기계어 대신 어셈블리 언어를 사용하는 이유

* 기계어의 모든 장점을 가지고 있음
* opcode를 읽기 편함
* 바이트로 표현된 opcode보다 적은 opcode만 암기하면 된다
  + 인자에 따라 주소 지정 모드가 결정되기 때문
* 프로그래머의 실수를 막을 수 있는 기능들이 있음(어셈블러가 지원)
  + “변수명”
  + 라벨
  + 매크로
  + 등

고수준 언어 대신 어셈블리어를 사용하는 이유

* 일반적으로 없음
* 하지만 예외적인 상황들이 있음
  + 일부 코드를 어셈블리어로 작성해 컴파일러보다 최적화를 더 잘하고 싶음
  + 컴파일러가 지원 안 하는 하드웨어 기능을 사용하고 싶음
  + 코드가 오작동하는 이유가 컴파일러 버그 때문이란 걸 확인하고 싶음
  + 컴파일러 버그를 우회하고 싶음
  + 고수준으로 작성한 언어가 실제 어떻게 동작하는 지 학인하고 싶음
  + 커스텀 하드웨어에서 간단한 제어 프로그램을 작성하고 싶음

어셈블리어의 분류

언매니지드 vs 매니지드 -> 언매니지드

고수준 vs 저수준 -> 저수준 (엄밀히 말하면, 기계가 바로 이해할 수 없기 때문에 고수준 언어라고도 할 수 있다)

컴파일 언어 vs 인터프리터 언어 -> 어셈블이 필요한 언어. 컴파일에 더 가까움

강타입 vs 약타입 -> 둘 다 아님! 타입이 없음.

절차적 vs OOP vs 함수형 -> 절차적 언어에 가깝지만 엄밀하게는 아님. 명령형 프로그램의 일부이다.

어셈블리어와 헬로 월드?

새로운 언어를 배울 때 가장 처음 작성하는 프로그램

화면에 ‘Hello World’를 출력하며 모든 세팅이 올바른지 확인하는 방법

하지만 우리는 어셈블리어로 헬로 월드를 작성할 수 없음

화면이 없기 때문에

그 대신 전에 봤던 기계어 코드를 어세블리어로 작성해 실행해 볼 계획

1. 어셈블리어로 코드 작성
2. 기계어로 어셈블
3. 어셈블된 결과인 이진 파일을 ROM에 넣어 실행

헬로 16코드 작성 및 어셈블

어셈블러로는 dasm or vasm 사용

코드 작성 (VS code)

clc

lda #$0A

adc #$06

sta $2200

-f3 옵션은 다른 정보를 포함하지 않고, 순수한 bin파일만 변환

opcode라고 했었던 부분을 Mnemonic이라 함

Mnemonic을 찾을 수 없다는 에러

1. Mnemonic임을 표시하기 위해 들여쓰기 필요 dasm assembler 제약사항
2. ISA 지정이 필요함

다음 지시문 사용

.PROCESSOR 6502 // ISA 지정

.ORG $8000 // 시작 주소 지정

.PROCESSOR 6502

.ORG $8000

clc

lda #$0A

adc #$06

sta $2200

.ORG $FFFC,0 // 0은 fill value 지정되지 않은 주소의 값을 0으로 채움

.WORD $8000

.WORD $0000

니모닉과 주소지정 모드

구문 (statement)

한줄 구문

코드 블록 구문

어셈블리어의 구문

명령어(instruction)

지시문(direction)

매크로(macro)

