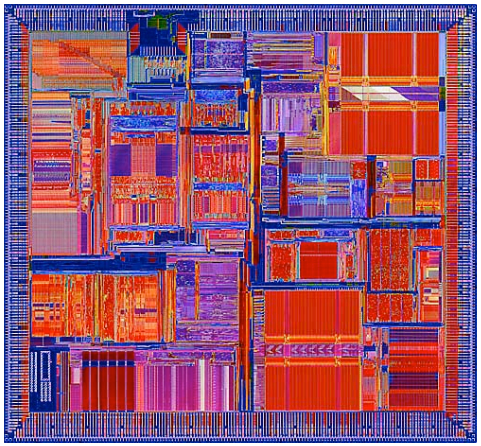
# Designing a CPU





## Let's build a computer!

CPU: "central processing unit"

computer: CPU + display + optical disk + metal case + power supply + ...


Last lecture: circuit that implements an adder

This lecture: circuit that implements a CPU



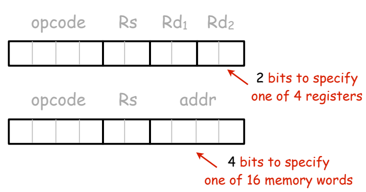
## TOY Lite

TOY machine.

- 256 16-bit words of memory.
- 16 16-bit registers.
- 1 8-bit program counter.
- 2 instruction types
- 16 instructions.


TOY-Lite machine.

- 16 10-bit words of memory.
- 4 10-bit registers.
- 1 4-bit program counter.
- 2 instruction types
- 16 instructions.




Goal: CPU circuit for TOY-Lite (same design extends to TOY, your computer)



## Primary Components of Toy-Lite CPU

✓ Arithmetic and Logic Unit (ALU)

Memory

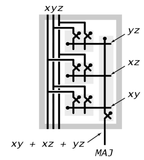
Toy-Lite Registers

Processor Registers: Program Counter and Instruction Register

"Control"

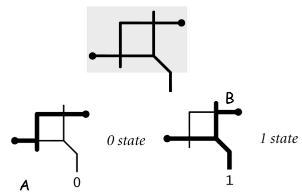
## A New Ingredient: Circuits With Memory

Combinational circuits.
- Output determined solely by inputs.
- Ex: majority, adder, decoder, MUX, ALU.




Sequential circuits.
- Output determined by inputs and current "state".
- Ex: memory, program counter, CPU.

Ex. Simplest feedback loop.
- Two controlled switches A and B, both connected to power, each blocked by the other.
- State determined by whichever switches first.
- Stable.



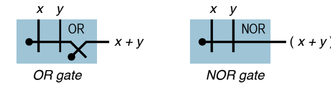

Aside. Feedback with an odd number of switches is a buzzer (not stable).

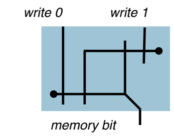
Doorbell: buzzer made with relays.

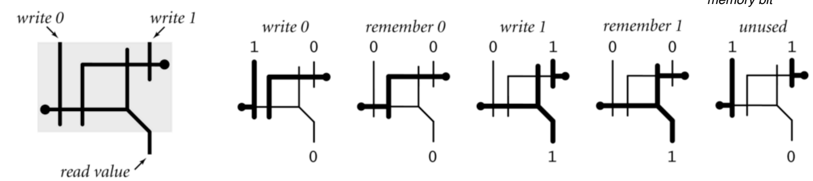


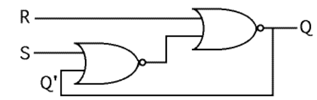
## SR Flip-Flop

SR Flip-flop.
- Two cross-coupled NOR gates
- A way to control the feedback loop.
- Abstraction that "remembers" one bit.
- Basic building block for memory and registers.



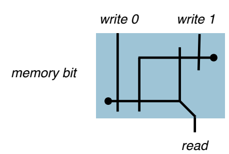




Caveat. Timing, switching delay.



## Memory Overview

Computers and TOY have several memory components.
- Program counter and other processor registers.
- TOY registers (4 10-bit words in Toy-Lite).
- Main memory (16 10-bit words in Toy-Lite).




Implementation.
- Use one flip-flop for each bit of memory.
- Use buses and multiplexers to group bits into words.

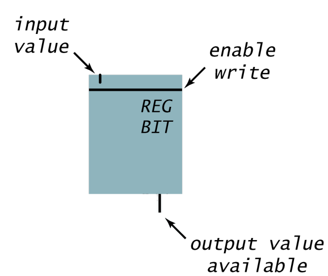
Access mechanism: when are contents available?
- Processor registers: enable write.
- Main memory: select and enable write.
- TOY register: dual select and enable write

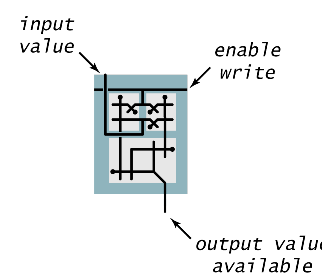
need to be able to read two registers at once



## Processor register Bit

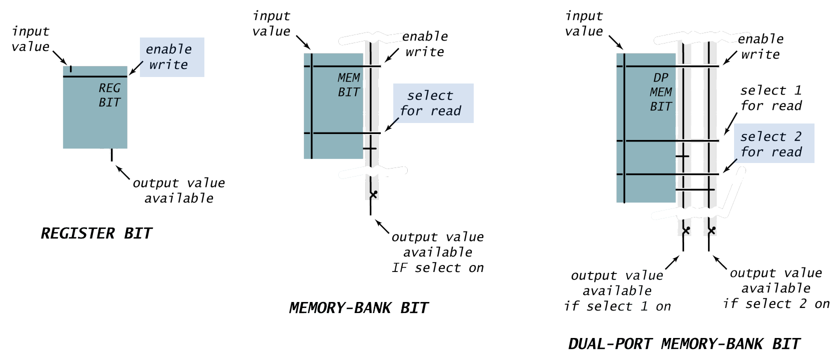
Processor register bit. Extend a flip-flop to allow easy access to values.



