Datapath & Control Unit

Computer Architecture
Dr. Guy Tel-Zur

~~8/4/2021~~
עדכון אחרון: 21/4/2022

מטרת ההרצאה

במצגת קצרה זו נעמוד על עקרון תכן מרכזי והוא הפרדה בין מסלול הנתונים לבקר.

נושאי המצגת:

א. החלוקה לבקר ולמסלול נתונים

ב. בעיית דוגמה ראשונה. תרשים לוגי של מסלול הנתונים ותרשים FSM לבקר וכן טבלת מצבים מסכמת

ג. בעיית דוגמה שנייה. כנ"ל עבור חישוב !N

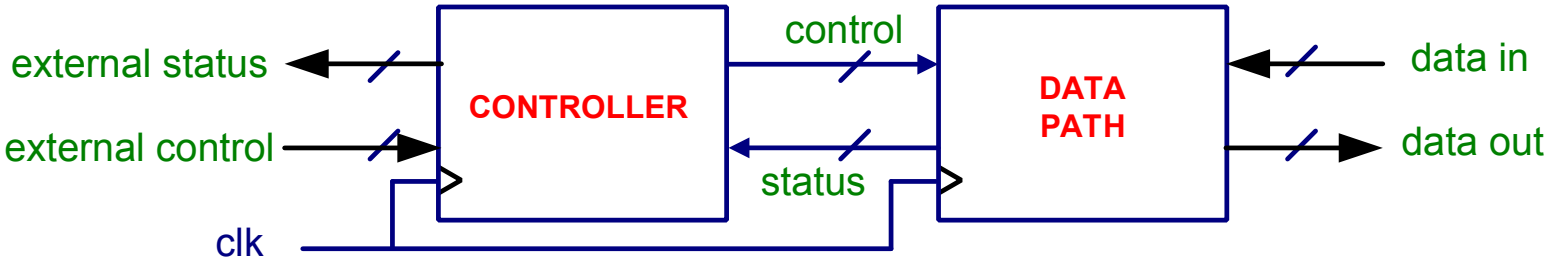
This presentation is based on:

- MIT Open course ware "**Computation Structures**" by Dr. Christopher J. Terman.
- References:
  - MIT Open course ware
  - MIT6.004 2020
  - Slides by Prof. Shmuel Wimer Technion/BIU

In particular we will discuss material from:
https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-004-computation-structures-spring-2017/c9/c9s1/#1

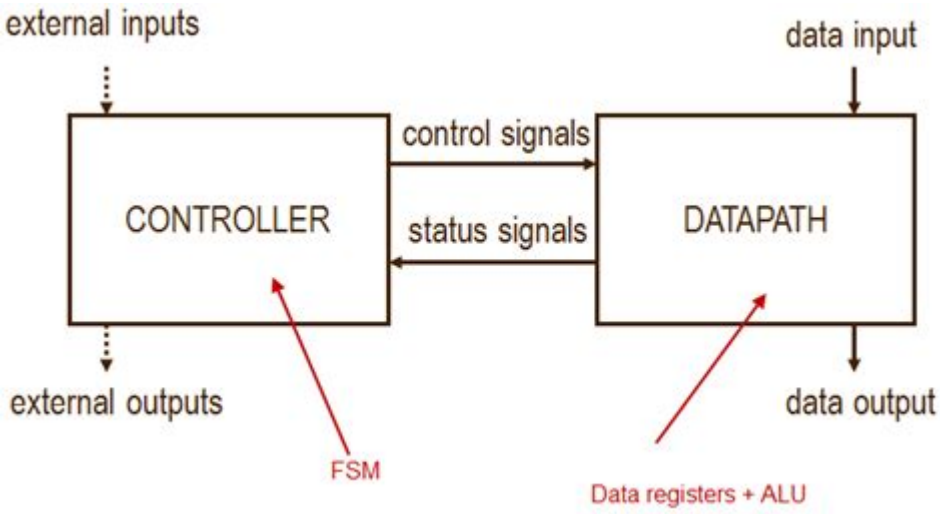
See 9.2.1, 9.2.2 @ MIT open course ware site

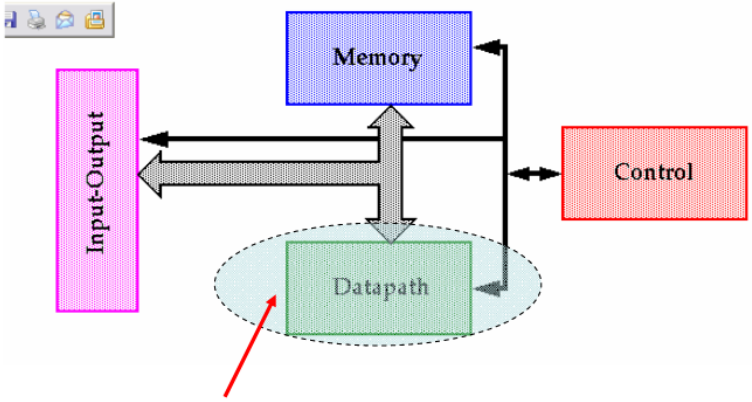
חלוקה לבקר ומסלול נתונים



- מסלול נתונים:
  - תפקיד – מאחסן נתונים, מעבד
    - רכיבים אופייניים – רגיסטרים, מסכם, ALU, בוררים
- בקר:
  - תפקיד – קובע מה יבצע מסלול הנתונים
  - מימוש - מערכת סדרתית כללית, למשל מימוש ע"י ROM ורגיסטר.
- קשר בין החלקים: אותות בקרה וסטטוס
- שני החלקים הם בדרך כלל מערכות סדרתיות
- החלוקה לא יחידה – יש מספר אפשרויות

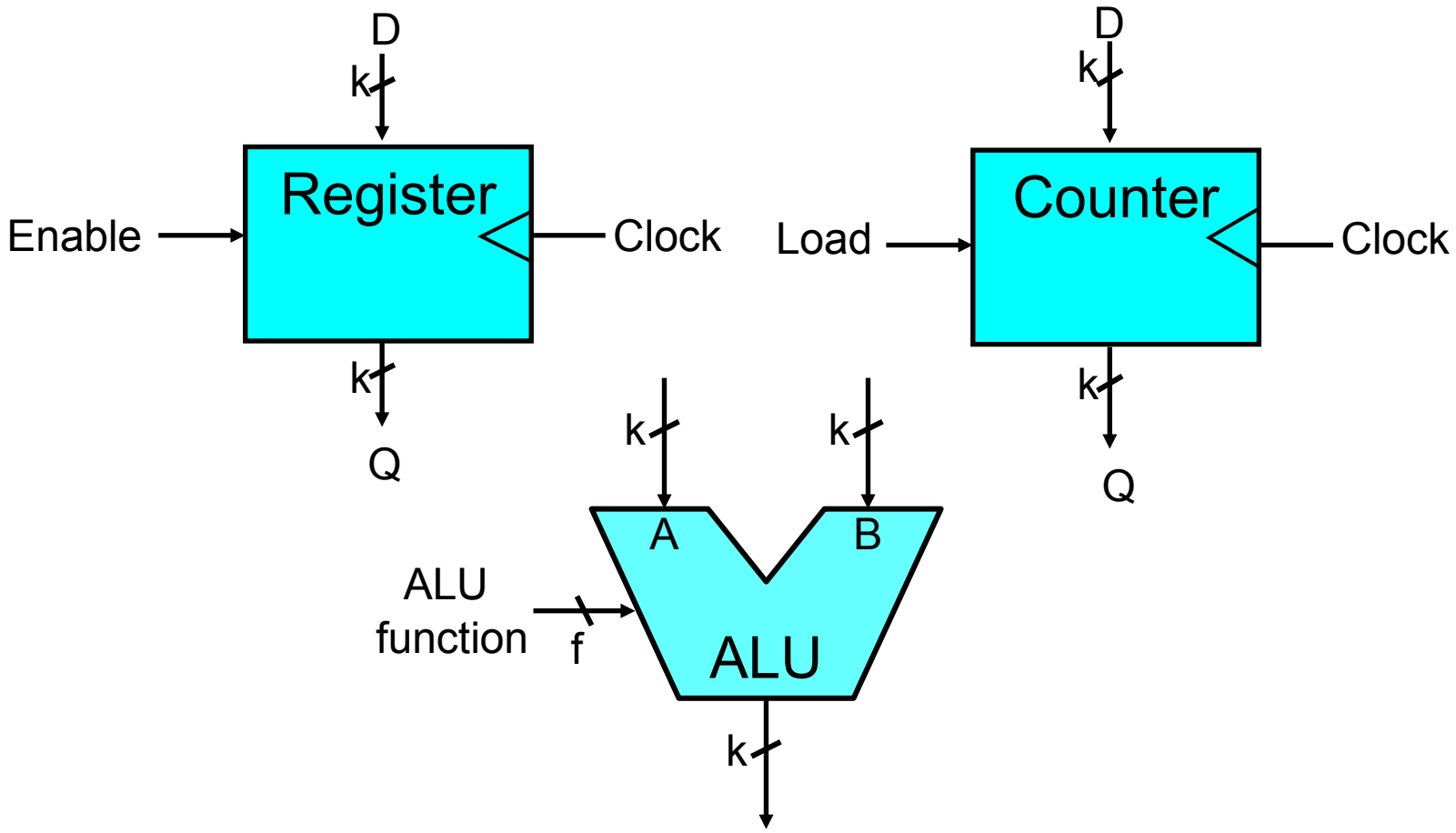
בהתייחס לסעיף האחרון בשקף הקודם, להלן תרשימים אלטרנטיביים

