06 CISC와 RISC

❑ CISC(Complex Instruction Set Computer)

❖ **1950년대 초 모리스 윌크스(M. V. Willkes)**

- 제어 장치를 소프트웨어적인 방법(마이크로 프로그래밍)으로 구현
- 당시에는 하드웨어가 아닌 소프트웨어적 구성이 사람의 흥미를 끌었으나
- 빠르고 저렴한 제어기억 장치가 필요하여 구현의 어려움

❖ **1964년 4월 IBM 시스템/360**

- 가장 큰 모델을 제외하고 모두 마이크로 프로그래밍 사용
- CISC(Complex Instruction Set Computer) 프로세서의 제어 장치를 구현하는 데 널리 사용

❖ **1970년대 후반**

- 명령어가 매우 복잡한 컴퓨터가 연구
- 인텔 계열 : 계속적인 명령어 추가 - 대표적인 CISC 프로세서

❖ **CISC 프로세서**

- 해독기 : 기계어를 제어 기억 장치에 저장된 마이크로 명령 루틴으로 실행

06 CISC와 RISC

❑ RISC(Reduced Instruction Set Computer)

❖ **1980년 버클리 그룹**

- 데이비드 패터슨(David Patterson) & 카를로 세퀸(Carlo Sequin)

❖ **마이크로 명령을 사용하지 않는 VLSI CPU 칩 설계**

❖ **1981년 스탠포드 대학교: 존 헤네시(John Hennessy)**

- MIPS라는 다른 칩 설계&제작
- SPARC 및 MIPS로 각각 발전

❖ **기존 제품과 호환할 필요가 없었으므로 시스템 성능을 극대화할 수 있는 새로운 명령어 세트를 자유롭게 선택**

- 초기 : 빠르게 실행할 수 있는 단순 명령이 강조
- 추후에는 빠르게 실행할 수 있는 명령의 설계가 좋은 성능을 보장한다는 것을 깨달음
- 하나의 명령어가 실행되는 데 걸리는 시간보다 초당 얼마나 많은 명령어를 시작할 수 있는지가 더 중요함

06 CISC와 RISC

❖ RISC 설계 당시 특징

- 상대적으로 적은 개수의 명령어
- 하나의 명령어가 실행되는 데 걸리는 시간보다 초당 얼마나 많은 명령어를 시작할 수 있는지가 더 중요함
- 명령어 개수가 대략 50개
  - 기존 VAX나 대형 IBM 메인 프레임의 명령어 개수인 200~300개보다 훨씬 적음

❖ VAX, 인텔, 대형 IBM 메인 프레임에 반기

- 컴퓨터를 설계하는 가장 좋은 방법은 레지스터 2개를 어떻게든 결합하고
- 명령어를 적게 하여
- 결과를 레지스터에 다시 저장하는 것
- 반기의 근거 : RISC 시스템은 선호할 가치가 있음
  - CISC가 명령어 1개로 처리하는 일을 RISC는 명령어 4~5개로 처리하더라도 기계어를 마이크로 명령으로 해독하지 않아도 되므로 10배 빠르면 이길 수 있음
  - 주기억 장치의 속도가 제어 기억 장치의 속도와 비슷해짐으로써 CISC의 해독으로 인한 손실이 더 커짐

06 CISC와 RISC

❖ CISC와 RISC 특징 비교

표 4-5 CISC와 RISC의 특징

CISC	RISC
하드웨어를 강조한다.	소프트웨어를 강조한다.
명령어 크기와 형식이 다양하다.	명령어 크기가 동일하고 형식이 제한적이다.
명령어 형식이 가변적이다.	명령어 형식이 고정적이다.
레지스터가 적다.	레지스터가 많다.
주소 지정 방식이 복잡하고 다양하다.	주소 지정 방식이 단순하고 제한적이다.
마이크로 프로그래밍(CPU)이 복잡하다.	컴파일러가 복잡하다.
프로그램 길이가 짧고 명령어 사이클이 길다.	모든 명령어는 한 사이클에 실행되지만 프로그램의 길이가 길다.
파이프 라인이 어렵다.	파이프 라인이 쉽다.

06 CISC와 RISC

❖ **RISC 기술의 기능적인 이점에도 CISC 시스템을 무너뜨리지는 못한 이유**

- 첫째, 이전 버전과 호환성 문제와 소프트웨어에 수십억 달러를 투자한 인텔 계열 회사들 때문
- 둘째, 인텔이 CISC 아키텍처에도 RISC 아이디어를 사용할 수 있었기 때문
  - 인텔 CPU에는 486부터 RISC 코어 포함
  - RISC 코어는 단일 CISC 방식으로 복잡한 명령어를 해석하면서 단일 데이터 경로 사이클에서 가장 단순한(보통 가장 일반적인) 명령어 실행
  - 일반 명령어는 빠르지만 일반적이지 않은 명령어는 느림
  - 하이브리드 방식 : 순수한 RISC 설계만큼 빠르지는 않지만 전반적인 성능 향상 - 기존 소프트웨어를 수정하지 않고 실행
- CISC는 하나의 프로그램에 사용되는 명령어 개수 최소화, 명령어 사이클 개수를 희생하는 접근법
- RISC는 명령어 사이클 개수를 줄이고 프로그램당 명령어 개수에 가치 부여

06 CISC와 RISC

❖ **CISC와 RISC의 비교**

표 4-6 CISC와 RISC의 비교

구분	CISC			RISC	
컴퓨터	IBM-360/168	VAX-11/780	Intel 80486	SPARC	MIPS R4000
개발 연도	1973년	1978년	1989년	1987년	1991년
명령어 개수	208개	303개	235개	69개	94개
명령어 크기	2~6바이트	2~57바이트	1~11바이트	4바이트	4바이트
범용 레지스터	16개	16개	8개	40~250개	32개
제어 메모리	420K비트	480K비트	246K비트	-	-
캐시 크기	64K바이트	64K바이트	8K바이트	32K바이트	128K바이트

06 CISC와 RISC

❑ 현대 컴퓨터 시스템의 주요 설계 원칙

- 모든 명령어는 하드웨어가 직접 실행한다.
- 어떤 명령어가 시작되었을 때 최대 효율을 발휘하는가?
- 명령어는 쉽게 해석할 수 있어야 한다.
- 읽기와 쓰기만 메모리를 참조하여야 한다.
- 많은 레지스터를 제공해야 한다.

Apple M3 CPU

출처(2023.10.30): https://www.servethehome.com/apple-m3-family-of-arm-cpus-launched-with-some-shaky-performance-claims/