(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2010년02월25일
(11) 등록번호 10-0944563
(24) 등록일자 2010년02월19일

(51) Int. Cl.
G06F 15/76 (2006.01) *G06F 9/06* (2006.01)
(21) 출원번호 10-2007-0082482
(22) 출원일자 2007년08월16일
심사청구일자 2007년08월16일
(65) 공개번호 10-2008-0015763
(43) 공개일자 2008년02월20일
(30) 우선권주장
11/504,964 2006년08월15일 미국(US)
(56) 선행기술조사문헌
KR1020060083168 A

(73) 특허권자
인텔 코오퍼레이션
미합중국 캘리포니아 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자
베넷, 스티븐 엠.
미국 97123 오레곤주 힐스보로 사우쓰이스트 씨그리드 스트리트6469
앤더슨, 앤드류 브이.
미국 97123 오레곤주 힐스보로 사우쓰이스트 68번 애비뉴 677
(뒷면에 계속)
(74) 대리인
백만기, 양영준

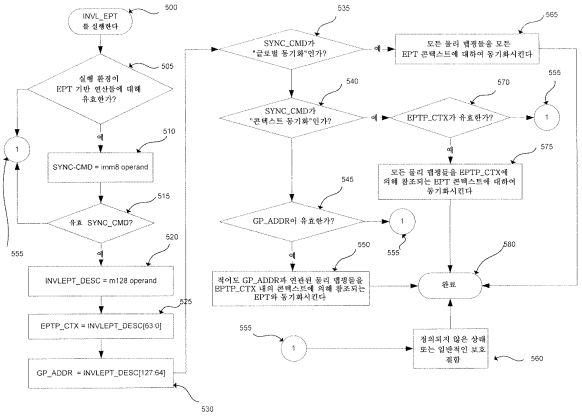
전체 청구항 수 : 총 34 항 심사관 : 성경아

(54) 변환 색인 버퍼의 확장 페이지 테이블에의 동기화

(57) 요 약

TLB(translation lookaside buffer)에 저장된, 가상화 기반 시스템의 게스트의 물리 주소(게스트 물리 주소)로부터 가상화 기반 시스템의 호스트의 물리 주소(호스트 물리 주소)로의 맵핑을, 가상화 기반 시스템의 확장 페이지 테이블(EPT)에 저장된 대응하는 맵핑과 동기화시키도록 명령어를 실행하는 로직을 포함하는 프로세서가 개시된다.

대 표 도



(72) 발명자

네이거, 길버트

미국 97212 오레곤주 포트랜드 노쓰이스트 11번 애비뉴 2424

얼릭, 리차드

미국 97124 오레곤주 힐스보로 노쓰이스트 오렌코 스테이션파크웨이 더블유 1564

로저스, 다이온

미국 97123 오레곤주 힐스보로 사우쓰웨스트 브룩우드 애비뉴 452

매드카르무크마나, 라제쉬

미국 97229 오레곤주 포트랜드 노쓰웨스트 밴스 드라이브 15132

러스트, 캠론

미국 97124 오레곤주 힐스보로 노쓰이스트 세컨드 드라이브 2556

쉰베르그, 세바스찬

미국 97124 오레곤주 힐스보로 사우쓰이스트 47번 애비뉴 1222

특허청구의 범위

청구항 1

가상화 기반 시스템의 프로세서로서,

프로세서 버스;

게스트 주소로부터 호스트 물리 주소로의 맵핑을 저장하는 버퍼;

명령어를 수신하고 피연산자를 수신하는 페치 로직;

상기 명령어를 디코딩하는 디코드 로직; 및

상기 명령어의 디코딩에 적어도 부분적으로 응답하여, 상기 버퍼에 저장된, 게스트 주소로부터 호스트의 물리 주소(호스트 물리 주소)로의 맵핑을, 확장 페이지 테이블(extended paging table: EPT)에 적어도 부분적으로 저장된 대응하는 맵핑과 동기화시키는 논리 회로

를 포함하고,

상기 동기화는 또한 상기 명령어의 상기 피연산자에 적어도 부분적으로 기초하며, 상기 피연산자는 콘텍스트 디스크립터(context descriptor)를 포함하는 프로세서.

청구항 2

제1항에 있어서,

상기 버퍼는 TLB(translation lookaside buffer)를 포함하는 프로세서.

청구항 3

제1항에 있어서,

상기 논리 회로는 마이크로코드 명령어들에 적어도 부분적으로 기초하여 동작하는 논리 회로를 포함하는 프로세서.

청구항 4

제1항에 있어서,

상기 EPT는 버스에 의해 상기 프로세서에 결합되는 메모리에 적어도 부분적으로 저장되는 프로세서.

청구항 5

제1항에 있어서,

상기 논리 회로는 또한 상기 명령어의 상기 피연산자로부터 도출되는 콘텍스트 디스크립터에 적어도 부분적으로 기초하여 상기 EPT로부터의 상기 맵핑을 선택하는 프로세서.

청구항 6

제1항에 있어서,

상기 논리 회로는 또한 상기 콘텍스트 디스크립터로부터 도출되는 EPT 포인터에 적어도 부분적으로 기초하여 상기 EPT로부터의 상기 맵핑을 선택하는 프로세서.

청구항 7

제1항에 있어서,

상기 논리 회로는 또한 상기 명령어의 피연산자에 적어도 부분적으로 기초하여 상기 게스트 주소를 선택하는 프로세서.

청구항 8

제2항에 있어서,

상기 맵핑의 동기화는 또한 상기 EPT에 저장된 상기 맵핑에 적어도 부분적으로 기초하여 상기 TLB에 저장된 상기 맵핑을 갱신하는 것을 포함하며,

상기 대응하는 맵핑은 또한 상기 TLB에 저장된 상기 맵핑과 동일한 게스트 주소를 갖는 상기 EPT에 저장된 맵핑을 포함하는 프로세서.

청구항 9

제2항에 있어서,

상기 맵핑의 동기화는 또한 상기 TLB에 저장된 맵핑을 플러싱(flushing)하는 것을 포함하는 프로세서.

청구항 10

제1항에 있어서,

상기 논리 회로는 또한 상기 명령어의 피연산자에 적어도 부분적으로 기초하여 상기 명령어의 실행 모드를 선택하는 프로세서.

청구항 11

제2항에 있어서,

상기 페치 로직은 또한 상기 명령어의 제1 피연산자, 상기 명령어의 제2 피연산자, 및 상기 명령어의 제3 피연산자를 수신하며,

상기 논리 회로는 또한,

상기 명령어의 상기 제1 피연산자로부터 도출되는 콘텍스트 디스크립터에 기초하여 상기 EPT에 적어도 부분적으로 저장된 상기 맵핑을 선택하고,

상기 명령어의 상기 제2 피연산자에 적어도 부분적으로 기초하여 상기 게스트 주소를 선택하고,

상기 명령어의 상기 제3 피연산자에 적어도 부분적으로 기초하여 상기 명령어의 실행 모드를 선택하고,

상기 명령어의 상기 실행 모드는,

상기 TLB에 저장되어 있고 상기 게스트 주소와 연관되는 단일 맵핑만이 상기 EPT 내의 대응하는 맵핑과 동기화되는 제1 모드;

상기 TLB에 저장되어 있고 상기 콘텍스트 디스크립터로부터 도출되는 EPT 콘텍스트와 연관되는 모든 맵핑들이 상기 EPT 내의 대응하는 맵핑들과 동기화되는 제2 모드; 및

상기 TLB에 저장되어 있고 임의의 EPT 콘텍스트와 연관되는 모든 맵핑들이 상기 EPT 내의 대응하는 맵핑들과 동기화되는 제3 모드 중 하나인 프로세서.

청구항 12

제1항에 있어서,

상기 게스트 주소는 또한 게스트 물리 주소를 포함하는 프로세서.

청구항 13

제1항에 있어서,

상기 게스트 주소는 또한 게스트 선형 주소를 포함하는 프로세서.

청구항 14

호스트 및 게스트를 포함하는 가상화 기반 시스템에서,

TLB에 적어도 부분적으로 저장된, 게스트 주소로부터 상기 호스트의 물리 주소(호스트 물리 주소)로의 변환(translation)을 포함하는 맵핑을, 상기 가상화 기반 시스템의 EPT에 적어도 부분적으로 저장된 대응하는 맵핑과 동기화시키는 단계; 및

명령어의 피연산자에 적어도 부분적으로 기초하여 상기 EPT에 적어도 부분적으로 저장된 상기 맵핑을 선택하는 단계

를 포함하고,

상기 피연산자는 콘텍스트 디스크립터를 포함하는 방법.

청구항 15

제14항에 있어서,

상기 명령어의 피연산자로부터 도출되는 콘텍스트 디스크립터에 부분적으로 기초하여 상기 EPT로부터의 상기 맵핑을 선택하는 단계를 더 포함하는 방법.

청구항 16

제14항에 있어서,

상기 콘텍스트 디스크립터로부터 도출되는 EPT 포인터에 부분적으로 기초하여 상기 EPT에 적어도 부분적으로 저장된 상기 맵핑을 선택하는 단계를 더 포함하는 방법.

청구항 17

제14항에 있어서,

상기 명령어의 피연산자에 적어도 부분적으로 기초하여 상기 게스트 주소를 선택하는 단계를 더 포함하는 방법.

청구항 18

제14항에 있어서,

상기 맵핑을 동기화시키는 단계는, 상기 EPT에 저장된 상기 맵핑에 적어도 부분적으로 기초하여 상기 TLB에 저장된 상기 맵핑을 갱신하는 단계를 더 포함하고,

상기 대응하는 맵핑은 또한 상기 TLB에 저장된 상기 맵핑과 동일한 게스트 주소를 갖는 상기 EPT에 저장된 맵핑을 포함하는 방법.

청구항 19

제14항에 있어서,

상기 맵핑을 동기화시키는 단계는, 상기 TLB로부터 저장된 상기 맵핑을 플러싱하는 단계를 더 포함하는 방법.

청구항 20

제14항에 있어서,

상기 명령어의 피연산자에 적어도 부분적으로 기초하여 상기 명령어의 실행 모드를 선택하는 단계를 더 포함하는 방법.

청구항 21

제14항에 있어서,

상기 명령어의 제1 피연산자로부터 도출되는 콘텍스트 디스크립터에 기초하여 상기 EPT에 적어도 부분적으로 저장된 상기 맵핑을 선택하는 단계;

상기 명령어의 제2 피연산자에 적어도 부분적으로 기초하여 상기 게스트 주소를 선택하는 단계; 및

상기 명령어의 제3 피연산자에 적어도 부분적으로 기초하여 상기 명령어의 실행 모드를 선택하는 단계를 더 포

함하고,

상기 명령어의 상기 실행 모드는,

상기 TLB에 저장되어 있고 상기 게스트 주소와 연관되는 단일 맵핑만이 상기 EPT 내의 대응하는 맵핑과 동기화되는 제1 모드;

상기 TLB에 저장되어 있고 상기 콘텍스트 디스크립터로부터 도출되는 EPT 콘텍스트와 연관되는 모든 맵핑들이 상기 EPT 내의 대응하는 맵핑들과 동기화되는 제2 모드; 및

상기 TLB에 저장되어 있고 임의의 EPT 콘텍스트와 연관되는 모든 맵핑들이 상기 EPT 내의 대응하는 맵핑들과 동기화되는 제3 모드 중 하나인 방법.

