

컴퓨터구조론

I. Digital Logic Circuit

1. Logic gates

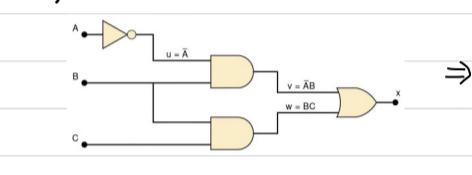
Name	Symbol	Algebraic function	Truth table	Name	Symbol	Algebraic function	Truth table
AND		$X = AB$		NAND		$X = (\bar{A}B)^*$	
OR		$X = A+B$		NOR		$X = (\bar{A}+\bar{B})^*$	
Inverter		$X = \bar{A}$		XOR		$X = (A \oplus B)$	
Buffer		$X = A$		XNOR		$X = (\bar{A} \oplus \bar{B})^*$	

2. Boolean algebra

i, (1) $x + 0 = x$ (2) $x \cdot 0 = 0$
(3) $x + 1 = 1$ (4) $x \cdot 1 = x$
(5) $x + x = x$ (6) $x \cdot x = x$
(7) $x + x' = 1$ (8) $x \cdot x' = 0$
(9) $x + y = y + x$ (10) $xy = yx$
(11) $x + (y + z) = (x+y) + z$ (12) $x(yz) = (xy)z$
(13) $x(y+z) = xy + xz$ (14) $x + yz = (x+y)(x+z)$
(15) $(xy)' = x'y'$ (16) $(xy)' = x' + y'$
(17) $(x')' = x$

3. Map simplification

i, truth table



A	B	C	u= \bar{A}	v= \bar{AB}	w= \bar{BC}	x= \bar{ABC}
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	1	0	1	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	1	1	1

ii, Sum-of-products form

i, sum-of-products (SOP) : 합의곱, AND 항이 서로 OR로 묶여 있다.
ex, $ABC + \bar{ABC}$, $AB + \bar{AB} + \bar{C} + D$
여기서는 각 항의 하나의 변수 앞뒤 넣으면 안됨 (\bar{ABC} , \bar{RST} X)

ii, product of sums (POS) : 합의곱, 2개 이상의 OR 항이 AND로 묶여 있다.

ex, $(A+\bar{B}+C) \cdot (A+c)$, $(A+\bar{B})(\bar{C}+D)F$

iii, Simplify Logic Circuits

DeMorgan's theorems \rightarrow SOP \rightarrow 공통인수 찾기와 합 출이기

by DeMorgan's theorems 여기서는 하나의 변수 앞뒤를 넣지 않기 설정

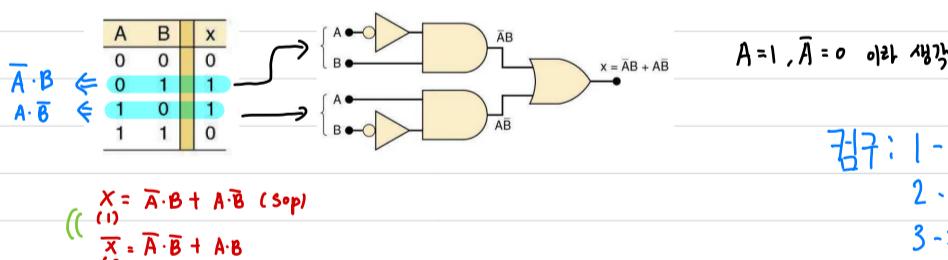


$$Z = ABC + \bar{A}(\bar{B} + C) = ABC + \bar{A}\bar{B} + \bar{AC} = ABC + \bar{A} + \bar{B} + \bar{C}$$

$$= AC(B+\bar{B}) + \bar{A}\bar{B} = AC + \bar{A}\bar{B} = A(\bar{B} + C)$$

IV, design Combinational Logic circuits

truth table (진리표) 작성 \rightarrow 출력이 1인것을 고르고 AND 항으로 변환 \rightarrow SOP \rightarrow 간략화



V, Karnaugh Map (K-map)

논리식을 간략화하거나 진리표를 상정하는 놀이처럼 변환하는데 사용

i, 2 variables

A	B	X
0	0	0
0	1	1
1	0	0
1	1	1

$$\{ X = \bar{A}\bar{B} + AB \} \Rightarrow \begin{array}{c} 0 & 1 \\ \bar{A} & B \\ \hline 0 & 0 \\ 1 & 0 \\ \end{array}$$

A=1, $\bar{A}=0$ 이라 생각

2-23

3-26

4-23

5-25

김7: 1-21

2-23

3-26

4-23

5-25

ii, 3 variables

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$$\{ X = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} \} \Rightarrow \begin{array}{c} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \\ \hline 0 & 1 & 1 & 0 \\ \bar{A} & \bar{B} & \bar{C} & C \\ \hline 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ \end{array}$$

Horizontally adjacent squares differ only in one variable.

Vertically adjacent squares differ only in one variable.

iii, Looping : 큰 그룹으로 묶었을 때 단지 공동인자만 최종식에 나타난다.

· 역형태와 예상 아닌 형태를 모두 취하고 있는 변수를 제거

1, two pairs : 변수 히트을 쇼할 수 있음

CD	CD	CD	CD
0	0	1	1
0	1	1	0
1	0	0	0

(d)

$X = \bar{A}BCD + ABCD + \bar{ABC}D + ABC\bar{D}$

$= \bar{ABC} + ABC$

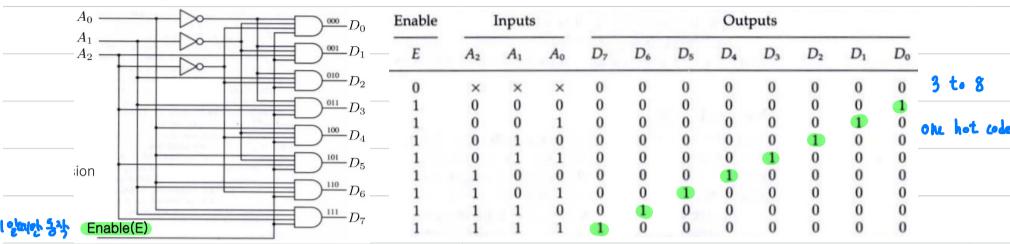
$X = \bar{ABC} + ABC$

$= \bar{ABC} +$

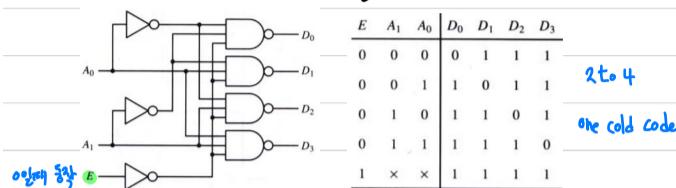
2. Digital components

1. Decoder

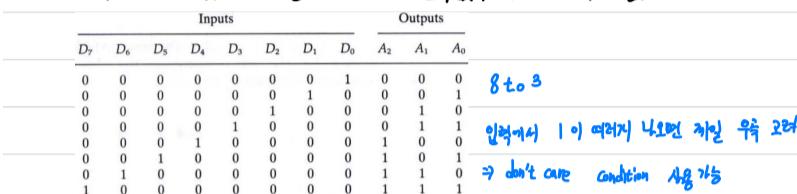
- n 비트 입력 정보를 최대 2^n 개의 출력으로 변환, n to m line decoder, $N \times M$ decoder



