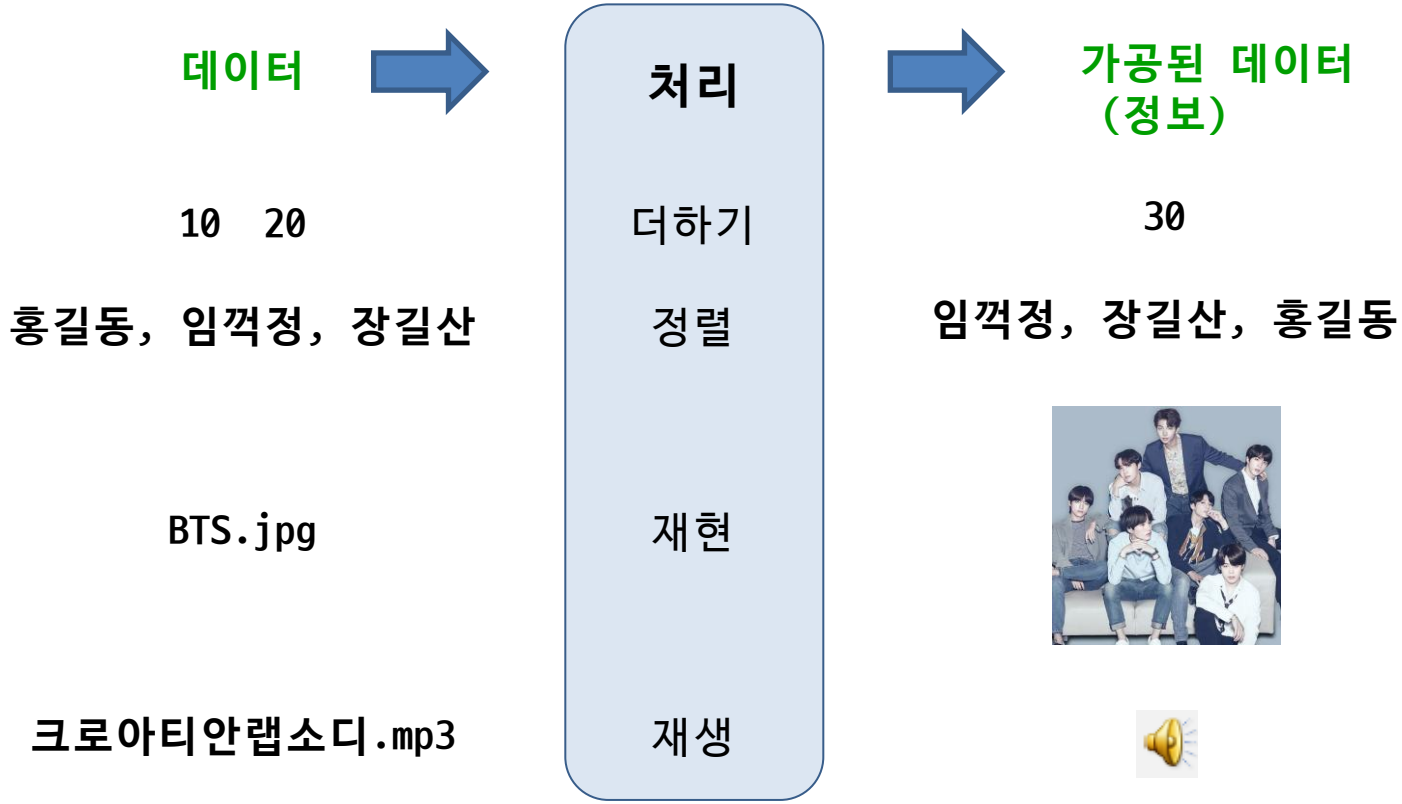
Introduction to Computer System Review

Jo, Heeseung

컴퓨터의 기능

컴퓨터는 데이터를 입력받아 처리한 후 출력하는 기계 이다.



컴퓨터의 사용 목적

컴퓨터가 어떤 작업을 하려면?

- 작업을 수행하기 위한 알고리즘을 만들고, 이를 컴퓨터가 처리할 수 있는 형식으로 표현
- 알고리즘 : 주어진 일(또는 문제)을 처리하여 원하는 결과(output)를 얻기까지의 과정을 명확(明確)하고 단계(段階)적으로 서술한 것

컴퓨터는 잘 시키는 일만 잘 할 수 있다

- 어떻게 잘 시킬 수 있을까?

계산기 vs. 컴퓨터

컴퓨터(computer)는 단순히 계산(compute)만하는 기계가 아니다

- 계산기는 정해진 기능만을 수행
- 계산기는 기능을 변경할 수 없음

컴퓨터는 프로그램을 통해 다양한 작업 가능

- 프로그램 : 알고리즘을 컴퓨터가 이해 가능하게 표현하여 구현한 것
- 현대적인 의미에서의 컴퓨터는 프로그램(명령어들의 리스트)에 따라 데이터를 처리하는 기계
- 컴퓨터는 프로그램을 실행하는 기계

컴퓨터는 데이터를 입력받아 처리한 후 출력하는 기계 인데,
프로그램을 통해 다양한 작업이 가능(Re-programmable) 하다.

컴퓨터의 구성 : 하드웨어 & 소프트웨어

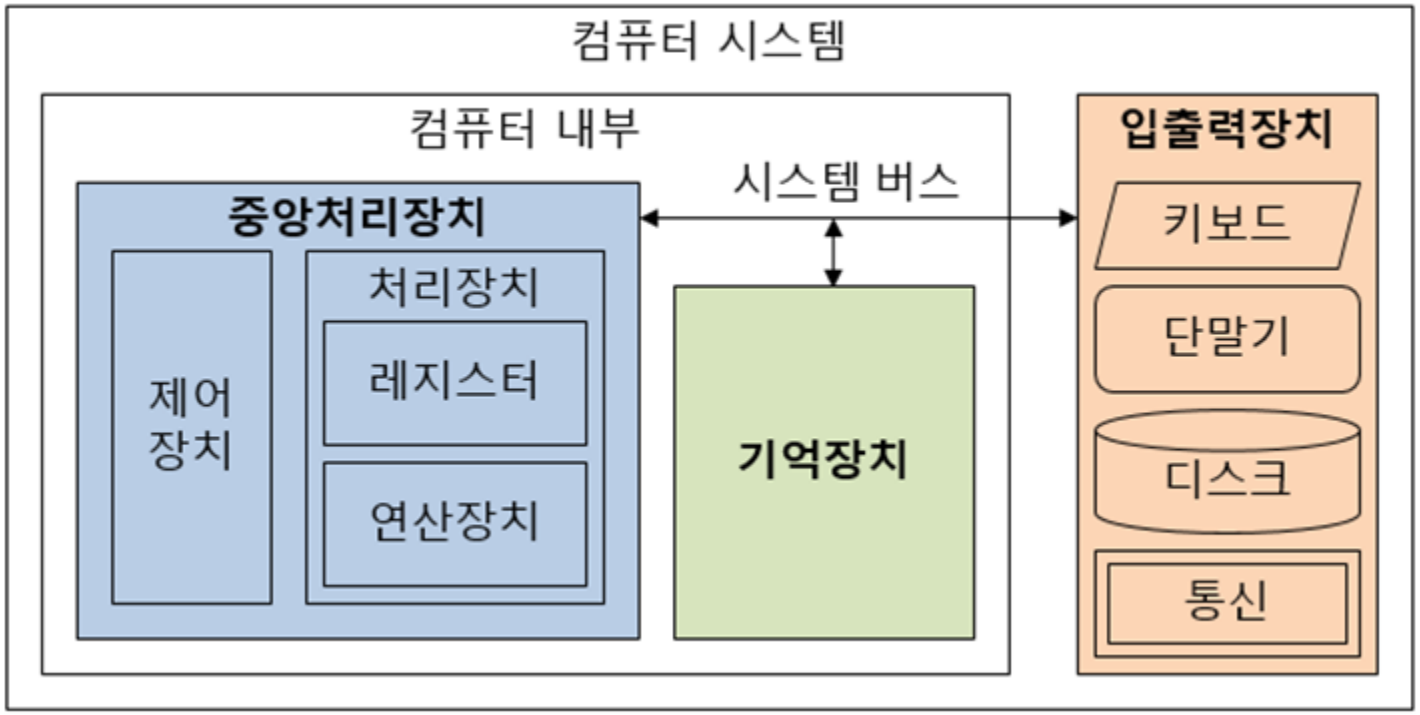
소프트웨어(software)

