Computer Architecture Lec 5a

Dr. Esti Stein

(Partly taken from Dr. Alon Schclar slides)

Based on slides by:
Prof. Myung-Eui Lee
Korea University of Technology & Education
Department of Information & Communication

Taken from: **M. Mano/Computer Design and Architecture 3rd Ed.**

An Instruction

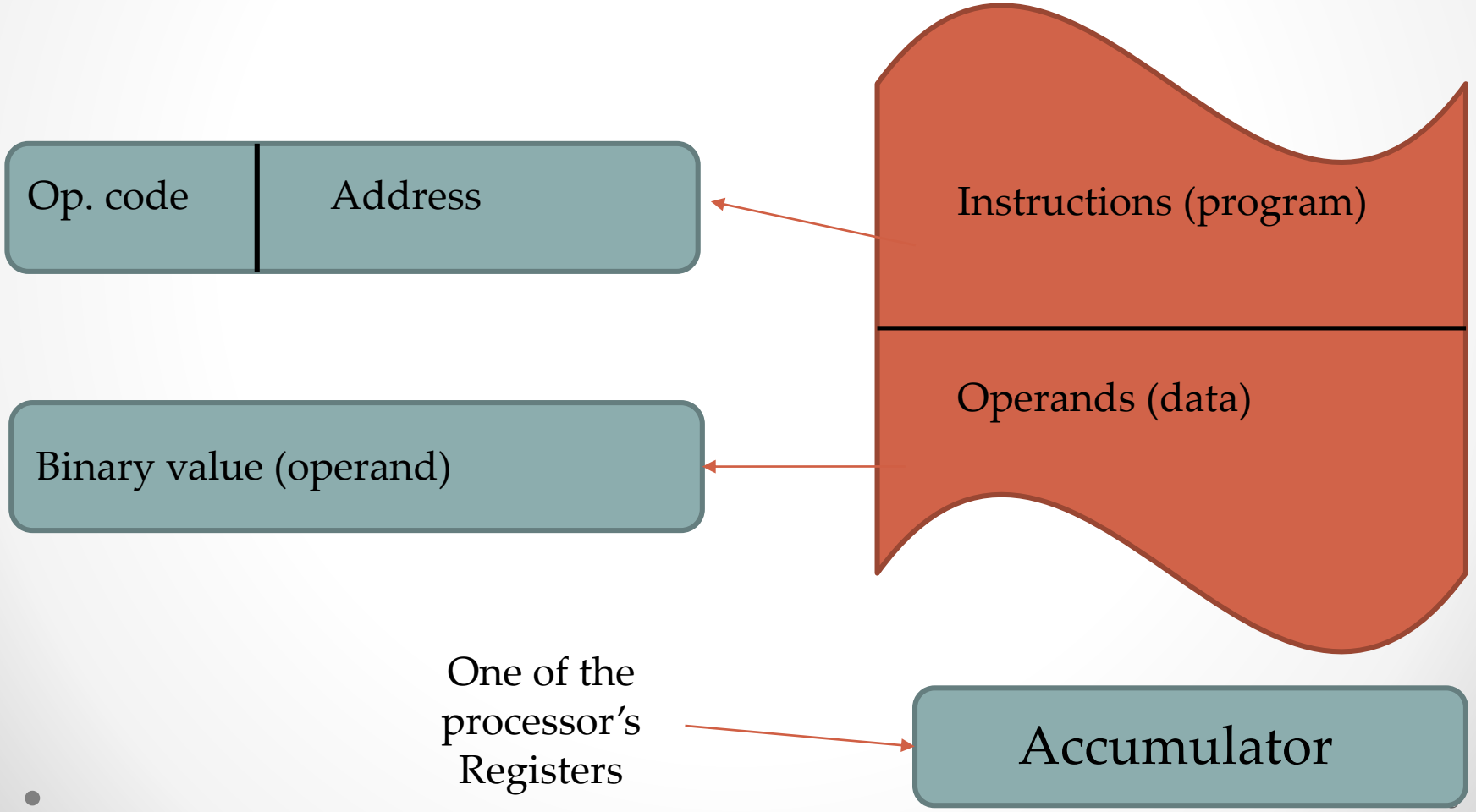
- A group of bits that instructs the computer to perform a specific operation.
- An instruction is usually divided into parts.

A group of bits that defines an operation:
add, subtract, shift, and etc.

An address in the memory

n bits define 2^n operations

A Stored Program Organization



The Instruction

All instructions and operands are 16 bits

4 bits ⇒ 16 possible operation codes

12 bits ⇒ *length* of memory is $2^{12} = 4096$

Size of memory is 4096 × 16

If an operation in the instruction does not need a memory operand, the rest of the bits can be used to expand the operation.
Examples:
clear accumulator, complement accumulator, read a character from the keyboard, etc.

