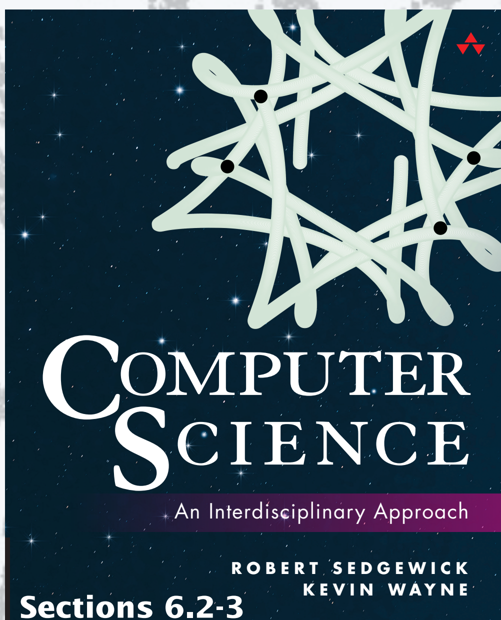
COMPUTER SCIENCE
SEDGEWICK / WAYNE



http://introcs.cs.princeton.edu

21. Central Processing Unit

COMPUTER SCIENCE
S E D G E W I C K / W A Y N E

21. CPU

- **Overview**
- Bits, registers, and memory
- Program counter
- Components and connections

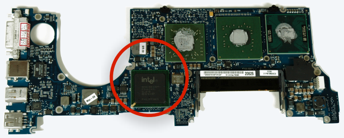
Let's build a computer!

CPU = Central Processing Unit

Computer
- Display
- Touchpad
- Battery
- Keyboard
- ...
- CPU (difference between a TV set and a computer)

Previous lecture
- Combinational circuits
- ALU (calculator)

This lecture
- Sequential circuits with *memory*
- CPU (computer)

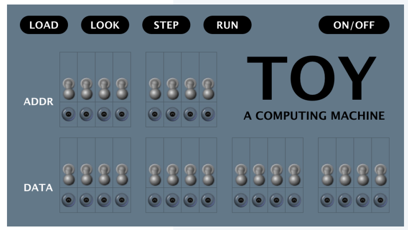


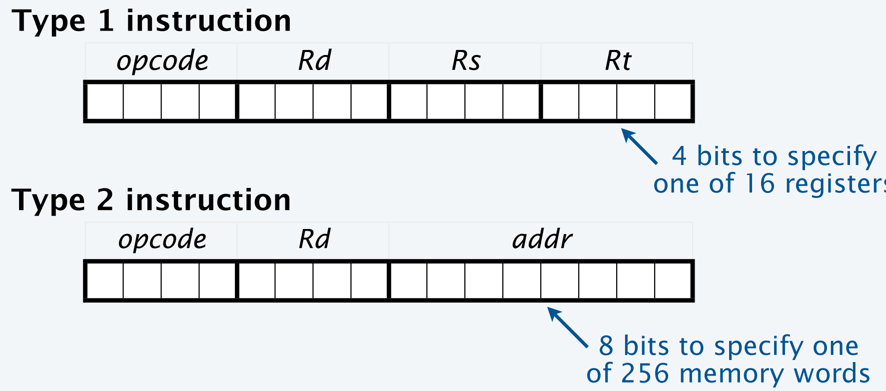
CPU

A smaller computing machine: TinyTOY

TOY instruction-set architecture.

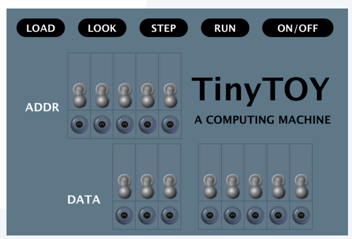
- 256 16-bit words of memory.
- 16 16-bit registers.
- 1 8-bit program counter.
- 2 instruction types
- 16 instructions.

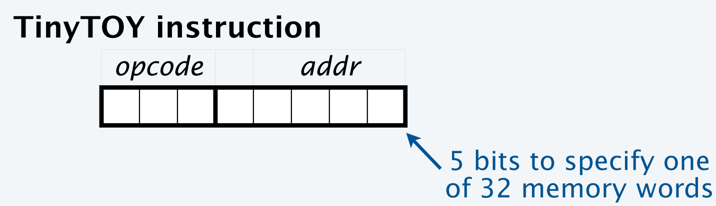




TinyTOY instruction-set architecture.

- 32 8-bit words of memory.
- 1 8-bit register.
- 1 5-bit program counter.
- 1 instruction type
- 8 instructions.





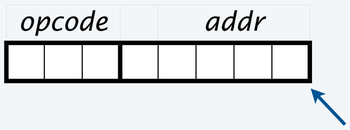
Purpose of TinyTOY. Illustrate CPU circuit design for a "typical" computer.

TinyTOY instruction set architecture

TinyTOY instructions.

- Halt
- Add
- AND
- XOR
- LOAD ADDRESS
- LOAD
- STORE
- BRANCH NEGATIVE

TinyTOY instruction



5 bits to specify one of 32 memory words

4 bits and 16 words in prototype machine

TinyTOY instruction-set architecture.

- 16 8-bit words of memory (expandable to 32).
- 1 8-bit register (R0).
- 1 4-bit program counter (expandable to 5).
- 8 instructions (only one type).

Review: the state of the machine

Contents of memory, registers, and PC at a particular time

- Provide a record of what a program has done.
- Completely determines what the machine will do.

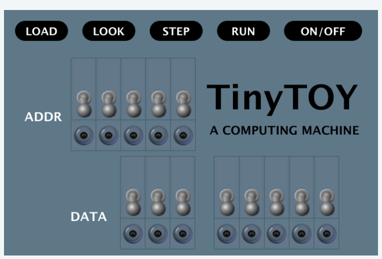
ALU and IR hold intermediate states of computation

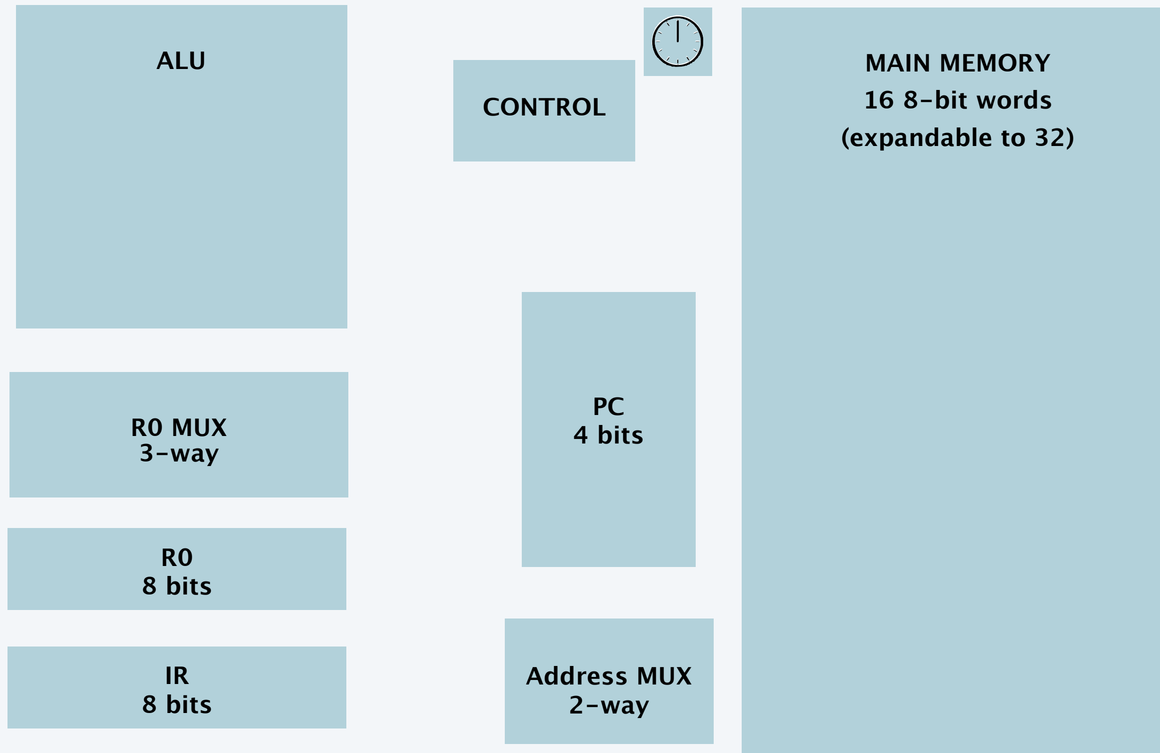
CPU circuit components for TinyTOY

TinyTOY CPU

- ALU (adder, AND, XOR)
- Memory
- Register (R0)
- PC (with incrementer)
- IR
- *MUXes (switch circuits)*
- *Control*
- *Clock*

stay tuned (this lecture)





Goal. Complete CPU circuit for TinyTOY (same design extends to TOY and to your computer).

Perspective

Q. Why TinyTOY?

A. TOY circuit width would be about 5 times TinyTOY circuit width.

Sobering fact. The circuit for your computer is *hundreds* to *thousands* of times wider.

Reassuring fact. Design of all three is based on the same fundamental ideas.

21. CPU

- Overview
- **Bits, registers, and memory**
- Program counter
- Connections

Sequential circuits

Q. What is a sequential circuit?

A. A digital circuit (all signals are 0 or 1) *with feedback* (loops).

Q. Why sequential circuits?

A. *Memory* (difference between a DFA and a Turing machine).

Basic abstractions

- On and off.
- Wire: Propagates an on/off value.
- Switch: Controls propagation of on/off values through wires.
- Flip-flop: *Remembers* a value.

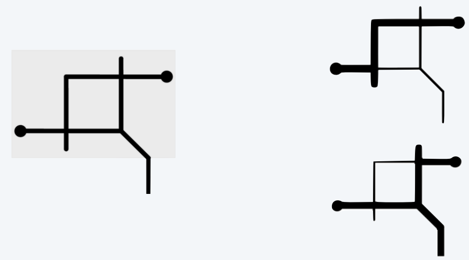
Simple circuits with feedback

Loops in circuits lead to time-varying behavior

- *Sequence* of switch operation matters.
- Need tight control (see next slide).

Example 1. Two switches, each blocked by the other.

- State determined by whichever switches first.
- Stable (once set, state never changes).
- *Basic building block for memory circuits.*



Example 2. Three switches, blocked in a cycle.

- State determined by whichever switches first.
- *Not* stable (cycles through states).

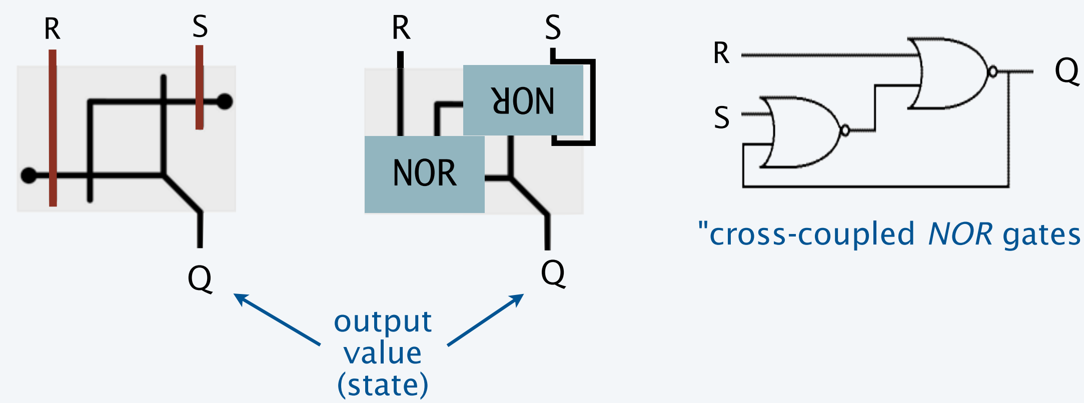
a "buzzer" →

A new ingredient: Circuits with memory

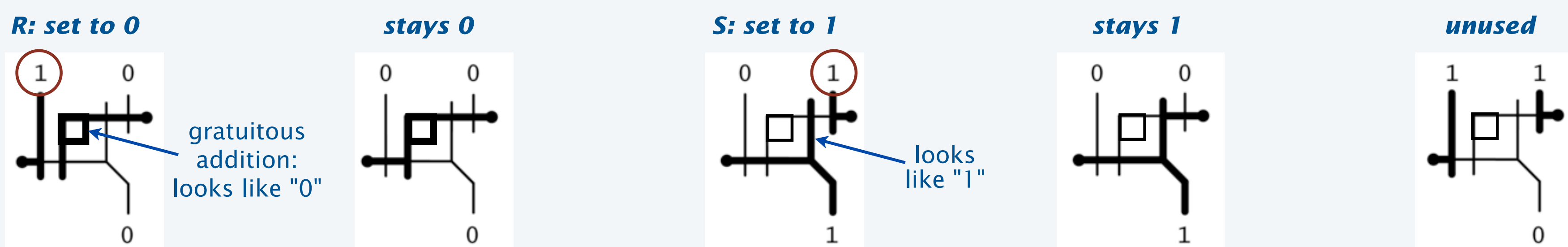
An *SR flip-flop* controls feedback.

