Computer System Architecture

(THIRD EDITION)

M. Morris Mano

PRENTICE HALL

명령어, 프로그래밍 1줄이 무조건 1 instruction은 아님

5-1. Instruction Codes

5-1 Instruction Codes

프로그램이 구동 중

- The user of a computer can control the process by means of a ***program***

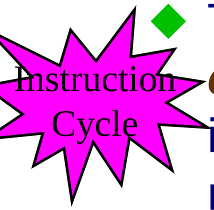
연산자 피연산자

- A program is a set of ***instructions*** that specify the operations, operand, the sequence(control)

+ 2 0 3

이진코드

- A instruction is a binary code that specifies a sequence of microoperations
- Instruction codes together with data are stored in memory(=Stored Program Concept)
- The computer reads each instruction from memory and ***places it in a control register***. The control then ***interprets the binary code*** of the instruction and proceeds to ***execute it*** by issuing a sequence of microoperations.



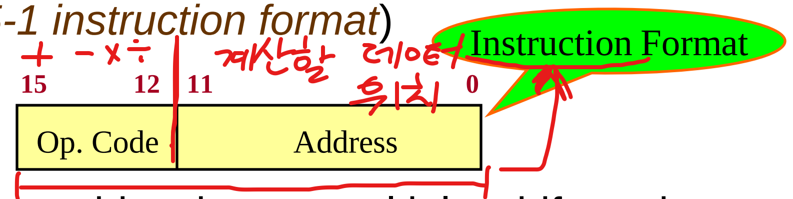
- Instruction Code :

  A group of bits that instruct the computer to perform a specific operation

  It is usually divided into parts(*refer to Fig. 5-1 instruction format*)
- Operation Code :

  The most basic part of an instruction code

  A group of bits that define such operations as add, subtract, multiply, shift, and complement(*bit 12-15 : 2^4 = 16 가지 distinct operations*)



5-1. Instruction Codes

◆ Stored Program Organization : ***Fig. 5-1***

The simplest way to organize a computer

» One processor register : AC(Accumulator) AC 레지스터가 많이 쓰인다 ☆

The operation is performed with the memory operand and the content of AC

» Instruction code format with two parts : Op. Code + Address

Op. Code : specify 16 possible operations(4 bit)

Address : specify the address of an operand(12 bit)

If an operation in an instruction code does not need an operand from memory, the rest of the bits in the instruction(***address field***) can be used for other purpose(따라서 16개 이상의 instruction을 사용 : Tab. 5-2 참고, 총 25 개 instruction)

Exam)
Clear AC, Increment AC,
Complement AC, ...

» Memory : 12 bit = 4096 word(Instruction and Data are stored)

if 2^{12}

Store each instruction code(***program***) and operand (***data***) in 16-bit memory word

◆ Addressing Mode 주소를 쉽게 알아내는 법 3가지 (원래는 많음)

1 Immediate operand address :

» the second part of an instruction code(***address field***) specifies an ***operand***

2 Direct operand address : ***Fig. 5-2(b)***

» the second part of an instruction code specifies the ***address of an operand***

3 Indirect operand address : ***Fig. 5-2(c)***

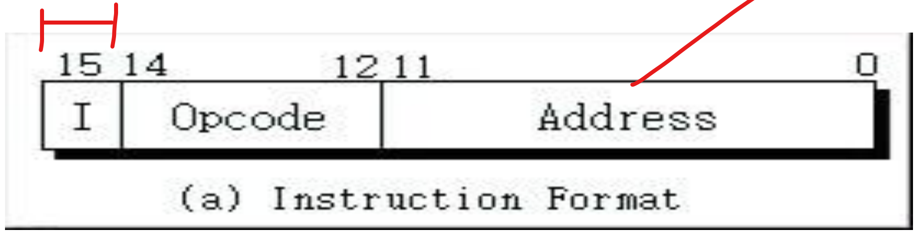
» the bits in the second part of the instruction designate an ***address of a memory word in which the address of the operand is found*** (Pointer로 사용됨)

I=0 : Direct,
I=1 : Indirect

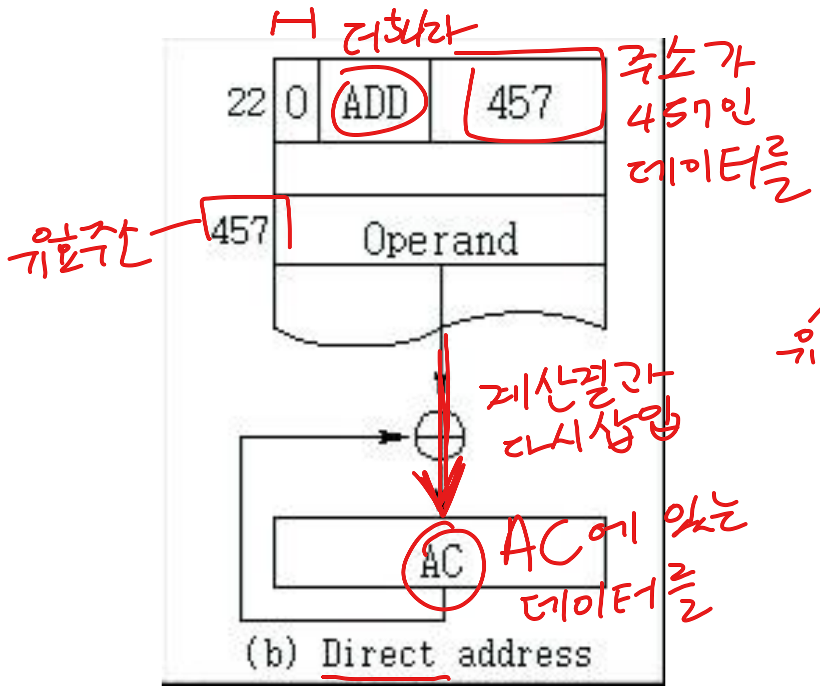
One bit of the instruction code is used to distinguish between a direct and an indirect address : ***Fig. 5-2(a)***

Fig. 5-2

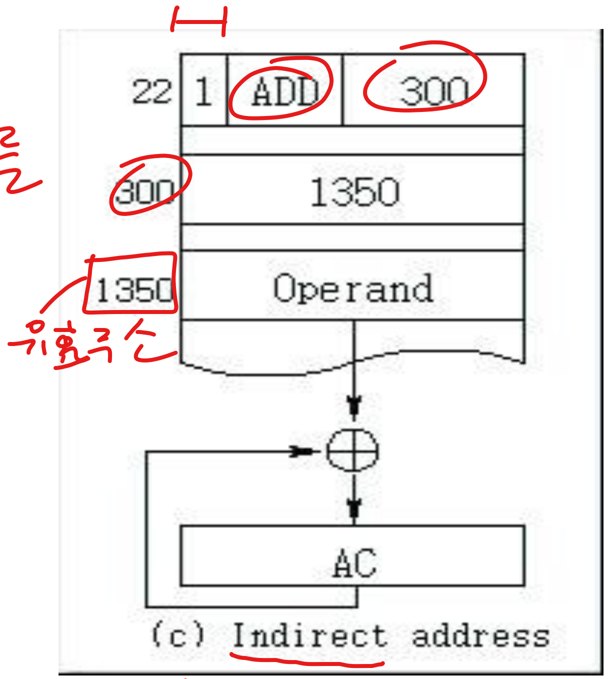
Immediate는 주소 없고, 바로 더함



(a) Instruction Format



(b) Direct address



(c) Indirect address

주소찾아가면 바로 피연산자 발견

300 찾아가면 진짜주소 발견

5-2. Computer Registers

유효한 주소 : 필요한 데이터를 가리킴. (주소를 가리키지 않음)

