[강좌] 게임에 영향을 주는 CPU. CPU구조 -4부.

게임에 영향을 주는 CPU. CPU구조 -4부.

PC에 자세히 알지 않아도, 모르는 사람은 없다고 생각되는 것이「Pentium 4 」라 생각 한다. 2006 년4 월 시점에서 가장 시장쉐어가 높고, 한편 가장 광범위하고 이용되고 있는 한편, 최근에는 여러 가지 제품들이 나오고 있다. 이번 페이지에서는 이 Pentium 4 CPU 에 대해서 생각해 보고 싶다.

Pentium 4.

Pentium 4를 설명하는데에 있어서 중요한 키워드가 되는 것은,「NetBurst (넷 버스트) 마이크로 아키텍쳐」이다.

자주 IT 계의 뉴스로 나오는「마이크로 아키텍쳐」(Micro Architecture)는 한마디로 하면「설계 밑거름」같은 것이다. CPU를 제조 하는데에 있어서는, 보다 큰 설계 개념으로「아키텍쳐」가 있어, 거기에 기초를 두어 하드웨어를 개발하기 위한 마이크로 아키텍쳐가 만들어진다(그림 1).

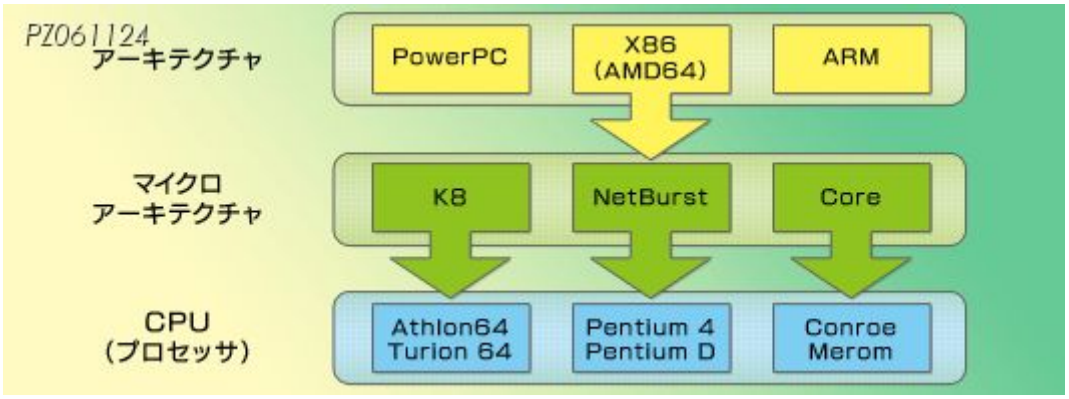


그림1

그럼, NetBurst라고 하는 마이크로 아키텍쳐에는 어떤 개념이 있는지 대답을 한다면,, 그것은 단순 명쾌하게「어떻게 동작 클럭을 올릴까」이다.

왜 동작 클럭 편중인가라고 하면, 이것에는 조금 옛날 이야기가 필요하게 된다. 20세기가 끝나려 하고 있던 1998~2000 년에, Intel 과 AMD는 격렬한 클럭 업 경쟁을 하고 있었다. 그리고 2000년 3월 6일, AMD 의 Athlon이 먼저 동작 클럭을 1GHz 로"대"돌파를 발표 하였다.그리고 2일 지나 Intel도 Pentium III/1GHz를 발표 했지만, 실제의 시장 투입도 Athlon/1GHz 쪽이 빨라서, Intel은 1GHz 도달 경쟁에서 완전하게 패배한 모습이 되었다.

물론, 그것이 이유가 전부는 아닐 것이지만, 지는 것을 싫어하는 Intel은, 동작 클럭 경쟁으로 절대로 지지 않기 때문에, 5GHz 근처의 동작 클럭을 실현하는 마이크로 아키텍쳐를 개발하려고 했을 것이다.그 결과적으로, NetBurst로 즉, 일반적으로 1stage로 조달할 수 있는 것을 4stage로 분할하는 등으로, 극단적인 super pipeline화를 실시했다. Intel은 이것을「하이퍼·파이프라인」이라고 부르고 있지만, 그 결과 태어난 것이 Pentium 4인 것이다.

Pentium III가 10 단(stage)이었던 파이프라인은, 초기의Pentium 4는 20 단 이였지만, Pentium 4는 페치(Fetch)와 디코드(Decode)의 파이프라인과 엑서큐트(Execute) 이후의 파이프라인이 분리하여, 엑서큐트만으로 20 단이 되어 버렸다. 파이프라인의 단수는 단순 계산으로Pentium III 의 배이상이 되었다고 말할 수 있다.

오랫만에 가정부로 예를 들면,「지금까지 가정부10 사람으로 작업하고 있었지만, 일의 스피드를10 배로 하기 위해, 가정부의 수를 100인으로 하고 싶다고 생각해, 가정부 100인를 수용할 수 있는 숙소의 건설 구상을 세웠다. 하지만, 갑자기100 인의 숙소라고 하는 것은 토지 면적적인 문제등 여러 가지 어려웠기 때문에, 건축 기술등이 머지않아 향상하는 것을 기대하면서, 일단20 인분의 숙소를 만들어 20 사람 고용했다」라고 하자.

그럼, 모처럼 20 사람을 고용했으므로, 달걀 후라이 하나를 만든다고 하면,

지금까지는 가정부 혼자서 만들고 있던 것을,「한 명은 냉장고로부터 달걀을 꺼내고, 한 명은 알을 꺼내고, 한 명은 프라이팬에 떨어뜨리고, 한 명은 구우고, 한 명은 접시에 담고, 한 명은 상을 차리는 일한다」라고 한 것 같은 분업제를 시행하는 것으로, 가정부1 인당의 작업량을 줄이고 고속화를 도모했다, 라고 하는 느낌이다.

단지 이 때, 아직 굽는 작업이 끝나지 않았는데, 다음의 알을 프라이팬 위에 떨어뜨릴 수는 없다.이와 같이, 가정부의 작업(=파이프라인1 단)에는,1 클럭이라고 하는 일정한 시간이 걸린다.