청구항 22

제14항에 있어서,

상기 게스트 주소는 또한 게스트 물리 주소를 포함하는 방법.

청구항 23

제14항에 있어서,

상기 게스트 주소는 또한 게스트 선형 주소를 포함하는 방법.

청구항 24

가상화 기반 시스템으로서,

프로세서;

버스에 의해 상기 프로세서에 결합되는 메모리; 및

TLB에 적어도 부분적으로 저장된, 게스트 주소로부터 호스트의 물리 주소(호스트 물리 주소)로의 변환을 포함하는 맵핑을, 상기 가상화 기반 시스템의 EPT에 적어도 부분적으로 저장된 대응하는 맵핑과 동기화시키고,

명령어의 피연산자에 적어도 부분적으로 기초하여 상기 EPT에 적어도 부분적으로 저장된 상기 맵핑을 선택하도록 명령어를 실행하는 상기 프로세서의 논리 회로

를 포함하고,

상기 피연산자는 콘텍스트 디스크립터를 포함하는 가상화 기반 시스템.

청구항 25

제24항에 있어서,

상기 프로세서는 또한 상기 명령어의 피연산자로부터 도출되는 콘텍스트 디스크립터에 부분적으로 기초하여 상기 EPT로부터 상기 맵핑을 선택하는 가상화 기반 시스템.

청구항 26

제24항에 있어서,

상기 프로세서는 또한 상기 콘텍스트 디스크립터로부터 도출되는 EPT 포인터에 부분적으로 기초하여 상기 EPT에 적어도 부분적으로 저장된 상기 맵핑을 선택하는 가상화 기반 시스템.

청구항 27

제24항에 있어서,

상기 프로세서는 또한 상기 명령어의 피연산자에 적어도 부분적으로 기초하여 상기 게스트 주소를 선택하는 가상화 기반 시스템.

청구항 28

제24항에 있어서,

상기 논리 회로는 또한 상기 EPT에 저장된 상기 맵핑에 적어도 부분적으로 기초하여 상기 TLB에 저장된 상기 맵핑을 갱신하며,

상기 대응하는 맵핑은 또한 상기 TLB에 저장된 상기 맵핑과 동일한 게스트 주소를 갖는 상기 EPT에 저장된 맵핑을 더 포함하는 가상화 기반 시스템.

청구항 29

제28항에 있어서,

상기 논리 회로는 또한 상기 TLB로부터 저장된 상기 맵핑을 플러싱하는 가상화 기반 시스템.

청구항 30

제24항에 있어서,

상기 논리 회로는 또한 상기 명령어의 피연산자에 적어도 부분적으로 기초하여 상기 명령어의 실행 모드를 선택하는 가상화 기반 시스템.

청구항 31

제24항에 있어서,

상기 논리 회로는 또한,

상기 명령어의 제1 피연산자로부터 도출되는 콘텍스트 디스크립터에 기초하여 상기 EPT에 적어도 부분적으로 저장된 상기 맵핑을 선택하고,

상기 명령어의 제2 피연산자에 적어도 부분적으로 기초하여 상기 게스트 주소를 선택하고,

상기 명령어의 제3 피연산자에 적어도 부분적으로 기초하여 상기 명령어의 실행 모드를 선택하고,

상기 명령의 상기 실행 모드는,

상기 TLB에 저장되어 있고 상기 게스트 주소와 연관되는 단일 맵핑만이 상기 EPT 내의 대응하는 맵핑과 동기화되는 제1 모드;

상기 TLB에 저장되어 있고 상기 콘텍스트 디스크립터로부터 도출되는 EPT 콘텍스트와 연관되는 모든 맵핑들이 상기 EPT 내의 대응하는 맵핑들과 동기화되는 제2 모드; 및

상기 TLB에 저장되어 있고 임의의 EPT 콘텍스트와 연관되는 모든 맵핑들이 상기 EPT 내의 대응하는 맵핑들과 동기화되는 제3 모드 중 하나인 가상화 기반 시스템.

청구항 32

제24항에 있어서,

상기 게스트 주소는 또한 게스트 물리 주소를 포함하는 가상화 기반 시스템.

청구항 33

제24항에 있어서,

상기 게스트 주소는 또한 게스트 선형 주소를 포함하는 가상화 기반 시스템.

청구항 34

제24항에 있어서,

상기 메모리는 또한 DRAM(Dynamic Random Access Memory)을 포함하는 가상화 기반 시스템.

명 세 서

발명의 상세한 설명

기 술 분 야

[0001] 관련 출원과의 교차 참조

[0002] 본 출원은, 본 발명(EPT 특허 출원)의 양수인에게 양도된 대리인 참조 번호가 P20462인 "Virtualizing Physical Memory in A Virtual Machine System"이라는 제목의 계류중인 미국 특허 출원 제11/036,736호와 관련된다.

배 경 기 술

[0003] 가상화(virtualization)는 가상화를 위한 하드웨어 및 소프트웨어 지원을 갖는 단일 호스트 머신이 호스트의 추상화를 제공하는 것을 가능하게 하며, 이에 따라 호스트 머신의 기초를 이루는 하드웨어가 하나 이상의 독립적으로 동작하는 가상 머신들로서 나타난다. 따라서, 각 가상 머신은 완비된 플랫폼으로서 기능할 수 있다. 종종, 가상화 기술을 이용하여, 다수의 게스트 오퍼레이팅 시스템들 및/또는 다른 게스트 소프트웨어가 동일한 하드웨어 플랫폼 상에서 실제로 물리적으로 실행하면서 다수의 가상 머신들 상에서 명백히 동시에 그리고 명백히 독립적으로 공존 및 실행할 수 있게 한다. 가상 머신은 호스트 머신의 하드웨어를 모방하거나 대안적으로 상이한 하드웨어 추상화를 함께 제공할 수 있다.

[0004] 가상화 시스템들은 호스트 머신을 제어하는 VMM(virtual machine monitor)을 포함할 수 있다. VMM은 가상 머신에서 동작하는 게스트 소프트웨어를 리소스들(예를 들어, 프로세서들, 메모리, IO 디바이스들)의 세트에 제공한다. VMM은 물리 호스트 머신의 컴포넌트들의 일부 또는 전부를 가상 머신에 맵핑할 수 있고, 가상 머신(예를 들어 가상 IO 디바이스들)에 포함되는, VMM 내의 소프트웨어로 에뮬레이트된 완전 가상 컴포넌트들을 생성할 수 있다. 따라서, VMM은 "가상 베어 머신(virtual bare machine)" 인터페이스를 게스트 소프트웨어에 제공한다고 할 수 있다. VMM은 하드웨어 가상화 아키텍처의 설비들을 이용하여 가상 머신에 서비스들을 제공하고 호스트 머신에서 실행중인 다수의 가상 머신들로부터 그리고 그 사이에 보호를 제공한다. 게스트 소프트웨어가 가상 머신에서 실행될 때, 게스트 소프트웨어가 하드웨어 플랫폼에서 직접 실행된다면, 게스트 소프트웨어에 의해 실행되는 특정 명령어들(예를 들어 주변 디바이스들에 액세스하는 명령어들)은 통상 하드웨어에 직접 액세스할 것이다. VMM에 의해 지원되는 가상화 시스템에서, 이러한 명령어들은, 본원에서 가상 머신 탈출(virtual machine exit)이라고 칭해지는 VMM으로의 이동을 야기할 수 있다. VMM은 소프트웨어의 이러한 명령어들을, 게스트 소프트웨어가 실행중인 가상 머신들과 양립하는 호스트 머신 하드웨어 및 호스트 머신 주변 디바이스들에 적합한 방식으로 처리한다. 마찬가지로, 호스트 머신에서 발생되는 특정 인터럽트들 및 예외들이 VMM에 의해 차단 및 관리되거나, 서비스를 위해 게스트 소프트웨어로 전달되기 전에 VMM에 의해 게스트 소프트웨어에 적응될 필요가 있을 수 있다. 그 후 VMM은 제어를 게스트 소프트웨어로 이동하고 가상 머신은 동작을 재개한다. VMM으로부터 게스트 소프트웨어로의 이동은 본원에서 가상 머신 진입(virtual machine entry)이라 칭해진다.

발명의 내용

해결 하고자하는 과제

[0005] 본 기술분야에 공지되어 있는 바와 같이, 페이지 테이블은 전형적인 프로세서 기반 시스템에서 선형 메모리로부터 물리 메모리로의 맵핑을 제공하기 위해 종종 사용된다. 페이지 테이블들은 일반적으로 메모리 상주 구조(memory-resident structure)들이기 때문에 선형 주소에 대응하는 물리 주소를 결정하기 위해 페이지 테이블에 액세스함으로써 메모리 액세스가 야기되고, 이는 처리 시간을 지연시킬 수 있다.

과제 해결수단

[0006] 이러한 문제점을 완화하기 위해, 많은 프로세서 구현들은 페이지 테이블 내의 값들에 기초하여, 사용중인 현재 선형-물리 메모리 맵핑들의 몇몇 서브세트가 캐싱되는 변환 색인 버퍼(translation lookaside buffer: TLB)라고 명명되는, 프로세서 내의 고속 메모리 또는 레지스터들의 뱅크를 포함한다. 이에 의해 프로세서는, 프로세서가 페이지 테이블에 액세스해야 하는 경우에 일반적으로 가능할 수 있는 것보다 선형 주소로부터 대응하는 물리 주소로의 변환에 보다 신속하게 액세스할 수 있다. 프로세서 구현들은 일반적으로, 페이지 테이블 내에 저장된 현재의 변환들에 기초하여 TLB 내의 모든 엔트리들을 무효화 또는 갱신하는 명령어들을 포함하여, TLB를 관리하는 명령어들을 제공한다.

효 과

[0007] 본 발명에 따르면, 프로세서는, 프로세서가 페이지 테이블에 액세스해야 하는 경우에 일반적으로 가능할 수 있는 것보다 선형 주소로부터 대응하는 물리 주소로의 변환에 보다 신속하게 액세스할 수 있다.

발명의 실시를 위한 구체적인 내용

[0008] 도 1: 도 1은 버스에 의해 프로세서에 통신 결합된 프로세서와 메모리를 포함하는 프로세서 기반 시스템에서 실행되는 프로세스를 도시한다. 도 1을 참조하면, 프로세스(105)가 선형 주소 공간(115)(선형 메모리 공간) 내의 메모리 위치(110)를 참조할 때, 머신(125)의 물리 메모리(145)(머신 물리 메모리) 내의 실제 주소(140)에 대한 참조가 메모리 관리(130)에 의해 생성되며, 이는 하드웨어(때때로 프로세서(120) 내로 통합됨) 및 소프트웨어(일반적으로 머신의 오퍼레이팅 시스템 내에 있음)로 구현될 수 있다. 메모리 관리(130)는, 다른 기능들 중에서 선형 주소 공간 내의 위치를 머신의 물리 메모리 내의 위치로 맵핑한다. 도 1에 도시된 바와 같이, 프로세스는 물리 머신에서 이용가능한 실제 메모리와 상이한 메모리 뷰를 가질 수 있다. 도 1에 도시된 예에서, 프로세스는, 메모리 관리 하드웨어 및 소프트웨어에 의해 그 자체가 10 내지 11 MB의 주소 공간을 갖는 물리 메모리의 일부로 실제 맵핑되는 0 내지 1 MB의 선형 주소 공간에서 동작하며, 프로세스 공간 주소로부터 물리 주소를 계산하기 위해, 오프셋(135)이 선형 주소에 추가될 수 있다. 선형 주소 공간으로부터 물리 메모리로의 보다 복잡한 맵핑들이 가능하며, 예를 들어, 선형 메모리에 대응하는 물리 메모리는 페이지들과 같은 부분들로 분할될 수 있고 물리 메모리 내의 다른 프로세스들로부터의 페이지들과 인터리브(interleave)될 수 있다.

