(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2009년02월18일
(11) 등록번호 10-0884351
(24) 등록일자 2009년02월11일

(51) Int. Cl.
*G06F 12/08* (2006.01)
(21) 출원번호 10-2003-7016529
(22) 출원일자 2003년12월17일
심사청구일자 2007년03월29일
번역문제출일자 2003년12월17일
(65) 공개번호 10-2004-0041550
(43) 공개일자 2004년05월17일
(86) 국제출원번호 PCT/US2002/012768
국제출원일자 2002년04월02일
(87) 국제공개번호 WO 2003/003218
국제공개일자 2003년01월09일
(30) 우선권주장
09/892,328 2001년06월26일 미국(US)
(56) 선행기술조사문헌
US6092182 A
전체 청구항 수 : 총 23 항

(73) 특허권자
**어드밴스드 마이크로 디바이시즈, 인코포레이티드**
미국 캘리포니아 94088-3453 서니베일 원 에이엠디 플레이스 메일 스톱68
(72) 발명자
**주라스키제럴드디.주니어**
미국텍사스78726오스틴미드파케코우브8009
(74) 대리인
**박장원**

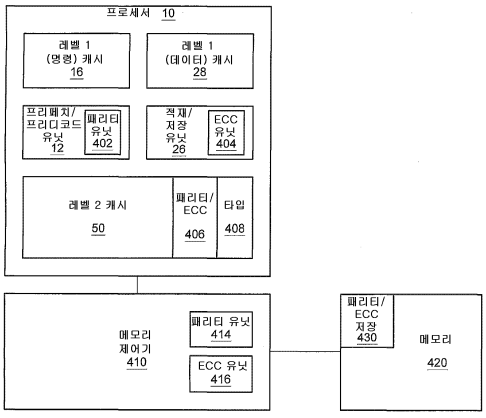
심사관 : 안철용

# (54) 타입 비트들을 이용한, 레벨 2 캐시에서의 ECC 및프리디코드 비트들의 저장 추적

## (57) 요 약

희생 명령 바이트들 및 데이터 바이트들을 저장하도록 구성된 마이크로프로세서(10)가 개시된다. 일 실시예에서, 마이크로프로세서는 프리디코드 유닛(12), 명령 캐시(16), 데이터 캐시(28) 및 레벨 2 캐시(50)를 포함한다. 프리디코드 유닛은 명령 바이트들을 수신하여 대응하는 프리디코드 정보를 발생시키는바, 이 프리디코드 정보는 명령 바이트들과 함께 명령 캐시에 저장된다. 데이터 캐시는 데이터 바이트들을 수신하여 저장한다. 레벨 2 캐시는 패리티 정보 및 프리디코드 정보와 함께 명령 캐시로부터의 희생 명령 바이트들을 수신하여 저장하고, 에러 정정 코드 비트들과 함께 데이터 캐시로부터의 희생 데이터 바이트들을 수신하여 저장하도록 구성된다. 표시자 비트들은 캐시 라인에 저장되어, 데이터 타입이 그 내에 저장되어 있는 지를 나타낸다.

*대 표 도* - 도9



## *특허청구의 범위*

**청구항 1**

명령 바이트들을 수신하고 대응하는 프리디코드 정보를 발생시키도록 구성된 프리디코드 유닛과;

상기 프리디코드 유닛에 결합되어, 상기 명령 바이트들 및 상기 명령 바이트들에 대응하는 프리디코드 정보를 저장하도록 구성된 명령 캐시와;

데이터 바이트들을 수신하도록 구성된 적재/저장 유닛과;

상기 적재/저장 유닛으로부터 상기 데이터 바이트들을 수신하여 저장하도록 구성된 데이터 캐시와; 그리고

복수의 캐시 라인들을 포함하는 레벨 2 캐시로 이루어지며,

여기서, 상기 각 캐시 라인은 희생 명령 바이트들 또는 희생 데이터 바이트들을 저장하도록 구성된 저장 영역(176) 및 에러 정정 코드 비트들 또는 패리티 정보 및 프리디코드 정보를 저장하도록 구성된 부가적인 저장 영역(174)을 포함하고;

상기 레벨 2 캐시는 상기 명령 캐시로부터 상기 희생 명령 바이트들을 수신하여 상기 복수의 캐시 라인들중 하나의 캐시 라인 내에 저장하고, 상기 데이터 캐시로부터 상기 희생 데이터 바이트들을 수신하여 상기 복수의 캐시 라인들중 부가적인 캐시 라인에 저장하도록 구성되고, 상기 레벨 2 캐시는 상기 저장된 희생 명령 바이트들에 대한 패리티 정보 및 프리디코드 정보를 수신하여 상기 하나의 캐시 라인 내에 포함된 부가적인 저장 영역에 저장하도록 구성되며, 상기 레벨 2 캐시는 상기 저장된 희생 데이터 바이트들에 대한 에러 정정 코드 비트들을 수신하여 상기 부가적인 캐시 라인 내에 포함된 부가적인 저장 영역 내에 저장하도록 구성되는 것을 특징으로 하는 마이크로프로세서.

**청구항 2**

제 1 항에 있어서,

상기 레벨 2 캐시에 전송되는 명령 바이트들에 대한 패리티 비트들을 발생시키고, 상기 레벨 2 캐시로부터 전송되는 명령 바이트들에 대한 패리티 비트들을 검사하도록 구성된 패리티 발생 및 검사 논리를 더 포함하는 것을 특징으로 하는 마이크로프로세서.

**청구항 3**

제 1 항에 있어서,

상기 레벨 2 캐시에 전송되는 데이터 바이트들에 대한 에러 정정 코드 비트들을 발생시키고, 상기 레벨 2 캐시로부터 전송되는 데이터 바이트들에 대한 에러 정정 코드 비트들을 검사하도록 구성된 에러 검사 및 정정 논리를 더 포함하는 것을 특징으로 하는 마이크로프로세서.

**청구항 4**

제 3 항에 있어서,

상기 에러 검사 및 정정 논리는 상기 레벨 2 캐시로부터 전송되는 데이터 바이트들 내의 적어도 하나의 비트 에러들을 정정하기 위해 상기 에러 정정 코드 비트들을 이용하도록 구성되는 것을 특징으로 하는 마이크로프로세서.

**청구항 5**

제 4 항에 있어서,

상기 각 캐시 라인은, (a) 명령 바이트들이 상기 캐시 라인의 저장 영역에 저장되어 있고, 프리디코드 비트들 및 패리티 비트가 상기 캐시 라인의 부가적인 저장 영역에 저장되어 있거나, 또는 (b) 데이터 바이트들이 상기 캐시 라인의 저장 영역에 저장되어 있고, 에러 정정 코드 비트들이 상기 캐시 라인의 부가적인 저장 영역에 저장되어 있는 지의 여부를 나타내는 1개 이상의 표시자 비트들을 저장하도록 구성되는 것을 특징으로 하는 마이크로프로세서.

**청구항 6**

명령 바이트들 및 데이터 바이트들을 수신하고, 상기 명령 바이트들에 의해 형성되는 명령들에 따라 상기 데이터 바이트들에 대해 동작하도록 구성된 프로세서와; 그리고

복수의 캐시 라인들을 포함하는 캐시로 이루어지며,

여기서, 상기 각 캐시 라인은 희생 명령 바이트들 또는 희생 데이터 바이트들을 저장하도록 구성된 저장 영역(176) 및 에러 정정 코드 비트들 또는 패리티 정보 및 프리디코드 정보를 저장하도록 구성된 부가적인 저장 영역(174)을 포함하고;

상기 캐시는 상기 프로세서로부터 상기 희생 명령 바이트들을 수신하여 상기 복수의 캐시 라인들중 하나의 캐시 라인 내에 저장하고, 상기 프로세서로부터 상기 희생 데이터 바이트들을 수신하여 상기 복수의 캐시 라인들중 부가적인 캐시 라인에 저장하도록 구성되고, 상기 캐시는 상기 저장된 희생 명령 바이트들에 대한 패리티 정보 및 프리디코드 정보를 수신하여 상기 하나의 캐시 라인 내에 포함된 부가적인 저장 영역에 저장하도록 구성되며, 그리고 상기 캐시는 상기 저장된 희생 데이터 바이트들에 대한 에러 정정 코드(ECC) 비트들을 수신하여 상기 부가적인 캐시 라인 내에 포함된 부가적인 저장 영역 내에 저장하도록 구성되는 것을 특징으로 하는 장치.

**청구항 7**

제 6 항에 있어서,

상기 프로세서가 상기 희생 데이터 바이트들을 요청하면, 상기 캐시는 상기 희생 데이터 바이트들 및 상기 저장된 대응하는 ECC 비트들을 상기 프로세서에 제공하도록 구성되는 것을 특징으로 하는 장치.

**청구항 8**

제 6 항에 있어서,

상기 프로세서가 상기 희생 명령 바이트들을 요청하면, 상기 캐시는 상기 희생 명령 바이트들, 상기 저장된 대응하는 패리티 정보 및 상기 저장된 대응하는 프리디코드 정보를 상기 프로세서에 제공하도록 구성되는 것을 특징으로 하는 장치.

**청구항 9**

제 8 항에 있어서,

상기 프로세서는 새로운 프리디코드 정보를 발생시키는 대신에, 상기 전달된 프리디코드 정보를 이용하도록 구성되는 것을 특징으로 하는 장치.

**청구항 10**

제 6 항에 있어서,

상기 캐시는 상기 프로세서와 공통 다이 위에 구현되는 레벨 2 캐시인 것을 특징으로 하는 장치.

**청구항 11**

제 6 항에 있어서,

상기 캐시는 상기 프로세서와 다른 다이 위에 구현되는 레벨 2 캐시인 것을 특징으로 하는 장치.

**청구항 12**

제 11 항에 있어서,

상기 각 캐시 라인은, 논리 블록들이 명령 바이트들에 대한 프리디코드 비트를 저장하고 있는지, 아니면 데이터 바이트들에 대한 에러 검사 및 정정 비트들을 저장하고 있는 지를 나타내는 표시자 비트를 저장하도록 구성되는 것을 특징으로 하는 장치.

**청구항 13**

제 6 항에 있어서,

상기 각 캐시 라인은, (a) 명령 바이트들이 상기 캐시 라인의 저장 영역에 저장되어 있고, 프리디코드 비트들 및 패리티 비트가 상기 캐시 라인의 부가적인 저장 영역에 있거나, 또는 (b) 데이터 바이트들이 상기 캐시 라인의 저장 영역에 저장되어 있고, ECC 비트들이 상기 캐시 라인의 부가적인 저장 영역에 저장되어 있음을 나타내는 표시자 비트를 저장하도록 구성되는 것을 특징으로 하는 장치.

**청구항 14**

제 13 항에 있어서, 상기 프로세서는,

상기 캐시에 전송되는 명령 바이트들에 대한 패리티 비트들을 발생시키고, 상기 캐시로부터 전송되는 명령 바이트들에 대한 패리티 비트들을 검사하도록 구성된 패리티 발생 및 검사 논리를 더 포함하는 것을 특징으로 하는 장치.

**청구항 15**

제 6 항에 있어서, 상기 프로세서는,

상기 캐시에 전송되는 데이터 바이트들에 대한 ECC 비트들을 발생시키고, 상기 캐시로부터 전송되는 데이터 바이트들에 대한 ECC 비트들을 검사하도록 구성된 에러 검사 및 정정 논리를 더 포함하는 것을 특징으로 하는 장치.

**청구항 16**

제 15 항에 있어서,

상기 에러 검사 및 정정 논리는 상기 캐시로부터 전송되는 데이터 바이트들 내의 적어도 하나의 비트 에러들을 정정하기 위해 상기 ECC 비트들을 이용하도록 구성되는 것을 특징으로 하는 장치.

**청구항 17**

명령 바이트들을 수신하는 단계와;

상기 명령 바이트들에 대한 프리디코드 정보를 발생시키는 단계와;

상기 명령 바이트들 및 상기 프리디코드 정보를 제 1 메모리에 저장하는 단계와;

상기 명령 바이트들이 상기 제 1 메모리 내에서 오버라이트되는 것에 응답하여, 패리티 정보와 함께 상기 프리디코드 정보 및 상기 명령 바이트들의 적어도 일부를 제 2 메모리에 출력하는 단계와, 여기서 상기 제 2 메모리는 상기 제 2 메모리 내에 포함되어 있는 복수의 캐시 라인들중 하나의 캐시 라인 내에 포함된 저장 영역에 상기 명령 바이트들을 저장하고, 상기 하나의 캐시 라인 내에 포함되어 있는 부가적인 저장 영역에 상기 프리디코드 정보를 저장하며;

데이터 바이트들을 수신하는 단계와;

상기 데이터 바이트들을 제 3 메모리에 저장하는 단계와; 그리고

상기 데이터 바이트들이 상기 제 3 메모리 내에서 오버라이트되는 것에 응답하여, 대응하는 에러 정정 코드 정보와 함께 상기 데이터 바이트들의 적어도 일부를 상기 제 2 메모리에 출력하는 단계를 포함하며,

여기서, 상기 제 2 메모리는 상기 제 2 메모리 내에 포함되어 있는 복수의 캐시 라인들중 부가적인 캐시 라인 내에 포함된 저장 영역에 상기 데이터 바이트들을 저장하고, 상기 부가적인 캐시 라인 내에 포함되어 있는 부가적인 저장 영역에 상기 에러 정정 코드 정보를 저장하는 것을 특징으로 하는 방법.

**청구항 18**

제 17 항에 있어서,

상기 방법은 상기 제 2 메모리의 각 캐시 라인에 표시자 비트를 저장하는 단계를 더 포함하고, 상기 표시자 비

트는 프리디코드 비트들 또는 에러 정정 코드 비트들이 상기 제 2 메모리에 저장되어 있는 지의 여부를 나타내는 것을 특징으로 하는 방법.

**청구항 19**

제 18 항에 있어서,

상기 제 2 메모리의 각 논리 블럭에 대해 표시자 비트를 저장하는 단계를 더 포함하며, 상기 표시자 비트는 프리디코드 비트들 또는 에러 검사 및 정정 비트들이 상기 제 2 메모리 내에 저장되어 있는 지의 여부를 나타내는 것을 특징으로 하는 방법.

**청구항 20**

제 18 항에 있어서,

적어도 1개의 저장된 패리티 비트와, 상기 프리디코드 정보 및 상기 명령 바이트들의 대응하는 저장된 부분을 프로세서에 전달하는 단계를 더 포함하고,

상기 패리티 비트가 정확한 경우, 상기 프로세서는 새로운 프리디코드 정보를 발생시키는 대신, 상기 전달된 프리디코드 정보를 이용하도록 구성되는 것을 특징으로 하는 방법.

**청구항 21**

주 시스템 메모리와;

상기 주 시스템 메모리에 결합된 메모리 제어기와;

상기 메모리 제어기에 결합되어, 명령 바이트들 및 데이터 바이트들을 수신하고, 상기 명령 바이트들에 의해 형성되는 명령들에 따라 상기 데이터 바이트들에 대해 동작하도록 구성된 마이크로프로세서와; 그리고

복수의 캐시 라인들을 포함하는 캐시로 이루어지며,

여기서, 상기 각 캐시 라인은 희생 명령 바이트들 또는 희생 데이터 바이트들을 저장하도록 구성된 저장 영역(176) 및 에러 정정 코드 비트들 또는 패리티 정보 및 프리디코드 정보를 저장하도록 구성된 부가적인 저장 영역(174)을 포함하고;

상기 캐시는 상기 마이크로프로세서로부터 상기 희생 명령 바이트들을 수신하여 상기 복수의 캐시 라인들중 하나의 캐시 라인 내에 저장하고, 상기 마이크로프로세서로부터 상기 희생 데이터 바이트들을 수신하여 상기 복수의 캐시 라인들중 부가적인 캐시 라인에 저장하도록 구성되고, 상기 캐시는 상기 저장된 희생 명령 바이트들에 대한 패리티 정보 및 프리디코드 정보를 수신하여 상기 하나의 캐시 라인 내에 포함된 부가적인 저장 영역에 저장하도록 구성되며, 그리고 상기 캐시는 상기 저장된 희생 데이터 바이트들에 대한 에러 정정 코드 비트들을 수신하여 상기 부가적인 캐시 라인 내에 포함된 부가적인 저장 영역 내에 저장하도록 구성되는 것을 특징으로 하는 컴퓨터 시스템.

**청구항 22**

제 21 항에 있어서,

상기 메모리 제어기는 상기 주 시스템 메모리로/로부터 전송되는 바이트들에 대한 패리티를 발생시키고 검사하도록 구성되는 패리티 발생 및 검사 논리를 더 포함하는 것을 특징으로 하는 컴퓨터 시스템.

**청구항 23**

제 21 항에 있어서,

상기 메모리 제어기는 상기 주 시스템 메모리로/로부터 전송되는 바이트들에 대한 에러 정정 코드 비트들을 발생시키고 검사하도록 구성되는 에러 정정 코드 발생 및 검사 논리를 더 포함하는 것을 특징으로 하는 컴퓨터 시스템.

## *명 세 서*

## 기 술 분 야

<1> 본 발명은 마이크로프로세서들에 관한 것으로서, 특히 마이크로프로세서 내에서의 가변 길이 명령들의 디코딩에 관한 것이다.

## 배 경 기 술

<2> x86 명령 세트에 대해 기록되는 소프트웨어 애플리케이션들의 수는 상당히 많다. 결과적으로, 보다 새롭고 보다 진보한 명령 세트들의 도입에도 불구하고, 마이크로프로세서 설계자들은 x86 명령 세트를 실행할 수 있는 마이크로프로세서들을 계속해서 설계해왔다.

<3> x86 명령 세트는 비교적 복잡하며, 다수의 가변 길이 명령들로 특징화된다. 도 1은 x86 명령 세트를 예시하는 일반적인 포맷을 도시한다. 도 1에 도시된 바와 같이, x86 명령은 1개 내지 5개의 선택적인 프리픽스 바이트들(102), 연산 코드(opcode) 필드(104), 선택적인 어드레싱 모드(Mod R/M) 바이트(106), 선택적인 스케일 인덱스 베이스(SIB) 바이트(108), 선택적인 변위 필드(110) 및 선택적인 즉시 데이터 필드(112)로 이루어진다.

<4> 연산 코드 필드(104)는 특정한 명령에 대한 기본 연산을 정의한다. 특정한 연산 코드의 디폴트 연산은 1개 이상의 프리픽스 바이트들(102)에 의해 변경될 수 있다. 예를 들어, 프리픽스 바이트들(102)중 하나는, 메모리 어드레싱에 이용되는 디폴트 세그먼트를 무효(override)로 하기 위해, 또는 프로세서에게 스트링 연산을 다수회 반복할 것을 명령하기 위해, 한 명령에 대한 어드레스 또는 오퍼랜드 사이즈를 바꾸는 데에 이용될 수 있다. 연산 코드 필드(104)는 만일 존재하는 경우 프리픽스 바이트들(102)의 뒤를 따르며, 1 또는 2 바이트의 길이를 가질 수 있다. 어드레싱 모드(Mod R/M) 바이트(106)는 이용되는 레지스터들 및 메모리 어드레싱 모드들을 지정한다. 스케일 인덱스 베이스(SIB) 바이트(108)는 스케일 및 인덱스 팩터들을 이용하는 32 비트의 베이스 관련 어드레싱(base-relative addressing)에만 이용된다. SIB 바이트(108) 내의 베이스 필드는 어떤 레지스터가 어드레스 계산을 위한 베이스 값을 포함하는 지를 지정하고, SIB 바이트(108) 내의 인덱스 필드는 어떤 레지스터가 인덱스 값을 포함하는 지를 지정한다. SIB 바이트(108) 내의 스케일 필드는, 인덱스 값이 어떠한 변위와 함께 베이스 값에 더해지기 전에 이 인덱스 값에 곱해지는 제곱(power of two)을 지정한다. 다음 명령 필드는 변위 필드(110)이다. 이는 선택적이며, 1 내지 4 바이트의 길이를 가질 수 있다. 이 변위 필드(110)는 어드레스 계산에 이용되는 상수를 포함한다. 선택적인 즉시 필드(112)(이 또한 1 내지 4 바이트의 길이를 갖는다)는 명령 오퍼랜드로서 이용되는 상수를 포함한다. 가장 짧은 x86 명령들은 단지 1 바이트의 길이를 가지며 단일 연산 코드 바이트를 포함한다. 80286 명령 세트들은 한 명령의 길이가 최대 10 바이트들이며, 80386 및 80486은 최대 15 바이트의 명령 길이를 가질 수 있다.

