(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2015년06월02일
(11) 등록번호 10-1524450
(24) 등록일자 2015년05월26일

(51) 국제특허분류(Int. Cl.)
G06F 9/30 (2006.01)
(21) 출원번호 10-2013-7008232
(22) 출원일자(국제) 2011년09월23일
심사청구일자 2013년03월29일
(85) 번역문제출일자 2013년03월29일
(65) 공개번호 10-2013-0064797
(43) 공개일자 2013년06월18일
(86) 국제출원번호 PCT/US2011/052913
(87) 국제공개번호 WO 2012/040552
국제공개일자 2012년03월29일
(30) 우선권주장
12/890,571 2010년09월24일 미국(US)
(56) 선행기술조사문헌
US07464255 B1

(73) 특허권자
인텔 코포레이션
미합중국 캘리포니아 95052 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자
포시스 앤드류 티
미국 워싱턴주 98034 커클랜드 노스이스트 137번 스트리트 6841
(74) 대리인
제일특허법인

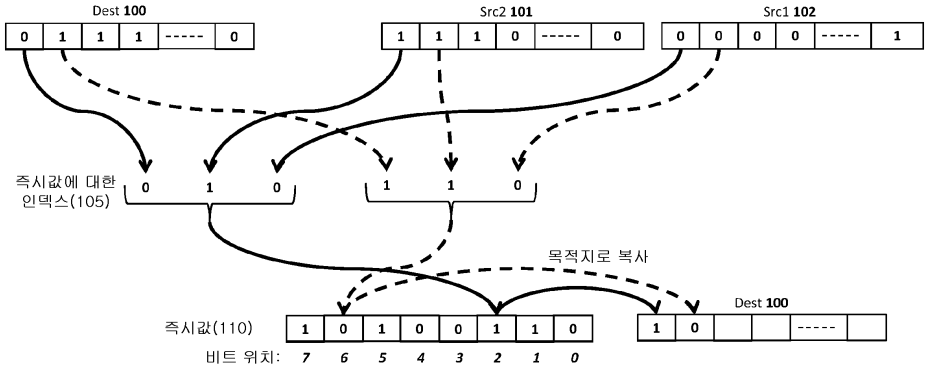
전체 청구항 수 : 총 27 항 심사관 : 지정훈

(54) 발명의 명칭 **범용 논리 연산 방법 및 장치**

(57) 요 약

테이블에 의해 명시된 임의의 논리 연산을 수행하는 장치 및 방법을 설명한다. 예를 들어, 컴퓨터 프로세서에서 논리 연산을 수행하는 방법의 일 실시예는, 두개 이상의 소스 오퍼랜드 각각으로부터 데이터를 판독하는 단계와, 소스 오퍼랜드로부터 판독된 데이터를 조합하여 인덱스 값을 생성하되, 인덱스 값은 명령과 함께 전송된 즉시 값 내의 비트의 서브세트를 식별하는 단계와, 즉시 값으로부터 비트를 판독하는 단계와, 즉시 값으로부터 판독된 비트를 목적지 레지스터 내에 저장하여 명령의 결과를 생성하는 단계를 포함한다.

대 표 도



명 세 서

청구범위

청구항 1

컴퓨터 프로세서에서 논리 연산을 수행하는 방법에 있어서,

두 개 이상의 소스 오퍼랜드들(operands)의 각각으로부터 데이터를 판독하는 단계와,

상기 두 개 이상의 소스 오퍼랜드들의 각각의 비트 위치에 대하여:

상기 두 개 이상의 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트와 상기 두 개 이상의 소스 오퍼랜드들 중 다른 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트를 조합하여 인덱스 값을 생성하는 단계 - 상기 인덱스 값은 명령과 연관된 즉시 값(immediate value) 내의 단일 비트를 식별함 - 및

상기 인덱스 값에 의해 식별된 상기 단일 비트를 상기 즉시 값으로부터 판독하는 단계와,

상기 명령의 결과를 생성하기 위해 상기 즉시 값으로부터 판독된 각각의 상기 단일 비트를 목적지 레지스터 내에 저장하는 단계를 포함하는

방법.

청구항 2

제 1 항에 있어서,

상기 두 개 이상의 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트와 상기 두 개 이상의 소스 오퍼랜드들 중 다른 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트를 조합하여 인덱스 값을 생성하는 단계는,

각각의 소스 오퍼랜드로부터 판독된 해당 비트들을 연결(concatenating)하여 상기 인덱스 값을 생성하는 단계를 포함하되,

연결된 상기 해당 비트들의 세트로 구성되는 상기 인덱스 값은 상기 즉시 값 내의 비트 위치를 식별하는

방법.

청구항 3

제 2 항에 있어서,

상기 즉시 값의 상기 식별된 비트 위치로부터 상기 단일 비트를 판독하는 단계와,

각각의 소스 오퍼랜드로부터 판독된 상기 해당 비트들의 비트 위치에 해당하는 비트 위치에서 상기 목적지 레지스터에 결과 비트 값을 저장하는 단계를 더 포함하는

방법.

청구항 4

제 2 항에 있어서,

상기 두 개 이상의 소스 오퍼랜드들은 N 비트 소스 레지스터들에 저장되고, 상기 목적지 레지스터는 N 비트 목적지 레지스터이며,

상기 N은 양수이고,

각각의 소스 오퍼랜드로부터 판독된 해당 비트들을 연결하여 상기 인덱스 값을 생성하는 단계는, 상기 두 개 이

상의 소스 오퍼랜드들의 모든 N 비트 위치들에서 동시에 수행되어, 상기 즉시 값에 대한 N 개의 인덱스들을 형성하며,

상기 N 개의 인덱스들은 상기 즉시 값 내의 N 비트 위치들을 식별하고,

상기 즉시 값으로부터의 모든 N 개의 인덱싱된 비트는 상기 목적지 레지스터에 동시에 복사되어 상기 명령의 결과를 형성하는

방법.

청구항 5

제 4 항에 있어서,

상기 N은 64인

방법.

청구항 6

제 1 항에 있어서,

상기 두 개 이상의 소스 오퍼랜드들은 3개의 소스 오퍼랜드들로 구성되고, 상기 즉시 값은 8 비트이며,

상기 방법은,

상기 3개의 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트들을 연결하여 상기 인덱스 값을 생성하는 단계를 더 포함하되,

대응하는 비트 위치들로부터의 3개의 연결된 해당 비트들의 세트로 구성되는 상기 인덱스 값은 상기 8 비트 즉시 값 내의 비트 위치를 식별하는

방법.

청구항 7

제 1 항에 있어서,

상기 즉시 값은 상기 명령과 함께 송신된 8 비트 값인

방법.

청구항 8

제 7 항에 있어서,

상기 8 비트 값은 imm8 값인

방법.

청구항 9

제 1 항에 있어서,

상기 즉시 값은 상기 명령과 함께 송신된 16 비트 값인

방법.