[0009] 메모리는 관례상 페이지들로 분할되는데, 각 페이지는 공지된 양의 데이터를 포함하고, 구현마다 다르며, 예를 들어 하나의 페이지는 4096 바이트의 메모리, 1MB의 메모리, 또는 특정 애플리케이션에 대해 요구될 수 있는 임의의 다른 양의 메모리를 포함할 수 있다. 메모리 위치들이 실행중인 프로세스에 의해 참조될 때, 메모리 위치들은 페이지 참조들로 변환된다. 전형적인 머신에서, 메모리 관리는 선형 메모리 내의 페이지에 대한 참조를 머신 물리 메모리 내의 페이지로 맵핑한다. 일반적으로, 메모리 관리는 프로세스 공간 페이지 위치에 대응하는 물리 페이지 위치를 특정하기 위해 페이지 테이블을 사용할 수 있다.

[0010] 가상 머신 환경에서 게스트 소프트웨어를 관리하는 일 양태는 메모리의 관리이다. 가상 머신에서 실행중인 게스트 소프트웨어에 의해 취해진 메모리 관리 동작들을 처리하는 것은 가상 머신 모니터 등의 제어 시스템에 대한 복잡성을 야기한다. 예를 들어, x86 프로세서의 일부로서 구현되는 페이지 테이블들을 포함할 수 있는 x86 플랫폼 상에서 구현되는 호스트 머신에 대한 가상화를 통해 2개의 가상 머신들이 실행되는 시스템을 고려한다. 또한, 각 가상 머신 자체는 가상 머신에서 실행중인 게스트 소프트웨어에 x86 머신의 추상화를 제공한다고 가정한다. 각 가상 머신에서 실행중인 게스트 소프트웨어는 게스트 선형 메모리 주소에 대한 참조들을 형성할 수 있고, 게스트 선형 메모리 주소는 게스트 머신의 메모리 관리 시스템에 의해 게스트 물리 메모리 주소로 변환된다. 그러나, 게스트 물리 메모리 자체는 호스트 프로세서 상의 하드웨어에서의 가상화 서브 시스템 및 VMM을 통한 호스트 물리 메모리에서의 추가적인 맵핑에 의해 구현될 수도 있다. 따라서, 게스트 물리 메모리가 사실상 호스트 물리 메모리에 직접 대응하지 않고 호스트 머신의 가상화 시스템을 통해 더 재맵핑됨에 따라, 예를 들어 게스트 x86 페이지 테이블 제어 레지스터들에 대한 참조들을 포함하는, 게스트 프로세스들 또는 게스트 오퍼레이팅 시스템에 의한 게스트 메모리에 대한 참조들은, 추가의 재처리 없이 호스트 머신의 페이지 테이블로 직접 전달될 수 없기 때문에 VMM에 의해 차단되어야 한다.

[0011] 도 2: 도 2는 일 실시예에서 특히 게스트 메모리의 맵핑에 관하여 호스트 머신에서 실행중인 하나 이상의 가상 머신들 사이의 관계를 도시한다. 도 2는 게스트 물리 메모리가 호스트 머신의 가상화 시스템을 통해 재맵핑되는 방법을 도시한다. 가상 머신 A(242) 및 가상 머신 B(257) 등의 각 가상 머신은 가상 머신들에서 실행되는 게스트 소프트웨어에 각각 가상 프로세서(245 및 255)를 제공한다. 각 머신은 게스트 오퍼레이팅 시스템 또는 다른 게스트 소프트웨어, 게스트 물리 메모리들(240 및 250)에 물리 메모리의 추상화를 각각 제공한다. 게스트 소프트웨어는 가상 머신들(242 및 257)에서 실행되기 때문에, 실제로 호스트 물리 메모리(260)를 이용하여 호스트 프로세서(265)에서 호스트 머신(267)에 의해 실행된다.

[0012] 도 2에 도시된 바와 같이, 본 실시예에서는, 가상 머신 A(242) 내의 주소 0에서 시작하는 물리 메모리 공간으로서 제공되는 게스트 물리 메모리(240)는 호스트 물리 메모리(260) 내의 몇몇 연속 영역(contiguous region)(270)에 맵핑된다. 마찬가지로, 가상 머신 B(257) 내의 게스트 물리 메모리(250)는 호스트 물리 메모리(260)의 상이한 부분(275)에 맵핑된다. 도 2에 도시된 바와 같이, 호스트 머신은 1024 MB의 호스트 물리 메모리를 가질 수 있다. 각 가상 머신(242, 257)에 256 MB의 메모리를 할당하는 경우, 한가지 가능한 맵핑은, 가상

머신 A(242)에 128-384 MB 범위를 할당하고, 가상 머신 B(257)에 512-768 MB 범위를 할당하는 것이다. 두 가상 머신(242, 257)은 0-256 MB의 게스트 물리 주소 공간을 참조한다. 다만 VMM은 각 가상 머신의 주소 공간이 호스트 물리 주소 공간의 상이한 부분에 맵핑된다는 것을 알고 있다.

[0013] 도 2에 도시한 가상 머신 및 메모리 맵핑은 일 실시예의 일 표현일 뿐이고, 다른 실시예들에서는, 호스트 머신 상에서 실행되는 실제 가상 머신의 수는 하나에서부터 다수에 이르기까지 다를 수 있고, 호스트 머신과 가상 머신의 실제 메모리 사이즈도 다를 수 있으며 가상 머신마다 가변적일 수 있다. 이 예에서는 간단하고 연속적으로 가상 머신에 메모리를 할당하는 것을 도시하고 있다. 더 일반적인 경우, 가상 머신에 할당되는 물리 메모리 페이지들은 연속적이지 않을 수 있고 서로 인터리브된 호스트 물리 메모리에 분산될 수 있으며, 그 페이지들은 VMM과 기타 호스트 프로세스들에 속한다.

[0014] 도 2에 도시한 바와 같은 시스템에서 가상 머신으로서 제공되는 프로세서 기반 시스템은 매우 복잡하게 가상 머신을 구현할 수 있다. 그러므로, 예컨대, 가상 머신은 게스트 물리 메모리의 풀 뷰(full view)를 게스트 OS에 제공할 수 있고, 가상 머신의 게스트 OS 및 가상 프로세서 또는 기타 가상 하드웨어가 제공하는 메모리 관리를 이용하여, 가상 머신 상에서 실행되는 게스트 소프트웨어에 대한 메모리 관리를 행할 수 있다. 예시적인 일 실시예에 있어서, 가상 머신은 메모리 관리용 페이지 테이블과 같은 x86 하드웨어 지원을 포함하는 x86 플랫폼을 게스트 OS에 제공할 수 있고, 또한 메모리 관리용 x86 하드웨어를 포함하는 x86 플랫폼인 호스트 플랫폼 상에서 실제로 실행될 수 있다. 추가의 메커니즘 없이, 본 실시예의 가상화 시스템은, 한가지 가능한 솔루션으로서, 물리 메모리를 재맵핑, 분할 및 보호하는 x86 페이지 테이블 섀도잉(shadowing)을 이용하여 VMM에 물리 메모리 가상화 알고리즘을 구현해야만 한다. 따라서, 예컨대, 게스트 소프트웨어가 가상 머신의 x86 페이지 테이블에 대한 액세스를 시도하는 경우, VMM은 가상화에 필요한 기능(예컨대, 물리 주소의 재맵핑)을 게스트 OS가 필요로 하는 기능에 오버레이(overlay)해야 한다.

[0015] 이 때문에, VMM은 게스트 소프트웨어에 의한 페이징 메커니즘의 이용을 둘러싼 각종 이벤트를 트랩(trap)해야 한다. 이것은, x86 문서에 기재되어 있는 바와 같이, x86 메모리 관리 시스템의 제어 레지스터(예컨대, CR0, CR3, C4)와 같은 제어 레지스터에 대한 기입, 페이징 및 메모리 액세스와 연관된 모델 특정 레지스터(MSR)(예컨대, 메모리 타입 레인지 레지스터(memory-type range register: MTRR))에 대한 액세스, 어떤 예외(예컨대, 페이지 결함)의 처리를 포함한다. 물리 메모리를 가상화하기 위한 x86 페이지 테이블의 이러한 이용은 복잡하고 상당한 성능 오버헤드(overhead)를 가져온다.

[0016] 도 3: 도 3은 가상 머신 환경(300)의 일 실시예를 도시한다. 본 실시예에 있어서, 프로세서 기반 플랫폼(316)은 VMM(312)을 실행할 수 있다. VMM은, 통상적으로 소프트웨어로 구현되지만, 상위 레벨 소프트웨어에 대한 가상 베어 머신 인터페이스를 에뮬레이트(emulate) 및 익스포트(export)할 수도 있다. 그러한 상위 레벨 소프트웨어는 표준 OS, 실시간 OS를 포함하거나, 또는 어떤 실시예들에서는 오퍼레이팅 시스템 기능이 제한된 스트립트다운(stripped-down) 환경일 수 있고 표준 OS에서 통상적으로 이용 가능한 OS 설비를 포함하지 않을 수 있다. 대안적으로, 예컨대, VMM(312)은 다른 VMM 내에서 실행되거나, 다른 VMM의 서비스를 이용하여 실행될 수 있다. VMM은, 어떤 실시예들에서는 예컨대 하드웨어, 소프트웨어, 펌웨어 또는 각종 기술의 조합으로 구현될 수 있다. 적어도 하나의 실시예에 있어서, VMM의 하나 이상의 컴포넌트는 하나 이상의 가상 머신에서 실행될 수 있고 VMM의 하나 이상의 컴포넌트는 도 3에 도시한 베어 플랫폼 하드웨어 상에서 실행될 수 있다. 베어 플랫폼 하드웨어 상에서 직접 실행되는 VMM의 컴포넌트를 여기서는 VMM의 호스트 컴포넌트라고 한다.

[0017] 플랫폼 하드웨어(316)는 퍼스널 컴퓨터(PC), 서버, 메인프레임, PDA(personal digital assistant) 또는 "스마트" 모바일 폰과 같은 핸드헬드 장치, 포터블 컴퓨터, 셋톱 박스, 또는 다른 프로세서 기반 시스템일 수 있다. 플랫폼 하드웨어(316)는 적어도 프로세서(318)와 메모리(320)를 포함한다. 프로세서(318)는 마이크로프로세서, 디지털 신호 처리기, 마이크로컨트롤러 등과 같은, 프로그램을 실행할 수 있는 임의의 유형의 프로세서일 수 있다. 프로세서는 실시예들에서 실행을 위한 마이크로코드, 프로그램가능 로직 또는 하드 코드 로직을 포함할 수 있다. 도 3에서는 그러한 프로세서(318)가 단 하나인 것으로 도시하였으나, 실시예에서 시스템에 하나 이상의 프로세서가 있을 수 있다. 또한, 프로세서(318)는 다중 코어, 다중 스레드 서포트 등을 포함할 수 있다. 메모리(320)는 각종 실시예들로서 하드 디스크, 플로피 디스크, RAM, ROM, 플래시 메모리, 이들의 조합, 또는 프로세서(318)에 의해 판독 가능한 임의의 다른 유형의 머신 매체를 포함할 수 있다. 메모리(320)는 프로그램 실행 및 기타 방법 구현을 행하기 위한 명령어 및/또는 데이터를 저장할 수 있다. 어떤 실시예들에 있어서, 본 발명의 어떤 요소들은 다른 시스템 컴포넌트, 예컨대 플랫폼 칩셋 또는 시스템의 하나 이상의 메모리 컨트롤러에 구현될 수 있다.

