A working computer from Nand and DFF

Made by: Vitaly Okolelov

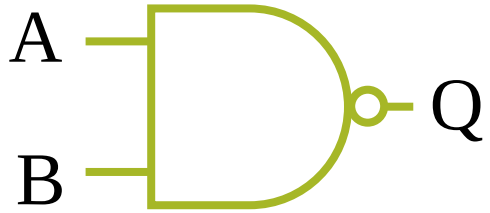
24/02/2021

About the project

1. Computer hardware
   - University individual project
   - Hardware Descriptive Language
2. Assembler
   - Individual project
   - C++
3. Nand & D Flip-flop calculator
   - Part of University team project
   - C++

Nand

What is Nand?

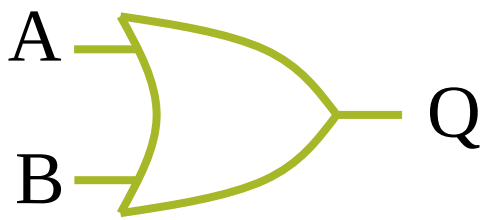


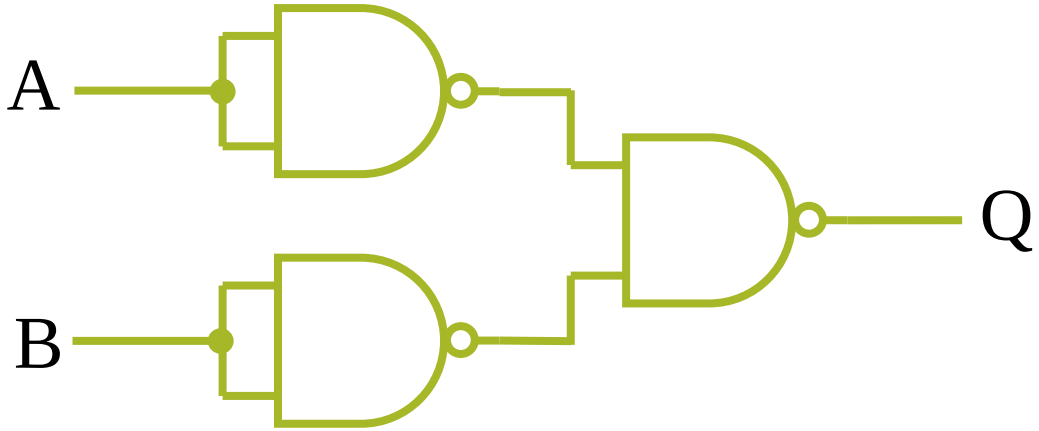
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

Functional completeness

Any logic function can be made from Nands!

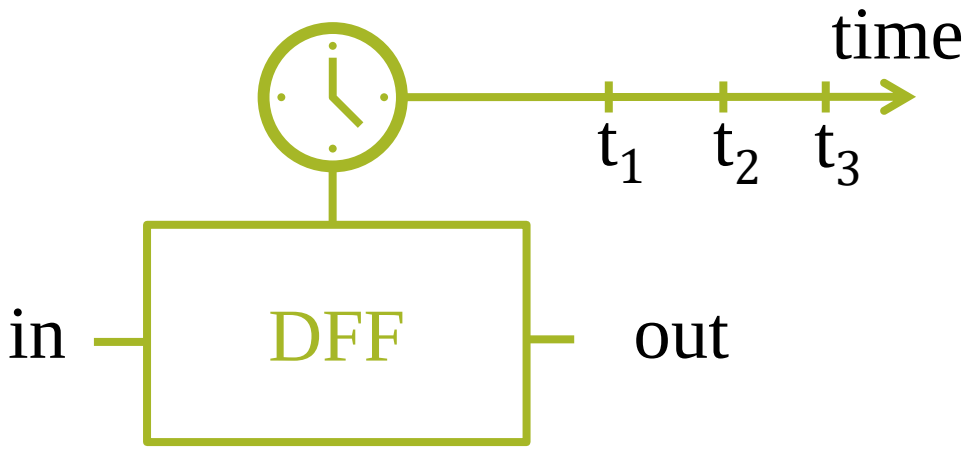
Example: Or





D Flip-Flop

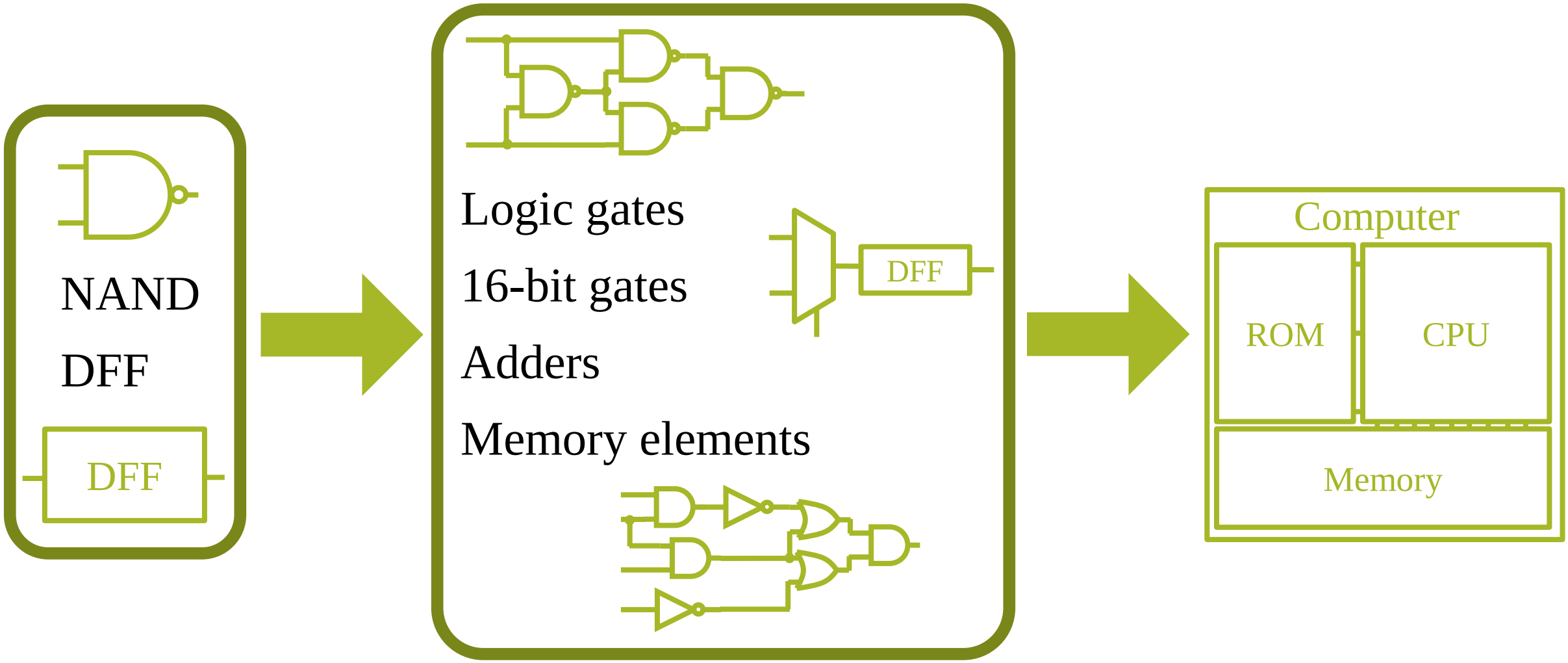
What is D Flip-Flop (DFF)?



$$out(t) = in(t-1)$$

D Flip-Flop can be constructed from NANDs!

