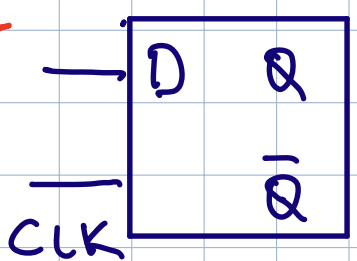
Edge-Triggered Clocking

Any value stored in machine are updated only on clock edge

State elements update internal storage on clock edge.

→ either rising or falling edge is active & cause state to be changed.

Latch

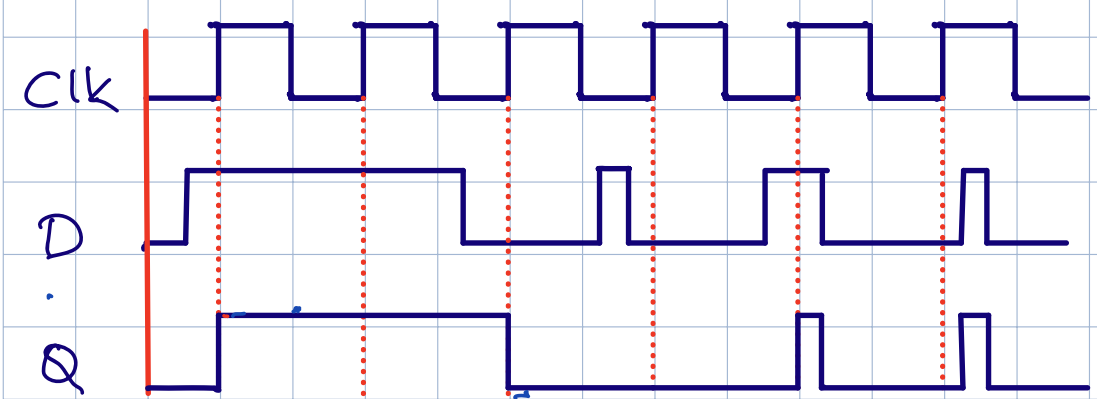


Clk ON: D → Q

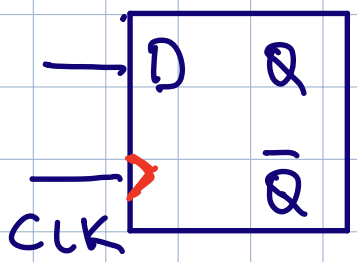
Clk OFF:

prev value → Q

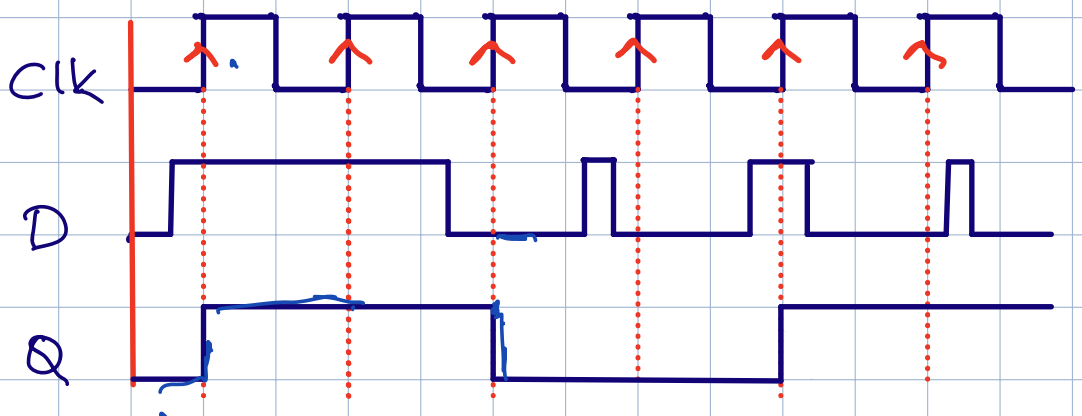
(Positive level sensitive latch)



D-Flip Flop (2 latches) (Depend on edges)



(Example for positive edge)



Instructions

lw, sw, add, sub, and, or, slt, beq, j

lw, sw: Memory

add, sub, and, or, slt: Arithmetic/Logical

beq, j: Branch/j

first 2 steps are identical (for every instr.)

1. Send PC to memory that contain the code
   Fetch instr. from memory
2. Read registers (one or 2 registers)

Action depends on instr class
→ use ALU after ready register

* If memory-reference ⇒ address calculation
* Arithetic-logical instr ⇒ operation execution
* Branch instr ⇒ Comparison (Sub)

After ALU, action required differs
→ from one instr. to another.

- Memory Reference → Access Memory
- Arithetic/Logical → Write data from ALU to register file
- Branch instr → Change next instr add (if true)

Register File

Reading = not require changing a state.
Only register number is supplied

Writing = It requires register number, data to write and clock controls writing into register.

- Do not show a write control signal when a state element is written on every clock edge.
  - If it is not updated on every clock, than a separate control signal

R-format (What is reqred?)

{Register file
{ALU

Lw/Sw Instructions (what is reqed?)

- Register File
- ALU
- Sign-extend unit
- Data Memory unit

Data Memory needs read signal but Register file doesn't.

Why? Unlike register file, reading a value of an invalid address can cause problems.

`lw $s2, 20($s3)`

read reg : $s3

$s2 ← Memory [20 + $s3]

`sw $s5, 40($s6)`

read regs: $s6, $s5

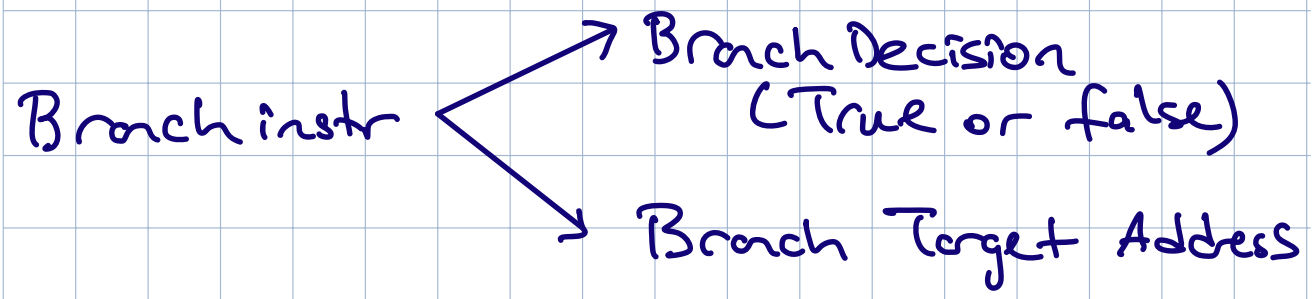
Memory [$s6 + 40] ← $s5

LW = Register file + Memory Addr calculation + Read from memory + Write into register

SW = Register file + Memory Addr Calculation + Write into memory location

Branch Instr. (what is required)

- Register file
- ALU (Compare for equality)
- Sign-extend
- Shift left (2-bits)
- Adder



Address → 16 bits

16 bits to 32 bits → Sign-extended

byte addr to word addr → Shift left by 2 bits

Reading Assignment

* pages B.26 – B.38 (from appendix)
* pages 244 – 259

(In order to revise digital design material you can read B1 – B20)