EliceAcademy

실전 압축 컴퓨터 개론

01 컴퓨터 구조

목차

컴퓨터가 동작하는
원리를 알아봅니다.

목차

컴퓨터가 동작하는
원리를 알아봅니다.

EliceAcademy

01

컴퓨터에 대해 알아야 하는 이유

컴퓨터를 알아야 하는 이유

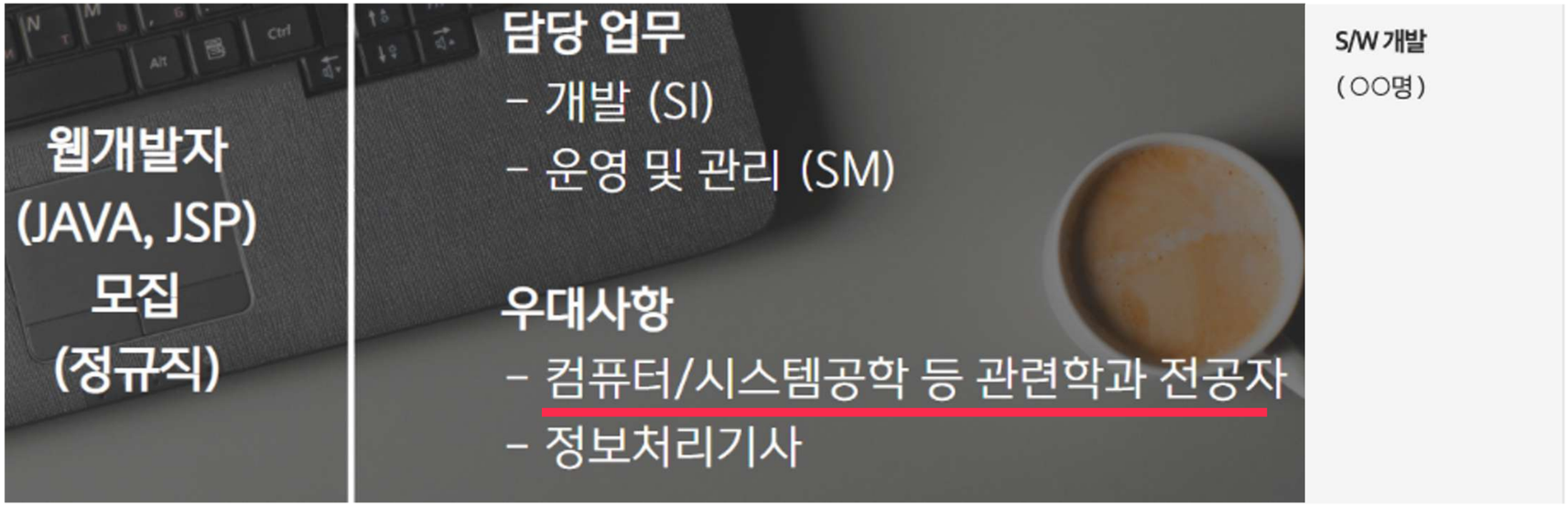
왜 컴퓨터에 대해 자세히 알아야 할까요?

컴퓨터를 알아야 하는 이유

컴퓨터로 코딩하니까

EliceAcademy

컴퓨터를 알아야 하는 이유



S/W 개발

(○○명)

담당업무

· 차량용 소프트웨어 개발

자격요건

· 학력 : 고졸이상 (졸업예정자 가능)

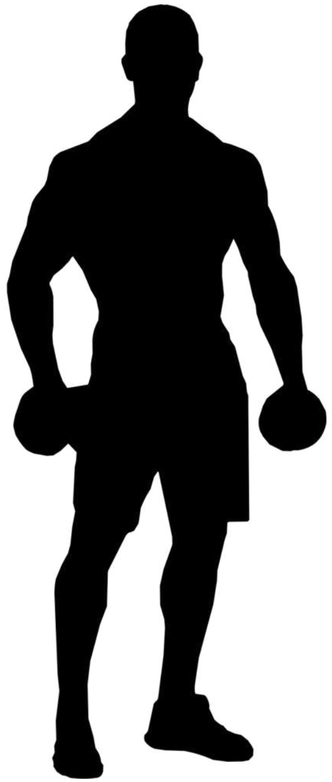
· 신입 ~ 경력 2년 이내.

우대사항

· 컴퓨터 관련 전공자.

· C/ C++ 활용 가능자.

반드시 알아야 할까?



운동을 할 때 다른 사람의 동작을 보고 따라할 수는 있겠지만,
우리의 몸에 대해 이해하고 있어야 효과적인 운동을 할 수 있음

컴퓨터를 몰라도 코딩은 할 수 있겠지만,
코드를 작성하는 컴퓨터 환경에 대해 이해하고 있어야
효과적인 코드를 작성할 수 있음

반드시 알아야 할까?

컴퓨터의 능력은 제한적

제한된 환경에서 효율적으로 동작하는 프로그램을 만들어야 함

내 컴퓨터가 뛰어나더라도

프로그램을 사용하는 사람의 컴퓨터는 그렇지 않을 수 있음

반드시 알아야 할까?

