Building a Modern CPU with Program Counter by TT

Dec. 2021 (Rev. 1)
Yama-chan

Abstract

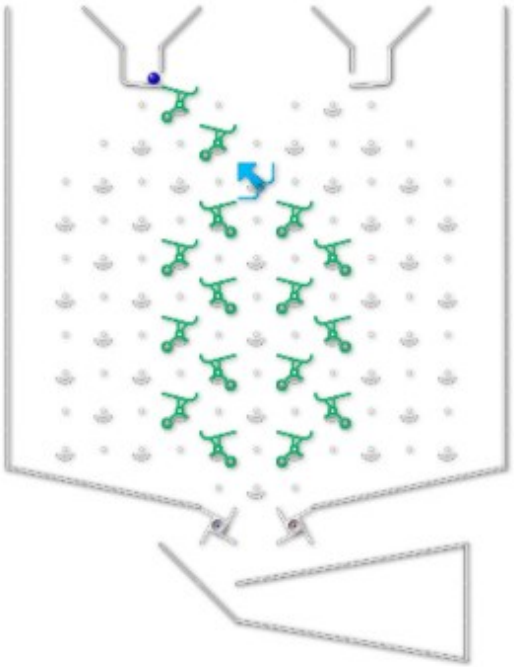
There is a pachinko-like game called Turing Tumble (TT, https://www.turingtumble.com). It has the potential to perform complex calculations like an electronic circuit, because the components can be arranged freely. This book describes a step-by-step method to build a modern CPU with a program counter in Turing tumble.

Contents

1. Memory

Consider a device consisting of a single bit and some ramps connecting it, as shown below. The direction of the bit can be set by hand to any direction you like. And the direction of the bit will determine which path the ball will follow and which lever it will press. In other words, the bit can MEMORY information about which way to go.

Since this configuration destroys the data after it was read, I will introduce a memory that retains the data as it is read in later chapters. Note that the property of holding the orientation is the same even if the bits of the circuit are changed to gear bits.



In the following, https://jessecrossen.github.io/ttsim is used for the simulation of TT.

When digitizing data, the leftward "↖" is set to 0 and the rightward "↗" is set to 1, just like an analog meter or selector switch.

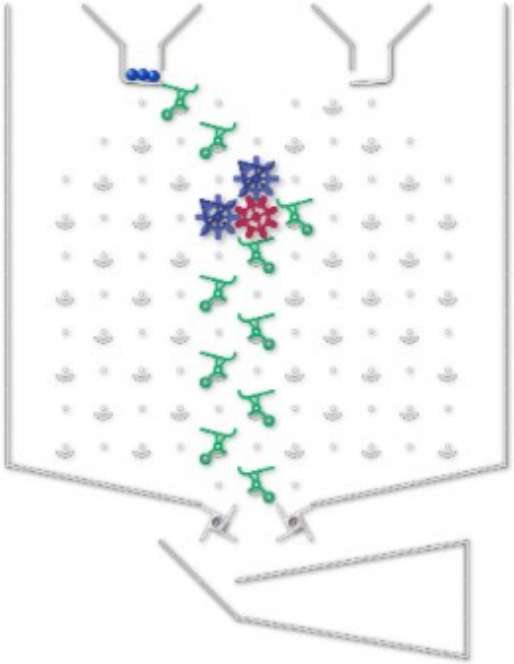
Key words for further learning

Magnetic-core memory

2-1. Latch

When you use two gear bits and a gear as shown below, the original ↖ state will only pass through once and change direction. However, if the direction before the ball passes is ↗, the gear bit immediately below it will also pass. In other words, the entire gear bit chain will have passed through twice, and the direction of the bit will return to the original ↗.

The way the direction of the bit does not change once it is turned to the right is similar to the way a door cannot be opened without twisting the handle once it is closed by a triangular part called a LATCH.

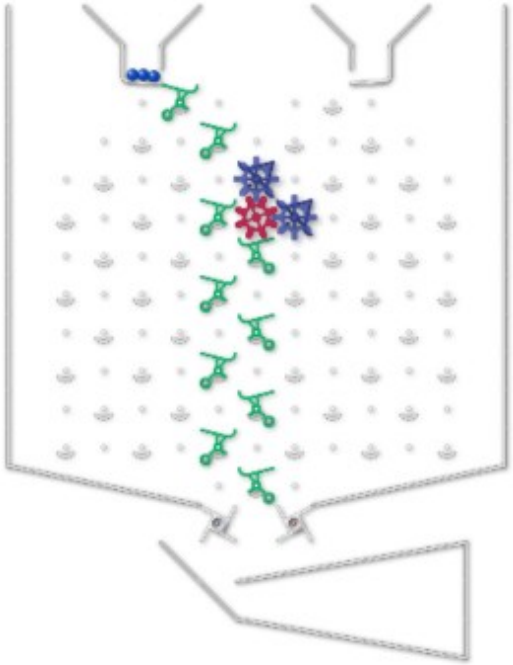


This change in state can be summarized in a table as follows. Before the ball passes through (IN), there are two possible states, ↖ and ↗. After the ball passes through (OUT), both states are ↗.

in	out
↖ = 0	↗ = 1
↗ = 1	↗ = 1

2-2. Reset

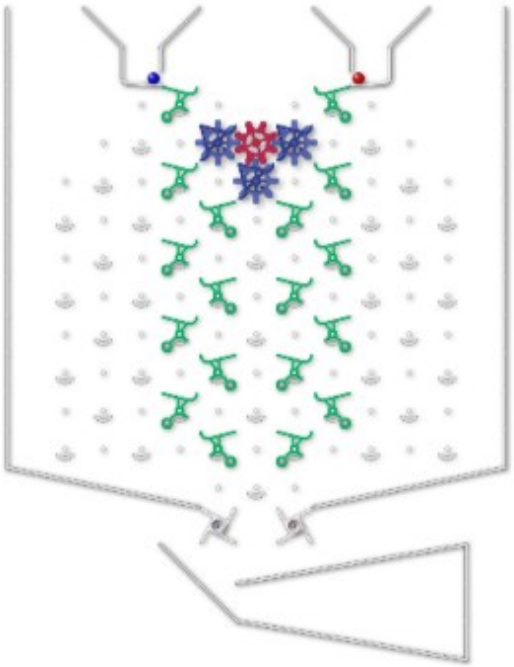
Unlike the previous example, the gear bit will always be ↖(0) after ball passing through. The table of state changes is shown below. This means that the latch was applied in the opposite direction to 2-1. Then, the data that becomes 1 after the ball is passed is called set, and the data that becomes 0 like this is called reset or clear.



in	out
↖ = 0	↖ = 0
↗ = 1	↖ = 0

2-3. Reset-Set pattern

The following is a combination of reset and set. Which of the levers a person presses first will determine whether it is reset to ↖ or set to ↗.



