Numbers and Memory

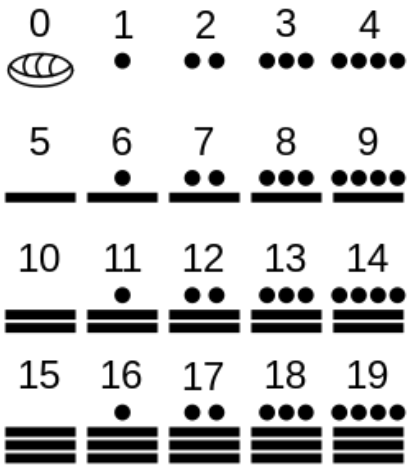
Kasper Brink
November 21, 2018

(Images taken from Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.)

Number Systems

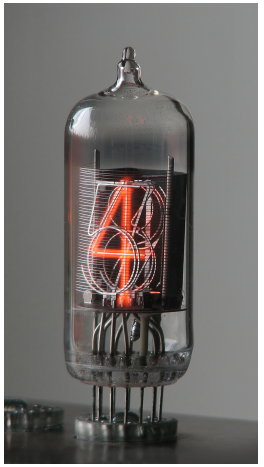
Which systems for representing numbers do you know?

- Arabic: 0 1 2 3 4 5 . . .
- Roman: MMXVII
- Mayan:



Positional Systems

- Everyday life: list of digits, (usually) decimal system
- Computers: bit patterns, always binary system
- Both are *positional systems*.



Number Systems: Decimal

- General form of the number representation:

$$\ldots\ d_2\ d_1\ d_0\ .\ d_{-1}\ d_{-2}\ \ldots$$

- Value of this decimal representation:

$$\ldots + 10^2 d_2 + 10^1 d_1 + 10^0 d_0 + 10^{-1} d_{-1} + 10^{-2} d_{-2} + \ldots$$

- Base or *radix* is 10

Number Systems: Binary

- General form of the number representation:

$$\ldots\ d_2\ d_1\ d_0\ .\ d_{-1}\ d_{-2}\ \ldots$$

- Value of this binary representation:

$$\ldots + 2^2 d_2 + 2^1 d_1 + 2^0 d_0 + 2^{-1} d_{-1} + 2^{-2} d_{-2} + \ldots$$

- Base is 2

- “Bit” is binary digit

- Leftmost bit is *most significant bit* (MSB)
  Rightmost bit is *least significant bit* (LSB)

Number Systems: Arbitrary Base

- General form of the number representation:

$$\ldots\ d_2\ d_1\ d_0\ .\ d_{-1}\ d_{-2}\ \ldots$$

- Value of this representation:

$$\ldots + a^2 d_2 + a^1 d_1 + a^0 d_0 + a^{-1} d_{-1} + a^{-2} d_{-2} + \ldots$$

- Base is a

Commonly Used Number Systems

Name	Base	Usage
decimal	10	everyday life
binary	2	implementation of computers
octal	8	discussing computers
hexadecimal	16	discussing computers

- Octal digits: $0, 1, \ldots, 7$
- Hexadecimal digits: $0, \ldots, 9, A, \ldots, F$

Converting to Another Base

- Convert decimal to binary

  $53_{\text{dec}} = 110101_{\text{bin}}$

- Convert binary to decimal

  $01101101_{\text{bin}} = 109_{\text{dec}}$

- Alternative method, based on repeated halving/doubling: see Tanenbaum, Appendix A.3.

Arithmetic on Binary Numbers

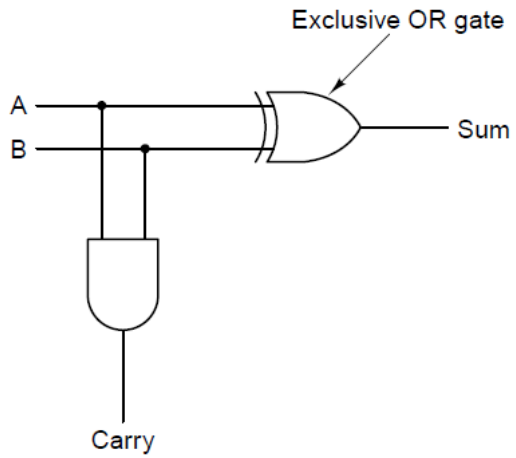
- Analogous to pen-and-paper addition/subtraction on decimal numbers

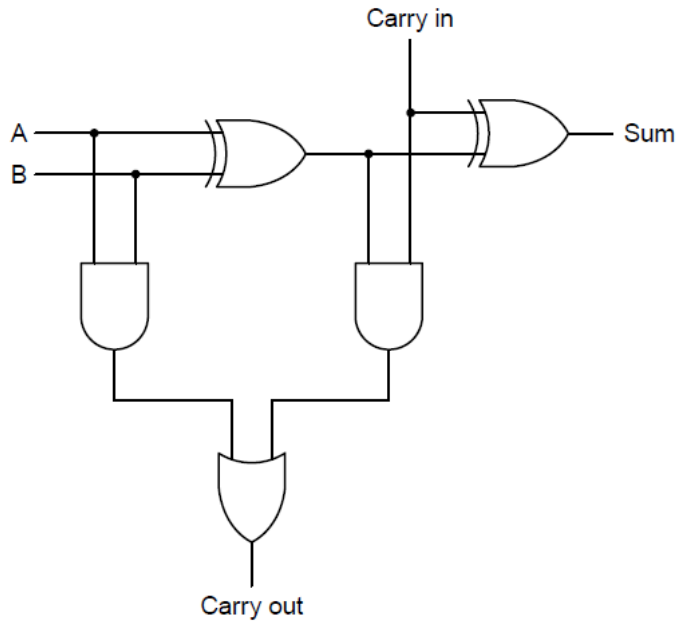
- Addition

  $1\,1011_{\text{bin}} + 1110_{\text{bin}} = 10\,1001_{\text{bin}}$

- Subtraction is similar (when result for a column < 0, must “borrow” from column on the right)

