Slide 0

4장. 레지스터 전송과 마이크로 연산

레지스터 전송언어

Slide 1

- 마이크로 연산 (micro-operation)
  - 레지스터에 저장된 데이타를 가지고 실행되는 동작
  - 하나의 클럭 펄스 동안에 실행되는 기본적인 동작
  - 예) shift, count, clear, load 등
- 디지탈 컴퓨터의 구조를 정의 하기위한 정보들
  1. 레지스터의 종류와 그 기능
  2. 레지스터에 저장된 이진 정보를 가지고 수행되는 마이크로 연산들
  3. 일련의 마이크로 연산을 시작시키는 제어기능
- 레지스터 전송언어
  - 레지스터간의 마이크로 연산을 기호로 간단 명료하게 표현
  - 디지탈 컴퓨터의 마이크로 동작 기술에 이용
  - 디지탈 시스템 설계에 이용

레지스터 전송

Slide 2

- 레지스터의 표시 : 기능을 나타내기 위하여 머릿글자 이용
- 예)
  - MAR (memory address register)
  - PC (program counter)
  - IR (instruction register) 등
- 레지스터의 블럭도 (그림 4-1)

레지스터 전송 (계속)

Slide 3

74 — 제4장 레지스터 전송과 마이크로 연산

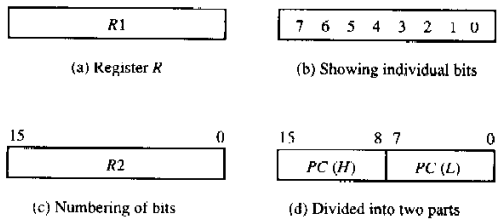


그림 4-1 레지스터의 블럭도

서 있다. 레지스터를 나타내는 가장 일반적인 방법은 그림 4-1(a)에서와 같이

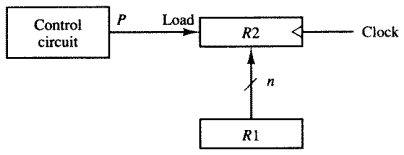
레지스터 전송 (계속)

Slide 4

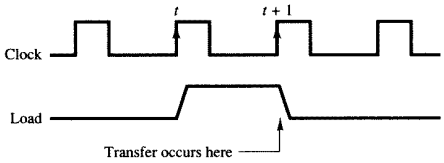
- 레지스터 사이의 정보 전송
  - 치환 연산자 사용 표시
  - 예)
    - R2 ←R1
    - R1 의 내용을 R2 로 옮기라는 의미
    - R2 는 병렬로드 기능이 있어야함
  - 조건이 있을때의 표시
  - 예)
    - if(P=1) then (R2 ←R1)
    - P 가 1 일때 R1 의 내용을 R2 로 옮김을 의미
    - P 는 0 과 1을 갖는 부울 변수
    - 다른 표현 →P: R2 ←R1
  - 하드웨어 블럭도 (그림 4-2)

레지스터 전송 (계속)

Slide 5



(a) Block diagram



(b) Timing diagram

그림 4-2 $P = 1$ 일 때 $R1$ 으로부터 $R2$ 로의 전송

  - 레지스터 전송을 나타내는 기호들 (표 4-1)

레지스터 전송 (계속)

Slide 6

76 제4장 레지스터 전송과 마이크로 연산

표 4-1 레지스터 전송의 기본 기호

Symbol	Description	Examples
Letters (and numerals)	Denotes a register	*MAR*, *R2*
Parentheses ( )	Denotes a part of a register	*R2*(0–7), *R2*(*L*)
Arrow ←	Denotes transfer of information	*R2* ← *R1*
Comma ,	Separates two microoperations	*R2* ← *R1*, *R1* ← *R2*

4.3 버스와 메모리 전송

버스와 메모리 전송

Slide 7

- 버스란
  - 모든 레지스터 사이의 정보전송이 독립적인 라인을 이용하면 너무 많아짐
  - 여러 레지스터 사이의 정보전송을 위한 공통의 전송라인
- 멀티플렉서를 이용한 버스의 구성 (그림 4-3)

