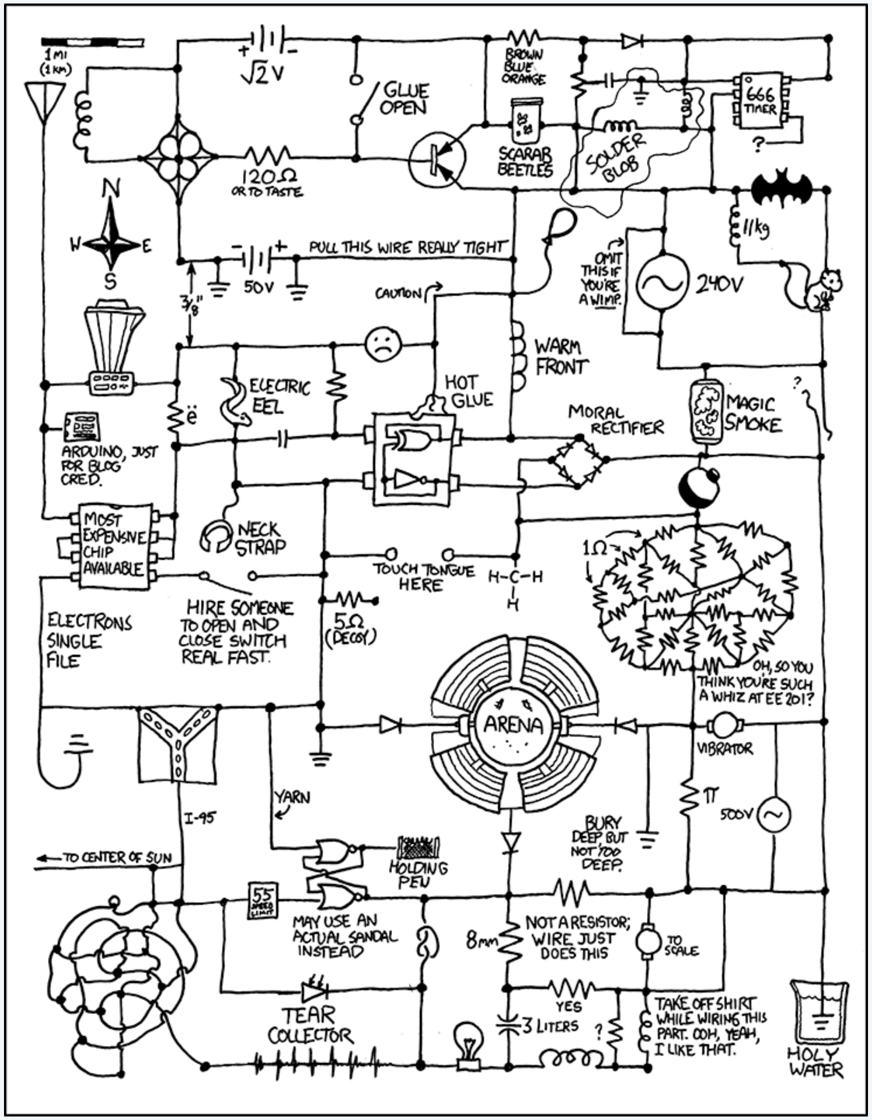
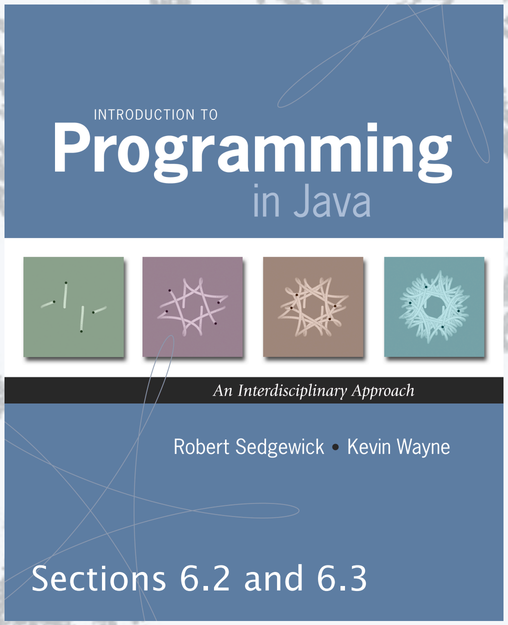

1 MI (1 KM)
+ √2 V −
GLUE OPEN
BROWN BLUE ORANGE
666 TIMER
?
120Ω OR TO TASTE
SCARAB BEETLES
SOLDER BLOB
N
W
E
S
PULL THIS WIRE REALLY TIGHT
50V
3/8"
CAUTION
OMIT THIS IF YOU'RE A WIMP.
240V
1kg
WARM FRONT
ë
ELECTRIC EEL
HOT GLUE
MORAL RECTIFIER
MAGIC SMOKE
?
ARDUINO, JUST FOR BLOG CRED.
MOST EXPENSIVE CHIP AVAILABLE
NECK STRAP
TOUCH TONGUE HERE
H–C–H
H
1Ω
ELECTRONS SINGLE FILE
HIRE SOMEONE TO OPEN AND CLOSE SWITCH REAL FAST.
5Ω (DECOY)
ARENA
OH, SO YOU THINK YOU'RE SUCH A WHIZ AT EE 201?
VIBRATOR
π
500V
YARN
I-95
BURY DEEP BUT NOT TOO DEEP.
← TO CENTER OF SUN
HOLDING PEN
55 SPEED LIMIT
MAY USE AN ACTUAL SANDAL INSTEAD
8mm
NOT A RESISTOR; WIRE JUST DOES THIS
TO SCALE
TEAR COLLECTOR
YES
3 LITERS
?
TAKE OFF SHIRT WHILE WIRING THIS PART. OOH, YEAH, I LIKE THAT.
HOLY WATER

http://xkcd.com/730/

COMPUTER SCIENCE

SEDGEWICK / WAYNE



INTRODUCTION TO

Programming

in Java

An Interdisciplinary Approach

Robert Sedgewick • Kevin Wayne

Sections 6.2 and 6.3

http://introcs.cs.princeton.edu

21. Central Processing Unit

Let's build a computer!

CPU = Central Processing Unit

Computer

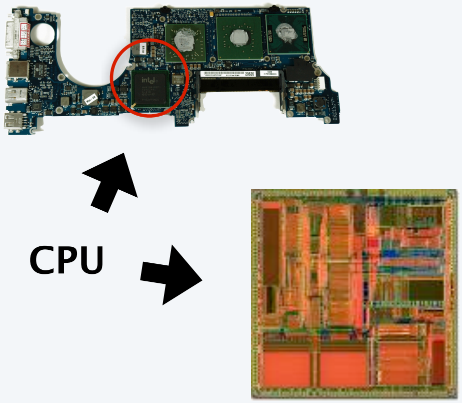
- Display
- Touchpad
- Battery
- Keyboard
- ...
- CPU (difference between a TV set and a computer)

Previous lecture

- Combinational circuits
- ALU (calculator)

This lecture

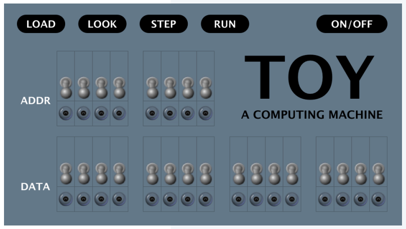
- Sequential circuits with *memory*
- CPU (computer)



A smaller computing machine: TinyTOY

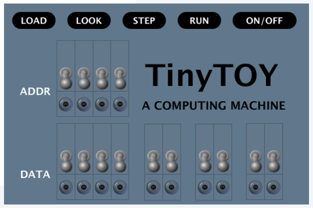
TOY instruction-set architecture.

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

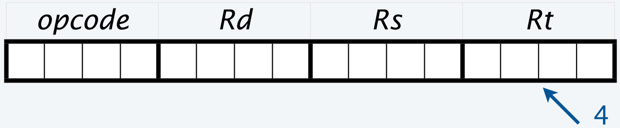


TinyTOY instruction-set architecture.

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

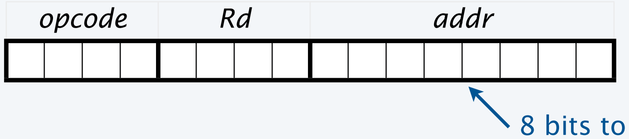


Type 1 instruction



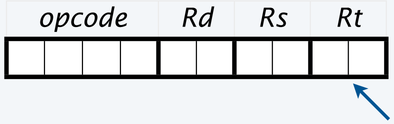
4 bits to specify one of 16 registers

Type 2 instruction



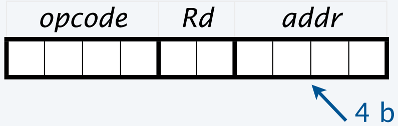
8 bits to specify one of 256 memory words

Type 1 instruction



2 bits to specify one of 4 registers

Type 2 instruction



4 bits to specify one of 16 memory words

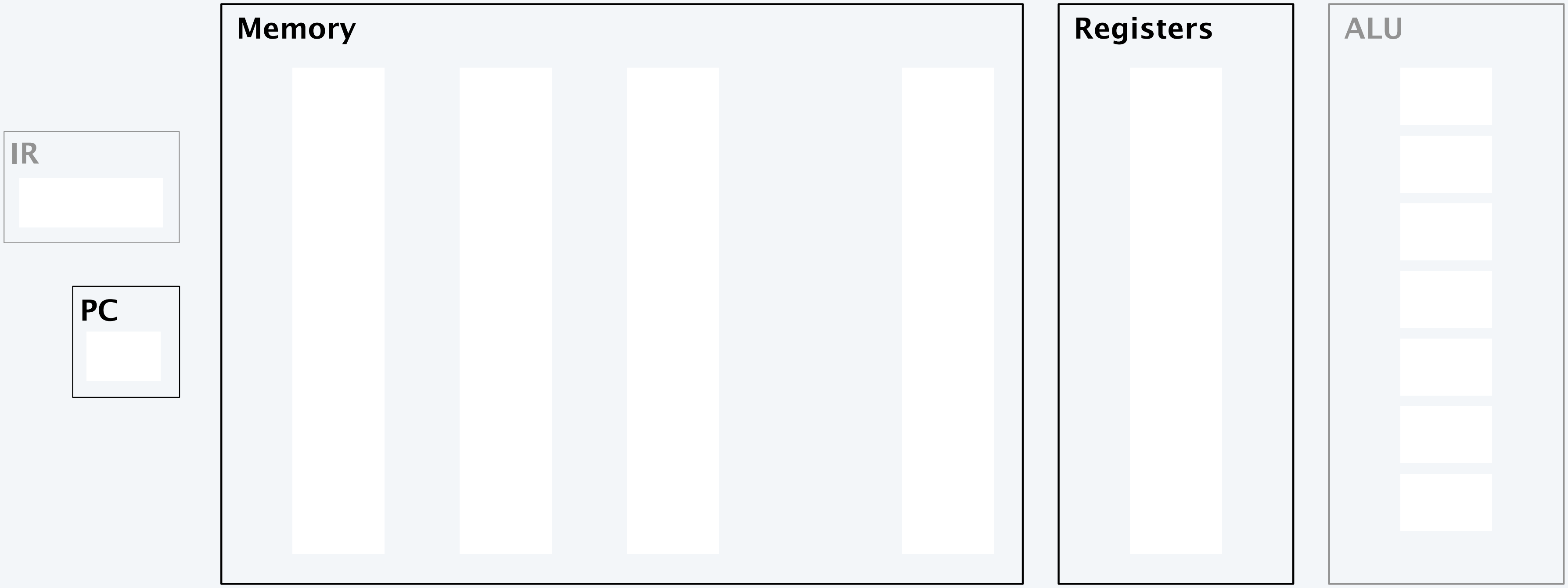
Purpose of TinyTOY. Illustrate CPU circuit design for a "typical" computer.

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.

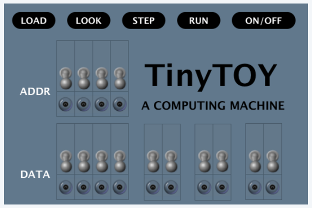
ALU and IR hold intermediate states of computation

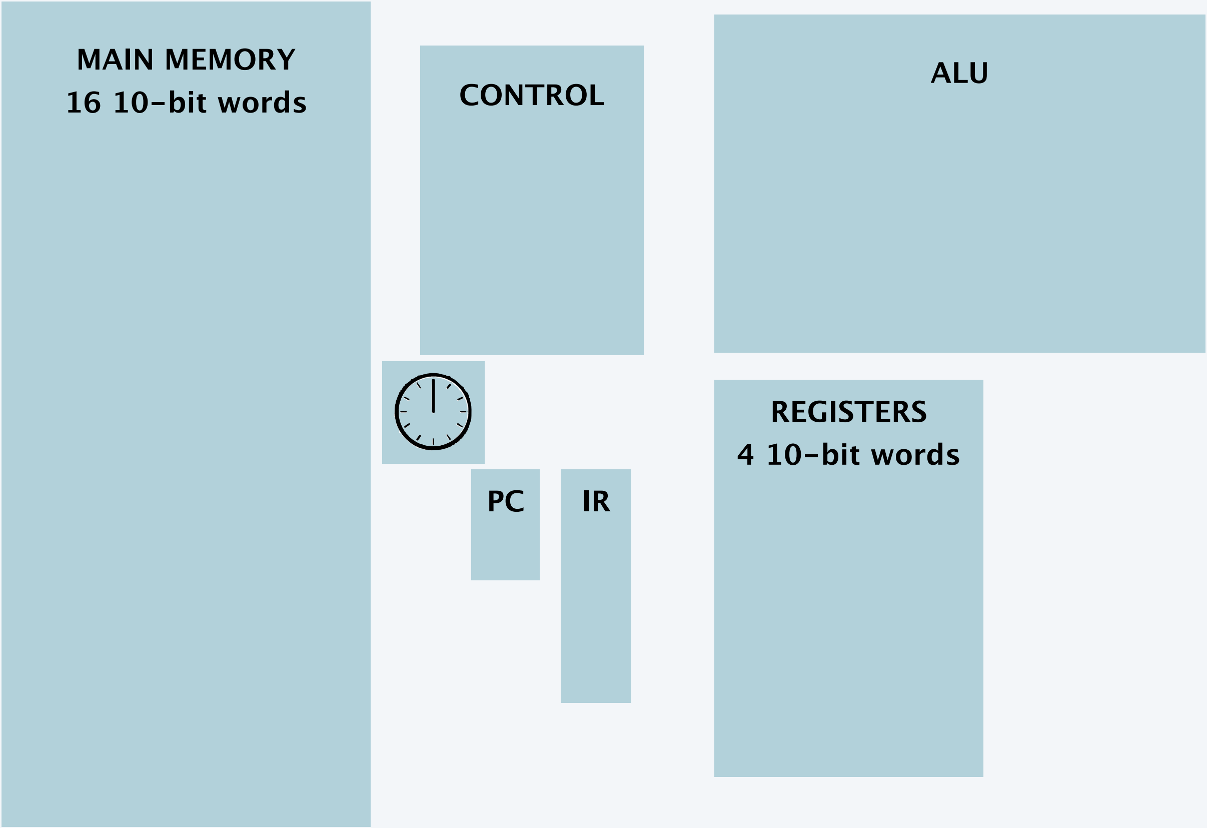


CPU circuit components for TinyTOY

TinyTOY CPU

- ALU
- Memory
- Registers
- PC
- Control
- Clock





MAIN MEMORY
16 10-bit words

CONTROL

ALU

REGISTERS
4 10-bit words

PC

IR

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

Perspective

Q. Why TinyTOY?

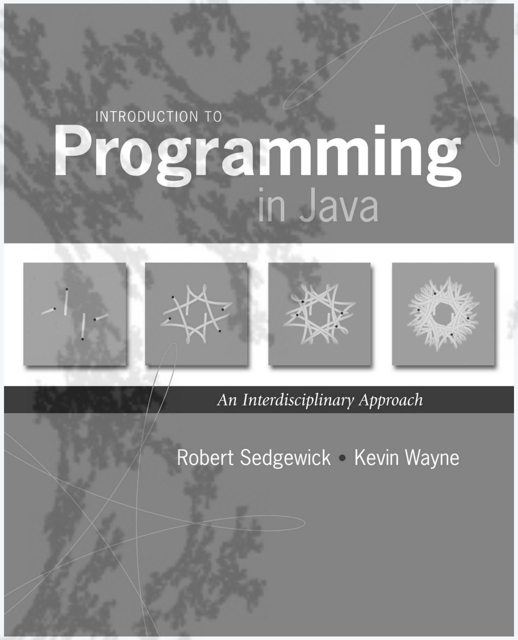
A. Toy circuit width would be about 5 times TinyTOY circuit width.