Hardware for Addition





Half Adder

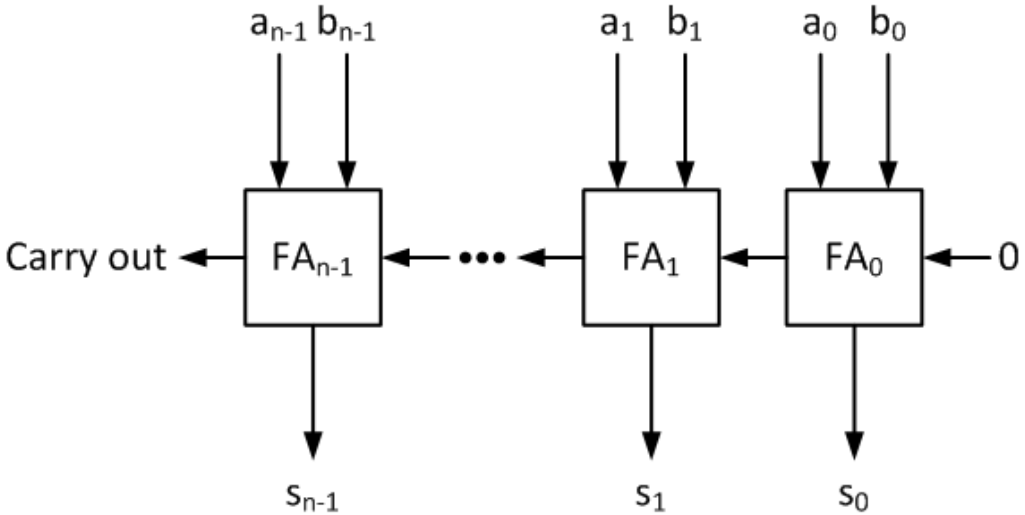
$$\text{Sum} = A \oplus B$$
$$\text{Carry} = AB$$

Full Adder

$$\text{Sum} = A \oplus B \oplus C_{in}$$
$$\text{Carry out} = AB + (A \oplus B)C_{in}$$

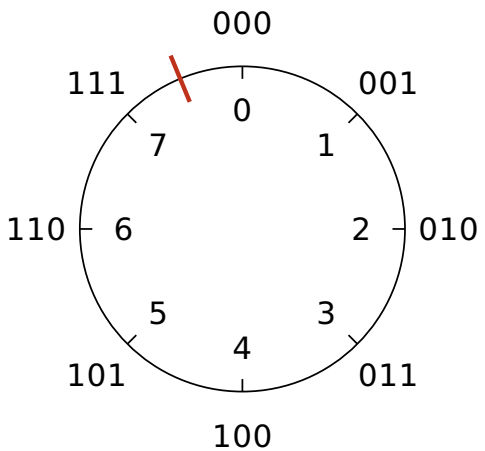
Hardware for Addition: *n*-bit Numbers

By chaining *n* full adders, we can create a circuit for the addition of *n*-bit numbers:



Finite Representation

- Binary system can represent arbitrarily large numbers
- In an actual computer, space for storing bits is limited
- CPU performs calculations using a fixed, limited number of bits (e.g. 32 or 64)
- If a calculation exeeds the largest number representable in n bits, the result “wraps around” to 0



(n-bit binary numbers: residue classes modulo 2^n)

Negative Numbers

Different representations possible.

- Signed Magnitude: MSB is sign bit, remaining bits are magnitude (absolute value of number)

  e.g. $0111_{\text{bin}} = 7_{\text{dec}}$

  $1111_{\text{bin}} = -7_{\text{dec}}$

- Two's Complement: based on idea of all calculations taking place *modulo* 2^n.
  Negative number x represented as $x + 2^n$.

- Other representations: see Tanenbaum.

We will always use two's complement representation.

Two's Complement

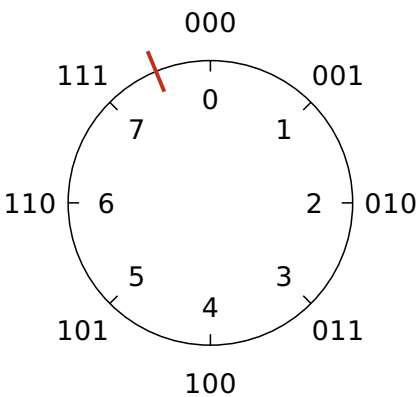
n bits $\rightarrow 2^n$ bit patterns

Unsigned:

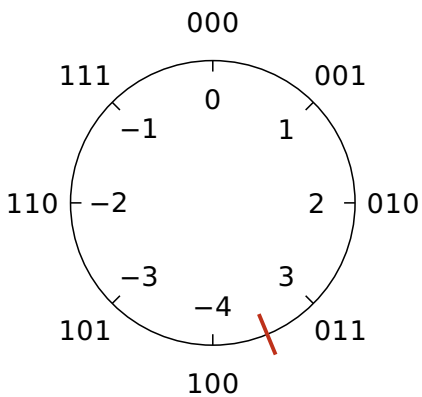
Sign	From	To	Represents
+	0000..0	1111..1	$0, \ldots, 2^n - 1$

Signed / Two's Complement:

Sign	From	To	Represents	
+	0000..0	0111..1	$0, \ldots, 2^{n-1} - 1$	(as unsigned)
-	1000..0	1111..1	$-2^{n-1}, \ldots, -1$	$(x \rightarrow x + 2^n)$



