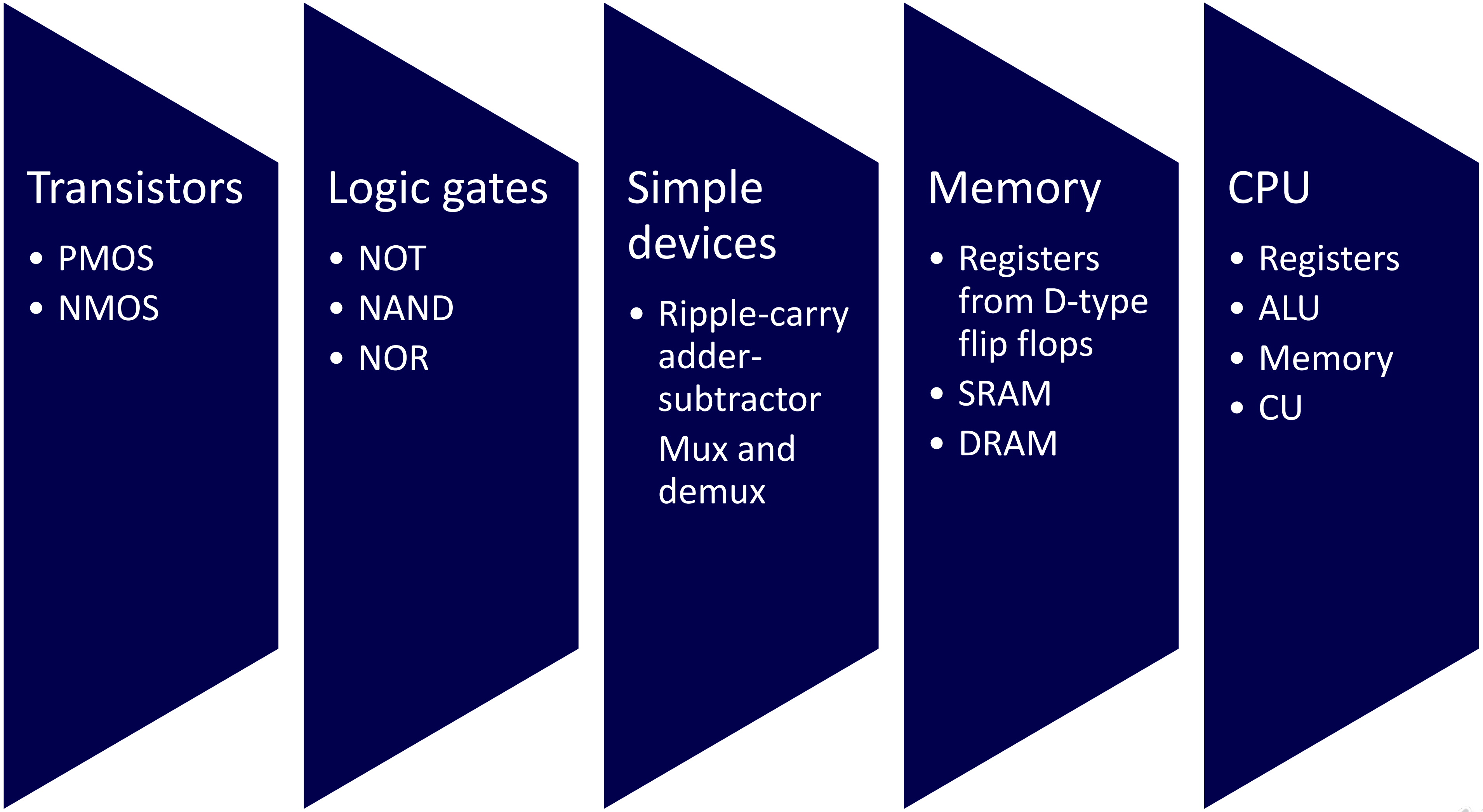
COMSM1302
Overview of Computer Architecture

Lecture 8
Simple CPU

The next step.



Transistors

- PMOS
- NMOS

Logic gates

- NOT
- NAND
- NOR

Simple devices

- Ripple-carry adder-subtractor

  Mux and demux

Memory

- Registers from D-type flip flops
- SRAM
- DRAM

CPU

- Registers
- ALU
- Memory
- CU

What we will learn in Part 2

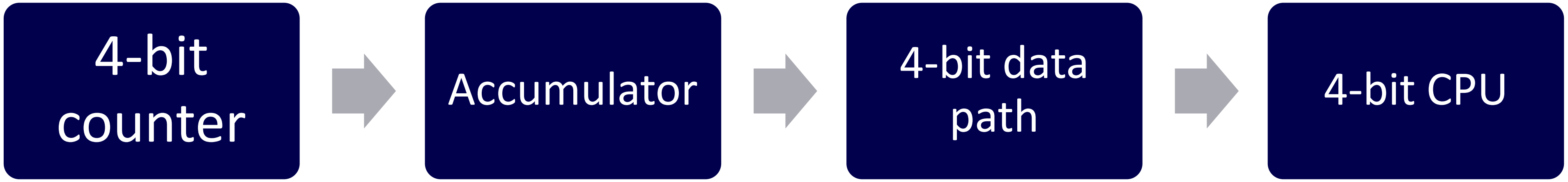
1. Week 5 and 7: Central Processing Unit (CPU) architecture

2. Week 8 and 9: ARM instruction set architecture and assembly programming.

3. Week 10: Assemblers and Compilers.

In this lecture

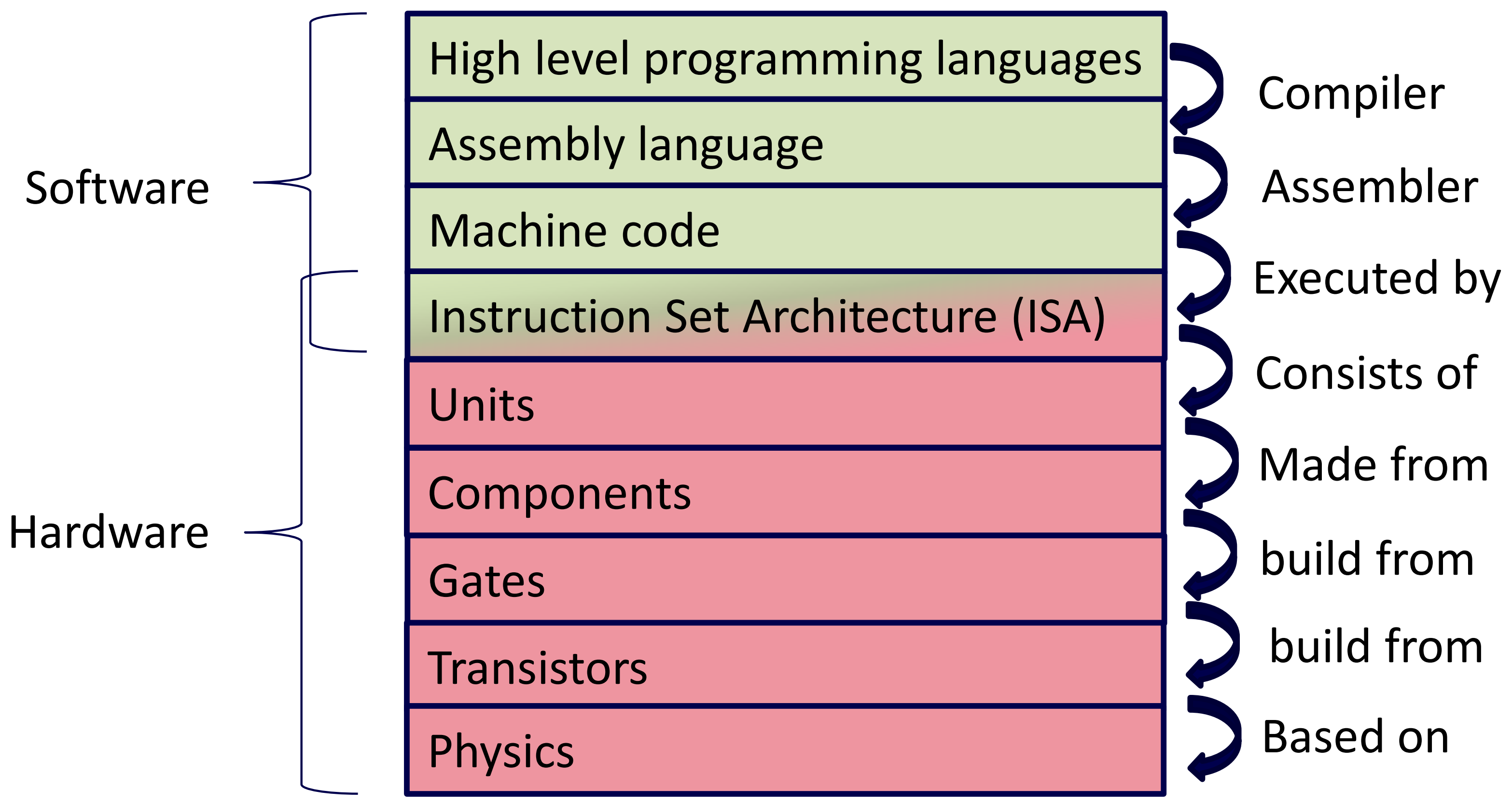
1. Computer layers from hardware to software.
2. From 4-bit counter to 4-bit CPU



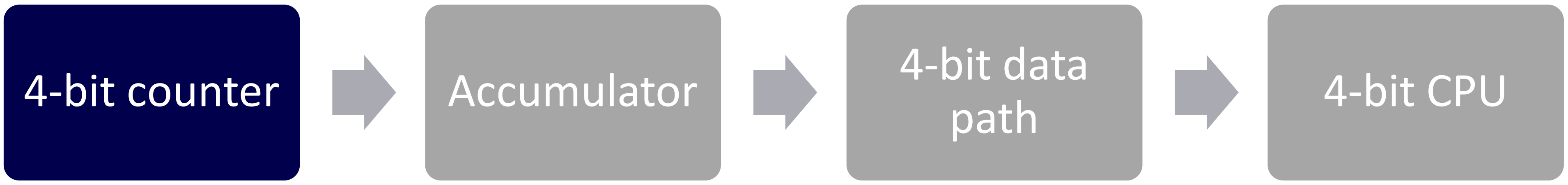
4-bit counter → Accumulator → 4-bit data path → 4-bit CPU

- Review memory, data sizes and clock
- Small assembly program

Layers

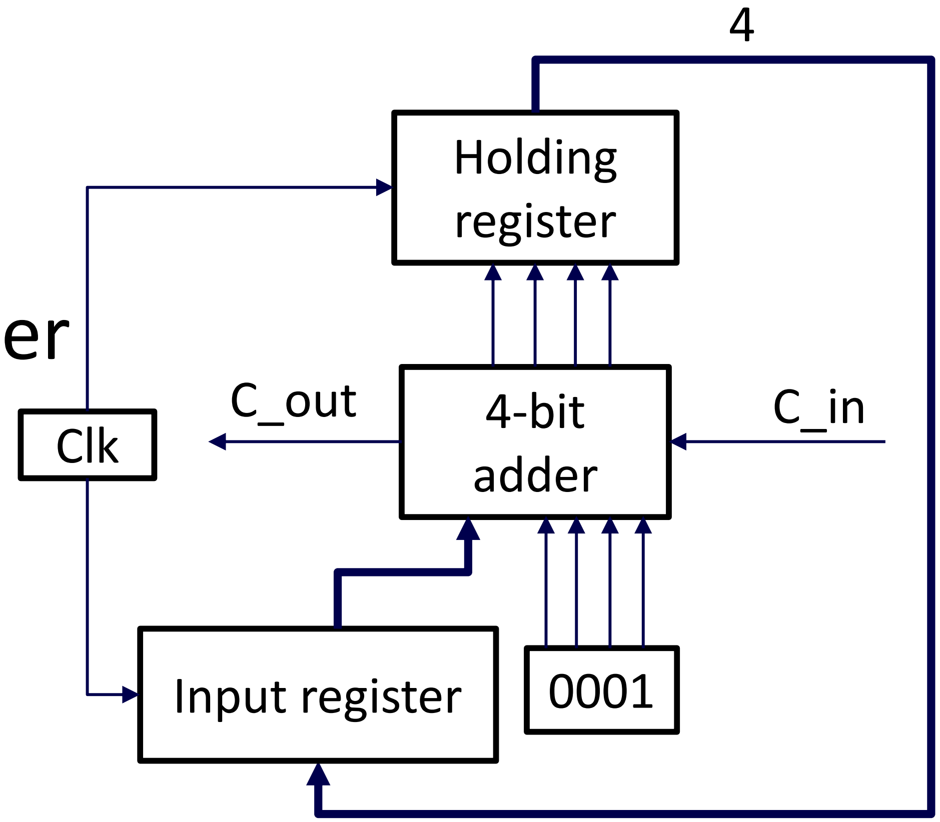

Software
Hardware
High level programming languages
Assembly language
Machine code
Instruction Set Architecture (ISA)
Units
Components
Gates
Transistors
Physics
Compiler
Assembler
Executed by
Consists of
Made from
build from
build from
Based on

