



4. 레지스터와 마이크로연산

목표

- 마이크로 연산과 그 구현 방법을 이해하고 활용할 수 있음.



레지스터 전송 언어



컴퓨터 구조 정의

- ✓ 레지스터 종류 및 기능
- ✓ 레지스터의 데이터를 이용한 마이크로 연산
 - 마이크로 연산 수행을 위한 제어

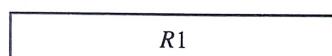


마이크로 연산

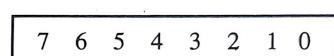
- Microoperation, μ -operation, μ -op
- 레지스터의 데이터를 대상으로 한 기본 연산
- 1 클록 펄스에 실행
- 레지스터 전송 언어 이용 표현
 - RTL, Register Transfer Language



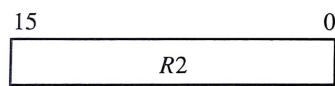
레지스터 표현



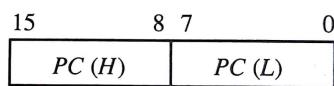
(a) Register R



(b) Showing individual bits



(c) Numbering of bits

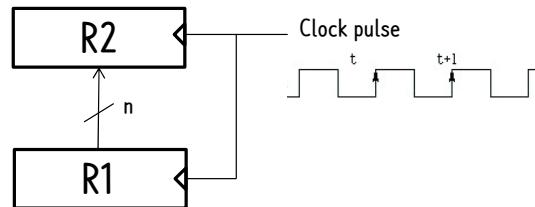


(d) Divided into two parts

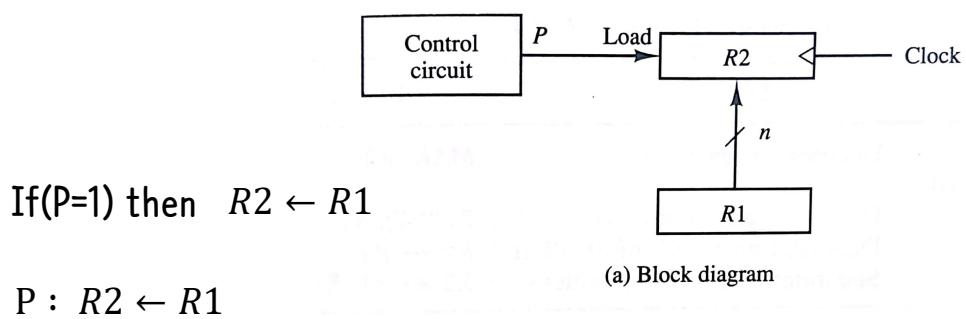


레지스터 전송

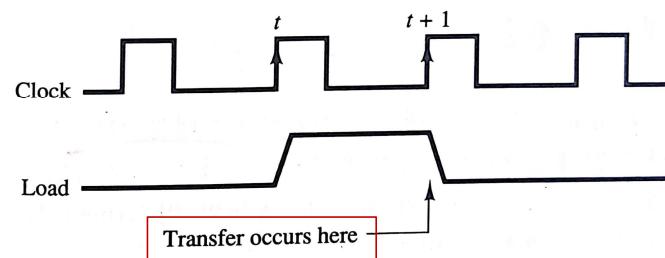
$$R2 \leftarrow R1$$



제어조건을 가진 레지스터 전송



(a) Block diagram



레지스터 전송 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 \leftarrow	Denotes transfer of information	$R2 \leftarrow R1$
Comma ,	Separates two microoperations	$R2 \leftarrow R1, R1 \leftarrow R2$



