# 친절한 임베디드 시스템 개발자 되기 강좌 : Register 넌 누구냐

Register 넌 누구냐

Register 넌 도대체 누구냐. Register의 사전적 의미는 무엇인지 부터 알기 위해 내친구 네이버에게 물어보면 다음과 같은 답을 돌려 줍니다요. - Resistor와 Register는 전혀 다른 이야기니까, 헷갈리면 안됩니다. 저항과 기억소자, 정말 다릅니다요 -

reg·is·ter〔〕〔L「뒤로 나르다;기록하다」의 뜻에서〕 n.

1 등록[등기]부(= bok);(특정인의) 명부

a parish[church] register 교구[교회]의 교적부《출생·세례·결혼·사망 등이 기록된》

the electoral register 선거인 명부

2 (생사 등의 공적인) 기록, 등기, 등록;표, 목록

call the register 출석을 부르다

3 (속도·수량 등의) 자동 기록기, (금전) 등록기, 기록 표시기

4【음악】 성역(聲域), 음역(音域);음전(音栓), (오르간의) 스톱;스톱의 손잡이(stop knob)

5 선박 등록부;선적 증명서

ship's register (세관의) 선적(船籍) 증명서

6 (특히 난방의) 통풍[온도, 환기] 조절 장치

7【사진】 (합성 사진을 만들기 위한) 정합(整合)

8【인쇄】 안팎 인쇄면의 정합(整合)

9 등기 우편

10【언어】 언어 사용역(域)

11【컴퓨터】 레지스터《CPU가 적은 양의 데이터나 처리하는 동안의 중간 결과를 일시적으로 저장하기 위해 사용하는 고속의 기억회로》

12《주로 미》 등기관(登記官)

a register of wills《미》 유언 검증관- naver 사전

뭐 적당하게 그 공통점을 찾는다면 기록하다 정도 되겠습니다. 재미 있는 사실은 CPU가 적은양의 데이터나 처리하는 동안의 중간 결과를 일시적으로 저장하기 위해 사용하는 고속의 기억회로 라는 점인데, 약간은 확장, 수정할 필요가 있겠습니다.

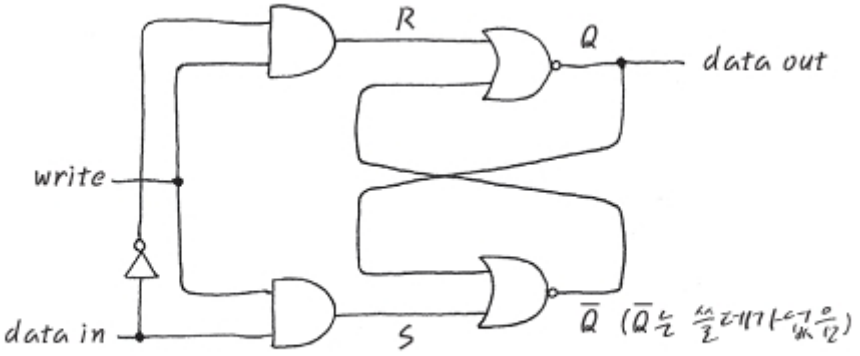
저는 이렇게 고치고 싶네요.Register는 Flip Flop의 집합이며, 이 Flip Flop이라는 것은 각각 1bit의 정보를 저장할 수 있는 것들을의미합니다. 결국 n-bit Register라는 것은 n bit의 정보를 저장할 수 있는 - 결국 n개의 Flip Flop으로 이루어진 - Flip Flop의 Group을 말하는 것입니다. 이런 의미에서 Register라는 것은 최소1 bit단위로 정보를 저장 또는 수정할 수 있습니다.

가끔 회로에 관한 이야기를 하다 보면 Latch라는 말들을 하기도 합니다. Latch란 1bit, 즉 1 또는 0인지를 기억할 수 있는 소자를 통칭하는 말인데, 그중 가장 대표적인 예가 Flip/ Flop이고요. Flip Flop은 속도가 빨라서 Register의 구현에 가장 많이 사용되지요.