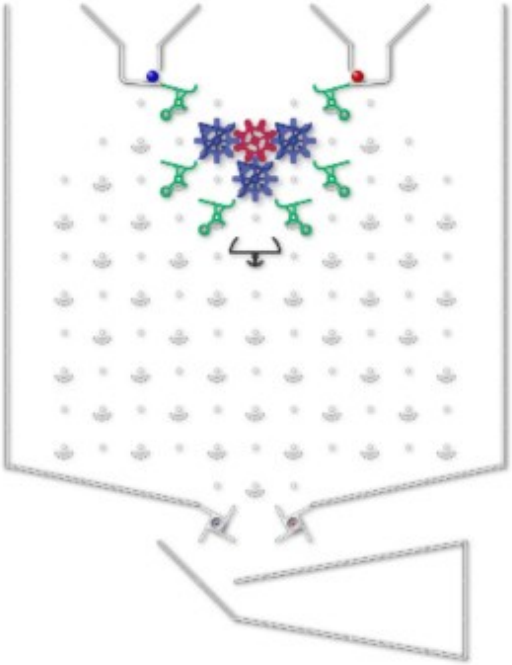
in		out
● ○	↖ = 0	↖ = 0
● ○	↗ = 1	↖ = 0
○ ●	↖ = 0	↗ = 1
○ ●	↗ = 1	↗ = 1

Key words for further learning

Flip-flop (SR latch)

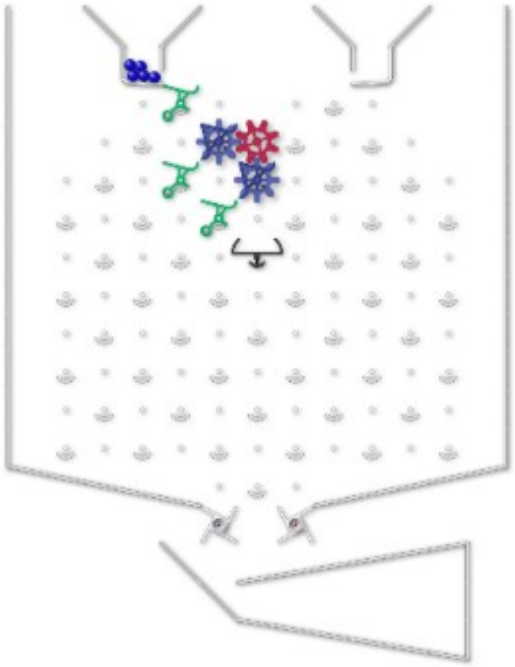
3-1. HALT

The next pattern is started by pressing one of the levers, but after the gear bit is rewritten, the ball is held by the intercept component. This means that the series of operations can be stopped from this parts.



3-2. Interrupt

This pattern is a reset, but it stops working after the gear bit is set to ↖. So, if we set the gear bit to ↗ by hand and press the lever, we can confirm the operation from ↗ to ↖ again. In this way, the operation may be INTERRUPTED to check the status or to change it from outside.

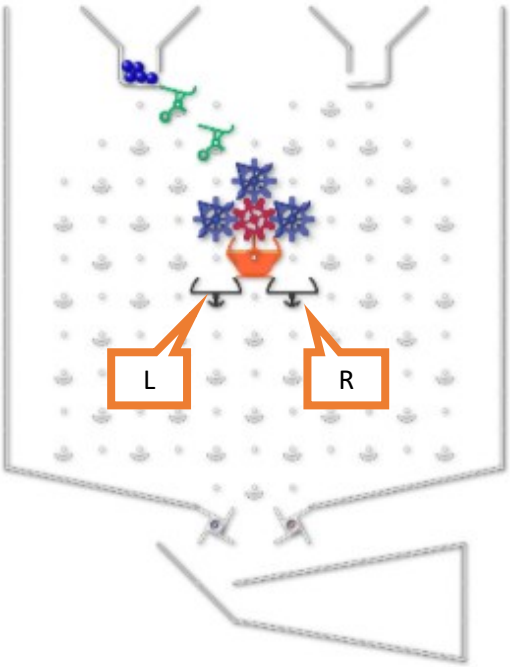


Key words for further learning

waiting for input

4-1. Non-destructive read

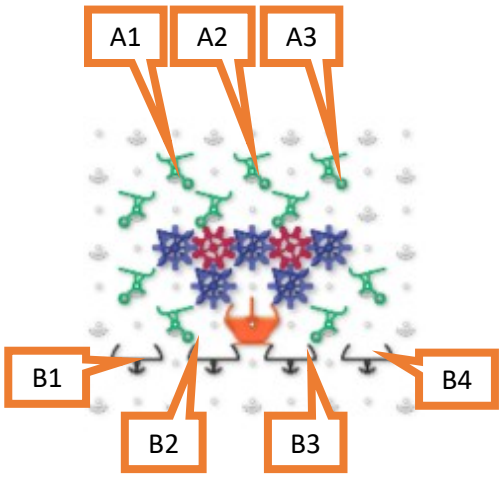
Arrange the gear bits as shown below. The direction of the bit may be set by hand in any direction you like. After the ball passes through the gear bit, it will be held in either L or R position depending on the direction of the bit. Also, unlike in the past, the direction of the gear bit does not change after the ball passes through it. This means that reading is done without destroying the information.



in	out	
↖	↖	L
↗	↗	R

4-2. RAM

The following is a combination of set, reset and non-destructive reading. Depending on which path the ball is inserted into, the read/write function can be selected, and the system can operate like a computer RAM.



