



5. 기본 컴퓨터의 구조와 설계

목표

- 컴퓨터의 기본 동작 원리를 이해하고, 구현할 수 있음.



명령어 코드, INSTRUCTION CODE



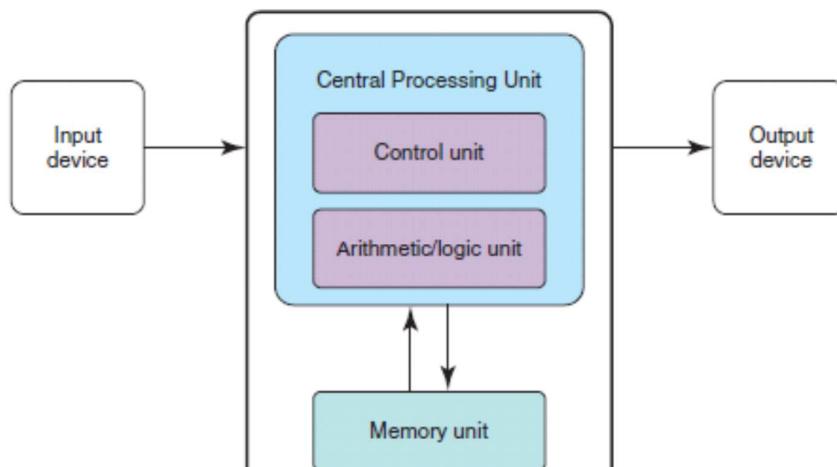
컴퓨터 구조 정의

- 레지스터 종류 및 기능
- 명령어 집합
 - 일련의 마이크로 연산들
- 마이크로 연산 수행을 위한 제어 및 타이밍



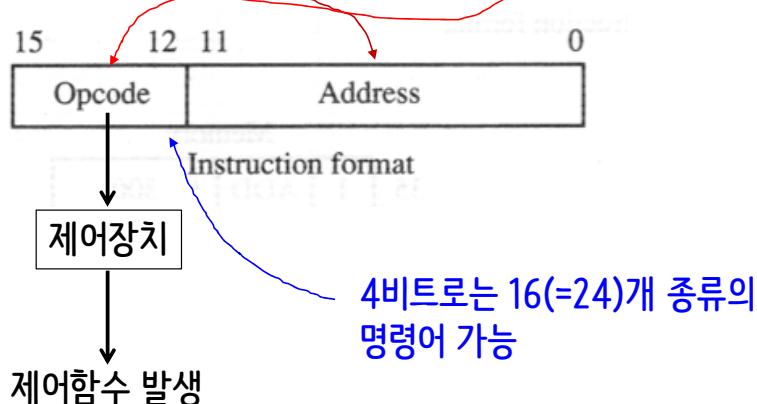
내장 프로그램 개념, Stored Program Concept

- Von Neumann Architecture
 - 기억장치에 데이터와 명령어가 함께 저장

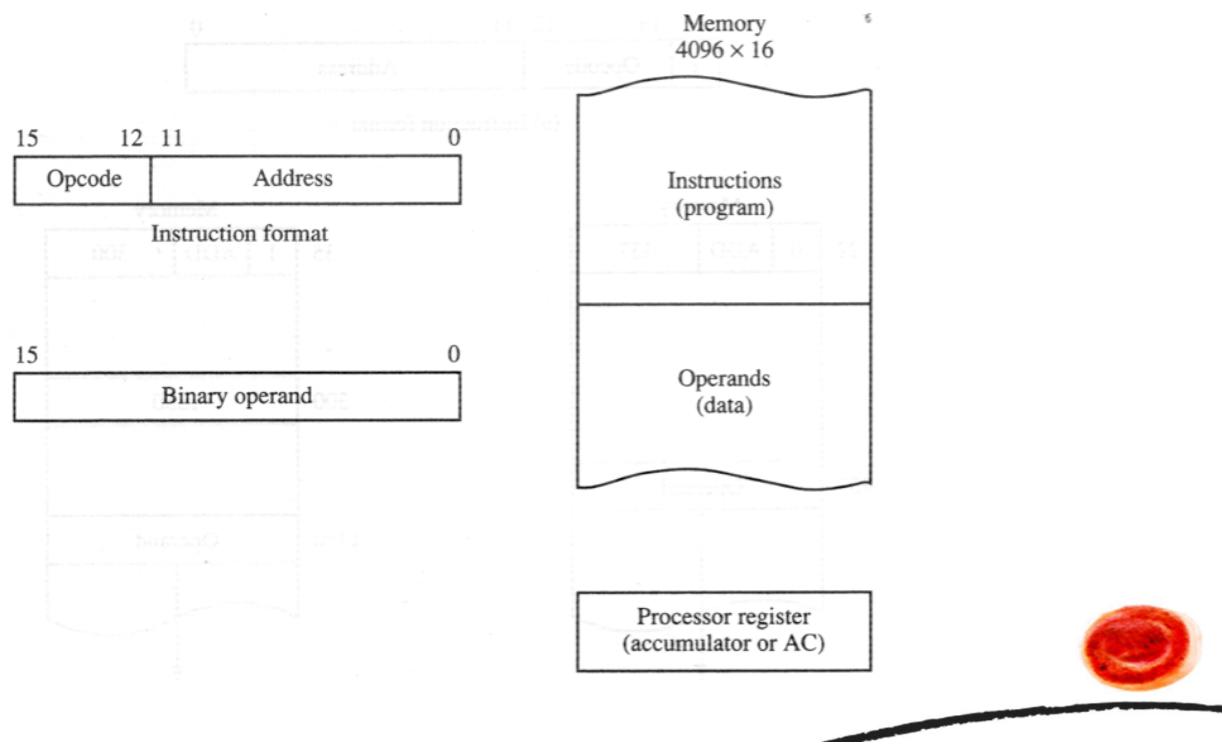


명령어

- 기능 수행을 나타내는 이진 코드(비트들의 집합)
- 일련의 마이크로 연산 수행
- 구성 :



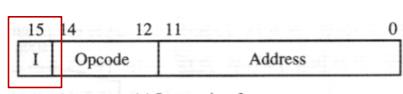
간단한 컴퓨터 구성



주소 모드, address mode

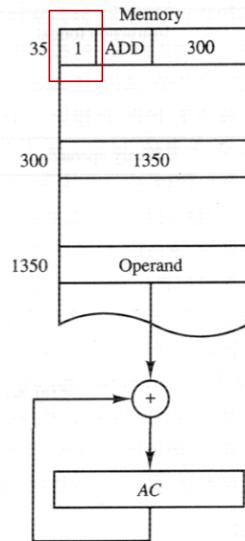
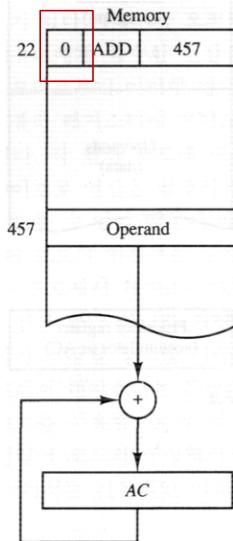
- **유효 주소(effective address) 결정 방법**
 - 명령어 'address' 필드 해석
- 종류
 - Immediate
 - 직접 주소, direct address
 - 간접 주소, indirect address

