(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
G06F 5/00 (2006.01)

(45) 공고일자 2006년04월07일
(11) 등록번호 10-0567937
(24) 등록일자 2006년03월30일

(21) 출원번호 10-1999-7007883
(22) 출원일자 1999년08월27일
번역문 제출일자 1999년08월27일
(86) 국제출원번호 PCT/US1998/003954
국제출원일자 1998년02월27일

(65) 공개번호 10-2000-0075810
(43) 공개일자 2000년12월26일
(87) 국제공개번호 WO 1998/38791
국제공개일자 1998년09월03일

(81) 지정국 국내특허 : 알바니아, 아르메니아, 오스트리아, 오스트레일리아, 아제르바이잔, 보스니아 헤르체고비나, 바르바도스, 불가리아, 브라질, 벨라루스, 캐나다, 스위스, 중국, 쿠바, 체코, 독일, 덴마크, 에스토니아, 스페인, 핀란드, 영국, 그루지야, 헝가리, 이스라엘, 아이슬랜드, 일본, 케냐, 키르키즈스탐, 북한, 대한민국, 카자흐스탄, 세인트루시아, 스리랑카, 리베이라, 레소토, 리투아니아, 룩셈부르크, 라트비아, 몰도바, 마다가스카르, 마케도니아공화국, 몽고, 말라위, 멕시코, 노르웨이, 뉴질랜드, 슬로베니아, 슬로바키아, 타지키스탄, 투르크맨, 터어키, 트리니아드토바고, 우크라이나, 우간다, 우즈베키스탄, 베트남, 폴란드, 포르투칼, 루마니아, 러시아, 수단, 스웨덴, 싱가포르, 가나, 세르비아 앤 몬테네그로, 짐바브웨,

AP ARIPO특허 : 케냐, 레소토, 말라위, 수단, 스와질랜드, 우간다, 가나, 감비아, 짐바브웨,

EA 유라시아특허 : 아르메니아, 아제르바이잔, 벨라루스, 키르키즈스탐, 카자흐스탄, 몰도바, 러시아, 타지키스탄, 투르크맨,

EP 유럽특허 : 오스트리아, 벨기에, 스위스, 독일, 덴마크, 스페인, 프랑스, 영국, 그리스, 아일랜드, 이탈리아, 룩셈부르크, 모나코, 네덜란드, 포르투칼, 스웨덴, 핀란드,

OA OAPI특허 : 부르키나파소, 베닌, 중앙아프리카, 콩고, 코트디브와르, 카메룬, 가봉, 기니, 말리, 모리타니, 니제르, 세네갈, 차드, 토고,

(30) 우선권주장 808,735 1997년02월28일 미국(US)

(73) 특허권자
브이엠 랩스, 인코포레이티드
미국, 캘리포니아 94040, 마운틴 뷰, 샌 안토니오 로드 520

(72) 발명자
밀러리차드지.
미국,캘리포니아94022,로스알토스힐스,라로마드라이브25055

카딜로루이스에이.
미국,캘리포니아94306,팔로알토,루즈벨트서클59

마티슨존지.
미국,캘리포니아95030,로스가토스,스카이뷰테라스23583

스미스에릭알.
미국,캘리포니아94086,서니베일,엔.마틸다애비뉴#시204450

(74) 대리인 이병호
정상구
신현문
이범래

심사관 : 성경아

(54) 프로세서용 명령 압축 및 압축 해제 시스템 및 방법

요약

복수의 명령들을 포함하는 명령 패킷을 발생시키는 시스템, 보다 짧은 압축된 명령이 보다 빈번히 사용된 명령에 대응하는 소정 길이의 압축된 명령을 갖는 압축된 명령을 명령 패킷 내의 명령에 할당하는 시스템, 및 처리 유닛들 중의 대응하는 것들에 대한 압축된 명령들을 포함하는 명령 패킷을 발생시키는 시스템을 구비한 압축 시스템을 갖는, 복수의 처리 유닛들(26,30,32,34,36)을 갖는 프로세서(170)에서 가변 길이 명령 패킷들에 포함된 가변 길이 명령들(110, 116)을 압축 및 압축 해제하는 시스템(170) 및 방법이 제공된다. 압축 해제 시스템(178)은 복수의 저장 위치들에 복수의 명령 패킷들을 저장하는 시스템, 저장 시스템 내의 선택된 가변 길이 명령 패킷을 지시하는 어드레스를 발생시키는 시스템, 및 각각의 처리 유닛들에 대한 가변 길이 명령을 발생시키기 위해 선택된 명령 패킷의 압축된 명령들을 압축 해제하는 압축 해제 시스템을 포함한다. 압축 해제 시스템(178)은 또한 압축 해제 수단으로부터 각각의 처리 유닛들로 가변 길이 명령들을 라우팅하는 시스템을 가질 수 있다.

대표도

도 1

색인어

명령 압축 시스템, 명령 압축 방법, 명령 압축 해제 시스템, 명령 압축 해제 방법, 명령어 패킷, 토큰 필드

명세서

기술분야

본 발명은 일반적으로 프로세서 내의 명령들에 대한 저장 공간을 감소시키기 위한 시스템 및 방법에 관한 것이며, 특히 프로세서 내의 메모리에 저장되는 매우 긴 명령어들을 압축 및 압축 해제하기 위한 시스템 및 방법에 관한 것이다.

배경기술

실시간 디지털 신호 처리, 실시간 비디오 처리 및 실시간 이미지 압축 해제와 같은 특정 태스크(task)들은 화소 디스플레이 데이터와 같은 유효량의 데이터를 실시간으로 신속히 처리하는 고속 처리 시스템들을 요한다. 이들 고속 처리 시스템들은 복잡한 프로세서들, 예를 들면 매 클록 주기당 5개의 개별적인 기능 유닛들을 위해 5개의 개별적인 명령들을 처리하는 매우 긴 명령어(VLIW) 프로세서들을 사용할 수 있다. 이들 프로세서들은 150비트에 이르는 길이의 매우 긴 명령어들을 사용하고, 이러한 매우 긴 명령어들을 저장하기 위해 큰 용량의 메모리를 필요로 한다. 이러한 매우 긴 명령어들을 저장하기 위한 매우 큰 용량의 메모리는 고가이다. 전형적인 VLIW 프로세서들에 대해, 이들 매우 긴 명령어들은 150비트에 이르는 길이일 수 있다. 이들 매우 긴 명령어들은 예를 들면 5개의 별개의 데이터의 부분들을 동시에 처리하기 위해 5개의 처리 유닛들을 허용하지만, 매우 긴 명령어들을 저장하기는 어렵다. 또한, 항상 매 클록 주기동안 모든 다중 기능 유닛들을 완전히 이용할 수는 없다. 그러나, 전형적인 VLIW 프로세서는 매 클록 주기당 각각의 기능 유닛에 할당된 고정된 수의 비트들을 갖고 있으므로, 유휴 기능 유닛들이 존재할 때, 매우 긴 명령어 내의 일부 비트들은 낭비된다. 매우 긴 명령어 내의 낭비된 비트들로 인해, 메모리 저장 공간 역시 낭비된다. 낭비된 메모리 공간으로 인해, 명령어들이 너무 크기 때문에 간단한 프로

그램조차도 명령 메모리를 가득 채울 수 있다. 또한, 비디오 압축 해제 프로그램들 또는 이미지 발생 프로그램들과 같은 보다 복잡한 프로그램들은 명령 메모리에 전체적으로 저장될 수 없고, 메모리로 연속적으로 다시 로드되어야 한다. 프로그램의 메모리로의 연속적인 재로딩은 프로세서의 속도를 허용될 수 없는 레벨들까지 느리게 만든다.

따라서, 매우 긴 명령어를 저장하는 데 필요한 메모리의 용량을 감소시키는 시스템 및 방법이 필요하다. 메모리 유닛 및 연산 유닛과 같은, 단지 2개의 처리 유닛들만을 갖는 종래의 처리 시스템 중 하나는, 메모리에 저장되는 처리 유닛들 각각에 대한 별개의 명령들을 갖는다. 다음으로, 프로세서가 다른 명령을 받아들일 준비가 되어 있을 때, 2개의 인접한 명령들이 특정 기준들에 기초하여 프로세서에 도입되기 전에 함께 조합될 수 있는지 여부가 결정된다. 인접한 명령들을 조합하기 위해, 명령은 메모리 명령(즉, 로드 또는 저장) 및 연산 논리 유닛 명령이어야 한다. 조합된 명령은 프로세서에 의해 보다 신속히 처리될 수 있다. 이러한 시스템은 프로세서의 처리 속도를 증가시키지만, 고가의 처리 비용을 수반하고, 전체 길이의 명령들이 메모리에 저장되기 때문에 명령에 요구되는 메모리의 양을 감소시키지 못한다.

예를 들면, 40비트 길이의 짧은 명령들 및 예를 들면, 80비트 길이의 긴 명령들을 모두 갖는 종래의 VLIW 프로세서 시스템들이 또한 존재한다. 짧은 명령들은 루프들을 개시하기 위해 사용되는 한편, 긴 명령어들은 실제적인 내부 루프들을 위해 사용된다. 짧은 명령들과 긴 명령들의 이러한 선택은 또한 증가된 처리 속도를 제공하고, 특정 명령들의 크기를 감소시킬 수 있지만, 명령 메모리 공간을 감소시키는 문제점을 적절히 다루지 못한다. 다른 VLIW 프로세서 시스템은 명령 캐시를 사용하고, 여기서 명령 캐시의 부분들은 시스템에서 각각의 처리 유닛들에 전용된다. 다시 한번, 이 시스템은 명령들의 처리 속도를 증가시킨다. 이러한 시스템 역시 명령 메모리 공간을 감소시키지만, 매우 긴 명령어에서 낭비된 비트가 여전히 존재한다. 또 다른 VLIW 시스템은 병렬 처리 및 처리 속도를 증가시키기 위해 여러 가지 유형들의 명령들을 함께 그룹화하지만, 관련된 명령 메모리 공간을 다루지 못한다. 또 다른 VLIW 시스템은 고정된 길이의 명령 패킷 내에 포함된 가변 길이 명령들을 갖는다. 이들 시스템 중 어느 것도 매우 긴 명령어들을 저장하는 데 요구되는 메모리 크기를 효율적으로 감소시키는 방법을 제공하지 못한다. 따라서, 이들 종래의 시스템은 고가이고, 큰 크기의 매우 긴 명령어들로 인해 시스템의 명령 메모리 내에 복잡한 프로그램들을 전체적으로 저장할 수 없다.

따라서, 매우 긴 명령어들을 저장하는 데 필요한 메모리의 용량을 감소시키고, 공지된 장치들의 이들 및 기타 문제점들을 피하는 시스템 및 방법에 대한 필요성이 대두되고 있고, 따라서 그것이 본 발명이 지향하는 목적이다.

발명의 상세한 설명

본 발명은 VLIW 프로세서에서 매우 긴 명령어들을 저장하는데 필요한 메모리의 양을 감소시키는 시스템 및 방법을 제공함으로써 상기 및 기타 문제점들을 다룬다. 본 발명은 메모리에 저장되어야 하는 매우 긴 명령어들의 크기를 감소시키고, 매우 긴 이들 명령어들을 이들의 크기를 감소시키기 위해 압축함으로써 이를 달성한다. 본 발명은 명령 패킷으로 알려진, 처리 유닛들에 의한 실행 직전에 압축 해제될 수 있는 포맷에서 다중 처리 유닛들의 각각에 대한 많은 압축된 명령들을 발생시키고, 저장할 수 있다. 명령들은 많은 방식들로 압축될 수 있다(즉, 각각의 명령의 크기가 감소됨). 예를 들면, 명령 패킷 내에는 전형적으로 약간의 미사용 비트들이 존재한다. 예를 들면, 처리 유닛이 유휴상태이고 비 오퍼레이션 명령이나 디폴트 명령을 실행할 수 있는 경우에도 전체 32-비트의 긴 처리 유닛 명령이 사용되기 때문에 미사용 비트들이 존재한다. 비 오퍼레이션 명령은 명령이 아닐 수 있고, 디폴트 명령은 예를 들면, 2개의 입력들을 함께 곱하는 곱셈 기능 유닛을 가질 수 있다. 그러나, 디폴트 또는 비 오퍼레이션 명령은 전체 32 비트를 요하지 않는다. 전형적인 VLIW 프로세서 시스템에서, 처리 유닛들의 단지 약 1/2이 임의의 주어진 시간에 실제로 처리되는 유효 명령들이다. 처리 유닛들의 나머지 1/2는 처리 디폴트 명령이다. 위에서 설명된 바와 같은, 이들 디폴트 명령들의 각각은 단축될 수 있다. 따라서, 일부 처리 유닛들은 실행되는 디폴트 명령들이기 때문에, 매우 긴 명령어가 본 발명에 따라 압축될 수 있다. 또한, 처리 유닛들에 의해 실행되는 대부분의 명령들은 매우 긴 명령어 내에서 사용할 수 있는 모든 비트들을 사용할 필요가 없으므로 이들 비트들은 압축될 수 있다. 또한, 디폴트 명령들을 포함하는 명령들은 짧은 코드를 각각의 보다 긴 명령에 할당함으로써 또한 압축될 수 있고, 이어서 이들 코드들은 실행 시점에 확장된다.