## Memory Bit Interface

Memory and TOY register bits: Add selection mechanism.

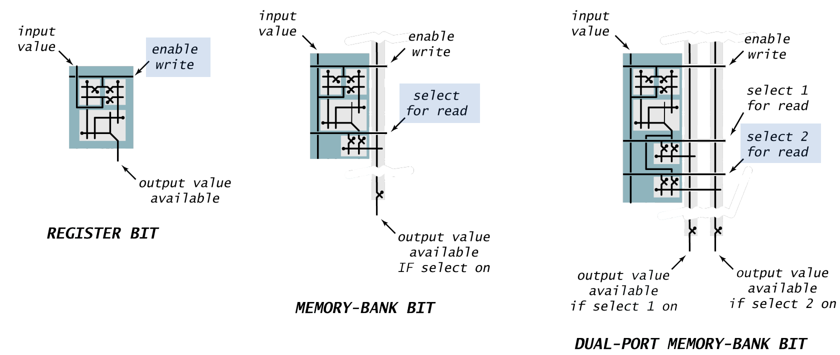



[ TOY PC, IR ] [ TOY main memory ] [ TOY registers ]



## Memory Bit: Switch Level Implementation

Memory and TOY register bits: Add selection mechanism.




[ TOY PC, IR ] [ TOY main memory ] [ TOY registers ]



## Processor Register

Processor register. ← don't confuse with TOY register

- Stores k bits.
- Register contents always available on output bus.
- If enable write is asserted, k input bits get copied into register.

Ex 1. TOY-Lite program counter (PC) holds 4-bit address.
Ex 2. TOY-Lite instruction register (IR) holds 10-bit current instruction.

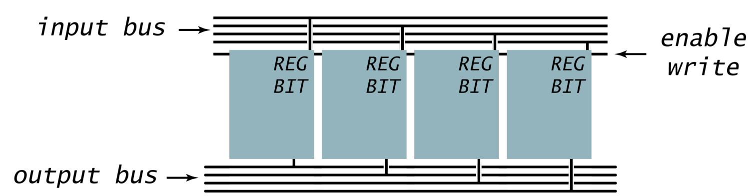



## Processor Register

Processor register. — don't confuse with TOY register

- Stores k bits.
- Register contents always available on output bus.
- If enable write is asserted, k input bits get copied into register.

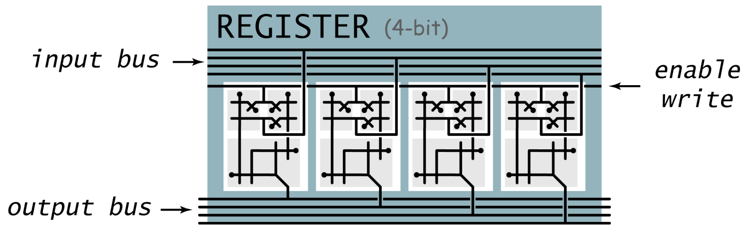
Ex 1. TOY program counter (PC) holds 8-bit address.
Ex 2. TOY instruction register (IR) holds 16-bit current instruction.



## Processor Register

Processor register. — don't confuse with TOY register
- Stores k bits.
- Register contents always available on output bus.
- If enable write is asserted, k input bits get copied into register.

Ex 1. TOY program counter (PC) holds 8-bit address.
Ex 2. TOY instruction register (IR) holds 16-bit current instruction.





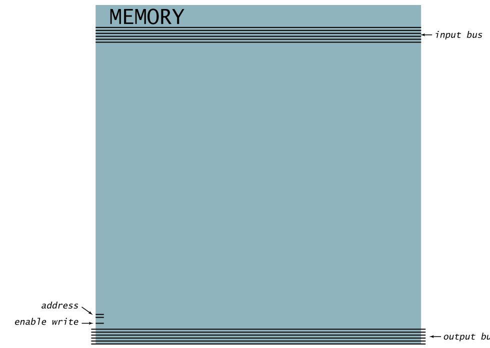

## Memory Bank

Memory bank.
- Bank of n registers; each stores k bits.
- Read and write information to one of n registers.
- Address inputs specify which one. — $\log_2 n$ address bits needed
- Addressed bits always appear on output.
- If write enabled, k input bits are copied into addressed register.

Ex 0 (for lecture). 4-by-6
(four 6-bit words)

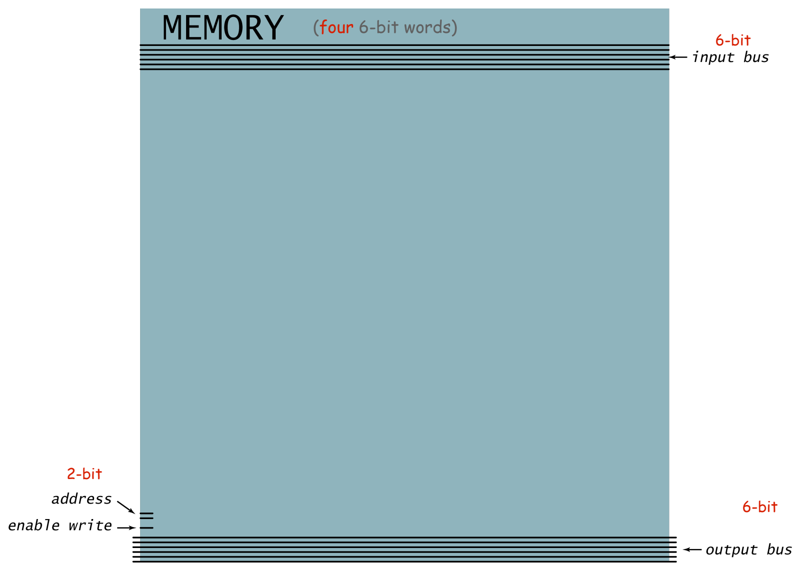
Ex 1. Main memory bank.
- TOY: 256-by-16
- TOY-Lite: 16-by-10

Ex 2. Registers.
- TOY: 16-by-16
- TOY Lite: 4-by-10
- Two output buses.