- 보통 프로그램을 의미
- 넓게는 프로그램과 관련된 문서도 소프트웨어에 포함시킴
- 문서의 종류
  - 사용자 문서(user documentation) : 모든 고객을 위한 인쇄된 책자
  - 시스템 문서(system documentation) : 소스코드, 설계문서
  - 기술 문서(technical documentation) : 설치, customizing, update 등

컴퓨터의 구성 : 하드웨어 & 소프트웨어

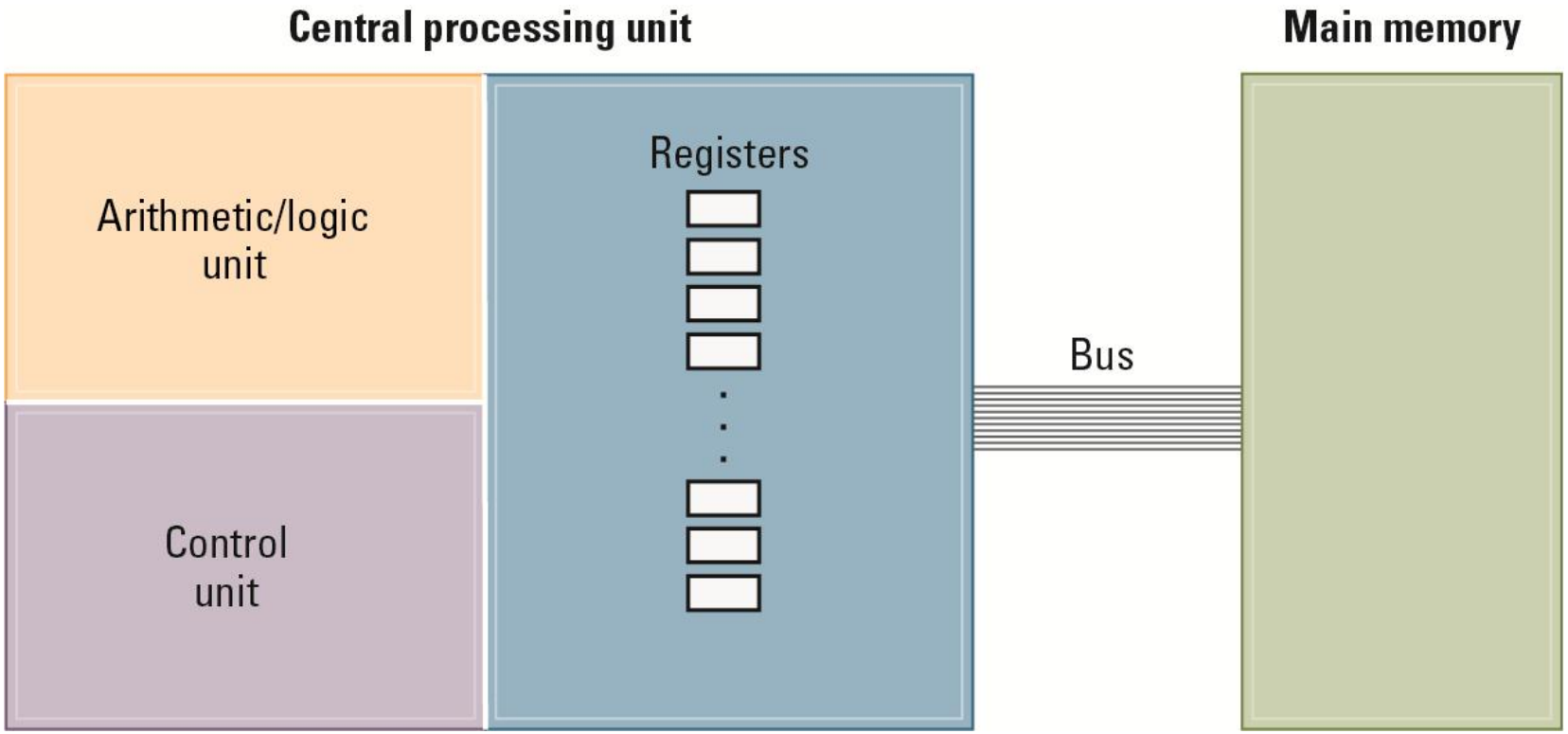
하드웨어(hardware)

- 알고리즘을 구현할 수 있는 물리적 기계 그 자체
  - 예) 중앙처리장치(CPU), 기억장치(memory), 입력장치, 출력장치



컴퓨터 구조

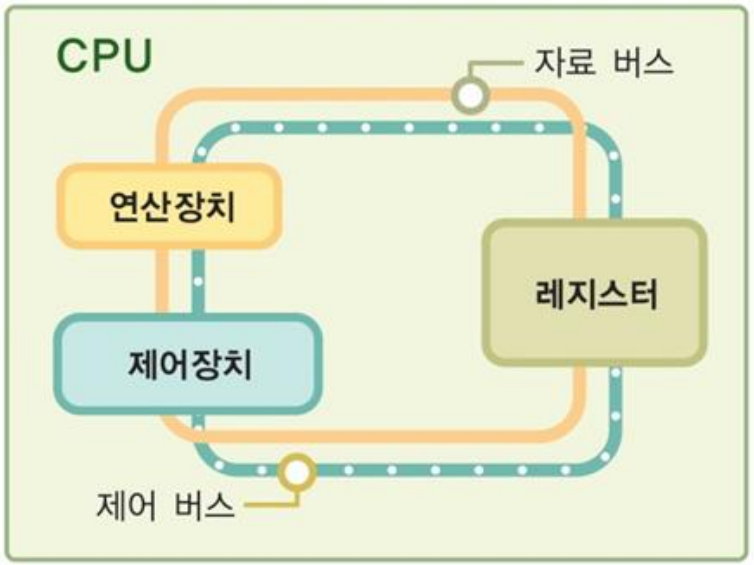
컴퓨터의 구성요소


Central processing unit
Main memory
Arithmetic/logic unit
Control unit
Registers
Bus