[0018] VMM(312)은 동일 또는 상이한 추상화를 각종 게스트에 제공할 수 있는 하나 이상의 가상 머신의 추상화를 게스트 소프트웨어에 제공한다. 도 3은 2개의 가상 머신(302, 314)을 보여준다. 각 가상 머신 상에서 실행되는 게스트 소프트웨어(303, 313)와 같은 게스트 소프트웨어는 게스트 OS(304 또는 306)와 같은 게스트 OS와, 각종 게스트 소프트웨어 애플리케이션(308, 310)을 포함할 수 있다. 게스트 소프트웨어(303, 313)는 게스트 소프트웨어(303, 313)가 실행되고 있는 가상 머신 내의 물리적 리소스(예컨대, 프로세서 레지스터, 메모리 및 I/O 디바이스)에 액세스하고, 기타 기능을 행할 수 있다. 예컨대, 게스트 소프트웨어(303, 313)는 가상 머신(302, 314)에 제공된 프로세서 및 플랫폼의 아키텍처에 따라, 모든 레지스터, 캐시, 구조, I/O 디바이스, 메모리 등에 액세스할 것으로 기대한다.

[0019] 일 실시예에 있어서, 프로세서(318)는 가상 머신 제어 구조(virtual machine control structure: VMCS)(324)에 저장된 데이터에 따라 가상 머신(302, 314)의 동작을 제어한다. VMCS(324)는 게스트 소프트웨어(303, 313)의 상태, VMM(312)의 상태, 어떻게 VMM(312)이 게스트 소프트웨어(303, 313)의 동작을 제어하기를 원하는지를 나타내는 실행 제어 정보, VMM(312)과 가상 머신 간의 천이(transition)를 제어하는 정보 등을 포함할 수 있는 구조이다. 프로세서(318)는 VMCS(324)로부터 정보를 판독하여 가상 머신의 실행 환경을 판정하고 그 행동을 제한한다. 일 실시예에 있어서, VMCS(324)는 메모리(320)에 저장된다. 어떤 실시예들에 있어서, 다중 VMCS 구조를 이용하여, 하나 이상의 가상 다중 가상 머신 내의 CPU를 지원한다.

[0020] VMM(312)은 가상 머신(302, 314)에서 실행되는 게스트 소프트웨어에 의해 액세스 가능한 물리 메모리를 관리해야 할 수도 있다. 일 실시예에서 물리 메모리 관리를 지원하기 위해서, 프로세서(318)는 확장된 페이지 테이블(EPT) 메커니즘을 제공한다. 본 실시예에 있어서, VMM(312)은 가상 머신(302 또는 314)으로의 제어의 천이 전에 제공되어야 할 물리 메모리 가상화와 연관된 필드에 값을 제공하는 물리 메모리 관리 모듈(326)을 포함할 수 있다. 이들 필드를 EPT 제어로서 통칭한다. EPT 제어는 예컨대, EPT 메커니즘이 인에이블되어야 하는지를 나타내는 EPT 인에이블 지시자와, 물리 메모리 가상화 메커니즘의 형태 및 시멘틱스(semantics)를 나타내는 하나 이상의 EPT 테이블 구성 제어를 포함할 수 있다. 이들에 대해서는 후술한다. 또한, 일 실시예에 있어서, EPT 테이블(328)은 VMM(312)이 게스트 소프트웨어(303, 313)에 배치할 수 있는 물리 주소 변환 및 보호 시멘틱스를 나타낸다.

[0021] 일 실시예에 있어서, EPT 제어는 VMCS(324)에 저장된다. 대안적으로, EPT 제어는 프로세서(318), 메모리(320)와 프로세서(318)의 조합, 또는 임의의 다른 저장 장소(들)에 존재할 수도 있다. 일 실시예에 있어서, 각 가상 머신(302, 314)에 대해 별도의 EPT 제어가 유지된다. 대안적으로, 두 가상 머신에 대해 동일한 EPT 제어가 유지되고, 각 가상 머신 진입 전에 VMM(312)에 의해 갱신된다.

[0022] 일 실시예에 있어서, EPT 테이블(328)은 메모리(320)에 저장된다. 대안적으로, EPT 테이블(328)은 프로세서(318), 메모리(320)와 프로세서(318)의 조합, 또는 임의의 다른 저장 장소(들)에 존재할 수도 있다. 일 실시예에 있어서, 각 가상 머신(302, 314)에 대해 별도의 EPT 테이블(328)이 유지된다. 대안적으로, 두 가상 머신(302, 314)에 대해 동일한 EPT 테이블(328)이 유지되고, 각 가상 머신 진입 전에 VMM(312)에 의해 갱신된다.

[0023] 일 실시예에 있어서, 프로세서(318)는 EPT 인에이블 지시자에 기초하여 EPT 메커니즘이 인에이블되어 있는지를 판정하는 역할을 하는 EPT 액세스 로직(322)을 포함한다. EPT 메커니즘이 인에이블되어 있는 경우, 프로세서는 EPT 제어 및 EPT 테이블(328)에 기초하여 게스트 물리 주소를 호스트 물리 주소로 변환한다.

[0024] 도시한 실시예에 있어서, 프로세서는 선형-게스트 물리(linear to guest-physical), 게스트 물리-호스트 물리 주소(guest-physical to host-physical address) 및 선형-호스트 물리(linear to host-physical) 변환들을 캐싱하는 TLB(323)를 더 포함할 수 있다. 선형-게스트 물리 및 선형-호스트 물리 변환들을 여기서는 "선형 변환들"이라고 한다. 게스트 물리-호스트 물리 및 선형-호스트 물리 변환들을 여기서는 "물리 변환들"이라고 한다.

[0025] 일 실시예에 있어서, 시스템(300)은 다중 프로세서들 또는 멀티-쓰레드 프로세서들을 포함하며, 로직 프로세서들의 각각은 별도 EPT 액세스 로직(322)과 연관되어 있으며, VMM(312)은 EPT 테이블들(328)을 구성하며 EPT는 로직 프로세서들의 각각에 대해 제어한다.

[0026] 게스트 소프트웨어(예컨대, 게스트 OS(304) 및 애플리케이션(308)을 포함하는 도면번호 303)에 의해 액세스가능한 리소스들은 "특권(privileged)" 또는 "비특권(non-privileged)"으로 분류될 수 있다. 특권 리소스들에 대하여, VMM(312)은 게스트 소프트웨어에 의해 요망되는 기능성을 용이하게 하면서, 이들 특권 리소스들을 통해 궁극적인 제어를 유지한다. 또한, 각 게스트 소프트웨어(303, 313)는 예외들(예컨대, 페이지 결함들, 일반적인 보호 결함들 등), 인터럽트들(예컨대, 하드웨어 인터럽트들, 소프트웨어 인터럽트들), 및 플랫폼 이벤트들(예컨

대, 초기화(INIT) 및 시스템 관리 인터럽트들(SMI들))과 같은 각종 플랫폼 이벤트들을 처리할 것으로 기대한다. 이들 플랫폼 이벤트들 중 일부는, 게스트 소프트웨어로부터 그리고 게스트 소프트웨어 사이에서의 보호를 위해 가상 머신들(302, 314)의 적절한 동작을 보장하기 위해 VMM(312)에 의해 조작되어야 하기 때문에 "특권" 이벤트이다. 게스트 오퍼레이팅 시스템 및 게스트 애플리케이션들 양자 모두 특권 리소스들에 대한 액세스를 시도할 수 있으며, 양자 모두 특권 이벤트들을 야기하거나 경험할 수 있다. 본 명세서에서는, 특권 리소스들에 대한 액세스 시도들 및 특권이 부여된 플랫폼 이벤트들을 "특권 이벤트들(privileged events)" 또는 "가상화 이벤트들(virtualization events)"로서 통칭한다.

[0027] 도 3a: 도 3a는 도 3의 실시예에서의 프로세서(318)의 일부 블록 레벨 특징을 하이 레벨로 나타낸다. 일반적으로, 도 3에서 도면번호 318로 도시된 것과 같은 프로세서는 도 3a에서 도면번호 337로 도시된 것과 같은 프로세서 버스 또는 버스들을 포함할 수 있다. 또한, 도 3a에 도시된 바와 같이, 프로세서는 하나 또는 다수의 뱅크들의 레지스터들(350)을 포함할 수 있으며, 각 레지스터는 32, 64, 128 또는 공지된 다른 수의 비트의 데이터를 저장하는 용량을 가질 수 있다. 각 레지스터 뱅크는, 예컨대 8, 32, 64 레지스터들과 같은 몇몇 레지스터들을 더 가질 수 있다. 일부 레지스터들은, 예를 들어 x86 실시예에서와 같이 CR 비트를 저장하기 위해 제어 및 상태 사용에 대해 전용일 수 있다. 다른 실시예에서, 프로세서에는 이 기술분야에 공지된 바와 같이 서로 다른 모드의 동작 및 상태 검사를 허용하기 위해 다른 제어 레지스터들 및 플래그들이 저장될 수 있다. 일반적으로, 도 3의 실시예에 도시된 것과 같은 프로세서는, 메모리, 캐시 또는 다른 기억 장치로부터 명령어들 및 데이터를 페치(fetch)하는 로직 또는 로직 회로(330), 명령어들을 디코딩하는 로직 또는 로직 회로, 및 명령어들을 실행하는 도면번호 334와 같은 실행 유닛들을 포함한다. 이들 기능 유닛들에 대한 수많은 변경들이 가능하며, 예를 들어 실행 유닛에서의 실행은 파이프라인되거나, 추론(speculation) 및 분기 예측을 포함하거나, 특정 프로세서나 애플리케이션과 관련된 다른 특징들을 가질 수 있다. 프로세서에는 산술, 그래픽 처리, 및 공지된 프로세서의 다른 수많은 특정 기능들을 위한 로직과 같은 기타 기능 로직(365)이 제공될 수 있다. 일부 실시예에는 온보드 캐시(on-board cache)(360)가 제공될 수 있다. 이 캐시는 공지된 바와 같이 128MB, 1GB 등의 다양한 크기를 가질 수 있다. 도 3을 참조하여 전술한 바와 같이, 프로세서(318)는 EPT 액세스 로직(322) 및 TLB(323)를 포함한다. EPT 액세스 로직은 일 실시예에서 EPT를 실장(populate), 제어, 및 관리하는 로직을 포함할 수 있으며, TLB는 일반적으로 프로세서 내에서 효율성 및 다른 목적으로 캐싱되는 페이지 테이블들로부터의 맵핑들을 포함하는 버퍼이다.

