(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.[7]
G06F 9/38

(45) 공고일자 2005년06월27일
(11) 등록번호 10-0496904
(24) 등록일자 2005년06월14일

(21) 출원번호 10-2001-0073012
(22) 출원일자 2001년11월22일

(65) 공개번호 10-2003-0042290
(43) 공개일자 2003년05월28일

(73) 특허권자 연세대학교 산학협력단
서울 서대문구 신촌동 134 연세대학교

(72) 발명자 문병인
경기도고양시덕양구행신2동793소만마을901-706

김문경
서울특별시동대문구답십리1동259-20

홍인표
경기도성남시분당구서현동330-7

김기창
인천광역시연수구동춘동924대림아파트207-404

이용석
서울 서초구 방배동 1-119 프레스턴 601호

(74) 대리인 이영필

심사관 : 성경아

(54) 레지스터 리네이밍 방법을 사용하는 조건부 실행 비순차적명령어 이슈 컴퓨터 시스템

요약

본 발명은 레지스터 리네이밍 방법을 사용하는 조건부 실행 비순차적 명령어 이슈 컴퓨터 시스템을 개시한다.

본 발명에 의하면, 순차적 상태 물리적 레지스터와 미리보기 상태 물리적 레지스터들 양자를 모든 논리적 레지스터들이 공유할 수 있도록 포함하고 있는 레지스터 파일, 레지스터 파일 내의 물리적 레지스터 중에서 어느 것들이 각각의 논리적 레지스터의 순차적 상태를 나타내는지를 지시하는 순차적 상태 지시기, 레지스터 파일 내의 물리적 레지스터 중에서 어느 것들이 각각의 논리적 레지스터의 구조적 상태를 나타내는지를 지시하는 리네임 상태 지시기, 물리적 레지스터들을 논리적 레지스터들에 할당하거나 혹은 프리한 것을 새로운 물리적 레지스터의 할당에 사용될 수 있는지를 알리는 물리적 레지스터 할당 지시기, 할당된 물리적 조건 레지스터의 수를 지시하는 물리적 조건 레지스터 할당 카운터, 조건부 실행 명령어의 실행 비실행 여부를 예측하기 위한 조건예측버퍼 및 미리보기 상태의 물리적 레지스터 주소 값들을 선입선출 형식으로 보유하고 있는 리오더 버퍼를 포함하여, 조건부 실행 비순차적 컴퓨터 시스템의 레지스터 리네이밍 및 순차적 상태의 추적을 가능하게 하며, 실행 예측에 대한 높은 정확성이 기반이 될 경우에 시스템의 성능을 크게 향상시킨다.

대표도

도 12

명세서

도면의 간단한 설명

도 1은 본 발명에 따른 조건부 실행 비순차적 시스템의 구조를 블록으로 도시한 것이다.

도 2는 조건부 실행 비순차적 시스템에서 수행되는 일련의 명령어들을 예시한 것이다.

도 3은 본 발명에 따른 조건부 실행 비순차적 시스템의 명령어 디코드 단계에서 명령어에 대한 실행 예측이 이루지는 과정을 도시한 것이다.

도 4는 본 발명에 따른 조건부 실행 비순차적 시스템의 물리적 레지스터 할당 지시기와 리네임 상태 지시기의 구성과, 레지스터 리네이밍 과정의 일예를 도시한 것이다.

도 5는 본 발명에 따른 조건부 실행 비순차적 시스템에서 물리적 조건 레지스터 할당 지시기, 물리적 조건 레지스터 할당 카운터 및 리네임 조건 상태 지시기를 사용하여 조건 레지스터의 갱신에 물리적 조건 레지스터를 할당하고, 조건 레지스터의 읽기를 물리적 조건 레지스터의 읽기로 리네이밍하는 과정의 일예를 도시한 것이다.

도 6은 본 발명에 따른 조건부 실행 비순차적 시스템의 명령어 이슈 큐와 리오더 버퍼의 구성 및 명령어 이슈 큐와 리오더 버퍼에 명령어를 저장하는 모습의 일 예를 도시한 것이다.

도 7은 본 발명에 따른 조건부 실행 비순차적 시스템에서 명령어 이슈 큐로부터 기능 유닛으로 명령어가 이슈되는 예를 도시한 것이다.

도 8은 본 발명에 따른 조건부 실행 비순차적 시스템에서 기능 유닛이 명령어 수행을 완료한 후에 그 결과를 레지스터 파일과 조건 레지스터 어레이에 저장하고, 리오더 버퍼에 명령어의 수행이 완료되었음을 알려주는 일 예를 도시한 것이다.

도 9는 본 발명에 따른 조건부 실행 비순차적 시스템에서 실행 예측된 조건부 실행 명령어를 리오더 버퍼로부터 커미트하는 과정의 일 예를 도시한 것이다.

도 10은 본 발명에 따른 조건부 실행 비순차적 시스템에서 조건 레지스터 갱신 명령어를 리오더 버퍼로부터 커미트하는 과정의 일 예를 도시한 것이다.

도 11은 본 발명에 따른 조건부 실행 비순차적 시스템에서 비실행 예측된 조건부 실행 명령어를 리오더 버퍼로부터 커미트하는 과정의 일 예를 도시한 것이다.

도 12는 본 발명에 따라 레지스터 리네이밍, 조건 레지스터의 리네이밍 및 순차적 상태 추적 과정을 관리하는 시스템을 블록으로 도시한 것이다.

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 비순차 방식의 명령어를 이슈하여 수행하는 컴퓨터 시스템에 관한 것으로서, 레지스터 리네이밍 방법을 사용하는 조건부 실행 비순차적 명령어 이슈 컴퓨터 시스템에 관한 것이다.

본 발명은 명령어(instruction)의 조건부 실행(conditional execution)을 특징으로 하는 명령어 세트 구조(instruction set architecture)를 채택한 비순차적(out-of-order) 명령어 이슈 컴퓨터 시스템(computer system)의 일반 레지스터(general register)와 조건 레지스터(condition register)에 대한 레지스터 리네이밍(register renaming), 물리적 레지스터(physical register)의 할당(allocation)과 퇴거(retire) 및 순차적 상태(in-order state)의 추적과 복구를 관리하는 방법에 관한 것이다.

또한 이러한 컴퓨터 시스템에서의 분기 예측 오류와 예외에 대비하여 순차적 상태를 추적하는 방법, 그리고 조건부 실행 명령어의 실행 예측 오류가 발생했을 때, 순차적 상태를 복구하고 프로그램(program)을 다시 시작하는 방법에 관한 것이다.

컴퓨터 시스템에서 프로그램의 명령어들을 순차적으로(in-order) 수행하는 것은 프로그램 수행의 기본이 되는 전제이다. 따라서 초기의 컴퓨터 시스템에서는 프로그램을 순차적으로 수행하였으며, 이러한 시스템에서 레지스터를 포함한 컴퓨터 시스템의 상태(state)는 프로그램에서 지정된 순서대로 쓰여지고 읽혀진다. 이러한 시스템에서는 명령어 상의 레지스터 주소(register address)와 실제 물리적 레지스터는 1대1 대응 관계를 가질 수 있으며, 레지스터 주소가 주어지면 정해진 물리적 레지스터를 쓰거나 읽으면 된다.

프로그램에서 각각의 명령어들은 데이터 의존성, 프로시져 의존성 (procedural dependency) 및 자원 충돌(resource conflict)이 존재하지 않는 경우에 한하여 기능 유닛으로 이슈되어 실행된다. 이러한 의존성, 특히 데이터 의존성은 실제 프로그램 수행 중에 자주 발생하며, 이로 인해 프로그램의 성능을 향상시키는 데 한계를 가지게 된다.

선행하는 명령어가 의존성으로 인해서 수행될 수 없을 때 후행 명령어 중에서 의존성이 존재하지 않는 것을 찾아서 이슈하고 수행할 수 있다면 컴퓨터 시스템의 성능을 향상시킬 수 있을 것이다. 여러 개의 기능 유닛을 가지고 있는 시스템에서 정수(integer) 유닛을 사용하는 명령어의 결과는 부동 소수점(floating-point) 유닛을 사용하는 명령어보다 더 빨리 그 결과가 나온다. 따라서 부동 소수점 연산의 결과를 소스로 사용하는 선행 명령어가 데이터 의존성 때문에 수행되지 못할 때,

정수 연산의 결과를 소스로 하는 후행 명령어는 어떠한 의존성도 존재하지 않는 경우가 자주 발생하며, 이러할 때에 후행 명령어를 선행 명령어보다 먼저 수행하도록 할 수 있다면 컴퓨터 시스템의 성능은 크게 향상된다. 이러한 비순차적 명령어 이슈(out-of-order instruction issue)의 필요성은 단일 사이클(cycle)에 다수의 명령어를 이슈하고 수행하는 수퍼스칼라 프로세서(superscalar processor)의 등장으로 더욱 커지게 되었다. 수퍼스칼라 구조에서는 다수의 명령어를 동시에 이슈할 수 있는 이슈 자원과 이 명령어들을 동시에 수행할 수 있는 다수의 기능 유닛들을 가지고 있다. 다수의 명령어를 동시에 수행하기 때문에 데이터 의존성으로 인해서 명령어를 이슈할 수 없을 경우 낭비되는 자원이 매우 크게 되며 비효율적으로 된다. 이러한 이유로 해서 많은 수퍼스칼라 구조에서는 비순차적 명령어 이슈 방법을 사용하여서 자원의 낭비를 줄이고 성능을 높인다.

비순차적 명령어 이슈를 할 때에 가장 주의할 점은 프로그램의 동작은 순차적으로 명령어를 이슈할 때와 같아야 한다는 것이다. 내부적으로는 비순차적 이슈를 사용하지만 각 명령어의 수행 또는 수행 결과를 포함해서 겉으로 나타나는 결과는 순차적 명령어 이슈를 사용할 때와 같아야 한다.

명령어를 비순차적으로 이슈할 때에는 레지스터를 포함해서 컴퓨터 시스템의 상태들이 비순차적으로 변경될 뿐만 아니라, 데이터 의존성에 의한 관계가 깨진다. 예를 들어 명령어 B가 명령어 A의 목적 레지스터를 소스 레지스터로 사용하고, 명령어 C가 동일한 레지스터를 목적 레지스터로 사용하는 경우에, C가 B보다 먼저 이슈된다면 명령어 B가 이슈될 때에는 소스 레지스터에 대한 최근의 쓰기는 C에 의한 쓰기가 된다. 따라서 명령어 B는 명령어 C의 수행 결과에 의한 값을 소스로 하게 되는 위험 요소가 생긴다. 그러나 프로그램의 정상적인 수행을 위해서는 명령어 B는 명령어 A에 의한 결과를 소스로 해야 한다. 이렇게 하기 위해서는 명령어 A와 명령어 C의 목적 레지스터 주소가 같더라도 실제로는 서로 다른 물리적 레지스터들을 할당받아서 각 명령어는 자신의 물리적 레지스터에 결과 값을 써야 하며, 명령어 B는 명령어 A에 할당된 물리적 레지스터에서 소스 값을 읽어야 한다. 이와 같이 단일 논리적 레지스터에 대해서 다수의 물리적 레지스터를 할당하고, 주어진 논리적 레지스터에 대해서 실제의 값을 저장하는 물리적 레지스터를 식별하는 것을 레지스터 리네이밍이라고 한다.

레지스터 리네이밍을 수행할 때에는 논리적 레지스터의 갱신에 대해서 물리적 레지스터를 할당하는 과정이 수반된다. 그리고, 논리적 레지스터의 읽기에 대해서 레지스터 값을 읽어야 할 해당 물리적 레지스터를 식별해 주어야 한다. 물리적 레지스터의 어레이는 유한하기 때문에, 물리적 레지스터를 할당하는 과정만 존재한다면 더 이상 할당할 물리적 레지스터가 없게 된다. 따라서 더 이상 참조되지 않는 물리적 레지스터를 퇴거시켜서, 그 레지스터를 새로운 물리적 레지스터의 할당에 사용할 수 있도록 해야 한다.

순차적으로 명령어를 수행할 때에는 프로세서는 프로그램에 의해서 정의되는 순차적 상태를 유지하고 있다. 따라서 분기 예측 오류나 인터럽트(interrupt)와 같은 예외가 발생하였을 때, 프로세서는 발생 시점의 순차적인 상태를 가지고 있으며, 이 상태에서 분기 예측 오류나 예외를 처리한다. 그러나 레지스터 리네이밍을 통해 비순차적 명령어 이슈 방법을 사용하는 프로세서는 순차적 상태를 유지하고 있지 않다. 분기 예측 오류나 예외의 경우에 순차적 명령어 수행의 경우와 같은 동작 모습을 보이기 위해서 분기 예측 오류나 예외 발생 시에 순차적 상태를 복구할 수 있어야 한다.

하나의 명령어가 완료되고 선행하는 모든 명령어가 완료되었을 때에는 그 명령어의 결과가 순차적 상태로 저장될 수 있다. 비순차적 컴퓨터 시스템에서 순차적 상태는 완료된 명령어의 연속적인 열(continuous string) 중에서 가장 최근까지의 완료된 명령어에 의해 변경이 완료된 상태를 의미한다. 분기 예측 오류 또는 예외의 경우에 발생 시점의 순차적 상태에서 명령어의 수행을 다시 시작하게 된다. 미리보기 상태(lookahead state)는 순차적 상태를 정의하는 마지막 명령어의 다음 명령어부터 레지스터 리네이밍이 이루어진 가장 최근의 명령어에 이르는 연속적인 명령어 열에 의해서 정의되며, 이 상태에 속하는 명령어가 수행을 완료하고, 선행 명령어가 모두 완료되었을 때에 그 명령어의 결과는 순차적 상태로 저장된다. 구조적 상태(architectural state)는 레지스터 리네이밍이 이루어진 최근까지의 명령어들에 의해서 정의된다. 따라서 구조적 상태는 순차적 상태와 미리보기 상태를 결합한 상태이면서 최신의 리네임 상태이다. 레지스터 리네이밍의 수행, 명령어들 간의 데이터 의존성 검사와 같은 정상적인 명령어 수행에 관한 것은 구조적 상태를 참조해서 이루어진다.

컴퓨터 시스템이 채택한 명령어 세트 구조에 따라서 시스템의 구조와 동작 및 성능이 크게 달라진다. 레지스터 리네이밍도 시스템의 명령어 세트 구조에 따라서 구현의 복잡도와 효율성이 많이 달라지며, 시스템의 명령어 세트 구조가 거의 모든 종류의 명령어들에 대해서 조건부 실행 형식(form)을 제공한다면 그 시스템의 레지스터 리네이밍은 매우 어렵고, 기존의 단순한 레지스터 리네이밍 방법은 이러한 시스템에 적용할 수 없다.

명령어 세트 구조가 어떤 명령어에 대해서 조건부 실행 형식을 제공한다는 것의 의미는 다음과 같다. 예를 들어서 두개의 소스 레지스터의 값들을 더해서 그 결과를 목적 레지스터에 저장하는 명령어를 ADD라고 하자. 명령어 ADD에 대해서 조건부 실행 형식을 제공하지 않는 경우에는 명령어 ADD는 항상 실행되어 덧셈의 결과를 목적 레지스터에 저장한다. 그러나, 명령어 ADD에 조건부 실행 형식을 제공하는 명령어 세트 구조에서는, 명령어 ADD는 명령어 수행 시의 조건 플래그(condition flag)들의 값들에 따라서 실제로 실행될 수도 있으며, 그렇지 않을 수도 있다. 이 경우 명령어 ADD가 조건 플래그들이 어떤 값들을 가질 때 실제로 실행되는지는 명령어의 조건 코드 영역(condition code field)에 의해서 결정된다. 즉 전체 명령어 영역 중에서 조건 코드 영역의 값에 따라서 명령어 ADD를 실제로 실행하기 위한 조건 플래그들 값들에 대한 조건이 달라지는 것이다. 이와 같이 어떤 종류의 명령어 영역 중에서 조건 코드 영역이 존재하고, 조건 코드 영역의 값에 따라서 그 명령어의 실제 실행을 위한 조건이 달라지며, 명령어 수행 시에 조건 코드에 의한 조건이 만족하냐에 따라서 실제 실행 여부가 결정될 때, 그 명령어에 대해서 조건부 실행 형식을 제공한다고 할 수 있다. 그리고, 명령어 세트 구조가 거의 모든 종류에 대해서 조건부 실행 형식을 제공할 때, 그 명령어 세트 구조는 명령어의 조건부 실행을 특징으로 하는 명령어 세트 구조라고 할 수 있는 것이다.

명령어 세트 구조가 어떤 종류의 명령어에 대해서 조건부 실행 형식을 제공한다고 해서 그 종류의 명령어가 항상 조건부 실행을 하는 것은 아니다. 조건 플래그 값들에 상관 없이 그 명령어를 실행하도록 하는 조건 코드 값이 존재한다. 예를 들어, 이와 같은 조건 코드 값이 명령어 ADD의 조건 코드 영역에 설정되어 있으면, 명령어 ADD는 조건 플래그들의 값에 상관 없이 항상 실행된다.

조건 플래그들의 종류는 명령어 세트 구조에 따라서 다르지만, 보통은 N, Z, V, C의 네 개가 존재하며, 각각은 한 비트이다. 이러한 조건 플래그들은 모든 명령어들에 의해서 변경될 수 있는 것은 아니며, 조건 플래그를 변경하는 종류의 명령어들에 의해서만 변경된다. 조건 플래그 N은 최근의 조건 플래그 변경 명령어의 결과가 음(minus)의 값을 갖는다는 의미이

고, 조건 플래그 Z는 영(zero)의 값을 갖는다는 의미이다. 그리고, 조건 플래그 V는 최근의 조건 플래그 변경 명령어의 실행에서 오버플로우(overflow)가 발생했다는 의미이고, 조건 플래그 C는 캐리(carry)가 발생했다는 의미이다. 이러한 조건 플래그들은 시스템의 한 레지스터에 저장되어 있는데 이 레지스터를 조건 레지스터라고 한다.

조건 레지스터가 존재하는 비순차적 시스템에서는, 논리적 레지스터를 물리적 레지스터로 리네이밍하듯이, 프로그램 모델(program model) 상의 조건 레지스터인 논리적 조건 레지스터(logical condition register)도 물리적 조건 레지스터로 리네이밍하는 것이 필요하다. 즉 비순차적으로 명령어를 이슈하고 실행하면서 조건 레지스터에 대한 명령어 간의 데이터 의존성을 올바르게 해결하기 위해서는, 각각의 논리적 조건 레지스터 갱신 인스턴스에 대해서 상이한 물리적 조건 레지스터를 할당하고 각각의 조건 레지스터 읽기는 적절한 물리적 조건 레지스터 읽기로 리네이밍해 주어야 한다.

