sourcemind

Computer Science and Software Engineering Fundamentals

Hovak Abramian

Introduction

About the instructor

- Hovak
- Wrote the first program in 1996.
- Ran it for the first time in 1998.
- Currently teaching in American University of Armenia

Course Outline

- Basic concepts
- Architecture
- Software Development Lifecycle

Session I

Outline

We are going to learn about:

- Number representations
- Boolean Algebra
- Hardware Components
- Processor Architecture

Learning Objectives

At the end of the session, you will be able to:

- Convert bases
- Identify hardware components
- Demonstrate the relationship between hardware and numbers.

What do these symbols mean?

1000

Numeral Systems

- Roman

XVI = 16

- Armenian

Ա = 1, Ժ = 10, ՌՋՀԵ = 1975

- Duodecimal System
- Hindu-Arabic Numeral System

Hindu-Arabic Numeral System

- The decimal notation system we use in our everyday lives.
- Originally from India, it was brought to Europe by al-Khwarazmi
- It has a base: The base for decimal is ten.
- It is a positional system: the position of a digit indicates the power of the base.

Muhammad ibn Musa al-Khwarizmi (c. 780-c. 850)

M. Tomczak *Statue of al-Khwarizmi in front of the Faculty of Mathematics of Amirkabir University of Technology in Tehran, IRAN*, 2013. Accessed Aug 28, 2019 from: http://commons.wikimedia.org/

Decimal (Base-10)

534

Five hundred and thirty four

$$534 = 5\times100 + 3\times10 + 4\times1$$
$$= 5\times10^2 + 3\times10^1 + 4\times10^0$$

Decimal (Base-10)

534

Five hundred and thirty four

Digits are coefficients

$$
\begin{aligned}
534 &= 5\times100 + 3\times10 + 4\times1 \\
&= 5\times10^2 + 3\times10^1 + 4\times10^0
\end{aligned}
$$

10 is the **base** or **radix**

The powers are correlated with the positions

Decimal (Base-10)

$$534_{10}$$

The base or radix is 10

Means the number is represented by powers of 10 and their coefficients

Binary (Base-2)

$$1101_{10} = 1\times1000 + 1\times100 + 0\times10 + 1\times1$$
$$= 1000 + 100 + 1$$

Implied radix is ten,
is often omitted.

$$1101_2 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$$
$$= 1\times2^0 + 0\times2^1 + 1\times2^2 + 1\times2^3$$
$$= 8 + 4 + 0 + 1 = 13$$

When calculating manually, it is more convenient to start with the least significant digit on the right and increment the power of the base for the next term.

sourcemind

Binary - Examples

There are only two digits: 0 and 1.

Leading 0's do not affect the number, hence a number always starts with 1.

Notice the incremental pattern.

$0_2 = 0$

$1_2 = 1$

$10_2 = 2$

$11_2 = 3$

$100_2 = 4$

$101_2 = 5$

$110_2 = 6$

$111_2 = 7$

$1000_2 = 8$

$1001_2 = 9$

$1010_2 = 10$

$1011_2 = 11$

Binary - Powers of Two

It useful to know powers of two up to 12 by heart.

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$
$2^9 = 512$
$2^{10} = 1024$
$2^{11} = 2048$
$2^{12} = 4096$

Binary - Addition

$$
\begin{array}{r}
\color{blue}{1} \\
1\,0\,1\,0\,1 \\
+\quad 1\,1\,0 \\
\hline
1\,1\,0\,1\,1
\end{array}
$$

$$
\begin{array}{r}
\color{blue}{1\,1\,1} \\
1\,0\,1\,0\,1 \\
+\quad 1\,1\,1 \\
\hline
1\,1\,1\,0\,0
\end{array}
$$

Binary addition works similar to decimal addition.

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 10 and when encountered, the 1 is carried over to the next column

sourcemind

Binary - Multiplication

$$
\begin{array}{r}
1\,1\,0\,1 \\
\times \qquad 1\,0\,1 \\
\hline
1\,1\,0\,1 \\
1\,1\,0\,1\,0\,0 \\
\hline
1\,0\,0\,0\,0\,0\,1
\end{array}
$$

Multiplication also works exactly like that of decimal systems.

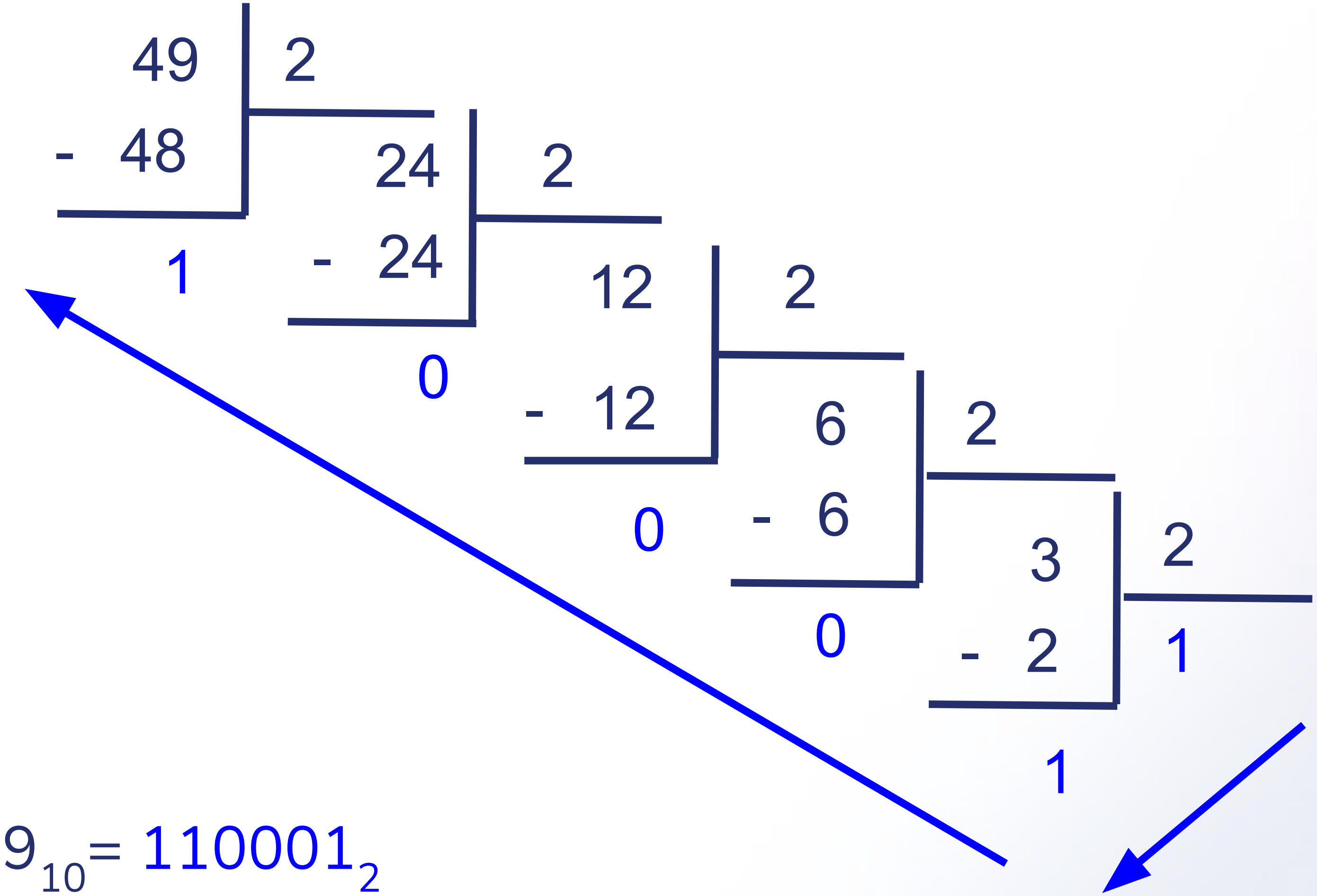
Anything multiplied by 1 is itself.

0 in the second operand results in a shift in the position.

Adding a 0 to the end of a binary number multiplies it by two, the same way that adding 0 to the end of a decimal number multiplies it by ten.

e.g. 6 and 60

Binary - Conversion from Decimal



$49_{10} = 110001_2$

In order to convert from decimal to binary (or **any** base), divide the number by two (target base) consecutively.

Continue until the quotient is less than the base.

Start from the right and take the quotient and the remainders in that order.

Note how numbers always start with a 1

Hexadecimal (Base-16)

The digits are:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$$534_{16} = 5\times16^2 + 3\times16^1 + 4\times16^0$$
$$= 5\times256 + 3\times16 + 4\times1$$
$$= 1280 + 48 + 4$$
$$= 1332$$

The base is 16 so we proceed with powers of 16

Hexadecimal - Examples

$0_{16} = 0$

$1_{16} = 1$

$2_{16} = 2$

$9_{16} = 9$

$A_{16} = 10$

$B_{16} = 11$

$F_{16} = 15$

$10_{16} = 16$

$1F_{16} = 31$

$A0_{16} = 160$

$A1_{16} = 161$

$A9_{16} = 169$

$AA_{16} = 170$

$B0_{16} = 176$

$FA_{16} = 250$

$FF_{16} = 255$

$100_{16} = 256$

$1000_{16} = 4096$

Octal (Base-8)

The digits are:

0 1 2 3 4 5 6 7 8

$$534_8 = 5\times8^2 + 3\times8^1 + 4\times8^0$$
$$= 5\times64 + 3\times8 + 4\times1$$
$$= 320 + 24 + 4$$
$$= 348$$

The base is 8 this time so we proceed with powers of 8

Notice how the last digit always indicates ones.

sourcemind

Hex to Binary and vice versa

Since 2, 8 and 16 are all powers of 2, a special correspondence exists between them that simplifies the process of direct conversions from one base to another and the problem becomes a matter of grouping or breaking and substitution.

- **Binary to hex**
  Gather the bits in groups of **four,** replace each group with a hex digit.

$$2B_{16} = 101011_2$$

$$\underbrace{0000}_{0}\ \underbrace{0000}_{0}\quad \underbrace{1111}_{F}\ \underbrace{1111}_{F}\quad \underbrace{0010}_{2}\ \underbrace{1011}_{B}$$

- **Hex to Binary**
  Replace each hex digit with the four bits that represent that digit.

Take advantage of this “shortcut” whenever possible and **do not take the long route** with all those divisions when you do not have to.

Hexadecimal - Digit correlations

$0_{16} = 0000 = 0$

$1_{16} = 0001 = 1$

$2_{16} = 0010 = 2$

$3_{16} = 0011 = 3$

$4_{16} = 0100 = 4$

$5_{16} = 0101 = 5$

$6_{16} = 0110 = 6$

$7_{16} = 0111 = 7$

$8_{16} = 1000 = 8$

$9_{16} = 1001 = 9$

$A_{16} = 1010 = 10$

$B_{16} = 1011 = 11$

$C_{16} = 1100 = 12$

$D_{16} = 1101 = 13$

$E_{16} = 1110 = 14$

$F_{16} = 1111 = 15$

Octal Conversions

- **Octal to/from decimal**
  Is carried out via the normal routine.

- **Binary to octal and vice versa**
  The mapping rules are applicable for groups of **three** bits

- **Hex to Octal and vice versa**
  An intermediate conversion to binary instead of decimal reduces the complexities. Convert to binary first and then group accordingly

$$2B_{16} = 0010\ 1011_2 = \cancel{000}\ 101\ 011_2 = 53_8$$

$$54_8 = 101\ 100_2 = 0010\ 1100_2 = 2C_{16}$$

sourcemind

So what is the meaning of 1000?

- Symbols do not have inherent meanings.
- There can be an infinite variety of radices and therefore and infinite amount of numbers.
- Can mean something completely different:
  Utility room on level 10
  A brand of soft drink, etc

Byte

- Computer data is organized in **bytes.**
- Each byte is **8 bits**.
- $2^8 = 256$ unique variations.
- Integers in the range

  0 - 255

- Larger integers require allocation of more bytes

Bit - Binary digit

1010 0101

Byte

Byte Values (Unsigned)

0000 0000	=	0	0000 **1000**	=	8
0000 000**1**	=	1	0000 **1010**	=	10
0000 00**10**	=	2	0000 **1111**	=	15
0000 00**11**	=	3	**1000 0000**	=	128
0000 0**100**	=	4	**1000 0001**	=	129
0000 0**101**	=	5	**1111 1110**	=	254
0000 0**110**	=	6	**1111 1111**	=	255

sourcemind

Byte Values in Hex

A byte can also be represented using **two hex digits** in order to make it more legible in a human-friendly manner.

$00_{16} = 0000\ 0000_2 = 0$

$01_{16} = 0000\ 0001_2 = 1$

$02_{16} = 0000\ 0010_2 = 2$

$09_{16} = 0000\ 1001_2 = 9$

$0A_{16} = 0000\ 1010_2 = 10$

$0B_{16} = 0000\ 1011_2 = 11$

$0F_{16} = 0000\ 1111_2 = 15$

$10_{16} = 0001\ 0000_2 = 16$

$11_{16} = 0001\ 0000_2 = 17$

$AA_{16} = 1010\ 1010_2 = 176$

$FE_{16} = 1111\ 1110_2 = 254$

$FF_{16} = 1111\ 1111_2 = 255$

$1A_{16} = 26$

$A0_{16} = 160$

$A1_{16} = 161$

$AE_{16} = 174$

$AF_{16} = 175$

$B0_{16} = 176$

Negative Numbers

The mathematical notation for indicating negative numbers on paper, is a leading minus sign, regardless of the base.

$$25_{10} = 11001_2 \qquad -25_{10} = -11001_2$$

Computers, however, only have bits and therefore the model has to be such that does not rely on other symbols.

Moreover, the fact that the lengths of numbers are limited by the upper bounds of their storage chunks (e.g byte), can be utilized to our advantage.

One’s Complement

The most rudimentary way to represent a negative number in a computer (byte), is to simply **flip** its bits:

- replace 0 with 1, 1 with 0

- first bit indicates sign.

While this model is perfectly capable of storing numbers, it results in undesirable arithmetic and algebraic properties (e.g. positive and negative zeros) and incompatibilities during operations.

In spite of the said disadvantage, one’s complement is still useful in certain situations and there is an operator for it: ~

Two’s Complement

A better way for storing, representing and working with negative numbers is **two’s complement.** It relies on the following mathematical property of negative numbers:

$$a + (-a) = 0$$

For a value that is stored in two’s complement notation, its inverse is the value that when added to it, results in all 0’s inside and an **overflown** 1 which is discarded.

It is somewhat analogous to the number of minutes in an hour. If it’s past 20 minutes and we add 40 minutes, it results in 1 hour and 0 minutes. That way, **40 is -20**.

Note how the 60-minute **upper boundary matters**.

Two's Complement

Positive numbers in two's complement representation remain as they are. This, is a positive number in 8-bit two's complement notation.

0110 1010

In order to get the negative counterpart of that number, the following operation is performed.

Step 1: Flip the bits **1001 0101**

Step 2: Add one **1001 0110**

The first bit still serves as a minus sign.

Two's Complement

Note that "two's complement" is the name of **both the representation and the operation**.

The operation is the equivalent of **multiplying something by -1**.

Applying the same operation to the two's complement representation of a negative number, yields a positive number.