컴퓨터의 환경, 즉 **컴퓨터의 구조**를 알아야

좋은 개발자가 될 수 있음

개발 공부를 하면서 맞닥뜨릴 수 있는 낯선 언어들을

이번 기회에 미리 학습해 봄

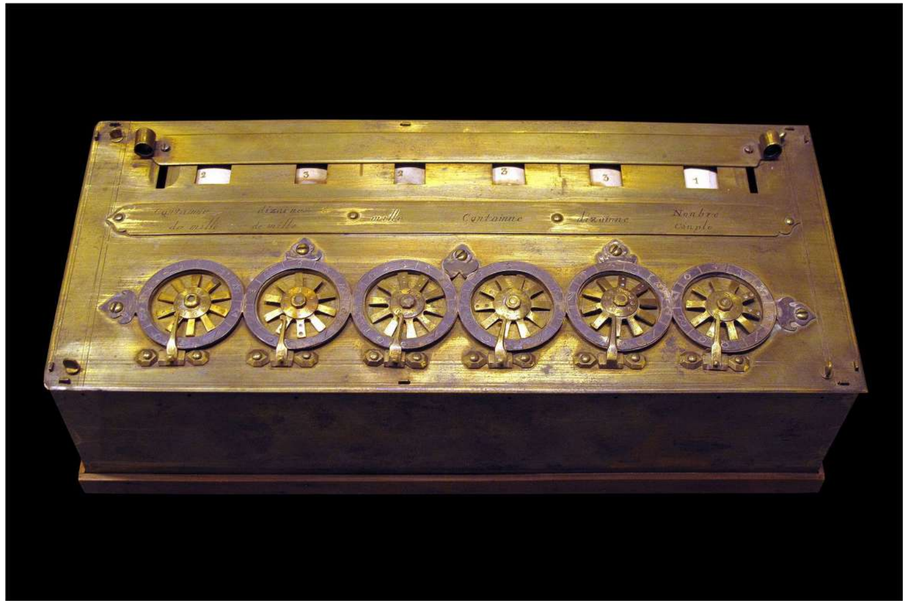
02

컴퓨터 동작 방식

컴퓨터

Compute(계산하다) + er

계산기



숫자를 계산해주는 기계

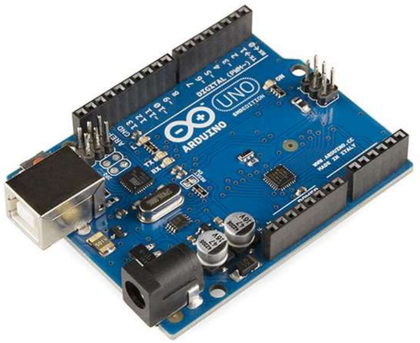
https://en.wikipedia.org

EliceAcademy

컴퓨터 종류

데스크톱 컴퓨터

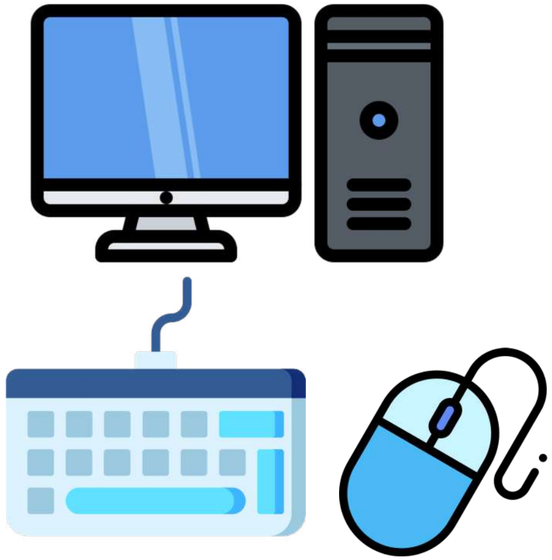
스마트폰



아두이노

https://en.wikipedia.org

컴퓨터 동작 방식



키보드와 마우스 같은 장치로 **명령어**를 전달하여 **데이터**가 처리됨

https://www.flaticon.com/

컴퓨터 동작 방식

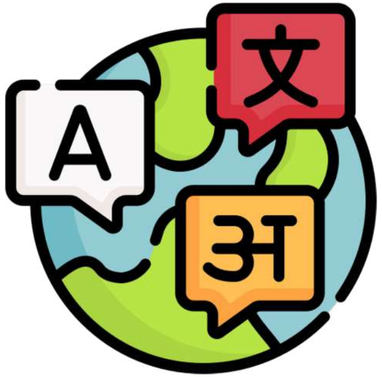
엑셀 문서를 열라는 **명령어**를 전달하면,

컴퓨터에 저장되어 있던 엑셀 문서 **데이터**를 찾아서 모니터에서 출력

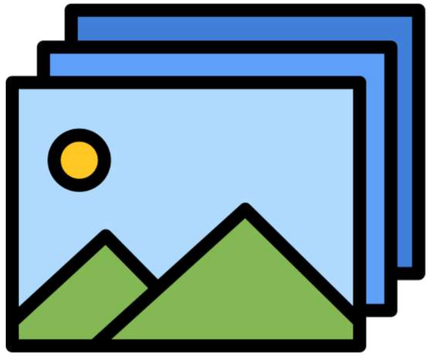
EliceAcademy

데이터

숫자



문자



이미지

https://www.flaticon.com/

EliceAcademy

03

데이터와 명령어

컴퓨터 동작 방식

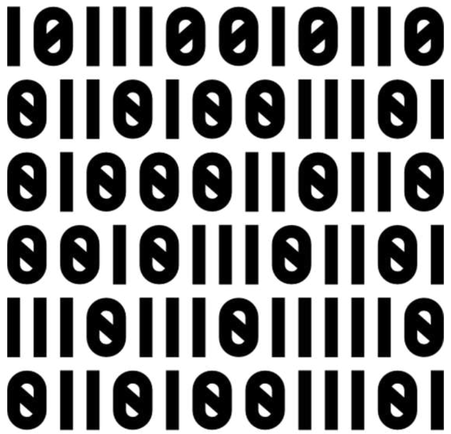
컴퓨터는 **데이터**와 **명령어**로 동작함

과연 "엑셀 문서를 열어"라는 명령어는 어떻게 알아듣는 것일까?