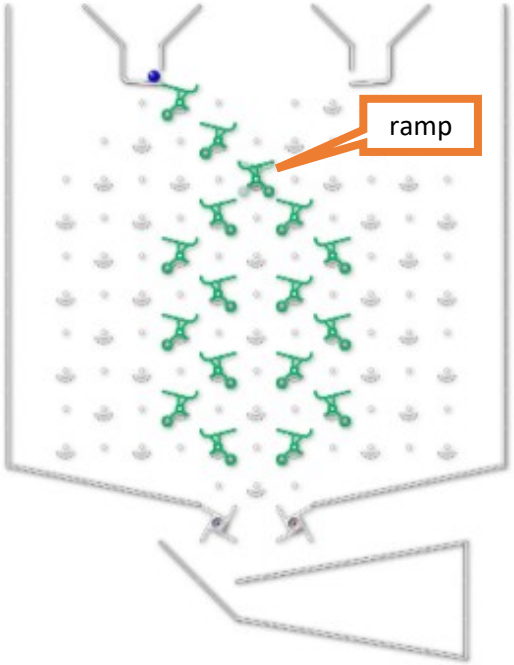
in		out		note
A1	↖	↖	B1	Clear
A1	↗	↖	B1	
A2	↖	↖	B2	Read
A2	↗	↗	B3	
A3	↖	↗	B4	Set
A3	↗	↗	B4	

Key words for further learning

SRAM、 DRAM

4-3. ROM

The following is a pattern in which the bits of the 1. pattern are changed to lamps, which can lead the ball to the left or right depending on the direction of the lamps. This kind of READ-ONLY MEMORY, which cannot be changed by the ball, has the advantage of being smaller than RAM. It can also be used to store important settings because it cannot be rewritten unexpectedly.

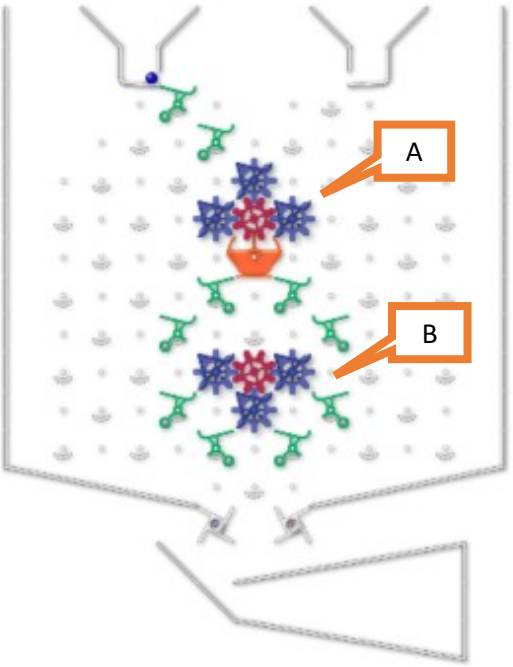


Key words for further learning

Mask ROM, Fuse PROM, UV-EPROM, EEPROM

5-1. Copy

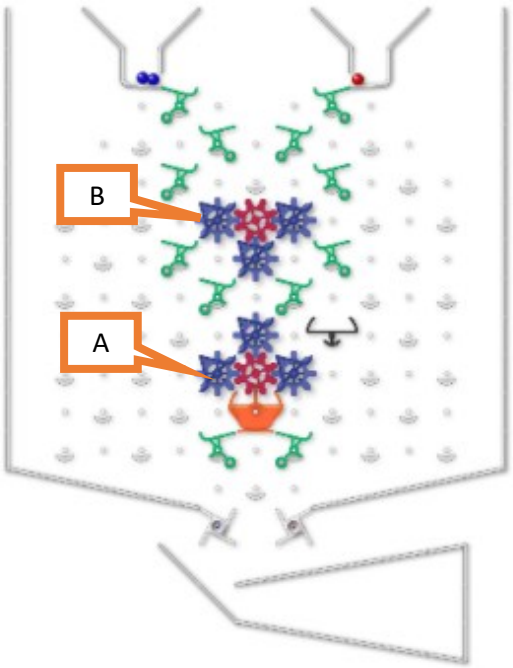
In this configuration, there are two memories from the top, A and B. By reading the memory in A and writing to the memory in B according to its value, the memory in the top can be copied to the memory in the bottom.



in		out	
A	B	A	B
↖	↖	↖	↖
↖	↗	↖	↖
↗	↖	↗	↗
↗	↗	↗	↗

5-2. Copy2

Unlike the previous example, the lower memory is A and the upper memory is B. After the first ball passes through memory A, its information determines which lever to press, i.e. the color of the second ball. Memory B's value is ultimately determined by the color of this second ball. This information flows upward as well, allowing the computer to determine the next value based on the current state.



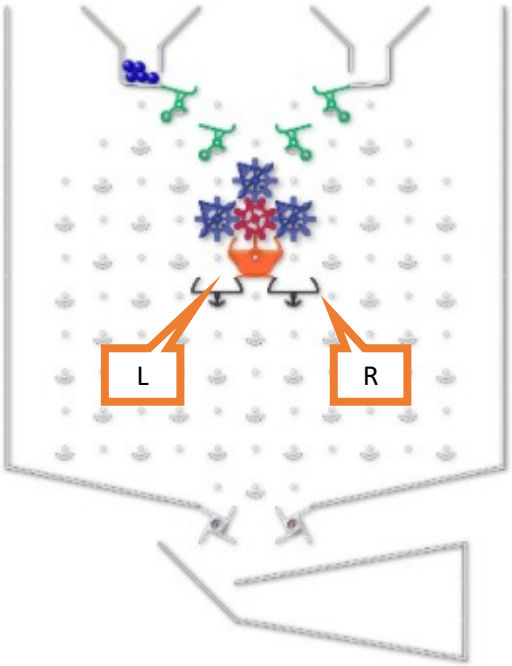
in		2nd	out	
B	A	ball	B	A
↖	↖	● ○	↖	↖
↗	↖	● ○	↖	↖
↖	↗	○ ●	↗	↗
↗	↗	○ ●	↗	↗

Key words for further learning

Feedback

6-1. IF

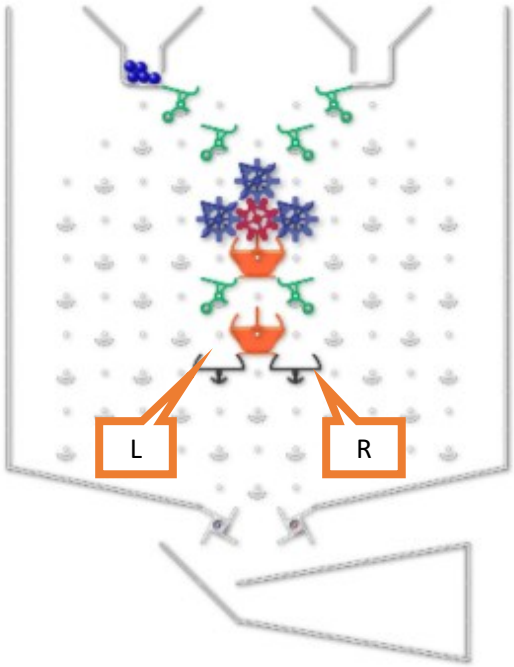
At the bottom, the value of RAM determines which intercept the ball will fit into. In other words, IF right or IF left, a conditional branch is made.



