(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년03월31일
(11) 등록번호 10-1024819
(24) 등록일자 2011년03월17일

(51) Int. Cl.
G06F 12/02 (2006.01)
(21) 출원번호 10-2004-0031024
(22) 출원일자 2004년05월03일
심사청구일자 2009년04월02일
(65) 공개번호 10-2004-0094382
(43) 공개일자 2004년11월09일
(30) 우선권주장
60/467,343 2003년05월02일 미국(US)
10/610,666 2003년06월30일 미국(US)
(56) 선행기술조사문헌
US6442664 B1
US5604864 B1
US20020144077 A1
US20020116590 A1
전체 청구항 수 : 총 31 항

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
페이나도,마르쿠스
미국98008워싱턴주벨레뷰노쓰이스트168번애비뉴7
잉글랜드,폴
미국98008워싱턴주벨레뷰노쓰업웨이16659
(74) 대리인
주성민, 이중희, 백만기

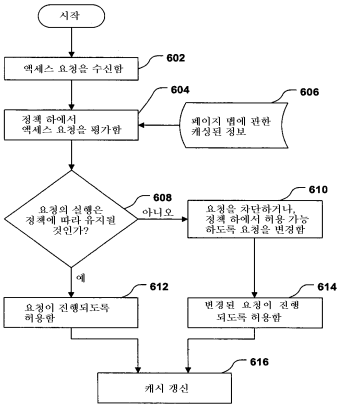
심사관 : 권오성

(54) 메모리 액세스 요청을 처리하는 방법을 수행하기 위한 컴퓨터 실행가능 명령들이 인코딩되어 있는 컴퓨터 판독가능 저장 매체, 컴퓨터 메모리 관리 방법, 및 메모리 액세스 제어 시스템

(57) 요 약

소정의 메모리 액세스 제어 알고리즘들이 효율적으로 구현될 수 있도록 해주는 메카니즘들이 개시된다. 메모리 액세스 제어가 주소 변환 맵(또는 맵들의 집합)에 대한 변화를 제어하는 것에 기초할 때, 특정한 맵 변화가 허용할 수 없는 방법으로 메모리가 액세스되는 것을 허용하는지를 판정할 필요가 있을 수 있다. 전체 맵의 평가를 수행하는 것보다 더 효율적으로 이러한 판정이 이루어질 수 있도록 해주기 위하여 맵에 관한 특정한 데이터가 캐시될 수 있다.

대 표 도 - 도6



특허청구의 범위

청구항 1

메모리 액세스 요청을 처리하는 방법을 수행하기 위한 컴퓨터 실행가능 명령들이 인코딩되어 있는 컴퓨터 판독가능 저장 매체로서, 상기 방법은,

메모리의 일부분을 액세스하기 위한 요청을 수신하는 단계 - 상기 요청은 주소 변환 맵을 통해 변환가능한 식별자를 통해 액세스될 상기 메모리의 일부분을 식별함 -;

상기 주소 변환 맵에 관한 캐싱된 정보를 기초로, 상기 요청의 실행이 상기 메모리에 대한 액세스를 제한하는 정책(policy)을 위반할지의 여부를 판정하는 단계 - 상기 캐싱된 정보는 미리 정해진 특성(a predetermined property)을 갖는 상기 주소 변환 맵의 페이지들의 집합을 식별하는 데이터를 포함함 - ;

상기 요청의 실행이 상기 정책을 위반하지 않을 것이라면, 상기 요청에 따라 상기 메모리에의 액세스를 허용하는 단계; 및

상기 요청의 실행이 상기 정책을 위반할 것이라면,

상기 요청을 차단하거나, 또는

상기 요청이 상기 정책을 위반하지 않도록 상기 요청을 변경하고 상기 변경한 요청을 수행하는 단계

를 포함하는 컴퓨터 판독가능 저장 매체.

청구항 2

제1항에 있어서,

상기 요청은 상기 메모리의 상기 일부분에 기록하기 위한 요청을 포함하는 컴퓨터 판독가능 저장 매체.

청구항 3

제1항에 있어서,

상기 주소 변환 맵은 상기 메모리에 저장되고, 상기 요청은 상기 주소 변환 맵이 저장되어 있는 메모리의 일부분에 기록하기 위한 요청을 포함하는 컴퓨터 판독가능 저장 매체.

청구항 4

제1항에 있어서,

상기 캐싱된 정보는 상기 주소 변환 맵의 루트(root)로부터 미리 정해진 거리에 위치된 상기 주소 변환 맵 내의 페이지들의 집합을 식별하는 데이터를 포함하는 컴퓨터 판독가능 저장 매체.

청구항 5

제1항에 있어서,

상기 캐싱된 정보는 특정 페이지(a specified page)에 대한 다수의 참조(reference)들을 나타내는 데이터를 포함하는 컴퓨터 판독가능 저장 매체.

청구항 6

제1항에 있어서,

상기 캐싱된 정보는 특정 페이지에 대한 다수의 참조들을 나타내는 데이터를 포함하고, 상기 참조들은 특정 속성(a specified attribute)을 갖는 컴퓨터 판독가능 저장 매체.

청구항 7

제1항에 있어서,

상기 캐싱된 정보는 상기 주소 변환 맵 내의 특정 페이지가 참조하는 다수의 페이지들을 나타내는 데이터를 포

함하는 컴퓨터 판독가능 저장 매체.

청구항 8

제1항에 있어서,

상기 캐싱된 정보는 상기 주소 변환 맵 내의 특정 페이지가 참조하고, 상기 특정 페이지가 특정 속성을 부여(assign)하는 다수의 페이지들을 나타내는 데이터를 포함하는 컴퓨터 판독가능 저장 매체.

청구항 9

제1항에 있어서,

상기 정책의 준수(compliance with said policy)는 집합 내의 페이지의 멤버쉽에 기초하여 판정되고, 상기 캐싱된 정보는 상기 집합의 진포함 집합(a proper superset of said set)을 포함하고, 상기 요청의 실행이 상기 정책을 위반할지의 여부를 판정하는 상기 단계는 상기 페이지가 상기 포함 집합의 멤버인지의 여부를 평가하는 단계를 포함하는 컴퓨터 판독가능 저장 매체.

청구항 10

제1항에 있어서,

상기 정책의 준수는 집합 내의 페이지의 멤버쉽에 기초하여 판정되고, 상기 캐싱된 정보는 상기 집합의 진부분 집합(a proper subset of said set)을 포함하고, 상기 요청의 실행이 상기 정책을 위반할지의 여부를 판정하는 상기 단계는 상기 페이지가 상기 부분 집합의 멤버인지의 여부를 평가하는 단계를 포함하는 컴퓨터 판독가능 저장 매체.

청구항 11

주소 변환 맵을 통해 액세스가 제공되는 컴퓨터 메모리를 관리하는 방법으로서,

상기 주소 변환 맵의 상태의 적어도 하나의 양상에 관한 정보를 저장하는 단계 - 저장된 정보는 미리 정해진 특성을 갖는 상기 주소 변환 맵의 페이지들의 집합을 식별하는 데이터를 포함함 - ;

상기 컴퓨터 메모리를 액세스하기 위한 요청을 수신하는 단계;

상기 저장된 정보에 적어도 부분적으로 기초하여, 상기 요청의 실행이 상기 컴퓨터 메모리에 대한 액세스를 제한하는 정책을 위반하지 않을 것이라고 판정하는 단계;

상기 요청이 실행되도록 허용하는 단계; 및

상기 요청의 실행 결과 발생하는 상기 주소 변환 맵의 상태를 반영하도록 상기 저장된 정보를 갱신하는 단계

를 포함하는 컴퓨터 메모리 관리 방법.

청구항 12

제11항에 있어서,

상기 요청은 상기 컴퓨터 메모리의 일부분에 기록하기 위한 요청을 포함하는 컴퓨터 메모리 관리 방법.

청구항 13

제11항에 있어서,

상기 주소 변환 맵은 상기 컴퓨터 메모리에 저장되고, 상기 요청은 상기 주소 변환 맵이 저장되어 있는 메모리의 일부분에 기록하기 위한 요청을 포함하는 컴퓨터 메모리 관리 방법.

청구항 14

제11항에 있어서,

상기 저장된 정보는 상기 주소 변환 맵의 루트로부터 미리 정해진 거리에 위치되어 있는 상기 주소 변환 맵 내의 페이지들의 집합을 식별하는 데이터를 포함하는 컴퓨터 메모리 관리 방법.

청구항 15

제11항에 있어서,

상기 저장된 정보는 특정 페이지에 대한 다수의 참조들을 나타내는 데이터를 포함하는 컴퓨터 메모리 관리 방법.

청구항 16

제11항에 있어서,

상기 저장된 정보는 특정 페이지에 대한 다수의 참조들을 나타내는 데이터를 포함하고, 상기 참조들은 특정 속성을 갖는 컴퓨터 메모리 관리 방법.

청구항 17

제11항에 있어서,

상기 저장된 정보는 상기 주소 변환 맵 내의 특정 페이지가 참조하는 다수의 페이지들을 나타내는 데이터를 포함하는 컴퓨터 메모리 관리 방법.

청구항 18

제11항에 있어서,

상기 저장된 정보는 상기 주소 변환 맵 내의 특정 페이지가 참조하고, 상기 특정 페이지가 특정 속성을 부여하는 다수의 페이지들을 나타내는 데이터를 포함하는 컴퓨터 메모리 관리 방법.

청구항 19

제11항에 있어서,

상기 정책의 준수는 집합 내의 페이지의 멤버쉽에 기초하여 판정되고, 상기 저장된 정보는 상기 집합의 진포함 집합을 포함하고, 상기 요청의 실행이 상기 정책을 위반하지 않을 것이라고 판정하는 상기 단계는 상기 페이지가 상기 포함 집합의 멤버인지의 여부를 평가하는 단계를 포함하는 컴퓨터 메모리 관리 방법.

청구항 20

제11항에 있어서,

상기 정책의 준수는 집합 내의 페이지의 멤버쉽에 기초하여 판정되고, 상기 저장된 정보는 상기 집합의 진부분 집합을 포함하고, 상기 요청의 실행이 상기 정책을 위반하지 않을 것이라고 판정하는 상기 단계는 상기 페이지가 상기 부분 집합의 멤버인지의 여부를 평가하는 단계를 포함하는 컴퓨터 메모리 관리 방법.

청구항 21

주소 변환 맵에 의해서 주소 지정(addressing)되는 메모리에 대한 액세스를 제어하기 위한 시스템으로서,

상기 메모리에 대한 액세스를 제한하는 정책을 저장하는 하나 이상의 저장 장치 장소들(one or more storage locations);

상기 주소 변환 맵에 관한 정보를 저장하는 캐시(cache) - 상기 캐시에 저장된 상기 정보는 미리 정해진 특성을 갖는 상기 주소 변환 맵의 페이지들의 집합을 식별하는 데이터를 포함함 - ; 및

상기 메모리를 액세스하기 위한 요청을 수신하고, 상기 캐시에 저장된 상기 정보에 적어도 부분적으로 기초하여, 상기 요청이 상기 정책 하에서 허용가능한 지의 여부를 판정하는 로직(logic)

을 포함하고,

상기 로직은 상기 요청이 상기 정책 하에서 허용가능하다고 판정되면 상기 요청이 처리되도록 허용하고,

상기 로직은 상기 요청이 상기 정책 하에서 허용가능하지 않다고 판정되면, (1) 상기 요청을 차단하거나, 또는 (2) 상기 요청을 상기 정책 하에서 허용가능한 형태로 변경하고 상기 변경된 요청이 처리되도록 허용하는 메모

리 액세스 제어 시스템.