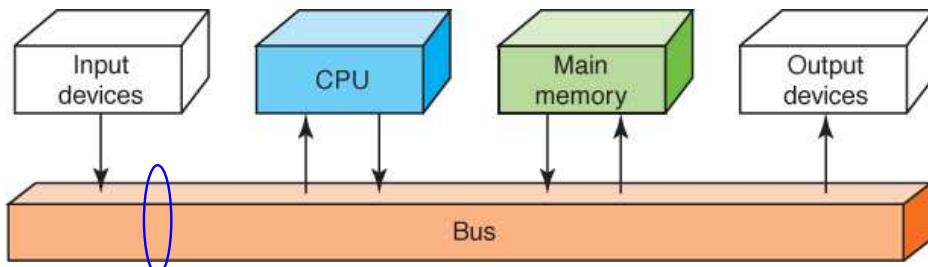
버스와 메모리전송



정보의 흐름

- 버스, BUS

- 각 구성요소간 데이터를 교환할 수 있는 전선들의 모음
- 데이터 : 신호

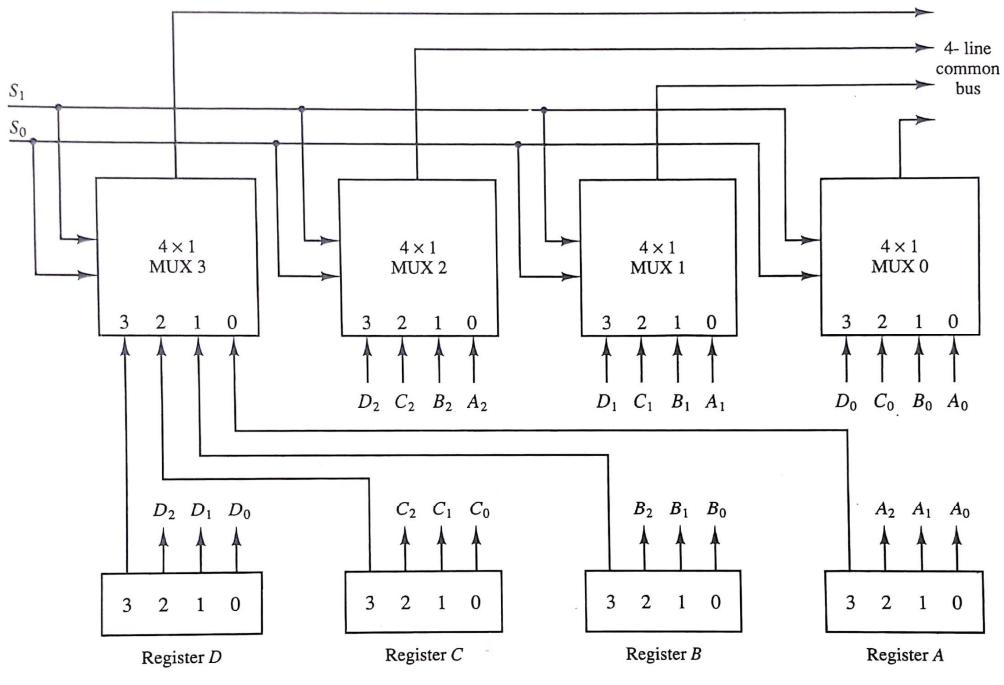


신호 전송을 위한 전선의 수 = 한번에 전달할 수 있는 비트의 수.
32 가닥의 전선 → 각각 1 비트의 데이터를 전송하기 때문에 한 순간에
32 비트의 데이터를 전송할 수 있음. ➔ 버스 폭(bus width)