Indicating the total number of bits is vital. The equivalent of our number in 16-bit two's complement notation is:

	0000 0000 0110 1010
1. Flip the bits	1111 1111 1001 0101
2. Add one	1111 1111 1001 0110

sourcemind

Two's Complement

The sum of any number and its complement is 0.

$$
\begin{array}{r}
\boxed{0110\ 0101} \\
+ \qquad\qquad \\
\boxed{1001\ 1011} \\
\hline
1\ \boxed{0000\ 0000}
\end{array}
$$

Discarded

sourcemind

Byte Values (Signed)

0000 0000	=	0	0100 0000	=	64
0000 0001	=	1	0111 1110	=	126
0000 0010	=	2	0111 1111	=	127
0000 0100	=	4	1000 0000	=	- 128
0000 1000	=	8	1000 0001	=	- 127
0001 0000	=	16	1111 1110	=	- 2
0010 0000	=	32	1111 1111	=	- 1

Byte

- $2^8 = 256$ unique variations.
- The most significant bit can act as a digit or a minus sign depending on the context
- Integers in the range

  0 to 255 unsigned

  -128 to 127 signed

Bit - Binary digit

Can act as a digit or a minus sign

1010 0101

Byte

Integer Overflow Bugs

Larger integers need the allocation of more bytes.

Exceeding the capacities of allocated memories during arithmetic operations might result in erroneous and unpredictable values.

The **sum of two positive integers** might end up being a **negative value.**

Such bugs can be exploited in order to carry **integer overflow attacks**

One such example is a boss in a game which can be killed by healing it.
https://www.reddit.com/r/programming/comments/1aigv9/integer_overflow_in_an_rpg_defeat_a_boss_by/

Fascinating Numbers

- $2^{32} = 4{,}294{,}967{,}295$
- IPv4 is 4 bytes (32 bits). E.g. 255.255.255.255
- The number of devices connected to a network that uses this protocol(Internet) is approx. 4 billion.
- $2^{128} \sim 3.4{\times}1038$
- IPv6 is 16 bytes.

  E.g. 2001:0db8:85a3:0000:0000:8a2e:0370:7334
- Safe key length for cryptographic purposes is 256 bits
- 2^{1024} is more than the number of protons in the observable universe

sourcemind

Characters

ASCII (American Standard Code for Information Exchange) is an older character encoding standard.

Extended ASCII represents a character with a single byte (hence 256 characters). Includes capital and small letters, digits, symbols, table borders, etc.

See the full list: http://www.asciitable.com/

ASCII Examples

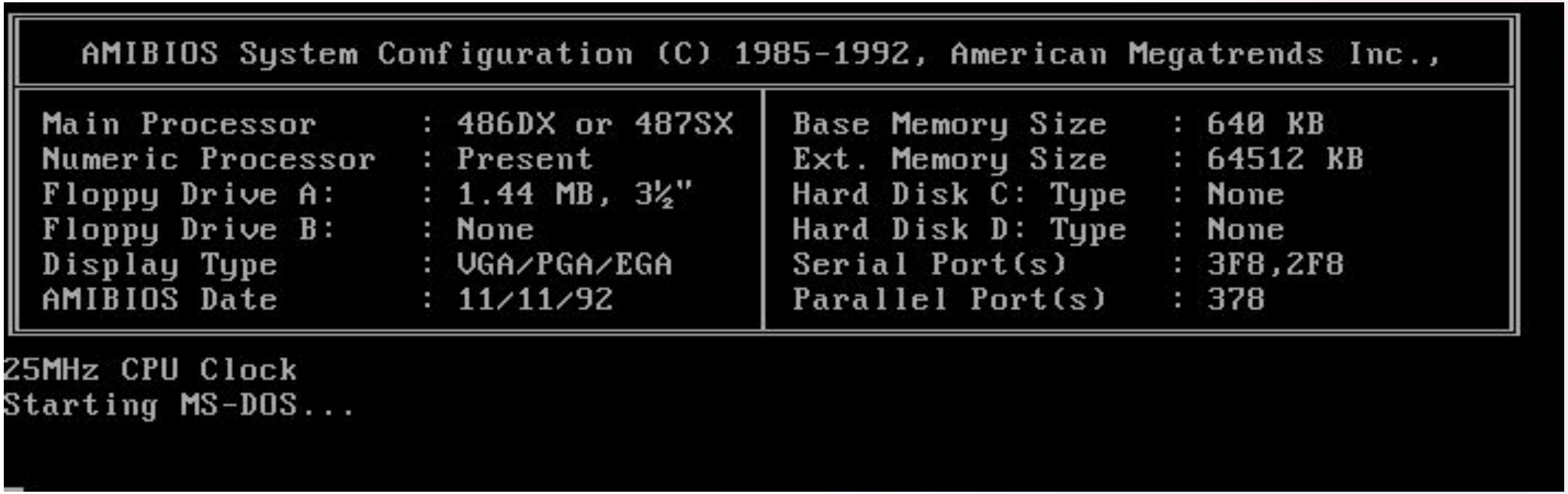


Table borders are characters.

ASCII Examples - ASCII Art

The practice of drawing pictures using ASCII characters is referred to as ASCII art.

ASCII Examples

Graphical user interface with shadows in ASCII

RGB Colors

One of the many ways of representing colors is by mixing various amounts of red, green and blue together. Each of those amounts can be represented by a number (byte)

Red: 0 - 255 Green: 0 - 255 Blue: 0 - 255

The following table shows a way of representing colors in computers. (Particularly CSS and web)

#000000	#0000FF	#8e7cc3
#111111	#00FF00	#987654
#AEAEAE	#FFFF00	#654321
#FFFFFF	#FF00FF	#123456
#EAFFFF	#00FFFF	#ABCDEF
#FF0000	#ff9900	#FEDCBA

One more byte can be allocated in order to indicate transparency.

That scheme is called RGBA, A stands for Alpha

Floating Point (Simplified)

0 00000000 00000000000000000000000

Sign

Exponent (8 bits)
11 for double

Significand (23 bits)
52 for double

$$(-1)^{\square} \times \square \times 10_b^{\square}$$

IEEE 754

The actual standard for floating point numbers is more complex and is specified by IEEE 754, covers rounding behaviors as well.

These numbers are intended for scientific purposes and are not suitable for currencies.

Rounding errors can potentially cause significant losses; The Ariane 5 disaster is a notable example.

https://www.bugsnag.com/blog/bug-day-ariane-5-disaster

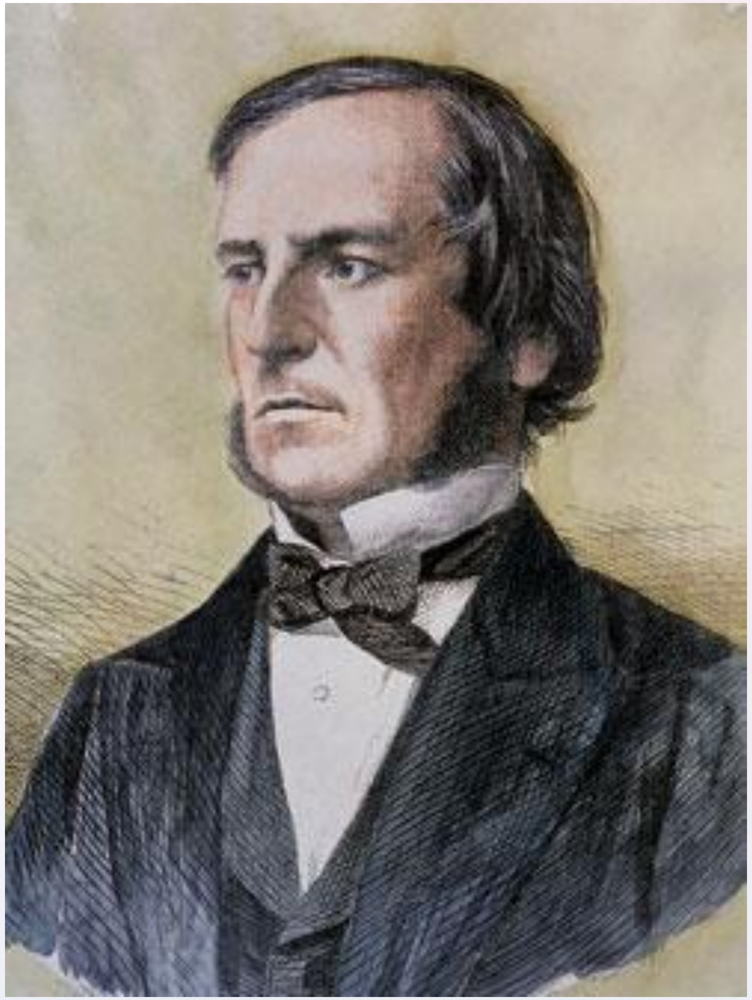
BREAK
5 minutes

sourcemind

Boolean Algebra

“**Boolean algebra:** a system of algebra in which there are only two possible values for a variable (often expressed as true and false or as 1 and 0) and in which the basic operations are the logical operations AND and OR” (Merriam Webster, 2021)

With the invention of transistors, electrical circuitry that implemented Boolean algebra became possible.
https://www.youtube.com/watch?v=sTu3LwpF6XI



George Boole (1815-1864)

Unknown [Drawing of George Boole] c. 1865 Accessed Aug 28, 2019 from: http://commons.wikimedia.org/

sourcemind

Boolean Operators - Truth Tables for NOT, AND, OR, XOR

&	T	F
T	T	F
F	F	F

x AND y is T if both x and y are T.

V	T	F
T	T	T
F	T	F

x OR y is T if either x and y are T.

⊕	T	F
T	F	T
F	T	F

x XOR y is T if only one of x and y are T. Soup or salad

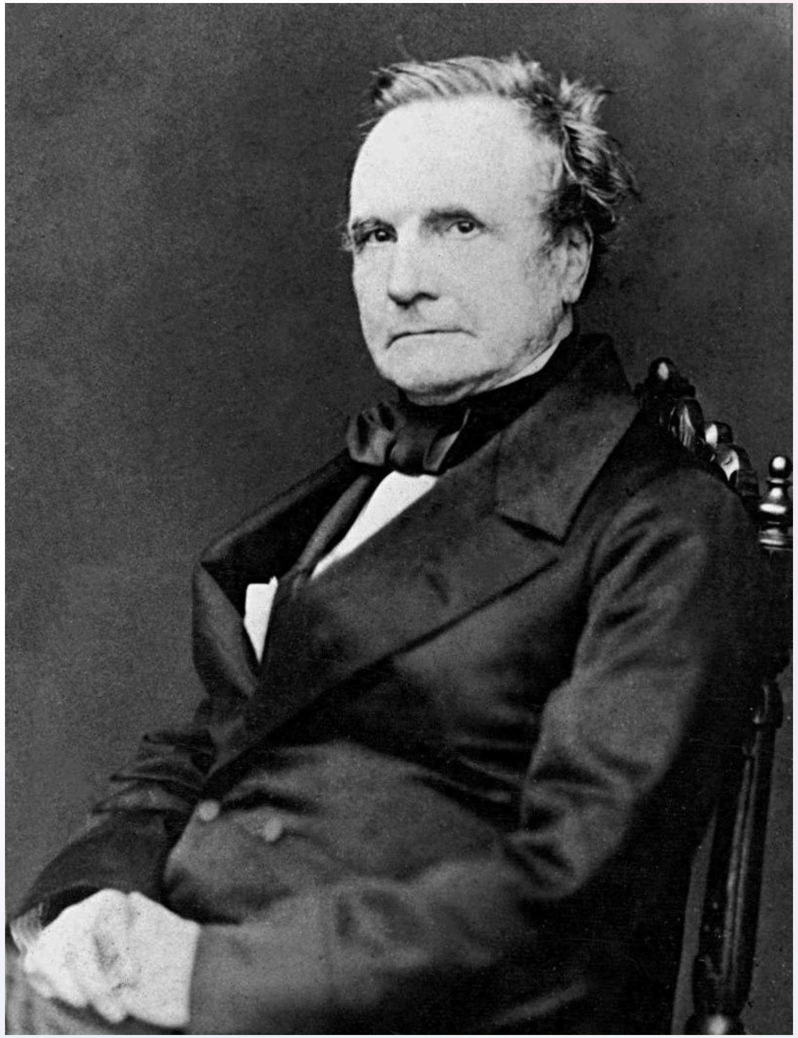
Inversion or NOT.

!	T	F
	F	T

sourcemind

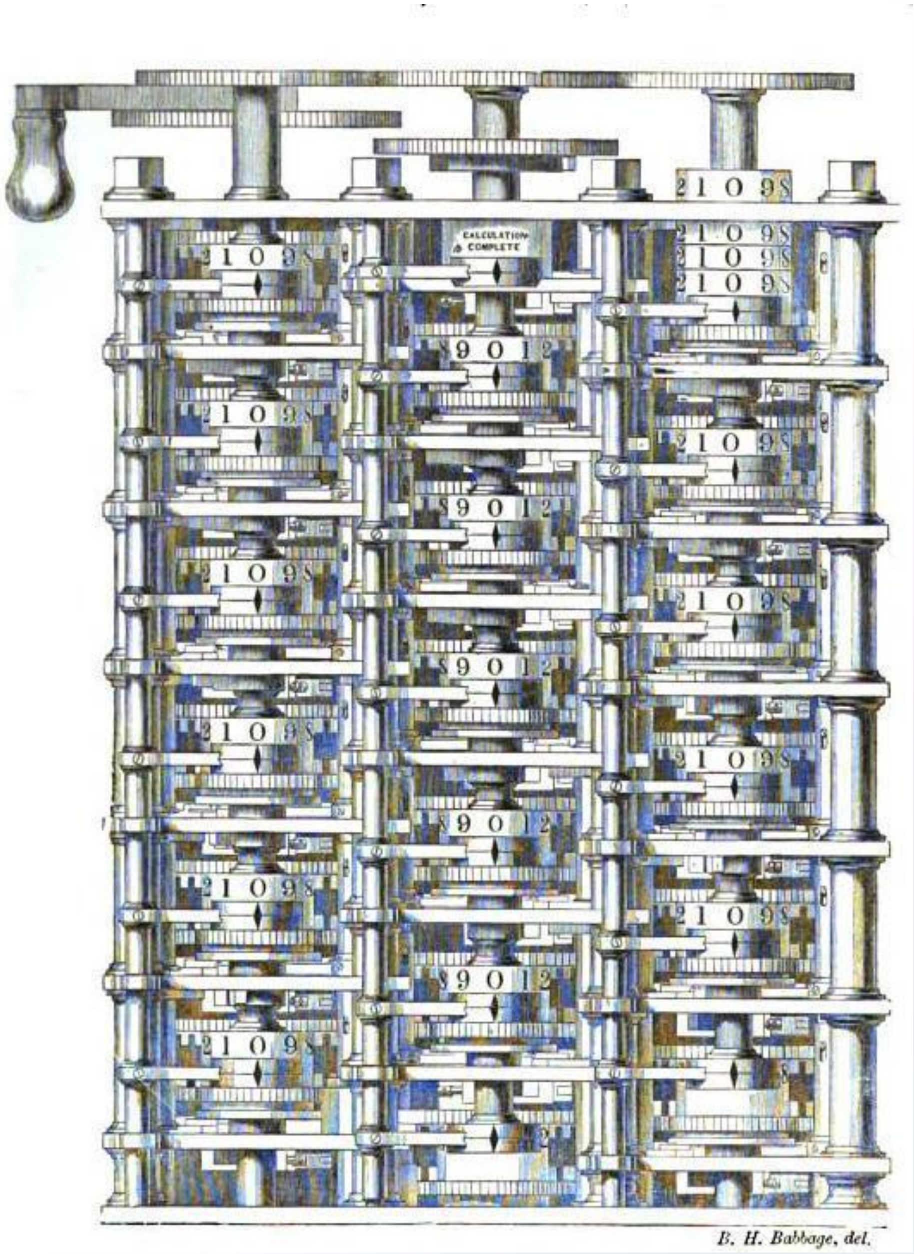
The Inventor of Computers

"**Charles Babbage**, (born December 26, 1791, London, England—died October 18, 1871, London), English mathematician and inventor who is credited with having conceived the first automatic digital computer." (Encyclopaedia Britannica, 2021)



By Unknown [Charles Babbage] Accessed Aug 28, 2019 from: http://commons.wikimedia.org/

sourcemind



Woodcut after a drawing by Benjamin Herschel Babbage Part of Charles Babbage's Difference Engine No. 1, as assembled in 1833, exhibited 1862, and later in the South Kensington Museum. 1853. Accessed Aug 28, 2019 from: http://commons.wikimedia.org/

sourcemind

The First Programmer

"**Ada Lovelace**, in full **Ada King, countess of Lovelace**, original name **Augusta Ada Byron, Lady Byron,** (born December 10, 1815, Piccadilly Terrace, Middlesex [now in London], England—died November 27, 1852, Marylebone, London), English mathematician, an associate of Charles Babbage, for whose prototype of a digital computer she created a program. She has been called the first computer programmer."
(Encyclopaedia Britannica, 2021)

Ada Lovelace (1815 - 1852)

A. E. Chalon "Watercolor portrait of Ada King, Countess of Lovelace (Ada Lovelace)" c. 1840, Science Museum Group. Accessed Aug 28, 2019 from: http://commons.wikimedia.org/

The First Binary Computer

- The first binary computer was completed by German engineer Konrad Zuse in 1941
- The original no longer exists.