청구항 22

제21항에 있어서,

상기 요청은 상기 메모리의 일부분에 기록하기 위한 요청을 포함하는 메모리 액세스 제어 시스템.

청구항 23

제21항에 있어서,

상기 주소 변환 맵은 상기 메모리에 저장되고, 상기 요청은 상기 주소 변환 맵이 저장되어 있는 메모리의 일부분에 기록하기 위한 요청을 포함하는 메모리 액세스 제어 시스템.

청구항 24

제21항에 있어서,

상기 캐시에 저장된 상기 정보는 상기 주소 변환 맵의 루트로부터 미리 정해진 거리에 위치되어 있는 상기 주소 변환 맵 내의 페이지들의 집합을 식별하는 데이터를 포함하는 메모리 액세스 제어 시스템.

청구항 25

제21항에 있어서,

상기 캐시에 저장된 상기 정보는 특정 페이지에 대한 다수의 참조들을 나타내는 데이터를 포함하는 메모리 액세스 제어 시스템.

청구항 26

제21항에 있어서,

상기 캐시에 저장된 상기 정보는 특정 페이지에 대한 다수의 참조들을 나타내는 데이터를 포함하고, 상기 참조들은 특정 속성을 갖는 메모리 액세스 제어 시스템.

청구항 27

제21항에 있어서,

상기 캐시에 저장된 상기 정보는 상기 주소 변환 맵 내의 특정 페이지가 참조하는 다수의 페이지들을 나타내는 데이터를 포함하는 메모리 액세스 제어 시스템.

청구항 28

제21항에 있어서,

상기 캐시에 저장된 상기 정보는, 상기 주소 변환 맵 내의 특정 페이지가 참조하고 상기 특정 페이지가 특정 속성을 부여하는 다수의 페이지들을 나타내는 데이터를 포함하는 메모리 액세스 제어 시스템.

청구항 29

제21항에 있어서,

상기 정책의 준수는 집합 내의 페이지의 멤버쉽에 기초하여 판정되고, 상기 캐시에 저장된 상기 정보는 상기 집합의 진포함 집합을 포함하고, 상기 로직은 상기 페이지가 상기 포함 집합의 멤버인지의 여부를 평가함으로써 상기 요청을 허용하는 것이 상기 정책을 위반할지의 여부를 판정하는 메모리 액세스 제어 시스템.

청구항 30

제21항에 있어서,

상기 정책의 준수는 집합 내의 페이지의 멤버쉽에 기초하여 판정되고, 상기 캐시에 저장된 상기 정보는 상기 집합의 진부분 집합을 포함하고, 상기 로직은 상기 페이지가 상기 부분 집합의 멤버인지의 여부를 평가함으로써

상기 요청을 허용하는 것이 상기 정책을 위반할지의 여부를 판정하는 메모리 액세스 제어 시스템.

청구항 31

제21항에 있어서,

상기 로직은 하드웨어와 소프트웨어 중 적어도 하나로 구현되는 메모리 액세스 제어 시스템.

청구항 32

삭제

청구항 33

삭제

청구항 34

삭제

청구항 35

삭제

청구항 36

삭제

청구항 37

삭제

청구항 38

삭제

청구항 39

삭제

청구항 40

삭제

청구항 41

삭제

청구항 42

삭제

청구항 43

삭제

청구항 44

삭제

청구항 45

삭제

명 세 서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

[0012] 본 출원은 2003년 5월 2일자로 출원된 "Techniques for Efficient Implementation of Memory Access Control"이란 명칭의 미국 가출원 60/467,343의 권리를 주장한다.

[0013] 본 발명은 일반적으로 컴퓨터 보안 분야에 관한 것이다. 보다 구체적으로는, 본 발명은 주소 변환 제어(address translation control)를 이용하여 분리형(isolated) 또는 "커튼형(curtained)" 메모리를 구현하기 위한 효율적인 기술에 관한 것이다.

[0014] 몇몇 환경에서는, 액세스가 제한된 분리 또는 "커튼된" 메모리 부분을 갖는 것이 바람직하다. 예를 들면, 하나의 컴퓨터는 서로 협력하는 두 개의 운영 체제를 가동시킬 수도 있는데, 여기서 하나의 운영 체제는 보안 상태에 있으며 나머지 하나는 보안 상태에 있지 않다. 이 경우, 보안되지 않은 운영 체제에 의해 액세스될 수 없는 비밀 정보를 저장할 수 있는 커튼형 메모리를 보안된 운영 체제가 가지는 것이 바람직하다.

[0015] 커튼형 메모리를 구현하는 방법중 하나는 주소 변환 제어를 통하는 것이다. 현재의 많은 컴퓨터들에서는 가상 메모리 시스템이 이용되는데, 이 가상 메모리 시스템에서는, 컴퓨터 상에서 가동되는 소프트웨어가 가상 주소를 이용하여 메모리의 주소를 지정하고 메모리 관리 유닛이 주소 변환 맵 세트를 이용하여 가상 주소를 물리적 주소로 변환한다. 전형적으로, 각 프로세스는 자기 자신의 주소 변환 맵을 가져서, 가상 주소 및 물리적 주소 간의 맵핑이 프로세스들 간에 변경되도록 한다. 소정의 프로세스의 주소 변환 맵을, 프로세스의 맵이 물리적 메모리의 소정의 블럭(예를 들면, 페이지)에 대한 임의의 가상 주소를 상기 프로세스에 노출시키지 않도록 구성하는 것이 가능하다. 이에 따라, 보안된 프로세스만이 물리적 메모리의 소정의 블럭에 대한 가상 주소를 가지도록 함으로써, 주소 변환 맵의 컨텐츠를 제어하여 커튼형 메모리를 구현하는 것이 가능하다.

[0016] 커튼형 메모리를 구현하는데에 이러한 메카니즘이 이용될 때 발생하는 하나의 문제점은, 주소 변환 맵이 메모리에 저장됨으로 인해, 메모리에 기입하는 모든 동작이 잠재적으로 맵들에 영향을 미칠 수 있어서 커튼형 메모리에 대한 가상 주소가 커튼형 메모리에 액세스되지 않아야 할 프로세스에 노출될 수도 있다는 것이다. 이러한 가상 주소가 노출되는 것을 방지하기 위한 방법중 하나는, 커튼형 메모리의 어떠한 페이지도, 커튼형 메모리에 액세스되지 않아야 하는 임의의 프로세스의 맵 내의 가상 주소를 가지지 않도록 하기 위해 메모리 상의 기록 동작이 수행될 때마다 모든 맵의 모든 엘리먼트를 체크하는 것이다. 그러나, 기록 동작이 자주 발생되는 경우, 이 기술은 효율적이지 않다.

[0017] 전술한 바로부터, 종래 기술의 문제점을 극복하기 위한 메카니즘이 필요하게 된다.

발명이 이루고자 하는 기술적 과제

[0018] 본 발명은 주소 변환 맵에 대한 변경을 효율적으로 제어하기 위한 메카니즘을 제공한다. 커튼형 메모리는, 커튼형 메모리의 하나의 블럭에 대한 하나의 가상 주소가 커튼형 메모리에 액세스하도록 허용되지 않은 프로세스(또는 그 밖의 다른 엔티티(entity))에 노출될 수도 있는 상태에 주소 변환 맵이 진입하지 못하도록 함으로써 구현될 수 있다. "정책(policy)"은, 어떤 메모리 액세스 동작이 허용되는지를 정의하며, 메모리 액세스 제어 시스템은 주소 변환 맵이 이 정책을 위반하는 임의의 상태에 진입하지 못하도록 동작할 수 있다.

[0019] 이러한 가상 주소가 노출될 수도 있는 상태는 종종 소정의 속성을 만족시키는 두 개 이상의 집합의 교집합(intersection)(또는 여집합(non-intersection)), 또는 소정의 속성을 만족시키는 페이지들의 수에 기초하여 정의될 수 있다. 정의된 집합의 구성요소들인 페이지들의 아이덴티티가 저장되거나 캐싱되어, 주소 변환 맵의 상태를 변경시킬 수도 있는 기록 동작이 수행될 때마다 이 집합의 멤버십이 계산될 필요가 없게 된다. 하나의 집합 내의 페이지들의 아이덴티티는, 예를 들어 비트 벡터로서 저장될 수 있으며, 합집합(union), 교집합 등과 같은 집합 연산은 이러한 비트 벡터에 대해 효율적으로 수행될 수 있다. 몇몇 경우에, 특정 속성을 만족시키는 정확한 집합이 계산되기 어려울 수도 있지만, 이 정책에 따르는 것은 소정의 잘 정의된 부분 집합 또는 포함 집합(superset)을 실질적인 집합에 대한 프록시로서 이용함으로써 보장될 수 있음이 수학적으로 증명가능할 수도 있다. 부분 집합 또는 포함 집합이 실제의 집합보다 비교적 계산하기 쉬울 경우, 부분 집합 또

는 포함 집합이 실제의 집합 대신에 이용될 수도 있다.

[0020] 또한, 소정의 기록 동작의 허용도가, 소정의 통계치의 카운트, 예를 들면, 소정의 속성을 만족시키는 페이지들의 수, 소정의 페이지에 대한 참조(reference)의 수 등의 면에서 정의될 수 있다. 이러한 통계치는 참조 카운터로서 효율적으로 저장되거나 캐싱될 수 있으며, 이 참조 카운터는 증분 및 감분 연산을 통해 갱신될 수 있다. 비트 벡터 또는 카운터는, 맵이 상태를 변경할 때마다 갱신될 수 있으며, 그 후 상기 정책 하에서 메모리 액세스 동작을 평가하는데에 효율적으로 이용된다.

[0021] 본 발명의 그 밖의 다른 특징들은 이하에 기술된다.

발명의 구성 및 작용

[0022] 이하의 바람직한 실시예의 상세한 설명과 전술한 발명의 요약은 첨부된 도면과 결부시켜 읽으면 더욱 잘 이해될 것이다. 본 발명을 예시하기 위해 도면에는 본 발명의 예시적인 구조가 도시되어 있지만, 본 발명은 개시된 특정 방법 및 수단으로 제한되는 것은 아니다.

[0023] 예시적인 컴퓨팅 구성

[0024] 도 1은 본 발명의 특징이 구현될 수 있는 컴퓨팅 환경의 예를 나타낸다. 이 컴퓨팅 시스템 환경(100)은 단지 적절한 컴퓨팅 환경의 일례이며 본 발명의 사용 또는 기능의 범위를 제한하기 위하여 의도된 것이 아니다. 컴퓨팅 환경(100)은 이 예시적인 오퍼레이팅 환경(100)에 도시된 컴포넌트들 중의 임의의 하나 또는 이들의 조합에 관한 임의의 종속성 또는 필요조건을 갖는 것으로 해석되어서는 안된다.