<5> x86 명령 세트의 복잡성은 고성능의 x86 호환성 마이크로프로세서들을 구현하는 데에 많은 어려움들을 야기시킨다. 특히, x86 명령들의 가변 길이는 명령들의 디코딩을 어렵게 한다. 명령들의 디코딩은 전형적으로 명령의 경계들을 결정한 다음, 명령 내의 각 필드(예를 들어, 연산 코드 및 오퍼랜드 필드들)를 식별할 것을 요구한다. 전형적으로, 디코딩은 실행 전에 명령이 명령 캐시로부터 페치될 때 일어난다.

<6> 명령들의 경계들을 결정하는 한 방법은, 각 명령 바이트에 대한 1개 이상의 프리디코드 비트들을 발생시키고 저장하는 단계를 포함하지만, 실제로 이들은 주 메모리로부터 판독된 다음 명령 캐시 내에 저장된다. 상기 프리디코드 비트들은 이들이 관련된 명령 바이트에 대한 정보를 제공한다. 예를 들어, 표명(assertion)된 프리디코드 시작 비트는 관련된 명령 바이트가 명령의 첫 번째 바이트임을 나타낸다. 일단 특정한 명령 바이트에 대한 시작 비트가 계산되면, 이는 명령 바이트와 함께 명령 캐시에 저장된다. "페치(fetch)"가 수행될 때, 다수의 명령 바이트들이 명령 캐시로부터 판독된 다음, 실행을 준비하기 위해 디코드된다. 페치시 개별적인 명령들에 대한 유효 마스크들을 발생시키기 위해, 어떠한 관련된 시작 비트들이 스캔된다. 유효 마스크는 일련의 비트들인바, 여기서 각 비트는 특정한 명령 바이트에 대응한다. 명령의 첫 번째 바이트, 명령의 마지막 바이트, 및 명령의 첫 번째 바이트와 마지막 바이트 사이의 모든 바이트들에 관련된 유효 마스크 비트들이 표명된다. 다른 모든 유효 마스크 비트들은 표명되지 않는다. 일단 유효 마스크가 계산되면, 이는 다른 명령들로부터 바이트들을 마스크 오프(mask off)하는 데에 이용될 수 있다.

<7> 도 2는 예시적인 유효 마스크를 도시한다. 도 2는 페치(120)의 일부분 및 그의 관련 시작 비트들(122)을 도시한다. 명령 B(128)에 대한 유효 마스크(126)가 발생된다고 가정하면, 시작 비트(122A) 및 이 시작 비트(122A)와 시작 비트(122B) 간의 모든 비트들이 마스크(126)를 발생시키기 위해 표명된다. 일단 발생되면, 유효 마스크(126)는 명령 B(128)의 일부가 아닌 페치(120) 내의 모든 바이트들을 마스크 오프하는 데에 이용될 수 있다.

<8> 상기 설명한 바와 같이, 프리디코드 정보는 디코드 횟수(decode times)를 줄이는 데에 특히 유용하다. 명령 캐시에 명령 바이트들과 함께 프리디코드 정보를 저장함으로써, 대응하는 명령이 (예를 들어, 루프 내에서) 다수회 실행된다고 할지라도, 프리디코드 정보는 단지 한번만 계산되면 된다. 그러나, 불행히도, 명령이 명령 캐시로부터 교체되거나 버려질 때, 어떠한 관련된 프리디코드 정보를 잃어버리게 된다. 다음에 명령이 명령 캐시 내로 판독될 때, 프리디코드 정보가 다시 한번 발생되어야 한다. 프리디코드 정보가 계산 완료되는 것을 기다림으로써 야기되는 시간 지연은 특히, 분기 오예측 또는 캐시 미스(cache miss)의 결과로서 명령이 명령 캐시 내로 판독될 때 성능에 손상을 줄 수 있다. 요구되기 전에 추론적으로 프리페치되는 명령들과 대조적으로, 분기 오예측들 또는 캐시 미스들로부터 비롯되는 페치들은 요구되는 명령들의 수신을 기다리는 동안 마이크로프로세서의 디코더들 및 기능 유닛들을 정체(stall)시킬 수 있다. 이 경우, 프리디코드 정보를 생성하는 데에 필요한 시간은 마이크로프로세서의 성능에 상당한 영향을 줄 수 있다.

<9> 상기 및 다른 이유들로 인해, 프리디코드 횟수를 줄이기 위한 방법 및 장치가 필요하다. 특히, 명령 캐시로부터 이전에 버려진 명령들에 대한 프리디코드 정보를 발생시키는 데에 필요한 시간을 줄이는 방법 및 장치가 필요하다.

<10> Rupaka Mahalingaiah의 미국 특허 제6,092,182호(명칭: "프리디코드 정보를 저장하는 데에 ECC/패리티 비트들을 이용")에서는, 이러한 문제들에 대한 하나의 가능한 해결책, 즉 레벨 2 캐시의 ECC/패리티 비트들에 프리디코드 정보를 저장하는 것을 제시했다. 유익하게는, 어떠한 경우들에서는, 희생 명령 바이트들(victimized instruction bytes)의 프리디코딩의 지연이 무시(bypass)될 수 있다.

<11> 그러나, 제시된 해결책은 레벨 2 캐시에 저장된 데이터를 단일 및 다수 비트 에러들로부터 보호하지 못한다. 현재의 마이크로프로세서들의 보다 높은 동작 주파수들 및 집적 레벨을 가정하면, 특히 캐시들에서의 저장 에러들이 문제가 될 수 있다. 따라서, 보다 빈번하게 발생되는 타입의 에러들(예를 들어, 단일 비트 에러들)의 일부를 최소한 검출할 수 있는 성능을 여전히 보유하면서, 버려지는 프리디코드 정보의 양을 줄일 수 있는 시스템 및 방법이 필요하다.

## *발명의 상세한 설명*

<12> 상기 문제들은 본원에서 설명되는 희생된 명령 프리디코드 정보를 저장하는 시스템 및 방법에 의해 해결된다. 일 실시예에서, 희생된 명령 프리디코드 정보를 저장하도록 구성된 마이크로프로세서는 프리디코드 유닛 및 적재/저장 유닛을 포함할 수 있다. 프리디코드 유닛은 명령 바이트들을 수신한 다음, 대응하는 프리디코드 정보를 발생시키도록 구성될 수 있다. 이 프리디코드 유닛은 또한 명령 캐시에 명령 바이트들 및 대응하는 프로디코드 정보를 저장하도록 구성될 수 있다. 적재/저장 유닛은 데이터 바이트들을 수신한 다음, 이 데이터 바이트들을 데이터 캐시에 저장하도록 구성될 수 있다. 명령 캐시 및 데이터 캐시가 함께 프로세서를 위한 "레벨 1" 캐시를 구성할 수 있다. 프로세서는 또한 다수의 캐시 라인들을 포함하는 레벨 2 캐시를 포함할 수 있는바, 각 캐시 라인은 희생 명령 바이트들 또는 희생 데이터 바이트들을 저장하도록 구성된 제 1 저장 영역, 및 에러 정정 코드 비트들 또는 패리티 정보 및 프리디코드 정보를 저장하도록 구성된 제 2 저장 영역을 포함한다. 레벨 2 캐시는 다수의 캐시 라인들중 제 1 캐시 라인의 명령 캐시로부터의 희생 명령 바이트들, 및 다수의 캐시 라인들중 제 2 캐시 라인의 데이터 캐시로부터의 희생 데이터 바이트들을 수신하여 저장하도록 구성된다. 레벨 2 캐시는 또한 저장된 희생 명령 바이트들에 대한 패리티 정보 및 프리디코드 정보를 수신하여 제 1 캐시 라인에 포함된 제 2 저장 영역에 저장하고, 저장된 희생 데이터 바이트들에 대한 에러 정정 코드(ECC) 비트들을 수신하여 제 2 캐시 라인에 포함된 제 2 저장 영역에 저장하도록 구성된다.

<13> 일 실시예에서, 마이크로프로세서는 또한 레벨 2 캐시에 저장된 명령 바이트들에 대한 패리티 비트들을 발생시키도록 구성된 패리티 발생 및 검사 논리를 포함할 수 있다. 이 패리티 발생 및 검사 논리는 또한 레벨 2 캐시로부터 판독된 명령 바이트들에 대해 패리티 비트들을 검사하도록 구성될 수 있다. 일부 실시예들에서, 명령 캐시는 또한 패리티 정보를 저장하도록 구성됨으로써, 명령 캐시 및 레벨 2 캐시 모두의 명령 바이트들에 대해 패리티 검사를 할 수 있다.

<14> 유사하게, 마이크로프로세서는 또한 레벨 2 캐시에 저장된 데이터 바이트들에 대해 에러 검사 및 정정 코드 비트들을 발생시키도록 구성된 에러 검사 및 정정 논리를 포함할 수 있다. 이 에러 검사 및 정정 논리는 레벨 2 캐시로부터 판독된 데이터 바이트들에 대해 ECC 비트들을 검사하도록 구성될 수 있다. 일부 실시예들에서, 상기 에러 검사 및 정정 논리는 레벨 2 캐시가 아닌 데이터 캐시로부터 판독된 데이터 바이트들의 적어도 1개의 비트 에러들을 정정하도록 구성될 수 있다.

<15> 일부 실시예들에서, 레벨 2 캐시는 캐시 라인들(예를 들어, 논리 로우들 그리고/또는 칼럼들)로 분할될 수 있는바, 각 캐시 라인은 희생 데이터 또는 희생 명령 바이트들을 저장하도록 구성된다. 표시자 비트(indicator bit)(이는 또한 "데이터 타입" 비트라고도 칭해짐)가 레벨 2 캐시의 각 캐시 라인에서 이용될 수 있는바, 이는 (a) 희생 명령 바이트들, 프리디코드 정보 및 패리티 정보, 또는 (b) ECC 정보 및 데이터 바이트들이 저장되어 있는 지를 나타낸다. 어떠한 실시예들에서, 레벨 2 캐시는 희생 명령 및 데이터 바이트들과 그들의 대응하는 ECC 비트들, 패리티 비트들, 프리디코드 비트들 및 표시자 비트들 만을 저장한다는 점에서 배타적(exclusive)이다. 다른 실시예들에서, 레벨 2 캐시는 아직 희생되지 않은 명령 바이트들 및 데이터 바이트들의 카피들을 또한 저장한다는 점에서 포괄적(inclusive)일 수 있다.

<16> 또한, 희생 프리디코드 정보를 저장하는 시스템이 고려된다. 일 실시예에서, 이 시스템은 명령 및 데이터 바이트들을 수신하도록 구성된 프로세서를 포함할 수 있다. 이 프로세서는 명령 바이트들에 의해 형성된 명령들에 따라 데이터 바이트에 대해 동작하도록 구성될 수 있다. 이 시스템은 또한 프로세서로부터 희생 명령 바이트들 및 희생 데이터 바이트들을 수신하여 저장하도록 구성된 캐시를 포함할 수 있다. 이 캐시는 희생 명령 바이트들에 대한 패리티 정보 및 프리디코드 정보를 수신하여 저장하도록 구성될 수 있다. 이 캐시는 또한 저장된 희생 데이터 바이트들에 대한 ECC 비트들을 수신하여 저장하도록 구성될 수 있다. 이 캐시는 희생 데이터 바이트들을 요구하는 프로세서에 응답하여 희생 데이터 바이트들 및 대응하는 ECC 비트들을 프로세서에 제공하도록 구성될 수 있다. 유사하게, 캐시는 희생 명령 바이트들을 요구하는 프로세서에 응답하여 저장된 희생 명령 바이트들, 대응하는 패리티 및 프리디코드 정보를 프로세서에 제공하도록 구성될 수 있다. 유익하게는, 프로세서는 명령 바이트들에 대해 새로운 프리디코드 정보를 재생하는 대신 명령 캐시로부터의 프리디코드 정보를 이용할 수 있다. 명령 바이트들(예를 들어, 자기 변경 코드)에 대한 기록의 경우, (예를 들어, 프리디코드 정보에 대해 무효화 상수를 저장함으로써) 저장된 프리디코드 정보는 무효화되며, 새로운 패리티 정보가 계산될 수 있다.

<17> 일부 실시예들에서, 캐시는 프로세서와 공통 다이 위에 형성되는 레벨 2 캐시로서 구현될 수 있다. 다른 실시예들에서, 캐시는 프로세서와 다른 다이 위에 형성될 수 있다. 일부 실시예들에서, 캐시는 희생 명령 바이트들(및 대응하는 프리디코드 및 패리티 정보)를 저장하도록 구성된 제 1 섹션, 및 희생 데이터 바이트들 및 대응하는 ECC 정보를 저장하도록 구성된 제 2 섹션으로 분할될 수 있다.

<18> 또한, 희생 프리디코드 정보를 저장하는 방법이 고려된다. 일 실시예에서, 이 방법은 명령 바이트들을 수신하는 단계 및 그 명령 바이트에 대응하는 프리디코드 정보를 발생시키는 단계를 포함한다. 이후, 명령 바이트들 및 프리디코드 정보가 제 1 메모리에 저장될 수 있다. 명령 바이트들이 제 1 메모리 내에서 오버라이트되는 것에 응답하여, 적어도 일부의 명령 바이트들 및 프리디코드 정보가 제 2 메모리로 출력될 수 있다. 제 2 메모리에 저장된 프리디코드 정보 및 명령 바이트들에 대응하는 적어도 1개의 패리티 비트가 발생되어 제 2 메모리에 저장될 수 있다. 이 방법은 또한 데이터 바이트들을 수신하는 단계 및 이 데이터 바이트들을 제 3 메모리에 저장하는 단계를 포함한다. 데이터 바이트들의 적어도 일부는 데이터 바이트들이 제 3 메모리 내에서 오버라이트되는 것에 응답하여 제 2 메모리에 저장될 수 있다. 데이터 바이트들은 ECC 정보를 가질 수 있는바, 이 ECC 정보는 발생되어 데이터 바이트들과 함께 제 2 메모리에 저장된다. 제 2 메모리 내의 각 캐시 라인에 대한 표시자 비트들이 또한 발생되어 그 내에 저장될 수 있는바, 이 표시자 비트들은 프리디코드 비트들 및 패리티 정보를 갖는 명령 바이트들이 제 2 메모리에 저장되어 있는지, 아니면 ECC 정보를 갖는 데이터 바이트들이 제 2 메모리에 저장되어 있는 지를 나타낸다.

<19> 본 발명의 다른 목적들 및 장점들은 첨부 도면들을 참조하여 설명되는 하기의 상세한 설명으로부터 명확해질 것이다.

## *실 시 예*

<31> 도 3은 마이크로프로세서(10)의 일 실시예의 블록도이다. 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 스테이션들(22A-22C), 다수의 기능 유닛들(24A-24C), 적재/저장 유닛(26), 데이터 캐시(28), 레지스터 파일(30), 재배열 버퍼(32) 및 MROM 유닛(34)을 포함한다. 본원에서 뒤에 글자가 붙은 특정한 참조 번호로 설명되는 요소들은 집합적으로 그 참조 부호 만으로 칭한다. 예를 들어, 예약 스테이션들(22A-22C)은 집합적으로 예약 스테이션들(22)로 칭할 수 있다.

<32> 프리페치/프리디코드 유닛(12)은 주 메모리 서브 시스템(미도시)으로부터 명령들을 수신하도록 결합되며, 또한 명령 캐시(16) 및 분기 예측 유닛(14)에 결합된다. 유사하게, 분기 예측 유닛(14)은 명령 캐시(16)에 결합된다.

분기 예측 유닛(14)은 또한 명령 정렬 유닛(18) 및 기능 유닛들(24A-24C)에 결합된다. 명령 캐시(16)는 또한 MROM 유닛(34) 및 명령 정렬 유닛(18)에 결합된다. 이 명령 정렬 유닛(18)은 또한 적재/저장 유닛(26) 및 각 디코드 유닛들(20A-20C)에 결합된다. 각 디코드 유닛들(20A-20C)은 예약 스테이션들(22A-22C)에 결합되며, 이 예약 스테이션들(22A-22C)은 또한 각 기능 유닛들(24A-24C)에 결합된다. 또한, 명령 정렬 유닛(18) 및 예약 스테이션들(22)은 레지스터 파일(30) 및 재배열 버퍼(32)에 결합된다. 기능 유닛들(24)은 또한 적재/저장 유닛(26), 레지스터 파일(30) 및 재배열 버퍼(32)에 결합된다. 데이터 캐시(28)는 적재/저장 유닛(26) 및 주 메모리 서브시스템에 결합된다. 마지막으로, MROM 유닛(34)은 명령 정렬 유닛(18)에 결합된다.

<33> 명령들은 프리페치/프리디코드 유닛(12)에 의해 주 메모리로부터 프리페치된다. 프리페치/프리디코드 유닛(12)은 가변 길이 명령들을 고정 길이 명령들로 프리디코드하는바, 상기 고정 길이 명령들은 이후 명령 캐시(16)에 저장된다. 명령들은, 이들이 프리페치 방식을 이용하여 실제로 요구되기 전에, 프리페치 및 프리디코드될 수 있다. 프리페치/프리디코드 유닛(12)에 의해 많은 프리페치 방식들이 이용될 수 있다. 프리디코드 유닛(12) 및 명령 캐시(16)를 보다 상세히 설명하기에 앞서, 도면에 도시된 예시적인 마이크로프로세서(10)의 실시예에 대한 일반적인 양상들에 대해 설명한다.

<34> 마이크로프로세서(10)는 분기 예측을 이용하여, 조건 분기 명령들 이후의 명령들을 추론적으로 페치한다. 분기 예측 유닛(14)은 분기 예측 동작들을 수행한다. 일 실시예에서는, 최대 2개의 분기 타겟 어드레스들이 명령 캐시(16)의 각 캐시 라인의 각 16 바이트 부분에 대해 저장된다. 프리페치/프리디코드 유닛(12)은, 특정한 라인이 프리디코드될 때, 처음 분기 타겟들을 결정한다. 캐시 라인에 대응하는 분기 타겟들에 대한 이후의 갱신들은 캐시 라인 내에서의 명령들의 실행으로 인해 이루어질 수 있다. 명령 캐시(16)는 페치되는 명령 어드레스의 표시를 분기 예측 유닛(14)에 제공한다. 이에 의해, 분기 예측 유닛(14)은 분기 예측을 형성할 때 어떤 분기 타겟 어드레스들을 선택할 것인 지를 결정할 수 있다. 명령 정렬 유닛(18) 및 기능 유닛들(24)은 분기 예측 유닛(14)에 갱신 정보를 제공한다. 분기 예측 유닛(14)은 캐시 라인의 16 바이트 부분 마다 2개의 분기 타겟들을 저장하기 때문에, 라인 내에서의 일부 분기 명령들에 대한 예측들은 분기 예측 유닛(14)에 저장되지 않을 수 있다. 명령 정렬 유닛(18)은 분기 예측 유닛(14)에 의해 예측되지 않았던 분기 명령들을 검출하도록 구성될 수 있다. 기능 유닛들(24)은 분기 명령들을 실행시키고, 예측된 분기 방향이 오예측되었는 지를 결정한다. 분기 방향은 "테이큰(taken)"이 될 수 있는바, 이 경우 이후 명령들은 분기 명령의 타겟 어드레스로부터 페치된다. 반대로, 분기 방향은 "낫 테이큰(not taken)"이 될 수 있는바, 이 경우 이후 명령들은 분기 명령에 연속하는 메모리 위치들로부터 페치된다. 오예측된 분기 명령이 검출될 때, 이 오예측된 분기 이후의 명령들은 마이크로프로세서(10)의 다양한 유닛들로부터 버려진다. 적절한 많은 분기 예측 알고리즘들이 분기 예측 유닛(14)에 의해 이용될 수 있다.