[0028] 도 4: 도 4는 가상 머신에서의 게스트 소프트웨어가 게스트 가상 주소를 참조하는 경우에 최종적으로 호스트 물리 주소를 계산하기 위해 전술한 확장 페이지 테이블들을 이용하는 처리의 일례를 나타낸다. 도시된 예는 단순한 32 비트 가상 주소화 및 단순한 페이지 테이블 포맷들을 이용하는 x86 플랫폼에서 실행되는 게스트 소프트웨어를 나타낸다. 당업자라면, 예를 들어, 다른 페이징 모드들(예컨대, 게스트 소프트웨어에서의 64비트 주소화), 다른 명령어 세트 아키텍처들(예컨대, 특히, Intel Itanium® 아키텍처, 64 비트 및 그외 변형들의 x86 아키텍처, PowerPC® 아키텍처) 및 다른 구성들을 이해하기 위해 본 예를 용이하게 확장할 수 있을 것이다.

[0029] 도 4에서, 가상 머신에서 실행되는 게스트 소프트웨어에 의해 게스트 가상 주소(410)에 대한 참조가 실행된다. 게스트(즉, 게스트 오퍼레이팅 시스템에 의해 구성되는)에서 활성화되는 메모리 관리 메커니즘은 가상 주소를 게스트 물리 주소로 변환하는 데 이용된다. 변환에 사용되는 각 게스트 물리 주소 및 결과로서의 게스트 물리 주소는 호스트 물리 메모리를 액세스하기 전에 EPT를 통해 호스트 물리 주소들로 변환된다. 이 처리는 이하 논의에서 상술한다.

[0030] 본 예에서, CR3 레지스터(420) 내의 적절한 비트(402)는 게스트 물리 메모리 내의 게스트의 페이지 디렉토리 테이블(PD 테이블)(460)의 베이스(base)를 가리킨다. 이 값(402)은 (본 예에서는 테이블들 내의 엔트리들이 각각 4 바이트이므로 4로 승산함으로써, x86 시멘틱스에 따라 적절히 조정된) 게스트 가상 주소(410)로부터 상위 비트들과 결합하여, 게스트의 PD 테이블(460) 내의 페이지 디렉토리 엔트리(PDE)의 게스트 물리 주소(412)를 형성한다. 이 값(412)은 EPT 테이블들(455)을 통해 변환되어 페이지 디렉토리 엔트리의 호스트 물리 주소(404)를 형성한다. 프로세서는 이 호스트 물리 주소(404)를 이용하여 페이지 디렉토리 엔트리에 액세스한다.

[0031] PDE로부터의 정보는 게스트의 페이지 테이블(470)의 베이스 주소(422)를 포함한다. 이 게스트 물리 주소(422)는 적절히 조정된 게스트 가상 주소(410)의 비트들 21:12와 결합하여 게스트의 페이지 테이블(470) 내의 페이지 테이블 엔트리의 게스트 물리 주소(432)를 형성한다. 이 게스트 물리 주소(432)는 EPT 테이블(465)을 통해 변환되어 게스트의 페이지 테이블 엔트리(PTE)의 호스트 물리 주소(414)를 형성한다. 프로세서는 이 호스트 물리 주소(414)를 이용하여 PTE에 액세스한다.

[0032] PTE로부터의 정보는 액세스되는 게스트 물리 메모리 내의 페이지의 베이스 주소(442)를 포함한다. 이 값은 게스트 가상 주소(410)의 하위 차수 비트들(11:0)과 결합하여, 액세스되는 메모리의 게스트 물리 주소(452)를 형성한다. 이 값(452)은 EPT 테이블(475)을 통해 변환되어 액세스되는 메모리의 호스트 물리 주소(424)를 형성한다.

[0033] EPT 테이블들이 게스트 물리 주소를 호스트 물리 주소로 변환하는 데 사용될 때마다, 프로세서는, 후술하는 바와 같이, 액세스가 EPT 테이블들에서의 제어들에 따라 허용됨을 유효하게 한다. 또한, EPT 테이블(455, 465, 475)은, 도 4에 구별하여 나타내었지만, 일 실시예에서 동일한 세트의 EPT 테이블들일 수 있다(즉, 게스트 물리로부터 호스트 물리로의 모든 주소 변환을 위해 단일 세트의 EPT 테이블들이 이용된다)는 것을 이해해야 한다.

[0034] 프로세서 기반 시스템에서의 선형 메모리 지원의 전형적인 구현에 있어서, 페이지 테이블 구조에 저장되어 있는 선형 주소들로부터 물리 주소들로의 맵핑들은 효율상의 이유로 TLB에 캐싱될 수 있다. 프로세서 명령어 세트에는, TLB를 관리하고, 프로세서 기반 시스템에서 실행되는 프로그램이 TLB 내의 특정 엔트리가 페이지 테이블 엔트리와 동기화되는 것을 보장할 수 있게 하는 명령어들이 포함될 수 있다. 따라서, 예를 들어, x86 아키텍처에는, MOV CR 명령어가 모든 TLB 엔트리의 글로벌 무효화(global invalidation)를 일으킬 수 있고, 이에 따라 주소들이 액세스될 때 엔트리들의 재동기화를 일으킬 수 있다. 대안적으로, x86의 예에서, 특정 선형 주소에 대해 TLB 내에 저장되는 맵핑을 무효화시키는 데에 INVLPG 명령어가 사용됨으로써, TLB 내의 엔트리가 갱신되고 페이지 테이블 내의 맵핑과 동기화되게 된다.

[0035] 앞서 논의된 바와 같이, 확장 페이지 테이블(EPT)을 통합하는 가상화 시스템을 포함하는 일 실시예에 있어서, TLB가 게스트 머신 메모리에서 실행되는 프로세스들에 대한 게스트 선형-호스트 물리 주소 변환들, 및 도 3의 도면부호 323을 참조하여 앞서 설명한 바와 같은, 호스트 머신 상에서 직접 실행되는 VMM 등의 프로세스들에 대해 호스트 선형-호스트 물리 맵핑들을 캐싱할 수 있다. 전자의 경우, 게스트 선형-호스트 물리 맵핑들은 EPT로부터 뿐만 아니라 게스트 내의 페이지 테이블들로부터도 얻어질 수 있으며, 후자의 경우, 맵핑들은 호스트 페이지 테이블들로부터 얻어질 수 있다. EPT에 저장되어 있는 맵핑들에 기초하여 게스트 물리-호스트 물리 메모리로부터 직접 얻어진 맵핑과 같은, 추가 유형의 맵핑이 또한 TLB에 저장될 수 있다.

[0036] 일 실시예에서, 새로운 커맨드가 프로세서 명령어 세트에 추가된다. 본 실시예에서, 새로운 커맨드 INVL_EPT는, 게스트 물리-호스트 물리 맵핑들로부터 얻어진 TLB 엔트리들을 관리하는 방식으로, VMM 등의 가상화 시스템의 호스트 머신 상에서 직접 실행되는 프로그램들을 제공한다. 구체적으로, 본 실시예에서는, INVL_EPT 명령어는 TLB에서의 게스트 물리-호스트 물리 및 선형-호스트 물리 맵핑들이 호스트 메모리 내에 상주하는 EPT 테이블들과 동기화됨을 보장하며, 동기화 범위, EPT 콘텍스트(context), 및 관련된 경우, 맵핑들이 동기화되어야 하는 게스트 물리 메모리 주소를 특정한다. 일반적으로 말해서, 콘텍스트는 시스템의 주소 공간의 일부를 정의한다. 게스트 물리-호스트 리 맵핑들에 대하여, EPT 콘텍스트는 현재 활성인 EPT 테이블에 의해 정의되며, 이는 본 실시예에서 EPT 포인터 또는 EPTP라는 용어의 레지스터로 참조된다.

[0037] 본 실시예에서, INVL_EPT 명령어는 3개의 피연산자(operand)를 가지며, 제1 피연산자는 명령어 모드 또는 변형 명세(variant specification)에 대한 값이며, 제2 피연산자는 INVL_EPT 명령어가 실행되는 EPT 콘텍스트에 등가인 EPT 포인터를 지정하는 값이며, 제3 피연산자는 무효화될 TLB 엔트리들과 연관된 게스트 물리 주소를 지정하는 값이다. 본 실시예에서, 제1 피연산자는 8비트 중간 값으로서 제공되며, 제2 및 제3 피연산자는 메모리 내에 블록으로서 제공되며, 각각 64비트를 차지한다. 다른 실시예들도 가능하다. 예를 들어, 피연산자들은 레지스터 또는 그 밖의 메모리 위치에 명시적으로 또는 암시적으로 제공될 수 있다.

[0038] 본 실시예에서의 제1 피연산자는 적어도 3개의 피정의 값(defined value)을 갖는 스위치 또는 플래그이며, 따라서 INVL_EPT 명령어는 가능한 3가지 모드 중 하나로 실행하는 것임을 지정한다.

[0039] 1. 개별 주소 모드: 이 모드에서는, 단일 게스트 물리 주소와 연관된 TLB 내의 물리 변환들은 전술한 제2 피연산자에 제공된 콘텍스트에 의해 참조되는 EPT 내의 그 주소에 대한 맵핑들에 기초하여, EPT와 동기화된다.

[0040] 2. 콘텍스트 모드: 이 모드에서는, 게스트 주소 파라미터(전술한 제3 피연산자)가 무시되고, 전술한 제2 피연산자에 지정된 EPT 콘텍스트 내의 TLB 내의 엔트리들이 EPT와 동기화된다.

[0041] 3. 글로벌 모드: 이 모드에서는, 게스트 주소 파라미터 및 EPT 콘텍스트 파라미터 양자 모두가 무시되고, 임의의 EPT 콘텍스트로부터 얻어진 TLB 엔트리들이 동기화된다.

[0042] 도 5: 도 5의 플로우차트는 일 실시예에서의 INVL_EPT 명령어의 실행을 나타낸다. 실행은 단계 500에서 시작한

다. 먼저, 프로세서는 현재의 실행 환경이 EPT 관련 연산들에 대해 유효함을 보장하기 위해, 단계 505에서 많은 테스트를 수행할 수 있다. 이 테스트들은 시스템이 가상화 연산 모드에 있으며; 페이징이 인에이블(enable)되어 있으며, 그 중에서도 특히, 어떠한 현재 에러 상태도 없음을 보장하기 위한 테스트를 포함할 수 있다. 실행 환경이 유효하지 않으면, 명령어는 단계 555 및 560을 통해 정의되지 않은 상태로 나가거나, 일반적인 보호 결함을 생성하거나, 정의되지 않은 연산 부호(opcode) 결함을 생성할 수 있다. 실행 환경이 유효하면, 단계 510에서 명령어 실행은 본 실시예에서 즉시 8비트 피연산자(immediate 8 bit operand)를 판독한다. SYNC_CMD라는 용어의 이 피연산자는 앞서 상세히 설명된 INVL_EPT 명령어에 대한 유효 모드를 나타낼 것으로 예상된다. 피연산자가 유효하지 않으면, 단계 515에서 명령어는 이전과 같이 단계 555 및 560으로 간다. 피연산자가 유효하면, 단계 520에서 메모리로부터 제2 및 제3 피연산자들을 판독하는 실행이 진행되며, 앞서 설명된 바와 같이, 피연산자들은 도면에서의 INVLEPT_DESC로 표시된 참조에서 메모리의 128 비트 블록에 있을 것으로 예상된다. 본 실시예에서, 처음 64 비트는 EPT 콘텍스트 또는 EPT 테이블 포인터이고, 단계 525에서 EPTP_CTX로서 추출되며, 다음 64 비트는 단계 530에서 GP_ADDR로서 추출된 명령어에 대한 게스트 물리 주소 파라미터이다.

