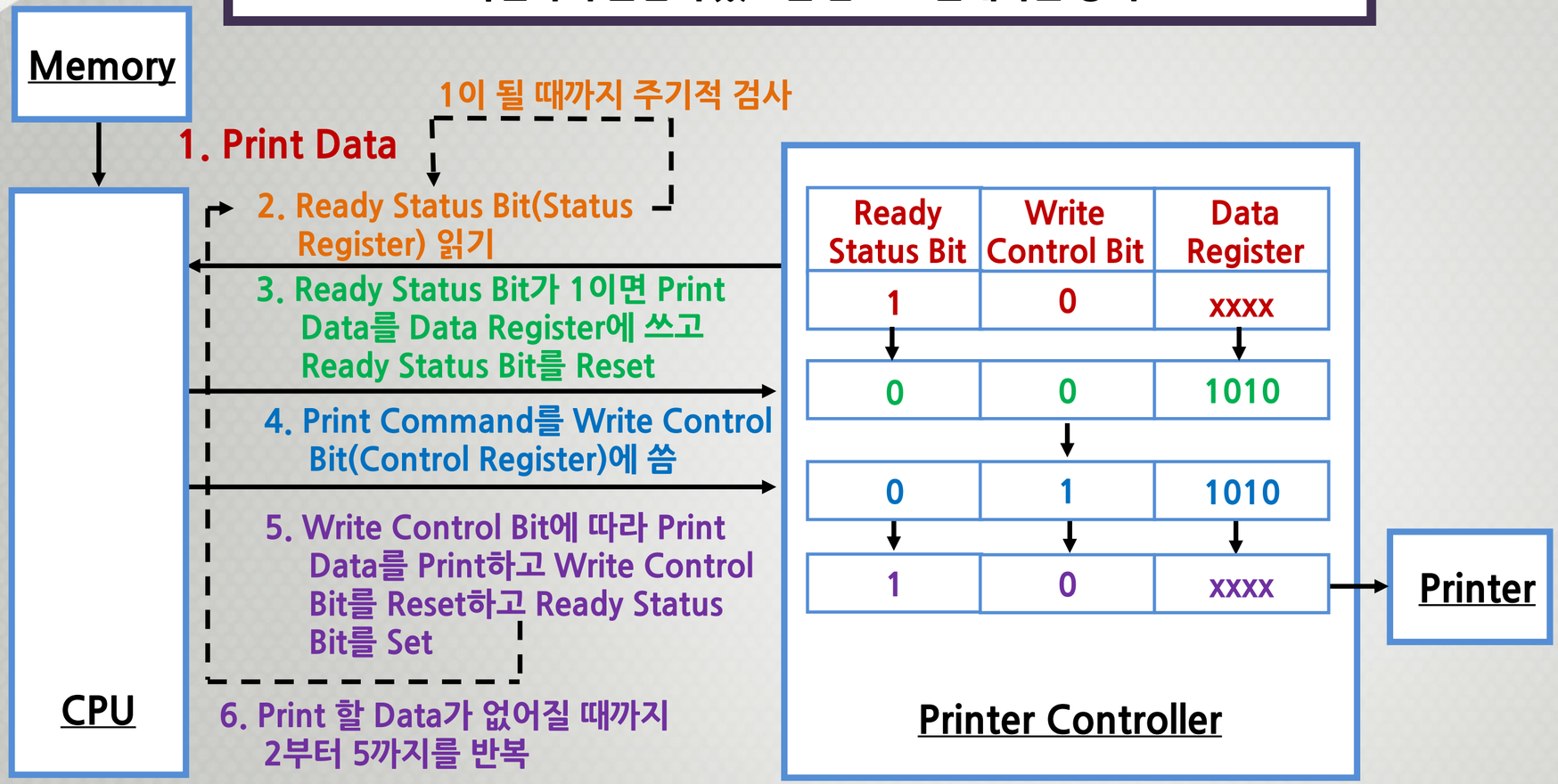
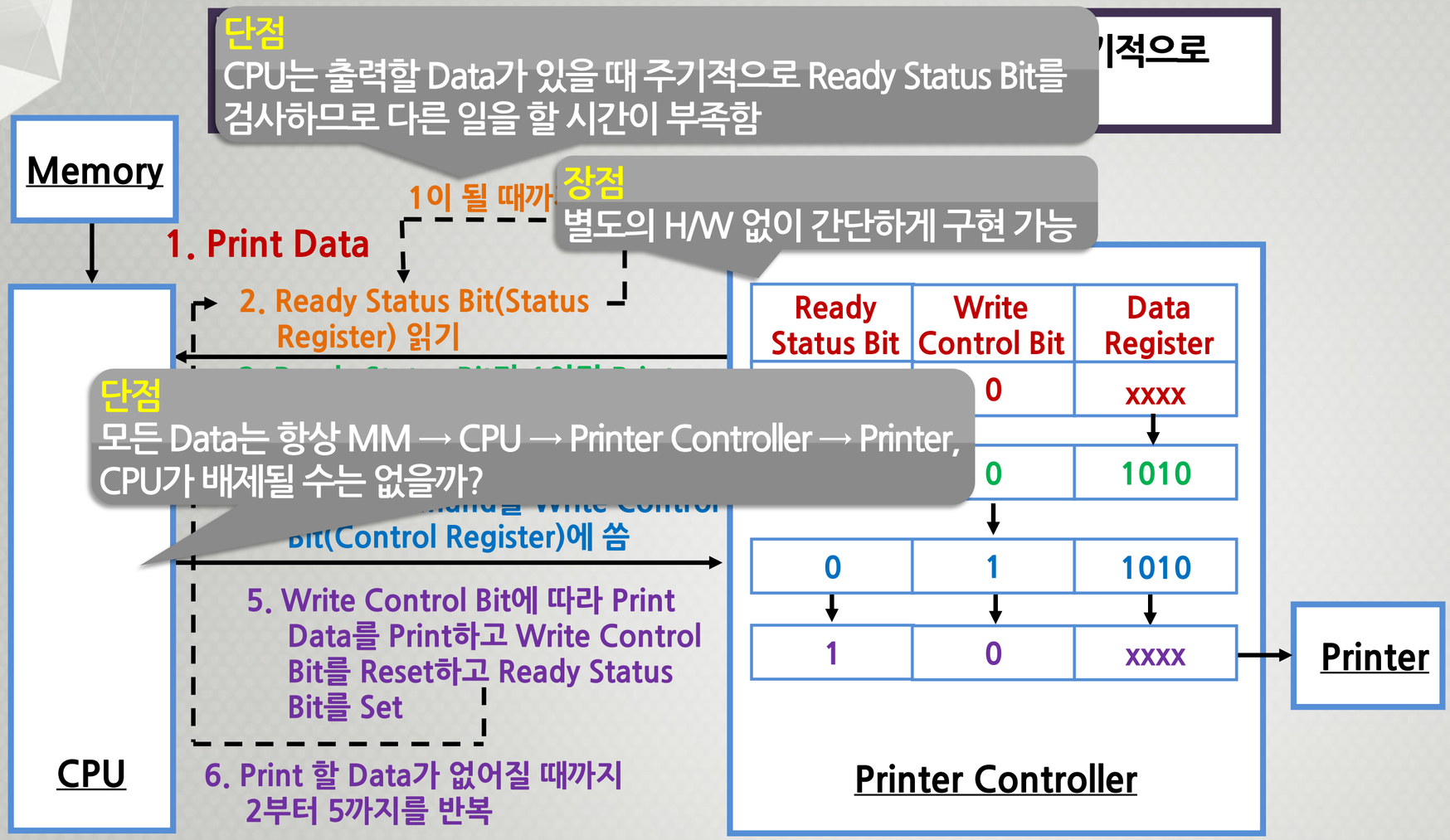
Programmed I/O(=Polling)의 동작 원리: CPU가 Printer로 Data를 출력하는 과정을 중심으로

