특집

인텔 MMX™ 기술고찰

金 京 曉
인텔 코리아

I. 서 론

오늘날의 PC가 처리하는 데이터 양과 복잡성은 점점 더 심화되가고 있기 때문에 마이크로프로세서에 대한 의존도가 커지고 있다. 새롭게 발표되는 통신, 게임 및 에듀테인먼트(Edutainment) 애플리케이션은 비디오, 3D, 애니메이션, 오디오, 가상현실 등을 제공하고 있기 때문에 강력한 마이크로프로세서 성능에 의존하는 정도가 한층 더해가고 있다.

MMX™ 기술은 멀티미디어 및 통신 애플리케이션을 가속시킬 수 있도록 설계되었다. MMX기술에는 새로운 명령어 및 데이터 형태가 포함되어 있으며 멀티미디어 및 통신 알고리즘이 주로 추구하는 병렬 프로세싱 기능을 십분 활용하면서도 기존의 운영체계와 애플리케이션과는 완벽한 호환성을 유지, 애플리케이션들이 좀 더 강력한 성능을 발휘할 수 있도록 해준다.

MMX기술은 인텔 아키텍처를 32비트로 확장시킨 인텔 386 프로세서이후 인텔 아키텍처에 추가된 가장 중요한 아키텍처 향상으로57개의 새로운 명령을 추가하여 오디오, 2D 및 3D그래픽, 비디오, 음성합성 및 인식, 데이터 알고리즘(Data Algorithm) 등에 사용되는 수리계산을 가속화 시켜준다. MMX기술이 장착된 프로세서는 강력한 컴퓨팅 능력이 요구되는 통신 및 멀티미디어 애플리케이션을 충분하게 실행할 수 있는 성능을 제공하면서도 동시에 다른 작업이나 애플리케이션을 추가로 실행할 수 있는 여유의 성능도 제공한다. 또한 소프트웨어 개발자들이 좀더 멀티미디어 기능이 풍부하고 멋진 애플리케이션을 개발할 수 있도록 해준다. MMX기술이 내장된 프로세서 기반 시스템은 인텔 프로세서들에 장착되는 것과 발맞추어 1997년 부터 대량으로 사용될 것으로 예상된다.

II. 본 론

MMX기술은 인텔 마이크로프로세서 아키텍처 개발자들과 소프트웨어 개발자들의 공동노력에 의해서 결정을 본 것으로 이들은 먼저 그래픽, MPEG 비디오, 전자음악, 음성인식, 이미지 프로세싱, 게임, 화상회의 등의 광범위한 소프트웨어를 분석했다. 위에 열거한 애플리케이션들은 가장 컴퓨팅 성능을 많이 사용하는 루틴(Routines)을 찾아내기 위해 그룹지워졌으며 최신 CAE 툴을 이용 섬세하게 분석이 되었다. 이러한 분석결과 모든 소프트웨어군에 걸쳐 적용되는 공통의 기본적인 특성들을 발견할 수 있었다. 여기서 발견된 특성들을 열거해 보면

- 소규모의 정수 데이터 형태(일례로 8비트 그래픽 픽셀, 16비트 오디오 샘플 등)
- 소규모 고도의 반복적인 루프(loops)
- 반복되는 곱하기 후 더하기(Multiples and accumulates)
- 컴퓨팅 성능을 많이 사용하는 알고리즘
- 고도의 병렬적인 작동

이러한 소프트웨어들을 가속시키기 위한 MMX 기술은 기본적인 범용 정수 명령들로 이루어져있으며 멀티미디어 및 통신 애플리케이션에 쉽게 사용될 수 있다.

MMX기술의 가장 중요한 특징은

- 단일 명령으로 64비트폭의 다중 데이터(Single Instruction Multiple Data)를 병렬로 처리
- 57개의 새로운 MMX 명령어
- 새로운 8개의 64비트폭의 MMX 레지스터
- 4개의 새로운 데이터 형태를 추가 한것이라 할수 있다.

