ARM Processors
ITMC301
L1 Introduction

By: Dr. Abdussalam Nuri Baryun

Mobile Computing Department, Faculty of Advance Technology, University of Tripoli

Course Evaluation

- 20% Quiz
- 30% mid-test
- 50% Final

الكتب المقررة:

- ARM documents (www.arm.com)
- A., Sloss, et al., ARM System Developer's Guide.
- Course handed sheets.

Used ARM processors


Smartphones


Automotive and ADAS Systems


Server and networking


Wearables


Tablets and Readers


Set-top and satellite receivers

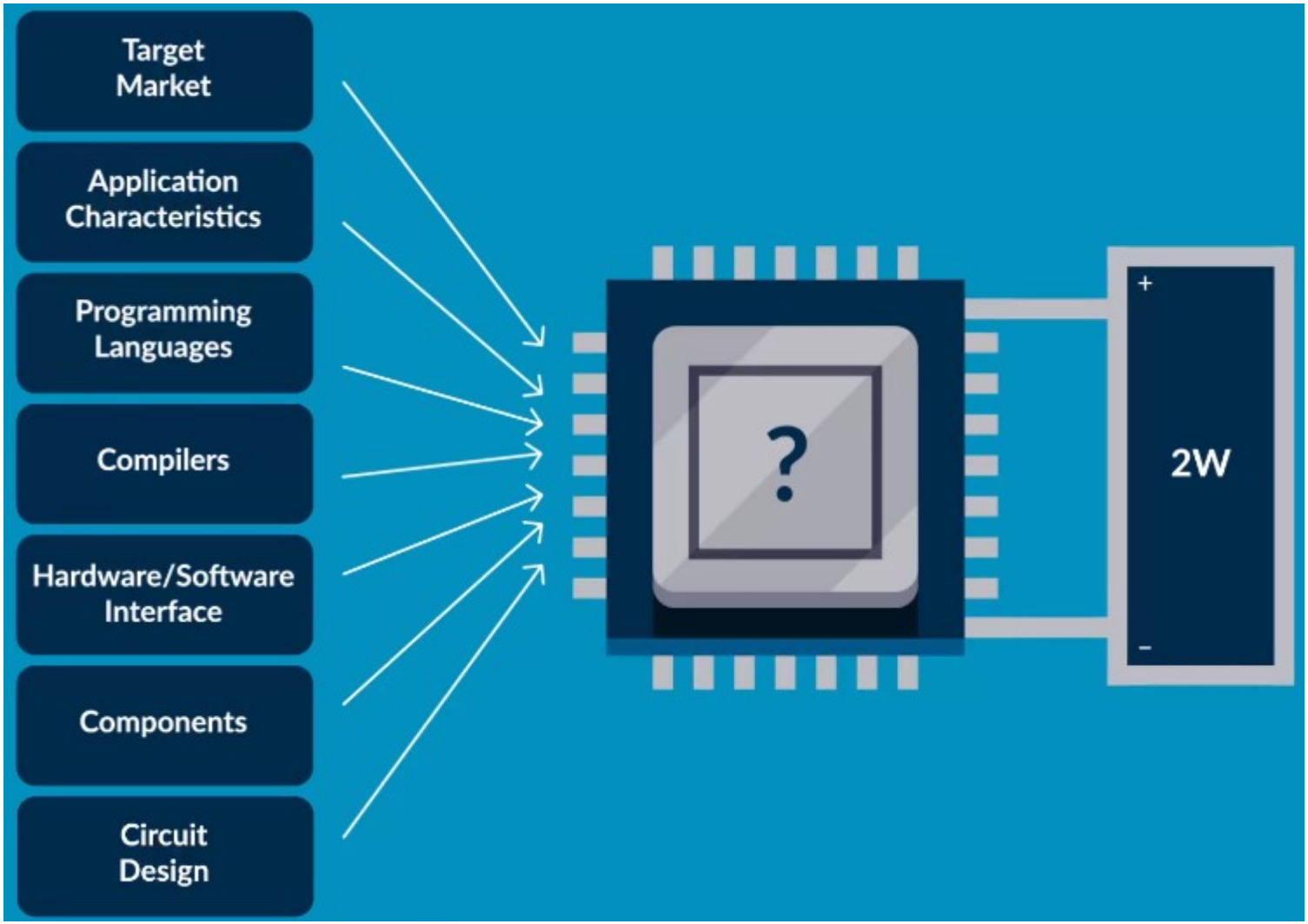

Home gateways


Robotics

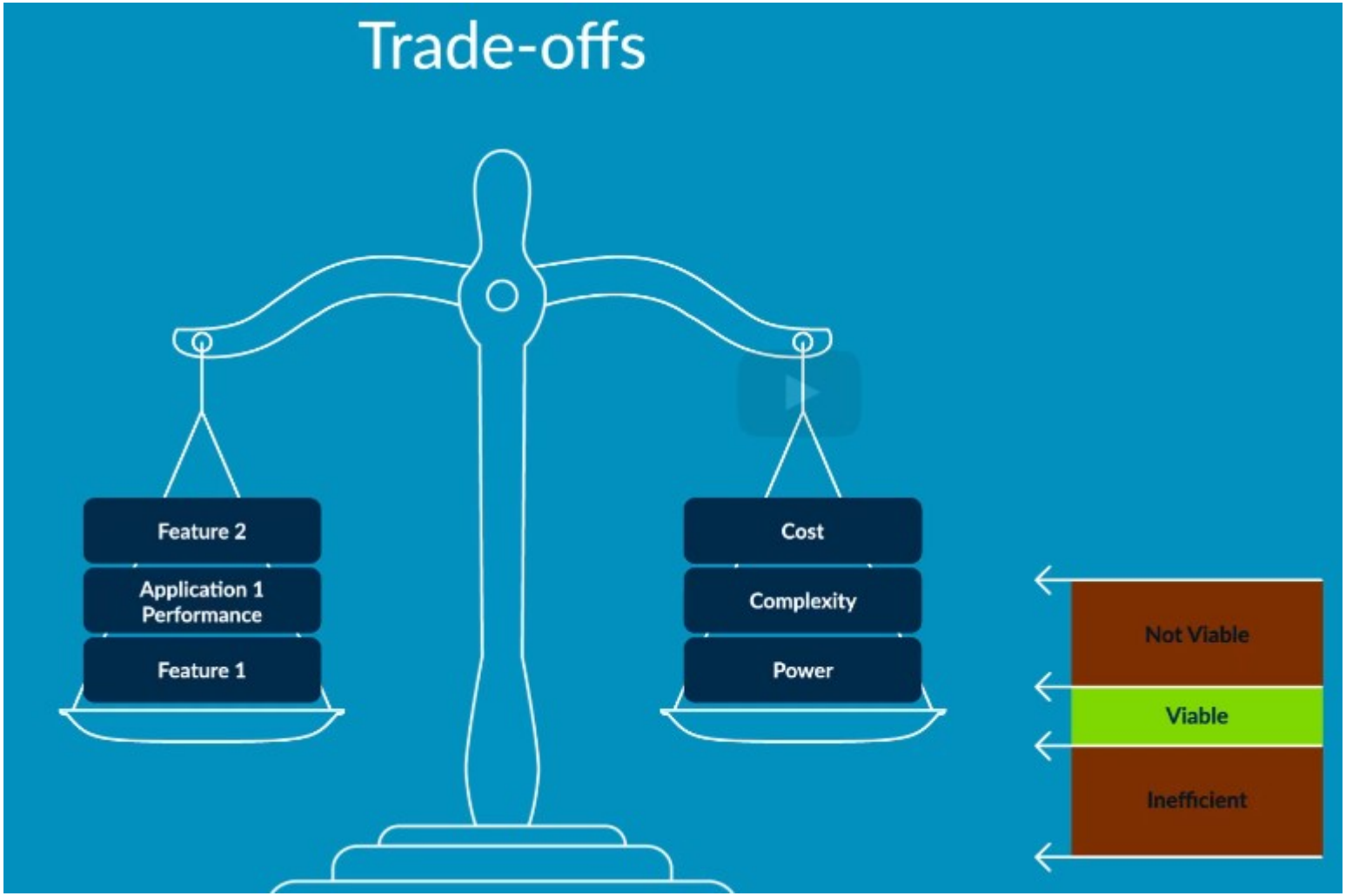
Definitions

- Processor, and Microprocessor:
- Microcontroller:
- Architecture, and Microarchitecture:
- Computer, and Microcomputer:
- Embedded System, SoC, and NoC
- MEMS

