운영체제 Chapter 2

1. 하드웨어의 구성

컴퓨터는 중앙처리장치, 메인메모리, 입력장치, 출력장치, 저장장치로 구성된다.

CPU는 명령어를 해석하여 실행하는 장치

2. 폰노이만의 구조 : 폰노이만의 구조는 CPU, 메모리, 입출력장치, 저장장치가 버스로 연결되어 있는 구조를 말한다.

그리고 폰노이만 구조에서 가장 중요한 특징은, ‘모든 프로그램은 메모리에 올라와야 실행할 수 있다는 것이다.’

4. 하드웨어 사양 관련 용어

클록 CPU의 속도와 관련된 단위

헤르츠 : 헤르츠는 클로긱이 발생하는 속도를 나타내는 단위, 1초에 클록틱이 한번이면 1hz, 1000번이면 1khz이다.

시스템 버스와 CPU 내부버스

시스템 버스는 메모리와 주변장치를 연결하는 버스로 FSB(front-side-bus), 즉 전면버스

CPU 내부 버스는 CPU 내부에 있는 장치를 연결하는 버스로 BSB, 즉 후면 버스라고 한다.

CPU 내부 버스가 시스템버스보다 빠르다.

그래서 이 속도차를 해결해야함

2 CPU와 메모리

CPU는 명령어를 해석하여 실행하는 장치. CPU는 산술논리 연산장치, 제어장치, 레지스터

데이터 레지스터(DR)메모리에서 가져온 데이터를 임시 보관할 때 사용. 범용 레지스터 불림

주소 레지스터(AR) : 데이터 또는 명령어가 저장된 메모리의 주소는 주소 레지스터에 저장된다.

PC : 다음에 실행할 명령어의 주소를 기억하고 있다가 제어장치에 알려준다.

명령어 레지스터(IR) : 명령어 레지스터는 현재 실행 중인 명령어를 저장한다.

메모리 주소 레지스터(MAR) : 메모리에서 데이터를 가져오거나 반대로 메모리로 데이터를 보낼 때 주소를 지정하기 위해 사용한다.

메모리 버퍼 레지스터(MBR) : 메모리에서 가져온 데이터나 메모리로 옮겨 갈 데이터를 임시로 저장한다. 메모리 버퍼 레지스터는 항상 메모리 주소 레지스터와 함께 동작한다.

버스의 종류  
 제어 버스, 주소 버스, 데이터 버스

버스의 대역폭은 한 번에 전달할 수 있는 데이터의 최대 크기를 말한다.

휘발성 메모리  
DRAM(동적): 일정 시간이 지나면 사라지므로 일정 시간마다 다시 재생시켜야 한다. 즉 시간이 지나면 데이터가 사라지기 때문에 재생이 필요하다는 의미. 메인메모리에 사용.  
  
SRAM(정적):전력이 공급되는 동안에는 데이터를 보관할 수 있어 재생할 필요가 없다. 따라서 속도는 빠르지만 가격이 비싸다. 고속 메모리에 사용

SDRAM은 DRAM이 발전된 형태로 SRAM과는 완전히 다른 종류의 램이다. SDRAM은 클록틱(펄스)이 발생할 때마다 데이터를 저장하는 동기 DRAM이다.

비휘발성 메모리  
플래시 메모리  
FRAM  
PRAM

2.3 부팅  
부팅 : 운영체제를 메모리에 올리는 과정  
마스터 부트 레코드 : 하드디스크의 첫 번째 섹터를 가리키며, 운영체제를 실행하기 위한 코드인 부트스트랩이 이곳에 저장되어 있다.  
부트스트랩 코드 : 운영체제를 메모리로 가져와 실행하는 역할을 하는 작은 프로그램이다.

버퍼는 속도에 차이가 있는 두 장치 사이에서 그 차이를 완화하는 역할을 한다.

스풀(SPOOL): 스풀은 CPU와 입출력장치가 독립적으로 동작하도록 고안된 소프트웨어적은 버퍼로, 대표적인 예는 프린터에 사용되는 스풀러이다.

캐시는 메모리와 CPU 간의 속도 차이를 완화하기 위해 메모리의 데이터를 미리 가져와 저장해두는 임시 장소이다.

캐시의 변경된 데이터를 메모리에 반영하는 데에는 즉시 쓰기 방식과 지연 쓰기 방식이 있다.

