(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2010년01월15일
(11) 등록번호 10-0937062
(24) 등록일자 2010년01월07일

(51) Int. Cl.
G06F 9/46 (2006.01) *G06F 9/445* (2006.01)
(21) 출원번호 10-2007-7029864
(22) 출원일자 2006년06월02일
심사청구일자 2007년12월21일
(85) 번역문제출일자 2007년12월21일
(65) 공개번호 10-2008-0010464
(43) 공개일자 2008년01월30일
(86) 국제출원번호 PCT/US2006/021652
(87) 국제공개번호 WO 2006/130876
국제공개일자 2006년12월07일
(30) 우선권주장
11/144,527 2005년06월02일 미국(US)
(56) 선행기술조사문헌
US5546568 B1
KR1020030085010 A
KR1020070001085 A
전체 청구항 수 : 총 19 항

(73) 특허권자
인텔 코오퍼레이션
미합중국 캘리포니아 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자
로스만, 마이클
미국 98374 워싱턴주 퓨알럽 183번 스트리트 이스트 11905
짐머, 빈센트
미국 98003 워싱턴주 페더럴 웨이 사우스 369번 스트리트 1937
(74) 대리인
백만기, 양영준

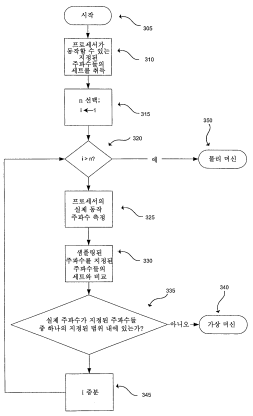
심사관 : 황승희

(54) 가상화 검출

(57) 요 약

프로세서 기반 시스템 상에서 실행되는 프로그램에서, 상기 시스템의 프로세서가 실행되고 있는 주파수의 하나 이상의 샘플을 취득하는 단계; 각각의 샘플을 소정 세트의 주파수들 중 적어도 하나와 비교하는 단계; 및 상기 비교의 결과에 적어도 부분적으로 기초하여 상기 프로그램이 가상 머신 상에서 실행되고 있는지를 결정하는 단계를 포함하는 방법이 개시된다.

대 표 도 - 도3



특허청구의 범위

청구항 1

프로세서 기반 시스템 상에서 실행되는 프로그램에서,

상기 시스템의 프로세서가 실행되고 있는 주파수의 하나 이상의 샘플을 취득하는 단계;

상기 프로세서의 모델 고유 레지스터(model-specific register, MSR)에 저장된 데이터에 적어도 부분적으로 기초하여 상기 프로세서의 예상 실행 주파수들의 세트를 결정하는 단계;

각각의 샘플을 상기 프로세서의 상기 예상 실행 주파수들의 세트 중 적어도 하나와 비교하는 단계; 및

상기 비교의 결과에 적어도 부분적으로 기초하여 상기 프로그램이 가상 머신 상에서 실행되고 있는지를 결정하는 단계

를 포함하는 방법.

청구항 2

제1항에 있어서, 상기 각각의 샘플을 상기 프로세서의 상기 예상 실행 주파수들의 세트 중 적어도 하나와 비교하는 단계는, 상기 샘플이 상기 프로세서의 상기 예상 실행 주파수들의 세트 중 적어도 하나의 지정된 범위 내에 있는지를 결정하는 단계를 더 포함하는 방법.

청구항 3

제2항에 있어서,

상기 프로그램이 가상 머신 상에서 실행되고 있는지를 결정하는 단계는,

각각의 샘플에 대해, 상기 프로세서의 상기 예상 실행 주파수들의 세트 중 적어도 하나가 상기 샘플의 지정된 범위 내에 있는 경우에, 상기 프로그램이 가상 머신에서 실행되고 있지 않은 것으로 결정하는 단계; 및

그렇지 않은 경우에, 상기 프로그램이 가상 머신에서 실행되고 있는 것으로 결정하는 단계

를 더 포함하는 방법.

청구항 4

제3항에 있어서, 상기 프로세서의 상기 예상 실행 주파수들의 세트는 식별 명령의 실행에 응답하여 상기 프로세서에 의해 보고되는 식별자에 적어도 부분적으로 기초하여 선택되는 방법.

청구항 5

제3항에 있어서,

틱 수(tick count)를 취득하기 위해 소정 간격(an interval) 동안 상기 프로세서의 모든 클록 사이클을 계수함으로써 상기 프로세서가 실행되고 있는 주파수의 샘플을 취득하는 단계;

상기 프로세서의 실시간 클록 상에서 상기 간격의 지속 기간을 측정하는 단계; 및

상기 지속 기간으로 상기 틱 수를 나눔으로써 상기 주파수를 계산하는 단계

를 더 포함하는 방법.

청구항 6

제5항에 있어서,

상기 소정 간격 동안 모든 클록 사이클을 계수하는 것은,

상기 간격의 시작에서 상기 프로세서의 제1 사이클 수를 취득하기 위한 명령을 실행하는 것과, 상기 간격의 끝에서 상기 프로세서의 제2 사이클 수를 취득하기 위한 명령을 실행하는 것과, 상기 제1 사이클 수와 상기 제2 사이클 수 간의 차를 취득하는 것을 더 포함하고,

상기 실시간 클록 상에서 상기 간격의 지속 기간을 측정하는 단계는,

상기 프로그램이 상기 프로세서의 상기 실시간 클록에 의해 결정되는 소정의 시간 동안 대기하게 하는 명령을 실행하는 단계를 더 포함하는 방법.

청구항 7

머신에 의해 액세스될 때, 상기 머신이,

시스템의 프로세서가 실행되고 있는 주파수의 하나 이상의 샘플을 취득하는 단계;

상기 프로세서의 모델 고유 레지스터(MSR)에 저장된 데이터에 적어도 부분적으로 기초하여 상기 프로세서의 예상 실행 주파수들의 세트를 결정하는 단계;

각각의 샘플을 상기 프로세서의 상기 예상 실행 주파수들의 세트 중 적어도 하나와 비교하는 단계; 및

상기 비교의 결과에 적어도 부분적으로 기초하여 프로그램이 가상 머신 상에서 실행되고 있는지를 결정하는 단계

를 포함하는 방법을 수행하게 하는 데이터를 저장한 머신 판독 가능 매체.

청구항 8

제7항에 있어서, 상기 각각의 샘플을 상기 프로세서의 상기 예상 실행 주파수들의 세트 중 적어도 하나와 비교하는 단계는, 상기 샘플이 상기 프로세서의 상기 예상 실행 주파수들의 세트 중 적어도 하나의 지정된 범위 내에 있는지를 결정하는 단계를 더 포함하는 머신 판독 가능 매체.

청구항 9

제8항에 있어서,

상기 프로그램이 가상 머신 상에서 실행되고 있는지를 결정하는 단계는,

각각의 샘플에 대해, 상기 프로세서의 상기 예상 실행 주파수들의 세트 중 적어도 하나가 상기 샘플의 지정된 범위 내에 있는 경우에, 상기 프로그램이 가상 머신 상에서 실행되고 있지 않은 것으로 결정하는 단계; 및

그렇지 않은 경우에, 상기 프로그램이 가상 머신에서 실행되고 있는 것으로 결정하는 단계

를 더 포함하는 머신 판독 가능 매체.

청구항 10

제9항에 있어서, 상기 프로세서의 상기 예상 실행 주파수들의 세트는 식별 명령의 실행에 응답하여 상기 프로세서에 의해 보고되는 식별자에 적어도 부분적으로 기초하여 선택되는 머신 판독 가능 매체.

청구항 11

제9항에 있어서,

상기 방법은,

틱 수를 취득하기 위해 소정 간격 동안 상기 프로세서의 모든 클록 사이클을 계수함으로써 상기 프로세서가 실행되고 있는 주파수의 샘플을 취득하는 단계;

상기 프로세서의 실시간 클록 상에서 상기 간격의 지속 기간을 측정하는 단계; 및

상기 지속 기간으로 상기 틱 수를 나눔으로써 상기 주파수를 계산하는 단계

를 더 포함하는 머신 판독 가능 매체.

