선택자에 관련된 분기 선택자가 복귀 명령에 대응하는지, 아니면 비복귀 분기 명령에 대응하는 지를 특정한다. 도시된 실시예에서, 복귀 선택자 값이 "1"이면, 관련된 분기 선택자는 복귀 명령을 특정한다. 대안적으로, 복귀 선택자 값이 "0"이면, 관련된 분기 선택자는 비복귀 명령을 특정한다. 도시된 실시예에서, 세트(S3)는 제 1 분기 선택자가 복귀 명령을 특정하고, 제 2 및 3 분기 선택자들이 비복귀 분기 명령들을 특정함을 나타낸다. 도시된 실시예에서, 그룹(50)은 2개의 비복귀 분기 명령들을 포함하고, 복귀 명령들은 포함하지 않는다. 세트(50)에 복귀 명령이 포함되어 있었다면, 복귀 명령에 의해 묘사되는 영역 내의 각 바이트에 대한 분기 선택자는 "1"로 세트될

것이다. 대안적으로, 그룹(50)이 제 3 비복귀 분기 명령을 포함하는 경우, 세트(53)의 제 1 비트는 비표명(deassertion)되며(즉, "0"으로 세트되며), 제 3 비복귀 분기 명령에 의해 묘사되는 영역 내의 각 바이트에 대한 분기 선택자는 "1"로 세트된다.

복귀 선택자들은 분기 선택자들이 가능한 분기 예측들 간에 보다 효율적으로 할당될 수 있게 한다. 예를 들어, 복귀 선택자들이 없는 실시예들에서, 2비트 분기 선택자는 1개의 순차 예측, 1개의 복귀 명령 분기 예측 또는 2개의 비복귀 분기 명령 예측들을 특정할 수 있다. 본 실시예에서, 분기 선택자들은 2개 이상의 복귀 명령들 또는 3개 이상의 비복귀 분기 명령들을 갖는 캐시 라인들에 대처할 수 없다. 캐시 라인이 복귀 명령들 또는 비복귀 분기 명령들을 초과하는 경우, 캐시 라인 내의 1개 이상의 분기 명령들이 예측될 수 없다. 복귀 선택자들은 각 분기 선택자가 많은 형태의 분기 명령들 간에 할당될 수 있게 한다. 예를 들어, 복귀 선택자들은 분기 선택자가 복귀 명령 분기 예측과 관련되는지, 아니면 비복귀 명령 분기 예측과 관련되는 지를 나타낼 수 있다. 이러한 방식으로, 비순차적인 분기 예측들은 캐시 라인 내의 분기 명령들의 형태에 따라 할당될 수 있다. 예를 들어, 3개의 분기 선택자들이 복귀 명령 예측들로서 할당되거나, 3개의 분기 선택자들이 비복귀 명령 분기 예측들로서 할당될 수 있다.

삭제

다음으로, 도 4는 그룹(50), 영역들(58, 60 및 62) 및 분기 선택자들(70) 세트의 일 실시예를 도시한다. 세트(70) 내의 분기 선택자들은 그룹(50) 내에 정의된 바이트 범위들에 대응한다. 도 4에 도시된 예에서, 연속적인 16개의 바이트들의 그룹에 대해 9개의 분기 선택자들이 이용된다. 따라서, 세트(70)는 도 3에 도시된 세트(52)가 점유하는 것 보다 분기 예측 저장부 내에서 더 작은 저장 공간을 점유함으로써, 분기 예측 저장부를 보다 작게 할 수 있다. 또한, 페치 어드레스에 응답하여 분기 선택자를 선택하기 위해 보다 좁은 폭의 선택 장치가 이용될 수 있다. 선택된 분기 선택자는 보다 신속하게 제공될 수 있으며, 이에 의해 예측된 페치 어드레스들이 각 클럭 주기에 대해 제공되는 보다 높은 주파수의 실현이 가능해진다.

일반적으로, 소정의 바이트 선택자에 대해 정의된 가장 큰 바이트 범위는 (이하 보다 상세히 설명되는 복귀 명령들을 제외하고) 가장 짧은 분기 명령과 동일해 질 수 있다. 바이트 범위의 대부분은 최대 크기가 되도록 선택된다. 그러나, 특정 조건들에 대처하기 위해, 도 4에 도시된 실시예는 최대 크기 보다 작은 2개의 바이트 범위들을 이용한다. 특히, 그룹(50)의 초기 바이트는 단일 바이트를 갖는 바이트 범위를 형성한다(이에 대해서는 하기 도 6을 참조하여 상세히 설명한다). 그룹(50)이 짝수 바이트이기 때문에, 초기 바이트에 대응하는 바이트 범위는 단지초기 바이트 만을 포함하고, 가장 큰 바이트 범위는 본 예에서는 2바이트이며, 다른 바이트 범위는 단일 바이트를 갖는 것으로 정의된다. 세트(70)에 있어서, 초기 바이트에 연속하는 그룹(50) 내의 바이트는 단일 바이트 범위가 되도록 선택된다. 이러한 선택에 의해, 도 5에 도시된 바와 같이, 페치 어드레스의 오프셋을 비교적 간단하게 디코드하여 분기 선택자를 선택할 수 있다.

바이트 범위들은 가장 짧은 분기 명령보다 크지 않게 선택되기 때문에, 분기 명령은 1 바이트 범위에서 시작하고 후속 바이트 범위에서 끝날 수 있다. 그러나, 비록 분기 명령들이 연속적인 명령 바이트들의 특정 그룹 내에서 연속적이라고 할 지라도, 기껏해야 1개의 분기 명령이 특정한 바이트 범위에서 끝난다. 바이트 범위의 마지막은 아닌 특정 바이트 범위 내에서 끝나는 분기 명령의 경우, 그 바이트 범위에 대한 분기 선택자는 그 분기 명령에 후속하는 명령 바이트들에 대응하는 분기 선택자들이 되도록 선택된다. 예를 들어, 바이트 범위(72)(바이트 3 및 4를 포함하고, 여기서 초기 바이트는 바이트 0의 번호가 부여된다)에 대한 분기 선택자는 예측된 분기(PB1)에 대응하는 분기 예측을 나타낸다. 상기 규칙이 이용되는바, 그 이유는 바이트 범위 내의 페치 어드레스가 (선행하는 바이트 범위에서 시작하는) 분기 명령을 페치하지 않기 때문이다. 따라서, 정확한 분기 예측은 후속 분기에 대한 예측이다.

반면, 분기 명령이 바이트 범위 내의 마지막 바이트에서 끝난다면, 바이트 범위에 대한 분기 예측기는 분기 명령(예를 들어, 바이트 범위(74))에 대응하는 분기 선택자이다. 따라서, 페치 어드레스가 예측된 분기(PB1)을 특정한다면(즉, 오프셋이 바이트 범위(74) 내에 있으면), 페치에 이용되는 분기 예측은 분기(PB1)에 대응하는 분기 예측이다.

다음으로, 도 5는 도 4의 예에서 도시된 것과 같은 분기 선택자들에 대한 바이트 범위들의 선택에 대응하는 표(76)를 도시한다. 표(76)의 "바이트 위치"라 명명된 열은 각 바이트 범위에 대응하는 그룹(50) 내의 바이트 위치들을 열거한다(즉, 각 바이트 범위 내에 있는 각 바이트에 대한 어드레스의 오프셋 부분). "분기 선택자 위치"라 명명된 열은 바이트 범위에 대응하는 분기 선택자의 세트(70) 내의 분기 선택자 위치를 나타낸다. "읽기 어드레스들"이라 명명된 열은 (후속 클럭 주기에 대해 예측된 페치 어드레스를 형성하기 위하여) 대응하는 바이트 범위 내의 분기 선택자를 선택하기 위해 디코드된 페치 어드레스 오프셋들을 (이진수 형태로) 열거한다. 읽기 어드레스들에서 "x"는 관심없는(don't care) 위치를 나타낸다. 마지막으로, "엔코딩 어드레스들"이라 명명된 열은, 분기 명령이 종료할 수 있고, 그 바이트 범위에 대한 분기 선택자가 그 분기 명령에 대응하는 분기 예측을 여전히 나타낼 수 있는 페치 어드레스 오프셋을 (이진수 형태로) 열거한다. 예를 들어, 분기 선택자 위치 2는 바이트 위치 3 또는 4에서 끝나는 분기 명령에 대한 분기 예측을 나타낼 수 있다. 보다 구체적으로, 바이

트 위치 2에서 끝나는 분기 명령은 분기 선택자 위치 2 내의 분기 선택자에 의해 표현되지 않는다. (그 이유는, 분기 명령은 분기 선택자 위치 2와 관련된 것과 다른 바이트 범위에서 시작되며, 이에 따라 페치 어드레스 오프셋이 분기 선택자 위치 2에 관련된 바이트 범위 내에 있는 경우, 페치되지 않기 때문이다).

표(76)의 "읽기 어드레스들"의 열은, 페치 어드레스 오프셋의 비교적 간단한 디코딩을 이용하여, 그 페치 어드레스에 대한 적절한 분기 선택자를 선택할 수 있음을 나타낸다. 분기 선택자 위치들 0 및 1에 대한 디코딩은 페치 어드레스 오프셋 비트들 각각을 포함하지만, 나머지 위치들에 대한 디코딩은 최하위 비트를 배제할 수 있다(왜냐하면, 이에 대해서는 관심이 없기 때문이다). 빠른 디코드 및 분기 선택자의 선택은 도 4에 도시된 바이트 범위들의 할당을 이용하여 달성될 수 있다.

다음으로, 도 6은 도 4 및 도 5에 도시된 바이트 범위들에 대한 분기 선택자들의 제 1 예(80)를 도시한다. 예(80)는 연속적인 명령 바이트들의 그룹의 바이트 0 만을 포함하는 바이트 범위의 이용을 도시한다. 예(80)는 연속적인 명령 바이트들의 제 1 그룹의 바이트 "E"에서 끝나는 명령(IN0)과, 제 1 그룹의 바이트 "F"에서 제 1 그룹에 순차적인 연속적인 명령 바이트들의 제 2 그룹의 바이트 0으로 확장되는 예측된 분기(PB0)와, 그리고 제 2 그룹의 바이트 "1"에서 시작하는 명령(IN1)을 도시한다.

분기 명령(PB0)은 제 2 그룹의 바이트 0까지 끝나지 않기 때문에, 제 2 그룹은 명령 캐시(16)로부터 페치될 필요가 있다. 따라서, 제 1 그룹의 바이트들 "E" 및 "F"에 대응하는 분기 선택자는 순차적인 것으로 나타나고, 이에 의해 제 2 그룹이 페치된다. 순차적인 페치 어드레스는 (제 1 그룹의 마지막 바이트 바로 다음에 오는 바이트들이 페치되기 때문에) 0으로 세트된 오프셋 부분을 포함한다. 따라서, 제 2 그룹의 바이트 "0"에 대응하는 분기 선택자가 선택된다. 분기 선택자는 분기 명령(PB0)에 대응하는 분기 예측을 선택하도록 코드화된다.

제 2 그룹의 바이트들 0 및 1이 도 4 및 도 5에 도시된 것과 유사한 바이트 범위를 형성하는 경우, 분기 명령(PB0)을 예측할 수 없게 된다. 본 실시예에서, 명령 바이트들의 그룹들에 걸쳐 확장된 분기들을 예측할 수 있도록, 바이트 범위들은 바이트"0"으로 구성된 바이트 범위를 포함하는 것으로 정의된다.

다음으로, 도 7은 도 4 및 도 5에 도시한 바이트 범위에 대한 분기 명령의 제 2 예(82)를 도시한다. 예(82)는 예측된 분기 명령(PB0)에 복귀 명령(RET)이 계속되는 상황을 나타낸다. 분기 명령(PB0)은 분기 선택자 위치 3에 대응하는 바이트 범위 내에 있는 바이트 "4"에서 끝난다(즉, 바이트 "4"는 바이트 범위의 마지막 바이트가 아니다). 따라서, 분기 선택자 위치 3의 분기 선택자는, 바이트 범위 내에서 끝나는 분기 명령에 관련하여 상기 설명한 바와 같이, PB0에 대한 분기 예측을 선택하지 않는다. 복귀 명령은 바이트 범위의 마지막 바이트에서 끝난다. 따라서, 분기 선택자 위치 3에 대한 분기 선택자는 복귀 명령을 나타낸다(즉, 예(82)의 엔코딩 "1"을 나타냄). 본 예에서, 분기 예측 1에 관련된 복귀 복귀 선택자 비트가 표명(assertion)되는바, 이는 분기 예측 1이 복귀 명령과 관련됨을 나타낸다.

반면, 도 8은 복귀 명령이 예측되지 않는 예(84)를 나타낸다. 예(84)는 예측된 분기 명령(PB0), 복귀 명령(RET) 및 명령(IN1)을 포함한다. 분기 명령(PB0)은 분기 선택자 위치 2에 대응하는 바이트 범위의 마지막 바이트에서 끝나며, 이에 따라 대응하는 분기 선택자는 분기 명령(PB0)에 대응하는 분기 예측을 선택하기 위해 엔코드된다. 복귀 명령은 분기 선택자 위치 3에 대응하는 바이트 범위 내에서 끝나고, 이에 따라 대응하는 분기 선택자는 복귀 명령을 나타내지 않는다. 이러한 방식으로, 명령(IN0)이 페치 어드레스에 의해 나타나면, 분기 선택자 위치 3의 분기 선택자가 복귀 명령을 오예측하지는 않을 것이다. 그러나, 복귀 명령이 페치 어드레스에 의해 나타낸 바이트이면, 예측 결과가 잘못된다.

