

How a Computer Works

What is it? What is it good for?

- Definition of “computer” prior to 1940s?
- *Mind tool, or intelligence amplifier*
- *A concept manipulator*

What does it do?

- Accepts inputs
- Manipulates them (transforms them)
- Produces outputs



- So, what distinguishes the computer from other machines or tools?
- With computers, what *else* is needed to do the transformation, then?
 - How is that *different* from other machines/tools??
- In that case, what would be another good descriptor?
- Example of *special-purpose* computer?

How?

A computer system

- **HARDWARE:** physical components
 - Input (such as?)
 - Processing: CPU (microprocessor chip)
 - Storage (two kinds; physically, radically different)
 - Output (examples?)
- **SOFTWARE:**
 - *Program (code):*—step-by-step instructions that tell the computer *what to do* and *with what data*.
Instructions are *imperative* and are carried out one after another.

- CPU

- Control and manipulation of data occur here.
- Simple minded!

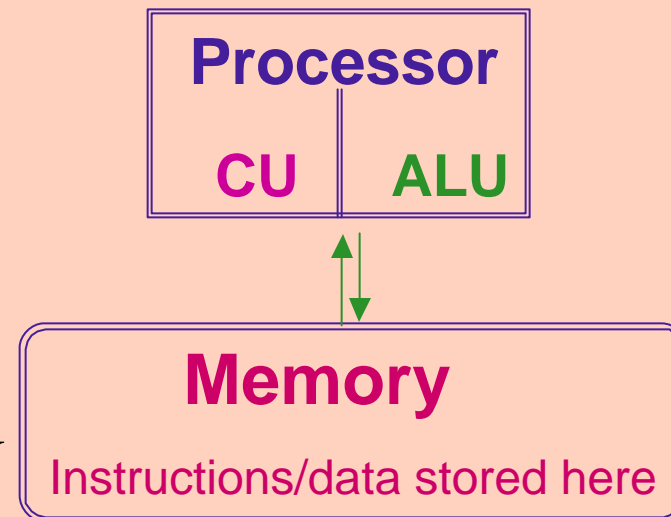
Two parts:

1. Control unit

- FETCHES INSTRUCTION
- DECODES INSTRUCTION
- Tells the Arithmetic & logic unit *what* to do.

2. Arithmetic & logic unit

- Adds, subtracts, multiplies, divides, and tests things (compares).



- **MEMORY:** a place to hold data/information and instructions.

“Two-level storage”

- Primary memory

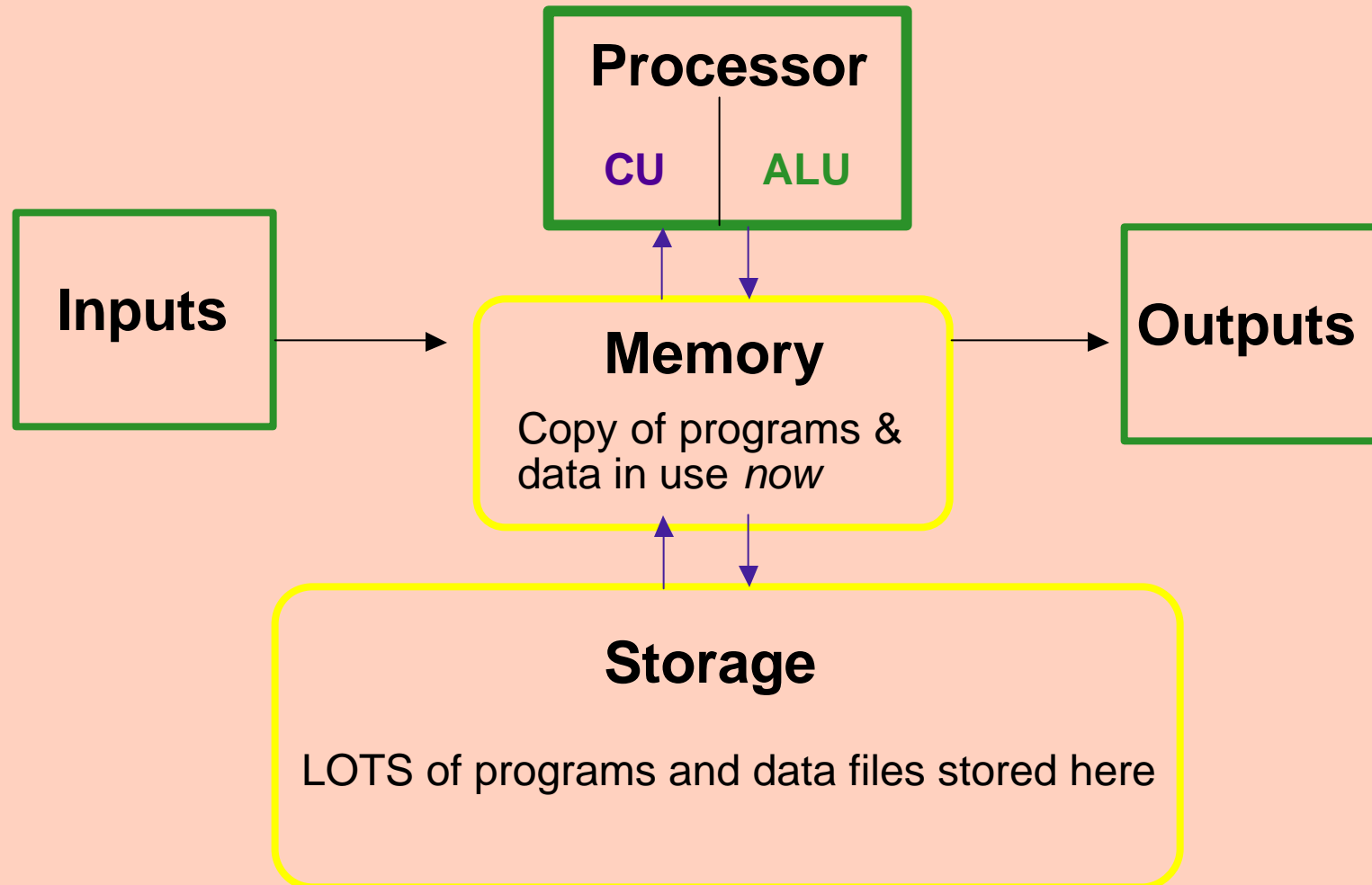
- RAM: “User” memory. Temporary. Volatile.
- Fully electronic (data stored as electrical charges—no moving parts). FAST!