청구항 10

명령을 처리하는 장치로서,

명령을 디코딩하고, 상기 명령에 의해 수행될 논리 연산을 식별하는 디코더부와,

상기 논리 연산을 실행하는 실행부를 포함하되,

상기 실행부는,

두 개 이상의 소스 오퍼랜드들 각각으로부터 데이터를 판독하는 동작과,

상기 두 개 이상의 소스 오퍼랜드들의 각각의 비트 위치에 대하여:

상기 두 개 이상의 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트와 상기 두 개 이상의 소스 오퍼랜드들 중 다른 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트를 조합하여 인덱스 값을 생성하는 동작 - 상기 인덱스 값은 명령과 연관된 즉시 값(immediate value) 내의 단일 비트를 식별함 - 및

상기 인덱스 값에 의해 식별된 상기 단일 비트를 상기 즉시 값으로부터 판독하는 동작과,

상기 명령의 결과를 생성하기 위해 상기 즉시 값으로부터 판독된 각각의 상기 단일 비트를 목적지 레지스터 내에 저장하는 동작을 수행하는

장치.

청구항 11

제 10 항에 있어서,

상기 두 개 이상의 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트와 상기 두 개 이상의 소스 오퍼랜드들 중 다른 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트를 조합하여 인덱스 값을 생성하는 동작은,

각각의 소스 오퍼랜드로부터 판독된 해당 비트들을 연결하여 상기 인덱스 값을 생성하는 동작을 포함하되,

연결된 상기 해당 비트들의 세트로 구성되는 상기 인덱스 값은 상기 즉시 값 내의 비트 위치를 식별하는

장치.

청구항 12

제 11 항에 있어서,

상기 실행부는,

상기 즉시 값의 상기 식별된 비트 위치로부터 상기 단일 비트를 판독하는 추가 동작과,

각각의 소스 오퍼랜드로부터 판독된 상기 해당 비트들의 비트 위치에 해당하는 비트 위치에서 상기 목적지 레지스터에 결과 비트 값을 저장하는 추가 동작을 수행하는

장치.

청구항 13

제 11 항에 있어서,

상기 두 개 이상의 소스 오퍼랜드들은 N 비트 소스 레지스터들에 저장되고, 상기 목적지 레지스터는 N 비트 목적지 레지스터이며,

상기 N은 양수이고,

각각의 소스 오퍼랜드로부터 판독된 해당 비트들을 연결하여 상기 인덱스 값을 생성하는 동작은, 상기 두 개 이상의 소스 오퍼랜드들의 모든 N 비트 위치들에서 동시에 수행되어, 상기 즉시 값에 대한 N 개의 인덱스들을 형성하며,

상기 N 개의 인덱스들은 상기 즉시 값 내의 N 비트 위치들을 식별하고,

상기 즉시 값으로부터의 모든 N 개의 인덱싱된 비트는 상기 목적지 레지스터에 동시에 복사되어 상기 명령의 결과를 형성하는

장치.

청구항 14

제 13 항에 있어서,

상기 N은 64인

장치.

청구항 15

제 10 항에 있어서,

상기 두 개 이상의 소스 오퍼랜드들은 3개의 소스 오퍼랜드들로 구성되고, 상기 즉시 값은 8 비트이며,

상기 3개의 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트들을 연결하여 상기 인덱스 값을 생성하는 동작을 더 포함하되, 대응하는 비트 위치들로부터의 3개의 연결된 비트들의 세트로 구성되는 상기 인덱스 값은 상기 8 비트 즉시 값 내의 비트 위치를 식별하는

장치.

청구항 16

제 10 항에 있어서,

상기 즉시 값은 상기 명령과 함께 송신된 8 비트 값인

장치.

청구항 17

제 16 항에 있어서,

상기 8 비트 값은 imm8 값인

장치.

청구항 18

제 10 항에 있어서,

상기 즉시 값은 상기 명령과 함께 송신된 16 비트 값인

장치.

청구항 19

표시 장치와,

명령을 저장하는 메모리와,

상기 명령을 처리하는 프로세서를 포함하되,

상기 프로세서는,

명령을 디코딩하고, 상기 명령에 의해 수행될 논리 연산을 식별하는 디코더부와,

상기 논리 연산을 실행하는 하드웨어 실행부를 포함하되,

상기 실행부는,

두 개 이상의 소스 오퍼랜드들의 각각으로부터 데이터를 판독하는 동작과,

상기 두 개 이상의 소스 오퍼랜드들의 각각의 비트 위치에 대하여:

상기 두 개 이상의 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트와 상기 두 개 이상의 소스 오퍼랜드들 중 다른 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트를 조합하여 인덱스 값을 생성하는 동작 - 상기 인덱스 값은 명령과 연관된 즉시 값(immediate value) 내의 단일 비트를 식별함 - 및

상기 인덱스 값에 의해 식별된 상기 단일 비트를 상기 즉시 값으로부터 판독하는 동작과,

상기 명령의 결과를 생성하기 위해 상기 즉시 값으로부터 판독된 각각의 상기 단일 비트를 목적지 레지스터 내에 저장하는 동작을 수행하는

컴퓨터 시스템.

청구항 20

제 19 항에 있어서,

상기 두 개 이상의 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트와 상기 두 개 이상의 소스 오퍼랜드들 중 다른 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트를 조합하여 인덱스 값을 생성하는 동작은,

각각의 소스 오퍼랜드로부터 판독된 해당 비트들을 연결하여 상기 인덱스 값을 생성하는 동작을 포함하되,

연결된 상기 해당 비트들의 세트로 구성되는 상기 인덱스 값은 상기 즉시 값 내의 비트 위치를 식별하는

컴퓨터 시스템.

청구항 21

제 20 항에 있어서,

상기 실행부는,

상기 즉시 값의 상기 식별된 비트 위치로부터 상기 단일 비트를 판독하는 추가 동작과,

각각의 소스 오퍼랜드로부터 판독된 상기 해당 비트들의 비트 위치에 해당하는 비트 위치에서 상기 목적지 레지스터에 결과 비트 값을 저장하는 추가 동작을 수행하는

컴퓨터 시스템.

청구항 22

제 20 항에 있어서,

상기 두 개 이상의 소스 오퍼랜드들은 N 비트 소스 레지스터들에 저장되고, 상기 목적지 레지스터는 N 비트 목적지 레지스터이며,

상기 N은 양수이고,

각각의 소스 오퍼랜드로부터 판독된 해당 비트들을 연결하여 상기 인덱스 값을 생성하는 동작은, 상기 두 개 이상의 소스 오퍼랜드들의 모든 N 비트 위치들에서 동시에 수행되어, 상기 즉시 값에 대한 N 개의 인덱스들을 형성하며,

상기 N 개의 인덱스들은 상기 즉시 값 내의 N 비트 위치들을 식별하고,

상기 즉시 값으로부터의 모든 N 개의 인덱싱된 비트는 상기 목적지 레지스터에 동시에 복사되어 상기 명령의 결과를 형성하는

컴퓨터 시스템.

청구항 23

제 22 항에 있어서,

상기 N은 64인

컴퓨터 시스템.

청구항 24

제 19 항에 있어서,

상기 두 개 이상의 소스 오퍼랜드들은 3개의 소스 오퍼랜드들로 구성되고, 상기 즉시 값은 8 비트이며,

상기 3개의 소스 오퍼랜드들의 각각으로부터 판독된 해당 비트들을 연결하여 상기 인덱스 값을 생성하는 동작을 더 포함하되, 대응하는 비트 위치들로부터의 3개의 연결된 비트들의 세트로 구성되는 상기 인덱스 값은 상기 8 비트 즉시 값 내의 비트 위치를 식별하는

컴퓨터 시스템.

청구항 25

제 19 항에 있어서,

상기 즉시 값은 상기 명령과 함께 송신된 8 비트 값인

컴퓨터 시스템.

