ARM 프로세서의 Cache와 MMU

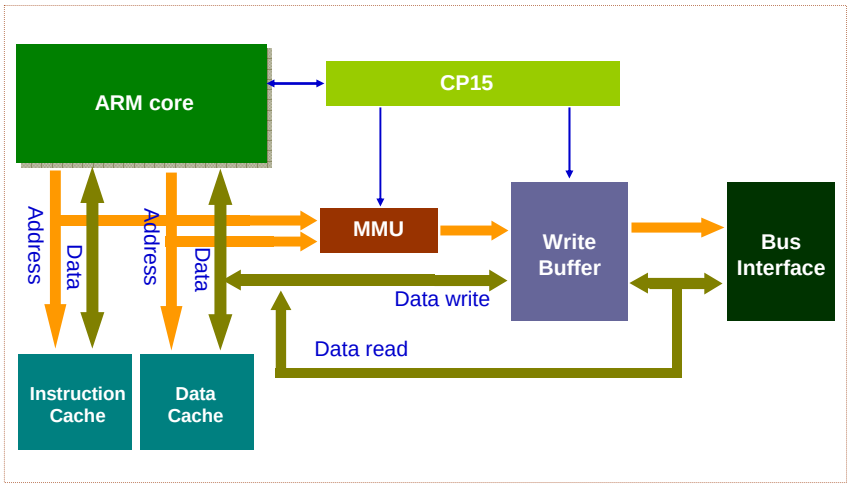
DIGNSYS
www.dignsys.com

교 육 목 표

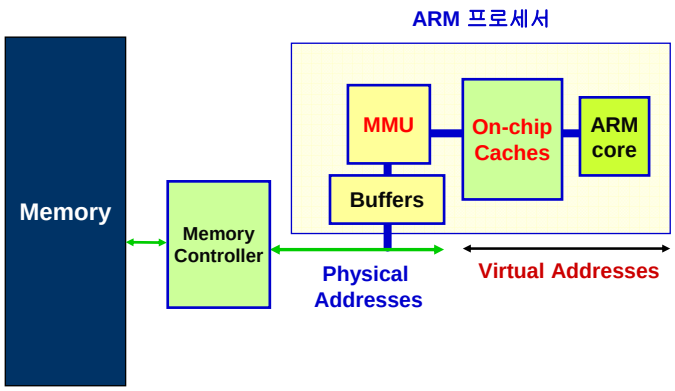
- **Cache와 Write Buffer**에 대하여 이해한다.
- **Memory Management Unit(MMU)**에 대해서 이해한다.
- **Tightly Coupled Memory(TCM)**에 대해서 이해한다.

목 차

Cache를 가지는 ARM 프로세서의 구조



ARM의 메모리 시스템



Cache를 가진 ARM 프로세서의 제어

- **CP15 인터페이스를 통해서 제어**
  - Cache, MPU or MMU, endian 제어 등
- **Coprocessor Register Transfer 명령**
  - MRC : Move to Register from Coprocessor
    - Coprocessor 레지스터 내용을 ARM 레지스터로 전송
  - MCR : Move to Coprocessor from Register
    - ARM 레지스터의 내용을 Coprocessor 레지스터로 전송

MCR/MRC{cond} p15, opcode_1, rd, cn, cm, opcode_2

p15 – coprocessor 15를 나타낸다
opcode_1 – always zero
rd – ARM의 source 또는 destination 레지스터
cn – primary CP15 register
cm – additional register name
opcode_2 – additional information 표시

CP15 레지스터 (ARM920T)

Register	용 도	비 고
0	ID code register	Opcode_2=0
	Cache type register	Opcode_2=1
1	Control Register	Cache, MMU enable, Endian Clock, 제어 등
2	Translation table base register	
3	Domain access control register	
5	Fault status register	
6	Fault address register	
7	Cache operation register	Cache control
8	TLB operation register	
9	Cache lockdown register	
10	TLB lockdown register	
13	FSCE PID register	Fast Context Switching Extension
14	Debug support register	DCC enabled
4, 11, 12	Reserved	