CPU 기초

중앙처리장치(CPU; Central Processing Unit) : 데이터의 조작

- 메모리에 저장된 프로그램과 자료를 이용하여 실제 작업을 수행하는 회로 장치
- 데이터 조작을 담당하는 컴퓨터 안의 회로
- 마이크로프로세서 : 중앙처리장치를 한 개의 칩으로 구현한 것
- 연산 장치, 제어 장치, 레지스터 등으로 구성됨



중앙처리장치의 주요 구성요소

연산장치(arithmetic/logic unit)

- 자료를 처리하고 계산하는 장치
- 산술연산 : 사칙연산
- 논리연산 : 논리합(OR), 논리곱(AND), 논리부정(NOT)

제어장치(control unit)

- 프로그램에 의해 주어지는 연산의 순서를 차례대로 실행하기 위해 기억장치, 연산장치, 입출력 장치에 제어신호 발생
- 이들 장치로부터 신호를 받아 다음에 처리할 작업들을 제어하는 역할

중앙처리장치의 주요 구성요소

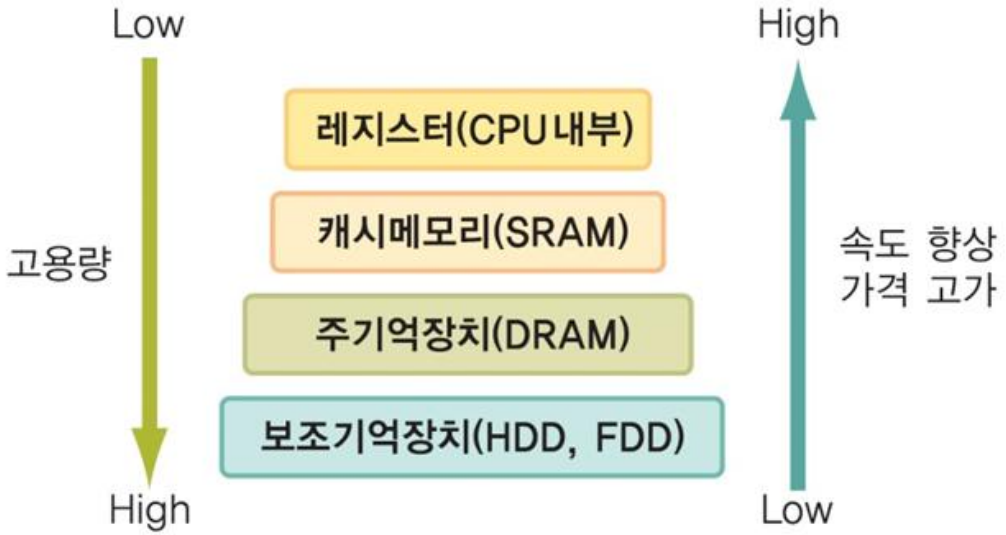
레지스터 장치(register unit)

- 레지스터(Register)
  - CPU내에서 정보를 임시로 저장하기 위해 사용
  - 여러 개의 레지스터를 가짐 (register file)
  - 성능에 매우 중요한 요소이므로 가격과 성능을 고려하여 결정
- 레지스터 종류
  - 범용 레지스터 : CPU에서 처리되는 데이터를 위한 임시 저장공간
    - 연산장치 회로의 입력들을 저장하고 있거나, 연산장치에서 계산된 결과를 저장하기 위한 공간으로 이용
  - 용도 지정 레지스터 : 임의 사용 불가능
    - 명령 레지스터, 프로그램 카운터 등

컴퓨터 내의 기억장치의 계층

기억장치 계층의 필요

- 기억장치의 속도와 용량, 가격과 그 쓰임새를 고려
- 기억장치의 속도가 빠르면 가격이 비쌈
- 동일한 비용으로 속도를 유지하려면 용량은 작아져야 함



컴퓨터 내의 기억장치의 계층

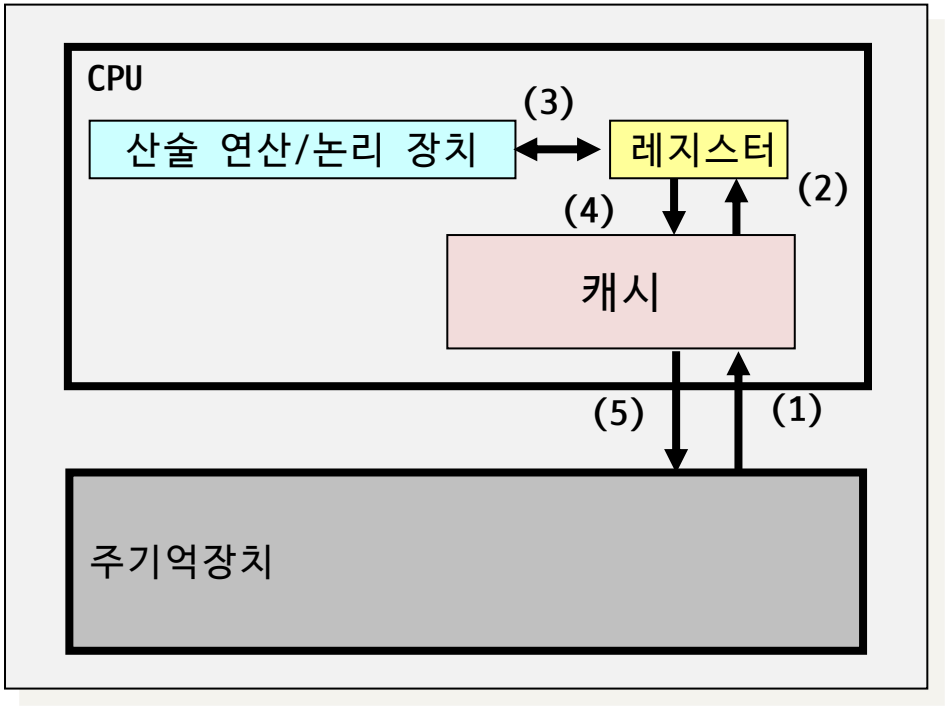
다양한 기억장치의 이용

- 레지스터 : 곧바로 사용될 데이터 보관
- 캐시 메모리
  - CPU 내부에 위치한 고속 메모리
  - 현재 자주 사용하는 주기억장치 부분의 복사본 유지
  - 캐시 메모리에 일어나는 변경들은 모아서 적절한 시점에 한꺼번에 주기억장치로 전달
  - 레지스터보다 느리지만 메인메모리보다 빠른 속도
  - 왜 쓰나? -> 주기억장치가 느리기 때문
- 주기억장치 : 가까운 시간 안에 사용될 데이터 보관
- 보조기억장치 : 당장 필요하지는 않은 데이터 보관

데이터 연산 처리 과정

주기억장치에 저장된 데이터에 대한 연산을 수행하려면..