Unsigned



Signed

Converting to Two's Complement

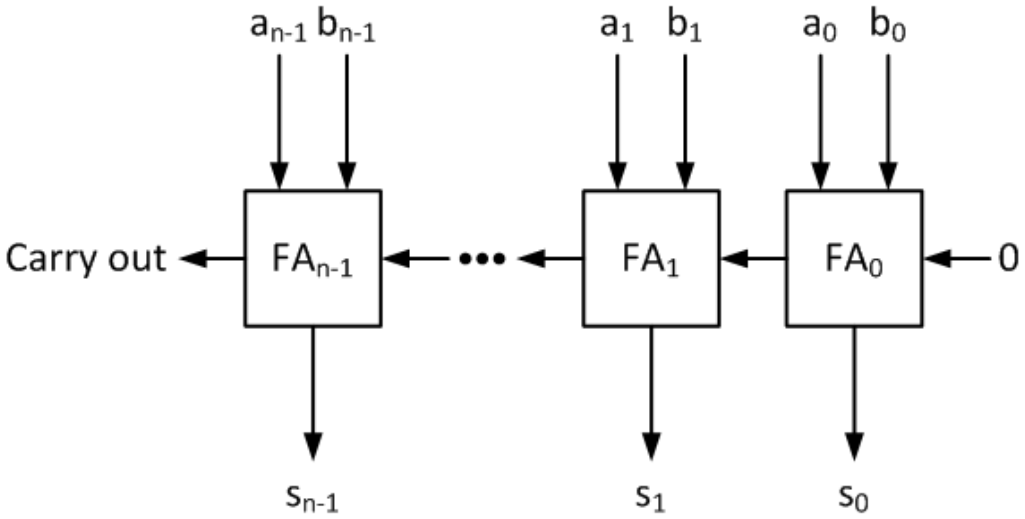
Two's complement representation of negative number m:

- find unsigned representation of magnitude $|m|$
- invert all bits
- add 1

Example (8 bits): $-57_{\text{dec}} = ??_{\text{bin}}$

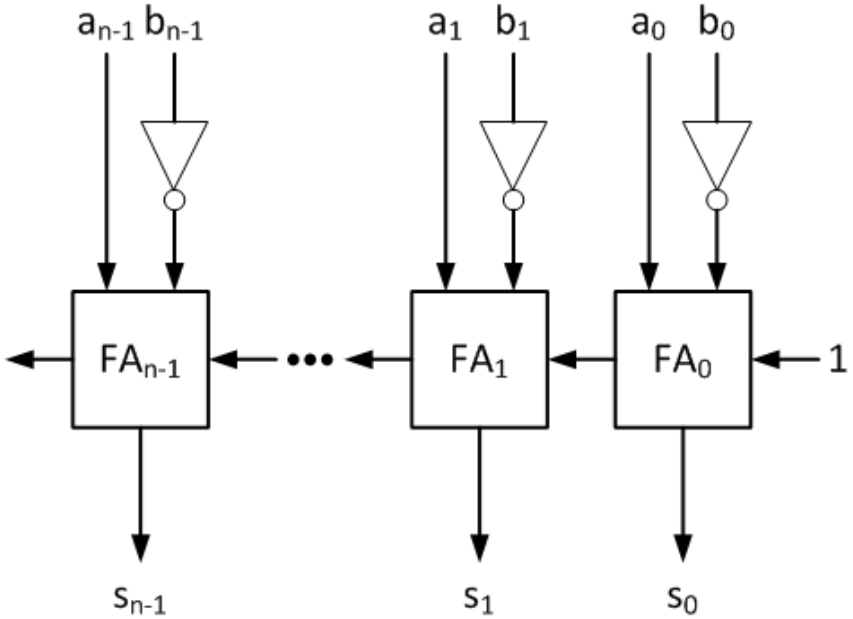
$$57_{\text{dec}} = 0011\,1001_{\text{bin}}$$
$$-57_{\text{dec}} = 1100\,0110_{\text{bin}} + 1 = 1100\,0111_{\text{bin}}$$

Hardware for Addition (recap)



Can we adapt this circuit to perform subtraction?

Hardware for Subtraction



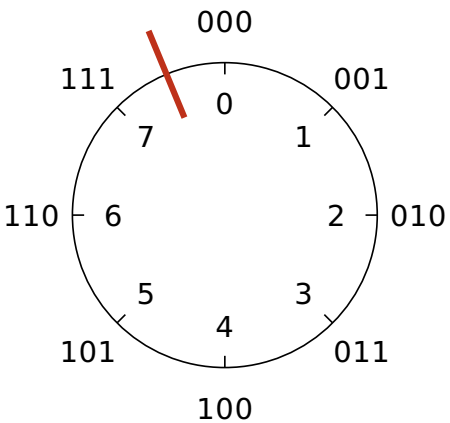
Notice that $a - b = a + (-b)$

Negation: invert all bits and add 1 (use carry-in of LSB!)

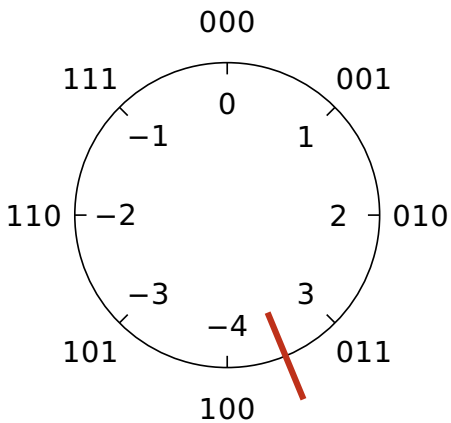
Overflow

Overflow occurs when result of computation does not fit in available bits.

- Unsigned: overflow when carry-out of MSB is 1.
- Signed: overflow when sign of operands is the same, and opposite to sign of result.