- NAND Gate decoder : 보수된 형태로 출력을 만드는 3-to-8 decoder

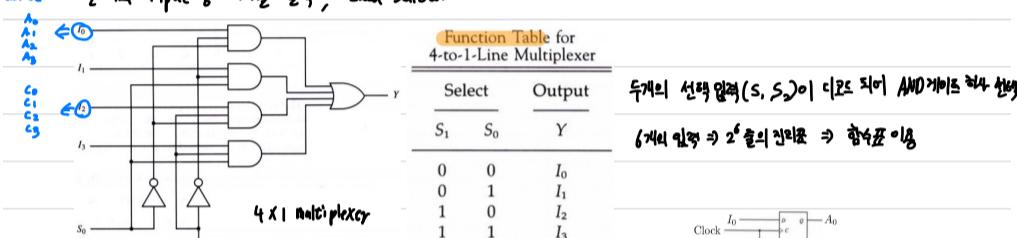


- Encoder : 디코더와 반대로 2^k개의 입력값에 대해 n개의 이진코드



2. Multiplexer

- 2^k개의 input 중 하나를 선택, data selector

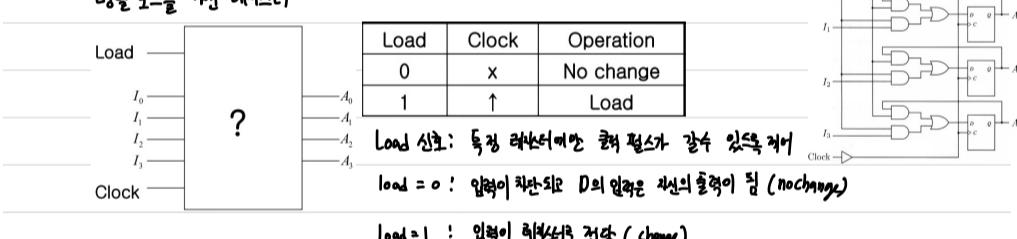


3. Register

- n 비트의 메모리는 n 비트의 이진 정보를 저장하기 위해 n 개의 플리프루트으로 구성

- clear 입력은 값이 0이 될 경우 출력을 모두 0으로 초기화

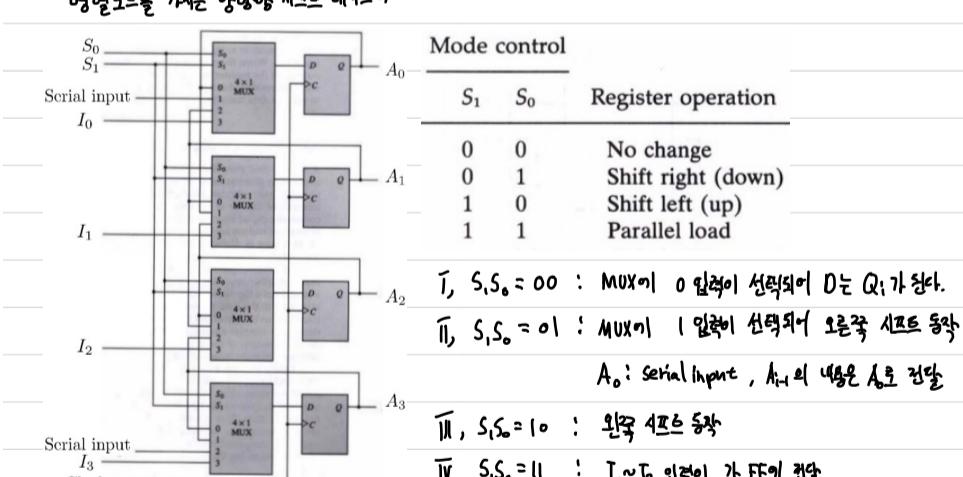
- 병렬 워드를 가진 메모리



4. Shift Register

- 레지스터에 저장된 이진 정보(Q)를 이동시킴

- 병렬로드를 가지는 양방향 시프트 레지스터

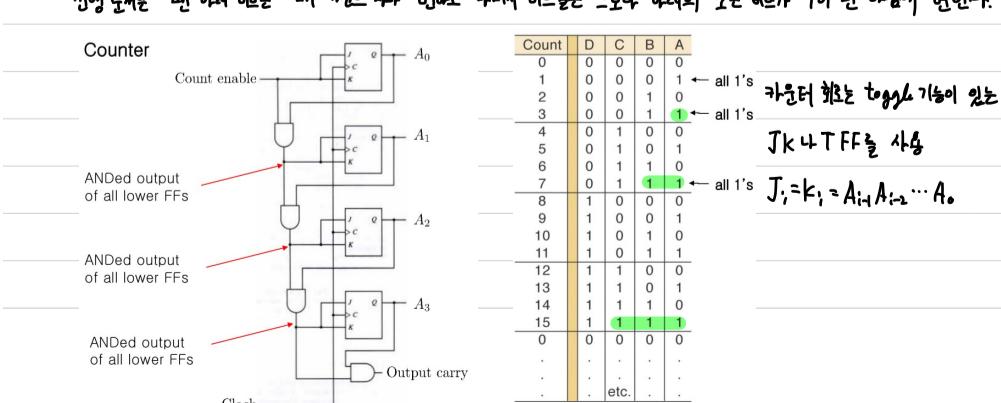


5. Binary Counter

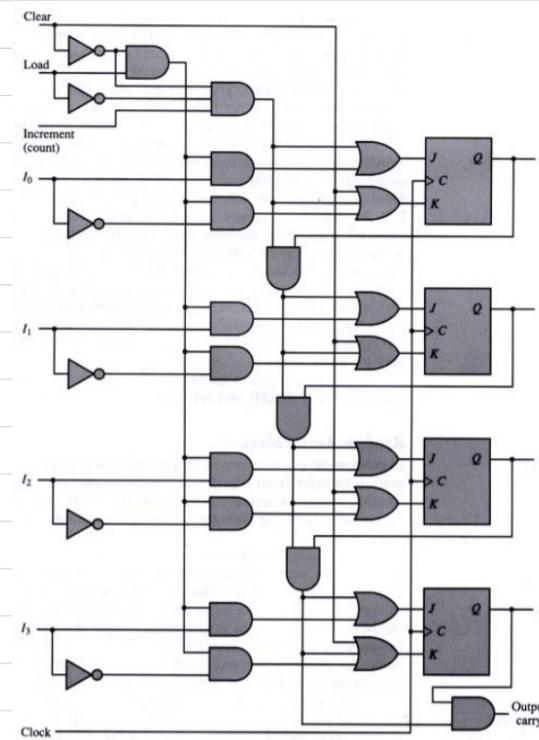
- 마지막 순서대로 상태 변화가 진행되는 레지스터

- n 개의 FF를 가진 n 비트 차인 카운터는 0에서 $2^n - 1$ 까지 카운트한다.

- 진행 순서는 맨 아래 비트는 매 카운트마다 변화하고 나머지 비트들은 그보다 아래로 모든 비트가 1이 된 다음에 변화한다.



. 병렬 워드를 가진 이진 컴퓨터 : 컴퓨터의 초기값 설정을 위한



$\bar{I}_j, clear = 1$: 모든 FF의 K 입력이 1로 세트 \Rightarrow 출력 = 0
 $\bar{I}_i, load = 1$: 나머지의 병렬 입력이 0으로 초기화.

6. Memory Unit

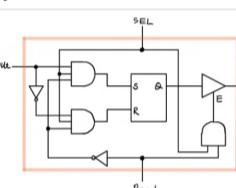
\bar{I}_j Word: an entity of bits that move in and out of storage as a unit.

Byte: eight bits

Each word is assigned address from $0, 1, \dots, 2^k - 1$

Selection of a specific word is done by applying the k-bit binary address to the address lines

\bar{I}_i , RAM (Random Access Memory)



SEL	Read	operation	Q(CE)	Output	S	R	E
0	x	No change	Q(CE)	H-Z	0	0	0
1	0	Writing	Input	H-Z	Input	Input	0
1	1	reading	Q(CE)	Q(CE)	0	0	1

3. Data Representation

1. Data types

i, binary to octal or hexadecimal

1	2	7	5	4	3
0	1	1	1	0	0
A	F	6	3		

octal
binary
hexadecimal

II, decimal to

i, binary: ex, 41.6875_{10}

Integer	Fraction
41	0.6875
20	$\frac{2}{2}$
10	$\frac{0}{2}$
5	$\frac{1}{2}$
2	$\frac{0}{2}$
1	$\frac{1}{2}$
0	$\frac{0}{2}$

$(41)_{10} = (101001)_2$ $(0.6875)_{10} = (0.1011)_2$ $(41.6875)_{10} = (101001.1011)_2$

ii, hex: repeat division until quotient is zero

$\frac{423}{16} = 26$ + remainder of 7
$\frac{26}{16} = 1$ + remainder of 10
$\frac{10}{16} = 0$ + remainder of 1
$423_{10} = 1A7_{16}$

2 Complements

T, 2's complement system : 부호를 갖는 2진수를 표현하는 체계 (Negation: 부정)

주어진 binary bit 이 1의 보수를 ($0 \rightarrow 1, 1 \rightarrow 0$) 취하고 LSB에 1을 더한다.

ex, -9를 2진수로 표현: $9_{10} : 1001_2 \rightarrow 1010 \rightarrow 1011 : -9_{10}$ (부호를 이용한 5비트식)

decimal 이진법정수(음수) \rightarrow binary : 양수의 binary (sing bit 포함) \Rightarrow 2의 보수
 binary bit (MSB=1) \rightarrow decimal : 부호는 음수이고 그는 2의 보수 취해서 구함
 binary bit (MSB=1), 2의 보수 \rightarrow 결과: 원래 binary decimal 값과 부호만 바꿈

MSB가 1이면 두비트으로
비트 전환하여 만듦

II, 부호확장

레지스터의 크기는 각 수를 저장할 2진 자리수 결정

양수 \rightarrow 0을 더붙임, 음수 \rightarrow 1을 더붙임

III, special case

$1000 = -2^3 = -8$
 $10000 = -2^4 = -16$
 $100000 = -2^5 = -32$

2의 보수를 하면 자가자진으로 (\because 그 비트로 나누낼 수 있는 음수한계값)
 N개의 크기 범위가 2의 부정체계에서 표현할 수 있는 전체 값의 범위
 $(-2^N \sim 2^N - 1)$

IV, Addition in 2's complement system

* 각 수의 부호비트가 크기 비트와 같은 방식으로 연산, 부호비트에서 자리를 뺀다.

i, 두 양수

$$\begin{array}{r} +9 \rightarrow 0 1001 \text{ (augend, 피가산수)} \\ +4 \rightarrow 0 0100 \text{ (addend, 가수)} \\ \hline 0 1101 \text{ (sum=+13)} \end{array}$$

→ sign bits

ii, 양수와 크기작은 음수

$$\begin{array}{r} +9 \rightarrow 0 1001 \text{ (augend)} \\ -4 \rightarrow 1 1100 \text{ (addend)} \\ \hline 1 0 0101 \text{ (sum=+5)} \end{array}$$

자리올림 무시 ↗ sign bit

iii, 양수와 크기큰 음수

$$\begin{array}{r} -9 \rightarrow 1 0111 \text{ (augend)} \\ +4 \rightarrow 0 0100 \text{ (addend)} \\ \hline 1 1011 \text{ (sum=-5)} \end{array}$$

→ sign bits

iv, 두 음수

$$\begin{array}{r} -9 \rightarrow 1 0111 \text{ (augend)} \\ -4 \rightarrow 1 1100 \text{ (addend)} \\ \hline 1 1 0011 \text{ (sum=-13)} \end{array}$$

→ sign bits

V, 크기가 같고 부호 반대

$$\begin{array}{r} -9 \rightarrow 1 0111 \text{ (augend)} \\ +9 \rightarrow 0 1001 \text{ (addend)} \\ \hline 0 0000 \text{ (sum=0)} \end{array}$$

→ sign bits

VI, Subtraction in 2's complement system

i, how to : 감수에 부정을 주한다. \rightarrow 이 것을 피감수에 더한다.

$$\begin{array}{l} (+9) - (+4) \Rightarrow \text{피감수} (+9) : 0100 \\ \quad \quad \quad \text{감수} (+4) : 00100 \xrightarrow{\text{부정}} (-4) : 11100 \\ \hline 01001 (+9) \\ - 00100 (+4) \\ \hline 01001 (-5) \end{array}$$

$$\begin{array}{l} (-4) - (+9) \Rightarrow \text{피감수} (-4) : 11100 \\ \quad \quad \quad \text{감수} (+9) : 01001 \rightarrow (-9) : 10111 \\ \hline 11100 (-4) \\ - 01001 (+9) \\ \hline 11001 (-13) \end{array}$$

vii, Similar at decimal

$$\begin{array}{l} M \geq N : 72532 - 13250 = 59282 \\ M = 72532 \\ 10's complement of N = +86750 \\ \hline \text{Sum} = 1 59282 \\ \text{End carry will be discarded} \\ \text{Result} = 59282 \end{array}$$

$$\begin{array}{l} M < N : 13250 - 72532 = -59282 \\ M = 13250 \\ 10's complement of N = +27468 \\ \hline \text{Sum} = 0 40718 \\ \text{Result} = \text{negative } 59282 \\ = 10's complement of 59282 \end{array}$$

iii, overflow : 결과가 주어진 크기 비트로 표현 못할 때 오류가 발생함 \Rightarrow 넘침 조건을 검출하는 최초 단계

Addition: 같은 부호의 수

ex, $+9 \rightarrow 0 1001$

$+8 \rightarrow 0 1000$

1 0001

incorrect magnitude

incorrect sign

Subtraction: 다른 부호의 수

ex, 9's 보수 of 546790₍₁₀₎

$\Rightarrow 999999 - 546790 = 453209$

· $(r-1)$'s 보수: $N = r^n - V = (r-1)$'s 보수 + 1

3. 소수점 표현

i, 고정 소수점: 분수를 표현하기 위해 레지스터의 맨 왼쪽이 숫자 (0.XXXX)