[0043] 이어서, 명령어의 실행은 단계 535 내지 580에 도시된 실행 흐름에서 SYNC_CMD 피연산자의 값에 의존하여 실제 동기화를 실행하도록 진행한다. 앞서 설명한 바와 같이, SYNC_CMD는 모든 EPT 콘텍스트에 기초하여 TLB의 글로벌 동기화를 수행하라는 지시, 또는 그 명령어의 피연산자에 의해 지정된 EPT 콘텍스트만의 동기화를 수행하라는 지시, 또는 마지막으로, 파라미터로서 제공된 EPT 콘텍스트에서 파라미터로서 전달된 게스트 물리 주소만의 동기화를 수행하라는 지시일 수 있다. 본 실시예에서, 도 5의 실행 흐름에 도시된 바와 같이, 단계 535에서 SYNC_CMD 값의 확인이 일어나고, 단계 540에서 두번째, 단계 545에서 세번째로 일어난다. 단계 535에서, SYNC_CMD의 값이 글로벌 동기화를 지시하면, 모든 EPT 콘텍스트들에 대한 모든 물리 맵핑들을 동기화하라는 명령어가 실행되어, 단계 580에서 실행이 완료된다. 그렇지 않고, SYNC_CMD가 단계 540에서 콘텍스트 특정 동기화를 지시하면, 제공된 EPTP_CTX 값이 유효한지를 확인하라는 실행이 진행된다. 예를 들어, 그 값 내에 예비 비트(reserved bit)가 설정되거나 EPTP 내의 주소가 무효이기 때문이 아니라면, 그 명령어의 실행은 단계 555 및 560에서 일반적인 보호 결함으로 종료된다. 제공된 콘텍스트가 유효하면, 단계 575에서 EPTP_CTX에 의해 참조되는 EPT 콘텍스트에 대한 모든 물리 맵핑들을 동기화시키는 실행을 계속하고, 이어서 단계 580에서 완료한다.

[0044] SYNC_CMD가 글로벌 동기화도 콘텍스트 와이드 동기화(context wide synchronization)도 아니고, 또한 유효 연산 모드이면, 본 실시예에서의 남아있는 유일한 가능성은 커맨드가 특정 게스트 물리 주소를 동기화하는 것이다. 이어서 단계 545에서 제공된 GP_ADDR 파라미터가 유효한지에 대한 확인을 실행한다. 무효 주소가 제공되면, 실행은 단계 555 및 560에서 일반적인 보호 결함으로 종료된다. 그렇지 않으면, 제공된 게스트 물리 주소 GP_ADDR와 연관된 모든 물리 맵핑들이, 단계 550에서 EPTP_CTX에 제공된 콘텍스트에 의해 참조되는 EPT와 동기화되며, 단계 580에서 실행을 완료한다.

[0045] 전술한 실시예들은 광범위하게 변경될 수 있다는 것이 본 기술분야의 당업자에게 명백할 것이다. 일부 실시예들에서, INVL_EPT와 등가인 커맨드가 이용 가능할 수 있으나, 그 중에서도 특히 파라미터의 사이즈, 이름, 수 및 포맷을 포함하여, 상이한 신택스(syntax)를 가질 수 있다. 공지되어 있는 바와 같이, 상이한 명령어 세트 아키텍처(instruction set architecture: ISA)가 존재하며, 유사한 커맨드가 포맷이 상이한 ISA 및 그 ISA와 일관되는 그 밖의 특징들에 제공될 수 있다. 일례로서, 인텔® Itanium 아키텍처에 기초하여 프로세서를 위한 EPT와 TLB를 무효화 및/또는 동기화하는 명령어는, 임의의 다른 ISA에 대한 명령어들이 그럴 수 있는 것처럼, 이 기술 분야의 당업자에 의해 앞서 제공된 실시예들의 설명에 기초하여 용이하게 시각화되고 설명될 수 있다.

[0046] 상기 언급된 실시예들에서의 EPT 콘텍스트에 관한 논의는 한정적인 것으로 인식되어서는 안 된다. 다른 실시예들에서, EPT의 하나의 예(instance)만이 있을 수 있고, 이외에, x86 예에서 논의된 바와 같이, 몇몇 예들이 상기에서 논의된 EPTP와 유사한 참조 레지스터 또는 포인터와 같은 참조 메커니즘으로 동작할 수 있다.

[0047] 다른 실시예들에서, 파라미터들의 수 및 포맷은 다양할 수 있다. 예를 들어, 전술한 실시예들에서, INVL_EPT 명령어는 1개의 즉시(immediate) 및 2개의 메모리 기반 피연산자들을 갖는다. 다른 실시예들에서, 더 많은 즉시 피연산자들이 사용될 수 있고, 다른 실시예들에서 모든 피연산자들이 메모리 기반일 수 있고; 또 다른 실시예들에서, 공지된 많은 다른 변경들 중에서, 피연산자들은 레지스터들 또는 프로세서 내의 다른 저장소로부터 판독될 수 있다.

[0048] 전술한 실시예들은 INVL_EPT 명령어에 대한 3가지 연산 모드를 참조하여 설명된다. 다른 실시예들에서, 이들 모드 중 일부 또는 전부가 빠질 수 있고, 다른 실시예들에서, 더 많은 모드들이 이용가능할 수 있다. 예를 들

어, 일부 실시예들에서, 개별 주소 무효화에 대한 모드가 없을 수 있고, 그러한 모드에서, 모든 TLB 엔트리들은 동기화될 것이다. 일부 실시예들에서는, 시스템에서 동작하는 EPT의 하나의 예만이 있을 수 있고, 그러한 실시예들에서, 콘텍스트 모드는 불필요할 수 있다. 대안적으로, 일부 실시예들에서, 개별 주소 동기화만이 이용가능할 수 있거나, 다른 실시예들에서, 글로벌 주소 동기화만이 사용될 수 있어, INVL_EPT를 참조하여 기술된 제1 피연산자를 불필요하게 할 수 있다.

[0049] 명령어 및 그 연산에 대한 이러한 변경들이 가능하지만, 많은 다른 것들 중에서, 다른 명령어들의 조합에 의해 IVNL_EPT 명령어의 일반적인 효과가 획득되는 변경들을 포함하여, 많은 다른 것들이 이 기술 분야의 당업자에 의해 쉽게 파악될 수 있다.

[0050] 전술한 설명에서는, 설명의 목적으로, 설명된 실시예들의 완벽한 이해를 제공하기 위하여 많은 특정 상세들이 설명되었지만, 이 기술분야의 당업자라면 많은 다른 실시예들이 이들 특정 상세들 없이도 실행될 수 있다는 것을 이해할 것이다.

[0051] 전술한 상세한 설명의 일부 부분들은 프로세서 기반 시스템 내의 데이터 비트들에 대한 연산의 알고리즘 및 기호 표현에 대해서 제공된다. 이들 알고리즘적 기술 및 표현들은 이 기술분야의 당업자들에 의해 그들의 연구의 내용을 이 기술 분야의 다른 당업자들에게 가장 효과적으로 전달하기 위해 사용되는 수단이다. 연산들은 물리량의 물리 조작을 필요로 하는 것들이다. 이 양들은 저장, 전송, 결합, 비교 및 달리 조작될 수 있는 전기적, 자기적, 광학적 또는 다른 물리 신호들의 형태를 취할 수 있다. 주로 공통 사용의 이유로, 이러한 신호들을 비트들, 값들, 요소들, 기호들, 문자들, 용어들, 숫자들 등으로서 지칭하는 것은 때때로 편리하다는 것이 입증되었다.

[0052] 그러나, 이들 및 유사한 용어들 모두가 적절한 물리량들과 연관되어야 하고, 이들은 그러한 양들에 적용되는 편리한 라벨들일 뿐이라는 것에 주목해야 한다. 설명으로부터 명백한 바와 같이 특별하게 달리 언급되지 않는다면, "실행" 또는 "처리" 또는 "컴퓨팅" 또는 "계산" 또는 "판정" 등의 용어들은, 프로세서 기반 시스템, 또는 프로세서 기반 시스템의 저장장치 내에 물리량으로서 표현되는 데이터를 유사하게 표현되는 다른 데이터로 조작 및 변환하는 유사한 전자 컴퓨팅 장치, 또는 그러한 다른 정보 저장장치, 전송 또는 표시 장치들의 동작 및 처리들을 지칭할 수 있다.

[0053] 실시예들의 설명에서, 첨부 도면들에 대한 참조가 행해질 수 있다. 도면들에서, 유사한 도면번호들은 몇몇 도면에 걸쳐 실질적으로 유사한 구성요소들을 설명한다. 다른 실시예들이 사용될 수 있고, 구조적, 논리적, 및 전기적 변경들이 행해질 수 있다. 또한, 서로 상이하지만 다양한 실시예들은 반드시 서로 배타적일 필요는 없다는 것이 이해되어야 한다. 예를 들어, 일 실시예에서 설명된 특정한 특징, 구조, 또는 특성은 다른 실시예들 내에 포함될 수 있다.

[0054] 또한, 프로세서에 구현된 실시예의 디자인은 창작으로부터 시뮬레이션 및 제조에 이르기까지 다양한 단계들을 거칠 수 있다. 디자인을 나타내는 데이터는 많은 방법으로 디자인을 표현할 수 있다. 우선, 시뮬레이션에서 유용한 바와 같이, 하드웨어는 하드웨어 기술 언어(hardware description language) 또는 다른 기능적 기술 언어(functional description language)를 이용하여 표현될 수 있다. 부가적으로, 로직 및/또는 트랜지스터 게이트들을 가진 회로 레벨 모델은 디자인 프로세스의 일부 단계들에서 만들어질 수 있다. 또한, 대부분의 디자인들은, 일부 단계에서, 하드웨어 모델의 다양한 장치들의 물리적 배치를 나타내는 데이터의 레벨에 도달한다. 종래의 반도체 제조 기술들이 사용되는 경우, 하드웨어 모델을 나타내는 데이터는 집적 회로를 만드는 데 사용되는 마스크에 대한 상이한 마스크 층들에 다양한 특징들의 존재 및 부재를 특정하는 데이터일 수 있다. 디자인의 임의의 표현에서, 데이터는 임의의 형태의 머신 판독 가능한 매체에 저장될 수 있다. 그러한 정보를 전송하기 위하여 변조되거나 달리 생성되는 광 또는 전기파, 메모리, 또는 디스크와 같은 자기 또는 광 저장장치는 머신 판독 가능한 매체일 수 있다. 이들 매체들의 임의의 것은 디자인 또는 소프트웨어 정보를 "운반" 또는 "지시"할 수 있다. 코드 또는 디자인을 지시 또는 운반하는 전기적 반송파가 전송되는 경우, 전기 신호의 복사, 버퍼링, 또는 재전송이 수행되는 한에서, 새로운 복사가 행해진다. 따라서, 통신 제공자 또는 네트워크 제공자는 실시예를 구성 또는 표현하는 물품(반송파)의 복사본들을 만들 수 있다.

