컴퓨터구조
기말고사 정리

8 ~ 12 주차 강의 내용 정리.

이시헌

2023-12-8

8주차 수업 정리.

1. 프로세스(process): 데이터를 처리하기 위해 현재 진행되는 작업.
   A. 컴퓨터에서의 처리: 전기 신호를 다른 신호로 변경시키는 것.
   B. 프로세싱(processing): 처리 과정 / 처리.
   C. 프로세서(processor): 처리 장치.
      i. 물리적인 프로세서: cpu.
      ii. 논리적인 프로세서: 번역 프로그램.
   D. 즉, cpu는 데이터를 처리한다.

2. cpu: { ALU(연산장치) + 레지스터(임시 저장소) + 제어장치 + 내부 버스들 }
   A. ALU: 산술 및 논리연산 장치. (cpu워드 단위로 데이터 처리.)
   B. 레지스터: cpu에 내장된 임시 저장소. (cpu워드 단위로 입출력.)
   C. 제어장치: 장치 제어 및 장치 제어를 위한 동기 용도의 타이밍 신호 생성.
   D. 버스: 물리적 신호선들.

3. 마이크로프로세서: 작은 프로세서. (일반적으로 우리 일상에 있는 모든 cpu.)
   A. 마이크로 컴퓨터: 마이크로프로세서를 사용한 컴퓨터.

4. 컴퓨터 워드가 64bit인 시스템에서 32bit 프로그램을 사용 가능하도록 하는 이유: 한 번에 처리하는 데이터의 양이 작은 프로그램의 경우, 32bit 프로그램이 더 빠르고 효율적으로 동작할 수 있다.

5. 보조 프로세서: cpu의 일을 대신 처리해주는 프로세서들.
    A. 대표적인 보조프로세서.
        i. FPU: 부동소수점 연산 장치.
        ii. GPU: 그래픽 처리 장치. (그래픽 가속기)
    B. 특징: GPU의 경우는, 메모리, I/O장치, 중앙처리장치를 모두 갖추고 있다. 다만, 컴퓨터의 모든 구성 요소를 갖췄음에도 cpu 없이는 동작하지 못한다는 점에서 여전히 보조 프로세서이다.

6. 코어. (cpu의 개별 코어에 대하여)
    A. 개별 코어는 하나의 완전한 cpu이다.
        i. 각 코어 내부에 저마다의 레지스터, ALU, 컨트롤러, 버스가 모두 포함되어 있다.
    B. L3캐시: 여러 코어 사이의 데이터 입출력을 완충하는 레지스터(버퍼).
    C. 프로세서는 모든 코어와 보조 프로세서를 포함한 하나의 칩셋을 지칭한다.

7. E 표기법: 10.0e+1 == 100.0 // e-1 하면 소수점 반대로 이동.

8주차 수업 정리 끝.

9주차 수업 정리.

1. 주기억장치(메모리): 일반적으로 RAM을 지칭함. 고속의 휘발성 메모리로, cpu가 바로 꺼내 쓸 수 있도록 데이터를 저장하는 임시 저장 장치.
   A. cpu의 레지스터를 대용량으로 만들지 않고 메모리를 따로 구성하는 이유.
      i. 비용과 크기 등의 물리적 사유.
      ii. 확장성(기억장치 증설의 편의성 증대)
   B. 당장 필요한 데이터들을 미리 불러와서 상주시킨다(보관한다). (바로 실행 가능하게끔 보관한다는 의미)
   C. 메모리가 클수록 프로그램을 동시에 더 많이 상주시킬 수 있다. (즉, 동시에 더 많은 작업을 시킬 수 있다.)
   D. 실행 전 / 실행 중 / 실행 후의 신호들을 섞이지 않게 논리적으로 구분해 줌. 이때 주소를 사용함.
   E. 램 상주 프로그램(TSR, Terminate and Stay Resident): 실행 후 종료될 때 제거되지 않고 메모리에 계속 상주하는 프로그램.
      i. 운영체제의 커널, 바이러스 백신 등의 백그라운드 프로세스들을 말함.