청구항 12

제11항에 있어서,

상기 소정 간격 동안 모든 클록 사이클을 계수하는 것은,

상기 간격의 시작에서 상기 프로세서의 제1 사이클 수를 취득하기 위한 명령을 실행하는 것과, 상기 간격의 끝에서 상기 프로세서의 제2 사이클 수를 취득하기 위한 명령을 실행하는 것과, 상기 제1 사이클 수와 상기 제2 사이클 수 간의 차를 취득하는 것을 더 포함하고,

상기 실시간 클록 상에서 상기 간격의 지속 기간을 측정하는 단계는,

상기 프로그램이 상기 프로세서의 상기 실시간 클록에 의해 결정되는 소정의 시간 동안 대기하게 하는 명령을 실행하는 단계를 더 포함하는 머신 판독 가능 매체.

청구항 13

시스템으로서,

프로세서;

상기 시스템 상에서 실행 가능한 프로그램을 저장한 저장 장치

를 포함하고,

상기 프로그램은,

상기 시스템의 프로세서가 실행되고 있는 주파수의 하나 이상의 샘플을 취득하고,

상기 프로세서의 모델 고유 레지스터(MSR)에 저장된 데이터에 적어도 부분적으로 기초하여 상기 프로세서의 예상 실행 주파수들의 세트를 결정하고,

각각의 샘플을 상기 프로세서의 상기 예상 실행 주파수들의 세트 중 적어도 하나와 비교하고,

상기 비교의 결과에 적어도 부분적으로 기초하여 상기 프로그램이 가상 머신 상에서 실행되고 있는지를 결정하는 시스템.

청구항 14

제13항에 있어서, 각각의 샘플을 상기 프로세서의 상기 예상 실행 주파수들의 세트 중 적어도 하나와 비교하는 상기 프로그램은, 상기 샘플이 상기 프로세서의 상기 예상 실행 주파수들의 세트 중 상기 적어도 하나의 지정된 범위 내에 있는지를 결정하기 위한 명령들을 더 포함하는 시스템.

청구항 15

제14항에 있어서, 상기 프로그램이 가상 머신 상에서 실행되고 있는지를 결정하는 상기 프로그램은, 각각의 샘플에 대해, 상기 프로세서의 상기 예상 실행 주파수들의 세트 중 적어도 하나가 상기 샘플의 지정된 범위 내에 있는 경우에 상기 프로그램이 가상 머신 상에서 실행되고 있지 않은 것으로 결정하고, 그렇지 않은 경우에는 상기 프로그램이 가상 머신에서 실행되고 있는 것으로 결정하기 위한 명령들을 더 포함하는 시스템.

청구항 16

제15항에 있어서, 상기 프로세서의 상기 예상 실행 주파수들의 세트는 식별 명령의 실행에 응답하여 상기 프로세서에 의해 보고되는 식별자에 적어도 부분적으로 기초하여 선택되는 시스템.

청구항 17

제15항에 있어서,

상기 프로그램은,

틱 수를 취득하기 위해 소정 간격 동안 상기 프로세서의 모든 클록 사이클을 계수함으로써 상기 프로세서가 실행되고 있는 주파수의 샘플을 취득하고,

상기 프로세서의 실시간 클록 상에서 상기 간격의 지속 기간을 측정하고,

상기 지속 기간으로 상기 틱 수를 나눔으로써 상기 주파수를 계산하기 위한 명령들을 더 포함하는 시스템.

청구항 18

제17항에 있어서,

상기 소정 간격 동안 모든 클록 사이클을 계수하는 것은,

상기 간격의 시작에서 상기 프로세서의 제1 사이클 수를 취득하기 위한 명령을 실행하는 것과, 상기 간격의 끝에서 상기 프로세서의 제2 사이클 수를 취득하기 위한 명령을 실행하는 것과, 상기 제1 사이클 수와 상기 제2 사이클 수 간의 차를 취득하는 것을 더 포함하고,

상기 실시간 클록 상에서 상기 간격의 지속 기간을 측정하는 명령들은,

상기 프로그램이 상기 프로세서의 상기 실시간 클록에 의해 결정되는 소정의 시간 동안 대기하게 하는 명령들을 더 포함하는 시스템.

청구항 19

머신 상에서 실행되는 프로그램에서,

상기 머신의 프로세서의 제1 사이클 수를 기록하는 단계;

상기 제1 사이클 수의 기록에 바로 이어서 상기 프로세서의 실시간 클록의 소정의 틱 수 동안 대기하는 단계;

상기 대기에 바로 이어서 제2 사이클 수를 기록하는 단계;

상기 소정의 틱 수와 등가인 시간으로 상기 제2 사이클 수와 상기 제1 사이클 수 간의 차를 나눔으로써 상기 프로세서의 측정된 주파수를 취득하는 단계;

상기 프로세서의 모델 고유 레지스터(MSR)에 저장된 데이터에 적어도 부분적으로 기초하여 상기 프로세서의 예상 실행 주파수들의 세트를 취득하는 단계;

상기 프로세서의 상기 예상 실행 주파수들 중 적어도 하나와 상기 프로세서의 상기 측정된 주파수를 비교하는 단계; 및

상기 예상 주파수와 상기 측정된 주파수 간의 차가 지정된 임계값을 초과하는 경우에 상기 머신은 가상 머신인 것으로 결정하는 단계

를 포함하는 방법.

명 세 서

배 경 기 술

<1> 특히, 데스크톱, 서버, 워크스테이션 또는 노트북 컴퓨터를 포함하는 컴퓨터와 같은 프로세서 기반 시스템에서, 개인 휴대 단말기, 즉 PDA, "스마트" 이동 전화 또는 휴대형 게임 시스템과 같은 핸드헬드 장치에서, 또는 게임 콘솔 또는 스테이션, 셋톱 박스 또는 다른 가정용 오락 장치에서 프로세서가 사용될 수 있다. 각각의 경우에, 프로세서는 기본 실행 사이클에 기초하여 동작하며, 하나의 프로세서 파라미터는 프로세서 사이클이 발생하는 주파수이다. 이 주파수는 초당 사이클, 즉 헤르츠(Hz), 또는 메가 헤르츠(MHz) 및 기가 헤르츠(GHz)와 같은 그의 배수들로 측정된다.

<2> 몇몇 경우에, 프로세서는 다수의 주파수로 동작할 수 있으며, 예를 들어 절전 모드에 있을 때, 프로세서는 고성능 모드에 있을 때 사용한 것보다 낮은 주파수의 동작 모드로 전환할 수 있다. 프로세서가 상이한 모드로 동작할 수 있는 상이한 주파수들의 세트는 통상적으로 프로세서의 제조자에 의해 지정되는 개별 값들의 세트이다.

<3> 몇몇 경우에, 프로세서 제조자는, 프로세서 상에서 실행되는 프로그램이 프로세서가 동작하도록 의도되어 있는 지정된 주파수 또는 주파수들을 결정하는 방법을 제공할 수 있다. 예를 들어, 프로그램은, 프로세서가, 프로그램이 저장된 테이블에 액세스함으로써 프로세서가 동작할 수 있는 대응하는 주파수 또는 주파수들을 결정할 수 있는 근거가 되는 모델 번호를 반환하게 하는 명령을 실행할 수 있다.

<4> 프로세서 기반 시스템들은 또한 실시간 클록을 제공할 수 있다. 이러한 시스템 상에서 실행되는 프로그램들은 실시간 클록에 액세스할 수 있으며, 이를 이용하여, 예를 들어 프로그램이 실시간 클록에 의해 측정되는 특정 시간 동안 중지하게 함으로써 프로그램 실행 동안 실제 기간을 프로그램 방식으로 결정할 수 있다.