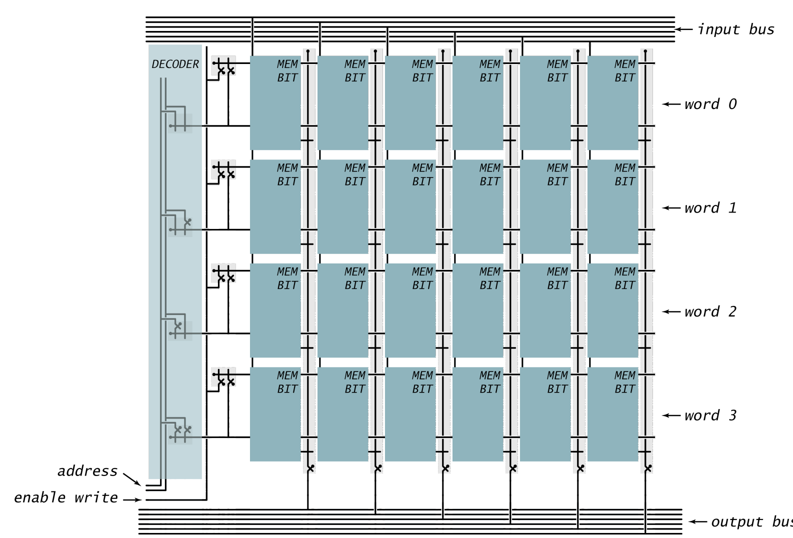

## Memory: Interface



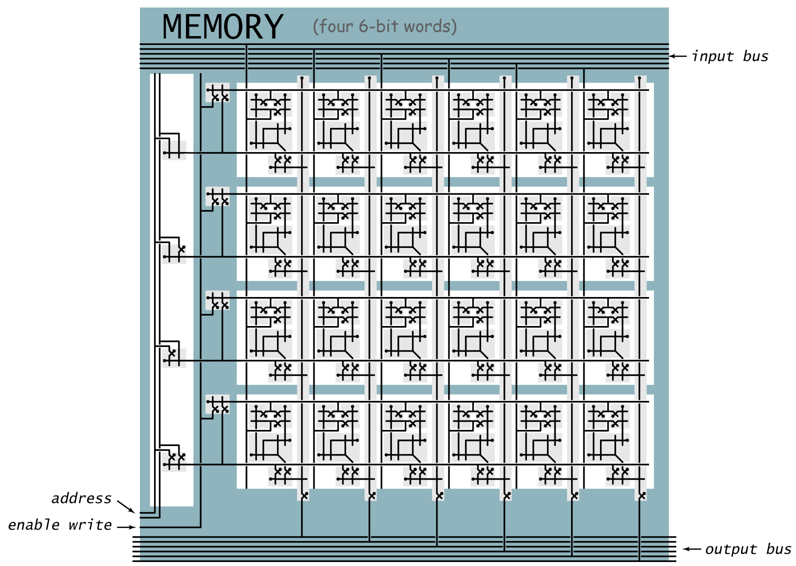



## Memory: Component Level Implementation

Decoder plus memory selection: connect only to addressed word.



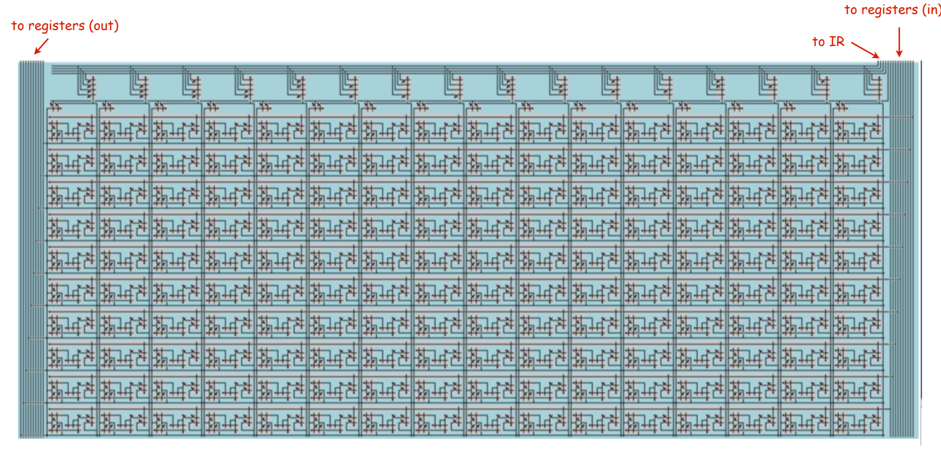
## Memory: Switch Level Implementation






## TOY-Lite Memory

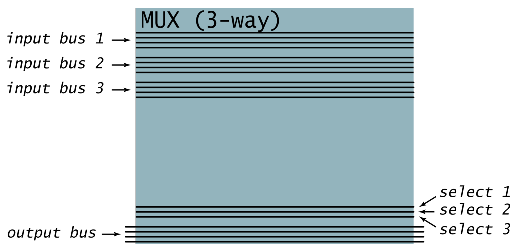
16 10-bit words
- input connected to registers for "store"
- output connected to registers for "load"
- addr connect to processor Instruction Register (IR)






## Another Useful Combinational Circuit: Multiplexer

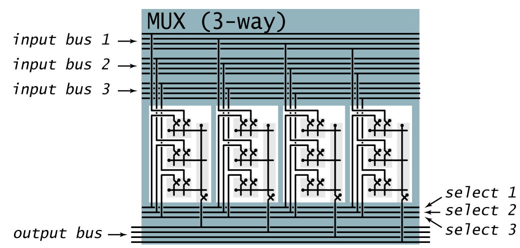
Multiplexer (MUX). Combinational circuit that selects among input buses.
- Exactly one select line i is activated.
- Copies bits from input bus i to output bus.





