
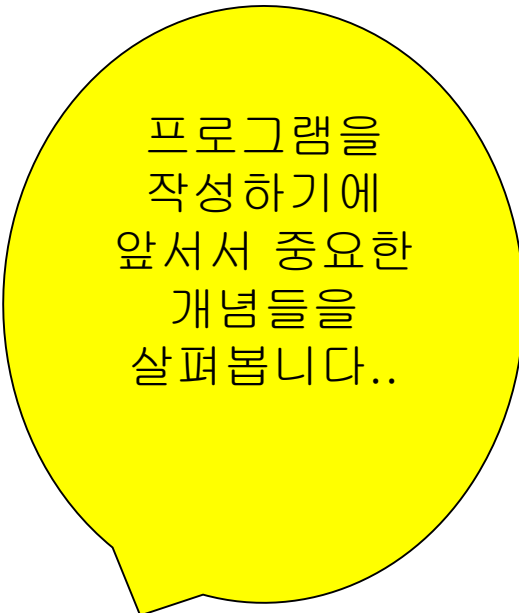


# 제 1장 프로그래밍의 개념

# 이번 장에서 학습할 내용

- 
- 프로그래밍 개념
  - 프로그래밍 언어
  - 알고리즘
  - 스크래치



프로그램을  
작성하기에  
앞서서 중요한  
개념들을  
살펴봅니다..



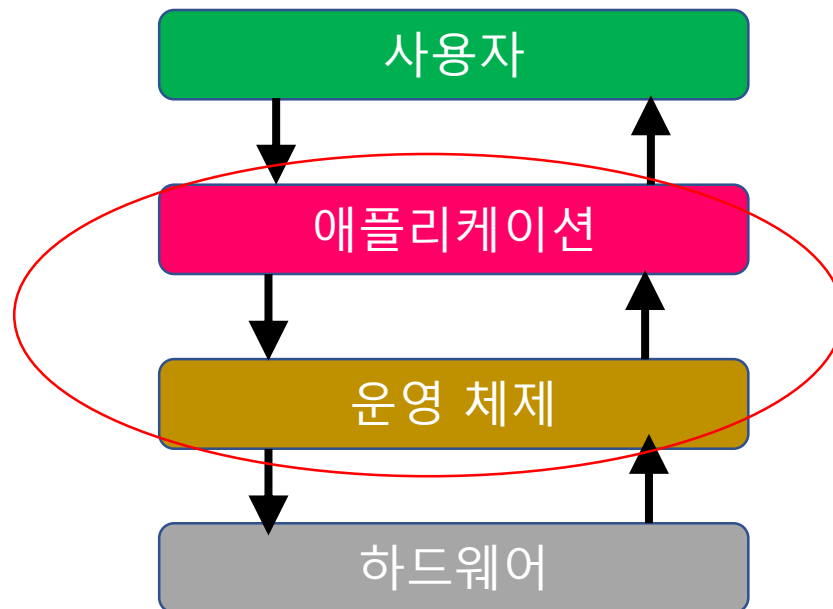
# 프로그램이란?

- 컴퓨터는 **범용적인 기계이다.** -> 컴퓨터를 사용하여 많은 작업을 할 수 있다.
- 컴퓨터를 범용적으로 만드는 것은 바로 **프로그램**이라는 개념을 사용하기 때문이다.



# 만약 컴퓨터에 프로그램이 없다면?

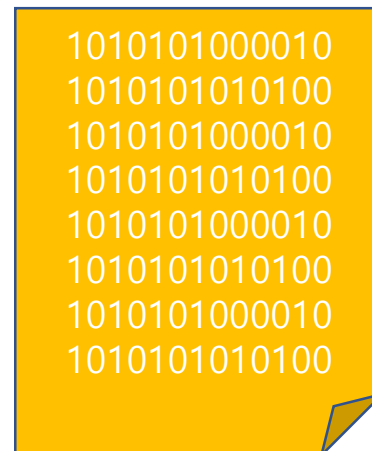
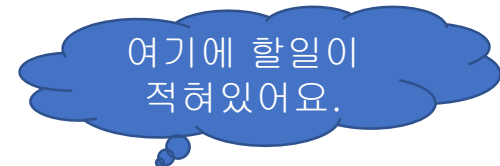
- 컴퓨터 하드웨어만 있고 프로그램이 없다면 컴퓨터는 그저 약간의 열과 소음을 발생하는 쓸모없는 기계에 불과하다.
- "윈도우즈"와 같은 운영 체제를 설치하고 추가로 여러가지 응용 프로그램을 설치해야만 비로소 유용하게 된다.



# 프로그램은 설치하도록 했을까?

Q) 왜 컴퓨터에서는 가전제품처럼 프로그램 설치 없이 바로 동작되도록 하지 않고 불편하게 사용자가 프로그램을 설치하게 하였을까 ?

A) 컴퓨터를 범용적인 기계로 만들기 위해서이다.  
컴퓨터는 프로그램만 바꾸어주면 다양한 작업을 할 수 있다.



# 계산기와 컴퓨터의 차이

정해진 기능만을 수행해요.  
기능을 변경할 수는 없어요.

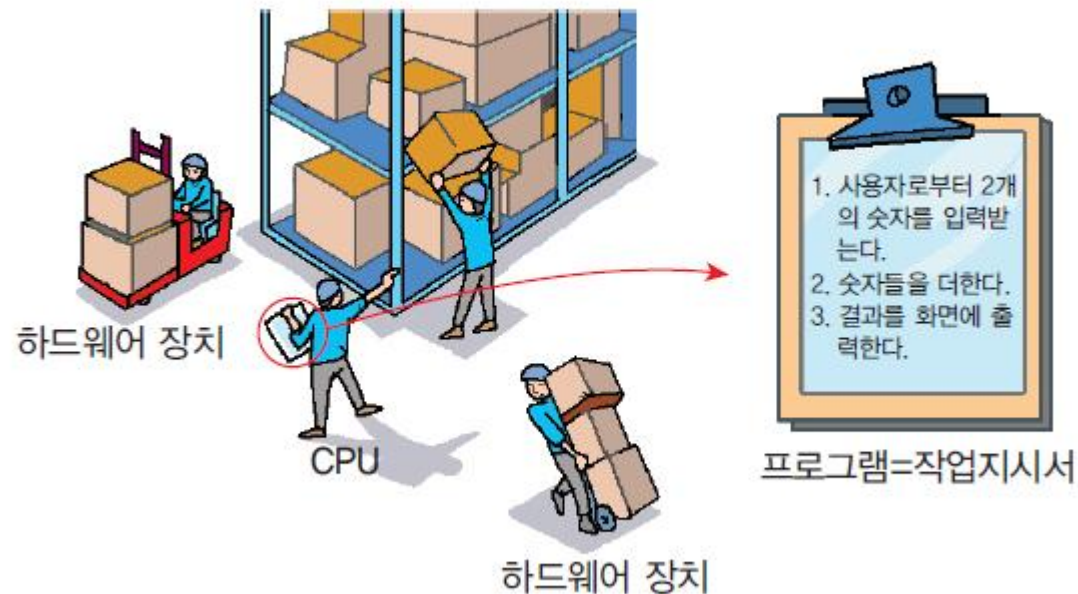


수행하는 기능을 쉽게 바꿀  
수 있다.



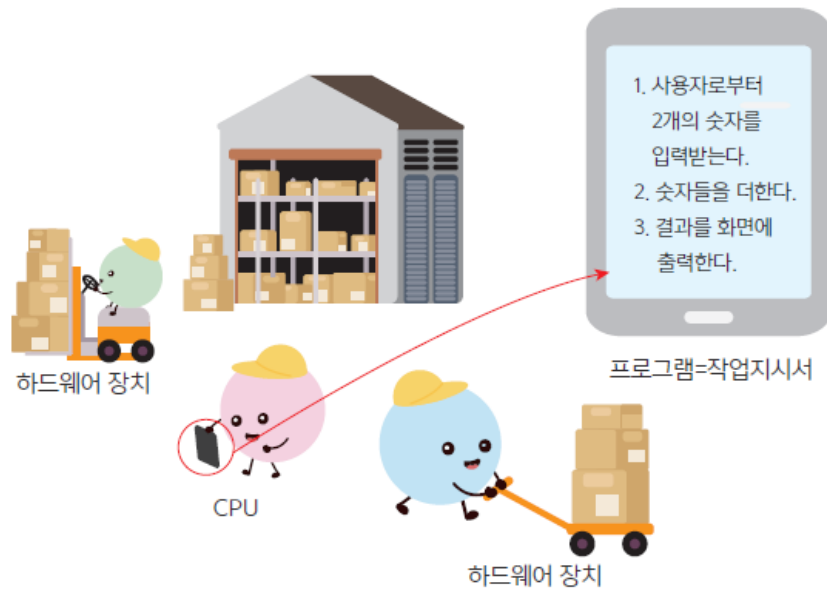
# 컴퓨터의 정의

- 컴퓨터(computer)는 단순히 계산(compute)만 하는 기계가 아닙니다.
- 현대적인 의미에서의 컴퓨터는 프로그램(명령어들의 리스트)에 따라 데이터를 처리하는 기계라고 할 수 있다