<5> 가상화는, 하드웨어 및 소프트웨어로, 또는 몇몇 경우에는 소프트웨어만으로 가상화를 지원하는 프로세서 기반

호스트 머신이 호스트의 추상화를 제공할 수 있게 하여, 호스트 머신의 기반 하드웨어가 독립적으로 동작하는 하나 이상의 가상 머신으로 보이게 하는 기술이다. 따라서, 각각의 가상 머신은 독립식 플랫폼(self-contained platform)으로서 기능할 수 있다. 종종, 가상화 기술은, 다수의 게스트 운영 체제 및/또는 다른 게스트 소프트웨어가 동일 하드웨어 플랫폼 상에서 실제로 물리적으로 실행되는 동안 외견상 동시에 그리고 외견상 독립적으로 다수의 가상 머신 상에 공존하고 실행될 수 있게 하는 데 이용된다. 가상 머신은 호스트 머신의 하드웨어를 모방하거나, 대안으로 대체로 상이한 하드웨어 추상화를 제공할 수 있다.

<6> 가상화 시스템들은 가상 머신에서 동작하는 게스트 소프트웨어에 한 세트의 자원들(예를 들어, 프로세서, 메모리, IO 장치)을 제공하며, 물리 호스트 머신의 컴포넌트들의 일부 또는 전부를 가상 머신으로 맵핑하거나, 완전 가상 컴포넌트들(fully virtual components)을 생성할 수 있다. 따라서, 가상화 시스템은 게스트 소프트웨어에 "가상 베어 머신(virtual bare machine)" 인터페이스를 제공한다고 할 수 있다.

발명의 상세한 설명

<11> 몇몇 실시예에서, 가상화 시스템들은 호스트 머신을 제어하는 가상 머신 모니터(VMM)를 포함할 수 있다. VMM은 가상 머신(VM)에서 동작하는 게스트 소프트웨어에 프로세서, 메모리 및 IO 장치와 같은 한 세트의 자원을 제공한다. VMM은 물리 호스트 머신의 컴포넌트들의 일부 또는 전부를 가상 머신으로 맵핑할 수 있으며, 가상 머신에 포함되는, VMM에서 소프트웨어로 에뮬레이트되는 완전 가상 컴포넌트들(예를 들어, 가상 IO 장치)을 생성할 수 있다. VMM은 하드웨어 가상화 아키텍처의 설비를 이용하여 가상 머신에 서비스를 제공하며, 호스트 머신 상에서 실행되는 다수의 가상 머신으로부터 그리고 이들 사이의 보호를 제공한다.

<12> 도 1은 가상 머신 환경(100)의 일 실시예를 나타낸다. 이 실시예에서, 플랫폼 하드웨어(116)는 VMM(112)을 실행할 수 있다. VMM은, 통상적으로는 소프트웨어로 구현되지만, 가상 베어 머신 인터페이스를 보다 하이 레벨의 소프트웨어로 에뮬레이트하고 익스포트(export)할 수 있다. 몇몇 실시예에서 이러한 보다 하이 레벨의 소프트웨어는 표준 OS, 실시간 OS를 포함하거나, 제한된 운영 체제 기능을 가진 스트립-다운(stripped-down) 환경일 수 있으며, 표준 OS에서 통상적으로 이용 가능한 OS 설비를 포함하지 않을 수 있다. 대안으로, 예를 들어, VMM(112)은 다른 VMM 내에서 또는 다른 VMM의 서비스를 이용하여 실행될 수 있다. VMM은 몇몇 실시예에서 예를 들어 하드웨어, 소프트웨어, 펌웨어, 또는 다양한 기술의 조합으로 구현될 수 있다.

<13> 플랫폼 하드웨어(116)는 개인용 컴퓨터(PC), 메인프레임, PDA 또는 "스마트" 이동 전화와 같은 핸드헬드 장치, 휴대형 컴퓨터, 셋톱 박스, 또는 다른 프로세서 기반 시스템일 수 있다. 플랫폼 하드웨어(116)는 적어도 프로세서(118) 및 메모리(120)를 포함한다. 프로세서(118)는 마이크로프로세서, 디지털 신호 프로세서, 마이크로컨트롤러 등과 같이 프로그램을 실행할 수 있는 임의 타입의 프로세서일 수 있다. 프로세서는 실시예들에서 실행을 위한 마이크로코드, 프로그램가능 논리 또는 하드 코드 논리(hard coded logic)를 포함할 수 있다. 도 1은 이러한 하나의 프로세서(118)만을 도시하고 있지만, 실시예에 있어서 시스템 내에는 하나 이상의 프로세서가 존재할 수 있다. 또한, 프로세서(118)는 다수의 코어, 다수의 스레드에 대한 지원 등을 포함할 수 있다. 다양한 실시예에서, 메모리(120)는 하드 디스크, 플로피 디스크, 랜덤 액세스 메모리(RAM), 판독 전용 메모리(ROM), 플래시 메모리, 상기 장치들의 임의 조합, 또는 프로세서(118)에 의해 판독 가능한 임의의 다른 타입의 머신 매체를 포함할 수 있다. 메모리(120)는 프로그램 실행 및 다른 방법 실시예들을 수행하기 위한 명령 및/또는 데이터를 저장할 수 있다.

<14> VMM(112)은 게스트 소프트웨어에 하나 이상의 가상 머신의 추상화를 제공하는데, 이는 다양한 게스트에게 동일 또는 상이한 추상화를 제공할 수 있다. 도 1은 2개의 가상 머신(102, 114)을 나타낸다. 각각의 가상 머신 상에서 실행되는 게스트 소프트웨어(103, 113)와 같은 게스트 소프트웨어는 게스트 OS(104 또는 106)와 같은 게스트 OS 및 다양한 게스트 소프트웨어 애플리케이션(108, 110)을 포함할 수 있다. 게스트 소프트웨어(103, 113)는 게스트 소프트웨어(103, 113)가 실행되고 있는 가상 머신들 내의 물리 자원들(예를 들어, 프로세서 레지스터, 메모리 및 I/O 장치)에 액세스하고, 다른 기능들을 수행할 수 있다. 예를 들어, 게스트 소프트웨어(103, 113)는 가상 머신(102, 114)에서 제공되는 프로세서 및 플랫폼의 아키텍처에 따라 모든 레지스터, 캐시, 구조, I/O 장치, 메모리 등에 액세스할 것으로 예상된다.

<15> 일 실시예에서, 프로세서(118)는 가상 머신 제어 구조(VMCS)(124)에 저장된 데이터에 따라 가상 머신들(102, 114)의 동작을 제어한다. VMCS(124)는 게스트 소프트웨어(103, 113)의 상태, VMM(112)의 상태, VMM(112)이 게스트 소프트웨어(103, 113)의 동작을 어떻게 제어하기를 원하는지를 나타내는 실행 제어 정보, VMM(112)과 가상 머신 간의 전이(transitions)를 제어하는 정보 등을 포함할 수 있는 구조이다. 프로세서(118)는 VMCS(124)로부터 정보를 판독하여, 가상 머신의 실행 환경을 결정하고 그의 거동을 강제한다. 일 실시예에서, VMCS(124)는

메모리(120)에 저장된다. 몇몇 실시예에서, 다수의 가상 머신을 지원하기 위해 다수의 VMCS 구조가 사용된다.

<16> 게스트 소프트웨어(예를 들어, 게스트 OS(104) 및 애플리케이션(108)을 포함하는 참조번호 103)에 의해 액세스될 수 있는 자원들은 "특권(privileged)" 또는 "비특권(non-privileged)"으로 분류될 수 있다. 특권 자원들에 대해, VMM(112)은 이들 특권 자원에 대한 근본적인 제어를 유지하면서 게스트 소프트웨어가 원하는 기능을 돕는다. 또한, 각각의 게스트 소프트웨어(103, 113)는 예외(예를 들어, 페이지 결함, 일반적인 보호 결함 등), 인터럽트(예를 들어, 하드웨어 인터럽트, 소프트웨어 인터럽트), 및 플랫폼 이벤트(예를 들어, 초기화(INIT) 및 시스템 관리 인터럽트(SMI))와 같은 다양한 플랫폼 이벤트를 처리할 것으로 예상된다. 이들 플랫폼 이벤트의 일부는 "특권"인데, 그 이유는 이들이 가상 머신들(102, 114)의 적절한 동작을 보증하기 위해, 그리고 게스트 소프트웨어로부터 그리고 게스트 소프트웨어 간의 보호를 위해 VMM(112)에 의해 처리되어야하기 때문이다. 게스트 운영 체제 및 게스트 애플리케이션 양자는 특권 자원에 액세스하려고 시도할 수 있으며, 이들 양자는 특권 이벤트를 유발하거나 경험할 수 있다. 본 명세서에서, 특권 플랫폼 이벤트 및 특권 자원들에 대한 액세스 시도는 총괄하여 "특권 이벤트" 또는 "가상화 이벤트"로 지칭된다.