- Effective Address

  The ***operand address*** in *computation-type instruction* or the ***target address*** in a *branch-type instruction*

5-2 Computer Registers

CPU안에는 많은 레지스터가 있다, 개수 종류는 회사마다 다름

- List of Registers for the Basic Computer : ***Tab. 5-1***
- Basic computer registers and memory : ***Fig. 5-3***

  Data Register(***DR***) : hold the operand(Data) read from memory 필요한 데이터 저장

  Accumulator Register(***AC***) : general purpose processing register 계산

  Instruction Register(***IR***) : hold the instruction read from memory 명령어 저장

  Temporary Register(***TR***) : hold a temporary data during processing 임시 저장

  Address Register(***AR***) : hold a memory address, 12 bit width 주소 저장

  Program Counter(***PC***) : 다음에 수행할 명령어 저장

  » hold the address of the next instruction to be read from memory after the current instruction is executed

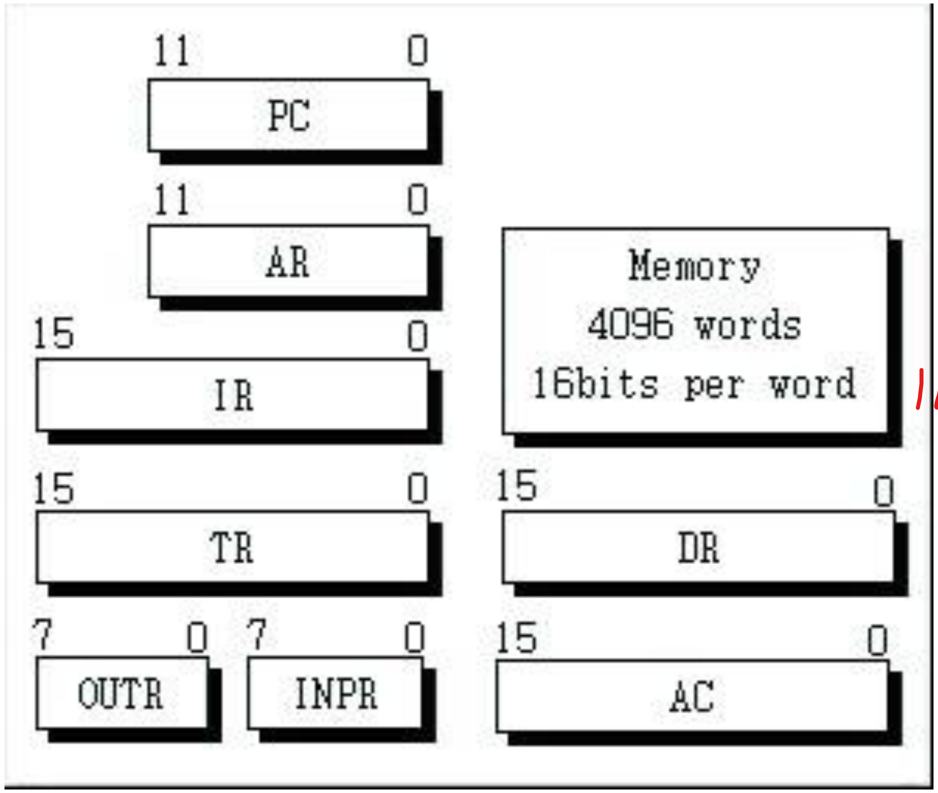
  » Instruction words are read and executed in sequence unless a branch instruction is encountered

  » A branch instruction calls for a transfer to a nonconsecutive instruction in the program

  » The address part of a branch instruction is transferred to PC to become the address of the next instruction

  » To read instruction, memory read cycle is initiated, and PC is incremented by one(next instruction fetch)

Fig. 5-3



1 word of 16 bits

5-2. Computer Registers

- Input Register(***INPR***) : receive an 8-bit character from an input device
- Output Register(***OUTR***) : hold an 8-bit character for an output device

◆ Common Bus System | 비트당 / 라인

- The basic computer has eight registers, a memory unit, and a control unit(*in Sec. 5-4*)
- Paths must be provided to transfer information from one register to another and between memory and registers
- A more efficient scheme for transferring information in a system with many registers is to use a common bus(*in Sec. 4-3*)
- The connection of the registers and memory of the basic computer to a common bus system : ***Fig. 5-4***
  - » The outputs of seven registers and memory are connected to the common bus
  - » The specific output is selected by mux(S0, S1, S2) :
    - Memory(7), AR(1), PC(2), DR(3), AC(4), IR(5), TR(6)
    - 외부 Device와의 입출력은 AC를 통해서 가능하기 때문에 INPR과 OUTR은 선택 없음
    - mux가 선택되어지면 memory 또는 register로 부터 데이터가 출력되어 bus위에 놓여진다
    - When LD(Load Input) is enable, the particular register receives the data from the bus
  - » Control Input : LD, INC, CLR, Write, Read
  - » Address Register : 별도의 Address bus 불필요(하나의 Bus로 address와 data 동시처리)
    - AC는 DR을 통해서만 memory read 가능(*p. 146, LDA 명령 참조*)
    - Memory write는 AC의 내용을 직접 write 가능(*p. 147, STA 명령 참조*)

