COMSM1302
Overview of Computer Architecture

Lecture 11
Computer architecture concepts

In the previous lecture

- Our instruction set.
- Instruction cycle.
- An example of assembly code.
- Assembly programming problem.

In this lecture

Assembly programming problem → Memory and CPU interface → CISC and RISC → Turing machine

- At the end of this lecture:
  - Von Neumann and Harvard architectures.
  - CISC and RISC architectures.
  - Turing machines.

Our CPU and computer architecture designs

Assembly programming problem → Memory and CPU interface → CISC and RISC → Turing machine

Assembly programming problem

- Write an assembly code using our ISA to calculate the difference between two two-digit numbers x and y.

1. After executing this program, the "A" register should contain the value $(x - y)$.
2. Your program should calculate correct answer for inputs that satisfy the following conditions:

   $0 \leq x - y \leq 15, 0 \leq x \leq 99, 0 \leq y \leq 99$, and the ones and tens of x are greater or equal to the ones and tens of y, respectively.

Store inputs in memory – 3/3

- Let x = 95 and y = 81
- x_0 = 0101 , x_1 = 1001, y_0 = 0001, and y_1 = 1000
- Now we can store these values in the memory and they can be accessed by our CPU.

Address	value
0xC	0x05
0xD	0x09
0xE	0x01
0xF	0x08

Plan

1. Some examples.
2. Store input data in the memory.
3. Discuss and analysis the problem.
4. Express the algorithm of our code as a flowchart.
5. Translate the over code operations to assembly instructions.
6. Sort out any memory issues.

Problem discussion – 1/3

- Let x = 95 and y = 81
- $x_0 = 5$, $x_1 = 9$, $y_0 = 1$, and $y_1 = 8$
- $x = x_0 + \; x_1 \;\; * 10$
- $y = y_0 + \; y_1 \;\; * 10$
- $x \; - y = \; x_0 - y_0 + (x_{1\,-}\, y_1) \;\; * 10$

- But we do not have multiplication instruction!

Problem discussion – 2/3

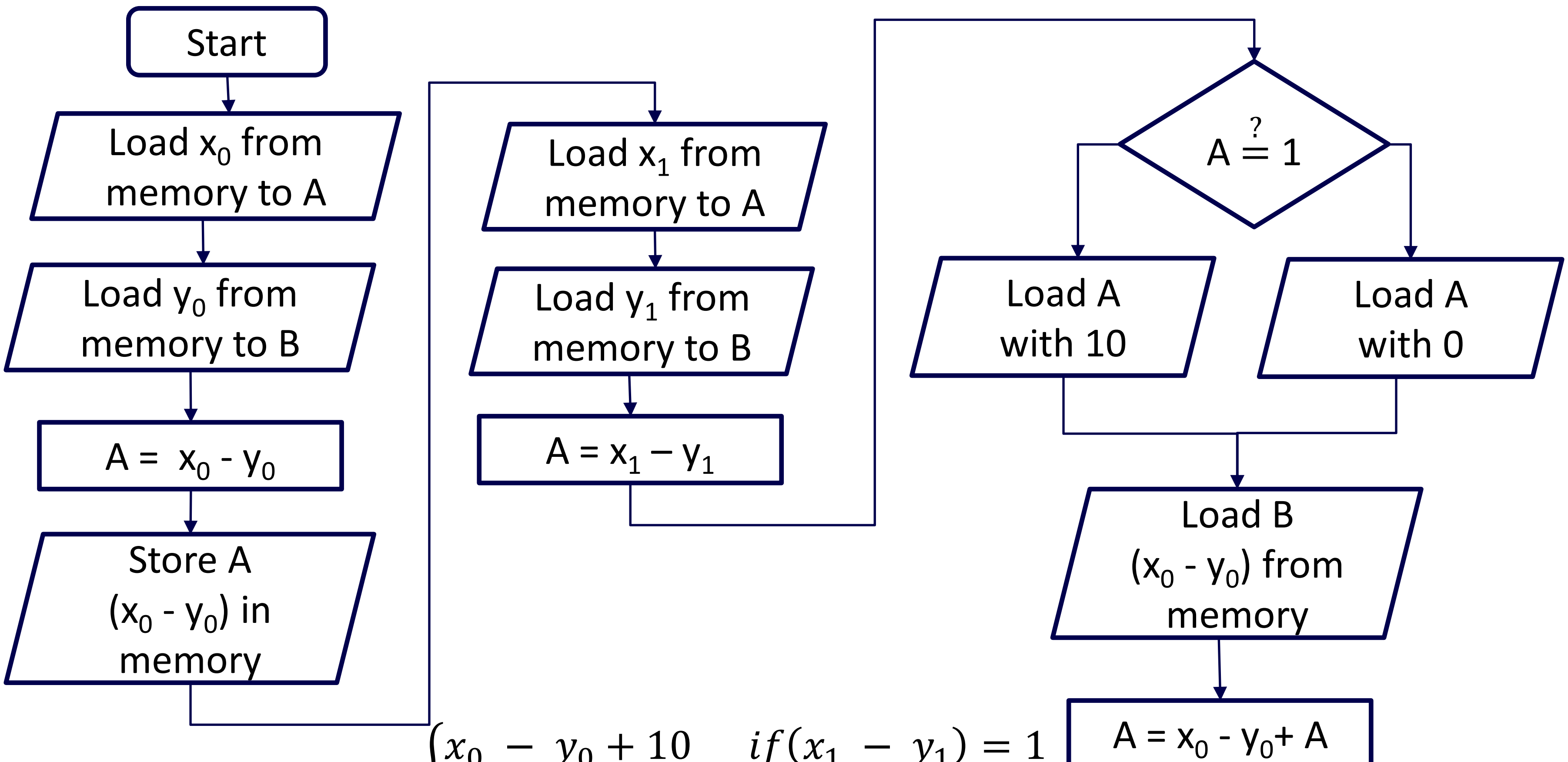
- We know $x - y \leq 15$
  - $0 \leq x_0 - y_0 \leq 9$
  - $0 \leq x_1 - y_1 \leq 1$, So $(x_1 - y_1)$ can be either 0 or 1.

- $x - y = x_0 - y_0 + (x_1 - y_1) * 10$
- $x - y = \begin{cases} x_0 - y_0 + 1 * 10 & if(x_1 - y_1) = 1 \\ x_0 - y_0 + 0 * 10 & if(x_1 - y_1) = 0 \end{cases}$

Problem discussion – 3/3

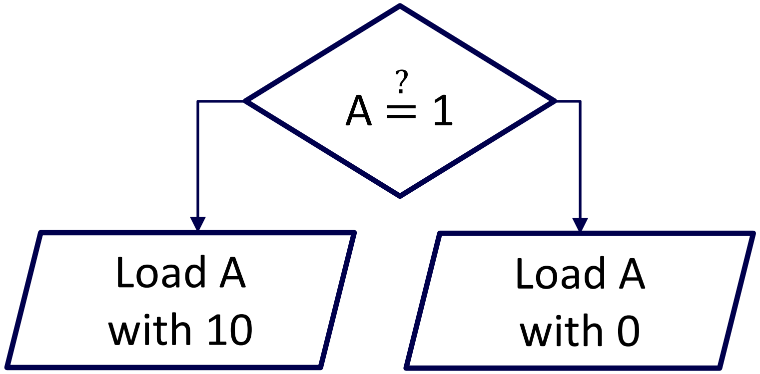
- $x - y = \begin{cases} x_0 - y_0 + 1 * 10 & if(x_1 - y_1) = 1 \\ x_0 - y_0 + 0 * 10 & if(x_1 - y_1) = 0 \end{cases}$
- $x - y = \begin{cases} x_0 - y_0 + 10 & if(x_1 - y_1) = 1 \\ x_0 - y_0 + 0 & if(x_1 - y_1) = 0 \end{cases}$
- We have managed to change the multiplication to comparison, but we also do not have comparison instruction!

Problem algorithm



$$x - y = \begin{cases} x_0 - y_0 + 10 & if(x_1 - y_1) = 1 \\ x_0 - y_0 + 0 & if(x_1 - y_1) = 0 \end{cases}$$

