# Lecture:

- Last Time:
  - Control Lines and the CPU
    - On-Off Switch
    - Decoder to select which register.
    - Mux to select ALU Output
    - CPU Overview: Data with some control lines.
    - High-level memory schematic
  - Questions

- Today:
  - Sequential Circuits: clocks, latches, and flip/flops:
  - CPU Control: PC/IR
  - MIPS pipeline architecture

- Next: Onto Assembly Language Program:
  - Assembly Level Programming
    - Three-address code
    - MIPS assembly code

Sequential Circuit

- Sequential circuit infuses a memory component
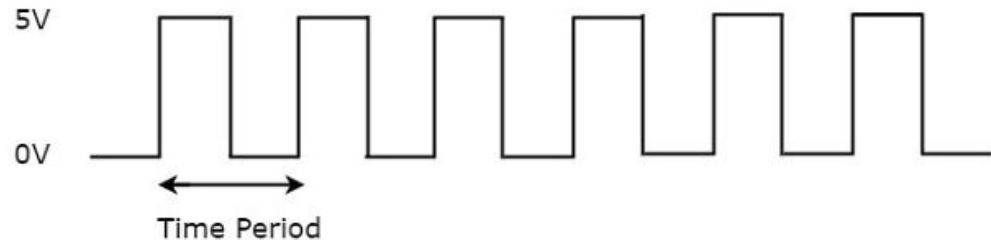- Memory serves as both input and output
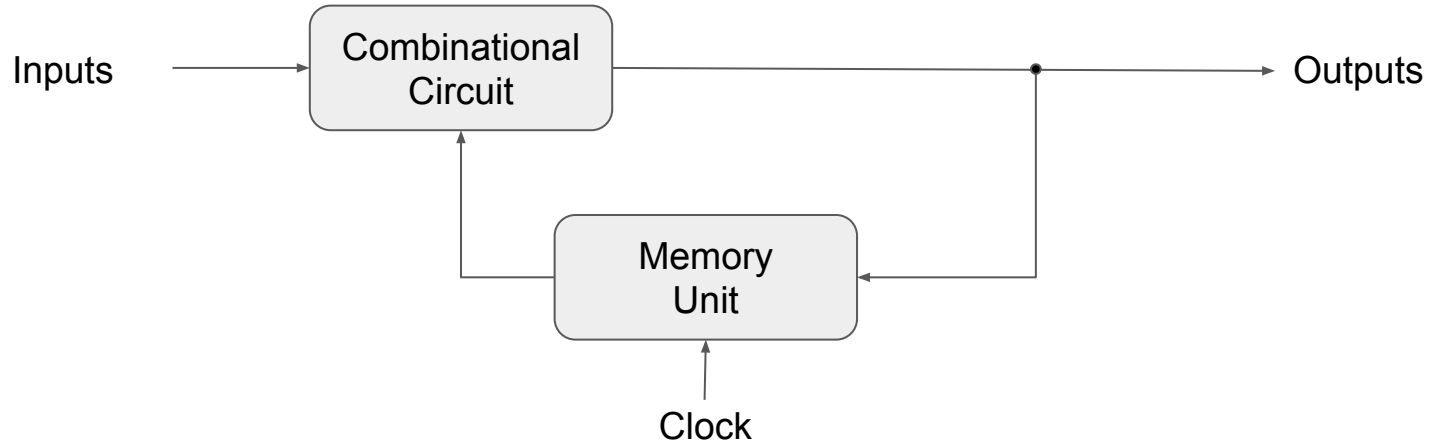
- A clock is used to trigger the update of the memory unit