가정부 1 사람으로 달걀 후라이를1 개 만드는데, 5 클럭 걸렸다고 하자. 이 상태로 가정부를10 사람으로 늘린다(파이프라인을10 단). 그럼 10사람 각자가 작업에 1 클럭 필요로 하므로, 작업시간은10 클락이 되어 버린다.

한편, 1번에 100개의 달걀 후라이를 만드는 경우,1사람으로 작업하는 경우는 5×100=500 클럭 걸리는데 대해,10 사람으로 한다면, 최초의1 개가 나오기까지는10 클락 걸리지만, 그 이후는1 클락 마다 달걀 후라이가 테이블에 줄서기 때문에,109 클락으로 작업은 완료하게 된다. 이것이, 파이프라인을 늘리는 메리트다.

다만, 도중에 메뉴가 달걀 볶음으로 바뀌어 버리면, 또는 처음의 10 클럭을 기다려야만 한다. 라는 것은, 파이프라인을 늘리는 것에 대한 단점(「패널티」)이다.

이 쯤에서 제 1회의 복습이 되지만, 중요하기 때문에 재차 그림 2에서 나타내 보았다. 파이프라인 단수를 늘린 Pentium 4 에는, 이런 메리트와 단점이 있는 것을 이해해 두고 싶다.

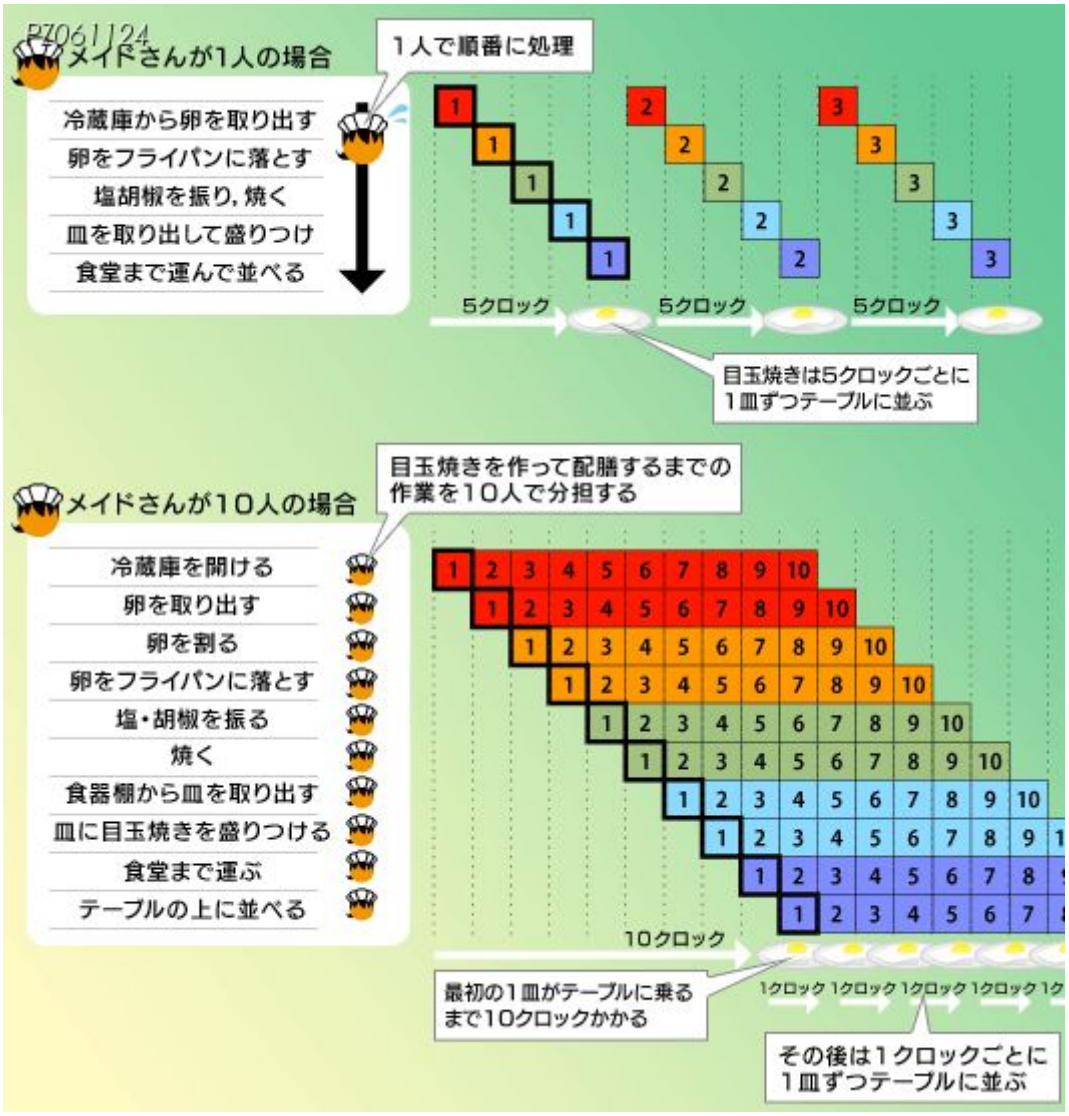


그림2

어쨌든 길다! Pentium4 의 파이프라인

방금전「페치와 디코드가 엑서큐트용의 파이프라인과 분리되었다」라고 썼지만, 이 부분에 대해, 좀 더 자세하게 설명하자. 먼저 페치와 디코드에 대해 설명한 것은 많이 전이 되어 버렸으므로, 잊고 있는 사람은 제1 회를 다시 읽으면 좋겠다.

그런데, 페치에 의해서 메모리로부터 읽어내지는 명령은, Pentium 4 에서「x86 명령」이라고 불린다.방금전의 그림1 에 나온 아키텍쳐가「x86 」이다.