<17> 도 1에서 전술하고 도시된 바와 같은 실시예에서의 가상 머신 환경의 동작은 도 2에 도시된 처리에 의해 설명된다. 도 2는 게스트 소프트웨어에서 발생하는 특권 이벤트를 처리하기 위한 실시예에서의 VM 환경의 동작, 및 게스트 소프트웨어에 의해 비특권 이벤트를 처리하기 위한 실시예의 동작을 나타낸다. 도 2는 도 1에 도시된 것과 같은 환경에서 발생할 수 있는 모든 동작 또는 모든 컴포넌트를 도시하지는 않는다. 이것은 단지 설명의 명료화를 위한 것이다. 도 2에는 소규모 세트의 컴포넌트들 및 소수의 특정 동작들이 나타나지만, 실시예에서의 VM 환경은 많은 다른 컴포넌트를 포함할 수 있으며, 이러한 실시예에서 많은 다른 동작이 발생할 수 있다.

<18> 도 2는 도 1에서 전술한 가상 머신 추상화(102) 및 플랫폼 하드웨어(116) 상에서 실행되는 게스트 소프트웨어(103)의 하나의 예시적인 동작 세트를 나타낸다. 동작들은 이들이 시스템 내의 어디에서(예를 들어, VMM(112)에서, 게스트 소프트웨어(103)에서 등) 발생하는지를 나타내는 블록들 내에 도시되어 있다. 전술한 VM 환경의 다른 컴포넌트들 외에, VM 추상화(102)는 참조번호 212에서 가상 머신 상태 및 게스트 소프트웨어(103)에 대한 다른 상태 정보를 저장할 수 있으며, 또한 많은 예 중 두 가지 예로서 가상 네트워크 접속 또는 범용 레지스터들의 세트와 같은 다른 자원들을 게스트에 제공할 수 있다. 물론, VM 상태, 게스트 상태 및 다른 VM 자원들을 구현하는 물리 자원들은 VM이 실행되는 플랫폼 하드웨어(116)에 의해 실제로 제공된다. 플랫폼 하드웨어는 메모리(120), VMCS(124) 및 프로세서(118)를 포함한다.

<19> 참조번호 240에서, 게스트 소프트웨어(103)는 비특권 자원(242)에 액세스한다. 비특권 자원들은 VMM(112)에 의해 제어될 필요가 없고, VMM(112)의 호출 없이 계속되는 게스트 소프트웨어에 의해 직접 액세스될 수 있어서, 게스트가 비특권 자원(242)에 액세스한 후에 참조번호 245에서 동작을 계속하는 것이 가능하게 된다. 비특권 플랫폼 이벤트 또한 VMM(112)의 개입 없이 처리될 것이다(이것은 도 2에는 도시되지 않는다).

<20> 참조번호 205에서, 게스트 소프트웨어(103)는 특권 자원에 액세스하려고 시도하고, 그리고/또는 특권 플랫폼 이벤트를 경험한다. 참조번호 205에서와 같이 그러한 특권 이벤트가 발생할 때, 제어는 VMM(112)으로 전달될 수 있다(207). 게스트 소프트웨어에서 VMM(112)으로의 제어의 전달(207)은 본 명세서에서 가상 머신 퇴장(virtual machine exit)으로서 지칭된다. 자원 액세스를 돕거나, 아니면 특권 이벤트를 적절히 처리한 후, VMM(112)은 참조번호 232에서와 같이 제어를 게스트 소프트웨어에 반환할 수 있으며, 이어서 게스트 소프트웨어는 동작을 재개한다(235). VMM(112)에서 게스트 소프트웨어로의 제어의 전달(232)은 가상 머신 입장(virtual machine entry)으로서 지칭된다. 일 실시예에서, VMM(112)은 본 명세서에서 가상 머신 입장 명령으로 지칭되는 전이를 트리거하도록 특별히 설계된 명령을 실행함으로써(230) 가상 머신 입장을 개시한다.

<21> 일 실시예에서, 가상 머신 퇴장이 발생할 때, 게스트 소프트웨어에 의해 사용되는 프로세서 상태의 컴포넌트들이 저장되고(210), VMM(112)에 의해 요구되는 프로세서 상태의 컴포넌트들이 로딩되며, 참조번호 220에서 VMM(112)에서 실행이 재개된다. 일 실시예에서, 게스트 소프트웨어에 의해 사용되는 프로세서 상태의 컴포넌트들은 VMCS(124)의 게스트 상태 영역에 저장되며, VMM(112)에 의해 요구되는 프로세서 상태의 컴포넌트들은 VMCS(124)의 모니터 상태 영역에 저장된다. 일 실시예에서, VMM(112)에서 게스트 소프트웨어로의 전이가 발생할 때, 가상 머신 퇴장시에 저장된(그리고 가상 머신 퇴장을 처리하는 동안 VMM(112)에 의해 수정되었을 수 있는) 프로세서 상태의 컴포넌트들이 복원되며(225), 참조번호 230에서 제어가 가상 머신(102, 114)으로 반환된다.

<22> 다른 실시예들에서, VM의 구조 및 게스트 소프트웨어에 대한 지원의 구성은 상이할 수 있다. 가상화를 지원하기 위해 호스트 머신 상에서 실행되는 소프트웨어는 VMM으로 지칭되거나 지칭되지 않을 수 있으며, 몇몇 예에서

가상 머신 지원 시스템은 하드웨어 컴포넌트 또는 지원을 갖지 않을 수 있다. 또 다른 실시예들에서, 전체 VMM 및 게스트는 도 1에 도시된 구조와 달리 실행 운영 체제 내에서 실행될 수 있다. 이 분야에 공지된 바와 같이, 가상 머신들의 많은 다른 구현도 가능하다.

<23> 도 1 및 도 2를 참조하여 설명된 실시예에서 전술한 바와 같이, 하드웨어에 의해 지원되는 VMM이 VM을 구현할 때, VM 내에서 실행되는 프로그램은 많은 점에서 물리 머신 환경과 구별할 수 없는 가상화된 게스트 머신 환경을 제공받을 수 있다. 하드웨어 지원으로, VMM은 가상 프로세서의 모델 고유 레지스터들과 같은 특권 자원들에 대한 액세스와 같은 특수 명령들을 트랩하고 정확히 처리하여, 물리 프로세서에 의해 반환되는 바와 같은 값들을 반환할 수 있으며, 또한 하드웨어에 대한 특권 액세스, 예를 들어 I/O 장치에 대한 부작용을 갖는 메모리 액세스가 실시예의 하드웨어에서의 가상화 지원과 함께, 전술한 VMM 및 VMCS의 동작에 의해 실시예에서 적절히 시뮬레이션될 수 있다. 구체적으로, 일례에서, 특정 플랫폼은, 예를 들어 기반 물리 프로세서와 동일한 프로세서 타입 및 모델인 가상 프로세서, 물리 머신의 버스들에 접속된 것들과 동일한 I/O 장치들 등을 제공함으로써 많은 점에서 기반 물리 하드웨어와 유사 또는 동일한 가상 하드웨어를 제공하는 VM을 제공할 수 있다. 이어서, 게스트에 의해 이루어지는 프로세서 또는 다른 하드웨어에 대한 플랫폼 또는 프로세서 고유 참조들은 적절한 응답을 위해 중간 VMM 및 VMCS를 통해 물리 플랫폼 및 VM에 전달될 수 있으며, 따라서 기반 물리 하드웨어의 유사한 복제인 환경을 게스트에게 제공할 수 있다. 이것은 게스트가 중재 가상화의 존재를 검출하는 것을 매우 어렵게 한다.