Clock: an electronic device

- Generates a periodic signal
- Forms a square wave
- Frequence = 1 / (Time Period)
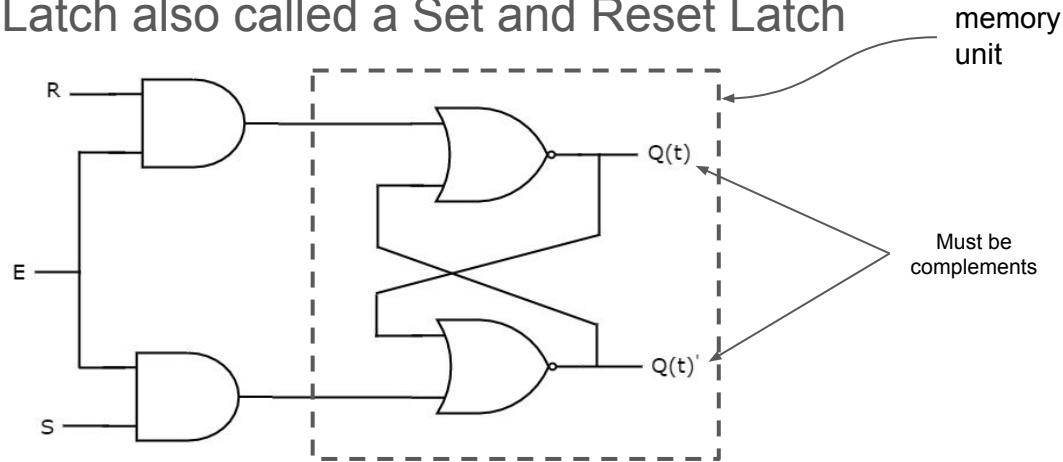- Used as a control signal to active parts of a circuit

- Types of Triggers:[*]
  - Level Trigger
  - Edge Trigger
- Level of Triggers:
  - Positive Level (Hi)
  - Negative Level (Lo)

```
5V ___     ___     ___     ___     ___     ___
   |   |   |   |   |   |   |   |   |   |   |   |
0V_|   |___|   |___|   |___|   |___|   |___|   |___
     <----->
    Time Period
```

* Recall you can active some GUI elements on a mouse click, mouse press, **OR** a mouse release

SR Latch

- SR Latch also called a Set and Reset Latch
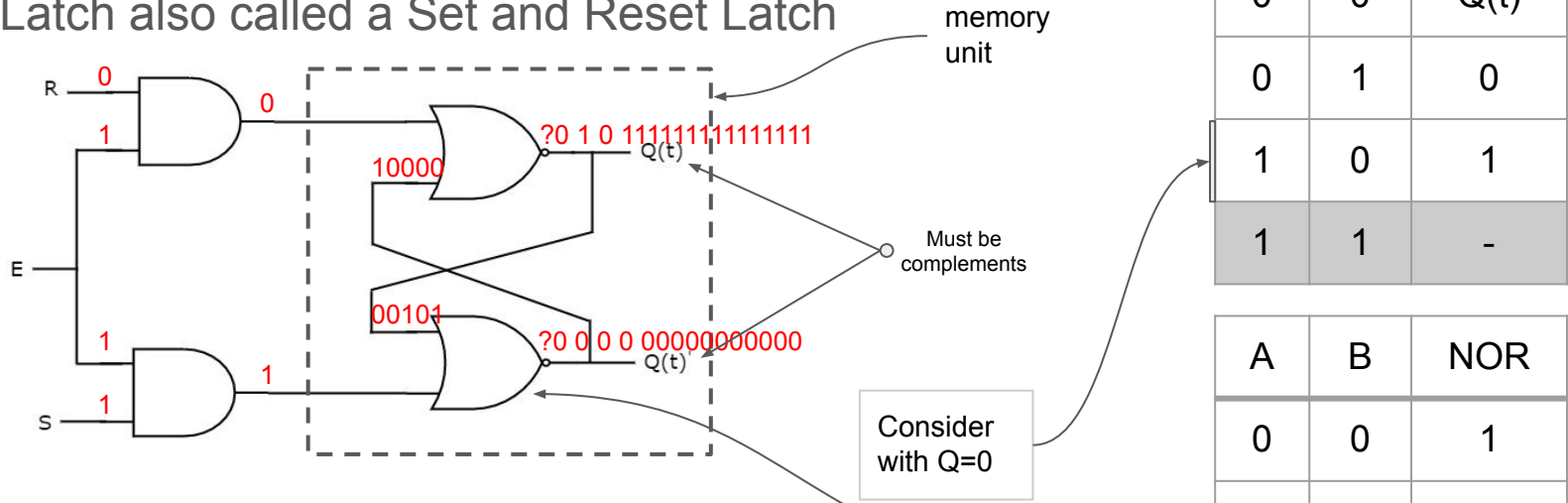

memory unit

Must be complements

- Functionally:
  - E: Used to enabled the circuit: i.e., trigger an update
  - R: Used to clear (reset) the memory unit
  - S: Used to set the memory unit