이 x86 명령은 CPU가 실행하기 쉬운 방법으로 디코드되는 것이지만, 이 때 디코드된 것을「μOp 」(Micro Operation , 마이크로 오퍼레이션)이라고 한다. 그리고, 많은 CPU의 x86 명령을 그대로 실행하는 것이 아니라, μOp 를 엑서큐트(실행) 하게 되어 있다.

μOp의 실행에 임하고, 극단적인 super pipeline화가 되고 있는 Pentium 4은 아무래도(이것 또 제1 회 에서 설명함) 분기 미스가 항상 따라다닌다. 거기서, 분기 미스에 의해서 파이프라인 하자드가 발생해, 파이프라인의 흐름이 정지(이것을「파이프라인 스톨」이라고 한다)했을 때에, 곧 또 다시 파이프라인에 흘리기 위해 Pentium 4 는「트레이스 캐쉬」(Trace Cache ,「실행 트레이스 캐쉬」라고도 말한다)라고 하는 구조를 L1 캐쉬 안에, 일반적인 데이터 캐쉬와는 별도로 준비해 있다.

트레이스 캐쉬의 사이즈는 12K로,μOp을 보존하기 위한 메모리 영역인 것이다.

트레이스 캐쉬에 μOp를 놓여져 있으면, 같은 명령을 매번 매번 페치 해 디코드할 필요가 없어지기 때문에 CPU의 처리 효율은 오른다. 디코드라고 하는 것은 꽤 부하의 높은 처리이므로, 디코드 끝난μOps를 트레이스 캐쉬로부터 가져와진다는 것은, 그만한 의미가 있다.

트레이스 캐쉬를 채용한 것에 의해, 엑서큐트용 파이프라인 안에서「μOp를 페치 한다」는 단(stage)이 설치되고 있는 것도, NetBurst 마이크로 아키텍쳐의 특징이다. 분기 예측을 실시해 명령을 페치 하는 기구를, 처음과 엑서큐트용 파이프라인안에 2개소 준비하는 것으로, 분기 예측 정도의 향상을 도모하고 있는 것으로 있다. 덧붙여서 이 기구는,「BTB」(Branch Target Buffer , 분기 예측 버퍼)라고 하는, 분기 예측의 정보를 격납해 두는 스페이스와 분기 예측 알고리즘에 의해서 성립되고 있다.

좀 더 구체적으로 설명하자. 그림 3은 초기 Pentium 4 의 파이프라인 구조를 나타낸 것이다. Pentium 4는 트레이스 캐쉬로부터 μOp을 페치 하고 파이프라인에 흘리는 처리를 실시한다. 이것을「μOp 큐잉」(μOp Queuing)(이)라고 하지만, 통과하기 때문에 여기는4 단으로 구성되어 있다.

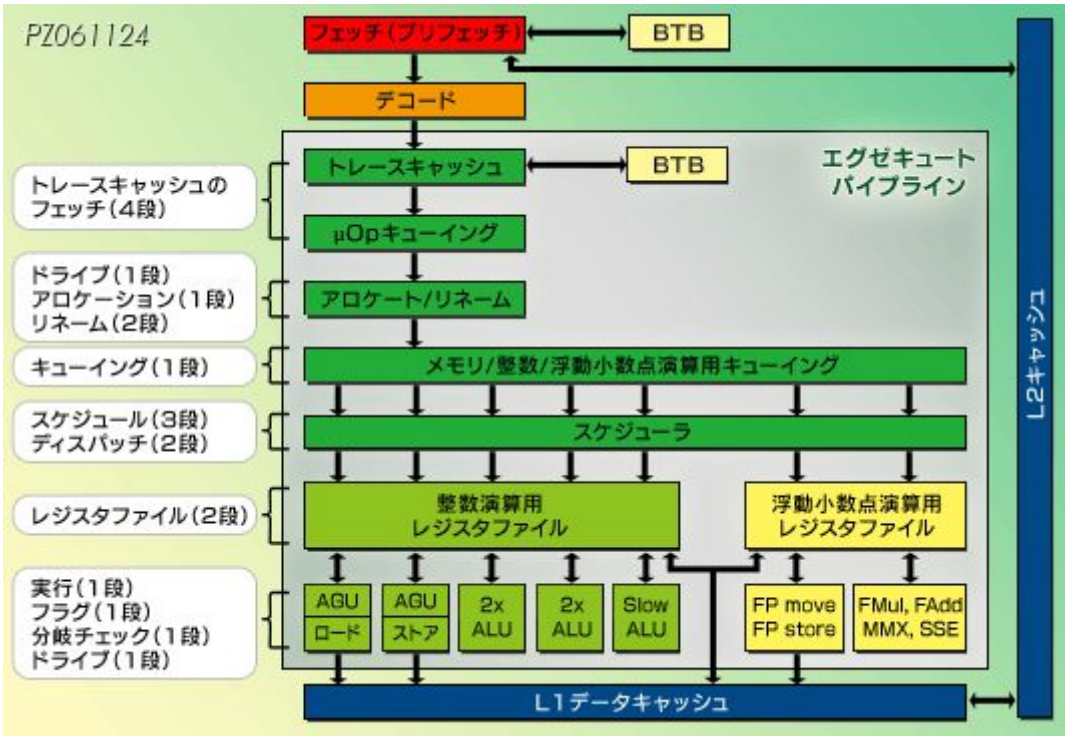


그림 3

L1 캐쉬로부터 다음의 명령을 꺼낸다는 것은, 그만큼 어려운 처리는 아니다. 더 말하면, 여기에 4단을 할애하는 마이크로 아키텍쳐라고 하는 것은, NetBurst 외에는 들었던 적이 없다. 단수가 너무 많다.

반대로 말하면, 여기까지 파이프라인의 단수를 늘리고 있기 때문에, 예를 들면 같은 0.13μ프로세스로 비교했을 때, AMD의 Athlon XP는 2.2GHz까지 였을 무렵, Pentium 4는 3.40GHz 까지 동작 클럭을 올릴 수 있었다.