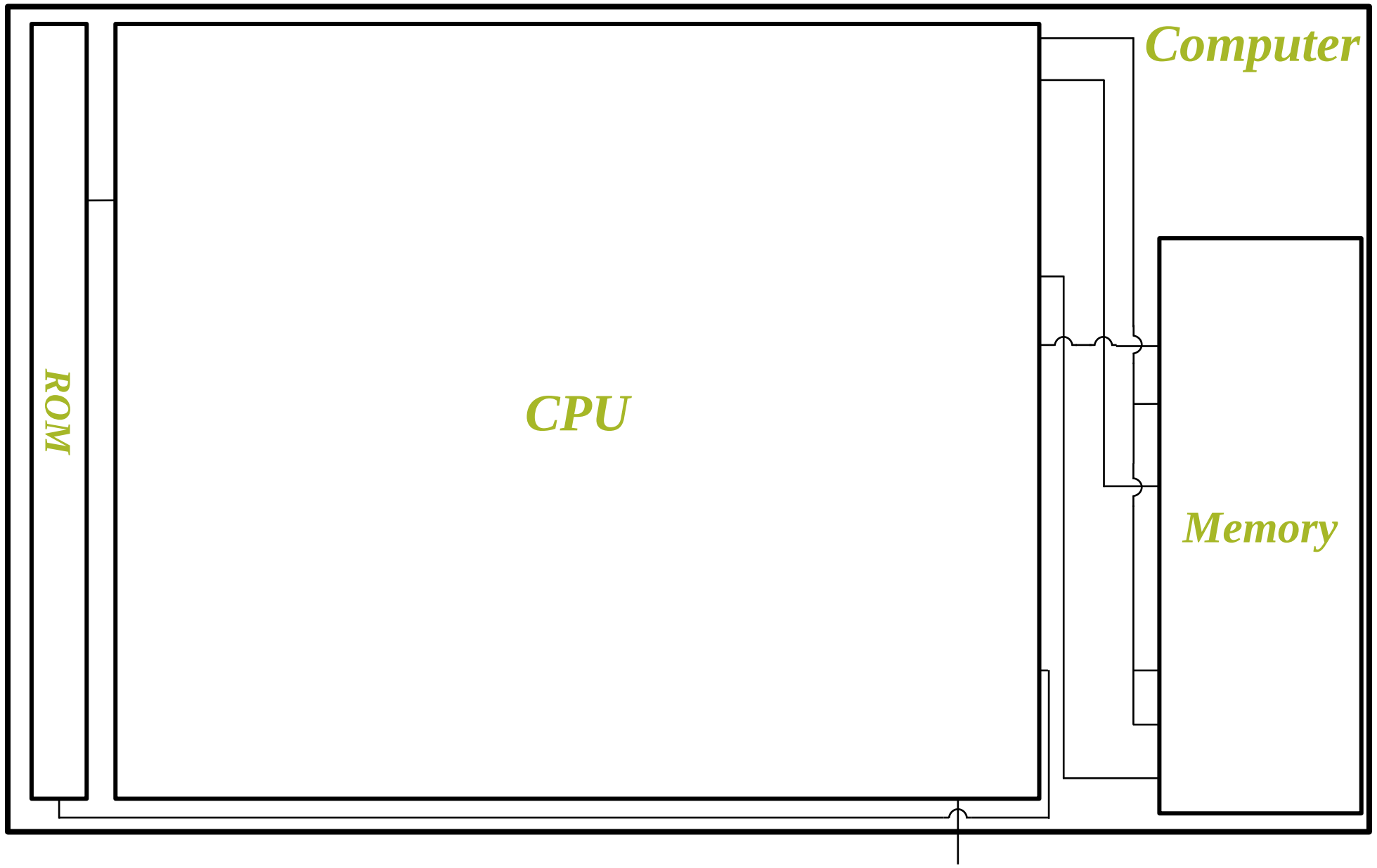
Implementing computer



Computer Diagram

- Memory
- ROM
- CPU

2,104,280 Nands
254,016 DFFs
in this design!



No internal structure shown here, can be provided only by request.

Memory

Random Access Memory (RAM)

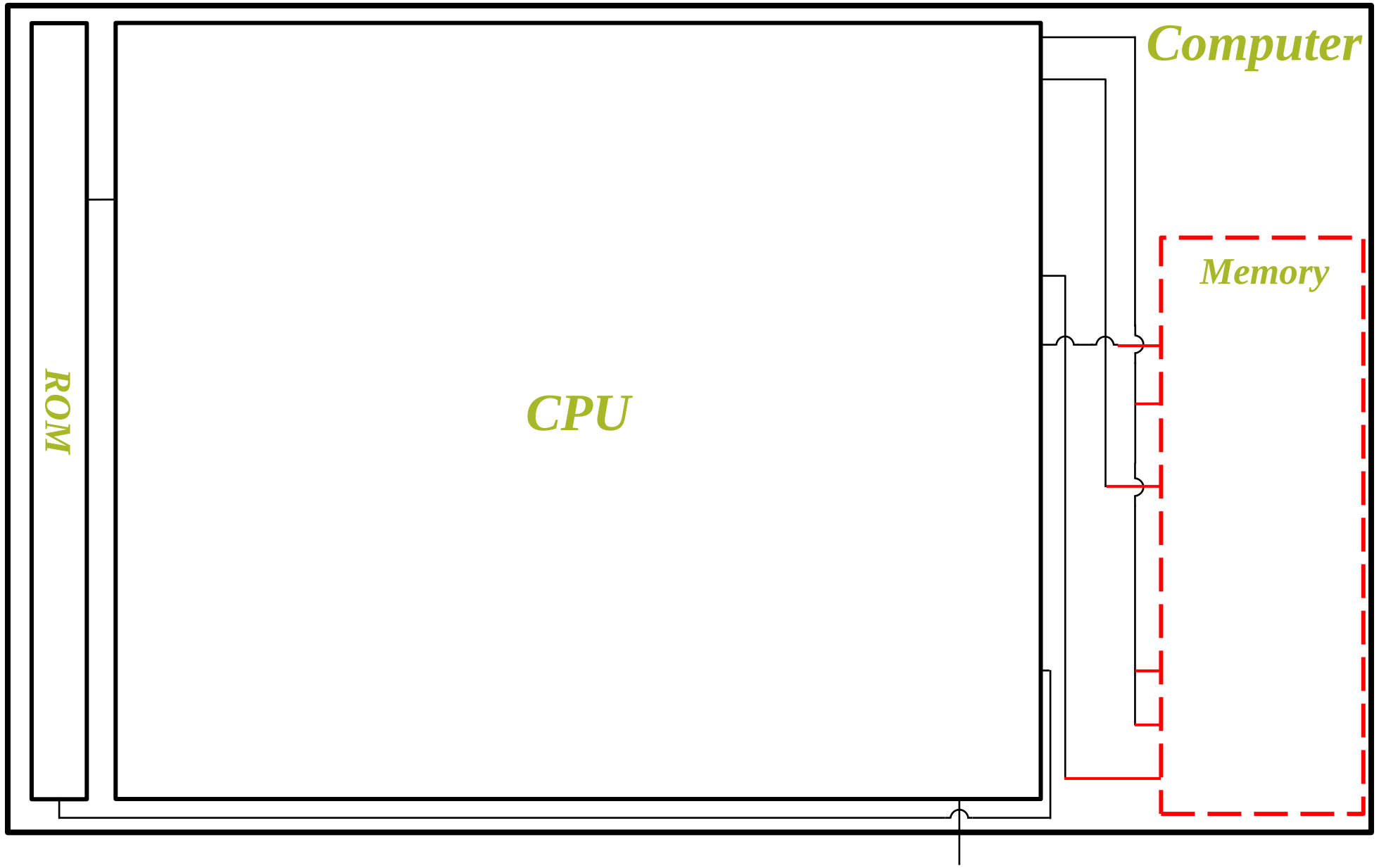
- 32 kB of data

Screen:

- 512×256 px
- 131072 bits
- 0 = white
- 1 = black

Keyboard:

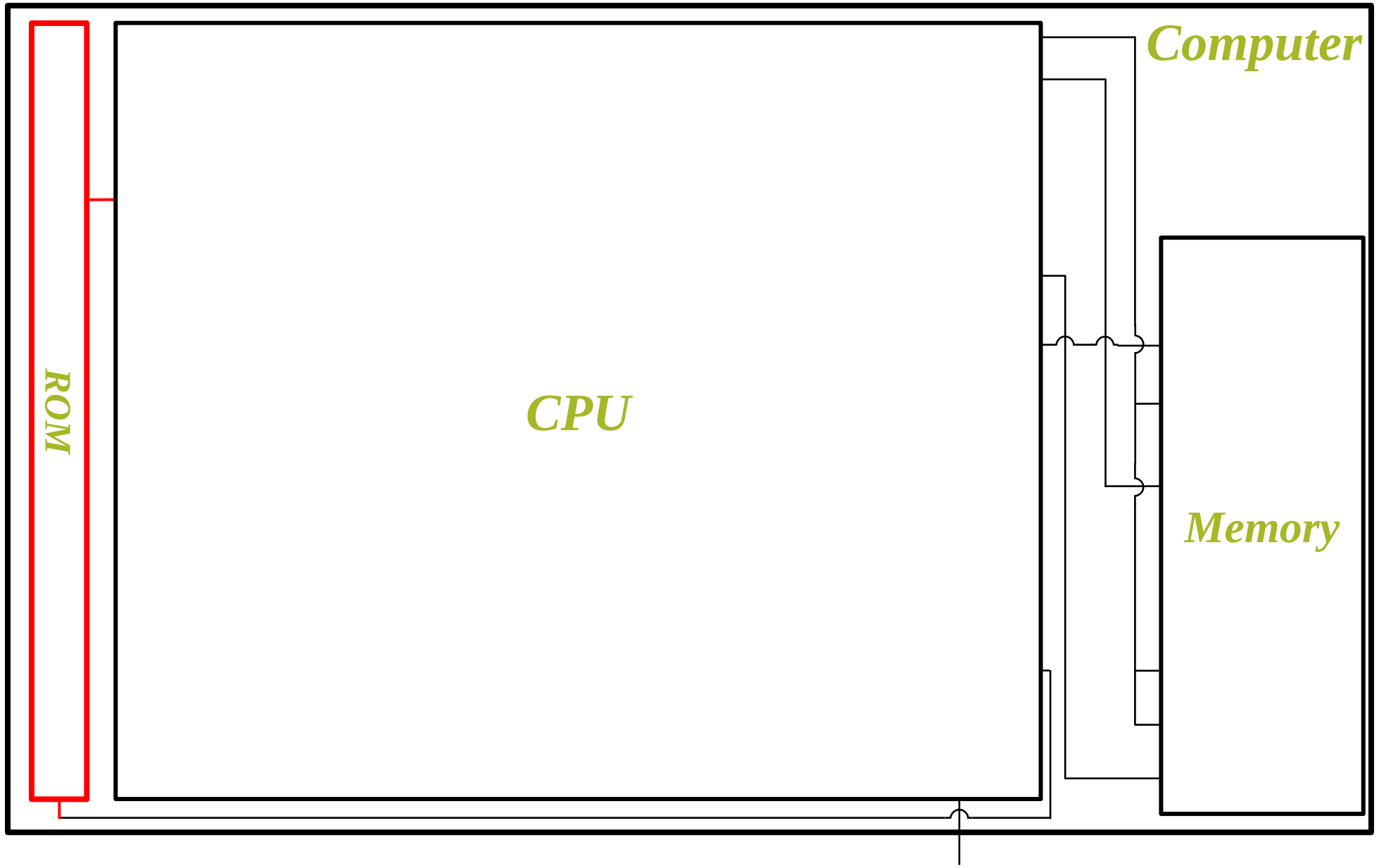
- 16 bit
- ASCII code



ROM

Read-Only Memory (ROM):