- Add control lines to switches in simple feedback loop.
- R (reset) sets state to 0.
- S (set) sets state to 1.
- Q (state) is always available.

component level **classic notation**



examples

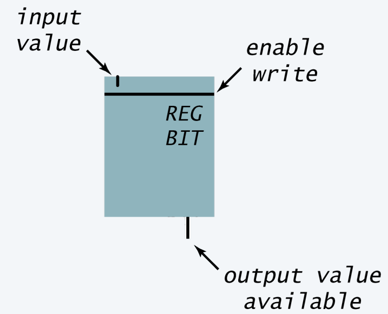


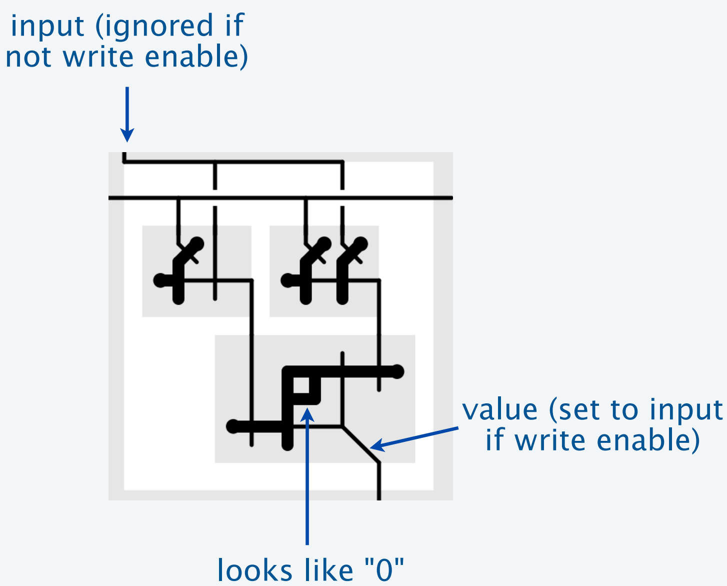
Caveat. Timing of switch vs. propagation delay.

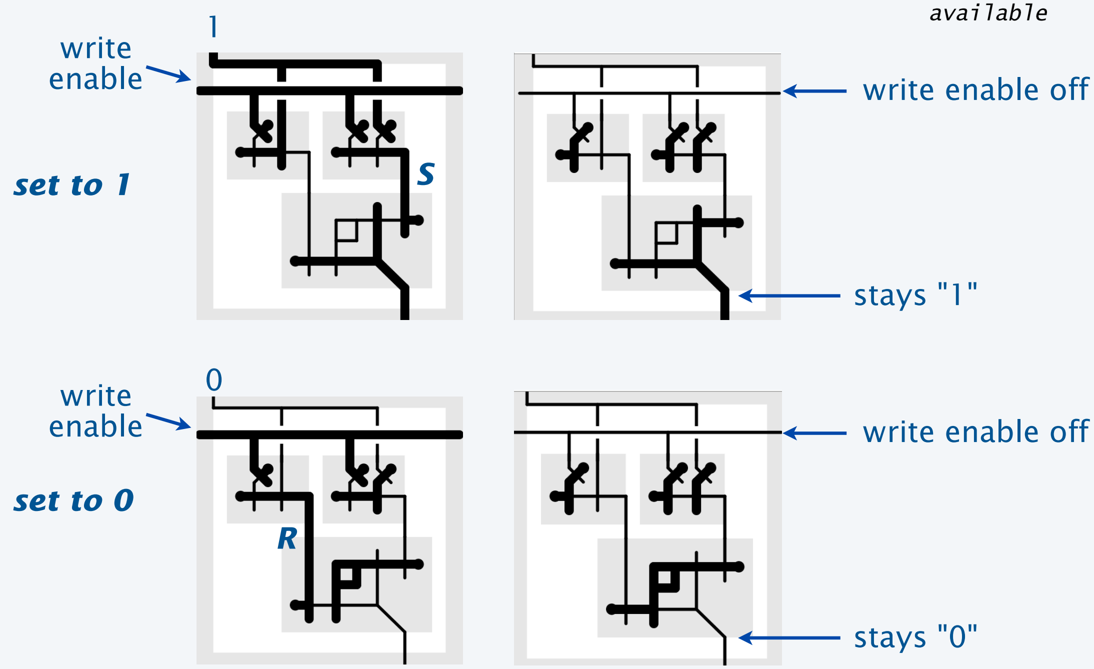
One bit in a register

Add logic to an SR flip-flop for more precise control

- Provide data value on an input wire instead of using S and R controls.
- Use *enable write* signal to control timing of write.
- Flip-flop value is always available.



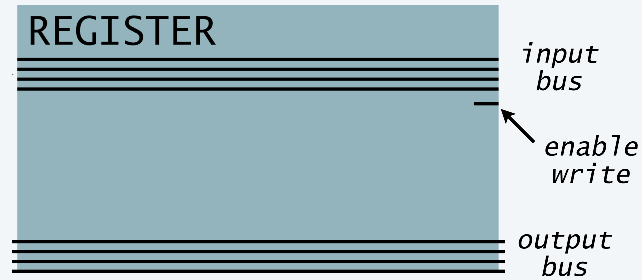




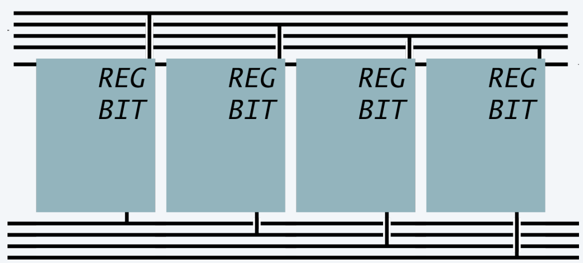
Registers

Register

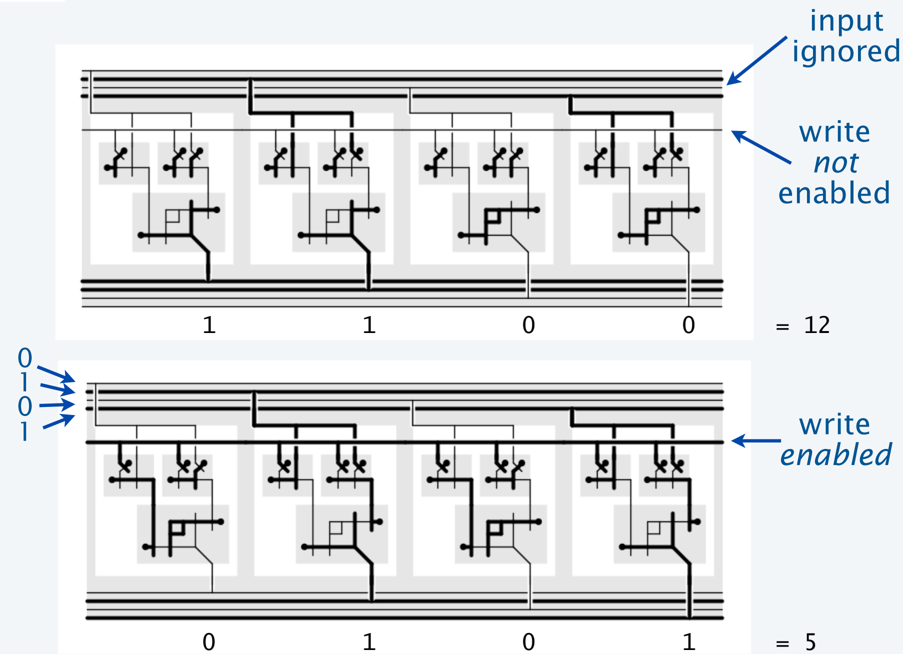
- Holds *W* bits.
- Input and output on *W*-wire busses.
- Register contents always available on output bus.
- Enable write puts *W* input bits into register.



component level



examples



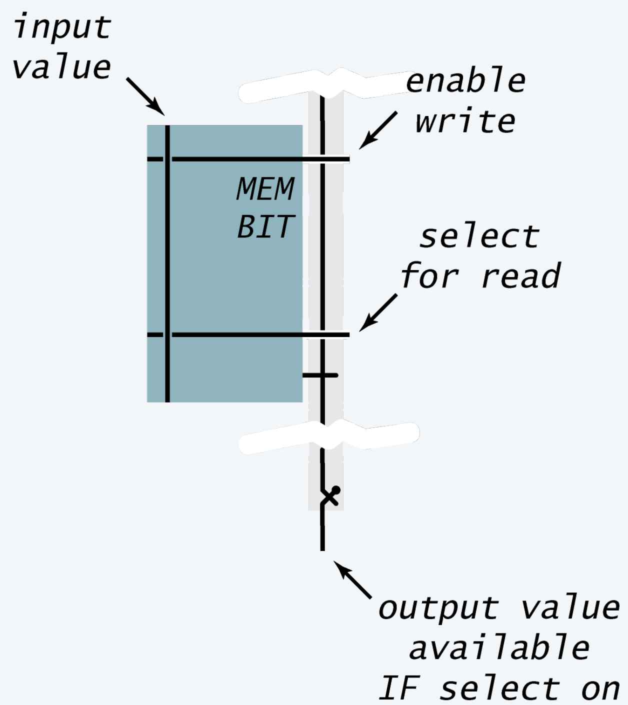
TinyTOY registers

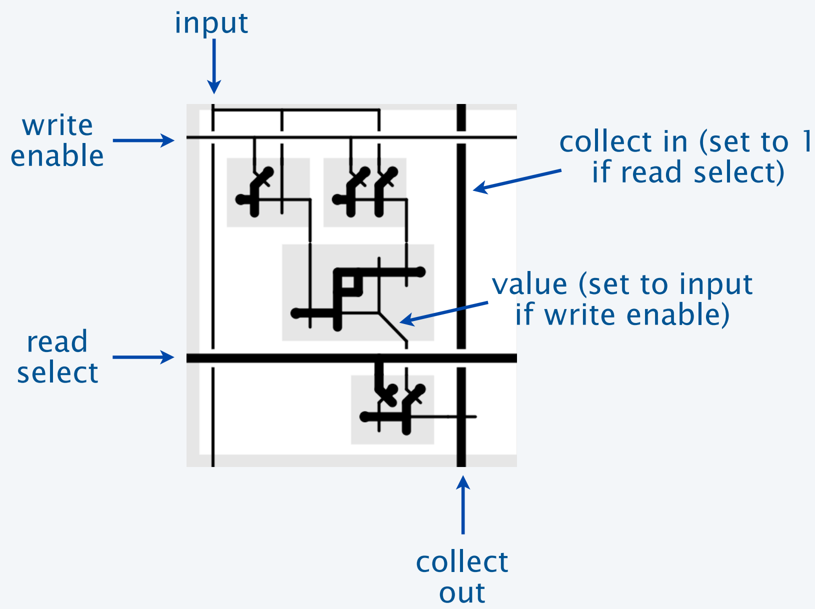
- PC holds 4-bit address.
- IR holds 8-bit instruction.
- R0 holds 8-bit data value.

One bit in main memory

Add a selection mechanism

- Flip-flop value is *not* always available.
- Use *select for read* signal to make it available.
- "1-hot" OR to collect the one bit value that is selected.