정수를 " " 오른쪽이 숫자 (XXXX.0)

· 실제 소수점이 존재하는 것은 아니지만, 마지막 것처럼 기정하고 저장된 숫자를 분수 or 정수 취급함

ii, 부동 소수점: 2 parts ($m \times r^e$)

m (가수, mantissa): unsigned, fixed point

e (지수, exponent): 소수점의 위치

ex, $6192.789 = m: 0.6192789, e: +14$

$1001.11 = m: 0.100111, e: 000100$: 맨 왼쪽 0은 앙수를 나타내고 소수점은 나타나 있지 않지만

부동소수점으로 간주됨

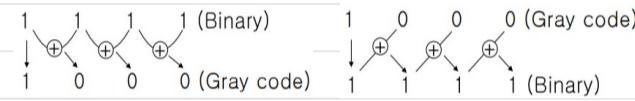
· normalization: Making the MSD of m nonzero, so that provide maximum possible precision

	Representation	Precision
Original number	12.34	100%
Normalized	0.1234×10^2	100%
Not normalized	0.0012×10^4	97.2%

4. Gray code

· change by only one bit

· Conversion (carry is ignored)



5. Error detection code

· 외부 쌍을 통해 error 발생할 수 있음

· Parity code [odd, even parity]: · only detection, no correction

· only odd number of errors can be detected

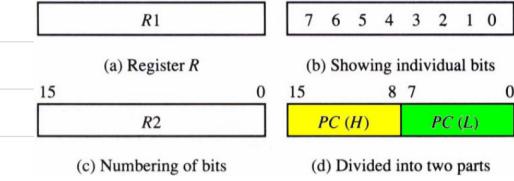
0 1 1

1 0 1

4. Register transfer and Microoperations

1. Register transfer

i. Block diagram



ii. Transfer

$P: R2 \leftarrow R1$: $\therefore P$: control signal generated in the control section

- transfer executed only if $P=1$
- source of $R1$ does not change

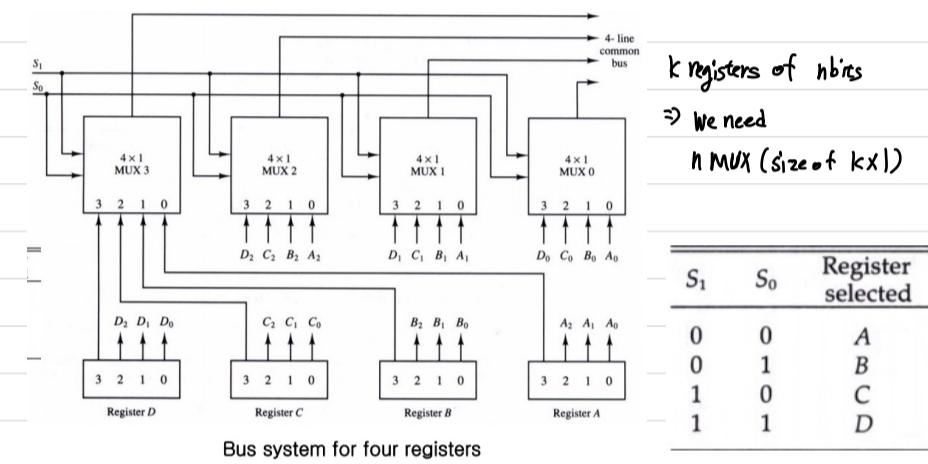
2. Bus and Memory transfers

i. Common Bus system

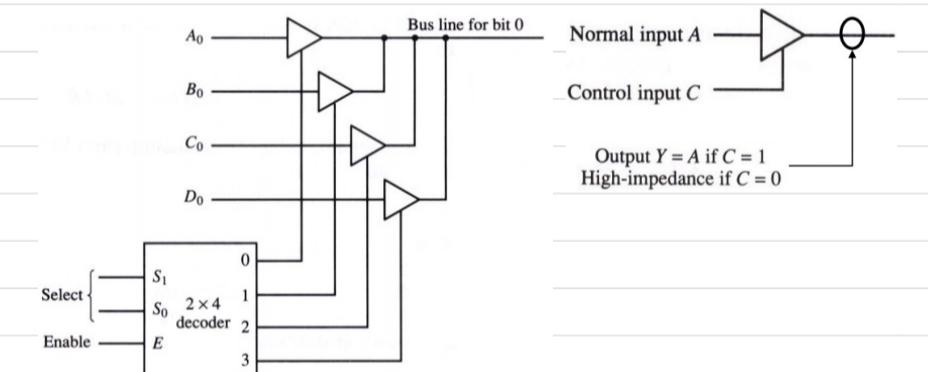


ii. Two ways of constructing a common bus system

i. Multiplexer method



ii. 3 state buffer



3. 算术 (Arithmetic) 마이크로 연산

- four categories of microoperation: 전송 연산, 산술 연산, 논리 연산, 시프트 연산

i. 산술연산

- addition, subtraction, increment, decrement, shift

$R3 \leftarrow R1 + R2$: 더하기 후 R3로 전송

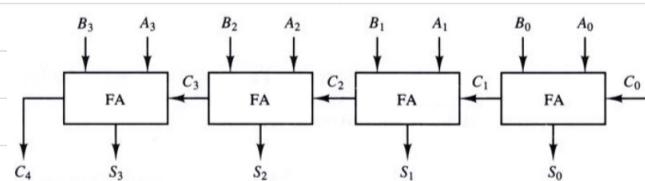
$R3 \leftarrow R1 - R2 (= R1 + \bar{R2} + 1)$: 빼기 후 R3로 전송

$R2 \leftarrow \bar{R2} + 1$: 2의 보수

- 곱셈, 나눗셈은 add, subtraction, shift 풀집합 구현

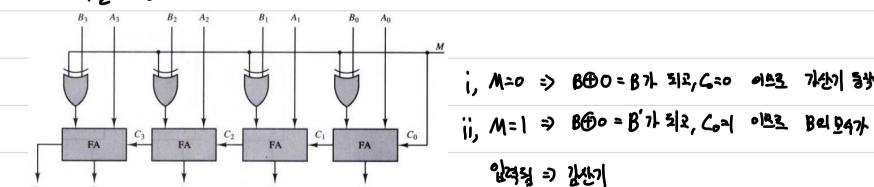
ii. 이진 가감산기

- 4비트 이진 가산기를 위해 4개의 FA(Full Adder)를 연결해서 구성



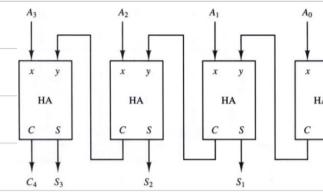
iii. 이진 가감산기

- 2의 보수를 이용한 턱셈

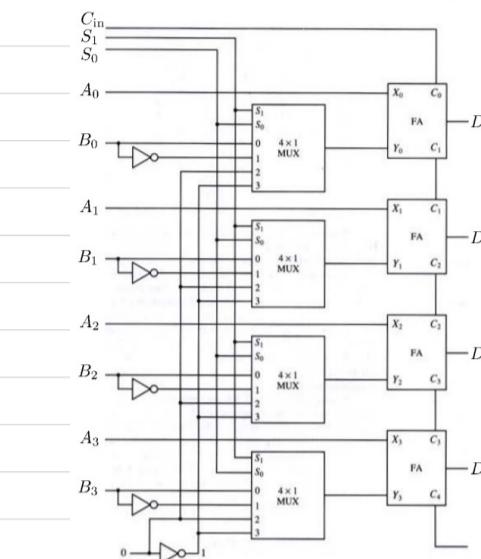


ii. binary incrementor

레지스터 간의 1 더함



iv. 산술회로



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

모든 비트가 1인 값은 1비트만 0인 값의 보수이다. 예상 (?)

4. 논리 마이크로 연산

- 레지스터에 저장된 비트열에 대한 이진 연산, 각 비트를 독립된 이진 변수로 간주
- 비트 연산 (OR, AND, Complement)와 구현을 위해 $OR \rightarrow V$, $AND \rightarrow 1$, $complement \rightarrow bar$ 3 표시
- + 기호가 제어 할수록 사용될 때는 OR 연산자를 네트워크

$P+Q : R1 \leftarrow R2+R3$, $R4 \leftarrow R5VR6$

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

가비트를 독립된 이진 변수로 간주해야함

하드웨어 구현: 각 비트마다 논리 게이트가 필요하며, 단지 논리학 (OR, AND, Complement, XOR)만 사용

· 응용: i, Select-set : B의 1이 해당하는 부울의 애스터 A의 비트를 1로 세트

$A \leftarrow AVB$ 와 같음

$\begin{array}{c} 1010 \\ 1100 \end{array}$ A before

$\begin{array}{c} 1110 \\ 1110 \end{array}$ A after

ii, select-complement : B의 1이 해당하는 부울의 애스터 A의 비트를 모두 0

$A \leftarrow A \oplus B$ 와 같음

$\begin{array}{c} 1010 \\ 1100 \end{array}$ B before

$\begin{array}{c} 0110 \\ 0110 \end{array}$ B after

iii, select-clear : B의 1이 해당하는 부울의 애스터 A의 비트를 모두 0

$A \leftarrow A \wedge B$ 와 같음

$\begin{array}{c} 1010 \\ 1100 \end{array}$ B before

$\begin{array}{c} 1000 \\ 1000 \end{array}$ B after

iv, select-mask : B의 1이 해당하는 부울의 애스터 A의 비트를 모두 1

$A \leftarrow A \vee B$ 와 같음

$\begin{array}{c} 1010 \\ 1100 \end{array}$ B before

$\begin{array}{c} 1000 \\ 1000 \end{array}$ B after

v, insert : 원하는 위치 비트를 마스크 시킨다. 원하는 값은 초기화

$A \leftarrow A \wedge B$

$A \leftarrow A \vee B$

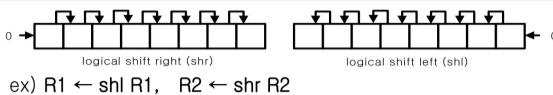
$A \leftarrow A \oplus B</$

5. 시프트 연산

- 비트들이 시프트될때 첫번째 FF은 직렬 입력을 통해 새로운 이진 정보를 받아드림

직렬 입력 정보에 따라 나뉨

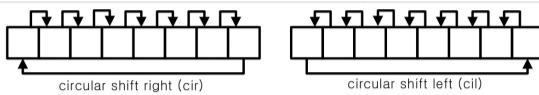
i, 논리 시프트 (shl, shr) : 입력 = 0



ex) $R1 \leftarrow \text{shl } R1$, $R2 \leftarrow \text{shr } R2$

10101110 → 01010111 (shr operation)

