Building to a Computer

CS 2130: Computer Systems and Organization 1

Xinyao Yi Ph.D.
Assistant Professor

Announcements

- Homework 2 due Monday

Need one more thing to build computers

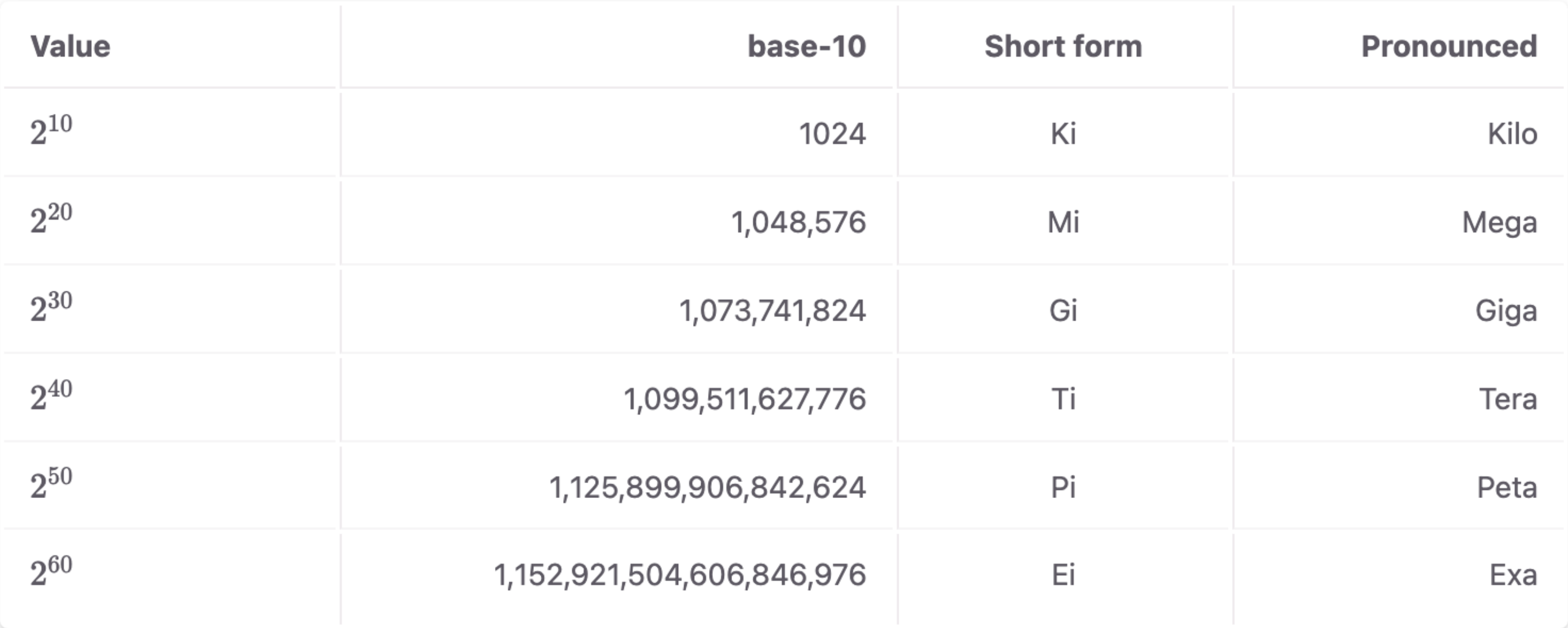
Memory and Storage

Registers $\approx$ KiB

- 6 gates each, $\approx$ 24 transistors
- Efficient, fast
- Expensive!
- Ex: local variables

These do not persist between power cycles

Memory and Storage



Value	base-10	Short form	Pronounced
2^{10}	1024	Ki	Kilo
2^{20}	1,048,576	Mi	Mega
2^{30}	1,073,741,824	Gi	Giga
2^{40}	1,099,511,627,776	Ti	Tera
2^{50}	1,125,899,906,842,624	Pi	Peta
2^{60}	1,152,921,504,606,846,976	Ei	Exa

Memory and Storage

Memory $\approx$ GiB

- Two main types: SRAM, DRAM
- DRAM: 1 transistor, 1 capacitor per bit
- DRAM is cheaper, simpler to build
- Ex: data structures, local variables