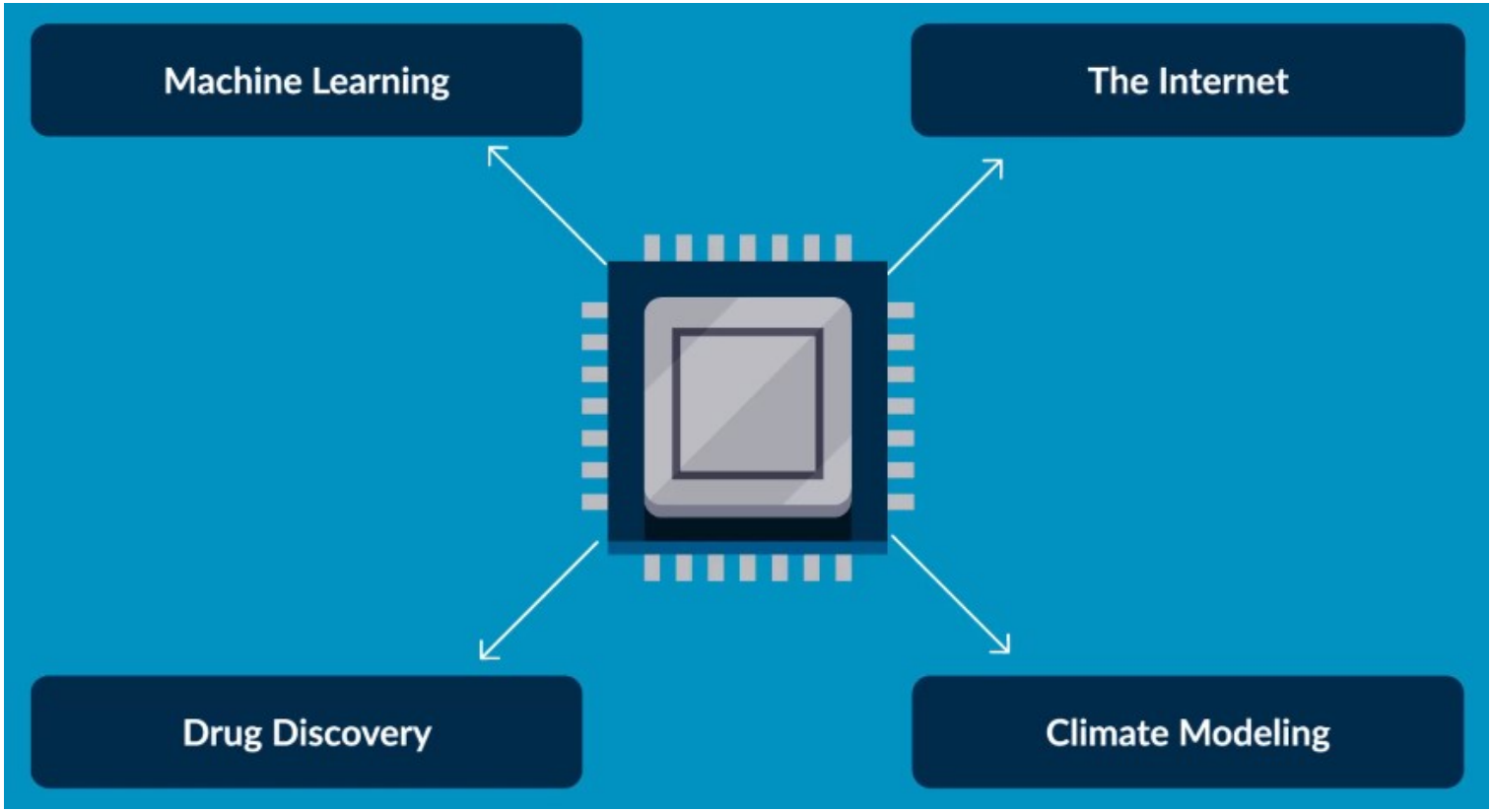
Microprocessor Design Factors



Design Trade-offs



High demand Applications



Processor Resources

- A general-purpose processor is a finite-state automaton that executes instructions held in a memory. The state of the system is defined by the values held