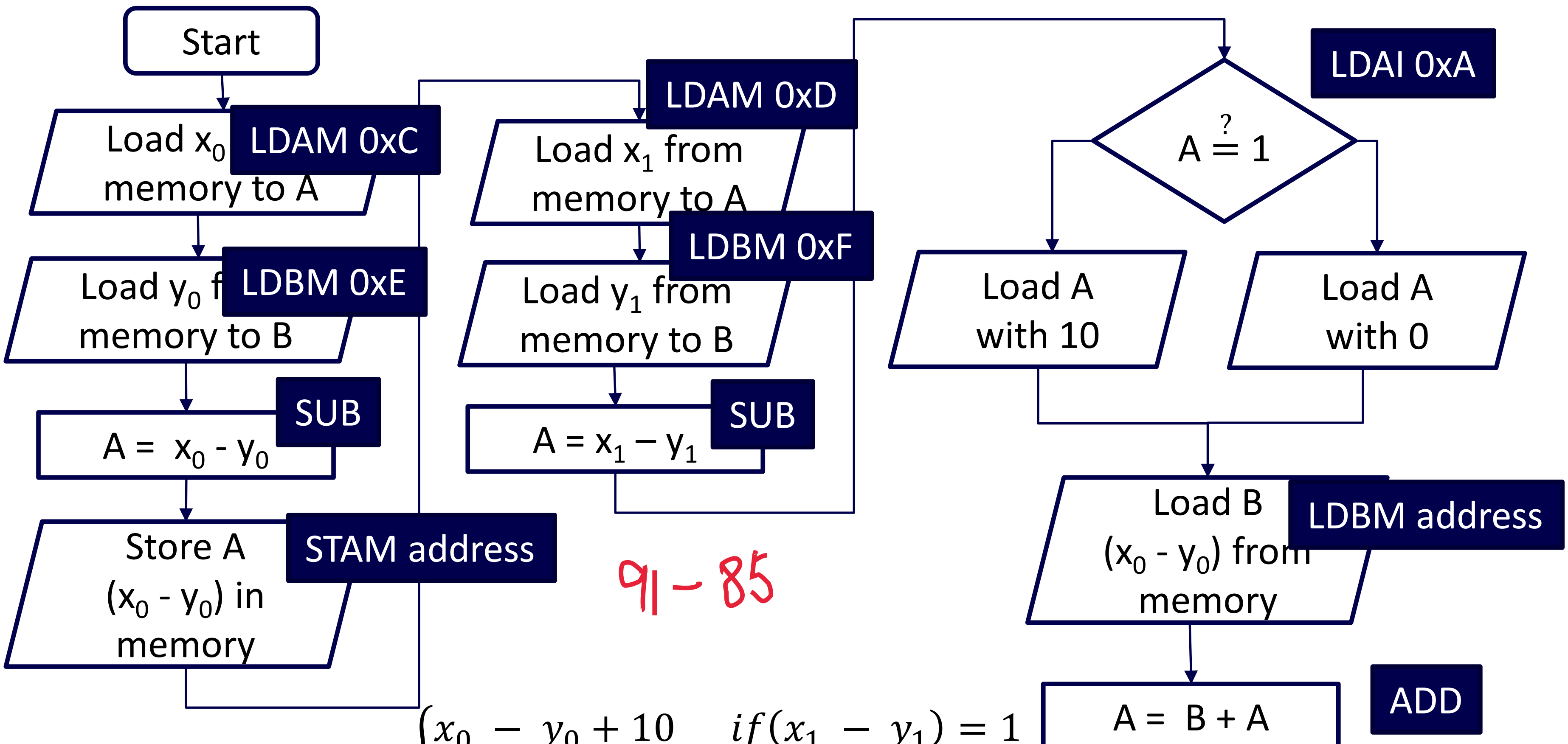
If - else statement



Address	value	
0xA	**0x00**	0
0xB	**0x0A**	10
0xC	0x01	
0xD	0x09	
0xE	0x05	
0xF	0x08	

- $if\,(x_1 - y_1) = 0$ load A with 0
- $if\,(x_1 - y_1) = 1$ load A with 10
- After calc $(x_1 - y_1)$, A will have 0 or 1.
- $A{+}0xA = \begin{cases} 0xA & if\ (x_1 - y_1) = 0 \\ 0xB & if\ (x_1 - y_1) = 1 \end{cases}$
- Load A with [A + 0xA]: LDAI 0xA.

Problem algorithm with instructions



$$x - y = \begin{cases} x_0 - y_0 + 10 & if(x_1 - y_1) = 1 \\ x_0 - y_0 + 0 & if(x_1 - y_1) = 0 \end{cases}$$

Problem assembly code

Add	Machine code	Current PC	Opcode	Operand	Next PC	Mnemonic	A	B
0x0		0x0				LDAM 0xC		
0x1						LDBM 0xE		
0x2						SUB		
0x3						STAM Addr		
0x4						LDAM 0xD		
0x5						LDBM 0xF		
0x6						SUB		
0x7						LDAI 0xA		
0x8						LDBM Addr		
0x9						ADD		

Store temp value – 1/3

Add	Machine code	Current PC	Opcode	Operand	Next PC	Mnemonic	A	B
0x0		0x0				LDAM 0xC		
0x1						LDBM 0xE		
0x2						SUB		
0x3						STAM Addr		
0x4						LDAM 0xD		
0x5						LDBM 0xF		
0x6						SUB		
0x7						LDAI 0xA		
0x8						LDBM Addr		
0x9						ADD		

Address	value
0xA	0x00
0xB	0x0A
0xC	0x05
0xD	0x09
0xE	0x01
0xF	0x08

Store temp value – 2/3

Add	Machine code	Current PC	Opcode	Operand	Next PC	Mnemonic	A	B
0x0		0x0				LDAM 0xC		
0x1						LDBM 0xE		
0x2	0110 XXXX					SUB		
0x3	0100 Addr					STAM Addr		
0x4						LDAM 0xD		
0x5						LDBM 0xF		
0x6						SUB		
0x7						LDAI 0xA		
0x8						LDBM Addr		
0x9						ADD		

Address	value
0xA	0x00
0xB	0x0A
0xC	0x05
0xD	0x09
0xE	0x01
0xF	0x08

Store temp value – 3/3

Add	Machine code	Current PC	Opcode	Operand	Next PC	Mnemonic	A	B
0x0		0x0				LDAM 0xC		
0x1						LDBM 0xE		
0x2	0110 0000					SUB		
0x3	0100 1010					STAM 0xA		
0x4						LDAM 0xD		
0x5						LDBM 0xF		
0x6						SUB		
0x7						LDAI 0x2		
0x8						LDBM 0xA		
0x9						ADD		

Address	value
0xA	0x00
0xB	
0xC	0x05
0xD	0x09
0xE	0x01
0xF	0x08

Diff between two two-digit numbers

Add	Machine code	Current PC	Opcode	Operand	Next PC	Mnemonic	A	B
0x0		0x0				LDAM 0xC		
0x1						LDBM 0xE		
0x2	0110 0000					SUB		
0x3	0100 1010					STAM 0xA		
0x4						LDAM 0xD		
0x5						LDBM 0xF		
0x6						SUB		
0x7						LDAI 0x2		
0x8						LDBM 0xA		
0x9						ADD		

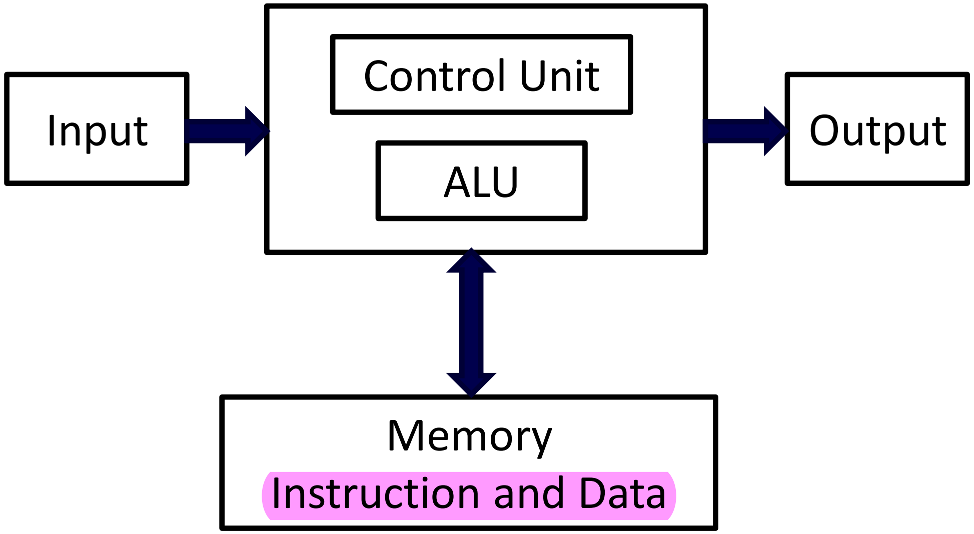
Address	**value**
0xA	0x00
0xB	
0xC	0x05
0xD	0x09
0xE	0x01
0xF	0x08

Our CPU and computer architecture designs

Assembly programming problem → **Memory and CPU interface** → CISC and RISC → Turing machine

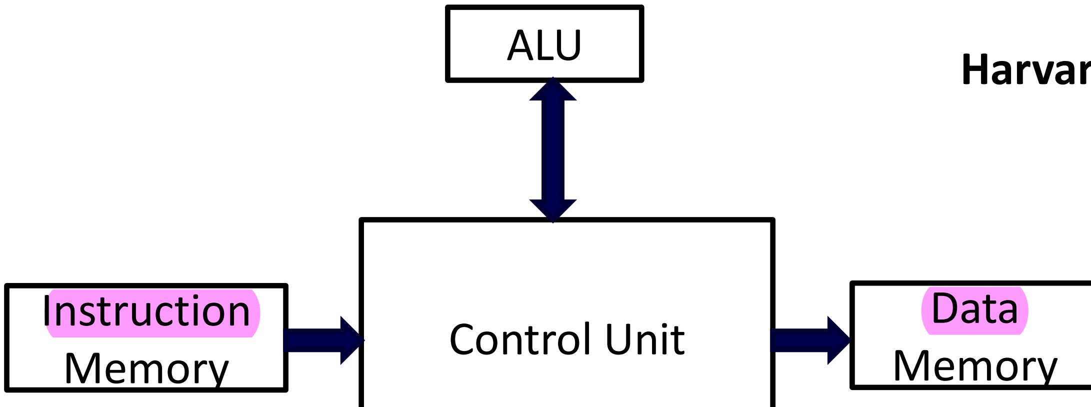
Von Neumann Architecture

John von Neumann

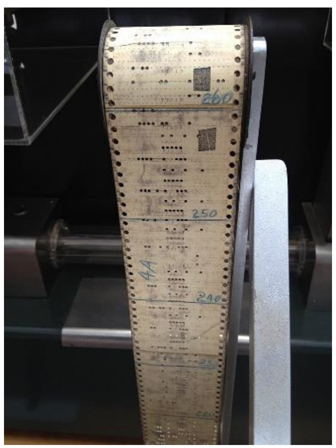


Von Neumann Architecture or Princeton Architecture

Harvard Architecture

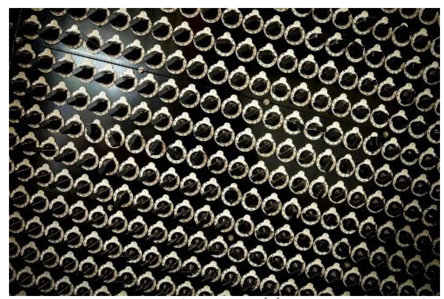


Harvard Architecture



https://upload.wikimedia.org/wikipedia/commons/thumb/f/fa/Harvard_Mark_I_program_tape.agr.jpg/800px-Harvard_Mark_I_program_tape.agr.jpg

Harvard Mark I relay-based computer - 1944



https://www.seas.harvard.edu/sites/default/files/styles/banner_large_wide_1x/public/2021-07/070621_Mark_1_SEC_5163.jpg?itok=DFGb3F0v

https://www.seas.harvard.edu/sites/default/files/styles/banner_large_wide_1x/public/2021-07/070621_Mark_1_SEC_5173.jpg?itok=93nZxp4u

Compare Harvard and Von Neumann architectures

- In Harvard architecture:
  1. Instruction's length does not have to match data width.
  2. Different busses for data and instructions.
  3. Instruction can be fetched, and data can be read or written at the same time.
  4. They cannot change their own instructions.

Our CPU and computer architecture designs.