# 프로그램 안에는 무엇이 들어있을까?

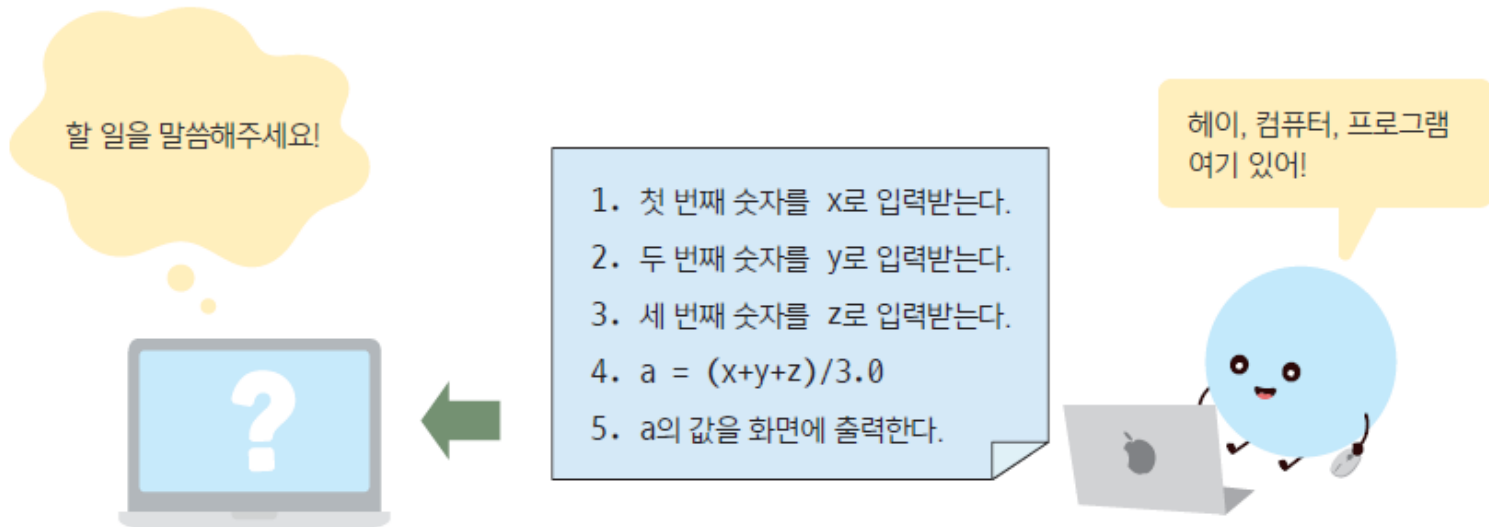
- 프로그램은 특정한 작업을 위한 작업 지시서라고 보면 된다.
- 작업을 지시하려면 명령어(instruction)들을 나열해야 한다. 프로그램 안에는 명령어들이 들어 있다.





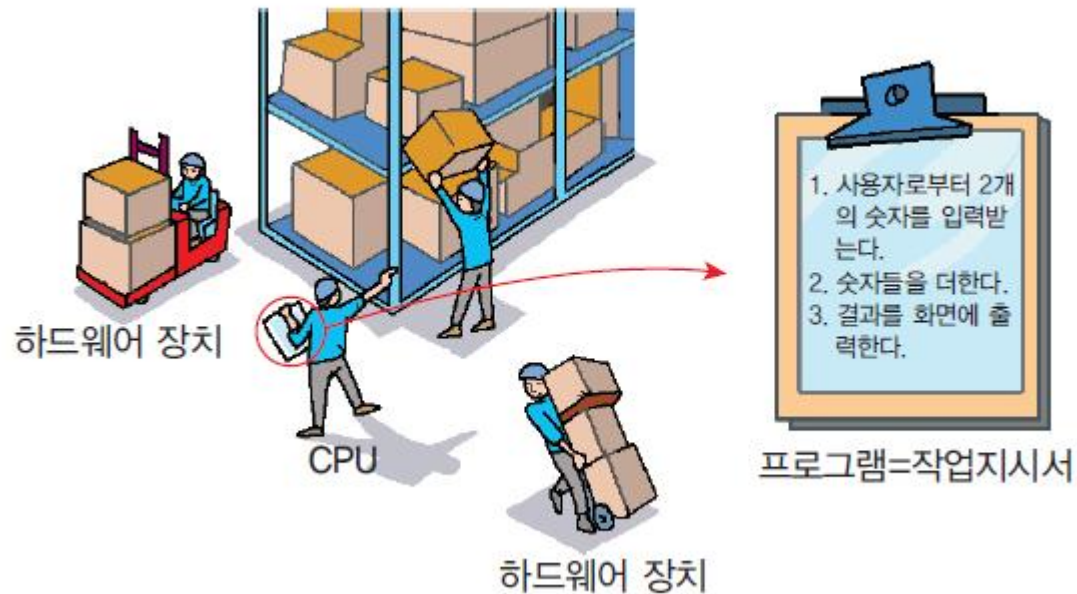
# 프로그램의 예

- 3개의 숫자를 받아서 평균을 계산하는 프로그램이라고 하자. 프로그램은 다음과 같은 지시사항들로 이루어 질 수 있다. 이 지시사항들이 바로 명령어이다.



# 프로그램==작업지시서

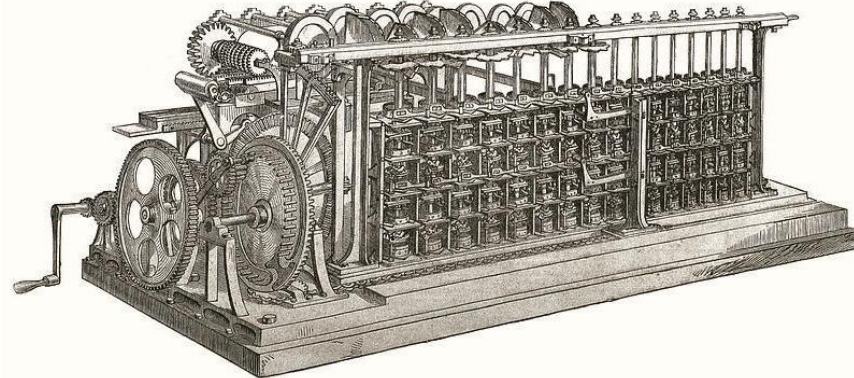
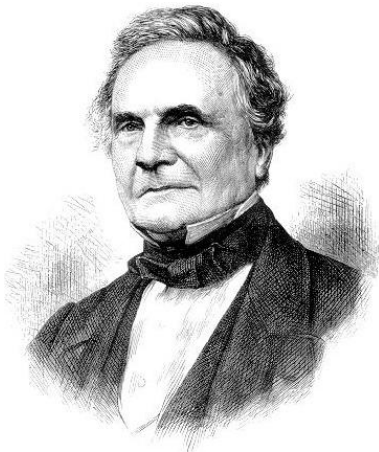
- 프로그램: 컴퓨터에게 해야 할 작업의 내용을 알려주는 문서



# 프로그램의 역사

실제로 만들어지지는 못했어요!

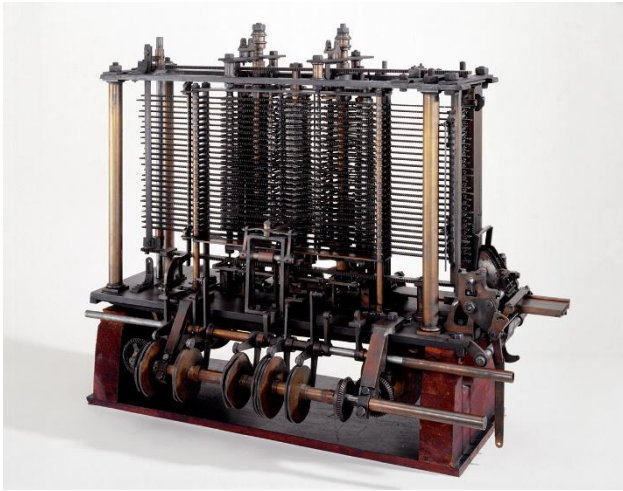
- 프로그래밍이 가능한 최초의 기계: **해석 기관(Analytical Engine)**
- 만든이: **찰스 배비지**
- 수천 개의 기어, 바퀴, 축, 레버 등이 증기로 작동



차분 기관

# 배비지의 해석기관

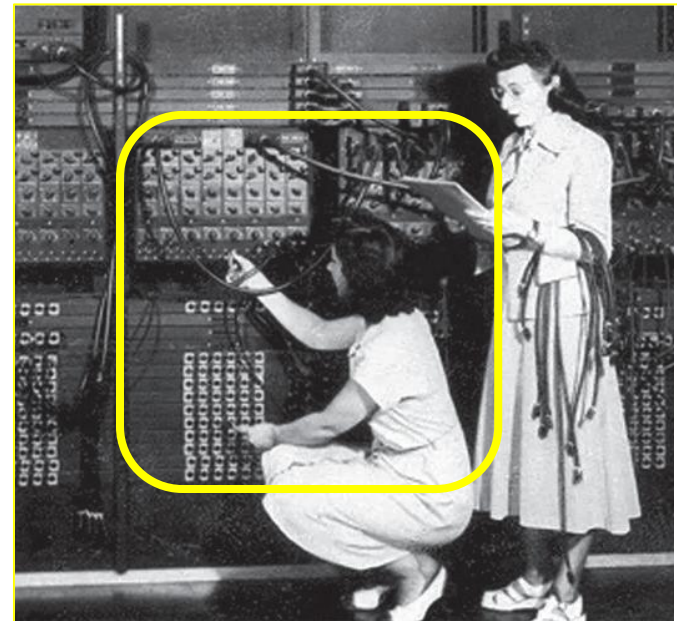
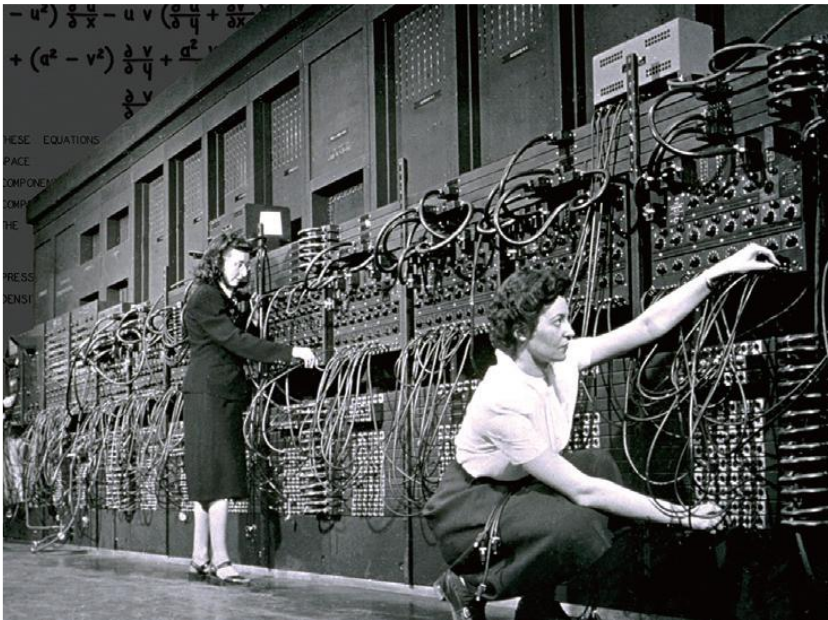
- 배비지의 해석 기관은 수천 개의 기어, 바퀴, 축, 레버 등으로 구성되어 증기로 작동하는 것으로 설계되었다.