From 4-bit counter to 4-bit CPU - Counter



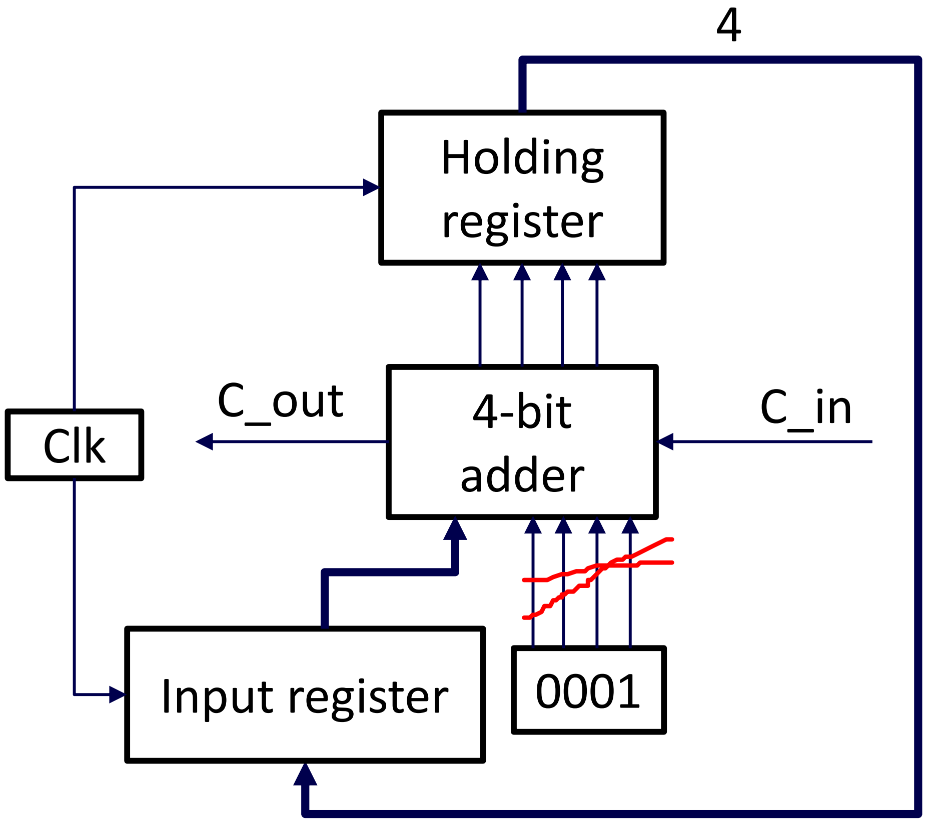
4-bit counter

- Design a 4-bit counter:
  - 4-bit adder
  - Constant one
  - High level triggered register
    - Holding register
    - Input register
  - Clock

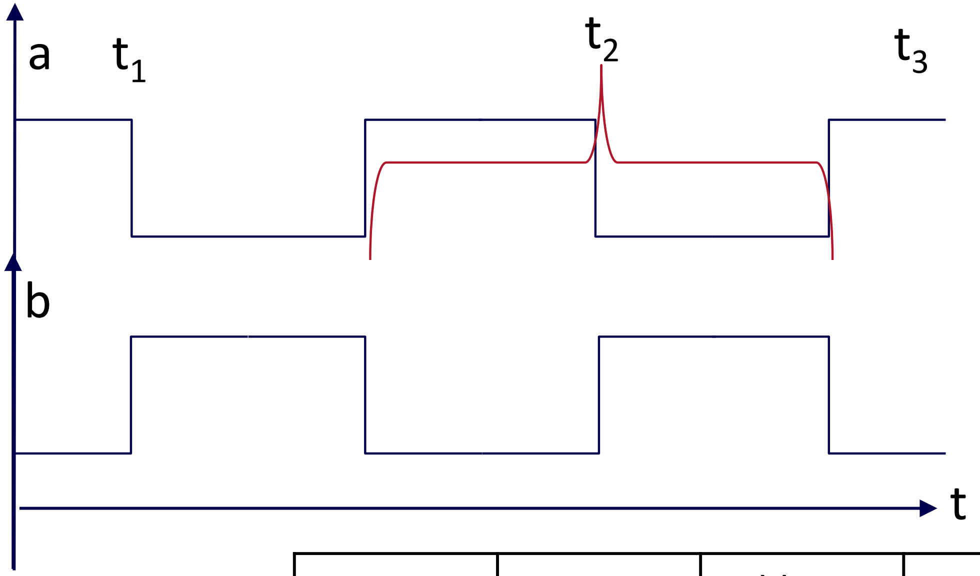


4-bit counter - questions

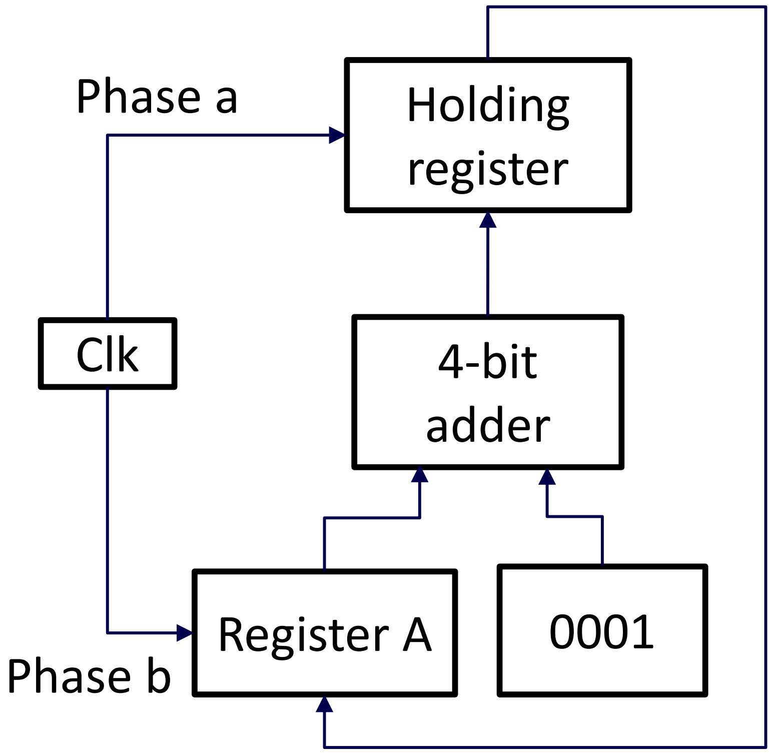
1. What is the range of this counter with and without considering the carry out bit?
2. Can you leave the second input disconnected and still achieve the same functionality? What should we use instead?



4-bit counter – (1/5)

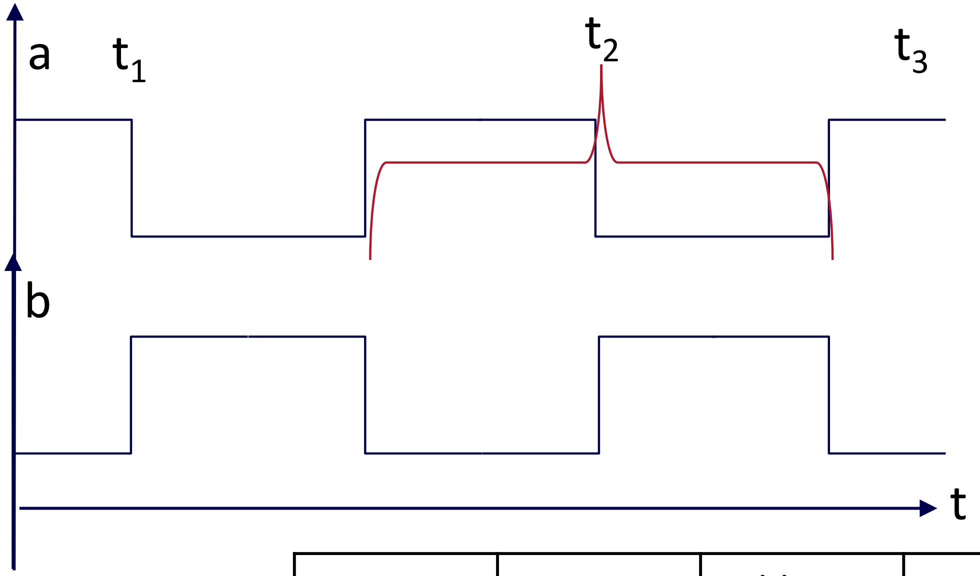


CLK	Reg A	Adder	H Reg
1a	0000	0001	0001
1b			
2a			
2b			
3a			
3b			

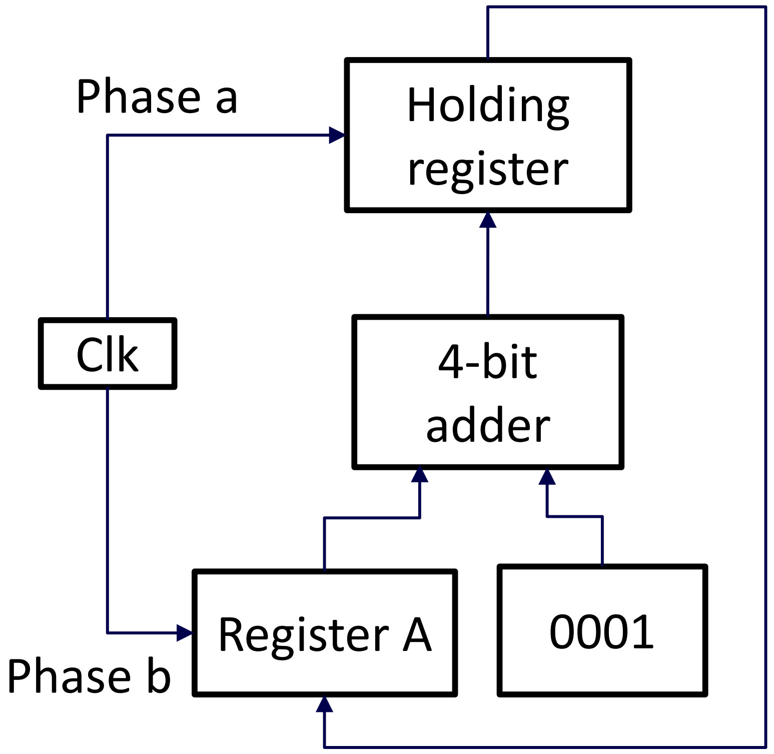


Assume A initial value is 0000

4-bit counter – (2/5)