Assembly programming problem → Memory and CPU interface → **CISC and RISC** → Turing machine

CISC and RISC

- In CISC (Complex Instruction Set Computer) one instruction can execute a whole sequence of hardware operations.

- In RISC (Reduced Instruction Set Computer), one instruction perform one hardware operation.

CISC

- CISC aims to simplify the compilation of high-level programming languages.

1. A large number of instructions.
2. Some instructions perform specialized tasks.
3. Variable length instructions.
4. Instructions that manipulate operands in memory.

RISC

- RISC attempts to reduce the execution time by simplifying the instruction sets.

1. Relatively few instructions.
2. Memory access limited to load and store.
3. All operations are done within the registers.
4. Fixed-length, easily decoded instructions.

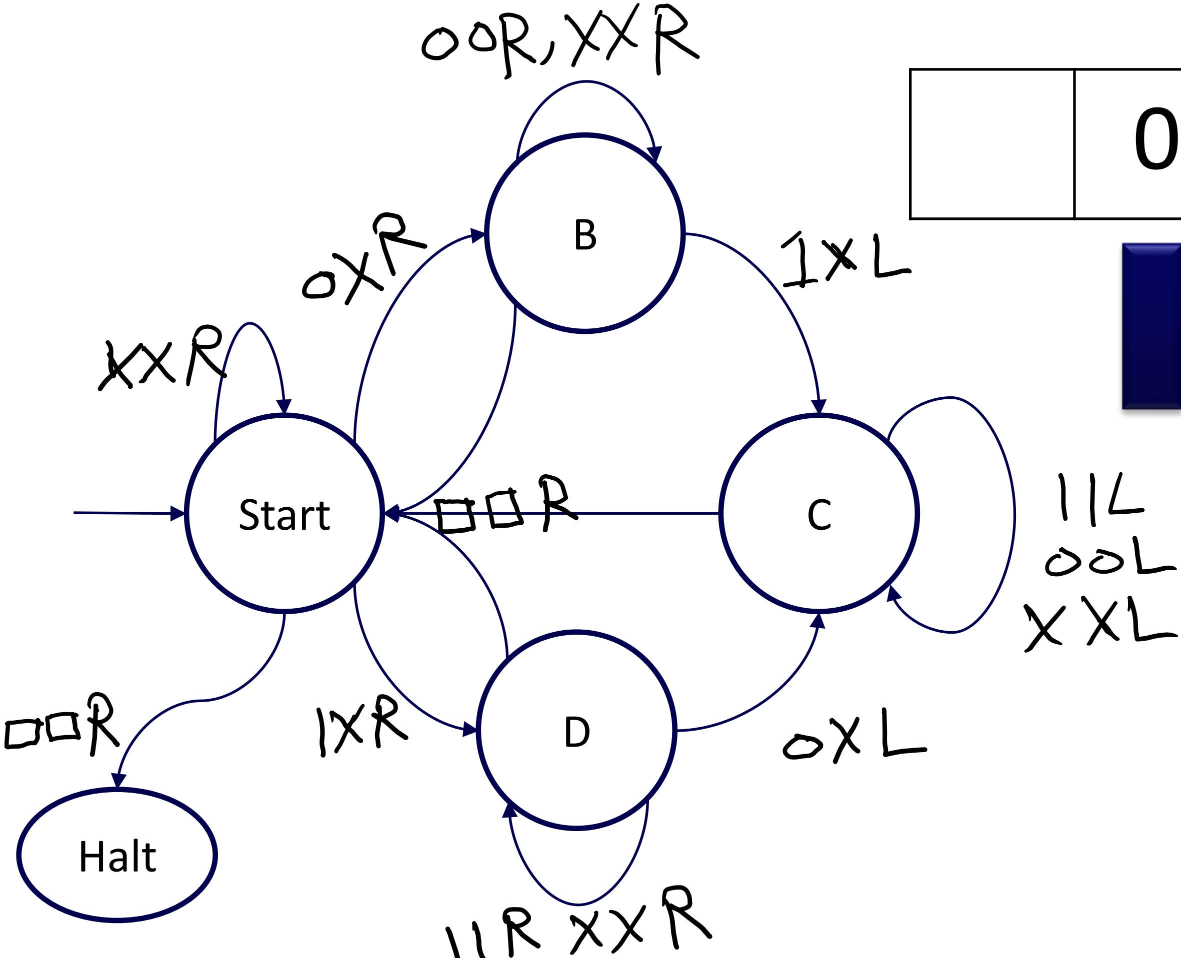
Our CPU and computer architecture designs

Assembly programming problem → Memory and CPU interface → CISC and RISC → Turing machine

Alan Turing

- Alan Turing was an English mathematician, computer scientist and logician.
- His work is widely acknowledged as foundational research of computer science
- Turing machine (1936).