<24> 대안으로, VMM 및 가상화 지원 시스템은 기반 물리 시스템과 다른 프로세서 또는 플랫폼에 기초하는 환경을 제공할 수 있다. 이 경우에도, 가상화 서브시스템 및 VMM의 주의 깊은 구현은 가상 머신 상에서 실행되는 프로그램이 임의의 간단한 방법으로 환경의 가상화된 특성을 검출하는 것을 방지할 수 있다.

<25> 몇몇 상황에서는, 프로그램이 그가 실행되고 있는 환경의 가상화된 특성을 인식하는 것이 중요할 수 있다. 예를 들어, 성능 임계 프로그램의 적절한 동작을 갖기를 원하는 프로그램 벤더에게는 프로그램이 최소 메모리 크기 및 프로세서 주파수와 같은 소정의 최소 능력을 가진 물리 하드웨어 상에만 설치되는 것을 보장하는 것이 필요할 수 있다. 가상화된 환경에서, VM은 메모리 크기 또는 프로세서 주파수, 또는 가상화된 하드웨어의 다른 파라미터들을 보고할 수 있는데, 이는 기반 하드웨어의 실제 능력을 정확히 반영하지 못할 수도 있다. 더욱이, VM 내에서의 프로그램 실행은 일반적으로 단지 VM 자체의 동작으로 인하여 오버헤드를 유발하며, 이러한 오버헤드는 프로그램에서의 소정의 성능 임계 처리에 대해 바람직하지 않을 수 있다. 보안 데이터를 처리하거나 표시하는 것들과 같은 다른 프로그램들은 하드웨어 장치를 인증하거나 승인된 하드웨어 플랫폼 상에서만 실행되기를 원할 수 있다. 이러한 프로그램이 실행된 플랫폼이 실제로, 승인되지 않은 플랫폼 상에서 실행되고, 승인된 물리 하드웨어 플랫폼을 시뮬레이션하도록 악의적으로 설계된 VM이었다면, 이러한 프로그램의 보안은, 프로그램이 프로그램이 실행되고 있었던 플랫폼이 가상화된 것을 검출하는 것이 불가능한 경우에는 손상될 수 있다.

<26> 프로그램이 프로그램이 가상화된 구현 상에서 실행되고 있음을 검출하는 프로세스가 도 3에 도시되어 있다. 도 3에서, 프로세서의 측정된 동작 주파수와 그의 지정된 유효 동작 주파수 세트 간의 비교 프로세스의 1회 이상의 반복이 일 실시예에서 발생한다. 프로세스의 시작(305)에서, 프로세서가 동작할 수 있는 한 세트의 유효 지정 주파수들이 얻어진다(310). i가 흐름도의 루프 변수를 나타내고, 단계 315에서 1부터 시작하며, 단계 320에서 퇴장에 의해 소정의 수 n으로 한정되는 반복 방식으로, 측정되고 지정된 주파수들의 n번의 비교가 이루어진다. 공지된 바와 같이, 도시된 기본 루프와 동등한 임의의 반복 방식이 이용될 수 있다. 몇몇 예에서, 루프는, 즉 n이 1일 때 생략될 수 있다.

<27> 프로세서 주파수의 결정은 프로세서가 동작할 수 있는 지정된 주파수들의 결정과 같이 이 분야에 공지되어 있다. 각각의 반복에서, 단계 325에서 프로세서의 실제 주파수가 측정되고, 이어서 단계 330에서 측정 값이 지정된 유효 주파수들의 세트와 비교된다. 단계 335에서 비교 결과가 평가된다. 임의의 비교가 지정된 주파수들에 대한 정상 허용 범위 밖에 있는 경우, 머신은 가상 머신이며, 단계 340에서 프로세스가 완료된다. 그렇지 않은 경우, 단계 345에서 루프가 반복되어 루프 카운터의 값이 증분된다. 범위 밖의 측정(out-of-range measurement) 없이 모든 n번의 테스트가 완료되는 경우, 즉 단계 320에서 루프를 나가는 경우, 이것은 단계 350에서 프로세스가 가상 머신이 아니라 물리 머신 상에서 실행되고 있음을 지시하는 결과를 의미한다.

<28> 일 실시예에서, 가상화는, IA-32 인텔 아키텍처 소프트웨어 개발자 매뉴얼(IA-32 문서)에 설명된 IA-32 인텔 아키텍처 플랫폼(IA-32)과 같은 인텔 아키텍처 프로세서 상의 내부 아키텍처 지원 및 VMM을 이용하여 구현된다. 일 실시예에서, 가상화된 프로세서 및 기반 물리 프로세서는 둘다 IA-32 프로세서들이며, 프로세서에 의해 실행되는 사이클들의 수, 실시간 클록의 값 및 프로세서의 식별번호를 결정하는 방법의 결정과 같은 특정 명령들을

지원한다. 예를 들어, IA-32 아키텍처에서, 판독 시간 스탬프 카운터(RDTSC) 명령이 기본 프로세서 사이클을 계수하는 데 사용될 수 있다. IA-32 RDMSR(모델 고유 레지스터(MSR)로부터의 판독) 및 CPUID 명령을 이용하여 프로세서의 모델, 타입 및 식별번호에 대한 다양한 파라미터를 결정할 수 있다. 이들은 CPU 타입을 포함하며, 이는 또한 IA-32 문서에 지정된 바와 같은 테이블로부터 특정 버스 속도로 지정되는 동작 주파수의 결정을 가능하게 한다. 또한, 이어서, 프로세서 주파수 구성 필드 및 스케일 가능 버스 속도 필드와 같은 IA-32 MSR 내의 다른 필드들을 테이블과 함께 이용하여 예상 프로세서 주파수를 구할 수 있다.

<29> 이들 명령, 또는 다른 아키텍처에서의 등가 세트를 이용하여 가상화된 환경을 검출할 수 있다. 그러한 하나의 IA-32 실시예에서, 도 3의 하이 레벨 프로그램은 도 4에 도시된 바와 같은 IA-32 아키텍처의 특정 명령들을 이용하는 프로그램으로서 구현될 수 있다. 가상화를 검출하기 위한 프로그램 프로세스(480)는, 프로그램이 먼저 그가 실행되고 있는 프로세서로부터 단계 410에서 RDTSC와 같은 명령을 이용하여 실행되는 전체 기본 클록 사이클들에 대한 값(Tc1)을 요청함으로써 시작된다. 이어서, 시스템의 실시간 클록(RTC)이 액세스되고(420), 단계 425에서 프로세스는 실시간 클록의 공지된 기간, 여기서는 n 틱(tick) 동안 대기하거나 루프한다(425). 이어서, 단계 430에서, 프로그램은 실행되는 프로세서 클록 사이클들에 대한 새로운 현재 값(Tc2)을 판독한다. 단계 460에서, 시간으로 나눈 이 두 값의 차이, 즉 (Tc2-Tc1)/n은 측정된 주파수(Fm)를 산출한다. 이어서, 단계 495에서, CPUID 또는 유사한 명령을 이용하여 프로세서의 식별 정보가 액세스된다. 이 정보는 프로세스(480)에 의해 판독될 수 있는 프로세서의 모델 고유 레지스터(MSR)들 내의 한 세트의 레지스터 값들로서 제공될 수 있다. 이어서, 단계 450에서, 취득된 값들 중 적어도 하나 이상을 이용하여 프로세서의 사양의 일부로서 공개된 소정의 테이블 내에 색인해서, 한 세트의 가능한 소정의 주파수들 및 이들 주파수에 대한 허용 범위를 산출할 수 있다. 단계 450에서, 프로그램은 측정된 주파수(Fm)에 가장 가까운 지정된 주파수(Fs) 및 프로세서에 대해 허용된 드리프트 또는 변동에 관한 관련 지정 범위 또는 허용 범위를 선택하며, 값 델타가 테이블로부터 판독되거나 프로세서에 대해 지정된 다른 데이터로부터 알려질 수 있다. 이어서, 단계 480에서, Fs와 Fm 사이의 차의 절대값이 계산되어, 델타와 비교된다. 차의 절대값이 델타를 초과하는 경우, 프로그램은 가상 머신 또는 가상화된 플랫폼 상에서 실행되고 있으며(470), 그렇지 않은 경우, 플랫폼은 가상화되지 않은 물리 플랫폼이다(490).