2. 보조기억장치 (주기억장치에 비하여 – 저비용, 비휘발성, 대용량)
   A. 주기억장치의 크기가 커지면 목표 데이터의 탐색 소요시간이 증가함. 필요한 만큼만 저장과 망각을 반복하도록 주기억장치와 보조기억장치를 분리시킬 필요가 있다.
   B. 주기억장치가 비싸서 적게 설치하는 것이 아니다. 고속으로 만들기 위하여 고비용이 된 것 일 뿐이고, 탐색속도를 위하여 일정 이상 설치하지 않는 것이다. 아무리 비싼 하이엔드 PC도 램을 128GB 이상 설치하지 않는 것을 떠올려라. 아무리 이진탐색이라도 주기억장치가 너무 크다면 느려질 수 밖에 없다.

3. n 비트로 표현 가능한 이진수 개수 = 2^n

A. 2002년 IEEE 1541 표준 지정 표기법.

i. Byte -> B

ii. bit -> b //소문자로 써야 비트라는 것을 명심하라.

B. GB(2진수 단위) / GiB(10진수 단위)

i. 16MB = $16 * 2^{10} * 2^{10} * 8$ bit $//2^{10} = 1024$

ii. 16MiB = $16 * 10^3 * 10^3 * 8$ bit $//10^3 = 1000$

4. 빅 엔디언 / 리틀 엔디언

A. 주기억장치를 낮은 주소부터 사용할지, 높은 주소부터 사용할지에 대한 논리적인 방법론.

B. 두 방법에는 장단점이 있다. 시스템에 따라 사용하는 방법이 다를 수 있다.

<table>
<tr><td rowspan="2">0x12345678
(16진수 주소)</td><td>빅 엔디언 : 0x12 - 0x34 - 0x56 - 0x78</td></tr>
<tr><td>리틀 엔디언 : 0x78 - 0x65 - 0x43 - 0x21</td></tr>
<tr><td colspan="2">낮은 주소(MSB) <- 메모리 주소 -> 높은 주소(LSB)</td></tr>
</table>

C. 빅 : 낮은 주소부터 낮은 비트 저장.

D. 리틀 : 높은 주소부터 낮은 비트 저장.

5. 메모리의 속도: 데이터의 이동 시간 + 읽고 쓰는 속도

A. 메모리에 읽기 쓰기를 하는 주체는 cpu임을 명심하라. (사용자가/cpu가/입출력장치가 메모리에 데이터를 넣는 것이 메모리 쓰기임.)

6. 읽기 쓰기 시간

   A. 읽기 시간 : cpu가 주소를 RAM에 보낸 순간 〈-〉 데이터를 읽어오는 동작 완료 순간.

   B. 쓰기 시간 : cpu가 주소와 데이터를 보낸 순간 〈-〉 데이터가 저장 완료되는 순간.

7. 액세스 타임(접근 시간): 데이터가 있는 위치를 찾는데 걸리는 시간.

   A. 즉, 읽기 쓰기 시간에는 액세스 타임이 포함되어 있다.

8. 저장장치의 속도에 대하여

   A. 저장장치의 읽기/쓰기 시간이 곧 속도이다.

   B. 이때 이 시간에는 '버스의 전송 소요 시간'과 '액세스 타임'이 있다.

      i. 버스의 속도는 대체로 비슷하므로, 저장장치의 성능은 일반적으로 액세스 타임과 거의 비례한다.

      ii. 예: RAM이 cpu 내부 레지스터보다 느린 것은, 액세스 타임이 더 길기 때문.

      iii. 물론 이 속도는 아주 복합적인 요인에 영향받으므로, 항상 액세스 타임이 저장장치의 성능을 대변하는 것은 아니다. 일반적으로 ALU에 가까운 장비일수록 고성능으로 제작된다는 점에서 생기는 경향성으로 이해하는 것이 좋겠다.