버스와 메모리 전송 (계속)

Slide 8

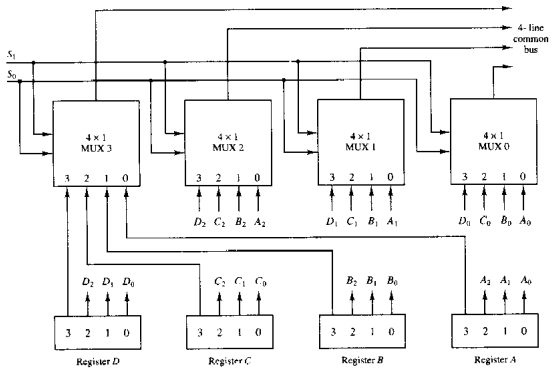


그림 4-3 네 개의 레지스터에 대한 버스 시스템

표 4-2 그림 4-3의 버스에 대한 함수표

버스와 메모리 전송 (계속)

- S1, S0 를 이용하여 버스에 실을 레지스터 선택
- 예) S1S0 = 00 이면 레지스터 A 의 정보가 버스에 실림
- 버스에 대한 함수표 (표 4-2)

Slide 9

표 4-2 그림 4-3의 버스에 대한 함수표

S_1	S_0	Register selected
0	0	A
0	1	B
1	0	C
1	1	D

버스와 메모리 전송 (계속)

Slide 10

- 여러 레지스터중 특정한 레지스터로의 버스 정보 전송
  - 모든 레지스터로 버스라인 연결
  - 특정 레시스터의 로드 신호를 active 시킴
  - 기호 표현 → BUS ←C, R1 ←BUS 혹은 R1 ←C (버스를 경유한다고 가정)

- 3-상태 버스 버퍼를 이용한 버스의 구성 (그림 4-5)

버스와 메모리 전송 (계속)

Slide 11

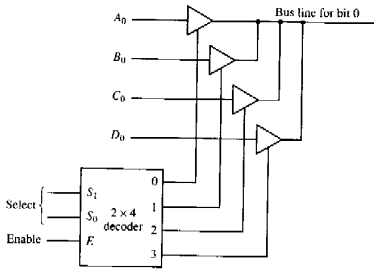


그림 4-5 3-상태 버퍼를 이용한 버스선

버스와 메모리 전송 (계속)

Slide 12

– 3-상태 버퍼 (그림 4-4)

제4장 레지스터 전송과 마이크로 연산

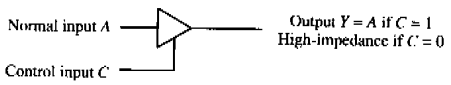


그림 4-4 3 상태 버퍼에 대한 그림 기호

* 3개의 상태를 갖음 : 0, 1, high-impedance
* 제어입력이 1 이면 보통 버퍼
* 제어 입력이 0 이면 개회로 (즉 전기적으로 연결이 끊어짐)
* 여러개의 3-상태 버퍼 출력을 연결하여 버스를 구성할수 있음
* 디코더 이용 하나의 3-상태 버퍼만이 활성화되게 함

버스와 메모리 전송 (계속)

Slide 13

- 메모리 전송
  - 메모리 읽기 기호 → Read: DR ←M[AR]
    * Read →읽기 신호가 활성화 되면
    * M[AR] →어드레스 레지스터 AR 을 이용하여 메모리 읽기를 활성화
    * DR →버스에 실린 데이타를 데이타 레지스터 DR 에 로드하라
  - 메모리 쓰기 기호 → Write: M[AR] ←R1
    * Write →쓰기 신호가 활성화 되면
    * R1 →레지스터 R1 을 선택해 R1 의 내용을 버스에 싣고
    * M[AR] →어드레스 레지스터 AR 이 가리키는 메모리에 내용을 로드하라

산술 마이크로 연산

- 마이크로 연산의 종류
  1. 레지스터 사이의 정보전송을 하는 전송 연산
  2. 레지스터에 저장된 수치에 대하여 산술연산을 하는 산술 연산
  3. 레지스터에 저장된 비수치 데이타에 대한 논리 연산
  4. 레지스터에 저장된 데이타에 대한 시프트 연산

