



C언어 – HW1

임베디드스쿨1기

Lv1과정

2020. 07. 24

박성환

1. 간단요약정리

1. 컴파일러

- 각기 다른 프로세서에서 해석하는 명령어(기계어 형태)로 우리가 프로그램 짠 Source Code를 중간에 해석해주는 역할
- CPU가 사용하는 ISA(Instruction Set Architecture)라고 부른다.

2. 어셈블러

- 기계어와 1:1 대응이 되는 언어로 형태는 비슷하나 프로세서 제조사마다 쓰는 명령어 체계가 다르기에 저마다 조금씩 그 형태를 달리함
- 컴파일러가 사실은 Source Code를 Assembly Code로 변경해주는 것이고 어셈블러가 Assembly Code를 기계어로 바꿔주는 역할을 함

3. Microprocessor VS Microcontroller

- Microcontroller는 원하는 동작에 특화되어 필요한 명령어 Set과 주변 메모리, 모듈 controller 등이 내장되어 있음
- Microprocessor는 범용적으로 사용되며 많은 명령어 Set으로 구성되어 있음, 여러 다중작업 처리에 유리함

2. HW1

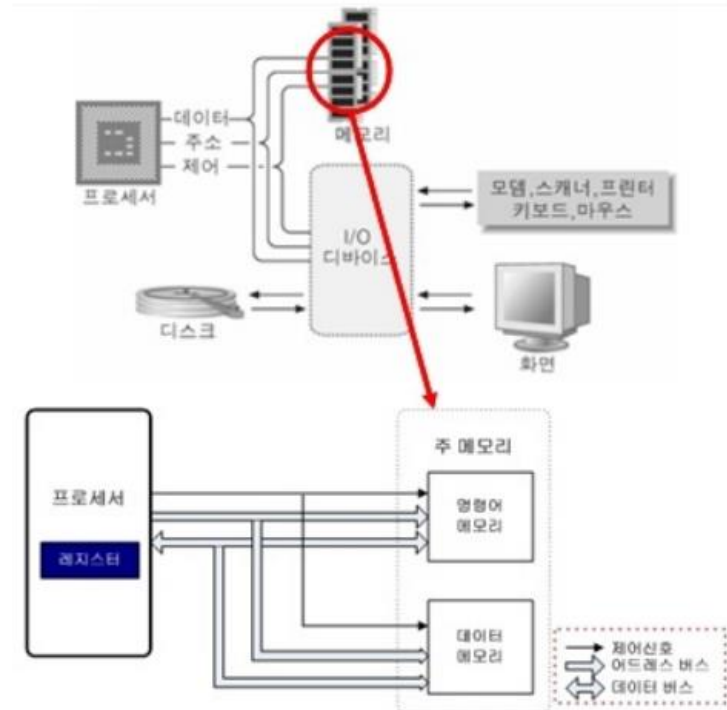
Q. C의 결과값이 136이 아닌 이유?

A. Char Data Type은 signed 변수로 -128 ~ 127 범위까지임
당연하게도 $125+10$ 은 127을 초과했기 때문에 Overflow가 발생함
컴파일 결과 127을 넘어서 -128 -127 -126 ... -121이 나옴