(Encyclopaedia Britannica, 2021)

Konrad Zuse (1910 - 1995)

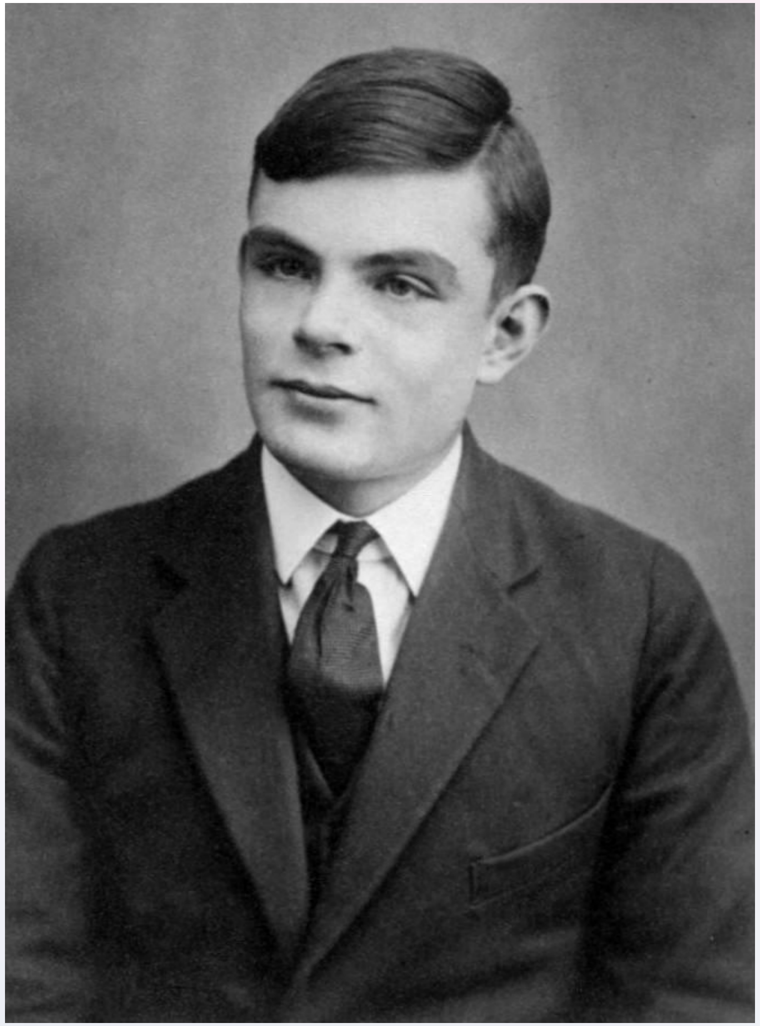
Wolfgang Hunscher, Dortmund. *Konrad Zuse.* 1992. Accessed Aug 28, 2019 from: http://commons.wikimedia.org/

The Father of Computer Science

The Turing Machine is a conceptual problem solving machine that:

- Consists of an infinite tape with cells that can store symbols
- Has a head that can move left or right and read and write symbols on the tape in each step

This machine is capable of solving anything that can be solved with an algorithm.



Alan Turing (1912 - 1954)

Unknown *[Alan Turing at age 16]* 1928. Accessed Aug 28, 2019 from: http://commons.wikimedia.org/

Turing Machine

The lambda symbol denotes an empty cell.

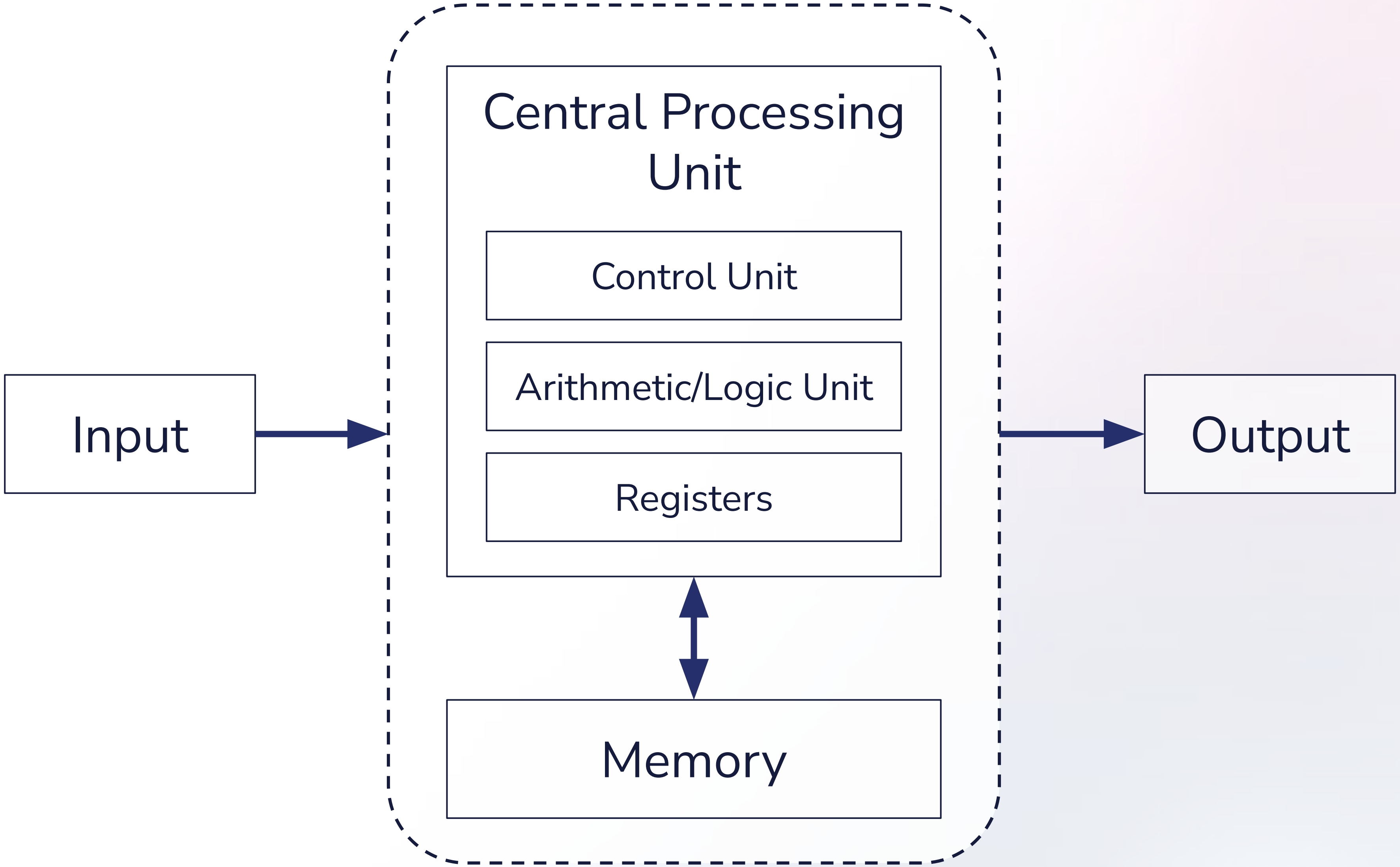
Von Neumann Architecture

- Same memory is used for both data and instructions.
- Makes self-modifying programs and programs that make other programs, possible. (Compilers, interpreters)
- A lot of modern computing is based on this model, namely IBM compatible personal computers.

John Von Neumann (1903-1957)

Los Alamos National Lab. John Von Neumann. 2008. Accessed Aug 28, 2019 from: http://commons.wikimedia.org/

Von Neumann Architecture



sourcemind

IBM-Compatible PC - Components

- RAM

  Also called primary memory or simply memory. Requires power and all data is lost once the power is cut off (i.e. volatile)

- CPU

- Storage

  Such as hard disk drives (HDD), solid state drives (SSD) etc.

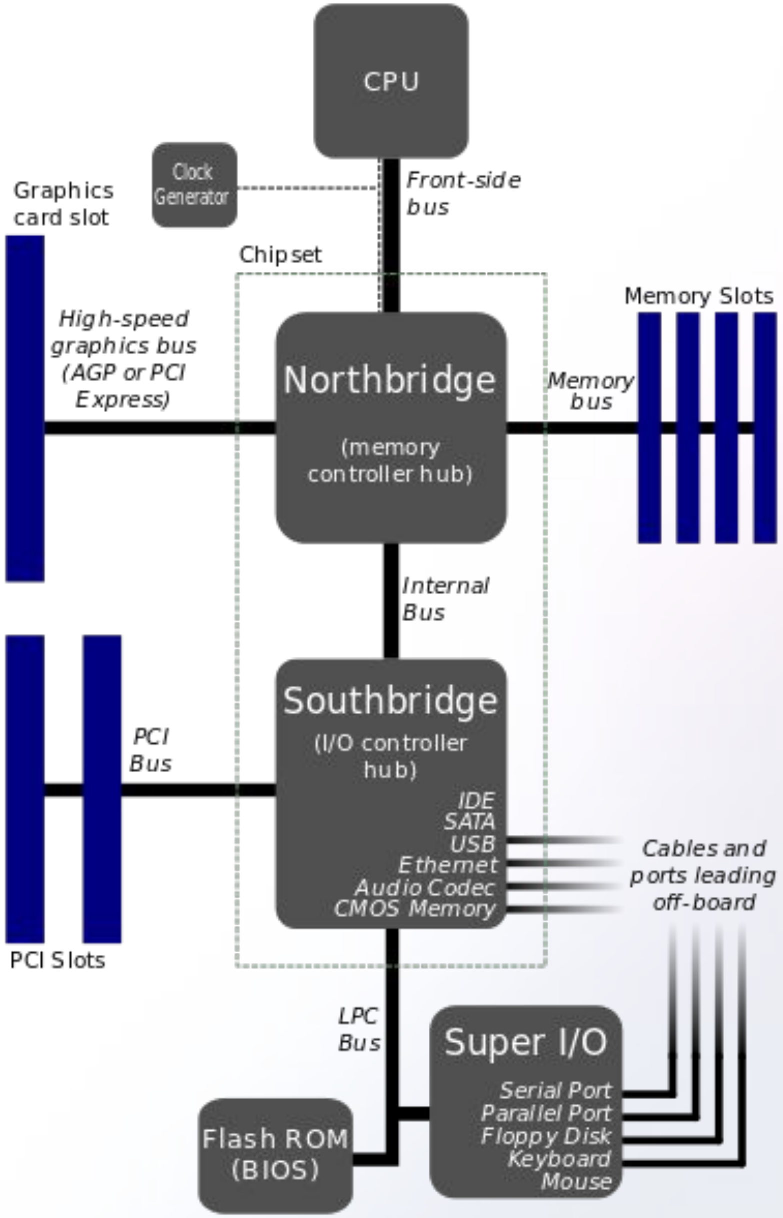
IBM-Compatible PC - Optional Components and Peripherals

- Output devices (Monitors, printers, etc.)
- Input devices (Keyboard, mouse, etc.)
- VGA (Video graphics adapter) processes visual outputs.

  Come with processors and memory of their own. Can be programmed for performing other tasks.

  Multiple cards can perform parallel work with technology such as CUDA or OpenCL
- Various PCI components for connectivity (LAN, wifi)

Chipset



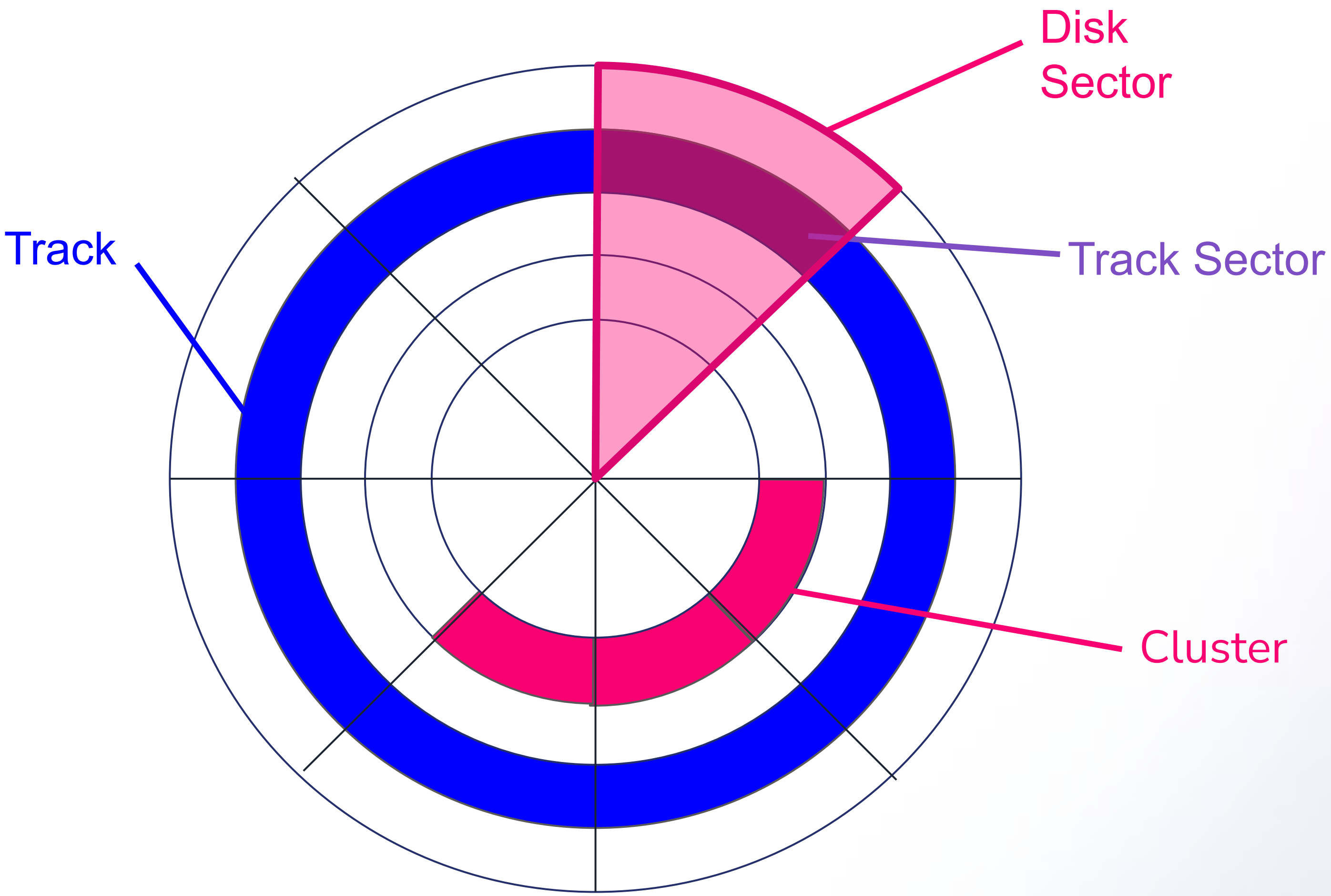
CPU
Clock Generator
Front-side bus
Graphics card slot
Chipset
Memory Slots
High-speed graphics bus (AGP or PCI Express)
Northbridge
(memory controller hub)
Memory bus
Internal Bus
Southbridge
(I/O controller hub)
IDE
SATA
USB
Ethernet
Audio Codec
CMOS Memory
PCI Bus
PCI Slots
Cables and ports leading off-board
LPC Bus
Super I/O
Serial Port
Parallel Port
Floppy Disk
Keyboard
Mouse
Flash ROM (BIOS)

Hard Disk Drives

- Magnetic storage remains relevant and abundant today
- Consists of platters made of aluminum or glass etc., with magnetic coating on one or both sides
- One or more of these platters, stacked on top of one another, spinning together
- Heads write by magnetizing the surface of the disk and read that magnetized surface.
- A gramophone track is a spiral, whereas hard drive tracks are concentric circles.