주목할 사항으로서, 예(84)에 도시된 코드 순서는 통상의 것과 다를 수도 있다. 왜냐하면, 복귀 명령을 실행하기 바로 전에 종종 스택으로부터 임시 변수들이 나오기 때문이다. 따라서, 테이큰인 것으로 예측된 분기는 종종 복귀 명령을 바로 앞서지 않을 것이다.

도 9는 복귀 명령이 오예측될 수 있는 예(86)를 도시한다. 예(86)에서, 복귀 명령 바로 다음에는 예측된 분기 명령(PB0)이 계속된다. 이러한 코드 순서는 그다지 빈번하지 않을 수 있는데, 그 이유는 분기 명령(PB0)을 실행하는 유일한 방법이 코드 순서의 다른 장소로부터 직접 명령을 분기하는 것이기 때문이다. 또한, 예(86)에서 복귀 명령은 직접 분기되는 경우에만 오예측된다. 명령(IN0)이 페치되어 실행되면, 복귀 명령은 정확하게 예측된다.

다음으로, 도 10은 연속적인 명령 바이트들의 그룹(50)의 예 및 대응하는 분기 선택자들(52) 및 복귀 선택자들(53)의 세트들을 도시한다. 도 10A에서, 그룹(50)은 비분기 명령들(IN0-IN5), 테이큰인 것으로 예측된 분기 명령(PB0 및 PB1) 및 복귀 명령(RET1)을 포함한다. 주목할 사항으로서, 명령들의 길이는 단지 예시의 목적으로 도시된 것이다. 명령들은 보다 많거나 적은 명령 바이트들을 점유할 수 있으며, 그룹(50)은 보다 많거나 적은 명령들을 포함할 수 있다. PB0, PB1 및 RET의 마지막 바이트들은 그룹(50)을 4개의 영역들로 분할한다.

세트(52)는 그룹(50)에 대한 분기 선택자들의 세트의 예이다. 세트(70) 내의 분기 선택자들은 그룹(50) 내에 정의된 바이트 범위들에 대응한다. 세트(70) 내의 분기 선택자들은 하기 도 13에 도시된 엔코딩을 이용한다. 따라서, 분기 선택자들은 1개의 순차 분기 선택자("0"의 분기 선택자 값으로 나타냄)와 3개의 비순차 분기 선택자들("1", "2" 또는 "3"의 분기 선택자 값으로 나타냄)를 포함한다. 상기 설명한 바와 같이, 복귀 선택자들은 상기 비순차 분기 선택자들을, 복귀 명령 분기 예측기들 또는 비복귀 명령 분기 예측기들로서 특정한다.

세트(53)는 비순차 분기 선택자에 대한 복귀 선택자들의 세트의 예이다. 도시된 실시예에서, 1개의 비트가 각각의 비순차 분기 선택자와 관련된다. 도시된 실시예에서, 가장 왼쪽에 있는 복귀 선택자 비트에는 분기 선택자들(3)이 관련되고, 중앙의 복귀 선택자 비트에는 분기 선택자(2)가 관련되며, 그리고 가장 오른쪽에 있는 선택자 비트는 분기 선택자(1)가 관련된다. 다른 실시예에서, 복귀 선택자들(53)은 단지 비순차 분기 선택자들의 일부하고만 관련될 수 있다. 복귀 선택자들(53)의 각 비트는, 복귀 선택자 비트와 관련된 분기 선택자가 복귀 명령을 특정하는지, 아니면 비복귀 분기 명령을 특정하는 지를 나타낸다. 예를 들어, 도시된 실시예에서, 분기 선택자(1)는 복귀 명령과 관련되고, 분기 선택자들(2 및 3)은 비복귀 분기 명령들과 관련된다. 따라서, 복귀 선택자(53)의 3번째 비트(또는 가장 오른쪽 비트)가 표명되며, 나머지 비트들은 비표명된다. 복귀 선택자 비트들에 의해, 비순차 분기 선택자들이 복귀 명령들 또는 비복귀 분기 명령에 대처할 수 있게 된다. 이러한 방식으로, 복귀 명령들과 비복귀 분기 명령들의 다수의 조합들이 효율적으로 예측될 수 있다.

복귀 선택자들이 없는 분기 예측 유닛의 실시예들에서, 어떠한 분기 선택자들이 복귀 명령 분기 예측들에 할당되고, 나머지 분기 선택자들은 비복귀 명령 분기 예측들에 할당된다. 예를 들어, 4개의 분기 예측들이 있는 분기 예측 유닛에서, 1개의 분기 선택자는 비순차 분기 예측일 수 있고, 1개의 분기 예측은 복귀 분기 예측일 수 있으며, 그리고 2개의 분기 선택자들은 비복귀 분기 명령 예측들이 될 수 있다. 이러한 실시예들에서, 단지 1개의 복귀 명령 만이 캐시 라인 내에서 효율적으로 예측될 수 있다. 유사하게, 단지 2개의 비복귀 분기 명령들 만이 캐시 라인 내에서 예측될 수 있다. 3개의 복귀 선택자 비트들을 부가함으로써, 3개의 비순차 분기 예측들은 복귀 명령들과 비복귀 분기 명령들의 보다 많은 조합을 예측할 수 있다. 예를 들어, 복귀 선택자들(53)은 3개의 비순차 분기 선택자들 모두가 복귀 명령 분기 예측들이 되도록 구성될 수 있다. 대안적으로, 복귀 선택자들(53)은 3개의 비순차 분기 선택자들 모두가 비복귀 분기 명령 예측들이 되도록 구성될 수 있다.

삭제

다음으로, 도 10B는 2개의 복귀 명령들 및 1개의 비복귀 분기 명령을 포함하는 그룹(50)에 대한 복귀 선택자 비트들(53)의 예를 도시한다. 도시된 실시예에서, 복귀 선택자(53)의 비트들은, 분기 선택자들 1 및 3이 복귀 명령들과 관련되고, 분기 선택자 2가 비복귀 분기 명령과 관련되도록 구성된다. 2개의 비순차 분기 예측들이 비복귀 분기 명령들에 할당되어 있는, 복귀 선택자 비트들이 없는 일 실시예에서, 2개의 복귀 명령들이 예측될 수 없다. 복귀 명령을 정확하게 예측하지 못하게 되면, 프로세서는 복귀 명령에 후속하는 명령들을 실행한 다음 버리기 때문에, 프로세서의 성능을 저하시킨다.

다음으로, 도 10C는 그룹(50)이 3개의 비복귀 분기 명령들을 포함하고 있는 실시예를 예시한다. 예시된 실시예에서, 복귀 선택자들(53)의 비트들은 비표명되어, 비순차 분기 예측들이 비복귀 분기 명령들과 관련됨을 나타낸다. 예시된 실시예에서, 분기 선택자 3은 PB0과 관련되고, 분기 선택자 2는 PB1과 관련되며, 그리고 분기 선택자 1은 PB2와 관련된다.

다음으로, 도 11은 분기 예측 유닛(14) 일 실시예의 일부를 도시한다. 분기 예측 유닛(14)의 다른 실시예들이 고려된다. 도 11에 도시된 바와 같이, 분기 예측 유닛(14)은 분기 예측 저장부(90), 방향 멀티플렉서(92), 복귀 선택자 멀티플렉서(93), 분기 선택자 멀티플렉서(94), 초기 예측 멀티플렉서(98), 최종 예측 멀티플렉서(100), NOR 게이트(95), AND 게이트(97), 갱신 논리 블럭(102) 및 디코더(104)를 포함한다. 분기 예측 저장부(90) 및 디코더(104)는 명령 캐시(16)로부터의 페치 어드레스 버스(106)에 결합된다. 명령 캐시(16)로 동시에 제공되는 페치 어드레스는 페치 어드레스 버스(106)를 통해 전달된다. 디코드 블럭(104)은 분기 선택자 멀티플렉서(94)로 선택 제어를 제공한다. 방향 멀티플렉서(92)를 위한 예측 제어들은 명령 캐시(16)로부터의 방향 선택 버스(108)를 거쳐 제공된다. 방향 선택 버스(108)는, 페치 어드레스 버스(106)를 통해 제공되는 페치 어드레스에 대응하는 캐시 라인을 저장하는 명령 캐시(16)의 방향을 제공한다. 또한, 선택 제어는 캐시 라인의 어느 부분이 페치되고 있는 지에 기초하여 디코더(104)에 의해 제공된다.

방향 멀티플렉서(92)는 페치 어드레스 버스(106) 상의 페치 어드레스에 의해 지시되는 분기 예측 저장부(90)의 열 내의 각 저장 위치의 내용들을 받도록 결합된다. 분기 선택자 멀티플렉서(94) 및 초기 예측 멀티플렉서(98)는 입력들로서 방향 멀티플렉서(92)의 출력의 일부를 수신하도록 결합된다. 복귀 선택자 멀티플렉서는 입력들로서 방향 멀티플렉서(92)의 출력의 복귀 선택자 부분을 받도록 결합된다. 분기 선택자 멀티플렉서(94)의 출력은 멀티플렉서들(93 및 98)을 위한 선택 제어를 제공한다. 초기 예측 멀티플렉서(98)는 명령 캐시(16)로부터 순차 어드레스 버스(110)를 통해 제공되는 순차 어드레스와 3개의 분기 예측 어드레스들(BP1, BP2 및 BP3) 중에서 선택한다. 멀티플렉서(98)의 출력 및 복귀 스택으로부터 복귀

어드레스 버스(112)를 통해 제공되는 복귀 어드레스는 최종 예측 멀티플렉서(100)에 제공되는바, 이 멀티플렉서(100)는 명령 캐시(16)로의 분기 예측 버스(114)를 제공한다. 멀티플렉서(94)의 출력은 OR 게이트(95)에 제공된다. 멀티플렉서(93)의 출력 및 NOR 게이트(95)의 출력은 AND 게이트(97)에 제공된다. AND 게이트의 출력은 최종 예측 멀티플렉서(100)에 대한 선택 제어를 제공한다.

명령 캐시(16)는 분기 예측 버스(114) 상에 제공되는 분기 예측을 후속 클럭 주기동안 페치 어드레스로서 이용한다. 갱신 논리 블럭(102)은 그 내에 저장된 분기 예측 정보를 갱신하는 데에 이용되는 갱신 버스(116)를 통해 분기 예측 저장부(90)에 결합된다. 갱신 논리 블럭(102)은 기능 유닛들(24) 및 디코드 유닛들(20)로부터 오예측 버스(118)를 통해 신호되는 오예측에 응답하여 갱신을 행한다. 또한, 갱신 논리 블럭(102)은 예측 버스(120) 상에 프리페치/프리디코드 유닛(12)에 의해 나타나는 새롭게 프리디코드된 명령에 응답하여 갱신을 행한다.

본 실시예에서, 분기 예측 저장부(90)는 명령 캐시(16) 내의 방향들과 같은 수의 방향을 갖도록 배열된다. 각 방향에 있어서, 캐시 라인 내에 존재하는 연속적인 명령 바이트들의 각 그룹에 대해 분기 예측 엔트리가 저장된다. 도 11의 실시예에서, 2개의 명령 바이트들의 그룹이 각 캐시 라인에 포함된다. 따라서, 분기 예측 엔트리(P_{00})는 제 1 방향에 있어서 연속적인 명령 바이트들의 제 1 그룹에 대응하는 분기 예측 엔트리이고, 분기 예측 엔트리(P_{01})는 제 1 방향에 있어서 연속적인 명령 바이트들의 제 2 그룹에 대응하는 분기 예측 엔트리이다. 유사하게, 분기 예측 엔트리(P_{10})는 제 2 방향에 있어서 연속적인 명령 바이트들 제 1 그룹에 대응하는 분기 예측 엔트리이고, 분기 예측 엔트리(P_{11})는 제 2 방향에 있어서 연속적인 명령 바이트들의 제 2 그룹에 대응하는 분기 예측 엔트리이다. 나머지도 마찬가지이다. 인덱스된 열의 각 분기 예측 엔트리들(P_{00} 내지 P_{31})은 분기 예측 저장부(90)의 출력으로서 제공되며, 이에 따라 방향 멀티플렉서(92)의 입력으로서 제공된다. 인덱스된 열은 캐시 내로의 인덱싱과 유사하다. 즉, 페치 어드레스들의 오프셋 부분의 일부가 아닌 다수의 비트들이 분기 예측 저장부(90)의 열들중 하나를 선택하는 데에 이용된다. 주목할 사항으로서, 분기 예측 저장부(90)는 명령 캐시(16)보다 적은 열들을 갖도록 구성될 수 있다. 예를 들어, 분기 예측 저장부(90)는 명령 캐시(16)의 열들의 수의 $\frac{1}{4}$을 갖는다. 이러한 경우, 명령 캐시(16)의 인덱스 비트들이지만 분기 예측 저장부(90)의 인덱스 비트들은 아닌 어드레스 비트들이, 분기 예측 정보와 함께 저장되고, 페치 어드레스의 대응하는 비트들에 대해 체크되어, 분기 예측 정보가 액세스되고 있는 명령 캐시(16)의 열과 관련됨을 확인할 수 있다.

방향 멀티플렉서(92)는, 명령 캐시(16)로부터 제공되는 방향 선택과, 페치 어드레스에 의해 참조되는 명령 바이트들의 그룹에 기초하여, 분기 예측 정보(P_{00}-P_{31}) 세트들중 하나를 선택한다. 도시된 실시예에서, 예를 들어 32 바이트 캐시 라인은 2개의 16 바이트 그룹으로 분할된다. 따라서, 페치 어드레스의 최하위 비트로부터 5번째 비트는 2개의 그룹들중 어느 것이 페치 어드레스를 포함하는 지를 선택하는 데에 이용된다. 최하위 비트로부터 5번째 비트가 0이면, 연속적인 명령 바이트들의 제 1 그룹이 선택된다. 최하위 비트중 5번째 비트가 1이면, 연속적인 명령 바이트들중 제 2 그룹이 선택된다. 주목할 사항으로서, 일 실시예에 따르면, 방향 선택 버스(108) 상에 제공되는 방향 선택은 이전 클럭 주기로부터의 분기 예측에 의해 생성된 방향 예측이 될 수 있다. 대안적으로, 방향 선택은 페치 어드레스와, 명령 캐시의 각 방향에 저장된 캐시 라인들을 특정하는 어드레스 태그들을 비교함으로 생성될 수 있다. 주목할 사항으로서, 어드레스 태그는, 캐시 라인 내의 오프셋도 아니고 명령 캐시 내의 인덱스도 아닌 어드레스의 일부이다.