https://www.flaticon.com/

EliceAcademy

기계어



컴퓨터는 0과 1(이진수) 밖에 알아듣지 못함

이를 **기계어**라고 함

용어 해설

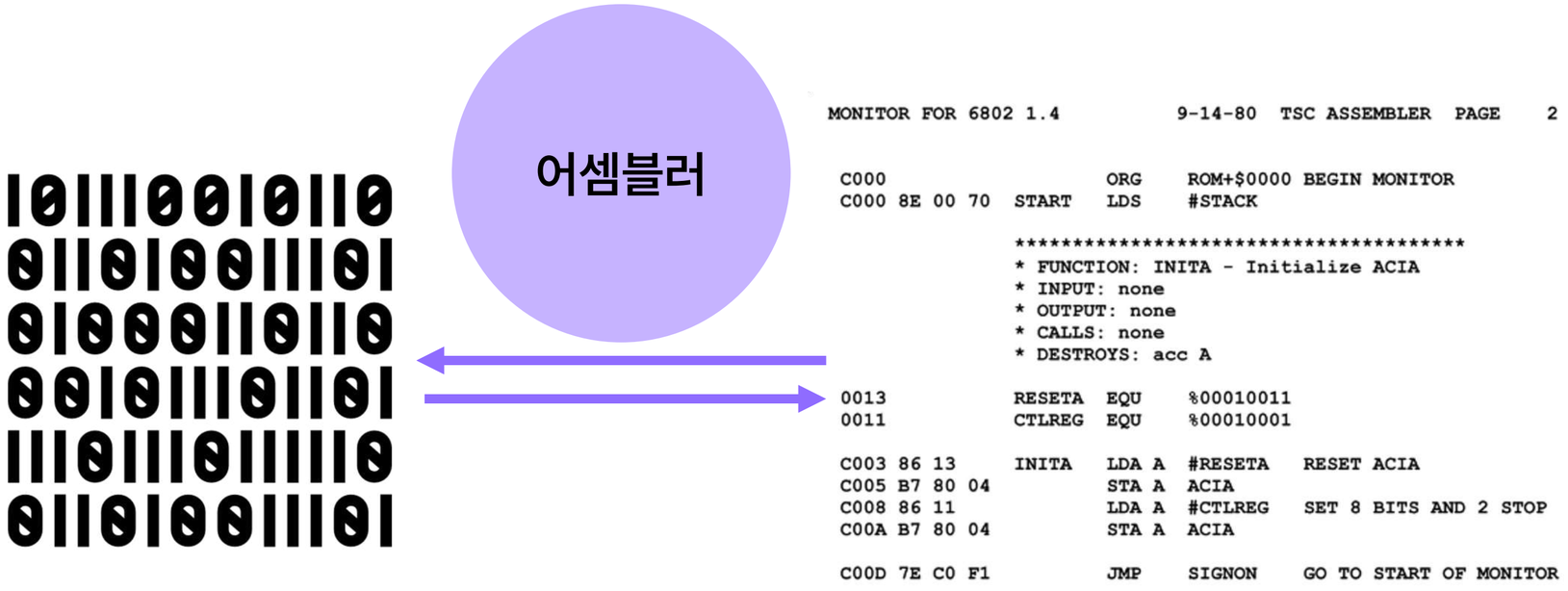
이진수

0과 1, 두 개의 숫자만으로 나타내는 수 체계

https://www.flaticon.com/

EliceAcademy

어셈블리어



컴퓨터가 사용하는 기계어와 일대일로 대응되는 프로그래밍 언어

어셈블리어를 기계어로 번역해주는 것을 어셈블러라고 함

https://en.wikipedia.org

EliceAcademy

고급 언어

```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2


 C000                   ORG    ROM+$0000 BEGIN MONITOR
 C000 8E 00 70  START   LDS    #STACK

                ***************************************
                * FUNCTION: INITA - Initialize ACIA
                * INPUT: none
                * OUTPUT: none
                * CALLS: none
                * DESTROYS: acc A

 0013           RESETA  EQU    %00010011
 0011           CTLREG  EQU    %00010001

 C003 86 13     INITA   LDA A  #RESETA   RESET ACIA
 C005 B7 80 04          STA A  ACIA
 C008 86 11             LDA A  #CTLREG   SET 8 BITS AND 2 STOP
 C00A B7 80 04          STA A  ACIA

 C00D 7E C0 F1          JMP    SIGNON    GO TO START OF MONITOR
```

컴파일러,
인터프리터

```
def add5(x):
   return x+5

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print '    %s [label="%s' % (nodename, label),
    if isinstance(ast[1], str):
       if ast[1].strip():
          print '= %s"];' % ast[1]
       else:
          print '"]'
    else:
       print '"];'
       children = []
       for n, child in enumerate(ast[1:]):
          children.append(dotwrite(child))
       print '    %s -> {' % nodename,
       for name in children:
          print '%s' % name,
```