Unsigned



Signed

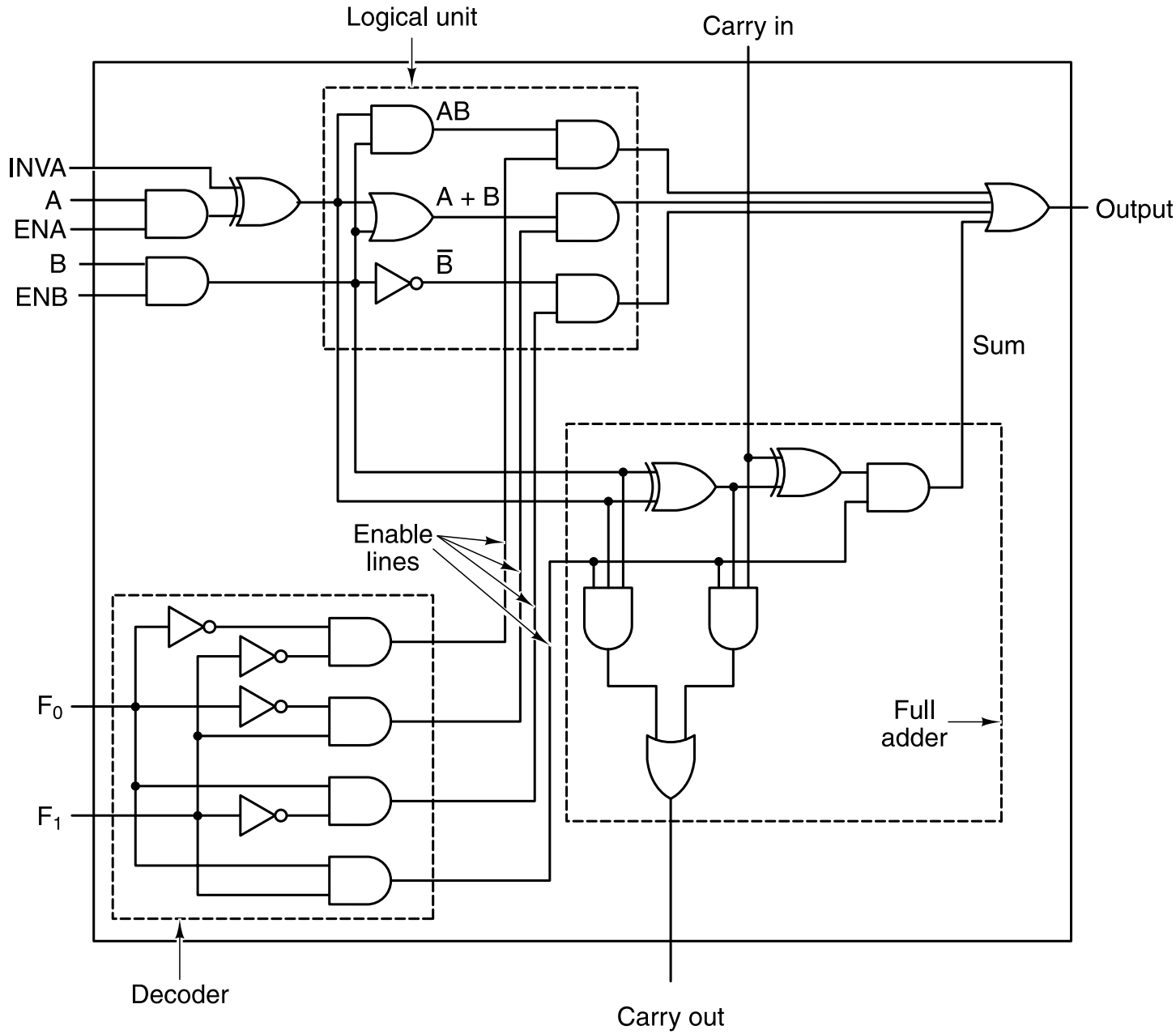
Detecting Overflow

CPU contains *flags* that are set or cleared according to the outcome of the last arithmetic operation:

- Carry (“C”): carry-out of MSB.
  Only relevant for *unsigned* computations.
- Overflow (“O”): sign of operands is the same and opposite to sign of result.
  Only relevant for *signed* computations.

CPU doesn’t know if it is performing signed or unsigned computation; depends on interpretation of the programmer.

ALU


Logical unit
Carry in
AB
A + B
$\overline{B}$
INVA
A
ENA
B
ENB
Output
Sum
Enable lines
F_0
F_1
Full adder
Decoder
Carry out

Memory

Combinational and Sequential Logic

Combinational circuit

- output of circuit depends only on input (at the current time)
- implements a Boolean function

Sequential circuit

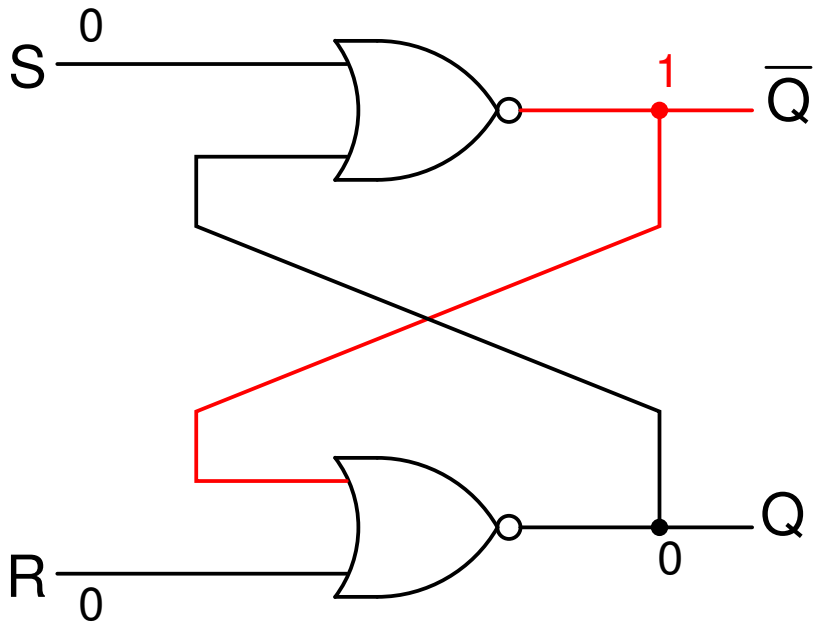
- output of circuit depends on input and *internal state*
- implements a finite state automaton
- “circuit with memory”

How to make a circuit with memory?

Is it possible to create a sequential circuit using only combinational components (gates, wires)?

