CSC258: Computer Organization

Devices

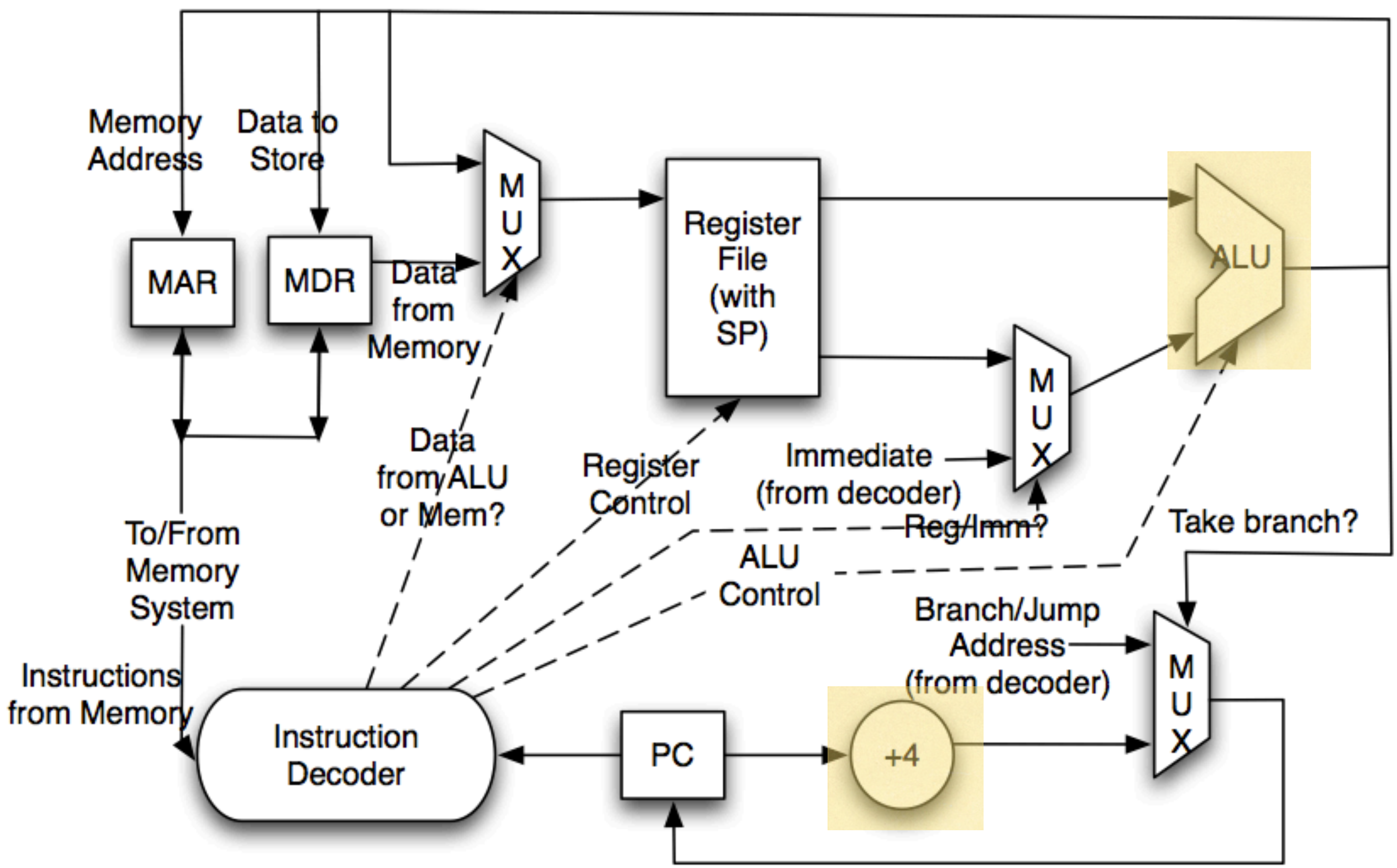
Your Mission …

- Generate the circuit that adds two one-bit numbers.