5-2. Computer Registers

» Accumulator(AC) : 3 종류의 입력 Path

1) Register Microoperation : clear AC, shfift AC,…

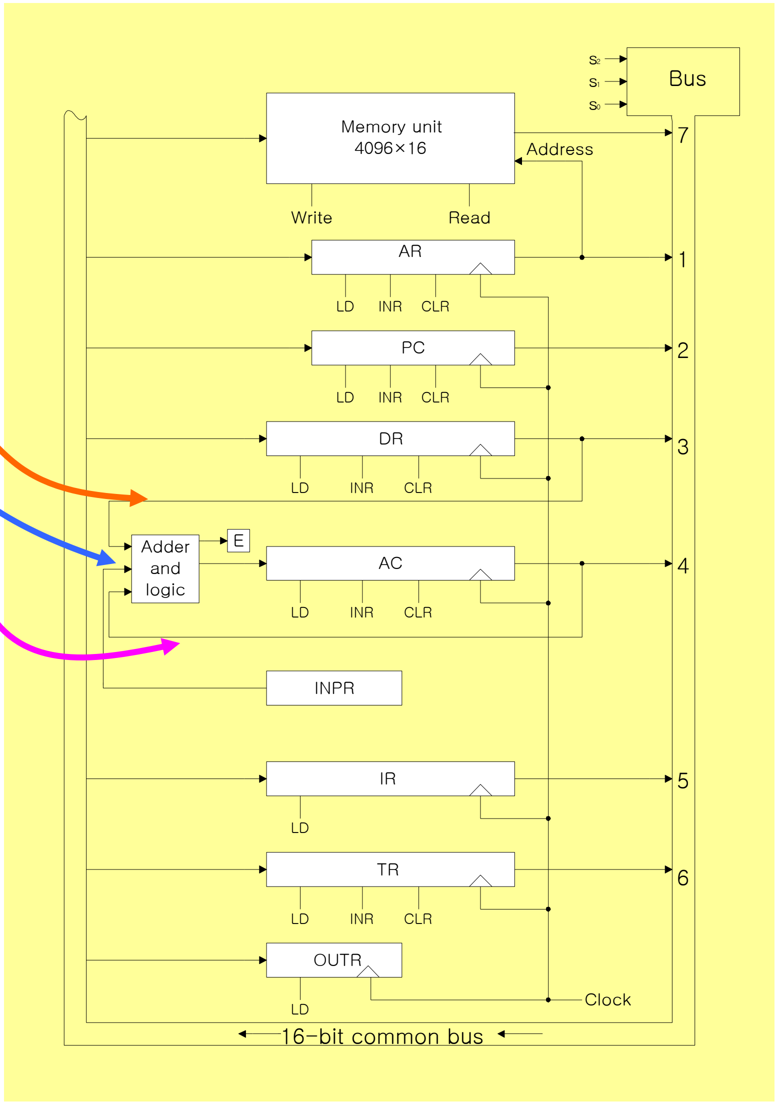
2) Data Register : **add DR to AC, and DR to AC**(연산결과는 AC에 저장하고 결과에 따라 End carry bit set/reset), memory READ(DR을 통해서만 가능)

3) INPR : 외부 Device에서 데이터 입력(*Adder & Logic을 거치지 않아도 됨*)

» Note) Two microoperations can be executed at the same time

$$DR \leftarrow AC : s_2 s_1 s_0 = 100(4), DR(load)$$

$$AC \leftarrow DR : DR \rightarrow Adder \,\&\, Logic \rightarrow AC(load)$$



5-3. Computer Instruction

Instruction 3가지

5-3 Computer Instruction

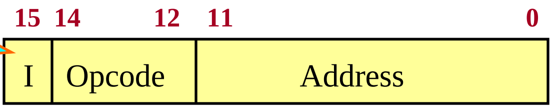
◆ 3 Instruction Code Formats : ***Fig. 5-5***

Memory-reference instruction 메모리에 특정 주소를 참조

» Opcode = 000 ~ 110

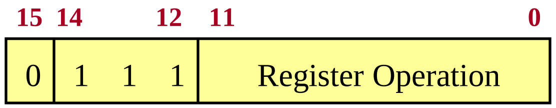
I=0 : 0xxx ~ 6xxx, I=1: 8xxx ~Exxx

I=0 : Direct,
I=1 : Indirect



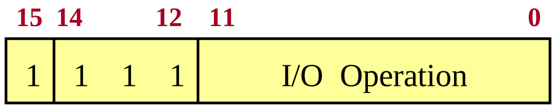
Register-reference instruction 현재 레지스터에 있는 데이터만으로 처리 함

» 7xxx (7800 ~ 7001) : CLA, CMA, ….



Input-Output instruction

» Fxxx(F800 ~ F040) : INP, OUT, ION, SKI, ….



Symbol	Hex Code I = 0	Hex Code 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
CMS	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

5-4. Timing and Control

- Instruction Set Completeness

  If the computer includes a sufficient number of instructions in each of the following categories

  Arithmetic, Logical, and shift : CMA, INC, ..

  Moving information to and from memory and AC : STA, LDA

  Program control : BUN, BSA, ISZ

  Input/Output : INP, OUT

5-4 Timing and Control

- Clock pulses

  얘가 일정한 시간간격을 두고 클럭 펄스 생성

  A master clock generator controls the timing for all registers in the basic computer

  The clock pulses are applied to all F/Fs and registers in system 모든 F/F와 R에 적용

  The clock pulses do not change the state of a register unless the register is enabled by a control signal

  The control signals are generated in the control unit : ***Fig. 5-6***

  » The control signals provide control inputs for the multiplexers in the common bus, control inputs in processor registers, and microoperations for the accumulator

- Two major types of control organization 제어하는 부분이 2가지!

  / Hardwired Control : *Chap. 5*

  » The control logic is implemented with gates, F/Fs, decoders, and other digital circuits

  » + Fast operation, - Wiring change(if the design has to be modified)

CPU 안의 제어장치가 하드웨어(회로)로 구성 / 빠르다, 변경 힘들다

5-4. Timing and Control

소프트웨어 구성 / 느리다, 변경 쉽다

2 Microprogrammed Control : *Chap. 7*

» The control information is stored in a control memory, and the control memory is programmed to initiate the required sequence of microoperations

» + Any required change can be done by updating the microprogram in control memory,
- Slow operation

◆ Control Unit : ***Fig. 5-6***

Control Unit = Control Logic Gate + 3 X 8 Decoder + Instruction Register + Timing Signal

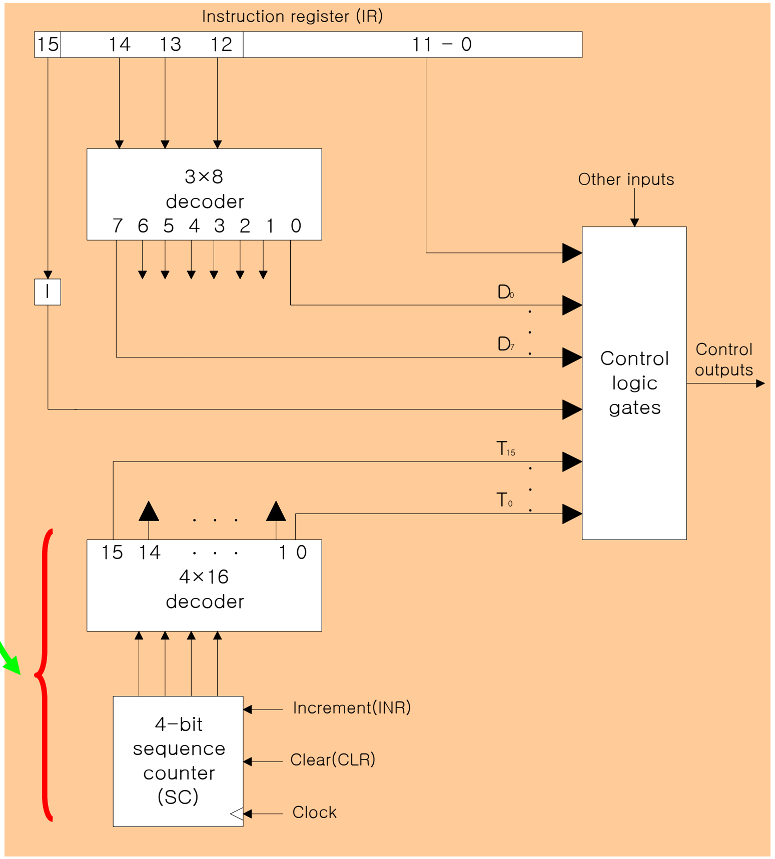
Timing Signal = 4 X 16 Decoder + 4-bit Sequence Counter

Exam) Control timing : ***Fig. 5-7***

» Sequence Counter is cleared when D_3T_4 =1 : $D_3T_4 : SC \leftarrow 0$

Memory R/W cycle time > Clock cycle time

» 만약 위와 같이 가정하지 않으면, wait cycle을 추가해야 함.