Slide 14

- 산술연산
  - 종류 : 덧셈, 뺄셈, 인크리멘트, 디크리멘트, 시프트등
  - 덧셈 연산의 기호 표현 → $\boxed{\text{R3} \leftarrow \text{R1} + \text{R2}}$
  - 뺄셈 연산의 기호 표현 → $\boxed{\text{R3} \leftarrow \text{R1} + \bar{R}2 + 1}$
    * 보통 보수 이용
    * $\bar{R}2$ 는 R2 의 1의 보수
    * 결국 위의 기호는 $\boxed{\text{R3} \leftarrow \text{R1 - R2}}$ 와 동일
  - 곱셈 나눗셈 연산은 연산회로가 조합회로로 꾸며져 있을경우만 마이크로 연산으로 취급

산술 마이크로 연산 (계속)

  - 연산회로가 없을 경우에는 덧셈 및 뺄셈 그리고 시프트 마이크로 연산으로 소프트웨어적으로 구현됨

Slide 15

- 이진 가산기
  - 전가산기 (full-adder) : 두비트 와 하나의 캐리 비트
  - 이진 가산기 (binary adder)
    * 두 이진수를 더하는 회로
    * 전가산기를 n개 직렬 연결 : n bit 2진 가산기
    * 4bit 이진 가산기 (그림 4-6)

산술 마이크로 연산 (계속)

Slide 16

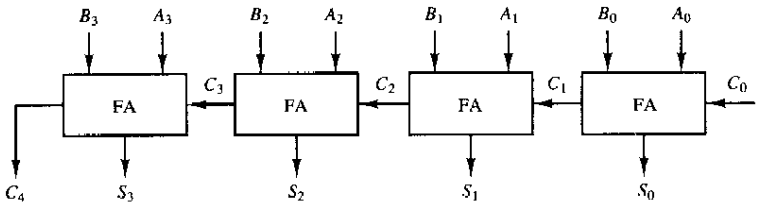


그림 4-6 4비트 이진 가산기

- 이진 가감산기
  - 뺄셈은 2의 보수를 이용한 덧셈으로 계산
  - 가산기와 감산기회로를 하나의 회로로 구현 (그림 4-7)

산술 마이크로 연산 (계속)

Slide 17

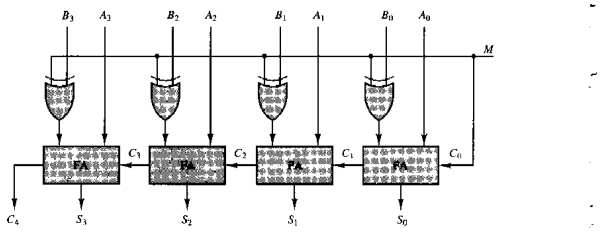


그림 4-7 4비트 가감산기

  - M = 0 일때
    - B ⊕0 = B 임으로 S = A+B
  - M = 1 일때
    - B ⊕1 = B'
    - C0 = 1 이므로 + 1
    - 결국 S = A + B' + 1 = A - B

산술 마이크로 연산 (계속)

Slide 18

- 이진 인크리멘터
  - 레지스터 값에 1 을 더하는 연산
  - 구현 방법
    1. 그림 2-10 처럼 이진 카운터 이용
    2. 반가산기 이용 (그림 4-8)

산술 마이크로 연산 (계속)

Slide 19

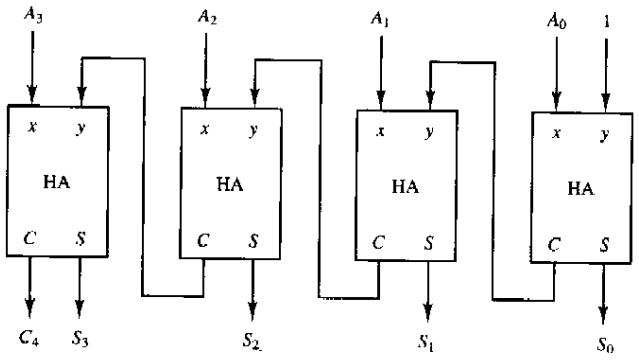


그림 4-8 4비트 이진 인크리멘터

산술 마이크로 연산 (계속)

