Table of Contents

[How you define a ‘Simple digital system’? 1](#_Toc100426492)

[What is register? 1](#_Toc100426493)

[Structure of a CPU 1](#_Toc100426494)

[What do you mean by ‘organization of a digital system’? 2](#_Toc100426495)

[How to refer a register? 2](#_Toc100426496)

[What is register Transfer? 3](#_Toc100426497)

[Control Function 4](#_Toc100426498)

[What is register Transfer Language? 4](#_Toc100426499)

[What do you mean by register transfer level? 5](#_Toc100426500)

[What is bus? 5](#_Toc100426501)

[How register transfer is done by bus? 6](#_Toc100426502)

[How to represent memory? 9](#_Toc100426503)

[Memory read 10](#_Toc100426504)

[Memory writes 11](#_Toc100426505)

[Register Microoperation 11](#_Toc100426506)

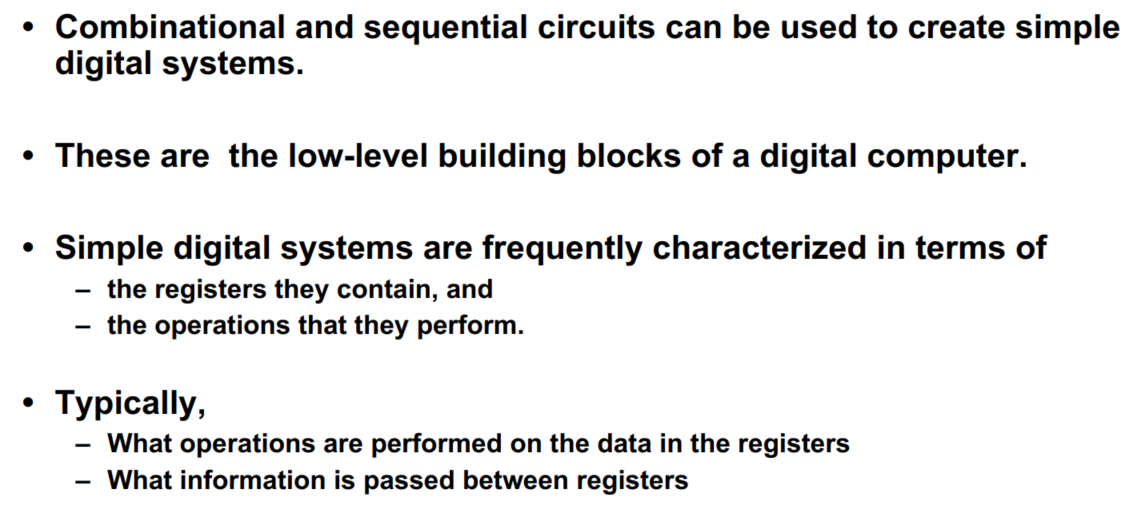
[Arithmetic Microoperations 11](#_Toc100426507)

[Logical Microoperations 11](#_Toc100426508)

[Shift Microoperations 11](#_Toc100426509)

[Arithmetic Logic Shift Unit 11](#_Toc100426510)

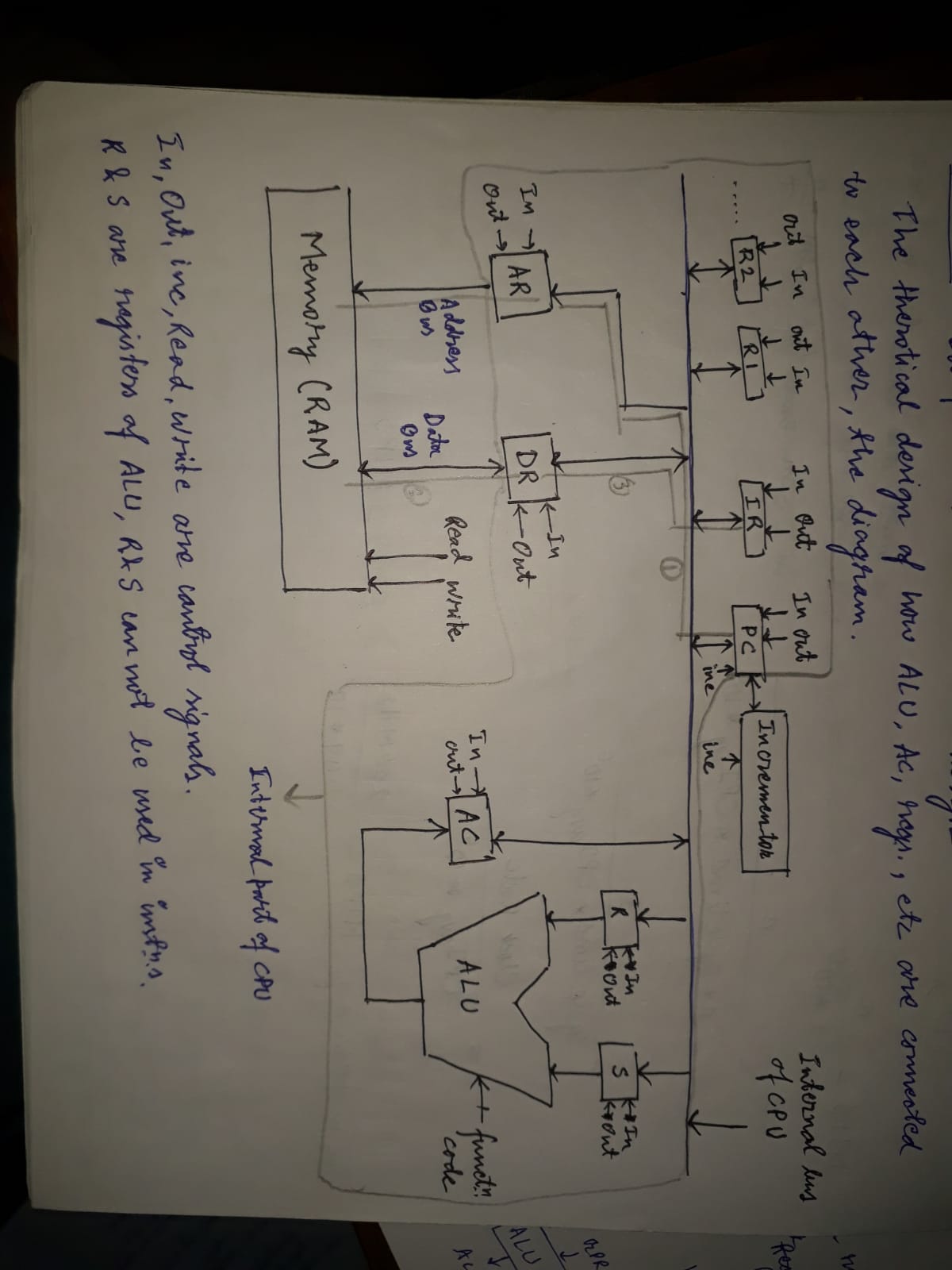
# How you define a ‘Simple digital system’?



