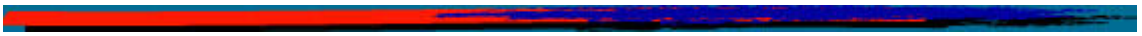
[강좌] 가정부로 이해한다. CPU 구조 -1부

가정부로 이해한다. CPU 구조 -1부.



CPU란.
CPU란? Central Processing Unit (중앙 연산처리 장치)의 약어로, 각종 계산이나 디바이스들의 제어를 하는 장치이다. 원래CPU란 무엇인가. 이것을 어려운 개념으로 설명한다면, 예를 들면 바벳지의 계산기계(계차기관)까지 거슬러 올라 간다면, 아무도 읽을 생각을 하지 않을 것이며, 불리언 대수의 이야기를 시작해도 역시 읽혀지지 않게되는 것은 명백하다. 또한 "전기의 기초로부터" 라고 하면서, 반도체에 대한 해설이나 진공관의 동작 원리로부터 시작을 하면, 반드시 편집부로부터 분노의 철퇴가 내려져 버릴 것이다. 그래서, 이러한 옛 이야기나 기초의 기초에 대해서는 논하지 않고, 게임과CPU의 서로 관련에 임해서 이야기를 해 나가고 싶다.

PC 의 경우, 아무리 고성능의 그래픽 카드가 있더라도, 그 자체가(그래픽카드가) 마음대로 표현하거나 실행하거나 하지 않는다. 또한, 아무리 고성능의HDD가 있더라도, 마음대로 읽고 쓰기를 하지 않는다. 이러한「디바이스」(주변기기)로 불리는 것들은, 모두 명령이 오는 것을 기다리고 있으며, 명령이 오면 거기에 따라서 동작을 한다. 하지만 유일하게 CPU는 이러한 디바이스들에게 명령을 내리는 입장에 있다. 그래서 CPU는 PC의 중심점 이라던지, 중추라든지 등으로 말해지고 있는 것이다.
이렇게 되면, CPU가 고속으로 처리할 수 있으면 할 수 있을수록, PC 전체는 속도는 빨라지게 된다. PC 전체를 하나의 대저택이라고 보았을 경우, CPU는 그 저택의 관리를 맡은 집사라고 생각하면 된다. 그리고, 그 아래에는 가정부라든지 정원사라든지 요리사라든지, 하는 여러 다양한 사람들이 각 디바이스에 상응한다고 생각하면 좋겠다.

그런데, 집사의 능력인 CPU의 능력을 말할 때 부수물인 것이「동작 클럭」이다. 동작클럭이 무엇인지 이 정체에 대해 다루어 보자.　동작 클락: 정확하게 말하면「동작 주파수」이지만, 편히 동작 클럭이라고 부르자. 이 동작클럭을 간단하게 말하면「어느 정도의 스피드로 명령을 처리할 수 있을까」(을)를 나타내는 수치로, 통상「Hz (헤르쯔)」라고 하는 단위를 사용해서 나타난다. 이 Hz는 라디오의 주파수라든지, 전원 콘센트(AC100V)등 여러 곳에서 말할 때 사용하는, 그 Hz 이다.

예를 들면「50Hz 」라고 하면「매초50 회」의 의미를 갖는다.「동작 클럭 50Hz 의CPU 」라면, 매초 50 회의 명령 처리를 실시하는 CPU 를 뜻하게 된다.　CPU 의 경우, 외부로부터 동작 속도를 결정하기 위한 신호를 넣어 줄 필요가 있다. 그래서 이 신호는 영어로「Clock Signal 」혹은「Clock Source 」라고 표기하고, "클럭"이라고도 한마디로 끝내는 경우도 많이 있다. 본고에서는 이하「입력 클럭」(FSB,HTT)이라고 한다.

다만, 이 입력 클럭은 반드시 동작 클락과 이콜은 아니다. Intel 과AMD 등의 메이커 CPU를 생각하면, (입력 클럭 : 동작 클럭)=(1　: 1)였던 예는,「Intel 486DX/50 」라고 하는, 상당히 예전의 CPU로 50MHz 까지 거슬러 올라가게 된다. Intel 486DX/50에서 50MHz(M은 100 만을 나타내는 값이므로, 5000만Hz가 된다.)의 입력 클럭은, 50MHz의 동작 클럭하고 같이 동작한다. 그러나, 이 이후는(입력 클럭 : 동작 클럭)=(1　: 1 이상)이라고 하는 관계가 되어, 예를 들면 2005 년10월에, 가장 동작 클럭이 높은 Intel의 CPU Pentium 4 670/3.80GHz 의 경우, 입력 클럭은200MHz , 동작 클럭은3.80GHz (G는10 억을 나타내는 값이며, 1GHz 는 10 억Hz의 의미)이다. 그리고 동작 클럭이 3.80GHz 라고 하는 것은, 매초38 억회의 속도로 처리를 하고 있는 것이 된다. 입력 클럭인 200MHz의 19 배가 되어 3800MHz =3.80GHz 동작하고 있는 것이다.

CPU 의 기본 구조와 파이프라인.

갑자기 동작 클럭의 이야기를 해서, 동작 클럭이 높으면 좋은 것인가 라고 생각할지도 모르지만, 이야기는 그렇게 단순하지 않다.

그림 1은, CPU의 구조를 간단히 단순화 한 것이다. 크게 나누면, (Fetch), (Decode), (Execute), (Write Back)로 4 분류로 나뉜다. 각각의 기능은 대략적으로,,,

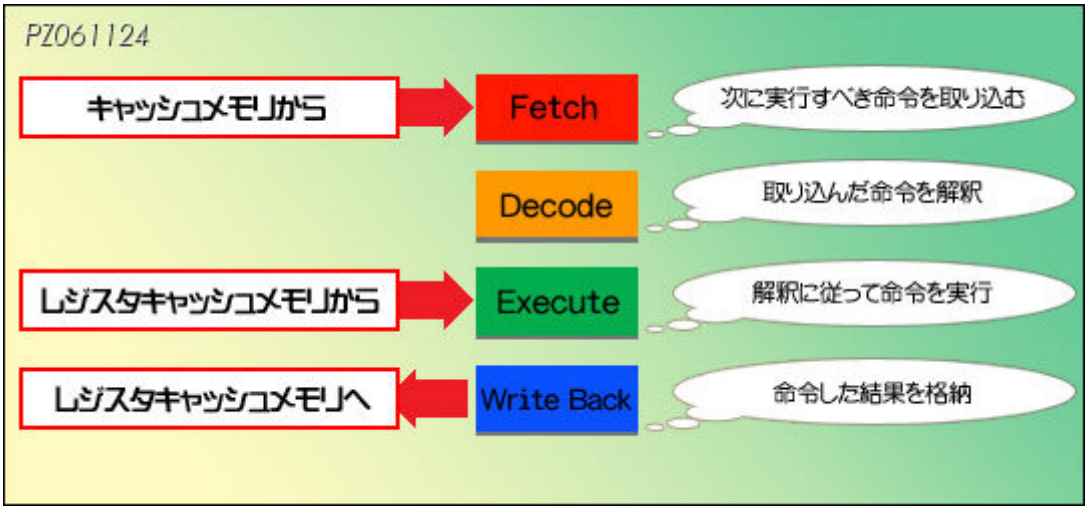


그림 1

페치(Fetch) : 메모리 혹은 캐쉬로부터, 다음에 처리해야 할 명령을 가져온다. 대저택의 예로 말하면, 집사가 예정표를 보고, 다음에 실시해야 할 처리를 확인하는 것이다.
디코드(Decode) : Fetch 한 명령을 보고, 실제로 처리해야 할 내용을 확정시킨다.「잔디 손질」이라면, 명령할 수 있도록 정원사를 부르든지, 원래 정원사를 부를 수 있도록 누군가 쉬고 있는 예로 가정부를 찾든지, 하는 순서다.

