명령어 세트

대전 공통3반 박재현

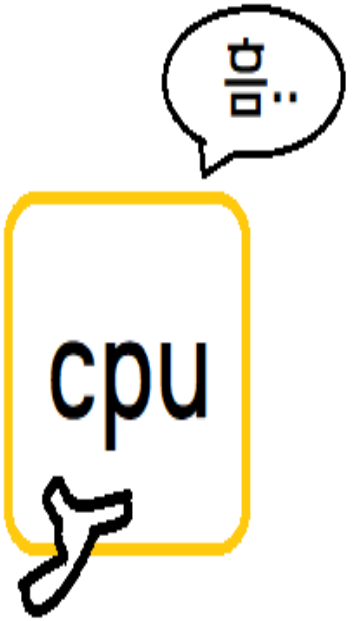
명령어 세트?!?

컴퓨터의 구조

→ 명령어 집합 종류, 수가 상이함

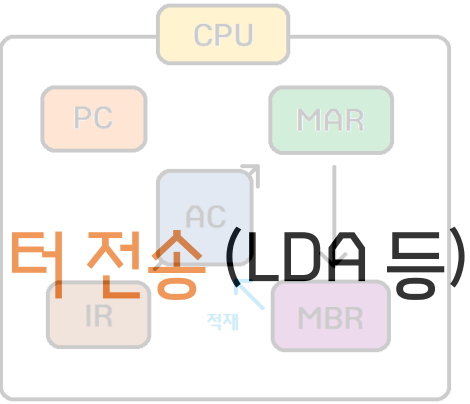
명령어 세트

- 명령어의 종류
- 명령어 형식
- 주소지정 방식



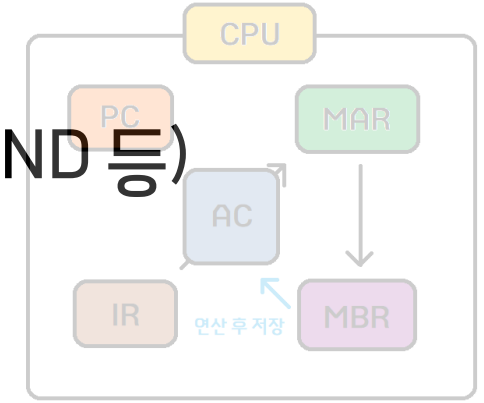
명령어 종류 – 크게 3가지!

데이터 전송 명령어



- 레지스터-레지스터, 레지스터 – 기억장치, 기억장치- 기억장치 데이터 전송 (LDA 등)

데이터 처리 명령어



- 컴퓨터에 연산 능력을 부여해줌 (산술 및 논리 연산, ADD, AND 등)

프로그램 제어 명령어

- 명령어 실행 순서를 변경하는 연산(???)

다시 한번!! 명령어는 크게 두가지로 구성됩니다.

연산 코드(operation code)

- CPU가 수행할 연산을 지정

- LDA, ADD 등의 동작

이제 명령어 형식을 알아볼까요?

오퍼랜드(operand)

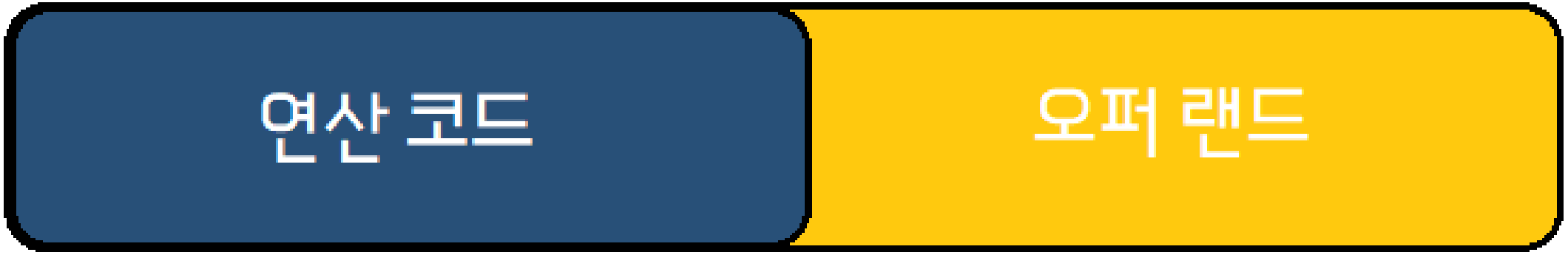
- 명령어가 사용할 데이터가 저장되어 있는 주소

명령어 형식 – 0 주소 명령어

(ex: PUSH, POP)

