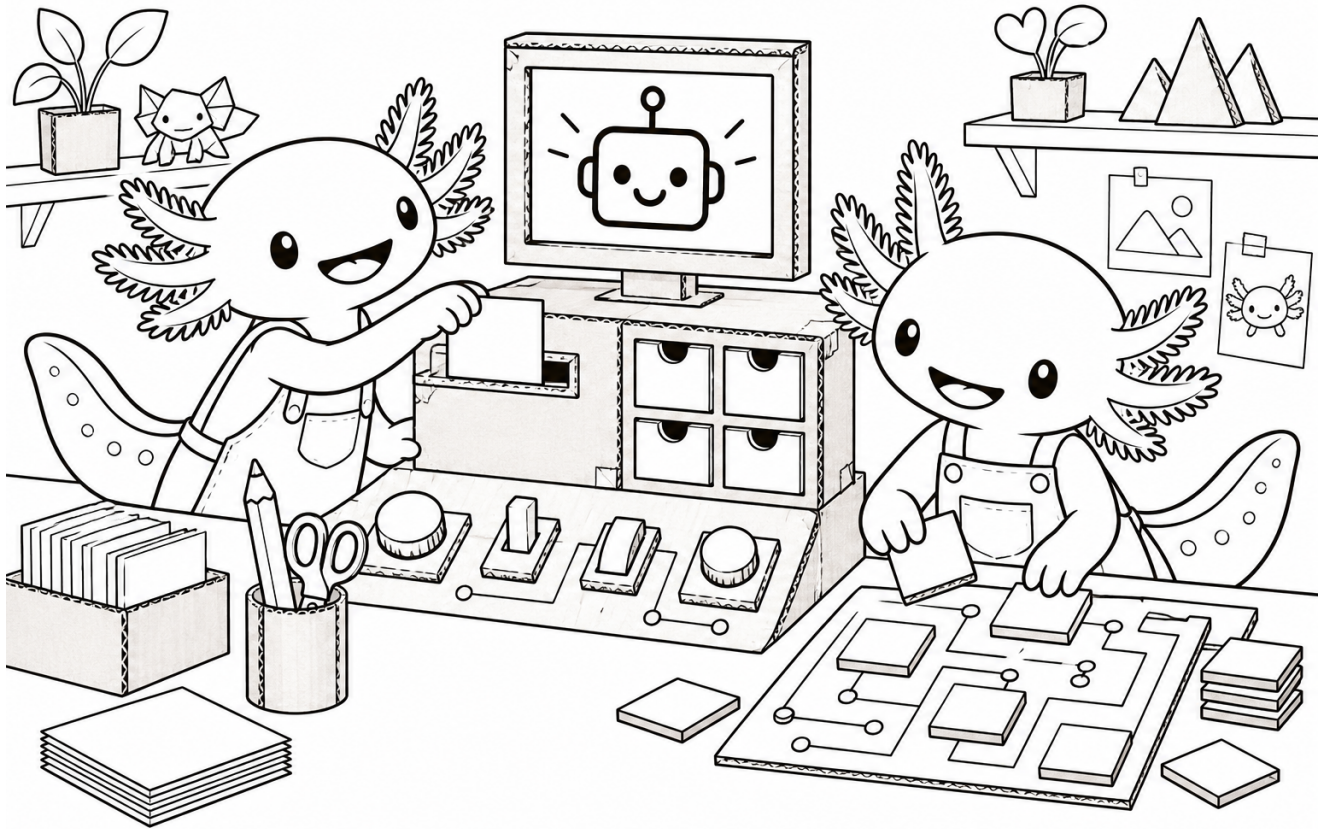


Paper Computer Engineering

Build and run a tiny computer with the axolotl lab team.



20 activities | Ages 7-10 with a grown-up

GATHER

This single-sided pack, a pencil, crayons and child-safe scissors. All model parts are supplied. Keep cut pieces for later pages.

CHOOSE A LAB VISIT

Build: 1-5. Logic: 6-9.
Programs: 10-15.
Messages: 16-19. Exhibit: 20.
Try a few pages at a time.

No electronics, software, glue or special kit needed.

The axolotls are pretend. You carry out the paper rules; this model is not an electronic computer. Guides and complete answers are included.

Meet the paper-machine team

Read the jobs. Write INPUT, CPU, MEMORY or OUTPUT in each blank.

1

A button supplies the number 2.

2

The processor follows ADD 3.

3

A labeled place keeps the number 5.

4

The display shows the result 5.