엑서큐트(Execute) : 확정한 명령을 실시하는 곳이 여기다. 정원사에「잔디의 손질을 해 주세요」라고 명령하는 것이 이 단계이다. 한마디로 하면, CPU 의 처리라고 하는 것은「데이터의 가공※1 」이므로, 처리전에는 가공전의 데이터를 읽어들여, 이 단계에서 데이터를 레지스터,캐쉬,혹은 메모리(모두 제2 해설)로부터 읽어내는 처리가 동시에 행해진다.
write back(Write Back) : 처리한 결과를 반영시키는 곳이다. 예를 들면,「1 ＋1 =」(이)라고 하는 계산 문제가 있다고 하면, 여기까지가 Execute 이고, 계산 문제에 2라는 답을 보여주는 작업을 실시하는 것이 write back다.

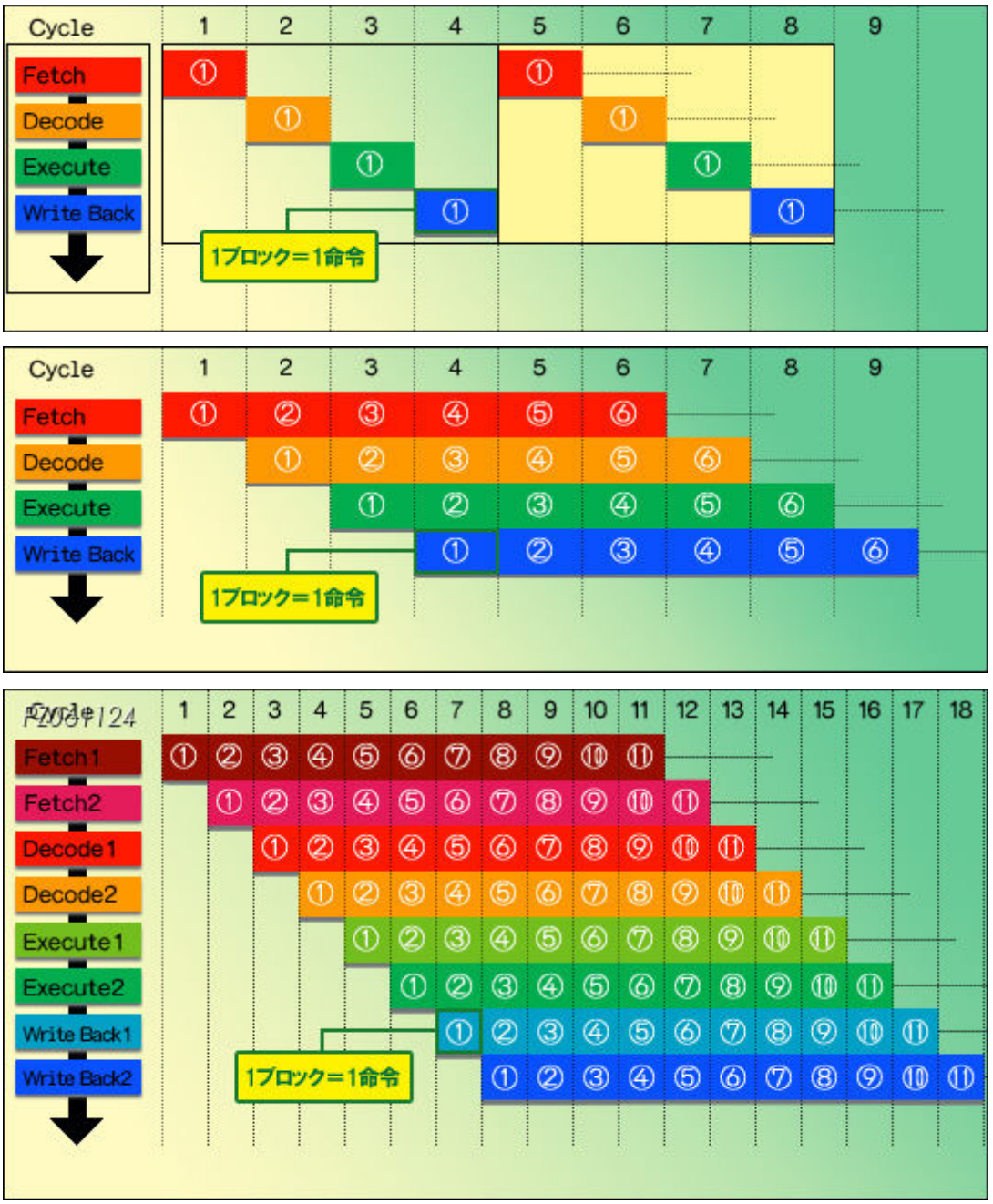
※1「가공」이라고 하는 표현을 이해하기 어렵다고 느낄지도 모르지만, CPU의 처리는 기본적으로「명령에 따라서, 특정의 데이터를 조작한다」라고 하는 것이다. 그것은 1을 더할지도 모르고, 2를 나눌지도 모른다. 단지「있는 데이터의 값에 근거해 다른 데이터를 처리한다」. 예로(「가정부A가 자고 있으면, 가정부B가 두드려 일으키게 한다」) 경우도 처럼, 단순하게「계산」이라고 하는 것보다는「데이터의 가공」쪽이 잘 어울린다.

그런데, 초기의 CPU는 이 4스테이지로 느리게 실행하고 있었다. 페치→디코드→엑서큐트→write back가 각 1 사이클로 처리될 수 있다고 가정하면, 하나의 명령을 완전하게 종료 하려면 4 사이클 걸린다고 하는 이미지이다.
　또한,「사이클」은「Cycle 」로, 원래는 동작 클럭의 Hz와 같은 의미이다. 이전에는 사이클 이라고 부르던 시대도 있었지만, 그 후, 주파수의 단위는 Hz로 통일되었기 때문에, 지금은 Hz 라고 부르는 것이 당연하지만, Hz 의 교류 파형 1개 1개를 셀 때는, 현재에도 사이클을 단위로 하는 것이 관습으로서 남아 있다. 그러니까 위의 문장도「하나의 명령을 완전하게 종료하려면4Hz 걸린다」라고 고쳐 써도 실수는 아니다. 덧붙여서「사이클 타임(CycleTime)」라고 하는 말도 있어, 이것은1 사이클에 걸리는 시간이란 뜻으로, 요컨데 1초를 사이클로 나눈 것이 된다. 즉 50Hz 의 경우, 사이클 타임은1/50 초=20ms (20 밀리 세컨드)다.