- \* 중앙 처리 장치(계산을 담당, mill이라고 불림)
- \* 메모리(중간 단계에서 임시로 숫자가 저장, store라고 불림)
- \* 출력 장치(출력 숫자를 나타내는 다이얼)
- \* 입력 장치(천공 카드)

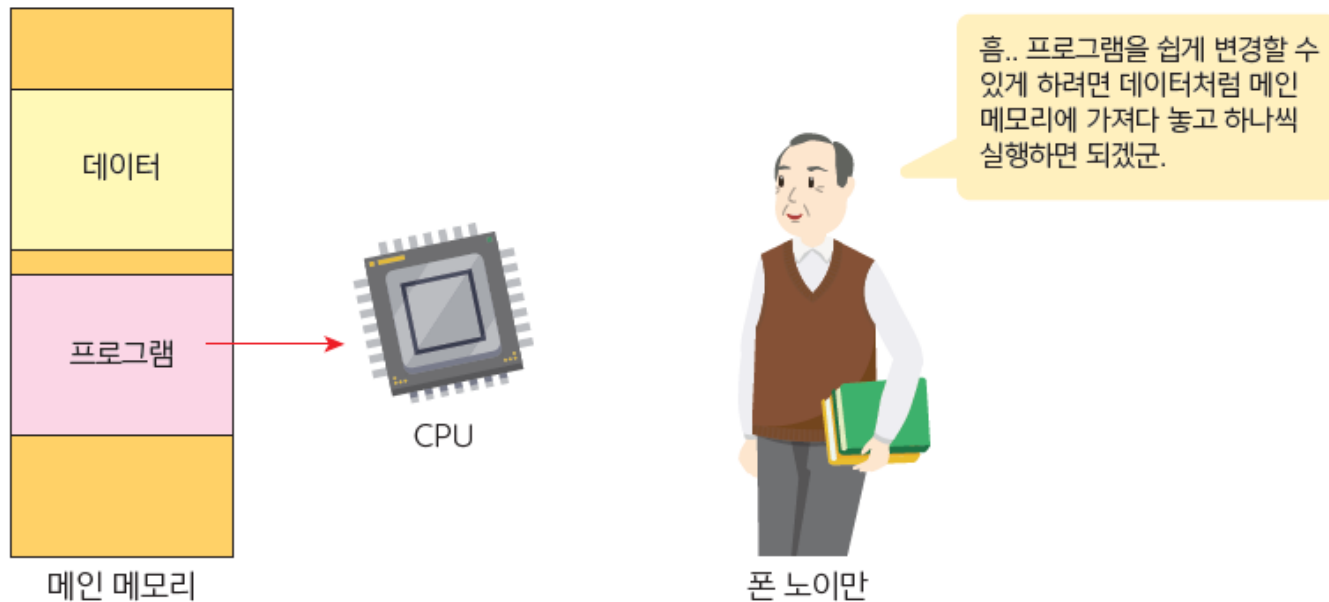
# 초기 컴퓨터의 프로그래밍

- 초기 컴퓨터인 ENIAC의 프로그램은 스위치에 의하여 기억되었고 프로그램을 변경할 때마다 그 많은 스위치들을 처음부터 다시 연결하여야 했다.



# 폰노이만 구조

- 프로그램은 **메인 메모리에 저장된다.** -> **쉽게 변경가능**
- 메인 메모리에 저장된 프로그램에서 명령어들을 순차적으로 가져와서 실행한다.





# 최초의 프로그래머

- 프로그램을 최초로 만든 사람은 **에이다 러브레이스(Ada Lovelace)**
- 에이다는 대문호 바이런의 친딸
- 배비지의 해석 기관에 매료되어 해석 기관을 위한 프로그램을 개발하였다.
- 서브루틴(subroutine), 루프(loop), 점프(jump) 등의 핵심적인 컴퓨터 프로그래밍 기본 원리를 고안



숫자의  
마술사(Enchantress of  
Number)!

# 에이다의 프로그램

Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

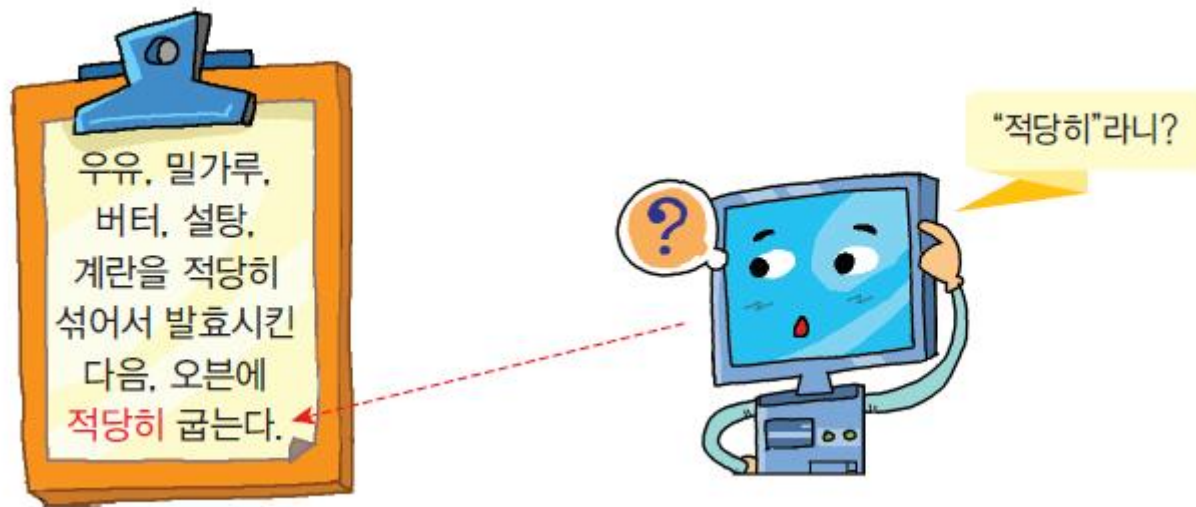
Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.												Working Variables.												Result Variables.			
						$V_1$	$V_2$	$V_3$	$V_4$	$V_5$	$V_6$	$V_7$	$V_8$	$V_9$	$V_{10}$	$V_{11}$	$V_{12}$	$V_{13}$	$V_{14}$	$V_{15}$	$V_{16}$	$V_{17}$	$V_{18}$	$V_{19}$	$V_{20}$	$V_{21}$	$V_{22}$	$V_{23}$	$V_{24}$	$V_{25}$	$V_{26}$	$V_{27}$	
						1	2	n																									
1	$\times$	$V_2 \times V_3$	$V_6$	$V_6 = V_2$	$-2n$	2	n	2n	2n	2n																							
2	$-$	$V_4 - V_1$	$V_4$	$V_4 = V_4$	$-2n-1$	1																											
3	$+$	$V_4 + V_1$	$V_4$	$V_4 = V_4$	$-2n+1$	1																											
4	$+$	$V_4 + V_1$	$V_4$	$V_4 = V_4$	$-2n+1$	1																											
5	$+$	$V_4 + V_1$	$V_4$	$V_4 = V_4$	$-2n+1$	1																											
6	$-$	$V_{13} - V_{11}$	$V_{13}$	$V_{13} = V_{13}$	$-1$	2																											
7	$-$	$V_3 - V_1$	$V_{10}$	$V_{10} = V_3$	$-n-1$	1																											
8	$+$	$V_3 + V_1$	$V_7$	$V_7 = V_3$	$-2+0=2$	2																											
9	$+$	$V_3 + V_1$	$V_{11}$	$V_{11} = V_3$	$-2n$	2																											
10	$\times$	$V_{11} \times V_{12}$	$V_{12}$	$V_{12} = V_{11}$	$-2n-1$	1																											
11	$+$	$V_{12} + V_{11}$	$V_{13}$	$V_{13} = V_{12}$	$-2n$	1																											
12	$-$	$V_{10} - V_1$	$V_{10}$	$V_{10} = V_1$	$-n-2$	1																											
13	$+$	$V_6 - V_1$	$V_6$	$V_6 = V_6$	$-2n-1$	1																											
14	$+$	$V_1 + V_1$	$V_7$	$V_7 = V_1$	$-2+1=3$	1																											
15	$\times$	$V_6 \times V_1$	$V_6$	$V_6 = V_6$	$-2n-1$	1																											
16	$\times$	$V_6 \times V_1$	$V_{12}$	$V_{12} = V_6$	$-2n-1$	1																											
17	$-$	$V_6 - V_1$	$V_6$	$V_6 = V_6$	$-2n-2$	1																											
18	$+$	$V_1 + V_1$	$V_7$	$V_7 = V_1$	$-3+1=4$	1																											
19	$+$	$V_6 + V_1$	$V_9$	$V_9 = V_6$	$-2n-2$	1																											
20	$\times$	$V_9 \times V_{11}$	$V_{11}$	$V_{11} = V_9$	$-2n-1$	1																											
21	$\times$	$V_{11} \times V_{12}$	$V_{12}$	$V_{12} = V_{11}$	$-2n-1$	1																											
22	$+$	$V_{12} + V_{11}$	$V_{13}$	$V_{13} = V_{12}$	$-2n$	1																											
23	$-$	$V_{10} - V_1$	$V_{10}$	$V_{10} = V_1$	$-n-3$	1																											
Here follows a repetition of Operations thirteen to twenty-three.																																	
24	$+$	$V_{13} + V_{11}$	$V_{13}$	$V_{13} = V_{11}$	$-n-1$	1																											
25	$+$	$V_1 + V_1$	$V_7$	$V_7 = V_1$	$-n+1$	1																											