5-5. Instruction Cycle

- Exam) Register transfer statement : $T_0 : AR \leftarrow PC$

  A transfer of the content of PC into AR if timing signal T_0 is active

  » 1) During T_0 active, the content of PC is placed onto the bus $(S_2 S_1 S_0)$

  » 2) LD(load) input of AR is enabled, the actual transfer occurs at the next positive transition of the clock(*T_0 rising edge clock*)

  » 3) SC(sequence counter) is incremented : $0000\,(T_0) \rightarrow 0000\,(T_1)$

T_0 : Inactive
T_1 : Active

5-5 Instruction Cycle Instruction 1개가 수행되는 절차

- Instruction Cycle

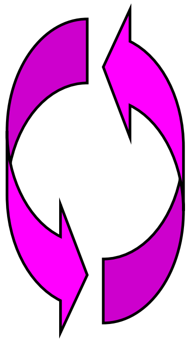
  1) Instruction Fetch from Memory 읽어오다

  2) Instruction Decode 해석

  3) Read Effective Address(if indirect addressing mode)

  4) Instruction Execution 연산

  5) Go to step 1) : Next Instruction[PC + 1]



Continue indefinitely unless HALT instruction is encountered

- Instruction Fetch : T0, T1(***Fig. 5-8***)

$$T_0 : AR \leftarrow PC$$
$$T_1 : IR \leftarrow M[AR],\ PC \leftarrow PC + 1$$

T0 = 1 $T_0 : AR \leftarrow PC$

» 1) Place the content of PC onto the bus by making the bus selection inputs $S_2S_1S_0$=010

» 2) Transfer the content of the bus to AR by enabling the LD input of AR

5-5. Instruction Cycle

T1 = 1 $T_1 : IR \leftarrow M[AR],\ PC \leftarrow PC+1$

» 1) Enable the read input memory
» 2) Place the content of memory onto the bus by making $S_2S_1S_0$= 111
» 3) Transfer the content of the bus to IR by enable the LD input of IR
» 4) Increment PC by enabling the INR input of PC

Instruction Decode : T2

$T_2 : D_0,\ldots, D_7 \leftarrow Decode\ \underbrace{IR(12-14)}_{\text{Op.code}},\ AR \leftarrow \underbrace{IR(0-11)}_{\text{Address}},\ I \leftarrow \underbrace{IR(15)}_{\text{Di/Indirect}}$

IR(12-14)에 따라 *Fig. 5-6* 에서 D0 - D7 출력

Instruction Execution : T3, T4, T5, T6

$IR(12-14) = 111$

D_7=1 { Register(I=0) ⟶ $D_7I'T_3$(Execute)
I/O (I=1) ⟶ D_7IT_3 (Execute)

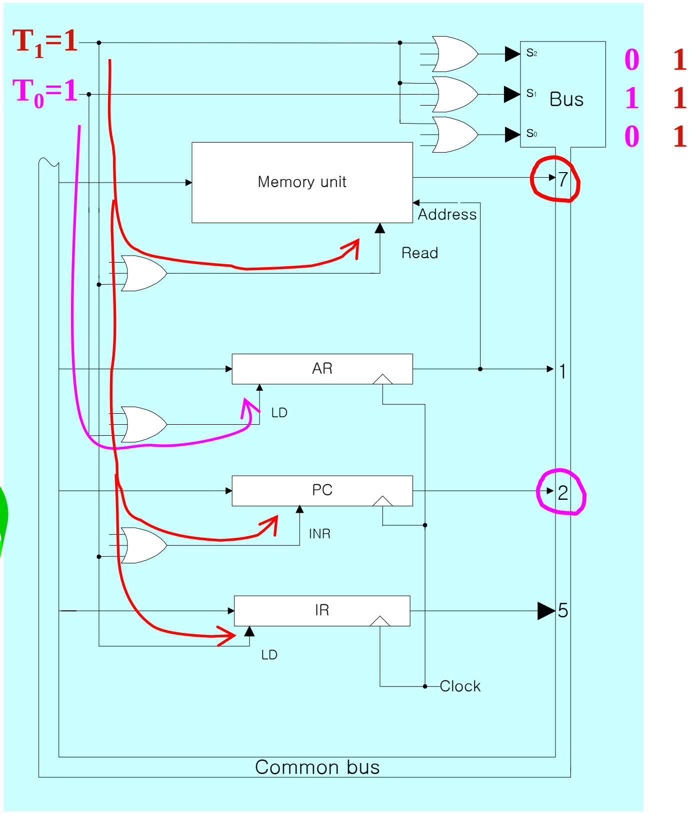
D_7=0 : Memory Ref. { Indirect(I=1) ⟶ $D_7'IT_3$($AR \leftarrow M[AR]$)
Direct (I=0) ⟶ nothing in T_3

Read effective Address

Register 와 I/O 명령은 T3에서 실행되며 Memory Ref. 명령은 T3에서 Operand의 effective address를 읽음

Memory Ref. 명령은 종류에 따라 T4, T5, T6을 갖음 : ***Fig. 5-11***

Flowchart for instruction cycle(Initial Configuration) : *Fig. 5-9*



5-6. Memory Reference Instruction

- Register Ref. Instruction

  r = $D_7 I' T_3$: 공통항

  IR(i) = B_i ← IR(0 -11)

  Address 로 사용되지 않음

  B_0 - B_{11} : **12 개의 Register Ref. Instruction (*Tab. 5-3*)**

5-6 Memory Ref. Instruction

3 X 8 Decoder

D_7 : Register or I/O = 1

IR(12,13,14) = 111

D_6 - D_0 : **7 개의 Memory Ref. Instruction(*Tab. 5-4*)**

- AND to AC

$$D_0T_4 : DR \leftarrow M[AR]$$
$$D_0T_5 : AC \leftarrow AC \wedge DR,\ SC \leftarrow 0$$