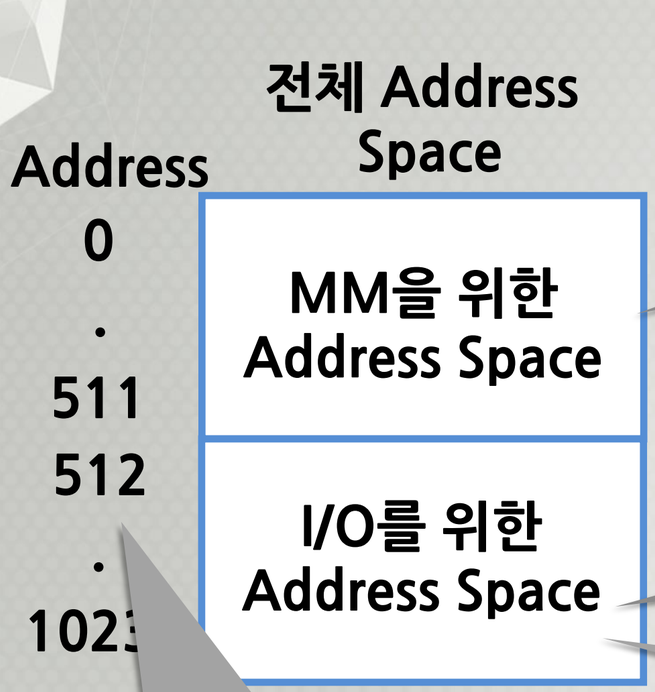
집안에서 주인이 외부에 손님이 왔는지 대문을 10분마다 주기적으로 확인하여 손님이 왔으면 집으로 안내하는 방식