주소 모드의 예



모드 비트

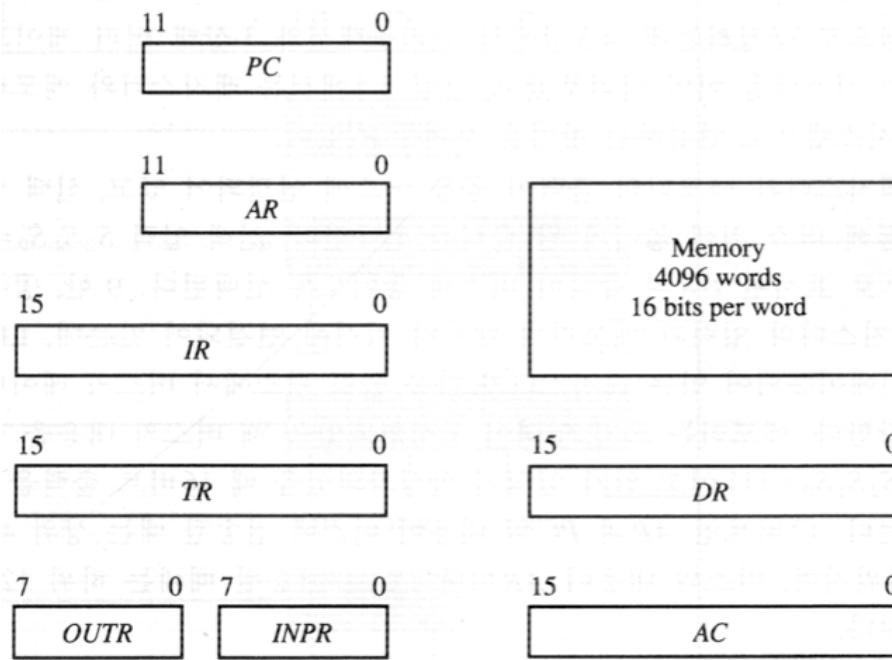
예를들어,
0 - 직접주소
1 - 간접주소



컴퓨터 레지스터



기본 컴퓨터의 레지스터



기본 컴퓨터의 레지스터

Register symbol	Number of bits	Register name	Function
<i>DR</i>	16	Data register	Holds memory operand
<i>AR</i>	12	Address register	Holds address for memory
<i>AC</i>	16	Accumulator	Processor register
<i>IR</i>	16	Instruction register	Holds instruction code
<i>PC</i>	12	Program counter	Holds address of instruction
<i>TR</i>	16	Temporary register	Holds temporary data
<i>INPR</i>	8	Input register	Holds input character
<i>OUTR</i>	8	Output register	Holds output character

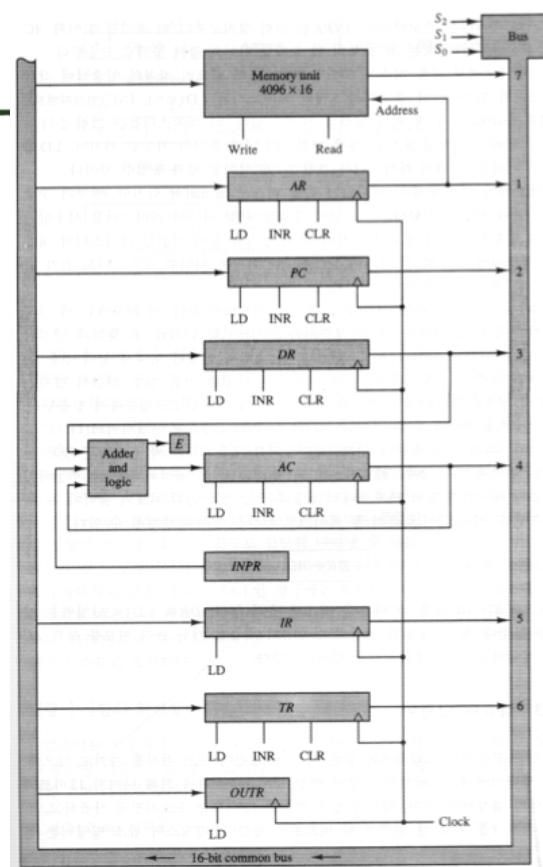
공통 버스 시스템

- 레지스터간 또는 레지스터와 메모리 간 정보 전송

- 구현

- 멀티플렉서 이용
- 3-상태 버퍼 게이트 이용 등…

기본 컴퓨터의 공통 버스



레지스터 제어 입력
LD - 적재
INC - 1증가
CLR - 0으로 함

컴퓨터 명령어, INSTRUCTION



기본 컴퓨터의 명령어 형식

15 14	12 11	0	
I	Opcode	Address	(Opcode = 000 through 110)

(a) Memory – reference instruction

15	12 11	0	
0 1 1 1	Register operation		(Opcode = 111, I = 0)

(b) Register – reference instruction

15	12 11	0	
1 1 1 1	I/O operation		(Opcode = 111, I = 1)

(c) Input – output instruction



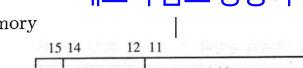
기본 컴퓨터 명령어 집합

- Instruction Set
- 실행할 수 있는 명령어들
- 컴퓨터가 제공하는 기본 기능

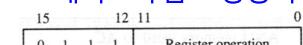
기본 컴퓨터 명령어 집합

Symbol	Hexadecimal code		Description
	I = 0	I = 1	
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

