친절한 임베디드 시스템 개발자 되기 강좌 : ARM Inside

ARM Inside이제부터, ARM Processor에 관한 이야기.
다른 종류의 Processor들도 있겠지만, 아무래도, Embedded 세계에서는 나름 지배자격으로 통하고 있는 ARM Processor를 다루어야 되지 않을까 해서 ARM을 선택했습니다. 아무래도 Embedded를 다루는 사람들이면, ARM을 다루지 싶어서 (다뤄봐야 할 것 같아서) ARM processor를 본격적으로 다뤄 보겠습니다. ARM Processor가 어떻게 동작하는 지 정도만 살펴 보아도, 다른 여러 가지 Processor에 대해서도 자신감이 붙을 지경이니까, 조금만 더 신경 써서 보세요. 힐끗.

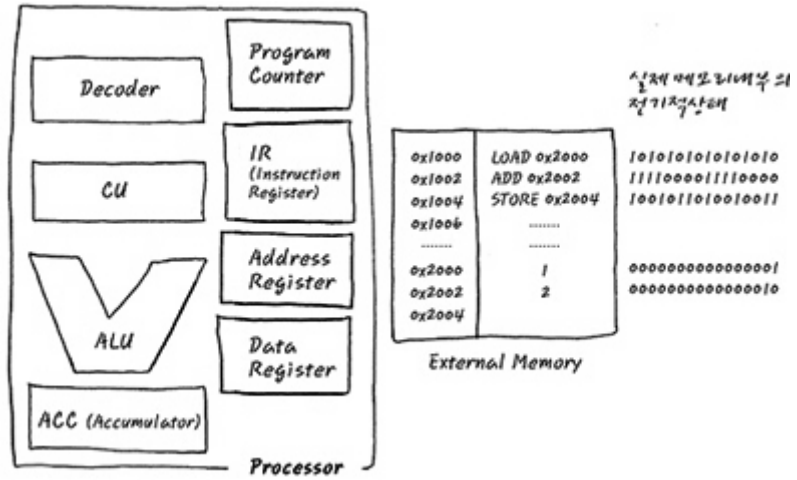
일단, ARM Processor에 대한 설명들은 다른 책들에서도 지겹도록 보실 수 있으니까, 저는 그런 시시껄렁한 이야기는 안 할 작정입니다. 이런 이야기들은 인터넷을 찾아 보셔도

수도 없이 나오니까, 또 하면 재미 없잖아요. 머리만 지끈지끈 아프고요. 그나마, 어떤 특징이 있는지 정도만, 나열한다면, 32Bit RISC processor이고, Big/ Little Endian 지원, Fast Interrupt Response정도 에요.

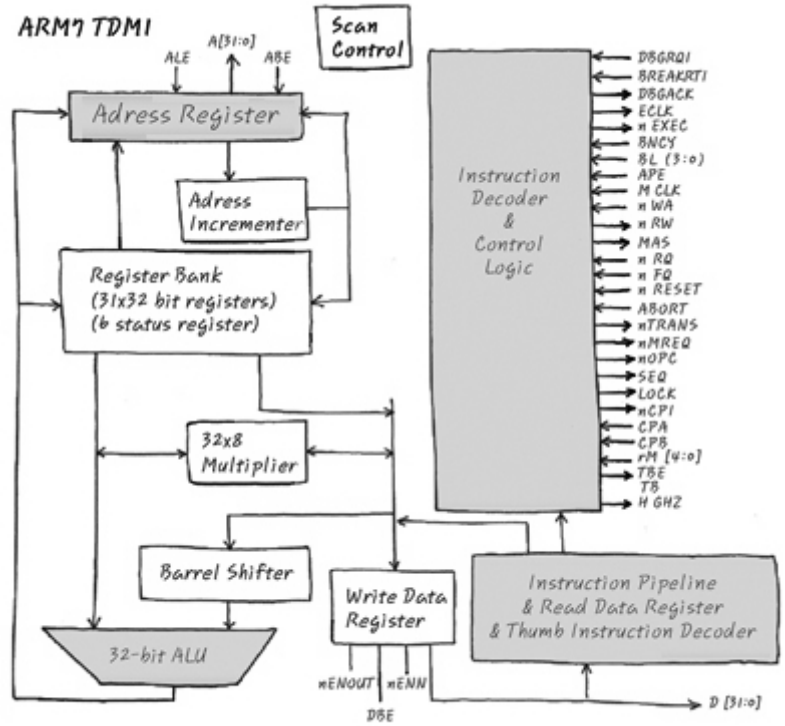
ARM core뒤에 붙는 7TDMI 라든가, 9EJ-S 어쩌구 하는 것들은, 별로 다루지 않는 것이 좋을 것 같아요. 어차피, ARM Core의 동작은 이런 여타 option들과 상관없이 - 완전 없다고 하면, 좀 곤란하지만 - 설명 가능하니까, Pass 입니다. 별로 그런 것들을 외우는데, 익숙하지 않은 터라 이 정도에서 마무리 짓는 것이 저로서는 최선의 선택이 아닌가 싶습니다.