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

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

SR Latch: Set at (1,0)

- SR Latch also called a Set and Reset Latch

memory unit

R 0

0

1

10000

?0 1 0 111111111111111 Q(t)

E

00101

1

1

?0 0 0 0 00000000000 Q(t)

S 1

Must be complements
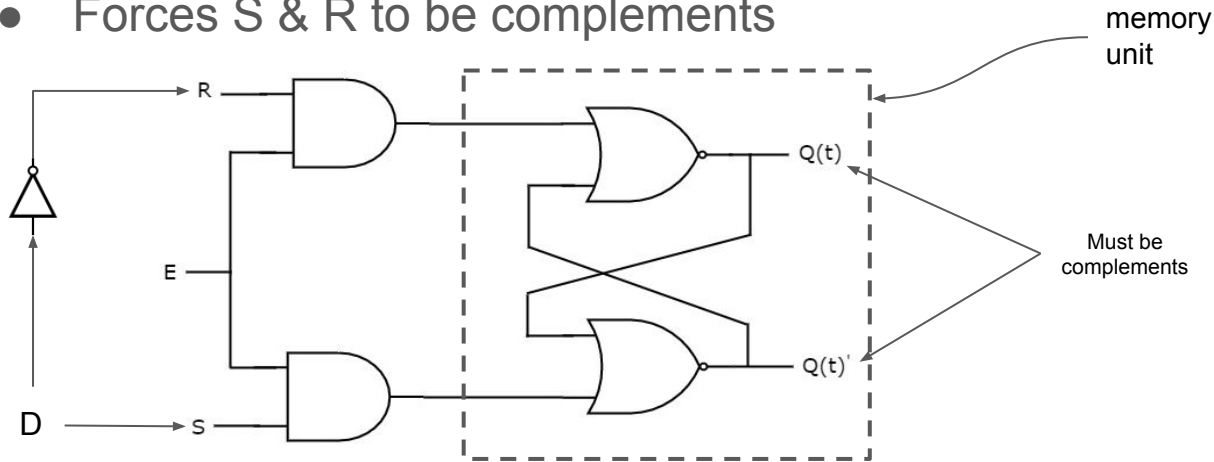
Consider with Q=0

- Functionally:
  - E: Used to enabled the circuit: i.e., trigger an update
  - R: Used to clear (reset) the memory unit
  - S: Used to set the memory unit

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

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

D Latch: Data Latch

- Forces S & R to be complements
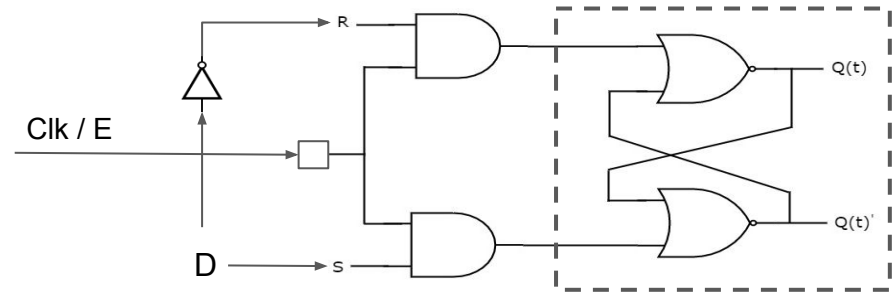

- Functionally:
  - E: Used to trigger the memory unit, i.e., perform a write
  - D':   Used to clear (reset) the memory unit
  - D: Used to set the memory unit

E	D	Q(t+1)
0	0	Q(t)
0	1	Q(t)
1	0	0
1	1	1

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Latches and Flip-Flops

Logically the same!