MMX기술의 기반은 단일 명령, 다중 데이터(SIMD, Single-Instruction Multiple-Data)라고 하는 것으로 이것은 여러종류의 정보가 64비트로 묶음화된후 단일 명령으로 처리될 수 있도록 해 병렬처리를 가능하도록 해주기 때문에 성능이 향상된다. MMX기술과 인텔 아키텍처의 수퍼스케일러 아키텍처는 함께 짝을 이루어 PC 성능을 현저히 향상시킬 것이다. MMX기술은 인텔 아키텍처 프로세서에 향후 내장될 예정이며 MS-DOS, 윈도우 3.1, 윈도우 95, OS/2, 유닉스 등 기존의 모든 운영체계와 100% 호환성을 유지하게 된다. 또한 인텔 아키텍처에 기반한 모든 소프트웨어들이 MMX 기술이 장착된 PC에서도 완벽하게 실행된다.

MMX기술은 아주 간단하게 정의된다. MMX기술은 현존 또는 향후 알고리즘을 기반으로 개발된 대부분의 PC 애플리케이션의 요구를 충족시켜준다. MMX 명령어는 특별한 것이 아니며 애플리케이션, 코덱(codec), 알고리즘, 드라이버 등에 사용될 수 있다.

1. MMX 데이터 형태

MMX 명령어 세트의 기본적인 데이터 형태는 복수의 정수 워드가 단일 64비트로 그룹화된 묶음 고정 정수(packed fixed-point integer)이다. 지원되는 데이터 형태는 부호화(signed) 혹은 비부호화된(Unsigned) 고정 정수, 바이트, 워드, 중복워드, 4중복워드 등이다.

네가지의 MMX기술 데이터 형태는 다음과 같다.

묶음 바이트	하나의 64비트 분량으로 그룹지워진 8개의 바이트
묶음 워드	하나의 64비트 분량으로 그룹지워진 4개의 16비트 워드
묶음 중복워드	하나의 64비트 분량으로 그룹지워진 2개의 32비트 중복워드
4중복워드	하나의 64비트 분량

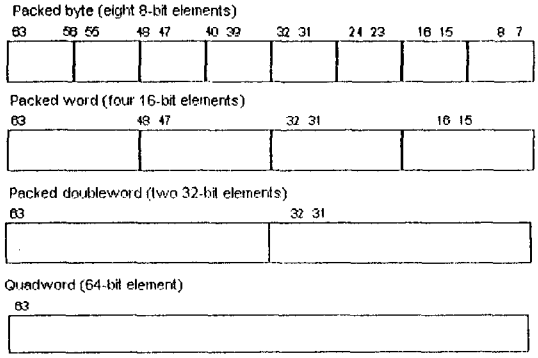
일례로 그래픽 픽셀 데이터는 보통 8비트 정수 혹은 바이트들로 이루어져 있다. MMX기술을 이용 8개의 픽셀을 그룹화하여 하나의 64비트로 만든다음 MMX레지스터로 이동시킨다. MMX명령이 실행되면 8개의 픽셀값을 MMX 레지스터에서 한번에 취하여 8개의 엘레멘트 모두에 대한 수리적 및 논리적 수행을 병렬로 처리한 다음 그 결과

를 MMX레지스터에 기록한다.

2. MMX 64비트 레지스터

MMX 기술은 8개의 64비트 일반범용 레지스터를 제공하며 이 레지스터는 부동소숫점 레지스터에 매핑(aliasing)하여 사용한다.

이들 64비트 레지스터들은 그림 1과 같은 64비트 묶음 데이타들을 처리하게된다.



〈그림 1〉 MMX 64비트 레지스터

3. MMX 호환성

MMX기술은 MMX 레지스터 및 스테이트를 부동소수점 레지스터 및 스테이트에 상에 매핑하여(aliasing)하여 사용함으로써, 기존의 운영체계 및 애플리케이션과 완벽한 호환성을 유지한다. (부동소숫점 레지스터들이 MMX명령코드 수행시 MMX레지스터들로 이용됨) 즉 MMX기술을 지원하기 위하여 물리적으로 새로운 레지스터나 스테이트 추가는 없다는 것이다. 운영체계가 MMX 코드를 저장하고 복원하기 위해서 부동소수점 스테이트와 상호작용하는 기본적인 표준 메카니즘을 이용하기만 하면된다. 일례로 과업변환(task switch)에 있어서 운영체계는 부동소수점이나 MMX 코드를 유지하기 위해 기존에 존재하던 FSAV와 FRSTR명령을 사용하게 된다.