Programmed I/O(=Polling)의 동작 원리: CPU가 Printer로 Data를 출력하는 과정을 중심으로



Programmed I/O에서 Data Register와 Status/Control Register를 어떻게 Addressing할 것인가? - Memory-Mapped I/O



Address Bit: 10 Bits →기억장소의 수: 1024
상위 512 Address : MM에 할당
하위 512 Address : I/O장치들에 할당

Data Register Address(Printer) : 512 번지
Status/Control Register Address(Printer) : 513 번지
→ b0: READY Status Bits, b7: Write Control Bits

MM Read/Write 신호 = I/O Read/Write 신호

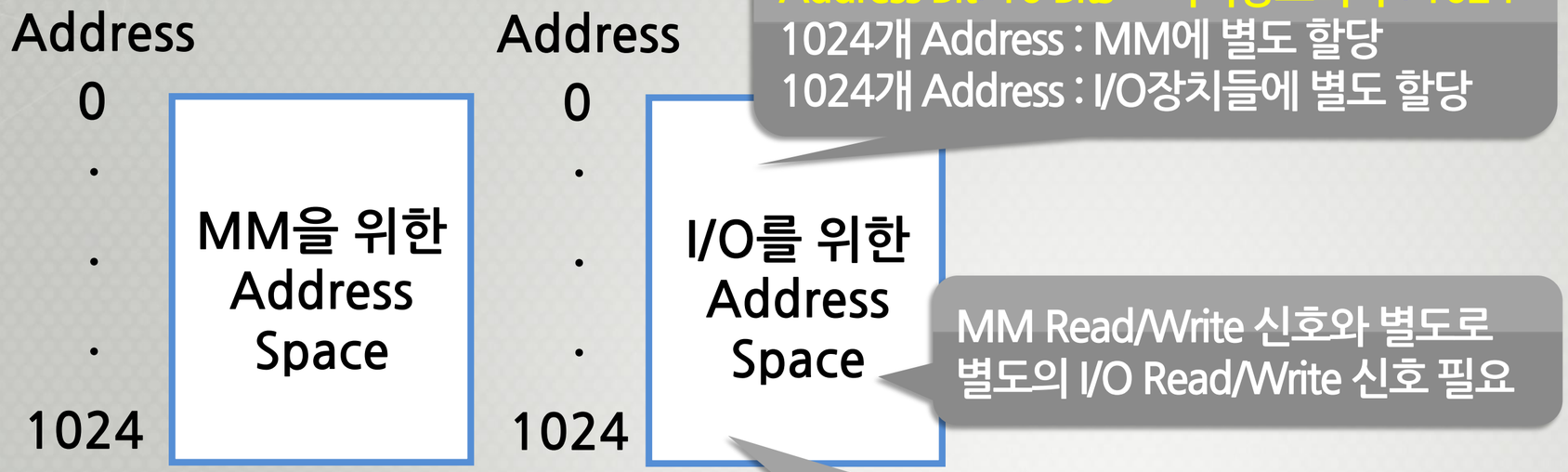
MM Address 영역의 일부를 I/O Controller 내 Register들의 Address로 할당하는 방식
→ MM Address 공간이 감소

Programmed I/O에서 Data Register와 Status/Control Register를 어떻게 Addressing할 것인가? - Memory-Mapped I/O

Memory-Mapped I/O Program	
TEST: LOAD 513;	Status/Control Register의 내용을 읽는다.
ANI 01H;	READY Status Bit를 제외한 모든 Bit들을 0으로 Clear한다. (8bit Register)
JZ TEST;	만일 READY Status Bit가 0이라면 TEST로 Jump한다.
LOAD 100;	Print 할 Data를 MM로 부터 읽어온다.
STOR 512;	Print 할 Data를 Data Register에 쓴다.
LOAD 80H;	AC에 Binary 10000000을 Load한다.
STOR 513;	Write Control Bit를 세팅한다.