Addressing Modes

Immediate addressing mode

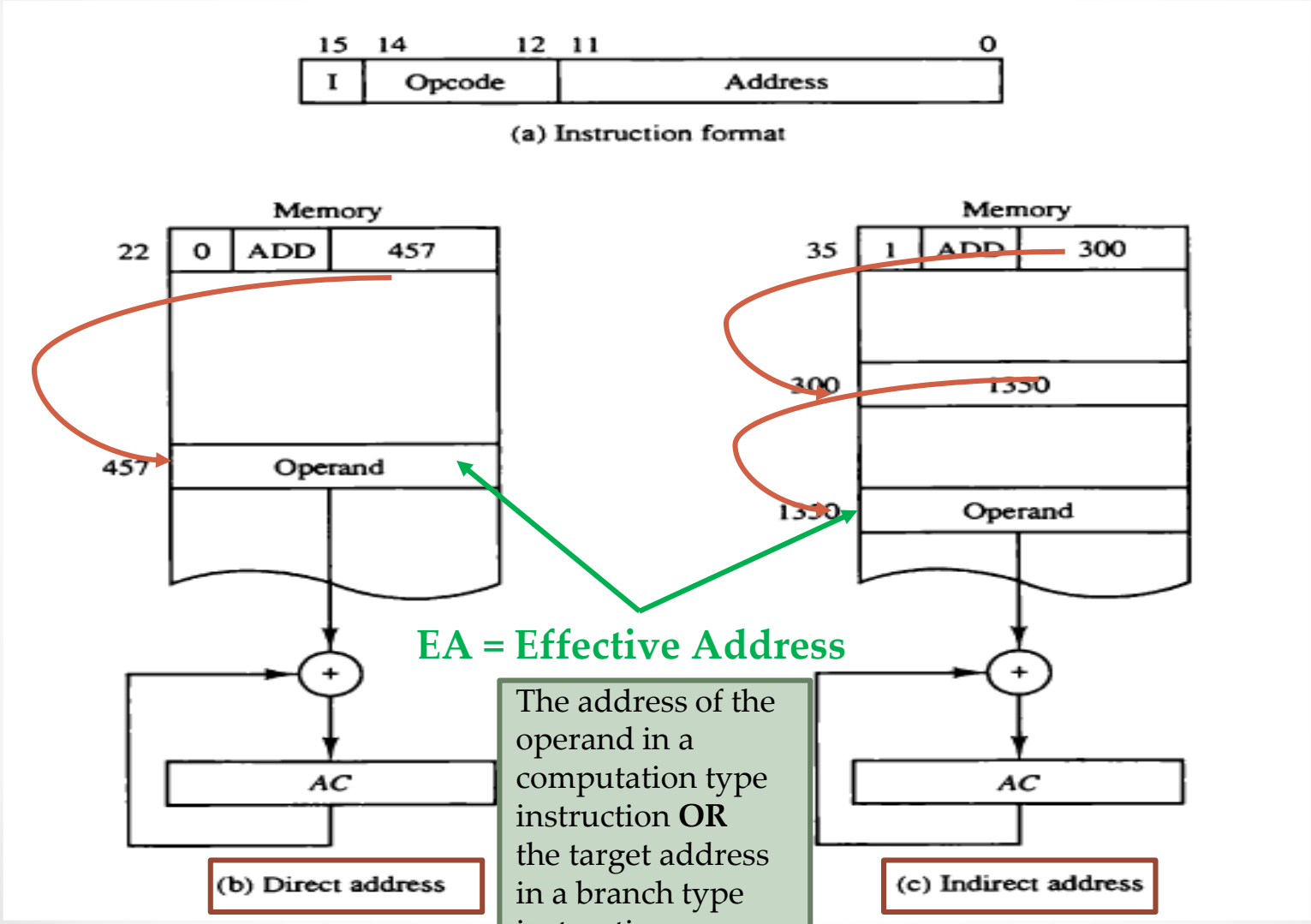
Direct addressing mode

Indirect bit

3 bits remains ⇒ 8 possible operations

Indirect addressing mode

Addressing Modes



The address of the operand in a computation type instruction **OR** the target address in a branch type instruction

Computer Registers

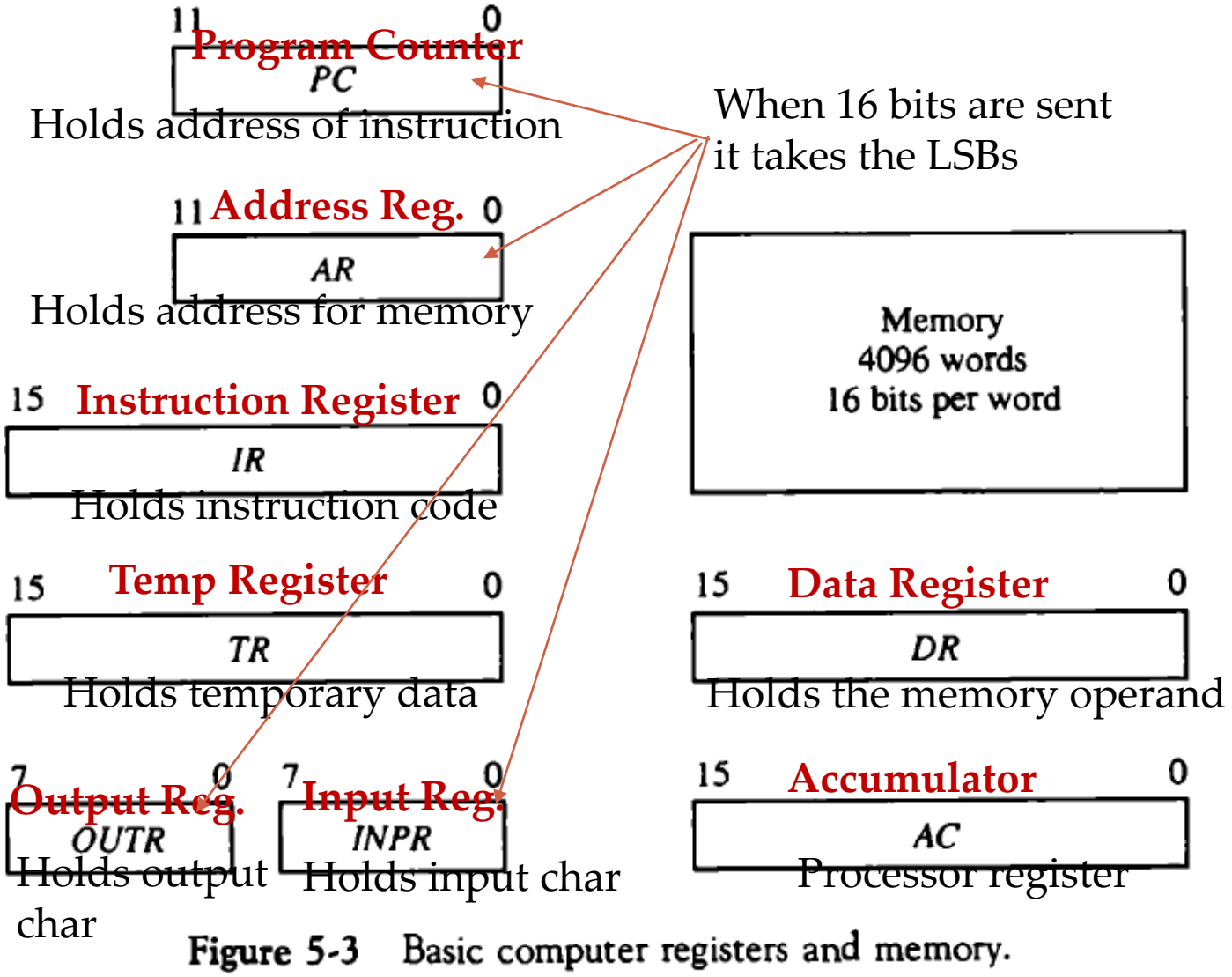


Figure 5-3 Basic computer registers and memory.

