

# Chapter I

컴퓨터공학과

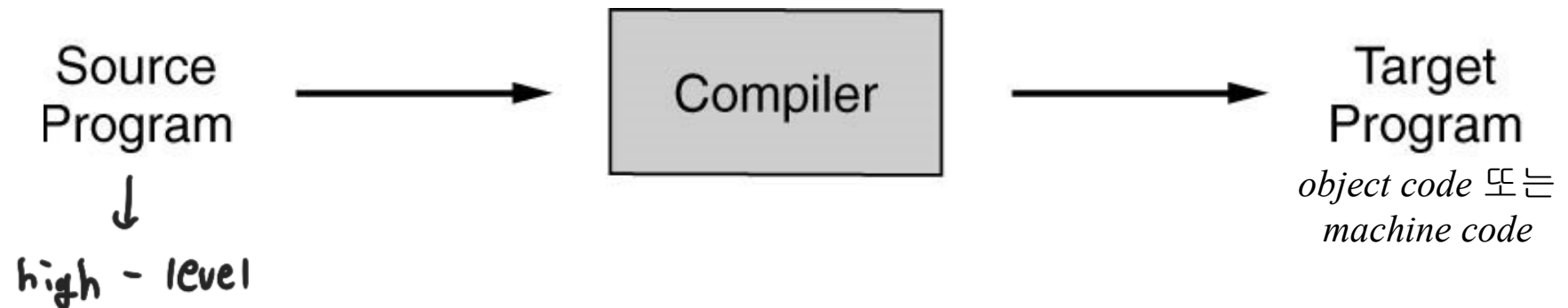
교수 홍 윤 식

*yshong @ inu.ac.kr*

# 컴파일러 정의 (1/2)

---

"A compiler is a computer program which translates programs written in a particular **high-level programming language** into **executable code** for a specific target computer."