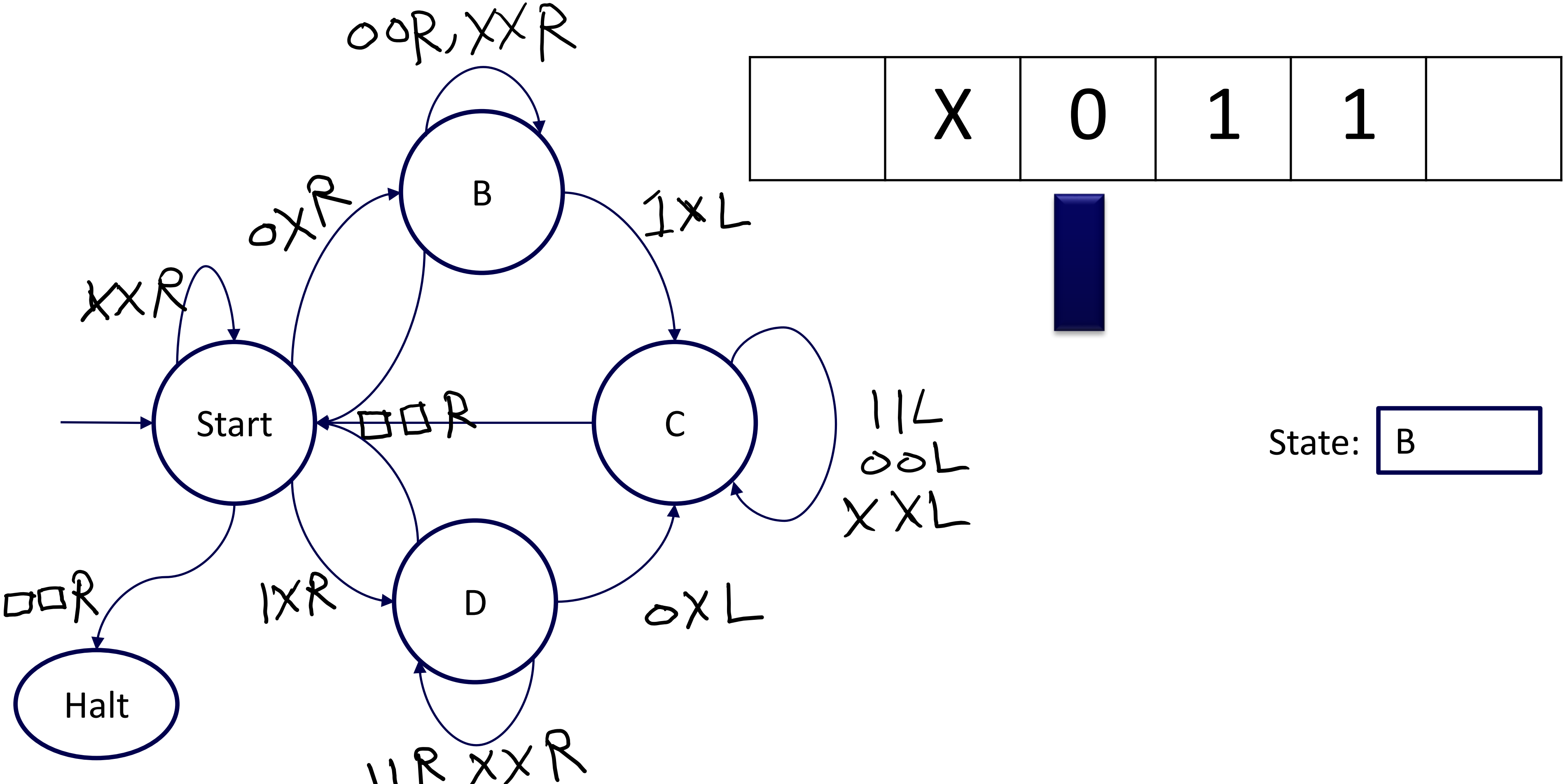
Turing machine - 1/15



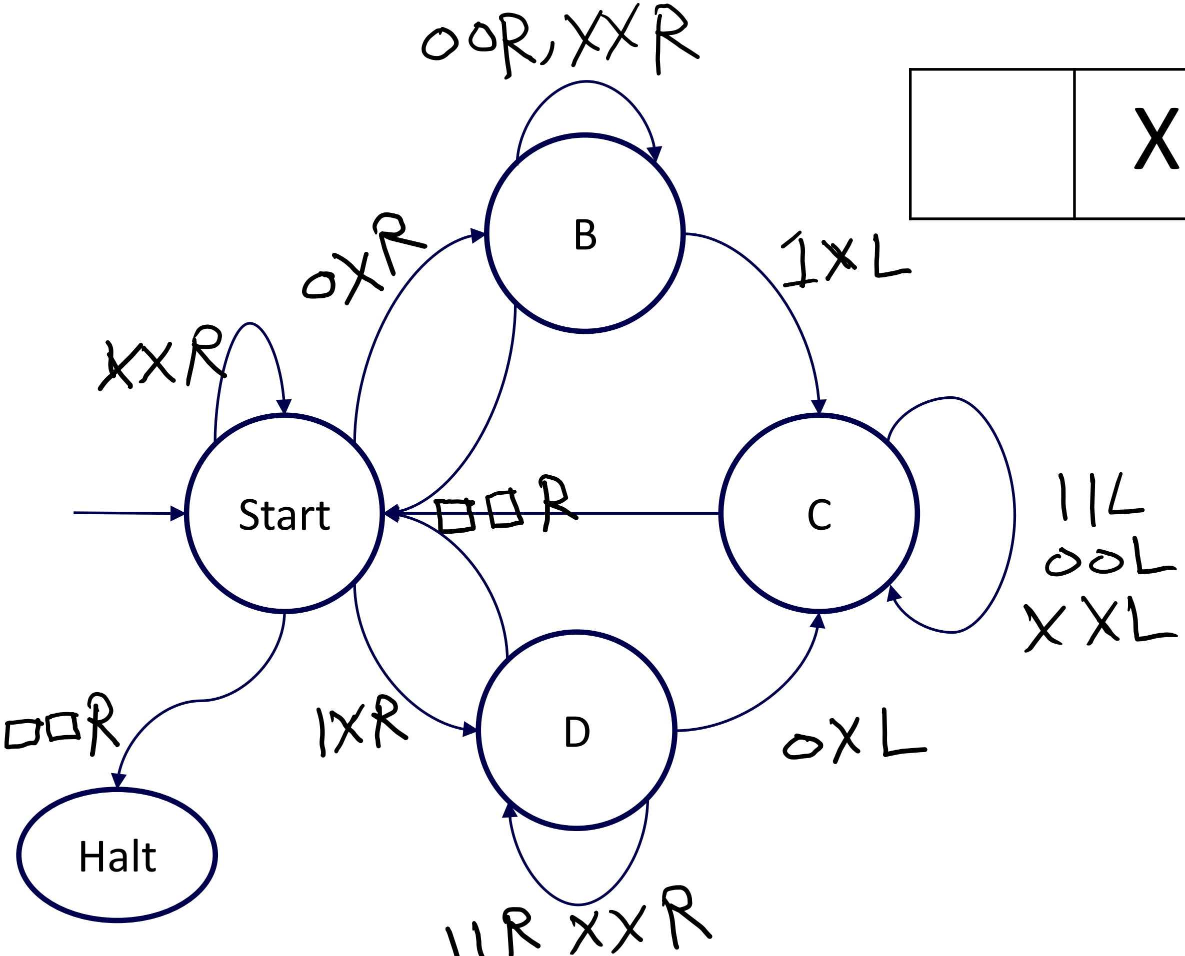
	0	0	1	1	

State: Start

Turing machine - 2/15



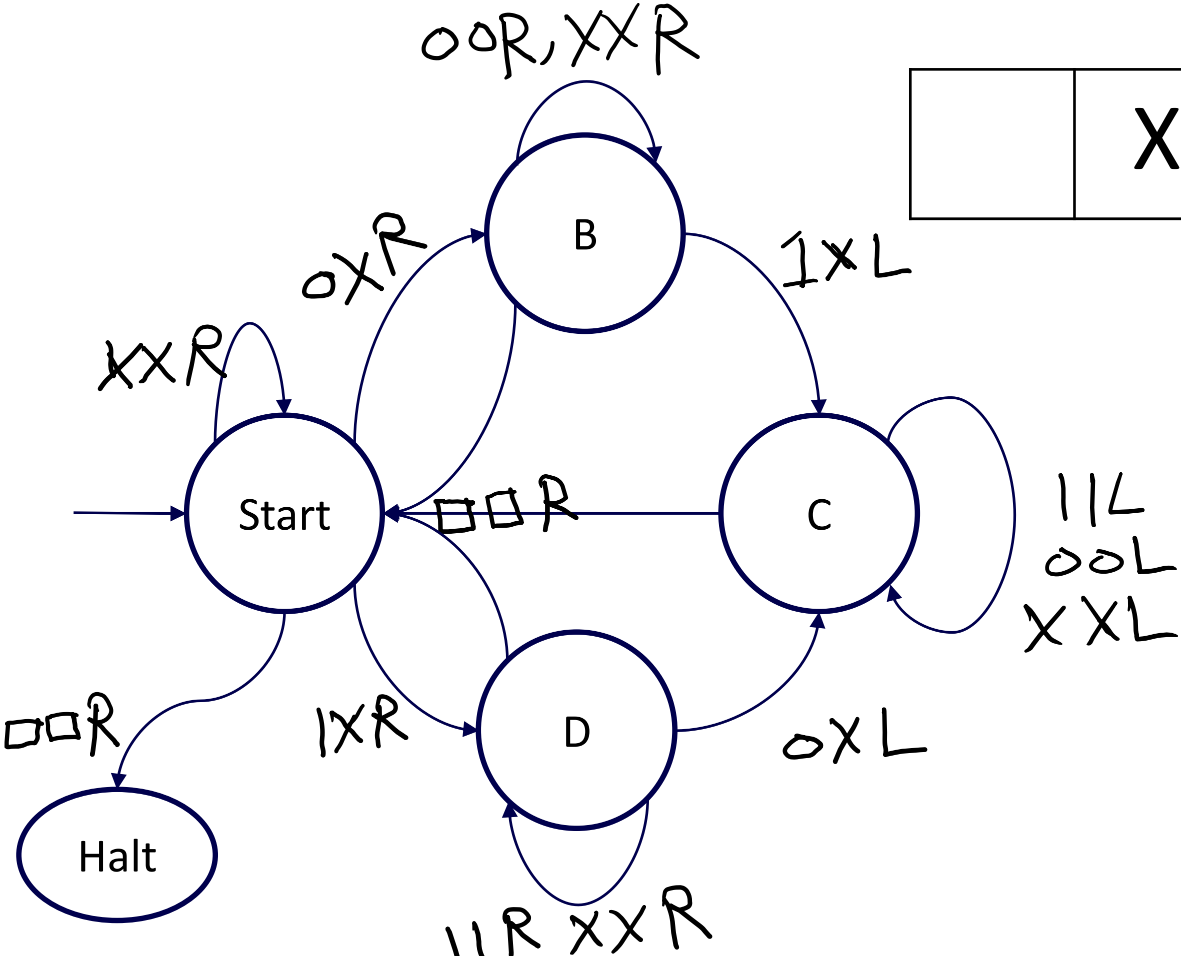
Turing machine - 3/15

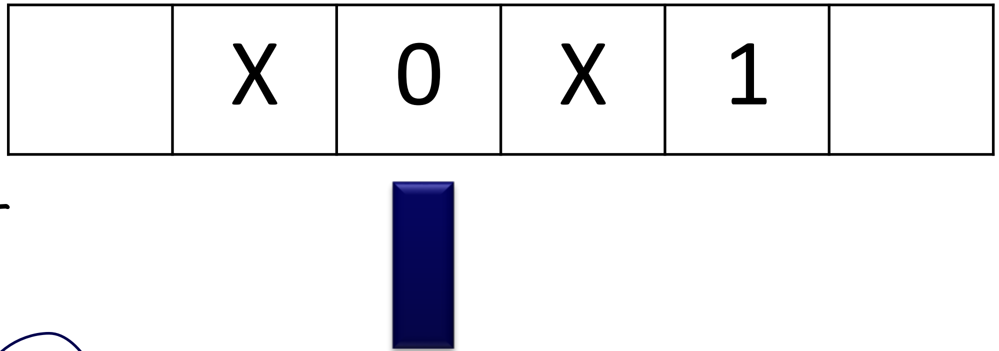


	X	0	1	1	

State: B