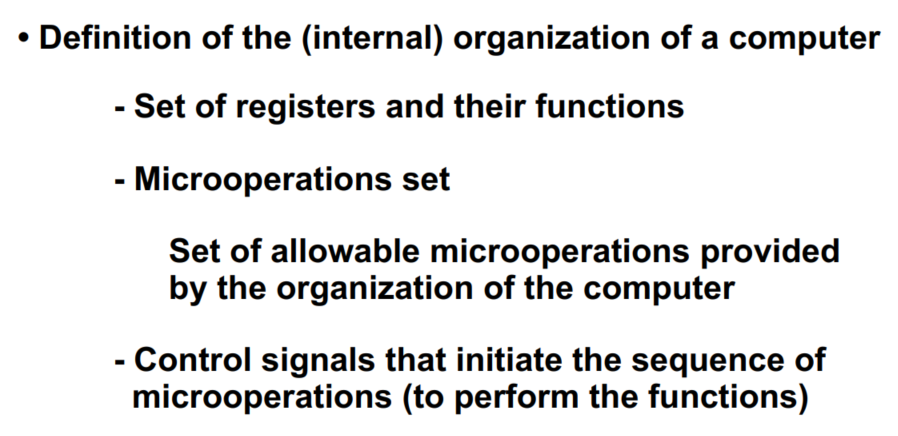
# What is register?

Registers are small memory units present in CPU.

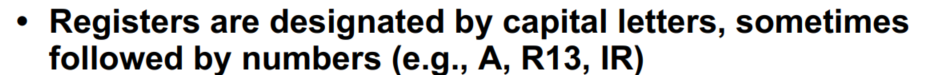
# Structure of a CPU

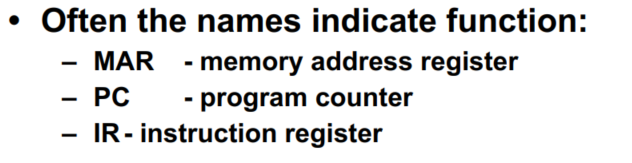


# What do you mean by ‘organization of a digital system’?

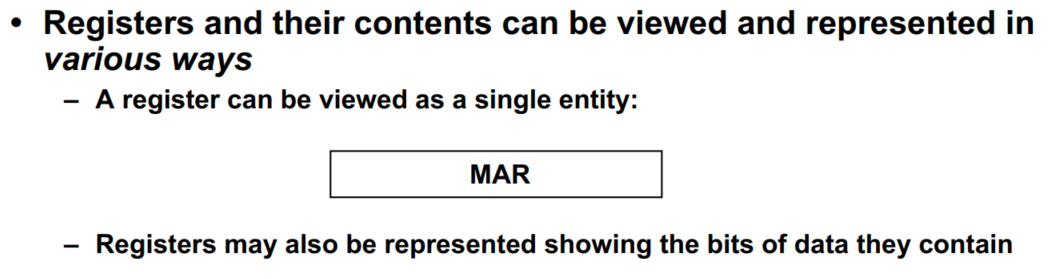


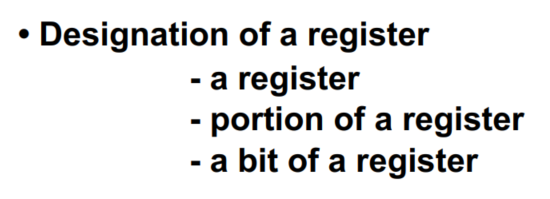
# How to refer a register?

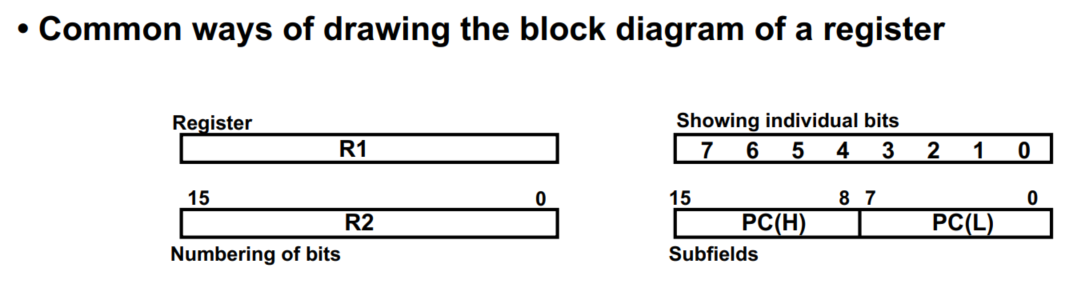




Program Counter stores till which instruction a program has executed.