청구항 26

제 25 항에 있어서,

상기 8 비트 값은 imm8 값인

컴퓨터 시스템.

청구항 27

제 19 항에 있어서,

상기 즉시 값은 상기 명령과 함께 송신된 16 비트 값인

컴퓨터 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명은 일반적으로 컴퓨터 프로세서 분야에 관한 것이다. 보다 구체적으로, 본 발명은 컴퓨터 프로세서에서 범용 논리 연산을 제공하는 장치 및 방법에 관한 것이다.

배 경 기 술

[0002] 컴퓨터 프로세서는 수리 연산과 논리 연산을 수행하기 위해 명령을 실행한다. 예를 들어, 수리 연산은 상이한 레벨의 정확성을 갖는 부동 소숫점 및 정수 가산, 감산, 승산, 제산을 포함한다. 논리 연산은 AND, OR, NAND, NOR, XOR, 좌/우 이동, 스위즐(swizzle), 선택 및 표결 등을 포함한다.

[0003] 특정한 상황(특히 3개 이상의 오퍼랜드가 수반될 때)에서, 비교적 간단한 논리 연산이 상당한 수의 명령의 실행을 필요로 할 수 있다. 예로서, "dest = (src1 AND dest) OR (src2 AND (NOT dest))"와 같은 논리는, 목적지 레지스터(dest) 내의 하나의 특정한 비트가 설정되면, 소스 레지스터 1(src1)이 선택되어야 하고, 그렇지 않으면 소스 레지스터 2(src2)가 선택되어야 한다는 것을 명시할 필요가 있다. 기존의 기법을 이용하여 이 연산을 수행하는 것은 다수의 상이한 명령(2개의 AND 연산, 1개의 OR 연산, 1개의 NOT 연산)의 실행을 필요로 한다. 여러 다른 비교적 간단한 기능은 기존의 기법을 사용하는 다수의 명령 시퀀스로 구현되어야 한다.

발명의 내용

해결하려는 과제

[0004] 따라서, 보다 적은 명령의 실행을 요구하는 논리 연산을 실행하는 새로운 기법이 유용할 것이다.

도면의 간단한 설명

[0005] 이하의 도면과 함께 후술하는 상세한 설명으로부터 본 발명을 보다 명확하게 이해할 수 있다.

도 1은 즉시 값의 인덱스가 3개의 소스 오퍼랜드로부터 생성되는 본 발명의 일 실시예를 도시한 도면,

도 2는 논리 연산을 수행하기 위한 즉시 값을 인덱싱하는 방법의 일 실시예를 도시한 도면,

도 3은 논리 연산을 수행하기 위해 즉시 값을 인덱싱하는 방법의 다른 실시예를 도시한 도면,

도 4는 본 발명의 일 실시예에서 채용되는 프로세서 구조를 도시한 도면,

도 5는 본 발명의 일 실시예에서 채용되는 시스템 구조를 도시한 도면,

도 6은 본 발명의 일 실시예에 따른 복수의 처리 요소를 도시한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0006] 설명을 위해, 후술하는 설명에서는 후술하는 본 발명의 실시예를 철저히 이해하도록 여러 특정한 세부 사항을 기술한다. 하지만, 당업자에게 이들 특정한 세부 사항의 일부가 없이 본 발명의 실시예가 실시될 수 있다는 것은 명확할 것이다. 다른 예에서, 본 발명의 실시예의 기본 원리가 모호하게 하는 것을 방지하기 위해 주지된 구조와 장치는 블록도로 도시한다.

[0007] 즉시 값은 명령과 연관된(예를 들어, 명백하게 명령의 일부분으로 포함되어지거나 명령과 조합된) 이진값이다. x86 명령 세트와 연관된 주지된 즉시 값은 8-비트 즉시 분류된 imm8이다. 명령에 대한 즉시 값은 컴파일시 명령과 함께 생성된다. 후술된 본 발명의 실시예는 imm8 혹은 imm16 즉시 값을 이용하지만, 본 발명의 기본 원리는 즉시 값의 어느 특정한 유형으로 제한되지 않는다. 실제, 본 발명의 기본 원리는 명령과 연관된 임의 유형의 이진 코드를 이용하여 구현될 것이다.

[0008] 본 발명의 일 실시예에서, 상이한 논리 연산(예를 들어, AND, NAND, XOR, 인버트, 선택, 표결 등)을 식별하는데 상이한 즉시 값이 이용된다. 본 실시예에서, 명령의 소스 오퍼랜드로부터의 특정한 비트를 조합하여, 명령의 즉시 값 내의 비트 위치를 식별하는 인덱스 값을 형성한다. 식별된 비트 위치로부터 판독된 비트가 목적지 레지스터에서 조합되어 논리 연산의 결과에 이른다. 즉, 즉시 값을 룩업 테이블로서 이용하여 각 논리 연산에 대한 결과에 도달한다.

[0009] 도 1은 소스 레지스터 1(src1)(102), 소스 레지스터 2(src2)(101), 및 목적지 레지스터(dest)(100)에 저장된 3개의 오퍼랜드로부터 비트를 판독하여, 즉시 값(110) 내의 비트 위치를 식별하는 인덱스(105)를 형성하는 특정한 일예를 도시한다. 식별된 비트 위치에서 이진값은 목적지 레지스터(100)에 복사된다. 예시를 위해 이 특정한 레지스터 배열을 이용하지만, 본 발명의 기본 원리는 임의의 특정 레지스터 세트로 제한되지 않는다. 예를 들어, 소스 오퍼랜드를 저장하는 데 "목적지" 레지스터를 이용하는 대신, 제 3의 전용 소스 레지스터(예를 들어, "소스 레지스터 3")를 이용할 수 있다.

[0010] 도 1에 도시된 특정한 예에서, 목적지 레지스터(100)와 소스 레지스터(101, 102)의 제 1 비트 위치로부터의 비트를 처음에 판독하여, 도시된(실선 화살표로 지시된) 바와 같이 인덱스 010을 형성한다. 일 실시예에서, 3개의 소스값의 비트별 연결로부터 인덱스를 형성한다. 인덱스 010은 즉시 값 110의 비트 위치 2를 식별한다(즉, 이진값 010은 2와 같다). 도시된 바와 같이, 비트 위치 2로부터의 비트는 즉시 값(도시된 예시에서의 이진값 1)으로부터 판독되어 제 1 비트 위치에서 목적지 레지스터에 저장된다. 모든 비트가 판독되어 인덱싱된 비트값이 즉시 값으로부터 목적지 레지스터에 복사될 때까지 소스 오퍼랜드 각각의 비트 각각에 대해 동일한 처리가 구현된다. 예를 들어 도 1에 도시된 바와 같이, 목적지 레지스터 및 소스 레지스터의 제 2 비트 위치로부터의 비트를 이용하여 110의 인덱스 값이 생성되고, 이 인덱스 값은 비트 위치 6(즉, 이진값 110은 6과 같음)에서 즉시 값을 인덱싱하는 데 이용된다. 비트 위치 6으로부터의 비트는 즉시 값(이진값 0)으로부터 판독되어 목적지 레지스터의 제 2 비트 위치에 저장된다.

