Computer Architecture Lec 5a

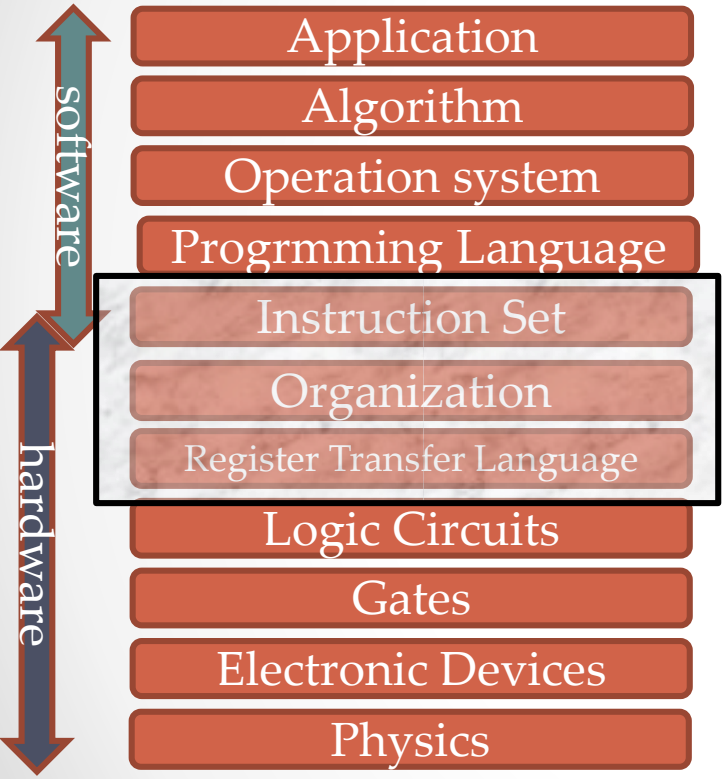
Dr. Esti Stein

(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.**

General Purpose Digital Computer



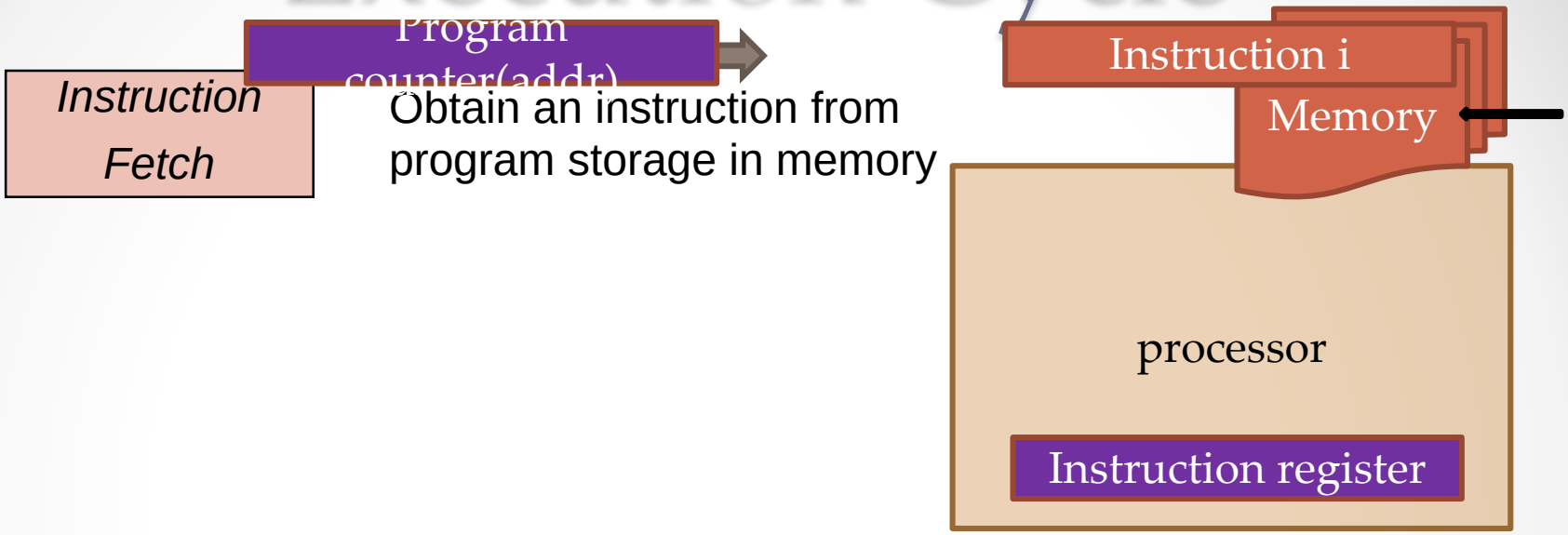
- Capable of executing various microoperations.
- Can be instructed as to what specific sequence of operations to perform.

A Program

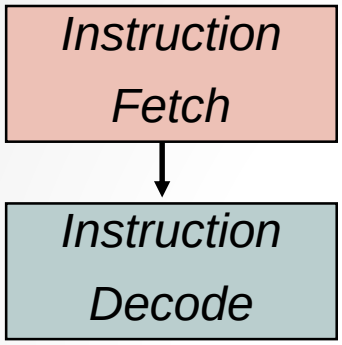
- The user of a computer can control the process by means of a ***program.***
- A program is a set of ***instructions*** that specify the operations, operand, and the sequence *(control)*
- A instruction is a binary code that specifies a sequence of microoperations