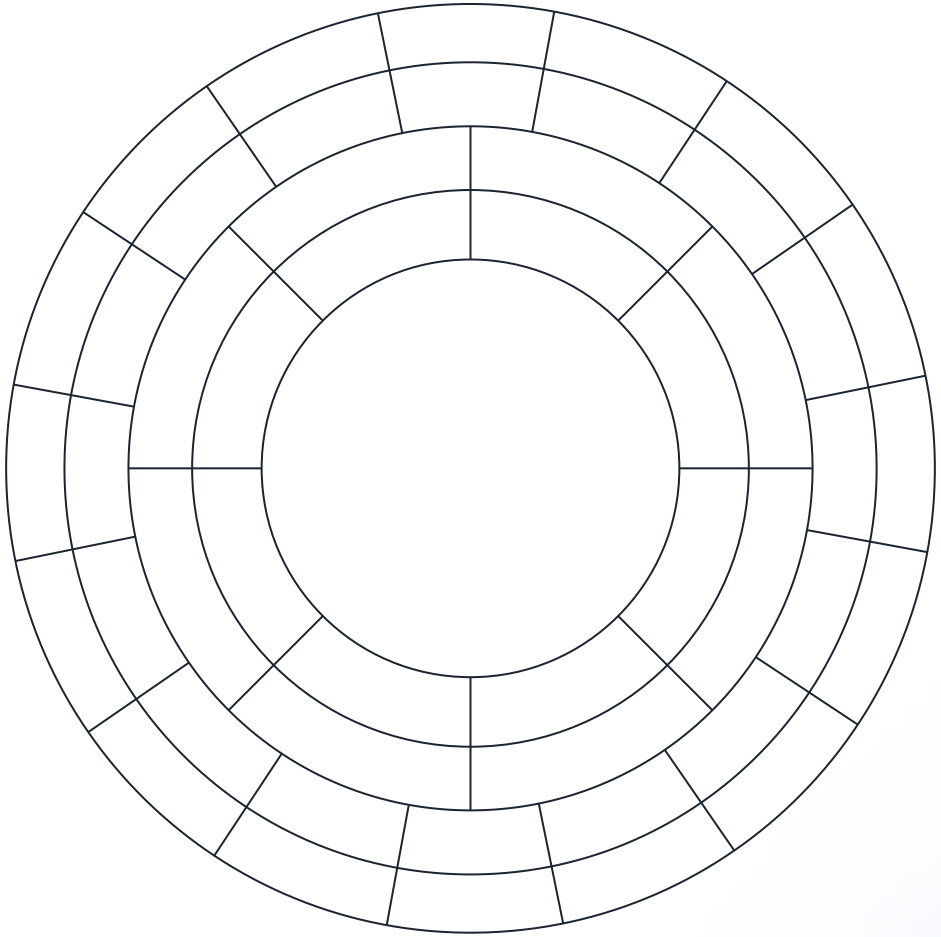
Evan-Amos "Internals of a 2.5-inch laptop hard disk drive" Accessed Jun 2, 2021 from: https://commons.wikimedia.org/wiki/File:Laptop-hard-drive-exposed.jpg

Hard Disk Drives



- Consists of platters made of aluminum or glass etc., with magnetic coating on one or both sides
- One or more of these platters, stacked on top of one another, spinning together
- Heads write by magnetizing the surface of the disk and read that magnetized surface, like that of a gramophone.
- A gramophone track is a spiral, whereas hard drive tracks are concentric circles.
- Sector length is usually 512 bytes.

Hard Disk Drives



Disks spin with a constant speed. Since heads read and write with a constant rate too, data becomes physically wider and occupies a wider arc the further it is from the center which is not efficient.

Newer hard drives implement zone bit recording schemes to use the surface area more efficiently.

Processor Architecture

- Every architecture has a number of registers, some are general purpose, some may be dedicated to special tasks or status flags.
- Every architecture also has its own instruction set.
- These micro-instructions accomplish a single, small task on the lowest level of the processor, such as moving data between registers and memory.
- Micro-instructions often correspond with a hardware component in the processor, they invoke an ALU, etc.

Architecture Examples

A few notable ones are:

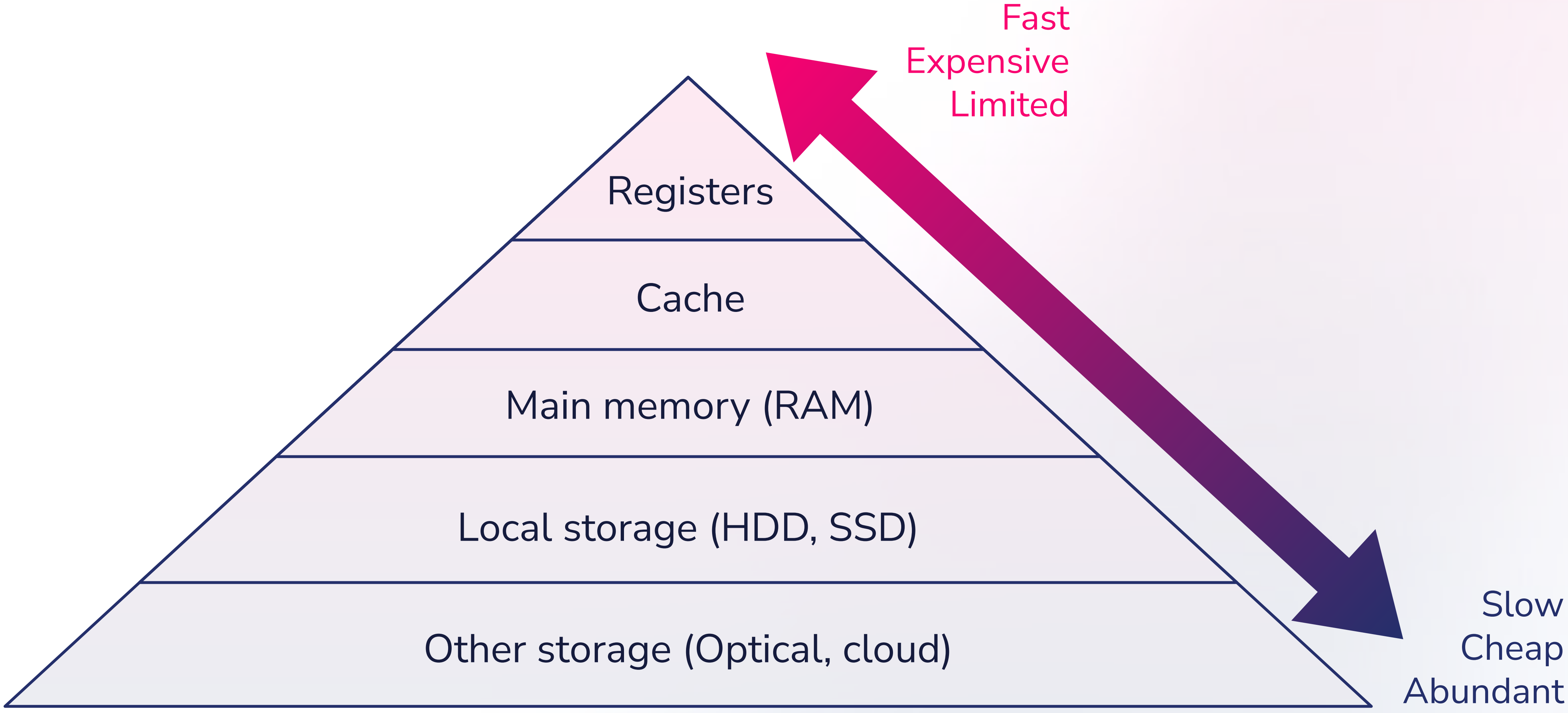
- Intel x86 family, dominates personal computing, such as x86-32 and x86-64. Compatible CPUs are also made by AMD.

- ARM, mostly on smartphones and smart TV.

  Apple M1 is also ARM-based.

- MIPS, used by Sony on their PlayStation consoles

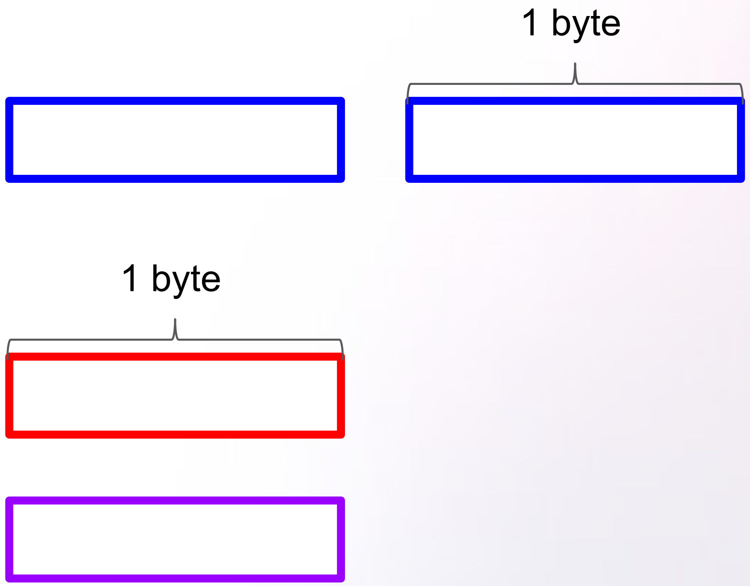
Memory Hierarchy



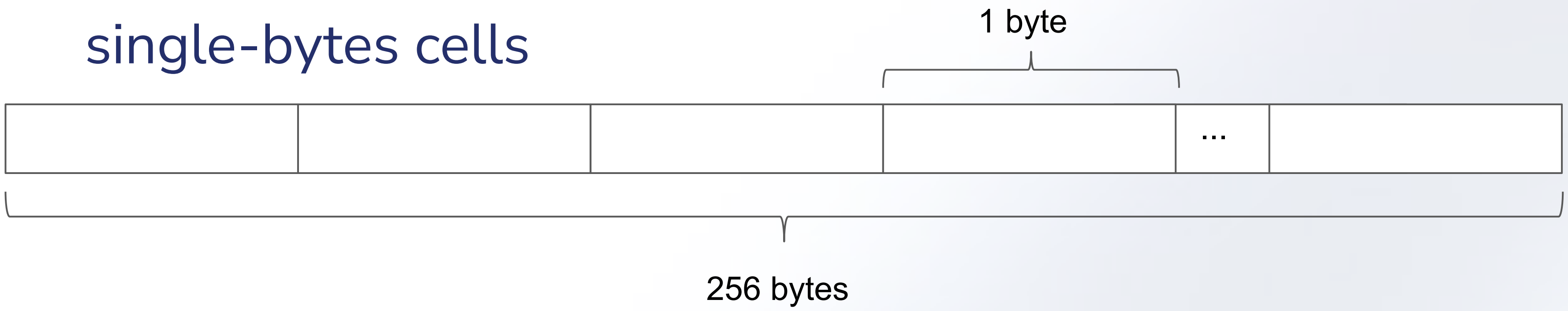
sourcemind

Processor Architecture - Registers

- Two general purpose registers, called A and B



- Instruction register
- Points to the next instruction
- Memory, consisting of 256 single-bytes cells



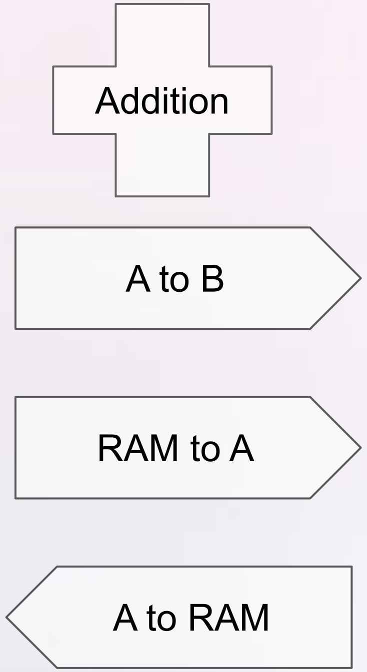
sourcemind

Processor Architecture - Arithmetic and Logic Units

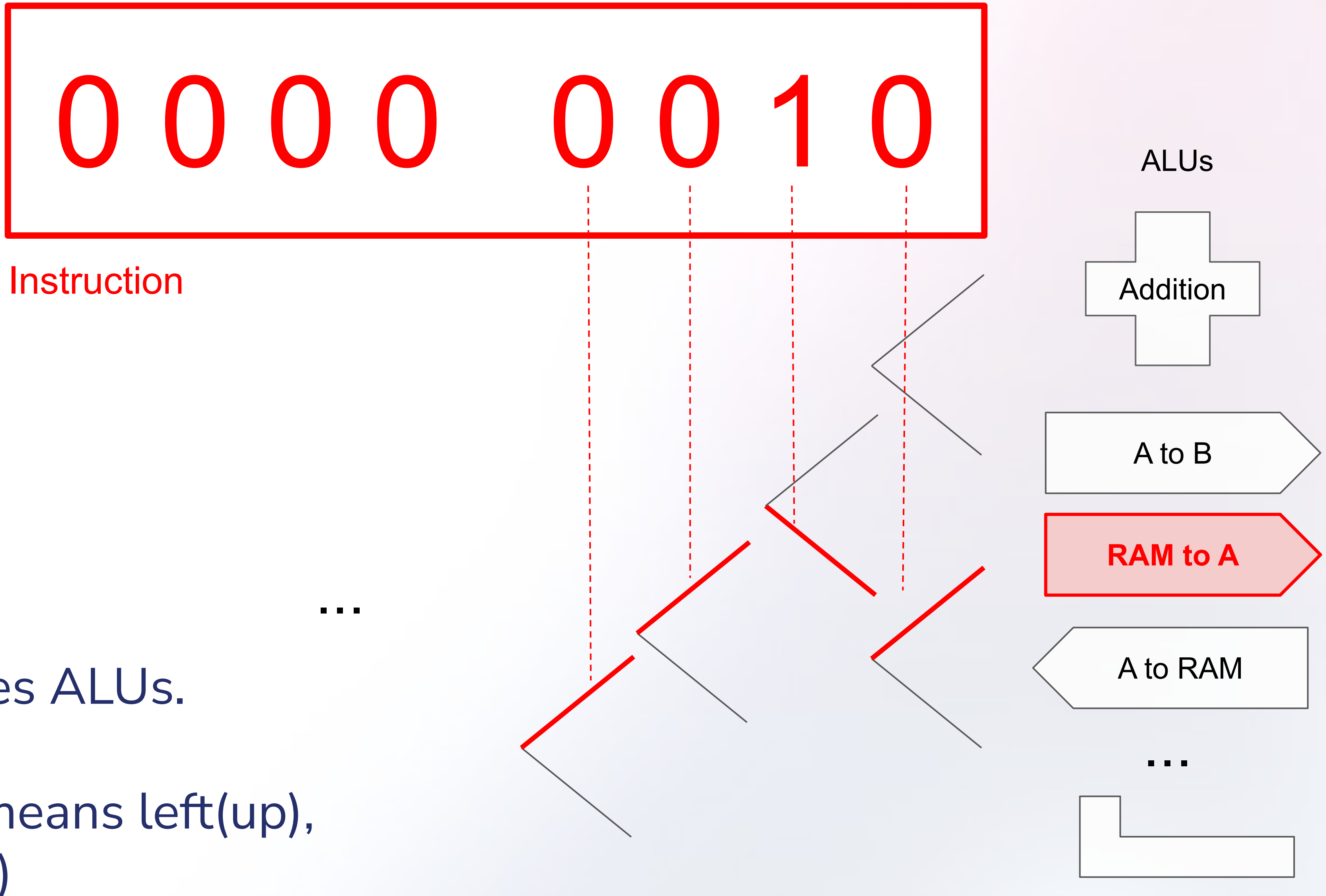
ALU are hardware circuitry designed to perform a specific task.