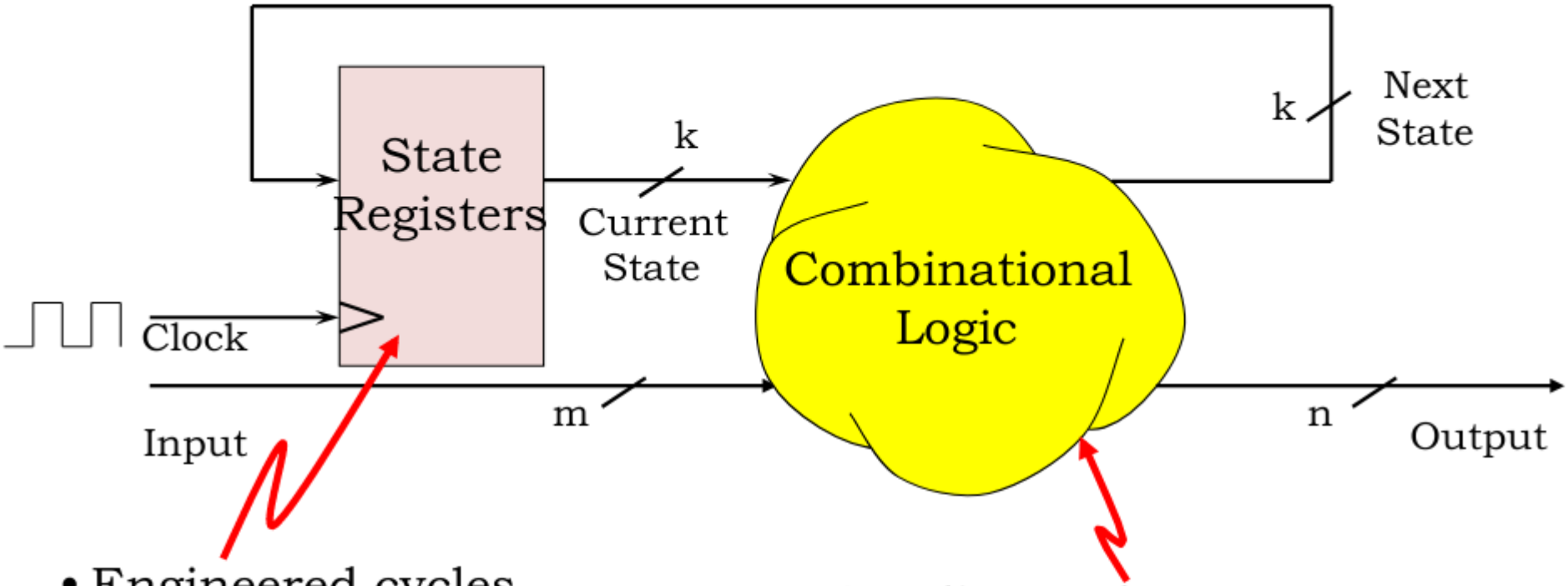




- We shall concentrate first on this.

רכיבים סטנדרטיים למסלול נתונים פשוט





- Engineered cycles
- Works only if dynamic discipline obeyed
- Remembers k bits for a total of 2^k unique combinations

- Acyclic graph
- Obeys static discipline
- Can be exhaustively enumerated by a truth table of 2^{k+m} rows and k+n output columns

דוגמת תכנון: מסכם סדרות

אורך הסדרה

- נדרש: מעגל לוגי המקבל סדרה טורית

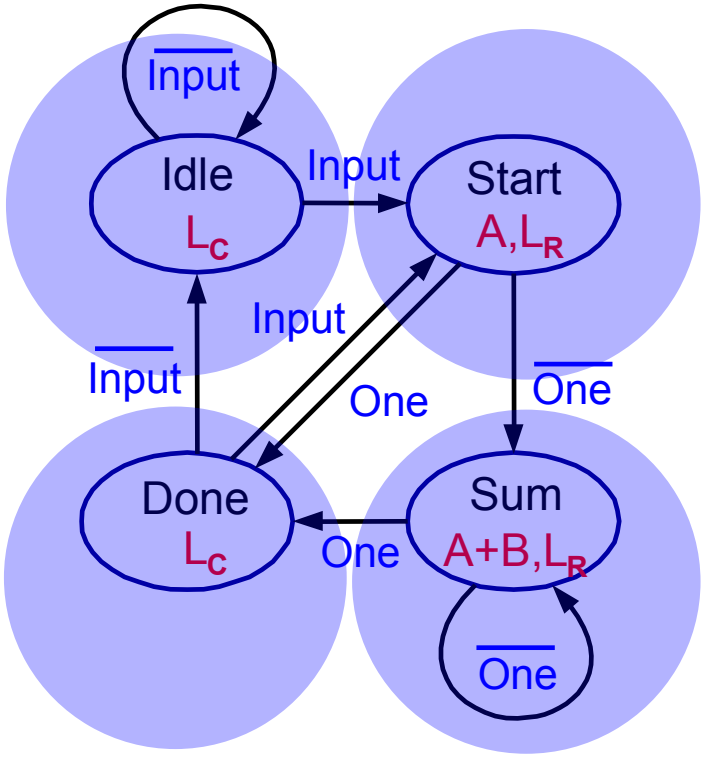
$$\ldots, 0, 0, n, x_1, x_2, \ldots, x_n, m, y_1, y_2, \ldots, y_m, 0, 0, \ldots$$

- ומחשב את הסכומים $\sum_n x_i, \sum_m y_i, \ldots$
- שלבי התכנון:

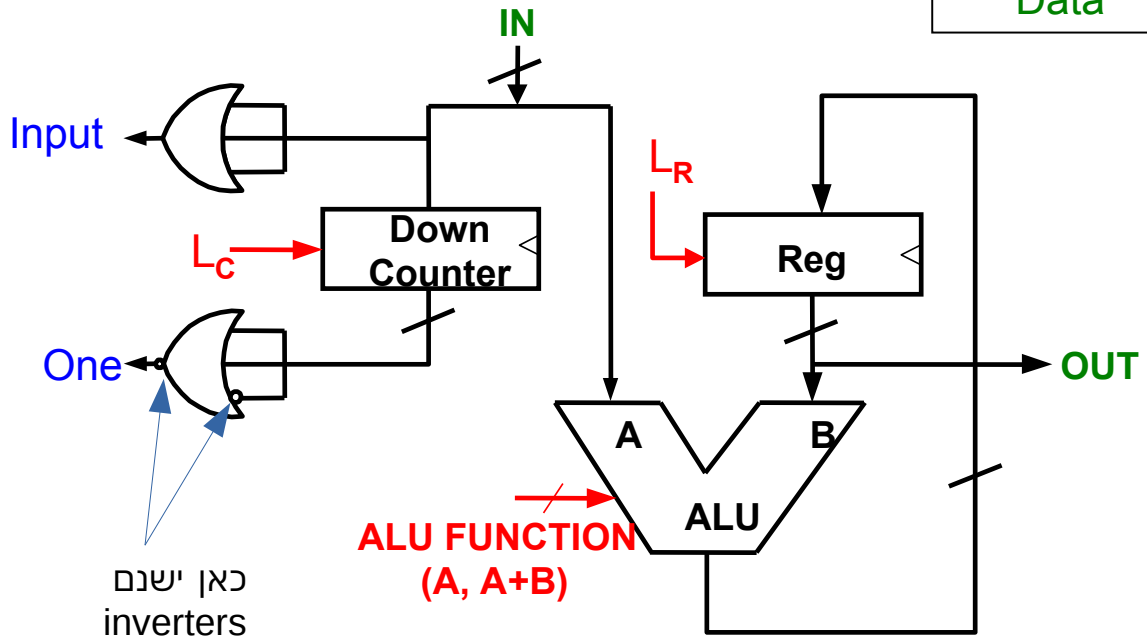
1. תכנון מסלול נתונים בעזרת רכיבים סטנדרטיים (מסכם, רגיסטר, מונה)
2. הגדרת קווי בקרה וסטטוס – דרישות של מסלול הנתונים מהבקר
3. תכנון בקר בעזרת רכיבים סטנדרטיים (רגיסטר, מסכם, מונה)