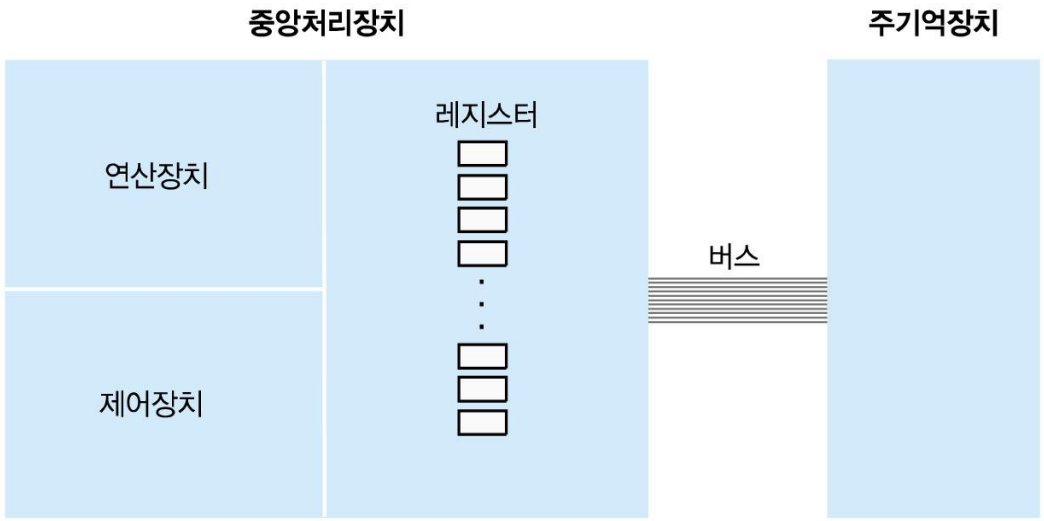
- 제어장치가 데이터를 주기억장치로부터 범용 레지스터로 전송
- 어느 레지스터가 데이터를 갖고 있는지 연산장치에 알려주고,
- 연산장치 안의 적절한 회로를 작동시키고,
- 연산장치에게 결과를 받을 레지스터가 어느 것인지 알려줌



버스

버스(bus)

- 비트 패턴을 전송하기 위해 컴퓨터의 CPU와 주기억장치는 버스(bus)라고 불리우는 전선묶음으로 연결되어 있음



- 버스를 통해 CPU는 주기억장치 회로에 해당 메모리셀의 주소와 읽기 신호를 보냄으로써 주기억장치로부터 데이터를 읽어옴
- CPU가 주기억장치에 데이터를 저장할 때에도 목적지 셀의 주소, 저장될 데이터, 적절한 전자 신호 등을 주기억장치 회로에 보냄

프로그램 내장(stored program) 개념

초기의 컴퓨터 : hard-wired programming

- CPU의 회로를 변경하도록 설계됨
- 스위치 조작으로 프로그래밍
- 데이터는 주기억장치에 저장, 프로그램 CPU의 일부로 생각

프로그램 내장 개념

- 프로그램도 비트 패턴으로 인코딩되어 주기억장치에 저장
- CPU는 주기억장치에서 명령들을 읽어와서 실행
- 또한 실행될 프로그램을 주기억장치 안에서 쉽게 변경

프로그램 내장형 컴퓨터(폰 노이만 기계)

- 컴퓨터 프로그램을 데이터와 동일하게 주기억장치 안에 저장
- 제어장치가 메모리에서 프로그램을 가져와서 명령을 해석하고 실행하도록 설계
- 제어장치 회로를 변경하는 대신, 컴퓨터의 메모리 내용을 변경하는 것만으로도 컴퓨터가 수행할 프로그램 변경가능

기계어

기계어란?

CPU는 비트패턴으로 인코딩된 명령들을 해석할 수 있도록 설계

기계어(machine language) : 인코딩체계 + 명령집합

- 기계가 인식할 수 있는 모든 기계 명령의 집합

기계 명령(machine instruction)

- 기계어에서 표현되는 기계 수준 명령
- 아주 간단한 명령들이지만, 다 모이면 복잡한 일을 수행
- 모든 프로그램은 결국 기계 명령들로 변환되어야 함

기계어 철학의 양대 축

컴퓨터 설계 시 기계 명령을 얼마나 간단히 만들 것인가?

단순하고, 빠르고, 효율적인 명령을 갖추도록 할까?

RISC

vs.

많은 수의 복잡하지만 강력한 명령을 갖추도록 할까?

CISC

RICS 구조

RISC(Reduced Instruction Set Computer) 구조

- "CPU는 최소의 기계 명령 집합을 실행하도록 설계되어야 한다!"
- 단순하고, 빠르고, 효율적인, 많이 쓰이는 소수의 명령들을 갖춤
  - RISC: 20~30개, CISC: 200 ~ 300개
- 효율적이고 빠르면서도 제작비용이 적음
- 저전력

예)

- Apple/IBM/Motorola의 PowerPC 프로세서
  - 애플의 매킨토시에 사용
- 퀄컴/텍사스인스트러먼츠의 ARM(Advanced RISC Machine) 기반 프로세서
  - 자동차모듈, 스마트폰, 네비게이션,디지털TV 등에서 많이 사용

CISC 구조

CISC(Complex Instruction Set Computer) 구조

- "CPU는 많은 수의 복잡한 명령들을 실행시킬 수 있어야 한다."
- 많은 수의 복잡하지만 강력한 명령을 갖춤
- 데스크 톱 컴퓨터 시장을 차지
  - 심지어 애플도 인텔 기반의 컴퓨터를 생산
- 프로그램하기 쉬움
  - RICS에서 여러 개 명령이 CISC에서는 하나의 명령으로 처리가능
- 점차 복잡해지는 소프트웨어에 잘 대처 가능
- 전력 많이 소모

예)

- Intel 프로세서

명령의 종류

RISC 구조이든 CISC 구조이든 기계 명령들은 모두 3그룹으로 분류

데이터 전송(Data transfer)

- 한 장소에서 다른 장소로 데이터를 복사

연산(Arithmetic/Logic)

- 기존의 비트 패턴을 사용하여 새로운 비트 패턴을 계산

제어(Control)

- 프로그램 실행을 지시

종류 1 : 데이터 전송

데이터 전송 (CPU <--> MEM)

- 데이터를 컴퓨터 내의 어느 한 장소에서 다른 장소로 옮길 것을 요청하는 명령들로 구성

실제로는 전송(transfer)가 아니라, 복사(copy)

- LOAD : MEM → CPU
- STORE/SAVE : CPU → MEM
- MOVE : MEM → MEM

I/O 명령

- CPU나 주기억장치가 아닌 프린터, 키보드, 디스플레이 화면, 디스크 장치와의 통신을 위한 명령들
- 별도로 취급

종류 2 : 연산

연산그룹

- 제어장치가 연산장치에 어떤 작업을 하도록 요청하는 명령들로 구성

종류

- AND / OR / XOR
  - 부울연산
- SHIFT / ROTATE
  - 레지스터의 내용을 그 안에서 오른쪽이나 왼쪽으로 이동하는 연산들
- ADD / SUB / MULT / DIV
  - 사칙 연산