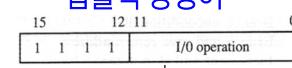
메모리 참조 명령어



레지스터 참조 명령어



입출력 명령어



명령어 집합의 완전성

- 데이터 처리에 ‘충분한’ 명령어를 제공

- 명령어 종류

- 산술, 논리, 시프트 명령어
- 메모리와 레지스터 간 데이터 전송 명령어
- 상태 검사 명령어, 프로그램 제어 명령어
- 입, 출력 명령어



타이밍과 제어



제어 장치

- 명령어 수행을 위한 제어 신호 생성

- 종류

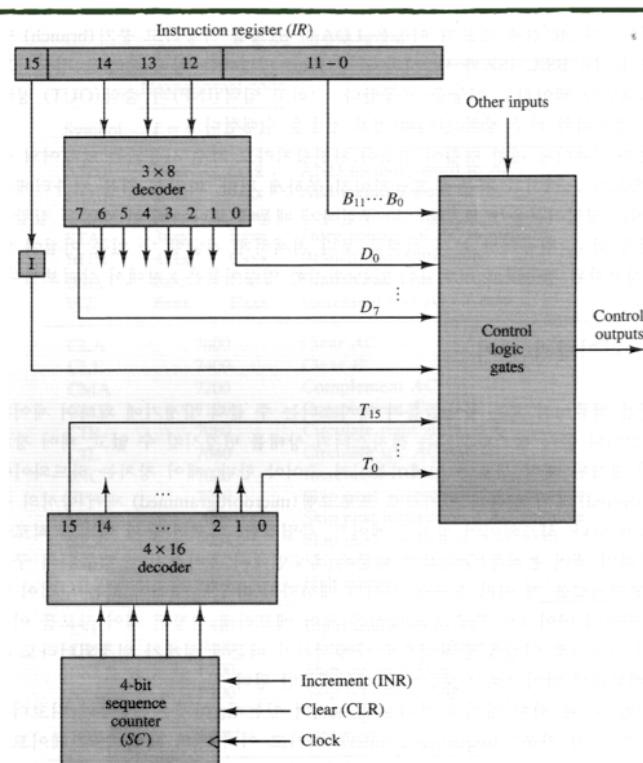
- Hardwired Control

- 게이트 수준에서 제어 논리 구현
- 고속/ 저유연성

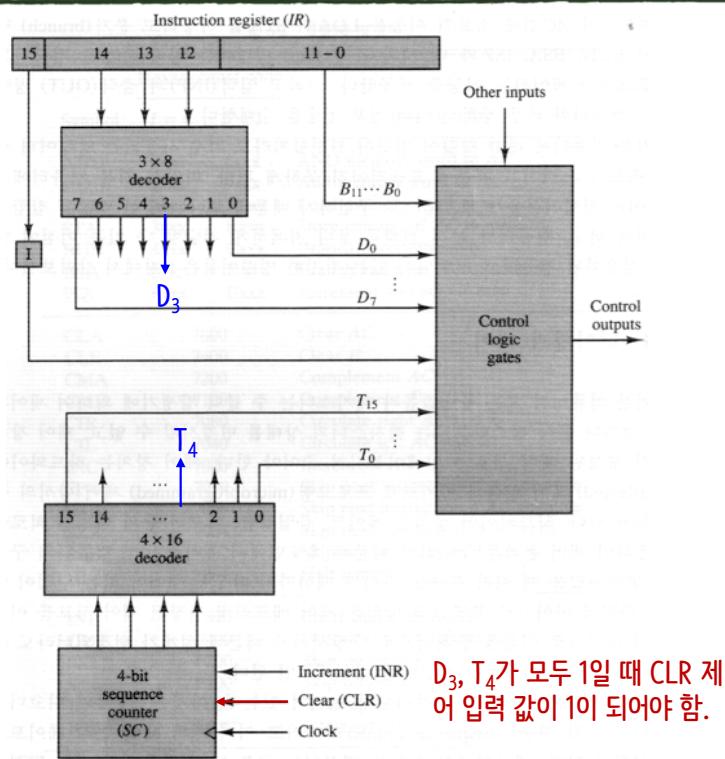
- Microprogrammed Control

- 제어 메모리에 저장된 제어 정보 이용
- 고유연성

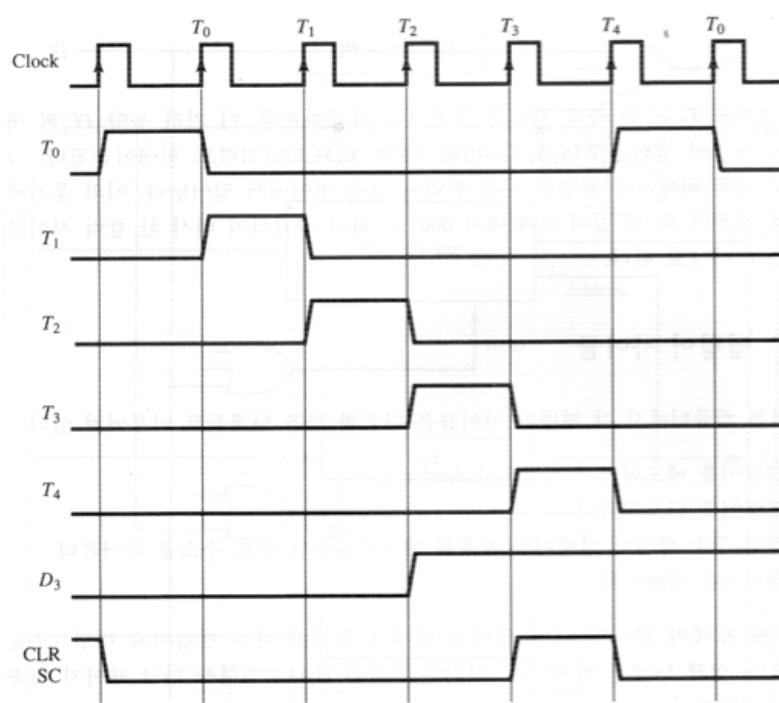
기본 컴퓨터의 제어 장치



기본 컴퓨터의 제어 장치 : $D_3T_4:SC \leftarrow 0$ 의 예



$D_3T_4:SC \leftarrow 0$ 의 제어 타이밍 예



명령어 사이클



명령어 사이클

- Instruction Cycle
- 한 명령어 실행을 위한 단계
 - 명령어 인출, Fetch
 - 명령어 해석
 - 간접 주소이면 유효 주소 인출
 - 명령어 실행



명령어 인출과 디코드

$T_0: AR \leftarrow PC$

$T_1: IR \leftarrow M[AR], PC \leftarrow PC + 1$

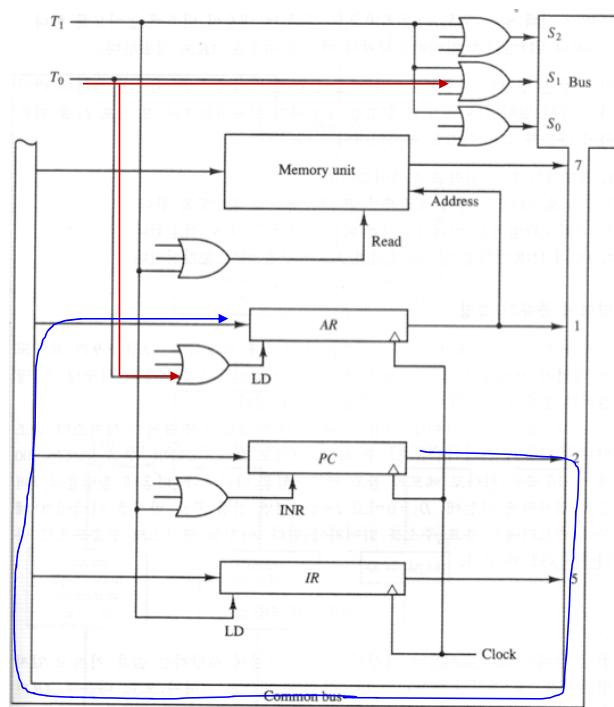
$T_2: D_0, \dots, D_7 \leftarrow \text{Decode } IR(12 - 14),$
 $AR \leftarrow IR(0 - 11), I \leftarrow IR(15)$