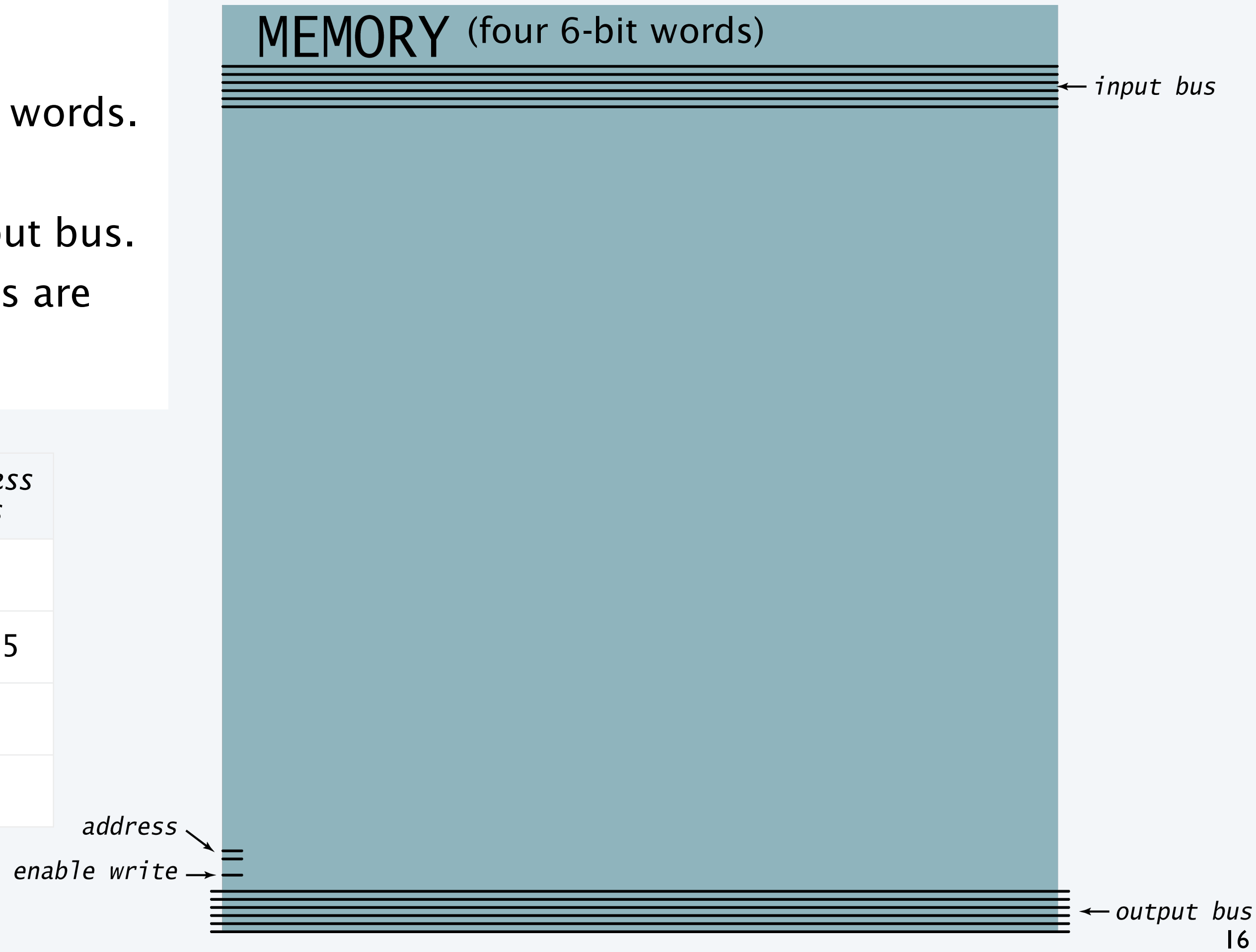


Main memory: interface

Main memory.
- *N* words; each stores *W* bits.
- Read and write data to one of *N* words.
- Address inputs select one word.
- Addressed word always on output bus.
- When write enabled, *W* input bits are copied into addressed word.

	number of words	*bits per word*	*address bits*
this slide	4	6	2
tinyTOY	16 or 32	8	4 or 5
TOY	256	16	8
your computer	1 billion	64	32

interface



Main memory bank: component level

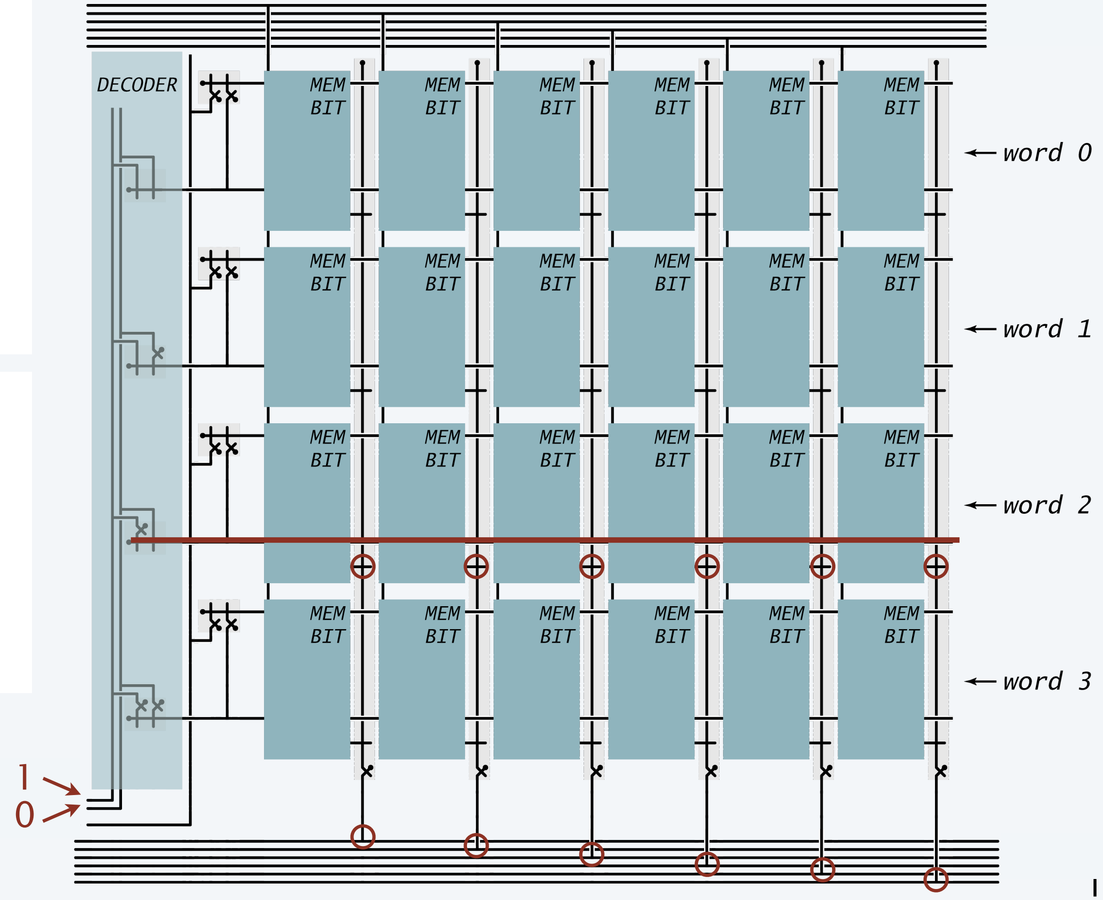
Main memory.

- *N* words; each stores *W* bits.
- Read and write data to one of *N* words.
- Address inputs select one word.
- Addressed word always on output bus.
- When write enabled, *W* input bits are copied into addressed word.

Basic mechanisms

- A decoder uses address to switch on one line (through the addressed word)
- "1-hot" OR gates at each bit position take word contents to the output bus.

component-level implementation (four six-bit words)

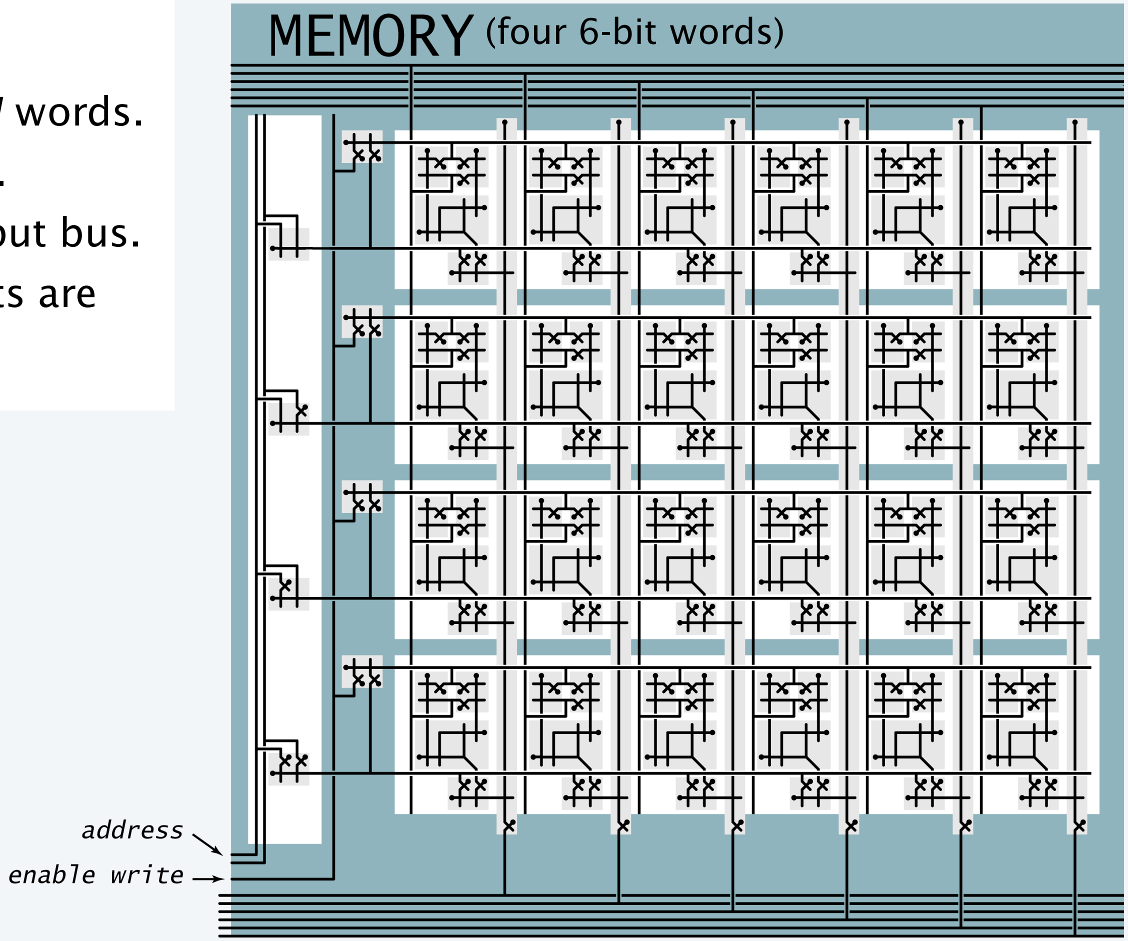


Main memory bank: switch level

Main memory.

