

(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。 Int. Cl. G06F 9/38 (2006.01)	(45) 공고일자 (11) 등록번호 (24) 등록일자	2006년04월18일 10-0572040 2006년04월11일
--	-------------------------------------	--

(21) 출원번호	10-2001-7001356	(65) 공개번호	10-2001-0053623
(22) 출원일자	2001년01월31일	(43) 공개일자	2001년06월25일
번역문 제출일자	2001년01월31일		
(86) 국제출원번호	PCT/US1999/001048	(87) 국제공개번호	WO 2000/07097
국제출원일자	1999년01월18일	국제공개일자	2000년02월10일

(81) 지정국 국내특허 : 일본, 대한민국,

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

(30) 우선권주장 09/127,094 1998년07월31일 미국(US)

(73) 특허권자 어드밴스드 마이크로 디바이시즈, 인코포레이티드
 미국 캘리포니아 94088-3453 서니베일 원 에이엠디 플레이스 메일 스톱68

(72) 발명자 위트데이비드비.
 미국텍사스78759오스틴패스파인더드라이브6318

(74) 대리인 박장원

심사관 : 성경아

(54) 명령들의 퇴거시 물리적인 레지스터들을 선택적으로 자유화하도록 구성된 프로세서

요약

레지스터 재명명 하드웨어를 포함하는 맵 유닛을 이용하는 프로세서가 개시된다. 맵 유닛은 명령 동작들을 스캔하여 내부 라인 의존성들을 검출함으로써 가상 레지스터 번호들을 발신지 레지스터들에 할당한다. 이후, 물리 레지스터 번호들은 가상 레지스터 번호들에 응답하여 발신지 레지스터 번호들에 맵핑된다. 맵 유닛은 (예를 들어, 맵 사일로 내에) 맵 유닛에 의해 처리되는 명령 동작들의 각 라인에 대응하는 현재의 예측 상태를 저장한다. 또한, 맵 유닛은 논리 레지스터들이 갱신되는 라인 갱신 논리 레지스터들 내의 명령 동작들의 표시 및 명령 동작들에 할당된 물리 레지스터 번호들을 저장한다. 라인을 갖는 명령 동작에 대한 예외 검출시, 라인에 대응하는 현재 예측 상태가 맵 사일로로부터 복구된다. 또한, 예외를 경험하는 명령 동작 이전의 라인 내의 명령 동작들에 대응하는 물리 레지스터 번호들이 현재 예측 상태로 복구된다. 맵 유닛은 조건 코드 결과 및 정수 결과를 저장하는 데에 동일한 물리 레지스터를 이용할 수 있다. 물리 레지스터를 식별하는 물리 레지스터 번호는 조건 코드 레지스터 및 정수 레지스터에 대해 기록된다. 맵 유닛은 하나 또는 그 이상의 명령 동작들의 퇴거시 구조 재명명 블록으로부터 이전의 재명명들을 꺼낸다. 꺼내어진 물리 레지스터 번호들은 갱신된 구조 상태에 대해 캡된다. 만일 캡 매치가 검출된다면, 꺼내어진 물리 레지스터는 자유화되지 않는다.

대표도

도 3

색인어

프로세서, 레지스터 재명명, 맵 유닛, 물리 레지스터

명세서

기술분야

본 발명은 프로세서들의 분야에 관한 것으로서, 특히 프로세서들의 레지스터 재명명(register renaming) 특성에 관한 것이다.

배경기술

EP-A-0 541 216호는 각각의 독립적인 명령 스트림에 대한 레지스터 재명명 유닛을 포함하는 데이터 처리 시스템을 개시한다. 이 레지스터 재명명 유닛은 재명명 동작을 수행하여, 레지스터 파일에서 처리 상태 레지스터들을 물리 레지스터들에 맵(mapping)시킨다. 물리 레지스터는 참조가 이루어진 모든 명령들이 성공적으로 종료되는 경우 자유화(free)된다.

US-A-5 805 918호는 동일한 레지스터의 서로 다른 부분들이, 서로 다른 명령 세트 아키텍처에 속하는 명령들에 의해 액세스되는 시스템을 개시한다.

US-A-5 546 554호는 레지스터 재명명을 위한 프로세서 내의 맵 유닛(mapping unit)을 개시한다. 각각의 물리 레지스터는 세 가지의 상태들, 즉 자유화(free), 경고(warning) 및 준비(ready) 상태 중 하나를 갖는다.

EP-A-0 463 628호는 레지스터 재명명을 위한 레지스터 맵핑을 개시한다.

수퍼스칼라 프로세서들은 클록 주기당 많은 명령들을 발생 및 실행시킴으로써, 그리고 설계에 부합하는 최고의 가능한 클록 주파수를 이용함으로써 고성능을 실현하고자 한다. 클록 주기당 명령들의 수를 증가시키기 위한 한 방법은 순서를 벗어난(out of order)실행이다. 순서를 벗어난 실행에서, 명령들은 프로그램 시퀀스(또는 "프로그램 순서")에서 지정된 것과는 다른 순서로 실행될 수 있다. 프로그램 시퀀스에서 서로 가까이 있는 임의의 명령들은 그들의 동시 실행을 저지하는 의존성을 가질 수 있는 반면, 프로그램 시퀀스 내의 후속 명령들은 이전의 명령들에 대한 의존성을 가지지 않을 수 있다. 따라서, 순서를 벗어난 실행은 (평균적으로) 동시에 실행되는 명령들의 수를 증가시킴으로써 수퍼스칼라 프로세서들의 성능을 향상시킬 수 있다.

불행히도, 순서를 벗어난 실행은 프로세서에 하드웨어를 추가시켜 프로세서를 더 복잡하게 한다. 예를 들어, 프로그램 순서에서 제 1 명령에 후속하는 제 2 명령은 제 1 명령에 의해 관독되는 저장 위치를 갱신할 수 있다. 다시 말해, 제 2 명령의 목적지 오퍼랜드가 제 1 명령의 발신지 오퍼랜드들중 하나가 될 수 있다. 정확한 프로그램 실행을 위해서는, 제 1 명령은 제 2 명령의 실행 전에 저장 위치에 저장된 값을 발신지 오퍼랜드로서 수신할 수 있어야만 한다. 마찬가지로, 제 1 및 2 명령들이 목적지 오퍼랜드로서 특정한 저장 위치를 갖는 경우, 제 2 명령의 결과는 제 1, 2 명령들을 실행한 후에, (그리고 저장 위치를 갱신하는 제 3 명령을 실행하기 전에) 저장 위치에 저장된 값이 되어야 한다.

일반적으로, 명령들은 하나 또는 그 이상의 발신지 오퍼랜드 및 하나 또는 그 이상의 목적지 오퍼랜드들을 가질 수 있다. 발신지 오퍼랜드들은 (목적지 오퍼랜드들인) 하나 또는 그 이상의 결과들을 생성시키도록 명령 정의에 따라 조종되는 입력 값들이다. 발신지 및 목적지 오퍼랜드들은 프로세서 외부의 메모리 위치에 저장된 메모리 오퍼랜드들이 될 수 있거나, 또는 프로세서 내에 포함된 레지스터 저장 위치들에 저장된 레지스터 오퍼랜드들이 될 수 있다. 프로세서에 의해 이용되는 명령 세트 구조(architecture)는, 구조 레지스터(architected register)들의 수를 규정한다. 이들 레지스터들은 명령 세트 구조에 의해 존재하는 것으로 규정되고, 명령들은 상기 구조 레지스터를 발신지 및 목적지 오퍼랜드들로서 이용하도록 코드화될 수 있다. 명령은 명령의 오퍼랜드 필드 내에서 레지스터 번호(또는 레지스터 어드레스)를 통해 특정 레지스터를 발신지 또는 목적지 오퍼랜드로서 지정한다. 레지스터 번호는 구조 레지스터들 중에서 선택된 레지스터를 유일하게(uniquely) 식별한다. 발신지 오퍼랜드는 발신지 레지스터 번호에 의해 식별되고, 목적지 오퍼랜드는 목적지 레지스터 번호에 의해 식별된다.

구조 레지스터들에 부가하여, 어떤 프로세서들은 명령을 실행하는 동안 일시적인 결과들을 보유하는 데에 이용될 수 있는 부가적인 마이크로구조(microarchitected) 레지스터들을 정의한다. 예를 들어, 어떠한 프로세서들은 가장 복잡한 명령들을 처리하는 데에 마이크로코딩(microcoding) 기술을 이용한다. 마이크로코드 루틴들은 복잡한 명령들에 응답하여 실행되며, 단순한 다수의 명령들을 포함한다. 마이크로코드 루틴들은 복잡한 명령을 실행하는 동안 일시적인 결과들을 발생시

킬 수 있다. 이들 마이크로구조 레지스터들(또는 잠정(temporary) 레지스터들)에는 구조 레지스터들로부터 잠정적 레지스터들을 유일하게 식별하기 위한 부가적인 레지스터 번호들이 할당된다. 본원에서 구조 레지스터들 및 잠정 레지스터들은 논리 레지스터들로 언급된다.

순서를 벗어난 실행을 이용하는 프로세서는 레지스터 오퍼랜드들에 대해 상기의 위험들을 경험할 수 있다. 이러한 위험들을 처리하는 방법이 레지스터 재명명이다. 레지스터 재명명에 있어서, 프로세서는 한 세트의 물리 레지스터들을 구비한다. 이 물리 레지스터들의 수는 프로세서의 명령 세트 아키텍처 및 마이크로아키텍처에 의해 지정된 논리 레지스터들의 수보다 많다. 명령들이 발행(issue)되면, 물리 레지스터들은 명령들의 목적지 레지스터 오퍼랜드들에 할당된다. 할당된 물리 레지스터를 식별하는 물리 레지스터의 번호가 각각의 목적지 오퍼랜드에 제공되고, 그리고 어떤 물리 레지스터들이 논리 레지스터들에 대응하는지의 표시(indication)가 프로세서에 의해 유지된다. 발신지 오퍼랜드들로서 논리 레지스터들을 갖는 후속 명령들에는 적절한 발신지 오퍼랜드를 읽어내기 위한 대응 물리 레지스터 번호가 제공된다. 각 명령의 목적지 오퍼랜드들을 저장하도록 서로 다른 물리 레지스터들을 할당함으로써, 명령들은 임의의 순서로 그들의 목적지 오퍼랜드들을 자유롭게 갱신할 수 있는 바, 이는 서로 다른 물리 저장 위치들이 갱신되기 때문이다.

불행히도, 목적지 오퍼랜드 명령들에 물리 레지스터 번호를 할당하고, 이들 물리 레지스터 번호를, 발신지 오퍼랜드들로서 목적지 오퍼랜드들을 갖는 후속의 명령들에 제공하는 과정이 복잡하고 느려질 수 있다. 슈퍼스칼라 프로세서들에 있어서 특히 어려운 것은, 명령들의 목적지 오퍼랜드들에 물리 레지스터 번호를 할당하고, 이들 명령들과 동시에 레지스터 재명명 하드웨어를 통과하는 후속 의존성 명령들에 상기 물리 레지스터 번호를 제공하는 것이다. 보다 높은 주파수에서 동작하면서 여전히 클럭 주기당 복수의 명령들의 재명명을 처리할 수 있는 레지스터 재명명 구조가 요망된다.

레지스터 재명명은 명령들이 예외 조건들을 경험할 때 어려움을 나타낸다. 여기에서 사용되는 예외란, 후속 명령들을 폐기(discard)하고 그리고 명령 페치(instruction fetch)를 다른 어드레스에서 시작할 것을 요구하는 명령들의 실행에 있어서의 에러를 말한다. 예를 들어, 분기 오예측은 예외 조건이다. 프로세서는 조건적 분기 명령들에 후속하는 명령들을 추론적으로 페치, 발행 및 실행하기 위하여 분기 예측을 수행할 수 있다. 예측이 부정확하면, 분기 명령 이후의 명령들은 폐기되며 분기 명령의 실행에 따라 명령들이 페치된다. 추가적인 예외 조건들로는, 메모리 오퍼랜드들의 어드레스에 대한 어드레스 변환 에러들 및 기타 다른 구조 또는 마이크로구조 에러 조건들이 포함될 수 있다.

레지스터 재명명은 예외로 인하여 후속적으로 폐기되는 명령들에 적용될 수 있기 때문에, 논리 레지스터들의 물리 레지스터들로의 맵핑은 예외를 경험하는 명령들에 부합되는 상태로 복구되어야 한다. 다시 말해, 논리 레지스터들의 물리 레지스터들로의 맵핑은 (프로그램 순서에서) 예외를 경험하는 명령 이전의 명령들의 실행을 반영해야만 하고, 예외를 경험하는 명령들에 후속하는 명령들의 실행을 반영하지 말아야 한다. 예외에 응답하여 폐치된 명령들이, 이들이 사용가능하게 되자마자 레지스터 재명명 하드웨어를 통과할 수 있도록, 레지스터 재명명 맵의 복구가 신속히 이루어지도록 하는 것이 바람직하다. 만일 새롭게 폐치된 명령들이 레지스터 재명명 하드웨어에 도달했을 때 레지스터 재명명의 맵의 복구가 여전히 행해지는 경우, 새롭게 폐치된 명령들은 레지스터 맵이 복구될 때 까지 중지되어야 한다. 이에 의해 프로세서의 성능이 저하된다.

또한, 레지스터 재명명 하드웨어는 일반적으로, 일단 대응 논리 레지스터가 후속명령들에 대응하는 값에 커미트(commit)된 경우, 특정 명령의 목적지 오퍼랜드에 이전에 할당된 물리 레지스터들을 재사용하기 위한 메커니즘을 포함한다. 물리 레지스터들을 가능한 효율적으로 이용하는 것이 바람직하며, 또한 후속 상태가 대응 논리 레지스터들에 커미트되었을 때 물리 레지스터들을 자유화 하기 위한 정확한 방법을 제공하는 것이 또한 바람직하다.

발명의 상세한 설명

상기 설명한 문제들은 레지스터 재명명 하드웨어를 포함하는 맵 유닛을 채용한 프로세서에 의해 대부분 해결된다. 맵 유닛의 다양한 실시예들은 레지스터 재명명 메커니즘의 성능 및 효율을 증대시키기 위한 다양한 기술을 활용한다.

본 발명의 일 실시예에 따르면, 맵 유닛은 인트라라인 종속성들(intraline dependencies)을 검출하기 위해 명령 동작들을 스캔(scan)함으로써 가상 레지스터 번호들을 발신지 레지스터들에 할당하도록 구성된다. 만일 종속성이 검출되지 않는 경우, 현재의 예측(lookahead) 상태를 표시하는 가상 레지스터 번호가 할당된다. 종속성이 검출되는 경우, 발행위치를 표시하는 가상 레지스터 번호가 할당되는 바, 이 발행위치에 발신지 레지스터가 의존하게 된다. 이어서, 물리 레지스터 번호들이 상기 가상 레지스터 번호들에 응답하여 상기 발신지 레지스터 번호들에 맵핑된다. 일 실시예에서, 가상 레지스터 번호들의 할당 및 가상 레지스터 번호들의 물리 레지스터 번호들에의 맵핑은 개별 파이프라인 스테이지에서 수행된다. 그러므로써, 레지스터 재명명 메커니즘은 보다 높은 클럭 주파수에서 동작할 수 있게 된다. 이 프로세서의 성능은 보다 높은 클럭 주파수들을 달성할 수 있을 정도로 증대될 수 있다.

다른 실시예에 따르면, 맵 유닛은 이 맵 유닛에 의해 처리되는 명령 동작들의 각 라인에 대응하는 현재의 예측 상태를 (예컨대, 맵 사일로(map silo)에 저장한다. 현재의 예측 상태는 명령 동작들의 라인에 관하여 레지스터 재명명을 수행하기에 앞서 각 논리 레지스터에 할당되는 물리 레지스터 번호들을 식별한다. 또한, 맵 유닛은 라인 내의 어떤 명령 동작들이 논리 레지스터를 갱신하는지 및 어떤 논리 레지스터들이 갱신되는지 및 명령 동작들에 할당된 물리 레지스터의 수에 관한 표시를 저장한다. 어느 한 라인과의 명령 동작에 대한 예외조건을 검출하에서, 상기 라인에 대응하는 현재의 예측 상태는 맵 사일로로부터 복원(restore)된다. 또한, 상기 예외를 결정하는 명령 동작들 이전의 라인내의 명령 동작들에 대응하는 물리 레지스터 번호들이 현재의 예측 상태내로 복원된다. 유익하게, 현재의 예측 상태는 예외 검출시 신속히 복구될 수 있다. 이 예외에 응답하여 폐지된 명령 동작들은 맵 유닛에 도달하는 즉시 재명명될 수 있는 바, 현재의 예측 상태의 복구가 이미 완료되기 때문이다. 그러므로, 프로세서의 성능이 증대된다.

또 다른 실시예에서, 맵 유닛은 조건 코드 결과 및 정수 결과를 저장하는 데에 동일한 물리 레지스터를 이용함으로써 물리 레지스터의 효율을 개선한다. 물리 레지스터를 식별하는 물리 레지스터 번호가 조건 코드 레지스터 및 정수 레지스터에 대해 기록된다. 재명명으로서 재사용될 물리 레지스터들의 정확한 자유화를 제공하기 위하여, 맵 유닛은 하나 또는 그 이상의 명령 동작들의 퇴거에 응답하여 커미트되는 논리 레지스터들의 세트에 대응하는 엔트리들을 갱신하기 이전에 구조재명명 블록으로부터 이전의 재명명들을 팝(pop)한다. 팝된 물리 레지스터 번호들을 자유화하기 전에, 상기 팝된 물리 레지스터들의 번호들은 구조 재명명 블록에 의해 유지되는 갱신된 구조 상태에 대해 캠(cam)된다. 만일 캠 매치(cam match)가 검출된다면, 팝된 물리 레지스터는 자유화되지 않는다. 유익하게, 물리 레지스터들이 보다 효율적으로 이용될 수 있으며, 또한 물리 레지스터의 보다 정확한 자유화가 성취될 수 있다. 조건 코드 레지스터 및 정수 레지스터 재명명에 관하여 물리 레지스터들을 보다 효율적으로 이용하는 것에 부가하여, 레지스터들의 자유 사용을 위하여 본 메커니즘을 이용하는 다른 최적화가 가능할 수 있다. 예를 들어, 레지스터 레지스터간 이동이 발신지 물리 레지스터 번호를 목적지 물리 레지스터 번호로서 복사함으로써 이루어질 수 있다. 본 메커니즘은 상기의 경우에서도 역시 물리 레지스터 번호의 부주의한 자유화를 막을 수 있다.

삭제

삭제

삭제

프로세서 내의 물리 레지스터들을 다루는 방법이 고려된다. 제 1 물리 레지스터는 명령 동작의 목적지 오퍼랜드에 할당된다. 목적지 오퍼랜드는 논리 레지스터를 식별한다. 이 명령 동작은 퇴거된다. 명령 동작의 퇴거에 응답하여, 구조 재명명 블록은 제 1 물리 레지스터에 대응하는 제 1 물리 레지스터 번호로 갱신된다. 또한, 명령 동작의 퇴거에 응답하여, 제 2 물리 레지스터 번호가 구조 재명명 블록으로부터 읽혀진다. 제 2 물리 레지스터 번호는 제 1 물리 레지스터 번호로 인해, 구조 재명명 블록 내의 논리 레지스터를 나타내던 것으로부터 변위(displacement)된다(즉, 상기 구조 재명명 블록내의 논리 레지스터를 나타내지 않게 된다). 제 2 물리 레지스터 번호의 자유화는, 제 2 물리 레지스터 번호가 구조 재명명 블록 내의 제 2 논리 레지스터를 나타낸다는 결정에 응답하여 금지된다.

맵 유닛과 구조 재명명 블록을 포함하는 프로세서가 고려된다. 맵 유닛은 제 1 물리 레지스터를 명령 동작의 목적지 오퍼랜드에 할당하도록 구성된다. 또한, 맵 유닛은 물리 레지스터들중 어느 것이 할당에 대해 자유화되는지를 표시하는 자유화 목록을 유지하도록 구성된다. 맵 유닛은 이 자유화 유닛으로부터 제 1 물리 레지스터를 선택하도록 구성된다. 명령 동작이 퇴거되고 있다는 표시 및 제 1 물리 레지스터에 대응하는 제 1 물리 레지스터 번호를 수신하도록 결합된 구조 재명명 블록은 명령 동작이 퇴거되고 있다는 표시에 응답하여 목적지 오퍼랜드에 의해 식별된 제 1 논리 레지스터에 이전에 대응하던 제 2 물리 레지스터를 식별하는 제 2 물리 레지스터 번호를 제공하도록 구성된다. 구조 재명명 블록은 제 1 물리 레지스터 번호를 포착하여, 이 제 1 물리 레지스터 번호를 명령 동작이 퇴거되고 있다는 표시에 응답하여 제 1 논리 레지스터에 대응하는 것으로서 보유하도록 구성된다. 또한, 구조 재명명 블록은 제 2 물리 레지스터의 번호가 또한 제 2 논리 레지스터에 대응하는지를 결정하도록 구성된다. 맵 유닛은 제 2 물리 레지스터가 제 2 논리 레지스터를 나타내는 경우, 제 2 물리 레지스터의 번호를 자유화 목록에 부가하는 것을 금지하도록 구성된다.