- Start with a truth table.
- Optimize the formula.
- Draw the circuit.

Where are we?

- Our goal, eventually, is to build a simple processor.
- For that, we need:
  - *The ability to execute simple operations (like "add")*
  - The ability to choose between values.
  - Storage for a small number of values (variables)
  - A circuit to process instructions from the user.


Memory Address
Data to Store
MAR
MDR
M U X
Data from Memory
Register File (with SP)
ALU
M U X
Data from ALU or Mem?
Register Control
Immediate (from decoder)
Reg/Imm?
Take branch?
ALU Control
To/From Memory System
Branch/Jump Address (from decoder)
M U X
Instructions from Memory
Instruction Decoder
PC
+4

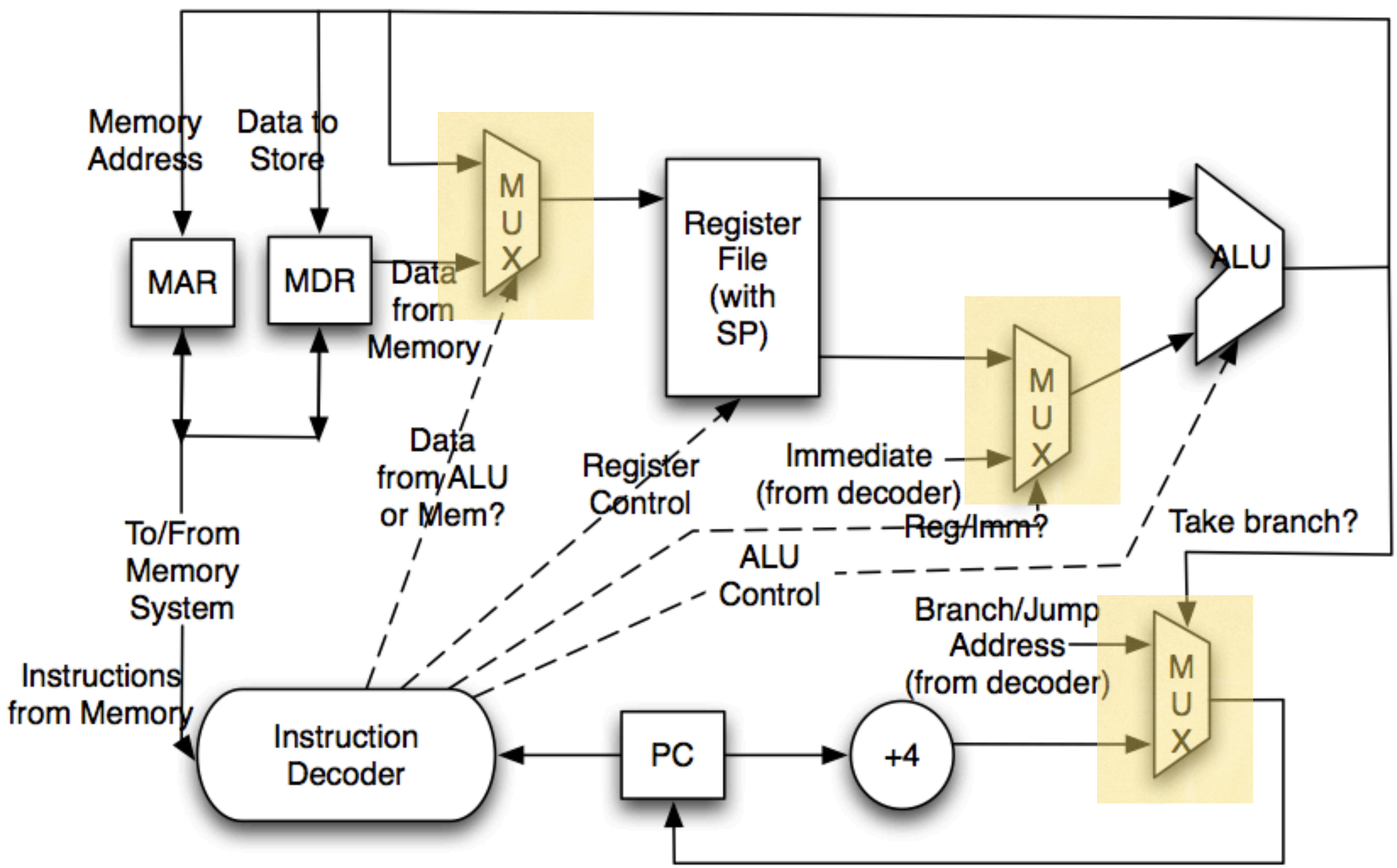
How do we get there?

