컴퓨터 구조

3장. 명령어

발표자 : 정승원 | CT Lab | jswfs1284@naver.com | 2024.01.03

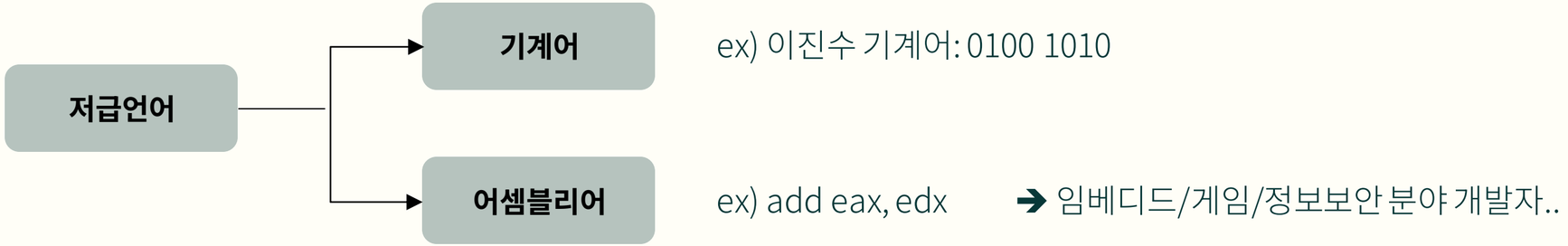
01 소스코드와 명령어

KEY WORD 고급언어, 저급언어, 기계어, 어셈블리어, 컴파일 언어, 인터프리터 언어

01

고급 언어, 저급 언어

- 고급 언어: C, JAVA, Python, …. 사람들을 위한 언어.
- 저급 언어: 컴퓨터가 이해하고 실행할 수 있는 언어. 어셈블리어의 경우 프로그램 실행 과정을 추적할 수 있음.



01

컴파일 언어, 인터프리터 언어

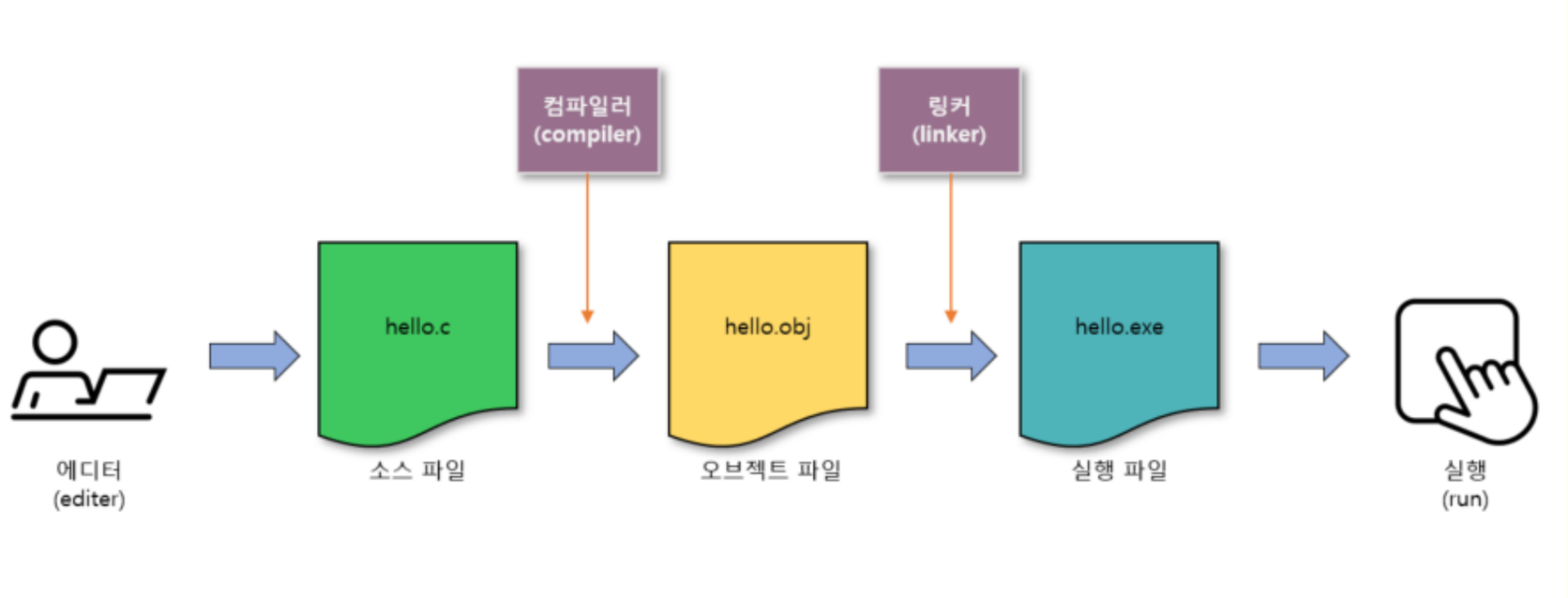
- **컴파일 언어**: 컴파일러가 소스 코드를 **object 코드(저급언어)로 변환**하여 실행. (ex. C언어)
  파일에 오류 있으면 실행 **불가**.
- **인터프리터 언어**: 인터프리터가 소스 코드를 **한 줄씩 저급언어로 변환하여 실행**. (ex. JAVA, 파이썬)
  파일에 오류 있어도 실행 **가능**. 단, 오류 없는 줄까지만 실행. 컴파일 언어보다 실행 속도 **느림**.