STACK 구조의 컴퓨터에서 사용

명령어 형식 – 1 주소 명령어



(ex: ADD A)

AC에 의해 데이터 처리

명령어 형식 – 2 주소 명령어

(ex: ADD A, B)

가장 일반적인 경우!

오퍼랜드에는 레지스터나 기억장치 주소 지정

(여기에서 레지스터 만으로 구성된 경우 RISC 명령어!)

명령어 형식 – 3 주소 명령어

(ex: ADD A, B, C)

연산 결과를 저장하기 위한 주소 하나!

프로그램의 길이를 짧게 해줘요!

~~하지만 명령어 길이가 겁나 길어집니다.~~

주소지정 방식 (addressing mode)

연산에 사용될 데이터가 기억장치의 어디에 위치하는지를 지정하는 방법

제한된 명령어 비트로 오퍼랜드를 지정하기에...

→ 다양한 방식이 제안되어 있다!!

잘 이용하면 자원을 효율적으로, 명령어를 짧게 할 수 있다

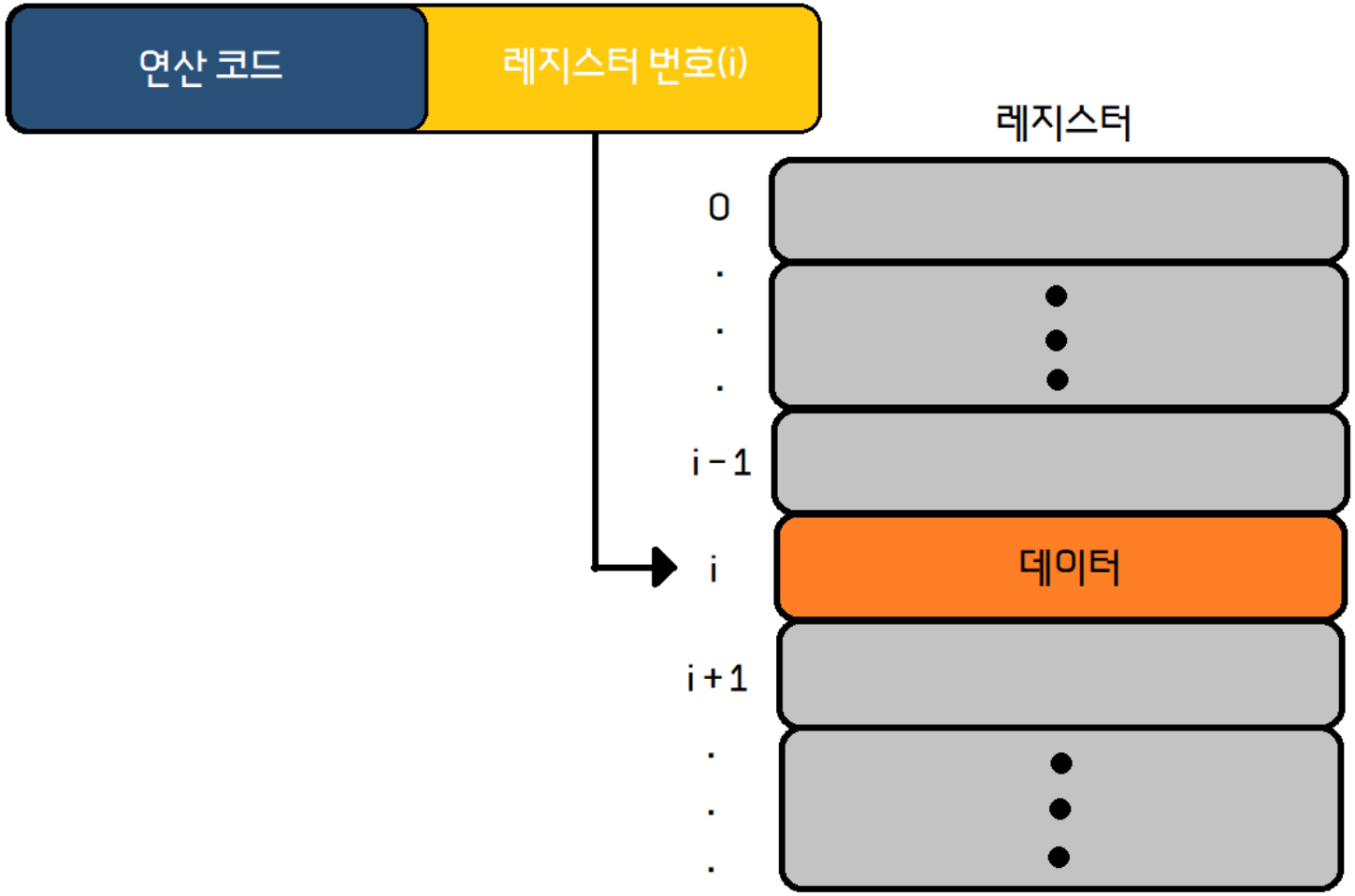
주소지정 방식 (addressing mode)