- 64 kB of Instructions

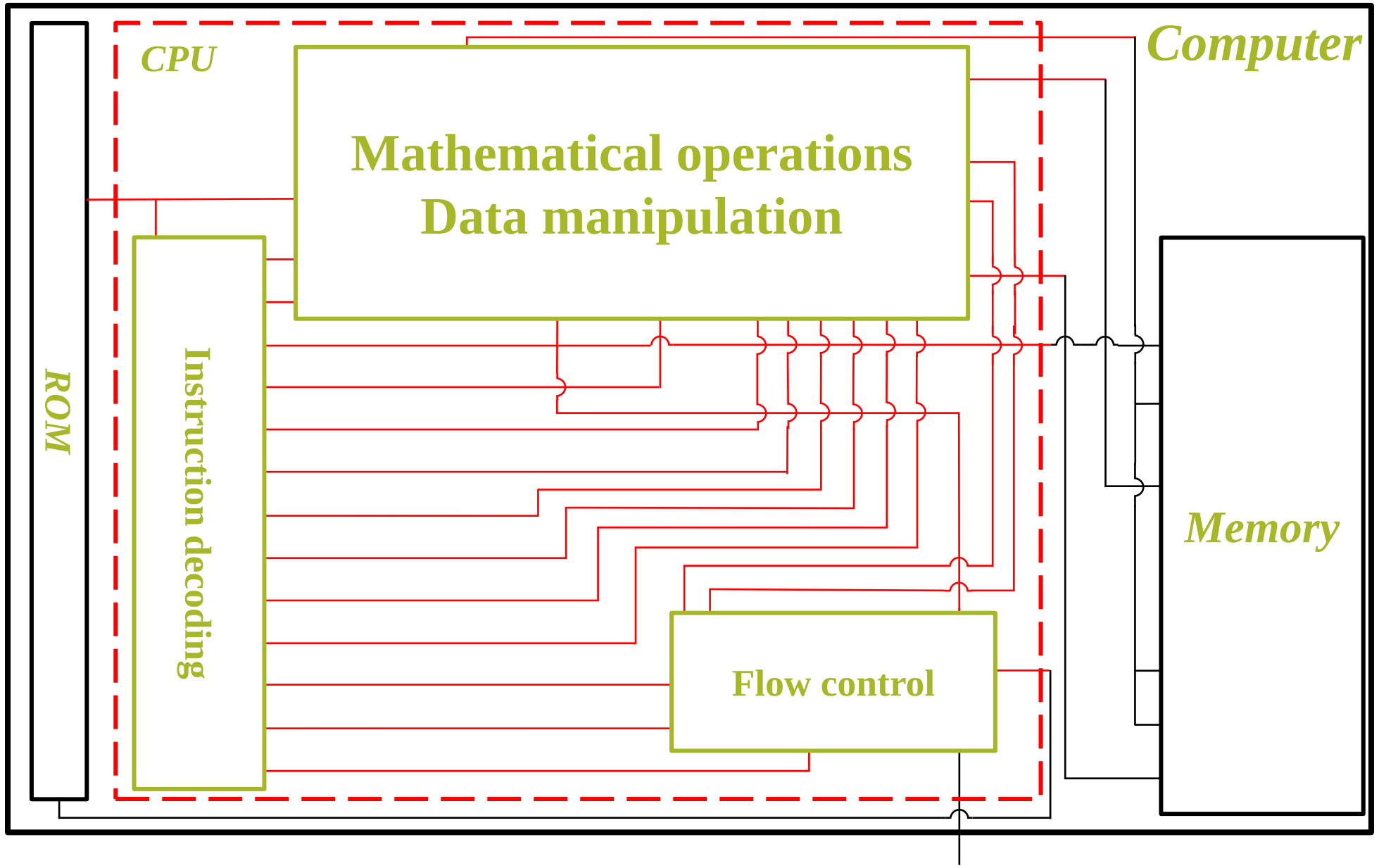


CPU

Central Processing Unit (CPU):

- Instruction decoding
- Mathematical operations & Data manipulation
- Flow control

Understanding software helps understanding hardware!



Assembler

- Assembly language – human
- Binary code – computer hardware

- Assembler function: converting assembly language into binary code

The assembly process:

1. Removing spaces/comments
2. Replacing variables/labels
3. Parsing each line into fields
4. Translating fields into binary
5. Combining into result

Assembly process – Example

1. Removing all spaces and comments

Before		After	
0	@R0	0	@R0
1	D = M	1	D=M
2	@result	2	@result
3	M = 0	3	M=0
4	(LOOP)	4	(LOOP)
5	@result	5	@result
6	M = D + M // update	6	M=D+M
7	@LOOP	7	@LOOP
8	D = D - 1; JGT	8	D=D-1;JGT
9	(END)	9	(END)
10	// endless loop	11	@END
11	@END	12	0;JMP
12	0; JMP		

Example

Before			After	
8	D = D - 1; JGT	→	8	D=D-1;JGT
10	// endless loop	→	10	

Assembly process – Example

2. Replacing all variables and labels

Before		After	
0	@R0	0	@0
1	D=M	1	D=M
2	@result	2	@16
3	M=0	3	M=0
4	(LOOP)	5	@16
5	@result	6	M=D+M
6	M=D+M	7	@4
7	@LOOP	8	D=D-1;JGT
8	D=D-1;JGT	10	@8
9	(END)	11	0;JMP
11	@END		
12	0;JMP		

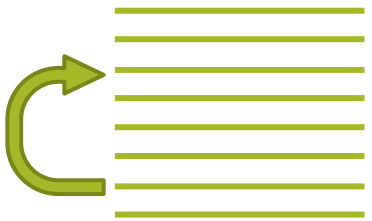
Variable – name for memory address

Example

Before			After	
2	@result	→	2	@16

Label – name for a line

Loops



If/else

Example

Before			After	
4	(LOOP)	→	4	
10	@LOOP	→	10	@4

Assembly process – Example

3. Parsing each line into its fields

Before		After				
		A	value			
		C	comp	dest	jump	
0	@0	0 A	0			
1	D=M	1 C	M	D	Null	
2	@16	2 A	16			
3	M=0	3 C	0	M	Null	
4	@16	4 A	16			
5	M=D+M	5 C	D+M	M	Null	
6	@4	6 A	4			
7	D=D-1;JGT	7 C	D-1	D	JGT	
8	@8	8 A	8			
9	0;JMP	9 C	0	Null	JMP	

	Assembly code
A	@*value*
C	*dest* = *comp*; *jump*

Example

Before			After			
A-instruction						
				value		
2	@16	→	A	16		
C-instruction						
				comp	dest	jump
7	D=D-1;JGT	→	C	D-1	D	JGT

Assembly process – Example

4. Translating individual fields into binary

	Before					After				
	A	*value*				**A**	*value*			
	C	*comp*	*dest*	*jump*		**C**	*comp*	*dest*	*jump*	
0	**A**	0			0	**0**	0000000000000000			
1	**C**	M	D	Null	1	**111**	1110000	010	000	
2	**A**	16			2	**0**	0000000000010000			
3	**C**	0	M	Null	3	**111**	0101010	001	000	
4	**A**	16			4	**0**	0000000000010000			
5	**C**	D+M	M	Null	5	**111**	1000010	001	000	
6	**A**	4			6	**0**	0000000000000100			
7	**C**	D-1	D	JGT	7	**111**	0001110	010	001	
8	**A**	8			8	**0**	0000000000001000			
9	**C**	0	Null	JMP	9	**111**	0101010	000	111	

Example

	Before				After
A-instruction					
		value			*value*
6	**A**	4	→	**0**	0000000000000100
C-instruction					
		comp	*dest*	*jump*	*comp*	*dest*	*jump*
7	**C**	D-1	D	JGT	→	**111**	0101010	010	001

Assembly process – Example

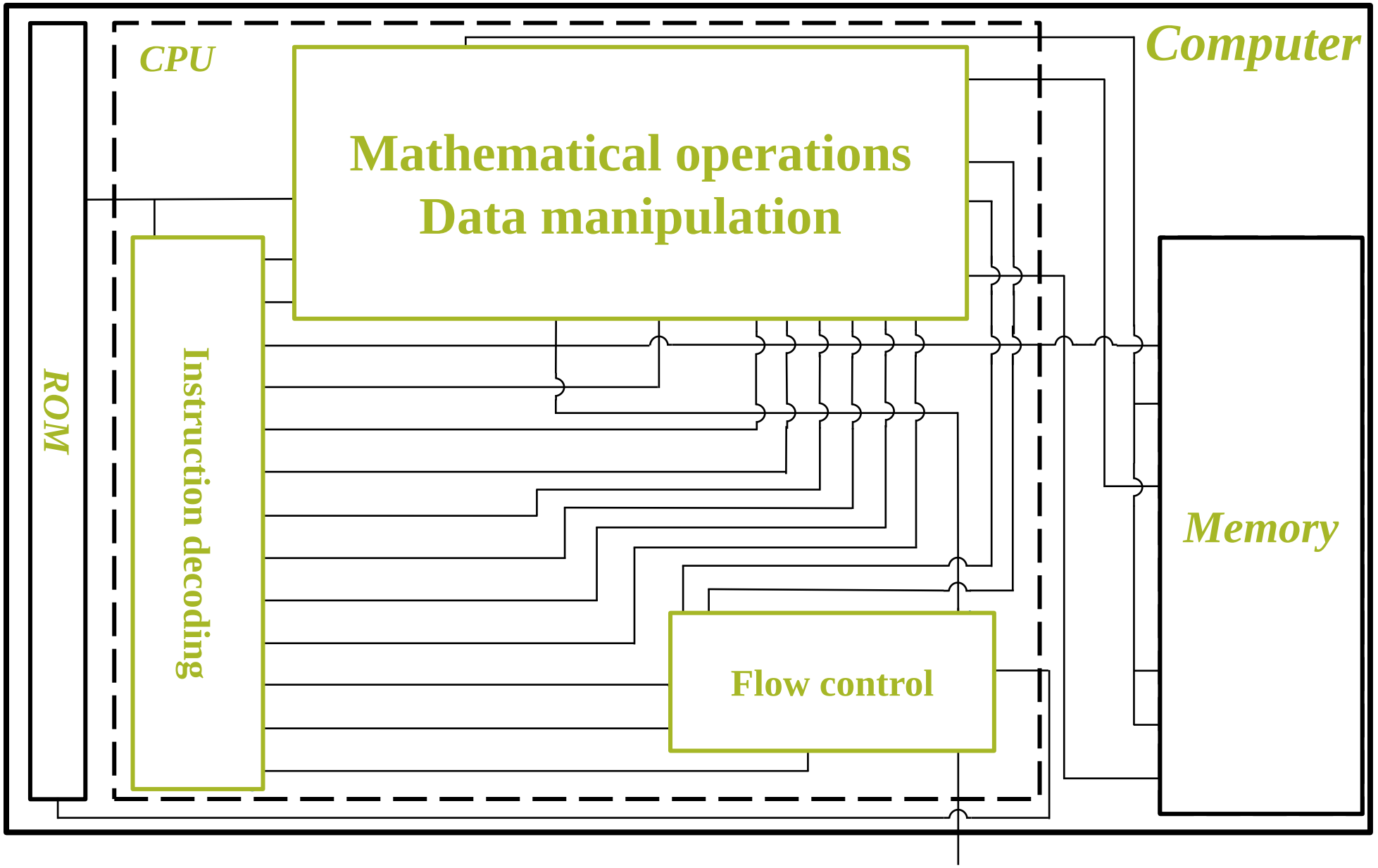
5. Combining translated fields into result