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

Execution Cycle

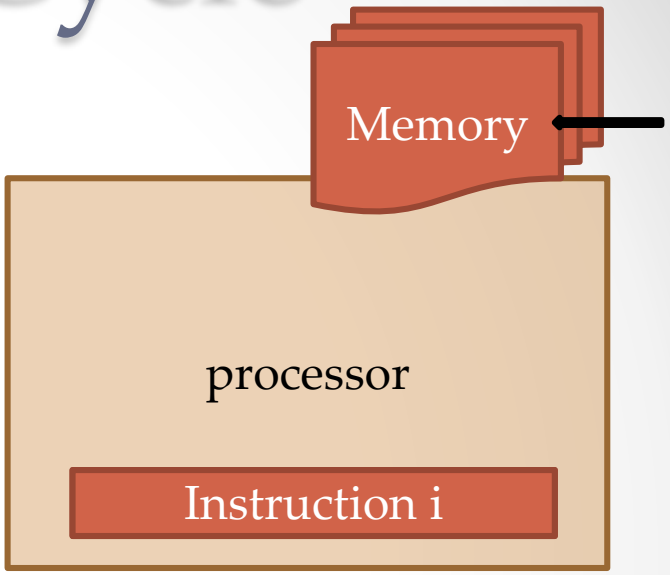


Execution Cycle

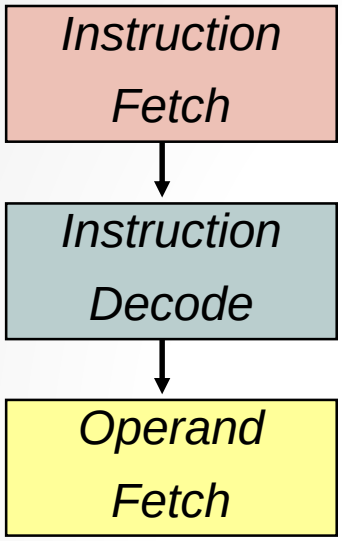


Obtain instruction from program storage in memory

Determine required actions and instruction size



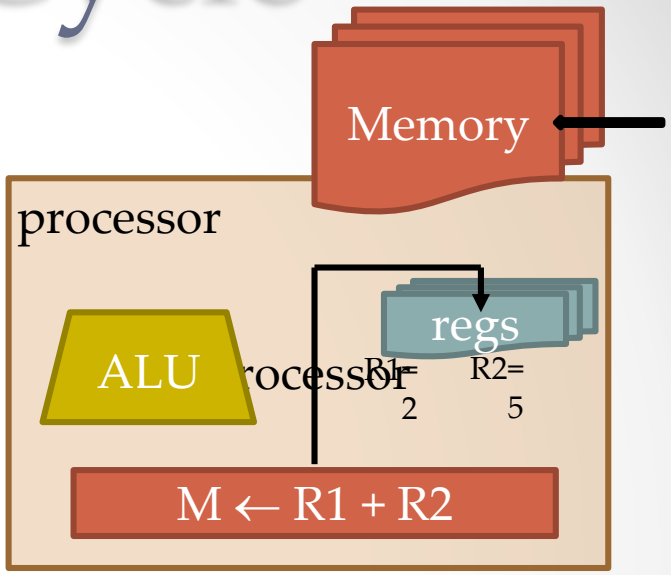
Execution Cycle



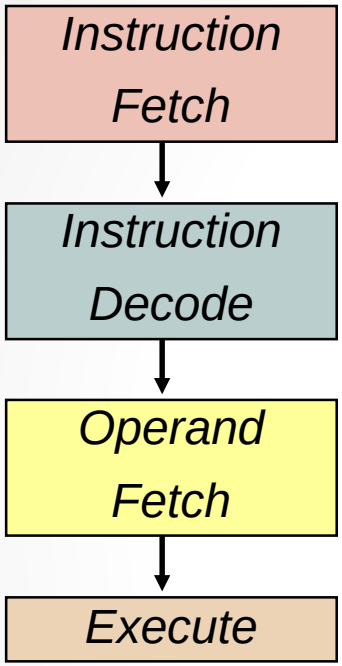
Obtain instruction from program storage in memory

Determine required actions and instruction size

Locate and obtain operand data



Execution Cycle

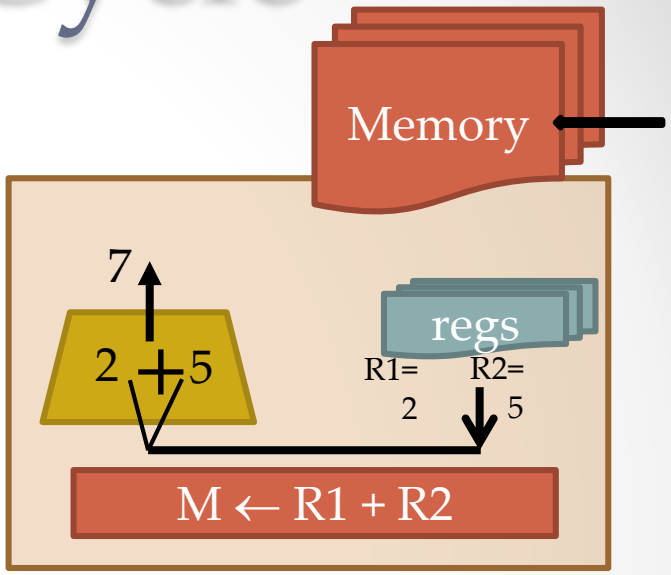


Obtain instruction from program storage in memory

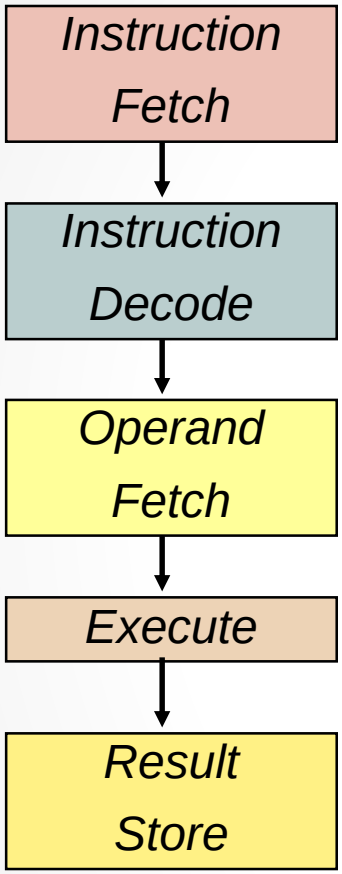
Determine required actions and instruction size

Locate and obtain operand data

Compute result value or status



Execution Cycle



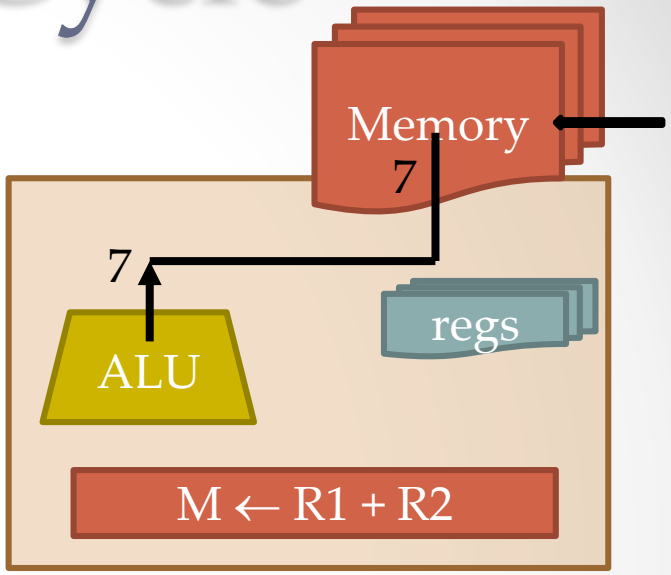
Obtain instruction from program storage in memory

Determine required actions and instruction size

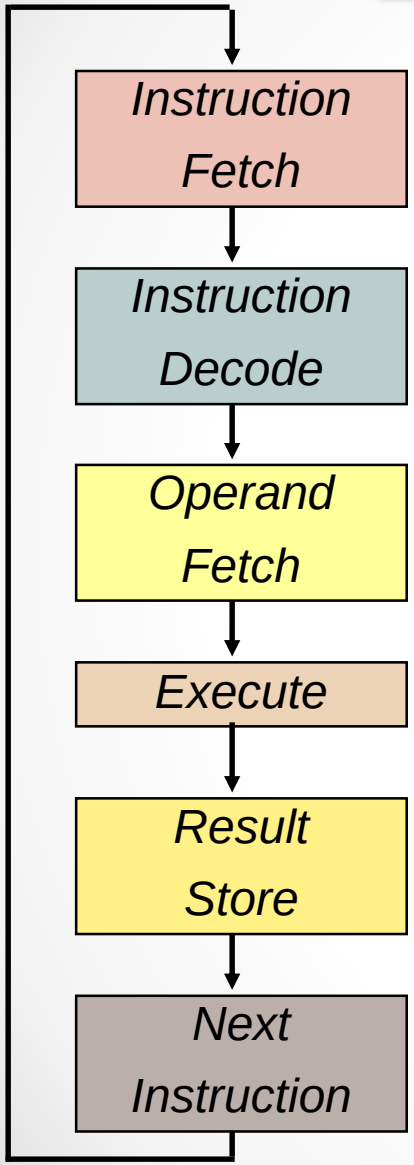
Locate and obtain operand data

Compute result value or status

Deposit results in storage



Execution Cycle



Obtain instruction from program storage in memory

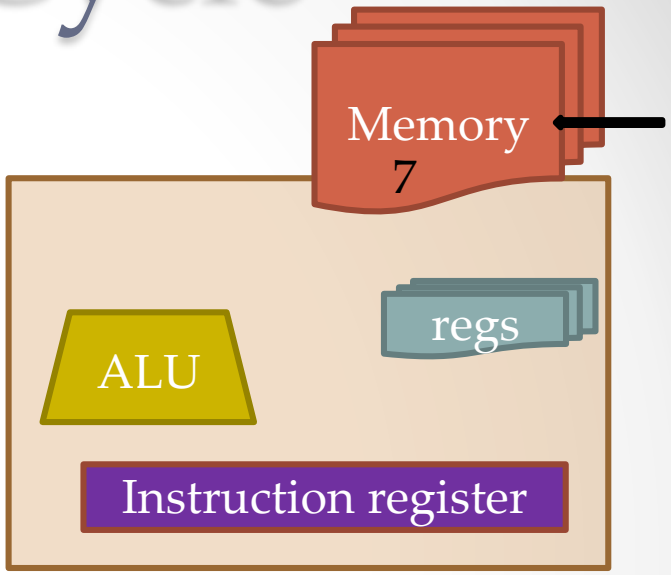
Determine required actions and instruction size

