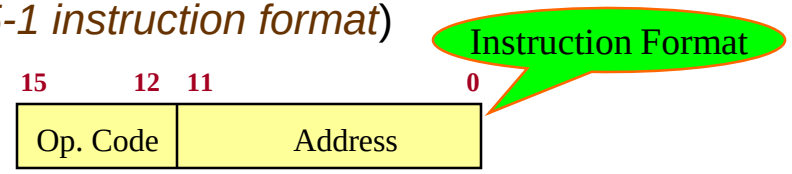
Chap. 5 Basic Computer Org. and Design

5-1 Instruction Codes

- The user of a computer can control the process by means of a ***program***
- A program is a set of ***instructions*** that specify the operations, operand, the sequence(control)
- 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 Cycle

- 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 : $2^4 = 16$ distinct operations*)



Stored Program Organization :

- The simplest way to organize a computer
  - » One processor register : AC(Accumulator)
    - 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

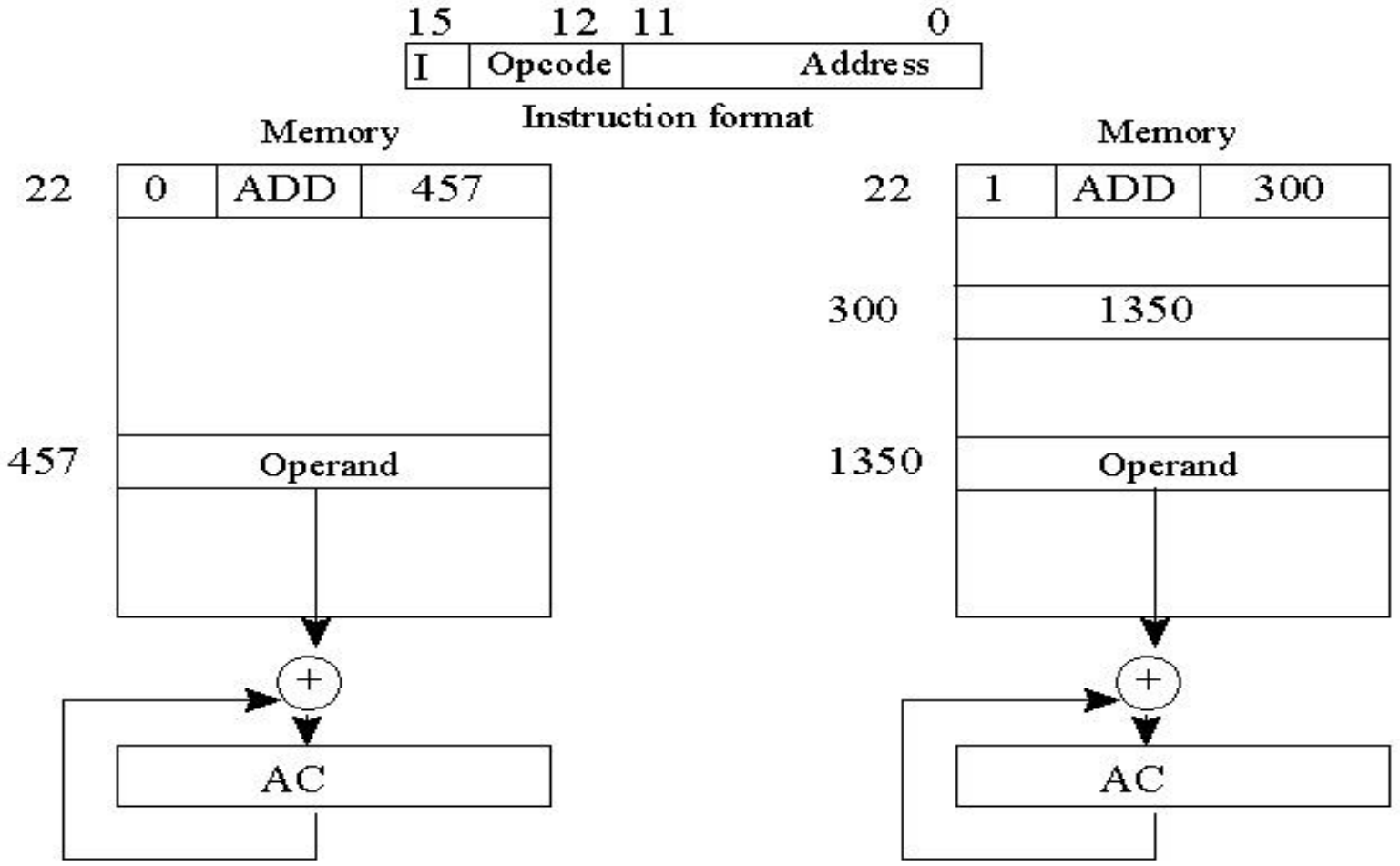
Exam)
Clear AC, Increment AC,
Complement AC, ...