프로세서 내의 물리 레지스터를 다루는 방법이 고려된다. 제 1 물리 레지스터는 명령 동작의 목적지 오퍼랜드에 할당된다. 목적지 오퍼랜드는 논리 레지스터를 식별한다. 명령 동작은 퇴거된다. 명령 동작에 응답하여, 구조 재명명 블록은 제 1 물리 레지스터에 대응하는 제 1 물리 레지스터 번호로 갱신된다. 또한, 명령 동작에 응답하여, 제 2 물리 레지스터 번호가 구조 재명명 블록으로부터 읽혀진다. 제 2 물리 레지스터 번호는 제 1 물리 레지스터 번호로 인해 구조 재명명 블록 내의 논리 레지스터를 나타내던 것으로부터 변위된다. 제 2 물리 레지스터는 자유화된다.

프로세서가 고려된다. 이 프로세서는 맵 유닛 및 구조 재명명 블록을 포함한다. 이 맵 유닛은 제 1 물리 레지스터를 명령 동작의 목적지 오퍼랜드에 할당하도록 구성되고, 또한 물리 레지스터들중 어느 것이 할당에 대해 자유화되는지를 표시하는 자유화 목록을 보유하도록 구성된다. 맵 유닛은 자유화 목록으로부터 제 1 물리 레지스터를 선택하도록 구성된다. 명령 동작이 되거되고 있다는 표시 및 제 1 물리 레지스터에 대응하는 제 1 물리 레지스터 번호를 수신하도록 결합된 구조 재명명 블록은, 명령 동작이 되거되고 있다는 표시에 응답하여 목적지 오퍼랜드에 의해 식별된 제 1 논리 레지스터에 이전에 대응하던 제 2 물리 레지스터를 식별하는 제 2 물리 레지스터 번호를 제공하도록 구성된다. 또한, 구조 재명명 블록은 제 1 물리 레지스터 번호를 포착한 다음, 이 제 1 물리 레지스터 번호를 명령 동작이 되거되고 있다는 표시에 응답하여 제 1 논리 레지스터에 대응하는 것으로서 보유하도록 구성된다. 상기 맵 유닛은 제 2 물리 레지스터 번호를 자유화 목록에 추가하도록 구성된다.

프로세서 및 I/O 장치를 포함하는 프로세서가 고려된다. 이 프로세서는 맵 유닛 및 구조 재명명 블록을 포함한다. 맵 유닛은 제 1 물리 레지스터를 명령 동작의 목적지 오퍼랜드에 할당하도록 구성되고, 또한 물리 레지스터들중 어느 것이 할당에 대해 자유화되는지를 표시하는 자유화 목록을 보유하도록 구성된다. 맵 유닛은 자유화 목록으로부터 제 1 물리 레지스터를 선택하도록 구성된다. 명령 동작이 되거되고 있다는 표시 및 제 1 물리 레지스터에 대응하는 제 1 물리 레지스터 번호를 수신하도록 결합된 구조 재명명 블록은, 명령 동작이 되거되고 있다는 표시에 응답하여 목적지 오퍼랜드에 의해 식별된 제 1 논리 레지스터에 이전에 대응하던 제 2 물리 레지스터를 식별하는 제 2 물리 레지스터 번호를 제공하도록 구성된다. 또한, 구조 재명명 블록은 제 1 물리 레지스터 번호를 포획하여, 이 제 1 물리 레지스터 번호를 명령 동작이 되거되고 있다는 표시에 응답하여 제 1 논리 레지스터에 대응하는 것으로서 보유하도록 구성된다. 맵 유닛은 제 2 물리 레지스터 번호를 자유화 목록에 추가하도록 구성된다. 프로세서에 결합된 I/O 장치는, 컴퓨터 시스템과 I/O 장치에 결합된 다른 컴퓨터 시스템과의 사이에서 통신을 행하도록 구성된다.

본 발명의 다른 목적들 및 장점들은 첨부 도면을 참조하여 설명되는 하기의 상세한 설명으로부터 명확해질 것이다.

도면의 간단한 설명

도 1은 프로세서의 일 실시예의 블록도이다.

도 2는 도 1에 도시된 맵 유닛, 맵 사일로, 및 구조 재명명 블록의 블록선도로서, 도 1에 도시된 프로세서의 일 실시예에 따라 이들 간의 상호연결을 나타낸 도면이다.

도 3은 도 1, 2에 도시된 맵 유닛의 일 실시예의 블록도이다.

도 4는 도 3에 도시된 레지스터 스캔 유닛의 일 실시예의 블록도이다.

도 5는 가상 레지스터 번호들에 대해 이용될 수 있는 하나의 엔코딩을 나타낸 표이다.

도 6는 도 4에 도시한 스캔 유닛의 일 실시예의 일부에 대한 블록도이다.

도 7은 도 4에 도시된 스캔 유닛의 일 실시예의 다른 부분에 대한 블록도이다.

도 8은 도 3에 도시된 가상/물리 레지스터 맵 유닛의 일 실시예의 블록도이다.

도 9는 도 1, 2에 도시된 맵 사일로의 일 실시예에서 저장된 정보를 나타낸 표이다.

도 10은 논리 레지스터에 대한 예시적인 예측 레지스터 상태를 나타내는 블록도이다.

도 11은 도 1, 2 및 3에 도시된 맵 유닛의 일 실시예에 따른 예측 상태를 복원하는 것을 나타낸 흐름도이다.

도 12는 도 1에 도시된 프로세서를 포함하는 컴퓨터 시스템의 일 실시예의 블록도이다.

본 발명은 다양한 변경들 및 변형적인 형태들을 가질 수 있기는 하지만, 본원의 도면 및 상세한 설명에서는 단지 본 발명의 특정 실시예들에 대해 설명할 것이다. 그러나, 도면들 및 상세한 설명은 본 발명을 이러한 특정 실시예들에 한정하지 않으며, 본 발명은 첨부된 청구항들에 의해 규정되는 본 발명의 정신 및 범위 내에서 모든 변경들, 등가물들 및 변형들을 포함한다.

실시예

이제 도 1을 참조하면, 프로세서(10)의 일 실시예의 블록도가 도시된다. 다른 실시예들이 또한 가능하며 고려된다. 도 1의 실시예에서, 프로세서(10)는 라인 예측기(12), 명령 캐시(I-캐시)(14), 정렬 유닛(16), 분기 히스토리 테이블(18), 간접 어드레스 캐시(20), 복귀 스택(22), 디코드(또는 복호화) 유닛(24), 예측기 미스 디코드 유닛(26), 마이크로코드 유닛(26), 맵 유닛(30), 맵 사일로(32), 구조 재명명 블록(34), 한 쌍의 명령큐들(36A-36B), 한 쌍의 레지스터 파일들(38A-38B), 한 쌍의 실행 코어들(40A-40B), 적재(또는 로드)/저장 유닛(42), 데이터 캐시(D-캐시)(44), 외부 인터페이스 유닛(46), PC 사일로 및 리디렉트 유닛(48) 및 명령 TLB(ITB)(50)를 포함한다. 라인 예측기(12)는 ITB(50), 예측기 미스 디코드 유닛(26), 분기 히스토리 테이블(18), 간접 어드레스 캐시(20), 복귀 스택(22), PC(Program Counter) 사일로 및 리디렉트 블록(48), 정렬 유닛(16) 및 I-캐시(14)에 연결된다. I-캐시(14)는 정렬 유닛(16)에 연결된다. 정렬 유닛(16)은 또한 예측기 미스 디코드 유닛(26) 및 디코드 유닛(24)에 연결된다. 디코드 유닛(24)은 또한 마이크로코드 유닛(28) 및 맵 유닛(30)에 연결된다. 맵 유닛(30)은 맵 사일로(32), 구조 재명명 블록(34), 명령큐들(36A-36B), 적재/저장 유닛(42), 실행 코어들(40A-40B), 및 PC 사일로 및 리디렉트 블록(48)에 연결된다. 명령큐들(36A-36B)은 서로 연결되며, 각각의 실행 코어들(40A-40B) 및 레지스터 파일들(38A-38B)에 연결된다. 레지스터 파일들(38A-38B)은 서로 연결되며, 각각의 실행 코어들(40A-40B)에 연결된다. 실행 코어들(40A-40B)은 또한 적재/저장 유닛(42), 데이터 캐시(44) 및 PC 사일로 및 리디렉트 유닛(48)에 연결된다. 적재/저장 유닛(42)은 PC 사일로 및 리디렉트 유닛(48), D-캐시(44), 및 외부 인터페이스 유닛(46)에 연결된다. D-캐시(44)는 레지스터 파일들(38)에 연결되고, 외부 인터페이스 유닛(46)은 외부 인터페이스(52)에 연결된다. 본원에서 문자가 뒤에 병기된 참조 번호로 언급되는 요소들은 총체적으로 참조 번호만으로 언급될 것이다. 예를 들어, 명령큐들(36A-36B)은 총체적으로 명령큐들(36)로서 언급된다.

도 1의 실시예에서, 프로세서(10)는 가변 바이트 길이의, 복합 명령 세트 계산(CISC) 명령 세트 구조를 이용한다. 예를 들어, 프로세서(10)는 x86 명령 세트 구조(IA-32로도 언급됨)를 이용할 수 있다. 다른 실시예들은 고정된 길이의 명령 세트 구조들 및 간략 명령 세트 계산(RISC) 명령 세트 구조들을 포함하는 다른 명령 세트 구조를 이용할 수 있다. 도 1에 도시된 특정한 것들은 상기의 구조들에서 생략될 수 있다.

라인 예측기(12)는 I-캐시(14)에 대한 페치 어드레스들을 발생시키도록 구성되며, 부가적으로 명령 동작들의 라인에 관한 정보를 정렬 유닛(16)에 제공하도록 구성된다. 일반적으로, 라인 예측기(12)는 프로세서(10)에 의해 이전에 추론적으로 페치된 명령 동작들의 라인들, 및 라인의 페치시 선택될 각 라인에 대응하는 하나 또는 그 이상의 다음 페치 어드레스들을 저장한다. 일 실시예에서, 라인 예측기(12)는 각각 명령 동작들의 한 라인을 규정하는 1K 엔트리들을 저장하도록 구성된다. 라인 예측기(12)는 예컨대 256 엔트리의 4개의 뱅크로 뱅킹(banking)되어, 바람직한 경우, 각각 듀얼 포팅(dual porting) 없이 동시 읽기 및 갱신하는 것이 가능하다.

라인 예측기(12)는 대응 명령 바이트들을 페치하기 위하여 I-캐시(14)에 다음 페치 어드레스를 제공한다. I-캐시(14)는 명령 바이트들을 저장하는 고속의 캐시 메모리이다. 일 실시예에 따르면, I-캐시(14)는, 예를 들어 64개의 바이트 캐시 라인들을 이용하는 256 Kbyte의 4웨이 세트 어소시에이티브 편성(4-way set associative organization)을 포함할 수 있다. 그러나, 그 어떤 I-캐시 구조도 적절할 수 있다. 또한, 다음 페치 어드레스는 명령 동작들의 대응 라인에 대한 정보를 페치할 수 있도록, 라인 예측기(12)의 입력에 다시 제공된다. 다음 페치 어드레스는 PC 사일로 및 리디렉트 유닛(48)에 보고되는 예외 조건들에 응답하여 ITB(50)에 의해 제공되는 어드레스에 의해 무효(override)로 될 수 있다.

라인 예측기에 의해 제공되는 다음 페치 어드레스는 (라인이 비분기 명령으로 종료(terminate)되는 경우) 라인 내의 마지막 명령에 후속하는 어드레스가 될 수 있다. 변형예로서, 다음 페치 어드레스는 라인을 종료시키는 분기 명령의 목표 어드레스가 될 수 있다. 또 다른 변형예에서, 라인은 복귀 명령에 의해 종료될 수 있는데, 이 경우 다음 페치 어드레스는 복귀 스택(22)으로부터 나온다.

페치 어드레스에 응답하여, 라인 예측기(12)는 페치 어드레스에서 시작하는 명령 동작들의 라인에 관한 정보를 정렬 유닛(16)에 제공한다. 정렬 유닛(16)은 I-캐시(14)로부터 페치 어드레스에 대응하는 명령 바이트들을 수신하고, 제공된 명령 동작 정보에 따라 한 세트의 발행 위치들로의 명령 바이트들을 선택한다. 보다 명확하게 설명하면, 라인 예측기(12)는 명령 동작들의 라인 내의 각 명령에 대한 시프트량을 제공하고, 라인을 구성하는 명령 동작들의 세트에 명령들을 맵핑한다.

명령은 다중 명령 동작들에 대응될 수 있으며, 이에 따라 그 명령에 대응하는 시프트량은 다중 발행 위치들로의 명령 바이트들을 선택하는 데에 이용될 수 있다. 발행 위치가 라인 내의 각각의 가능한 명령 동작에 대해 제공된다. 일 실시예에서, 명령 동작들의 라인은 6개까지의 명령들에 대응하는 8개까지의 명령 동작들을 포함할 수 있다. 일반적으로, 본원에서 이용되는 바와같이, 명령 동작들의 라인은 디코드 유닛(24)에 동시에 발행되는 한 그룹의 명령 동작들을 말한다. 명령 동작들의 라인은 마이크로프로세서(10)의 파이프라인을 통해 명령큐들(36)로 유닛으로서 진행된다. 명령큐들(36)에 저장될 때, 개별 명령 동작들은 그 어떠한 순서로도 실행될 수 있다.

디코드 유닛(24) (및 명령큐들(36)까지의 후속 파이프라인 스테이지들) 내의 발행 위치들 이들 파이프라인 스테이지들 내의 하드웨어에 대한 라인 내의 명령 동작들의 프로그램 순서를 정의한다. 정렬 유닛(16)에 의해 발행 위치에 정렬되는 명령 동작은 명령큐들(36A-36B) 내에 저장될 때 까지 그 발행 위치 내에 머무른다. 따라서, 제 1 발행 위치는, 제 1 발행 위치 내의 명령 동작이 프로그램 순서에서 동시에 제 2 발행 위치 내에 있는 명령 동작 이전인 경우, 제 2 발행 위치의 앞에 있는 것으로 설명될 수 있다. 마찬가지로, 제 1 발행 위치는, 제 1 발행 위치 내의 명령 동작이 프로그램 순서에서 동시에 제 2 발행 위치 내에 있는 명령 동작의 뒤에있는 경우, 제 2 발행 위치의 뒤에 있는 것으로 설명될 수 있다. 발행 위치들 내의 명령 동작들은 또한 라인 내의 다른 명령 동작들의 이전 또는 뒤에 있는 것으로서 설명될 수 있다.

본원에서 이용되는 바와 같이, 명령 동작(또는 ROP)은 실행 코어들(40A-40B) 내의 실행 유닛이 단일 객체(entity)로서 실행하도록 구성되는 동작이다. 단순한 명령들은 단일 명령 동작에 대응할 수 있으며, 좀 더 복잡한 명령들은 다중 명령 동작들에 대응할 수 있다. 보다 복잡한 명령들 중 일부는 마이크로코드 루틴들로서 마이크로코드 유닛(28) 내에서 실행될 수 있다. 또한, 비-CISC 명령 세트들을 이용하는 실시예들에서 각 명령에 대해 단일 명령 동작을 이용할 수 있다(즉, 명령 및 명령 동작은 상기 실시예들에서 같은 뜻일 수 있다). 일 특정 실시예에서, 라인은 6개까지의 명령들에 대응하는 8개까지의 명령 동작들을 포함할 수 있다. 또한, 특정 실시예에서 분기 명령이 검출되는 경우 6개 명령들 및/또는 8개 명령 동작들 이하에서 라인을 종료할 수 있다. 바람직한 경우에, 라인에 대한 명령 동작들에 대해 부가적인 한정들이 이용될 수 있다.

라인 예측기(12)에 의해 발생된 다음 페치 어드레스는 분기 히스토리 테이블(18), 간접 어드레스 캐시(20), 및 복귀 스택(22)으로 라우팅(routing)된다. 분기 히스토리 테이블(18)은 다음 페치 어드레스에 의해 식별된 라인을 종료시킬 수 있는 조건 분기 명령에 대한 분기 히스토리를 제공한다. 라인 예측기(12)는 라인을 종료시키는 조건 분기 명령이 테이큰(taken)으로서 또는 닛테이큰(not taken)으로서 예측되어야 하는 지를 결정하기 위하여, 분기 히스토리 테이블(18)에 의해 제공된 예측을 이용할 수 있다. 일 실시예에서, 라인 예측기(12)는 테이큰 또는 닛테이큰을 선택하는 데에 이용될 분기 예측을 저장할 수 있으며, 그리고 분기 히스토리 테이블(18)은 라인 예측기 예측을 취소하고 다른 다음 페치 어드레스가 선택되게 하는 보다 정확한 예측을 제공하는 데에 이용된다. 간접 어드레스 캐시(20)는 빈번하게 변하는 간접 분기 타겟 어드레스들을 예측하는 데에 이용된다. 라인 예측기(12)는 이전에 발생된 간접 타겟 어드레스를 다음 페치 어드레스로서 저장할 수 있다. 간접 어드레스 캐시(20)는, 대응 라인이 간접 분기 명령에 의해 종료되는 경우, 라인 예측기(12)에 의해 제공되는 다음 페치 어드레스를 무효화할 수 있다. 또한, 명령 동작들의 라인 내의 마지막 명령에 후속하는 어드레스는 라인이 서브루틴 호출 명령에 의해 종료되는 경우 복귀 스택(22)에 푸쉬(push)될 수 있다. 복귀 스택(22)은 그의 맨 위에 저장되는 어드레스를 복귀 명령에 의해 종료된 라인들에 대한 잠정적인 다음 페치 어드레스로서 라인 예측기(12)에 제공한다.

상기 언급한 블록들에 다음 페치 어드레스 및 명령 동작 정보를 제공하는 것에 부가하여, 라인 예측기(12)는 다음 페치 어드레스 및 명령 동작 정보를 PC 사일로 및 리디렉트 유닛(48)에 제공하도록 구성된다. PC 사일로 및 리디렉트 유닛(48)은 페치 어드레스 및 라인 정보를 저장하며, 명령들의 올바른 퇴거 뿐 아니라 예외시 명령 폐칭을 리디렉트하는 역할을 한다. PC 사일로 및 리디렉트 유닛(48)은 페치 어드레스, 및 프로세서(10) 내에서 아직 처리되지 않은 명령 동작들의 다중 라인들에 대응하는 명령 동작 정보를 저장하는 순환 버퍼를 포함할 수 있다. 명령들의 라인의 퇴거시, PC 사일로 및 리디렉트 유닛(48)은 각각 조건 분기 및 간접 분기의 실행에 따라서 분기 히스토리 테이블(18) 및 간접 어드레스 캐시(20)를 갱신할 수 있다. 예외 처리시, PC 사일로 및 리디렉트 유닛(48)은 예외-야기 명령에 후속하는 복귀 스택(22)으로부터의 엔트리들을 추방(purge)할 수 있다. 또한, PC 사일로 및 리디렉트 유닛(48)은 예외를 야기하는 명령의 표시를 맵 유닛(30), 명령큐들(36), 및 적재/저장 유닛(42)으로 라우팅시켜, 이들 유닛들이 예외를 야기하는 명령에 후속하는 명령들을 취소하고 이에 따라 추론적인 상태를 복구할 수 있게 한다.

일 실시예에서, PC 사일로 및 리디렉트 유닛(48)은 시퀀스 번호(R#)를 각각의 명령 동작에 할당하여 프로세서(10) 내에서 아직 처리되지 않은 명령 동작들의 순서를 식별한다. PC 사일로 및 리디렉트 유닛(48)은 라인을 갖는 각각의 가능한 명령 동작에 R#를 할당할 수 있다. 만일 라인이 명령 동작들을 그 최대 수 보다 훨씬 적게 포함한다면, 할당된 R#의 일부는 그 라인에 대해 이용되지 않을 것이다. 그러나, PC 사일로 및 리디렉트 유닛(48)은 다음 R#의 세트를 명령 동작들의 다음 라인에 할당하도록 구성될 수 있으며, 이에 따라 할당되었지만 사용되지 않은 R#는 명령 동작의 대응 라인이 퇴거될 때 까지

사용되지 않은 채로 남아있게 된다. 이러한 방식으로, 주어진 라인에 할당된 R#의 일부는 프로세서(10) 내의 라인을 식별하는 데에 이용될 수 있다. 일 실시예에서, 최대 8개의 ROP가 라인에 할당될 수 있다. 따라서, 각 라인 내의 첫 번째 ROP에는 8의 배수인 R#가 할당될 수 있다. 이에 따라, 사용되지 않은 R#들은 자동으로 스킵(skip)된다.

상기에서는 라인 예측기(12)가 다음 어드레스들을 예측하고 명령 동작들의 라인에 대한 명령 동작 정보를 제공하는 것에 대해 설명하였다. 이러한 동작은 각 페치 어드레스가 라인 예측기(12) 내에서 히트(hit)하는 한 일어난다. 라인 예측기(12) 내에서 미스(miss)를 검출하게 되면, 정렬 유닛(16)은 I-캐시(14)로부터의 대응 명령 바이트들을 예측기 미스 디코드 유닛(26)으로 라우팅시킨다. 예측기 미스 디코드 유닛(26)은 미스된 페치 어드레스에 의해 지정된 오프셋에서 시작되는 명령들을 디코드하고 명령 동작의 라인 정보 및 다음 페치 어드레스를 발생시킨다. 예측기 미스 디코드 유닛(26)은 프로세서(10) 설계시 명령 동작들의 라인에 어떠한 제한(예를 들어, 명령 동작들의 최대수 및 명령들의 최대수, 분기 명령에서의 종료, 등)을 가한다. 라인의 디코드를 완료하게 되면, 예측기 미스 디코드 유닛(26)은 상기 정보를 저장할 위하여 라인 예측기(12)에 제공한다. 예측기 미스 디코드 유닛(26)은 명령들이 디코드될때, 이 명령들을 디스패치하도록 구성될 수 있다. 변형예로서, 예측기 미스 디코드 유닛(26)은 명령 라인 정보를 디코드하여, 이를 저장할 위하여 라인 예측기(12)에 제공할 수 있다. 이후, 미스된 페치 어드레스는 라인 예측기(12)에서 재시도될 수 있으며, 히트가 검출될 수 있다. 또한, 라인 예측기(12)에서 히트가 검출될 수 있으며, I-캐시(14)에서 미스가 발생할 수 있다. 대응 명령 바이트들은 외부 인터페이스 유닛(46)을 통해 페치되어 I-캐시(14)에 저장될 수 있다.