방향 멀티플렉서(92)에 의해 제공되는 선택된 분기 예측 엔트리는 연속적인 명령 바이트들의 그룹에 대응하는 분기 선택자들의 세트, 복귀 선택자들의 세트 및 분기 예측들(BP1, BP2 및 BP3)을 포함한다. 분기 선택자들은 분기 선택자 멀티플렉서(94)에 제공되는바, 이는 디코더(104)에 의해 제공되는 선택 제어에 기초하여 분기 선택자들중 하나를 선택한다. 디코더(104)는 페치 어드레스의 오프셋을 연속적인 명령 바이트들의 그룹에 디코드하여, (예를 들어, 일 실시예에서는, 표(76)의 "읽기 어드레스" 열에 따라), 대응하는 분기 선택자를 선택한다. 예를 들어, 연속적인 명령 바이트들의 그룹이 16 바이트이면, 디코더(104)는 페치 어드레스의 최하위 4개의 비트들을 디코드한다. 이러한 방식으로, 분기 선택자가 선택된다.

선택된 분기 선택자는 선택 제어들을 초기 예측 멀티플렉서(98) 및 복귀 선택자 멀티플렉서(93)에 제공하는 데에 이용된다. 복귀 선택자들은 복귀 선택자 멀티플렉서(93)에 제공되며, 멀티플렉서(93)는 멀티플렉서(94)에 의해 제공되는 분기 선택자에 기초하여 복귀 선택자 비트들중 하나를 선택한다. 일 실시예에서는, 분기 선택자의 엔코딩이 멀티플렉서 선택 제어로서 직접 이용될 수 있다. 다른 실시예들에서는, 분기 선택자 멀티플렉서(94)와 멀티플렉서들(93 및 98) 사이에 논리 블럭이 삽입될 수 있다. 멀티플렉서(98)는 페치되는 연속적인 명령 바이트들의 그룹에 대응하는 분기 예측 저장부(90)에 저장된 다수의 분기 예측들, 및 페치되고 있는 연속적인 명령 바이트들의 그룹에 후속하는 연속적인 명령 바이트들의 그룹의 후속 어드레스로부터, 분기 예측을 선택한다.

멀티플렉서(100)는 멀티플렉서(98)에 의해 제공되는 분기 예측과 복귀 어드레스 스택으로부터의 복귀 어드레스 간의 선택을 한다. 멀티플렉서(100)는 AND 게이트(97)의 출력에 의해 제어된다. AND 게이트(97)는, 멀티플렉서(94)에 의해 제공되는 분기 선택자가 비순차적이고, 그리고 분기 선택자와 관련된 복귀 선택자 비트가 분기 선택자가 복귀 명령과 관련된다고 나타내는 경우, 멀티플렉서(100)가 복귀 어드레스를 선택하게 한다. 도시된 실시예에서, 순차 분기 선택자에 대한 분기 선택자 값은 "0"이며, 복귀 선택자 비트에 관련된 분기 예측이 복귀 명령에 대응하는 경우, 복귀 선택자 비트가 표명된다. 따라서, NOR 게이트(95)의 출력은 분기 선택자의 어떠한 비트가 표명되는 경우 표명되며, 그리고 AND 게이트(97)는, NOR 게이트(95)의 출력이 표명되고 멀티플렉서(93)에 의해 제공되는 복귀 선택자 비트가 표명되는 경우 표명된다. 도시된 실시예에서, 멀티플렉서(100)는 AND 게이트(97)의 출력이 표명될 때 복귀 어드레스를 선택한다.

복귀 스택 구조(도면 미도시)는 마이크로프로세서(10)에 의해 이전에 페치된 서브루틴 호출 명령에 대응하는 복귀 어드레스를 저장하는 데에 이용된다. 일 실시예에서, 분기 예측 저장부(90)에 의해 저장된 분기 예측들은 분기 예측이 서브루틴 호출 명령에 대응한다는 표시를 포함한다. 서브루틴 호출 명령들은 명령 스트림을 서브루틴 호출 명령의 목표 어드레스로 돌리는 것에 부가하여, 순차 명령(복귀 어드레스)의 어드레스를 저장하는, 분기 명령들의 서브셋이다. 예를 들어, x86 마이크로프로세서 구조에서, 서브루틴 호출 명령(CALL)은 복귀 어드레스를 ESP 레지스터에 의해 표시되는 스택에 넣는다.

서브루틴 복귀 명령은 분기 명령들의 다른 서브셋이다. 서브루틴 복귀 명령은 가장 최근에 실행된 서브루틴 호출 명령에 의해 저장된 복귀 어드레스를 목표 어드레스로서 이용한다. 따라서, 분기 예측기 자신이 서브루틴 호출 명령에 대응한다는 표시를 포함할 때, 서브루틴 호출 명령에 대한 순차 어드레스가 복귀 스택의 꼭대기에 놓여진다. 서브루틴 복귀 명령이 존재하면(특정한 분기 선택자 엔코딩에 의해 나타남), 이전에 예측으로서 이용되지 않았던 복귀 스택의 꼭대기에 가장 가까운 어드레스가 어드레스 예측으로서 이용된다. 이전에 예측으로서 이용된 적이 없었던 복귀 스택의 꼭대기에 가장 가까운 어드레스는, (방향 선택 버스(108) 상에 제공하는 것과 유사하게 복귀 스택에 제공되는, 복귀 어드레스의 예측된 방향과 함께) 복귀 스택에 의해 복귀 어드레스 버스(112) 상에 옮겨진다. 분기 예측 유닛(14)은 복귀 어드레스가 언제 예측으로서 선택되는 지를 복귀 스택에 알린다. 예시적인 복귀 스택 구조에 대한 부가적인 세부 내용들은, 동일 양수인에게 양도된 동시 계류중인 특허 출원인, 1995년 10월 30일 마하링가이아(Mahalingaiah) 등에 의해 출원되었으며 그 명칭이 "슈퍼스칼라 마이크로프로세서를 위한 추론적인 복귀 어드레스 예측 유닛(Speculative Return Address Prediction Unit for a Superscalar Microprocessor)"인 특허 출원 제08/550,296호에 개시되어 있다. 이의 전체 내용은 본원의 참조로서 인용된다.

순차 어드레스는 명령 캐시(16)에 의해 제공된다. 순차 어드레스는 페치 어드레스 버스(106) 상의 페치 어드레스에 의해 지시되는 명령 바이트들의 그룹에 대한, 주 메모리 내의 연속적인 명령 바이트들의 다음 그룹을 특정한다. 주목할 사항으로서, 일 실시예에 따르면, 순차 어드레스가 선택되면 그 순차 어드레스에 대한 방향 예측이 제공된다. 이러한 방향 예측은 페치 어드레스에 대해 선택된 방향과 동일하게선택될 수 있다. 대안적으로, 순차 어드레스에 대한 방향 예측은 분기 예측 저장부(90) 내에 저장될 수 있다.

상기 설명한 바와 같이, 갱신 논리 블럭(102)은, 프리페치/프리디코드 유닛(12)의 대응하는 연속적인 명령 바이트들의 그룹을 프리디코드하면서, 분기 오예측 검출시 또는 분기 명령 검출시 분기 예측 엔트리를 갱신하도록 구성된다. 각 분기 예측에 대응하는 분기 예측 엔트리는, 예측이 수행될 때 갱신 논리 블럭(102)에 저장될 수 있다. 분기 태그는 (분기 태그 버스(122)를 통해) 페치되고 있는 명령들과 함께 전달되며, 이에 따라 프리디코딩하는 동안 오예측이 검출되거나 분기 예측이 검출되는 경우, 대응하는 분기 예측 엔트리가 분기 태그에 의해 특정될 수 있다. 일 실시예에서, 도 12에 도시된 분기 예측 엔트리 뿐 아니라, 분기 예측 엔트리가 페치되게 하는 페치 어드레스 및 분기 예측 엔트리가 저장되는 방향이 저장된다.

분기 오예측이 검출되면, 대응하는 분기 태그가, 분기 명령을 실행하는 기능 유닛(24) 또는 디코드 유닛들(20)로부터 오예측 버스(118)에 상에 제공된다. 디코드 유닛들(20)이 분기 태그를 제공하는 경우, 오예측은 이전에 검출되지 않은 형태가 된다(예를 들어, 그룹 내에는, 대응하는 분기 예측들을 이용하여 예측될 수 있는 것 보다 많은 분기 명령들이 있다). 디코드 유닛들(20)은 무조건 분기 명령들(즉, 항상 목표 어드레스를 선택하는 분기 명령)의 오예측을 검출한다. 기능 유닛들(24)은, 이전에 검출되지 않는 조건 분기 명령로 인한, 또는 부정확한 테이큰/낫테이큰 예측으로 인한 오예측을 검출할 수 있다. 갱신 논리(102)는 상기 언급한 저장부로부터 대응하는 분기 예측 엔트리를 선택한다. 이전에 검출되지 않은 분기 명령의 경우, 분기 예측 엔트리 내의 분기 예측들중 하나는 이전에 검출되지 않은 분기 명령에 할당된다. 일 실시예에 따르면, 이전에 검출되지 않은 분기 명령에 대한 분기 예측을 저장하기 위해 분기 예측들중 하나를 선택하는 알고리즘은 다음과 같다. 분기 명령이 서브루틴 복귀 명령이면, 그 명령에 대한 분기 선택자는 복귀 스택을 나타내는 값이 되도록 선택된다. 그렇지 않으면, 현재 낫테이큰인 것으로 예측된 분기 예측이 선택된다. 각 분기 예측이 현재 테이큰인 것으로 예측되는 경우, 분기 예측은 임의로 선택된다.

새롭게 검출된 분기 명령에 대한 분기 선택자는 선택된 분기 예측을 나타내도록 세트된다. 또한, 새롭게 검출된 분기 명령 이전의 제 1 분기 명령과 새롭게 검출된 분기 명령 간의 바이트 범위에 대응하는 분기 선택자들은 새로운 예측에 대응하는 분기 선택자로 세트된다. 도 15는 분기 선택자들을 갱신하는 한 방법을 나타낸다. 예측을 낫테이큰인 것으로 예측되게 하는 오예측된 테이큰 예측에 대해, 오예측된 예측에 대응하는 분기 선택자들은 오예측된 분기 명령에 후속하는 바이트에 대응하는 분기 선택자로 세트된다. 이러한 방식으로, 명령들이 이후 클럭 주기에서 다시 페치되는 경우, 후속 분기 명령에 대한 예측이 이용될 것이다.

프리페치/프리디코드 유닛(12)이 연속적인 명령 바이트들의 그룹을 프리디코딩하면서 분기 명령을 검출할 때, 캐시 라인에 대해 명령 캐시에 저장된 무효한 프리디코드 정보 때문에 프리디코드가 수행되는 경우, 프리페치/프리디코드 유닛(12)은 연속적인 명령 바이트들의 그룹에 분기 태그를 제공한다(경우(i)). 대안적으로, 주 메모리 서브시스템으로부터 페치되고 있는 캐시 라인 상에서 프리디코딩이 수행되고 있는 경우, 프리페치/프리디코드 유닛(12)은 프리디코드되고 있는 연속적인 명령 바이트들의 그룹의 어드레스, 그 그룹 내의 분기 명령의 끝 바이트의 오프셋, 및 그룹을 저장하기 위해 선택되는 명령 캐시의 방향을 제공한다(경우(ⅱ)). 경우(ⅰ)에서, 갱신은 상기 분기 오예측의 경우와 유사하게 수행된다. 경우(ⅱ)에서, 명령들의 그룹에 대해 분기 예측 저장부(90)에 저장된 유효 분기 예측 엔트리는 아직 없다. 이러한 경우, 갱신 논리 블럭(102)은 검출된 분기에 대하여 선택된 분기 선택자에 대해, 검출된 분기 이전의 분기 선택자들을 초기화한다. 또한, 검출된 분기에 후속하는 분기 선택자들은 순차적인 값으로 초기화된다. 대안적으로, 명령 캐시(16)의 대응하는 캐시 라인이 할당될 때, 각 분기 선택자들은 순차적으로 초기화된 다음, 상기 경우(i)와 유사한 방법으로 프리디코드되는 동안 분기 명령들의 검출을 통해 갱신될 수 있다.

갱신의 발생시, 갱신 논리 블럭(102)은 페치 어드레스 인덱스 및 대응하는 방향과 함께, 갱신된 분기 예측 엔트리를 버스(90) 상으로 전달하며, 이들은 분기 예측 저장부(90)에 저장된다. 주목할 사항으로서, 분기 예측 저장부(90)를 단일 포트의 저장부로서 유지하기 위해, 분기 예측 저장부(90)는 분기 홀딩 레지스터를 이용할 수 있다. 갱신된 예측 정보는 분기 홀딩 레지스터에 저장되며, 페치 어드레스 버스(106)의 아이들(idle) 주기시 분기 예측 저장부로 갱신된다. 예시적인 캐시 홀딩 레지스터 구조는, 동일 양수인에게 양도된 동시 계류중인 특허 출원인, 1995년 6월 7일 트란(Tran) 등에 의해 출원되었으며 그 명칭이 "어레이를 위한 지연된 갱신 레지스터(Delayed Update Register for an Array)"인 특허 출원 제08/481,914호에 개시되어 있으며, 이의 전체 내용은 본원의 참조로서 인용된다.

