Computer Architecture Lab

Lab 09: DMA

# 1. 실험 목표

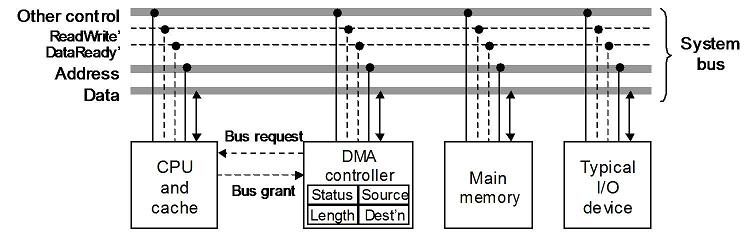
Direct Memory Access(DMA)를 통해 CPU의 효율을 높인다.

# 2. 내용

\* Main Memory, DMA controller와 I/O Device를 모듈로 구현한다.

\* Bus의 종류는 control, address, data bus 3가지가 있다.

- CPU는 address와 data bus를 이용하여 DMA와 통신한다.



\* I/O Device가 CPU에 Interrupt를 보낸다.

- CPU는 항상 interrupt enable 상태라고 생각한다.

\* CPU가 DMA Controller에게 command(status, length, source, dest)를 보낸다.

- status : CPU와 control information을 주고 받기 위한 register. ex) DMA Controller의 작업이 끝났는가

- length : 전송된(될) data word의 수

- source : data source의 address

- dest : data가 전송될 목적지 (각 32bit or 16bit)

\* DMA Controller가 CPU에 Bus Request(BR) 신호를 보낸다.

- BR : DMA Controller가 CPU에게 버스를 release 하도록 요청하는 신호로써, 활성화시 CPU는 현재의 명령실행을 끝내고, address bus, data bus, Read/Write line을 disable한다.

\* CPU의 bus control circuitry가 이를 받아 DMA Controller에게 Bus Grant(BG) 신호를 보낸다.

- CPU는 BG를 통해 bus가 disable 상태임을 표시한다. 즉, BG=1이면 DMA는 RD/WR 제어를 활성화하여 address bus에 직접주소지정을 하고 data bus를 통해 메모리와 직접 통신을 수행한다.

- Bus Request, Grant는 별도의 communication line을 이용한다.

\* DMA Controller가 bus를 이용하여 data를 전송하는 동안 CPU는 해당 bus를 사용하지 않는 작업을 해야한다.

- DMA Controller가 bus를 이용하여 data를 전송하는 동안 CPU는 memory bus를 이용한 data 전송을 할 수 없으므로 CPU는 그 동안 cache를 이용한 작업을 수행한다.

\* DMA Controller는 작업 완료시 BR을 disable하고 interrupt를 발생시킨다.

- interrupt는 interrupt handler function을 구현한 후, 이를 call 하는 방식으로 구현한다.

\* interrupt가 발생하면 CPU는 BG를 disable하고 bus를 정상으로 돌린다.

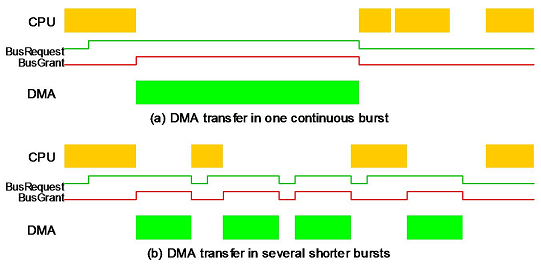
\*Extra point

- Cycle stealing : DMA Controller가 한번에 한 data word를 전송하고 bus를 CPU에게 반환, CPU는 자신의 동작의 지연 없이 한번의 메모리 사이클을 DMA에 할당한다.

- 오랜 시간동안 bus가 DMA controller에 점유되어 CPU의 performance가 떨어지는 것을 방지하기 위하여 DMA Controller가 Data를 한번에 모두 전송하지 않고 4byte씩 나누어서 전송하는 것을 구현한다.

DMA Controller는 data의 한 단위 전송이 끝나면 CPU에게 알리고 일정 delay 후에 다시 bus request를 한다.

Controller의 한 단위 전송이 끝나면 CPU가 bus를 사용할 수 있게 되는데, CPU가 bus를 사용하는 작업이 끝나면 bus grant를 해준다.



# 3. 실험 과정

Test Scenario는 다음과 같다.

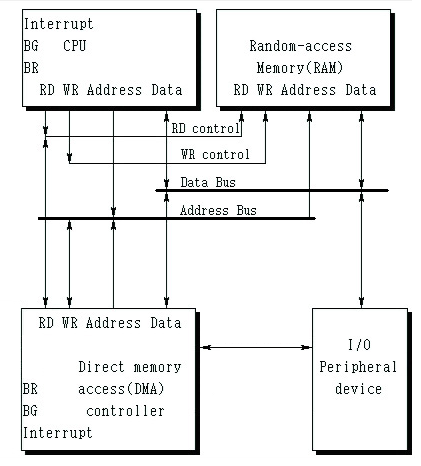
- I/O module의 buffer에는 integer 10개를 둔다.

- DMA Controller가 I/O module의 integer data를 memory space에 옮긴다.

- 옮긴 결과를 보여준다.

- DMA Controller가 data를 옮기는 동안 동시에 CPU가 다른 작업을 하고 있음을 보여준다.

\* 전체적 구조는 아래와 같다.



# 4. 실험 결과

이번 실험은 기본적인 면을 최대한 살려 구현하는 것을 목표로 하였다. DMA가 워낙 어렵고 타이밍을 맞추기 힘든 프로젝트라서 부가 점수보다는 기본적인 동작을 수행하는 것을 최우선 목표로 잡았다. 따라서 Cycle stealing은 구현하지 않았고 데이터가 I/O 버퍼에 있으면 연속적으로 읽어서 메모리에 쓰게 구현하였다.

# 5. 느낀 점

드디어 이번학기 컴퓨터 아키텍쳐 실험의 전 과정이 끝났다. 비록 최후의 9번 DMA 실험을 완벽하게 완수하지 못하여 아쉽지만 그래도 기본적인 요구조건은 구현하여 만족스럽다. 다음 10번 실험은 extra credit이므로 부담 없이 수행할 수 있어서 기분이 좋다. 한 학기동안 미숙한 코드를 채점해주신 조교님께 감사드리며 이만 이번학기 최후의 실험 보고서를 마치고자 한다.