제4장 마이크로프로세서 내부 들여다보기

원칙적으로 마이크로프로세서는 (일반 컴퓨터도 동일함) 논리소자로 구현된 것이므로 '1' , '0' 의 코드의 조합으로 모든 것이 처리됩니다. 따라서, 프로그램이라는 것도 그냥 보면 '1', '0' 의 조합에 불과한 것이며, 이는 인간이 보기에는 너무 어려운 암호 숫자가 되지요. 따라서 인간이 그래도 알기 쉽도록 하기 위해서 동작 기능에 따는 약어를 쓰며, 이를 Mnemomic (기억하기 쉬운)이라 하고, 이것으로 단위기능을 제어하는 명령어를 만듭니다.

☞ 부록에 있는 명령어 관련 자료를 자주 보고 Mnemonic과 그 기능을 이해해야만 프로그램을 작성할 수 있습니다.

이러한 명령어의 개수가 많으면 여러 다양한 기능을 특정한 하나의 명령어로 바로 구현할 수가 있습니다. 반면 마이크로프로세서의 내부회로가 각각의 명령어를 구분하고 동작시키는데는 오히려 시간이 많이 걸릴 수 있습니다. 여러분이 쉽게 이해하기 위한 예를 들면, 5×4는 5+5+5+5 형태로도 표현할 수 있습니다. 어느 것이 더 간편할까요? 구구단을 알고 있는 사람은 5×4가 더 빠르고 쉬운 법입니다. 그러나 구구단을 외우고 있어야 합니다. 즉 회로가 더 복잡해지며, 이에 따른 노력이 더 요구된다는 것입니다. 명령어에서는 어떨까요? 동일한 의미가 성립됩니다. 예전에는 명령어 개수가 많은 것(Complex Instruction Set Computer : CISC)이 많이 사용되었으나, 요즘에는 명령어 개수가 적은 것(Reduced Instruction Set Computer : RISC)이 프로그램 수행이 더 빨라 더 많이 사용되고 있습니다. 그러면 우리가 이번 실험에 사용하는 PIC16계열은 어떨까요? 총 35개의 명령어만 존재합니다. 그리고 모든 명령어는 single word(메모리 한 단위만 차지함)입니다. 그러면 이것은 많은 것입니까, 적은 것입니까? 명령어 개수로는 적은 것입니다. 그러나, working register가 많아 동작상으로 많은 것입니다.

이제 여러분은 data sheet 첫 장의 "High Performance RISC-like CPU" 라는 내용을 이해하신 것입니다. 다음으로 중요한 기능이 얼마나 속도가 빠르냐 하는 것입니다. 여러분이 PC를 구입한다면 가장 먼저 CPU속도에 관심을 갖지 않은가요? 마찬가지입니다. 마이크로프로세서를 선택할 때 최고 속도는 중요합니다.

즉 명령어를 처리하는데는 시간이 소요된다는 의미입니다. 왜냐고요? 명령어는 프로그램으로 기억소자에 기억된 내용이며, 이것을 어떤 순서에 (프로그램의 흐름) 따라서 순차적으로 읽어와 수행하기 때문입니다. 이러한 순차적인 기능을 만들어 주는 것을 clock이라고 하며 CPU의 속도를 결정합니다. Clock 1개는 신호가 1에서 0으로 변하여 다시 1이 될 때까지 또는 0에서 1로 다시 0으로 될 때까지입니다. 이것을 1 clock 또는 1 cycle이라고 합니다. Clock의 기본단위는 주파수입니다. 즉 하나의 pulse가 반복되는 시간의 역수를 의미하지요. Data sheet에 DC-20㎒로 되어있습니다. 20㎒는 최고 속도이고 DC는 최저속도입니다.

☞ 마이크로프로세서도 동일 이름이지만 최고 속도에 따라서 가격이 서로 다릅니다. 구분하여 사용해야 합니다.

☞ 많은 프로세서가 최저속도가 DC가 아니고 높은 것이 많이 있습니다. DC는 아주 느리게 동작시킬 수 있다는 것이므로 여러분이 한 명령어의 동작 상태를 눈으로 확인 할 수 있다는 의미이며, 또한 동작 속도가 느리면 전기소모가 아주 적어지므로 건전지 등을 전원으로 사용하는 경우에는 아주 좋은 규격입니다.