  - » Memory : 12 bit = 4096 word(Instruction and Data are stored)
    - Store each instruction code(***program***) and operand (***data***) in 16-bit memory word

Addressing Mode

- Immediate operand address :
  - » the second part of an instruction code(***address field***) specifies an ***operand***
- Direct operand address : ***Fig. 5-2(b)***
  - » the second part of an instruction code specifies the ***address of an operand***
- 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***
- One bit of the instruction code is used to distinguish between a direct and an indirect address : ***Fig. 5-2(a)***

I=0 : Direct,
I=1 : Indirect


15
12
11
0
I
Opcode
Address
Instruction format
Memory
22
0
ADD
457
457
Operand
AC
Memory
22
1
ADD
300
300
1350
1350
Operand
AC

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

5-2 Computer Registers

- Basic computer registers and memory :
  - Data Register(***DR***) : hold the operand(Data) read from memory
  - Accumulator Register(***AC***) : general purpose processing register
  - Instruction Register(***IR***) : hold the instruction read from memory
  - Temporary Register(***TR***) : hold a temporary data during processing
  - Address Register(***AR***) : hold a memory address, 12 bit width
  - Program Counter(***PC***) :
    - » 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)

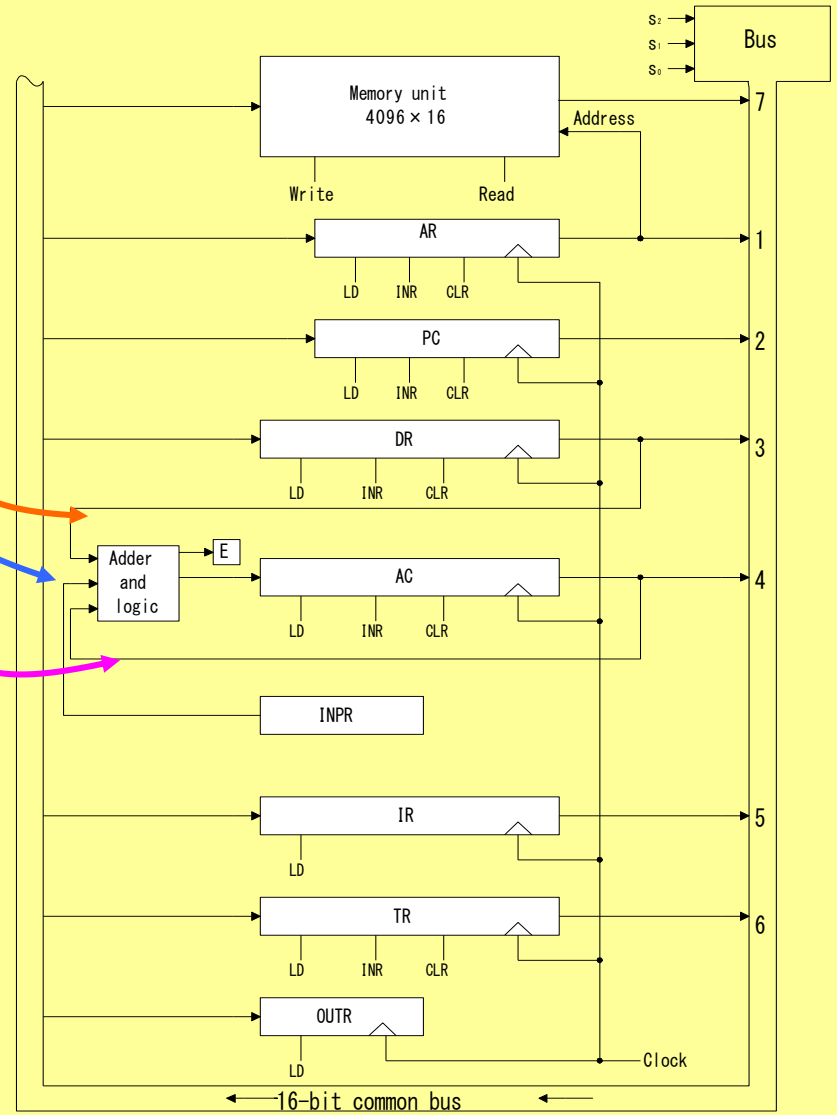
- Input Register(***INPR***) : receive an 8-bit character from an input device
- Output Register(***OUTR***) : hold an 8-bit character for an output device

Common Bus System

- The basic computer has eight registers, a memory unit, and a control
- 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
- 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)
    - When LD(Load Input) is enable, the particular register receives the data from the bus
  - » Control Input : LD, INC, CLR, Write, Read
  - » Address Register :

- Accumulator(AC) :
  - 1) Register Microoperation : clear AC, shfift AC,…
  - 2) Data Register : **add** DR **to** AC, **and** DR **to** AC
  - 3) INPR
- Note) Two microoperations can be executed at the same time