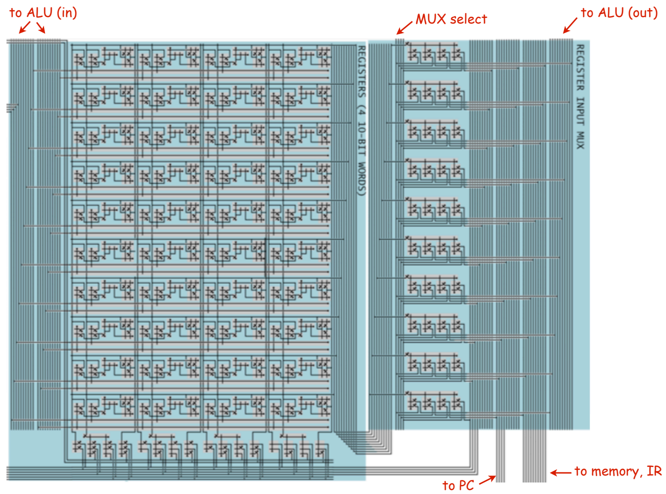

## Nuts and Bolts: Buses and Multiplexers

Multiplexer (MUX). Combinational circuit that selects among input buses.
- Exactly one select line i is activated.
- Copies bits from input bus i to output bus.



## Toy-Lite Registers

4 10-bit words
- Dual-ported to support connecting two different registers to ALU
- Input MUX to support input connection to ALU, memory, IR, PC






## Primary Components of Toy-Lite CPU

- ✓ ALU
- ✓ Memory
- ✓ Registers
- ✓ Processor Registers: Program Counter and Instruction Register

Not quite done.
Need to be able to increment.

"Control"



## How To Design a Digital Device

How to design a digital device.
- Design interface: input buses, output buses, control wires.
- Determine components.
- Determine datapath requirements: "flow" of bits.
- Establish control sequence.

Warmup. Design a program counter (3 devices, 3 control wires).

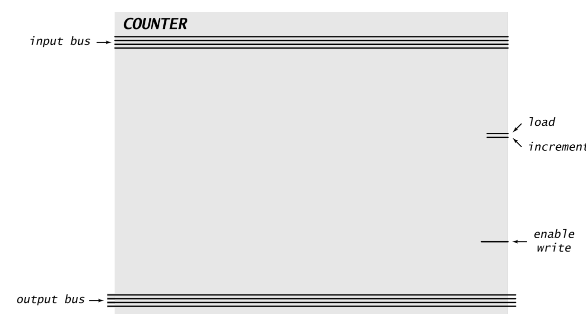
Goal. Design TOY-Lite computer (10 devices, 27 control wires).



## Program Counter: Interface

Counter. Holds value that represents a binary number.
- Load: set value from input bus.
- Increment: add one to value.
- Enable Write: make value available on output bus.

Ex. TOY-Lite program counter (4-bit).



## Program Counter:  Components

Components.
- Register.
- Incrementer.
- Multiplexer (to provide connections for both load and increment).



## Program Counter:  Datapath and Control

Datapath.
- Layout and interconnection of components.
- Connect input and output buses.

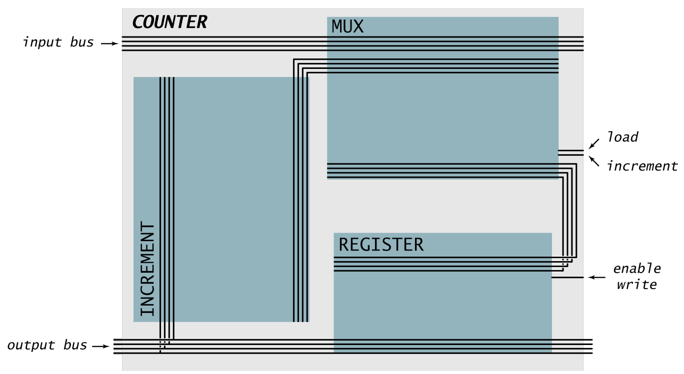
Control.  Choreographs the "flow" of information on the datapath.



## Program Counter:  Datapath and Control

Datapath.
- Layout and interconnection of components.
- Connect input and output buses.

Control.  Choreographs the "flow" of information on the datapath.