TOY
TinyTOY

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.

COMPUTER SCIENCE
SEDGEWICK / WAYNE



http://introcs.cs.princeton.edu

21. CPU

- Bits and registers
- Main memory and register banks
- Program counter
- Putting the pieces together

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).

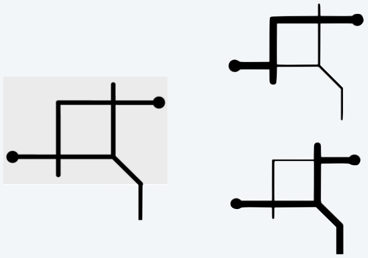
Basic abstractions

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

A new ingredient: Circuits with memory

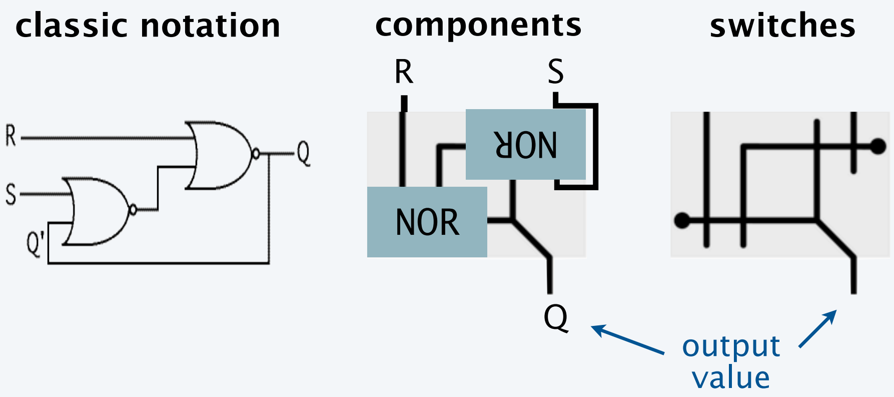
Feedback leads to circuits with one of two states

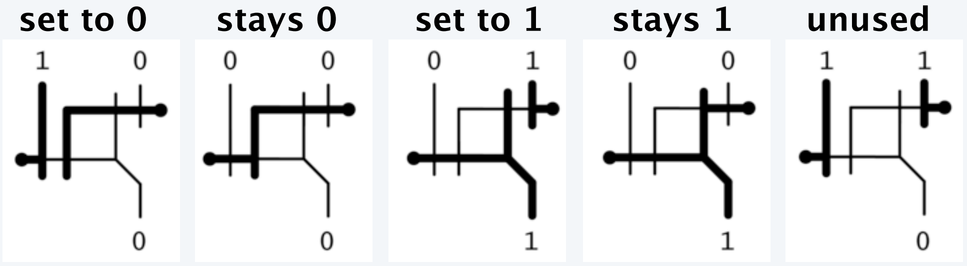
- Ex. two switches, each blocked by the other.
- State determined by whichever switches first.
- Stable (once set, state never changes).



An *SR flip-flop* is two cross-coupled NOR gates.

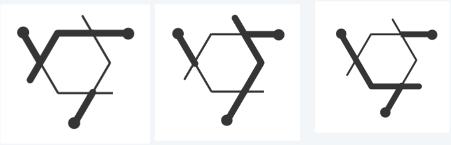
- Adds an extra line to each switch.
- R (reset) sets state to 0.
- S (set) sets state to 1.





Note. Feedback with three switches is *not* stable.

a "buzzer" →

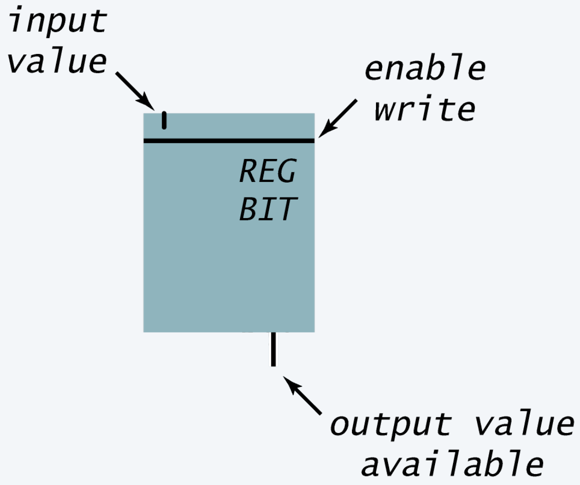


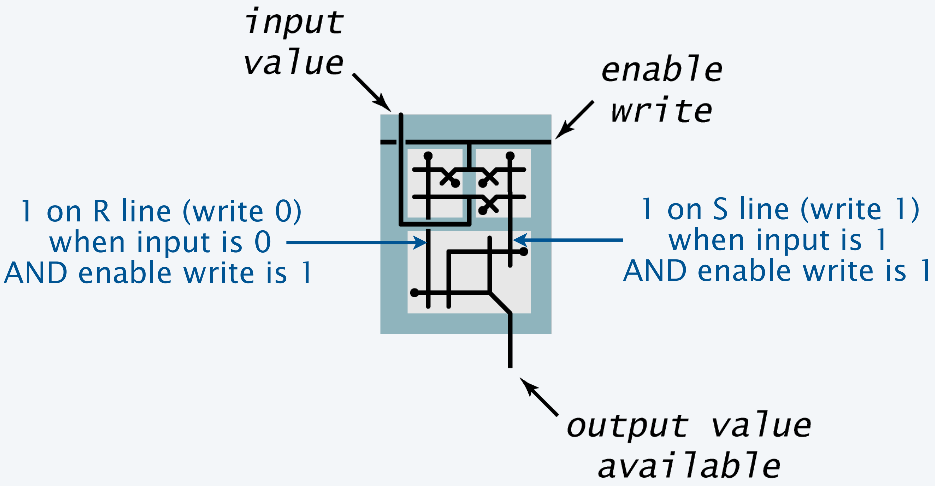
Caveat. Timing of switch vs. propagation delay.

One bit in a processor register (PC and IR)

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.




input value
enable write
1 on R line (write 0) when input is 0 AND enable write is 1
1 on S line (write 1) when input is 1 AND enable write is 1
output value available

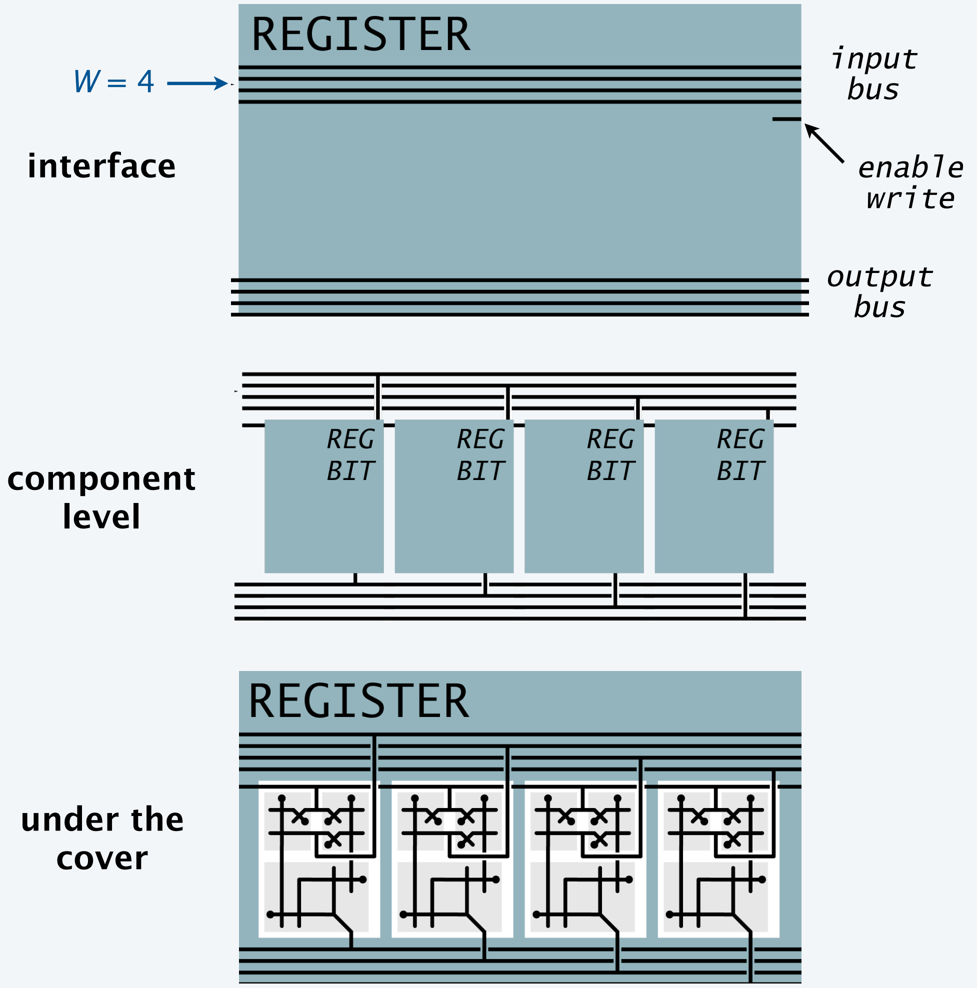
Processor registers

Processor registers (PC and IR)

- Store *W* bits.
- Input and output on *W*-wire busses.
- Register contents always available on output bus.
- When enable write is asserted, *W* input bits get copied into register.

Ex 1. PC holds 4-bit address.

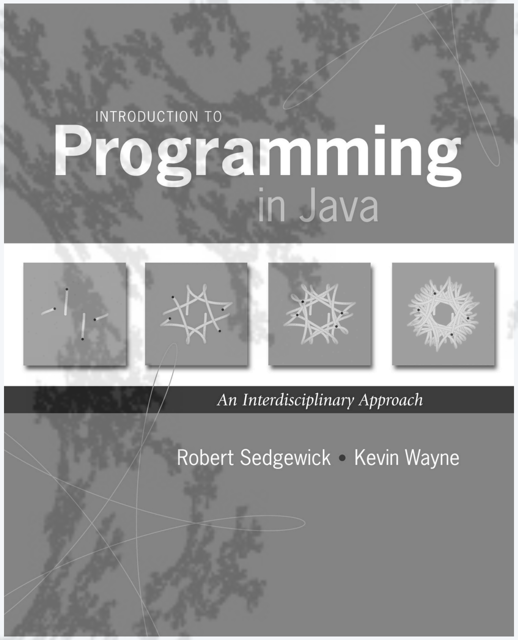
Ex 2. IR holds 10-bit current instruction.



interface

component level

under the cover

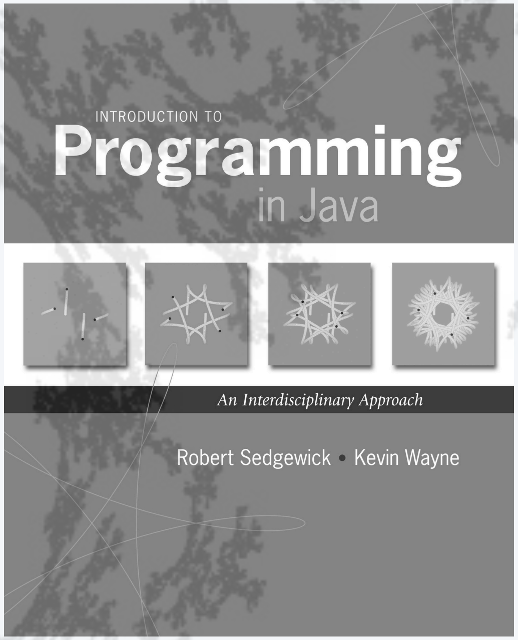


http://introcs.cs.princeton.edu

21. CPU

- Bits and registers
- Main memory and register banks
- Program counter
- Putting the pieces together