Locate and obtain operand data

Compute result value or status

Deposit results in storage

Determine next instruction (not the next in case of **branch**)



An Instruction

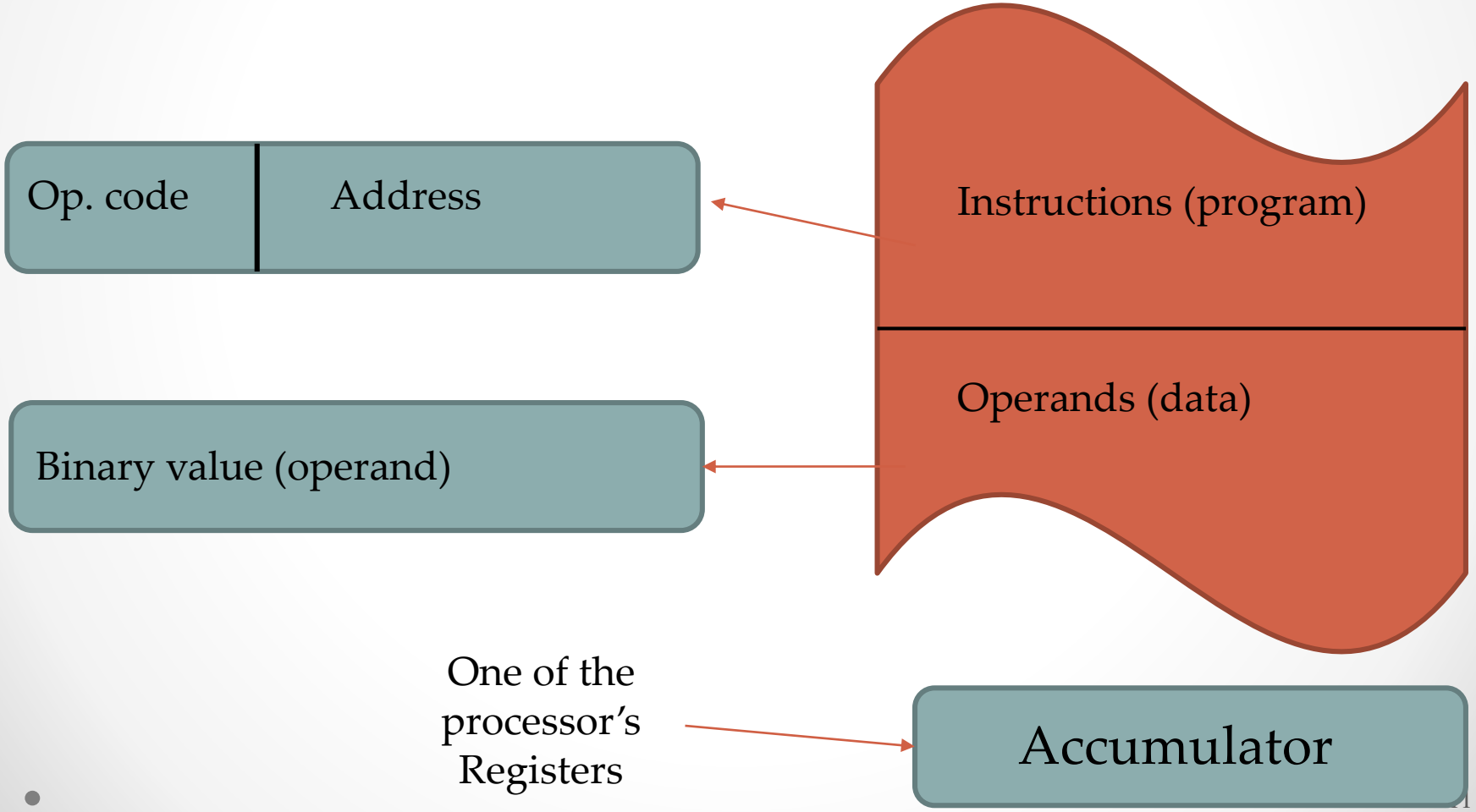
- A group of bits that instructs the computer to perform a specific operation.
- An instruction is usually divided into parts.

