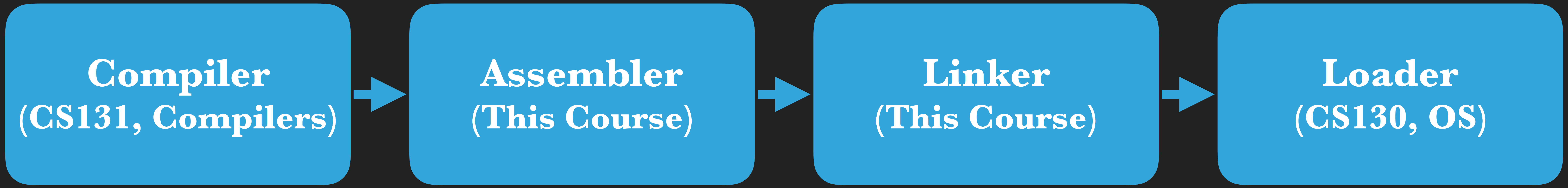
DISCUSSION 6: CALL & SDS

WEIQI WU

WHAT IS CALL?

- C: Compiler
- A: Assembler
- L: Linker
- L: Loader

CALL PIPELINE



COMPILER

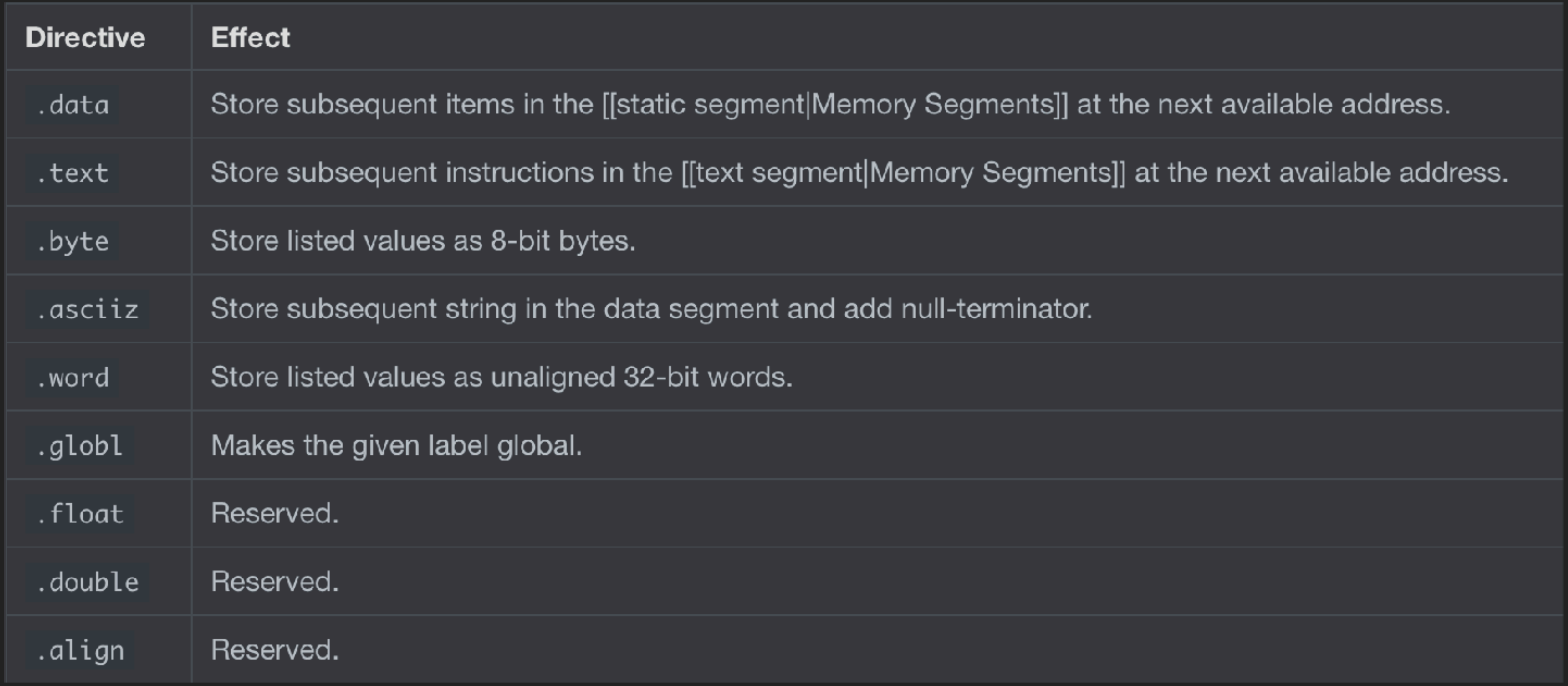
- Translates high-level programming language into assembly
- Input: foo.c, foo.cpp, etc.
- Output: foo.s
- Pipeline: Lexer, Parser, Semantic Analysis + Optimization, Code generation
- Many compilers for a programming language: GCC C, Turbo C, Quick C, etc.

ASSEMBLER

- Translates assembly codes to machine codes
- Input: foo.s
- Output: foo.o
- Read and Uses **Directives**
- Expand pseudo-instructions into basic one (look up your RISC-V Green Card)
- Produce Machine language & object file

DIRECTIVES

- Give directions to assembler, but do not produce machine instructions



Directive	Effect
`.data`	Store subsequent items in the [[static segment|Memory Segments]] at the next available address.
`.text`	Store subsequent instructions in the [[text segment|Memory Segments]] at the next available address.
`.byte`	Store listed values as 8-bit bytes.
`.asciiz`	Store subsequent string in the data segment and add null-terminator.
`.word`	Store listed values as unaligned 32-bit words.
`.globl`	Makes the given label global.
`.float`	Reserved.
`.double`	Reserved.
`.align`	Reserved.