COMPUTER SCIENCE
SEDGEWICK / WAYNE



INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick • Kevin Wayne

http://introcs.cs.princeton.edu

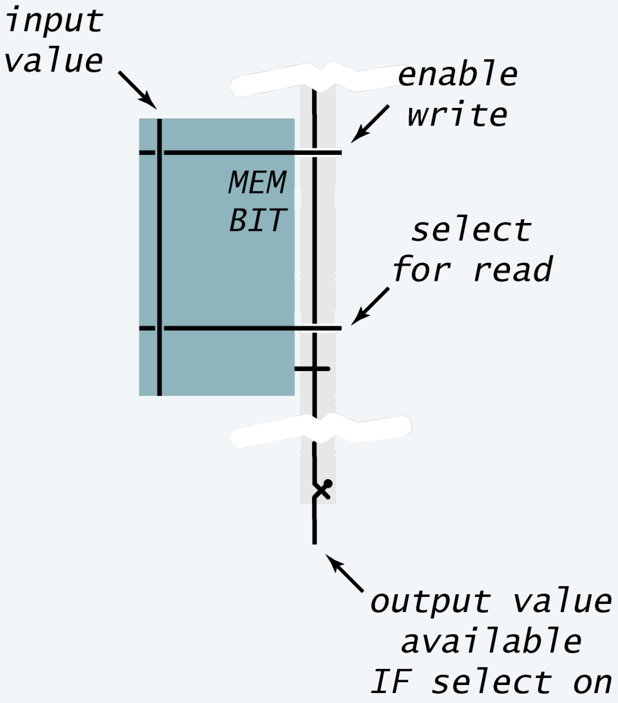
21. CPU

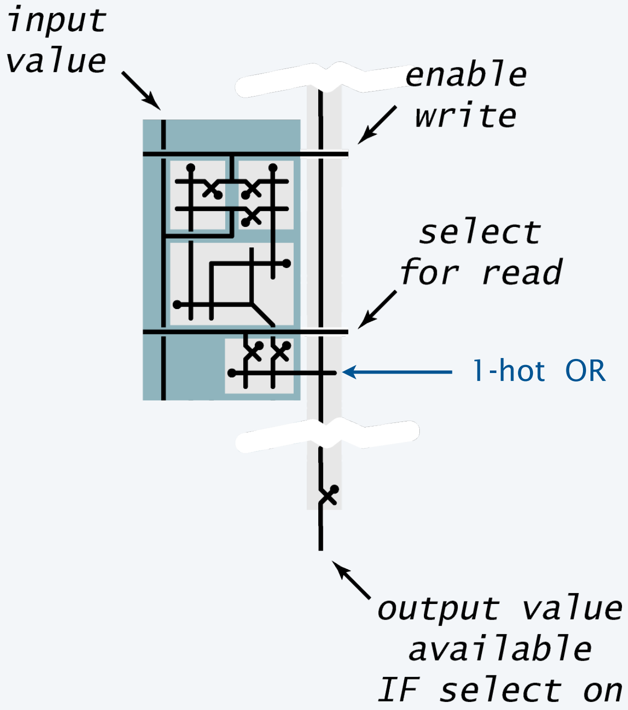
- Bits and registers
- **Main memory and register banks**
- Program counter
- Putting the pieces together

One bit in a main memory bank

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.





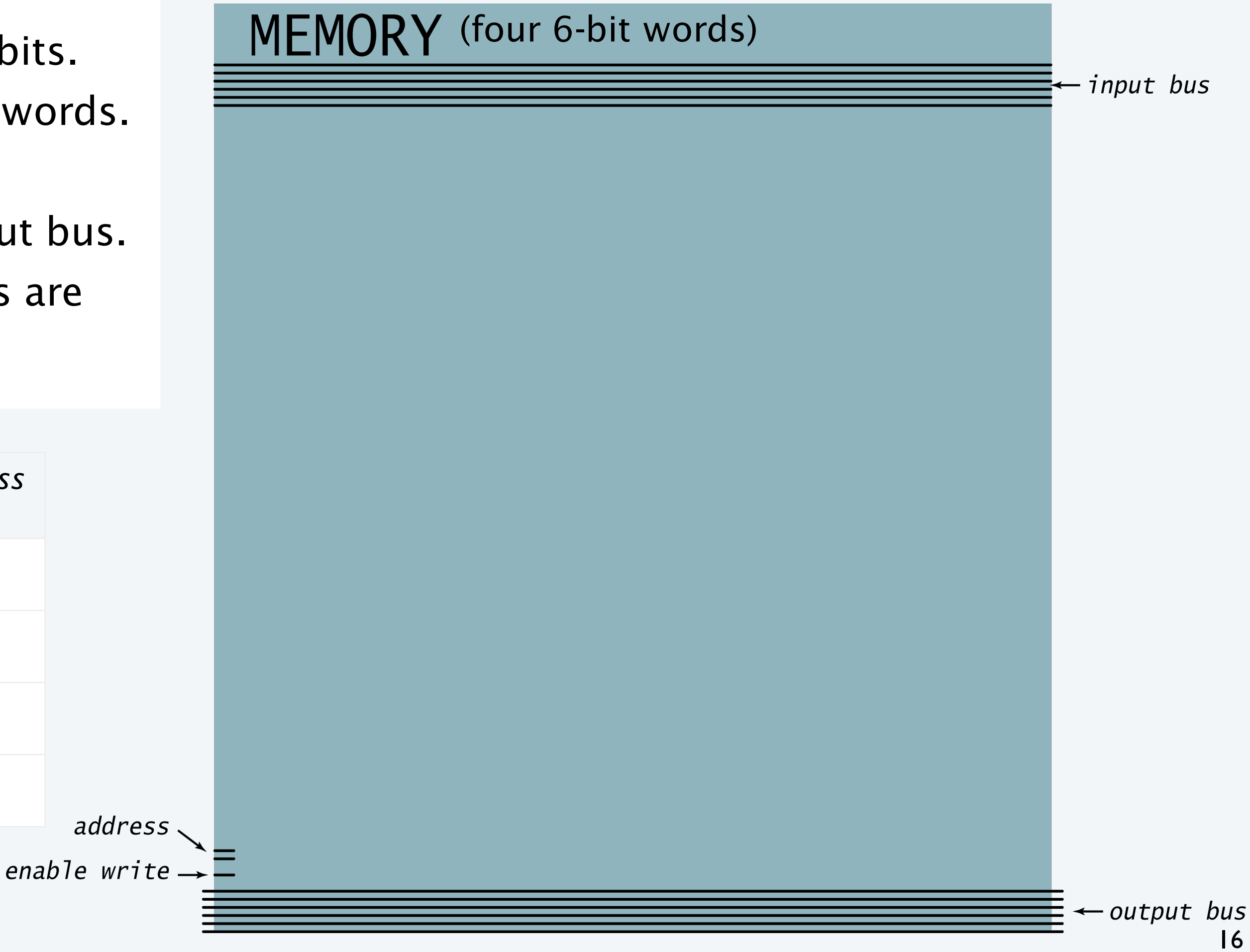
Main memory bank: interface

Main memory bank.

- Bank of *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	10	4
TOY	256	16	8
your computer	1 billion	64	32

interface



Main memory bank: component level

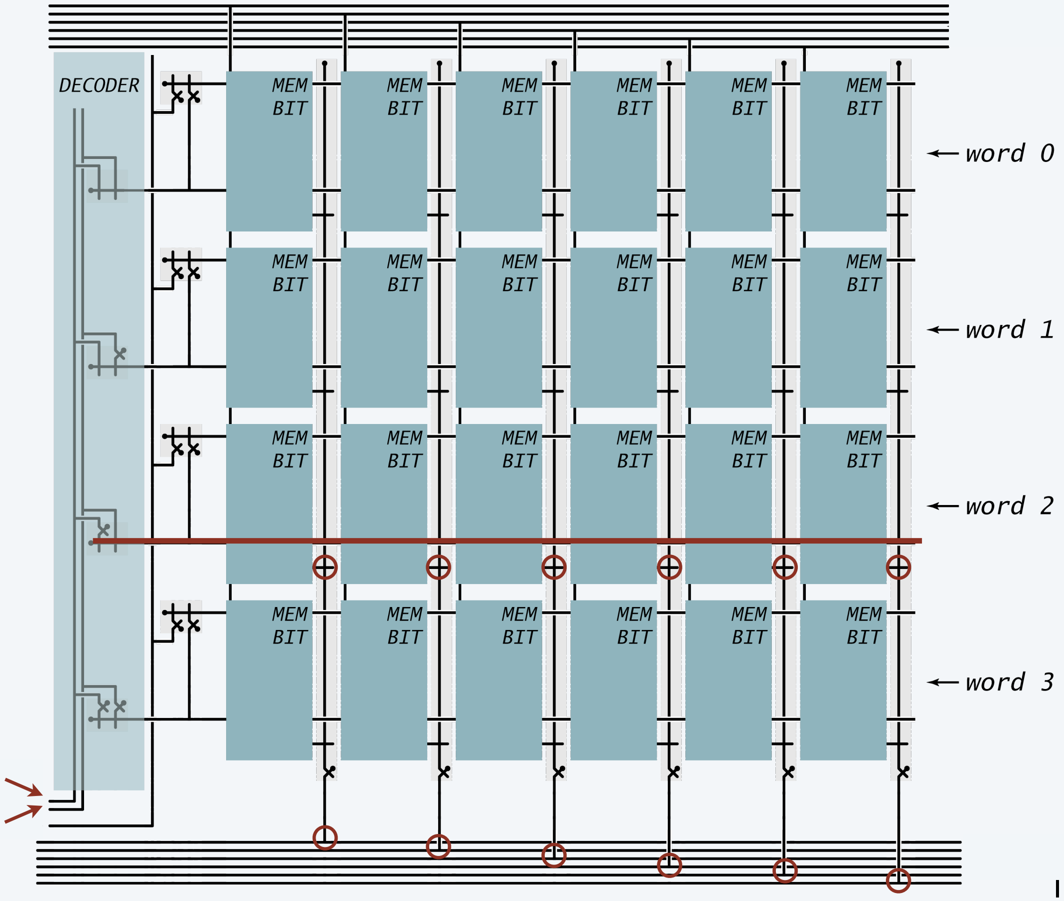
Main memory bank.

- Bank of *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)



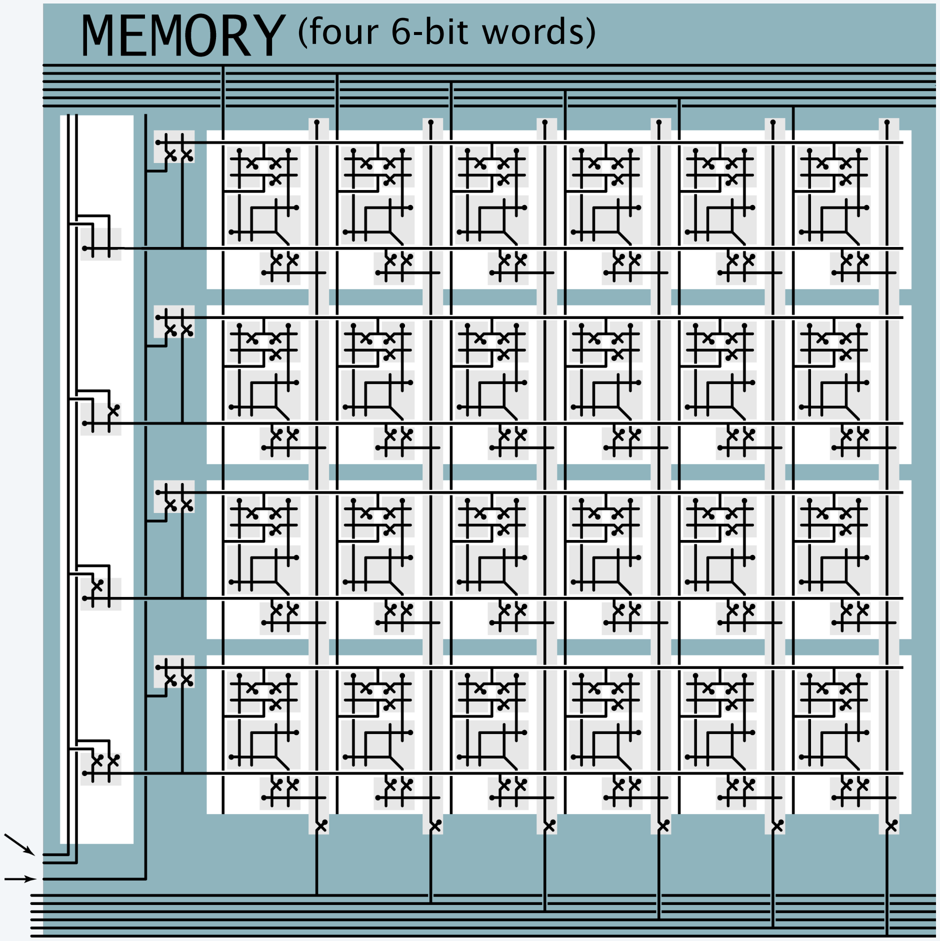
Example: Read word 2 (10)

Main memory bank: switch level

Main memory bank.

- Bank of *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



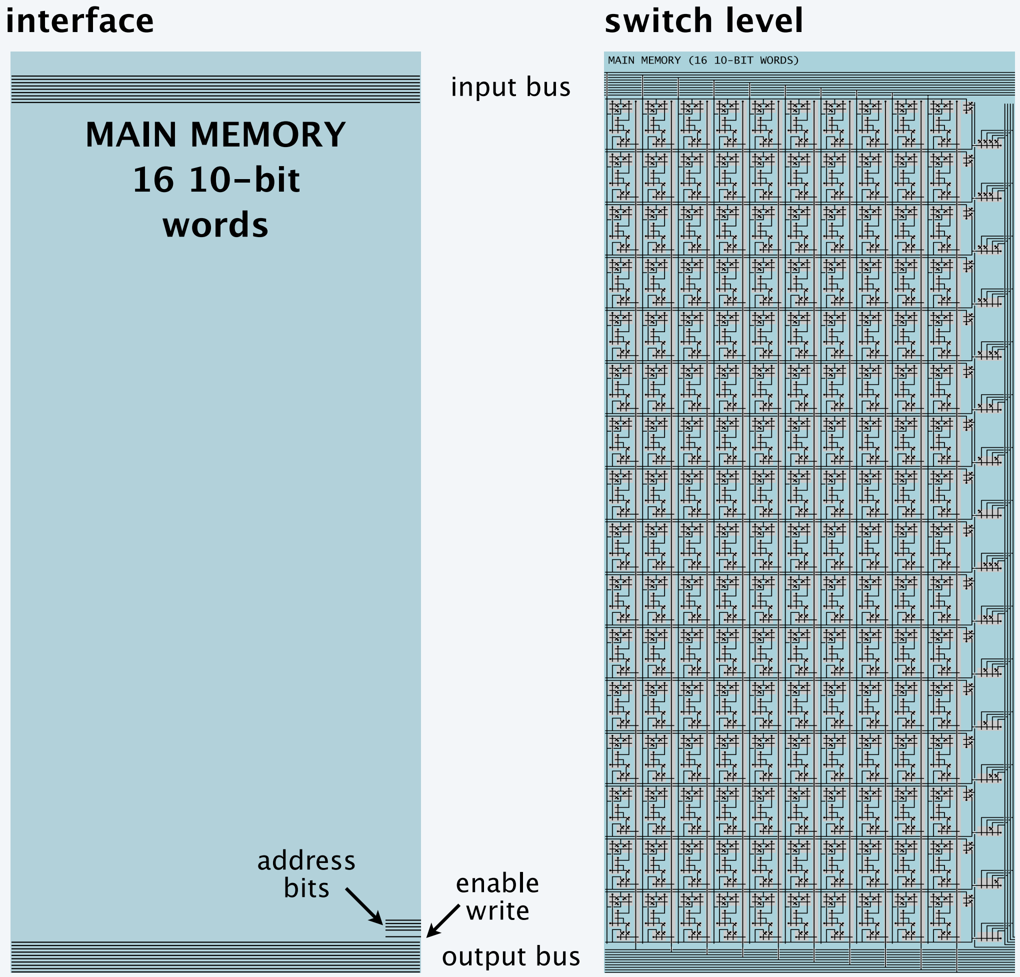
TinyTOY main memory bank

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 registers
- Address bits from PC, IR, registers
- *Enable write* from "control"



interface

switch level

MAIN MEMORY
16 10-bit
words

input bus

address bits

enable write

output bus

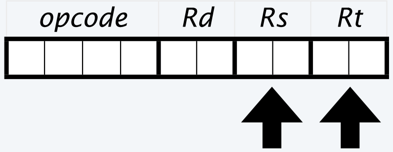
MAIN MEMORY (16 10-BIT WORDS)