- Secondary memory (Auxiliary)

- Permanent, long-term, plentiful, cheaper.
- Examples?
- Access: at least 10,000 times slower than primary memory. Why?

- Why do we need secondary?

Logical organization



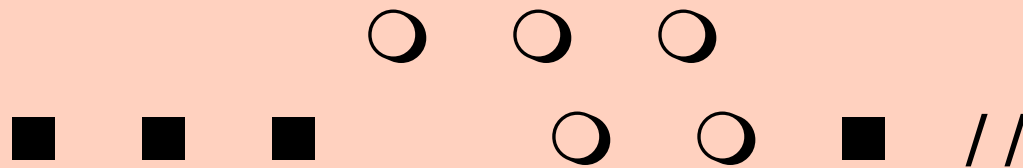
Data/information representation

- How humans communicate with each other...
- What are the five kinds of “information”?
- We want to use a *single* way to represent *all* these forms of communication:
 - Because we want to use an *electronic* computer to manipulate them all.
 - The most basic component: the SWITCH ...
 - Therefore, to use switches to represent our many forms of communicating, we first need to *encode* those forms.

- *Data & instructions* can be encoded as **numbers**, which are associated with *parts* of an electronic machine (switches) and their *state* at a given moment.
- What kind of basic switch do we use every day?
 - how many states/conditions does it have?
 - Why use such a simple switch?
- First: representing *decimal numbers*

Something about numbering systems:

- Additive: *//////////* (twelve)



- Positional: 12 (ten plus two)

- “Face” value
- “Place” value—depends upon the **base**
- **Base** determines *number* of unique symbols used

- Decimal system (base ten): Arabic nos. 0–9



ten^5	ten^4	ten^3	ten^2	ten^1	ten^0
100,000	10,000	1,000	100	10	1 (place value)

- All positional numbering systems work alike.

0 thru 9

– **Decimal:**

Decimal digits

Base 10

0 and 1

– **Binary:**

BInary digit**S**
called “Bits”

Base 2

←	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
←	64	32	16	8	4	2	1	(place value)

							0
							1
						1	0
						1	1
				1		0	0
				1		0	1
				1		1	0
				1		1	1
		1		0		0	0
		1		0		0	1
		1		0		1	0

- Making sense?
 - How do you represent decimal value 37 in *binary* code?
 - What about decimal value 63?

- Just for culture; try these on your own...
 - What is the decimal number value of each of these?

1111

10111

1000110

- Recap: *Decimal* numbers can be represented & stored *logically* in *binary* form as bits, and *physically* with switches.