- 프로그래밍에서 MM 관련 Instruction들을 I/O 장치 Control에도 사용 가능 → 프로그래밍 용이

Programmed I/O에서 Data Register와 Status/Control Register를 어떻게 Addressing할 것인가? - Isolated I/O



Address Bit: 10 Bits →기억장소의 수: 1024
1024개 Address : MM에 별도 할당
1024개 Address : I/O장치들에 별도 할당

MM Read/Write 신호와 별도로
별도의 I/O Read/Write 신호 필요

MM Address 영역과는 별도로
I/O Controller내 Register들의
Address 공간을 할당하는 방식
→ MM Address공간이
줄어들지 않음

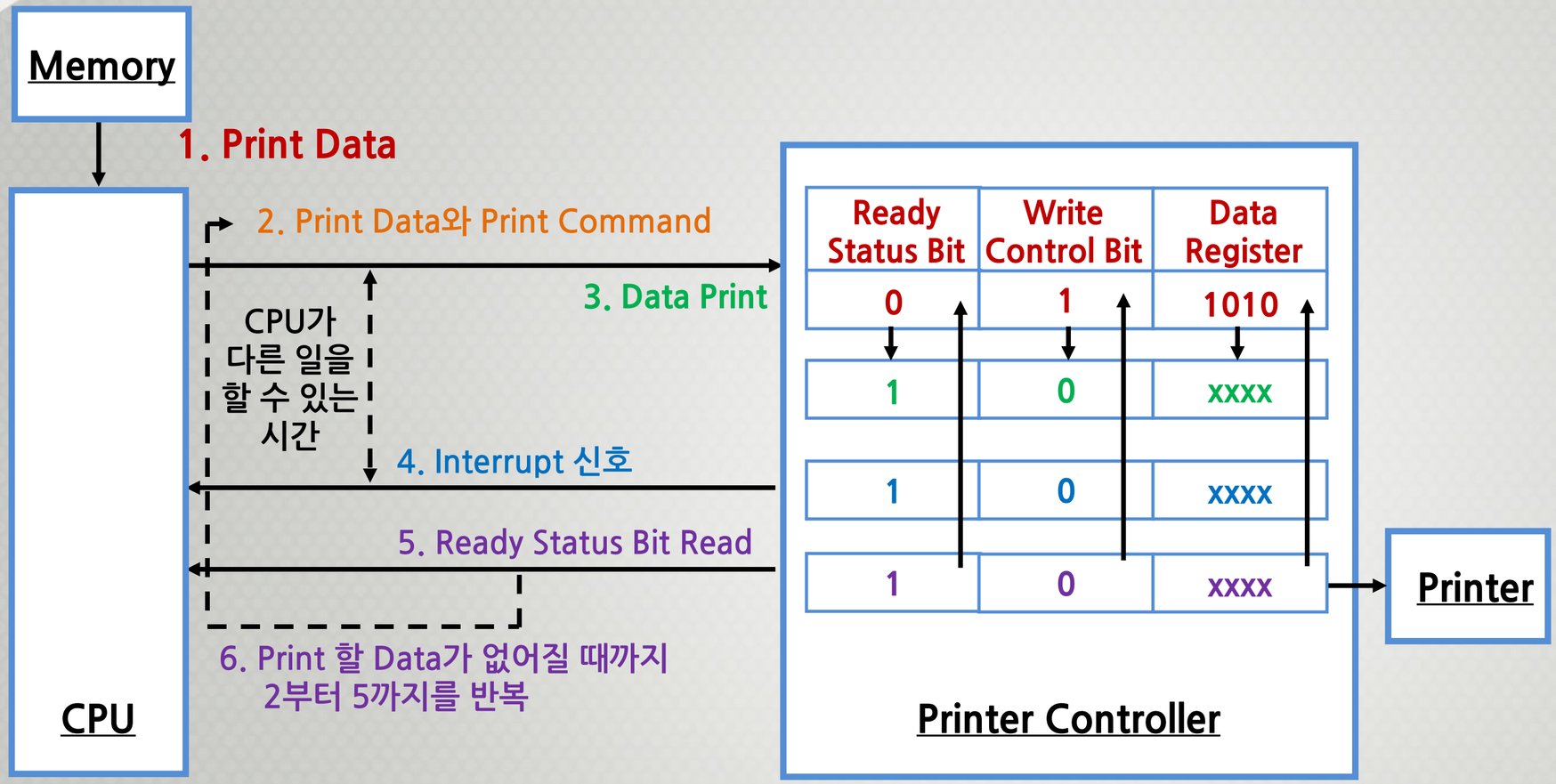
I/O 장치의 Data Register Address : 0 번지
Status/Control Register Address : 1 번지
→ b0: READY Status Bits, b7: WRITE Control Bits

