Chapter 1-2 (컴퓨터 구성)

Index

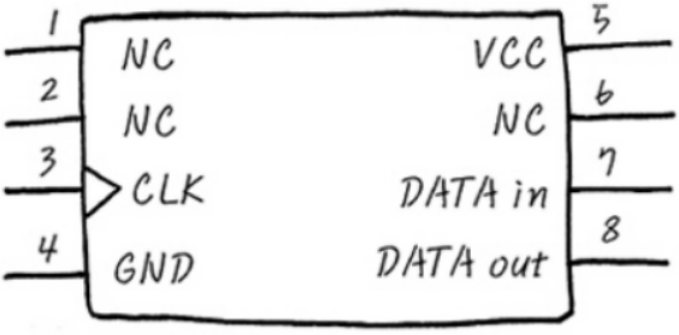
논리회로

- 디지털 신호를 input으로 넣었을 때 원하는 output을 만들어 내기 위해 논리적 순서에 의해 데이터를 manipulation(조작)하는 회로

- **NOT** : 입력의 반대를 출력
- **OR** : 입력의 한 개 이상이 1이라면 출력이 1
- **AND** : 입력이 모두 1이라면 출력이 1
- **XOR** : 홀수 개의 1이라면 출력이 1, 짝수 개의 1이라면 출력이 0
- **NOR** : 입력의 한 개 이상이 1이라면 출력이 0
- **NAND** : 입력이 모두 1이라면 출력이 0
- **XNOR** : XOR에 NOT을 연결한 것

IC(Intergrated Circuit)

- 수많은 논리회로가 합쳐져 하나의 package로 만든 chip



- **NC** : ‘No connection’의 약자로 사용하지 않은 pin
- **CLK** : clock trigger pin
- **GND** : Ground , **VCC** : 전원
- **DATA in/out** : I/O를 위한 pin

레지스터

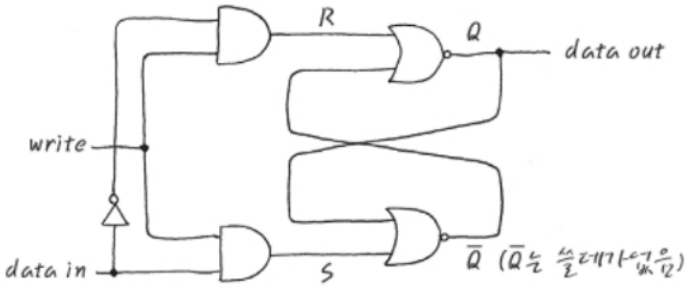
- CPU가 데이터를 처리하는 동안의 중간 결과를 일시적으로 저장하기 위해 사용되는 고속의 저장공간, 플립플롭의 집합

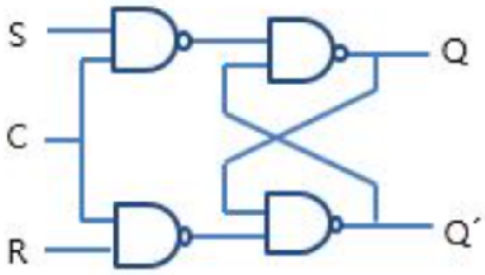
플립플롭(Flip flop, F/F’)

1-bit의 정보를 저장할 수 있는 기억소자

RS 플립플롭

- R=1, S=0 - Q=0 (Reset)
- R=0, S=1 - Q=1 (Set)
- R=0, S=0 - Q=저장된 값을 그대로 유지 (No change)





write 신호를 인가해서 플립플롭을 제어하는 걸 **‘level trigger flip flop**’ 또는 ‘**latch**’

clock 신호를 인가해서 CPU 동작 타이밍에 맞춰 제어하는 걸 ‘**edge trigger flip flop**’ 또는 ‘**flip flop**’