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

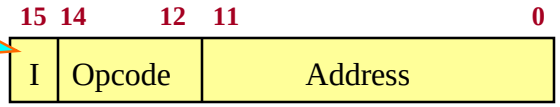


5-3 Computer Instruction

3 Instruction Code Formats :

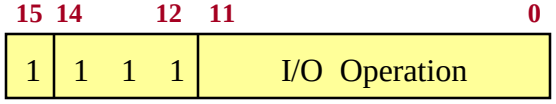
- Memory-reference instruction
  - » Opcode = 000 ~ 110
    - I=0 : 0xxx ~ 6xxx, I=1: 8xxx ~Exxx

I=0 : Direct,
I=1 : Indirect



- Register-reference instruction
  - » 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

- 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

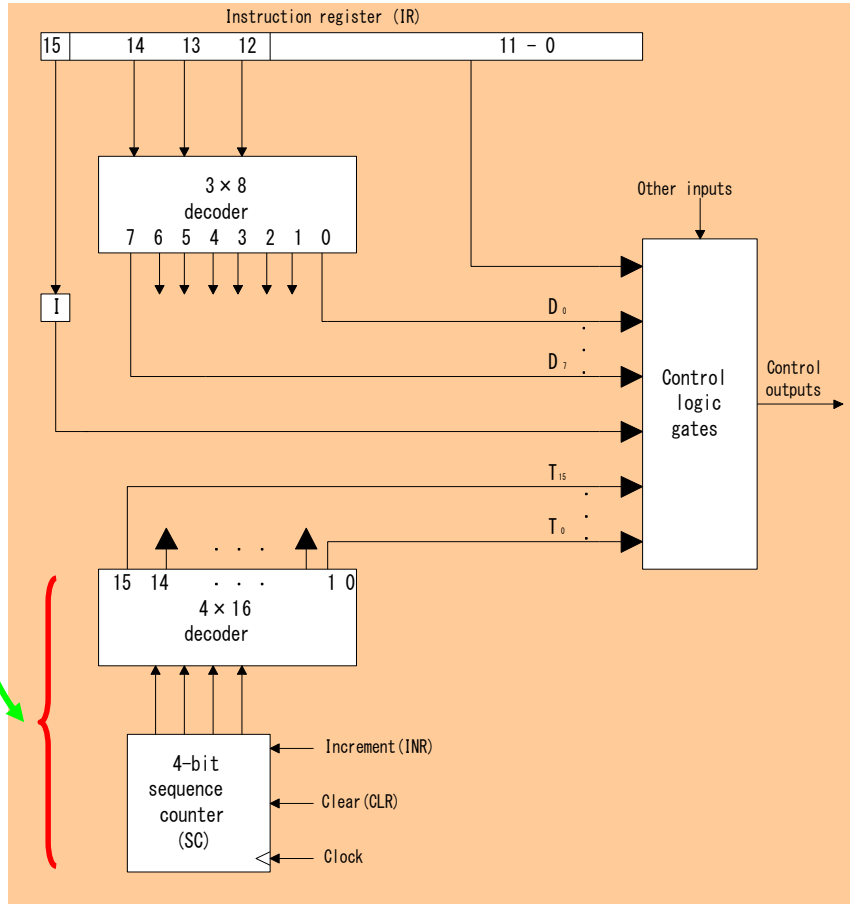
5-4 Timing and Control

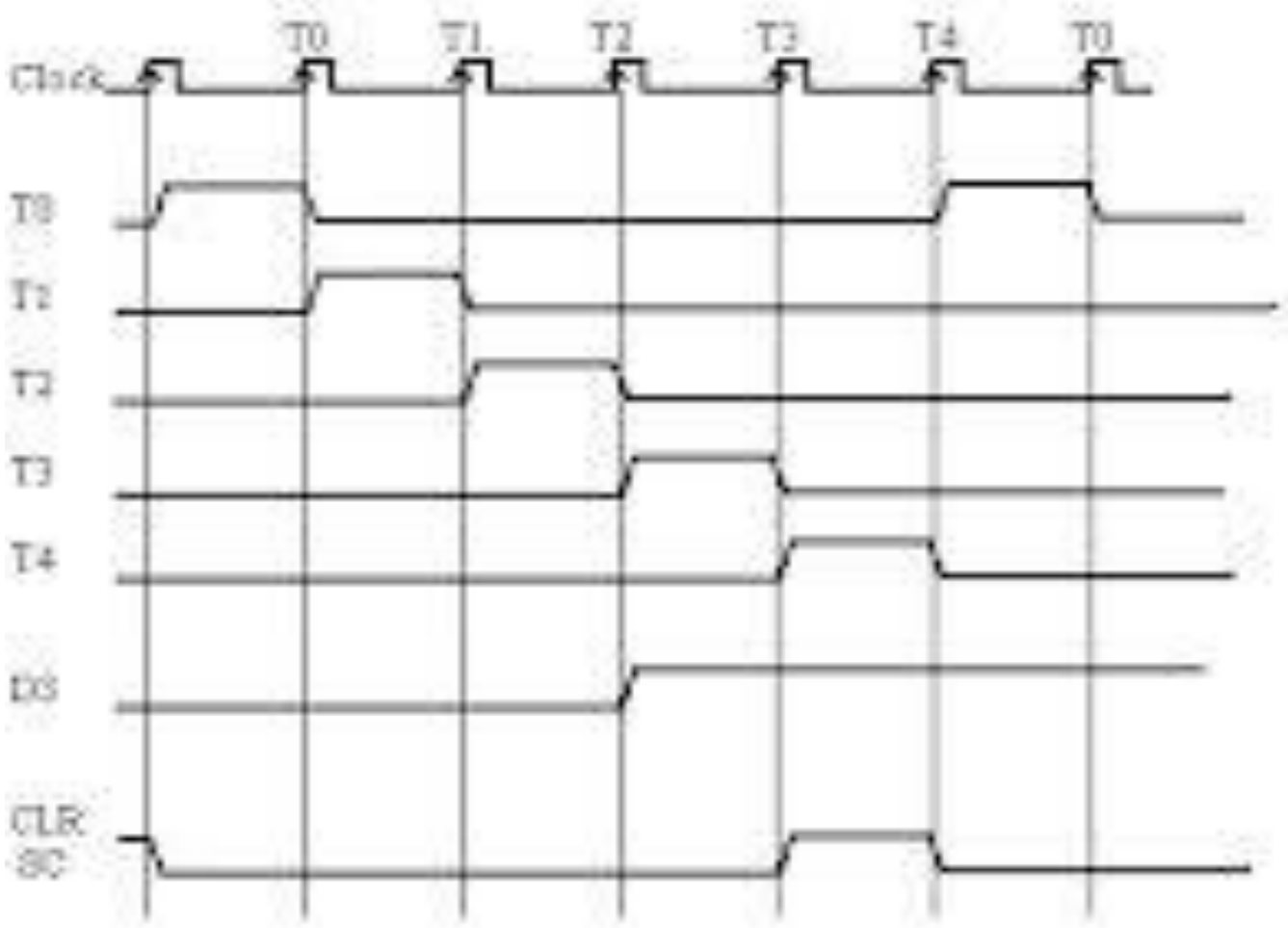
- Clock pulses
  - A master clock generator controls the timing for all registers in the basic computer
  - The clock pulses are applied to all F/Fs and registers in system
  - The clock pulses do not change the state of a register unless the register is enabled by a control signal
  - The control signals 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
  - Hardwired Control :
    - » 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)

- Microprogrammed Control : *Chap. 6*
  - » 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