그래서, 4 사이클은 너무나 효율이 나쁘다. 여기서 고안된 것이「파이프라인(Pipeline)」라고 하는 방식이다. 페치·디코드·엑서큐트·write back를 동시에 독립하고 실행할 수 있도록 하면, 한 번에 처리할 수 있는 명령의 수는, 파이프라인 구조를 뽑지 않을 때(그림2 상)와 비교해서 4 배로 증가한다(그림2 중).

이것은, 물통 릴레이를 이미지 하면 알기 쉽다. 물을 100m 옮긴다고 하여, 혼자서 이것을 옮기게 되면, 옮길 수 있는 양은 적은 양일 것이다. 그런데 이것을25m 씩4 구간으로 나누어4 사람이25m 씩 옮기도록 하면, 옮길 수 있는 양은 시간당4 배가 된다.

하지만 25m는 혼자서 옮기지 않으면 안 되기 때문에, 아직 고속화의 문제가 있다. 100m를 100 분할해 100 사람 데려 오면, 거의 걷지 않고 물통을 옮길 수 있게 되기 때문에, 옮길 수 있는 양은 압도적으로 증가할 것이다. 이러한 상태로, 파이프라인을 한층 더 세분화해 고속화하는 것이「super pipeline(Super Pipeline)」라고 하는 기법이다(그림 2 하).



(그림 2) 상(파이프라인을 이용하고 있지 않다), 중(파이프라인 사용), 하(super pipeline)의 예.

가로방향에 있는 숫자가 사이클을 나타낸다. 상 에서는 엑서큐트 밖에 실시할 수 없지만, 중 에서는 페치·디코드·엑서큐트·write back의 세 개를 동시에 실시할 수 있으므로, 4 명령째 페치가 시작되어 있다. 하 에서는, 5

명령째의 페치가 시작되어 있는 것을 알 것이다.

파이프라인의 존재를 안다면, CPU의 동작 클럭이 CPU의 능력을 반드시 의미하지 않는다는 것을 알 수 있다.

CPU 성능을 크게 좌우한다 슈퍼 스칼라와 아웃 오브 오더.

앞의 대저택의 이야기를 다시 예로 들면, 세탁물을 처리한다고 하는 일을 한다고 하자. 집사가 한 명의 가정부에게 시키고 있다면, 이것은 아무리 신속히 일을 시켜도 한계는 있다. 빠르게 시키더라도, 세제 세탁10 초, 헹굼15 초, 건조10 초……로 할 수 없고, 한 번에100kg 양으로 하라고 말하는 것도 터무니 없는 이야기다. 하지만, 여기서 다른 가정부를 여러명 찾아, 함께 일 시키면 세탁의 효율은 상당히 오를 것이다. 이와 같이, 병행해 실행할 수 있는 처리를 동시에 실시하게 하는 것이「슈퍼 스칼라(Superscalar , superscalar라고도 한다)」라고 불리는 기법이다.

물론, 모든 처리를 병렬에 실시하게 할 수는 없다. 세탁한 후, 와이셔츠에는 풀먹임을 한다고 하자.이 경우, 세탁이 끝난 것을 분담 해 닥치는 대로 풀먹임 하는 것은 어긋나는 일이 될 것이다. 그래서, 어느 가정부가 우선 세탁물 중에서 와이셔츠를 골라내, 그것을 다른 가정부가 풀먹임 한다고 하는 순서가 되야지 제대론 된 일이 될 것이다. 즉, 슈퍼 스칼라를 항상 실행할 수 있다고는 할 수 없는 것으로,「어떻게 작업을 동시에 실행 할까」가 집사의 열쇠가 된다.

CPU 의 경우, 이 처리를 실시하는 것은 Dispatch 라고 하는 기구로, 슈퍼 스칼라를 이용하는 파이프라인에는 이 Dispatch의 스테이지가 추가되고 있다(그림3 좌하).

