(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) Int. Cl.[6]
G06F 9/38

(45) 공고일자 1999년06월01일
(11) 등록번호 10-0190252
(24) 등록일자 1999년01월19일

(21) 출원번호 10-1991-0010880
(22) 출원일자 1991년06월28일
(65) 공개번호 특1992-0001321
(43) 공개일자 1992년01월30일

(30) 우선권주장 547,629 1990년06월29일 미국(US)
(73) 특허권자 디지탈 이큅먼트 코포레이션 피셔, 아더 더블유.
미국 매사츄세츠주 01754-1499 메이나드 파우더밀 로드 111
(72) 발명자 사이트스, 리챠드 엘.
미합중국, 매사츄세츠 01505, 보일스톤, 워렌 스트리트 21
위테크, 리챠드 티.
미합중국, 매사츄세츠 01460, 리틀톤, 실버 비치 레인 8
(74) 대리인 나영환, 도두형

심사관 : 하유정

(54) 고속 프로세서에서의 브랜치 처리 방법 및 장치

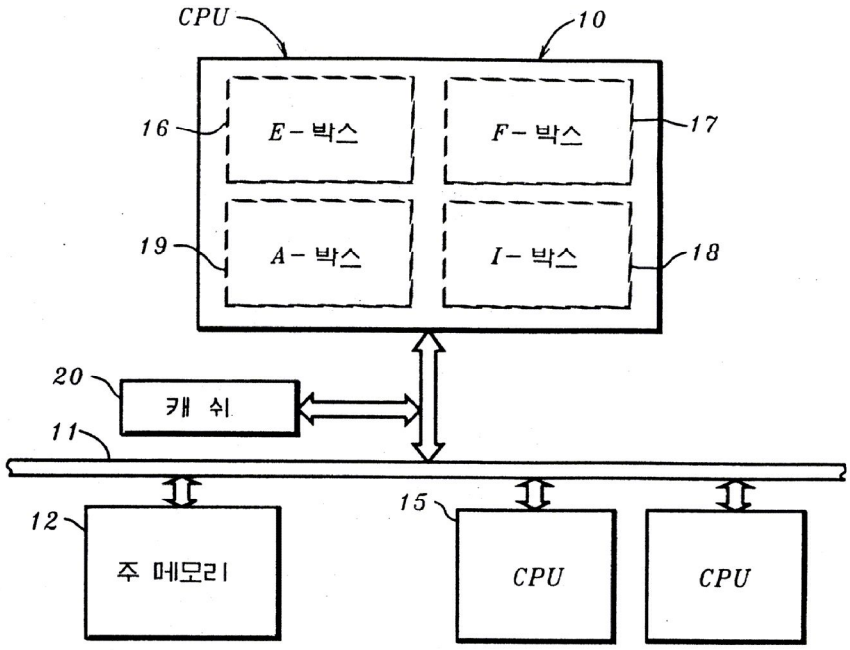
요약

고속 프로세서에서의 브랜치 처리 방법 및 장치

RISC형 (감소된 명령 세트)의 고성능 CPU는 표준화된 일정 명령 크기를 사용하며, 단순화된 메모리 액세스 데이타 폭 및 어드레스 지정 모드만을 제공한다. 이 명령 세트는 레지스터 대 레지스터 동작 및 레지스터의 로드 및 스토어 동작 방식으로 제한된다.

종래에 설정된 데이타 구조의 사용을 가능하게 하는 바이트 조정명령은 비정렬된 로드 및 스토어 동작과 함께, 내부 레지스터에서의 바이트 추출, 삽입 및 마스킹을 행할 수 있는 유용성을 포함한다. 로드/로크 및 스토어 /조건부 명령의 제공은 아토믹 바이트의 기입을 가능케한다. 조건부 이동 명령을 제공함으로써, 많은 짧은 브랜치가 함께 제거될 수 있다. 조건부 이동 명령은 레지스터를 테스트하며 조건이 일치할때 제3레지스터로 제2레지스터를 이동시키는데, 이러한 기능은 짧은 브랜치를 위해 대용될 수 있으므로 명령 스크림의 순차성을 유지할 수 있다. 브랜치의 타켓을 예상하고 이러한 예상에 따라 새로운 명령을 선추출함으로 인해 수행능력을 가속화할 수 있다. 브랜치 예상 규칙은 모든 순방향 브랜치를 받아들이지 않고 모든 역방향 브랜치(통상, 루프)를 받아들이므로써 수행된다. 또 다른 수행능력에 대한 개선점은 점프 및 서브 루틴명령으로 점프등을 위해 예상된 타켓 어드레스의 힌트를 제공하도록 표준 크기의 명령으로된 미사용 비트를 이용할 수 있다는 점이다. 이와같이 타켓은 실제 어드레스가 계산되어 레지스터에 배치되기 전에 선추출될 수 있다. 아울러, 점트 명령의 미사용 변위 부분이 점프, 즉 점프, 서브 루틴으로 점프 및 서브 루틴으로부터 복귀등에 대한 실제 형태를 정의한는 필트를 포함할 수 있으므로 명령이 실행되기 전에 선추출을 할수 있도록 스택에 예상된 타켓 어드레스를 배치할 수 있다. 프로세서는 가변 메모리 페이지 크기를 사용할 수 있기 때문에 가상 어드레스 지정을 수행하기 위한 번역 버퍼에서의 엔트리들이 최적으로 사용될 수 있다. 그래뉼 어리티 힌트는 이 엔트리에 대한 페이지 크기를 한정하도록 페이지 테이블 엔트리에 가산된다. 또 다른 특징은 선추출 명령을 추가할 수 있다는 점인데, 이 명령은 데이타 블록이 사용되기전에 메모리 계층에 있는 고속 액세스 캐쉬로 데이타 블록을 이동시키는 역활을 한다.

대표도



명세서

[발명의 명칭]

고속 프로세서에서의 브랜치 처리 방법 및 장치

[도면의 간단한 설명]

제1도는 본 발명의 특징을 이용할 수 있는 CPU를 채용한 컴퓨터 시스템의 전기 블록도.

제2도는 제1도의 프로세서에 사용되는 데이타 형태에 관한 도면.

제3도는 제1도의 CPU의 명령 유닛 또는 I-박스에 대한 전기 블록도.

제4도는 제1도의 CPU의 정수 실행 유닛 또는 E-박스의 전기 블록도.

제5도는 제1도의 CPU의 어드레스 지정 유닛 또는 A-박스의 전기 블록도.

제6도는 제1도의 CPU의 부동 소수잠 실행 유닛 또는 F-박스의 전기 블록도.

제7도는 제1도 내지 제6도의 CPU에 있어서의 파이프 라이닝에 대한 타이밍도.

제8도는 제1도 내지 제6도의 CPU의 명령세트에 사용되는 명령포맷을 나타내는 도면.

제9도는 제1도 내지 제6도의 PCU에 사용되는 가상 어드레스의 포맷을 나타내는 도면.

제10도는 제1도 내지 제6도의 CPU에 사용되는 페이지 테이블 엔트리의 포맷을 나타내는 도면.

제11도는 제1도 내지 제6도의 CPU 에 사용되는 어드레스 지정 번역기의 도면.

* 도면의 주요부분에 대한 부호의 설명

10 : CPU　　11 : 시스템 버스

12 : 주메모리　　20 : 캐쉬

21 : I-캐쉬　　30 : 브렌치 예상 회로

31 : 서브 루틴 복귀 스택　　32 : 어드레스 발생회로

40 : 산술/논리 유닛　　42 : 정수 승산기

49 : 로드 실로　　50 : 기입 버퍼

52 : 외부 인터페이스　　57 : 헤드 포인터

58 : 테일 포인터　　61 : 부동 소수잠 레지스터 파일

[발명의 상세한 설명]

본 발명은 디지탈 컴퓨터에 관한 것으로서, 보다 구체적으로는 감소된 명령 세트를 실행하는 고성능 프로세서에 관한 것이다.

복잡한 명령세트 또는 CISC 프로세서는 종종 복잡한 메모리 액세싱 모드를 가진 메모리 대 메모리 명령을 포함하여, 그들의 명령 세트에 다수의 명령을 갖는 것을 특징으로한다. 일반적으로 명령은 변화가 가능한 길이로 되어있는데, 예컨대 간단한 명령은 1바이트의 길이만을 갖지만 수십 바이트에 달하는 길이를 갖는 명령도 있다. VAX™ 명령 세트는 CISC의 주요한 예이며, 1 내지 2바이트의 OP 코드와 0 내지 6개의 오퍼랜드 스페시파이어(specifier)를 가진 명령을 사용하는데, 각각의 오퍼랜드 스페시파이어는 길이가 1 바이트 내지 다수의 바이트로 되어 있다. 오퍼랜드 스페시피이어의 크기는 어드레스 모드, 변위 크기(바이트, 워드 또는 롱워드)등에 따른다. 오퍼랜드 스페시파이어의 제1바이트는 그 오퍼랜드의 지정 모드를 나타내며, OP코드는 오퍼랜드의 수, 즉 1, 2 또는 3를 정의한다. OP코드 자체가 디코드되어도 오퍼랜드 스페시파이어가 아직 디코드 되지 않은 경우에는, 프로세서 명령의 전체 길이를 알지 못한다. VAX형 프로세서의 다른 특징은 쾌드(quad)워드 또는 롱워드 참조를 포함하여 바이트 또는 바이트 스트링 메모리 참조를 사용한다는 것인데, 즉 메모리 참조는 비정렬된 바이트 참조를 포함해서, 1바이트로부터 다수 워드로 변할 수 있는 길이로 될수 있다.

감소된 명령 세트 또는 RISC프로세서는 소수의 명령이 간단히 디코드될 수 있고, 모든 산술/논리 연산이 레지스터 대 레지스터 방식으로 수행되는 것을 특징으로 한다. 다른 특징은 복잡한 메모리의 액세스를 제공하지 않는 특징이 있으며, 모든 메모리 액세스는 레지스터의 로드/스토어동작이고,상대적으로 간단한 소수의 어드레싱 모드가 있기는 하지만 단지 오퍼랜드 어드레스를 특정화하는 몇몇 방법만이 제안되고 있다. 명령은 단지 하나의 길이로만 이루어지지만 메모리 액세스는 일반적으로 정렬된 표준 데이타폭으로 이루어진다. 명령 실행은 마이크로 코딩과 구별되는 직접 배선형으로 이루어진다. 일정한 명령 사이클 시간에서 명령이 비교적 간단하게 정의될 수 있기 때문에 하나의 짧은 사이클내에서 모든 명령을 실행할 수 있다.(그 이유는 파이프 라이닝이 수 사이클 동안 실제적인 실행을 전파하기 때문이다).

CISC프로세서의 하나의 특징은 소스 코드를 기입하는데 있다. 강력한 명령의 다양성, 메모리 액세싱 모드 및 데이타 형태로 인해 코드의 각 라인에대해 보다 많은 작업을 수행할 수 있지만(실제로, 컴파일러는 상기 열거한 것을 최대로 이용하는 코드를 발생새키지 않음), 소스 코드를 압축하는데 있어서의 이득은 실행 시간의 희생이 뒤따른다. 특히, 파이프 라인식의 명령 실행은 현재 시스템에서 요구되는 성능 레벨을 달성하는데 필요하므로, 연속적인 명령의 데이타 또는 상태 의존성 , 및 메모리 액세스 시간 대 기계사이클 시간에 있어서의 큰 차이가 과도하게 스톨(stall) 및 예외를 발생시키며 실행을 느리게 한다. RISC프로세서의 장점이 코드 실행 속도에 있는 반면 그 단점은 코드의 각 라인에 의해 양호하게 달성되지 못한다는 점이며, 소정의 임무를 달성하기 위해서는 코드의 길이가 아주 커야 한다. VAX 코드의 하나의 라인은 RISC코드의 다수의 라인과 동일하게 이루어질 수 있다.