후에, Assembly를 살펴 볼 때도 이런 구분은 짓지 않고 가겠사옵니다. 왜 이리 불친절 하냐고 하시면 곤란하고요, 괜한 걸로 헷갈리는 건 저는 딱 질색이거든요.

자, 우리 간단한 CPU의 구조에 대해서 다루어 본적이 있지요?



뭐, 이런 구조였지요. Processor는 Decoder, CU, ALU, ACC, PC, Instruction Register, Address Register, Data Register등으로 구성되어 있고요. 이 그림을 ARM7TDMI 구조에 맞게 다시 자세하고 복잡하게 살펴보면 다음과 같고요, 이런 걸 ARM Architecture라는 이름으로 많이 소개가 되지요. (최근에 나온 더 새로운 ARM architecture도 많이 있지만, 실은 이런 기본적인 구조에서부터 출발해서 더 advance하게 만들어진 것들이니까 기본부터 착실히 한다고 생각하시면 더욱 기초가 튼실해 지고 좋겠죠)



〈 ARM7-TDMI의 아키텍쳐 〉

뭐 똑같지요? 저는 뭐 달라진 건 그다지 모르겠습니다만, 배치를 다시 하고 몇 가지가 늘어나긴 했지요? 아래 그림에서 Barrel Shifter라든가, Register Bank등의 새로운 것 빼고는 기본적인 구조는 같습니다. 여기에서 Register Bank란, 32Bit 크기의 범용 Register 31개, Status Register 6개로 한대 묶여 있는 Register의 묶음을 말하며, ARM core에서 임시 저장공간을 담당하고 있고요, 뭐, 말이 임시 저장공간이지, 어떤 Register는 특정 용도로 사용되기도 한다고 Register Section에서 이럴 줄 알고 수다를 떤 바가 있습니다.

Barrel Shifter는 Shift (왼쪽, 오른쪽), 또는 Rotation (오른쪽)을 실행할 32Bit WORD와 Shift 수를 입력 받아, Shift 수만큼 Shift한 32Bit결과를 출력하는 특별한 회로로서, 산술 연산을 시도할 때, 유용하게 사용됩니다. 일종의 Shift 가속기인 셈인 거죠.

나머지 block들은 Simplified CPU Model에서 다룬 그대로 입니다. 너무 어렵게 생각하면 안됩니다. 잠시 뚫어져라 이 그림을 쳐다보고 기억해 두면 더욱 좋습니다. 나머지 Control Logic에 그려져 있는 개미 같은 in/output은 Control Logic이 발생하는 control 신호와 동작을 하기 위한 Input이라고 보시면 됩니다. 어렵게 생각하면 안 된다는 사실.