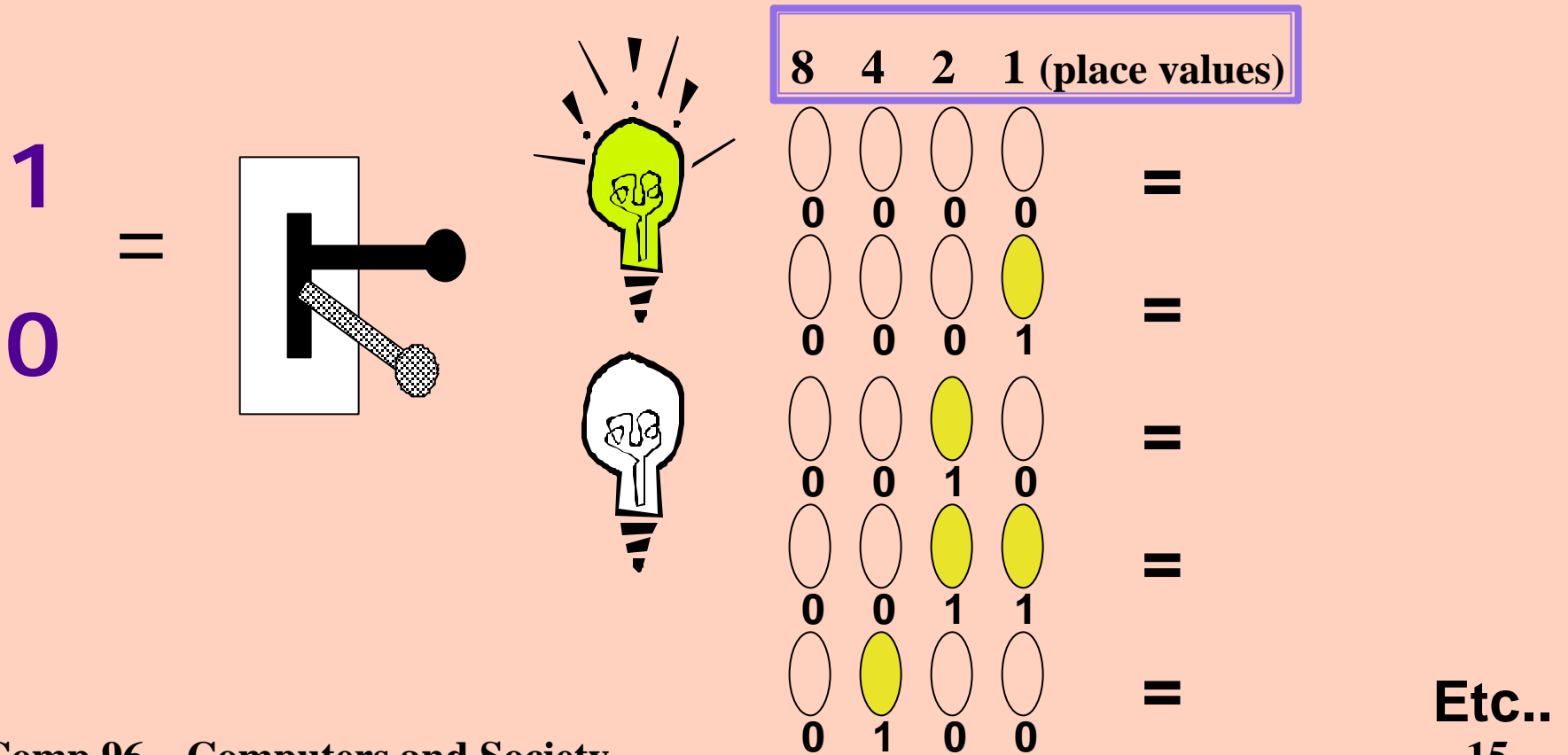
- Numbers are associated w/machine parts and what *condition* each part is in at that moment:

1 0 0 1
On Off Off On

- Binary system: Allows computer to represent *decimal* values as a collection of on/off signals.

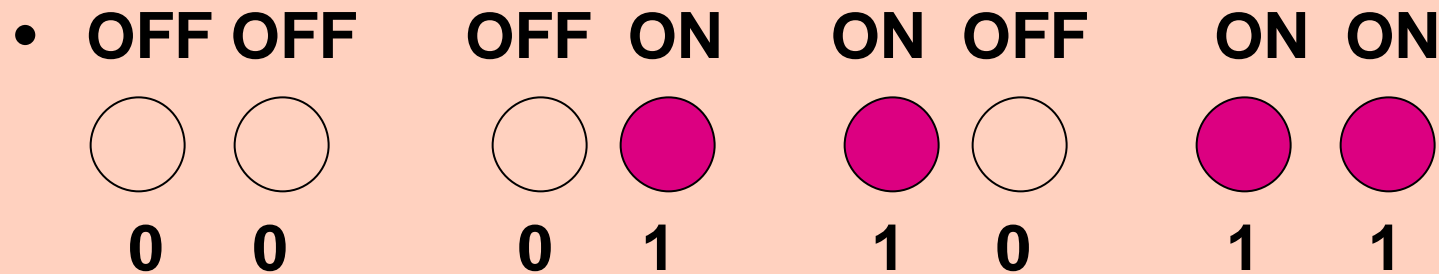


- Logical structure, then:
 - Binary **0**s and **1**s: **Binary digits. Bits.**
 - **Bit**: smallest, most basic *data unit* in a computer.
- So how do we represent a bit "in the box"?