These do not persist between power cycles

Memory and Storage

Disk $\approx$ GiB-TiB

- Two main types: flash (solid state), magnetic disk
- Magnetic drive
  - Platter with physical arm above and below
  - Cheap to build
  - Very slow! Physically move arm while disk spins
- Ex: files

Data on disk does persist between power cycles

Putting it all together

- Information modeled by voltage through wires (1 vs 0)
- Transistors
- Gates: & | ~ ^
- Multi-bit values: representing integers, floating point numbers
- Multi-bit operations using circuits
- Storing results using registers, clocks
- Memory

Code

How do we run code? What do we need?

Consider the following code:

```
...
8:    x = 16
9:    y = x
10:   x += y
...
```

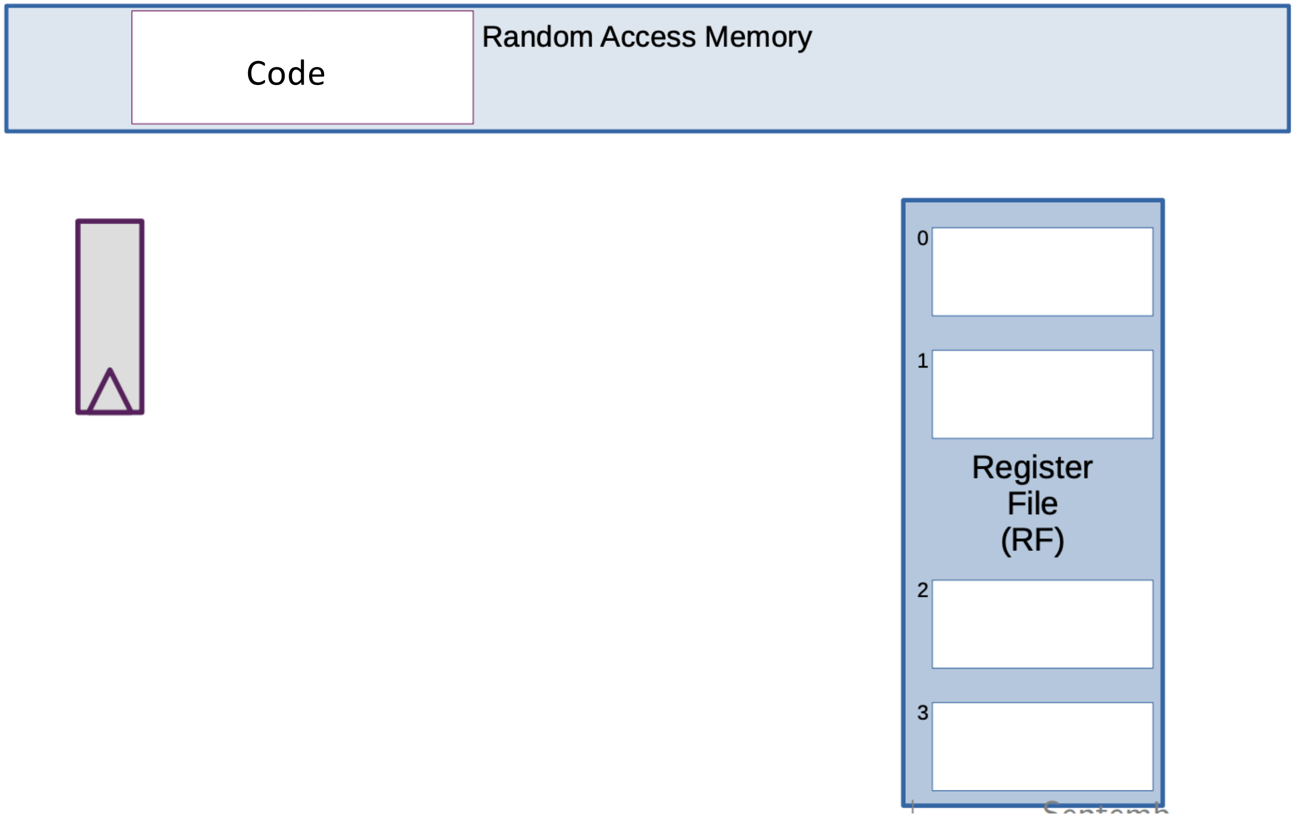
What is the value of *x* after line 10?