in	out
↖	L
↗	R

6-2. NOT

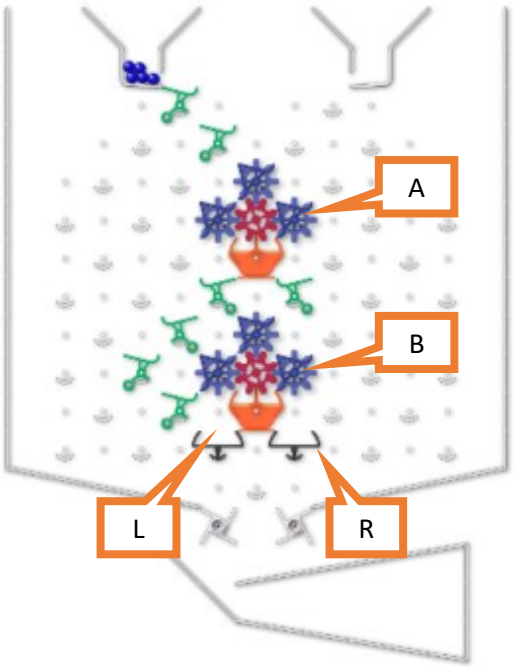
In this example, unlike the previous one, the ball goes to the right, NOT the left if it is ↖, and to the left, NOT the right if it is ↗.



in	out
↖	R
↗	L

6-3. AND

This has two memories, but the ball will only go into the right intercept if memory A AND memory B both have ↗(1).



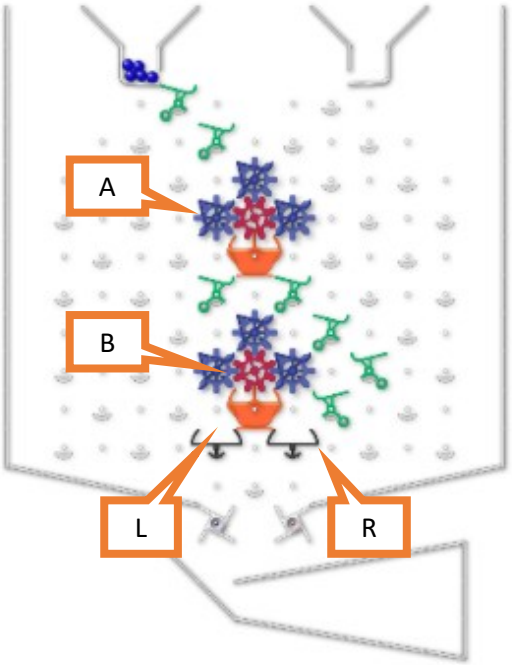
in		out
A	B	
↖	↖	L
↖	↗	L
↗	↖	L
↗	↗	R

Key words for further learning

Random logic

6-4. OR

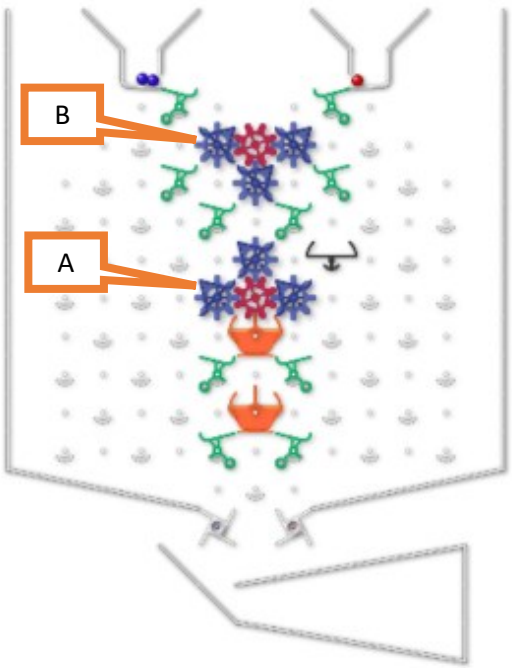
Now, when either Memory A OR Memory B is ↗(1), the ball goes into the right intercept.



in		out
A	B	
↖	↖	L
↖	↗	R
↗	↖	R
↗	↗	R

7-1. NOT2

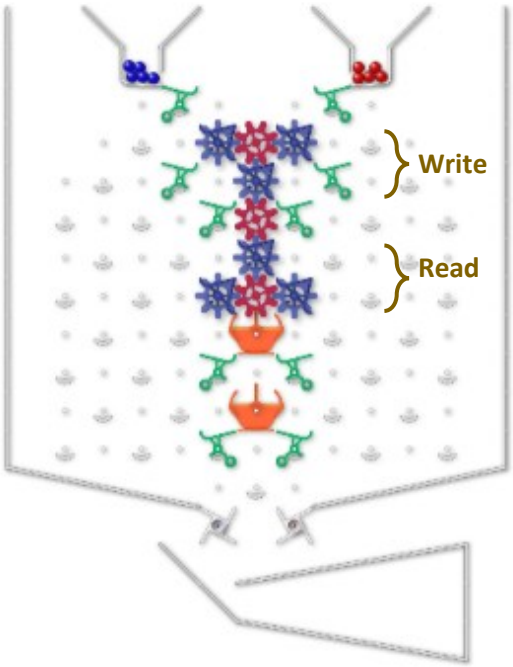
This pattern is similar to 5-2. but the route is such that it becomes NOT after reading A. Therefore, B will contain NOT(A).



in		2nd	out	
B	A	ball	B	A
↖	↖	○ ●	↗	↖
↗	↖	○ ●	↗	↖
↖	↗	● ○	↖	↗
↗	↗	● ○	↖	↗

7-2. T Flip-flop

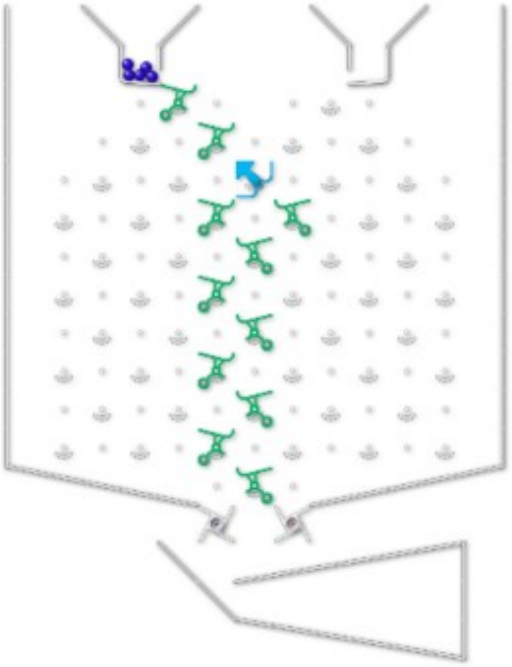
The following example shows the connection of memory-A and memory-B in 7-1. In such an arrangement, the state of the next memory is determined by the state of the previous one. As shown in the table, there are four possible patterns for the ball and bit, but only two patterns are obtained after the operation. As a result, an operation in which the state changes alternately like a TOGGLE switch is realized.