명령어 인출과 디코드

$T_0: AR \leftarrow PC$

$T_1: IR \leftarrow M[AR], PC \leftarrow PC + 1$

두 마이크로 연산을 위한 버스
시스템에서 레지스터 전송 예:

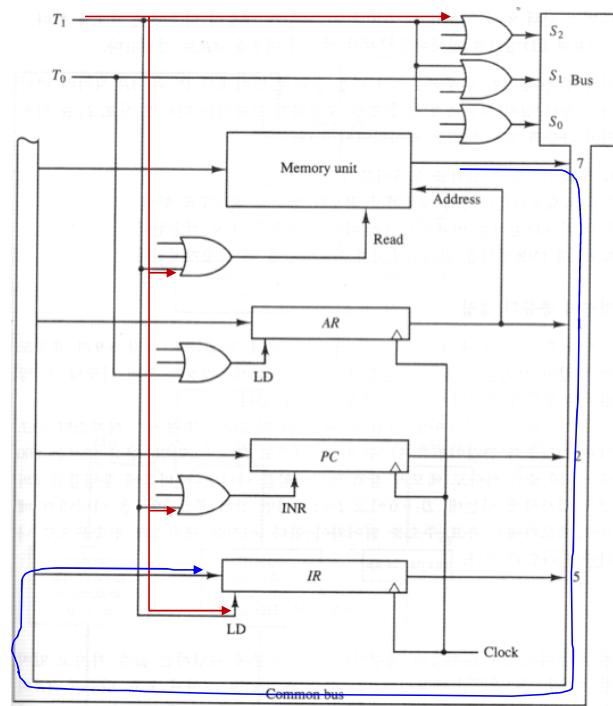


명령어 인출과 디코드

$T_0: AR \leftarrow PC$

$T_1: IR \leftarrow M[AR], PC \leftarrow PC + 1$

두 마이크로 연산을 위한 버스
시스템에서 레지스터 전송 예:



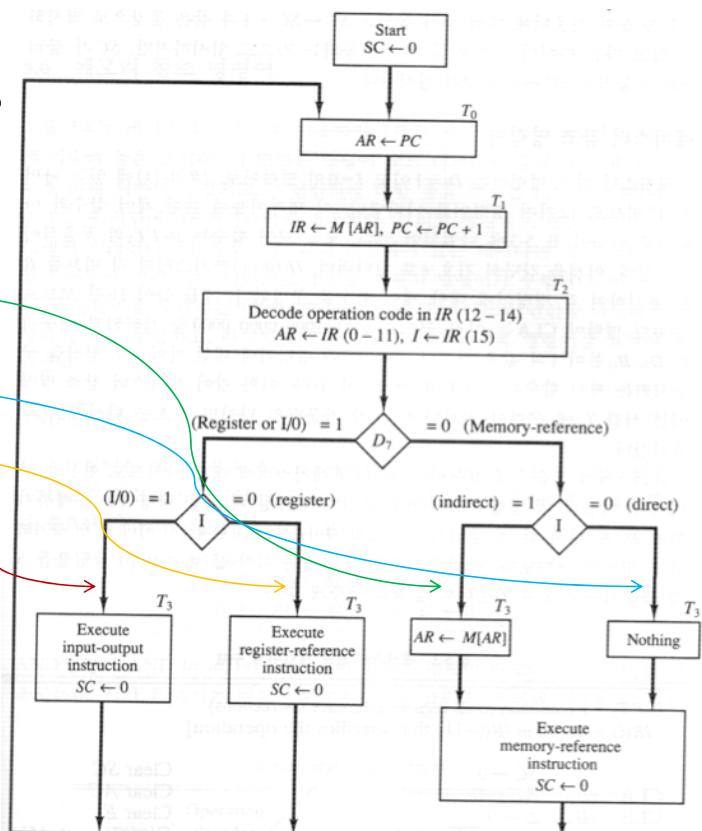
명령어 사이클 흐름도

$D'_7 IT_3: AR \leftarrow M[AR]$

$D'_7 I' T_3: -$

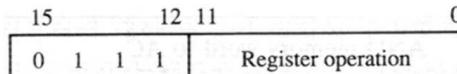
$D_7 I' T_3: Reg. Inst.$

$D_7 IT_3: I/O Inst.$



레지스터 참조 명령어

각 레지스터 참조 명령어를 위해 이 명령어들이 사용하지 않는 ‘Address’ 필드를 이용



$D_7 I' T_3 = r$ (common to all register-reference instructions)
 $IR(i) = B_i$ [bit in IR(0–11) that specifies the operation]

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



메모리 참조 명령어



메모리 참조 명령어 종류

Symbol	Operation decoder	Symbolic description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1, \text{ If } M[ar] + 1 = 0 \text{ then } PC \leftarrow PC + 1$



메모리 참조 명령어 - AND

$D_0 T_4: DR \leftarrow M[AR]$

$D_0 T_5: AC \leftarrow AC \wedge DR, SC \leftarrow 0$



메모리 참조 명령어 - ADD

$D_1 \ T_4: DR \leftarrow M[AR]$

$D_1 \ T_5: AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$



메모리 참조 명령어 - LDA

$D_2 \ T_4: DR \leftarrow M[AR]$

$D_2 \ T_5: AC \leftarrow DR, SC \leftarrow 0$



메모리 참조 명령어 - STA

$D_3 \ T_4: M[AR] \leftarrow AC, SC \leftarrow 0$



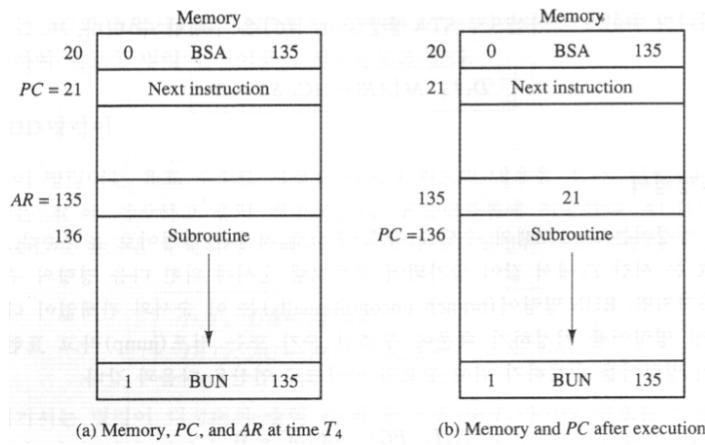
메모리 참조 명령어 - BUN

$D_4 \ T_4: PC \leftarrow AR, SC \leftarrow 0$



메모리 참조 명령어 - BSA

- Branch and Save return Address
- 명령어의 유효 주소에 Return Address 저장
- 분기주소 : (유효주소+1)