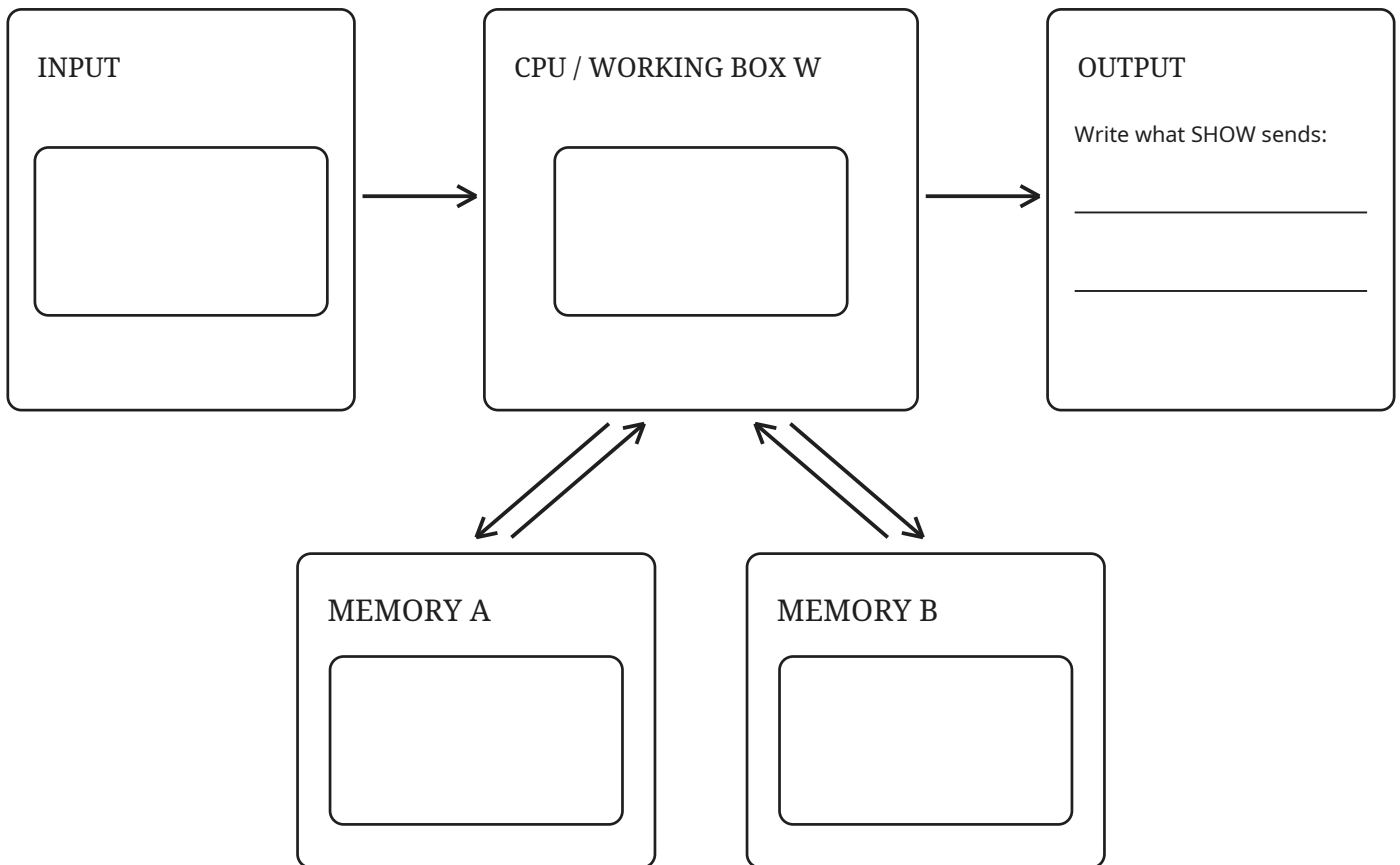
CPU = central processing unit

Write INPUT, CPU and OUTPUT in the three boxes. Draw arrows to connect them in order. Tell someone where MEMORY helps.

This is one simplified journey. Real computer memory can hold both instructions and data. The parts cooperate; memory is not a person's memory.

Build your machine board

Keep this page whole. Cut the number cards on the next page. The smaller program cards arrive in Activity 12.



STORE: W -> A or B. LOAD: A or B -> W.

PROGRAM / move the NEXT marker along these six spaces

1	2	3	4	5	6

Start W, A and B at 0. Use cards or pencil. INPUT holds a chosen starting number. SET puts the chosen number into W. SHOW copies W to the output record.

W models one working register. A and B are two data slots. The separate program row is a teaching simplification; it is also stored information.

Make the reusable card kit

Cut the dashed borders. Put a 0 in W, A and B on your board; write the third 0 on a blank card.