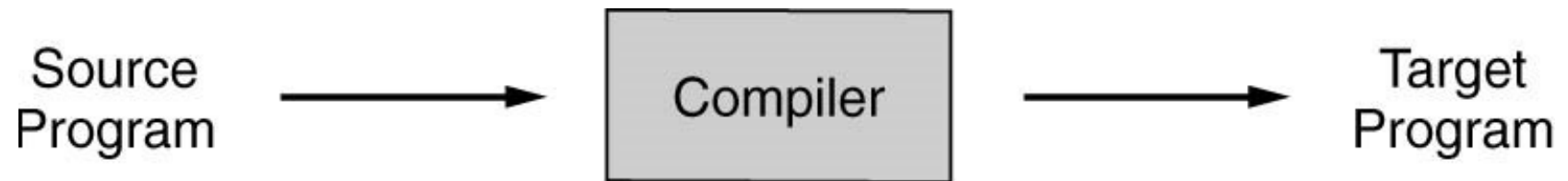


## 컴파일러 정의 (2/2)

---

예: **c compiler** on desktop with Intel **quad-core i7**

- c 소스 코드를 읽어서 quad-core i7의 machine code를 출력

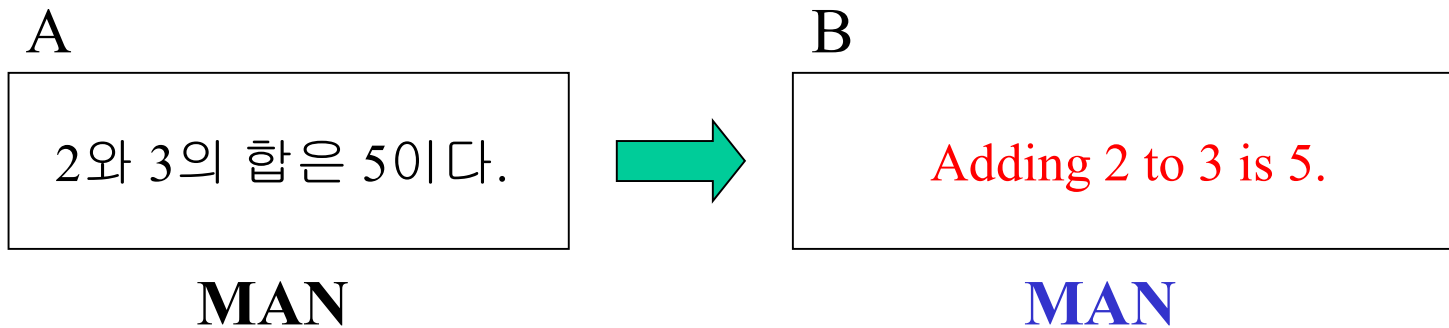


High  $\Rightarrow$  low

# 컴파일(*Compile*)이란 ? (1/3)

---

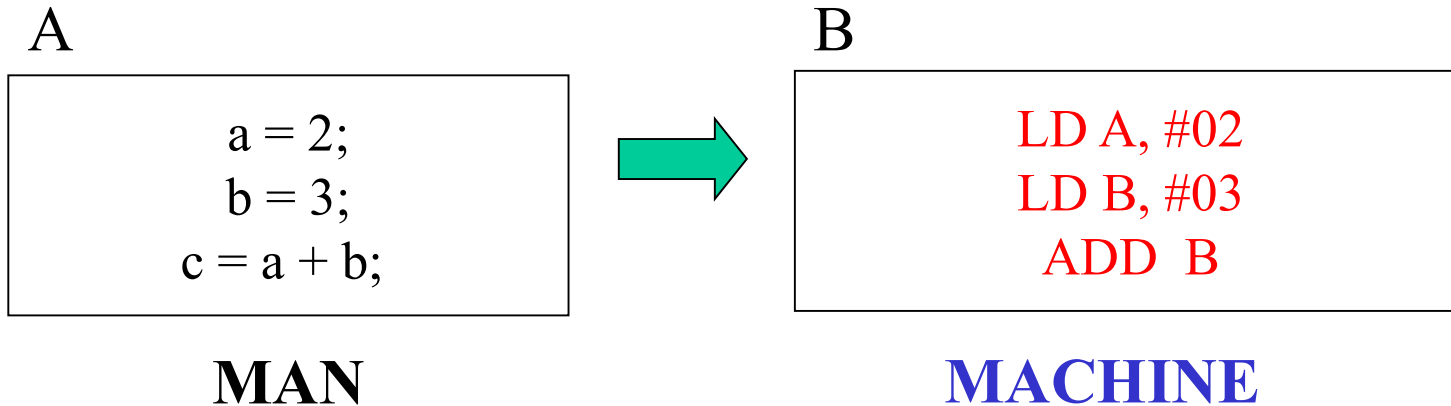
- A와 B는 의미는 같지만 표현이 다르다.



## 컴파일(*Compile*)이란 ? (2/3)

---

- A와 B는 의미는 같지만 표현이 다르다.



# 컴파일(Compile)이란 ? (3/3)

---

- A와 B는 의미는 같지만 표현이 다르다.

A

2와 3의 합은 5.  
2 더하기 3은 5.  
2에다 3을 더하면 5.  
.....



a = 2;  
b = 3;  
c = a + b;

**MAN** – *source code*  
written in **programming language**

B

LD A, #02  
LD B, #03  
ADD B

**MACHINE**

– **executable** machine code

↳ **exe**

# What is Programming Languages?

---

- Programming languages(PL) are *notations*.

$$b^2 - 4ac \rightarrow B**2 - 4.0 * A * C$$

- Fortran(1957)은 수학 기호 (+, \*) 를 사용하여 공식 (*formula*)을 PL형태로 표현

- FOR-TRAN  
→ FORmula TRANslator



IBM 704 mainframe computer

# What is Programming Languages?

---

- Programming languages are *notations for algorithms*.