Before					After	
A		*value*				
C		*comp*	*dest*	*jump*		
0	**0**	0000000000000000			0	**0**0000000000000000
1	**111**	1110000	010	000	1	**111**1110000010000
2	**0**	0000000000010000			2	**0**0000000000010000
3	**111**	0101010	001	000	3	**111**0101010001000
4	**0**	0000000000010000			4	**0**0000000000010000
5	**111**	1000010	001	000	5	**111**1000010001000
6	**0**	0000000000000100			6	**0**0000000000000100
7	**111**	0001110	010	001	7	**111**0001110010001
8	**0**	0000000000001000			8	**0**0000000000001000
9	**111**	0101010	000	111	9	**111**0101010000111

Test correctness?

Load it into the computer!

CPU

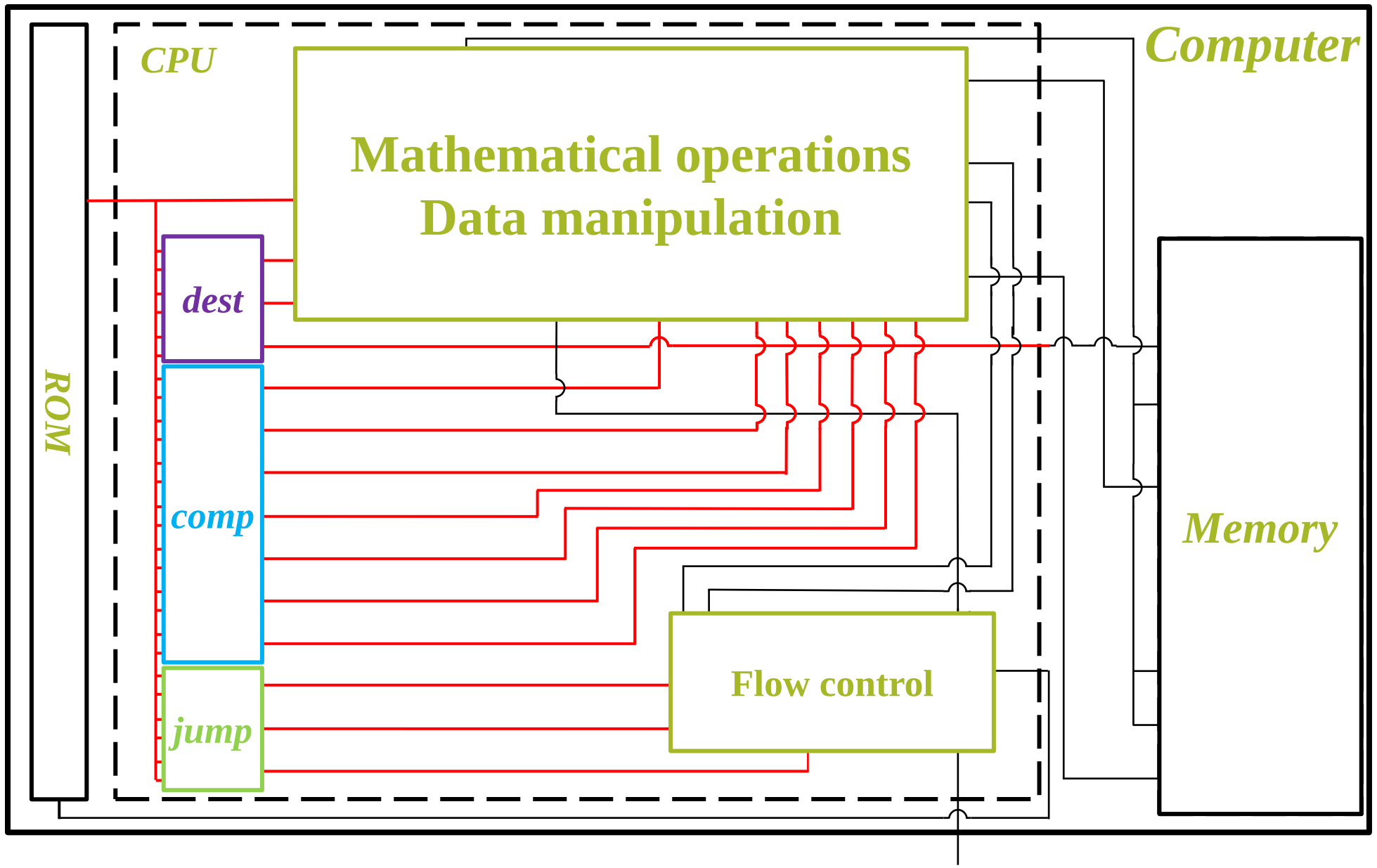

Computer
CPU
Mathematical operations
Data manipulation
ROM
Instruction decoding
Flow control
Memory