0	1	2	3
4	5	6	7
8	9	0	1
2	3	4	5
6	7	8	9
_____	_____	_____	_____

NEXT

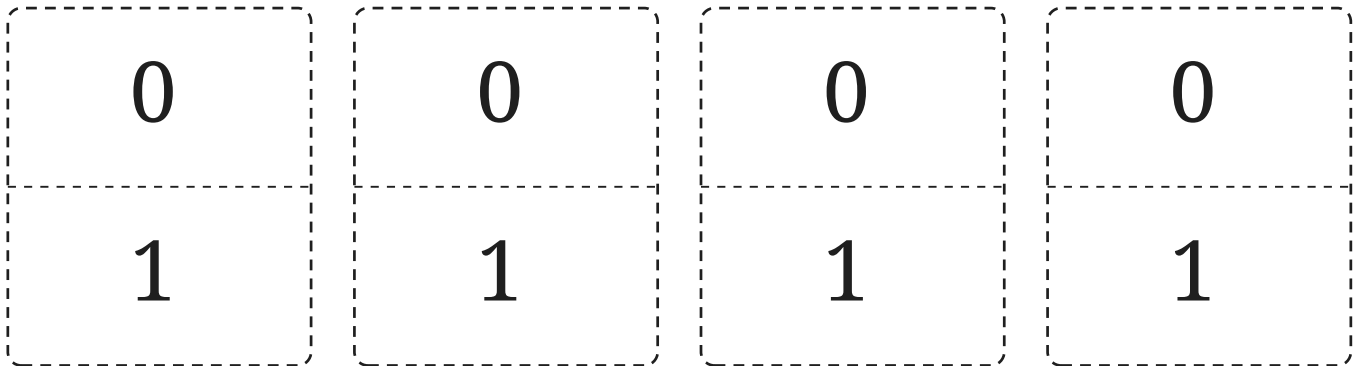
Blanks make extra copies. When you STORE or LOAD, keep the original value in its place and use a second card for the copy. Mark blank cards in pencil.

These cards fit the working and memory spaces on page 3. Use the same PDF paper size for both sheets, and ask a grown-up to help with scissors.

Flip a two-state tile

Cut four rectangles. Fold each on its middle dotted line so 0 is on one face and 1 is on the other.

A bit has two possible values: 0 or 1. Our folded tile shows one face at a time. Flip a tile to change its state. Keep these four tiles for Activity 5.



Start one tile at 0. Follow every instruction.

Start	FLIP once	FLIP again	KEEP it
0			

Two bits can show four patterns. Fill the missing pair:

00 01 10 _____

Electronic bits are represented by physical states, often voltage ranges. Our 0 and 1 are labels, not a claim that 0 means no electricity anywhere.

Four bits, sixteen numbers

Place your four bit tiles below. Add a column's value only when its tile shows 1.

8	4	2	1

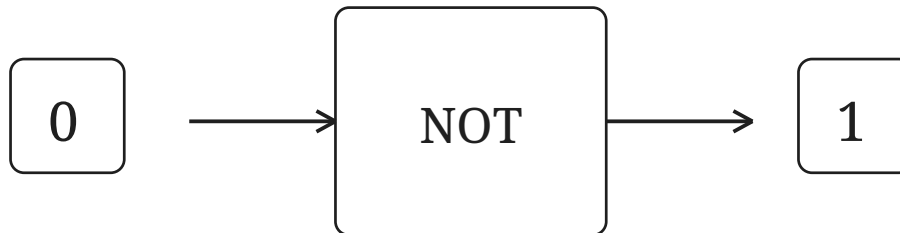
Example: 1001 means $8 + 1 = 9$. The 4 and 2 columns are 0, so they add nothing. The left column always stays 8 in this four-bit model.

Number to make	8	4	2	1
0				
3				
6				
9				
12				
15				

Four bits have 16 possible patterns, representing 0 through 15 in this unsigned-number code. Other codes can give bit patterns other meanings.

NOT changes the bit

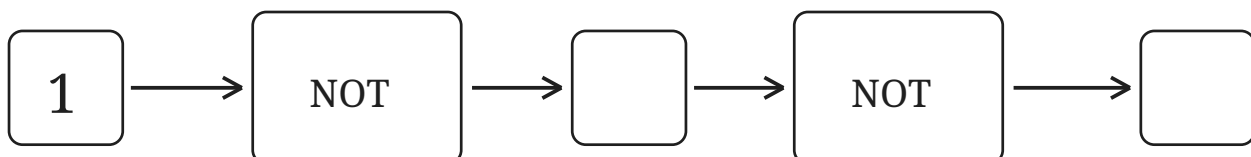
A NOT gate gives the opposite value: 0 becomes 1, and 1 becomes 0. Fill each output.



Worked example

Input	Output after NOT
0	
1	

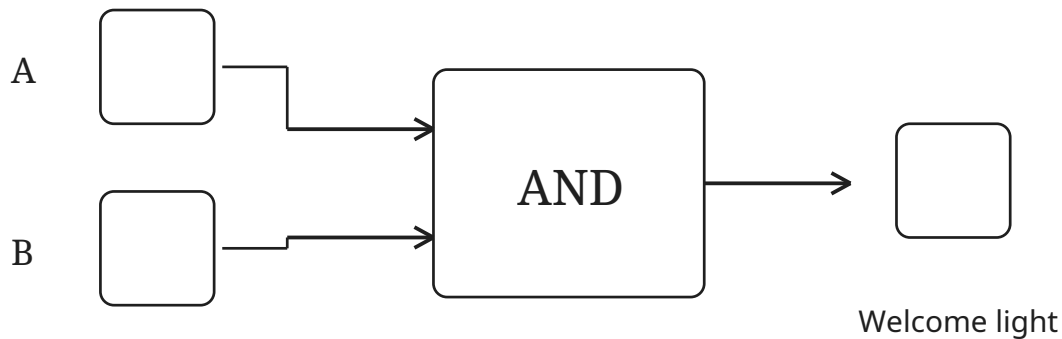
Now follow two NOT gates in a row.