9. 가상 메모리: 보조기억장치의 일부를 주기억장치로 여기게끔 논리적으로 구성된 메모리.

   A. OS가 메모리 부족시에 자동으로 보조기억장치의 일부를 가상 메모리로 잡아서 사용한다.

10. 물리 주소 / 가상 주소

A. 물리 주소: 실제 메모리의 주소.

B. 가상 주소: 가상메모리의 주소를 다루기 위하여, 주기억장치에 저장된 보조기억장치의 내부 주소들.

11. 하드디스크 스와핑(스왑, 페이징)

A. 메모리 부족 시 램의 내용 일부를 보조기억장치에 옮기고, 지금 상주시켜야 하는 다른 데이터를 램으로 집어넣는 행위.

B. 스왑을 할 때, 가장 낮은 우선순위(사용 빈도)를 가진 메모리의 데이터를 스왑 파일(페이지)로 만들어서 보조기억장치에 저장한다. 이 때, 이 스왑 파일 자체도 가상메모리에 속한다.

C. 페이지 테이블: 가상메모리와 실제 메모리를 모두 합한 논리적 메모리의 모든 주소를 모아서 만든 표.

D. 쓰래싱(thrashing): 페이징이 과도하게 일어나는 상태. 그로 인하여 작업의 진전이 거의 없거나 아예 없는 상태. (요즘은 이럴 일이 거의 없다.)

9주차 수업 정리 끝.

10주차 수업 정리.

1. 캐시 메모리: cpu와 주기억장치 사이의 메모리.
   A. cpu 내부의 레지스터가 주기억장치에서 데이터를 가져오는데 소요되는 액세스 타임 감소를 위해 사용. (메모리 계층 간의 속도 차이를 완충시켜서, 기억장치의 평균 액세스 타임을 향상시킴.)
   B. 일반적으로 cpu 내부나, cpu와 아주 가까운 곳에 위치함.
   C. cpu 내부의 레지스터보다는 느리고, RAM보다는 빠름.
   D. 실행중인 프로그램의 일부를 보관함. (RAM에 상주중인 프로그램 중, 당장 필요할지도 모르는 부분들을 보관함.)
   E. cpu 내부의 ALU와 레지스터에 가까울수록 작은 숫자를 붙임. L1-L2-L3-… (일반적으로 L1, L2캐시는 코어에 내장, L3캐시는 코어들 사이에 연결. (코어간 병렬연산이 가능케 함->L3가 크면 멀티코어 성능 향상))
   F. 캐시 용량을 무한정 늘리지 않는 이유
      i. 경제적 이유(비싸다) + 물리적 부피의 한계
   G. cpu가 액세스 한 내용은 일단 캐시에 복사된다. 이후 cpu가 동일 내용을 다시 요구시, 캐시에서 빠르게 응답할 수 있다.
      i. 이때 cpu가 원하는 내용을 캐시에서 발견하는 것을 hit이라 하고, 발견하지 못하면 miss라 한다.
   H. 캐시에 기록하는 데이터도, 데이터 블록 단위로 만들고 없어진다.
   I. 주의! RAM은 cpu와 별개로 보아도 되지만, RAM은 cpu의 속도에 영향을 끼친다. 다만, 연산 자체가 빨라지거나 느려지는 것은 아니고, 연산해야 할 데이터를 주고받는 것에 지연이 걸리는 것.
2. 디스크 캐시: 주기억장치의 일부를 보조기억장치로 사용하는 것.
   A. 가상메모리와 반대다. OS가 자동으로 주기억장치의 일부를 할당하여 사용한다. (많게는 메모리의 1/3가량도 할당한다.)
   B. OS가 가상메모리와 디스크 캐시의 비율을 적절히 관리해준다. 서로 반대로 작동하기에, 적절한 비율이 있는 것.

