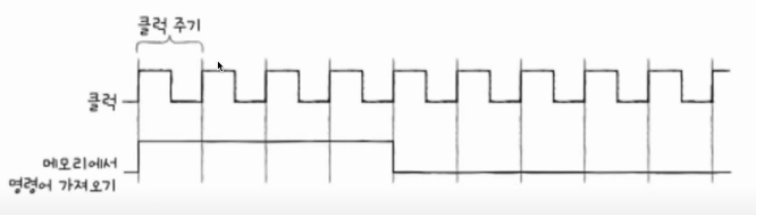
05. Faster CPU

yongzzai

Summary of Chap.05 of book [1]

1 빠른 CPU를 위한 설계

* 컴퓨터의 부품들은 **클럭신호**에 맞춰 일사불란하게 움직임.
* CPU는 명령어 사이클 흐름에 맞춰 명령어들을 실행함.



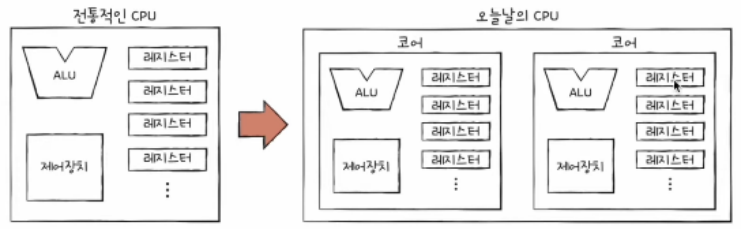
* 일반적으로 클럭신호가 빠르게 반복되면 일반적으론 CPU가 더 빨라짐. (무조건은 아님)

1.1 클럭 속도

* **클럭속도**는 헤르츠 (Hz) 단위로 측정하며, Hz는 1초에 클럭이 반복되는 횟수를 나타냄.
* 1초에 한번 반복되면 1Hz, 1초에 100번 반복되면 100Hz임.
* 하지만 클럭속도를 높이기만 한다고해서 속도가 계속 올라가는 것은 아님 (발열 문제 발생) ** 클럭속도를 높이는 방법 외에는 **코어** 개수를 늘리는 방법과 **스레드** 개수를 늘리는 방법이 있음.

1.2 코어

* 전통적으로, '명령어를 실행하는 부품'은 원칙적으로 하나만 존재함. 하지만, 오늘날의 CPU는 '명령어를 실행하는 부품'이 여러개 존재하고 이 부품을 **코어**라고 부름.



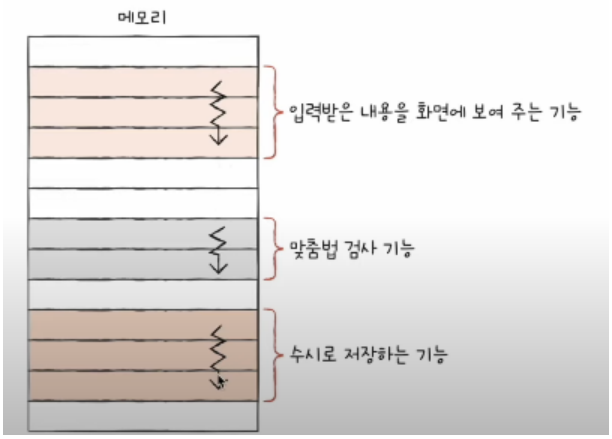
* 이렇게 코어를 여러개 가지고 있는 CPU를 멀티코어라고 부름. (듀얼, 트리플코어, 쿼드코어...)
* 하지만 코어수에 비례해서 속도가 무조건 증가하진않음. (팀플을 생각해보셈)

1.3 스레드

* **스레드**는 아래와 같이 분류되어 정의함.

1. **하드웨어 스레드**
하나의 코어가 동시에 처리하는 명령어 단위를 의미함.
예를 들어, 하나의 코어가 한번에 하나의 명령어를 실행할 수 있으면 1코어 1스레드 CPU임.
만약 두개의 코어가 있고 각 코어가 두개의 명령어를 수행할 수 있으면 2코어 4스레드 CPU임.