Yes, by using feedback.

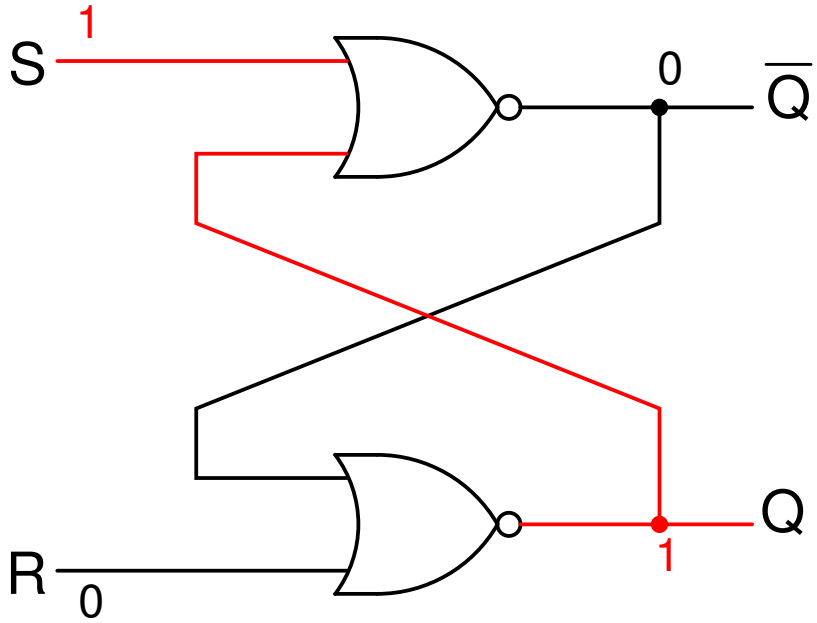
SR Latch



For S = R = 0 there are two stable states:

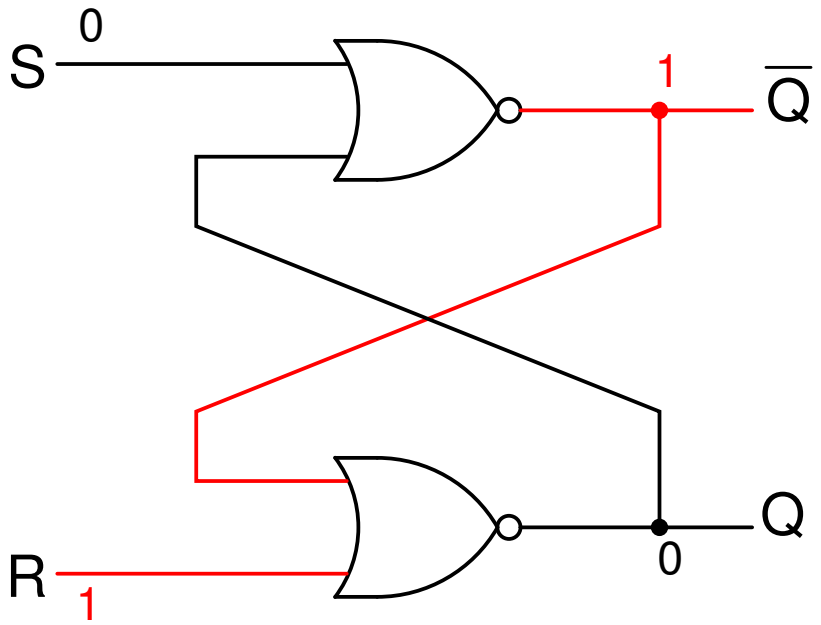
- Q=1, $\overline{Q}$=0
- Q=0, $\overline{Q}$=1

SR Latch



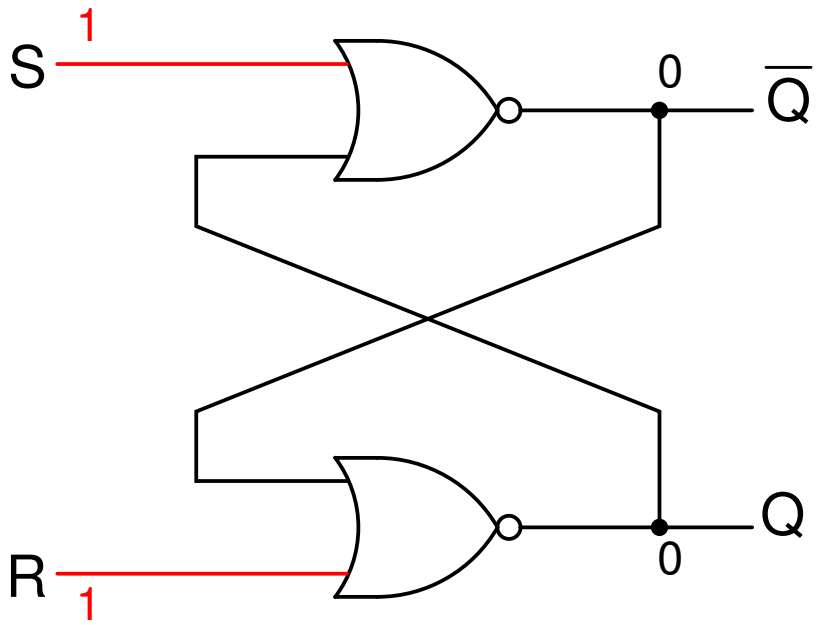
If S = 1 en R = 0, then we must have $\overline{Q} = 0$ en Q = 1.