- *N* words; each stores *W* bits.
- Read and write data to one of *N* words.
- Address inputs select one word.
- Addressed word always on output bus.
- When write enabled, *W* input bits are copied into addressed word.

switch–level implementation



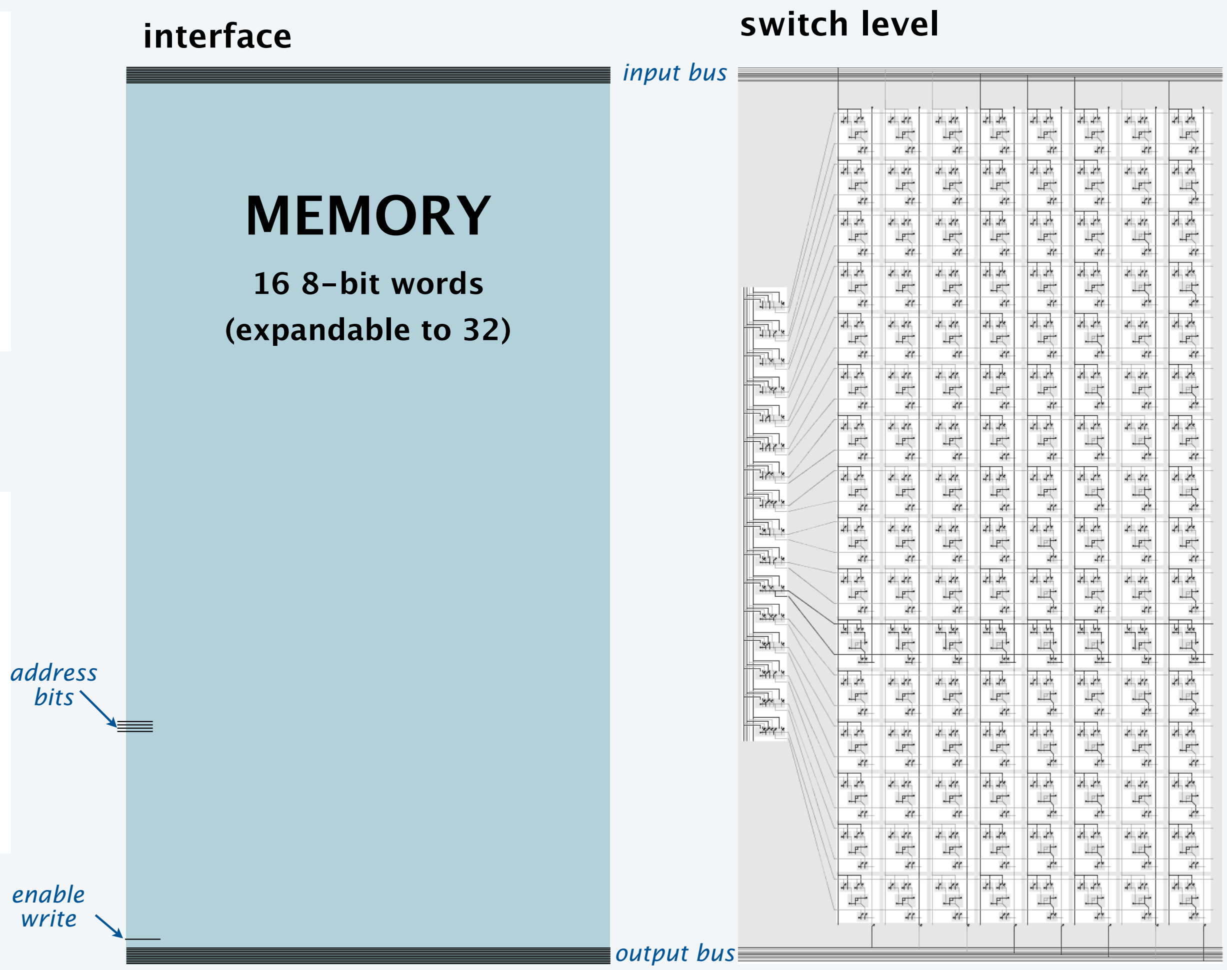
TinyTOY main memory bank

Interface

- Input bus for "store"
- Output bus for "load"
- Address bits to select a word
- *Enable write* control signal

Connections

- Input bus from registers
- Output bus to IR and R0
- Address bits from PC, IR, R0
- *Enable write* from "control"



COMPUTER SCIENCE
S E D G E W I C K / W A Y N E

21. CPU

- Overview
- Bits, registers, and memory
- **Program counter**
- Components and connections

Designing a digital circuit: overview

Steps to design a digital (sequential) circuit

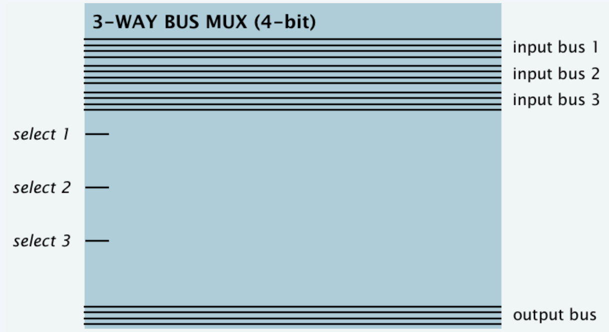
- Design interface: input busses, output busses, control signals.
- Determine components.
- Determine datapath requirements: "flow" of bits.
- Establish control sequence.

Warmup. Design TinyTOY program counter (PC). ← Three components and three control signals

Another useful combinational circuit: Multiplexer

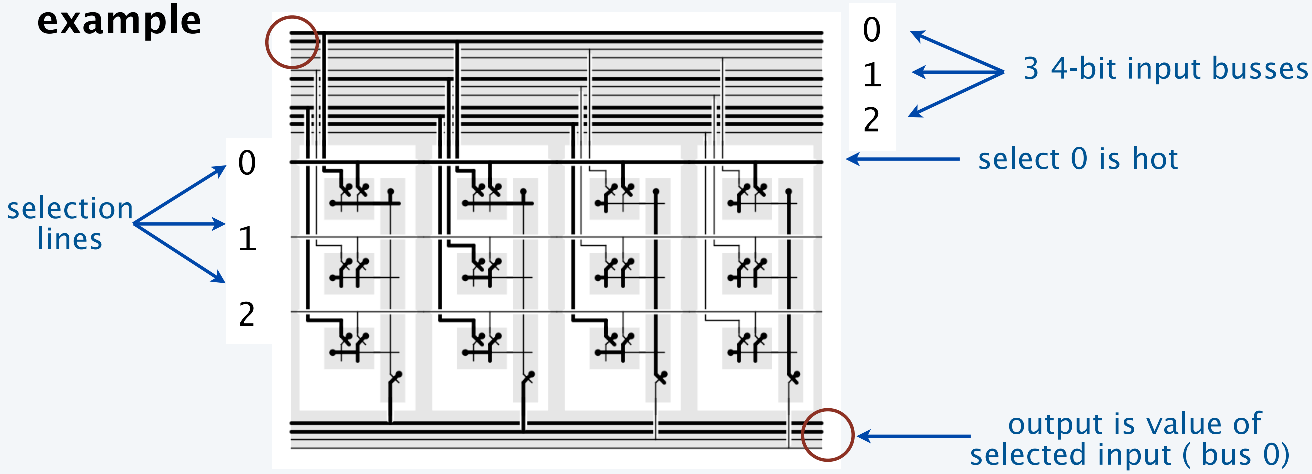
Bus multiplexer (MUX).

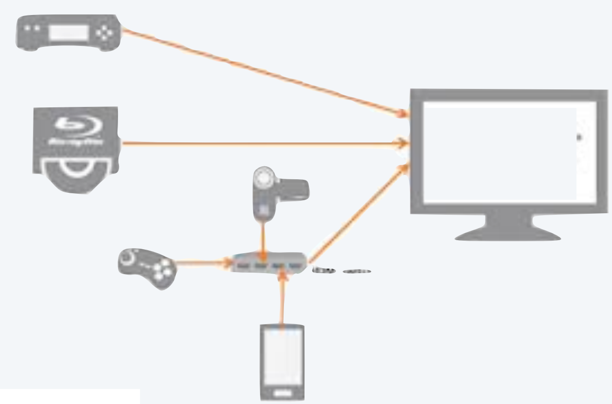
- Combinational circuit to select among input buses.
- Exactly one select line *i* is activated.
- Puts bit values from input bus *i* onto output bus.



Note: MUX in text takes binary select specification

example



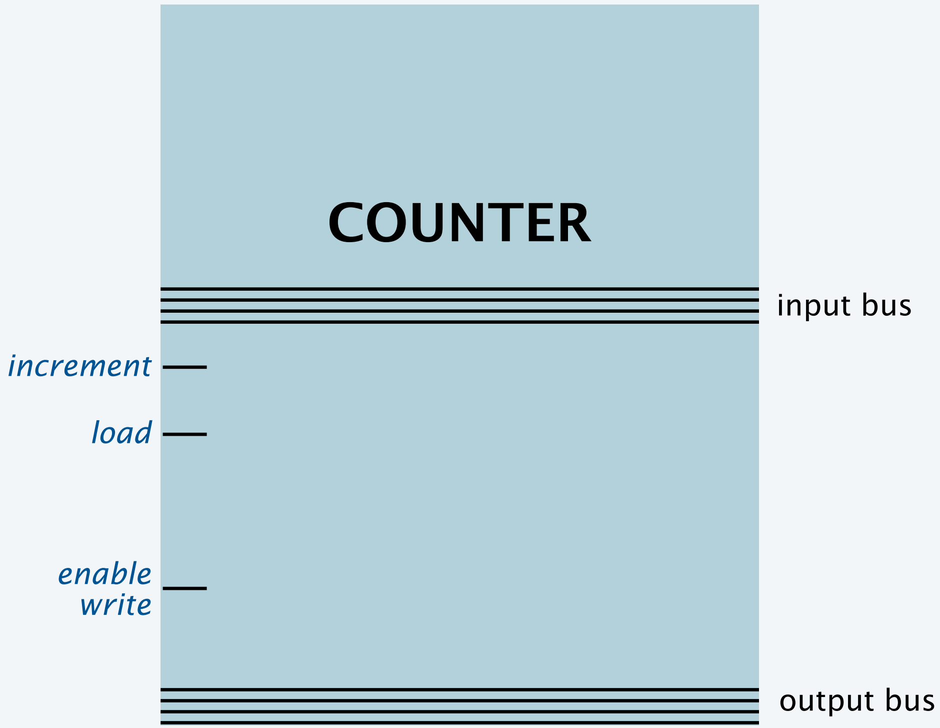


Typical use. Connect a component in different ways at different times.

Counter interface

A *Counter* holds a value and supports 3 control signals:

- *Increment.* Add 1 to value.
- *Load.* Set value from input bus.
- *Enable write.* Make value available on output bus.



Components inside

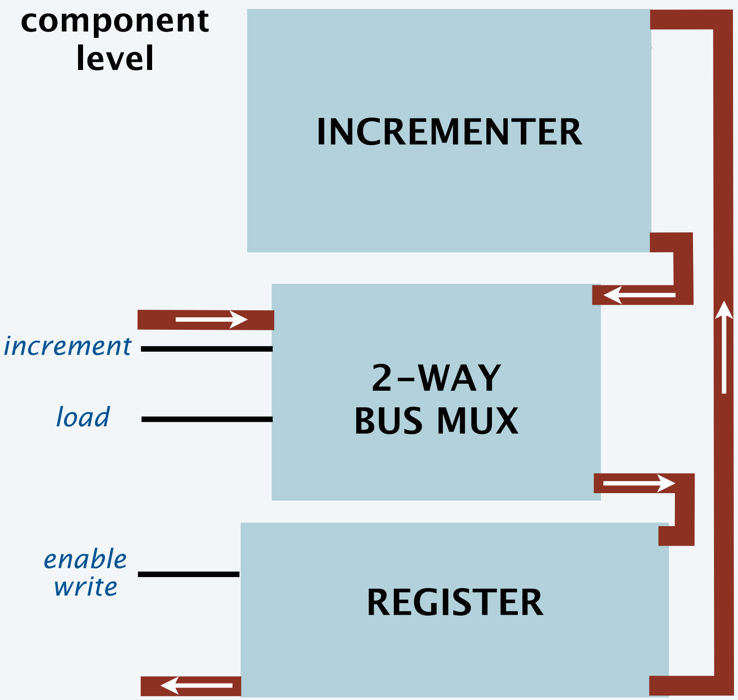
- Register.
- Incrementer (add 1).
- 2-way MUX.