<35> 명령 캐시(16)는 프리페치/프리디코드 유닛(12)으로부터 수신된 명령들을 저장하도록 제공되는 고속의 캐시 메모리이다. 이후, 저장된 명령들은 명령 캐시(16)로부터 페치되어 명령 정렬 유닛(18)에 전송된다. 일 실시예에서, 명령 캐시(16)는 세트 어소시에이티브 구조(set-associative structure)로서 구성될 수 있다. 명령 캐시(16)는 액세스 시간의 속도를 더하기 위해 방향 예측 방식(way prediction scheme)을 추가적으로 이용할 수 있다. 예를 들어, 명령들의 각 라인을 식별하는 태그들을 액세스한 다음 이 태그들과 페치 어드레스를 비교하여 방향을 선택하는 대신, 명령 캐시(16)는 액세스되는 방향을 예측할 수 있다. 이러한 방식으로, 어레이를 액세스하기 전에 방향이 추론적으로 선택된다. 방향 예측을 이용하게 되면, 명령 캐시(16)의 액세스 시간은 직접 맵핑 캐시(direct-mapped cache)와 유사해질 수 있다. 명령 바이트들이 판독된 후에는, 검증을 위해 태그 비교가 수행된다. 방향 예측이 부정확하다면, 정확한 명령 바이트들이 페치되고 (프로세싱 파이프라인 더 아래에 있는) 부정확한 명령 바이트들은 버려진다. 주목할 사항으로서, 명령 캐시(16)는 완전 어소시에이티브(fully associative), 세트 어소시에이티브, 또는 직접 맵핑 구성들로 구현될 수 있다.

<36> MROM 유닛(34)은 "빠른 경로 명령들(fast-path instructions)"의 시퀀스들을 저장하도록 구성된 판독 전용 메모리이다. 빠른 경로 명령들은 디코더들(20A-20C) 및 기능 유닛들(24A-24C)에 의해 디코드되고 실행될 수 있는 명령들이다. 반대로, "MROM 명령들"은 디코더들(20A-20C) 및 기능 유닛들(24A-24C)에 의해 직접 디코딩 또는 실행되기에는 너무 복잡한 명령들이다. 명령 캐시(16)가 MROM 명령을 출력하면, MROM 유닛(34)은 빠른 경로 명령들의 시퀀스를 출력함으로써 응답한다. 보다 구체적으로, 바람직한 동작을 달성하기 위해, MROM 유닛(34)은 MROM 명령을 정의된 빠른 경로 명령들의 서브셋(subset)으로 분석(parse) 및 변환한다. MROM 유닛(34)은 빠른 경로 명령들의 서브셋을 디코드 유닛들(20A-20C)로 디스패치한다.

<37> 일단 명령 바이트들이 명령 캐시(16)로부터 페치되면, 이 명령 바이트들은 명령 정렬 유닛(18)으로 전달된다. 명령 정렬 유닛(18)은 디코드 유닛들(20A-20C)중 하나로 명령들을 전송한다. 레지스터 오퍼랜드 정보가 또한 검

출되어 레지스터 파일(30) 및 재배열 버퍼(32)로 전송된다. 또한, 명령들이 1개 이상의 메모리 연산들의 수행을 요구한다면, 명령 정렬 유닛(18)은 메모리 연산들을 적재/저장 유닛(26)으로 디스패치한다. 각각의 디코드된 명령은 오퍼랜드 어드레스 정보, 및 그 명령에 포함될 수 있는 변위 또는 즉시 데이터와 함께 예약 스테이션들(22)로 디스패치된다.

<38> 마이크로프로세서(10)는 순서를 벗어난 실행(out-of-order execution)을 지원하며, 이에 따라 재배열 버퍼(32)를 이용하여 레지스터 판독 및 기록 동작들에 대한 최초 프로그램 시퀀스의 트랙을 유지하고, 레지스터 재명명(renaming)을 구현하며, 추론적인 명령 실행 및 분기 오예측 복구를 허용하고, 그리고 정확한 예외들을 용이하게 한다. 재배열 버퍼(32) 내의 일시 저장 위치는 레지스터의 갱신을 포함하는 명령의 디코드시 예약된다. 일시 저장 위치는 명령의 추론적인 실행으로부터 비롯되는 추론적인 레지스터 상태를 저장한다. 분기 예측이 부정확하다면, 오예측된 경로를 따라 추론적으로 실행된 명령들로부터의 결과들은 이들이 레지스터 파일(30)에 기록되기 전에 재배열 버퍼(32) 내에서 무효화될 수 있다. 유사하게, 특정한 명령이 예외를 야기시킨다면, 예외를 야기시키는 명령 이후의 명령들은 버려질 수 있다. 이러한 방식으로, 예외들은 "정확하다"(즉, 예외를 야기시키는 명령 이후의 명령들은 예외 이전에 완료되지 않는다). 주목할 사항으로서, 특정한 명령이 프로그램 순서에 있어서 자신 보다 앞선 명령들 이전에 실행된다면, 이 특정한 명령은 추론적으로 실행된다. 앞선 명령들은 분기 명령 또는 예외 야기 명령이 될 수 있는바, 이러한 경우 추론적인 결과들은 재배열 버퍼(32)에 의해 버려질 수 있다.

<39> 명령 정렬 유닛(18)의 출력들에 제공되는 디코드된 명령들 및 즉시 또는 변위 데이터는 각 예약 스테이션들(22)에 직접 전송된다. 일 실시예에서, 각 예약 스테이션(22)은 대응하는 기능 유닛에 발행(issue)되기를 기다리는 최대 3개의 미결 명령들(pending instructions)에 대한 명령 정보(즉, 디코드된 명령들 뿐 아니라 오퍼랜드 값들, 오퍼랜드 태그들 그리고/또는 즉시 데이터)를 홀딩할 수 있다. 주목할 사항으로서, 도면에 도시된 실시예에서, 각 예약 스테이션(22)은 전용 기능 유닛(24)에 결합된다. 따라서, 예약 스테이션들(22) 및 기능 유닛들(24)에 의해 3개의 전용 "발행 위치들"이 형성된다. 다시 말해, 발행 위치 0은 예약 스테이션(22A) 및 기능 유닛(24A)에 의해 형성된다. 예약 스테이션(22A)에 정렬 및 디스패치된 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.

<40> 특정한 명령의 디코드시, 필요한 오퍼랜드가 레지스터 위치라면, 레지스터 어드레스 정보가 재배열 버퍼(32) 및 레지스터 파일(30)에 동시에 전송된다. 당업자라면, x86 레지스터 파일이 8개의 32 비트 리얼 레지스터들(real registers)(즉, 전형적으로 EAX, EBX, ECX, EDX, EBP, ESI, EDI 및 ESP라 칭해짐)을 포함하는 것을 이해할 수 있다. x86 마이크로프로세서 아키텍쳐를 이용하는 마이크로프로세서(10)의 실시예들에서, 레지스터 파일(30)은 32 비트 리얼 레지스터들 각각에 대한 저장 위치들을 포함한다. MROM 유닛(34)에 의해 이용하기 위한 추가적인 저장 위치들이 레지스터 파일(30) 내에 포함될 수 있다. 재배열 버퍼(32)는, 이러한 레지스터들의 내용을 변경함으로써 순서를 벗어난 실행을 가능하게 하는 결과들에 대한 일시 저장 위치들을 포함한다. 디코드시 리얼 레지스터들중 하나의 내용을 변경하는 것으로 결정되는, 재배열 버퍼(32)의 일시 저장 위치는 각 명령에 대해 예약된다. 따라서, 특정 프로그램을 실행하는 동안 다양한 시점(point)들에서, 재배열 버퍼(32)는 소정 레지스터의 추론적으로 실행된 내용들을 포함하는 1개 이상의 위치들을 가질 수 있다.

<41> 소정 명령의 디코드에 이어서, 재배열 버퍼(32)가 상기 소정 명령에서 오퍼랜드로서 이용된 레지스터에 할당된 이전 위치 또는 위치들을 갖는 것으로 결정되면, 재배열 버퍼(32)는 대응하는 예약 스테이션에 1) 가장 최근에 할당된 위치의 값, 또는 2) 상기 값이 기능 유닛(이것이 결국 이전 명령을 실행하게 될 것이다)에 의해 생성되지 않았다면, 가장 최근에 할당된 위치에 대한 태그를 전송한다. 재배열 버퍼(32)가 소정의 레지스터에 대해 예약된 위치를 갖는 다면, 오퍼랜드 값(또는 재배열 버퍼 태그)은 레지스터 파일(30)로부터가 아닌 재배열 버퍼(32)로부터 제공된다. 재배열 버퍼(32)가 요구되는 레지스터에 대해 예약된 어떠한 위치도 갖지 않는다면, 상기 값은 레지스터 파일(30)로부터 직접 제공된다. 오퍼랜드가 메모리 위치에 대응한다면, 오퍼랜드 값은 적재/저장 유닛(26)을 통해 예약 스테이션에 제공된다.

<42> 일 특정 실시예에서, 재배열 버퍼(32)는 동시에 디코드된 명령들을 한 유닛(unit)으로서 저장 및 처리하도록 구성된다. 본원에서 이러한 구성은 "라인 지향(line-orientd)"이라 칭한다. 몇 개의 명령들을 함께 처리함으로써, 재배열 버퍼(32) 내에서 이용되는 하드웨어가 단순해질 수 있다. 예를 들어, 본 실시예에 포함된 라인 지향 재배열 버퍼는, 명령 정렬 유닛(18)에 의해 1개 이상의 명령들이 디스패치될 때 마다, 3개의 명령들에 관련된 명령 정보에 충분한 저장 장소(storage)를 할당한다. 대조적으로, 종래의 재배열 버퍼들에는, 실제로 디스패치되는 명령들의 수에 따라 가변량의 저장 장소가 할당된다. 이러한 가변량의 저장 장소를 할당하기 위해서는, 상당

히 많은 수의 논리 게이트들이 필요할 수 있다. 동시에 디코드된 명령들 각각이 실행될 때, 명령 결과들은 레지스터 파일(30) 내에 동시에 저장된다. 이후, 상기 저장 장소는 동시에 디코드된 명령들의 다른 세트에 할당하기 위해 비워진다. 또한, 명령 마다 이용되는 제어 논리 회로의 양이 감소되는데, 그 이유는 제어 논리가 몇 개의 동시 디코드된 명령들에 대해 공유(amortize)되기 때문이다. 특정한 명령을 식별하는 재배열 버퍼 태그는 2개의 필드들, 즉 라인 태그(line tag) 및 오프셋 태그(offset tag)로 분할될 수 있다. 라인 태그는 특정한 명령을 포함하는 동시 디코드된 명령들의 세트를 식별하고, 오프셋 태그는 세트 내의 어떤 명령이 특정한 명령에 대응하는 지를 식별한다. 주목할 사항으로서, 명령 결과들을 레지스터 파일(30) 내에 저장하고 대응하는 저장 장소를 비우는 것을 명령들의 "퇴거(retiring)"라 칭한다. 또한, 주목할 사항으로서, 마이크프로세서(10)의 다양한 실시예들에서는 어떠한 재배열 버퍼 구성이라도 이용될 수 있다.

<43> 상기 언급한 바와 같이, 예약 스테이션들(22)은 대응하는 기능 유닛(24)에 의해 명령들이 실행될 때 까지 이 명령들을 저장한다. 명령은, (i) 이 명령에 대한 오퍼랜드들이 제공되었고, 그리고 (ii) 동일한 예약 스테이션들(22A-22C) 내에 있으며 프로그램 순서에 있어서 상기 명령 이전의 명령들에 대한 오퍼랜드들이 아직 제공되지 않은 경우, 실행을 위해 선택된다. 주목할 사항으로서, 기능 유닛들(24)중 하나에 의해 명령이 실행될 때, 명령의 결과는 결과를 기다리는 어떠한 예약 스테이션들(22)로 직접 전달됨과 동시에, 재배열 버퍼(32)를 갱신하기 위해 전달된다(이 기술은 대개 "결과 전송(result forwarding)"이라 칭한다). 명령은 실행을 위해 선택되며, 관련된 결과가 전송되는 클럭 주기 동안 기능 유닛(24A-24C)에 전달된다. 이 경우, 예약 스테이션들(22)은 전송된 결과를 기능 유닛(24)으로 보낸다.

<44> 일 실시예에서, 각 기능 유닛들(24A-24C)은 덧셈 및 뺌셈의 정수 산술 연산들 뿐 아니라, 시프트, 회전, 논리 연산들 및 분기 연산들을 수행하도록 구성된다. 주목할 사항으로서, 부동 소수점 연산들을 수행하기 위한 부동 소수점 유닛(미도시)이 또한 이용될 수 있다. 이 부동 소수점 유닛은 MROM 유닛(34)으로부터 명령들을 수신한 다음 재배열 버퍼(32)와 통신하여 명령들을 완료하는 코프로세서(coprocessor)로서 동작할 수 있다. 또한, 기능 유닛들(24)은 적재/저장 유닛(26)에 의해 수행되는 적재 및 저장 메모리 연산들을 위한 어드레스 발생을 수행하도록 구성될 수 있다.

<45> 기능 유닛들(24) 각각은 또한 조건 분기 명령들의 실행에 대한 정보를 분기 예측 유닛(14)에 제공한다. 분기 예측이 부정확했다면, 분기 예측 유닛(14)은 명령 처리 파이프라인에 들어온 오예측된 분기 이후의 명령들을 파기(flush)하고, 명령 캐시(16) 또는 주 메모리로부터 필요한 명령들이 페치되게 한다. 주목할 사항으로서, 이러한 상황들에서, 최초 프로그램 시퀀스에 있어서, 오예측된 분기 명령 이후 발생된 명령들의 결과들(추론적으로 실행되어 적재/저장 유닛(26) 및 재배열 버퍼(32)에 일시 저장되었던 것들을 포함)은 버려진다.

<46> 기능 유닛들(24)에 의해 생성된 결과들은, 레지스터 값이 갱신되는 경우에는 재배열 버퍼(32)에 보내지고, 메모리 위치의 내용들이 변경되는 경우에는 적재/저장 유닛(26)에 보내진다. 결과가 레지스터에 저장되는 경우, 재배열 버퍼(32)는 명령이 디코드될 때 레지스터의 값에 대해 예약된 위치에 상기 결과를 저장한다. 기능 유닛들(24) 및 적재/저장 유닛(26)으로부터의 결과들을 전송하기 위한 다수의 결과 버스들(38)이 포함된다. 이 결과 버스들(38)은 발생된 결과 뿐 아니라, 실행될 명령을 식별하는 재배열 버퍼 태그를 전달한다.

<47> 적재/저장 유닛(26)은 기능 유닛들(24)과 데이터 캐시(28) 간에 인터페이스를 제공한다. 일 실시예에서, 적재/저장 유닛(26)은 미결 적재들 또는 저장들에 대한 데이터 및 어드레스 정보를 위한 8개의 저장 위치들을 갖는 적재/저장 버퍼를 구비하도록 구성된다. 버퍼가 가득차면, 명령 정렬 유닛(18)은 적재/저장 유닛(26)이 미결 적재 또는 저장 요구 정보를 위한 장소를 가질 때 까지 기다린다. 적재/저장 유닛(26)은 또한 데이터 코히런시(즉, 일관성)가 유지될 수 있도록, 미결 저장 메모리 연산들에 대한 적재 메모리 연산들의 의존성 체크를 수행한다. 메모리 연산은 마이크로프로세서(10)와 주 메모리 서브 시스템 간의 데이터 전송이다. 메모리 연산들은 메모리에 저장된 오퍼랜드를 이용하는 명령의 결과가 되거나, 또는 데이터 전송은 야기시키지만 다른 어떠한 연산도 야기시키지 않는 적재/저장 명령의 결과가 될 수 있다. 또한, 적재/저장 유닛(26)은, 예를 들어 세그먼트 레지스터들 및 x86 마이크로프로세서 아키텍쳐에 의해 정의되는 어드레스 변환 메커니즘에 관련된 다른 레지스터들과 같은 특별 레지스터들을 위한 특별한 레지스터 저장 장소를 포함할 수 있다.

<48> 일 실시예에서, 적재/저장 유닛(26)은 적재 메모리 연산들을 추론적으로 수행하도록 구성된다. 저장 메모리 연산들은 프로그램 순서로 수행될 수 있지만, 예측된 방향(predicted way)으로 추론적으로 저장될 수 있다. 예측된 방향이 부정확하다면, 저장 메모리 연산 이전의 데이터는 이후 예측된 방향으로 복구(restore)되며, 정확한 방향으로 저장 메모리 연산이 수행된다. 다른 실시예에서, 저장들은 또한 추론적으로 실행될 수 있다. 추론적으로 실행된 저장들은 갱신 이전의 캐시 라인의 카피와 함께 저장 버퍼 내에 놓여질 수 있다. 추론적으로 실행된

저장이 이후 분기 오예측 또는 예외로 인해 버러진다면, 캐시 라인은 버퍼에 저장된 값으로 복구된다. 주목할 사항으로서, 적재/저장 유닛(26)은 어떠한 양의 추론적인 실행(추론적인 실행이 전혀 없음을 포함한다)을 수행하도록 구성될 수 있다.

<49> 데이터 캐시(28)는 적재/저장 유닛(26)과 주 메모리 서브 시스템 간에 전송되는 데이터를 일시 저장하도록 제공된 고속의 캐시 메모리이다. 일 실시예에서, 데이터 캐시(28)는 8방향의 세트 어소시에이티브 구조로 최대 16킬로 바이트의 데이터를 저장할 수 있는 용량을 갖는다. 명령 캐시(16)와 유사하게, 데이터 캐시(28)는 방향 예측 메커니즘을 이용할 수 있다. 이해될 사항으로서, 데이터 캐시(28)는 세트 어소시에이티브 및 직접 맵핑 구성들을 포함하는 특정한 많은 메모리 구성들로 구현될 수 있다.

<50> x86 마이크로프로세서 아키텍쳐를 이용하는 마이크로프로세서(10)의 일 특정 실시예에서, 명령 캐시(16) 및 데이터 캐시(28)는 선형으로 어드레스된다. 선형 어드레스는 명령에 의해 지정되는 오프셋, 및 x86 어드레스 변환 메커니즘의 세그먼트 부분에 의해 지정되는 베이스 어드레스로부터 형성된다. 선형 어드레스들은 주 메모리를 액세스하기 위해 물리 어드레스들로 선택적으로 변환될 수 있다. 선형 어드레스에서 물리 어드레스로의 변환은 x86 어드레스 변환 메커니즘의 페이징 부분(paging portion)에 의해 지정된다. 주목할 사항으로서, 선형으로 어드레스된 캐시는 선형 어드레스 태그들을 저장한다. 물리 태그들(미도시)의 세트가 선형 어드레스들을 물리 어드레스들에 맵핑하고 변환 앨리어스들(translation aliases)을 검출하는 데에 이용될 수 있다. 또한, 물리 태그 블럭은 선형 물리 어드레스 변환을 수행할 수 있다.

<51> 프리페치/프리디코드 유닛 및 명령 캐시 구성

<52> 상기 언급한 바와 같이, 예를 들어 인텔의 82491/82492 캐시 SRAM들과 같은, 마이크로프로세서들을 위한 외부 캐시들로서 이용하도록 설계된 많은 메모리 디바이스들은 패리티 정보를 저장하도록 구성될 수 있다. 일 실시예에서, 8개의 모든 데이터 비트들에 대해 하나의 패리티 비트가 저장될 수 있다. 짝수 패리티가 바람직하다고 가정하면, $01101011_2$의 데이터 바이트는 표명된 비트들의 총 수가 짝수가 되도록 표명된 패리티 비트를 가질 것이다. 패리티 비트는 마이크로프로세서에 의해 발생된 다음, 데이터 바이트와 함께 외부 캐시에 저장될 수 있다. 마이크로프로세서가 캐시로부터 데이터를 역으로(back) 판독할 때, 이 마이크로프로세서는 표명된 데이터 및 패리티 비트들을 셀 수 있다. 결과값이 선택된 패리티와 일치하지 않으면, 패리티 에러가 발생하며, 마이크로프로세서는 적절한 조치(예를 들어, 메모리 에러가 발생했음을 운영 제체에 신호한다)를 취할 수 있다. 다른 메모리 디바이스 구성들은 에러 검사 및 정정(ECC)을 위한 부가적인 비트들을 할당할 수 있다.