일 실시예에서, 라인 예측기(12) 및 I-캐시(14)는 물리적 어드레스 지정을 이용한다. 그러나, 예외 검출시, PC 사일로 및 리디렉트 유닛(48)에는 논리(또는 가상)어드레스가 제공될 것이다. 따라서, 리디렉트 어드레스들은 라인 예측기(12)로의 제시(presentation)를 위하여 ITB(50)에 의해 변환된다. 또한, PC 사일로 및 리디렉트 유닛(48)은, 상대 분기 타겟 어드레스들과 같은 PC 상대 계산들에서 이용되는 가상의 예측 PC 값을 유지한다. 각 라인에 대응하는 가상의 예측 PC는, 대응 물리 어드레스가 라인 예측기(12)에 의해 발생된 물리 페치 어드레스와 매치하는 지를 식별하기 위하여 ITB(50)에 의해 변환된다. 만일 미스 매치가 발생한다면, 라인 예측기(12)는 정확한 물리 어드레스에 의해 갱신되며, 정확한 명령들이 페치된다. PC 사일로 및 리디렉트 유닛(48)은 또한 보호 경계들을 넘는 페칭 등에 관련된 예외들을 처리한다. PC 사일로 및 리디렉트 유닛(48)은 또한 가장 최근에 퇴거된 명령들의 어드레스를 표시하는 퇴거 PC 값을 유지한다.

상기 설명한 바와 같이, 디코드 유닛(24)은 다수의 발행 위치들에서 정렬 유닛(16)으로부터 명령 동작들을 수신하도록 구성된다. 디코드 유닛(24)은 (명령 바이트들에 대응하는 명령 동작들중 어느 것이 특정 발행 위치에서 발생될 것이라는 표시와 함께) 각 발행 위치에 평행하게 정렬된 명령 바이트들을 디코드한다. 디코드 유닛(24)은 각 명령 동작에 대한 발신지 및 목적지 오퍼랜드들을 식별하고, 실행 코어들(40A-40B)에 의해 이용되는 명령 동작 엔코딩을 발생시킨다. 디코드 유닛(24)은 또한 마이크로코드로 실행되는 명령들에 대해 마이크로코드 유닛(28)으로부터 마이크로코드 루틴들을 페치하도록 구성된다.

일 특정 실시예에 따르면, 다음과 같은 명령 동작들 즉, 정수, (멀티미디어를 포함하는) 부동 소수점 덧셈, (멀티미디어를 포함하는) 부동 소수점 곱셈, 분기, 적재, 저장 어드레스 발생 및 저장 데이터가 프로세서(10)에 의해 지원된다. 각 명령 동작은 2개까지의 발신지 레지스터 오퍼랜드들 및 한 개의 목적지 레지스터 오퍼랜드를 이용할 수 있다. 일 특정 실시예에 따르면, 단일 목적지 레지스터 오퍼랜드가 정수 ROP들에 할당되어, 정수 결과 및 조건 코드(또는 플래그들) 갱신을 저장할 수 있다. 대응하는 논리 레지스터들은 정수 연산의 퇴거지 대응하는 PR#을 수신할 것이다. 특정한 명령(예를 들어, ESP 및 지정된 목적지 레지스터를 갱신하는 명령)들은 두 개의 목적지 레지스터들을 갱신하기 위하여 동일한 타입의 두 개의 명령 동작들을 발생시킬 것이다.

디코드된 명령 동작들과 발신지 및 목적지 레지스터 번호들이 맵 유닛(30)에 제공된다. 맵 유닛(30)은 물리 레지스터 번호들(PR#들)을 각 명령 동작의 각각의 목적지 레지스터 오퍼랜드 및 발신지 레지스터 오퍼랜드에 할당함으로써 레지스터 재명명을 수행하도록 구성된다. 물리 레지스터 번호들은 레지스터 파일들(38A-38B) 내의 레지스터들을 식별한다. 또한, 맵 유닛(30)은 큐 번호(IQ#)를 각 명령 동작에 할당하여, 명령 동작을 저장하기 위해 할당된 명령큐들(36A-36B)내의 위치를 식별한다. 맵 유닛(30)은 또한, 명령 동작의 발신지 오퍼랜드에 할당된 각각의 물리 레지스터 번호를 갱신하는 명령들의 큐 번호들을 제공함으로써 각 명령 동작에 대한 의존성들의 표시를 제공한다. 맵 유닛(30)은 물리 레지스터 번호들과 각 명령 동작에 할당된 명령 큐 번호들(및 대응 논리 레지스터 번호들)로 맵 사일로(32)를 갱신한다. 또한, 맵 사일로(32)는, 명령들의 라인 및 PC 사일로에 대한 명령들의 라인을 식별하는 R# 전에, 논리 레지스터들에 대응하는 예측 상태를 저장하도록 구성될 수 있다. 상기 설명된 PC 사일로와 유사하게, 맵 사일로(32)는 엔트리들의 원형 버퍼를 포함할 수 있다. 각 엔트리는 명령 동작들중 한 라인에 대응하는 정보를 저장하도록 구성될 수 있다.

맵 유닛(30) 및 맵 사일로(32)는 또한 PC 사일로(48)로부터 퇴거 표시를 수신하도록 구성될 수 있다. 명령 동작들의 라인 퇴거시, 맵 사일로(32)는 라인에 할당된 목적지 물리 레지스터 번호들 및 대응하는 논리 레지스터 번호들을 저장할 위하여

구조 재명명 블록들(34)에 전달한다. 구조 재명명 블록(34)은, 각 논리 레지스터에 대해 전송된 레지스터 상태를 나타내는, 각 논리 레지스터에 대응하는 물리 레지스터 번호를 저장한다. 새로운 물리 레지스터 번호로 대응하는 논리 레지스터를 갱신할 때, 구조 재명명 블록(34)으로부터 교체된 물리 레지스터 번호들은 이후 명령들로의 할당을 위하여 물리 레지스터 번호들의 자유화 목록로 복귀된다. 일 실시예에서, 물리 레지스터 번호를 자유화 목록로 복귀시키기 전에, 물리 레지스터 번호들은 구조 재명명 블록(34) 내에 남아있는 물리 레지스터 번호들과 비교된다. 만일 물리 레지스터 번호가 교체된 후에도 구조 재명명 블록(34) 내에 여전히 표현된다면, 물리 레지스터 번호는 자유화 목록에 추가되지 않는다. 이러한 실시예들은 동일한 물리 레지스터 번호가 하나 이상의 명령 결과를 저장하는 데에 이용되는 경우에 이용될 수 있다. 예를 들어, x86 명령 세트 구조를 이용하는 실시예는 부동 소수점 오퍼랜드들을 저장하기에 충분히 큰 물리 레지스터들을 제공할 수 있다. 이러한 방식에서는, 모든 물리 레지스터가 모든 타입의 오퍼랜드를 저장하는 데에 이용될 수 있다. 상기 실시예에서, 프로세서(10)는 정수 결과 및 명령의 조건 코드 결과를 저장하기 위하여 단일의 물리 레지스터를 할당할 수 있다. 물리 레지스터에 대응하는 조건 코드 결과를 오버라이트하는 명령의 이후의 퇴거는 동일한 정수 레지스터를 갱신하지 않을 수 있으며, 이에 따라 물리 레지스터는 새로운 조건 코드 결과 전송시 자유롭게 되지 않을 수 있다. 유사하게, 물리 레지스터에 대응하는 정수 레지스터를 갱신하는 명령의 이후의 퇴거는 조건 코드 레지스터를 갱신하지 않을 수 있으며, 이에 따라 물리 레지스터는 새로운 정정수 결과 전송시 자유롭게 되지 않을 수 있다.

또한, 맵 유닛(30) 및 맵 사일로(32)는 PC 사일로(48)로부터 예외 명령들을 수신하도록 구성된다. 예외-야기 명령 동작을 포함하는 라인에 후속하는 명령 동작들의 라인들은 맵 사일로(32) 내에서 무효한 것으로 표시된다. 명령 동작들의 이후 라인들에 대응하는 물리 레지스터 번호들은 퇴거를 위해 대응 라인들을 선택할 때 자유롭게 사용가능해지며 (그리고, 구조 재명명 블록(34)은 무효화된 목적지 레지스터들에 의해 갱신되지 않는다). 또한, 맵 유닛(30)에 의해 유지되는 예측 레지스터 상태는 예외-야기 명령에 대응하는 예측 레지스터 상태로 복귀된다.

명령 동작들의 라인, 발신지 물리 레지스터 번호들, 발신지 큐 번호들, 및 목적지 물리 레지스터 번호들은 맵 유닛(30)에 의해 할당된 큐 번호들에 따라 명령큐들(36A-36B)에 저장된다. 일 실시예에 따르면, 명령큐들(36A-36B)은 대칭이며, 모든 명령들을 저장할 수 있다. 또한, 특정한 명령 동작에 대한 의존성들은 어떠한 명령큐에 저장된 다른 명령 동작들에 대해 발생될 수 있다. 예를 들어, 맵 유닛(30)은 명령큐들(36A-36B)중 하나에 명령 동작들의 라인을 저장하고, 명령큐들(36A-36B)중 다른 하나에 명령 동작들의 다음 라인을 저장한다. 명령 동작은, 명령 동작이 종속적인 이전의 명령 동작들이 실행되고 레지스터 파일들(38A-38B)을 갱신시키며 (그리고 명령 동작이 실행에 대해 스케줄링될 때 까지) 적어도 명령큐들(36A-36B)에 남는다. 일 실시예에서, 명령 동작들은 퇴거될 때 까지 명령큐들(36A-36B)에 남는다.

실행을 위한 특정한 명령 동작의 스케줄링시, 명령큐들(36A-36B)은 특정한 명령 동작이 레지스터 파일들(38A-38B)을 갱신하게 될 클록 주기를 결정한다. 실행 코어들(40A-40B) 내의 다른 실행 유닛들은 다른 수의 파이프라인 스테이지들 (및 이에 따라, 다른 레이턴시들)을 이용할 수 있다. 또한, 특정한 명령들은 다른 것들 보다도 파이프라인 내에서 더 많은 레이턴시를 경험할 수 있다. 따라서, (클록 주기들의 수 내에서) 특정한 명령 동작에 대한 레이턴시를 측정하는 카운트다운이 발생된다. 명령큐들(36A-36B)은 갱신이 일어날 때 까지 지정된 수의 클록 주기들을 기다린 다음, 특정한 명령 동작에 의존하는 명령이 스케줄링될 수 있음을 나타낸다. 각 명령큐들(36A-36B)은 명령큐 내의 명령 동작들에 대한 카운트다운을 유지하며, 카운트다운의 종료시 의존적인 명령 동작들의 스케줄링을 내부적으로 허용한다. 또한, 명령큐는 카운트다운의 종료시 다른 명령큐에 대한 표시들을 제공한다. 이후, 다른 명령큐는 의존성 명령 동작들을 스케줄링할 수 있다. 다른 명령큐에 대한 명령 동작 종료의 이러한 지연된 전송은 레지스터 파일들(38A-38B)이 실행 코어들(40A-40B)중 하나에 의해 제공되는 결과들을 다른 레지스터 파일로 전송할 수 있게 한다. 각각의 레지스터 파일들(38A-38B)은 프로세서(10)에 의해 이용되는 물리 레지스터들의 세트를 구현하며, 실행 코어들(40A-40B)중 하나에 의해 갱신된다. 이후, 갱신들은 다른 레지스터 파일로 전송된다. 주목할 사항으로서, 명령큐들(36A-36B)은, 일단 그의 의존성이 만족스러워졌을 때 (즉, 큐 내에서의 순서에 있어서 순서를 벗어날 때) 명령을 스케줄링할 수 있다.

명령큐(30A)로부터 스케줄링된 명령 동작들은 레지스터 파일(38A)로부터 발신지 물리 레지스터 번호들에 따라 발신지 오퍼랜드들을 읽으며, 실행을 위하여 실행 코어(40A)에 전달된다. 실행 코어(40A)는 명령 동작을 실행시키며, 레지스터 파일(38A) 내의 목적지에 할당된 물리 레지스터를 갱신한다. 어떠한 명령 동작들은 목적지 레지스터들을 갖지 않으며, 이러한 경우 실행 코어(40A)는 목적지 물리 레지스터를 갱신하지 않는다. 또한, 실행 코어(40A)는 명령 동작의 R# 및 (만일 있다면) 명령 동작에 대한 예외 정보를 PC 사일로 및 리더렉트 유닛(48)에 보고한다. 명령큐(36B), 레지스터 파일(38B), 및 실행 코어(40B)는 동일한 방식으로 작동한다.

일 실시예에서, 실행 코어(40A) 및 실행 코어(40B)는 대칭적이다. 각각의 실행 코어(40)는, 예를 들어 부동 소수점 덧셈 유닛, 부동 소수점 곱셈 유닛, 두 개의 정수 유닛, 분기 유닛, 적재 어드레스 발생 유닛, 저장 어드레스 발생 유닛, 및 저장 데이터 유닛을 포함할 수 있다. 실행 유닛들의 다른 구성들이 가능하다.

목적지 레지스터들을 갖지 않는 명령 동작들을 중에는 저장 어드레스 발생, 저장 데이터 연산들, 및 분기 연산들이 있다. 저장 어드레스/저장 데이터 연산들은 결과들을 적재/저장 유닛(42)에 제공한다. 적재/저장 유닛(42)은 메모리 데이터 연산들을 수행하기 위한 D-캐시(44)로의 인터페이스를 제공한다. 실행 코어들(40A-40B)은 명령들의 어드레스 오퍼랜드들을 기초로, 적재 ROP들 및 저장 어드레스 ROP들을 실행시켜 적재 및 저장 어드레스를 각각 발생시킨다. 보다 구체적으로, 적재 어드레스들 및 저장 어드레스들은, (실행 코어들(40A-40B) 및 D-캐시(44) 간의 결합을 직접 경유하여) 실행 코어들(40A-40B)을 실행시켜 발생될 때, D-캐시(44)로 전달된다. D-캐시(44) 내에서 히트하는 적재 어드레스들은 데이터가 D-캐시(44)로부터 레지스터 파일들(38)로 라우팅되게 한다. 반면, 히트의 저장 어드레스들은 저장큐 엔트리로 할당된다. 이후, 저장 데이터는 (데이터를 레지스터 파일들(38A-38B)로부터 적재/저장 유닛(42)으로 라우팅시키는 데에 이용되는) 저장 데이터 명령 동작에 의해 제공된다. 저장 명령의 퇴거시, 데이터는 D-캐시(44)에 저장된다. 또한, 적재/저장 유닛(42)은, (외부 인터페이스(46)를 통해) 이후의 캐시 파일들에 대한 D-캐시(44)를 잃어버린 적재/저장 동작들을 적재/저장 어드레스들을 저장하고, 잃어버린 적재/저장 동작들을 재시도하는 적재/저장 버퍼를 포함할 수 있다. 적재/저장 유닛(42)은 또한 적재/저장 메모리 의존성들을 처리하도록 구성된다.

이제 도 2를 참조하면, 맵 유닛(30), 맵 사일로(32), 및 구조 재명명 블록(34)의 일 실시예의 블록도가 도시되는데, 이는 프로세서(10)의 일 실시예에 따른 이들 간의 상호 연결을 강조한다. 바람직한 경우, 부가적인, 대용의, 또는 보다 적은 상호연결을 이용하는 다른 실시예들이 또한 가능하며 고려된다.

디코드 유닛(24)은, 맵 유닛(30) 및 맵 사일로(32)에 연결된 ROP 정보 버스(60)에 연결된다. 명령 동작들의 라인(또는 ROP들의 라인)에 관련된 정보는 디코드 유닛(24)에 의해 ROP 정보 버스(60) 위에 제공된다. 라인 내의 각각의 ROP에 대해, 디코드 유닛(24)은 유효 표시, ROP가 목적지 레지스터를 기입했는지의 표시, R#, 논리 목적지 레지스터 번호, 및 논리 발신지 레지스터 번호들(2개까지) 중에서 적어도 하나를 제공한다. 맵 유닛(30)은 각각의 ROP에 IQ#를 할당하고, 목적지 레지스터를 기입하는 각각의 ROP에 PR#를 할당한다. 맵 유닛(30)은 할당된 PR# 및 IQ#를 목적지 PR#IQ# 버스(62)를 통해 맵 사일로(32)에 제공한다. 또한, 맵 유닛(30)은 현재의 예측 레지스터 상태 버스(64)를 통해 현재의 예측 레지스터 상태를 맵 사일로(32)에 제공한다. 일반적으로, "예측 레지스터 상태"란 용어는 프로그램 시퀀스의 실행시 (즉, 프로그램 시퀀스에서 특정한 지점 이전에 각 명령의 실행에 후속하여, 그리고 프로그램 시퀀스 내에서 특정한 지점에 후속하는 각 명령의 실행 이전에) 특정한 포인트에서 논리 레지스터들의 상태(즉, 그 내에 저장된 값들)를 식별하는 것을 말한다. 현재의 예측 레지스터 상태는 맵 유닛(30)에 의해 처리되고 있는 ROP들의 라인 이전의 논리 레지스터들에 대응하는 물리 레지스터들의 세트를 식별한다. 즉, 현재의 예측 레지스터 상태는 각 논리 레지스터에 대응하는 물리 레지스터 번호를 저장한다. 또한, 본 실시예에서, 현재의 예측 레지스터 상태는 식별된 물리 레지스터를 갱신하는 명령의 IQ#와, 이 IQ#가 여전히 유효한 지를 (즉, 명령이 아직 퇴거되지 않았는 지를) 나타내는 유효 비트를 포함한다. 맵 사일로(32)는 ROP들의 라인에 대한 엔트리를 할당하고, 현재의 예측 레지스터 상태와 맵 유닛(30)에 의해 제공된 할당된 PR#들 및 IQ#들을 저장한다. 또한, 맵 사일로(30)는 ROP들이 유효한 지를 포획하는데, 이 ROP들은 논리 레지스터들을 갱신하며, 이 논리 레지스터들은 ROP 정보 버스(60)로부터의 ROP들에 의해 갱신된다. 예시적인 맵 사일로 엔트리는 하기에 도시된다(도 9).

일반적으로, 본원에서 이용되는 "사일로"란 용어는 명령, 명령 동작, 또는 명령 동작들의 라인에 대응하는 정보를 저장하는 구조를 말한다. 사일로는 프로그램 순서로 정보를 유지하며, 명령은, 명령이 (예외 조건들의 부재시) 프로그램 순서로 퇴거될 때, 사일로의 정점(top)(또는 꼬리)로부터 사일로의 바닥(또는 헤드)으로 논리적으로 이동한다. 본원에서 이용되는 바와 같이, 명령은 (예를 들어, 명령의 목적지에 할당된 물리 레지스터 번호로 구조 재명명 블록(34)을 갱신하도록 함으로써, 또는 명령에 대응하는 저장 데이터로 D-캐시(44)를 갱신하도록 함으로써) 명령의 결과가 구조 상태로 커밋될 때 퇴거된다.

맵 사일로(32)는 퇴거 유효 라인(66)을 통해 퇴거 유효 신호를 수신하고, 예외 정보 버스(68)를 통해 예외 유효 표시 및 R#를 수신하도록 연결된다. 퇴거 유효 라인(66) 및 예외 정보 버스(68)는 PC 사일로(48)에 연결된다. 어서트(assert)된 퇴거 유효 신호에 응답하여, 맵 사일로(32)는 퇴거된 레지스터 정보를 퇴거 레지스터/PR# 버스(70)를 통해 사일로의 헤드에 있는 엔트리로부터 구조 재명명 블록(34)에 제공한다. 보다 구체적으로, 퇴거 레지스터/PR# 버스(70)는 갱신될 논리 레지스터 번호 및 대응하는 물리 레지스터 번호를 전달할 수 있다. 본 실시예에서, ROP들의 퇴거는 완전한 라인에 대해 동시에 일어난다(즉, PC 사일로(48)는 PC 사일로(48) 및 맵 사일로(32)의 헤드에 있는 라인 내의 각 ROP들이 일단 성공적으로 실행되면, PC 사일로(48)는 퇴거 신호를 보낸다. 따라서, 본 실시예에서는 가장 오래된 라인에 대한 퇴거 신호가 이용될 수 있다. 부분적인 퇴거에 대한 다른 실시예들이 제공될 수 있으며, 개별적인 명령 동작을 통해 저장 장치를 구성할 수 있는데, 이러한 경우에 있어서의 퇴거는 명령 동작 등에 의해 일어날 수 있다.

퇴거 레지스터/PR# 버스(70)를 통해 지정된 논리 레지스터들에 대응하는 엔트리들을 갱신하기 전에, 구조 재명명 블록(34)은 상기 논리 레지스터들에 대응하는 현재의 물리 레지스터 번호들을 읽는다. 즉, 구조 재명명 블록(34)으로부터 비롯

된 물리 레지스터 번호들("이전의 물리 레지스터 번호들")은 구조 재명명 블록(34)으로부터 팝된다. 구조 재명명 블록(34)은 맵 유닛(30)에 연결된 이전의 PR# 버스(72)를 통해 이전의 PR#들을 제공하고, 지정된 논리 레지스터 엔트리들을 퇴거 레지스터/PR# 버스(70)를 통해 제공된 PR#에 의해 갱신한다.