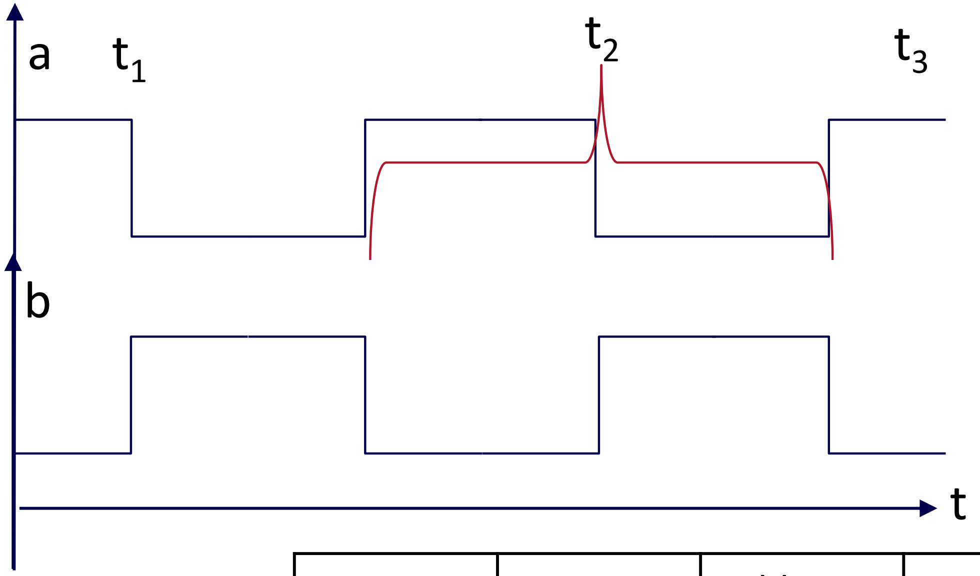


CLK	Reg A	Adder	H Reg
1a	0000	0001	0001
1b	0001	0010	0001
2a			
2b			
3a			
3b			

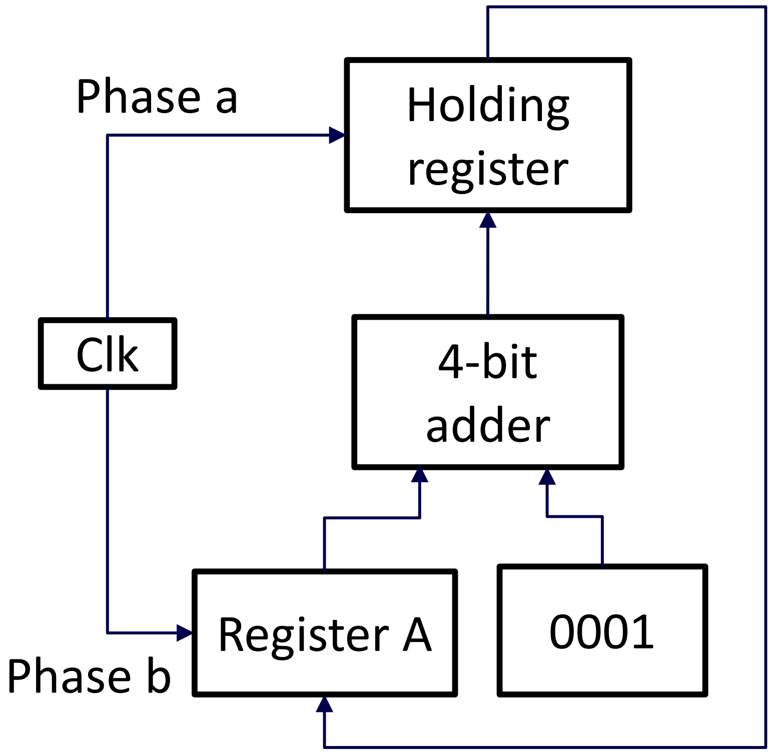


Assume A initial value is 0000

4-bit counter – (3/5)

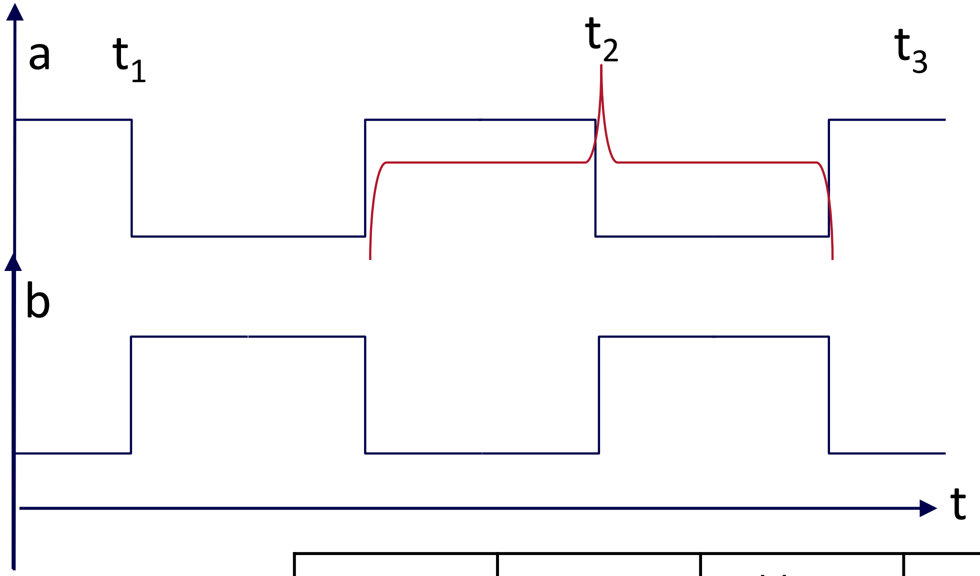


CLK	Reg A	Adder	H Reg
1a	0000	0001	0001
1b	0001	0010	0001
2a	0001	0010	0010
2b			
3a			
3b			

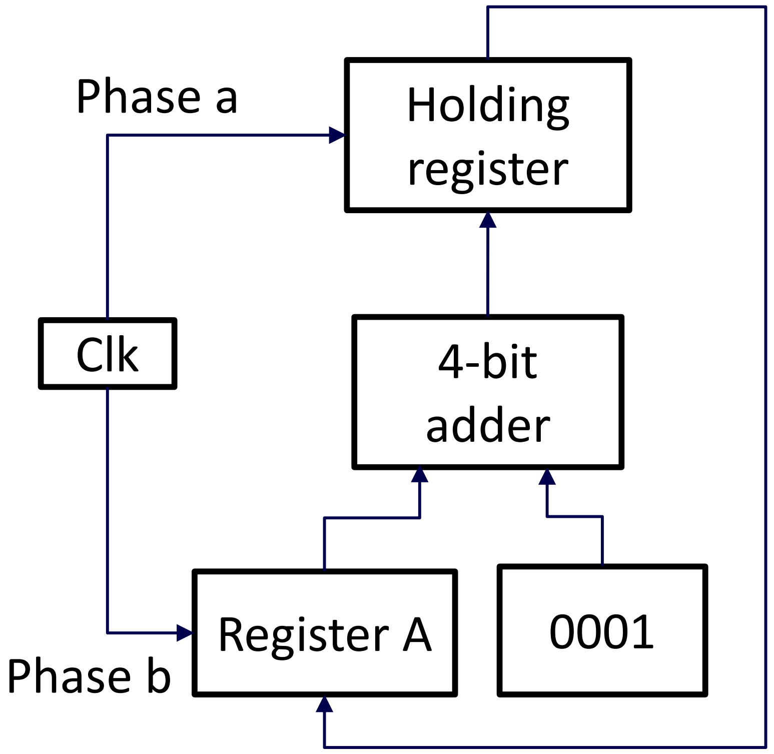


Assume A initial value is 0000

4-bit counter – (4/5)

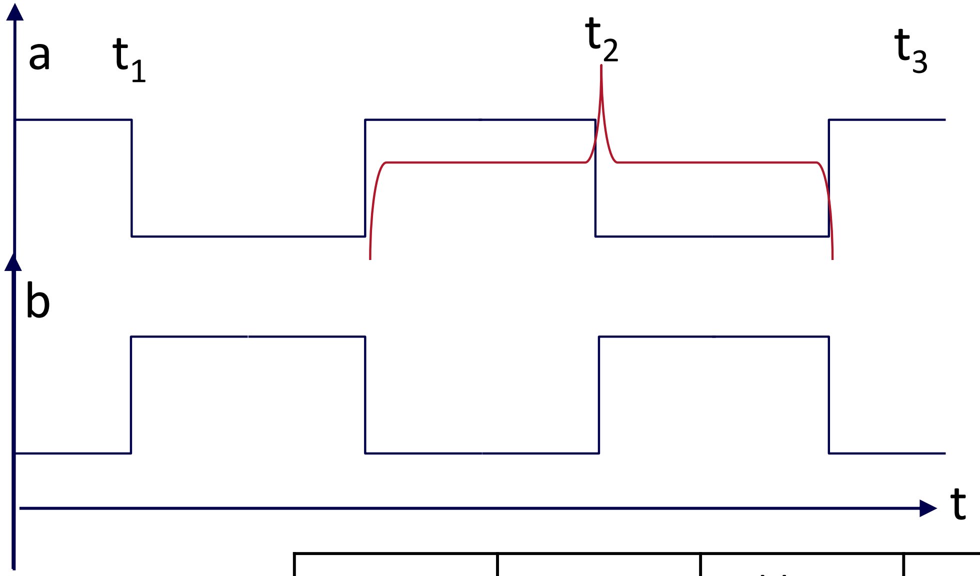


CLK	Reg A	Adder	H Reg
1a	0000	0001	0001
1b	0001	0010	0001
2a	0001	0010	0010
2b	0010	0011	0010
3a			
3b			

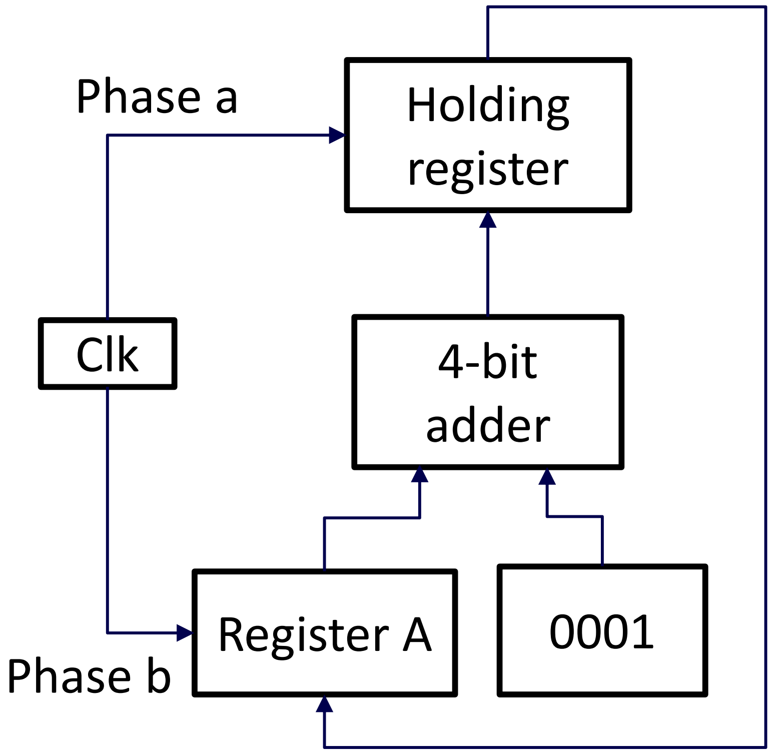


Assume A initial value is 0000

4-bit counter – (5/5)

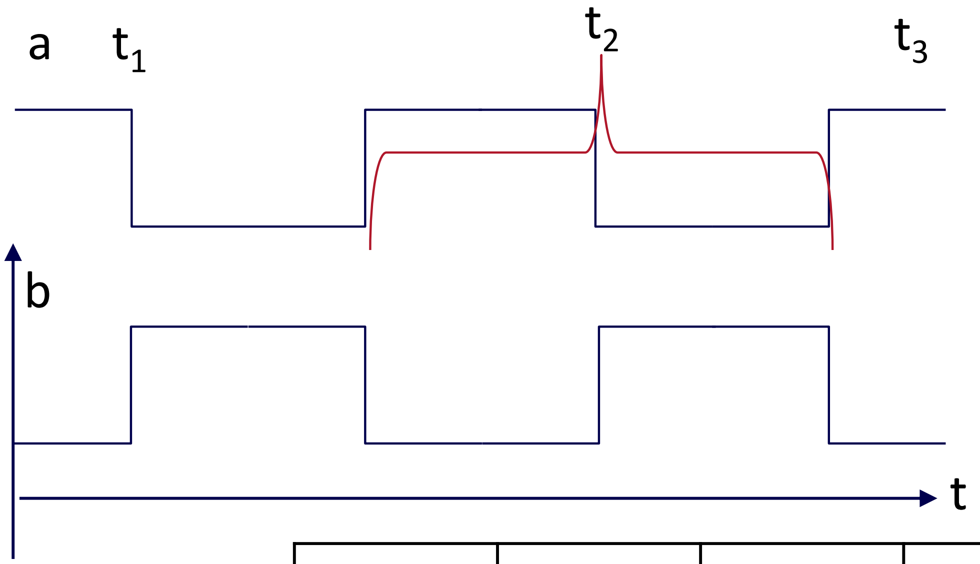


CLK	Reg A	Adder	H Reg
1a	0000	0001	0001
1b	0001	0010	0001
2a	0001	0010	0010
2b	0010	0011	0010
3a			
3b			