מסכם סדרות

מקרא:
Control
Status
Data



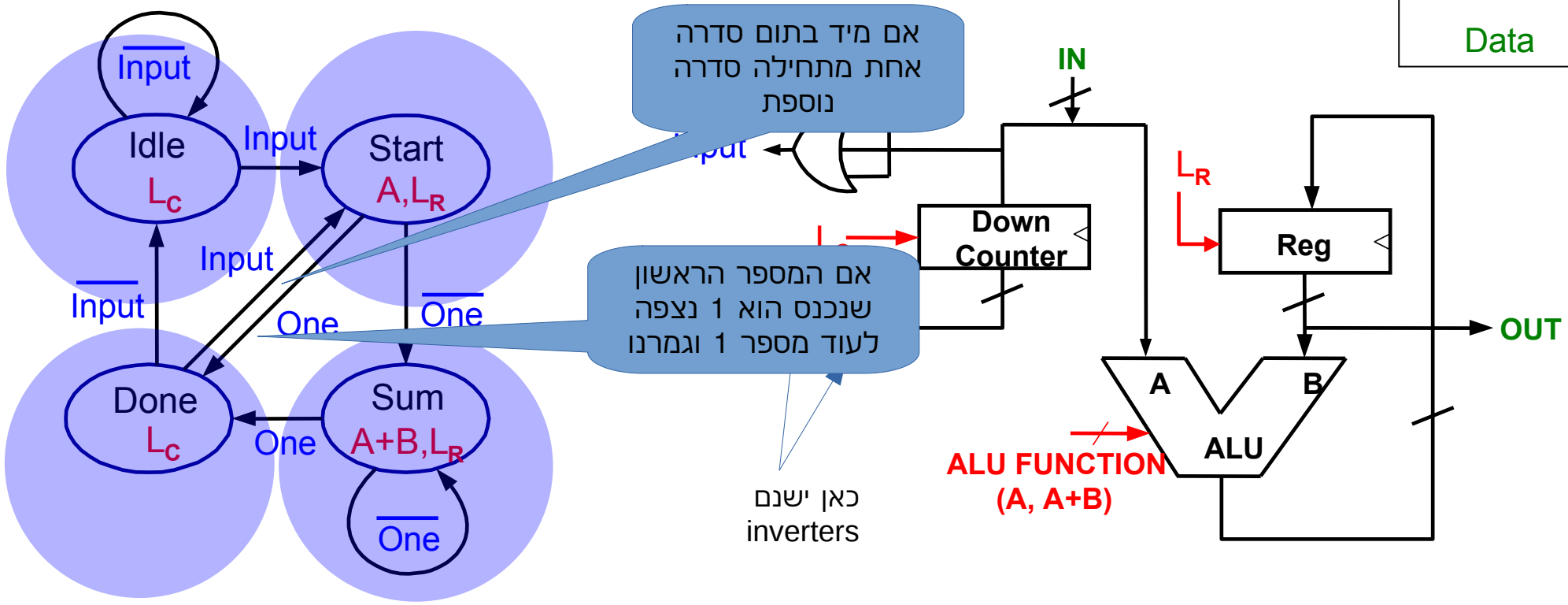
בקר



מסלול נתונים

מסכם סדרות

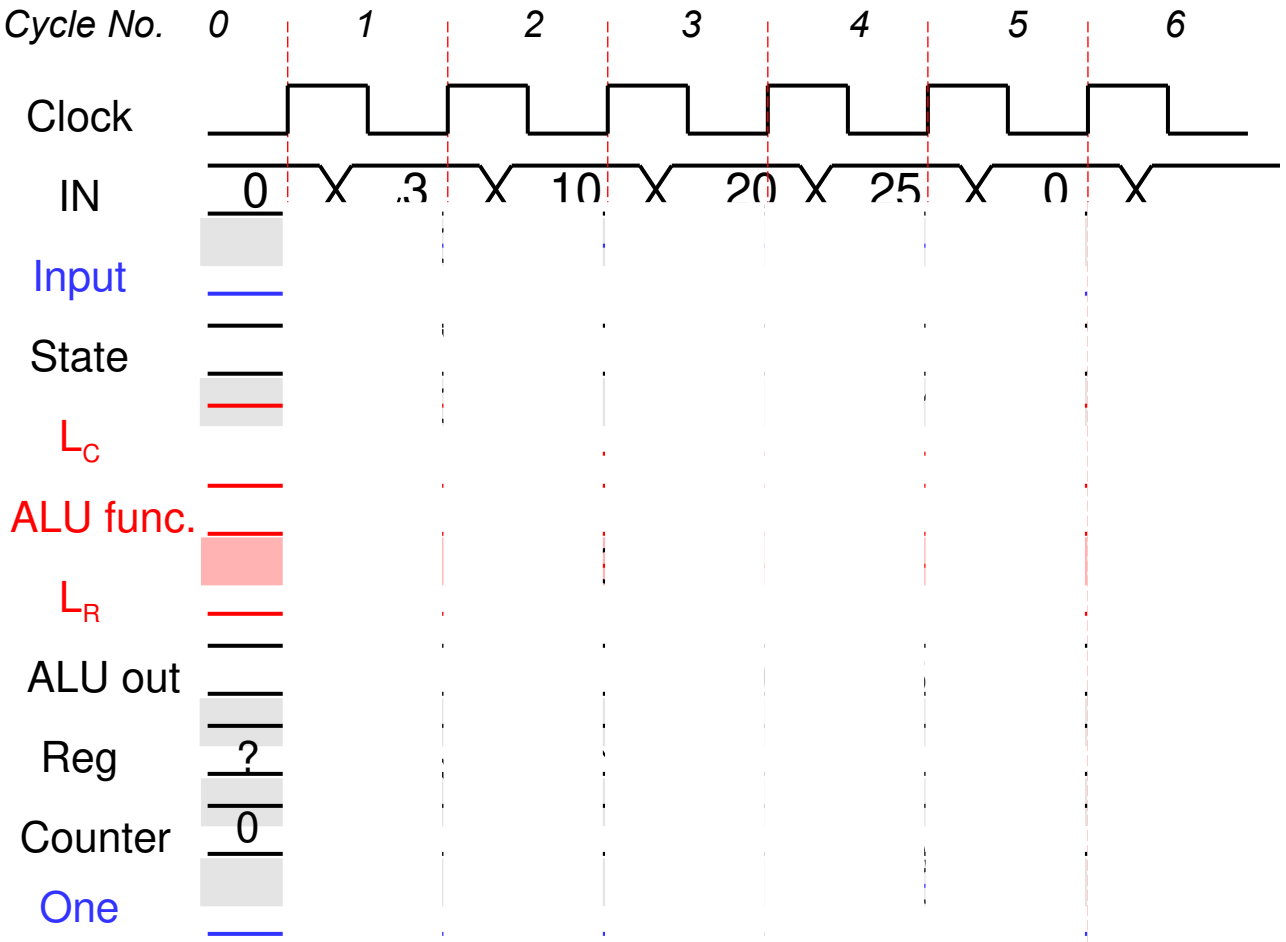
מקרא:
Control
Status
Data

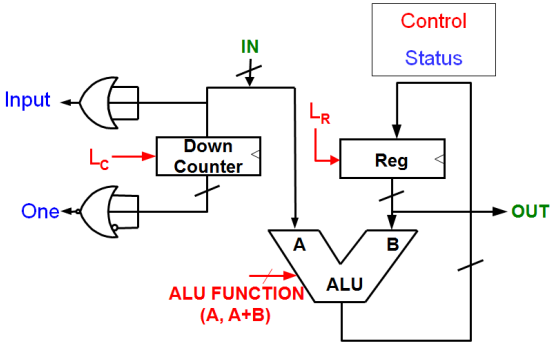


בקר

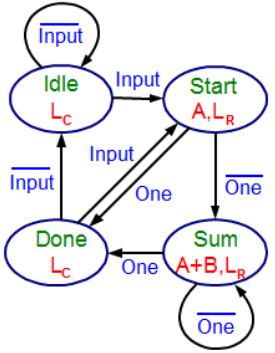
מסלול נתונים

דוגמא לסיכום סדרת 3 מספרים





מסלול נתונים



בקר

טבלת המצבים

S	Status		Control			S’	Out
Current state	input	one	L_c	L_R	ALU Func	Next state	Register output
IDLE	0	x	1	0	-	IDLE	x
IDLE	1	x	1	0	-	START	x
START	x	0	0	1	A	SUM	A+B
START	x	1	0	1	A	DONE	A
SUM	x	0	0	1	A+B	SUM	A+B
SUM	x	1	0	1	A+B	DONE	A+B
DONE	0	x	1	0	-	IDLE	A+B
DONE	1	x	1	0	-	START	A+B

מדוע החלוקה לבקר ומסלול נתונים יעילה?