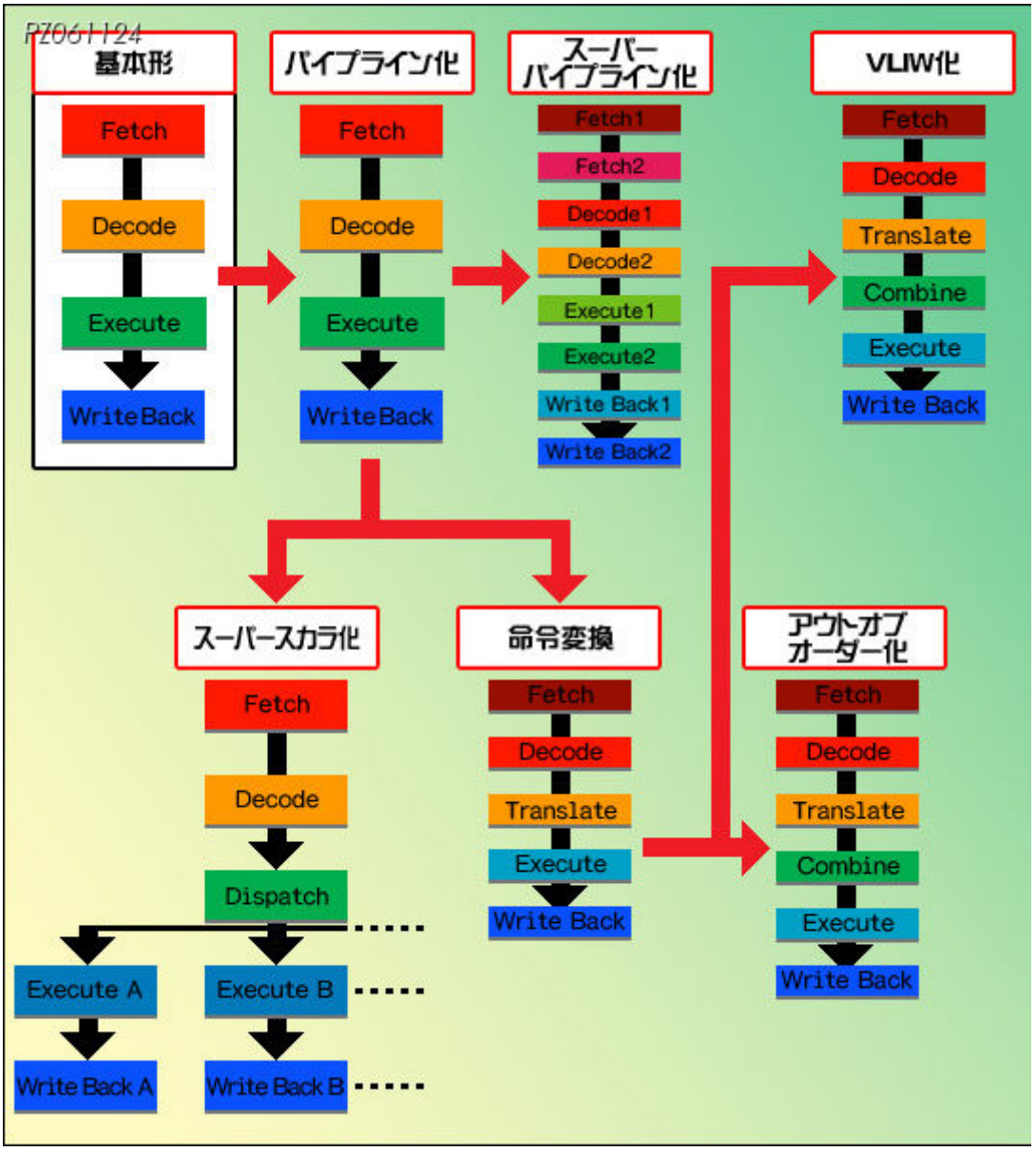


그림 3) 파이프라인, super pipeline, 슈퍼 스칼라, 명령 변환, 아웃 오브 오더,VLIW .적색 화살표는, 기능확장의 흐름을 나타내고 있다

그런데 이번 이야기는, CPU 일반 이라고 하는 것보다, 다소「x86 」(엑스하치로크) 계 CPU 에 특화한 이야기가 되어 있다. x86의 개요에 대해 여기서 깊게 말하는 것은 피하고 있지만, 현재 대부분의 유저들이 사용하는 PC에는 Intel또는 AMD의 CPU을 사용하고 있고,, 이들 모두가 x86계 CPU 이다.

일반적으로 x86계 CPU로 불려진 것은, 1995년에 등장했다. Intel의 CPU "Intel 80386" 가 토대가 되고 있다. 단지 그Intel 80386 (은)는, 원래1978년에 Intel이 발표한「Intel 8086 」의 상위 호환을 유지하고 있고, 이 Intel 8086 의 근본이 되는 것이, 1968년에 등장한 세계 최초의 마이크로 프로세서로 불리고 있는「Intel 4004 」이다.

갑자기 역사의 이야기를 가져와 무엇을 말하고 싶은가 하면 , 현재의 x86계 CPU는, 이러한 먼 옛날의 설계를 질질 끌고 있는 관계로, 하드웨어적으로는 매우 고속화하기 어려운 명령 체계가 되고 있는 것이다.예로, 가정부가 외국인이라면, 집사가 일본어로 말해도 전혀 모르는 것과 같은 정도이다. 또한 집사는 어떤 말을 전할려고 할때, 번역을 해서 전달하지 않으면 제대로 전달이 되지 않을 것이다.

이것이 실제의 대저택이라면「제대로 말을 아는 통역관을 준비해라」라고 말한다, 아주 지당한 이야기가 되지만, CPU의 세계에서는, 실은「명령 변환」(그림3 중앙하, 디코드와 엑서큐트의 사이에 기다리고 있는 트랜스레이트(Translate)가 그것이다)라고 하는 형태로 번역 해 주는 편이 효율이 좋거나 한다. 이러한 이유는, 예로

말을 알아듣지만 둔한 가정부 와 말을 알아듣지 못하지만 움직임이 기민한 가정부가 있다면, 번역의 수고를 들이서라도 기민하게 움직이는 가정부를 사용하는 편이 토탈에서는 빠르게 된다.※2

※2 점점 설정이 나누어지게 되어 어렵게 된 것 같다. 가정부를 낸 것은 실패였는지?

덧붙여서CPU 의 세계에서는, 전자(말을 아는 둔한 가정부)를「CISC (Complex Instruction Set Computer , 복합 명령 세트 컴퓨터)」, 후자(말의 모르는 기민한 가정부)를「RISC (Reduced Instruction Set Computer , 축소 명령 세트 컴퓨터)」라고 부른다. 왜 RISC가 고속인가 하면, 크게는, 처리하는 명령을 단순함에 있는 것으로, 디코드를 고속으로 실시할 수 있게 되어 있는 점에 있다. 복잡한 명령을 해석할 수 있도록 하면, 아무래도 회로 규모는 크고 복잡하게 되어, 결과적으로 가정부는 하나 하나 사전을 찾으면서 명령을 해석해야 하기 때문에, 기민하게 움직일 수 없게 된다.그러니까, 집사측에서 번역해 명령을 간결하게 해 주면 곧 이해할 수 있어 움직일 수 있다는 것이다.

이런 번역(집사가 해주는 번역)에 맞추어,「순서외 실행」(Out of order execution 그림3 우하)로 불리는 메카니즘도 도입되었다. 지금까지 설명해 온 방식은 모두 In order Execution로 불리고 있는 것으로,「결과가 나오고 나서 다음의 처리에 착수한다」를 원칙으로 하고 있다. 가정부에게 접시닦이와 청소와 세탁을 시키는 경우, 우선「접시닦이를 해라」라고 명해「접시닦이 끝났습니다~」의 보고를 받아「다음은 청소를 실시할 수 있다」라고 명령, 그 완료 보고를 받아 다음에 세탁의 명령이라고 하는 상태로, 매회 매회, 처리가 제대로 끝나는 것을 기다려 다음의 명령을 내리는 방식이 인 오더다. 이것에 대해 아웃 오브 오더에서는,「접시닦이와 청소와 세탁을 할 수 있다」라고 정리해 명해 두어 그것을 어떤 차례로 처리할까는 가정부 맡김이 된다.

　여기서 응용력이 없는 가정부의 경우, 솔직하게 접시닦이를 하고, 끝나면 청소를 하고, 그것이 끝나면 세탁에 걸린다고 하는 상태로, 스피드업의 효과는 없다. 그러나, 응용력이 있는 가정부라면, 우선 세탁기를 움직여 두어 세탁＆지나가 끝날 때까지 접시닦이라던지 청소라던지를 병행해 실시할 수 있을 것이다. 세탁물이 대량으로 있고, 몇회인가로 나누어 씻을 필요가 있으면, 우선 세탁기를 움직이면서 접시닦이를 해, 다음에 이번은 세탁기를 움직이면서 청소라고 하는 느낌으로, 완전하게 병행해 실시할 수 있다.

이와 같이, 복수의 처리를 동시에 실시할 수 있는 것이 슈퍼 스칼라이다. 이 때, 집사는 다음에「접시닦이와 청소와 세탁 끝났습니다~」라고 정리하고 보고를 받게 되어 있어 그것이 어떤 순서로 행해졌는지를 관여할 필요는 없다.
　무엇보다, 앞서 말한 바처럼, 가정부의 솜씨가 나쁘면 오히려 시간이 걸려 버리기 때문에, 이용하는 차이에는 잘 concurrent processing 할 수 있는 명령을 줄 필요가 있다. 그러한 이유로 대부분의 x86은 아웃 오브 오더를 슈퍼 스칼라나 명령 변환과 함께 이용된다.