복수의 명령들을 포함하는 명령 패킷이 발생되고, 소정 길이의 압축된 명령이 명령 패킷 내의 명령에 할당되는, 본 발명에 따른 명령 압축 및 압축 해제 시스템 및 방법이 제공된다. 보다 짧게 압축된 명령들은 보다 빈번하게 발생하는 명령들에 대해 사용된다. 압축된 명령들을 포함하는 명령 패킷이 발생되고, 압축될 때 상기 처리 유닛들의 각각의 오퍼레이션을 제어할 것이다. 압축 해제는 복수의 상기 명령 패킷들을 복수의 저장 위치들에 저장하고, 저장 시스템 내의 선택된 가변 길이 명령 패킷을 지시하는 어드레스를 발생시키고, 상기 처리 유닛들의 각각에 대한 가변 길이 명령을 발생시키기 위해 상기 선택된 명령 패킷 내의 상기 압축된 명령들을 압축 해제함으로써 일어난다. 본 발명은 또한 상기 압축 해제된 가변 길이 명령들을 상기 처리 유닛들의 각각으로 라우팅할 수도 있다.

도면의 간단한 설명

도 1은 다중 처리 유닛들을 포함하는 파이프라인 프로세서의 블록도.

도 2는 도 1에 나타낸 처리 유닛들을 제어하기 위해 사용될 수 있는 매우 긴 명령어의 도면.

도 3a 및 3b는 본 발명에 따른 압축된 명령의 2가지 상이한 포맷들을 예시하는 도면들.

도 4는 압축되지 않은 매우 긴 명령어가 어떻게 압축될 수 있는지를 도시하는 도면.

도 5는 본 발명에 따른 매우 긴 명령어(VLIW) 압축 해제 시스템의 블록도.

도 6은 도 5에 나타낸 VLIW 압축 해제 시스템의 보다 상세한 블록도.

도 7은 복수의 명령 패킷들을 포함하는 명령 메모리를 도시한 도면.

도 8은 본 발명에 따른 이러한 메모리 블록들의 어드레싱을 나타내는 명령 메모리 블록들의 도면.

도 9는 본 발명에 따라 압축된 VLIW 명령 패킷을 압축되지 않은 매우 긴 명령어로 압축 해제하는 것을 도시한 도면.

실시예

본 발명은 프로세서 내의 명령 메모리의 크기를 감소시키는 시스템, 특히 VLIW 프로세서에서 매우 긴 명령어들을 압축 및 압축 해제하는 시스템에 특히 적용할 수 있다. 이러한 관점에 있어서 본 발명이 설명될 것이다. 그러나, 본 발명의 시스템 및 방법은 더 많은 용도를 가짐을 인식해야 할 것이다.

도 1은 본 발명에 따른 명령 압축 및 압축 해제 시스템을 포함할 수 있는 매우 긴 명령어(VLIW) 프로세서(20)의 도면이다. VLIW 프로세서(20)는 오퍼레이션 코드 소스 버스(22) 및 실행 제어 유닛(ECU)(26), 레지스터들의 세트(28), 승산기 유닛(MUL)(30), 연산 논리 유닛(ALU)(32), 레지스터 제어 유닛(RCU)(34), 및 메모리 유닛(MEM)(36)과 같은 복수의 시스템 유닛들을 전기적으로 상호접속시키는 결과 버스(24)를 포함할 수 있다. 본 발명은 도시된 구조로만 제한되지 않고, 예를 들면 1개 이상의 MUL을 포함할 수 있지만, 예를 들면 어떠한 ALU도 포함하지 않을 수 있다. ECU, MUL, ALU, RCU 및 MEM 유닛들은 처리 유닛들로 알려져 있다. VLIW 프로세서에서, 처리 유닛들은 병렬로 함께 접속될 수 있으므로 각각의 처리 유닛은 매우 긴 명령어에 포함된 명령을 동시에 처리할 수 있다. ECU(26)는 VLIW 프로세서 내에서 명령들의 검색 및 실행을 제어한다. 레지스터(28)는 프로세서 내의 다양한 처리 유닛들에 의해 이용되는 데이터를 저장하고, MUL 유닛(30)은 2개의 레지스터들로부터 2조각들의 데이터를 승산하며, 그 곱 값을 다른 레지스터에 저장하고, ALU(32)는 다양한 연산 기능들 및 데이터의 조각들에 대한 논리 오퍼레이션들을 수행한다. RCU(34)는 특별한 특정 레지스터들을 제어하고, MEM 유닛(36)은 프로세서 내의 다양한 저장 시스템들에 대한 다른 처리 유닛들의 액세스를 제어한다. 일반적으로, 위에서 논의된 각각의 처리 유닛들이 별개의 명령들을 동시에 실행할 수 있기 때문에 도시된 VLIW 프로세서는 매 클록 주기당 5개에 이르는 명령들을 실행할 수 있다.

ECU(26)는 또한 명령 메모리 버스(42)에 의해 복수의 명령 메모리들(38, 40)에 접속될 수도 있다. 명령 메모리들은 랜덤 액세스 메모리(RAM)(38) 및 판독 전용 메모리(ROM)(40)일 수도 있다. 이들 메모리들은 또한 플래쉬 메모리, 또는 전기적으로 소거/프로그램 가능한 판독 전용 메모리(EEPROM)와 같은 임의의 유형의 저장 장치일 수도 있다. 이들 명령 메모리들은 매우 긴 명령어들로서, ECU 유닛에 의해 다양한 처리 유닛들로 라우팅되는 명령들을 저장한다. 명령 ROM(40)은 빈번히 사용된 명령들을 사용할 수 있고, 이들 명령들은 RAM에 저장될 필요가 전혀 없다.

MEM 유닛(36)은 데이터 메모리 버스(48)에 의해 데이터 메모리들(44, 46)에 접속된다. 데이터 메모리들(44, 46)은 RAM 및 ROM일 수 있지만, 소거/프로그램 가능한 판독 전용 메모리(EPROM), 플래쉬 메모리, 또는 EEPROM과 같은 임의의 기타 유형의 메모리일 수도 있다. 이들 데이터 메모리들은 VLIW 프로세서에 의해 작동되는 데이터를 저장한다. 데이터 ROM(46)은 VLIW 프로세서에 의해 빈번히 사용되는 데이터 또는 데이터 구조들을 저장할 수 있다. 모든 이들 처리 유닛들을 동시에 제어하기 위해, 도 2에 도시된 바와 같은, 매우 긴 명령어가 사용될 수 있다.

도 2는 도 1에 나타낸 모든 처리 유닛들을 제어하기 위해 사용될 수 있는 매우 긴 명령어(60)의 예의 도면이다. 본 발명은 매우 긴 명령어 내의 임의의 특정 순서의 명령들 또는 매우 긴 명령어 내의 임의의 특정 수의 명령들로 제한되지 않는다. 예를 들면, VLIW 프로세서는 2개의 MUL 유닛들을 가질 수 있으므로 각각의 매우 긴 명령어는 2개의 MUL 명령들을 가질

수 있다. 매우 긴 명령어는 큰 수의 비트들, 예를 들면 160개의 비트들로 형성될 수 있고, 처리 유닛들 중 개개의 것들을 개별적으로 제어하는 복수의 명령어들을 포함하는 부분들을 포함한다. 예를 들면, ECU_CTRL은 ECU 유닛을 제어하는 32-비트 명령어(62)일 수 있다. MUL_CTRL 명령어(64)는 MUL 유닛을 제어할 수 있고, 32비트 길이일 수도 있다. ALU_CTRL 명령어(66)는 ALU 유닛을 제어할 수 있고, 32비트 길이일 수도 있다. RCU_CTRL 명령어(68)는 RCU 유닛을 제어할 수 있고, 16비트 길이일 수도 있다. 마지막으로, MEM_CTRL 명령어(70)는 MEM 유닛을 제어할 수 있고, 64비트에 이르는 길이일 수 있다. 각각의 처리 유닛에 대한 각각의 이들 명령어들의 포맷들은 당업계에 잘 공지되어 있으며, 프로세서 유닛들에 대한 제어 신호들과 명령어 간에 처리가 거의 없거나 또는 전혀 없는 RISC 스타일의 프로세서 구조에 따른다. 도시된 바와 같이, 모든 이들 명령어들은 매우 긴 명령어를 형성하기 위해 함께 조합된다. 이러한 매우 긴 명령어는 160비트에 이르는 길이일 수 있다. 보다 많은 처리 유닛들을 갖는 VLIW 프로세서는 보다 긴 매우 긴 명령어를 가질 수 있기 때문에, 본 발명은 임의의 특정 길이의 매우 긴 명령어로 제한되지 않는다. 도시된 바와 같이, 이들 다양한 명령어(62-70)는 VLIW(60)를 형성하기 위해 조합될 수 있다. 인식할 수 있듯이, 명령 메모리들(38 및 40)이 복잡한 프로그램들에 대한 경우일 수 있듯이, 많은 수의 이러한 VLIW들을 저장해야 하는 경우, 매우 큰 크기의 메모리들을 필요로 한다. 본 발명은 이후에 설명될 바와 같은 매우 긴 명령어를 압축 및 압축 해제하는 시스템 및 방법을 제공함으로써 이를 피하게 된다.

도 3a는 처리 유닛들 중 개별적인 것에 대한 본 발명에 따른 압축된 명령어(110)의 포맷의 도면이다. 도시된 압축된 명령어는 16비트 형태이다. 압축된 명령어는 또한 유사한 포맷을 갖는 32비트 및 48비트 형태를 가질 수도 있고, 도 3b를 참조하여 아래에서 설명될 것과 같이 0비트 포맷(디폴트 명령)을 가질 수도 있다. 0비트 압축된 명령은 이하에서 설명될 것이다. 가장 빈번히 사용된 명령들의 가장 많은 양의 압축을 초래하기 때문에, 바람직하게는, 보다 짧은 길이의 압축된 명령어들은 보다 빈번히 발생하는 미압축된 명령어들로 할당된다. 예를 들면, 16비트 형태들은 대다수의 명령들에 대해 사용될 수 있고, 보다 긴 형태들(32비트 길이, 48비트 길이 또는 64비트 길이)은 모든 다른 명령들을 위해 사용된다. 일부 명령들은 직접 데이터와 같은 일부 데이터가 16비트 압축된 명령에 적합할 수 없기 때문에 보다 긴 형태들을 사용한다. 16비트 압축된 명령어(110)는 스톱 비트(113), 소스 레지스터 필드(114) 및 목적지 레지스터 필드(115)를 포함할 수 있는 토큰 필드(112)를 포함할 수 있다. 토큰, 소스 레지스터 및 목적지 레지스터 필드들에 할당된 비트들의 수는 변화될 수 있고, 도시된 압축된 명령은 단지 일 예이다. 스톱 비트는 이러한 특정 명령이 압축된 명령 패킷 내의 최종 명령인 경우 "1"로 설정된다. 명령이 명령 패킷 내의 최종 명령이 아닌 경우, 스톱 비트는 설정되지 않는다(즉, 이는 "0"). 따라서, 16비트의 압축된 명령에 대해, 스톱 비트는 하나의 압축된 명령 패킷이 종료되고 다음의 압축된 명령 패킷이 시작되는 ECU 유닛 및 프로세서를 지시한다.