종류 3 : 제어

제어그룹

- 데이터를 조작하는 대신 프로그램의 실행을 조종하는 명령들로 구성

JUMP / BRANCH : 대표적

- 조건부 점프(conditional jump) : C의 if, while, for 문
- 무조건 점프(unconditional jump) : C의 goto 문

덧셈 명령 예

메모리에 저장된 값들에 대한 덧셈 c = a + b

1. 덧셈에 사용될 값 중의 하나를 메모리에서 가져와 레지스터에 넣는다.

```
LOAD R1, MEM_A
```

2. 덧셈에 사용될 또 다른 값을 메모리에서 가져와 또 다른 레지스터에 넣는다.

```
LOAD R2, MEM_B
```

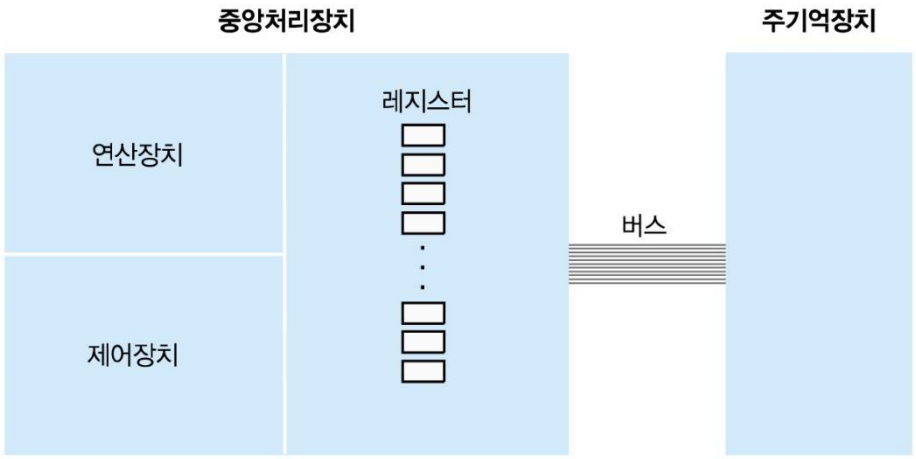
3. 단계 1, 2에서 사용된 레지스터들을 입력으로 사용하고 결과는 또 다른 레지스터에 저장하도록 덧셈 회로를 작동시킨다.

```
ADD R3 R1 R2   // R3 = R1+R2
```

4. 결과를 주기억장치에 저장한다.

```
STORE R3, MEM_C
```

5. 멈춘다.

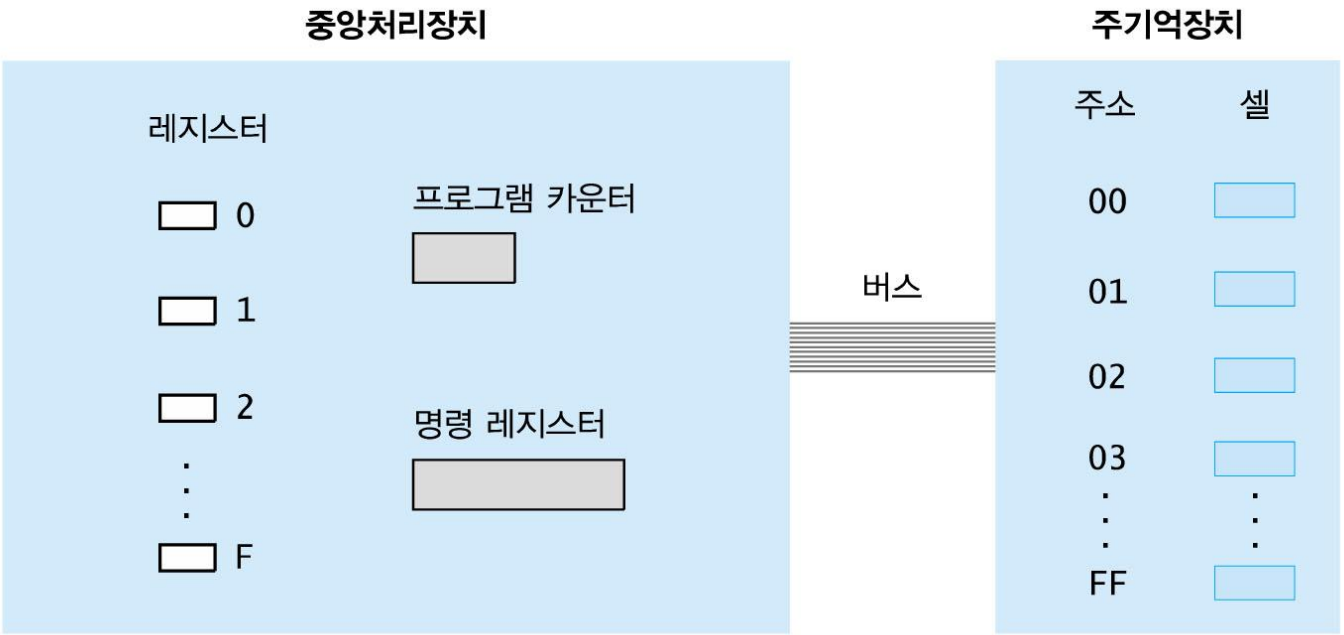


가상 기계어

컴퓨터에서 명령들은 어떻게 인코딩될까?

가상 컴퓨터를 만들어보자!

- 주기억장치 : 256개의 주기억 장치 셀(cell)
- 범용 레지스터 : 16개



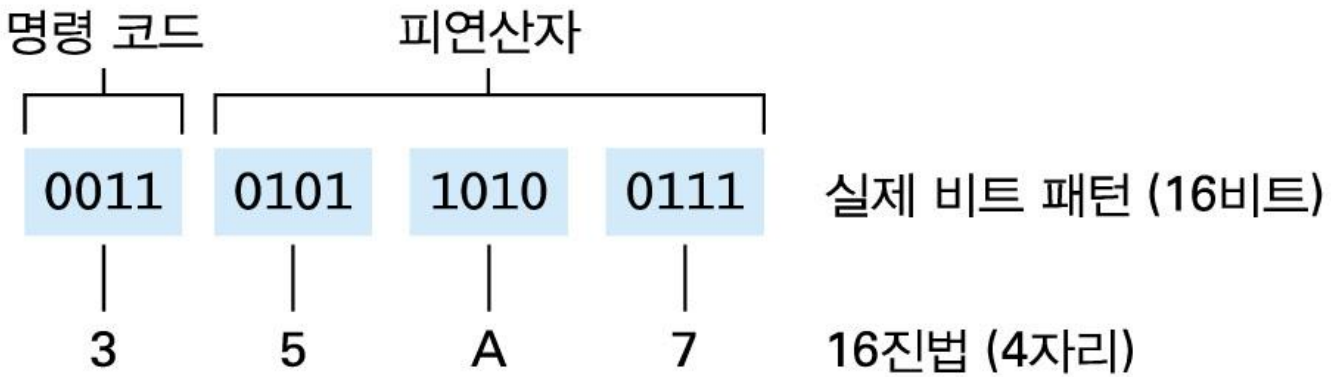
가상 기계어

기계 명령 구조

- 명령 코드(op-code, operation code) : 어느 명령인지 지정
- 피연산자(operand) : 명령어 대상 (명령코드가 사용할 정보)