Bookkeeping

What do we need to keep track of?

- **Code** - the program we are running
    - RAM (Random Access Memory)
- **State** - things that may change value (i.e., variables)
    - Register file - can read and write values each cycle
- **Program Counter (PC)** - where we are in our code
    - Single register - byte number in memory for next instruction

Building a Computer

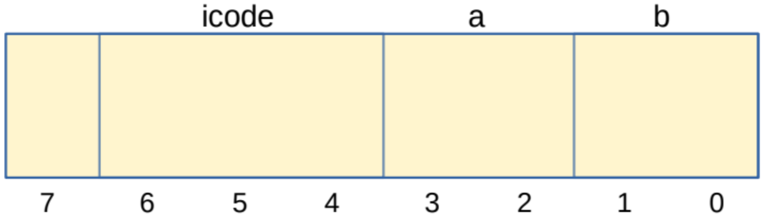


Encoding Instructions

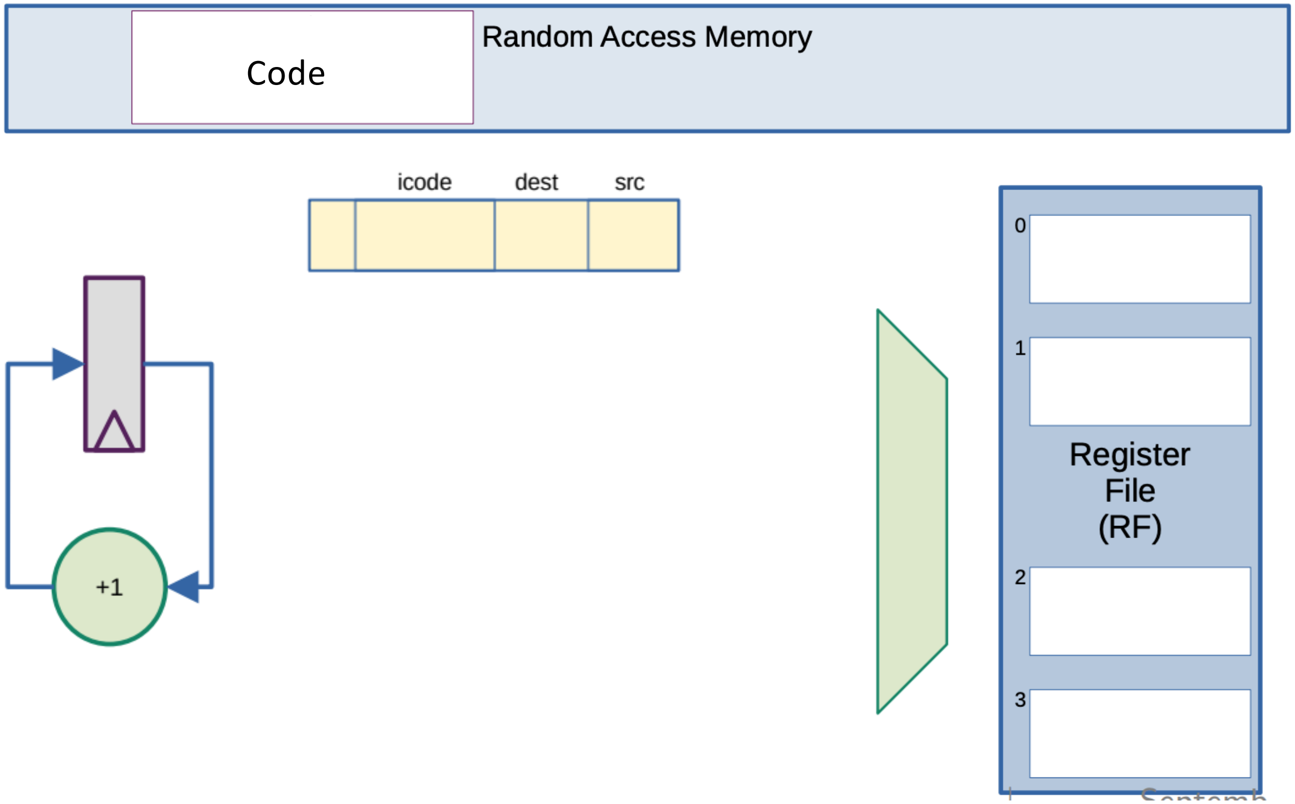
Encoding of Instructions (**icode** or **opcode**)