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

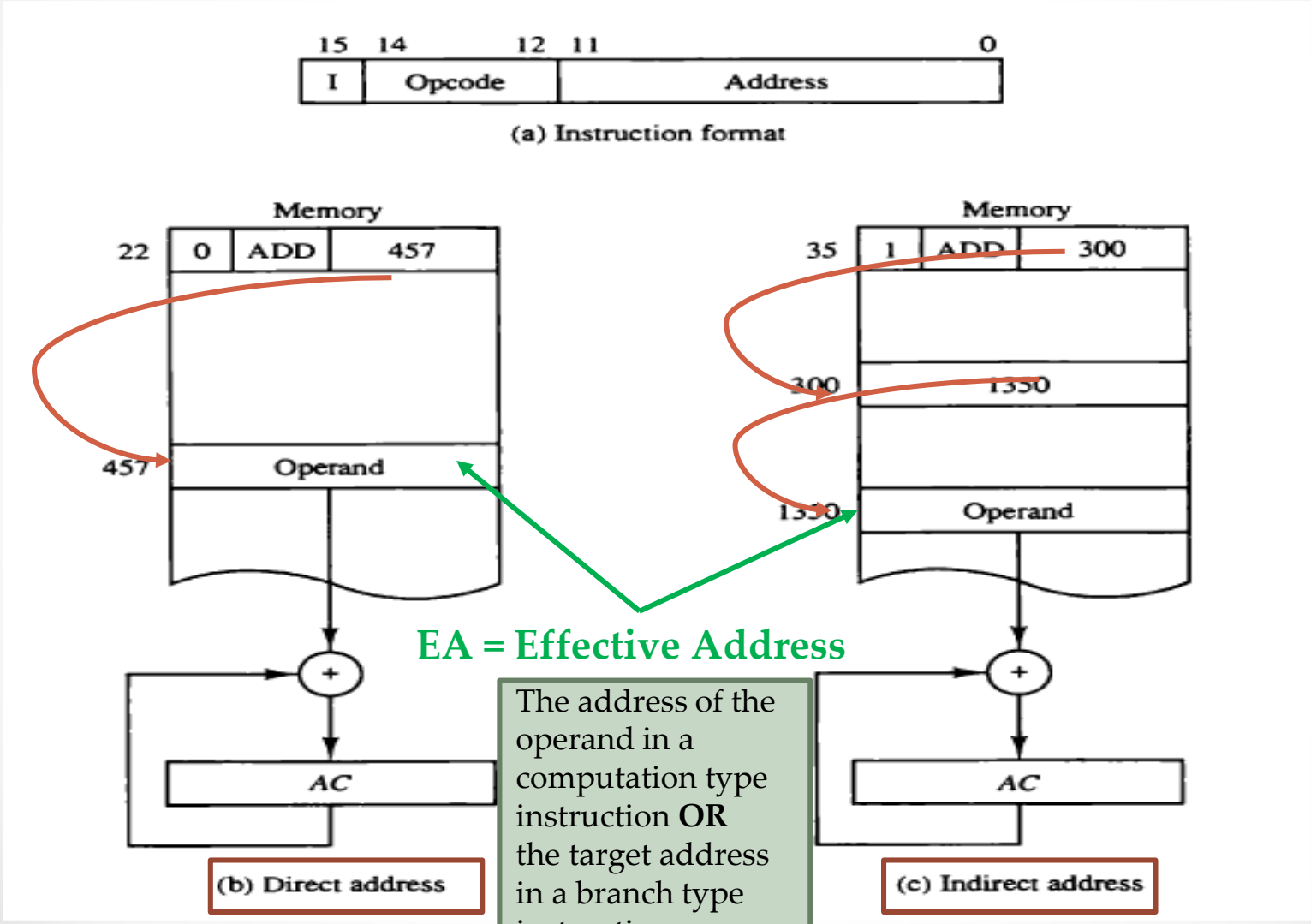
Direct addressing mode

Indirect bit

3 bits remains ⇒ 8 possible operations

Indirect addressing mode

Addressing Modes



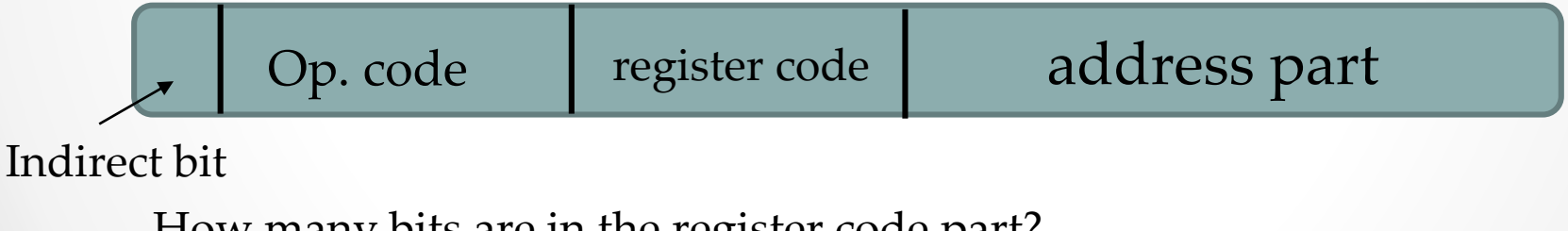
The address of the operand in a computation type instruction **OR** the target address in a branch type instruction

QUIZ1

A computer uses memory unit with 256K words of 32 bits each.
A binary instruction code is stored in one word of the memory.

What is the length (in bits) of one instruction?

The instruction has four parts:
1. An indirect bit
2. An operation code
3. A register code part, to specify one of 64 registers.
4. An address part



How many bits are in the register code part?

How many bits are in the address part?

How many bits are in the operation code part?

How many bits are in a data operand?

Computer Registers

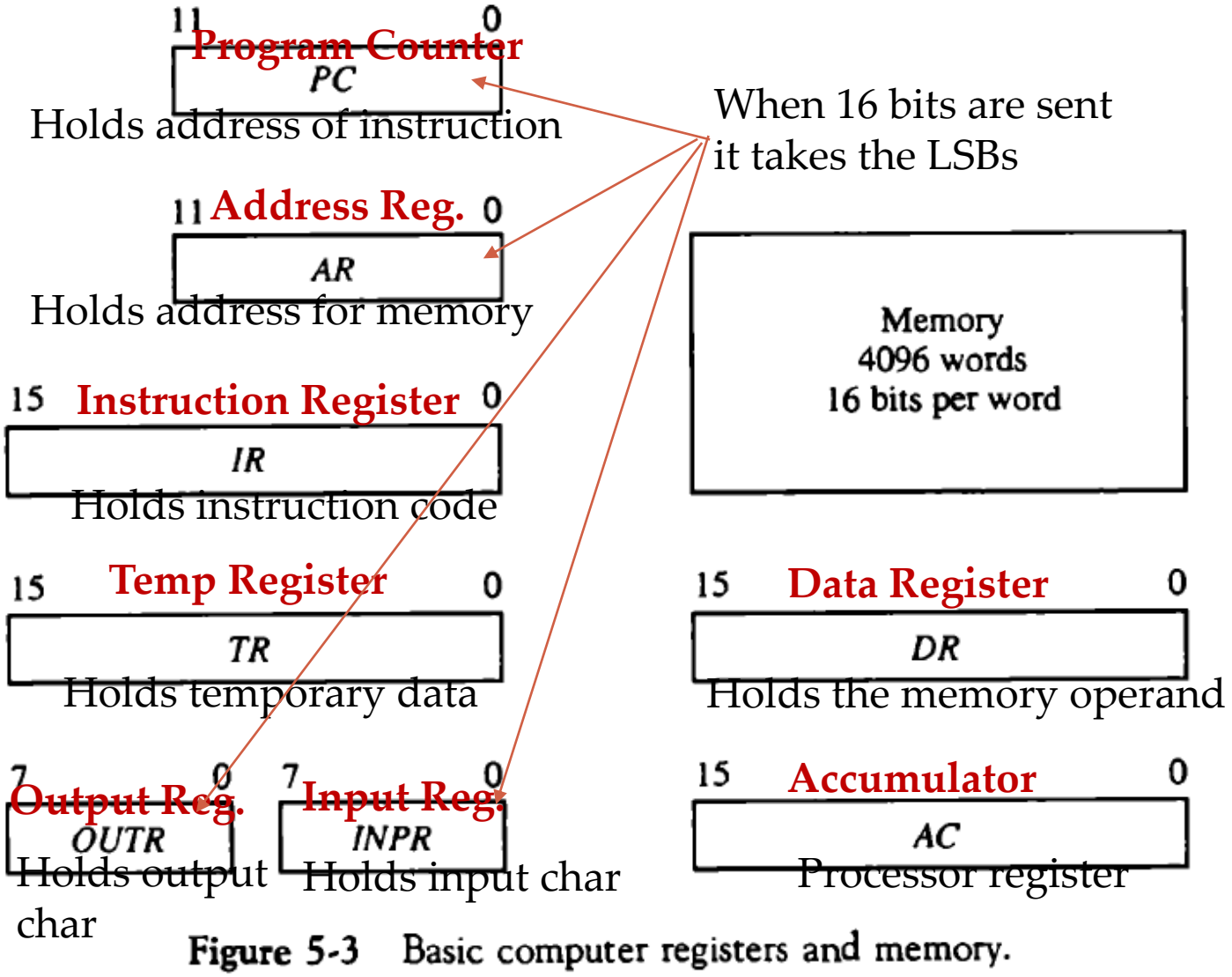


Figure 5-3 Basic computer registers and memory.

Register with INC, LD, CLR

AC
AR
DR
PC
TR

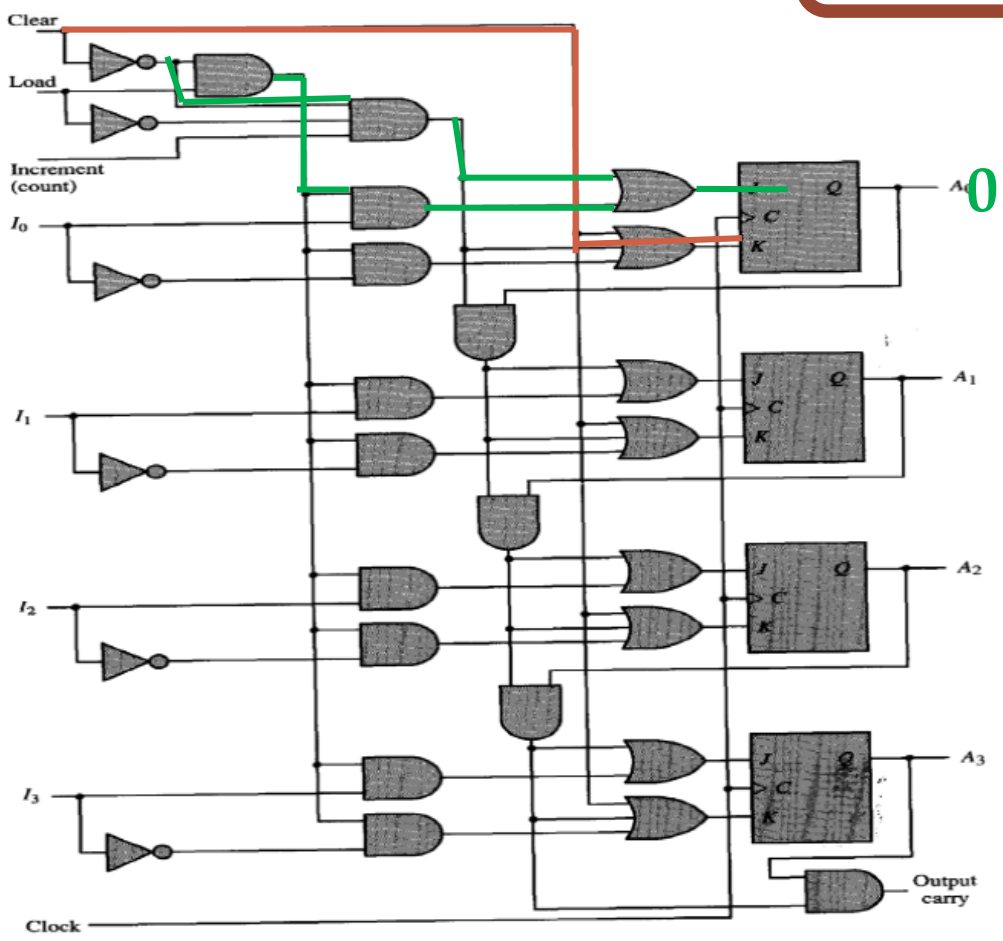


Figure 2-11 4-bit binary counter with parallel load and synchronous clear.