주목할 사항으로서, 분기 명령이 정확하게 예측되면, 대응하는 분기 예측 또한 갱신된다. 분기 명령 이전 실행들을 나타내는 카운터(분기 명령의 테이큰/낫테이큰 예측을 형성하는 데에 이용됨)는, 예를 들어 증분 또는 감분될 필요가 있다. 이러한 갱신들은 대응하는 분기 예측의 폐기시 수행된다. 폐기는 재배열 버퍼(32)로부터의 폐기 태그 버스(124) 상의 분기 태그에 의해 표시된다.

주목할 사항으로서, 도 11의 구조는 예측된 분기 선택자를 이용하여 한층 더 개선될 수 있다. 예측된 분기 선택자는 각 분기 예측 엔트리와 함께 저장되며, 대응하는 연속적인 명령 바이트들의 그룹의 이전의 페치시 선택된 분기 선택자로 세트된다. 예측된 분기 선택자는 분기 예측을 선택하는 데에 이용되어, 분기 예측 생성 경로로부터 분기 선택자 멀티플렉서(94)를 제거한다. 하지만, 분기 선택자 멀티플렉서(94)는, 선택된 분기 선택자가 예측된 분기 선택자와 같은 지를 검증하는 데에 여전히 이용될 수 있다. 선택된 분기 선택자와 예측된 분기 선택자가 동일하지 않으면, 선택된 분기 선택자는 계속되는 클럭 주기 동안 정확한 분기 예측을 제공하는 데에 이용되며, 부정확한 분기 예측의 페치는 무효화된다.

도 12는 도 10에 도시된 분기 예측 유닛(14)의 일 실시예에 의해 이용되는 예시적인 분기 예측 엔트리(130)를 도시한다. 분기 예측 엔트리(130)는 분기 선택자들의 세트(136), 복귀 선택자들의 세트(137), 제 1 분기 예측(BP1)(131), 제 2 분기 예측(BP2)(132) 및 제 3 분기 예측(BP3)(134)을 포함한다. 분기 선택자들의 세트(136)는 분기 예측 엔트리(130)에 대응하는 연속적인 명령 바이트들의 그룹의 각 바이트 범위에 대한 분기 선택자를 포함한다. 일 실시예에서, 복귀 선택자들(136)의 세트는 각각의 비순차 분기 선택자에 대한 한 비트를 포함한다.

제 2 분기 예측(132)은 도 12의 분해도에 도시된다. 다른 분기 예측들은 또한 이와 유사하게 구성될 수 있다. 제 2 분기 예측(132)은 목표 어드레스에 대응하는 명령 바이트들을 포함하는 캐시 라인에 대한 인덱스(140) 및 상기 캐시 라인에 대한 방향 선택(144)을 포함한다. 일 실시예에 따르면, 인덱스(140)는 목표 어드레스의 오프셋 부분 및 인덱스를 포함한다. 인덱스(140)는 방향 선택(144)에 의해 지시되는 방향의 태그에 결합되어, 분기 목표 어드레스를 형성한다. 대안적으로, 전체 분기 목표 어드레스는 인덱스 필드(140)에 저장될 수 있다. 방향 예측은 전체 분기 목표 어드레스에 부가하여 제공되거나, 명령 캐시(16)의 인덱스된 열의 태그들에 대한 태그 비교를 이용하여 수행될 수 있다.

또한, 예측기(146)는 각 분기 예측에 대해 저장된다. 예측기(146)는, 대응하는 분기 명령이 실행되고 테이큰일 때 마다 증분되며, 대응하는 분기 명령이 실행되고 낫테이큰일 때 마다 감분된다. 예측기(146)의 최상위 비트는 테이큰/낫테이큰 예측으로서 이용된다. 최상위 비트가 세트되면, 분기 명령은 테이큰인 것으로 예측된다. 반대로, 최상위 비트가 클리어되면, 분기 명령은 낫테이큰인 것으로 예측된다. 일 실시예에서, 예측 카운터는 2비트 포화 카운터이다. 이 카운터는, 이진수 "11"에서 증분될 때 포화되며, 이진수 "01"에서 감분될 때 포화된다. 다른 실시예에서, 예측기는 스트롱(strong)(이진수 1) 또는 위크(weak)(이진수 0) 테이큰 예측을 나타내는 단일 비트이다. 스트롱 테이큰 예측이 오예측되는 경우, 이는 위크 테이큰 예측이 된다. 위크 테이큰 예측이 오예측되는 경우, 분기는 낫테이큰인 것으로 예측되며, 분기 선택자는 갱신된다(즉, 낫테이큰인 것으로 오예측된 분기의 경우). 마지막으로, 호출 비트(148)가 제 1 분기 예측(132)에 포함된다. 호출 비트(148)는 세트될 때, 대응하는 분기 명령이 서브루틴 호출 명령임을 나타낸다. 호출 비트(148)가 세트되면, 현재의 페치 어드레스 및 방향은 상기 설명한 바와 같이 복귀 스택 구조 내에 저장된다.

다음으로, 도 13은 예시적인 분기 및 복귀 선택자 엔코딩을 나타낸 표(138)를 도시한다. 표(138)는 복귀 선택자 엔코딩들 및 분기 엔코딩들에 기초한 분기 예측의 타입을 나타낸다. 분기 선택자 엔코딩들은 최상위 비트를 첫 번째로 갖는 이진 엔코딩들이다. 복귀 선택자 엔코딩들은 최상위 비트를 첫 번째로 갖는 이진 엔코딩들이다. "X"는 상관없는 비트를 나타낸다. 예를 들어, 분기 선택 엔코딩이 "00"이면, 분기 예측은 복귀 선택자 비트들의 상태에 상관없이 순차적이다. 분기 선택 엔코딩이 "01"이면, 최하위 복귀 선택자 비트가 표명되는 경우 분기 예측은 복귀이다. 대안적으로, 상기 복귀 선택자의 최하위 비트가 비표명되는 경우, 분기 예측은 비복귀 분기 목표이다. 유사한 방식으로, 분기 선택자의 값이 "01"이면, 제 2 최상위 복귀 선택 비트는 그 분기 예측이 복귀 또는 비복귀 분기 목표인지를 나타낸다. 마찬가지로, 분기 선택자의 값이 "11"이면, 최상위 복귀 선택자 비트는 분기 예측이 복귀 또는 비복귀 분기 목표인지를 나타낸다.

다음으로, 도 14는 도 4 및 도 5에서 정의된 바이트 범위들에 따라 명령 캐시(16)로부터 페치되는 연속적인 명령 바이트들의 그룹에 대한 유효 마스크의 생성을 나타낸 흐름도이다. 유효 마스크는 그룹 내의 각 명령 바이트에 대한 비트를 포함하는바, 이는 그 명령 바이트가 예측된 명령 스트림의 일부인지 아닌지를 나타낸다. 마스크 내의 유효 비트들은 페치 어드레스의 오프셋 부분에 의해 선택되는 바이트와 선택된 분기 예측에 의해 예측되는 분기 명령 중에서 세트된다. 이러한 바이트들은 명령 캐시(16)로부터 읽혀지는 연속적인 명령 바이트들의 그룹으로부터 페치되는 명령들을 포함한다. 페치 어드레스의 오프셋 이전의 바이트들은 페치되지 않고, 테이큰인 것으로 예측된 분기 명령에 후속하는 바이트들도 페치되지 않는다. 분기 명령의 끝은 거의 대응하는 분기 선택자의 마지막 경우(instance)에 의해 위치된다. 그러나, 분기 명령이 바이트 범위 내에서 끝나는 경우들에서는, 분기 명령의 마지막 바이트를 커버하는 조정이 이루어질 수 있다.

현재 분기 선택자(즉, 페치 어드레스를 디코딩함으로써 분기 예측 유닛(14)에 의해 선택된 분기 선택자)는 연속적인 명령 바이트들의 그룹에 대응하는 분기 선택자들과 XNOR 조합되어(단계 150), 바이트 범위 마다 한 비트를 포함하는 예비 마스크를 생성한다. 현재 분기 선택자와 동일한 분기 선택자에 대응하는 각 바이트 범위는 예비 마스크 내의 세트 비트에 의해 표현되고, 현재의 분기 선택자와 다른 분기 선택자들을 포함하는 바이트 범위들은 예비 마스크 내의 클리어 비트에 의해 표현된다. 제 1 마스크는 1개 이상의 바이트(본 실시예에서는 2 바이트)을 갖는 바이트 범위에 대응하는 예비 마스크의 각 비트를 복사함으로써 예비 마스크로부터 생성된다(단계 152). 일반적으로, 비트는, 비트 범위를 나타내는 바이트들의 총 수가 그 바이트 범위 내의 바이트들의 수와 같아질 때 까지, 카피된다. 또한, 제 2 마스크는 제 1 마스크를 오른쪽으로 시프팅함으로써 생성된다(단계 154).

테이큰인 것으로 예측된 분기 명령이 바이트 범위 내에서(바이트 범위의 끝은 아님) 끝나는 경우, 유효 마스크는 제 1 마스크와 제 2 마스크를 논리합함으로써 생성된다(결정 블럭 156 및 단계 158). 이러한 방식으로, 분기 명령이 끝나는 바이트 범위 내에 있어서 분기 명령의 일부인 바이트들은 유효한 것으로 확인된다. 반면, 분기 명령이 바이트 범위의 끝에서 끝나면, 그 바이트 범위에 대한 분기 선택자는 분기 명령을 나타낸다. 이러한 경우, 제 1 마스크는 제 2 마스크를 이용하여 수정하지 않아도 정확하다. 따라서, 제 1 마스크가 유효 마스크로서 선택된다(단계 160).

다음으로, 도 15는 오예측된 분기 명령에 응답하여 연속적인 명령 바이트들의 그룹의 분기 선택자를 갱신하는 데에 이용되는 단계들을 도시한 흐름도이다. 프리디코딩 동안 발견되는 분기 명령로 인한 갱신이 또한 유사하게 수행될 수 있다. 오예측은 예측 정보가 예측 저장부(90)에 저장되지 않는 분기 명령의 검출 결과이거나, 또는 대응하는 예측기가 낫테이큰을 나타내게 하는 부정확한 테이큰/낫테이큰 예측의 결과일 수 있다.

오예측 검출시, 분기 예측 유닛(14)은 "끝 지시기(end pointer)"를 사용한다. 이는 대응하는 연속적인 명령 바이트들의 그룹 내의 오예측된 분기 명령의 끝 바이트의 오프셋이다. 또한, 분기 예측 엔트리는 오예측에 응답하여 수신된 분기 태그를 이용하여 갱신을 하기 위해 선택된다. 분기 예측 유닛(14)은 끝 지시기를 갱신 마스크에 디코드한다(단계 170). 갱신 마스크는 연속적인 명령 바이트들의 그룹 내의 각 바이트 범위에 대해 1 비트를 포함한다. 분기 명령의 끝 바이트를 포함하는

바이트 범위 이전의 바이트 범위들에 대응하는 비트들이 세트되고 (그리고 끝 바이트가 바이트 범위 내의 마지막 바이트인 경우, 분기 명령의 끝 바이트를 포함하는 바이트 범위에 대응하는 비트가 세트되며) 나머지 비트들은 클리어된다. 따라서, 갱신 마스크는, 분기 명령에 선행하여 분기 명령을 포함하는 각 바이트 범위를 특정한다.

분기 예측 유닛(14)은 현재의 분기 선택자를 특정한다. 오예측된 테이큰/낫테이큰 예측들에 대해, 현재의 분기 선택자는 오예측된 분기 명령에 대응하는 분기 선택자이다. 검출되지 않은 분기로 인한 오예측에 대해, 현재의 분기 선택자는 검출되지 않은 분기 명령의 끝 바이트를 포함하는 바이트 범위에 대응하는 분기 선택자이다. 현재의 분기 선택자는 각 분기 선택자들과 XNOR되어 분기 마스크를 생성한다(단계 172). 분기 마스크는 현재의 분기 선택자에 적합한 분기 선택자를 갖는 각 바이트 범위에 대해 세트되는 비트들과, 그리고 현재의 분기 선택자에 적합하지 않은 분기 선택자를 갖는 각 바이트 범위에 대해 클리어되는 비트들을 포함한다.

단계(170)에서 생성되는 갱신 마스크와 단계(172)에서 생성되는 분기 마스크는 이후 AND 조합되어, 최종 갱신 마스크를 생성한다(단계 174). 최종 갱신 마스크는, 새로운 분기 선택자로 갱신될 연속적인 명령 바이트들의 그룹의 각각의 바이트 범위에 대해 세트되는 비트들을 포함한다. 테이큰인 것으로 오예측된 분기에 대해, 새로운 분기 선택자는 이 테이큰인 것으로 오예측된 분기 명령에 후속하는 바이트 범위의 분기 선택자이다. 검출되지 않은 분기에 대해, 새로운 분기 선택자는 갱신 논리 블럭(102)에 의해 이전에 검출되지 않은 분기에 할당된 분기 예측 저장부를 나타내는 분기 선택자이다.