CPU가 메모리보다 아주 빠른 경우, 단위 명령당 보다 많은 작업을 한다는 점에서 유리하지만, 그렇지 않은 경우에는 CPU가 항상 명령을 전달하기 위해 메모리를 기다리고 있어야 하기 때문에 결국 서브 루틴으로 구성되는 것을 포함하는 복잡한 명령으로 된다. CPU 및 메모리 속도가 보다 균형을 이루는 경우 RISC 개념과 같은 간단한 방법이 보다 실용화될 수 있는데, 이때에는 메모리 시스템이 각 사이클에서 하나의 명령 및 몇몇 데이타를 전달할 수 있는 것으로 가정한다. 고속 액세스 사이클 뿐만 아니라 계층 메모리 기술은 고속 메모리 속도를 제공한다. CISC대 RISC선택에 영향을 주는 다른 요인은 CPU의 VLSI구성에 따른 오프칩 대 온칩의 상호 접속 비용이 변화될 수 있다는 점이다. 보드 대신에 칩상에 구성하는 것이 경제성을 변화시키는데, 첫째로 이것은 하나의 칩상에 아키텍처를 간단히 만들 수 있게 하므로, 메모리 창조에 대한 오프칩을 제거하도록 다수의 온칩 메모리가 가능하다는 점이다. 비교에 따른 다른 요인은 CISC의 문제점으로써 보다 복잡한 명령 및 어드레스 지정 모드를 추가함으로써 명령 실행공정 단계를 복잡하게 하므로 속도가 저하된다는 점이다. 복잡한 기능은 등가의 연속된 간단한 명령보다 고속으로 기능 실행을 할 수 있지만 명령 사이클 시간이 길어 질 수 있고, 모든 명령 실행을 느리게할 수 있으므로 명령 실행 속도의 감소를 보상할 수 있을 정도로 번반적인 기능을 향상시키도록 추가의 기능이 증가되어야 한다. 여러가지 면을 고려해 볼때, RISC프로세서의 장점은 단점을 능가하는 것으로 생각되며 이것이 기존의 소프트웨어를 이용하는 것이 아닌 경우보다 새로운 프로세서가 RISC특성에 맞게 설계되어야 한다. 문제는 사업을 시작한 이래 오퍼레이터의 교육뿐만아니라 그 자체의 코드 비용을 포함하여 과거 10년 내지 15년 동안 가장 널리 사용되어온 CISC형 프로세서를 사용하는 응용 프로그램 및 데이타 명령에 수년의 작업 경력을 투자했다는 사실이다. 심지어 궁극적으로 달성되리라고 생각되는 장점이 실질적인 면이 있다 하더라도 새로운 프로세서의 아키텍쳐에 맞게 모든 코드 및 데이타 구조를 다시 고치는데 다른 작업 비용 및 번거로움이 반드시 정당화 될수는 없다.

따라서, 본 발명의 목적은 RISC형 프로세서 아키텍쳐의 모든 기능을 유리하게 달성할 수 있고 기존의 CISC형 프로세서에 사용되는 데이타 구조 및 코드가 고성능 프로세서의 사용을 위해 번역될 수 있도록 하는 것이다.

본 발명의 일실시예에 따르면, 고성능 프로세서는 표준형의 일정 명령 크기를 사용하는 것으로서, 간단한 어드레스 지장 모드를 사용하는 간단한 메모리 액세스 데이타 폭 만을 허용하는 RISC형을 제공한다. 명령 세트는 레지스터 대 레지스터 동작(ALV 등을 사용하는 산술 및 논리형 연산) 및 메모리가 창조되는 레지스터 로드/스토어 동작으로 제한되지만, 메모리 대 메모리 동작, 또는 ALV 또는 다른 논리 기능이 행하여지는 레지스터 대 메모리 동작에는 제한되지 않는다. 명령에 의해 수행되는 기능은 비마이크로 코드형 구성을 제공하고 짧은 사이클 동안 간단히 디코드 및 실행하는데 제한된다. 온칩의 플로팅 처리가 제공되며 온-칩 명령 및 데이타 캐쉬가 실시예에 채용된다.

바이트 조정 명령은 사전 설정된 데이타 구조를 사용할 수 있다. 이러한 명령은 비정렬된 로드 및 스토어 명령에 따라 레지스터 바이트를 추출, 삽입 및 마스킹하기에 따른 유용성을 포함하므로, 바이트 어드레스는 실제 메모리 동작이 본래 쾌드 워드로 정렬된 것이라 하더라도 사용할 수 있다.

로드/로크 및 스토어/조건부 명령은 아토믹 바이트 기입 구성을 제공한다. 바이트 어드레스로 다중 바이트(즉, 쾌드 워드)로 정렬된 메모리에 기입하기 위해서, CPU가 쾌드 워드(또는 롱워드)를 로드하고, 이 위치를 로크하며 바이트 어드레스로 레지스터에 기입하는 동시에 나머지 쾌드 워드를 방해받지 않은 상

태로하여 로드/로크 동작 이후에 다른 프로세서에 의해 쾌드 워드가 기입되었는지의 여부에 따라 조건부로 메모리에 갱신된 쾌드 워드를 기입한다.

본 발명의 하나의 특징에 따른 다른 바이트 조정 명령은 바이트 비교 명령이다. 레지스터에서 쾌드 워드의 모든 바이트는 다른 레지스터에서 대응 바이트와 비교된다. 그 결과 제3레지스터에는 단일의 바이트만 남게 된다(비교된 각 바이트에 대한 하나의 비트). 이러한 동작은 특정한 하드웨어 위치와 별개로 범용 레지스터도 행하여지기 때문에 다수의 바이트의 비교가 연속적으로 행하여질 수 있으며, 인터럽트등을 위한 어떤 추가의 상태가 고려되지 않는다. 이러한 바이트의 비교는 바이트 제로 명령으로 유리하게 사용될 수 있는데, 이 명령에서 선택된 쾌드 워드의 바이트가 제로가되며 이들 바이트들은 레지스터의 하위 바이트에서 비트에 의해 선택된다. 즉 바이트 비교의 결과는 다른 레지스터의 제로 바이트에 사용될 수 있다.

실행속도는 명령 스트림의 순차성에 크게 의존하는데, 브랜치는 시퀀스를 방해하여 추출된 명령 스트림이 플러쉬(flush)되어 새로운 시퀀스가 시작되는 동안 스톨(stall)을 발생시킨다. 조건부 이동 명령을 제공함으로써 많은 짧은 브랜치가 함께 제거될 수 있다. 조건부 이동 명령은 레지스터를 테스트하며 조건이 일치하는 경우 제2레지스터를 제3레지스터로 이동시키는데, 이러한 기능은 짧은 브랜치를 위해 대체될 수 있으므로 명령스트림의 순차성을 유지하게끔 한다.

브랜치를 방지할 수 없는 경우, 브랜치의 타켓을 예상하고 이러한 예상에 따라 새로운 명령을 추출함으로 인해 실행을 가속화할 수 있다. 본 발명의 일실시예의 특징에 따르면, 브랜치 예상 규칙은 모두 순방향 브랜치를 받아들이지 않고 모든 역방향 브랜치를 받아들이도록 함으로써 수행된다. 편집시, 코드는 순방향이 아닌 역방향의 가장 최근의 경로를 확실히 하도록 재배열되므로 예상된 경로가 자주 받아 들여지지 않고 적당한 명령이 추출되게 한다.

다른 실행 개선점은 표준 크기의 명령에서 미사용 비트를 이용하여 예상된 타켓 어드레스의 힌트를 서브루틴 명령등에 대한 점프를 위해 제공하는 것이다. 그러므로, 타켓은 실제 어드레스가 계산되기 전에 추출되어 레지스터에 놓일 수 있다. 명령이 실행될때 힌트의 타켓 어드레스가 계산된 어드레스와 일치하고 추출된 어드레스가 이미 파이프 라인에 있으면 고속실행이 가능하다. 힌트는 컴파일러에 의해 점프 명령에 추가된다.

추가로, 점프 명령의 미상용 변위부는 점프의 실제 형태를 정의하는 필드를 포함할 수 있는데, 즉 점프의 실제 형태는, 점프, 서브 루틴으로의 점프 및 서브 루틴으로부터의 복귀등이므로 명령이 실행되기전에 추출을 제공하도록 스택에 예상된 타켓 어드레스를 위치 설정하거나, 또는 힌트에 의해 정의된 동작에 다른 적절한 행위를 취한다. 힌트는 하드웨어에 의해 무시될 수 있으며, 만약 그렇게 되는 경우 코드가 적절히, 즉 천천히 실행을 행한다.

본 발명의 일실시예의 하나의 특징에 따르면, 프로세서가 변화 가능한 메모리 페이지 크기를 사용하기 때문에 가상 어드레스를 지정하는 구성을 위한 번역 버퍼에서의 엔트리가 최적으로 사용될 수 있다. 그래뉼 어리티 힌트는 상기 엔트리에 대한 페이지 크기를 정의하도록 페이지 테이블 엔트리에 추가된다. 다수의 순서 페이지가 동일한 보호 및 액세스 권리를 공유하는 경우 모든 이들 페이지는 동일한 페이지의 테이블 엔트리와 관련되게 되므로 번역 버퍼의 사용이 보다 효과적으로 될 수 있다. 번역 버퍼에 있는 힌트가 증가하는 경우 페이지 테이블을 액세스하는데 따른 고장의 수가 최소화된다.

본 발명의 또 다른 특징은 데이타 블록이 사용되기 전에 메모리층에 있는 고속 액세스 캐쉬로 데이타 블록을 이동시키는 역할을 하는 선추출 명령을 추가하는데 있다. 이러한 선추출 명령은 벡터 프로세서의 기능과 유사한 기능을 수행하는 컴파일러에 의해 삽입되게 되지만 벡터 하드웨어를 필요로하지 않는다. 선추출 명령은 메모리 예외, 또는 보호 또는 액세스 위반을 발생하지 않기 때문에 선추출이 실패한 경우 실행이 늦어진다. 다시 말해서, 명령이 선택적인 경우, 즉 프로세서가 그 명령을 실행하지 못하는 경우 정상적인 코드가 문제를 야기하지 않고 실행을 행한다.

본 발명의 신규한 특징들은 첨부한 특허 청구 범위에 제시되어 있다. 본 발명의 그밖의 특징 및 장점등은 첨부한 도면과 관련한 이하의 특정 실시예를 통해 보다 잘 이해할 수 있을 것이다.

제1도를 참조하면, 본 발명의 특징을 이용할 수 있는 제1실시예에 따른 컴퓨터 시스템은 시스템 버스를 통해 액세스 되는 I/O유닛(13)과 함께 시스템 버스(11)에 의해 주 메모리(12)에 접속되는 CPU(10)를 포함한다. 이 시스템은 독립형 작업부로부터 중간 범위의 다중 프로세서까지 다수의 레벨로 될 수 있는데, 이 경우에 CPU(15)와 같은 다른 CPU들은 시스템 버스(11)를 통해 주 메모리(12)를 액세스 한다.

CPU(10)는 본 발명의 특징이 다중 칩 형태로 구성된 프로세서에 사용될 수 있지만, 단일 칩의 집적 회로 장치이다. 단일 칩내에는 정수 실행 유닛(16, 이하 E-박스라함)과 함께 부동 소숫점 실행 유닛(17, 이하 F-박스라함)이 포함된다. 명령추출 및 디코딩은 명령 유닛(18) 또는 I-박스에서 수행되고, 어드레스 유닛 또는 A-박스(19)는 어드레스 발생, 메모리 관리 기입 버퍼링 및 버스 인터페이스의 기능을 수행한다. 메모리는 제1실시예에서 명령 유닛(18) 및 어드레스 유닛(19)에 온 칩 명령 및 데이타 캐쉬가 포함되는 계층으로 되어 있다. 한편, 큰 제2레벨의 캐쉬(20)에는 오프 칩이 제공되는데, 이 오프 칩은 어드레스 유닛(19)에서 캐쉬 제어기에 의해 제어된다.

CPU(10)는 모든 명령이 일정 크기로 되어 있는 즉 32비트 또는 하나의 롱워드로 되어 있는 이하 기술되는 명령 세트를 사용한다. 사용되는 명령 및 데이타 형태는 제2도에 도시한 바와 같이 바이트, 워드, 롱워드 및 쾌드 워드를 위한 것이다. 본 명세서에서, 바이트는 8비트로 되어 있고, 워드는 16비트 또는 2바이트이며, 롱워드는 32비트 또는 4바이트이며, 쾌드 워드는 64비트 또는 8바이트이다. CPU(10)내의 데이타 경로 및 레지스터는 일반적으로 64비트 또는 쾌드 워드 크기이며, 그리고 메모리(12) 및 캐쉬는 기본 전송 단위로써 쾌드 워드를 사용한다. 기능 수행은, 종래 소프트웨어 단계에서 사용되는 데이타 형태와 호환성이 있도록 특정한 독특한 명령에 의해 바이트 조정을 행하고, 또한 단지 쾌드 워드 또는 롱워드의 로드 및 스토어의 특성을 유지하지만, 단지 쾌드 워드 또는 롱워드의 로스 및 스토어만을 제공함으

로써 향상된다.