컴파일 언어

인터프리터 언어

obj파일, exe파일

(C언어 컴파일 과정)

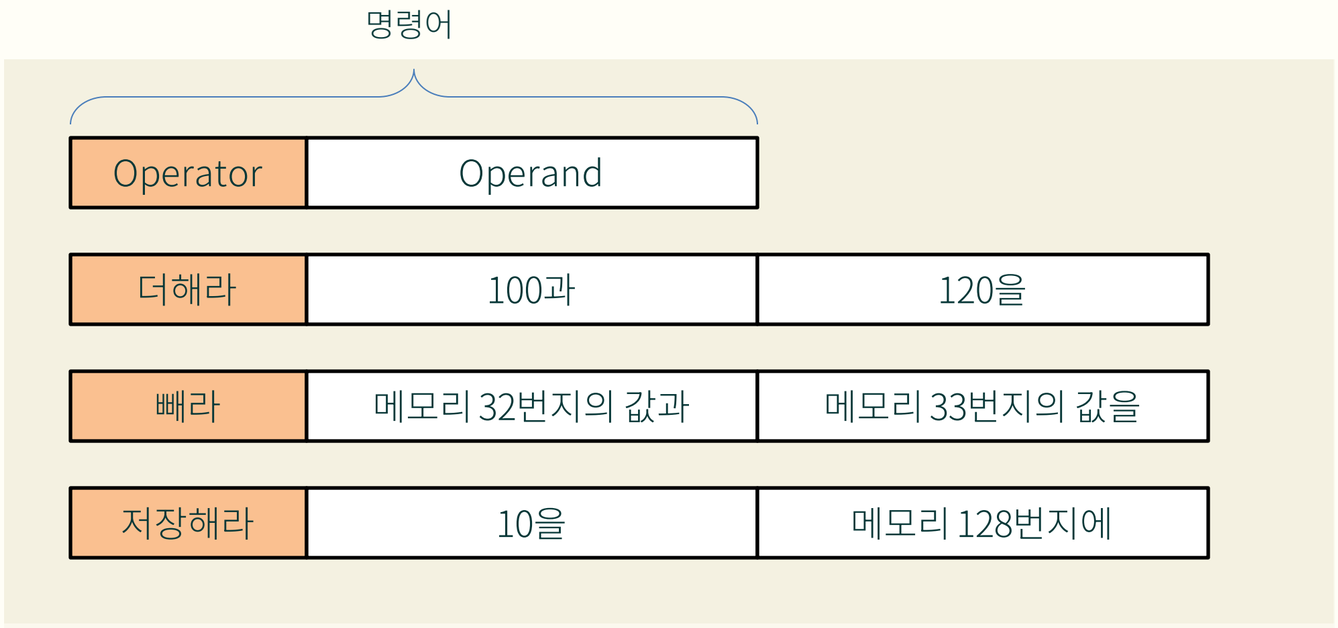


이미지 출처: https://byunghyun23.tistory.com/2

02 명령어의 구조

KEY WORD 명령어, 연산코드, 오퍼랜드, 주소 지정 방식

Operator 연산자/연산코드, Operand 피연산자

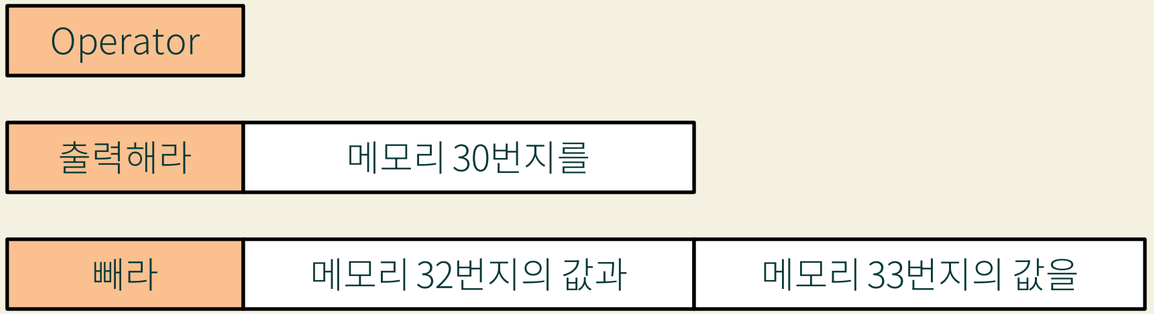



```
push    rbp
mov     rbp, rsp
mov     DWORD PTR [rbp-4], 1
mov     DWORD PTR [rbp-8], 2
mov     edx, DWORD PTR [rbp-4]
mov     eax, DWORD PTR [rbp-8]
add     eax, edx
mov     DWORD PTR [rbp-12], eax
mov     eax, 0
pop     rbp
ret
```

어셈블리어 예시
(붉은 글씨가 Operator, 검은 글씨가 Operand)

02

Operand 피연산자



0-주소 명령어

1-주소 명령어

2-주소 명령어

3-주소 명령어

…

02