3. 폰노이만 구조 VS하버드 구조

1. 폰노이만 구조

1) 폰노이만 구조

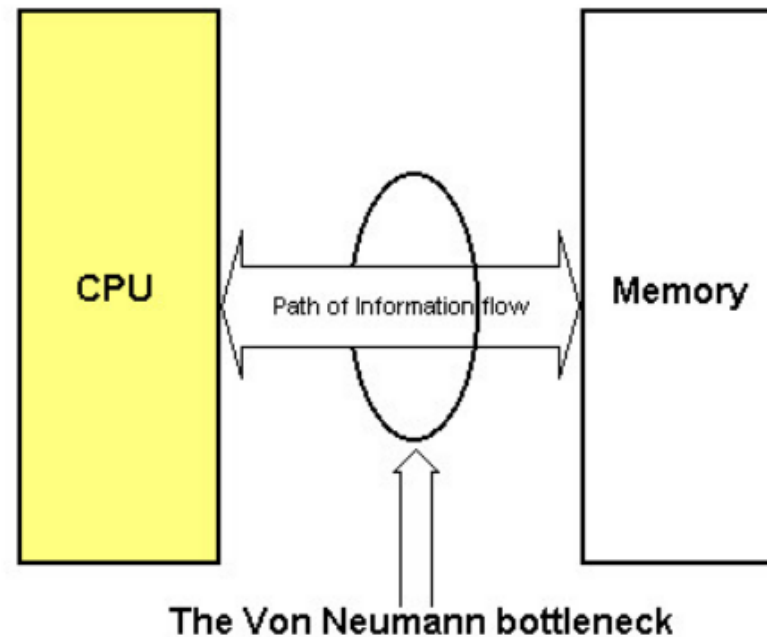


- 폰 노이만이 고안한 내장 메모리 순차처리 방식이다.
- 데이터 메모리와 프로그램 메모리가 구분되어 있지 않아 하나의 버스를 가지고 있는 구조를 말한다.
- CPU는 메모리로부터 명령을 읽고, 메모리로부터 데이터를 읽고 쓰기도 하는데, 명령과 데이터는 같은 신호 버스와 메모리를 사용하기 때문에 동시에 접근하는 것은 불가능하다.
- 폰 노이만 구조는 소프트웨어(프로그램)만 교체하면 되기 때문에, 그 이전의 컴퓨터들보다 범용성이 크게 향상된다.
- CPU, 메모리, 프로그램 구조를 갖는 프로그램 내장방식 컴퓨터 아이디어를 처음 제시하였고, 그 이후에 나온 컴퓨터는 대부분 폰 노이만의 설계를 기본 구조로 한다.

3. 폰노이만 구조 VS하버드 구조

1. 폰노이만 구조

2) 폰노이만 구조의 단점 - 병목현상



- 기억장소의 지연현상을 일컫는데, 나열된 명령을 순차적으로 수행하고, 그 명령은 일정한 기억장소의 값을 변경하는 작업으로 구성되는 폰 노이만 구조에서 병목현상은 나타날 수 밖에 없다.
- 메모리의 값을 읽고 쓰는 구조이기 때문에 명령과 데이터를 접근할때 병목현상이 생길 수 밖에 없다.
- 이러한 문제를 완화하기 위해 하버드 구조가 나타나게 되었다.