$1! = 1, 2! = 2 \times 1, 3! = 3 \times 2 \times 1, 4! = 4 \times 3 \times 2 \times 1, \dots$

$1! = 1, 2! = 2 \times 1!, 3! = 3 \times 2!, 4! = 4 \times 3!, \dots$



```
int fact(int n) {  
    if (n <= 0) return 1;  
    return (n * (fact(n-1)));  
}
```

조건문

(conditional expression)

순환(recursion)



# How about Human language?

---

*Chinese vs. English:*

표의(表意)문자

상형(象形)문자

pictorial (WYSIWYG) vs. phonetic  
hieroglyphic vs. alphabetical

표음(表音)문자

알파벳

표기 방식(notation)이 다르다 !

**Glyphic** : 상형 문자의.

**Hieroglyphic** : (고대 이집트의) 상형 문자.

**Phonetic** : 음성(발음)을 나타내는. <참조> **phonic**

**WYSIWYG** : What you see is What you get.

# How about Human language?

---

*Chinese vs. English:*

pictorial (WYSIWYG) vs. phonetic  
hieroglyphic vs. alphabetical

*Japanese vs. English:*

wa-ta-shi-wa ni-hon-go wa-ka-ri ma-sen.  
I Japanese understand don't.

I don't understand Japanese.

문장 구조(structure)가 다르다 !

# Development of Human Language

---

- 르네 에티앙블 - 인류가 지구상에 와서 살다 죽어 간 것이 **100만**년이나 되었지만 문자를 사용하기 시작한 것은 **6,000년**밖에 되지 않았다.
- 한자, 상형문자 → 알파벳(26) → 대중 교육이 가능
  - 아랍어 알파벳(**29자**) → 코란을 기록 → 이슬람교 세력 확장과 함께 널리 퍼짐
  - → 페니키아 알파벳 → 그리스 알파벳(**24자**, 아랍어 알파벳에서 모음을 표기하기 위해 특정 기호를 차용)
- **Please be advised that ~ → PLS B ADVSD THT ~**

# 설형 문자(1/2)

- 생성 배경

- 지형적 요인

- 지정학적으로 개방된 위치
    - 교역이 활발해 지면서 거래 장부가 필요
    - 점토 판의 95%가 영수증, 청구서, 재산 목록, 대추 야자와 보리의 측정 단위