העיקרון - הפרדת מערכת סדרתית עם מספר מצבים גדול לשתי מערכות סידרתיות:

- **באחת (DP=DataPath) מספר מצבים גדול (רוב המצבים)**

  התכנון אינו בכלים מסורתיים ל-FSM

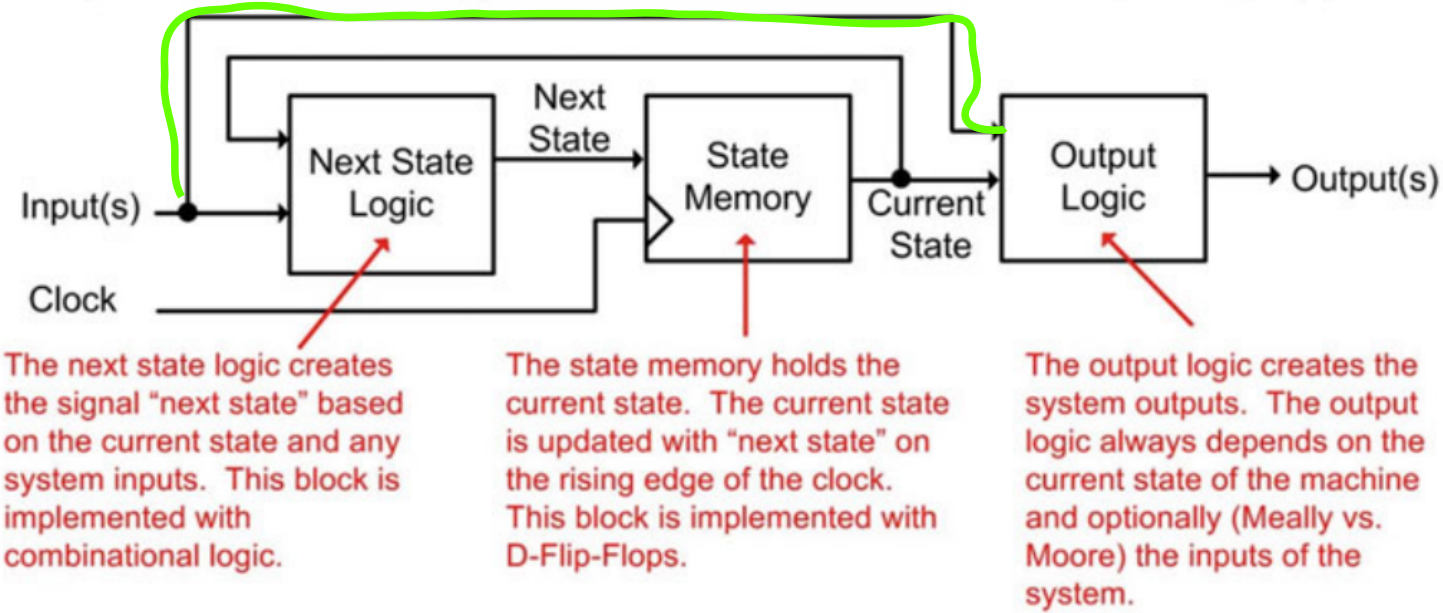
- **באחרת (בקר) מספר המצבים קטן**

  ומתבצע תכנון FSM רגיל

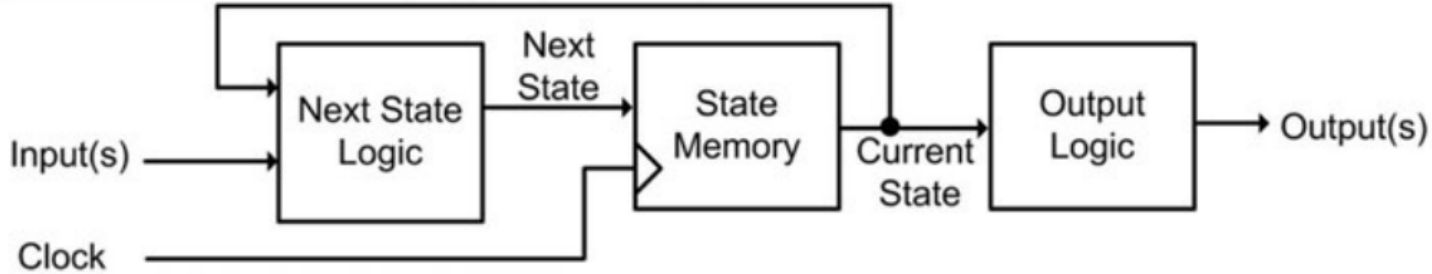
- אם במסלול הנתונים n מצבים ובבקר k מצבים, כמה מצבים יש במכונה ללא פירוק?
- אם ב-FSM יש K פליפ-פלופים, מה סיבוכיות התכנון (כמה מצבים יש)?

Main Components of a Finite State Machine

Mealy Machine – The output(s) depend on both the current state and system input(s).



Moore Machine – The output(s) depends only on the current state.

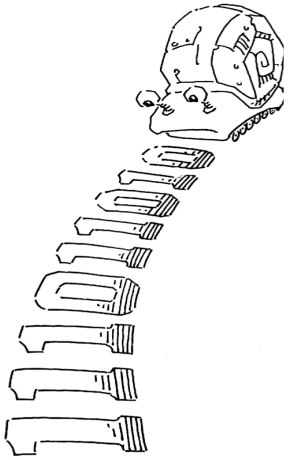


Credit: Brock J. LaMeres, "INTRODUCTION TO LOGIC CIRCUITS & LOGIC DESIGN WITH VERILOG"

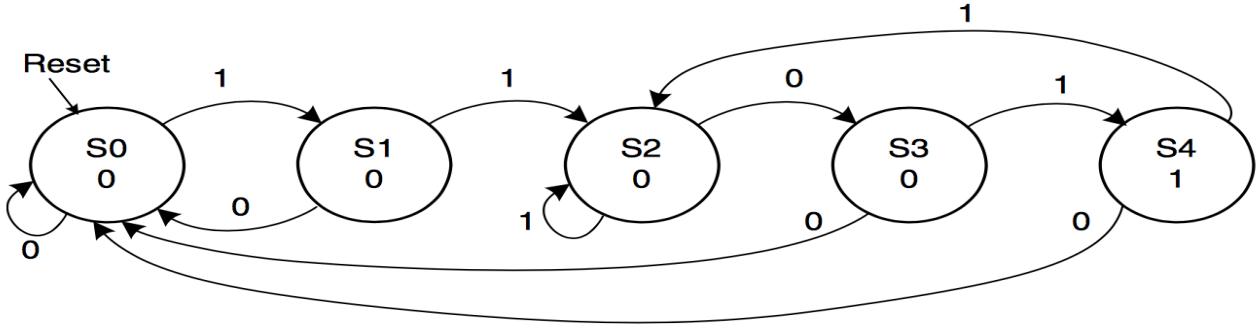
FSM Example: Smiling Snail (H&H Ch. 3 and Mutlu L07, 2022)

https://safari.ethz.ch/digitaltechnik/spring2022/lib/exe/fetch.php?media=digitaldesign-comparch-2022-lecture7-hdl-verilog-afterlecture.pdf

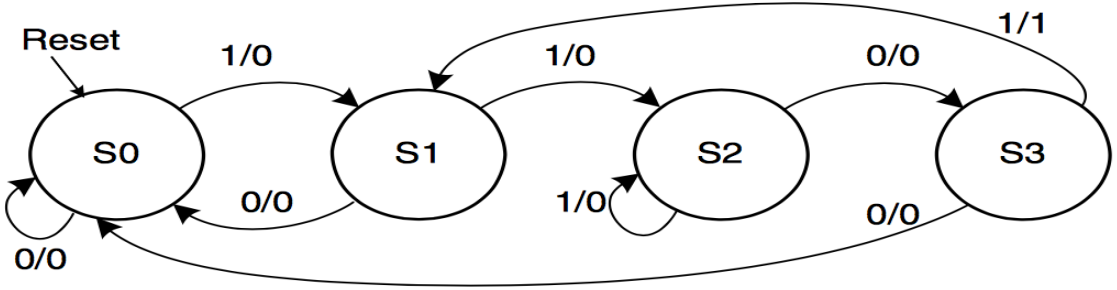
- A snail crawls down a paper tape with 1's and 0's on it
- The snail smiles whenever the last four digits it has crawled over are 1101
- Design Moore and Mealy FSMs of the snail's brain