One bit in a TinyTOY register bank

Need a second selection mechanism to read two registers at once

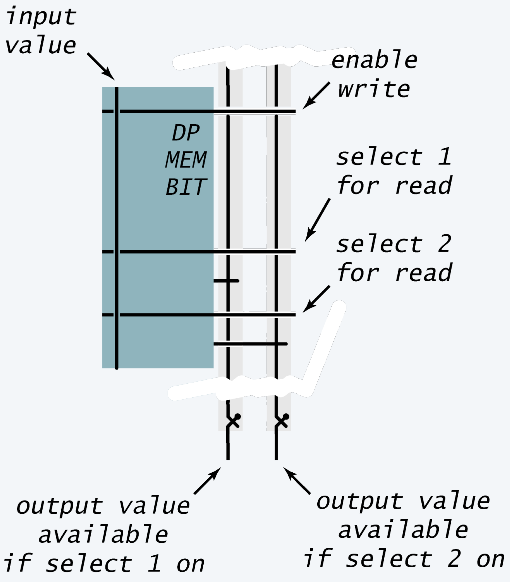
- Flip-flop value is *not* always available.
- Use *select 1 for read* signal to make it available on output line 1.
- Use *select 2 for read* signal to make it available on output line 2.
- "1-hot" OR to collect the selected bit values on each line.

Type 1 instruction

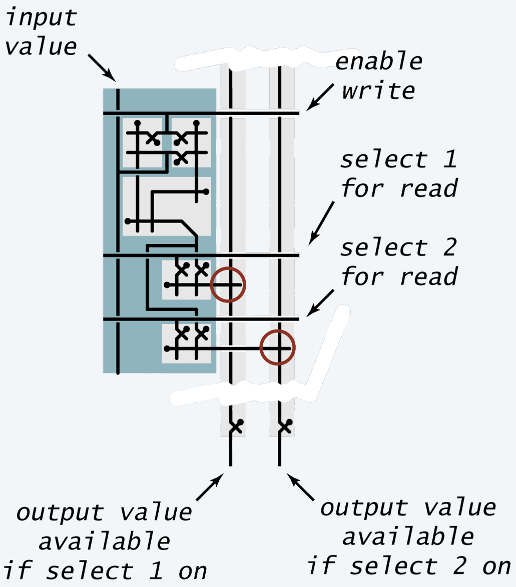


"Dual port" memory bit

interface


input value
enable write
DP MEM BIT
select 1 for read
select 2 for read
output value available if select 1 on
output value available if select 2 on

switch level


input value
enable write
select 1 for read
select 2 for read
output value available if select 1 on
output value available if select 2 on

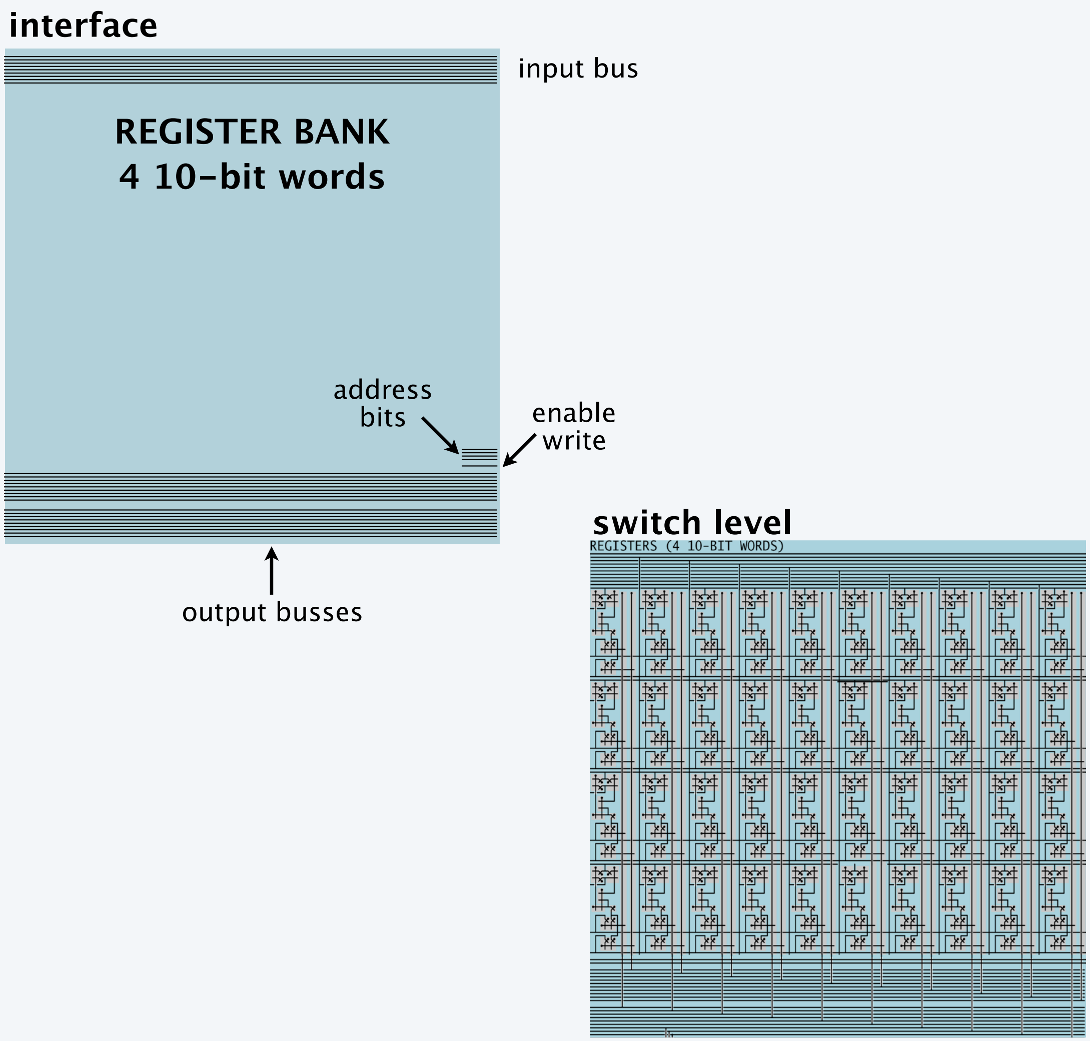
TinyTOY register bank

Interface

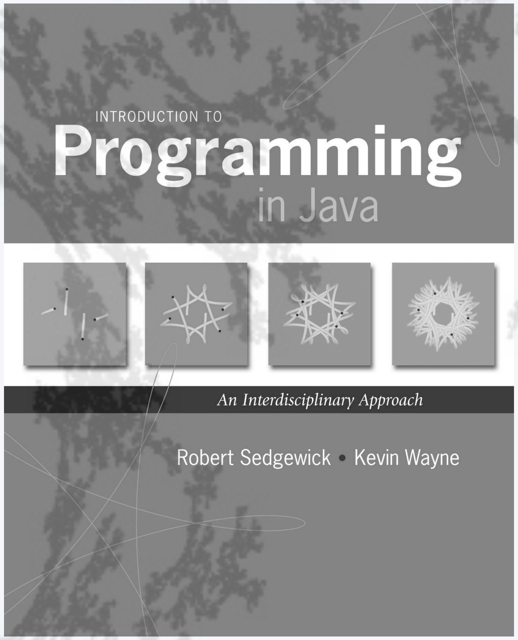
- Input bus
- Two output busses
- Address bits to select word
- Address bits to select 2nd word
- *Enable write* control signal

Connections

- Input bus from MUX (stay tuned)
- Output busses to ALU, memory
- Address bits from IR
- *Enable write* from "control"



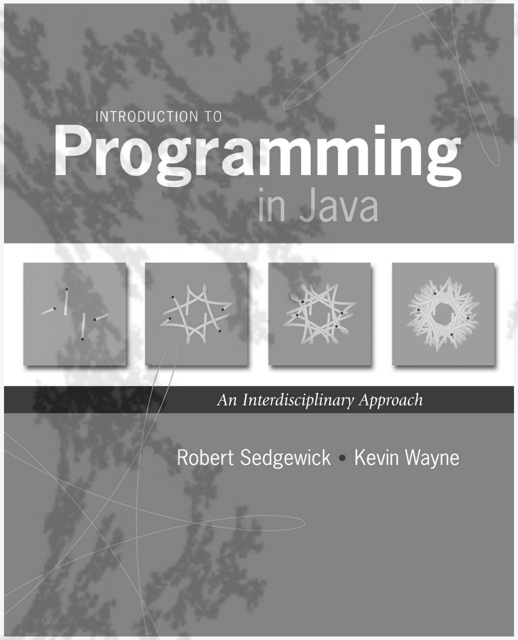
COMPUTER SCIENCE
SEDGEWICK / WAYNE



http://introcs.cs.princeton.edu

21. CPU

- Bits and registers
- **Main memory and register banks**
- Program counter
- Putting the pieces together



http://introcs.cs.princeton.edu

21. CPU

- Bits and registers
- Main memory and register banks
- **Program counter**
- Putting the pieces together

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.

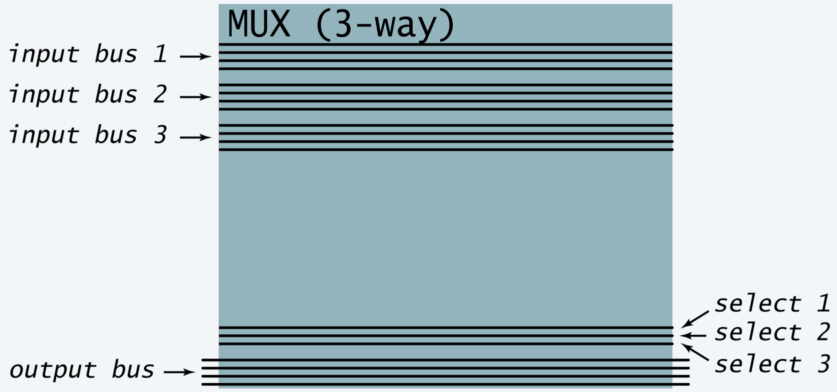
Warmup. Design tinyTOY program counter (PC). ← Three components and three control signals

Another useful combinational circuit: Multiplexer

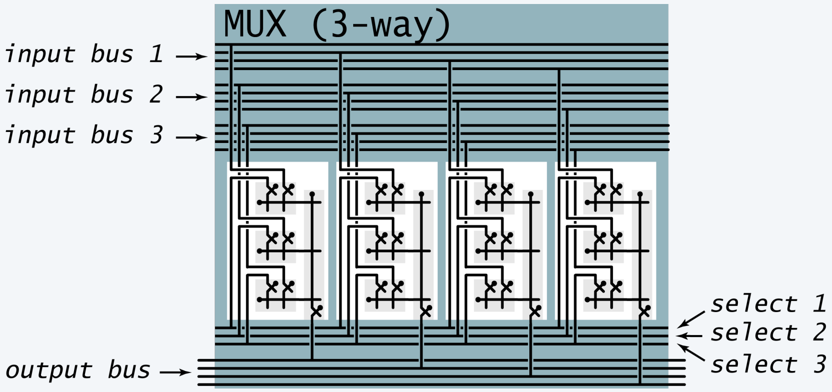
Multiplexer (MUX). Combinational circuit that selects among input buses.

- Exactly one select line *i* is activated. ← unary input (MUX in text takes binary input)
- Puts bit values from input bus *i* onto output bus.

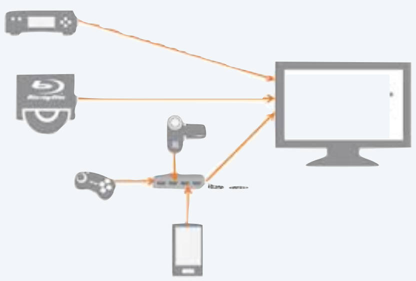
interface



switch level



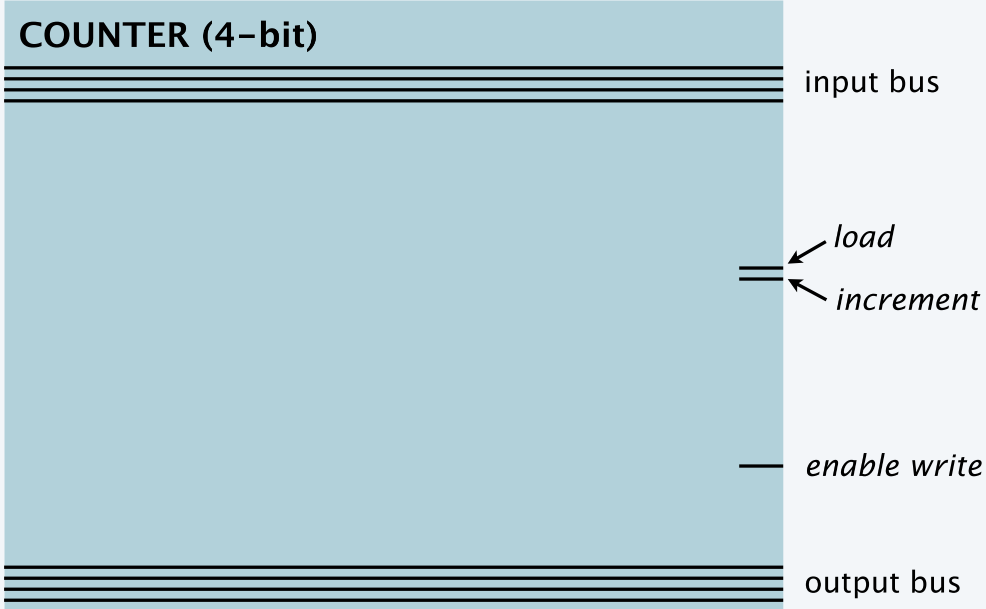
Typical use. Connect a component in different ways at different times.