Instruction decoding

A-instruction

C-instruction

	Assembly code
A	@*value*
C	*dest* = *comp*; *jump*

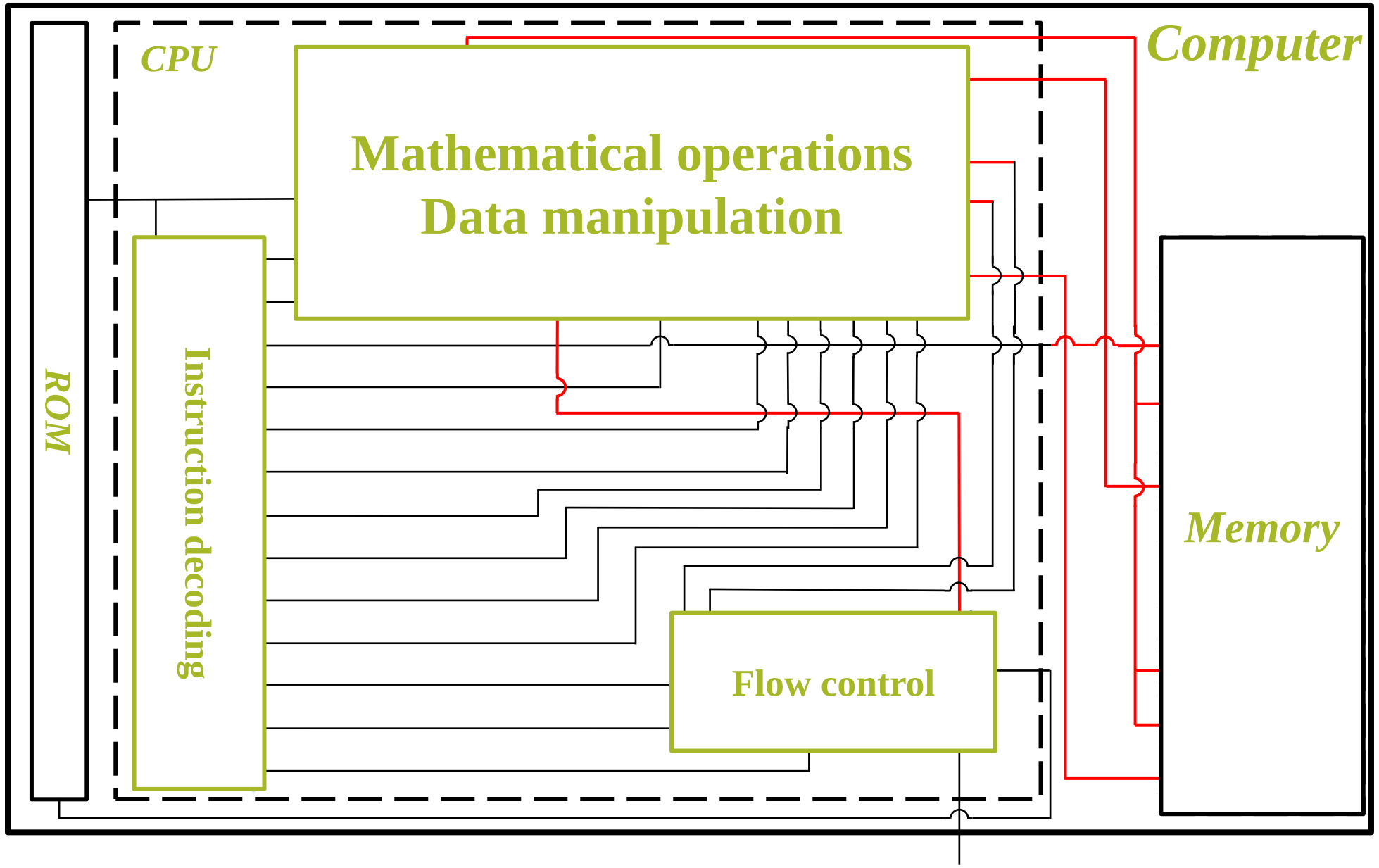


Execution

ARegister

DRegister

Arithmetic Logic Unit

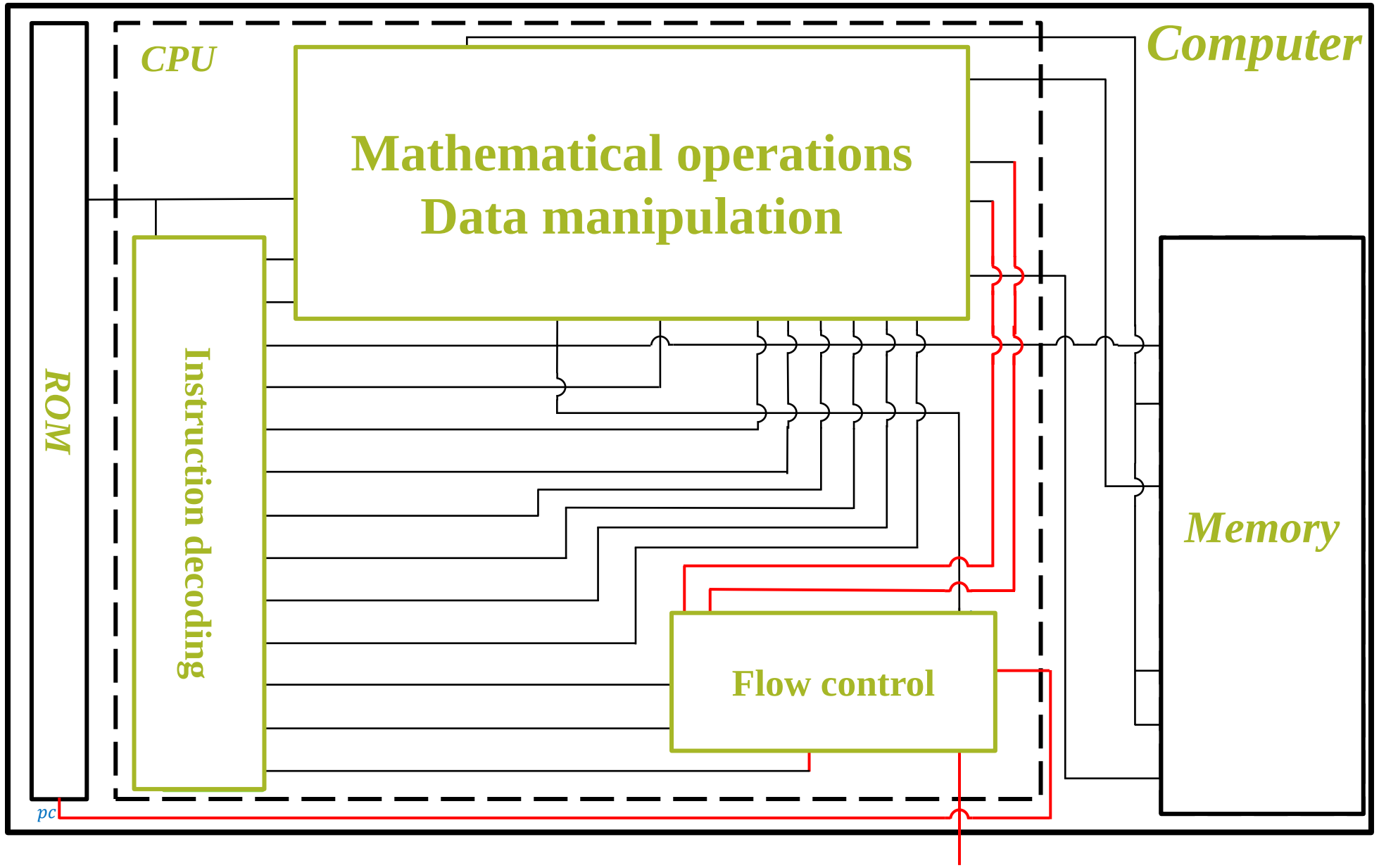


Flow control

Jump conditions

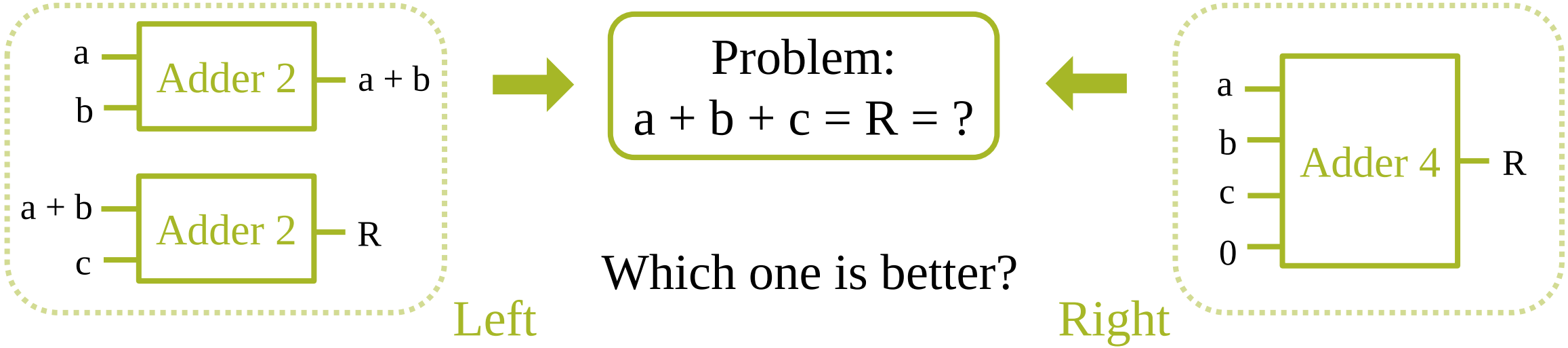
vs.