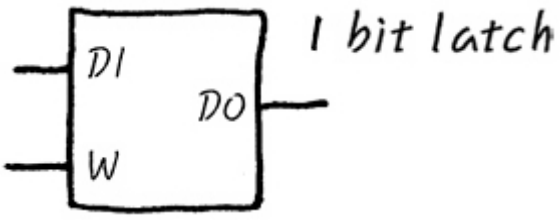
그러니까, 7 bit Latch나, 7 bit Register나 7개짜리 Flip Flop 한묶음이나, 매 한가지 이야기에요.

그리고, Register는 굳이 CPU만 연산중간이나 결과에 사용한다고 볼 수는 없고, CPU 내부에는 메모리 대신에 메모리보다 졸라 더 빠른 Register가 쓰이는 구나 정도로 이해를 하셔야 합니다. 왜냐하면, 그렇게 이해 하고 있으면 CPU외부에 Register는 없다고 생각을 가둘 수도 있으니까, 조금 위험합니다. CPU내부말고도, 그 외부에도 Register는 있을 수가 있습니다. 때에 따라서는 Memory도 이런 Register의 모음으로 구성할 때도 있습니다만. 용도에 따라 다른거니까 CPU만 Register를 사용할 수 있다는 편견은 버려주세요. 결국에는 고속이며 비싸니까 CPU내부에 몇개 안들어 있고 그러다 보니 CPU가 정말 필요로 하는 정보를 그때 그때 저장하고 버리다 보니 결과적으로 일시적인 기억장치처럼 정의가 내려진 것입니다.

음. 한가지 덧붙이자면, n-bit Register는 n개의 Latch로 이루어 지며, 이런 Register들의 output에는 간단한 data-processing operation을 할 수 있는 gate combination이 덧붙여 질 수도 있습니다. 이 의미는 어떤 특정한 Regsiter는 특정한 task를 수행하기 위해서 존재할 수도 있으며, 이 특정한 Register에 약속된 특정 값을 write 함 으로서 특정한 일을 시킬 수도 있음을 의미하지요. 이런 특징 때문에 Register를 이용한 Memory mapped I/O의 구현이 가능해 집니다. I/O를 다룰 때, 다시 한번 언급하겠지만 성미가 급한 관계로 먼저 말해 버렸으니 참으로 저의 참을성 회로에 문제가 있는 것 같습니다.
Register의 동작을 설명하기 위해서는 Flip Flop이라는 논리 회로를 이해 해야 합니다. Flip - 프라이팬을 홱 뒤집다, Flop - 벌렁 드러눕다. 라는 의미로 드러누웠다가 홱 뒤집혔다가 다시 드러두웠다가 하는 회로 입니다. 참으로 태평한 세상이군요. 저 같으면 sunday afternoon 회로라고 이름지었을 텐데, 아무래도 Engineer들이란 단순하기 짝이 없습니다. 그래도 행태로만 본다면 이름 참 잘 지었구나 하는 생각이 들 때도 있습니다만.
어쨌거나,가장 간단한 형태의  Flip Flop이라는 것은 실은 NOR gate 두개의 output을 서로의 input으로 다시 feed back 하는 형태로 생겼습니다. R-S (Reset Set) F/F라고 부르기도 합니다. 잘 보시면, Data In은 data input이고, write는 Write Enable이라고 해서 Write TRUE가 되어야 Data In을 위쪽 AND gate나 아래쪽 AND gate로 넣을 수가 있습니다.
Write 신호가 항상 TRUE라는 가정아래. R은 Reset이고, S는 Set이라는 이름을 붙여서 살펴 본다면, R = 1이고, S= 0 이면 R쪽의 NOR gate의 output은 무조건 0이 되며, R = 0이고, S = 1이면 R쪽의 NOR gate output은 무조건 1이 됩니다아아아아.
잘 생각해 보세요. R = 1인 case에는 같은 NOR gate의 다른 input과 상관 없이 OR해서 1이 output되며, 이것이 inverting되니까 0이 되겠죠? 복잡하게 생각마시고, 이렇게 간단하게 생각하면 편리합니다.