in		out	
ball	bit	bit	ball
● ○	↖	↖	○ ●
● ○	↗	↖	○ ●
○ ●	↖	↗	● ○
○ ●	↗	↗	● ○

7-3. T Flip-flop2

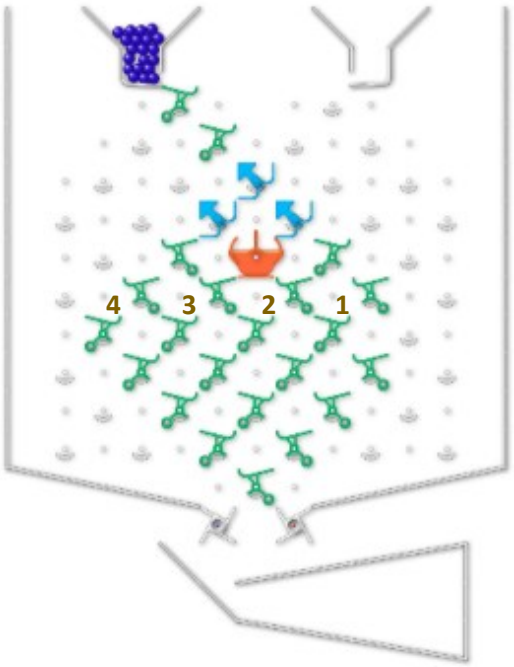
The previous result alternates between left and right if you look at the bits. In fact, this is the same as the behavior with only one bit. So, the same behavior can be achieved by simply arranging them as follows. Also, unlike 7-2, this pattern is not affected by the color of the ball.



in	out
↖	↗
↗	↖

7-4. Phase

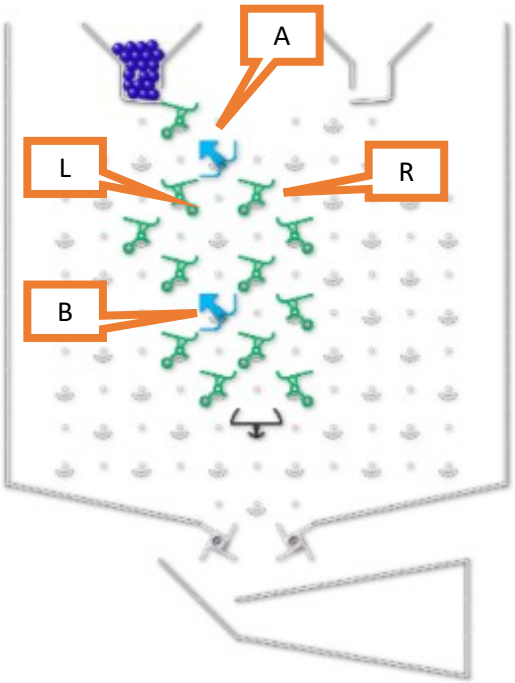
If three bits are placed as follows, each time the ball passes through, the direction of the bit changes, thereby splitting the route in four ways. This will always return to first state after four times, just as the PHASES of the moon repeat regularly. This kind of pattern can be used to perform repetitive operations.



in	out	
bit	ball	bit
↖ ↖	1	↗ ↗
↗ ↖	2	↖ ↗
↖ ↗	3	↗ ↖
↗ ↗	4	↖ ↖

8-1. Adder

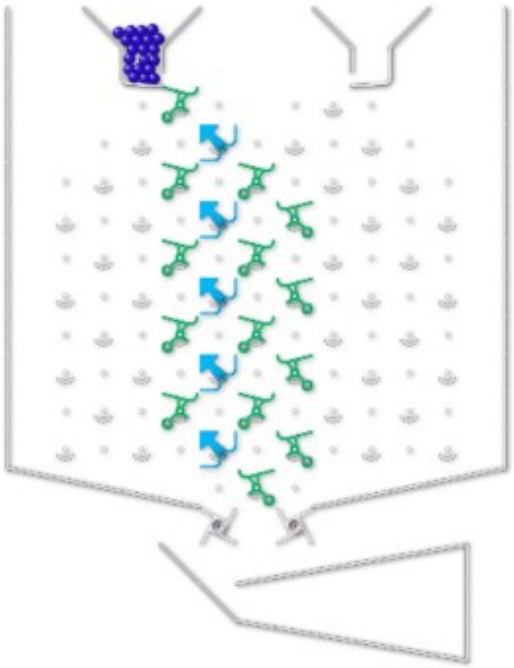
In the binary system, the numbers increase from 00b, 01b, 10b to 11b. We will reproduce this. Memory A indicates the first digit. A ball passing through it means that one is added. Memory A always changes as the ball passes. In addition, when it changes from ↗(=1) to ↖(=0), there is an advance. On the board, this happens when you take the left route. In this case, the memory B located below is changed.



in		ball	out	
B	A		B	A
↖	↖	R	↖	↗
↖	↗	L	↗	↖
↗	↖	R	↗	↗
↗	↗	L	↖	↖

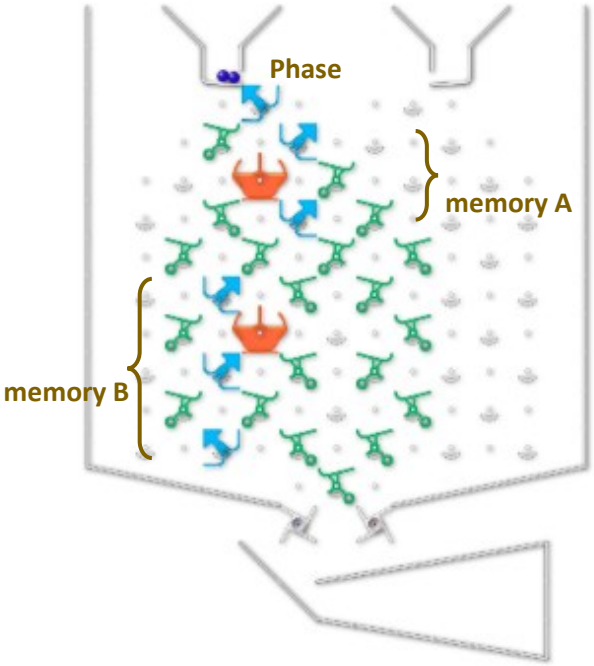
8-2. Counter

The number to count can be easily increased. In this example, we can count up to 2×2×2×2×2=32. However, on this board, the ball will be lost before that.