Slide 20

- 산술회로
  - 병렬 가산기를 이용 표 4-3 의 모든 기능 수행

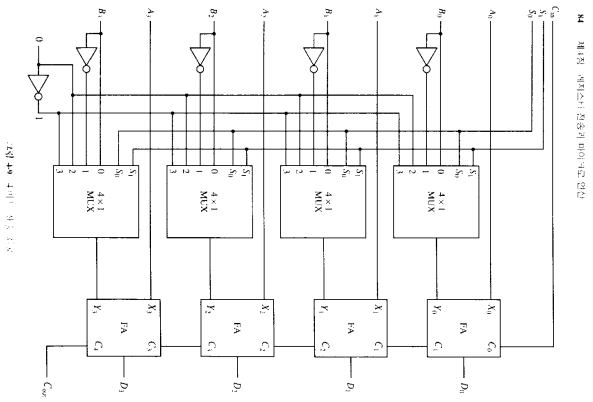
표 4-3 산술 마이크로 연산

Symbolic designation	Description
$R3 \leftarrow R1 + R2$	Contents of $R1$ plus $R2$ transferred to $R3$
$R3 \leftarrow R1 - R2$	Contents of $R1$ minus $R2$ transferred to $R3$
$R2 \leftarrow \overline{R2}$	Complement the contents of $R2$ (1's complement)
$R2 \leftarrow \overline{R2} + 1$	2's complement the contents of $R2$ (negate)
$R3 \leftarrow R1 + \overline{R2} + 1$	$R1$ plus the 2's complement of $R2$ (subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of $R1$ by one
$R1 \leftarrow R1 - 1$	Decrement the contents of $R1$ by one

산술 마이크로 연산 (계속)

Slide 21

- 4bit 전가산기와 멀티플렉서로 구현한 4bit 산술회로 (그림 4-9)



* 이진 가산기의 출력 → $\boxed{D = A + Y + Cin}$
* S1, S0, Cin 에 따른 동작표 (표 4-4)

산술 마이크로 연산 (계속)

Slide 22

4.3 논리 마이크로 연산 ••• 85

표 4-4 산술 회로의 함수표

Select			Input	Output	
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	Microoperation
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract with borrow
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

논리 마이크로 연산

- 비트열에 대한 논리 연산
- 예)
  - P: R1 ←R1 ⊕R2

    →P 가 1일때 R1 의 각비트와 R2 의 각비트를 exclusive-OR 한후 결과를 R1 에 저장

Slide 23

  - R1 = 1010 이고 R2 = 1100 이었다면 연산후 R1 = 0110 이 됨
- 마이크로 논리연산 기호
  - OR →∨
  - AND →∧
  - A' →$\bar{A}$
  - +
    - 산술연산에서는 덧셈의미
    - 제어함수에 사용될 때에는 OR 연산 의미

논리 마이크로 연산 (계속)

∗ 예) P+Q: R1 ←R2 + R3, R4 ←R5 ∨ R6
→P 혹은 Q 둘중의 하나가 1이되면 R2 와 R3 를 더하여 R1 에 저장하고 R5
Slide 24 와 R6 를 비트 AND 하여 R4 에 저장하라

- 논리 마이크로 연산표
  - 두개의 이진변수로 수행되는 16가지 논리연산 (표 4-5)
  - 표 4-5 에 대한 대수적인 표현 (표 4-6)

논리 마이크로 연산 (계속)