3. RAM

A. RAM(Random Access Memory) / SAM(Sequential Access Memory)

i. 랜덤 - 어디에 접근하든 액세스타임이 일정. (이진탐색 기반)

ii. 순차 - 접근 위치에 따라 액세스타임이 달라짐. (일반적으로 OS가 만든 인덱스를 기반으로 앞에서부터 탐색)

B. SRAM / DRAM

i. Static-RAM: 전기 공급이 있는 한 정보 기록 유지.

ii. Dynamic-RAM: 시간에 따라 정보가 소멸하여, 주기적으로 신호를 재생시켜야 함. + 구조가 간단해 집적과 양산이 용이함.

C. SDRAM (Synchronous-DRAM)

i. 메모리 컨트롤러가 아닌, 시스템 버스에 직접 동기하는 DRAM.

ii. 버스로부터 오는 신호에 맞추어, 램이 신호를 내보내는 것.

iii. 메모리 컨트롤러의 연산 요구량이 감소함.

D. DDR SDRAM (Double Data Rate - SDRAM)

i. 1회의 클럭 펄스에 2회의 신호를 송출하는 SDRAM.

ii. 버스로부터의 신호(클럭 펄스)의 상승 하강 시점에 각각 신호 전송.

iii. DDR 뒤에 붙는 숫자는, 버스로부터의 신호 당 램의 전송 신호 수가 배수로 늘어났음을 의미함.

DDR	1	2	3	4
1회 클럭 주기당 송출 신호 수	2bit	4bit	8bit	16bit

4. 데이터 전송 속도 표현

A. bps: bit/s

B. Kibit : 10진수 기준 / Kbit : 2진수 기준.

C. 대역폭: 신호의 전달에 사용되는 주파수의 폭. (고 주파수: 신호 전송량 큼: 대역폭이 큼.

10주차 수업 정리 끝.

11주차 수업 정리.

1. 보조기억장치: 온갖 데이터를 안전하게 저장해두는 장치.
   A. OS, 드라이버, 응용 프로그램 등등, 무엇이든 장기 저장한다.
   B. drive: 보조기억장치를 논리적으로 인식하는 단위. (거의 모든 보조기억장치를 OS에서 드라이브라 부른다.)

2. SSD: Solid State Drive
   A. 기계적으로 움직이는 부분이 없는 보조기억장치. 주로 낸드 플래시 메모리를 사용한다. (메모리는 DRAM과 같은 구조이나, 장치의 회로 구성은 다르다.)
   B. HDD에 비하여 탐색속도가 월등히 빠르다. (SSD: 랜덤엑세스 / HDD: 순차 탐색)

3. 드라이브 문자: 여러 드라이브를 구분하기 위하여 OS가 부여하는 식별자. (A, B는 플로피 디스크를 위해 배정되어 있었다.)

4. 드라이브 인식: ROM BIOS 혹은 드라이브 장치에 내장된 드라이버로 OS 시동 중에 인식된다. (OS가 주체적으로 인식한다.)
   A. ODD같은 외부 드라이브는 대게 운영체제만으로 인식된다.

5. 가상 드라이브
   A. 내 컴퓨터에 물리적으로는 존재하지 않지만, OS가 논리적으로 갖고 있는 드라이브를 말한다. 종류는 다음과 같다.
      i. 네트워크상 드라이브 – SAMBA 등
      ii. 파일을 드라이브로 – 예: .iso 파일 (디스크 이미지 파일) 등
      iii. 장치를 드라이브로 – 예: 램디스크, 디스크 파티션 등

6. 디스크 파티션: 하나의 디스크를 논리적으로 다수의 드라이브로 나눈 것. 이때, 각각의 논리적 드라이브가 파티션이다.

7. 자기 디스크: 원형 평판(platter)에 자화 물질을 코팅하여 만들어진 디스크. 헤드로 디스크 표면을 자화 = 쓰기 / 현재 자화 상태를 읽기 = 읽기.

11주차 수업 정리 끝.