[0025] 본 발명은 많은 다른 범용 또는 특수 목적 컴퓨팅 시스템 환경 또는 구성과 함께 동작된다. 본 발명과 함께 사용하기에 적합할 수 있는 잘 알려진 컴퓨팅 시스템, 환경, 및/또는 구성의 예는 퍼스널 컴퓨터, 서버 컴퓨터, 핸드헬드(hand-held) 또는 랩탑 장치, 멀티프로세서 시스템, 마이크로프로세서 기반 시스템, 셋 탑 박스(set top box), 프로그래머블 고객 전자장치, 네트워크 PC, 미니컴퓨터, 메인프레임 컴퓨터, 내장 시스템, 상기의 시스템 또는 장치 중의 임의의 것을 포함하는 분산 컴퓨팅 환경을 포함할 수 있지만, 이에 한정되는 것은 아니다.

[0026] 본 발명은 컴퓨터에 의해 실행되는 프로그램 모듈 등의 컴퓨터 실행가능 명령의 일반적인 컨텍스트에서 기술될 것이다. 일반적으로, 프로그램 모듈은 특정 태스크를 수행하거나 특정 추상 데이터 유형을 구현하는 루틴, 프로그램, 오브젝트, 컴포넌트, 데이터 구조 등을 포함한다. 본 발명은 또한 통신 네트워크 또는 다른 데이터 전송 매체를 통해 링크된 원격 프로세싱 장치에 의해 태스크를 수행하는 분산 컴퓨팅 환경에서 실행될 수 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈과 다른 데이터는 메모리 저장 장치를 포함하는 국부 및 원격 컴퓨터 저장 매체 내에 위치할 수 있다.

[0027] 도 1을 참조하면, 본 발명을 구현하는 예시적인 시스템은 컴퓨터(110)의 형태의 범용 컴퓨팅 장치를 포함한다. 컴퓨터(110)의 컴포넌트는 프로세싱 유닛(120), 시스템 메모리(130), 및 시스템 메모리를 포함하는 다양한 시스템 컴포넌트를 프로세싱 유닛(120)에 결합하는 시스템 버스(121)를 포함할 수 있지만, 이에 한정되는 것은 아니다. 시스템 버스(121)는 다양한 버스 아키텍처 중의 임의의 것을 사용하는 국부적 버스, 주변 버스, 및 메모리 버스 또는 메모리 컨트롤러를 포함하는 몇가지 유형의 버스 구조 중의 임의의 것일 수 있다. 예로서, 이러한 아키텍처는 산업 표준 아키텍처(ISA) 버스, 마이크로 채널 아키텍처(MCA) 버스, 인핸스드 ISA(Enhanced ISA; EISA) 버스, 비디오 일렉트로닉스 표준 어소시에이션(VESA) 국부적 버스, 및 주변 컴포넌트 상호접속(PCI) 버스(또한 메자닌 버스(Mezzanine bus)로 공지됨)를 포함하지만, 이에 한정되는 것은 아니다.

[0028] 컴퓨터(110)는 일반적으로 다양한 컴퓨터 판독가능 매체를 포함한다. 컴퓨터 판독가능 매체는 컴퓨터(110)에 의해 액세스될 수 있는 임의의 이용가능 매체일 수 있으며, 휘발성 및 불휘발성 매체, 분리형 및 비분리형 매체 양쪽 모두를 포함한다. 예로서, 컴퓨터 판독가능 매체는 컴퓨터 저장 매체 및 통신 매체를 포함할 수 있지만, 이에 한정되는 것은 아니다. 컴퓨터 저장 매체는 컴퓨터 판독가능 명령, 데이터 구조, 프로그램 모듈 또는 다른 데이터 등의 정보의 저장을 위한 임의의 방법 또는 기술로 구현되는 휘발성 및 불휘발성, 분리형 및 비분리형 매체를 둘다 포함한다. 컴퓨터 저장 매체는 RAM, ROM, EEPROM, 플래쉬 메모리 또는 다른 메모리 기술, CD-ROM, DVD(digital versatile disk) 또는 다른 광학 디스크 저장 장치, 자기 카세트, 자기 테이프, 자기 디스크 저장 장치 또는 다른 자기 저장 장치, 또는 컴퓨터(110)에 의해 액세스될 수 있고 원하는 정보를 저장하는 데 사용될 수 있는 임의의 다른 매체를 포함할 수 있지만, 이에 한정되는 것은 아니다. 통신 매체는 일반적으로 컴퓨터 판독가능 명령, 데이터 구조, 프로그램 모듈, 또는 반송파 또는 전송 메카니즘 등의 변조 데이터 신호의 다른 데이터를 구현하며, 임의의 정보 전달 매체를 포함한다. 용어 "변조 데이터 신호"는 정보를 신호로 인코

딩하는 것과 같은 방법으로 설정되고 변경된 그 특성 중의 하나 이상을 갖는 신호를 의미한다. 예로서, 통신 매체는 유선 네트워크 또는 직접 유선 네트워크 등의 유선 매체, 및 음향, RF, 적외선 및 다른 무선 매체 등의 무선 매체를 포함하지만, 이에 한정되지 않는다. 상술한 것들 중의 임의의 조합이 컴퓨터 판독가능 매체의 범위내에 포함되어야 한다.

[0029] 시스템 메모리(130)는 ROM(131) 및 RAM(132) 등의 휘발성 및/또는 불휘발성 메모리의 형태의 컴퓨터 저장 매체를 포함한다. 시동시 등의 컴퓨터(110)내의 소자들간에 정보를 전송하는 것을 돕는 기본 루틴을 포함하는 기본 입출력 시스템(133; BIOS)은 일반적으로 ROM(131)에 저장된다. RAM(132)은 일반적으로 프로세싱 유닛(120)에 즉시 액세스될 수 있고 및/또는 프로세싱 유닛(120)에 의해 즉시 동작하는 프로그램 모듈 및/또는 데이터를 포함한다. 예로서, 이에 한정되지 않지만, 도 1은 오퍼레이팅 시스템(134), 애플리케이션 프로그램(135), 다른 프로그램 모듈(136), 및 프로그램 데이터(137)를 도시한다.

[0030] 컴퓨터(110)는 또한 다른 분리형/비분리형, 휘발성/불휘발성 컴퓨터 저장 매체를 포함할 수 있다. 단지 예로서, 도 1은 비분리형 불휘발성 자기 매체로부터 판독하거나 그 자기 매체로 기입하는 하드 디스크 드라이브(140), 분리형 불휘발성 자기 디스크(152)로부터 판독하거나 그 자기 디스크로 기입하는 자기 디스크 드라이브(151), 및 CD-ROM 또는 다른 광학 매체 등의 분리형 비휘발성 광학 디스크(156)로부터 판독하거나 그 광학 디스크로 기입하는 광학 디스크 드라이브(155)를 도시한다. 예시적인 오퍼레이팅 환경에서 사용될 수 있는 다른 분리형/비분리형, 휘발성/불휘발성 컴퓨터 저장 매체는 자기 테이프 카세트, 플래쉬 메모리 카드, DVD(Digital versatile disk), 디지털 비디오 테이프, 고체 상태 RAM, 고체 상태 ROM 등을 포함하지만 이에 한정되지 않는다. 하드 디스크 드라이브(141)는 일반적으로 인터페이스(140) 등의 비분리형 메모리 인터페이스를 통해 시스템 버스(121)에 접속되고, 자기 디스크 드라이브(151) 및 광학 디스크 드라이브(155)는 일반적으로 인터페이스(150) 등의 분리형 인터페이스에 의해 시스템 버스(121)에 접속된다.

[0031] 상술하고 도 1에 도시된 드라이브 및 그 관련 컴퓨터 저장 매체는 컴퓨터(110)를 위한 컴퓨터 판독가능 명령, 데이터 구조, 프로그램 모듈 및 다른 데이터의 저장을 제공한다. 도 1에서, 예를 들어, 하드 디스크 드라이브(141)는 운영 체제(144), 애플리케이션 프로그램(145), 다른 프로그램 모듈(146), 및 프로그램 데이터(147)를 저장하는 것으로 도시된다. 이들 컴포넌트는 운영 체제(134), 애플리케이션 프로그램(135), 다른 프로그램 모듈(136), 및 프로그램 데이터(137)와 동일할 수도 있고 다를 수도 있다. 운영 체제(144), 애플리케이션 프로그램(145), 다른 프로그램 모듈(146), 및 프로그램 데이터(147)는 최소한 다른 동등물임을 나타내기 위하여 다른 번호를 부여하였다. 사용자는 키보드(162) 및 일반적으로 마우스, 트랙볼, 또는 터치 패드라 불리우는 포인팅 장치(161) 등의 입력 장치를 통해 컴퓨터(20)에 명령 및 정보를 입력할 수 있다. 다른 입력 장치(도시하지 않음)는 마이크로폰, 조이스틱, 게임 패드, 위성 안테나, 스캐너 등을 포함할 수 있다. 이들 및 다른 입력 장치는 종종 시스템 버스에 결합된 사용자 입력 인터페이스(160)를 통해 프로세싱 유닛(120)에 접속되지만, 병렬 포트, 게임 포트 또는 유니버설 시리얼 포트(USB) 등의 다른 인터페이스 및 버스 구조에 의해 접속될 수 있다. 모니터(191) 또는 다른 유형의 디스플레이 장치는 또한 비디오 인터페이스(190) 등의 인터페이스를 통해 시스템 버스(121)에 접속된다. 모니터 외에도, 컴퓨터는 또한 출력 주변 인터페이스(190)를 통해 접속될 수 있는 스피커(197) 및 프린터(196) 등의 다른 주변 출력 장치를 포함할 수 있다.

[0032] 컴퓨터(110)는 원격 컴퓨터(180) 등의 하나 이상의 원격 컴퓨터로의 논리적 접속을 이용한 네트워크 환경에서 동작할 수 있다. 원격 컴퓨터(180)는 퍼스널 컴퓨터, 서버, 라우터, 네트워크 PC, 피어(peer) 장치, 또는 다른 공통 네트워크 노드일 수 있으며, 도 1 에는 메모리 저장 장치(181)만이 도시되어 있지만, 일반적으로 컴퓨터(110)에 관하여 상술한 많은 소자 또는 모든 소자를 포함할 수 있다. 도 1에 도시된 논리적 접속은 근거리 통신망(LAN; 171) 및 원거리 통신망(WAN; 173)을 포함하지만, 다른 네트워크를 포함할 수도 있다. 이러한 네트워크 환경은 사무실, 기업 광역 컴퓨터 네트워크, 인트라넷, 및 인터넷에서 공통적이다.