[0011] 본 발명의 일 실시예에 따른 방법이 도 2에 도시된다. 201에서, 오퍼랜드 레지스터(예를 들어, dest, src1, src2)의 지정된 비트 위치로부터 값이 판독된다. 다수의 값이 다수의 인덱스 값으로부터 동시에 병렬로 판독될 수 있다. 202에서 값이 조합되어 즉시 값에 대한 인덱스를 형성하고, 203에서 이 인덱스는 즉시 값 내의 비트 위치를 식별하는 데 이용된다. 상술한 바와 같이, 일 실시예에서, 소스 오퍼랜드 내의 해당 비트 위치로부터의 비트를 연결하여, 즉시 값에 대한 인덱스를 형성한다. 204에서, 인덱스 값에 의해서 식별된 비트 위치에서의 값이 판독되고, 205에서 인덱싱된 비트 위치로부터의 값은 목적지 레지스터로 복사된다.

[0012] 상술한 기법을 이용하여, 상이한 논리 연산을 구현 하는 데 상이한 즉시 값이 명시될 수 있다. 제한되지는 않지만 일 예로서, 후술하는 유형의 연산은 이하의 즉시 값을 이용하여 수행될 수 있다.

[0013] 1. 모든 것에 대해 결과 설정. 즉시 값 = 11111111. 본 예에서, 즉시 값으로부터 판독되는 임의의 비트는 목적지 레지스터에서 이진값 1의 결과를 갖게 할 것이다.

[0014] 2. 복사중 인버팅. 즉시 값 =01010101. 원래 데이터를 파괴하는 Current NOT 명령(예를 들어, 현재 x86 명령 세트에서 구현됨)은 제자리에서 인버팅한다. 그에 반해, 위 즉시 값을 이용하는 상술한 기법은 상이한 레지스터에 복사한다. 이 명령은 dest = NOT src1의 형태를 취할 수 있다.

[0015] 3. 기존 논리 연산을 모방. 즉시 값 = 10001000. 이 즉시 값은 소스 레지스터 1과 소스 레지스터 2의 값을 AND 연산(즉, dest = src1 AND src2)시킨다. 비트 0~3과 비트 7~4이 동일하기 때문에 목적지 레지스터(dest)의 값은 결과에 영향을 미치지 않는다는 점에 유의하라. 따라서, 일 실시예에서 목적지 레지스터는 전력 소모를 저감하기 위해 판독되지 않는다.

[0016] 하기의 테이블(테이블 A)는 기존의 x86 논리 연산과 동등한 imm8 값을 도시한다.

테이블 **A**

기존 연산	즉시 값
dest = NOT src1	01010101
dest = src1 OR src2	11101110
dest = src1 AND src2	10001000
dest = src1 ANDN src2	00100010
dest = src1 XOR src2	01100110

[0017]

[0018] 4. 2개의 기존 논리 연산을 조합. 즉시 값 = 01111000. 이 즉시 값은 소스 레지스터 1과 소스 레지스터 2의 AND를 목적지 레지스터의 값과 XOR된 결과를 얻게 한다. 명령은 dest = (src1 AND src2) XOR dest의 형태를 취할 수 있다. 상술한 바와 같이, 기존의 기법을 사용해 이 연산을 구현하는 유일한 방법은 많은 시간과 처리 자원을 소모하여 다수의 상이한 명령을 실행하는 것이다. 그와 반대로, 위에서 지정된 즉시 값을 이용하면 본 명세서에서 설명된 다른 연산과 동일한 시간에 결과를 제공할 수 있다.

[0019] 5. 선택. 즉시 값 = 10101100. 본 연산의 목적은, 목적지에서 해당하는 비트가 설정되는 경우 소스 레지스터 1을 선택하고, 그렇지 않으면 소스 레지스터 2를 선택하는 것이다. 종래에, 이 세계는 dest = (src1 AND dest) OR (src2 AND (NOT dest))와 같은 논리를 필요로 하고, 이것은 엄청난 수의 명령을 필요로 한다. 그와 반대로, 위에서 지정된 즉시 값을 이용하면 본 명세서에 명시된 다른 즉시 값 연관만큼 효율적으로 결과를 제공할 수 있다.

[0020] 6. 표결. 즉시 값 = 11101000. 본 연산을 이용하면, 3개의 입력 중 가장 인기 있는 값이 선택된다. 예를 들어, 목적지 레지스터에 있는 비트가 0이라면, 소스 레지스터 1에 있는 비트는 1이고, 소스 레지스터 2에 있는 비트는 0이며, 0이 가장 인기 있는 출력이다. 따라서, 0이 선택되어 목적지 레지스터에 저장된다.

[0021] 상기 연산은 단지 설명할 목적으로 제공되었다는 점에 유의해야 한다. 발명의 기본 원리는 임의의 특정한 연산 세트로 한정되지 않는다. 가상적으로 비제한된 수의 논리 연산은 본 발명의 기본 원리에 따라 상이한 즉시 값을 이용하여 구현될 수 있다(8비트 즉시 값이더라도, 단지 256개의 논리 연산이 가능하다).

[0022] 이하의 의사 코드는 64비트 오퍼랜드를 구비하는 프로세서 파이프라인에서 imm8의 즉시 값을 이용하여 구현될 때의 본 발명의 일 실시예를 나타낸다.

```
for (i = 0; i < 64; i++) {
  index = (dest[i]<<2) | (src2[i]<<1) | (src1[i]);
  dest[i] = imm8[index];
}
```

[0023]

[0024] 본 구현에서, 비트 위치 i(0~63)에서 목적지 레지스터(dest), 소스 레지스터 1(src1), 및 소스 레지스터 2(src2)로부터의 비트값 각각을 연결하여, 인덱스 값을 계산한다. <<2와 <<1 연산의 효과는 각기 비트를 좌측으로 2와 1만큼 이동하는 것이며, 그렇게 함으로써 비트를 정렬하여 인덱스를 적절하게 형성한다. 예를 들어, 오퍼랜드의 각각으로부터의 비트값이 1이라면, dest[i]<<2는 100이고, src2[i]<<1의 결과는 010이고, src1[i]의 결과는 001이다. 그 결과에 대해 비트 단위의 OR 연산이 수행된다(| 연산자에 의해서 지시되는 바와 같음). 최종 결과는 imm8 내의 특정한 비트를 식별하는 인덱스 값이다. 바깥쪽 for() 루프는 단지 의사 코드 설명용이며, 일 실시예에서 모든 64개의 연산이 하드웨어의 분리된 복사본으로 동시에 수행되는 점에 유의하라.

[0025] 이하의 의사 코드는, 각기 16개의 32비트 패키징된 데이터 요소를 저장하는 512비트 레지스터와 imm8의 즉시 값을 활용하여 프로세서 파이프라인으로 구현될 때의 본 발명의 다른 실시예를 나타낸다.

```
for (n = 0; n < 16; n++) {
 if(mask[n] != 0) {
   for (j = 0; j < 32; j++) {
      i = 32*n + j;
      index = (dest[i]<<2) | (src2[i]<<1) | (src1[i]);
      dest[i] = imm8[index];
    }
 }
}
```

[0026]

[0027] 본 실시예에서, 마스크 레지스터는 16개의 32비트 패키징된 데이터 요소 각각과 연관된 비트를 저장하는 데 이용된다. 마스크[n] != 0 테스트는 데이터 요소와 연관된 마스크 비트가 0인 경우, 데이터 요소에 대해 목적지 레지스터(dest)에 저장된 현재값이 변하지 않고 남아 있다는 것을 가리킨다. 하지만 만약 마스크 비트가 0이 아니라면, 다음 FOR 루프가 수행되고, 그 결과 (상술한 바와 같이) 그 데이터 요소에 대해 인덱스 값이 산출된다. 다시, for() 루프가 도시되고, 일 실시예에서 512개의 연산이 동시에 수행된다.