일반적으로, 이전의 PR#는 PR#들의 자유화 목록에 부가되기에 (그리고 후속적인 ROP의 목적지 레지스터로의 할당에) 적절하다. 그러나, 본 실시예에서, 프로세서(10)는 물리적인 레지스터 이용의 효율을 개선하기 위하여 물리적인 레지스터 공유 기술을 이용한다. 예를 들어, 물리적인 레지스터는 정수값 및 조건 코드값(또는 플래그 값)을 저장하도록 할당될 수 있다. 일부 물리적인 레지스터 저장 장치는 정수값을 저장하며, 다른 것들은 조건 코드값을 저장한다. 따라서, 예를 들어 PR#가 할당되었던 정수 레지스터의 갱신시, 이전의 PR#가 팝될 때, PR#는 여전히 그 내에 저장된 조건 코드들을 나타낼 수 있다(반대의 경우도 마찬가지이다). 구조 재명명 블록(34)은 이전의 PR#를 갱신된 구성 상태와 비교하여, 레지스터들이 실질적으로 자유롭게 사용되기에 적절한 지를 결정한다(다른 실시예들이 한 클럭 주기 내에서 갱신 및 비교를 달성할 수 있음에도 불구하고, 이전의 PR# 버스(72)로부터 PR#들을 포획하고 포획된 번호들을 구조 재명명 블록(34)으로 복귀시키는 레지스터(75)에 의해 도 2에 도시됨). 예를 들어, 구조 재명명 블록(34)은 논리 레지스터들에 대응하는 PR#들을 저장하기 위한 내용 주소화 메모리(CAM: Content Addressable Memory)를 이용한다. 구조 재명명 블록(34)은 캠이 이전의 PR# 버스(72)를 통해 전달된 각각의 PR#에 대응하는 캠 매치 버스(74)를 통해 캠 매치 신호를 전달할 수 있다. 맵 유닛(30)은 대응하는 캠 매치 신호가 어서트되지 않은 경우 이전의 PR# 버스(72)를 통해 지정된 레지스터들을 자유롭게 사용할 수 있다. 유익하게는, 물리적인 레지스터의 이용이 더 효율적이 될 수 있으며, 그럼에도 불구하고 물리적인 레지스터들은 자유롭게 사용될 수 있다. 주목할 사항으로서, 고려되는 다른 실시예들에서는, 개별적인 물리 레지스터들이 명령 동작에 응답하여 갱신된 각각의 논리 레지스터에 할당될 수 있다.

주목할 사항으로서, 이전의 PR#가 구조 재명명 블록(34)으로부터 팝될 때 자유롭게 사용되지 못하는 경우, 여전히 이전 PR#에 의해 표현되는 논리 레지스터를 갱신시키는 명령의 이후의 퇴거는 이전 PR#를 자유롭게 사용할 수 있다. 이후의 퇴거시, 캠 매치는 검출되지 않을 수 있다.

본원에서와 같이, 물리 레지스터는 재명명 하드웨어에 의해 처리되고 있는 명령의 목적지 오퍼랜드로의 할당에 대해 유용해지는 경우에 "자유롭게 사용가능해진다". 본 실시예에서, 물리 레지스터는 물리 레지스터가 할당되는 논리 레지스터를 갱신하는 이후 명령의 퇴거시 자유롭게 사용가능해진다. 다른 실시예들은 변형적인 방법들로 레지스터를 자유롭게 사용할 수 있다.

주목할 사항으로서, 라인 내의 하나 또는 그 이상의 명령 동작들이 동일한 논리 레지스터를 갱신할 수 있다. 따라서, 맵 사일로(32) 또는 구조 재명명 블록(34)중 하나는 각 논리 레지스터에 대한 가장 오래된 갱신(즉, 프로그램 순서에 있어서 마지막 갱신)을 식별하기 위하여 퇴거되는 논리 레지스터들을 스캔하는 로직을 포함하며, 그리고 구조 재명명 블록(34) 내에서의 가장 오래된 갱신에 대응하는 물리 레지스터 번호를 저장한다. 더 새로운 갱신은 상기 설명과 유사하게 자유롭게 사용가능해질 수 있다(즉, 어떠한 매치도 일어나지 않는 경우 캠되고 자유롭게 사용가능해진다).

맵 사일로(32)는 물론 PC 사일로(48)로부터 예외 표시를 수신할 수 있다. PC 사일로(48)는 예외 유효 신호를 어서트하고, 예외를 경험하는 명령의 R#를 예외 정보 버스(68)를 통해 맵 사일로(34)에 제공할 수 있다. 맵 사일로(32)는 (라인 내의 각 ROP에 대해 일정한 일부 R#를 이용하여) 예외를 경험하는 명령 동작을 포함하는 ROP들의 라인에 대응하는 사일로 엔트리를 선택한다. 맵 사일로(32)는 예측 레지스터 상태 버스(76)의 복귀시, 선택된 엔트리 내에 저장된 현재의 예측 레지스터 상태를 맵 유닛(30)에 제공한다. 맵 유닛(30)은 예측 레지스터 상태를 복귀된 상태로 되돌린다. 또한, 맵 사일로(32)는 논리 레지스터 번호들, PR#들, 및 라인 내의, 하지만 예외를 경험하는 ROP 이전의 ROP들의 IQ#들을 제공한다. 맵 유닛(30)은 제공된 PR# 및 IQ#들을 이용하여 복귀된 예측 상태를 갱신한다. 유익하게는, 예측 상태는 신속하게 복귀된다. 예외 조건에 응답하여 폐치된 명령들은 재명명의 신속한 복귀로 인해 맵 유닛(30)에 도달할 때 재명명될 수 있다.

또한, 예외에 응답하여, 예외를 경험하는 ROP 이후의 ROP들에 할당된 물리 레지스터들이 자유롭게 사용가능해진다. 맵 사일로(32)는 자유화 PR# 버스(78) 상에서 자유롭게 사용가능하게 될 PR#들을 맵 유닛(30)으로 전달한다. 일 실시예에서, 맵 사일로(32)는 PR#들이 클럭 주기당 한 라인의 속도로 자유롭게 사용가능하도록 구성될 수 있다. 또한, 물리적인 레지스터가 할당되었던 ROP들이 퇴거되지 않았기 때문에, 물리 레지스터들은 캠을 위하여 구조 재명명 블록(34)으로 전달될 필요가 없다.

이제 도 3을 참조하면, 맵 유닛(30)의 블록도가 도시된다. 다른 실시예들이 또한 가능하며 고려된다. 도 3의 실시예에서, 맵 유닛(30)은 레지스터 스캔 유닛(80), IQ/PR# 제어 유닛(82), 예측 레지스터 상태(84), 가상/물리 레지스터 맵 유닛(86), 자유화 목록 제어 유닛(88), 및 자유화 목록 레지스터(90)를 포함한다. 레지스터 스캔 유닛(80)은 (도 2에 도시된 ROP 정보 버스(60)의 일부인) 버스(60A)를 통해 디코드 유닛(24)으로부터 발신지 및 목적지 레지스터 번호들과 (각각에 대한 유

효 표시를) 수신하도록 연결된다. 레지스터 스캔 유닛(80)은 목적지 레지스터 번호들 및 발신지 가상 레지스터 번호들을 가상/물리 레지스터 맵 유닛(86)으로 전달하도록 구성된다. IQ#/PR# 제어 유닛(82)은 (도 2에 도시된 ROP 정보 버스(60)의 일부인) 버스(60B)를 통해 버스에 연결되어, 목적지 레지스터 번호들 및 목적지 레지스터 번호들에 대응하는 유효 표시를 수신한다. 명령큐들(36A-36B)은 꼬리 포인터 버스(92)를 통해 꼬리 포인터들을 제공하는데, 이는 각 큐 내의 엔트리가 현재 그 큐의 꼬리인 지를 나타낸다. 또한, IQ#/PR# 제어 유닛(82)은 목적지 PR#/IQ# 버스(62)에 접속된다. 가상/물리 레지스터 맵 유닛(86)은 예측 레지스터 상태 버스(76) 및 예측 레지스터 상태(84)에 연결되며, 이 예측 레지스터 상태(84)는 현재의 예측 레지스터 상태 버스(64)에 연결된다. 또한, 가상/물리 레지스터 맵 유닛(86)은 발신지 PR#들, 발신지 IQ#들, 목적지 PR#들, 및 발신지/목적지 PR# 및 IQ# 버스(94)위 라인 내의 각 ROP에 대한 IQ#를 명령큐들(36A-36B)에 제공하도록 연결된다. 자유화 목록 제어 유닛(88)은 다음 자유화 PR# 버스(96) 및 할당된 PR# 버스(99)를 통해 IQ#/PR# 제어 유닛(82)에 연결되며, 자유화 목록 레지스터(90)에 연결된다. 또한, 자유화 목록 제어 유닛(88)은 이전 PR# 버스(72), 캠프 매치 버스(74), 및 자유화 PR# 버스(78)에 연결된다.

도 3의 실시예에서, 맵 유닛(30)은 두 개의 저장 장치 파이프라인 디자인을 이용하여 레지스터 재명명을 수행한다. 첫 번째 스테이지에서는, 레지스터 스캔 유닛(80)은 각 발신지 레지스터에 가상 레지스터 번호들을 할당한다. 동시에, IQ#/PR# 제어 유닛(82)은 (명령큐들(36A-36B)에 의해 제공된 꼬리 포인터들을 기초로 하여) IQ#들을 각 ROP에 할당하고, PR#들을 목적지 레지스터를 갖는 ROP들에 할당한다. 두 번째 스테이지에서는, 가상/물리 레지스터 맵 유닛(86)이 (현재의 예측 상태 및 할당된 PR#들에 기초하여) 가상 레지스터 번호들을 물리 레지스터 번호들에 맵핑하고, IQ#/PR# 제어 유닛(82)에 의해 할당된 물리 레지스터 번호들을 대응하는 ROP의 발행 위치로 라우팅시킨다.

레지스터 스캔 유닛(80)에 의해 할당된 가상 레지스터 번호들은 물리 레지스터 번호에 대한 발신지를 식별한다. 예를 들어, 본 실시예에서, 발신지 레지스터들에 대응하는 물리 레지스터 번호들은 (맵 유닛(30)에 의해 이전에 처리된 ROP들의 라인들에 대응하는 갱신들을 나타내는) 예측 레지스터 상태(84) 또는 (이전 ROP의 목적지 오퍼랜드가 발신지 오퍼랜드와 같은 경우, 즉 내부라인 의존성이 존재하는 경우에는) ROP들의 라인 내의 이전 발행 위치로부터 얻어질 수 있다. 즉, 발신지 레지스터 번호에 대응하는 물리 레지스터 번호는 내부라인 의존성이 검출되지 않는 경우 예측 레지스터 상태(84)에 의해 보유되는 물리 레지스터 번호이다. 레지스터 스캔 유닛(80)은 내부라인 의존성 체크를 효율적으로 수행한다. 다른 실시예들은 바람직한 경우 발신지 오퍼랜드들의 다른 발신지들을 제공할 수 있다.

물리 레지스터 번호 맵으로부터 파이프라인 스테이지들로의, 내부라인 의존성 체크/목적지 물리 레지스터 할당을 분리함으로써, 각 스테이지는 더 높은 주파수에서 작동할 수 있게 된다. 따라서, 도 3에 도시된 맵 유닛(30)의 실시예는 발신지 물리 레지스터 번호들의 결정과 함께 내부라인 의존성 체크 및 목적지 물리 레지스터 할당을 수행하는 다른 실시예들보다 더 높은 주파수에서 작동가능할 수 있다. 가상 레지스터 번호들을 이용함으로써, 기능들의 분리를 허용하며, 그리고 하기의 도 8에 도시된 바와 같이, 비교적 단순하고 효과적인 발신지 물리 레지스터 번호들의 맵을 허용한다.

IQ#/PQ# 제어 유닛(82)은 명령큐들(36A-36B)중 하나의 꼬리 포인터를 시작으로 명령큐 번호들을 할당한다. 다시 말해, 라인 내의 첫 번째 ROP는 IQ#로서 선택된 명령큐의 꼬리 포인터를 수신하고, 나머지 ROP들은 꼬리 포인터로부터 증가하는 순서로 IQ#를 수신한다. IQ#/PQ# 제어 유닛(82)은 라인 내의 각각의 ROP들을 동일한 명령큐들(36A-36B)로 할당하고, ROP들의 다음 라인을 나머지 명령큐들(36A-36B)에 할당한다. IQ#/PQ# 제어 유닛(82)은 할당된 ROP들의 번호의 표시를 ROP 할당 버스(98)를 통해 명령큐들(36A-36B)에 전달한다. 이로써, 수신 명령큐는 그의 꼬리 포인터를 갱신하여 그 큐에 대한 ROP들의 할당을 나타낼 수 있다.

IQ#/PQ# 제어 유닛(82)은 그의 자유화 목록 제어 유닛(88)으로부터 자유화 PR#들의 세트를 수신한다. 자유화 PR#들의 세트는 명령 동작들의 라인 내의 목적지 레지스터들에 할당된다. 일 실시예에서, 프로세서(10)는 라인 내의 논리 레지스터 갱신들의 수를 4개로 한정한다(즉, 예측기 미스 디코드 유닛(26)이 다섯 번째 논리 레지스터 갱신을 만난다면, 라인은 이전 명령에서 종료된다). 이에 따라, 자유화 목록 제어 유닛(88)은 자유화 목록(90)로부터 4개의 PR#들을 선택하고, 선택된 레지스터들을 다음 자유화 RP# 버스(96)를 통해 제어 유닛(82)으로 전달한다. 제어 유닛(82)은 할당된 PR# 버스(99)를 통해 실질적으로 할당되었던 PR#들과 응답하며, 그리고 자유화 목록 제어 유닛(88)은 자유화 목록로부터 할당된 물리 레지스터들을 삭제한다. 다른 실시예들은, 어떠한 한정도 없는 것(즉, 각 ROP가 갱신될 수 있다)을 포함하여, 라인 내의 갱신들의 수에 대한 다른 한정들을 이용할 수 있다.

자유화 목록 제어 유닛(88)은 물리 레지스터들의 자유화를 조종하고, 이후 명령들로의 할당에 대해 레지스터들을 선택하도록 구성된다. 자유화 목록 레지스터(90)는, 예를 들어 각 물리 레지스터에 대응하는 비트를 저장할 수 있다. 비트가 세트되면, 대응하는 레지스터는 자유롭게 사용가능해진다. 비트가 제거되면, 대응하는 레지스터는 할당된다(즉, 자유롭게 사용가능해지지 않는다). 자유화 목록 제어 유닛(88)은 자유화 목록을 스캔하여 제어 유닛(82)으로 전달하기 위한 레지스터들을 선택한다. 예를 들어, 자유화 목록 제어 유닛(88)은 자유화 목록 레지스터(90)의 각 끝으로부터 첫 번째 두 개의 자유

화 레지스터들을 스캔하여 본 실시예에서 제공되는 네 개의 레지스터들을 신속하게 선택할 수 있게 한다. 이러한 스캔들은 각 끝으로부터 두 개를 선택하여 한 연산을 하게 한다(하나는 다른 것 전에 수행되고 할당된 물리 레지스터를 자유화 목록로부터 제거한다).

자유화 목록 제어 유닛(88)은 이전의 PR# 버스(72)를 통해 구조 재명명 블록(34)으로부터 팝된 이전의 물리 레지스터 번호들을 수신한다. 이후, 각각의 이전 물리 레지스터 번호에 대응하는 캐시 매치 신호들이 캐시 매치 버스(74)를 통해 수신된다. 대응하는 캐시 매치 신호가 디어서트(deassert)되는 각각의 이전 PR#가 자유화 목록 제어 유닛(88)에 추가된다. 또한, 자유화 RR# 버스(78)를 통해 수신된 물리 레지스터 번호들은 자유화 목록에 무조건적으로 추가된다.

예측 레지스터 상태(84)는 가상/물리 레지스터 맵 유닛(86)에 제시되는 ROP들의 라인에 대응하는 갱신들 이전에 예측 레지스터 상태를 저장한다. 보다 구체적으로, 예측 레지스터 상태(84)는 각각의 논리 레지스터에 대응하는 물리 레지스터 번호, 및 (본 실시예에서는) 목적지 레지스터로서 할당된 물리 레지스터 번호를 갖는 ROP에 대응하는 명령류 번호를 저장한다. 각 클럭 주기마다, 예측 레지스터 상태(84)는 현재의 예측 레지스터 상태 버스(64)를 통해 현재의 예측 레지스터 상태를 맵 사일로(32)에 전달한다. 가상/물리 레지스터 맵 유닛(86)은 PR#의 발신지가 예측 레지스터 상태(84)임을 나타내는 가상 레지스터 번호를 갖는 각 발신지 레지스터에 대한 예측 레지스터 상태(84)에 의해 표시되는 대응하는 논리 레지스터의 PR# 및 IQ#를 공급한다. 가상 레지스터 번호가 이전 발행 위치를 나타내는 발신지 레지스터들은 제어 유닛(82)에 의해 할당된 대응하는 PR# 및 IQ#를 공급한다. 또한, 가상/물리 레지스터 맵 유닛(86)은 ROP들의 라인에 의해 재정된 논리 목적지 레지스터들 및 제어 유닛(82)에 의해 할당된 목적지 PR#들/IQ#들에 따라서 예측 레지스터 상태(84)를 갱신한다.

가상/물리 레지스터 맵 유닛(86)은 또한, (상기 설명한 바와 같이) 예외 조건에 응답하여 복구 예측 레지스터 상태 버스(76)를 통해 맵 사일로(32)에 의해 제공되는 복구 예측 레지스터 상태를 수신하도록 구성된다. 가상/물리 레지스터 맵 유닛(86)은 맵 사일로(32)에 의해 제공된 복구 예측 상태에 의해, 레지스터 스캔 유닛(80) 및 IQ#/PR# 제어 유닛(82)으로부터의 입력들에 따라 발생된 다음 예측 레지스터 상태를 무효로할 수 있다.

주목할 사항으로서, 본 실시예에서, IQ#들은 대응하는 ROP에 의존하는 명령류 엔트리들을 나타내기 위하여 각각의 발신지 오퍼랜드에 대해 라우팅된다. 명령류들(36A-36B)은 의존적인 ROP를 실행을 위하여 스케줄링하기 전에 대응하는 명령류 내에서의 ROP들의 완료를 기다린다.

이제 도 4를 참조하면, 레지스터 스캔 유닛(80)의 일 실시예가 도시된다. 다른 실시예들이 또한 가능하며 고려된다. 도 4의 실시예에서, 레지스터 스캔 유닛(80)은 ROP들의 라인 내의 각 발행 위치에 대응하는 스캔 유닛을 포함한다. 예를 들어, 본 실시예는 (그 이상 또는 그 이하가 실행될 수 있기는 하지만) 8개의 발행 위치들에 대항하는 8개의 스캔 유닛들(100A-100H)을 포함한다. 스캔 유닛들(100A-100H)은 가상의 예측 레지스터 상태를 전달하기 위하여 직렬 체인 내에 연결되며, 동시에 발신지 및 목적지 레지스터 번호들을 수신한다.

각각의 스캔 유닛들(100A-100H)은 도 3에 도시된 ROP 버스(60A)의 일부에 연결된다. 좀 더 명확하게 설명하면, 각각의 스캔 유닛들(100A-100H)은 대응하는 발행 위치들 내의 ROP의 발신지 및 목적지 번호들을 수신하도록 연결된다. 따라서, 버스(60AA)는 스캔 유닛(100A)에 연결되어, 발행 위치 0(즉, ROP들의 라인 내에서 프로그램 순서에 있어서 첫 번째 ROP)에 대한 발신지 및 목적지 레지스터 번호들을 제공한다. 유사하게, 버스(60AB)는 스캔 유닛(100B)에 연결되어, 발행 위치 1에 대한 발신지 및 목적지 레지스터 번호들을 제공한다. 다른 버스들(60AC-AH)은 도시된 바와 같은 순서의, 남아 있는 발행 위치들에 대응하는 발신지 및 목적지 레지스터 번호들을 제공한다. 각각의 스캔 유닛(100A-100H)은 각 발신지 레지스터에 대한 발신지 가상 레지스터 번호를 제공하도록 구성되며, 이는 이후에 가상/물리 레지스터 맵 유닛(86)에 전달된다.

스캔 유닛(100A)은 가상의 현재 예측 레지스터 상태를 수신하도록 결합된다. 가상의 현재 예측 레지스터 상태는 각각의 논리 레지스터에 대응하는 가상의 레지스터 번호를 포함한다. 가상의 현재 예측 상태 내의 가상의 레지스터 번호들은 PR# (및 IQ#)의 발신지가 예측 레지스터 상태(84)임을 나타낸다. x86 명령 세트 구조를 이용하는 다른 실시예에서, 예를 들어, 예측 레지스터 상태(84)는 구성된 정수 레지스터들에 대응하는 8개의 레지스터들과, 조건 코드 비트들의 그룹(이러한 그룹핑은 어떠한 것들이 다양한 명령들에 의해 갱신되는 지에 따라 선택된다. 예를 들어, 0비트가 한 그룹으로 선택되고, C 비트는 다른 그룹으로 선택되고, Z, A, P 및 S 비트들은 제 3 그룹으로 선택된다)에 대응하는 3개의 레지스터들과, 구성된 부동 소수점/MMX 레지스터들에 대응하는 8개의 레지스터들과, 여기서 한 레지스터는 부동 소수점 조건 코드 레지스터에 대응하며, 그리고 잠정 마이크로코드 레지스터들에 대응하는 8개의 레지스터들을 포함한다.

