How to structure an interactive program: reinventing the Elm architecture

By Daniel Trujillo

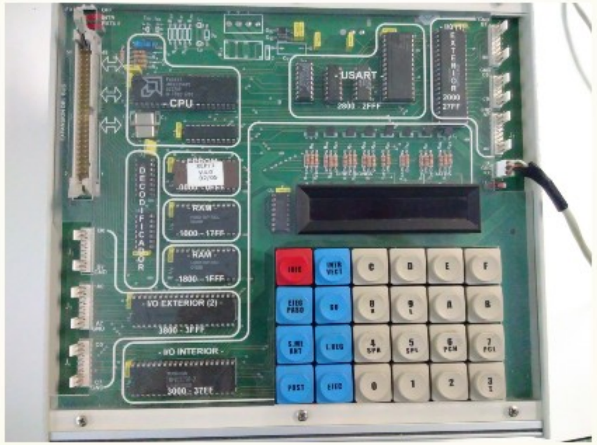
https://git.sr.ht/~gdanix/

What is this about?

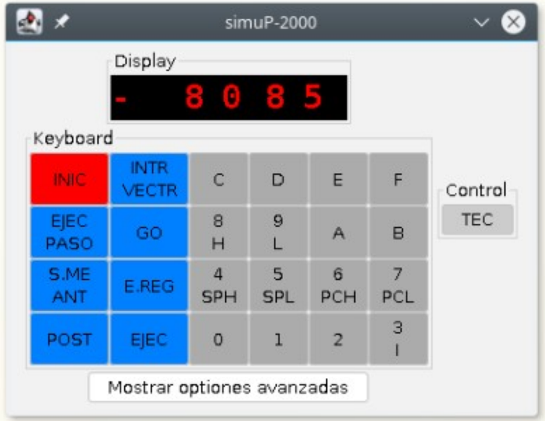
- Nansei simulator
- Functional core, imperative shell
- Application architecture

Nansei

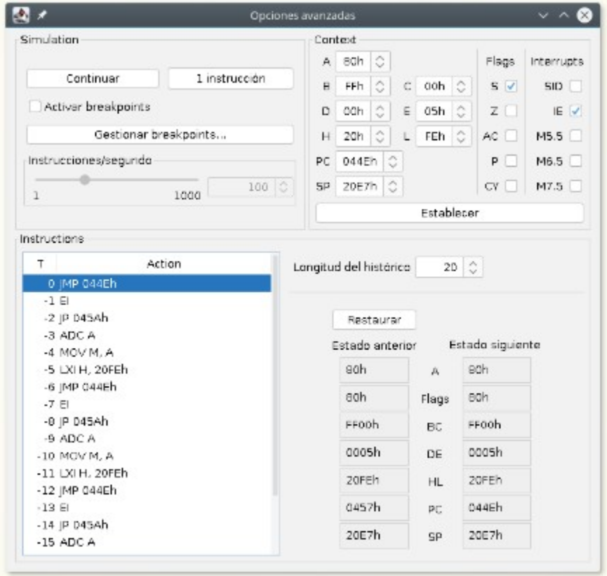
Alecop's microP-2000



Nansei
The simulator



Nansei



Activar breakpoints [ ]

Instrucciones/segundo: 100 (1 – 1000)

				Flags		Interrupts	
A	80h			S	✓	SID	
B	FFh	C	00h	Z		IE	✓
D	00h	E	05h	AC		M5.5	
H	20h	L	FEh	P		M6.5	
PC	044Eh			CY		M7.5	
SP	20E7h						

T	Action
0	JMP 044Eh
-1	EI
-2	JP 045Ah
-3	ADC A
-4	MOV M, A
-5	LXI H, 20FEh
-6	JMP 044Eh
-7	EI
-8	JP 045Ah
-9	ADC A
-10	MOV M, A
-11	LXI H, 20FEh
-12	JMP 044Eh
-13	EI
-14	JP 045Ah
-15	ADC A

Longitud del histórico: 20

Estado anterior		Estado siguiente
80h	A	80h
80h	Flags	80h
FF00h	BC	FF00h
0005h	DE	0005h
20FEh	HL	20FEh
0457h	PC	044Eh
20E7h	SP	20E7h

The architecture

The Object-Oriented way (1)

- Classes: Memory, CPU, Register...
- Main loop:
  - Fetch instruction from memory && supply to CPU
  - Or, maybe, just clock tick
  - It mutates
  - Sleep

