(19) 대한민국특허청(KR)
(12) 특허공보(B1)

(51) Int. Cl.[5]
G06H 13/00

(45) 공고일자 1994년06월23일
(11) 공고번호 특1994-0005786

(21) 출원번호	특1988-0002102	(65) 공개번호	특1988-0011667
(22) 출원일자	1988년02월29일	(43) 공개일자	1988년10월29일

(30) 우선권주장 032,214 1987년03월30일 미국(US)

(71) 출원인 인터내셔널 비지네스 머신즈 코포레이션 하워드 지. 피거로아
미합중국, 뉴욕 10504, 아몬크

(72) 발명자 제리 듀안 딕슨
미합중국, 플로리다 33431, 보카 래이턴, 엔필드 스트리트 801
가이 질 스토매이어 2세
미합중국, 플로리다 33415, 웨스트 팜 비치, 이스트 캐롤써클 2104

(74) 대리인 이병호

심사관 : 허상무 (책자공보 제3667호)

(54) 퍼스널 컴퓨터 운영 방법

요약

내용 없음.

대표도

도1

명세서

[발명의 명칭]

퍼스널 컴퓨터 운영 방법

[도면의 간단한 설명]

제1도는 본 발명의 방법이 수행되는 퍼스널 컴퓨터 시스템의 개략도.

제2도는 본 발명에 사용된 캐쉬 데이타 구조의 개략도.

제3도는 페이지가 어떻게 해쉬(hash) 테이블로부터 참조되는가를 예시하는 개략도.

제4도는 프리 페이지 리스트를 도시하는 개략도.

제5도는 LRU 리스트 체인을 도시하는 개략도.

제6도는 본 발명에 여러 프로그램 및 하드웨어가 일반적으로 어떻게 관련되어 있는가를 도시하는 개략도.

제7도 내지 제12도는 본 발명의 방법을 도시하는 흐름도.

제13도는 에러 리스트 데이타 구조를 도시하는 블럭도.

* 도면의 주요부분에 대한 부호의 설명

12 : 프로세서 14 : 주메모리

15 : 입출력 회로 16 : 디스크 제어기

18 : 디스크 드라이브 20 : 드라이브 캐쉬

21 : 데이타 버퍼 26 : 프리 리스트 포인터

28 : LRU 포인터 30 : MRU 포인터

[발명의 상세한 설명]

본 발명은 기억 매체의 결함에 기인한 디스크 섹터 에러를 처리하기 위하여, 직접 액세스 기억 장치(DASD) 캐쉬를 가지고 있는 데이타 처리 시스템을 운영하는 방법을 개선하는 것에 관한 것이다. 특히, 본 발명은 DASD 캐쉬내의 불량한 섹터를 취급하는 문제에 대한 낮은 비용의 해결책을 제공하는 고성

능 퍼스널 컴퓨터의 운영 방법에 관한 것이다.

주메모리에 접속된 처리 장치에서의 처리를 위한 정보를 기억시키기 위해 고정 또는 하드 디스크 드라이브를 사용하는 것은 공지되어 있다. 정보는 예정된 패턴의 실린더 및 섹터에 따라서 디스크에 기억되는데, 각 섹터는 소정수의 바이트를 내포하고 있다. 드라이브는 복수의 헤드를 포함하며, 데이타가 기억되는 디스크의 각 면에 대해 각각 한개의 헤드가 존재하게 된다. 데이타는 한번에 한 섹터씩 디스크로부터 판독된다. 원하는 섹터를 억세스하기 위하여, 헤드는 먼저 원하는 섹터를 내포하고 있는 실린더로 이동되어야 하고, 디스크는 원하는 섹터에 도달될때 까지 헤드를 통과하여 회전하며, 다음에 섹터가 판독되고 그 내용은 버퍼내에 놓이게 된다. 디스크상의 데이타를 억세스하는데 요구되는 총 시간량을 고려해 볼때, 헤드의 물리적 이동중에 주된 지연이 발생한다. 그러므로, 처리에 대량의 I/O 동작이 수반될때, 개선된 성능을 성취하기 위해 가능한한 많은 헤드 이동의 정도를 감소시키는 것이 아주 바람직하다.

DASD 캐싱은 공지된 기술이며 헤드 이동량 및 물리적 I/O 동작의 양을 감소시킴으로써 시스템 성능을 개선하는 방법을 제공한다. 그러한 기술에 따라서, 주메모리의 일부는 데이타의 섹터의 페이지를 기억시키기 위한 캐쉬로서 사용된다. 원하는 섹터가 최초로 억세스된 경우, 그 섹터뿐 아니라 가까운 부가적인 하나 이상의 섹터들이 캐쉬내로 판독되며 그러한 섹터들로의 후속적인 억세스는 디스크 드라이브 속도 대신에 주 메모리 속도로 이루어진다. 처리될 다음 데이타가 이미 처리된 데이타 부근에 기억되는 확률이 높기 때문에 성능의 개선이 이루어진다.

이 기술에서 공지된 하나의 문제점은, 디스크 기억 매체에 결함이 존재한다는 사실로 인해 야기된다. 그러므로, 그러한 결함을 내포하고 있는 섹터는 불량한 것으로 간주되며, 사용될 수 없다. 그러한 불량 섹터는 통상적으로 포맷팅에 의해 식별되며, 후에 불량 섹터를 단순히 스킵함으로써 그 이용을 회피한다. 이 문제는 양호한 섹터가 초기에 캐쉬내로 판독될때, 동일한 페이지의 부근 섹터가 에러를 포함하거나 또는 불량한 것일 수 있기 때문에 캐싱 시스템에서 더 복잡해진다. 우리가 알고 있는 종래 기술 범위내에서, 이러한 문제는 두가지 방법으로 해결되었다. 첫째로, 불량 섹터를 내포하는 어떤 페이지는 그 자체가 결함이 있다고 간주되어 에러 신호 또는 메시지를 사용자에게 보낸다. 둘째로, 양호한 섹터만을 캐쉬로 보내도록 디스크 제어기가 구성될 수 있다. 이와 같은 해결방법은 하드웨어 해결책이며 성취하는데 비교적 복잡하며 비용이 많이 든다. 본 발명은 비교적 저가인, 고성능 퍼스널 컴퓨터에서 특히 유용한 효과적인 저가 해결법을 제공하기 위하여 소프트웨어 또는 프로그래밍에 의해 쉽게 구현될 수 있는 방법에 관한 것이다.

본 발명의 목적은 DASD 캐싱 시스템에서 디스크 섹터 에러를 처리하는 새로운 방법을 제공하는 것이다.

다른 목적은 CASD 캐싱 시스템을 가지고 있는 퍼스널 컴퓨터에서 섹터 에러를 처리하는 문제에 대한 낮은 비용의 해결책을 제공하는 것이다.

또 다른 목적은 DASD 캐싱 시스템에서 섹터 에러를 처리하는 방법으로서 공지된 하드웨어에 프로그램함으로써 쉽게 구현될 수 있는 방법을 제공하는 것이다.

또 다른 목적은 디스크 에러를 처리하는 문제를 쉽게 해결하면서, 사용자에게 명료한 캐싱 기능을 제공하기 위하여 상업적으로 이용가능한 디스크 오퍼레이팅 시스템(DOS)과 함께 쉽게 사용될 수 있는 방법을 제공하는 것이다.

간략이 말해서 본 발명에 따르면, 복수의 섹터를 내포하는 각 페이지가 주기억 장치내로 판독될때, 각 섹터가 양호한가 또는 불량한가에 관한 정보를 기억하는 개별 테이블이 만들어진다. 어떠한 섹터로의 후속 억세스는 그러한 테이블내의 정보를 참조하거나 조사하고, 불량하거나 에러가 있다고 지적된 섹터는 스킵함으로써 이루어진다.

본 발명의 다른 목적 및 장점은 첨부된 도면과 관련하여 설명된 다음의 설명으로부터 명백하게 될 것이다.

다음 설명은 두 부분으로 편성된다. 첫째는 DASD 캐쉬의 일반적인 동작을 설명하며 그러므로 본 발명의 방법이 운영되는 환경에 대하여 기술한다. 두번째 부분은 어떻게 매체 에러가 처리되는가를 상세하게 기술한다. 그러나, 본 발명은 어떤 특정의 소프트웨어 구현에 있는 것이 아니고 방법에 있다는 것을 이해해야 한다.

캐쉬(CACHE)

상기 방법은 양호하게 IBM 퍼스널 컴퓨터 AT와 같은 종래의 데이타 처리 시스템(10)에서 실행된다. 상기 시스템(10)은 통상 주메모리(14)와 I/O 장치(15)에 결합된 프로세서(12)를 포함한다. 이와 같은 시스템은 디스크 드라이브(18)에 각각 결합된 하나 이상의 디스크 제어기(16)를 포함할 수도 있다. 메모리(14) 내의 데이타 구조로써 캐쉬(20)와 데이타 버퍼(21)가 형성되는데, 후자는 디스크 드라이브(18)로부터 전송되는 데이타의 각 섹터를 버퍼하기 위해 DOS에 의해 이용된다.