[0033] LAN 네트워크 환경에서 사용될 때, 컴퓨터(110)는 네트워크 인터페이스 또는 어댑터(170)를 통해 LAN(171)에 접속된다. WAN 네트워크 환경에서 사용될 때, 컴퓨터(110)는 일반적으로 인터넷 등의 WAN(173)을 통해 통신을 구축하는 모뎀(172) 또는 다른 수단을 포함한다. 내장형 또는 외장형일 수 있는 모뎀(172)은 사용자 입력 인터페이스(160) 또는 다른 적절한 메카니즘을 통해 시스템 버스(121)에 접속될 수 있다. 네트워크 환경에서, 컴퓨터(110)에 관하여 도시된 프로그램 모듈 또는 그 부분은 원격 메모리 저장 장치에 저장될 수 있다. 예로서, 한정되지는 않지만, 도 1은 메모리 장치(181)에 상주하는 원격 애플리케이션 프로그램(185)을 도시한다. 도시된 네트워크 접속은 예시적인 것이며, 컴퓨터들간의 통신 링크를 구축하는 다른 수단이 사용될 수 있다.

[0034] 주소 변환을 이용한 메모리 액세스

[0035] 컴퓨터 시스템 내의 메모리(예를 들면, 도 1에 도시된 RAM(132))는 각 바이트마다 물리적 주소를 갖는다. 이에 따라, 메모리를 구성하는 바이트들은 번호가 매겨진 것으로 보일 수 있으며 여기서 각 바이트는 자신의 번호에 의해 명백하게 식별될 수 있다. 이 경우, 이 번호는 물리적 주소를 형성한다. 예를 들면, 256 바이트 메모리에서, 이 바이트들은 0에서 2^{28}-1 까지의 물리적 주소를 가질 수 있다. 그러나, 현대의 컴퓨터 시스템에서, 메모리는 일반적으로 자신의 물리적 주소에 의해 액세스되지 않으며, 오히려 가상 주소에 의해 액세스된다. 물리적 주소를 가상 주소로 변환하는데에 주소 변환 맵이 이용된다.

[0036] 도 2는 주소 변환 맵과, 실제의 컴퓨터 시스템에서의 사용의 예를 나타낸 도면이다. 도 2에 도시된 이 예시적인 주소 변환 맵은 "페이징(paging)" 구조이며, 여기서 메모리는 "페이지"라 불리는 블럭들에 할당되어 있다. 도 2는 INTEL x86 프로세서 상에서 이용되는 페이징 구조를 나타낸다.

[0037] 도 2에서, 페이지 디렉토리(202)는 페이지 테이블 204(1), 204(2), 204(3)과 같은 페이지 테이블들(즉, 페이지 테이블들의 물리적 베이스 주소)들에 대한 포인터들의 어레이를 포함한다. 그리고, 각 페이지 테이블은 페이지들(예를 들면, 페이지 206(1), 206(2), 206(3), 206(4))의 베이스 주소들에 대한 포인터들의 어레이를 포함하고 또한 전술한 바와 같은, 판독 전용/판독-기록 속성과 같은 정보, 존재/존재하지 않음(present/non-present) 비트 등을 포함할 수도 있다. 페이지들은 RAM(132)에서 고정된 길이를 갖는 부분들이다. 또한, 전형적으로 페이지 디렉토리 및 페이지 테이블들도 RAM(132)에 저장된다. 특정 페이지의 위치 결정을 위해 페이지 디렉토리(레벨 1) 및 페이지 테이블(레벨 2) 양쪽 모두를 거칠 필요가 있기 때문에 도 2에 도시된 페이징 구조는 2 레벨 페이징 구조이다. 본 기술 분야에 통상의 지식을 가진 자라면, 임의의 수의 레벨을 갖는 페이징 구조를 설계하는 것이 가능하며 본 발명은 그러한 모든 페이징 구조에 적용될 수 있음을 알 것이다. 또한, INTEL x86 프로세서는 전형적으로 도 2에 도시된 2 레벨 페이징 구조를 사용하지만 1 레벨 또는 3 레벨 페이징 구조를 이용하도록 구성될 수도 있음이 본 기술 분야에 공지되어 있다.

[0038] 도 2의 페이징 구조에서, 하나의 페이지 상의 임의의 바이트는, 페이지 디렉토리 오프셋(211), 페이지 테이블 오프셋(212), 및 페이지 오프셋(213)을 포함하는 가상 주소(210)에 의해 식별될 수 있다. 이에 따라, 물리적 주소의 위치 결정을 위해, 메모리 관리 유닛(MMU)(220)은 페이지 디렉토리 오프셋(211)을 이용하여 페이지 디렉토리(202) 내에 특정 엔트리의 위치를 결정한다. 이 엔트리는 페이지 테이블의 물리적 베이스 주소이며, 이에 따라 MMU(220)는 페이지 테이블중 하나(예를 들면, 페이지 테이블 204(1))의 위치 결정을 위해 이 주소를 디레퍼런스(dereference)한다. 그 후, MMU(220)는 페이지 테이블 오프셋(212)을 식별된 페이지 테이블로의 인덱스로서 사용하여 그 오프셋에서 발견되는 엔트리를 검색한다. 이 엔트리는 페이지(예를 들면, 페이지 206(1))의 물리적 베이스 주소가며, 이에 따라 MMU는 페이지 오프셋(213)을 식별된 페이지의 베이스 주소에 추가하여 물리적 메모리의 특정 바이트의 위치 결정을 행한다. MMU(220)는 또한, 도 3과 관련하여 이하에 설명하는 바와 같이, 페이지가 판독 전용 또는 판독-기록으로 표시되어 있는지 여부와, 그 페이지가 존재하거나 존재하지 않는 것으로 표시되어 있는지 여부 등과 같은 정보를 고려하도록 구성될 수도 있다.

[0039] 도 2의 페이징 구조는 또한 페이지 디렉토리로의 포인터를 포함하는 저장 장소(201)를 포함한다. MMU(220)는, 가상 주소(210)를 변환하기 시작할 때 이 포인터를 이용하여 페이지 디렉토리(202)의 위치를 결정한다. INTEL x86 프로세서의 예에서, 저장 장소(201)는 CR3로 명칭된 레지스터에 대응하는데, 즉 INTEL x86 프로세서 상에서, 레지스터 CR3는 현재의 컨텍스트에 대한 페이지 디렉토리의 물리적 주소를 저장한다. 이에 따라, 다른 변환 테이블 세트(즉, 페이지 디렉토리 및 페이지 테이블의 두 개 이상의 세트)를 구축하고, 새로운 페이지 디렉토리의 베이스 주소를 저장 장소(201)에 단순히 기입하는 것으로도 어떤 변환 테이블의 세트가 적용되는지를 변경하는 것이 가능하다. 이 기술의 통상적인 사용 방법중 하나는, 컴퓨터 상에서 동작하는 각 프로세스에 대해 자기 자신의 페이지 디렉토리 및 페이지 테이블들을 갖게 하는 것인데, 여기서 "컨텍스트 스위치"(즉, 무엇보다도 가상 메모리 시스템이 새로운 프로세스의 주소 스페이스에 대해 포인트하도록 하는 동작)가, 새로운 프로세스의 페이지 디렉토리의 베이스 주소를 저장 장소(201)에 기입함으로써 수행된다. 각 프로세스가 자기 자신의 페이지 디렉토리를 갖는 경우, 현재 가동중인 프로세스의 아이덴티티는 어떤 값이 저장 장소(201)에 로드되는지를 결정한다.

[0040] 페이지들에 대한 포인터를 포함하는 것 외에도, 페이지 테이블들 및 페이지 디렉토리들은 또한 그 페이지들에 대한 "속성"을 포함할 수도 있다. 도 3은, 포인터 및 속성 양쪽 모두를 포함하는 예시적인 페이지 테이블 204(1)을 상세하게 나타낸 도면이다. 페이지 테이블 204(1) 내의 각 엔트리는 특정 페이지의 주소(302)와, 엔트리에 의해 포인트되는 페이지가 "판독 전용"인지 여부를 나타내는 비트(304)와, 엔트리에 의해 포인트되는 페이지가 "존재"하는지 여부를 나타내는 비트(306)를 포함한다. 이에 따라, 페이지 테이블 204(1) 내의 제1 엔트

리(301)가 페이지 206(1)(도 2에 도시됨)에 대해 포인트할 경우, 비트(304)는, 0 또는 1로 세트되어 있는지 여부에 따라 MMU(220)(도 2에 도시됨)가 페이지 206(1)가 판독 및 기록 양쪽, 혹은 판독만 되도록 허용하는지 여부를 나타낸다. 마찬가지로, 비트(306)는, 페이지 206(1)가 메모리 내에 존재하는지 여부를 나타낸다. (예를 들어 메모리 내의 다른 페이지들에 대한 공간 확보를 위해 페이지 206(1)의 컨텐츠가 디스크로 이동된 경우 비트(306)는 0으로 세트되어서 페이지 206(1)가 존재하지 않음을 나타낼 수도 있다.) 그 밖의 속성들도 또한 페이지 테이블 204(1) 내에 저장될 수 있다.

[0041] 메모리 액세스 제어를 위한 주소 변환 맵의 이용

[0042] 메모리가 가상 주소에 의해 액세스되는 시스템에서, 가상 주소가 소정의 물리적 주소로 변환되지 않도록 주소 변환 맵이 구성되는 경우, 그 물리적 주소에 의해 표시되는 메모리는 액세스가능하지 않다는 경험적 지식(observation)에 기초하여 메모리에 대한 액세스를 제한하는 시스템을 구현할 수 있다. 예를 들면, 도 2와 관련하여 전술한 페이징 구조에서, 메모리의 소정의 페이지(예를 들면, 페이지 206(1))는, 맵을 통해 그 페이지에 도달하는 경로가 없도록 함으로써 액세스되지 않도록 될 수 있다. 이러한 경로가 없을 때, 그 페이지로 변환될 수 있는 가상 주소(210)가 없을 것이다. 모든 메모리 액세스가 가상 주소에 의해 행해지는 시스템에서, 메모리의 소정의 페이지(또는 그 밖의 부분)로의 가상 주소를 부정하도록 주소 변환 맵에 대해 제어함으로써 효율적으로 메모리의 그 부분이 액세스되지 않도록 된다. 메모리의 소정의 물리적 주소 지정을 허용하는 시스템에서도, 메모리는, 물리적 주소에 기초한 액세스 요구에 대한 제어로 주소 변환 맵에 대한 제어를 보충함으로써 액세스 가능하지 않게 될 수 있다.