또한, 확장된 마스크가 생성된다(단계 176 및 단계 178). 확장된 마스크는, 분기 선택자에 대응하는 분기 예측이 새롭게 발견된 분기 명령에 재할당되기 때문에, 또는 분기 예측이 이제 낫테이큰임을 나타내기 때문에, 어떤 분기 선택자들이 소거될 것인 지를 나타낸다. 확장된 마스크는, 먼저, 현재의 분기 선택자 대신 새로운 분기 선택자를 이용하는 것을 제외하고, 분기 마스크와 유사한 제 2 마스크를 생성함으로써 발생된다(즉, 이 마스크는 연속적인 명령 바이트들의 그룹에 대응하는 분기 선택자들과 새로운 분기 선택자를 XNOR 조합함으로써 생성된다(단계 176)). 결과적으로 얻을 수 있는 마스크는 이후, 최종 갱신 마스크의 반전에 대해 AND 조합되어, 확장된 마스크를 생성한다(단계 178). 세트된 확장 마스크의 비트들에 대응하는 분기 선택자들은, 확장 마스크의 비트가 세트되는 마지막 바이트 범위 바로 다음에 오는 바이트 범위의 분기 선택자를 나타내도록 갱신된다. 이러한 방식으로, 분기 선택자가 이전에 나타낸 분기 예측은 소거되고, 다음의 분기 선택자로 대체된다. 단계(180) 동안, 분기 선택자들은 최종 갱신 마스크 및 확장 마스크에 응답하여 갱신된다.

다음으로, 도 16은 마이크로프로세서(10)를 포함하는 컴퓨터 시스템(200)을 도시한다. 컴퓨터 시스템(200)은 또한 버스 브리지(220), 주 메모리(204) 및 다수의 입력/출력(I/O) 디바이스들(206A-206N)을 포함한다. 다수의 I/O 디바이스들(206A-206N)은 집합적으로 I/O 디바이스(206)라 불린다. 마이크로프로세서(10), 버스 브리지(202) 및 주 메모리(204)는 시스템 버스(208)에 결합된다. I/O 디바이스들(206)은 버스 브리지(202)와 통신하기 위해 I/O 버스(210)에 결합된다.

버스 브리지(202)는 I/O 디바이스들(206)과 시스템 버스(208)에 결합된 장치들 간의 통신을 보조하기 위해 제공된다. 전형적으로, I/O 디바이스들(206)은 마이크로프로세서(10) 및 시스템 버스(208)에 결합된 다른 장치들 보다 더 긴 버스 클럭 주기들을 필요로 한다. 따라서, 버스 브리지(202)는 시스템 버스(208)와 입력/출력 버스(210) 사이에 버퍼를 제공한다. 또한, 버스 브리지(202)는 한 버스 프로토콜로부터 다른 버스 프로토콜로 트랜잭션들을 변환한다. 일 실시예에서, 입력/출력 버스(210)는 확장된 산업 표준 구조(Enhanced Industry Standard Architecture(EISA)) 버스이고, 버스 브리지(202)는 시스템 버스 프로토콜을 EISA 버스 프로토콜로 변환한다. 다른 실시예에서, 입력/출력 버스(210)는 주변요소 상호 접속(Peripheral Component Interconnect(PCI)) 버스이고, 버스 브리지(202)는 시스템 버스 프로토콜을 PCI 버스 프로토콜로 변환한다. 주목할 사항으로서, 많은 시스템 버스 프로토콜들이 존재한다. 마이크로프로세서(10)는 어떠한 적절한 시스템 버스 프로토콜이라도 이용할 수 있다.

I/O 디바이스들(206)은 컴퓨터 시스템(200)과 이 컴퓨터 시스템 외부의 다른 장치들 간의 인터페이스를 제공한다. 예시적인 I/O 디바이스들로는 모뎀, 직렬 또는 병렬 포트, 사운드 카드 등이 있다. I/O 디바이스들(206)을 주변 장치라 부를 수도 있다. 주 메모리(204)는 마이크로프로세서(10)가 이용하기 위한 데이터 및 명령들을 저장한다. 일 실시예에서, 주 메모리(204)는 적어도 하나의 동적 랜덤 액세스 메모리(DRAM) 및 DRAM 메모리 제어기를 포함한다.

주목할 사항으로서, 비록 도 17에 도시된 컴퓨터 시스템(200)은 1개의 버스 브리지(202)를 포함하고 있지만, 컴퓨터 시스템(200)의 다른 실시예들은 다수의 다른 또는 같은 I/O 버스 프로토콜들로 변환하기 위한 다수의 버스 브리지들(202)을 포함할 수 있다. 또한, 마이크로프로세서(10)에 의해 참조되는 명령들 및 데이터를 고속의 메모리 저장부에 저장함으로써, 컴퓨터 시스템(200)의 성능을 향상시키는 캐시 메모리가 포함될 수 있다. 캐시 메모리는 마이크로프로세서(10)와 시스템 버스(208) 사이에 삽입되거나, "룩어사이드(lookaside)" 구성으로 시스템 버스(208) 상에 존재할 수 있다. 또한, 주목할 사항으로서, 버스 브리지(202), 주 메모리(204) 및 캐시 메모리의 기능들은, 마이크로프로세서(10)와 인터페이스하는 칩셋(chipset)에 집적될 수 있다.

또한, 주목할 사항으로서, 상기 설명은 다양한 신호들의 표명에 대해 설명한다. 본원에서, 신호는, 특정 조건을 나타내는 값을 전달할 때 "표명된다"라고 말한다. 반대로, 신호는, 특정 조건이 없음을 나타내는 값을 전달할 때 "비표명된다"라고 말한다. 신호는, 논리 0값을 전달할 때, 또는 역으로 논리 1값을 전달할 때, 표명된 것으로 정의될 수 있다. 또한, 상기 설명에서는 많은 값들이 버려지는 것으로 설명되었다. 이러한 값은 많은 방식으로 버려질 수 있지만, 일반적으로 값을 변형함으로써, 이를 수신하는 논리 회로가 그 값을 무시하도록 하는 것을 포함한다. 예를 들어, 값이 한 비트로 구성된다면, 그 값의 논리 상태를 반전시켜 그 값을 버리도록 할 수 있다. 값이 n비트 값이면, n비트 엔코딩들중 하나가 그 값이 무효임을 나타낼 수 있다. 값을 무효 엔코딩으로 세트하게 되면, 그 값이 버려지게 된다. 또한, n비트 값은, 세트될 때 n비트 값이 유효함을 나타내는 유효 비트를 포함할 수 있다. 유효 비트의 리셋은 그 값을 버리는 것을 포함할 수 있다. 값을 버리는 다른 방법들이 또한 이용될 수 있다.

하기의 표 1은 x86 명령 세트를 이용하는 마이크로프로세서(10) 일 실시예에 대한 빠른 경로, 이중 디스페치 및 MROM 명령들을 나타낸다.

표 1

x86 빠른 경로, 이중 디스패치 및 MROM 명령들

X86 명령	명령 분류
AAA	MROM
AAD	MROM
AAM	MROM
AAS	MROM
ADC	빠른 경로
ADD	빠른 경로
AND	빠른 경로
ARPL	MROM
BOUND	MROM
BSF	빠른 경로
BSR	빠른 경로
BSWAP	MROM
BT	빠른 경로
BTC	빠른 경로
BTR	빠른 경로
BTS	빠른 경로
CALL	빠른 경로/이중 디스패치
CBW	빠른 경로
CWDE	빠른 경로
CLC	빠른 경로
CLD	빠른 경로
CLI	MROM
CLTS	MROM
CMC	빠른 경로
CMP	빠른 경로
CMPS	MROM
CMPSB	MROM
CMPSW	MROM
CMPSD	MROM

x86 빠른 경로, 이중 디스패치 및 MROM 명령들

X86 명령	명령 분류
CMPXCHG	MROM
CMPXCHG8B	MROM
CPUID	MROM
CWD	MROM
CWQ	MROM
DDA	MROM
DAS	MROM
DEC	빠른 경로
DIV	MROM
ENTER	MROM
HLT	MROM
IDIV	MROM
IMUL	이중 디스패치
IN	MROM
INC	빠른 경로
INS	MROM
INSB	MROM
INSW	MROM
INSD	MROM
INT	MROM
INTO	MROM
INVD	MROM
INVLPG	MROM
IRET	MROM
IRETD	MROM
Jcc	빠른 경로
JCXZ	이중 디스패치
JECXZ	이중 디스패치
JMP	빠른 경로
LAHF	빠른 경로
LAR	MROM
LDS	MROM
LES	MROM
LFS	MROM
LGS	MROM
LSS	MROM
LEA	fast path
LEAVE	이중 디스패치
LGDT	MROM
LIDT	MROM
LLDT	MROM
LMSW	MROM
LODS	MROM
LODSB	MROM
LODSW	MROM
LODSD	MROM
LOOP	이중 디스패치
LOOPcond	MROM
LSL	MROM
LTR	MROM
MOV	빠른 경로
MOVCC	빠른 경로
MOV.CR	MROM
MOV.DR	MROM
MOVS	MROM
MOVSB	MROM
MOVSW	MROM
MOVSD	MROM
MOVSX	빠른 경로
MOVZX	빠른 경로
MUL	이중 디스패치
NEG	이중 디스패치
NOP	빠른 경로
NOT	빠른 경로
OR	빠른 경로
OUT	MROM
OUTS	MROM
OUTSB	MROM
OUTSW	MROM
OUTSD	MROM
POP	이중 디스패치
POPA	MROM
POPAD	MROM
POPF	MROM
POPFD	MROM

x86 빠른 경로, 이중 디스패치 및 MROM 명령들

X86 명령	명령 분류
PUSH	빠른 경로/이중 디스패치
PUSHA	MROM
PUSHAD	MROM
PUSHF	빠른 경로
PUSHFD	빠른 경로
RCL	MROM
RCR	MROM
ROL	빠른 경로
ROR	빠른 경로
RDMSR	MROM
REP	MROM
REPE	MROM
REPZ	MROM
REPNE	MROM
REPNZ	MROM
RET	이중 디스패치
RSM	MROM
SAHF	빠른 경로
SAL	빠른 경로
SAR	빠른 경로
SHL	빠른 경로
SHR	빠른 경로
SBB	빠른 경로
SCAS	이중 디스패치
SCASB	MROM
SCASW	MROM
SCASD	MROM
SETcc	빠른 경로
SGDT	MROM
SIDT	MROM
SHLD	MROM
SHRD	MROM
SLDT	MROM
SMSW	MROM
STC	빠른 경로
STD	빠른 경로
STI	MROM
STOS	MROM
STOSB	MROM
STOSW	MROM
STOSD	MROM
STR	MROM
SUB	빠른 경로
TEST	빠른 경로
VERR	MROM
VERW	MROM
WBINVD	MROM
WRMSR	MROM
XADD	MROM
XCHG	MROM
XLAT	빠른 경로
XLATB	빠른 경로
XOR	빠른 경로

주 :
SIB 바이트를 포함하는 명령들 또한 이중 디스패치 명령들로 간주된다.

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

삭제

주목할 사항으로서, 분기 선택자들을 이용하는 분기 예측 메커니즘들의 다른 양상들은, 동일 양수인에게 양도된 동시 계류중인 하기의 특허 출원들, 즉 1996년 11월 19일 트란(Tran)에 의해 출원되었으며 그 명칭이 "분기 선택자들을 이용하여 분기 예측을 선택하는 분기 예측 메커니즘(Branch Prediction Mechanism Employing Branch Selectors to Select a Branch Prediction)"인 특허 출원 제 08/752,691호와, 그리고 1997년 8월 24일 트란에 의해 출원되었으며 그 명칭이 "분기 예측들을 신속하게 특정하기 위한 명령 캐시 내의 바이트 범위들에 관련된 분기 선택자들(Branch Selectors Associated with Byte Ranges within an Instruction Cache for Rapidly Identifying Branch Predictions)"인 제08/957,596호에 개시되어 있다. 이들의 내용은 본원의 참조로서 인용된다.

상기 설명을 완전히 이해하게 되면, 당업자들에게 많은 변형들 및 수정들이 명백해질 것이다. 하기의 청구항들은 이러한 모든 변형들 및 수정들을 포함한다.

(57) 청구의 범위

청구항 1.

명령 캐시(16)로부터 페치되는 다수의 명령들을 포함하는 연속적인 명령 바이트들의 그룹에 대응하는 페치 어드레스를 수신하도록 결합된 분기 예측 저장부(90)와, 여기에서 상기 분기 예측 저장부(90)는 상기 연속적인 명령 바이트들의 그룹에 관해 다수의 분기 예측들, 다수의 분기 선택자들 및 복귀 선택자를 포함하는 엔트리를 저장하도록 구성되고, 상기 다수의 분기 선택자들 중 적어도 하나는 상기 연속적인 명령 바이트들의 그룹 내의 제 1 명령에 대응하고, 상기 복귀 선택자는 상기 분기 선택자들 중 적어도 하나에 대응하고 상기 적어도 하나의 분기 선택자가 비-복귀 분기 명령 또는 복귀 명령과 연관되어있는 지를 식별하고, 상기 다수의 분기 선택자들 중 하나와 상기 복귀 선택자는 상기 제 1 명령이 페치되는 경우 선택될 특정한 분기 예측을 식별하며; 그리고,

상기 다수의 분기 선택자들과 상기 복귀 선택자를 수신하도록 상기 분기 예측 저장부(90)에 결합되는 선택 메커니즘부(94, 104, 98, 100)를 포함하여 구성되며, 여기에서 상기 선택 메커니즘부는 상기 연속적인 명령 바이트들의 그룹을 페치하도록 사용된 페치 어드레스의 다수의 최하위 비트들에 응답하여 상기 다수의 분기 선택자들 중 특정한 하나의 분기 선택자를 선택하고 상기 특정 분기 선택자와 상기 복귀 선택자에 응답하여 상기 특정한 분기 예측을 선택하며, 상기 특정한 분기 예측은 상기 연속적인 명령 바이트들에 관해 상기 분기 예측 저장부(90)에 저장된 상기 다수의 분기 예측들 중 하나를 포함하는 것을 특징으로 하는 분기 예측 장치(14).

청구항 2.

제 1 항에 있어서,

상기 연속적인 명령 바이트들의 그룹은 상기 연속적인 명령 바이트들의 그룹을 저장하는 명령 캐시(16)의 캐시 라인의 일부를 포함하는 것을 특징으로 하는 분기 예측 장치.