in the memory locations together with the values held in certain registers within the processor itself.
- Each instruction defines a particular way the total state should change and it also defines which instruction should be executed next.

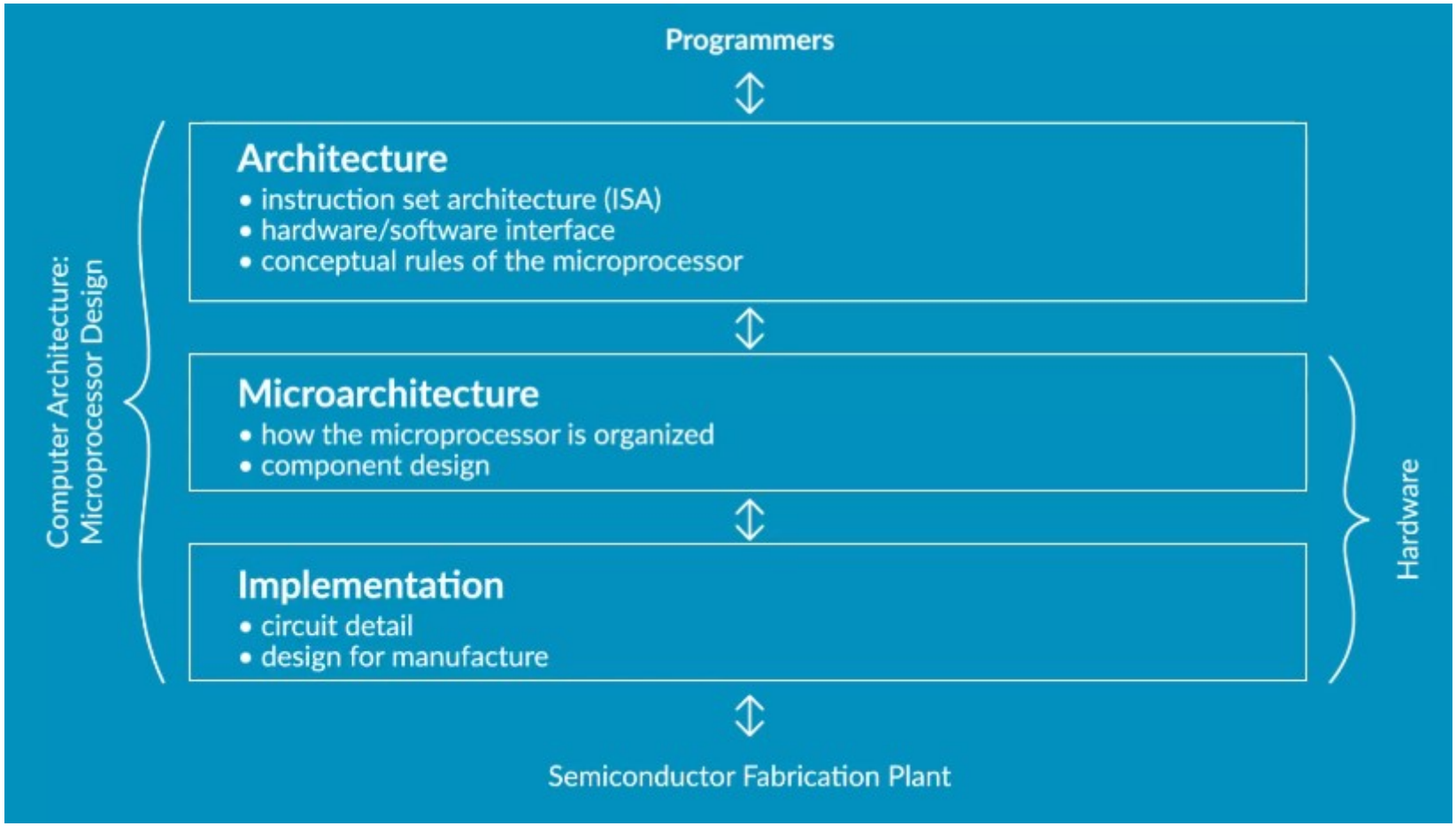
What processors do?

