폰 노이만 머신

* CPU

ALU(산술논리장치), register(명령 레지스터, 데이터 레지스터, 프로그램 레지스터 등 -> 기억장치)

* RAM(주기억장치)

전원 공급이 멈추면 정보 사라짐

Process와 data로 구분되어 있음

Process 영역에는 instruction이, data 영역에는 변수 값이 저장되는 방식임

* SSD, HDD(보조기억장치)

연산자(Operater)

피연산자(Operand)

-> ADD 1 2 일 때, ADD는 operator// 1, 2는 operand //전체 단위명령을 기계어로 전환한 값은 instruction이라 함(01001110 00000100 00000000)

변수(variable)

* 메모리가 필요하고, 변하고, 주소로 바뀜

상수(constant)

* Instruction에 operand로 존재하며, 변하지 않음

bit 1개 증가 -> 데이터 표현의 개수(= 저장 용량) 2배

* 5bit = 25
* 6bit = 26

16 진수(0,1,2,3,4…9,A,B,C,D,E,F) 계산법

* FF = F \* 161 + F \* 160  = 15 \* 16 + 15 = 255

bit 단위

* 1 nibble = 4 bit
* 1 byte = 8 bit

byte

1. ASC(아스키)에 의해 영어 한 문자 표현 가능
2. byte machine: 메모리를 1byte 단위로 주소를 정해서 관리하는 시스템

아스키 코드(American Standard Code)

* 미국이 영어를 이진법으로 표현하기 위해 정해놓은 코드
* 영어 각 문자에 1 byte(8글자) 배열이 할당되어 있음
* A: 65, a: 97, 0: 48 🡪 외우기
* Unicode - 16bit//C언어 코딩 시 Unicode나 ASC 중 결정하고 코딩해야함

bit, byte

1KB = 1024bit = 210byte

1MB = 220byte

1GB = 230byte

n의 보수: 합해서 n이 되는 수

데이터 값이 정수 표현 범위를 벗어난 경우

1. bit를 증가 -> 2배
2. 부호를 쓰지 않고 처리 시도

부동소수점(floating-point)

* 정규화(normalization) 과정을 통해 부호, 가수, 지수로 분류되어 저장
* 부호: (-1)n, n = 1 or 0으로 표현
* 부호 1bit, 지수 8bit, 가수 23bit 할당
* 컴퓨터가 0.1 을 정확히 표현할 수 없는 이유
* 컴퓨터는 정수형 데이터를 처리할 때 속도가 더 빠름 -> 정수 지향

화소(pixel)

* R, G, B(빛의 삼원색)을 사용
* 한 pixel에는 R, G, B의 정보가 각 0~255의 값으로 들어감
* 예시: (255, 255, 255) = 흰색, (0, 0, 0) = 검은색

Sound

* 파형의 좌표 값으로 저장
* 좌표 값을 더 촘촘하게 저장할수록 소리는 정확하고, 용량은 커짐
* 헤르츠(Hz): 촘촘한 정도

C 프로그래밍

* main 함수는 반드시 1개 존재해야 함(void main, int main, main)
* int main()일 때, main 앞쪽(이 경우 int)은 출력값 자료형, 뒤쪽(void)는 입력값
* return 0; -> 운영체제(윈도우 혹은 linux)에 제공하는 결과값
* printf: 지정 값을 출력하는 함수. f는 format(형식)의 약자로, 형식을 지정하여 출력한다는 의미
* #include <stdio.h>: stdio.h을 include해야 printf 등의 함수를 사용할 수 있음

#include: 전처리 지시자

stdio.h: 헤더파일

* 프로그래밍에서의 전처리: 이진수 형태로 변환하는 것

자료형

* 정수형: char(1), short(2), int(4 ~2), long(4~8), long long(8)

“unsigned”를 정수형 앞에 붙이면 양수만 다룸. 예: -128 ~ 127 🡪 1 ~ 255

* 부동 소수형: float, double
* Void: 3가지 용도가 있음

1. 함수의 입력 값이 없을 때
2. 함수의 결과 값이 없을 때
3. void \*

상수(literal)

* 100 -> int
* ‘A’(65) -> int
* 10,000,000,000 -> long long
* 3.14 -> double
* 2.718F -> float
* “ABCD” -> const char \*

연산자

* 산술: +, -, \*, /, %
* 증감: ++, --(1증가, 1감소 -> 사용 예시: ++a;, a++;)
* [식은 실행시간에 평가되면 하나의 결과값을 가진다]

예): ++a -> 증가한 사실이 먼저. 증가한 뒤의 값 // a++ 원래 값이 먼저. 증가하기 전 값

* 치환: a = 100; // b = a + 200;