

<자료 정리 (1/2)>

1. C언어를 배우는 이유는?

C언어를 어셈블리어로 컴파일하고, 어셈블리어를 기계어로 어셈블리해서 사용자와 기계간 소통이 가능하게 된다. 결국 인간이 기계를 제어하고자 사용하는 언어다.

자세히 뜯어 보면 컴퓨터 중앙처리장치(CPU)가 사용하는 2진수인 기계어가 있다. 기계어는 CPU마다 표현방식이 다르기도 하고, 좀더 인간이 이해하기 쉽게 만든 것이 어셈블리어다. (어셈블리어는 프로그래머 전용언어이기 때문에 추후에 자료구조 공부에서 배워보기로 하고 넘어가자.) 이 어셈블리어를 나같은 일반인들을 위해 작성된 것이 고급언어인 C,C++,자바 등등.. 이다.

즉, 기계어(1101011) -> 어셈블리어(ADD,MOV,POP..) -> C, C++,JAVA..

2. Embedded란?

임베디드는 전자회로를 내장한 초소형 컴퓨터 시스템라 할 수 있다. (나는) 앞으로 임베디드 시스템 제어의 기초가 되는 Microcontroller를 위주로 공부하게 된다.

(위키백과 참고)

임베디드 시스템([영어](#): embedded system, 내장형 시스템)은 기계나 기타 제어가 필요한 시스템에 대해, 제어를 위한 특정 기능을 수행하는 [컴퓨터 시스템](#)으로 장치 내에 존재하는 전자 시스템이다.

3. Microprocessor와 Microcontroller 차이는?

-Microprocessor은 범용목적을 가진 칩으로 off chip device이다. (외부에 RAM, ROM, I/O 장착됨)

특징: 비쌈, 폰노이만 구조, 느린 속도, 고전력

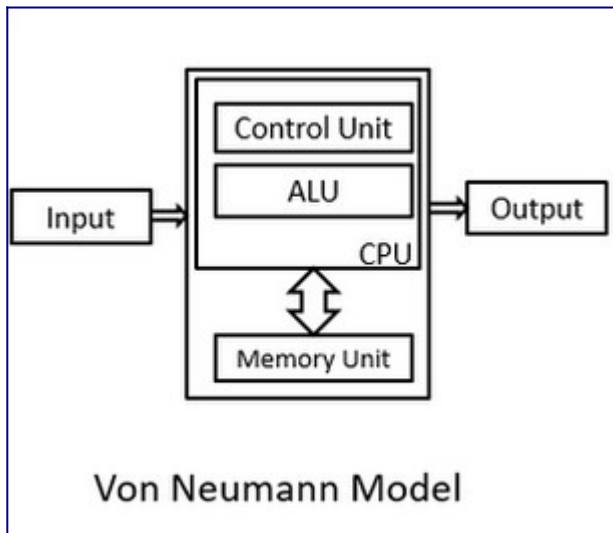
사용예: Core i3,i7 컴퓨터, 모바일, 비디오게임, TV..

-Microcontroller는 특정목적을 가진 칩으로 on chip device이다. (내부에 RAM,ROM,IO,CPU가 장착됨)

특징: 저렴, 하버드 구조, 빠른 속도, 저전력 등의 특징을 가지고 있다.

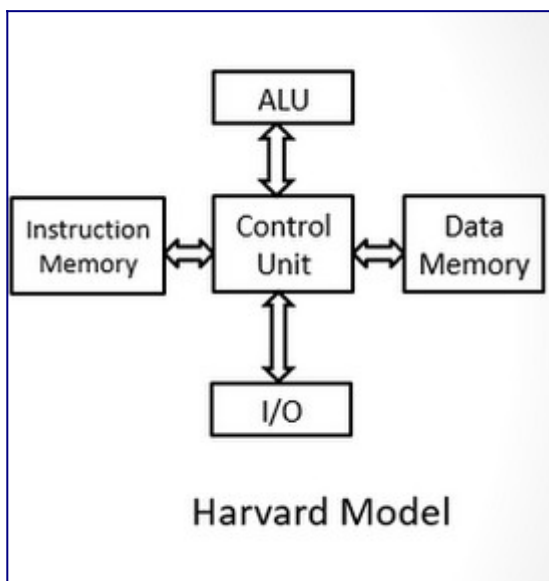
사용예: 세탁기, 전자레인지, 청소기..