- What about representing numbers used only as *text*; and *letters* and *symbols*?

- Binary **codes**: unique *bit patterns* of 1s and 0s
- 2 switches (two-bit code) can represent four different things:



Just
for
Culture:

- 4 switches (four-bit code): enough *different* combinations to represent 16 things.

- Four switches, two possible conditions each: 2^4
- Enough to represent all decimal digits (used as **text**).

- Eight-bit code (8 switches): 256 unique things
- Schemes: ASCII; EBCDIC; Unicode ...

8-bit ASCII sample

0	0101	0000
1	0101	0001
2	0101	0010
A	0100	0001
B	0100	0010
C	0100	0011

Hello

01001000 01100101 01101100 01101100 01101111

- Representing Pictures (RH-X) (*just for culture*)
- Representing Sounds (RH-X) (*just for culture*)
- Representing Instructions (RH-X) (*more shortly*)

“Symbol-processing machine”

Units of storage:

Single binary digit	bit
7 or 8 bits:	byte (one character)
1 kilobyte (KB):	about 1000 bytes (2^{10})
1 megabyte (MB):	about a million bytes (2^{20})
1 gigabyte (GB):	about a billion bytes (2^{30})
1 terabyte (TB):	about a trillion bytes (2^{40})

Analog and digital

- Most everything around us: *continuously varying* intensities or values.
 - ANALOG: quality reduction w/reproduction.



- Everything represented in the computer is stored as *discrete*, “countable” units. What are the units *called* (logical form)? Physical form?

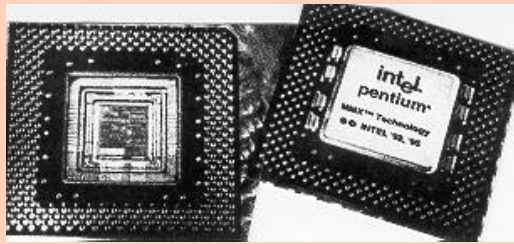
- DIGITAL: copy is exactly like original.



Word Game (RH-X)

- Who/what was the *processor* in this game?
- What was the *basic instruction set* used for?
- How did you know *what* to do in *what order*?

- How are *instructions* different from *data*?
- What was the output?
- What served as the input (raw data)?



Silicon-based unit

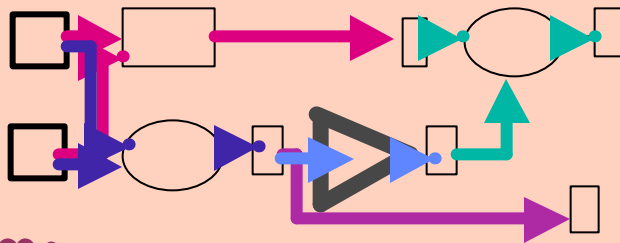
CPU (*obeys orders to transform raw data into meaningful info.*)

Basic instruction set:

Primitive commands (log'l) **it can** do with

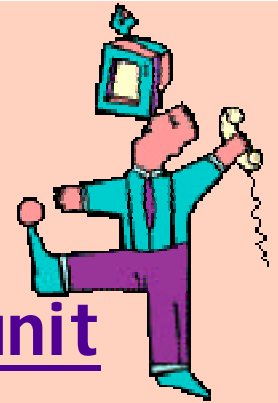
STO; ADD; SUB; MUL; DIV; INC; CMP; JMP...

hardwired computer circuits (phy'l).



Program:

Tells CPU *which* to do in *what* order.



Carbon-based unit

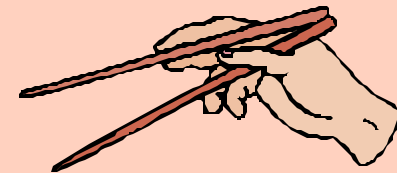
You (*obey orders to transform pages of words into meaningful message.*)

Basic instruction set:

Simple commands **you can** do with

GOTO #; SELECT LINE #, FORWARD #; BACKUP #; ;...

Hardwired skills; hands; eyes...



Program:

Tells you *which* to do in *what* order.