덧붙여서 이 명령 변환 효율을 철저하게 추구한 것이, Transmeta (트란스메타)의 CPU "Crusoe" 와 "Efficeon"이다. 이쪽은 독자적인 CMS(Code Morphing Software)라고 하는 구조로 x86 명령을 독자 명령으로 변환한 뒤, 한층 더「VLIW 」(Very Long Instruction Word , 그림3 오른쪽상) 으로 불리는 방식을 사용해 고효율화의 실현에 성공했다. 하지만, 이것에 계속 되는 메이커는 전혀 없고, Transmeta 자체가 비지니스의 방향으로 전환되어, CPU 사업의 대부분을 홍콩의 ulture.com Technology 에 매각 되었지만, 실제로는 일부의 메이커에서 Efficeon의 제공을 계속해 하고 있지만, 본지 독자로서는, 이미 뒤쫓을 필요는 없을 것이다.

파이프라인과 분기 예측의 관계

파이프라인이나 super pipeline가 일반적으로 채용되게 되어, 파이프라인 단수(stage?)도 많아졌다. Pentium 시리즈로 보면, 7 ~8 단(Pentium)→ 10 단(Pentium II/III)→ 20 단(Willamette/Northwood 코어Pentium 4)→ 31 단(Prescott 코어Pentium 4)(이)라고 하는 느낌이다.

그럼 왜, 파이프라인단수가 많아진 것일까.

방금전 설명했다. CPU 의 파이프라인은, 반도체의 주력 구성요소인「트랜지스터」라고 하는 소자를 조합해 구성되어 있다.

여기서 만일, 그림4 상 과 같은 파이프라인에 대하여, 각 단이 모두 트랜지스터10 개를 직렬에 늘어놓은 구조로 되어 있다고 하자. 이 트랜지스터가 각1ns (1 나노초,1 나노는10 의 마이너스9 승)으로 동작한다고 가정하면, 파이프라인의 각 단은 트랜지스터10 개분=10ns 그리고 처리가 완료한다.사이클 타임은 주파수의 역수로1ns =1GHz 그러니까,10ns는 100MHz . 이 구성에서는, 동작 클락을100MHz 이상으로 하면, 트랜지스터의 속도가 따라 잡지 못하게 된다.

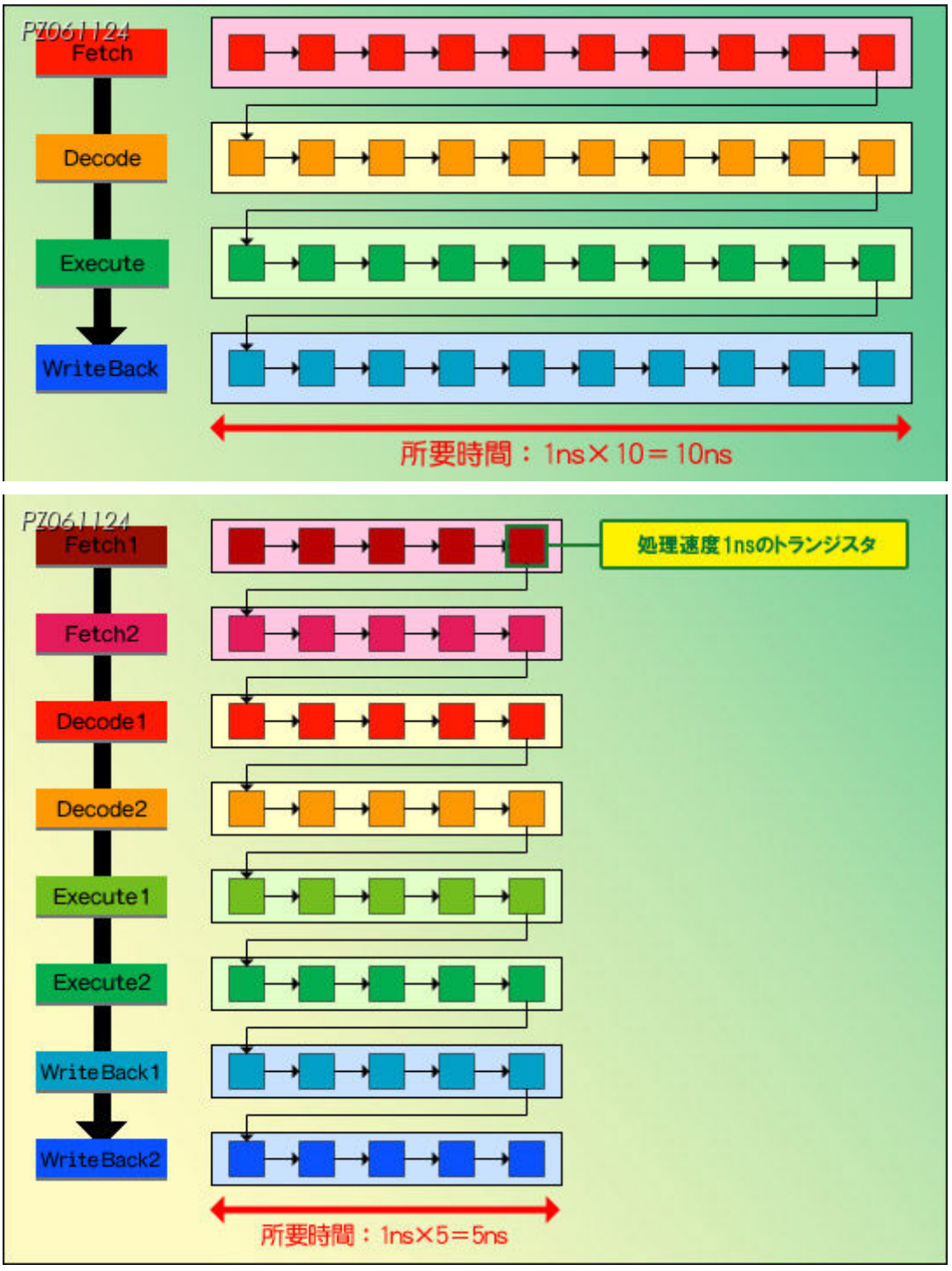


그림 4

그럼, 파이프라인을 그림4 하와 같이 분할해, 파이프라인 각 단이5 트랜지스터로 구성되도록 다시 만들면 어

떻게 될까? 각 단의 처리가 끝날 때까지의 시간은5ns 에 단축되기 때문에,200MHz 까지 동작 클락을 올려지게 된다.

이 예는 너무나 단순화하고 있지만, 실제의 CPU와 파이프라인의 관계에 대한 정도로 생각해 있어. 파이프라인의 단수가 증가하면, 동작 클락을 올리기 쉬워진다 의미가 된다.