ALU output flags



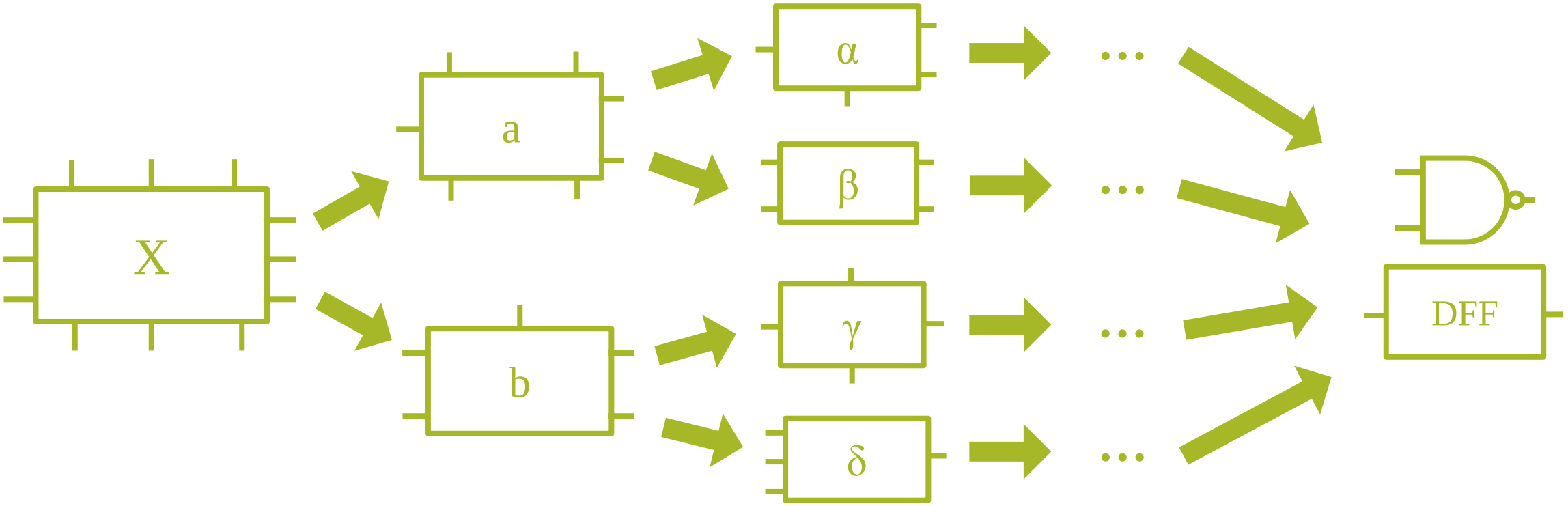
Improving the hardware design

- Fact: many ways of implementing the same functionality

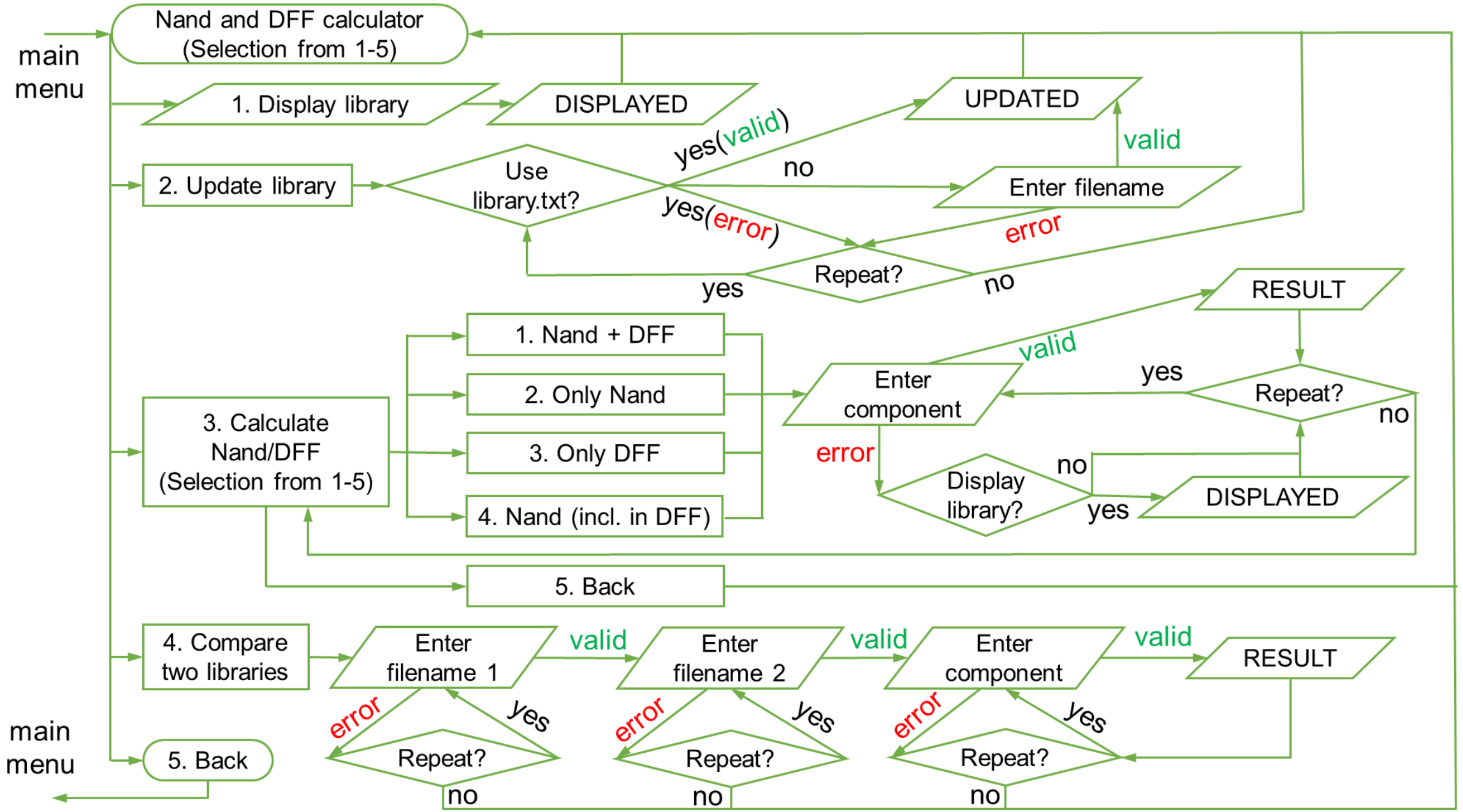


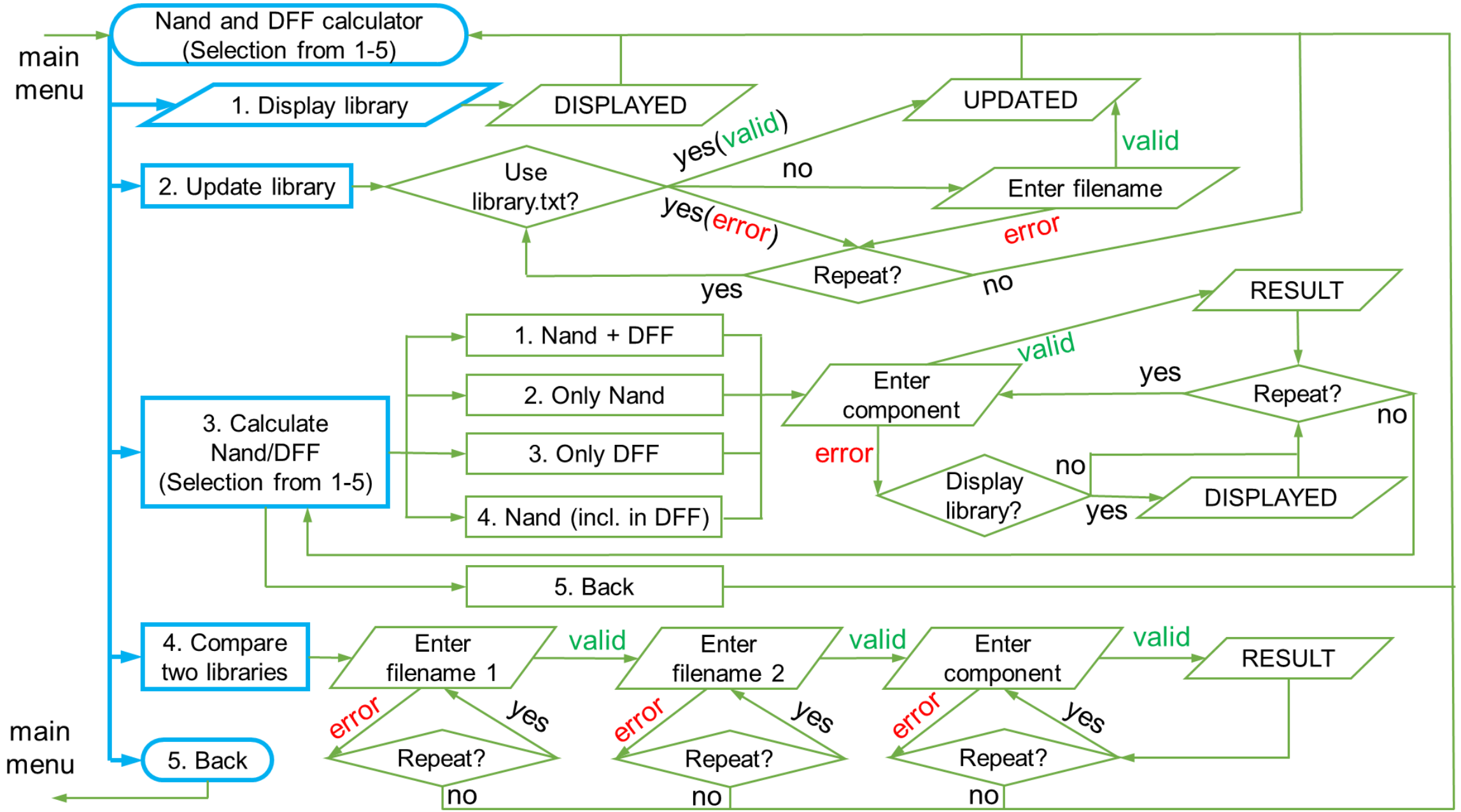
- Challenge: compare and find better ways

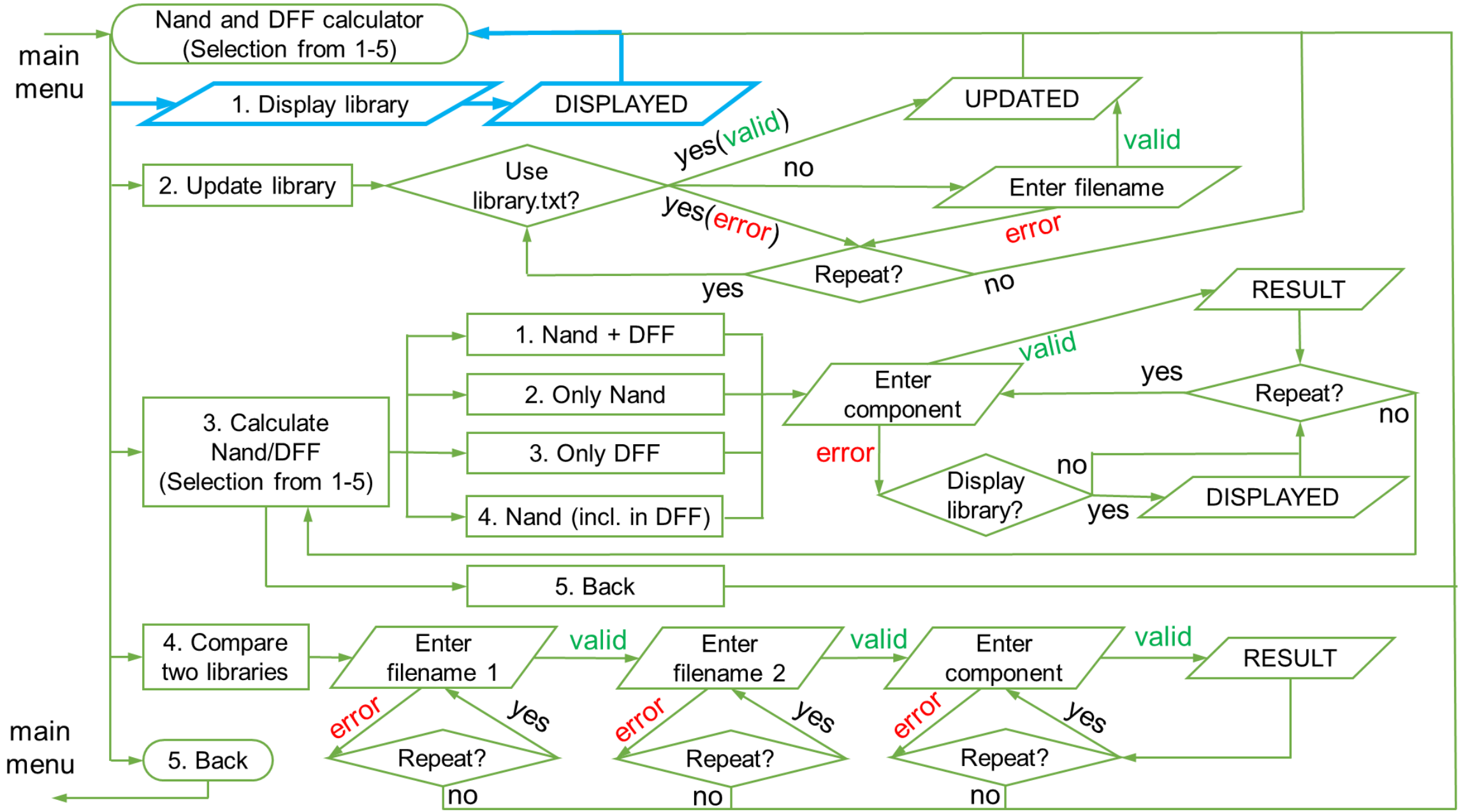
Nand & D Flip-flop calculator (transistor calculator in V2)