SR Latch



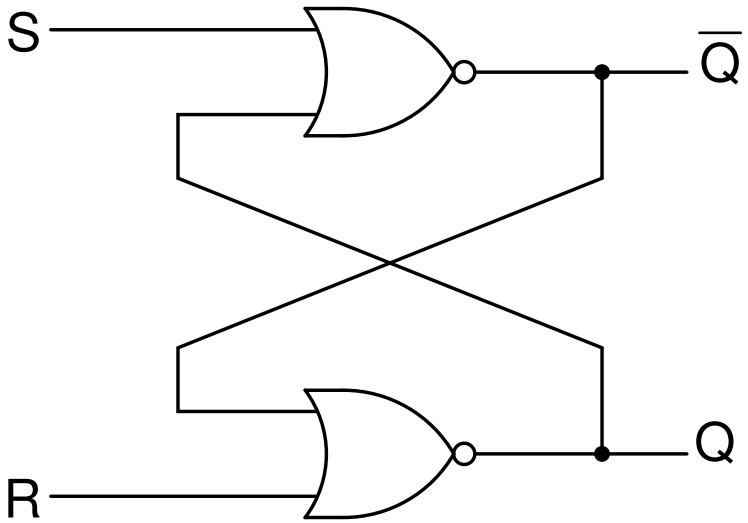
If S = 0 en R = 1, then we must have Q = 0 en $\overline{Q}$ = 1.

SR Latch



If S = R = 1, then we must have Q = $\overline{Q}$ = 0.
The latch can end up in either of the two stable states, depending on which input goes to 0 first.

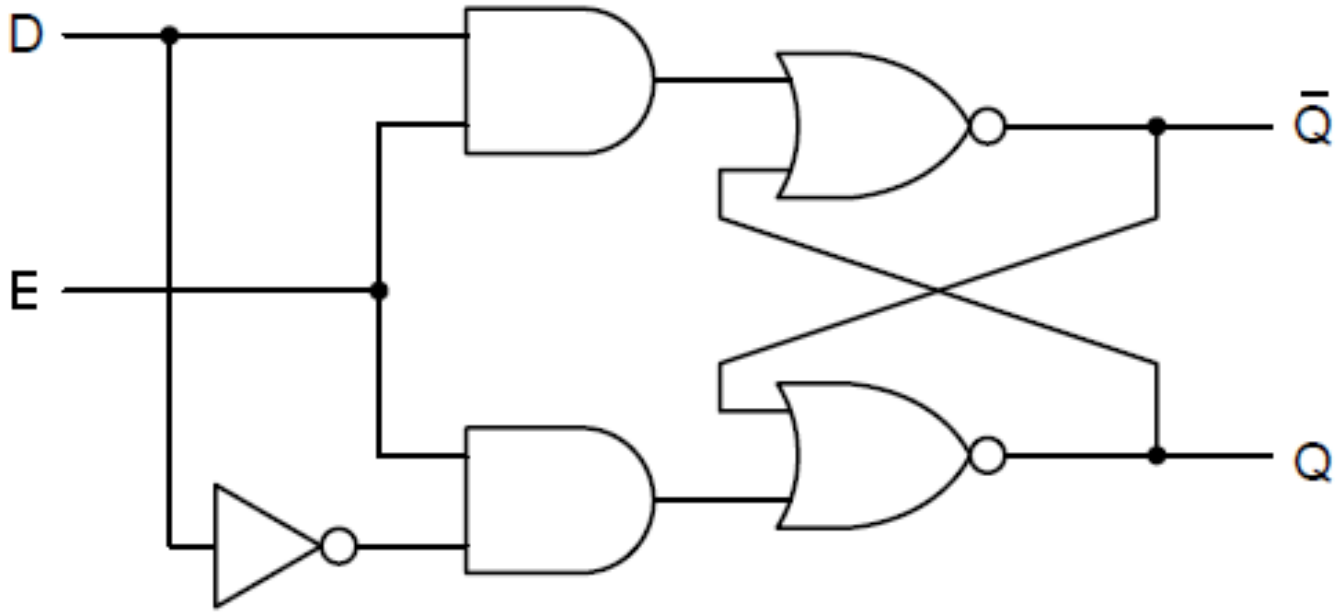
SR Latch – States



S	R	Q	$\overline{Q}$	Action
0	0	Q_{prev}	$\overline{Q}_{prev}$	Hold
1	0	1	0	Set
0	1	0	1	Reset
1	1	0	0	Forbidden

SR Latch is a one-bit memory element, with S = Set and R = Reset.

D Latch



- avoid the state with $S = R = 1$
- if $E = 0$ the latch maintains its value
- if $E = 1$ the value of D is stored in the latch