제 2 도를 참조하면, 캐쉬(20)는 다수의 캐쉬 페이지(22-1 내지 22-n)를 포함하는 페이지 구조(21)로 이루어져 있다. 또한 캐쉬(20)는 해쉬 테이블(24), 프리(free) 리스트 포인터(26), LRU(least recently used) 포인터(28), MRU(most recently used) 포인터(30) 및 에러 리스트(31)를 포함하는데, 그의 상세한 설명은 지금부터 기술되게 된다. 캐쉬가 얼마나 큰지에 관계없이 항상 존재하는 여섯개의 부분이 있는데 다음과 같다.

캐쉬 페이지(22)-이것은 기억 장치내 캐쉬 페이지의 구조를 정의한다. 이것은, 어떤 물리적인 페이지가 표현되어지는 태그(tag)에 관한 정보, LRU 리스트의 일부, 해쉬 테이블 컨플릭트(conflict) 리스트의 일부 및, 섹터에 의해 그룹지워진 캐쉬 페이지의 실제 데이타를 포함한다.

최저 최근 사용(LRU) 포인터(28)-이것은 캐쉬내 최저 최근 사용 페이지인 캐쉬 페이지를 지시한다. 즉, 캐쉬내 다른 페이지 모두는 LRU 포인터에 의해 지시된 페이지보다 최근에 억세스된 것이다.

최대 최근 사용(MRU) 포인터(30)-이것은 캐쉬내 최대 최근 사용 페이지인 캐쉬 페이지를 지시한다. 즉, MRU 포인터에 의해 지시된 페이지 이상으로 최근에 억세스된 캐쉬내의 페이지는 없다.

캐쉬 해쉬 테이블(24)-이것은 캐쉬 페이지에 대한 포인터의 벡터이다. 상기 벡터는 311개 항목 길이이다. 한 섹터가 요청을 받으면, 캐쉬는 상기 요청을 차단하고, 상기 섹터가 캐쉬내에 있는가를 판정하기 위해 이 테이블내로 해쉬한다. 해쉬 테이블을 이용하므로써, 캐쉬내에 많은 수의 페이지가 있는 경우라 할지라도, 섹터가 캐쉬내에 있는지를 판정하는데, 매우 적은 시간이 걸린다.

프리 리스트 포인터(26)-이것은 캐쉬내에 현재 사용되지 않는 캐쉬 페이지를 지시한다. 처음에 캐쉬내의 모든 페이지는 이 리스트에 있다. 페이지가 캐쉬내에 있지 않은 것이 발견되면, 그 페이지는 프리 리스트로부터 제거되어 캐쉬내에 놓이게 된다.

에러 리스트(31)-이하 상세히 기술될 상기 데이타 구조는 페이지 에러의 이력(history)을 나타내기 위해 이용된다.

테이블 1은 캐쉬 페이지(22)의 필드를 리스트하고 있다. 이 구조내의 필드는 아래에서 정의된다. 모든 포인터는 두 바이트로서 유지된다는 것을 주목하자. 이것은 인텔 8086/80186/80286/80386 계열과 같은 세그먼트형 구조를 위한 것이다. 포인터에 기억된 값은 세그먼트(8086/80186) 또는 셀렉터(80286/80386)이다. 0으로 가정된 오프셋은 각 데이타 구조와 관련되어 있다.

여러 필드의 의미는 다음과 같다.

필드 1 이 필드는 페이지내의 어느 섹터가 존재하고 유효 데이타를 포함하는가 및 어느 섹터가 에러로 인해 상실되어 있는가를 나타내는 비트 또는 플래그를 포함한다.

필드 2 이 필드는 이는 드라이브에 이 특정 페이지가 포함되어 있는가를 결정한다.

필드 3 이 필드는 드라이브상의 페이지의 개시 섹터의 관련 블록 어드레스(RBA)를 포함한다.

필드 4 이 필드는 이 페이지보다 덜 최근에 사용된 다음 페이지를 지시한다. 만일 이것이 최저 최근 사용 페이지라면, 이 필드는 값 0을 포함하게 된다. 이 필드는 또한 프리 리스트에 페이지를 함께 결합시키는 데에도 이용된다.

필드 5 이 필드는 이 페이지보다 더 최근에 사용된 다음 페이지를 지시한다. 만일 이것이 가장 최근에 사용된 페이지라면, 이 필드는 값 0을 포함하게 된다.

필드 6 이 필드 해쉬(hash) 테이블의 컨플릭트 리스트에 있는 다음 페이지를 지시한다.

필드 7 이 필드는 해쉬 테이블의 불일치 리스트에 있는 이전 페이지를 지시한다.

필드 8 이 필드는 데이타 DASD로부터 빠져나온 페이지에 대한 데이타를 포함한다. 만일 이 페이지가 여러번 참조되면, 요청된 페이지의 부분은 그 장치로부터 판독되기 보다는 이 버퍼로부터 복사되게 된다.

[테이블 1]

캐쉬 페이지 22

필 드	내 용	바 이 트
1	존재 플래그	2
2	드라이브 ID	1
3	RBA	4
4	LRU 리스트의 다음 페이지	2
5	LRU 리스트의 이전 페이지	2
6	컨플릭트 다음 페이지	2
7	컨플릭트 이전 페이지	2
8	페이지 버퍼	1 내지 8섹터

각 RBA는 다음 공식으로 결정된다.

RBA=((CN×NH)+HN)×SPT+SN(공식 1)

여기에서,

CN=실린더수

NH=드라이브당 헤드의 수

HN=헤드수

SPT=트랙당 섹터의 수

SN=섹터 수

해쉬 인덱스는 다음 공식으로 결정된다.

해쉬 인덱스=RBA shr log(페이지 사이즈) mod 311(공식 2)

여기에서,

shr=페이지 사이즈의 2를 베이스로 한 로그에 따른 비트 수만큼 우로 시프트된 RBA

페이지 사이즈=페이지당 섹터의 수

mod=잔여분을 복귀시키는 모듈로 함수

331의 값은 비교적 큰 소수이기 때문에 해쉬 테이블 길이에 양호하다. 331이 큰 소수이기 때문에, 캐쉬를 억세스하는 동안에 응용 프로그램이 그 수나 또는 그 배수중 하나를 이용하게 될 가능성이 적다. 이것은 응용 프로그램이 해싱 알고리즘과 동기할 기회를 감소시켜 컨플릭트 체인을 짧게 유지하며, 그래서 해쉬가 빠르게 해결될 수 있다.

제 3 도를 참조하면, 테이블(24)의 여러가지 인덱스 위치 0 내지 310은 0을 포함하거나 또는 그위치에 연쇄된 제 1 의 캐쉬 페이지에 대한 포인터를 포함한다. 만약 선택된 해쉬 테이블 항목이 0의 값을 포함한다면, 그 항목에 의해 지시되는 어떤 페이지도 없으며 그 페이지는 캐쉬내에 없다. 이것은 "페이지 폴트"로 간주된다. 인덱스 위치 1은 예를들어, 캐쉬내에 함께 연관되어 있는 페이지가 없다는 것을 나타내는 제로를 포함한다. 한편 만약 해쉬 테이블 항목이 0이 아니라면, 캐쉬 페이지(22)를 지시하기 위해 포인터로서 사용되는 값이 있다. 캐쉬 페이지내의 드라이브 및 RBA는 요청된 드라이브 및 RBA와 비교된다. 그것들이 동일함이 판명되면, "페이지 히트"로 간주되고 요청된 페이지는 캐쉬에 존재한다. 그것들이 동일하지 않으면, 그 페이지에서 내의 컨플릭트 다음 값(Conflict Next Value)은 다음 캐쉬 페이지를 지시하기 위해 이용한다. 만일 0의 값이 이 필드로부터 픽업된다면, 컨플릭트 체인의 끝에 도달했다는 것이며, 요청된 페이지는 캐쉬에 없다. 이것도 또한 "페이지 폴트"이다. 설명한 것처럼 페이지 A 내지 F는 다른 길이의 세 분리 체인을 형성하기 위해 테이블(24)로 부터 청구된다.

만일 페이지(22)가 캐쉬에서 발견되면, 그 해쉬 테이블 항목에 대한 컨플릭트 체인은 발견된 페이지가 컨플릭트 체인의 선두에 있게 되도록 기록된다. 여기서의 개념은 컨플릭트 체인이 가장 최근에 사용된 순서로 유지된다면, 컨플릭트 체인 아래에서의 주사는 다시 억세스되는 최근 사용된 페이지의 가능성 때문에 짧게 된다는 것이다. 이것은 캐쉬 페이지가 하나 이상의 물리적 섹터를 나타내는 경우에 특히 정확하다.

페이지가 캐쉬에 없음이 판명되면, 새로운 페이지가 할당된다(이것을 수행하는 정확한 의미는 나중에 논의되게 된다). 페이지는 디스크로부터 판독되어 페이지 버퍼에 놓인다(필드 8). 새로운 캐쉬 페이지 구조는 페이지가 캐쉬가 없다는 것을 판정하는데 사용된 것과 비슷한 알고리즘으로 해쉬 테이블과 관련하여 구성된다. 여기에는 중요한 한가지 차이점이 있는데 해쉬 테이블 항목에 대한 컨플릭트 체인이 분해되지 않는다는 것이다. 대신에, 컨플릭트 체인이 가장 최근에 사용된 것에 관해 순서대로 유지되도록 컨플릭트 체인의 선두에 새로운 페이지가 삽입된다. 또한, LRU 체인 포인터는 이 페이지를 가장 최근 사용된 것으로서 나타내도록 다시 순서가 정해진다.