Gate boxes are simplified circuit symbols. A real gate has electrical inputs and an output. These are logic rules, not instructions for wiring electronics.

AND needs both

The lab's welcome light is 1 only when both flags A and B are 1. Complete every input case.



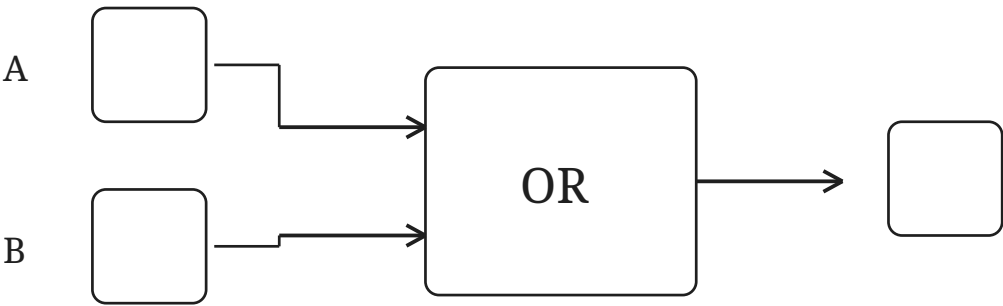
Flag A	Flag B	AND output
0	0	
0	1	
1	0	
1	1	

Circle the only row that turns the light on. Why?

AND means both inputs must be 1. This complete list of cases is called a truth table; it lets you check the rule systematically.

OR welcomes either

A greeting signal is 1 when A is 1, B is 1, or both are 1. Fill the outputs and compare the rules.



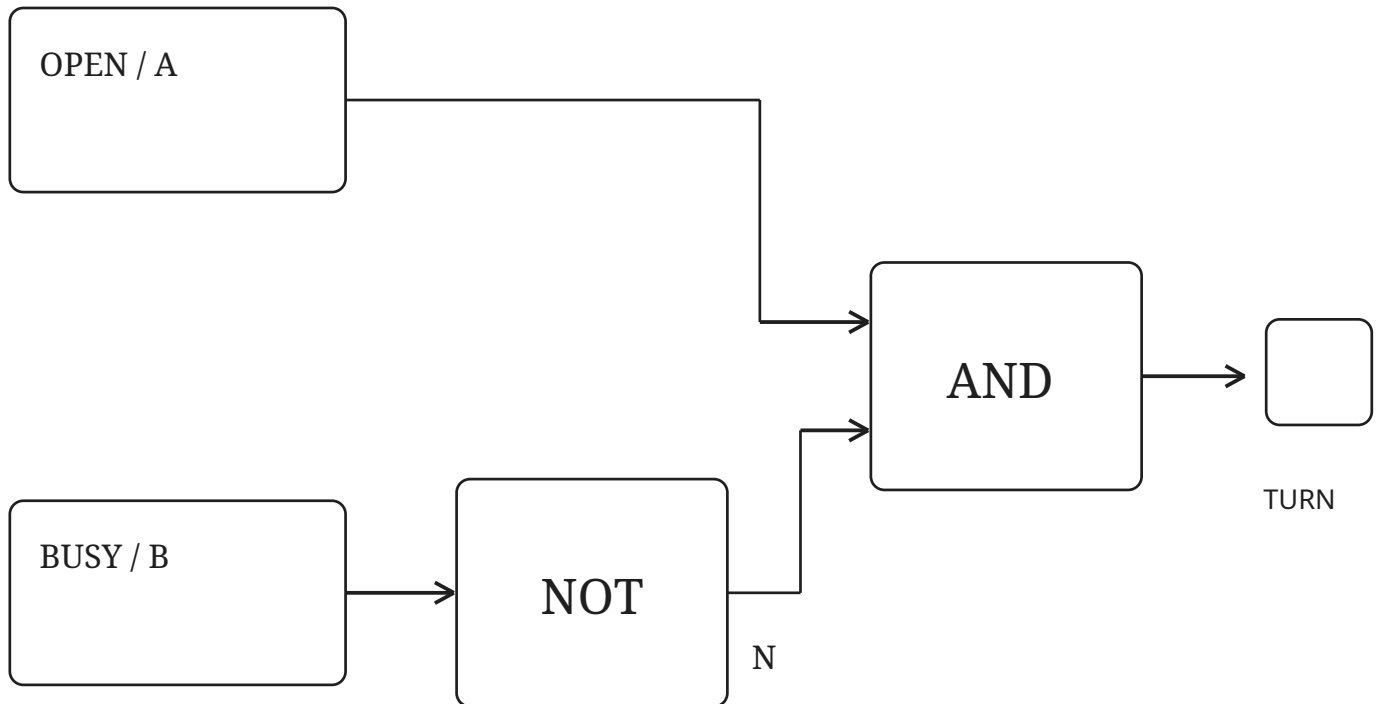
A	B	OR output	AND output
0	0		
0	1		
1	0		
1	1		

Which two input rows give different OR and AND results?