그림3 를 돌아보자.「allocate/ rename」로부터 「스케쥴러」까지는, μOp를 슈퍼 스칼라 처리하기 위한 전단계 정도로 이해하여도 좋다. 병행해 실행할 수 있는 처리를 동시에 실시하게 하는 것이 슈퍼 스칼라이다는 것은, 제1 회에서 설명했던 대로이지만, 여기는 μOp 큐잉, 슈퍼 스칼라 처리를 위한 대기열을 유지하고(스케쥴러), 디스팟치를 실시해, 실제로 슈퍼 스칼라 처리를 실시한다, 라고 하는 흐름이 된다.

포인트는, 스케쥴러의 스테이지에서만 파이프라인이 5단이 있다는 것이다.「레지스터 파일」은 제2 회에 설명한 레지스터로, 여기는 그렇게 깊게 생각하지 않아도 좋다, 간단히 설명하자면, μOp를 실행하는 전단계에서, 일시적으로 데이터를 두거나 하는 정도에 파악해 두면 OK이다. 단지, 여기에도 2단이 준비되어 것은, 전연 생각할 수 없는 사항이기도 하다.

이런 식으로 파이프라인의 단수를 늘려 간 결과, Pentium 4의 효율=IPC를, Pentium III와 비교해서 최대 30 % 떨어지고 있다고 한다. 그렇다면, IPC가 Pentium III의 70 % 정도로, 여기에서 클럭을 배로 동작 시키면 70 ×2 =140 % 라고 하는 계산이 성립되므로, 동작 클럭이 오르면 자연히 해소되는 레벨의 문제였다. (적어도 NetBurst 마이크로 아키텍쳐가 발표된 시점에서는) Intel에서는 그만큼 문제는 아니었던 것이다.

정수 연산과 부동 소수점 연산의 차이

다시 그림 3을 보면, 레지스터 파일아래에 있는 것이, 엑서큐트의"본체"가 되는 실행 유닛이다. 실행 유닛은「정수 연산」「부동 소수점 연산」으로 크게 2개 로 나뉘고 있지만, 우선은 정수 연산의 실행 유닛에 대해 보기로 하자.

그림 4

●프로그램(=게임)의 제어를 실시하는 정수 연산 유닛

정수 연산이란, 문자 그대로 정수의 연산을 하는 것이다. 그야말로「1 +1 」,「2 ×3 」등의 계산을 실행한다. 정수 연산은 주로 프로그램의 제어에 이용되고 있어,「몬스터는 ○몸이 있다」,「앞으로 ○발 먹으면 죽는다」,「지금○턴째」라고 하는 것이 주된 담당이다. 요컨데, 정수 연산은 게임의 전체적인 퍼포먼스와 관계되지만, 전체적에서 정수 연산 성능이 좋다고 하여도, 체감 하는 것은 꽤 어렵다.

정수 연산 유닛(그림 4 , 그림 2에서 일부 발췌) 가운데,「ALU는「Alithmetic Logic Unit 」의 약어로,「논리 연산 장치」이다. 그리고 ALU는 정수 연산을 실행하는 유닛이라고 하고 있다. 덧붙여서,「AGU 」(Address Generation Unit , 주소 생성 장치)는, 데이터의 로드(Load)/ 스토어(Store , 세이브의 뜻)이 올바르게 실시할 수 있도록 돌보는 것으로, 이것도 꽤 중요한 유닛이라고 할 수 있다.

Pentium 4에서, 단순 명령(Simple Instruction)은 2 배속 ALU (2x ALU)가 2조, 복잡한 명령(Complex Instruction)은 등속 ALU (Slow ALU)가 1조로 되어 있다. AGU는 로드와 세이브가 각1 조가 되고 있다. 단순 명령이라든지 복잡한 명령이라든지 여러 가지 있지만, 뭐, 여기는 ALU 자리가 3조, AGU 로드와 세이브용으로 1 쌍씩 있다고 이해해 주면 문제 없다.

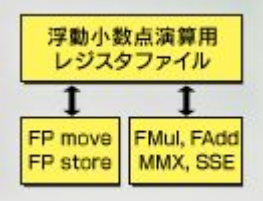


그림 5

●게임의 리얼리티와 관계되는 부동 소수점 연산 유닛

계속하여, 부동 소수점 연산 유닛(「Floating point number Processing Unit」,FPU)은, 우선 부동 소수점 연산에 대해 말하면,「소수점이 이동하는 소수의 연산」을 한다. 예로 .123 라면 1.23 ×10의 2 승, 4567 그렇다면 4.567×10의 3승으로서 표현하는 것이 부동 소수점이다. 여기를 설명하면 꽤 난해하게 되므로,「이러한 편이 컴퓨터로부터 취급하기 쉬우니까」라고 이해해 두면 좋겠다.

부동 소수점 연산의 특징은, 표현력에 있다.
　예를 들면, 밝기의 표현에 대해서 생각해 보자. 100이라고 하는 밝기가 있었다고 해서,「100 보다 조금 밝다」를 표현하려고 했을 때, 정수 연산이라면 이것은101 (이)가 되지만, 부동 소수점이라면100.1 (1.001 ×10의 3승)과, 100.01 (1.0001 ×10의 3승)이나, 100.00000001 (1.0000000001 ×10의 3승)등으로 표현할 수 있다. 밝기의 그라데이션과 같은 것을 표현하고 싶었을 때 , 어느 쪽이 보다 매끄럽게 실시할 수 있을까는, 말할 필요도 없다고 생각한다.

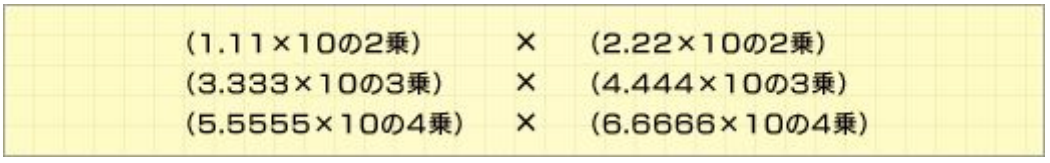
그리고, 이 표현력은 게임의 리얼리티로 연결된다. 부동 소수점 연산 성능이 오르면, 게임의 속도를 유지한 채로, 화면의 리얼리티를 향상 할 수 있다. 반대로, 같은 화면의 리얼리티를 유지한다면, 게임의 퍼포먼스를 향상할 수 있게 된다.