컴퓨터 시스템의 명령어 세트 구조가 명령어의 조건부 실행을 특징으로 하면 기존의 레지스터 리네이밍 방법을 적용할 수 없다. 예를 들어서 명령어 A가 논리적 레지스터 Ra을 목적 레지스터로 하는 조건부 실행 명령어이고, 명령어 A 다음의 명령어 B가 논리적 레지스터 Ra을 소스 레지스터로 사용한다고 하자. 그리고, 명령어 A 이전에 최신의 물리적 레지스터 할당에 의해서, 논리적 레지스터 Ra에 대해서 물리적 레지스터 PRa가 할당되었다고 하자. 그리고, 명령어 A에 대한 레지스터 리네이밍 동작에 의해서 명령어 A의 논리적 목적 레지스터 Ra에 대해서 물리적 목적 레지스터(physical destination register)가 할당되며, 이 물리적 레지스터를 PRb라고 하자. 상기 가정 하에, 명령어 B의 논리적 소스 레지스터 Ra는 물리적 레지스터 PRb로 리네이밍된다. 조건부 실행 명령어 A의 조건이 맞아서 명령어 A가 실제로 실행이 된다면 명령어 B의 논리적 소스 레지스터 Ra에 대한 리네이밍은 올바르게 된 것이다. 그러나, 명령어 A의 조건이 맞지 않아서, 명령어 A가 실제로 실행되지 않았다면 명령어 B의 소스 레지스터 Ra을 물리적 레지스터 PRb로 리네이밍한 것은 잘못된 것이며, 명령어 B의 소스 레지스터 Ra은 물리적 레지스터 PRa로 리네이밍되어야 한다. 반대로, 명령어 A의 논리적 목적 레지스터 Ra에 대해서 물리적 레지스터를 할당하지 않고, 명령어 B의 논리적 소스 레지스터 Ra를 물리적 레지스터 PRa로 리네이밍할 수도 있다. 이 경우, 조건부 실행 명령어 A의 조건이 맞지 않아서 명령어 A의 실행이 이루어지지 않았다면, 명령어 B의 논리적 소스 레지스터 Ra에 대한 리네이밍은 올바르게 된 것이다. 그러나, 명령어 A의 조건이 맞아서 명령어 A가 실행이 되었다면, 명령어 A의 논리적 목적 레지스터 Ra에 대해서 물리적 목적 레지스터가 할당되고, 명령어 B의 논리적 소스 레지스터 Ra는 이 물리적 레지스터로 리넹이밍되었어야 한다.

이와 같이, 조건부 실행 명령어가 존재하는 시스템의 레지스터 리네이밍이 어려운 것은 조건부 실행 명령어와 그 후속 명령어들의 레지스터 리네이밍 시점에서 조건부 실행 명령어의 실행 여부가 결정되지 않는다는 것이다. 이것은 조건부 분기 명령어(conditional branch instruction)에 의한 프로시져 의존성과 비슷한 성격을 가지고 있다. 이러한 문제점을 제거하기 위해서 조건부 실행 명령어의 후속 명령어들에 대한 레지스터 리네이밍을 조건부 실행 명령어의 실행 여부가 결정될 때까지 정지 시킬 수도 있으나, 이렇게 할 때에는 컴퓨터 시스템의 성능이 매우 낮아지고 비순차적 명령어 이슈의 이점을 살리지 못한다.

비순차적 컴퓨터 시스템에서 레지스터 리네이밍 및 순차적 상태의 추적을 관리하는 중에 리오더 버퍼를 이용하여 상이한 논리적 목적 레지스터 인스턴스의 레지스터 값을 저장하고 순차적 상태를 추적하는 방법이 있다. 그 외에, 리오더 버퍼의 연합적 찾기(associative lookup)라는 단점을 없애기 위해서 미래 파일(future file)을 리오더 버퍼에 추가해서 사용하는 방법, 리오더 버퍼의 기능을 명령어 창(instruction window)에 포함시키는 방법, 그리고 각각의 논리적 목적 레지스터 인스턴스의 값을 개별적인 임시 레지스터 어레이에 보유하는 방법 등이 있다. 이러한 방법들의 몇 가지 단점들 중의 하나로서, 순차적 상태를 추적하고 복구하기 위해서 임시 저장 장치의 레지스터 값들이 순차적 상태를 저장하고 있는 레지스터 파일 또는 레지스터 어레이로 전송되어야 한다는 것이 있다. 이러한 단점을 제거하기 위해서 단일 레지스터 파일에 미리보기 상태의 레지스터 값과 순차적 상태의 레지스터 값을 모두 저장하는 방법도 존재한다.

그러나, 상기 방법들은 명령어의 조건부 실행이라는 특징이 없는 명령어 세트 구조를 채택한 컴퓨터 시스템에만 적용될 수 있다. 즉 그 자체로서의 상기 모든 방법들은, 명령어의 조건부 실행을 특징으로 하는 명령어 세트 구조를 채택한 컴퓨터 시스템에는 적용될 수 없다는 단점을 가지고 있다.

발명이 이루고자 하는 기술적 과제

본 발명이 이루고자 하는 기술적인 과제는, 상기의 문제점들을 해결하기 위해, 레지스터 리네이밍 방법을 사용하는 조건부 실행 비순차적 명령어 이슈 컴퓨터 시스템을 제공하는데 있다.

본 발명의 목적은 명령어의 조건부 실행을 특징으로 하는 비순차적 명령어 이슈 컴퓨터 시스템에서 레지스터 리네이밍과 물리적 레지스터의 할당과 퇴거를 관리하는 방법을 제공하는 것이다. 그리고, 본 발명의 또 하나의 목적은 분기 예측 오류와 예외에 대비하여 순차적 상태를 추적하는 방법을 제공하고, 조건부 실행 명령어의 실행 예측 오류가 발생했을 때, 순차적 상태를 복구하고 프로그램을 다시 시작하는 방법을 제공하는 것이다.

발명의 구성 및 작용

상기 기술적 과제를 해결하기 위한 본 발명에 의한, 레지스터 리네이밍 방법을 사용하는 조건부 실행 비순차적 명령어 이슈 컴퓨터 시스템은, 순차적 상태를 나타내는 물리적 레지스터와 미리보기 상태를 나타내는 물리적 레지스터들 양자를 모든 논리적 레지스터들이 공유할 수 있도록 포함하고 있는 레지스터 파일; 논리적 레지스터 수만큼의 엔트리들을 가지고 있으며, 각 엔트리들은 물리적 레지스터의 주소를 값으로 가지며, 상기 레지스터 파일 내의 물리적 레지스터 중에서 어느 것들이 각각의 논리적 레지스터의 순차적 상태를 나타내는지를 지시하는 순차적 상태 지시기; 논리적 레지스터 수만큼의 엔트리들을 가지고 있으며, 각 엔트리들은 물리적 레지스터의 주소를 값으로 가지며, 상기 레지스터 파일 내의 물리적 레지스터 중에서 어느 것들이 각각의 논리적 레지스터의 구조적 상태인 최신의 리네임 인스턴스를 나타내는지를 지시하는 리네임 상태 지시기; 상기 물리적 레지스터 중에서 어느 것들이 논리적 레지스터들에 할당되고, 어느 것들이 프리해서 새로운 물리적 레지스터의 할당에 사용될 수 있는지를 알려주며, 물리적 레지스터 수만큼의 엔트리들을 가지고 있으며, 각 엔트리들은 한 비트로 이루어지며, 각 엔트리는 대응하는 물리적 레지스터가 논리적 레지스터에 할당되어 있는 상태인지 아닌지를 지시하는 물리적 레지스터 할당 지시기; 조건 레지스터의 상이한 리네임 인스턴스들을 나타내는 물리적 조건 레지스터들을 포함하는 조건 레지스터 어레이; 물리적 조건 레지스터의 주소를 값으로 가지는 단일 엔트리로 이루어지며, 상기

조건 레지스터 어레이 중에서 어떤 물리적 조건 레지스터가 논리적 조건 레지스터의 순차적 상태를 저장하고 있는지를 지시하며, 물리적 조건 레지스터의 값이 미리보기 상태에서 순차적 상태로 저장될 때에는 그 값을 나타내며, 조건 레지스터의 순차적 상태를 저장하는 물리적 조건 레지스터의 주소 값을 갱신하면, 기존에 조건 레지스터의 순차적 상태를 저장하고 있던 물리적 조건 레지스터는 더 이상 참조되지 않고 퇴거되도록 하는 순차적 조건 상태 지시기; 물리적 조건 레지스터의 주소를 값으로 가지는 단일 엔트리로 이루어지며, 상기 논리적 조건 레지스터의 구조적 상태인 최신의 리네임 인스턴스를 저장하고 있는 물리적 조건 레지스터를 알려주는 리네임 조건 상태 지시기; 논리적 조건 레지스터 갱신에 할당할 물리적 조건 레지스터의 주소 값을 저장하는 영역 및 현재 할당되어 있어서 프리하지 않은 물리적 조건 레지스터의 수를 저장하는 영역을 포함하는 단일 엔트리를 포함하며, 논리적 조건 레지스터의 갱신에 물리적 조건 레지스터를 할당할 때에는 프리한 물리적 조건 레지스터의 주소 값을 받아서 논리적 조건 레지스터에 할당하며, 이때에 상기 리네임 조건 상태 지시기의 값은 할당된 물리적 조건 레지스터의 주소로 그 값을 갱신함으로써, 조건 레지스터에 대한 최신의 리네임 인스턴스를 가리키도록 하는 물리적 조건 레지스터 할당 지시기; 상기 할당된 물리적 조건 레지스터의 수를 지시하는 물리적 조건 레지스터 할당 카운터; 조건부 실행 명령어의 실행 비실행 여부를 예측하기 위한 조건 예측 버퍼; 및 미리보기 상태의 물리적 레지스터 주소 값들을 선입선출 형식으로 보유하고 있으며, 명령어의 레지스터 리네이밍을 수행한 후에 논리적 레지스터 주소 값, 할당받은 물리적 레지스터의 주소 값 및 할당받은 물리적 조건 레지스터의 주소 값을 포함하는 정보가 최상위 엔트리에 기록되며, 해당 엔트리가 최하위 엔트리에 도달했고, 명령어 수행을 완료했을 때에는 그 엔트리는 커미트되며, 커미트는 대응되는 순차적 상태 지시기 엔트리의 물리적 레지스터 주소 값과 순차적 조건 상태 지시기의 물리적 조건 레지스터 주소 값을 갱신함으로써 이루어지는 리오더 버퍼를 포함하는 것을 특징으로 한다.

명령어를 비순차적으로 수행하면서, 데이터 의존성(data dependency)을 올바르게 해결하기 위해서는 논리적 목적 레지스터(logical destination register)에 대한 쓰기(write) 인스턴스(instance)마다 새로운 물리적 레지스터를 할당하고, 논리적 소스 레지스터(logical source register)를 최신의 리네임 상태(rename state)를 저장하고 있는 물리적 레지스터로 리네이밍하는 것이 필요하다. 또한 분기 예측 오류(branch misprediction) 및 예외(exception) 발생 시 순차적 상태에서 다시 시작할 수 있도록 순차적 상태를 추적 및 복구할 수 있어야 한다.

조건부 실행을 특징으로 하는 명령어 세트 구조를 채택한 비순차적 시스템에서는, 조건부 실행 명령어의 실행 여부가 후속하는 명령어의 레지스터 리네이밍에 미치는 영향 때문에 기존의 방법으로는 레지스터 리네이밍 및 순차적 상태의 추적을 수행할 수 없다.

본 발명에서는 일반 레지스터의 순차적 상태 및 최신의 리네임 상태를 저장하는 물리적 레지스터들을 모두 포함하는 물리적 레지스터 어레이(array)로 구성된 레지스터 파일(register file), 순차적 상태 지시기(in-order state indicator), 리네임 상태 지시기(rename state indicator), 물리적 레지스터 할당 지시기(physical register allocation indicator), 조건 레지스터의 순차적 상태 및 최신의 리네임 상태를 저장하는 물리적 조건 레지스터(physical condition register)들을 모두 포함하는 조건 레지스터 어레이(condition register array), 순차적 조건 상태 지시기(in-order condition state indicator), 리네임 조건 상태 지시기(rename condition state indicator), 물리적 조건 레지스터 할당 지시기(physical condition register allocation indicator) 및 물리적 조건 레지스터 할당 카운터(count of allocated physical condition registers) 등을 사용하여, 명령어의 조건부 실행을 특징으로 하는 명령어 세트 구조를 채택한 비순차적 컴퓨터 시스템에서 효율적이면서 저 비용으로 일반 레지스터와 조건 레지스터에 대한 물리적 레지스터의 할당과 리네이밍 및 순차적 상태의 추적과 복구를 관리한다.

본 발명의 컴퓨터 시스템은 순차적 상태를 나타내는 물리적 레지스터와 미리보기 상태를 나타내는 물리적 레지스터들 양자를 모든 논리적 레지스터들이 공유할 수 있도록 포함하고 있는 레지스터 파일, 레지스터 파일 내의 물리적 레지스터 중에서 어느 것들이 각각의 논리적 레지스터의 순차적 상태를 나타내는지를 지시하는 순차적 상태 지시기, 레지스터 파일 내의 물리적 레지스터 중에서 어느 것들이 각각의 논리적 레지스터의 구조적 상태(최신의 리네임 인스턴스)를 나타내는지를 지시하는 리네임 상태 지시기, 물리적 레지스터 중에서 어느 것들이 논리적 레지스터들에 할당되고, 어느 것들이 프리(free)해서 새로운 물리적 레지스터의 할당에 사용될 수 있는지를 알려주는 물리적 레지스터 할당 지시기, 조건 레지스터의 상이한 리네임 인스턴스들을 나타내는 물리적 조건 레지스터들을 포함하는 조건 레지스터 어레이, 조건 레지스터 어레이 중에서 어떤 물리적 조건 레지스터가 논리적 조건 레지스터의 순차적 상태를 저장하고 있는지를 지시하는 순차적 조건 상태 지시기, 논리적 조건 레지스터의 구조적 상태(최신의 리네임 인스턴스)를 저장하고 있는 물리적 조건 레지스터를 알려주는 리네임 조건 상태 지시기, 논리적 조건 레지스터 갱신에 할당할 물리적 조건 레지스터를 지시하는 물리적 조건 레지스터 할당 지시기, 할당된 물리적 조건 레지스터의 수를 지시하는 물리적 조건 레지스터 할당 카운터, 조건부 실행 명령어의 실행 비실행 여부를 예측하기 위한 조건 예측 버퍼(condition prediction buffer) 및 미리보기 상태의 물리적 레지스터 주소 값들을 선입선출 형식으로 보유하고 있는 리오더 버퍼를 가지는 것을 특징으로 한다. 이러한 구성 요소들에 의해서 본 발명은 명령어의 조건부 실행을 특징으로 하는 명령어 세트 구조를 채택한 비순차적 명령어 이슈 컴퓨터 시스템(이하 조건부 실행 비순차적 시스템이라는 명칭 사용)에서의 레지스터 리네이밍, 물리적 레지스터의 할당과 퇴거 및 순차적 상태 추적을 관리한다.

보통의 단일 레지스터 파일이 순차적 상태를 저장하는 물리적 레지스터들과 미리보기 상태를 저장하는 물리적 레지스터들을 모두 포함하고 있으므로, 미리보기 상태의 물리적 레지스터 값을 위한 임시 저장 장치가 필요하지 않으며 순차적 상태를 보전하기 위해서 임시 저장 장치에서 레지스터 파일로 데이터를 전송할 필요도 없다. 또한 모든 물리적 레지스터들은 어떤 논리적 레지스터에 대해서도 할당될 수 있기 때문에 물리적 레지스터의 활용도를 높일 수 있으며, 레지스터 파일의 크기를 감소시킬 수 있다.

순차적 상태를 저장하는 물리적 레지스터는 상기 순차적 상태 지시기에 의해서 결정된다. 물리적 레지스터의 값이 미리보기 상태에서 순차적 상태로 저장될 때에는 물리적 레지스터의 값을 전송하는 것이 아니라 순차적 상태 지시기가 적절히 변경된다.

논리적 목적 레지스터에 물리적 레지스터를 할당할 때에는 물리적 레지스터 할당 지시기에 의해서 프리한 물리적 레지스터의 주소 값을 받아서 논리적 목적 레지스터에 할당한다. 이 과정에서 리네임 상태 지시기의 엔트리 중에서 논리적 목적 레지스터에 대응되는 엔트리는 할당된 물리적 레지스터의 주소로 그 값을 갱신함으로써, 최신의 리네임 인스턴스를 가리키도록 한다.

순차적 상태 지시기와 리네임 상태 지시기는 각각 논리적 레지스터 수만큼의 엔트리들을 가지고 있으며, 각 엔트리들은 물리적 레지스터의 주소를 값으로 가지기 때문에 각기 논리적 레지스터당 몇 비트만을 필요로 하므로, 상기 상태들을 지시하는 매우 저렴한 방법이다.

물리적 레지스터 할당 지시기는 물리적 레지스터 수만큼의 엔트리들을 가지고 있으며, 각 엔트리들은 한 비트로 이루어져 있다. 각 엔트리는 대응하는 물리적 레지스터가 논리적 레지스터에 할당되어 있는 상태인지 아닌지를 지시한다.

논리적 소스 레지스터를 리네임할 때에는 대응하는 리네임 상태 지시기 엔트리 값을 읽어서 최신의 리네임 인스턴스인 물리적 레지스터를 확인한다.

조건 레지스터의 순차적 상태를 저장하는 물리적 조건 레지스터는 순차적 조건 상태 지시기에 의해서 결정된다. 물리적 조건 레지스터의 값이 미리보기 상태에서 순차적 상태로 저장될 때에는 물리적 조건 레지스터의 값을 전송하는 것이 아니라 순차적 조건 상태 지시기가 적절히 변경된다.

논리적 조건 레지스터의 갱신에 물리적 조건 레지스터를 할당할 때에는 물리적 조건 레지스터 할당 지시기에 의해서 프리한 물리적 조건 레지스터의 주소 값을 받아서 논리적 조건 레지스터에 할당한다. 이 과정에서 리네임 조건 상태 지시기의 값은 할당된 물리적 조건 레지스터의 주소로 그 값을 갱신함으로써, 조건 레지스터에 대한 최신의 리네임 인스턴스를 가리키도록 한다.

순차적 조건 상태 지시기와 리네임 조건 상태 지시기는 각각 물리적 조건 레지스터의 주소를 값으로 가지는 단일 엔트리로 이루어졌다.

물리적 조건 레지스터 할당 지시기는 다음의 논리적 조건 레지스터 갱신에 할당할 물리적 조건 레지스터의 주소 값을 저장하는 영역 및 현재 할당되어 있어서 프리하지 않은 물리적 조건 레지스터의 수를 저장하는 영역을 포함하는 단일 엔트리로 이루어져 있다.

논리적 조건 레지스터를 물리적 조건 레지스터로 리네임할 때에는 리네임 조건 상태 지시기의 값을 읽어서 최신의 리네임 인스턴스인 물리적 조건 레지스터를 확인한다.

명령어의 레지스터 리네이밍을 수행한 후에 논리적 레지스터 주소 값, 할당받은 물리적 레지스터의 주소 값 및 할당받은 물리적 조건 레지스터의 주소 값 등이 리오더 버퍼의 최상위 엔트리(top entry)에 기록된다. 해당 엔트리가 리오더 버퍼의 최하위 엔트리(bottom entry)에 도달했고, 명령어 수행을 완료했을 때에는 그 엔트리는 커미트된다. 커미트는 물리적 레지스터의 값을 전송하는 것에 의해서 이루어지는 것이 아니라, 대응되는 순차적 상태 지시기 엔트리의 물리적 레지스터 주소 값과 순차적 조건 상태 지시기의 물리적 조건 레지스터 주소 값을 갱신함으로써 이루어진다.