Synchronization

Sequential logic: asynchronous or synchronous

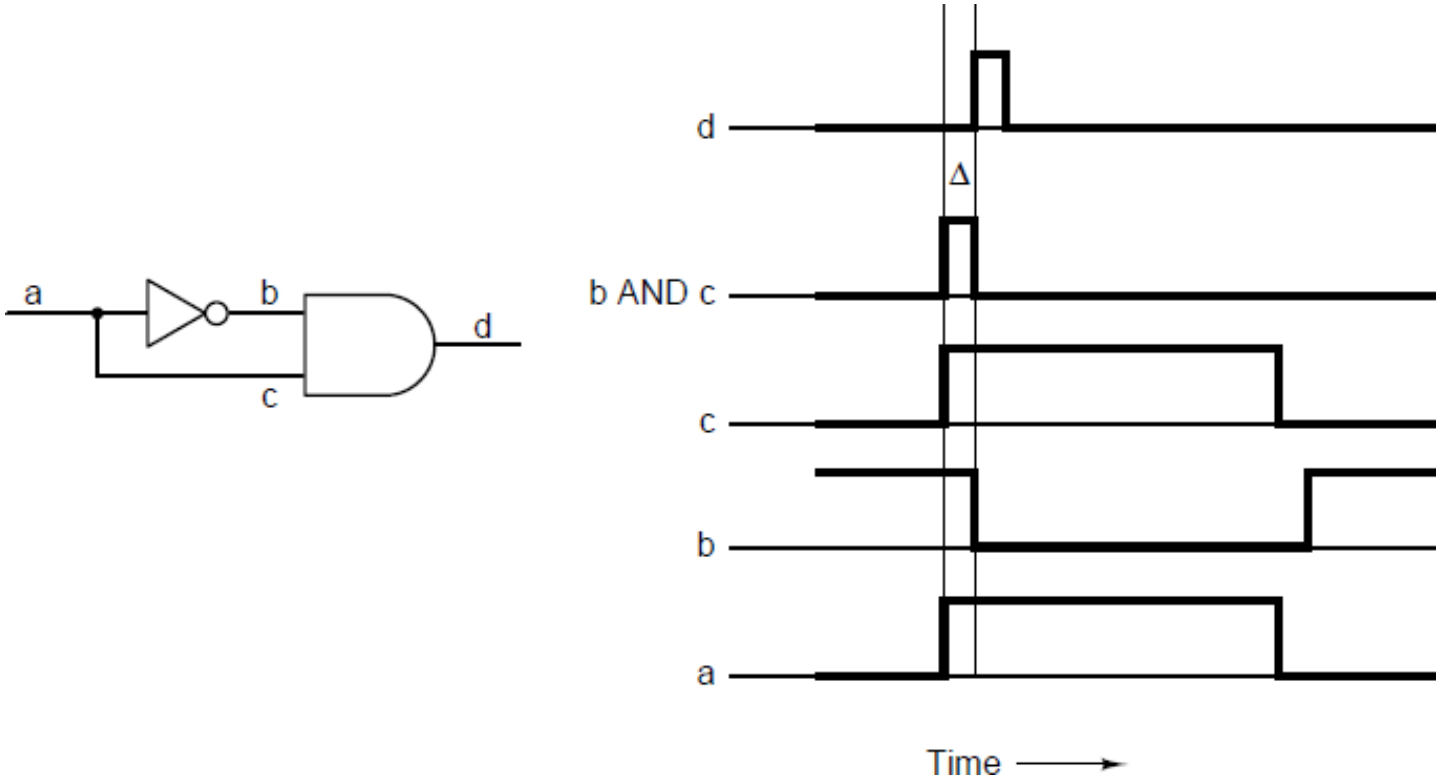
Asynchronous circuit:

- state changes immediately when input changes
- difficult to coordinate between multiple automata

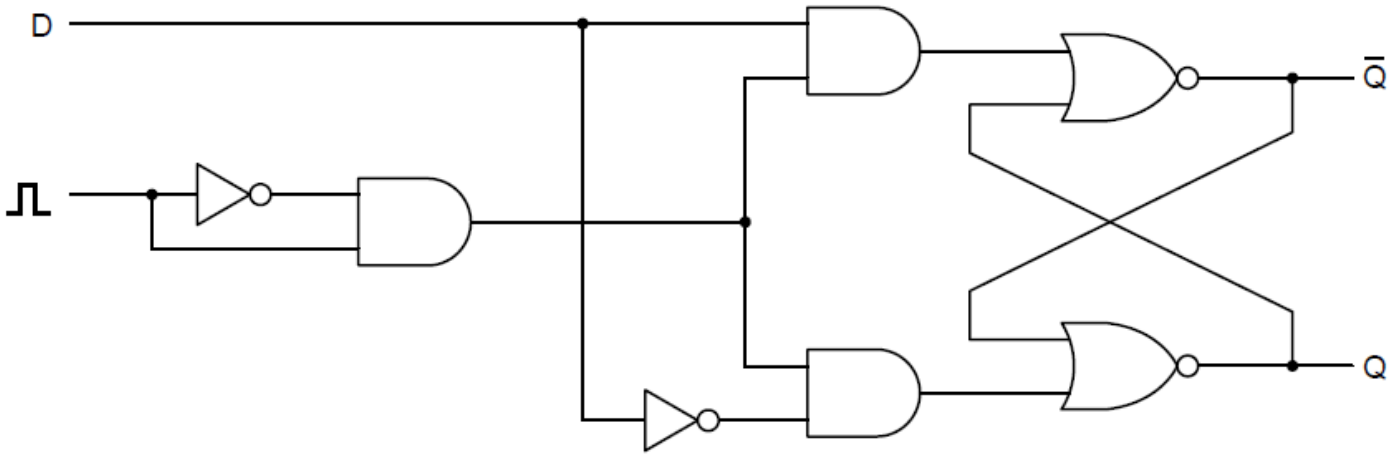
Synchronous circuit:

- state only changes when clock-input allows this
- on constant value of the clock: *level* triggered
  (“clocked D latch”, with a clock pulse on the E-input)
- on transition 0 → 1 (or 1 → 0): *edge* triggered
  (this is called a flip-flop)

