#cs #study

0) 대전제

- 열거된 내용을 이해하기 위해 모든 수단을 쓰는 건 좋다
- 단, Quiz는 AI를 이용하여 바로 답을 찾지 마라

1) 목차

- 컴퓨터 구조 전반
- 컴퓨터가 이해하는 정보

2) 컴퓨터 구조 전반

정보

- 컴퓨터는 프로그래밍 언어를 이해할 수 있는가?

```
정답은 No!

컴퓨터가 알아듣는 언어로 별도의 번역 절차를 걸치지 않으면 개발자가 쓰는 프로그래밍 언어는 이해 못한다
```

- 컴퓨터가 이해하는 정보
  - 데이터 : 가공되지 않은 정보. 어떠한 뜻 없이 0과 1로 나열된 숫자의 조합.
  - 명령어 : 데이터 가공 규칙.

핵심 부품

- 컴퓨터의 핵심 부품은 CPU, 메모리, 캐시 메모리, 보조기억장치, 입출력장치이다

```
정답은 Yes!

컴퓨터는 계산기이며 계산식을 풀 두뇌(CPU), 처리 과정을 기억할 장소(RAM), 계산기에 식을 입력하고 계산 결과를 보여주기 위한 장치(입출력장치), 계산기의 숫자 위치 등 구조를 반영구적으로 저장하기 위한 장치(보조기억장치)가 있어야 계산기가 돌아가겠죠?
```

- CPU
  - 정보를 읽고 해석하고 실행하는 부품
  - 구조 : ALU(산술논리연산장치), CU(제어장치), 레지스터
  - ALU : 사칙 연산, 논리 연산 등을 수행할 회로

    - CU : 명령어를 해석하고 그에 맞는 제어 신호를 전송
    - 레지스터 : 계산하는 중간 과정 값 등을 저장하기 위한 임시 저장 장치
- 메모리와 캐시 메모리
    - 메모리 : 실행 중인 프로그램을 구성하는 데이터와 명령어를 저장
        - 특징 : 휘발성, 주소 개념을 통한 빠른 데이터 접근
        - 주소 : 메모리에 저장하는 값들의 위치를 나타낸다
    - 캐시 메모리 : CPU가 메모리에 저장한 값에 빨리 접근하기 위해 사용
- 보조기억장치 : 장치 전원 여부와 상관없이 자료를 보관하기 위해 사용
    - 특징 : 비휘발성, RAID
- 입출력장치 : 컴퓨터 외부에 연결되어 컴퓨터 내부와 정보를 교환하는 장치
- 메인보드 : 위에 열거한 부품들을 포함하여 컴퓨터에 필요한 부품을 설치해주는 기판
    - 버스 : 메인보드에 연결한 컴퓨터 부품들이 정보를 교환하는 통로

Quiz

- 문자 'C'는 ascii 코드로 썼을 때, 데이터인가 명령어인가?
- 메모리와 보조기억장치의 차이를 설명하시오.
- 컴퓨터에 "1 + 2의 답을 구하라"고 입력했다. CPU와 메모리에서 어떤 과정을 거쳐 계산을 수행하고 답을 낼 지 생각하고 순서를 적어보시오.

3) 컴퓨터가 이해하는 정보

데이터

- 비트(bit) : 컴퓨터가 이해할 수 있는 가장 작은 정보 단위. 0과 1로 이루어져 있는 이진수.

```
1 bit에 몇 가지 정보를 담을 수 있는가?

정답은 2개다. 왜냐하면 0(No), 1(Yes)로 생각해도 No, Yes 두 개의 정보를 담을 수 있기 때문이다.
```

데이터 : 숫자

- 컴퓨터는 무리수를 정확히 표현할 수 있을까?

```
정답은 No!

컴퓨터는 엄연히 한계가 있는 물리적인 장치입니다. 당연하게도 무한히 수를 표현하면 컴퓨터가 망가지겠죠?
```

- 부동소수점 : 최대한 많은 숫자 범위를 표현하기 위한 소수점 표기 방식. 부호, 지수, 가수로 분리하여 저장한다. 저장 시에는 과학적 표기법에 맞추고 저장한다.

    - 부호 : 0, 1로 구분하여 숫자가 음수인지 양수인지 판별
    - 지수 : 제곱수. 2^n 에서 n에 해당한다.
        - 바이어스 : 지수를 저장하기 위해 사전에 할당된 비트 수. 2^(비트수 - 1) - 1 값을 지수에 더한다.
    - 가수 : 숫자의 나열.
    - 예시 : 1110.10 * 2^-15 -> 지수는 111010, 가수는 -15

Quiz

- 8 비트는 얼마 만큼의 정보를 표현할 수 있는가?
- 102.4를 32비트(부호 1 비트, 지수 8 비트, 가수 23 비트) 부동소수점으로 표현하시오.