Operator 연산자

- **연산코드의 종류는 다양하며, 기본적인 연산자는 다음 네 가지 유형:**
  - ① 데이터 전송
  - ② 산술/논리 연산
  - ③ 제어 흐름 변경
  - ④ 입출력 제어

- **CPU마다 명령어의 종류와 생김새가 다름**

02

Operator 연산자

- **대표적인 연산 코드 종류**

데이터 전송

- MOVE: 데이터를 옮겨라
- STORE: 메모리에 저장하라
- LOAD (FETCH): 메모리에서 CPU로 데이터를 가져와라
- PUSH: 스택에 데이터를 저장하라
- POP: 스택의 최상단 데이터를 가져와라

note 스택에 대한 개념은 100쪽 〈좀 더 알아보기〉에서 설명합니다.

제어 흐름 변경

- JUMP: 특정 주소로 실행 순서를 옮겨라
- CONDITIONAL JUMP: 조건에 부합할 때 특정 주소로 실행 순서를 옮겨라
- HALT: 프로그램의 실행을 멈춰라
- CALL: 되돌아올 주소를 저장한 채 특정 주소로 실행 순서를 옮겨라
- RETURN: CALL을 호출할 때 저장했던 주소로 돌아가라

note 프로그래밍 언어를 학습한 독자들은 함수를 떠올리면 됩니다. CALL과 RETURN은 함수를 호출하고 리턴하는 명령어입니다.