새로운 페이지(22)는 요청된 섹터가 캐쉬에서 발견되지 않을 때에만 캐쉬(20)에 놓이게 된다. 이 방법은 전술되었다. 여기서는, 페이지가 캐쉬 구조에 놓일 수 있도록 실제적으로 어떻게 할당되는지에 관해 설명한다. 페이지가 할당되는 것을 두가지 방법이 있다. 캐쉬 프리 리스트내에 한개 또는 그 이상의 페이지가 있다면, 한 페이지는 프리 리스트로부터 제거되어 새로운 페이지에 할당된다. 만약 캐쉬 프리 리스트내에 어떤 페이지도 없다면, LRU 포인터에 의해서 지시된 페이지는 캐쉬 구조로부터 제거되어 새로운 페이지로 할당된다. 제 4 도는 캐쉬 프리 리스트의 구조를 도시하고 있다. 프리 리스트 포인터(26)는 리스트 선두에 있는 캐쉬 페이지(22)를 지시하며, 포인터 필드중의 하나는 다음 프리 페이지를 지시한다. 처음에, 모든 캐쉬 페이지는 프리 리스트에 있으며, 해쉬 테이블 혹은 LRU 리스트에 의해 지시되는 페이지는 없다. 디스크에의 요청이 이루어지고 캐쉬 페이지 폴트가 있을때에는, 최종적으로 프리 리스트에 어떠한 페이지도 없게될때까지 프리 리스트로부터 더 많은 페이지(22)가 제거되게 된다. 이러한 일이 발생하면, 페이지는 LRU 리스트에 근거하여 재사용된다. 제 5 도는 리스트의 끝을 지사하는 LRU 및 MRU 포인터(28, 30)와, LRU 다음(Next) 포인터 및 LRU 이전(Previous) 포인터에 의해 연되어 있는 여러 페이지(22)가 있는 LRU 리스트의 구조를 도시하고 있다.

캐쉬는 항상 LRU 리스트로부터 페이지를 제거하기 전에 프리 리스트로부터 할당하려고 시도하기 때문 않으면, CACHESYS는 BIOS(44)를 호출하여 디스크(18)로부터 판독된 데이타를 버퍼(21)로 가져오도록 하며, 이때부터 데이타가 응용 프로그램에 이용가능하게 된다. 더욱이, 이와 같은 데이타를 최초로 이용하는 경우 데이타는 또한 후속 동작에 이용가능하게 되도록 캐쉬(20)에 위치되게 된다. 부가적으로, 가까운 연속한 섹터는 또한 사전에 팻치되거나 캐쉬(20)로 판독된다.

제 7 도를 참조하면, 인터럽트를 인터셉트함으로써 절차(50)가 호출되면, 단계(100)는 원하는 섹터의 RBA로부터 캐쉬 테이블(24)로의 지표(index)를 결정 또는 계산한다. 다음에 단계(102)는 그 지표에서의 해쉬 테이블의 내용을 PAGE PTR로서 알려진 변수로 놓고 단계(104)는 컨플릭트 체인의 끝을 표시하기 위해 PAGE PTR이 제로가 되는지의 여부를 판정한다. 만일 제로라면, 단계(106)에서 새로운 페이지가 판독되고, 이 새로운 페이지는 단계(108)에 의해 캐쉬에 놓이게 되며, 이후 단계(110)는 DOS 및 응용 프로그램으로 되돌아간다. 만일 단계(104)의 결과로서 페이지 포인터(PAGE PTR)가 제로가 되지 않으면, 단계(112)(114)는 페이지 포인터가 원하는 페이지를 지시하고 있는지를 판정한다. 이것은 먼저 단계(112)에서 데이타를 포함하는 원하는 드라이브와 페이지 포인터 드라이브 명세를 비교하고, 만일 그것이 올바른 드라이브라면, 단계(114)에서 페이지 포인터의 RBA와, RBA 마스크와 논리적(AND)을 취한 RBA를 비교함으로써 행해진다. 만일 단계(112)(114)가 부정(negative)결과를 발생하면, 단계(116)는 컨플릭트 체인에서 다음 페이지를 지시하기 위해 페이지 포인터를 갱신하고, 이 공정은 원하는 페이지에 도달하여 단계(114)에서의 공정 결과가 단계(118)로 전달될때까지 계속되게 된다.

단계(118-130)의 일반적인 목적은 캐쉬 페이지내에 요청된 섹터가 있는지의 여부를 검사하기 위한 것이

다. 만일 요청된 섹터가 있으면, 단계(130)로부터의 공정 결과가 복귀 단계(110)으로 전달되고 섹터 데이타는 그 캐쉬 페이지(22)로 부터 버퍼(21)로 전달된다. 만일 요청된 섹터가 없으면, 원하는 섹터를 포함하는 페이지에서의 판독을 위한 단계(130)로부터의 부정 결과가 단계(106)로 되돌아간다. 단계(118-130)동안에 어떤 일이 일어나는지 이해하기 위해서는, 단계(106)를 참작하고 새로운 페이지가 캐쉬로 어떻게 판독되는지를 고려해보는 것이 바람직하다.

단계(106)는 일반적인 단계이며, 거기에 포함된 세부 단계는 제 8 도 내지 제12도에 도시되어 있다. 그러나, 이들 여러 단계를 상세하게 설명하기 전에, 이들 단계에서 사용된 몇몇 변수 및 데이타 구조에 대한 설명이 이루어지게 된다.

[에러 처리]

테이블 2에는 공정에 사용되는 상이한 변수의 의미가 리스트되어 있다. 이들 변수는 공정의 개시시에 초기 설정된다. 에러 리스트(31)(제 2 도)는 제13도에 개략적으로 도시된 바와 같이, 에러 해쉬 테이블(150)과, 컨플릭트 체인(152) 및 프리 리스트(154)를 포함하는데, 1페이지당 1항목(entry)이 있다. 컨플릭트 체인과 프리 리스트는 제 3 도 및 제 4 도에 대해 설명된 절차와 비슷하게 구성되고 처리된다. 해쉬 테이블을 이용하는 이유는 에러 리스트가 비교적 적다할지라도 물리적 입출력(I/O)이 있을때마다 그것이 주사되고 탐색 시간을 가능한한 짧게 유지하는 것이 바람직하기 때문이다. 해쉬 테이블(150)은 64지표 길이인데, 이런 길이가 선택되는 이유는 그것이 2의 거듭제곱이 되고 페이지 RBA가 제산(나눈셈)보다는 시프트와 마스크로 용이하게 조작될 수 있기 때문이다. 항목의 소수보다 이 수를 선택한 이유는 디스크상의 에러가 보통 거의 균일하게 분포되어 있고, 캐쉬가 물리적 입출력을 행할 때마다 이 테이블이 억세스되게 되므로 속도가 매우 중요하기 때문이다. 해쉬 테이블(150)은 에러 리스트 항목의 컨플릭트 체인을 지시한다.

[테이블 2]

변수

변 수	의 미
RBA	판독될 페이지의 RBA
BUFFER POINTER	데이타가 판독되게 될 포인터
PAGE SIZE	페이지내의 섹터수(2, 4 또는 8)
RBA MASK	=NOT(PAGE SIZE-1)(즉, OFFFFFFFC)
RBA SHIFT	=log(PAGE SIZE)(즉, PAGE SIZE 2, 4 또는 8에 대해서 1, 2 또는 3)
SEC COUNT MASK	PAGE SIZE-1
ERROR MASK	=비트수=PAGE SIZE 한 워드에서 우측으로 치우쳐 정렬된 것(즉, PAGE SIZE 2, 4 또는 8에 대해 0003, 000F, 00FF)

테이블 3은 에러 리스트내의 각 항목의 구조를 정의한다.

[테이블 3]

에러 리스트 구조

필 드	내 용	바 이 트	의 미
1	존재 플래그	2	이 필드는 페이내의 섹터가 에러를 갖고 있다는 것을 나타내는 일련의 플래그 또는 비트를 포함한다. "불량한(BAD)" 섹터는 필드내의 대응하는 위치에서 0비트로 표시되고 유효 데이타가 존재하는 섹터는 1비트로 표시된다.
2	드라이브 ID	1	이 필드는 페이지가 위치되는 드라이브의 드라이브 번호를 포함한다.
3	블럭 어드레스	4	이것은 페이지의 제 1 섹터의 RBA이다.
4	다음 컨플릭트 항목	2	이 필드는 컨플릭트 체인내의 다음 에러 리스트 구조 또는 항목을 지시한다.
5	이전 컨플릭트 항목	2	이 필드는 컨플릭트 체인내의 이전 에러 리스트 구조를 지시한다.