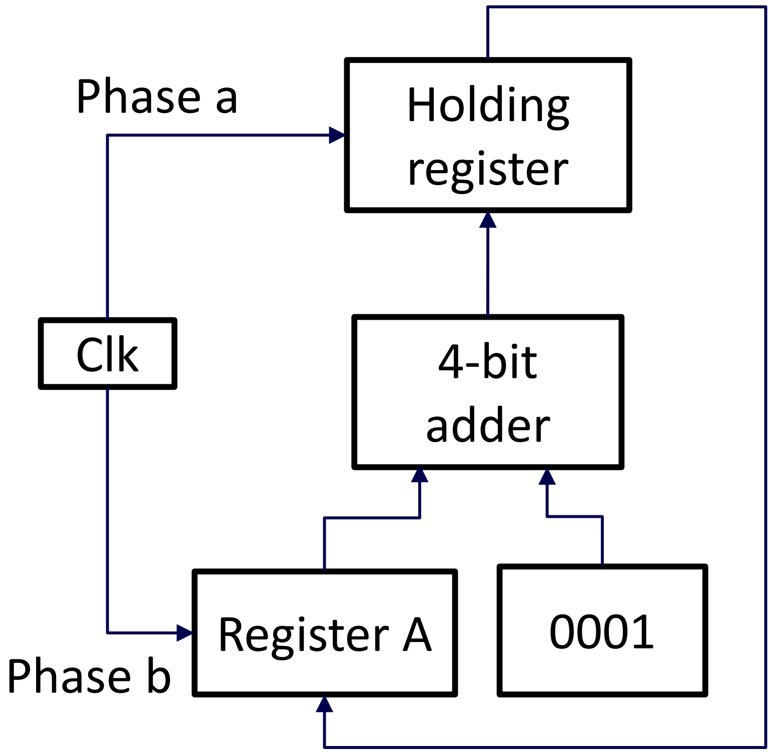


Assume A initial value is 0000

4-bit counter – example - 2

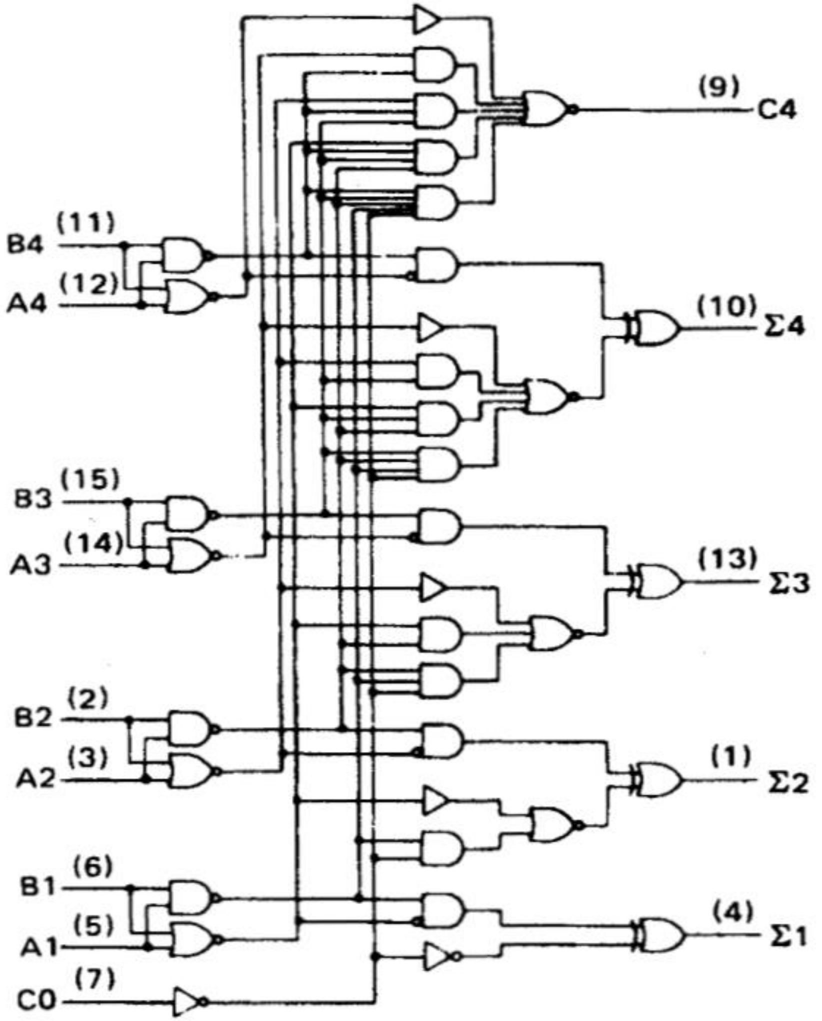


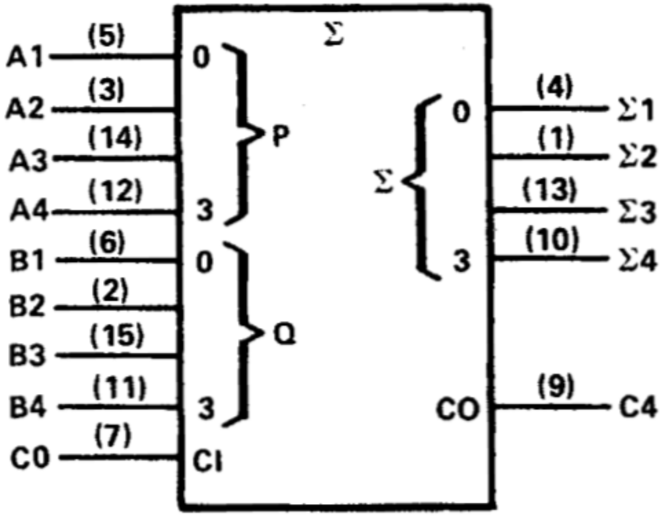
CLK	Reg A	Adder	H Reg
1a	1101	1110	1110
1b			
2a			
2b			
3a			
3b			

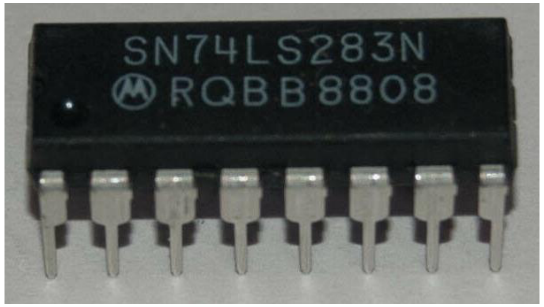


Assume A initial value is 1101

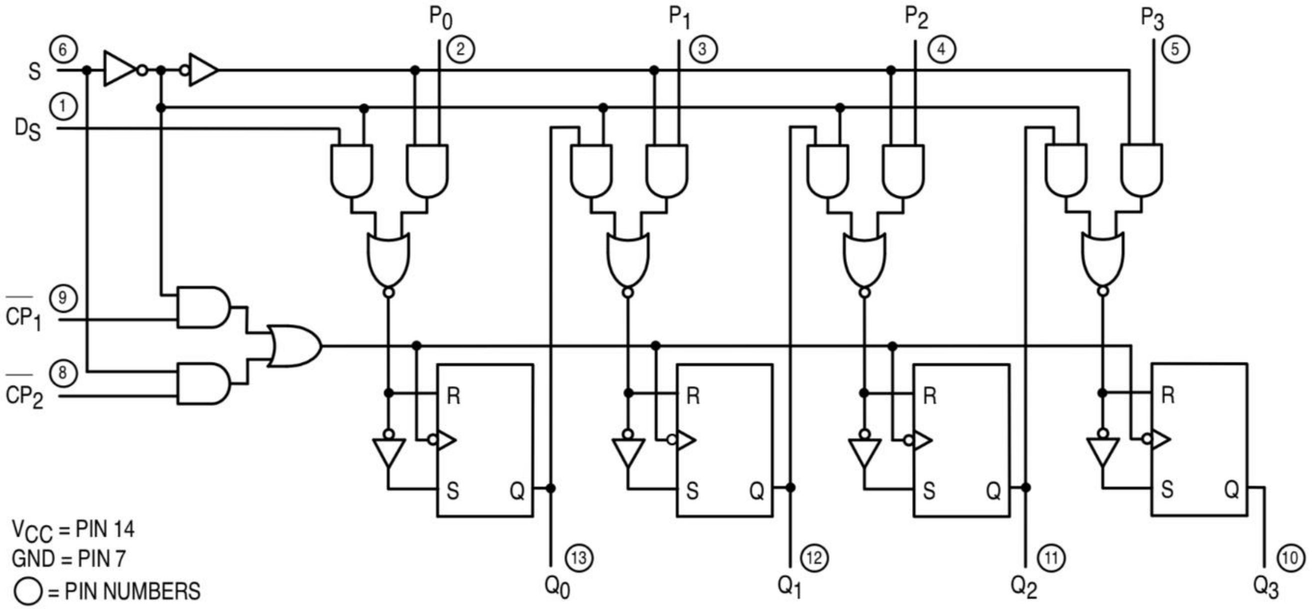
4-BIT BINARY FULL ADDER

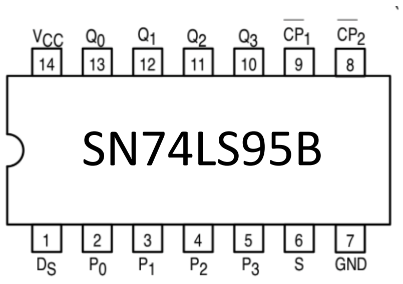


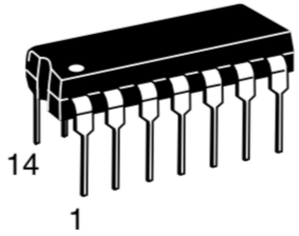




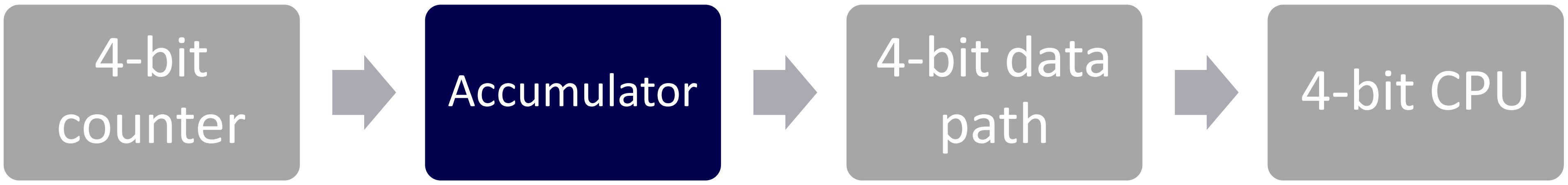
Registers


S
6
DS
1
CP1
9
CP2
8
P0
2
P1
3
P2
4
P3
5
R
S
Q
Q0
13
Q1
12
Q2
11
Q3
10
VCC = PIN 14
GND = PIN 7
○ = PIN NUMBERS


SN74LS95B
VCC 14
Q0 13
Q1 12
Q2 11
Q3 10
CP1 9
CP2 8
1 DS
2 P0
3 P1
4 P2
5 P3
6 S
7 GND


14
1

From 4-bit counter to 4-bit CPU - Accumulator



4-bit counter

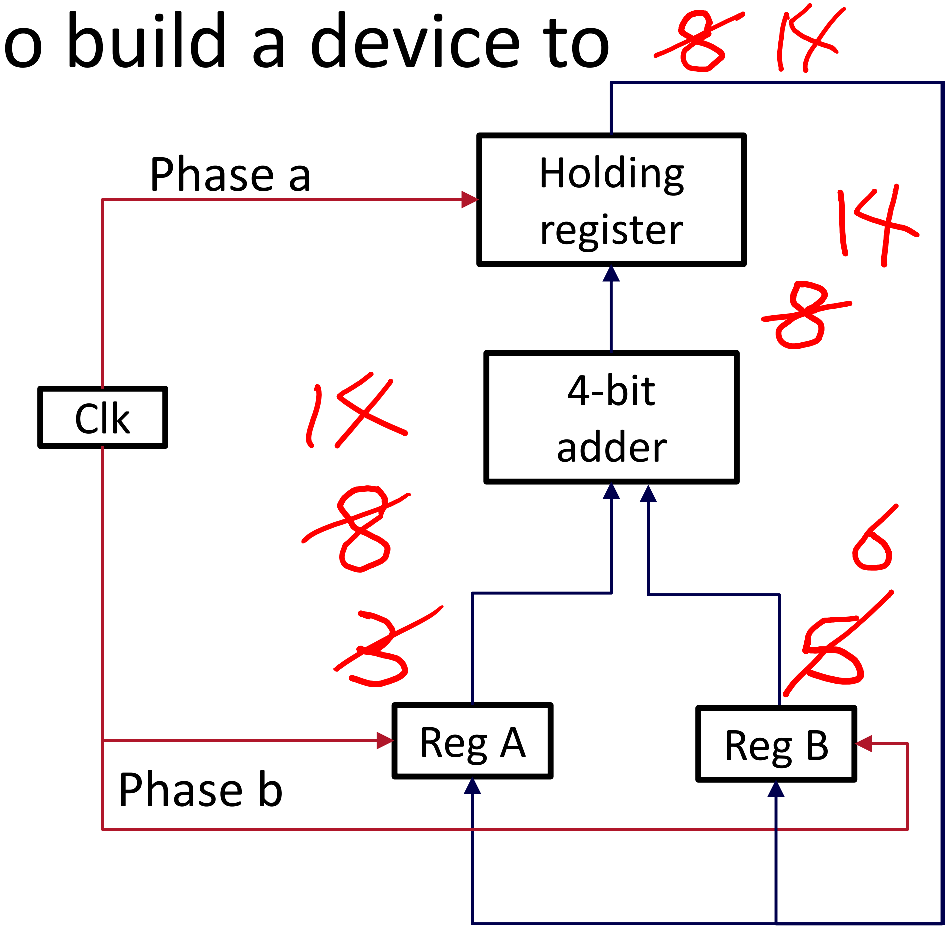
Accumulator

4-bit data path

4-bit CPU

Accumulator

- Upgrade our 4-bit counter to build a device to accumulate numbers.
- How to sum 3,5, and 6.
  1. Start from cleared registers
  2. Load 3 to A
  3. Load 5 to B
  4. Store sum of A and B in A
  5. Load 6 to B
  6. Store sum of A and B in A