산술/논리 연산

- ADD / SUBTRACT / MULTIPLY / DIVIDE: 덧셈 / 뺄셈 / 곱셈 / 나눗셈을 수행하라
- INCREMENT / DECREMENT: 오퍼랜드에 1을 더하라 / 오퍼랜드에 1을 빼라
- AND / OR / NOT: AND / OR / NOT 연산을 수행하라
- COMPARE: 두 개의 숫자 또는 TRUE / FALSE 값을 비교하라

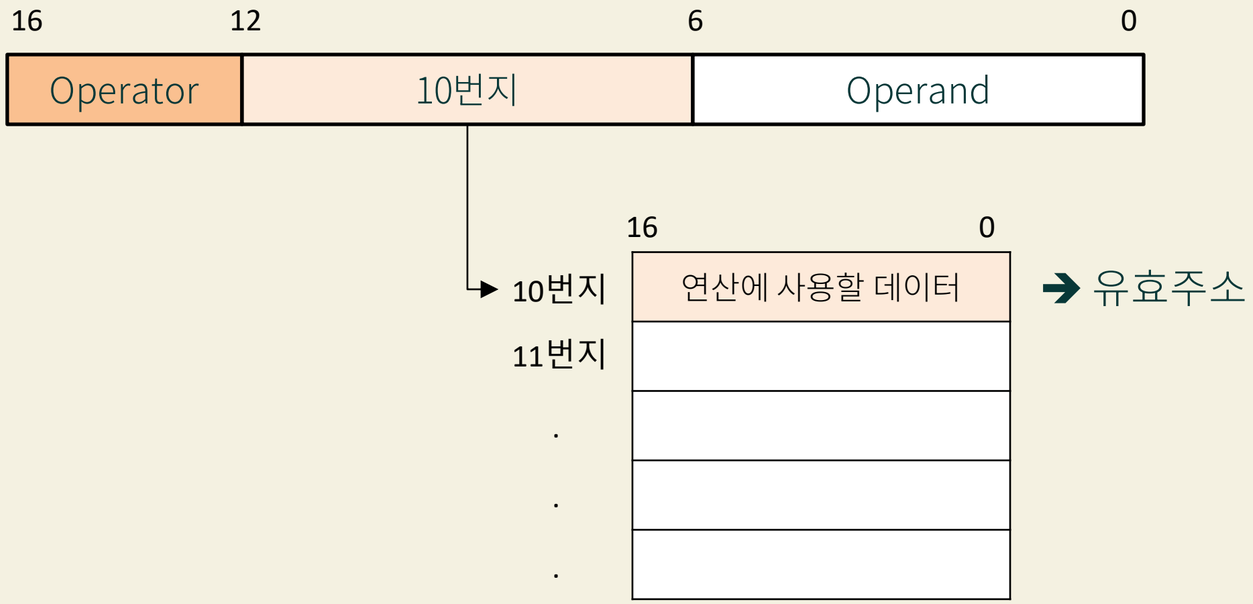
입출력 제어

- READ (INPUT): 특정 입출력 장치로부터 데이터를 읽어라
- WRITE (OUTPUT): 특정 입출력 장치로 데이터를 써라
- START IO: 입출력 장치를 시작하라
- TEST IO: 입출력 장치의 상태를 확인하라