Counter interface

A *counter* holds a value that represents a binary integer and supports three control signals:

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



Components inside

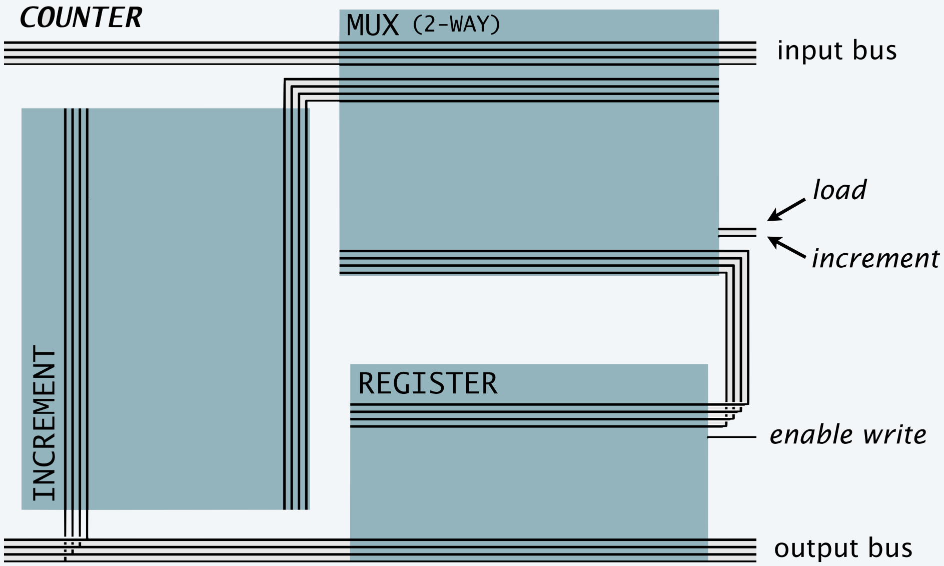
- Input and output busses.
- Processor register.
- Incrementer (add 1).
- 2-way MUX.

TinyTOY PC: 4-bit counter

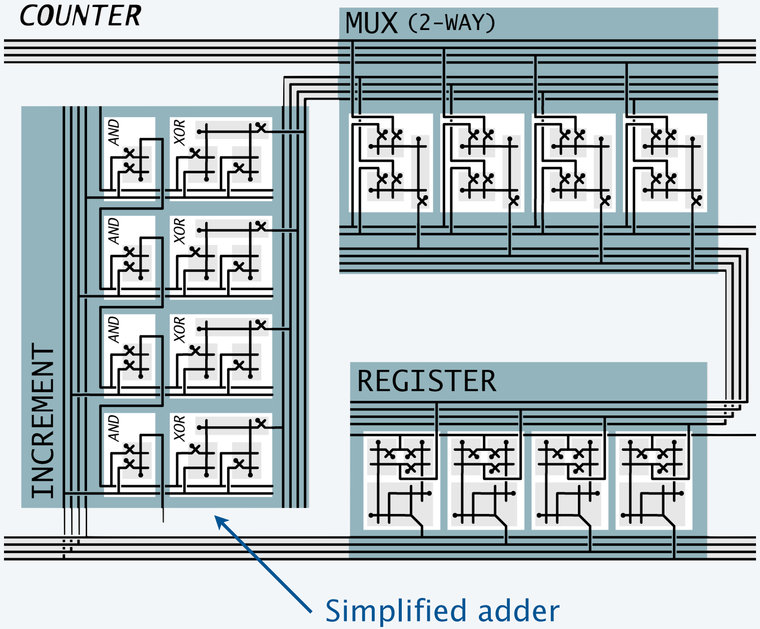
Counter layout and implementation

Layout and connections establish data paths where information travels.

component-level implementation



switch-level implementation



Simplified adder
with y = 0001

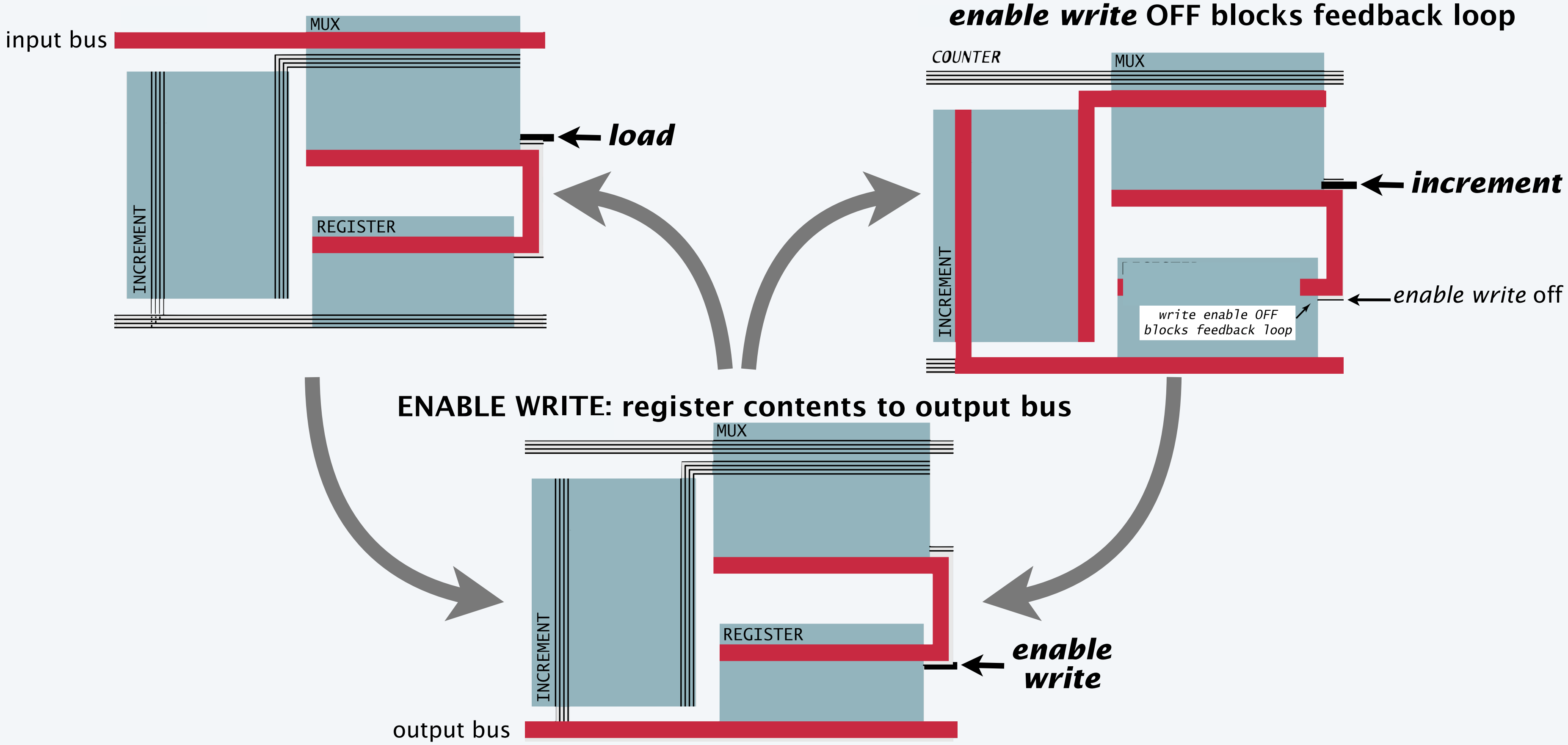
Next: Sequence of control signals that effects desired behavior

Control signal sequences and data paths for counter

LOAD: input bus to MUX output to register

INCREMENT: register output to incrementer
incrementer output to MUX
increment selects copy to register input
enable write OFF blocks feedback loop

ENABLE WRITE: register contents to output bus



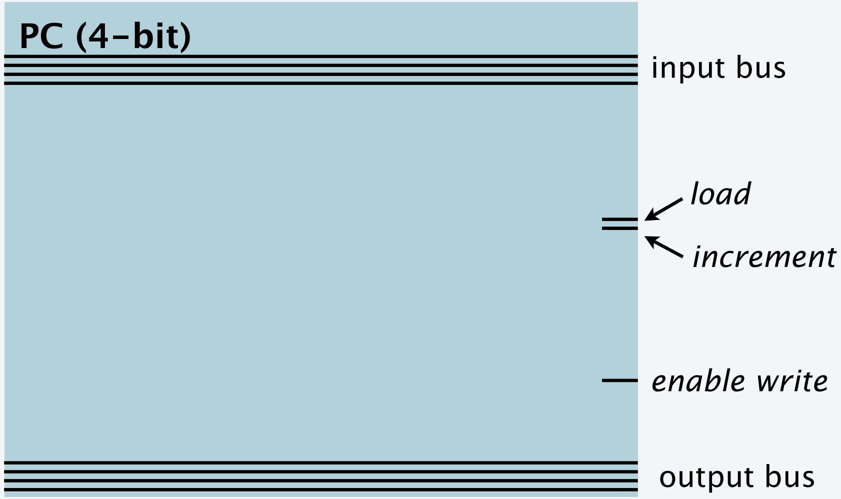
Summary of TinyTOY PC circuit

The *program counter* holds an address and supports two control signal sequences:

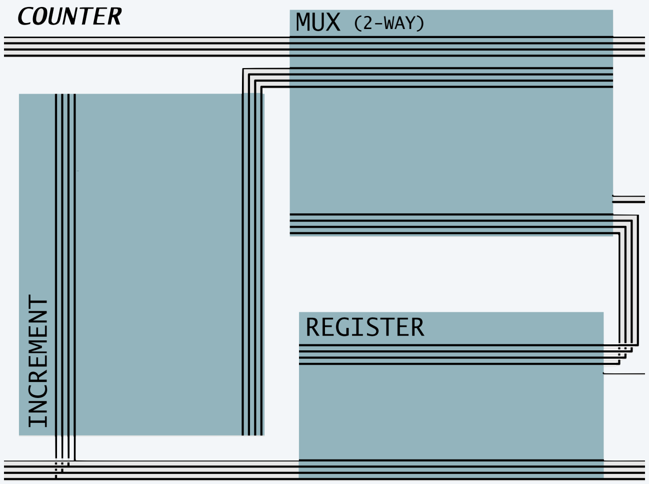
- *Load, then enable write.* Set value from input bus (example: branch instruction).
- *Increment, then enable write.* Add one to value.

Value is written to an internal processor register and available on output bus in both cases.

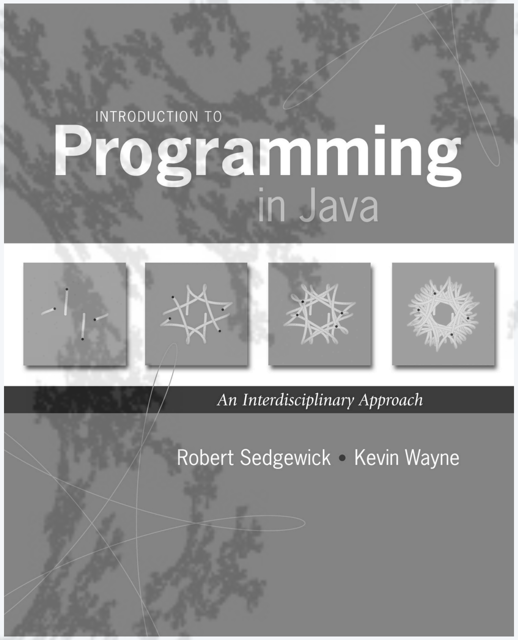
interface



component-level implementation



Next. CPU circuit (10 components, 27 control signals).

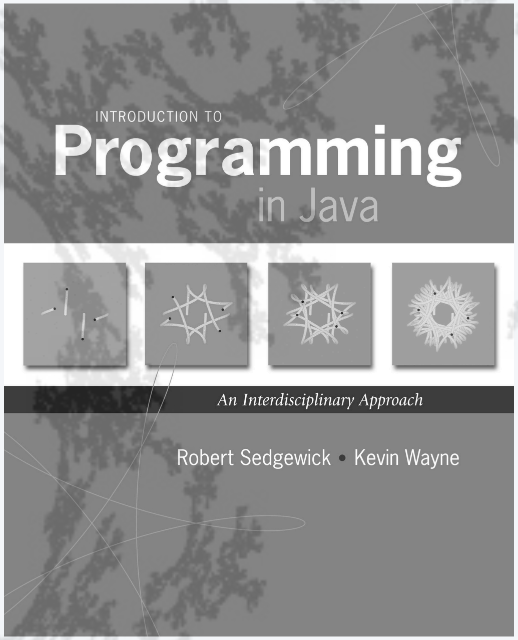


http://introcs.cs.princeton.edu

21. CPU

- Bits and registers
- Main memory and register banks
- **Program counter**
- Putting the pieces together

COMPUTER SCIENCE
SEDGEWICK / WAYNE


INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick • Kevin Wayne

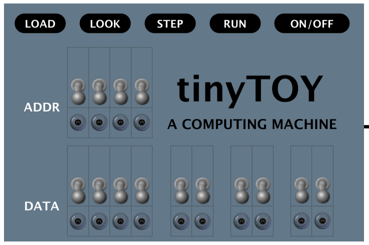
http://introcs.cs.princeton.edu

21. CPU

- Bits and registers
- Main memory and register banks
- Program counter
- **Putting the pieces together**