DSA56v10 Cached ARM 프로세서 6

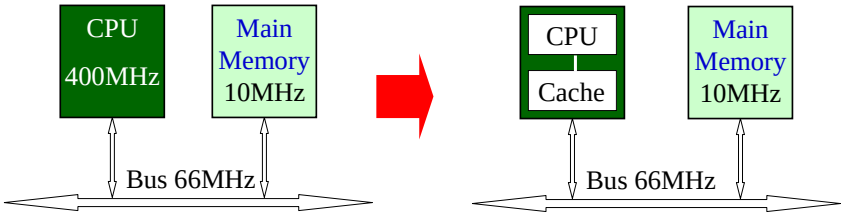
CP15 레지스터 (XScale)

Register	용 도	비 고
0	ID code register	Opcode_2=0
	Cache type register	Opcode_2=1
1	Control Register	Cache, MMU enable, Endian Clock, 제어 등
2	Translation table base register	
3	Domain access control register	
5	Fault status register	
6	Fault address register	
7	Cache operation register	Cache control
8	TLB operation register	
9	Read buffer operation	
10	TLB lockdown register	
13	FSCE PID register	Fast Context Switching Extension
14	Debug support register	
15	Test & Clock control	
11, 11, 12	Reserved	

DSA56v10 Cached ARM 프로세서 7

목 차

Cache 메모리

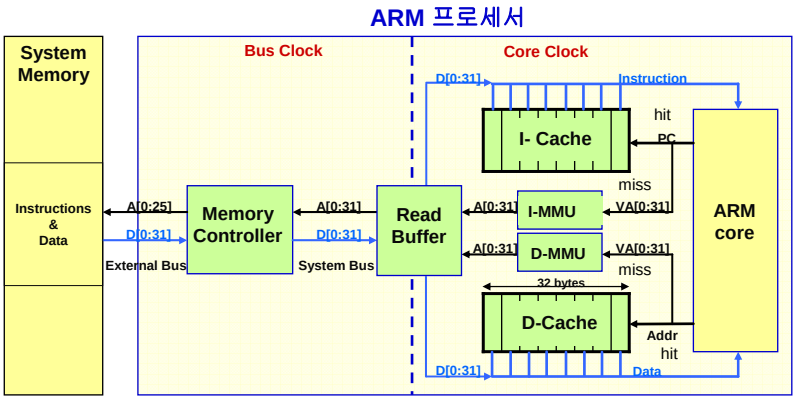


고속의 CPU가 버스 및 메모리 속도에 의존적이며 늦다

CPU 주변에 고속의 메모리를 두고 자주 사용되는 명령과 데이터를 저장하여 시스템 성능을 개선

- **Cache 메모리**
  - 프로세서가 최근에 억세스 한 메모리의 내용을 보관하고 있다가 다시 요청하면 메모리 억세스 없이 전달
- **재 사용되는 메모리 공간에 대한 억세스가 없다**
  - 속도가 느린 메모리 시스템의 성능개선
  - 버스의 사용량을 줄여 시스템 성능 향상
  - 전력 소모를 줄일 수 있다

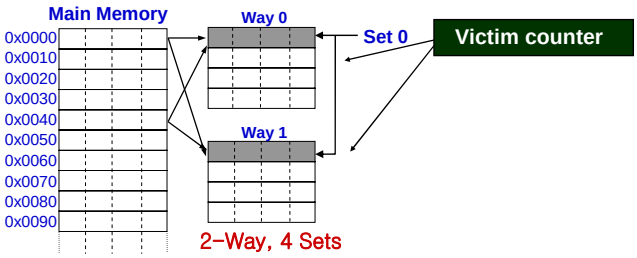
Cache와 메모리 Read



Cache의 성능

- 프로세서가 읽고자 하는 명령이나 데이터가 **Cache** 내에 존재(**Cache Hit**) 하는 회수가 많아야 **Cache**의 성능이 우수하다.
- **CPU**가 데이터나 명령을 읽고자 하는데 **Cache** 내에 원하는 명령이나 데이터가 없으면(**Cache Miss**), **Cache** 제어기는 시스템 메모리 장치에서 **line** 크기 만큼 명령이나 데이터를 읽어 **Cache** 메모리에 저장(**Line Fill**) 한다.
  - Cache Line : Cache가 관리하는 최소한의 데이터 단위
    - 4 word, 8 word 정도의 크기를 가진다.
  - Cache miss 가 많으면 시스템 성능이 떨어 진다.