부동소수점 스테이트를 저장/복원하는 부동소수점 명령은 MMX스테이트도(일례로 컨텍스트 변환시) 처리한다. 다시말해서 운영체계와 상호작용하기 위한 부동소수점 아키텍처에서 사용하는 기술이 MMX기술에서도 사용된다는 것이다. MMX 기술은 새로운 예외(exception)나 스테이트 정보를 추가하지 않기 때문에 현재의 운영체계는 MMX명령을 사용하는 애플리케이션을 수행할 수 있다.

4. MMX 기술 존재여부 검색

인텔 마이크로프로세서에 MMX기술이 장착되어 있는지 여부를 알기 위해서는 CPUID 명령을 실행시켜서 세트 비트를 검색해 보면된다. 이러한 방법을 통하여 소프트웨어 개발자들은 자신들이 개발하는 소프트웨어가 실행할 수 있는 특수한 코드가 무엇인지를 알 수 있다. 설치나 소프트웨어를 실행할 때 소프트웨어는 마이크로프로세서에게 MX기술이 지원되는지를 문의한 다음 MMX코드가 포함되어 있거나 포함되어있지 않는가 여부에 따라 프로그램을 설치하거나 실행한다.

5. MMX 명령어

MMX명령어는 다음과 같은 기능적 분야를 내포한다.

- 더하기, 빼기, 곱하기, 수학적 쉬프트(Arithmetic shift), 곱하고-더하기 등 기본적인 수학적 작업
- 비교 작업
- 데이터를 함께 묶음하기(pack), 작은 데이터 형태에서 큰 데이터 형태로 묶음풀기(unpack) 등 새로운 데이터 형태간의 변환을 하기 위한 변환 명령
- AND, AND NOT, OR, XOR 등과 같은 논리적 작업
- 쉬프트 작업
- MMX 레지스터에서 레지스터로의 전송, 메모리에 64비트 및 32비트 로드/저장 등을 위한 데이터 전송(MOV) 명령

수학적 및 논리적 명령은 여러가지 묶음 정수 데이터 형태를 지원하도록 설계되었다. 이러한 명령은 지원되는 데이터 형태에 따라 각기 서로 다른 op 코드를 가지고 있다. 그 결과로 새로운 MMX

기술 명령은 57op 코드로 되어 있다.

MMX 기술은 인텔 프로세서에서 쉽게 신속하게 병렬 파이프라인으로 할당될 수 있는 범용 기본적인 명령이다. 이렇게 범용 접근방식을 사용함으로써 MMX 기술은 현존하는 인텔 마이크로프로세서 뿐만이 아니라 차세대 프로세서의 성능을 향상할 수 있도록 한다.

1) MMX 명령어 세트 요약

아래 표에 MMX 명령과 이에 상응하는 Mnemonic이 기능적인 범주에 따라 분류되어 있다.

〈표 1〉 MMX 명령어 세트 요약

	Mnemonic	Number of Different Opcodes	Description
Arithmetic	PADD「B,W,D」	3	Add with wrap-around on 「byte, word, doubleword」
	PADDS「B,W」	2	Add signed with saturation on 「byte, word」
	PADDUS「B,W」	2	Add unsigned with saturation on 「byte, word」
	PSUB「B,W,D」	3	Subtract with wrap-around on 「byte, word, doubleword」
	PSUBS「B,W」	2	Subtract signed with saturation on 「byte, word」
	PSUBUS「B,W」	2	Subtract unsigned with saturation on 「byte, word」
	PMULHW	1	Packed multiply high on words
	PMULLW	1	Packed multiply low on words
	PMADDWD	1	Packed multiply on words and add resulting pairs
Comparison	PCMPEQ「B,W,D」	3	Packed compare for equality 「byte, word,doubleword」
	PCMPGT「B,W,D」	3	Packed compare greater than 「byte,word, doubleword」
Conversion	PACKUSWB	1	Pack words into bytes(unsigned with saturation)
	PACKSS「WB,DW」	2	Pack 「words into bytes, doublewords into words」 (signed with saturation)
	PUNPCKH「BW,WD, DQ」	3	Unpack(interleave) high-order 「bytes, words, doublewords」 from MMX register
	PUNPCKL「BW,WD, DQ」	3	Unpack(interleave) low-order 「bytes, words, doublewords」 from MMX register
Logical	PAND	1	Bitwise AND
	PANDN	1	Bitwise AND NOT
	POR	1	Bitwise OR
	PXOR	1	Bitwise XOR
Shift	PSLL「W,D,Q」	6	Packed shift left logical 「word, doubleword, quadword」 by amount specified in MMX register or by immediate value
	PSRL「W,D,Q」	6	Packed shift right logical 「word, doubleword, quadword」 by amount specified in MMX register or by immediate value
	PSRA「W,D」	6	Packed shift right arithmetic 「word, doubleword」 by amount specified in MMX register or by immediate value
Data Transfer	MOV「D,Q」	4	Move 「doubleword, quadword」 to MMX register or from MMX register
FP & MMX State Mgmt	EMMS	1	Empty MMX state