5비트 폭일 수 있는 토큰 필드(112)는 오퍼레이션("op") 코드, 제어 워드 및 미압축된 명령의 형태 워드에 대응하는 토큰을 저장한다. 토큰들이 선택됨으로써 각각의 토큰은 명령 세트에서 단지 하나의 미압축된 명령에 대응한다. 따라서, 토큰 필드는 이하에서 설명되는 바와 같은 압축 해제 시스템이 그 명령 뿐만 아니라 처리 유닛에 대한 실제적인 명령에 의해 영향을 받는 모든 처리 유닛을 결정하게 한다. 토큰 필드는 사실상 처리 유닛을 식별하고 실제적인 미압축된 명령을 식별한다. 그러나, 토큰은 임의의 방식으로 미압축된 명령들로 할당될 수 있고, 최상의 압축은 상기한 바와 같이, 가장 짧은 명령어들이 가장 빈번히 사용된 명령들에 할당될 때 발생한다.

5비트일 수 있는 소스 레지스터 필드(114)는 프로세서 내의 어떤 레지스터가 명령에 대한 소스 데이터를 저장하기 위해 사용되는지를 결정할 수 있다. 미압축된 명령으로부터 소스 레지스터 어드레스가 압축되고, 이것이 소스 레지스터 필드로 입력된다. 이 실시예에서, 소스 필드가 5비트이기 때문에, 32에 이르는 레지스터들(2^5=32)이 특정될 수 있다. 마찬가지로, 목적지 레지스터 필드(118)는 5비트를 가지므로, 이것은 또한 32개에 이르는 레지스터들이 특정될 수 있다. 미압축된 명령으로부터 목적지 레지스터 어드레스가 압축되고 이것이 필드로 입력된다. 그러나, 본 발명은 압축된 명령 내의 임의의 특정 크기 필드들로 제한되지 않는다.

op 코드, 소스 레지스터 어드레스 및 목적지 레지스터 어드레스 외에, 이후에 설명될 바와 같이 토큰 필드로 암호화될 수 있고 압축 해제 시스템에 의해 재발생될 수 있는 미압축된 명령 내의 제어 워드 또는 op 코드의 형태와 같은 다른 비트들이 존재할 수 있다. 압축을 추가로 증가시키기 위해, 디폴트 명령들은 매우 긴 명령어로부터 제거될 수 있고, 0 비트 명령어로 압축되는 것으로 생각될 수 있다. 압축 해제 시스템은 명령 패킷 내에 압축된 명령어를 갖지 않는 각각의 처리 유닛에 대한 디폴트 명령들을 자동으로 발생시킨다. 이러한 디폴트 명령들은 No_오퍼레이션(No_Op) 명령들일 수 있지만, 특정 용도에 대해 주문받은 디폴트 명령들일 수도 있다. 예를 들면, 그래픽스 처리 시스템에 대해, 디폴트 명령들은 명령들의 루프가 처리되게 할 수 있다. 디폴트 명령들은 또한 프로세서에 다운로드될 수 있고 따라서 디폴트 명령들이 용이하게 변화되거나 주문될 수 있다. 압축 시스템은 아래에서 설명될 바와 같이 패드 명령을 부가할 수도 있다.

도 3b는 본 발명에 따른 32비트 압축된 명령(116)의 포맷의 예이다. 도시된 바와 같이, 이러한 32비트 압축된 명령은 위에서 설명된 바와 같이 동일한 유형의 데이터를 포함하는 토큰 필드(112), 소스 레지스터 필드(114) 및 목적지 레지스터 필

드(115)를 가질 수 있다. 그러나, 16비트보다 더 긴 임의의 압축된 명령에 대해, 압축된 명령이 패킷 내의 최종 압축된 명령인지를 나타내는 패킷 지시자들의 말단의 위치가 제거된다. 도시된 바와 같이, 보다 긴 압축된 명령 내에는 패킷의 말단(EP)/패킷의 비말단(NEP) 필드(117) 및 제 2 토큰 필드(118)가 존재할 수 있다. EP/NEP 필드는 이 시스템이 특정 압축된 명령이 압축된 명령 패킷의 말단에 있는지 여부를 결정하게 하고, 16비트 길이의 압축된 명령에서 스톱 비트(113)와 동일한 기능을 수행한다.

보다 긴 이들 압축된 명령들에 대해, 토큰 필드(112)는 압축된 명령이 16비트보다 길고 압축된 명령이 적절한 기능 유닛으로 라우팅되어야 하는 시스템을 지시하는 op_코드를 포함한다. 이러한 압축 해제 하드웨어는 16비트 경계들 상의 압축된 명령들을 처리하기 때문에, 하드웨어는 명령이 패킷의 말단에 존재하는지 여부를 나타내는 EP/NEP 필드를 다음으로 검토한다. 64비트 길이의 압축된 명령에 대해, 압축된 명령의 2번째 및 4번째 16비트 부분들의 시작점에 있는 EP/NEP 필드일 수 있다. 제 2 토큰 필드(118)는 실제 오퍼레이션들이 기능 유닛에 의해 수행되도록 지시하는 실제 토큰을 포함할 수 있다. EP/NEP 필드를 사용하는 것은 압축된 명령의 길이와 무관하게 디컴프레서가 패킷 지시자의 말단을 용이하게 위치시킬 수 있게 한다. 이제, 본 발명에 따른 압축의 예가 설명될 것이다.

도 4는 본 발명에 따른 미압축된 매우 긴 명령어(VLIW)(120) 및 대응하는 압축된 명령 패킷(121)의 도면이다. 매우 긴 명령어(120)는 본 발명에 따라 컴파일러 또는 어셈블러에 의해 압축될 수 있다. 이 예에서 도시된 바와 같이, 미압축된 명령어는 레지스터(1)의 내용들이 레지스터(2)의 내용들에 부가되는 ADD 오퍼레이션이 프로세서 내에서 발생하게 할 수 있다. ADD 명령을 보완하기 위해, 메모리 명령으로부터 LOAD 레지스터는 MEM 유닛에 의해 완성될 수 있고, ADD 명령은 ALU 유닛에 의해 완성될 수 있다. 따라서, 도시된 바와 같이, MEM_CTRL 워드(122)는 48비트 길이의 LOAD 명령을 포함하고, ALU_CTRL 워드(125)는 32비트 길이의 ADD 명령을 포함하고, 다른 처리 유닛들에 대한 제어 워드들은 No_오퍼레이션(No_Op) 또는 디폴트 명령들이다. 이들 No_오퍼레이션(No_Op) 또는 디폴트 명령어들(123, 124 및 126)은 ECU에 대해 32비트 길이이고, RCU 유닛에 대해 16비트 길이이며, MUL 유닛에 대해 32비트 길이이다. 따라서, 이러한 미압축된 매우 긴 명령어에 요구되는 전체적인 비트수는 단지 2개의 처리 유닛들이 사용되더라도 160비트이다.

이러한 매우 긴 명령어를 압축하기 위해, 여러 가지 상이한 액션들이 발생한다. 먼저, 디폴트 또는 No_오퍼레이션 명령들은 0비트 길이 명령들로 압축된다. 본질적으로, 디폴트 및 No_오퍼레이션 명령들은 아래 설명될 바와 같이 이들 명령들이 압축 해제 시스템에 의해 재삽입될 수 있기 때문에 매우 긴 명령어로부터 제거된다. 따라서, 본 실시예에서 디폴트 명령들이 없는 매우 긴 명령어는 MEM 명령(122) 및 ALU 명령(126)만을 갖고, 80비트 길이이다. 이제, MEM 및 ALU 명령들은 명령 패킷의 길이를 추가로 감소시키기 위해 압축된다.

48비트 MEM 미압축된 명령(122)은 도시된 바와 같이 16비트의 압축된 MEM 명령(127)으로 압축된다. 미압축된 명령의 스톱 비트(128) 및 제어 비트(130)는 6비트 스톱 비트 및 토큰 필드(132, 134)로 압축된다. 토큰 필드(134)는 MEM 명령이 명령 패킷 내의 최종 명령이 아니기 때문에 본 실시예에서 스톱 비트(132)를 필요로 하지 않는다. 미압축된 명령의 32비트의 직접 번호(136)는 5비트 소스 레지스터 필드(138)에 적합하게 압축되고, 5비트 레지스터 어드레스(140)는 목적지 레지스터 필드(142)에 놓인다. 따라서 압축된 MEM 명령(127)은 16비트 길이다.

또한 32비트 ALU 명령(126)은 스톱 비트(154) 및 토큰 필드(156), 소스 레지스터 필드(160) 및 목적지 레지스터 필드(166)를 갖는 16비트 압축된 명령(144)으로 압축된다. 위와 같이 스톱 비트(146), 제어 비트들(148), op 코드(150) 및 형태 비트들(152)의 전체 12비트들은 압축되고 스톱 비트(154) 및 토큰 필드(156)에 놓인다. 마찬가지로, 소스 레지스터 및 목적지 레지스터 필드(160, 166)가 또한 발생된다. 토큰, 소스 레지스터 및 목적지 레지스터 필드들은 압축된 명령(144)을 형성하기 위해 조합된다. 다음으로, 2개의 압축된 명령들(127, 144)이 32비트 길이의 명령 패킷(121)을 형성하기 위해 함께 조합된다. 보다 미압축된 명령들 및 보다 적은 No_Op 명령들을 갖는 매우 긴 명령어에 대해, 명령 패킷은 보다 길고, 보다 압축된 명령들을 포함할 수 있다. 예를 들면, 이러한 32비트 압축된 명령 패킷(121)은 160비트의 미압축된 매우 긴 명령어 대신에 명령 메모리에 저장될 수 있다. 달성된 압축량은 압축되는 명령들에 좌우된다. 그러나, 가장 큰 용량의 압축을 달성하기 위해, 가장 빈번히 사용되는 명령들은 가장 적게 압축된 명령에 할당되는 것이 바람직하다. 따라서, 위에서 도시된 바와 같이, 대부분의 프로그램들에 공통인 ADD 명령은 32비트로부터 16비트로 압축된다. 본 발명은 임의의 특정 할당 도식으로만 제한되지 않는다.

오퍼레이션에서, 프로그램은 소정량의 메모리 공간을 점유하는 많은 매우 긴 명령어들(VLIW)의 시퀀스를 포함한다. 각각의 개별적인 VLIW는 위에서 설명한 바와 같이 명령 패킷 내의 압축된 명령들의 수에 따라 16비트 길이 내지 128비트 길이일 수 있는 압축된 명령 패킷으로 압축된다. 이어서, 이러한 압축된 명령 패킷의 각각이 순차 메모리 위치들에 놓임으로써, 이러한 압축된 명령 패킷들을 갖는 프로그램의 메모리에 점유된 공간은 원시 프로그램에 의해 점유된 메모리 공간보다

현저히 적다. 따라서, 프로그램은 압축된 명령 패킷들로 압축됨으로써, 유효 메모리 공간의 사용이 최대화된다. 다음으로, 아래 설명되는 바의 압축 해제 시스템이 처리 유닛들에 의한 실행 직전에 압축된 명령 패킷들의 시퀀스를 VLIW들로 되돌려 압축 해제시킨다.

도 5는 VLIW 프로세서(20)에 의한 실행 전에 압축된 명령 패킷들을 압축 해제하는, 본 발명에 따른 압축 해제 시스템(170)의 블록도이다. 압축 해제 시스템은 예를 들면 ECU 내에 위치할 수 있다. 압축 해제 시스템(170)은 128비트 폭일 수 있는 명령 메모리(172)를 액세스할 수 있다. 명령 메모리는 복수의 압축된 명령 패킷들을 포함할 수 있다. 명령 메모리의 폭은 본 발명의 범위에서 벗어나지 않는 임의의 목적하는 크기로 선택될 수 있다. 이러한 압축된 명령 패킷들은 디폴트 명령이 존재하지 않는 한, 도 1에 도시된 처리 유닛들 각각에 대해 압축된 명령을 포함할 수 있다. 명령 메모리는 제 1 의 64비트 메모리(174) 및 제 2 의 64비트 메모리(176)로 구성될 수 있다. 명령 메모리(172)는 ECU(26)에 의해 제어된다. 128비트의 모든 클록 주기는, 예를 들면, 명령 메모리로부터 어떤 비트들이 현재 명령 패킷을 구성하는지를 결정하고, 명령 패킷 내의 압축된 명령들을 압축 해제하고 모든 처리 유닛에 대해 미압축된 명령들을 갖는 160비트 폭의 매우 긴 명령어를 출력하는 디컴프레서(178)로 판독될 수 있다. 간단히 말하자면, 디컴프레서(178)는 명령 패킷 내의 압축된 명령들을 분리하고, 이어서 각각의 압축된 명령을 압축 해제하고, 이를 그의 대응하는 프로세서에 적용한다. 각각의 압축된 명령을 압축 해제하기 위해, 토큰 필드, 소스 레지스터 필드 및 목적지 레지스터 필드가 확장되고, 확장된 데이터가 미압축된 명령어를 형성하기 위해 함께 조합된다. 각각의 이들 미압축된 명령어들 및 임의의 디폴트 명령들은 160비트의 매우 긴 명령어를 형성하기 위해 함께 조합된다. 디컴프레서의 오퍼레이션이 이하에서 보다 상세히 설명될 것이다.