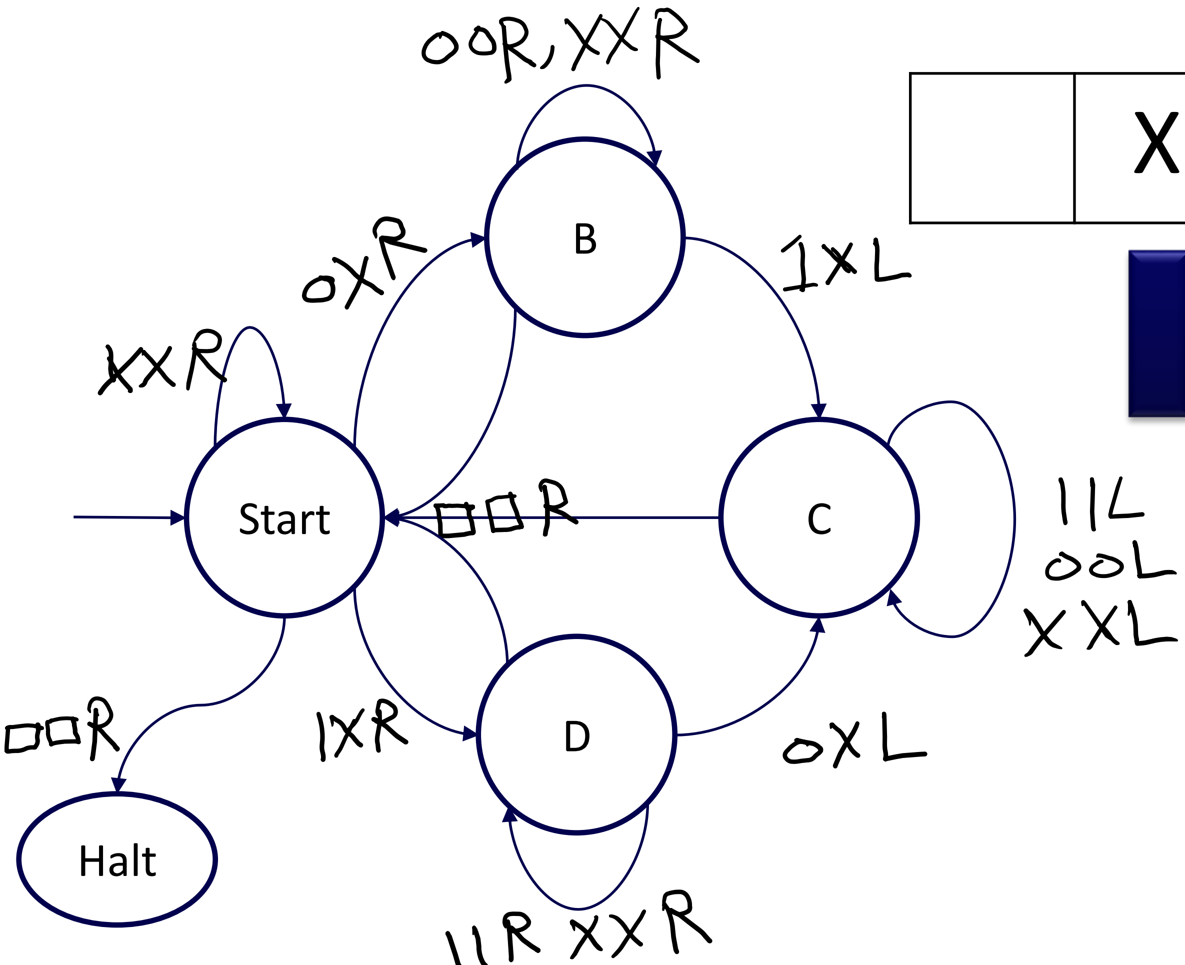
Turing machine - 4/15





State: C

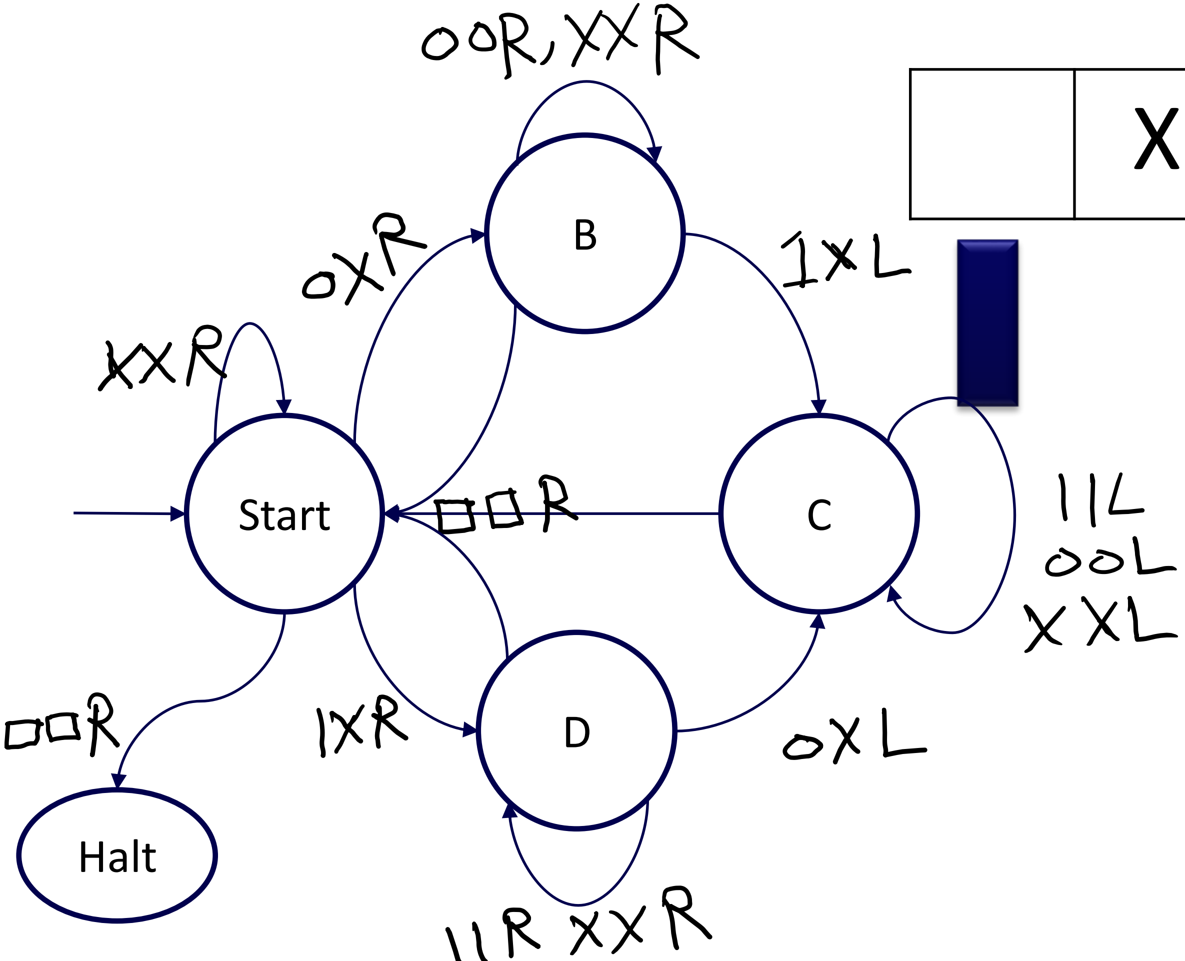
Turing machine - 5/15



	X	0	X	1	

State: C

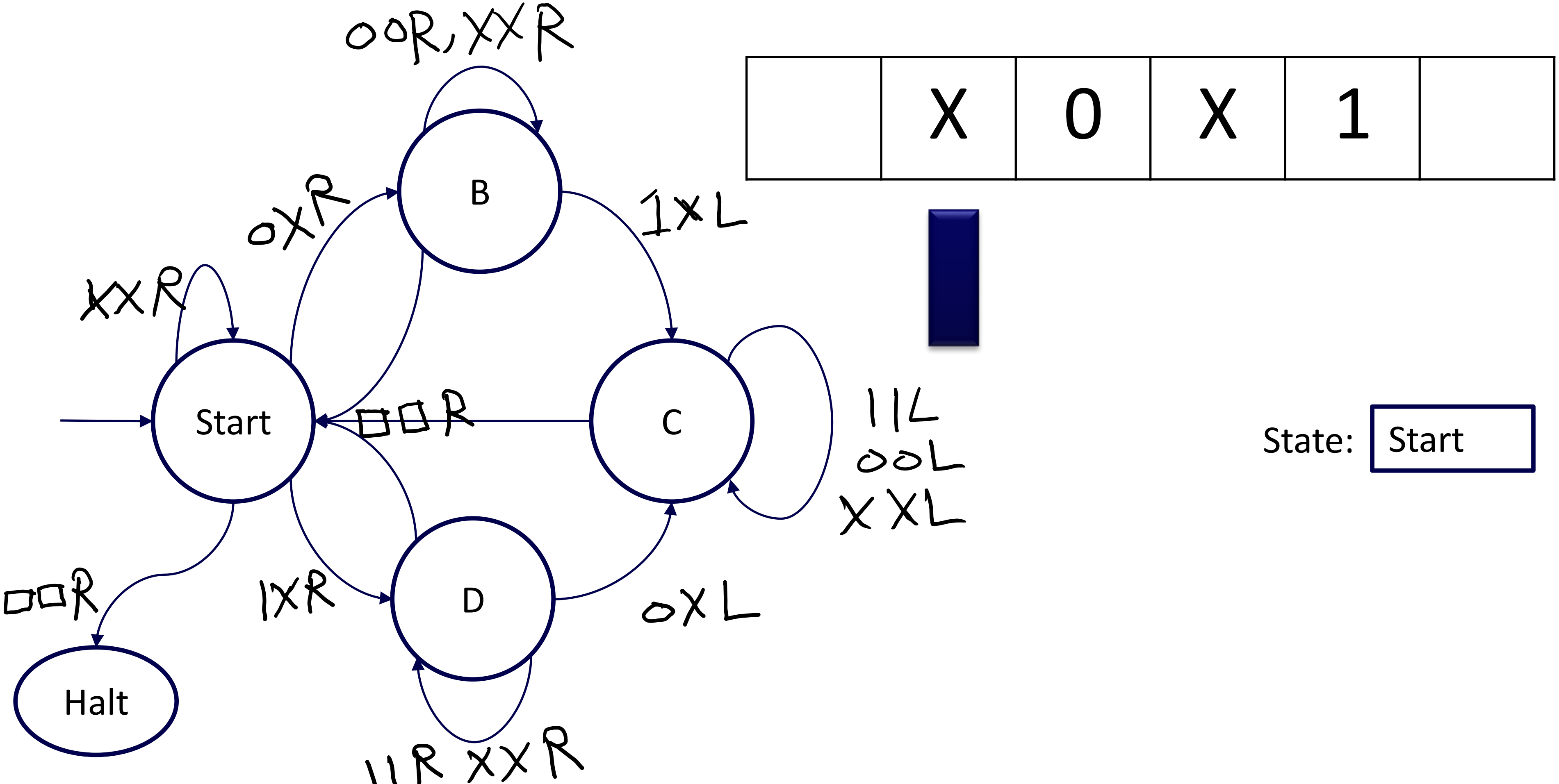
Turing machine - 6/15



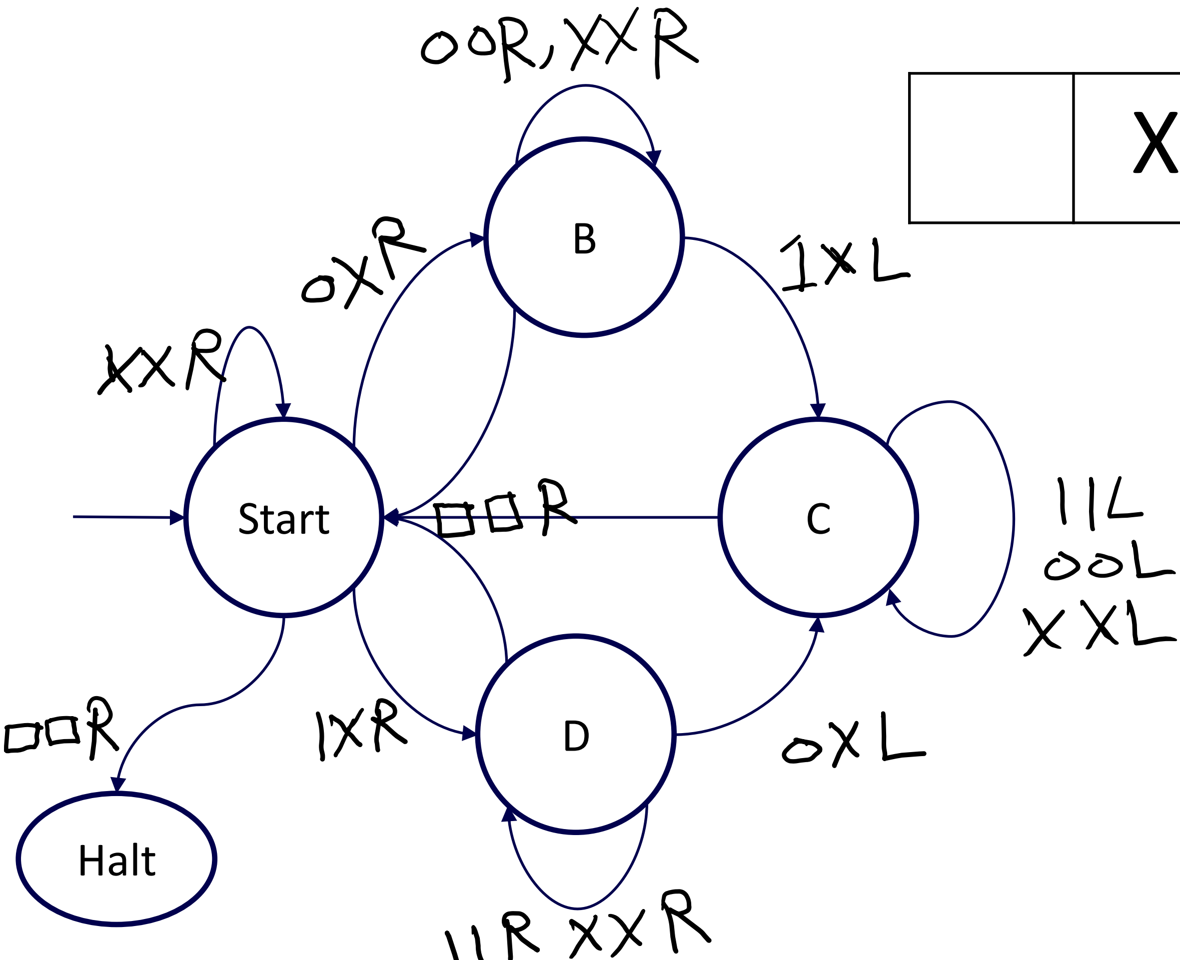
	X	0	X	1	

State: C

Turing machine - 7/15