스캔 유닛(100A)은 발신지 레지스터 번호들을 기초로 하여 가상의 현재 예측 레지스터 상태로부터 발신지 가상 레지스터 번호들을 할당한다. 또한, 발행 위치 0 내의 ROP가 목적지 레지스터를 갱신한다면, 스캔 유닛(100A)은 대응하는 논리 목

적지 레지스터에 대한 새로운 가상 레지스터 번호를 삽입함으로써 가상의 현재 예측 상태를 갱신한다. 새로운 가상 레지스터 번호는 대응하는 논리 레지스터에 대한 PR# 및 IQ#의 발신지가 발행 위치 0임을 나타낸다. 초기의 가상 레지스터 번호 대신에 삽입된 새로운 가상의 레지스터 번호를 갖는 갱신된 예측 레지스터 상태가 스캔 유닛(100B)에 전달된다.

스캔 유닛(100B)은 스캔 유닛(100A)으로부터 갱신된 예측 레지스터 상태를 수용하고, 갱신된 예측 레지스터 상태로부터의 가상 레지스터 번호들을 발신지 레지스터 번호들에 할당한다. 또한, 스캔 유닛(100B)은, 발행 위치 1 내의 ROP가 논리 레지스터를 갱신하는 경우, 발행 위치 1을 나타내는 새로운 가상 레지스터 번호들을 스캔 유닛(100A)에 의해 제공되는 갱신된 예측 레지스터 상태에 삽입한다. 스캔 유닛들(100C-100H)은 이전의 스캔 유닛들에 의해 제공된 갱신된 예측 레지스터 상태에 응답하여, 각각의 발행 위치들 3-8 내의 ROP들의 발신지 레지스터들에 대한 가상 레지스터 번호들을 할당하며, 그리고 만일 있는 경우, 목적지 레지스터 번호에 따라서, 갱신된 예측 레지스터 상태를 갱신한다.

따라서, 만일 라인 내의 이전의 ROP가 ROP의 발신지 레지스터 번호에 의해 지정된 논리 레지스터를 갱신한다면, 이전 발행 위치를 나타내는 가상의 레지스터 번호가 할당된다. 그렇지 않으면, 대응하는 논리 레지스터에 대한 현재의 예측 레지스터 상태를 나타내는 가상 레지스터 번호가 할당된다. 다시 말해, 내부라인 의존성 및 (현재의 예측 레지스터 상태를 통한) 명령들의 이전 라인에 대한 의존성이 가상 레지스터 번호들에 의해 표시된다.

스캔 유닛(100H)에 의해 제공된 갱신된 예측 레지스터 상태는 가상의 다음 예측 레지스터 상태이며, 이는 각각의 스캔 유닛들(100A-100H)에 의해 제공된 발신지 가상 레지스터 번호들 및 목적지 레지스터 번호들과 함께 가상/물리 레지스터 맵 유닛(86)에 전달된다. 이후, 가상/물리 레지스터 맵 유닛(86)은, 이후의 ROP들의 라인에 대한 예측 레지스터 상태(84)를 갱신하기 위하여, ROP들의 라인에 대응하는 다음 예측 레지스터 상태를 발생시킬 수 있다.

주목할 사항으로서, 일 실시예에서, 각각의 스캔 유닛(100A-100F)은 정수, 잠정적인, 그리고 조건 코드 레지스터들을 처리하는 정수/잠정 스캔 회로, 및 부동 소수점 레지스터들과 부동 소수점 조건 코드 레지스터를 처리하는 부동 소수점 스캔 회로를 포함한다. 따라서, 각 발신지 및 목적지 레지스터가 부동 소수점 또는 정수 레지스터인지의 표시가 레지스터 번호와 함께 라우팅되며, 가상 레지스터 번호가 레지스터에 할당될 것인지, 새로운 레지스터 번호로 대체할 것인지 결정하는 데에 이용된다.

이제 도 5를 참조하면, 가상 레지스터 번호들의 예시적인 엔코딩을 나타내는 표(102)가 도시된다. 다른 엔코딩이 또한 가능하며 고려된다. 명확하게 설명하면, 예를 들어, 표(102)에 도시된 최상위 비트(MSB)는 표에 도시된 것으로부터 반전될 수 있다. 또 다른 엔코딩이 또한 가능하다.

표(102)는 MSB가, 대응하는 PR#/IQ#에 대한 발신지가 예측 레지스터 상태(84)에 의해 유지된 현재의 예측 상태인지, 또는 라인 내의 이전 발행 위치에 할당된 목적지 PR#/IQ#인지를 결정하는 가상 레지스터 번호 엔코딩을 도시한다. 예를 들어, MSB가 제거된다면, 가상 레지스터 번호는 발신지가 현재의 예측 상태이며 가상 레지스터 번호의 최하위 비트들(LSB들)이 현재의 예측 상태 내의 논리 레지스터 번호임을 나타낸다. 만일 MSB가 세트된다면, 가상 레지스터 번호는 발신지가 이전 발행 위치임을 나타낸다. 이러한 경우에서의 LSB들은 이전 발행 위치 번호이다.

표(102)에 도시된 엔코딩을 이용하는 실시예에서, 스캔 유닛(100A)에 제공되는 가상의 현재 예측 상태는 MSB 제거에 의한 엔코딩, 및 LSB들에 제공되는 대응하는 논리 레지스터 번호를 포함한다. 스캔 유닛들(100A-100H)에 의해 갱신된 예측 레지스터 상태에 삽입되는 새로운 가상 레지스터 번호들은 MSB 세트에 의한 엔코딩 및 LSB들에 제공된 삽입 스캔 유닛에 대응하는 발행 위치 번호를 포함한다.

이제 도 6을 참조하면, 정수/잠정 스캔 회로(110)의 일 실시예의 일부가 도시되는데, 이는 각각의 스캔 유닛들(100A-100H)의 일 실시예에서 이용될 수 있다. 다른 실시예들이 또한 가능하며 고려된다. 도 6의 실시예에서, 대응하는 ROP의 목적지 레지스터 번호는 목적지 버스(112)를 통해 전달되며, 대응하는 ROP의 제 1 발신지(SRC1) 레지스터 번호는 SRC1 버스(114)를 통해 전달되고, 대응하는 ROP의 제 2 발신지(SRC2) 레지스터 번호는 SRC2 버스(116)를 통해 전달된다. 버스들(112, 114 및 116)은 버스(60A)의 일부를 포함한다(예를 들어, 정수/잠정 스캔 회로(110)가 스캔 유닛(110A)의 일부 이라면, 버스들(112, 114 및 116)은 버스(60AA)의 일부가 된다). 목적지 버스(112)는 다수의 목적지 식별자 유닛들(예를 들어, 도 6에 도시된 유닛들(118A 및 118B))에 연결된다. 각각의 목적지 식별자 유닛들(118A-118B)은 전달/기입 제어 유닛에 연결된다(예를 들어, 전달/기입 제어 유닛들(120A 및 120B)이 각각 목적지 식별자 유닛들(118A 및 118B)에 연결된다). 각각의 전달/기입 제어 유닛(120A-120B)은 또한 이전의 스캔 유닛(100A-100H)에 의해 제공된 가상의 레지스터 입력, 및 이후의 스캔 유닛(100A-100H)으로의 가상 레지스터 출력에 연결된다(예를 들어, 전달/기입 제어 유닛(120A)은 가상의 EAX 입력 버스(122A) 및 가상의 EAX 출력 버스(124A)에 연결되며, 전달/기입 제어 유닛(120B)은 가상의 EBX 입력 버스(122B) 및 가상의 EBX 출력 버스(124B)에 연결된다). 또한, SRC1 버스(114)는 다수의 SRC1 식별자 유닛들(예

를 들어, 도 6에 도시한 SRC1 식별자 유닛들(126A 및 126B)에 연결된다. SRC2 버스(116)는 유사하게 다수의 SRC2 식별자 유닛들(예를 들어, 도 6에 도시한 SRC2 식별자 유닛들(128A 및 128B))에 연결된다. 각각의 SRC1/SRC2 식별자 유닛들(126A-126B)은 도 6에 도시된 대응 스위치(130A-130D)의 인에이블 제어부에 연결된다. 스위치들(130A-130B)은 가상의 EAX 입력 버스(122A)에 연결된 입력들을 갖는 반면, 스위치들(130C-130D)은 가상의 EBX 입력 버스(122B)에 연결된 입력들을 갖는다. 스위치들(130A 및 130C)은 SRC1 가상 레지스터 버스(132A)에 연결되는 반면, 스위치들(130B 및 130D)은 SRC2 가상 레지스터 버스(132B)에 연결된다.

일반적으로, 정수/잠정 스캔 회로(110)는 대응하는 발행 위치 내의 ROP의 정수/잠정 발신지 레지스터들에 대한 가상 레지스터 번호들을 할당하고, 정수 목적지 레지스터에 대한 새로운 가상 레지스터 번호들을 대응하는 발행 위치에 삽입한다. 목적지 식별자 유닛, 전달/기입 제어 유닛, SRC1 식별자 유닛, 및 SRC2 식별자 유닛의 각 세트는 논리 정수, 잠정, 또는 조건 코드 레지스터들 중 하나에 대응한다. 예를 들어, 목적지 식별자 유닛(118A), 전달/기입 제어 유닛(120A), SRC1 식별자 유닛(126A), 및 SRC2 식별자 유닛(128A)은 EAX 레지스터에 대응한다. 유사하게, 목적지 식별자 유닛(118B), 전달/기입 제어 유닛(120B), SRC1 식별자 유닛(126B), 및 SRC2 식별자 유닛(128B)은 EBX 레지스터에 대응한다. 목적지 식별자 유닛, 전달/기입 제어 유닛, SRC1 식별자 유닛, 및 SRC2 식별자 유닛(미도시)의 다른 세트들은 논리 정수, 잠정, 또는 조건 코드 레지스터들 중 다른 것들에 대응한다. 이후 EAX 레지스터 하드웨어에 대해 설명할 것이다. EBX 레지스터 하드웨어는 EBX 레지스터에 대해 유사하게 작동하며, 다른 유사한 하드웨어(미도시)는 나머지 레지스터들에 대해 유사하게 작동한다.

목적지 식별자 유닛(118A)은 목적지 레지스터 버스(112) 상의 목적지 레지스터 번호가 EAX 레지스터를 선택하는 지를 결정한다. 따라서, 목적지 식별자 유닛(118A)은 EAX가 선택되었는 지를 결정하기 위하여 목적지 레지스터 번호를 디코드하는데, 이 디코드는 목적지 레지스터 번호가 유효함을 나타내는 유효 신호와, 목적지 레지스터 번호가 정수/잠정/조건 코드 레지스터(즉, 부동 소수점 레지스터가 아님)임을 나타내는 정수 신호에 의해 유효해진다. 만일 목적지 레지스터가 EAX 레지스터이면, 목적지 식별자 유닛(118A)은 전달/기입 제어 유닛(120A)에, 정수/잠정 스캔 회로(110)가 가상의 EAX 출력 버스(124A) 상에서 이용되는 발행 위치에 대응하는 가상 레지스터 번호를 삽입하도록 신호를 보낸다. 그렇지 않으면, 목적지 식별자 유닛(118A)은 전달/기입 제어 유닛(120A)에, 가상의 EAX 입력 버스(122A)를 통해 제공된 가상 레지스터 번호를 가상의 EAX 출력 버스(124A)로 전달하도록 신호를 보낸다.

유사하게, SRC1 식별자 유닛(126A)은, EAX가 선택되는 지를 결정하기 위하여 SRC1 레지스터를 디코드하고, 이 디코드를 SRC1 레지스터 번호가 유효함을 나타내는 유효 신호와, SRC1 레지스터 번호가 정수/잠정/조건 코드 레지스터임을 나타내는 정수 신호에 의해 유효화함으로써, SRC1 레지스터 버스(114) 상의 SRC1 레지스터 번호가 EAX 레지스터를 선택하는 지를 결정한다. 만일 EAX가 SRC1으로 선택된다면, SRC1 식별자 유닛(126A)은 스위치(130A)를 구동시켜 가상 EAX 입력 버스(122A)를 통해 제공된 가상 레지스터 번호를 SRC1 가상 레지스터 버스(132A)로 드라이브한다. SRC2 식별자 유닛(128A)은 SRC1 식별자 유닛(126A)과 유사하지만, SRC2 레지스터 버스(116)를 통해 제공된 SRC2 레지스터 번호에 대해 작동하며, 스위치(130B)를 제어하여 SRC2 가상 레지스터 버스(132B)를 드라이브시키거나, 또는 드라이브시키지 않는다.

이러한 방식으로, 갱신된 예측 상태는 (가령 124A-124B와 같은 가상의 출력 버스들을 통해) 다음 스캔 유닛으로 전달되며, 발신지 가상 레지스터 번호들은 (가령 SRC1 가상 레지스터 버스(132A) 및 SRC2 레지스터 번호 버스(132B)와 같은 버스들을 통해) 할당될 수 있다. 정수/잠정 스캔 회로(110)는 각각의 가상 정수/잠정 입력들을 전달/기입 제어 유닛(120A) (및 다른 전달/기입 제어 유닛들)로 라우팅시킴으로써 레지스터 대 레지스터 이동 연산들을 처리하도록 변경될 수 있다. 부가적인 신호가 현재 발행 위치에서의 ROP가 레지스터-레지스터 이동임을 나타내는 데에 이용될 수 있다. SRC1 레지스터 번호 및 이동 신호에 응답하여, SRC1 레지스터 번호에 대응하는 가상 정수/잠정 입력은 목적지 레지스터에 대응하는 가상 정수/잠정 출력으로 라우팅된다. 또한, 대응하는 발행 위치 내의 ROP는 명령큐들(36A-36B)로 전달되는 것이 저지되는데 (예를 들어, 그의 유효 비트가 리셋된다), 이는 발신지 가상 레지스터 번호를 목적지 가상 레지스터 번호로서 라우팅시킴으로써 이동이 이루어지기 때문이다. 상기 실시예에서, 자유롭게 사용가능해지기 전에 퇴거된 물리 레지스터들의 캐밍(camming)은 목적지 PR#의 우연한 초기 자유화를 방해한다.

이제 도 7을 참조하면, 부동 소수점 스캔 회로(140)의 일 실시예의 일부가 도시되는데, 이는 각각의 스캔 유닛들(100A-100H)의 일 실시예에서 이용될 수 있다. 다른 실시예들이 또한 가능하며 고려된다. 스캔 회로(140)는 정수/잠정 스캔 회로(110)와 유사하게 SRC1 레지스터 버스(114) 및 SRC2 레지스터 버스(116)를 수신하고, SRC1 가상 레지스터 버스(132A) 및 SRC2 가상 레지스터 버스(132B)를 통해 발신지 가상 레지스터 번호들을 드라이브시킬 수 있다. 도 7에 도시된 부분은 가상 stI 입력(x86 명령 세트 구조에서 규정된 st0 내지 st7 레지스터들 중 하나, $0 < I < 7$)을 수신하고, 회로(140)가 이용되는 발행 위치("현재의 발행 위치")에 가상의 stI 출력을 제공한다. SRC1 레지스터 버스(114)에 연결된 SRC1 식별자 유닛(142A)은 SRC1 레지스터 번호가 stI 레지스터를 선택하고 있는 지를 결정한다(즉, 레지스터 번호는 stI이고, 레지스터는

부동 소수점 레지스터이며, 그리고 SRC1 레지스터는 유효하다). 만일 SRC1 레지스터 번호가 stI 레지스터를 선택한다면, SRC1 식별지 유닛(142A)은 스위치(130E)를 구동시켜 가상 stI 입력 버스(144)를 통해 제공된 가상 레지스터 번호를 SRC1 가상 레지스터 버스(132A)로 드라이브시킨다. 유사하게, SRC2 레지스터 버스(116)에 연결된 SRC2 식별지 유닛(142B)은 SRC2 레지스터 번호가 stI 레지스터를 선택하는 지를 결정하고, 스위치(130F)를 선택적으로 제어하여 가상 stI 입력 버스(144)를 통해 제공된 가상 레지스터 번호를 SRC2 가상 레지스터 버스(132B)로 드라이브시킨다.

또한, 도 7에 도시된 부동 소수점 스캔 회로(114)의 일부는 레지스터 stI에 대응하는 출력 가상 레지스터 번호를 가상 stI 출력 버스(146) 상에 제공한다. 다양한 입력 가상 레지스터 번호들은 디코드 유닛(24)에 의해 제공되는 다양한 제어 신호들에 의해 제어되는, 출력 버스(146) 상에서의 출력 가상 레지스터 번호로서 선택가능할 수 있다. x86 부동 소수점 명령들은 부동 소수점 레지스터 세트를 스택으로서 처리한다. st0은 스택의 정점에 있는 레지스터이고, st1은 정점 다음의 레지스터이다. 어떠한 명령들은 스택을 (현재의 st0 레지스터를 레지스터 st1 등으로 하면서) 푸쉬되거나, (현재의 st1 레지스터를 st0 등으로 하면서) 팝되게 한다. 또한, 스택 레지스터의 정점(st0)과 나머지 레지스터들중 하나를 바꾸는 교환 명령들이 제공된다.

가상 stI 출력을 선택함으로써, 스위치들(130G-130K) 및 한 세트의 입력 가상 레지스터 버스들(144, 148, 150, 152 및 154)을 이용하여 상기의 많은 상황들을 처리하고자 한다. 만일 현재 발행 위치 내의 ROP가 부동 소수점 스택 또는 stI 레지스터에 개별적으로 영향을 주지 못한다면, 전달 신호가 (가상 stI 입력 버스(144)와 가상 stI 출력 버스(146) 사이에 연결된) 스위치(130G)에 어서트되고, 가상 stI 입력은 가상 stI 출력으로서 제공된다. 반면에, 현재 발행 위치 내의 ROP가 stI 레지스터를 갱신한다면, 기입 신호가 (버스(152)와 버스(146) 사이에 연결된) 스위치(130J)에 어서트되고, (버스(152)를 통해 전달된) 현재 발행 위치에 대응하는 가상 레지스터 번호가 가상 stI 출력 버스(146)에 전달된다. 만일 현재 발행 위치 내의 ROP가 스택을 푸쉬 또는 팝시킨다면, 대응하는 신호가 (버스(154)와 버스(146) 사이에 연결된) 스위치들(130K) 및 (버스(152)와 버스(146) 사이에 연결된) 스위치들(130I)에 각각 어서트된다. 이로써, 현재 발행 위치에 제공된 갱신된 예측 상태 내의 stI-1 및 stI+1 레지스터들에 대응하는 가상 레지스터 번호들이 제공된다. 최종적으로, 현재 발행 위치에 두 개 이전의 발행 위치로부터의 가상 st0 입력이 버스(148)를 통해 제공된다. 만일 현재의 ROP가 FEXC 명령들의 두 번째 반이라면, EXCH 신호가 스위치(130H)에 어서트되며, 현재 발행 위치 이전의 발행 위치 2 로부터의 st0에 대응하는 가상 레지스터 번호가 가상 stI 출력으로서 선택된다.

주목할 사항으로서, FEXC 명령의 첫 번째 반을 처리하기 위하여, st0에 대응하는 부동 소수점 스캔 회로(140)의 부분은 이전의 스캔 유닛으로부터 각각의 가상 stI 입력들을 포함하여, 모든 레지스터를 가상의 st0 출력으로서 임의로 선택할 수 있다. 또한, 주목할 사항으로서, 부동 소수점 상태 레지스터의 스택의 정점(TOS) 필드 및 부동 소수점 태그 워드는 부동 소수점 조작에 의해 영향을 받는다. TOS 및 태그 워드에 대한 예측 값들은 푸쉬, 팝, 및 교환을 통해 전파될 수 있다. TOS 및 태그 워드의 현재의 예측 카피는 예측 레지스터 상태(84) 내에 유지될 수 있다. 또한, 각 발행 위치에 대응하는 TOS 및 태그 워드의 값은 예외 복구를 위해 맵 사일로(32)에 저장될 수 있으며 (그리고 마지막 값은 예측 레지스터 상태(84)내에 갱신될 수 있다). 또한, 레지스터 스캔 유닛(80)은 (태그 워드에 의해 나타난 바와 같이) 무효한 레지스터의 이용을 검출하고, 이후의 예외 처리를 위해 레지스터를 이용하여 ROP에 의한 예외에 주목할 수 있다.

주목할 사항으로서, 정수/잠정 스캔 회로(110) 및 부동 소수점 스캔 회로(140)는 정수 대 부동 소수점 및 부동 소수점 대 정수 이동을 처리할 수 있다. 만일 ROP의 발신지 레지스터가 정수인 것으로 표시되면, 정수/잠정 스캔 회로(110)는 발신지 가상 레지스터 번호를 제공한다. 반면에, ROP의 발신지 레지스터가 부동 소수점인 것으로 표시되면, 부동 소수점 스캔 회로(140)는 발신지 가상 레지스터 번호를 제공한다. 만일 목적지 레지스터가 부동 소수점으로 표시된다면, 새로운 부동 소수점 가상 레지스터 번호가 부동 소수점 스캔 회로(140)에 의해 갱신된 예측 레지스터 상태에 제공된다. 반면에, 목적지 레지스터가 정수로서 표시된다면, 새로운 정수 가상 레지스터 번호가 정수/잠정 스캔 회로(110)에 의해 갱신된 예측 레지스터 상태에 제공된다. 따라서, 부동 소수점과 정수 레지스터간 이동을 갖는 발행 위치에는 부동 소수점 발신지 가상 레지스터 번호가 할당되며, 목적지 레지스터 번호는 정수를 나타낸다. 반면, 정수와 부동 소수점간 이동을 갖는 발행 위치에는 정수 발신지 가상 레지스터 번호가 할당되며, 목적지 레지스터 번호는 부동 소수점을 나타낸다.