● Latches:
  ○ RS Latch
  ○ D Latch
  ○ JK Latch
  ○ T Latch

● Flip-Flop:
  ○ RS Flip-Flop
  ○ D Flip-Flop
  ○ JK Flip-Flop
  ○ T Flip-Flop

Differences:

Latches	Flip-flops
Different underlying technology is used	
Uses a enable signal	Uses a clock signal
Level triggered	Edge triggered
Uses a enable signal	Uses a clock signal
Consumes less power	Consumes more power
Faster	Slower
Used for transitioning (Consider a lock, e.g., at the Panama Canal.)	Used as storage

4-bit Register

- D Flip-Flop
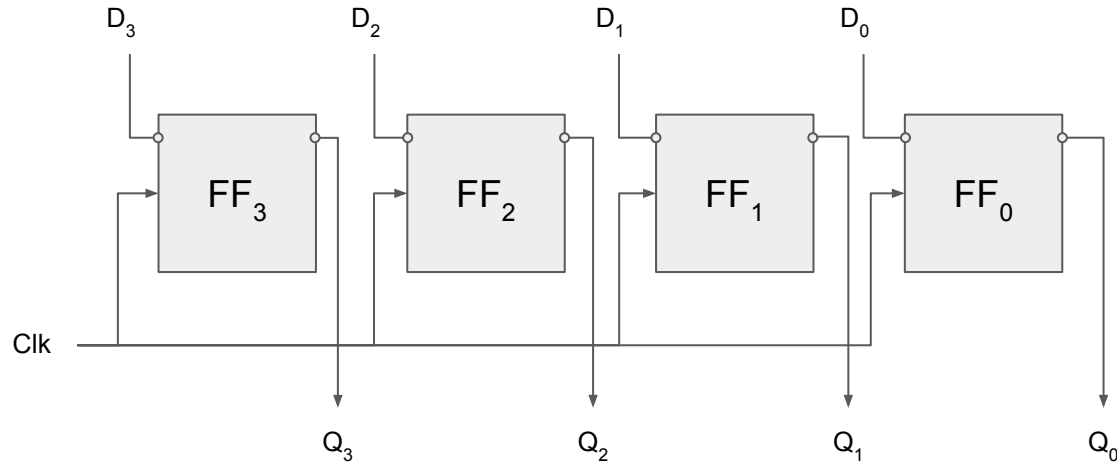
What do we have now?

1. Basic understanding of circuitry and its relationship to Boolean algebra.
2. Knowledge of "state" and how it is used to build registers and memory units
3. Recognize how we can build components to build a CPU
   - ALU, registers, decoders, multiplexers, etc., as well as main memory.
4. Almost ready for the abstract all this away in favor of an ISA
   Instruction Set Architecture

But first, let's put all the pieces together to create the MIPS microarchitecture!

MIPS: CPU Control

```
int f( int a) {
    int x;
    int y;
    int b;


    x = a + 2;
    b = ( x > 0 )
    if ( b )
        x = x + b;
    y = x * 3;
    y = y << 2 ;

    return y;
}
```

- The CPU executes specific circuitry in stepwise fashion
    - Fetch, Decode, Execute, Mem Access, WriteBack
    - Unnamed Latches are used between each phase
- PC: Program Counter
    - Holds the address of the NEXT instruction to be executed
    - The PC is increment by 1 instruction, i.e., 4 bytes, every cycle
    - Each instruction requires 4 bytes to encode
- IR: Instruction Register
    - Stores the current instruction to be executed
    - Each instruction is decoded (electronically) to trigger operations in the circuitry
    - Three types of encodings: R, I, J
        - Operations involving the **R**egisters and ALU
        - Operations involving **I**mmediate values
        - Operations involving **J**umps and Branches