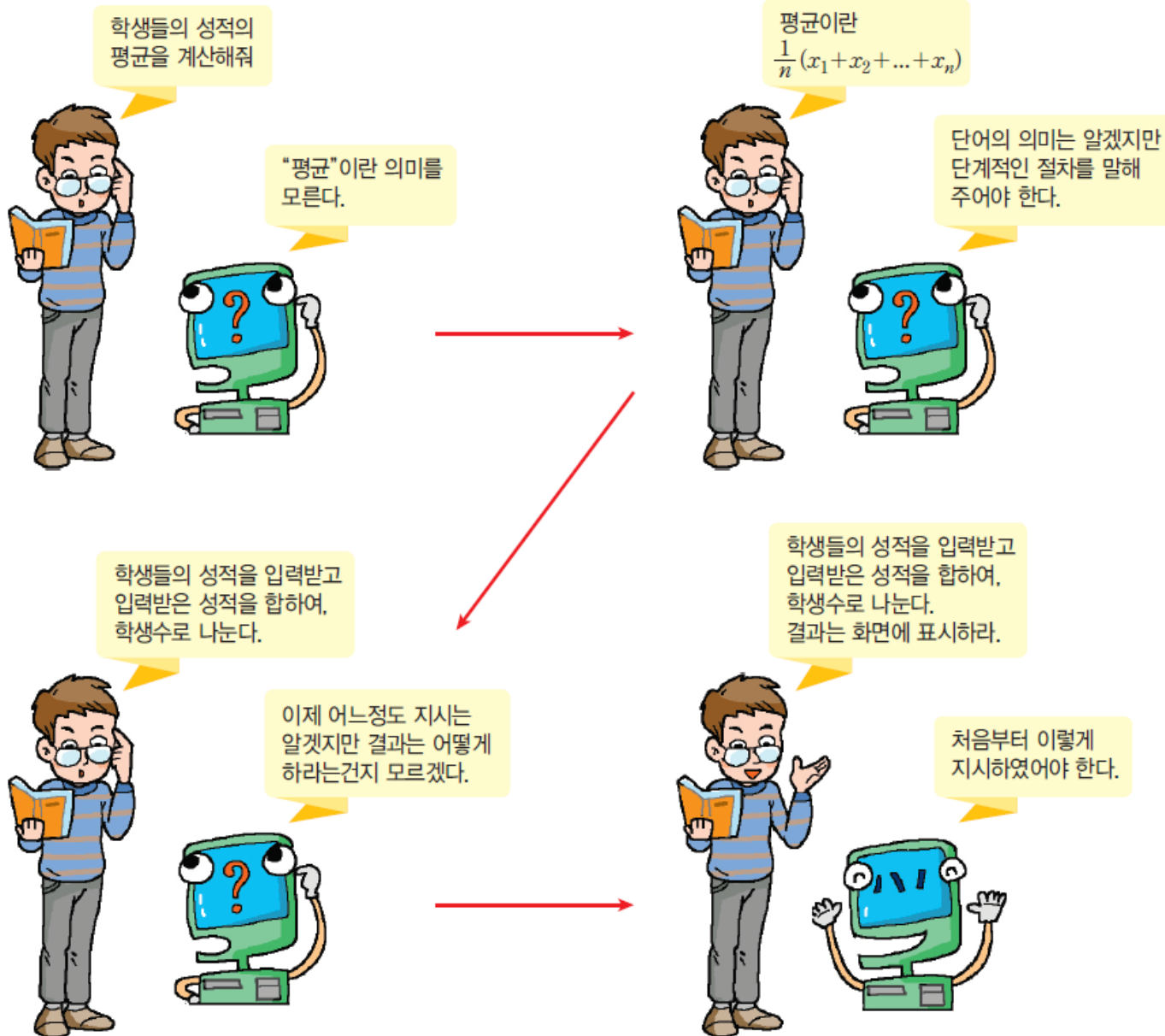
1842년에 에이다는 분석 엔진에 대한 이탈리아 엔지니어 Luigi Menabrea의 기사를 번역하는 과정에서 여백에 주석을 추가하였다. 에이다의 주석은 기사 자체보다 3배 더 길며, 해석 엔진을 사용하여 Bernoulli 수를 계산하는 방법을 자세하게 설명한다. 이 주석은 컴퓨터의 초기 역사에서 많은 사람들이 최초의 컴퓨터 프로그램, 즉 기계가 수행하도록 설계된 알고리즘으로 간주한다.



# 얼마나 자세하게 작업을 지시해야 할까?

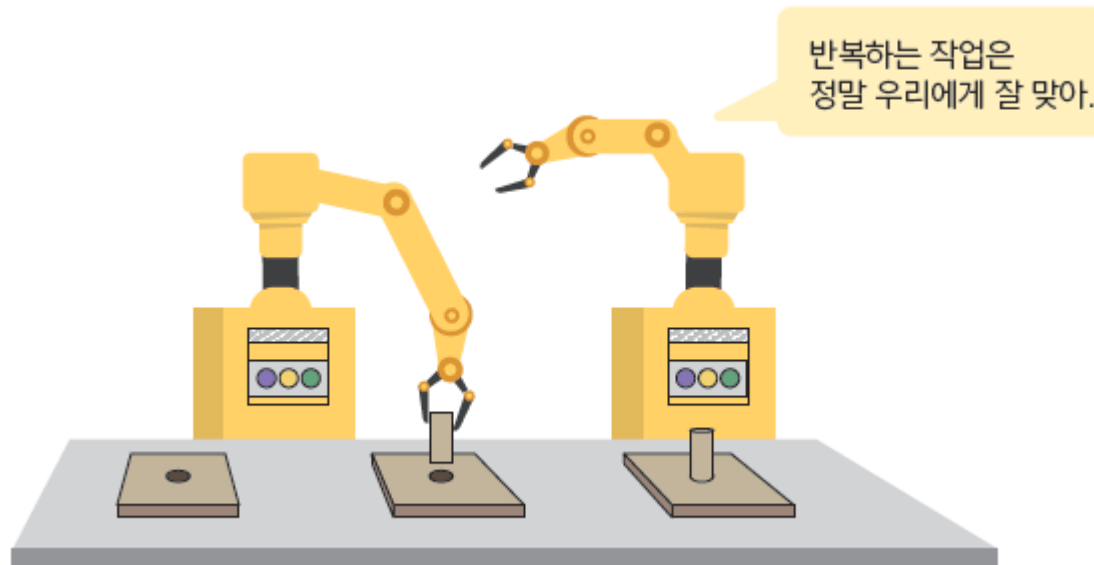
- 상식이나 지능이 없기 때문에 아주 자세하고 구체적으로 일을 지시하여야 한다.






# 컴퓨터의 장점

- 작업을 굉장히 빠르게 하고 정확하게 하며 몇 번을 반복해서 시켜도 불평이 없다.



# 이번 장에서 학습할 내용

- 
- 프로그래밍 개념
  - 프로그래밍 언어
  - 알고리즘
  - 스크래치



# 컴퓨터가 이해하는 언어

- 컴퓨터가 알아듣는 언어는 한가지이다. 즉 0과 1로 구성되어 있는 “001101110001010...”과 같은 기계어이다.
- 컴퓨터는 모든 것을 0과 1로 표현하고 0과 1에 의하여 내부 스위치 회로들이 ON/OFF 상태로 변경되면서 작업을 한다.

우리말 아니?

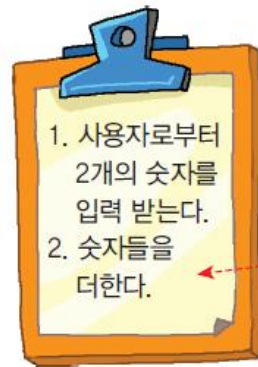


아니 기계어 밖에 몰라



# 기계어(machine language)

- 0과 1로 구성되어 있는 "001101110001010..."과 같은 2진수이다.

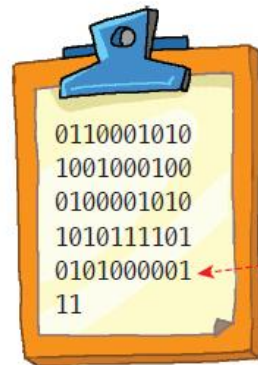


한글로 된 작업 지시서

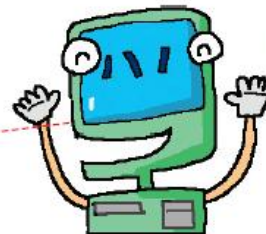


컴퓨터

한글은 너무 어려워.

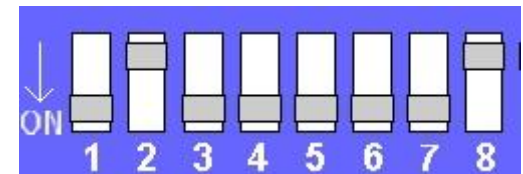


기계어로 된 작업 지시서



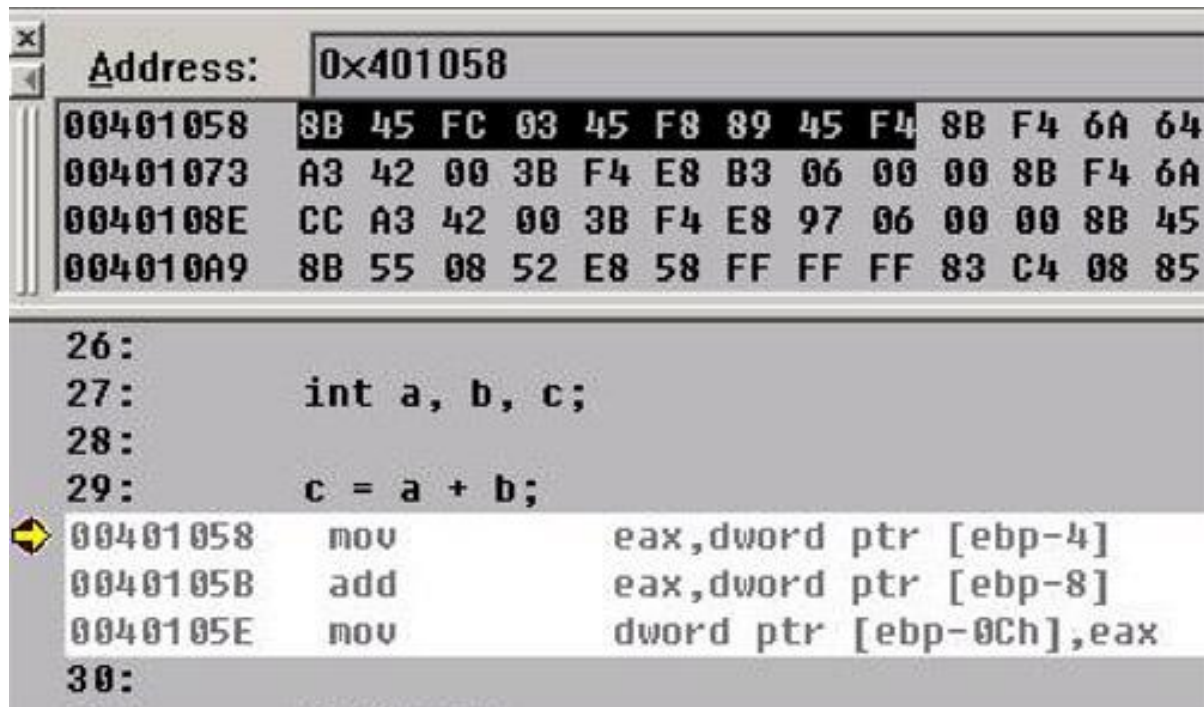
컴퓨터

역시  
이진수가 최고야.