메모리 참조 명령어 - BSA

$D_5 \quad T_4: M[AR] \leftarrow PC, AR \leftarrow AR + 1$

$D_5 \quad T_5: PC \leftarrow AR, SC \leftarrow 0$

메모리 참조 명령어 - ISZ

$D_6 T_4: DR \leftarrow M[AR]$

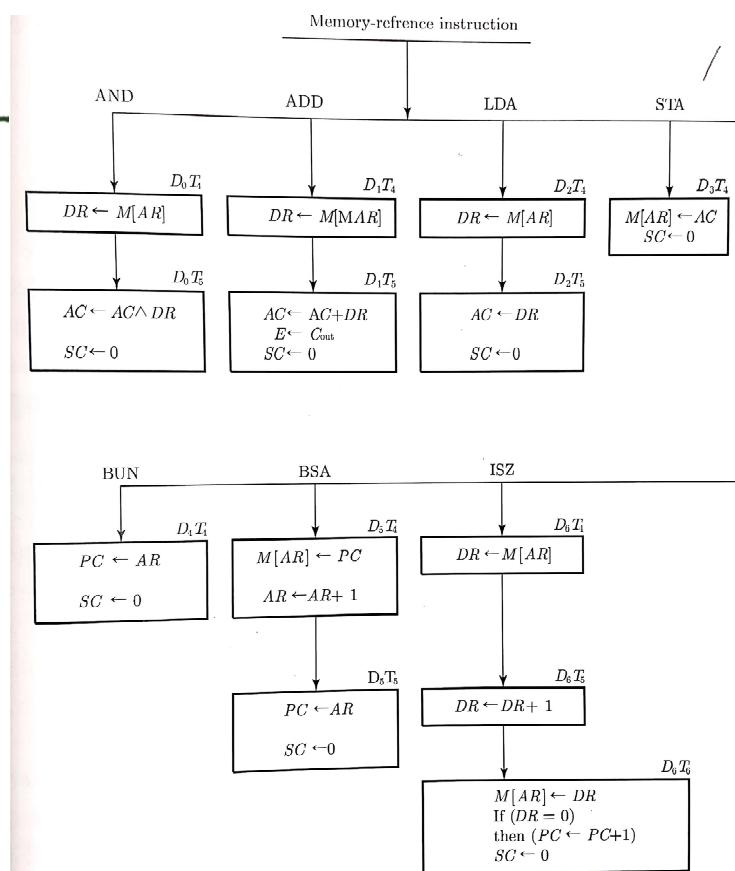
$D_6 T_5: DR \leftarrow DR + 1$

$D_6 T_6: M[AR] \leftarrow DR,$

if ($DR = 0$) then ($PC \leftarrow PC + 1$),

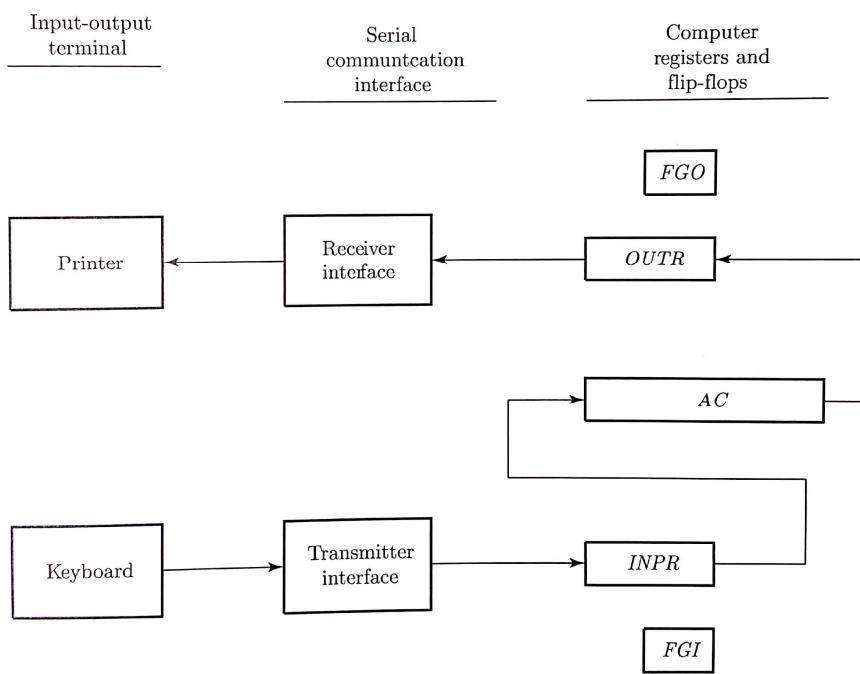
$SC \leftarrow 0$

메모리 참조
명령어
흐름도



입출력과 인터럽트

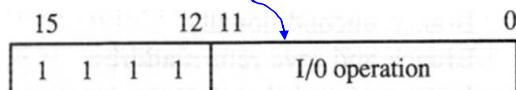
간단한 입출력 구성



입출력 명령어

$D_7IT_3 = p$ (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
INP	pB_{11} :	$AC(0-7) \leftarrow INPR, FGI \leftarrow 0$	Input character
OUT	pB_{10} :	$OUTR \leftarrow AC(0-7), FGO \leftarrow 0$	Output character
SKI	pB_9 :	If ($FGI = 1$) then ($PC \leftarrow PC + 1$)	Skip on input flag
SKO	pB_8 :	If ($FGO = 1$) then ($PC \leftarrow PC + 1$)	Skip on output flag
ION	pB_7 :	$IEN \leftarrow 1$	Interrupt enable on
IOF	pB_6 :	$IEN \leftarrow 0$	Interrupt enable off