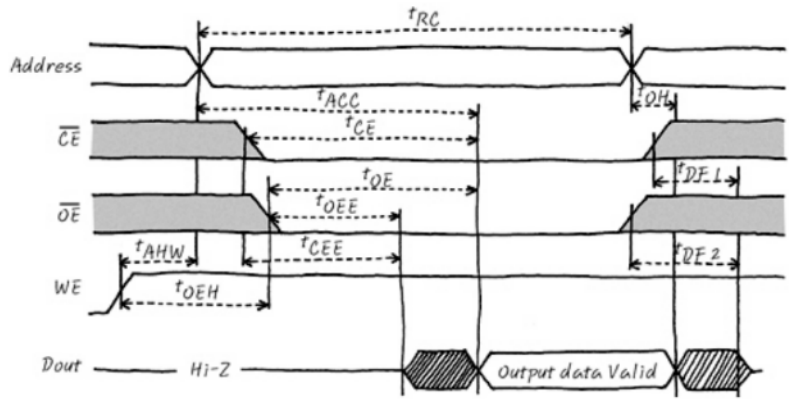
- **General purpose register**
  - Address register - 메모리에 R/W할 때 데이터가 들어 있는 주소를 임시 저장하는 레지스터
  - Data register - 메모리에 R/W할 때 쓰려는 값 또는 읽은 값을 임시 저장하는 레지스터
  - Instruction register - 메모리에서 읽어온 명령어를 저장하는 레지스터
- **Special purpose register**
  - **Program counter(PC)** - 현재 실행되고 있는 주소를 가리키는 레지스터
  - **Stack pointer(SP)** - 사용중인 stack의 최상단 주소를 가리키는 레지스터
  - **Linked register(LR)** - 복귀할 주소를 가리키는 레지스터
  - **Status register(SR)** - 시스템/서브시스템의 현재 상태를 저장하고 알려주는 레지스터
- **I/O register**

디지털 회로는 논리회로들의 집합인 ‘conbinational 회로’와 플립플롭 같은 기억소자들의 집합인 ‘sequential 회로’로 구성

동기화(Synchronization)는 1) 박자를 맞추다, 2) 순서를 맞추다

신호가 switching될 때 완전한 low 또는 high 신호가 되기까지의 10~90%까지 소요되는 시간을 전달소요시간(**delay**)라고 하며, 이에 관한 전기적 특성을 **스위칭 특성**

전체 시스템 속도는 가장 느린 소자의 속도에 맞춰야 오작동없이 작동한다.



SYMBOL	PARAMETER	MIN	MAX
t_{RC}	Read Cycle Time	65	
t_{ACC}	Address Access Time		65
t_{CE}	$\overline{CE}$ Access Time		65
t_{OE}	$\overline{OE}$ Access Time		25
t_{CEE}	$\overline{CE}$ to output Low-Z	0	
t_{OEE}	$\overline{OE}$ to output Low-Z	0	
t_{OH}	Output Date Hold Time	0	
t_{DF1}	$\overline{CE}$ to output High-Z		25
t_{DF2}	$\overline{OE}$ to output High-Z		25

색깔이 어두운 부분 : 어떤 값이든 가질 수 있고 시스템 동작에 영향을 미치지 않은 부분

High impedance(Hi-z) : 아무 유효한 신호가 없는 상태

t_CE : CE(Chip Enable) pin에 low active 신호가 인가된 뒤 유효한 output data가 나오기까지의 시간

t_RC : read cycle time, 최소 저만큼의 시간동안 address line에 주소값을 대고 있어야 제대로 된 data를 얻을 수 있다는 뜻

Bus

- 장치들이 정보 공유를 위해서 공유하는 선들의 집합

아비터(Arbiter)

- bus 점유권을 중재하며 bus 사용권을 결정하는 것 - CU(Control Unit)

Memory

- **MCP(Multi-chip Package)** - PSRAM+NOR, SDRAM+NAND
    - **SRAM(Static RAM)** : 1개 cell이 트랜지스터 6개로 구성된 가장 비싼 RAM

    - **DRAM(Dynamic RAM)** : 1개 cell이 트랜지스터 1개에 캐패시터 1개로 구성된 값싼 RAM, SRAM에 비해 회로가 단순, 일정 시간마다 refresh가 필요, read할 때마다 precharge도 필요
        - **DDR SDRAM(Double Data Rate Synchronous DRAM)** : CPU 동작에 박자와 순서를 맞춰가며 동작해 CPU 성능을 최대한으로 활용하는 DRAM, data를 2배로 빨리 전송할 수 있는 메모리
    - **PSRAM(Pseudo SRAM)** : 구조적으로는 DRAM, HW적으로 SRAM처럼 쓸 수 있는 메모리
    - **NOR Flash** : cell이 병렬로 연결 , address line과 data line을 모두 갖고 있어 byte 단위로 random access가 가능한 메모리
    - **NAND Flash** : cell이 직렬로 연결, address line과 data line이 없어 집적도가 높고 page단위로 read/write이 가능한 메모리
- **XIP(Execte In-place)** : 메모리 상에서 직접 프로그램을 실행할 수 있는 기술

RAM

- address pin(A0~A7)과 data pin(D0~D7) 그리고 control pin(RD, WR)으로 이뤄져 있다.
- 읽을 때는 RD에 high, 쓸 때는 WD에 high