제3도를 참조하면, 명령 유닛(18) 또는 I-박스가 상세히 도시되어 있다. 명령 유닛(18)의 주요한 기능은 E-박스(16), A-박스(19) 및 F-박스(17)로 명령을 발생시키는 것이다. 명령 유닛(18)은 8K 바이트의 명령 스트림 데이타를 스토어하는 명령 캐쉬(21)를 포함하며, 이 명령 스트림 데이타의 쾌드 워드(2개의 명령)은 파이프 라인이 진행하는 각 사이클동안 명령 레지스터(22)에 로드된다. 양호한 실시예에서, 명령 유닛(18)은 병령로 디코더(23, 24)에 상기 두 명령을 디코드한 다음 요구된 재원이 검사회로(25)에 의해 2개의 명령으로서 이용 가능한가를 검사한다. 재원이 이용 가능한 경우 이중 발생이 가능하며 양명령은 버스(26, 27)상에 레지스터 어드레스를 인가함으로써 발생될 수 있으며 CPU(10)에 적절한 요소로 마이크로 제어 버스(28, 29)상의 비트를 제어한다. 재원이 단지 제1명령으로서만 이용되는 경우나 또는 명령이 이중으로 발생되지 않는 경우에는 명령 유닛(18)은 디코더(23)로부터 단지 제1명령만을 발생한다. 명령 유닛(18)은, 재원이 디코더(24)로부터의 제2명령으로서 이용되는 경우, 즉 제1명령이 아닌 경우에는, 명령을 발생시키지 않는다. 명령 유닛(18)은 제1명령으로서 재원이 이용될때까지 명령을 발생하지 않는다. 디코더(23)로부터 단지 한쌍의 명령중 제1명령이 발생하는 경우 명령 유닛(18)은 다시 이중 명령 발생하도록 명령 레지스터(22)로 다른 명령을 보내지 않는다. 이중 명령 발생은 메모리 또는 명령 캐쉬(21)로부터 추출되는 정렬된 쾌드 워드 쌍으로만 시도되며 정렬된 쾌드 워드로써 명령 레지스터(22)로 로드된다.

명령 유닛(18)은 레지스터(22)속으로 로드되는 명령 스트림의 명령에따라 브랜치 예상 회로(30)를 포함한다. 서브 루틴 복귀 스택(31)과 함께 예상 회로(30)는 브랜치 어드레스를 예상하고 어드레스 발생 회로(32)로 하여금 필요 이전에 명령 스트림을 선추출하도록 하는데 사용된다. 서브 루틴 복귀 스택(31, 4개의 엔트리를 가짐)은 이하 기술하는 바와 같은 점프, 서브 루틴으로의 점프 및 복귀 명령으로 히트 비트에 의해 제어된다. 가상 PC(프로그램 카운터, 33)는 선택된 순서로 명령 스트림 데이타에 대한 어드레스를 발생하도록 어드레스 발생 회로(32)에 포함된다.

하나의 브랜치 예상 방법은 조건부 브랜치를 예상하도록 브랜치 변위의 단일 비트 값을 사용함으로 회로(30)는 입력(35)에 나타나는 브랜치 명령의 단일 비트 변위에 응답하게 된다. 신호 비트가 네가티브인 경우 이것은 브랜치를 받아들인 것을 예고하고 있으며 어드레스 지정 회로(32)는 추출되는 새로운 어드레스 시퀀스의 제1어드레스를 발생하도록 레지스터 Ra에 변위를 가산한다. 신호가 포지티브인 경우에는 브랜치가 받아들여질 수 없는 것임을 예상하고 있으며, 제공된 명령 스트림이 순차적으로 계속된다.

명령 유닛(18)은 최근에 사용된 명령 스트림 어드레스 번역 및 8K 바이트 페이지에 대한 보호 정보를 캐쉬하도록 8엔트리와 관련된 번역 버퍼(TB, 36)를 포함한다. 64비트 어드레스가 정상적으로 가능하지만, 실제적으로는 43비트 어드레스가 적당하다. 매 사이클마다 43비트의 가상 프로그램 카운터(33)가 명령 스트림(TB, 36)에 제공된다. 가상 PC와 관련된 페이지 테이블 엔트리(PTE)가 TB(36)에 캐쉬되는 경우 가상 PC를 포함하는 페이지용 보호 비트 및 페이지 프레임 번호(PFN)가 어드레스 번역 및 액세스 검사를 완료하도록 명령 유닛(18)에 의해 사용된다. 이와같이 물리적 어드레스가 명령 캐쉬(21)의 어드레스 입력(37)에 인가된다. 또한 캐쉬 미스가 있으면 이러한 명령 스트림의 물리적 어드레스는 어드레스 유닛(19)을 통해 버스(38)에 의해 캐쉬(20) 또는 메모리(12)에 인가된다. 양호한 실시예에서, 명령 스트림(TB, 36)은 힌트의 가능성이 증가하도록 이하 규정되어 있는 임의의 4개의 그래뉼 어리티 힌트 블록 크기를 지지한다.

제4도에는 실행 유닛 또는 E-박스(16)가 상세히 예시되어 있다. 실행 유닛(16)은 산술/논리 유닛(ALU, 40), 배럴 쉬프터(41) 및 정수 승산기(42)를 포함하는 16비트의 정수 실행 데이타 경로를 포함한다. 또, 실행 유닛(16)은 R0 내지 R31을 포함하고 32개의 레지스터, 즉 64비트폭의 레지스터 파일(43)을 포함한다. 여기서 R31은 모두 제로로써 배선 결합된다. 레지스터 파일(43)은 4개의 판독 포트와, 정수 실행 데이타 경로 및 어드레스 유닛(19)에 대한 오퍼랜드의 소스를 제공하도록 하는 2개의 기입 포트를 갖는다.

버스 구조(44)는 명령 유닛(18)으로부터 버스(28, 29) 상에 디코드된 명령의 제어 비트에 의해 특정화되는 ALU(40), 쉬프터(41) 또는 승산기(42)의 선택된 입력에 레지스터 파일(43)의 2개의 판독 포트를 접속하고, 적절한 기능의 출력을 그 결과를 스토어하는 기입 포트중의 하나에 접속한다. 즉, 명령으로부터의 어드레스 필드는 명령을 실행하는데 사용되는 레지스터를 선택하도록 버스(26, 27)에 의해 인가되며, 제어 비트(28, 29)는 ALU등에서의 연산을 정의하며 버스 구조(44)의 내부 구조가 사용될때를 정의한다.

제5도에는 A-박스 또는 어드레스 유닛(19)이 상세히 예시되어 있다. A-박스(19)는 5개 부분의 기능부를 갖는데, 즉 번역 버퍼(48)를 사용하는 어드레스 번역부와, 데이타를 입중계하기 위한 로드 실로(49)와, 기입 데이타를 출중계하기 위한 기입 버퍼(50)와, 데이타 캐쉬에 대한 인터페이스(51)와, 버스(11)에 대한 외부 인터페이스(52)등이다. 어드레스 번역을 위한 데이타 경로는 기입 및 판독을 위한 제2포트 세트 및 PC를 통해 레지스터 파일(43)를 액세싱함으로써 효과적인 어드레스를 발생시키기 위한 변위 가산기(53)와, 어드레스 버스(54)상에 물리적 어드레스를 발생하기 위한 데이타 TB(48)와, 파이프 라이닝에 필요한 먹스(MUX) 및 바이패서 등을 갖는다.

32개의 엔트리와 전적으로 연관된 데이타 번역 버퍼(48)는 8K바이트 페이지를 위해 최근에 사용되는 데이타 스트림 페이지 테이블 엔트리를 캐쉬한다. 각 엔트리는 임의 크기의 4개의 그래뉼 어리티 힌트 블록을 지지하며, 검출기(55)는 이하에 기술되는 그래뉼 어리티 힌트에 응답하여 가상 어드레스 버스(56)로부터 물리적 어드레스 버스(54)로 통과되는 가상 어드레스의 하위 비트수를 변화시킨다.

로드 및 스토어 명령을 위하여, 유효한 43비트의 가상 어드레스가 버스(56)를 통해 TB(48)에 제공된다. 공급된 가상 어드레스의 PTE가 TB(48)에서 캐쉬된 경우 어드레스를 포함하는 페이지를 위한 PFN 및 보호 비트가 어드레스 번역 및 어드레스 검사를 완료하도록 어드레스 유닛(19)에 의해 사용된다.

기입 버퍼(50)는 2가지 목적으로 사용되는데, 그 하나의 목적은 스토어 데이타를 수신하기 위해 고 대역폭(그러나 유한한)의 재원을 제공함으로써 CPU 스톨 사이클수를 최소화 하는 것이다. 이것은 외부 캐쉬(20)가 데이타를 받아들일 수 있는 속도보다 큰 속도일 수 있는 하나의 쾌드워드의 매 CPU사이클의

최대 속도로 CPU(10)가 스토어 데이타를 발생할 필요가 있기 때문에 요구된다. 두번째의 목적은 CPU(10)로부터 외부 캐쉬(20)속으로 데이타가 기입될 수 있는 속도를 최소화할 목적으로 스토어 데이타를 정렬된 32비트의 캐쉬 블록으로 모으려는 것이다. 기입 버퍼(50)는 8개의 엔트리를 갖는다. 기입 버퍼 엔트리는 버퍼가 데이타의 기입을 포함하지 않는 경우에는 유효하지 않지만 기입될 데이타를 포함하고 있는 경우에는 유효하다. 기입 버퍼(50)는 헤드 포인터(57)와 테일 포인터(58)를 포함한다. 헤드 포인터(57)는 가장 긴 시간 기간동안 유효한 기입 버퍼 엔터리를 유효하게 하는 경향이 있으며, 테일 포인터(58)는 다음으로 유효하게 되는 유효한 버퍼 엔트리 슬롯에 속한다. 기입 버퍼(50)가 완전히 채워지거나 완전히 빈 경우 헤드 및 테일 포인터는 동일한 유효(무효)에트리를 가르킨다. 기입 버퍼(50)에 새로운 스토어 명령이 제공될때마다 명령에의해 발생되는 물리적 어드레스는 각각의 유효 기입 버퍼 엔트리에서 어드레스와 비교된다. 어드레스가 유효 기입 버퍼 엔트리에서의 어드레스와 같은 정렬된 32비트의 블록에 있으면 스토어 데이타가 그 엔트리 속에서 병합되고 엔트리의 롱워드 마스크 비트가 갱신된다. 기입 버퍼속에서 어떤 정합 어드레스도 발견되지 않으면 테일 포인터(58)가 지정한 엔트리 속으로 스토어 데이타가 기입되며, 엔트리가 유효하게 되고 테일 포인터(58)가 다음 엔트리로 증가된다.

어드레스 유닛(19)은 데이타 캐쉬(59, D캐쉬)의 채움이 요구될때까지 매 사이클마다 새로운 로드 또는 스토어 명령을 받아들일 수 있는 완전하게 병합된 메모리 창조 파이프 라인을 포함한다. 데이타 캐쉬(59)의 라인들은 단지 로드 미스로 할당되기 때문에 어드레스 유닛(19)은 로드 미스가 발생할때까지 매 사이클마다 새로운 명령을 받아들일 수 있다. 로드 미스가 발생하는 경우 명령 유닛(18)은 레지스터 파일(43)의 로드 포트를 사용하는 모든 명령, 즉 로드, 스토어, 점프 서브 루틴등의 명령의 발생을 중지한다.

각각의 데이타 캐쉬(59)의 결과, 즉 룩업이 이후의 파이프 라인에서 알려지고(단계 S7), 명령이 파이프 단계(S3)에서 발생되기 때문에 데이타 캐쉬(59)를 미스한 로드 명령 이후에 어드레스 유닛(19)의 파이프 라인에 2개의 명령이 존재할 수 있다. 이러한 2개의 명령은 다음과 같이 처리된다. 먼저 데이타 캐쉬(59)를 힌트한 로드를 완료하고, 둘째는 실로(49)에 로드 미스를 놓고 제1로드 미스가 완료된 후에 순서에 따라 재생한다. 셋째로 파이프 라인에 대하여 정상 시간에서 데이타 캐쉬(59)에 스토어 명령을 제공한다. 이들은 로드 미스와 관련하여 순서대로 기입 버퍼(50)에 제공된다.

제6도에 보다 상세히 예시되어 있는 온칩의 파이프 라인형 부동 소숫점 유닛(17) 또는 F-박스는 이하 기술되는 명령 세트에 따라 DEC 및 IEEE 플로팅 명령을 실행할 수 있다. 부동 소숫점 유닛(17)은 32개의 엔트리, 64비트의 부동 소숫점 레지스터 파일(61) 및 부동 소숫점 산술 및 논리 유닛(62)을 포함한다. 제산 및 승산은 제산/승산 회로(63)에서 수행된다. 버스 구조(64)는 디코드된 명령의 제어 비트로 명령 유닛(18)의 버스(28, 29)상에 표시한 바와 같이 레지스터 파일(61)의 2판독 포트를 적절한 기능 회로에 상호 접속한다. 연산을 위해 선택된 레지스터는 명령 디코드로부터의 출력 버스(26, 27)에 의해 정의된다. 부동 소숫점 유닛(17)은 매 사이클마다 명령을 받아들일 수 있으나 부동 소숫점 제산 명령은 단지 매 여러개의 사이클마다 받아들여질 수 있으므로 예외가 있을 수 있다. 모든 플로팅 명령에 대해 1사이클 이상 지연된 것이 예시되어 있다.

예시된 실시예에서, CPU(10)는 8K바이트의 데이타 캐쉬(59)와 8K바이트의 명령 캐쉬(21) 가지며, 캐쉬들의 크기는 이용 가능한 칩 영역에 따른다. 온칩 데이타 캐쉬(59)는 기입을 통해 직접 맵 되는 판독 할당식 물리적 캐쉬이며 32비트(1-헥사 워드) 블록을 갖는다. 시스템은 도시 생략된 무효 버스를 사용함으로써 메모리(12)와 일치하는 데이타 캐쉬(59)를 유지할 수 있다. 데이타 캐쉬(59)는 데이타 어레이(66)에 롱워드 패리티를 가지며 태그 기억부(67)에는 각 태그 엔트리에 대한 패리티 비트가 있다.