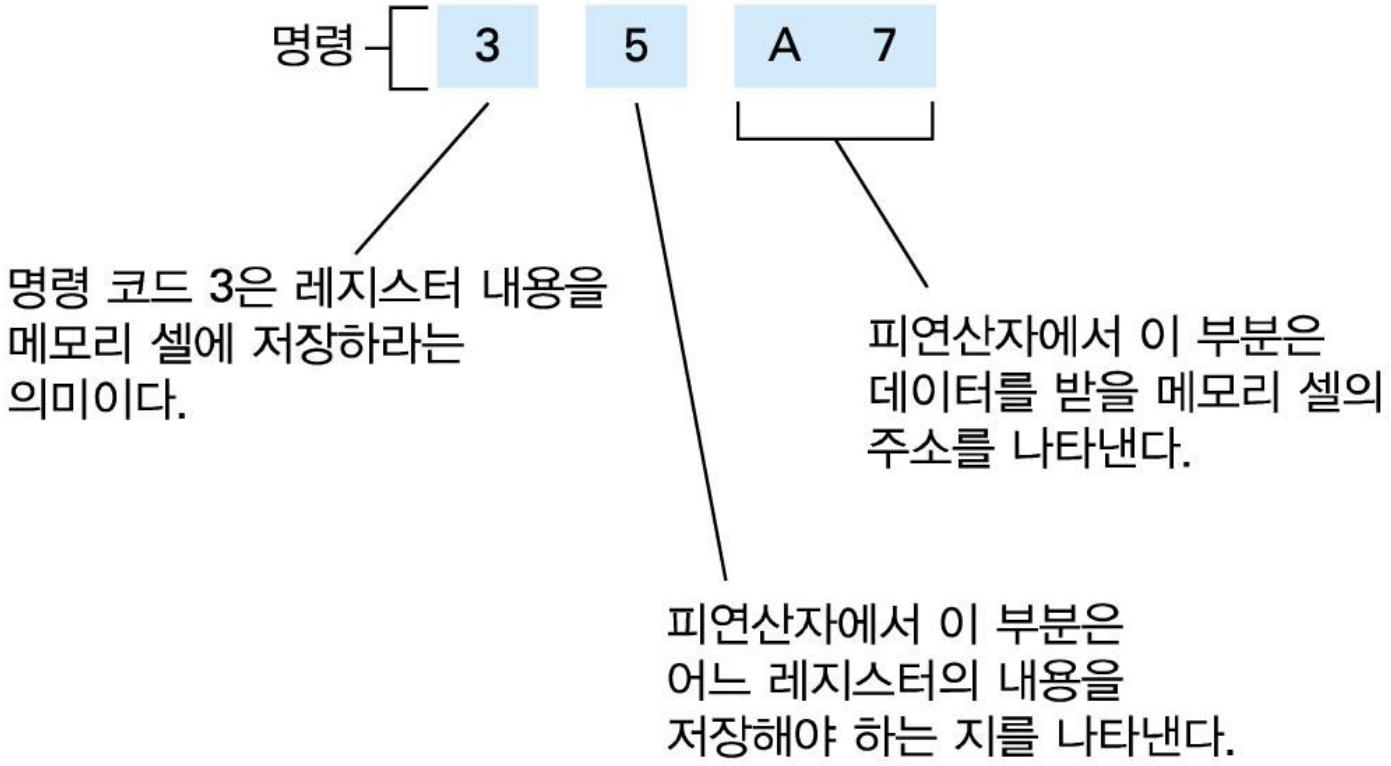
가상 컴퓨터에서 사용되는 기계어 : 2 바이트 = 16 비트

- 명령코드(op-code) : 최초의 4 비트 (16개의 명령 가능)
- 피연산자(operand) : 4 (레지스터 번호) + 8 (메모리주소/값)
  - 4 bit = 16개의 레지스터 사용가능
  - 8 bit = 256B의 메모리 사용가능



명령 35A7의 해석

"5번 레지스터의 내용을 메모리 주소 A7에 저장하라"

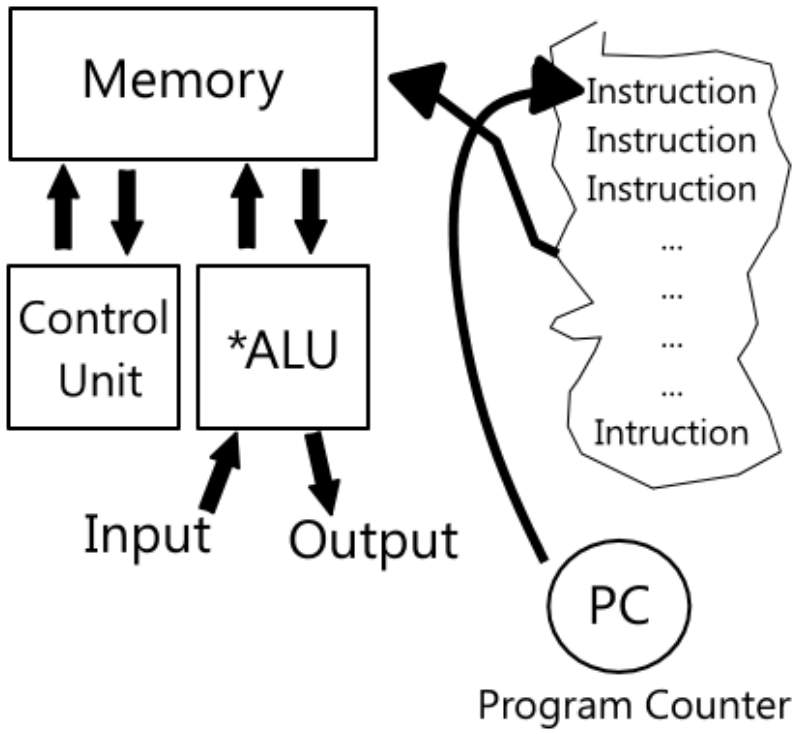


프로그램의 실행

프로그램의 실행

CPU가 프로그램을 실행시키려면?