즉시 쓰기(write through):캐시에 있는 데이터가 변경되면 이를 즉시 메모리에 반영하는 방식이다. 메모리와의 빈번한 데이터 전송으로 인해 성능이 느려진다는 것이 단점이지만, 메모리의 최신 값이 항상 유지되기 때문에 급작스러운 정전에도 데이터를 잃어버리지 않는다.

지연 쓰기(write back) : 캐시에 있는 데이터가 변경되면 이를 즉시 메모리에 반영하는 것이 아니라 변경된 내용을 모아서 주기적으로 반영하는 방식으로, 카피백이라고도 한다. 메모리와의 데이터 전송 횟수가 줄어들어 시스템의 성능을 향상할 수 있으나 메모리와 캐시된 데이터 사이의 불일치가 발생할 수도 있다는 것이 단점이다.

인터럽트  
폴링 : CPU가 입출력장치의 상태를 주기적으로 검사하는 방식. 효율이 떨어진다.  
인터럽트 : 폴링방식을 해결하기 위해서 나온 방식.

인터럽트 방식의 동작 과정

1. CPU가 입출력 관리자에게 입출력 명령을 보낸다.
2. 입출력 관리자는 명령받은 데이터를 메모리에 가져다놓거나 메모리에 있는 데이터를 저장장치로 옮긴다.
3. 데이터 전송이 완료되면 입출력 관리자는 완료 신호를 CPU에 보낸다.

CPU에게 알려주기 위해 인터럽트 번호를 사용한다. 윈도우 운영체제에서는 IRQ라고 부름

직접 메모리 접근(Direct Memory Access,DMA)

메모리 매핑 입출력  
 CPU가 사용하는 메모리 공간과 직접 메모리 접근을 통해 들어오거나 나가는 데이터를 위한 공간을 분리하는 것이다. 메모리의 일정 공간을 입출력에 할당하는 기법을 메모리 매핑 입출력(memory mapped I/O,MMIO)이라고 한다.

사이클 훔치기 : CPU가 양보한다. 같이 사용하는 경우

병렬 처리 시 고려 사항

* 상호 의존성이 없어야 병렬 처리가 가능
* 각 단계의 시간을 거의 일정하게 맞춰야 병렬 처리가 원만하게 이루어진다.
* 전체 작업 시간을 몇 단계로 나눌지 잘 따져 보아야 한다.

병렬 처리 기법  
 1. 명령어 패치(Instruction Fetch, IF): 다음에 실행할 명령어를 명령어 레지스터에 저장한다.

2. 명령어 해석(Instruction Decode,ID) : 명령어를 해석한다.

3. 실행(Execution, EX) : 해석한 결과를 토대로 명령어를 실행한다.

4. 쓰기(Write Back,WB) : 실행된 결과를 메모리에 저장

-파이프 라인 기법의 문제 : 데이터 위험, 제어 위험, 구조적 위험  
 데이터 위험 : 데이터의 의존성 때문에 발생  
 해결방안 : 파이프라인의 명령어 단계를 지연하여 해결ㅉ

제어 위험 : if goto 문 같은 명령에서 발생하는 제어 위험(control hazard)은 PC값을 갑자기 변화시켜 발생하는 위험.  
 해결 방안 : 분기 예측이나 분기 지연 방법으로 해결  
 구조 위험 : 서로 다른 명령어가 같은 자원에 접근하려 할 때 발생하는 문제  
 해결방안 : 해결하기 어렵다.

* 슈퍼스칼라 기법  
  파이프라인을 처리할 수 있는 코어를 여러 개 구성하여 복수의 명령어가 동시에 실행되도록 하는 방식

코어를 2개 구성하여 각 단계에서 동시에 실행되는 명령어가 2개라는 점. 이 경우 전체적으로 동시에 실행되고 슈퍼스칼라 기법에서는 처리되는 명령어가 상호 의존성 없이 독립적이어야 하며, 이를 위한 처리도 컴파일러에서 이루어지도록 조정

슈퍼파이프라인 기법  
 파이프라인 기법을 강화한 것.

슈퍼파이프라인 슈퍼스칼라 기법

병렬 처리 기법을 모두 합쳐 놓은 것

VLIW(Very Long Instruction Word)  
 CPU가 병렬 처리를 지원하지 않을 경우 소프트웨어적으로 병렬 처리를 하는 방법.