명령 캐쉬(21)는 8K바이트 또는 16K바이트일 수 있으며, 예컨대, 다이영역에 따라 크거나 작을 수 있다. 전술한 것은 물리적 어드레싱을 위해 TB(36)를 이용하지만 TB(36)는 가상 캐쉬일 수 있으며 그와 같은 경우에는 메모리(12)와 일치하지 않는다. 캐쉬(21)가 물리적으로 어드레스된 캐쉬인 경우 칩은 메모리와 그의 결합성을 유지할 수 있는 회로를 포함하게 되는데, 즉 기입 버퍼(50)의 엔트리가 버퍼 인터페이스(52)로 보내지는 경우 어드레스가 복재된 명령 캐쉬(21)와 비교되게 되며, 그리고 명령 캐쉬(21)의 대응 블록이 조건부로 무효화되고, 무효 버스는 명령 캐쉬(21)에 접속되게 된다.

CPU(10)에 있는 주데이타 경로 및 레지스터는 모두 64비트의 폭으로 되어 있다. 즉, 정수 레지스터(43)의 각각 및 부동 소숫점 레지스터(61)의 각각은 64비트의 레지스터이며, ALU(40)는 2개의 64비트 입력(40a, 40b)과 64비트 출력(40c)을 갖는다. 하나의 버스 이상으로 구성되어 있는 실행 유닛(16)에 있는 버스 구조(44)는 정수 레지스터(43)와 ALU(40)의 입·출력 사이에 오퍼랜드를 전송하기 위한 16비트 폭의 데이타 경로를 갖는다. 명령 디코더(23, 24)는 ALU(41, 62)에 대한 입력으로서 어떤 레지스터의 오퍼랜드가 사용되는가를 선택하고 레지스터(43, 61)중 어느 것이 ALU(또는 다른 기능 유닛) 출력을 위한 용도인가를 선택하도록 정수 레지스터(43) 및/또는 부동 소숫점 레지스터(61)의 어드레싱 회로에 인가되는 레지스터 어드레스 출력(26, 27)을 만들어낸다.

이중 발생 결정은 다음 요구 조건에 따라 회로(25)에 의해 만들어지는데, 여기서는 단지 제1칼럼으로부터의 하나의명령과 제2칼럼으로부터의 하나의 칼럼이 하나의 사이클동안 발생될 수 있다.

칼럼 A	칼럼 B
정수 연산	부동 소숫점 연산
부동 소숫점 로드/스토어	수 로드/스토어
부동 소숫점 브랜치	정수 브랜치
JSR	

즉, CPU(10)는 정수로드 또는 스토어를 가진 정수 브랜치는 예외로 하고 정수 연산명령과 함께 정수 로드 또는 스토어 명령의 이중 발생을 제공할 수 있다. 물론, 회로(25)는 동일 사이클동안 2개의 명령을

발생하기 전에 재원이 이용 가능한가를 검사할 수 있다.

중요한 특징은 제1도 내지 제6도의 CPU(10)의 RISC특징이다. CPU(10)에 의해 실행되는 명령은 일반적으로 동일 크기로 되며, 이 경우에 있어서는 가변 길이의 명령 대신 32비트의 크기로 된다. 명령은 대개 가변 사이클수동안이 아닌 1기계 사이클동안 실행된다(이하 기술된 바와 같이 파이프라인식으로 수행되며, 스톨이 없는 것으로 가정한다). 명령 세트는 단지 레지스터 대 레지스터 산술/논리 형태의 연산 또는 레지스터 대 메모리(또는메모리 대 레지스터)로드/스토어 형태의 연산을 포함하며, 간접방식등과 같은 복잡한 메모리 어드레스 지정모드가 없다. ALU(40)에서 연산을 수행하는 명령은 항상 레지스터 파일(43)로부터 또는 명령 필드로부터 그의 오퍼랜드를 가지며 레지스터 파일(43)에 그 결과를 기입하는데 이들 오퍼랜드는 결코 메모리로부터 얻어지지 않으며 그 결과는 결코 메모리로 기입되지 않는다. 메모리로부터의 로드는 항상 레지스터 파일(43, 61)에 있는 레지스터로 이루어지고 메모리로의 스토어는 항상 레지스터 파일에 있는 레지스터로부터 이루어지도록 한다.

제7도를 참조하면, CPU(10)는 정수 연산 및 메모리 참조 명령을 위한 7개 단계의 파이프라인을 갖는다. 명령 유닛(18)은 명령 캐쉬(21)의 히트/미스를 결정하는 7개 단계의 파이프라인을 갖는다. 제7도는 실행 유닛(16), 명령 유닛(18) 및 어드레스 유닛(19)의 파이프라인을 위한 파이프라인 다이어그램이다. 부동 소숫점 유닛(17)은 실행 유닛(16)의 파이프라인과 병렬인 파이프라인을 한정하지만 본질적으로 실행하기 위한 몇개의 단계를 사용한다. 7개의 단계는 S0 내지 S6을 가르키는데, 여기서 각 단계는 1기계 사이클(클록 사이클) 동안 실행되게 된다. 제1의 4개의 단계 S0, S1, S2 및 S3는 명령 유닛(18)에서 실행되며 마지막 3개의 단계 S4, S5 및 S6은 실행 유닛(16) 또는 어드레스 유닛(19)중 하나 또는 다른 하나에서 실행된다.

모든 박스에서 레지스터 파일(43, 61)에 기입됨이 없이 다음 명령의 오퍼랜드로써 하나의 명령의 결과가 사용되도록 하는 바이패스가 있다.

파이프라인의 제1단계(S0)는 명령 추출 또는 1F단계인데, 그 동안 명령 유닛(18)은 베이스로써 PC(33) 어드레스를 사용하여 명령 캐쉬(21)로부터 2개의 새로운 명령을 추출한다. 제2단계(S1)는 스왑단계인데, 그 동안 2개의 추출된 명령은 그들이 동일 시간에 발생되는가를 알아내도록 회로(25)에 의해 평가된다. 제3단계(S2)는 디코드 단계인데, 그 동안 2개의 명령을 제어신호(28, 29) 및 레지스터 어드레스(26, 27)를 발생하도록 디코더(23, 24)에서 디코드된다. 제4단계(S3)는 연산 명령에 대한 레지스터 파일(43)에 대한 액세스 단계이며, 모든 명령을 위한 발생 검사 결정점 및 명령 발생 단계이다. 제5단계(S4)는 그것이 연산 명령인 경우, 예컨대 ALU(40)에서의 연산 사이클이며, 또한 명령 유닛(18)은 어드레스 발생기(32)에서 새로운 PC(33)를 연산하고, 그것이 메모리참조 명령인 경우 어드레스 유닛(19)이 가산기(53)를 이용하여 유효한 데이타 스트림 어드레스를 계산한다. 제6단계 (S5)는 그것이 연산 명령인 경우 ALU(40)에 연산을 위한 2사이클이며, 데이타 TB(48)는메모리 참조를 위해 단계를 조사한다. 최종 단계(S6)는 레지스터 기입부를 가진 연산 명령을 위한 기입단계이며, 예컨대, 그 기간 동안 ALU(40)의 출력(40c)은 기입포트를 통해 레지스터 파일(43)로 기입되며, 이는 데이타 캐쉬(59) 또는 명령 스트림 또는 데이타 스트림 참조를 위한 명령 캐쉬(21)의 히트/미스 결정성이다.

CPU(10) 파이프라인은 명령 처리를 위한 이들 7개의 단계(S0-S6)를 4개의 정적 및 3개의 동적 실행 단계로 분할한다. 제1의 4개의 단계(S0 내지 S3)는 명령추출, 스왑, 디코드 및 발행 조작으로 구성된다. 이들 단계(S0 내지 S3)는 명령이 다수의 사이클 동안 동일한 파이프라인 단계에서 유효하게 남겨지는 한편 재원을 기다리고 다른 이유를 위해 스토링 한다는 점에서 정적이다. 이들 스톨은 파이프라인 프리즈(freeze)라고 부른다. 파이프라인 프리즈는 제로명령이 발생될 때나 한쌍중 하나의 명령이 발생될 때 그리고 제2명령이 발생하는 단계에서 유지되는 동안 발생될 수 있다.

라이프 라인 프리즈는 유효명령 또는 명령이 발생되도록 제공되거나 진행되지 않음을 의미한다.

모든 발생 조건을 만족시키기 위해서는 명령이 완료쪽으로 파이프라인을 통해 계속적으로 발생되어야 한다. 명령이 S3에서 발생된 후에는 그 명령은 소정의 파이프 단계(S4-S6)에서 유지되지 않는다. 모든 재원 경쟁이 명령이 계속적으로 발생되기전에 해소되도록 하기 위해서는 발생 단계(S3, 회로 25)에서 끝나야 한다. 발생 단계(S3)후 명령을 중단시키는 유일한 수단은 고장조건이다.

고장은 다수의 원인으로부터 발생될 수 있는데, 이들은 2개의 부류로 그룹화될 수 있는데, 즉 인터럽트를 포함하는 예외조건과 비예외조건이다. 이들 2개의 조건간의 근본적인 차이점으로서 예외는 파이프라인이 고장조건의 원인이 되는 명령 이후에 추출되는 모든 명령으로부터 플러쉬되어야 함을 필요로 하고 재지시어드레스에서 명령 추출을 재개한다는 점이다. 비예외고장 조건에 대한 예는 브랜치의 예상미스, 서브루틴호출, 복귀 예상 미스 및 명령 캐쉬(21)미스이다. 데이타 캐쉬(59)미스는 고장 조건을 발생시키지 않고 파이프라인 프리즈의 원인을 야기할 수 있다.

예외의 경우에 있어서, CPU(10)는 먼저 예외 명령후에 발생되는 모든 명령에 대한 고장을 일으킨다. 몇몇 에러 조건의 특성으로 인해 이것은 기입 사이클 만큼 늦게 발생될 수 있다. 다음에, 예외명령의 어드레스는 내부 프로세서 레지스터에서 래치된다. 파이프라인이 완전히 드레인된 경우 프로세서가 PAL코드 디스패치에 의해 주어진 어드레스에서 명령 실행을 개시한다. 파이프라인은 정수 및 소숫점 레지스터 파일(43, 61) 양자에 대한 모든 미제의 기입이 완료될때 드레인되며, 모든 미제의 명령은 모든 명령이 기계적인 검사의 부재시 예외없이 완료되는 것을 보장하는 방식으로 파이프라인에 있는 점을 통과한다.

제8도를 참조하면, 제1도 내지 제7도의 CPU(10)에 의해 실행되는 명령 세트의 여러가지 형태 명령의 포맷이 예시되어 있다. 하나의 형태는 메모리 명령(70)인데, 이는 31:26 비트에서 6비트의 OP코드와, 비트25:21 및 20:16에서 2개의 5비트 레지스터 어드레스 필드 Ra 및 Rb, 그리고 비트 15:0에서 16비트로 신호된 변위를 포함한다. 이러한 명령은 레지스터(43)와 메모리(메모리(12) 또는 캐쉬(59, 20))간에 데이타를 전송하고, 유효한 어드레스를 레지스터 파일의 레지스터에 로드하며, 서브루틴점프를 위해 사용된다. 변위 필드15:0는 바이트 오프세트이며, 이것은 가상 어드레스를 형성하도록 레지스터(Rb)의 크기로 연장되어 가산된다. 가상 어드레스는 메모리 로드/스토어 어드레스 또는 특정한 명령에 따른 결과값

으로서 사용된다.

브랜치 명령 포맷(71)이 제8도에 예시되어 있는데, 이는 비트31:26에서 6비트의 OP코드, 비트25:21에서 5비트의 어드레스 필드, 그리고 비트20:0에서 21비트의 브랜치 변위를 포함한다. 이러한 변위는 롱워드 오프 세트로써 간주되며 좌측으로 2비트 변위되고, 64비트까지 신호 연장되고 타켓가상 어드레스를 형성하도록 PC(33)의 갱신된 크기에 가산된다(오버플로워 무시).

연산 명령(72, 73)은 제8도에 예시한 포맷으로 되어 있는데, 3개의 레지스터 오퍼랜드를 위한 하나의 포맷(72)과 2개의 레지스터 오퍼랜드를 위한 하나의 포맷(73)은 문자(literal)그대로이다. 연산포맷은 정수 레지스터 연산을 수행하고, 레지스터 파일(43)에 2개의 소스 오퍼랜드와 하나의 목적지 오퍼랜드를 제공하는 명령을 위해 사용된다. 소스 오퍼랜드중의 하나는 일정한 문자일 수 있다. 비트-12는 연산명령이 2개의 소스 레지스터 연산 또는 하나의 소스 레지스터 및 문자를 위한 것인지를 정의한다. 비트31:26의 6비트 OP코드에 추가하여 연산포맷은 산술 및 논리 연산을 위한 광범위의 선택을 할 수 있도록 비트11:5에서 7비트 기능필드를 갖는다. 소스 레지스터 Ra는 비트25:21의 각 경우에 특정되어 있으며, 목적 레지스터 Rc는 비트4:0에 특정된다. 비트-12가 제로(0)인 경우 소스 레지스터 Rb가 비트20:16에 정의되며, 비트-12가 1이면 8비트의 일정한 문자가 비트20:13의 명령에 의해 형성된다. 이 문자는 범위 0 내지 255에서 양정수로써 간주되며 제로로부터 64비트로 연장된다.