# 기계어

- 기계어의 예



The screenshot shows a debugger window with a memory dump and assembly code. The memory dump is located at address 0x401058 and contains four rows of 16 bytes each. The assembly code is located below the memory dump and consists of four instructions. The first instruction is at address 00401058 and is a 'mov' instruction. The second instruction is at address 0040105B and is an 'add' instruction. The third instruction is at address 0040105E and is a 'mov' instruction. The fourth instruction is at address 30: and is a 'mov' instruction.

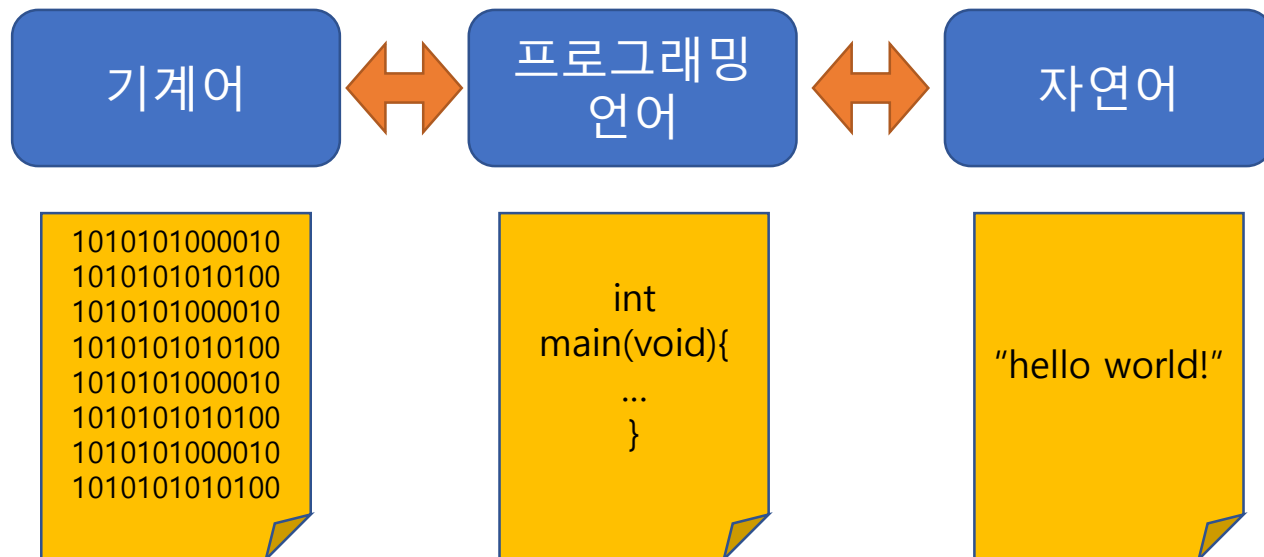
Address	Hex
00401058	8B 45 FC 03 45 F8 89 45 F4 8B F4 6A 64
00401073	A3 42 00 3B F4 E8 B3 06 00 00 8B F4 6A
0040108E	CC A3 42 00 3B F4 E8 97 06 00 00 8B 45
004010A9	8B 55 08 52 E8 58 FF FF FF 83 C4 08 85

Address	Disassembly
26:	
27:	int a, b, c;
28:	
29:	c = a + b;
00401058	mov eax,dword ptr [ebp-4]
0040105B	add eax,dword ptr [ebp-8]
0040105E	mov dword ptr [ebp-0Ch],eax
30:	

# 프로그래밍 언어

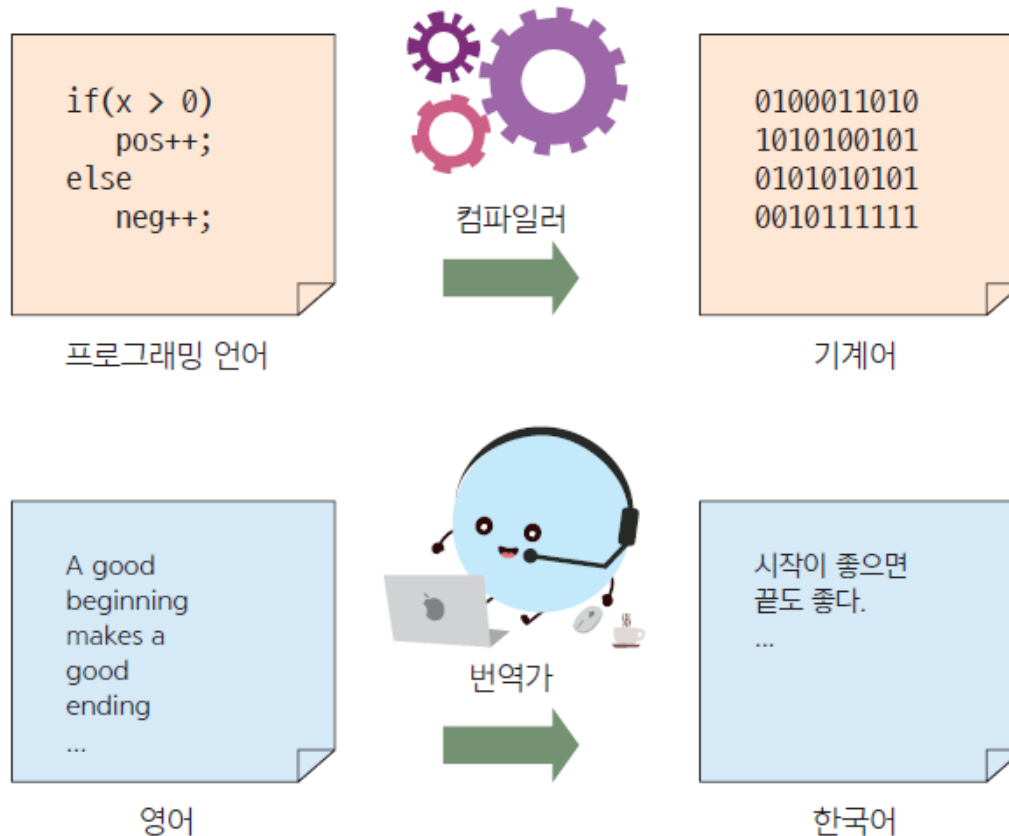
- 기계를 사용할 수는 있으나 이진수로 프로그램을 작성하여야 하기 때문에 아주 불편하다.
- 프로그래밍 언어는 자연어와 기계어 중간 쯤에 위치
- 컴파일러가 프로그래밍 언어를 기계어로 통역





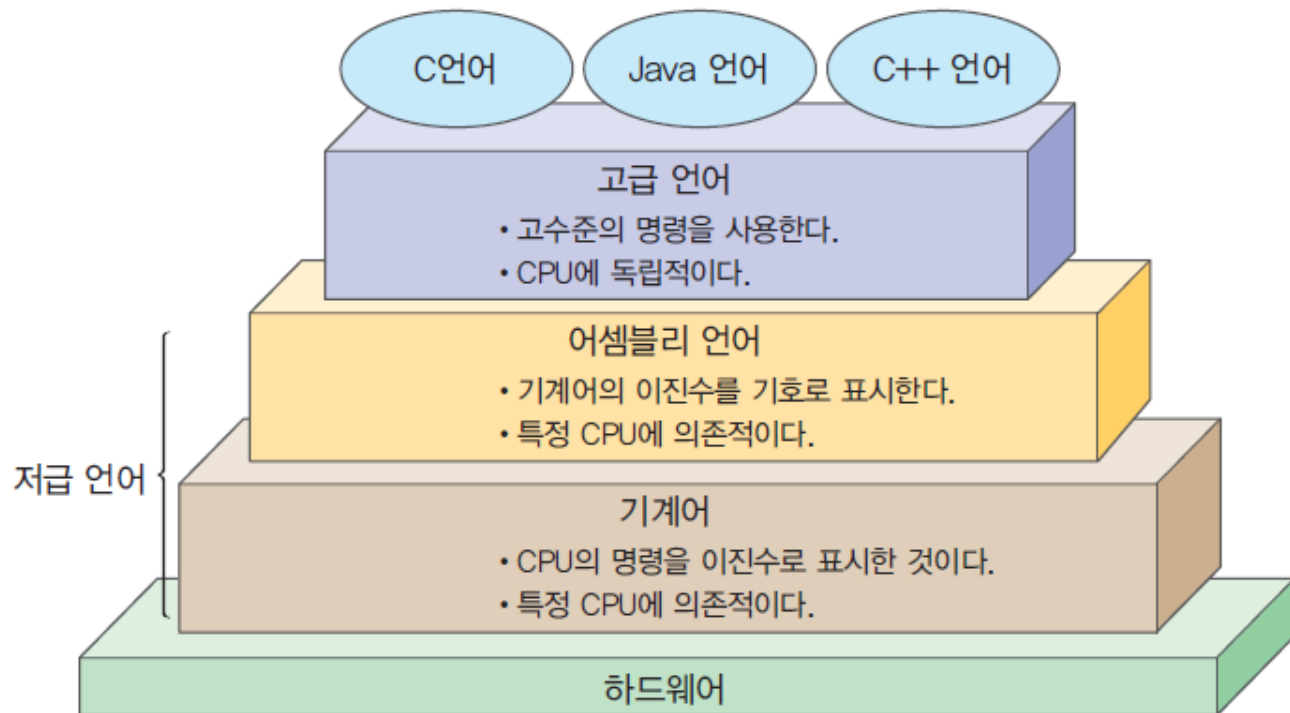
# 컴파일러

- 컴파일러(compiler)는 인간과 컴퓨터 사이의 통역이라 할 수 있다.