버스, Bus

- 레지스터간 전송을 위한 공유 전송 라인
- 어떤 시점에 하나의 레지스터만 전송 가능

버스 구현의 예

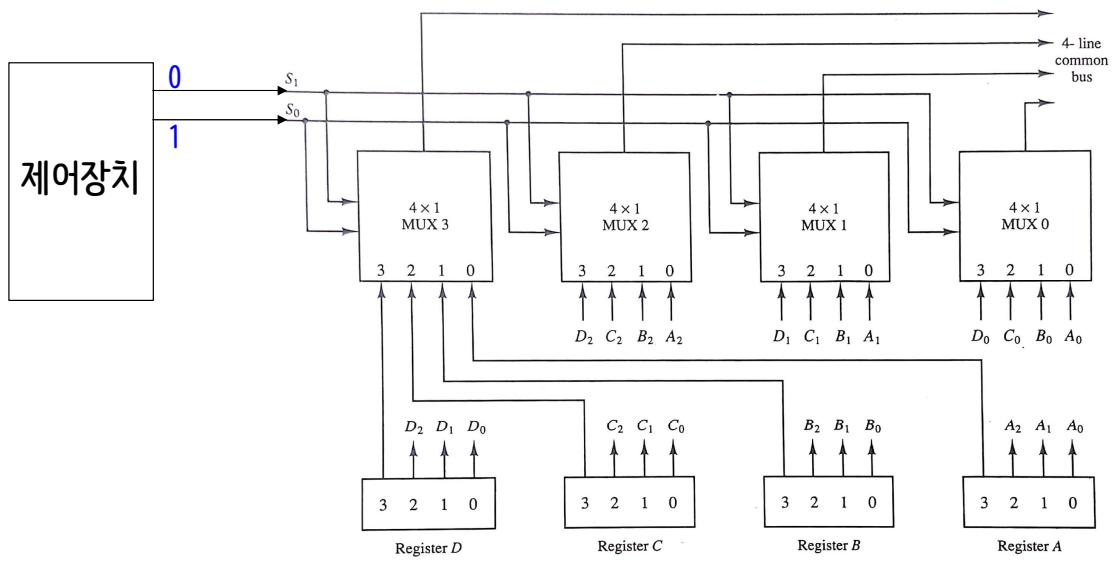


버스 구현의 예

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

레지스터 B의 내용을 버스에 올리기 위한 S_1 , S_0 의 값은?

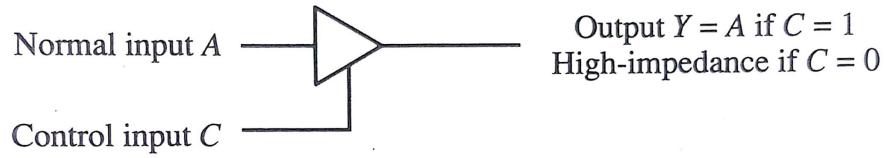
버스 구현의 예



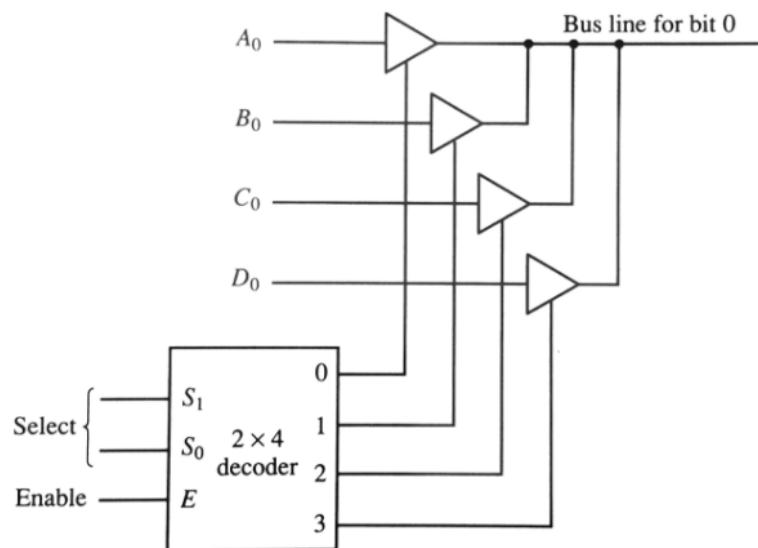
3-상 게이트

- 세가지 상태를 가지는 게이트
 - 0
 - 1
 - High impedance
 - 고저항 상태
 - 게이트 출력이 다른 회로와 연결되어 있지 않는 효과

3-상 버퍼



3-상 버퍼를 이용한 버스



3-상태 버퍼를 이용한 2 비트 버스

- 버스 폭(bus width) : 2

디코더와 3-상태 버퍼를 이용하여 4개의 2비트 레지스터로
구성된 버스 설계 방법은?



메모리 전송

- M - 메모리 표현
- AR - 주소 레지스터: MAR
- DR - 데이터 레지스터: MBR
- 메모리 Read $r : DR \leftarrow M[AR]$
- 메모리 Write $w : M[AR] \leftarrow DR$