Our architecture has ALUs for copying things to and from memory and from one register to another, along with an ALU that performs addition and so on.

- 0 - Addition
- 1 - A to B
- 2 - Memory to A
- 3 - A to memory
- … up to 256.



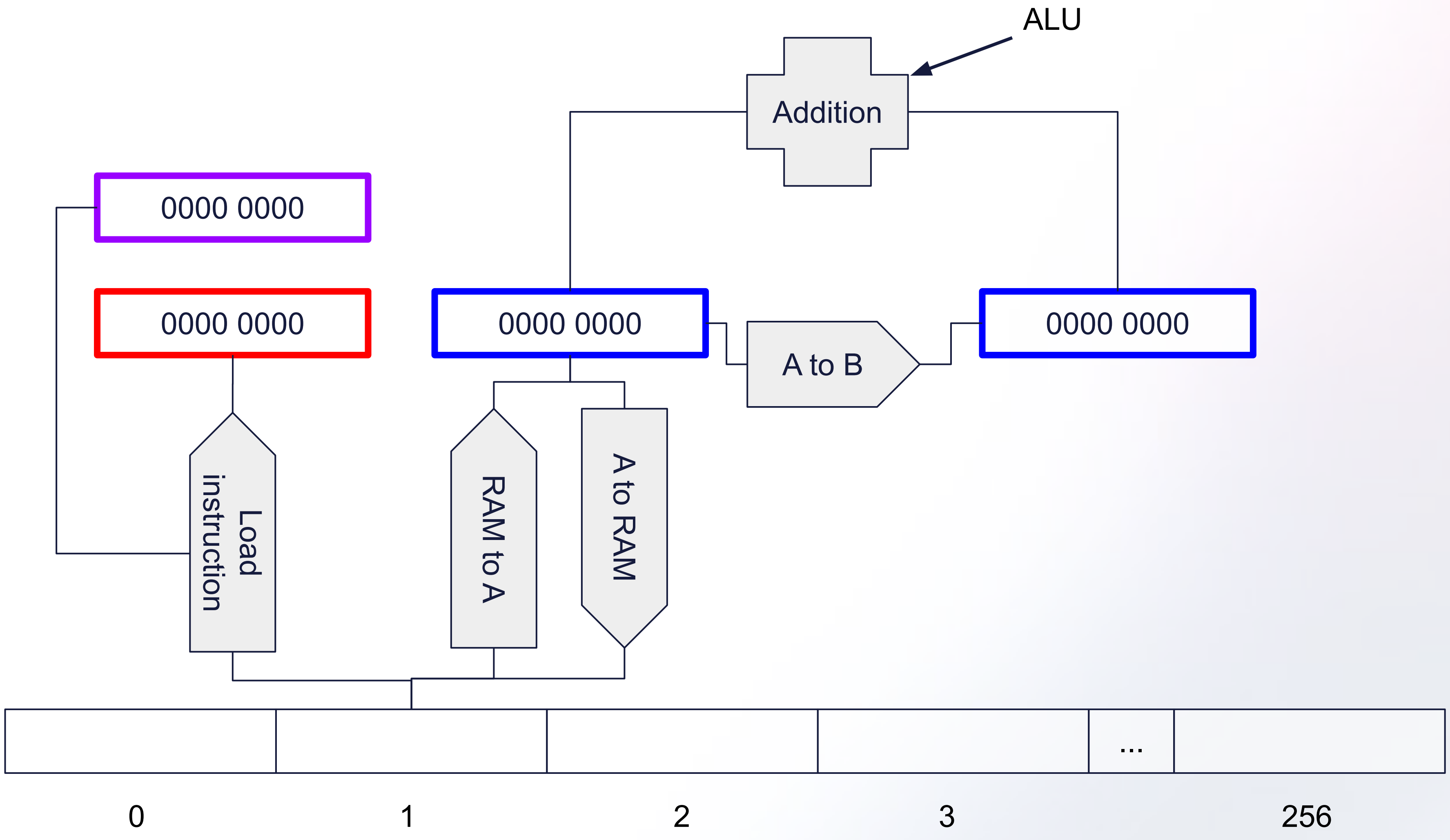
Processor Architecture - Control Unit



Control unit activates ALUs.

For our circuitry, 0 means left(up), 1 leads right (down)

Processor Architecture - Simple Implementation



Processor Architecture - Fetch, Decode, Exe

Using our architecture, we will simulate the Fetch-decode-exe cycle and run a program that calculates the sum of 6 and 7.

0000 0010 0000 0111

0000 0001

0000 0010 0000 0110

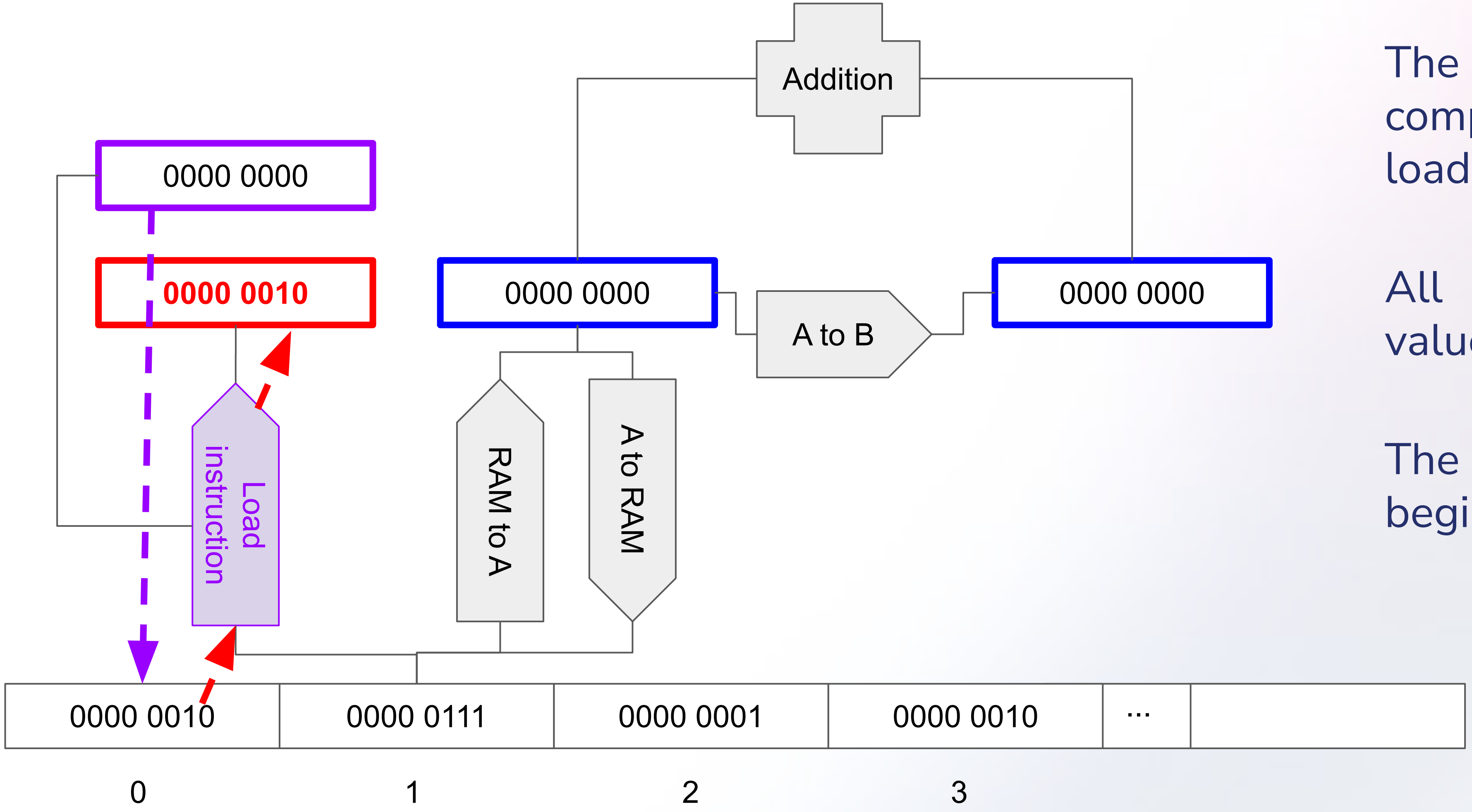
0000 0000

...

Updated values at each step are in **BOLD**

Activated components at each step are highlighted

Processor Architecture - Fetch

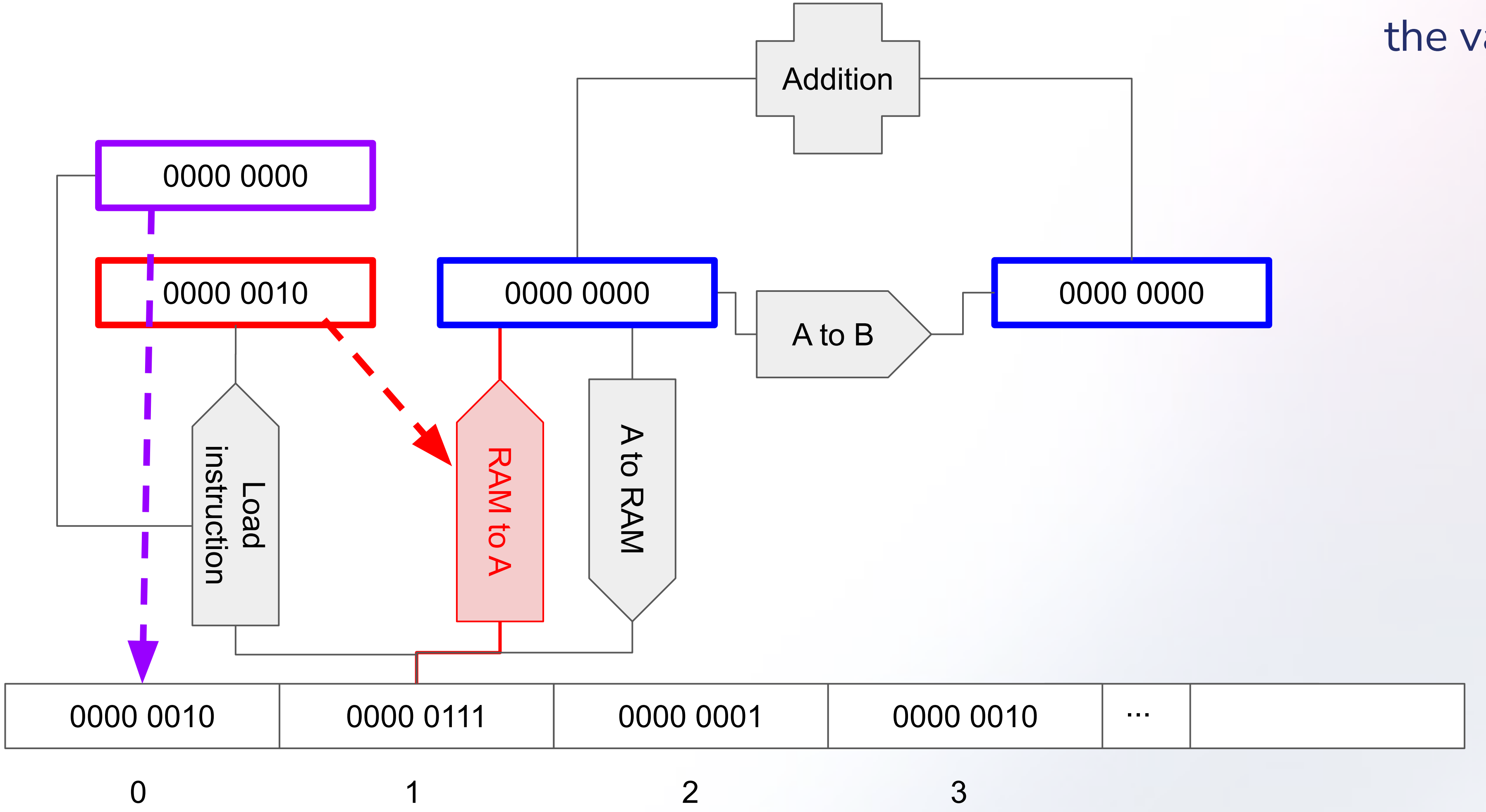


The initial state of our computer has a program loaded in its memory.

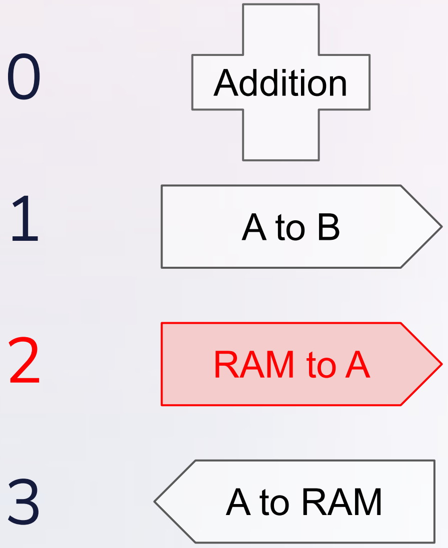
All registers start with the value 0.

The fetch-decode-exe cycle begins.