즉시 주소지정 방식 (immediate addressing mode)

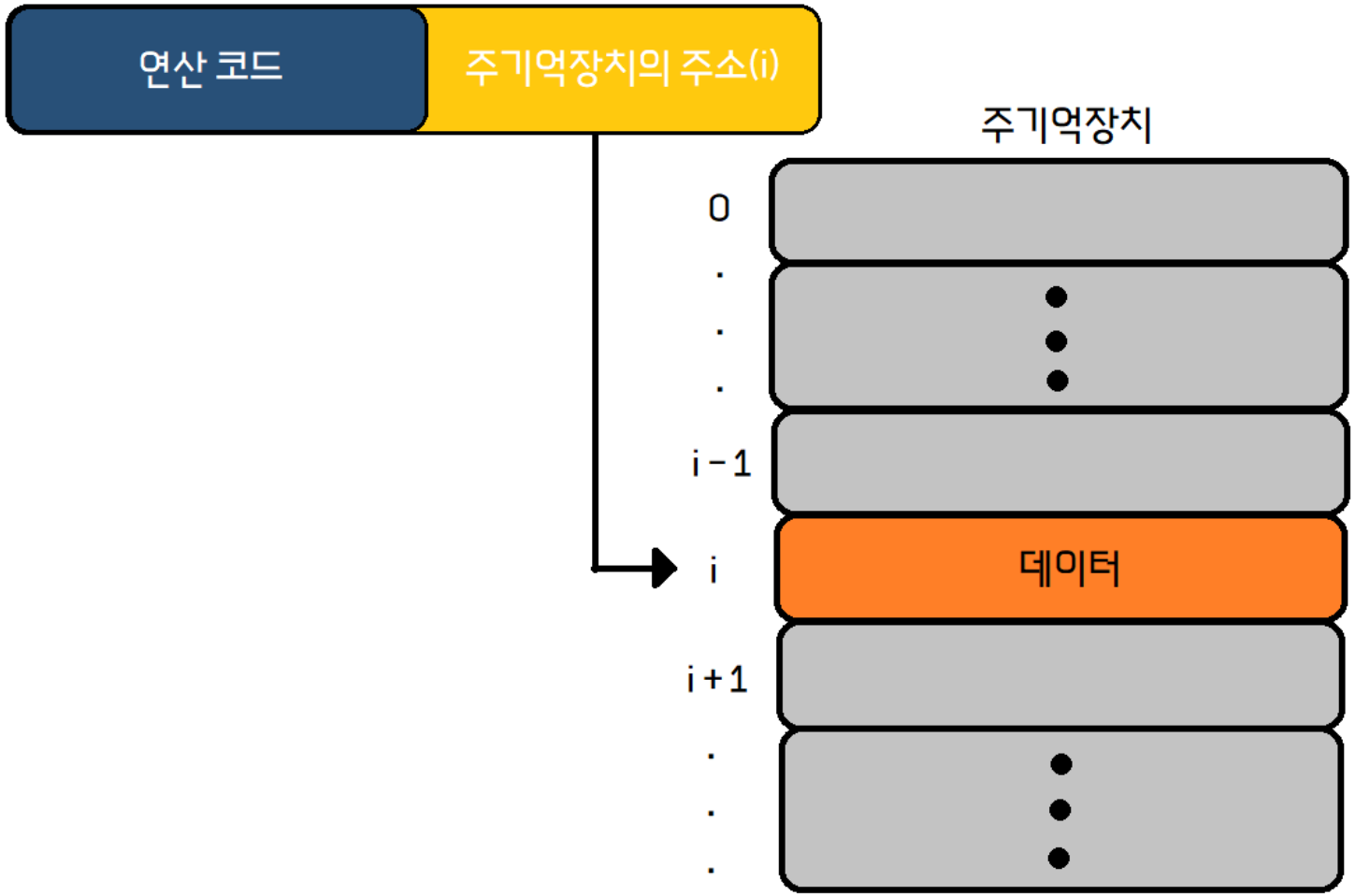
오퍼랜드의 내용 = 실제 데이터