제 8 도를 참조하면, 새로운 페이지에서의 판독을 위한 공정의 개시는 단계(132)로 시작된다. 제 8 도에 도시된 일반적인 공정은 캐쉬가 판독될 페이지에서 에러를 이미 검출했는지를 판정한다. 단계(132)에서, 원하는 섹터의 RBA는 RBA 마스크와 논리적이 취해지며, 그 결과는 에러 해쉬 테이블(150)로의 지표를 계산하기 위해 단계(134)에서 이용된다. 지표는 RBA 시프트의 양만큼 단계(132)로부터의 결과를 우측으로 시프트하여 모듈로 64 연산을 행한 값의 나머지로서 계산된다. 다음에, 단계(136)에서 변수 PTR(포인터)가 단계(134)로 부터 계산된 지표에 위치한 에러 해쉬 테이블의 내용에 설정된다. 만일 PTR이 제로로 설정되지 않으면, 단계(138)는 단계(140)으로 분기되며, 컨플릭트 체인에 에러 항목이 있는 상태를 표시한다. 다음에, 단계(140)는 드라이브 ID가 에러가 발생한 페이지를 포함하는 드라이브의 ID와 동일한지를 판정한다. 만일 같지 않으면, 단계(144)는 포인터를 지표하고 컨플릭트 체인의 다음 항목으로 나아간다.

만일 단계(140)의 결과가 긍정이면, 단계(142)는 원하는 RBA가 에러 리스트 항목의 RBA와 일치하는지를 판정한다. 만일 그렇지 않다면, 컨플릭트 리스트내의 다음 항목으로 나아간다. 단계(138)로부터의 긍정 출력을 관련 페이지가 전에 에러를 갖고 있지 않고, 그러므로 페이지내의 섹터수를 판독하기 위해 디스크 제어기에 단일 명령어를 전송함으로써 전체 페이지를 판독하기 위한 시도가 이루어지게 된다는 것을 표시한다. 단계에, 캐쉬가 동작 상태에 있으면 캐쉬에 더 많은 페이지를 동적으로 추가하는 것이 가능하다. 이것은 캐쉬가 오직 소프트웨어로만 구현되는 경우에 특히 바람직하다. 그 배후의 이유는 오퍼레이팅 시스템이, 주기억장치가 완전하게는 이용되지 않는다는 것을 알고, 미사용 기억 장치의 일부(또는 전부)를 캐쉬에 제공함으로써 그 기억 장치의 일부가 양호한 이용상태에 놓일 수 있게 된다는 것을 판정할 수도 있다는 것이다.

LRU 리스트의 끝으로 부터 하나 또는 그 이상의 페이지를 제거함으로써 그 역(캐쉬로부터의 페이지 제거)도 또한 가능하다. 이것은 페이지가 기억 장치내에서 물리적으로 연속이 되는 것을 보장할 수 없기 때문에 바람직하지 않을 수도 있다. 그러나 충분히 정교한 오퍼레이팅 시스템이 주어진다면, 기억 장치의 단편화는 문제가 되지 않을 수도 있다.

페이지가(히트 또는 할당중의 새로운 페이지에 의해) 캐쉬내에서 억세스될때마다 그 페이지는 LRU 리스트내의 그 현재 위치로부터 제거되어, MRU 포인터에 의해 지시되는 리스트의 선두로 이동된다. 캐쉬 페이지 구조가 포인터를 근거로 하고 있기 때문에, LRU 리스트에서의 페이지 이동은 단순하게, 공지된 방식으로, 포인터 값을 주위로 이동함으로써 이루어진다.

캐쉬 피닝(영구적으로 캐쉬내에 페이지를 위치시키는 것)은 본 설계로 쉽게 수행된다. 페이지를 피닝하기 위해서는 LRU 리스트로부터 페이지를 제거하기만 하면 된다. 페이지가 LRU 리스트로부터 재할당되기 때문에, LRU 리스트에 없는 페이지는 결코 재할당될 수 없다. 피닝은 어떤 페이지가 캐쉬내에서 그것들을 유지하기에 충분하게 자주 억세스되지 않을때 바람직하다. 이것은 디렉토리, 할당 비트맵, 시스템 파일 등과 같은 파일 시스템 구조를 대형 파일의 판독에 의해 플러시(flush)되는 일 없이 캐쉬내에서 유지시켜 양호한 성능을 얻을 수 있도록 한다. 유용한 방법으로 캐쉬 피닝을 수행하기 위해서는, 캐쉬와 오퍼레이팅 시스템 또는 적어도 오퍼레이팅 시스템의 유틸리티(utility) 사이의 협동을 이루는 것이 바람직하다. 이 유틸리트는 다른 신호가 있을때까지 요청된 페이지가 LRU에 위치되지 않는다는 것을 캐쉬에 알린다. 다음에, 이 유틸리티는 섹터, 파일등을 판독하고 모든 다른 캐쉬 요청이 페이지를 LRU에 위치시켜야 한다는 것을 캐쉬에 신호한다.

캐쉬는 일부의 페이지가 LRU의 일부가 되지 않을수도 있고 그래서 억세스될때 LRU내로 삽입되지 않아야 된다는 사실을 인식할 필요가 있다. 이것은 LRU 다음 필드 및 LRU 이전 필드가 0이 되는지 알기위해 검사함으로써 캐쉬 히트가 발생할때 행해질 수 있다. 만일 이들 필드가 0이라면, 그 페이지는 LRU에 위치되지 않게 된다. 피닝을 지원하기 위해 특수 비트나 플래그가 캐쉬 페이지 구조에 포함될 필요는 없다.

제 6 도는 발명의 방법이 소위 CACHESYS로 불리는 일련의 절차로 구체화되는 일반적인 방식을 예시하고 있는데, 이 절차는 통상의 프로그램 및 하드웨어와 대화하여 본 발명의 방법을 실행하게 된다. 응용 프로그램(40)이 디스크(18)로 부터 데이타를 판독할 필요가 있을 때에는, DOS(42)가 호출되어, 통상의 인터럽트(13H)를 이용하여 기본 입력 및 출력 시스템(BIOS, 44)을 호출한다. CACHESYS(50)는 DOS(42)와 BISO(44) 사이에 삽입되며, 다음에 상세히 설명되는 여러 절차를 인터셉트하여 실행하기 위한 인터럽트 처리 루틴으로서 구성될 수 있다. CACHESYS(50)는 DOS에 대해서는 BIOS(44)로서 동작하는 것처럼 보이고, BIOS(44)에 대해서는 CACHESYS 시스템이 DOS(42)처럼 보인다. 즉, CACHESYS(50)의 동작은 DOS(42)와 BIOS(44)에 대하여 투명(transparent)하다. BIOS(44)의 제어하에, 디스크(18)로부터의 데이타는 하드웨어(46)를 통해 주기억 장치내의 데이타 버퍼(21)로 전송되게 되며, 통상적인 방식으로 DOS를 거쳐 응용 프로그램에서 이용가능하게 된다. CACHESYS(50)가 동작하고 있을때, 인터럽트 13H 호출의 인터셉트에 응답하여, 만일 필요한 데이타를 포함하는 원하는 섹터가 이미 드라이브 캐쉬(20)내에 있으면, 데이타는 버퍼(21)에 놓이게 되며 응용 프로그램에서 이용가능하게 된다. 만일 데이타가 캐쉬(20)에 있지 (142)로부터의 긍정 판정은 페이지가 이전에 에러를 갖고 있고, 매번 한 섹터만을 판독하기 위해 디스크 제어기에 복수의 명령어를 전송함으로써 한 섹터씩 판독되게 된다는 것을 표시한다.

제 9 도를 참조하면, BIOS를 호출하여, 단계(160)는 페이지 사이즈, 페이지당 섹터수 및 개시 섹터의 RBA를 얻는다. 만일 판독중에 에러가 발생하지 않았다면, 단계(162)는 페이지에 모든 섹터가 존재한다는 것을 표시하는 존재 비트를 설정하는 단계(164)로 분기한다. 즉 불량 섹터가 없거나, 이와 같은 섹터의 판독중에 부딪히는 에러가 없으며, 각 섹터가 유효 데이타가 존재한다. 단계(166)는 섹터가 판독되게 되는 데이타 버퍼(21)의 어드레스를 지시하기 위한 변수 CURPTR를 설정한다. 다음에 단계(168-176)는 표시된 여러가지 변수를 초기 설정한다. 다음에, BIOS에의 호출이 이루어지고, 단계(178)에서 현재 RBA와 관련된 한 섹터가 판독된다. 만일 에러가 발생하지 않으면, 단계(180)는 단계(182)로 분기한다. 만일 판독중에 에러가 발생하면, 이와 같은 에러와 관련된 존재 플래그는 비트 마스크와의 배타적-OR를 취함으로써 단계(184)에서 제로로 설정하여 에러 또는 섹터가 불량하다는 것을 표시하게 된다. 다음에, 단계(186)는 단계(178)의 판독 동작으로 부터 유래한 복귀 코드에 따라 에러 코드를 설정한다. 단계(182-188)는 다음 섹터를 표시하기 위해 현재 RBA, 현재 포인터, 비트 마스크 및 카운터를 갱신하고, 공정은 카운트가 제로로 감소될때까지 단계(190)에 의해 반복된다. 단계(192)는 다음에 방금 판독된 페이지가 에러 테이블(31)에 있는지를 판정한다.