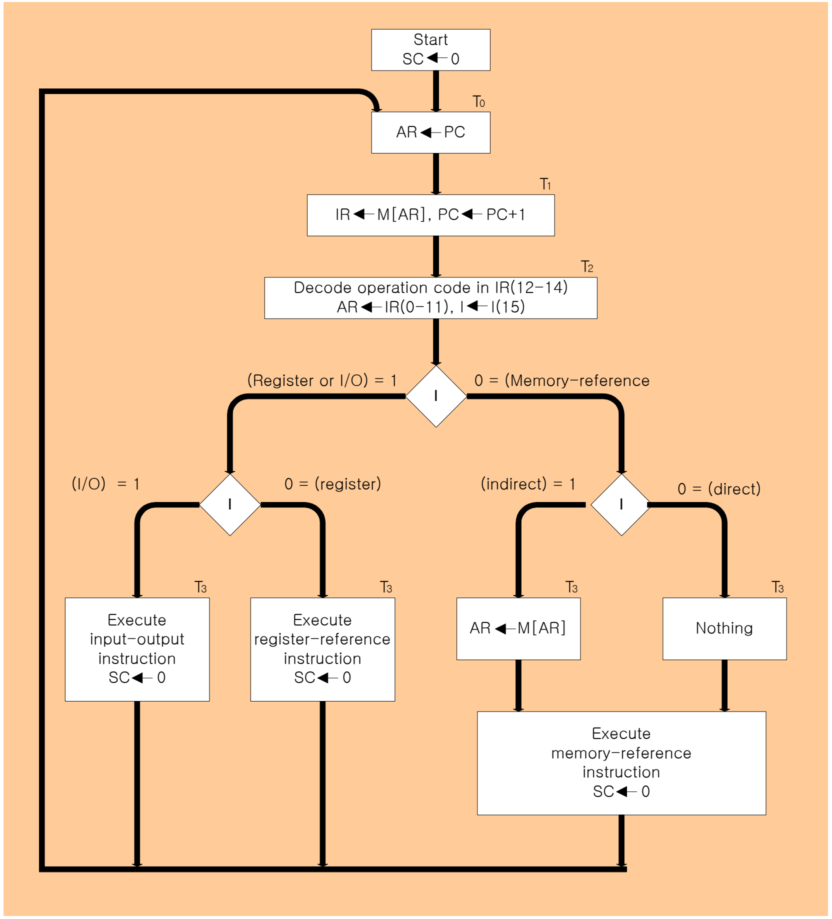
- ADD to AC

$$D_1T_4 : DR \leftarrow M[AR]$$
$$D_1T_5 : AC \leftarrow AC + DR,\ E \leftarrow C_{out},\ SC \leftarrow 0$$

- LDA : memory read

$$D_2T_4 : DR \leftarrow M[AR]$$
$$D_2T_5 : AC \leftarrow DR,\ SC \leftarrow 0$$

Fig. 5-9 Flowchart for instruction cycle(initial)



5-6. Memory Reference Instruction

명령어들

- STA : memory write

  $D_3T_4 : M[AR] \leftarrow AC, SC \leftarrow 0$

- BUN : branch unconditionally

  $D_4T_4 : PC \leftarrow AR, SC \leftarrow 0$

- BSA : branch and save return address

  $D_5T_4 : M[AR] \leftarrow PC, AR \leftarrow AR + 1$

  $D_5T_5 : PC \leftarrow AR, SC \leftarrow 0$

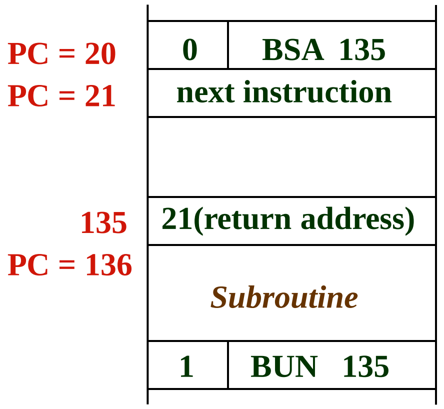
  Return Address : save return address (135 $\leftarrow$ 21)

  Subroutine Call : *Fig. 5-10* $\rightarrow$

  $D_5T_4 : M[135] \leftarrow 21(PC), 136(AR) \leftarrow 135 + 1$

  $D_5T_5 : 136(PC) \leftarrow 136(AR), SC \leftarrow 0$

Fig. 5-10 Example of BSA



- ISZ : increment and skip if zero

  $D_6T_4 : DR \leftarrow M[AR]$

  $D_6T_5 : DR \leftarrow DR + 1$

  $D_6T_6 : M[AR] \leftarrow DR, \text{ if } (DR = 0) \text{ then } (PC \leftarrow PC + 1), SC \leftarrow 0$

- Control Flowchart : *Fig. 5-11*

  Flowchart for the 7 memory reference instruction

  » The longest instruction : ISZ(T6)

  » 따라서 3 bit Sequence Counter로 구현가능(현재 4 비트는 확장에 대비함)

5-7. Input-Output and Interrupt

5-7 Input-Output and Interrupt

- Input-Output Configuration : ***Fig. 5-12***
  - Input Register(***INPR***), Output Register(***OUTR***)
    - » These two registers communicate with a communication interface serially and with the AC in parallel
    - » Each quantity of information has eight bits of an alphanumeric code
  - Input Flag(***FGI***), Output Flag(***FGO***)

    1 : Ready
    0 : Not ready

    - » FGI : ***set*** when INPR is ready(입력데이터가 있을 때), ***clear*** when INPR is empty
    - » FGO : ***set*** when operation is completed(데이터 출력 완료), ***clear*** when output device is in the process of printing
- Input-Output Instruction : ***Tab. 5-5***
  - $p = D_7IT_3$: 공통항
  - $IR(i) = B_i \longleftarrow IR(6 - 11)$

    Address 로 사용되지 않음

  - ***B_6 - B_{11}*** : **6 개의 I/O Instruction**
- Program Interrupt
  - I/O Transfer Modes

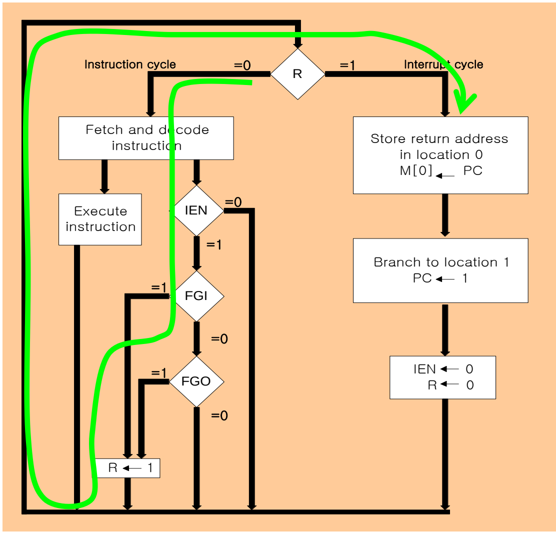
    멈춘 후 I/O 작업　　　ex) ctrl+c V

    - » 1) Programmed I/O, 2) Interrupt-initiated I/O, 3) DMA, 4) IOP

      저성능 ⟵ ⟶ 고성능

    - » 본 교과서에서는 2) Interrupt-initiated I/O 방식 사용(FGI 또는 FGO가 1이면 Int. 발생)
    - » Maskable Interrupt 사용(ION 또는 IOF 명령을 사용하여 Int. mask 가능)

5-7. Input-Output and Interrupt


Instruction cycle
=0
R
=1
Interrupt cycle
Fetch and decode instruction
Execute instruction
IEN
=0
=1
FGI
=1
=0
FGO
=1
=0
R ← 1
Store return address in location 0 M[0] ← PC
Branch to location 1 PC ← 1
IEN ← 0 R ← 0

Interrupt Cycle : ***Fig. 5-13***

- » During the execute phase, IEN is checked by the control
  - IEN = 0 : the programmer does not want to use the interrupt, so control continues with the next instruction cycle
  - IEN = 1 : the control circuit checks the flag bit, If either flag set to 1, R F/F is set to 1
- » At the end of the execute phase, control checks the value of R
  - R = 0 : 보통의 instruction cycle로 들어감
  - R = 1 : Instruction cycle로 들어감