Register with INC, LD, CLR

AC
AR
DR
PC
TR

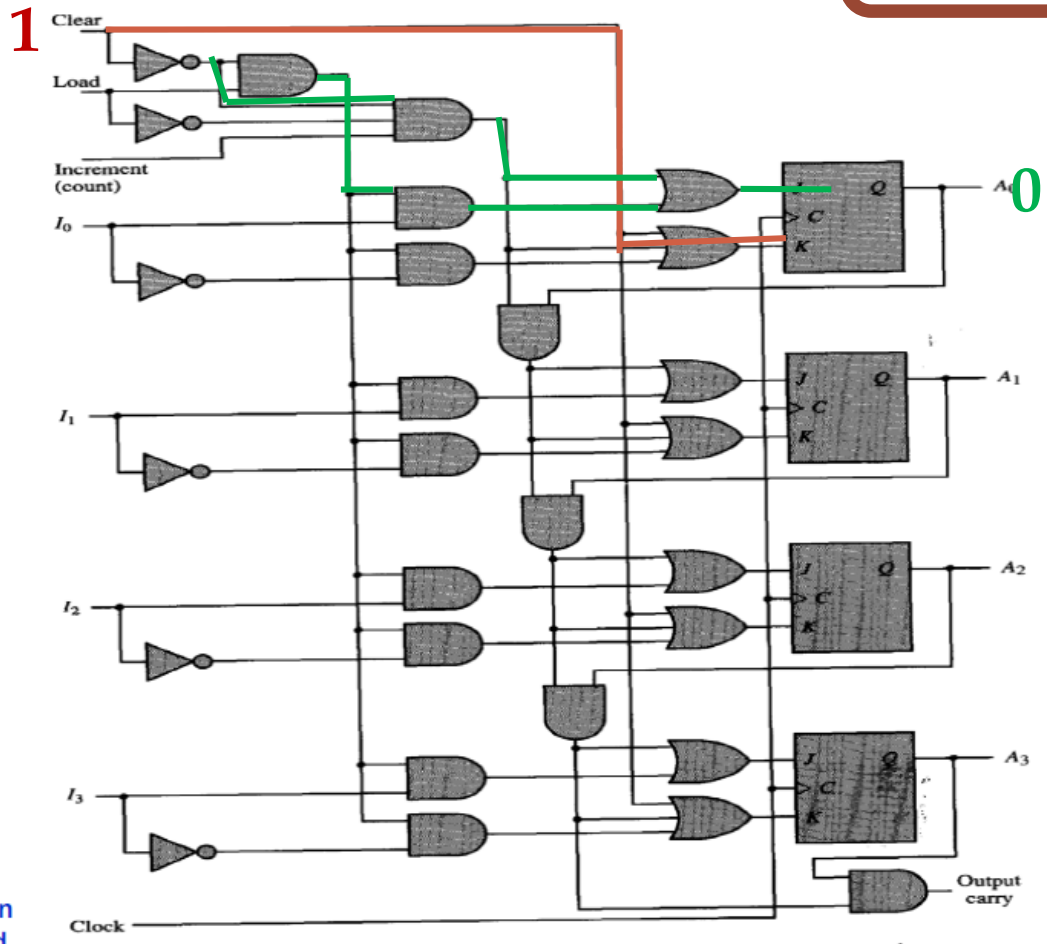


Figure 2-11 4-bit binary counter with parallel load and synchronous clear.

Taken from: M. Mano/Computer Design and Architecture 3rd Ed.

Register with INC, LD, CLR

AC
AR
DR
PC
TR

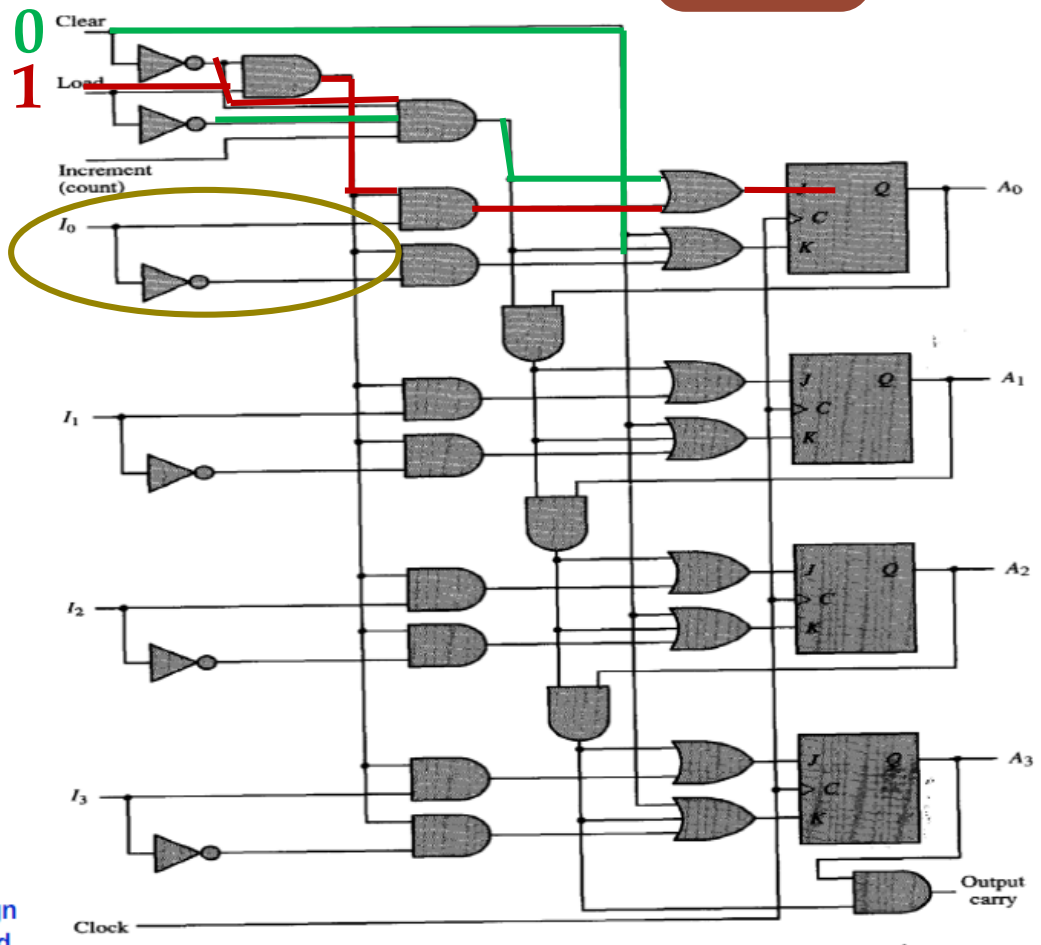


Figure 2-11 4-bit binary counter with parallel load and synchronous clear.

Taken from: M. Mano/Computer Design and Architecture 3rd Ed.