- It is a common misconception that computers spend their time computing, that is, carrying out arithmetic operations on user data.
- In practice they spend very little time 'computing' in this sense. Although they do a fair amount of arithmetic, most of this is with addresses in order to locate the relevant data items and program routines. Then, having found the user's data, most of the work is in moving it around rather than processing it in any transformational sense.

What processors spend time on?

Typical dynamic instruction usage.

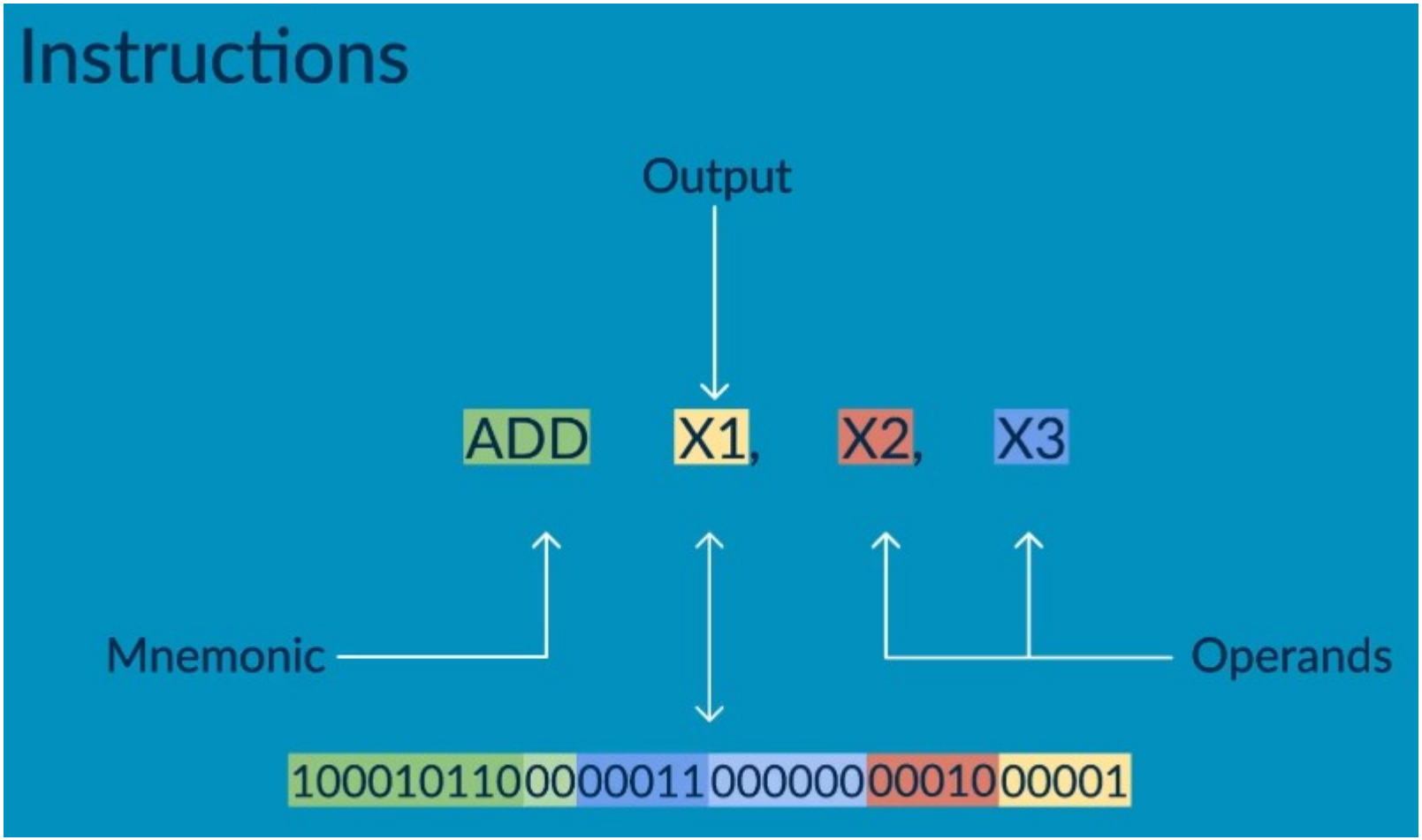
Instruction type	Dynamic usage
Data movement	43%
Control flow	23%
Arithmetic operations	15%
Comparisons	13%
Logical operations	5%
Other	1%