<30> 이러한 처리의 정확성은, 이용되는 실제 가상화 방법에 관계없이 가상화된 실시간 클록이 물리 실시간 클록과 동일해야 하는 가능성에 의존한다. 따라서, 단계 420 및 425에서와 같이 실시간 클록에 액세스하는 프로그램이 가상화된 환경에서 실행되고 있을 수 있더라도, 가상화된 환경이 소정 타입의 시간 임계 기능들을 적절히 수행해야 하는 경우, 가상화된 환경은 기반 시스템의 실시간 클록에 대한 직접적인 액세스를 제공해야 한다. 일반적으로, 생산 품질 가상화 시스템은 게스트가 볼 때 실시간 클록을 가상화하지 않는 대신, 기반 시스템의 실시간 클록에 대한 직접적인 액세스를 제공하는 것으로 가정할 수 있다. 이것은 게스트에 대해 가상화를 검출하기 위해 보여진 바와 같이 사용될 수 있는 실제 물리 머신에 대한 액세스 윈도우를 제공한다. 가상 머신에서 게스트로서 실행되는 프로그램이 물리 실시간 클록에 대한 액세스를 갖는 경우, 프로그램은 가상 머신에 의해 제공되는 가상 프로세서의 외관상의 주파수를, 전술한 바와 같이 가상 프로세서와 동일한 물리 프로세서가 동작하도록 지정되는 주파수들과 비교할 수 있다. 가상화에 수반되는 오버헤드로 인하여, 그리고 가상 환경의 컴포넌트들이 소프트웨어로 에뮬레이트되므로, 가상 프로세서의 측정 주파수는 시간에 따라 변할 가능성이 매우 높고, 일반적으로는 대응하는 물리 프로세서의 동작 주파수의 정상적인 예상 변동 범위 밖에 있을 가능성이 있으며, 따라서 플랫폼의 가상화된 특성은 정상적인 주파수 변동 밖의 편차를 검출함으로써 검출될 수 있다. 일반적으로, 가상화된 프로세서가 모델 및 사양에 있어서 기반 물리 프로세서와 동일한 경우, 가상화된 주파수는 특정 버스 속도에서 물리 프로세서의 실제 주파수보다 낮을 것이다. 가상화된 프로세서가 예를 들어 보다 느린 프로세서의 프로세서 모델 정보를 제공함으로써 기반 물리 프로세서보다 낮은 동작 주파수를 갖는 프로세서로서 게스트에 제공되는 경우에도, 가상화의 기본 특성에 의해 발생하는 가상화된 주파수의 거의 피할 수 없는 변동은 전술한 처리에 의해 여전히 높은 확률로 검출될 것이다. 보다 높은 정확성을 위해, 프로세스는 정상 범위 밖의 측정 주파수를 구하기 위해 여러 번 반복될 수 있다.

<31> 전술한 실시예는 IA-32 프로세서에서 이용 가능한 타입의 하이 레벨 아키텍처 특징들, 즉 클록 사이클 카운터 및 실시간 클록의 가용성에 기초한다. 그러나 도 1에 도시된 처리의 일반적인 흐름은 특정 아키텍처에 의존하지 않는다. 특정 상세들은 도 1에 도시된 것들 및 전술한 IA-32 명령들과 다를 수 있지만, 대부분의 현대적인 프로세서 기반 시스템들은 프로세서의 실제 동작 주파수를 측정하는 방법, 및 프로세서의 지정된 동작 주파수들을 결정하는 방법을 제공한다. 따라서, 다른 실시예들에서, 프로세서의 측정 주파수가 프로세서의 지정 주파수에 가까운지를 결정하는 많은 대안적인 방법이 이용될 수 있음을 이 분야의 전문가는 이해할 것이다.

<32> 몇몇 실시예들은, 머신에 의해 액세스될 때 실시예의 프로세스를 수행하는 명령들을 저장한 머신 또는 머신 판독 가능 매체를 포함할 수 있는 소프트웨어 프로그램 제품 또는 소프트웨어로서 제공될 수 있다. 다른 실시예들에서, 프로세스들은 이들 프로세스를 수행하기 위한 하드와이어드 논리를 포함하는 특정 하드웨어 컴포넌트들에 의해, 또는 프로그래밍된 컴포넌트들 및 커스텀 하드웨어 컴포넌트들의 임의 조합에 의해 수행될 수 있다.

<33> 위의 설명에서, 설명의 목적으로, 전술한 실시예들의 충분한 이해를 제공하기 위해 다수의 특정 상세가 설명되었지만, 이 분야의 전문가는 이러한 특성 상세들 없이도 많은 다른 실시예가 실시될 수 있음을 이해할 것이다.

<34> 위의 상세한 설명의 몇몇 부분은 프로세서 기반 시스템 내에서의 데이터 비트들에 대한 동작들의 알고리즘들 및 심볼 표현들과 관련하여 제공되어 있다. 이러한 알고리즘적 기술 및 표현들은 이 분야의 전문가들이 이 분야의 다른 전문가들에게 그들의 연구의 내용을 가장 효과적으로 전달하기 위해 사용하는 수단이다. 동작들은 물리적인 양의 물리적 조작을 요구하는 동작들이다. 이러한 양은 저장, 전달, 조합, 비교, 아니면 조작될 수 있는 전기, 자기, 광학 또는 다른 물리 신호들의 형태를 취할 수 있다. 이들 신호를 비트, 값, 요소, 심볼, 문자, 용어, 번호 등으로 지칭하는 것은 주로 공동 이용의 이유에서 때때로 편리한 것으로 입증되었다.

<35> 그러나 이들 및 유사한 용어 모두는 적절한 물리적인 양들과 연관되어야 하며, 이들 양에 적용되는 편리한 라벨들일 뿐이라는 것을 명심해야 한다. 설명으로부터 명백하듯이, 구체적으로 달리 언급되지 않는 한, "실행" 또는 "처리" 또는 "컴퓨팅" 또는 "계산" 또는 "결정" 등의 용어들은 프로세서 기반 시스템의 저장 장치 내에서 물리적인 양들로서 표현되는 데이터를 조작하여 다른 그러한 정보 저장, 전송 또는 표시 장치들 상에 유사하게 표현되는 다른 데이터로 변환하는 프로세서 기반 시스템, 또는 유사한 전자 컴퓨팅 장치의 동작 및 프로세스를 지칭할 수 있다.

<36> 실시예들의 설명에 있어서, 첨부된 도면들이 참조될 수 있다. 도면들에서, 여러 도면 전반에서 동일한 번호는 실질적으로 유사한 컴포넌트를 나타낸다. 다른 실시예들이 이용될 수 있으며, 구조적, 논리적, 전기적 변경이 이루어질 수 있다. 더욱이, 다양한 실시예들은 상이하지만 서로 배타적일 필요는 없다는 것을 이해해야 한다. 예를 들어, 일 실시예에서 설명되는 특정 특징, 구조 또는 특성은 다른 실시예들에 포함될 수 있다.

<37> 또한, 프로세서에서 구현되는 실시예의 설계는 고안에서 시뮬레이션을 거쳐 제조까지 다양한 단계를 거칠 수 있다. 설계를 나타내는 데이터는 다수의 방식으로 설계를 표현할 수 있다. 먼저, 시뮬레이션에서 유용하듯이, 하드웨어는 하드웨어 기술 언어 또는 다른 기능 기술 언어를 이용하여 표현될 수 있다. 또한, 논리 및/또는 트랜지스터 게이트들을 갖춘 회로 레벨 모델이 설계 프로세스의 몇몇 단계에서 생성될 수 있다. 또한, 대부분의 설계는, 일부 단계에서, 하드웨어 모델에서 다양한 장치의 물리적 배치를 표현하는 데이터의 레벨에 이른다. 통상의 반도체 제조 기술이 이용되는 경우에, 하드웨어 모델을 표현하는 데이터는 집적 회로를 생성하기 위하여 사용되는 마스크들의 상이한 마스크 층들 상의 다양한 특징(features)의 유무를 지정하는 데이터일 수 있다. 임의의 설계 표현에 있어서, 데이터는 임의 형태의 머신 판독 가능 매체에 저장될 수 있다. 이러한 정보를 전송하기 위해 변조되거나 생성되는 광학 또는 전기 파(wave), 메모리, 또는 디스크와 같은 자기 또는 광학 저장 장치가 머신 판독 가능 매체일 수 있다. 임의의 이들 매체는 설계 또는 소프트웨어 정보를 "운반" 또는 "지시"할 수 있다. 코드 또는 설계를 지시 또는 운반하는 전기 반송파가 전송될 때, 전기 신호의 복사, 버퍼링 또는 재전송이 수행되는 정도까지 새로운 사본이 만들어진다. 따라서, 통신 제공자 또는 네트워크 제공자는 실시예를 구성 또는 표현하는 물건(반송파)의 사본들을 만들 수 있다.