[0028] 상술한 본 발명의 실시예가 3개의 소스 오퍼랜드로부터 수행되지만, 본 발명의 기본 원리는 임의의 수의 오퍼랜드를 사용하여 구현될 수 있다. 예를 들어, 후술하는 의사 코드는 어떻게 4개의 오퍼랜드(3개의 소스와 목적지)가 16비트 즉시 값(imm16)으로 인덱스 값을 생성시키는 데 이용되는지를 나타낸다.

```
for (n = 0; n < 16; n++) {
 for (j = 0; j < 32; j++) {
    i = 32*n + j;
    index = (dest[i]<<3) | (src3[i]<<2)
          | (src2[i]<<1) | (src1[i]);
    dest[i] = imm16[index];
 }
}
```

[0029]

[0030] 대신에, 소스 중 하나는 마스크 레지스터로부터 올 수도 있다.

```
for (n = 0; n < 16; n++) {
 for (j = 0; j < 32; j++) {
    i = 32*n + j;
    index = (mask[n]<<3) | (dest[i]<<2)
          | (src2[i]<<1) | (src1[i]);
    dest[i] = imm16[index];
 }
}
```

[0031]

[0032] 상기 코드에서 지시된 바와 같이, 본 실시예에서, 제 4 오퍼랜드는 마스크 레지스터에 저장될 수 있다. 마스크 레지스터(3만큼 왼쪽으로 이동됨), 목적지 레지스터(2만큼 왼쪽으로 이동됨), 소스 레지스터 2(1만큼 왼쪽으로 이동됨) 및 소스 레지스터 1로부터 해당 비트에 대해 비트 단위 OR을 수행함으로써 인덱스 값이 생성된다. 결과적인 4비트값은 16비트 즉시 값 imm16에 대한 룩업으로서 사용된다.

[0033] 바람직한 프로세서 구조

[0034] 도 3은 본 발명의 실시예가 시행될 수 있는 바람직한 처리 코어(300)를 도시한다. 일반적인 처리 코어(300)는 CISC(Complex Instruction Set), RISC(Reduce Instruction Set) 및 VLIW(Very Long Instruction Word)와 같은 많은 상이한 유형의 처리 코어 구조를 기술한다고 생각한다. 도면의 일반적인 처리 코어(300)는 1)명령을 페치

하는 페치부(303)(예를 들어, 캐시 및/또는 메모리로부터), 2)명령을 디코딩하는 디코딩부(304), 3)실행부(306)에 대한 명령의 발행 타이밍 및/또는 순서를 판정하는 스케쥴부(305)(특히 스케줄러는 옵션임), 4)명령을 실행하는 실행부(306)(전형적인 명령 실행부는 분기 실행부, 정수 연산 실행부(예를 들어, ALU), 부동 소숫점 연산 실행부(예를 들어 FPU) 및 메모리 액세스 실행부를 포함함), 5)명령의 성공적인 종료를 나타내는 실행종료부(307)를 포함한다. 특히, 처리 코어(300)는 마이크로코드(308)를 채용하거나 채용하지 않을 수 있다. 마스크 레지스터(302)는 상술한 바와 같이 본 발명의 실시예에 따라 이용될 수 있다.

[0035] 바람직한 컴퓨터 시스템

[0036] 이하, 본 명세서에서 상세하게 설명한 명령을 실행하는 데 적합한 바람직한 시스템이다. 랩톱, 데스크톱, 핸드헬드 PC, PDA(personal digital assistant), 엔지니어링 워크스테이션, 서버, 네트워크 장치, 네트워크 허브, 스위치, 임베디드 프로세서, DSP(digital signal processor), 그래픽 장치, 비디오 게임 장치, 셋톱 박스, 마이크로 컨트롤러, 핸드폰, PMP(portable media player), 핸드헬드 장치, 및 다양한 다른 전자 장치에 대한 분야에서 주지된 다른 시스템 디자인 및 구성도 적합하다. 일반적으로, 본 명세서에서 설명한 바와 같은 프로세서 및/또는 기타 실행 논리를 병합할 수 있는 다양한 시스템 및 전자 장치도 적합하다.

[0037] 도 4를 참조하면, 본 발명의 일 실시예에 따른 시스템(400)의 블록도가 도시된다. 시스템(400)은 그래픽 메모리 컨트롤러 허브(GMCH : graphic memory controller hub)(420)와 연결되는 하나 이상의 처리 요소(410, 415)를 포함할 수 있다. 추가적인 처리 요소(415)의 옵션 종류는 도 4에 점선으로 표시되어 있다.

[0038] 각각의 처리 요소는 단일 코어일 수도 있고, 또는 그 대신에 다수의 코어를 포함할 수 있다. 처리 요소는 집적된 메모리 컨트롤러 및/또는 집적된 I/O 제어 논리와 같은 처리 코어 옆에 다른 온다이(on-die) 요소를 옵션으로 포함할 수 있다. 또한 적어도 일 실시예에 대해, 처리 요소의 코어는 그들이 코어당 하나의 하드웨어 스레드 컨텍스트 보다 많이 포함할 수 있는 다중 스레드될 수 있다.

[0039] 도 4는 GMCH(420)가 예를 들어 DRAM(dynamic random access memory)일 수 있는 메모리(440)에 연결될 수 있다는 것이 도시된 도면이다. DRAM은, 적어도 일 실시예에 대해, 비휘발성 캐시와 연관될 수 있다.

[0040] GMCH(420)는 칩셋 또는 칩셋의 일부일 수 있다. GMCH(420)는 프로세서(410, 415)와 통신하여, 프로세서(410, 415)와 메모리(440) 사이의 상호 작용을 제어할 수 있다. GMCH(420)는 또한 프로세서(410, 415)와 시스템(400)의 다른 요소 사이의 가속 버스 인터페이스로서 활동할 수 있다. 적어도 일 실시예에 관하여, GMCH(420)는 FSB(frontside bus)(495)와 같은 다중-드롭 버스를 통해 프로세서(410, 415)와 통신한다.

[0041] 또한, GMCH(420)는 표시 장치(440)(예를 들어, 플랫 패널 디스플레이)와 연결된다. GMCH(420)는 집적된 그래픽 가속기를 포함할 수 있다. GMCH(420)는 또한 다양한 주변 장치를 시스템(400)에 연결하는 데 사용될 수 있는 입출력(I/O) 컨트롤러 허브(ICH)(450)에 연결된다. 도 4의 실시예에서 예시로 도시된 것은 외부 그래픽 장치(400)이며, 이것은 다른 주변 장치(470)와 함께 ICH(450)에 연결된 별개의 그래픽 장치일 수 있다.