- 변수 초기값 설정, 상수를 사용할 때 사용!
- 처리속도가 가장 빠름!

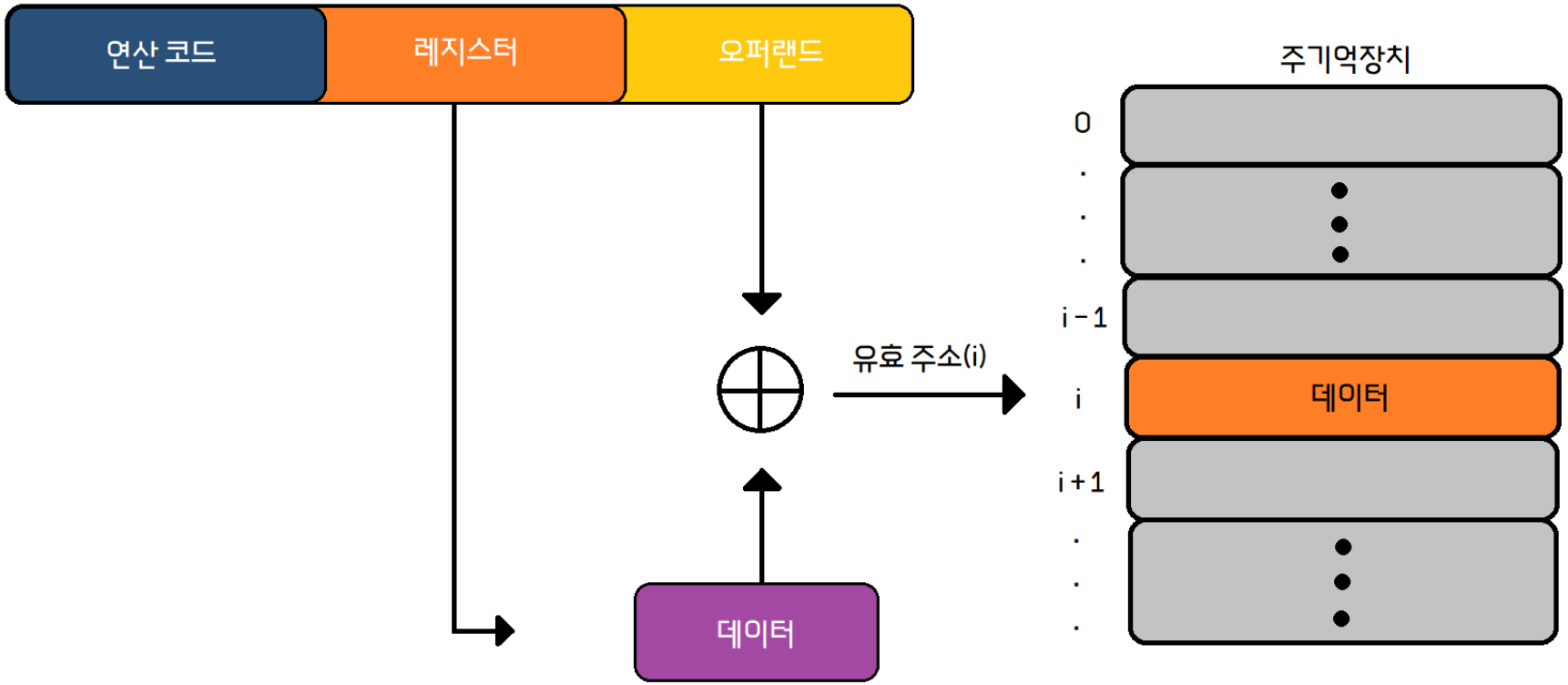
레지스터 주소지정 방식 (register addressing mode)