이제 도 8을 참조하면, 가상/물리 레지스터 맵 유닛(86)의 일 실시예의 블록도가 도시된다. 다른 실시예들이 또한 가능하며 고려된다. 도 8의 실시예에서, 가상/물리 레지스터 맵 유닛(86)은 발신지 IQ# mux(160), 다음 예측 IQ# mux(162), 발신지 PR# mux(164), 다음 예측 PR# mux(168), 트랩 IQ# mux(170), 및 트랩 PR# mux(172)을 포함한다. 발신지 IQ# mux(160)은 예측 레지스터 상태(84)로부터 각 논리 레지스터에 대응하는 현재의 예측 IQ#를 수신하도록, 그리고 IQ#/PR# 제어 유닛(82)에 의해 할당된 목적지 IQ#들을 수신하도록 연결된다. 유사하게, 다음 예측 IQ# mux(162)는 현재의 예측 IQ# 및 목적지 IQ#들을 수신하도록 연결된다. 발신지 IQ# mux(160)의 출력은 명령큐들(36A-36B)에 파이프라인되며, 다음 예측 IQ# mux(162)의 출력은 트랩 IQ# mux(170)에 입력으로서 연결된다. 트랩 IQ# mux(170)은 또한 복구 예측 레지스터 상태 버스(76A)(복구될 IQ#들을 전달하는 복구 예측 레지스터 상태 버스(76)의 일부)에 연결된다. 발신지 PR#

mux(164)는 예측 레지스터 상태(84)로부터 각 논리 레지스터에 대응하는 현재의 예측 PR#를 수신하도록, 그리고 IQ#/PR# 제어 유닛(82)에 의해 할당된 목적지 PR#들을 수신하도록 연결된다. 유사하게, 다음 예측 PR# mux(168)는 현재의 예측 PR#들 및 목적지 PR#들을 수신하도록 연결된다. 발신지 PR# mux(164)의 출력은 명령큐들(36A-36B)에 파이프라인된다. 다음 예측 PR# mux(168)의 출력은 트랩 PR# mux(172)에 연결되며, 이는 또한 복구 예측 레지스터 상태 버스(76B)(복구될 PR#들을 전달하는 복구 예측 레지스터 상태 버스(76)의 일부)에 연결된다. 레지스터 스캔 유닛(80)에 의해 할당된 발신지 가상 레지스터 번호들은 발신지 IQ# mux(160) 및 발신지 PR# mux(164)에 선택 제어 신호들로서 제공된다. 가상의 다음 예측 상태는 다음 예측 IQ# mux(162) 및 다음 예측 PR# mux(168)에 선택 신호들로서 제공된다. PC 사일로(48)로부터의 트랩 제어 신호들은 트랩 IQ# mux(170) 및 트랩 PR# mux(172)에 대한 선택 신호들을 제공한다.

일반적으로, 발신지 IQ# mux(160) 및 발신지 PR# mux(164)는 레지스터 스캔 유닛(80)에 의해 제공되는 대응하는 발신지 가상 레지스터 번호에 응답하여 각 ROP의 각 발신지 오퍼랜드에 대한 발신지 IQ# 및 PR#를 선택한다. Mux(160)는 예를 들어, 도시된 바와 같이 mux(160)에 대한 입력들을 수신하고, 선택 제어 신호로서 대응하는 발신지 가상 레지스터 번호를 수신하도록 연결된 mux들의 병렬 세트(각 ROP의 각 발신지 레지스터에 각각 하나)로서 구현될 수 있다. 유사하게, mux(164)는 도시된 바와 같이 mux(164)에 대한 입력들을 수신하고, 선택 제어 신호로서 대응하는 발신지 가상 레지스터 번호를 수신하도록 연결된 mux들의 병렬 세트(각 ROP의 각 발신지 레지스터에 각각 하나)로서 구현될 수 있다. 만일 발신지 가상 레지스터 번호가 현재의 예측 상태가 IQ#/PR#의 발신지임을 나타낸다면, 발신지 가상 레지스터 번호에 포함된 논리 레지스터 번호는 예측 레지스터 상태(84)에 의해 제공된 IQ# 및 PR#중 하나를 선택하는 데에 이용된다. 반면에, 만일 발신지 가상 레지스터 번호가 이전의 발행 위치를 나타낸다면, 발행 위치 번호는 제어 유닛(82)에 의해 할당된 목적지 IQ#/PR# 중 하나를 선택하는 데에 이용된다. 제어 유닛(82)은, 예를 들어 각 발행 위치에 대응하는 목적지 IQ# 및 PR#를 제공할 수 있다. 반면에, 제어 유닛(82)은 (라인 내의 ROP들의 수보다 적은) 한정된 수의 IQ# 및 PR#를 제공할 수 있다. 상기 실시예에서는, 목적지 IQ#/PR# 결합중 하나를 선택하기 위하여 발신지 가상 레지스터 번호 및 각 ROP에 대한 목적지 레지스터 유효 표시들에 대한 논리가 수행될 수 있다.

유사하게, 다음 예측 IQ# mux(162) 및 다음 예측 PR# mux(168)는 레지스터 스캔 유닛(80)에 의해 제공되는 대응하는 가상의 다음 예측 상태에 응답하여 각 논리 레지스터에 대한 IQ# 및 PR#를 선택한다. Mux(162)는, 예를 들어 도시된 바와 같이 mux(162)에 대한 입력들을 수신하고 선택 제어 신호로서 대응하는 가상 레지스터 번호를 수신하도록 연결된 mux들의 병렬 세트(각 논리 레지스터에 하나)로서 구현될 수 있다. 유사하게, mux(168)는 도시된 바와 같이 mux(168)에 대한 입력들을 수신하고 선택 제어 신호로서 대응하는 가상 레지스터 번호를 수신하도록 연결된 mux들의 병렬 세트(각 논리 레지스터에 하나)로서 구현될 수 있다. 만일 가상 레지스터 번호가 현재의 예측 상태가 특정한 논리 레지스터의 IQ#/PR#에 대한 발신지임을 나타낸다면, 가상 레지스터 번호에 포함된 논리 레지스터 번호는 예측 레지스터 상태(84)에 의해 제공된 IQ# 및 PR# 중 하나를 선택하는 데에 이용된다. 반면에, 만일 가상 레지스터 번호가 발행 위치를 나타낸다면, 발행 위치 번호는 제어 유닛(82)에 의해 할당된 목적지 IQ#/PR# 중 하나를 선택하는 데에 이용된다.

도 8에 도시된 바와 같은 가상 레지스터 번호들을 기초로 하여 IQ#들 및 PR#들의 선택을 물리적으로 분리시킴으로써 장점을 얻을 수 있다. 비교적 많은 수의 버스들이 비교적 작은 양의 회로(즉, mux(160, 162, 164 및 168)로 표현되는 mux들)로 라우팅된다. 따라서, 가상/물리 레지스터 맵 유닛(86)에 의해 점유되는 영역은 예측 레지스터 상태(84) 및 IQ#/PR# 제어 유닛(82)으로부터의 버스에 의해 차지된다. IQ# 및 PR#는 개별적인 값들이기 때문에, 개별적인 mux들의 세트로 값들을 라우팅시키게 되면 가상/물리 레지스터 맵 유닛(86)에 의해 차지하는 영역을 감소시킨다. 일 실시예에서, IQ# 내의 비트들의 수 및 PR# 내의 비트들의 수는 거의 같다. 따라서, 가상/물리 레지스터 맵 유닛(86)의 한측 상에 IQ# 버스들을 와이어링하고, 가상/물리 레지스터 맵 유닛(86)의 다른측에 PR# 버스들을 와이어링하게 되면, 양측 상의 와이어링 공간이 효율적으로 이용되는 완전한 대칭 배치가 된다. 또한, 명령큐들(36A-36B)은 (언제 현재의 ROP들이 의존하는 ROP들이 완료되고, 이에 따라 ROP들의 현재의 세트중 하나가 스케줄링될 수 있는지를 결정하기 위하여 IQ#들을 수신하는) 스케줄링 부분 및 PR#들 및 명령 실행에 이용되는 다른 정보를 저장하지만, IQ#들은 저장하지 않는) 명령 저장장치로 물리적으로 분할될 수 있다. 따라서, 명령큐들(36A-36B)은 IQ# mux들 근처에서는 스케줄링 부분으로, 그리고 PR# mux들 근처에서는 명령 저장 부분으로 물리적으로 구성될 수 있어, 영역을 감소시킬 수 있게 된다.

도 8에 도시된 가상/물리 레지스터 맵 유닛(86)은 또한 예외 조건들에 응답하여 예측 레지스터 상태(84)에 대한 다음 예측 레지스터 상태의 맵을 처리한다. 트랩 IQ# mux(170)는 트랩들이 실행시, mux(162)에 의해 제공된 다음 예측 상태를 무효화하기 위하여 맵 사일로(32)로부터의 복구 예측 레지스터 상태에 대응하는 IQ#들을 라우팅시키는 것으로 인식될 때 이용된다. PC 사일로(48)는 mux(170)에 선택 제어 신호로서 트랩 제어 신호를 제공한다. 유사하게, 복구 예측 레지스터 상태 내의 PR#는 트랩 신호에 응답하여 트랩 PR# mux(172)를 통해 선택될 수 있다. 주목할 사항으로서, 다른 실시예들에서는, 예외 조건들로부터 복구하는 다른 방법들이 이용될 수 있다. 예를 들어, 예외 조건들은 퇴거에 대해 처리될 수 있다.

이제 도 9를 참조하면, 맵 사일로(32) 내의 엔트리(즉, ROP들의 라인에 대응하는 엔트리)에 저장된 정보를 나타낸 표(180)가 도시된다. 다른 실시예들이 또한 가능하며 고려된다.

표(180)에 나타낸 바와 같이, 라인에 대응하는 R#는 R# (라인 부분) 필드에 저장된다. 저장된 R#는 PC 사일로(48)에 의해 ROP들에 할당되는 R#들의 라인 부분이다. 라인 부분은 라인 내의 각 ROP에 대해 동일하며, R#의 오프셋 부분은 특정한 ROP의 라인 내의 발행 위치를 식별한다. 따라서, (총괄적으로 라인에 대응하는) 사일로 엔트리는 ROP에 대한 R#의 라인 부분과 저장된 R#를 비교함으로써 예외를 경험하는 ROP와 결합될 수 있다.

또한, 라인 내의 어떤 ROP들이 유효하다는 표시가 맵 사일로 엔트리의 라인 필드 내의 유효한 ROP들에 저장된다. 예를 들어, 이 표시는 ROP에 대한 비트가 될 수 있다. 만일 비트가 세트된다면, 라인 내의 대응하는 ROP가 유효해진다. 만일 비트가 제거된다면, 라인 내의 대응하는 ROP가 무효로 된다. 또한, ROP들중 어느 것이 목적지 논리 레지스터를 갖는다는 표시가 ROP 레지스터 기입 필드에 저장된다. 또한, 이 표시는 ROP에 대한 비트가 될 수 있다. 만일 비트가 세트된다면, 라인 내의 대응하는 ROP는 목적지 레지스터를 갱신한다. 만일 비트가 제거된다면, 라인 내의 대응하는 ROP는 목적지 레지스터를 갱신하지 않을 것이다. ROP들중 어느 것이 목적지 레지스터를 갖는다는 표시는 할당된 PR#들 및 할당된 IQ#들이기에 설명되는 예외의 경우 복구 상태의 일부가 된다는 결정을 하는 데에 이용된다.

목적지 레지스터들을 갖는 ROP들에 할당된 PR#들 및 IQ#들은 각각 엔트리의 할당된 PR# 및 할당된 IQ#에 유지된다. 또한, 각 목적지 레지스터의 논리 레지스터 번호는 논리 레지스터 번호 필드에 저장된다. 논리 레지스터 번호들은 복구 예측 상태 내의 논리 레지스터가 하기에서 설명되는 바와 같이 할당된 PR#들 및 할당된 IQ#들을 수신할 것인 지를 결정하는 데에 이용된다. 또한, 라인의 성공적인 퇴거시, 논리 레지스터 번호들 및 대응하는 PR#들은 구조 재명명 블록(34)에 저장을 위하여 전달된다.

라인 내의 어느 ROP들이 조건 코드 레지스터를 갱신한다는 표시가 CC 기입 필드에 저장된다. 개별적으로 갱신되는 조건 코드의 각 부분은 CC 기입 필드내의 비트로 표시될 수 있으며, 비트들의 세트는 ROP 레지스터 기입 필드내에 표시된 각 레지스터 기입과 결합될 수 있다. (할당된 PR# 및 IQ# 필드들에 저장된) 대응하는 ROP의 PR# 및 IQ#는 예외 검출시 예측 레지스터 상태 내의 예측 조건 코드 레지스터를 복구하는 데에 이용될 수 있다. 또한, CC 기입 필드는, 할당된 PR#들 내의 PR#들이 ROP들의 성공적인 퇴거시 구성된 조건 코드 상태에 대응함을 표시한다. 상기 설명한 바와 같이, 본 실시예에서, 단일의 물리 레지스터는 정수 레지스터 갱신 및 대응하는 조건 코드 갱신들을 저장하는 데에 이용된다.

맵 사일로 엔트리에 대응하는 ROP들의 라인을 디스패치하기 전에 예측 레지스터 상태(84) 내에 저장되는 현재의 예측 레지스터 상태는 엔트리의 현재 예측 레지스터 상태 필드에 저장된다. 현재의 예측 레지스터 상태는 라인 내에서의 예외의 경우 예측 레지스터 상태(84)를 복구하기 위한 기초 역할을 한다.

궁극적으로, FP TOS 및 각 발행 위치에 대응하는 유효 비트들은 엔트리의 FP TOS 및 유효 비트 필드 내에 저장된다. FP TOS 및 예외를 경험하는 ROP에 대응하는 유효 비트들은 예외 검출시 예측 FP TOS 및 예측 레지스터 상태(84) 내의 태그 워드로 복구된다.

이제 도 10을 참조하면, 예시적인 예측 레지스터 상태 엔트리(182)의 블록도가 도시되는데, 이는 논리 레지스터에 대한 예측 레지스터의 일 실시예에 의해 이용될 수 있다. 엔트리(182)는 유효 필드(184), IQ# 필드(186), 및 PR# 필드(188)를 포함한다. 유효 필드(184)는 IQ# 필드(186)가 유효한지 아닌 지를 나타낸다. IQ#는 대응하는 ROP가 명령큐들(36A-36B)로부터 완료될 때 까지 유효하다. 따라서, 예측 상태(84)는 실행 코어(40) 및 적재/저장 유닛(42)으로부터 IQ#들이 완료되고 있다는 표시를 수신할 수 있다. 매치를 검출하게 되면, 예측 상태(84)는 유효 표시를 리셋시킬 수 있다. 유효 표시는, 예를 들어 세트된 경우에는 유효를, 제거된 경우에는 무효를 나타내는 비트가 될 수 있다. IQ# 필드(186)는 가장 최근에 갱신된 논리 레지스터인 ROP의 IQ#를 저장하고, PR# 필드(188)는 ROP에 목적지 레지스터로서 할당된 물리 레지스터의 PR#를 저장한다.

이제 도 11을 참조하면, PC 사일로(48)에 의해 신호화된 예외 조건에 응답하여 맵 사일로(32)의 동작을 나타낸 흐름도가 도시된다. 다른 실시예들이 또한 가능하며 고려된다. 도 11에 도시된 단계들이 이해의 용이를 위해 특정한 순서로 도시되기는 하였지만, 모든 적절한 순서가 이용될 수 있다. 또한, 단계들은 맵 사일로(32) 내에서 이용되는 결합 로직 내에서 동시에 수행될 수 있다.

단계(190)에 도시한 바와 같이, 맵 사일로(32)는 PC 사일로(48)에 의해 제공된 R#를 맵 사일로(32)에 저장된 엔트리들의 R#(라인 부분) 필드에 대해 캡시킨다. PC 사일로(48)에 의해 제공된 R#는 특정한 ROP를 식별한다. 그러나, 맵 사일로

(32)에 대한 체크의 목적으로, R#의 라인 부분이 캡된다. 제공된 R# 보다 더 최근의 엔트리들(즉, 프로그램 순서에 있어서 예외 이후의 명령들)은 맵 사일로(32) 내에서 삭제된다. 삭제된 엔트리들의 할당된 PR# 필드에 저장된 PR#는 자유롭게 사용가능해 진다. 일 실시예에서, 삭제된 엔트리들의 PR#들은 클록 주기당 한 엔트리의 속도로 다수의 클록 주기들을 통해 자유롭게 사용가능해 진다. 캡이 매치를 나타내는 사일로 엔트리는 선택된 맵 사일로 엔트리이다.

선택된 맵 사일로 엔트리에 저장된 현재의 예측 레지스터 상태(즉, 예외를 경험하는 ROP를 포함하는 ROP들의 라인 이전의 현재의 예측 상태는 현재의 예측 레지스터 상태(84) 내의 예측 상태로 복구된다(단계 192). 또한, 예측 레지스터 상태(84) 내의 FP TOS 및 유효 비트들은 예외를 경험하는 ROP의 발행 위치에 대해 선택된 맵 사일로 엔트리에 저장된 값으로 복구된다(단계 194).

선택된 맵 사일로 엔트리 내의 ROP 레지스터 기입 필드는 예외를 경험하는 ROP 이전의 기입들에 대해 마스크된다. 다시 말해, 예외를 경험하는 ROP 이후의 ROP 레지스터 기입들은 마스크 오프되어, 이들은 마스크 이후의 기입들로 보이지 않는다. (마스크 이후의) 남아있는 기입들은 각 레지스터에 대한 가장 최근의 기입을 검출하기 위하여 스캔된다(즉, 남아있는 기입들중 두 개 또는 그 이상이 동일한 레지스터라면, 좀 더 최근의 기입이 유지된다). 현재의 예측 상태는 이 결과들에 의해 갱신된다(단계 196). 주목할 사항으로서, 단계(192) 및 단계(196)는 복구 예측 레지스터 상태를 예측 레지스터 상태(84)로 전송하기 전에 맵 사일로(32)에서 수행될 수 있다. 변형예로서, 현재의 예측 레지스터 상태는 예측 레지스터 상태(84)로 복구되며, 이후 단계(196)에 의해 갱신된다.

또한, 선택된 맵 사일로 엔트리 내의 ROP 레지스터 기입 필드(즉, 단계(196)의 마스크 이전의 초기 값들)는 예외를 경험하는 ROP 이후의 레지스터 기입들에 대해 마스크된다. 다시 말해, 예외를 경험하는 ROP 이전의 레지스터 기입들은 마스크 오프된다. 남아있는 레지스터 기입들의 PR#들은 삭제된다(단계 198). 도 11에 도시된 흐름도는 유익하게는 예외에 응답하여 예측 상태를 복구하는 신속한 방법을 제공한다.

주목할 사항으로서, 발신지 가상 레지스터 번호 및 가상 발신지 레지스터 번호라는 용어는 상기에서 이용되었다. 이들 용어는 동일한 의미를 갖는다. 또한, 주목할 사항으로서, 상기 설명한 바와 같이, 프로세서(10)에 의해 이용되는 명령 세트 구조에서 지정된 각 명령이 단일 명령 동작을 맵시키는 실시예들은 본원에서 규정된 명령 동작의 의미 내에서 고려된다.

이제 도 12를 참조하면, 버스 브리지(202)를 통해 다양한 시스템 구성요소들에 결합된 프로세서(10)를 포함하는 컴퓨터 시스템(200)의 일 실시예의 블록도가 도시된다. 다른 실시예들이 또한 가능하며 고려된다. 도시된 시스템에서, 주 메모리(204)는 메모리 버스(206)를 통해 버스 브리지(202)에 결합되며, 그래픽 제어기(208)는 AGP 버스(210)를 통해 버스 브리지(202)에 결합된다. 궁극적으로, 다수의 PCI 장치들(212A-212B)은 PCI 버스(214)를 통해 버스 브리지(202)에 결합된다. 제 2 버스 브리지(216)가 또한 제공되어, EISA/ISA 버스(220)를 통해 하나 또는 그 이상의 EISA 또는 ISA 장치들(218)로의 전기적인 인터페이스를 제공한다. 프로세서(10)는 외부 인터페이스(52)를 통해 버스 브리지(202)에 결합된다.

버스 브리지(202)는 프로세서(10), 주 메모리(204), 그래픽 제어기(208), 및 PCI 버스(214)에 부착된 장치들 간의 인터페이스를 제공한다. 버스 브리지(202)에 결합된 장치들 중 하나로부터 동작이 수신될 때, 버스 브리지(202)는 연산의 타겟을 식별한다(예를 들어 특정한 장치, 또는 PCI 버스(214)의 경우에는 그 타겟이 PCI 버스(214) 위에 있는 지를 식별한다). 버스 브리지(202)는 연산을 타겟 장치에 라우팅시킨다. 버스 브리지(202)는 일반적으로 상기 연산을 발신지 장치 또는 버스에 의해 이용되는 프로토콜로부터 타겟 장치 또는 버스에 의해 이용되는 프로토콜로 변환한다.