제8도는 부동 소숫점 레지스터(61)의 연산 명령을 수행하는 명령을 위해 사용되는 부동 소숫점 연산 명령 포맷(74)을 나타내고 있다. 부동 소숫점 연산 명령은 비트15:5에서의 11비트 기능 필드와 함께 종전과 같이 비트31:26에 6비트의 OP코드를 포함한다. 3개의 오퍼랜드 필드가 있는데, 즉 Fa, Fb 및 Fc가 있는데, 그 각각은 정수 또는 명령에 의해 정의되는 부동 소숫점 오퍼랜드를 특정화하고, 단지 레지스터(13)를 Fa, Fb 및 Fc에 의해서만 특정화되지만 이 레지스터는 정수 또는 부동 소숫점값을 포함할 수 있다. literal은 지지되지 않는다. 소숫점 변환은 제8도의 부동 소숫점 연산 포맷(74)의 서브 세트를 사용하며 레지스터 대 레지스터 변환 연산을 수행하고, Fb 오퍼랜드는 소스를 특정화하고, Fa 오퍼랜드는 레지스터-31(모두 제로)이어야 한다.

제8도의 다른 명령 포맷(75)은 연장된 프로세서 기능을 특정화하는데 사용되는 특권형 아키텍처 라이브러리(PAL 또는 PAL코드) 명령을 위한 것이다. 이들 명령에서 6비트의 OP코드는 종전과 같이 비트31:26에서 제공되고 26비트의 PAL코드 기능 필드25:0는 동작을 특정화 한다. PAL 코드 명령을 위한 소스 및 목적 오퍼랜드는 개재의 명령 정의로 특정화되고 고정된 레지스터에 공급된다.

제8도의 명령 포맷에서 6비트의 OP코드 필드31:26는 단지 26 또는 64개의 다른 명령만을 부호화한다. 따라서 명령 세트는 64로 제한되게 된다. 그러나, 명령 포맷(72, 73, 74)에서 기능 필드는 비트31:26에서 동일한 OP코드를 가진 여러가지 명령을 제공한다. 또, 점프 명령에서 핸드비트는 이하 설명하는 바와 같이 JSR, RET와 같은 변동을 제공한다.

제9도를 참조하면, 내부 어드레스 버스(56)상에 어세트(assert)되는 가상 어드레스의 포맷(76)이 예시되어 있다. 이 어드레스는 정상적으로는 64비트의 폭으로 되어 있지만 다음 수년내에 이들의 실제 구성은 아주 작은 어드레스를 사용하게 될 것이다. 예컨대, 43비트의 어드레스는 8-데타 바이트의 어드레스 지정 범위를 제공한다. 이 포맷은 예컨대 사용되는 페이지 크기에 따라 13비트 내지 16비트 크기의 바이트 오프셋(77)을 포함한다. 페이지가 8K바이트이면 페이지 필드(77)내의 바이트는 13비트이고, 16K바이트 페이지에 대해서는 필드(77)는 14비트이고, 32K 바이트 페이지에 대해서는 15비트이고, 64K바이트 페이지는 16비트이다.

도시한 포맷(76)은 구성에 따라 크기가 변할 수 있는 Seg1, Seg2 및 Seg3의 라벨이 붙은 3개의 세그먼트 필드(78, 79. 80)를 포함한다. 세그먼트 Seg1, Seg2 및 Seg3는 예컨대 10 내지 13비트일 수 있다. 각 세그먼트의 크기가 10비트이며, Seg3에 의해 정의된 세그먼트는 1K 페이지이고, Seg2에 대한 세그먼트는 1M 페이지이며, 그리고 Seg1에 대한 세그먼트는 1G 페이지이다. 세그먼트 번호 필드 Seg1, Seg2 및 Seg3는 소정의 구성을 위해 동일 크기로 되어 있다. 세그먼트 번호 필드는 페이지 크기의 함수이다. 소정 레벨에서의 모든 페이지 테이블 엔트리는 1페이지를 초과하지 않기 때문에 페이지 테이블을 액세스하는 페이지 스왑이 최소화된다. PTE에서의 페이지 프레임 번호(PEN)필드는 항상 32비트폭으로 되기 때문에 페이지 크기가 증가하면 가상 및 물리적 어드레스 크기가 증가하게 된다.

물리적 어드레스는 기껏해야 48비트이지만, 프로세서는 상위비트의 몇몇의 수를 구성하지 않기 때문에 물리적 어드레스 공간은 보다 작게 구성할 수 있다. 2개의 최상위 구성의 물리적 어드레스 비트는 캐싱 정책 또는 어드레스 공간에 의존적인 구성형태를 선택한다. 구성이 다르면 다르게 사용될 수 있음은 물론 시스템에 적절하게 이들 비트에 대한 제약을 가한다. 예컨대, 30비트29:0의 물리적 어드레스 공간을 가진 작업부에서 비트29는 메모리와 I/O사이에서 선택될 수 있으며, 비트28는 I/O 공간에서 캐싱을 인에이블 또는 디스에이블 할 수 있고 메모리 공간에서 제로가 되어야 한다.

전형적으로, 다중 프로그래밍 시스템에서, 동시에 물리적 메모리(12, 캐쉬)에서 여러가지 처리가 남게 될 수 있으므로 하나의 처리가 다른 처리 또는 작동 시스템중 어느 하나와 관련되지 않도록 CPU(10)에 의해 메모리보호 및 다중 어드레스 공간이 사용된다. 소프트웨어의 신뢰성을 보다 개선하도록 4개의 계층 액세스 모드가 메모리 액세스 제어를 제공한다. 이들은 적어도 특권을 갖고 있는 커넬, 실행, 슈퍼바이저 및 유리등이다. 보호는 개개의 페이지 레벨에서 특정화되는데, 여기서 각 페이지는 4개의 액세스 모드 각각에 대해 액세스 불가능한 판독전용, 또는 판독/기입일 수 있다. 액세스 가능한 페이지는 단지 데이타 또는 명령 액세스를 갖도록 제한될 수 있다.

페이지 테이블 엔트리 또는 PTE(81)는 번역버퍼(36, 48) 또는 동작 시스템에 의해 메모리(12)에서 설정되는 페이지 테이블에 기억되어 있으며, 이는 제10도에 예시되어 있다. PTE(81)는 쾌드 워드 폭으로 되어 있으며 비트63:32에서 32비트 페이지 프레임 번호 또는 PTE(82)을 포함함과 아울러 보호 특징등과 같은 것을 구성하도록 표A에 제시한 바와 같은 비트15:0를 가진 필드(83)에 특정한 소프트웨어 및 하드웨어 제어정보를 포함한다.

특정한 특징은 2비트6:5에서 그래뉼 어리티 힌트(84)이다. 소트프웨어는 페이지 블록이 큰 단일의 페이지로써 다루어질 수 있도록 번역 버퍼(36, 48)에 힌트를 공급하도록 이들 비트를 비제로 값으로 세트될 수 있다. 블록은 8N 페이지의 정렬된 그룹인데, 여기서 N은 PTE6:5의 값, 즉 하위값 제로(페이지 크기+3N)을 가진 가상어드레스에서 시작하는 1-, 8-, 64-, 또는 512 페이지의 그룹이다. 블록은 가상 또는 물리적으로 정렬되는 물리적으로 인접한 페이지 그룹인데, 블록내에는 PFN의 낮은 3N비트는 식별맵을 설명하고 있고(즉, 페이지 필드내의 비트에 추가됨으로써 물리적 어드레스의 일부분으로 사용됨), 상위(32-3N) PFN비트는 모두 동일하다. 블록내에서, 모든 PTE은 비트15:0에 대해 동일한 값을 갖는데, 즉 동일한 보호, 고장, 그래뉼 어리티, 표A의 유효비트를 갖는다. 하드웨어는 8, 64 또는 512의 개별 TB 엔트리 대신에 단일의 TB 엔트리를 가진 전체 블록을 맵하도록 이러한 히트를 사용할 수 있다.

그래뉼 어리티 힌트는 실제로 동일한 보호, 고장 및 유효비트를 가진 인접한 가상 페이지로 맵되는 프레임 버퍼 또는 비-페이지 풀과 같은 큰 메모리 구조에 적당할 수 있음을 유의해야 한다. 그래뉼 어리티 힌트의 사용예는 디스플레이용의 비디오 프레임의 기억부인데, 여기서 하나의 프레임을 정의하는 데이타 블록은 고분해능의 칼라 디스플레이용의 64, 즉 8KB 페이지를 정류할 수 있으며, 이 프레임을 위해 물리적 어드레스를 맵하는 64페이지 테이블 엔트리 사용을 방지하도록 101 대신 사용될 수 있다. 이것은 예컨대, 스크린상에 수직라인을 그리는 프레임 버퍼를 참조하는 경우 물리적 메모리(12)로부터 TB(48)로의 다량의 PTE의 스왑핑을 방지한다.

제11도를 참조하면, 버스(56)상의 가상 어드레스는 TB(48)에서 PTE를 찾아내는데 사용되며, 찾아내지 못하는 경우에는 Seg1필드(78)가 내부 레지스터(86)에 기억된 베이스 어드레스에서 발견된 제1페이지 테이블(85)로 인덱스하도록 사용된다. 테이블(85)에서 Seg1인텍스에서 발견된 엔트리(87)는 제2페이지 테이블(88)을 위한 베이스 어드레스이며, 이를 위해 Seg2필드(79)가 엔트리(89)에 인덱스하는데 사용된다. 엔트리(89)는 제3페이지 테이블(90)의 베이스에 해당하며, Seg3필드(90)는 PTE(91)에 인덱스 하는데 사용되는데, 이느 버스(54)상에 물리적 어드레스를 발생하도록 가산기(92)에 가상 어드레스로부터 바이트 오프셋(77)과 결합된 물리적 페이지의 프레임수이다.

전술한 바와 같이, 바이트 오프셋(77)의 크기는 그래뉼 어리티 힌트(84)에 따라 변할 수 있다.

제8도의 명령 포맷을 사용하여, 제1도의 CPU가 9가지 형태의 명령을 포함하는 명령 세트를 실행한다. 이들은 정수로드 및 스토어명령, 정수제어 명령, 정수산술, 논리 및 변동 명령, 바이트 조정, 소숫점 로드 및 스토어, 소숫점 제어, 소숫점 산술 및 여러가지 잡다한 것을 포함한다.

정수로드 및 스토어 명령은 제8도의 메모리 포맷(70)을 사용하여 다음과 같은 것을 포함한다.

LDA - 어드레스 로드

LDAH - 어드레스 하이로드(쉬프트 하이)

LDL - 사인 익스텐드 롱워드 로드

LDQ - 쾌드 워드 로드

LDL_L - 사인 익스텐드 롱워드 로드 록크

LDQ_L - 쾌드 워드 로드 록크

LDQ_U - 비정렬 쾌드 워드 로드

STL - 롱워드 스토어

STQ - 쾌드 워드 스토어

STL_C - 조건부 롱워드 스토어

STQ_C - 조건부 쾌드 워드 스토어

STQ_ - 비정렬 쾌드 워드 스토어

이들 각각에 대하여 가상 어드레스는 레지스터 Rb를 사인 익스텐드 16비트 변위에 가산함으로써 계산된다(LDAH에 대한 사인 익스텐드 변위의 65536배).

로드 명령 LDL 및 LDQ에 대하여 소스 오퍼랜드는 계산된 어드레스에서 메모리로부터 추출되며 롱워드인 경우 사인 익스텐드 되어 레지스터 Ra로 기입된다. 데이타가 자연적으로 정열되지 않으면 정열 예외가 발생된다.

스토어 명령 STL 및 STQ에 대하여, 레지스터 Ra의 내용이 계산된 가상 어드레스에서 메모리로 기입된다. 로드 어드레스 명령 LDA 및 LDAH는 로드 명령 LDL 및 LDQ와 유사하지만 어드레스가 계산된 후 연산이 중지되고 16비트로 계산된 가상 어드레스가 레지스터 Ra에 기입된다.