<38> 실시예들은 머신에 의해 액세스될 때 머신이 청구되는 발명에 따라 프로세스를 수행하게 할 수 있는 데이터를 저장한 머신 판독 가능 매체를 포함할 수 있는 프로그램 제품으로서 제공될 수 있다. 머신 판독 가능 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 광자기 디스크, ROM, RAM, EPROM, EEPROM, 자기 또는 광학 카드, 플래시 메모리, 또는 전자 명령을 저장하기에 적합한 다른 타입의 매체/머신 판독 가능 매체를 포함할 수 있지만, 이에 한정되지 않는다. 더욱이, 실시예들은 또한 프로그램 제품으로서 다운로드될 수 있고, 프로그램은 통신 링크(예를 들어, 모뎀 또는 네트워크 접속)를 경유하여 반송파 또는 다른 전파 매체 내에 구현된 데이터 신호를 통해 원격 데이터 소스에서 요청 장치로 전송될 수 있다.

<39> 다수의 방법은 그들의 가장 기본적인 형태로 설명되었지만, 청구되는 발명의 기본 범위를 벗어나지 않고, 임의 방법들로부터 단계들이 추가 또는 삭제될 수 있으며, 임의의 설명되는 메시지로부터 정보가 추가 또는 삭제될 수 있다. 많은 추가적인 수정 및 적응이 이루어질 수 있음은 이 분야의 전문가들에게 자명할 것이다. 특정 실시예들은 청구되는 발명을 한정하려고 제공되는 것이 아니라 설명하기 위해 제공되는 것이다. 청구되는 발명의

범위는 위에 제공되는 특정 예들에 의해 결정되는 것이 아니라 아래의 청구범위에 의해서만 결정된다.

도면의 간단한 설명

<7> 도 1은 일 실시예에서의 가상화된 환경의 하이 레벨 블록도.

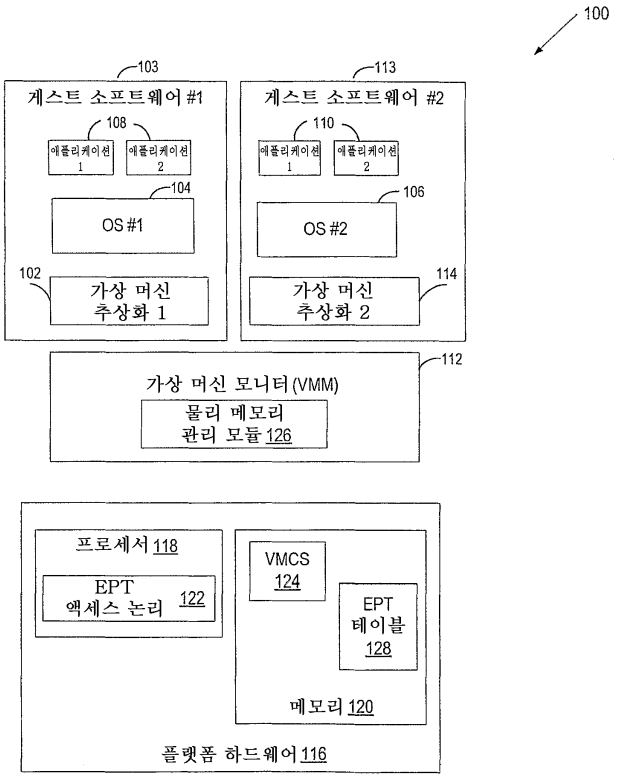
<8> 도 2는 일 실시예에서의 가상화된 환경의 동작의 하이 레벨 흐름도.

<9> 도 3은 일 실시예에서의 처리의 하이 레벨 흐름도.

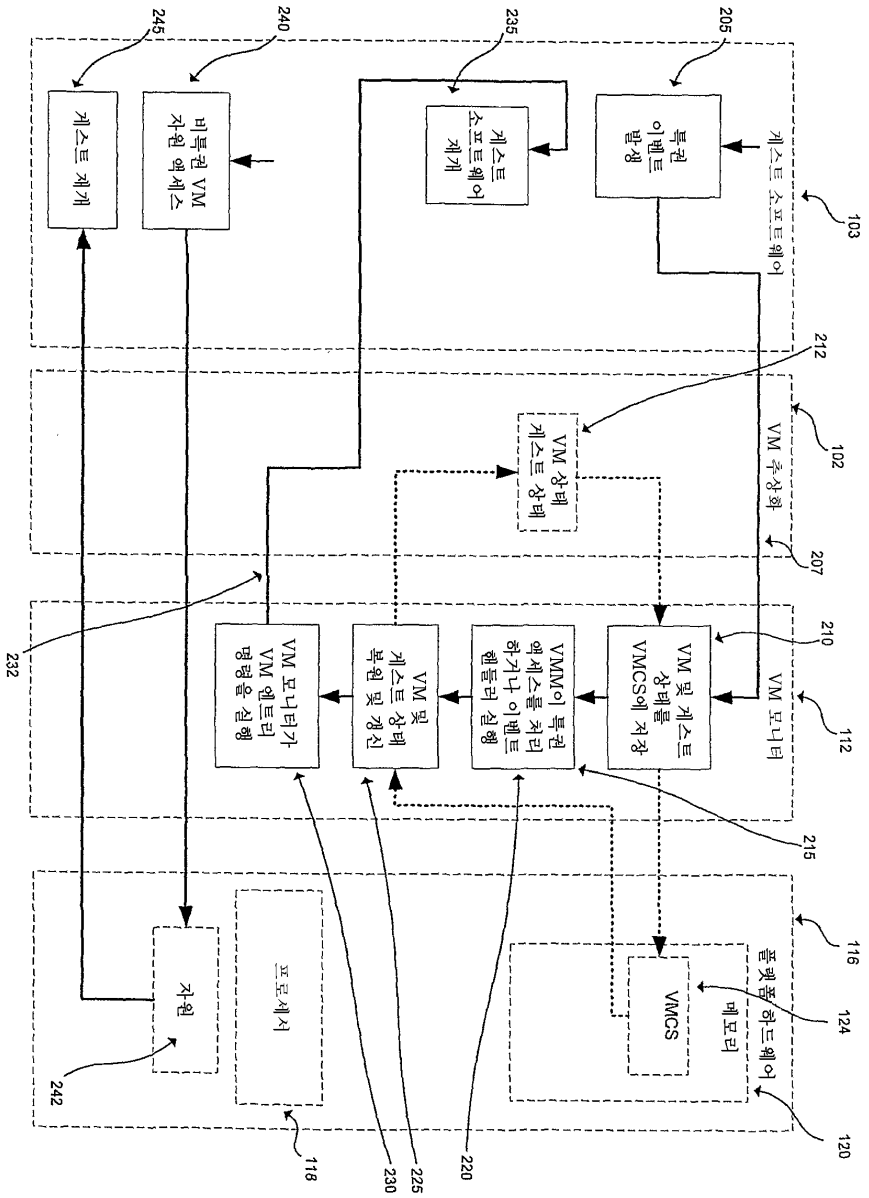
<10> 도 4는 일 실시예에서의 처리의 하이 레벨 흐름도.