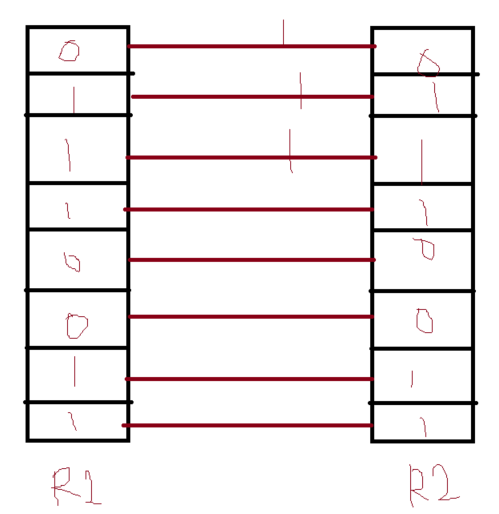
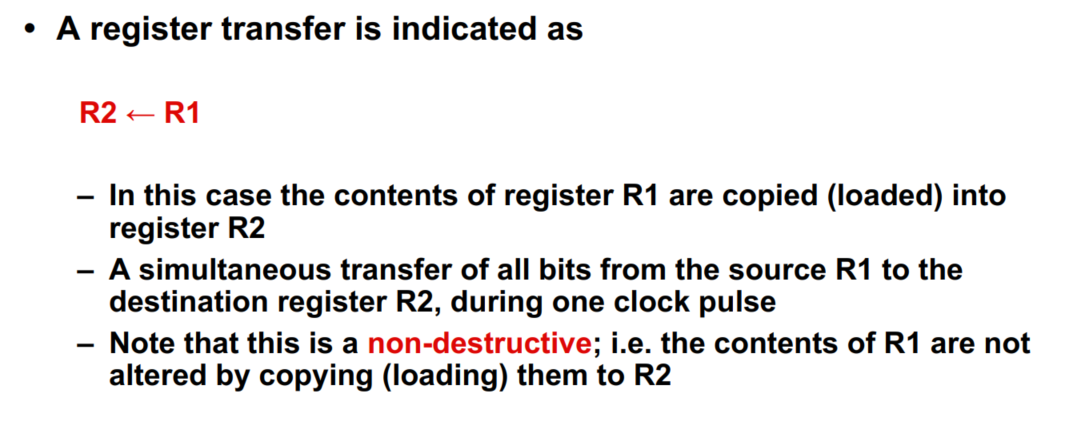
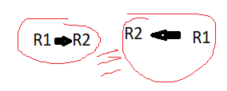




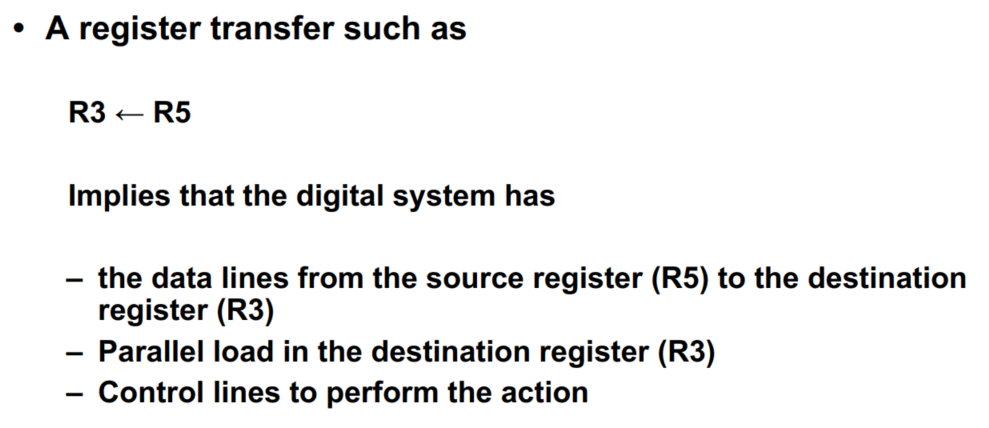
# What is register Transfer?



Example of register transfer

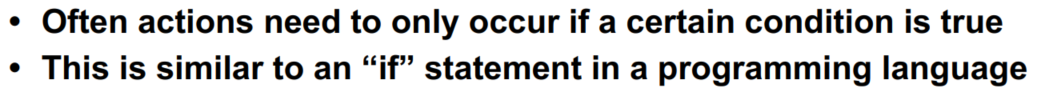


Another example of register transfer:





# Control Function

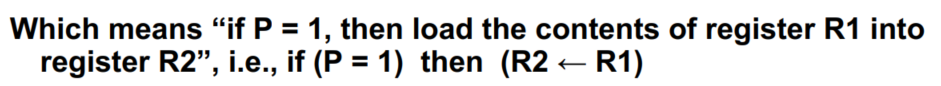


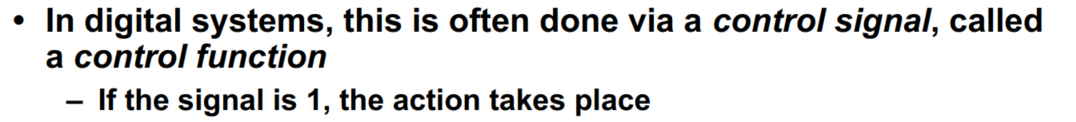
if(p==1) {

R2🡸R1

}

This is same as **P: R2🡸R1**

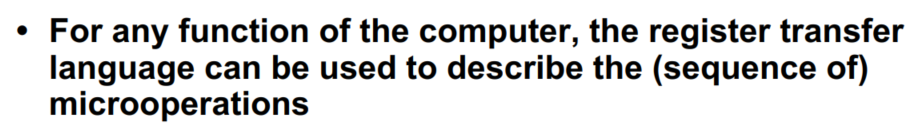


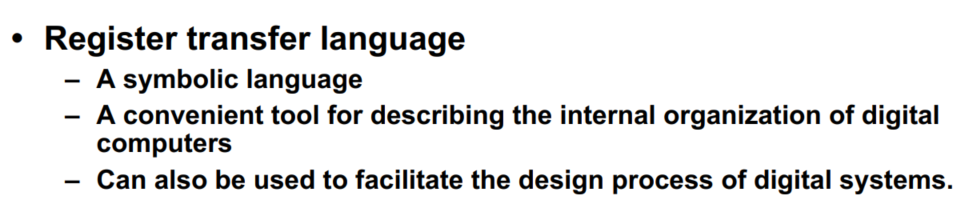


# What is register Transfer Language?

|  |  |
| --- | --- |
| A🡸B | Transfer content of register B into register A |
| AR🡸DR(AD) | Transfer content of AD portion of register DR into AR |
| A🡸Constant, eg: A🡸#5 | Transfer a binary constant into register A |
| ABUS🡸R1, R2🡸ABUS | Transfer content of R1 into **bus A**, and at the same time transfer content of **bus A** into R2 |
| AR | AR is Address Register |
| DR | DR is Data Register |
| M[R] | Memory word specified by register R |
| DR🡸M | Memory read operation: transfer content of memory word specified by AR into DR |
| M🡸DR | Memory write operation: transfers content of DR into memory word specified by AR |





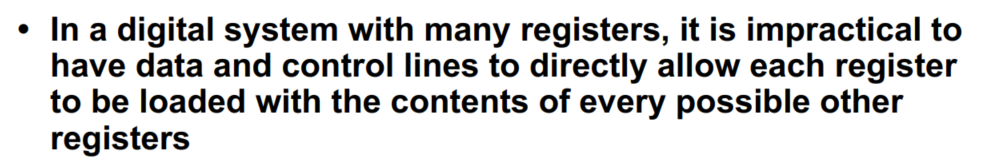


# What do you mean by register transfer level?



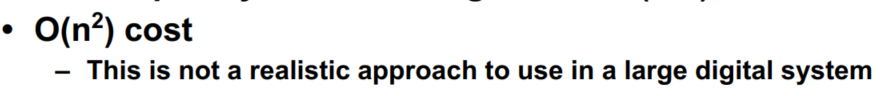


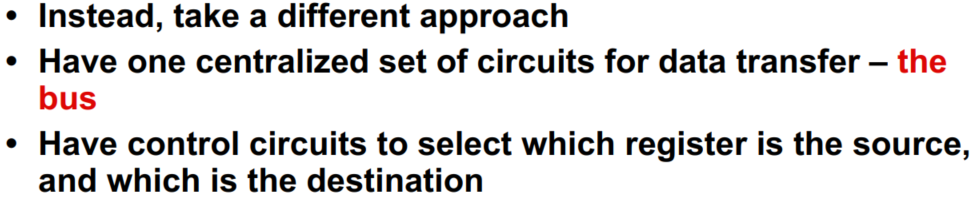
# What is bus?