로드 룩크 및 스토어 콘디숀얼 명령(LDL_L, LDQ_L, STL_L 및 STQ_L)은 이하 기술되는 아키텍쳐의 중요한 특징을 제공한다. 특히, 이러한 명령결합은 공유된 메모리 위치의 아토믹 갱신을 제공함으로써 다중 프로세서 또는 파이프 라인식 프로세서에서 데이타의 완전무결을 보증하는 역할을 한다. 이런 형태의 다른 명령에서와 같이 가상 어드레스는 그 명령에 제공된 사인-익스텐드 16비트 변위에 그 명령에 특정된 레지스터 Rb의 내용을 가산함으로써 계산된다. LDL_L 또는 LDQ_L 명령이 고장없이 실행되는 경우 CPU(10)가 제5도의 록크형 물리적 어드레스 레지스터(95)에 버스(54)로부터 물리적인 목적 어드레스를 기록하며 록크 플래그(96)를 세트한다. 록크 플래그(96)는 조건부 스토어 명령이 실행될 때 여전히 세트된 경우 스토어가 발생되어, 즉 오퍼랜드가 물리적 어드레스에서 메모리에 기입되고, 로크 플래그(96)의 값, 즉 1이 Ra에 복귀되고 록크 플래그가 제로(0)로 세트되지만 록크 플래그가 제로인 경우 메모리에 대한 스토

어가 발생하지 않고 Ra에 복귀된 값이 제로가 된다.

CPU(10)에 대한 로크 플래그가 세트되고 다른 CPU(15)가 메모리(12)에서 물리적 어드레스의 로크 범위내에 스토어된 경우 CPU(10)내의 로크 플래그(96)가 클리어 된다. 결국, CPU(10)가 메모리(12)에 대한 모든 기입을 모니터하며, 레지스(95)에 있는 어드레스가 정합된 경우 플래그(96)가 클리어 된다. 로크된 범위는 레지스터(59)에서 로크된 물리적 어드레스를 포함하는 2^N 바이트의 정렬된 블록인데, 이 값 2^N은 CPU의 구성에 따라 변하게 되며 적어도 8바이트이며(최소로크 범위는 정렬된 쾌드 워드)이고, 이 값은 대개 CPU의 페이지 크기이다(최대로크 범위는 하나의 물리적 페이지이다). CPU(10)의 록크 플래그(96)는 CPU가 임의의 예외, 인터럽트 또는 호출 PAL 코드 명령과 조우할때 클리어 된다.

명령순서

LDQ_L

수정

STQ_L

BEQ

브랜치가 실현되지 않는 경우 CPU(10)상에 공유메모리(12)내의 아토믹 판독수정-기입 데이타가 실행되지만, 브랜치가 받아들여지면 스토어가 메모리(12)에서의 위치를 수정하지 않기 때문에 명령순서가 이루어질 때까지 반복되게 된다. 즉, 레지스터 Ra가 제로와 동일하게 브랜치를 받아들이는데, 이것은 조건부 스토어 명령에 의해 Ra로 복귀되는 록크 플래그의 값이 제로(0)임을 의미한다(스토머가 계속되지 않는다). 명령순서가 부록 A에 보다 상세히 예시되어 있다.

2개의 록크로드 명령이 어떤 개입스토어 조건없이 실행된 경우 제2의 1이 록크 플레그(96) 및 레지스터(95)에 제1의 상태를 다시 고쳐쓴다. 2개의 조건부 스토어 명령이 어떤 개입이 없이 로드 록크 명령을 실행하는 경우, 제1스토어가 록크 플래그(96)를 클리어 하기 때문에 제2스토어가 항상 고장나게 된다.

언얼라인먼트 로드 LDQ_U 및 LDL_U는 로드 LDQ 또는 LDL과 같지만 가상어드레스의 하위 3비트가 클리어되므로(로드 언얼라인드 명령은 바이트 어드레스를 위해 사용된다). 정렬된 과드 워드 또는 롱워드가 추출된다.

또, 어떠 정렬된 고장이 알려지지 않지만 바이트 어드레스(비정렬된 어드레스)를 알 수 있으면 그것은 간단한 LDQ 또는 LDL 명령을 위한 것이다. 언얼라이언트 로드 명령은 이하 기술되는 바와 같이 바이트조정을 위해 사용된다. 언얼라인트 스토어 명령 STQ_U은 STQ명령과 유사하지는 않지만, 그것은 가상 어드레스의 하위 3개의 비트를 이동시키며, 언얼라이언트 어드레스로 인한 고장을 알리지 않는다.

명령의 제어형태는 8개의 조건부 브랜치 명령, 비조건부 브랜치, 서브 루틴으로 브랜치, 서브루틴 명령으로 점프, 브랜치 명령 포맷(71) 또는 제8조의 메모리 명령 포맷(70)을 사용하는 모든 것을 포함한다. 이들 제어 명령은 브랜치 명령 포맷과 메모리 명령 포맷을 사용하는데, 브랜치 명령 포맷을 사용하면 다음과 같다.

BEQ - 레지스터가 제로일때 브랜치

BNE - 레지스터가 제로가 아닐때 브랜치

BLT - 레지스터가 제로보다 작을때 브랜치

BLE - 레지스터가 제로와 같거나 작을때 브랜치

BGT - 레지스터가 제로보다 클때 브랜치

BGE - 레지스터가 제로와 같거나 클때 브랜치

BLBC - 레지스터의 하위 비트가 클리어될때 브랜치

BLBS - 레지스터의 하위비트가 세트될때 브랜치

BR - 비조건부 브랜치

BSR - 서브루틴으로 브랜치

다음에 메모리 명령 포맷을 사용하면

JMP - 점프

JSR - 서브루틴으로 점프

RET - 서브루틴으로부터 복귀

JSR-COROUTING -서브루틴 복귀로 점프.

조건부 브랜치 명령에 대하여, 레지스터 Ra가 테스트되고, 그리고 특정한 관계가 진인 경우 PC는 정수 가상 어드레스로 로드되지만, 실행은 다음 순서 명령을 계속한다. 조건부 또는 비조건부 브랜치중 어느 하나의 변위는 사인된 롱워드 오프셋으로서 처리되는데 이는 좌측으로 2비트 변위되어(롱워드 경계를 어드레스 하도록) 64비트까지 사인 연장된 다음 타켓 가상 어드레스를 형성하도록 갱신된 PC에 가산된다. 조건부 또는 비조건부 브랜치 명령은 단지 PC와 관련성이 있는, 즉 ±1M 롱워드의 순방향/역방향 브랜치 거리를 제공하는 12비트의 사인된 변위이다.

비조건부 브랜치 명령 BR 또는 BSR에 대하여, BR 또는 JMP(즉, 갱신된 PC)에 따르는 명령 어드레스는 레지스터 Ra에 기입되고 그후에 타켓 가상 어드레스를 가진 PC를 로딩한다. BR 및 BSR은 동일한 동작을 하는데, 이들은 단지 힌트에서 브랜치 예상 로직과 다르며, BSR은 서브루틴 호출로써 예상되며(브랜치 예상 스택상에 복귀어드레스를 푸시), BR은 브랜치로 예상된다(푸시없음).

점프 및 복귀명령에 대하여, 상기 명령(갱신된 PC)에 따르는 명령 어드레스는 레지스터 Ra에 기입되며 그후 타켓 가상 어드레스를 가진 PC를 로드한다. 새로운 PC는 레지스터 Rb로부터 공급되며, Rb의 2개의 하위비트는 무시된다. Ra 및 Rb는 동일한 레지스터를 특정화 할 수 있으며, 종전값을 사용하는 타켓 계산은 새로운 값의 할당이전에 이루어진다.

모든 4개의 명령, JMP, JSR, RET 및 JSR-COROUTINE은 동일한 동작을 하게 되는데, 이들은 단지 브랜치 예상 로직에 대한 힌트에서 다르다. 명령의 변위필드(변위용으로 사용되지 않음)는 정보를 전달하는데 사용된다. 4개의 다른 OP코드는 disp15:14에 다른 비트 패턴을 세트하며 힌트 오퍼랜드는 disp13:0를 세트한다. 이들 비트는 다음과 같이 사용된다.

15:14	의미	예상 타켓 15:0	예상스택작용
00	JMP	PC+{4*disp13:0}	--
01	JSR	PC+{4*disp13:0}	푸시 PC
10	RET	예상 스택	팝
11	JSR- co	예상 스택	팝, 푸시 PC

이러한 구성은 가능한 동 워드 타켓 어드레스의 하위 16비트 내역을 가능케하며(초기에 유용한 명령 캐쉬(21)에 대한 액세스의 개시를 가능케 하는 비트), 또 복귀로부터 호출을 분리할 수 있게 한다(다른 낮은 주파수 동작으로부터). 이러한 테이블에 따른 명령은 단지 힌트로써 사용될 수 있음을 유의해야 한다. 즉 이러한 비트의 세팅을 정정하는 것은 수행능력을 개선시킬 수 있지만 동작을 올바르게 하는데는 필요치 않다.

이와 같이, CPU로 하여금 높은 성능을 달성하게 하고, 브랜치 예상 모델에 따른 힌트를 이용하기 위해서는 다음과 같은 것이 제공되어야 한다.

(1) 연산된 브랜치(JSR, RET, JMP)를 다수로 구성하기 위해서 레지스터 Rb가 액세스 되기 전에 예상된 타켓 명령 캐쉬(21) 어드레스의 양호한 추측을 형성하는데는 실질적인 성능 이득이 필요하다.

(2) CPU에 제1명령 캐시(21)가 페이지(8-64KB)보다 작게 구성될 수 있다. 올바른 예상 서브루틴 복귀경로는 양호한 성능을 위해 중요하므로, 경우에 따라서 CPU가 예상되는 서브루틴 복귀명령 캐시(21) 어드레스의 작은 스택을 포함할 수도 있다.

결국, CPU(10)는 3종류의 브랜치 예상힌트, 즉 타켓 어드레스, 복귀 어드레스 스택 작용 및 받아들여지는 조건부 브랜치를 제공한다.

계산된 브랜치(JSR/RET/JMP)에 대하여, 그 밖의 미사용 변위비트는 가장 가능성 있는 타켓 어드레스의 하위 16비트를 특정화하는데 사용된다. 이들 비트를 사용한 PC와 관련된 계산은 정확히 조건부 브랜치에 사용되는 PC와 관련된 계산일 수 있다. 하위 16비트는 가장 큰 가능한 페이지 내에서 명령 캐시(21) 블록을 특정화하는데 충분하므로 가장 가능성 있는 타켓에 대해 초기 명령 캐시(21)의 액세스를 브랜치 예상로직이 충분히 개시할 수 있는 것으로 예상된다.

모든 브랜치에 대하여, 힌트 또는 OP코드 비트는 간단한 브랜치, 서브 루틴 호출, 서브 루틴 복귀 및 상호 루틴 링크를 구별하는데 사용된다. 이러한 구별로 인해 브랜치 예상 로직이 예상된 복귀 어드레스의 정확한 스택을 유지할 수 있다.

조건부 브랜치에 대하여, 타켓 변위의 사인은 브랜치 예상로직에 의해 실현/비현실 힌트로써 사용된다. 순방향 조건부 브랜치(포지티브변위)는 실현되지 않는 것으로 예상되며, 역방향 조건부 브랜치(네가티브변위)는 실현되는 것으로 예상된다. 조건부 브랜치는 예상복귀 어드레스 스택에 영향을 끼치지 않는다.

정수 산술 명령은 가산, 감산, 승산 및 레지스터(43)의 정수에 대한 사인 및 비사인 비교연산을 수행한다. 레지스터(43)의 정수는 정수 레지스터(43)로 그 결과를 복귀시킨다. 이들 명령은 제8도의 정수 연산 포맷(3개의 레지스터, 2개의 레지스터 및 문자)중 어느 하나를 사용하여 다음과 같은 것을 포함한다.

ADDL - 롱워드 가산

ADDQ - 콰드 워드 가산

CMPEQ - 사인된 콰드 워드가 동일한가를 비교

CMPLT - 사인된 콰드 워드가 보다 작은가를 비교

CMPLE - 사인된 콰드 워드가 동일하거나 작은가를 비교

CMPULT - 비사인된 콰드 워드가 작은가를 비교

CMPULE - 비사인된 콰드 워드가 동일하거나 작은가를 비교

MULL - 롱워드 승산

MULQ - 콰드 워드 승산

UMULH - 비사인된 콰드 워드 승산 하이

SUBL - 롱워드 감산

SUBL - 콰드 워드 감산

ADDL 명령에 대하여, 레지스터 Ra가 레지스터 Rb 또는 문자에 가산되고 사인 연장된 32비트의 합이 레지스터 Rc에 기입되며, Ra 및 Rb의 상위 32비트가 무시된다. ADDQ 명령에 대하여, 레지스터 Ra가 레지스터 Rb 또는 문자에 가산되고 16비트 합이 Rc에 기입된다.

비사인된 비교명령은 캐리를 테스트하는데 사용될 수 있으며, ADD를 사용한 2개의 값을 가산하고 비사인된 합이 하나의 입력보다 작은 경우 캐리에는 최상위 비트가 없게 된다.

비교명령에 대하여, 래지스터 Ra는 레지스터 Rb 또는 문자와 비교되며, 특정한 관계가 사실일때 값 1이 레지스터 Rc에 기입되며, 이와 반대로 값이 제로(0)이면 레지스터 Rc에 기입된다.