Set Associative 방식



Victim counter : 어느 way에 replace 할지 결정

- Random
- cyclic*(round robbin)
- Least Recently Used(LRU) : 사용 않 됨

Cache Lockdown

- **Cache의 일정 부분을 update 되지 않도록 한다.**
  - 중요한 명령이나 데이터를 항상 Cache에 있도록 하여 성능 증가 및 보장
- **Victim counter가 지정한 위치에 가지 않도록 하여 가능**
  - ARM core의 경우 CP15의 register 9번으로 제어
  - Lockdown 예

```
MOV   r2, #lockdown_base<<26    ; victim pointer 지정
MCR   p15, 0, r2, c9, c0, 1     ; write Icache victim and lockdown base
```

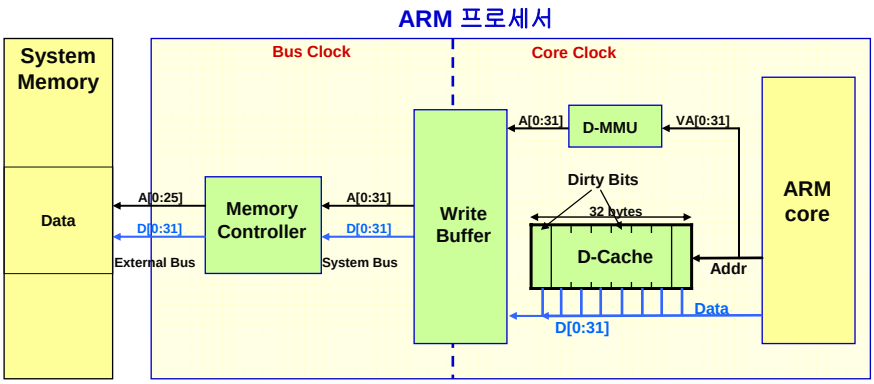
- **Cache flush 전에는 반듯이 Lockdown을 해제 하여야 한다.**

Cache Flush

- Cache의 내용과 메모리의 내용이 다른 경우 Cache를 비우고 새로 데이터를 메모리에서 읽어와야 한다.
  - Self modifying code가 사용된 경우
  - MPU나 MMU의 값이 변경된 경우
- Cache Flush 및 Invalidate는 CP15 레지스터에 의해서 한다.

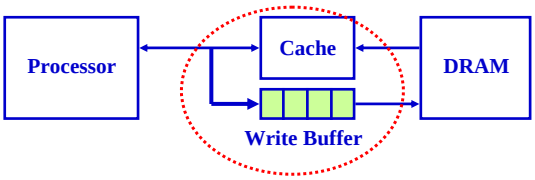
```
MCR     p15, 0, r0, c7, c7, 0        ; Invalidate ID cache
```


Cache와 메모리 Write



Write Buffer

- **Core와 버스의 서로 다른 속도차 극복**
  - Core speed로 write buffer에 데이터를 write
    - Write 후 CPU는 다른 작업 처리 가능
  - Bus speed로 write buffer의 내용이 메모리로 write
  - Write buffer는 FIFO 형태의 구조를 가진다



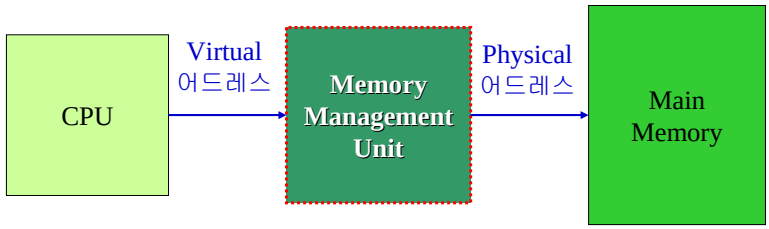
Write Buffer와 Cache

- **Write Through**
  - CPU가 특정 주소에 명령이나 데이터를 write하는 경우, 해당하는 명령이나 데이터가 Cache 메모리에 있을 때, Cache 메모리와 외부 메모리에 모두 쓰기 동작을 한다.
  - Write buffer를 거치지 않고 메모리에 저장
- **Write Back**
  - CPU가 특정 주소에 명령이나 데이터를 write하는 경우, 해당하는 명령이나 데이터가 Cache 메모리에 있을 때, Cache 메모리에만 쓰기 동작을 하고, 외부의 메모리에는 나중에 기록 된다.
  - Write buffer를 사용한다.