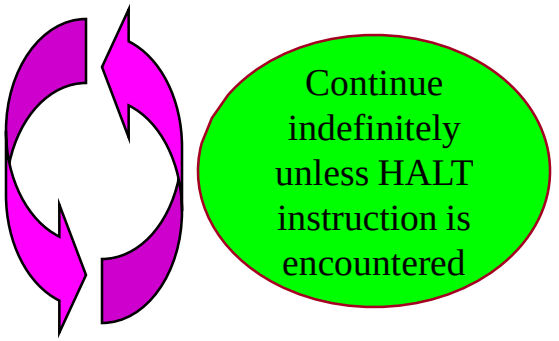
Example of control timing signals

- 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 $(S_2S_1S_0)$
    - » 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 Cycle
  - 1) Instruction Fetch from Memory
  - 2) Instruction Decode
  - 3) Read Effective Address(if indirect addressing mode)
  - 4) Instruction Execution
  - 5) Go to step 1) : Next Instruction[PC + 1]



- Instruction Fetch : T0, T1

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

  - 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

- 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

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

Op.code **Address** **Di/Indirect**

- **IR(12-14) *Fig. 5-6* D0 - D7**

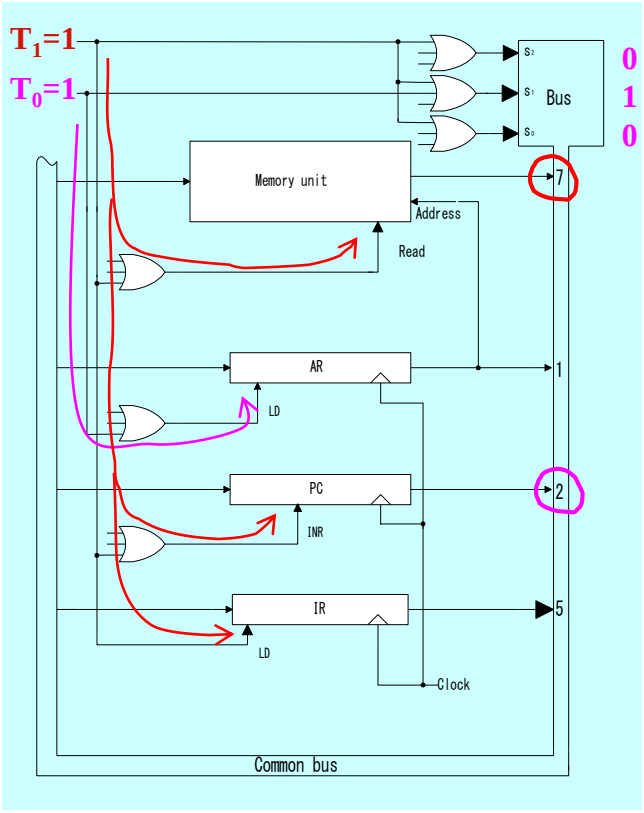
Instruction Execution : T3, T4, T5, T6

$IR(12-14) = 111$

- 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

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



- Register Ref. Instruction
  - r = $D_7 I' T_3$:
  - IR(i) = B_i ← IR(0 -11)
  - ***B_0 - B_{11}*** : **12 Register Ref. Instruction**