2) MMX 명령 예제들

아래에는 MMX 기술에 대한 5가지의 예제를 간단한 설명하고 있다. 이해를 돕기 위해 여기에 소개하는 것은 16비트 워드 데이터 형태로 되어있는데 참고로 아래에 설명되어 있는 예제는 8비트 혹은 32비트 묶음 데이터 형태에서도 있을 수 있다.

a) 아래의 예제는 wrap around하는 묶음된 더하기 워드(PACKED ADD WORD)를 보여주고 있다. 8개의 16비트 엘레멘트로써 4개의 더하기 연산을 한번에 하는데 모든 더하기 연산은 서로 독립적이며 병렬로 수행된다. 이 예제에서 맨오른쪽의 결과는 16비트에서 나타나는 최대치를 넘게된다. 따라서 wrap around를 하게 된다. 이것은 보통의 인텔 아키텍춰의 수학적 행태이다. FFFFh + 8000h는 17비트 결과치를 가져온다. 그러나 17번째 비트는 wrap around하기 때문에 없어지기 때문에 결과는 7FFFh가 된다.

a3	a2	a1	FFFFh
+	+	+	+
b3	b2	b1	8000h
a3+b3	a2+b2	a1+b1	7FFFh

b) 다음의 예제는 부호화되지 않은(unsigned) 포화상태(saturated)로 묶음된 더하기 워드명령(Pack Unsigned ADD with Saturation ,Word)이다. 여기에서는 위에서 이용했던 것과 동일한 데이터 값을 사용한다. 왼쪽 끝에 있는 더하기는 16비트내에 들어갈 수 없는 결과가 나오고 따라서 이 예제에서는 포화가 발생한다. 포화란 더하기 결과가 overflow 되거나 빼기 결과가 underflow가 되는 것을 말한다. 부호화된지 않은(unsigned) 16비트 워드에서 표현할 수 있는 최대치와 최소치는 각각 FFFFh와 0×0000이고 부호화된 16비트 워드의 경우의 최대치와 최소치는 각각 7FFFh와 0×8000이다. 앞에 열거된 사항은 일례로 3D 그래픽 구로드 쉐이딩 룹(Gouraud shading loop)상황에서 화소(畵素, pixel) 연산을 하던중 검은 화소가 갑자기 흑색으로 바뀌는 wrap around 더하기를 방지하기 때문에 대단히 중요하다.

a3	a2	a1	FFFFh
+	+	+	+
b3	b2	b1	8000h
a3+b3	a2+b2	a1+b1	FFFFh

여기에 설명된 명령은 부호화되지 않은 묶음 더하기 포화 워드 (PADDUSW)이다. 묶음 더하기 포화 연산은 부호화된 혹은 부호화되지 않은 경우 두가지가 각기 존재한다. 부호화되지 않은(65,535 10진수)로 처리되는 FFFFh란 숫자는 부호화되지 않은 0x8000(32,768)에 더해저 그 결과 부호화되지 않은 16비트 값으로 최대치인 FFFFh로 되며 동시에 포화된다.

c) 다음의 예제는 곱하기 더하기 연산(multiply add) 에 사용되는 명령을 보여주는 데 이것은 벡터-도트-프로덕트, 매트릭스 멀티플라이어, FIR, FIR필터, FFT, DCT 등의 시그널 프로세싱 알고리즘에서 기초가 되는 것이다. 이 명령은 묶음 곱하기 더하기 명령(PMADD)이다.