TinyTOY: Interface

CPU is a circuit inside the machine



TinyTOY
A computing machine

Interface to outside world

- Switches and lights
- ON/OFF
- RUN

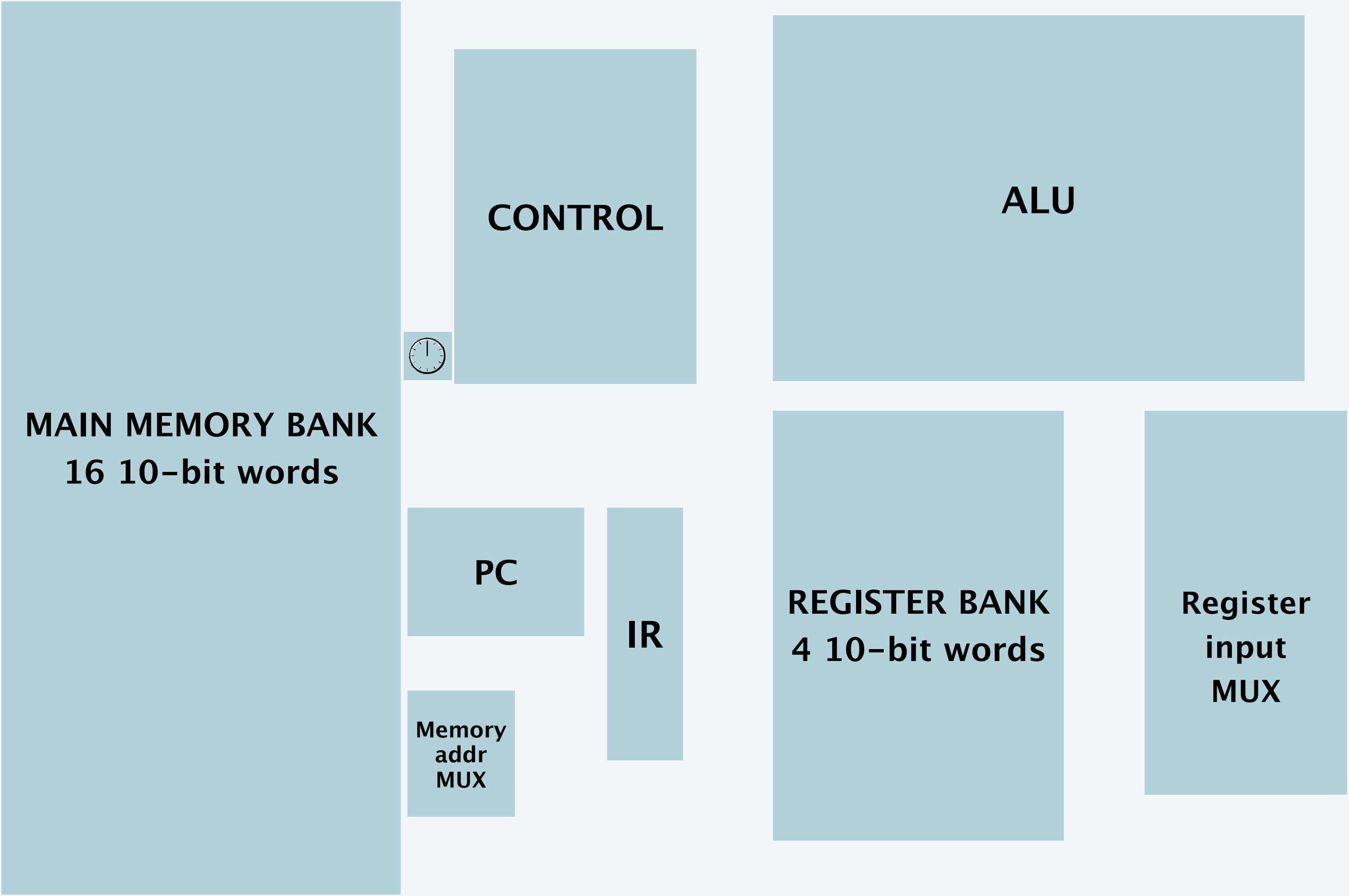
Connections (omitted)

- ADDR to PC
- DATA to memory bank input bus
- Buttons to control signals that implement memory load/store

TinyTOY CPU components and layout

Components

- ALU
- Main memory
- Registers
- PC
- IR
- MUX switches
- Control
- Clock



Review: Program counter and instruction register

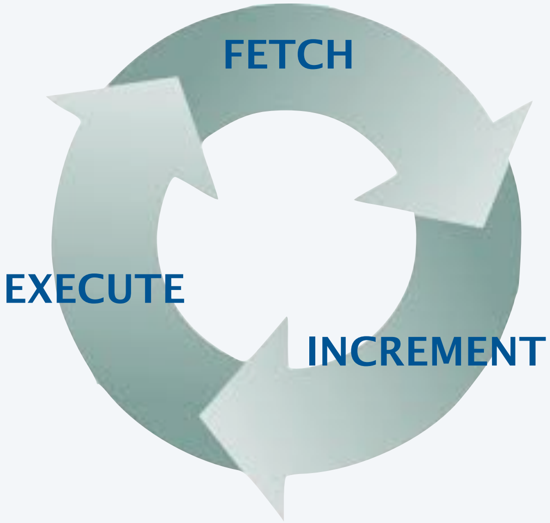
TOY operates by executing a sequence of instructions.

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 control lines and data paths

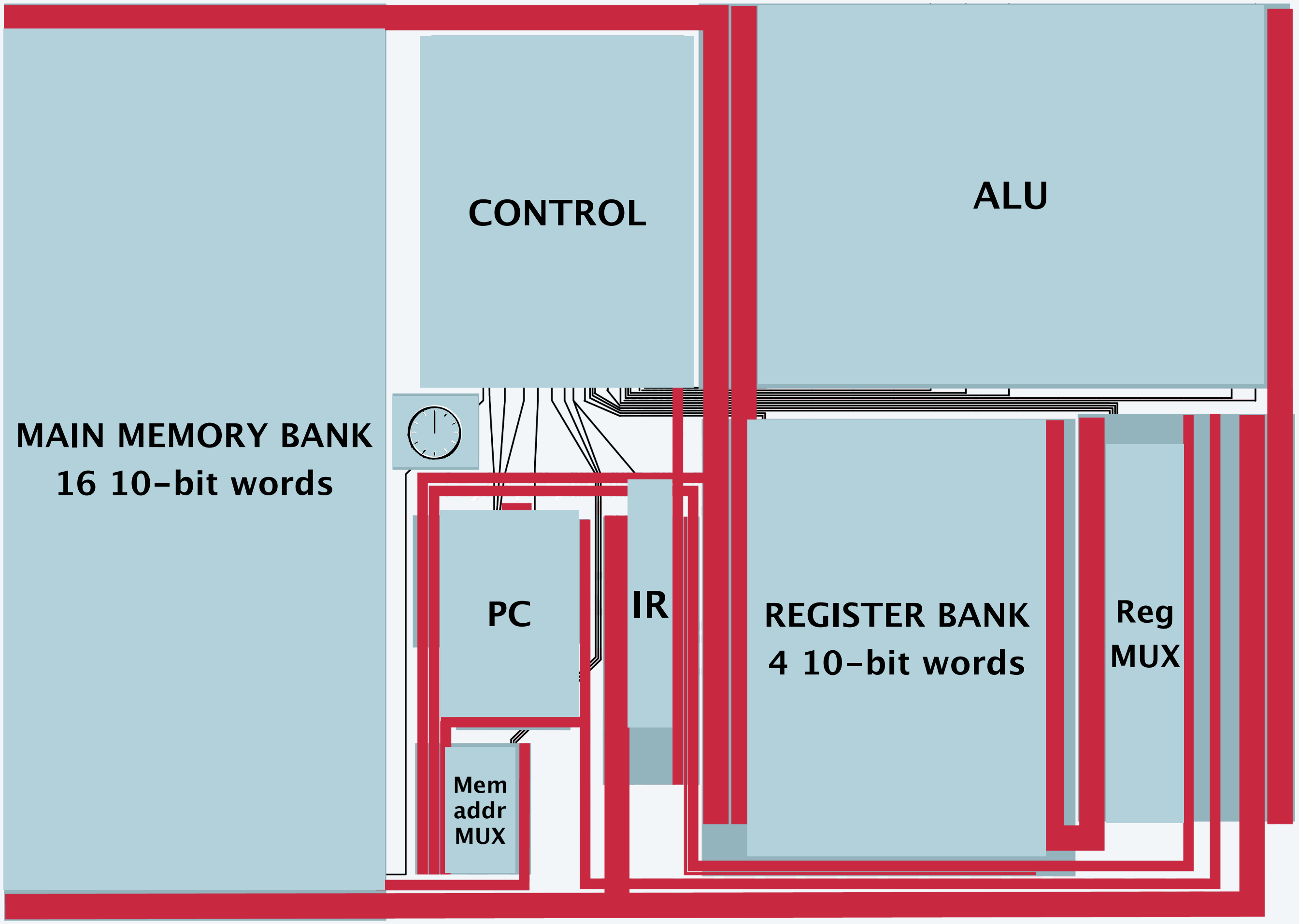
Control lines

- *PC to MA MUX*
- *IR enable write*
- *increment PC*
- *PC enable write*
- *memory enable write*
- *reg bank enable write*
- ...

[20-30 additional lines]

control lines —

data paths 🟥

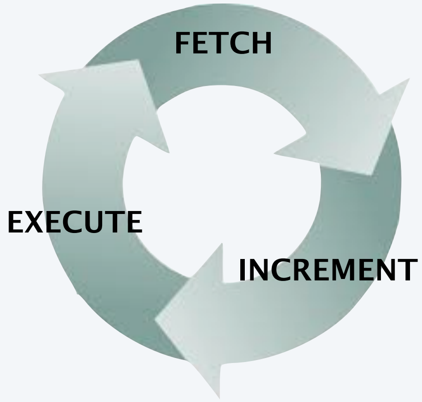


TinyTOY control sequences

Control sequences choreograph the flow of information through the CPU.

- Example 1: Fetch
- Example 2: Increment
- Example 3: ALU instruction

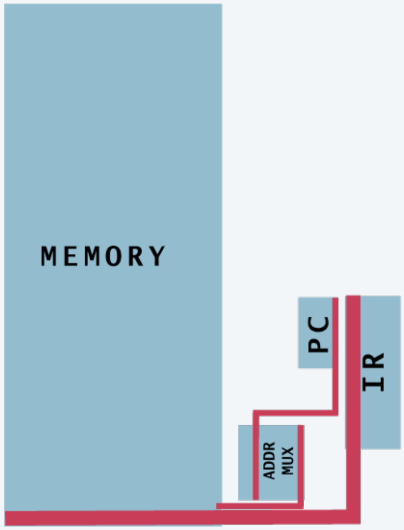
[small sequence for each instruction]



FETCH			*PC to mem addr MUX* *IR enable write*	1
INCREMENT			*PC increment* *PC enable write*	2
EXECUTE	0	halt	...	
	1	add	*[IR opcode to decoder]* *[decoder to ALU select]* *switch reg input MUX to ALU* *reg bank enable write*	3
	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	...	

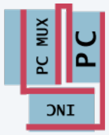
Datapath for an add instruction

FETCH



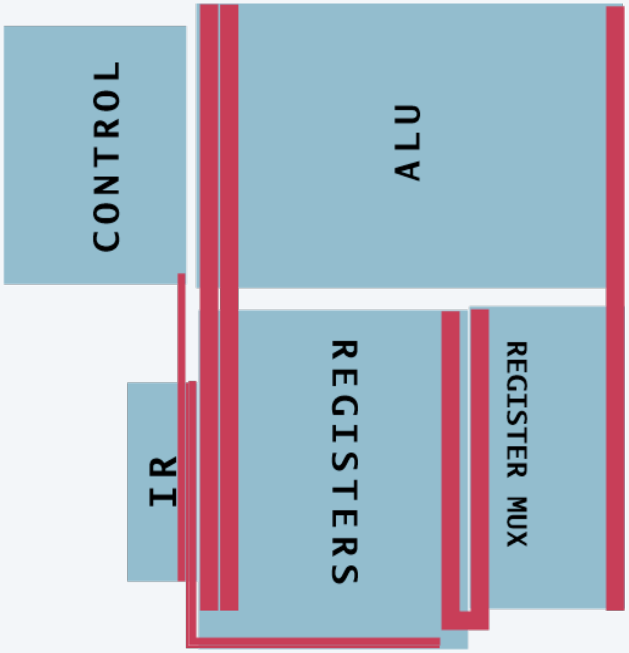
PC to mem addr MUX
IR enable write

INCREMENT



PC increment
PC enable write

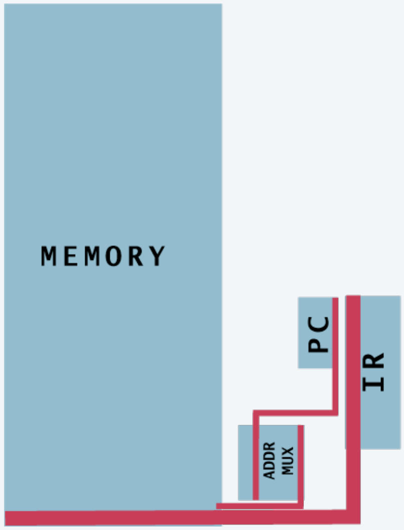
EXECUTE



[IR opcode to control/decoder]
[decoder to ALU select]
switch reg input MUX to ALU
reg bank enable write

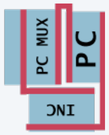
Datapath for a load instruction

FETCH



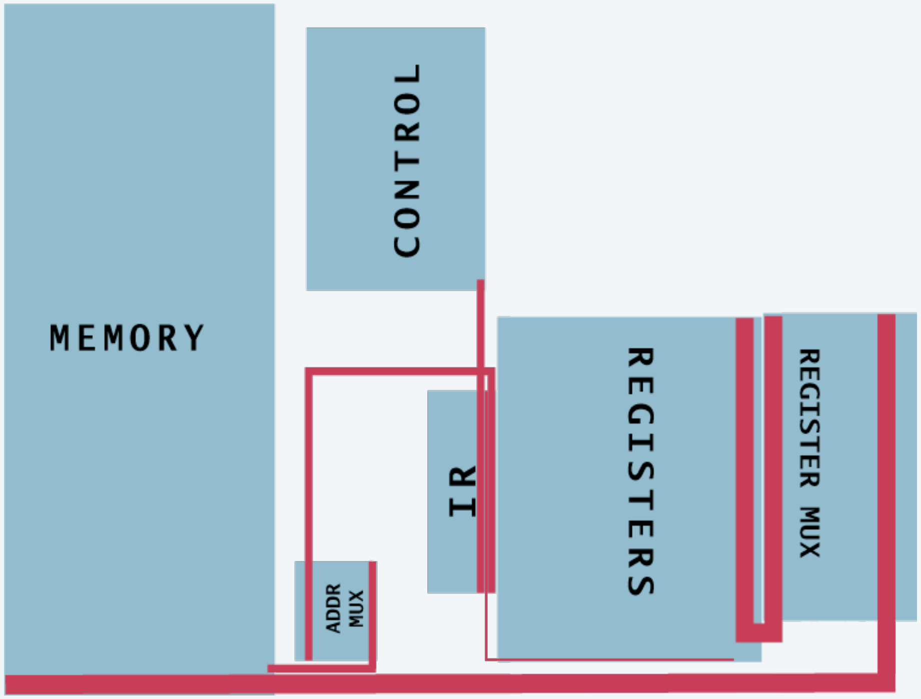
PC to mem addr MUX
IR enable write

INCREMENT



PC increment
PC enable write

EXECUTE

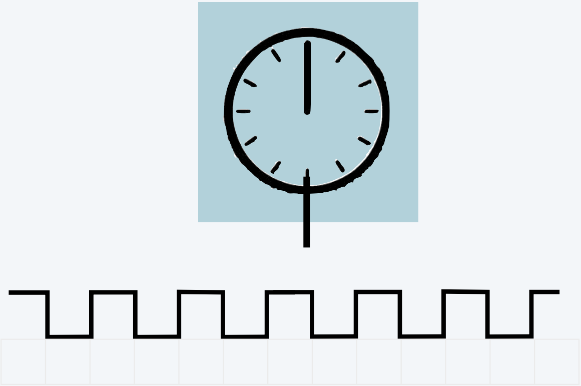


[IR opcode to control/decoder]
[decoder to ALU select]
[IR to addr MUX]
switch reg input MUX to memory
reg bank enable write

Clock

A CLOCK provides a regular ON-OFF pulse.