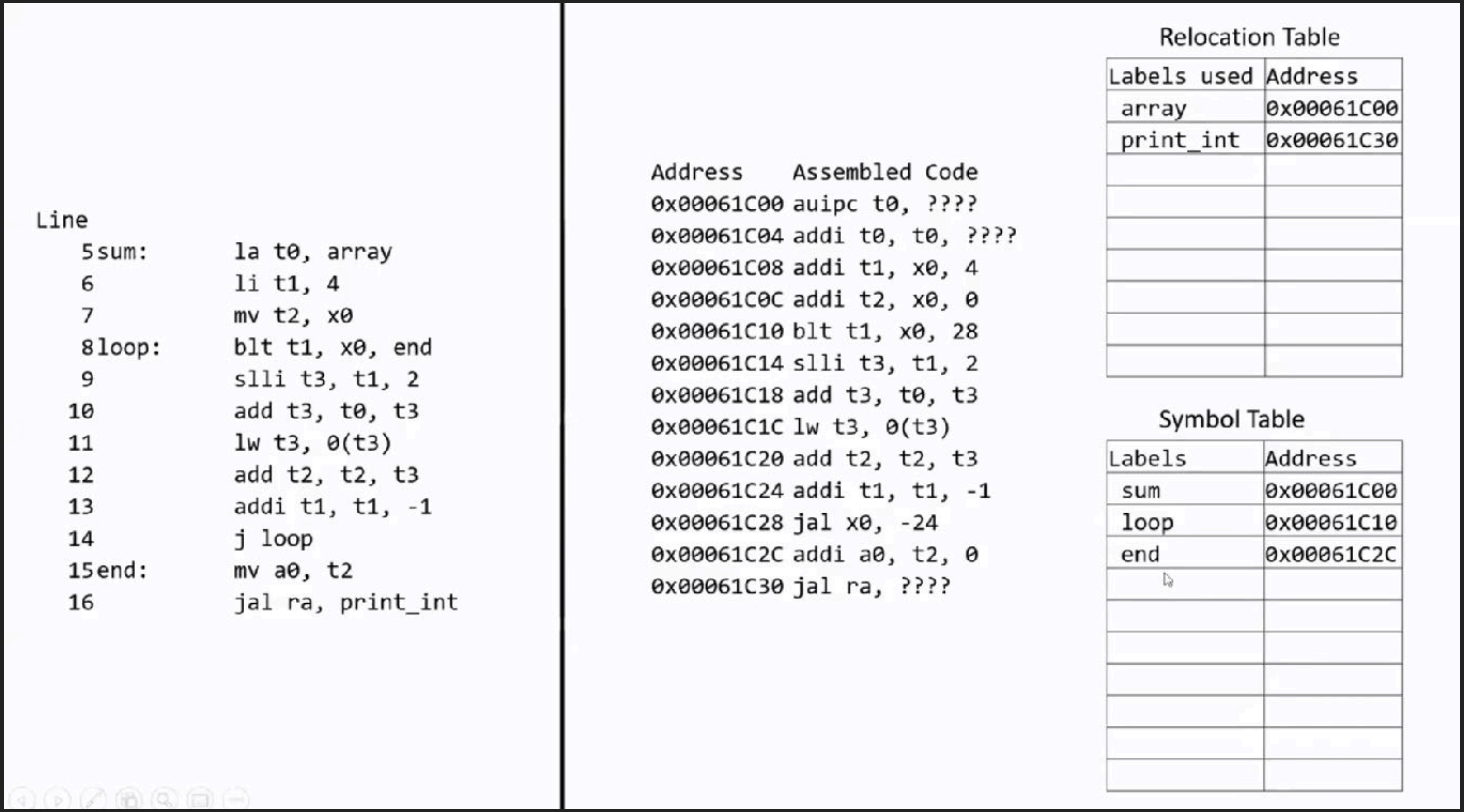
SYMBOL TABLE

- List of file labels and data that can be referenced across files
- Key: Label, Value: information about label (which section, offset within that section…)
- Contains:
  1. Labels: function calling
  2. Data: .data segment, vars which may accessed across files

RELOCATION TABLE

- Identify lines of code that need Linker to handle the address
- Contains:
  1. Any external label jumped to: jal, jalr
  2. Any piece of data in static section: la

EXAMPLE



```
Line
   5 sum:      la t0, array
   6           li t1, 4
   7           mv t2, x0
   8 loop:     blt t1, x0, end
   9           slli t3, t1, 2
  10           add t3, t0, t3
  11           lw t3, 0(t3)
  12           add t2, t2, t3
  13           addi t1, t1, -1
  14           j loop
  15 end:      mv a0, t2
  16           jal ra, print_int
```

```
Address    Assembled Code
0x00061C00 auipc t0, ????
0x00061C04 addi t0, t0, ????
0x00061C08 addi t1, x0, 4
0x00061C0C addi t2, x0, 0
0x00061C10 blt t1, x0, 28
0x00061C14 slli t3, t1, 2
0x00061C18 add t3, t0, t3
0x00061C1C lw t3, 0(t3)
0x00061C20 add t2, t2, t3
0x00061C24 addi t1, t1, -1
0x00061C28 jal x0, -24
0x00061C2C addi a0, t2, 0
0x00061C30 jal ra, ????
```

Relocation Table

Labels used	Address
array	0x00061C00
print_int	0x00061C30

Symbol Table

Labels	Address
sum	0x00061C00
loop	0x00061C10
end	0x00061C2C

LINKER

- Input: foo.o, libc.o
- Output: foo.out
- Pipeline:
  - Take text segment from .o file and put them together
  - Take data segment from .o file, put them together, and concatenate this onto end of text segments
  - Resolve references, fill in all **absolute** addresses

LOADER

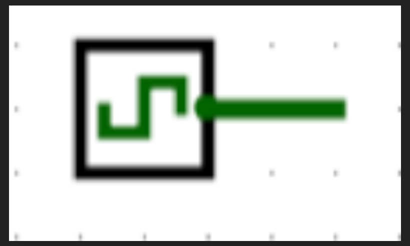
- Input: foo.out (exec file on disk)
- Output: (program load into memory & running)
- Operating system task

LOADER… WHAT DOES IT DO?

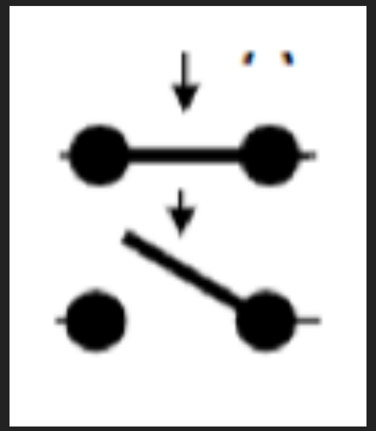
- Reads exec file’s header to determine size of text and data segments
- Creates new address space (text, data, stack segments) for program
- Copies instructions and data from exec file into new address space
- Copies arguments passed to the program onto stack
- Initialize machine registers, sp -> 1st free stack location
- Jump to start-up routine that copies arguments from stack to register & sets the PC

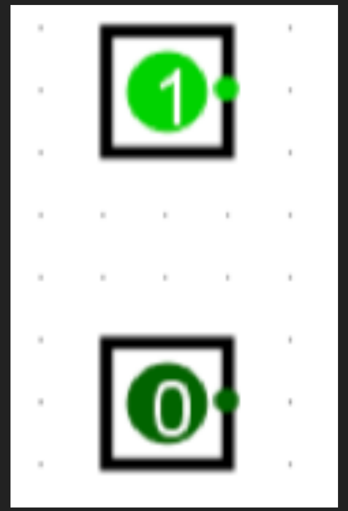
WHAT IS SDS?

- S: synchronous
  - All operations coordinated by a central clock



- D: digital
  - 1/0 Representation, discrete values
  - High Voltage(Vdd) = True = On switch = 1
  - Low Voltage(0V) = False = Off switch = 0

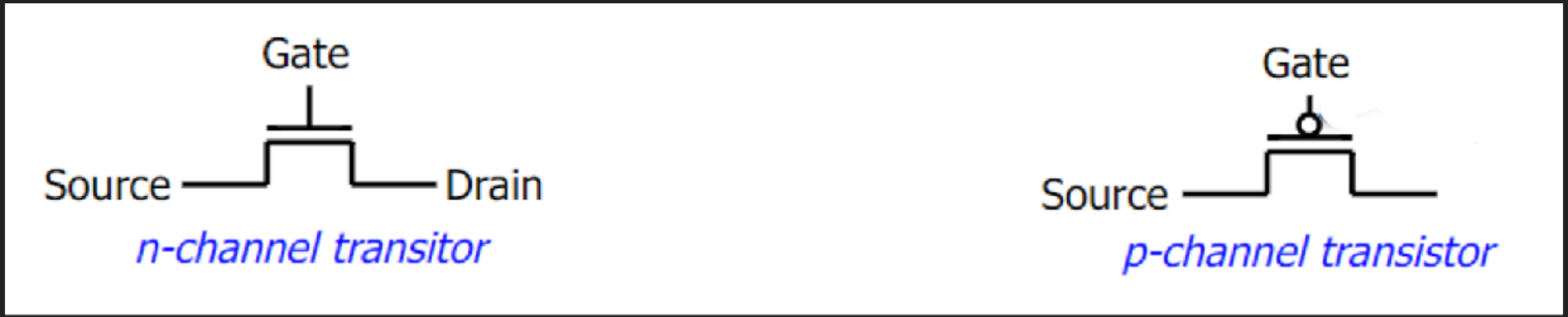




- S: system
  - Combinational Logic: output is a function of the inputs only
  - Sequential Logic: circuits that store history information