순차적 상태 지시기에 의해서 순차적 상태를 저장하는 물리적 레지스터의 주소 값을 갱신할 때에는, 기존의 순차적 상태를 나타내는 물리적 레지스터는 더 이상 참조되지 않기 때문에 퇴거되며, 차후의 물리적 레지스터 할당에 사용된다.

그리고, 순차적 조건 상태 지시기에 의해서 조건 레지스터의 순차적 상태를 저장하는 물리적 조건 레지스터의 주소 값을 갱신하면, 기존에 조건 레지스터의 순차적 상태를 저장하고 있던 물리적 조건 레지스터는 더 이상 참조되지 않기 때문에 퇴거되며, 차후의 물리적 조건 레지스터 할당에 사용된다.

본 발명의 시스템에서는, 조건부 실행 명령어가 실제로 실행될 지에 대한 예측이 이루어지며, 이 예측에 근거해서 레지스터 리네이밍이 이루어진다. 명령어 커미트 시에 예측이 틀린 경우는 이를 복구하기 위한 작업이 이루어진다.

도 1은 본 발명의 레지스터 리네이밍 방법을 사용하여 비순차적으로 명령어를 이슈하고 수행하는 조건부 실행 컴퓨터 시스템의 전체적인 구성을 보여준다.

명령어들은 명령어 캐쉬(instruction cache)로부터 페치(fetch)되어서 명령어 주소(instruction address)와 같이 선입선출 방식의 명령어 페치 큐(instruction fetch queue)에 순서대로 저장된다. 수퍼스칼라 컴퓨터 시스템에서는 시스템의 사양에 따라서 단일 사이클에 페치되는 명령어의 수가 달라진다. 명령어 페치 과정에서는 분기 예측(branch prediction)이 이루어진다. 명령어 페치 큐에 저장된 명령어 중에서 가장 오래된 명령어들은 명령어 디코드 유닛(instruction decode unit)에 의해서 디코드되고, 디코드된 명령어들은 명령어 페치 큐에서 제거된다. 명령어를 디코드하는 과정에서는 명령어를 내부적으로 사용하는 오피 코드(op code)와 소스 및 목적 레지스터 주소들로 변환된다. 여기에서 단일 명령어가 가질 수 있는 목적 레지스터 수는 한 개이며, 소스 레지스터 수는 두 개이다. 명령어 디코드 과정에서는 명령어의 실제 실행 여부를 예측하는 작업도 이루어진다. 이 과정에서 조건부 실행 명령어의 경우는 명령어 주소와 조건 예측 버퍼를 이용해서 실제 수행 여부를 예측한다. 명령어 디코드에 의해서 변환된 명령어들은 레지스터 리네이밍 유닛(register renaming unit)으로 전달되며, 레지스터 리네이밍 유닛은 데이터 의존성을 해결해 주는 역할을 한다. 레지스터 리네이밍 유닛에 의해서 논리적 목적 레지스터에 대해서 물리적 목적 레지스터를 할당하며, 논리적 소스 레지스터에 대해서는 실제로 대응되는 물리적 소스 레지스터(physical source register)로 변환하여 준다. 또한, 조건 레지스터를 갱신하는 명령어나 조건 레지스터의 값을 읽어서 명령어 수행에 사용하는 명령어에 대해서도, 조건 레지스터에 대한 물리적 조건 레지스터 할당이나 조건 레지스터의 읽기에 대한 물리적 조건 레지스터로의 리네이밍이 이루어진다. 다만, 실제로 실행되지 않을 것으로 예측된 명령어에 대해서는 이와 같은 리네이밍 작업의 대부분이 이루어지지 않는다.

상기 명령어 페치, 디코드 및 레지스터 리네이밍은 명령어들의 순차적인 순서대로 이루어진다.

레지스터 리네이밍이 이루어진 명령어는 명령어 이슈 큐로 전달되어 명령어 이슈 큐의 엔트리 하나를 차지한다. 또한 리오더 버퍼에도 목적 레지스터 주소와 물리적 조건 레지스터 주소에 대한 정보가 저장된다. 명령어 이슈 큐는 최하위 포인터(bottom pointer)와 최상위 포인터(top pointer)에 의해 순환적으로 관리된다. 즉 최하위 포인터는 명령어 이슈 큐에 존재하는 명령어 중에서 가장 오래된 명령어 엔트리를 가리키며, 최상위 포인터는 다음에 명령어를 받아들일 엔트리를 가리킨다. 명령어가 명령어 이슈 큐로 전달될 때에 해당 엔트리는 명령어의 실제 실행에 대한 예측 비트(이하 실행 예측 비트라고 함), 오피 코드, 물리적 목적 레지스터 주소, 물리적 소스 레지스터 주소들, 목적 레지스터로서의 물리적 조건 레지스터(이하 물리적 목적 조건 레지스터(physical destination condition register)라고 함)의 주소, 소스 레지스터로서의 물리적 조건 레지스터(이하 물리적 소스 조건 레지스터(physical source condition register)라고 함)의 주소 및 해당 명령어에 대응되는 리오더 버퍼의 엔트리를 가리키는 포인터를 저장한다. 리오더 버퍼는 각 명령어의 논리적 목적 레지스터 주소 및 물리적 목적 레지스터 주소 양자, 물리적 목적 조건 레지스터 주소, 명령어 주소, 조건 레지스터를 갱신하는 명령어임을 알려주는 비트(이하 조건 갱신 비트라고 함), 조건부 실행 명령어인지를 알려주는 비트(이하 조건부 비트라고 함), 실행 예측 비트 및 명령어 수행 결과로서 실제로 실행되는 명령어인지를 알려주는 비트(이하 실행 결과 비트라고 함)를 명령어들의 순차적인 순서대로 저장하고 있어서, 연속적으로 완료된 명령어 열에 대한 순차적 상태를 추적하기 위해서 사용된다. 리오더 버퍼는 명령어 이슈 큐와 같이 최하위 포인터와 최상위 포인터에 의해서 순환적으로 관리되며, 또한 선입선출 구조를 가진다.

명령어 이슈 큐에 존재하는 명령어에 대해서 소스 레지스터들 및 소스 조건 레지스터의 데이터가 모두 준비되고 사용하고자 하는 기능 유닛 중에서 사용할 수 있는 것이 존재할 경우에, 각 명령어는 기능 유닛으로 이슈된다. 사용하는 기능 유닛의 종류는 명령어의 종류에 따라서 결정되는데, 분기 명령어(branch instruction)는 분기 유닛(branch unit)으로, 정수 연산 명령어(integer arithmetic instruction)는 정수 연산 유닛(integer arithmetic unit)으로, 메모리(memory)를 엑세스(access)하는 명령어는 로드/스토어 유닛(load/store unit)으로 이슈된다. 명령어 이슈는 비순차적으로 이루어질 수 있기 때문에 선행 명령어가 이슈되지 않은 상태로 명령어 이슈 큐에 존재하더라도 후행하는 명령어가 먼저 이슈되고, 명령어 이슈 큐에서 빠져나갈 수 있다. 명령어가 이슈된 후에 해당 엔트리는 비어있게 되고, 명령어 이슈는 비순차적으로 이루어지기 때문에 명령어 이슈 후에는 중간에 비워 있는 엔트리들을 이슈되지 않은 엔트리들을 사용하여서 압축(compressing)할 필요가 있다. 예를 들어서 명령어 이슈 후에 최상위 포인터와 최하위 포인터 값의 차이가 7이고, 최상위 엔트리와 최하위 엔트리 사이에 유효하게 명령어를 가지고 있는 엔트리가 세 개일 때에는 유효한 엔트리들을 최상위 엔트리쪽으로 최대한 천이(shift)시키고, 최하위 포인터 값을 조정하여, 최상위 포인터와 최하위 포인터 값이 차이가 3이 되도록 하며, 최상위 엔트리와 최하위 엔트리 사이에 존재하는 모든 엔트리들은 유효하게 명령어를 포함하도록 한다.

기능 유닛으로 이슈된 명령어들은 명령어 수행을 완료한 후에는 그 결과와 조건 레지스터 결과를 각각 물리적 목적 레지스터와 물리적 조건 레지스터에 쓰면서, 리오더 버퍼에게 명령어가 완료되었음을 알려준다. 후에 해당 엔트리가 리오더 버퍼의 최하위 엔트리가 되었을 때에는 그 엔트리는 커미트되고, 대응하는 명령어의 수행 결과를 순차적 상태로 만든다.

이제부터는 명령어 열의 예를 제시하고, 이 명령어 열을 이용하여, 본 발명의 제시하는 조건부 실행 비순차적 명령어 이슈 시스템에서의 레지스터 리네이밍, 물리적 레지스터의 할당 및 퇴거와 순차적 상태의 추적을 설명한다.

도 2는 본 발명의 방법을 설명하기 위해서 인위적으로 만들어진, 그러나 프로그램 수행의 일부분이 될 수 있는 6개의 명령어 열을 보여준다. 도 2에서는 첫째 명령어(200)가 레지스터 리네이밍을 수행할 단계의 상황에서, 논리적 레지스터 R2와 논리적 레지스터 R3은 최신의 물리적 레지스터 할당에 의해 각각 물리적 레지스터 PR1과 물리적 레지스터 PR8을 할당받았으며, 조건 레지스터의 갱신에 대한 최신의 물리적 조건 레지스터 할당은 물리적 조건 레지스터 PCR2를 사용하였다는 것을 가정한다. 그리고, 물리적 레지스터 중에서 주소 값이 낮은 것부터 차례대로, PR2, PR4, PR5, PR7, PR9가 프리하다는 것과, 도 2의 여섯째 명령어(250)에 대한 레지스터 리네이밍 때까지 PR0부터 PR9까지의 물리적 레지스터 중에서 새로이 프리하게 되는 것이 없다고 가정한다. 그 외에, 조건부 실행 명령어인 도 2의 둘째 명령어(210)는 실제로 실행될 것으로 예측되고, 도 2의 다섯째 명령어(240)는 실제로 실행되지 않을 것(비실행)으로 예측된다. 그리고, 도 2의 둘째 명령어(210)에 대한 실행 예측은 맞았지만, 도 2의 다섯째 명령어(240)에 대해서는 실행 예측 오류가 발생한다는 것을 가정한다.

상기 가정 하에서 도 2의 여섯 명령어들(200 내지 250)의 논리적 레지스터에 대한 리네이밍은 도 2의 가운데 열(260)과 같으며, 논리적 조건 레지스터에 대한 리네이밍은 도 2의 맨 오른쪽 열(270)과 같다. 첫째 명령어(200)는 무조건부 실행(unconditional execution) 명령어로서, 논리적 레지스터 R2에서 논리적 레지스터 R3을 빼서 그 결과를 논리적 레지스터 R1에 저장하고, 그 결과에 따라서 조건 레지스터의 값을 설정하는 것이다. 조건 레지스터는 N, Z, V, C 네 개의 비트로 이루어진 레지스터이다. 첫째 명령어(200)의 논리적 목적 레지스터 R1은 이전의 쓰기에 의한 R1의 값과 구분되기 위해서 물리적 레지스터를 할당받아야 한다. 물리적 레지스터를 할당할 때에는 물리적 레지스터 할당 지시기에 의해서 프리한 것으로 지시된 것 중에서 주소 값이 가장 작은 것을 할당한다. 따라서 상기 가정에 의해서, 도 2의 첫째 명령어(200)의 논리적 목적 레지스터 R1은 물리적 목적 레지스터 PR2를 할당받는다. 그리고 상기 가정에 의해서 도 2의 첫째 명령어(200)의 논리적 소스 레지스터 R2와 R3은 각각 물리적 소스 레지스터 PR1과 PR2로 리네이밍된다. 도 2의 첫째 명령어(200)는 명령어 수행 결과 값에 따라서 조건 레지스터를 갱신하는 명령어이다. 따라서 조건 레지스터의 갱신에 물리적 조건 레지스터를 할당한다. 물리적 조건 레지스터는 순환적으로 할당되고 상기 가정에 의해서 최신으로 할당된 것이 PCR2이므로, 도 2의 조건 레지스터 갱신에 할당되는 물리적 조건 레지스터는 PCR3이 된다. 도 2의 둘째 명령어(210)는 이전 명령어의 무부호(unsigned) 연산에서 실제 결과(캐리를 고려한 결과)가 0보다 크면(조건 레지스터의 비트 중에서 C는 1이고 Z는 0일 때) 실행되는 조건부 실행 명령어이다. 상기 도 2에 대한 가정에 의해서 도 2의 둘째 명령어(210)는 실제로 실행될 것으로 예측되었기 때문에, 무조건부 실행 명령어처럼 리네이밍이 이루어진다. 도 2의 논리적 목적 및 소스 레지스터에 대한 레지스터 리네이밍은 도 2의 첫째 명령어(200)의 레지스터 리네이밍과 같은 원리에 의해서 이루어지며, 그 결과는 도 2의 가운데 열과 같다. 도 2의 둘째 명령어(210)는 조건부 실행 명령어이기 때문에 조건 레지스터 값에 따라서 실제 수행 여부가 결정된다. 따라서 도 2의 둘째 명령어(210)의 수행에는 논리적 조건 레지스터에 대한 읽기 과정이 수반되는데, 이것은 적절한 물리적 조건 레지스터의 읽기로 변환되어야 한다. 도 2의 첫째 명령어(200)에 의해서 최신의 조건 레지스터 갱신이 이루어지며, 해당 조건 레지스터 갱신은 물리적 조건 레지스터 PCR3을 할당받았기 때문에 도 2의 둘째 명령어(210)의 논리적 조건 레지스터는 물리적 조건 레지스터 PCR3으로 리네이밍된다. 무조건부 실행 명령어인 도 2의 셋째 명령어(220)와 넷째 명령어(230)에 대한 레지스터 리네이밍은 도 2의 첫째 명령어(200)에 대한 것과 같은 원리로 이루어지며, 다만 도 2의 셋째 명령어(220)는 조건 레지스터를 갱신하는 명령어가 아니기 때문에 물리적 조건 레지스터의 할당은 이루어지지 않는다. 도 2의 다섯째 명령어(240)는 도 2의 넷째 명령어(230)의 수행 결과에 의한 조건 레지스터 값에 의존하는 조건부 실행 명령어이다. 상기 가정에 의해서 도 2의 다섯째 명령어(240)는 비실행 예측되었기 때문에, 도 2의 다섯째 명령어(240)가 실

행되지 않는다는 가정에 의한 레지스터 리네이밍 동작이 이루어진다. 따라서 도 2의 다섯째 명령어(240)에 대해서는 물리적 목적 레지스터 할당도 이루어지지 않으며, 논리적 소스 레지스터의 레지스터 리네이밍도 수행되지 않는다. 다만 비실행 예측이 맞는지 아니면 예측 오류인지를 검사하기 위해서는 조건 레지스터의 값을 읽어야 하며, 논리적 조건 레지스터의 읽기는 물리적 조건 레지스터 PCR4로 리네이밍된다. 도 2의 여섯째 명령어(250)에 대한 레지스터 리네이밍은 도 2의 셋째 명령어(220)의 것과 같은 원리에 의해서 이루어진다.

도 3은 명령어에 대한 실행 예측 또는 비실행 예측이 명령어 디코드 단계에서 이루어지는 과정을 보여준다.도 3의 해싱(hashing)은 명령어의 주소를 입력으로 하고, 조건 예측 버퍼의 엔트리 주소를 출력으로 하는 다대 일 대응 함수이다. 명령어 주소의 비트 수를 32비트라고 했을 때, 명령어의 주소 집합은 232의 원소로 이루어진 집합이다. 그리고, 도 3의 조건 예측 버퍼는 L개의 엔트리들을 가지므로, 도 3의 조건 예측 버퍼 엔트리의 주소 집합은 L개이다. 해싱이 이루어지기 위해서는, 조건 예측 버퍼 주소 집합의 수 L이 명령어 주소의 집합 232보다 작아야 한다 (즉 (log2L)이 32보다 작아야 한다). 이와 같은 조건에서, 해싱은 도 3의 명령어 주소 집합의 원소들을 조건 예측 버퍼 엔트리 주소의 집합의 원소들로 대응시키는 다대 일 대응 함수이며, 실제 구현은 다양하게 할 수 있다. 다만 해싱을 구현할 때에는 단일 조건 예측 버퍼 엔트리 주소 값에 대응하는 명령어 주소 값들의 수가 모든 조건 예측 버퍼 엔트리 주소 값들에 대해서 같도록 하는 것이 좋다. 그 예로서, 명령어 주소 32비트 중에서 하위 (log2L)비트만 뽑아내는 해싱이 있다.

도 3의 조건 예측 버퍼의 각 엔트리는 조건부 실행 명령어의 실행 또는 비실행 예측을 위한 카운터 값을 가지고 있는데, 도 3에서 각 카운터 값은 K비트로 이루어졌다고 가정한다.

도 3에서 명령어 디코드 유닛이 디코딩에 의해 각 명령어를 오피 코드와 논리적 목적 레지스터 및 논리적 소스 레지스터들로 변환하면서 각 명령어가 조건부 실행 명령어인지 무조건부 실행 명령어인지를 판별한다. 그 결과 명령어가 무조건부 실행 명령어이면 당연히 그 명령어는 실행되는 것으로 예측되며, 이 예측은 항상 정확하다. 명령어가 조건부 실행 명령어인 경우에는 명령어의 주소를 해싱한 결과 값을 주소로 하는 조건 예측 버퍼 엔트리의 카운터 값을 읽어서 그 값이 기준값 이상인 경우에는 명령어가 실행되는 것으로 예측하며, 카운터 값이 기준값보다 작으면 실행되지 않는 것으로 예측한다. 여기서, 기준값은 카운터의 비트 수에 따라서 달라지며, 카운터의 최상위 비트가 1이고 그 외의 비트들은 0인 무부호 수이다. 예를 들어 카운터가 2비트로 이루어진 경우에는 카운터의 기준값은 2진수 10이 되며, 카운터 값이 2진수 10보다 작을 경우에는 비실행으로 예측되고 카운터 값이 2진수 10과 같거나 더 크면 명령어가 실행되는 것으로 예측된다.

도 3의 조건 예측 버퍼 엔트리들의 카운터 값은 항상 일정한 값을 가지고 있는 것이 아니라, 역동적으로 변화하는 값들이다. 즉 조건부 실행 명령어의 명령어 수행에 의해서 조건을 검사한 결과 실제로 실행되는 것으로 판별된 경우에는 그 명령어에 대응하는 조건 예측 버퍼 엔트리의 카운터 값을 1만큼 증가시키며, 실제로 실행되는 않는 것으로 판별된 경우에는 카운터 값을 1만큼 감소시킴으로써, 실제 실행 여부의 과거 결과를 기반으로 조건부 실행 명령어의 실행 여부를 예측하도록 한다. 단, 카운터의 값이 최고값(모든 비트가 1인 값)인 경우에는 카운터가 값이 더 이상 증가하지 않으며, 최소값(모든 비트가 0인 값)인 경우에는 카운터 값이 더 이상 감소하지 않는다. 여기서 명령어에 대응하는 조건 예측 버퍼 엔트리란, 조건부 실행 명령어의 실행 여부를 예측하기 위해서 사용된 엔트리로서, 명령어 주소 값을 입력으로 한 해싱에 의한 출력 값을 주소로 하는 조건 예측 버퍼 엔트리이다.