CPU

- 논리회로의 집합체, 약속된 신호를 주면 약속된 동작을 수행하는 단순한 원리
- CU(Control Unit), Decoder, ALU(Arithmetic & Logical Unit), Register set등으로 구성
- decoder에서 명령어 해석 → CU로 각종 제어 신호를 발생 → ALU 등에게 동작을 명령, 저장하는 용도로 레지스터 활용

MCU(Micro Controller Unit)

- CPU 이외에 여러가지 기능이 한 개의 chip에 내장된 것

ARM core CPU의 동작 구조 : fetch → decode → execute

```
word a = 1;
word b = 2;
word c;

void add() {
    c = a + b;
    return;
}
```

```
주소        어셈블리
0x1000      LOAD 0x2000   ; a를 data register에 load
0x1002      ADD 0x2002    ; Data register에 저장된 값(a)과 b를 더해 data register에 저장.
0x1004      STORE 0x2004  ; Data register에 저장된 값(a + b)를 c에 저장.
```

LOAD 0x2000

1. PC는 0x1000번지를 가리킨다.
2. PC값이 ‘address register’에 저장된 뒤 해당 주소에 접근
3. **해당 주소에서 명령어를 읽어와 ‘instruction register’에 저장 (Fetch)**
4. **Decoder가 명령어를 해석함**과 동시에 PC는 증가해 0x1002번지를 가리킨다. **(Decode)**
5. CU는 0x2000번지에 있는 데이터를 읽어오라고 제어신호를 발생
6. 0x2000번지에 있는 **전역변수 a의 값1이 ‘data register’에 저장 (Exeute)**
7. 이 값은 ALU를 통해 연산할지도 모르니 ACC(Accumulator)에 임시 저장

ADD 0x2002

1. ‘address register’에 저장된 PC가 가리키는 주소에서 명령어를 읽어와 ‘instrunction register’에 저장
2. Decoder가 명령어를 해석함과 동시에 PC는 증가해 0x1004번지를 가리킨다.
3. 0x2002번지의 데이터를 ‘data register’에 저장한 뒤 ACC값과 덧셈한 뒤 ACC에 저장

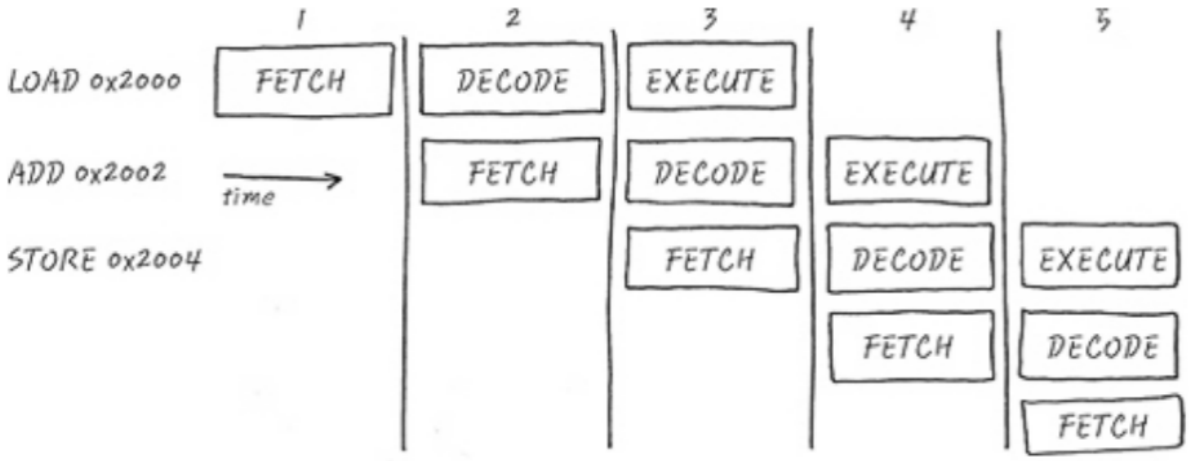
STORE 0x2004

1. ‘address register’에 저장된 PC가 가리키는 주소에서 명령어를 읽어와 ‘instrunction register’에 저장
2. Decoder가 명령어를 해석함과 동시에 PC는 증가해 0x1006번지를 가리킨다.
3. CU는 ACC값을 0x2004번지에 저장하라고 제어신호를 날려 결과값을 저장

Fetch-PC, address register, instrunction register

Decode-decoder

Execute-CU, data register, ALU, ACC



파이프라인

- 각 단계를 중첩시키거나 하나의 시간에 여러 단계를 수행하도록 하는 기법

PC값은 execute 단계의 2개 명령어 밑을 가리킨다.