CISC와 RISC란 무엇인가?

MCU에 대한 카탈로그와 매뉴얼에서 CISC와 RISC 같은 약어들을 본 적이 있을 것이다. 이런 약어들은 무슨 뜻이며, 서로 어떻게 다른 것일까? 본 기고에서는 CISC와 RISC에 대한 간단한 설명과 함께 이들 각각의 장단점을 살펴보고자 한다.

자료제공/ST마이크로일렉트로닉스

CISC는 'COMPLEX INSTRUCTION SET COMPUTER'의 약어이고, RISC는 'REDUCED INSTRUCTION SET COMPUTER'의 약어이다. 복합 명령어 방식에서는 하나의 명령어 실행으로 일련의 복잡한 프로세스들을 수행한다. CISC 아키텍처는 MCU의 성능을 향상시키기 위해 복합 명령어를 이용한다. 그러나 축소 명령어 방식에서는 하나의 명령어 실행으로 간단한 프로세스들을 매우 신속하게 수행한다. RISC 아키텍처는 다수의 축소 명령어들을 신속하게 실행하여 전반적인 MCU 성능을 향상시킨다. 본 기고에서는 각 방식의 특성에 대해 설명한다.

CISC

MCU 아키텍처 개발을 위한 원래의 기반을 제공한 것은 CISC라는 개념이었다. 1970년경에 컴퓨터와 반도체 IC 기술의 결합으로 콤팩트 컴퓨터 IC 유닛 형태의 MCU가 실현됐다. 이 시기에는 전자계산기가 유행하면서 제조업체들은 너나 할 것 없이 보다 뛰어난 계산 성능을 갖는 보다 유용한 계산기 개발경쟁에 뛰어들었다. 오늘날에도 업체들은 저마다 다른 명령어 포맷과 실행 런타임을 사용한다. 그 기본 규칙은 하나의 명령어 실행으로 가능한 한 많은 프로세스들을 수행하는 것이다(자세한 내용은 'CISC의 특성' 섹션 참조). MCU가 발명되자 엔지니어들의 노력은 각각의 명령어에 대해 가능한 한 효율적으로 동작하는 아키텍처를 개발하는 데 집중됐다. 그러다가 1980년경에 RISC라는 혁명적인 MCU 아키텍처가 발표됐다.

RISC

복합 명령어를 실행하는 데는 일반적으로 오랜 시간이 걸리며, 그러기 위해서는 MCU용의 복잡한 로직 회로도 설계해야 한다. 역으로, 다수의 간단한 명령어들을 조합하여 신속하게 실행할 수 있다면 CISC로 행하는 것보다 효율적으로 계산을 수행할 수 있다. RISC는 이러한 아이디어로부터 개발됐다. RISC 아키텍처는 스탠포드 대학과 캘리포니아 대학교 버클리 캠퍼스의 연구 결과 개발됐다. 이 새로운 MCU 아키텍처의 개발을 최초로 발표한 것은 PATTERSON와 DITZEL였다. 이들의 발표는 당시의 MCU 전문가들을 놀라게 만들었다. 그들 대부분은 MCU의 전반적인 효율성이 가능한 한 많은 복합 프로세스들을 하나의 명령어 안에 집어넣는 방식으로만 향상될 수 있다는 생각에 사로잡혀 있었다.

RISC 아키텍처는 고정된 길이의 명령어를 사용하며, 파이프라인이라는 개념을 채택하고 있다. 프로세서들은 파이프라인 내에서 여러 단계로 나누어진 다음 병렬 프로

세싱을 통해 조금씩 수행된다. 따라서 RISC는 클럭 주기당 하나의 명령어를 실행하는 것처럼 보인다**(그림 1참조)**. 이 아키텍처는 또한 다른 새로운 방법들을 이용해 명령어를 신속하게 실행한다(자세한 내용은 'RISC의 특성' 섹션 참조).

