1강 - 프로그래밍 기초

목차

- 프로그래밍을 시작하기 위한 배경 지식
- 프로그래밍의 목적
- 프로그래밍 언어의 종류

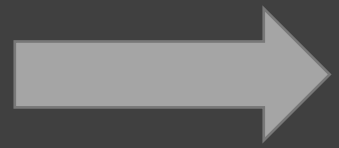
컴퓨터란?

컴퓨터란?

- 개인용 컴퓨터 (PC)
  - Desktops / Laptops
  - Personal mobile devices
- 서버 (Server)
  - Super computers
  - Warehouse-scale computing
- 임베디드 컴퓨터 (Embedded computer)
  - Chips

컴퓨터란?

- 개인용 컴퓨터 (PC)
  - Desktops / Laptops
  - Personal mobile devices
- 서버 (Server)
  - Super computers
  - Warehouse-scale computing
- 임베디드 컴퓨터 (Embedded computer)
  - Chips



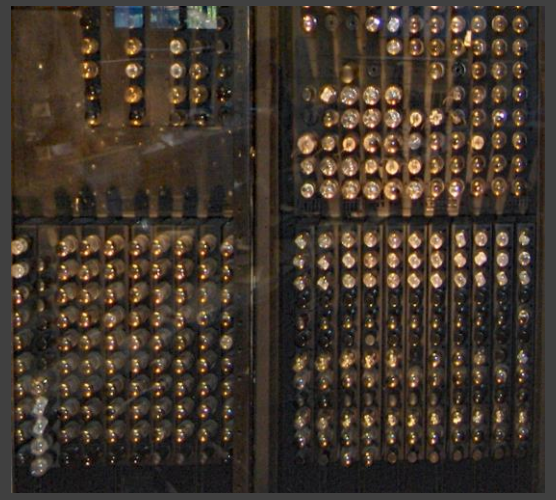
- 프로세서
- 메모리
- I/O

컴퓨터란?

- 프로세서
  - 계산 담당 (i.e. Logic 설계)
- 메모리
  - 기억 담당
- I/O
  - 데이터 송수신 담당

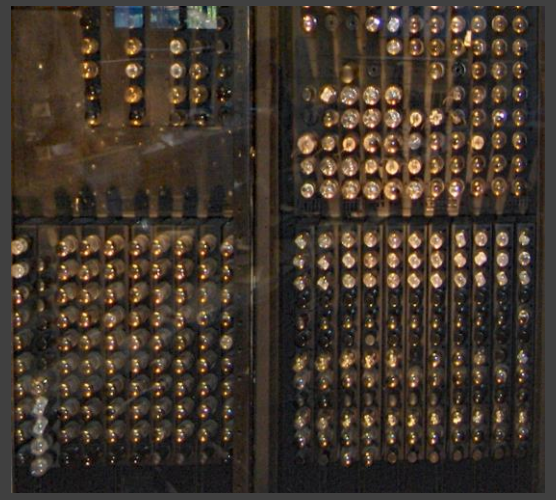
프로세서

- 진공관
- 트랜지스터
- IC
- Microcontroller / micro-processor



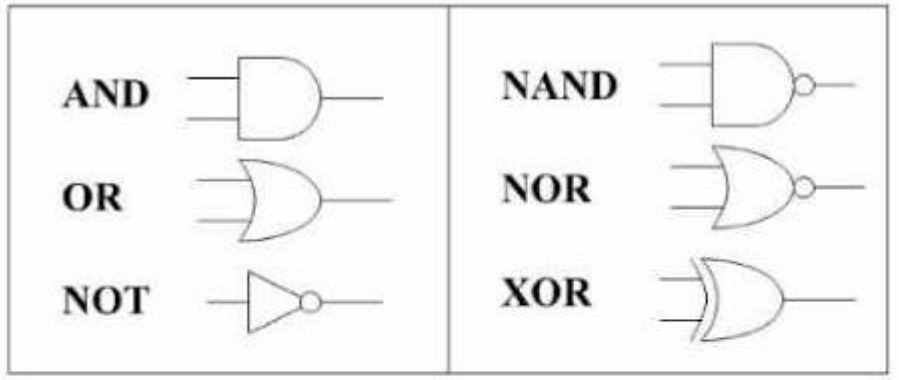
프로세서

- 진공관
- 트랜지스터 - Electronic switch (0, 1) i.e. 2진법/Binary
- IC - Lots of transistors
- Microcontroller / micro-processor - Lots of IC



프로세서

- Essentially...
  - Use of transistors mean binary expressions.
  - binary expressions mean logic gates.
  - Logic gates mean computations.
  - Computations mean processor!