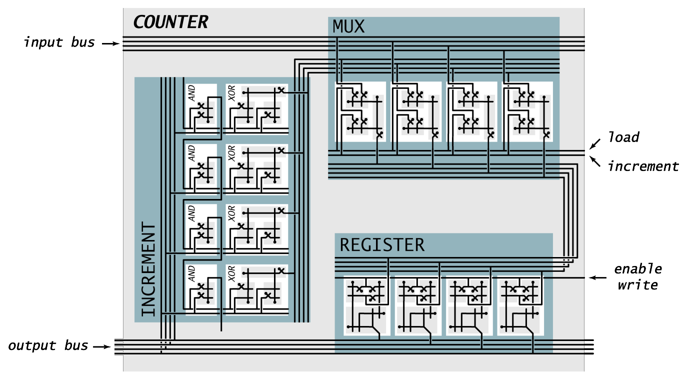




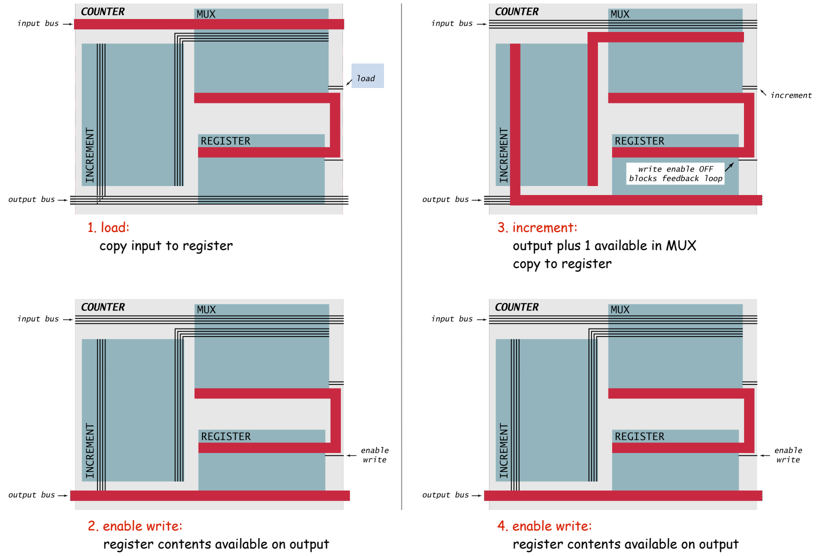

## Program Counter:  Datapath and Control

Datapath.
- Layout and interconnection of components.
- Connect input and output buses.

Control.  Choreographs the "flow" of information on the datapath.



## Program Counter:  Datapath and Control






## Primary Components of Toy-Lite CPU

✓ ALU

✓ Memory

✓ Toy-Lite Registers

✓ Processor Registers: Program Counter and Instruction Register

"Control"



## How To Design a Digital Device

How to design a digital device.
- Design interface:  input buses, output buses, control wires.
- Determine components.
- Determine datapath requirements:  "flow" of bits.
- Establish control sequence.

Warmup.  Design a program counter (3 devices, 3 control wires).

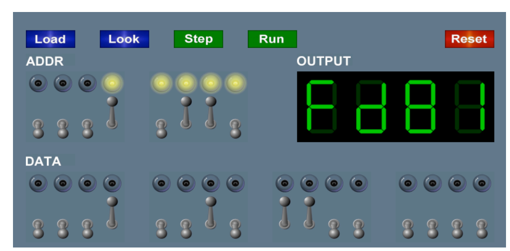
Next.  Design TOY-Lite computer (10 devices, 27 control wires).



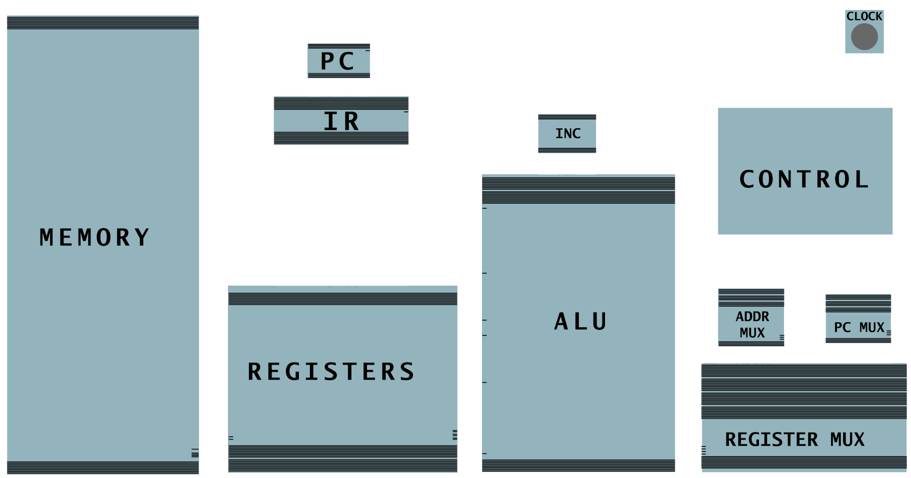
## TOY-Lite: Interface

CPU is a circuit.

Interface: switches and lights.
- set memory contents
- set PC value
- press RUN
- [details of connection to circuit omitted]