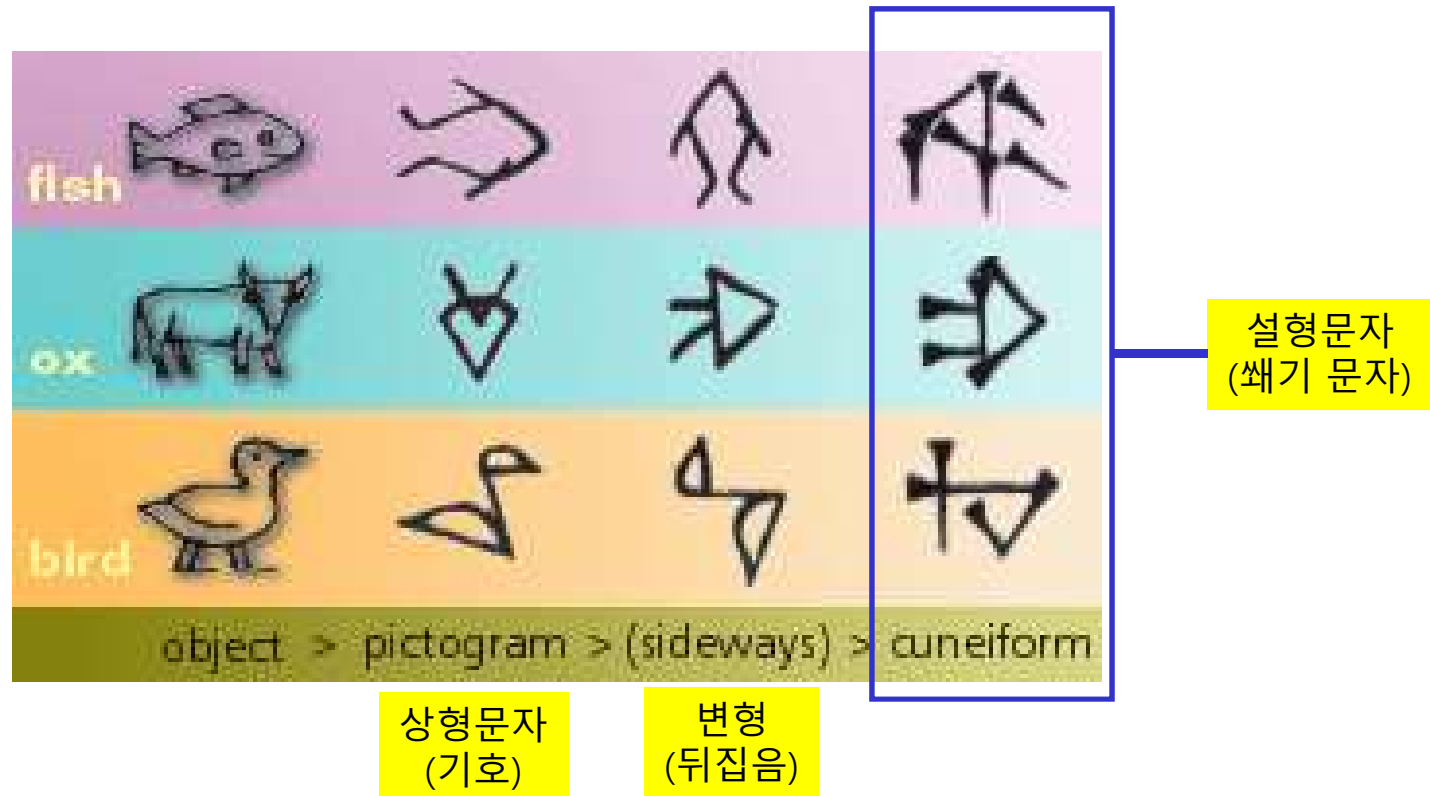


- 사회 환경적 요인

- 농경지는 비옥하나 상습 침수로 인해 점토 이외의 원료가 부족
    - 점토 판에 쐈기풀로 표시



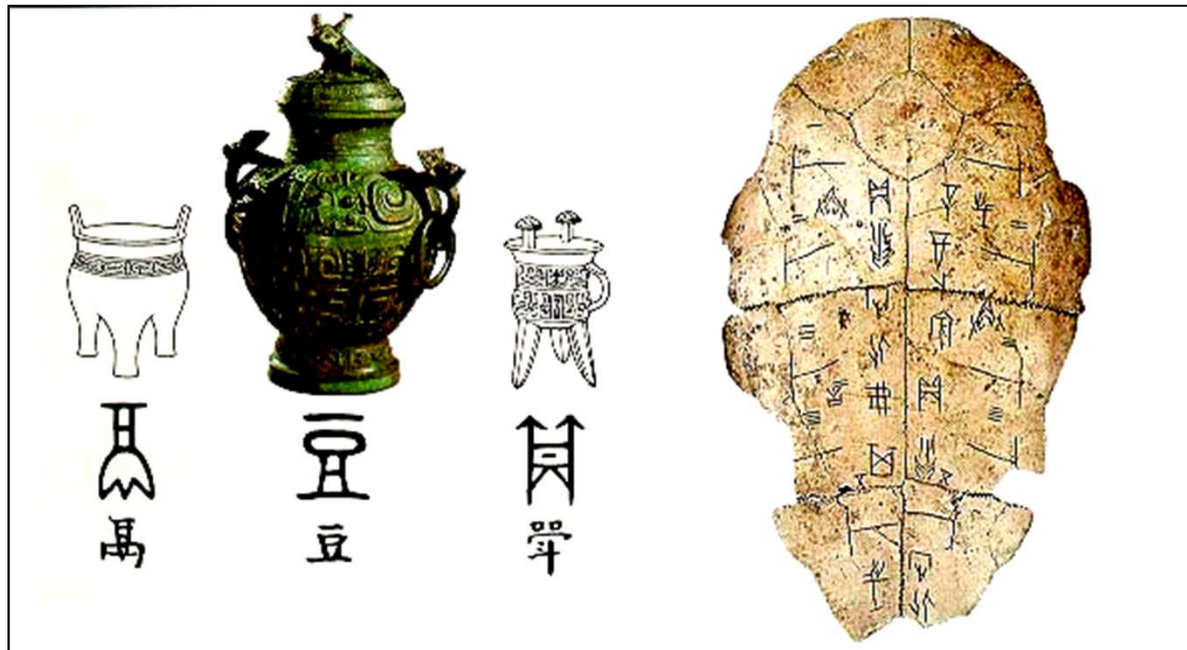
## 설형 문자(2/2)