직접 주소지정 방식 (direct addressing mode)



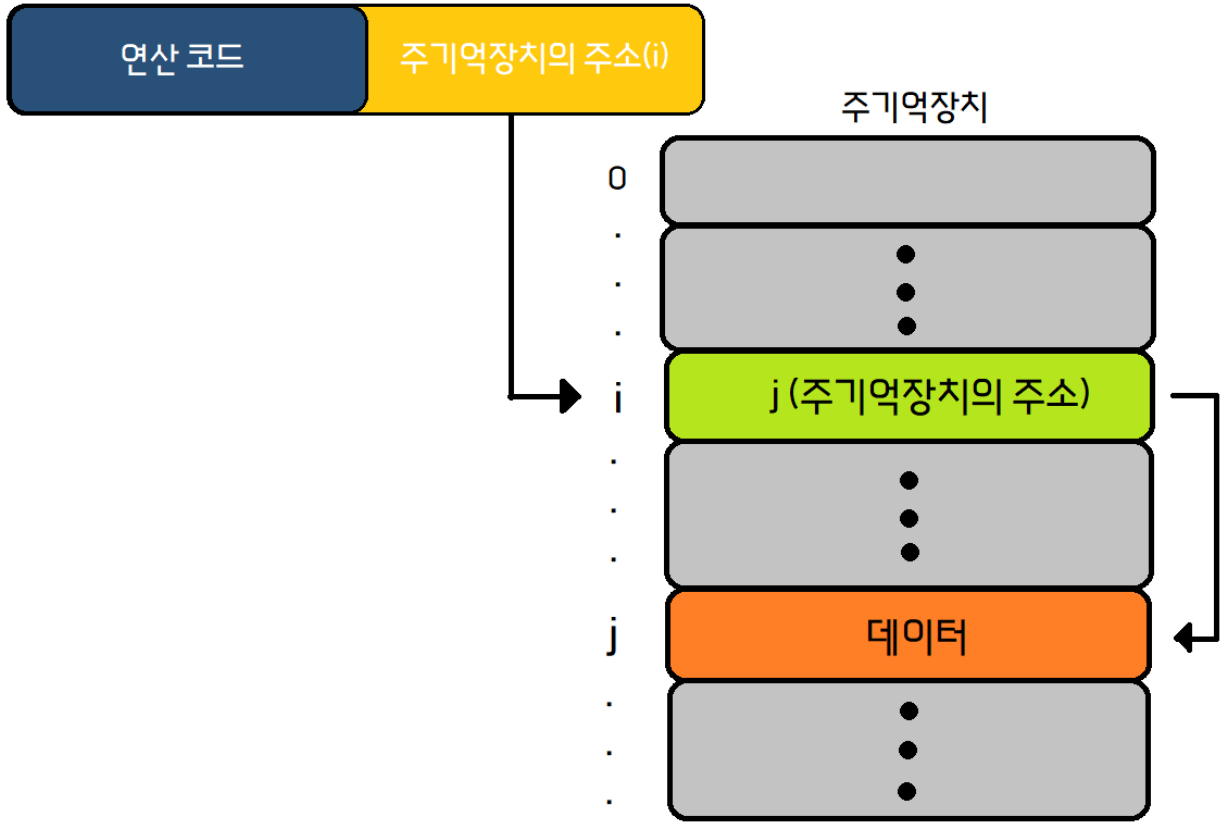
변위 주소지정 방식 (displacement addressing mode)



여기서 레지스터는 프로그램 카운터 (PC), 인덱스 레지스터를 씁니다.