In this rule, OR includes the case where both inputs are 1. It does not mean 'one but never both.' The greeting is a fictional use of the logic.

Connect a two-gate rule

A game allows a turn when OPEN = 1 and BUSY = 0. First find NOT BUSY, then use AND.



A: OPEN	B: BUSY	N: NOT B	A AND N
0	0		
0	1		
1	0		
1	1		

Only A=1 and B=0 allows a turn. Complete the intermediate column before the final column; a gate's output can become another gate's input.

Save a number. Get it back.

Use your board and cards. Begin W=0, A=0, B=0. Record W and A after each command.

SET n

Put number n in W.

STORE A

Copy W into A; keep W.

LOAD A

Copy A into W; keep A.

SHOW

Copy W to the output record.

Command	W after	A after	Output so far
SET 3			
STORE A			
SET 1			
LOAD A			
SHOW			

After LOAD A, does A still contain its number?

A and B are addresses: labels that tell you which slot to use. STORE replaces the value in the named slot. LOAD makes a copy; it does not empty memory.

Run six small instructions

Start W=0, A=0, B=0. Follow the program from top to bottom. Write the state after every step.

ADD n

Replace W with $W + n$. Keep A and B.

STOP

End the program. Nothing else changes.

Step	Instruction	W	A	Output so far
1	SET 2			
2	ADD 3			
3	STORE A			
4	ADD 1			
5	SHOW			
6	STOP			

The output is _____. The saved value in A is _____.

Each row is one instruction in our model, not one real CPU clock cycle. Real processors can take several cycles or overlap work. All paper values here stay 0-9.

Put the program in order

Cut these six cards. Use all of them once in your board's six program spaces. Move NEXT as you run them.

SHOW

SET 0

STOP

STORE A

LOAD A

SET 2

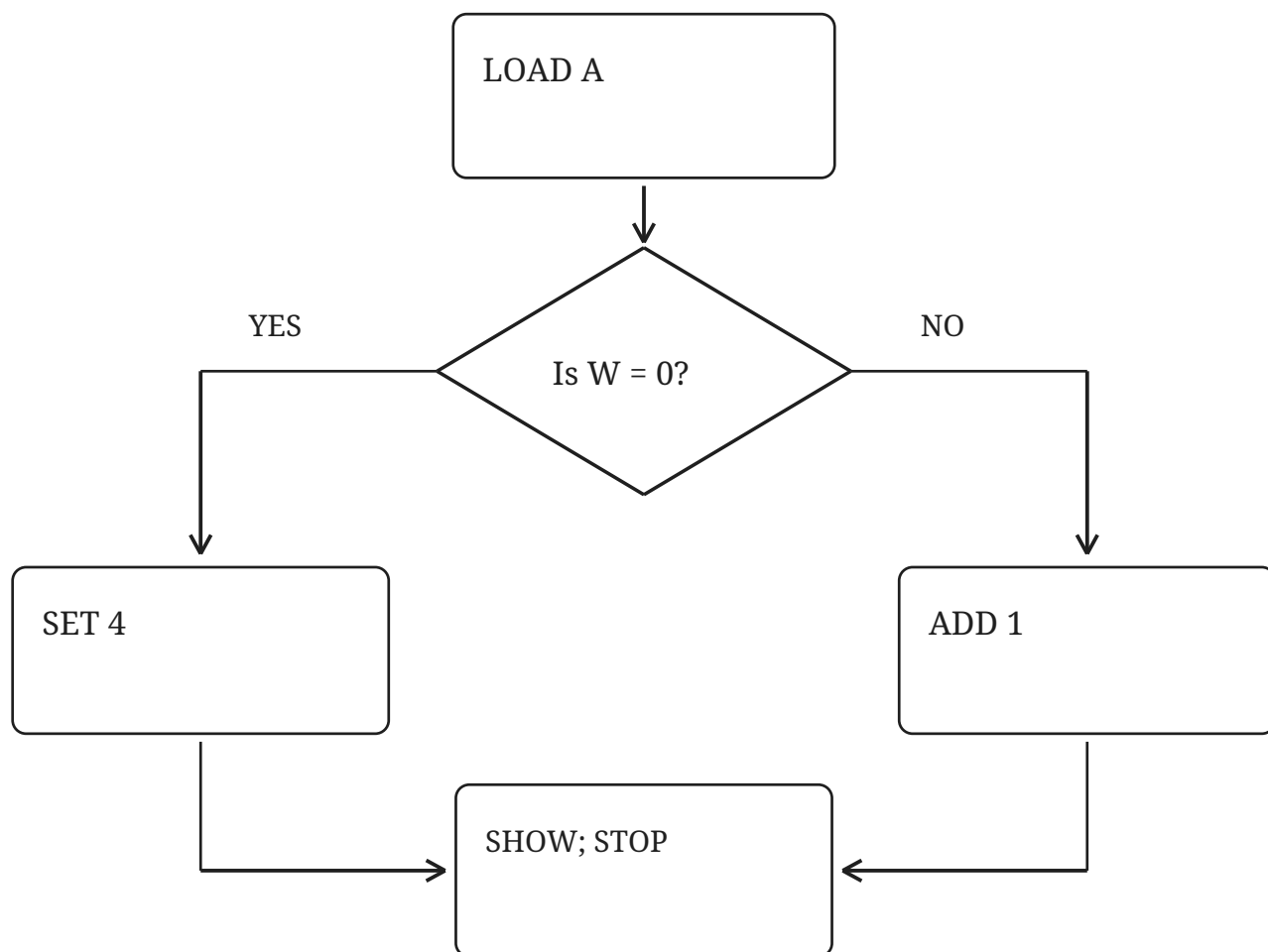
Goal: first set W to 2 and save it in A. Change W to 0. Bring the saved 2 back, show it, then stop. Start all slots at 0.