사람이 이해하기 쉽게 작성된 프로그래밍 언어

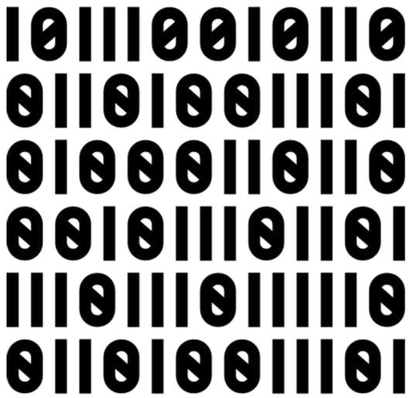
마찬가지로 번역 역할을 하는 **컴파일러**, **인터프리터**가 있음

https://en.wikipedia.org

EliceAcademy

컴퓨터 언어

기계어



어셈블리어

```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2


C000                    ORG     ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START    LDS     #STACK

                ***************************************
                * FUNCTION: INITA - Initialize ACIA
                * INPUT: none
                * OUTPUT: none
                * CALLS: none
                * DESTROYS: acc A

0013            RESETA  EQU     %00010011
0011            CTLREG  EQU     %00010001

C003 86 13      INITA   LDA A   #RESETA   RESET ACIA
C005 B7 80 04           STA A   ACIA
C008 86 11              LDA A   #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A   ACIA

C00D 7E C0 F1           JMP     SIGNON    GO TO START OF MONITOR
```

고급 언어

```
def add5(x):
   return x+5

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print '    %s [label="%s' % (nodename, label),
    if isinstance(ast[1], str):
       if ast[1].strip():
          print '= %s"];' % ast[1]
       else:
          print '"]'
    else:
       print '"];'
       children = []
       for n, child in enumerate(ast[1:]):
          children.append(dotwrite(child))
       print '    %s -> {' % nodename,
       for name in children:
          print '%s' % name,
```