Programmed I/O에서 Data Register와 Status/Control Register를 어떻게 Addressing할 것인가? - Isolated I/O

Isolated I/O Program	
TEST: IN 1;	Status/Control Register의 내용을 읽는다.
ANI 01H;	READY Status Bit를 제외한 모든 Bit들을 0으로 Clear한다.
JZ TEST;	만일 READY Status Bit가 0이라면 TEST로 Jump한다.
LOAD 100;	Print 할 Data를 MM로 부터 읽어온다.
OUT 0;	Print 할 Data를 Data Register에 쓴다.
LOAD 80H;	AC에 Binary 10000000을 Load한다.
OUT 1;	Write Control Bit를 Setting한다.

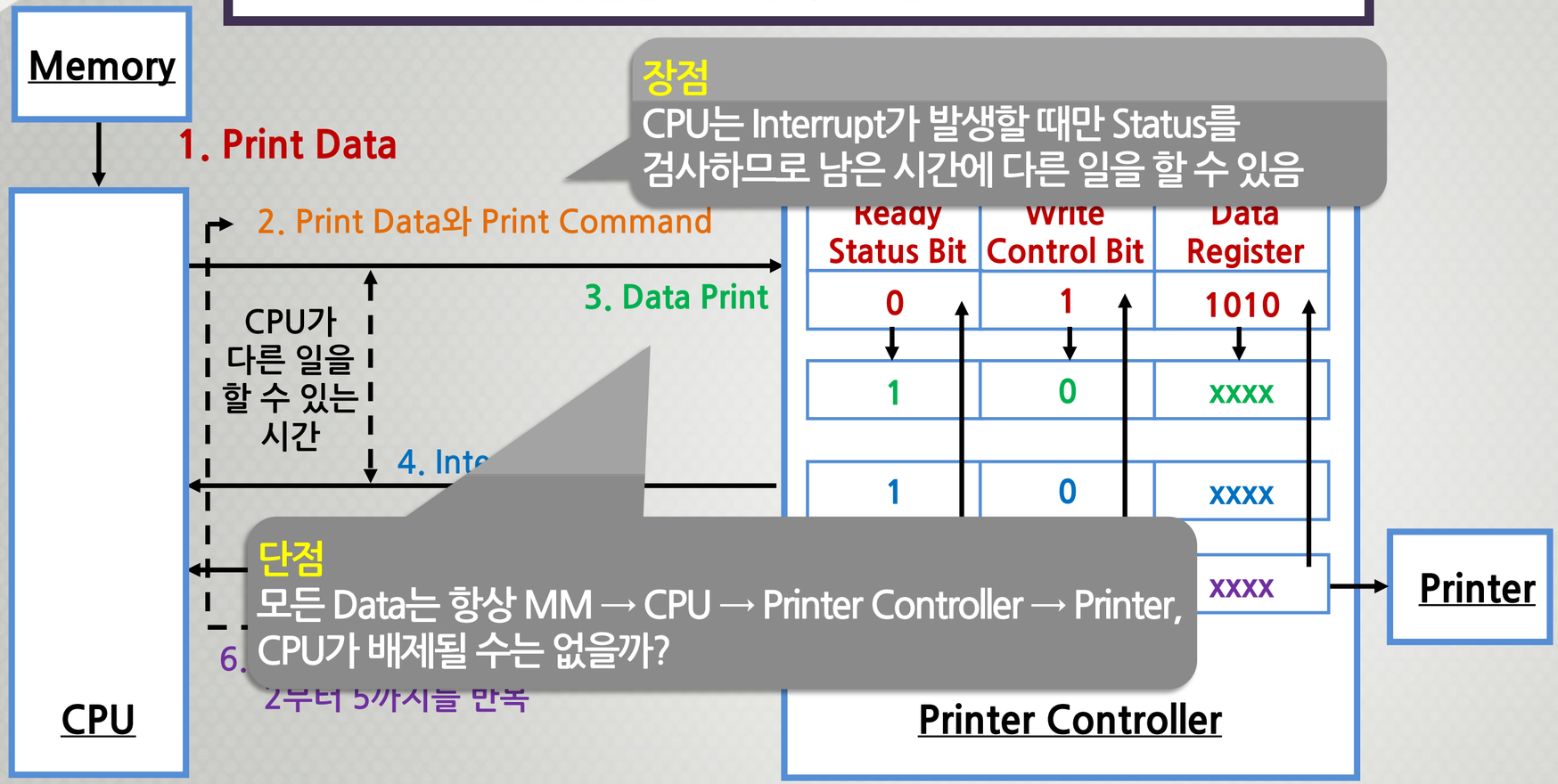
- I/O 장치 Control를 위해서 별도의 I/O Instruction 사용
  → 프로그래밍 불편