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

Register with INC, LD, CLR

AC
AR
DR
PC
TR

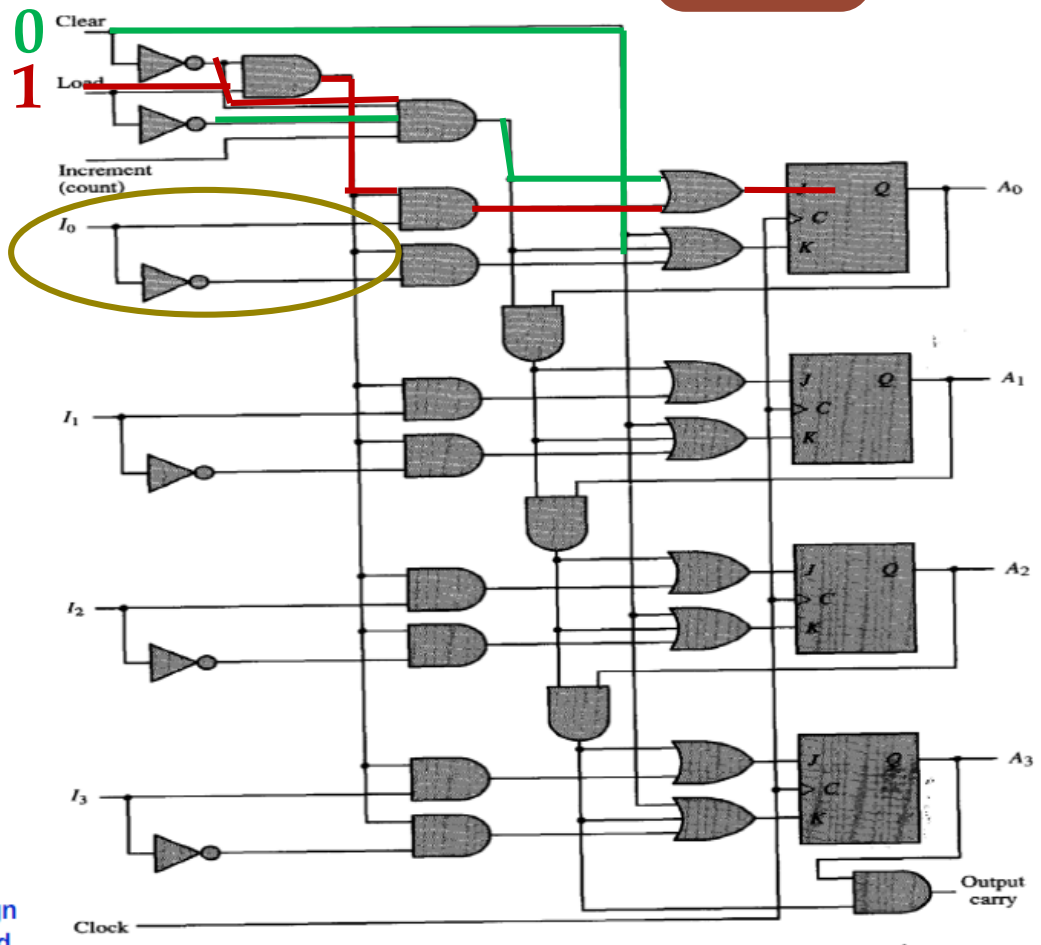


Figure 2-11 4-bit binary counter with parallel load and synchronous clear.