- PC의 경우 상대 주소지정 방식 (relative addressing mode)
- 인덱스 레지스터의 경우 인덱스 주소지정 방식(indexed addressing mode)이라고 합니다.

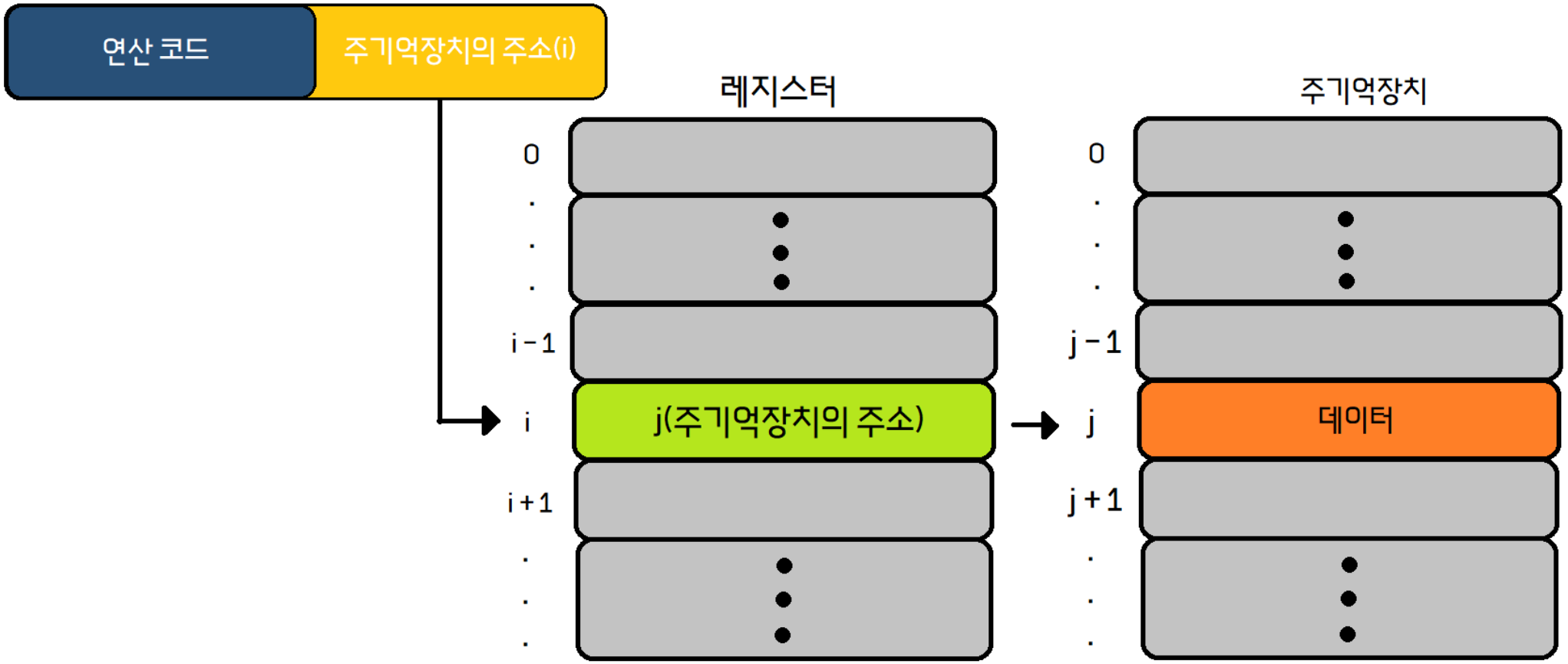
간접 주소지정 방식 (indirect addressing mode)



참조된 데이터가 스택 형태일 경우, 그 다음 데이터의 주소는 따로 지정하지 않아도 알 수 있으므로

묵시적 주소지정 방식(implied addressing mode)이라고 합니다.

레지스터 간접 주소지정 방식



이제 CPU는 명령어를 수행하게 됩니다.