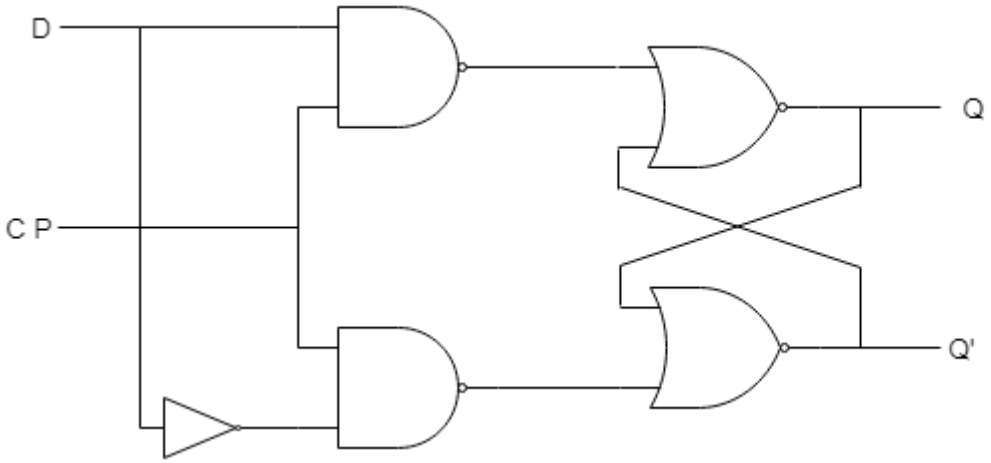
Moore



Mealy

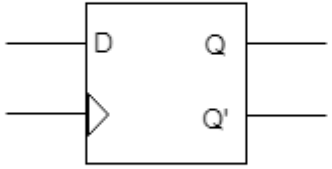


D flip flop תזכורת



(a) Logic diagram with Nand gates

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



(b) Graphic Symbol

fig. Clocked D flip flop

BASIS OF COMPARISON	MEALY MACHINE	MOORE MACHINE
Description	Mealy machine changes its output based on its current input and present state.	Output of Moore machine only depends on its current state and not on the current input.
States	Mealy machine will have same or fewer states than Moore machine.	It has more states than Mealy machine.
Output	Output is placed on transition.	Output is placed on transition.
Value Of Output Function	The value of the output function is a function of the transitions and the changes, when the input logic on the present state is done.	The value of the output function is a function of the current state and the changes at the clock edges, whenever state changes occur.
Reaction To Inputs	Mealy machines react faster to inputs. They generally react in the same clock cycle.	More logic is required to decode the output resulting in more circuit delays. They generally react one clock cycle later.
Output And State	Asynchronous output generation through the state changes synchronous to the clock.	Both output and state change synchronous to the clock edge.
States Requirement	Generally requires fewer states for synthesis.	Generally requires more states for synthesis.
Hardware Requirement	Requires less hardware to design.	More hardware is required to design.
Counter	A counter is not a Mealy machine.	A counter is a Moore machine.
Design	Not necessarily easy to design.	Easy to design.

השוואה בין שני המודלים.

לא להתעכב על זה

Example: Factorial

factorial(N) = N! = N*(N-1)*…*1

C:

```
int a = 1;
int b = N;
do {
  a = a * b;
  b = b - 1;
} while (b != 0)
```

```
initially:     a =   1, b = 5
after iter 1:  a =   5, b = 4
after iter 2:  a =  20, b = 3
after iter 3:  a =  60, b = 2
after iter 4:  a = 120, b = 1
after iter 5:  a = 120, b = 0
Done!
```

Example: Factorial

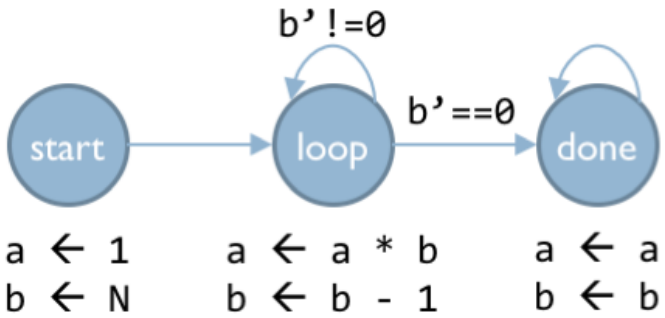
factorial(N) = N! = N*(N-1)*…*1

C:

```
int a = 1;
int b = N;
do {
  a = a * b;
  b = b - 1;
} while (b != 0)
```

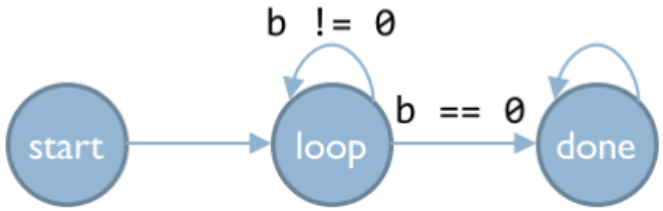
```
start:  a ←   1, b ← 5
loop:   a ←   5, b ← 4
loop:   a ←  20, b ← 3
loop:   a ←  60, b ← 2
loop:   a ← 120, b ← 1
loop:   a ← 120, b ← 0
done:
```

High-level FSM:



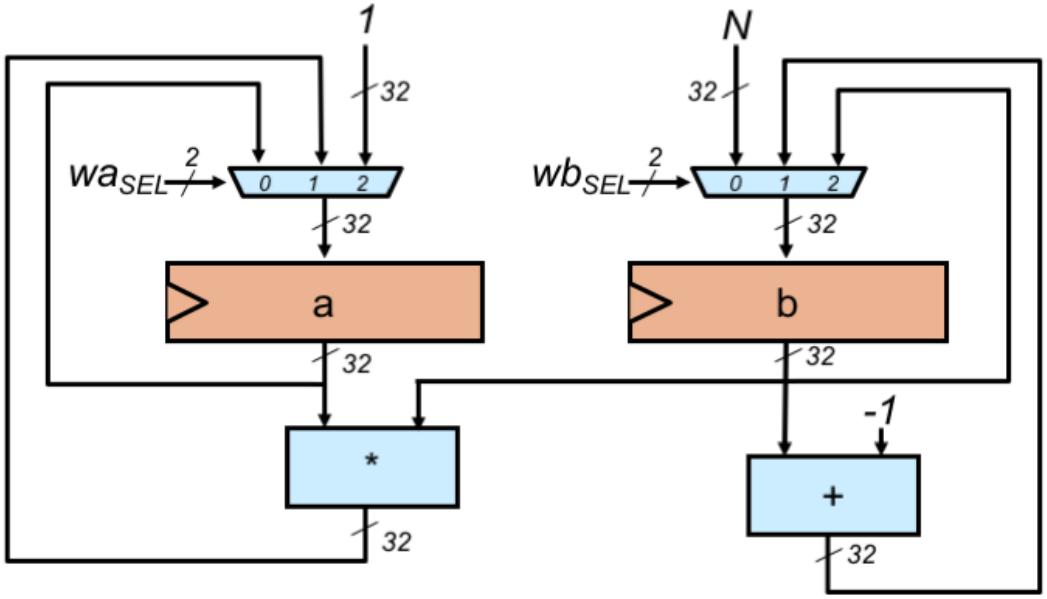
- Helpful to translate into hardware
- D-registers (a, b)
- 2-bits of state (start, loop, done)
- Boolean transitions (b'==0, b'!=0)
- Register assignments in states (e.g., a ← a * b)

Datapath for Factorial

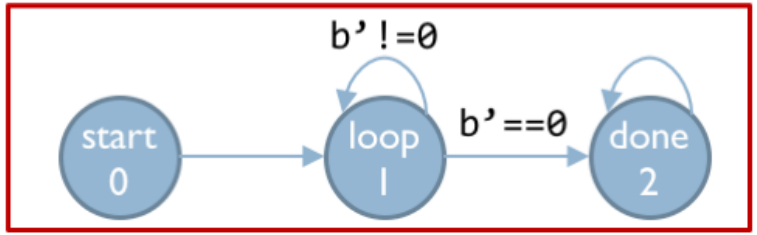



```
a ← 1        a ← a * b        a ← a
b ← N        b ← b - 1        b ← b
```

- Draw registers
- Draw combinational circuit for each assignment
- Connect to input muxes