<53> 서버들과 같은 고기능(high-end) 시스템들이 전형적으로 패리티 및 ECC를 지원하기는 하지만, 많은 저가격으로부터 미드 범위(mid-range) 시스템 설계자들은 비교적 낮은 데이터 에러 발생 가능성때문에 이러한 기능들을 채용하지 않는다. 이러한 시스템들에서, 레벨 2 캐시들 내의 패리티 및 ECC 비트들은 프리디코드정보를 저장하는 데에 이용될 수 있다. 이는 유익하게는 시스템 레벨에서 고액의 하드웨어 변경을 요구하지 않으면서 성능을 개선할 수 있다.

<54> 도 4는 프리디코드 유닛(12) 및 명령 캐시(16)의 일 실시예의 세부 사항들을 나타낸 도면이다. 본 실시예에서, 프리디코드 유닛(12)은 버스 인터페이스 논리(52)를 통해 캐시(50)에 결합된다. 캐시(50)는, 마이크로프로세서(10)와 동일한 실리콘에 존재하거나, 또는 근처에 있는, 예를 들어 마이크로프로세서(10)에 가까이 있는 보조카드(daughtercard) 또는 마더보드 위에 결합된 개별적인 실리콘에 존재하는 낮은 레이턴시의 높은 대역폭 메모리를 포함한다. 캐시(50)는 정적 랜덤 액세스 메모리(SRAM)들, 동기 다이내믹 액세스 메모리(SDRAM)들, 또는 다른 타입의 낮은 레이턴시 메모리들을 포함할 수 있다. 캐시(50)는 마이크로프로세서(10)와 동일한 실리콘에 존재하거나 개별적인 실리콘에 존재할 수 있다. 일부 실시예들에서, 캐시(50)는 "레벨 2" 캐시라 칭할 수 있는데, 그 이유는 이 캐시가 마이크로프로세서(10)의 기능 유닛들에 두 번째로 가까운 캐시이기 때문이다. 즉, 레벨 1 명령 캐시(16) 및 데이터 캐시(28)의 뒤에 있기 때문이다. 캐시(50)는 또한 "외부 캐시"로서 칭해질 수 있는데, 그 이유는 이 캐시가 마이크로프로세서의 외부에 있기 때문이다.

<55> 버스 인터페이스 논리(52)는 멀티플렉서들, 버퍼들, 트랜스시버들, 드라이버들, 또는 다른 어떠한 타입의 버스 인터페이스 논리들, 즉 마이크로프로세서(10)와 캐시(50) 간의 데이터, 어드레스 및 제어 신호들의 전송을 가능하게 하거나 개선할 수 있는 논리들을 포함할 수 있다. 일부 실시예들에서는, 마이크로프로세서(10)와 레벨 2 캐시(50) 간에 어떠한 버스 인터페이스 논리(52)도 필요없다. 예를 들어, 마이크로프로세서(10) 및 레벨 2 캐시(50)가 물리적으로 서로 충분히 가깝고 이들의 출력 트랜지스터들의 드라이브 성능이 충분히 높다면, 마이크로프로세서(10) 및 캐시(50)는 버스 인터페이스 논리(52) 없이 서로 결합될 수 있다.

<56> 프리디코드 논리(12)는 분기 예측 유닛(14)으로부터 프리페치되는 명령 어드레스들을 수신한 다음, 이들을 버스(68), 버스 인터페이스 논리(52) 및 메모리 버스(56)를 통해 캐시(50)에 전달하도록 구성된다. 레벨 2 캐시(50)가 요구되는 어드레스에 대응하는 명령 바이트들을 저장하고 있다면, 소정 수의 명령 바이트들(예를 들어, 32 바이트들의 한 캐시 라인)이 버스(56), 버스 인터페이스 논리(52) 및 버스(68)를 통해 프리디코드 유닛(12)에 전달된다. 요구되는 명령 바이트들이 레벨 2 캐시(50) 내에 저장되어 있지 않으면, 캐시(50)는 주 메모리 서브시스템으로부터 요구되는 명령 바이트들을 검색(retrieve)하도록 구성된다. 일단 요구되는 명령 바이트들이 주 메모리 서브 시스템으로부터 전달되면, 이들은 명령 캐시(16)로 전달된다. 이들은 또한 캐시(50)에 저장될 수 있다.

<57> 프리디코드 유닛(12)이 요구되는 명령 바이트들을 수신하면, 이는 각 명령 바이트에 대한 프리디코드 정보를 발생시킨다. 도면에 도시된 실시예에서, 프리디코드 유닛(12)은 각 명령 바이트에 대해 하나의 시작 비트를 발생시킨다. 프리디코드 유닛(12)이 요구되는 어드레스의 출력에 응답하여 32 바이트들(예를 들어, 하나의 캐시 라인)을 수신한다고 가정하면, 이 프리디코드 유닛(12)은 각 명령 바이트에 대해 하나의 시작 비트, 즉 총 32 개의 시작 비트들을 발생시키도록 구성될 수 있다. 일단 시작 비트들이 발생되면, 이 시작 비트들은 명령 캐시(16)로 전달되어, 자신들의 관련된 명령 바이트들과 함께 저장된다. 명령 캐시(16)는 다수의 캐시 라인 저장 위치들(64) 및 다수의 프리디코드 정보 저장 위치들(62)로 국부적으로 구성되는바, 여기서 하나의 프리디코드 정보 저장 위치는 각 캐시 라인 저장 위치에 대응한다. 명령 바이트들 및 프리디코드 비트들은 또한 명령 정렬 유닛(18)으로 직접 전송될 수 있다.

<58> 일단 명령 캐시(16) 내의 모든 캐시 라인 저장 위치들(62)이 가득차면, 다수의 서로 다른 알고리즘들을 이용하여 어떤 캐시 라인이 캐시 미스에 대해 교체되어야 하는 지를 결정할 수 있다. 예를 들어, 가까운 미래에 필요하게 될 정보를 버리게 되는 가능성을 줄이기 위해, 캐시 액세스들의 순서를 기록하는 최장 시간 미사용(LRU) 교체 방식이 이용될 수 있다. 오버라이트될 캐시 라인이 선택되면, 대응하는 프리디코드 저장 위치(62)에 저장된 관련된 프리디코드 정보가 명령 캐시(16)로부터 프리디코드 유닛(12) 및 버스 인터페이스 논리(52)를 통해 캐시(50)로 출력된다. 프리디코드 유닛(12)은 또한 오버라이트되는 명령 캐시 정보에 대응하는 어드레스를 전달한다.

<59> 캐시(50)가 프리페치 유닛(12) 및 명령 캐시(16)로부터 프리디코드 비트들 및 대응하는 어드레스를 수신할 때, 이 캐시(50)는 대응하는 어드레스에 관련된 패리티 비트 저장 위치들(60)에 프리디코드 비트들을 저장하도록 구성된다. 일부 실시예들에서, 명령 캐시(16)에 오버라이트되는 실제 명령 바이트들은 또한 프리디코드 비트들과 함께 저장을 위해 캐시(50)로 출력될 수 있다. 이러한 구성은 자기 변경 코드를 지원하는 마이크로프로세서(10)의 일부 실시예들에서 유익하다. 예를 들어, 저장 어드레스들은 저장 명령이 명령 캐시(16) 내에 저장된 명령 바이트들을 오버라이트할 것인 지를 결정하기 위해 명령 캐시(16)에 의해 "스누핑(snooping)"될 수 있다. 저장 명령이 명령 캐시(16) 내에 저장된 명령 바이트들을 오버라이트할 것이라면, 이 저장 명령은 명령 캐시(16) 및 데이터 캐시(28)에 대해 수행되어 바람직한 명령 바이트들을 변경할 수 있다. 이러한 구성에서, 프리디코드 유닛(12)은 변경된 명령에 대한 프리디코드 정보를 재계산한 다음, 재계산된 프리디코드 비트들을 명령 캐시(16) 내에 저장하도록 구성될 수 있다. 캐시(50)가 라이트 백 캐시(write-back cache)로서 구성된다면, 변경된 명령 바이트들은 이후, 변경된 명령 바이트들을 저장하는 캐시 라인 저장 위치가 명령 캐시(16)에 오버라이트될 때, 캐시(50)에 기록될 수 있다.

<60> 프리디코드 정보 (및 이와같이 구성되는 경우에는, 명령 바이트들)이 캐시(50)에 기록된 후, 명령 캐시(16) 내의 캐시 라인 저장 위치 및 프리디코드 정보 저장 위치는 새로운 명령 바이트들 및 새로운 프리디코드 정보로 안전하게 오버라이트될 수 있다. 프리디코드 유닛(12)이 캐시(50)로 출력된 명령 바이트들에 대한 프리페치 또는 페치 요구를 수신하면, 이 프리디코드 유닛(12)은 요구된 어드레스를 캐시(50)로 출력한다. 캐시(50)가 대응하는 명령 바이트들 및 프리디코드 정보를 여전히 저장하고 있으면(즉, 이들이 교체되지 않았으면), 이 캐시(50)는 이들을 프리디코드 유닛(12)에 전달하도록 구성될 수 있다. 프리디코드 유닛(12)은 명령 바이트들 및 대응하는 프리디코드 정보를 명령 정렬 유닛(18) 및 명령 캐시(16)로 전송할 수 있다. 본 실시예에서는 유익하게는, 요구되는 명령 바이트들에 대한 새로운 프리디코드 정보를 발생시키는 과정은 무시될 수 있다. 상기 언급한 바와 같이, 이는 분기 오예측 그리고/또는 캐시 미스가 발생하고 기능 유닛들(24A-24C)이 정체될 위험이 있는 마이크로프로세서(10)의 일부 실시예들에서 유익하다.

<61> 반면, 캐시(50)가 바람직한 명령 바이트들을 다른 어드레스에 위치하는 다른 명령 바이트들로 교체한다면, 이 캐시(50)는 주 메모리로부터 요구되는 바이트들을 검색한 다음, 이들을 프리디코드 유닛(12)으로 전달하도록 구성될 수 있다. 캐시(50)는, 자신이 전송되는 명령 바이트들에 대한 유효한 프리디코드 정보를 갖고 있지 않음을

나타내기 위해 프리디코드 유닛(12)에 명령 바이트들을 전송할 때에, 특정한 제어 신호를 표명하도록 구성될 수 있다.

<62> 다른 실시예들에서, 제어 유닛(80)은 캐시(50)를 모니터하고, 주 메모리로부터의 새로운 명령 바이트들로 교체되는 명령 저장 위치(58)에 대응하는 패리티/ECC 저장 위치들(60) 내에 프리디코드 비트들의 특정한 무효화 시퀀스를 저장한다. 이는 새로운 명령 바이트들에 대한 프리디코드 정보를 효과적으로 "초기화(initialization)"한다. 이러한 구성에서, 캐시(50)는, 대응하는 명령 바이트들이 프리디코드 유닛(12)에 의해 요구될 때, 프리디코드 비트들의 무효화 시퀀스를 출력하도록 구성될 수 있다. 유익하게는, 이러한 구성에서, 캐시(50)는 표준 캐시가 될 수 있는바, 이 표준 캐시는 패리티/ECC를 지원하고, 자신이 출력하는 프리디코드 비트들이 유효한지 무효한지를 알기 위해 어떠한 경우에도 변경될 필요가 없다. 대신, 프리디코드 유닛(12)은 캐시(50)로부터 수신하는 프리디코드 비트들을 검사하도록 구성될 수 있다. 프리디코드 비트들이 소정의 무효화 시퀀스와 일치하면, 프리페치 유닛(12)은 새로운 프리디코드 정보를 계산하도록 구성될 수 있다. 프리디코드 비트들이 소정의 무효화 시퀀스와 일치하지 않으면, 프리페치 유닛(12)은 레벨 2 캐시(50)로부터의 프리디코드 비트들을 명령 캐시(16) (및 필요한 경우, 명령 정렬 유닛(18)/디코드 유닛들(20A-20C))로 전송하도록 구성될 수 있다.

<63> 주목할 사항으로서, 버스들(56 및 68)은 어떠한 알맞은 크기의 폭, 예를 들어 16, 32, 64 또는 128 비트가 될 수 있다. 이용되는 버스 라인들의 수는 데이터, 어드레스 및 제어 버스들을 다중화함으로써 감소될 수 있다. 일부 실시예들에서, 프리디코드 유닛(12) 및 레벨 2 캐시(50)는 전달 및 수신되는 어드레스들에 대한 패리티 비트들을 발생시키도록 구성될 수 있다. 그러나, 이러한 패리티 비트들은 저장될 필요가 없으며, 각 어드레스가 전송된 후 버려질 수 있다.

<64> 또 다른 실시예에서는, 시작 비트들 외에 프리디코드 정보가 프리디코드 유닛(12)에 의해 발생되어 명령 캐시(16)에 저장될 수 있다. 예를 들어, 관련된 명령 바이트가 그 명령의 최종 바이트인지를 식별하는 끝 비트들, 및 관련된 명령 바이트가 연산 코드 바이트인지를 나타내는 연산 코드 바이트들이 또한 프리디코드 유닛(12)에 의해 발생되어 명령 캐시(16)에 저장될 수 있다. 이러한 프리디코드 비트들의 일부 또는 전부는, 캐시(50) 내에서 이용가능한 패리티 또는 ECC 비트들(60)의 수에 따라, 명령 바이트들이 캐시(16) 내에서 교체될 때 캐시(50)로 전달될 수 있다.

<65> 일부 실시예들에서, 디코드 유닛들(20A-20C)은 프리디코드 유닛(12)에 의해 제공되고 명령 캐시(16)로부터 판독된 프리디코드 정보가 언제 부정확한 지를 검출하도록 구성될 수 있다. 프리디코드 정보의 부정확성은 이 프리디코드 정보를 생성하는 데에 이용되는 방법에 따라 서로 다른 이유들로 인해 발생한다. 부정확한 프리디코드 정보의 경우, 디코드 유닛들(20A-20C)은 프리디코드 유닛(12)이 프리디코드 정보를 재생하는 동안 정체되도록 구성될 수 있다. 이후, 새로운 프리디코드 정보가 명령 캐시(16) 내에, 즉 부정확한 프리디코드 정보 위에 기록될 수 있다. 다른 실시예에서, 디코드 유닛들(20A-20C)은 부정확한 프리디코드 정보를 단순히 버리고 디코딩을 완료한다.

<66> 도 5는 명령 캐시(16) 및 캐시(50)의 일 실시예를 보다 상세히 도시한다. 상기 설명한 바와 같이, 명령 캐시(16)는 명령 저장 위치들(62) 및 프리디코드 정보 저장 위치들(64)을 갖도록 구성될 수 있다. 각 프리디코드 정보 저장 위치는 1개의 명령 저장 위치에 대응한다. 예를 들어, 저장 위치(80)에 저장된 프리디코드 정보는 명령 저장 위치(82)에 저장된 명령 바이트들에 대응한다. 도면에 도시된 실시예에서, 저장 위치(80)는 위치(82)에 저장된 각 명령 바이트에 대응하는 시작 비트들을 저장한다.

<67> 도면에 도시된 바와 같이, 캐시(50)는 명령 캐시(16)와 동일한 바이트들을 저장할 수 있다. 예를 들어, 코드 세그먼트(72)로부터의 명령들이 처음으로 요구될 때, 이들은 주 메모리로부터 레벨 2 캐시(50) 및 명령 캐시(16) 내로 판독될 수 있다. 그러나, 주목할 사항으로서, 레벨 2 캐시(50)는 코드 세그먼트(72)에 대한 프리디코드 데이터를 갖지 않을 것이다. 따라서, 소정의 무효화 상수(이 경우에는, 00000000...)이 위치(90)에 저장된다. 프리디코드 유닛(12)이 명령 바이트들을 수신할 때, 이는 무효화 상수를 검출한 다음, 코드 세그먼트(72)에 대한 프리디코드 정보를 계산하기 시작한다. 이러한 프리디코드 정보는 이후 명령 캐시(16) 내의 저장 위치(80)에 저장된다.

<68> 코드 세그먼트(78)의 명령들이 처음으로 요구될 때에, 동일한 과정이 이루어질 것이다. 그러나, 일단 코드 세그먼트(78)에 대한 명령 바이트들이 명령 캐시(16) 내에서 교체되면, 프리디코드 정보가 레벨 2 캐시(50)에 역으로 기록되고 위치(94)에 저장된다. 코드 세그먼트(78)가, 명령 캐시(16) 내에서 교체된 후에 그리고 레벨 2 캐시(50) 내에서 교체되기 전에, 다시 요구된다면, 코드 세그먼트(78)에 대한 명령 바이트들 및 프리디코드 정보가 위치들(94-96)로부터 판독된 다음 명령 캐시(16)에 저장될 수 있다. 도면은 이러한 상태를 도시한다.

<69> 일부 실시예들에서는, 어떠한 소정의 비트들의 시퀀스가 무효화 상수로서 이용될 수 있기는 하지만, 프리디코드 유닛(12)에 의해 발생될 것 같지 않은 프리디코드 비트 시퀀스를 선택하는 것이 유익하다. 예를 들어, 16 바이트의 최대 명령 길이 및 32 바이트의 캐시 라인 길이를 가정하면, 프리디코드 유닛(12)은 적어도 하나의 시작 바이트가 없는 32개의 연속적인 명령 바이트들을 갖는 캐시 라인을 수신할 것이다. 따라서, 32개의 제로들을 포함하는 무효화 상수를 선택하는 것이 유익한데, 그 이유는 이것이 잘못된 무효화들의 수를 줄일 수 있기 때문이다. 잘못된 무효화는, 프리디코드 유닛(12)에 의해 계산된 프리디코드 비트들이 소정의 무효화 상수와 같을 때에 일어날 수 있다. 이러한 잘못된 무효화가 일어나면, 캐시(50)로부터 판독된 최초 프리디코드 비트들이 정확했다고 할지라도, 이 프리디코드 비트들은 버려지고 새로운 비트들이 계산될 것이다.

<70> 일 실시예에서, 상기 설명된 특징들은 마이크로프로세서(10) 내에서 단독으로 구현될 수 있다. 예를 들어, 제어 유닛(80)의 기능은 생략되거나 마이크로프로세서(10) 내에 포함될 수 있다. 유사하게, 일부 실시예들에서는, 버스 인터페이스 논리(52) 또한 생략되거나 마이크로프로세서(10) 내에 포함될 수 있다. 상기 설명된 기능들을 마이크로프로세서(10) 내에서 전적으로 구현하게 되면 유익하게는, 개시된 프리디코드 정보를 저장하는 컴퓨터 시스템의 구현 비용을 줄일 수 있다. 다른 실시예들에서는, 바람직한 기능들이 마이크로프로세서(10)와 1개 이상의 인터페이스 또는 지원 칩들 간에 분산될 수 있다.

<71> 도 6은 레벨 2 캐시(50)에 프리디코드 정보를 저장하는 방법의 일 실시예를 도시한 흐름도이다. 먼저, 명령 캐시(16)로부터 명령 바이트들이 요구된다(단계 140). 요구되는 바이트들이 명령 캐시(16) 내에 저장되어 있으면, 이들은 대응하는 프리디코드 정보와 함께 정렬 유닛(16) 및 디코드 유닛들(20A-20C)에 전달된다(단계 142 및 152). 반면, 요구되는 명령 바이트들이 명령 캐시(16) 내에 저장되어 있지 않으면, 요구된 어드레스가 레벨 2 캐시(50)로 전송된다(단계 142 및 144). 요구되는 명령 바이트들이 레벨 2 캐시(50)에 저장되어 있지 있으면, 이들은 주 메모리 서브 시스템들로부터 판독된다(단계 146 및 148). 요구되는 명령 바이트들이 주 메모리 서브 시스템으로부터 수신되면, 이들은 무효화 상수와 함께 레벨 2 캐시(50)에 저장된다(단계 148 및 150). 상기 언급한 바와 같이, 무효화 상수는 레벨 2 캐시(50) 내의 프리디코드 저장 위치를 효과적으로 초기화하여, 프리디코드 유닛(12)에 새로운 프리디코드 정보가 발생되어야 함을 신호한다. 프리디코드 유닛(12)이 명령 바이트들 및 프리디코드 비트들을 수신하면, 이 프리디코드 비트들을 검사하여, 이들이 무효화 상수와 같은 지를 결정한다(단계 154 및 156). 프리디코드 비트들이 무효화 상수와 같으면, 프리디코드 유닛(12)은 새로운 프리디코드 정보를 발생시키도록 구성된다(단계 156 및 164). 명령 바이트들 및 새로운 프리디코드 정보는 이후 명령 캐시(16)에 저장된다(단계 158). 명령 바이트들 및 새로운 프리디코드 정보의 저장이 명령 캐시(16) 내에 이전에 저장된 캐시 라인의 교체를 필요로 한다면, 오버라이트되는 이전에 저장된 캐시 라인에 대응하는 프리디코드 정보는 레벨 2 캐시(50)에 역으로 저장된다(단계 160 및 166).