- We can already do NAND -- and that's everything right?

- We'll look at shifters and adders in a couple of weeks, and we'll use those to create an ALU.

Where are we?

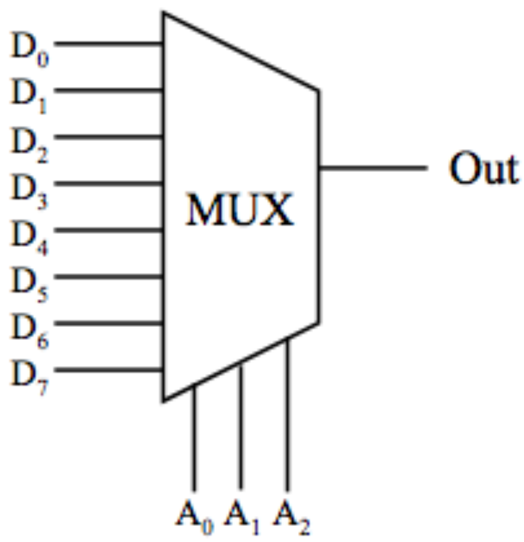
- Our goal, eventually, is to build a simple processor.
- For that, we need:
  - The ability to execute simple operations (like “add”)
  - *The ability to choose between values.*
  - Storage for a small number of values (variables)
  - A circuit to process instructions from the user.



Memory Address
Data to Store
MAR
MDR
MUX
Data from Memory
Register File (with SP)
ALU
MUX
Immediate (from decoder)
Reg/Imm?
Data from ALU or Mem?
Register Control
ALU Control
Take branch?
To/From Memory System
Instructions from Memory
Branch/Jump Address (from decoder)
MUX
Instruction Decoder
PC
+4

Multiplexer (Mux)

- Sometimes, we need “select” or “choose”
- A mux takes multiple inputs and a selector input
  - The selector chooses which input to output



A_1	A_0	Out
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

... and the Reverse

- Since we "mux", we should be able to "demux"
  - Take one input and a selector, and produce the input on one of the outputs