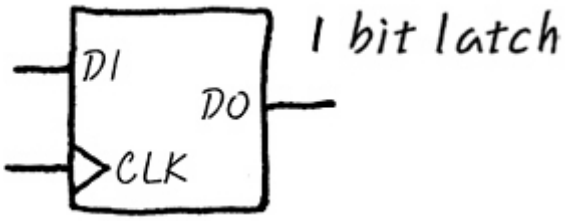



결국 R은 Data out에 대한 Reset, S는 Data out에 대한 set이라고 보면 되며, R과 S는 같은 값을 가져서는 안된다. 자 이런 식이라면, R과 S만 잘 control하면 Data out에 뭔가를 넣을 수 있겠습니다.
그러면 Data In은 S와 R에 들어갈 Data니까 S쪽에는 Data 그대로, R쪽에는 Data를 Invert해서 넣으면 되고더 중요한 것은 이런 Flip Flop 회로는 다음 input이 들어올 때 까지 Data Out을 유지하고 있다는 점입니다. 결국 Memory 기능을 가지고 있다는 뜻입니다요. 이런 Flip flop을 RS Flip flop라고 부르고요.
다음 input이라는 것은 Write 신호가 TRUE가 된 경우에만 Data In이 입력이 되고 D0에 기억 - 유지 하고 있어요.오호, 상당히 편리한 사실이군요. 사람들이 이런 신기한 기능을 놓칠리가 없습니다. 정말 기가 막히다니까요. 그래서 이런 1bit를 저장할 수 있는 Flip Flop 회로를 다음과 같이 한개의 기호로 나타내기 시작했습니다.

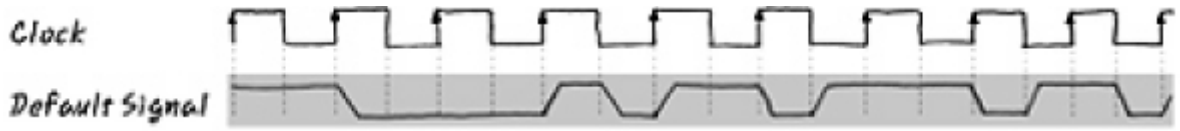



보시다 시피 DI는 Data In, DO는 Data out, W는 Write Enable 신호 입니다. 이런 Latch를 이용하여뭔가를 잠시 저장하는데 Latch들을 이용합니다. DO에 뭔가 Data processing 회로를 달아서 DO가 TRUE가 되는 순간 뭔

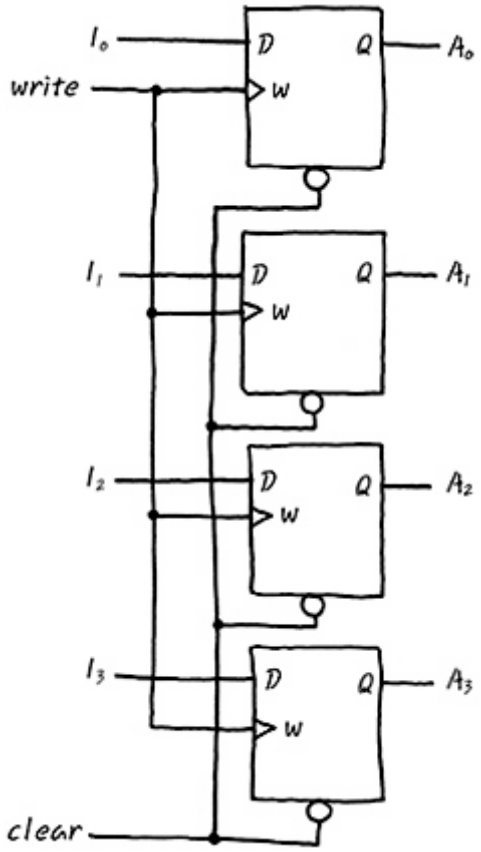
가 일을 할 수 있게도 하지요.
이런 걸 일컬어 Level Trigger latch라고 부릅니다. W가 High를 유지할 때, Write가 가능하고 그 외에는 값을 기억하고 있다는 뜻입니다. 왜 굳이 Level Trigger라고 부르느냐,
하면 Edge Trigger D type latch라는 것도 있으니까 그렇습니다.뭐 여하튼 이 Level Trigger Latch는 Write 신호가 High로 올라가 있는 동안 입력된 Data (DI)를
기억해 줍니다. Edge Trigger는 RS F/F두개를 묶어서 만드는데 그 구조까지 여기에 그림을 그릴려고 하니 머리가 지끈 지끈. 어쨌거나 Edge Trigger latch는 clk가 올라가는 순간이나, 내려가는 순간에만write가 가능하고 그것이 아닌 순간에는 그 전에 write된 값을 기억합니다.
이런 Flip Flop은 아래와 같이 표현되고요,