<72> 주목할 사항으로서, 도면의 단계들은 단지 설명을 위해 일련의 방법들로 묘사되어 있다. 상기의 많은 단계들은 결합되거나 또는 동시에 수행될 수 있다. 예를 들어, 단계들(150 및 154)은 동시에 수행될 수 있다. 또한, 상기 언급한 바와 같이, 무효화 상수의 이용을 포함하는 단계들은 선택적이다. 즉, 생략될 수 있다.

<73> 패리티 그리고/또는 ECC 정보와 희생 명령 및 데이터 바이트들을 저장

<74> 상기 설명된 실시예들의 한 단점은, 프리디코드 정보를 저장하는 데에 이용되는 패리티 그리고/또는 ECC(에러 검사 및 정정) 비트들의 결과로서, 레벨 2 캐시에 저장된 명령 및 데이터 바이트들이 단일 및 다중 비트 에러들로부터 보호되지 못한다는 점이다. 그러나, 일부 실시예들에서, 상기 설명된 방법 및 시스템들은 레벨 2 캐시에 저장된 데이터 및 명령 바이트들에 단일 그리고/또는 다중 비트 에러들에 대한 적어도 얼마간의 보호를 제공하도록 변경될 수 있다.

<75> 도 7A는 저장된 명령 및 데이터 바이트들을 단일 및 다중 비트 에러들로부터 보호하면서, 레벨 2 캐시에 프리디코드 정보를 저장하는 시스템의 이러한 실시예를 도시한다. 캐시들은 종종 로우들 그리고/또는 칼럼들(이들은 종종 캐시 라인들로 칭해진다)로 구성된다. 도면에 도시된 바와 같이, 명령 캐시(16)의 각 캐시 라인은 저장된 명령 바이트들(184A)에 대응하는 패리티 비트(180A) 및 프리디코드 정보(182A)를 저장할 수 있다. 이전의 실시예들과 관련하여 설명된 바와 같이, 프리디코드 비트들(182A)은 명령 바이트들(184A)에 대한 정보(예를 들어, 시작 바이트, 끝 바이트 및 연산 코드 바이트 정보)를 제공한다. 이전에 언급한 바와 같이, 프리디코드 비트들(182A)은 명령 바이트들(184A)을 디코드하는 데에 필요한 시간을 줄이는 데에 이용될 수 있다. 예를 들어, 명령 바이트들(184A) 내의 시작 및 끝 바이트들을 나타내는 프리디코드 비트들(182A)은 명령들을 명령 바이트들(184A) 내에서 신속하게 정렬하는 데에 이용될 수 있다. 프리디코드 비트들(182A) 및 명령 바이트들(184A)의 합에 기초하여 짝수 패리티 또는 홀수 패리티를 발생시키도록 패리티 비트(180A)가 계산될 수 있다. 다른 실시예

에서, 패리티 비트(180A)는 명령 바이트들(184A)에만 기초하여(즉, 명령 바이트들(184A) 내의 비트 에러들의 홀수들 만을 검출할 수 있는 성능을 가정하여) 계산될 수 있다. 본 실시예에서는, 명령 바이트들(184A)에 보다 많은 보호가 제공되고, 프리디코드 비트들(182A)은 보호되지 않을 수도 있다. 이는, 프리디코드 비트들(182A)이 단지 예측적인 실시예들에 적용될 수 있다. 예를 들어, 프리디코드 비트들(182A)은 이후 무효한 것으로 결정될 수 있다.

<76> 데이터 캐시(28)는 또한 다수의 캐시 라인들을 갖도록 구성될 수 있다. 각 캐시 라인은 자기 자신의 데이터 바이트들(188A) 및 대응하는 ECC 비트들(186A)의 세트를 가질 수 있다. 상기 설명한 바와 같이, ECC 비트들(186A)은 데이터 바이트들(188A) 내의 단일 및 약간의 다중 비트 에러들의 검출을 가능하게 하기 위해 선택될 수 있다. 일부 실시예들에서, ECC 비트들(186A)은 또한 단일 비트 에러들이 검출되어 정정될 수 있게 하는 알고리즘에 따라 선택될 수 있다.

<77> 본원에서 이용되는 ECC 정보는 에러 정정 코드 정보를 말한다. 디지털 전자 시스템들에서, 정보는 이진 포맷(즉, 1 및 0)으로 표현된다. 이진 정보가 한 지점에서 다른 지점으로 전달되거나, 메모리에 저장될 때에는, 항상 에러가 발생할 수 있는 가능성이 있다. 예를 들어, 1이 0으로 해석되거나, 0이 1로 해석될 수 있다. 이는 매체 결함, 전기적 잡음, 구성 요소 고장, 불량한 연결, 노화로 인한 퇴보, 방사 및 다른 요인들에 의해 야기될 수 있다. 비트가 잘못 해석되는 경우, "비트 에러"가 발생했다고 한다. 에러 정정은 에러들을 검출한 다음, 이러한 비트 에러들을 정정하는 과정이다. 에러 정정은 하드웨어 또는 소프트웨어 모두에서 수행될 수 있지만, 전형적으로 소프트웨어 해결책은 너무 느리기 때문에, 높은 데이터 속도에 대한 에러 정정은 바람직하게는 특정한 목적의 하드웨어에서 수행된다. 메모리 밀도 및 동작 주파수가 증가함에 따라, 단일 또는 다중 비트 에러들을 가질 가능성 또한 증가한다. 따라서, 에러 정정 코드들을 지원하는 것이 더욱 중요하게 되었다. 예를 들어, 일부 휴대용 시스템들에서, 에러 정정 코드들은 메모리 리프레시 속도를 상당히 감소시킴으로써 시스템의 배터리 수명을 늘리는 것에 의존한다. 유익하게는, 본원에 개시된 시스템들 및 방법들은 서로 다른 많은 ECC 알고리즘들에 의해 구현될 수 있다. 특정한 알고리즘의 선택은 메모리 구조 및 ECC 정보를 저장하는 데에 이용되는 비트들의 수에 의존할 수 있다. 또한, 주목할 사항으로서, 일부 실시예들에서, ECC 비트들(186A) 및 데이터 바이트들(188A)은 ECC 알고리즘의 일부로서 결합될 수 있다.

<78> 본원에서, 패리티 검사는 메모리 시스템의 단일 비트 에러들을 검사하는 방법으로서 불충분하다. 패리티 지원 메모리는 전형적으로 모든 바이트에 대해 1개의 여분의 메모리 저장 비트를 제공하지만, 다른 구성들이 또한 가능하다. 이러한 여분의 비트는 에러 검출을 위해 정보를 저장하는 데에 이용된다. 이러한 시스템 메모리에 저장된 모든 데이터 바이트는 8개의 리얼 데이터 비트들을 포함하는바, 각 비트는 1 또는 0의 값이다. 따라서, 바이트 내의 0과 1의 총 수를 셀 수 있다. 예를 들어, 10110011 바이트는 3개의 0 및 5개의 1을 갖는다. 00100100 바이트는 6개의 0과 2개의 1을 갖는다. 따라서, 일부 바이트들은 짝수 개의 1을 갖고, 일부 바이트들은 홀수 개의 1을 갖는다. 패리티 검사가 가능하면, 바이트가 메모리 내에 기록될 때 마다, 패리티 발생기/검사기라 칭하는 논리 회로가 바이트를 검사하여, 이 데이터 바이트가 짝수 개의 1을 갖는지 홀수 개의 1을 갖는 지를 결정한다. 데이터 바이트가 짝수 개의 1을 가지면, 9번째(또는 패리티) 비트가 1로 설정되고, 그렇지 않은 경우에는 0으로 설정된다. 결과는, 최초 8개의 데이터 비트들에 얼마나 많이 1이 있었다고 할지라도, 9개의 모든 비트들이 함께 검사될 때에는 항상 홀수 개의 1이 존재한다는 것이다. 이러한 구현은 "홀수 패리티"라 칭한다. 또한, 패리티 발생기/검사기가 합이 항상 짝수가 되게 하는 짝수 패리티를 가질 수 있다.

<79> 데이터가 메모리로부터 역으로 판독될 때, 패리티 회로는 이때 검사기의 역할을 한다. 이는 9개의 모든 비트들을 역으로 판독하여, 1이 짝수개 있는지 홀수개 있는 지를 결정한다. 1이 짝수 개가 있다면, 홀수 개의 비트들에는 반드시 에러가 있다. 이것이 패리티 메모리가 단일 비트 에러들을 검출하는 데에 이용되는 방법이다. 상기의 ECC의 예와 달리, 패리티 발생기/검사기 회로는 1 비트가 부정확하다는 것을 검출할 수는 있지만, 어떤 비트가 부정확한지는 결정할 수 없다. 유익하게는, 패리티 발생 및 검사는 메모리 연산의 판독 및 기록과 동시에 수행될 수 있다. 따라서, 패리티 발생 및 검사는 메모리 시스템의 동작 속도의 저하를 초래하지 않는다.

<80> 일부 실시예들에서는, 에러 검사 및 정정 또한 동시에 수행될 수 있다. 다른 실시예들에서는, 이것이 일련의 동작으로 수행되어 시스템의 동작 속도를 약간 떨어뜨리는데, 그 이유는 에러 검사 및 정정 회로가 검출된 어떠한 에러들을 정정하는 데에 걸리는 시간 때문이다. 그러나, 이러한 지연은 메모리 아키텍처의 구현에 의존하며, 비교적 낮다(예를 들어, 2 내지 3%).

<81> 도 7A를 다시 보면, 일 실시예에서, 명령 및 데이터 캐시(28)는 세트 어소시에이티브 방식으로 구성될 수 있다. 따라서, 각 세트는 한 칼럼에 대응하는 다수의 "방향들(ways)"을 포함할 수 있다. ECC 비트들(186A)은 특정한

캐시 라인 내의 데이터(즉, 단일 칼럼 및 로우 교점 또는 단일 방향)에 대응한다. 명령 캐시(16) 또한 유사한 방식으로 구성될 수 있다. 따라서, 명령 캐시(16)의 각 캐시 라인은 자기 자신의 패리티 및 프리디코드 비트들(180A 및 182A)의 세트를 가질 수 있다. 일부 실시예들에서, 레벨 2 캐시(50)는 프로세서(10)의 다이 위에서 구현될 수 있다. 유익하게는, 이에 의해 레벨 2 캐시(50)는 프로세서(10)와 동일한 주파수에서 동작할 수 있게 된다.

<82> 일 실시예에서, 레벨 2 캐시(50)는 충돌 미스(conflict miss)의 결과로서 메모리 서브 시스템에 역으로 기록될 희생, 즉 카피백 블럭들(copy-back blocks) 만을 포함하는 "배타적인" 캐시로서 구성될 수 있다. 희생 또는 카피백이라는 용어들은, 레벨 1 캐시(즉, 명령 캐시(16) 및 데이터 캐시(28))에 이전에 보유되었지만 보다 새로운 데이터를 위한 공간을 만들기 위해 오버라이트(희생 또는 퇴거)되어야 했던 캐시 블럭들을 말한다. 일 실시예에서, 레벨 2 캐시(50)는 256 킬로 바이트가 될 수 있고, 명령 캐시(16) 및 데이터 캐시(28)는 각각 64 킬로 바이트가 될 수 있는바, 이에 의해 총 384 킬로 바이트의 전용 저장 공간을 갖는 프로세서(10)를 제공한다. 주목할 사항으로서, 이러한 캐시 사이즈들은 예시적인 것들로서, 프로세서(10)의 정확한 구현에 따라 다른 사이즈들도 가능하다. 유사하게, 레벨 2 캐시(50)는 프로세서(10)에 대해 오프칩(off chip)으로 구현될 수 있다. 다른 실시예들에서, 레벨 2 캐시(50)는 명령 캐시(16) 및 데이터 캐시(28) 내의 모든 데이터의 카피를 포함하는 "포괄적인" 캐시로서 구현될 수 있다. 또한, 일부 실시예들에서, 명령 캐시(16) 및 데이터 캐시(28)는 서로 다른 사이즈들을 가질 수 있다. 일부 실시예들에서, 레벨 2 캐시(50)는 또한 세트 어소시에이티브 캐시로서 구현될 수 있다.

<83> 도면에 도시된 바와 같이, 일단 명령 캐시(16)의 하나의 특정한 캐시 라인이 새로운 명령 바이트들을 저장할 필요가 있으면, 이전에 프리디코드된 명령 바이트들(182A 및 184A)은 "희생된다". 레벨 2 캐시(50)는 (a) 프리디코드 정보 및 패리티 정보를 갖는 명령 바이트들, 또는 (b) ECC 정보를 갖는 데이터 바이트들을 저장할 수 있기 때문에, 어떤 타입의 정보가 그 내에 저장되는 지를 나타내는 표시자 비트(예를 들어, 코드 표시자 비트(190))가 저장될 수 있다. Demux(316)는 이 표시자 비트(190)를 이용하여 패리티 검사 유닛(314) 또는 에러 검사 및 정정 유닛(312)으로부터의 출력을 출력(320)으로서 선택할 수 있다. Demux(316)는 또한 패리티 검사 유닛(314) 또는 에러 검사 및 정정 유닛(312)으로부터의 에러 표시의 수신에 응답하여 에러 인터럽트 신호(318)를 표명, 즉 출력하도록 구성될 수 있다. 예를 들어, 패리티 검사 유닛(314)이 패리티 에러가 발생되었다고 결정하는 경우, demux(316)는 인터럽트 신호(318)를 표명하도록 구성될 수 있다. 유사하게, demux(316)는 에러 검사 및 정정 유닛(312)으로부터의 대응하는 에러 신호에 응답하여 에러 신호(318)를 표명하도록 구성될 수 있다. 도면에 도시된 바와 같이, 에러 검사 및 정정 유닛(312)은 ECC 비트들(186B) 및 데이터 바이트들(188B)을 수신하도록 구성될 수 있다.

<84> 패리티 검사 유닛(314)은 패리티 비트(180B), 프리디코드 비트들(182B) 및 명령 바이트들(184B)을 수신하도록 구성될 수 있다. 상기 언급한 바와 같이, 패리티 비트(180B)는 프리디코드 비트들(182B) 및 명령 바이트들(184B)의 표명된 비트들의 수를 결정함으로써 발생될 수 있다. 도면에 도시된 바와 같이, 표시자 비트들(190 및 192)은 명령 코드 또는 데이터 바이트들이 레벨 2 캐시(50)의 대응하는 캐시 라인에 저장되어 있는 지를 나타낼 수 있다. Demux(316)는 표시자 또는 데이터 타입 비트들(170)(예를 들어, 비트들(190 및 192))을 수신하여 패리티 검사 유닛(314) 또는 ECC 유닛(312)으로부터의 출력을 선택하도록 구성될 수 있다.

<85> 도면에 도시된 바와 같이, 명령 캐시(16)의 각 캐시 라인에는 하나의 패리티 비트(180A)가 저장될 수 있다. 예를 들어, 패리티 비트(180A)는 캐시 라인(300)에 대한 패리티를 나타내는 데에 이용될 수 있다. 유사하게, ECC 비트들(186A)은 데이터 캐시(28)의 캐시 라인(302) 내의 데이터 바이트들(188A)에 에러 검사 및 정정을 제공하는 데에 이용될 수 있다. 레벨 2 캐시(50)가 직접 맵핑 캐시로서 도시되기는 했지만, 이 레벨 2 캐시(50)는 다수의 칼럼들 또는 "방향들"을 갖는 세트 어소시에이티브 또는 완전 어소시에이티브 캐시로서 구현될 수 있다. 도면에 도시된 바와 같이, 레벨 1 캐시 라인들(300 및 302)은 명령 캐시(16) 및 데이터 캐시(28) 내에 기록되는 새로운 데이터에 의해 오버라이트(즉, 희생)되는 데이터에 응답하여 레벨 2 캐시(50) 내에 기록될 수 있다.

<86> 도면에 도시되지는 않았지만, 패리티 유닛(314)은 또한 명령 캐시(16)로부터의 판독들(이는 프로세서의 정렬, 디코드 및 실행 유닛들로 전송되는 명령들에 대한 판독들, 및 레벨 2 캐시(50)에 전송된 희생 명령들에 대한 판독들을 포함한다)에 대한 패리티를 검사하는 데에 이용될 수 있다. 다른 실시예에서는, 패리티 유닛(314)의 다른 예가 구현되어 이러한 기능을 제공할 수 있다. 유사하게, 에러 검사 및 정정 유닛(312)(또는 그의 다른 예)은 또한 데이터 캐시(28)로부터의 판독들에 대한 에러 검사 및 정정을 수행하는 데에 이용될 수 있다.

<87> 일 실시예에서, 레벨 2 캐시(50)는 2개의 부분들로 구현될 수 있는바, 각 부분은 서로 다른 타입의 정보를 저장

한다. 예를 들어, 레벨 2 캐시(50)의 제 1 부분은 명령 바이트들, 대응하는 프리디코드 비트들 및 패리티 비트들을 독점적으로 저장하도록 구성될 수 있다. 레벨 2 캐시(50)의 제 2 부분은 데이터 바이트들 및 ECC 비트들을 독점적으로 저장하도록 구성될 수 있다. 유익하게는, 레벨 2 캐시의 이러한 실시예는 데이터 타입 비트들(170)(예를 들어, 코드 비트(190) 및 데이터 비트(192))을 저장해야 하는 필요성을 잠재적으로 줄일 수 있다. 그러나, 이러한 타입의 구성은, 데이터 바이트들에 대한 명령 바이트들의 서로 다른 비(ration)를 갖는 서로 다른 프로그램들이 실행될 때, 레벨 2 캐시(50)의 효율성을 잠재적으로 줄일 수 있다. 주목할 사항으로서, 데이터 타입 비트들(170)은 (명령 캐시(16) 또는 데이터 캐시(28))로부터 정보가 비롯되는 레벨 1 캐시의 부분을 식별함으로써 발생될 수 있다. 예를 들어, 명령 캐시(16)로부터의 정보는 이 정보가 명령 코드 정보임을 나타내는 데이터 타입 비트를 자동으로 수신할 수 있다. 유사하게, 데이터 캐시(28)로부터 비롯되어 레벨 2 캐시(50) 내에 저장되는 정보는, 이 정보가 데이터임을 나타내는 데이터 타입 표시자 비트(예를 들어, 데이터 비트(192))를 자동으로 수신할 수 있다.

<88> 도 7B는 희생 명령 바이트들 및 프리디코드 정보를 저장하는 시스템의 다른 실시예를 도시한다. 이 실시예에서, 패리티 정보는 명령 캐시(16)에 저장되지 않으며, ECC 정보는 데이터 캐시(28)에 저장되지 않는다. 이는 유익하게는 명령 캐시(16) 및 데이터 캐시(28)를 구현하는 데에 필요한 다이의 공간을 줄일 수 있다. 대신, 패리티 유닛(402)은, 명령 바이트들(184A)이 명령 캐시(16)로부터 판독되어 레벨 2 캐시(50)에 기록될 때, 이들에 대한 패리티 비트들을 발생시킬 수 있다. 유사하게, 레벨 2 캐시는, 데이터 바이트들(188A)이 데이터 캐시(28)로부터 판독되어 레벨 2 캐시(50)에 저장될 때, 이들에 대한 ECC 비트들을 발생시키도록 구성될 수 있다. 시스템의 나머지 부분들은 도 7A의 실시예와 동일한 방식으로 동작할 수 있다.