Address

5-6 Memory Ref. Instruction

D_7 : Register or I/O = 1

3 X 8 Decoder

D_6 - D_0 : **7 Memory Ref. Instruction**

IR(12,13,14) = 111

- AND to AC

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

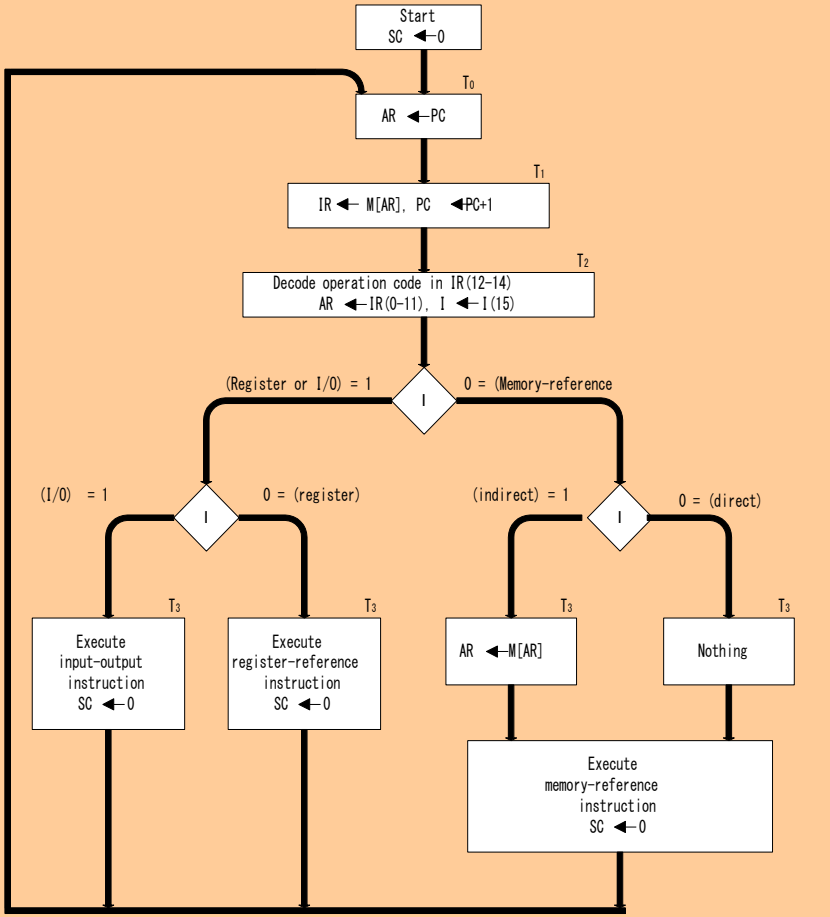
- ADD to AC

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

- 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)



- 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

$$D_5T_4 : M[AR] \leftarrow PC, AR \leftarrow AR + 1$$
$$D_5T_5 : PC \leftarrow AR, SC \leftarrow 0$$

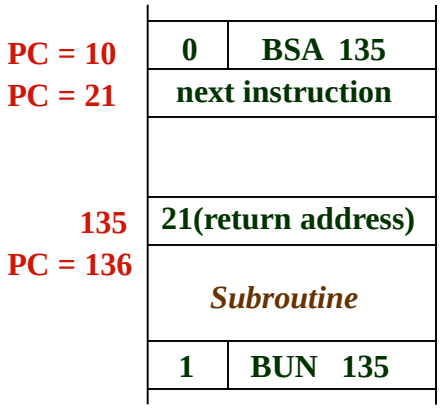
  - Return Address : save return address (135 ← 21)
  - Subroutine Call : ***Fig. 5-10***

$$D_5T_4 : M[135] \leftarrow 21(PC), 136(AR) \leftarrow 135 + 1$$
$$D_5T_5 : 136(PC) \leftarrow 136(AR), SC \leftarrow 0$$

- ISZ : increment and skip if zero

$$D_6T_4 : DR \leftarrow M[AR]$$
$$D_6T_5 : DR \leftarrow DR + 1$$
$$D_6T_6 : M[AR] \leftarrow DR, if\ (DR = 0)\ then\ (PC \leftarrow PC + 1), SC \leftarrow 0$$