점토 판에 곡선을 그리기 어렵기 때문에 직선으로 표현

# 갑골 문자

- 중국 은(殷)나라 때 사용했던 문자
  - 특징 : 거북 등이나 소 뼈에 문자를 새김
  - 문자 수 : 대략 3,000자
  - 내용 : 길흉화복을 점치는 복점(卜占)



# Development of Human Language

---

- **1st written language(설형문자) :**  
**Sumerian, 3500 B.C.**
  - Chinese, Shang Dynasty, 2000 B.C
- **1st alphabet: Phoenician, 1100 B.C.; only consonants(자음)**
- **1st complete alphabet: Greek, 800 B.C.; consonants + vowels(모음)**

ENGLISH	GREEK
a A	$\alpha$
b B	$\beta$
d D	$\delta \Delta$
f F	$\phi \Phi$
g G	$\gamma \Gamma$
l L	$\lambda \Lambda$
s S	$\sigma \Sigma$
w W	$\omega \Omega$

```

0000100100101110011001100110100101101100011001010X
00101111001100011001000100000101001100111011000110:
01100100000101110001110100000101000101110011100110:
0110010101111000011101000010001000001010000010010X
0010111001100111011011000110111101100010011000010:
0111100101110000011001010000100100100000011011010:
0110100101101111011011100000101000001001001011100:
0110100101101110001110100000101000001001001000010X
00100000000110000000010100000100101110011011000010:
00111000000101100001001010111001101110000000001010X
0101010101000101001000110010000000011000100001010X
0000101000001001011100110111010000100000001001010:
0101110100001010000010010110110101101111011101100X

```



# How about This?

---

```
main:
    !#PROLOGUE# 0
    save %sp,-128,%sp

    !#PROLOGUE# 1
    mov 1,%o0
    st %o0,[%fp-20]
    mov 2,%o0
    st %o0,[%fp-24]
    ld [%fp-20],%o0
    ld [%fp-24],%o1
    add %o0,%o1,%o0
    st %o0,[%fp-28]
    mov 0,%i0
    nop
```