CMOS TRANSISTOR

- Act as voltage-controlled switches



Negative

Off: Voltage at Gate is low

On: V(Gate) > V(Threshold)

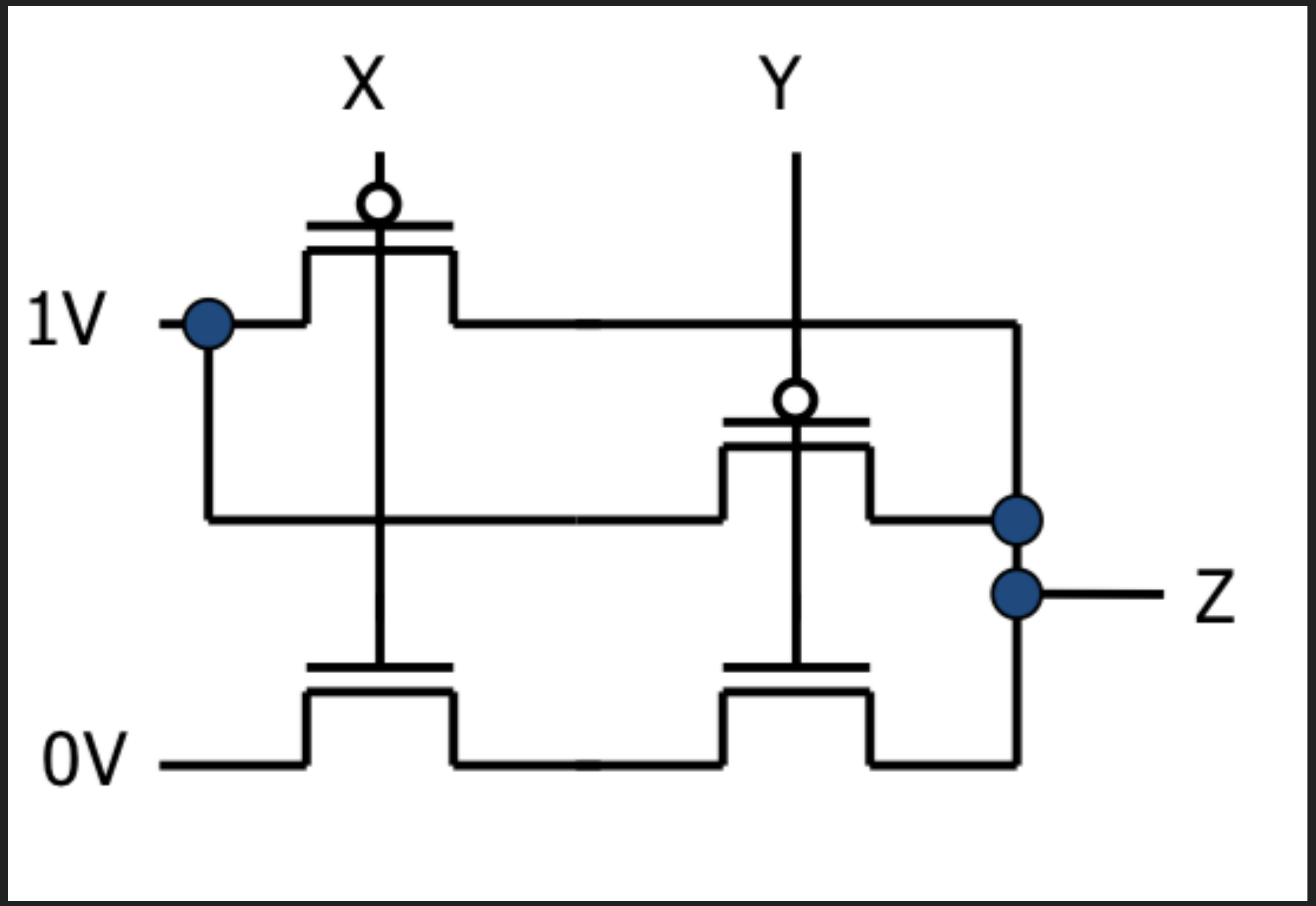
Positive

On: Voltage at Gate is low

Off: V(Gate) > V(Threshold)

CMOS NETWORKS

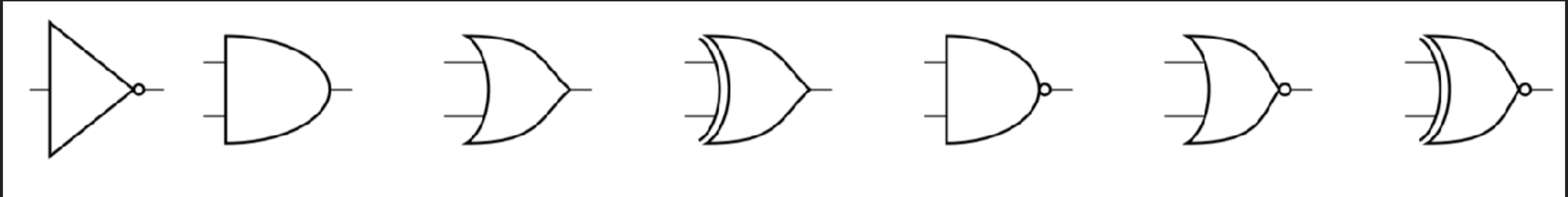
- A two-input network example (which is a NAND gate)