3개의 1-bit 레지스터와 8X1 RAM의 버스 연결 예



산술 마이크로 연산



마이크로 연산 종류

- 전송 마이크로 연산
- 산술 마이크로 연산
- 논리 마이크로 연산
- 시프트 마이크로 연산



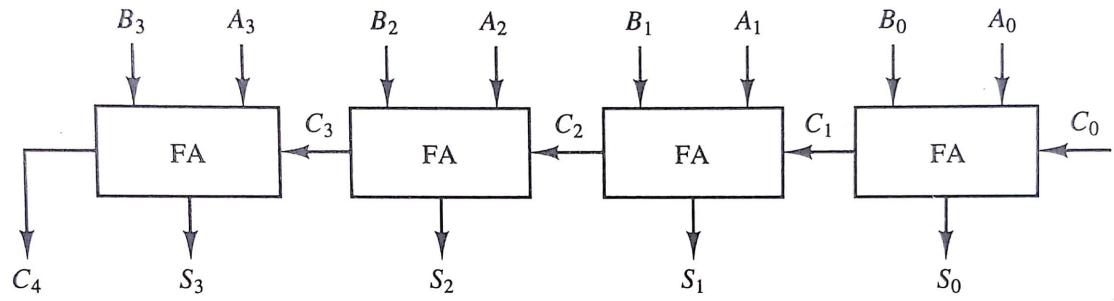
산술 마이크로 연산

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 $R3$
$R2 \leftarrow \bar{R2}$	Complement the contents of $R2$ (1's complement)
$R2 \leftarrow \bar{R2} + 1$	2's complement the contents of $R2$ (negate)
$R3 \leftarrow R1 + \bar{R2} + 1$	$R1$ 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