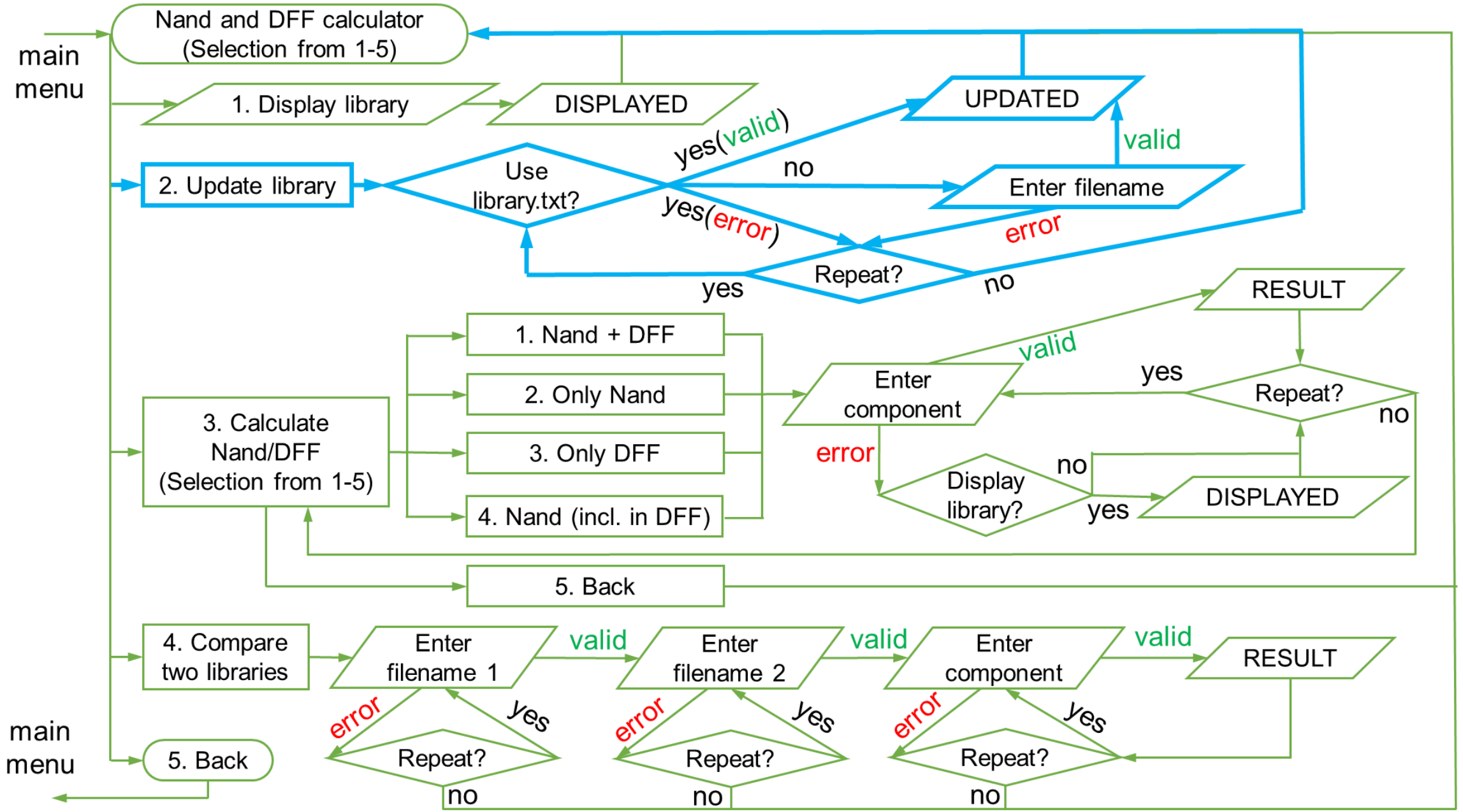


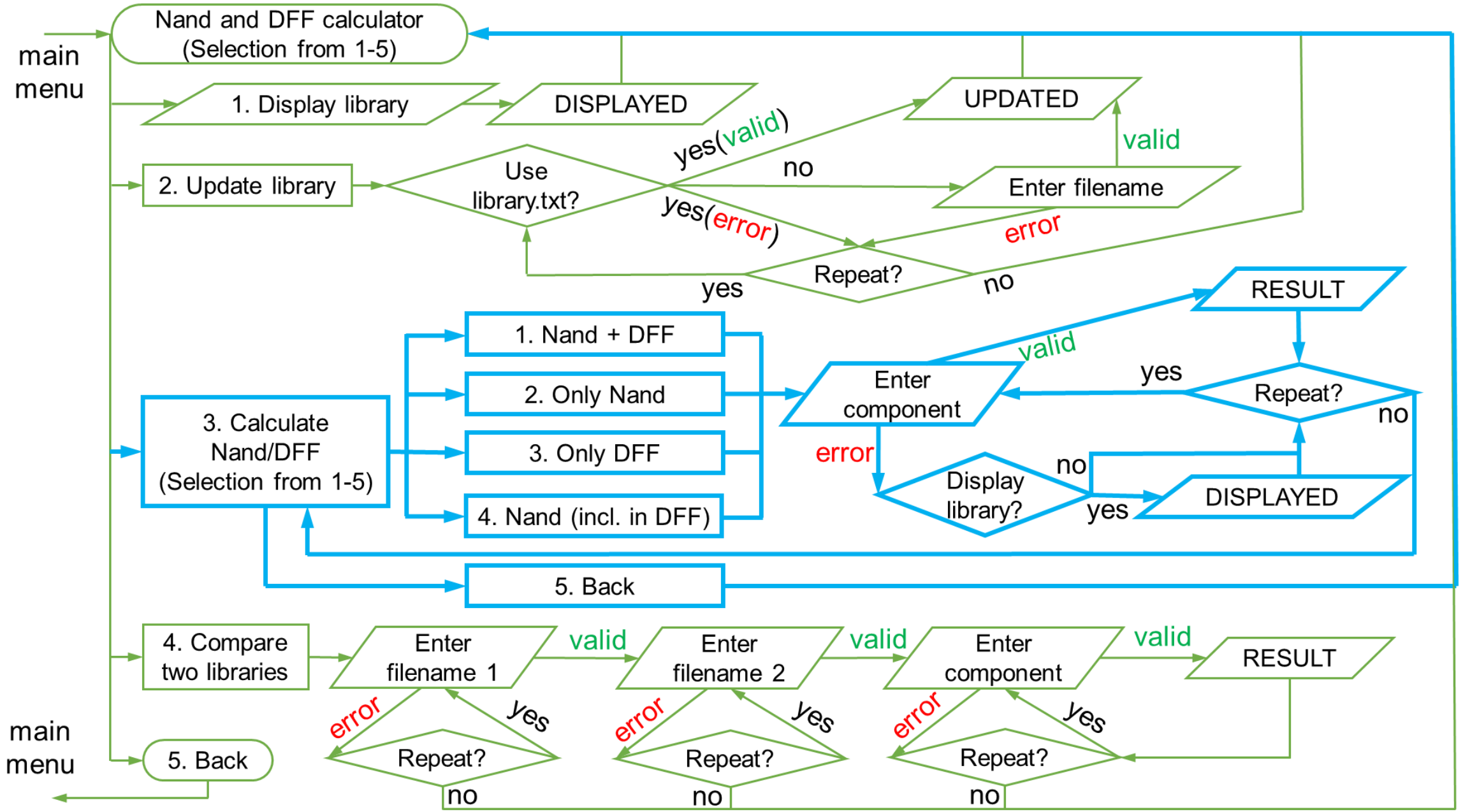
- Criteria: the number of Nands and DFFs (the number of transistors in V2)
- Program input: a text file with the structure of the desired component and subcomponents down to primitive components