만일 그 페이지가 에러 테이블에 있지 않으면, 단계(192)는 단계(193)(제10도)로 분기하며, 여기서 새로운 에러 포인터를 설정함으로써 에러 프리 리스트(154)로부터 요소(element)를 얻는다. 다음에, 단계(194)는 에러 프리 리스트가 비어 있는지, 즉 새로운 에러 포인터가 제로가 되는지를 판정한다. 만일 그렇다면, 단계(194)로부터 단계(204)로 분기가 이루어지고, 여기서 존재 비트 및 에러 코드가 되돌려진다. 만일 에러 프리 리스트가 비어 있지 않으면, 단계(196-202)는 새로운 에러 항목을 에러 해쉬 테이블에 삽입한다. 이것은 단계(186)에서 에러 프리 포인터를 다음의 에러 프리 포인터에 동일하게 설정하고, 단계(198)에서 현재의 새로운 에러 포인터를 대응하는 존재 비트에 동일하게 설정하고, 에러 해쉬 테이블의 지표로부터의 값에 따라 다음의 새로운 에러 포인터를 설정하고, 단계(202)에서 새로운 에러

포인터에 에러 해쉬 지표를 설정함으로써 행해진다.

제11도에 도시된 절차는 이전의 에러에 대한 요청된 섹터를 검사한다. 단계(206-218)는 단계(118-130)와 동일하지만 입구점과 출구점에 따라 변화한다. 그러므로 이와 같은 변화로 인해, 단계(118-130)는 요청된 섹터가 캐쉬 페이지에 있는지 알기 위해 검사하고, 모든 섹터가 한 페이지내에 있을 때에는 단계(130)로부터 긍정 결과를 발생한다. 부정 결과는 요청된 섹터 모두가 한 페이지내에 있는 것이 아니고, 그래서 오퍼레이팅 시스템이 형의 에러 회복을 실행하기 위한 시도를 해야 한다는 것읖 표시한다. 다른 한편으로, 단계(206-218)는 요청된 섹터가 이전 에러를 갖고 있었는지 검사하는데 이용된다. 단계(218)로부터의 긍정결과는 그 페이지가 에러를 갖고 있지만 요청된 섹터는 갖고 있지 않다는 것을 표시하며, 반면에, 단계(218)로 부터의 결과가 부정이면, 요청된 섹터가 에러를 갖고 있다는 것을 표시한다.

단계(206)에서, RBA와 마스크 SECCOUNT는 논리적이 취해지고 그 결과는 변수 시프트 카운트에 기억된다. 다음에, 단계(208)는 존재 비트를 시프트 카운트의 양만큼 우측으로 시프트된 포인터와 동일하게 설정한다. 단계(210)는 마스크 시프트를 페이지 사이즈에서 섹터수를 뺀값과 동일하게 설정한다. 단계(212)에서는 마스크 시프트에 따른 양만큼 에러 마스크를 우측으로 시프트함으로써 테스트 마스크를 작성한다.

이후, 단계(214)에서 존재 비트는 테스트 마스크와 논리적이 취해지고, 다음에 단계(216)에서 테스트 마스크와 논리 배타적-OR가 취해지며, 이때 단계(218)는 그 결과를 테스트한다. 이와 같은 절차는 루프에 인가하거나 복수의 결정 경로없이 요청된 섹터에 에러가 있는지를 빠르게 판정하기 때문에 장점이 된다. 또한 이것은 페이지 사이즈가 8섹터이고, 마지막으로 페이지를 판독했을 때의 그 페이지의 섹터 5(6번째 섹터)가 에러를 갖고 있으며 그래서 그 대응하는 존재 비트가 제로로 설정되었다고 가정한 다음의 예를 이용하여 보다 잘 이해될 수 있다. 테이블 4에는, 요청된 섹터가 에러를 갖고 있는 페이지내에 있지만 요청된 섹터가 그 에러중 하나가 아닐때의 절차가 어떻게 동작하는지의 예가 도시되어 있다. 테이블 5에는 요청된 섹터가 이전의 판독중에 에러를 갖고 있었을 때의 절차의 예가 도시되어 있다. 이 예는 페이지의 섹터 4에서 시작하여 2개의 섹터를 판독하게 된다.

[테이블 4]

요청된 섹터에 에러가 없음

0000	0000	1111	1111	에러 마스크
0000	0000	0000	0011	테스트 마스크
0000	0000	1101	1111	현재 플래그
0000	0000	0001	1011	단계 212
0000	0000	0000	0011	단계 214
0000	0000	0000	0000	단계 216

[테이블 5]

요청된 섹터에 에러가 있음

0000	0000	1111	1111	에러 마스크
0000	0000	0000	0011	테스트 마스크
0000	0000	1101	1111	현재 플래그
0000	0000	0000	1101	단계 212
0000	0000	0000	0001	단계 214
0000	0000	0000	0010	단계 216

제12도를 참조하면, 단계(220)는 단계(218)(제11도)에서의 부정 판정으로 이루어지며 PTR 존재 비트에 따라 존재 비트가 설정되도록 한다. 단계(222)는 카운트를 PAGE SIZE와 동일하게 설정하며, 단계(224)는 CURRBA를 RBA와 동일하게 설정하고 단계(226)는 CURPTR을 데이타 버퍼에의 포인터와 동일하게 설정하며, 단계(228)는 비트 마스크를 1로 설정한다. 다음에, 단계(230)는 존재 비트와 비트 마스크를 논리적을 취함으로써 현재 섹터가 전에 에러를 갖고 있었는지를 판정한다. 만일 그 결과가 제로가 아니면, 단계(232)에서는 BIOS 호출을 이용하여 CURRBA에서 한 섹터를 판독한다. 다음에, 단계(234)는 이와 같은 판독중에 에러가 있었는지를 판정한다. 만일 에러가 있었다면, 단계(236)는 에러가 있었다는 것을 표시하기 위한 현재 플래그를 설정하고 단계(238)에서 에러 코드를 세이브한다. 단계(240) 및 그 이후 단계는 단계(240)에서 CURRBA를 증분함으로써 다음 섹터로 이동한다. 다음에, CURPTR가 단계(242)에서 섹터당 바이트 수만큼 증분되고, 비트 마스크가 단계(244)에서 좌측으로 1만큼 시프트된다. 카운트는 단계(246)에서 증분된다. 카운트가 제로가 되면, 단계(204)(제10도)에서 분기가 이루어진다. 그렇지 않으면, 단계(248)에서는 모든 섹터가 판독되지 않았는지를 판정하고, 그러므로 공정을 반복하기 위해 단계(230)로 분기가 이루어진다.

다음은 절차의 동작을 요약한 것이다. 매체의 결함은 캐쉬 페이지 구조내의 현재 플래그로 관리된다. 디스크로부터 페이지 판독의 시도가 이루어질때 만일 에러가 되돌아오면, 캐쉬는 다음 방식으로 회복된다.

1. 페이지의 최초 섹터로 되돌아간다.

2. 한번에 한 섹터씩 페이지 버퍼로 섹터를 판독하기 시작한다.

3. 만일 섹터를 판독하는 동안에 에러에 부딪히면, 그 섹터에 대한 현재 플래그가 클리어되고(0으로 설정되고), 섹터가 판독되었을때처럼 적당한 포인터가 갱신된다.

4. 페이지내의 모든 섹터가 판독되었으면(한번에 한 섹터씩), 존재 비트는 캐쉬로부터 요청된 하나 또는 그 이상의 섹터가 누락되었는지 즉, 대응하는 존재 비트가 0으로 설정되었는지 판정하기 위해 주사된다. 만일 그렇다면, 장치로 부터의 판독중에 수신된 최종 에러가 요청자에게 되돌려진다.

이 설계는 캐쉬를 한층 복잡하게 함으로써 데이타의 손실없이 매체내의 에러에 대응하는 "홀(hole)"을 캐쉬가 가질 수 있도록 한다. 이것은 또한 캐쉬가 오퍼레이팅 시스템의 작업을 더 어렵게 만들지 않고 오퍼레이팅 시스템이 연산 재시도 및 에러 상황에서의 데이타 회복을 시도하도록 허용한다.

캐쉬가 에러에 부딪히면, 페이지의 어느 섹터에 에러가 있으며, 에러가 있는 섹터에 대응하는 "홀"을 페이지에 남겨 두는지 판정한다. 또한 에러 회복 공정에 의해 존재 플래그로서 언급되는 16비트 값이 발생된다. 만일 이들 플래그에 어떤 제로 비트가 있으면, 페이지는 하나 또는 그 이상의 에러를 갖고 있다.

만일 페이지에 에러가 있으면, 캐쉬는 그 페이지가 전에 에러를 갖고 있었는지를 판정한다. 이것은 그 페이지가 리스트내에 있는지 알기 위해 페이지의 RBA로 에러 리스트에 해쉬함으로써 행해진다. 만일 그 페이지가 리스트상에 있지 않으면, 그 페이지는 리스트에 부가된다. 만일 그 페이지가 전에 에러를 갖고 있었다면, 다음중 하나가 행해질 수 있다.