Fig. 5-10 Example of BSA



5-7 Input-Output and Interrupt

Input-Output Configuration :

- 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, ***clear*** when INPR is empty
  - » FGO : ***set*** when operation is completed, ***clear*** when output device is in the process of printing

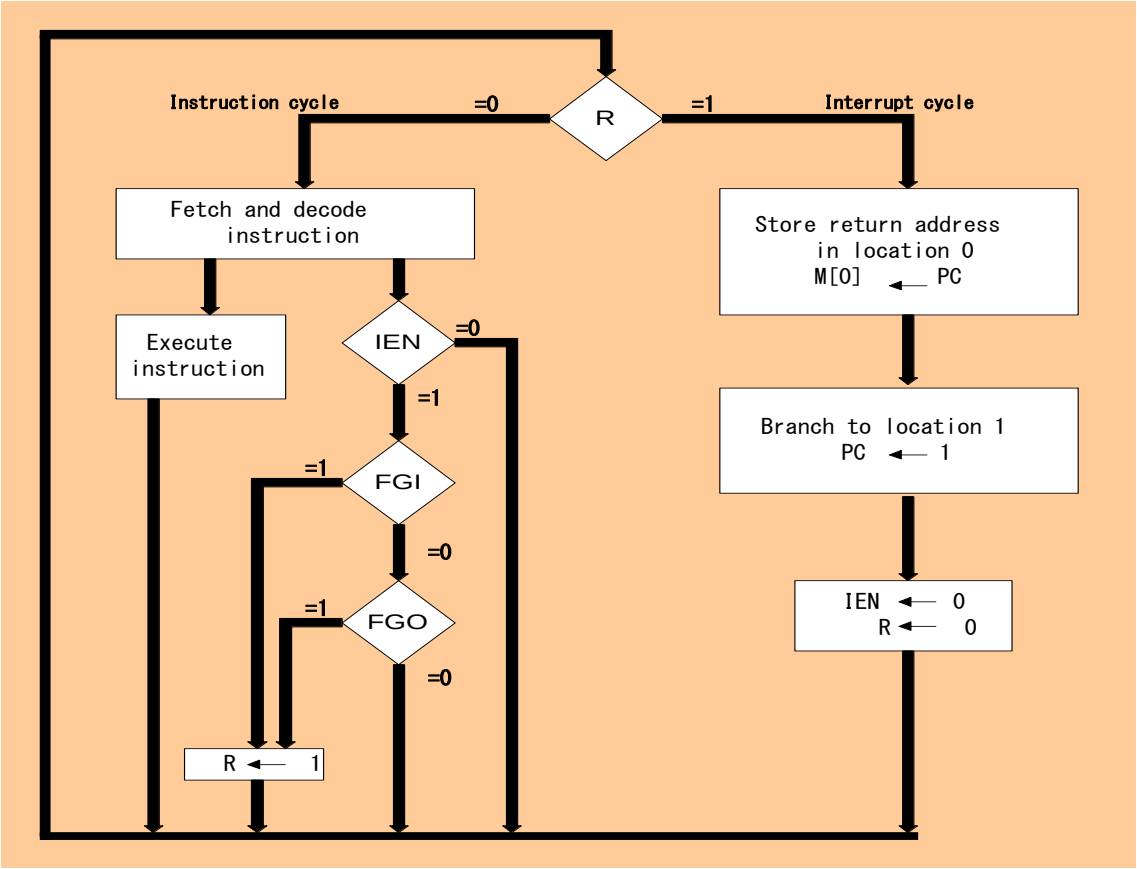
Input-Output Instruction : *Tab. 5-5*

- p = D_7IT_3 : Common to all I?O instructions
- IR(i) = B_i ⟵ IR(6 -11) : B (bit in IR(6-11) that specifies the instruction)
- ***B_6 - B_{11}*** : **6 I/O Instruction**

Program Interrupt

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

- 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



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
- 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
- 5. A 16-bit common bus
- ***6. Control Logic Gates***
- ***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

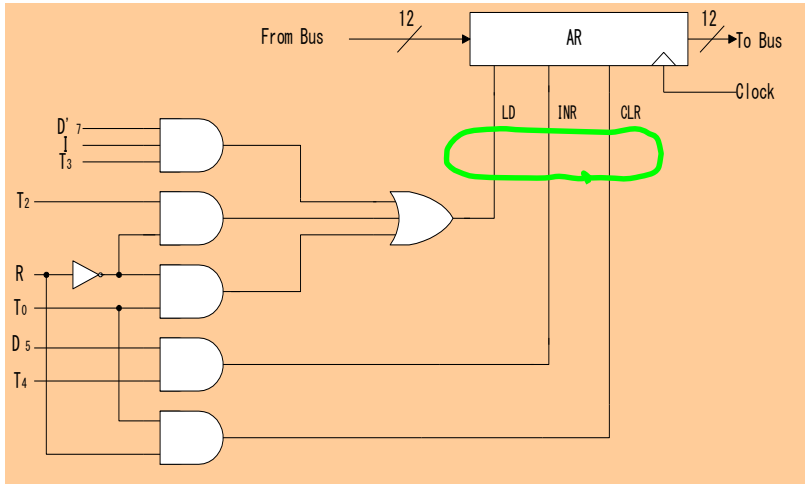
Register Control : AR

- Control inputs of AR : ***LD, INR, CLR***
- ***Find all the statements that change the AR in Tab. 5-6*** → $R'T_0 : AR \leftarrow PC$, $R'T_1 : AR \leftarrow IR(0-11)$, $D_7'IT_3 : AR \leftarrow M[AR]$, $RT_0 : AR \leftarrow 0$, $D_5T_4 : AR \leftarrow AR+1$