그림 5는 그림 2의 일부를 발췌한 것이다.「FP move」는 Floating Point number (부동 소수점수(실수))의 소수점수의 이동을 실행하는 부분이고,「FP store」는 스토어(세이브)를 실시한다. 그리고「FMul 」「FAdd 」가 각각 부동 소수점의 가산, 곱셈을 실행하게 되어 있다.

　단지, 이들 부동 소수점 연산 유닛은, 과거의 x86 아키텍쳐의 CPU 와 호환성을 가질 필요가 있기 위해, 성능은 그만큼 떨어졌다. 한편 Intel은, 표현력의 높음으로 연결되는 부동 소수점 연산 성능을, 압도적으로 향상시키고 싶다고 생각하여. Intel은 기존의 부동 소수점 연산 유닛은 호환성 유지를 위해 그대로 하면서, 과거의 CPU와 호환성을 잘라 버린 고속의 부동 소수점 연산 회로를 도입했다. 이것이「SSE 」(Streaming SIMD Extentions)이다.

SSE을 풀어 쓰면「인터넷 또는 스트리밍SIMD 확장 명령」이다.
※ 그럼「SIMD 라는 뭐지?」라고 하는 이야기가 된다. 이것은「Single Instruction / Multiple Data 」의 약어로, 하나의 명령(Single Instruction)으로 복수의 데이터(Multiple Data)을 취급하는 방식을 가리킨다.

예를 들면, 이하와 같은 명령이 있었다고 하자.



일반적인 연산에서는, 이것들은 위로부터 차례로 처리되어 가는 것이지만, 이것이 SIMD의해 연산된다면, 아래와 같이 된다.

요컨데, SIMD 연산에서는「같은 명령(여기에서는 곱셈)으로 처리할 수 있는 것은, 정리해 읽어내고, 정리해 처리한다」같게 했다. 그리고, SIMD 연산을 이용하고 고속화를 도모한 것이 SSE이다.
　SSE는 Pentium III에서 채용된 확장 명령이지만, Pentium 4에서는 새롭게「SSE2 」(스트리밍SIMD 확장 명령 2)가 추가되었다. 이것은 간단하게 말하면, SIMD 연산에 있어서의「(1.11 ×10 의2 승)×(2.22 ×10 의2 승)」이라고 한, 연산 하나 하나의 고속화를 도모한 것이다.

부동 소수점 연산은, 정수 연산과 비교하면 CPU 부하가 많이 걸려서(=데이터를 읽어내고 있는 시간과 비교해서, 연산하고 있는 시간이 상대적으로 길다), 순수하게 CPU의 연산 성능이 퍼포먼스에 영향을 주기 쉽다. 더 말하면, 동일한 파이프라인이라면, 동작 클럭의 높은 편이 성능은 좋아진다. 그러한 의미로,SSE 나 SSE2는 NetBurst 아키텍쳐와 궁합의 좋은 명령이라고 할 수 있을 것이다.

그러나, SSE와 SSE2 에는 하나 큰 약점이 있다. 그것은 SSE와 SSE2는 과거의 CPU와 호환성이 없는 신명령이므로, 프로그램측으로부터 이용할 수 있도록 최적화되어 있지 않으면, 이용할 수 없다는 점이다. 또한 게임에서도 SSE/SSE2 에 지원하는 타이틀은 그만큼 많지 않다. 그래서 게임에서 ,Pentium 4가 최고 성능을 발휘할 수 없는 경우가 많은 것이다.

이것은 왜냐하면, 게임의 경우 SSE와 SSE2에 최적화하려고 하면, 최적화해야 할 포인트가 매우 많아져 버리기 때문에, 예를 들면 MPEG-2등의 비디오 encode 소프트는, 1개의 처리만 필요로 어플리케이션만 필요하므로, 최적화해야 할 포인트가 한정되어 있다. 따라서, 거기를 SSE와 SSE2에 최적화하면, 처리를 고속화할 수 있다. 반면 게임은, 전장의 씬이 있다면, 빛의 표현과 같은 특정의1개만 SSE나 SSE2에 최적화했다고 하자. 그러나, 이 때 폭발시의 연기도 동시에 최적화되어 있지 않으면, 씬 전체적으로의 퍼포먼스 향상에는 기여하지 않는다. 즉, 1개의 처리만 있는 어플리케이션과 비교해서, 최적화의 메리트가 보이기 어려운 것이다. 물론, 모두 최적화하면 효과는 있지만, 그런 일을 하고 있으면 언제까지 지나도 게임은 발매되지 않는다. 그러니까, 별로 최적화가 진행되지 않은 것이다.

파이프라인의 효율을「어떻게든 한다」유익하게.

Hyper-Threading 테크놀러지

정수, 혹은 부동 소수점의 실행 유닛은 엑서큐트의 파이프라인에 포함되어 있으므로, 파이프라인 스톨의 영향을 받는다. 그 말은, 이것도 오덕 파이프라인의 효율은 별로 좋지 않은 것이다. 그래서 그것을 해결할 수 있도록 도입된 것이「Hyper-Threading 테크놀러지」이다.

HT 테크놀러지에 대해서는 제2 장에서 간단하게 접했지만, 우선 간단하게 정의와 같은 것을 나타내면,「실행 유닛이 비어 있는 부분을 사용하고, 사용하고 있는 쓰레드와는 다른 쓰레드를 사용하는 기술」이다.

예로서 주인의 명령을 받은 집사가「오늘의 저녁 식사는 중화」라고 결정했다는 상황을 가정해 보자.요리 담당의 가정부가 10 사람 있고, 중화 요리가 자신있는 것은 그 중 3 사람이라 하자. 이 때, 남는 7 사람은 아무것도 하는 것이 없기 때문에, 게으른 잠을 탐내게 된다.