[0055] 실시예들은, 머신에 의해 액세스될 때 그 머신이 청구된 요지에 따른 처리를 실행하게 할 수 있는 데이터를 저장한 머신 판독 가능한 매체를 포함할 수 있는 프로그램 제품으로서 제공될 수 있다. 머신 판독 가능한 매체는 플로피 디스켓, 광 디스크, DVD-ROM 디스크, DVD-RAM 디스크, DVD-RW 디스크, DVD+RW 디스크, CD-R 디스크, CD-RW 디스크, CD-ROM 디스크 및 자기-광 디스크, ROM, RAM, EPROM, EEPROM, 자기 또는 광 카드, 플래시 메모리, 또는 전자 명령어들을 저장하는데 적절한 다른 타입의 매체/머신 판독 가능한 매체를 포함할 수 있지만, 이

에 한정되는 것은 아니다. 또한, 실시예들은 프로그램 제품으로서 다운로드될 수 있고, 여기서 프로그램은 반송파에 포함된 데이터 신호 또는 통신 링크(예를 들어, 모뎀 또는 네트워크 접속)를 통한 다른 전파 매체(propagation medium)에 의해 원격 데이터 소스로부터 요청 장치로 전달될 수 있다.

[0056] 많은 방법이 그들의 가장 기본적인 형태로서 기술되었지만, 청구된 요지의 기본적인 범위를 벗어나지 않고, 임의의 방법들로부터 단계들이 추가되거나 삭제될 수 있고, 임의의 기술된 메시지들로부터 정보가 추가되거나 삭제될 수 있다. 이 기술 분야의 당업자들에게는 다른 많은 변경 및 수정이 행해질 수 있다는 것이 명백할 것이다. 특정 실시예들은 청구된 요지를 한정하기 위하여 제공된 것이 아니라 그것을 예시하기 위하여 제공된 것이다. 청구된 요지의 범위는 상기에서 제공된 특정 실시예들에 의해 결정되는 것이 아니라 하기의 청구범위에 의해서만 결정된다.

도면의 간단한 설명

[0057] 도 1은 프로세스와 물리 메모리 사이의 관계를 도시하는 도면.

[0058] 도 2는 일 실시예에서 가상 머신들과 호스트 머신 사이의 관계를 개략적으로 도시하는 도면.

[0059] 도 3은 일 실시예에서 가상 머신 환경의 고급 구조를 도시하는 도면.

[0060] 도 3a는 일 실시예에서 프로세서를 기능 레벨로 나타내는 도면.

[0061] 도 4는 일 실시예에서 확장 페이지 테이블들을 이용하는 주소 계산을 도시하는 도면.

[0062] 도 5는 일 실시예에서 명령어 실행의 흐름을 도시하는 도면.

[0063] <도면의 주요 부분에 대한 부호의 설명>

[0064] 120, 318: 프로세서

[0065] 303, 313: 게스트 소프트웨어

[0066] 302, 314: 가상 머신 추상화

[0067] 312: 가상 머신 모니터(VMM)