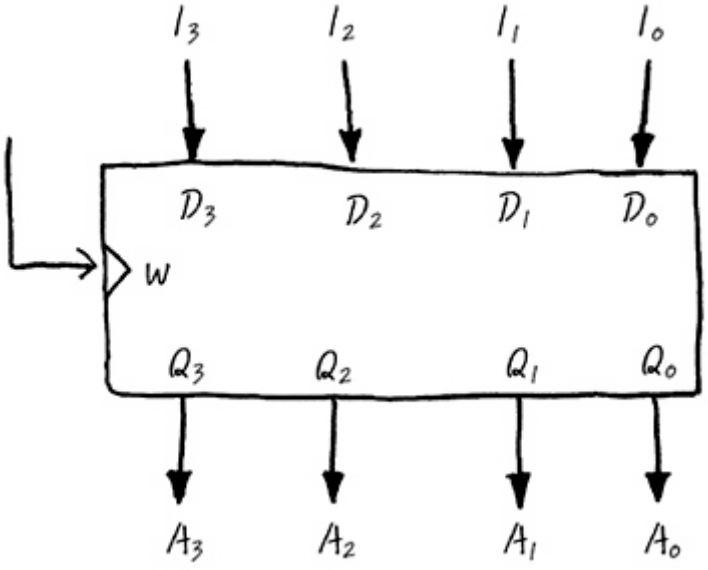

흔히들 Level Trigger Flip Flop을 Latch, Edge Trigger Flip Flop을 그냥 Flip Flop이라고부르는 경향이 있는데, 뭐 매한 가지 입니다. 또 어떨 때는 Latch는 Clock에 따른 동기화가 없는 녀석, Flip Flop은 Clock을 넣어서 동기화 하는 녀석 뭐 이렇게 나누기도 하지요.
CLK부분의 작은 삼각형은 Edge Trigger 방식이라는 뜻입니다. CLK이 0에서 1로 변할 때 값을 write하는 경우에는 rising edge trigger라고 부르고, CLK이 1에서 0으로 변할 때 값을 write하는 경우에는 falling edge trigger라고 부른답니다. Rising Edge Trigger는 아래 처럼 Clock이 0에서 1로 올라갈 때의 아래 열의 ㅓ default signal을 input으로 받아 드려서 Data를 저장 (유식한 말로는 Capture한다고 들 합디다) 합니다.




이런 1bit Latch를 엮어 n-bit Latch (n bit Register)를 만들기도 하는데, 그런건 아래 그림과 같이 여러개의 1bit Latch를 엮어서 만들어 냅니다. I0, I1, I2, I3는 모두 Input Data이며, A0, A1, A2, A3는 Output Data이며, Write 신호가 Edge일경우에만 Input Data로 Output들이Update되며 그 내용을 유지하고 있습니다.
Regsiter란 이런 Latch 여러개 또는 Flip flop 여러개를 엮어서 n bit로 만든 것을 말합니다. 그러니까 아래는 Write신호에 대해 Edge triggering 되는 4bit Latch 또는 4bit Register라고 보시면 됩니다.




이런 경우라면 4bit Latch 또는 4bit Register가 되겠습니다. 이런 Latch를아래처럼 그림 하나로 나타내기도 합니다. ^^ 역시나 W에 삼각형 표시가 있으니까Edge Trigger 방식이라고 봐야겠네요.




이런 Register의 저장 특성을 이용하여, MCP/ CPU내부에서 여러가지 용도로 이용합니다. I/O로 이용하기도 하고, CPU core 내의 임시 저장 공간으로 사용하기도 하고, CPU의 Register file로 이용하기도 합니다. 참 편리한 것이지요.

이런 의미에서 Register는 크게 2가지 종류로 나눌 수 있습니다. 그것들은 General Purpose Register, Special Purpose Register로 나눌 수 있습니다.