디컴프레서로부터 160비트의 매우 긴 명령어가 160비트 폭일 수 있는 매우 긴 명령 레지스터(180)로 판독된다. 매우 긴 명령 레지스터는 미압축된 160비트의 매우 긴 명령어를 저장하고, 이들 명령들을 각각의 개별적인 처리 유닛으로 라우팅한다. 명령들의 처리 유닛들로의 라우팅은 또한 디컴프레서에 의해 행해질 수도 있다. 도시된 바와 같이, 매우 긴 명령 레지스터는 32비트 명령어를 ECU(26)로 라우팅하고, 48비트 명령어를 MEM 유닛(36)으로 라우팅하며, 16비트 명령을 RCU 유닛(34)으로 라우팅하고, 32비트 명령을 ALU(32)로 라우팅하며, 32비트 명령어를 MUL 유닛(30)으로 라우팅한다. 매우 긴 명령 레지스터의 보다 상세한 오퍼레이션이 이후에 설명될 것이다. 따라서, 오퍼레이션에서, 16비트 내지 128비트 길이일 수 있는 압축된 명령 패킷은 처리 유닛들의 각각으로 라우팅되는 160비트의 매우 긴 명령어로 되돌려 압축 해제된다. 압축 시스템과 조합된 이러한 압축 해제 유닛은 매우 긴 명령어를 저장하는데 필요한 메모리의 양을 감소시키고, 보다 큰 프로그램이 명령 메모리에 저장될 수 있게 한다. 이제, 압축 해제 시스템에 대한 보다 상세한 사항이 설명될 것이다.

도 6은 본 발명에 따른 압축 해제 시스템(170)의 보다 상세한 도면이다. 위에서 설명된 바와 같이, 압축 해제 시스템은 명령 메모리(172), 디컴프레서(178) 및 매우 긴 명령 레지스터(180)를 포함할 수 있다. 압축 해제 시스템(170)은 또한 어드레스 발생기(190)를 포함할 수 있다. 어드레스 발생기는 ECU 유닛에 의해 제공된 시작 어드레스로부터 다음_패킷_시작 어드레스를 발생시킬 수 있다. 명령 메모리가 위에서 설명된 바와 같은, 실제로 2개의 메모리 부분들인 경우, 어드레스 발생기는 제 1 메모리를 어드레싱하기 위한 제 1 어드레스(ALEFT) 및 제 2 메모리를 어드레싱하기 위한 제 2 어드레스(ARIGHT)를 발생시킬 수도 있다. ALEFT 어드레스를 발생시키기 위해, 오프셋 회로(192)는 4를 시작 어드레스에 부가할 수 있고, 시작 어드레스의 비트들을 우측으로 3개의 위치들을 이동시킬 수 있다. 이어서, 이동된 ALEFT 어드레스는 ALEFT 레지스터(196)에 저장될 수 있다. 마찬가지로, ARIGHT 어드레스는 3개의 위치들만큼 우측으로 시작 어드레스의 비트들을 이동시킬 수 있는 제 2 오프셋 회로(194)를 사용함으로써 발생될 수 있다. ARIGHT 어드레스는 ARIGHT 레지스터(198)에 저장될 수 있다.

본 실시예에서 명령 메모리(172)는 전체 128비트 폭일 수 있다. 그러나, 128비트 폭의 메모리는 128비트 폭 메모리를 형성하기 위해 논리적으로 함께 접속될 수 있는 보다 작은 메모리들을 제조하는 데 실질적이지 못하다. 도시된 바와 같이, 명령 메모리는 랜덤 액세스 메모리(RAM)(204) 및 판독 전용 메모리(ROM)(206)일 수 있다. 명령 메모리는 또한 단지 RAM이거나 또는 단지 ROM일 수 있다. RAM(204)에 대해, 128비트 폭 메모리는 제 1의 64비트 메모리(208) 및 제 2의 64비트 메모리(210)로서 구현될 수 있다. 제 1 및 제 2 메모리들은 연속적으로 라벨되는 16비트 부분들로 분할된다. 도시된 바와 같이, 아래에 보다 상세히 기재하는 어드레싱 도식은 단일의 128비트 메모리로서 두 메모리들을 어드레스함으로써 메모리들로부터 2개의 64비트 출력들은 단일의 128비트 데이터 스트림으로 조합된다. ROM(206) 명령 메모리가 이용되는 경우, 128비트 명령 메모리는 4개의 32비트 폭 ROM들(212, 214, 216 및 218)로서 구현될 수 있다. RAM들에 의해서와 같이, ROM들은 128비트 폭 메모리로서 어드레스되고, 각각의 ROM으로부터 별개의 32비트 출력들은 128비트 출력을 형성하기 위해 함께 조합된다.

RAM들(208, 210)로부터의 출력 및 ROM들(212, 214, 216 및 218)로부터의 출력 모두는 RAM 데이터가 액세스되었는지 여부 또는 ROM 데이터가 액세스되었는지 여부를 선택하는 디컴프레서(178) 내의 선택기(220)에 도입된다. 선택기의 출력은 압축된 명령들과 함께 적어도 하나의 가변 길이 명령 패킷을 포함하는 단일의 128비트 폭 데이터 스트림이다. 페치된 가변 길이 명령 패킷은 명령 패킷들 내의 압축된 명령들을 압축 해제할 수 있는 라우팅 선택 논리 유닛(222)에 도입될 수

있다. RAM들 및 ROM들로부터의 64비트 데이터 부분들은 아래 설명될 바와 같이 스와프될 수 있고, 128비트 길이의 데이터 스트림이 64비트 메모리 블록들 중 하나에서 시작될 수 있다. 이러한 스와핑은 제 1 및 제 2 메모리 블록 중 하나에서 시작하는 데이터 스트림에 대한 별개의 디코더가 필요치 않기 때문에 압축 해제에 필요한 하드웨어를 축소시킨다. 라우팅 선택 논리 유닛은 프로그램된 논리 어레이(PLA)일 수 있다. 라우팅 선택 논리는 또한 아래에서 설명하는 바와 같이 미압축된 명령들을 적절한 처리 유닛들로 라우팅시킬 수 있다.

압축된 명령들을 압축 해제하기 위해, 라우팅 선택 논리(222)는 유입되는 명령 패킷 스트림을 수신하고, 라우팅 선택 논리가 위에서 설명된 바와 같이 명령 패킷의 최종 압축된 명령의 패킷의 말단 지시자(스톱 비트 또는 EP/NEP 비트들)에 위치되기 때문에 명령 패킷이 종료되는 것을 결정할 수 있다. 라우팅 및 선택 논리(222)는 명령 패킷의 말단을 결정한 후 명령 패킷 내의 압축된 명령들을 분리시키고, 임의의 디폴트 명령들을 마찬가지로 삽입한다. 따라서, 라우팅 및 선택 논리가 명령 패킷의 제 1 압축된 명령이 패킷의 말단에 있음을 검출한 경우, 그 명령은 압축 해제되고, 디폴트 명령은 모든 다른 처리 유닛들에 대해 발생된다. 이러한 실시예에서, 명령 패킷의 최소 크기는 16비트이므로, 모든 처리 유닛들에 대한 디폴트 명령들을 포함하는 명령 패킷은 16비트 길이이다.

디폴트 명령이 아닌 각각의 압축된 명령은 먼저 VLIW 프로세서 내의 모든 토큰들과 압축된 명령의 토큰을 비교하고, 대응하는 미압축된 명령 op 코드를 발생시킴으로써 디코딩될 수 있다. 토큰들의 어떤 미압축된 명령들로의 할당은 위에서 설명된 바와 같은 압축 시스템에 의해 수행된다. 일단 적절한 미압축된 명령 op 코드가 결정되면 그것은 일시적으로 저장된다. 다음으로, 라우팅 및 선택 논리가 위에서 설명된 바와 같은 소스 레지스터 필드를 판독하고, 시스템 내의 32개의 레지스터들 중의 하나를 어드레스하는 소스 레지스터 어드레스를 발생시킨다. 다음으로, 라우팅 및 선택 논리는 목적지 레지스터 필드를 판독하고 사용할 수 있는 레지스터들 중의 하나에 대응하는 목적지 레지스터 어드레스를 발생시킨다. 이어서 라우팅 및 선택 논리는 op 코드, 소스 레지스터 어드레스, 목적지 레지스터 어드레스 및 미압축된 명령을 형성하기 위해 임의의 추가 비트들을 함께 조합한다. 각각의 압축된 명령들이 압축 해제된 후, 임의의 디폴트 명령들이 발생되고, 라우팅 및 선택 논리(222)는 복수의 멀티플렉서들(224, 226, 228, 230 및 232)을 제어함으로써 각각의 처리 유닛들에 미압축된 명령들을 라우팅시킨다. 명령들은 매우 긴 명령 레지스터(180)에 일시적으로 저장된다. 예를 들면, ECU 유닛에 대한 매우 긴 명령어 내의 명령어는 라우팅 및 선택 논리가 ECU 멀티플렉서만을 선택적으로 인에이블시키기 때문에 ECU 멀티플렉서(224)로 라우팅된다. 마찬가지로, 각각의 미압축된 명령들은 적절한 처리 유닛으로 라우팅된다.

삭제

도 7은 복수의 압축된 명령 패킷들을 갖는 명령 메모리(172)의 도면이다. 위에서 설명된 바와 같이, 명령 메모리는 64비트 폭일 수 있는 제 1 메모리(174) 및 역시 64비트 폭일 수 있는 제 2 메모리(176)를 포함할 수 있다. 복수의 압축된 명령들을 포함하는 복수의 명령 패킷들은 명령 메모리에 저장된다. 위에서 설명한 바와 같이, 128비트는 매 클록 주기당 명령 메모리로 판독된다. 많은 경우들에서, 128비트는 명령 패킷들의 길이가 변화하기 때문에 1개 이상의 명령 패킷을 포함할 수 있다. 이러한 시스템에 대한 어드레싱 도식은 도 8을 참조하여 아래에 설명될 것이다. IP0 라벨된 제 1 명령 패킷(250)은 32비트 길이일 수 있고, 명령 메모리 내의 임의의 16비트 경계에 임의로 정렬될 수 있다. 따라서, IP0 패킷은 명령 메모리에서 임의의 16비트 경계(즉, 0비트, 16비트, 32비트, 48비트, 64비트, 80비트, 96비트 또는 112비트 길이) 상에서 시작할 수 있다. 80비트 길이와 동일하거나 또는 이하의 임의의 명령 패킷(즉, 도시된 실시예에서 80, 64, 48, 32 또는 16비트)은 명령 메모리 내의 16비트 경계들 상에 임의로 할당될 수 있다. 80비트보다 큰 임의의 명령 패킷(즉, 도시된 실시예에 대해 96, 112 또는 128)은 아래에 설명되는 바와 같이 명령 메모리의 특정 어드레스들에서 시작되거나 또는 정렬되어야 한다.

IP0의 명령들이 실행되는 제 1 클록 주기 후, IP1로 라벨되고 16비트 폭인 다음 명령 패킷(252)은 IP0 명령과 동일한 시점에 ECU에 의해 판독되지만, 제 2 클록 주기까지 압축 해제되지 않고 실행되지 않는다. IP2로 라벨된 제 3 명령 패킷(254)은 48비트 길이일 수 있고, 제 1 메모리와 제 2 메모리 간의 경계선을 교차한다. 그러나, 이 시스템에서, 제 1 및 제 2 메모리들은 128비트를 검색하도록 병렬로 어드레스됨으로써 명령 패킷들은 메모리들 간의 경계선을 교차할 수 있다. IP3으로 라벨된 명령 패킷(256)은 64비트 폭이고, 메모리의 128비트 경계를 교차하고 다음 메모리 위치에 계속되어야 한다. 128비트 경계에 교차하는 명령 메모리의 임의의 명령 패킷을 정확히 판독하기 위해, 제 1 메모리는 제 2 메모리 상의 어드레스보다 더 큰 어드레스로 어드레스되어야 하며, 따라서 제 2 메모리 내의 데이터가 먼저 판독되고, 이어서, 위에서 설명한 바와 같이 메모리 블록들을 스와프시키는 제 1 메모리의 데이터가 판독된다. 제 1 메모리 및 제 2 메모리를 어드레스하는 시스템이 이후로 설명될 것이다.