CISC의 특성

(1) 명령어의 포맷이나 길이에 관한 규칙이 없음

각각의 명령어는 사양을 충족시키기에 가장 적합한 포맷과 크기를 갖도록 설계됐다. 따라서 그 실행 시간은 명령어 자체에 좌우된다. 명령어의 길이가 고정될 필요가 없는 것은 각각의 명령어가 프로세싱 성능을 극대화하도록 설계되었기 때문이다.(명령어의 길이를 고정시킨다면 실제로 파이프라인의 프로세싱 효율을 억제할 수 있다.)

(2) MICRO-ROM 방식의 명령어 디코딩

MICRO-ROM 방식은 로직의 크기를 줄이고 비교적 복잡한 프로세스들을 수행할 수 있지만, 랜덤 로직 방식보다 오랜 프로세싱 시간을 필요로 한다.

(3) 하나 이상의 클럭 주기로 하나의 명령어 실행

하나 이상의 클럭 주기를 사용하여 하나의 명령어를 실행한다. 여러 클럭 주기가 요구되더라도, 복잡한 프로세스 결과를 전반적으로 보다 효율적으로 단일 패스에 생성하는 것을 목표로 한다.

RISC의 특성

(1) 고정길이 명령어

RISC는 신속한 프로세싱을 달성하기 위한 파이프라인 프로세싱의 일환으로서 고정길이 명령어를 사용하며, 클럭 주기당 하나의 명령어를 실행한다.