X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

COMBINATIONAL LOGIC

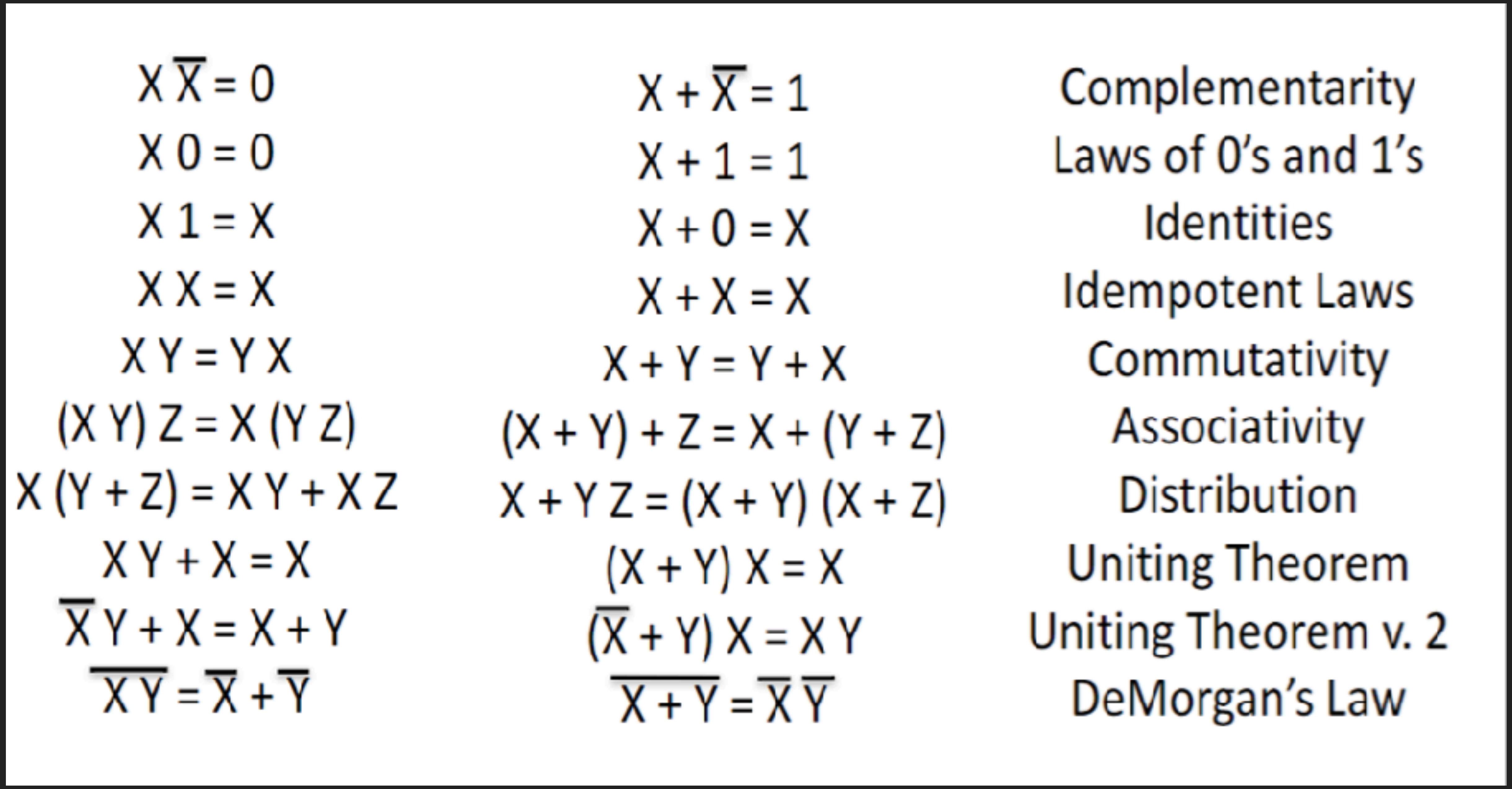
- Logic gates


NOT, AND, OR, XOR, NAND, NOR, XNOR

- AND: A · B / AB
- OR: A + B
- NOT: $\bar{A}$
- XOR: A ⊕ B

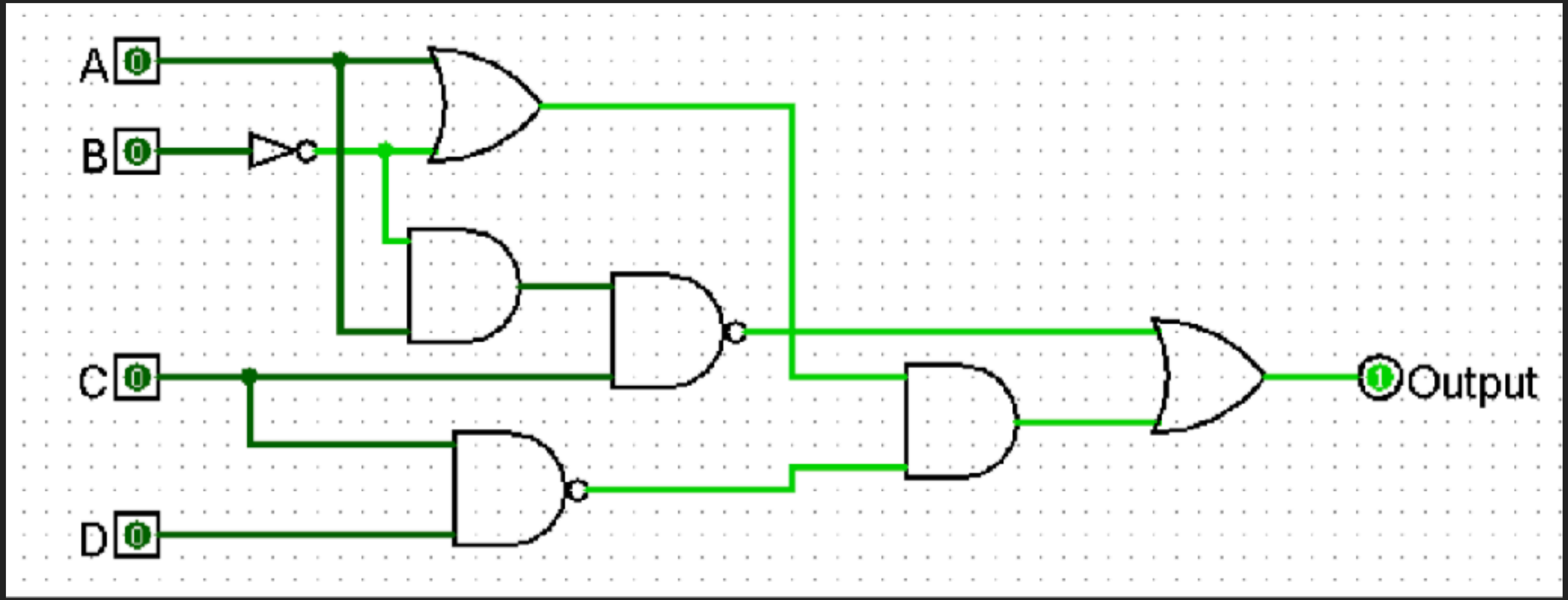
TRUTH TABLE & BOOLEAN ALGEBRA

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0



$X\,\overline{X} = 0$	$X + \overline{X} = 1$	Complementarity
$X\,0 = 0$	$X + 1 = 1$	Laws of 0's and 1's
$X\,1 = X$	$X + 0 = X$	Identities
$X\,X = X$	$X + X = X$	Idempotent Laws
$X\,Y = Y\,X$	$X + Y = Y + X$	Commutativity
$(X\,Y)\,Z = X\,(Y\,Z)$	$(X + Y) + Z = X + (Y + Z)$	Associativity
$X\,(Y + Z) = X\,Y + X\,Z$	$X + Y\,Z = (X + Y)\,(X + Z)$	Distribution
$X\,Y + X = X$	$(X + Y)\,X = X$	Uniting Theorem
$\overline{X}\,Y + X = X + Y$	$(\overline{X} + Y)\,X = X\,Y$	Uniting Theorem v. 2
$\overline{X\,Y} = \overline{X} + \overline{Y}$	$\overline{X + Y} = \overline{X}\,\overline{Y}$	DeMorgan's Law

CIRCUIT & BOOLEAN EXPRESSION



$$\overline{A\overline{B}C} + (A+\overline{B})(\overline{CD})$$

$$= \overline{A} + B + \overline{C} + (A+\overline{B})(\overline{C}+\overline{D})$$

$$= \overline{A} + B + \overline{C} + A\overline{C} + A\overline{D} + \overline{B}\,\overline{C} + \overline{B}\,\overline{D}$$