Register with INC, LD, CLR

AC
AR
DR
PC
TR

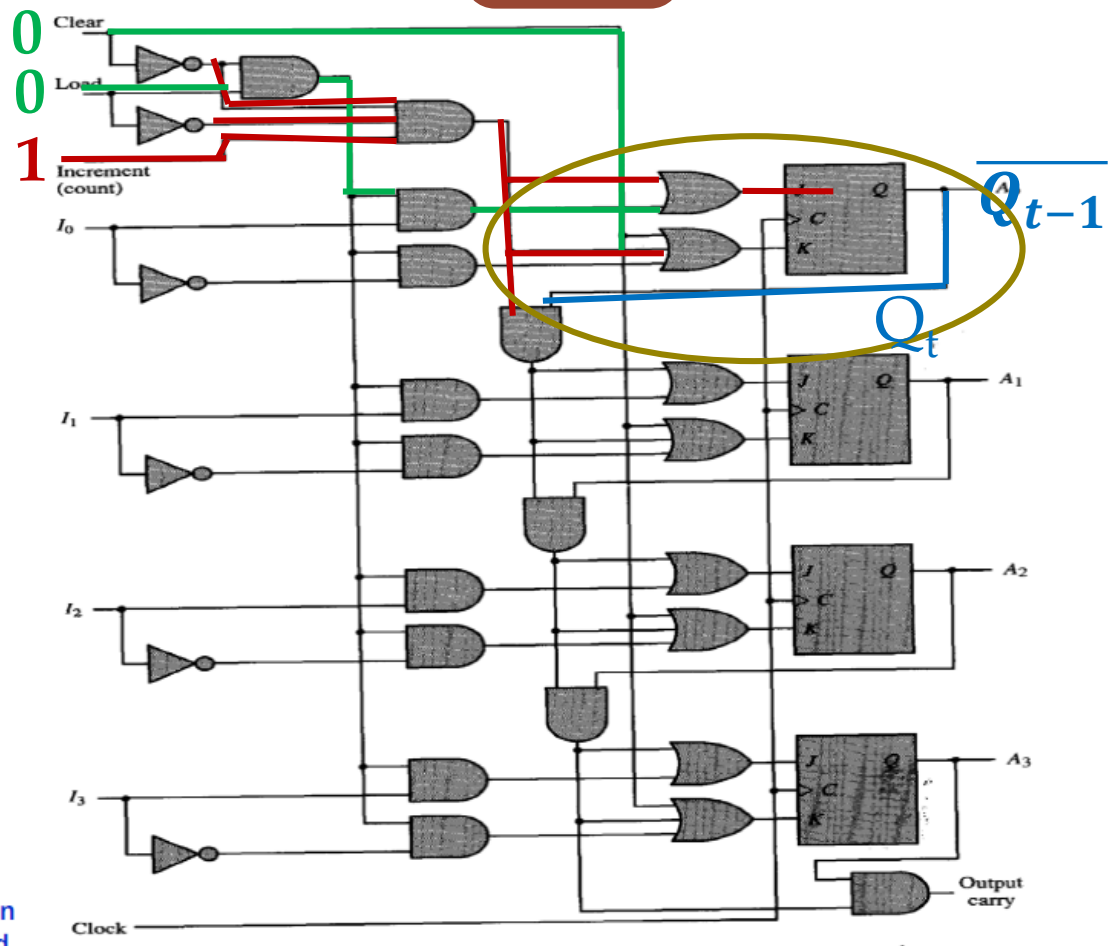


Figure 2-11 4-bit binary counter with parallel load and synchronous clear.

Taken from: M. Mano/Computer Design and Architecture 3rd Ed.

The Bus

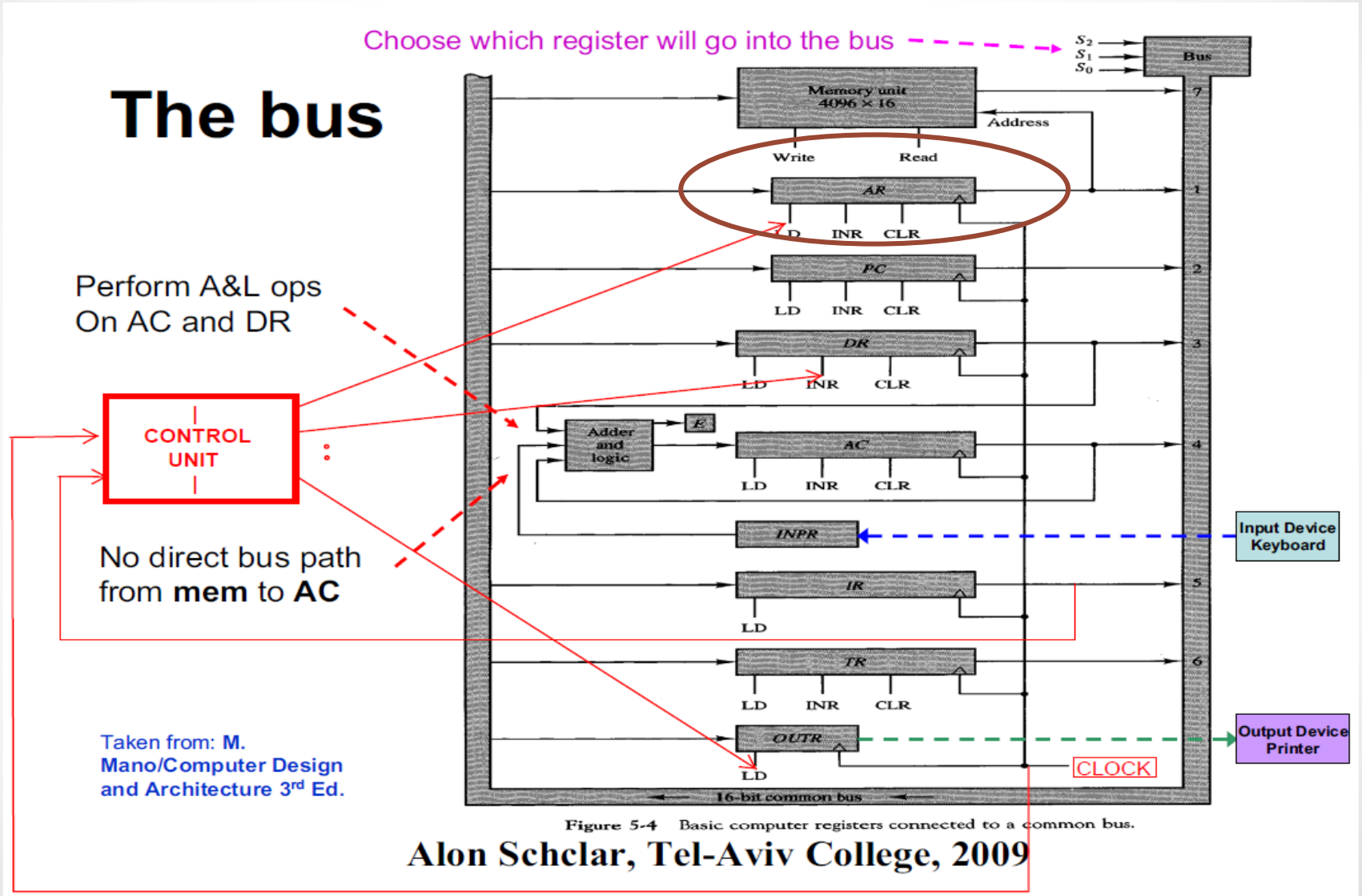


Figure 5-4 Basic computer registers connected to a common bus.