IP4로 라벨된 제 5 명령 패킷(258)은 128비트 길이이고, 임의로 정렬될 수 없다. 128비트 길이인 임의의 명령 패킷에 대해, 명령 패킷은 명령 메모리의 시작점(비트-0) 또는 메모리의 중간 지점(비트 64)에서만 시작할 수 있다. 이러한 정렬은,

예를 들면, 128비트 명령 패킷이 16비트 경계에 정렬되는 경우, 명령 패킷은 제 1 메모리 어드레스들(16-64), 제 2 메모리 어드레스들(65-128), 및 제 1 메모리 어드레스들(0-16)을 점유할 것이기 때문에 필요하다. 하드웨어 어드레싱 시스템은 명령 패킷의 2가지 상이한 부분들이 제 1 메모리에 포함되기 때문에 명령 패킷을 용이하게 어드레스할 수 없다.

도시된 실시예에 대한 가변 길이 명령 패킷들에 대해 허용된 할당들의 예가 설명될 것이다. 128비트 길이 패킷은 메모리 어드레스(0 또는 64)중 하나에서 시작할 수 있다. 112비트 길이 패킷은 메모리 어드레스(0, 16, 64 또는 80)에서 시작할 수 있다. 96비트 길이 패킷은 메모리 어드레스(0, 16, 32, 64, 80 또는 96)에서 시작할 수 있다. 80비트 길이 또는 더 짧은 명령 패킷(즉, 80, 64, 48, 32 또는 16비트)은 메모리 어드레스(0, 16, 32, 48, 64, 80, 96, 112 또는 128)에서 시작할 수 있다.

80비트보다 큰 명령 패킷들이 명령 메모리 내에 적절히 정렬되는 것을 보장하기 위해, 긴 명령 패킷들이 적절한 경계에서 시작되는 것을 보장하기 위해 사용되는 패드 명령(260)이 제공된다. 패드 명령은 압축 시스템에 의해 명령 패킷 스트림으로 삽입되고, 이어서, 압축 해제 시스템에 의해 낭비된다. 패드 명령은 임의의 오퍼레이션이 처리 유닛들에서 발생되지 않도록 한다. 이제, 명령 패킷들의 판독을 수행하기 위한 제 1 및 제 2 메모리의 어드레싱을 위한 시스템이 설명될 것이다.

도 8은 제 1 및 제 2 메모리로부터 명령 패킷들을 수행하는 어드레싱 시스템을 보여주는 도면이다. 제 1 메모리의 메모리 맵(280) 및 제 2 메모리의 메모리 맵(282)이 도시된다. 이들 메모리 맵들은 명령 패킷의 가장 작은 크기이기 때문에 16비트 세그먼트들로 분할된다. 도시된 바와 같이, 제 1 메모리 위치는 4개의 16비트 세그먼트들에 대응하는 어드레스들(0, 1, 2 및 3)을 갖고, 제 2 메모리는 어드레스들(4, 5, 6 및 7)을 갖고, 제 1 메모리는 어드레스들(8, 9, 10 및 11)을 갖는다. 따라서, 제 1 및 제 2 메모리들은 하나의 큰 128비트 메모리로 어드레스된다.

명령 메모리의 128비트 부분이 클록 주기 0에서 메모리들로부터 판독되는 제1 시점에서, ALEFT 및 ARIGHT 어드레스들은 모두 0으로, 어드레스들(0, 1, 2, 3, 4, 5, 6 및 7)이 판독될 수 있음을 의미한다. IP0에 대해, 위에서 설명한 바와 같이, 32비트 명령 패킷에 대응하는 어드레스들(0 및 1)은 판독되고 압축 해제된다. 이어서, 어드레싱 시스템은 다음 명령 패킷(IP1)이 단지 16비트이고, 메모리는 증가될 필요가 없음을 결정한다. 따라서, ALEFT 및 ARIGHT 어드레스들은 0으로 유지된다. 이후, 명령 패킷(IP1)은 명령 메모리로부터 판독되고, 압축 해제되고, 실행된다. IP2는 단지 48비트 길이이기 때문에, 어드레싱 시스템은 ARIGHT 또는 ALEFT 어드레스들 중 어느 것도 증가시키지 않고, IP2가 판독된다. 그러나, 어드레싱 시스템은 IP3이 128비트 경계를 교차하는 것을 결정함으로써 ALEFT 어드레스는 그에 의해 증가되고 어드레스들(8, 9, 10 및 11)이 액세스될 수 있다. 따라서, IP3은 위치들(6, 7, 8 및 9)로부터 판독된다. IP3을 판독하기 위해, 제 2 메모리의 데이터가 먼저 판독되고, 제 1 메모리의 데이터가 다음으로 판독된다.

어드레싱 시스템은 이후 다음 명령 패킷(IP4)이 128비트 길이이고, 그 앞에 패드 명령을 갖는 것을 또한 결정한다. 어드레싱 시스템이 패드 명령과 마주침에 따라, 이는 패드 명령을 판독하고 이를 낭비한다. 이어서, 다음 명령 패킷이 128비트 길이이기 때문에, ARIGHT 및 ALEFT 어드레스들 모두가 증가됨으로써 프로세서는 어드레스들(16, 17, 18, 19, 12, 13, 14 및 15)을 액세스할 수 있다. 이러한 경우에, 명령 패킷은 제 2 메모리(즉, 어드레스들(12, 13, 14 및 15))로부터 먼저 판독되어야 하고, 다음으로 제 1 메모리(즉, 어드레스들(16, 17, 18 및 19))로부터 판독되어야 한다. 명령 패킷이 128비트 경계에 교차할 때 존재할 수 있는 제 2 메모리에서 명령 패킷이 시작하는 임의의 시점에서, 제 2 메모리로부터의 데이터는 제 1 메모리의 어드레스보다 적은 어드레스에 의해 판독됨으로써 제 1 메모리의 데이터가 먼저 처리된다.

도 9는 매우 긴 명령어(292)로 압축 해제되는 압축된 명령어(290)를 나타내는 도면이다. 도시된 바와 같이, 압축된 명령 패킷은 단지 80비트 길이이고, 각각의 처리 유닛들에 대한 압축된 명령들을 포함하며, 매우 긴 명령 패킷은 160비트 길이이다. 위에서 설명한 바와 같이, 16비트 ECU 압축된 명령(294)은 32비트 ECU 명령(296)으로 압축 해제된다. 16비트 MEM 압축된 명령(298)은 48비트 MEM 명령(300)으로 압축 해제된다. 16비트 ALU 압축된 명령(302)은 32비트 ALU 명령(304)으로 압축 해제된다. 마찬가지로, 16비트 MUL 압축된 명령(306)은 32비트 MUL 명령(308)으로 압축 해제되고, 16비트 RCU 압축된 명령(310)은 16비트 RCU 명령(312)으로 압축 해제된다. 따라서, 본 발명에 따른 압축 및 압축 해제 시스템은 매우 긴 명령 패킷을 저장하는 데 필요한 메모리의 양을 크게 감소시킬 수 있다.

본 발명의 특정 실시예를 참조하여 기재되었지만, 본 발명의 원리들 및 정신에서 벗어나지 않고 본 실시예의 변화들이 이루어질 수 있음이 당업자들에 의해 이해될 것이고, 본 발명의 범위는 첨부된 청구항들에 의해 정의된다.

(57) 청구의 범위

청구항 1.

복수의 처리 유닛들을 갖는 프로세서에 대해 매우 긴 명령어 패킷들에 포함된 가변 길이의 명령들을 압축 및 압축 해제하는 시스템에 있어서, 각각의 매우 긴 명령어 패킷은 상기 처리 유닛들의 각각에 대한 명령을 포함하고, 상기 시스템은:

복수의 저장 위치들에 복수의 상기 가변 길이 명령 패킷들을 저장하는 수단으로서, 각 명령 패킷은 상기 처리 유닛들 중의 연관된 처리 유닛들에 대한 압축된 명령들을 포함하는, 상기 저장 수단과;

상기 저장 수단으로부터 선택된 가변 길이 명령 패킷을 액세스하고 검색하는 수단과;

상기 가변 길이 명령 패킷을 상기 압축된 명령들로 파싱(parsing)하는 수단과;

각각의 상기 연관된 처리 유닛들에 대한 가변 길이 명령을 발생시키기 위해 상기 압축된 명령들의 각각을 압축 해제하는 수단과;

상기 압축 해제 수단으로부터의 가변 길이의 압축 해제된 명령들을 상기 연관된 처리 유닛들로 라우팅하는 수단을 포함하는, 시스템.

청구항 2.

제 1 항에 있어서, 상기 압축된 명령은 토큰 필드, 소스 레지스터 필드 및 목적지 레지스터 필드를 포함하고, 상기 압축 해제 수단은 상기 토큰 필드로부터 오퍼레이션 코드를 발생시키는 수단, 상기 소스 레지스터 필드로부터 소스 레지스터 어드레스를 발생시키는 수단, 상기 목적지 레지스터 필드로부터 목적지 레지스터 어드레스를 발생시키는 수단, 및 상기 압축되지 않은 명령을 형성하기 위해 상기 오퍼레이션 코드, 상기 소스 레지스터 어드레스 및 상기 목적지 레지스터 어드레스를 조합하는 수단을 포함하는, 시스템.

청구항 3.

제 2 항에 있어서, 토큰들은 상이한 길이들을 가지며, 보다 짧은 길이를 갖는 토큰이 보다 빈번히 발생하는 압축되지 않은 명령에 할당되는, 시스템.

청구항 4.

제 1 항에 있어서, 상기 압축 해제 수단은 상기 가변 길이 명령 패킷에 압축된 명령을 갖지 않는 연관된 처리 유닛에 대한 압축 해제된 디폴트 명령들을 발생시키는 수단을 포함하는, 시스템.

청구항 5.

제 4 항에 있어서, 상기 디폴트 명령 발생 수단은 각각의 상기 처리 유닛들에 대한 상기 디폴트 명령들을 선택적으로 변화시키는 수단을 포함하는, 시스템.

청구항 6.

제 1 항에 있어서, 상기 가변 길이 명령 패킷은 패킷 지시자의 말단을 포함하고, 상기 액세스 및 검색 수단은 패킷 지시자의 상기 말단을 검출하는 수단을 포함하는, 시스템.

청구항 7.

제 1 항에 있어서, 상기 가변 길이 명령 패킷들은 상기 가변 길이 명령 패킷의 상기 가변 길이를 조절하기 위한 패드 명령을 갖고, 상기 파싱 수단은 상기 가변 길이 명령 패킷으로부터 상기 패드 명령을 제거하는 수단을 포함하는, 시스템.

청구항 8.

제 1 항에 있어서, 상기 액세스 및 검색 수단은 상기 가변 길이 명령 패킷의 상기 가변 길이에 기초하여 시작 어드레스를 오프셋하여 명령 패킷 어드레스를 발생시키는 어드레스 발생기 수단 및 상기 명령 패킷 어드레스를 저장하는 수단을 포함하는, 시스템.

청구항 9.

제 8 항에 있어서, 상기 저장 수단은 제 1 메모리 및 제 2 메모리를 포함하고, 상기 명령 패킷 어드레스는 상기 제 1 메모리를 어드레싱하기 위한 제 1 어드레스, 및 상기 제 2 메모리를 어드레싱하기 위한 제 2 어드레스를 포함하여 상기 선택된 명령 패킷이 상기 제 1 메모리로부터 판독되고 이어서 상기 제 2 메모리로부터 판독되는, 시스템.

청구항 10.

제 9 항에 있어서, 상기 명령 패킷이 상기 제 2 메모리와 상기 제 1 메모리 사이의 경계선을 오버랩하도록 상기 명령 패킷의 제 1 부분이 상기 제 2 메모리에 저장되고 상기 명령 패킷의 제 2 부분이 상기 제 1 메모리에 저장될 때 상기 어드레스 발생기는 상기 제 2 메모리로부터 상기 명령 패킷을 판독하고 이어서 상기 제 1 메모리로부터 상기 명령 패킷을 판독하는 수단을 포함하는, 시스템.