<89> 도 8A는 프리디코드 정보를 저장하는 방법의 일 실시예를 도시한다. 본 실시예에서, 바람직한 명령 바이트들의 세트를 적재하기 위해 페치가 시작된다(단계 240). 페치에 응답하여, 명령 캐시를 탐색하여, 이 명령 캐시가 바람직한 명령 바이트들을 저장하는 지를 결정한다(단계 242). 명령 캐시가 바람직한 명령 바이트들을 저장하고 있으면, 명령 바이트들 및 대응하는 프리디코드 정보가 명령 캐시로부터 판독된다(단계 252). 이는 "명령 캐시 히트(instruction cache hit)"라 칭하며, 전형적으로 바람직한데, 그 이유는 명령 캐시는 전형적으로 메모리 시스템의 나머지 부분 보다 낮은 레이턴시를 갖기 때문이다. 그러나, 바람직한 명령 바이트들이 명령 캐시에 저장되어 있지 않으면, 명령 바이트들은 레벨 2 캐시로부터 요구될 수 있다(단계 244). 상기 언급한 바와 같이, 일부 실시예들에서, 레벨 2 캐시는 프로세서 및 레벨 1 캐시와 동일한 칩, 즉 다이 위에서 구현될 수 있다. 바람직한 정보가 레벨 2 캐시에 저장되어 있지 않으면(단계 246), 명령 바이트들 및 대응하는 패리티 정보가 주 메모리로부터 판독될 수 있다(단계 248). 당업자라면 이해할 수 있는 바와 같이, 주 메모리 서브 시스템의 일부로서 레벨 2 캐시의 범위를 넘는 부가적인 레벨의 캐시들이 있을 수 있다. 일부 실시예들에서, 주 메모리 서브 시스템은 패리티 정보 또는 ECC를 지원하도록 구성될 수 있다. 예를 들어, 주 메모리 서브 시스템은 주 메모리에 기록되고 주 메모리로부터 판독되는 데이터에 대한 패리티 비트들 또는 ECC 비트들을 저장하도록 구성될 수 있다. 그러나, 다른 실시예들에서, 주 메모리 서브 시스템은 패리티 그리고/또는 ECC 정보를 저장 또는 지원하지 않을 수 있다.

<90> 일부 실시예들에서, 레벨 2 캐시는 레벨 1 캐시에 모든 정보의 이중 카피를 저장하도록 구성될 수 있다. 이러한 실시예들에서, 주 메모리로부터 판독된 명령 바이트들은 이 명령 바이트들에 대해 아직 발생되지 않은 프리디코드 비트들에 대한 무효화 상수, 패리티 비트 및 표시자 비트(예를 들어, 데이터 타입 비트)와 함께 레벨 2 캐시에 저장될 수 있다. 프리디코드 비트들이 이미 발생되었거나, 또는 이들이 일단 발생되면, 이들은 무효화 상수를 오버라이트하는 레벨 2 캐시 내에 저장될 수 있다(단계 250). 그러나, 상기 언급한 바와 같이, 다른 실시예들에서, 레벨 2 캐시는 "배타적"이 될 수 있으며, 이에 따라 레벨 1 캐시로부터의 희생 정보 만을 저장한다. 이러한 실시예들에서, 주 메모리 서브 시스템으로부터 판독된 명령 바이트들은 프로세스의 이 시점에서 레벨 2 캐시에 저장되지 않는다. 일단 명령 바이트들이 주 메모리 서브 시스템으로부터 수신되면, 1개 이상의 프리디코드 유닛들이 명령 바이트들에 대한 프리디코드 비트들을 발생시키도록 구성될 수 있다(단계 252).

<91> 주 메모리로부터 판독된 정보에 대한 패리티 검사를 지원하는 실시예들에서, 프리디코드 유닛(또는 패리티 유닛)은 패리티를 검사하도록 구성되어, 주 메모리로부터 데이터가 정확하게 수신되었음을 보장한다(단계 254). 주목할 사항으로서, 일부 실시예들에서, 패리티 검출은 프리디코드 유닛이 대응하는 프리디코드 비트들을 발생시킴과 동시에, 또는 그 전에 수행될 수 있다. 에러가 있으면, 프리디코드 유닛은 프로세서에 에러 신호 또는 인터럽트를 표명하도록 구성될 수 있다(단계 256). 패리티 검사가 에러를 나타내지 않는다면, 주 메모리 서브 시스템으로부터 판독된 프리디코드 비트들을 검사하여, 이들이 무효화 상수와 같은 지를 결정한다(단계 258).

<92> 프리디코드 비트들이 무효화 상수와 같다면, 프리디코드 유닛은 새로운 프리디코드 정보를 발생시키도록 구성될

수 있다(단계 260). 프리디코드 비트들이 무효화 상수와 같지 않다면, 프리디코드 유닛은 명령 바이트들 및 이전에 발생된 프리디코드 비트들을 명령 캐시에 저장하도록 구성될 수 있다(단계 262). 주목할 사항으로서, 일부 실시예들에서, 명령 캐시는 또한 명령 바이트들에 대한 패리티 비트들을 저장하도록 구성될 수 있다. 다른 실시예들에서, 명령 바이트들에 대한 패리티 비트들은, 명령 바이트들이 명령 캐시로부터 판독되고 레벨 2 캐시로 저장될 때(즉, 희생될 때)에 발생될 수 있다. 따라서, 일부 실시예들에서, 패리티 검사는 레벨 2 캐시에 대해서만 수행될 수 있다. 다른 실시예들에서, 패리티 검사는 명령 캐시(즉, 레벨 1 캐시) 및 레벨 2 캐시 모두에 대해 수행될 수 있다. 일부 구현들에서, 패리티 검사는 또한 주 메모리 서브 시스템에 대해 수행될 수 있다.

<93> 프리디코드 비트들이 무효화 상수와 같다면, 프리디코드 유닛은 명령 바이트들, 프리디코드 비트들 및 패리티 정보를 명령 캐시 내에 저장하도록 구성될 수 있다(단계 262). 데이터가 즉시 필요하다면, 명령 바이트들은 또한 처리를 위해 디코딩 유닛들에 제공될 수 있다. 이후의 어떠한 적절한 시점에서, 명령 바이트들 및 프리디코드 정보를 저장하고 있는 명령 캐시의 대응하는 캐시 라인은 오버라이트, 즉 희생될 수 있다(단계 264). 이 경우, 명령 캐시로부터의 프리디코드 정보 및 명령 바이트들은 대응하는 패리티 비트 및 데이터 타입 비트와 함께 레벨 2 캐시 내에 저장될 수 있다(단계 262).

<94> 도 8B는 희생 데이터 바이트들을 처리하는 방법의 일 실시예를 도시한다. 먼저, 데이터 바이트들이 적재/저장 유닛에 의해 요구 또는 페치된다(단계 370). 이러한 페칭 프로세스의 일부로서, 레벨 1 캐시, 즉 데이터 캐시가 검사되어, 바람직한 데이터 바이트들이 그 내에 저장되어 있는 지를 결정한다(단계 372). 데이터 바이트들이 데이터 캐시에 저장되어 있으면, 이들은 그들의 대응하는 ECC 정보와 함께 데이터 캐시로부터 판독될 수 있다(단계 374). 그러나, 바람직한 데이터 바이트들이 데이터 캐시에 저장되어 있지 않으면, 데이터는 레벨 2 캐시로부터 요구될 수 있다(단계 376). 데이터가 레벨 2 캐시에 저장되어 있지 않으면(단계 378), 바람직한 데이터 바이트들은 주 메모리로부터 판독될 수 있다(단계 380).

<95> 이전에 언급한 바와 같이, 도 8A와 관련하여 설명된 패리티 정보와 유사하게, ECC 정보는 (a) 레벨 2 캐시 내에서만, (b) 레벨 1 캐시 및 레벨 2 캐시 모두에서, 또는 (c) 레벨 1 캐시, 레벨 2 캐시 및 주 메모리 서브 시스템에서 지원될 수 있다. (예를 들어, 주 메모리 서브 시스템 및 레벨 2 캐시 내에서와 같은) 다른 결합들이 또한 가능하다. 주 메모리로부터의 ECC 정보를 지원하는 실시예들에 있어서, 일단 데이터 바이트들이 주 메모리 서브 시스템으로부터 수신되면, 대응하는 ECC 정보가 검사될 수 있다(단계 382). 정보가 부정확하다면, 에러 검사 및 정정 유닛은 에러를 정정하고자 시도할 수 있다. 에러가 정정 불가능하다면, 에러가 신호될 수 있다(단계 384). ECC 정보가 정확하다면(또는 에러가 정정된다면), 데이터 바이트들과 함께 ECC 정보가 레벨 2 캐시에 저장될 수 있다(단계 386). 도 8A와 관련하여 상기에서 언급한 바와 같이, 본 예는 포괄적인 레벨 2 캐시를 가정한다. 일부 실시예들에서, 레벨 2 캐시는 배타적이 될 수 있으며, 주 메모리로부터 직접 판독된 데이터는 레벨 1 캐시에서 희생될 때 까지 레벨 2 캐시에 저장되지 않을 수도 있다.

<96> 이후의 적절한 시점에서, 데이터 바이트들 및 ECC 정보를 저장하는 레벨 1 캐시 라인은 희생, 즉 오버라이트될 수 있다(단계 392). 이 경우, 데이터 캐시로부터의 데이터 바이트들은 대응하는 ECC 비트들 및 데이터 타입 비트와 함께 레벨 2 캐시 내에 저장될 수 있다(단계 366). 데이터 타입 비트는, 저장된 정보가 (패리티 및 프리디코드 정보를 갖는 명령 바이트들과 대조적으로) 데이터 및 ECC 정보임을 나타내는 데에 이용될 수 있다.

<97> 도 9는 희생 데이터 바이트들에 대한 패리티 그리고/또는 ECC 보호를 여전히 제공하면서, 희생 명령 바이트들에 대한 프리디코드 정보를 저장하도록 구성된 컴퓨터 시스템의 일 실시예를 도시한다. 본 실시예에서, 프로세서(10)는 레벨 1 명령 캐시(16) 및 레벨 1 데이터 캐시(28)를 포함한다. 프로세서(10)는 또한 프리디코드/프리페치 유닛(12) 및 적재/저장 유닛(26)을 포함한다. 도면에 도시된 바와 같이, 프리페치/프리디코드 유닛(12) 및 적재/저장 유닛(26)은 각각 패리티 발생 및 검사 유닛(402)과 에러 검사 및 정정 유닛(404)을 포함할 수 있다.

<98> 본 실시예에서, 레벨 2 캐시(50)는 주 레벨 2 캐시(50)에 저장된 대응하는 바이트들에 대한 패리티 그리고/또는 ECC 정보를 저장하도록 구성된 다수의 저장 위치들(406)을 갖도록 구성된다. 본 실시예에서, 컴퓨터 시스템은 또한 메모리 제어기(410)를 포함한다. 이 메모리 제어기(410)는 패리티 유닛(414)을 포함할 수 있는바, 이 유닛은 주 메모리(420)로부터 비롯되는 바이트들에 대한 패리티를 발생시키고 검사하도록 구성된다. 패리티가 지원되는 실시예들에 있어서, 주 메모리 서브 시스템(420)은 패리티 정보를 저장하기 위한 다수의 메모리 저장 위치들(430)을 갖도록 구성될 수 있다.

<99> 다른 실시예들에서, 메모리 제어기(410)는 주 메모리 서브 시스템(420)에 저장된 정보에 대한 에러 정정 코드들을 발생시키고 정정하도록 구성된 에러 검사 및 정정 유닛(416)을 포함할 수 있다. 주목할 사항으로서, 메모리(420), 메모리 제어기(410) 및 프로세서(10) 간에 정보를 전송하는 데에 이용되는 버스들에 대한 부가적인 에러

검사 및 정정 정보가 발생될 수 있다. 이 정보는 본원에서 설명되는 ECC 그리고/또는 패리티 정보에 부가적인 것이다. 상기 설명한 바와 같이, 일부 실시예들에서, 메모리 서브 시스템(420) 및 메모리 제어기(410)는 패리티 그리고/또는 ECC 정보를 지원하지 않도록 구성될 수 있다. 또한, 도면에 도시된 바와 같이, 일부 실시예들에서, 레벨 2 캐시(50)는 각 캐시 라인에 대해 데이터 타입 비트들(408)을 저장하도록 구성될 수 있다.

<100> 레벨 1 명령 캐시(16)에서 패리티 비트들을 이용하여 검출된 단일 비트 에러들은, 캐시 라인을 버리고 (그 자체가 ECC 정보에 의해 보호될 수 있는) 메모리(420)로부터 대응하는 명령 바이트들을 판독함으로써 정정될 수 있다. 반대로, 레벨 1 데이터 캐시(28)에 저장된 데이터는 변경되어, 메모리(420) 내의 데이터의 최초 카피(즉, 스테일 카피(stale copy))가 에러들을 정정하는 데에 이용되는 것을 막을 수 있다. 따라서, 일부 실시예들에서, 데이터 캐시(28)에 ECC 정보를 갖는 것은 메모리(420)로부터의 데이터 바이트들을 카피하는 대신 에러들을 정정하는 대안적인 메커니즘을 제공하는 데에 특히 유용하다.

<101> 예시적인 컴퓨터 시스템

<102> 도 10은 마이크로프로세서(10)를 이용하는 컴퓨터 시스템(500)의 일 실시예의 블록도이다. 도시된 시스템에서, 주 메모리(504)는 메모리 버스(506)를 통해 버스 브리지(502)(이는 메모리 제어기의 역할을 함)에 결합되고, 그래픽 제어기(508)는 AGP 버스(510)를 통해 버스 브리지(502)에 결합된다. 궁극적으로, 다수의 PCI 디바이스들(512A-512B)은 PCI 버스(514)를 통해 버스 브리지(502)에 결합된다. 또한, 제 2 버스 브리지(516)가 제공되어, EISA/ISA 버스(520)를 통한 1개 이상의 EISA 또는 ISA 디바이스들(518)에 대한 전기적인 인터페이스를 제공한다. 마이크로프로세서(10)는 CPU 버스(524)를 통해 버스 브리지(502)에 결합된다.

<103> 버스 브리지(502)는 마이크로프로세서(10), 주 메모리(504), 그래픽 제어기(508), 및 PCI 버스(514)에 결합된 디바이스들 간에 인터페이스를 제공한다. 버스 브리지(502)에 연결된 디바이스들중 하나로부터 동작이 수신되면, 버스 브리지(502)는 동작의 타겟(예를 들어, 특정한 디바이스, 또는 PCI 버스(514)의 경우에는, 그 타겟이 PCI 버스(514) 상에 있는 지를)을 식별한다. 버스 브리지(502)는 상기 동작을 타겟 디바이스로 보낸다. 버스 브리지(502)는 일반적으로 상기 동작의 프로토콜을 소스 디바이스 또는 버스에 의해 이용되는 프로토콜로부터 타겟 디바이스 또는 버스에 의해 이용되는 프로토콜로 변환한다.

<104> 제 2 버스 브리지(516)는 PCI 버스(514)에 대한 인터페이스를 ISA/EISA 버스에 제공할 뿐 아니라, 필요한 경우 부가적인 기능을 통합할 수 있다. 예를 들어, 일 실시예에서, 제 2 버스 브리지(516)는 PCI 버스(514)의 소유권을 조정하기 위한 마스터 PCI 조정기(미도시)를 포함한다. 또한, 제 2 버스 브리지(516)의 외부에 있거나, 또는 제 2 버스 브리지(516)에 통합되는 입/출력 제어기(미도시)가 컴퓨터 시스템(500) 내에 포함되어, 필요한 경우 키보드 및 마우스(522), 그리고 다양한 직렬 및 병렬 포트들에 대한 동작을 지원한다. 또한, 다른 실시예들에서는, 외부 캐시 유닛(미도시)이 마이크로프로세서(10)와 버스 브리지(502) 사이의 CPU 버스(524)에 결합될 수 있다. 대안적으로, 외부 캐시는 버스 브리지(502)에 결합될 수 있으며, 이 외부 캐시에 대한 캐시 제어 논리는 버스 브리지(502)에 통합될 수 있다.

<105> 주 메모리(504)는, 애플리케이션 프로그램들이 저장되고 이로부터 마이크로프로세서(10)가 주로 실행되는 메모리이다. 적절한 주 메모리(504)는 DRAM(동적 랜덤 액세스 메모리) 및 바람직하게는 다수의 SDRAM(동기 DRAM)의 뱅크들을 포함한다.

<106> PCI 디바이스들(512A-512B)은, 예를 들어 네트워크 인터페이스 카드, 비디오 가속 장치(video accelerator), 오디오 카드, 하드 또는 플로피 디스크 드라이브 제어기, SCSI(소형 컴퓨터 시스템 인터페이스) 어댑터들 및 전화 카드와 같은 다양한 주변 디바이스들의 예이다. 유사하게, ISA 디바이스(518)는 다양한 타입의 주변 디바이스들(예를 들어, 모뎀, 사운드 카드) 및 많은 데이터 획득 카드들(예를 들어, GPIB 또는 필드 버스 인터페이스 카드)의 예이다.

<107> 그래픽 제어기(508)는 디스플레이(526) 상에서의 텍스트 및 이미지들의 표현(rendering)을 제어하기 위해 제공된다. 그래픽 제어기(508)는, 주 메모리(504)로/로부터(즉, 주 메모리(504) 사이에서) 효과적으로 시프트될 수 있는 3차원 데이터 구조들을 표현하기 위해 당업계에 일반적으로 알려진 전형적인 그래픽 가속 장치를 구현할 수 있다. 이에 따라, 그래픽 제어기(508)는, 버스 브리지(502) 내의 타겟 인터페이스에 대한 액세스를 요구하고 수신함으로써 주 메모리(504)에 대한 액세스를 얻을 수 있다는 점에서, AGP 버스(510)의 마스터가 될 수 있다. 전용 그래픽 버스는 주 메모리(504)로부터 데이터를 신속하게 검색할 수 있게 한다. 특정한 동작들에 있어서, 그래픽 제어기(508)는 또한 AGP 버스(510) 상에 PCI 프로토콜 트랜잭션들을 발생시키도록 구성될 수 있다. 따라서, 버스 브리지(502)의 AGP 인터페이스는 AGP 프로토콜 트랜잭션들 뿐 아니라 PCI 프로토콜 타겟 및 개시 트랜

잭션들(initiator transactions) 모두를 지원하는 기능을 포함할 수 있다. 디스플레이(526)는 이미지 또는 텍스트가 제공될 수 있는 어떠한 전자 디스플레이이다. 적절한 디스플레이(526)는 음극선관("CRT"), 액정 디스플레이("LCD") 등을 포함한다.

<108> 주목할 사항으로서, AGP, PCI, 및 ISA 또는 EISA 버스들이 상기 설명에서 예로서 이용되었지만, 필요한 경우 어떠한 버스 아키텍쳐들로도 대체될 수 있다. 또한, 주목할 사항으로서, 컴퓨터 시스템(500)은 추가적인 마이크로프로세서들을 포함하는 멀티프로세싱 컴퓨터 시스템이 될 수 있다.

<109> 또한, 주목할 사항으로서, 본 설명은 다양한 신호들의 표명을 설명한다. 본원에서, 어떤 신호가 특정한 조건을 나타내는 값을 전달한다면, 그 신호는 "표명"된다. 반대로, 어떤 신호가 특정한 조건의 결여를 나타내는 값을 전달한다면, 그 신호는 "비표명(deassertion)"되거나 "표명되지 않는다". 신호는, 논리 0 값을 전달하거나, 또는 반대로 논리 1 값을 전달할 때에, 표명되는 것으로 정의된다. 또한, 상기 설명에서는 다양한 값들이 버려지는 것으로 설명되었다. 이러한 값은 다양한 방법들로 버려질 수 있지만, 일반적으로 이 값을 수신하는 논리 회로에 의해 무시될 수 있도록 값을 변경하는 것을 포함한다. 예를 들어, 값이 비트를 포함한다면, 이 값의 논리 상태가 반전되어 이 값을 버릴 수 있다. 값이 n 비트 값이라면, n 비트 코드들중 하나는 이 값이 무효함을 나타낼 수 있다. 이 비트를, 코드를 무효로 하는 값으로 설정하게 되면, 이 값이 버려지게 된다. 또한, n 비트 값은 유효 비트를 포함하는바, 이것이 설정되면, n 비트 값이 유효함을 나타낸다. 유효 비트의 재설정은 값을 버리는 것을 포함한다. 이러한 값을 버리는 다른 방법들이 또한 이용될 수 있다.