ii) 순환시프트 (crl, cir) : 훌역 → 입역, 손실 없음

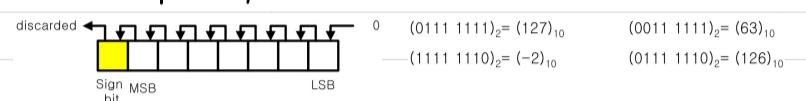


iii, 산술 시프트 : 부호가 있는 이진수를 시프트, sign bit is unchanged

- Shift right : division by 2



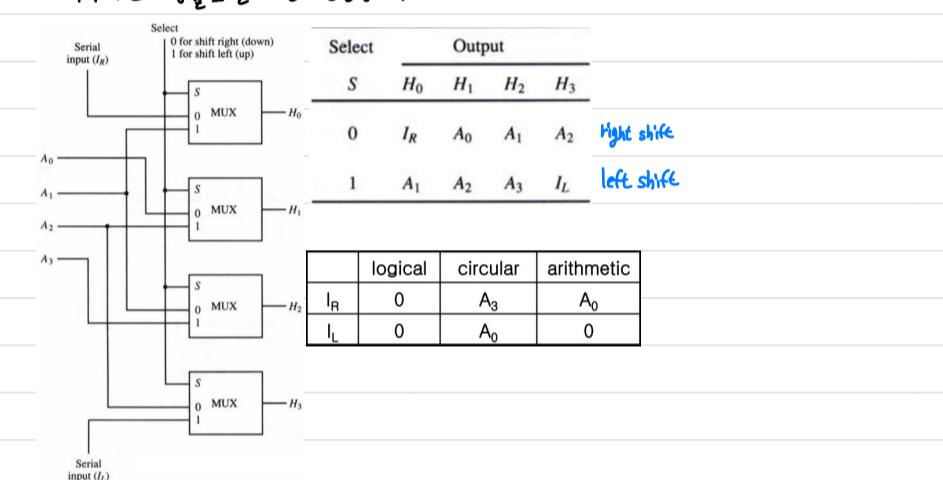
- shift-left: multiplication by 2



이때 R_{n-1} 과 R_{n+2} 가 다른 값이었다면 부호 비트의 분산이 발생 \Rightarrow 레이스터 값에 2를 곱하여 오버플로가 일어날

$$\text{Ehrl: } V_s = R_{n_1} \oplus R_{n_2}$$

- 하드웨어 구현: 별도의 키드를 가진 양방향 레지스터



I. Instruction code: 명령어

I, program: 사용자가 컴퓨터를 control하는 코드

- set of instructions that specify the operations(작업), operands(작업상수) and the sequence

II, Computer instruction(컴퓨터 명령):

- binary codes (specifies a sequence of Microoperations)
- computer read each instruction from memory, and place it in a control register
- and control interprets and proceeds to execute by issuing a sequence of microoperations.

III, instruction code (명령어코드):

- A group of bits that tells computer to perform a specific operation
- operation code part is a group of bits that define such operations (add, subtract...)

IV, Operation(작업):

VS Microoperation

- control unit receives instruction from memory \rightarrow interprets operation code part
- \Rightarrow issues a sequence of control signals to initiate microoperations of specified operation
- sometimes called a macrooperation

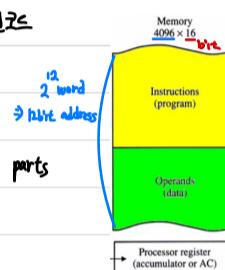
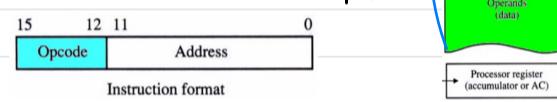
V, address part in instruction code

- instruction code also must specify registers or the memory words where the operands are to be found as well as registers or the memory words where the result is to be stored
- Memory word: specified by address

Processor register: 2개의 메모리 주소에 해당하는 4비트를 위한 16비트의 이진코드

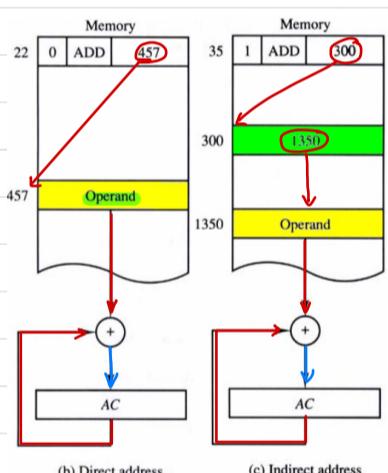
VI, stored program organization(제작):

- One processor register + an instruction code format with two parts



V, indirect address

두번의 메모리 주소



immediate operand: specified operand itself

2. Computer register

computer needs processor registers for manipulating data and a register for holding a memory address

I, 데이터 레지스터 (DR): 16bits, 메모리에서 읽어온 피연산자 저장

II, 누산기 (AC): 16bits, 복잡 계산 레지스터

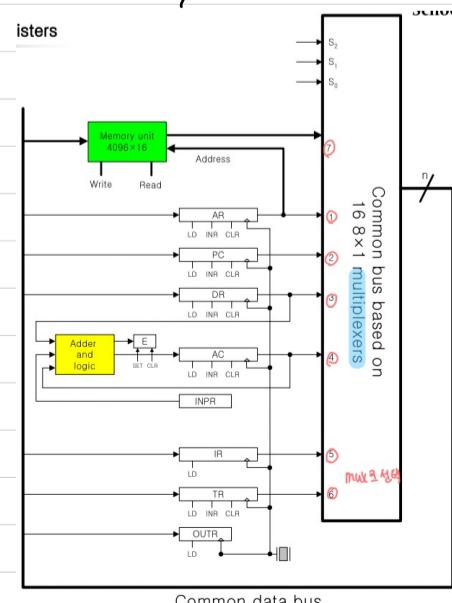
III, 명령어 레지스터 (IR): 16bits, 메모리에서 읽어온 명령어 저장

IV, 임시 레지스터 (TR): 16bits, 계산 도중의 임시 데이터 저장

V, 메모리 주소 레지스터 (AR): 12bits, 메모리 주소 저장

VI, 프로그램 카운터 (PC): 12bits, 명령어 주소 저장

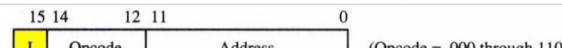
Common bus system



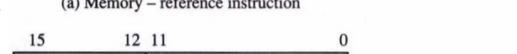
- 8 registers + Memory + control unit
- specific output is selected by a MUX
- 메모리: 쓰기 입력 활성화 - 8비트의 내용 반영
- 입기 입력 활성화 - 16비트 출력을 버스에 올려놓기
- $DR \leftarrow AC, AC \leftarrow DR$
- $S_2, S_1 = 100$ 일때 AC 내용을 버스 위에 올려놓기
- DR의 LD 입력을 인레이블 + 동일한 클럭동안 DR의 내용을 가산기의 차도를 통해 AC에 넣기

3. Computer Instructions

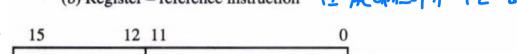
I, 3비트로 구성된 연산 코드가 통해서 나머지 15비트의 의미가 결정된다.



(a) Memory – reference instruction



(b) Register – reference instruction 주로 AC 레지스터에 대한 연산자정



(c) Input – output instruction

Hexadecimal code

Symbol	$I = 0$	$I = 1$	Description
AND	0XXX	8XXX	AND memory word to AC
ADD	1XXX	9XXX	Add memory word to AC
LDA	2XXX	AXXX	Load memory word to AC
STA	3XXX	BXXX	Store content of AC in memory
BUN	4XXX	CXXX	Branch unconditionally
BSA	5XXX	DXXX	Branch and save return address
ISZ	6XXX	EXXX	Increment and skip if zero
CLA	7800		Clear AC
CLE	7400		Clear E
CMA	7200		Complement AC
CME	7100		Complement E
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instruction if AC positive
SNA	7008		Skip next instruction if AC negative
SZA	7004		Skip next instruction if AC zero
SZE	7002		Skip next instruction if E is 0
HLT	7001		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

메모리에 대한 접근은 필요 없고, 나머지 12비트를 이용해 연산의 종류를 나타내는 사용

4. Timing and Control

- two major type of control organization.