Processor Architecture - Decode

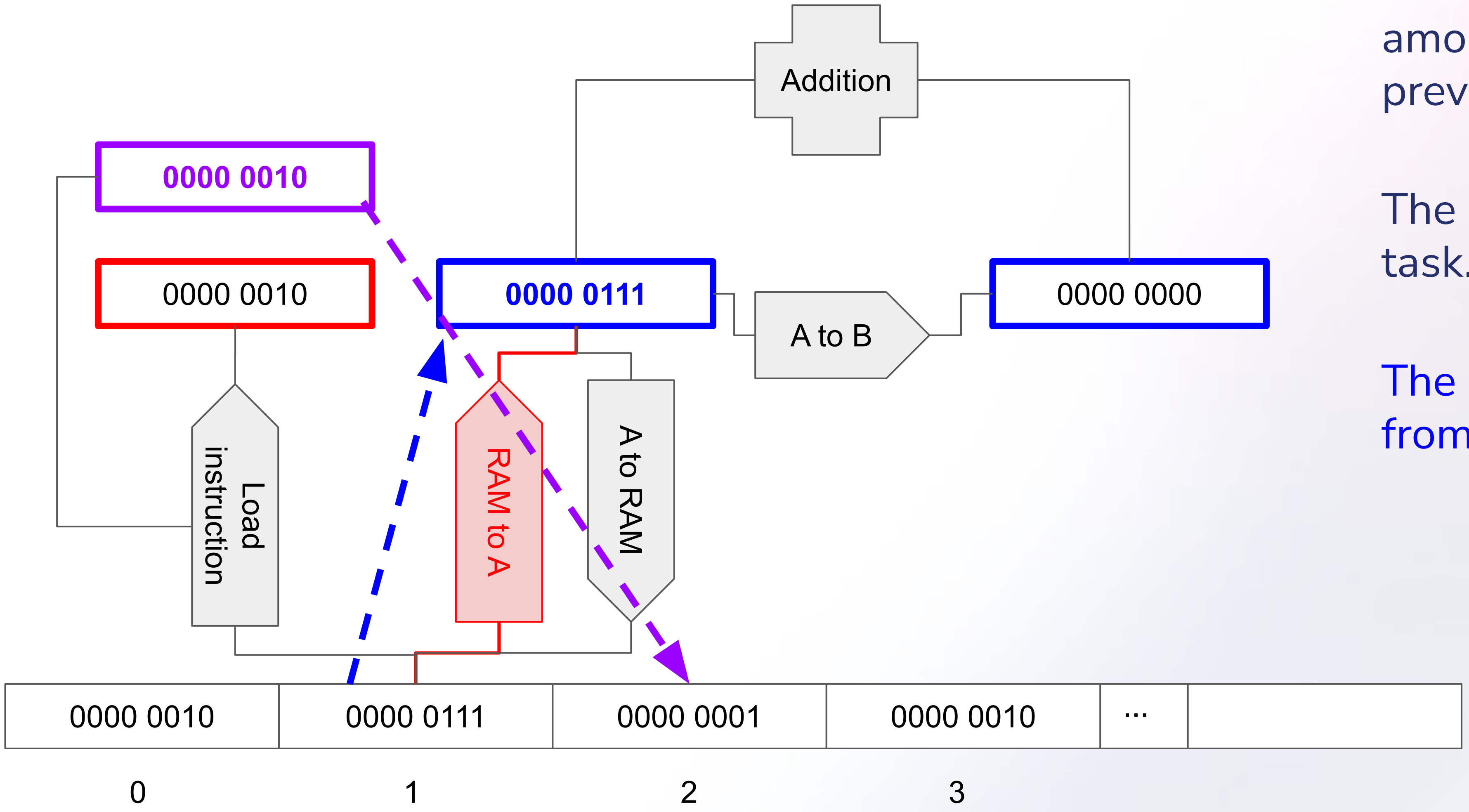


The ALU that corresponds to the value of 2 is activated.



sourcemind

Processor Architecture - Execute

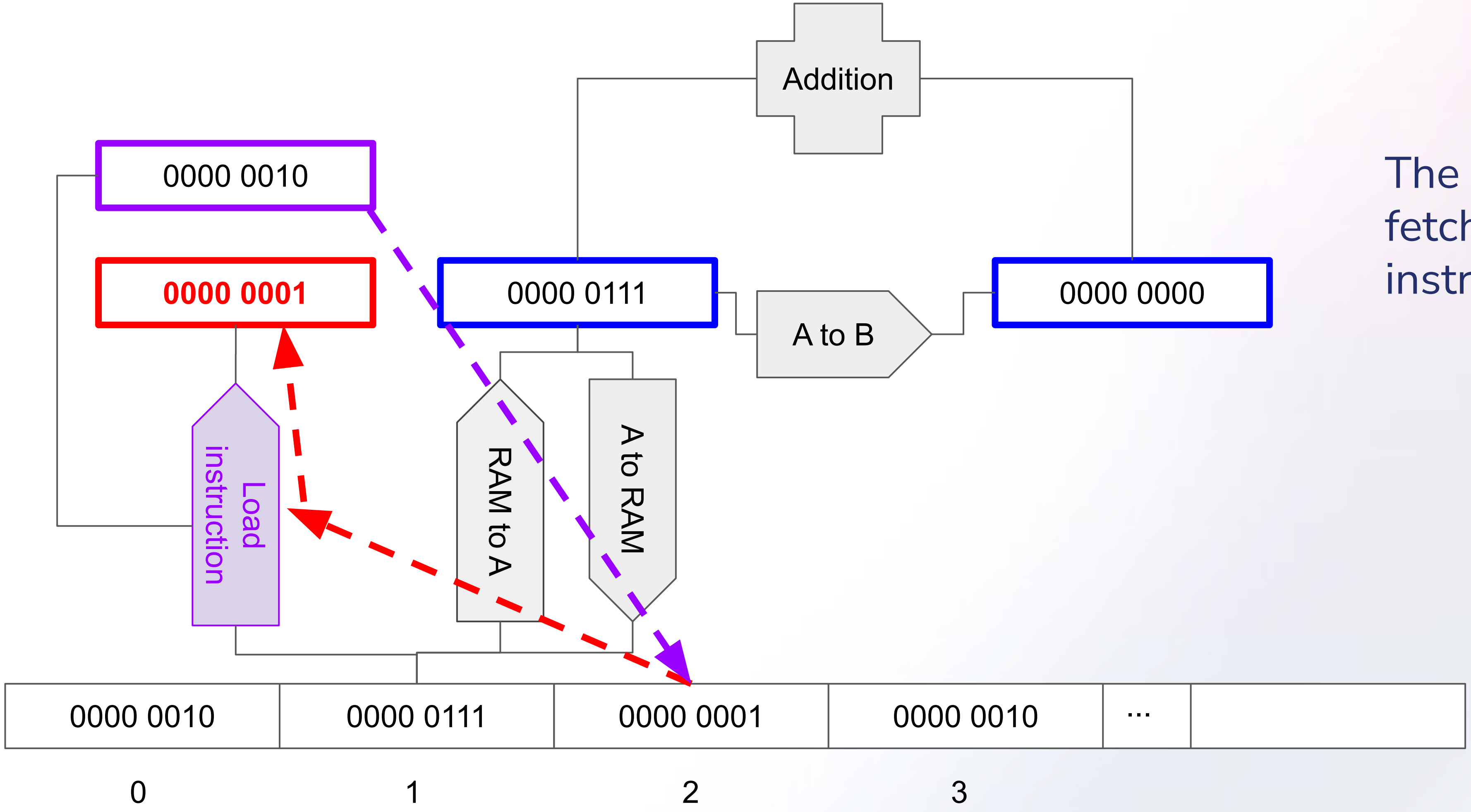


The instruction register is incremented by the necessary amount depending on the previous instruction.

The activate ALU performs its task.

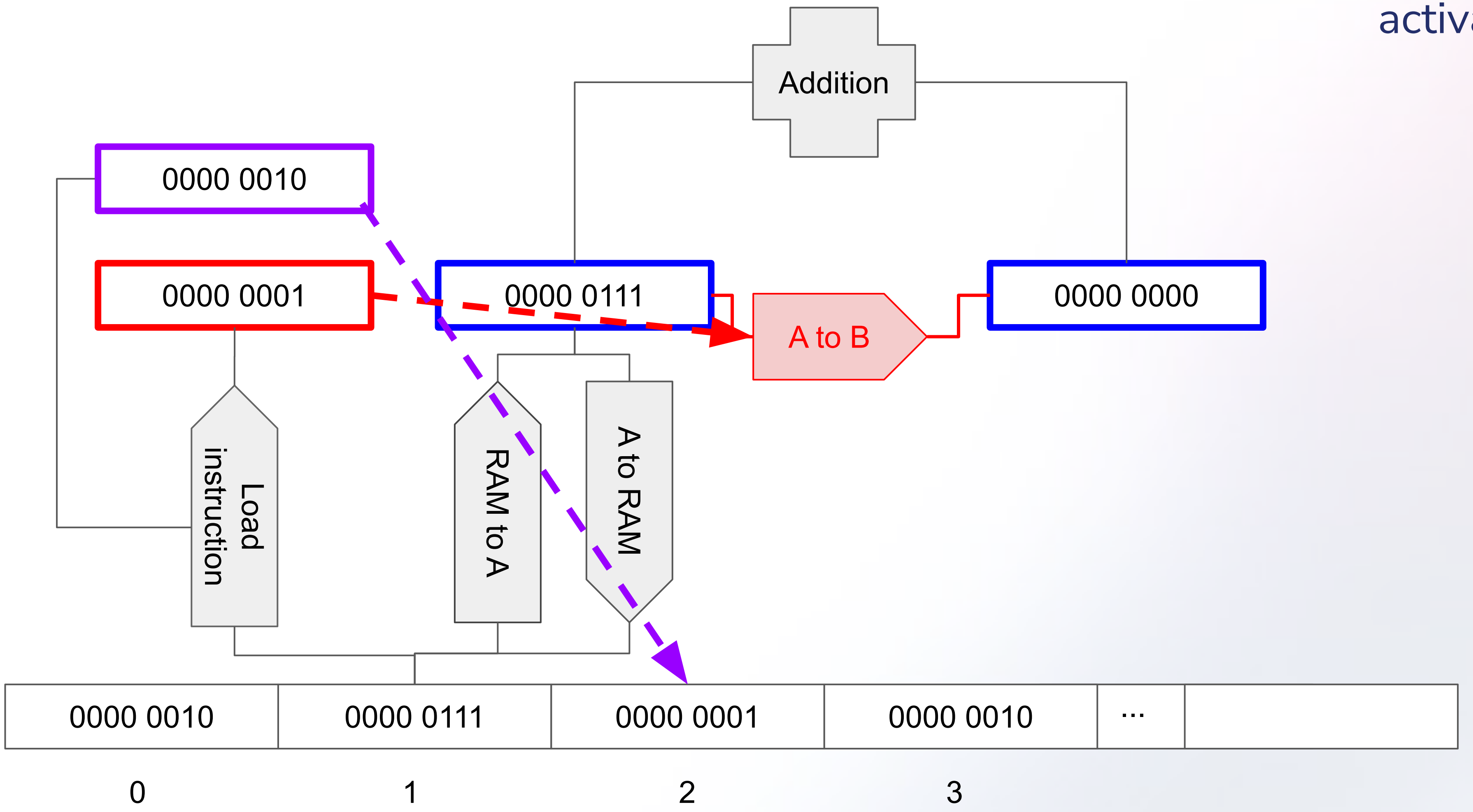
The value has been moved from RAM to A

Processor Architecture - Fetch

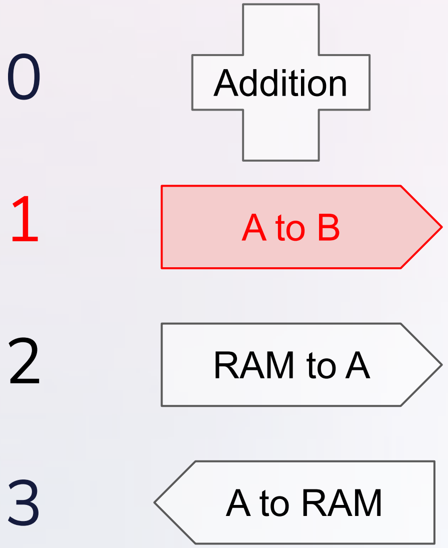


The instruction located at 2 is fetched and moved to the instruction register.

Processor Architecture - Decode

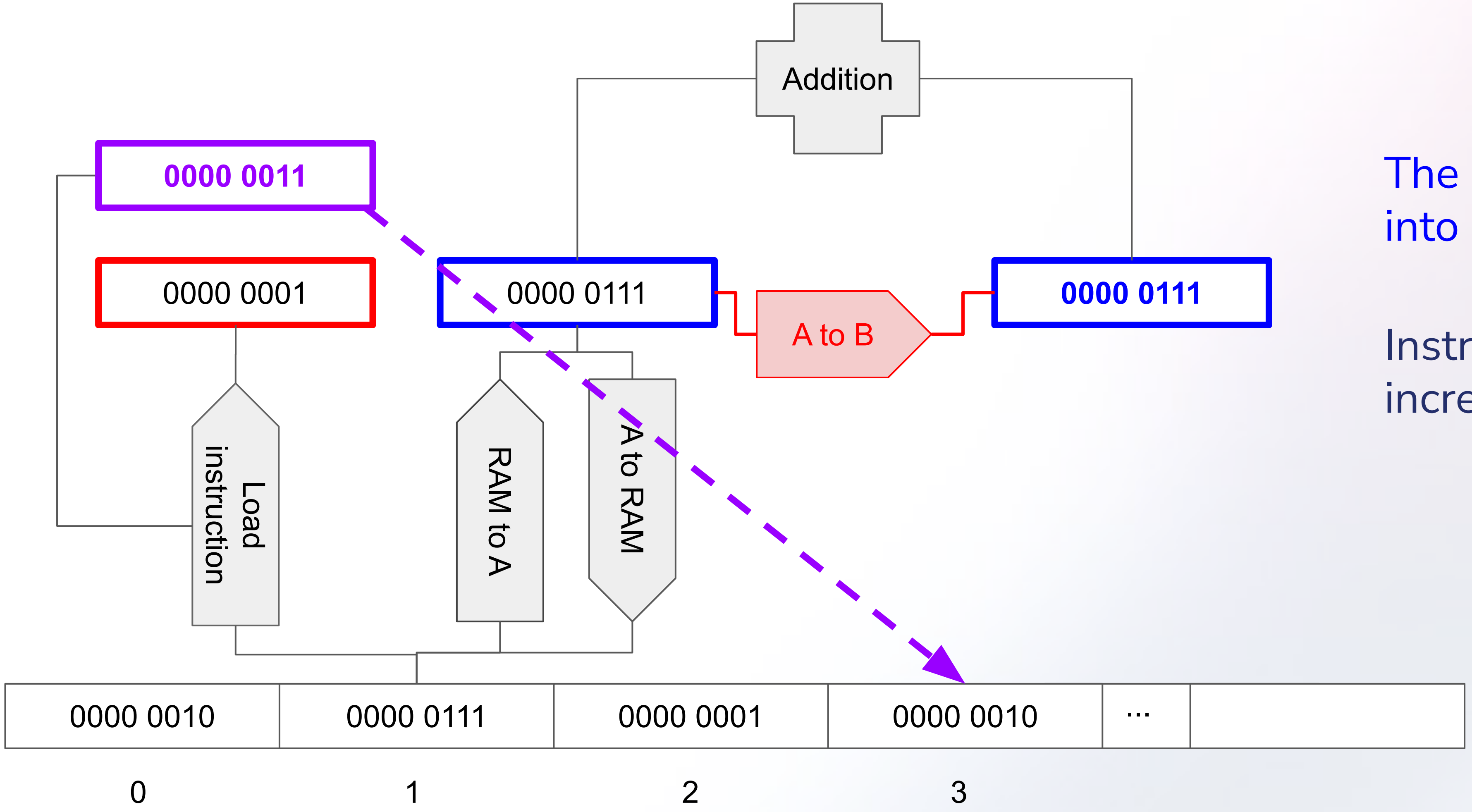


The ALU that corresponds with instruction code 1 is activated.