<110> 프리디코드 정보 및 희생 명령 그리고/또는 데이터 바이트들을 저장하는 시스템 및 방법에 대해 설명했다. 본원에서는 특정한 실시예들에 대해 상세히 설명했지만, 다른 실시예들이 가능하며 고려된다. 이해될 사항으로서, 본 발명의 도면 및 상세한 설명은 본 발명을 개시된 특정한 형태들로 한정하지 않는다. 본 발명은 첨부된 청구항들에 의해 정의되는 본 발명의 정신 및 범위 내에서 모든 변형들, 등가들 및 대안들을 포함한다.

## *산업상 이용 가능성*

<111> 본 발명은 마이크로프로세서들에 적용할 수 있다.

## *도면의 간단한 설명*

<20> 도 1은 일반적인 x86 명령 포맷의 블록도이다.

<21> 도 2는 유효 마스크의 일 실시예를 도시한 블록도이다.

<22> 도 3은 마이크로프로세서의 일 실시예의 블록도이다.

<23> 도 4는 도 3의 명령 캐시와 레벨 2 캐시 간의 인터페이스의 일 실시예의 세부사항들을 도시한다.

<24> 도 5는 도 4에 도시된 명령 캐시의 일 실시예와 도 4의 레벨 2 캐시의 일 실시예 간의 관계의 세부사항들을 도시한다.

<25> 도 6은 레벨 2 캐시에 프리디코드 정보를 저장하는 방법의 일 실시예를 도시한 흐름도이다.

<26> 도 7A-B는 희생 프리디코드 정보를 저장하는 시스템의 다른 실시예들을 도시한 도면들이다.

<27> 도 8A-B는 희생 프리디코드 정보를 저장하는 방법의 다른 실시예들을 도시한 흐름도들이다.

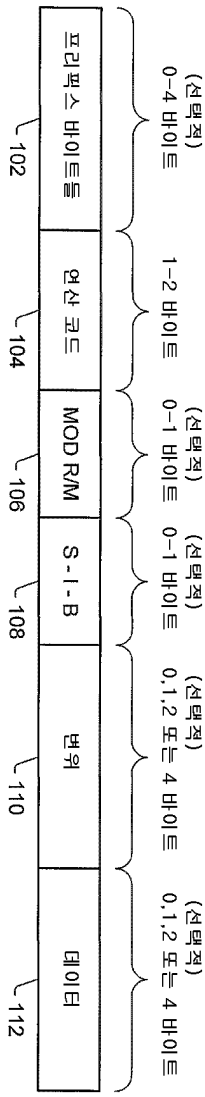
<28> 도 9는 희생 프리디코드 정보를 저장하는 시스템의 다른 실시예를 도시한 도면이다.

<29> 도 10은 도 3의 마이크로프로세서를 이용하는 컴퓨터 시스템의 일 실시예를 도시한 도면이다.

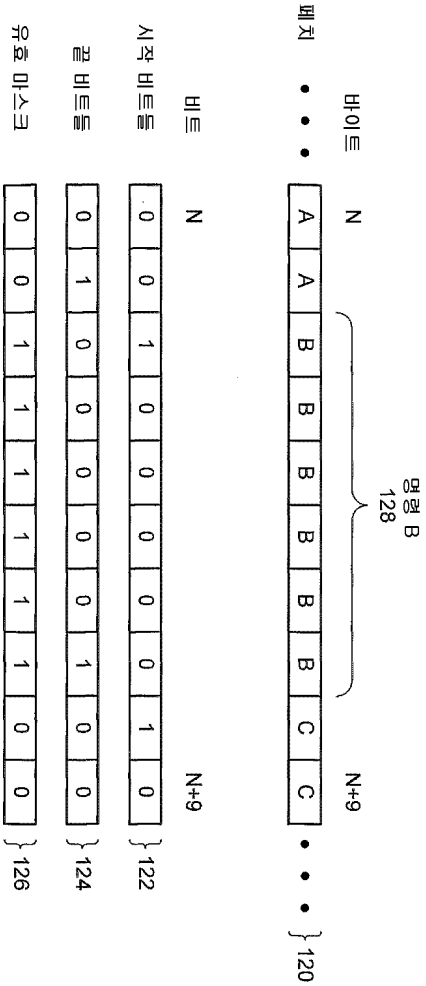
<30> 본 발명은 다양한 변형들 및 대안적인 형태들을 갖지만, 도면은 본 발명의 특정한 실시예들을 예시적으로 도시하고, 상세한 설명은 이들에 대해 상세히 설명한다. 그러나, 이해될 사항으로서, 도면들 및 상세한 설명은 본 발명을 개시된 특정한 형태로 한정하지 않으며, 본 발명은 첨부된 청구항들에 의해 규정되는 본 발명의 정신 및 범위 내에 포함되는 모든 변형들, 등가들 및 대안들을 포함한다. 본원에서 이용되는 표제(heading)들은 단지 구성 목적을 위한 것으로서, 상세한 설명 또는 청구항의 범위를 한정하는 데에 이용되는 것으로 의도되지 않는다. 본원에서 전체적으로 이용되는 표현 "~ 수도"는 강제적인 의미(즉, ~야만 한다는 의미)가 아니라, 허용의 의미(즉, ~수도 있다는 의미)로 쓰인다. 유사하게, 표현 "~를 포함한다", "~를 포함하는"은 단지 ~ 만을 포함하는 의미가 아니라 그 외의 다른 어떤 것도 함께 포함할 수 있음을 의미한다.

## 도면

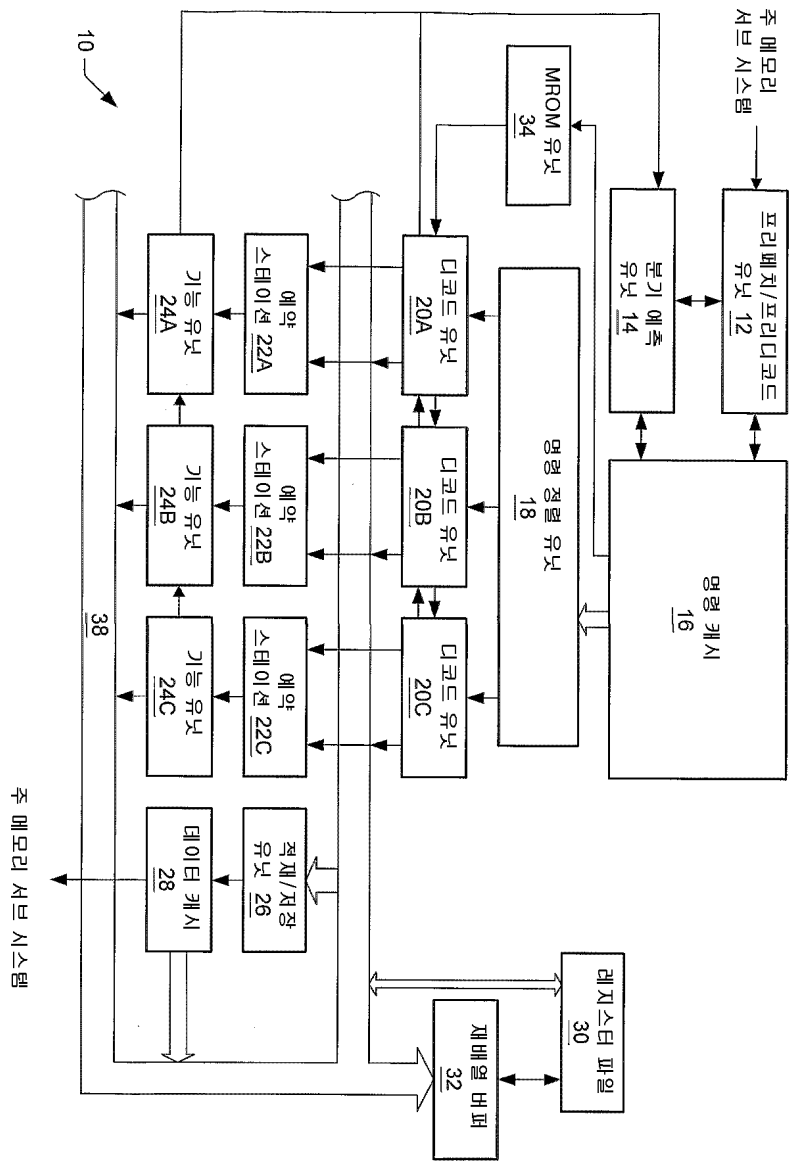
### 도면1


프리픽스 바이트들
연산 코드
MOD R/M
S - I - B
변위
데이터
102
104
106
108
110
112
(선택적) 0-4 바이트
1-2 바이트
(선택적) 0-1 바이트
(선택적) 0-1 바이트
(선택적) 0,1,2 또는 4 바이트
(선택적) 0,1,2 또는 4 바이트

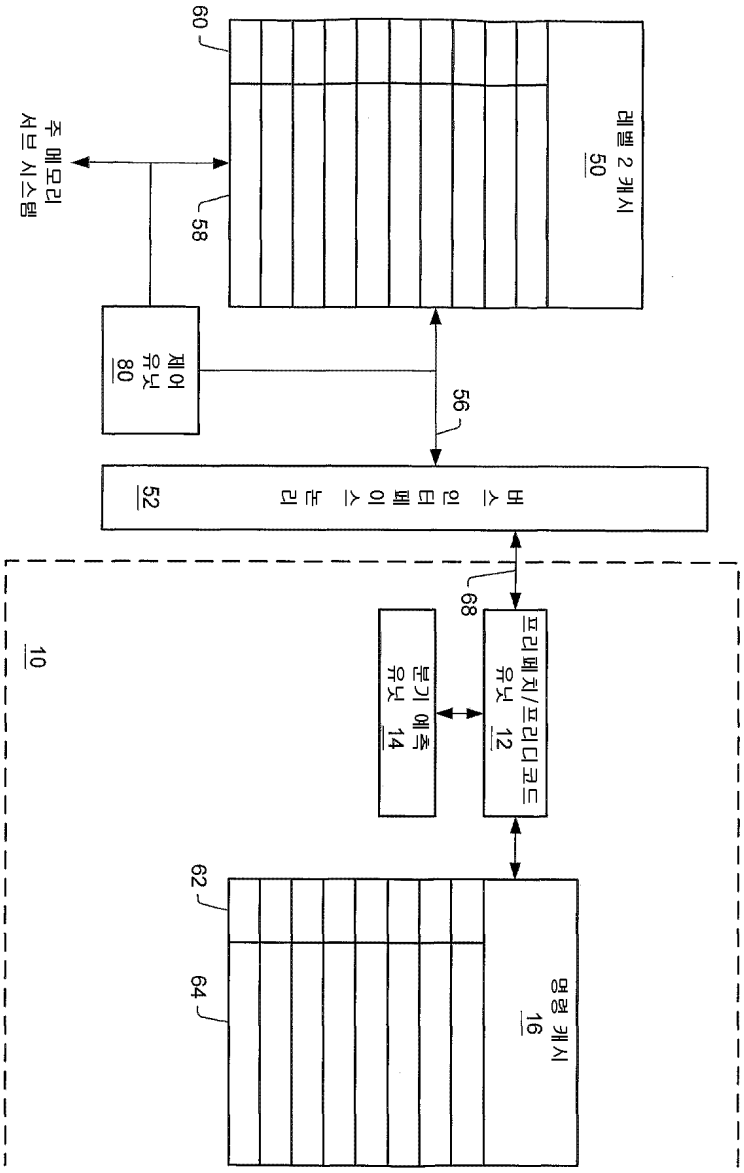
도면2


페지
바이트
N
명령 B
128
A
A
B
B
B
B
B
B
B
C
C
N+9
120
비트
N
N+9
시작 비트들
0
0
1
0
0
0
0
0
0
1
0
122
끝 비트들
0
1
0
0
0
0
0
0
1
0
0
124
유효 마스크
0
0
1
1
1
1
1
1
1
0
0
126

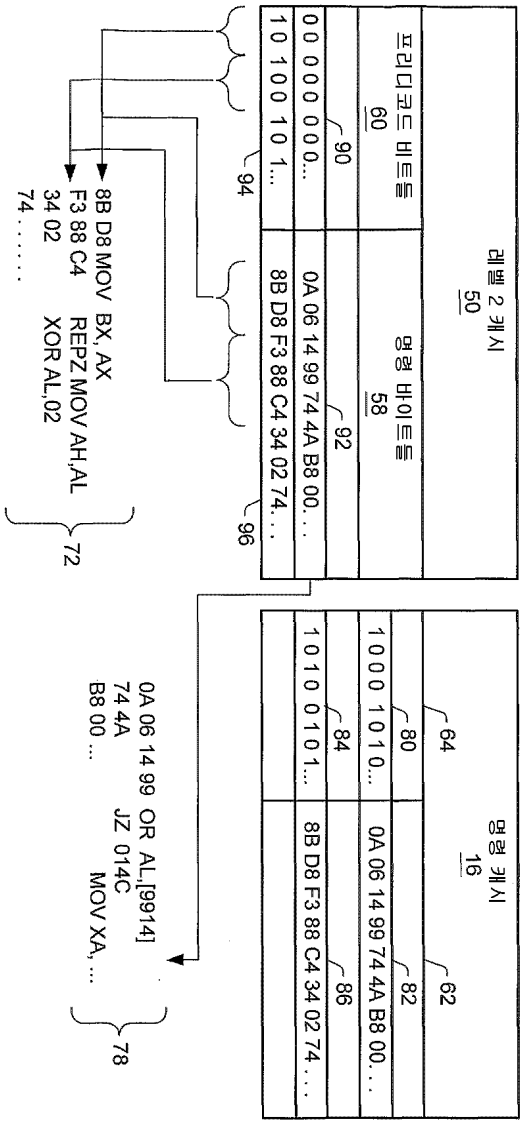
도면3


10
주 메모리 서브 시스템
MROM 유닛 34
프리페치/프리디코드 유닛 12
분기 예측 유닛 14
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
적재/저장 유닛 26
데이터 캐시 28
레지스터 파일 30
재배열 버퍼 32
38
주 메모리 서브 시스템

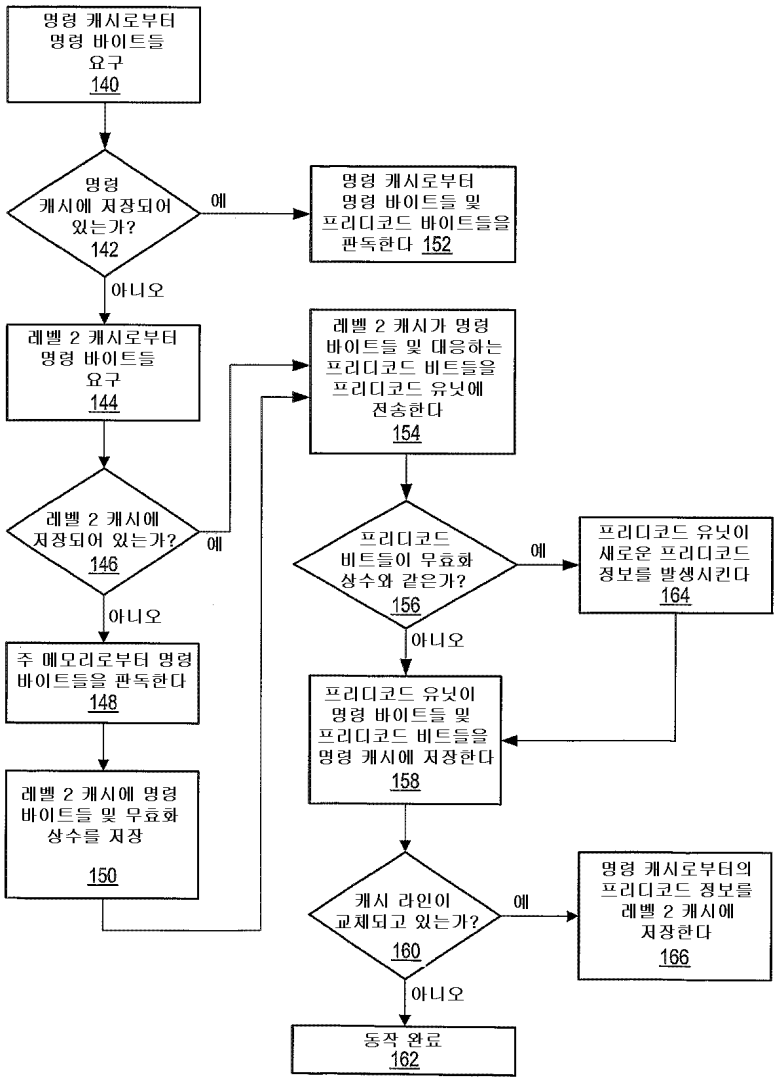
도면4


60
레벨 2 캐시
50
주 메모리
서브 시스템
58
제어
유닛
80
56
버스
인터페이스
논리
52
68
10
분기 예측
유닛 14
프리페치/프리디코드
유닛 12
62
64
명령 캐시
16

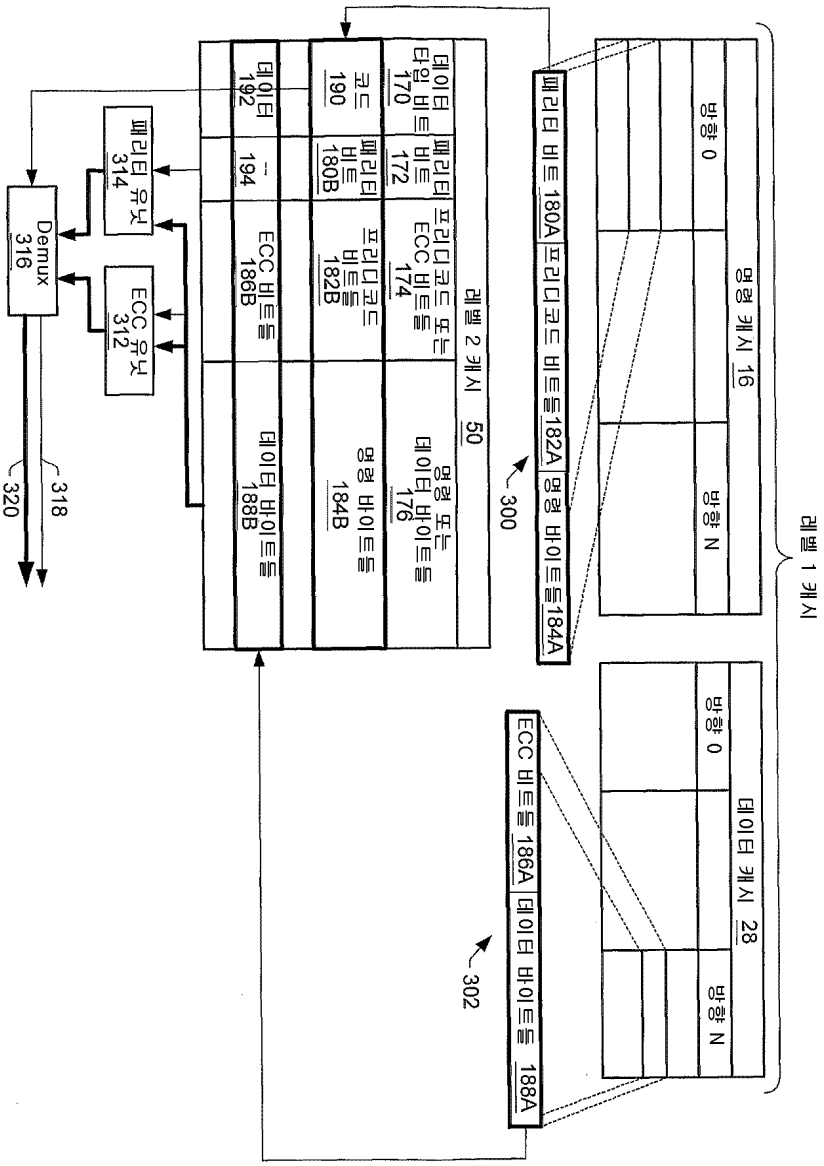
도면5


레벨 2 캐시
50
프리디코드 비트들
60
90
00 000 000...
10 100 10 1...
94
명령 바이트들
58
92
0A 06 14 99 74 4A B8 00...
8B D8 F3 88 C4 34 02 74...
96
8B D8 MOV BX, AX
F3 88 C4 REPZ MOV AH,AL
34 02 XOR AL,02
74 .....
72
명령 캐시
16
64
80
1000 1010...
84
1010 0101...
62
82
0A 06 14 99 74 4A B8 00...
86
8B D8 F3 88 C4 34 02 74...
0A 06 14 99 OR AL,[9914]
74 4A JZ 014C
B8 00 ... MOV XA, ...
78