[0043] 메모리에 대한 액세스를 제어하기 위해 주소 변환 맵의 컨텐츠를 제어하는 기술은 이하와 같이 공식으로 기술될 수 있다. 여기서, S는 잠재적으로 메모리에 액세스할 수 있는 소스의 집합인 것으로 가정한다. 또한 P는 메모리의 어떤 부분이 어떤 소스에 의해 액세스될 수 있는지를 정의하는 정책인 것으로 가정한다. 이에 따라, $s \in S$가 소스인 경우, MP(s)는 주소 변환 맵을 통해 소스 s에 액세스할 수 있는 메모리 부분(예를 들면, 가상 주소를 갖는 메모리 장소의 집합)을 나타내며, NA(P, s)는, 소스 s가 정책 P 하에서 액세스가 허용되지 않는 메모리의 부분들을 나타낸다. (각 프로세스가 자기 자신의 주소 변환 맵을 갖는 경우, 소스의 개념이 프로세스의 예보다 일반화된 것으로 이해될지라도, 각 프로세스는 서로 다른 "소스"로서 보여질 수 있다.) 이에 따라, 정책의 규약은 $NA(P,s) \cap MP(s) = \Phi$라는 조건이 만족되는 한 보장될 수 있다. 이 조건은 도 4에 도시되며, 여기서 메모리 장소의 집합으로서의 메모리(132)와, 주소 변환 맵핑을 통해 소스에 보여질 수 있는 메모리 장소의 집합으로서의 MP(s)와, 소스 s가 정책 P 하에서 액세스되지 않는 메모리 장소의 집합으로서의 NA(P,s)가 도시된다. 소스 s가 주소 변환 맵핑을 통해 주소 지정할 수 있는 장소(MP(s))중 어느 것도, 소스 s가 정책 P 하에서 액세스되는 것이 허용되지 않는 메모리 장소의 집합에 포함되지 않기 때문에, 도 4에 도시된 조건은 소스 s에 따라 정책 P를 효율적으로 규약한다.

[0044] 이에 따라, 메모리(132)의 부분들에 대한 소스 s의 액세스를 제어하는 문제는 몇몇 예시적인 상황에서 감소되어 도 4에 도시된 조건이 항상 진실(true)인 것을 보장하게 된다. 이 문제를 해결하기 위한 하나의 해결책은, 주소 변환 맵핑, 정책, 또는 현재의 소스를 변경할 가능성을 갖는 임의의 동작(예를 들면, 메모리 기록, CR3 레지스터의 로드 등)을 평가하는 것이다. 본 발명은 이러한 평가가 효율적으로 되도록 해주는 기술을 제공한다.

[0045] 도 4에 도시된 조건은 메모리 액세스 제어를 구현하는 데에 이용될 수 있는 하나의 조건의 예에 지나지 않음을 알 것이다. 도 4의 테마에 대한 그 밖의 다른 변수, 예를 들면, 주소 변환 맵에 포함되는 메모리 장소의 집합, 소스 s가 액세스는 가능하지만 기록(또는 판독)이 허용되지 않는 메모리 장소의 집합 등을 포함하는 것들이 가능하다. 그러나, 메모리 액세스 제어에 대한 조건들은 전형적으로 두 개 이상의 메모리 장소의 집합의 여집합(non-intersection)을 검증하는 것을 포함함을 알 것이다.

[0046] 또한, MP(s)가 소스 s에게 보여질 수 있는 "맵핑된 페이지"인 것으로 보여질 수 있지만, 메모리 액세스 제어의 개념은 페이징 구조를 사용하는 시스템에 한정되지 않음을 알아야 한다. 전형적인 구현시, 어떤 메모리 장소가 소스가 정책 하에서 기록이 허용되는지, 혹은 어떤 메모리 장소가 소스로 맵핑되는지에 대한 결정이 페이지 마다(per page basis) 행해진다. 그러나, 본 발명은 메모리가 페이지마다 할당되거나, 메모리에 대한 액세스가 페이지 마다 허용 또는 제한되는 경우에 한정되는 것은 아니다.

[0047] 액세스 변환을 위한 일반화된 모델

[0048] 도 2에 도시되고 전술된 주소 변환 맵은 방향성을 갖는 라벨형 그래프(directed labeled graph)의 모델을 이용하여 일반화될 수 있다. 이하에서는 주소 변환 맵의 소정의 유형에 대한 일반화된 모델에 대해 설명한다.

[0049] 이 모델에서, B는 기본 집합이며, L은 하나의 알파벳이다. B 및 L이 주어질 경우, G=(V, E)가 에지 라벨을 갖는 방향성 그래프이며 이에 따라 V⊆B 및 E⊆{(v,w,l):v∈V, w∈V, l∈L}이 된다. E의 임의의 구성요소는 라벨 l을 갖는, 버텍스(vertex) v로부터 버텍스 w까지의 방향성을 갖는 에지로서 해석될 수 있다. 이 버텍스들도 또한 라벨될 수 있다.

[0050] 도 5는 전술한 모델에 따른 그래프를 나타낸 도면이다. 그래프(500)는 버텍스들(502, 504, 506, 508, 510, 512)을 포함한다. 이들 버텍스들은 도시된 방식으로 에지들(522, 524, 526, 528, 530, 532, 534)에 의해 연결된다. 각 에지는 하나의 알파벳으로부터의 하나의 심볼로 라벨링된다. 이 예에서, 알파벳은 심볼들, A, B 및 C를 포함한다. 이에 따라, 에지들(522, 524)은 심볼 A로 라벨되고, 에지들(526, 528, 532)은 심볼 B로 라벨되고, 에지들(530, 534)은 심볼 C로 라벨된다. 또한, 그래프(500)에서 버텍스들이 아닌 기본 집합의 구성요소들(예를 들면, 구성요소(550, 552))이 존재할 수도 있다.

[0051] 그래프(500)의 컴포넌트들이 도 2에 도시된 주소 변환 맵의 소정의 컴포넌트들에 대응함을 알아야 한다. 예를 들면, 도 2에서 페이지 디렉토리(202), 페이지 테이블(204(1)-204(3)), 및 페이지(206(1)-206(4))는 그래프에서 버텍스들로서 보여질 수 있다. 이들 버텍스들을 연결하는 포인터들(예를 들면, 페이지 테이블 204(1) 내의 엔트리로부터 페이지들 206(1) 및 206(2)로의 포인터들)은 그래프의 에지들로서 보여질 수 있다. 그리고, 도 3과 관련하여, 엔트리의 속성들(304, 306)(예를 들면, 판독 전용 및 존재 비트들)이 에지에 대한 라벨로서 보여질 수 있다. 따라서, "알파벳"은 속성들과 치환가능한 집합이다. (두 개의 바이너리 속성들이 존재하는 도 3의 예에서, 4 개의 가능한 조합이 존재하며 이에 따라 알파벳으로 된 4개의 심볼들이 존재한다.) 속성들이 사용되지 않는 경우에, 알파벳은 "닐(nil)" 심볼로 구성될 수 있다. 또한, 메모리의 할당되지 않은 페이지들은 입력되는 에지들을 갖지 않는 기본 집합의 구성요소들에 대응한다.

[0052] 전술한 바와 같은 그래프의 모델 내에서, "상태"를 정의하는 것이 가능하다. B 및 L이 주어질 경우, "상태"는 쌍(R,G)이며, 여기서 G는 전술한 바와 같이 정의된 방향성을 갖는 라벨형 그래프이며 R⊆V는 G의 버텍스들의 집합이다. R은 "루트(root) 버텍스들"의 집합이다. 루트 버텍스들은, 그래프에 대한 루트로서 합법적으로 기능할 수 있는 기본 집합 내의 버텍스들의 집합을 나타낸다. 도 2의 예에서, 합법적인 디렉토리의 집합(즉, INTEL x86 프로세서 상의 CR3 레지스터와 같은 저장 장소(201)에 로드되도록 허용된 값들)은 "루트 버텍스들"의 집합이다. B 및 L이 주어질 경우, S는 모든 상태들의 집합이다.

[0053] 전술한 바와 같이 정의된 모델에 따르면, 주소 변환 메카니즘(address translation mechanism; ATM)은 이하와 같이 모델링될 수 있다.

[0054] - 버텍스들의 기본 집합 B

[0055] - 알파벳 L(비어 있을 가능성이 있음)

[0056] - 초기 상태 $s_0 \in S$(S는 하나의 상태임)

[0057] - 상태 천이 룰들의 집합(비어 있을 가능성이 있음)

[0058] - 주소 변환 기능

[0059] - 글로벌 플래그

[0060] 상태 천이 룰들은 ATM을 하나의 상태에서 다른 상태로 변경한다. 이에 따라, 상태 천이 룰들의 집합을, r_i:S→S(여기서, i는 소정의 인덱스)로 정의하는 것이 가능하며, 이는 ATM의 현재 상태를 변경한다. ATM은 이하의 천이 룰들의 유형들중 어느 것이나 가질 수 있다.

[0061] - G의 에지들의 변경(추가, 제거, 다시 라벨링)

[0062] - G의 버텍스들을 추가하거나 제거함

[0063] - 루트 집합 R을 변경함

[0064] 예를 들면, 도 2 및 도 3의 예에서, 하나의 페이지에 대한 하나의 포인터를 제거하거나 하나의 페이지의 속성을 변경하는 것은 그래프의 에지의 변경에 대응한다. 새로운 페이지 디렉토리들, 새로운 페이지 테이블들, 또는 새로운 데이터 페이지들을 추가하는 것은 버텍스를 추가하거나 제거하는 것에 대응한다. 베이스 주소가 저장 장소(201)(예를 들면, 레지스터 CR3)에 로드될 수 있는 새로운 페이지 디렉토리를 정의하는 것은 루트 집합의 변경에 대응한다. 본질적으로, 현재의 상태는, 주소 변환 수단에 의해 어떤 메모리 장소들이 잠재적으로 액세

스가능하지를 정의한다.

[0065] 전술한 바와 같이, 메모리에 대한 액세스는 주소 변환 맵에 대한 조건을 제한함으로써 제어될 수도 있으며 이에 따라 주소 변환 맵은, 소스가 정책 하에서 액세스가 허용되지 않는 메모리의 부분에 대한 임의의 가상 주소를 소스에 노출시키지 않는다. 또한, 이전에 알려진 바와 같이, 이들 조건들의 계속되는 존재는, 이 조건의 진실(truth)에 잠재적으로 영향을 미칠 수도 있는 동작이 수행될 때 평가될 수 있다. 메모리 액세스 제어에 대한 이 기술을 보는 방식중 하나는, ATM의 적법한 상태가 S의 소정의 부분 집합 T로 제한되거나, 현재의 상태에 대한 소정의 속성(또는 술어(predicate)) P가 항상 진실이어야 한다는 것이다.

[0066] 소정의 속성 P(이는 전술한 정책 P와는 다른 것임)가 주어질 경우, 상태를 s로부터 $r_i(s)$로 변경할 수도 있는 조치(소정의 i에 대한 r_i의 실행)를 수행하기 위한 요구는, $P(r_i(s))$가 진실인지 여부, 즉 r_i의 실행으로부터 발생될 새로운(제안된) 상태가 속성 P를 가지게 될지 여부를 판정하도록 평가될 수 있다. P의 진실이, 메모리 액세스에 대한 제한이 침해되지 않을 것을 암시하는 경우, $P(r_i(s))$의 진실은, r_i의 실행에 의해 야기된 상태 변화가 진행되도록 허용되어야 함을 의미한다. 이와 다른 경우, 이 동작은 진행되도록 허용되어서는 안된다.

[0067] 모든 메모리 기록은 ATM의 상태를 잠재적으로 변경할 수도 있음을 알아야 한다. 이에 따라, 이하와 같은 두 개의 규약이 이루어져야 한다.