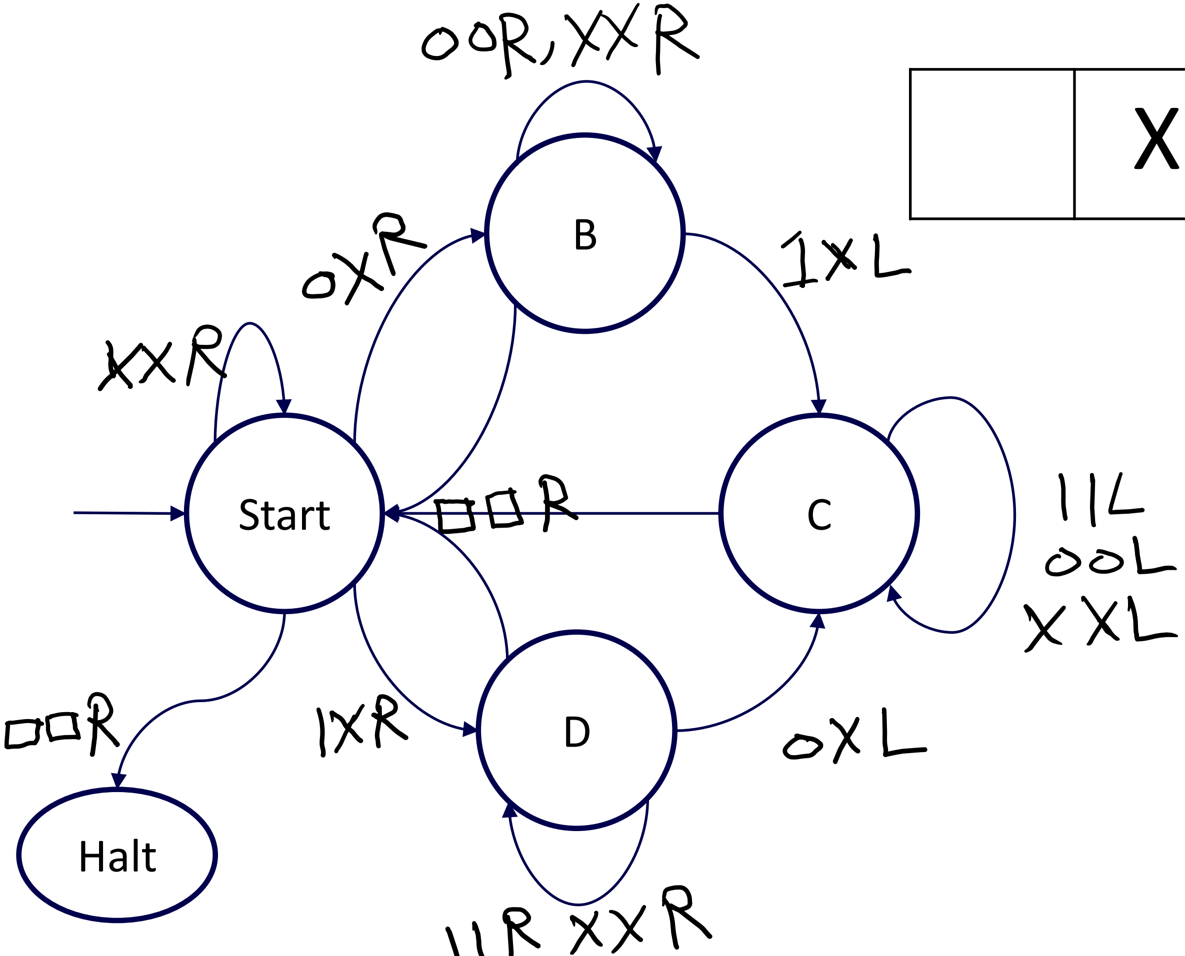
Turing machine - 8/15

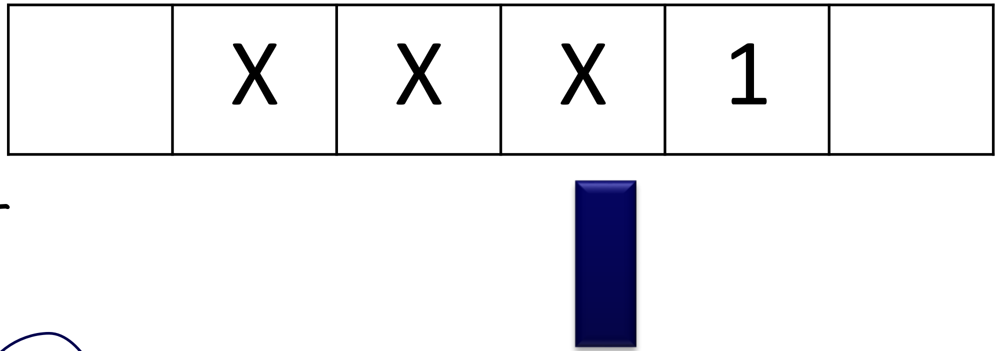


	X	0	X	1	

State: Start

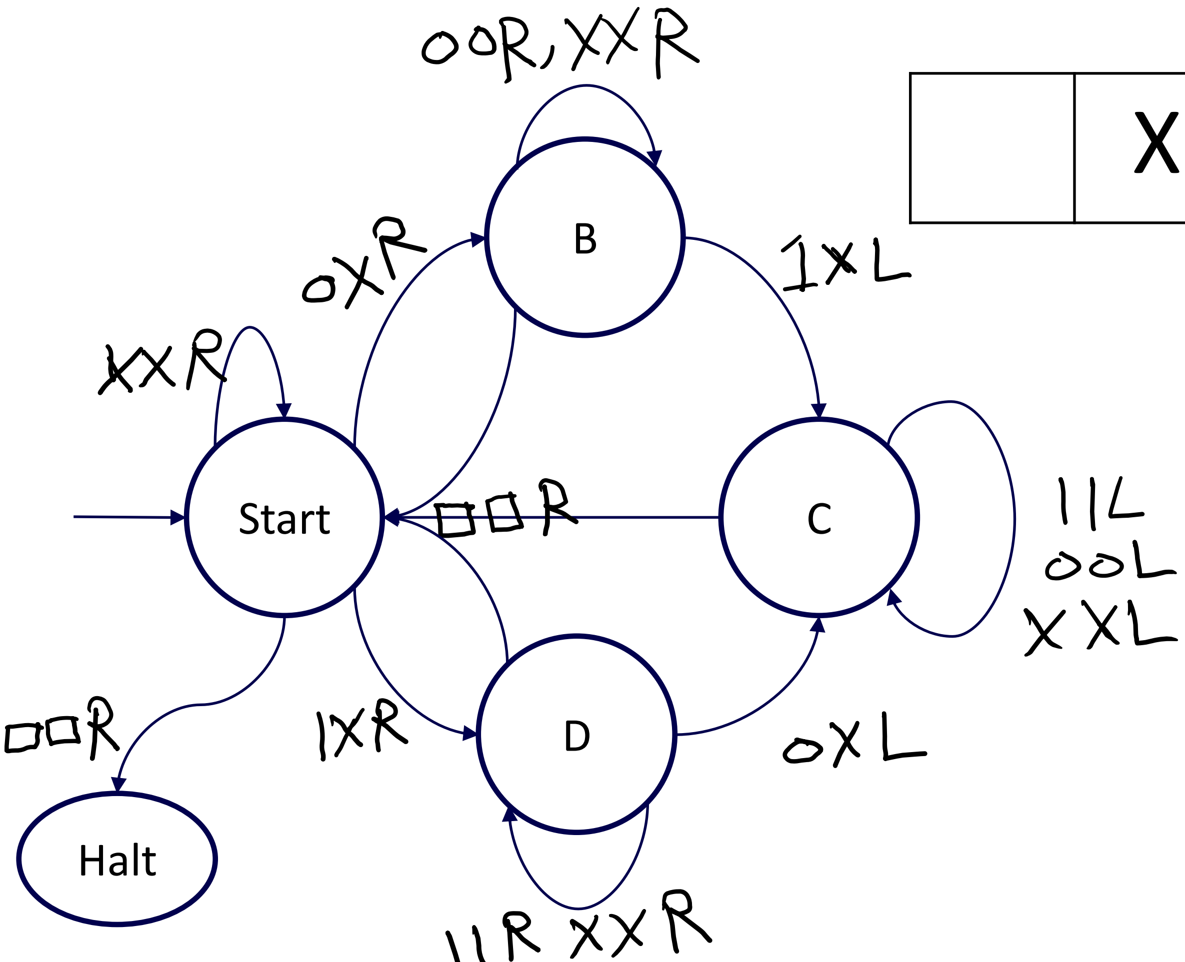
Turing machine - 9/15





State: B

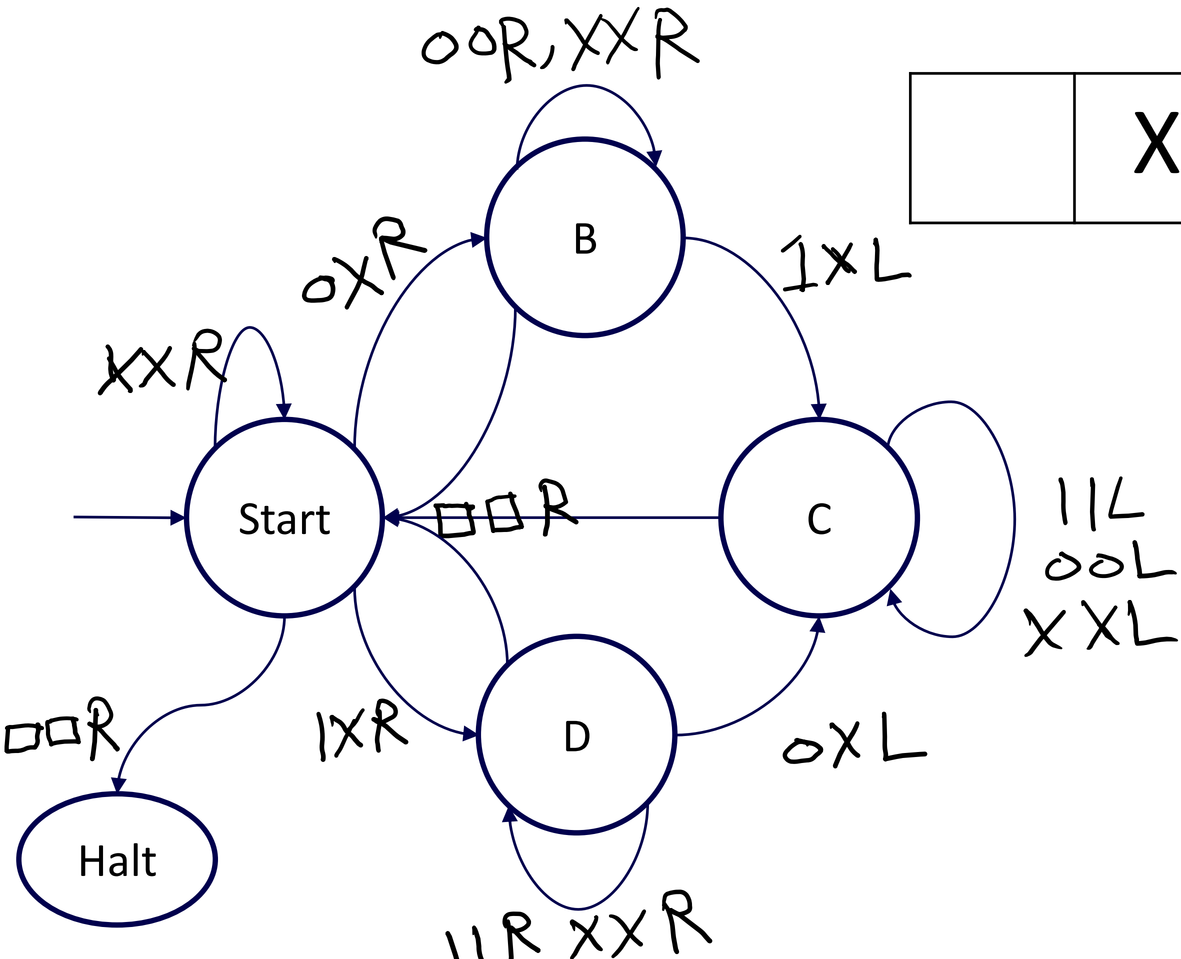
Turing machine - 10/15