Reminder – register with parallel load

IR
OUTR

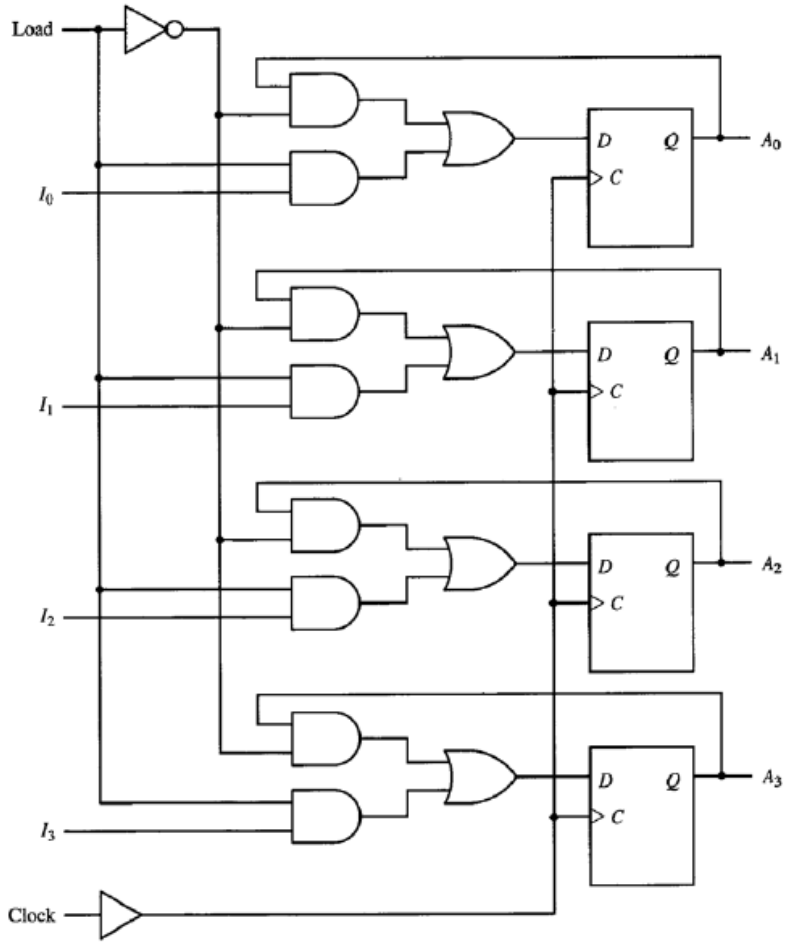
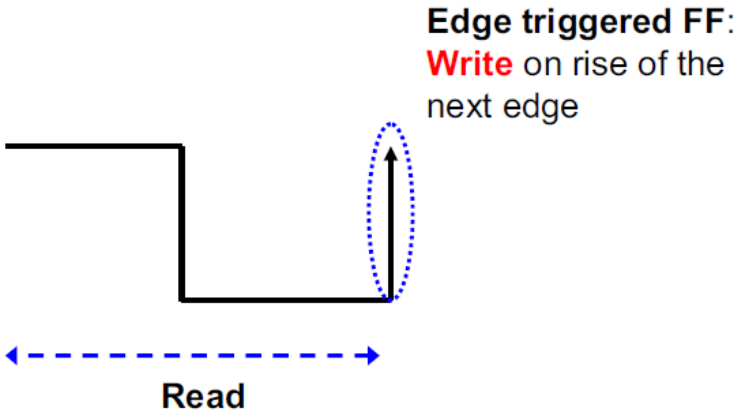


Figure 2-7 4-bit register with parallel load.

Taken from: **M. Mano/Computer Design and Architecture 3rd Ed.**

Alon Schclar, Tel-Aviv College, 2009

The clock cycle



The Bus

» Accumulator(***AC***) : 3 Path

- 1) Register Microoperation : clear AC, shfift AC,…
- 2) Data Register : **add DR to AC, and DR to AC**
  End carry bit set/reset), memory **READ**

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

- 3) INPR : Device
  Adder & Logic

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

$$DR \leftarrow AC : s_2s_1s_0 = 100(4), DR(load)$$
$$AC \leftarrow DR : DR \rightarrow Adder\ \&\ Logic \rightarrow AC(load)$$