목 차

MMU (Memory Management Units)

- ❑ 메모리 보호(protection) 기능
- ❑ 어드레스 변환(translation) 기능
  - ❖ CPU에서 사용되는 logical 한 Virtual 어드레스를 physical 어드레스로 변환



MMU의 필요성

- **Dynamic 한 메모리 관리**
  - 프로그램 실행 중에 수시로 access permission 관리, cacheable / bufferable 특성의 변환 가능
- **Virtual address를 지원한다.**

MMU의 구성

- **Translation Lookaside Buffer (TLB)**
  - 최근에 사용된 Virtual address를 physical address로 변화하는 정보와 access permission에 대한 정보를 저장하고 있는 일종의 Cache
- **Translation Table Walking Logic**
  - TLB를 update 하고 관리하는 기능을 가진 logic
- **Access Control Logic**

Translation Table

- **Physical 메모리에 있는 translation 정보를 가지고 있는 Table**
- **Level 1 Translation Table**
  - 4096개의 32비트 translation table entry
    - 4GB 메모리를 virtual address 1MB 단위로 나누어 관리
    - Virtual address 비트 [31:20]로 정렬
  - Physical memory에 대한 1MB section 단위의 address translation 정보와 access control 정보를 가지거나, 레벨 2 table에 대한 주소 정보를 가진다.
- **Level 2 Translation Table**
  - 64KB(large page), 4KB(small page), 1KB(tiny page) 단위의 translation table을 정보를 가지고 있다.
  - 각 translation table에는 address translation 정보와 access control 정보를 가진다.

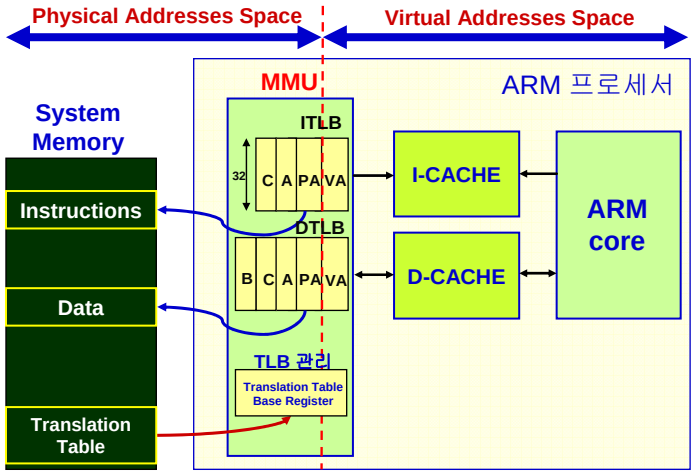
DSA56v10 Cached ARM 프로세서 22

Translation Lookaside Buffer (TLB)

- **최근에 사용된 Virtual address를 physical address로 변화하는 정보와 access permission에 대한 정보를 저장하고 있는 일종의 Cache**
- **TLB가 Virtual 어드레스에 대한 translation table entry를 가지고 있으면 access control logic이 access 가능을 판단**
  - 접근이 허용되면 virtual address를 physical address로 변환 후 access
  - 접근의 허용이 않 되면 CPU에 Abort 구동
- **TLB에 virtual 어드레스에 대한 정보가 없으면 translation table walking logic에서 table 정보를 physical 메모리에서 읽어 TLB update**