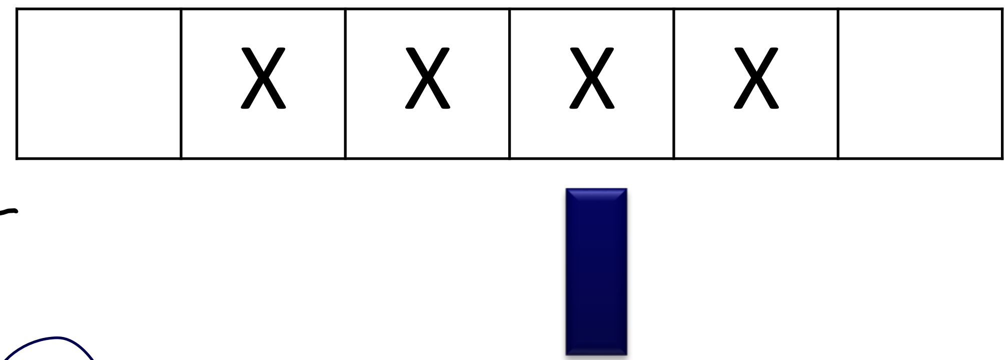


	X	X	X	1	

State: B

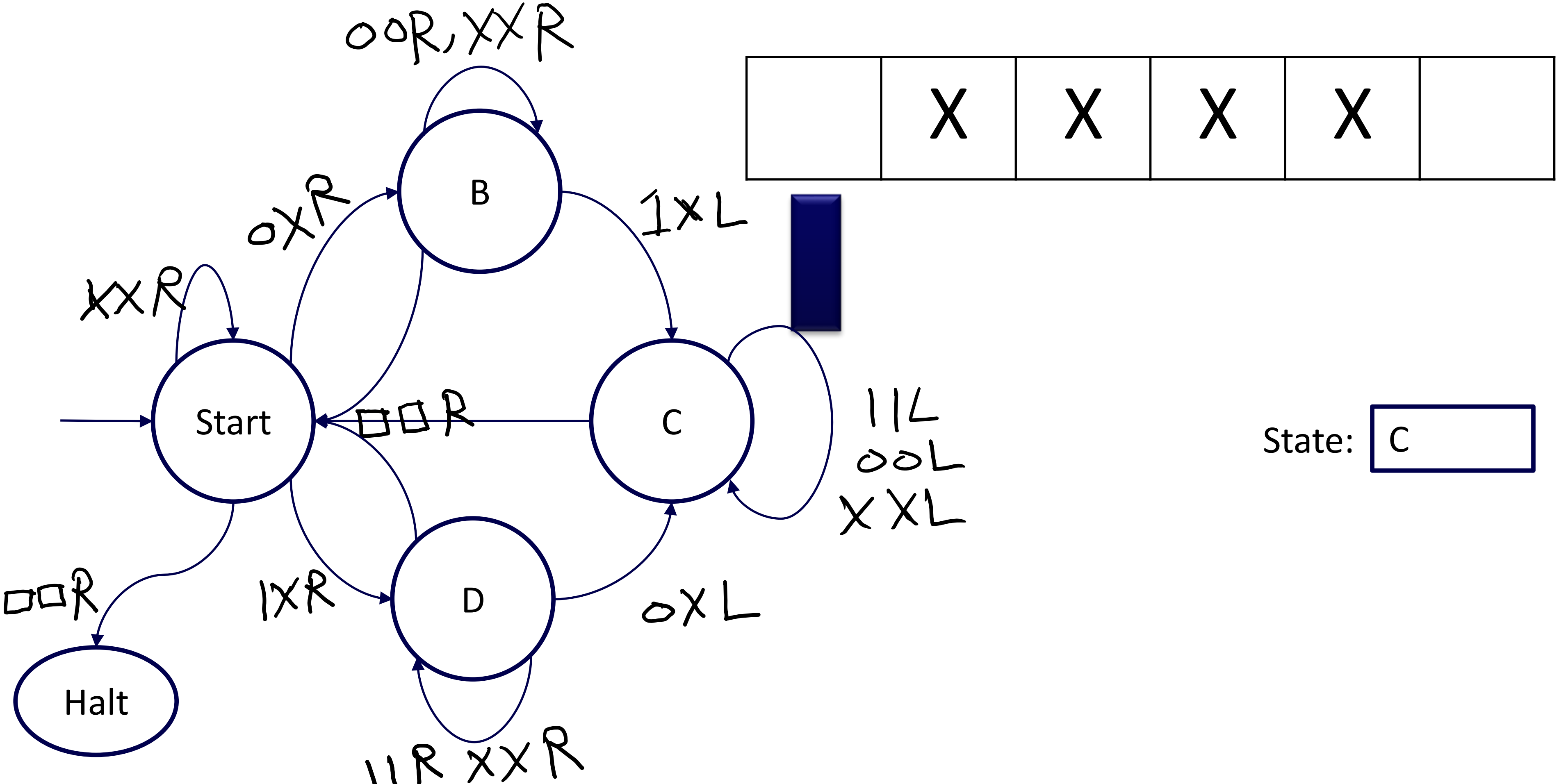
Turing machine - 11/15



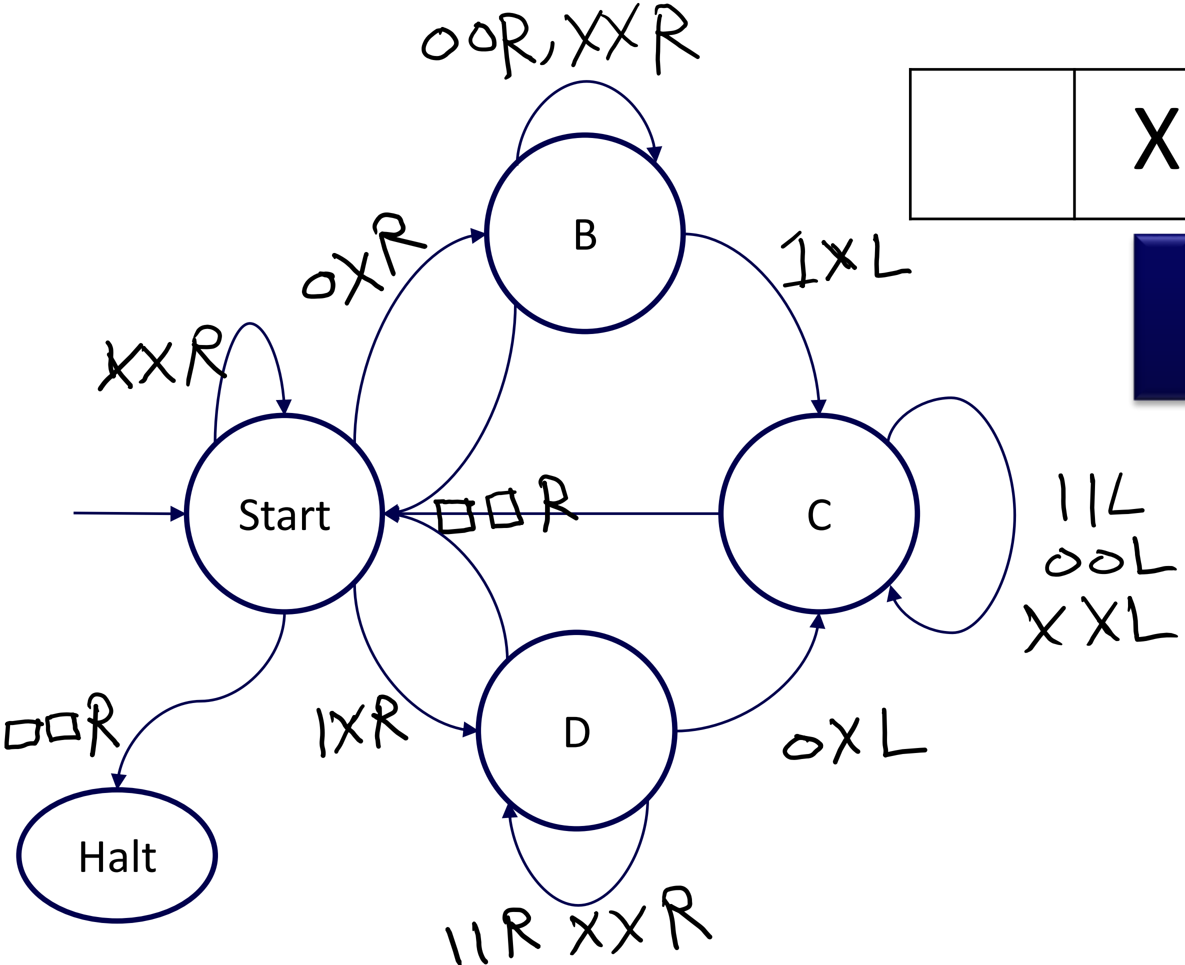


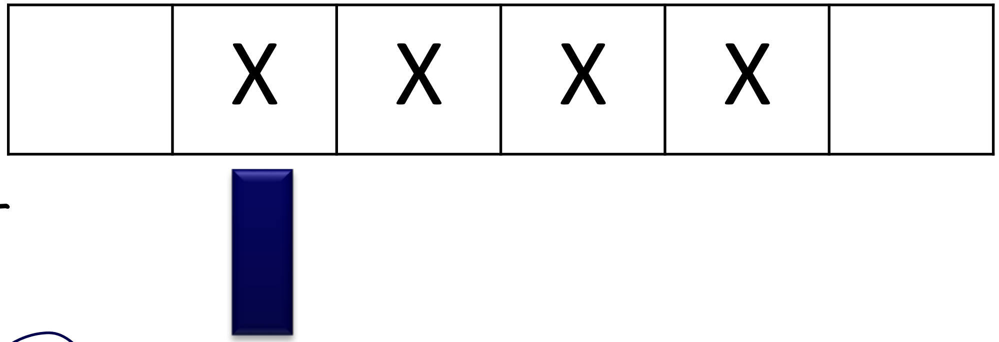
State: C

Turing machine - 12/15



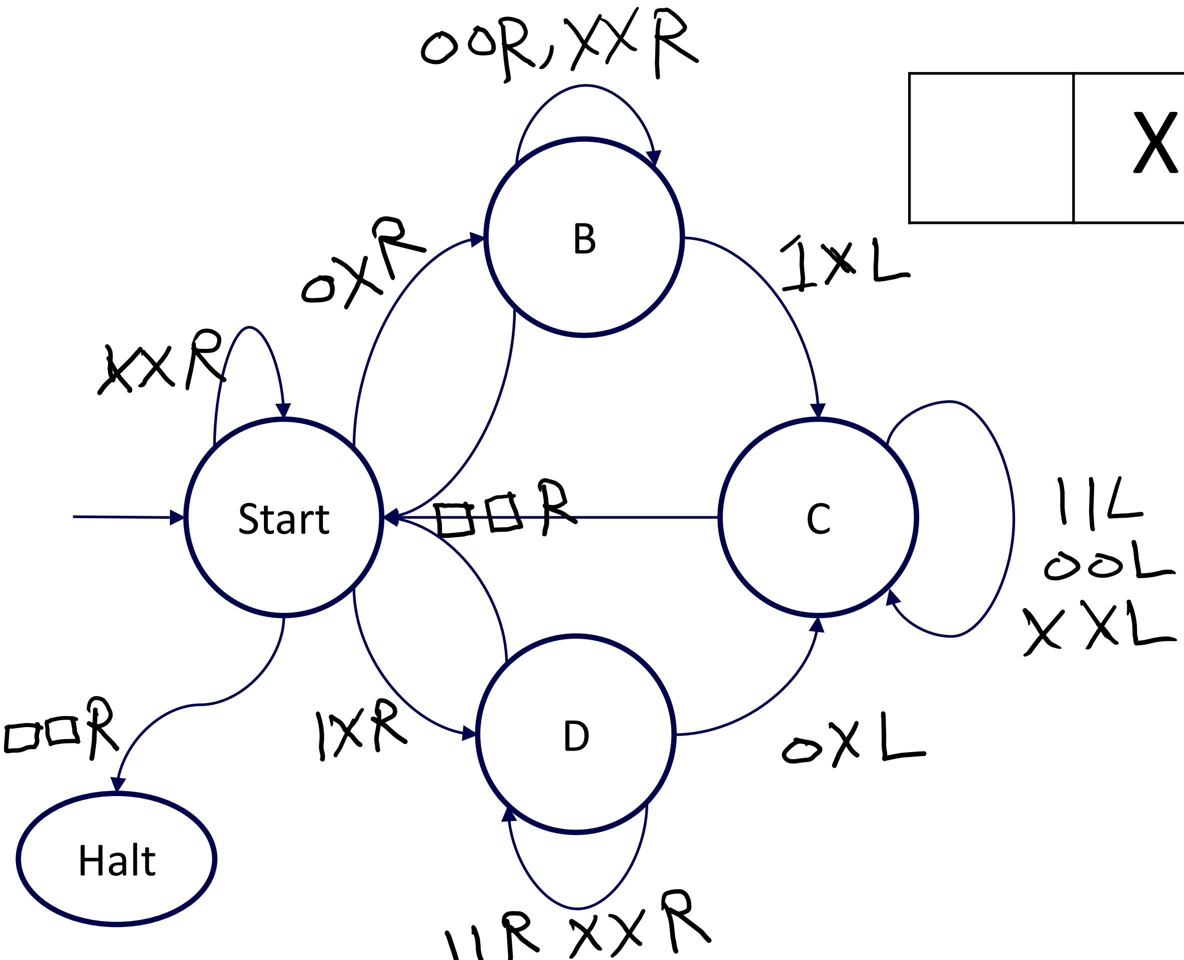
Turing machine - 13/15