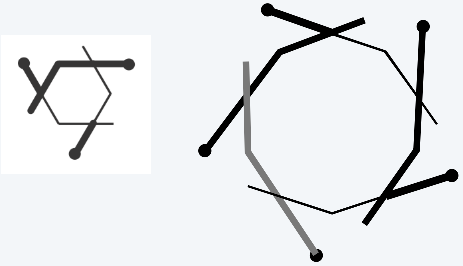
Requirement. Clock cycle longer than max switching time.

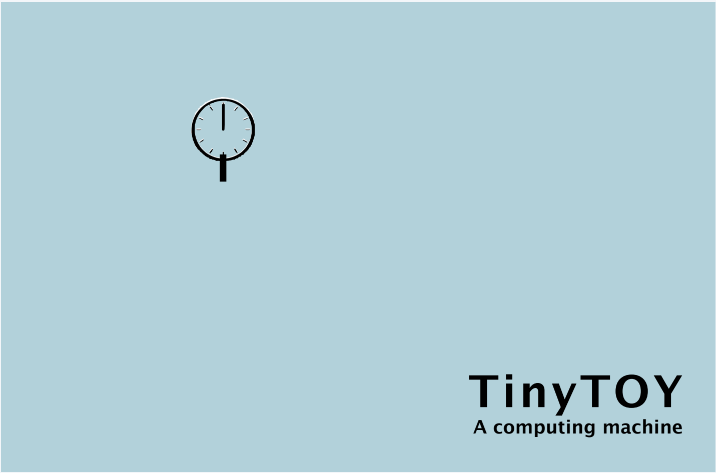


Q. How to implement a clock?

A. Use an external device.

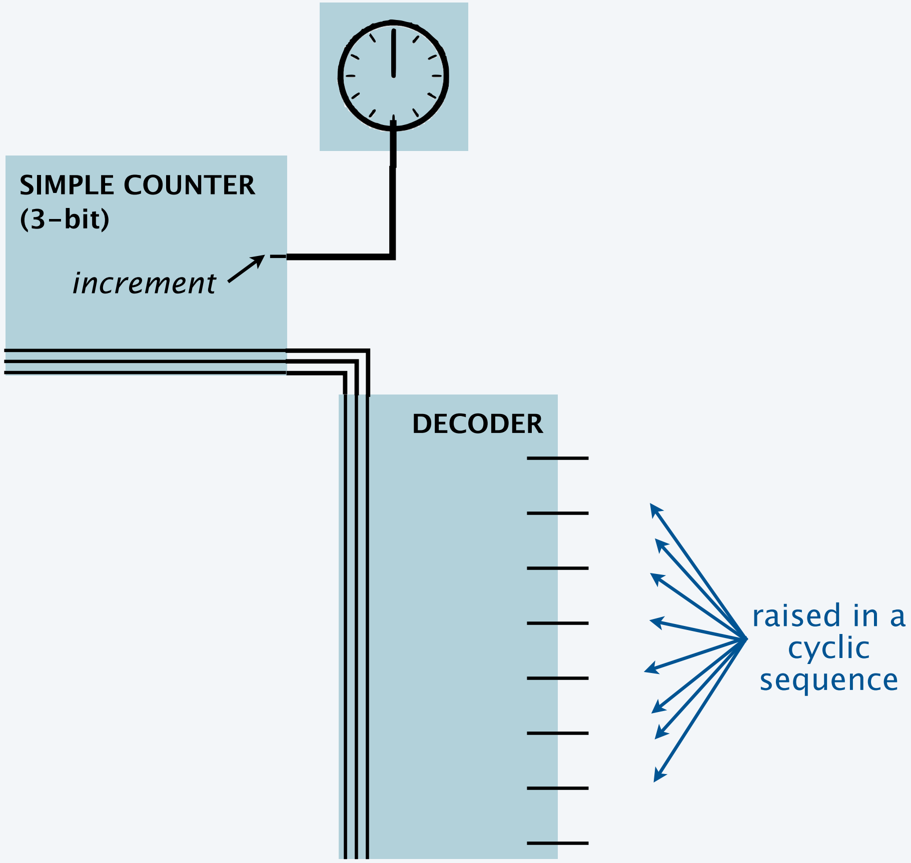
A. Use a *buzzer.*





Last step

Use a clock to raise a sequence of signals.



Brute force approach

- Connect clock to simple counter.
- Connect counter to decoder.

input bus and MUX not needed

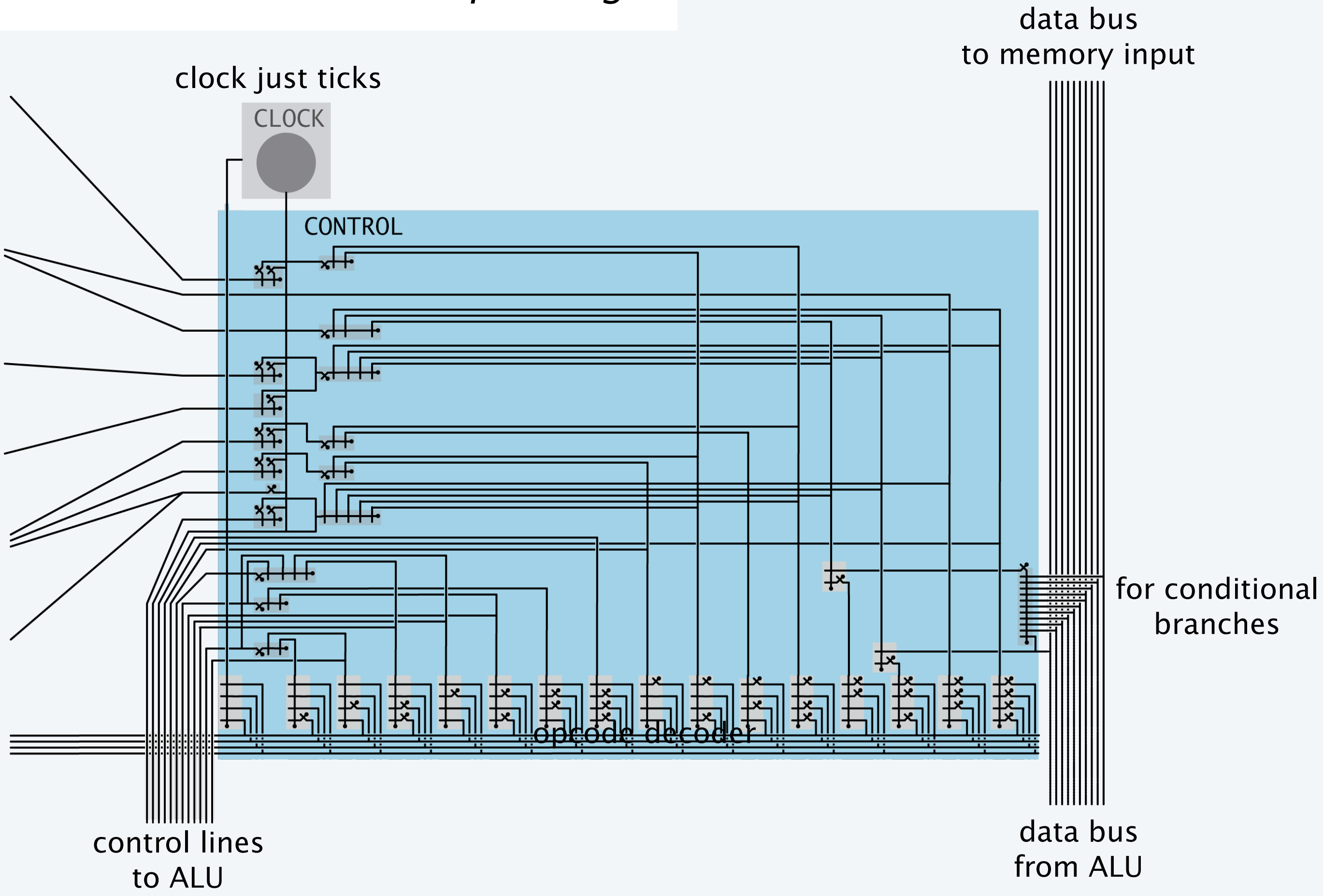
Result: A *sequence* of (control) signals.

Note. TinyTOY circuit (and your computer) uses a more efficient clocking scheme.

One final combinational circuit: Control

Control. The circuit that determines *control line sequencing*.

Control lines to processor registers and ALU are raised in sequence determined by clock and opcode.



Tick-Tock

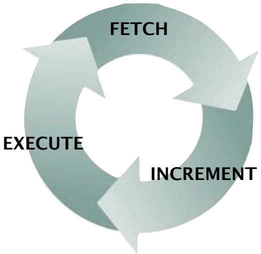
CPU is a circuit, driven by a clock.

Initalize via console switches.

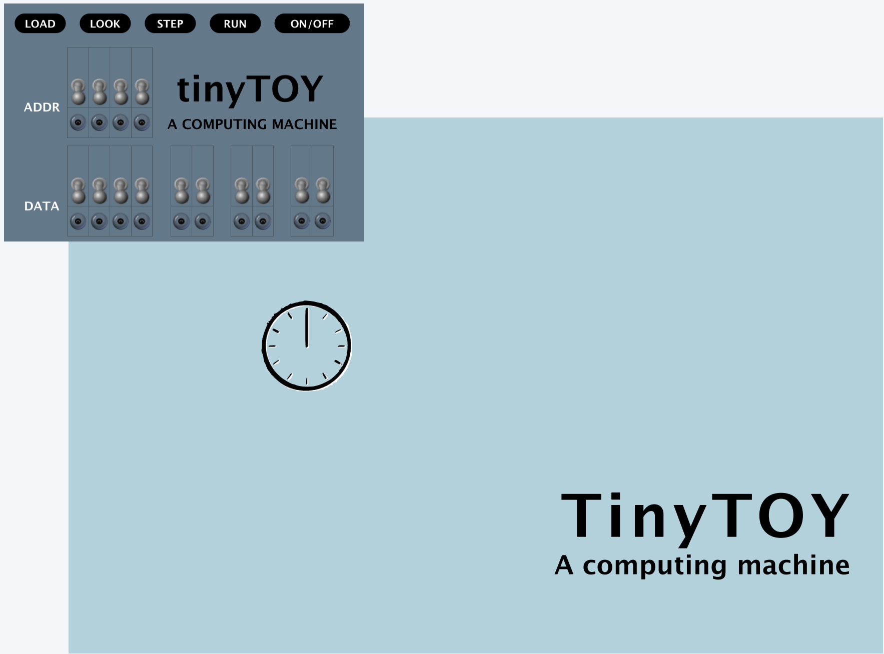
Press RUN: clock starts ticking

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

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

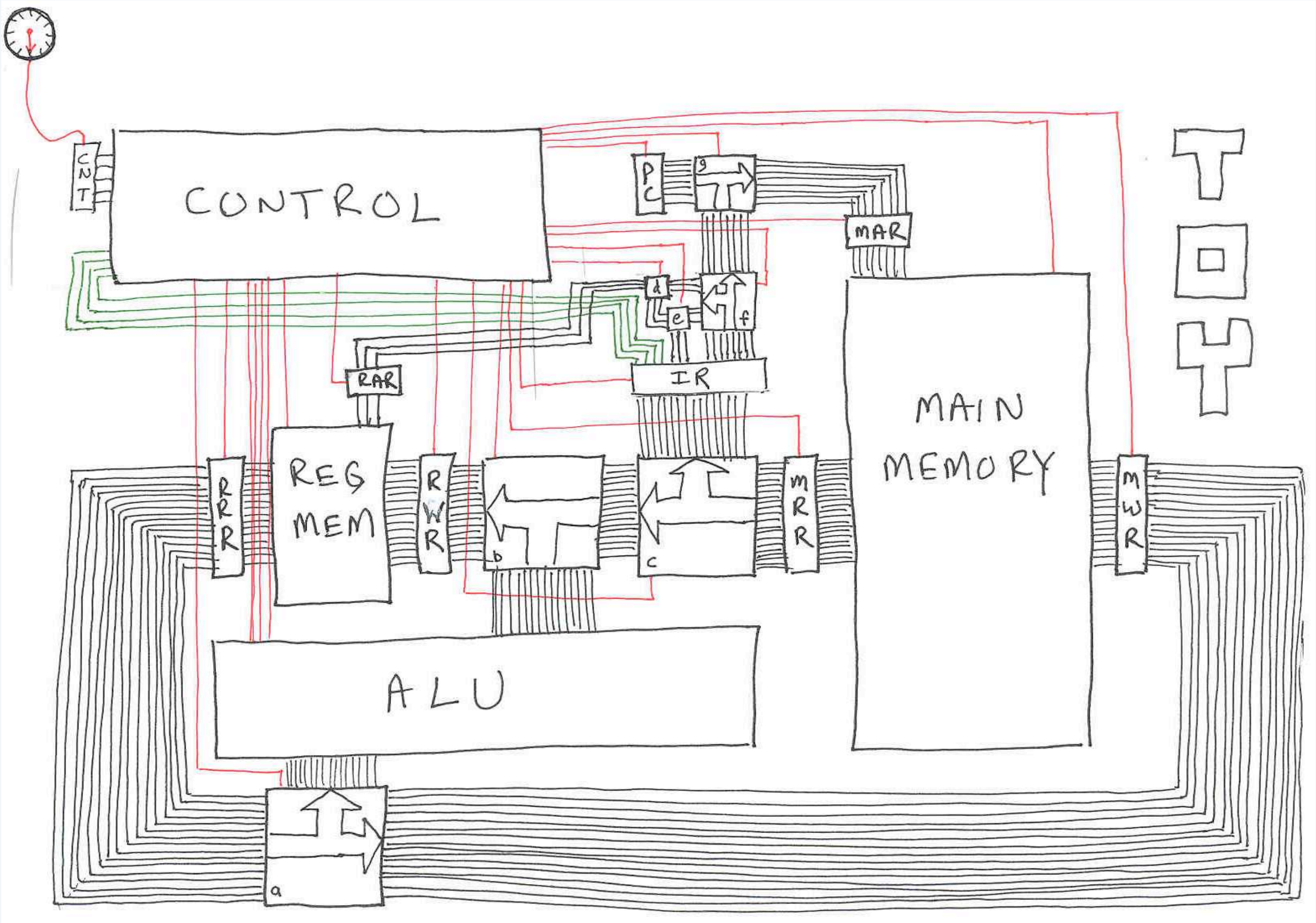


Faster clock? Faster computer!