각 입출력 명령어를 위해 입출력 명령어에서 사용하지 않는 'Address' 필드를 이용

입출력 방법

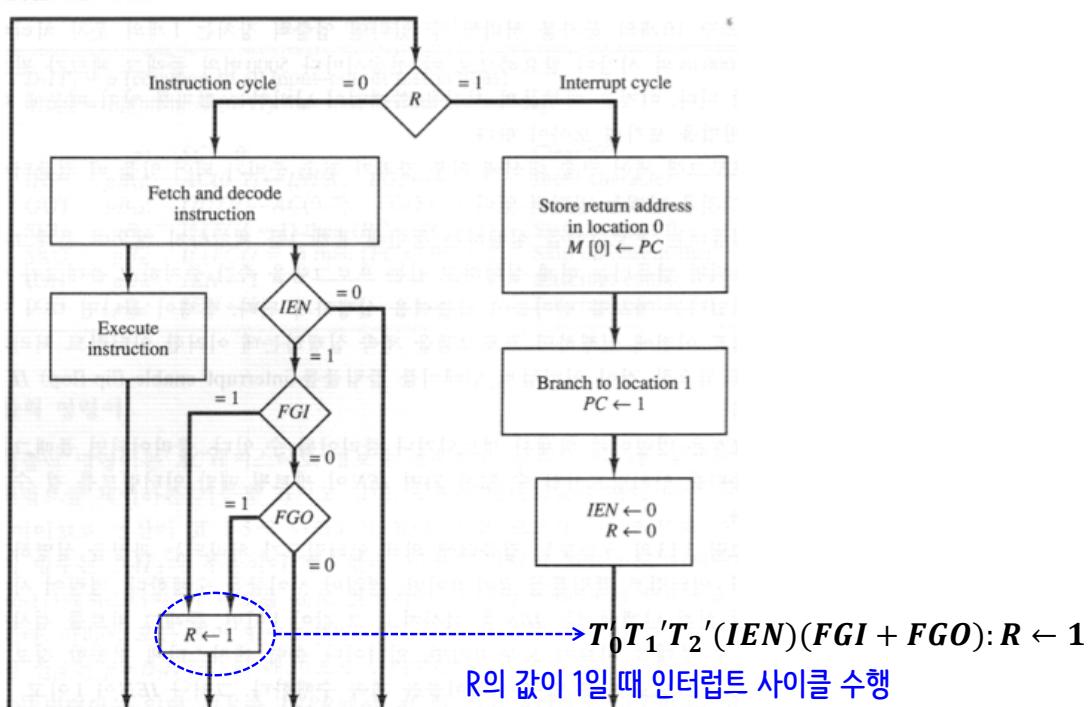
- 프로그램 제어 전송
 - Program Controlled Transfer
 - 프로그램이 입출력 플래그를 계속 검사
 - 효율 감소
- 인터럽트
 - 입출력 완료시 장치가 프로세서에게 알리는 방법
 - 효율 증가