Pulse Generator

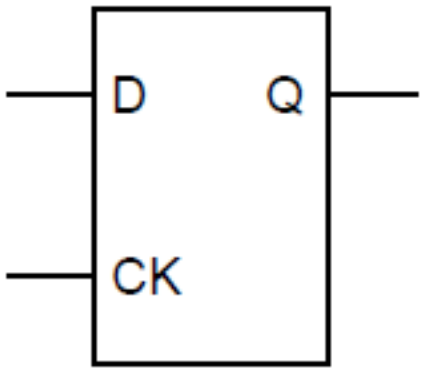


D Flip-flop

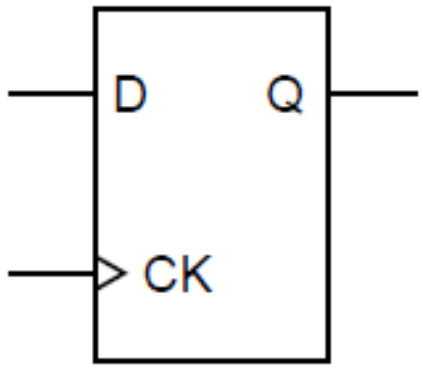


D flip-flop: D latch with pulse generator

Symbols for latch and flip-flop

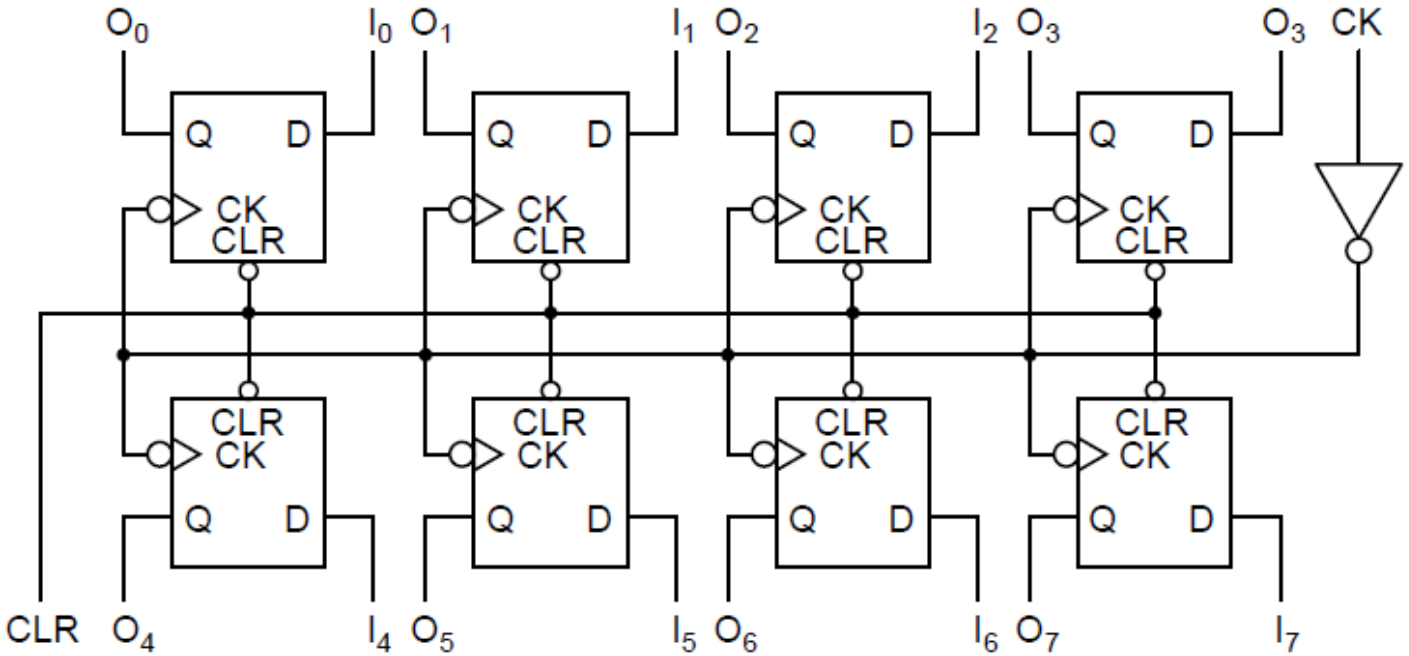


D latch
level triggered



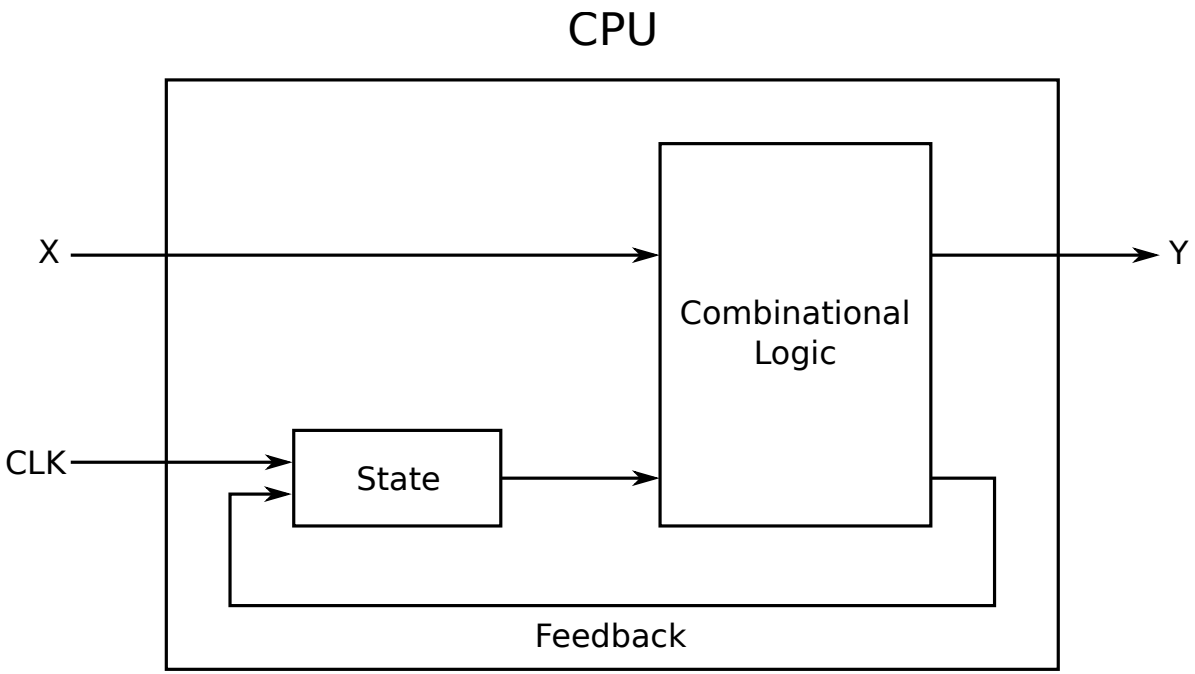
D flip-flop
edge triggered

Register



Register: a set of flip-flops with a shared clock, that store or retain an input word based on shared control signals.

Sequential Logic



- Synchronous sequential logic: implementation of finite automaton
- Processor is a set of connected finite automata, with combinational logic for computation

Summary

Numbers

- Binary arithmetic
- Two’s complement representation
- ALU

Memory

- SR-Latch: Feedback
- Latches and Flip-flops
- General form of sequential logic circuit