4-비트 이진 가산기

- 두 4-비트 레지스터의 값을 덧셈
- 4개의 전가산기 이용

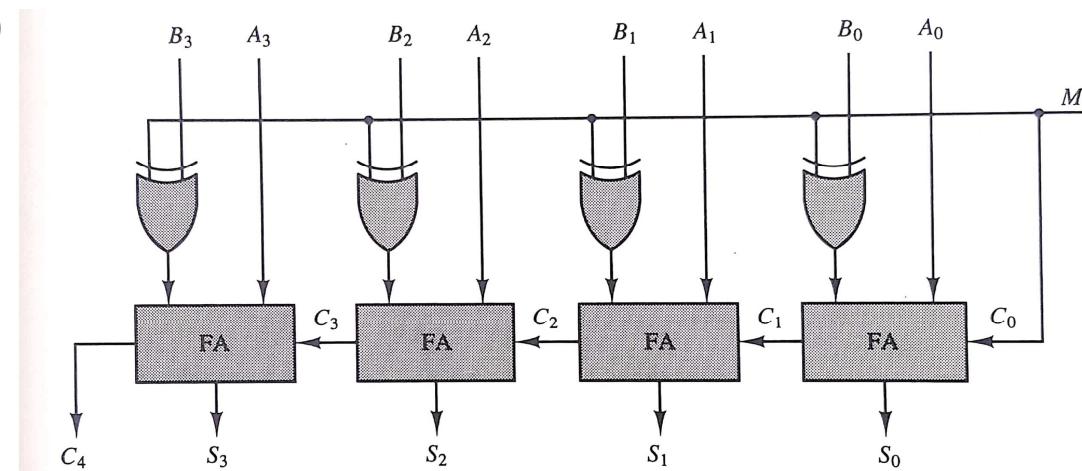


4-비트 가감산기

- 두 4-비트 레지스터의 값을 덧셈/뺄셈 수행
- 동작 제어, M

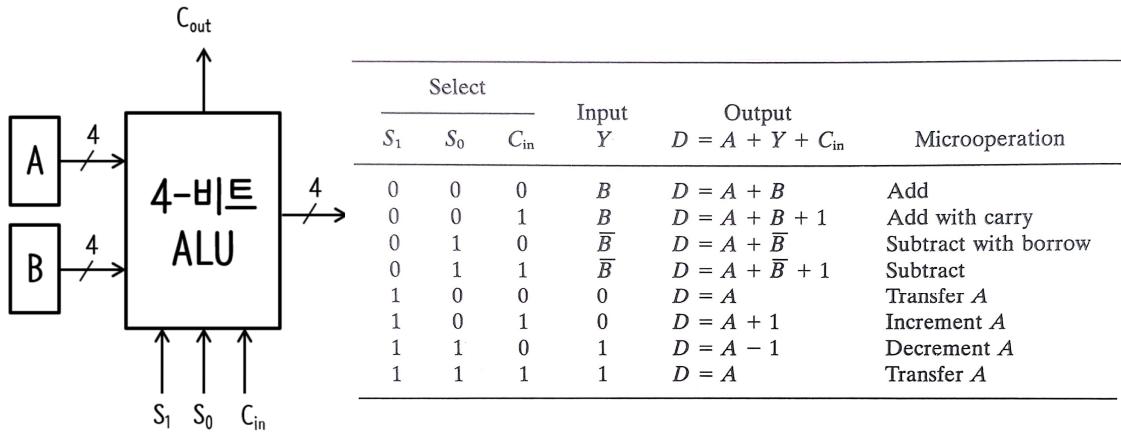
- 1 : 뺄셈

- 0

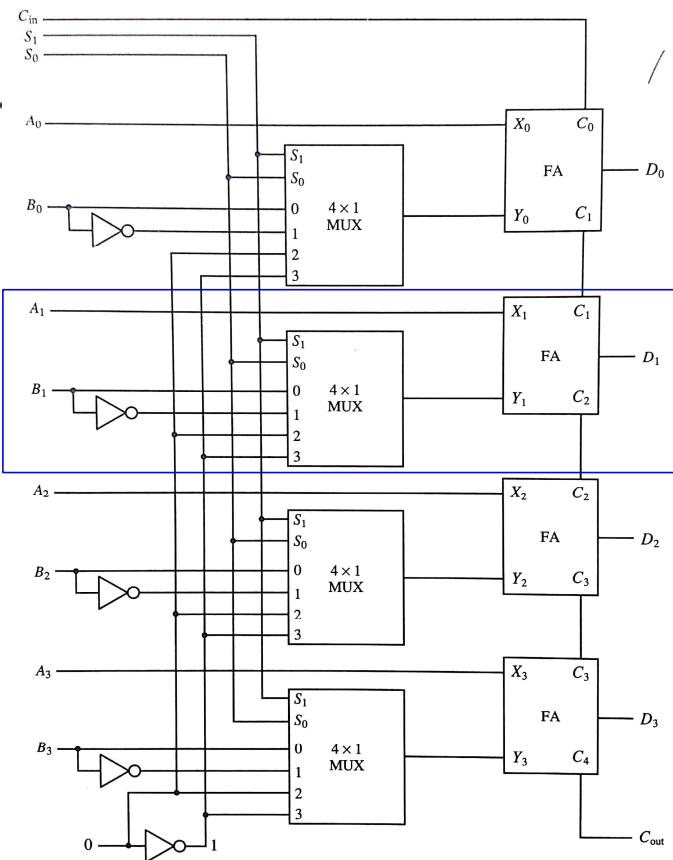


4-비트 산술 연산 장치

- 두 4-비트 레지스터의 값을 이용, 산술 연산



4-비트 산술 연산 장치



논리 마이크로 연산



논리 마이크로 연산

- 두 레지스터의 대응되는 비트간 연산 수행
- 논리 마이크로 연산의 예 :