PCI 버스(214)에 대한 ISA/EISA 버스에 인터페이스를 제공하는 것에 부가하여, 제 2 버스 브리지(216)는 또한 바람직한 경우 부가적인 기능을 통합할 수 있다. 예를 들어, 일 실시예에서, 제 2 버스 브리지(216)는 PCI 버스(214)의 소유권을 조정하기 위한 마스터 PCI 조정기(미도시)를 포함한다. 제 2 버스 브리지(216) 외부의, 또는 이와 통합된 입/출력 제어기(미도시)가 또한 컴퓨터 시스템(200) 내에 포함될 수 있는데, 이는 바람직한 경우, 키보드 및 마우스(222)와 직렬 및 병렬 포트들의 작동을 지원한다. 다른 실시예들에서는 또한, 프로세서(10)와 버스 브리지(202) 사이의 외부 인터페이스(52)에 외부 캐시 유닛(미도시)이 결합될 수 있다. 변형예로서, 외부 캐시가 버스 브리지(202)에 결합될 수 있으며, 외부 캐시에 대한 캐시 제어 로직이 버스 브리지(202) 내에 통합될 수 있다.

주 메모리(204)는 응용 프로그램이 저장되고 프로세서(10)가 주로 작동하게 되는 메모리이다. 적절한 주 메모리(204)는 DRAM(동적 임의 접근 메모리) 및 바람직하게는 다수의 SDRAM(동기 DRAM) 뱅크들을 포함한다.

PCI 장치들(212A-212B)은, 예를 들어 가령 네트워크 인터페이스 카드들, 비디오 액셀러레이터들, 오디오 카드들, 하드 또는 플로피 디스크 드라이브들 또는 드라이브 제어기들, SCSI(소형 컴퓨터 시스템 인터페이스) 어댑터들 및 전화 카드들과 같은 다양한 주변 장치들의 예이다. 유사하게, ISA 장치(218)는, 가령 모뎀; 사운드 카드; 및 가령 GPIB 또는 필드 버스 인터페이스 카드들과 같은 다양한 데이터 획득 카드들과 같은 다양한 형태의 주변 장치들의 예이다.

그래픽 제어기(208)는 디스플레이(226) 상에서의 텍스트 및 이미지들의 렌더링을 제어하기 위하여 제공된다. 그래픽 제어기(208)는 주 메모리(204) 내로, 그리고 주 메모리(204)로부터 효과적으로 시프트될 수 있는 3차원 데이터 구조들을 렌더링하는 것으로 종래 기술에 널리 공지되어 있는 전형적인 그래픽 액셀러레이터를 구현할 수 있다. 따라서, 그래픽 제어기(208)는, 이것이 버스 브리지(202) 내의 타겟 인터페이스로의 접속을 요청 및 수신하여 주 메모리(204)로 접속할 수 있다는 점에서 AGP 버스(210)의 마스터가 될 수 있다. 전용 그래픽 버스는 주 메모리(204)로부터 데이터를 신속하게 검색할 수 있게 한다. 어떠한 연산들에 대해, 그래픽 제어기(208)는 또한 AGP 버스(210) 상에서 PCI 프로토콜 트랜잭션을 발생 시키도록 구성될 수 있다. 따라서, 버스 브리지(202)의 AGP 인터페이스는 PCI 프로토콜 타겟 및 이니시에이터 트랜잭션들 뿐만 아니라 AGP 프로토콜 트랜잭션 등을 지원하는 기능을 포함할 수 있다. 디스플레이(226)는 이미지 또는 텍스트가 제시될 수 있는 모든 전자 디스플레이이다. 적절한 디스플레이(226)로는 음극선관("CRT"), 액정표시소자("LCD") 등이 있다.

주목할 사항으로서, AGP, PCI 및 ISA 또는 EISA 버스들을 예로 들기는 하였지만, 바람직한 모든 버스 구조들로 대체될 수 있다. 또한, 주목할 사항으로서, 컴퓨터 시스템(200)은 부가적인 프로세서들(예를 들어, 컴퓨터 시스템(200)의 선택적인 구성요소로서 나타낸 프로세서(10a))를 포함하는 멀티프로세싱 컴퓨터 시스템이 될 수 있다. 프로세서(10a)는 프로세서(10)와 유사할 수 있다. 좀 더 명확하게 설명하면, 프로세서(10a)는 프로세서(10)의 동일한 복사본이 될 수 있다. 프로세서(10a)는 (도 12에 도시된 바와 같이) 프로세서(10)와 외부 인터페이스(52)를 공유할 수 있으며, 또는 독립적인 버스를 통해 버스 브리지(202)에 연결될 수 있다.

상기 설명에 따르면, 프로세서는 레지스터 재명명 기술을 이용하는 것으로 설명되었다. 일 실시예에서, 재명명 기술은 가상 레지스터 번호를 이용하여 스테이지들로 분배된다. 유익하게는, 이 기술을 이용하여 고주파수 연산이 가능할 수 있다. 다른 실시예에서, 예외들로부터의 신속한 복구는 각각의 ROP들에 대응하는 현재의 예측 상태를 격납하고, 라인 내에 레지스터 기입들을 노트함으로써 수행된다. 또 다른 실시예에서는, 재명명 레지스터들의 자유화가 수행되어, 동일한 재명명 레지스터가 하나 이상의 논리 레지스터에 대응할 수 있게 한다. 이로써, 물리적인 레지스터가 효율적으로 이용될 수 있게 된다.

상기 설명된 실시예들에 대한 다양한 변형들 및 변경들이 이루어질 수 있다. 본 발명은 첨부된 청구항들의 범위 내에 포함되는 상기 모든 변형들 및 변경들을 포함한다.

(57) 청구의 범위

청구항 1.

프로세서 내의 물리 레지스터를 관리하는 방법으로서,

제 1 물리 레지스터를 제 1 명령 동작의 목적지 오퍼랜드에 할당하는 단계와, 여기서 상기 오퍼랜드는 논리 레지스터를 식별하고;

상기 제 1 명령 동작을 퇴거하는 단계와;

상기 퇴거에 응답하여 상기 제 1 물리 레지스터에 대응하는 제 1 물리 레지스터 번호로 구조 재명명 블록을 갱신하는 단계와, 여기서 상기 제 1 물리 레지스터 번호는 레지스터 파일 내의 상기 제 1 물리 레지스터를 식별하며;

상기 퇴거에 응답하여 상기 구조 재명명 블록으로부터 제 2 물리 레지스터 번호를 읽는 단계와, 여기서 상기 제 2 물리 레지스터 번호는 상기 제 1 물리 레지스터 번호로 인해 상기 구조 재명명 블록 내의 상기 논리 레지스터를 나타내던 것으로부터 변위(displacement)되며;

상기 갱신에 이어서, 상기 제 2 물리 레지스터 번호가 또한 상기 구조 재명명 블록 내의 제 2 논리 레지스터를 나타내는지 결정하는 단계와; 그리고

상기 제 2 물리 레지스터 번호가 또한 상기 구조 재명명 블록 내의 상기 제 2 논리 레지스터를 나타낸다는 결정에 응답하여, 상기 제 2 물리 레지스터 번호에 대응하는 제 2 물리 레지스터를 자유화하는 것을 금지하는 단계를 포함하며, 상기 제 2 물리 레지스터 번호는 상기 레지스터 파일 내의 상기 제 2 물리 레지스터를 식별하는 것을 특징으로 하는 방법.

청구항 2.

제 1 항에 있어서,

상기 제 2 물리 레지스터 번호가 상기 구조 재명명 블록 내의 상기 제 2 논리 레지스터를 나타내지 않는다는 결정에 응답하여, 상기 제 2 물리 레지스터를 자유화하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 3.

제 2 항에 있어서,

상기 제 2 물리 레지스터 번호가 또한 상기 구조 재명명 블록 내의 제 2 논리 레지스터를 나타낸다는 상기 결정을 수행하기 위해, 상기 갱신 이후, 상기 구조 재명명 블록 내에 저장된 상기 물리 레지스터 번호들과 상기 제 2 물리 레지스터 번호를 비교하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 4.

제 3 항에 있어서,

상기 금지 단계는, 상기 제 2 물리 레지스터 번호에 대해 적어도 하나의 매치를 검출하는 상기 비교 단계에 응답하여 수행되는 것을 특징으로 하는 방법.

청구항 5.

제 4 항에 있어서,

상기 자유화 단계는 상기 제 2 물리 레지스터 번호에 대해 어떠한 매치도 없음을 검출하는 상기 비교 단계에 응답하여 수행되는 것을 특징으로 하는 방법.

청구항 6.

제 1 항에 있어서,

상기 할당 단계에 응답하여, 상기 제 1 물리 레지스터 번호를 사일로에 저장하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 7.

제 6 항에 있어서,

상기 퇴거 단계는 상기 제 1 물리 레지스터 번호가 상기 사일로의 헤드에 도달할 시에 수행되는 것을 특징으로 하는 방법.

청구항 8.

제 1 항에 있어서,

프로그램 순서에서 상기 제 1 명령 동작 이후의 제 2 명령 동작을 퇴거하는 단계와, 여기서 상기 제 2 명령 동작은 자신의 목적지 레지스터에 할당된 제 3 물리 레지스터를 갖고, 상기 목적지 레지스터는 상기 제 2 논리 레지스터를 포함하며;

상기 퇴거 단계에 응답하여 상기 제 3 물리 레지스터에 대응하는 제 3 물리 레지스터 번호로 상기 구조 재명명 블록을 갱신하는 단계와;

상기 퇴거 단계에 응답하여 상기 구조 재명명 블록으로부터 상기 제 2 물리 레지스터 번호를 읽는 단계와; 그리고

상기 제 2 물리 레지스터 번호를 자유화하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 9.

제 1 물리 레지스터를 명령 동작의 목적지 오퍼랜드에 할당하도록 구성된 맵 유닛과, 여기서 상기 맵 유닛은 또한 어느 물리 레지스터들이 할당에 대해 자유화되는 지를 나타내는 자유화 목록을 유지하도록 구성되고, 상기 맵 유닛은 상기 자유화 목록으로부터 상기 제 1 물리 레지스터를 선택하도록 구성되며; 그리고

상기 명령 동작이 퇴거되고 있다는 표시 및 상기 제 1 물리 레지스터에 대응하는 제 1 물리 레지스터 번호를 수신하도록 결합된 구조 재명명 블록과, 여기서 상기 제 1 물리 레지스터 번호는 레지스터 파일 내의 상기 제 1 물리 레지스터를 식별하고, 상기 구조 재명명 블록은 상기 명령 동작의 퇴거 이전에 상기 목적지 오퍼랜드에 의해 식별되는 제 1 논리 레지스터에 대응하는 제 2 물리 레지스터를 식별하는 제 2 물리 레지스터 번호를 제공하도록 구성되며, 상기 구조 재명명 블록은 상기 명령 동작이 퇴거되고 있다는 표시에 응답하여 상기 제 2 물리 레지스터 번호를 제공하고, 상기 제 2 물리 레지스터 번호는 상기 레지스터 파일 내의 상기 제 2 물리 레지스터를 식별하며, 상기 구조 재명명 블록은 상기 제 1 물리 레지스터 번호를 포획하고, 상기 명령 동작이 퇴거되고 있다는 상기 표시에 응답하여 상기 제 1 물리 레지스터 번호를 상기 제 1 논리 레지스터에 대응하는 것으로서 유지하도록 구성되며, 그리고 상기 구조 재명명 블록은 또한 상기 제 2 물리 레지스터 번호가 또한 제 2 논리 레지스터에 대응하는지의 여부를 결정하도록 구성되며;

상기 맵 유닛은 상기 제 2 물리 레지스터가 상기 제 2 논리 레지스터를 나타내는 경우, 상기 제 2 물리 레지스터 번호를 상기 자유화 목록에 추가하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 10.

제 9 항에 있어서,

상기 맵 유닛은 상기 제 2 물리 레지스터가 상기 제 2 논리 레지스터를 나타내지 않는 경우, 상기 물리 레지스터 번호를 상기 자유화 목록에 추가하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 11.

제 9 항에 있어서,

상기 구조 재명명 블록은, 상기 제 1 물리 레지스터 번호를 포획한 이후 상기 구조 재명명 블록 내의 상기 물리 레지스터 번호들과 상기 제 2 물리 레지스터 번호를 비교함으로써, 상기 제 2 물리 레지스터 번호가 추가적으로 상기 제 2 논리 레지스터에 대응하는지의 여부를 결정하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 12.

제 11 항에 있어서,

상기 구조 재명명 블록은 내용 주소화 메모리(content addressable memory)를 포함하는 것을 특징으로 하는 프로세서.

청구항 13.

제 9 항에 있어서,

상기 맵 유닛에 결합된 사일로를 더 포함하고,

상기 사일로는 상기 맵 유닛에 의해 상기 명령 동작에 대해 할당될 때 상기 제 1 물리 레지스터 번호를 저장하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 14.

제 13 항에 있어서,

상기 사일로는 또한 상기 구조 재명명 블록에 결합되고, 상기 사일로는 상기 명령 동작의 퇴거시 상기 제 1 물리 레지스터 번호를 제공하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 15.

프로세서에서 물리 레지스터를 관리하기 위한 방법으로서,

제 1 물리 레지스터를 명령 동작의 목적지 오퍼랜드에 할당하는 단계와, 여기서 상기 목적지 오퍼랜드는 논리 레지스터를 식별하고;

상기 명령 동작을 퇴거하는 단계와;

상기 퇴거에 응답하여 상기 제 1 물리 레지스터에 대응하는 제 1 물리 레지스터 번호와 구조 재명명 블록을 갱신하는 단계와, 여기서 상기 제 1 물리 레지스터 번호는 레지스터 파일 내의 상기 제 1 물리 레지스터를 식별하고;

상기 퇴거에 응답하여 상기 구조 재명명 블록으로부터 제 2 물리 레지스터에 대응하는 제 2 물리 레지스터 번호를 읽는 단계와, 여기서 상기 제 2 물리 레지스터 번호는 상기 제 1 물리 레지스터 번호로 인해 상기 구조 재명명 블록 내의 상기 논리 레지스터를 나타내던 것으로부터 변위되고, 상기 제 2 물리 레지스터 번호는 상기 레지스터 파일 내의 상기 제 2 물리 레지스터를 식별하며; 그리고

상기 제 2 물리 레지스터를 자유화하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 16.

제 1 물리 레지스터를 명령 동작의 목적지 오퍼랜드에 할당하도록 구성된 맵 유닛과, 여기서 상기 맵 유닛은 또한 어느 물리 레지스터들이 할당에 대해 자유화되는 지를 나타내는 자유화 목록을 유지하도록 구성되고, 상기 맵 유닛은 상기 자유화 목록으로부터 상기 제 1 물리 레지스터를 선택하도록 구성되며; 그리고

상기 명령 동작이 되거되고 있다는 표시 및 상기 제 1 물리 레지스터에 대응하는 제 1 물리 레지스터 번호를 수신하도록 결합된 구조 재명명 블록을 포함하며, 상기 제 1 물리 레지스터 번호는 레지스터 파일 내의 상기 제 1 물리 레지스터를 식별하고, 상기 구조 재명명 블록은 상기 명령 동작의 되거 이전에 상기 목적지 오퍼랜드에 의해 식별되는 제 1 논리 레지스터에 대응하는 제 2 물리 레지스터를 식별하는 제 2 물리 레지스터 번호를 제공하도록 구성되고, 상기 구조 재명명 블록은 상기 명령 동작이 되거되고 있다는 상기 표시에 응답하여 상기 제 2 물리 레지스터 번호를 제공하며, 상기 제 2 물리 레지스터 번호는 상기 레지스터 파일 내의 상기 제 2 물리 레지스터를 식별하고, 그리고 상기 구조 재명명 블록은 상기 제 1 물리 레지스터 번호를 포획하고, 상기 명령 동작이 되거되고 있다는 상기 표시에 응답하여 상기 제 1 물리 레지스터 번호를 상기 제 1 논리 레지스터에 대응하는 것으로서 유지하도록 구성되며,

상기 맵 유닛은 상기 제 2 물리 레지스터 번호를 상기 자유화 목록에 부가하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 17.

프로세서와 상기 프로세서에 결합된 주변 장치를 포함하는 컴퓨터 시스템으로서,

상기 프로세서는,

제 1 물리 레지스터를 명령 동작의 목적지 오퍼랜드에 할당하도록 구성된 맵 유닛과, 여기서 상기 맵 유닛은 또한 어느 물리 레지스터들이 할당에 대해 자유화되는 지를 나타내는 자유화 목록을 유지하도록 구성되고, 상기 맵 유닛은 상기 자유화 목록으로부터 상기 제 1 물리 레지스터를 선택하도록 구성되며; 그리고

상기 명령 동작이 되거되고 있다는 표시 및 상기 제 1 물리 레지스터에 대응하는 제 1 물리 레지스터 번호를 수신하도록 결합된 구조 재명명 블록을 포함하며, 상기 제 1 물리 레지스터 번호는 레지스터 파일 내의 상기 제 1 물리 레지스터를 식별하고, 상기 구조 재명명 블록은 상기 명령 동작의 되거 이전에 상기 목적지 오퍼랜드에 의해 식별되는 제 1 논리 레지스터에 대응하는 제 2 물리 레지스터를 식별하는 제 2 물리 레지스터 번호를 제공하도록 구성되고, 상기 구조 재명명 블록은 상기 명령 동작이 되거되고 있다는 상기 표시에 응답하여 상기 제 2 물리 레지스터 번호를 제공하며, 상기 제 2 물리 레지스터 번호는 상기 레지스터 파일 내의 상기 제 2 물리 레지스터를 식별하고, 그리고 상기 구조 재명명 블록은 상기 제 1 물리 레지스터 번호를 포획하고, 상기 명령 동작이 되거되고 있다는 상기 표시에 응답하여 상기 제 1 물리 레지스터 번호를 상기 제 1 논리 레지스터에 대응하는 것으로서 유지하도록 구성되며,

상기 맵 유닛은 상기 제 2 물리 레지스터 번호를 상기 자유화 목록에 부가하도록 구성되며, 그리고

상기 주변 장치는 상기 컴퓨터 시스템과 다른 컴퓨터 시스템 간에 통신하도록 구성되는 것을 특징으로 하는 컴퓨터 시스템.

청구항 18.

제 17 항에 있어서,

상기 구조 재명명 블록은 또한 상기 제 2 물리 레지스터 번호가 추가적으로 제 2 논리 레지스터에 대응하는 지의 여부를 결정하도록 구성되고, 상기 맵 유닛은 상기 제 2 물리 레지스터가 상기 제 2 논리 레지스터를 나타내는 경우 상기 제 2 물리 레지스터 번호를 상기 자유화 목록에 부가하는 것을 금지하도록 구성되는 것을 특징으로 하는 컴퓨터 시스템.

청구항 19.

제 17 항에 있어서, 제 2 프로세서를 더 포함하고,

상기 제 2 프로세서는,

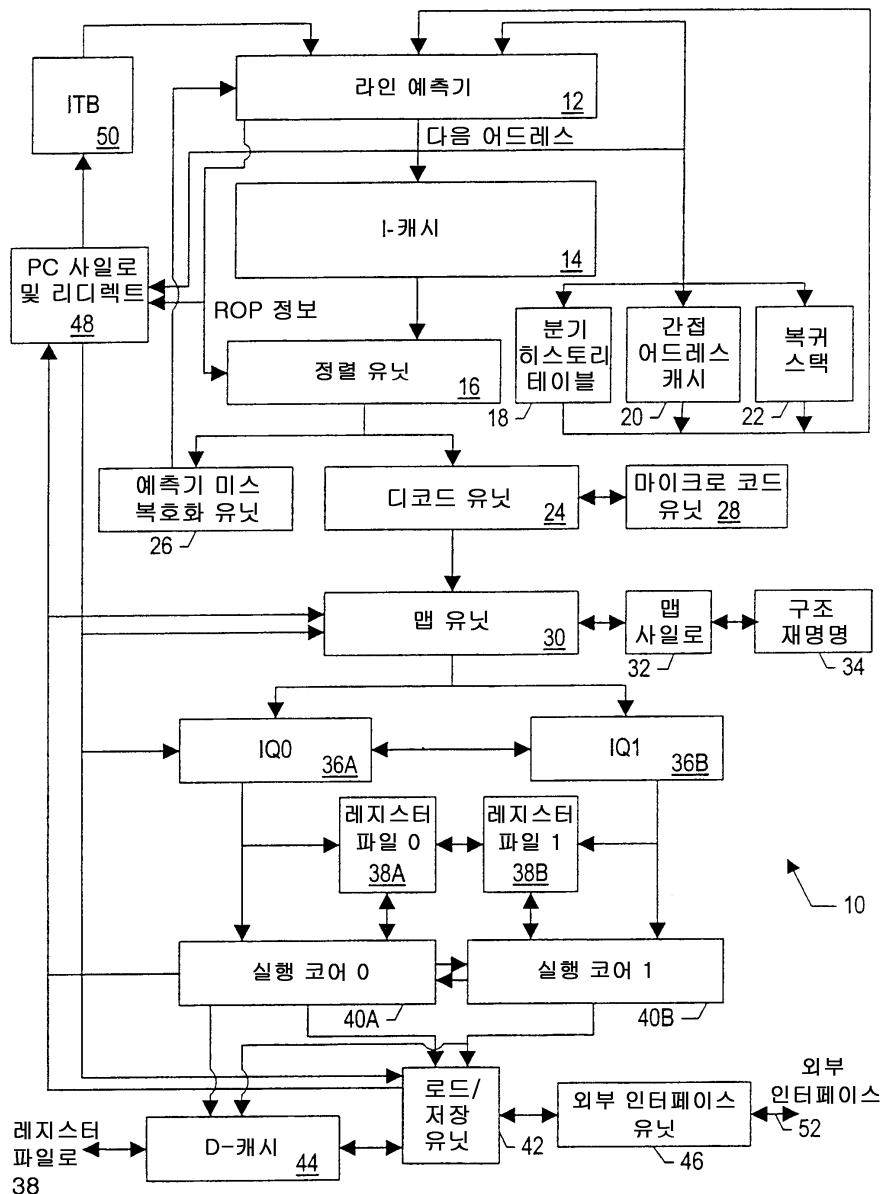
제 1 물리 레지스터를 명령 동작의 목적지 오퍼랜드에 할당하도록 구성된 맵 유닛과, 여기서 상기 맵 유닛은 또한 어느 물리 레지스터들이 할당을 위해 자유화되는 지를 나타내는 자유화 목록을 유지하도록 구성되고, 상기 맵 유닛은 상기 자유화 목록으로부터 상기 제 1 물리 레지스터를 선택하도록 구성되고; 그리고