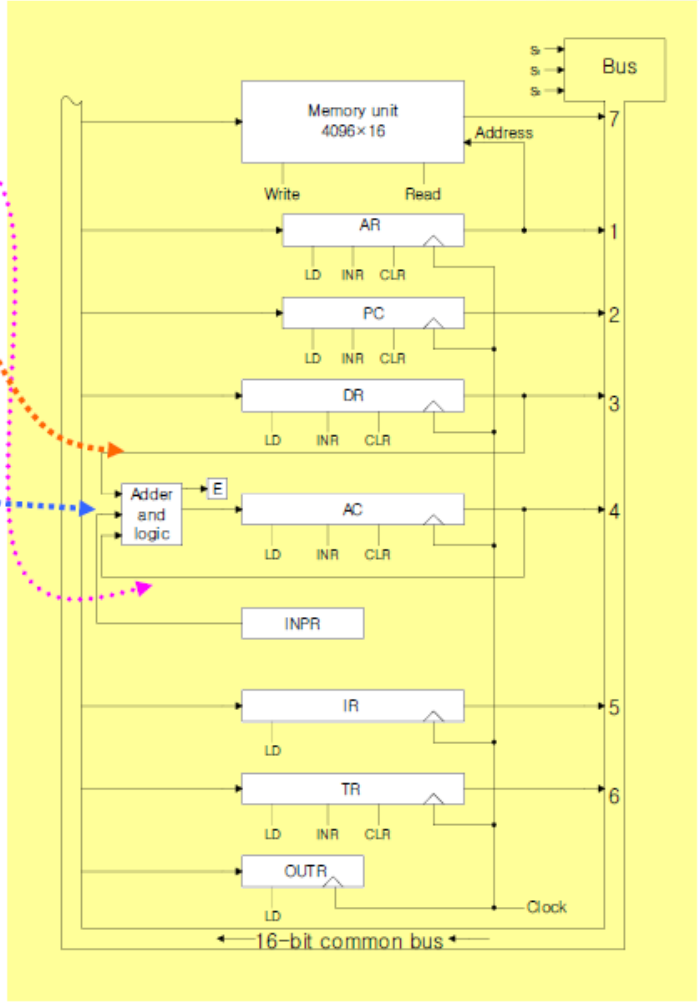
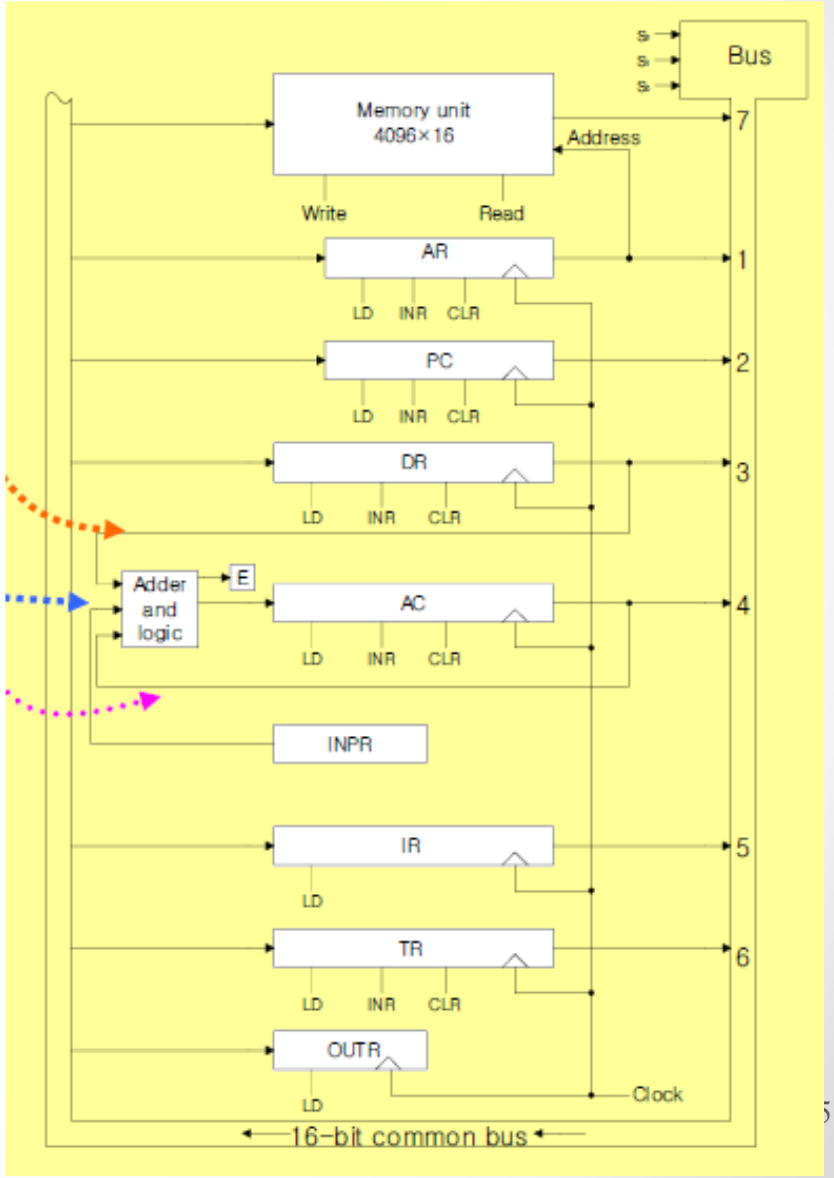


Fig. 5-4 Basic computer registers connected to a common bus

The Bus - examples

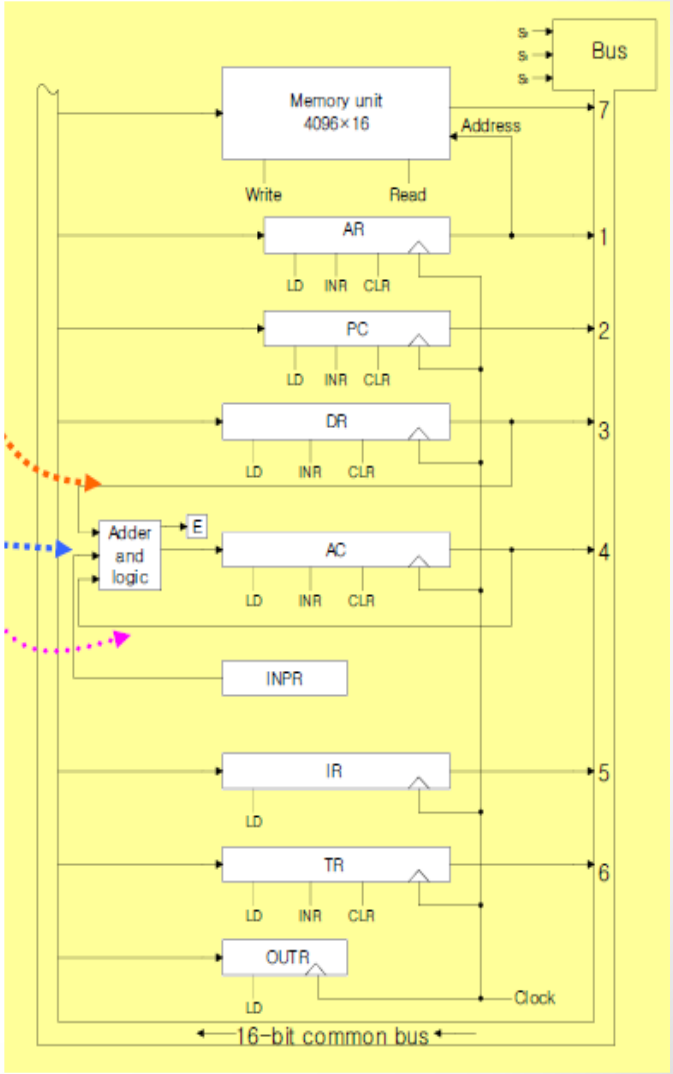
- **TR ←DR**
  - Place DR on BUS
    - $S_2S_1S_0$=011 (DR num is 3)
  - Insert BUS content to TR
    - Enable LD (load) input of TR