# Is This Better Now?

---

```
#include <stdio.h>

int main()
{
    int x, y, z;

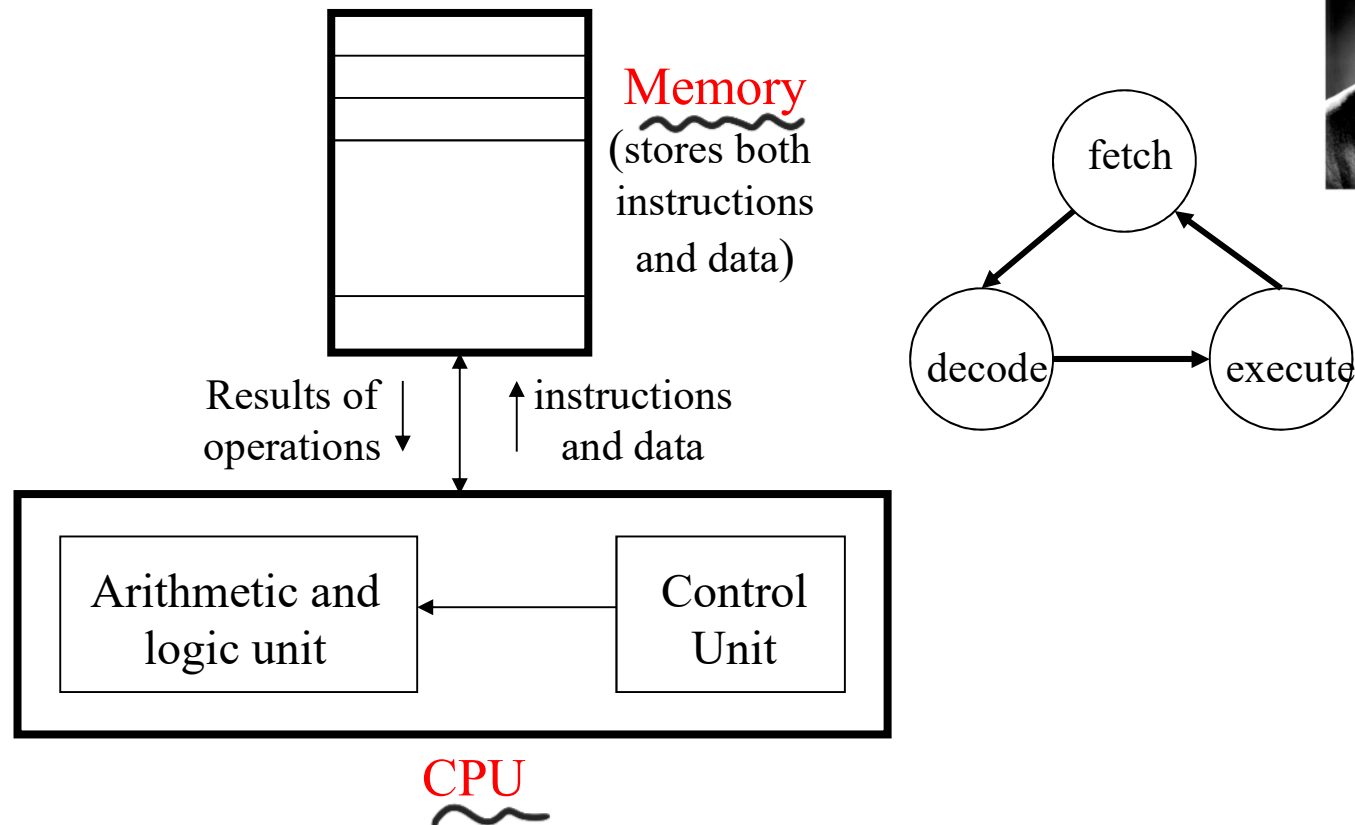
    x = 1;
    y = 2;
    z = x+y;

    return 0;
}
```

# Why Compilers? A Brief History

존 폰 노이만

- Stored program computer by J. von Neumann (1940년대 말)
  - Necessary to write *sequence of codes*



# Why Compilers? A Brief History

---

- Machine language : Assembly language

`c7 06 0000 0002 → MOV x, 2`

- A concise, *machine-independent* form

`x = 2`

- FORTRAN language and its compiler by [John W. Backus](#) (1950년대)

# Why Compilers? A Brief History

---

- *Classification* of languages according to the complexity of their grammars by Noam Chomsky
  - Type 0, Type 1, Type 2, and Type 3 grammars
  - Context-free grammar (*Type 2*)

Who's Chomsky? → <http://www.chomsky.info>