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

Register with INC, LD, CLR

AC
AR
DR
PC
TR

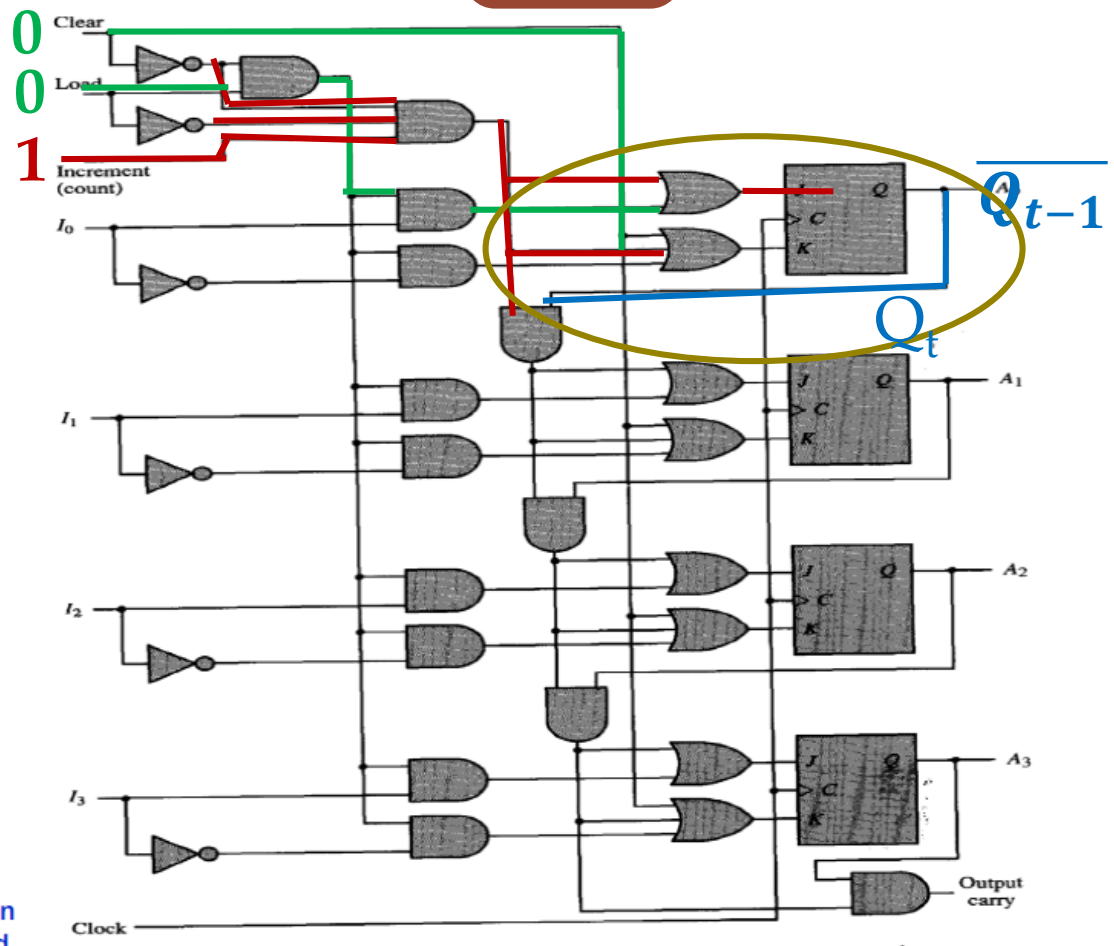
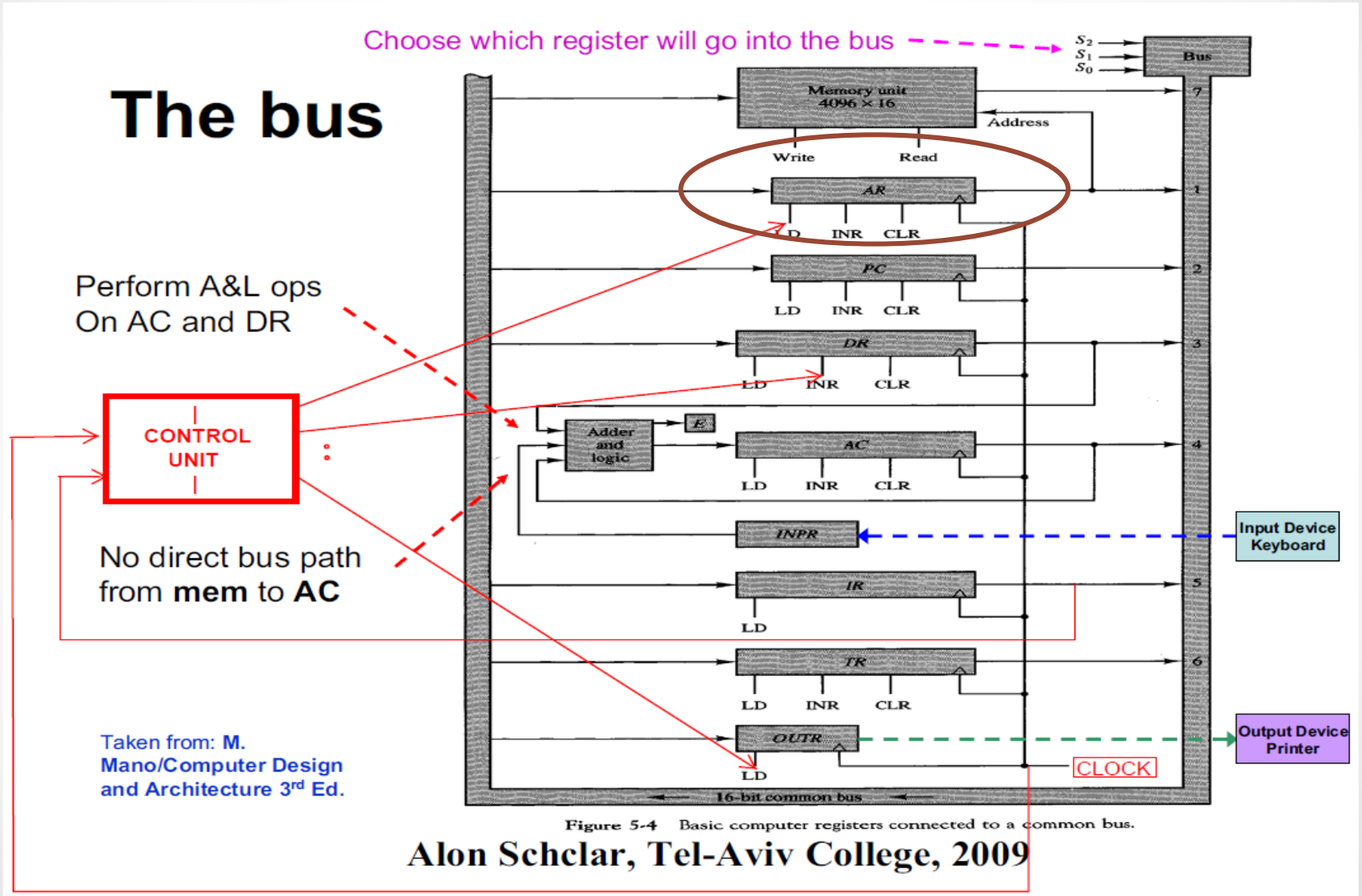


Figure 2-11 4-bit binary counter with parallel load and synchronous clear.

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

The Bus


Choose which register will go into the bus
The bus
Perform A&L ops
On AC and DR
CONTROL UNIT
No direct bus path
from mem to AC
Taken from: M. Mano/Computer Design and Architecture 3rd Ed.
Memory unit 4096 × 16
Address
Write
Read
AR
PC
DR
Adder and logic
E
AC
INPR
IR
TR
OUTR
LD
INR
CLR
Bus
16-bit common bus
Input Device Keyboard
Output Device Printer
CLOCK
Figure 5-4 Basic computer registers connected to a common bus.
Alon Schclar, Tel-Aviv College, 2009

Reminder – register with parallel load

IR
OUTR

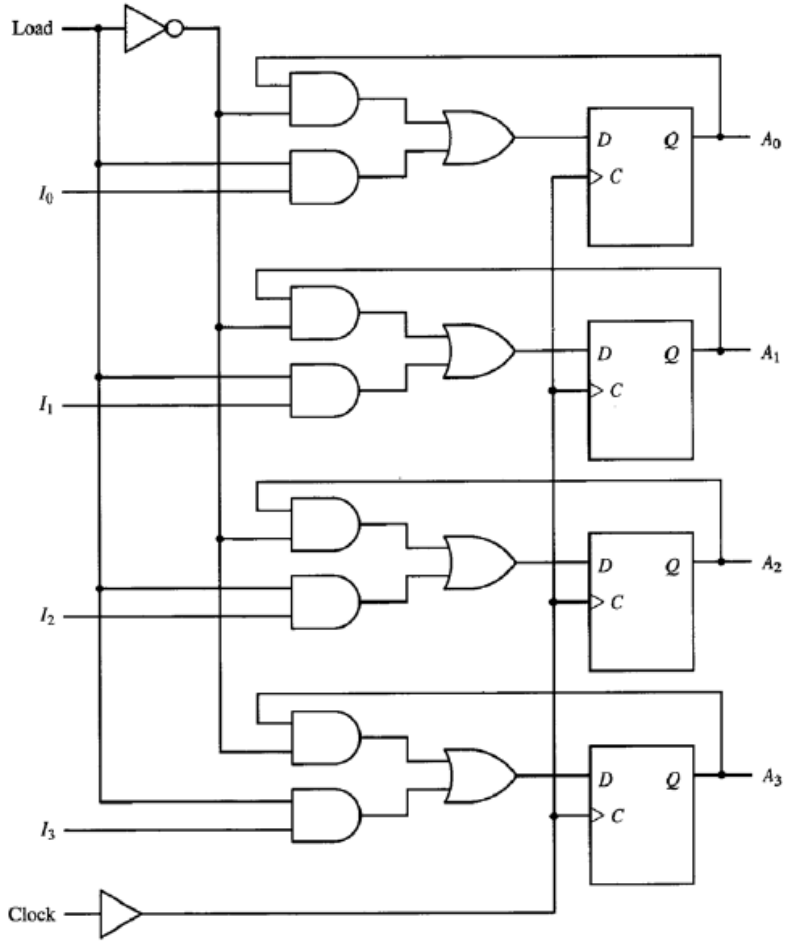
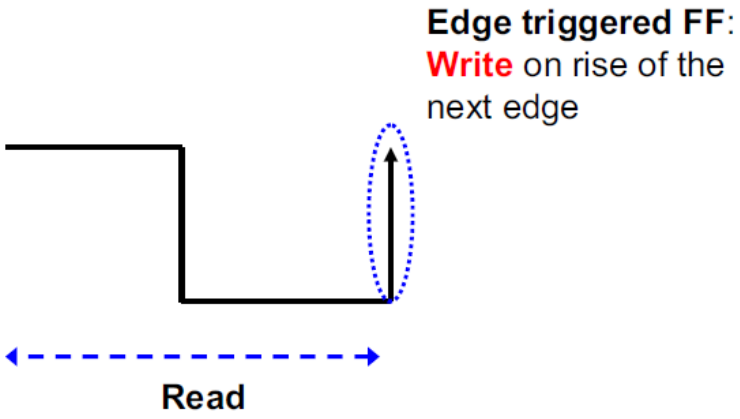


Figure 2-7 4-bit register with parallel load.