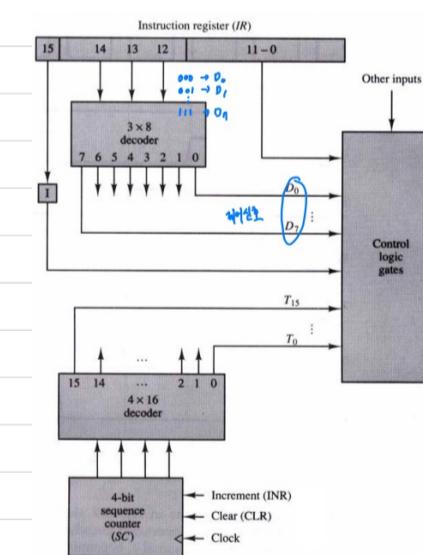
I, Hardwired control

- 부록 디자인 회로를 이용하여 제어회로 구현
- 장점: 비싸움, 단점: Design change \rightarrow Wiring change

II, Microprogrammed control

- 메모리에 저장된 제어 정보를 이용하여 마이크로 연산을 순차적으로 수행
- 장점: 솔루션 변경되면 메모리를 초기화하면 됨, 단점: slow

III, 제어 장치 흐름도

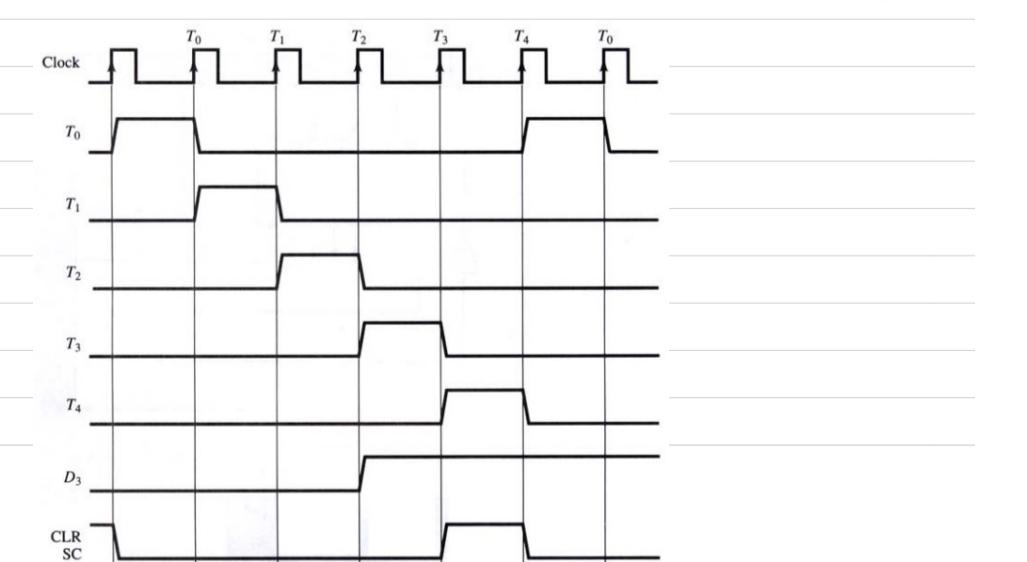


control logic gates I/O

Inputs to Control Logic Circuit

- Outputs of 2 decoders, 1 flip-flop, IR(0~11)
- AC(0~15) to check if $AC=0$
- DR(0~15) to check if $DR=0$
- the values of 7 flip-flops : I, S, E, R, IEN, FGI, FGO

- ① Signals to control the inputs of the nine registers
- ② Signals to control the read and write inputs of memory
- ③ Signals to set, clear, or complement the flip-flops
- ④ Signals for S_2, S_1, S_0 to select a register for the bus
- ⑤ Signals to control the AC adder and logic circuit



5. instruction cycle

T_1 , 명령어가 실행되기 위한 네 단계

i, 명령어를 메모리에서 가져온다 (fetch)

ii, 명령어를 파악한다 (decode)

iii, 간접 주소일 경우 메모리로부터 유효 주소를 얻어낸다 $AR \leftarrow M[AR]$

iv, 명령어 실행

II, fetch and decode

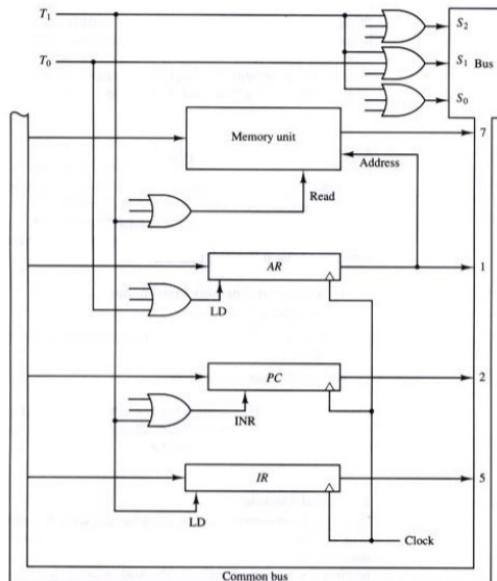
· PC는 첫 명령어 주소를 갖고 있음

· SC는 초기에 0으로 T_0 를 기록하고 매 클럭마다 T_0, T_1, T_2 순으로 변함, 이 타이밍 변수에 맞춰 다음의 단계 실행

i, $T_0: AR \leftarrow PC$ · AR 만 메모리의 주소 입력에 연결

· $S_0, S_1 = 0$ 하여 PC 내용이 버스에 놓임, AR의 LD를 enable 하여 버스의 내용을 AR로 전송

· 다음 클럭 사이에서 PC와 AR 사이에 데이터 전송 일어남



ii, $T_1: IR \leftarrow M[AR], PC \leftarrow PC + 1$ · 메모리에서 얻어온 명령어를 IR에 저장

1, Memory Read enable

2, $S_0, S_1 = 1$ 하여 메모리의 내용이 버스에 놓임

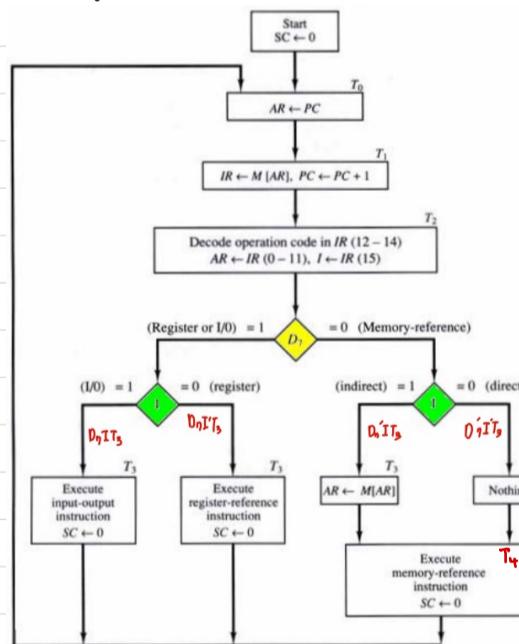
3, IR의 LD 입력을 enable 하여 버스의 내용을 IR로 전송

4, PC의 INR 입력을 enable 하여 PC의 값 1 증가

iii, $T_2: D_0 \dots D_7 \leftarrow \text{Decode } IR(2-4), AR \leftarrow IR(0-11), I \leftarrow IR(15)$

· IR의 연산 코드 내용이 다르게 되고, 간접 주소가 I에 전해지며, 주소는 AR로 전송

III, Determine type of instruction (at T_3)



i, $D_7'IT_3: AR \leftarrow M[AR]$, 메모리에서 유효 주소를 얻어온다

ii, $D_7'IT_3:$ 아무런 일 X, 실제로 명령어가 수행되는

시점인 T_4 타이밍 변수를 얻어 주면서 실행 차용하는

증가 동작 수행

iii, $D_7'IT_3:$ 해석의 차로 명령어 수행

iv, $D_7'IT_3:$ 입출력 명령어 수행

IV, 랙스터 칩을 통한 명령어

· IR(0-11)이 있는 4비트 12비트 12비트 명령어 4비트, T_3 에서 속행됨

$D_7'IT_3 = r$ (common to all register-reference instructions)

$IR(i) = B_i$ [bit in IR(0-11) that specifies the operation]

CLA	$rB_{11}: AC \leftarrow 0$	Clear SC
CLE	$rB_{10}: E \leftarrow 0$	Clear AC
CMA	$rB_9: AC \leftarrow \overline{AC}$	Clear E
CME	$rB_8: E \leftarrow \overline{E}$	Complement AC
CIR	$rB_7: AC \leftarrow \text{shr } AC, AC(15) \leftarrow E, E \leftarrow AC(0)$	Complement E
CIL	$rB_6: AC \leftarrow \text{shl } AC, AC(0) \leftarrow E, E \leftarrow AC(15)$	Circulate right
INC	$rB_5: AC \leftarrow AC + 1$	Circulate left
SPA	$rB_4: \text{If } (AC(15) = 0) \text{ then } (PC \leftarrow PC + 1)$	Increment AC
SNA	$rB_3: \text{If } (AC(15) = 1) \text{ then } (PC \leftarrow PC + 1)$	Skip if positive
SZA	$rB_2: \text{If } (AC = 0) \text{ then } PC \leftarrow PC + 1$	Skip if negative
SZE	$rB_1: \text{If } (E = 0) \text{ then } (PC \leftarrow PC + 1)$	Skip if AC zero
HLT	$rB_0: S \leftarrow 0$ (S is a start-stop flip-flop)	Skip if E zero

	Clear SC
	Clear AC
	Clear E
	Complement AC
	Complement E
	Circulate right
	Circulate left
	Increment AC
	Skip if positive
	Skip if negative
	Skip if AC zero
	Skip if E zero
	Halt computer

6. Memory Reference Instructions

· 유효 주소는 $T_2(I=0)$ or $T_3(I=1)$ 이 AR로 전송

· 명령어 수행은 T_4 에서 시작

i, AND to AC : AC와 유효 주소로 지정된 메모리 위치의 각 비트상에 대해 AND 논리 연산 수행 후 AC로 전송

· $D_6T_4: DR \leftarrow M[AR]$

$D_7T_5: AC \leftarrow AC \wedge DR, SC \leftarrow 0$

ii, ADD to AC : $D_6T_4: DR \leftarrow M[AR]$

$D_7T_5: AC \leftarrow AC + DR, E \leftarrow \text{Carry}, SC \leftarrow 0$

iii, LDA (Load to AC) : $D_6T_4: DR \leftarrow M[AR]$

$D_7T_5: AC \leftarrow DR, SC \leftarrow 0$

· 버스 시스템에서는 지정된 맵기위하여 버스에서 AC 값을 만들기 어렵다

iv, STA (Store AC) : AC의 내용을 유효 주소로 지정된 메모리 위치에 전송

· $D_6T_4: M[AR] \leftarrow AC, SC \leftarrow 0$

v, BUN : 프로그램 수행을 유효 주소로 지정된 명령어로 옮겨간다.