# 프로그래밍 언어의 분류

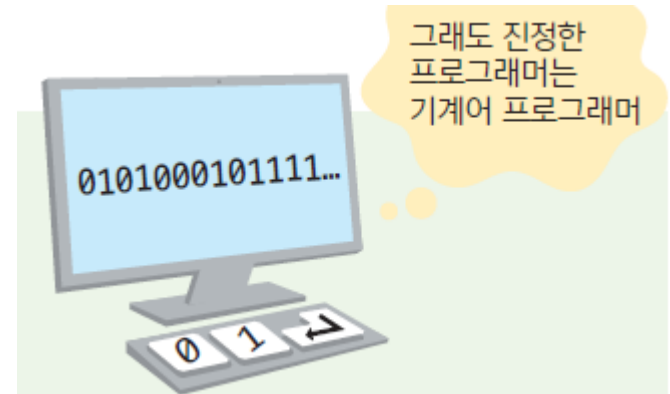
- 기계어(machine language)
- 어셈블리 언어(assembly language)
- 고급 언어(high-level language)



# 기계어

- 특정 컴퓨터의 명령어(instruction)를 이진수로 표시한 것
- 0과 1로 구성
- 하드웨어에 종속

```
00001111 10111111 01000101  
11111000 00001111 10111111  
01001101 11111000 00000011  
10100001 01100110 10001001  
01000101 11111010
```



# 어셈블리어

- CPU의 명령어들을 이진수가 아닌 영어의 약자인 기호로 표기
- 기계어보다는 더 높은 수준에서 프로그램을 작성하는 것을 가능
- 기호와 CPU의 명령어가 일대일 대응
- 어셈블러(assembly): 기호를 이진수로 변환하는 프로그램

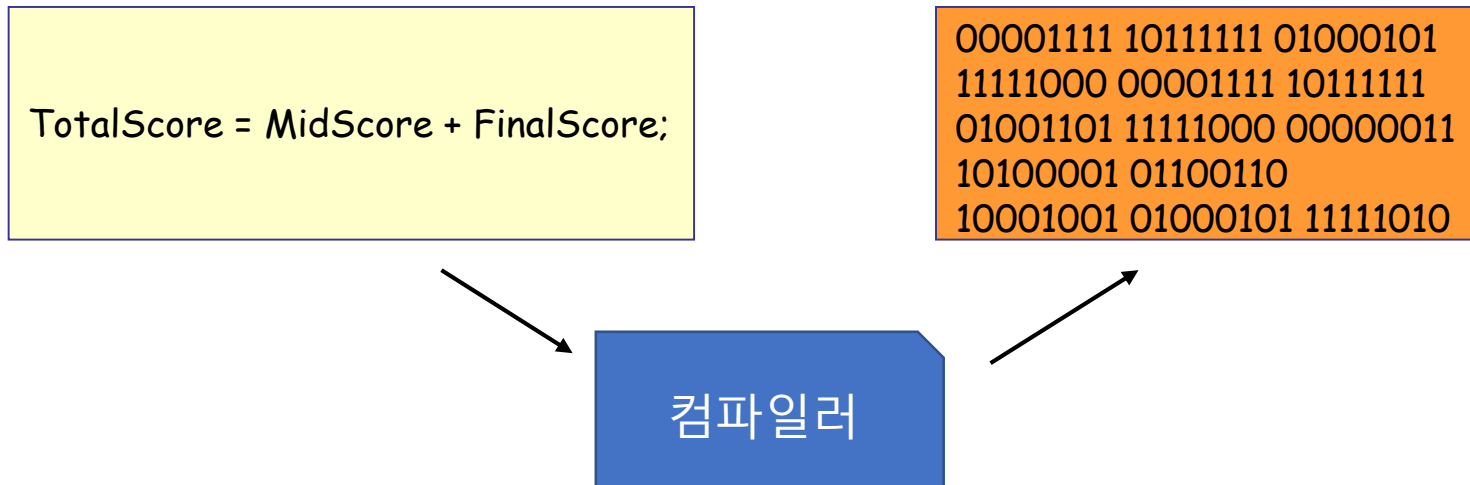
```
MOV AX, MIDSCORE  
MOV CX, FINALSORE  
ADD AX, CX  
MOV TOTALSCORE, AX
```

```
00001111 10111111 01000101  
11111000 00001111 10111111  
01001101 11111000 00000011  
10100001 01100110 10001001  
01000101 11111010
```

어셈블러

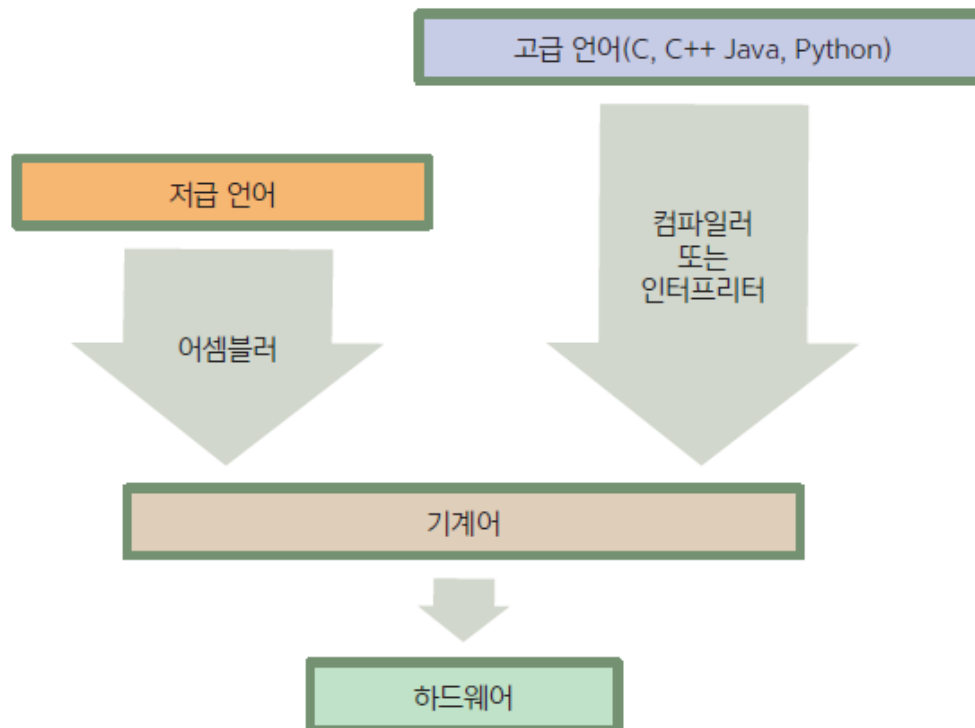
# 고급언어

- 특정한 컴퓨터의 구조나 프로세서에 무관하게, 독립적으로 프로그램을 작성할 수 있는 언어
- C, C++, JAVA, FORTRAN, PASCAL
- 컴파일러: 고급 언어 문장을 기계어로 변환하는 프로그램



# 저급 언어와 고급 언어

- 이들을 고급 언어라고 부르는 이유는, 이 언어들의 구성이 기계어보다는 인간의 언어에 가깝기 때문이다. 이와는 반대로 어셈블리 언어 등은 기계어에 가깝기 때문에 저급 언어로 분류된다.



# 고급 언어의

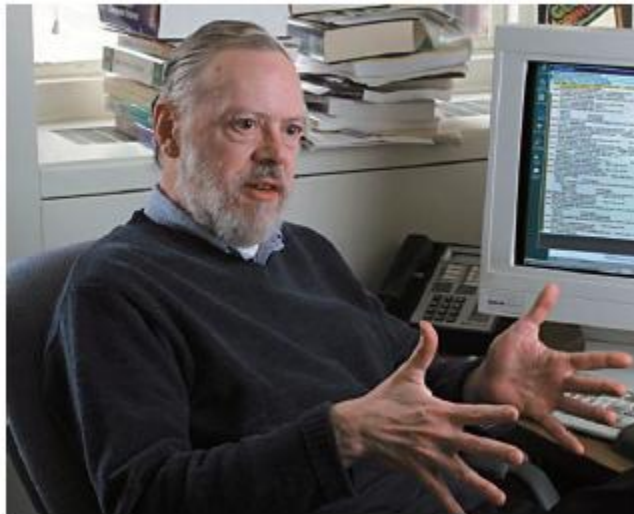


Made by FREE-VECTORS.NET

언어	특징	예제
FORTRAN	포트란은 1950년대에 가장 먼저 만들어진 언어로서 수치 계산과 과학 기술 계산에 적합하다.	<pre>PROGRAM HELLO   PRINT '(A)', 'Hello World'   STOP END</pre>
COBOL	코볼은 1959년 만들어진 비즈니스 사무 처리 언어이다. 이 언어는 구어체 문장 형태로 기술할 수 있도록 설계되었다.	<pre>IDENTIFICATION DIVISION.   PROGRAM-ID. HELLO-WORLD.   PROCEDURE DIVISION.     DISPLAY 'Hello World'.   STOP RUN.</pre>
Python	파이썬은 1991년 귀도 반 로섬(Guido van Rossum)이 개발한 인터프리트 언어이다. 초보자들이 배우기 쉬우며 인공지능, 데이터 과학 분야에서 많이 사용된다.	<pre>print("Hello World")</pre>
C	C언어는 1970년대 초반, UNIX 운영체제를 위하여 AT&T의 벨 연구소에서 일하던 데니스 리치에 의하여 만들어졌다.	<pre>int main(void) {   printf("Hello World\n");   return 0; }</pre>
C++	C++는 1983년 벨 연구소의 스트로스트룹에 의하여 개발된 언어로서 C언어에 클래스 개념을 비롯하여 여러 가지 객체지향적인 특징들을 추가한 언어이다.	<pre>int main() {   cout &lt;&lt; "Hello World" &lt;&lt; endl;   return 0; }</pre>
Java	자바는 1995년 선 마이크로시스템의 제임스 고슬링에 의하여 개발된 객체 지향 언어이다.	<pre>public class Hello {   public static void main(String[] args)   {     System.out.println("Hello World");   } }</pre>

# C언어의 소개

- 1970년대 초 AT&T의 Dennis Ritchie 에 의하여 개발
- B언어->C언어
- UNIX 운영 체제 개발에 필요해서 만들어짐
- 처음부터 전문가용 언어로 출발