3. 폰노이만 구조 VS하버드 구조

1. 폰노이만 구조

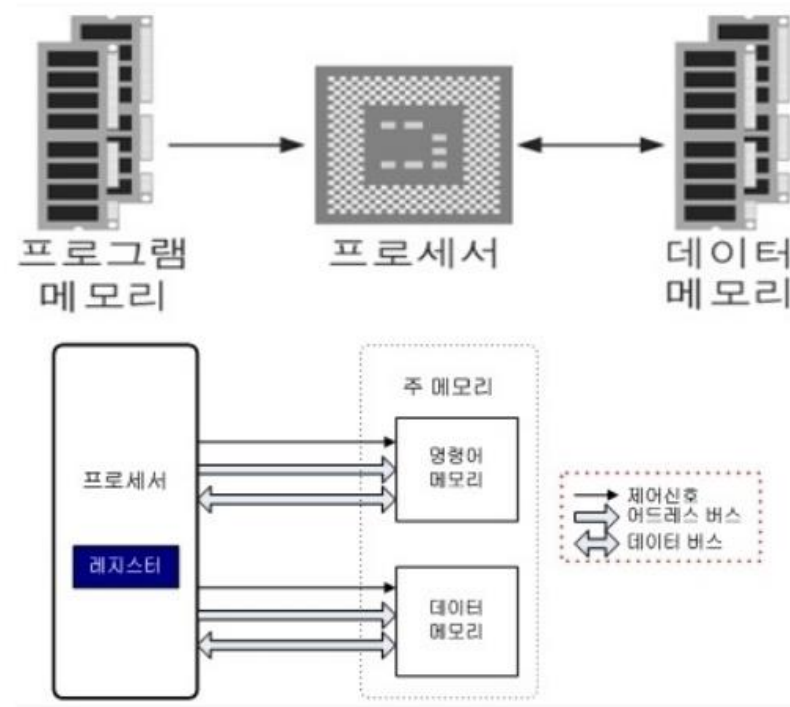
3) 폰노이만 구조의 단점 대책

해결방안	설명
병렬처리 개념 도입	<ul style="list-style-type: none">- 다중 프로세서/다중 데이터 스트림 처리가 가능한 병렬처리 컴퓨터 등장- 명령어 병렬처리 기술(Pipeline, Super scalar 등) 적용- SMP(Shared Memory Processor), MPP(Massively Parallel Processor)형태의 병렬처리- 하나의 Processor에서 여러개의 thread를 처리하는 multi-core 및 hyper-threading 기술 출현
주 기억 장치 병목 해결	<ul style="list-style-type: none">- Bus를 instruction용과 data용으로 분리- CPU에 Memory Controller를 내장하는 방식 도입- CPU와 주기억장치사이에 고속의 cache memory 구성- 최근에는 Bus를 읽기용과 쓰기용으로 세분화하는 Hyper transport 기술 출현
Harvard 아키텍처와 병행 사용	<p>최근 폰 노이만과 하버드 구조를 함께 사용한 고성능 CPU 개발</p> <ul style="list-style-type: none">- Harvard Architecture: 칩에 내장된 캐시 메모리는 명령어 캐시와 데이터 캐시로 분리- Von Neumann Architecture: Cache가 적중하지 못한 경우 데이터와 명령어가 구분되지 않는 메모리에서 데이터를 로딩

3. 폰노이만 구조 VS 하버드 구조

2. 하버드 구조

1) 하버드 구조

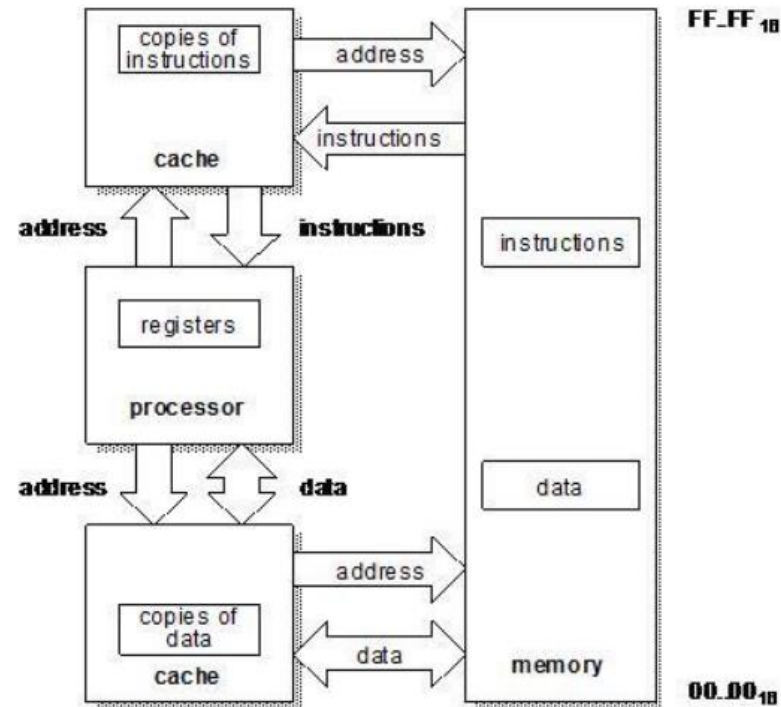


- 폰 노이만 구조와 가장 큰 다른점은 명령용 버스와 데이터용 버스가 물리적으로 분할되어 있다는 점이다.
- 하버드 구조에서는 명령을 메모리로부터 읽는 것과 데이터를 메모리로부터 읽는 것을 동시에 할 수 있다.
- 폰 노이만 구조에서 생기는 병목현상이 적어 명령의 처리를 끝내자마자 다음의 명령을 읽어들이 수 있기 때문에 더 빠른 속도를 낼 수 있다.
- 하지만 이러한 처리 속도를 높이려면 많은 전기 회로가 필요하다는 것이 단점이다.
- 이러한 문제를 완화하기 위해 수정된 하버드 구조가 도입되었다.