A	B	A AND B	A OR B	NOT A	A NAND B	A NOR B	A XOR B
0	0	0	0	1	1	1	0
0	1	0	1	1	1	0	1
1	0	0	1	0	1	0	1
1	1	1	1	0	0	0	0

프로세서

- Example:
  - A + B = ? 이라는 계산에서
  - A = 2진법 수로 표현 가능 (0/1 로 표현 가능)
  - B = 2진법 수로 표현 가능 (0/1 로 표현 가능)
  - ? = 2진법 수로 표현 가능 (0/1 로 표현 가능)

  - Add는?
    - 1000110010100000

So far...

- 컴퓨터는 Binary expression으로 생각한다.

- 프로그래밍은 Binary expression이 아니다.
  - 그러면 컴퓨터는 코드를 어떻게 이해하는가?

```
5   class MyCsvUtil:
6       def __init__(self, base_path, sub_folder, csv_file):
7           self.base_path = base_path
8           self.sub_folder = sub_folder
9           self.csv_file = csv_file
10          print ("AM: util_general.py: MyGeneralUtil: class constructor" + self.sub_folder)
11
12      def get_data_from_csv_file(self):
13          print ("AM: get_data_from_csv_file.py: get_test_data method: START... ")
14
15          return_dict = {}
16          input_data = []
17
18          with open(self.base_path + "/" + self.sub_folder + "/" + self.csv_file) as csvfile:
19              reader = csv.DictReader(csvfile)
20              counter = 0
21              for row in reader:
22                  item = {}
23                  item["field01"] = row['field01']
24                  item["field02"] = row['field02']
25                  input_data.append(item)
26                  counter += 1
27
28          return_dict["data"] = input_data
29          metadata_details = {}
30          metadata_details["count"] = str(counter)
31          return_dict["metadata"] = metadata_details
32
33          return return_dict
34
```

어셈블리 언어 (Assembler)

- Assembly라는 프로그래밍 언어는
  - Symbolic expression -> 기계어 (binary expression으로 바꿔준다)
- 즉, ADD A,B 라는 표현을 1000110010100000 으로 바꿔준다.
- i.e. Human language -> Computer language

어셈블리 언어 (Assembler)

```
Swap (int v[], int k)
{ int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

```
Swap:
LSL X10, X1, 3
ADD X0, X0, X10
LDUR X9, [X10,0]
LDUR X11, [X10,0]
STUR X11, [X10,0]
STUR X9, [X10,8]
BR X10
```

```
00000000000010101
01111011010100000
00000000010100100
00000000010101110
10111010100000000
00000001010100000
00000001010111010101
11010100011110000
00001010100000000
00001010111010011
```

Problems with Assembler

- Assembly language 는 다 좋은데, 다 좋은데 생산성이 너무 떨어진다
  - 메모리 하나하나 다 프로그래밍 해야하기 때문...

- 생산성이 좋은 언어를 쓸 수 없을까?
  - 사람의 언어처럼 자연스러운 프로그래밍이 가능해야함.
  - Assembler처럼 메모리 단위 프로그래밍 같은 복잡함이 없어야함.

High-level language vs Low-level language

- 큰 그림에서의 프로그래밍을 하고 메모리 연산을 고려하지 않는 언어 == High-level language
  - 생산성 GOOD
  - C, C++, C#, Python, Java, Javascript, Julia...

- 큰 그림에서의 생산성은 없지만 메모리 연산 등 작은 레벨에서의 동작을 구현한 언어 == Low-level language
  - Assembly, SIMD

컴파일러 / 인터프리터

- High-level 언어도 결국 프로세서 / 메모리를 사용해야하기 때문에 low-level operation이 필요함.

- High-level -> Low-level 변환은 어떻게 하는가?
  - Compiler: High-level script를 low-level script로 번역
    - GCC, Clang, MSVC...

  - Interpreter: High-level 코드 한줄 한줄을 low-level 코드로 통역
    - MATLAB, Python...

다양한 프로그래밍 언어들

- 컴파일러를 사용하는 언어 (프로그램 실행 속도가 빠른 편)
  - C, C++, C#,
- 인터프리터를 사용하는 언어 (유저가 쉽게 사용하는 편)
  - Python, MATLAB
- 하이브리드 언어 (반반)
  - Java

다양한 프로그래밍 언어들

- 빠른 속도를 요구하거나, 적은 메모리로 프로그램을 돌려야할 때 (e.g. 로보틱스, 고성능 서버, 임베디드 칩)
  - C, C++, C#
- 쉽고 빠르게 무언가를 만들어야할 때 (e.g. 웹사이트 개발, 앱 개발, 실험)
  - Python, MATLAB
- 특수 언어
  - Java / Kotlin - 안드로이드 앱 개발
  - Objective C / Swift - iOS 앱 개발
  - MATLAB / Julia - 행렬/수식 계산