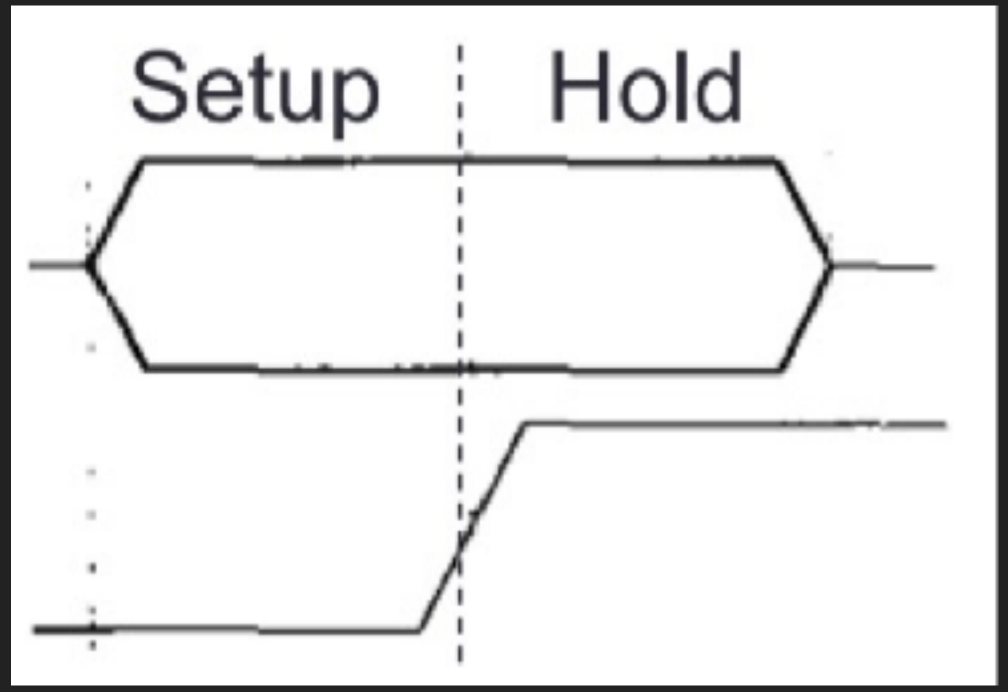
$$= \overline{A} + B + \overline{C} + A\overline{D} + \overline{B}\,\overline{D}$$