단지, 중국요리에 사용하는 냄비만 바빠서 사용할 수 없지만, 오븐은 비어 있다. 즉, 오븐을 사용하여, 식후의 디저트용으로 케이크를 만들려고 하면 만들 수 있다(그림 6).

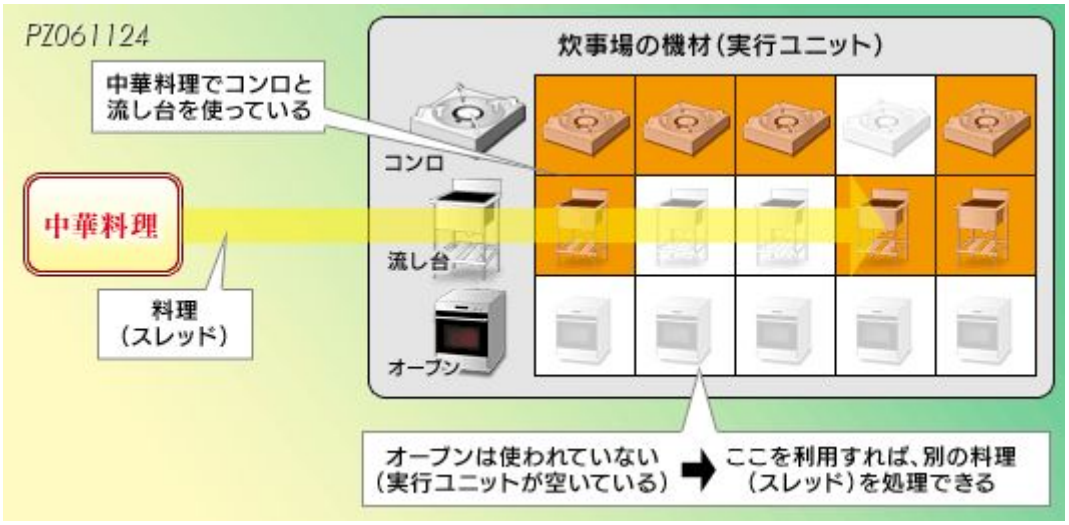


그림 6

거기서, 양식 담당의 가정부인 2 사람을 흔들어 깨우고, 케이크를 만들게 한다. 물론, 그렇게 결정했더니 일식 담당의5 사람은 자고 있는 것이지만, 양식 담당까지 자고 있는 것보다는, 취사장을 효율적으로 이용할 수 있는 것이다.그래서 HT 테크놀러지라고 하는 것은, 취사장(=실행 유닛)을 효율적으로 이용하기 위해, 자고 있는 가정부를 두드려 일으켜 일하게 하는(=실행 유닛의 빈 곳에 스렛드를 사용하는) 기술인 것이다.

이 때, 밖에서 보면 취사장에서는 가정부가 요리하고 있을 뿐이지만, 실제 작업으로서는 2 종류의 요리를 동시에 만들고 있다. 이와 같이HT 테크놀러지는, CPU는 1개 이면서, 2개의 CPU 인 것 같이 동시에 복수의 스렛드를 처리시킬 수 있는 것이다(그림 7). 반대로 말하면, 주인이「중화식 라면 과, 디저트는 살구씨 두부로 해라」라는 명령을 할 경우에는, 양식 담당의 가정부는 하는 것이 없어진다. 즉, multi-thread에 대응한 명령이 아니면, HT 테크놀러지의 효과는 얻을 수 없기 때문에, 이 점은 기억해 두자.

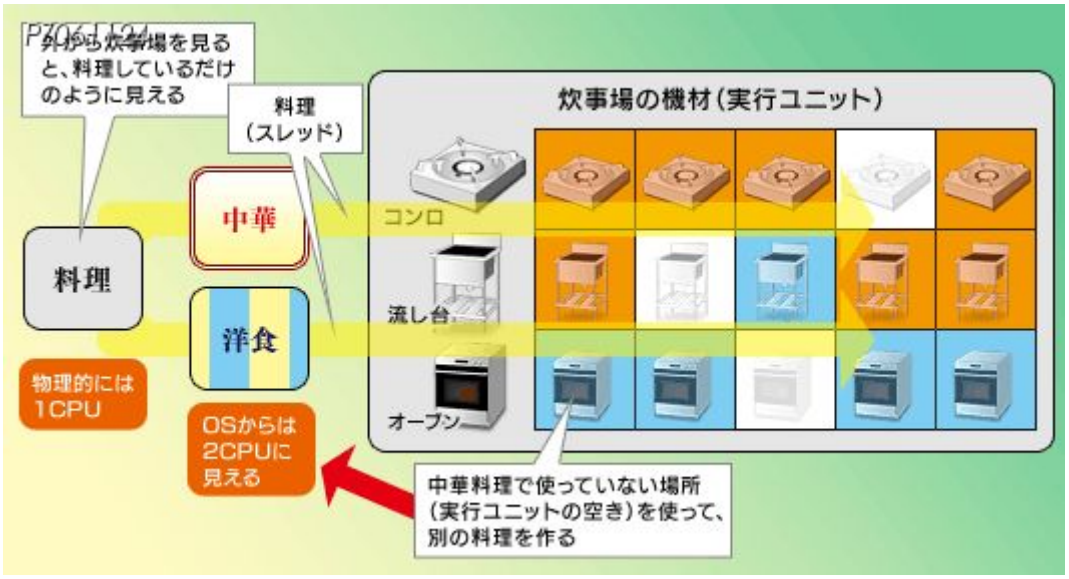


그림 7

파이프라인을31 단으로 하여, 고클럭화로 돌진했다 Pentium 4

여기까지 Pentium 4의 기초 지식이지만, Pentium 4는 최초부터 HT 테크놀러지를 채용하고 있던 것은 아니였다. 또한, 파이프라인 단수에도 변화가 있었다. 거기서, 우선은 계통를 그림 8로 정리해 보았다.