Program space	My instruction	W after
1		
2		
3		
4		
5		
6		

Program cards are smaller than the six board spaces. If you prefer not to cut, write the instructions in the spaces or point to the next step.

Follow a branch

A decision can choose the next instruction. Trace the arrows once for each starting value in A.



Starting A	YES or NO?	Output
0		
2		
5		

Follow only the chosen branch. The arrows are the program order, not wires carrying electricity. Check the condition after LOAD A.

Find the wrong memory slot

The engineer wants to show the saved 4. Run this buggy program from W=0, A=0, B=0, then change ONE instruction.

Step	Instruction	W	A	B
1	SET 4			
2	STORE A			
3	SET 1			
4	LOAD B			
5	SHOW			
6	STOP			

Before the repair, SHOW sends: _____

Replace instruction ____ with: _____

Run the fixed program again. A useful repair has a test: it must show 4 and leave A holding 4. Look carefully at the address named by LOAD.

Write a program with a goal

Make the output 8. Start with SET 2. Use exactly two ADD commands, each adding a number from 1 through 5.

Use six instructions in this order. Fill both ADD blanks. STORE A remembers the intermediate value. Then run your program and record the state.

Instruction	W after	A after
SET 2		
ADD ____		
STORE A		
ADD ____		
SHOW		
STOP		

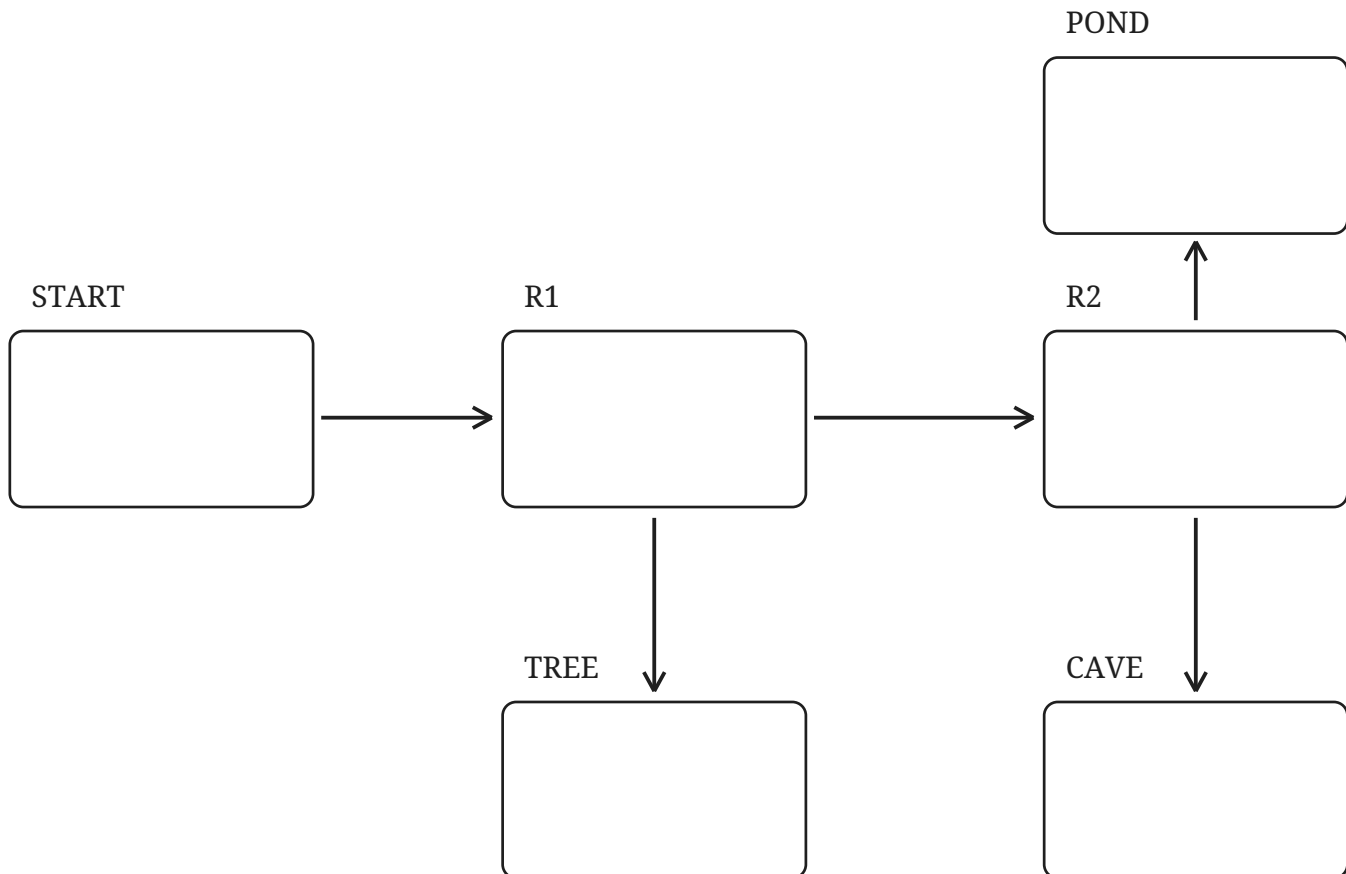
My two additions are ____ and _____. They add to _____.