Control FSM for Factorial

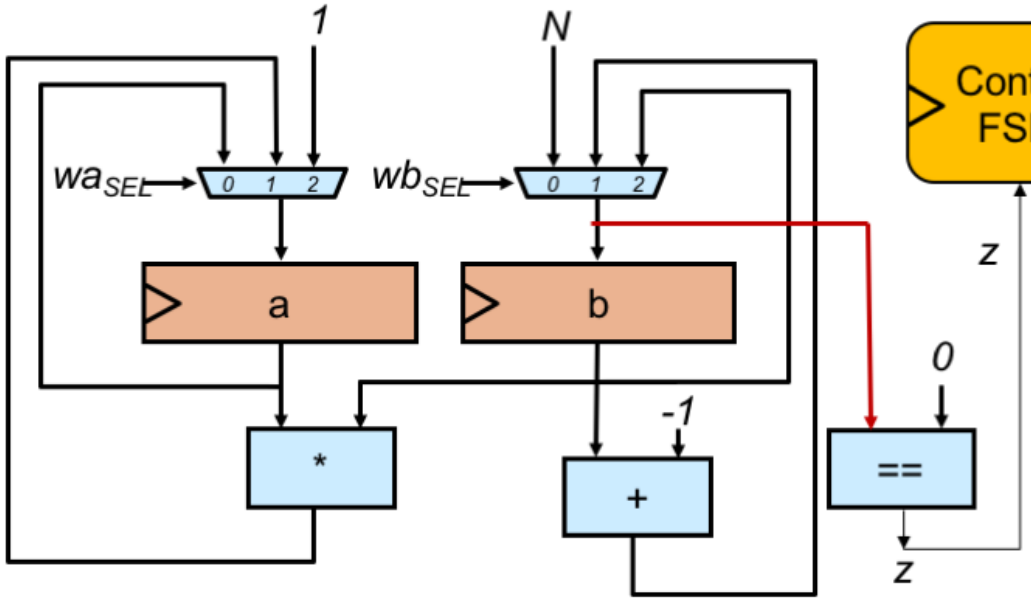


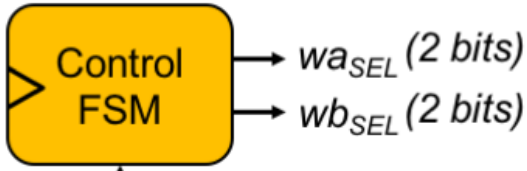
a ← 1 b ← N

a ← a * b b ← b - 1

a ← a b ← b

- Draw combinational logic for transition conditions
- Implement control FSM:
  - States: High-level FSM states
  - Inputs: Transition logic outputs
  - Outputs: Mux select signals

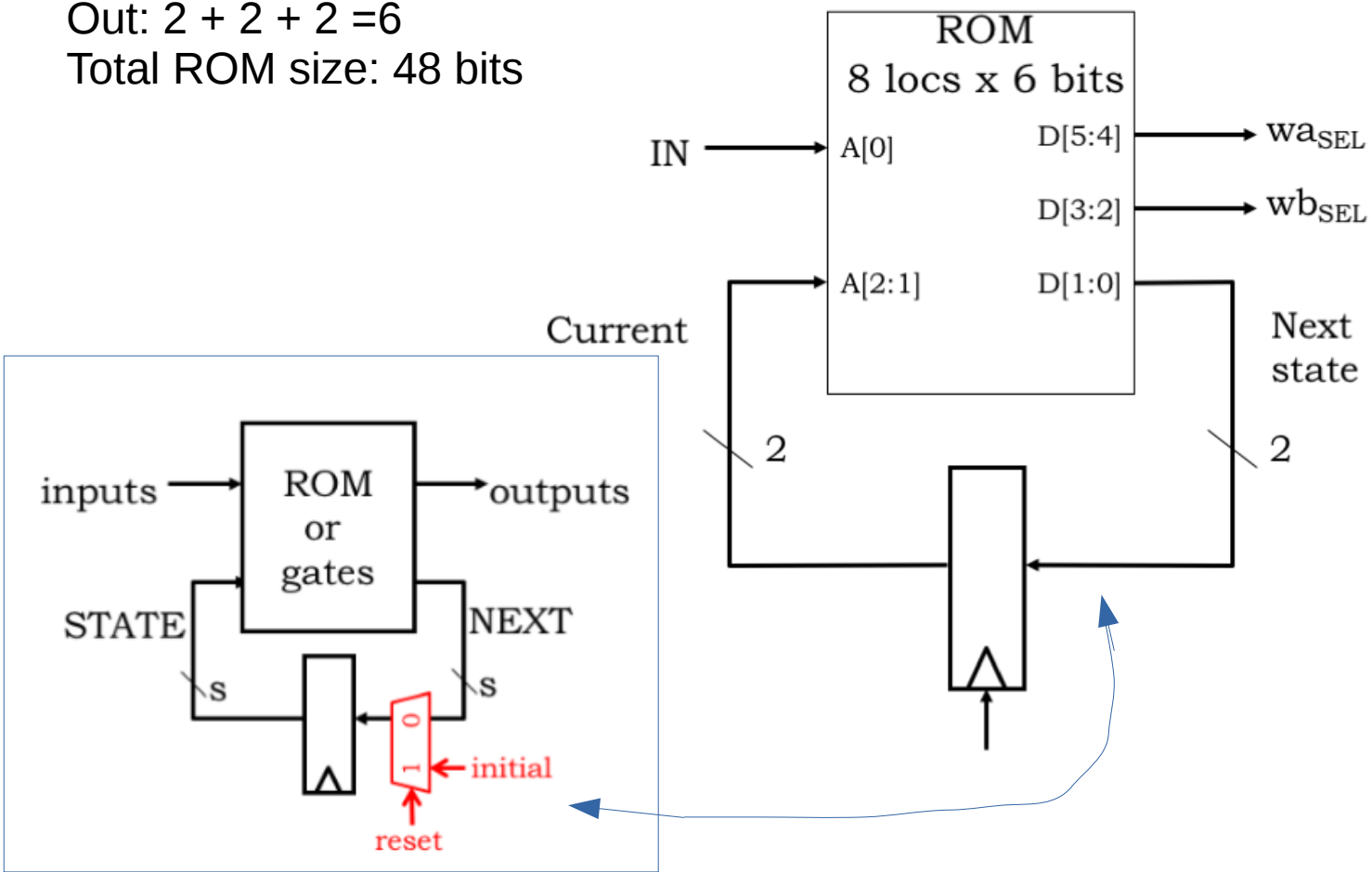




S	Z	wa_{SEL}	wb_{SEL}	S'
00	0	10	00	01
00	1	10	00	01
01	0	01	01	01
01	1	01	01	10
10	0	00	10	10
10	1	00	10	10