컴퓨터는 기계어를 통해 **데이터**와 **명령어**를 처리함

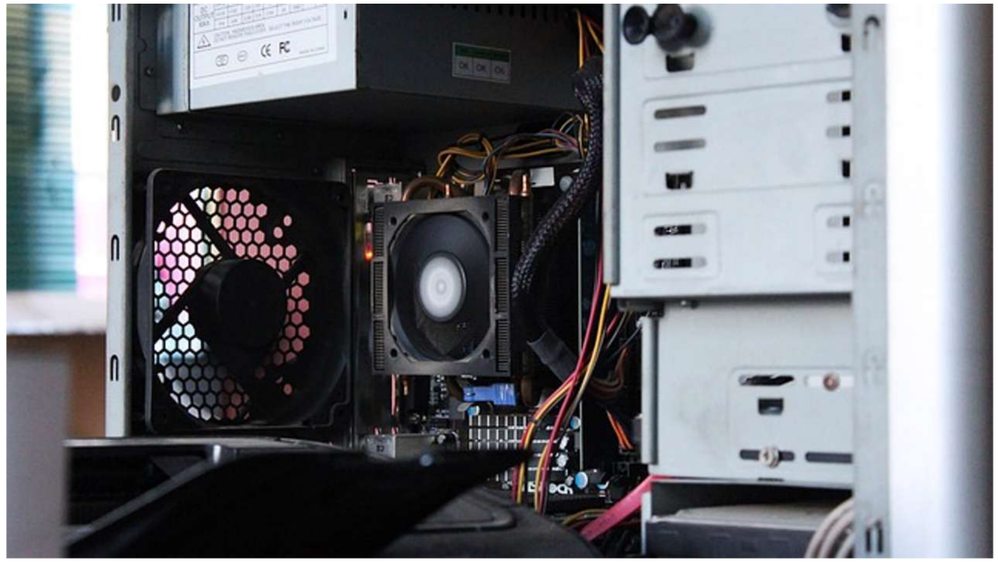
사람은 기계어를 이용하기 위해 **프로그래밍 언어**를 사용함

https://en.wikipedia.org

04

메모리

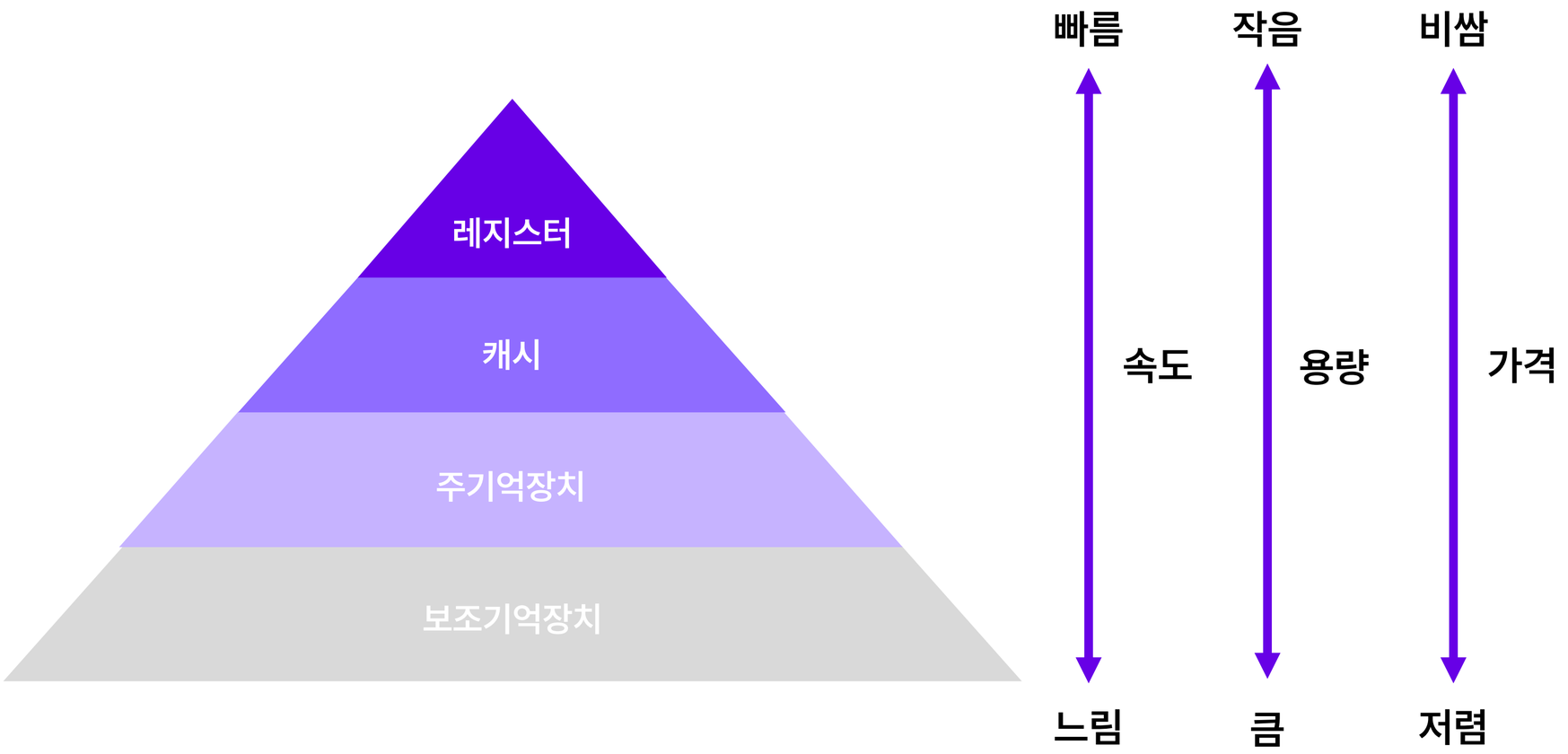
컴퓨터 구성



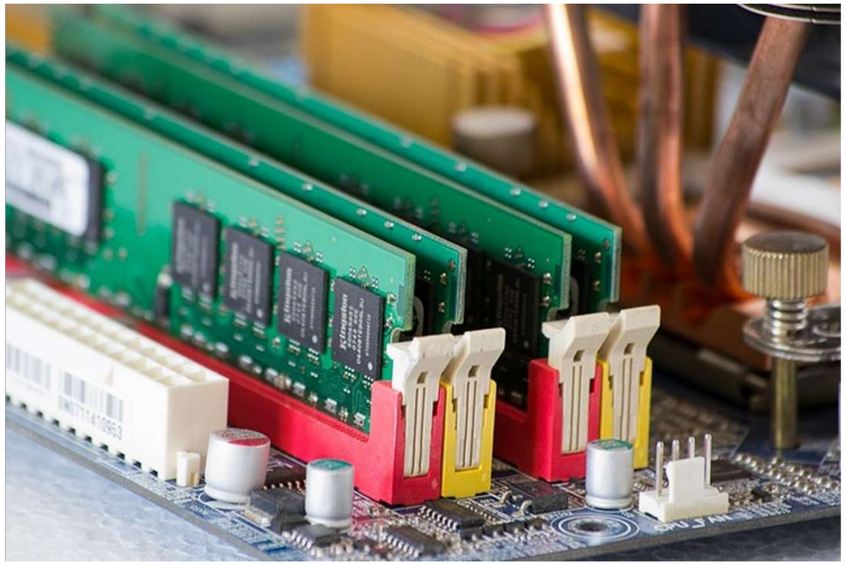
컴퓨터를 구성하는 핵심 부품으로 CPU, 주기억장치, 보조기억장치, 입출력장치가 있음

EliceAcademy

기억장치 종류



주기억장치



현재 실행되고 있는 데이터와 명령어를 **저장**

크게 **ROM**, **RAM**으로 분류

ROM과 RAM

ROM(Read-only Memory)	RAM(Random Access Memory)
읽기만 가능	읽기 쓰기가 자유롭게 가능
전원이 꺼져도 데이터가 사라지지 않는 **비휘발성** 메모리	전원이 꺼지면 데이터가 사라지는 **휘발성** 메모리
처리 속도가 느림	처리 속도가 빠름
가격이 쌈	가격이 비쌈

ROM과 RAM

ROM

읽기 전용 메모리라고 불리며, 컴퓨터를 구동하기 위한 기본적인 데이터들이 담겨있다.

전원을 꺼도 데이터가 지워지지 않는다.

RAM

랜덤 액세스 메모리라고 불리며, 시스템의 단기 데이터 스토리지로 데이터에 빠르게 액세스할 수 있도록 컴퓨터가 실시간으로 사용하는 정보를 저장한다.

많은 애플리케이션을 실행할수록 더 많은 메모리가 필요하다.