Demonstration of the interrupt cycle : ***Fig. 5-14***

- » The memory location at address 0 as the place for storing the return address
- » Interrupt 발생시 항상 Branch to memory location 1
- » Interrupt cycle에서 항상 IEN=0 으로 함(*따라서 ISR에서 Interrupt를 받기 위해서는 ISR 앞부분에서 반드시 ION 명령을 실행해야 함*)

The condition for R = 1

$$T_0'T_1'T_2'(IEN)(FGI+FGO): R \leftarrow 1$$

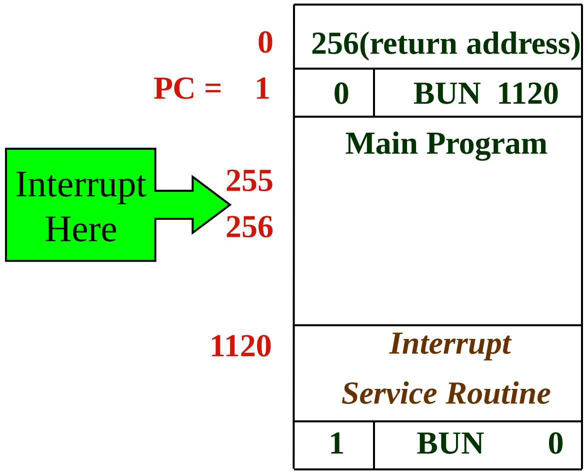
Modified Fetch Phase

- » Modified Fetch and Decode Phase

Save Return Address(PC) at 0

Jump to 1(PC=1)

$$RT_0 : AR \leftarrow 0, TR \leftarrow PC$$
$$RT_1 : M[AR] \leftarrow TR, PC \leftarrow 0$$
$$RT_2 : PC \leftarrow PC + 1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$$



5-8. Complete Computer Description

5-8 Complete Computer Description

- The final flowchart of the instruction cycle : Fig. 5-15
- The control function and microoperation : Tab. 5-6

5-9 Design of Basic Computer

- The basic computer consists of the following hardware components
  1. A memory unit with 4096 words of 16bits
  2. Nine registers : AR, PC, DR, AC, IR, TR, OUTR, INPR, and SC(***Fig. 2-11***)
  3. Seven F/Fs : I, S, E, R, IEN, FGI, and FGO
  4. Two decoder in control unit : 3 x 8 operation decoder, 4 x 16 timing decoder(***Fig. 5-6***)
  5. A 16-bit common bus(***Fig. 5-4***)
  6. ***Control Logic Gates*** : *Fig. 5-6의 오른쪽 Box 부분에서* ***Control Output 설계***
  7. ***Adder and Logic circuit connected to the AC input***
- Control Logic Gates
  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 F/Fs
  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

5-9 Design of Basic Computer

Register Control : AR

$AR \leftarrow ?$

Control inputs of AR : ***LD, INR, CLR***

Find all the statements that change the AR in Tab. 5-6

$R'T_0 : AR \leftarrow PC$

$R'T_1 : AR \leftarrow IR(0-11)$

$D_7'IT_3 : AR \leftarrow M[AR]$

$RT_0 : AR \leftarrow 0$

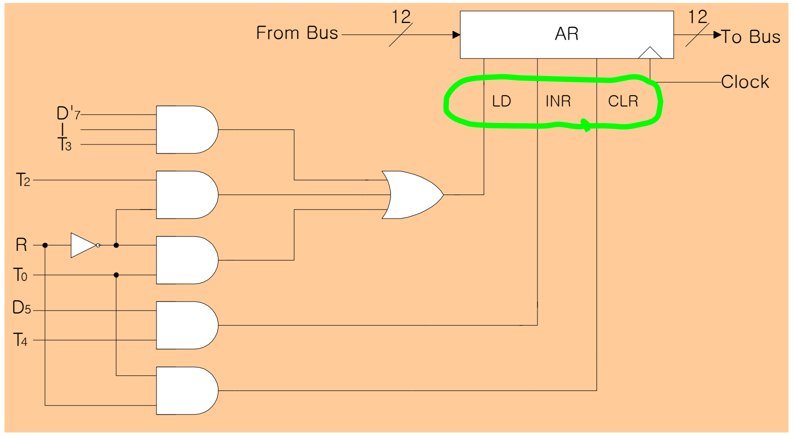
$D_5T_4 : AR \leftarrow AR+1$

Control functions

$LD(AR) = R'T_0 + R'T_1 + D_7'IT_3$

$CLR(AR) = RT_0$

$INR(AR) = D_5T_4$



Memory Control : READ

Control inputs of Memory : ***READ, WRITE***

$M[AR] \leftarrow ?$

Find all the statements that specify a ***read operation*** in Tab. 5-6

$? \leftarrow M[AR]$

Control function

$READ = R'T_1 + D_7'IT_3 + (D_0 + D_1 + D_2 + D_3)T_4$

F/F Control : IEN

$IEN \leftarrow ?$

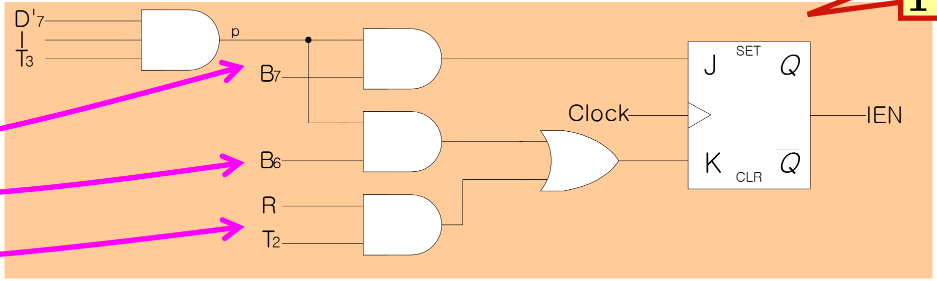
Control functions

$pB_7 : IEN \leftarrow 1$

$pB_6 : IEN \leftarrow 0$

$RT_2 : IEN \leftarrow 0$

J	K	Q(t+1)
0	1	0
1	0	1



5-9. Design of Basic Computer

Bus Control

Encoder for Bus Selection : ***Tab. 5-7***

- » $S_0 = x_1 + x_3 + x_5 + x_7$
- » $S_1 = x_2 + x_3 + x_6 + x_7$
- » $S_0 = x_4 + x_5 + x_5 + x_7$

$x_1 = 1$: $Bus \leftarrow AR = Find\ ? \leftarrow AR$

- » $D_4T_4 : PC \leftarrow AR$
  $D_5T_5 : PC \leftarrow AR$
- » Control Function : $x_1 = D_4T_4 + D_5T_5$