****
그러면 구체적으로 기억소자에 기억된 내용(명령어)은 어떻게 수행될까요?
****

먼저 수행할 프로그램이 들어있는 번지를 알아야 하며, 당연히 이를 기억하고 있는 기억장소가 있습니다. 이것을 PC(program counter)라고 합니다. PC는 전원을 인가하면 (power on reset된다고 함) 표준적으로는 0000으로 설정됩니다. 그러나 프로세서 종류에 따라 reset 시작번지가 다를 수도 있으므로 data sheet를 참고해야 합니다. 이런 관계를 가장 잘 나태 주는 표가 **memory map**입니다. 우리가 실험에 사용할 PIC16F876의 memory map를 data sheet에서 꼭 찾아보시오.

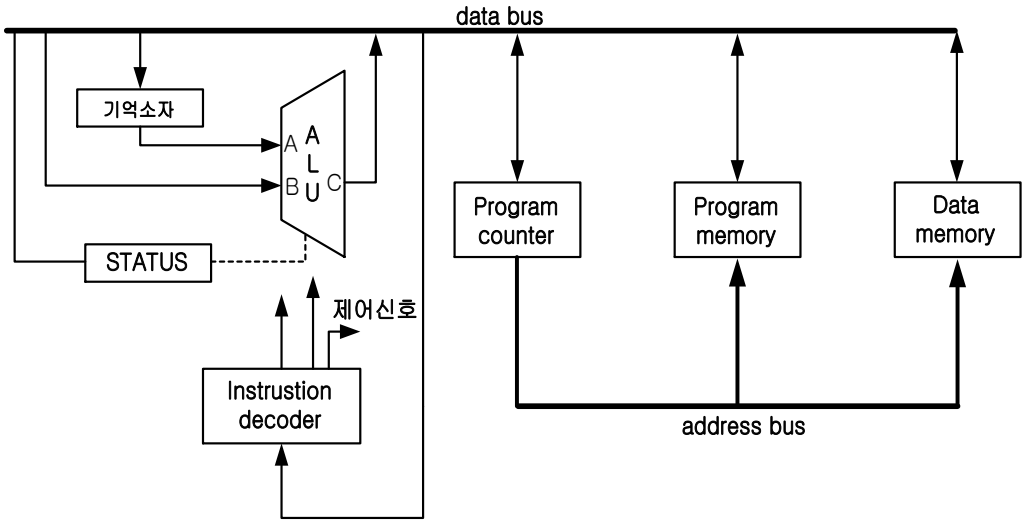

그림 4-1. 마이크로프로세서의 내부 구성(1 Chip 기준)

컴퓨터의 최초 동작은 PC 값이 address bus(주소버스)를 통해서 메모리를 선택하고, 선택된 메모리에 기억된 내용을 읽어오게 됩니다. 즉 전원이 투입되면 0000번지가 선택되어, 이 번지에 기억된 내용이 data bus에 실리게 된다는 것입니다. 이때 명령어를 읽어오겠습니까? 데이터를 읽어오겠습니까? 처음 읽어오는 것이므로 당연히 명령어이겠지요. 즉 초기에 내부 회로가 프로그램 메모리를 선택하여 명령어(instruction)를 읽어오지요. 그리고, 당연히 instruction decoder라는 곳에 저장되어, 명령어를 분석하여 각 명령어에 해당되는 제어신호를 만들어 내지요. 제어신호에 의해서 선택된 data memory의 내용으로 ALU가 연산을 하고 결과를 기억소자에 저장합니다.

이렇게 따져보면 한 명령어는 내부적으로는 2단계의 동작과정을 거쳐 완성됩니다.

instruction cycle = fetch cycle + execute cycle

Fetch cycle은 명령을 가져와 해석하는 것을 의미하며, execute cycle은 명령어를 수행하는 것으로 간단하게는 data를 처리하는 것이라고 생각하시면 됩니다. 그리고, 각각의 기능은 내부적으로 여러 단계를 거치게 되므로 clock 1개만으로 처리되지 못합니다.

Data sheet의 그림3-2를 보면 fetch cycle이나 execute cycle 모두 4개의

cycle을 필요로 합니다. 그러면 1개의 명령어를 처리하는데 총 8 cycle이 필요하게 됩니다. 1개의 명령어 수행 시 많은 clock이 요구된다는 것은 명령어의 처리속도를 떨어뜨리게 되는 것이지요. 따라서, 이것을 보다 빨리 처리하기 위해서는 프로그램 메모리와 데이터 메모리가 공통으로 사용하는 프로세서 내부의 버스를 각각 기능별로 나누는 것입니다. 즉 통합된 데이터 버스를 순수한 데이터 버스와 명령어 버스로 나누는 것입니다. 그렇게 하면 fetch와 execute의 두 기능이 동시에 일어는 것이 아니므로 중복하여 처리할 수 있지요. 즉 각 bus입장에서는 중복이 아니라 연속적으로 수행되는 것입니다. 이러한 방법을 pipe line 구조라고 하며 컴퓨터에서는 흔하게 사용하는 방법입니다. 당연히 내부회로구성이 변해야 되겠지요. 전자의 컴퓨터 구성법을 Von-neumann 구조라고 하며, 후자를 Harvard 구조라고 합니다. 요즘은 대부분의 CPU에서 속도를 빠르게 하기 위하여 Harvard 구조를 사용합니다. 따라서 실험에 사용하는 프로세서는 한 명령어 수행시간이 4 cycle이 되며, 시간으로는 400 *ns*가 됩니다.