State: Start

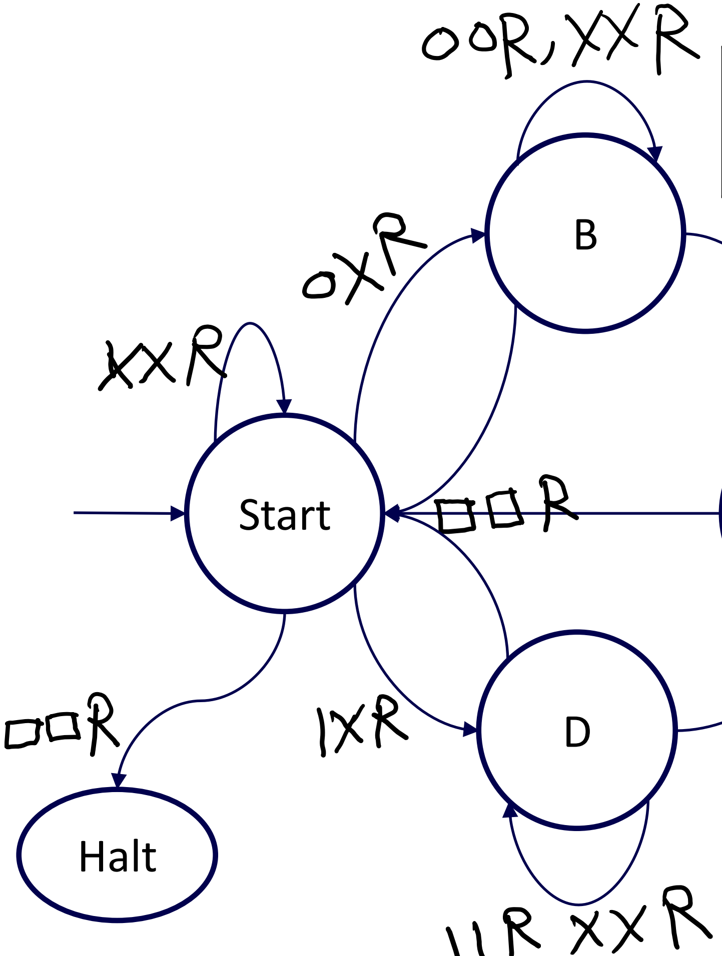
Turing machine - 14/15

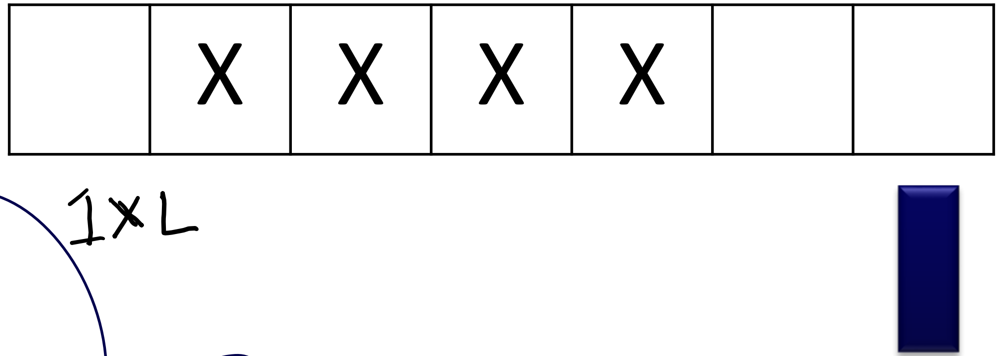


	X	X	X	X	

State: Start

Turing machine - 15/15





State: Halt

Summary

- Wrote an assembly code.
- Von Neumann and Harvard architectures.
- CISC and RISC.
- Turing machine.