MIPS Instruction Set

	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
R	op	rs	rt	rd	sh	func

	6 bits	5 bits	5 bits	16 bits		
I	op	rs	rt	imm		

	6 bits	26 bits				
J	op	addr				

- Jump and Branch examples:
  - jal *address* ⇔ function call
  - jr rs ⇔ function return
  - beq rs, rt, *imm* ⇔ if (rs == rt) goto *imm*
- ALU examples:
  - add rd, rs, rt ⇔ rd = rs + rt   → ADD $r0, $r2, $r3
  - ori rd, rs, *imm* ⇔ rd = rs | *imm*   → ORI $r0, $r2, 5
- Memory examples:
  - la rs, *label* ⇔   → $rs = &A
  - lb rt, imm(rs) ⇔ rt = SignExt(M_1 [rs + *imm*])   → $rt = (char) A[0]
  - lwu rt, imm(rs) ⇔ rt = M_4 [rs + *imm*]   → $rt = (unsigned word) (A[16])
  - sh rt, imm(rs) ⇔ M_2 [rs + *imm*] = rt   → A[32] = (half) $rt

MIPS Pipeline Architecture

- Fives Stages →
- Major Sections:
  - PC update
  - ALU
  - Memory
- PC update
  - next instruction
  - branch
  - (PC + 4) / ALU mux
- ALU
  - imm / register mux
  - register / PC mux
- Memory
  - Harvard Model
  - ALU / memory mux
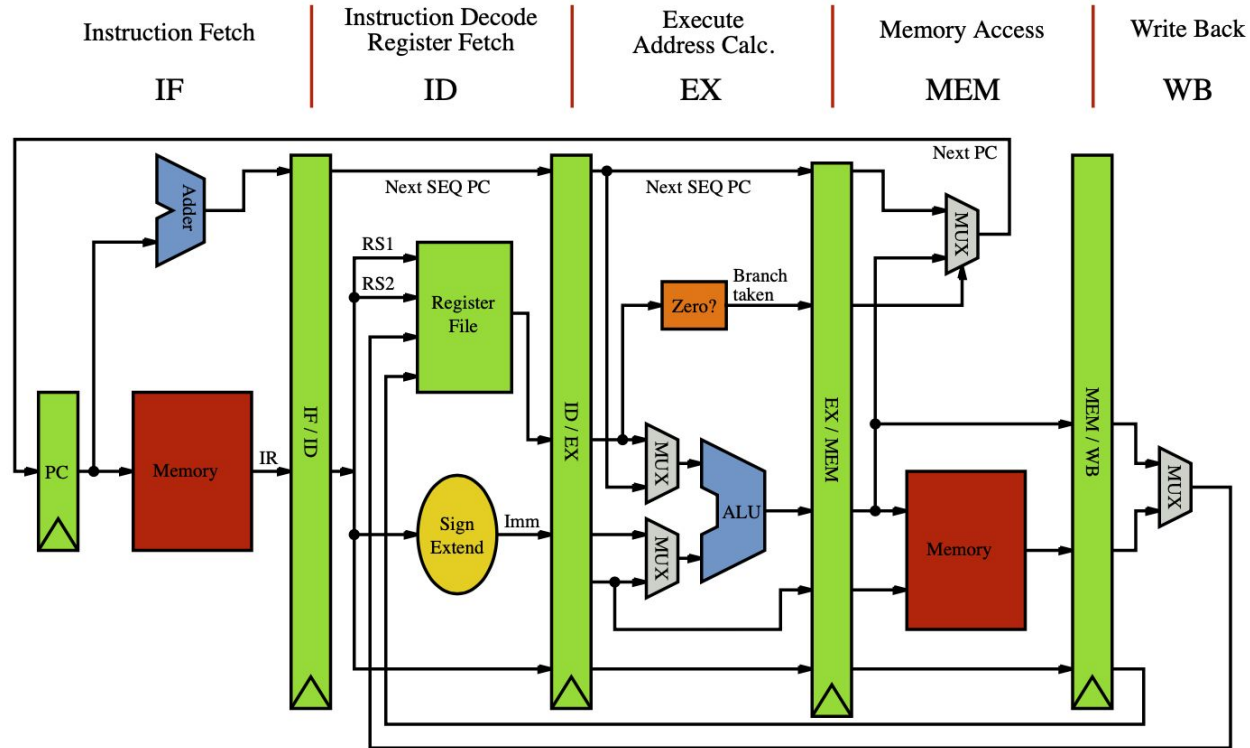
Instruction Fetch — IF
Instruction Decode Register Fetch — ID
Execute Address Calc. — EX
Memory Access — MEM
Write Back — WB