[0042] 대신에, 추가적인 혹은 다른 처리 요소가 시스템(400)에 마련될 수 있다. 예를 들어, 추가 처리 요소(415)는 프로세서(410)와 동일한 추가 프로세서, 프로세서(410)와 이기종인 혹은 비대칭적인 추가 프로세서, 가속기(예를 들어, 그래픽 가속기 혹은 디지털 신호 처리부(DSP)), FPGA(field programmable gate array), 혹은 임의의 다른 처리 요소를 포함할 수 있다. 구조적, 미세 구조적, 열적, 전력 소모 특성 등을 포함하는 다양한 장점의 측정면에서 물리적 자원(410, 415) 사이에는 여러 가지 차이점이 있을 수 있다. 이들 차이는 처리 요소(410,415) 중에서 비대칭과 이종과 같이 명백하게 드러날 수 있다. 적어도 하나의 실시예에 대해, 다양한 처리 요소(410, 415)가 동일한 다이 패키지에 배치될 수 있다.

[0043] 도 5를 참조하면, 본 발명의 일 실시예에 따른 제 2 시스템(500)의 블록도가 도시되어 있다. 도 5에 도시된 바와 같이, 멀티프로세서 시스템(500)은 점대점 상호 접속 시스템이고, 점대점 상호 접속(550)을 통해 연결된 제 1 처리 요소(570) 및 제 2 처리 요소(580)를 포함한다. 도 5에 도시된 바와 같이, 처리 요소(570, 580) 각각은 제 1 프로세서 코어 및 제 2 프로세서 코어(즉, 프로세서 코어(574a, 574b) 및 프로세서 코어(584a, 584b))를 포함하는 멀티코어 프로세서일 수 있다.

[0044] 대신에, 하나 이상의 처리 요소(570, 580)는 가속기 또는 FPGA와 같은 프로세서와 다른 요소일 수 있다.

[0045] 단지 2개의 처리 요소(570, 580)만 도시되었지만, 본 발명의 범주는 그렇게 제한되지 않는다는 점이 이해될 것

이다. 다른 실시예에서, 하나 이상의 추가 처리 요소가 주어진 프로세서에 존재할 수 있다.

[0046] 제 1 처리 요소(570)는 MCH(memory controller hub)(572) 및 점대점(P-P) 인터페이스(576, 578)를 더 포함할 수 있다. 마찬가지로, 제 2 처리 요소(580)는 MCH(582) 및 P-P 인터페이스(586, 588)을 포함할 수 있다. 프로세서(570, 580)는 점대점(PtP) 인터페이스 회로(578, 588)를 사용하는 PtP 인터페이스(550)를 통해 데이터를 교환할 수 있다. 도 5에서 도시된 바와 같이, MCH의 572, 582는 프로세서를 개별 메모리, 즉, 개별 프로세서에 국부적으로 부착되는 주 메모리의 일부 일 수 있는 메모리(542) 및 메모리(544)에 연결한다.

[0047] 프로세서(570, 580)는 점대점 인터페이스 회로(576, 594, 586, 598)를 사용하는 개별 PtP 인터페이스(552, 554)를 통해 각각 칩셋(590)과 데이터를 교환할 수 있다. 칩셋(590)은 고성능 그래픽 인터페이스(539)를 통해서 고성능 그래픽 회로(538)와 데이터를 교환할 수도 있다. 본 발명의 실시예는 임의의 수의 처리 코어를 가지는 임의의 프로세서 내 또는 도 5의 각 PtP 버스 에이전트 내에 배치될 수 있다. 일 실시예에 있어서, 임의의 프로세서 코어는 국부 캐시 메모리(도시 생략함)를 포함하거나 또는 연관될 수 있다. 또한, 공유 캐시(도시 생략함)는 두 프로세서 중 어느 하나의 프로세서 외측에 포함되거나, 여전히 p2p 상호 접속을 통해 프로세서와 연결될 수 있고, 그 결과 프로세서가 저전력 모드에 놓이는 경우 어느 하나의 프로세서 또는 두 프로세서 모두의 국부 캐시 정보가 공유 캐시에 저장될 수 있다.

[0048] 제 1 처리 요소(570) 및 제 2 처리 요소(580)는 각각 P-P 상호 접속(576, 586, 584)을 통해 칩셋(590)에 연결될 수 있다. 도 5에서 도시된 바와 같이, 칩셋(590)은 인터페이스(594, 598)를 포함한다. 또한, 칩셋(590)은 칩셋(590)을 고성능 그래픽 엔진(548)과 연결하기 위한 인터페이스(592)를 포함한다. 일 실시예에서, 버스(549)는 그래픽 엔진(548)을 칩셋(590)과 연결하는 데 사용될 수 있다. 대신에, 점대점 상호 접속(549)이 이들 구성 요소를 연결할 수 있다.

[0049] 차례로, 칩셋(590)은 인터페이스(596)를 통해 제 1 버스(516)과 연결될 수 있다. 일 실시예에서, 제 1 버스(516)는 PCI(peripheral component interconnect) 버스, 또는 PCI 고속 버스나 기타 제 3 세대 상호 접속 버스일 수 있지만, 본 발명의 범주는 그렇게 제한되지 않는다.

[0050] 도 5에 도시된 바와 같이, 제 1 버스(516)를 제 2 버스(520)에 연결하는 버스 브릿지(518)와 함께 다양한 I/O 장치(514)가 제 1 버스(516)에 연결될 수 있다. 일 실시예에 있어서, 제 2 버스(520)는 LPC(low pin count) 버스일 수 있다. 일 실시예에 있어서, 예를 들어 키보드/마우스(522), 통신 장치(526) 및 코드(530)를 포함할 수 있는 디스크 드라이브나 다른 대용량 저장 장치와 같은 데이터 저장부(528)를 포함하는 다양한 장치가 제 2 버스(520)에 연결될 수 있다. 또한, 오디오 I/O(524)는 제 2 버스(520)에 연결될 수 있다. 다른 구조도 가능하다는 점에 유의하라. 예를 들어, 도 5의 점대점 구조 대신에, 시스템이 멀티-드롭 버스나 다른 그러한 구조를 구현할 수 있다.

[0051] 도 6을 참조하면, 본 발명의 일 실시예에 따른 제 3 시스템(600)의 블록도가 도시되어 있다. 도 5와 도 6에서 유사한 요소에는 유사한 참조 번호가 부여되고, 도 6의 다른 부분을 모호하게 하는 것을 방지하기 위해 도 5의 특정한 부분은 도 6에서 생략되었다.

[0052] 도 6은 처리 요소(570, 580)가 각기 집적 메모리 및 I/O 제어 논리("CL")(572, 582)를 포함할 수 있다는 점이 도시되어 있다. 적어도 일 실시예에 대해, CL(572, 582)은 도 4 및 도 5와 관련하여 상술한 바와 같은 MCH(memory controller hub logic)를 포함할 수 있다. 또한, CL(572, 582)은 I/O 제어 논리를 포함할 수도 있다. 도 6은 메모리(542, 544)가 CL(572, 582)에 연결되는 것 뿐만 아니라 I/O 장치(614)도 제어 논리(572, 582)에 연결된다는 점이 도시되어 있다. 레거시 I/O 장치(615)는 칩셋(590)에 연결된다.

[0053] 본 명세서에서 개시된 메카니즘의 실시예는 하드웨어, 소프트웨어, 펌웨어 혹은 그러한 구현 방안의 조합으로 구현될 수 있다. 본 발명의 실시예는 적어도 하나의 프로세서, 데이터 저장 시스템(휘발성 및 비휘발성 메모리 및/또는 저장 요소를 포함함), 적어도 하나의 입력 장치 및 적어도 하나의 출력 장치를 포함하는 프로그램가능한 시스템 상에서 실행되는 컴퓨터 프로그램 또는 컴퓨터 코드로서 구현될 수 있다.