4. 폰 노이만 구조



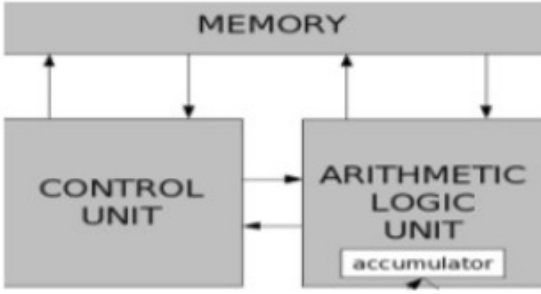
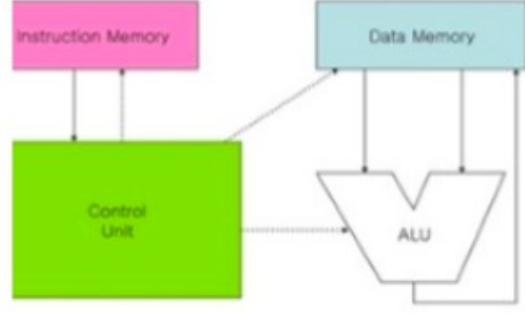
CPU, RAM, IO구조의 프로그램 내장 방식 범용 컴퓨터 구조이다. 하나의 버스로 명령어 메모리와 데이터 메모리를 함께 저장하고 처리하는 방식으로, 병목현상이 발생할 수 있다. 이를 해결하고자 하버드 구조를 사용하게 됨.

5. 하버드 구조



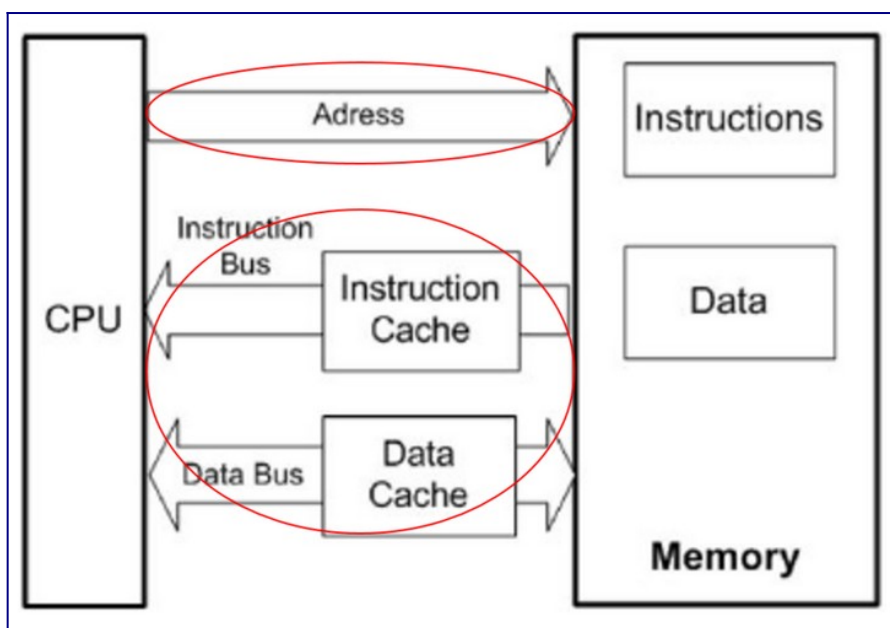
폰노이만에서 단점 보완. 명령어와 데이터 메모리를 분리하여 병렬적으로 처리하는 구조. 처리속도 개선
메모리로 부터 동시에 읽기 가능

6. 비교

폰 노이만(Von Neumann) 아키텍처	구분	하버드(Havard) 아키텍처
		
명령어, 데이터 메모리 구분 없음 명령어 읽을 때 데이터 접근불가	구조	명령어, 데이터 메모리 인터페이스 분리 명령어, 데이터 병행처리 가능
공용 메모리로 구현비용 저렴 단순한 하드웨어 구조, 구현이 간단함	장점	파이프 라이닝으로 병렬처리 구현 고성능 처리 가능
메모리 접근의 병목 현상 발생 파이프라인 해저드(구조적 해저드)	단점	두 개의 버스/메모리로 공간 많이 차지 복잡 하드웨어 구조, 높은 비용
일반적인 컴퓨터에서 대다수 사용	활용	고성능 요구하는 특수 분야(DSP, Embedded)

7. 폰노이만과 하버드 아키텍처의 결합

최신 컴퓨터는 둘 다 결합된 구조다. CPU의 캐시 메모리를 명령어캐시와 데이터캐시로 분리함. CPU
내부는 하버드, CPU 외부는 폰노이만으로 이루어짐



<자료 정리 (2/2)>

1. 변수란 무엇인가?

