MEMO

1. 명령어(Instruction)의 구성

명령어는 크게 두 부분, 연산자(OP Code) 부분과 주소(번지, Operand) 부분으로 구성됨.

OP Code	Operand

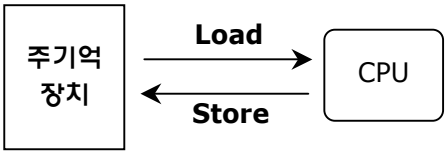
- **연산자** (OP Code)
  - 명령어에서 연산 동작을 지정하는 부분으로 명령어의 종류를 표현함.

- **주소(번지)부** (Operand)
  - 연산의 대상이 되는 데이터의 위치를 나타내는 부분

2. 연산자(OP Code)의 기능

- **함수연산 기능**
  - 산술 연산(+, −, ×, ÷) 및 논리 연산(AND, OR, NOT)을 수행하는 기능

- **자료전달 기능**
  - 주기억장치와 CPU 간의 자료 이동을 수행하는 기능



  - Load : 주기억장치에 있는 데이터를 처리하기 위해 CPU로 적재하는 작업
  - Store : 처리가 완료된 CPU 내의 정보를 주기억장치에 저장시키는 작업

- **제어 기능**
  - 프로그램의 실행순서를 변경하는 기능
  - IF : 조건에 따라 프로그램의 수행순서를 제어할 수 있는 조건부 분기문
  - GOTO : 조건과 관계없이 지정한 명령문으로 제어를 이동시키는 무조건 분기문

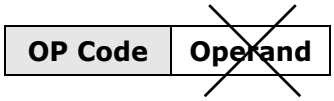
- **입 · 출력 기능**
  - INPUT, OUTPUT 등의 연산자들로 주변장치와의 입출력작업을 수행하는 기능

MEMO

3. 명령어(Instruction)의 형식

명령어는 사용되는 Operand의 개수와 따라 0-주소 명령, 1-주소 명령, 2-주소 명령, 3-주소 명령으로 구분될 수 있다.

- 0-**주소 명령**(0-Address Instruction)
  - Operand **없이** OP Code**만으로 구성되는 명령어 형식**



  - Stack**을 이용하여 연산을 수행**
  - **단항연산에 적합**
  - 0-**주소 명령으로 프로그램을 작성하면 프로그램의 길이가 길어질 수 있음.**
  - **대표적인** 0-**주소 명령어** : PUSH, POP

- 1-**주소 명령**(1-Address Instruction)
  - OP Code**와** 1**개의** Operand**로 구성되는 명령어 형식**

OP Code	Operand1

  - **누산기**(Accumulator)**를 이용하여 연산을 수행**

- 2-**주소 명령**(2-Address Instruction)
  - OP Code**와** 2**개의** Operand**로 구성되는 명령어 형식**
  - **연산결과를 위한** Operand 1**개와 입력자료를 위한** Operand 1**개로 구성**

OP Code	Operand1	Operand2

  - **가장 일반적인 연산의 형태**
  - **연산 후 입력자료의 값이 변화함**
  - Operand1**은 연산 후 결과값이 저장되는 레지스터**

- 3-**주소 명령**(3-Address Instruction)
  - OP Code**와** 3**개의** Operand**로 구성되는 명령어 형식**
  - **연산결과를 위한** Operand 1**개와 입력자료를 위한** Operand 2**개로 구성**

OP Code	Operand1	Operand2	Operand3

  - **연산 후 입력자료의 값이 보존됨.**

- Operand1은 연산 후 결과값이 저장되는 레지스터
- Operand2와 Operand3은 연산을 위한 입력자료를 위한 레지스터
- 명령어의 수행시간이 가장 길다.
- 프로그램의 길이는 가장 짧다.

- Stack을 사용하는 명령어 형식? 0-주소 명령
- 누산기를 사용하는 명령어 형식? 1-주소 명령
- 자료의 주소지정이 필요없는 명령어 형식? 0-주소 명령
- 연산후 입력자료가 소멸되는 명령어 형식? 2-주소 명령
- 연산후 입력자료가 보존되는 명령어 형식? 3-주소 명령

MEMO