PMADD명령은 16비트 묶음 데이터 형태에서 시작하여 32비트 묶음 데이터 형태를 도출해 낸다. 이 명령은 모든 상관된 엘레멘트를 곱하여 4개의 32비트 결과를 도출해 낸다음 왼쪽의 두개를 더하여 하나의 결과를 오른쪽의 두개를 더하여 다른 하나의 결과를 도출해 낸다. 곱하기 더하기 연산을 완성하기 위해서 위에서 나온 두 개의 결과는 accumulator로 이용되는 다른 하나의 레지스터에 더해지게 된다.

a3	a2	a1	a0
*	*	*	*
b3	b2	b1	b0
a3*b3+a2*b2		a1*b1+a0*b0	

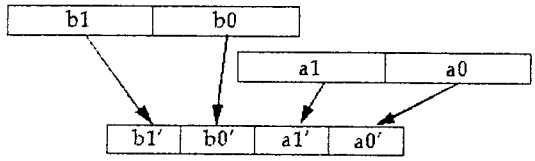
d) 아래 예제는 묶음 병렬 비교이다. 여기에서는 4쌍의 16비트 워드를 비교, 참(FFFFh)와 거짓(0000h)라는 결과를 만들어 낸다. 이 결과는 각각의 참의 조건에서 1 또는 거짓의 조건에서 0이라는 값을 갖는 묶음 마스크(packed mask)이다. 이 예제에서는 묶음 워드 데이터에서 "~보다 큰" 비교의 예를 보여준다. 새로운 조건 코드 플래그가 있거나 이 명령에 의해 영향을 받는 인텔 아키텍춰 조건 코드 플래그가 존재하지 않는다.

묶음 비교 결과는 브랜치나 일련의 브랜치 명령을 이용하지 않고 대신 논리동작을 이용하여 서로 다른 입력중에서 엘레멘트를 선택하는 매스크로 사용될 수 있다. 브랜치 명령대신 논리동작을 사용하는 것은 깊은 파이프라인을 가지고 있거나 분기예측(브랜치 프레딕션)을 사용하는 고성능의 프로세서의 성능 향상에 있어서 매우 중요하다. 입력중인 데이터의 비교연산 결과에 근거한 브랜치는 보통 예측하기가 어렵다. 그 이유는 입력중인 데이터는 무작위로 변할 수 있기 때문이다. 조건 선택능력을 이용하여 데이터 선택에 사용되는 브랜치를 없애버리는 것은 MMX 명령의 병렬연산과 함께 MMX 기술의 핵심이 되는 기능이다.

23	45	16	34
gt?	gt?	gt?	gt?
31	7	16	67
0000h	FFFFh	0000h	0000h

e) 아래는 묶음(pack) 변환 명령 예제이다. 이 명령은 4개의 32비트 값을 취하여 4개의 16비트 값을 가지도록 묶음을 하게 되며 32비트 소스 값이 16비트 결과에 맞추어 넣을수 없으면 포화(saturation)를 하게 된다. 또한 정 반대의 작업을 수행하는 명령도 있다. 일례로 묶음된 바이트 데이터 형태를 묶음된 워드 데이터 형태로 묶음을 푸는 것이다.

묶음명령과 묶음풀기 명령 등은 새롭게 묶음된 데이터 형태들을 변환하기 위해 존재한다. 이것은



이미지 필터링과 같이 중급 연산에서 고도의 정확성이 요구되는 알고리즘이 필요로 할때 중요하다. 이미지에 사용되는 필터는 보통 필터 상호계수와 일군의 인접 이미지 화소사이의 모든 값을 더하는 일군의 곱하기 연산을 포함한다. 이러한 곱하기-더하기 연산은 화소의 원래 데이터 형태인 8비트보다 더 정확하여야 한다. 이를 해결하기 위해선 이미지의 8비트 화소를 16비트 워드로 묶음풀기해서 overflow 염려없이 16비트 워드 형태로 연산을 하는 것이다. 그런 다음 필터를 거친 화소를 메모리에 저장하기 전에 8비트 화소형태로 다시 묶음을 하는 것이다.