sourcemind

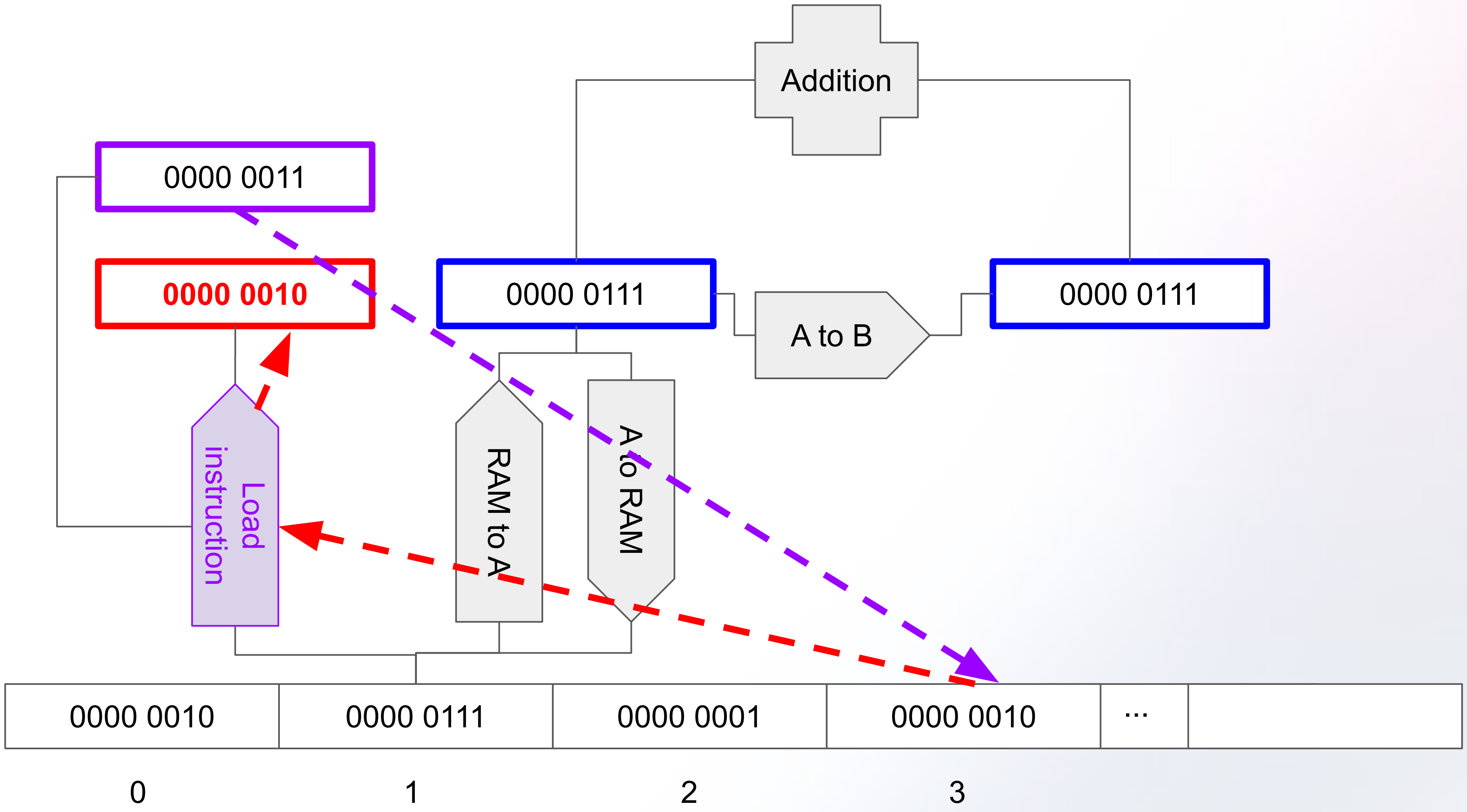
Processor Architecture - Execute



The contents of A is copied into B.

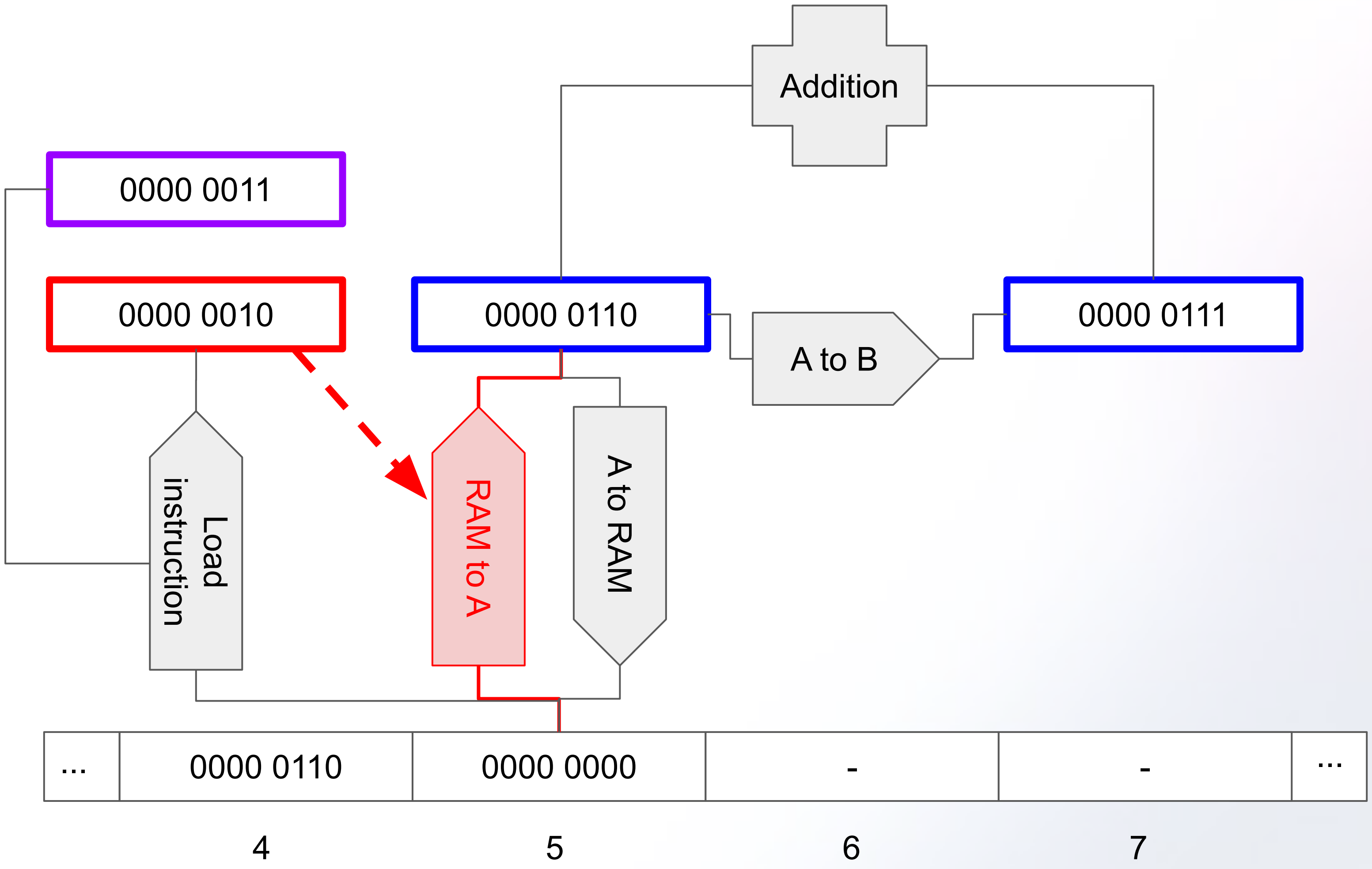
Instruction pointer is incremented accordingly.

Processor Architecture - Fetch

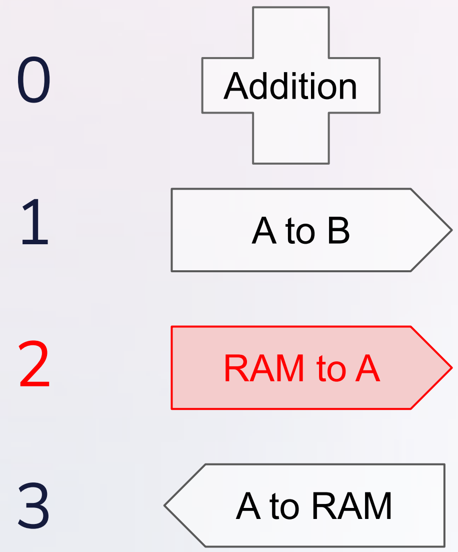


sourcemind

Processor Architecture - Decode

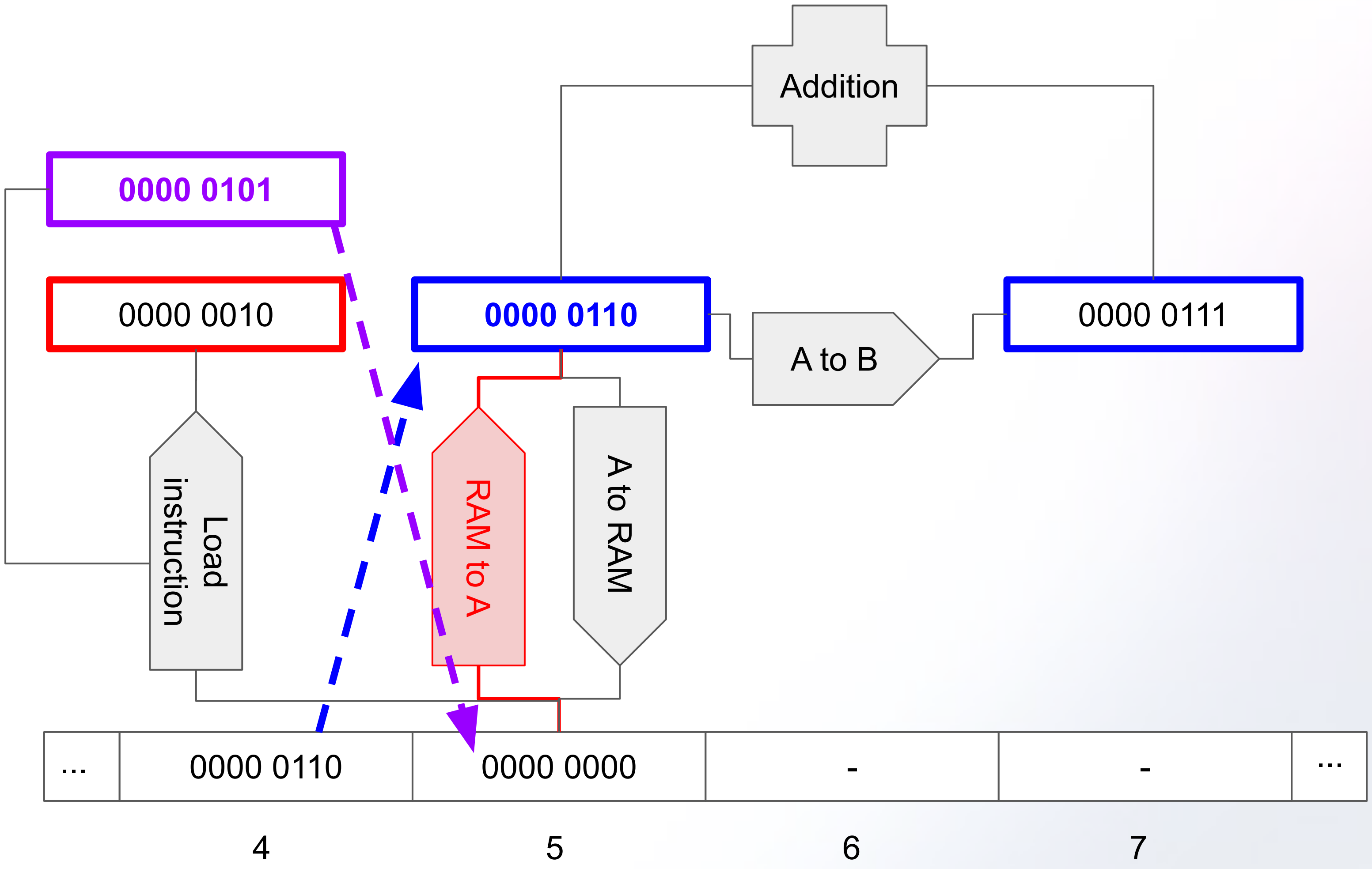


The ALU that corresponds to the value of 2 is again activated.