~라면, 오로지 파이프라인 단수를 늘리면 좋을까 말하면, 파이프라인 단수를 늘려 동작 클락을 올리면, 부작용도 크다. 그 중에서 첫번째는「파이프라인 하자드(Pipeline Hazard)」라고 불리는 현상이다..

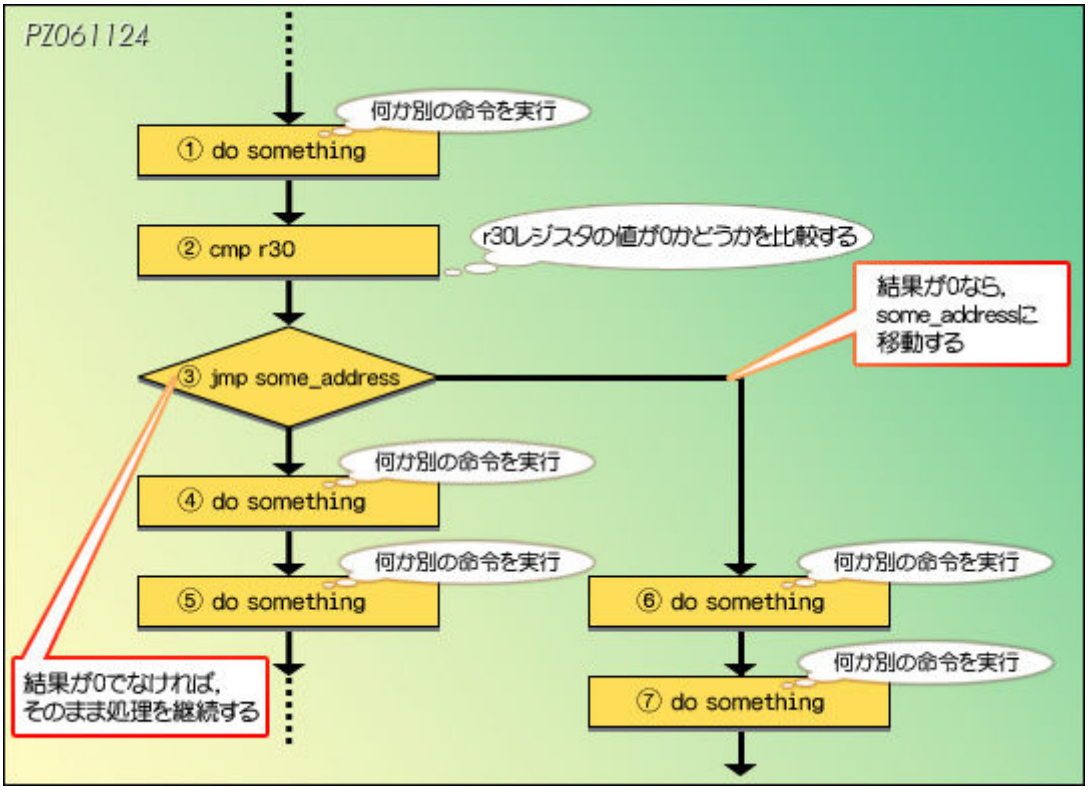


그림5

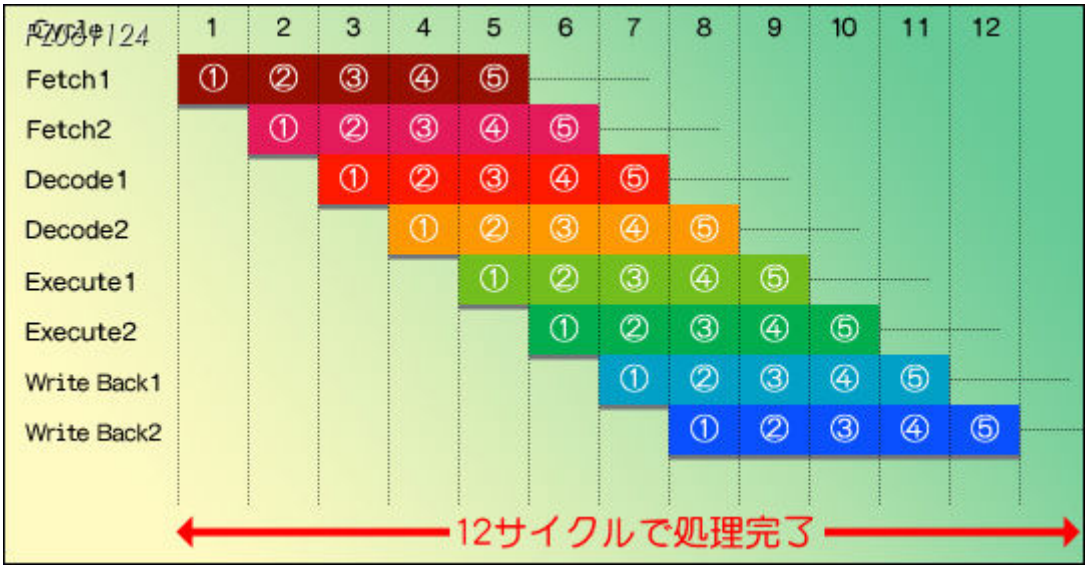


그림 6

그림 7

이러한 문제는, 단수가 증가하거나 아웃 오브 오더를 가지고 있거나 하면 더욱 심각하여, 그 만큼 수백 사이클를 소비해 버리는 경우가 된다. 또한, 그림5에서 (6)과 (7)에 명령에 대한 캐쉬가 들어가 있지 않거나 하면, 그 명령을 메모리로부터 읽어낼 필요가 생기기 때문에, 지연은 수천 사이클에 이르러도 전혀 이상하지 않다.
이것을 일반적으로, 「분기」에 의한 패널티라고 한다. 즉, 한개 명령이 끝나기 까지, 다음 명령을 실행할 수 없게 되어 파이프라인 해저드(파이프라인 무용지물)가 생기게 된다.

다른 예로, 저택에서 약간의 파티가 열린다고 한다. 거기서 집사는 가정부를 대량 동원하고, 요리의 준비를 진행시키게 될 것이다. 물품 종류도 많기 때문에, 계획·사전 준비·조리·상을 차리는 일과 분담하고, 각각 몇명의 가정부를 할당하고 작업을 실시하고 있다.

그런데, 식전 술, 스프에 이어 고기요리에 착수한 근처에서, 처음으로 「출석자들이 채식주의자 집단이었다」는 것이 밝혀지거나 하면, 큰소란이 일어나게 될 것이다. 우선 만들다 만 고리요리(또는, 경우에 따라서는 , 스프)를 조달해 밥으로 한다고 하는 정도로 끝나면 다행이지만, 스테이크든지 새요리의 사전 준비는 전부 소용없게 될 것이며, 그 채식주의자가 「치즈나 버터」를 싫어하면, 메뉴 자체를 전부 다시 생각할 필요가 있을 지도 모르다. 이렇게 되면, 원래 사 들여 둔 식재에서는 요리를 낼 수 없을지도 모르기 때문에, 급히 메뉴를 다시 결정하고 , 계획를 서두르지 않으면 안 된다.