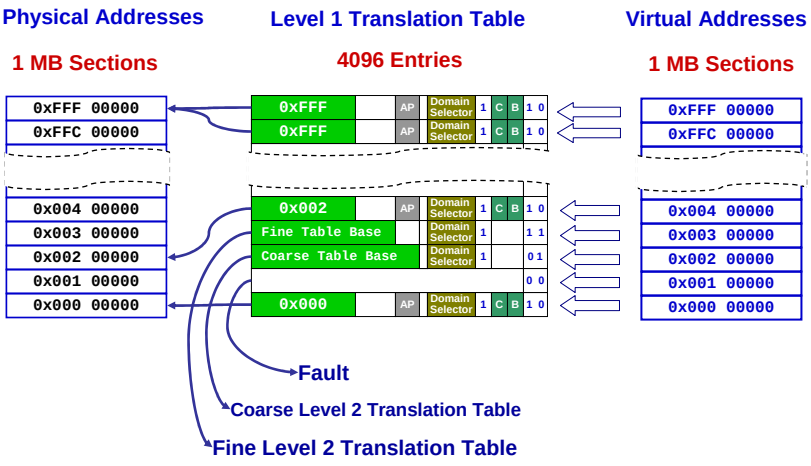
DSA56v10 Cached ARM 프로세서 23

MMU와 메모리 구성



DSA56v10 Cached ARM 프로세서 24

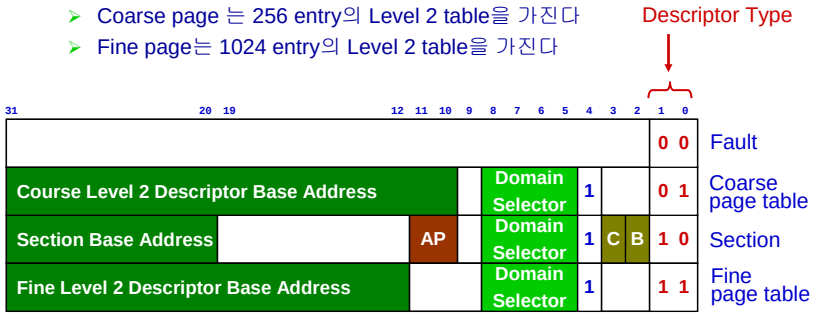
MMU의 어드레스 변환



DSA56v10 Cached ARM 프로세서 25

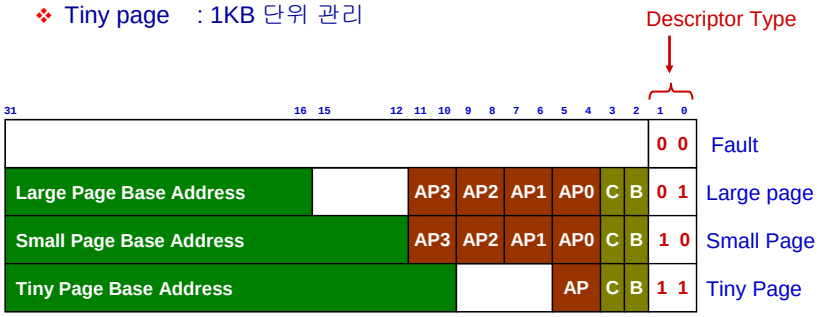
Level 1 Descriptor

- **Level 1 Descriptor의 종류**
  - Fault
  - Section descriptor
  - Coarse or fine page descriptor
    - Coarse page 는 256 entry의 Level 2 table을 가진다
    - Fine page는 1024 entry의 Level 2 table을 가진다



Level 2 Descriptor

- **Level 2 Descriptor의 종류**
  - Fault
  - Large page : 64KB 단위 관리
  - Small page : 4KB 단위 관리
  - Tiny page : 1KB 단위 관리



Cache와 Write Buffer 제어

- **Section 또는 page 별로 Cache와 Write Buffer의 사용여부 결정**
  - Cacheable
    - Page 내의 데이터가 Cache될 수 있음을 나타낸다
  - Bufferable
    - Page 내의 데이터가 write buffer에 write될 수 있음을 나타낸다.
  - Memory mapped I/O 장치의 경우에는 Cache와 Write Buffer가 반드시 disable 되어 있어야 한다.
- **Cacheable 과 Bufferable에 의한 메모리 시스템 특징**

C	B	의 미	Cache의 Write 동작
0	0	Cache 불가, 쓰기 버퍼 불가	
0	1	Cache 불가, 쓰기 버퍼 동작	
1	0	Cache 동작, 쓰기 버퍼 불가	Write-through Cache
1	1	Cache 동작, 쓰기 버퍼 동작	Write-back Cache

Access Permission

- **Section 또는 page 별로 메모리의 access permission 제한**
  - Access 권한은 각 descriptor의 AP 정보와 S(System) 비트와 R(Rom) 비트에 의해서 제어
    - S, R비트는 control 레지스터의 비트 8과 9이다
  - Access가 불가하면 permission fault가 발생

AP	S	R	Access Permission	
			Supervisor	User
00	0	0	No access	No access
00	1	0	Read only	No access
00	0	1	Read only	Read only
00	1	1	Reserved	
01	X	X	Read / Write	No Access
10	X	X	Read / Write	Read only
11	X	X	Read / Write	Read / Write
XX	1	1	Reserved	