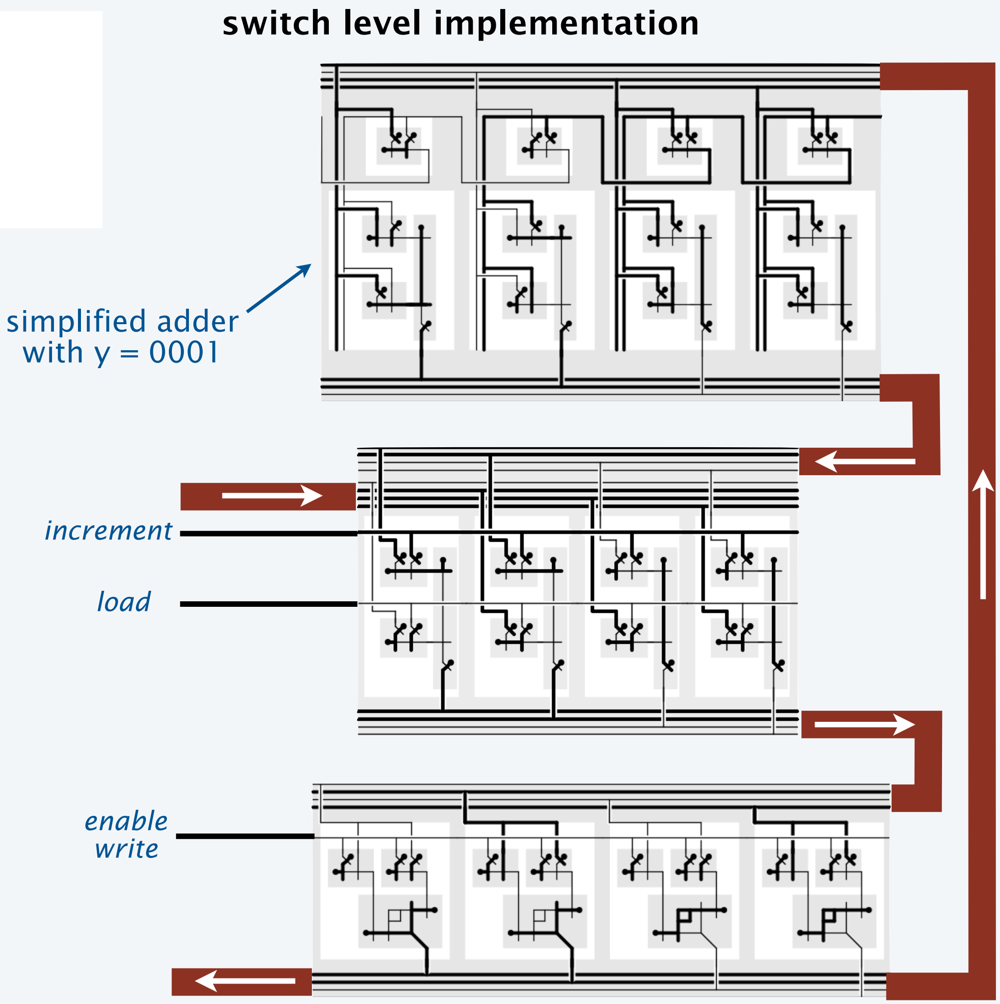
TinyTOY PC: 4-bit counter

Counter layout and implementation

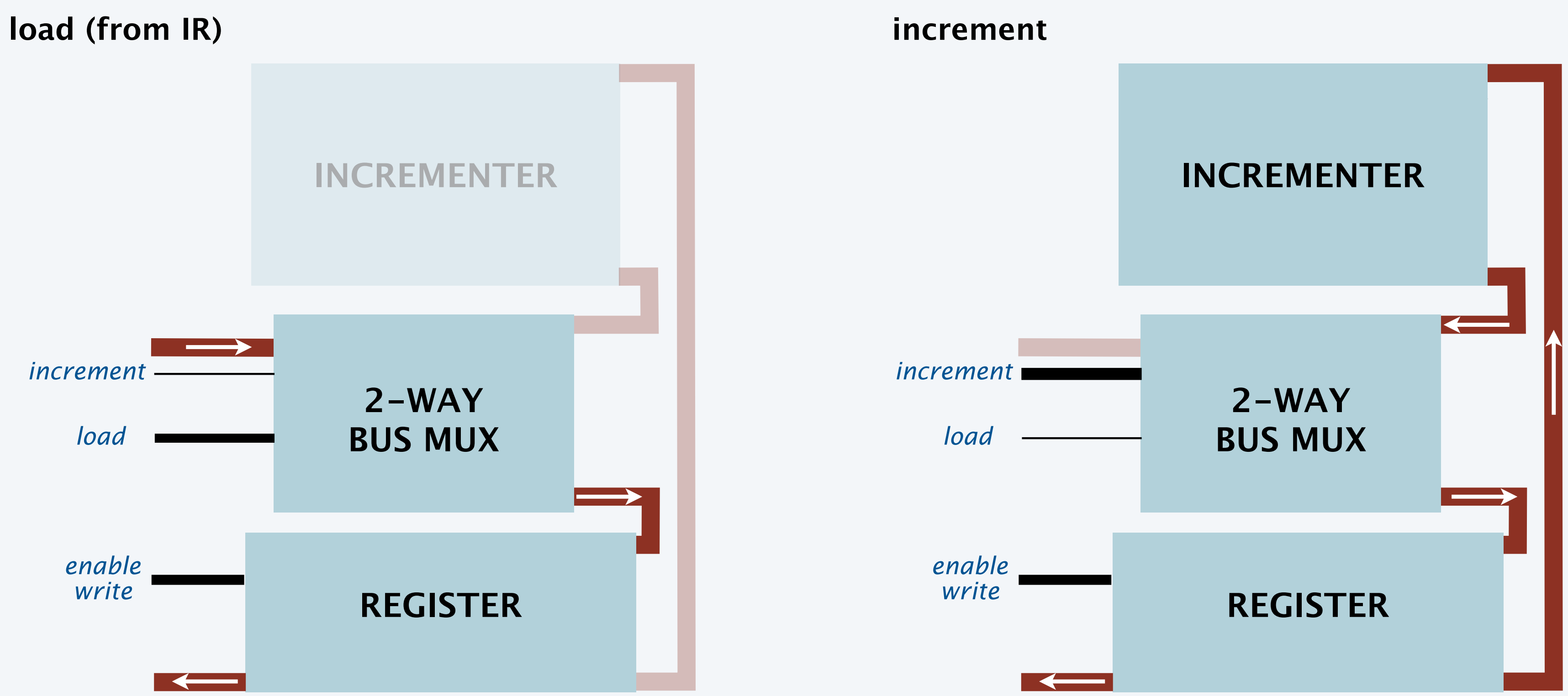
Layout and connections establish

- data paths (busses)
- control signals





Counter operation



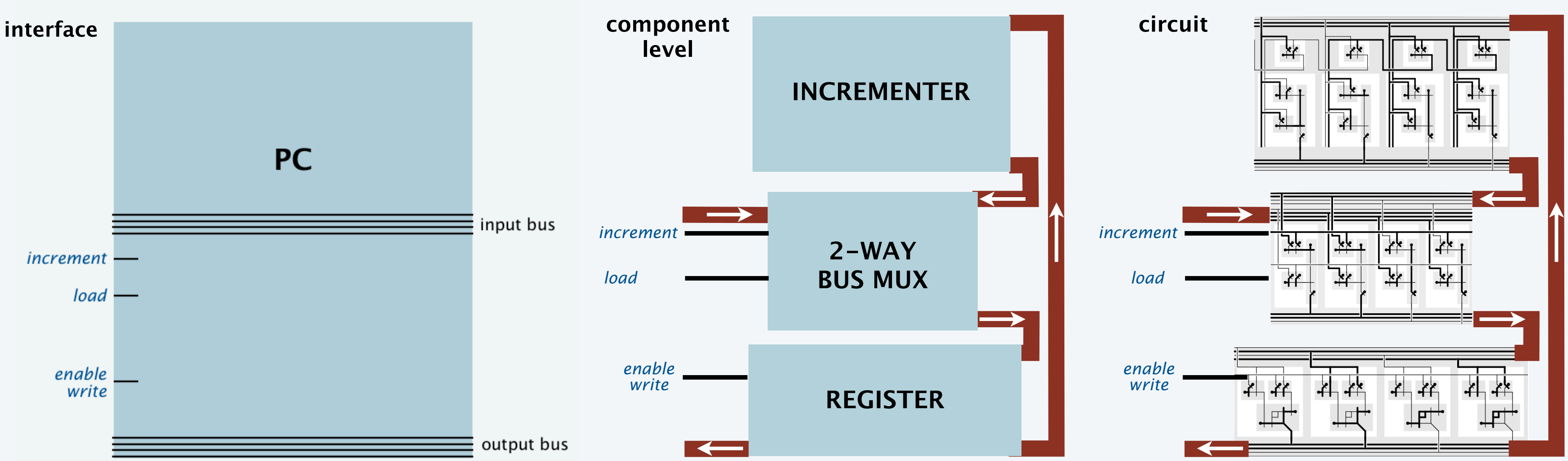
Important note: *Enable write* is a pulse, to avoid feedback (stay tuned)

Summary of TinyTOY PC (counter) circuit

The *PC* is three components and supports three control signals:

- *Load, then enable write.* Set value from input bus (example: branch instruction).
- *Increment, then enable write.* Add one to value.

Value is written to the PC and available on output bus in both cases.



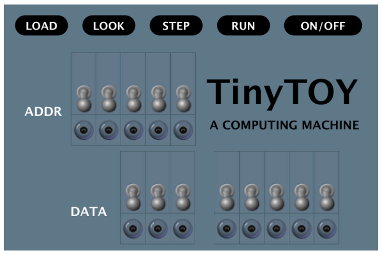
Next. CPU circuit (10 components, 20+ control signals).

21. CPU

- Overview
- Bits, registers, and memory
- Program counter
- **Components and connections**

TinyTOY: Interface

CPU is a circuit inside the machine

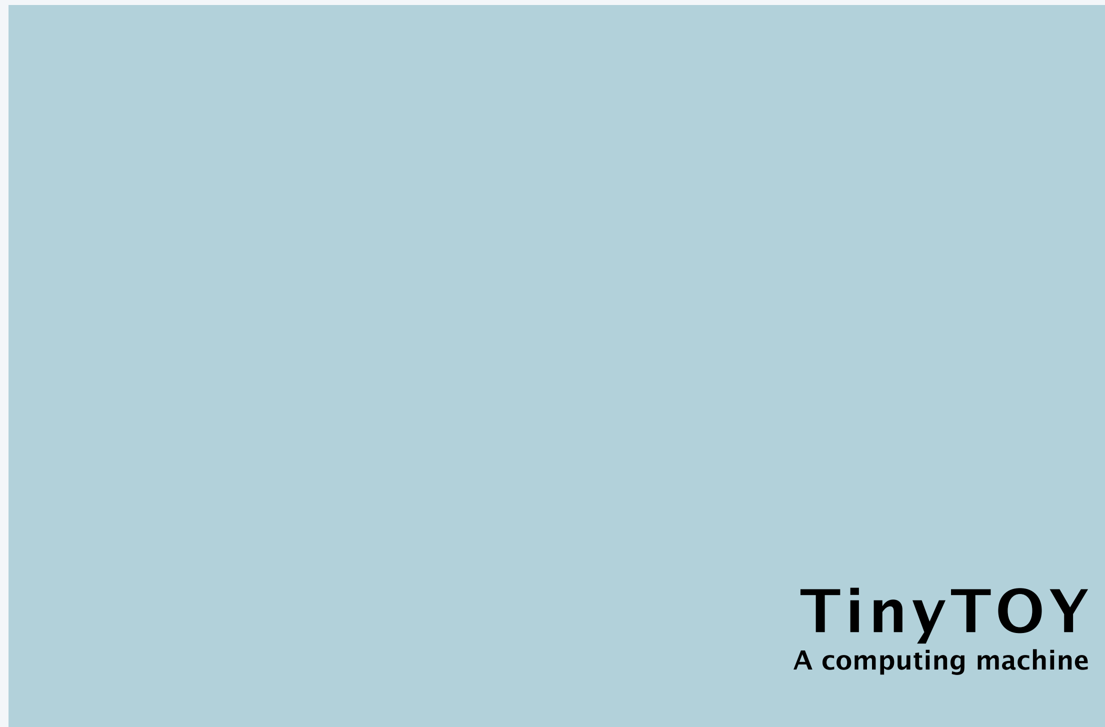


Interface to outside world

- Switches and lights
- ON/OFF
- RUN

Connections to outside (omitted)

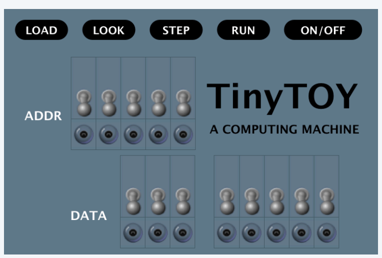
- ADDR to PC
- DATA to memory bank input bus
- Buttons to control lines that activate memory load/store