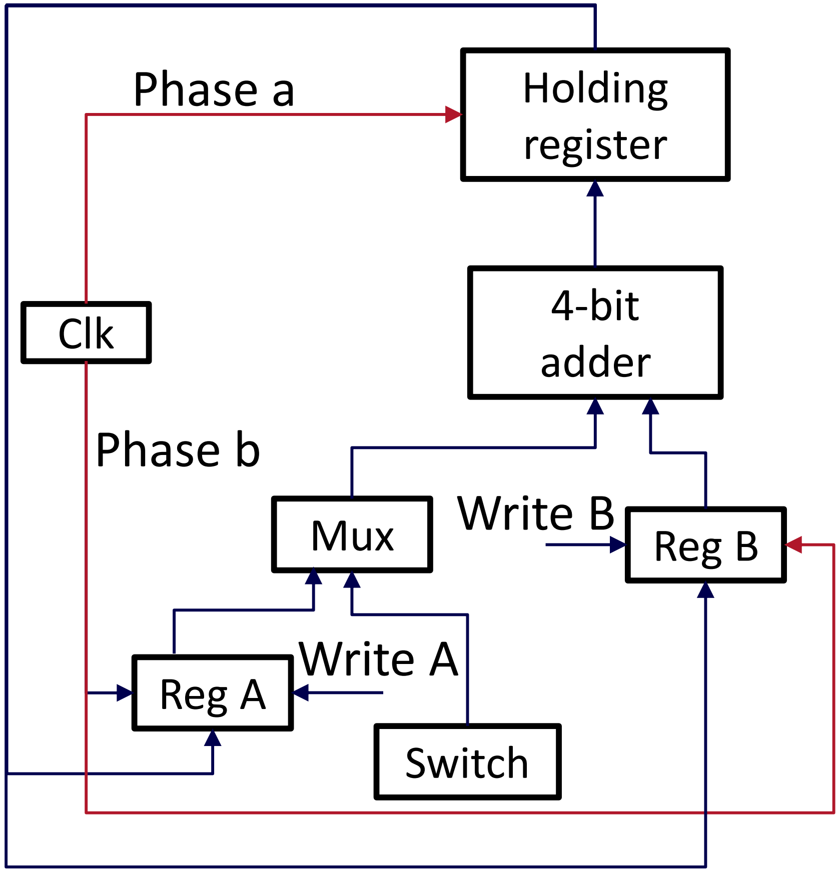


Simple data path – thinking questions

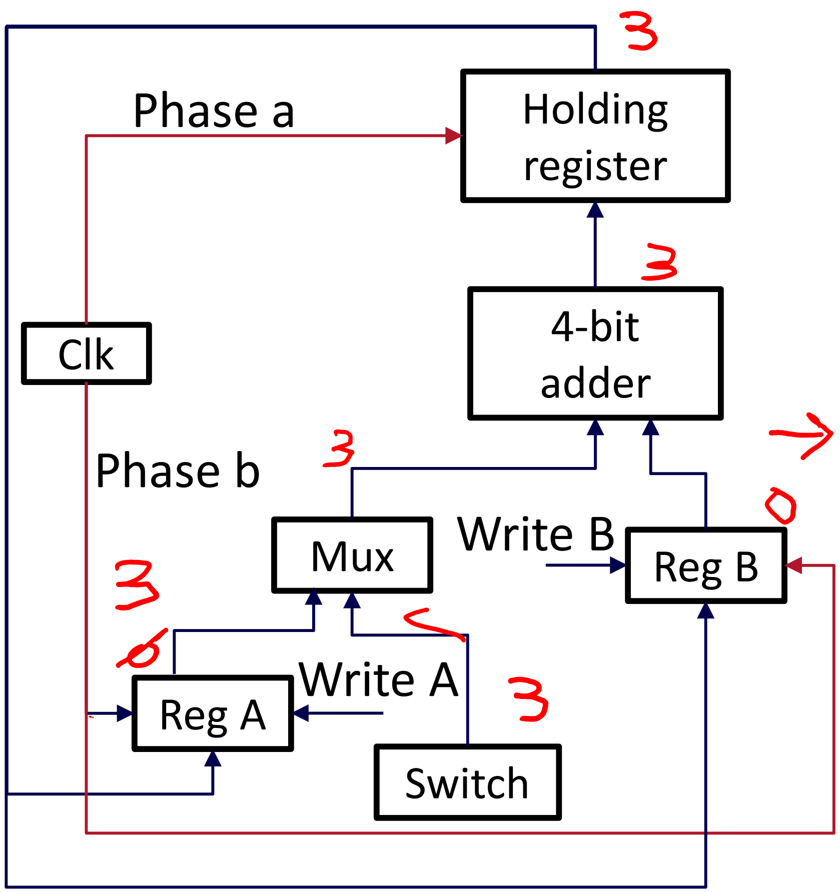
- Think about
  1. How to load the input numbers to the registers?
  2. How will we control which register A or B to update when they are both connected to the same clock phase?
  3. How to switch between the input device and registers?

Simple data path – advanced design

1. To input number, we need a 4-bit switch.
2. To control which registers to write to in the clock phase b, we need a control signal (1: enable, 0: disable).
3. To choose between the input-switch and the reg A, we need a mux.



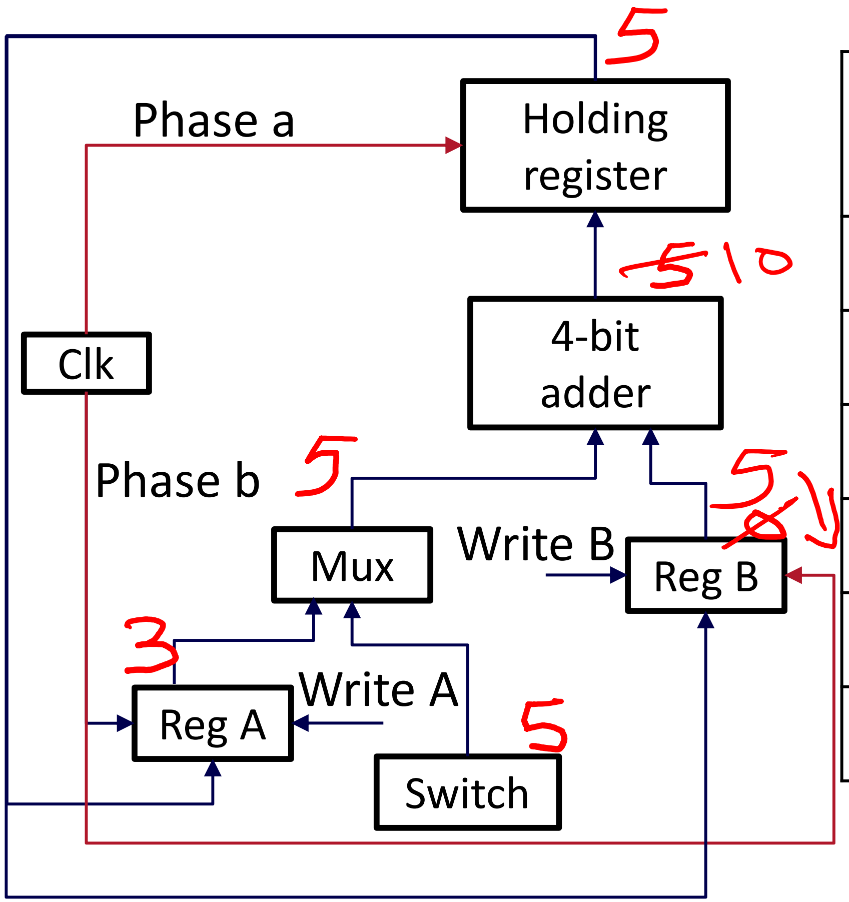
Simple data path (1/5)



- Add 3,5, and 6 and store the result in register A.

Clk	SW	Mux	Write A	Write B	A	B	Adder
1a	0011	SW	x	x	0000	0000	0011
1b	0011	SW	True	False	0011	0000	0011
2a							
2b							
3a							
3b							

Simple data path (2/5)

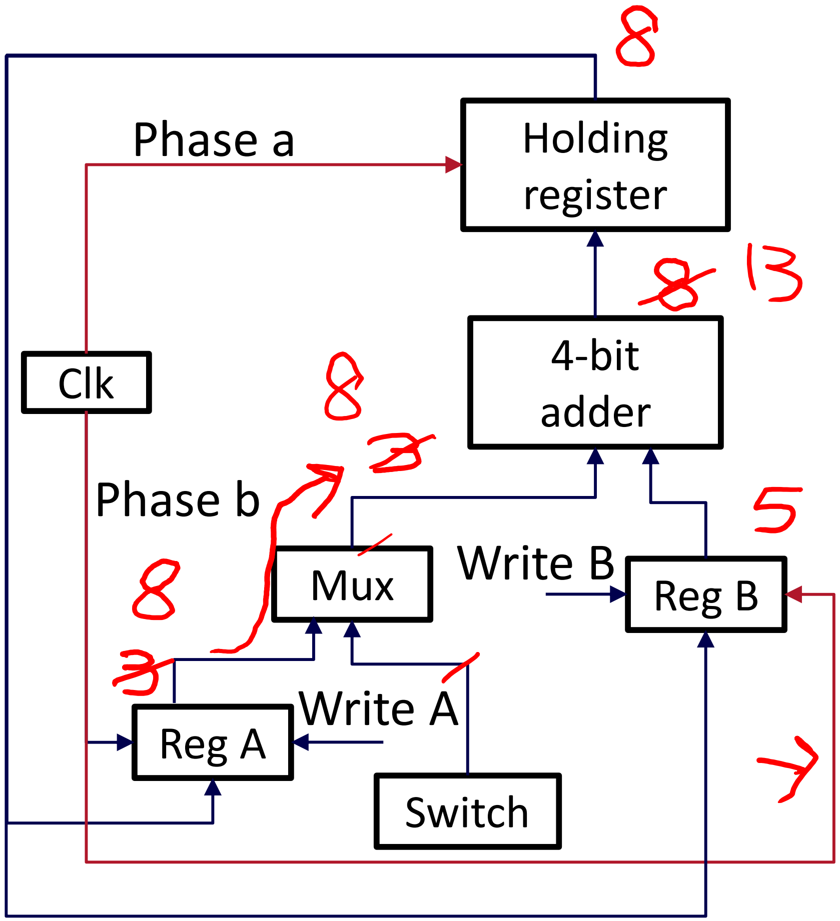


- Add 3,5, and 6 and store the result in register A.

Clk	SW	Mux	Write A	Write B	A	B	Adder
1a	0011	SW	x	x	0000	0000	0011
1b	0011	SW	True	False	0011	0000	0011
2a	0101	SW	x	x	0011	0000	0101
2b	0101	SW	False	True	0011	0101	1010
3a							
3b							

Simple data path (3/5)

- Add 3,5, and 6 and store the result in register A.