[0054] 도 5에서 도시된 코드(530)와 같은 프로그램 코드는 입력 데이터에 적용되어, 본 명세서에서 설명된 기능을 수행하고 출력 정보를 생성할 수 있다. 출력 정보는 하나 이상의 출력 장치에 주지된 방식으로 적용될 수 있다. 본 출원의 목적을 위해, 처리 시스템은 예를 들어 DSP(digital signal processor), 마이크로컨트롤러, ASIC(application specific integrated circuit), 혹은 마이크로프로세서와 같은 프로세서를 구비하는 임의의 시스템을 포함한다.

[0055] 프로그램 코드는 고수준 절차형 언어 또는 객체 지향 프로그래밍 언어로 구현되어 처리 시스템과 통신할 수 있

다. 만약 필요하다면, 프로그램 코드는 또한 어셈블리 또는 기계어로 구현될 수 있다. 사실, 본 명세서에서 설명된 메카니즘은 어느 특정한 프로그래밍 언어로 범주가 제한되지 않는다. 어쨌든 그 언어는 컴파일된 언어 또는 인터프리티드 언어일 수 있다.

[0056] 적어도 일 실시예의 하나 이상의 형태는 기계 판독 가능한 매체 상에 저장된 리프리젠티브 데이터(representative data)에 의해 구현되어, 기계에 의해 판독될 때 본 명세서에서 설명된 기법을 수행하는 논리를 기계가 조작하게 할 수 있다. "IP cores"라고 알려진 그러한 리프리젠테이션은 유형의 기계 판독가능한 매체 상에 저장되고, 다양한 사용자나 제조 설비에 제공되어 실질적으로 논리나 프로세서를 만드는 제조 기계에 탑재될 수 있다.

[0057] 하드디스크와, 플로피디스크, 광디스크, CD-ROM(compact disk read-only memory), CD-RW(compact disk rewritable) 및 광자기 디스크를 포함하는 임의의 다른 유형의 디스크, ROM(read-only memory)과, DRAM(dynamic random access memory), SRAM(static random access memory)과 같은 RAM(random access memory), EPROM(erasable programmable read-only memory), 플래시 메모리, EEPOM(electrically erasable programmable read-only memory)과 같은 반도체 장치, 자기 또는 광학 카드, 또는 임의의 다른 유형의 전기적 명령을 저장하는 데 적합한 매체를 포함하는, 그러한 기계 판독가능 저장 매체는, 제한없이, 기계 또는 장치에 의해 제조되거나 형성된 비일시적이며 유형의 입자 배열을 포함할 수 있다.

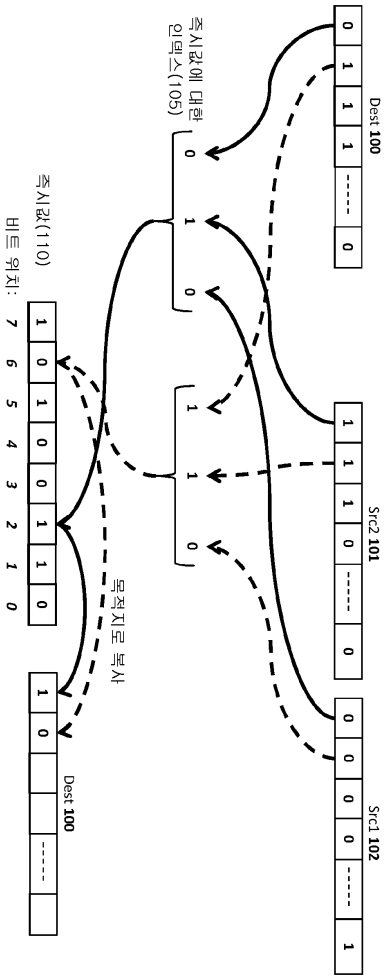
[0058] 따라서, 본 발명의 실시예는, 본 발명의 연산 실시예를 수행하는 명령이나, 본 명세서에서 설명된 구조, 회로, 장치, 프로세서 및/또는 시스템 특징을 정의하는 HDL과 같은 설계 데이터를 포함하는, 비일시적이고 유형(有形)의 기계 판독 가능 매체를 포함할 수도 있다. 그러한 실시예는 프로그램 제품이라고 불릴 수도 있다.

[0059] 본 명세서에 개시된 명령의 특정 연산은 하드웨어 성분에 의해 수행되고, 명령과 함께 프로그래밍된 회로 또는 다른 하드웨어가 연산을 수행하게 하거나 적어도 일부 그렇게 하는 기계 실행가능한 명령으로 구현될 수 있다. 회로는 예를 들면 범용 프로세서나 전용 프로세서 또는 논리 회로를 포함할 수 있다. 연산은 또한 옵션으로 하드웨어와 소프트웨어의 조합에 의해 수행될 수 있다. 실행 논리 및/또는 프로세서는, 기계 명령이나, 기계 명령으로부터 파생된 하나 이상의 제어 신호에 응답하는 지정 회로 또는 특정 회로 또는 기타 논리를 포함하여, 명령 지정 결과 오퍼랜드를 저장할 수 있다. 예를 들어, 본 명세서에 개시된 명령의 실시예는 도 4, 5, 6 중 하나 이상의 시스템에서 실행될 수 있고, 명령의 실시예는 시스템에서 실행될 프로그램 코드에 저장될 수 있다. 또한, 이들 도면의 처리 요소는 본 명세서에서 상세하게 설명한 상세 파이프라인 및/또는 구조(예를 들어, 유효한 구조 또는 비효율적인 구조) 중 하나를 활용할 수 있다. 예를 들어, 유효한 구조의 디코딩부는 명령을 디코딩하여, 디코딩된 명령을 벡터부 혹은 스칼라부 등에 전달할 것이다.

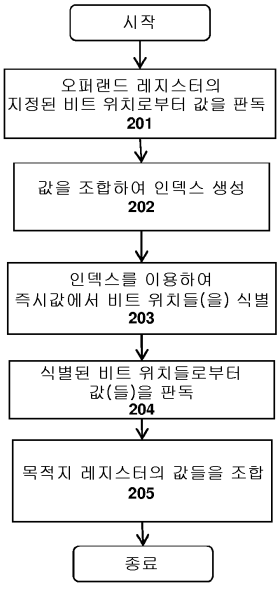
[0060] 상술한 설명 전반에 걸쳐, 설명을 위해, 본 발명을 철저히 이해하도록 여러 구체적인 세부사항을 기술했다. 그러나, 이들 특정한 상세 내용의 일부가 없어도 본 발명이 실시될 수 있다는 점이 당업자에게 명백할 것이다. 본 발명의 사상 및 범주는 후술하는 청구범위 면에서 판정되어야 한다.