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

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

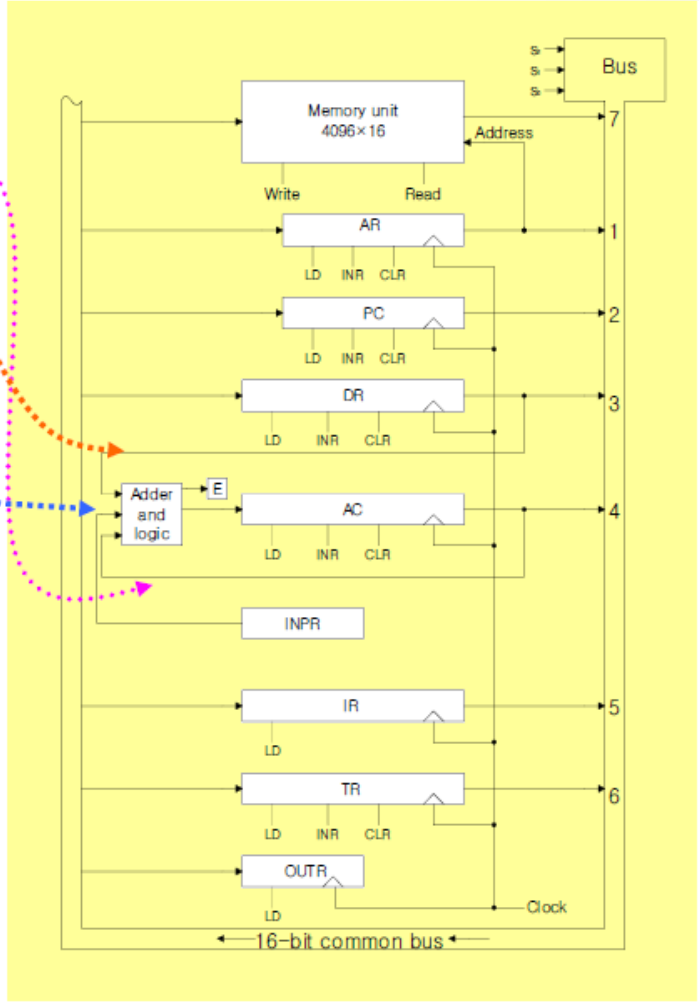
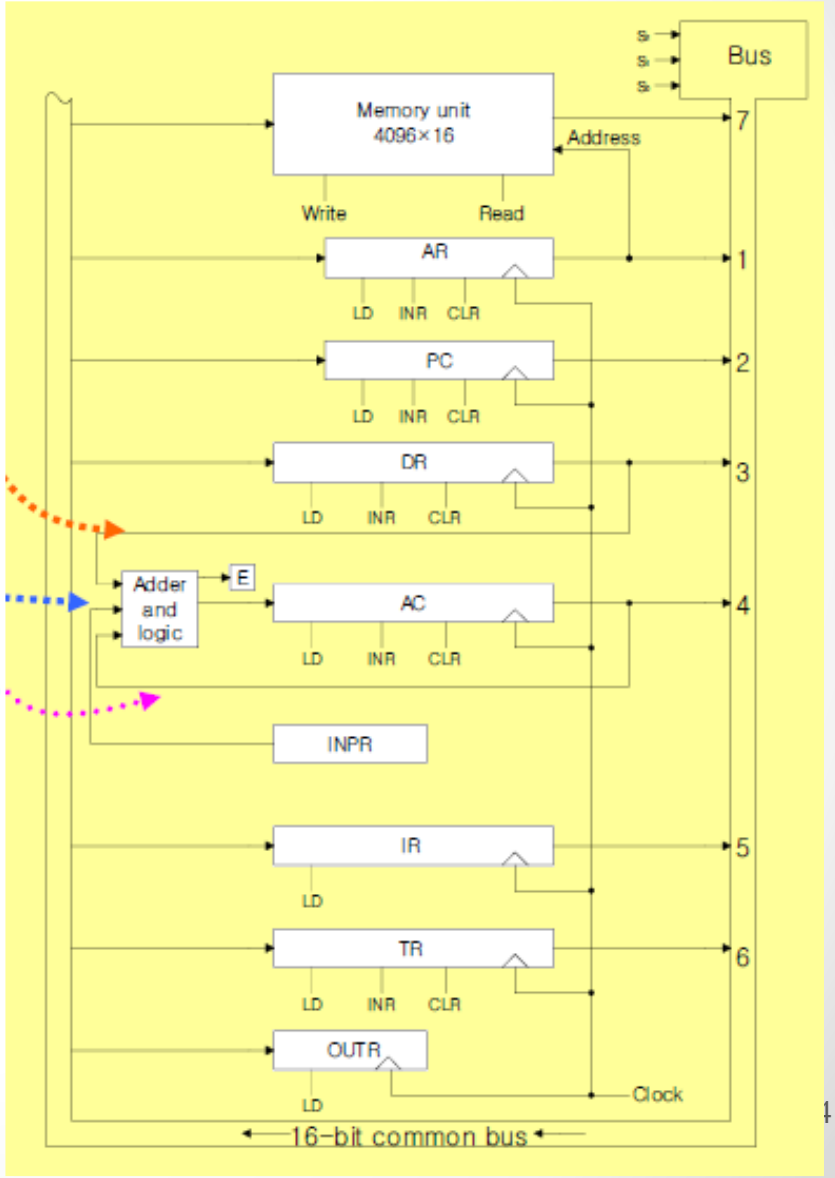


Fig. 5-4 Basic computer registers connected to a common bus

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

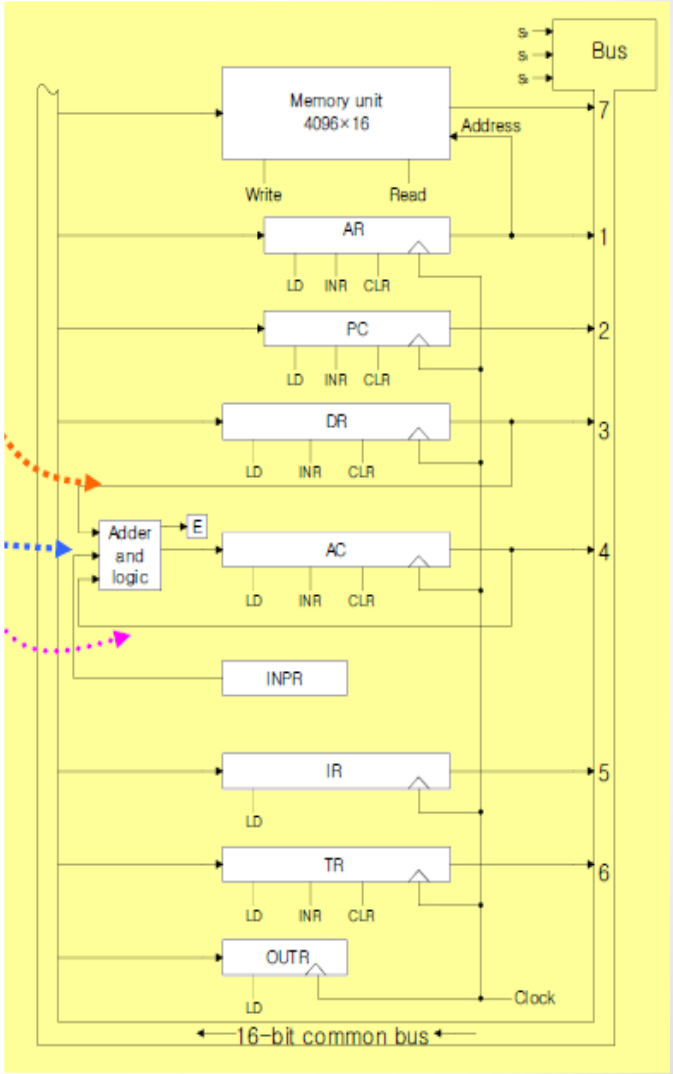
- **DR ←MEM[AR]**
  - AR is connected to address input of MEM
  - Enable READ input of MEMORY
  - Place MEMORY content (MEM[AR]) on BUS
    - $S_2S_1S_0$=111 (MEM num is 7
  - Insert BUS content to DR
    - Enable LD (load) input of DR