1. 이전의 존재 플래그를 유지한다.

2. 존재 플래그를 바로 발생된 플래그를 대치한다. 이것은 소프트웨어 캐쉬가 구현한 것이다.

3. 존재 플래그의 2셋트를 논리적을 취하고, 그 결과를 세이브한다. 이것은 페이지내의 섹터가 에러를 갖고 있었다면, 특별히 요청되지 않으면 판독되지 않게 된다는 것을 의미한다.

4. 존재 플래그의 2셋트를 논리적 OR를 취하고 그 결과를 세이브한다. 이것은 페이지내의 섹터가 성공적으로 판독되었다면, 캐쉬는 항상 그 섹터의 판독을 시도하게 된다는 것을 의미한다.

만일 존재 플래그내의 모든 비트가 모두 제로라면, 그 항목은 에러 리스트에서 제거된다(적어도 위치되지는 않는다). 이것은 몇몇 이유로 인해 행해지는데, 그 첫번째 이유는 에러 리스트가 더욱 작아질 수 있도록 한다는 것이다. 또한 제조 매체 결함이 있을때 오퍼레이팅 시스템은 전체 트랙을 할당-해제하는 경향이 있기 때문에, 오퍼레이팅 시스템은 일반적으로 이들 영역의 판독을 시도하지 않는다. 두번째 이유는 만일 페이지가 양호한 섹터를 갖고 있지 않으면, 오퍼레이팅 시스템은 일반적 에러 회복 상태에 있고 성능이 위험하지 않다는 것이다. 또한 캐쉬는 페이지에 거의 관계가 없으므로 그 에러의 트랙을 유지함으로써 캐쉬가 공간을 낭비하는 일이 없다.

캐쉬 누락을 초래하는 캐쉬에의 요청이 이루어지면, 시스템은 먼저 판독할 페이지가 전에 에러를 갖고 있었는지 판정한다. 만일 그 페이지가 이전에 에러를 갖고 있었다면, 시스템은 요청된 섹터중 하나가 에러로서 플래그되어 있는지를 본다. 만일 에러를 갖고 있는 요청된 섹터가 없다면, 에러가 있는 섹터를 스킵함으로써 페이지의 섹터가 한번에 하나씩 판독된다. 그러므로 에러가 회피된다.

요청된 섹터가 에러를 갖고 있었다면, 캐쉬에 의해 판독되게 되므로 요청자에 의한 에러 재시도가 허용된다. 에러 회피가 행해지는 것은 그 섹터가 명시적으로 요청되지 않은 경우에 캐쉬가 전에 에러가 있었던 섹터를 사전에 팻치하는 것을 방지할 뿐이다.

이 기술에 숙련된 사람을 첨부된 청구범위에서 정의된 바와 같은 본 발명의 정신 및 범위를 벗어나지 않고 다른 변화 및 변형이 이루어질 수 있다는 것을 알 수 있다.

(57) 청구의 범위

청구항 1

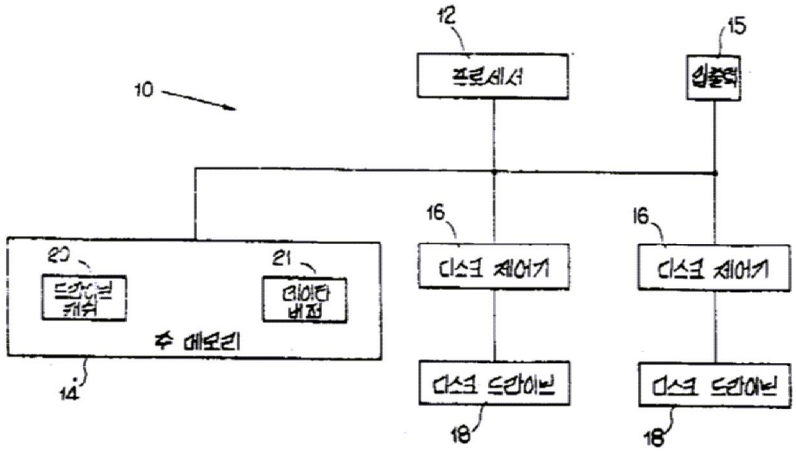
기억 매체 결함이 있는 디스크로부터 한번에 한 섹터씩 연속적으로 판독되는 데이타 섹터의 페이지를 기억시키기 위한 DASD 캐쉬를 제공하는 주메모리를 가지며, 오퍼레이팅 시스템이 전송중인 섹터를 버퍼링하여 응용 프로그램하에 처리하기 위해 상기 DASD 캐쉬와 상기 주메모리 사이의 데이타 전송을 관리하도록 되어 있는 퍼스널 컴퓨터를 운영하는 방법에 있어서, 상기 디스크로부터의 소정의 섹터로부터 데이타를 판독하기 위한 요청 신호를 제공하는 단계와, 상기 요청 신호에 응답하여, 상기 디스크상의 연속적인 위치에 위치되는 소정의 섹터 및 추가적인 사전 팻치된 섹터를 포함하는 페이지를 상기 디스크로부터 판독하는 단계로서, 상기 캐쉬에서 상기 판독중에 에러가 발생되지 않는 섹터로부터 판독되는 데이타를 기억시키는 단계와, 이와 같은 섹터로부터의 유효 데이타가 상기 캐쉬에 존재한다는 것을 표시하기 위해 이와 같은 섹터에 대해 상기 캐쉬에서 데이타 표시를 설정하는 단계를 포함하고 있는 판독 단계와, 매체 결함을 내포하고 있는 섹터로부터 데이타를 판독 하기 위한 시도가 이루어질 때마다 에러 플래그를 선택적으로 발생하는 단계 및, 상기 에러 플래그에 응답하여, 상기 캐쉬에서, 매체 결함을 내포하고 있는 섹터의 적어도 하나의 에러 코드를 설정하는 단계를 포함해서 이루어진 퍼스널 컴퓨터 운영방법.

청구항 2

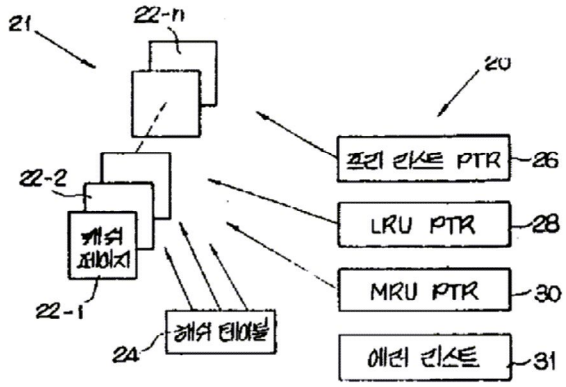
제 1 항에 있어서, 상기 캐쉬에 동시에 기억되고 있고, 적어도 하나의 에러 플래그가 발생되었던 판독과 관련된 적어도 하나의 섹터를 포함하고 있는 그런 페이지의 히스토리를 상기 캐쉬에서 작성 및 기억시키는 단계와, 상기 요청 신호에 응답하여, 요청된 섹터를 포함하는 페이지에 대해 그 안의 엔트리에 대한 상기 히스토리를 억세스 하는 단계를 포함하고 있는 퍼스널 컴퓨터 운영방법.

도면

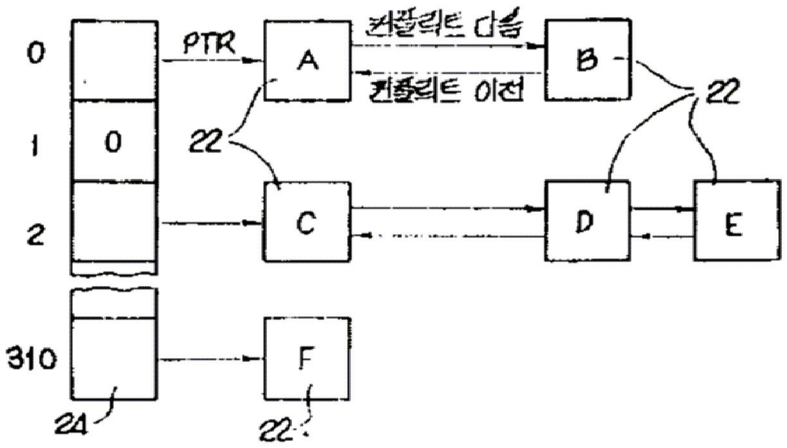
도면1


12
프로세서
15
입출력
10
16
디스크 제어기
16
디스크 제어기
20
트랙버퍼 캐쉬
21
데이타 버퍼
주 메모리
14
디스크 드라이브
디스크 드라이브
18
18

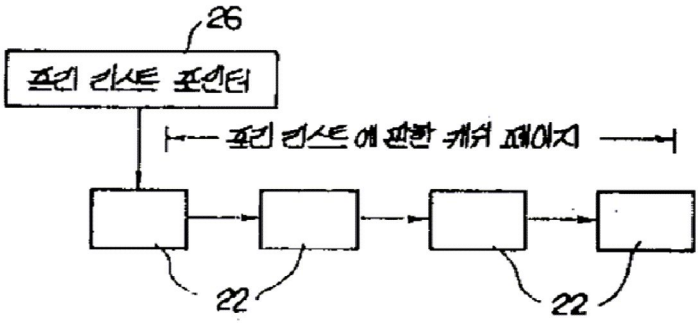
도면2


21
22-n
20
프리 리스트 PTR
26
22-2
LRU PTR
28
캐쉬 페이지
MRU PTR
30
22-1
에러 리스트
31
24
해쉬 테이블

