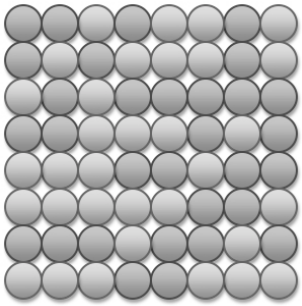
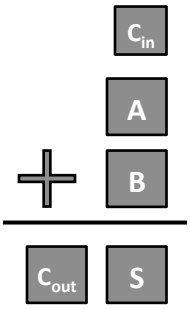


# UNIT 8B
## Computer Organization: Levels of Abstraction

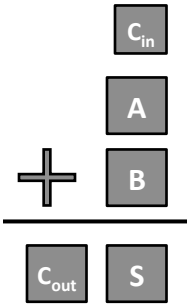


# A Full Adder




| A | B | $C_{in}$ | $C_{out}$ | S |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

# A Full Adder



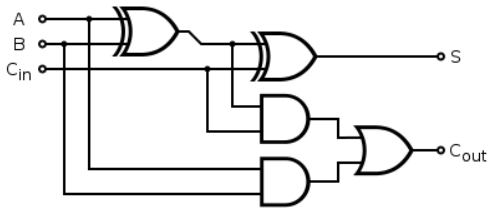

| A | B | $C_{in}$ | $C_{out}$ | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