· $D_6T_4: PC \leftarrow AR, SC \leftarrow 0$

(a) Memory, PC, and AR at time T_4

(b) Memory and PC after execution

vi, BSA (Branch and Save return address)

· $D_6T_4: M[AR] \leftarrow PC, AR \leftarrow AR + 1$

$D_7T_5: PC \leftarrow AR, SC \leftarrow 0$

\bar{X}, ISZ (increment and skip if zero)

· $D_6T_4: DR \leftarrow M[AR]$

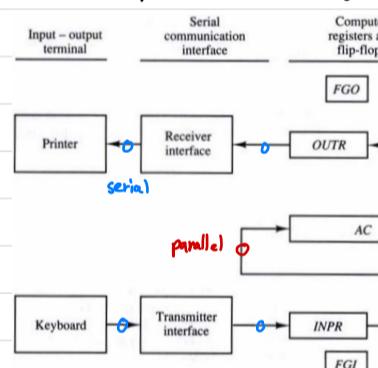
$D_7T_5: DR \leftarrow DR + 1$

$D_8T_6: M[AR] \leftarrow DR, \text{if } (DR = 0) \text{ then } (PC \leftarrow PC + 1), SC \leftarrow 0$

7. I/O and Interrupt

7.1 I/O

· each quantity of information has eight bits of an alphanumeric code



i, input procedure

initially, $FGI = 0 \rightarrow$ 키를 입력하고 8비트가 INPR로 shift
 $\rightarrow FGI = 1$ set (set 동안 INPR 정보를 보존 X) \rightarrow 컴퓨터가 FGI를
 채크하여 그 값이 1이면 INPR \rightarrow AC 정보 전달 \rightarrow FGI clear

ii, Output procedure

initially, $FGO = 1 \rightarrow$ 컴퓨터 치고, 1이면 AC \rightarrow OUTR 정보 전달
 \rightarrow FGO clear \rightarrow 출력장치가 정보가져감 \rightarrow FGO set

II, I/O 명령어

· AC로 값을 전달하고, 플래그 비트를 검사하여, 인터럽트 처리

$D_7IT_3 = p$ (common to all input-output instructions)

$IR(i) = B_i$ [bit in IR(6-11) that specifies the instruction]

$p: SC \leftarrow 0$	Clear SC
$pB_{11}: AC(0-7) \leftarrow INPR, FGI \leftarrow 0$	Input character
$pB_{10}: OUTR \leftarrow AC(0-7), FGO \leftarrow 0$	Output character
$pB_9: \text{If } (FGI = 1) \text{ then } (PC \leftarrow PC + 1)$	Skip on input flag
$pB_8: \text{If } (FGO = 1) \text{ then } (PC \leftarrow PC + 1)$	Skip on output flag
$pB_7: IEN \leftarrow 1$	Interrupt enable on
$pB_6: IEN \leftarrow 0$	Interrupt enable off

여기서 진단된 명령어는 이전 명령어로 돌아와
 다시 플래그를 검사하도록 하는 보기 명령어.
 \Rightarrow 플래그가 0이면 / 철예 까지 계속 됨.

IV, 랙스터 칩을 통한 명령어

· IR(0-11)이 있는 4비트 12비트 12비트 명령어 4비트, T_3 에서 속행됨

$D_7IT_3 = r$ (common to all register-reference instructions)

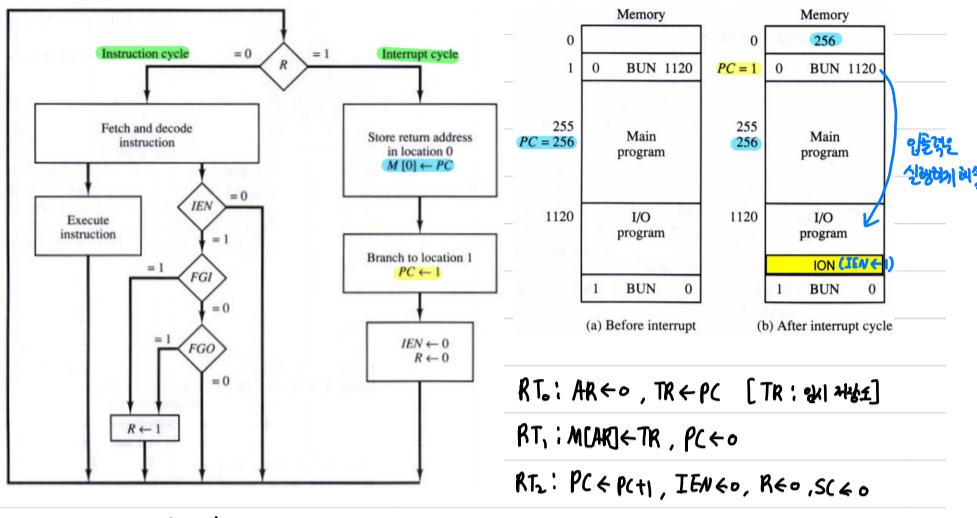
$IR(i) = B_i$ [bit in IR(0-11) that specifies the operation]

CLA	$rB_{11}: AC \leftarrow 0$	Clear SC
CLE	$rB_{10}: E \leftarrow 0$	Clear AC
CMA	$rB_9: AC \leftarrow \overline{AC}$	Clear E
CME	$rB_8: E \leftarrow \overline{E}$	Complement AC
CIR	$rB_7: AC \leftarrow \text{shr } AC, AC(15) \leftarrow E, E \leftarrow AC(0)$	Complement E
CIL	$rB_6: AC \leftarrow \text{shl } AC, AC(0) \leftarrow E, E \leftarrow AC(15)$	Circulate right
INC	$rB_5: AC \leftarrow AC + 1$	Circulate left
SPA	$rB_4: \text{If } (AC(15) = 0) \text{ then } (PC \leftarrow PC + 1)$	Increment AC
SNA	$rB_3: \text{If } (AC(15) = 1) \text{ then } (PC \leftarrow PC + 1)$	Skip if positive
SZA	$rB_2: \text{If } (AC = 0) \text{ then } PC \leftarrow PC + 1$	Skip if negative
SZE	$rB_1: \text{If } (E = 0) \text{ then } (PC \leftarrow PC + 1)$	Skip if AC zero
HLT	$rB_0: S \leftarrow 0$ (S is a start-stop flip-flop)	Skip if E zero

	Clear SC
	Clear AC
	Clear E
	Complement AC
	Complement E
	Circulate right
	Circulate left
	Increment AC
	Skip if positive
	Skip if negative
	Skip if AC zero
	Skip if E zero
	Halt computer

III, Interrupt cycle

- 증치를 사용한 통신 방법(제어 전송)은 프로세서와 일련의 상호 송수 차이로 인해 번들링된다.
- 인터럽트: 외부 장치가 전송 준비가 되어 있을 때 컴퓨터에게 알림
- 플러그가 씨스 되면 컴퓨터는 현재 실행하고 있는 프로그램을 중지하고 일련을 실행
- IEN (interrupt enable FF): IEN=1 set 일련은 인터럽트 가능



8. Complete Computer Description

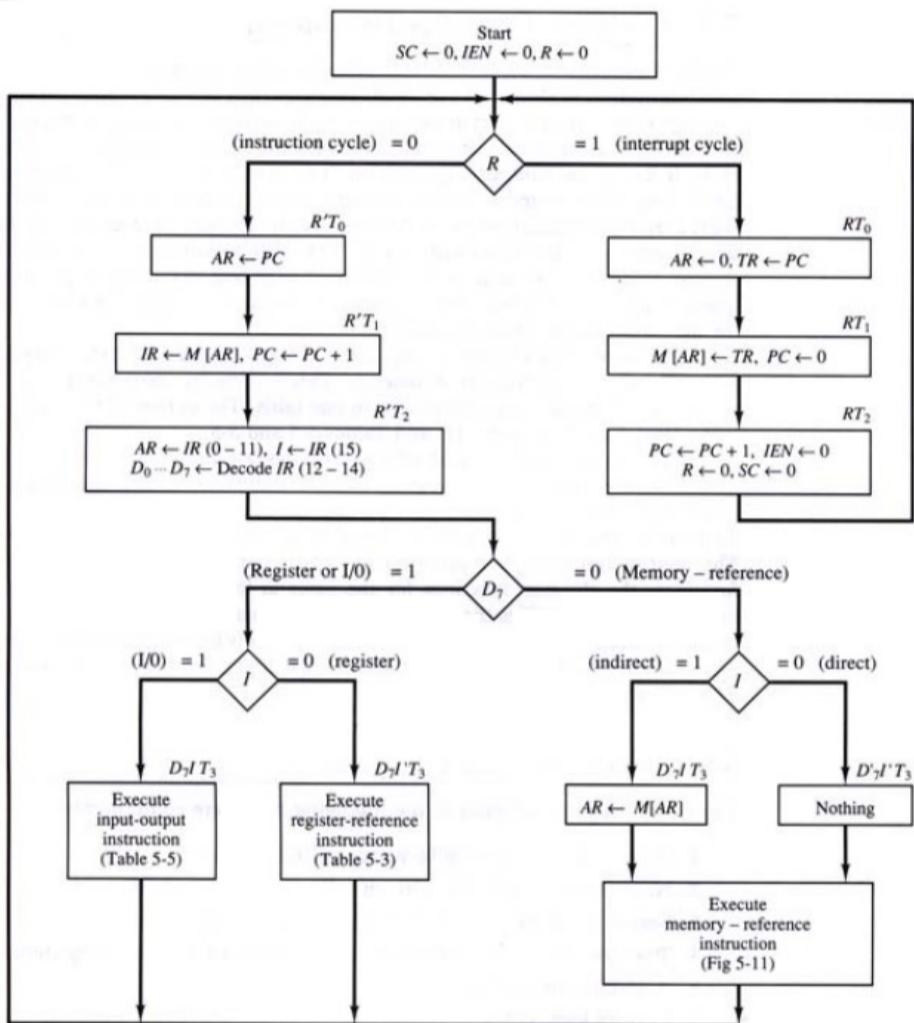


Figure 5-15 Flowchart for computer operation.

9. Design of Basic Computer

I, Control of Register and Memory

레지스터 차이 일련에는 LD(로드), INR(인크리먼트), CLR(클리어)가 연결되어 있음

• 표에서 AR 내용은 번경시에는 모든 문장을 찾아야 함

$R'T_0: AR \leftarrow PC$

$R'T_2: AR \leftarrow IR(0\sim11)$

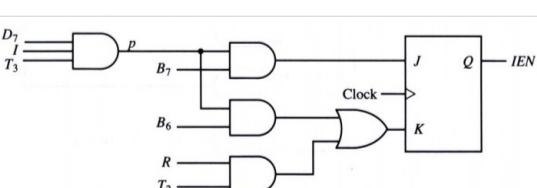
$D'_7IT_3: AR \leftarrow M[AR] \Rightarrow LD(AR) = R'T_0 + R'T_2 + D'_7IT_3$

$RT_0: AR \leftarrow 0$

$RT_4: AR \leftarrow AR + 1$

II, Control of Single FFs

ex) IEN control



$pB_7: IEN \leftarrow 1$

$pB_6: IEN \leftarrow 0 \Rightarrow$

$RT_2: IEN \leftarrow 0$

III, Control of Common bus

Inputs	Outputs	Register selected for bus
$x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ S_2 \ S_1 \ S_0$		None
0 0 0 0 0 0 0 0 0 0 0 0		AR
1 0 0 0 0 0 0 0 0 0 0 1		PC
0 1 0 0 0 0 0 0 0 1 0		DR
0 0 1 0 0 0 0 0 1 0 0		AC
0 0 0 1 0 0 0 1 0 0 1		IR
0 0 0 0 1 0 0 1 0 1 0		TR
0 0 0 0 0 1 1 1 1 1 1		Memory