2. **소프트웨어 스레드**
하나의 프로그램에서 독립적으로 실행되는 단위를 의미함.



예를 들어, 3가지 기능을 동시에 실행하고 싶을 때는 위와 같이 실행됨.
참고로 하드웨어 스레드가 하나여도 소프트웨어적 스레드를 여러개 만들 수 있음.

1.4 스레드와 레지스터

* 멀티스레드를 설계할 때에 가장 중요한 역할을 하는 핵심 부품은 **레지스터**임. 하나의 명령어를 실행하기 위해 꼭 필요한 레지스트들을 편의상 '레지스터 세트' 라고 표기함.
* 하나의 명령어를 실행하기 위해 필요한 레지스터 세트들을 여러개 가지고 있다면 멀티스레드를 설계할 수 있게됨.
* 참고로 하드웨어 스레드는 **논리 프로세서**라고 부르기도 함.

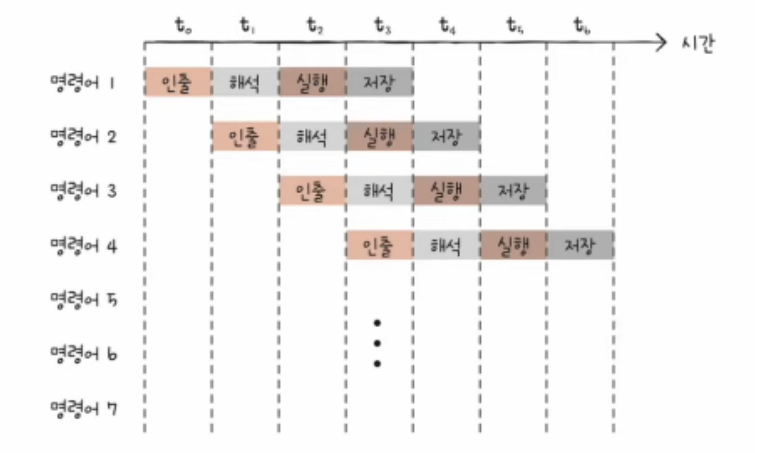
2 명령어 병렬 처리

2.1 명령어 파이프라인

* 하나의 명령어가 처리되는 과정을 비슷한 시간간격으로 나누면 다음과 같은 **명령어 파이프라인**으로 정의할 수 있음.

1. **명령어 인출** (Instruction Fetch)
2. **명령어 해석** (Instruction Decode)
3. **명령어 실행** (Execute Instruction)
4. **결과 저장** (Write Back)

* 이때 같은 단계가 겹치지만 않는다면 CPU는 '각 단계를 동시에 실행할 수 있음'.



2.2 파이프라인 위험

* 명령어 파이프라인이 항상 이상적으로 처리되지 못할 수 있는데 이러한 경우를 **파이프라인 위험**이라고 함.
* 파이프라인 위험에는 세가지 종류가 있음.
1. **데이터 위험**
명령어간의 의존성이 존재하는 경우에 발생함. 모든 명령어를 동시에 처리할 수 없고 이전 명령어를 처리해야만 실행가능한 경우임. 2. **제어 위험**
프로그램 카운터의 갑작스러운 변화에 의한 위험. (예를 들어, 어떤 명령어를 실행하면 메모리 주소가 변경되는 경우 병렬적으로 처리되던 다른 명령어들을 실행하기 어려워짐.) 3. **구조적 위험**
서로다른 명령어가 같은 CPU부품을 사용하려고 할 때 발생함.

2.3 비순차적 명령어 처리

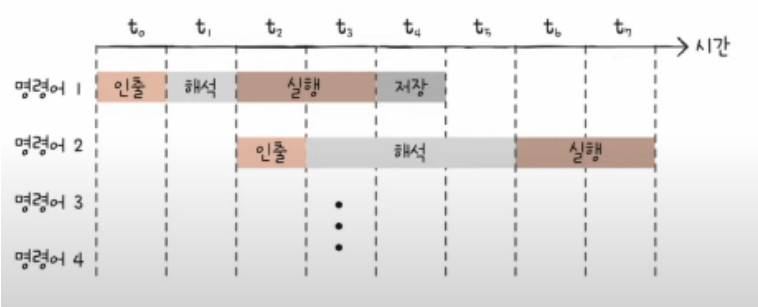
* 비순차적 명령어 처리는 파이프라인의 중단을 방지하기 위해 명령어를 순차적으로 처리하지 않는 방식을 의미함. 의존성이 없는 명령어들 간의 순서를 바꾸는 방식으로 실행됨.