도면

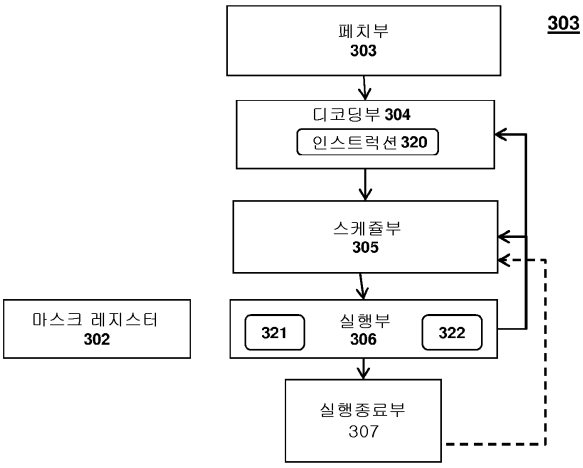
도면1


즉시값에 대한 인덱스(105)
즉시값(110)
비트 위치: 7 6 5 4 3 2 1 0
목적지로 복사
Dest 100
Src2 101
Src1 102

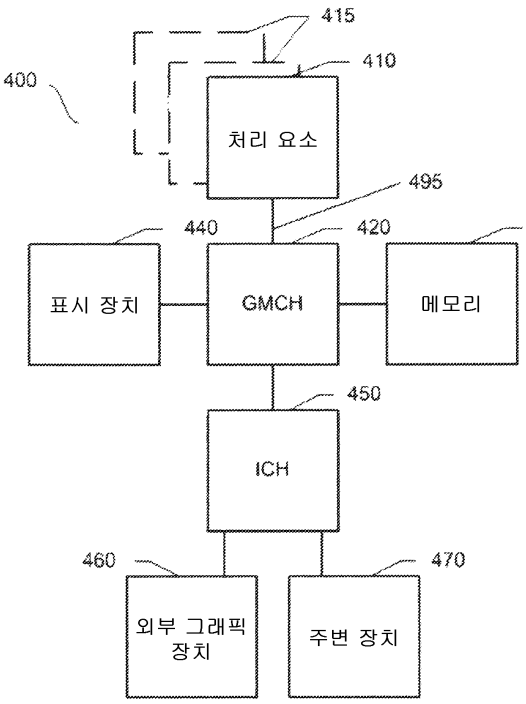
도면2


시작
오퍼랜드 레지스터의 지정된 비트 위치로부터 값을 판독 201
값을 조합하여 인덱스 생성 202
인덱스를 이용하여 즉시값에서 비트 위치들(을) 식별 203
식별된 비트 위치들로부터 값(들)을 판독 204
목적지 레지스터의 값들을 조합 205
종료

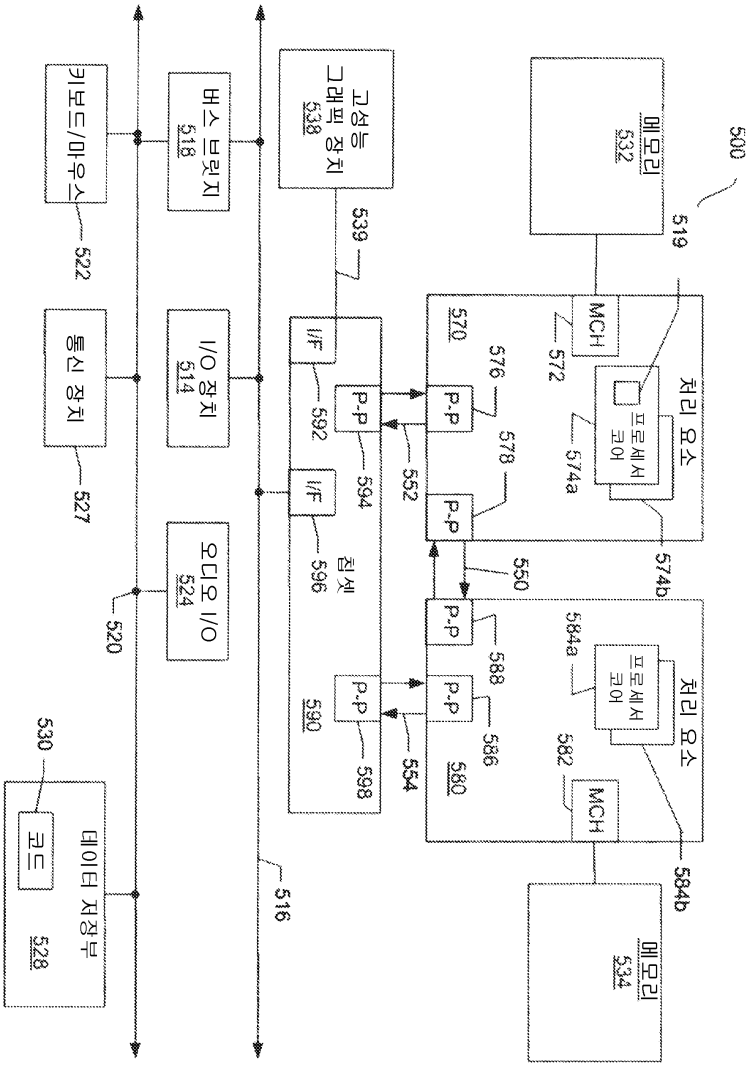
도면3


303
페치부
303
디코딩부 304
인스트럭션 320
스케쥴부
305
마스크 레지스터
302
321
실행부
306
322
실행종료부
307

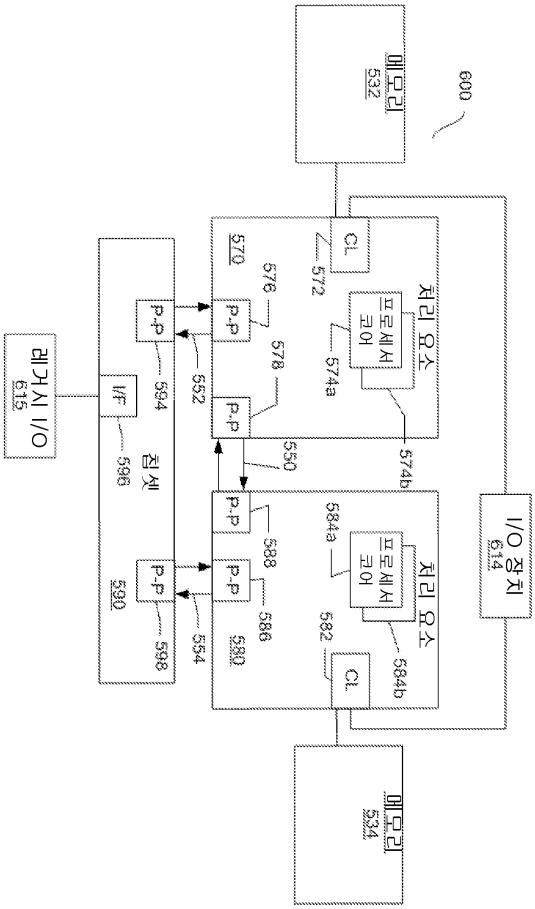
도면4


415
410
400
처리 요소
495
440
420
표시 장치
GMCH
메모리
450
ICH
460
470
외부 그래픽 장치
주변 장치

도면5


500
519
메모리
532
MCH
572
처리 요소
프로세서
코어
574a
574b
570
576
578
P-P
552
550
584a
584b
588
586
582
580
554
590
592
594
596
598
I/F
칩셋
고성능
그래픽 장치
539
538
516
520
버스 브릿지
518
I/O 장치
514
오디오 I/O
524
키보드/마우스
522
통신 장치
527
데이터 저장부
528
코드
530
메모리
534

도면6


600
메모리
532
CL
572
처리 요소
570
프로세서
코어
574a
574b
576
578
P-P
550
552
554
580
582
584a
584b
586
588
590
592
594
596
598
칩셋
I/F
레거시 I/O
615
I/O 장치
614
메모리
534