ROM과 RAM

ROM

Mask ROM (Mask Read Only Memory)

: 제조 공정에서 데이터를 기록하여 데이터를 읽기만 가능

PROM (Programmable ROM)

: 사용자가 데이터를 한 번 기록할 수 있음

EPROM (Erasable PROM)

: 저장되어 있는 데이터를 지우고 쓸 수 있음

EEPROM (Electrically EPROM)

: 전기를 통해 데이터를 지우고 쓸 수 있음

RAM

SRAM (Static RAM)

: 전원이 공급되는 동안만 내용을 저장

DRAM (Dynamic RAM)

: 커패시터(capacitor)를 이용해 전하를 축적하여 전원이 끊겨도 내용을 저장

SDRAM (Synchronous DRAM)

: 시스템의 클럭 속도와 동기화하여 빠르고 효율적으로 동작

메모리 주소

주소	데이터
1번지	10110010
2번지	01000100
...	...
100번지	01100011
101번지	10110101
102번지	11110000
...	...

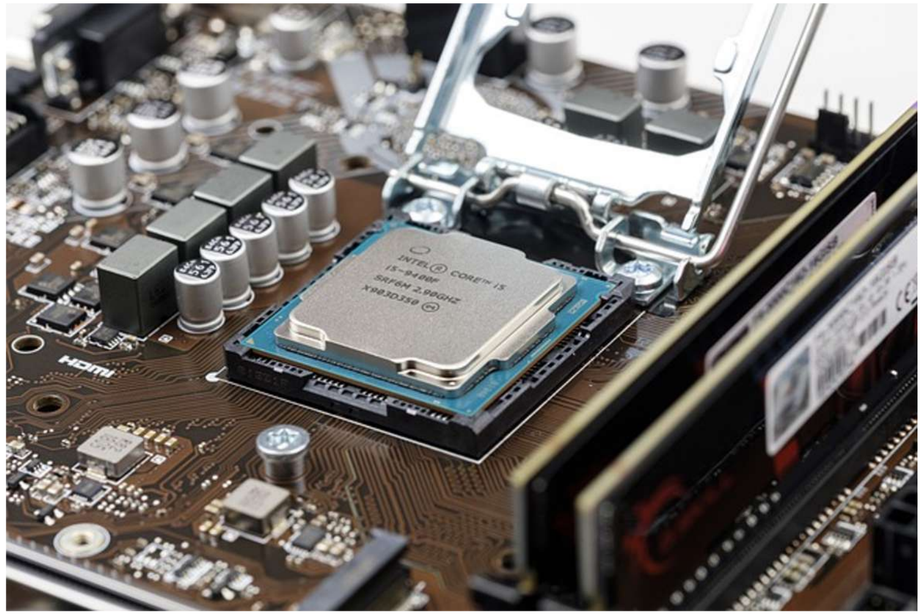
컴퓨터는 메모리에 저장된 데이터를 주소를 참조해서 데이터를 식별함

EliceAcademy

05

CPU

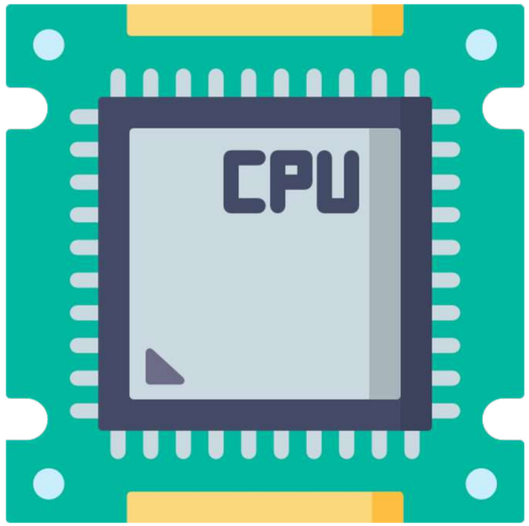
Central Processing Unit



프로세서(processor)라고도 함, 데이터와 명령어를 **처리**하는 제어 장치

클럭과 코어가 CPU의 성능을 결정함

CPU 구성



- 산술 논리 연산 장치(Arithmetic and Logical Unit)
- 레지스터(Register)
- 제어 장치(Control Unit)

논리 연산

참과 거짓에 대한 두 가지 값으로 수행하는 연산으로

논리곱(AND), 논리합(OR), 부정(NOT) 등이 있음

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

AND 연산

A	B	A ∨ B
0	0	0
0	1	1
1	0	1
1	1	1

OR 연산

산술 논리 연산 장치

다양한 산술 연산을 계산하는 디지털 회로

- **논리 회로로 구성된 다양한 연산 장치**
  - 가산기(Adder): 덧셈 연산을 수행하는 논리 회로
  - 보수기(Complementor): 2의 보수를 계산해주는 논리 회로

- **레지스터**
  - 누산기(Accumulator): 계산 결과가 저장
  - 상태 레지스터(Status Register): 연산 결과 나오는 오버플로우나 부호, 캐리 등의 상태를 저장
  - 인덱스 레지스터(Index Register): 데이터가 저장되어 있는 주소 저장

용어 해설

논리 회로

불 대수(Boolean algebra)를 이용한 입력 값에 대한 논리 연산을 수행하여 출력하는 것

제어 장치

처리할 명령어를 해독 및 실행하며 CPU의 전체적인 흐름을 제어

- 명령어 레지스터(Instruction Register): 처리할 명령어를 기억하는 레지스터
- 디코더(Decoder): 명령어 레지스터로부터 받은 명령어를 해석하는 명령어 해독기
- 메모리 주소 레지스터(Memory Address Register): 기억장치에 저장된 데이터의 주소를 임시로 저장하는 레지스터
- 메모리 버퍼 레지스터(Memory Buffer Register): 기억장치에서 읽어온 데이터 자체를 임시로 저장하는 레지스터
- 프로그램 카운터(Program Counter): 다음에 실행할 명령어 주소를 기억하는 레지스터