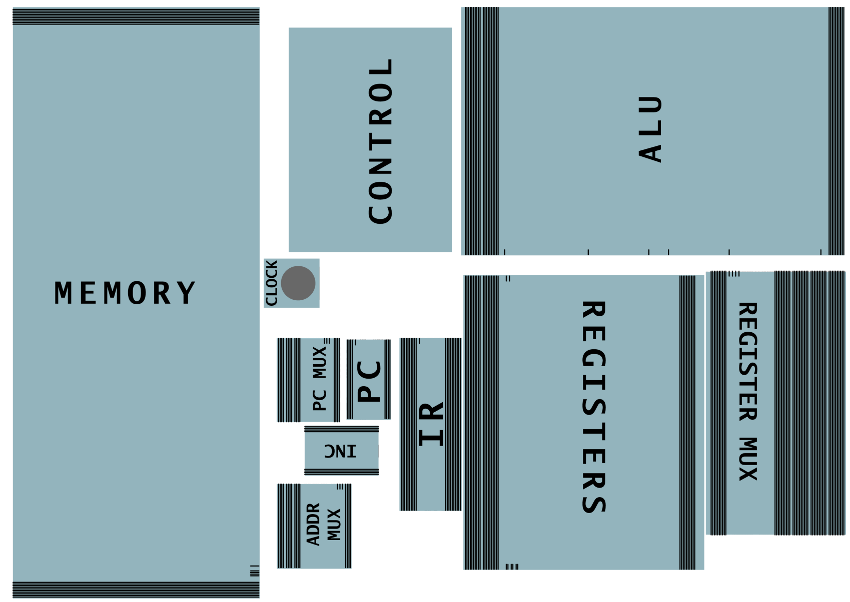


## TOY-Lite: Components






## TOY-Lite: Layout



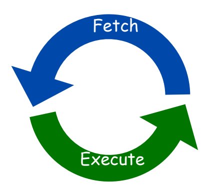



## TOY-Lite Datapath Requirements: Fetch

Basic machine operation is a cycle.
- Fetch
- Execute

Fetch.
- Memory[PC] to IR
- Increment PC

Execute.
- Datapath depends on instruction



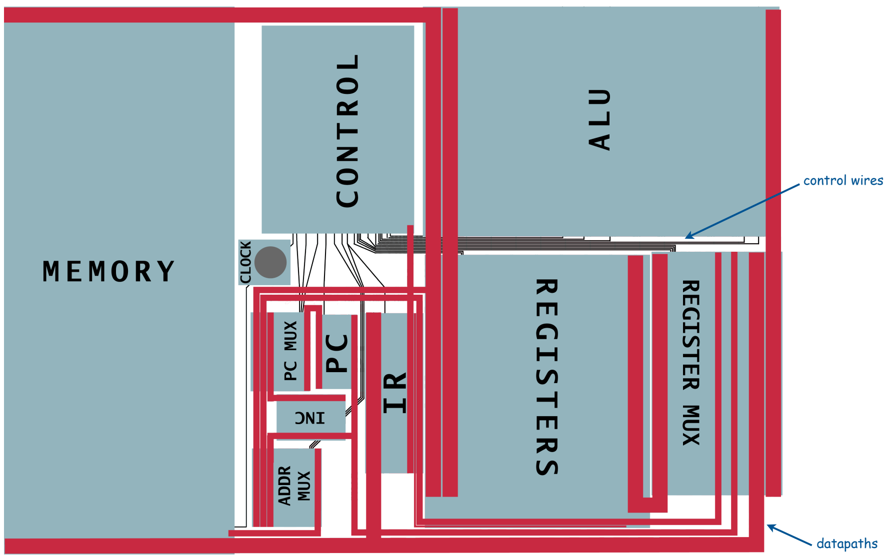



## TOY-Lite Datapath Requirements: Execute

Instructions determine datapaths and control sequences for execute

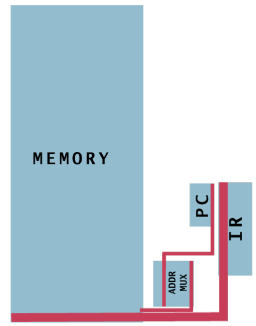
| | | |
|---|---|---|
| | | . . . |
| 0 | halt | . . . |
| 1 | add | IR opcode to control<br>control to ALU<br>two registers to ALU<br>ALU to register MUX |
| 2 | subtract | |
| 3 | and | |
| 4 | xor | |
| 5 | shift left | |
| 6 | shift right | |
| 7 | load address | . . . |
| 8 | load | . . . |
| 9 | store | . . . |
| A | load indirect | . . . |
| B | store indirect | . . . |
| C | branch zero | . . . |
| D | branch positive | |
| E | jump register | |
| F | jump and link | |

## TOY-Lite: Datapaths and Control



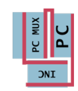



## Datapath: Add



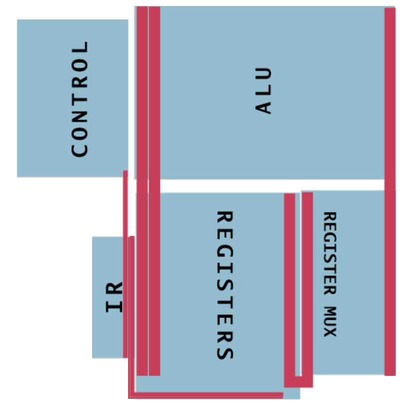

**fetch:**

Memory[PC] to IR




**increment**

increment PC

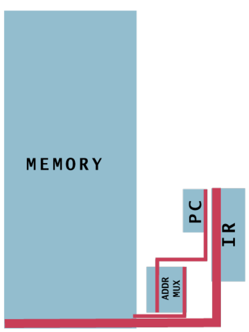



**execute:**

IR opcode to control
control to ALU
two registers to ALU
ALU to register MUX

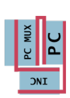


## Datapath: Load



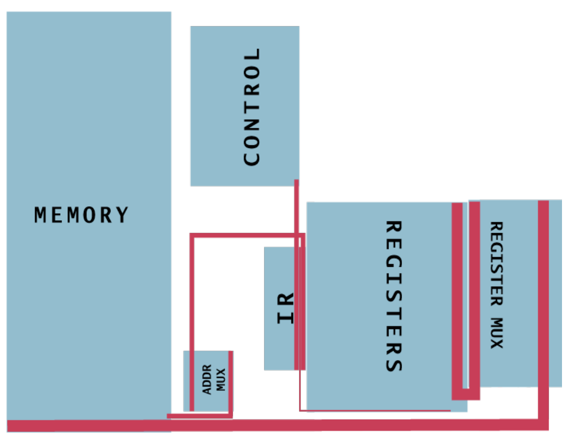

**fetch:**

Memory[PC] to IR




**increment**

increment PC




**execute:**

IR opcode to control
IR to addr MUX
memory to register MUX



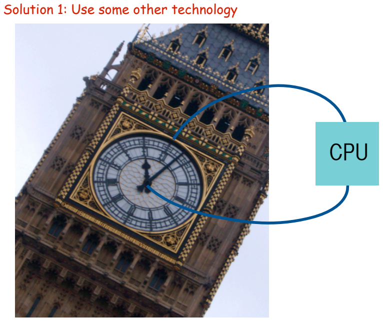
## Last step

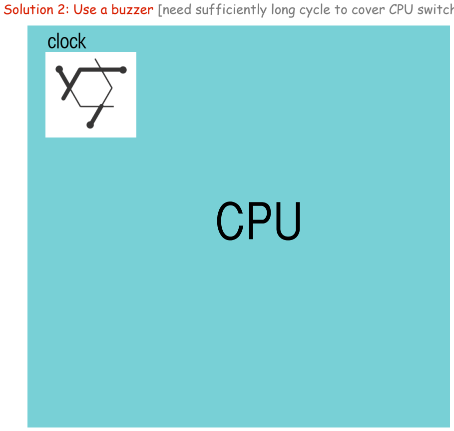
Control. Each instruction corresponds to a sequence of control signals.