$AR \leftarrow ?$

- 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

$$READ = R'T_1 + D_7'IT_3 + (D_0 + D_1 + D_2 + D_6)T_4$$

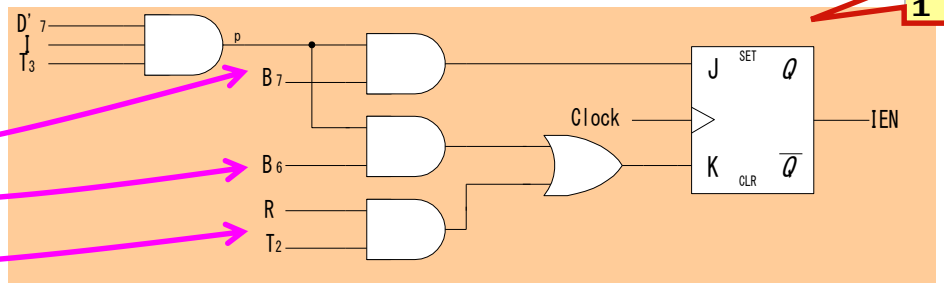
F/F Control : IEN

$IEN \leftarrow ?$

- Control functions

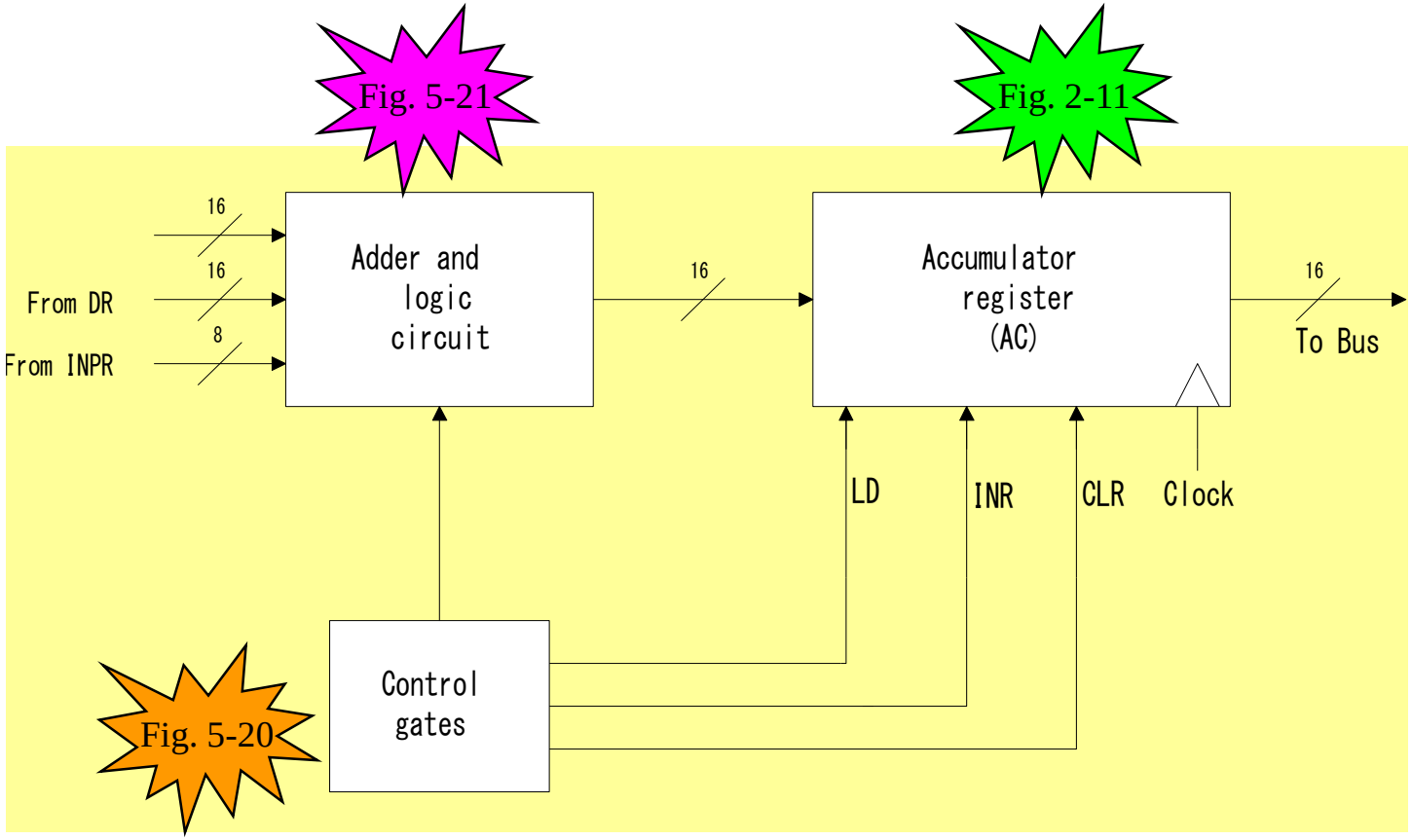
$$pB_7 : IEN \leftarrow 1$$
$$pB_6 : IEN \leftarrow 0$$
$$RT_2 : IEN \leftarrow 0$$