☞ Bus는 회로상의 선을 의미하며 정보가 다니는 통로입니다. 도로를 생각하시면 됩니다. Data bus는 양방향 통행가능 1차선 도로이며, address bus는 단방향성(일방통행)1차선 도로와 같습니다. 당연히 2차선으로 만들면 속도가 빨라지겠지요!

☞ Data bus가 양방향성이 되어야하는 것은 이해할 수 있겠지요?

다음은 한 명령어가 한번에 처리할 수 있는 정보량이 얼마인지 알아보도록 하겠습니다. 즉 data bus가 구체적으로 몇 개의 선 혹은 bit로 되어 있는가를 의미합니다. 왜냐하면, 선 한개에는 동시에 한 신호만 전송되기 때문에 유용한 정보를 가지려면 최소한 몇 개의 bit가 모여야 됩니다. 여러 가지 역사적인 흐름이 있었겠지만, 인간이 사용하는 정보는 8bit를 기본단위로 나타낼 수 있습니다. 8bit는 경우의 수가 256개로 256가지의 서로 다른 정보를 정의할 수 있는 양입니다. 구체적으로는 PC 자판에 있는 영문자, 기호, 제어키 등을 다 포함하여도 충분하며, 따라서 단위 정보를 나타내기에는 부족이 없겠지요. 따라서 우리는 8bit를 새로운 단위로 한 byte라고 합니다.

****
그러면 CPU의 성능을 구분하는 기본단위는 무엇이 될까요?
****

해당 CPU가 한 명령어로 처리할 수 있는 정보량인 data bus의 bit 수입니다. 우리가 8bit CPU라고 부르는 것은 data bus가 8bit라는 것이고, 32bit CPU라고 부르는 것은 data bus가 32bit로 한 명령어로 처리되는 정보량이 0~4,294,967,295까지 라는 것입니다. 당연히 bit수가 올라가면 내부회로가 복잡해지겠지요. 그러나 한 명령어로 처리할 수 있는 데이터의 량이 커지므로 아주 편리합니다. 숫자 개념으로 예를 들면 8bit는 최대 255까지 숫자를 연산할 수 있으나, 32bit는 4,294,967,295의 크기까지를 한 명령어로 처리 할 수 있습니다. CPU가 가지는 이러한 차이는 사용자 입장에서는 처리속도만 문제가 되지 사용 방법이나 프로그램 작성에서는 원칙적으로 동일합니다. 또한 모든 용도가 큰 숫자를 다루는 것이 아니며, 따라서 아직도 8bit CPU가 가장 많이 사용되는 CPU입니다.

****
명령어와 bit 수와의 관계는 무관하나요?
****

그리고 앞에서 설명한 Von-neumann 구조는 data bus를 공용으로 사용하기 때문에 명령어도 8bit 단위로 만들어져야 하며, 이는 최대 명령어 개수가 256가지로 제한된다는 것입니다. 그러나 Harvard구조는 명령어 bus가 별도로 있으므로 데이터 bus bit 수와는 무관하게 명령어 bus bit 수를 만들 수 있습니다. 즉 사용하고자 하는 명령어의 기능에 따라서 8bit보다 더 큰 bit 수로 만들 수가 있겠지요. 이는 한 명령어로 더 다양한 기능을 구현할 수 있다는 의미이며, PIC16에서는 14bit가 사용되고 있습니다. 14bit는 명령어와 8bit인 데이터를 동시에 처리 가능하다는 것입니다. 당연히 처리속도가 빨라지겠지요. 따라서 거의 모든 명령어는 single instruction cycle(400 *ns*)로 동작됩니다. 명령어의 동작상 불연속 동작을 하는 몇 개의 branch 기능 명령어는 two cycle를 필요로 합니다.