변수는 데이터의 저장과 참조를 위해 할당된 메모리 공간이다. 그러니까 메모리를 사용하기 위해서 반드시 필요한 것이다. 예를들어, a라는 변수에 2(값) 정수를 저장하면, 컴퓨터가 a라는 변수의 주소값에 메모리에 저장한다. 그리고 내가 만약 값을 변경하면, a의 변수의 주소값에 참조해서 들어가 2(값,데이터)을 변경한다.

메모리공간을 가지고 생각하는 것들이 아직은 생소하고 어렵지만, 익숙해질때까지 지속적으로 접근해보자..

2. 변수 Data Type

C의 각 변수에는 특정 유형이 있다.

데이터 타입명	메모리 크기	데이터 유효값	설명
char	1 Byte	-128 ~ 127	1byte 정수형 데이터 타입
int	4 Bytes	-2,147,483,648 ~ 2,147,483,647	4byte 정수형 데이터 타입
unsigned int	4 Bytes	0 ~ 4,294,967,295	4byte 양수 정수형 데이터 타입
short int	2 Bytes	-32,768 ~ 32,767	2byte 정수형 데이터 타입
unsigned short int	2 Bytes	0 ~ 65,536	2Byte 양수 정수형 데이터 타입
float	4 Bytes	1.17e-38 ~ 3.4e+38	4byte 실수형 데이터 타입
double	8 Bytes	2.22e-308 ~ 1.79e+308	8byte 실수형 데이터 타입
bool	1 Bytes	true, false	1byte 논리형 데이터 타입

변수 선언시 유의점은

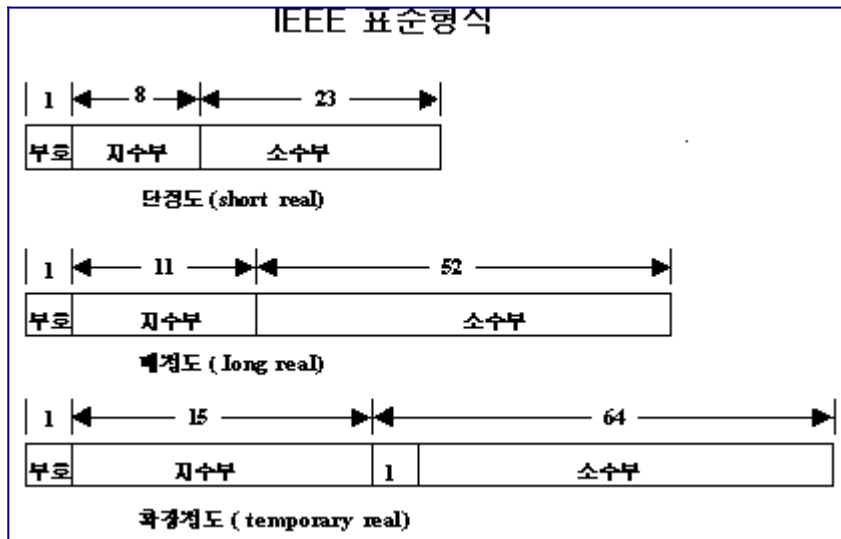
- 2-1. 변수는 숫자로 시작할 수 없다.
- 2-2. 변수명은 알파벳 대소문자, 숫자, 언더바(_) 로만 선언이 가능하다.
- 2-3. 띄어쓰기 안됨, 총 63자까지 허용, 키워드 사용 불가
- 2-4. 대소문자를 구분한다.