[0068] 316: 베어 플랫폼 하드웨어

[0069] 320: 메모리

[0070] 322: EPT 액세스 로직

[0071] 326: 물리 메모리 관리 모듈

[0072] 328: EPT 테이블들

[0073] 336: 버스

[0074] 355: 제어 및 상태 레지스터들

[0075] 350: 레지스터들

[0076] 330: 페치 로직

[0077] 332: 디코드 로직

[0078] 334: 실행 로직

[0079] 322: EPT 액세스 로직

[0080] 365: 기타 기능 로직

[0081] 360: 온보드 캐시

[0082] 420: CR3 제어 레지스터

[0083] 424: 호스트 물리 주소

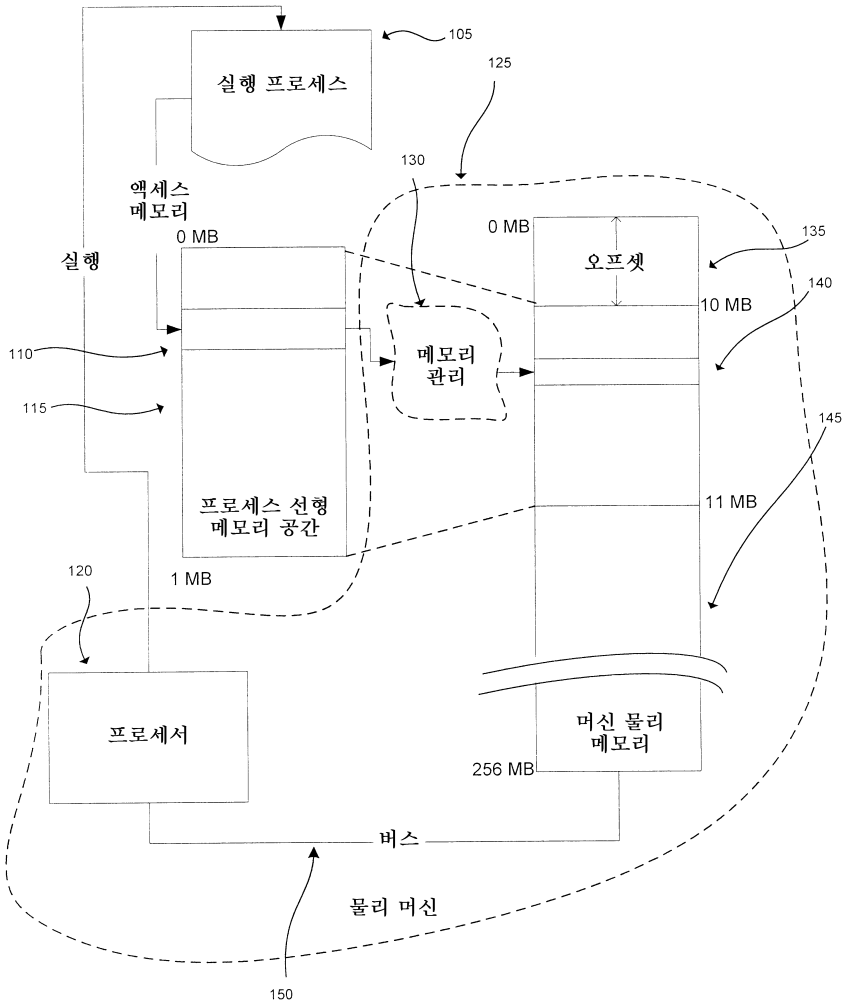
[0084] 455, 465, 475: EPT

[0085] 460: PD 테이블

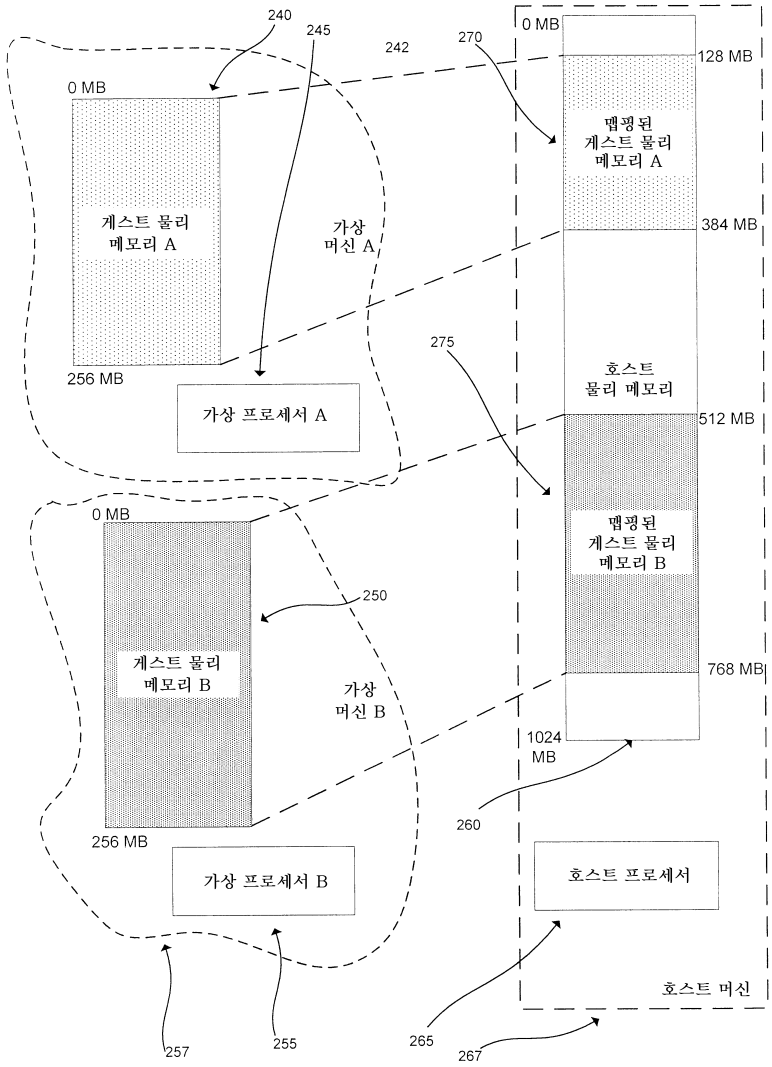
[0086] 470: 페이지 테이블

도면

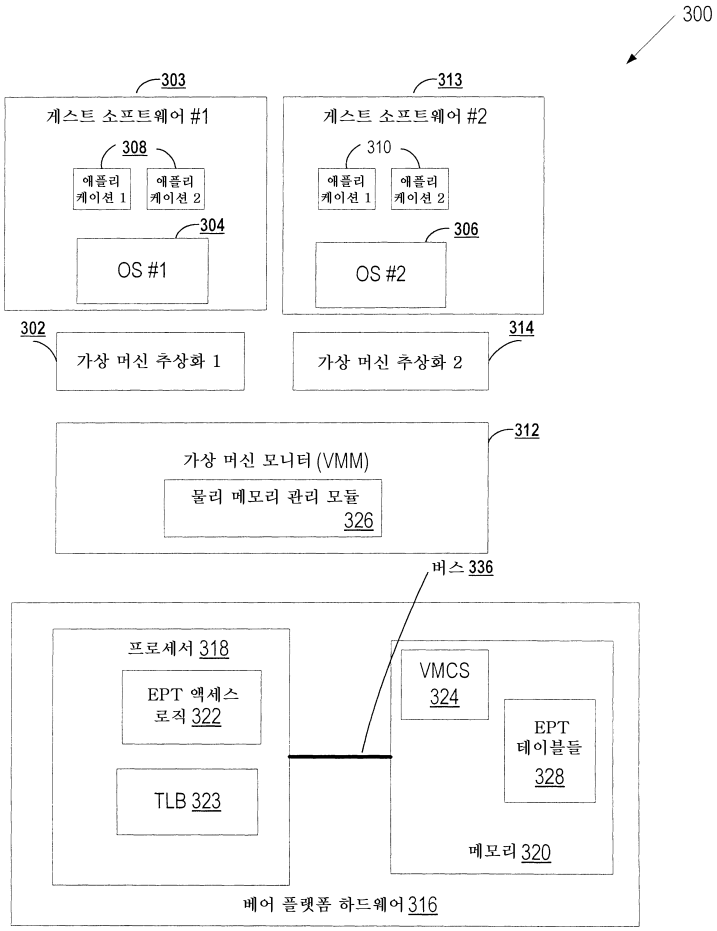
도면1



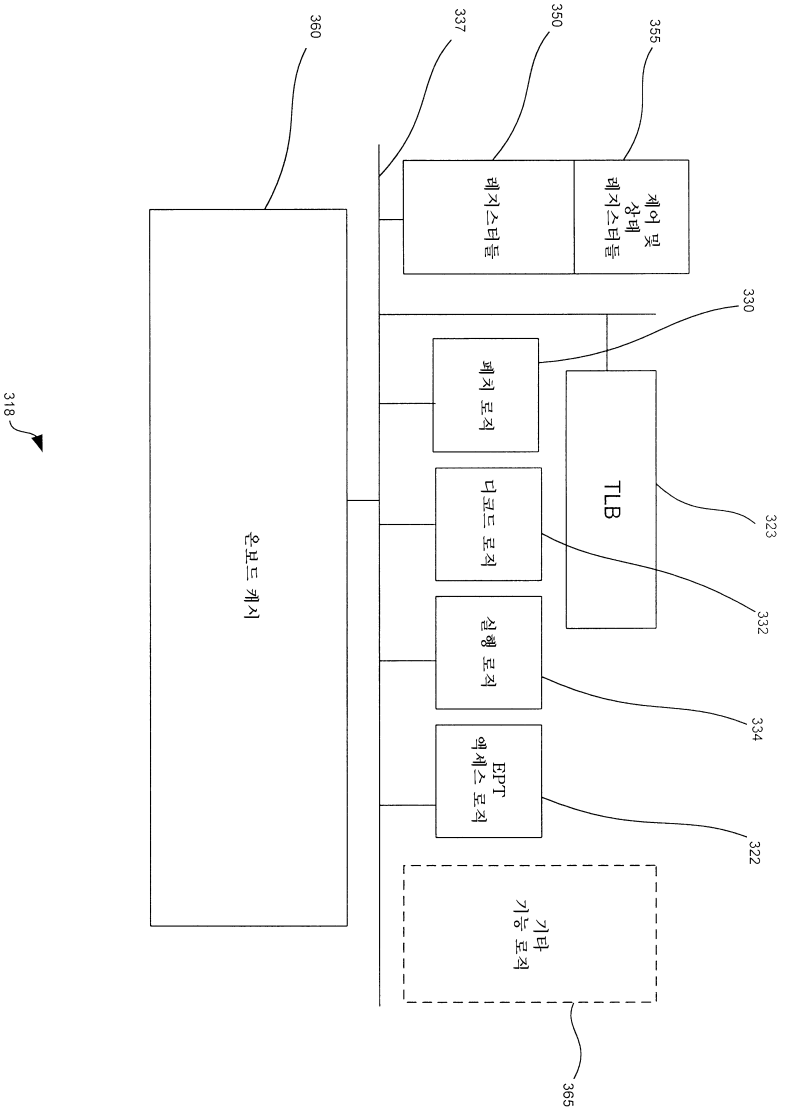
도면2


240
245
242
270
0 MB
128 MB
0 MB
맵핑된
게스트 물리
메모리 A
게스트 물리
메모리 A
가상
머신 A
384 MB
256 MB
275
호스트
물리 메모리
가상 프로세서 A
512 MB
0 MB
맵핑된
게스트 물리
메모리 B
250
게스트 물리
메모리 B
가상
머신 B
768 MB
1024
MB
260
256 MB
가상 프로세서 B
호스트 프로세서
호스트 머신
257
255
265
267

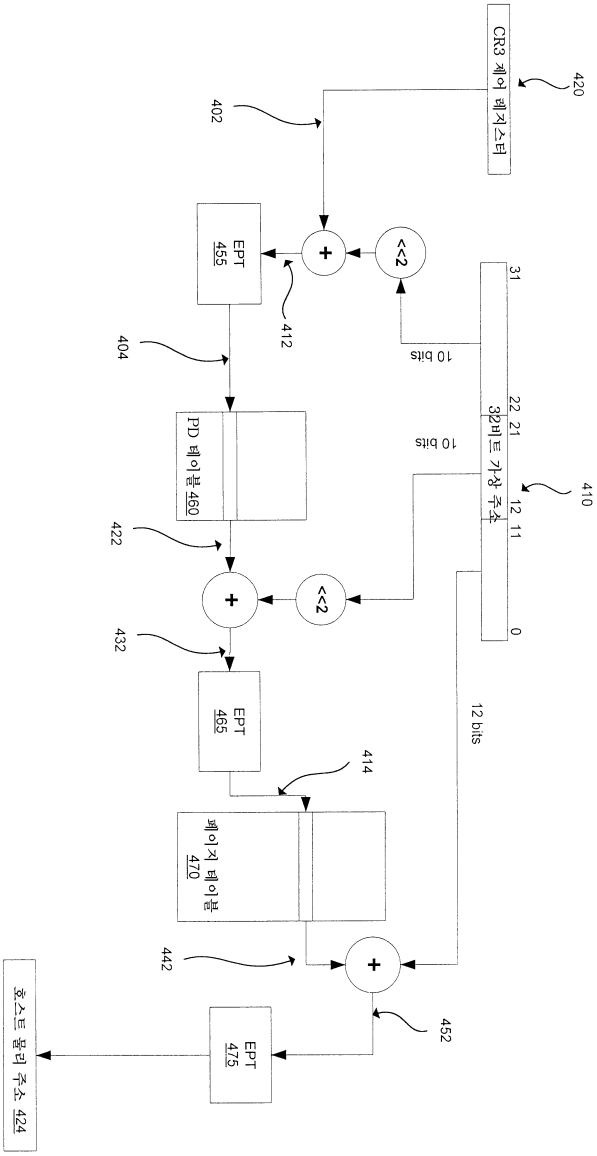
도면3


300
303
게스트 소프트웨어 #1
308
애플리케이션 1
애플리케이션 2
304
OS #1
313
게스트 소프트웨어 #2
310
애플리케이션 1
애플리케이션 2
306
OS #2
302
가상 머신 추상화 1
314
가상 머신 추상화 2
312
가상 머신 모니터 (VMM)
물리 메모리 관리 모듈
326
버스 336
프로세서 318
EPT 액세스 로직 322
TLB 323
VMCS 324
EPT 테이블들 328
메모리 320
베어 플랫폼 하드웨어 316

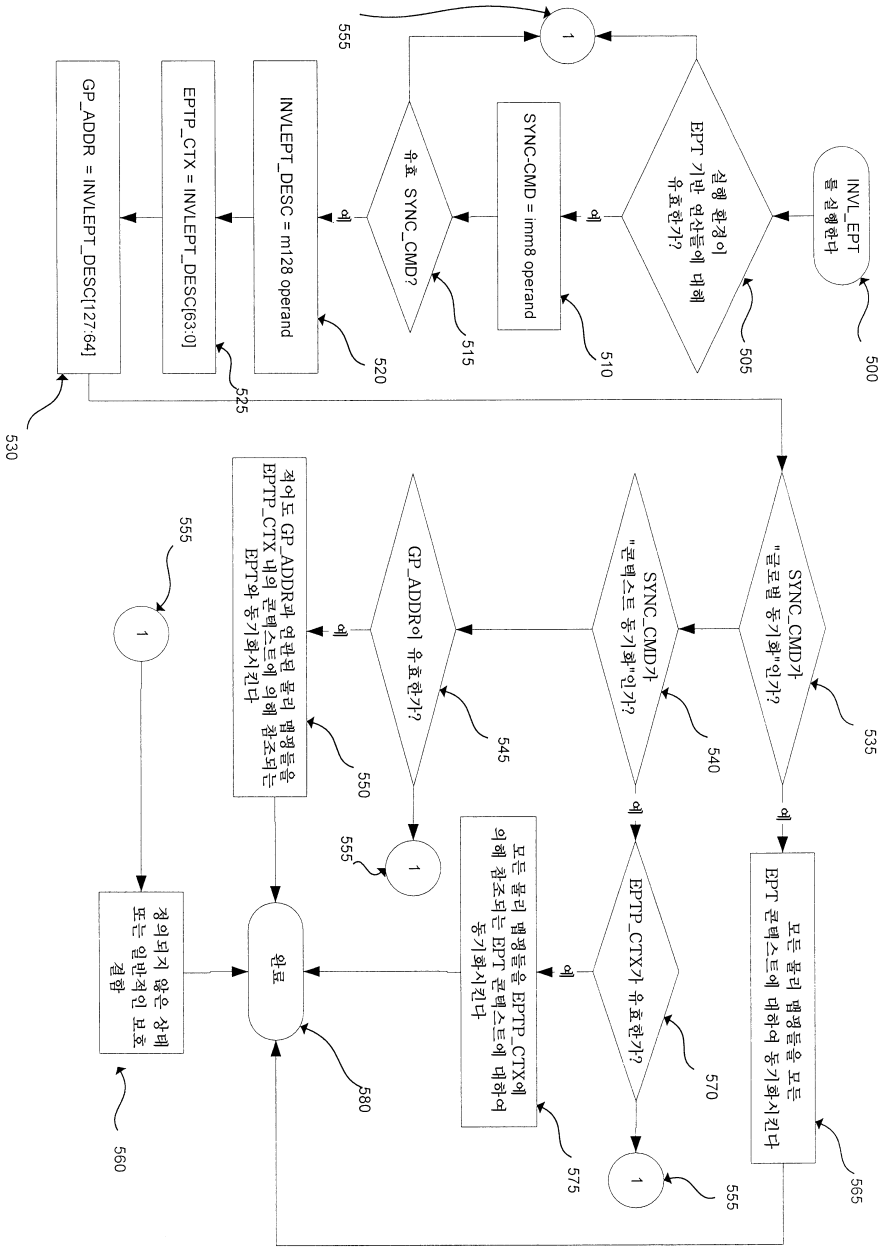
도면3a


318
360
337
350
355
레지스터들
제어 및 상태 레지스터들
330
페치 로직
TLB
323
디코드 로직
332
실행 로직
334
EPT 액세스 로직
322
기타 기능 로직
365
온보드 캐시

도면4


CR3 제어 레지스터
420
402
EPT 455
412
404
PD 테이블 460
422
432
EPT 465
414
페이지 테이블 470
442
EPT 475
452
호스트 물리 주소 424
<<2
<<2
10 bits
10 bits
12 bits
32비트 가상 주소
410
31
22
21
12
11
0

도면5


INVL_EPT 를 실행한다
500
실행 환경이 EPT 기반 연산들에 대해 유효한가?
505
예
SYNC-CMD = imm8 operand
510
유효 SYNC_CMD?
515
예
INVLEPT_DESC = m128 operand
520
EPTP_CTX = INVLEPT_DESC[63:0]
525
GP_ADDR = INVLEPT_DESC[127:64]
530
1
555
SYNC_CMD가 "글로벌 동기화"인가?
535
예
모든 물리 맵핑들을 모든 EPT 콘텍스트에 대하여 동기화시킨다
565
SYNC_CMD가 "콘텍스트 동기화"인가?
540
예
EPTP_CTX가 유효한가?
570
1
555
모든 물리 맵핑들을 EPTP_CTX에 의해 참조되는 EPT 콘텍스트에 대하여 동기화시킨다
575
GP_ADDR이 유효한가?
545
예
적어도 GP_ADDR과 연관된 물리 맵핑들을 EPTP_CTX 내의 콘텍스트에 의해 참조되는 EPT와 동기화시킨다
550
1
555
1
555
정의되지 않은 상태 또는 일반적인 보호 결함
560
완료
580