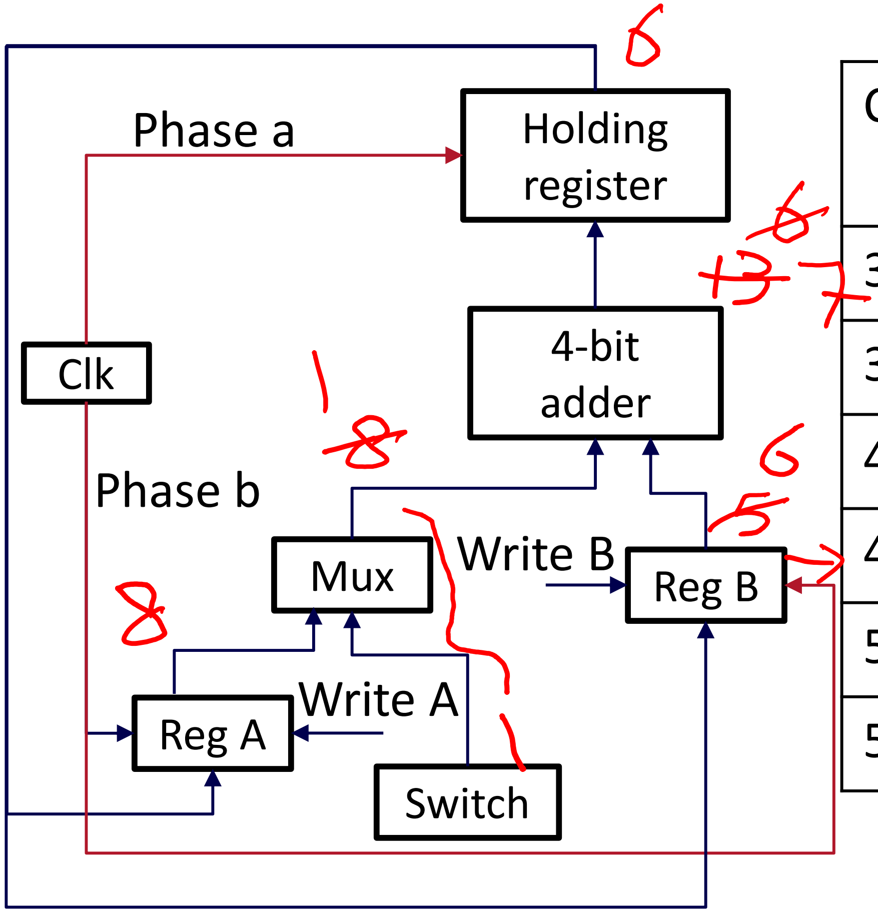


Clk	SW	Mux	Write A	Write B	A	B	Adder
1a	0011	SW	x	x	0000	0000	0011
1b	0011	SW	True	False	0011	0000	0011
2a	0101	SW	x	x	0011	0000	0101
2b	0101	SW	False	True	0011	0101	1010
3a	x	Reg A	x	x	0011	0101	1000
3b	x	Reg A	True	False	1000	0101	1101

Simple data path (4/5)

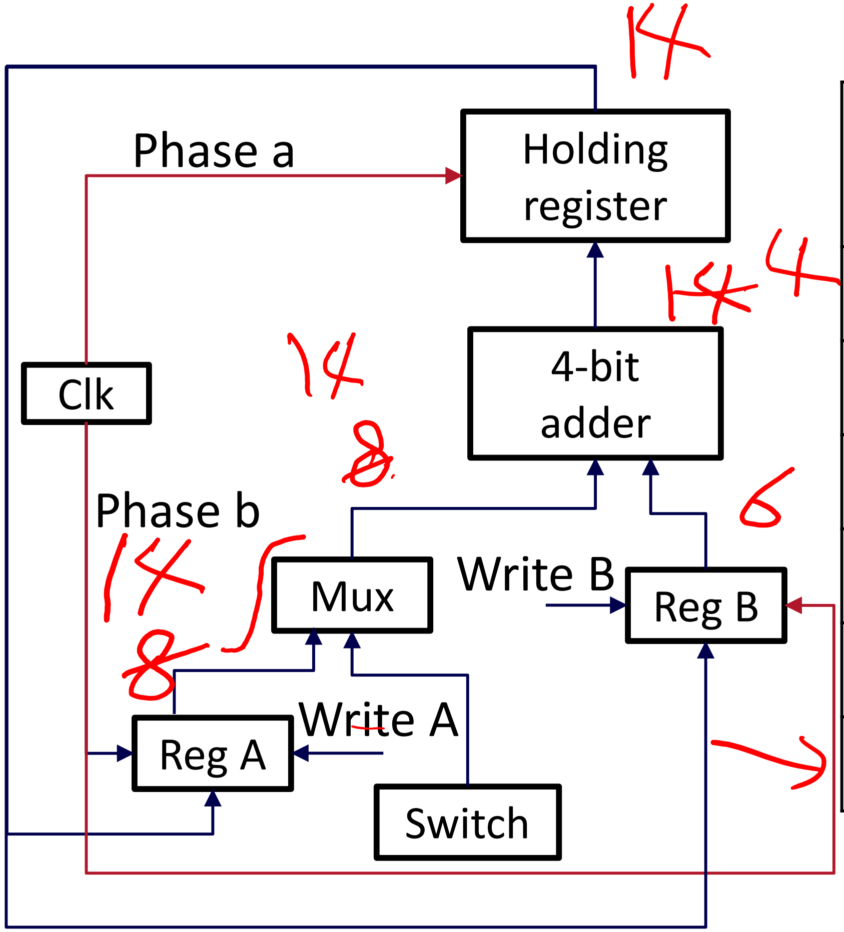
B+1

- Add 3,5, and 6 and store the result in register A.



Clk	SW	Mux	Write A	Write B	A	B	Adder
3a	x	Reg A	x	x	0011	0101	1000
3b	x	Reg A	True	False	1000	0101	1101
4a	0001	SW	x	x	1000	0101	0110
4b	0001	SW	False	True	1000	0110	0111
5a							
5b							

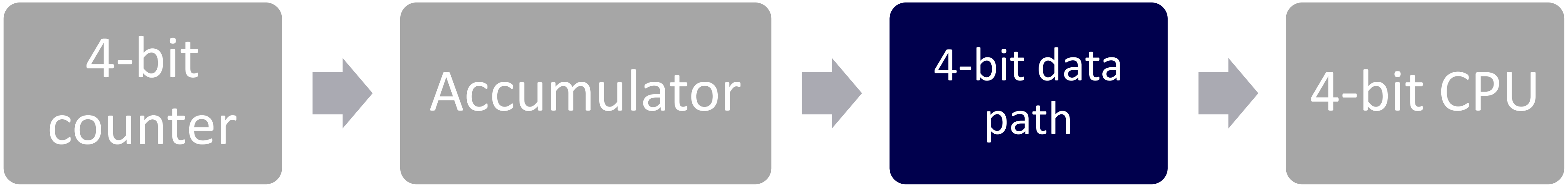
Simple data path (5/5)



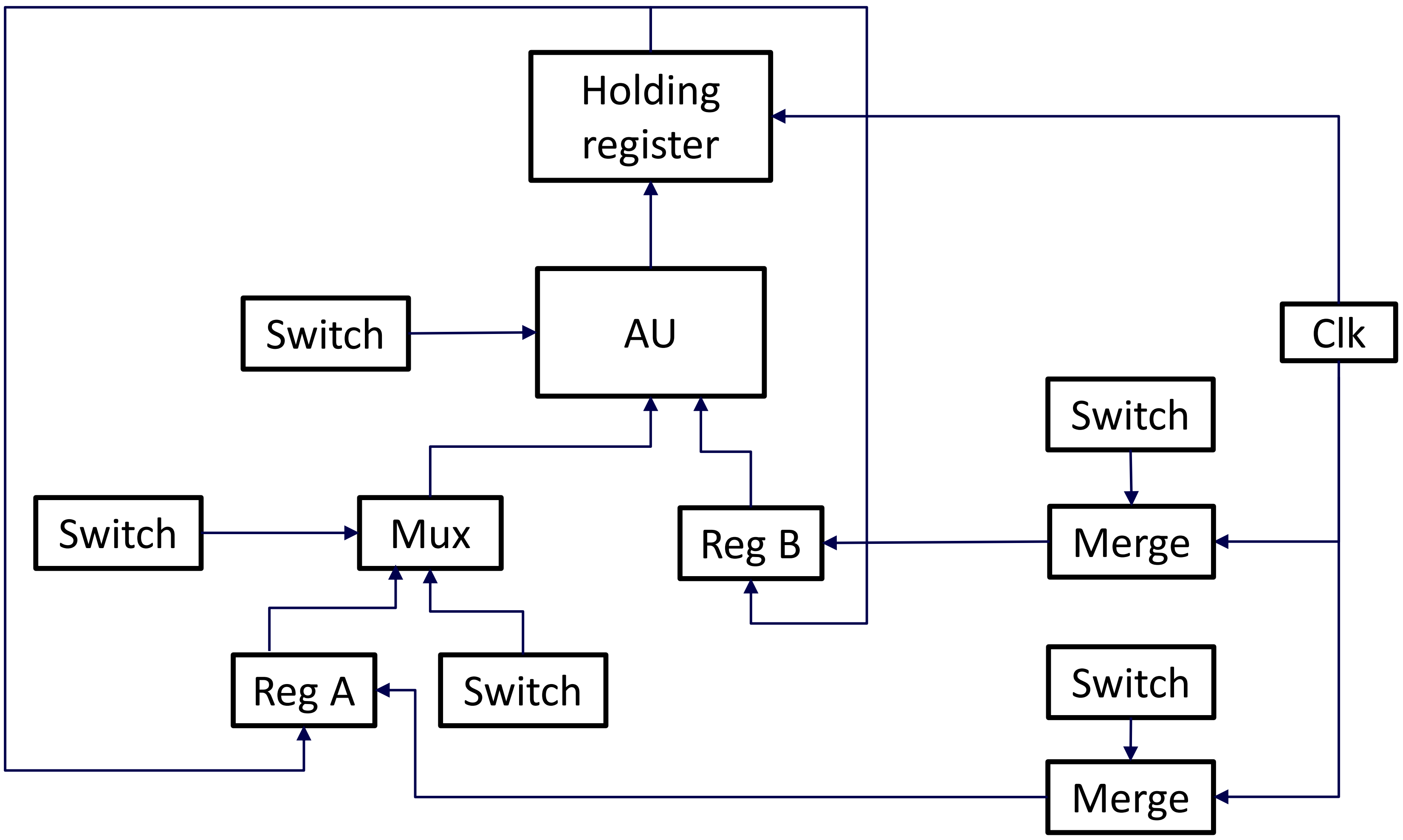
- Add 3,5, and 6 and store the result in register A.

Clk	SW	Mux	Write A	Write B	A	B	Adder
3a	x	Reg A	x	x	0011	0101	1000
3b	x	Reg A	True	False	1000	0101	1101
4a	0001	SW	x	x	1000	0101	0110
4b	0001	SW	False	True	1000	0110	0111
5a	x	Reg A	x	x	1000	0110	1110
5b	x	Reg A	True	False	1110	0110	0100

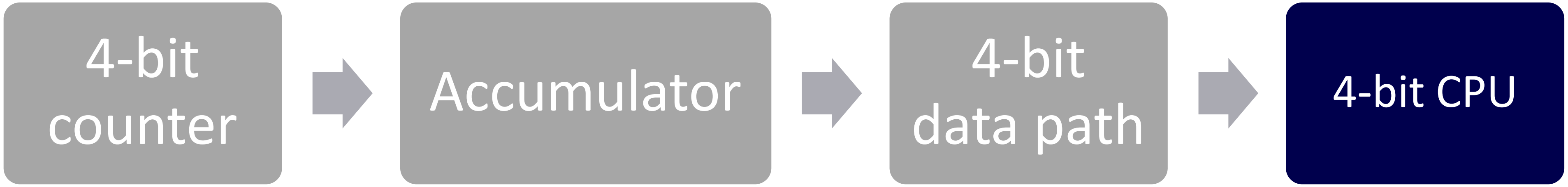
From 4-bit counter to 4-bit CPU – Data path



4-bit data path - manual



From 4-bit counter to 4-bit CPU – Data path



4-bit CPU concept

1. Store data and instructions in memory.

2. Fetch instruction from memory.

3. Execute these instructions.

Data in memory

- How are data stored in memory?
- Data sizes:
  - Nipple = 4 bits
  - Byte = 8 bits
  - Halfword = 16 bits
  - Word = 32 bits
- Store 3,5, and 6.