도 4는 명령어의 논리적 레지스터들에 대한 리네이밍 과정을 도 2의 둘째 명령어(210) 예에 적용하여 보여준다. 도 4의 물리적 레지스터 할당 지시기는 물리적 레지스터 한 개당 한 개의 엔트리를 가지고 있으며, 도 4에서는 물리적 레지스터의 수를 M개로 가정하고 있다. 도 4의 리네임 상태 지시기는 논리적 레지스터 한 개당 한 개의 엔트리를 가지고 있으며, 도 4에서는 논리적 레지스터의 수를 N개로 가정한다. 논리적 레지스터는 명령어 세트 구조에 의해 정해지는 레지스터이기 때문에 그 수도 역시 명령어 세트 구조에 의해서 결정된다. 물리적 레지스터로 이루어진 레지스터 파일은 논리적 레지스터들의 순차적 상태에 해당하는 인스턴스와, 미리보기 상태에 해당하는 인스턴스를 모두 포함하기 때문에 물리적 레지스터의 수는 논리적 레지스터의 수보다 많아야 한다.

물리적 레지스터 할당 지시기의 각 엔트리는 대응하는 물리적 레지스터가 논리적 레지스터의 인스턴스로서 할당되어 있는지를 알려주는 비트인 프리 비트(F)로 이루어졌다. 물리적 레지스터 할당 지시기의 엔트리 값이 1일 때에는 대응하는 물리적 레지스터는 프리하여 새로운 물리적 레지스터 할당에 사용될 수 있다는 것을 의미하며, 엔트리의 값이 0일 때에는 대응하는 물리적 레지스터가 어떤 논리적 레지스터의 리네임 인스턴스로서 할당되어 있기 때문에 새로운 물리적 레지스터 할당에 사용할 수 없다는 것을 의미한다.

도 4의 우선권 인코더(encoder with priority)는 물리적 레지스터 할당 지시기의 모든 엔트리들의 값들을 입력으로 받아서 값이 1인 것 중에서 그 주소 값이 가장 작은 엔트리의 주소 값을 출력한다. 이렇게 함으로써, 우선권 인코더는 프리한 물리적 레지스터 중에서 그 주소 값이 가장 작은 물리적 레지스터의 주소 값을 출력하는 역할을 한다.

도 4의 리네임 상태 지시기의 각 엔트리는 대응하는 논리적 레지스터에 대한 최신의 리네임 인스턴스에 의해 할당된 물리적 레지스터의 주소 값(PRA)을 가지고 있다. 예를 들어서 리네임 상태 지시기의 첫째 엔트리(주소가 0)는 R0에 대한 최신의 리네임 인스턴스에 의해 할당된 물리적 레지스터 주소 값을 가지고 있기 때문에 첫째 엔트리의 값을 읽음으로써, 소스 레지스터 R0에 대한 올바른 값을 읽기 위한 물리적 레지스터의 주소 값을 알 수 있다. 따라서 소스 레지스터에 대한 리네이밍은 리네임 상태 지시기에 의해서 이루어진다.

도 2에서의 가정에 의해서 조건부 실행 명령어인 도 2의 둘째 명령어(210)는 실행될 것으로 예측되어 실행 예측 비트는 1로 세팅되므로, 도 2의 둘째 명령어(210)에 대해서는 무조건부 명령어의 경우처럼 레지스터 리네이밍이 이루어진다. 그리고, 도 2의 둘째 명령어(210)를 리네이밍할 때에는 물리적 레지스터 PR0부터 물리적 레지스터 PR3까지는 프리하지 않으며, 물리적 레지스터 PR4는 프리하다. 따라서 물리적 레지스터 할당 지시기의 첫째 엔트리부터 넷째 엔트리까지의 프리 비트 값은 0이고, 물리적 레지스터 PR4에 대응하는 다섯째 엔트리의 프리 비트 값은 1이다. 그 결과 우선권 인코더는 물리적 레지스터 PR4의 주소 값인 4를 출력하며, 도 2의 논리적 목적 레지스터 R1은 물리적 목적 레지스터 PR4를 할당받는다. 물리적 레지스터의 할당과 함께, 물리적 레지스터 PR4는 더 이상 프리하지 않기 때문에 물리적 레지스터 PR4에 대응하는 물리적 레지스터 할당 지시기 엔트리의 프리 비트는 0으로 바뀐다. 그리고, 논리적 레지스터 R1에 대한 최신의 리네임 인스턴스는 물리적 레지스터를 PR4로 하기 때문에, 논리적 레지스터 R1에 대응하는 리네임 상태 지시기 엔트리의 값은 물리적 레지스터 PR4의 주소 값으로 바뀐다. 도 4의 명령어의 논리적 소스 레지스터 R2와 논리적 소스 레지스터 R3에

대해서는 각 논리적 레지스터에 대한 최신의 리네임 인스턴스인 물리적 레지스터로 리네임해 주어서, 각각의 논리적 소스 레지스터의 올바른 데이터를 읽을 수 있게 해야 한다. 논리적 소스 레지스터 R2에 대해서 최신의 리네임 인스턴스로서의 물리적 레지스터는, 도 2에서의 가정에 의해서 물리적 레지스터 PR1이고, 논리적 소스 레지스터 R3에 대해서는 물리적 레지스터 PR8이다. 이러한 가정은 논리적 레지스터 R2와 R3에 대응하는 도 4의 리네임 상태 지시기 엔트리들의 값에 반영되어 있다. 따라서 논리적 레지스터 R2와 R3에 대한 리네이밍은 각 논리적 레지스터에 대응하는 리네임 상태 지시기 엔트리들의 값인 PR1의 주소 값인 1과 PR8의 주소 값인 8을 읽어냄으로써 이루어진다. 결과적으로 논리적 레지스터 R2와 R3은 각각 물리적 레지스터 PR1과 PR8으로 리네이밍된다.

도 4의 레지스터 리네이밍에 의해서 알아낸 물리적 레지스터 주소는 명령어 이슈 큐와 리오더 버퍼로 전달되어 저장된다. 그리고, 해당 명령어의 실행 예측 비트(P)도 명령어 이슈 큐와 리오더 버퍼의 같은 엔트리들에 저장되어, 해당 엔트리의 명령어가 실행 예측되었는지 또는 비실행 예측되었는지를 알려준다. 그 외에 2개의 물리적 소스 레지스터 주소들은 명령어 이슈 큐로, 그리고 논리적 목적 레지스터 주소는 리오더 버퍼로 각각 저장된다.

도 5는 도 2의 첫째 명령어(200)를 예로서 사용하여 레지스터 리네이밍 동작 중에서 조건 레지스터에 대한 리네이밍 동작을 보여 준다. 도 5의 물리적 조건 레지스터 할당 지시기는 조건 레지스터의 갱신에 할당할 물리적 조건 레지스터의 주소 값을 가지는 것으로서, 물리적 조건 레지스터의 할당이 이루어질 때마다 그 값이 1씩 증가한다. 도 5의 물리적 조건 레지스터 할당 카운터는 조건 레지스터의 갱신에 할당된 물리적 조건 레지스터의 수를 저장하고 있으며, 물리적 조건 레지스터 할당 카운터에 저장된 값이 컴퓨터 시스템에 존재하는 조건 레지스터 어레이 내의 물리적 조건 레지스터의 수와 같을 경우에는 더 이상의 물리적 조건 레지스터 할당이 이루어질 수 없으며, 해당 명령어는 레지스터 리네이밍 단계에서 정지(stall)되고 이미 할당되어 있던 물리적 조건 레지스터가 프리하게 되기를 기다린다. 도 5의 리네임 조건 상태 지시기는 최신의 조건 레지스터 갱신에 할당된 물리적 조건 레지스터의 주소 값을 가지고 있다. 리네임 조건 상태 지시기의 값은 조건 레지스터 갱신에 대한 물리적 조건 레지스터 할당이 이루어질 때마다 할당된 물리적 조건 레지스터의 주소 값으로 갱신된다.

도 2의 가정에 의해서, 첫째 명령어(200)가 레지스터 리네이밍을 수행할 단계에서 최신의 조건 레지스터 갱신에 할당된 물리적 조건 레지스터는 PCR2이다. 물리적 조건 레지스터 할당 지시기는 순환적으로 동작하며, 할당이 이루어질 때마다 그 값이 1씩 증가하므로, 도 2의 첫째 명령어(200)의 레지스터 리네이밍 단계에서는 그 값이 물리적 조건 레지스터 PCR3의 주소 값인 3을 저장하고 있다. 따라서 도 2의 첫째 명령어(200)의 조건 레지스터 갱신에 할당되는 물리적 조건 레지스터는 PCR3이며, 이것은 도 2의 물리적 조건 레지스터 할당 지시기의 값인 PCR3의 주소 값을 읽어냄으로써 이루어진다. 그리고, 물리적 조건 레지스터의 할당이 이루어졌으므로, 물리적 조건 레지스터 할당 지시기의 값은 1만큼 증가하여 물리적 조건 레지스터 PCR4의 주소 값 4로 갱신되고, 물리적 조건 레지스터 할당 카운터의 값도 1만큼 증가한다. 도 5의 리네임 상태 지시기의 값도 물리적 조건 레지스터 PCR3의 할당에 따라서 PCR3의 주소 값인 3으로 갱신된다. 도 2의 첫째 명령어(200)는 조건 레지스터를 갱신하는 명령어이기는 하지만, 명령어 수행을 위해서 조건 레지스터의 값을 읽는 명령어는 아니다. 따라서 조건 레지스터의 읽기에 대한 리네이밍은 수행할 필요가 없으며, 리네임 조건 상태 지시기로부터 출력되는 최신의 조건 레지스터 갱신에 대응하는 물리적 조건 레지스터 PCR2의 주소 값인 2는 쓸모가 없다. 그러나 도 2의 둘째 명령어(210)와 같이 명령어 수행을 위해서 조건 레지스터를 읽는 명령어는 도 5의 리네임 상태 지시기의 값으로부터 최신의 조건 레지스터 갱신에 해당하는 물리적 조건 레지스터의 주소 값을 읽어내어 명령어 수행에 사용한다.

도 5의 조건 레지스터에 대한 리네이밍에 할당된 물리적 조건 레지스터(물리적 목적 조건 레지스터)의 주소 값은 명령어 이슈 큐와 리오더 버퍼의 해당 엔트리에 저장된다. 그리고, 조건 레지스터를 읽는 명령어에 대해서는 해당 명령어 이전의 상황에서 최신의 조건 레지스터 갱신에 대응하는 물리적 조건 레지스터(물리적 소스 조건 레지스터)의 주소 값을 명령어 이슈 큐의 해당 엔트리에 저장한다.

도 6은 명령어 이슈 큐와 리오더 버퍼의 구성을 보여주며, 도 2의 여섯째 명령어(250)를 레지스터 리네이밍한 후 명령어 이슈 큐와 리오더 버퍼로 저장할 때의 상황이다.

명령어 이슈 큐는 최상위 포인터와 최하위 포인터를 사용하여 순환적으로 엔트리를 할당 및 제거한다. 최상위 포인터는 새로 입력되는 명령어를 저장할 엔트리를 가리키며, 최하위 포인터는 명령어 이슈 큐 엔트리 중에서 가장 오래된 엔트리를 가리킨다. 따라서 유효한 명령어를 가지고 있는 명령어는 최상위 포인터와 최하위 포인터 사이에 존재한다. 명령어 이슈가 비순차적으로 이루어지기 때문에 선행하는 엔트리보다 후행하는 엔트리가 먼저 이슈될 수 있다. 따라서 최상위 엔트리와 최하위 엔트리 사이의 명령어가 이슈되어서 더 이상 유효한 명령어를 가지고 있지 않은 엔트리들이 중간에 존재할 수 있다. 이와 같은 이유 때문에 명령어 이슈와 함께 명령어 이슈 큐를 압축하는 과정이 이루어져야 한다. 명령어 이슈 큐의 압축이란 유효한 엔트리들을 최상위 엔트리 쪽으로 천이시키고, 압축의 양에 따라서 최하위 포인터 값을 조정하는 것을 의미한다. 예를 들어서 최상위 엔트리와 최하위 엔트리 사이에 7개의 엔트리가 존재하는데 그 중에서 유효한 엔트리는 3개뿐일 경우에 3개의 유효한 엔트리들을 모두 최상위 엔트리 쪽으로 천이시키고, 최하위 포인터는 천이된 유효한 엔트리 중에서 가장 오래된 엔트리를 가리키도록 한다. 이 경우 최하위 포인터와 최상위 포인터는 값이 차이가 3이 된다.

명령어 이슈 큐의 각 엔트리는 그 엔트리가 유효한 명령어를 가지고 있는지를 지시하는 유효 비트(V), 실행 예측 비트(P), 내부적인 오피 코드, 물리적 목적 레지스터 주소(PDRA), 첫째 물리적 소스 레지스터 주소(PSRA1), 둘째 물리적 소스 레지스터 주소(PSRA2), 물리적 목적 조건 레지스터 주소(PDCRA), 물리적 소스 조건 레지스터 주소(PSCRA) 및 리오더 버퍼 엔트리에 대한 포인터(RBP)를 저장하고 있다. 오피 코드는 명령어 디코드 과정에서 만들어진 것을 저장한 것이고, 물리적 목적 및 소스 레지스터 주소들은 도 4의 레지스터 리네이밍 과정에 의해서 얻어진 값들이며, 물리적 목적 및 소스 조건 레지스터 주소들은 은 도 5의 조건 레지스터에 대한 리네이밍 과정에 의해서 얻어진 값들이다. 명령어가 레지스터 리네이밍을 거쳐서 명령어 이슈 큐에 저장되면서 그 명령어의 논리적 목적 레지스터, 물리적 목적 레지스터 및 물리적 목적 조건 레지스터의 주소들이 리오더 버퍼에 저장된다. 명령어 이슈 큐 엔트리에 존재하는 리오더 버퍼 포인터는 해당 명령어에 대한 목적 레지스터 주소들을 저장한 리오더 버퍼 엔트리의 위치를 가리키는 것이다.

리오더 버퍼는 명령어 이슈 큐와 같이 최상위 포인터 및 최하위 포인터에 의해 순환적으로 다루어진다. 그러나 리오더 버퍼는 선입선출 형식이기 때문에 최상위 엔트리와 최하위 엔트리 사이에 유효하지 않은 엔트리가 존재할 수 없으며, 따라서 압축이 필요하지 않다.

리오더 버퍼의 각 엔트리는 대응하는 명령어의 논리적 목적 레지스터 주소(LDRA), 물리적 목적 레지스터 주소(PDRA), 물리적 목적 조건 레지스터 주소(PDCRA), 명령어의 주소(PC), 조건 레지스터 갱신 명령어인지를 알려주는 조건 갱신 비트(U), 조건부 실행 명령어인지를 알려주는 조건부 비트(C), 실행 예측 비트(P) 및 명령어 수행 결과로서 실제로 실행되는 명령어인지를 알려주는 실행 결과 비트(R) 및 해당 명령어의 수행이 완료되었음을 알려주는 완료 비트(W)로 구성되어 있다. 논리적 목적 레지스터 주소, 조건 갱신 비트, 조건부 비트 및 실행 예측 비트는 명령어 디코드 단계에서 만들어지고, 물리적 목적 레지스터 주소와 물리적 목적 조건 레지스터 주소는 도 4와 도 5의 레지스터 리네이밍에 의해서 생성되며, 명령어 주소는 명령어 페치 때의 주소이다. 이렇게 생성된 값들은 리오더 버퍼의 최상위 엔트리에 저장한다. 그 외의 영역인 실행 결과 비트는 명령어가 리오더 버퍼에 저장될 때에는 0이고, 명령어 수행에 의해 최종 값이 결정된다. 완료 비트는 명령어가 리오더 버퍼에 저장될 때에는 0이며, 명령어 수행이 끝나면서 1로 세팅된다. 명령어의 레지스터 리네이밍까지는 순차적 방식으로 이루어지고, 리오더 버퍼는 레지스터 리네이밍을 거친 명령어의 목적 레지스터 주소들을 순차적으로 저장하기 때문에 리오더 버퍼에서 각 엔트리의 순서는 대응하는 명령어들의 순차적인 순서와 같다. 즉 리오더 버퍼에서 엔트리들의 순서는 명령어들의 순차적인 순서와 같으며, 최하위 포인터는 명령어의 순차적인 순서에 의해서 가장 오래된 엔트리이다.

리오더 버퍼 엔트리의 영역 중에서 완료 비트는 그 엔트리에 대응되는 명령어의 수행 결과가 저장되었음을 의미한다. 어떤 엔트리가 리오더 버퍼의 최하위 엔트리가 되었을 때 완료 비트가 1인 경우, 그 엔트리는 커미트된다. 커미트되는 최하위 엔트리에 대응하는 명령어가 무조건부 실행 명령어인 경우(즉, 조건부 비트가 0인 경우), 최하위 엔트리의 물리적 목적 레지스터 주소 값에 대응되는 물리적 레지스터는 해당 엔트리의 논리적 목적 레지스터 주소 값에 대응되는 논리적 레지스터의 순차적 상태를 저장하는 물리적 레지스터가 된다. 그리고, 최하위 엔트리의 물리적 목적 조건 레지스터 주소 값에 대응되는 물리적 조건 레지스터는 조건 레지스터의 순차적 상태를 저장하는 물리적 조건 레지스터가 된다. 이렇게 함으로써, 순차적 상태를 추적할 수 있는 이유는 다음과 같다. 어떤 엔트리가 최하위 엔트리에 도착했다는 것은 이전의 모든 엔트리들이 모두 커미트되었다는 것을 의미한다. 따라서 최하위 엔트리에 대응되는 명령어가 수행을 완료했을 때에는 최하위 엔트리에 대응되는 명령어까지 연속적으로 그 수행이 완료되었으므로, 최하위 엔트리에 대응되는 명령어의 수행 결과를 순차적 상태로 저장할 수 있다. 명령어의 수행 결과를 순차적 상태로 저장한다는 것은 명령어의 물리적 목적 레지스터와 물리적 목적 조건 레지스터를 순차적 상태에 포함시킨다는 것을 의미하며, 이 작업이 리오더 버퍼의 최하위 엔트리를 커미트할 때 이루어진다.