02

주소 지정 방식

- **예) 명령어의 길이가 16이고, 연산자의 길이가 4인 경우…, 2-주소 명령어는 ▼**



- **주소 지정 방식** addressing mode **: 연산에 사용할 데이터 위치를 찾는 방법**

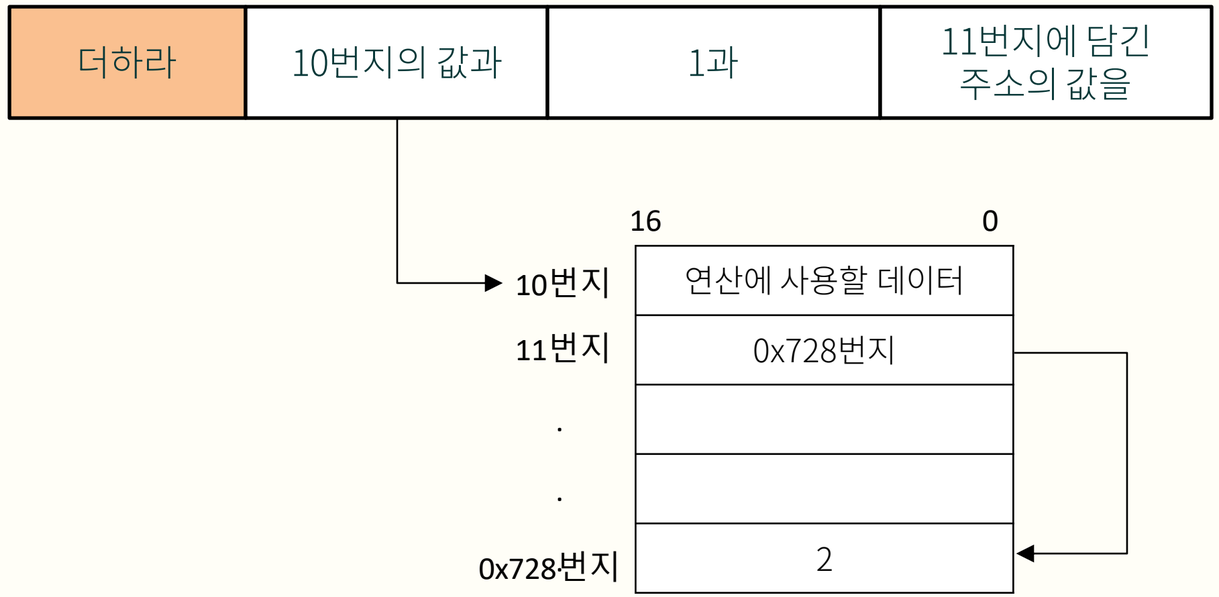
02

주소 지정 방식

① **즉시 주소 지정 방식**: 연산에 사용할 데이터

② **직접 주소 지정 방식**: 유효 주소 (메모리 주소)