- **DR ←MEM[AR]**
  - AR is connected to address input of MEM
  - Enable READ input of MEMORY
  - Place MEMORY content (MEM[AR]) on BUS
    - $S_2S_1S_0$=111 (MEM num is 7
  - Insert BUS content to DR
    - Enable LD (load) input of DR



The Bus – concurrent data transfer

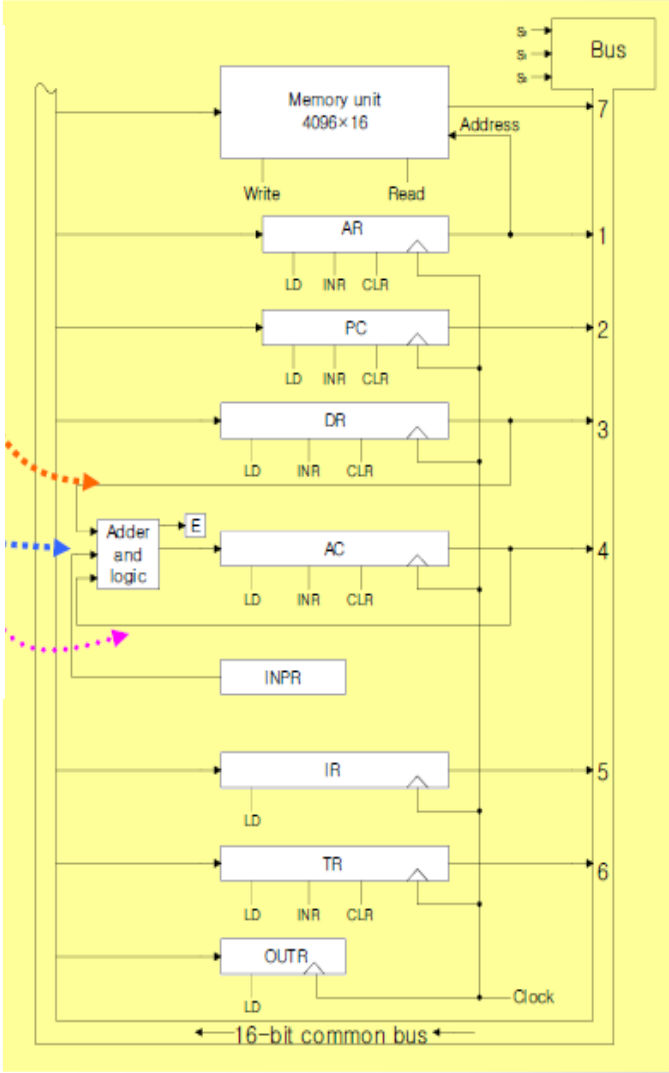
- During the same clock cycle
  - The content of any register can be applied onto the bus and
  - an operation can be performed in the adder and logic circuit
- The clock transition at the end of the cycle
  - transfers the content of the bus into the target register and
  - the output of the adder and logic circuit into AC.



The Bus – concurrent data transfer

- **DR ←AC**
  - Place AC on BUS
    - $S_2S_1S_0$=100 (AC num is 4)
  - Insert BUS content to DR
    - Enable LD (load) input of DR
- **AC ←DR**
  - DR connected to AC via the **Adder & Logic** (aka A&L or ALU)
  - Instruct ALU to let DR pass through
  - Enable LD (load) input of AC

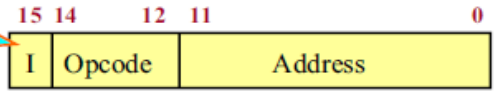
- **Can be executed in the same clock cycle**



Computer Instruction

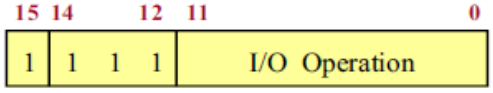
- 5-3 Computer Instruction
  - 3 Instruction Code Formats : ***Fig. 5-5***
    - Memory-reference instruction
      - » Opcode = 000 ~ 110
        - I=0 : 0xxx ~ 6xxx, I=1: 8xxx ~Exxx

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
CMA	7200		Complement AC
CME	7100		Comp m 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		Interrup
IOF	F040		Inter

Instruction Set Completeness

A computer should have a set of instructions so that the user can evaluate any function that is known to be computable.

A complete set must include sufficient number of instructions from the following categories:

1. Arithmetic, logical, and shift - CMA, INC, CIR, AND..
2. Moving information between the registers, and between the registers and memory – LDA, STA
3. Program control, and status check – BUN, ISZ, BSA..
4. Input and output – INP, OUT..

A complete set of instructions, but not efficient.

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
    - » 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 (This chapter)

- 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)