$S_0 = X_1 + X_3 + X_5 + X_7$

$S_1 = X_2 + X_3 + X_6 + X_7$

$S_2 = X_4 + X_3 + X_6 + X_7$

인려 일련에 대한 개별 논리는 해당 메시스터로 각각 논하는 경우

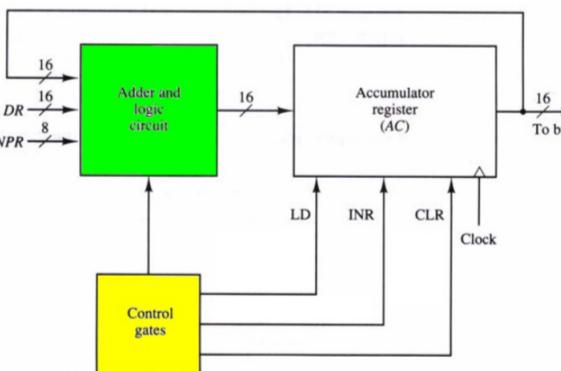
ex) $x_i = [AR \rightarrow BUS]$

$D_4T_4: PC \leftarrow AR, D_5T_5: PC \leftarrow AR \Rightarrow x_i = D_4T_4 + D_5T_5$

TABLE 5-6 Control Functions and Microoperations for the Basic Computer

Fetch	$R'T_0: AR \leftarrow PC$
Decode	$R'T_1: IR \leftarrow M[AR], PC \leftarrow PC + 1$
Indirect	$R'T_2: D_0, \dots, D_{11} \leftarrow \text{Decode } IR(12-14), AR \leftarrow IR(0-11), I \leftarrow IR(15)$
Interrupt:	$D'_7IT_3: AR \leftarrow M[AR]$
	$T_0 T_1 T_2 (IEN)(FGI + FGO): R \leftarrow 1$
	$RT_0: AR \leftarrow 0, TR \leftarrow PC$
	$RT_1: M[AR] \leftarrow TR, PC \leftarrow 0$
	$RT_2: PC \leftarrow PC + 1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$
Memory-reference:	
AND	$D_0T_4: DR \leftarrow M[AR]$
ADD	$D_0T_5: AC \leftarrow AC \wedge DR, SC \leftarrow 0$
LDA	$D_1T_4: DR \leftarrow M[AR]$
STA	$D_1T_5: AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$
BUN	$D_2T_4: DR \leftarrow M[AR]$
BSA	$D_2T_5: AC \leftarrow DR, SC \leftarrow 0$
ISZ	$D_3T_4: M[AR] \leftarrow AC, SC \leftarrow 0$
	$D_3T_5: PC \leftarrow AR, SC \leftarrow 0$
	$D_4T_4: M[AR] \leftarrow PC, AR \leftarrow AR + 1$
	$D_4T_5: PC \leftarrow AR, SC \leftarrow 0$
	$D_5T_4: DR \leftarrow M[AR]$
	$D_5T_5: DR \leftarrow DR + 1$
	$D_6T_4: M[AR] \leftarrow DR, \text{ if } (DR = 0) \text{ then } (PC \leftarrow PC + 1), SC \leftarrow 0$
Register-reference:	
	$D_7I'T_3 = r \text{ (common to all register-reference instructions)}$
	$IR(i) = B_i \ (i = 0, 1, 2, \dots, 11)$
CLA	$r: SC \leftarrow 0$
CLE	$rB_{11}: AC \leftarrow 0$
CMA	$rB_{10}: E \leftarrow 0$
CME	$rB_9: AC \leftarrow \overline{AC}$
CIR	$rB_8: E \leftarrow \overline{E}$
CIL	$rB_7: AC \leftarrow shr AC, AC(15) \leftarrow E, E \leftarrow AC(0)$
INC	$rB_6: AC \leftarrow shl AC, AC(0) \leftarrow E, E \leftarrow AC(15)$
SPA	$rB_5: AC \leftarrow AC + 1$
SNA	$rB_4: \text{If } (AC(15) = 0) \text{ then } (PC \leftarrow PC + 1)$
SZA	$rB_3: \text{If } (AC(15) = 1) \text{ then } (PC \leftarrow PC + 1)$
SZE	$rB_2: \text{If } (AC = 0) \text{ then } (PC \leftarrow PC + 1)$
HLT	$rB_1: \text{If } (E = 0) \text{ then } (PC \leftarrow PC + 1)$
	$rB_0: S \leftarrow 0$
Input-output:	
	$D_7IT_3 = p \text{ (common to all input-output instructions)}$
	$IR(i) = B_i \ (i = 6, 7, 8, 9, 10, 11)$
INP	$p: SC \leftarrow 0$
OUT	$pB_{11}: AC(0-7) \leftarrow INPR, FGI \leftarrow 0$
SKI	$pB_{10}: OUTR \leftarrow AC(0-7), FGO \leftarrow 0$
SKO	$pB_9: \text{If } (FGI = 1) \text{ then } (PC \leftarrow PC + 1)$
ION	$pB_8: \text{If } (FGO = 1) \text{ then } (PC \leftarrow PC + 1)$
IOF	$pB_7: IEN \leftarrow 1$
	$pB_6: IEN \leftarrow 0$

10. Design of Accumulator Logic



$D_0T_5: AC \leftarrow AC \wedge DR$

$D_1T_5: AC \leftarrow AC + DR$

$D_2T_5: AC \leftarrow DR$

$pB_{11}: AC(0-7) \leftarrow INPR$

$rB_9: AC \leftarrow AC'$

$rB_7: AC \leftarrow shr AC, AC(15) \leftarrow E$

$rB_6: AC \leftarrow shl AC, AC(0) \leftarrow E$

$rB_{11}: AC \leftarrow 0$

$rB_5: AC \leftarrow AC + 1$

AND with DR

Add with DR

Transfer from DR

Transfer from INPR

Complement

Shift right

Shift left

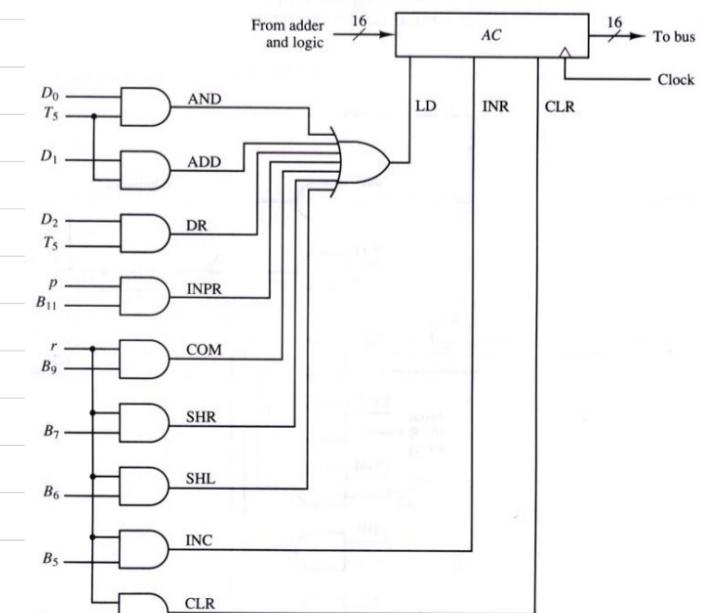
Clear

Increment

LD

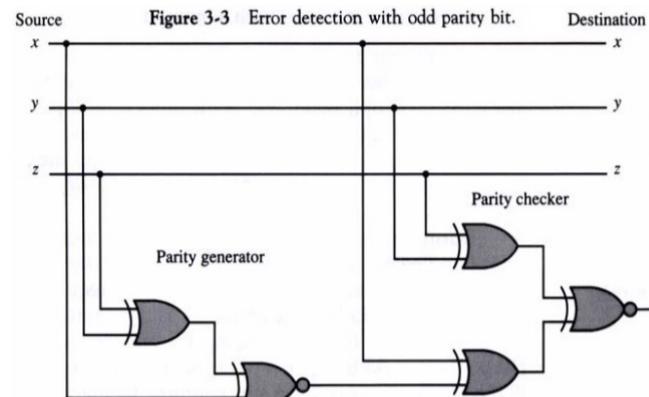
CLR

INR



Decimal digit	BCD 8421	2421	Excess-3	gray
0	0000	0000	0011	0010
1	0001	0001	0100	0110
2	0010	0010	0101	0111
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1100
6	0110	1100	1001	1101
7	0111	1101	1010	1111
8	1000	1110	1011	1110
9	1001	1111	1100	1010

Only 1 bit changes



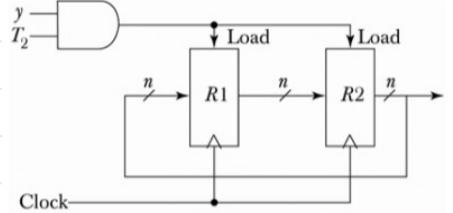
$$x \oplus y \oplus z = p$$

$$E = \overline{x} \oplus y \oplus z$$

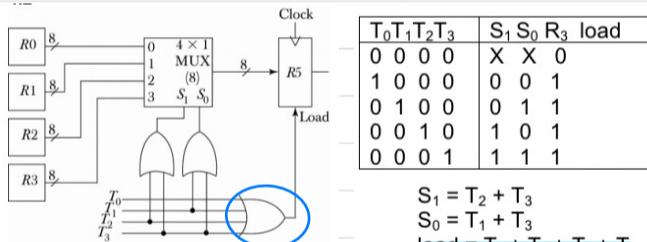
TABLE 3-7 Parity Bit Generation

Message xyz	P(odd)
000	1
001	0
010	0
011	1
100	0
101	1
110	1
111	0

4-1, $y_{T_2} : R_2 \leftarrow R_1, R_1 \leftarrow R_2$



4-2



$$S_1 = T_2 + T_3$$

$$S_0 = T_1 + T_3$$

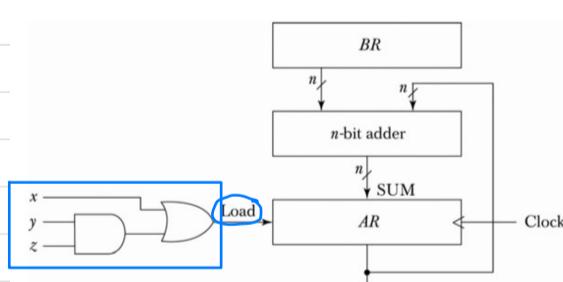
$$\text{load} = T_0 + T_1 + T_2 + T_3$$

4-6

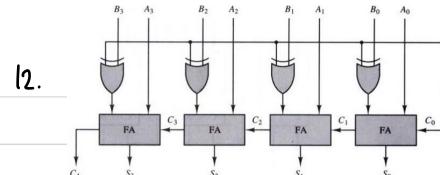
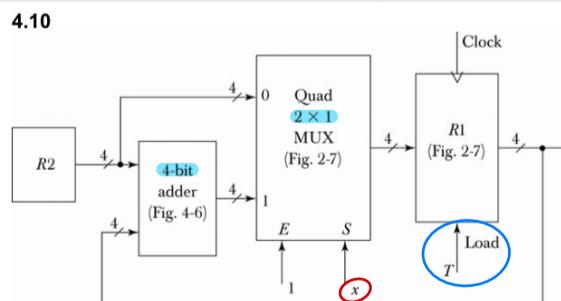
16 registers of 32 bits \Rightarrow 32 MUX size of (16×1)