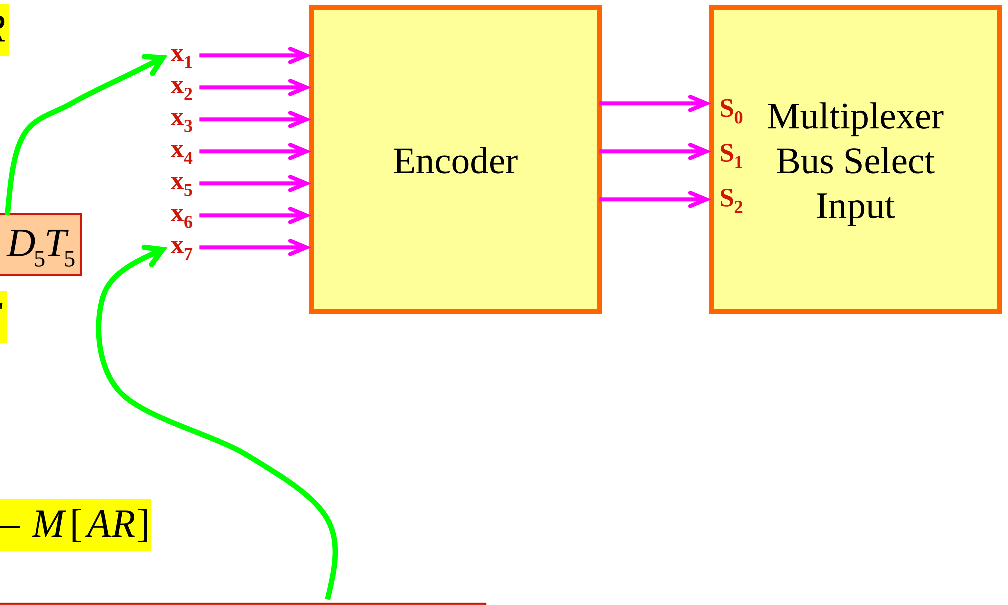
$x_2 = 1$: $Bus \leftarrow PC = Find\ ? \leftarrow PC$

“

“

$x_7 = 1$: $Bus \leftarrow Memory = Find\ ? \leftarrow M[AR]$

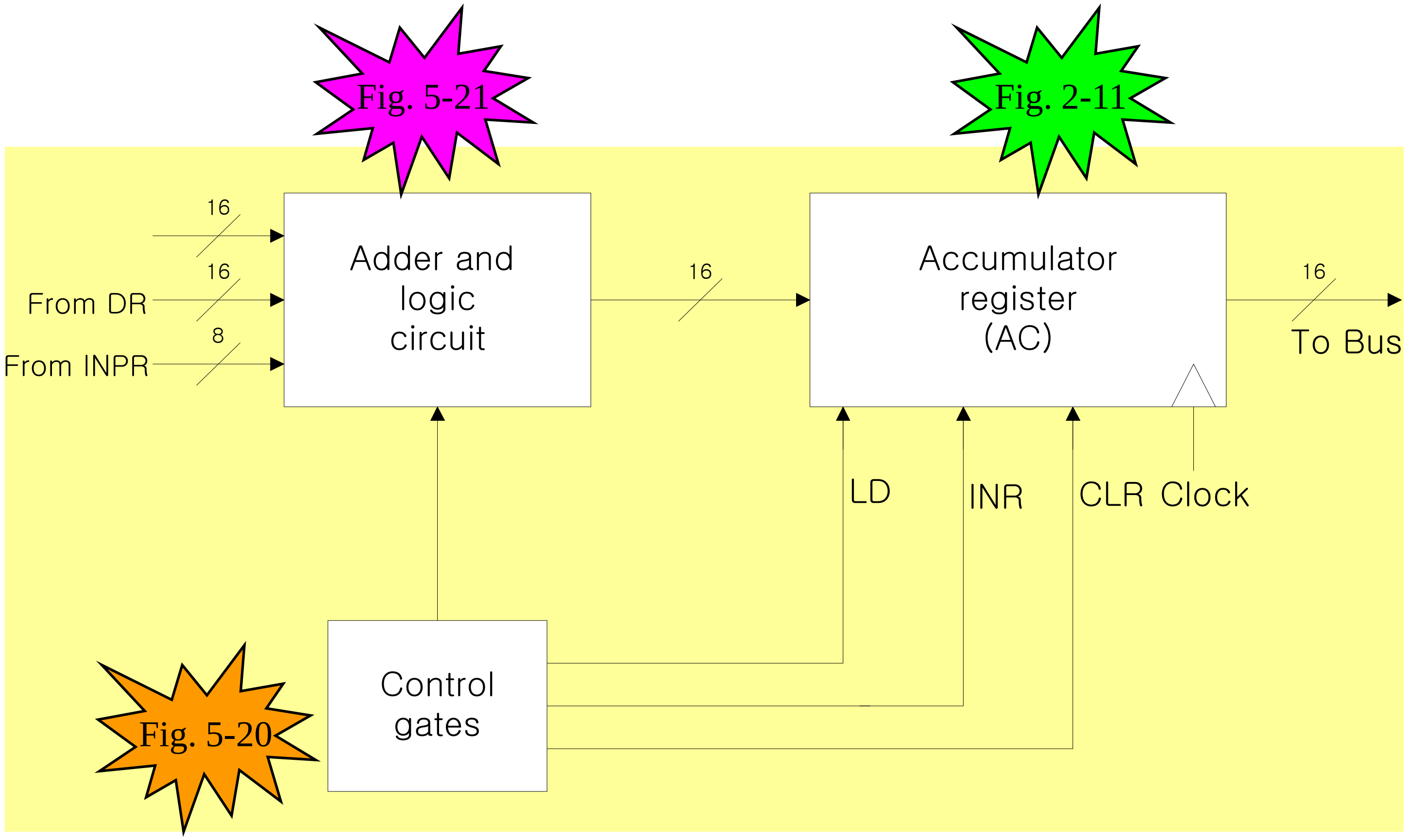
- » Same as Memory Read
- » Control Function : $x_7 = R'T_1 + D_7'IT_3 + (D_0 + D_1 + D_2 + D_3)T_4$