이것은 소규모의 파티라면 「약간의 트러블」의 레벨이지만, 가정부 수십명이라든지 수백명등으로 준비해 온 파티가 되면 「대참사」이다. "분기 미스"라고 하는 것은 이 경우로 「메뉴의 판단 미스」이다. 또한 일반 파티라면 그런 메뉴에 대해 사전 조사를 할 수 있지만, CPU는 이것을 사전 조사 할 수 없는 점 이다. 가능한 것은 「예전에는 이런 느낌의 메뉴로 호평이 있어기 때문에, 이번은 이것으로 가자」라고 "예측" 하는 정도만 가능하다. 따라서, 대참사를 일으키지 않게 「어디까지 정밀하게 예측할까」가 몹시 중요하게 되는 것이다.

이야기를 되돌리면, 이러한 대규모 분기 미스를 일으키지 않기 위해서는, 분기 명령에 관해서 「그것은 어디에 분기 할까」를 예측하고, 분기 미스를 줄이려는 구조가 준비되어 있어야 한다. 이 분기 예측의 메카니즘은 여러가지 있지만, 단순하게 「직전의 분기가 어느 쪽을 향했는지 기억하고 있다」가 , 루프 제어(있는 조건을 채울 때까지 프로그램의 일부를 반복해 실행하는 것)등을 동적으로 해석해 예측하는 것이다. 과거의 분기 이력을 쭉 기록하고 판단하는 것 등도 존재한다. 또한, 분기 미스 했을 때에 어떻게 리커버리를 고속화하는지, 라고 하는 관점에서의 강화도 행해지고 있어 이것도 다양한 형태로 실행되고 있다. 극단적인 예로, 「Itenium 」이라고 하는 서버 전용 CPU는 그림5 의(3)에서, 분기 방향의 결과가 나오기 전에 (4)(5) 와 (6)(7)의 양쪽 모두를 파이프라인에 실어 두어서 잘못되는 편을 없애고 있다.

이야기를 되돌리면, 이러한 대규모 분기 미스를 일으키지 않게, 분기 명령에 관해서는「그것은 어디에 분기할까」를 예측하고, 분기 미스를 줄이려는 구조가 준비되어 있다.이 분기 예측의 메카니즘은 여러가지 있어, 단순하게「직전의 분기가 어느 쪽을 향했는지 기억하고 있다」레벨의 것으로부터, 루프 제어(있는 조건을 채울 때까지 프로그램의 일부를 반복해 실행하는 것)등을 동적으로 해석해 예측하는 것, 과거의 분기 이력을 쭉 기록하고 판단하는 것 등도 존재한다.또, 분기 미스 했을 때에 어떻게 리커버리를 고속화하는지, 라고 하는 관점에서의 강화도 행해지고 있어 이것도 다양한 형태로 실장되고 있다.극단적인 예라고,(본지 독자에게는 별로 관계가 없다)「Itenium 」(아이테니움)이라고 하는 서버 전용CPU 그럼, 도5 의(3)에 두고, 분기 방향의 결과가 나오기 전에(4)(5)(와)과(6)(7)의 양쪽 모두를 파이프라인에 실어 두어서 잘못하는 편을 버린다고 하는 강렬한 사양이 되고 있다.

출처 : 4gamers 작가:大原雄介 편집: felfin

원본출처 : 4gamer.net

본 내용의 해당 저작자와 번역자는 저작권법의 보호를 받습니다.

IP Address : 211.230.xxx.148

댓글 첫번째 (1/2) 페이지의 시작 부분 입니다.함성길

제대로 못 옮겨온 부분이 있어서.. 문맥이 자연스럽지 않은 부분이 있습니다.. 그리고 글자 제한 수에 걸려, 약간 빠진 문맥이 있고 , 읽기 편하게 편집을 하지 못하였습니다. 마지막으로 작가가 너무 쉽게 글을 연재한거라,, 한번 읽어 두시면 좋은 정보가 될 것 같습니다. 11/24 19:21 001

함성길 ■ ■

오오 평소에 cpu에 대해서 알고 싶었는데 이번엔 한자도 안빼먹고 다 읽어봐야 겠군요. 잘 보겠습니다. 감사합니다. *GF™ 6,222 β.10

11/25 12:03 002

2백승훈 ■ ■

스크랩좀해갈게요 *SM™ 3,537 y.10

11/25 16:11 003

이호성 ■ ■

사이클과 Hz는 분명히 다른 개념입니다. 사이클은 하나의 주기를 나타 냅니다.

하나의 명령을 완전하게 종료하려면4Hz 걸린다 => "하나의 명령을 완전 하게 종료하려면

4개의 클럭을 필요로 한다 " 가 맞습니다. Hz는 Clock/second

입니다. 모르시는 것 같지는 않은것 같구 읽는 사람이 자꾸 헷갈리니까 드리는 말씀 입니다

주파수 : 시간당 주기 횟수 (Hz)

몇클럭이 1사이클이냐 ? => 보통 이런개념

예컨데 8051- MCU는 12클럭이 1머신 사이클(1개명령을 처리하는데 필요한 clock 수)입니다. *SM™ 3,788

y.10
11/26 00:16 004
* 심보현 ■ ■

와 이렇게 해주시니 더 깊이 공부 할수 있겠군요. 추천 날립니다. *PZ™ 6,385 β.11
11/26 02:34 005
* 이주호 ■ ■

수고하셨어요 정말 좋은 글이네요. 스크랩 하겠습니다.~ *GF™ 7,638 ε.11
11/26 02:52 006
* 234이지완 ■ ■ 함성길
심보현님,
[그런데, 초기의 CPU는 이 4스테이지로 느리게 실행하고 있었다. 페치→디코드→엑서큐트→write back가 각 1 사이클로 처리될 수 있다고 가정하면, 하나의 명령을 완전하게 종료 하려면 4 사이클 걸린다고 하는 이미지이다.]
[또한,「사이클」은「Cycle 」로, 원래는 동작 클럭의 Hz와 같은 의미이다. 이전에는 사이클 이라고 부르던 시대도 있었지만, 그 후, 주파수의 단위는 Hz로 통일되었기 때문에, 지금은 Hz 라고 부르는 것이 당연하지만, Hz의 교류 파형 1개 1개를 셀 때는, 현재에도 사이클을 단위로 하는 것이 관습으로서 남아 있다. 그러니까 위의 문장도「하나의 명령을 완전하게 종료하려면4Hz 걸린다」라고 고쳐 써도 실수는 아니다. 덧붙여서「사이클 타임(CycleTime)」라고 하는 말도 있어, 이것은1 사이클에 걸리는 시간이란 뜻으로, 요컨데 1초를 사이클로 나눈 것이 된다. 즉 50Hz 의 경우, 사이클 타임은1/50 초=20ms (20 밀리 세컨드)다.]