표 4-5 두 변수의 16개 함수에 대한 진리표

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Slide 25

표 4-6 16개의 논리 마이크로 연산

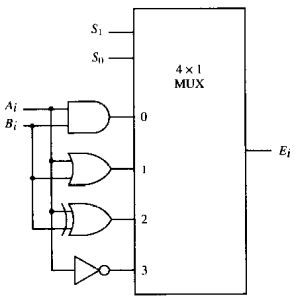
Boolean function	Microoperation	Name
$F_0 = 0$	$F \leftarrow 0$	Clear
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge \overline{B}$	
$F_3 = x$	$F \leftarrow A$	Transfer A
$F_4 = x'y$	$F \leftarrow \overline{A} \wedge B$	
$F_5 = y$	$F \leftarrow B$	Transfer B
$F_6 = x \oplus y$	$F \leftarrow A \oplus B$	Exclusive-OR
$F_7 = x + y$	$F \leftarrow A \vee B$	OR
$F_8 = (x + y)'$	$F \leftarrow \overline{A \vee B}$	NOR
$F_9 = (x \oplus y)'$	$F \leftarrow \overline{A \oplus B}$	Exclusive-NOR
$F_{10} = y'$	$F \leftarrow \overline{B}$	Complement B
$F_{11} = x + y'$	$F \leftarrow A \vee \overline{B}$	
$F_{12} = x'$	$F \leftarrow \overline{A}$	Complement A
$F_{13} = x' + y$	$F \leftarrow \overline{A} \vee B$	
$F_{14} = (xy)'$	$F \leftarrow \overline{A \wedge B}$	NAND
$F_{15} = 1$	$F \leftarrow$ all 1's	Set to all 1's

논리 마이크로 연산 (계속)

Slide 26

- 하드웨어 구현
  - 16개의 논리 마이크로 연산중에서 보통 4개만 (AND, OR, XOR, 보수) 구현
  - 마이크로 연산 논리회로의 한단 (그림 4-10)

88 — 제4장 레지스터 전송과 마이크로 연산



(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \overline{A}$	Complement

(b) Function table

그림 4-10 논리 회로의 한 단

논리 마이크로 연산 (계속)

Slide 27

- 응용
  - selective-set
    - B 레지스터의 1인 부분에 해당하는 레지스터 A 의 bit 만 setting 시킴 (논리 OR 와 같음)
    - 예) 4bit A 레지스터의 상위 두비트를 set 하라

```
      1010   : A
OR    1100   : B
      ----
      1110
```

논리 마이크로 연산 (계속)

Slide 28

- selective-complement
  - B 레지스터의 1인 부분에 해당하는 레지스터 A 의 bit 만 보수화 시킴 (논리 XOR 와 같음)
  - 예) 4bit A 레지스터의 상위 두비트만 보수화 하라

```
      1010   : A
XOR   1100   : B
      ----
      0110
```

논리 마이크로 연산 (계속)

Slide 29

- selective-clear
  - B 레지스터의 1인 부분에 해당하는 레지스터 A 의 bit 만 clear 시킴 $\rightarrow$A $\leftarrow$A $\wedge$ $\bar{B}$) 와 일치
  - 예) 4bit A 레지스터의 상위 두비트만 clear 하라

```
1010   : A
1100   : B
----
0010
```

논리 마이크로 연산 (계속)

Slide 30

- mask
  - B 레지스터의 0인 부부에 해당하는 레지스터 A 의 bit 만 clear 시킴 (논리 AND 와 같음)
  - 예) 4bit A 레지스터의 하위 두비트만 clear 하라

```
1010  : A
1100  : B
----
1000
```

논리 마이크로 연산 (계속)

Slide 31

- insert
  - A 레지스터의 특정 bit 들을 다른 특정 bit 값으로 바꿈
  - 해당 bit들을 mask 시킨후 원하는 값을 selective-set 시킴
  - 예) 4bit A 레지스터의 하위 3비트를 001 로 바꾸어라

```
1010  : A
1000  : B (masking, AND)
----
1000  : A
0001  : B (selective-set, OR)
----
1001
```

논리 마이크로 연산 (계속)

Slide 32

- clear
  - A 레지스터와 B 레지스터가 같은 값이면 A 를 0으로 clear 하라 (XOR 와 같음)
    $\rightarrow$ $\boxed{\text{A} \leftarrow \text{A} \oplus \text{B}}$
  - 예)

```
1010   : A
1010   : B
----
0000
```