청구항 3.

제 2 항에 있어서,

상기 캐시 라인의 일부는 상기 캐시 라인의 절반을 포함하는 것을 특징으로 하는 분기 예측 장치.

청구항 4.

제 1 항, 제 2 항 또는 제 3 항에 있어서,

상기 다수의 분기 선택자들 중 적어도 하나는 상기 특정한 분기 예측으로서 순차적인 어드레스를 식별하고, 여기에서 상기 순차적인 어드레스는 메인 메모리 내의 상기 연속적인 명령 바이트들의 그룹의 바로 다음의 제 2 연속적인 명령 바이트들의 그룹에 대응하는 것을 특징으로 하는 분기 예측 장치.

청구항 5.

제 1 항, 제 2 항 또는 제 3 항에 있어서,

상기 다수의 분기 선택자들 중 적어도 하나는 상기 특정한 분기 예측으로서 비-순차적인 어드레스를 식별하는 것을 특징으로 하는 분기 예측 장치.

청구항 6.

제 5 항에 있어서,

상기 다수의 분기 선택자들 중 적어도 하나는 상기 복귀 선택자의 상태에 의존하여 상기 특정한 분기 예측으로서 복귀 스택 어드레스 또는 비-복귀 분기 명령의 목표 어드레스를 식별하는 것을 특징으로 하는 분기 예측 장치.

청구항 7.

제 6 항에 있어서,

비-순차적인 어드레스를 식별하는 상기 다수의 분기 선택자들 중 상기 적어도 하나는 상기 복귀 선택자가 주장된(asserted) 경우 상기 복귀 스택 어드레스를 식별하고, 상기 복귀 선택자가 주장해제된(deasserted) 경우 상기 비-복귀 분기 명령의 목표 어드레스를 식별하는 것을 특징으로 하는 분기 예측 장치.

청구항 8.

제 7 항에 있어서,

상기 분기 예측 저장부(90)는 다수의 복귀 선택자들을 저장하도록 구성되고, 각각의 복귀 선택자는 비-순차적인 어드레스를 식별하는 분기 선택자에 대응되고, 상기 복귀 선택자는 상기 비-순차적인 어드레스들을 식별하는 분기 선택자들이 복귀 스택 어드레스를 식별하는지 또는 비-복귀 분기 명령의 목표 어드레스를 식별하는 지를 식별하는 것을 특징으로 하는 분기 예측 장치.

청구항 9.

다수의 명령들을 포함하는 연속적인 명령 바이트들의 그룹에 대응하는 분기 예측을 선택하는 방법으로서,

다수의 명령들을 포함하는 연속적인 명령 바이트들의 그룹에 관해 분기 예측 저장부(90)에 다수의 분기 예측들을 포함하는 엔트리를 저장하는 단계와;

상기 분기 예측 저장부(90)에 다수의 분기 선택자들을 저장하는 단계와, 여기에서 상기 다수의 분기 선택자들 중 적어도 하나는 상기 다수의 명령들 중 제 1 명령에 대응하고;

상기 분기 예측 저장부(90)에 복귀 선택자를 저장하는 단계와, 여기에서 상기 복귀 선택자는 상기 다수의 분기 선택자들 중 적어도 하나에 대응하고 상기 적어도 하나의 분기 선택자가 복귀 명령 또는 비-복귀 분기 명령과 연관되는지를 식별하며, 여기에서 상기 다수의 분기 선택자들 중 적어도 하나와 상기 복귀 선택자는 상기 다수의 명령들 중 상기 제 1 명령이 페치되는 경우 특정한 분기 예측이 선택되는 지를 식별하고;

상기 연속적인 명령 바이트들의 그룹을 식별하는 페치 어드레스를 사용하여, 상기 연속적인 명령 바이트들의 그룹을 페치하고 상기 다수의 분기 선택자들과 상기 복귀 선택자를 동시에 페치하는 단계와;

상기 페치 어드레스에 응답하여 상기 다수의 분기 선택자들 중 하나를 선택하는 단계와;

상기 다수의 분기 선택자들 중 하나와 상기 복귀 선택자에 응답하여 상기 다수의 분기 예측들 중 하나를 선택하는 단계를 포함하고, 여기에서 상기 특정한 분기 예측은 상기 다수의 명령들 중 미리 결정된 명령이 페치되는 경우 선택되는 것을 특징으로 하는 분기 예측 선택 방법.

청구항 10.

제 9 항에 있어서,

상기 분기 예측은 상기 연속적인 명령 바이트들의 그룹에 대한 상기 분기 예측 저장부에 저장된 다수의 분기 예측들 중 하나를 포함하는 것을 특징으로 하는 분기 예측 선택 방법.

청구항 11.

제 10 항에 있어서,

분기 오예측에 응답하여 상기 다수의 분기 예측들을 갱신하는 단계를 더 포함하는 것을 특징으로 하는 분기 예측 선택 방법.

청구항 12.

제 11 항에 있어서,

상기 갱신 단계는, 상기 분기 오예측이 이전에 검출되지 않은 분기 명령의 결과인 경우, 상기 분기 예측 저장부에 저장된 상기 다수의 분기 예측들 중 하나를 상기 연속적인 명령 바이트들의 그룹 내의 이전에 검출되지 않은 분기 명령에 할당하는 단계를 포함하는 것을 특징으로 하는 분기 예측 선택 방법.

청구항 13.

제 12 항에 있어서,

상기 이전에 검출되지 않은 분기 명령을 반영하기 위해, 그리고 상기 다수의 분기 예측들 중 하나에 의해 미리 예측된 분기 명령에 대응하는 분기 선택자들을 삭제하기 위해, 상기 다수의 분기 선택자들을 갱신하는 단계를 더 포함하는 것을 특징으로 하는 분기 예측 선택 방법.

청구항 14.

제 9 항에 있어서,

상기 특정한 분기 예측은 메인 메모리 내에서 상기 연속적인 명령 바이트들의 그룹 바로 다음인 제 2 연속적인 명령 바이트들의 그룹에 대응하는 순차적인 어드레스를 포함하는 것을 특징으로 하는 분기 예측 선택 방법.

청구항 15.

제 9 항에 있어서,

상기 특정한 분기 예측은 미리 페치된 서브루틴 호출 명령에 대응하는 복귀 스택 어드레스 또는 상기 복귀 선택자에 의존하는 비-복귀 분기 명령의 목표 어드레스를 식별하는 것을 특징으로 하는 분기 예측 선택 방법.

청구항 16.

제 15 항에 있어서,

상기 특정한 분기 예측은 상기 복귀 선택자가 주장된 경우 상기 복귀 스택 어드레스를 식별하고 상기 복귀 선택자가 주장 해제된 경우 상기 목표 어드레스를 식별하는 것을 특징으로 하는 분기 예측 선택 방법.

청구항 17.

제 9 항 내지 제 16 항 중 어느 하나의 항에 있어서,

상기 다수의 분기 선택자들 중 하나를 선택하는 단계는 상기 페치 어드레스의 다수의 최하위 비트들을 디코딩하는 단계를 포함하는 것을 특징으로 하는 분기 예측 선택 방법.

청구항 18.

제 17 항에 있어서,

상기 페치 어드레스의 다수의 최하위 비트들은 상기 연속적인 명령 바이트들의 그룹으로의 오프셋을 포함하고, 이에 의해 상기 연속적인 명령 바이트들의 그룹 중 어느 하나는 상기 오프셋의 상이한 값을 통해 식별가능한 것을 특징으로 하는 분기 예측 선택 방법.

청구항 19.

마이크로프로세서로서,

다수의 명령 바이트들의 캐쉬 라인들을 저장하도록, 그리고 상기 마이크로프로세서의 명령 처리 파이프라인에 대한 페치 어드레스 수신시 명령 바이트들 그룹을 제공하도록 구성된 명령 캐쉬(16)와;

상기 명령 캐쉬(16)에 결합된 제 1 항, 제 2 항 또는 제 3 항에 따른 분기 예측 장치(14)를 포함하는 마이크로프로세서.

청구항 20.

제 19항에 있어서,

상기 다수의 분기 선택자들 중 하나는 상기 페치 어드레스의 다수의 최하위 비트들을 디코딩함으로써 선택되는 것을 특징으로 하는 마이크로프로세서.

청구항 21.

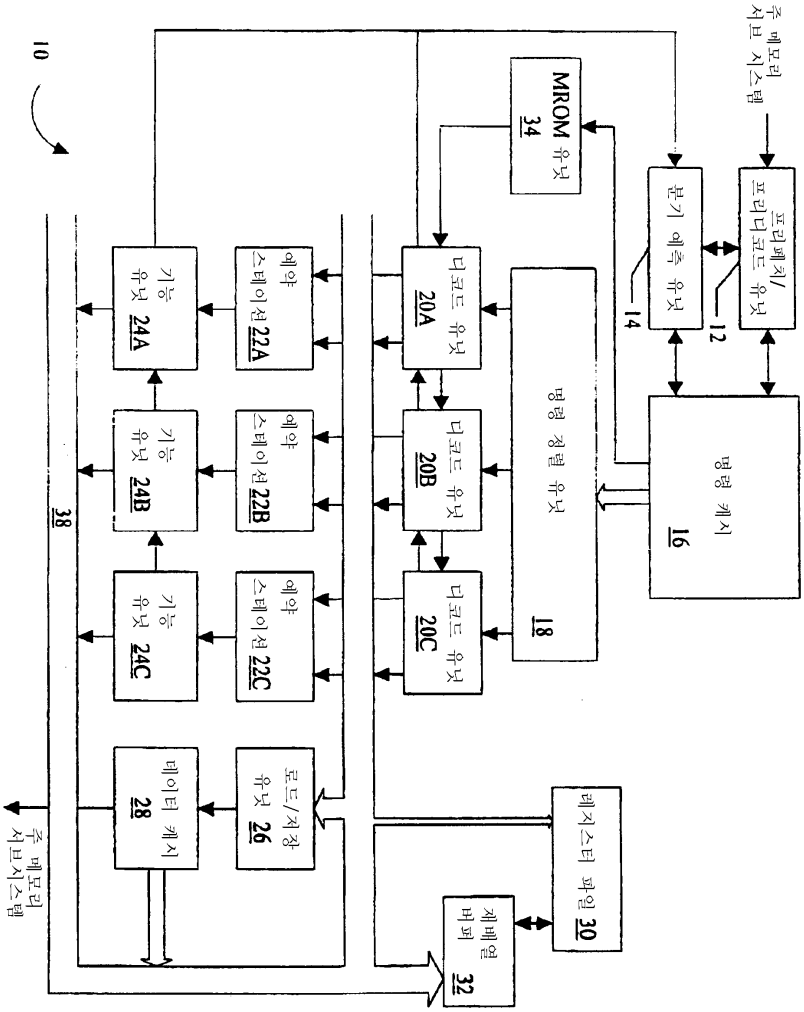
삭제

청구항 22.