도면

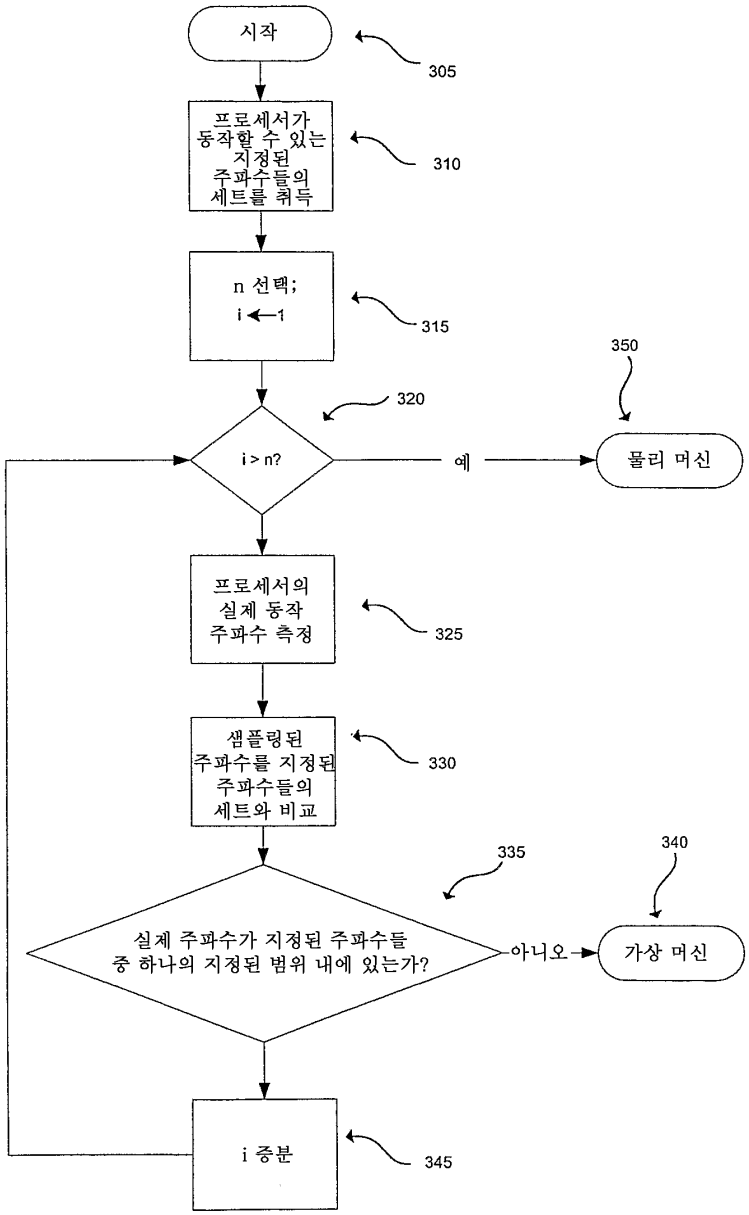
도면1



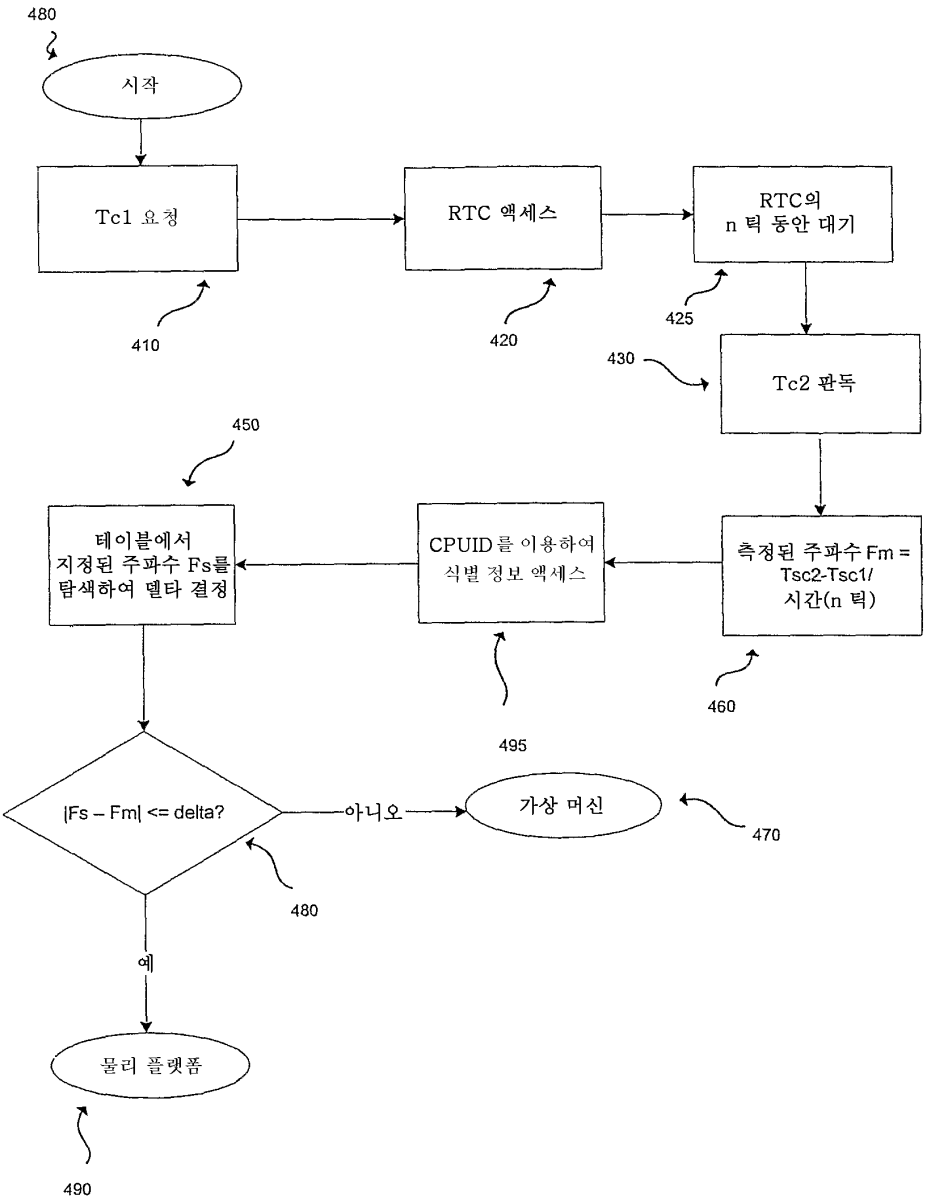
도면2


게스트 소프트웨어
103
특권 이벤트 발생
205
게스트 소프트웨어 게게
235
비특권 VM 자원 액세스
240
게스트 게게
245
VM 추상화
102
VM 상태 게스트 상태
212
207
VM 모니터
112
VM 및 게스트 상태를 VMCS에 저장
210
VMM이 특권 액세스를 처리하거나 이벤트 핸들러 실행
VM 및 게스트 상태 복원 및 갱신
VM 모니터가 VM 엔트리 명령을 실행
232
215
플랫폼 하드웨어
120
VMCS
124
메모리
프로세서
자원
242
116
118
230
225
220

도면3


시작
305
프로세서가 동작할 수 있는 지정된 주파수들의 세트를 취득
310
n 선택;
i ← 1
315
320
i > n?
예
350
물리 머신
프로세서의 실제 동작 주파수 측정
325
샘플링된 주파수를 지정된 주파수들의 세트와 비교
330
335
340
실제 주파수가 지정된 주파수들 중 하나의 지정된 범위 내에 있는가?
아니오
가상 머신
i 증분
345

도면4


480
시작
Tc1 요청
410
RTC 액세스
420
RTC의
n 틱 동안 대기
425
430
Tc2 판독
450
테이블에서
지정된 주파수 Fs를
탐색하여 델타 결정
CPUID를 이용하여
식별 정보 액세스
측정된 주파수 Fm =
Tsc2-Tsc1/
시간(n 틱)
460
495
|Fs – Fm| <= delta?
아니오
가상 머신
470
480
예
물리 플랫폼
490