Chapter 6. Storage and Other I/O Topics

I/O Systems

- interact wth computers
- long- term memory
- Input, Ouput, Storage
- Data rate: 초당 바이트 수 or 초당 전송 횟수

Bus

- Parallel set of wires for data and synchronization of data transfer
- 단점
  - bottleneck
  - limited by physical factors: 와이어 길이, 연결수
- 대안
  - higth-speed serial connections with switches ⇒ 스위치 기반 고속 직렬 연결

- Processor-Memory buses
  - 짧고 빠름
  - 메모리 구조에 맞춰 설계
- I/O buses
  - 길고 다중 연결 허용
  - Bridge ⇒ Processor-Memory bus와 연결

- 장점
  - Versatility(다용도성)
    - 새 장치 쉽게 추가 가능
    - same bus standard 사용 시스템간 주변 장치 이동 가능
  - Low cost : 단일 와이어 세트가 여러 방식으로 공유
- 단점
  - Bottleneck
    - 버스의 bandwidth(대욕폭)이 최대 I/O throughput 제한
    - 버스의 길이, 연결된 장치 수
    - Widely varying latencies, data transfer rates
      - 매우 광범위한 지연시간, 전송률을 지원해야함

Bus Basics

- Control Lines
  - request, acknowledgements ⇒ 요청, 응답 신호 전달
  - 데이터 라인의 정보(데이터, 주소, 명령) 나타냄.
- Data Lines
  - 데이터, 주소, 복잡한 명령 전송

- **Bus Transaction**
  - 요청, 응답을 포함하는 버스 작업순서
    1. sending address
    2. receiving or sending Data

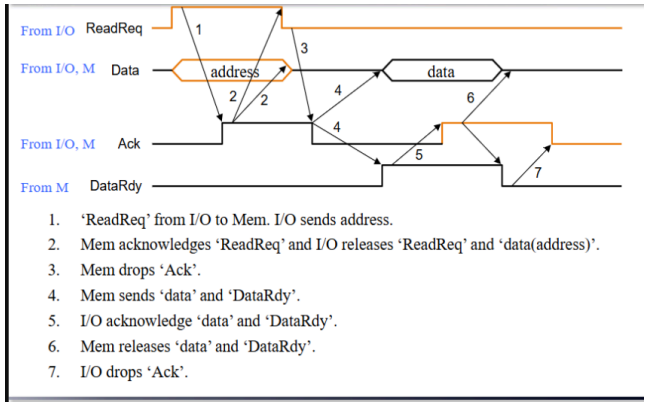
동기 버스 vs. 비동기 버스 (Synchronous vs. Asynchronous Bus)

특징	동기 버스 (Synchronous Bus)	비동기 버스 (Asynchronous Bus)
클럭	제어 라인에 클럭 포함	클럭을 사용하지 않음
통신 프로토콜	클럭에 상대적인 고정 프로토콜	핸드셰이킹(handshaking) 프로토콜 필요
장점	로직이 적어 매우 빠르게 실행 가능	광범위한 장치를 수용 가능; 클럭 스큐(skew) 걱정 없이 길어질 수

특징	동기 버스 (Synchronous Bus)	비동기 버스 (Asynchronous Bus)
		있음
단점	버스의 모든 장치가 동일 클럭 속도로 작동해야 함; 클럭 스큐 방지를 위해 고속 시 길이를 길게 할 수 없음	더 복잡한 제어 로직 필요

- **Synchronous Bus**
  - BReq가 high 시 버스 요청
  - BGrant high시 Master가 버스 제어
  - Master가 Commad, address 버스에 실음
  - Wait=1 일때 준비안됨, Slave가 준비될때까지 Master 멈춤 ⇒ variable latency
  - Wait = 0인 클럭부터 유효, data 전송

- **Asynchronous Bus**



1. 'ReadReq' from I/O to Mem. I/O sends address.
2. Mem acknowledges 'ReadReq' and I/O releases 'ReadReq' and 'data(address)'.
3. Mem drops 'Ack'.
4. Mem sends 'data' and 'DataRdy'.
5. I/O acknowledge 'data' and 'DataRdy'.
6. Mem releases 'data' and 'DataRdy'.
7. I/O drops 'Ack'.