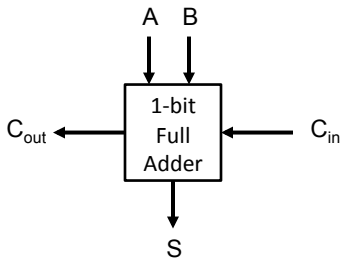
$S = A \oplus B \oplus C_{in}$

$C_{out} = ((A \oplus B) \wedge C) \vee (A \wedge B)$

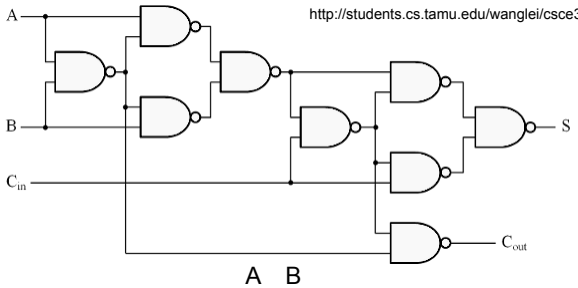


# Full Adder (FA)

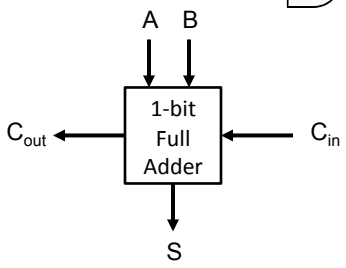


# Another Full Adder (FA)

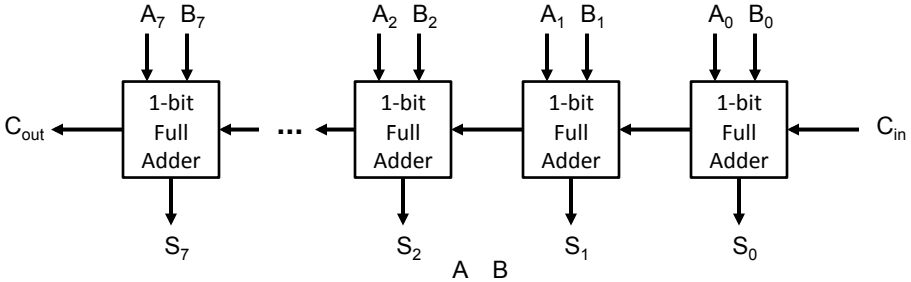


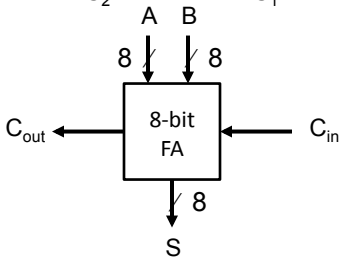

http://students.cs.tamu.edu/wanglei/csce350/handout/lab6.html





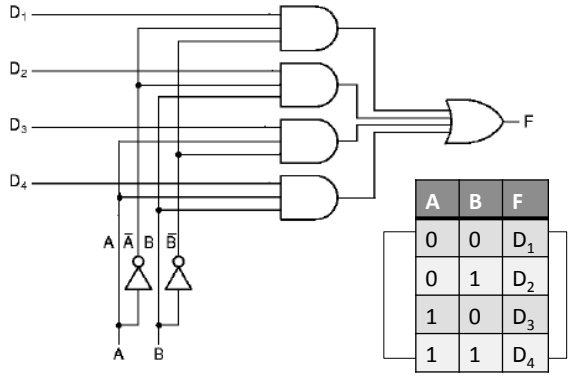

# 8-bit Full Adder



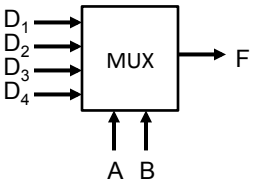



# Multiplexer (MUX)

- A multiplexer chooses between a set of inputs.



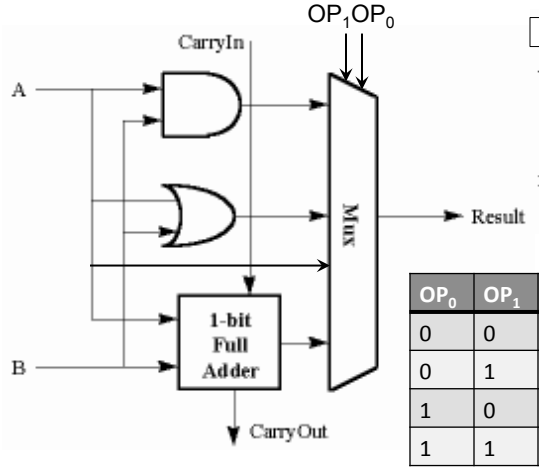

| A | B | F |
|---|---|---|
| 0 | 0 | $D_1$ |
| 0 | 1 | $D_2$ |
| 1 | 0 | $D_3$ |
| 1 | 1 | $D_4$ |

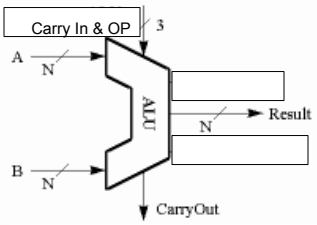



http://www.cise.ufl.edu/~mssz/CompOrg/CDAintro.html



# Arithmetic Logic Unit (ALU)



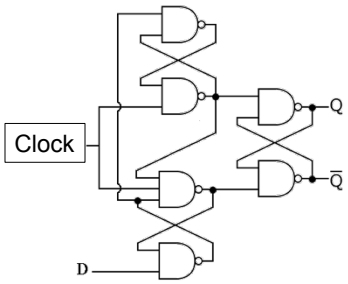



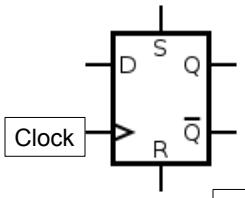

| $OP_0$ | $OP_1$ | F |
|---|---|---|
| 0 | 0 | A ∧ B |
| 0 | 1 | A ∨ B |
| 1 | 0 | A |
| 1 | 1 | A + B |

http://cs-alb-pc3.massey.ac.nz/notes/59304/l4.html

# Flip Flop

- A flip flop is a sequential circuit that is able to maintain (save) a state.
  - Example: D (Data) Flip-Flop – sets output Q to input D when clock turns on. (Images from Wikipedia)



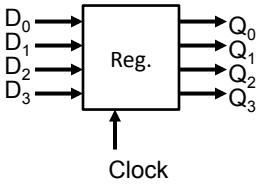



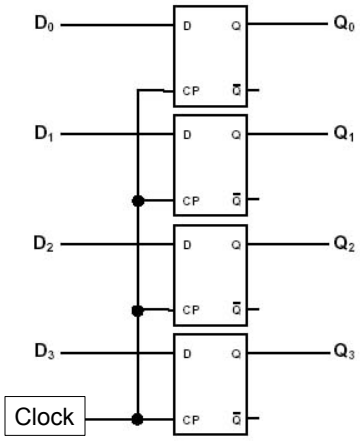

S=Set Q to 1,
R=Reset Q to 0



# Registers

- A register is just a set of edge-triggered flip-flops. Registers are triggered by a clock signal.