도 6은 도 2의 여섯째 명령어(250)가 명령어 이슈 큐와 리오더 버퍼에 저장되는 모습을 보여준다. 그리고 도 6의 리오더 버퍼의 완료 비트들의 값을 보면 도 2의 여섯째 명령어(250)가 리오더 버퍼에 저장될 때에 도 2의 첫째 및 둘째 명령어(200, 210)들은 동작 수행이 완료되었음을 알 수 있다. 여섯째 명령어(250)는 명령어 이슈 큐의 최상위 포인터가 가리키는 엔트리에 저장되고, 리오더 버퍼에서도 최상위 포인터가 가리키는 엔트리에 저장된다. 도 6에서 도 2의 첫째 명령어부터 다섯째 명령어(200 내지 240)까지는 이미 저장되어 있는 모습을 보여 준다. 또한 도 2에서 알 수 있듯이 명령어 이슈 큐 엔트리의 리오더 버퍼 포인터 값은 각 엔트리에 대응되는 리오더 버퍼 엔트리의 주소 값으로 설정된다. 그리고, 명령어 이슈 큐와 리오더 버퍼에 엔트리를 저장하고서는 각각의 최상위 포인터는 1씩 증가하여, 새로운 명령어를 받아들일 엔트리를 가리키게 된다.

도 7은 명령어 이슈 큐로부터 기능 유닛으로 명령어를 이슈하는 모습 중에서 도 2의 둘째 명령어(210)를 이슈하는 모습을 보여 준다. 명령어 이슈 큐에 존재하는 명령어들 중에서 소스 레지스터들 및 조건 레지스터의 데이터가 사용 가능하고, 명령어를 수행할 종류의 기능 유닛이 사용 가능할 때에 그 명령어는 기능 유닛으로 이슈된다. 단 비실행 예측된(실행 예측 비트가 0인) 명령어에 대해서는 명령어 실행이 이루어지지 않고, 단지 실행 예측의 정확성만을 검사(비실행 예측된 것이 맞는지에 대한 검사)하기 때문에 이러한 명령어의 이슈 시에는 소스 레지스터들의 데이터 사용 가능 여부를 검사하지 않는다. 명령어 이슈는 상기 기술했듯이 비순차적으로 이루어진다.

명령어를 이슈할 때에는 기능 유닛이 수행할 동작을 알려주기 위해서 해당 명령어 이슈 큐 엔트리의 오피 코드를 기능 유닛에 전달한다. 그리고 실행 예측 비트(P)를 전달하여, 실행 예측 비트가 0인 경우(비실행 예측인 경우)에는 오피 코드에 의해서 주어진 동작들 수행하지도 않고, 수행 결과를 물리적 레지스터나 물리적 조건 레지스터에 저장하지도 못하게 한다. 비실행 예측된 조건부 실행 명령어는 물리적 목적 레지스터나 물리적 목적 조건 레지스터를 할당받지 못했으므로 이와 같이 하는 것은 당연하다. 그리고, 이슈되는 명령어에 대응되는 리오더 버퍼 엔트리의 주소인 리오더 버퍼 포인터도 기능 유닛으로 전달되어 저장된다.

상기 오피 코드, 실행 예측 비트 및 리오더 버퍼 포인터 외에, 실행 예측된 조건부 실행 명령어나 무조건부 명령어의 경우(실행 예측 비트가 1인 경우)에는 다음과 같은 것들이 명령어 이슈 큐로부터 기능 유닛으로 전달된다. 첫째, 명령어 실행을 완료한 후에 그 결과를 저장할 물리적 레지스터의 주소가 명령어 이슈 큐 엔트리로부터 읽혀져서 기능 유닛에 전달되고, 조건 레지스터 갱신 명령어의 경우에는 조건 레지스터의 결과를 저장할 물리적 조건 레지스터의 주소도 전달된다. 그리고, 기능 유닛이 동작을 수행하기 위해서는 동작의 대상이 되는 데이터가 필요하다. 이러한 데이터로서 2개의 소스 데이터를 가지며, 데이터 전달은 해당 명령어 이슈 큐 엔트리의 물리적 소스 레지스터 주소 값들을 이용하여 레지스터 파일에서 소스 데이터들을 읽어내어 기능 유닛에 전달함으로써 이루어진다. 또한 조건 레지스터를 읽는 명령어에 대해서는 명령어 이슈 큐 엔트리의 물리적 소스 조건 레지스터 주소를 이용하여 조건 레지스터 어레이로부터 소스 조건 데이터를 읽어내어 기능 유닛에 전달한다.

비실행 예측된 조건부 실행 명령의 경우(실행 예측 비트가 0인 경우) 상기 오피 코드, 실행 예측 비트 및 리오더 버퍼 포인터 외에, 소스 조건 데이터만이 기능 유닛으로 전달되어 저장된다. 이것은 실행 예측의 정확성을 검사하기 위한 것이다.

도 7에서 이슈되는 명령어인 도 2의 둘째 명령어(210)는 실행 예측된(P가 1인) 조건부 실행 명령어이므로, 실행 예측 비트, 오피 코드, 물리적 목적 레지스터 주소, 소스 데이터, 소스 조건 데이터 및 리오더 버퍼 포인터의 전달이 명령어 이슈 과정에서 이루어진다. 다만, 도 7에서 이슈되는 명령어는 조건 레지스터를 갱신하는 명령어가 아니므로, 물리적 목적 조건 레지스터의 주소 값은 의미가 없으며, 그 값에 대한 전달도 이루어지지 않는다.

도 8은 기능 유닛에서 도 2의 넷째 명령어(230)를 수행하고서 그 결과를 레지스터 파일의 해당 물리적 레지스터(PR7)에, 그리고 조건 레지스터 결과를 해당 물리적 조건 레지스터(PCR4)에 쓰는 과정을 보여준다. 기능 유닛이 실행 예측된 조건

부 실행 명령어나 무조건부 실행 명령어의 수행을 끝내어서 결과가 나오면 도 8과 같이 기능 유닛이 명령어 이슈 큐로부터 받았던 물리적 목적 레지스터 주소 값과 결과 데이터를 사용해서 레지스터 파일에 쓰기 작업을 하며, 조건 레지스터 갱신 명령어의 경우에는 물리적 목적 조건 레지스터 주소 값과 결과 조건 데이터를 사용해서 조건 레지스터 어레이의 해당 물리적 조건 레지스터를 갱신한다. 이와 동시에 기능 유닛은 해당 명령어가 끝났음을 리오더 버퍼에 알려 주다. 이것은 기능 유닛이 명령어 이슈 큐로부터 받았던 리오더 버퍼 포인터 값을 리오더 버퍼에 전달하고, 리오더 버퍼는 리오더 버퍼 포인터가 가리키는 엔트리의 완료 비트(W)를 0에서 1로 갱신함으로써 이루어진다. 그리고 해당 명령어의 실행 결과 비트(R)를 리오더 버퍼에 전달하여 대응되는 엔트리에 저장하도록 한다. 이것은 해당 엔트리의 커미트 시에 실행 예측 비트(P)와 실행 결과 비트의 비교를 통해 실행 예측이 맞았는지를 알기 위해서이다.

그러나, 비실행 예측된 조건부 실행 명령어에 대해서 기능 유닛은 해당 오피 코드에 대한 동작도 이루어지지 않으며, 결과를 저장하는 작업도 이루어지지 않는다. 다만 명령어의 소스 조건 데이터 값을 이용하여 실행 예측이 맞았는지를 검사한 후에, 대응되는 리오더 버퍼 엔트리에 명령어 수행이 완료되었다는 것을 알려주는 동작과 실행 결과 비트의 값을 전달하는 동작만이 수행된다.

도 9는 도 2의 둘째 명령어(210)를 예로 사용하여, 실행 예측된(실행 예측 비트가 1인) 조건부 실행 명령어가 리오더 버퍼의 최하위 엔트리가 되었을 때 커미트되는 과정을 보여 준다. 어떤 엔트리가 리오더 버퍼의 최하위 엔트리가 되었다는 것은 이전의 엔트리들은 모두 커미트되었다는 것을 의미한다. 따라서 이전의 엔트리까지의 명령어들은 모두 수행이 완료되었고, 이전의 명령어까지의 연속적인 명령어 열의 결과가 시스템의 순차적 상태를 형성하고 있음을 의미한다. 따라서 최하위 엔트리의 명령어가 수행을 완료하여 완료 비트가 1인 경우에는 최하위 엔트리까지의 연속적인 명령어 열은 모두 수행이 완료되었고, 따라서 최하위 엔트리에 대응되는 명령의 결과를 순차적 상태로서 저장할 수 있다.

도 9의 순차적 상태 지시기는 논리적 레지스터의 수와 같은 N개의 엔트리로 이루어져 있으며, 각 엔트리는 대응하는 논리적 레지스터의 순차적 상태를 저장하고 있는 물리적 레지스터의 주소 값(PRA)을 저장하고 있다.

실행 예측이 맞은 실행 예측된 조건부 실행 명령어는 무조건부 실행 명령어의 수행과 다를 것이 없다. 즉, 상기 기술에서 실행 예측된 조건부 실행 명령어는 레지스터 리네이밍, 명령어 이슈 및 기능 유닛에 의한 명령어 수행과 결과 저장에 있어서 무조건부 실행 명령어와 차이가 없었는데, 실행 예측된 조건부 실행 명령어의 실행 예측까지 맞은 경우에는 리오더 버퍼에서의 커미트도 무조건부 실행 명령어의 커미트와 같은 원리로 이루어진다. 도 9의 명령어가 실행 예측이 맞은 실행 예측된(실행 예측 비트와 실행 결과 비트가 1인) 명령어에 해당한다.

도 2를 참조해 보면, 둘째 명령어(210)의 레지스터 리네이밍에 의해서 논리적 레지스터 R1에 물리적 레지스터 PR4를 할당하기 이전의 물리적 레지스터 할당 중에서 논리적 레지스터 R1에 대한 최신의 물리적 레지스터 할당은 도 2의 첫째 명령어(200)에 의해서 이루어졌으며, 할당된 물리적 레지스터는 PR2이다. 도 9는 도 2의 둘째 명령어(210)가 커미트될 때의 모습이므로, 도 2의 둘째 명령어(210) 이전의 연속적인 명령어들은 모두 커미트된 상태이다. 따라서 도 2의 첫째 명령어(200)에 대응되는 리오더 버퍼 엔트리는 이미 커미트된 상태이다. 도 2의 첫째 명령어(200)는 무조건부 실행 명령어이기 때문에 도 2의 첫째 명령어(200)의 실행이 컴퓨터 시스템의 상태를 변화시키며, 그 변화 중의 하나가 커미트에 의한 레지스터의 순차적 상태 변화이다. 도 2의 첫째 명령어(200)는 논리적 목적 레지스터가 R1이고, 물리적 목적 레지스터는 PR2이므로 해당 리오더 버퍼 엔트리의 커미트에 의해서 논리적 레지스터 R1의 순차적 상태를 저장하는 물리적 레지스터는 PR2가 된다. 이것은 논리적 레지스터 R1에 대응되는 순차적 상태 지시기 엔트리가 그 값을 물리적 레지스터 PR2의 주소 값(2)으로 갱신함으로써 이루어지며, 그 값은 도 2의 둘째 명령어(210)의 리오더 버퍼 엔트리가 커미트될 때까지는 유지된다.

도 9에서 리오더 버퍼의 최하위 엔트리인 도 2의 둘째 명령어(210)는 수행이 완료되어 있는 상태이다. 따라서 도 2의 둘째 명령어(210)는 커미트될 수 있다. 해당 엔트리의 내용을 보면 논리적 목적 레지스터는 R1이며, 물리적 목적 레지스터는 PR4이다. 그리고, 도 2의 둘째 명령어(210)는 실행 예측된 명령어이므로 기능 유닛에서 명령어의 주어진 동작이 수행되었으며, 그 결과도 물리적 레지스터에 저장되어 있다. 또한 도 2의 둘째 명령어(210)는 실행 예측이 맞았으므로, 커미트에 의해서 명령어 실행 결과를 컴퓨터 시스템의 상태에 반영한다. 따라서 해당 엔트리가 커미트되면서 논리적 레지스터 R1에 대한 순차적 상태를 저장하는 물리적 레지스터는 이전의 PR2에서 도 2의 둘째 명령어(210)의 물리적 목적 레지스터이고, 명령어의 수행 결과를 저장하고 있는 PR4로 바뀌게 되며, 이것은 논리적 레지스터 R1에 대응되는 순차적 상태 지시기 엔트리의 값을 물리적 레지스터 PR2의 주소 값(2)에서 물리적 레지스터 PR4의 주소 값(4)으로 갱신함으로써 이루어진다.

도 9에서 물리적 레지스터 PR2가 논리적 레지스터 R1의 순차적 상태에서 축출되면서, 물리적 레지스터 PR2의 값은 더 이상 그 사용 용도를 가지지 못하게 된다. 물리적 레지스터 PR2는 논리적 레지스터의 순차적 상태를 저장하고 있는 것도 아니며, 물리적 레지스터 PR2를 참조하는 명령어들도 더 이상 존재하지 않는다. 따라서 물리적 레지스터 PR2는 퇴거시켜서 프리하게 함으로써 차후의 새로운 물리적 레지스터 할당에 사용할 수 있도록 해야 한다. 이것은 도 9에서 지시하듯이, 커미트되는 엔트리의 논리적 레지스터 R1에 대응되는 순차적 상태 지시기 엔트리의 커미트 이전의 값(PR2의 주소 값인 2)을 물리적 레지스터 할당 지시기에 전달하고, 물리적 레지스터 할당 지시기는 이 값을 주소로 하는 엔트리(PR2에 대응되는 엔트리)의 값을 0에서 1로 변경함으로써 이루어진다.

도 9의 명령어는 조건 레지스터를 갱신하는 명령어가 아니다. 따라서 커미트에 의해서 조건 레지스터의 상태가 변하지 않으므로, 조건 레지스터의 순차적 상태는 변화가 없다.

조건부 실행 명령어는 커미트되면서 실행 결과 비트에 따라서 조건 예측 버퍼의 해당 엔트리 값을 변경시킨다. 즉, 실제로 실행되어야 하는 명령어인 경우(실행 결과 비트(R)가 1인 경우)에는 조건 예측 버퍼의 해당 엔트리 값을 1만큼 증가시키며, 실제로 실행되지 않아야 하는 명령어인 경우(실행 결과 비트가 0인 경우)에는 조건 예측 버퍼의 해당 엔트리 값을 1만큼 감소시킨다. 도 9의 명령어는 실행 결과 비트가 1인 조건부 실행 명령어이므로 커미트되면서 조건 예측 버퍼의 해당 엔트리 값을 1만큼 증가시킨다. 단, 해당 엔트리의 카운터 값이 이미 최고값인 경우(모든 비트가 1인 경우)에는 카운터 값이 변화되지 않는다. 카운터 값을 변화시킬 조건 예측 버퍼의 해당 엔트리는 커미트되는 명령어에 대한 실행 예측을 수행할 때 사용한 엔트리이며, 커미트되는 엔트리의 명령어 주소 값(도9의 명령어에 대해서는 PC2)을 사용한 해싱의 결과 값을 주소 값으로 하는 조건 예측 버퍼 엔트리이다.

도 9는 실행 예측이 맞은 실행 예측된 조건부 실행 명령어의 커미트에 관한 것이며, 무조건부 실행 명령어의 커미트처럼 이루어진다. 그러나 실행 예측이 틀린 실행 예측된(실행 예측 비트가 1이고, 실행 결과 비트는 0인) 조건부 실행 명령어의 커미트는 다르게 동작한다. 이 경우에는 실행 예측이 틀렸기 때문에, 커미트되는 명령어에 대해서 이루어졌던 물리적 레지스터 할당도 틀렸으며, 커미트되는 명령어 이후의 명령어 중에서 커미트되는 명령어의 논리적 목적 레지스터를 논리적 소스 레지스터로 사용하는 명령어에 대한 레지스터 리네이밍도 틀려지게 된다. 따라서, 커미트되는 명령어를 포함해서 그 이후의 명령어들은 모두 취소시키고, 커미트되는 명령어 이전까지의 명령어 수행에 의한 순차적 상태를 사용해서 컴퓨터 시스템을 복구하고, 실행 예측이 틀린 명령어의 페치부터 다시 시작해야 한다. 실행 예측이 틀린 실행 예측된 조건부 실행 명령어의 커미트에 따르는 복구 작업은 다음과 같은 동작들을 포함한다.

첫째, 명령어 페치 큐, 명령어 이슈 큐 및 리오더 버퍼의 모든 엔트리들을 비워야 한다. 이것은 최상위 포인터와 최하위 포인터를 같은 값으로 설정하는 것에 의해 이루어지며, 명령어 이슈 큐의 경우에는 모든 엔트리들의 유효 비트를 0으로 만드는 작업이 수반된다.

둘째, 일반 레지스터에 대한 상태를 복구해야 한다. 자세히 설명해서, 순차적 상태 지시기의 각 엔트리들의 값을 대응되는 리네임 상태 지시기의 엔트리들로 복사한다. 그리고, 순차적 상태를 저장하고 있는 물리적 레지스터만을 프리하지 않은 것으로 한다. 순차적 상태를 저장하고 있는 물리적 레지스터들의 주소 값은 순차적 상태 지시기 엔트리들의 값에 의해서 알 수 있으며, 이 값들을 물리적 레지스터 할당 지시기에 전송하여, 대응되는 엔트리들만을 0(프리하지 않음)으로 하고, 나머지 엔트리들은 1(프리함)로 세팅한다. 이와 같이 함으로써, 차후의 명령어 수행에서 소스 레지스터 리네이밍에 의해, 논리적 소스 레지스터들은 순차적 상태를 저장하고 있는 물리적 레지스터들로 리네이밍되므로, 순차적 상태에서 명령어 수행을 다시 시작하는 효과를 볼 수 있다. 그리고 순차적 상태를 저장하지 않고 있던 물리적 레지스터들은 순차적 상태의 복구 후에는 참조되는 일이 없으므로 프리하게 만들어, 이 물리적 레지스터들을 사용하여 새로운 물리적 레지스터 할당을 수행할 수 있다.

셋째, 조건 레지스터에 대한 상태를 복구해야 한다. 즉, 순차적 조건 상태 지시기의 값을 리네임 조건 상태 지시기로 복사하고, 물리적 조건 레지스터 할당 지시기의 값은 순차적 조건 상태 지시기의 값에 1을 더한 값으로 변경시킴으로써, 순차적 상태를 저장하고 있는 물리적 조건 레지스터의 다음 주소를 갖는 물리적 조건 레지스터부터 할당할 수 있도록 한다. 그리고, 물리적 조건 레지스터 할당 카운터의 값을 1로 세팅함으로써, 조건 레지스터의 순차적 상태를 저장하고 있는 물리적 조건 레지스터를 제외한 물리적 조건 레지스터들은 프리하도록 한다.

마지막으로, 실행 예측이 틀린 명령어는 조건부 실행 명령어이므로 실행 결과 비트에 따라서 조건 예측 버퍼의 해당 엔트리의 값을 변경시킨다. 그리고, 명령어 페치를 위한 프로그램 카운터의 값을 커미트되는 명령어의 주소로 설정하여, 커미트되는 명령어부터 다시 페치하여 수행하도록 한다. 단, 해당 명령어의 실행 예측이 다시 틀리게 되는 것을 막기 위해서, 페치되는 첫째 명령어는 실행되지 않는 조건부 실행 명령어라는 것을 지시하는 상태 비트가 컴퓨터 시스템에 존재하도록 하고 그 값을 1로 설정한다.