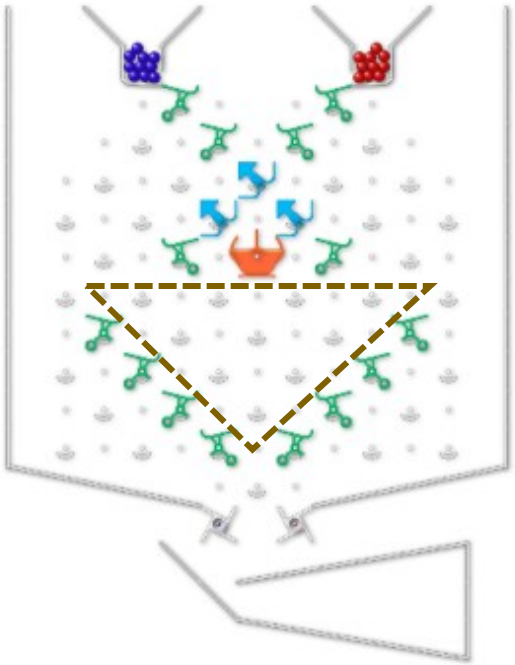
8-3. Calculator

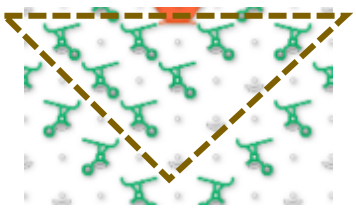
The following example shows the calculation of 011b+11b=110b. The calculation is done in the order of the first digit, then the second digit, which is controlled by the phase bit. Memory B is a counter, and if the first digit read from the upper side of memory A is ↗(=1), ball will be led to memory B. Memory A is destroyed after reading. And the second digit is read from the lower side of the second digit of memory A, as before. This leads to the second digit of memory B, which is +10b.

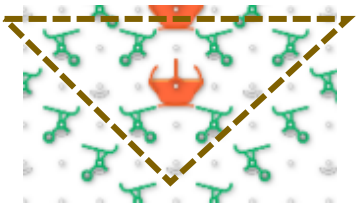


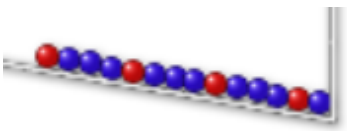
9-1. Program

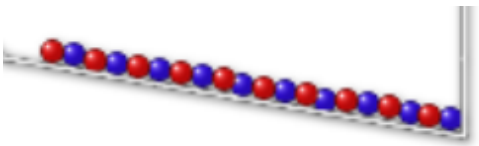
Make a pattern like the one below. It doesn't work as it is, but you can PROGRAM various patterns by placing parts in the 10 triangle-shaped holes.

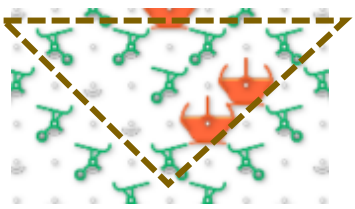


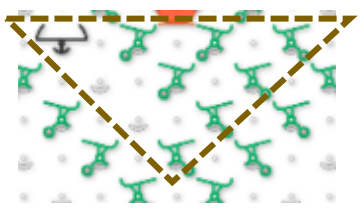


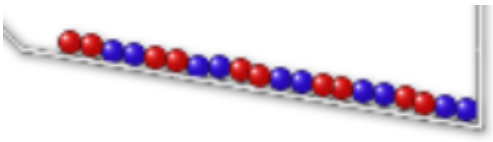


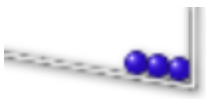






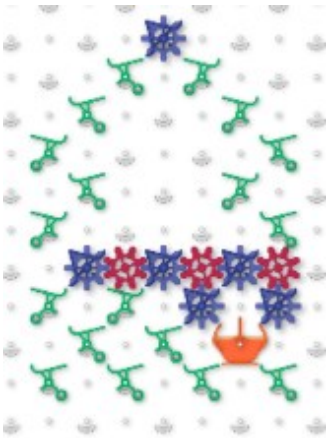






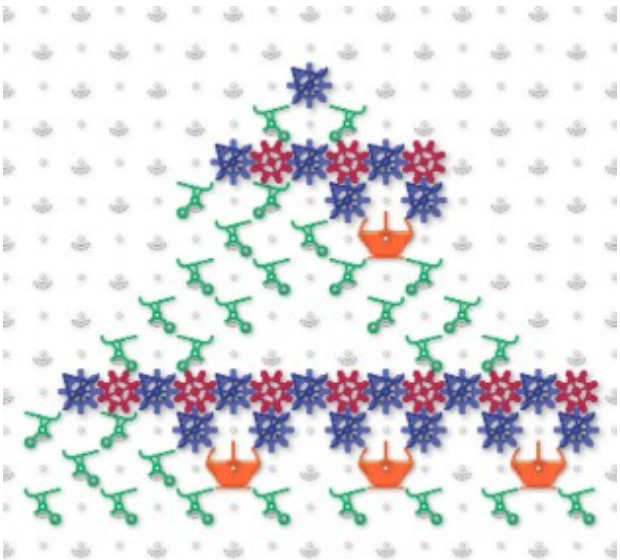
9-2. Program2

As in the previous example, this is also divided into four strips depending on the phase. In other words, it is programmable, but it is similar to the adder configuration of 8-1. 9-1 required three bits, but this configuration has two sets of gear bits. The configuration from here on is difficult to reproduce on a real board because there are too many parts, so it is better to use a simulator.



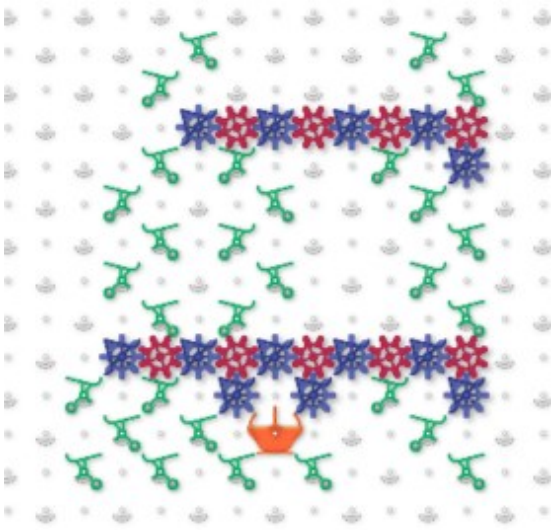
9-3. Program3

If you want to add more phases, you can add a gear bit chain like this.