6. MMX를 사용한 애플리케이션 예제

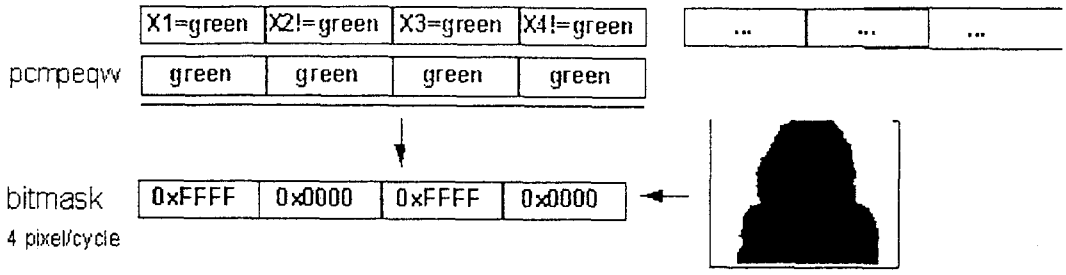
이제부터는 기본적인 코드 구조를 사용하기 위한 MMX 명령어 세트를 사용한 실제 응용 예제를 보여준다.

1) 조건부 선택

멀티미디어 애플리케이션은 반드시 대량의 데이터 집합을 처리해야만 한다. 때에 따라서는 입력중인 데이터를 대상으로 이루어진 조건 쿼리(query)를 기반으로 데이터를 선택해야 할 때가 있다. 인텔은 인텔 프로세서에 성능 향상 및 깊은 파이프라인 등의 마이크로 아키텍처 기능을 채택하여 성능을 향상시켰다.

파이프라인 구조에서 잘못된 분기 예측은 파이프라인을 지우고 새로이 채우기 시작하여야 하므로 이때 성능이 저하된다. 그러므로 분기예측(브랜치 프레딕션)은 파이프라인 수행에 아주 중요하다. 아래에 소개되는 예제는 특히 데이터 의존적이기때문에 예측을 하기가 매우 여러운 상황에서 어떻게 브랜치 명령의 사용을 효율적으로 제거하는가를 보여준다. 그로마 키잉 예제는 MMX 명령어 세트를 이용한 조건 선택이 어떻게 브랜치 예측오류를 줄여주는 가를 보여주는 좋은 예제이다. 아울

〈그림 2〉



〈그림 3〉

러 어떻게 다중 선택 연산을 병렬로 처리하는 가도 보여준다.

. 크로마 키잉

TV 일기예보에서 아나운서가 기상도 이미지와 중첩된 화면은 흔하게 볼 수 있다. 이 예제에서 여자의 이미지를 봄에 활짝핀 꽃 화면과 중첩하기 위해서 녹색 스크린을 사용했다. (그림 2 참조) 여기에서는 4개의 16비트 화소를 병렬로 처리하는 것을 보여주며 또한 MMX 명령들이 8개의 8비트 화소를 병렬로 처리해서 성능을 향상시킬 수 있도록 해준다.

맨 먼저 녹색 배경에 있는 여자의 그림에서 4개의 화소를 취한다. (그림 3). 아래의 데이터에서 맨 위쪽줄은 녹색, 비녹색, 녹색, 비녹색 등을 반복하는 화소를 나타낸다. 묶음 비교 명령은 이 데이터들을 위한 매스크를 만들어 준다. 이 매스크는 참 혹은 거짓을 나타내는 불리언 (boolean) 값의 모두 0 혹은 모두 1을 나타내는 워드의 연속으로 되어있다. 이렇게 하면 어떤 것이 원하는 배경이고 어떤 것이 아닌지를 알 수 가 있다. 결과는 아래 그림 3 의 shadow picture를 이용하여 나타나 있다.

이 매스크는 여자가 있는 화면에서 대응되는 4개의 화소와 마찬가지로 봄꽃이 화창한 화면으로부터 대응되는 4개의 화소에 사용된다. (그림 4)