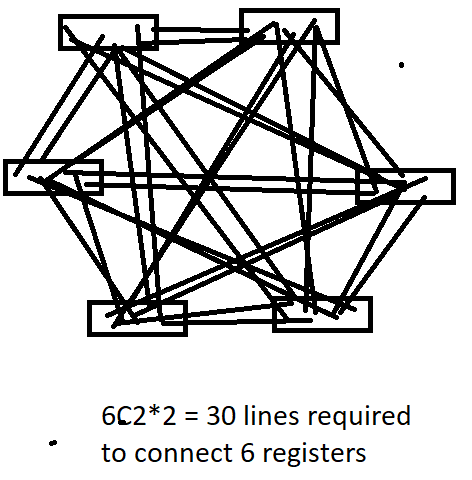


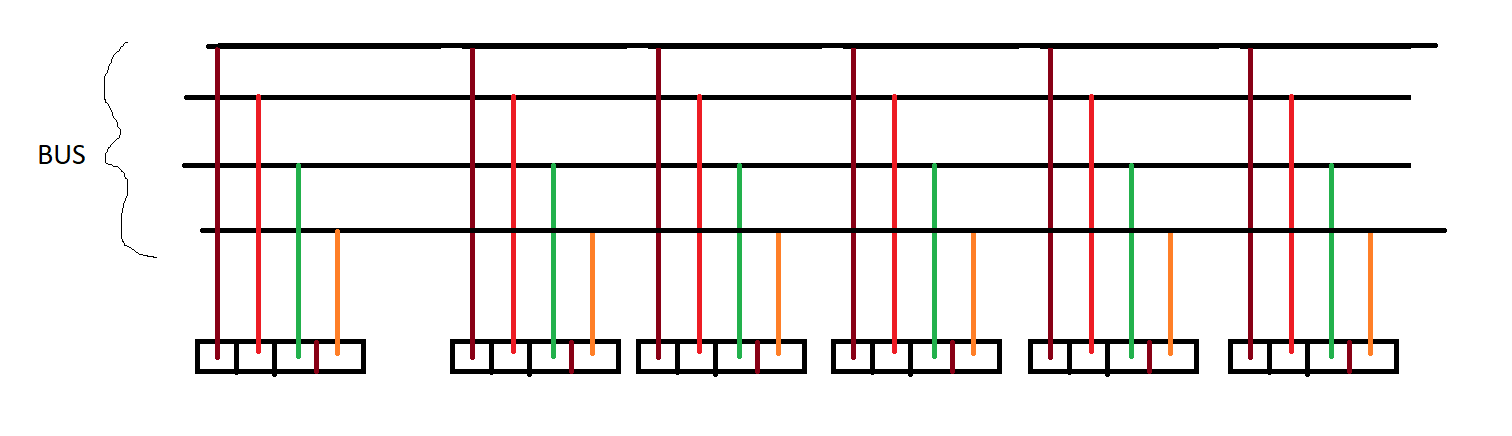


To connect n registers we need nC2 \* 2 = n(n-1) lines

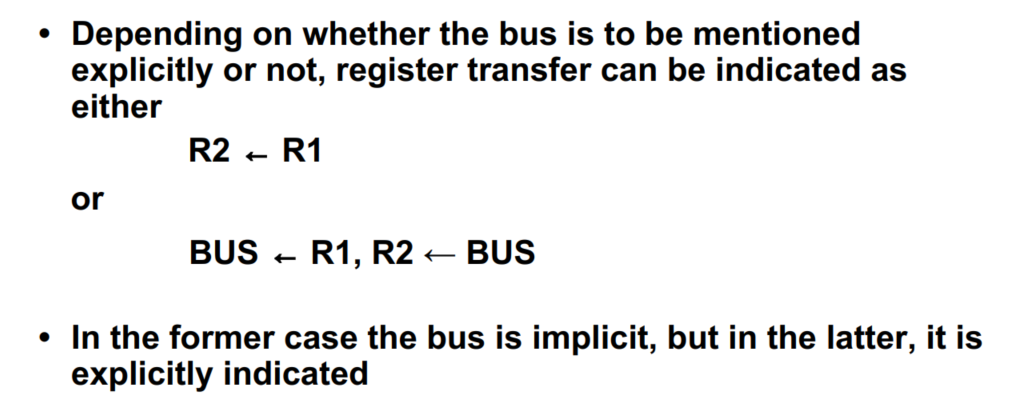


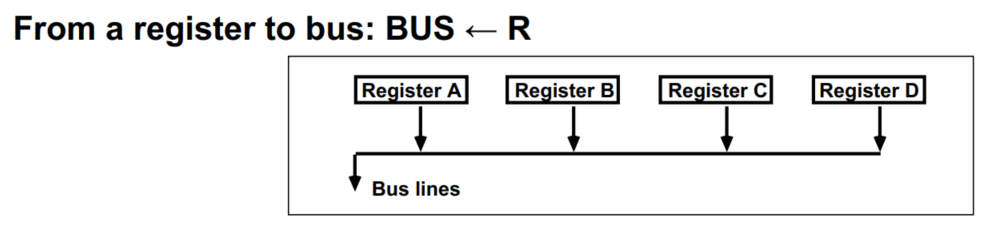


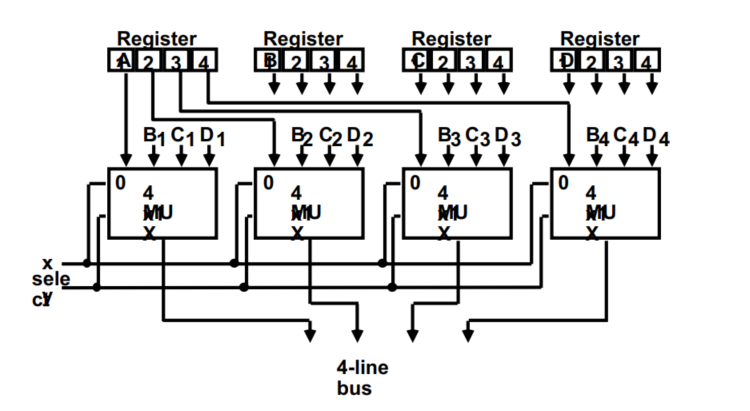




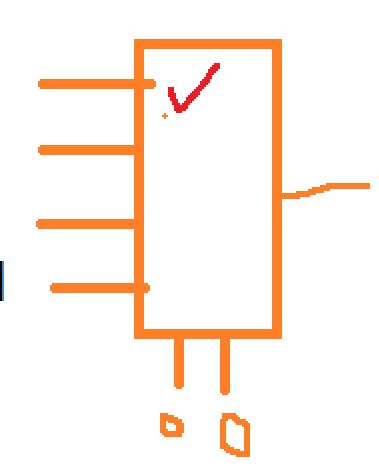
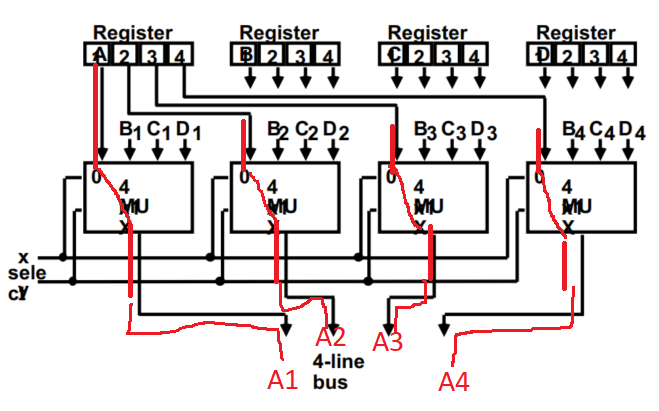
# How register transfer is done by bus?