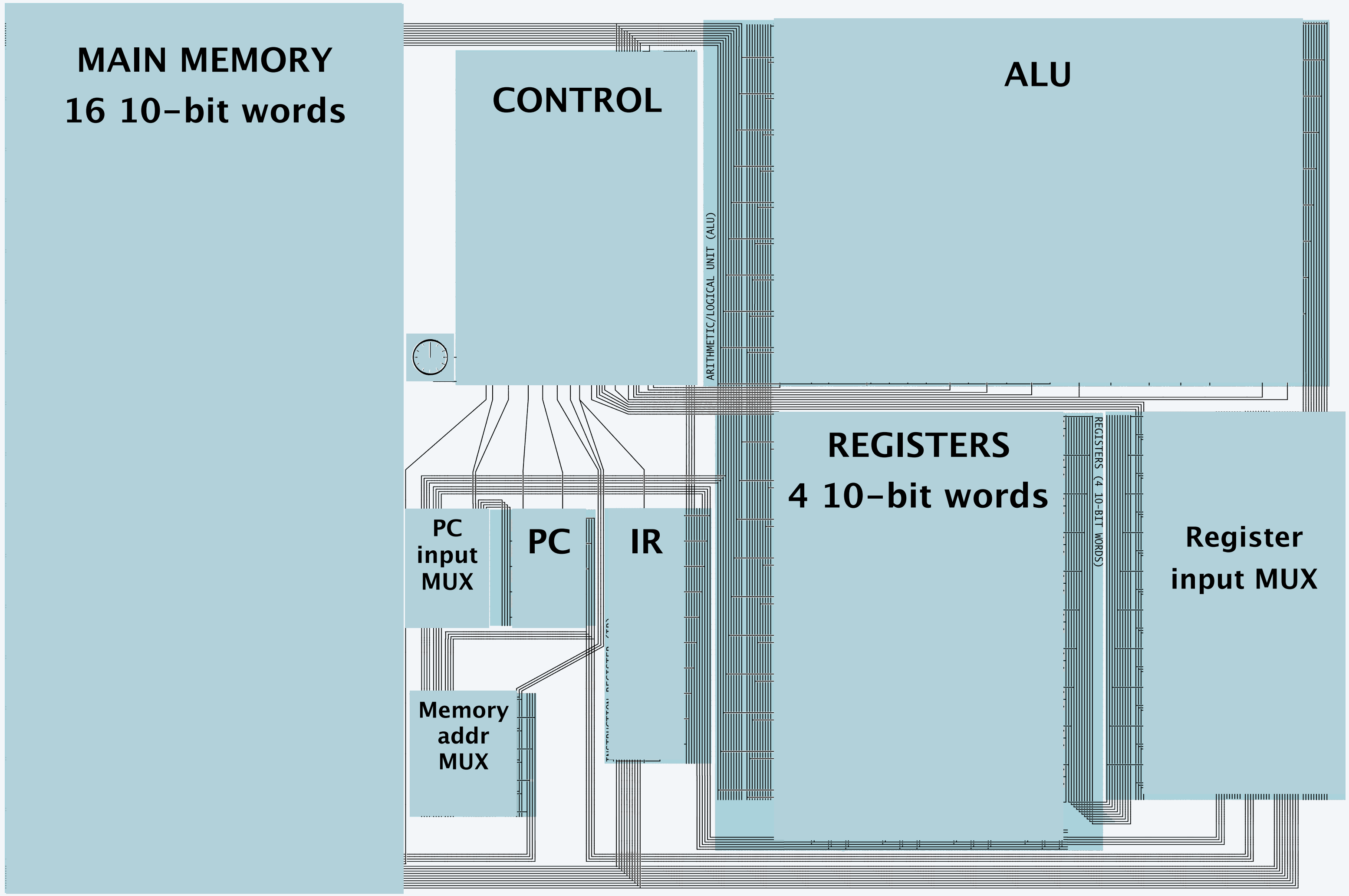


And THAT . . . is how your computer works!

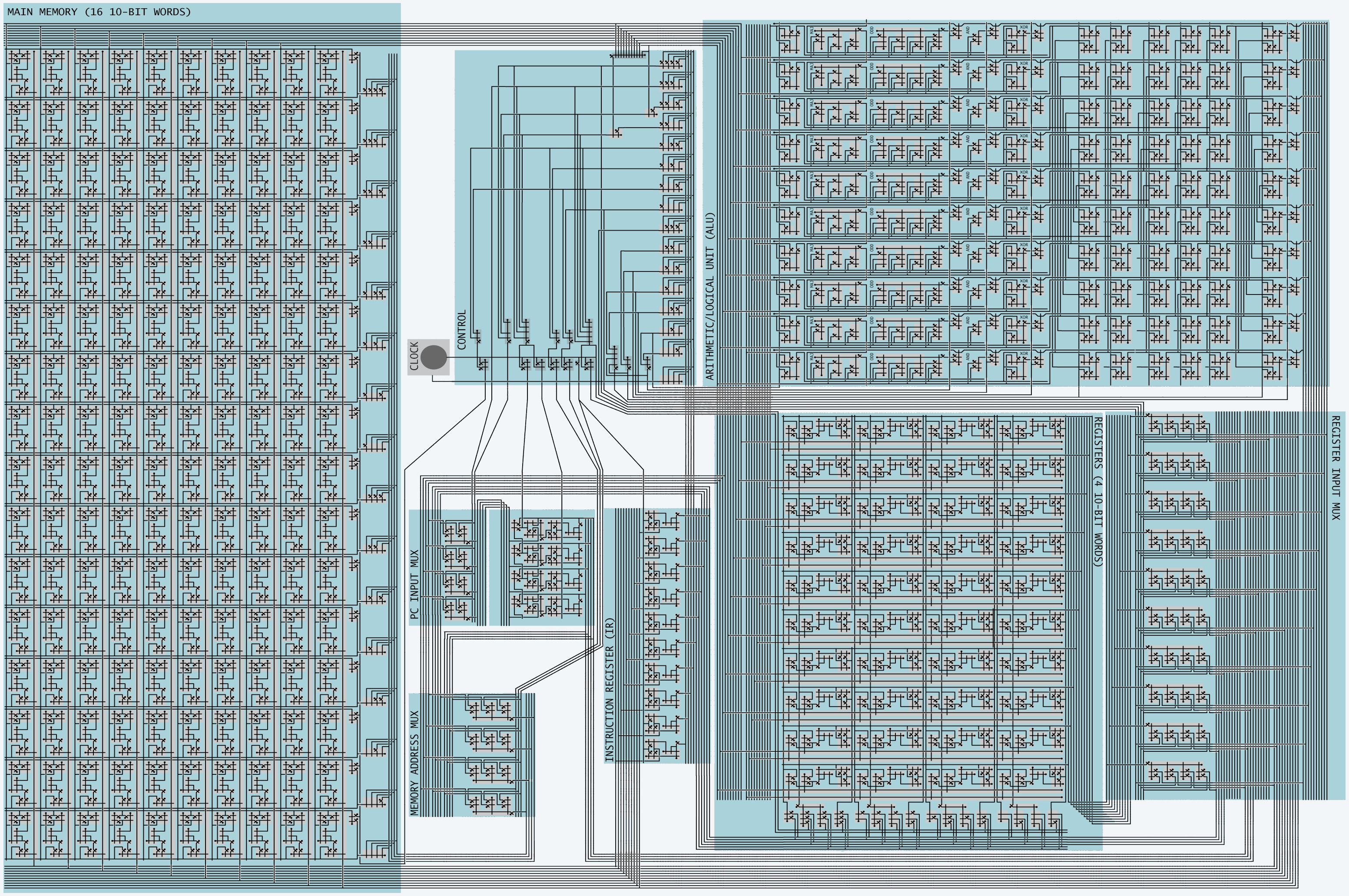
TOY "Classic", back-of-envelope design (circa 2005)



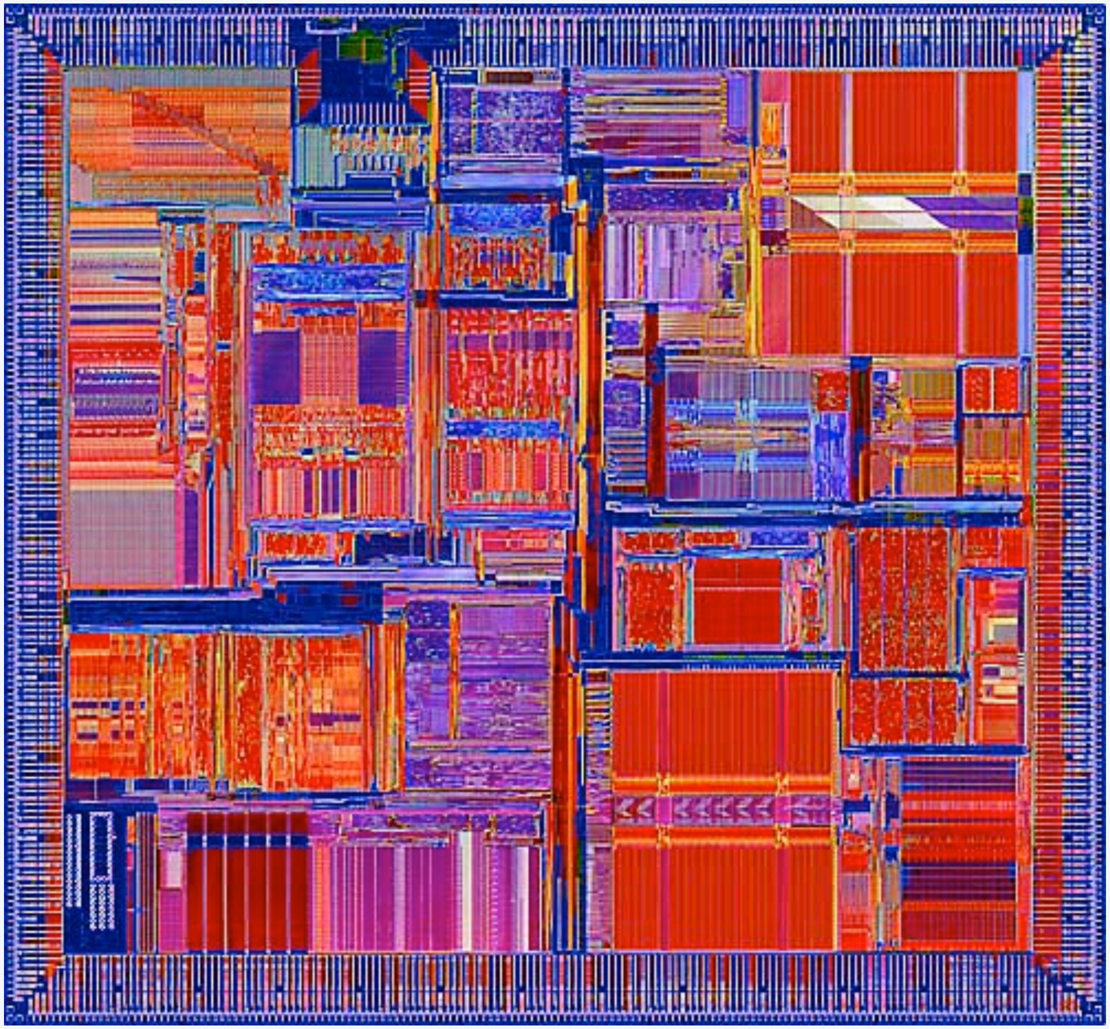
TinyTOY CPU component-level view



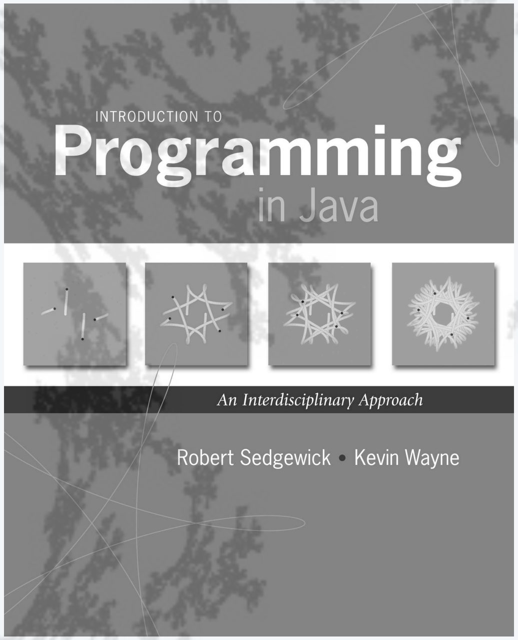
TinyTOY CPU switch-level view


MAIN MEMORY (16 10-BIT WORDS)
CLOCK
CONTROL
ARITHMETIC/LOGICAL UNIT (ALU)
REGISTERS (4 10-BIT WORDS)
REGISTER INPUT MUX
PC INPUT MUX
MEMORY ADDRESS MUX
INSTRUCTION REGISTER (IR)

A real microprocessor (MIPS R10000)



COMPUTER SCIENCE
SEDGEWICK / WAYNE



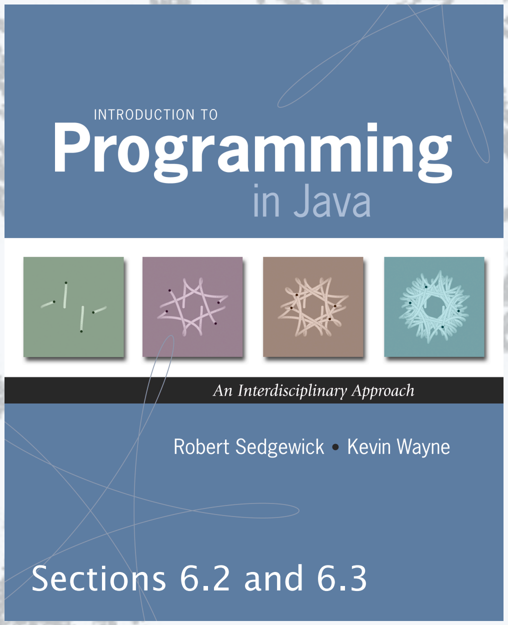
http://introcs.cs.princeton.edu

21. CPU

- Bits and registers
- Main memory and register banks
- Program counter
- **Putting the pieces together**

COMPUTER SCIENCE

SEDGEWICK/WAYNE



INTRODUCTION TO

Programming

in Java

An Interdisciplinary Approach

Robert Sedgewick • Kevin Wayne

Sections 6.2 and 6.3

http://introcs.cs.princeton.edu

21. Central Processing Unit