9-4. JMP 0

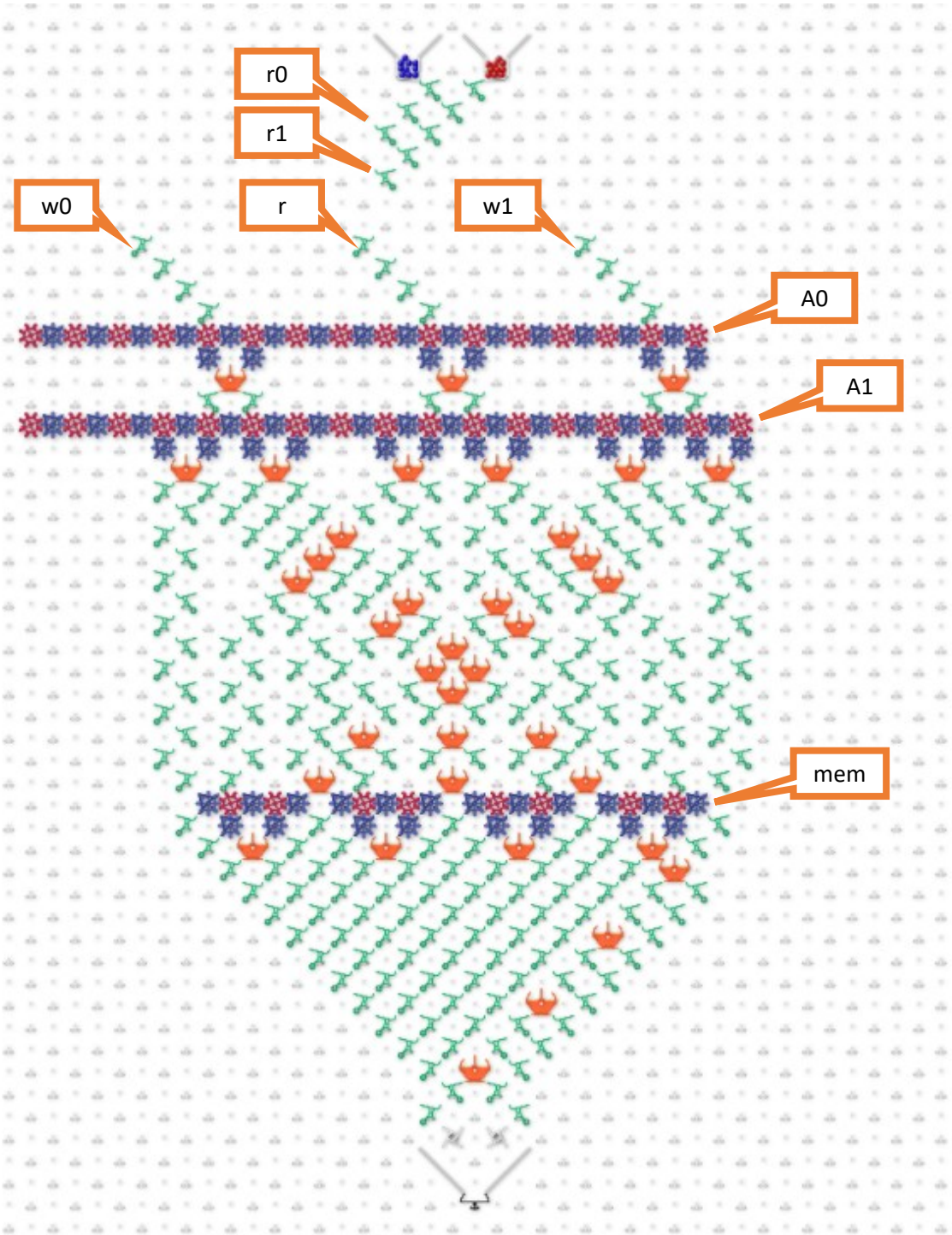
This is 9-2 with an additional route for clearing. The left part is the same as before, but a route for clearing is created on the right side of each gear bit chain. By assigning the red ball to the clear route, for example, you can return to the top of the program at any time you like.



10-1. RAM2

We will show how to use another TT board to configure RAM, an important component of the CPU, which can RANDOM (irregular order) ACCESS to MEMORIES. The configuration is as shown in the figure. In this case, r0, r1 and w0, w1, r are not connected. They are accessed from the main CPU board, which is configured on a separate TT board.

Set up A0 and A1, which are connected to the main board by gear bits, in advance to determine which memory to access. The main board contacts w0, w1, or r using a lamp. In response, the memory in the RAM board is rewritten or read. The result is obtained by the ball coming out of r0 and r1.