도 10은 도 2의 넷째 명령어(230)를 예로 사용하여, 조건 레지스터를 갱신하는 명령어의 커미트 과정을 보여준다.

도 10의 순차적 조건 상태 지시기는 조건 레지스터의 순차적 상태를 저장하는 물리적 조건 레지스터의 주소 값을 저장하고 있는 것으로서, 조건 레지스터 갱신 명령어의 커미트 시에 그 값이 변화한다.

도 2를 참조해 보면, 넷째 명령어(230)의 조건 레지스터 갱신에 물리적 조건 레지스터 PCR4를 할당하기 이전의 물리적 조건 레지스터 할당 중에서 최신의 것은 도 2의 첫째 명령어(200)에 의해서 이루어졌으며, 할당된 물리적 조건 레지스터는 PCR3이다. 도 10은 도 2의 넷째 명령어(230)가 커미트될 때의 모습이므로, 도 2의 넷째 명령어(230) 이전의 연속적인 명령어들은 모두 커미트된 상태이다. 따라서 도 2의 첫째 명령어(200)에 대응되는 리오더 버퍼 엔트리는 이미 커미트된 상태이다. 도 2의 첫째 명령어(200)의 커미트는 컴퓨터 시스템의 상태를 변화시키며, 그 변화 중의 하나가 커미트에 의한 조건 레지스터의 순차적 상태 변화이다. 도 2의 첫째 명령어(200)의 조건 레지스터 갱신에 할당된 물리적 조건 레지스터는 PCR3이므로 해당 리오더 버퍼 엔트리의 커미트에 의해서 조건 레지스터의 순차적 상태를 저장하고 있는 물리적 조건 레지스터는 PCR3으로 갱신된다. 이것은 순차적 조건 상태 지시기 값을 물리적 조건 레지스터 PCR3의 주소 값(3)으로 갱신함으로써 이루어지며, 그 값은 도 2의 넷째 명령어(230)의 커미트에 이해서 PCR4의 주소 값(4)으로 갱신된다.

도 10의 커미트에 의해서 물리적 조건 레지스터 PCR3은 더 이상 조건 레지스터의 순차적 상태를 저장하고 있지도 않으며, 이후의 명령어에 의해서 참조될 일도 없다. 따라서 물리적 조건 레지스터 PCR3은 프리하게 되어야 하며, 이것은 물리적 조건 레지스터 할당 카운터의 값을 1만큼 감소시키는 것에 의해서 이루어진다.

도 10의 커미트에 의한 순차적 상태 지시기와 물리적 레지스터 할당 지시기의 변경은 도 9에서 이루어진 것과 같은 원리로 이루어지며, 그 과정이 도 10에 나와 있다.

도 11은 도 2의 다섯째 명령어(240)를 예로 사용해서, 비실행 예측된(실행 예측 비트가 0인) 조건부 실행 명령어의 커미트 과정을 보여준다.

도 11에서 커미트되는 명령어는 비실행 예측된 명령어이므로, 레지스터 리네이밍도 이루어지지 않았고 기능 유닛에서의 명령어 수행도 이루어지지 않았다. 그러나 실행 결과 비트가 1이므로 실제로는 제대로 명령어 실행이 이루어졌어야 한다. 이러한 실행 예측 오류에 의해서, 도 11의 명령어 수행도 틀려졌고, 이후의 명령어들에 있어서의 레지스터 리네이밍도 틀려지게 된다. 따라서 컴퓨터 시스템을 복구하고, 커미트되는 명령어부터 다시 페치하여 수행하도록 해야 한다.

도 11에서의 명령어 커미트에 의한 컴퓨터 시스템의 복구는 상기 기술한 실행 예측이 틀린 실행 예측된 명령어의 커미트에서의 복구와 같다. 다만 복구 후 페치되는 첫째 명령어는 실행되는 조건부 실행 명령어라는 것을 지시하는 상태 비트가 컴퓨터 시스템에 존재하도록 하고 그 값을 1로 해야 한다는 것이 상기 복구 과정과의 차이이다.

도 11에서 보여 주듯이 도 11의 명령어의 실행 결과 비트가 1이므로, 해당 조건 예측 버퍼 엔트리의 카운터 값은 1만큼 증가한다. 그리고, 복구 후 명령어 페치가 도 11의 커미트되는 명령어부터 이루어지도록 프로그램 카운터의 값을 해당 명령어 주소인 PC5로 변경한다.

도 11은 실행 예측이 틀린 비실행 예측된(실행 예측 비트가 0이고, 실행 결과 비트는 1인) 조건부 실행 명령어의 커미트에 관한 것이며, 실행 예측이 맞은 비실행 예측된(실행 예측 비트와 실행 결과 비트가 0인) 조건부 실행 명령어의 커미트는 이와 다르다. 즉, 실행 예측이 맞았기 때문에, 해당 명령어의 커미트로 인한 컴퓨터 시스템의 복구 작업은 필요가 없으며, 컴퓨터 시스템은 정상적으로 계속 동작을 수행해 나가면 된다. 다만 실행되지 않는 조건부 실행 명령어에 의해서는 컴퓨터 시스템의 상태가 변하지 않으므로, 해당 명령어의 커미트로 인한 상태 변화는 존재하지 않는다. 즉, 실행 예측이 맞은 비실행 예측된 명령어의 커미트는, 단지 해당 리오더 버퍼 엔트리가 리오더 버퍼로부터 없어지는 것(최하위 포인터가 1만큼 증가하는 것)에 의해서 이루어진다. 물론, 해당 조건 예측 버퍼 엔트리의 변경은 이 경우에도 이루어진다.

도 12는 비순차적 시스템에서 물리적 레지스터 할당 지시기, 리네임 상태 지시기, 순차적 상태 지시기, 물리적 조건 레지스터 할당 지시기, 물리적 조건 레지스터 할당 카운터, 리네임 조건 상태 지시기, 순차적 조건 상태 지시기, 조건 예측 버퍼, 리오더 버퍼 및 명령어 이슈 큐 등을 이용하여, 레지스터 리네이밍 및 순차적 상태의 추적을 관리하는 전체 시스템의 개관을 보여준다.

명령어 디코드 유닛에 의해 디코드된 명령어는 도 12의 오피 코드, 한 개의 논리적 목적 레지스터 주소(RX) 및 두 개의 논리적 소스 레지스터 주소들(RY와 RZ)로 변환된다. 그리고, 도 12의 조건 갱신 비트(U), 조건부 비트(C) 및 수행 예측 비트(P)도 명령어의 디코드 단계에서 얻어진다.

도 12의 물리적 레지스터 할당 지시기와 우선권 인코더는 논리적 목적 레지스터(LDRA=RX)에 대해서 물리적 목적 레지스터(PDRA=PRX)를 할당하고, 물리적 목적 레지스터(PRX)에 대응되는 물리적 레지스터 할당 지시기 엔트리의 프리 비트(F) 값을 0으로 만들어, 할당된 물리적 레지스터가 더 이상 프리하지 않다는 것을 표시한다. 또한, 논리적 목적 레지스터(RX)에 대응하는 리네임 상태 지시기 엔트리의 값을 할당된 물리적 레지스터(PRX)의 주소 값으로 변경하여, 논리적 목적 레지스터(RX)에 대한 최신의 리네임 상태를 변경한다. 그리고, 도 12의 리네임 상태 지시기는 첫째 논리적 소스 레지스터(LSRA1=RY)를 첫째 물리적 소스 레지스터(PSRA1=PRY)로 리네임하고, 둘째 논리적 소스 레지스터(LSRA2=RZ)를 둘째 물리적 소스 레지스터(PSRA2=PRZ)로 리네임한다.

도 12의 물리적 조건 레지스터 할당 지시기는 물리적 목적 조건 레지스터(PDCRA=PCRX)를 조건 레지스터의 갱신에 할당하고, 물리적 조건 레지스터 할당 카운터의 값을 1만큼 증가시켜 할당된 물리적 조건 레지스터의 수가 1개 증가했음을 알려준다. 또한 리네임 조건 상태 지시기의 값을 할당된 물리적 조건 레지스터의 주소 값(PCRX)으로 변경하여, 조건 레지스터에 대한 최신의 리네임 상태를 변경한다. 그리고, 도 12의 리네임 조건 상태 지시기는 조건 레지스터의 읽기를 물리적 소스 조건 레지스터(PSCRA=PCRZ)의 읽기로 리네임한다.

이렇게 만들어진 물리적 목적 레지스터 주소(PRX), 물리적 소스 레지스터 주소들(PRY와 PRZ), 물리적 목적 조건 레지스터 주소(PCRX) 및 물리적 소스 조건 레지스터 주소(PCRZ)는 도 12의 명령어 이슈 큐로 전달되어 최상위 포인터가 가리키는 엔트리에 저장된다. 그리고, 논리적 목적 레지스터 주소(RX), 물리적 목적 레지스터 주소(PRX) 및 물리적 목적 조건 레지스터 주소(PCRX)를 도 12의 리오더 버퍼의 최상위 포인터가 가리키는 엔트리에 저장하고, 이 엔트리의 포인터 값인 최상위 포인터 값을 명령어 이슈 큐에 전달하여 명령어 이슈 큐의 최상위 포인터가 가리키는 엔트리의 리오더 버퍼 포인터(RBP)로서 저장한다. 이러한 동작과 함께 명령어 이슈 큐의 최상위 포인터가 가리키는 엔트리의 유효 비트를 1로 세팅하여, 그 엔트리가 유효한 명령어를 가지고 있는 것으로 한다. 그 외에, 몇 가지 부가적인 정보들(명령어 주소(PC), 조건 갱신 비트(U), 조건부 비트(C) 및 실행 예측 비트(P))이 명령어 이슈 큐 또는 리오더 버퍼 엔트리에 함께 저장된다. 그리고, 명령어 이슈 큐와 리오더 버퍼 큐의 최상위 포인터들을 각각 1씩 증가시켜 다음 명령어에 해당하는 내용들을 받아들일 엔트리들을 가리키도록 한다.

도 12의 명령어 이슈 큐에서 기능 유닛이 사용 가능하고, 소스 데이터들 및 소스 조건 데이터가 준비된 명령어는 도 12의 기능 유닛으로 이슈된다. 명령어가 이슈되면서, 실행 예측 비트(P), 명령어 이슈 큐의 오피 코드, 물리적 목적 레지스터 주소(PDRA), 물리적 목적 조건 레지스터 주소(PDCRA) 및 리오더 버퍼 포인터(RBP)를 기능 유닛에 전달하고, 레지스터 파일과 조건 레지스터 어레이를 엑세스하여 2개의 소스 데이터와 1개의 소스 조건 데이터를 기능 유닛에 제공한다.

도 12의 기능 유닛이 명령어 수행을 완료하면 물리적 목적 레지스터 주소(PDRA)와 물리적 목적 조건 레지스터 주소(PDCRA)를 이용하여 명령어 수행 결과를 레지스터 파일과 조건 레지스터 어레이에 저장한다. 그리고, 리오더 버퍼 포인터 값을 리오더 버퍼에 전달하여 해당 명령어의 수행이 완료했음을 알려주면서, 실행 결과 비트(R)를 전달하여 해당 리오더 버퍼 엔트리에 저장한다.

도 12에서, 리오더 버퍼의 최하위 포인터가 가리키는 엔트리의 완료 비트가 1이면, 즉 해당 명령어의 수행이 완료되었으면, 최하위 포인터가 가리키는 엔트리는 커미트된다. 커미트에 의해서, 최하위 포인터가 가리키는 엔트리의 논리적 목적 레지스터 주소 값(LDRA=RJ)에 해당하는 순차적 상태 지시기 엔트리의 값은 최하위 포인터가 가리키는 엔트리의 물리적 목적 레지스터 주소 값(PDRA=PRJ)으로 바뀌며, 해당 순차적 상태 지시기 엔트리에 저장되어 있던 물리적 레지스터 주소 값에 대응되는 물리적 레지스터 할당 지시기 엔트리의 프리 비트(F) 값을 1로 세팅하여, 해당 물리적 레지스터를 프리하게 만들어 차후의 물리적 레지스터 할당에 사용할 수 있도록 한다. 그리고, 커미트되는 엔트리의 물리적 목적 조건 레지스터의 주소 값(PDCRA=PCRJ)으로 순차적 조건 상태 지시기의 값을 변경하고, 물리적 조건 레지스터 할당 카운터의 값을 1만큼 감소시켜 물리적 조건 레지스터가 1개가 프리해졌음을 알린다. 그 외에, 커미트되는 명령어가 조건부 실행 명령어인 경우에는 실행 결과 비트의 값에 따라서 조건 예측 버퍼의 해당 엔트리를 변경시킨다.

컴퓨터 시스템의 초기(reset) 상태에서는 각 논리적 레지스터는 어떤 값을 가지고 있어도 된다. 그리고 모든 물리적 레지스터는 쓰레기 값(garbage value)을 가지고 있다. 이와 같더라도 리네임 상태 지시기, 물리적 레지스터 할당 지시기 및 순차적 상태 지시기가 서로간에 일관된 상태를 유지해야 한다. 이를 위해서 초기 상태에서는 논리적 레지스터 R0부터 RN-1

까지의 순차적 상태의 값이 PR0부터 PRN-1까지의 물리적 레지스터에 저장되어 있으며, 그 외의 물리적 레지스터들은 할당되지 않아 프리하도록 설정한다. 그리고 리네임 상태 지시기는 최신의 리네임 상태가 순차적 상태와 같도록 설정한다. 이와 같이 하기 위해서, 순차적 상태 지시기와 리네임 상태 지시기는 첫째 엔트리부터 N번째 엔트리까지를 첫째 물리적 레지스터 PR0부터 N번째 물리적 레지스터 PRN-1까지의 주소 값으로 초기화 하고, 물리적 레지스터 할당 지시기는 첫째 엔트리부터 N번째 엔트리까지의 값을 0으로 하고 그 외의 엔트리들은 1로 초기화 함으로써, PR0부터 PRN-1까지의 물리적 레지스터들만 프리하지 않게 한다.

컴퓨터 시스템의 초기 상태에서, 일반 레지스터들의 상태뿐 아니라 조건 레지스터의 상태들도 일관된 모습을 유지해야 한다. 이를 위해서 초기 상태에 순차적 조건 상태 지시기와 리네임 조건 상태 지시기의 값을 0(PCR0의 주소 값)으로 하고, 물리적 조건 레지스터 할당 지시기의 값을 1(PCR1의 주소 값)로 하여 초기에는 물리적 조건 레지스터 PCR0가 조건 레지스터의 순차적 상태 값과 최신의 리네임 상태의 값을 저장하고 있는 것으로 설정하고, 물리적 조건 레지스터 할당 카운터를 1로 하여 할당된 물리적 조건 레지스터는 PCR0 1개뿐이라고 설정한다.

본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 본 개시된 실시예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 상기의 설명에 포함된 예들은 본 발명에 대한 이해를 위해 도입된 것이며, 이 예들은 본 발명의 사상과 범위를 한정하지 않는다.

또한 본 발명에 따른 상기의 각 단계는 일반적인 프로그래밍 기법을 이용하여 소프트웨어적으로 또는 하드웨어적으로 다양하게 구현할 수 있다는 것은 이 분야에 통상의 기술을 가진 자라면 용이하게 알 수 있는 것이다.

그리고 본 발명의 일부 단계들은, 또한, 컴퓨터로 읽을 수 있는 기록매체에 컴퓨터가 읽을 수 있는 코드로서 구현하는 것이 가능하다. 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록장치를 포함한다. 컴퓨터가 읽을 수 있는 기록매체의 예로는 ROM, RAM, CD-ROM, CD-RW, 자기 테이프, 플로피디스크, HDD, 광 디스크, 광자기 저장장치 등이 있으며, 또한 캐리어 웨이브(예를 들어 인터넷을 통한 전송)의 형태로 구현되는 것도 포함한다. 또한 컴퓨터가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어, 분산방식으로 컴퓨터가 읽을 수 있는 코드로 저장되고 실행될 수 있다.

발명의 효과

본 발명에 의하면, 레지스터 리네이밍, 물리적 레지스터의 할당과 퇴거 및 순차적 상태 추적을 관리하는 것에 관하여 본 발명의 기반의 되는 방법은 다음과 같이 기존 방법들의 단점들을 제거한다.

본 발명의 기반이 되는 방법은 순차적 상태를 나타내는 레지스터들로 미리보기 상태를 나타내는 레지스터들의 내용을 전송하는 과정을 제거한다. 그리고 단순한 리오더 버퍼 방법의 연합적 찾기 과정을 피하도록 하였으며, 레지스터 리네이밍 과정과 레지스터 읽기를 분리할 수 있도록 하여, 각각은 별도의 파이프라인 단계에서 이루어진다. 이것은 클락 주파수를 높일 수 있도록 한다. 그리고, 보통의 단일 레지스터 어레이가 미리보기 상태의 물리적 레지스터와 순차적 상태의 물리적 레지스터를 구분하지 않고 보유하도록 하여, 레지스터 값을 읽고 쓰는 과정을 순차적 이슈 시스템의 것으로 단순화 한다. 또한 레지스터 어레이 내의 모든 물리적 레지스터는 어떤 논리적 레지스터에도 할당될 수 있도록 하였다. 이와 같이 논리적 레지스터들 간에 모든 물리적 레지스터들을 공유함으로써, 레지스터 파일의 크기를 최소화하고 물리적 레지스터들의 활용도와 시스템의 성능을 높이도록 한다.

본 발명은, 상기와 같은 장점을 지닌 방법을 기반으로 해서, 명령어의 조건부 실행을 특징으로 하는 명령어 세트 구조를 채택한 비순차적 명령어 이슈 컴퓨터 시스템에서 레지스터 리네이밍, 물리적 레지스터의 할당과 퇴거 및 순차적 상태 추적을 효율적으로 관리하는 방법을 제공하는 것이다. 기존의 방법들은 명령어의 조건부 실행을 특징으로 하는 명령어 세트 구조를 채택한 비순차적 컴퓨터 시스템에 적용될 수 없다는 점을 고려하면, 본 발명의 유용성이 명백해진다.

조건부 실행 비순차적 컴퓨터 시스템의 레지스터 리네이밍 및 순차적 상태의 추적을 가능하게 하며, 실행 예측에 대한 높은 정확성이 기반이 될 경우에 시스템의 성능을 크게 향상시킨다.

레지스터 리네이밍은 비순차적 컴퓨터 시스템의 구현을 위해서 필수적인 요소이다. 그러나, 이전의 레지스터 리네이밍 방법들은 조건부 실행을 특징으로 하는 컴퓨터 시스템에 적용할 수 없기 때문에 조건부 실행을 특징으로 하는 컴퓨터 시스템을 비순차적 시스템으로 구성하는 것은 불가능했다. 그러나 본 발명에 의해서 조건부 실행을 특징으로 하는 비순차적 컴퓨터 시스템의 구현이 가능하게 되었으며, 조건 실행 예측의 정확성이 높다면, 컴퓨터 시스템 전체의 성능이 크게 향상될 것이다.

(57) 청구의 범위

청구항 1.