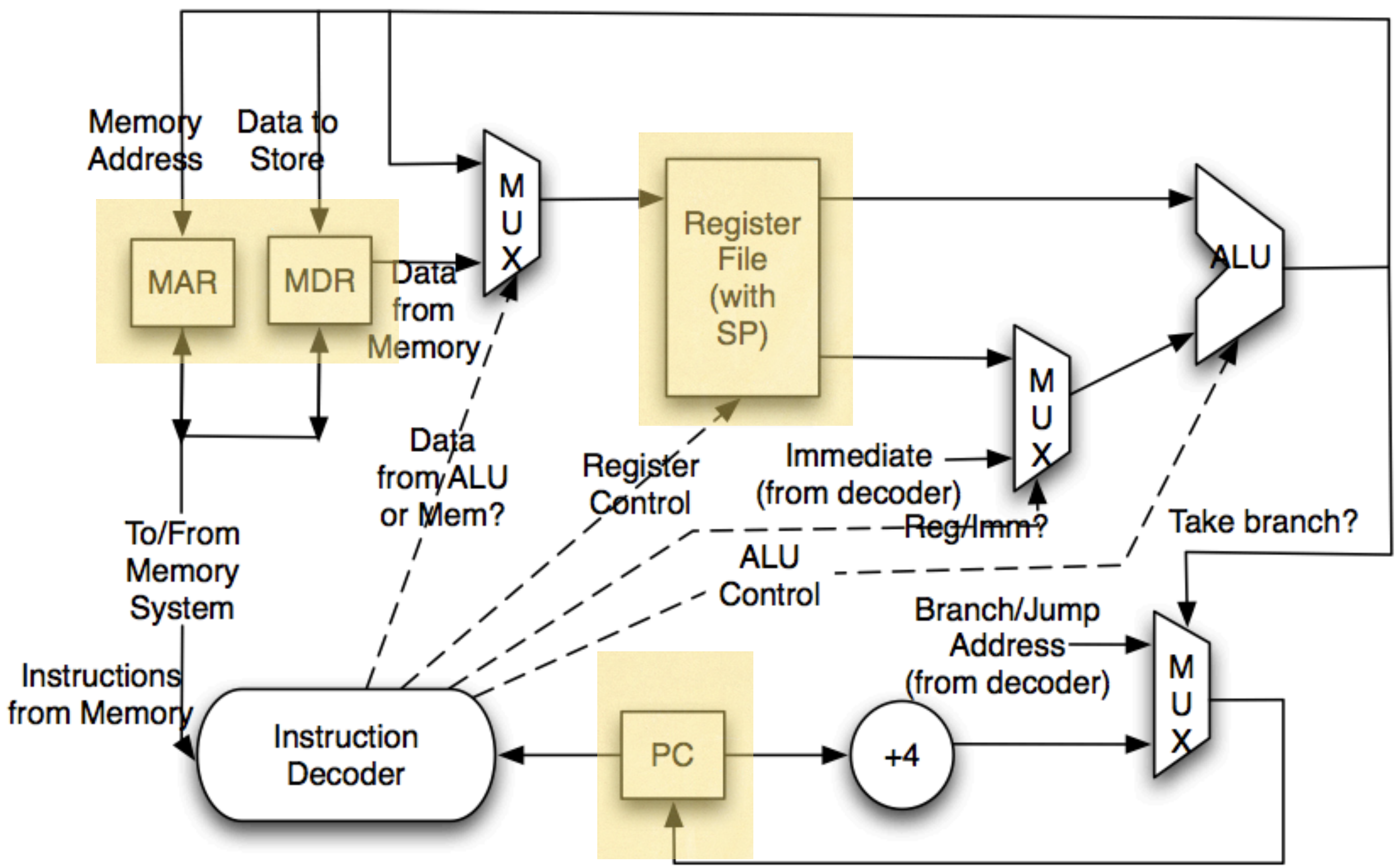
- The text also covered encoders and decoders.
  - An encoder takes 2^N inputs and produces a N-bit (compacted) output
  - A decoder takes an N-bit input and produces 2^N (extracted) outputs

Where are we?

- Our goal, eventually, is to build a simple processor.
- For that, we need:
  - The ability to execute simple operations (like “add”)
  - The ability to choose between values.
  - *Storage for a small number of values (variables)*
  - A circuit to process instructions from the user.

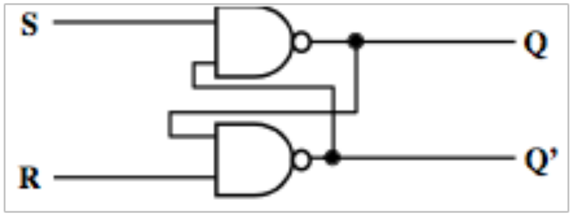
How do we get there?

- We need some device that can store data.
- We also need some concept of “time”, so we know when to store new data.
- We’ll start putting together these pieces today.