5-10. Design of Accumulator Logic

5-10 Design of Accumulator Logic

- Circuits associated with AC : ***Fig. 5-19***



5-10. Design of Accumulator Logic

◆ Control of AC : ***Fig. 5-20***

Find the statement that change the AC : $AC \leftarrow ?$

$D_0T_5 : AC \leftarrow AC \wedge DR$

$D_1T_5 : AC \leftarrow AC + DR$

$D_2T_5 : AC \leftarrow DR$

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

$rB_9 : AC \leftarrow \overline{AC}$

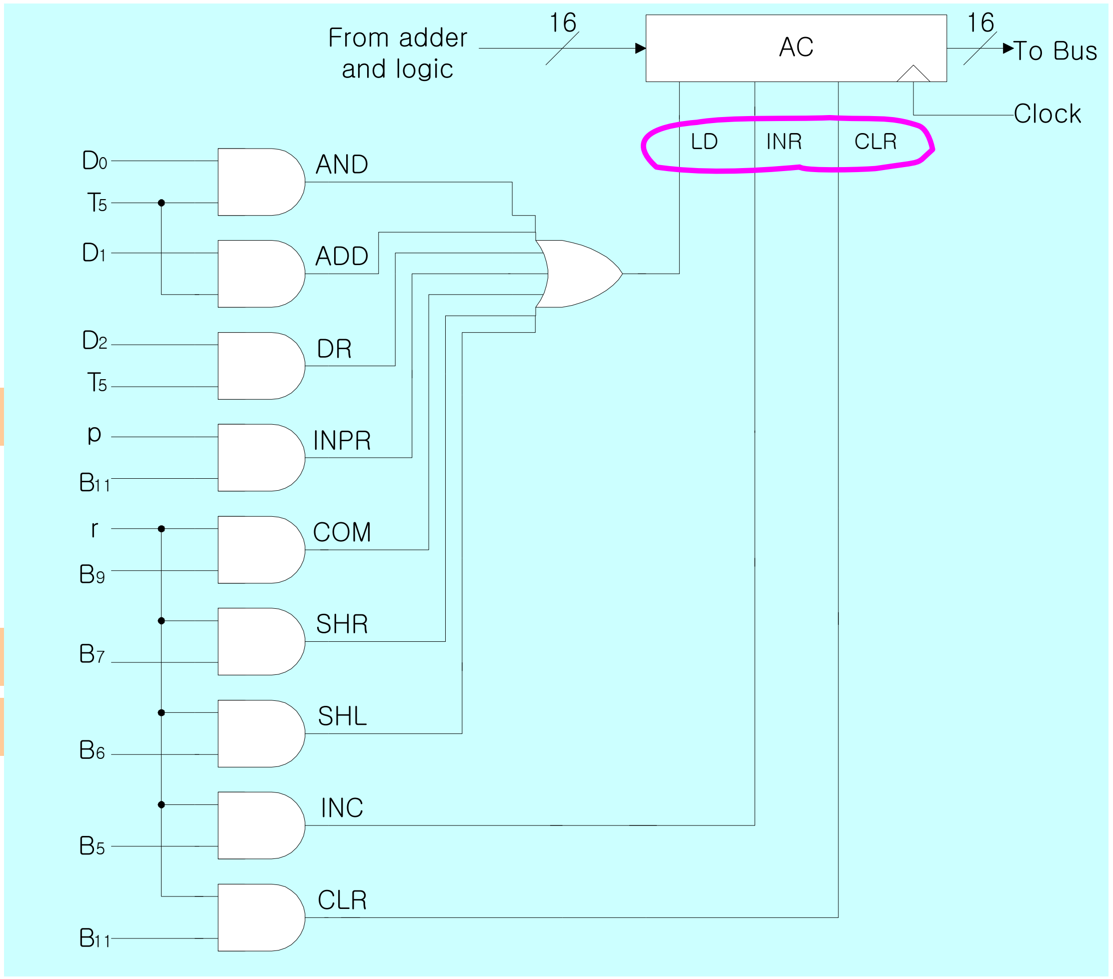
$rB_7 : AC \leftarrow shr\ AC, AC(15) \leftarrow E$

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

LD

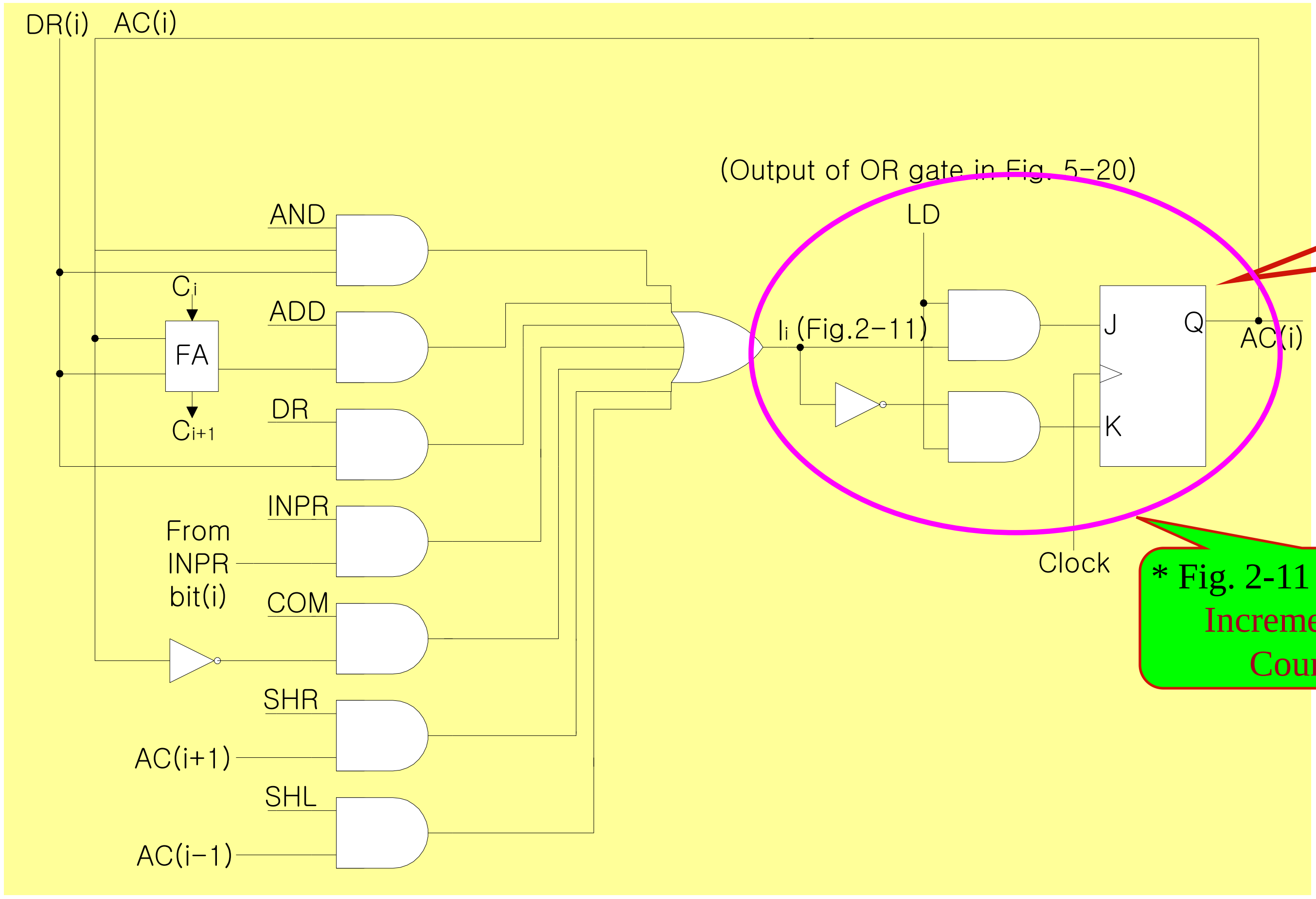
$rB_{11} : AC \leftarrow 0$ → **CLR**

$rB_5 : AC \leftarrow AC + 1$ → **INR**



5-10. Design of Accumulator Logic

◆ Adder and Logic Circuit : ***Fig. 5-21*** ***(16 bit = 16 개 필요)***



J	K	Q(t+1)
0	1	0
1	0	1

* Fig. 2-11 로 대체 가능
Increment, Clear,
Count 기능

Mano Machine

Integration !

Fig. 5-4 : Common Bus*(p.130)*

Fig. 2-11 : Register*(p. 59)*

Fig. 5-6 : Control Unit*(p. 137)*

Fig. 5-16, 17,18 : Control Logic Gate*(p.161- 163)*

- Fig. 5-4의 모든 Component의 Control Input
- 각각의 Register, Memory, F/Fs, Bus Selection

Fig. 5-20 : AC control*(p.165)*

Fig. 5-21 : Adder and Logic*(p.166)*