명령어의 구조

텍스트, 스크린샷, 의류, 디자인이(가) 표시된 사진

자동 생성된 설명

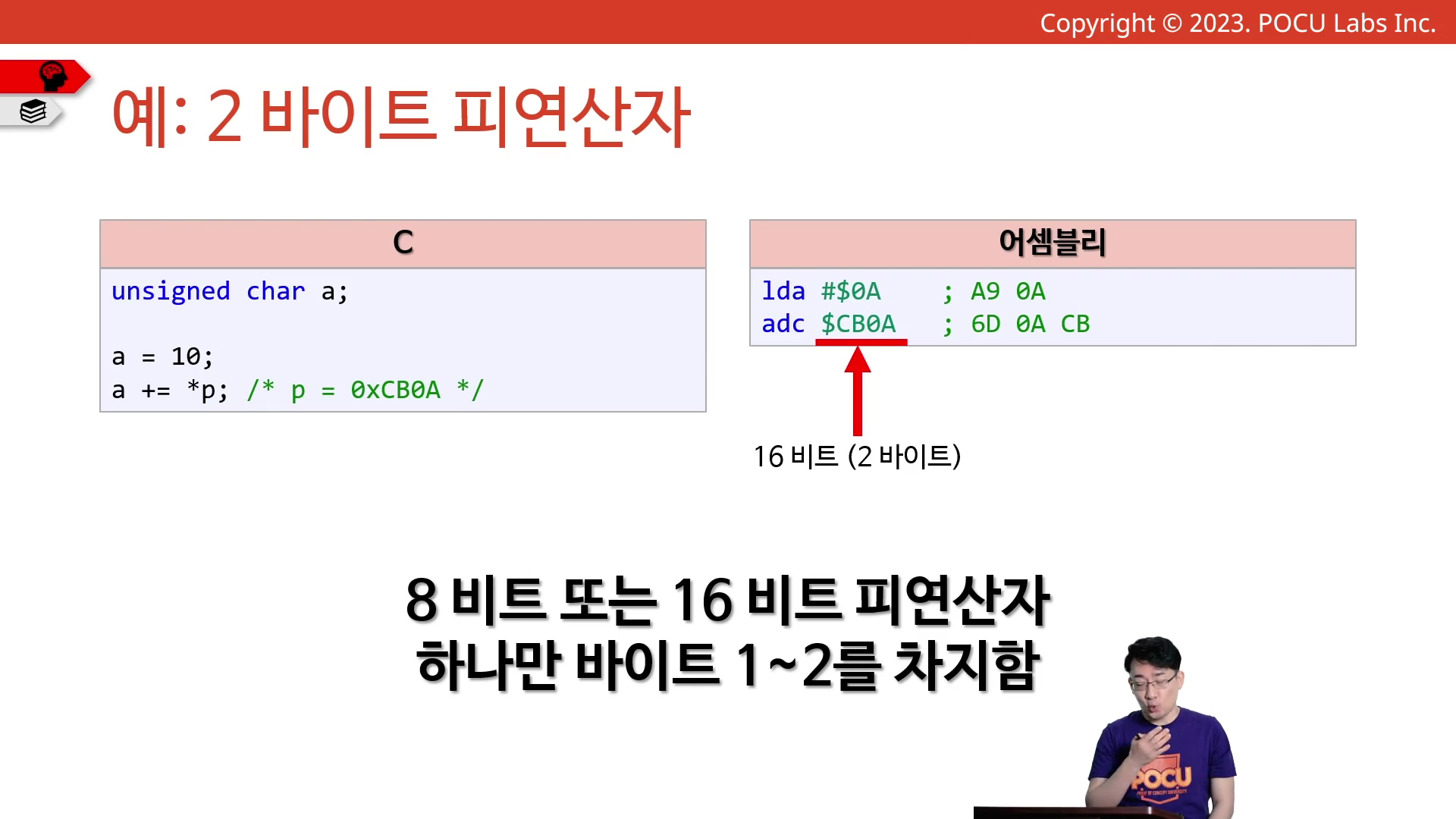
명령어는 니모닉 + 피연산자

텍스트, 스크린샷, 의류, 사람이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 사람, 인간의 얼굴이(가) 표시된 사진

자동 생성된 설명



텍스트, 스크린샷, 의류, 사람이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 의류, 사람이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 사람, 의류이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 사람, 의류이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 의류, 사람이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 의류, 사람이(가) 표시된 사진

자동 생성된 설명

A 레지스터의 값을 하드웨어 적으로 직접 확인할 수 있는 방법은 없다.

A 레지스터의 값을 다시 출력하도록 해야한다.

POBS Emulator에서는 가능하다

텍스트, 스크린샷, 사람, 의류이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 사람이(가) 표시된 사진

자동 생성된 설명

텍스트, 컴퓨터, 스크린샷, 사람이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 의류, 사람이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 웹사이트, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

텍스트, 만화 영화, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

텍스트, 만화 영화, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 만화 영화이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 만화 영화이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 도표, 사람이(가) 표시된 사진

자동 생성된 설명

사실 이건 똑똑한 하드웨어 설계자들이 최적화를 해줬기 떄문이다

이것이 바로 CPU 파이프라인

CPU 파이프라인이란?

CPU의 명령어 처리 속도를 향상하는 하드웨어 설계

단일 코어가 둘 이상의 명령어를 동시에 실행할 수 있게 해 줌

명령어 수준 병렬성(instruction-level parallelism)

텍스트, 스크린샷, 사람, 의류이(가) 표시된 사진

자동 생성된 설명

텍스트, 의류, 스크린샷, 사람이(가) 표시된 사진

자동 생성된 설명