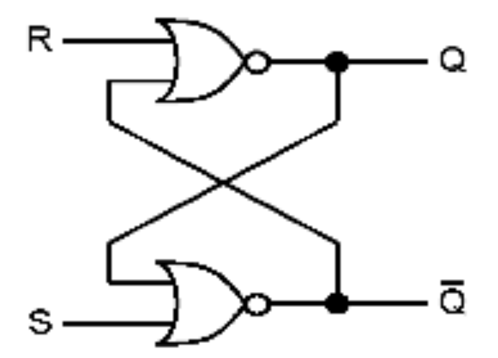

Memory Address
Data to Store
MAR
MDR
M U X
Data from Memory
Register File (with SP)
ALU
M U X
Immediate (from decoder)
Reg/Imm?
Data from ALU or Mem?
Register Control
ALU Control
Take branch?
To/From Memory System
Instructions from Memory
Instruction Decoder
PC
+4
Branch/Jump Address (from decoder)
M U X

Storing Data

- Our circuits so far provide an output given some inputs



- The other half of the equation is storage
  - Use feedback to “freeze” a value
- These are SR latches
  - They rely on feedback.

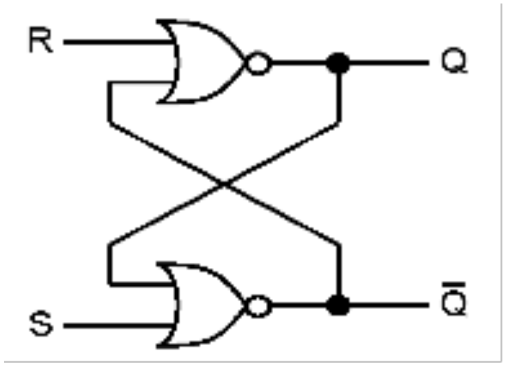


Your Mission …

- Provide the truth table and the circuit for an SR latch.

SR Latches

- R is labeled “reset”
- S is labeled “set”



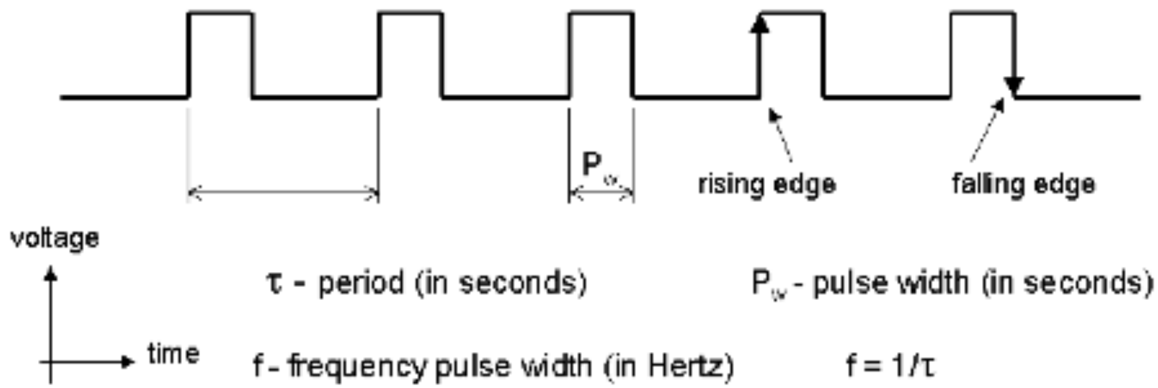
- When R is high, Q becomes 0
- When S is high, Q becomes 1
- Otherwise, Q keeps its previous value.

R	S	Q	$\neg$Q
0	0	Q	$\neg$Q
0	1	1	0
1	0	0	1
1	1	?	?

Your Mission …

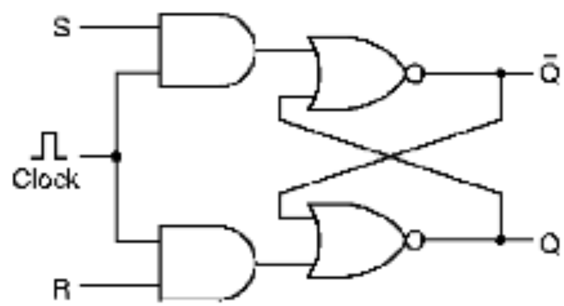
- Add a clock input to your latch truth table.
- Think about what you want the clock input to mean: when does the device trigger?