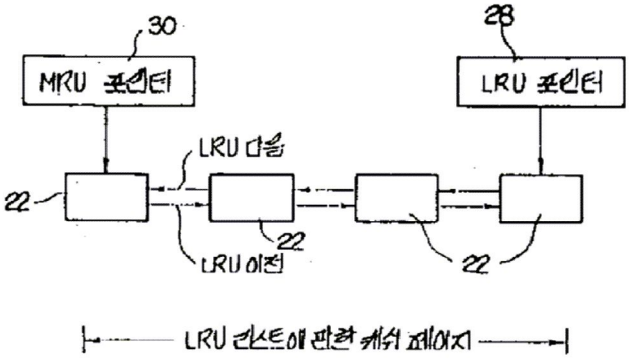
도면3


0
PTR
A
컨플릭트 다음
B
컨플릭트 이전
22
1
0
22
22
2
C
D
E
310
F
24
22

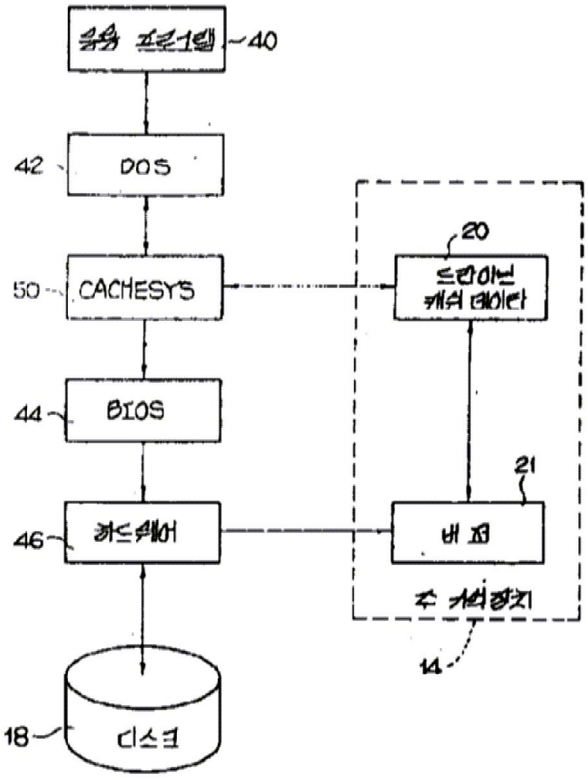
도면4


26
프리 리스트 포인터
프리 리스트에 관한 캐쉬 페이지
22
22

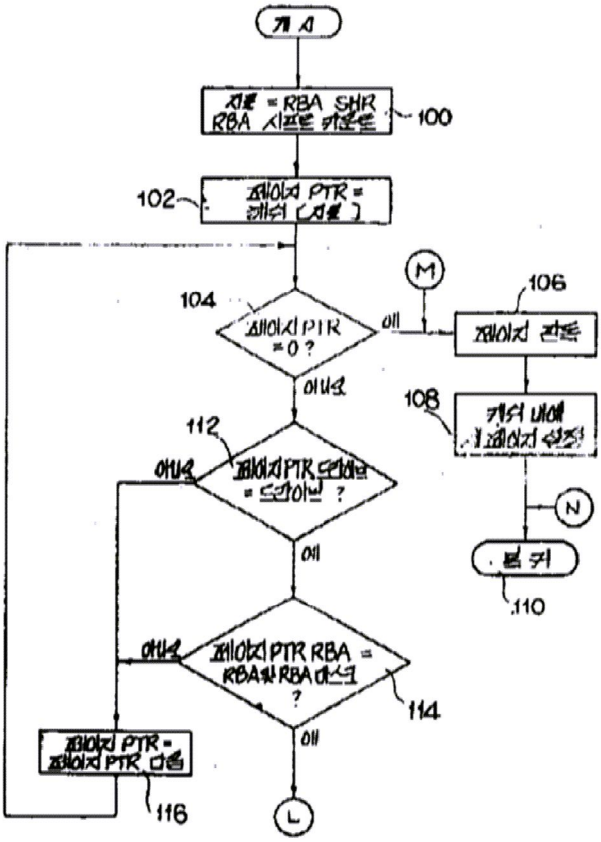
도면5


30
MRU 포인터
28
LRU 포인터
LRU 다음
22
22
LRU 이전
22
LRU 리스트에 관한 캐쉬 페이지

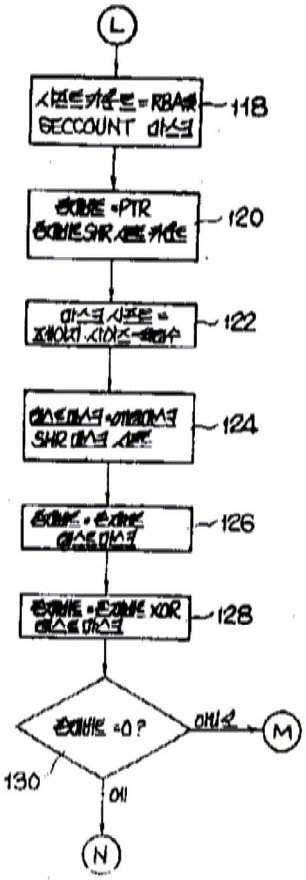
도면6


응용 프로그램
40
42
DOS
50
CACHESYS
44
BIOS
46
하드웨어
18
디스크
20
드라이브 캐쉬 데이타
21
버퍼
주 캐쉬영역
14