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

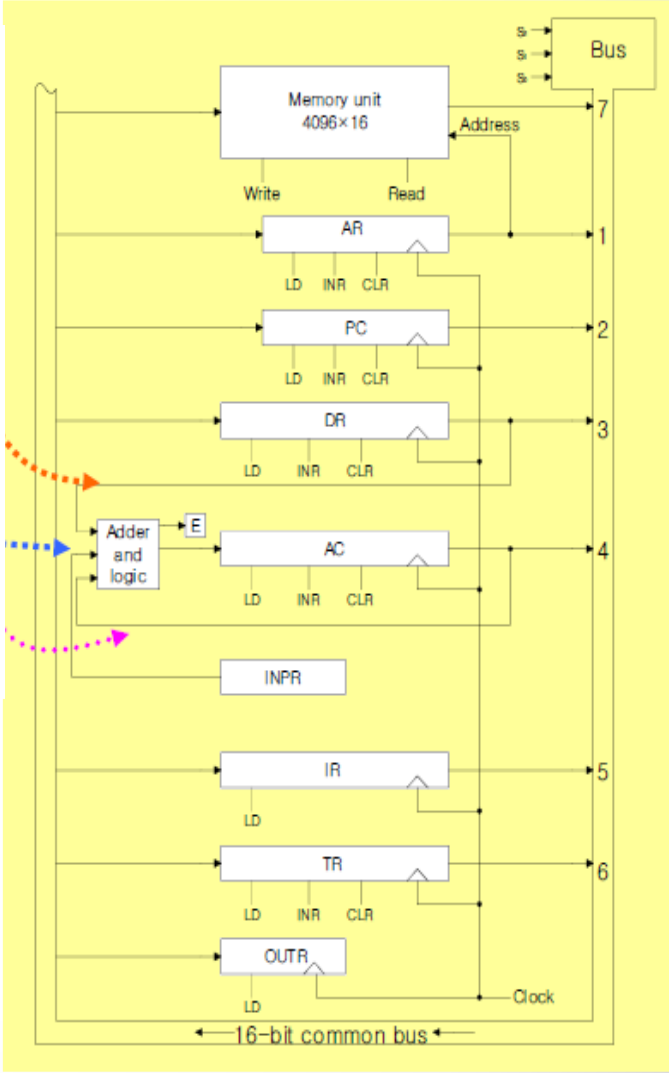
- The clock transition at the end of the cycle
  - transfers the content of the bus into the target register and
  - the output of the adder and logic circuit into AC.



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

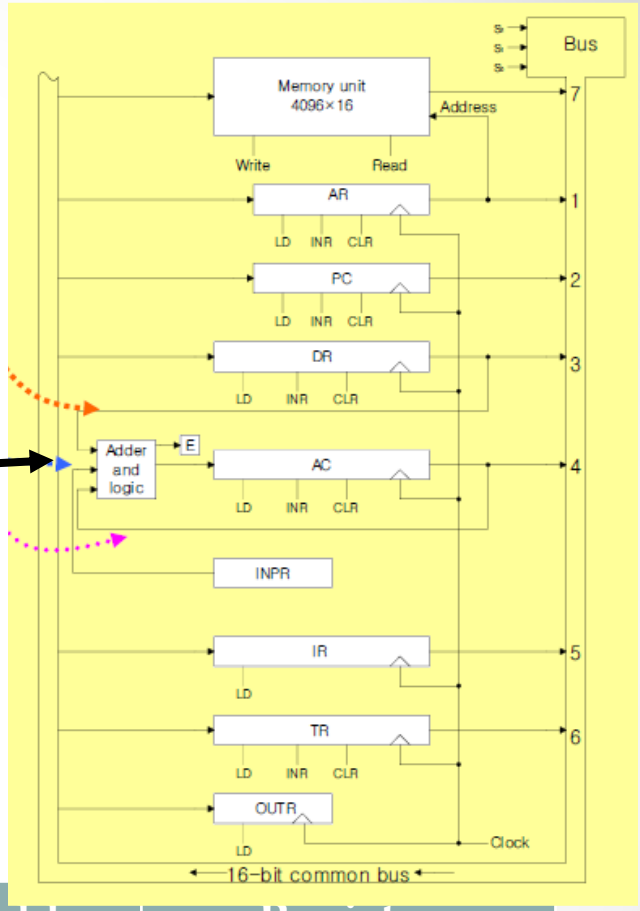
- **Can be executed in the same clock cycle**



QUIZ2

The following control inputs are active in the bus system shown in [The Bus slide](#) or here

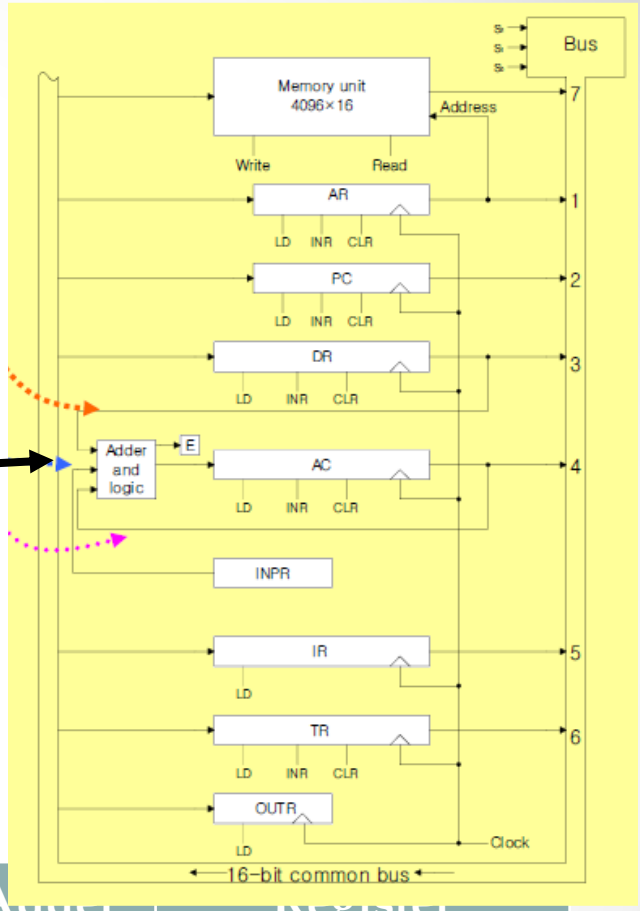
For each case specify the register transfer that will be executed during the next clock transition



S_2	S_1	S_0	LD of register	Memory oper.	Adder oper.	Register Transfer
1	1	1	IR	Read	-	
1	1	0	PC	-	-	
1	0	0	DR	Write	-	
0	0	0	AC	-	Add	

QUIZ3

The following register transfer are active in the bus system shown in The Bus slide or here
For each case fill the table below

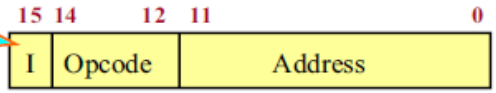


S_2	S_1	S_0	LD of register	Memory oper.	Adder oper.	Register Transfer
						AR ← PC
						IR ← M[AR]
						M[AR] ← TR
						AC←DR, DR←AC

Computer Instruction

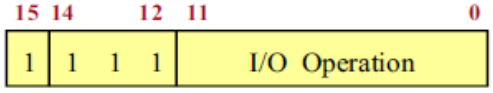
- 5-3 Computer Instruction
  - 3 Instruction Code Formats : ***Fig. 5-5***
    - Memory-reference instruction
      - » Opcode = 000 ~ 110
        - I=0 : 0xxx ~ 6xxx, I=1: 8xxx ~Exxx

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