황색 화살표는, 프로세스 룰을 포함한 큰 변혁, 백색의 화살표는 섬세한 업데이트, 흑색의 화살표는 파생한 한정 모델을 나타낸다. 같은 배경색 모델은, 같은「CPU 코어 세대」이다.

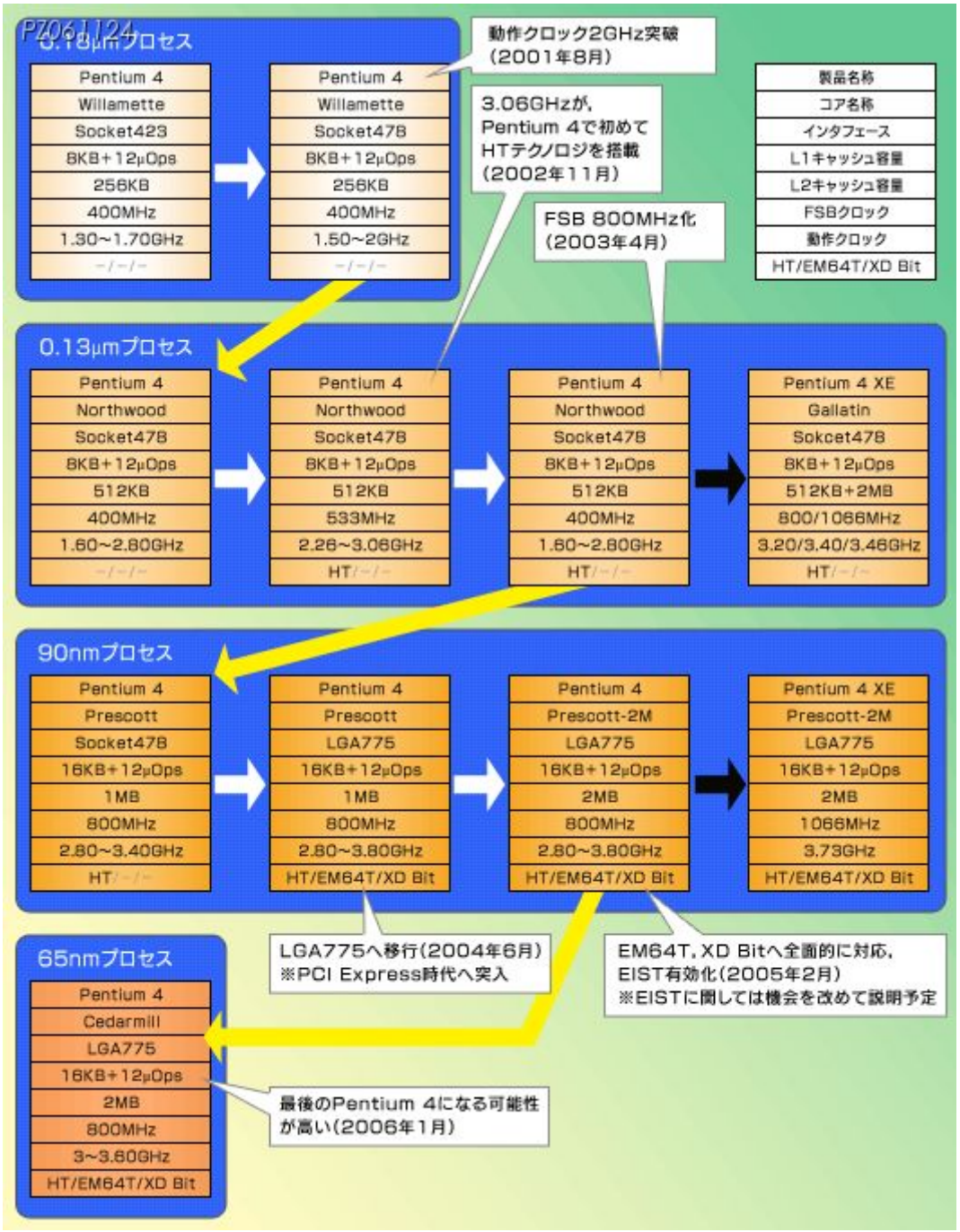


그림 8

최초의 Pentium 4는, 0.18μm 프로세스로 제조되었고, 개발 코드네임은「Willamette 」이다.「개발 코드네임 =CPU 코어」라고 하는 관점으로부터「Willamette 코어판 Pentium 4 」등이라고도 불린다.처음은 1.40/1.50GHz 만이었지만, 후에1.70GHz ,1.30GHz 의 순서에 추가되어 이것이 Pentium 4 의 기초가 되었다.

초기의CPU 소켓은 423 핀의「Socket423」이였지만, 곧바로「Socket478 」라고 불리는 소형의 소켓에 이행해, 동작 주파수는 2GHz를 돌파한다.

이것에 계속 되는 제품이, 0.13μ프로세스로 제조된 개발 코드네임「Northwood 」이다. Northwood 코어는 1.60 ~3.40GHz와 폭넓게 전개되어 전개 중에서「FSB 」(Front Side Bus)(으)로 불리는 시스템 버스 속도도 자꾸자꾸 고속화하며 갔다. Pentium 4에서, FSB는 CPU와 칩셋(노스 브릿지)를 연결하는 버스로, Willamette 시대에는 400MHz 이였다. FSB 클럭은 533MHz을 거쳐 800MHz으로 Northwood 코어는 2 배가 되고 있다.

좀 더 세세하게 보자.Northwood 코어에서는 FSB 클럭을 533MHz로 끌어올렸을 때에, 최상위가 되는 것이 3.06GHz이다. 그리고 방금전 설명했던 HT 테크놀러지가 처음으로 지원 되었다. 그 후「AMD는 Athlon 64를 투입하였다」라고 하는 이야기가 전개되어, AMD에 지고 싶지 않은 Intel은 선제 공격으로 FSB 클럭을

800MHz로 끌어올리고 HT 테크놀러지를 모든 모델에서 지원하게 하였고, 한층 더 동작 클럭도 3.40GHz 까지 끌어올렸다.