Q. How do we create the sequence?
A. Need a "physical" clock.

Solution 2: Use a buzzer [need sufficiently long cycle to cover CPU switching]

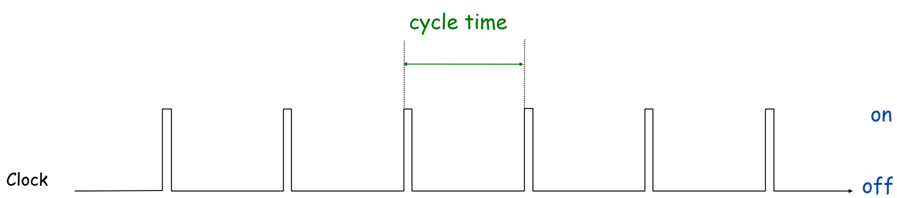
Solution 1: Use some other technology



## Clock


Clock.
- Fundamental abstraction: regular on-off pulse.
  - on: fetch phase
  - off: execute phase
- "external" device.
- Synchronizes operations of different circuit elements.
- Requirement: clock cycle longer than max switching time.






## How much does it Hert?

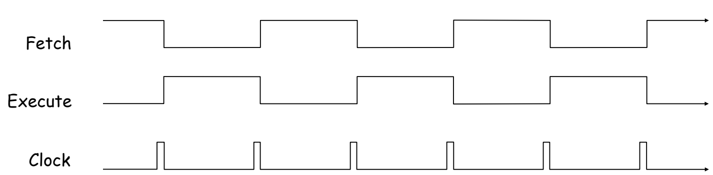
Frequency is inverse of cycle time.
- Expressed in hertz.
- Frequency of 1 Hz means that there is 1 cycle per second.
  - 1 kilohertz (kHz) means 1000 cycles/sec.
  - 1 megahertz (MHz) means 1 million cycles/sec.
  - 1 gigahertz (GHz) means 1 billion cycles/sec.
  - 1 terahertz (THz) means 1 trillion cycles/sec.

Heinrich Rudolf Hertz
(1857-1894)



## Clocking Methodology

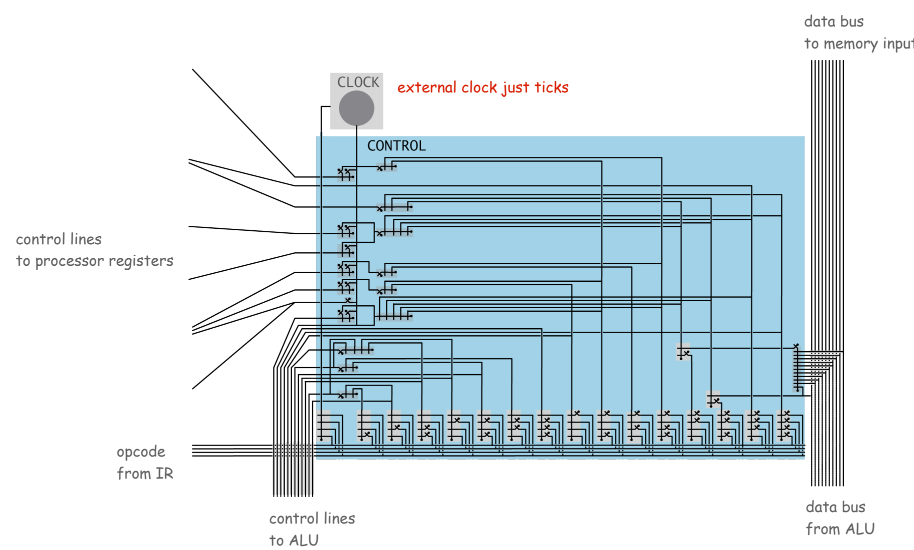
Two-cycle design.
- Each control signal is in one of four epochs.
  - fetch [set memory address from pc]
  - fetch and clock [write instruction to IR]
  - execute [set ALU inputs from registers]
  - execute and clock [write result of ALU to registers]






## One Last Combinational Circuit: Control

Control. Circuit that determines control line sequencing.



## Tick-Tock

CPU is a circuit, driven by a clock.

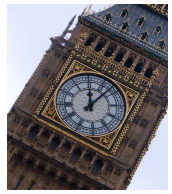
Switches initialize memory, PC contents

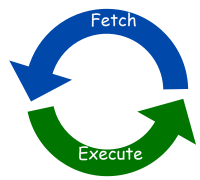
Clock ticks
- fetch instruction from memory[PC] to IR
- increment PC
- execute instruction

[details of instruction execution differ]

- fetch next instruction
- ...

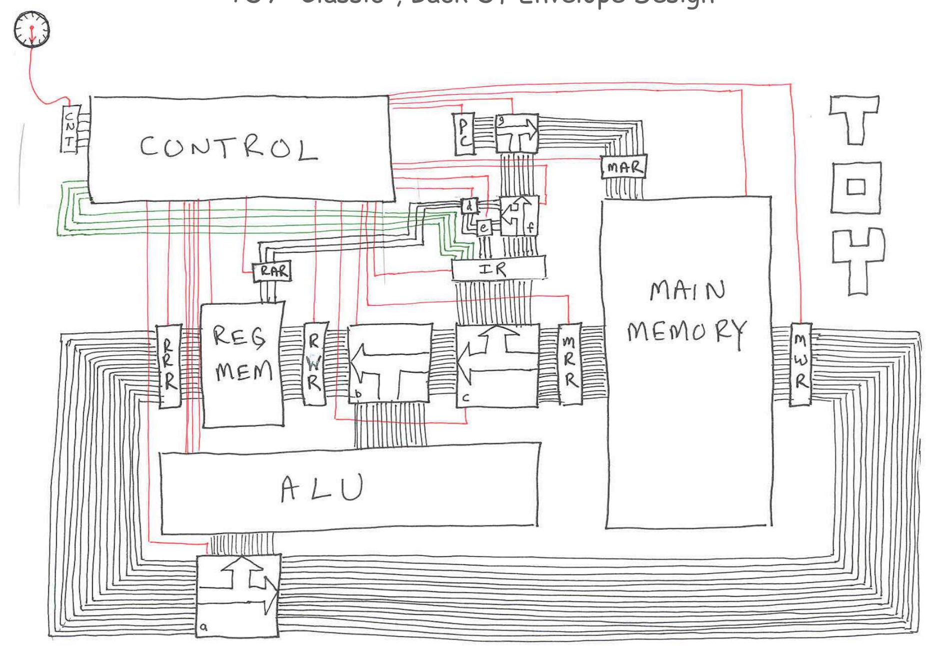
That's all there is to it!