증산 명령은 레지스터 Rb 또는 문자의 내용만큼 레지스터 Ra를 승산시키며, 그적이 레지스터 Rc에 기입된다. MULL에 대하여, 그적은 32비트의 사인 연장된 값이며, MULQ은 64비트적이 된다. 비사인된 과드 워드 승산 하이명령 UMULH에 대하여, 레지스터 Ra 및 또는 문자는 128비트 결과값을 만들어 내도록 비사인된 수로써 승산되고 상위 64비트는 레지스터 Rc에 기입된다.

감산명령에 대하여, 레지스터 Rb 또는 문자는 레지스터 Ra에서 감산되며 2차는 목적 레지스터 Rc에 기입된다. 그 차이는 SUBL용의 사인 연장된 32비트값 또는 SUBQ용의 64비트 값이다. 비사인된 비교명령은 바로우(borrow)를 테스트 하는데 사용될 수 있는데, 비사인된 미뉴엔드(Ra)가 비사인된 서브 트라헨드(Rb)보다 덜 비사인된 경우에는 바로우(borrow)될 수 있다.

논리명령은 연산 포맷으로 되어 있으며 과드 워드 부울대수 연산을 수행한다. 이들 명령은 다음과 같다.

AND - 논리곱

BIS - 논리합

XOR - 논리차

BIC - 상보성 논리곱

ORNOT - 상보성 논리합

EQV - 논리등가

이들 명령은 레지스터 Ra와 레지스터 Rb 또는 문자간에 지정된 부울대수 기능을 수행하며 그 결과를 목적 레지스터 Rc에 기입한다.

NOT기능은 제로(Ra=R31)를 가진 OR NOT를 행하므로써 수행될 수 있다.

변위명령은 연산 포맷으로 되어 있으며, 다음과 같이 좌우논리변위 및 시프트(41)에서 오른쪽 산술변위를 수행한다.

SLL - 좌논리변위

SRL - 우논리변위

SRA - 우산술변위

일반적으로, 산술적으로 좌측으로 변위가 사용되는 경우에는 논리변위가 행하여 지기 때문에 산술적인 좌측변위 명령이 없다. 어드레스 연산에 있어서 소량의 두 전력을 곱하기 위해 논리적으로 좌측변위가 이용된다. 산술적인 좌측변위는 그것이 오버플로워 검출을 요구하기 때문에 보다 복잡하게 된다. 정수곱은 오버플로워 검사로 산술적인 좌측변위를 수행하도록 사용되어야 한다. 비트 필드 추출은 2개의 논리변위로 수행될 수 있으며, 사인연장은 좌측논리변위 및 우측논리변위로 행해질 수 있다. 논리변위에 대하여, 레지스터 Ra는 레지스터 Rb 또는 문자의 계수에 의해 논리적으로 좌측 또는 우측의 0 내지 63 비트만큼 변위되며, 그 결과는 빈 비트위치로 전파된 제로 비트와 함께 레지스터 Rc로 기입된다. 이와 달리, 산술적으로 우측으로 변위되도록 하는 명령에 대하여, 레지스터 Rb는 레지스터 Ra 또는 문자에서의 계수에 의해 산술적으로 0 내지 63비트 만큼 우측으로 변위된다. 그 결과는 레지스터 Rc에 기입되며 이때 사인비트(R_{BV}^{63}가 빈 비트 위치속으로 전파된다.

처리능력 개선을 제공하는 중요한 특징은 조건부 이동 정수 명령이다. 이들 명령은 브랜치 없이 조건을 수행하며 명령 스트림의 순서를 유지한다. 이들 명령은 연산 포맷으로 구성되어 있으며 다음과 같은 것을 포함한다.

CMOVEQ - 레지스터가 제로일때 조건부 이동

CMOVNE- 레지스터가 제로가 아닐때 조건부 이동

CMOVLT - 레지스터가 제로보다 작을때 조건부 이동

CMOVLE - 레지스터가 제로와 같거나 작을때 조건부 이동

CMOVGT - 레지스터가 제로보다 클때 조건부 이동

CMOVGE - 레지스터가 제로와 동일하거나 클때 조건부 이동

CMOVLBC - 레지스터의 하위비트가 클리어 될때 조건부 이동

CMOVLBS - 레지스터의 하위비트가 세트될때 조건부 이동

이들 조건부 이동 명령을 실행하는데 있어서, 레지스터 Ra가 테스트이고 특정한 관계가 참이면 레지스터 Rb에서의 값이 레지스터 Rc에 기입된다. 이러한 대안성이 있으므로써 실행속도에 유리하다.

예컨대, 명령 CMOVEQ Ra, Rb, Rc는 CMOV 방식이 보다 고속화되도록 여러가지 구성으로 될 수 있는 사실을 제외하고는 정확히 다음 라벨과 동일하다. 즉 BNE Ra 라벨, OR R31, Rb, Rc 라벨… 2개의 레지스터의 보다 큰 크기를 찾아내기 위한 무브랜치 순서, 즉

R_1=MAX(R_1R_2)는

CMPLT R_1, R_2, R_3 : R_1R_2인 경우 R_3=1

CMOVEN R_3, R_2, R_1 : NOT(R_1R_2)가 이니면 실행

R_1R_2이면 R_2를 R_1으로 이동.

물론, 브랜치를 사용하지 않으므로써 얻는 장점을 명령 스트림이 연속적으로 추출된다는 점이며 명령 캐쉬 또는 선 추출행렬을 플러시 할 필요가 없다는 점이다. 조건부 이동은 브랜치가 올바르게 예상된다 하더라도 브랜치 보다 빠르다. 브랜치가 올바르게 예상되지 않으면, 이것이 브랜치 동작을 제거하기 때문에 조건부 이동이 고속으로 된다.

또 다른 중요한 특징은 레지스터 내에 바이트 오퍼랜드에 대한 연산을 위해 명령을 제공하는 것이다. 이들은 로드/스토어 명령에서 전폭의 64 비트 메모리 액세스를 제공하며 다양한 폭의 바이트 연산이 가능한 다양한 인 레지스터 바이트 조정명령과 결합된다. 이에 따른 장점은 메모리에서 바이트 연산을 제공하는 아키텍쳐용으로 기입된 코드를 사용할 수 있고, 쾌드 워드 정렬 경계를 채우도록 메모리의 액세스를 제한한다는 점이다. 바이트 조정명령은 제8도의 연산 포맷(72,73)으로 되어 있으며, 다음과 같이 비교바이트, 추출바이트, 마스크 바이트 및 제로 바이트 명령을 포함한다.

CMPBGE - 바이트 비교

EXTBL - 하위 바이트 추출

EXTWL - 하위 워드 추출

EXTLL - 하위 롱워드 추출

EXTQL - 하위 과드 워드 추출

EXTWH - 상위 워드 추출

EXTLH - 상기 롱워드 추출

EXTQH - 상위 과드 워드 추출

INSBL - 하위 바이트 삽입

INSWL - 하위 워드 삽입

INSLL - 하위 롱워드 삽입

INSQL - 하위 과드 워드 삽입

INSWN - 상위 워드 삽입

INSLH - 상위 롱워드 삽입

INSQH - 상위 과드 워드 삽입

MSKBL - 하위 바이트 마스크

MSKWL - 하위 워드 마스크

MSKLL = 하위 롱워드 마스크

MSKQL - 하위 과드 워드 마스크

MSKWH - 상위 워드 마스크

MSKLH - 상위 롱워드 마스크

MSKQH - 상위 과드 워드 마스크

ZAP - 제로 바이트

ZAPNOT - 비 제로 바이트

비교 바이트 명령은 레지스터 Ra와 Rb(또는 Ra과 문자)의 대응 바이트 간에 8개의 병렬된 비사인 바이트의 비교를 행하며, 레지스터 Rc의 하위 8개 비트에 8개의 결과를 스토어하고, 레지스터 Rc의 상위 56비트는 제로로 세트된다. Rc의 비트 0은 바이트-0, Rc의 비트 1대 바이트 등에 대응한다. 그 결과 비트는 Ra의 대응 바이트가 Rb(비사인)보다 크기가 동일할때 Rc에 세트된다.

추출 바이트 명령은 0 내지 7바이트 만큼 레지스터 Ra를 변위하고(하위를 위해 오른쪽으로 변위, 상위를 위해 왼쪽으로 변위), 그 다음에 레지스터 Rc에서 1, 2, 4 및 8바이트를 추출하고, 변위되는 바이트 수는 레지스터 Rb의 비트 2:0로 측정되고, 추출되는 바이트수는 기능코드로 특정되며, 나머지 바이트는 제로(0)로 채워진다. 추출된 바이트의 상위 명령은 8에서 레지스터 Rb의 비트 2:0만큼 특정된 양만큼을 뺀 바이트 수만큼 좌측으로 변위된다. 이들 추출바이트 명령은 바이트 조정에 특히 유리한데, 여기서 비정렬된 메모리의 다중바이트는 부록에 있는 바이트 추출에 대한 예에 제시한 바에 따라 동작하게 된다.

삽입바이트 명령은 레지스터 Ra로부터 바이트를 변위하여 그 바이트를 제로 필드로 삽입하고 레지스터 Rc에 그 결과를 스토어하고, 비트 2:0의 레지스터 Rb는 0 내지 7바이트의 변위량을 선택하며 기능코드는 1, 2, 4 또는 8 바이트의 필드폭을 선택한다. 이들 삽입 바이트 명령은 임의 바이트 정렬로 레지스터에 위치설정된 바이트, 워드, 롱워드 또는 과드 워드 데이타를 발생할 수 있다.

바이트 마스크 명령 MSKxL 및 MSKxH 세트는 제로(0)로 레지스터 Ra의 바이트를 선택하여 그 결과를 레지스터 Rc에 스토어하며, 레지스터 Rb 2:0는 제로 바이트 필드의 개시위치를 선택하며, 기능코드는 1, 2, 4 또는 8바이트의 최대폭을 선택한다. 마스크 명령은 임의 바이트 정렬로 레지스터로 확산될 수 있는 바이트, 워드, 롱워드 또는 제로의 과드 워드 필드를 발생시킨다.

제로 바이트 명령 ZAP 및 ZAPNOT 세트는 제로로 레지스터 Ra의 바이트를 선택하여 그 결과를 레지스터 Rc에 스토어하고, 레지스터 Rb7:0는 바이트를 제로로 선택하는데, 여기서 Rb의 비트 0는 바이트 9에 대응하고 Rb의 비트 1은 바이트 -1에 대응한다. 결과 바이트는 Rb의 대응 비트가 ZAP에 대해 1 및 ZAPNOT에 대해 제로인 경우 제로로 세트된다.

부록 A에 있어서, 명령 순서는 바이트 동작이 어떻게 전술한 바이트 명령세트를 사용하여 달성되는가를 나타내도록 주어져 있다.

부동 소숫점 명령은 5개 데이타 포맷 각각에서 소숫점 오퍼랜드에 따라 동작하는데, F_플로팅은 VAX 단정도이고, D 플로팅은 8비트 지수를 가진 VAX 배정도이며, G 플로팅은 11비트 지수를 가진 VAX 배정도이고, S 플로팅은 IEEE 단정도이며, 그리고 T플로팅은 11비트 지수를 가진 IEEE배 정도이다. 단정도 값은 64 비트 레지스터(61)의 상위 32비트로 로드되며 하위 32비트는 제로로 된다. 데이타 변환 명령은 소숫점과 과드 워드 정수포맷, 단정도와 배정도, 그리고 과드 워드와 롱워드 정수간의 오퍼랜드를 변환하도록 제공된다.

CPU(10)에 대해서는 어떤 글로발 플로팅 포인트 프로세서 상태도 없다.

즉 기계상태는 데이타 포맷간에 스위치 되지 않고 그 대신 데이타 포맷의 선택이 각 명령으로 인코드된다.

소숫점수는 3개의 필드로 표시되는데, 즉 사인(Sing), 지수(exponent) 및 분수이다. 사인필드는 1비트이고, 지수필드는 8 또는 11비트이며 분수는 23, 52 또는 55비트이다. 여러개의 다른 라운딩 모드가 제공되는데, VAX포맷에 대하여 라운딩이 정상(가압) 또는 찹(chopped)되며, 그 반면에 IEEE 포맷에 대해서는 라운딩이 가장 둥근 4가지 형태로 된다. 여기에서는 소숫점 명령에 의해 발생될 수 있는 6가지 예외가 있는데, 모든 것은 산술 예외 트램에 의해 신호되며 이들 예외는 동작을 무효화하고 제로만큼 제산, 오버플로워, 언더플로워, 정확하지 못한 결과 및 정수 오버플로워이다.

메모리 포맷의 부동 소숫점 명령은 다음과 같은 것을 포함한다.