in		out	
ball	mem	ball	mem
w0	X	r0	↖
w1	X	r0	↗
r	↖	r0	↖
	↗	r1	↗

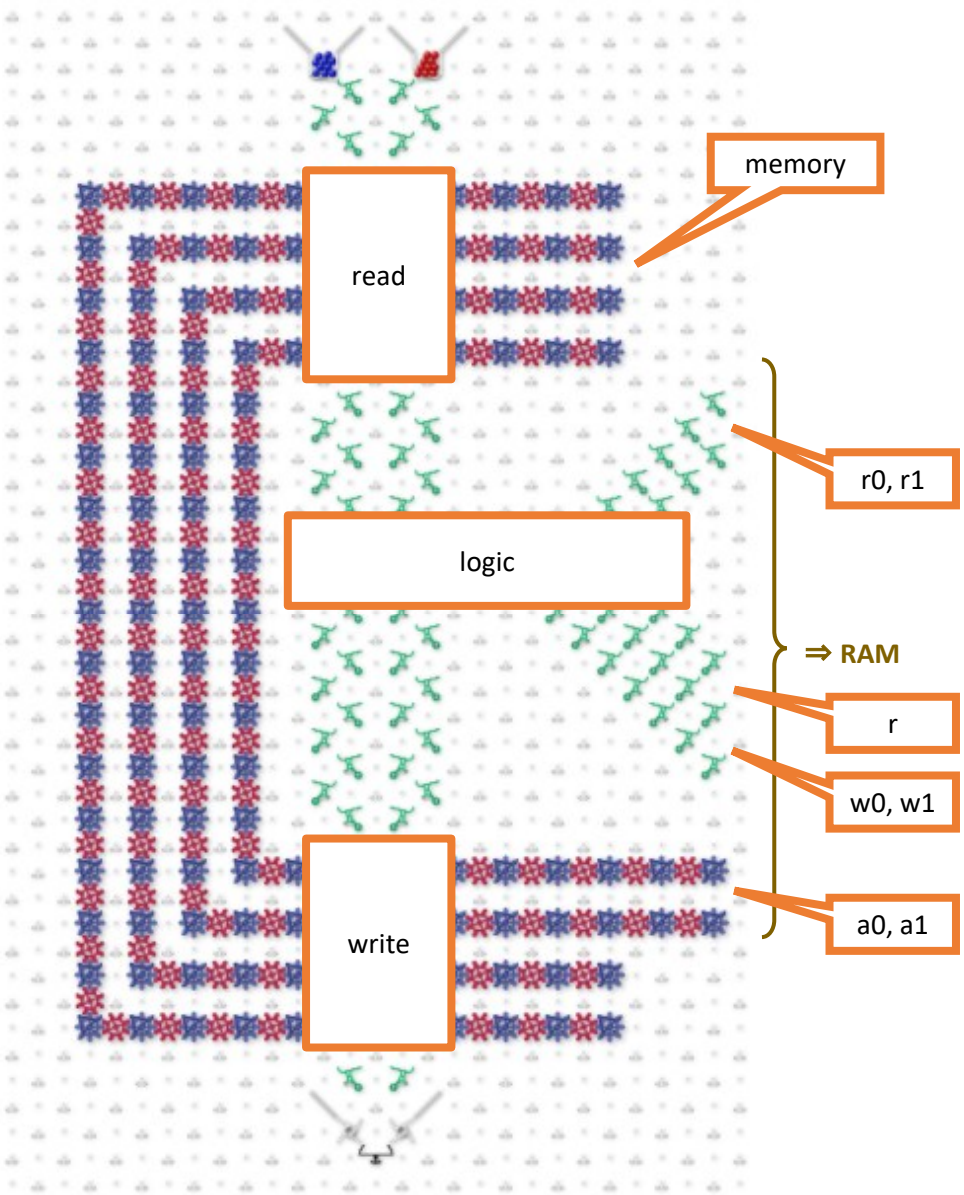
X = ↖ or ↗

Key words for further learning

Sequential access

10-2. Building CPU

We aim to build a CPU using a gear bit chain nested in a "C" shape as shown below.



The CPU

reads instructions,

reads data,

performs operations,

and writes data

in a single execution. This can be realized as a regularly repeating operation by using a phase like 7-4.

In order to realize the CPU, a gear bit chain that corresponds to the program counter, address registers, instruction registers, data registers, and flag registers is prepared as memory. The C-shaped RAM address bus is also nested here.
Since the gear bit chain is C-nested,

it is read when passing through the upper stage,

the results are branched according to the path, and further operations are performed as necessary at the bifurcation

and written to memory at the lower stage.

In other words, at each intersection of the horizontal gear bit chain and the vertical line ramps, write as in 2-3 or read as in 4-1. If the gear bit chain and line ramps only intersect without data processing, use the read pattern.

A closer look at the procedure of CPU instruction execution. Reading an instruction or data requires two phases. First, the value of the program counter or address register is copied to the address bus, which is a gear bit chain physically connected to the memory. In the next phase, a ball is played on the line ramp r and the result is written to the instruction register or data register. At this time, if the program counter was configured as shown in 9-2, the +1 operation can be performed.

In the operating phase, an arithmetic device such as the one shown in 8-3 is created in the logic section in advance, and the data register is updated by the ball. Here, the instruction register determines which route to take, i.e., which operation to perform. In addition, the flag register is also updated depending on the situation such as overflow of digits.

When writing data, the data register and address bus are set in advance. The ball that reads the data register passes through w0 or w1 and writes data to the RAM.

Repeating the above, the CPU performs calculations and data processing.

Other CPU instructions include flag branching, jumping, and input/output. All of these can be realized by rewriting the relevant registers such as instruction registers and program counters in the intermediate phases.

Key words for further learning

Stored-program computer

10-3. Differences from electric computers

In electric circuits, the nature of transistors allows for high-speed operations.

Key words for further learning

DTL、TTL、CMOS

While electric circuits are vulnerable to electromagnetic noises, TT is vulnerable to vibrations.

Key words for further learning

Digital electronics

Electric circuits can be easily branched and parallelized because there are countless electrons in the circuit.

Key words for further learning

Fan-out

Since TT has one ball, it is necessary to place circuits in series or use phases well for branching process. If we increase the number of balls in TT, we need a new mechanism to control the timing skillfully.

Key words for further learning

Multiplexer, Demultiplexer

Particle physics says that gravity is far weaker than the electromagnetic force. Therefore, the effect of reducing the mass of the marble is greater when the TT components are made smaller, and unlike semiconductor electric circuits, speed improvement cannot be expected.

Key words for further learning

Die shrink, Fundamental interaction

Of course, TT is not affected by the global shortage of semiconductors.

Appendix A. Developmental References

For further understanding, some references are given at the end.

www.nandgame.com

An electronic circuit building game that runs in your browser, where you can build your first CPU from NAND, which NOT and AND logic.

Minecraft, Mario Maker

These games allow you to arrange items in any way you like. You can see some videos that challenges to recreate a computer by placing components that work like switches as needed.

MCPU, Minimal 8Bit CPU
https://lniv.fe.uni-lj.si/courses/iv/mcpu.pdf

The simplest CPU known to the author, with only four instructions but with sufficient functionality.

intel 8080、MCS80

It is an early CPU, and the materials are available for easy study.

Ladder logic, Relay logic

It is intuitive to understand because the switches are controlled by coils, and logic such as AND and OR can be built, and NOT can be handled by NC contacts. It can also hold a value by setting up a latch circuit, which was used in early computers such as the Zuse Z3 and the Harvard Mark I.

Analytical Engine by Charles Babbage

More than 100 years ago, an attempt was made to create a computer by machine.

Turing machine

The author became interested in the Turing Tumble because of its claim that "the Turing Tumble has Turing completeness." Mechanisms that satisfy Turing completeness, such as Turing machines, are said to be capable of any computation.

Appendix B. Turing Tumble Cards

Bonus Parts

Let's cut it out and figure out a pattern.