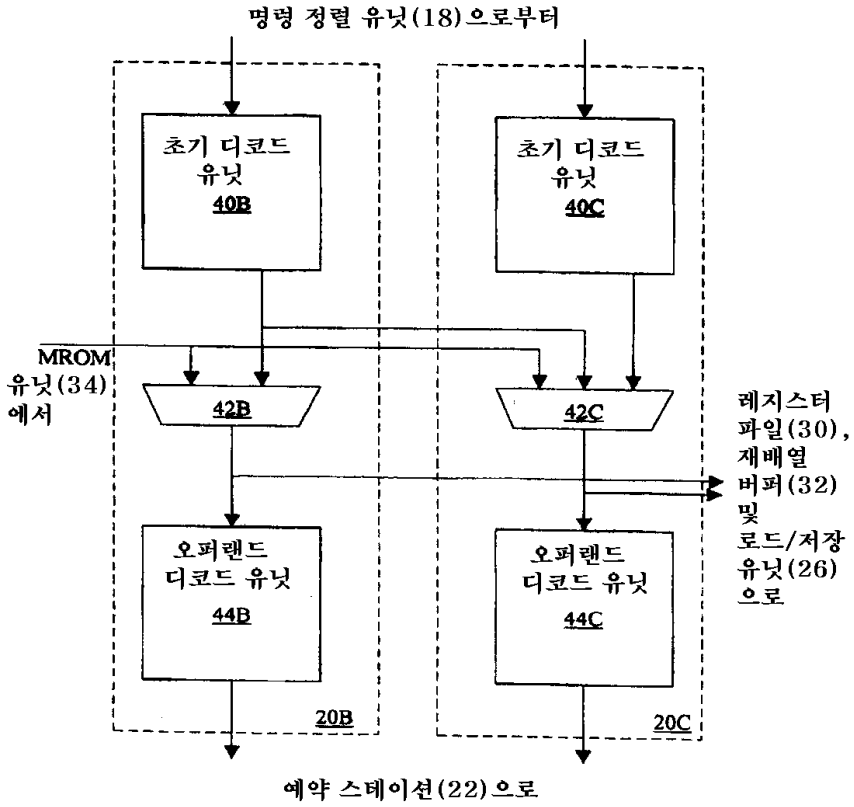
삭제

도면

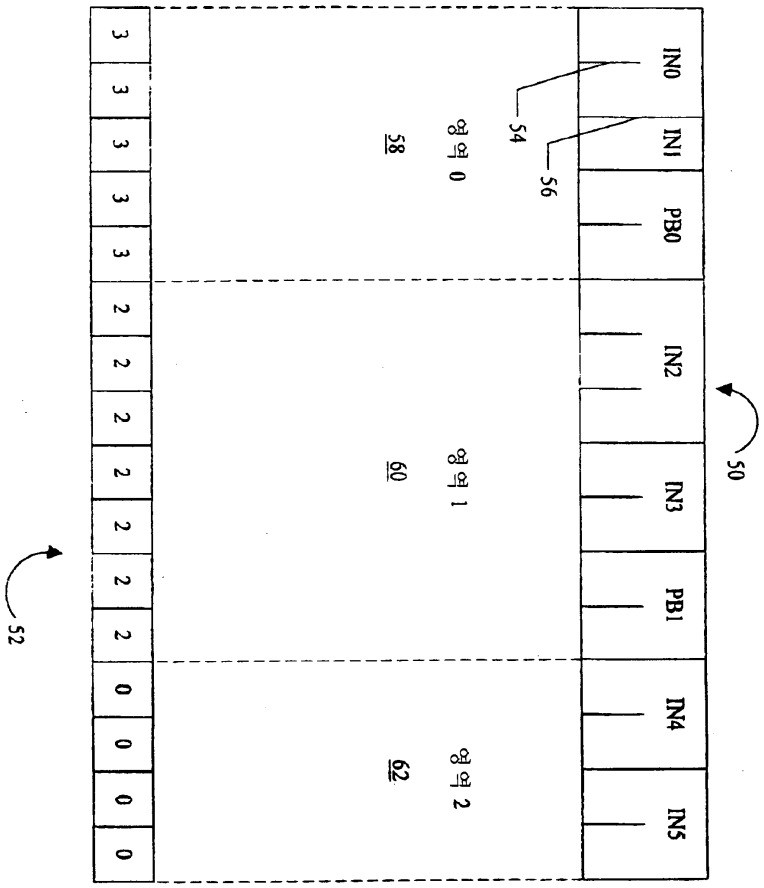
도면1


10
주 메모리 서브 시스템
프리페치/ 프리디코드 유닛
분기 예측 유닛
12
14
MROM 유닛 34
명령 캐시 16
명령 정렬 유닛 18
디코드 유닛 20A
디코드 유닛 20B
디코드 유닛 20C
예약 스테이션 22A
예약 스테이션 22B
예약 스테이션 22C
기능 유닛 24A
기능 유닛 24B
기능 유닛 24C
로드/저장 유닛 26
데이터 캐시 28
38
레지스터 파일 30
재배열 버퍼 32
주 메모리 서브시스템

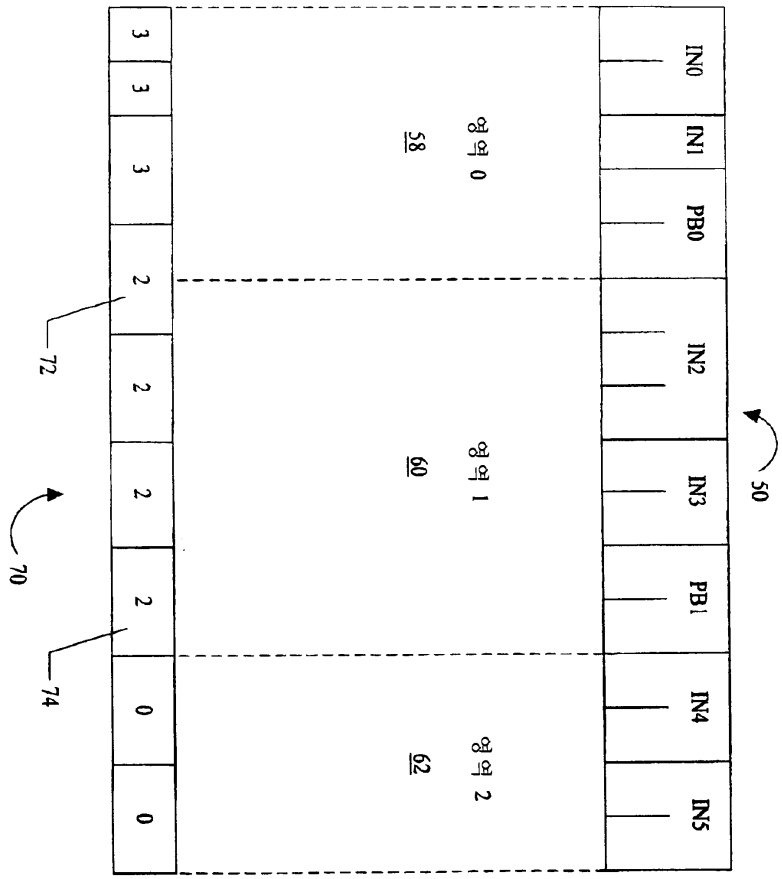
도면2


명령 정렬 유닛(18)으로부터
초기 디코드 유닛 40B
초기 디코드 유닛 40C
MROM 유닛(34)에서
42B
42C
레지스터 파일(30), 재배열 버퍼(32) 및 로드/저장 유닛(26)으로
오퍼랜드 디코드 유닛 44B
오퍼랜드 디코드 유닛 44C
20B
20C
예약 스테이션(22)으로

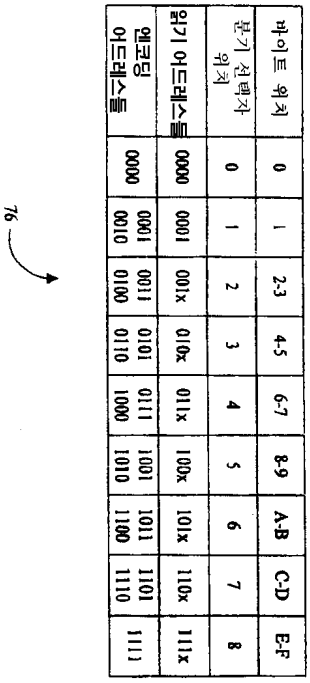
도면3


IN0
IN1
PB0
IN2
IN3
PB1
IN4
IN5
54
56
50
52
영역 0
58
영역 1
60
영역 2
62
3
3
3
3
3
2
2
2
2
2
2
2
0
0
0
0

도면4



도면5



바이트 위치		0	1	2-3	4-5	6-7	8-9	A-B	C-D	E-F
분기 선택자 위치		0	1	2	3	4	5	6	7	8
읽기 어드레스들	0000	0001	001x	010x	011x	100x	101x	110x	111x	
엘코딩 어드레스들	0000	0001	0010	0011	0100	0101	0110	0111	1001	1010	1011	1100	1101	1110	1111