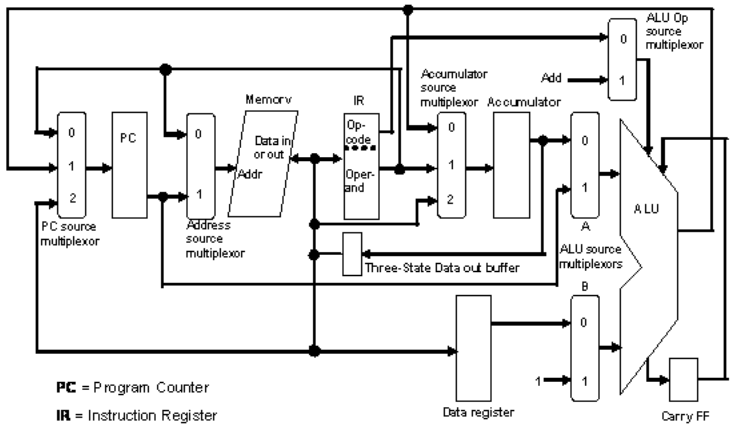

http://cpuville.com/register.htm

# Central Processing Unit (CPU)

- A CPU contains:
  - Arithmetic Logic Unit to perform computation
  - Registers to hold information
    - Instruction register (current instruction being executed)
    - Program counter (to hold location of next instruction in memory)
    - Accumulator (to hold computation result from ALU)
    - Data register(s) (to hold other important data for future use)
  - Control unit to regulate flow of information and operations that are performed at each instruction step

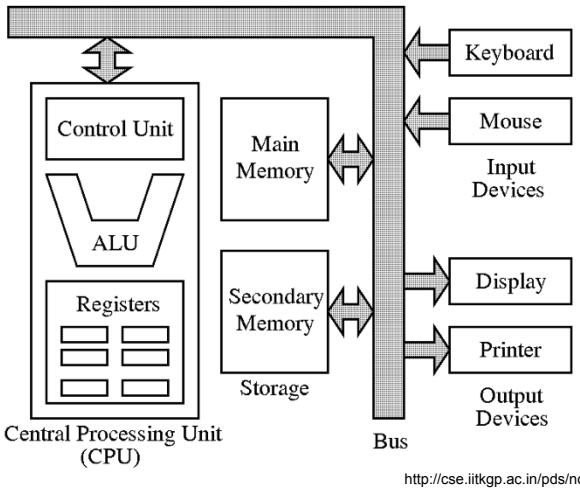


# A sample CPU




http://cpuville.com/main.htm

# Computer




http://cse.iitkgp.ac.in/pds/notes/intro.html



# Abstraction

- We can use layers of abstraction to hide details of the computer design.
- We can work in any layer, not needing to know how the lower layers work or how the current layer fits into the larger system.

  -> transistors
  -> gates
  -> circuits (adders, multiplexors, flip-flops)
  -> central processing units (ALU, registers, control)
  -> computer

# von Neumann Architecture

- Most computers follow the **fetch-decode-execute** cycle introduced by John von Neumann.
  - Fetch next instruction from memory.
  - Decode instruction and get any data it needs (possibly from memory).
  - Execute instruction with data and store results (possibly into memory).
  - Repeat.



# Programming a Machine

- All instructions for a program are stored in computer memory in binary, just like data.
- A program is needed that translates human readable instructions (e.g. in Ruby) into binary instructions ("machine language").
  - An interpreter is a program that translates one instruction at a time into machine language to be executed by the computer.
  - A compiler is a program that translates an entire program into machine language which is then executed by the computer.