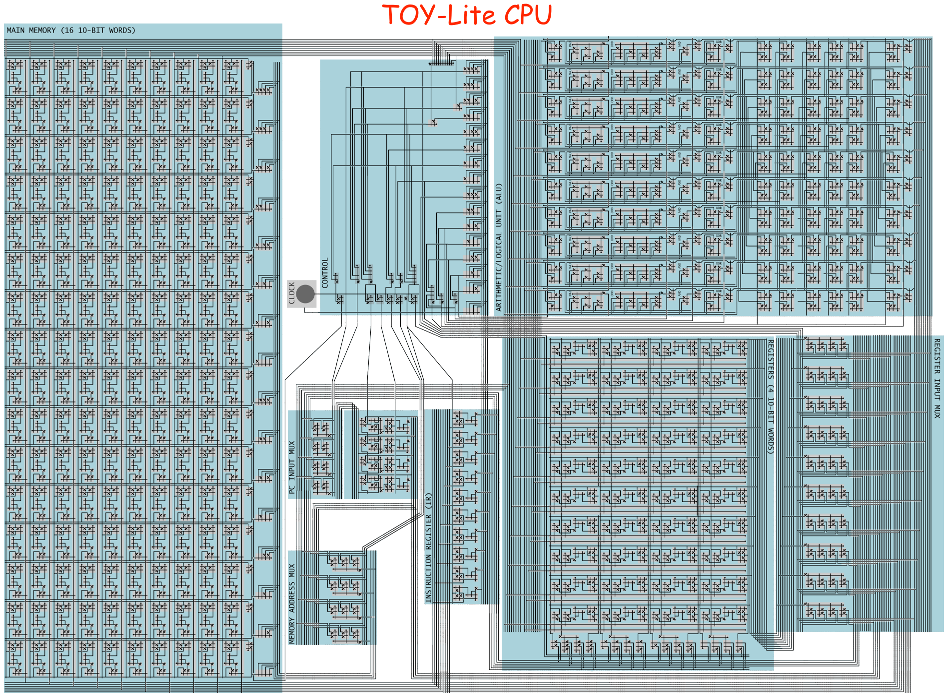



## TOY "Classic", Back Of Envelope Design



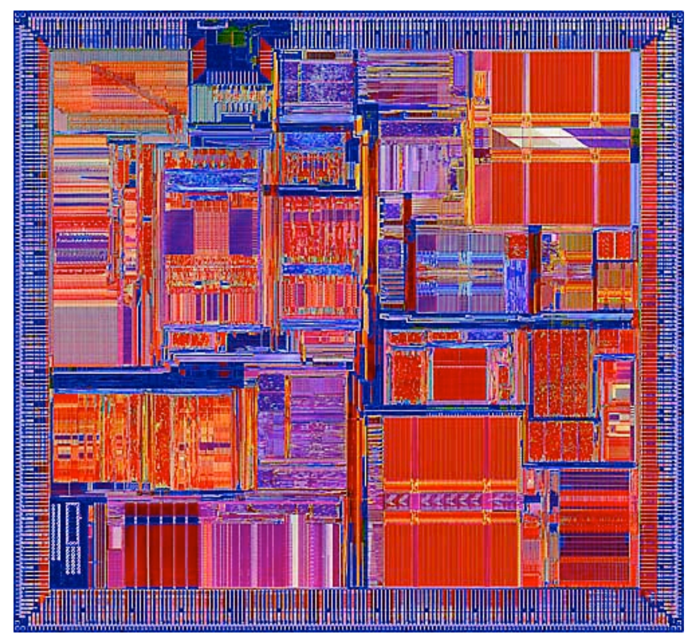



## TOY-Lite CPU






## Real Microprocessor (MIPS R10000)



## Layers of Abstraction

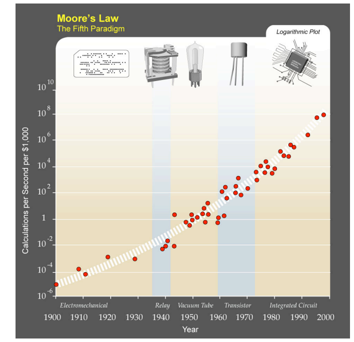
| Abstraction | Built From | Examples |
|---|---|---|
| Abstract Switch | raw materials | transistor, relay |
| Connector | raw materials | wire |
| Clock | raw materials | crystal oscillator |
| Logic Gates | abstract switches, connectors | AND, OR, NOT |
| Combinational Circuit | logic gates, connectors | decoder, multiplexer, adder, ALU |
| Sequential Circuit | logic gates, clock, connector | flip-flop |
| Components | decoder, multiplexer, adder, flip-flop | registers, ALU, counter, control |
| Computer | components | TOY |



## History + Future

Computer constructed by layering abstractions.
- Better implementation at low levels improves everything.
- Ongoing search for better abstract switch!

History.
- 1820s: mechanical switches.
- 1940s: relays, vacuum tubes.
- 1950s: transistor, core memory.
- 1960s: integrated circuit.
- 1970s: microprocessor.
- 1980s: VLSI.
- 1990s: integrated systems.
- 2000s: web computer.
- Future: quantum, optical soliton, …




Ray Kurzweil
http://en.wikipedia.org/wiki/Image:PPTMooresLawai.jpg