청구항 11.

제 1 항에 있어서, 상기 라우팅 수단은 상기 처리 유닛들에 부착된 복수의 멀티플렉서들과, 상기 명령들이 연관된 처리 유닛들로 라우팅되도록 상기 멀티플렉서들을 선택적으로 인에이블시키는 수단을 포함하는, 시스템.

청구항 12.

복수의 처리 유닛들을 갖는 프로세서에 대해 매우 긴 명령어 패킷들에 포함된 가변 길이 명령들을 압축 및 압축 해제하는 방법에 있어서, 각각의 매우 긴 명령어 패킷은 각각의 상기 처리 유닛들에 대한 명령을 포함하고, 상기 방법은:

복수의 저장 위치들에 복수의 상기 가변 길이 명령 패킷들을 저장하는 단계로서, 각 명령 패킷은 상기 처리 유닛들 중 연관된 처리 유닛들에 대한 압축된 명령들을 포함하는, 상기 저장 단계와;

상기 저장 위치들로부터 선택된 가변 길이 명령 패킷을 액세스하고 검색하는 단계와;

상기 가변 길이 명령 패킷을 상기 압축된 명령들로 파싱하는 단계와;

상기 연관된 처리 유닛들의 각각에 대한 가변 길이 명령을 발생시키기 위해 상기 각각의 압축된 명령들을 압축 해제하는 단계와;

상기 압축 해제 단계로부터의 상기 가변 길이의 압축 해제된 명령들을 상기 연관된 처리 유닛들로 라우팅하는 단계를 포함하는, 방법.

청구항 13.

제 12 항에 있어서, 상기 압축된 명령은 토큰 필드, 소스 레지스터 필드 및 목적지 레지스터 필드를 포함하고, 상기 압축 해제 단계는 상기 토큰 필드로부터 오퍼레이션 코드를 발생시키는 단계, 상기 소스 레지스터 필드로부터 소스 레지스터 어드레스를 발생시키는 단계, 상기 목적지 레지스터 필드로부터 목적지 레지스터 어드레스를 발생시키는 단계, 및 상기 압축되지 않은 명령들을 형성하기 위하여 상기 오퍼레이션 코드, 상기 소스 레지스터 어드레스 및 상기 목적지 레지스터 어드레스를 조합하는 단계를 포함하는, 방법.

청구항 14.

제 13 항에 있어서, 토큰들은 상이한 길이들을 갖고, 보다 짧은 길이를 갖는 토큰이 보다 빈번히 발생하는 압축되지 않은 명령에 할당되는, 방법.

청구항 15.

제 12 항에 있어서, 압축 해제 단계는 상기 가변 길이 명령 패킷에 압축된 명령을 갖지 않는 연관된 처리 유닛에 대한 압축 해제된 디폴트 명령들을 발생시키는 단계를 포함하는, 방법.

청구항 16.

제 15 항에 있어서, 상기 디폴트 명령 발생 단계는 각각의 상기 처리 유닛들에 대한 상기 디폴트 명령들을 선택적으로 변화시키는 단계를 포함하는, 방법.

청구항 17.

제 12 항에 있어서, 상기 가변 길이 명령 패킷은 패킷 지시자의 말단을 포함하고, 액세스 및 검색 단계는 패킷 지시자의 상기 말단을 검출하는 단계를 포함하는, 방법.

청구항 18.

제 12 항에 있어서, 상기 가변 길이 명령 패킷들은 상기 가변 길이 명령 패킷의 상기 가변 길이를 조절하기 위한 패드 명령을 갖고, 파싱 단계는 가변 길이 명령 패킷으로부터 상기 패드 명령을 제거하는 단계를 포함하는, 방법.

청구항 19.

제 12 항에 있어서, 상기 액세스 및 검색 단계는 상기 가변 길이 명령 패킷의 상기 가변 길이에 기초하여 시작 어드레스를 오프셋하여 명령 패킷 어드레스를 발생시키는 단계 및 상기 명령 패킷 어드레스를 저장하는 단계를 포함하는, 방법.

청구항 20.

제 19 항에 있어서, 상기 저장 단계는 제 1 메모리 및 제 2 메모리를 사용하는 단계를 포함하고, 상기 명령 패킷 어드레스는 상기 제 1 메모리를 어드레싱하기 위한 제 1 어드레스 및 상기 제 2 메모리를 어드레싱하기 위한 제 2 어드레스를 포함하여 상기 선택된 명령 패킷이 상기 제 1 메모리로부터 판독되고 이어서 상기 제 2 메모리로부터 판독되는, 방법.

청구항 21.

제 20 항에 있어서, 상기 명령 패킷이 상기 제 2 메모리와 상기 제 1 메모리 사이의 경계선을 오버랩하도록 상기 명령 패킷의 제 1 부분이 상기 제 2 메모리에 저장되고 상기 명령 패킷의 제 2 부분이 상기 제 1 메모리에 저장될 때 상기 어드레스 발생 단계는 상기 제 2 메모리로부터 상기 명령 패킷을 판독하고 이어서 상기 제 1 메모리로부터 상기 명령 패킷을 판독하는 단계를 포함하는, 방법.

청구항 22.

제 12 항에 있어서, 상기 라우팅 단계는 상기 처리 유닛들에 부착된 복수의 멀티플렉서들을 사용하는 단계를 포함하고, 상기 명령들이 연관된 처리 유닛들로 라우팅되도록 상기 멀티플렉서들을 선택적으로 인에이블시키는 단계를 포함하는, 방법.

청구항 23.

복수의 처리 유닛들을 갖는 프로세서에 대해 매우 긴 명령어 패킷들에 포함된 가변 길이의 명령들을 압축 및 압축 해제하는 시스템에 있어서, 각각의 매우 긴 명령어 패킷은 상기 처리 유닛들의 각각에 대한 명령을 포함하고, 상기 시스템은:

상기 처리 유닛들의 대응하는 처리 유닛들에 대한 명령들을 포함하는 매우 긴 명령어 패킷을 발생시키는 수단과;

상기 명령어 패킷 내의 각 명령에 압축된 명령 코드를 할당하는 수단으로서, 상기 코드들은 상이한 길이들을 가지며, 보다 짧은 압축된 명령 코드는 보다 빈번히 발생하는 명령들에 할당되는, 상기 할당 수단과;

상기 처리 유닛들의 대응하는 처리 유닛들에 대한 압축된 명령들을 갖는 명령 패킷을 발생시키기 위해서 각 명령에 대한 상기 압축된 명령 코드들을 조합하는 수단을 포함하는, 시스템.

청구항 24.

제 23 항에 있어서, 상기 명령들은 오퍼레이션 코드, 소스 레지스터 어드레스, 및 목적지 레지스터 어드레스를 포함하고, 상기 할당 수단은 상기 오퍼레이션 코드로부터 토큰 필드를 발생시키는 수단, 상기 소스 레지스터 어드레스로부터 소스 레지스터 필드를 발생시키는 수단, 상기 목적지 레지스터 어드레스로부터 목적지 레지스터 필드를 발생시키는 수단을 포함하고, 상기 토큰 필드, 상기 소스 레지스터 필드 및 상기 목적지 레지스터 필드를 상기 압축된 명령으로 조합하는 수단을 더 포함하는, 시스템.

청구항 25.

제 24 항에 있어서, 상기 조합 수단은 패킷 지시자의 말단을 상기 명령 패킷 내의 상기 최종 압축된 명령에 부가하는 수단을 포함하는, 시스템.

청구항 26.

제 23 항에 있어서, 상기 할당 수단은 0(zero) 길이 압축된 명령들을 상기 디폴트 오퍼레이션에 할당함으로써 상기 매우 긴 명령어 패킷으로부터 디폴트 명령을 제거하는 수단을 포함하는, 시스템.

청구항 27.

제 23 항에 있어서, 상기 조합 수단은 상기 압축된 명령 패킷이 명령 메모리에 정확하게 정렬되도록 복수의 패드 비트들을 상기 압축된 명령 패킷에 삽입하는 수단을 포함하는, 시스템.

청구항 28.

복수의 처리 유닛들을 갖는 프로세서에 대해 매우 긴 명령어 패킷들에 포함된 가변 길이의 명령들을 압축 및 압축 해제하는 방법에 있어서, 각각의 매우 긴 명령어 패킷은 상기 처리 유닛들의 각각에 대한 명령을 포함하고, 상기 방법은:

상기 처리 유닛들의 대응하는 처리 유닛들의 각각에 대한 명령들을 포함하는 매우 긴 명령어 패킷을 발생시키는 단계와;

상기 명령어 패킷 내의 각 명령에 압축된 명령 코드를 할당하는 단계로서, 상기 코드들은 상이한 길이들을 가지며 보다 짧은 압축된 명령 코드는 보다 빈번히 발생하는 명령들로 할당되는, 상기 할당 단계와;

상기 처리 유닛들의 대응하는 처리 유닛들에 대한 압축된 명령들을 갖는 명령 패킷을 발생시키기 위해서 각 명령에 대한 상기 압축된 명령 코드들을 조합하는 단계를 포함하는, 방법.

청구항 29.

제 28 항에 있어서, 상기 명령들은 오퍼레이션 코드, 소스 레지스터 어드레스, 및 목적지 레지스터 어드레스를 포함하고, 상기 할당 단계는 상기 오퍼레이션 코드로부터 토큰 필드를 발생시키는 단계, 상기 소스 레지스터 어드레스로부터 소스 레지스터 필드를 발생시키는 단계, 상기 목적지 레지스터 어드레스로부터 목적지 레지스터 필드를 발생시키는 단계를 포함하고, 상기 할당 단계는 상기 토큰 필드, 상기 소스 레지스터 필드 및 상기 목적지 레지스터 필드를 상기 압축된 명령으로 조합하는 단계를 더 포함하는, 방법.

청구항 30.

제 29 항에 있어서, 조합 단계는 패킷 지시자의 말단을 상기 명령 패킷 내의 상기 최종 압축된 명령에 부가하는 단계를 포함하는, 방법.

청구항 31.

제 28 항에 있어서, 상기 할당 단계는 0 길이 압축된 명령들을 상기 디폴트 오퍼레이션에 할당함으로써 상기 매우 긴 명령어 패킷으로부터 디폴트 명령을 제거하는 단계를 포함하는, 방법.

청구항 32.

제 28 항에 있어서, 상기 조합 단계는 상기 압축된 명령 패킷이 명령 메모리에 정확하게 정렬되도록 복수의 패드 비트들을 상기 압축된 명령 패킷에 삽입하는 단계를 포함하는, 방법.

청구항 33.

복수의 처리 유닛들을 갖는 프로세서에 대해 매우 긴 명령어 패킷들에 포함된 가변 길이의 명령들을 압축 및 압축 해제하는 시스템에 있어서, 각각의 매우 긴 명령어 패킷은 상기 처리 유닛들의 각각에 대한 명령을 포함하고, 상기 시스템은:

상기 처리 유닛들의 대응하는 처리 유닛들에 대한 명령들을 포함하는 매우 긴 명령어 패킷을 발생시키는 수단과;

상기 명령어 패킷 내의 각 명령에 압축된 명령 코드를 할당하는 수단으로서, 상기 코드들은 상이한 길이들을 가지며, 보다 짧은 압축된 명령 코드는 보다 빈번히 발생하는 명령들에 할당되는, 상기 할당 수단과;

상기 처리 유닛들의 대응하는 처리 유닛들에 대한 압축된 명령들을 갖는 명령 패킷을 발생시키기 위해서 각 명령에 대한 상기 압축된 명령 코드들을 조합하는 수단과;

복수의 저장 위치들에 복수의 상기 가변 길이 명령 패킷들을 저장하는 수단으로서, 각 명령 패킷은 상기 처리 유닛들 중의 연관된 처리 유닛들에 대한 압축된 명령들을 포함하는, 상기 저장 수단과;

상기 저장 수단으로부터 선택된 가변 길이 명령 패킷을 액세스하고 검색하는 수단과;

상기 가변 길이 명령 패킷을 상기 압축된 명령들로 파싱하는 수단과;

상기 각각의 연관된 처리 유닛들에 대한 가변 길이 명령을 발생시키기 위해 상기 압축된 명령들의 각각을 압축 해제하는 수단과;

상기 압축 해제 수단으로부터의 가변 길이의 압축 해제된 명령들을 상기 연관된 처리 유닛들로 라우팅하는 수단을 포함하는, 시스템.