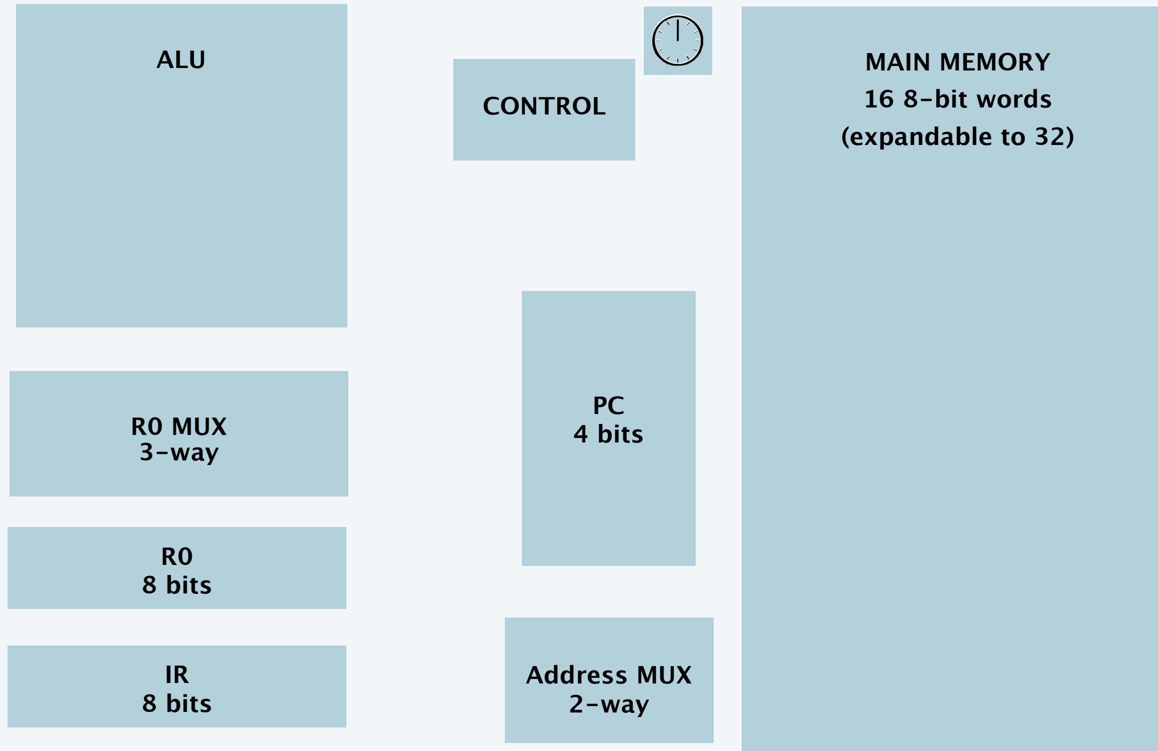


Review: CPU circuit components for TinyTOY

TinyTOY CPU

- ALU (adder, AND, XOR)
- Memory
- Register (R0)
- PC (with incrementer)
- IR
- MUXes (switch circuits)
- *Control*
- *Clock*





Review: Program counter and instruction register

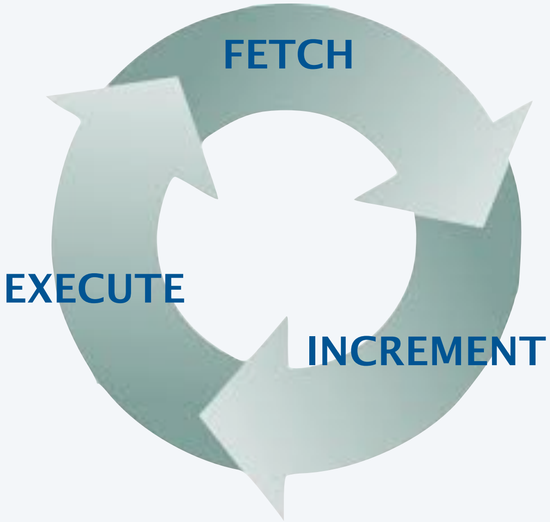
TOY operates by executing a sequence of instructions.

Critical abstractions in making this happen

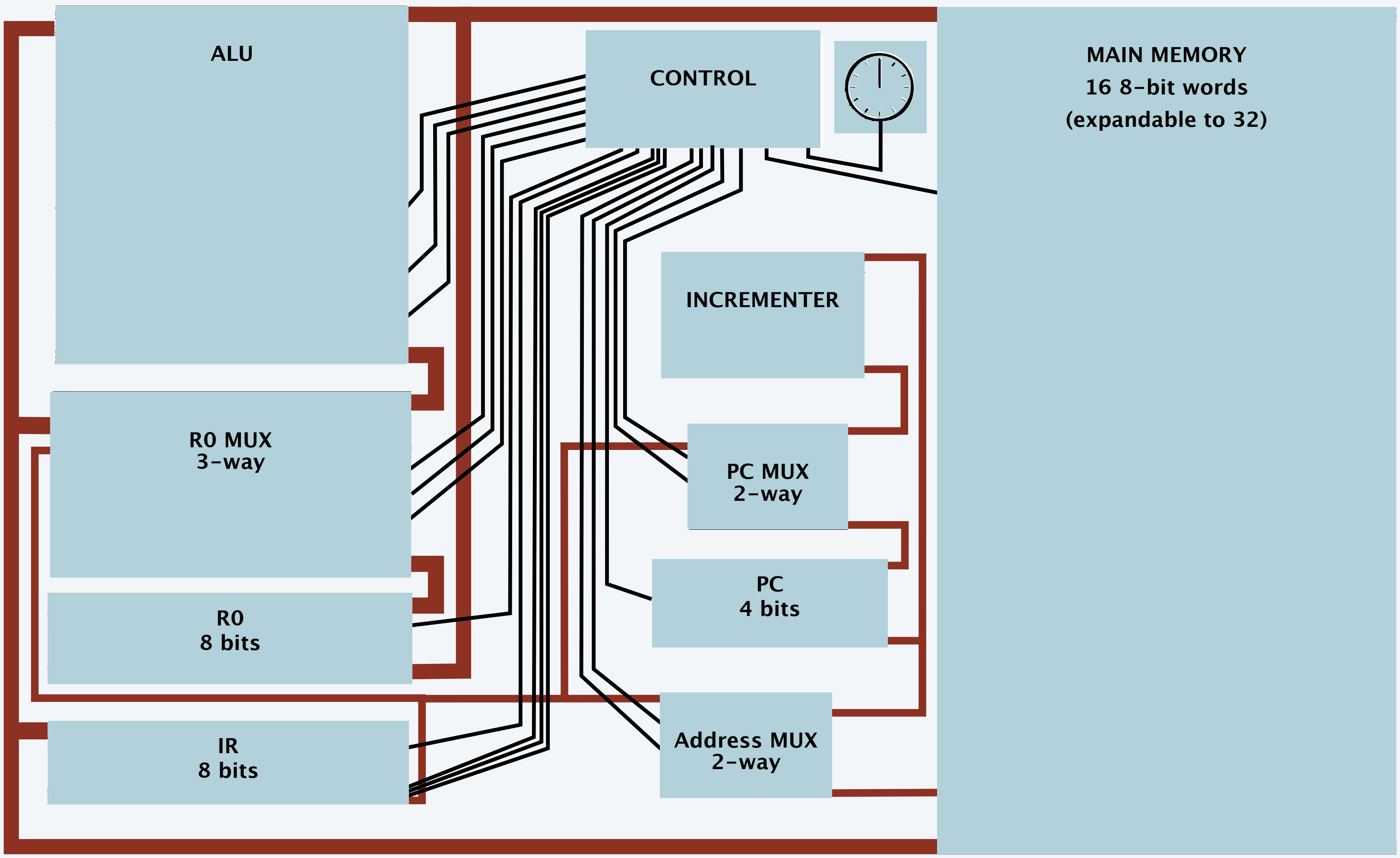
- Program Counter (PC). Memory address of next instruction.
- Instruction Register (IR). Instruction being executed.

Fetch-increment-execute cycle

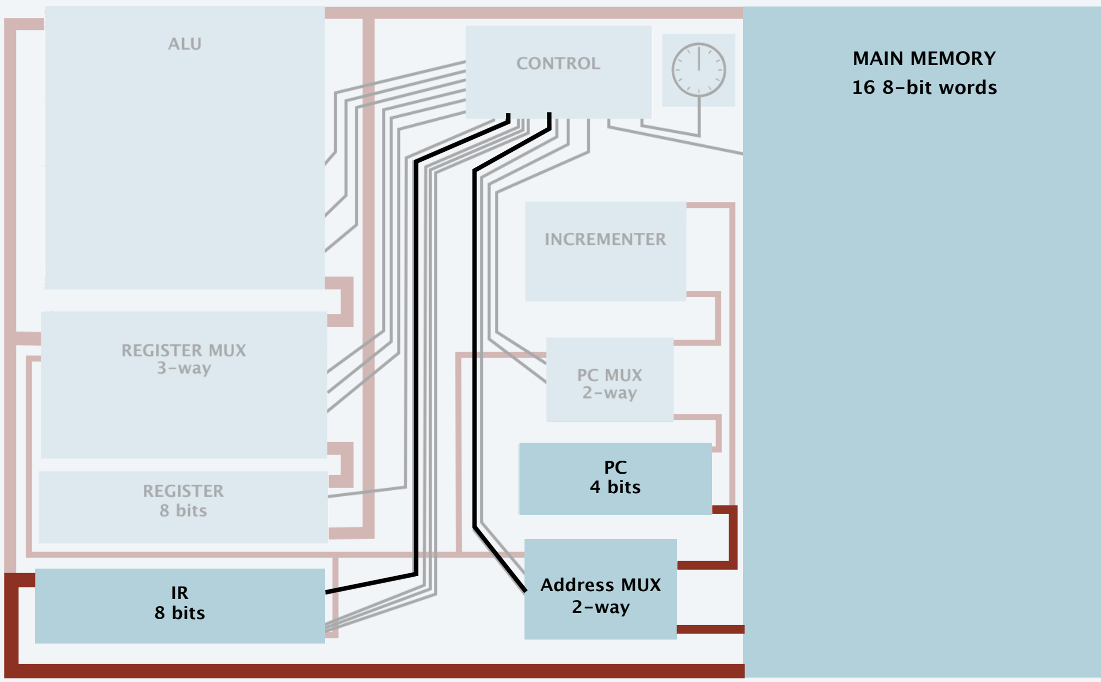
- Fetch: Get instruction from memory into IR.
- Increment: Update PC to point to *next* instruction.
- Execute: Move data to or from memory, change PC, or perform calculations, as specified by IR.



