14강. 명령어 집합 구조, CISC와 RISC

왜 “파이프라인닝에 유리한” 명령어 형태가 중요한가?

파이프라인은 명령어 실행 과정을 여러 단계로 나누어 동시에 처리하는 방식입니다. 그렇기 때문에 각 단계가 규칙적이고 예측 가능할수록 효율이 좋아집니다.

만약 명령어의 길이나 형식이 제각각이라면, 어떤 명령어는 해석이 오래걸리고, 어떤 명령어는 금방 끝나면서 파이프라인의 흐름이 끊기고, 효율성에 직접적이 영향을 준다고 할 수 있습니다.

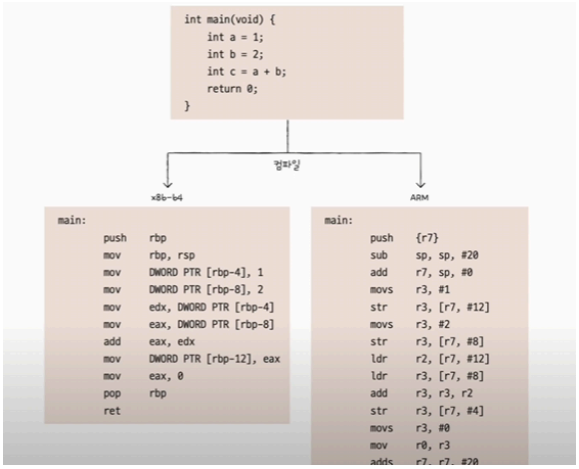
명령어 집합 (ISA: Instruction Set Architecture)

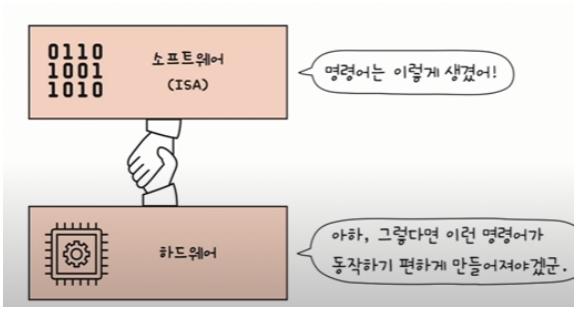
CPU가 이해하고 실행할 수 있는 명령어들의 규격 / 약속입니다.

- 어떤 데이터형?
- 레지스터 종류 / 개수
- 명령어 형식 (필드, 배치, 길이)
- 주소지정방식 (addressing modes)
- 예외 / 인터럽트 모델, 메모리 모델, 시스템 명령 등

같은 C코드라도 ISA가 다르면, 컴파일러는 서로 다른 기계어를 생성합니다.

결국 ISA는 CPU의 언어이며, 하드웨어가 소프트웨어를 어떻게 받아들일지를 규정하는 약속입니다.



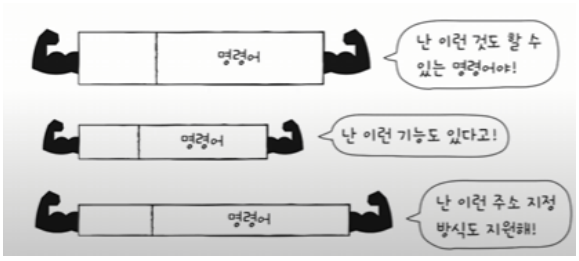


현대의 명령어 집합 구조는 크게 CISC와 RISC 두 가지 흐름으로 나눌 수 있습니다.

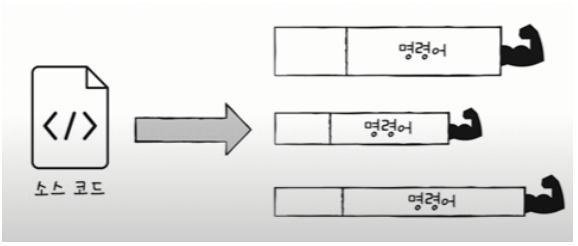
CISC (Complex Instruction Set Computer)

CISC는 이름 그대로 복잡한 명령어 집합을 제공합니다.

x86, x86-64 계열이 대포적이고, 이들은 가변 길이 명령어를 사용하여 상대적으로 적은 명령어만으로도 다양한 기능을 수행할 수 있습니다. 과거에 메모리가 비싸고 한정적일 때, 이러한 특징이 큰 장점이었지만, 파이프라이닝 관점에서는 불리합니다.