Find a different pair that also works: ____ and _____.

Several programs can meet the same goal. Start a fresh run with W, A and B at 0. Keep the STORE command between your two additions.

Route an addressed packet

Cut the three small packet cards. Move each from START to its destination using the router rules below.

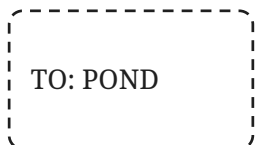


R1: TREE -> TREE. POND or CAVE -> R2.

R2: POND -> POND. CAVE -> CAVE.

A router chooses the next stop from the destination.

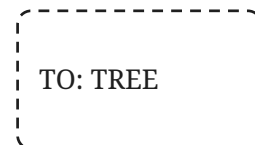
Record only the router names below, such as R1, R2.



Routers: _____



Routers: _____



Routers: _____

Our route map is invented and has fixed choices. Real packets carry data plus addressing/control information; networks use more complex rules and can change routes.

Rebuild the pond message

Packets arrived out of order. Keep only destination POND, message 7. Put parts 1, 2 and 3 in order.

TO: POND / MESSAGE 7

PART 3 OF 3

POND

TO: CAVE / MESSAGE 7

PART 2 OF 3

SLEEP

TO: POND / MESSAGE 7

PART 1 OF 3

MEET

TO: POND / MESSAGE 7

PART 2 OF 3

AT

Write the words in order:

1

2

3

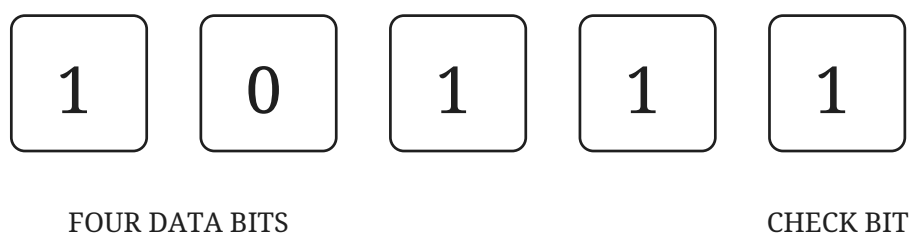
Which packet does not belong to this message destination?

Destination, message and part labels are a paper protocol we made for this activity. Real protocols can provide sequencing and reassembly in different ways.

Add a parity check

A parity bit is an extra check bit. Choose it so the TOTAL number of 1s, including the check bit, is even.

Even counts here are 0, 2 or 4. Example: data 1011 has three 1s. Add parity 1 to make four 1s in 10111.



Data	Count its 1s	Parity: 0 or 1	Total 1s
0000			
1010			
1110			
1000			

Zero is an even count. The check bit is separate from the four-bit data value. One parity bit can detect any single flipped bit, but it cannot locate that bit.

Can the check miss a change?

The sender made 10111: data 1011 plus check bit 1. Compare each received row with the sender.

SENT

1	0	1	1	1
---	---	---	---	---

Last box is the check bit.

A

1	0	1	1	1
---	---	---	---	---

Number of 1s: ____
EVEN / ODD: ____

Same as SENT? YES / NO

B

1	0	0	1	1
---	---	---	---	---

Number of 1s: ____
EVEN / ODD: ____

Same as SENT? YES / NO

C

0	1	1	1	1
---	---	---	---	---

Number of 1s: ____
EVEN / ODD: ____

Same as SENT? YES / NO

Which changed row still passes the EVEN check? _____

An odd number of flipped bits changes parity. Some changes, including two flipped bits, leave parity even. Passing this check does not prove the message is unchanged.

Open the paper-computer lab

Design a tiny machine that adds 1 to its input. Show an input, a program, an output and two tests.

DRAW YOUR MACHINE / label INPUT, CPU, MEMORY and OUTPUT

Use SET for the chosen input, ADD 1, SHOW and STOP. Your story can be a counter for axolotl visitors, books or imaginary supplies. Keep the input from 0 through 8.

Test input	SET command	Expected output	Observed output
0			
2			

One thing my paper model leaves out:

Explain how you checked your design. The expected outputs are goals; observed outputs come from following your actual instructions. Repair a mismatch and test again.