도면6

바이트 위치	E	F	0	1
명령	IN0	PB0		IN1
분기 선택자	0		3	2

80

도면7

바이트 위치	2	3	4	5
명령	IN0	PB0		RET
분기 선택자	3		1	

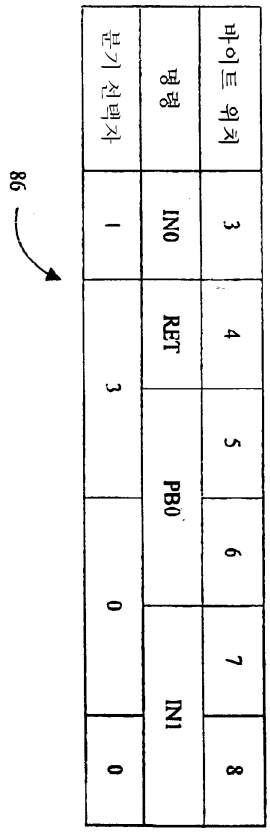
82

도면8

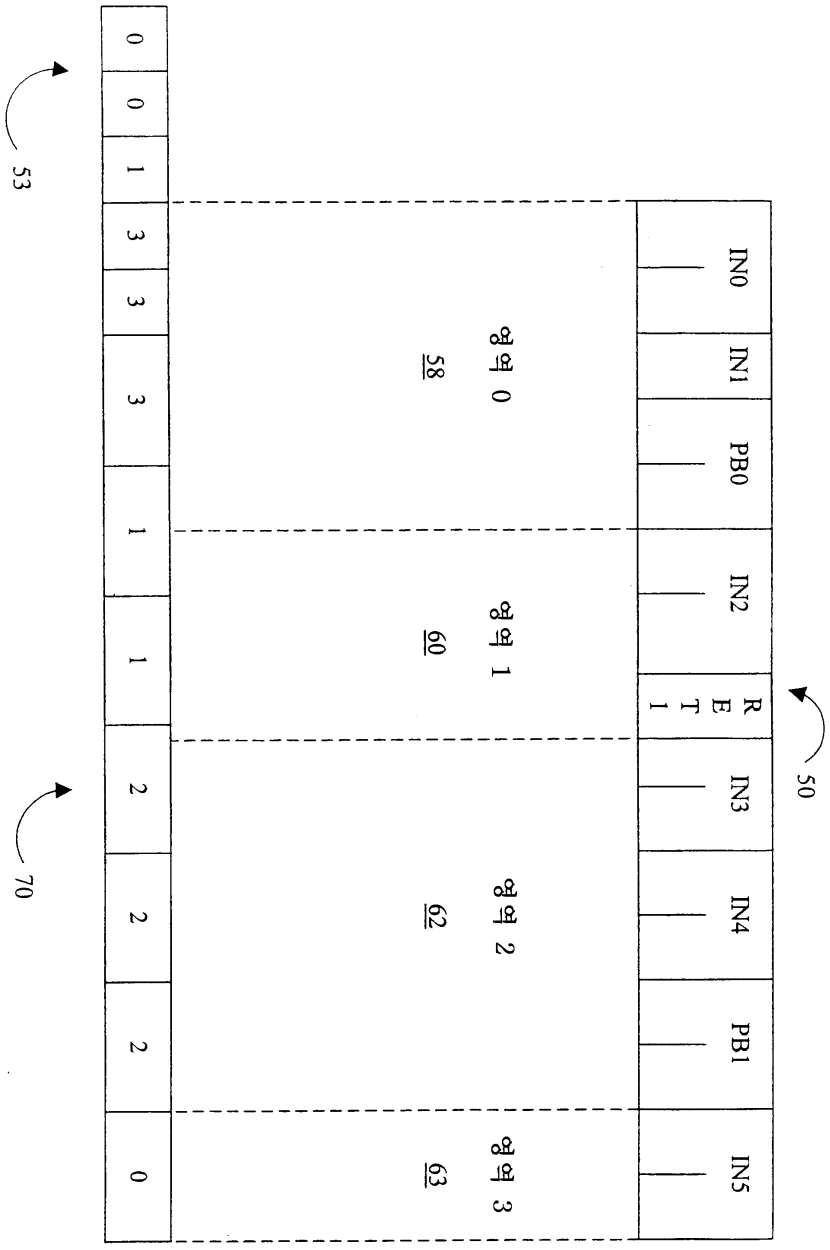
84

분기 선택자	명령	바이트 위치
3	PB0	2
		3
0	RET	4
	IN1	5

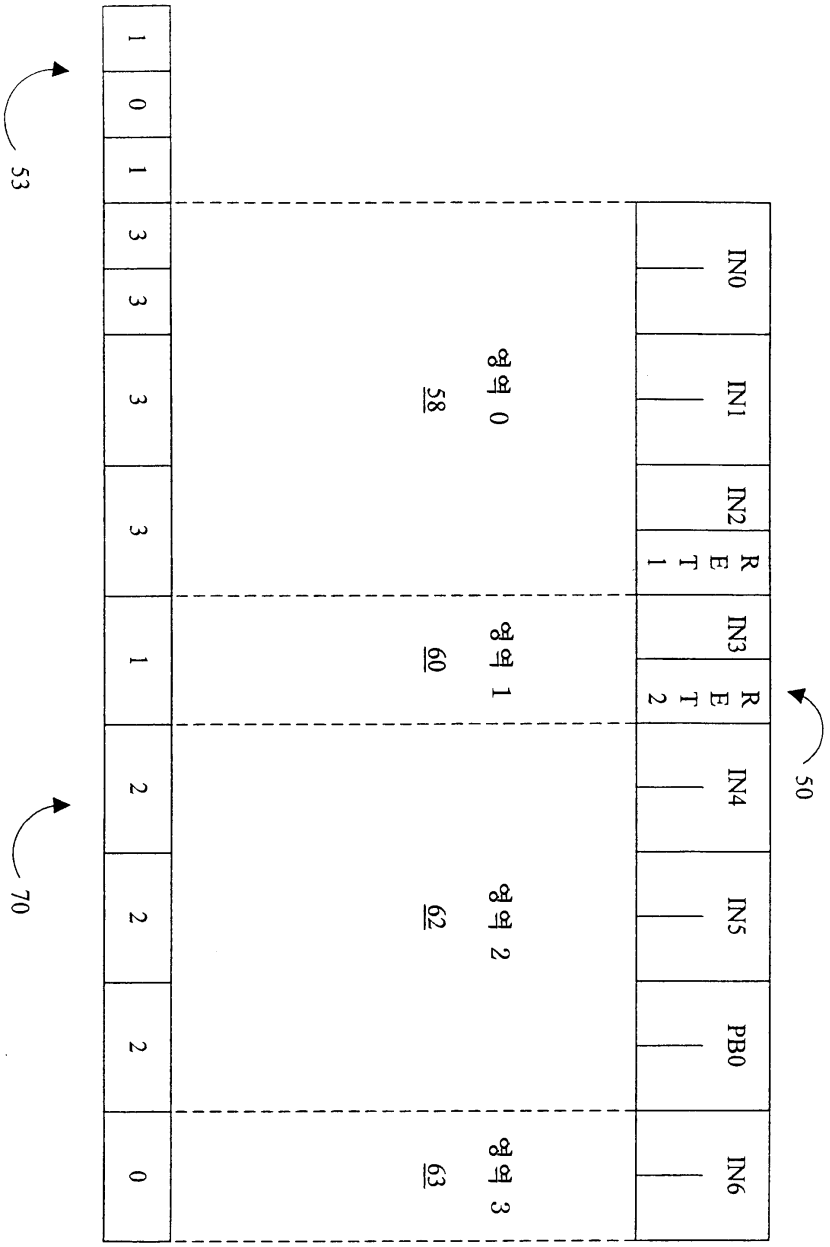
도면9



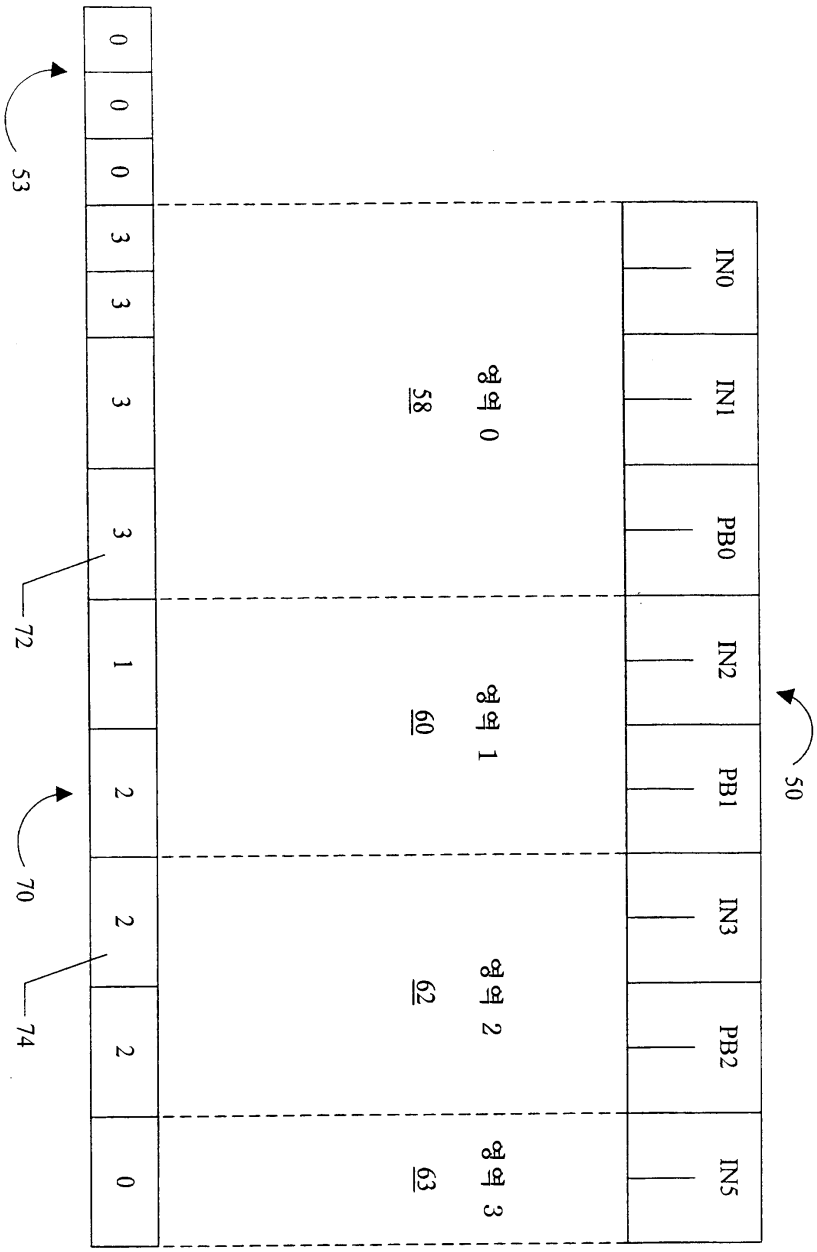
도면10a


0
0
1
3
3
3
1
1
2
2
2
0
53
70
IN0
IN1
PB0
IN2
RET1
IN3
IN4
PB1
IN5
50
영역 0
58
영역 1
60
영역 2
62
영역 3
63

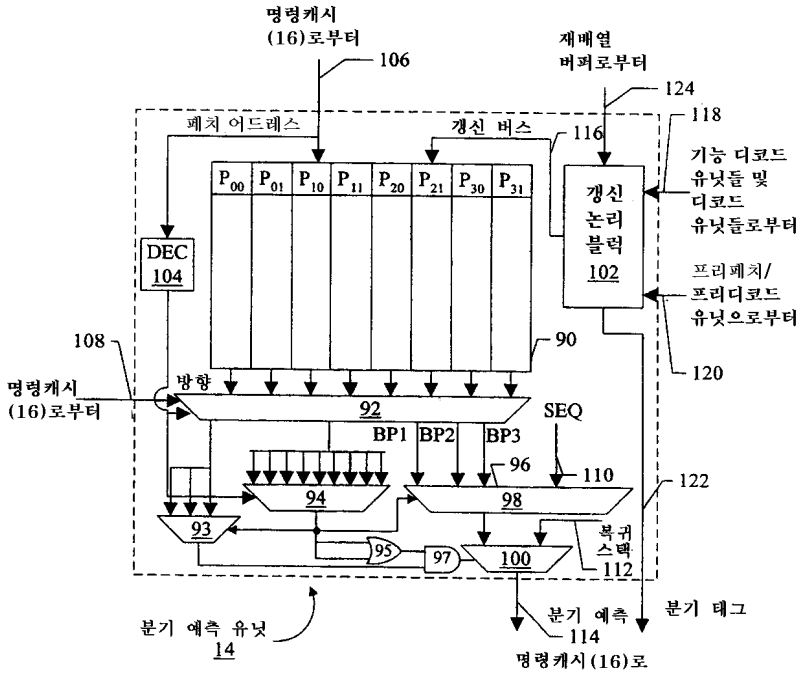
도면10b


IN0
IN1
IN2
RET1
IN3
RET2
IN4
IN5
PB0
IN6
영역 0
58
영역 1
60
영역 2
62
영역 3
63
50
53
70
1
0
1
3
3
3
3
1
2
2
2
0

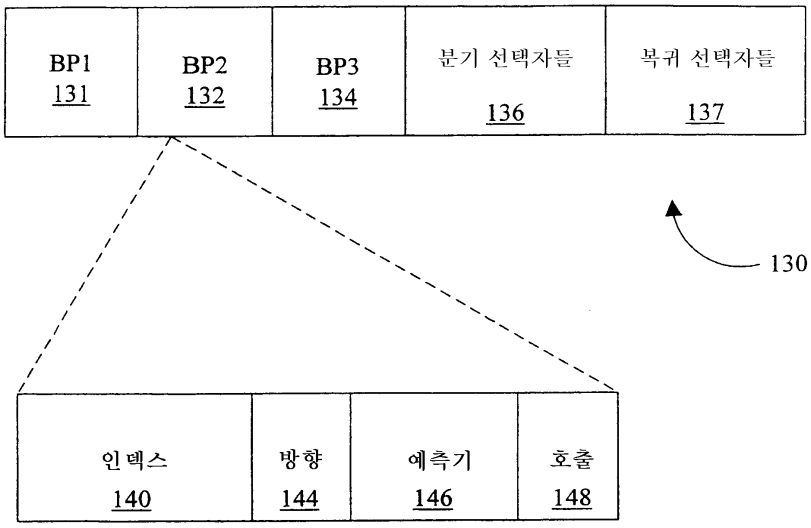
도면10c


0
0
0
3
3
3
3
3
1
2
2
2
0
53
72
70
74
50
IN0
IN1
PB0
IN2
PB1
IN3
PB2
IN5
영역 0
58
영역 1
60
영역 2
62
영역 3
63

도면11


명령캐시
(16)로부터
106
재배열
버퍼로부터
124
118
페치 어드레스
갱신 버스
116
기능 디코드
유닛들 및
디코드
유닛들로부터
갱신
논리
블럭
102
DEC
104
프리페치/
프리디코드
유닛으로부터
120
108
90
명령캐시
(16)로부터
방향
92
SEQ
BP1
BP2
BP3
96
110
122
94
98
93
95
97
100
복귀
스택
112
분기 예측 유닛
14
분기 예측
114
분기 태그
명령캐시(16)로

도면12


BP1
131
BP2
132
BP3
134
분기 선택자들
136
복귀 선택자들
137
130
인덱스
140
방향
144
예측기
146
호출
148

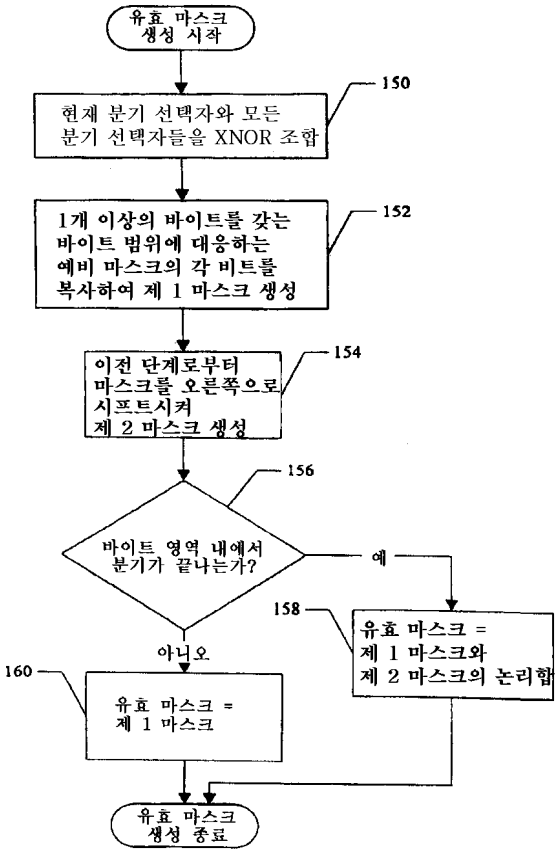
도면13

분기 및 복귀 선택기 엔코딩

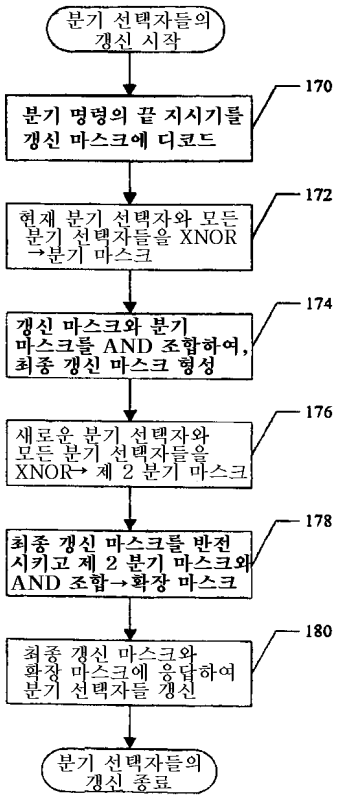
분기 선택자	복귀 선택자	분기 예측
00	XXX	순차적
01	XX1	복귀 1
01	XX0	분기예측 1
10	X1X	복귀 2
10	X0X	분기예측 2
11	1XX	복귀 3
11	0XX	분기예측 3

138

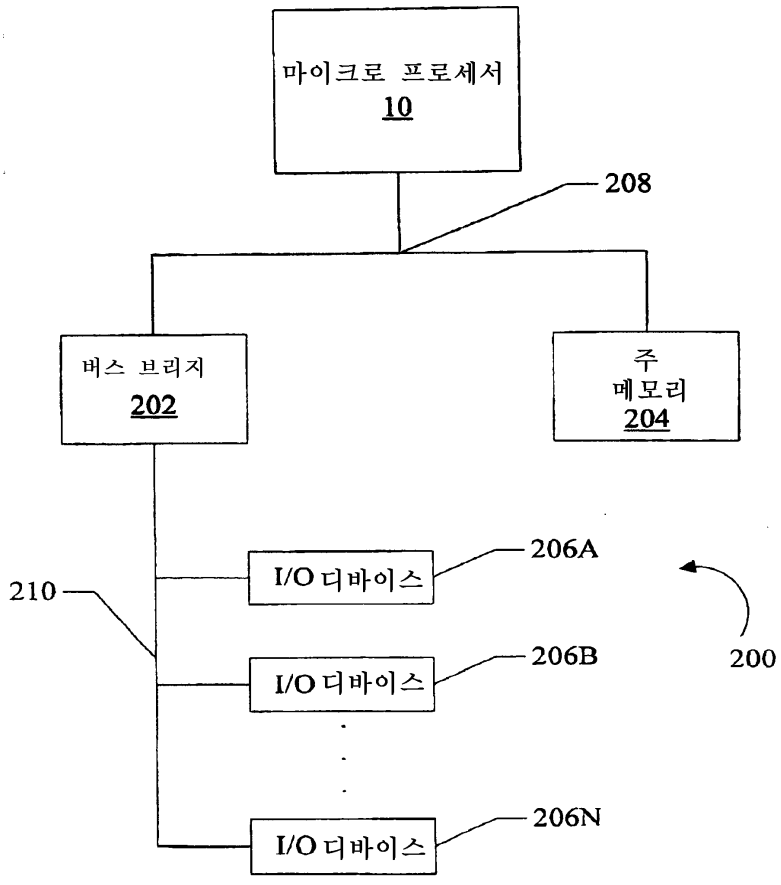
도면14



도면15


분기 선택자들의 갱신 시작
분기 명령의 끝 지시기를 갱신 마스크에 디코드
170
현재 분기 선택자와 모든 분기 선택자들을 XNOR →분기 마스크
172
갱신 마스크와 분기 마스크를 AND 조합하여, 최종 갱신 마스크 형성
174
새로운 분기 선택자와 모든 분기 선택자들을 XNOR→ 제 2 분기 마스크
176
최종 갱신 마스크를 반전시키고 제 2 분기 마스크와 AND 조합→확장 마스크
178
최종 갱신 마스크와 확장 마스크에 응답하여 분기 선택자들 갱신
180
분기 선택자들의 갱신 종료

도면16


마이크로 프로세서
10
208
버스 브리지
202
주
메모리
204
I/O 디바이스
206A
210
I/O 디바이스
206B
200
I/O 디바이스
206N