게다가 AMD는 Athlon 64의 상위 모델로서 Athlon 64 FX인 특별판을 투입했다. Intel은 이것에 대항하여, Northwood 코어를 베이스로 한, L3 캐쉬를 2MB 내장한(본래는 서버/ 워크스테이션용이다 Xeon MP) 「Gallatin 」라고 하는 코어를 서둘러 출시하고, "신제품"Pentium 4 Extreme Edition (Pentium 4 XE)를 발매했다.

그 후,Intel은 90nm (0.09 μm) 프로세스의 「Prescott 」코어로, 제품 라인을 옮겨놓아 간다. Prescott 코어는 우선 Socket478로 등장했지만, 그 후 「LGA775 」라고 하는 새로운 CPU 소켓에 이행 해, 동시에 「프로세서 넘버」를 3 자리로 바꾸었다.「Pentium 4 5xx 」,「Pentium 4 6xx 」라고 하는 표기에는, 본 기억이 있다고 생각한다. 덧붙여 3자리 숫자가 가지는 의미에 대해서는 다음 번 설명하고 싶다.

덧붙여서, Prescott은 파이프라인 단수가 Northwood까지의 20 단으로부터, 31 단으로 증가했다.이 외 ,Willamette/Northwood (와)과 비교했을 때, L1의 데이터 캐쉬 용량이 지금까지의 8KB에서 16KB으로 늘어 났고, L2 캐쉬 용량은 512KB에서 1MB로 확장되었다. 장래적으로 동작 클락이 올라 가면, 메인 메모리와 속도의 차이가 커지고, 메인 메모리에 데이터를 읽으러 가게 되었을 때의 패널티가 커진다(=CPU (으)로부터 보았을 때의 대기 시간이 상대적으로 길어진다) (뜻)이유이지만, 캐쉬 용량의 증가는, 이 문제에의 배려라고 봐야 할 것이다.물론, 여기에는 기술적인 진보도 있다.

이런 세부적인 확장은 「20 단에서 31 단으로 한층 더 파이프라인을 세분화해 고속 동작시키도록 한 반면, 파이프라인 스톨의 패널티가 커지기 때문에, 분기 예측을 강화했다」라고 한 느낌이다.

Prescott 의 확장된 기능과 최후의 Cedarmill

한층 더 Prescott을 새롭게 한것은 「SSE3 」라고 불리는 명령군의 추가이며, 「EM64T 」을 대응하는 「XD Bit 」의 탑재를 하고 있다.

SSE3는 SSE2를 한층 더 확장한 것으로 이해해도 문제 없다. EM64T는 Extended Memory 64 Technology이지만, 이것은 한마디로 하면 x86 명령의 64bit 확장판 「AMD64 」의 호환 명령이다. AMD64 에 관해서는, Athlon 64에 대한 글을 쓸 때 설명하고 싶다. 현재는, EM64T를 지원하는 Pentium 4이라면 「Windows XP Professional 64-bit Edition」과, 차기 Windows「Windows Vista 」의 64bit에 지원하는 정도로 인식하면 된다.

또한 XD Bit는 eXecute Disable Bit로, 이것을 이해 하는데 2개 정도의 예비 지식이 필요하게 된다. 하나는「메모리에는, 프로그램을 저장하는 곳과 데이터를 저장하는 곳의2 종류로 나눌 수 있다. 프로그램용은 한 번 쓰면 읽어내기 전용으로 상관없지만, 데이터용의 곳은 읽고 쓰기가 발생한다」라고 하는 것.다른 하나는 「프로그램의 오동작을 일으키는 더미 데이터와 부정(악성) 프로그램이 나타나 있다」라고 하는 것이다.

XD Bit는, 이러한 「메모리상에 있는, 데이터 용도의 장소에 있는 데이터 모든 것」에 대해서, 문자 대로 엑서큐트(실행)를 Disable로 하는 기능이다. 이것에 의해, 이 손의 부정 프로그램은 완전하게 블록 할 수 있다.

덧붙여 XD Bit는 AMD의 Athlon 64에서 탑재하고 있는 「NX Bit 」(No eXecution Bit)와 완전히 같은 것이다. 이들은 모든 Windows XP Service Pack 2에서 「DEP」(Data Execution Protection) 기능과 연동해 동작한다.

Windows와 연동해 움직이기 때문에, 꽤 안전한 기능이라고 할 수 있다. 게이머로써는, 하나만 주의할 점이

있다. 그것은, 게임의 카피 가이드 소프트(또는 암호화 소프트)다. 카피 가이드 소프트 등은, 프로그램의 내용을 들여다 보는 여러 가지 부적당한 것이므로, 메모리의 데이터용도의 장소에서, "자기 자신"을 고쳐 쓰면서 실행하게 되어 있어서, XD Bit를 활성화 해 두면, 게임이 기동하지 않게 되어 버리는 경우가 있다.

기능 이외에 관심을 가지면, Prescott 코어에서는 지금까지의 Pentium 4들이 가지고 있던 접촉용 핀을 가지지 않는「LGA775 」로 구성되어 있다. CPU 소켓에 바뀌게 된 것도 토픽이다. 또한, Prescott 코어와 사양은 거의 같으면서, 제조 프로세스가65nm (으)로 진화한「Cedarmill」라고 하는 코어가, 2006년1 월에 출시되고 있다.

그러나 Cedahmill 코어가 Pentium 4라 하지만, 4GHz 나 5GHz 를 넘지는 않았다. Northwood가 3.40GHz 를 달성한 Pentium 4 이지만, Prescott는 3.80GHz까지 밖에 동작 클럭을 향상 시키지 못하였다. 그리고 Cedarmill도 같은 클럭인 3.80GHz 그대로다. 그리고, Cedarmill은 마지막 Pentium 4 용 코어가 되어, NetBurst 마이크로아키텍쳐의 CPU에서 최후가 되는 CPU로 주목받고 있다.

출처 : 4gamers 작가:大原雄介 편집: felfin

원본출처 : 4gamer.net