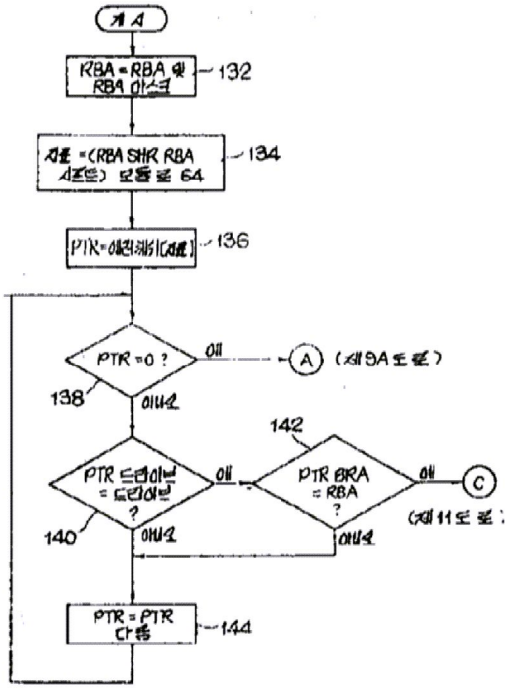
도면7a


A
100
102
104
M
106
108
112
N
110
114
116
L

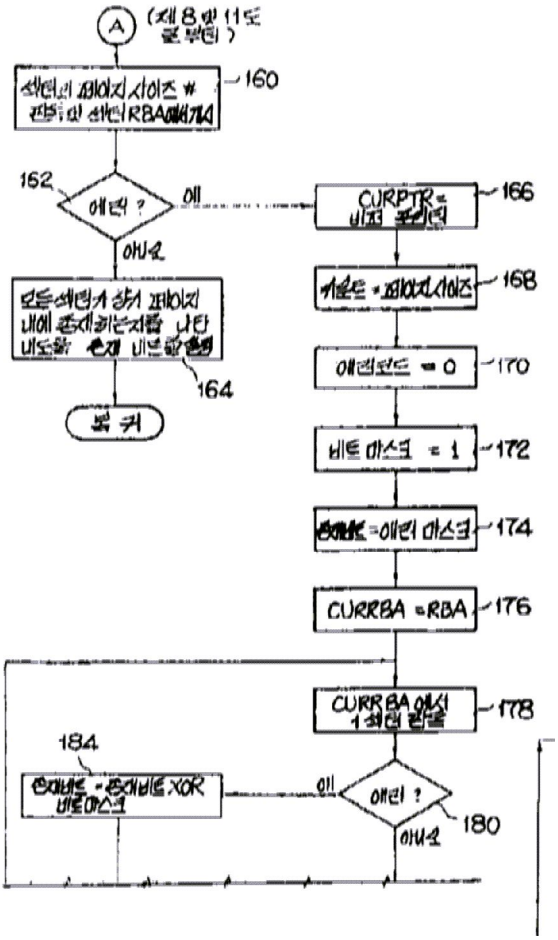
도면7b


L
118
120
122
124
126
128
130
M
N

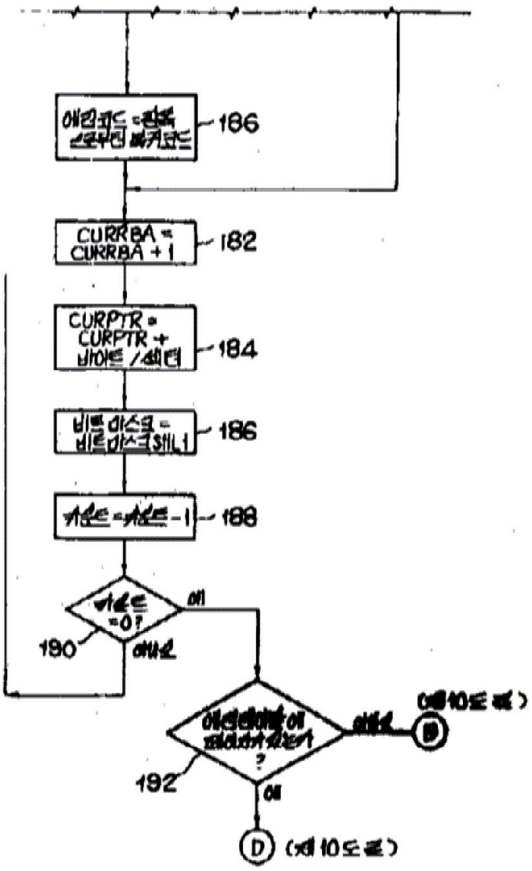
도면8


A
RBA = RBA 및 RBA 마스크
132
134
136
PTR = 0 ?
아니오
138
A (제9A도로)
142
PTR 드라이브 = 드라이브 ?
PTR BRA = RBA ?
140
C
(제11도로)
PTR = PTR 다음
144

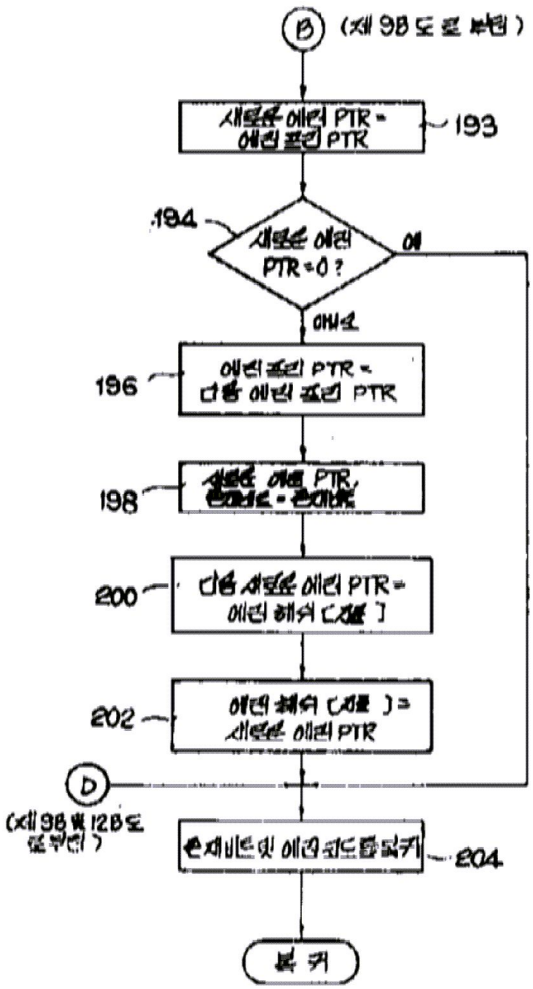
도면9a


A (제8B, 11도로부터)
160
162
에러 ?
예
아니오
CURPTR = 비지 포인터
166
168
170
172
174
CURRBA = RBA
176
CURR BA에서 1 체인 판독
178
184
에러 ?
180
164
복귀

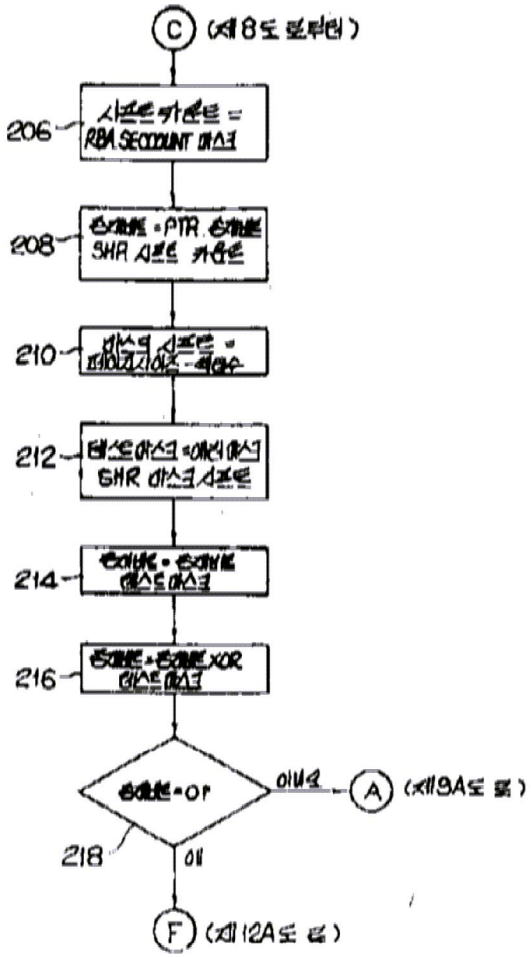
도면9b



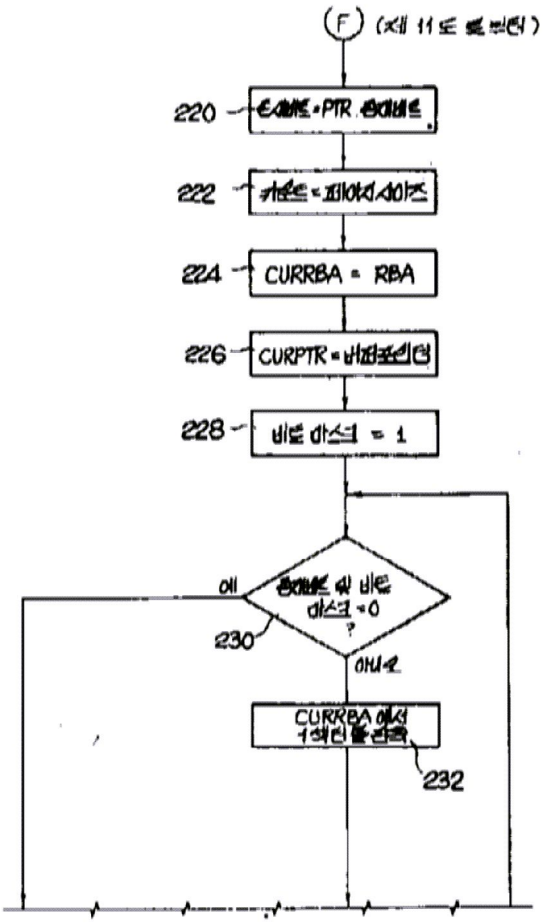
도면10



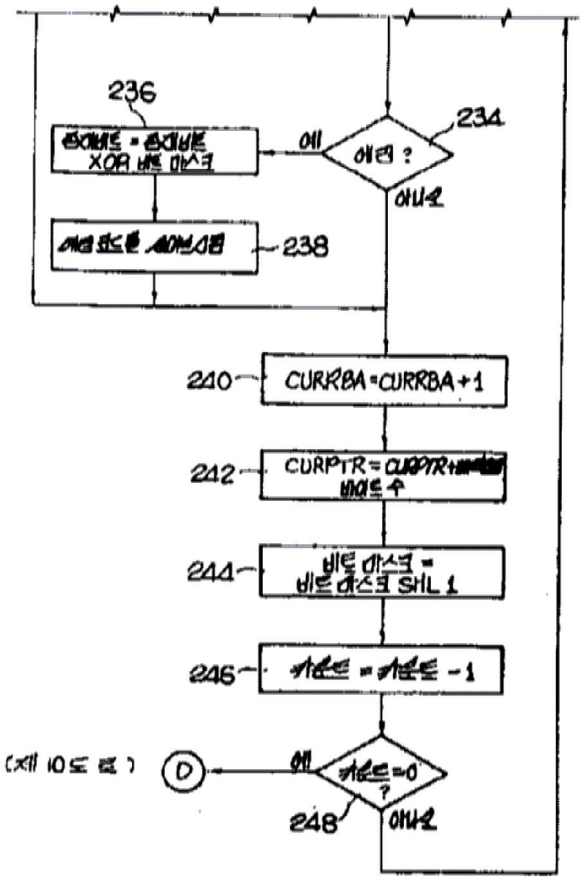
도면11



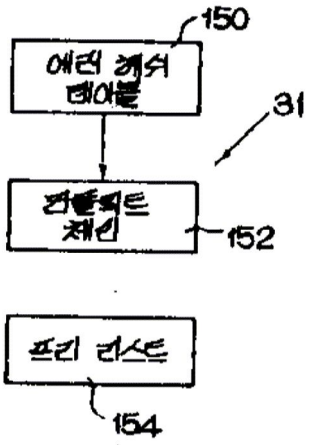
도면12a



도면12b


236
234
예
에러 ?
아니오
238
240
CURRBA=CURRBA+1
242
244
246
248
예
아니오

도면13


150
31
152
154