Interrupt Driven I/O의 동작 원리: CPU가 Printer로 Data를 출력하는 과정을 중심으로

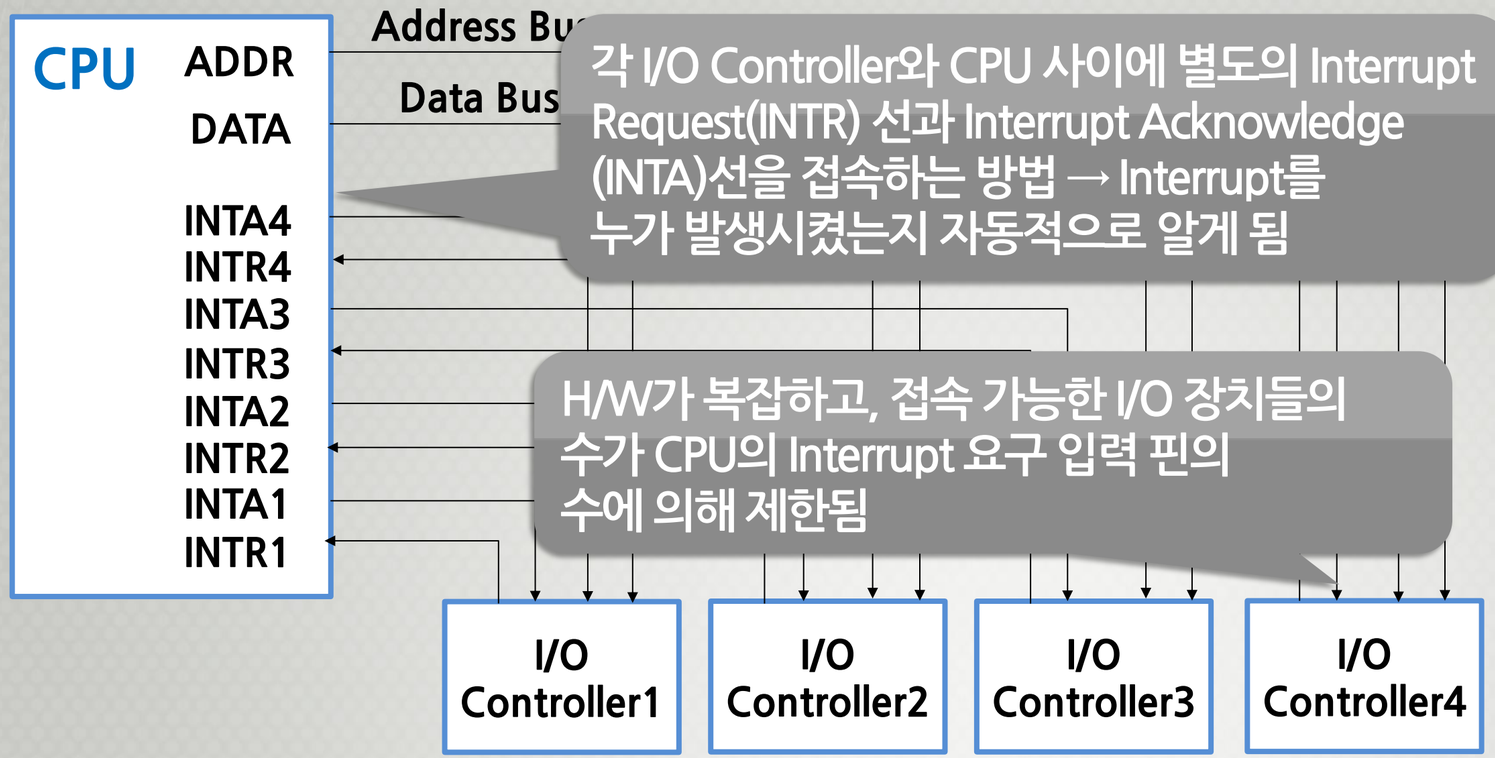


Interrupt Driven I/O의 동작 원리: CPU가 Printer로 Data를 출력하는 과정을 중심으로

집안에서 주인이 외부에 손님이 왔는지 대문에 설치한 초인종이 울릴 때만 확인, 손님을 집안으로 안내

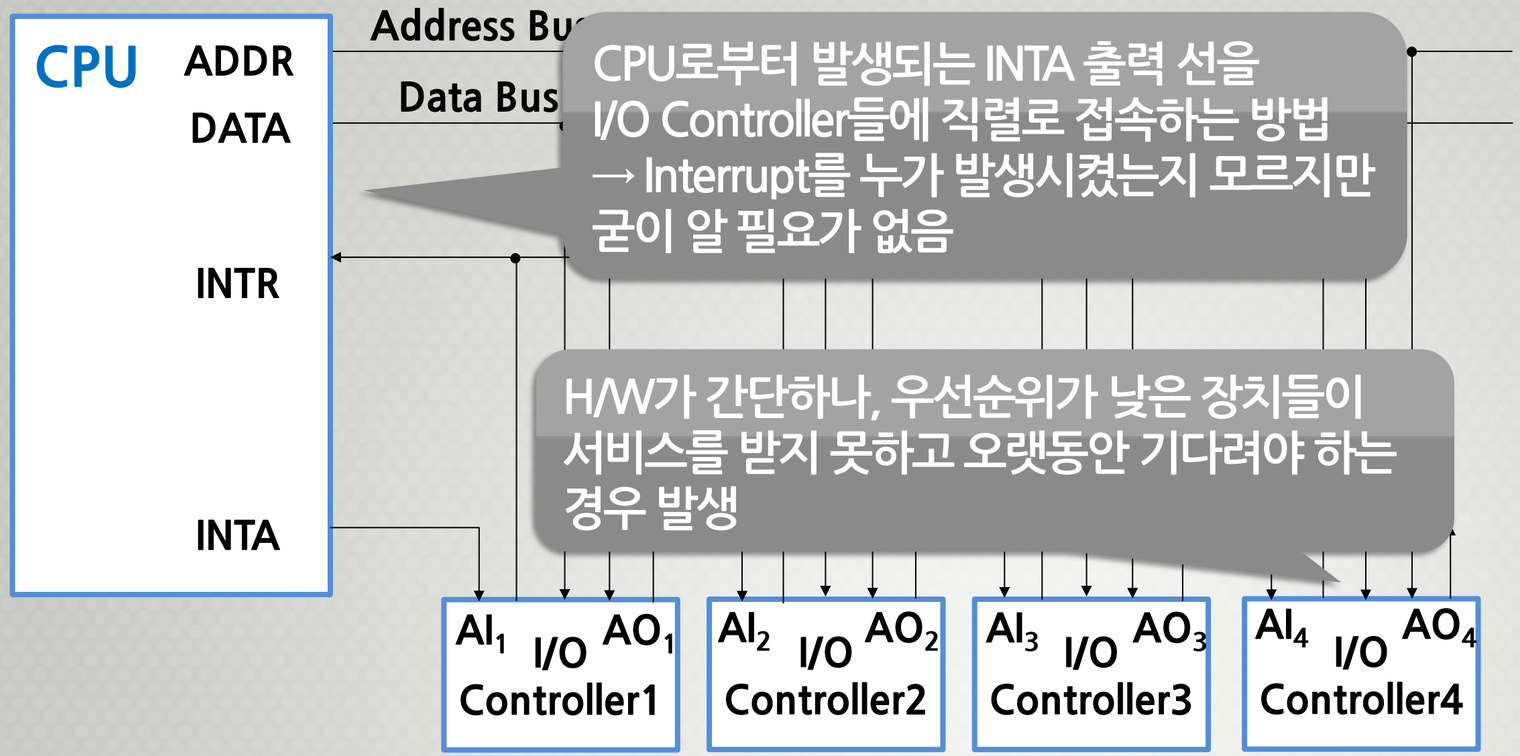


초인종을 어떻게 구현할까? 병렬 구현 방식 → Multiple Interrupt Line