For the grown-up

Read aloud, share the roles, and give each paper rule time to become familiar.

Keep the parts working

Print single-sided in the matching paper size. Keep physical page 3 whole as the machine board. Cut number cards on page 4, folded bit tiles on page 5, instruction cards on page 13 and packet cards on pages 17-18. No glue is required. An adult helps with scissors.

A small instruction set

SET n replaces W with n . ADD n replaces W with $W+n$. STORE copies W into a named slot; LOAD copies a named slot into W . SHOW records W without changing it. STOP ends a run. Reset W , A and B to 0 between runs, unless a page gives another start.

Models have boundaries

The board separates program cards from two data slots for clarity. Real computers store instructions and data, use many registers and memory locations, and may overlap instructions. A paper instruction is not one clock cycle. Gates use labeled boxes, not wiring plans.

Choose a comfortable level

Ages 7-10 is an editorial starting point, not a tested learning outcome. Try building and bits first. Introduce one gate at a time. Branching, routing and parity can be later stretch visits. Accept pointing and spoken answers.

Look for reasoning

Ask which value changed, which stayed, and which rule explains it. For parity, compare a detected one-bit change with an undetected two-bit change. Sources support the concepts; this workbook still needs parent-child and home-printer playtesting.

All characters, programs, packet destinations and example records were authored for this paper lab. They are not measurements of a real computer.

Answers: activities 1-10

Check the rules together. Drawings can vary while the model behavior stays precise.

- 1 1 INPUT; 2 CPU; 3 MEMORY; 4 OUTPUT. Journey: INPUT -> CPU -> OUTPUT. Discuss how MEMORY holds information the CPU can use and update.
- 2-3 Cards should fit inside W and memory slots. Start all three at 0, using a blank for an extra 0. SHOW records a copy. Keep the board whole and retain the cut kit.
- 4 Starting 0: FLIP -> 1, FLIP -> 0, KEEP -> 0. The missing two-bit pattern is 11.
- 5 With columns 8,4,2,1: 0=0000; 3=0011; 6=0110; 9=1001; 12=1100; 15=1111.
- 6 NOT: 0 -> 1 and 1 -> 0. Two-gate chain: 1 -> 0 -> 1. Two NOTs restore the original bit.
- 7 AND outputs for input rows 00,01,10,11: 0,0,0,1. Only the last row turns the welcome light on.
- 8 OR outputs: 0,1,1,1. AND: 0,0,0,1. The rows 01 and 10 differ; OR accepts either flag alone.
- 9 For A,B rows 00,01,10,11: NOT B is 1,0,1,0. Final A AND NOT B is 0,0,1,0. A=1,B=0 is the only allowed turn.
- 10 After each command, W/A is 3/0, 3/3, 1/3, 3/3, 3/3. No output appears until SHOW, which records 3. LOAD leaves A holding 3.

Explain a result by pointing to the input or instruction that caused it. Treat decorative colors as choices, not as encoded values unless a page supplies a rule.

More answers and sources

Activities 11-20. These original exercises use simplified, explicitly stated rules.

- 11 W/A after steps: 2/0, 5/0, 5/5, 6/5, 6/5, 6/5. Output first becomes 6 at SHOW and stays 6 after STOP. B stays 0.
- 12-13 12: SET 2, STORE A, SET 0, LOAD A, SHOW, STOP. W: 2,2,0,2,2,2; A: 0,2,2,2,2,2; output 2. 13: A=0 -> YES -> 4; A=2 -> NO -> 3; A=5 -> NO -> 6.
- 14-15 14: buggy W is 4,4,1,0,0,0; A is 0,4,4,4,4,4; B stays 0. Replace line 4 with LOAD A; output 4. 15: ADD pairs 1+5,2+4,3+3,4+2,5+1. Saved A is 2 plus the first ADD; output 8.
- 16-17 16: POND: START-R1-R2-POND; CAVE: START-R1-R2-CAVE; TREE: START-R1-TREE. 17: MEET AT POND. Exclude the CAVE packet even though its part number is 2.
- 18-19 18: data counts 0,2,3,1; parity 0,0,1,1; totals 0,2,4,2. 19: A has 4 ones, even, unchanged; B has 3, odd, changed; C has 4, even, changed. C passes despite two flipped data bits.
- 20 SET 0, ADD 1, SHOW, STOP gives 1. SET 2, ADD 1, SHOW, STOP gives 3. Check labeled parts, both traces and an honest model limit. Other decoration and stories can vary.

CONCEPT REFERENCES / checked October 2026

IEEE: technav.ieee.org/topic/computers/

CS Unplugged: csunplugged.org/en/topics/binary-numbers/

Yale CS 201: zoo.cs.yale.edu/classes/cs201/spring_2025/lectures/Gates.html

Cloudflare: cloudflare.com/learning/network-layer/what-is-a-packet/

CS Unplugged: csunplugged.org/en/topics/error-detection-and-correction/

References support computing concepts, not a learning-outcome claim or endorsement of MomoMango. The source module also records complete reference URLs.