여기서 우리는 한 단위일이 수행되려면 명령이 있어야 하고 데이터가 있어야 합니다. 만약 명령어가 8bit로 만들어졌다면, 데이터는 별도의 8bit로 추가

되어야 할 것입니다. 따라서 8bit를 두 번 처리해야만 한 단위 일이 완성됩니다. 그러나 우리가 실험에 사용하는 PIC16 계열은 한 명령어로 명령과 데이터를 동시에 처리할 수 있으며, 이런 명령어를 single instruction이라고 합니다. 명령어에 대한 자세한 내용은 부록의 명령어를 참고하십시오.

☞ 이 내용은 CPU의 처리 속도와 바로 연관되므로(clock만 높다고 꼭 빠른 수행 속도를 가지는 것이 아닐 수 있음을 의미함) 세심히 따져보아야 하며, CPU의 선택에 중요한 요소입니다.

****

명령어에 의해서 처리되어 얻어진 data은 어디에 저장될까요?

****

앞에서 프로그램 즉 명령어(명령과 데이터의 집합체)는 프로그램 메모리(영구적인 내용을 저장하는 ROM 형태의 기억소자)에 저장된다고 하였습니다. 그리고 그것이 PC에 들어있는 주소값에 따라서 읽혀져 수행되지요. 그러면 수행중 발생된 데이터는 어디에 저장할까요? 별도의 저장할 장소가 필요하겠지요. ALU 연산 결과 발생한 데이터의 저장은 영구적인 저장이 아니므로 RAM을 사용하며, PIC16F876에는 1 chip 마이크로프로세서로 마이크로프로세서 내부에 72의 특수목적용(제어용) 기억장치(register)와 368개의 범용(데이터용) 기억장치가 있습니다.

특수목적용은 마이크로프로세서 내부를 제어 관리하는 용도로 사용되는 것으로 마이크로프로세서 동작과 밀접한 관계가 있습니다. 따라서 여러분이 함부로 변경하면 안됩니다. 어떤 의미에서는 이 **특수목적용 register를 활용하는 법을 배우는 것이 마이크로프로세서를 배우는 것**이 됩니다. 여러번 언급된 PC도 여기에 속하는 register입니다. 자세한 것은 data sheet에서 확인하십시오.

마이크로프로세서의 내부 구성도를 보면 알 수 있겠지만, 특수목적용 register 중에 아주 특별하게 사용하는 1개의 register가 있습니다. ALU의 출력과 입력사이에 들어있는 W-reg.(working register)입니다. 이를 다른 마이크로프로세서에서는 A 또는 ACC(accumulator)reg. 라고도 합니다. 이 레지스터는 ALU의 연산에 절대적으로 관여하는 기억소자에 여러분이 가장 많이 사용하게 되는 레지스터입니다.