sourcemind

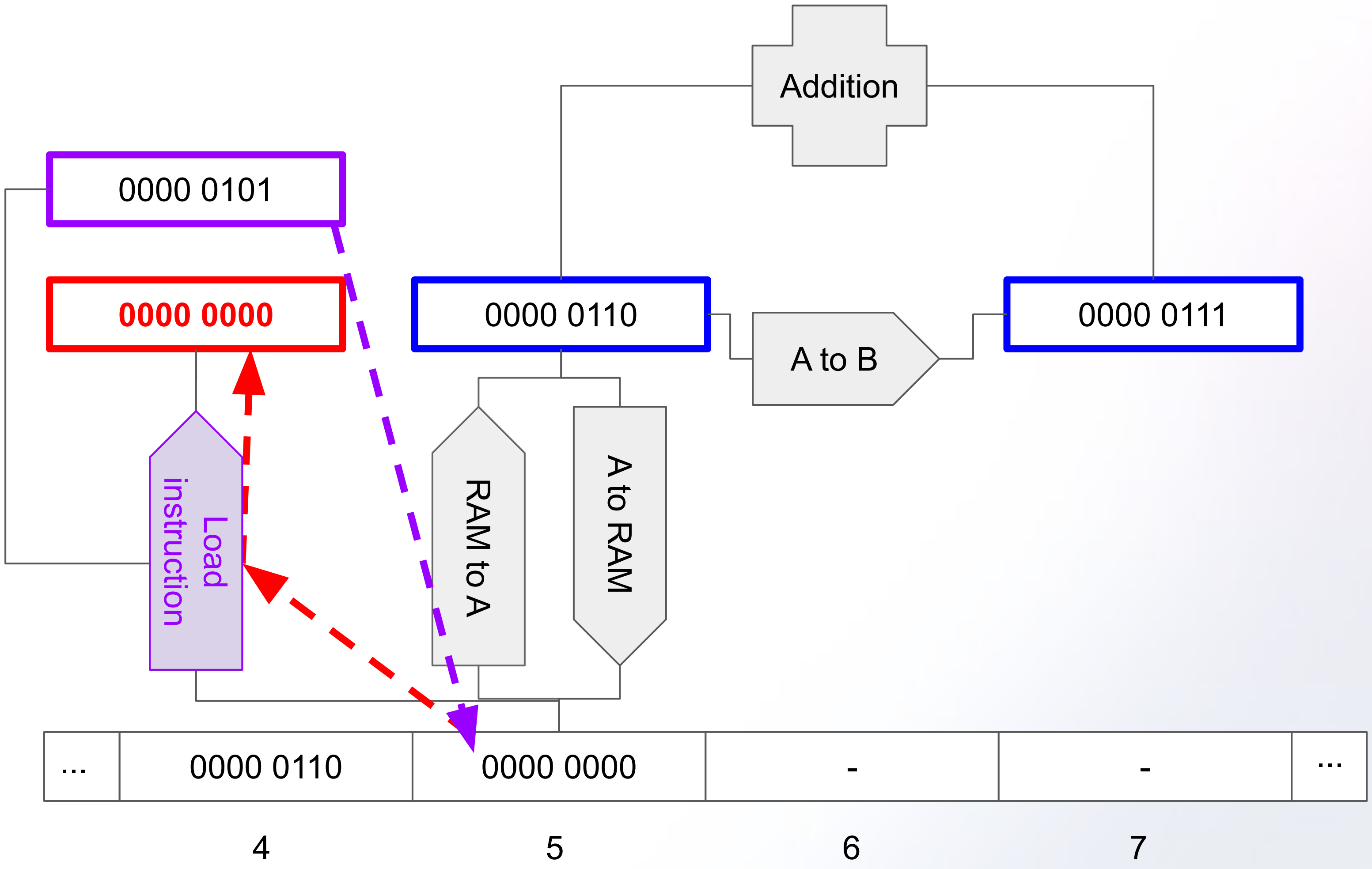
Processor Architecture - Execute



sourcemind

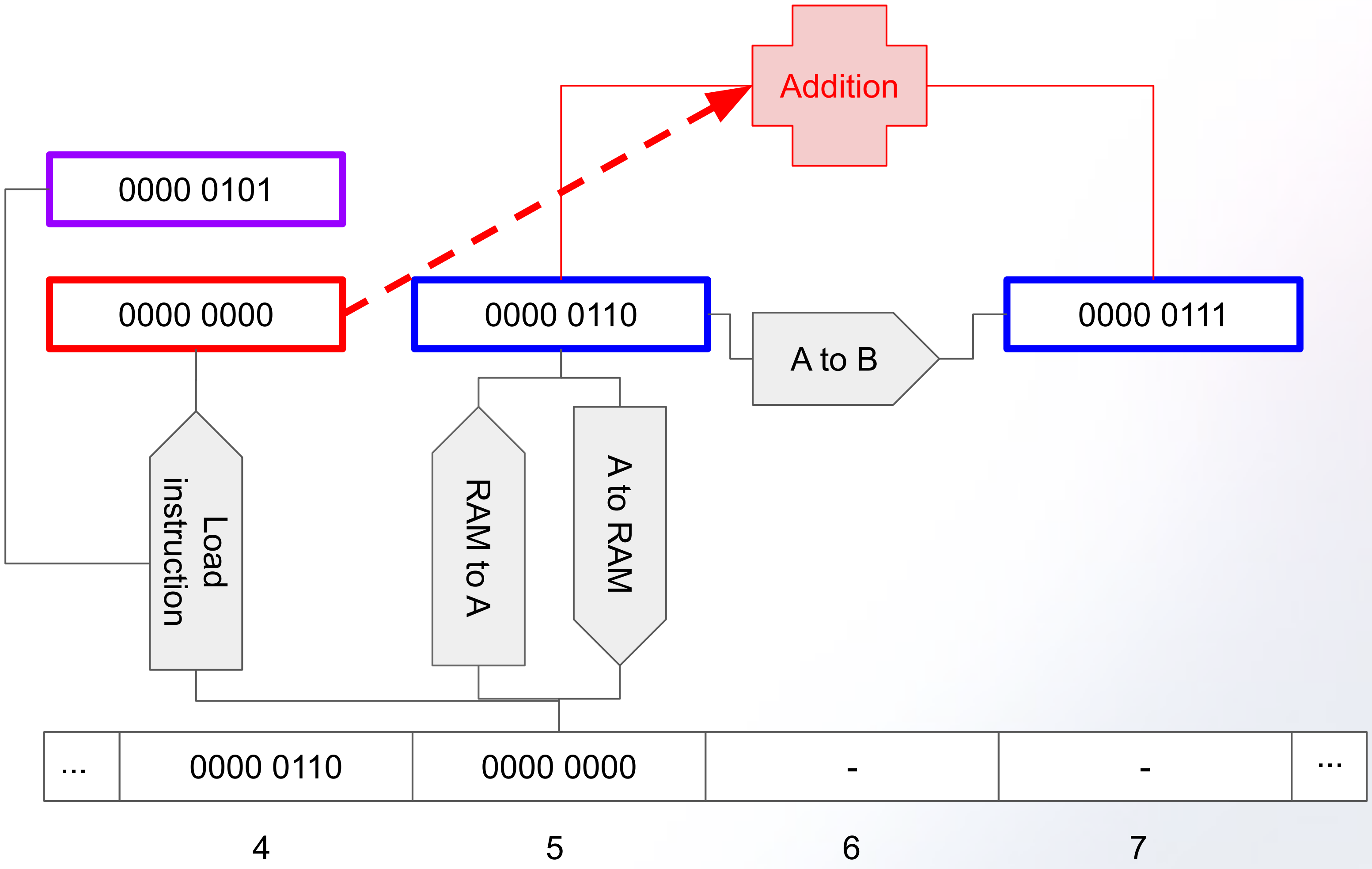
Processor Architecture - Fetch

Fetching the next instruction.



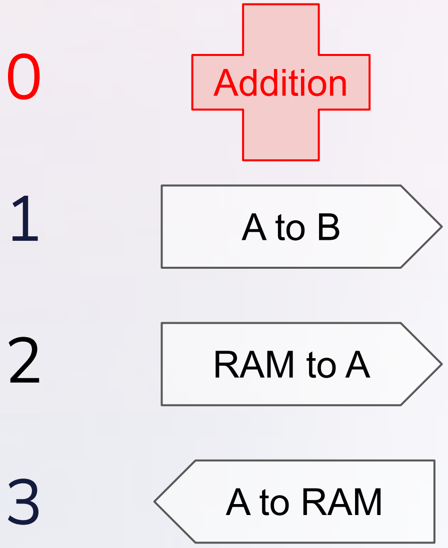
sourcemind

Processor Architecture - Decode



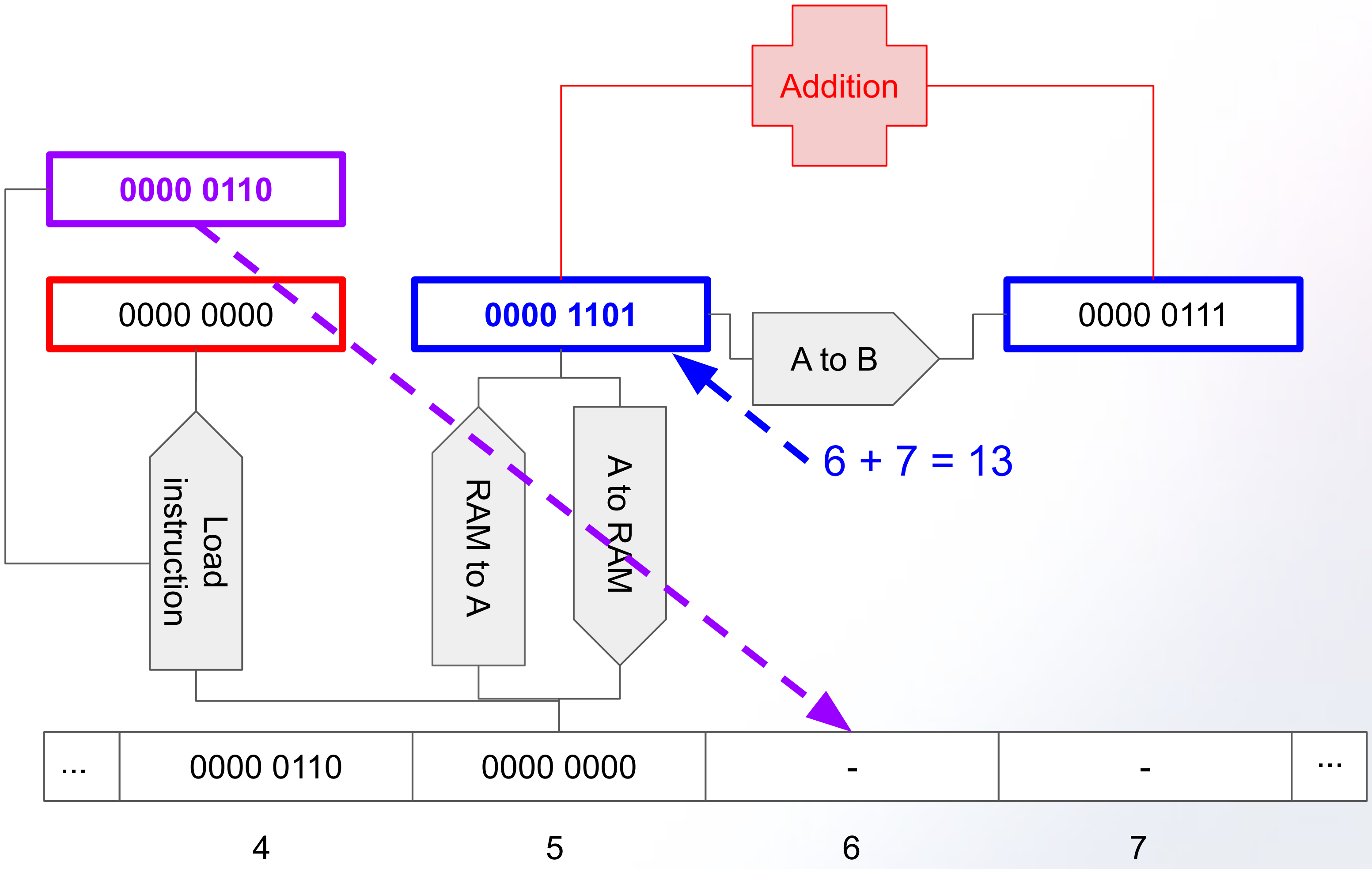
0 is the code for our Addition ALU.

It adds A and B and puts the result in A.



sourcemind

Processor Architecture - Execute



At the end of our program, A contains the result of 6 + 7 which is 13.

The value remains inside the register. It can be moved to another register or RAM for further use.

sourcemind

Processor Architecture

- Processor cycles are measured in Hertz
- Some micro-instructions are executed in single clock tick. Some architectures have complex instructions that take longer and require multiple ticks.
- Processor speeds are therefore sometimes measured in Flops. (Floating point operation per second)
- Overclocking and Underclocking is the practice of modifying the frequency of the cycles for increased performance or reduced heat and power consumption

Assembly

There is also a more legible form of machine code, called assembly. Instead of binary numbers, it uses mnemonics, but is still at the same lowest machine code level.

The pseudo-code below is compatible with our example CPU.

MOV Ax, 0x07	⟷	0000 0010 0000 0111
MOV Bx, Ax	⟷	0000 0001
...		...

There is a one-to-one correspondence between pseudo-assembly instructions and machine code.

sourcemind

Hex Editors and Disassemblers

- Hex editors are programs that opens files and present them as binary in the way they are stored.

- A disassembler converts binary executable machine code into assembly.

- End-user License Agreements (the thing where everyone clicks “accept” without ever reading it) often include a clause that prohibits disassembly of proprietary software.

Hex Editor

hovak@Hovak-UX31A ~/Desktop

File Edit View Search Terminal Help

```
hovak@Hovak-UX31A ~/Desktop $ hexdump ./example.exe
0000000 5a4d 0090 0003 0000 0004 0000 ffff 0000
0000010 00b8 0000 0000 0000 0040 0000 0000 0000
0000020 0000 0000 0000 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0000 0000 0000 00e0 0000
0000040 1f0e 0eba b400 cd09 b821 4c01 21cd 6854
0000050 7369 7020 6f72 7267 6d61 6320 6e61 6f6e
0000060 2074 6562 7220 6e75 6920 206e 4f44 2053
0000070 6f6d 6564 0d2e 0a0d 0024 0000 0000 0000
0000080 85ec a15b e4a8 f235 e4a8 f235 e4a8 f235
0000090 eb6b f23a e4a9 f235 eb6b f255 e4a9 f235
00000a0 eb6b f268 e4bb f235 e4a8 f234 e463 f235
00000b0 eb6b f26b e4a9 f235 eb6b f26a e4bf f235
00000c0 eb6b f26f e4a9 f235 6952 6863 e4a8 f235
00000d0 0000 0000 0000 0000 0000 0000 0000 0000
00000e0 4550 0000 014c 0003 7cc3 4110 0000 0000
00000f0 0000 0000 00e0 010f 010b 0a07 7800 0000
0000100 a600 0000 0000 0000 739d 0000 1000 0000
0000110 9000 0000 0000 0100 1000 0000 0200 0000
0000120 0005 0001 0005 0001 0004 0000 0000 0000
0000130 4000 0001 0400 0000 4f7f 0001 0002 8000
0000140 0000 0004 1000 0001 0000 0010 1000 0000
0000150 0000 0000 0010 0000 0000 0000 0000 0000
0000160 7604 0000 00c8 0000 b000 0000 8958 0000
```

Disassembler

```
hovak@Hovak-UX31A ~/Desktop
File  Edit  View  Search  Terminal  Help
hovak@Hovak-UX31A ~/Desktop $ objdump -d ./example.exe

./example.exe:     file format pei-i386


Disassembly of section .text:

01001000 <.text>:
 1001000:	c8 6f dd 77          	enter  $0xdd6f,$0x77
 1001004:	f0 6b dd 77          	lock imul $0x77,%ebp,%ebx
 1001008:	7d 8f                	jge    0x1000f99
 100100a:	df 77 fd             	fbstp  -0x3(%edi)
 100100d:	d5 df                	aad    $0xdf
 100100f:	77 83                	ja     0x1000f94
 1001011:	78 dd                	js     0x1000ff0
 1001013:	77 1b                	ja     0x1001030
 1001015:	76 dd                	jbe    0x1000ff4
 1001017:	77 cc                	ja     0x1000fe5
 1001019:	d7                   	xlat   %ds:(%ebx)
 100101a:	dd 77 00             	fnsave 0x0(%edi)
 100101d:	00 00                	add    %al,(%eax)
 100101f:	00 ed                	add    %ch,%ch
 1001021:	d2 3d 77 00 00 00    	sarb   %cl,0x77
 1001027:	00 23                	add    %ah,(%ebx)
```

Additional Links

Two’s Complement
https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html

YouTube counter on Gangnam Style
https://www.exploringbinary.com/gangnam-style-video-overflows-youtube-counter/

ASCII Table
https://www.asciitable.com/

Color Picker
https://htmlcolorcodes.com/color-picker/

Motherboard
https://www.youtube.com/watch?v=b2pd3Y6aBag

Registers and RAM
https://www.youtube.com/watch?v=fpnE6UAfbtU

Storage
https://www.youtube.com/watch?v=TQCr9RV7twk

Magnetic storage
https://www.youtube.com/watch?v=wteUW2sL7bc

Zone bit recording
https://en.wikipedia.org/wiki/Zone_bit_recording

32-bit x86 Architecture
http://flint.cs.yale.edu/cs421/papers/x86-asm/asm.html

CPU manufacturing process
https://www.youtube.com/watch?v=qm67wbB5GmI

Addition ALU
https://www.youtube.com/watch?v=1I5ZMmrOfnA
https://www.electronics-tutorials.ws/combination/comb_7.html