$$= \overline{A} + B + \overline{C} + \overline{D}$$

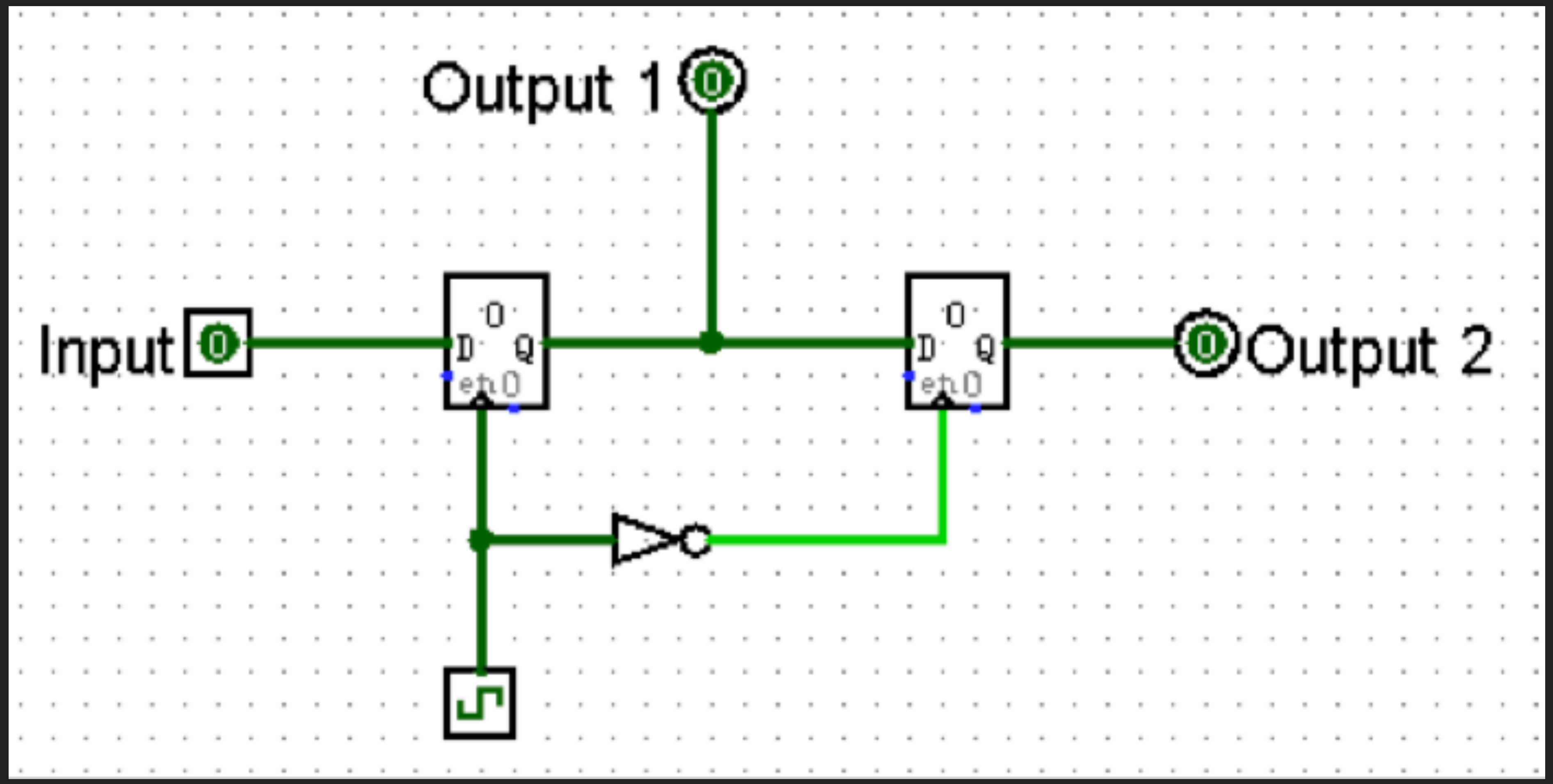
- 2021 Midterm I

SEQUENTIAL LOGIC

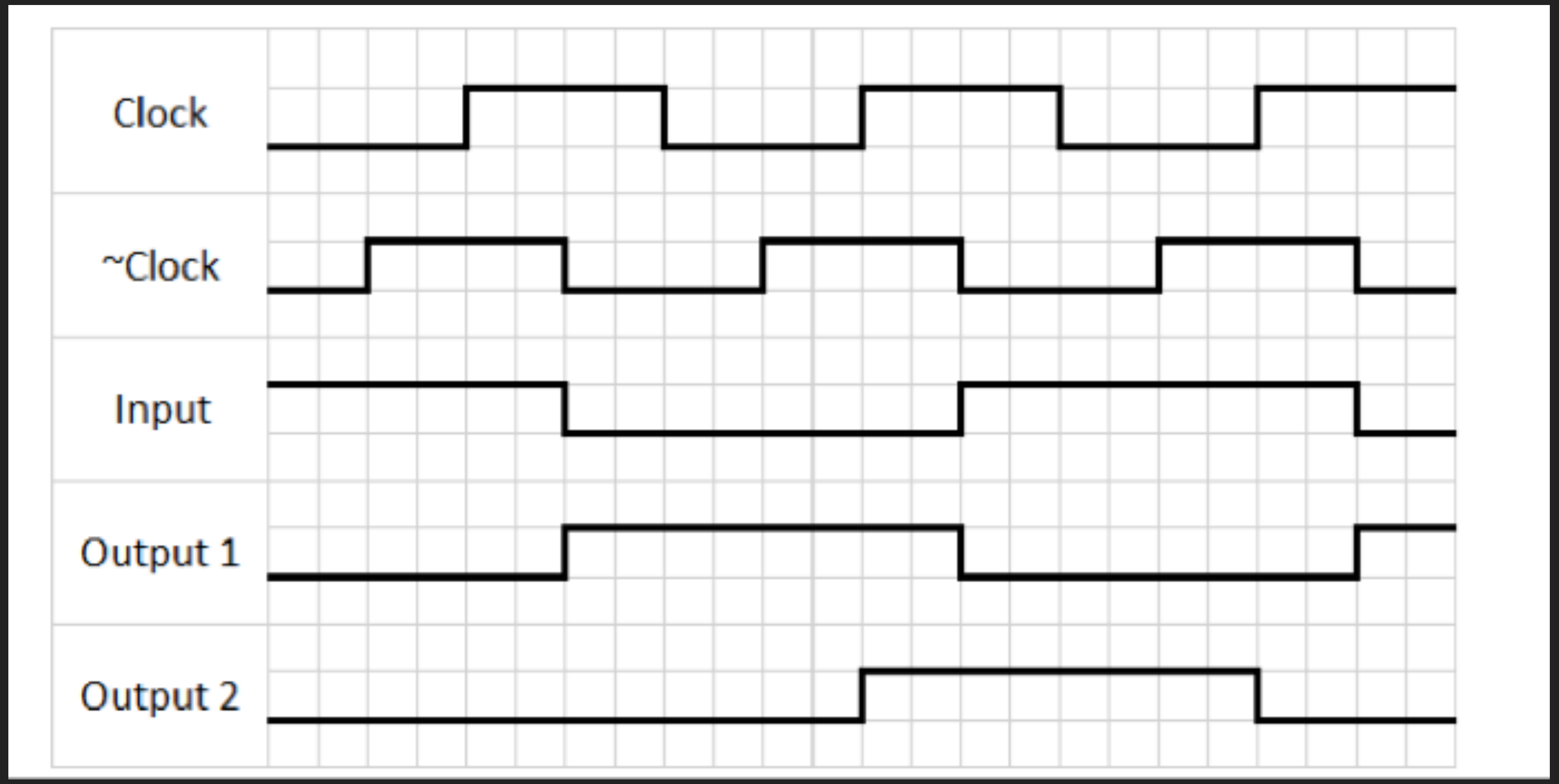
- State elements (e.g. registers) change value based on a clock signal
- clk-to-q delay: the time **between** rising edge of clock signal and the time register's output reflects the input change
- Setup time: the amount of time **before** rising edge of clock that the input must be stable
- Hold time: the amount of time **after** rising edge of clock that the input must be stable