General Purpose Register -- Address Register : 외부메모리에 쓰거나 읽을 때, data가 들어 있는 주소를 가르키는 값을 넣어두는 Register- Data Register : 외부메모리에서 읽어온 값을 임시 저장하는 Register이며, 자세히는 Data input register, Data output register로 구성.- Instruction Pipeline Register : 외부 메모리에서 읽어온 Op-code (명령어)를 저장하는 Register Special Purpose Register -- Program Counter : 현재 실행되고 있는 주소를 가르키는 register- Stack pointer : 현재 사용하고 있는 stack영역에서 마지막에 데이터가 push된

곳의 주소를 가르키는 register- Linked Register : 방금까지 수행하다가 jump를 했을 경우, 돌아갈 곳의 주소를 가르키는 register (말이 좀 이상하니까, 이해하려 하지 말고, 그런게 있다 정도로만 지나가 주세요. 부탁)- Status Register : MCU의 현재 상태를 나타내는 Register. 예를 들면, 현재의 mode라던가, 계산의 결과값의 상태등을 저장.* 위에 언급한 Stack이라는걸 더 자세히 알고 싶으신 분은 Stack에 관한 소고등을 확인해 주세요. I/O Register

- I/O Register는 기억장소처럼 보이지만, 실제로 D Flip Flop으로 구성되어 방금 저장된 값을 기억하기도 하고, 밀어내는 역할 을 함. 위에서 언급한 Register들은 보통 MPU의 word값과 같은 크기를 갖지만, I/O Register들은 필요한 크기 만큼만 갖는다. 3bit Register도 있을 수 있고, 7bit Register도 있을 수 있음. 엄밀히 말하면 n bit Latch라고 말하는 게 정확함.

(Device contorl section의 I/O Register 부분에서 다시 다뤄요.)

I/O control을 위하여 Register를 setting한다는 것은 정확히 표현하자면, Latch에 값을 담는다고 보면 됩니다.

I/O Register는 기억장소처럼 보이지만, 실제로 D Flip Flop으로 구성되어 방금 저장된 값을 기억하기도 하고, 주변 장치로 그 값을 밀어내는 역할을 합니다. 보통의 Register들은보통 MPU의 word값과 같은 크기를 갖지만, I/O Register들은 필요한 크기 만큼만 갖습니다.3bit Register도 있을 수 있고, 7bit Register도 있을 수 있습니다. 그러니까 엄밀히 말하면 n bit Latch라고 말하는 게 정확하죠.

이런 I/O 제어 register들은 Readonly, Writeonly, Read-Write 중의 하나의 Property를 가지고 있으며 이런 특성에 맞추어서 Access가 이루어져야 합니다. Write 속성은 이제까지 보아왔듯이 저장 속상이고요, Read 속성은 Latch의 output 부분에따로 read할 수 있는 line이 연결되어 있어요.

Writeonly의 경우에 I/O register를 읽게 되면 실제 값과 다른 값이 읽히기도 합니다. 또한 Readonly, Read-Write의 경우에도 실제 할당은 8bit이지만 그것보다 더 작은 bit만 구현되어 있어서읽어보면 실제 값과 다르게 보일 수도 있으니, 필요한 bit를 잘 masking 해서 읽어야 합니다.

뭐, 장황하게 썼지만, General Purpose Register는 다음에 나올 CPU 동작 예에서 둘러볼 것이고,Special Purpose Register는 이후에 더 자세히 다룰 꺼니까, 너무 걱정 마세요.

너무 걱정되시는 분은 ARM inside부분을 먼져 읽으셔도 무방할꺼라 생각하고 있습니다. 지금은 뭐 적당히 특별한 경우에 사용되는 구나 정도로 패쓰~ 하셔도 됩니다. 하지만 아주 중요한 Register들이니, ARM session에서는 꼭 잘 알아두셔야합니다. 무슨 용도로 쓰이는지!

저장공간으로서의 Register는 Latch로 이루어져 있으며, 고속이며, 비싼 경향이 있습니다. 그러다보니, 많이 만들지 않고, MCU내부에만 두고, MCU의 외부일 수록 점점 더 속도보다는, 대용량이면서싼 저장 매체를 사용하게 되죠.

PC로 치자면, Register < RAM < Hard disk 정도지요.

더 자세한 CPU Design이나, 동작원리를 보고 알고 싶으시면, M. Morris Mano의 Computer system architecture를 보시면 자세히 나와있으니까, 읽어 보세요. 예를 든 그림도 이 책에서 따왔다니까요.

RS 플립플롭이 어떻게 동작하는지 자세히 알고 싶으시면~ http://www.youtube.com/watch?v=--pv3MZMoo0를 참조해 주세요. 아주 재밌게 해 놓았네요!