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



5-10 Design of Accumulator Logic

Circuits associated with AC :



Control of AC : *Fig. 5-20*

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

$D_0T_5 : AC \leftarrow AC \wedge DR$

$D_1T_5 : AC \leftarrow AC + DR$

$D_2T_5 : AC \leftarrow DR$

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

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

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

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

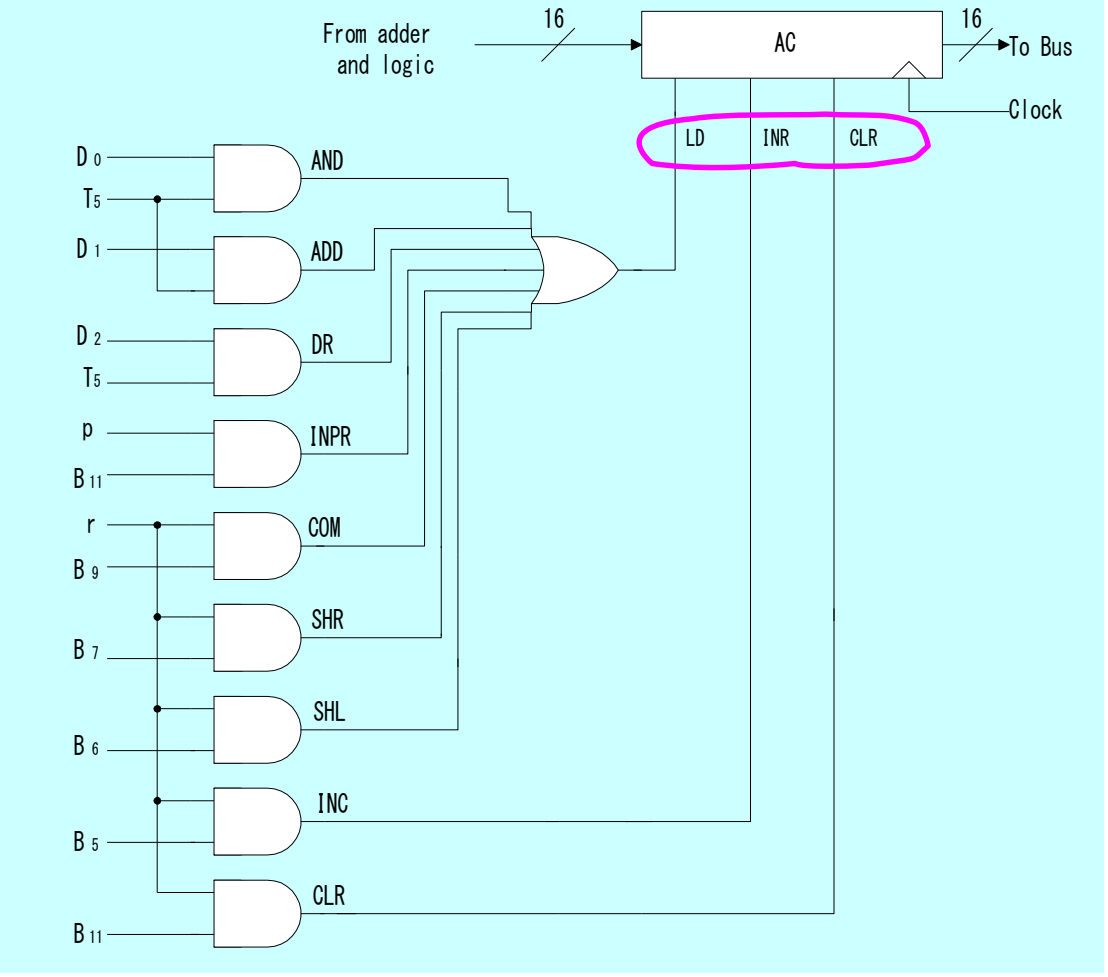
$rB_{11} : AC \leftarrow 0$

$rB_5 : AC \leftarrow AC + 1$

LD

CLR

INR



Adder and Logic Circuit : *Fig. 5-21*