Control FSM Hardware

In: 2^3 = 8
Out: 2 + 2 + 2 =6
Total ROM size: 48 bits



ROM contents

A[2:0]	D[5:0]
000	10 00 01
001	10 00 01
010	01 01 01
011	01 01 10
100	00 10 10
101	00 10 10

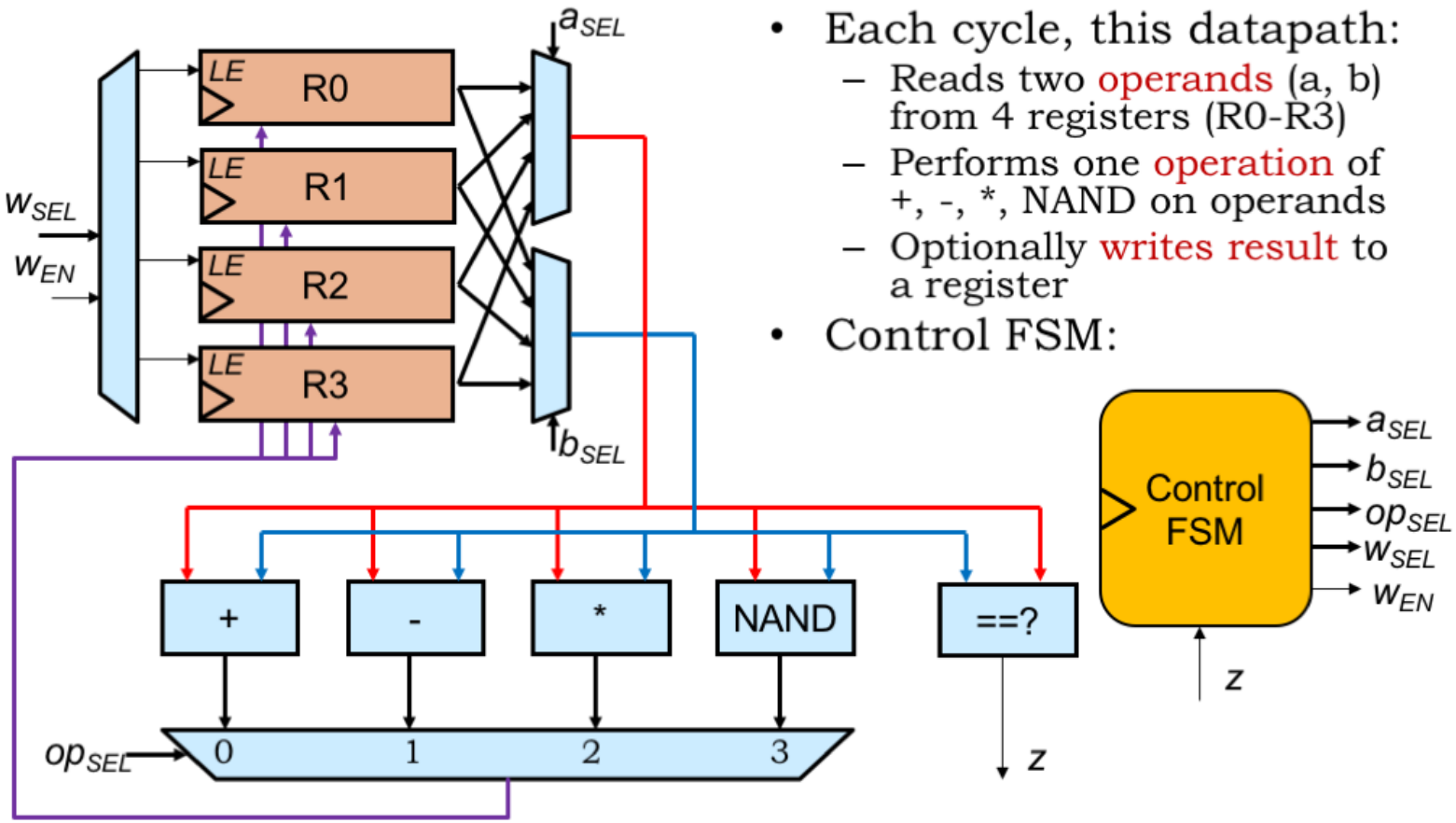
So Far: Single-Purpose Hardware

- Problem→ Procedure (High-level FSM)→ Implementation

- Systematic way to implement high-level FSM as a datapath + control FSM
  - Is this implementation an FSM itself?
  - If so, can you draw the truth table?

- How should we generalize our approach so we can solve many problems with one set of hardware?
  - More storage for operands and results
  - A larger repertoire of operations
  - General-purpose datapath

A Simple Programmable Datapath



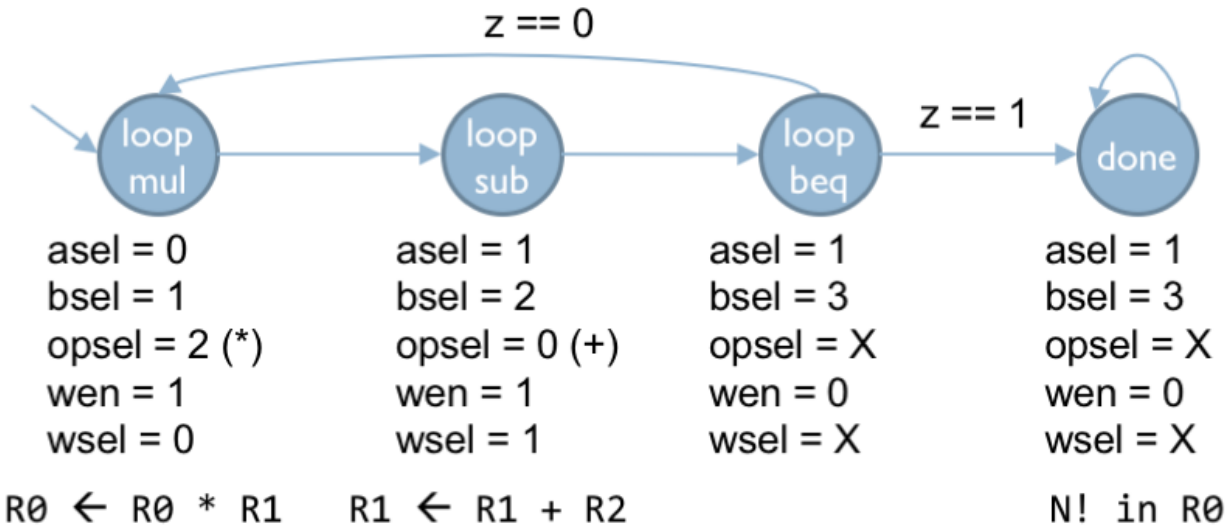
- Each cycle, this datapath:
  - Reads two operands (a, b) from 4 registers (R0-R3)
  - Performs one operation of +, -, *, NAND on operands
  - Optionally writes result to a register
- Control FSM:

A Control FSM for Factorial

- Assume initial register contents:

```
R0 value = 1
R1 value = N
R2 value = -1
R3 value = 0
```

- Control FSM:

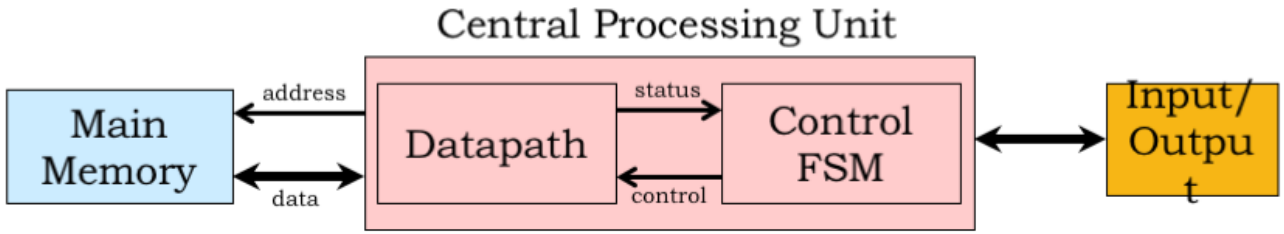


New Problem → New Control FSM

- You can solve many more problems with this datapath!
  - Exponentiation, division, square root, …
  - But nothing that requires more than four registers
- By designing a control FSM, we are programming the datapath
- Early digital computers were programmed this way!
  - ENIAC (1943):
    - First general-purpose digital computer
    - Programmed by setting huge array of dials and switches
    - Reprogramming it took about 3 weeks

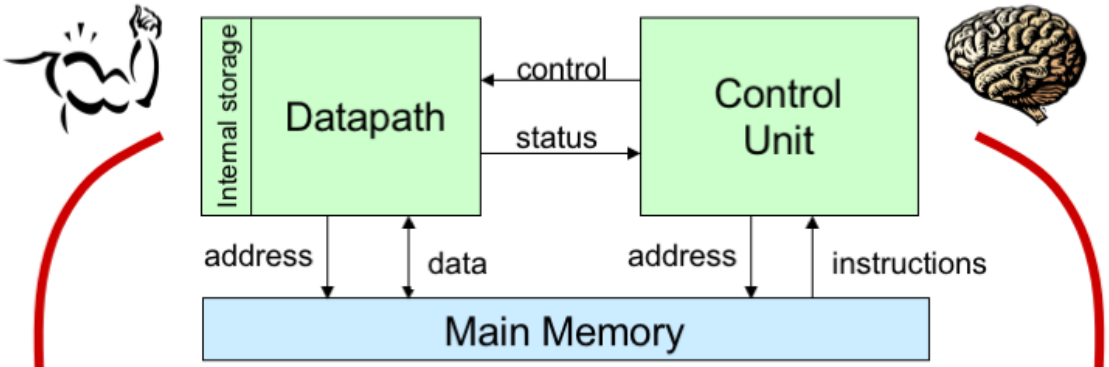
The von Neumann Model

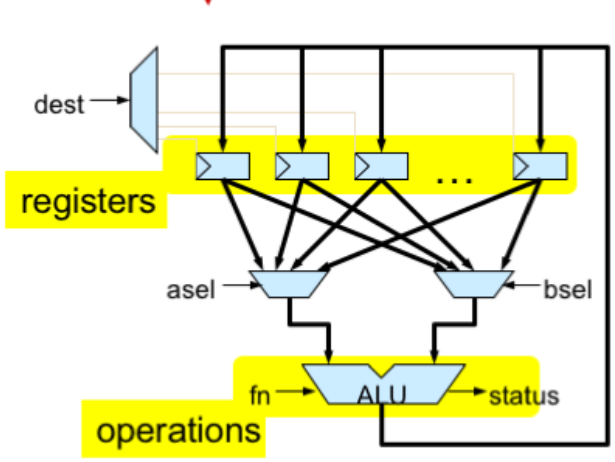
- Many approaches to build a general-purpose computer. Almost all modern computers are based on the von Neumann model (John von Neumann, 1945)
- Components:

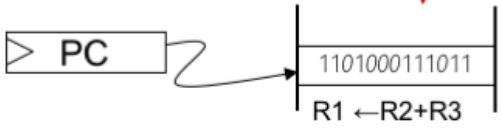


- Central processing unit:
  Performs operations on values in registers & memory
- Main memory:
  Array of W words of N bits each
- Input/output devices to communicate with the outside world

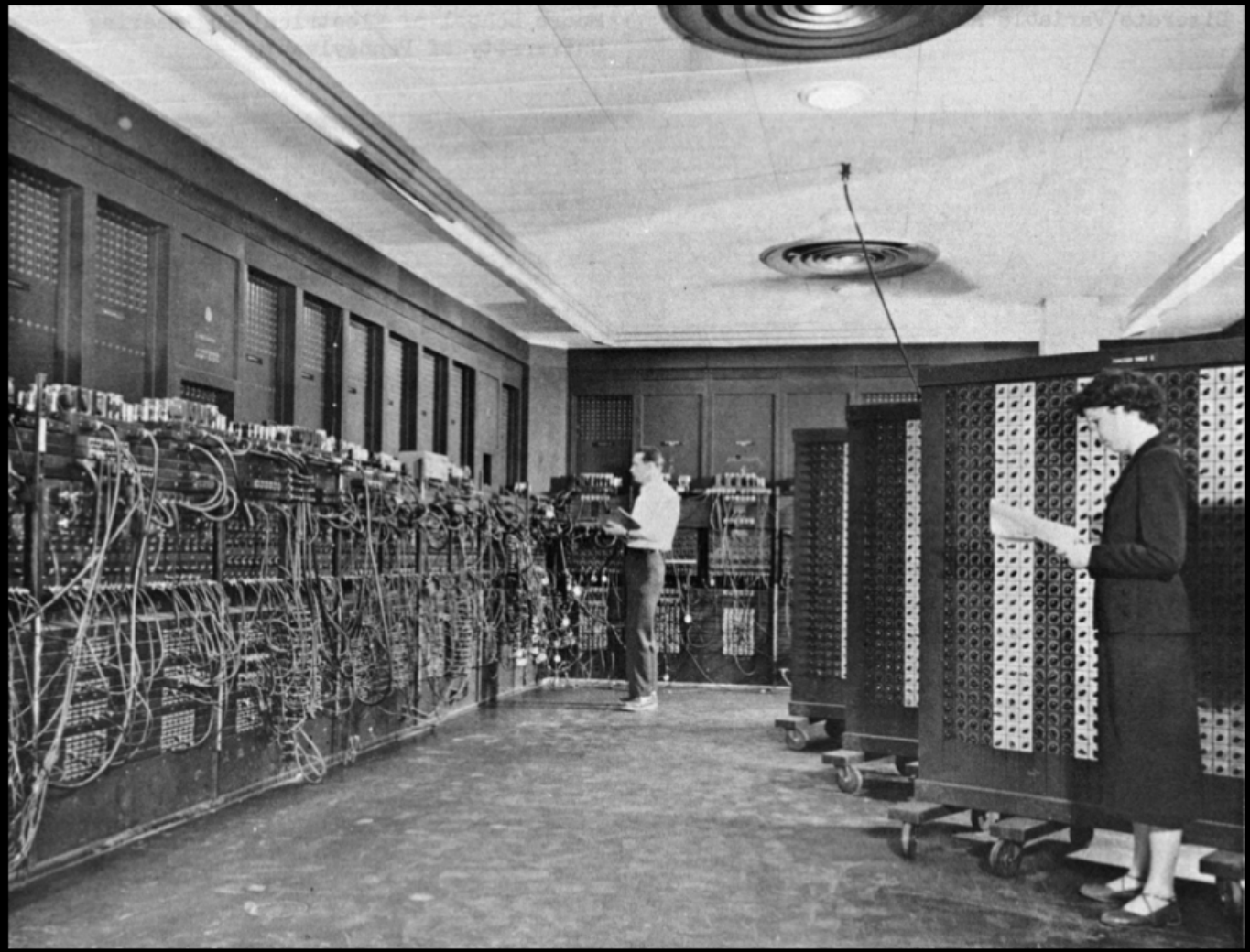
Anatomy of a von Neumann Computer



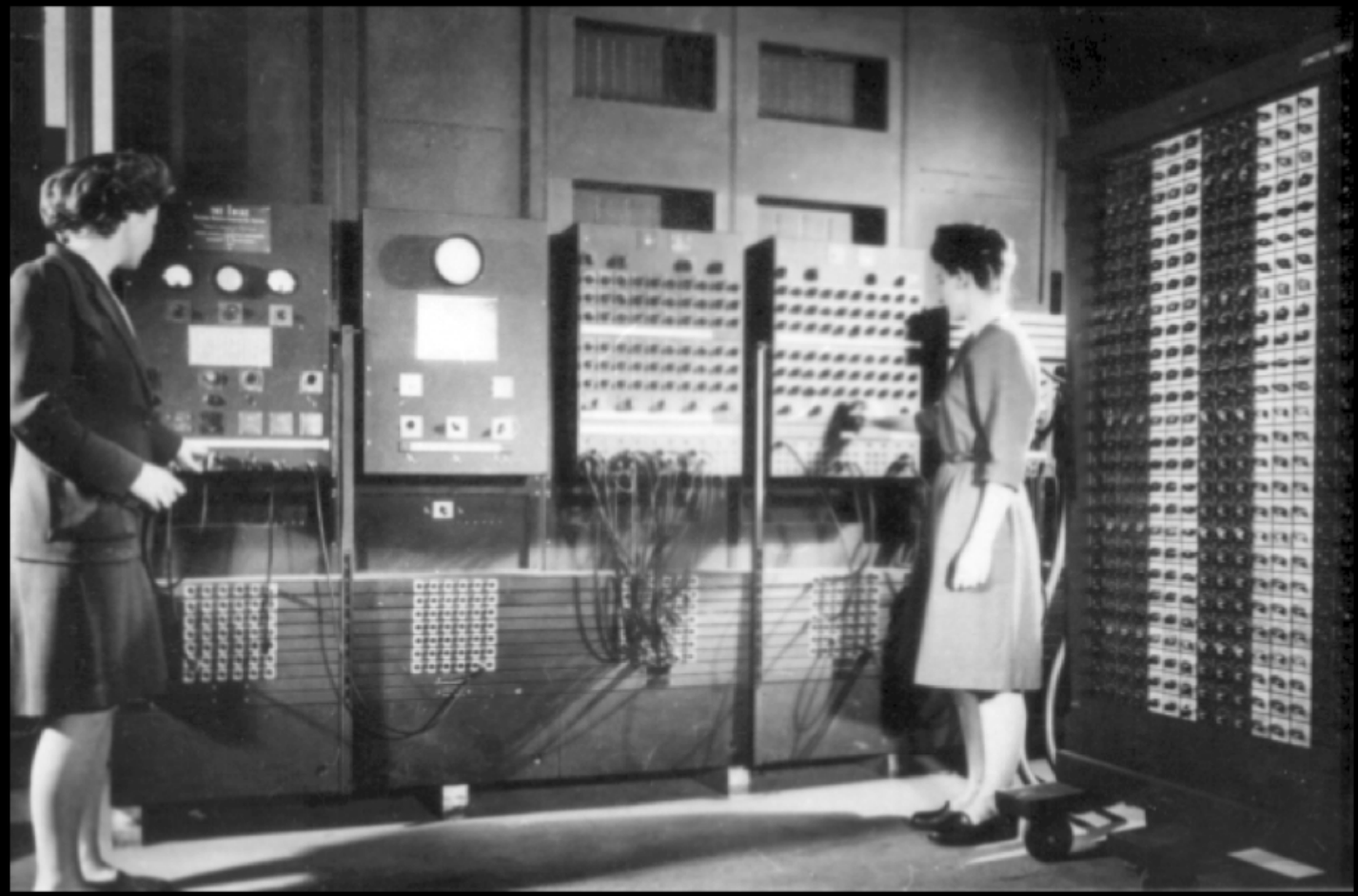




- Instructions coded as binary data
- Program Counter or PC: Address of the instruction to be executed
- Logic to translate instructions into control signals for datapath



"Eniac" by Unknown - U.S. Army Photo.



U.S. Army Photo.

עד כאן מצגת זו

לקריאה נוספת אודות FSM ראו סעיף 3.4 ב- H&H