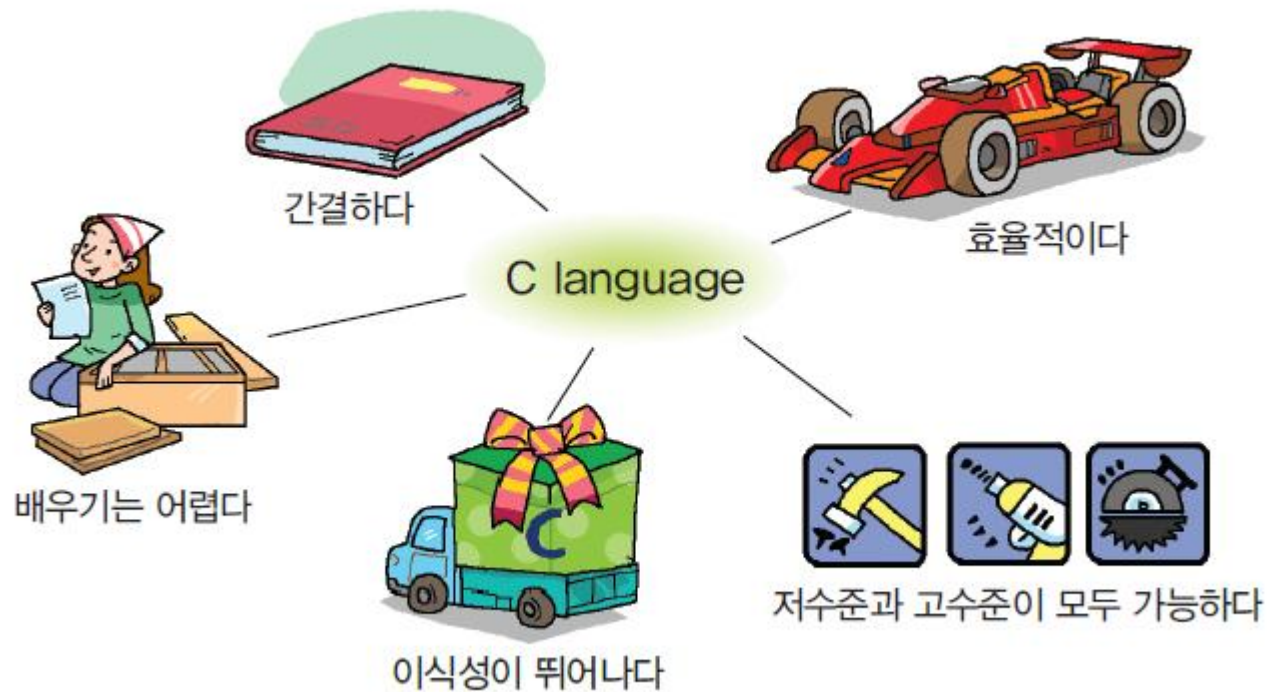
# C언어의 버전

- K & R C
  - 1978년 "C Programming Language" 책 출간
  - 비공식적인 명세서 역할
- ANSI C
  - 1983년 ANSI(American National Standards Institute)는 X3J11이라는 위원회에 의한 표준
- C99
  - 1999년에 ISO에 의한 표준
  - C++에서 사용되는 특징 추가
  - 점차 많은 컴파일러에서 지원
- C11
  - ISO에 의하여 2011년 12월에 발표된 C언어 표준이다.
- C17, C18
  - 2018년 6월에 ISO/IEC 9899:2018로 발표된 C17은 현재 표준이다. 새로운 언어 기능이 추가된 것은 없고 C11 버전의 기술적인 결함만 수정되었다.

# C언어의 특징

- 간결하다.
- 효율적이다.
- C 언어는 하드웨어를 직접 제어하는 하는 저수준의 프로그래밍도 가능하고 고수준의 프로그래밍도 가능하다.
- C언어는 이식성이 뛰어나다.
- C언어를 이해하면 컴퓨터 하드웨어가 어떻게 동작하는 지를 이해할 수 있다.

# C언어의 특징



# C언어의 단점

- 초보자가 배우기가 어렵다는 것이다.
- 하드웨어를 제어하기 위하여 꼭 필요한 요소인 포인터 등을 잘못 사용하는 경우가 많다.



# 앞으로도 C언어는 사용될 것인가?

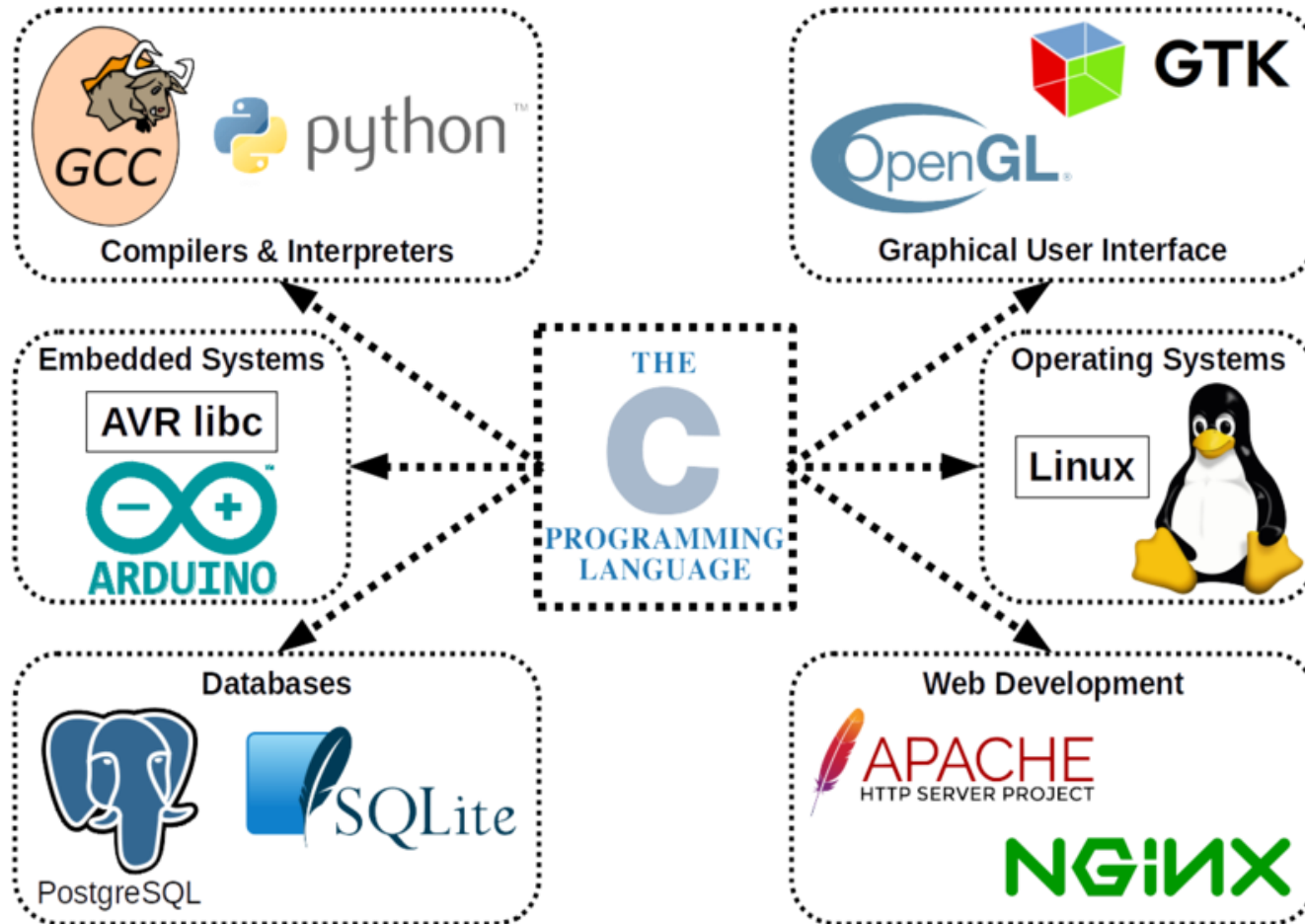
- C언어는 C++와 JAVA의 공통적인 부분이다.
- 실행 속도가 중요한 프로그램은 C언어로 구현된다.
- 임베디드 시스템에서는 C언어가 많이 사용된다

임베디드 시스템: 임베디드 시스템이란 특수 목적의 시스템으로 컴퓨터가 장치 안에 MP3 플레이어, 핸드폰등이 여기에 속한다.



스마트폰도 CPU와 플래시 메모리 등이 들어가 있는 임베디드 시스템이다.

# C언어의 사용처



# 이번 장에서 학습할 내용



- 프로그램의 이해
- 프로그래밍 언어
- 알고리즘
- 스크래치

프로그램을  
작성하기에  
앞서서 중요한  
개념들을  
살펴봅니다..



# 알고리즘

Q) 오븐의 사용법만 배우고 음식 재료만 있으면 누구나 요리가 가능한가?

A) 요리법을 알아야 한다.





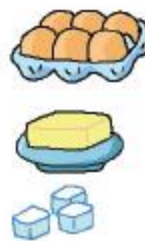
# 알고리즘이란?

- 알고리즘(algorithm): 문제를 풀기 위하여 컴퓨터가 수행하여야 할 단계적인 절차를 기술한 것
- (예) 전화번호부에서 특정한 사람(박철수라고 가정하자)의 전화번호를 찾는 문제를 생각하여 보자.



# 빵을 만드는 알고리즘

- ① 빈 그릇을 준비한다.
- ② 이스트를 밀가루, 우유에 넣고 저어준다.
- ③ 버터, 설탕, 계란을 추가로 넣고 섞는다.
- ④ 따뜻한 곳에 놓아두어 발효시킨다
- ⑤ 170~180도의 오븐에서 굽는다



# 1부터 10까지의 합을 구하는 알고리즘

① 1부터 10까지의 숫자를 직접 하나씩 더한다.

$$1 + 2 + 3 + \dots + 10 = 55$$

② 두수의 합이 10이 되도록 숫자들을 그룹핑하여 그룹의 개수에 10을 곱하고 남은 숫자 5를 더한다.

$$\begin{array}{l} (0 + 10) = 10 \\ (1 + 9) = 10 \\ (2 + 8) = 10 \\ (3 + 7) = 10 \\ (4 + 6) = 10 \end{array} \quad \left. \vphantom{\begin{array}{l} (0 + 10) = 10 \\ (1 + 9) = 10 \\ (2 + 8) = 10 \\ (3 + 7) = 10 \\ (4 + 6) = 10 \end{array}} \right\} \begin{array}{c} 10 * 5 = 50 \\ + \\ 5 \\ = \\ 55 \end{array}$$



③ 공식을 이용하여 계산할 수도 있다.

$$10(1+10)/2=55$$

# 알고리즘의 기술

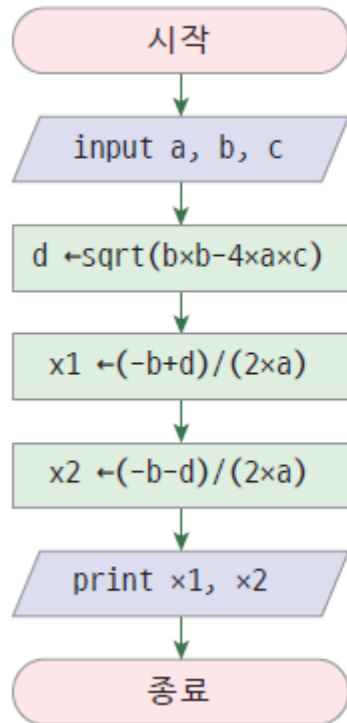
- 자연어(natural language)
- 순서도(flowchart)
- 의사 코드(pseudo-code)



컴퓨터 앞에 바로  
앉지 말고 알고리즘을  
구상하여야 합니다.