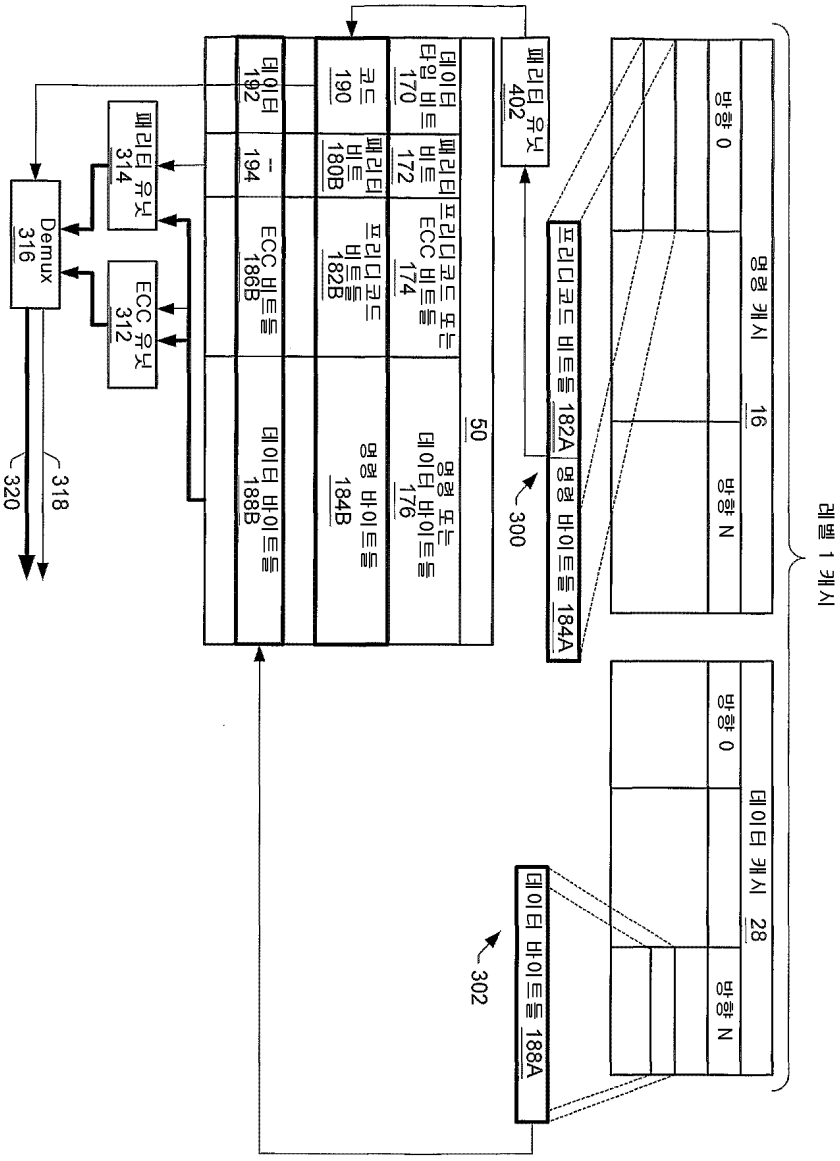
도면6


명령 캐시로부터 명령 바이트들 요구 140
명령 캐시에 저장되어 있는가? 142
예
명령 캐시로부터 명령 바이트들 및 프리디코드 바이트들을 판독한다 152
아니오
레벨 2 캐시로부터 명령 바이트들 요구 144
레벨 2 캐시에 저장되어 있는가? 146
예
레벨 2 캐시가 명령 바이트들 및 대응하는 프리디코드 비트들을 프리디코드 유닛에 전송한다 154
아니오
주 메모리로부터 명령 바이트들을 판독한다 148
레벨 2 캐시에 명령 바이트들 및 무효화 상수를 저장 150
프리디코드 비트들이 무효화 상수와 같은가? 156
예
프리디코드 유닛이 새로운 프리디코드 정보를 발생시킨다 164
아니오
프리디코드 유닛이 명령 바이트들 및 프리디코드 비트들을 명령 캐시에 저장한다 158
캐시 라인이 교체되고 있는가? 160
예
명령 캐시로부터의 프리디코드 정보를 레벨 2 캐시에 저장한다 166
아니오
동작 완료 162

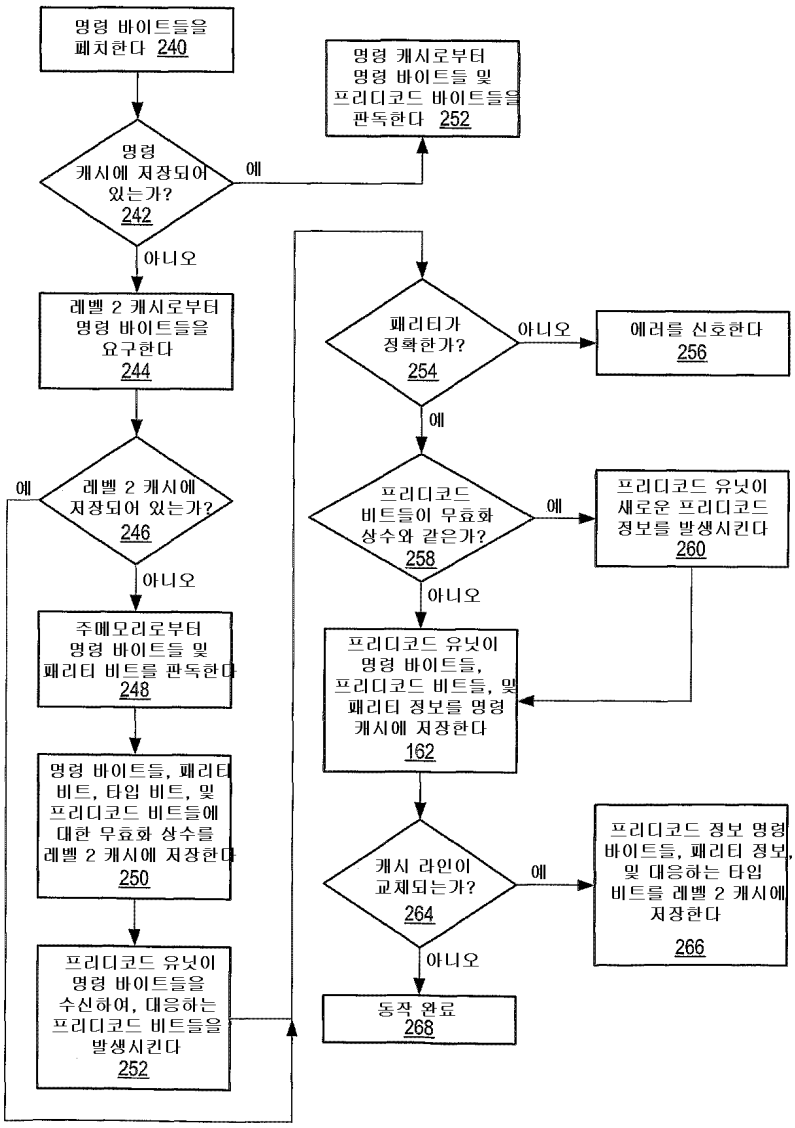
도면7A


레벨 1 캐시
명령 캐시 16
방향 0
방향 N
데이터 캐시 28
방향 0
방향 N
패리티 비트 180A
프리디코드 비트들182A
명령 바이트들184A
ECC 비트들 186A
데이터 바이트들 188A
300
302
레벨 2 캐시 50
데이터 타입 비트 170
패리티 비트 172
프리디코드 또는 ECC 비트들 174
명령 또는 데이터 바이트들 176
코드 190
패리티 비트들 180B
프리디코드 비트들 182B
명령 바이트들 184B
데이터 192
194
ECC 비트들 186B
데이터 바이트들 188B
패리티 유닛 314
ECC 유닛 312
Demux 316
318
320

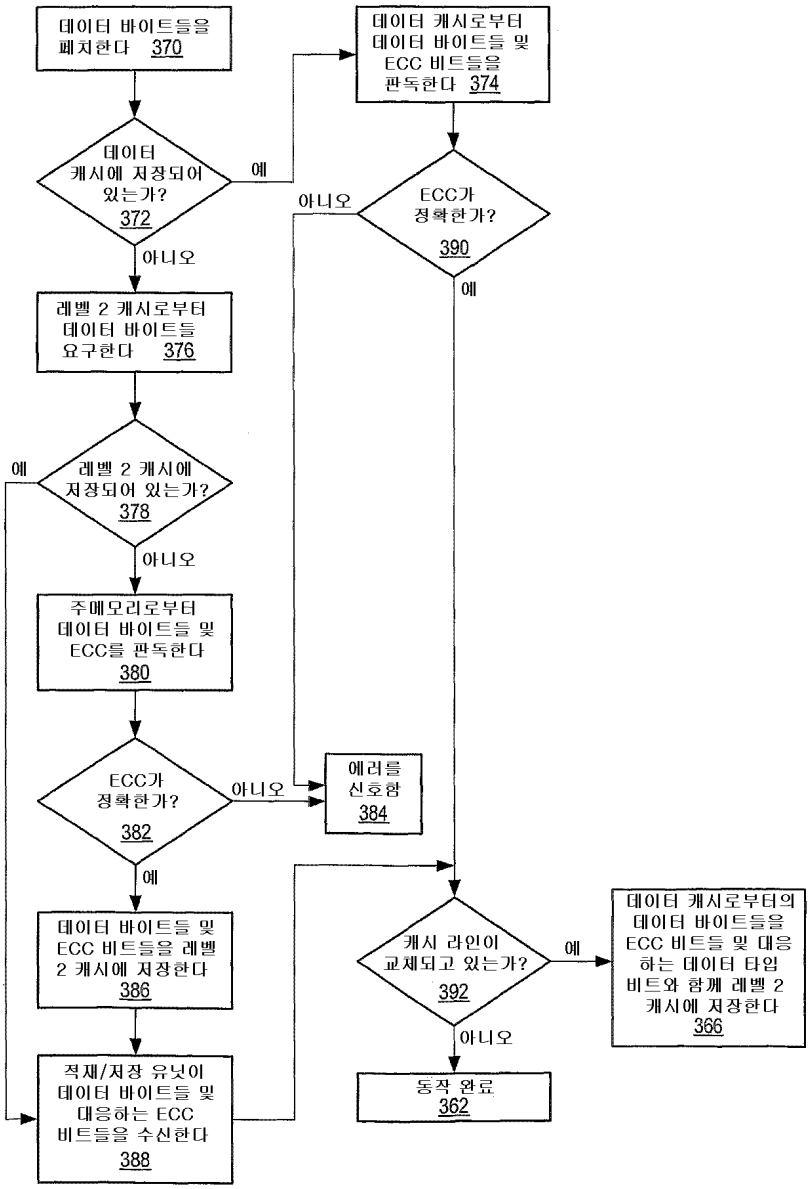
도면7B


레벨 1 캐시
명령 캐시 16
방향 0
방향 N
데이터 캐시 28
방향 0
방향 N
프리디코드 비트들 182A
명령 바이트들 184A
데이터 바이트들 188A
300
302
50
데이터 타입 비트 170
패리티 비트 172
프리디코드 또는 ECC 비트들 174
명령 또는 데이터 바이트들 176
코드 190
패리티 비트 180B
프리디코드 비트들 182B
명령 바이트들 184B
데이터 192
194
ECC 비트들 186B
데이터 바이트들 188B
패리티 유닛 402
패리티 유닛 314
ECC 유닛 312
Demux 316
318
320

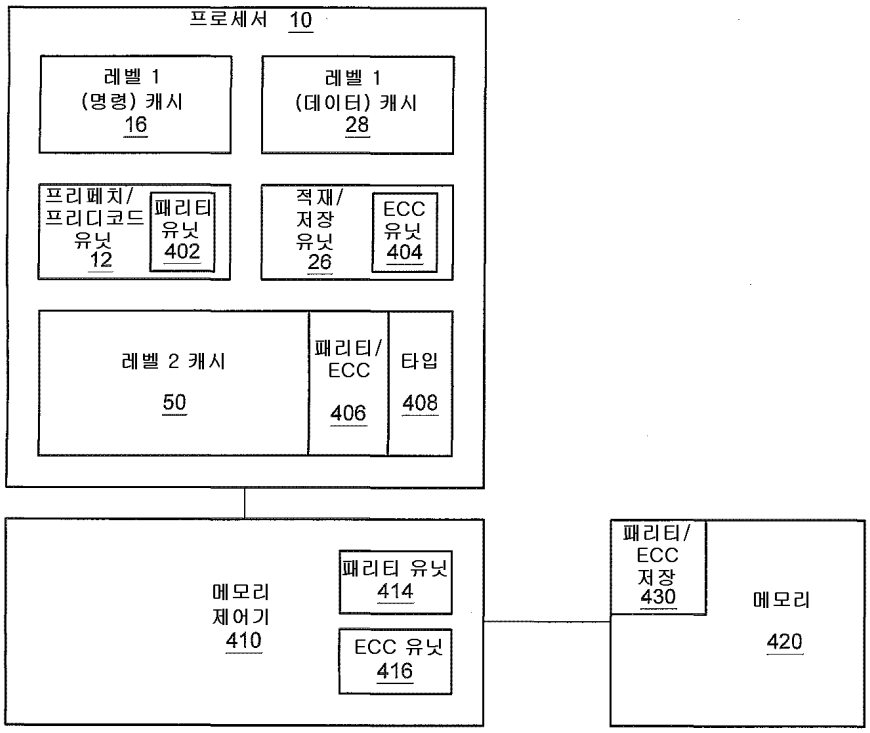
도면8A


명령 바이트들을 페치한다 240
명령 캐시에 저장되어 있는가? 242
예
명령 캐시로부터 명령 바이트들 및 프리디코드 바이트들을 판독한다 252
아니오
레벨 2 캐시로부터 명령 바이트들을 요구한다 244
레벨 2 캐시에 저장되어 있는가? 246
예
아니오
주메모리로부터 명령 바이트들 및 패리티 비트를 판독한다 248
명령 바이트들, 패리티 비트, 타입 비트, 및 프리디코드 비트들에 대한 무효화 상수를 레벨 2 캐시에 저장한다 250
프리디코드 유닛이 명령 바이트들을 수신하여, 대응하는 프리디코드 비트들을 발생시킨다 252
패리티가 정확한가? 254
아니오
에러를 신호한다 256
예
프리디코드 비트들이 무효화 상수와 같은가? 258
예
프리디코드 유닛이 새로운 프리디코드 정보를 발생시킨다 260
아니오
프리디코드 유닛이 명령 바이트들, 프리디코드 비트들, 및 패리티 정보를 명령 캐시에 저장한다 162
캐시 라인이 교체되는가? 264
예
프리디코드 정보 명령 바이트들, 패리티 정보, 및 대응하는 타입 비트를 레벨 2 캐시에 저장한다 266
아니오
동작 완료 268

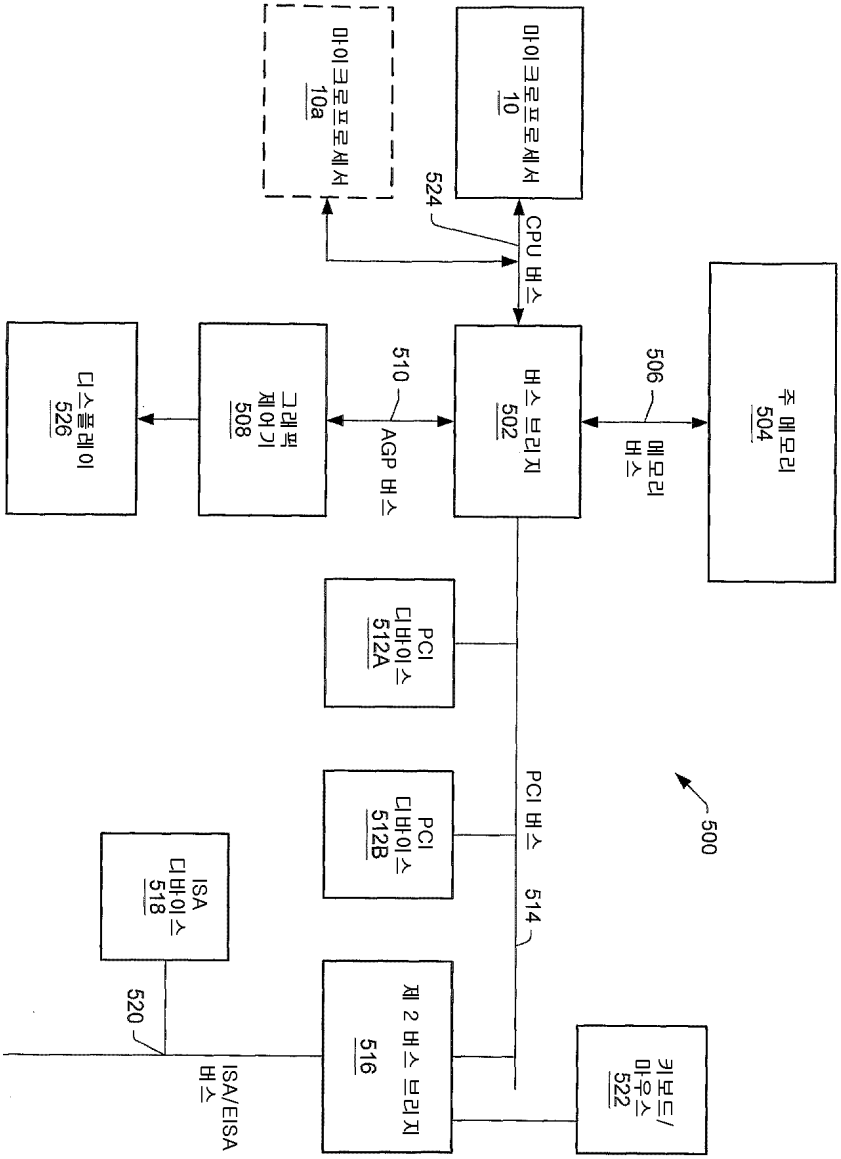
도면8B


데이터 바이트들을 페치한다 370
데이터 캐시에 저장되어 있는가? 372
예
아니오
데이터 캐시로부터 데이터 바이트들 및 ECC 비트들을 판독한다 374
ECC가 정확한가? 390
아니오
예
레벨 2 캐시로부터 데이터 바이트들 요구한다 376
레벨 2 캐시에 저장되어 있는가? 378
예
아니오
주메모리로부터 데이터 바이트들 및 ECC를 판독한다 380
ECC가 정확한가? 382
아니오
에러를 신호함 384
예
데이터 바이트들 및 ECC 비트들을 레벨 2 캐시에 저장한다 386
적재/저장 유닛이 데이터 바이트들 및 대응하는 ECC 비트들을 수신한다 388
캐시 라인이 교체되고 있는가? 392
예
데이터 캐시로부터의 데이터 바이트들을 ECC 비트들 및 대응하는 데이터 타입 비트와 함께 레벨 2 캐시에 저장한다 366
아니오
동작 완료 362

도면9


프로세서 10
레벨 1 (명령) 캐시 16
레벨 1 (데이터) 캐시 28
프리페치/프리디코드 유닛 12
패리티 유닛 402
적재/저장 유닛 26
ECC 유닛 404
레벨 2 캐시 50
패리티/ECC 406
타입 408
메모리 제어기 410
패리티 유닛 414
ECC 유닛 416
패리티/ECC 저장 430
메모리 420

도면10


마이크로프로세서 10
마이크로프로세서 10a
524
CPU 버스
주 메모리 504
506
메모리 버스
버스 브리지 502
510
AGP 버스
그래픽 제어기 508
디스플레이 526
PCI 디바이스 512A
PCI 디바이스 512B
PCI 버스 514
500
ISA 디바이스 518
520
ISA/EISA 버스
제 2 버스 브리지 516
키보드/마우스 522