- Numeric mapping from icode to operation

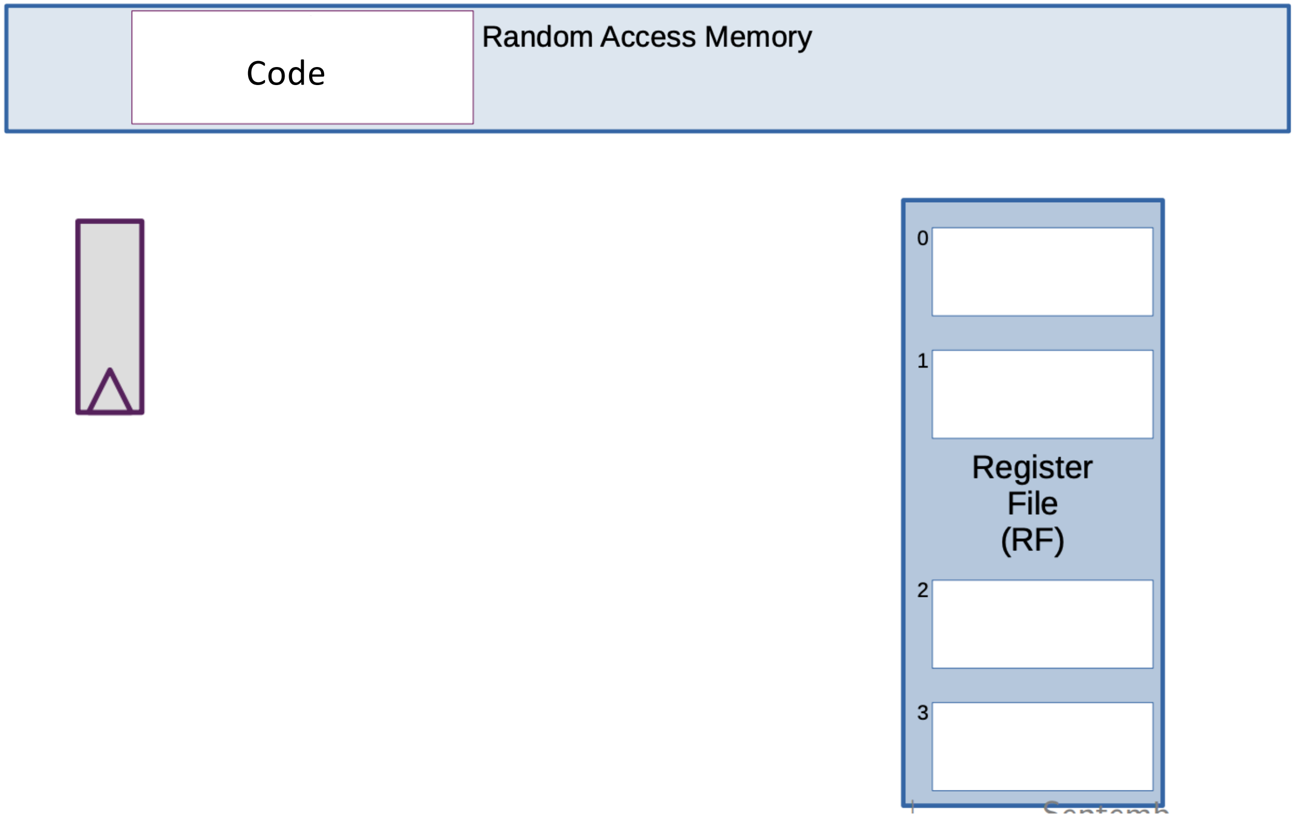
icode	meaning
0	`rA = rB`
1	`rA &= rB`
2	`rA += rB`
...	...



Building a Computer



Building a Computer


Random Access Memory
Code
0
1
Register
File
(RF)
2
3