3. 폰노이만 구조 VS 하버드 구조

2. 하버드 구조

2) 수정된 하버드 구조

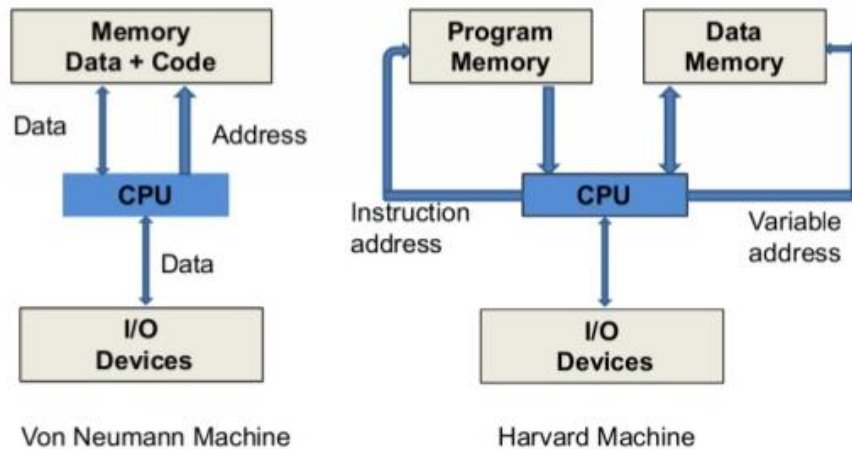


- 하버드 구조에서 사용했던 통합 캐시 메모리를 분리하여 하나의 클럭 사이클에서 적재(Load)와 저장(Store) 명령어를 동시에 실행할 수 있도록 해준다.
- 캐시 메모리 장치는 명령용과 데이터용으로 구분되어 있다.
- 하버드 구조를 캐시메모리 장치의 적용하였고, 폰 노이만 구조를 CPU 외부(주 메모리)에 적용하였다.
- 성능이 좋은 CPU 설계에서는 수정된 하버드 구조를 도입하고 있다.

3. 폰노이만 구조 VS 하버드 구조

3. 구조 한눈 비교

Von Neumann vs. Harvard Architecture



구분	Von Neumann	Harvard
목적	CPU는 한번에 하나의 명령어만 실행가능	병렬처리를 위해 메모리 구분
메모리	<ul style="list-style-type: none"> - 하나의 메모리 공유 - 단일 read-write memory 사용 	<ul style="list-style-type: none"> - 명령어, 데이터 메모리 분리 - 명령어 메모리는 read-only memory도 사용가능
프로세스	메모리 -> FI -> 메모리 -> DI -> 메모리 -> EI -> Store(메모리) 순차적으로 수행	<ul style="list-style-type: none"> - 명령어 메모리 -> FI - Store -> 데이터 메모리 - 동시에 명령어와 데이터 처리 가능
장점	공용 메모리 사용으로 상대적 구현 비용 저렴	파이프라이닝 기술 사용을 위한 최적의 환경 제공
단점	파이프라이닝 시 메모리 공유 문제 발생	<ul style="list-style-type: none"> - 별도 메모리 사용으로 구현 비용 증가 - 회로 구조 복잡
적용사례	일반적인 범용 CPU	<ul style="list-style-type: none"> - Microchip Technology의 PIC - Atmel AVR

4. 출처

<https://sangcho.tistory.com/entry/%ED%8F%B0-%EB%85%B8%EC%9D%B4%EB%A7%8C-%EA%B5%AC%EC%A1%B0-vs-%ED%95%98%EB%B2%84%EB%93%9C-%EA%B5%AC%EC%A1%B0>

<http://www.jidum.com/jidums/view.do?jidumId=394>



감사합니다.