3. Single/Double Precision

3-1. Single precision: 단정밀도, float (32bit)

3-2. Double Precision: 배정밀도, double (64bit)

부동소수점 방식은 IEEE 754 표준을 따르며, 고정소수점에 비해 매우 크거나 작은 값을 표현 할 수 있지만 연산속도가 느리다.



부호(sign)

양수와 음수를 판단하는 방법이며 가장 왼쪽 비트(MSB)이다.

0: 양수, 1: 음수

지수(exponent)

부호부 다음 8비트로 표현된다. 예를들어 아래와 같이 2^e 로 계산을 한다.

가수(mantissa or fraction)

가수는 실질적인 데이터를 의미한다.

그렇다면 이런 부동소수점은 컴퓨터에서 어떻게 저장하는지 실제로 계산해보자!!

이진수 변환, 정규화, 지수부 바이어스 표현법, 정형화 과정을 거쳐야 하므로 복잡하지만 차근히...

예를들어, -123.45 부동소수점으로 표현하면?

먼저 2진수로 변환 (이진수 변환)한다.

소수를 2진수로 변환하는것은 정수부와 소수부를 따로 생각하면 된다. 아래와 같이 정수부는 2로 계속 나누어주고 소수부는 2를 계속 곱해주면 된다.

2를 계속 곱해 소수점 밑이 0이 되도록 하고 이를 역순으로 작성한다.

$$0.25 * 2 = 0.5$$

$$0.5 * 2 = 1.0$$

여기서 이진수 표현으로 하면 $-123.25 = 1111011.01$ 이 된다.

이제, 정규화 (정규화: $1.xxxx * 2^E$ 형태로 바꾼값)을 직접 표현한다.

$1.xxxx * 2^E$ 승 형태로 만들면 $1.11101101 * 2^6$ 이 된다.

여기서 frac에는 11101101이다.

마지막으로, 지수부 바이어스(e는 exp필드의 비트 패턴으로 계산하고 바이어스는 $2^{(x-1)}-1$)로 표현한다. float에서 지수는 -126에서 127의 범위를 가진다.

여기서 E는 6이므로 exp에는 133 이다. $133 = 10000101(2)$

결국,

(부호)1_(기수)1000010 1_(가수) 1110110 10000000 00000000

계산하면 이런 결과가 나온다.

3. Type Casting

기존 데이터 타입을 다른 데이터 타입으로 바꾸는 것을 말한다.

일반 변수의 캐스팅은 작은형에서 큰형 으로만 한다. 반대의 경우 강제로 캐스팅 할 수 있지만, 원래의 데이터 값을 잃을 수 있으니 주의하도록 하자.

int -> unsigned int -> long -> unsigned long -> long long -> unsigned long long -> float -> double -> long double

The standard header **<stdint.h>**는 다음과 같다.

```

/* TYPE DEFINITIONS */
typedef signed char int8_t;
typedef short int16_t;
typedef long int32_t;
typedef long long int64_t;

typedef unsigned char uint8_t;
typedef unsigned short uint16_t;
typedef unsigned long uint32_t;
typedef unsigned long long uint64_t;

typedef signed char int_least8_t;
typedef short int_least16_t;
typedef long int_least32_t;
typedef long long int_least64_t;

typedef unsigned char uint_least8_t;
typedef unsigned short uint_least16_t;
typedef unsigned long uint_least32_t;
typedef unsigned long long uint_least64_t;

typedef signed char int_fast8_t;
typedef short int_fast16_t;
typedef long int_fast32_t;
typedef long long int_fast64_t;

typedef unsigned char uint_fast8_t;
typedef unsigned short uint_fast16_t;
typedef unsigned long uint_fast32_t;
typedef unsigned long long uint_fast64_t;

typedef long intptr_t;
typedef unsigned long uintptr_t;

typedef long long intmax_t;
typedef unsigned long long uintmax_t;

```

<코딩HW>

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     char c = 127;
6     c = c + 1;
7
8     printf("%d\n", c);
9     return 0;
10 }

```

결과 -128이 나오는 이유는?

char이 저장해야하는 범위를 넘어서 overflow발생한다.

따라서, 다시 최솟값부터 시작하게 되어 -128값이 나오게 된다.

char은 부호를 고려하는 1byte 메모리이며,

1개의 부호비트와 7개의 비트를 모두 데이터 비트를 사용한다.

char을 10진수로 표현하면 -128~127이 된다.