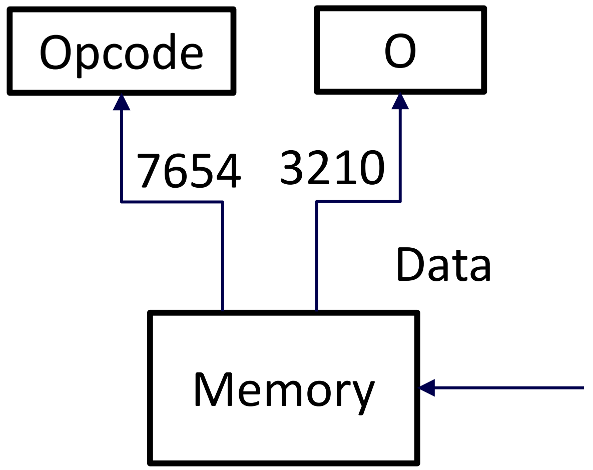
Data

7	6	5	4	3	2	1	0	Address
x	x	x	x	0	0	1	1	0b0000
x	x	x	x	0	1	0	1	0b0001
x	x	x	x	0	1	1	0	0b0010
x	x	x	x	x	x	x	x	0b0011
…	…	…	…	…	…	…	…	…
x	x	x	x	x	x	x	x	0b1111

Instructions in memory

- Add 3,5, and 6, then subtract 2

	Opcode				Operand				
	7	6	5	4	3	2	1	0	Address
Instruction	x	x	x	0	0	0	1	1	0b0000
	x	x	x	0	0	1	0	1	0b0001
	x	x	x	0	0	1	1	0	0b0010
	x	x	x	1	0	0	1	0	0b0011
	…	…	…	…	…	…	…	…	…
	x	x	x	x	x	x	x	x	0b1111



4-bit CPU concept

1. Store data and instructions in memory.

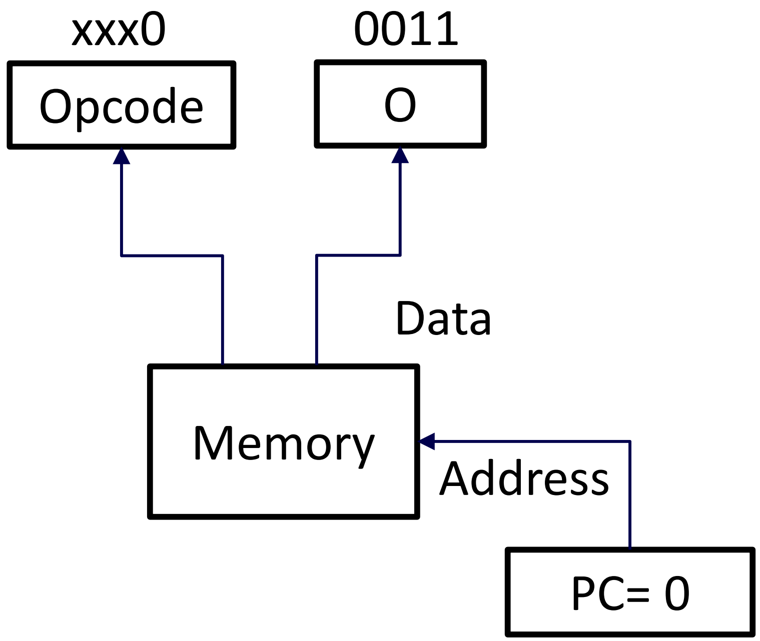
2. Fetch instruction from memory.

3. Execute these instructions.

Fetch instructions (1/4)

- Add 3,5, and 6, then subtract 2

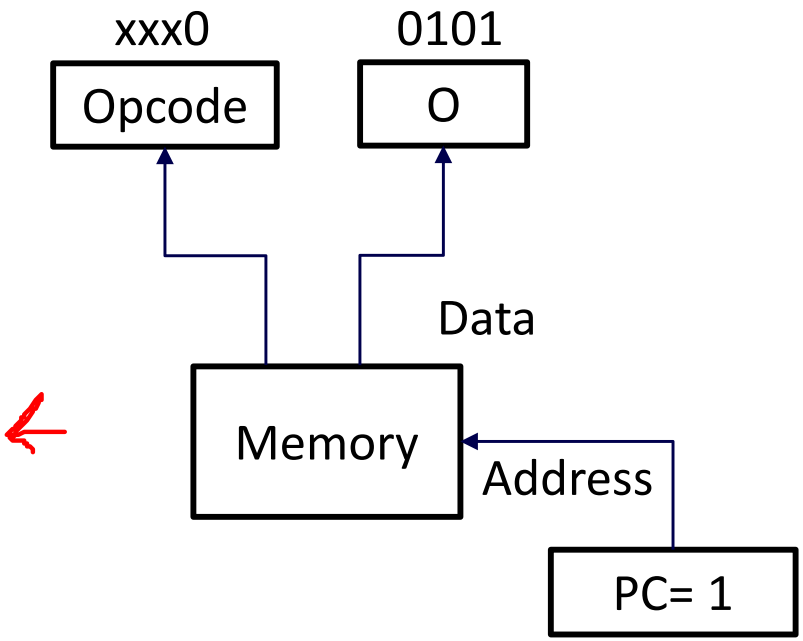
	Opcode				Operand				
	7	6	5	4	3	2	1	0	Address
Instruction	x	x	x	0	0	0	1	1	0b0000
	x	x	x	0	0	1	0	1	0b0001
	x	x	x	0	0	1	1	0	0b0010
	x	x	x	1	0	0	1	0	0b0011
	…	…	…	…	…	…	…	…	…
	x	x	x	x	x	x	x	x	0b1111



Fetch instructions (2/4)

- Add 3,5, and 6, then subtract 2

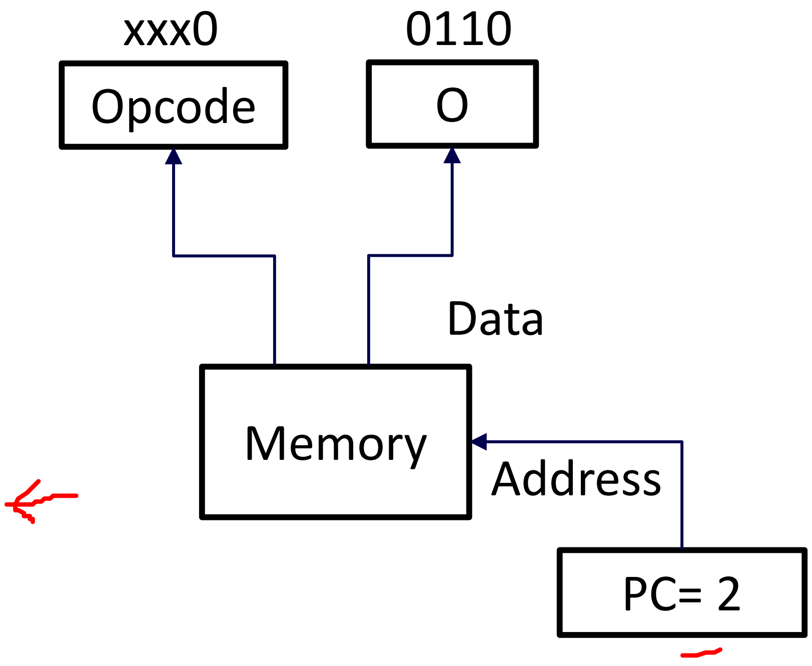
	Opcode				Operand				
	7	6	5	4	3	2	1	0	Address
Instruction	x	x	x	0	0	0	1	1	0b0000
	x	x	x	0	0	1	0	1	0b0001
	x	x	x	0	0	1	1	0	0b0010
	x	x	x	1	0	0	1	0	0b0011
	…	…	…	…	…	…	…	…	…
	x	x	x	x	x	x	x	x	0b1111



Fetch instructions (3/4)

- Add 3,5, and 6, then subtract 2

	Opcode				Operand				
	7	6	5	4	3	2	1	0	Address
Instruction	x	x	x	0	0	0	1	1	0b0000
	x	x	x	0	0	1	0	1	0b0001
	x	x	x	0	0	1	1	0	0b0010
	x	x	x	1	0	0	1	0	0b0011
	…	…	…	…	…	…	…	…	…
	x	x	x	x	x	x	x	x	0b1111

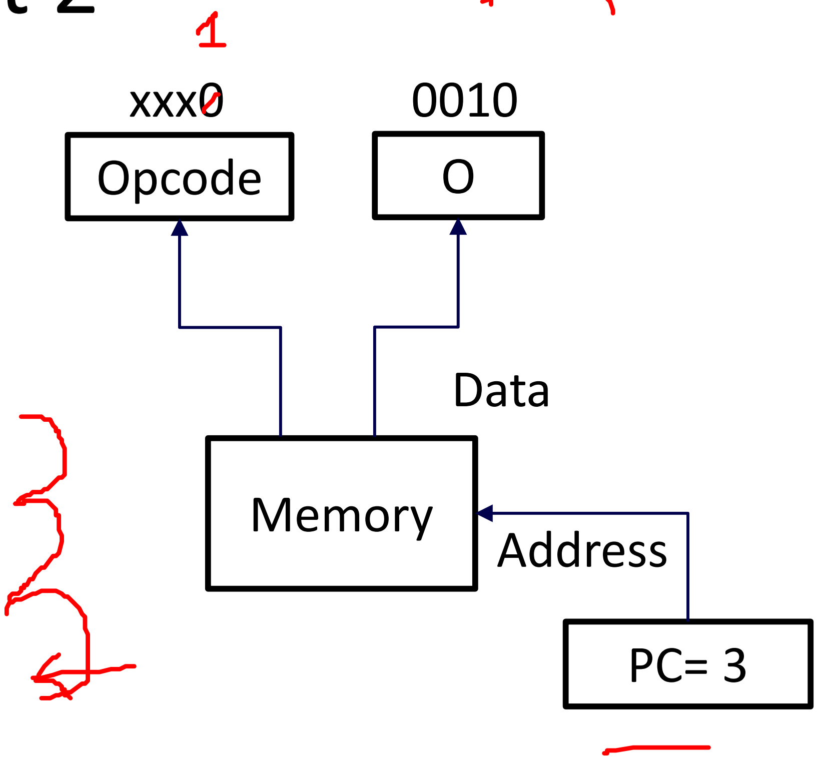


Fetch instructions (4/4)