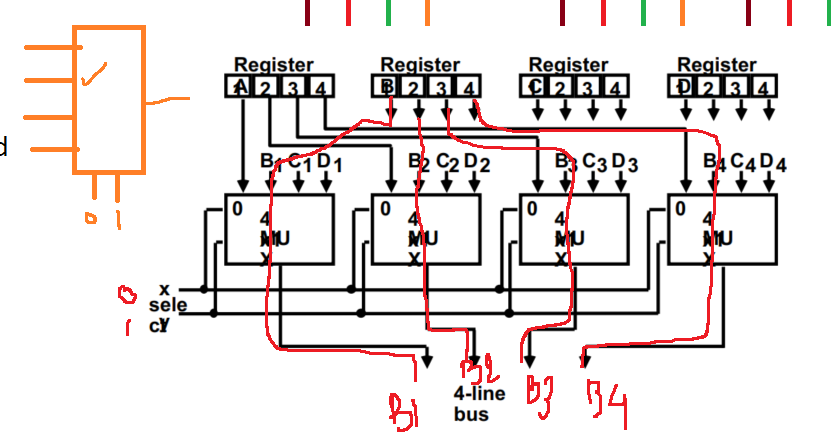




If (x,y)==(0,0) then A1, A2, A3, A4 will come out:



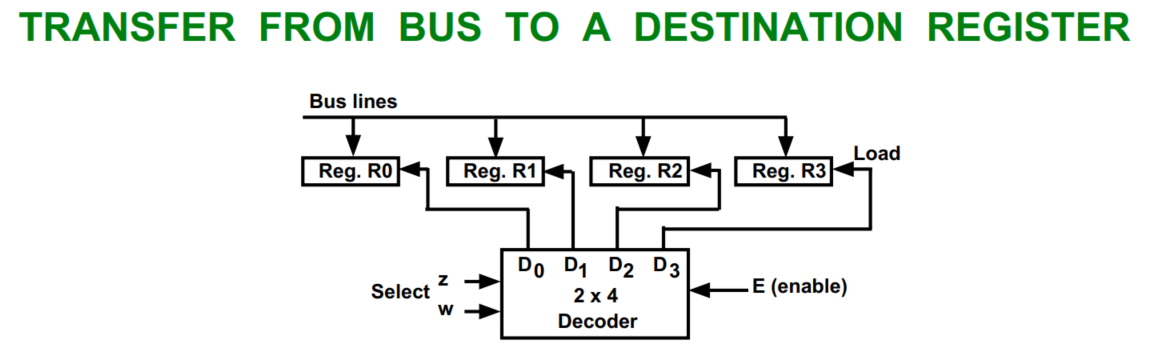
If (x,y)==(0,1) then B1, B2, B3, B4 will come out:

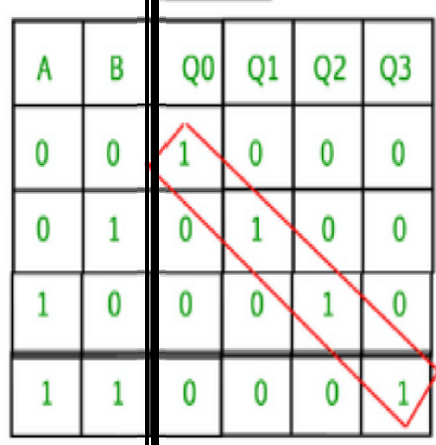
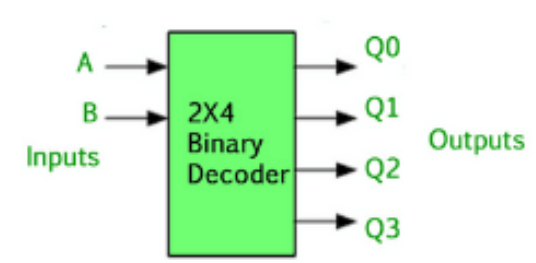


Draw the diagrams for C1, C2, C3, C4 to come out.

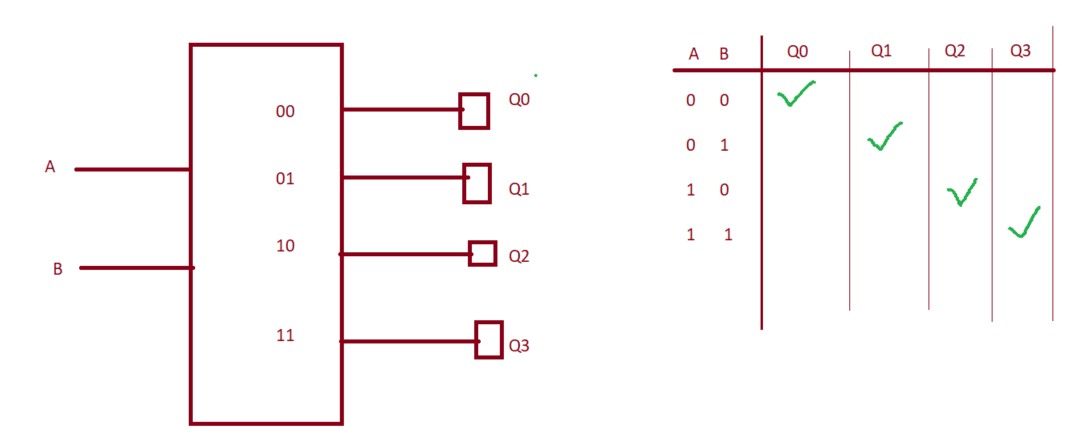
Draw the diagrams for D1, D2, D3, D4 to come out.