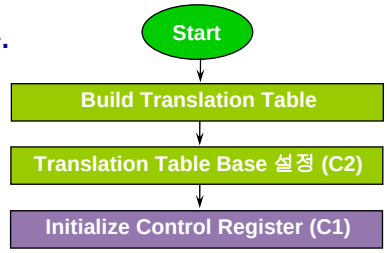
Domain Control

- **MMU의 Access는 기본적으로 "DOMAIN"의 의해서 제어 된다.**
  - 개별적인 Access permission을 갖도록 제어 하는데 사용
  - 16개까지의 domain 지정 가능
    - DACR(Domain Access Control Register)는 각 domain 별로 2비트씩 할당
- **Domain 과 Access Permission**

Domain	의 미	설 명
00	No Access	모든 access에 대하여 domain fault 발생
01	Client	Page Table의 Section descriptor 나 page descriptor의 AP(Access Permission) 비트 정보를 따른다.
10	Reserved	Reserved (No Access)
11	Manager	Page Table의 Section descriptor 나 page descriptor의 AP(Access Permission) 비트 정보를 무시하고 무조건 access를 허용한다.

MMU 설정

- **MMU를 설정하는 과정은 다음과 같다.**
  1. Build Translation Table
     - Physical 메모리에 설정
     - Virtual to physical translation 정보
     - Cacheable / Bufferable 정보
     - Access Permission
  2. Translation Table Base 설정
     - CP15의 c2 사용
  3. Initialize Control Register
     - Enable Cache, MMU
     - Clocking 모드 설정
     - Endian 모드 설정
     - Vector location 설정




```
MOV   r0,  =TTBase          ; Translation table base
MCR   p15, 0, r0, c2, c0, 0 ; set TT base
```

```
MRC   p15, 0, r0, c1, c0, 0 ; read control register
ORR   r0, r0, #0x1
MCR   p15, 0, r0, c1, c0, 0 ; enable MMU
```

목 차

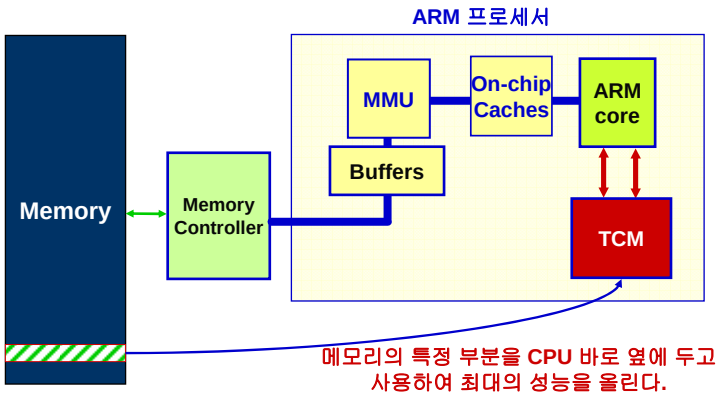
Cache를 가진 ARM 프로세서

Cache와 Write Buffer

Memory Management Unit (MMU)

- Tightly Coupled Memory (TCM)

Tightly Coupled Memory (TCM)



•Cache는 계속 update 되므로, 성능을 보장해야 하는데 필요한 명령이나 데이터가 항상 Cache에 있다고 보장 할 수 없다.

TCM 초기화

- **Programmer's Model**
  - CP15의 control register (c1)를 사용한다.
- **Instruction/Data TCM의 초기화**
  - ① Enable TCM / Load mode
  - ② TCM에서 사용 될 명령 또는 데이터를 TCM 메모리에 저장

```
MOV     r0, #0              ; Initialize pointer
LDR     r1, =ImageTop       ; define end of code image
MRC     p15, 0, t2, c1, c0, 0   ; read control register
ORR     r2, r2, #&C0000
MCR     p15, 0, r2, c1, c0, 0   ; enable Instruction TCM and Load Mode
CopyLoop
LDMIA   r0, {r2 – r9}       ; load 8 registers from main memory
STMIA   r0!, Pr2 – r9}      ; store 8 registers into instruction TCM
CMP     r1, r0              ; cjheck if limit reached
BGT     CopyLoop            ; Repeat if more to do
```


질의 응답