데이터 : 문자

- 컴퓨터는 'C'를 쓰면 별도의 번역 없이 'C'로 알아듣는가?

```
정답은 No!

컴퓨터가 이해할 수 있게 이진수로 변환하는 '인코딩' 과정을 거쳐야 합니다. 반대로 컴퓨터 이진수를 사람이 이해할 수 있는 문자로 변환하는 과정을 '디코딩'이라고 합니다.
```

- 문자 집합 : 컴퓨터에게 적용하기 위한 문자 규칙
    - 아스키 코드 : 초창기 컴퓨터에서 사용한 문자집합. 8비트 내, 0 ~ 127의 수를 각 고유 문자에 일대일 대응. 128은 패리비 비트로 오류 검증용.
        - 단점 : 다양한 문자를 표현하지 못함. 알파벳 말고 인간의 언어 표현 수단 없음.
    - 유니코드 : 언어, 특수문자, 이모티콘 등. 다양한 문자를 표현하기 위해 탄생한 문자 집합
        - 가변 길이 인코딩 : utf-8, utf-16, utf-32처럼 각 비트 수에 따라 별도의 인코딩 방식 적용. 즉, 각 문자가 8 bit, 16 bit, 32 bit 이진수로 변환된다.
        - 통일된 규격이기에 각 자연어마다 다른 체계를 적용할 필요가 없음
    - base64 : 64진법으로 문자를 표현하는 문자 집합.
        - 문자 외 이진 데이터 인코딩 시 사용. 64진법이어서 대용량 데이터를 저장하는 데 용이함.
        - 패딩 : 문자열이 딱 64진법으로 나누어 떨어지지 않는 경우, 남는 자리를 0으로 채운다.

Quiz

- 'abc'를 base64로 인코딩한 결과는?
    - 어째서 그 결과가 나오는가?
    - 'ab'를 인코딩하면 어떤 결과가 나오는가?
- 본인에게 능숙한 프로그래밍 언어로 "한국"을 utf-8, utf-16, utf-32로 변환하여 값을 확인해봅시다. 어째서 차이가 있을까요?

명령어

- 100과 120을 더해라. 이 문장에서 컴퓨터가 명령을 수행할 대상과 동작은 무엇일까요?

```
대상은 100, 120, 동작은 덧셈입니다.

이 때, 컴퓨터가 수행할 동작은 "연산자(opcode)"라 하고 동작에 사용할 데이터를 저장한 장소를 "피연산자(operand)"라 합니다.
```

- 피연산자를 보면 저장 방식을 알 수 있다
  - 대상이 그대로 저장 : 아, 메모리에서 바로 값을 가져올 수 있구나
  - 대상의 주소값을 저장 : 아, 메모리에서 별도 주소로 접근이 필요하구나
- 연산자 종류
  - 데이터 전송 : MOVE, STORE, LOAD(FETCH), PUSH, POP
  - 산술/논리 연산
    - 사칙연산 : ADD/SUBSTRACT/MULTIPLY/DIVIDE
    - 증감 : INCREMENT/DECREMENT
    - 논리 : AND/OR/NOT
    - 비교 : COMPARE
  - 제어 흐름 변경 : JUMP, CONDITIONAL JUMP, HALT, CALL, RETURN
  - 입출력 제어 : READ, WRITE, START IO, TEST IO
- 기계어와 어셈블리어
  - 기계어 : 이진수로 표현한 언어. 말 그대로 컴퓨터(기계)가 이해할 수 있는 언어
  - 어셈블리어 : 사람이 이해할 수 있게 아주 간단하게 기계어를 번역한 언어

Quiz

- 100과 120을 더해라. 이 문장을 CPU가 수행했을 때, 피연산자 120에 저장된 값이 0x78이다. 이는 120이 저장된 것인가 아니면 주소값인가?

명령어 사이클

- 명령어를 실행하는 과정은 여러 순환 주기를 가지고 있다
- 인출 사이클(fetch cycle) : 메모리에 있는 명령어를 CPU로 가져온다
- 실행 사이클(execution cycle) : CPU로 가져온 명령어를 실행
- 간접 사이클(indirect cycle) : 메모리에 저장된 값이 주소인 경우, 추가적인 메모리 접근이 필요할 때
- 인터럽트 사이클(interrupt cycle) : 간섭 신호가 들어온 경우에 그쪽 신호로 별도 접근이 필요한 경우

다음주에 계속