순차적 상태를 나타내는 물리적 레지스터와 미리보기 상태를 나타내는 물리적 레지스터들 양자를 모든 논리적 레지스터들이 공유할 수 있도록 포함하고 있는 레지스터 파일;

논리적 레지스터 수만큼의 엔트리들을 가지고 있으며, 각 엔트리들은 물리적 레지스터의 주소를 값으로 가지며, 상기 레지스터 파일 내의 물리적 레지스터 중에서 어느 것들이 각각의 논리적 레지스터의 순차적 상태를 나타내는지를 지시하는 순차적 상태 지시기;

논리적 레지스터 수만큼의 엔트리들을 가지고 있으며, 각 엔트리들은 물리적 레지스터의 주소를 값으로 가지며, 상기 레지스터 파일 내의 물리적 레지스터 중에서 어느 것들이 각각의 논리적 레지스터의 구조적 상태인 최신의 리네임 인스턴스를 나타내는지를 지시하는 리네임 상태 지시기;

상기 물리적 레지스터 중에서 어느 것들이 논리적 레지스터들에 할당되고, 어느 것들이 프리해서 새로운 물리적 레지스터의 할당에 사용될 수 있는지를 알려주며, 물리적 레지스터 수만큼의 엔트리들을 가지고 있으며, 각 엔트리들은 한 비트로 이루어지며, 각 엔트리는 대응하는 물리적 레지스터가 논리적 레지스터에 할당되어 있는 상태인지 아닌지를 지시하는 물리적 레지스터 할당 지시기;

조건 레지스터의 상이한 리네임 인스턴스들을 나타내는 물리적 조건 레지스터들을 포함하는 조건 레지스터 어레이;

물리적 조건 레지스터의 주소를 값으로 가지는 단일 엔트리로 이루어지며, 상기 조건 레지스터 어레이 중에서 어떤 물리적 조건 레지스터가 논리적 조건 레지스터의 순차적 상태를 저장하고 있는지를 지시하며, 물리적 조건 레지스터의 값이 미리보기 상태에서 순차적 상태로 저장될 때에는 그 값을 나타내며, 조건 레지스터의 순차적 상태를 저장하는 물리적 조건 레지스터의 주소 값을 갱신하면, 기존에 조건 레지스터의 순차적 상태를 저장하고 있던 물리적 조건 레지스터는 더 이상 참조되지 않고 퇴거되도록 하는 순차적 조건 상태 지시기;

물리적 조건 레지스터의 주소를 값으로 가지는 단일 엔트리로 이루어지며, 상기 논리적 조건 레지스터의 구조적 상태인 최신의 리네임 인스턴스를 저장하고 있는 물리적 조건 레지스터를 알려주는 리네임 조건 상태 지시기;

논리적 조건 레지스터 갱신에 할당할 물리적 조건 레지스터의 주소 값을 저장하는 영역 및 현재 할당되어 있어서 프리하지 않은 물리적 조건 레지스터의 수를 저장하는 영역을 포함하는 단일 엔트리를 포함하며, 논리적 조건 레지스터의 갱신에 물리적 조건 레지스터를 할당할 때에는 프리한 물리적 조건 레지스터의 주소 값을 받아서 논리적 조건 레지스터에 할당하며, 이때에 상기 리네임 조건 상태 지시기의 값은 할당된 물리적 조건 레지스터의 주소로 그 값을 갱신함으로써, 조건 레지스터에 대한 최신의 리네임 인스턴스를 가리키도록 하는 물리적 조건 레지스터 할당 지시기;

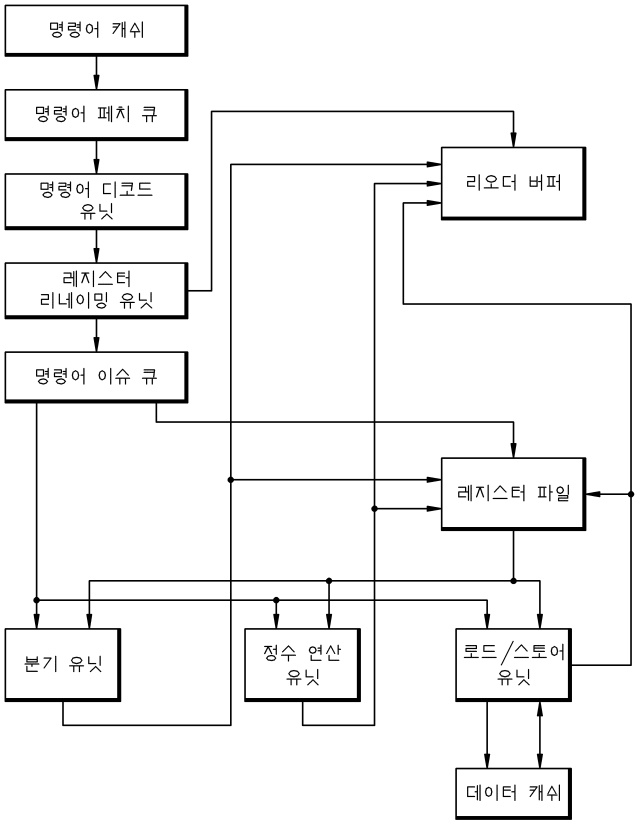
상기 할당된 물리적 조건 레지스터의 수를 지시하는 물리적 조건 레지스터 할당 카운터;

조건부 실행 명령어의 실행 비실행 여부를 예측하기 위한 조건 예측 버퍼; 및

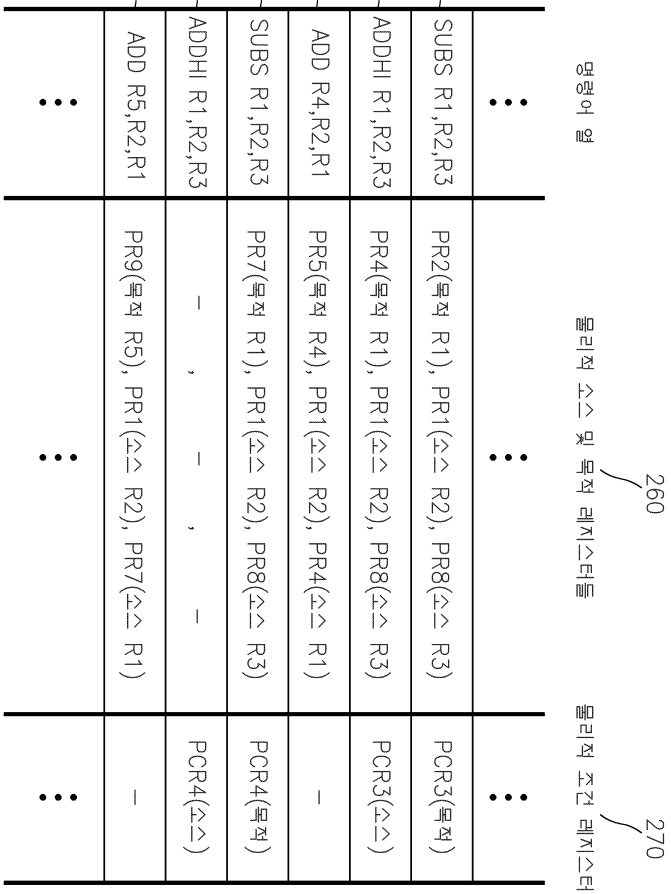
미리보기 상태의 물리적 레지스터 주소 값들을 선입선출 형식으로 보유하고 있으며, 명령어의 레지스터 리네이밍을 수행한 후에 논리적 레지스터 주소 값, 할당받은 물리적 레지스터의 주소 값 및 할당받은 물리적 조건 레지스터의 주소 값을 포함하는 정보가 최상위 엔트리에 기록되며, 해당 엔트리가 최하위 엔트리에 도달했고, 명령어 수행을 완료했을 때에는 그 엔트리는 커미트되며, 커미트는 대응되는 순차적 상태 지시기 엔트리의 물리적 레지스터 주소 값과 순차적 조건 상태 지시기의 물리적 조건 레지스터 주소 값을 갱신함으로써 이루어지는 리오더 버퍼를 포함하는 것을 특징으로 하는 레지스터 리네이밍 방법을 사용하는 조건부 실행 비순차적 명령어 이슈 컴퓨터 시스템.

도면

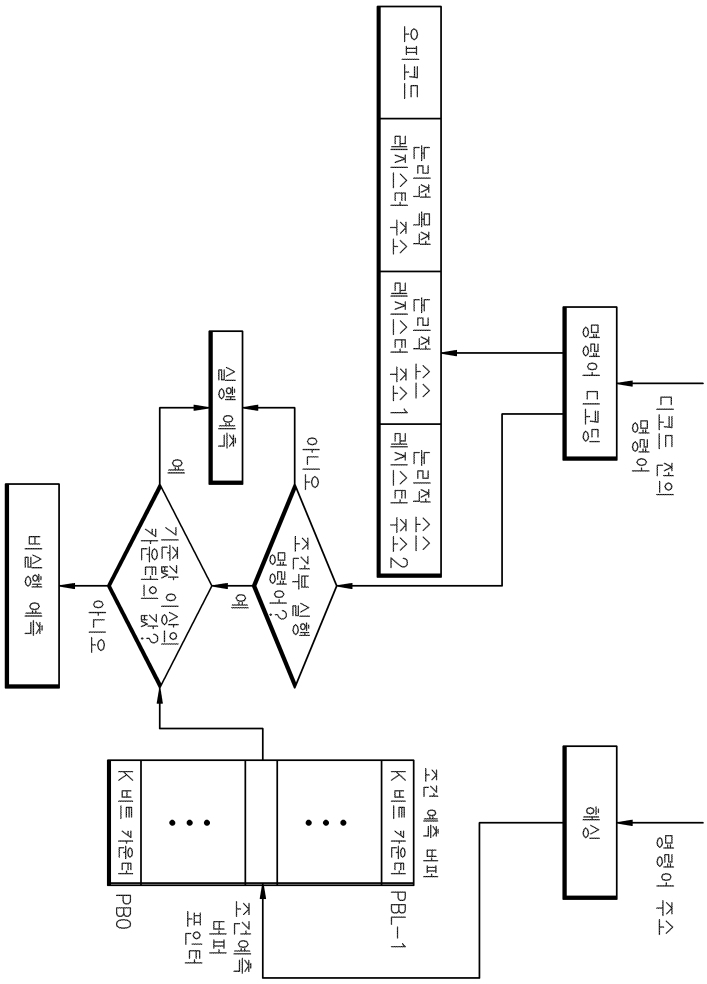
도면1


명령어 캐쉬
명령어 페치 큐
명령어 디코드 유닛
레지스터 리네이밍 유닛
명령어 이슈 큐
리오더 버퍼
레지스터 파일
분기 유닛
정수 연산 유닛
로드/스토어 유닛
데이터 캐쉬

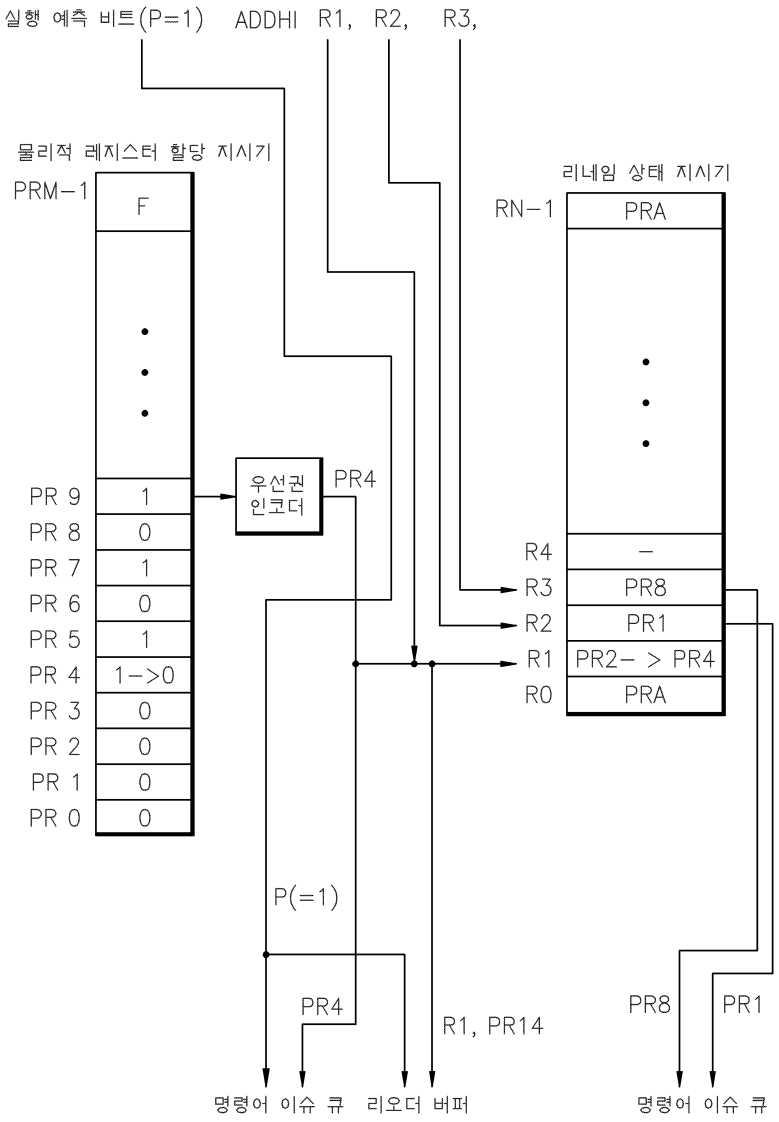
도면2


260
270
명령어 열
물리적 소스 및 목적 레지스터들
물리적 조건 레지스터
200
SUBS R1,R2,R3
PR2(목적 R1), PR1(소스 R2), PR8(소스 R3)
PCR3(목적)
210
ADDHI R1,R2,R3
PR4(목적 R1), PR1(소스 R2), PR8(소스 R3)
PCR3(소스)
220
ADD R4,R2,R1
PR5(목적 R4), PR1(소스 R2), PR4(소스 R1)
–
230
SUBS R1,R2,R3
PR7(목적 R1), PR1(소스 R2), PR8(소스 R3)
PCR4(목적)
240
ADDHI R1,R2,R3
–, –, –
PCR4(소스)
250
ADD R5,R2,R1
PR9(목적 R5), PR1(소스 R2), PR7(소스 R1)
–

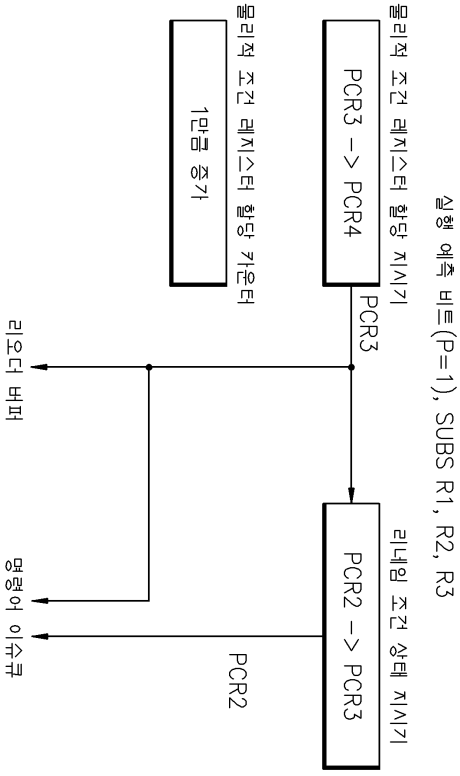
도면3


오피코드
논리적 목적 레지스터 주소
논리적 소스 레지스터 주소 1
논리적 소스 레지스터 주소 2
명령어 디코딩
디코드 전의 명령어
실행 예측
예
아니오
조건부 실행 명령어?
기준값 이상의 카운터의 값?
예
아니오
비실행 예측
K 비트 카운터
조건 예측 버퍼
PB0
PBL-1
조건 예측 버퍼 포인터
해싱
명령어 주소

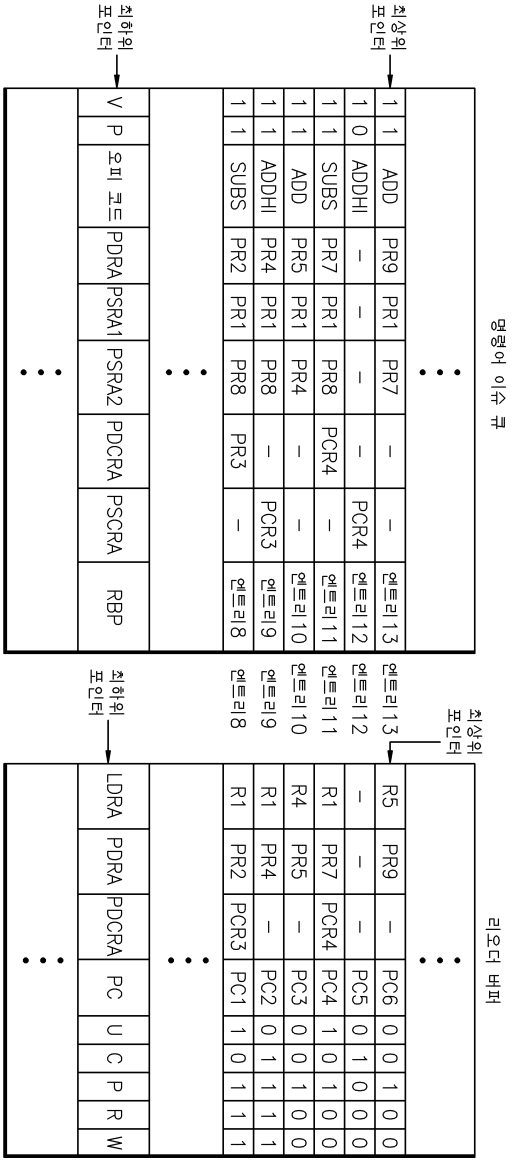
도면4


실행 예측 비트(P=1)
ADDHI R1, R2, R3,
물리적 레지스터 할당 지시기
PRM-1
F
PR 9
1
PR 8
0
PR 7
1
PR 6
0
PR 5
1
PR 4
1->0
PR 3
0
PR 2
0
PR 1
0
PR 0
0
우선권 인코더
PR4
리네임 상태 지시기
RN-1
PRA
R4
-
R3
PR8
R2
PR1
R1
PR2- > PR4
R0
PRA
P(=1)
PR4
R1, PR14
PR8
PR1
명령어 이슈 큐
리오더 버퍼
명령어 이슈 큐