LDF - F플로팅로드

LDD - D플로팅로드(G플로팅로드)

LDS - S플로팅로드(롱워드 정수 로드)

LDT - T플로팅로드(과드 워드 정수 로드)

LDF - F플로팅 스토어

STF - D플로팅 스토어(G플로팅 스토어)

STS - S플로팅 스토어(롱워드 정수 스토어)

STT - T플로팅 스토어(과드 워드 정수 스토어)

각각의 로드 명령은 메모리로부터 특정한 형태의 부동 소숫점 데이타를 추출하고, 이 형태의 소숫점 레지스터 포맷에 맞도록 바이트를 재배열한 다음 레지스터 세트(61)에 있는 레지스터 Fa에 그것을 기입하며, 가상 어드레스는 레지스트 Fb를 사인 연장된 16비트 변위에 추가함으로써 연산된다. 스토어 명령은 레지스터 Rb를 사인 연장된 16비트 변위에 추가함으로써 연산된 가상 어드레스로 레지스터 Fa의 내용이 메모리 위치에 스토어 되도록 하며, 바이트는 상기 소숫점 데이타 형태를 위한 메모리 포맷에 일치하지 않는 방식으로 재배열된다.

소숫점 브랜치 명령은 전술한 정수 브랜치 명령과 동일 방식으로 동작하는데, 소숫점 레지스터 Fa에서의 값이 테스트되며 PC는 조건부로 변하게 된다. 이들 소숫점 브랜치 명령은 다음과 같은 것을 포함한다.

FBEQ - 동일한 플로팅 브랜치

FBNE - 동일하지 않은 플로팅 브랜치

FBLT - 작은 플로팅 브랜치

FBLE - 동일하거나 작은 플로팅 브랜치

FBGT - 큰 플로팅 브랜치

FBGE - 동일하거나 큰 플로팅 브랜치

레지스터 Fa가 테스트되고 특정한 관계가 참인 경우 PC가 타켓 가상 어드레스로 로드되며, 이런 경우가 아닌 경우에 다음 순서 명령을 계속 실행한다. 변위는 사인된 롱워드 오프셋으로서 처리되는데, 이는 변위가 롱워드 경계를 어드레스 하도록 좌측으로 2비트 변위되고, 사인 연장된 64비트까지 변위된 다음 타켓 가상 어드레스를 형성하도록 갱신된 PC에 가산된다.

소숫점 산술을 위한 연산포맷 명령은 가산, 감산, 승산, 제산, 비교, 절대값, 카피를 포함하며 레지스터(61)에서 64 비트 레지스터 값에 대한 동작을 변환한다. 각 명령은 소스 및 수치로 되어 있는 목적 포맷을 특정화할 뿐만 아니라 사용되는 라운딩 모드 및 트랩핑 모드를 특정화 한다.

이들 소숫점 연산 명령은 테이블 B에 나열되어 있다.

소숫점 조건부 이동 명령은, 소숫점 레지스터(61)가 정수 레지스터(43) 대신에 사용된다는 점을 제외하고는 정수 조건부 이동 명령에 대응한다. 정수 조건부 이동과 함께 이들 명령들은 브랜치 명령을 제거하는데 사용될 수 있다.

CPU(10)는 그 명령 세트내에 여러가지의 잡다한(miscellaneous) 명령을 갖고 있는데, 그 모두는 전술한 명령 포맷을 이용하지만 이제까지 설명한 카테고리에는 맞지 않는다.

다음은 잡다한 명령들이다.

CALL - PAL - 호출 특권 아키텍쳐 라이브러리 루틴

FETCH - 데이타 블록 선추출

FETCH_M - 선추출, 의도수정

DRAINT - 명령 파이프 라인 드레인

MB - 메모리 장벽

RCC - 판독 사이클 카운터

제8도의 포맷(75)을 사용하는 CALL_PAL 명령은 PAL 코드(25:00 비트명령)에 대한 트랩(trap)의 원인이 된다. 이 명령은 예외없이 모든 이전의 명령이 완전히 보장될때까지 발생되지 않는다. 예외가 이들 이전의 명령중 하나에 대하여 발생되는 경우 예외 스택 프레임에 있는 연속성 PC는 CALL_Pal 명령을 가르킨다.

FETCH 명령은 Rb의 내용으로 주어진 가상 어드레스를 포함하는 정렬된 512 바이트 블록을 선추출한다. Rb에 있는 이 어드레스는 정렬된 512바이트의 블록 데이타를 지정하는데 사용된다. 이 동작은 512 바이트 블록 데이타(큰블록)의 전부 또는 일부분을 메모리 계층의 고속 액세스부로 이동시키려는 것이다. 이와같이 FETCH 명령은 고속 실행을 제공할 수 있는 CPU(10)에 대한 힌트이다. 특정한 CPU의 구성이 이 기술을 실시하지 못하면 힌트는 무시될 수 있다.

FETCH_M 명령은 데이타의 몇몇 또는 전부에 대한 수정이 예상되는 추가의 힌트를 제공하는데, 이것은 몇몇의 라이트 백 캐시 설계에 고속 동작을 제공한다. 그 이유는 데이타 블록이 소유자로써 10개의 캐시속으로 리드되기 때문인데, 따라서 기입이 캐시에 있는 블록데이타로 실행되는 경우 고장을 발생시키지 않는다. FETCH에 의해서는 예외가 발생되지 않는데, 동일한 어드레스를 이용하는 로드(FETCH_M 경우에 있어서의 스토어)가 잘못된 경우 선추출 요구가 무시된다. FETCH 명령은 100 사이클 정도의 소프트 버리(bury) 메모리 대기시간에 도움을 주기 위한 것이며, 10 사이클 정도의 메모리(대기 시간은 그다지 구성상 중요하지 않는데, 그 이유는 코드스케쥴링이 상기 짧은 대기시간을 감소시키기 위해 사용되기 때문이다.

DRAINT 명령은 산술 트랩을 발생시키지 않고 모든 종래의 명령이 완전히 보장될때까지 명령 발생을 스톨한다. 이것은 파이프 라인식 구성에 있어서, 모든 이전의 산술 명령이 DRAINT가 발생된 후의 임의의 명령전에 임의의 산술 트랩을 발생함이 없이 이전의 산술 명령이 완료되는 것을 소프트웨어로 하여금 보장할 수 있도록 하기 위함이다. 예컨대, 이전 명령에 대한 모든 예외가 현재의 예외처리 분위기에서 확실히 처리되도록 예외 처리자를 변화시키기 전에 사용된다.

메모리 장벽 명령 MB는 모든 이전의 로드 및 스토어가 완료될때까지 모든 미래의 로드 또는 스토어가 완료되지 않도록 하는 것을 보장한다.

MB 명령의 부재시, 다른 물리적 위치에 대한 로드 및 스토어는 순서없이 완료되도록 허용된다. MB 명령은 메모리의 액세스가 직렬로 되게 한다.

판독 사이클 카운터 명령 RCC은 레지스터 Ra가 CPU 사이클 카운터의 내용으로 기입되도록 한다. 사이클 카운터의 하위 32비트는 N개의 CPU 사이클당 증분되는 비사인된 정수인데, 여기서 N은 범위 1 내지 16으로 구성된 특정한 정수이다. 카운터는 특정하게 구성된 값에서 제로로 된다.

이제까지 본 발명을 전술한 특정 실시예와 관계하여 설명하였지만, 본 발명에 숙련된 자는 본 발명의 영역 및 범위를 벗어남이 없이 본 발명에 여러가지 수정 및 변화가 가능함을 알 수 있을 것이다.

테이블 A 페이지 테이블 엔트리

페이지 테이블 엔트리의 필드는 다음과 같이 설명된다 :

비트	설명
<0>	유효 (V)-PFN 필드의 유효성을 나타낸다.
<1>	폴트온리드(FOR)- 세트시, 폴트온리드 예외가 페이지에 임의의 위치를 리드하려는 시도시에 발생한다.
<2>	폴트온라이트(FOW)- 세트시 폴트온라이트 예외가 페이지에 임의의 위치를 라드트할려는 시도시에 발생한다.
<3>	폴트온실행(FOE)- 세트시 폴트온 실행예외가 페이지에명령 실행을 할려는 시도시에발생한다.
<4>	어드레스 공간매치(ASM)- 세트시 상기 PTE 가 모든 어드레스 공간수와 일치한다. 주어진 VA 에 대하여 ASM이 시종일관 모든 처리에서 세트 되어야한다.
<6:5>	그래뉼어리티 힌트 (GH)
<7>	미래사용을 위해 보존
<8>	커널리드 이네이블(KRE)이 비트는 커널보드로부터 리드를 이네이블한다. 이비트가 0이고 LOAD 또는 명령추출이 커널모드 동안시도되면 액세스 위반이 발생한다. 이 비트는 V=0 일때도 유효하다.
<9>	실행리드 이네이블(ERE)- 이비트는 실행모드로부터 리드를 이네이블 한다. 이 비트가 0이고 실행모드 동안 LOAD 또는 명령추출이 시도 되면 액세스 위반이 발생한다. 이 비트는 V=0 일때도 유효하다.

<10> 슈퍼바이져 리드이네이블(SRE)- 이비트는 슈퍼바이져 모드로부터 리드를 이네이블한다. 이 비트가 0이고 슈퍼바이져 모드동안 LOAD 또는 명령추출이 시도되면 액세스 위반이 발생한다. 이 비트는 V=0 일때도 유효하다.

<11> 유저리드 이네이블(VRE)- 이비트는 유저모드로부터리드를 이네이블한다. 이 비트가 0이고 유저모드 동안 LOAD 또는 명령추출이 시도되면 액세스 위반이 발생한다. 이 비트는 V=0 일때도 유효하다.

<12> 커널라이트 이네이블(KWE)- 이비트는 커널모드로부터 라이트를 이네이블한다. 이 비트가 0이고 커널모드 동안 STORE 가 시도되면 액세스 위반이 발생한다. 이 비트는 V=0 일때 유효하다.

<13> 실행라이트 이네이블(KWE)- 이 비트는 실행 모드로부터 라이트를 이네이블한다. 이 비트가 0이고 실행모드 동안 STORE 가 시도되면 액세스 위반이 발생한다.

<14> 슈퍼바이져 라이트 이네이블(SWE)- 이 비트는 슈퍼바이져 모드로부터 라이트를 이네이블한다. 이 비트가 0이고 슈퍼바이져 모드동안STORE 가 시도되면 액세스 위반이 발생한다.

<15> 유저라이트 이네이블(UWE)- 이비트는 유저모드로 부터 라이트를 이네이블한다. 이 비트가 0이고 유저모드 동안 STORE 가 시도되면 액세스 위반이 발생한다.

<31:16> 소프트웨어를 위해보존

<63:32> 페이지 프레임번호(PFN)-PFN 필드는 항상 페이지 경계를 가르킨다. v 가 세트되면 PFN이 물리적 어드레스를 얻도록 가상어드레스의 페이지 비트내에서 바이트와 연결된다. v 가 클리어되면 이 필드는 소프트웨어에 의해 사용될 수 있다.

테이블B - 부동 소숫점 산술연산

니모닉	비트 연산
CPYS	사인 카피
CPYSN	사인 카피 부정
CPYSE	사인 및 지수 카피
CPYSEE	사인 및 확장된 지수 카피
CVTQL	콰드워드를 롱워드로 변환
CVTLQ	롱워드를 콰드워드로 변환
FCMOV	플로팅 조건부 이동

니모닉	산술 연산
ADDF	F 플로팅 가산
ADDD	D 플로팅 가산
ADDG	G 플로팅 가산
ADDS	S 플로팅 가산
ADDT	T 플로팅 가산
CMPD	D 플로팅 비교
CMPG	G 플로팅 비교
CMPS	S 플로팅 비교
CMPT	T 플로팅 비교
CVTDQ	D 플로팅을 콰드워드로 변환
CVTGQ	G 플로팅을 콰드워드로 변환
CVTSQ	S 플로팅을 콰드워드로 변환
CVTTQ	T 플로팅을 콰드워드로 변환
CVTQD	콰드워드를 D 플로팅으로 변환
CVTQF	콰드워드를 F 플로팅으로 변환
CVTQG	콰드워드를 G 플로팅으로 변환
CVTQS	콰드워드를 S 플로팅으로 변환
CVTQT	콰드워드를 T 플로팅으로 변환
CVTFG	F 플로팅을 G 플로팅으로 변환
CVTDF	D 플로팅을 F 플로팅으로 변환
CVTGF	G 플로팅을 F 플로팅으로 변환
CVTST	S 플로팅을 T 플로팅으로 변환
CVTTS	T 플로팅을 S 플로팅으로 변환

DIVF	F	플로팅	제산
DIVD	D	플로팅	제산
DIVG	G	플로팅	제산
DIVS	S	플로팅	제산
DIVT	T	플로팅	제산
MULF	F	플로팅	승산
MULD	D	플로팅	승산
MULG	G	플로팅	승산
MULS	S	플로팅	승산
MULT	T	플로팅	승산
SUBF	F	플로팅	감산
SUBD	D	플로팅	감산
SUBG	G	플로팅	감산
SUBS	S	플로팅	감산
SUBT	T	플로팅	감산