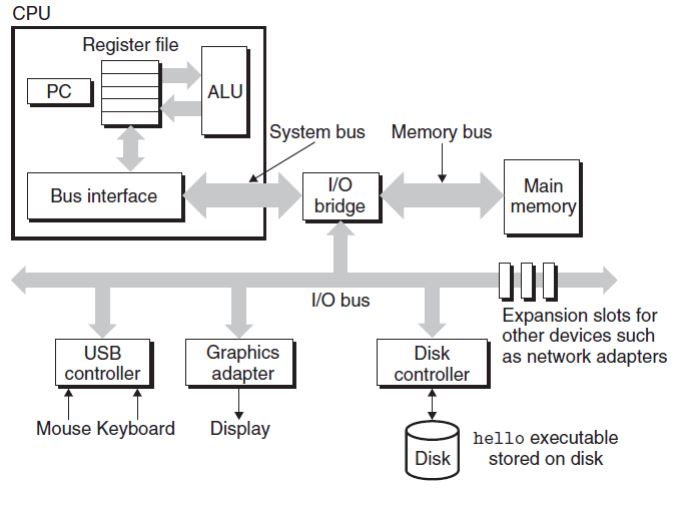
1. 컴퓨터 시스템 개요
   1. 컴퓨터의 구성요소
   2. 프로세서
      1. 중앙처리장치
      2. 제어장치, 산술논리 처리기, 레지스터
      3. 메모리
         1. 주 기억 장치
         2. 캐쉬 메모리
         3. 저장 장치
      4. 입출력 장치
         1. 키보드 마우스 모니터 등
      5. 버스
   3. 폰노이만 아키텍쳐
      1. 메모리에 내장된 프로그램을 프로세서가 실행
      2. I/O , Processor, Memory 형태
   4. 컴퓨터 시스템의 일반적 구조



* 1. ISA
     1. 컴퓨터 하드웨어와 소프트웨어 사이의 인터페이스
     2. 컴퓨터 시스템의 상태와 동작을 추상화
     3. 어셈블리어도 종류가 다르다. 컴파일러가 ISA에 맞춰서 번역해주는 것
     4. ISA를 지원하기 위한 하드웨어를 구현하는 구조 : Microarchitecture
  2. 프로세서의 구성과 기능
     1. 프로세서의 구성 : ALU, CONTROL UNIT, REGISTER, CACHE
     2. 프로세서의 기능
        1. 메모리로부터 명령과 데이터를 호출, 저장, I/O (레지스터)
        2. 명령어 해석 (제어장치)
        3. 명령어 실행 (ALU)
     3. 명령 실행 싸이클
        1. 인출 – 레지스터가 지정한 위치 명령어 호출
        2. 해석 – ISA에 따라 명령어를 해석하여 제어 신호 생성
        3. 실행 – 제어 신호에 따라 명령어 실행
        4. 저장 – 명령어 실행 결과를 메모리에 저장
  3. 하드웨어의 기본 단위 – 부울 대수
     1. 특정 로직 게이트에 들어가면 전기 신호에 따라 BOOL 논리 연산이 가능
     2. 부울 대수와 로직 게이트의 조합으로 반도체 회로등이 설계된다.

1. 프로그래머 관점에서 컴퓨터 시스템의 이해
   1. 컴퓨터 시스템
      1. 컴퓨터 시스템은 물리적인 하드웨어의 작업을 논리적 단위로 추상화하여 프로그래머에게 제공.
      2. ISA , 가상 메모리, 프로세스와 파일 등
      3. 추상화의 특성과 그 실체를 파악해야 올바른 프로그램을 작성할 수 있다.

Ex) INT, FLOAT 타입의 한계

* + 1. 버그, 최적화, 시스템 소프트웨어, 악성코드 방어등의 문제해결을 위해 하드웨어 레벨의 시스템을 이해해야 한다.
  1. 컴퓨터 시스템의 실체
     1. 메모리
        1. 메모리는 유한하므로 효율적 할당과 관리가 필요
        2. 메모리 참조버그는 프로그램에 커다란 영향을 끼친다.
        3. 메모리 성능을 극대화할 수 있다. CACHE FRIENDLY CODE
     2. 컴파일
        1. 헤더와 합쳐진 코드는 컴파일러에 의해 어셈블리어로 번역되고 어셈블리어는 어셈블러에 의해 목적 코드로 변환된다. 그리고 링커가 참조된 라이브러리들과 합쳐서 실행 가능한 프로그램을 만들어 낸다.
        2. 애매한 Linking Error 등의 버그를 잘 잡기위해서 컴파일러가 무슨일을 하는지 잘 알 필요가 있다.

1. 컴퓨터 시스템의 실제 예
   1. 아두이노, 라즈베리 파이
      1. 오픈소스 임베디드 개발 보드
      2. 실제 모든 작업(I/O, Memory, ALU)은 AVR컨트롤러 칩 내부에서 처리하고 나머지는 그 연장선
   2. AVR 시스템
      1. 8bit RISC 아키텍처
         1. 8 bit 범용 레지스터
         2. 16 bit는 간접 주소 포인터로 사용
      2. 플레쉬 메모리, 주 메모리
      3. 범용 입출력 라인들, 10bit ADC, 타이머, 카운터, 인터럽트…
      4. 크게보면 기본적인 폰 노이만 컴퓨터 모델은 변하지 않는다.