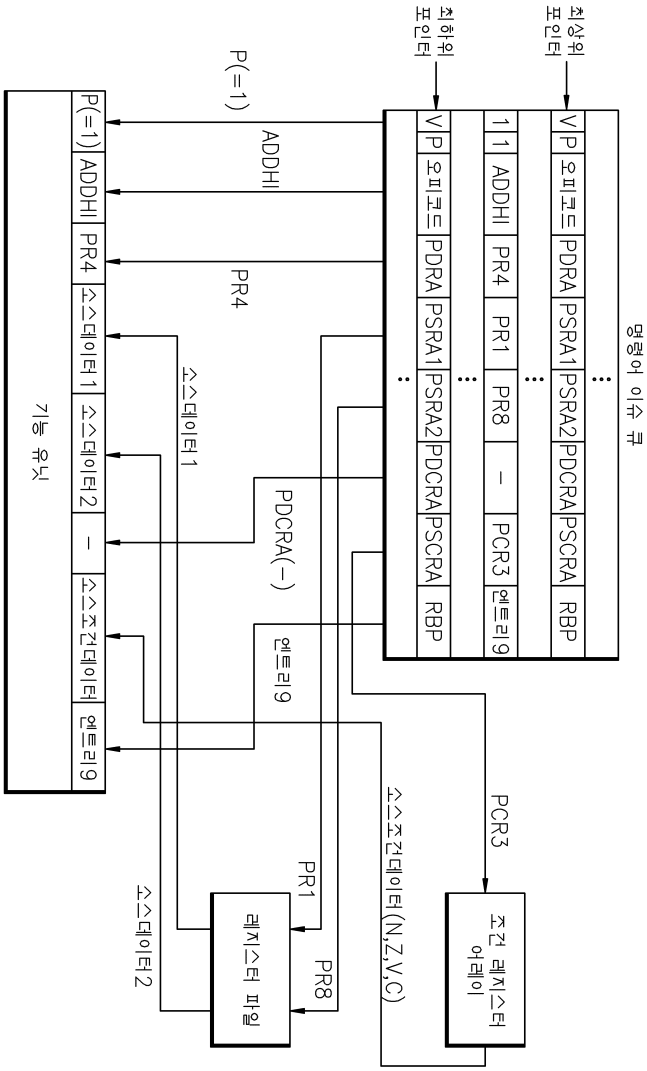
도면5


실행 예측 비트(P=1), SUBS R1, R2, R3
물리적 조건 레지스터 할당 지시기
PCR3 -> PCR4
PCR3
물리적 조건 레지스터 할당 카운터
1만큼 증가
리오더 버퍼
명령어 이슈큐
리네임 조건 상태 지시기
PCR2 -> PCR3
PCR2

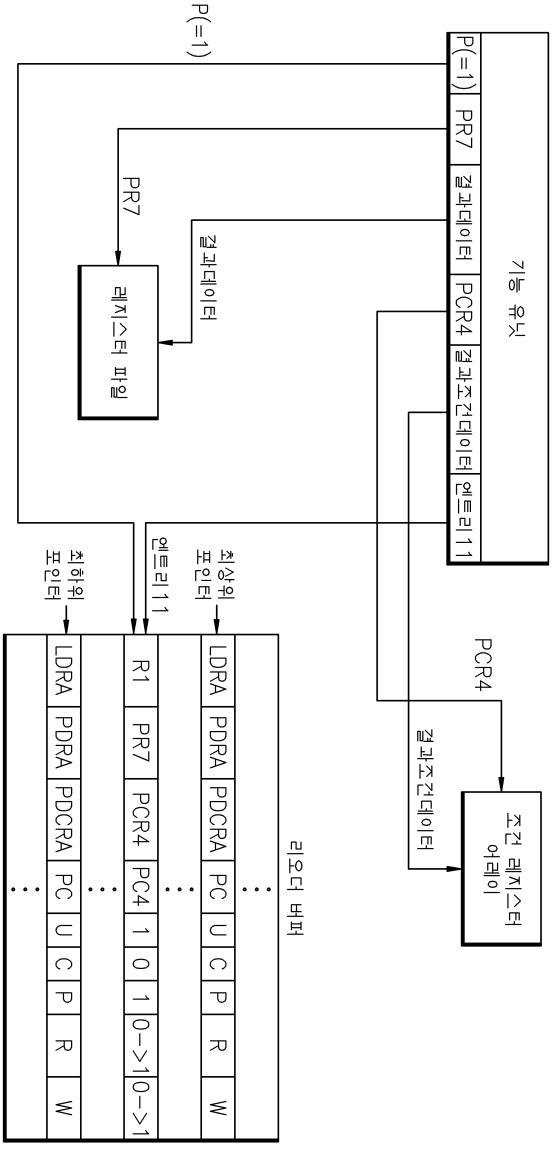
도면6


명령어 이슈 큐
최상위 포인터
최하위 포인터
1 1 ADD PR9 PR1 PR7 — — 엔트리13
1 0 ADDHI — — — — PCR4 엔트리12
1 1 SUBS PR7 PR1 PR8 PCR4 — 엔트리11
1 1 ADD PR5 PR1 PR4 — — 엔트리10
1 1 ADDHI PR4 PR1 PR8 — — 엔트리9
1 1 SUBS PR2 PR1 PR8 PR3 — 엔트리8
V P 오피 코드 PDRA PSRA1 PSRA2 PDCRA PSCRA RBP
리오더 버퍼
최상위 포인터
최하위 포인터
엔트리13
엔트리12
엔트리11
엔트리10
엔트리9
엔트리8
R5 PR9 — PC6 0 0 1 0 0
— — — PC5 0 1 0 0 0
R1 PR7 PCR4 PC4 1 0 1 0 0
R4 PR5 — PC3 0 0 1 0 0
R1 PR4 — PC2 0 1 1 1 1
R1 PR2 PCR3 PC1 1 0 1 1 1
LDRA PDRA PDCRA PC U C P R W

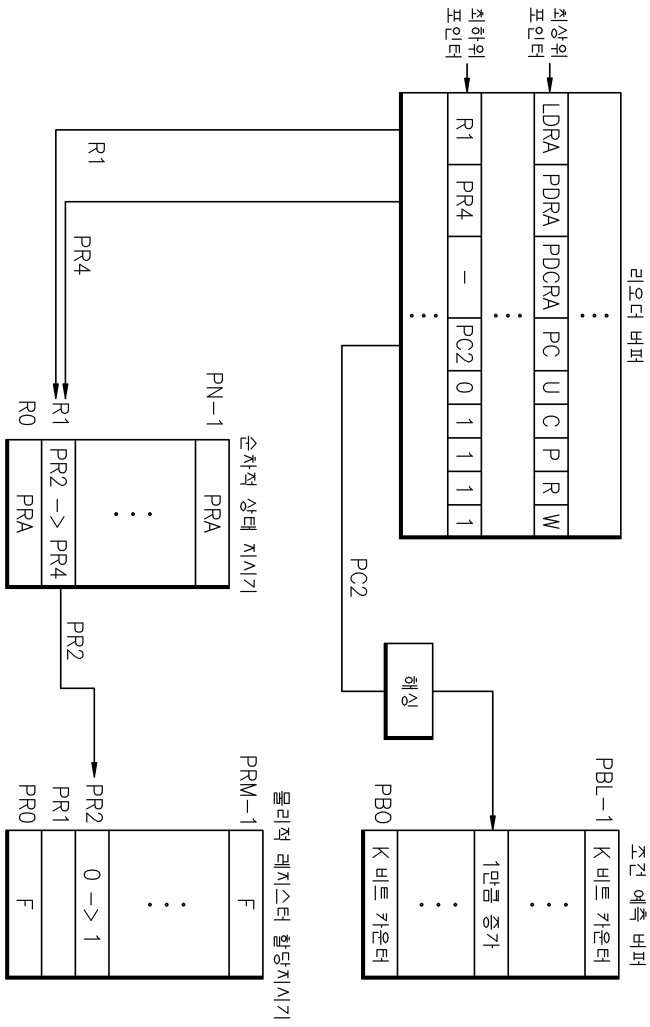
도면7


명령어 이슈 큐
최상위 포인터
최하위 포인터
V P
오피코드
PDRA
PSRA1
PSRA2
PDCRA
PSCRA
RBP
1 1
ADDHI
PR4
PR1
PR8
—
PCR3
엔트리9
P(=1)
ADDHI
PR4
소스데이터1
소스데이터2
—
소스조건데이터
엔트리9
기능 유닛
PDCRA(—)
소스데이터1
PR4
P(=1)
ADDHI
PR1
PR8
소스데이터2
레지스터 파일
PCR3
조건 레지스터 어레이
소스조건데이터(N,Z,V,C)

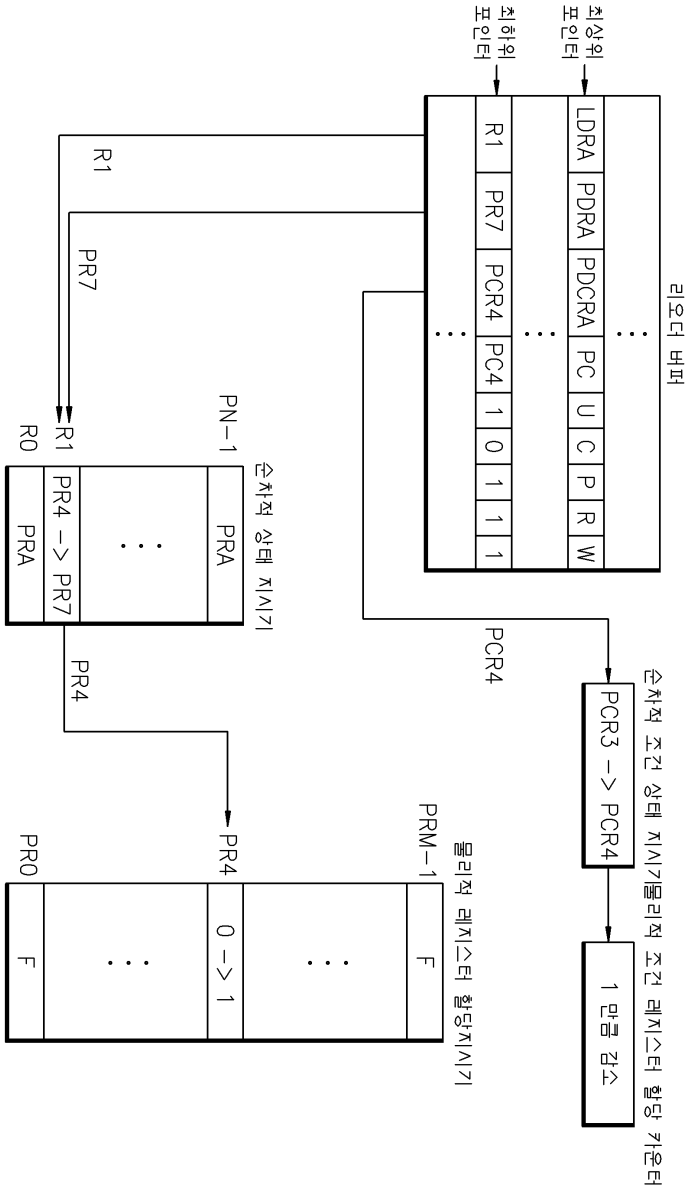
도면8


P(=1)
P(=1)
PR7
결과데이터
PCR4
결과조건데이터
엔트리11
기능 유닛
PR7
결과데이터
레지스터 파일
PCR4
결과조건데이터
조건 레지스터 어레이
최하위 포인터
엔트리11
최상위 포인터
LDRA
PDRA
PDCRA
PC
U
C
P
R
W
R1
PR7
PCR4
PC4
1
0
1
0->1
0->1
리오더 버퍼

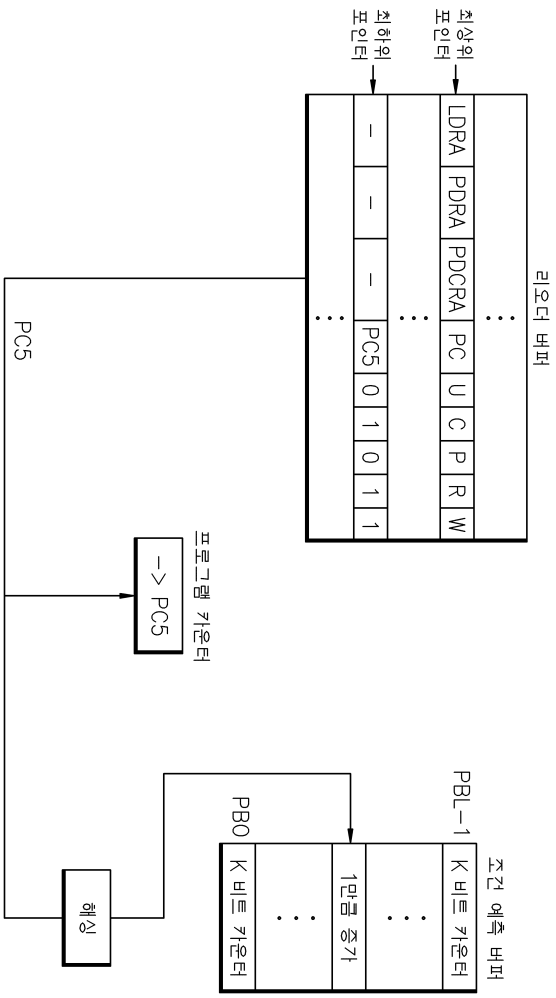
도면9


최상위 포인터
최하위 포인터
LDRA
PDRA
PDCRA
PC
U
C
P
R
W
R1
PR4
–
PC2
0
1
1
1
1
리오더 버퍼
R1
PR4
PC2
해싱
PBL–1
PB0
K 비트 카운터
1만큼 증가
K 비트 카운터
조건 예측 버퍼
PN–1
R0
R1
PRA
PR2 –> PR4
PRA
순차적 상태 지시기
PR2
PRM–1
PR2
PR1
PR0
F
0 –> 1
F
물리적 레지스터 할당지시기

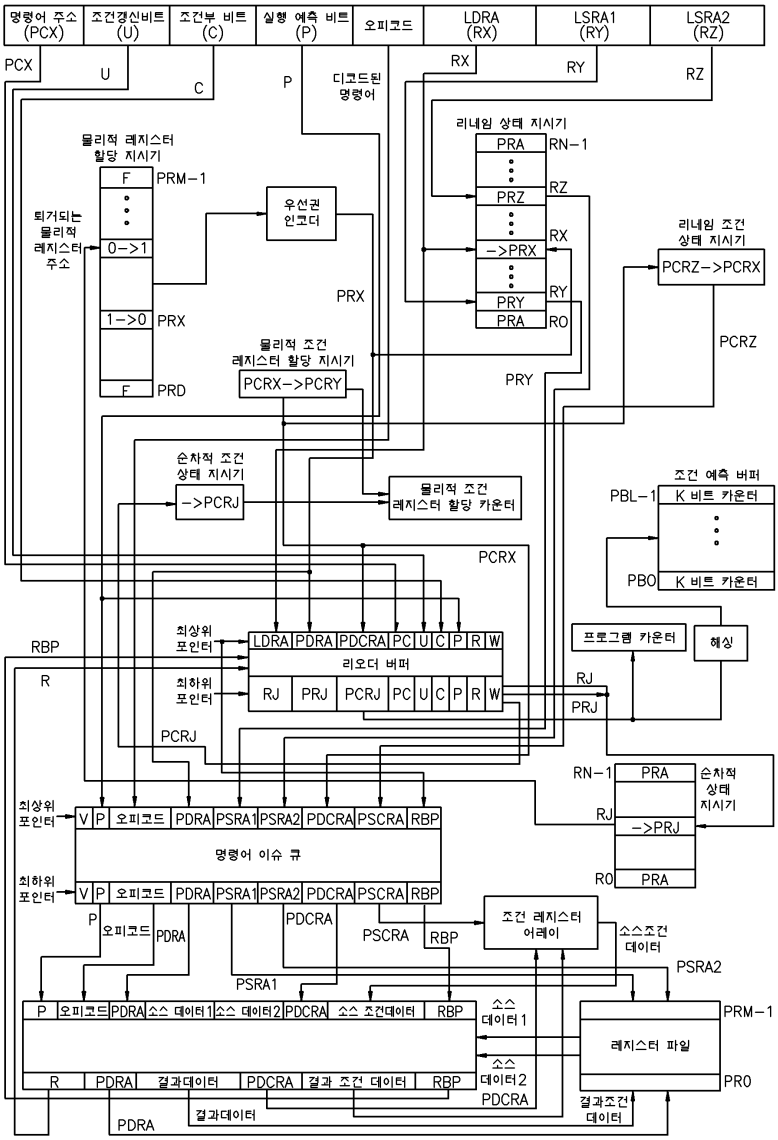
도면10


최상위 포인터
최하위 포인터
LDRA
PDRA
PDCRA
PC
U
C
P
R
W
R1
PR7
PCR4
PC4
1
0
1
1
1
리오더 버퍼
R1
PR7
PCR4
PR4
R1
R0
PR4 -> PR7
PRA
PN-1
PRA
순차적 상태 지시기
PCR3 -> PCR4
순차적 조건 상태 지시기
1 만큼 감소
물리적 조건 레지스터 할당 카운터
PR4
PR0
F
0 -> 1
PRM-1
F
물리적 레지스터 할당지시기

도면11


최상위 포인터
최하위 포인터
LDRA
PDRA
PDCRA
PC
U
C
P
R
W
PC5
0
1
0
1
1
리오더 버퍼
PC5
프로그램 카운터
-> PC5
PBL-1
PB0
K 비트 카운터
1만큼 증가
K 비트 카운터
조건 예측 버퍼
해싱

도면12


명령어 주소 (PCX)
조건경신비트 (U)
조건부 비트 (C)
실행 예측 비트 (P)
오피코드
LDRA (RX)
LSRA1 (RY)
LSRA2 (RZ)
PCX
U
C
P
디코드된 명령어
RX
RY
RZ
물리적 레지스터 할당 지시기
F
PRM-1
퇴거되는 물리적 레지스터 주소
0->1
1->0
PRX
F
PRD
우선권 인코더
PRX
리네임 상태 지시기
PRA
RN-1
PRZ
RZ
->PRX
RX
PRY
RY
PRA
RO
리네임 조건 상태 지시기
PCRZ->PCRX
PCRZ
물리적 조건 레지스터 할당 지시기
PCRX->PCRY
PRY
순차적 조건 상태 지시기
->PCRJ
물리적 조건 레지스터 할당 카운터
조건 예측 버퍼
PBL-1
K 비트 카운터
PBO
K 비트 카운터
PCRX
RBP
R
최상위 포인터
최하위 포인터
LDRA PDRA PDCRA PC U C P R W
리오더 버퍼
RJ PRJ PCRJ PC U C P R W
프로그램 카운터
해싱
RJ
PRJ
PCRJ
RN-1
PRA
순차적 상태 지시기
RJ
->PRJ
R0
PRA
최상위 포인터
V P 오피코드 PDRA PSRA1 PSRA2 PDCRA PSCRA RBP
명령어 이슈 큐
최하위 포인터
V P 오피코드 PDRA PSRA1 PSRA2 PDCRA PSCRA RBP
P
오피코드
PDRA
PDCRA
PSCRA
RBP
조건 레지스터 어레이
소스조건 데이터
PSRA2
PSRA1
P 오피코드 PDRA 소스 데이터1 소스 데이터2 PDCRA 소스 조건데이터 RBP
소스 데이터1
레지스터 파일
PRM-1
PR0
소스 데이터2
PDCRA
R PDRA 결과데이터 PDCRA 결과 조건 데이터 RBP
결과조건 데이터
결과데이터
PDRA