Σniac datapath

- Memory
  - Address
  - Data
  - W
  - R

- MB
  - A→MB
  - decoder
  - MB→MA
  - MB→IC
  - IC+1→IC

- OP
  - decoder

- MA
  - IC→MA

- IC
  - +1

- add/sub
  - B
  - subtract

- zero
**Σniac datapath**

ld A, C3
**SiC pro datamaph**

```
ld A, C3 fetching instruction
```
∑niac datapath

ld A, C3 fetching instruction
ld A, C3 decoding instruction
ld A, C3  load C3 into MA and increment IC
∑niac datapath

ld A, C3 fetch Mem[C3]
ld A, C3  load Mem[C3] into B
**Σniac datapath**

```
ld A, C3  load Mem[C3] into A
```
add A, C10 load addr. of next instruction into MA
The diagram represents the Sniac datapath. The main components include:

- **Memory**: Data flow to the Memory block.
- **MB**: Data flows from Memory to MB, and MB to IC.
- **OP**: The OP block receives data from MB.
- **IC**: Interconnects MA and IC.
- **MA**: Data flows from IC and IC+1.
- **IC**: Interconnects MB and IC.
- **B**: Performs subtract operation.
- **A**: Receives data from B.
- **add/sub**: Adds or subtracts data.
- **zero**: Indicates zero condition.

The instruction flow is as follows:

1. **Fetch Instruction**: 0x01
2. **Operation**: 0x46
3. **Data Flow**: 0x46
4. **Add/Subtract**: 0x03
5. **Result**: 0x03

The text at the bottom indicates the instruction: `add A, C10 fetch instruction`.
Sigma iac datapath

```
add A, C10 decode instruction
```
add A, C10  load C10 into MA and increment IC
\[ \Sigma \text{niac datapath} \]

```
add A, C10  fetch Mem[C10]
```
add A, C10  load Mem[C10] into B
Σniac datapath

\[ A = A + B \]
| t0 | Mem ⇒ MB: | R, ld_mb |
| t1 | MB ⇒ OP: | ld_op |
| t2 | Inc IC: | IC+1 > IC, ld_ic |
| t3 | MB > MA, ld_ma, set E | IC > MA, ld_ma |

I

| t0 | Mem ⇒ MB: | A > MB, ld_mb |
| t1 | MB ⇒ B: | ld_b |
| t2 | zero, ld_a | sub, ld_a | ld_a |
| t3 | IC > MA, ld_ma, reset E | |