이 문단을 어떻게 읽어 셨길래,, 자꾸 햇갈린다고 하시는지..
.. 보시면, 틀리다고는 할 수 없다고 하고 있습니다.
즉, 맞는 얘기가 않을지라도, 어느정도 통용이 되는 말로,,틀린 말은 아니라는 말로 들리네요. 11/26 05:11 007
함성길 ■ ■

조금은 어렵지만 열심히 공부하겠습니다 ^.^ 좋은 자료 감사합니다... ^.^;; *PZ™ 5,506 y.10
11/26 11:05 008
한규현 ■ ■

함성길 님 Cycle과 Hz 그리고 주파수의 개념 부터 상기 시켜드리죠
엄밀한 의미의 주파수는 단위시간당 어느정도 주기가 반복 되는가 입니다.
그 주파수의 단위가 Hz입니다.
Cycle의 정확한 의미는 처음 시작한 점으로 되돌아 오는것 즉 1주기를 의미 합니다.
그런데 어떻게 Cycle = Hz 입니까 ?
하나의 명령을 수행 하는데[4Hz] 걸린다는 말을 풀어볼까요
하나의 명령을 수행 하는데 [4 주기/초] 걸린다 말이 이상 하지 않아요?
4주기면 4주기이고 , 4초이면 4초이지 이것은 말이 성립 하지 않은군요 *SM™ 3,788 y.10
11/26 15:02 009
* 심보현 ■ ■ 함성길

--; 정의를 몰고, 알고을 말한게 아니라..필요도 없지만.,(그걸 상기시켜 줄 필요도 없는데, 무슨 말을 하고 싶은 것인지 모르겠네요.)
작가는 그런 상황 또는 그쪽 상황을 말하고 있는데.. 그걸 틀리다 맞다 하면 뭐합니까?
이미,, 그렇게 쓰더라도 맞지는 않지만 실수는 아니다.. 라고 해놓았는데..

그쪽(일본) 세계도 모를 는 상황인데...
그런가 보죠.. 그쪽에서는 작가 설명대로 예전에는 사이클로도 쓰여지다가 Hz로 통일이 되어서,
예전 스타일로 말해도 이해할 수 있다는/틀리지는 않다는 말로..

이런 생각에,, 글을 제대로 읽고 댓글을 다시는지.. 하는 댓글을 단겁니다. 11/26 15:40 010
함성길 ■ ■

[성공과 실패를 결정하는 1%의 CPU원리]란 책에서본 내용이지만 모르겠네요 ㅋㅋ *DK™ 4,428 y.10
11/26 18:25 011
2이민기 ■ ■

함성길님 !
이건 소설을 번역 한게 아닙니다. 기술적인 내용을 전달 하기 위한 것 아닙니까 ?
중요한 것은 가능한 정확한 용어 사용이 필요 하다는 것입니다.
또한가지 제가 전자공학을 공부 시작 할 때부터 배운 개념이고 (1979년) 그당시 대부분 볼만한 전자 기술서적이 일본서적 번역판 이었습니다. 그럼에도 Cycle = Hz로 사용 하는 책은 없었습니다.
그이전 이라면 머 할 말이 없습니다만, 물론 가끔 전원 주파수를 50 ~ 60 Cycle /sec로 표시한 책은 보았습니다만, Cycle 과 Cycle /sec는 다르죠
굳이 원인을 찾아보자면 말 할때 /sec를 생략 해서 말 해버리는 경우는 있죠 그러나 글로 쓸때는 /sec를 생략 하지 않습니다. 번역 하실때 함성길님은 이상하다고 생각하지 않으셨나요
[하나의 명령을 수행 하는데[4 Cycle] 걸린다는 말은 성립 하지만
하나의 명령을 수행 하는데 [4 Hz] 걸린다 말은 성립 되지 않습니다.]
일본작가가 잘못된 개념을 가지고 썼군요 *SM™ 3,788 y.10
11/26 21:37 012
* 심보현 ■ ■ 함성길
네, 알겠습니다. 심보현님으로 인해 더욱 알게 되었습니다.

제가 댓글 단 것은, 앞서 전한 것처럼,,
작가는 그런게 관습으로 남아 있어서,,뜻은 통할 수 있다.. 로,
여기서.. 문제는 그 관습이 정확히 무엇인지 모른다는 것 같습니다. 11/26 22:15 013
함성길 ■ ■

어익후~ 그정도론 그냥 애교로 봐주시지;;; 딴지 무섭네요 -ㅁ-; *XB™ 8,721 β.13
11/26 22:47 014
엄성진 ■ ■

함성길님 제가 미안 합니다.
님이 하는만큼 수고도 못하면서 괜히 딴지 거는 것 같아서
괜히 쪼그만한 것에 흥분 하는 버릇이 있어서, 님의 의도는 알고 있었습니다만
못된 버릇이 튀어 나왔군요 전에 공부할때 용어하나 정확한 개념을 잡지 못해 수개월 동안
헤멘적이 있어서 그런겁니다. 이해 하십시요 수고 하세요 다른 분들께도 죄송합니다. *SM™ 3,788 y.10
11/26 23:07 015
* 심보현 ■ ■ 함성길
아니에요.. 의견을 쓸라고 있는게 댓글인데,, 죄송할 것 까지는..
또한 전.. 심보현님으로 인해 더 알게 되었습니다. 11/27 04:07 016
함성길 ■ ■

성길님 수고 많으셧습니다.. ^^*
성길님 기분 안상하셨다니 다행이네요.. 다음에도 좋은 정보 부탁 드려요~
이렇게 수고해 주시는데.. 까칠한 댓글보면 다음에 글 작성할 맘 생기겠어요.. 흐흐
둥글게 둥글게 좀 삽시다..~ ^_____^ *BF™ 19,379 y.11
11/27 09:22 017
* 12신창훈 ■ ■ 공민환
394-9
역시 한번에 이해는 안되지만.. 느낌은 쪼~~~~금 오는것 같은.. 스크랩해두겠습니다 ^^ *SM™ 4,352 y.11
11/29 16:19 018
* 공민환 ■ ■ 함성길
[승훈]님,[호성]님,[보현]님,[주호]님,[지완]님,[규현]님,[민기]님,[성진]님,[창훈]님,[민환]님
감사합니다.^^ 12/01 10:36 019
함성길 ■ ■ 김지욱
아아,, 어렵지만 몇번정도 계속읽어주면 이해할수 있을것 같습니다
스크랩 추천 들어갑니다 ^^ 12/01 21:02 020
* 김지욱 ■ ■

어렵네요 ;; 흠.. 하지만 인쇄해서 몇번이고 읽어 이해가 될때까지 계속 해보겠습니다!
감사합니다! 추천꾸욱! *SM™ 3,607 y.10
12/02 15:51 021
* 이광로 ■ ■

음 ..좀 어렵긴 하지만 계속 보면 알겠지요 ..추천.합니다. *PZ™ 5,258 y.11
12/03 22:05 022
* 백종길 ■ ■