- 프로그램들이 메모리에 위치
  - 프로그램이 실행되려면 먼저 프로그램이 실행 가능한 상태로 준비되어 있어야 함
- 컴퓨터는 필요한 대로 명령들을 메모리에서 CPU로 복사
- 일단 CPU로 옮겨진 명령은 해석되고 실행됨



**ALU(Arithmetic Logic Unit)*

용도 지정 레지스터

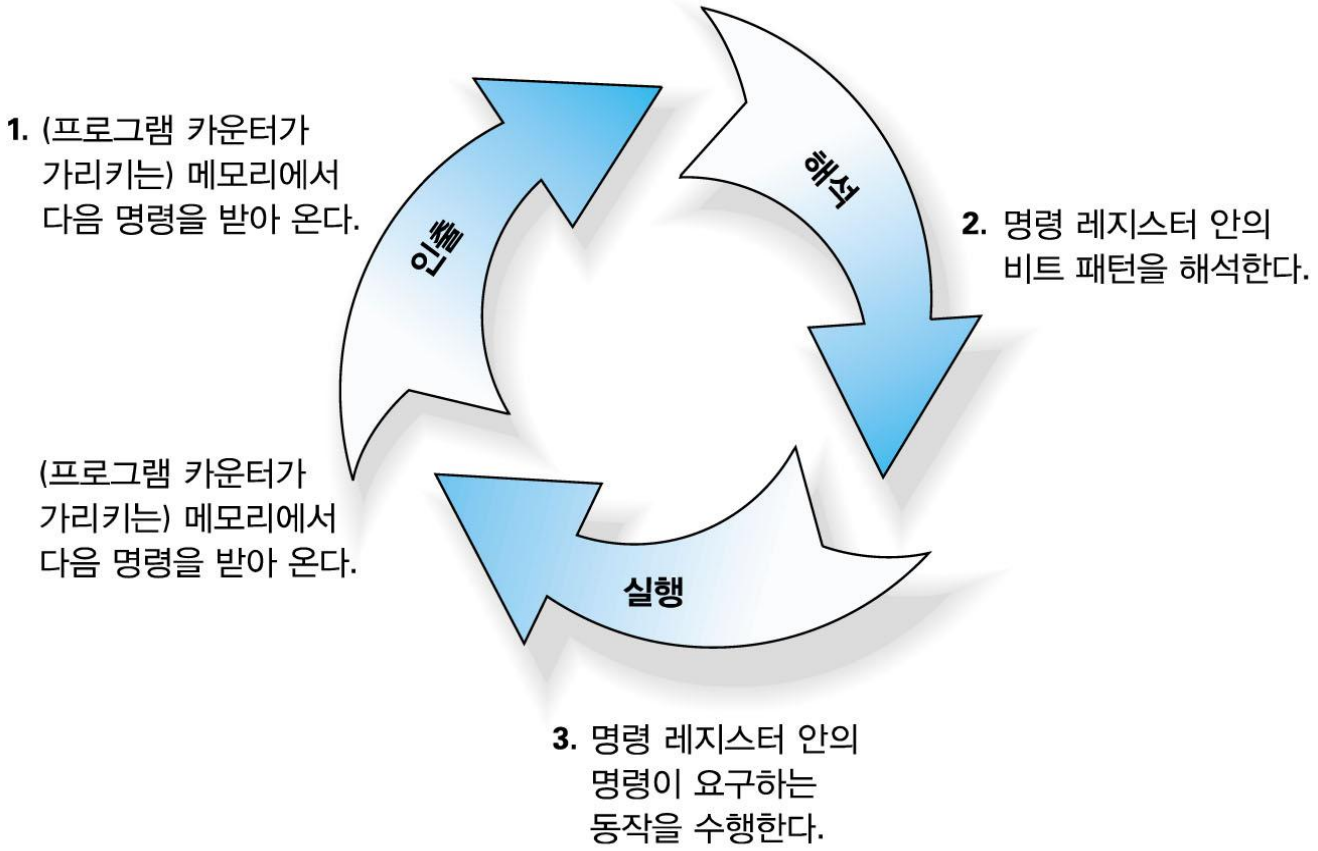
프로그램의 실행은 2개의 용도 지정 레지스터로 제어

- 프로그램 카운터(PC : Program Counter) : 8 비트
  - 다음에 실행될 명령의 주소
  - 컴퓨터가 현재 프로그램의 어느 부분에 와 있는지 추적하는 수단으로 사용
- 명령 레지스터(IR; Instruction Register) : 16 비트
  - 현재 실행 중인 명령을 보관

기계 주기

기계 주기(machine cycle)

- CPU의 작업은 기계주기라 불리는 3단계 과정을 반복함으로써 알고리즘을 실행시킴



프로그램 실행의 예 : 더하기 연산

ex) c = a + b ;