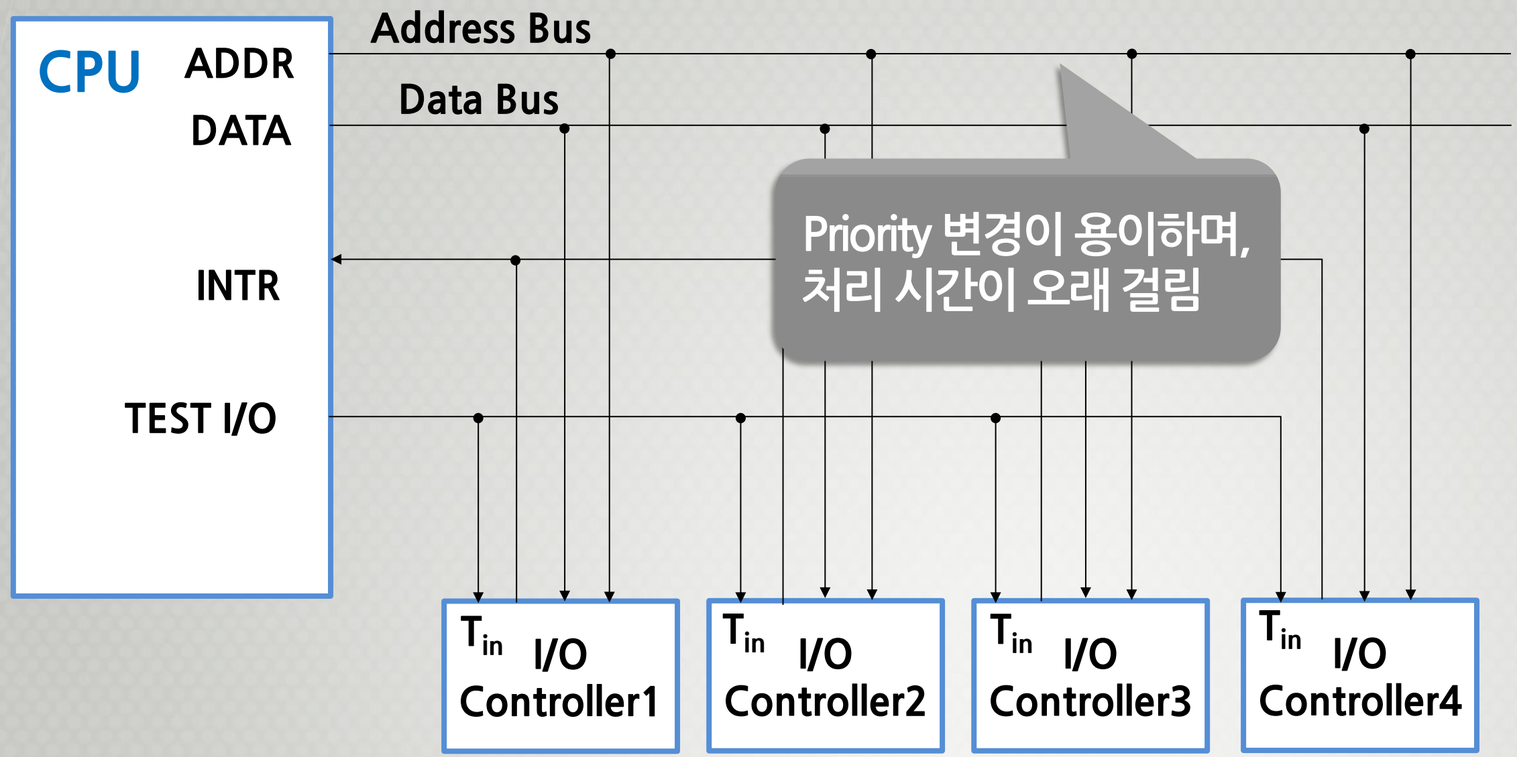
- I/O Controller2가 INTR2 신호를 Set → CPU는 INTA2 신호를 Set함으로써 그 Controller에게 Interrupt 요구를 인식하였음을 알리고, Interrupt를 위한 서비스를 시작 → I/O Controller2는 INTR2 신호를 해제(0으로 Reset) → CPU도 INTA2 신호를 해제

초인종을 어떻게 구현할까? 직렬 구현 방식 = Daisy-Chain 방식



- Interrupt를 요구한 I/O 장치는 AI_n 입력을 받는 즉시 자신의 고유(ID) 번호, 즉 Interrupt vector를 Data Bus를 통하여 CPU로 전송
  (참고: Interrupt 벡터는 해당 I/O 장치를 위한 Interrupt Service Routine의 시작 Address)

초인종을 어떻게 구현할까? S/W Polling 기반 구현 방식



CPU가 모든 I/O Controller들에 접속된 TEST I/O 선을 이용하여 Interrupt를 요구한 장치를 검사하는 방식
→ CPU가 Interrupt 발생주체를 직접 찾음