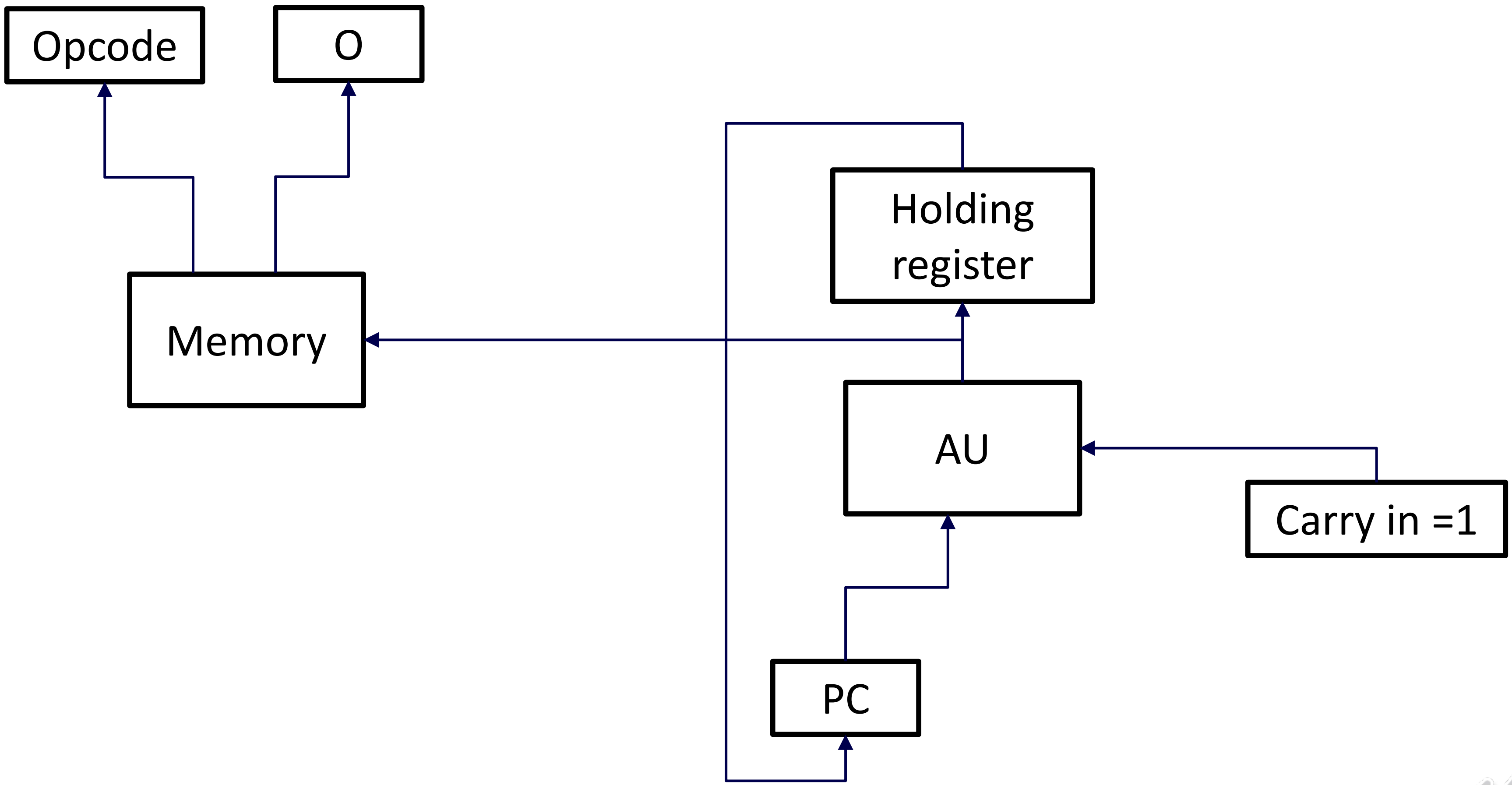
- Add 3,5, and 6, then subtract 2

A = 12

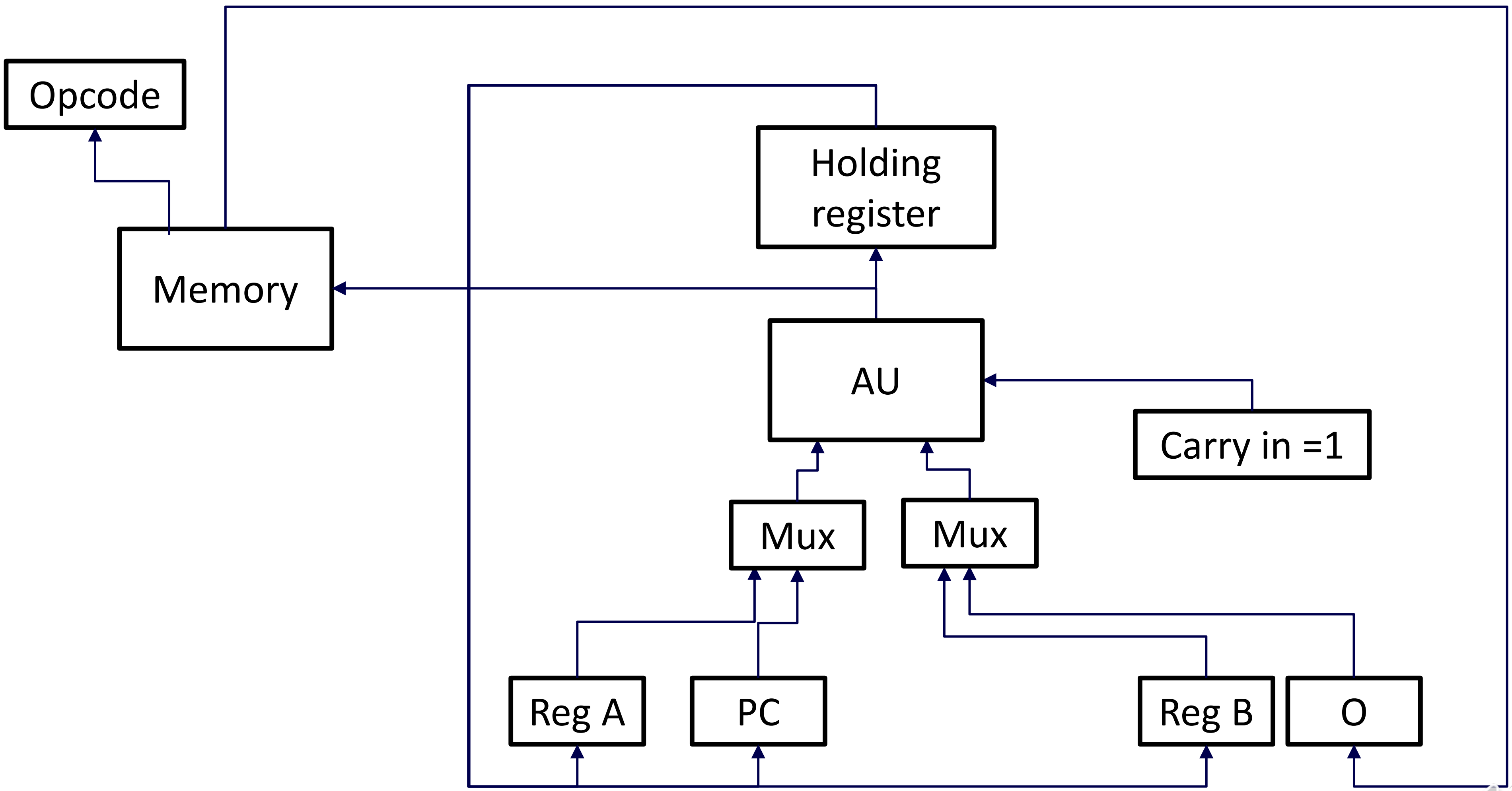
	Opcode				Operand				
	7	6	5	4	3	2	1	0	Address
Instruction	x	x	x	0	0	0	1	1	0b0000
	x	x	x	0	0	1	0	1	0b0001
	x	x	x	0	0	1	1	0	0b0010
	x	x	x	1	0	0	1	0	0b0011
	…	…	…	…	…	…	…	…	…
	x	x	x	x	x	x	x	x	0b1111



Program counter



4-bit data path with PC and Memory



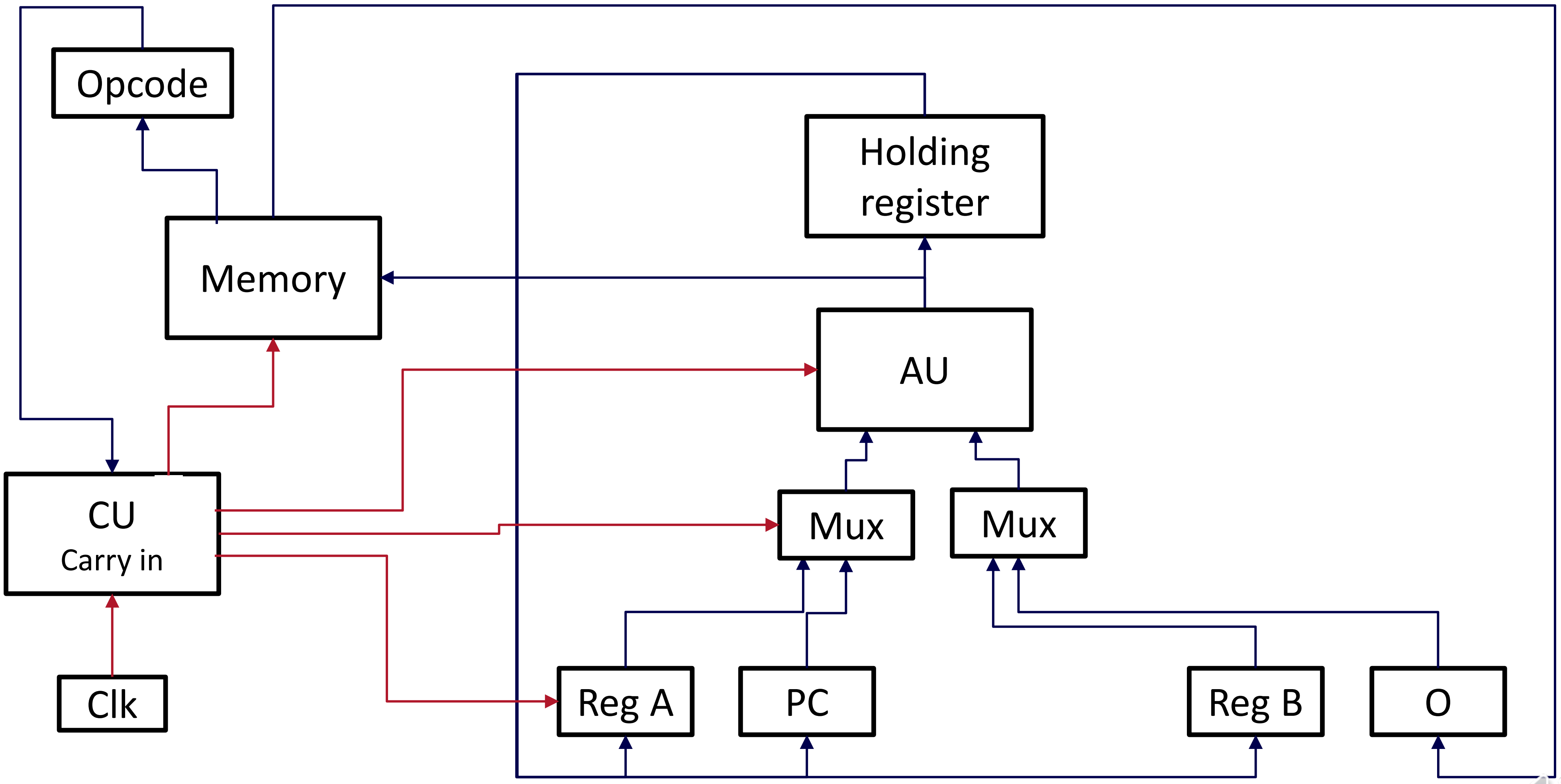
4-bit CPU concept

1. Store data and instructions in memory.

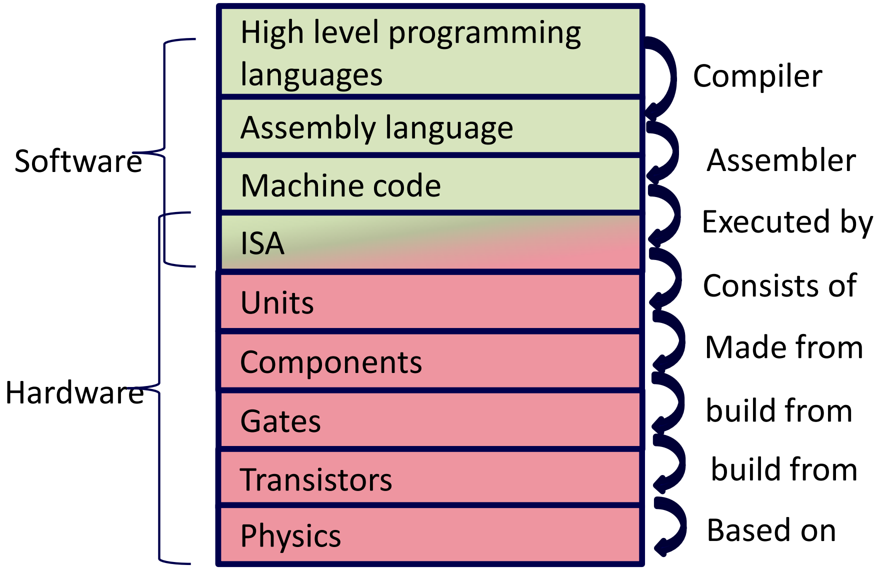
2. Fetch instruction from memory.

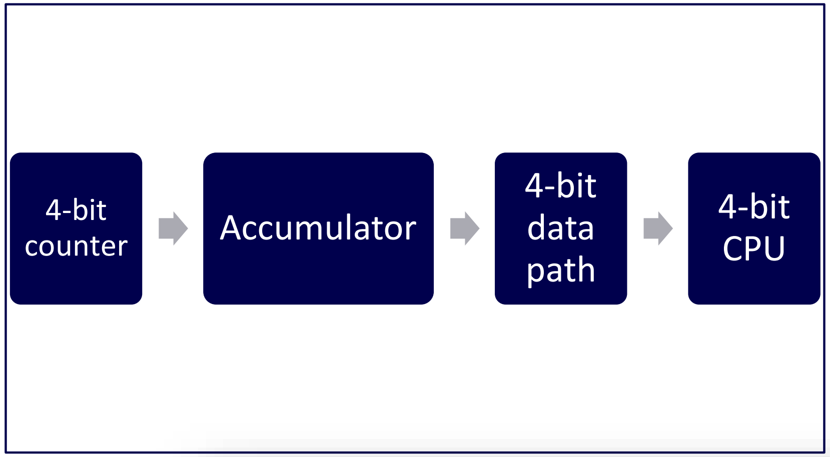
3. Execute these instructions.

4-bit CPU


Opcode
Memory
Holding register
AU
CU
Carry in
Clk
Mux
Mux
Reg A
PC
Reg B
O

Summary





How to sum 3,5, and 6.

1. Start from cleared registers
2. Load 3 to A
3. Load 5 to B
4. Store sum of A and B in A
5. Load 6 to B
6. Store sum of A and B in A

Opcode				Operand				
7	6	5	4	3	2	1	0	Address
x	x	x	0	0	0	1	1	0b0000
x	x	x	0	0	1	0	1	0b0001
x	x	x	0	0	1	1	0	0b0010
x	x	x	1	0	0	1	0	0b0011
…	…	…	…	…	…	…	…	…
x	x	x	x	x	x	x	x	0b1111