TinyToy data paths and control lines



Fetch (every instruction)



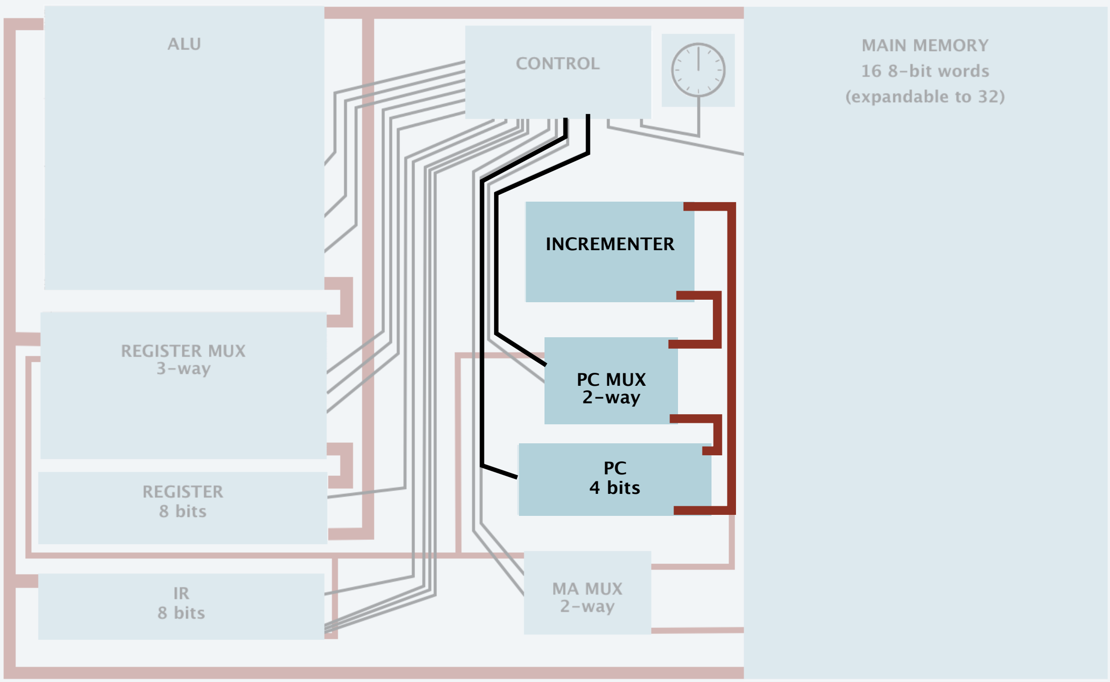
Data paths

- PC to Address MUX
- Address MUX to memory
- Memory to IR

Control signals

- Address MUX select 1
- IR Write Enable

Increment (review)



Data paths

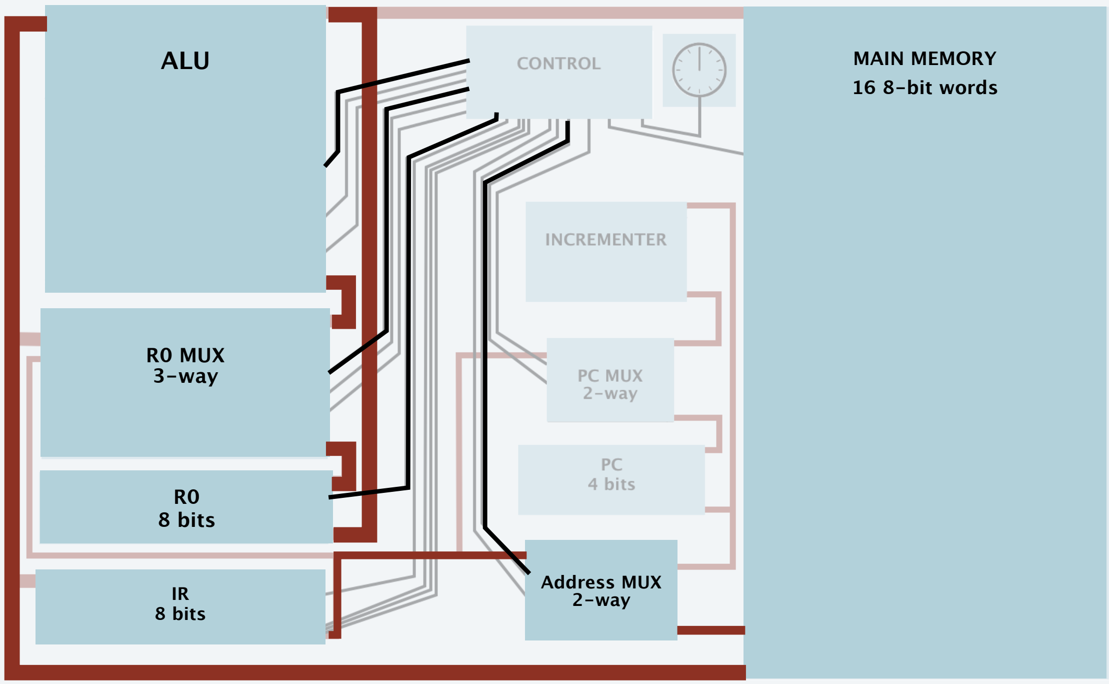
- PC to Incrementer
- Incrementer to PC MUX
- PC MUX to PC

Control signals

- PC MUX select 0
- PC write enable

Note: Occurs during execution of every instruction (except branch).

Add instruction



Data paths

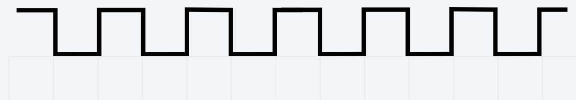
- IR to Address MUX
- Address MUX to memory
- Memory to ALU
- R0 to ALU
- ALU to R0 MUX
- R0 MUX to R0

Control signals

- ALU Select ADD
- Address MUX select 0
- R0 MUX Select 0
- R0 Write Enable

Clock

A CLOCK provides a regular ON-OFF pulse.

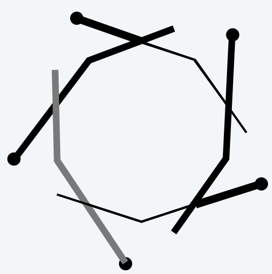


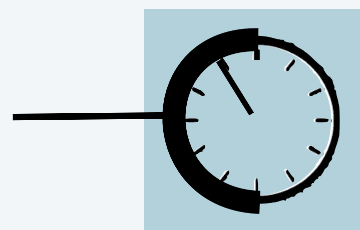
Requirement. Clock cycle longer than max switching time.

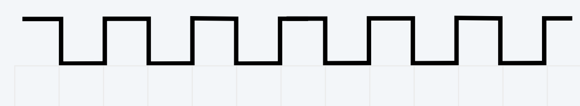
Q. How to implement a clock?

A. Use an external device.

A. Use a *buzzer circuit*.

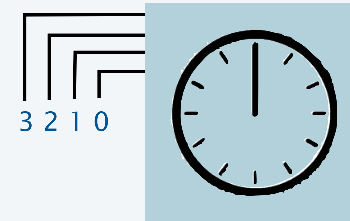




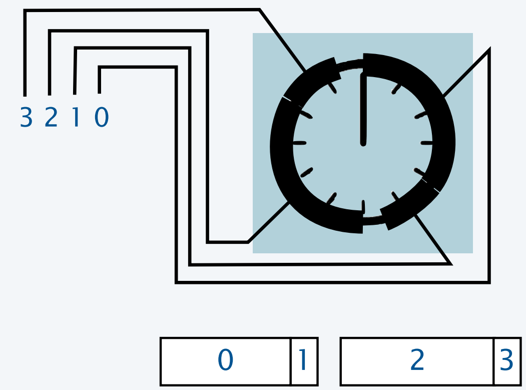


TinyTOY clock

Two-cycle design. Each control signal is in one of four epochs.



epoch	*name*	*example*
0	fetch	set MA from PC
1	fetch/write	load IR from memory
2	execute	set ALU inputs
3	execute/write	load R0 from ALU



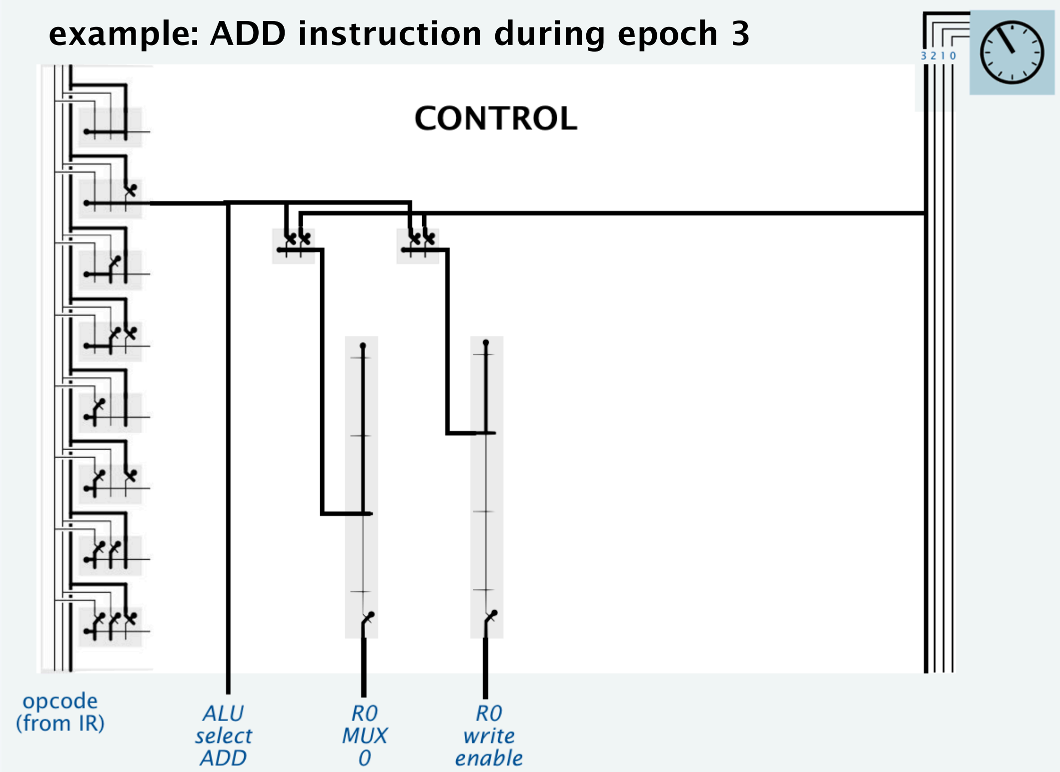
Key feature. A *sequence* of signals.

One final combinational circuit: Control

Control. Circuit that determines *control line sequencing*.

Control line sequencing

- Clock cycles through four epochs, raising one line at a time
- Determines sequence of at most 4 sets of control lines for each instruction



Key feature. A simple combinatorial circuit.

Tick-Tock

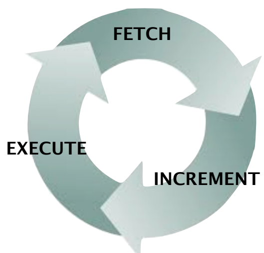
CPU is a circuit, driven by a clock.

Initalize via console switches.

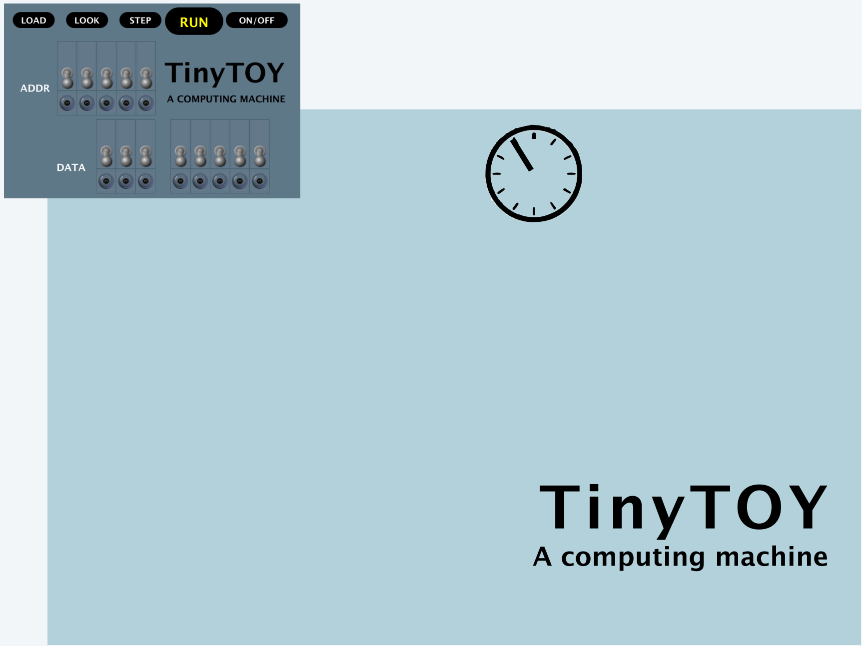
Press RUN: clock starts ticking

- *PC to mem addr MUX*
- *IR enable write*
- *PC increment*
- *PC enable write*

.
.
.
[details of instruction execution differ]
.
.
.