기본 컴퓨터에서의 인터럽트 처리

- 1 비트 IEN 이용

- 1 : 인터럽트 - 입, 출력이 완료되어 다음 입, 출력 가능
- 0 : 인터럽트 X

인터럽트 사이클 처리 흐름



인터럽트 사이클

- 주요 기능

- 인터럽트 처리 루틴을 처리하기 전 복귀 주소 저장,
 - 처리 루틴으로의 분기

→ 하드웨어 적으로 처리

- 복귀 주소 및 각종 상태 저장

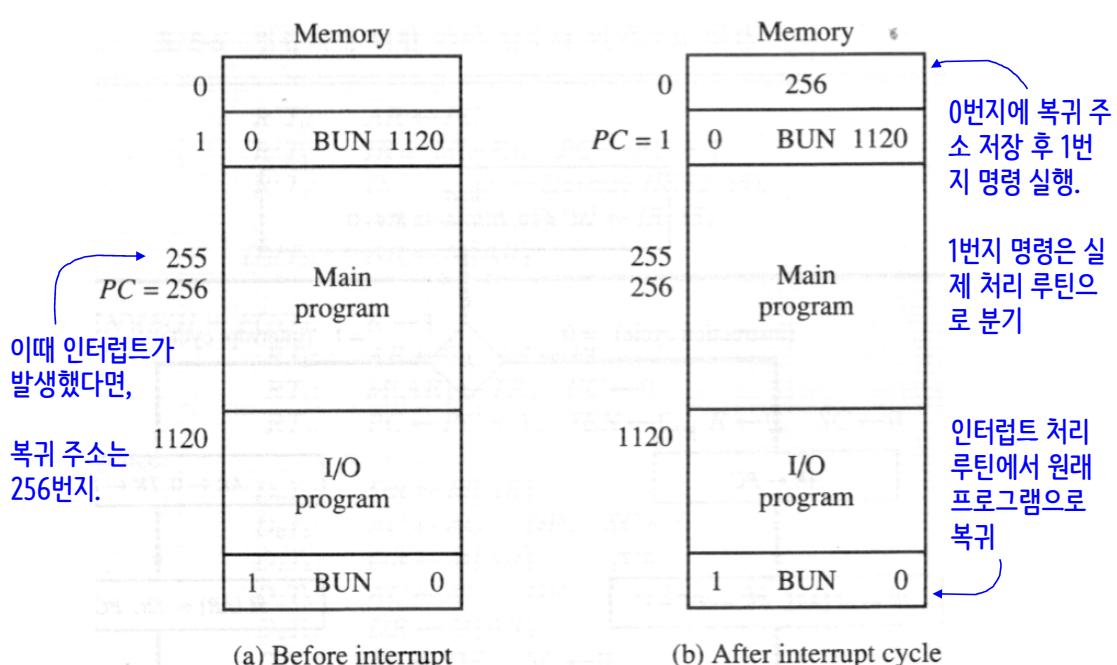
- 보통은 스택(stack)에 저장

- 기본 컴퓨터의 경우

- 메모리 0 번지에 복귀 주소 저장
- 메모리 1 번지로 부기 → 처리 루틴으로의 부기 명령



인터럽트 사이클 수행 예



인터럽트 사이클 수행

$RT_0: AR \leftarrow 0, TR \leftarrow PC$

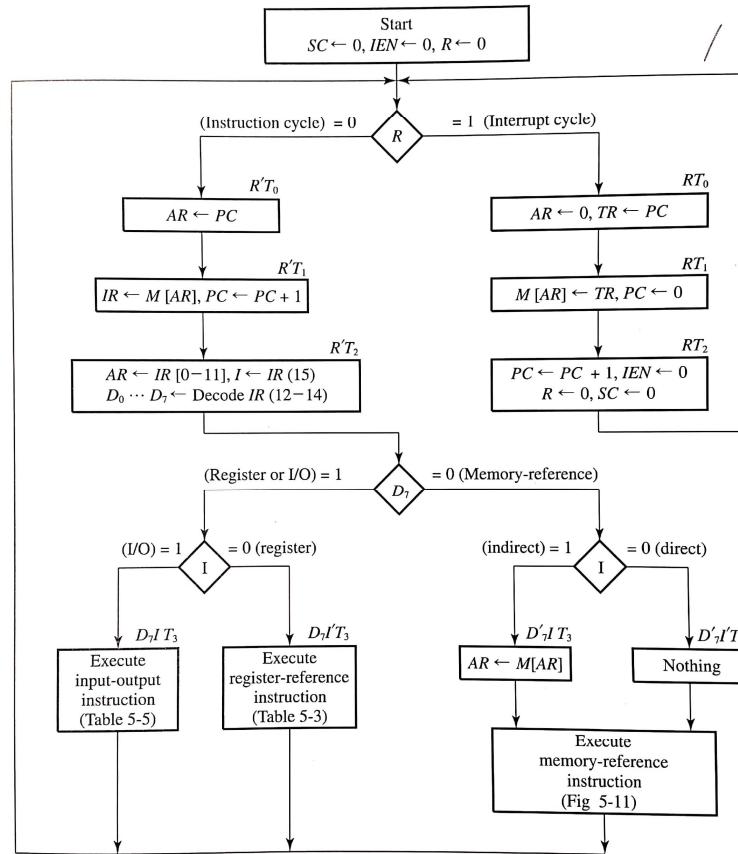
$RT_1: M[AR] \leftarrow TR, PC \leftarrow 0$

$RT_2: PC \leftarrow PC + 1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$

기본 컴퓨터에 대한 완전한 기술

Fetch	$H' T_0 : AR \leftarrow PC$ $H' T_1 : IR \leftarrow M[AR]$, $PC \leftarrow PC+1$
Decode	$H' T_2 : D_0, \dots, D_7 \leftarrow \text{Decode } IR(12-14), AR \leftarrow IR(0-11), I \leftarrow IR(15)$
Indirect	$D_1 T_3 : AR \leftarrow M[AR]$
Interrupt:	$T_0' T_1' T_2' : (IEV) \vee (FGI+FGO) : R \leftarrow -1$ $RT_0 : AR \leftarrow 0, TR \leftarrow PC$ $RT_1 : M[AR] \leftarrow TR, PC \leftarrow 0$ $RT_2 : PC \leftarrow PC+1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$
Memory-reference:	
AND	$D_0 T_4 : DR \leftarrow M[AR]$ $D_0 T_5 : AC \leftarrow AC \wedge DR, SC \leftarrow 0$
ADD	$D_1 T_4 : DR \leftarrow M[AR]$ $D_1 T_5 : AC \leftarrow AC + DR, E \leftarrow C_{ms}, SC \leftarrow 0$
LDA	$D_2 T_4 : DR \leftarrow M[AR]$ $D_2 T_5 : AC \leftarrow DR, SC \leftarrow 0$
STA	$D_3 T_4 : M[AR] \leftarrow AC, SC \leftarrow 0$
BUN	$D_4 T_4 : PC \leftarrow AR, SC \leftarrow 0$
BSA	$D_5 T_4 : M[AR] \leftarrow PC, AR \leftarrow AR+1$ $D_5 T_5 : PC \leftarrow AR, SC \leftarrow 0$
ISZ	$D_6 T_4 : DR \leftarrow M[AR]$ $D_6 T_5 : DR \leftarrow DR+1$ $D_6 T_6 : M[AR] \leftarrow DR, \text{ if } (DR=0) \text{ then } (PC \leftarrow PC+1), SC \leftarrow 0$
Register-reference:	$D_7 T_3 = r$ (common to all register-reference instructions) $IR(i) = B_i (i=0, 1, 2, \dots, 11)$ $r : SC \leftarrow 0$
CLA	$rB_1 : AC \leftarrow 0$
CLE	$rB_2 : E \leftarrow 0$
CMA	$rB_3 : AC \leftarrow \overline{AC}$
CME	$rB_4 : E \leftarrow \overline{E}$
CIR	$rB_5 : AC \leftarrow \text{shl } AC, AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	$rB_6 : AC \leftarrow \text{shl } AC, AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	$rB_7 : AC \leftarrow AC+1$
SPA	$rB_8 : \text{ If } (AC(15)=0) \text{ then } (PC \leftarrow PC+1)$
SNA	$rB_9 : \text{ If } (AC(15)=1) \text{ then } (PC \leftarrow PC+1)$
SZA	$rB_{10} : \text{ If } (AC=0) \text{ then } (PC \leftarrow PC+1)$
SZE	$rB_{11} : \text{ If } (E=0) \text{ then } (PC \leftarrow PC+1)$
HLT	$rB_{12} : S \leftarrow 0$
Input-output:	$D_7 T_3 = p$ (common to all input-output instructions) $IR(i) = B_i (i=6, 7, 8, 9, 10, 11)$ $p : SC \leftarrow 0$
INP	$pB_6 : AC(0-7) \leftarrow INPR, FGI \leftarrow 0$
OUT	$pB_7 : OUTR \leftarrow AC(0-7), FGO \leftarrow 0$
SKI	$pB_8 : \text{ If } (FGI=1) \text{ then } (PC \leftarrow PC+1)$
SKO	$pB_9 : \text{ If } (FGO=1) \text{ then } (PC \leftarrow PC+1)$
ION	$pB_{10} : IEN \leftarrow -1$
IOF	$pB_{11} : IEN \leftarrow 0$

컴퓨터 동작 흐름도



기본 컴퓨터 설계



기본 컴퓨터 설계

- 구성

- 4096X16 메모리
- 레지스터 : AR, PC, DR, AC, IR, TR, OUTR, INPR, SC
- 플립플롭 : I, S, E, R, IEN, FGI, FGO
- 2개의 디코더
- 16비트 버스 ; 16개의 8X1 멀티플렉서 이용
- 제어 논리 게이트
- ALU



제어 장치

- 제어 논리 회로(제어 논리 게이트)를 이용하여 제어 신호 생성

- 입력 : AC, DR, 7개의 플립플롭
- 출력 :
 - 9개 레지스터의 입력 제어 신호
 - 메모리 읽기/쓰기 제어 신호
 - 플립플롭의 세트, 클리어, 보수화 신호
 - 버스 제어(S_2, S_1, S_0)
 - 가산 논리 회로 제어 신호