ARM core에 세상에 발을 들여 놓는 다는 것은 마치 아이가 아빠랑 대중목욕탕에 들어서자마자, 뜨거운 물에 들어가는 것과 같습니다. 왜냐하면, 아이들은 뜨거운 물에 들어가기를 무서워하지요. 발끝에 몇 번이나 물을 축여 본 후에도 못 들어 가고, 주위만 맴돌고 있습니다. 하지만, 어른들은 곧바로 뜨거운 물에 들어가서, 심지어 놀랍게도 아무렇지 않은 얼굴로 "시원하다"는 매운 꼼장어 물에 씻어 먹는 멘트를 날리지요. 젠장 덜덜덜

ARM도 마찬가지로, 잘 모르는 사람에게는 참으로 희한한 세상이며, 어디서부터 익숙해 져야 할지 막막하지만, 아는 사람들에게는 그다지 어려운 세계는 아닌 것입니다. 지금부터 몇 가지만 익숙해 지면, ARM이라는 것도 별거 아니니까, 잠시 "화이삼"입니다.

가장 먼저 익숙해 져야 하는 것은 Mode, Register, 그리고 Exception 이에요. 중요한 개념이니까 곧 이어 나오는 section에서 더 자세히 다룰까 생각 중이죠. ARM이라는 것은 MCU의 CPU부분인 core만을 의미하며, 이외 I/O나 Timer, Interrupt같은 부분은 후에 더 자세히 다루는 것이 더욱 좋을 것 같습니다. ? 기대해 주세요 -

뭐, 다른 Core들도 그다지 다르지 않으니까, 널리 알려진 ARM7 Core 그림으로 모든걸 대신 하려고 하는데, 괜찮겠지요?

어째서 ARM이 저전력이며, 고성능 Processor라서 Embedded system에 적합하다는 말인가? Micro Processor는 흔히들 CISC (Complex Instruction Set Computer)와 RISC (Reduced Instruction Set Computer)로 나누는데 ARM은 이중 RISC범주에 속한다고 봐야 하지요. ARM의 약자가 바로 Advanced RISC Machine 아니겠습니까? CISC는 많은 수의 명령어와, 데이터 형태, 그리고 Addressing 기법들을 모두 채택하고 있습니다. 그렇기 때문에, Chip의 크기가 크며, 명령어가 복잡하고, Chip이 복잡하게 생겼습니다. 그러다 보니, 이런 많은 양의 기능을 누가 쓰겠냐 하며 저같이 머리 나쁜 사람을 위하여, 많이 사용되는 명령어, 데이터형태, Addressing기법 등 만을 모아 단순한 Micro processor를 만들게 되었지요. 그것이 바로 RISC입니다 RISC Microprocessor는 명령어 길이가 고정되어 있고, 그 종류가 많이 않고, 적은 수의 Addressing 기법이 그 특징이 되겠습니다. 그러다 보니, 자연스레 chip의 복잡도가 단순해지고, 크기도 작아지고, 전력소비도 줄어들게 된 거죠. 하지만, 모든 일에는 Trade off가 있듯이, RISC Microprocessor는 명령어들이 단순하여, CISC에서는 한 줄이면 해결 될 일들이, 번잡스럽게도 RISC에서는 몇 줄에 걸쳐서 같은 일을 해결해야 할 때도 발생하게 된 거에요. 뭐 Performance도 당연히 떨어지겠습니다만, Embedded system의 특성상 저전력이면서, 심플한 구조의 RISC구조가 더 적당하게 되어 버린 것이지요. 한가지 궁금 할 까봐, 코멘트 하나 더 하자면, CISC는 Micro Programming 방식으로, RISC는 Hard Wired 방식으로 디자인 된답니다. Hard wired 방식은 기계어 한 줄을 실행하기 위하여, 아예 그에 해당하는 논리회로를 구성하여 구현하는 방식입니다. 반면에 Micro Programming 방식은 기계어 하나를 실행하기 위하여, 그 밑에 많은 양의 작은 Micro Instruction들이 숨어서 실행되는 방식인 거죠.

by 히언 | 2009/06/02 20:39 | 마이크로프로세서 | 트랙백 | 핑백(1) | 덧글(2)