레지스터

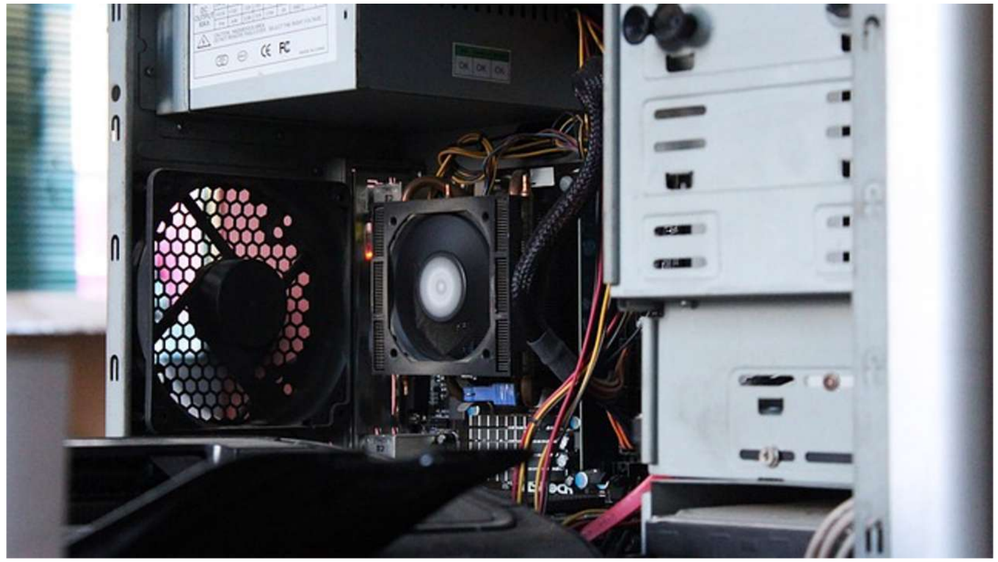
CPU가 처리할 데이터를 일시적으로 보관하는 기억 장치

- 입출력 주소 레지스터(I/O Address Register): 입출력 장치의 주소를 지정
- 입출력 버퍼 레지스터(I/O Buffer Register): 입출력 장치와 CPU 간의 데이터 교환을 위해 사용
- 스택 포인터(Stack Pointer): 프로그램의 실행을 관리하는 스택의 흐름을 제어

06

기타 장치

기타 장치



컴퓨터를 구성하는 장치 중 주기억장치와 CPU 외에도 입출력 장치와 보조기억장치도 있음

EliceAcademy

입출력장치

사용자가 원하는 데이터와 명령어를 컴퓨터에 전달하는 **입력 장치**와

컴퓨터가 사용자에게 결과를 보여주기 위한 **출력 장치**

EliceAcademy

입출력장치

조이스틱 (입력 장치)

스피커 (출력 장치)

프린터 (출력장치)

EliceAcademy

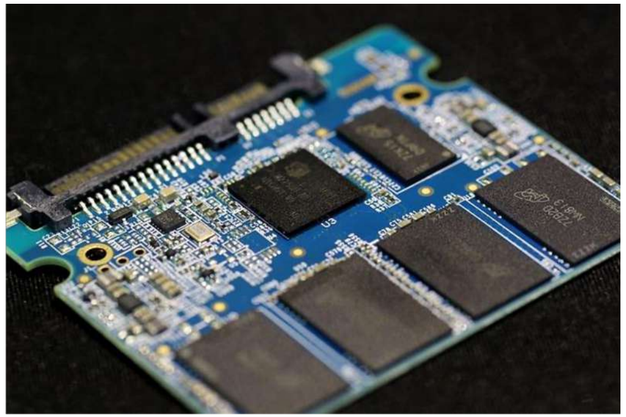
보조기억장치

전원이 꺼지면 데이터가 사라지는 주기억장치의 단점을 보완한 기억장치

용량이 크고 주기억장치에 비해 상대적으로 저렴하지만, 속도가 느림

EliceAcademy

보조기억장치



SSD(Solid State Driver)

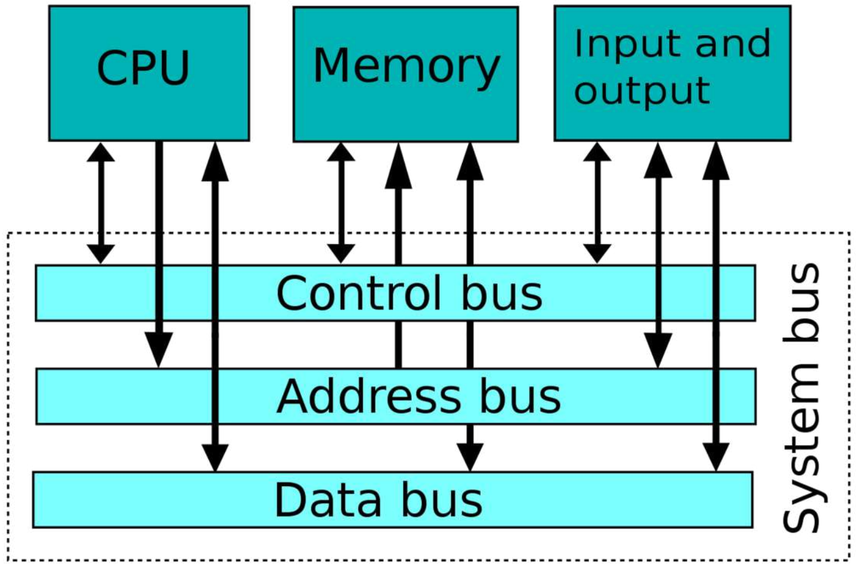
CD-ROM

USB 플래시 드라이브

07

컴퓨터 동작 흐름

시스템 버스



컴퓨터 시스템 내에서 신호를 전송하는 경로

CPU, 메모리, 입출력장치 등 모든 장치는 시스템 버스를 통해 데이터를 주고 받음

https://en.wikipedia.org/wiki/System_bus

시스템 버스 종류

제어 버스 (Control bus)