[0068] - 알고리즘은 P(s)를 연산해야 한다(빈번할 가능성이 있음).

[0069] - 전형적으로, 새로운 상태 s'는 오래된 상태 s로부터 얻어진다.

[0070] 오래된 상태가 속성 P를 갖고 있었는 경우, P(s)를 가정하고 그 생성된 s'로의 변경(의 제한된 수)가 P의 침해로 되는지 여부만을 분석함으로써 P(s')를 결정하는 복잡성을 감소시키는 것이 가능할 수도 있다.

[0071] 본 발명은 P의 진실이 효율적으로 평가되도록 허용하는 기술을 제공한다. 이하에 기술하는 바와 같이, 많은 경우에 있어서, 상태 천이 하에서 P의 진실을 확인하기 위해 어떤 테스트들이 수행될 필요가 있는지와 어떤 테스트들을 피할 수 있는지를 결정하는데에 후에 이용될 수 있는, ATM의 현재의 상태에 관한 소정의 대표적인 정보를 저장(혹은 캐싱)함으로써 이 효율성이 획득될 수 있다.

[0072] 예시적인 속성 등급들

[0073] 속성 P의 유형중 하나는, 버텍스들의 집합으로 표현될 수 있는 속성이다. 예를 들면, 도 4에 도시되고 전술된 조건은 본질적으로, MP(s) 및 NA(P,s)의 집합들이 서로 교차하지 않는 속성이다. 버텍스들의 집합들과, 이들 집합들간의 관계로 표현될 수 있는 많은 속성들은 하나의 집합 내에 버텍스들의 아이덴티티를 저장(또는 캐싱)함으로써 효율적으로 구현될 수 있다.

[0074] ATM이 메모리 액세스 제어 조건을 만족시키는 상태에 있는지를 평가하는데에 유용할 수 있는 집합들의 예는 이하와 같다.

[0075] 1. 루트 버텍스들로부터 거리 k에 있는 버텍스들의 집합. 공식으로 나타내면, S가 버텍스들의 집합이고 w가 버텍스일 경우, $d_k(S,w)$가, S 내의 소정의 버텍스으로부터 버텍스 w까지의 길이 k의 (방향성을 갖는) 경로가 존재한다는 제어문(statement)을 나타내는 경우, $S_d = \{v \in V: dk(S,v)\}$ 이다. 그 후, S가 루트 버텍스일 경우, S_d는 루트로부터의 거리 d에서의 페이지들의 집합을 나타낸다. 예를 들면, 버텍스(502)가 그래프(500)의 루트일 경우, 루트 버텍스로부터 거리 1을 갖는 버텍스의 집합은 버텍스들(502, 510)로 구성되는데, 이는 이들 버텍스들중 하나가 하나의 에지를 가로질러 루트로부터 도달할 수 있기 때문이다. 도 2에 도시된 페이지 맵을 참조하면, 페이지 디렉토리(202)는 루트로부터 거리 1이며, 페이지 테이블 204(1) 내지 204(3)은 루트로부터 거리 2이다. 이에 따라, 도 2의 예에서, 페이지 디렉토리 및 페이지 테이블의 주소들은, 각각 루트로부터 거리 1 및 2인 페이지들의 아이덴티티를 저장함으로써 캐싱될 수 있다.

[0076] 2. 에지 라벨들에 의해 결정되는 집합들. 예를 들어, 도 5를 참조하면, 인-에지(in-edge) 라벨된 "A"를 갖는 버텍스들의 집합은 버텍스들(504, 510)로 구성되고 인-에지 라벨된 "B"를 갖는 버텍스들의 집합은 버텍스들(504, 506, 512)로 구성된다. 속성들이 에지 라벨들에 대응하는 도 2의 페이지 맵에서, 하나의 집합은 소정의 속성을 갖는 페이지들로서 정의될 수도 있다. 예를 들면, 이는, 판독 전용으로 표시된 페이지들의 집합을 정의(및 캐싱)하는데에 유용할 수도 있으며 이 경우 판독 전용 비트가 "온(on)"(도 3에 도시된 참조 번호 304)인 페이지들의 집합이 정의될 수 있다. (하나의 페이지가 하나의 페이지 맵에서 한번 이상 참조되는 것이 가능한데,

이 경우 하나의 페이지에 대한 서로 다른 참조들은 서로 다르게 자신의 판독 전용 속성 집합을 가질 수도 있으며; 이 경우, 이 집합의 정의는 충돌을 해결할 수도 있는데, 예를 들면, 그 페이지에 대한 적어도 하나의 참조가 판독 전용 속성을 가지는 경우, 혹은 그 페이지에 대한 모든 참조가 판독 전용 속성을 가지는 경우 등에 그 페이지가 그 집합 내에 있는 것이다.

[0077] 국부적 속성 및 국부적이 아닌 속성들간에 구별이 이루어질 수 있다. 국부적 특성은 소정의 버텍스 상으로 발생되는 에지들로부터 계산될 수 있다. 즉, 버텍스 v가 v로 발생되는 에지들만으로부터의 속성 P를 가지는지를 결정하는 것이 가능한 경우, P는 국부적인 것으로 된다. 이와 다른 경우, P는 국부적이 아니다. 국부적 속성의 예는, "버텍스는 판독-기록으로 라벨된 인-에지를 갖는다"라는 것이다. 국부적이 아닌 속성의 예는, "(x86 머신 상의) 페이지는 판독-기록 맵핑을 갖는다"라는 것이다.

[0078] 3. 몇몇 속성을 갖는 k개의 에지들의 타겟인 버텍스들의 집합. 공식으로 나타내면, P, Q가 술어이고 w가 버텍스인 경우 이하와 같이 된다.

[0079] $In\text{-}deg_{P,Q}(w) = |\{v \in V : P(v) \text{ 및 } (v,w,l) \in E \text{ 및 } Q(l)\}|$

[0080] 하나의 집합은 소정의 인-디그리(in-degree)를 갖는 버텍스들의 집합으로 정의될 수도 있다.

[0081] $\{v \in V : In\text{-}deg_{P,Q}(v) = k\}$

[0082] 유사하게, 집합들은 불균등하게 정의될 수 있는데, 예로써, 소정의 속성을 갖는 k개 이상의(혹은 k개보다 적은) 에지들의 타겟인 버텍스들의 집합을 들 수 있다.

[0083] 예를 들면, 도 5를 참조하면, 적어도 하나의 "C" 라벨된 인-에지를 갖는 버텍스들의 집합은 버텍스들(508, 512)로 구성된다. 도 2의 페이지 맵을 참조하면, 이 집합 정의 유형은 페이지들의 카테고리를 캐싱하는데에 사용될 수 있는데, 예로써, 두 개 이상의 맵핑을 갖는 페이지들의 집합, 정확히 하나의 판독 전용 맵핑을 갖는 페이지들의 집합 등을 들 수 있다.

[0084] 4. 유사한 집합이 아웃-디그리(out-degree)에 기초하여 정의될 수도 있는데, 즉, 소정의 속성을 갖는 k개의 아웃 에지들(out edges)(혹은 k 개 이상의 아웃 에지들, 혹은 k개 보다 적은 아웃 에지들)을 갖는 버텍스들의 집합. 예를 들어, 도 5를 참조하면, 정확히 두 개의 "A" 라벨된 아웃 에지들을 갖는 버텍스들의 집합은 버텍스(502)로 구성된다. 도 2는 유사한 예들을 포함하는데, 예를 들면 적어도 3개의 아웃 에지들(즉, 다른 페이지들에 대한 참조)을 갖는 페이지들의 집합이 페이지 디렉토리(202)를 포함한다.

[0085] 이들 집합들은 통상의 집합 연산(예를 들면, 합집합, 교집합, 여집합(complement), 차집합(set difference))을 통해 결합될 수 있다. 예를 들면, S_2가 루트로부터 거리 2에 있는 페이지들의 집합인 경우, x86 CPU의 소정의 구성 장치 내의 판독-기록 맵핑들을 갖는 페이지들의 집합은 이하와 같이 표현될 수 있다.

[0086] ({x:x는 큰 페이지의 인-에지를 가짐} 교집합 {x:x는 r/w 인-에지를 가짐} 교집합 S_2) 합집합 ({x:x는 작은 페이지의 인 에지를 가짐} 교집합 {x:x는 r/w 맵핑을 가짐} 교집합 S_3)

[0087] 일반적(naive) 알고리즘은 모든 버텍스 v를 통과하고, 그 집합에 속하는지를 테스트함으로써 각각의 상태 변화시에 이 집합들을 재계산할 수도 있다. 이것은 비용이 많이 들 수 있다. 알고리즘이, 상술한 유형들의 집합들로 표현될 수 있는 상태 속성들을 계산하면, 알고리즘은 앞서 설명된 바와 같은 캐싱 스킴들(caching schemes)을 이용할 수 있다.

[0088] 캐싱 스킴들:

[0089] 상태 변화들의 효율적인 계산용으로 데이터를 캐싱하는데에 다양한 스킴들이 이용될 수 있다. 실례의 캐싱 스킴들이 아래에 설명된다.

[0090] 스킴 1: 단순 집합 캐싱

[0091] 이 스킴은 집합을 명시적으로 계산하여, 이를 저장(캐싱)한다. 각각의 후속되는 상태 변화 시에, 알고리즘은 캐시를 갱신한다. 일 예에서, 캐시는 다음과 같은 액세스 연산들에 노출되어 유지될 수 있다.

[0092] -Init() -- 캐시를 빈 집합와 같은 소정의 잘 정의된 값으로 초기화한다.

[0093] -Add(S) -- 캐시에 S(단일 원소 혹은 원소들의 집합)를 추가한다.

[0094] -Remove(S) -- 캐시로부터 S(단일 원소 혹은 원소들의 집합)를 제거한다.

[0095] -Showcache(S) -- 현재 캐시되어 있는 모든 원소들을 리턴한다.

[0096] 캐시는 (예를 들어 효율을 높이기 위해) 추가적인 액세스 연산들에 노출될 수도 있다.

[0097] 이러한 캐시를 나타내기 위한 방법중 하나는 비트 벡터를 통하는 것이다. 예를 들어, 시스템이 2^{16}개의 물리적 메모리 페이지를 갖는다면, 2^{16} 비트 길이(즉, 8Kbytes)인 벡터가 각 페이지에 대한 부울 값을 나타낼 수 있다. n번째 비트는, n번째 페이지가 정의된 집합 내에 있는지 여부에 따라 온 또는 오프이다. 따라서, 정의된 페이지들 집합이 주어지면, 집합 내의 멤버쉽은 페이지당 한 비트의 비용으로 캐시될 수 있다. 비트별 "or" 및 "and" 연산자를 이용함으로써, 합집합 및 교집합과 같은 집합 연산들이 이러한 유형의 표현들로 수행되기에 매우 간편함을 알 것이다.