청구항 34.

복수의 처리 유닛들을 갖는 프로세서에 대해 매우 긴 명령어 패킷들에 포함된 가변 길이의 명령들을 압축 및 압축 해제하는 방법에 있어서, 각각의 매우 긴 명령어 패킷은 상기 처리 유닛들의 각각에 대한 명령을 포함하고, 상기 방법은:

상기 처리 유닛들의 대응하는 처리 유닛들의 각각에 대한 명령들을 포함하는 매우 긴 명령어 패킷을 발생시키는 단계와;

상기 명령어 패킷 내의 각 명령에 압축된 명령 코드를 할당하는 단계로서, 상기 코드들은 상이한 길이들을 가지며, 보다 짧은 압축된 명령 코드는 보다 빈번히 발생하는 명령들에 할당되는, 상기 할당 단계와;

상기 처리 유닛들의 대응하는 처리 유닛들에 대한 압축된 명령들을 갖는 명령 패킷을 발생시키기 위해서 각 명령에 대한 상기 압축된 명령 코드들을 조합하는 단계와;

복수의 저장 위치들에 복수의 상기 명령 패킷들을 저장하는 단계와;

상기 저장 수단으로부터 선택된 명령 패킷을 액세스하고 검색하는 단계와;

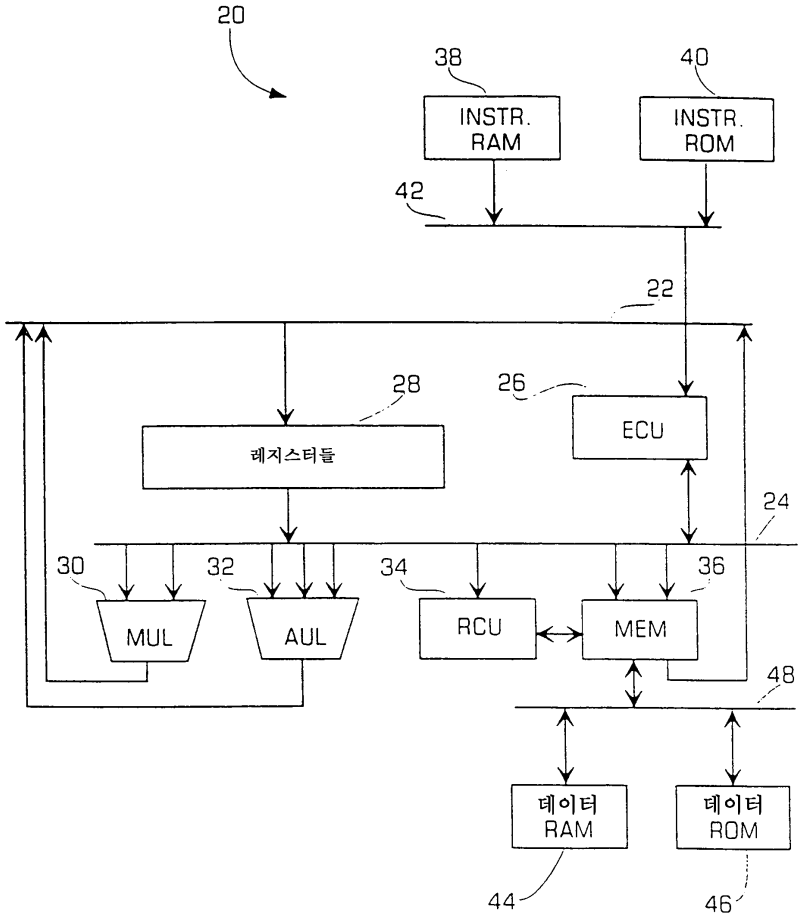
상기 명령 패킷을 상기 압축된 명령들로 파싱하는 단계와;

상기 연관된 처리 유닛들의 각각에 대한 가변 길이 명령을 발생시키기 위해 상기 각각의 압축된 명령들을 압축 해제하는 단계와;

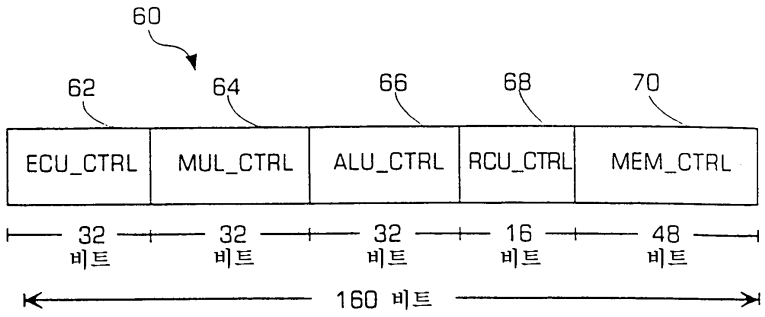
상기 압축 해제 단계로부터의 상기 가변 길이의 압축 해제된 명령들을 상기 연관된 처리 유닛들로 라우팅하는 단계를 포함하는, 방법.

도면

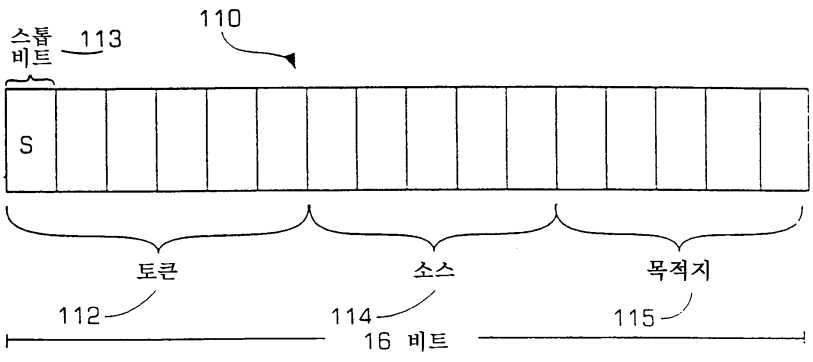
도면1


20
38
INSTR. RAM
40
INSTR. ROM
42
22
28
레지스터들
26
ECU
24
30
32
34
36
MUL
AUL
RCU
MEM
48
데이터 RAM
데이터 ROM
44
46

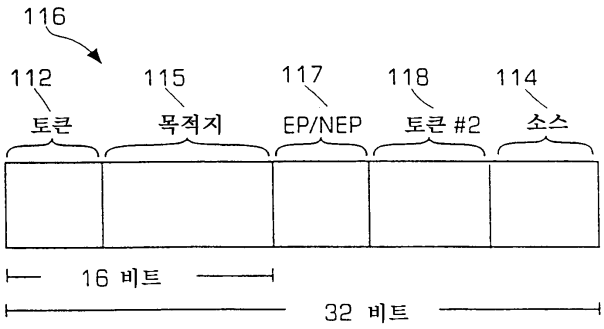
도면2


60
62
64
66
68
70
ECU_CTRL
MUL_CTRL
ALU_CTRL
RCU_CTRL
MEM_CTRL
32 비트
32 비트
32 비트
16 비트
48 비트
160 비트