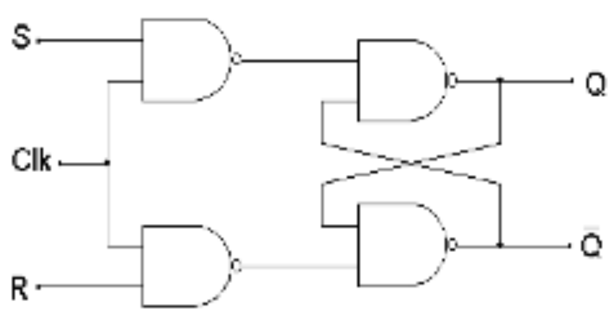
Adding Timing to Circuits



- Idea: We need to hold a value long enough for it to stabilize and to use it for computation
- Solution: Add a "clock" that tells all of the latches in the circuit to change their value
  - Pulse width must be large enough to let values stabilize
  - Period must be long enough to let computation finish

Gated Latches





Cl	S	R	Q	¬Q
0	x	x	Q	¬Q
1	0	0	Q	¬Q
1	0	1	0	1
1	1	0	1	0
1	1	1	?	?

Characteristic Equation

- Every gated-latch can be defined as a function based on time
  - This is called its *characteristic equation.*
- For the SR gated latch:
  $Q(t + 1) = S(t) + \neg R(t)Q(t)$

S	R	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	?

11 Transition Problems

- For an SR latch, the "11" state is unstable
  - Both Q and Q' are 0
  - A transition to the "00" state yields undefined outputs
- Solution: Eliminate the "11" state

D Latch

- Instead of two inputs, the D latch has one
  - D stands for “data”
- Easily made from an SR latch
  - How?

D	Q	Q’
0	0	1
1	1	0

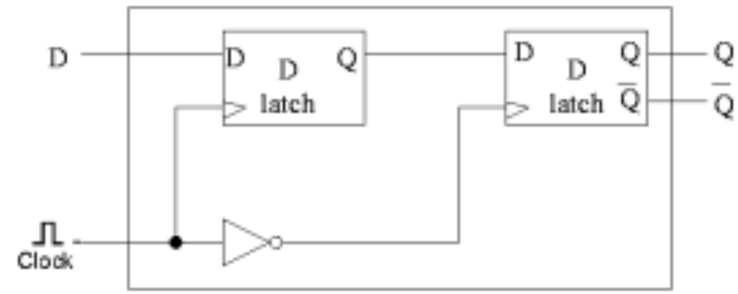
Transparency

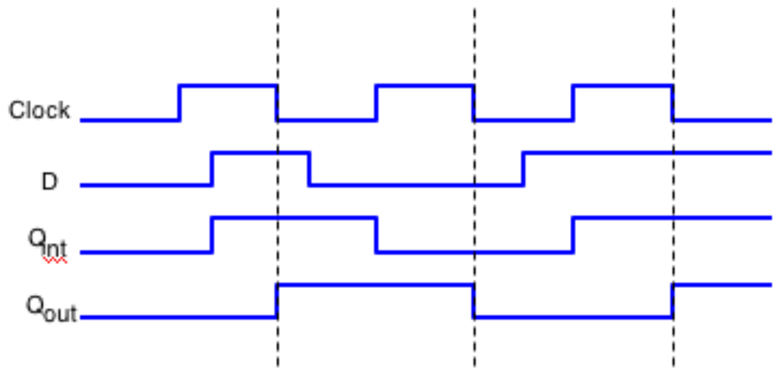
- All latches are "transparent"
- When the input changes, the output immediately changes (within some propagation delay)

- Problem: What happens if an input changes in the middle of a clock cycle?
  - This happens often! Many circuits use latches that feed into other latches
  - If the cycle is longer than the propagation delay, problems occur