$$P : R1 \leftarrow R1 \oplus R2$$

$$P + Q : R1 \leftarrow R2 + R3, R4 \leftarrow R5 \vee R6$$

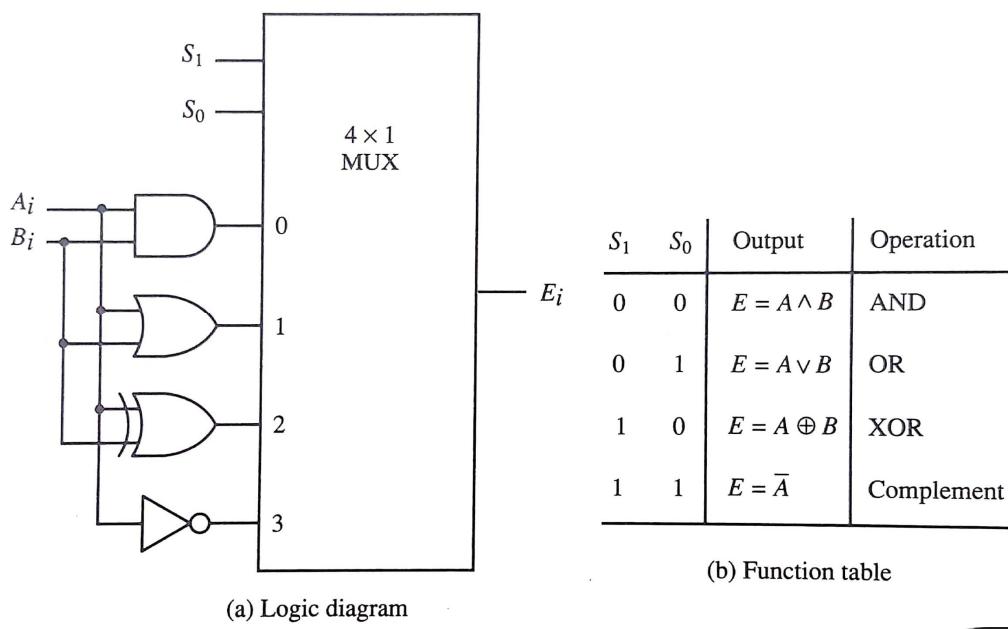


논리 마이크로 연산

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

논리 연산 장치 구현

- AND, OR, XOR, complement 수행



논리 연산 응용

- selective-set

- 선택된 비트의 값을 1로 설정
- OR 연산 이용

1010 : 레지스터
OR 1100 : 비트 선택

1110

- selective-clear

- 선택된 비트의 값을 0으로 설정
- NOT과 AND 연산 이용

1010 : 레지스터
1100 : 비트 선택
NOT
AND 0011

0010



논리 연산 응용

- mask

- 선택된 비트를 제외한 비트의 값을 0으로 클리어
- AND 연산 이용

1010 : 레지스터
AND 1100 : 비트 선택

1000

- insert

- 비트의 값을 원하는 값으로 변경
- mask를 한 후 OR 연산 이용

레지스터에 저장된 값 1011_1010의 비트
3~0의 값을 0111으로 변경을 하고자 한다.
어떻게?



시프트 마이크로 연산



시프트 마이크로 연산

- 레지스터의 값을 1-비트 왼쪽/오른쪽 이동
- 사용 목적
 - 데이터 직렬 전송
 - 곱셈/나눗셈
- 종류
 - 논리 시프트
 - 순환 시프트
 - 산술 시프트