[0098] 스킴 2: 포함 집합, 부분 집합

[0099] 메모리 액세스 제어를 수행하는 기초 알고리즘의 상세 내역들에 따라, 캐시는 정확한 타겟 집합을 포함할 필요가 없을 수도 있다. 예를 들어, 캐시는 타겟 집합의 소정의 포함 집합이나 소정의 부분 집합을 캐시하는 것으로 충분할 수 있다. 이것은 캐시를 유지하는 비용을 감소시킬 수 있다. 도 3의 예에서, 메모리 액세스 제어 조건은 MPS(s)가 NA(P, s)와 교차하지 않는 것을 요구한다. 그러나, NA(P, s)의 정확한 멤버들을 계산하는 것이 불편하거나 비실용적이면, NA(P, s)의 소정의 포함 집합을 계산하여 캐시하고, 그런 다음 MP(s)가 NA(P, s)의 계산된 포함 집합과 교차하지 않도록 하는 것이 가능할 수도 있다. 이 기술은 이와 다른 경우 허용될 수 있는 소정의 상태 변화들의 거절을 유발할 수 있으나, 허용되지 않아야 할 임의의 상태 변화들을 허용하지 않을 것이며, 이로써 메모리 액세스 제어에 대한 조건을 보존할 것이다.

[0100] 스킴 3: 예약된 에지 표현

[0101] 전형적으로, 에지들은 소스 버텍스 내에 혹은 그와 함께 저장된다. 예를 들어, 도 2에서, 페이지 디렉토리와 페이지 테이블들은 다른 페이지들에 대한 포인터들 및 그들의 속성들을 저장한다. 버텍스가 주어지면, 모든 나가는(outgoing) 에지들의 타겟들을 찾는 것은 전형적으로 쉽다. 이와 동시에, 모든 인-에지들의 소스들을 찾는 것은 전형적으로 비용이 많이 든다. 버텍스가 그의 인-에지들에 관한 정보를 아무 것도 전달하고 있지 않기 때문에, 모든 인-에지들을 찾기 위해 모든 에지들의 철저한 검색이 요구될 수도 있다.

[0102] 알고리즘이 버텍스들의 인-에지들, 또는 이들로부터 구해질 수 있는 정보에 대한 고속 액세스를 필요로 하면, 각 버텍스의 인-에지들에 관한 정보를 그 버텍스과 어떻게든 관련된 데이터 구조 내에 명시적으로 저장하는 것이 바람직할 수 있다. 상기 "어떻게든 관련된"이란 용어는, 버텍스가 주어지면, 데이터 구조를 찾는 것(예를 들어, 어레이 검색)이 쉽다는 것을 의미한다.

[0103] 가장 극단적인 경우에, 데이터 구조는 모든 인-에지들을 저장한다. 이 경우에, 데이터 구조는 상기에서 정의된 것과 같은, 원소들이 에지들인 캐시일 수 있다. (또한, 캐시들은 집합들 또는 다중 집합들을 저장할 수 있다.) 이 구조가 저장되는 저장 장치는 버텍스의 인-에지들의 개수에 비례하고, 이러한 유형의 구조들이 모든 버텍스들에 대해 유지된다면, 총 저장 장치는 그래프 내의 에지들의 개수에 비례한다.

[0104] 종종 적은 용량의 저장 장치를 필요로 할 수 있는 유도된 정보를 저장하는 것으로 충분하다. 예를 들어, 알고리즘은 각 버텍스의 인-에지들의 개수만을 저장할 수 있다. 이 경우, 캐시는 참조 카운터로서 구현될 수 있다. 참조 카운터는 전형적으로 다음과 같은 액세스 연산들에 노출된다.

[0105] -Init() -- 캐시를 0과 같은 소정의 잘 정의된 값으로 초기화한다.

[0106] -Increment()

[0107] -Decrement()

[0108] -GetValue()

[0109] 참조 카운터들 (혹은 그와 유사한 데이터 구조들)의 통상적인 이용 방법중 하나는 집합들을 구성하는 것이다. 예를 들어, 예시적인 메모리 액세스 제어 알고리즘은 인-에지들이 없는 버텍스들의 집합, 즉 참조 카운트가 0인 버텍스들의 집합을 계산해야 할 수도 있다. 참조 카운터들의 콜렉션은 다음과 같이 이 집합의 캐시(스킴 1)를 제어할 수 있다. 즉, 참조 카운터의 값이 바뀔 때마다, 알고리즘은 카운터가 영으로 되었는지를 테스트한다. 그렇다면, 알고리즘은 버텍스를 캐시에 추가한다. 마찬가지로, 알고리즘은 0인 참조 카운터가 다른 값을 얻는

이벤트가 있는지 감시한다. 이 이벤트에서, 알고리즘은 버텍스를 캐시로부터 제거한다.

[0110] 다음은 캐싱을 이용하는 예들이다.

[0111] - d=1, 2, 3에 대한 S_d의 포함 집합 S_d'의 캐싱

[0112] - d=2, 3에 대해: 캐시는 (a) 명시적으로 저장되거나, 혹은 (b) 참조 카운터에 의해 구동될 수 있음

[0113] - 국부적인 라벨 속성을 계산: "인-에지들의 판독-기록을 가짐" 및 "인-에지들의 많고/적은 페이지를 가짐"

[0114] - 국부적이 아닌 속성들을 계산: "판독-기록 맵핑을 가짐"

[0115] - S_2 내의 버텍스들의 인-에지들의 판독-기록의 횟수에 대해 참조 카운터를 이용.

[0116] 이 정보는 국부적이 아닌 속성 "판독-기록 맵핑을 가짐"의 계산 속도를 증가시키기 위해 사용될 수 있다.

[0117] 저장된 정보를 이용한 메모리 액세스 제어를 위한 예시적인 프로세스

[0118] 도 6은 본 명세서에 개시된 기술을 이용하여, 메모리 액세스 제어를 수행하기 위한 예시적인 프로세스를 나타낸다.

[0119] 처음에, 메모리를 액세스하기 위한 요청이 수신된다(단계 602). 액세스 요청이 수신될 때, 메모리 액세스 제어 시스템은 요청을 평가하여 이 요청의 실행이 메모리 액세스를 관장하는 정책에 따르는 것인지의 여부를 판정한다(단계 604). 메모리 액세스 정책들의 예들은 위에서 기술되었다. 한가지 예로서, 정책은 소정의 페이지들을 소스들의 집합에 대한 출입 금지 영역(off-limits)으로서 정의하고, 정책은 출입 금지 영역 페이지들 중 하나에 대해, 생성이나, 그 페이지를 액세스하도록 허용되지 않은 소스들 중 하나에게 보여질 수도 있는 맵핑을 초래하게 되는 임의의 액세스 요청을 금지할 수 있다. 이 요청의 평가는 저장되거나 캐싱된 캐시된 정보에 의해 도움을 받을 수 있다(단계 606). 이 저장되거나 캐싱된 정보는 예를 들면 정당한 페이지 디렉토리들을 포함하는 것으로 알려진 페이지들의 집합인 페이지 맵(들)에 관한 정보를 포함할 수 있다.

[0120] 요청을 실행하는 것이 정책의 준수를 유지할 것으로 판정되면(단계 608), 이 요청은 처리되도록 허용된다(단계 612). 그렇지 않으면, 이 요청은 차단되거나, 또는 정책을 위반하지 않을 형태로 변경된다(단계 610). 정책을 위반하지 않는 형태로 요청을 변경하는 한가지 예는 다음과 같다. 요청이, 출입 금지 영역 페이지에 맵핑을 초래하게 될, 페이지 테이블에 엔트리를 기록하고자 하면, 엔트리는 기록되지만 페이지의 "존재" 비트가 턴오프되도록 요청이 변경될 수 있다. 따라서, 새롭게 맵핑된 페이지를 액세스하기 위한 임의의 미래의 시도는 예외를 발생시키게 되어,예외 핸들러가 출입 금지 페이지에 대한 액세스를 궁극적으로 방해할 수 있다. 이러한 방식으로(혹은 소정의 다른 방식으로) 요청이 변경되면, 변경된 요청이 처리되도록 허용된다(단계 614). 변경되거나 변경되지 않은 요청이 수행된 후, 요청의 실행이 캐싱된 정보에 대한 변경을 유발한다면, 캐시는 갱신될 수 있다(단계 616).

[0121] 상술한 예들은 단순히 설명을 목적으로 제공되었고, 본 발명을 제한하려고 해석되어서는 안된다. 본 발명은 다양한 실시예들을 참조하여 설명되었으나, 본 명세서에서 사용된 단어들은 설명 및 예시의 단어들일 뿐 제한하는 단어들이 아니다. 또한, 비록 본 발명은 특정한 수단, 물질, 및 실시예들을 참조하여 설명되었으나, 본 발명은 본 명세서에 개시된 특정물들로만 제한되는 것은 아니다. 오히려, 본 발명은 첨부된 청구범위의 범위 내에 있는 기능적으로 등가적인 모든 구조, 방법, 및 용도들까지 확장된다. 본 명세서의 교시를 이해하는 당업자는 본 발명에 대해 다양한 변형을 할 수 있을 것이고, 이러한 변형들은 본 발명의 범위 및 사상을 벗어나지 않고서 이루어질 것이다.

발명의 효과

[0122] 본 발명에 따르면 소정의 메모리 액세스 제어 알고리즘들이 효율적으로 구현될 수 있도록 해주는 메카니즘들이 제공된다.

도면의 간단한 설명

[0001] 도 1은 본 발명의 특징이 구현될 수도 있는 컴퓨팅 환경의 블럭도.

[0002] 도 2는 주소 변환 맵을 통해 가상 주소 지정을 구현하는 메모리 시스템의 블럭도.

[0003] 도 3은 속성들을 갖는 예시적인 페이지 테이블의 블럭도.

[0004] 도 4는 메모리 액세스 제어를 구현하는데에 이용될 수도 있는 조건을 나타내는 두 개의 예시적인 교차하지 않는 집합의 블럭도.

[0005] 도 5는 주소 변환 맵을 나타내는, 방향성을 갖는 라벨형 그래프의 블럭도.

[0006] 도 6은 예시적인 메모리 액세스 제어 프로세스의 흐름도.