제어 신호 전송에 사용

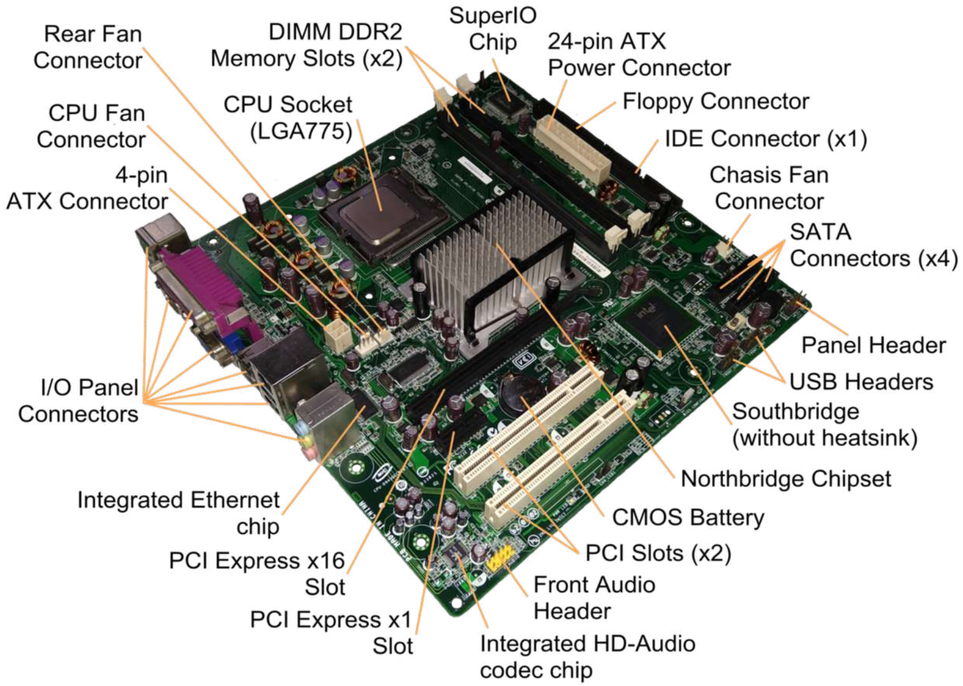
주소 버스 (Address bus)

메모리나 입출력장치의 주소 전송에 사용

데이터 버스 (Data bus)

데이터 전송에 사용

메인보드

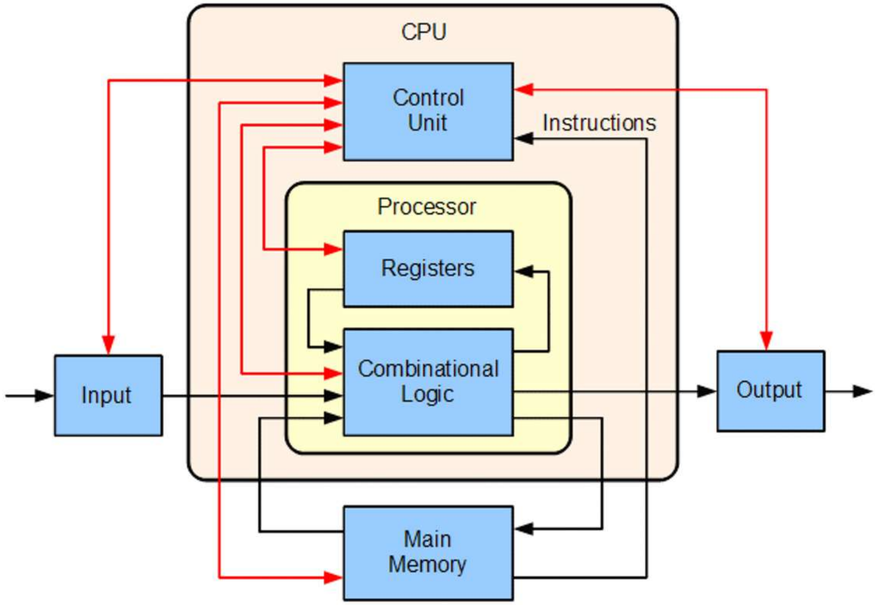


컴퓨터 장치들은 메인보드에 있는 시스템 버스를 통해 서로 연결되어 있음

다른 말로 마더보드라고도 함

https://ko.wikipedia.org/wiki/메인보드

전체 동작 흐름



1. 입력 장치로 데이터와 명령어를 입력 받음
2. CPU에서 레지스터 및 산술 논리 연산 장치를 통해 명령어를 처리
3. 처리된 결과는 메모리에 저장하거나 출력 장치를 통해 출력

https://ko.wikipedia.org/wiki/중앙_처리_장치#클럭

CPU 동작 흐름