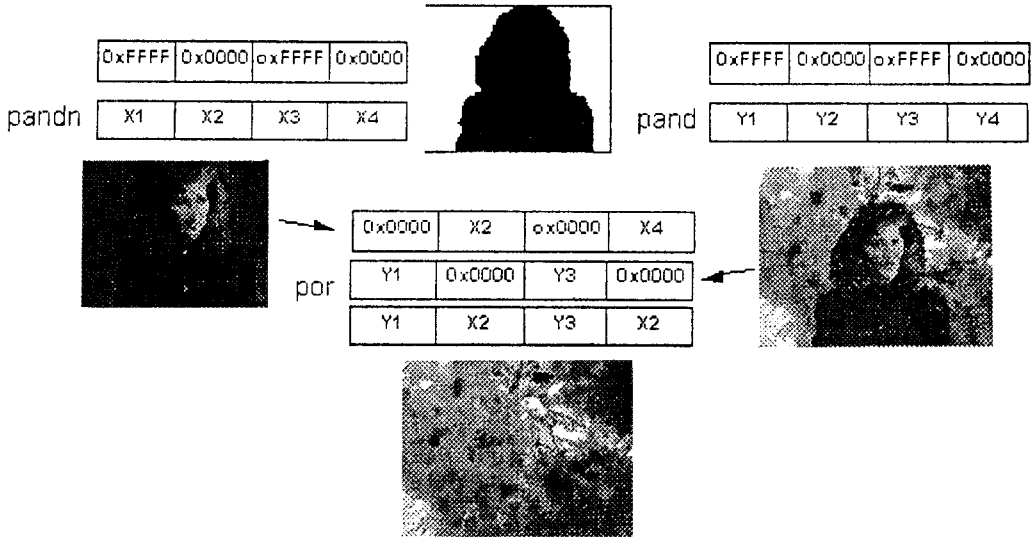
"AND NOT", "AND" 등의 명령이 매스크를 이용하여 봄꽃이 화창한 화면과 여자가 있는 화면에서 남겨 둬야할 화소를 가려낸다. 이제 "OR"명령이 필요없는 화소를 0으로 변화시키며 최종 완성된 화면을 만들어 낸다. 4개의 화소가 단지 4개의 MMX명령으로 분기명령(브랜치) 없이 맵핑되었다.

이 예제를 실행하면서 PANDN명령은 AND작업을 하기전에 매스크의 모든 비트를 변환한다.

MMX기술을 사용하지 않으면 각각의 화소는 따로 처리되며 조건 브랜치가 필요하지만 MMX명령을 이용하면 8개의 8비트 화소를 병렬로 처리할 수 있으며 조건 브랜치도 필요없게 된다.

2) Vector Dot Product

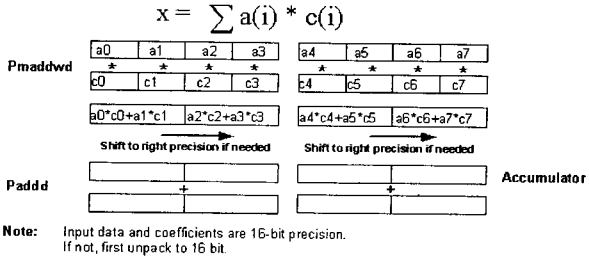
Vector dot product는 이미지, 오디오, 비디오, 사운드와 같은 내츄럴 데이터의 싱글 프로세싱에 이용되는 가장 기본적인 알고리즘중의 하나이다.



〈그림 4〉

아래의 예제는 PMADD 명령이 vector dot product를 사용하는 알고리즘의 처리속도를 어떻게 향상시키는지를 보여준다. PMADD명령은 4개의 곱하기와 2개의 더하기를 한번에 처리한다. 앞에서 설명된 바와 같이 PMADD 명령은 PADD명령과 짝을 이루어 8개의 곱하기 더하기 작업을 수행하는데 8개의 엘레멘트 벡터는 2개의 PMADD명령과 2개의 PADD명령에 이루어진다.

PMADD 명령에 의해 지원되는 정확도가 만족하다고 가정할때 8개 엘레멘트 벡터에 대한 dot-product 예제는 8개의 MMX 명령 즉 2개의 PMADD, 2개의 PADD, 2개의 쉬프트(곱하기 작업후 정확도를 맞추기 위해 필요할 경우), 하나의 벡터를 로드하기위한 2개의 메모리 이동명령(다른 하나의 벡터는 PMADD명령을 이용 로드할 수 있으며 PMADD명령은 메모리에서 온 오퍼랜드를 이용하게 된다) 등을 이용 끝낼 수 있다.



MMX기술이 사용되었을 때에 그렇지 않을 때의 명령을 비교해보면 다음과 같은 결과가 나온다. MMX기술을 사용하면 1/3의 명령만 필요하게 된다.