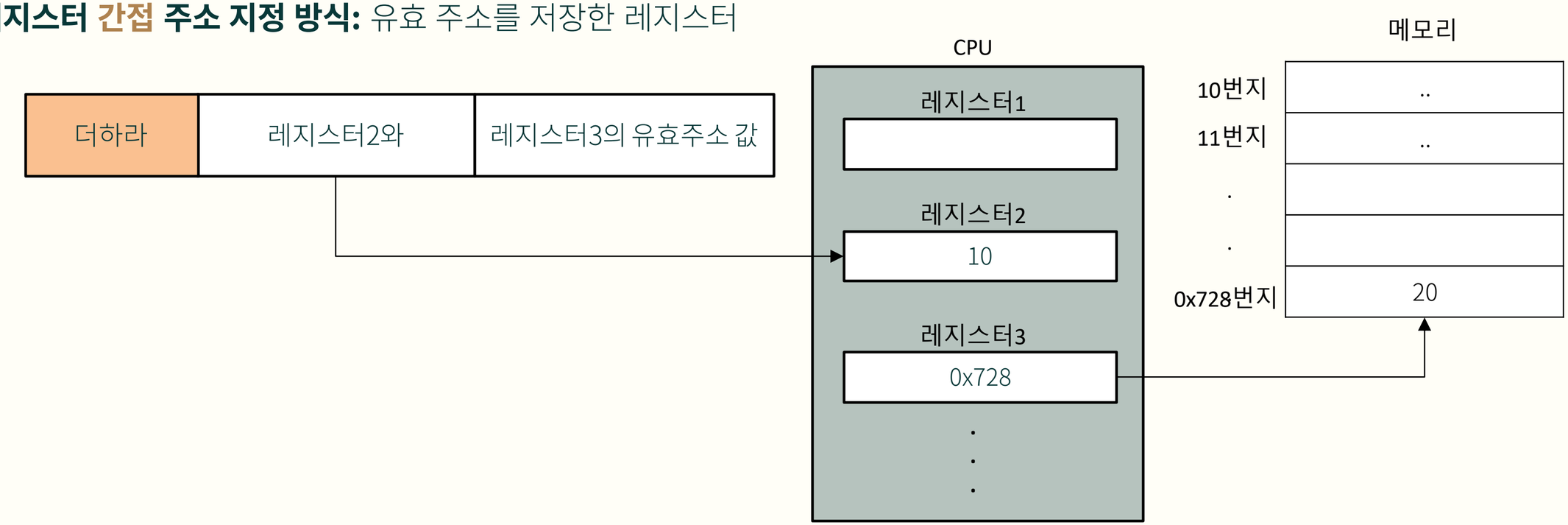
③ **간접 주소 지정 방식:** 유효 주소의 주소



주소 지정 방식

④ **레지스터 주소 지정 방식:** 유효 주소 (레지스터 주소)

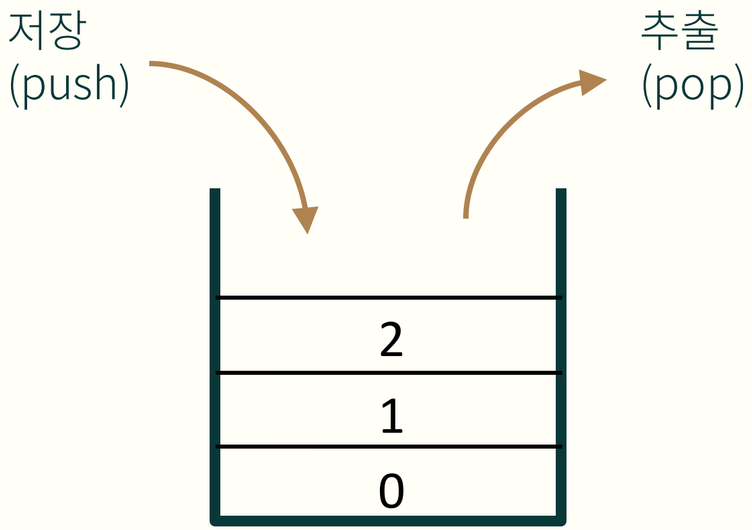
⑤ **레지스터 간접 주소 지정 방식:** 유효 주소를 저장한 레지스터



스택, 큐

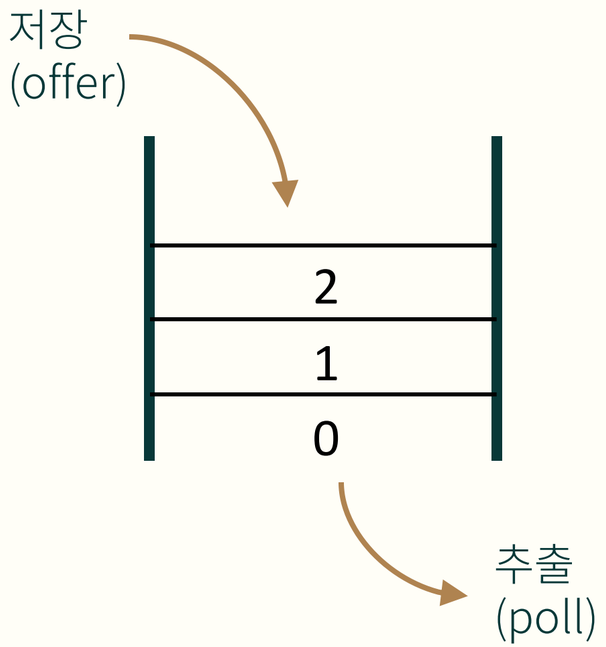
스택 Stack

: LIFO (Last in First Out)



큐 Queue

: FIFO (First in First Out)



02

오늘 배운 것

01. 고급 언어와 저급 언어의 차이를 이해하였다.

02. 컴파일 언어와 인터프리터 언어의 차이를 이해하였다.

03. 명령어를 구성하는 Operator(또는 Operation code)와 Operand에 대해 학습하였다.

04. 명령어의 주소 지정 방식에 대해 학습하였다.

감쟘다

다음 발표 일정 : **4장, 내일 19:00 주히방**