We have just seen Register to bus (Register🡺Bus) transfer. Now we’ll see Bus🡺Register Transfer:

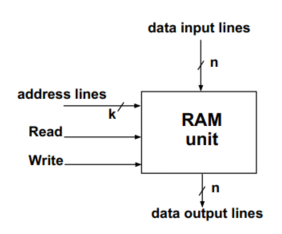


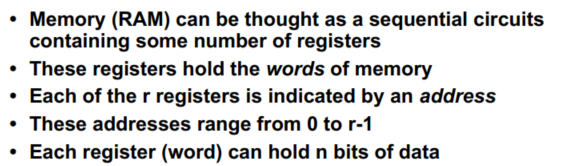


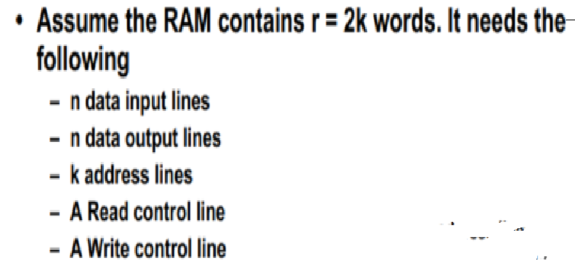




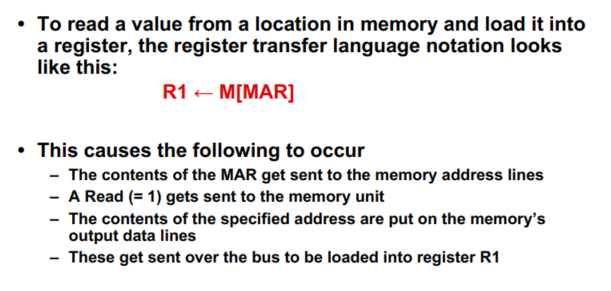
# How to represent memory?