\Rightarrow 4 selection lines

4-8,



4-11,

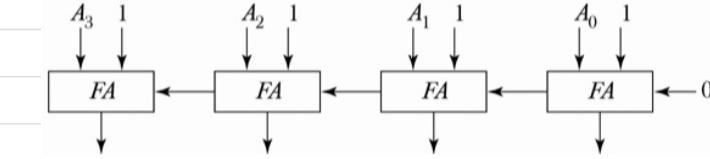


$$\begin{aligned} M &= 1 \oplus B_1 \oplus C_1 \\ C_1 &= M \oplus 1 \\ &= 1100 - (1000) \\ &= 1100 + (1000) \\ &= \frac{1100}{1000} \\ &= \frac{1000}{1000} \\ &= C_4 S_3 S_2 S_1 S_0 \end{aligned}$$

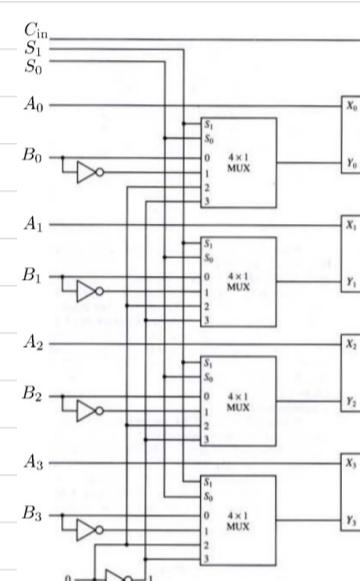
13, A-1 : decrementer

$$A + (1 \oplus 1 \oplus 1 \oplus 1) = A + (1111)$$

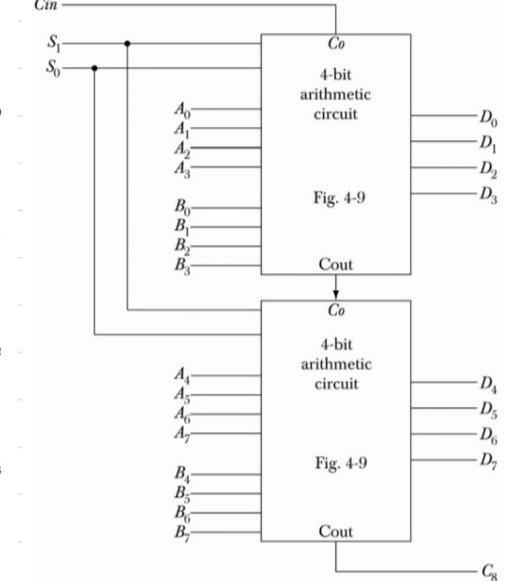
$$0001 \Rightarrow 1111$$



14.



4.14



Select	S ₁ S ₀ C _{in}	Input Y	Output D = A + Y + C _{in}	Microoperation
0 0 0	B	D = A + B	1	Add
0 0 1	B	D = A + B + 1	1	Add with carry
0 1 0	\bar{B}	D = A + \bar{B}	1	Subtract with borrow
0 1 1	\bar{B}	D = A + \bar{B} + 1	1	Subtract
1 0 0	0	D = A	1	Transfer A
1 0 1	0	D = A + 1	1	Increment A
1 1 0	1	D = A - 1	1	Decrement A
1 1 1	1	D = A	1	Transfer A

19, 713번 721번 1번 \rightarrow 2번 \rightarrow 3번, 페어링 X

4.19 (a) AR = 11110010

BR = 11111111 CR = 10111001 DR = 1110

1010

BR = 11111111 CR = 10111001 DR = 11101010

(b) CR = 10111001

DR = 11101010 (AND)

CR = 10101000

BR = 11111111

+1

BR = 00000000 AR = 11110001 DR = 11101010

(c) AR = 11110001 (-)

CR = 10101000

AR = 01001001; BR = 00000000; CR = 10101000; DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 11101010

BR = 00000000

AR = 11110001 DR = 1110101

5.3. 1: AR, 2: PC, 3: DR, 4: AC, 5: IR, 6: TR, 7: M[AR]

$O \leftarrow M[AR]$: Read
 $M[AR] \leftarrow O$: Write

5.5

a, $AR \leftarrow PC$ M[AR] \leftarrow AR IR $\leftarrow M[AR]$

b, $DR \leftarrow TR$ AC는 DR 동기화로 같은 값을
 $AC \leftarrow AC + DR$

c, $AC \leftarrow DR$, $DR \leftarrow AC$ AC는 DR 동기화로 같은 값을
 $AC \leftarrow AC + DR$ DR은 안바뀜
 $AC \leftarrow DR$, $DR \leftarrow AC$

5.6 (코드 공부)

5.9. 초기조건: $AC = A937$, $PC = 021$ (실행되면 1주기당), $E=1$ (코드 공부)

	E	AC	PC	AR	IR
Initial	1	A937	021	—	—
CLA	1	0000	022	800	7800
CLE	0	A937	022	400	7400
CMA	1	56C8	022	200	7200
CME	0	A937	022	100	7100
CIR	1	D49B	022	080	7080
CIL	1	526F	022	040	7040
INC	1	A938	022	020	7020
SPA	1	A937	022	010	7010
SNA	1	A937	022	008	7008
SZA	1	A937	022	004	7004
SZE	1	A937	022	002	7002
HLT	1	A937	022	001	7001

5.10 021: 0083, 083: 피연산자 B8F2, AC=A937 (코드 공부)
 피연산자

	PC	AR	DR	AC	IR
Initial	021	—	—	A937	—
AND	022	083	B8F2	A832	0083
ADD	022	083	B8F2	6229	1083
LDA	022	083	B8F2	B8F2	2083
STA	022	083	—	A937	3083
BUN	083	083	—	A937	4083
BSA	084	084	—	A937	5083
ISZ	022	083	B8F3	A937	6083

11. T_0 : $AR \leftarrow PC$, T_1 : $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$, T_2 : $I \leftarrow IAC(15)$, $AR \leftarrow IR(0 \sim 1)$, 메모리작업

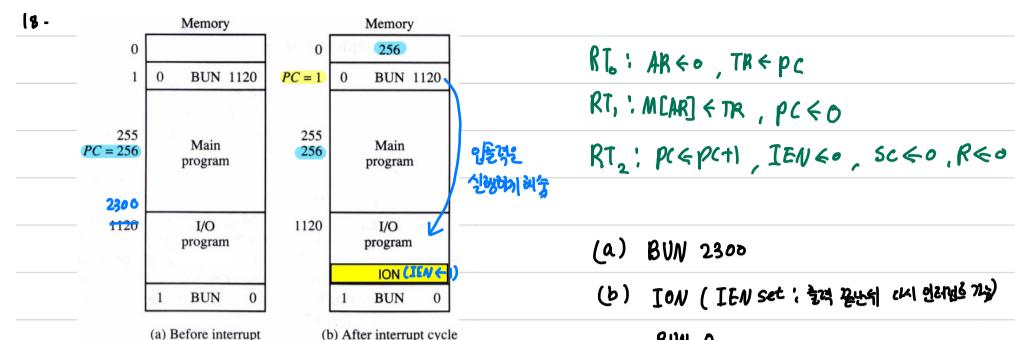
12. $AC = 1EC3$ Memory (코드 공부)
 $PC = 3AF$ 3AF 932E a, fetch; RT_0 : $AR \leftarrow PC$
 32E 09AC RT_1 : $IR \leftarrow M[AR]$
 9AC 8B9F $932E \Rightarrow ADD | 32E$
 $ADD \hookrightarrow 8B9F$

b, $AC = 1EC3$
 $DR = 8B9F$
 0A62, E4

c, $PC \leftarrow PC+1$ $AR = 3AF$
 $IR = 932E$ $DR = 8B9F$
 $SC \leftarrow 0$
 $AC = 0A62$

(코드 공부)

13. XOR	$D_0 T_4$:	$DR \leftarrow M[AR]$
	$D_0 T_5$:	$AC \leftarrow AC \oplus DR$, $SC \leftarrow 0$
ADM	$D_1 T_4$:	$DR \leftarrow M[AR]$
	$D_1 T_5$:	$DR \leftarrow AC$, $AC \leftarrow AC + DR$ AC 초기화
	$D_1 T_6$:	$M[AR] \leftarrow AC$, $AC \leftarrow DR$, $SC \leftarrow 0$
SUB	$D_2 T_4$:	$DR \leftarrow M[AR]$
	$D_2 T_5$:	$DR \leftarrow AC$, $AC \leftarrow DR$
	$D_2 T_6$:	$AC \leftarrow AC$ AC가 나중에 필요
	$D_2 T_7$:	$AC \leftarrow AC + 1$
	$D_2 T_8$:	$AC \leftarrow AC + DR$, $SC \leftarrow 0$
XCH	$D_3 T_4$:	$DR \leftarrow M[AR]$
	$D_3 T_5$:	$M[AR] \leftarrow AC$, $AC \leftarrow DR$, $SC \leftarrow 0$
SEQ	$D_4 T_4$:	$DR \leftarrow M[AR]$
	$D_4 T_5$:	$TR \leftarrow AC$, $AC \leftarrow AC \oplus DR$
	$D_4 T_6$:	If $(AC = 0)$ then $(PC \leftarrow PC + 1)$, $AC \leftarrow TR$, $SC \leftarrow 0$
BPA	$D_5 T_4$:	If $(AC < 0 \wedge AC(15) = 0)$ then $(PC \leftarrow AR)$, $SC \leftarrow 0$



(a) BUN 2300
 (b) ION (IEN set: 헤더 끝에서 대시 인터럽트 가능)

BUN 0