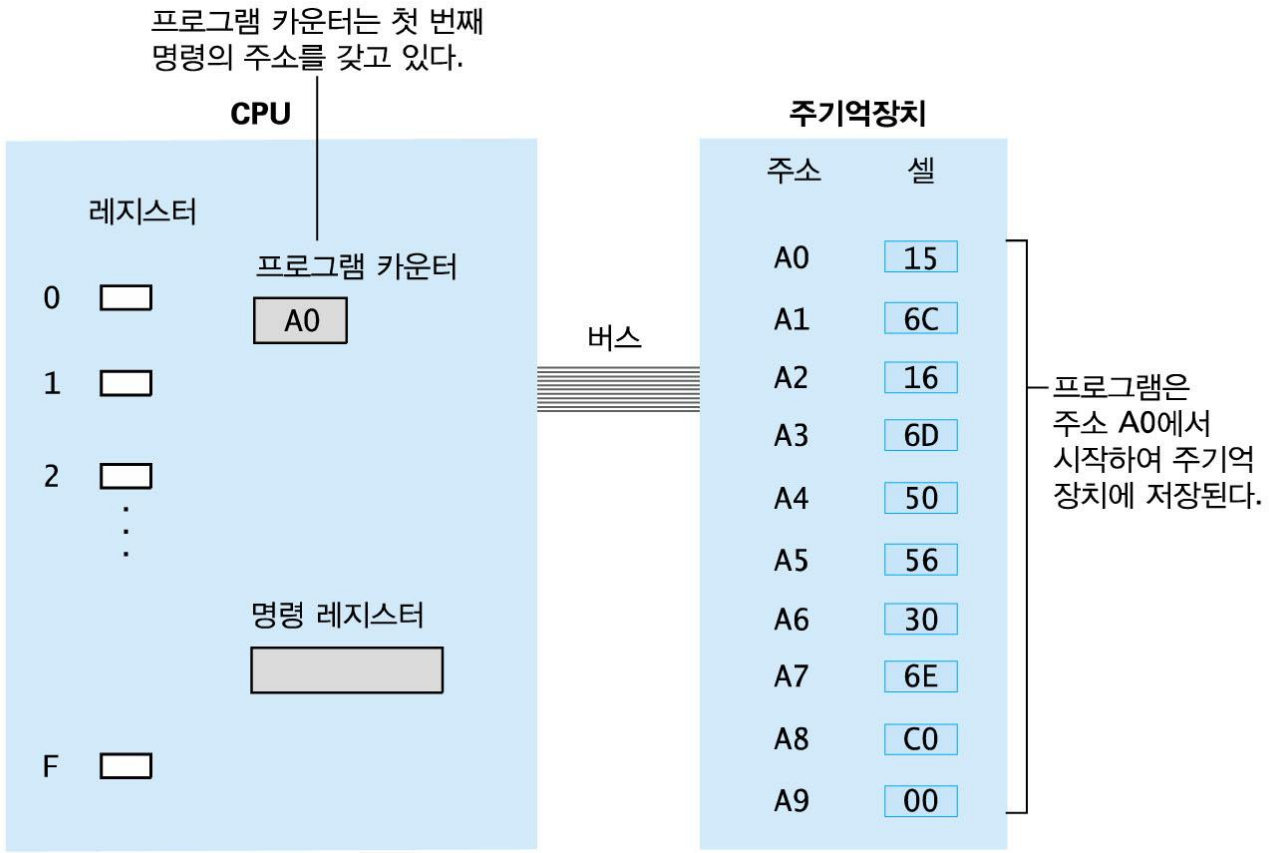
156C : LOAD

166D : LOAD

5056 : ADD

306E : STORE

C000 : HALT

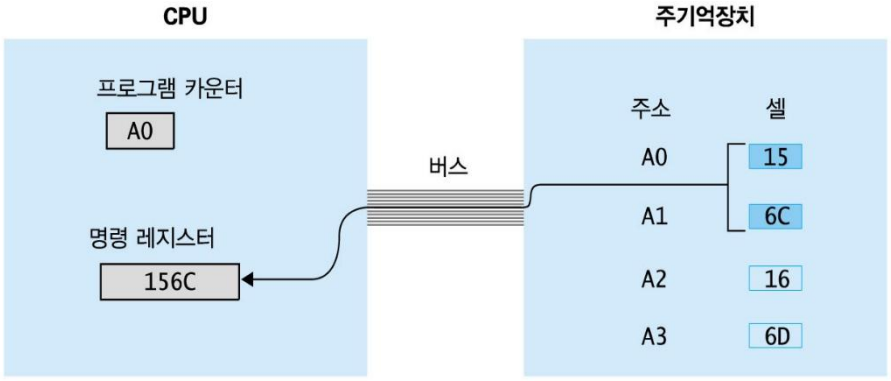


주기억장치에 저장되어 실행 준비된 프로그램

프로그램 실행의 예

156C : LOAD

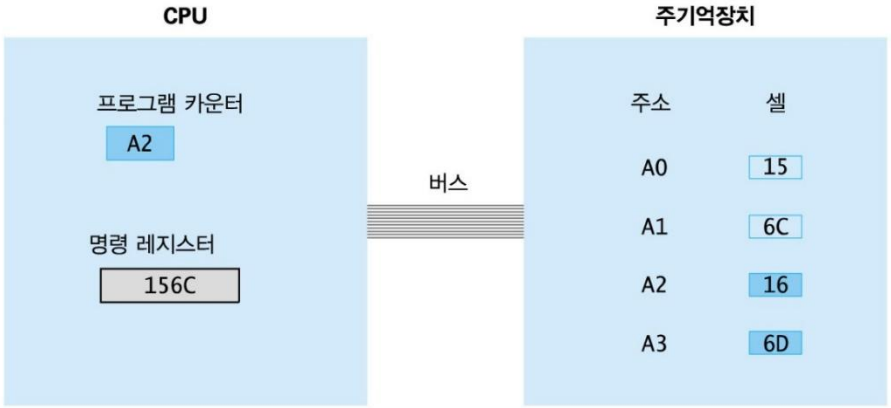
PC: A0	MEM A0: 15
	MEM A1: 6C
인출	IR ← 156C,
	PC ← A2
해석	LOAD R5, 6C로 해석
실행	R5 ← MEM 6C 내용



a. 인출 단계를 시작할 때 주소 A0에서 시작되는 명령을 메모리에서 가져와서 명령 레지스터에 넣는다.

166D : LOAD

PC: A2	MEM A2: 16
	MEM A3: 6D
인출	IR ← 166D
	PC ← A4
해석	LOAD R6, 6D로 해석
실행	R6 ← MEM 6D 내용



b. 그런 다음 프로그램 카운터가 증가되어 다음 명령을 가리킨다.

다른 장치와의 통신

다른 장치와의 통신

컴퓨터와 다른 장치 사이의 통신

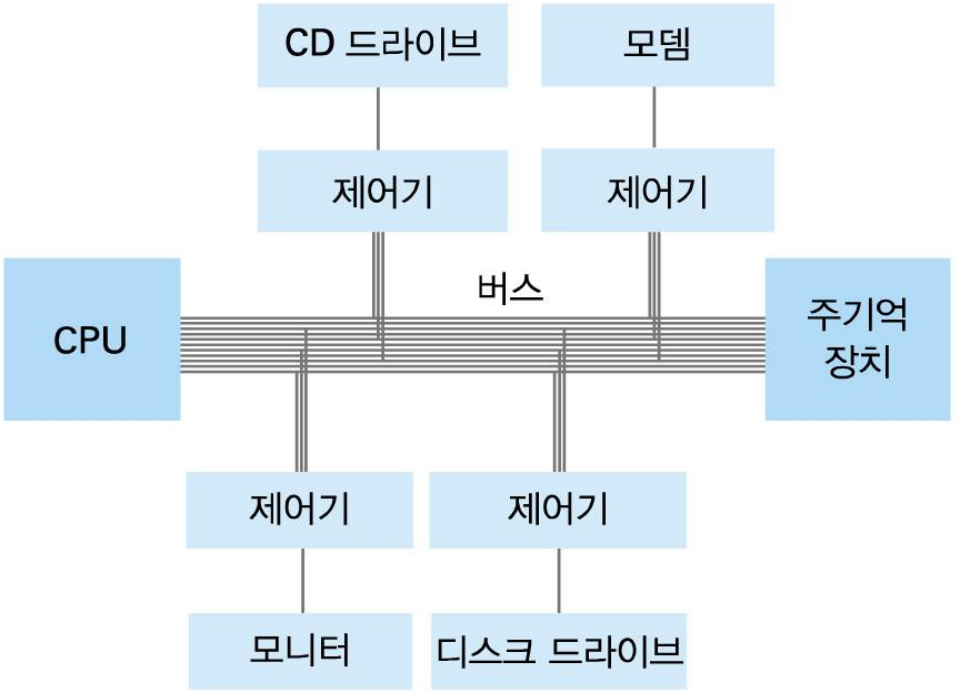
- 보통 제어기(controller)라고 불리는 중개장치를 통해 처리

주변 장치(peripheral)

- 컴퓨터의 (주로 입출력을 담당하는) 외부 장치들
- 예: disk, monitor, keyboard, printer, etc…

제어기(controller)

- CPU와 주변장치간의 통신을 담당
- 포트(port)라고 불리는 연결단자(connector)에 케이블로 연결
- 컴퓨터의 내부적 특성에 맞는 형식을 제어기가 부착된 주변장치 특성에 맞는 형식으로 또는 그 반대방향으로 메시지와 데이터를 변환



제어기

보통, special-purpose의 small computer로 구현

- 즉, 별도의 CPU (controller chip) : 제어기 동작을 지시하는 프로그램을 수행
- 별도의 memory를 가짐

각 장치유형마다 전용 제어기가 존재함

- 새 주변장치를 구입하면 그 안에 포함되어 있음

범용 제어기의 등장

- USB(범용직렬버스), Firewire와 같은 표준을 이용하여 하나의 제어기가 다양한 장치의 취급이 가능해짐

컴퓨터와 제어기의 통신 방법

컴퓨터와 제어기의 통신

- 각 제어기는 컴퓨터의 CPU와 주기억장치를 연결하는 버스에 연결을 만들어 통신함
- 보통, 버스에 직접 연결됨
  - IBM-PC: 버스 위의 slot에 삽입
- CPU는 주기억장치와 통신할 때와 동일한 방식으로 연결된 제어기들과 통신 가능