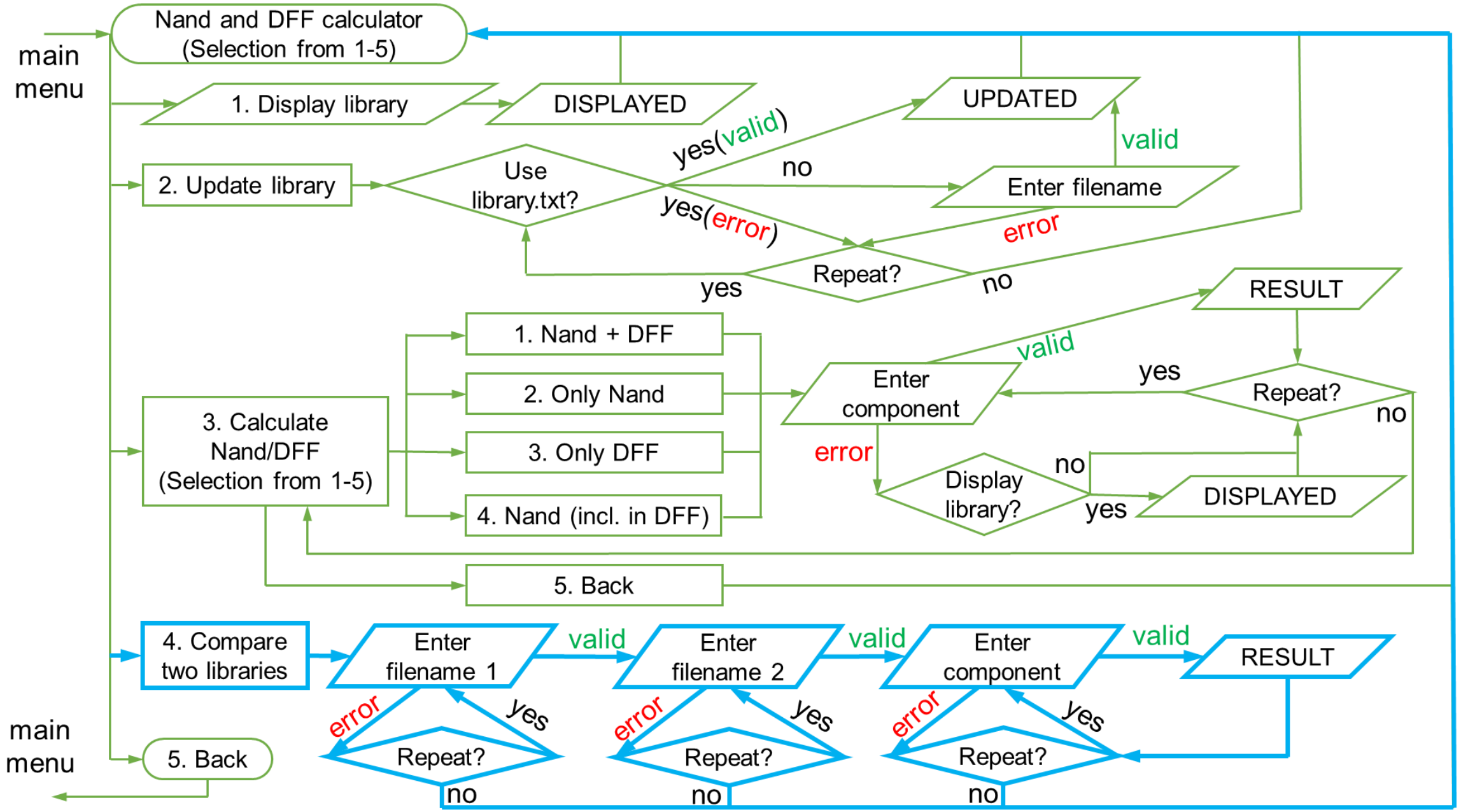

main menu
Nand and DFF calculator (Selection from 1-5)
1. Display library
DISPLAYED
2. Update library
Use library.txt?
yes(valid)
no
yes(error)
UPDATED
valid
Enter filename
error
Repeat?
yes
no
3. Calculate Nand/DFF (Selection from 1-5)
1. Nand + DFF
2. Only Nand
3. Only DFF
4. Nand (incl. in DFF)
Enter component
valid
RESULT
yes
Repeat?
no
error
Display library?
no
yes
DISPLAYED
5. Back
4. Compare two libraries
Enter filename 1
valid
Enter filename 2
valid
Enter component
valid
RESULT
error
yes
Repeat?
no
main menu
5. Back


main menu
Nand and DFF calculator (Selection from 1-5)
1. Display library
DISPLAYED
2. Update library
Use library.txt?
yes(valid)
no
yes(error)
UPDATED
valid
Enter filename
error
Repeat?
yes
no
3. Calculate Nand/DFF (Selection from 1-5)
1. Nand + DFF
2. Only Nand
3. Only DFF
4. Nand (incl. in DFF)
Enter component
valid
RESULT
yes
Repeat?
no
error
Display library?
no
yes
DISPLAYED
5. Back
4. Compare two libraries
Enter filename 1
valid
Enter filename 2
valid
Enter component
valid
RESULT
error
yes
Repeat?
no
main menu
5. Back

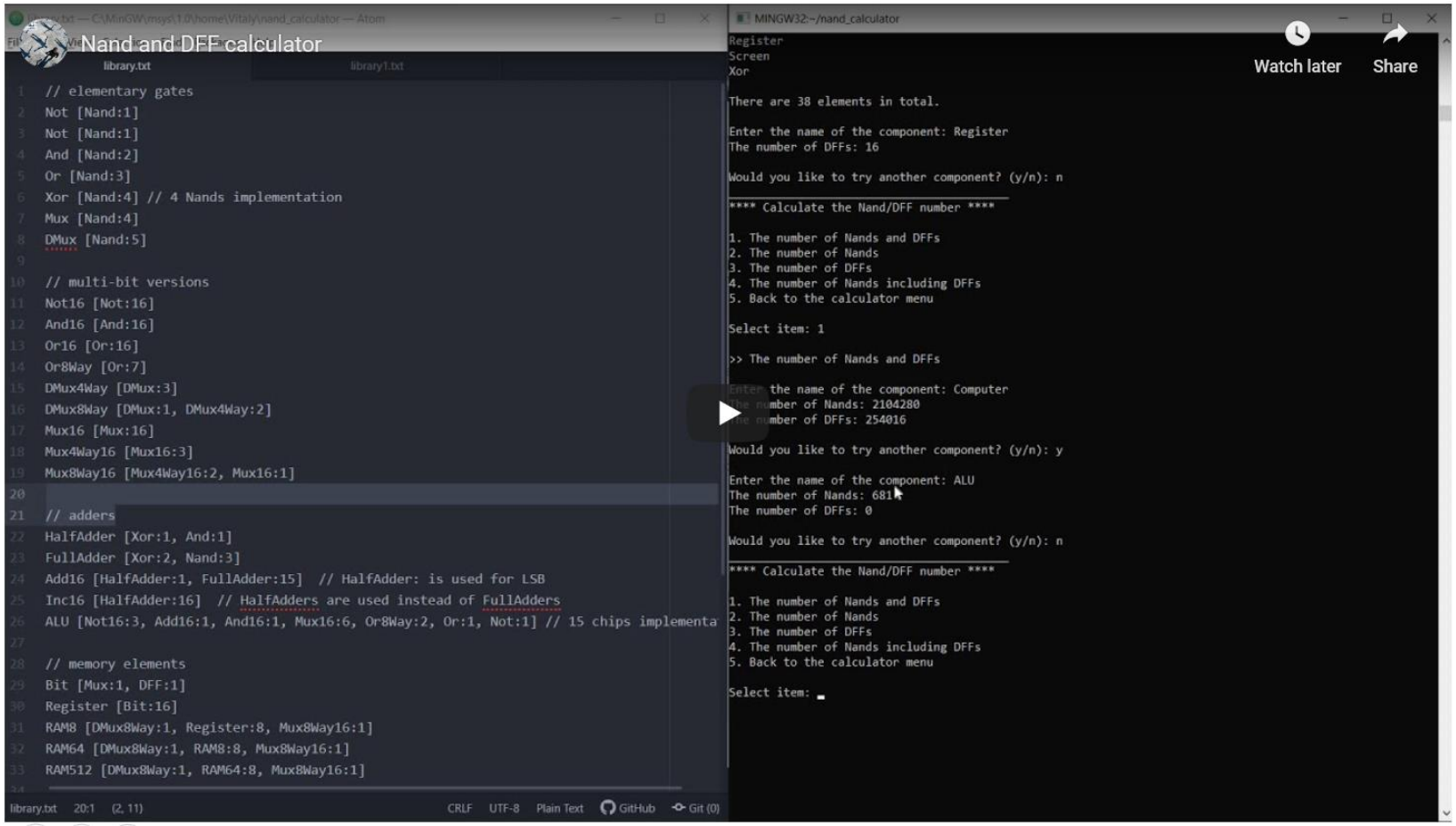

main menu
Nand and DFF calculator (Selection from 1-5)
1. Display library
DISPLAYED
2. Update library
Use library.txt?
yes(valid)
no
yes(error)
UPDATED
valid
Enter filename
error
Repeat?
yes
no
3. Calculate Nand/DFF (Selection from 1-5)
1. Nand + DFF
2. Only Nand
3. Only DFF
4. Nand (incl. in DFF)
Enter component
valid
RESULT
yes
Repeat?
no
error
Display library?
no
yes
DISPLAYED
5. Back
4. Compare two libraries
Enter filename 1
valid
Enter filename 2
valid
Enter component
valid
RESULT
error
yes
Repeat?
no
main menu
5. Back


main menu
Nand and DFF calculator (Selection from 1-5)
1. Display library
DISPLAYED
2. Update library
Use library.txt?
yes(valid)
no
yes(error)
UPDATED
valid
Enter filename
error
Repeat?
yes
no
3. Calculate Nand/DFF (Selection from 1-5)
1. Nand + DFF
2. Only Nand
3. Only DFF
4. Nand (incl. in DFF)
Enter component
valid
RESULT
yes
Repeat?
no
error
Display library?
no
yes
DISPLAYED
5. Back
4. Compare two libraries
Enter filename 1
valid
Enter filename 2
valid
Enter component
valid
RESULT
error
yes
Repeat?
no
main menu
5. Back


main menu
Nand and DFF calculator (Selection from 1-5)
1. Display library
DISPLAYED
2. Update library
Use library.txt?
yes(valid)
no
yes(error)
UPDATED
valid
Enter filename
error
Repeat?
yes
no
3. Calculate Nand/DFF (Selection from 1-5)
1. Nand + DFF
2. Only Nand
3. Only DFF
4. Nand (incl. in DFF)
Enter component
valid
RESULT
yes
Repeat?
no
error
Display library?
no
yes
DISPLAYED
5. Back
4. Compare two libraries
Enter filename 1
valid
Enter filename 2
valid
Enter component
valid
RESULT
error
yes
Repeat?
no
main menu
5. Back


main menu
Nand and DFF calculator (Selection from 1-5)
1. Display library
DISPLAYED
2. Update library
Use library.txt?
yes(valid)
no
yes(error)
UPDATED
valid
Enter filename
error
Repeat?
yes
no
3. Calculate Nand/DFF (Selection from 1-5)
1. Nand + DFF
2. Only Nand
3. Only DFF
4. Nand (incl. in DFF)
Enter component
valid
RESULT
yes
Repeat?
no
error
Display library?
no
yes
DISPLAYED
5. Back
4. Compare two libraries
Enter filename 1
valid
Enter filename 2
valid
Enter component
valid
RESULT
error
yes
Repeat?
no
main menu
5. Back

Demonstration



- Input file structure
- Output examples
- User guidance

https://youtu.be/_LZWHakxQ3Q