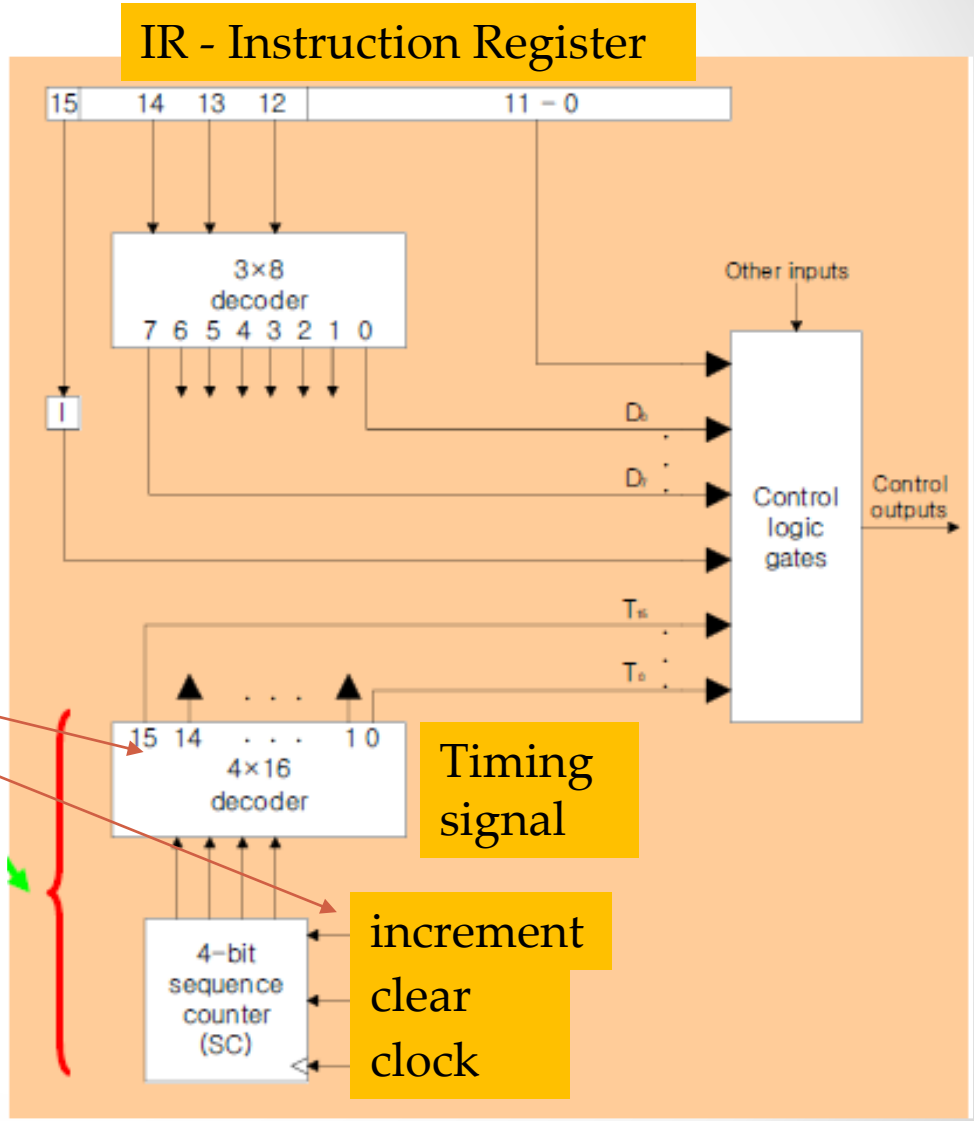
Microprogrammed Control (chapter 7)

- 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

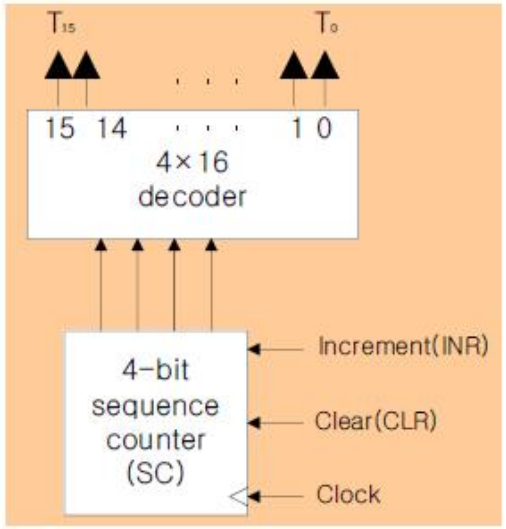
The Control Unit

The Sequence Counter (SC) is incremented synchronously:
$T_0, T_1, T_2, T_3, ..$

The SC is cleared when in RTL we write symbolically:
$SC \leftarrow 0$



The Control Unit



$D_3T_4 : SC \leftarrow 0$

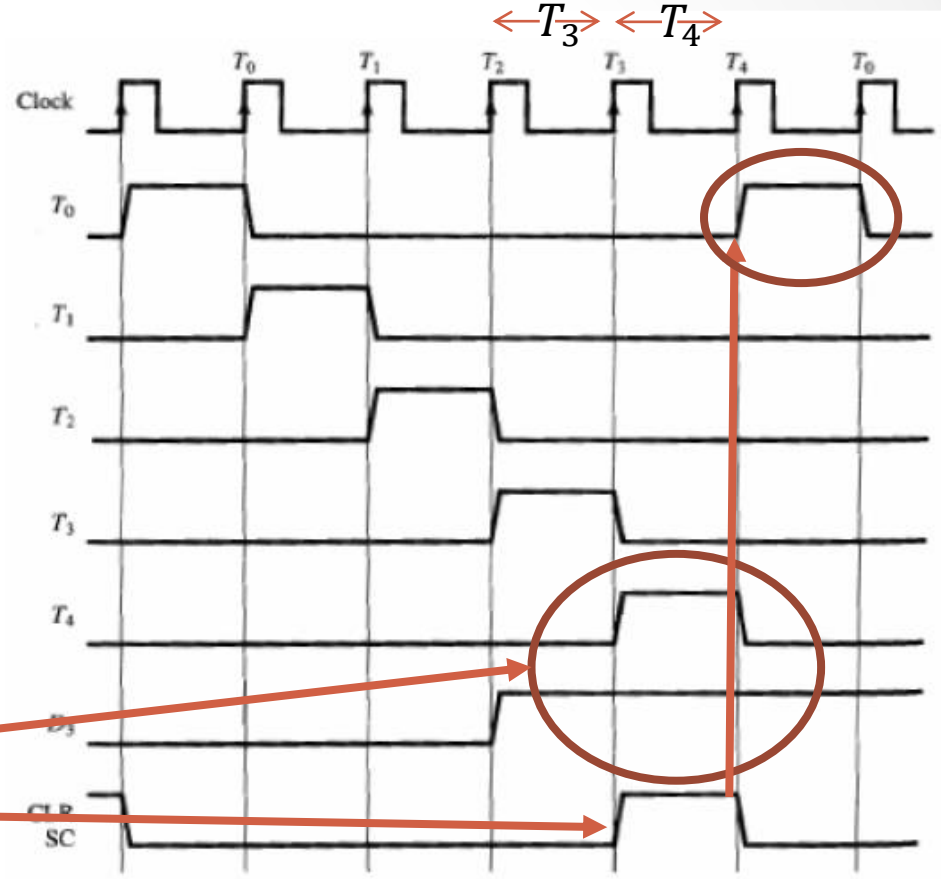


Figure 5-7 Example of control timing signals.

Taken from: **M. Mano/Computer Design and Architecture 3rd Ed.**

Clock and Memory

- The memory read/write cycle is initiated with the rising edge of a timing signal.
- It is assumed that a memory cycle time is less than the clock cycle time.
- The memory R/W cycle is complete by the time the next clock goes through its positive cycle.
- This assumption is not valid in most of computers.

Clock and Timing Signals

Example:

$T_0 : AR \leftarrow PC$

1. T_0 is active during the entire clock cycle interval.
2. During this time the content of the PC is placed onto the bus, and the LD of AR is enabled.
3. The actual transfer occurs when the clock goes through a positive transition (end of cycle).
4. On positive transition SC goes from 0000 to 0001.
5. T_1 is active and T_0 is inactive.