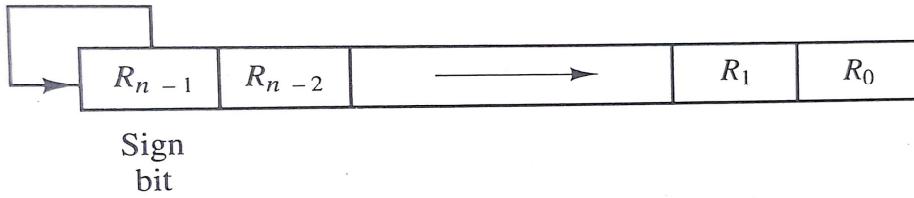
산술 시프트

- 부호가 있는 이진수 시프트
 - 왼쪽 시프트 - 2를 곱한 효과
 - 오른쪽 시프트 - 2를 나눈 효과

시프트 방법

- 왼쪽 시프트 - 노리 시프트와 동일

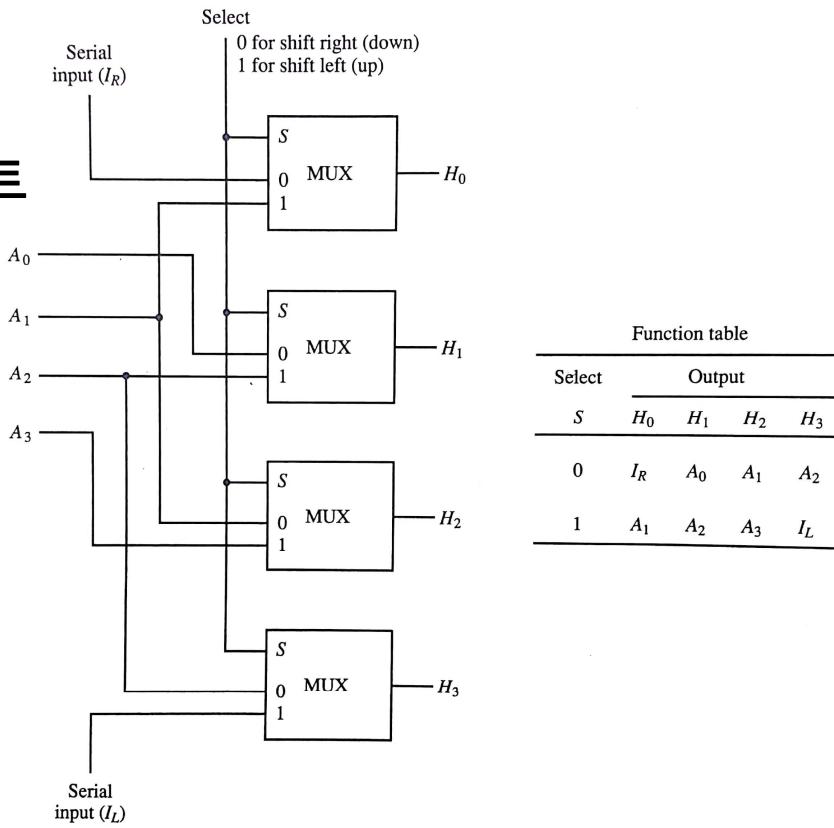
- 오른쪽



시프트 마이크로 연산

Symbolic designation	Description
$R \leftarrow \text{shl } R$	Shift-left register R
$R \leftarrow \text{shr } R$	Shift-right register R
$R \leftarrow \text{cil } R$	Circular shift-left register R
$R \leftarrow \text{cir } R$	Circular shift-right register R
$R \leftarrow \text{ashl } R$	Arithmetic shift-left R
$R \leftarrow \text{ashr } R$	Arithmetic shift-right R

4-비트 시프트 연산 장치



산술 논리 시프트 연산 장치



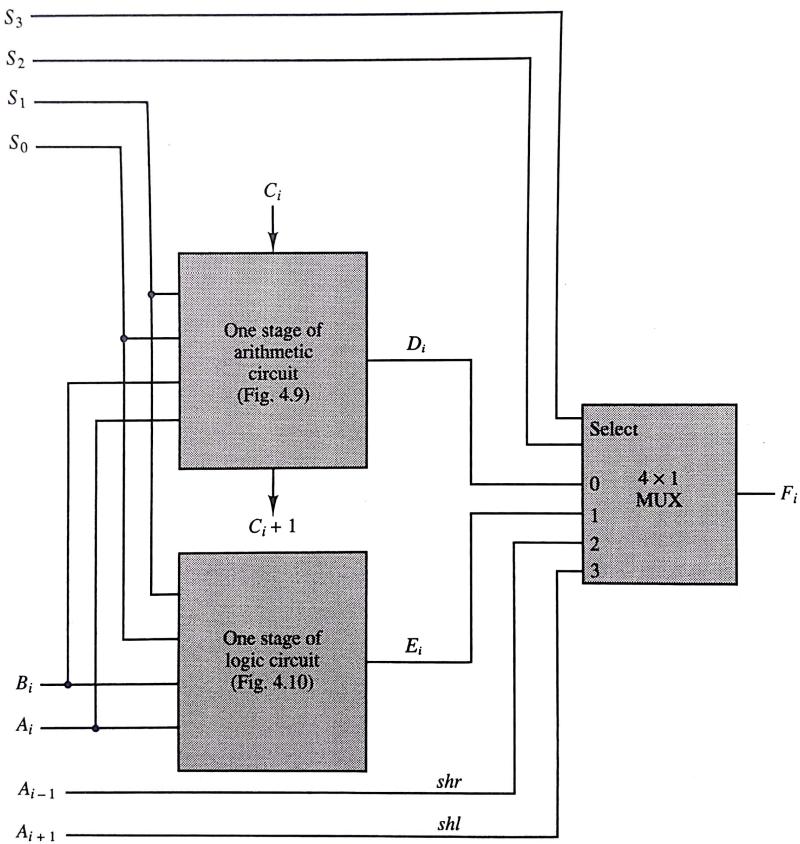
산술 논리 연산 장치

- Arithmetic Logic Unit, **ALU**
- 산술 연산, 논리 연산, 시프트 연산 수행
 - 각 연산 장치를 위한 별도의 레지스터를 두지 않음

ALU 기능표

Operation select					Operation	Function
S_3	S_2	S_1	S_0	C_{in}		
0	0	0	0	0	$F = A$	Transfer A
0	0	0	0	1	$F = A + 1$	Increment A
0	0	0	1	0	$F = A + B$	Addition
0	0	0	1	1	$F = A + B + 1$	Add with carry
0	0	1	0	0	$F = A + \bar{B}$	Subtract with borrow
0	0	1	0	1	$F = A + \bar{B} + 1$	Subtraction
0	0	1	1	0	$F = A - 1$	Decrement A
0	0	1	1	1	$F = A$	Transfer A
0	1	0	0	\times	$F = A \wedge B$	AND
0	1	0	1	\times	$F = A \vee B$	OR
0	1	1	0	\times	$F = A \oplus B$	XOR
0	1	1	1	\times	$F = \bar{A}$	Complement A
1	0	\times	\times	\times	$F = \text{shr } A$	Shift right A into F
1	1	\times	\times	\times	$F = \text{shl } A$	Shift left A into F

ALU 하드웨어 구현의 예



Q&A