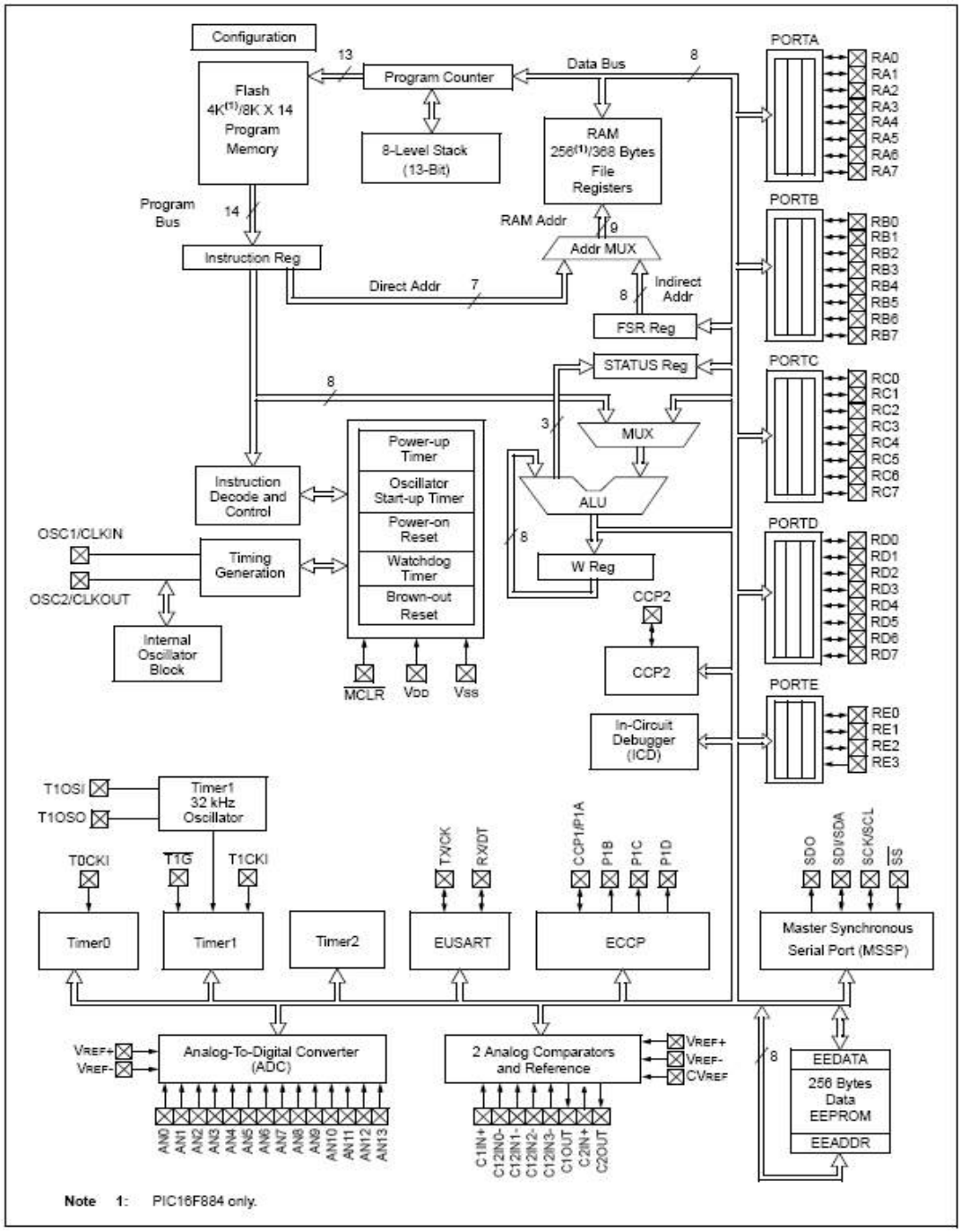

그림 4-2. PIC16F876의 내부 구성도

****

ALU와 결합된 W-reg. 또는 ACC reg. 가 어떻게 동작되나요?

****

ALU는 기능상 2개의 입력을 필요로 합니다. 다시말하면, 더하거나 빼거나 등의 연산은 데이터가 2개 있어야 합니다. 그러나, CPU의 내부 data bus는 1개 밖에 없습니다. 따라서 1개의 data bus로는 동시에 서로 다른 2가지 데이터를 보낼 수 없습니다.

이는 ALU가 동작하기 위해서는 먼저 ALU의 한쪽 입력에 데이터를 옮겨서 기억시켜 두고 사용해야 합니다. 이런 일을 매번 하려면 처리속도가 떨어지게 됩니다. 그런데, 일반적으로 어떤 연산을 하게 되면 그전 결과를 계속 사용하는 경우가 아주 많습니다. 따라서 ALU의 출력에 바로 출력결과를 저장할 수 있는 기억소자 즉 W(또는 ACC)로 명명되는 레지스터를 두고, 그 레지스터의 출력을 ALU의 입력으로 연결시킨 것입니다. 이 구조가 ALU를 가장 효율적으로 동작시킬 수 있지 않습니까? 따라서 모든 ALU는 이런 구조로 만들어집니다. 우리는 ALU가 하드웨어적으로 이런 구조를 가지고 있다는 것을 알아야 하고, 또 당연히 **ALU 관련 명령어는 모두 W(또는 ACC)reg.와 연관되어 동작**된다는 알아야 합니다. ALU 관련 명령어의 기본 형태는 모두 W 레지스터를 기본으로 연산을 하며, 원칙적으로 연산결과가 바로 W 레지스터에 저장됩니다. 부록에서 ALU와 관계되는 명령어를 확인하여 봅시다.

W 연산 data ---> W

☞ 여러분은 컴퓨터가 동시에 여러 가지 일을 하는 것처럼 느끼지만, CPU내부에서는 서로 다른 일을 교대로 순차적으로 처리하는 것입니다. 따라서, CPU의 처리속도가 빨라야 우리가 느낄 때 여러 가지 일을 동시에 처리하는 것처럼 느낍니다.

■ HOME WORK D

1) PC에 사용되는 CPU는 CISC입니까, RISC입니까?

2) 컴퓨터와 PC의 차이점을 요약해 봅시다.

3) 컴퓨터와 컨트롤러의 차이는 무엇일까요?

4) PIC16F876의 명령어 중 각 기능별로 1개씩 설명해 보시오.

5) 부록의 명령어를 이용하여 5+4+3을 하여 W 레지스터에 저장하는 프로그램을 작성하시오.