# 알고리즘의 예

- 2차 방정식의 근을 구하는 알고리즘



- Step 1: input a, b, c
- Step 2:  $d \leftarrow \sqrt{b^2 - 4ac}$
- Step 3:  $x_1 \leftarrow \frac{-b+d}{2a}$
- Step 4:  $x_2 \leftarrow \frac{-b-d}{2a}$
- Step 5: print x1, x2

# 알고리즘의 기술

- **순서도(flow chart):** 프로그램에서의 논리 순서 또는 작업 순서를 그림으로 표현하는 방법



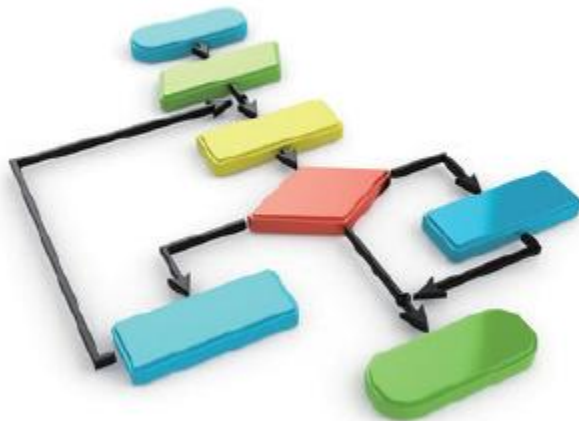
# 알고리즘의 예

- 학교 홈페이지에 로그인하는 알고리즘을 순서도로 표시해보자.



# 의사코드

- 의사 코드(pseudo code)는 자연어보다는 더 체계적이고 프로그래밍 언어보다는 덜 엄격한 언어로서 알고리즘의 표현에 주로 사용되는 코드를 말한다
- 예를 들어서 학생 10명의 성적을 입력받아서 평균을 계산하는 알고리즘을

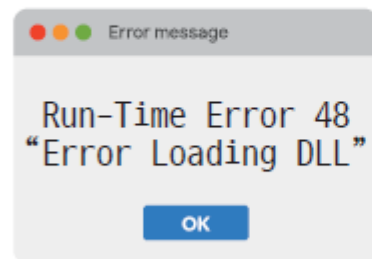


```
total ← 0
counter ← 1
while counter ≤ 10
    input grade
    grade ← grade + total
    counter ← counter + 1
average ← total / 10
print average
```



# 알고리즘의 중요성

- 우리가 컴퓨터에게 좋지 못한 알고리즘을 주는 경우에는 나쁜 결과를 얻을 수밖에 없다.
- 이때까지 스마트폰이나 가전제품, 자동차가 많은 오류를 일으킨 것을 생각해보라.
- 알고리즘에 오류가 없어야 컴퓨터 프로그램도 논리적인 오류가 없이 동작하게 된다.



# 알고리즘을 만드는 방법

1. 문제를 한 번에 해결하려고 하지 말고 더 작은 크기의 문제들로 분해한다.
2. 문제가 충분히 작아질 때까지 계속해서 분해한다.

① 방을 청소한다.

② 거실을 청소한다.

③ 부엌을 청소한다.

① 환기를 시킨다.

② 물건들을 정리한다.

③ 진공 청소기를 돌린다.

④ 걸레질을 한다.



환기



물건정리

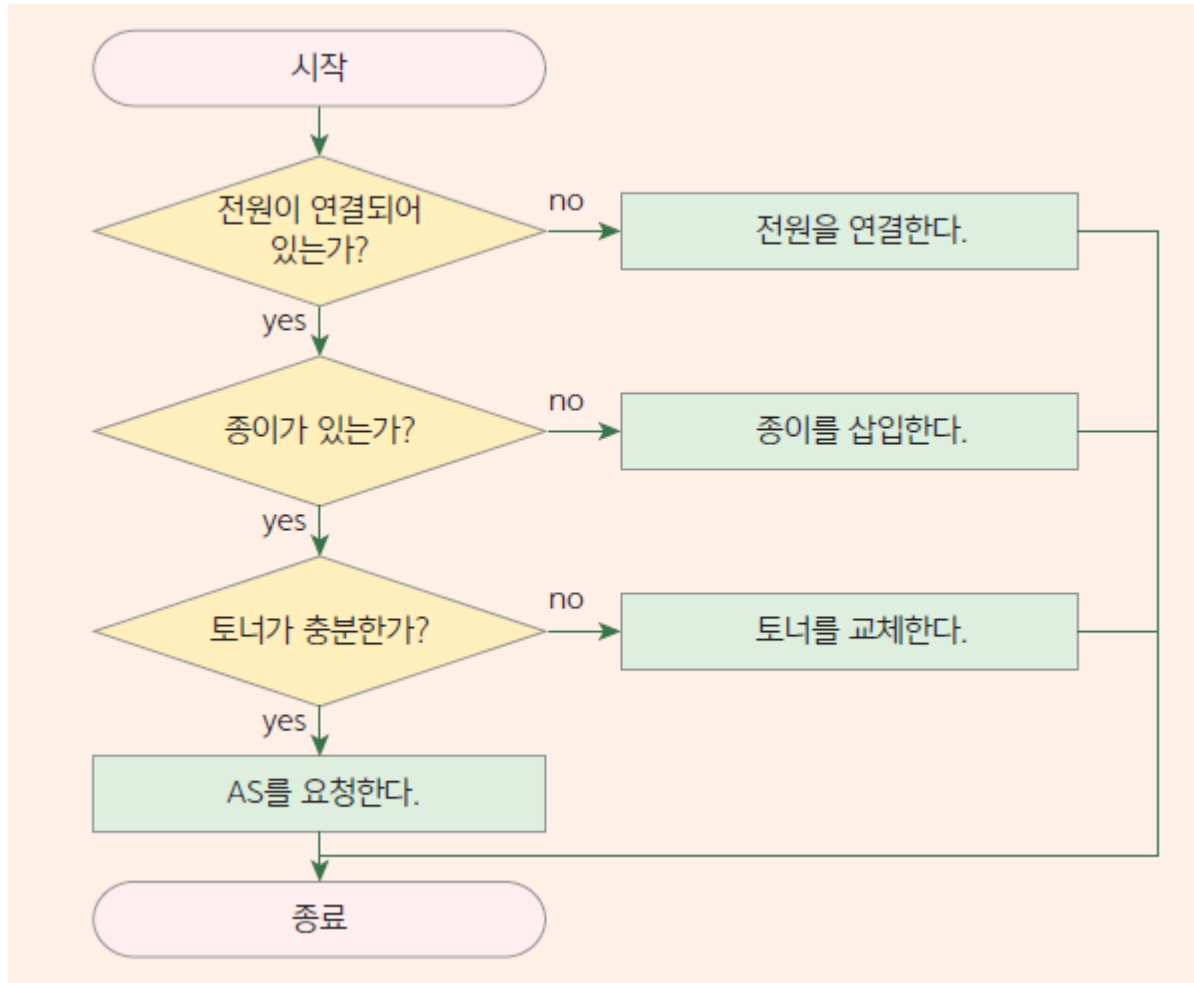


진공청소기



걸레질

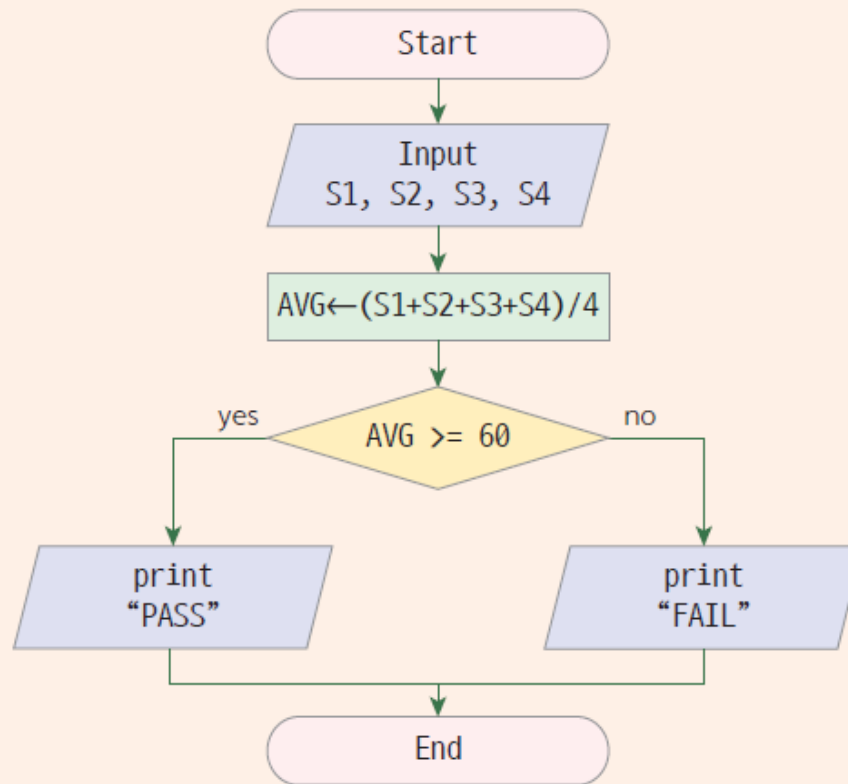
# Lab: 프린터 고장을 처리하는 알고리즘



# Lab: 합격, 불합격을 판단하는 알고리즘



Solution



# Q & A