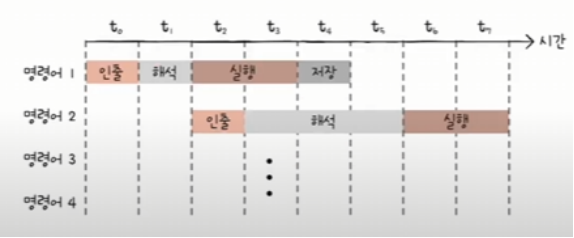


가변길이

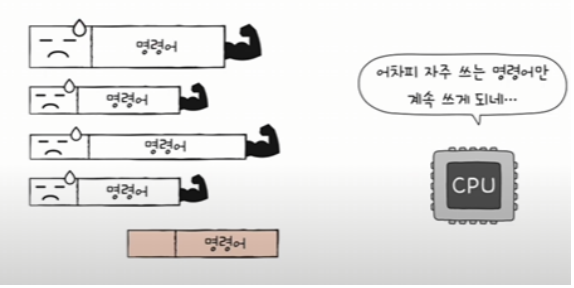


상대적으로 적은 명령어

그 이유는 명령어의 크기와 실행 시간이 일정하지 않아 예측이 어렵고, 복잡한 명령어를 처리하는 데 어러 클럭 주기가 필요하기 때문입니다. 게다가 복잡한 명령어 중 상당수는 실제 프로그램에서 사용 빈도가 낮아, 하드웨어 자원만 낭비되는 경우가 많습니다.



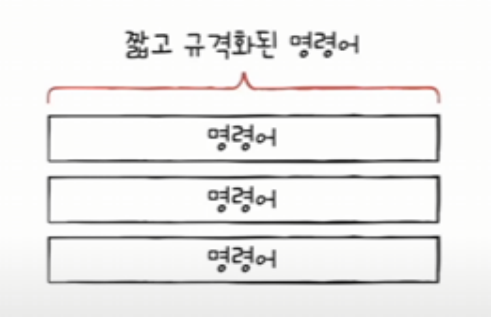
파이프라이닝



사용빈도

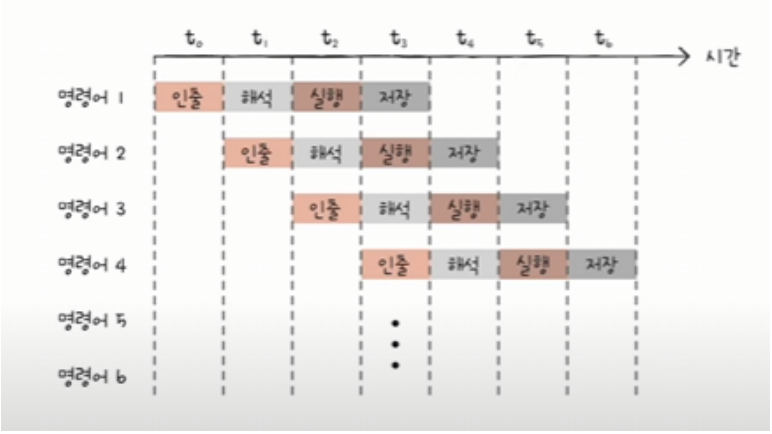
RISC (Reduced Instruction Set Computer)

RISC는 단순하고 규격화된 명령어를 채택합니다. 보통 고정 길이 명령어를 사용하며. 메모리 접근은 load와 store 명령으로만 제한하고 연산은 모두 레지스터를 통해 수행합니다.



고정 길이

명령어의 길이와 형식이 균일하므로 해석이 단순하고, 실행 단계도 일정하여 파이프라이닝에 유리합니다.



유리한 파이프라이닝

단점으로는 명령어 종류가 적기 때문에 프로그램을 실행하려면 CISC보다 더 많은 명령어가 필요할 수 있습니다.

CISC & RISC 정리

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