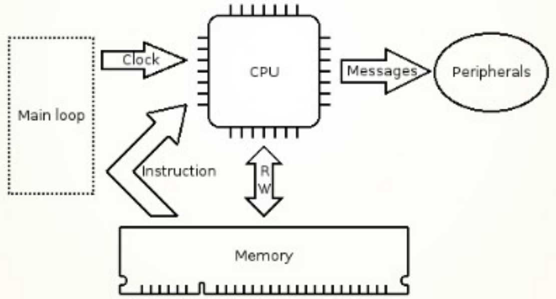
The architecture

The Object-Oriented way (2)

- The CPU is at the center af the application: God class?
- Very intuitive and simple, but:
  - Everything mutates: Memory, CPU and peripherals
  - Different callbacks forces us to have some global variables

The architecture

The Object-Oriented way (3)



The architecture

Can we do it better? And functional?

The architecture

The Functional way (1)

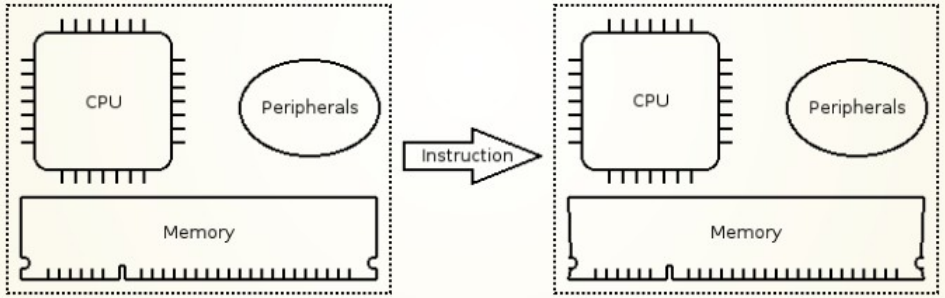
The central concept? Instructions are functions

The objects? The whole system

How are these dispatched? What is the main loop from OO?

The architecture

The functional way (2)



The architecture

The functional way (3)

What is "interaction"? A list of events and reactions

A global state that gets processed with every event

Doesn't sound familiar?

The architecture

The functional way (4)

It's the **foldLeft** algorithm!

```
lastState = eventList.foldLeft(processFunction)(initialState)
```

with processFunction:

(currentState, event) -> newState

The architecture

The functional way (5)

But what about the `eventList`?

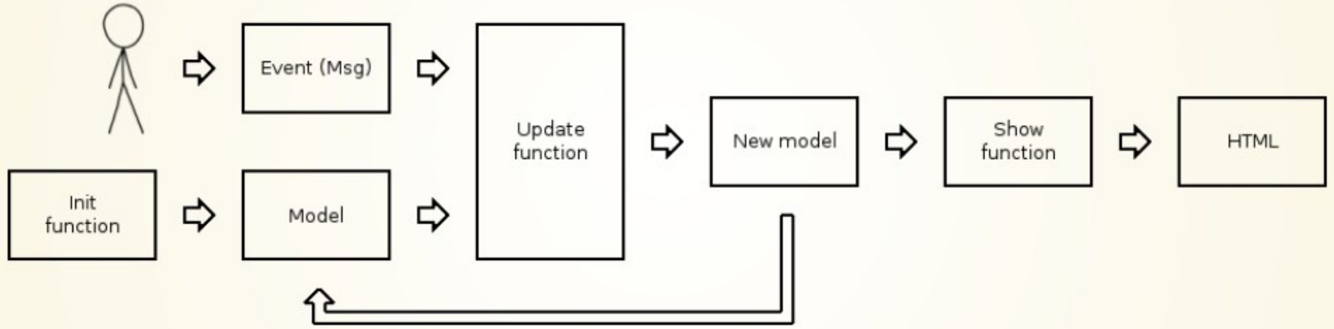
Streams!

The Elm Architecture

1. Wait for user input.
2. Send a message to `update`
3. Produce a new `Model`
4. Call `view` to get new HTML
5. Show the new HTML on screen
6. Repeat!

Source: Elm docs

The Elm Architecture



The end

- The Elm architecture is a simple but effective way of adding interactivity to purely functional programs
- Stream libraries (FS2, Monix, ScalaZ-Stream, etc..), or Stream and LazyList from Scala standard library can be used to **abstract away the future**
- Need GUI frameworks built around this architecture? Is web enough? -> Elm/Haskell