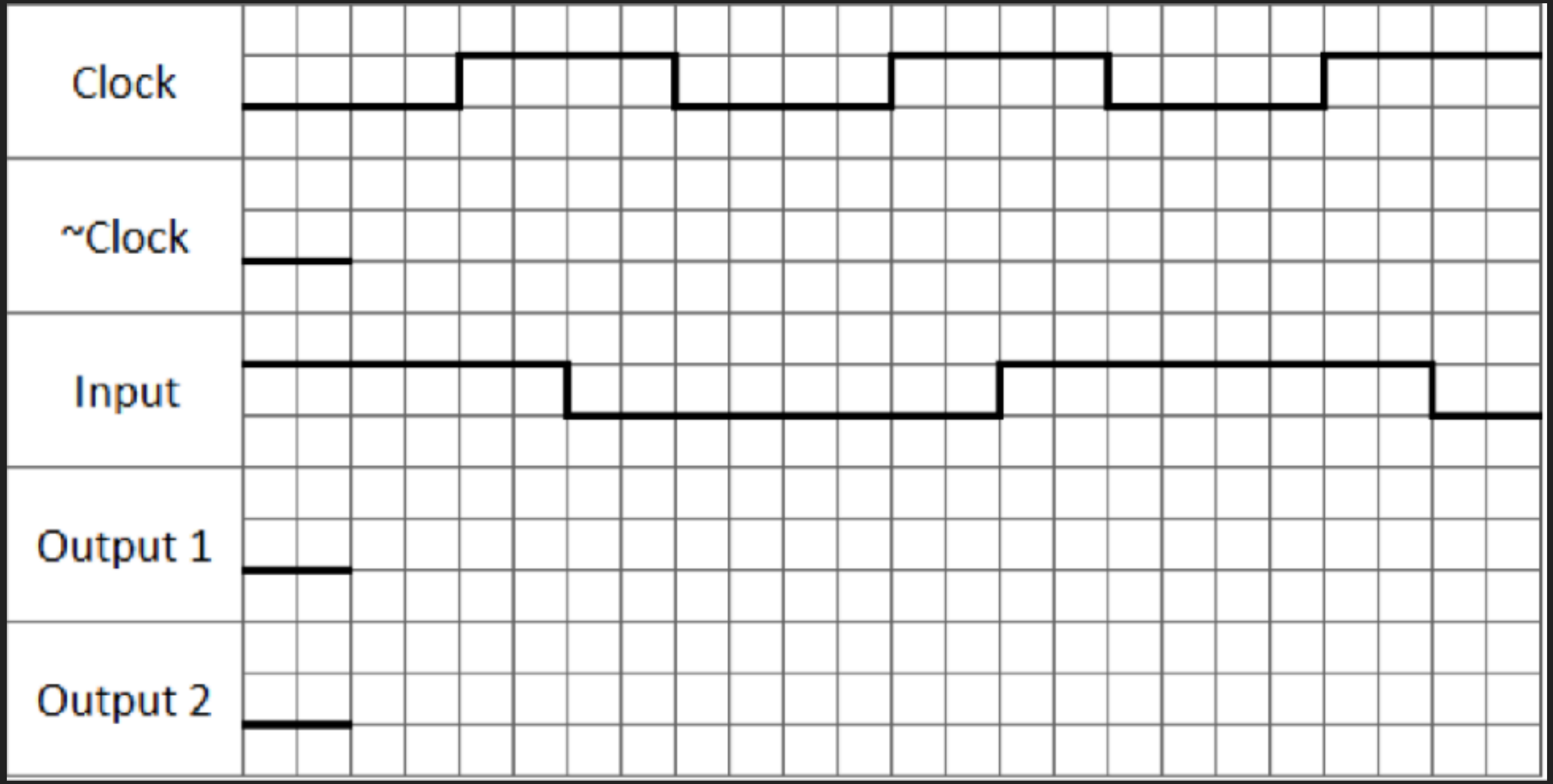


TIMING DIAGRAM FOR SDS



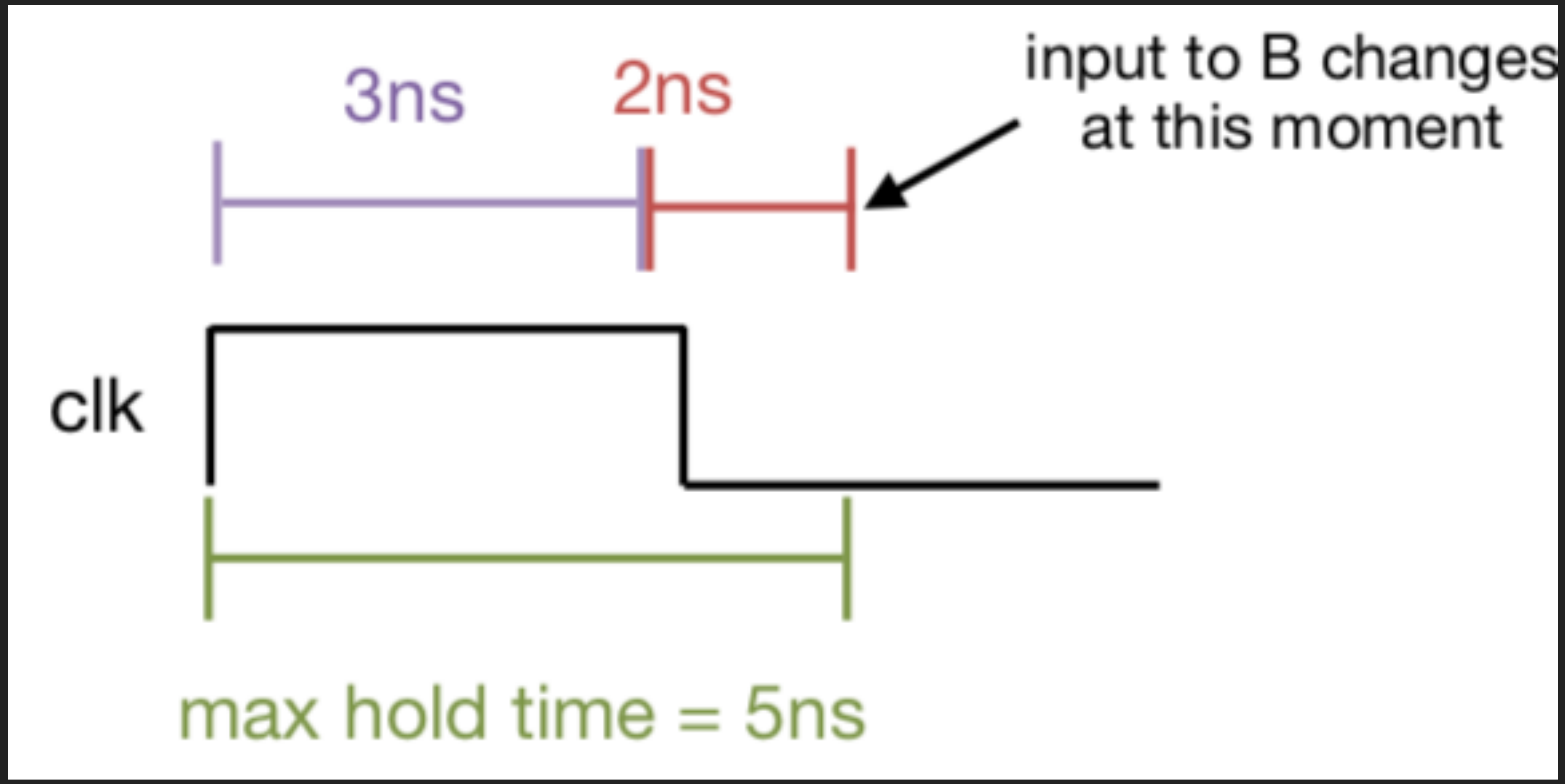
- 2021 Midterm 1
- NOT: 2ns, Reg clk-to-q: 2ns, setup: 2ns

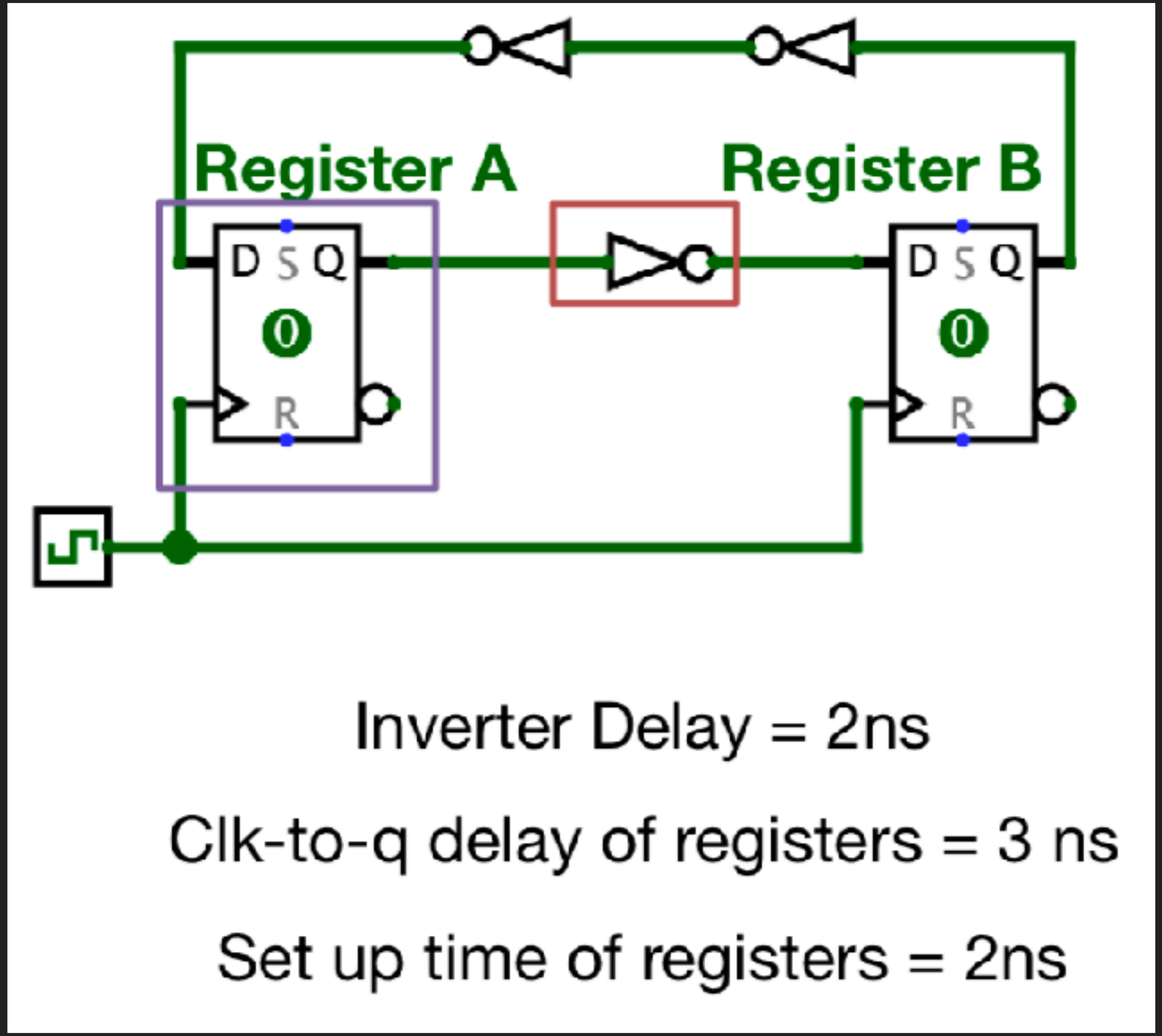




TIME CONSTRAINT FOR SDS

- Maximum Hold time for Register B
  = clk-to-q delay of A
  + combinational delay
  = 3ns + 2ns = 5ns