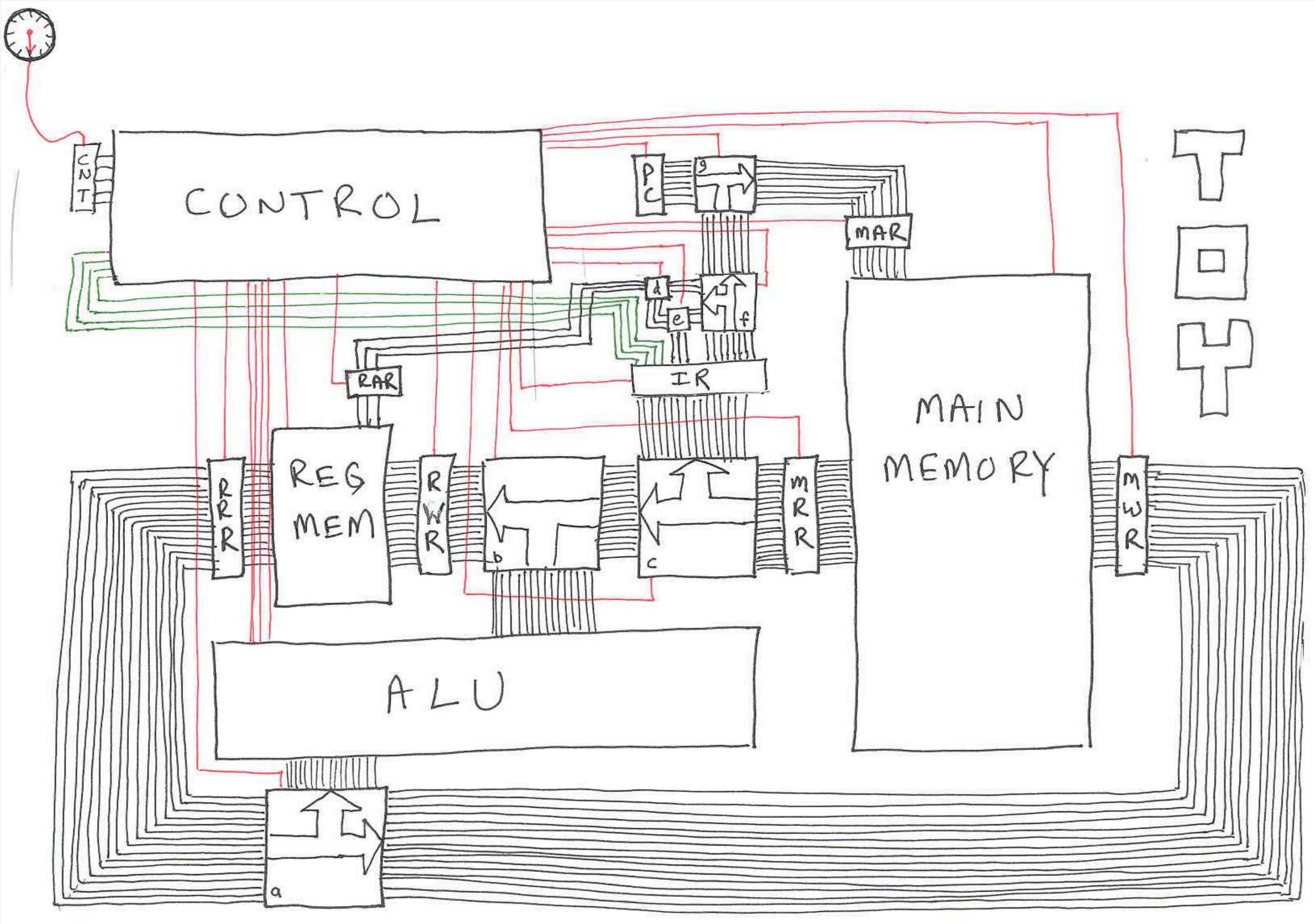


Faster clock? Faster computer!

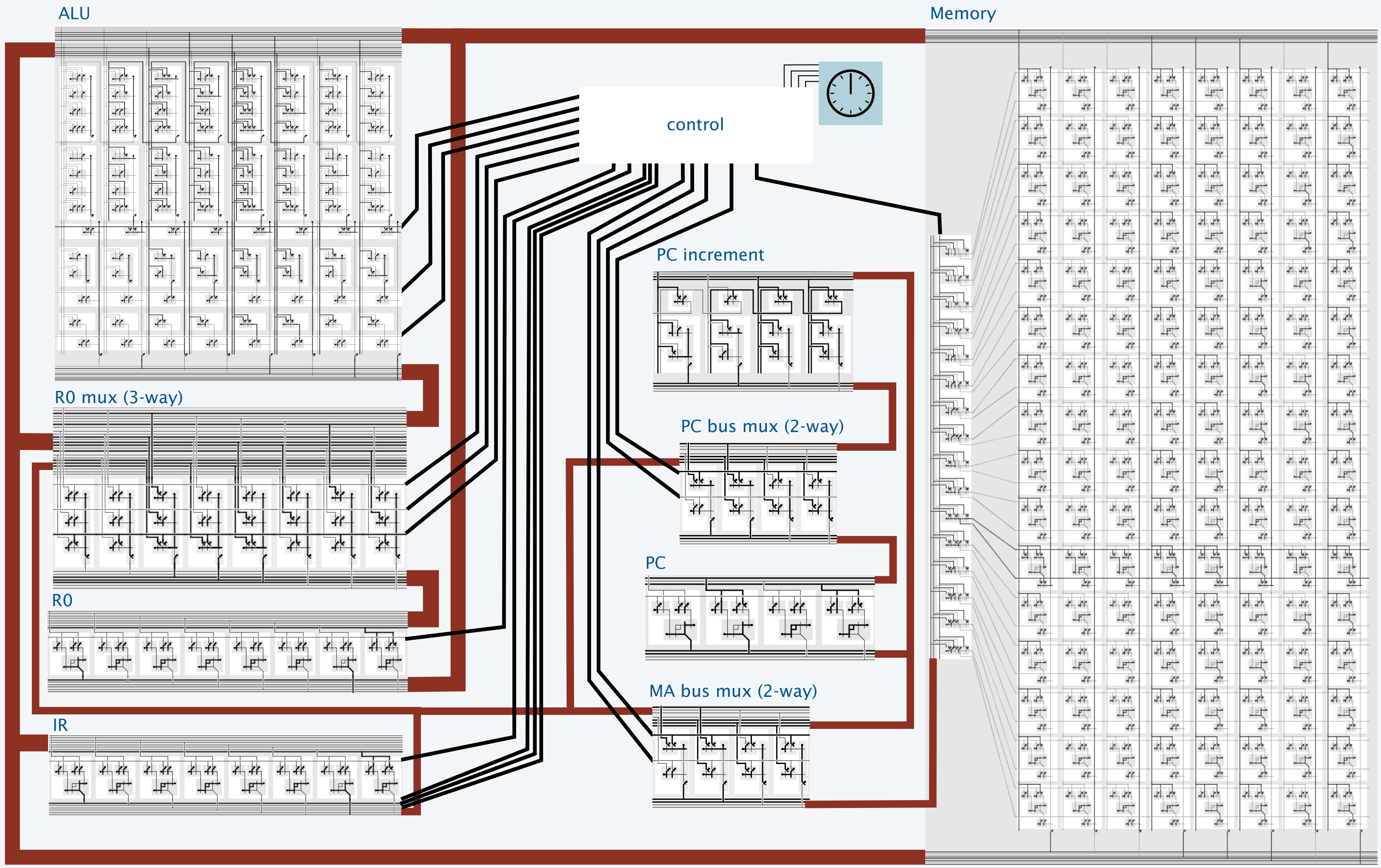


And THAT . . . is how your computer works!

TOY "Classic", back-of-envelope design (circa 2005)

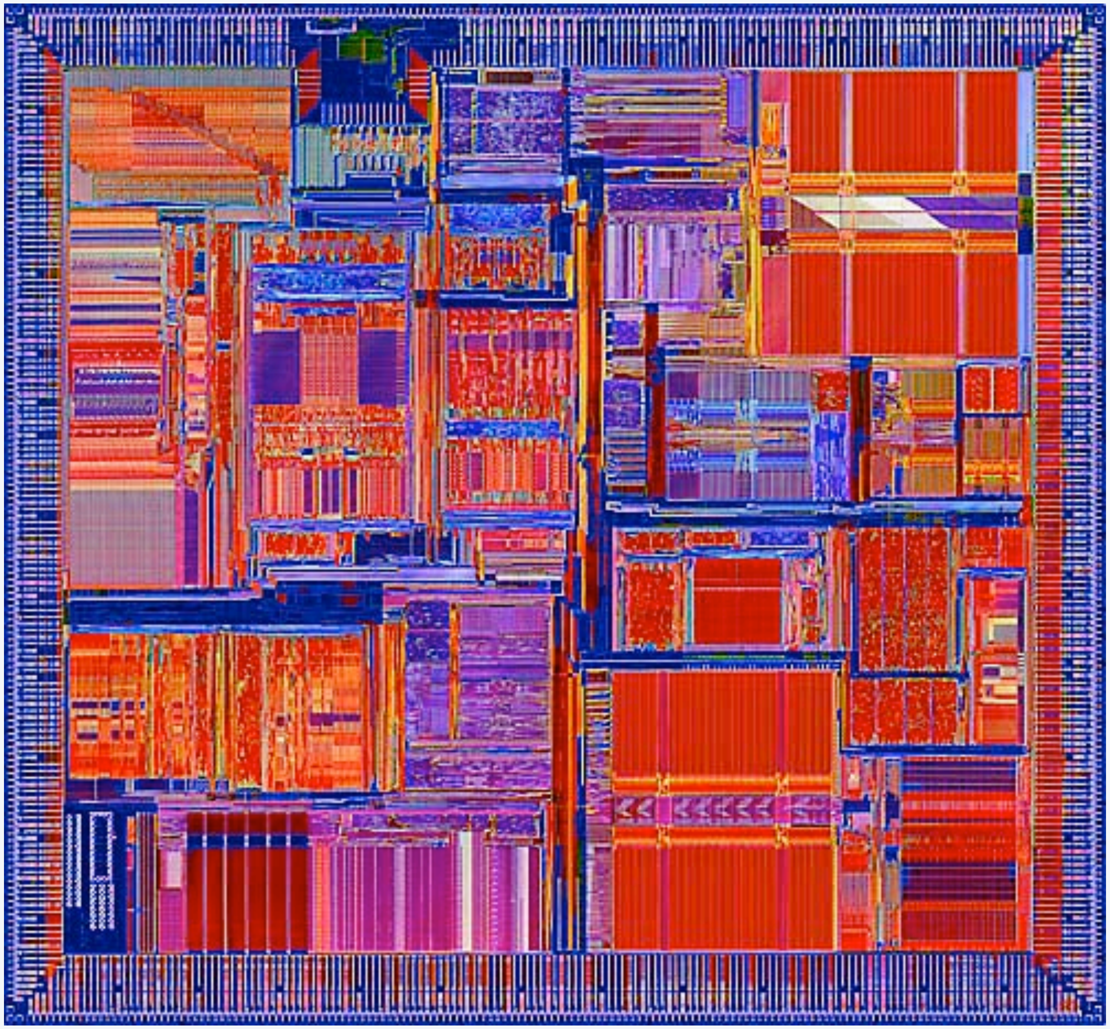


TinyTOY CPU

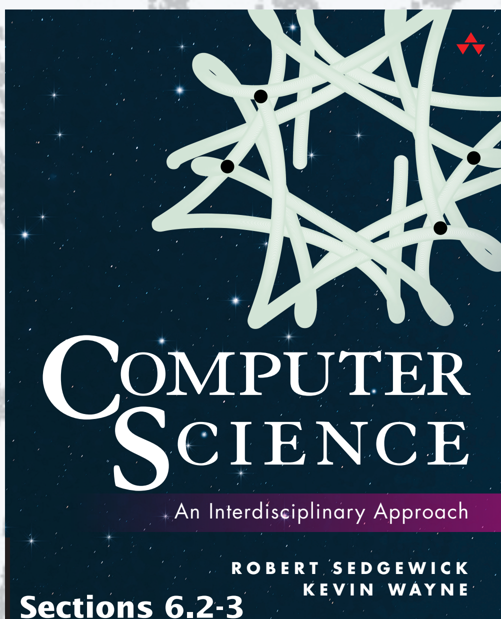


ALU
Memory
control
PC increment
R0 mux (3-way)
PC bus mux (2-way)
PC
R0
MA bus mux (2-way)
IR

A real microprocessor (MIPS R10000)



COMPUTER SCIENCE
SEDGEWICK / WAYNE



http://introcs.cs.princeton.edu

21. Central Processing Unit