[0007] <도면의 주요 부분에 대한 부호의 설명>

[0008] 110 : 컴퓨터

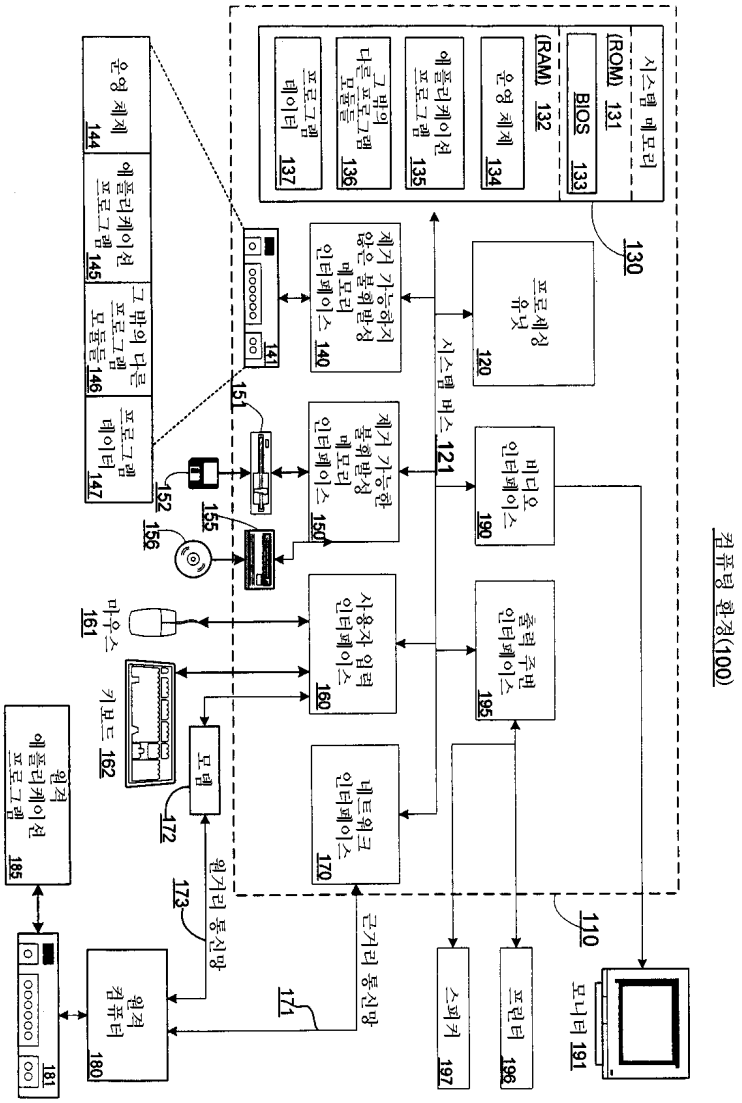
[0009] 120 : 프로세싱 유닛

[0010] 121 : 시스템 버스

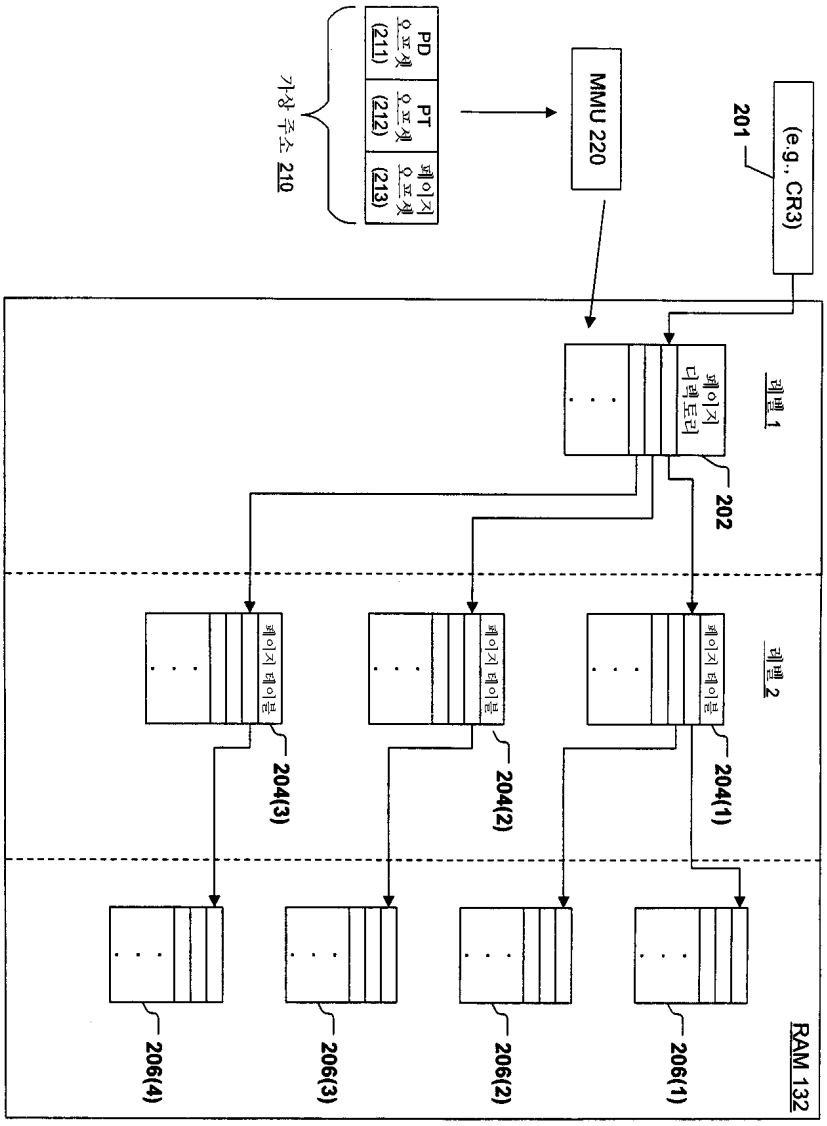
[0011] 130 : 시스템 메모리

도면

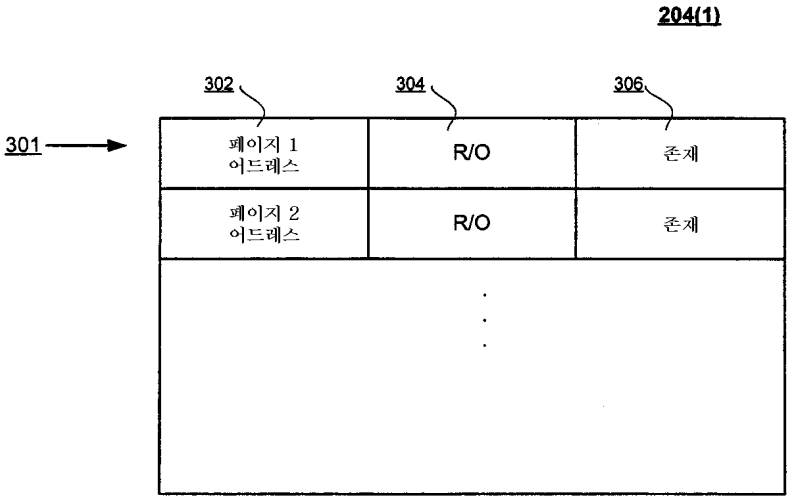
도면1



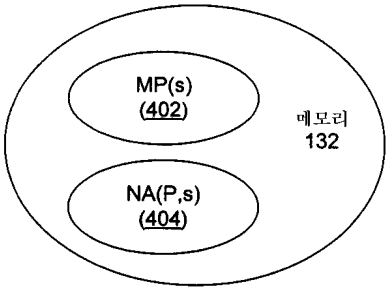
도면2


(e.g., CR3)
201
MMU 220
PD 오프셋 (211)
PT 오프셋 (212)
페이지 오프셋 (213)
가상 주소 210
레벨 1
페이지 디렉토리
202
레벨 2
페이지 테이블
204(1)
204(2)
204(3)
206(1)
206(2)
206(3)
206(4)
RAM 132

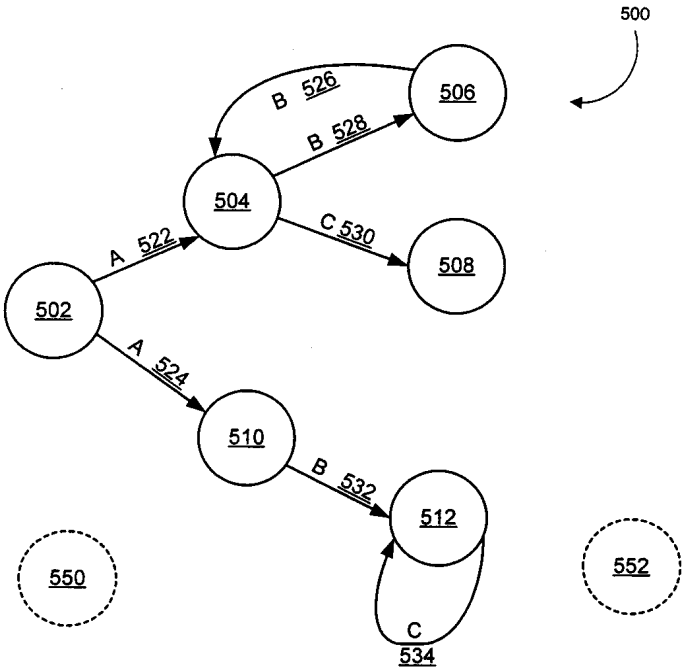
도면3


204(1)
302
304
306
301
페이지 1 어드레스
R/O
존재
페이지 2 어드레스
R/O
존재

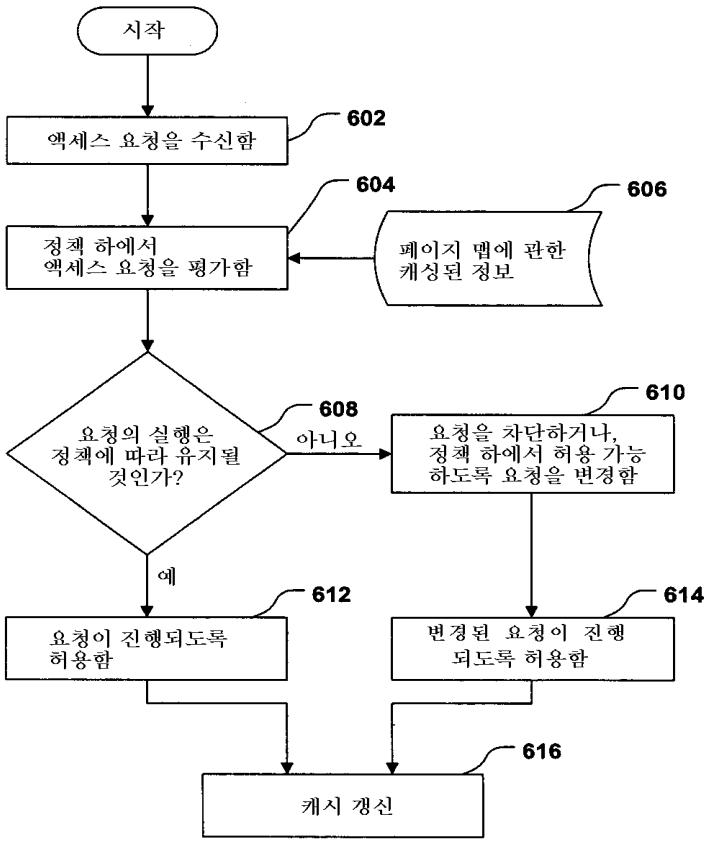
도면4


MP(s)
(402)
NA(P,s)
(404)
메모리
132

도면5


500
506
B 526
B 528
504
C 530
508
A 522
502
A 524
510
B 532
512
550
552
C
534

도면6


시작
액세스 요청을 수신함
602
604
606
정책 하에서
액세스 요청을 평가함
페이지 맵에 관한
캐싱된 정보
요청의 실행은
정책에 따라 유지될
것인가?
608
아니오
610
요청을 차단하거나,
정책 하에서 허용 가능
하도록 요청을 변경함
예
612
614
요청이 진행되도록
허용함
변경된 요청이 진행
되도록 허용함
616
캐시 갱신