3 명령어 집합 구조

* CPU의 종류마다 명령어의 생김새, 연산, 주소 지정 방식 등은 모두 다름. 예를 들어, 인텔의 CPU에서 만든 실행파일을 ARM CPU로 옮겨와서 특별한 설정없이 실행하면 실행되지 않음.
* **명령어 집합**은 CPU가 이해할 수 있는 명령어들의 모음을 의미함.
* 명령어 집합은 크게 두개의 카테고리로 나뉘어짐.

3.1 Complex Instruction Set Computer (CISC)

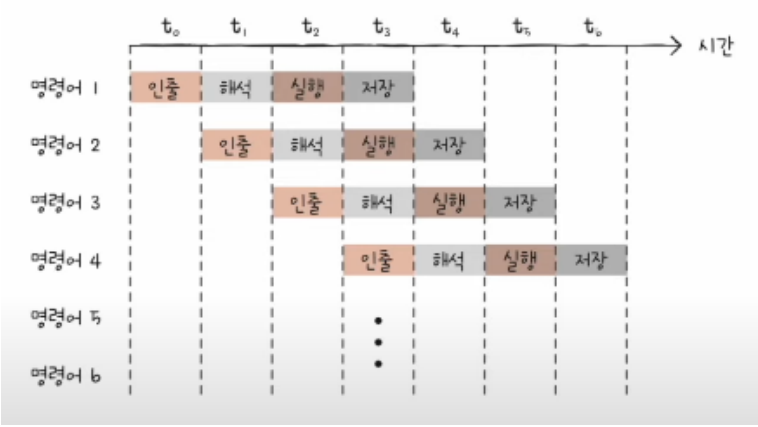
* 복잡한 명령어 집합을 활용하는 CPU이며 대표적으로 x86, x86-64는 CISC기반의 명령어 집합 구조임.
* 명령어의 형태와 크기가 다양한 가변 길이 명령어를 활용. 명령어 하나하나가 강력한 기능을 제공하기 때문에 적은 수의 명령어로도 프로그램을 실행할 수 있음.
* 강력해보이지만, 명령어가 워낙 복잡하고 다양한 기능을 제공하는 탓에 명령어의 크기와 실행시간이 일정하지 않음. 이러한 이유로 명령어 하나를 실행하는 데에 여러 클럭 주기를 필요로 함.
* 각 명령어 별로 실행에 필요한 시간이 달라지면서 파이프라이닝에 불리함.



* 여기에 더해 대다수의 복잡한 명령어는 사용 빈도가 낮음.

3.2 Reduced Instruction Set Computer (RISC)

* **RISC**는 CISC의 복잡성과 파이프라이닝이 어렵다는 단점을 해결하기 위해 짧고 규격화된 명령어 집합을 사용함.
* RISC는 짧고 규격화된 명령어를 활용하기 때문에 파이프라이닝에 유리함.



* 또한 RISC는 메모리 접근을 최소화하고 레지스터을 적극적으로 활용함.

3.3 CISC, RISC 비교 요약

CISC	RISC
복잡하고 다양한 명령어	단순하고 적은 명령어
가변 길이 명령어	고정 길이 명령어
다양한 주소 지정 방식	적은 주소 지정 방식
프로그램을 이루는 명령어의 수가 적음	프로그램을 이루는 명령어의 수가 많음
여러 클럭에 걸쳐 명령어 수행	1클럭 내외로 명령어 수행
파이프라이닝하기 어려움	파이프라이닝하기 쉬움

References

[1] Minchul Kang. 혼자 공부하는 컴퓨터 구조+운영체제. Hanbit Media, 2022.