I/O

- Memory Mapped I/O vs. Special I/O Instructions
- Polling vs. Interrupt

- DMA(Direct Memory Access)
- OS(Operating Systems)
  - I/O 하드웨어와 I/O를 요청하는 프로그램 간의 인터페이스 역할

ISA for I/O

- Memory Mapped I/O (in MIPS)
  - loads → input, store → output
  - 주소 공간의 일부를 communication paths to I/O device(통신경로) 에 할당
  - 특정 주소는 일반메모리 주소가 아님. ⇒ I/O 장치의 레지스터에 접근, RAM접근 X

OS & I/O

⇒ OS는 다음 상황을 알아야 함

- I/O device 작업 완료 시
- I/O 작업 오류 발생 시

Polling

- OS가 state register를 주기적으로 확인
- state register에 정보를 둠.
- 장점 : 간단하며 프로세서가 모든 것을 제어(status, information)
- 단점: polling overhead가 많은 CPU time 소비
- Device Register
  - **Control register**
    - read/ write 상태를 나타냄 with Ready Bit(0→1 : 준비 완료)
  - **Data register**
    - 실제 데이터 포함
    - Load or Store into Data register 시 (1→0으로 재설정)

❓ CPU가 I/O가 준비될때 까지 아무일 안하고 확인만 하는 건 낭비(”Spin-wating”)
⇒ Exception mechanism 사용

I/O Interrupt

- I/O 장치가 작업을 완료하거나 주의가 필요할때 Interrupt
- Exception
  - CPU 내부에서 (e.g undefined opcode, overflow, syscall..) 발생
- Interrupt
  - 외부 I/O device 에서 발생
- Interrupt은 Exception와 유사하지만 instruction excution에 동기화 X
- 명령어 사이 handler 호출 가능
- Cause information은 인터럽트를 건 장치 식별

Interrupt Handling Procedure

1. running procedure를 떠나 PC와 같은 status of CPU 저장
2. ISR(Interrupt Service Routine) ⇒ 특정 인터럽트에 대응
3. CPU 상태 restore, running program으로 복귀

Handling Exceptions

- Exception은 System Control Coprocessor(CP0)에 의해 관리 in MIPS
- EPC(Exception Program Counter): 문제가 발생했거나 인터럽트된 명령의 PC 저장
- Cause Register: 문제의 종류 저장(0 - undefined opcode, 1 - overflow)

Vectored Interrupts

- 인터럽트 핸들러이 주소가 cause에 의해 결정
- 인터럽트 처리 명령어는 인터럽트를 직접처리 or 실제 핸들러로 점프

Interrupt Driven Data Transfer

- 프로세스
  1. I/O Interrupt
  2. PC 저장
  3. ISR로 점프
  4. 전송
  5. 복귀(jr)

- 장점: 사용자 프로그램은 actual transfer동안만 중단
- 단점: special hardware 필요
  - Interrupt 발생(I/O device)
  - Interrupt 감지(processor)
  - 상태 저장과 복귀(processor)

인터럽트 기반 I/O 방식에서 대규모 데이터 블록 전송시 processor가 data transfer에 관여해야함

⇒ Direct Memory Access

- CPU 외부에서 작동
- Bus의 Master처럼 작동
- CPU intervention(개입)없이 메모리로 or 메모리에서 data block 전송
- High Bandwidth I/O devices(고대역폭) block 전송에 적합

- DMA Process
  - CPU는 DMAC에 시작주소, 방향, 길이를 전송, “start” 명령
  - DMAC는 주변 장치, 메모리에 handsake signals제공, cpu대신 직접 데이터를 전송