대부분의 MMX명령은 하나의 클럭 사이클에 수행될수 있기 때문에 단순한 명령어 갯수의 비율보다 더욱 향상된 성능을 기대할 수 있다.

3) 매트릭스 곱하기

흥미진진한 3D게임이 거의 매일 발표되고 있다. 보통 3D오브젝트를 만드는 컴퓨팅작업은 4X4 매트릭스에 기반을 두고 있으며 이것은 4개의 엘레멘트벡터를 여러번 곱해서 이루어진다. 벡터는 X, Y, Z축을 가지고 있으며 각각의 화소에 원근교정 정보(perspective corrective information)를 가지고 있으며 각각의 화소에 원근교정 정보 (perspective corrective information)를 가지고 있다. 4X4 매트릭스는 각각의 화소에 대한 원근 교정정보를 적용하여 회전, 확대, 천이, 업데이트 등을 하는 데 사용된다. 4X4 매트릭스는 여러 벡터에 이용된다.

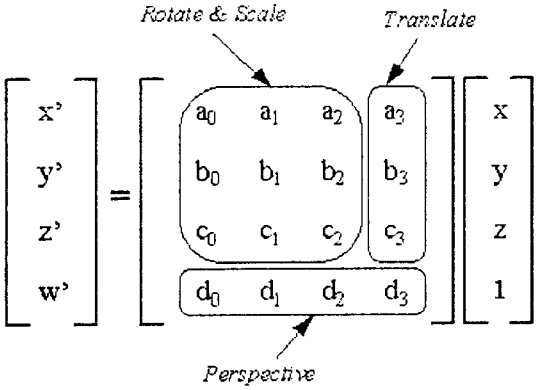
	MMX기술을 이용하지 않았을 경우의 명령의 수	MMX기술을 이용했을 경우 명령 수
로드	16	4
곱하기	8	2
쉬프트	8	2
더하기	7	1
기타	–	3
저장	1	1
합계	40	13

이미 16비트 정수 혹은 부동소수점 데이터를 사용하는 애플리케이션은 PMADD 명령을 충분히 이용할 수 있다. 매트릭스안에서 각각의 횡줄마다 하나의 PMADD명령을 사용할수 있으며 총 4개가 있다. MMX기술을 사용했을 때와 하지 않았을 때 명령어 수를 비교해 보면 이 작업은 다음과 같은 결과가 나온다.

MMX기술을 사용하면 MMX기술을 사용하지 않았을 때보다 절반의 명령으로 작업을 수행할 수 있다.

III. 요 약

MMX 기술은 멀티미디어 및 통신 애플리케이션



$X' = a_0x + a_1y + a_2z + a_3$

	MMX기술을 이용하지 않았을 경우의 명령의 수	MMX기술을 이용했을 경우 명령 수
로드	32	8
곱하기	16	24
더하기	12	2
기타	7	112
저장	4	34
합계	64	28

을 좀더 강력하게 실행할 수 있도록 해준다. MMX기술은 데이터를 병렬로 처리할 수 있도록 해주는 새로운 데이터 형태와 명령을 추가한 새로운 프로세서 아키텍춰이다. MMX기술은 기존 운영체계 및 애플리케이션과 완벽하게 호환된다.

MMX기술은 PC 플랫폼에 새로운 향상을 가져와 새로운 애플리케이션과 PC의 새로운 활용을 가능케 해준다. 또한 PC를 통신 및 멀티미디어 기기로 자리매김할 수 있도록 하는 업계의 새로운 파라다임을 설정할 것이다. MMX 기술을 채용한 프로세서가 장착된 시스템은 인텔이 MMX기술을 장착한 여러세대의 프로세서를 발표하는 1997년부터 대량으로 사용될 전망이다.

참 고 문 헌

[1] Intel Architecture MMXTM Technology Developers' Manual(Order Number 243013)

[2] Intel Architecture MMXTM Technology Programmers' Reference Manual(Order Number 243007)

저 자 소 개

金 京 曉

1959年 10月 25日生
1990年～1993年 서강대학교 경영대학원졸
1978年～1982年 한국항공대학 전자공학 졸

1988年 10月～현재 인텔코리아 재직
1982年 1月～1988年 10月 삼성전자 재직