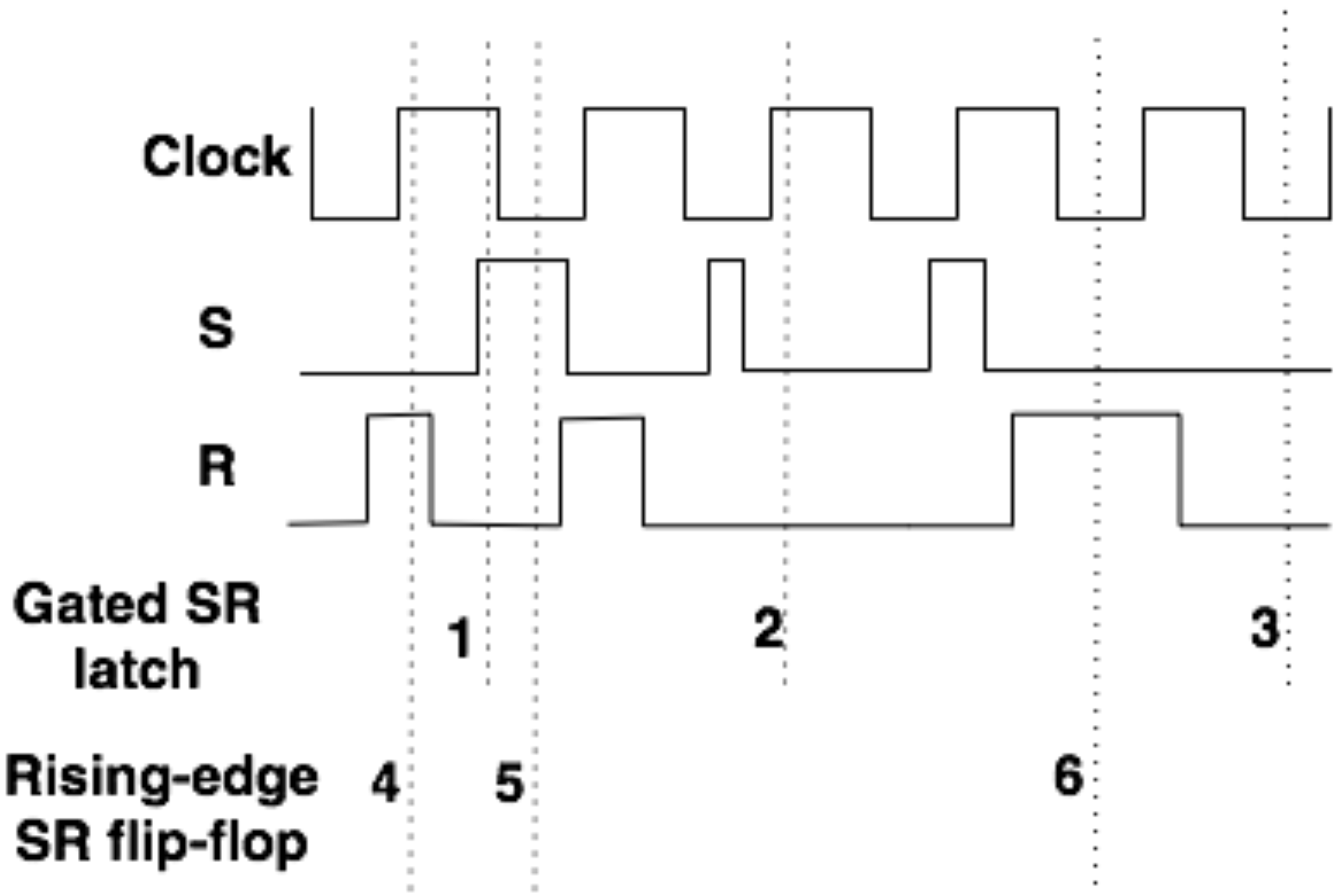
Master-Slave Flip-Flop

- Output of the flip-flop is delayed to break a circuit into a *sequence* of circuits operating in parallel.
- Prevents value from being propagated early





Latches and Waveforms



Solution