12주차 수업 정리.

1. drive(논리적) / disc(물리적) 저장장치.

2. 저장 매체: 데이터를 저장하기 위한 장치.
   A. 매체라 부르는 이유: 인간 입장에서 데이터와, 데이터가 저장된 상태를 구분하여 보기 때문. 즉, 저장 방식에 따라서 다른 매체라고 칭할 수 있음.
   B. FDD -> CD -> HDD -> SSD … 모두 매체가 계속 바뀐 것이다. (더 많이, 더 빨리 읽고 쓸 수 있는 **물질과 방식**으로 매체를 바꿔온 것.)

3. HDD: 트랙 + 섹터.
   A. 트랙: 기록이 가능한 한 개의 동심원 영역.
   B. 섹터: (원판 기반 저장장치들의) 읽고 쓰는 최소 단위.
   C. 디스크 용량: 섹터의 크기 * 섹터의 수.

4. 포맷(format): 데이터의 저장 방식을 변경하고, 기존 데이터를 읽을 수 없도록 함.
   A. 다른 말로 하자면, 파일 시스템을 변경하는 것. 그렇기에 포맷을 하면, OS가 디스크를 인지할 수 있도록 일부 공간을 할당하여 디스크 사용을 위한 정보를 저장한다. (파일시스템을 기록한다)

5. 디스크 기록 밀도: 한 섹터의 저장 밀도 + 한 장의 디스크의 섹터 밀도.
   A. 즉, 하나의 원판을 더 많이 쪼개고, 쪼갠 하나의 섹터에 더 많이 기록할수록, 기록 밀도가 높은 것.
   B. 단, 엑세스 타임을 고려하면 고밀도가 항상 좋은 것은 아니다.
6. 참고. 모든 보조기억장치는 저마다의 버퍼와 컨트롤러를 지닌다. 버퍼에 일단 저장해두고, 순차적으로 기록하는 것으로, 속도 차이를 완화한다.

7. 보조기억장치를 위한 인터페이스 발전 방향

A. 병렬 버스(페러렐 버스) -> 직렬 버스(시리얼 버스).

B. RAM의 발전 방향과 반대로 보이지만, 병렬 신호의 처리속도 한계가 있기에, 직렬 버스 자체의 성능을 올리는 것이 더 효율적이다. 그리고 이미 SATA의 속도가 충분하기에, 병렬로 구성할 이유가 없기도 하다.

8. RAID (Redundant Array od Inexpensive Disks)

A. 더 빠른 속도를 얻거나, 더 안전하게 저장하기 위하여, 복수의 디스크를 하나로 묶어서, 하나의 드라이브처럼 사용하는 것.

B. RAID-0 : 이론상 두 배 빨라짐. 안정성 없음.

C. RAID-1 : 이론상 두배 안전해짐. 속도 향상 없음.

D. 위의 두 가지를 잘 조합하고 다양한 기술을 적용하여 속도와 안정성의 균형을 잡는 것이 RAID 기술이다.

RAID	2	3	4	5	6
예시를 기준으로 한 설명	해밍코드 저장용 디스크 3개 + 데이터 디스크 4개 = 7bit 데이터 기반 저장.	패리티 비트 저장용 디스크 1개 + 데이터 디스크 4개 = 4bit 기반 데이터 저장 + 오류 수정용 패리티 비트 1bit 저장.	3과 유사하나 블록 단위로 패리티 비트를 구함. 총 디스크 개수는 여전히 5개.	4와 유사하나, 블록단위의 패리티 비트들을 하나의 디스크에 몰아넣지 않고, 5개의 디스크에 분산 저장.	5와 유사하나, 블록단위의 패리티 비트를 둘로 나누어 5개의 디스크에 분산 저장.

12주차 수업 정리 끝.

13주차 수업 내용은 별도의 기록물을 볼 것.

이상으로 컴퓨터 구조 수업의 8 ~ 12주차 수업 내용 정리 끝!