(2) 랜덤 로직 방식의 명령어 디코딩

랜덤 로직은 빠른 프로세싱을 수행하기 위해 채택되

그림 1. 3단계 파이프라인의 예

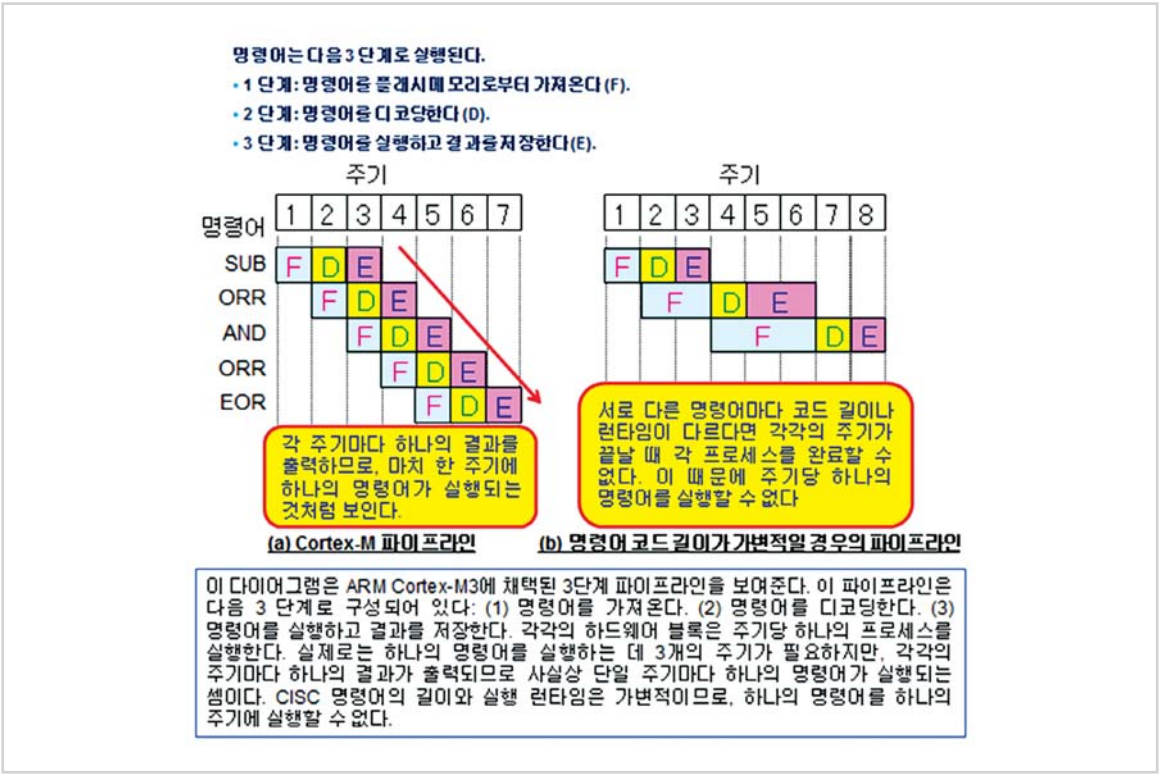

표 1. CISC와 RISC의 비교

	CISC	RISC
로직 크기	대형 MICRO ROM 사용	비교적 소형 랜덤 로직 사용
MCU 개발 주기	길다	짧다
고성능 잠재력	목적에 따라 다름. MIPS, DMIPS 및 COREMARK와 같은 일반적인 벤치마크 데이터 참조.	
코드 효율성	높다	높지 않다
파이프라인 해저드	해당사항 없음	대응책 필요

지만, 하나의 명령어가 하나의 복잡한 프로세스에 상응하지는 않는다. 이 같은 경우, MICRO-ROM 방식은 효율적이지 못하다.

(3) 단일 클럭 프로세싱

파이프라인 프로세싱은 단일 클럭 시스템을 사용한다.

(4) 파이프라인 프로세싱

앞서 언급했듯이, 파이프라인 프로세싱에서는 클럭 주기 당 하나의 명령어가 실행된다.

CISC와 RISC의 장단점

(1) 로직 회로의 크기

RISC의 경우에는 소형 로직으로 충분한데, 이는 각각의 명령어가 단순한 프로세스에 상응하기 때문이다. 그러나 CISC의 경우에는 각각의 명령어에 대해 실행되는 프로세스들이 복잡하므로 대형 로직이 필요하다. 이러한 이유 때문에 MICRO-ROM 방식을 채택한다.

(2) 개발 주기

이는 최종 사용자에게는 직접적인 영향을 미치지 않을지 모르지만, RISC는 로직 크기가 작기 때문에 MCU 로직 회로를 개발하기 위한 노력과 비용이 비교적 적게 든다. 그러나 오늘날의 첨단 자동 설계 툴은 대형 로직과 소형 로직 간의 차이점을 상쇄시키는 데 도움을 줄 수 있다.

(3) 고성능의 잠재력

어느 쪽의 성능이 더 우수한지 딱 잘라 말하기는 힘들다. 오늘날의 몇몇 CISC MCU는 RISC의 장점을 일부 공유하고 있으며, 몇몇 RISC MCU는 CISC의 장점을 일부 공유하고 있기 때문이다. 따라서 MCU의 종류를 선택할 때는 MIPS, DMIPS 및 COREMARK와 같이 MCU 평가를 위해 발행된 벤치마크 데이터를 참조해야 한다.

(4) 코드 밀도

CISC 아키텍처에서는 하나의 명령어 실행으로 다수의 프로세스를 수행한다. 그러나 RISC 아키텍처에서는 다수의 명령을 사용하여 하나의 프로세스를 수행하므로 RISC는 동일한 프로세스들을 실행하는 면에서는 불리하다. 전형적인 RISC 제품인 ARM CORTEX-M 시리즈는 코드 효율성을 향상시키기 위해 16비트 명령어와 32비트 명령어를 둘 다 채택하도록 설계됐다.

(5) 파이프라인 해저드

분기 명령어와 같은 이벤트들이 필요 없는 주기들을 생성하여 파이프라인을 중단시키는 경우가 있다. 이러한 경우를 파이프라인 해저드(PIPELINE HAZARD)라고 한다. 이론상 파이프라인 해저드를 완전히 없앨 수는 없지만 피해를 최소화하기 위한 조치를 취할 수는 있다. 예를 들어, 분기 명령어 문제를 다루기 위해 CORTEX-M 시리즈에는 분기 목적지 예측 기능이 탑재되어 있다.

이 장의 개요를 **표 1**에서 볼 수 있다. 위의 설명이 보여주듯이, CISC와 RISC는 저마다 장단점을 갖고 있다. 따라서 우리가 제공할 수 있는 가장 실용적인 팁은 자신의 필요에 가장 적합한 MCU를 선택해야 한다는 것이다. SN