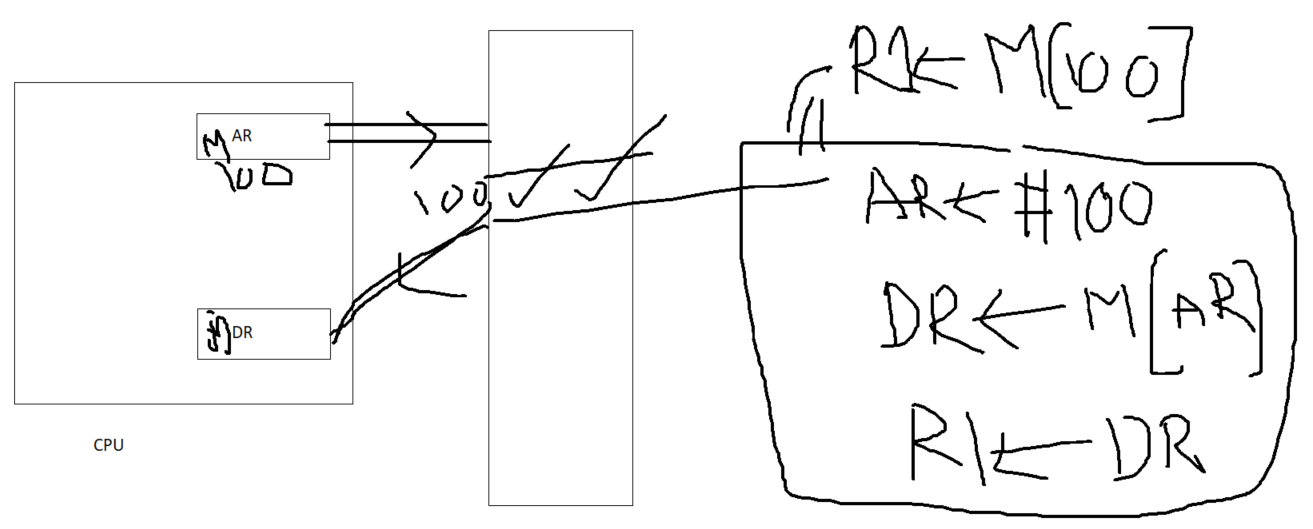




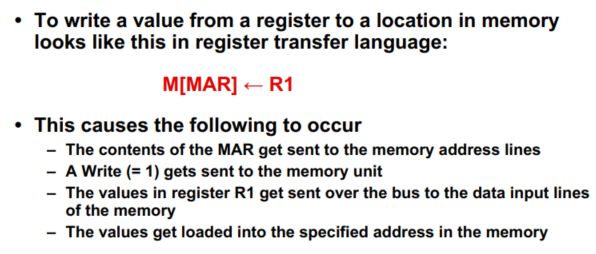


# Memory read



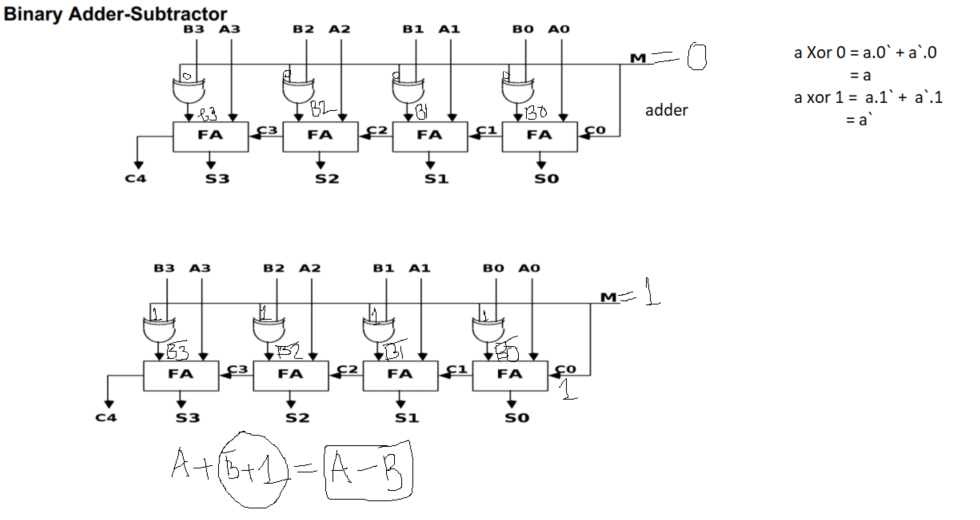


# Memory writes

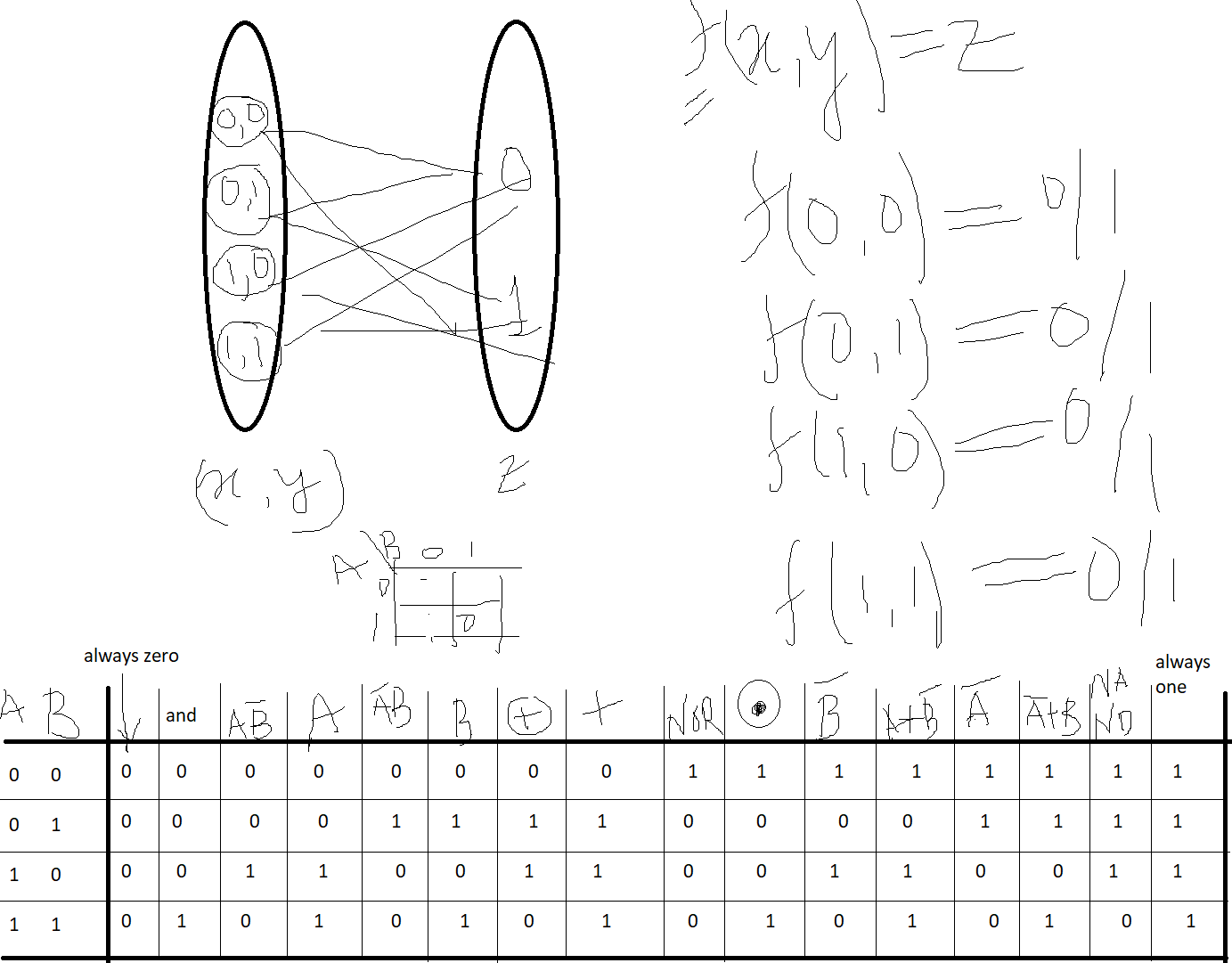


# Register Microoperation

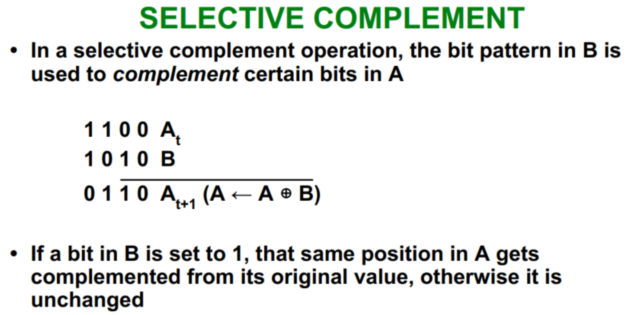
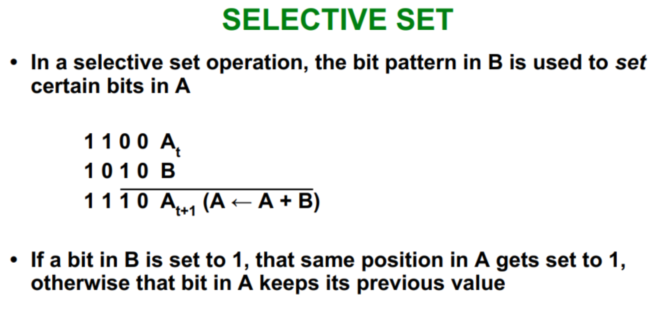
# Arithmetic Microoperations



# Logical Microoperations



# Application of Logic Microoperations



# Shift Microoperations

# Arithmetic Logic Shift Unit