AR 레지스터 제어

- AR의 값을 변경하는 제어연산/마이크로 연산

$R'T_0: AR \leftarrow PC$

$R'T_2: AR \leftarrow IR(0 - 11)$

$D'_7IT_7: AR \leftarrow M[AR]$

$RT_0: AR \leftarrow 0$

$D_5 T_4: AR \leftarrow AR + 1$



AR 레지스터 제어

- AR의 제어입력 논리 결정

$$LD(AR) = R'T_0 + R'T_2 + D'_7IT_3$$

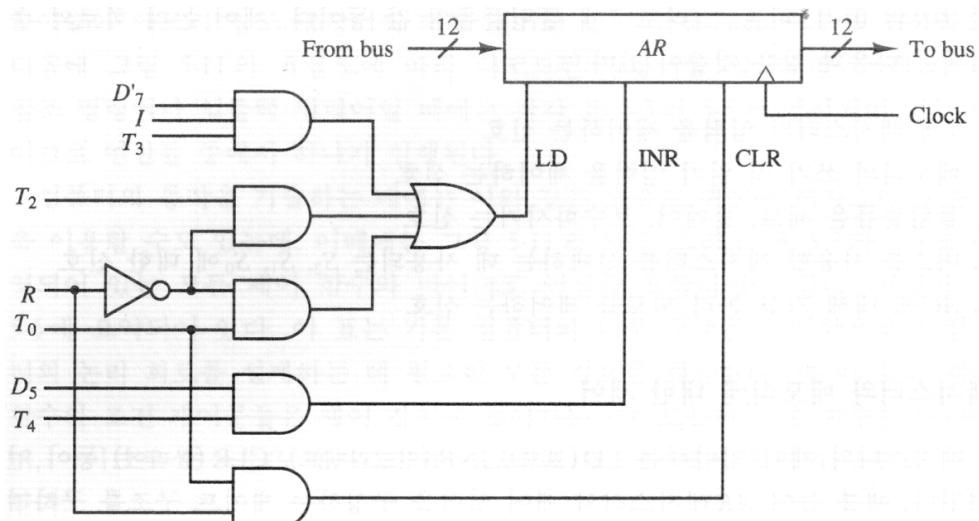
$$CLR(AR) = RT_0$$

$$INR(AR) = D_5T_4$$



AR 레지스터 제어

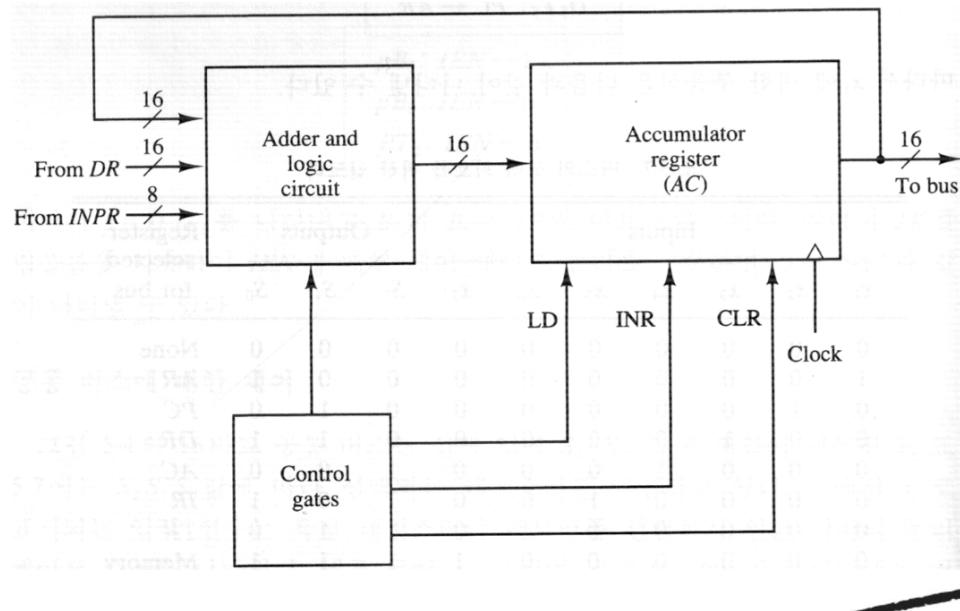
- AR의 제어입력 논리회로 작성



누산기 논리의 설계

AC 레지스터 제어

- AC와 관련된 회로



AC 레지스터 제어

- AC의 값을 변경하는 제어연산/マイ크로 연산
 - P.131 참고.
- AC의 제어입력 논리 결정

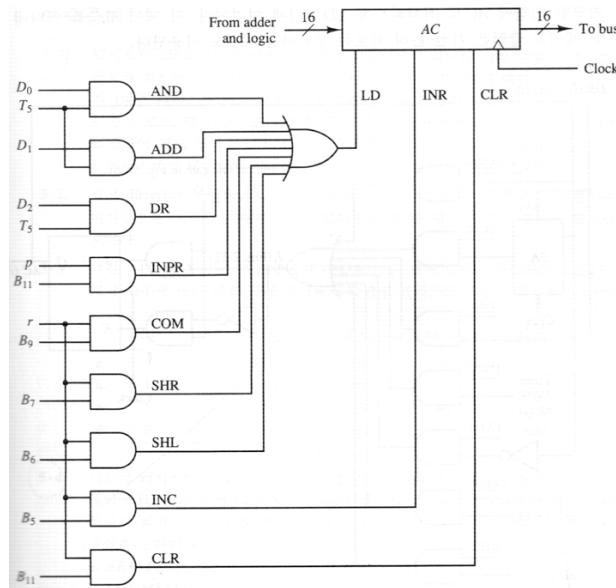
$$LD(AC) = \dots$$

$$CLR(AC) = \dots$$

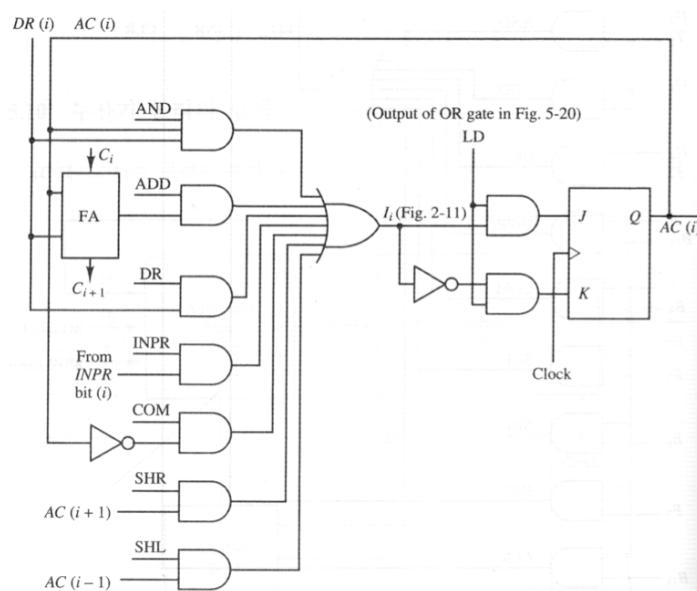
$$INR(AC) = \dots$$

AC 레지스터 제어

- AC의 제어입력 논리회로 작성



AC의 1 bit 논리 회로



Q&A