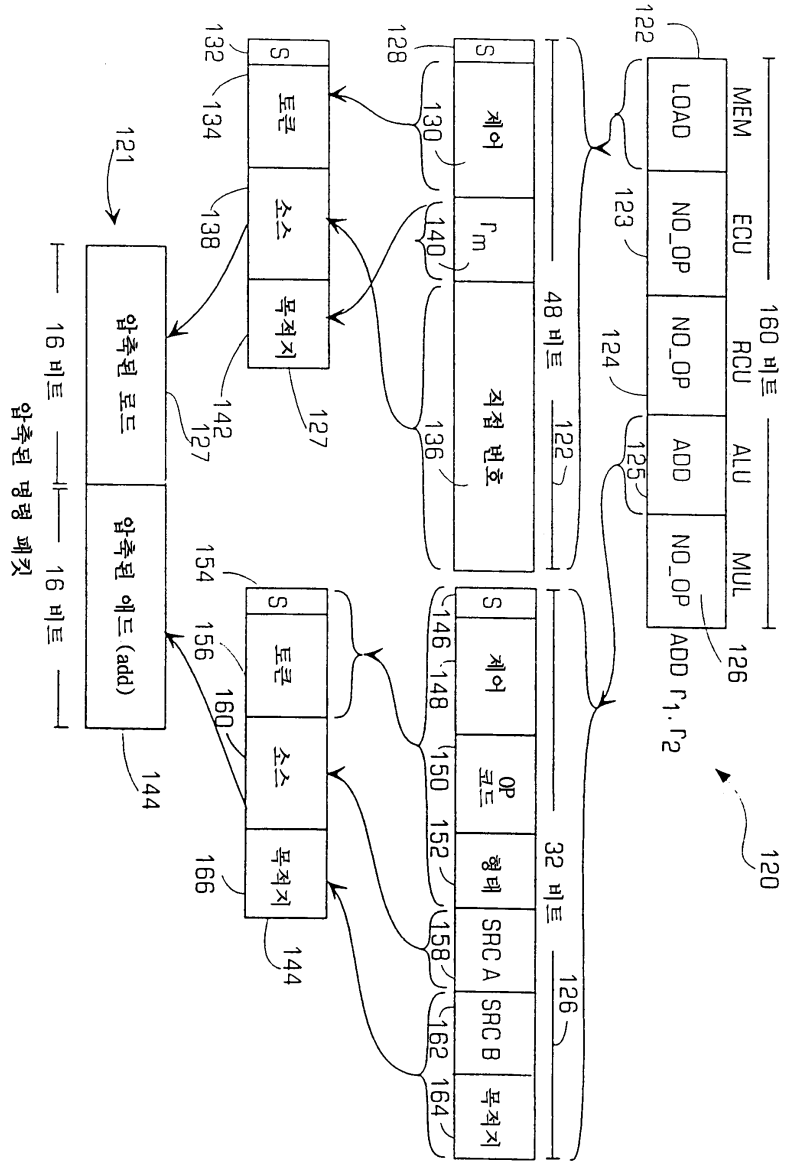
도면3a


스톱 비트
113
110
S
토큰
소스
목적지
112
114
115
16 비트

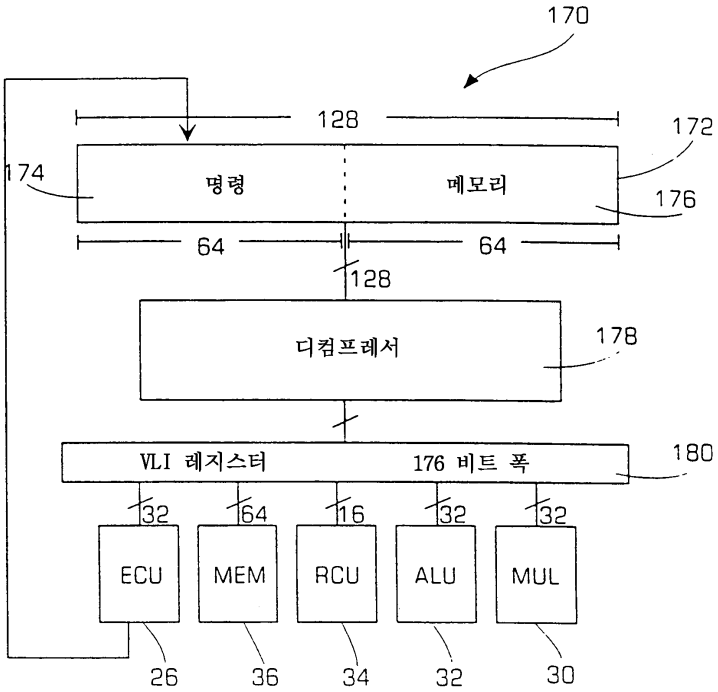
도면3b


116
112
토큰
115
목적지
117
EP/NEP
118
토큰 #2
114
소스
16 비트
32 비트

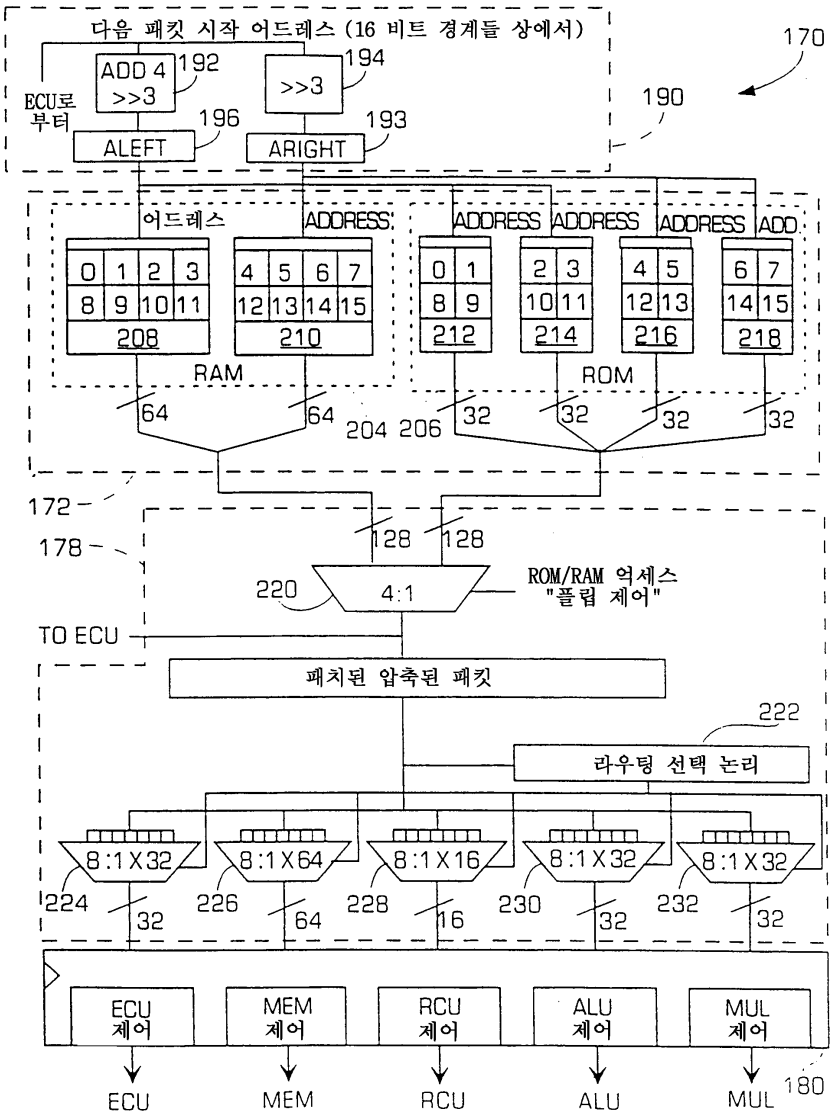
도면4


MEM
ECU
RCU
ALU
MUL
160 비트
LOAD
NO_OP
NO_OP
ADD
NO_OP
ADD r1, r2
122
123
124
125
126
120
48 비트
32 비트
S
제어
rm
직접 번호
토큰
소스
목적지
OP 코드
형태
SRC A
SRC B
목적지
121
압축된 로드
압축된 애드 (add)
16 비트
16 비트
압축된 명령 패킷
127
128
130
132
134
136
138
140
142
144
146
148
150
152
154
156
158
160
162
164
166

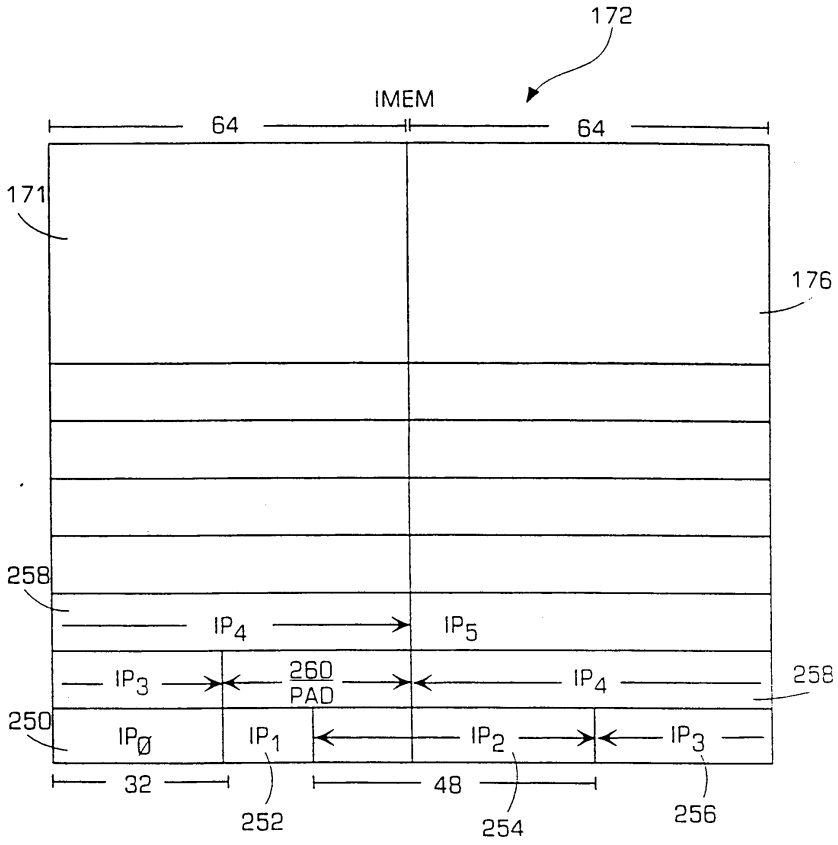
도면5


170
128
172
174
명령
메모리
176
64
64
128
디컴프레서
178
VLI 레지스터
176 비트 폭
180
32
64
16
32
32
ECU
MEM
RCU
ALU
MUL
26
36
34
32
30

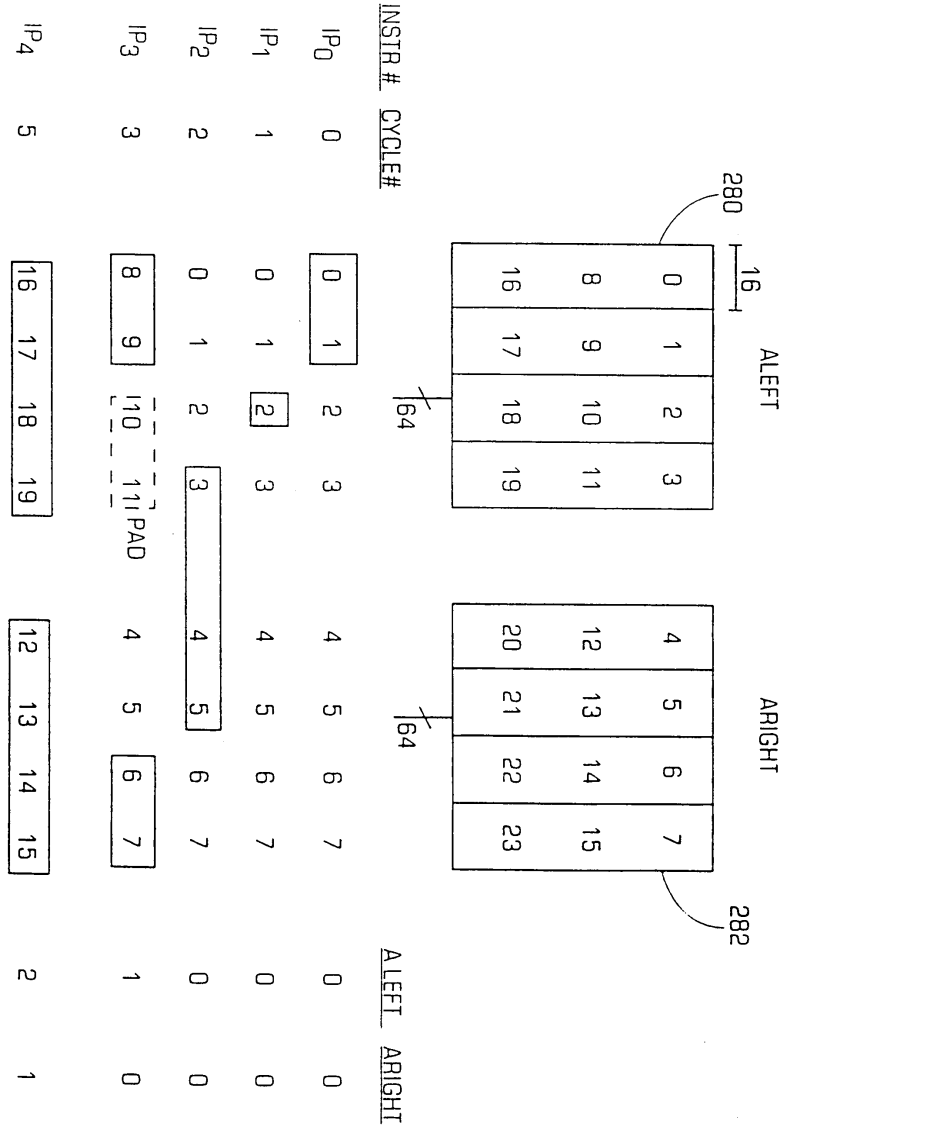
도면6


다음 패킷 시작 어드레스 (16 비트 경계들 상에서)
ADD 4 >>3
192
>>3
194
ECU로부터
196
ALEFT
193
ARIGHT
190
170
어드레스
ADDRESS
ADDRESS
ADDRESS
ADDRESS
ADD
0 1 2 3
8 9 10 11
208
4 5 6 7
12 13 14 15
210
RAM
0 1
8 9
212
2 3
10 11
214
4 5
12 13
216
6 7
14 15
218
ROM
64
64
204
206
32
32
32
32
172
178
128
128
220
4:1
ROM/RAM 억세스
"플립 제어"
TO ECU
패치된 압축된 패킷
222
라우팅 선택 논리
8:1 X 32
8:1 X 64
8:1 X 16
8:1 X 32
8:1 X 32
224
32
226
64
228
16
230
32
232
32
ECU 제어
MEM 제어
RCU 제어
ALU 제어
MUL 제어
180
ECU
MEM
RCU
ALU
MUL

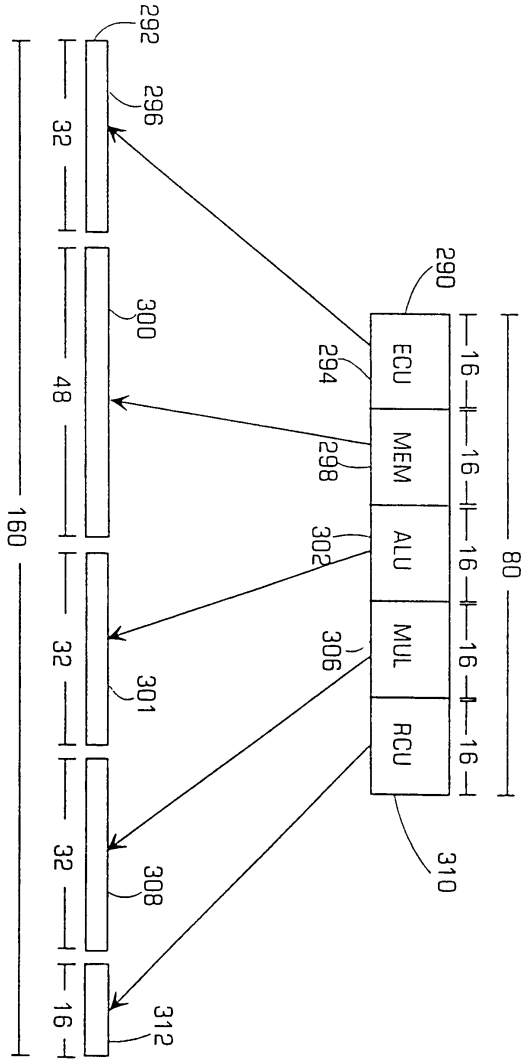
도면7


172
IMEM
64
64
171
176
258
IP4
IP5
IP3
260
PAD
IP4
258
250
IPØ
IP1
IP2
IP3
32
48
252
254
256

도면8


INSTR #
CYCLE#
IP0
IP1
IP2
IP3
IP4
ALEFT
ARIGHT
280
282
16
64
64
PAD

도면9


292
296
32
300
48
294
298
302
32
306
301
32
308
16
312
160
290
ECU
MEM
ALU
MUL
RCU
16
16
16
16
16
80
310