시프트 마이크로 연산

Slide 33

- 사용
  1. 직렬 전송시
  2. 산술이나 논리연산 및 데이타 처리시
- 종류
  1. 논리 시프트 (logical shift)
  2. 순환 시프트 (circular or rotate shift)
  3. 산술 시프트 (arithmetic shift)
- 논리 시프트 (logical shift)
  - 직렬 입력 0
  - 기호는 shl (shift left), shr (shift right)
  - 예)
    $\rightarrow$ $\boxed{\text{R1} \leftarrow shl\ \text{R1}}$: R1 을 왼쪽으로 한비트 shift

시프트 마이크로 연산 (계속)

→ $\boxed{\text{R2} \leftarrow shr\ \text{R2}}$: R2 를 오른쪽으로 한비트 shift

- 순환 시프트 (circular or rotate shift)
  - 직렬입력은 직렬출력에서 가져옴 →정보 손실 없음
  - 기호는 cil (cicular left), cir (circular right)

Slide 34

- 산술 시프트 (arithmetic shift)
  - 부호가 있는 이진수를 시프트
  - 왼쪽으로 한비트 shift →이진수에 곱하기 2와 같음
  - 오른쪽으로 한비트 shift →이진수에 나누기 2와 같음
  - 오른쪽 shift 시 부호비트를 보존하기위하여 msb 는 변함없음
  - 왼쪽 shift 시 lsb 에는 0 이 채워지고 msb 는 msb 보다 하나적은 비트 가 이동됨
    - 이때 msb 값이 바뀌면 overflow 가 발생한것임
    - overflow check : $V_s = R_{n-1} \oplus R_{n-2}$
      →V_s 가 1 이면 overflow 발생한 것임

시프트 마이크로 연산 (계속)

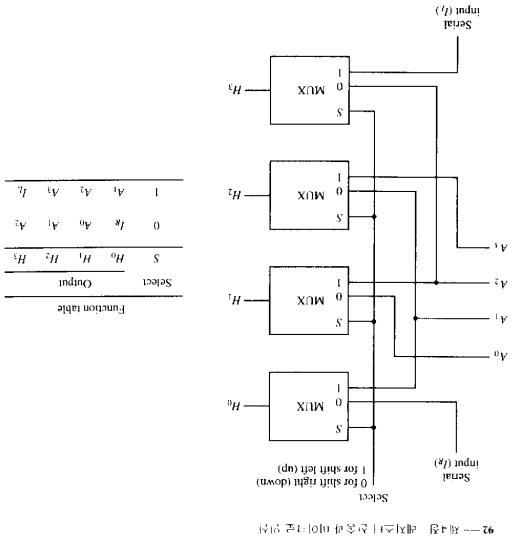
  - 예) A = 1110 이면
    - 십진수로 -2 이다
    - 오른쪽으로 한비트 shift →A = 1111 →십진수로 -1
    - 왼쪽으로 한비트 shift →A = 1100 →십진수로 -4
    - 왼쪽으로 또 한비트 shift →A = 1000 →십진수로 -8
    - 왼쪽으로 또 한비트 shift →A = 0000 →십진수로 0 (V_s 는 1 이므로 overflow)

Slide 35

- 하드웨어 구현
  - 멀티플렉서를 이용한 구현 (그림 4-12)

시프트 마이크로 연산 (계속)

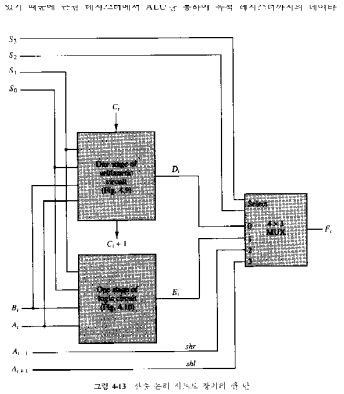
Slide 36



산술 논리 시프트 장치

Slide 37

- 보통 산술 논리 및 시프트 연산은 ALU 라는 하나의 장치에 조합회로로 구현됨
- 산술 논리 시프트를 하는 장치의 한단 (그림 4-13)



산술 논리 시프트 장치 (계속)

Slide 38

- 산술 논리 시프트 장치에 대한 함수표 (표 4-8)