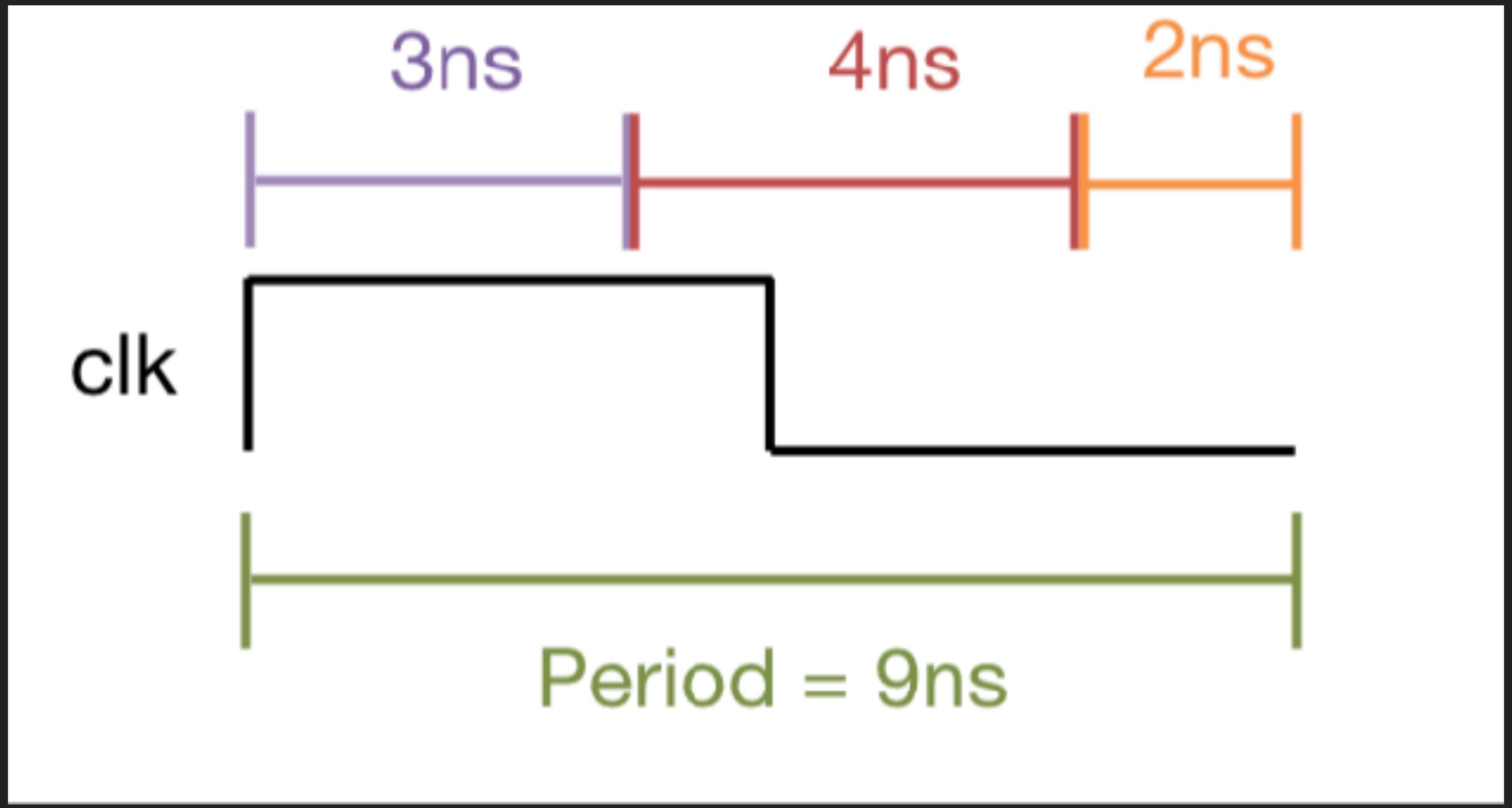
Inverter Delay = 2ns

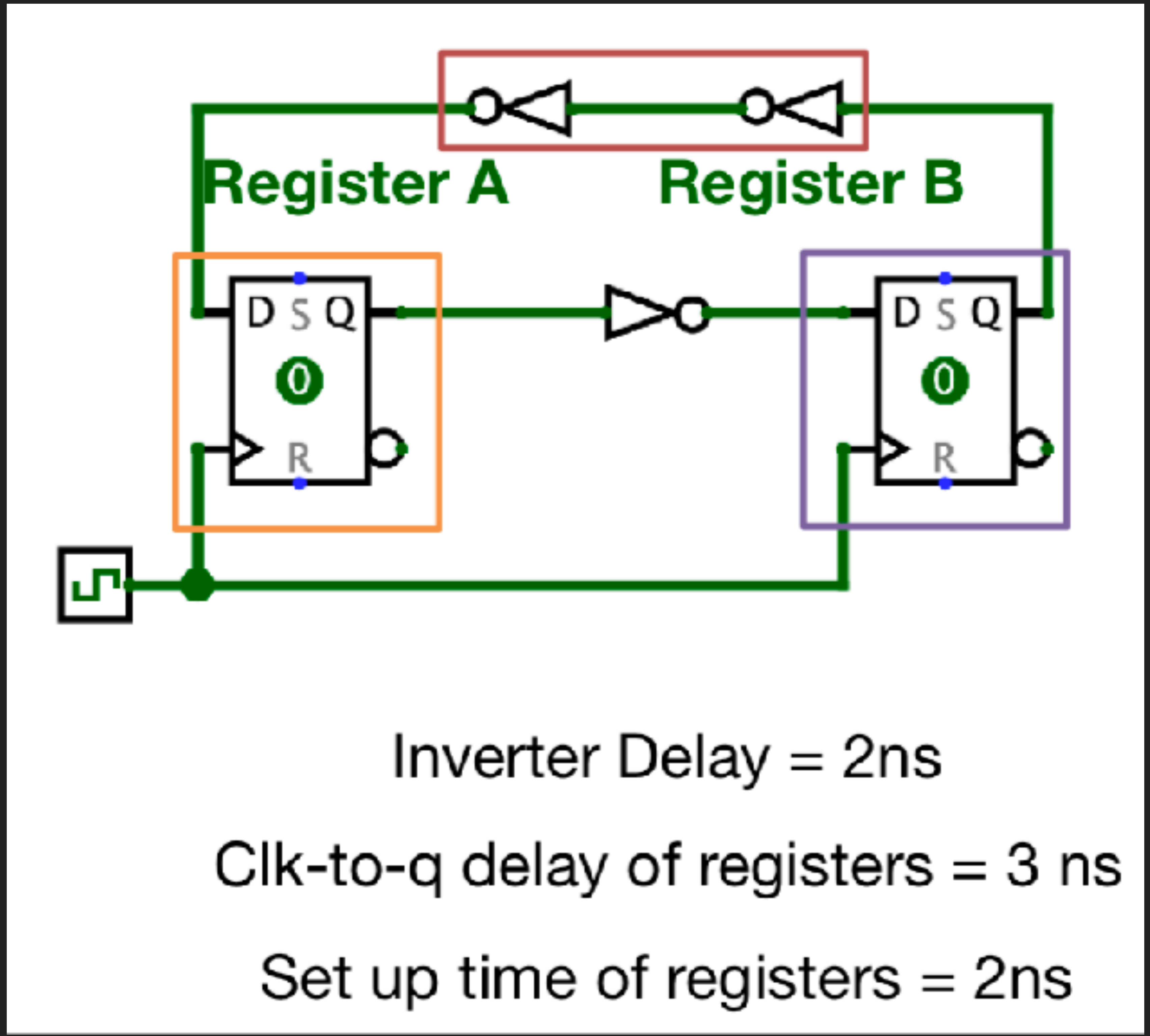
Clk-to-q delay of registers = 3 ns

Set up time of registers = 2ns

TIME CONSTRAINT FOR SDS

- Minimum clock cycle time
  = clk-to-q delay
  + longest combinational delay
  + setup time
  = 3ns + 4ns + 2ns = 9ns





Inverter Delay = 2ns

Clk-to-q delay of registers = 3 ns

Set up time of registers = 2ns

SUMMARY

- CALL:
  - Concept & Function of Compiler, Assembler, Linker, Loader
- SDS:
  - Circuit, Boolean Expression, Truth Table
  - Sequential Analysis