상기 명령 동작이 되거되고 있다는 표시 및 상기 제 1 물리 레지스터에 대응하는 제 1 물리 레지스터 번호를 수신하도록 결합된 구조 재명명 블록을 포함하며, 상기 구조 재명명 블록은 상기 명령 동작의 되거 이전에 상기 목적지 오퍼랜드에 의해 식별되는 제 1 논리 레지스터에 대응하는 제 2 물리 레지스터를 식별하는 제 2 물리 레지스터 번호를 제공하도록 구성되고, 상기 구조 재명명 블록은 상기 명령 동작이 되거되고 있다는 상기 표시에 응답하여 상기 제 2 물리 레지스터 번호를 제공하며, 그리고 상기 구조 재명명 블록은 상기 제 1 물리 레지스터 번호를 포획하고, 상기 명령 동작이 되거되고 있다는 상기 표시에 응답하여 상기 제 1 물리 레지스터 번호를 상기 제 1 논리 레지스터에 대응하는 것으로서 유지하도록 구성되며,

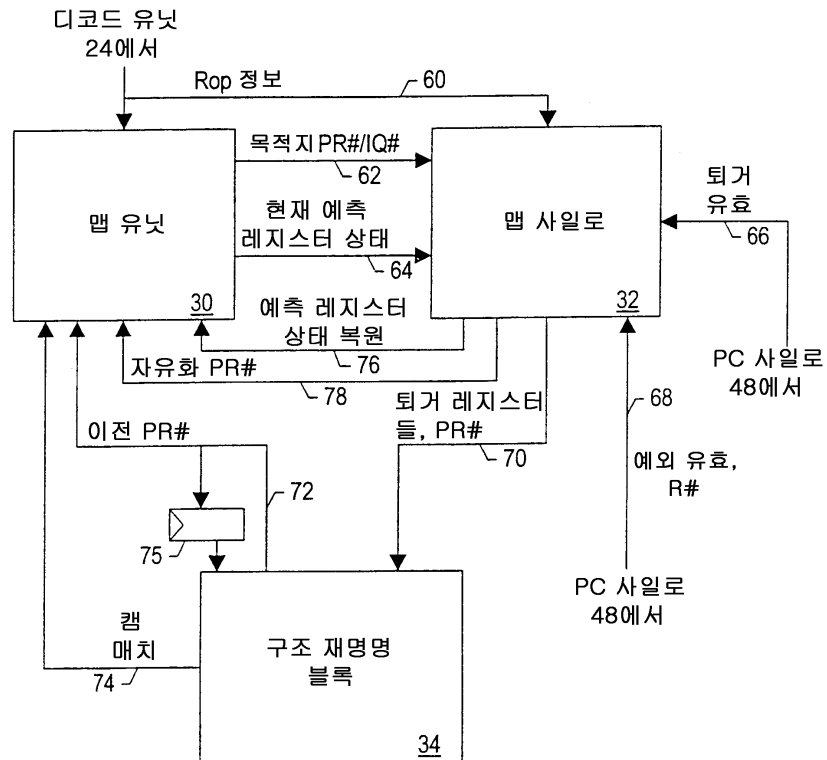
상기 맵 유닛은 상기 제 2 물리 레지스터 번호를 상기 자유화 목록에 추가하도록 구성되는 것을 특징으로 하는 컴퓨터 시스템.

도면

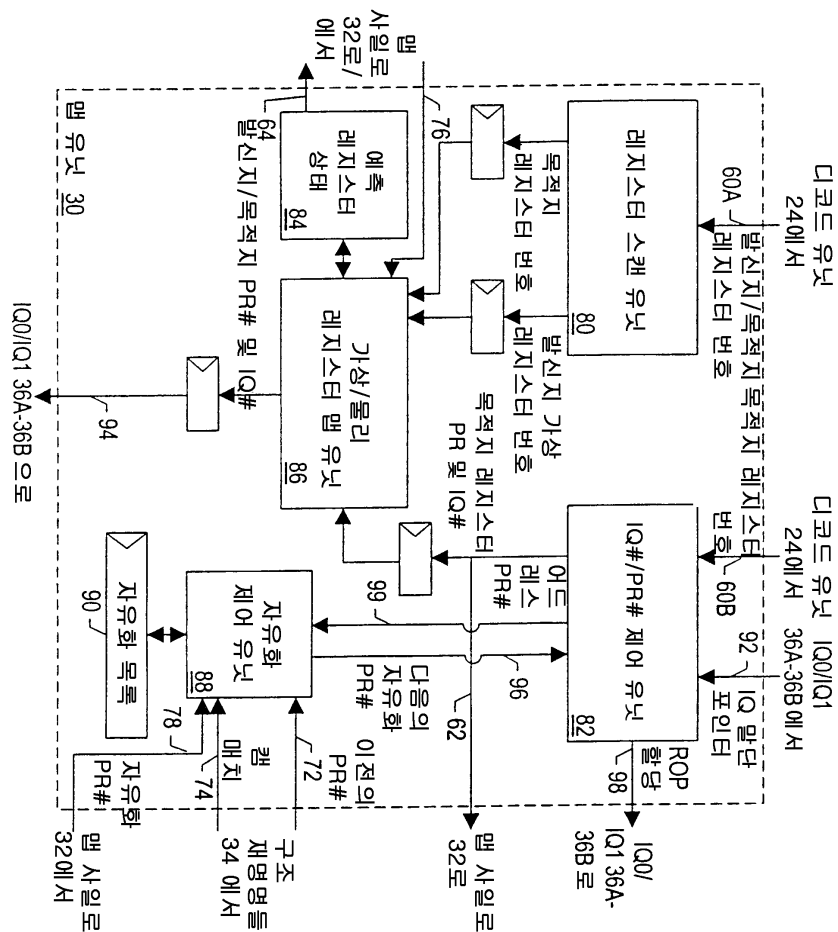
도면1



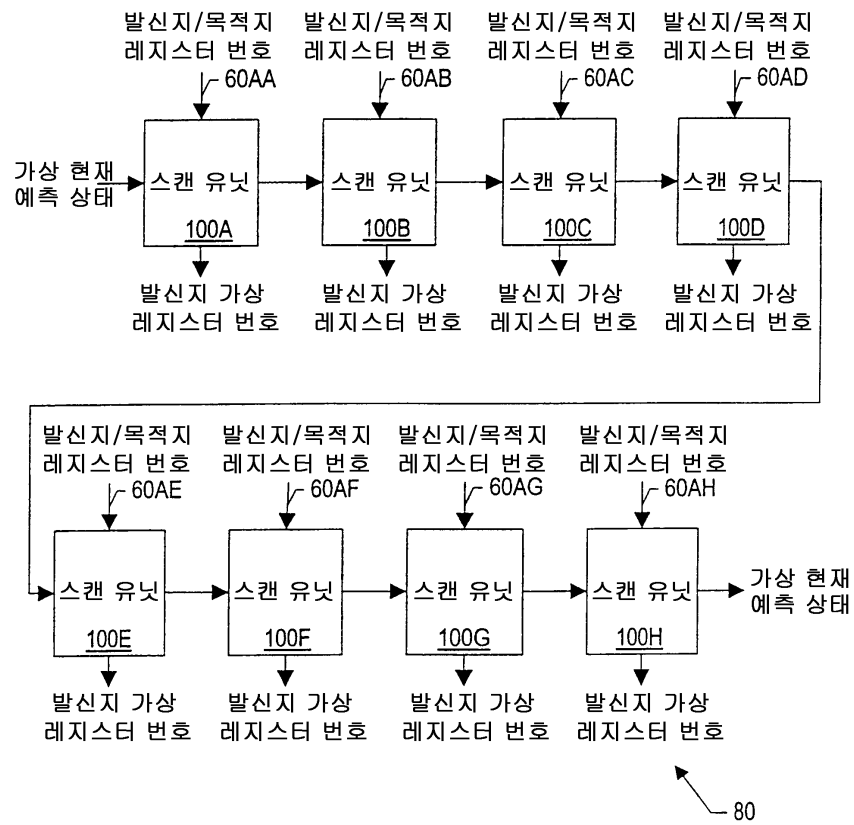
도면2



도면3



도면4

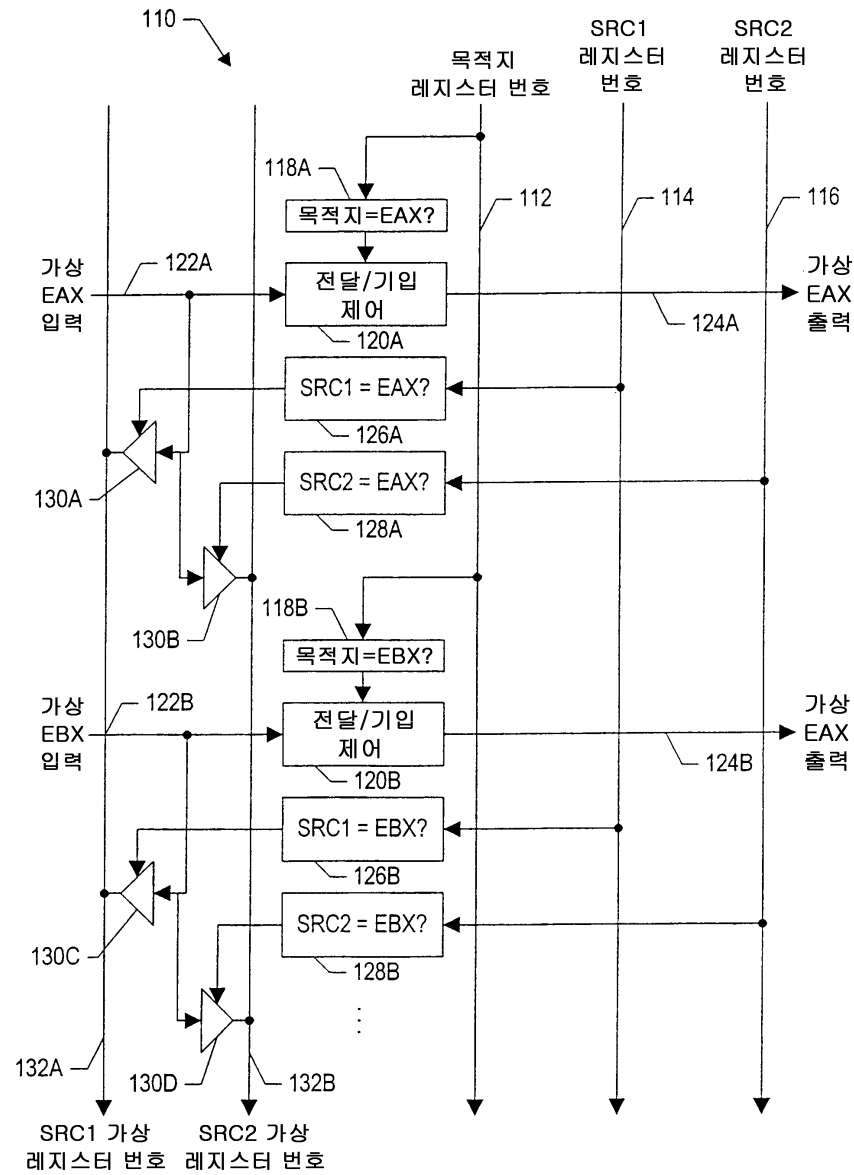


도면5

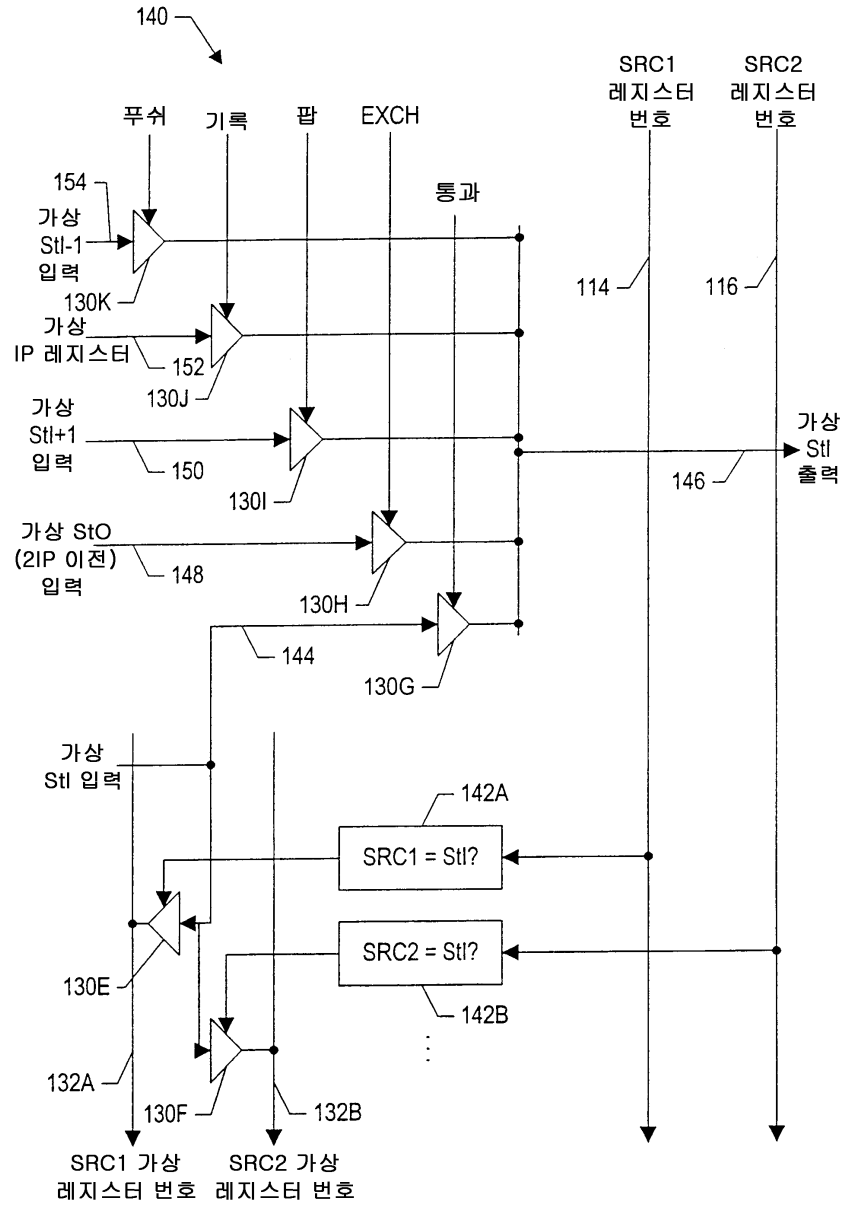
MSB	LSBs
0	논리 레지스터 번호
1	발행 위치 번호

102

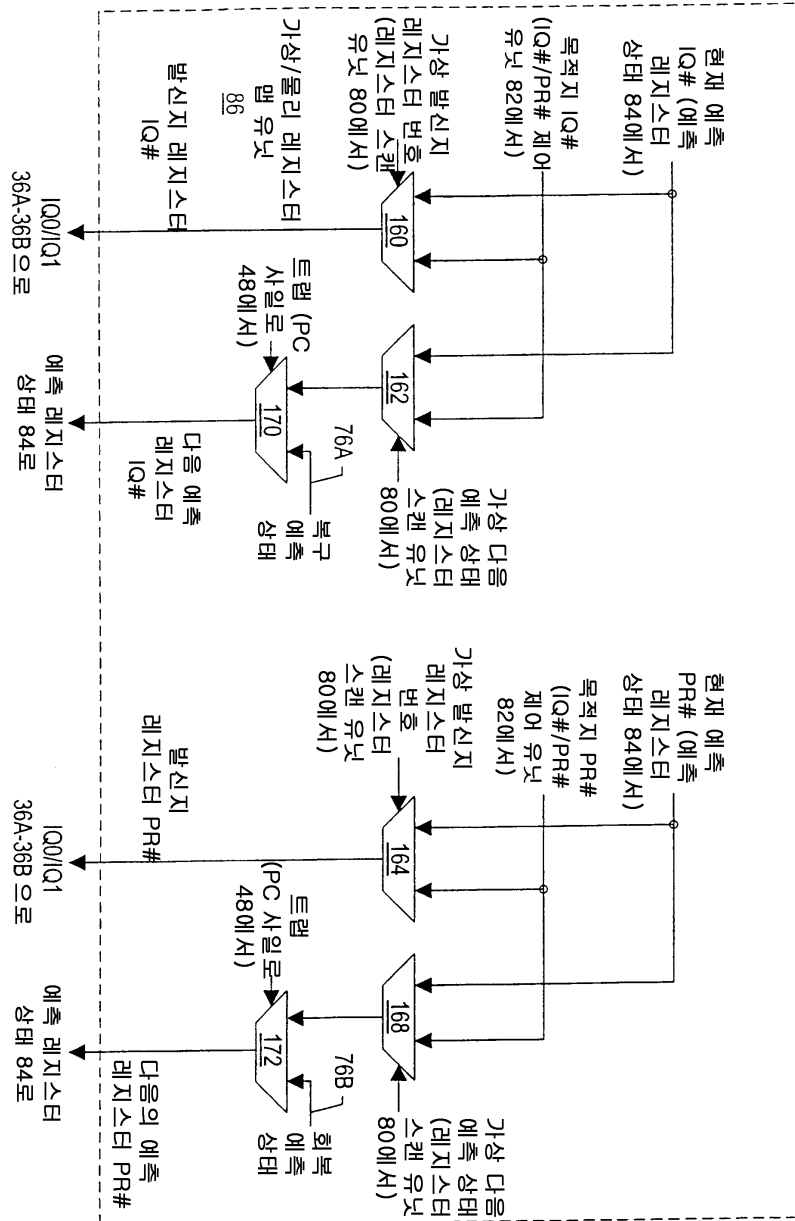
도면6



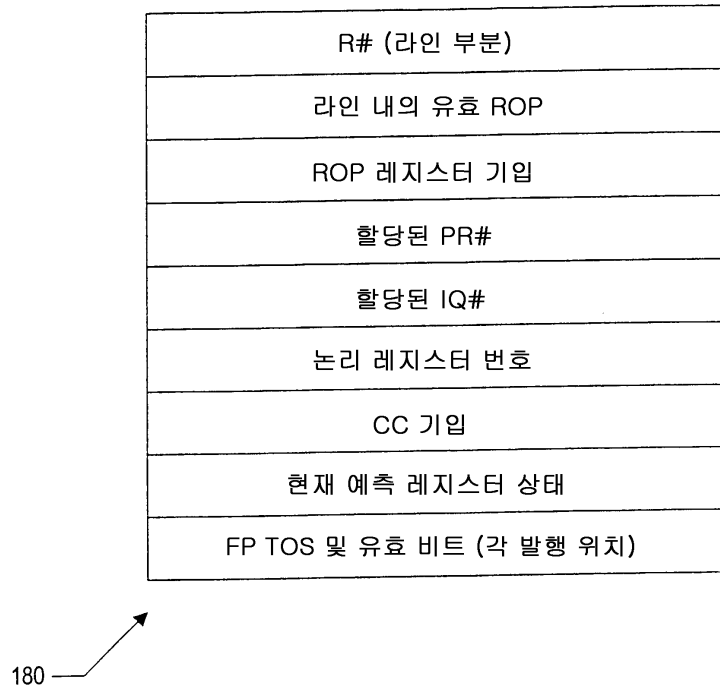
도면7



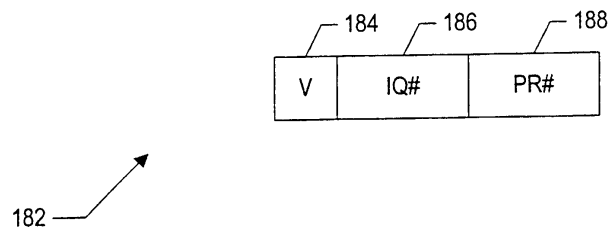
8판도



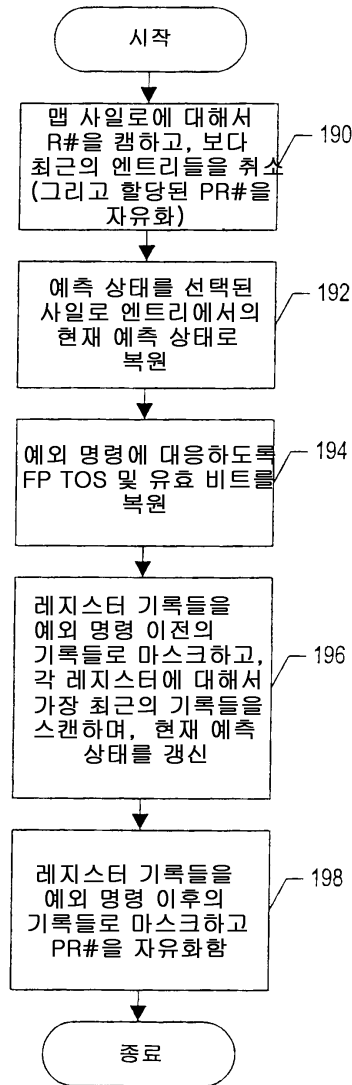
도면9



도면10



도면11



도면12