Programmers
Architecture
• instruction set architecture (ISA)
• hardware/software interface
• conceptual rules of the microprocessor
Microarchitecture
• how the microprocessor is organized
• component design
Implementation
• circuit detail
• design for manufacture
Semiconductor Fabrication Plant
Computer Architecture: Microprocessor Design
Hardware

Instruction Set Architecture (ISA)

- ISAs are related to microprocessor
- Instruction Types
  - Arithmetic
  - Load and Store
  - Branch

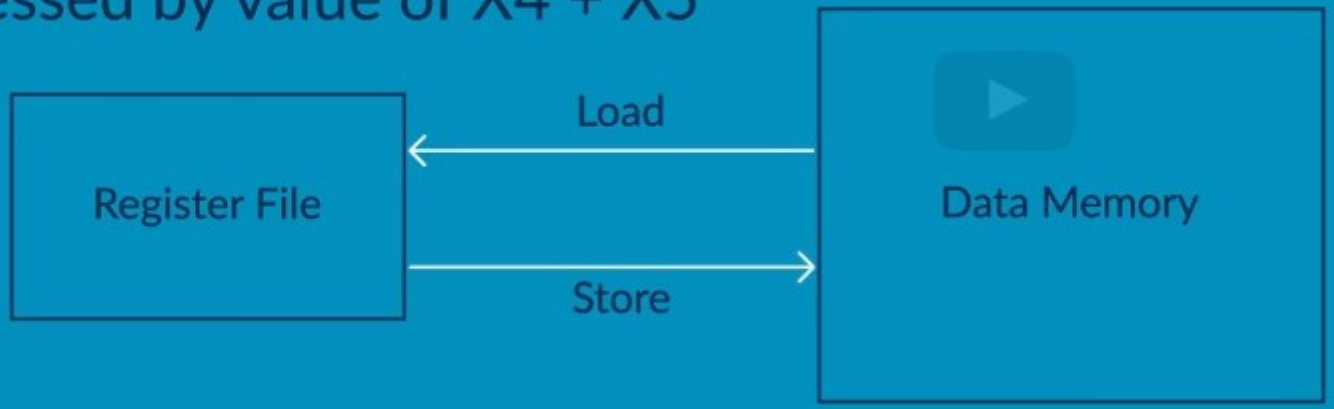
How Do You Tell a Microprocessor What to Do



Load and Store Instructions

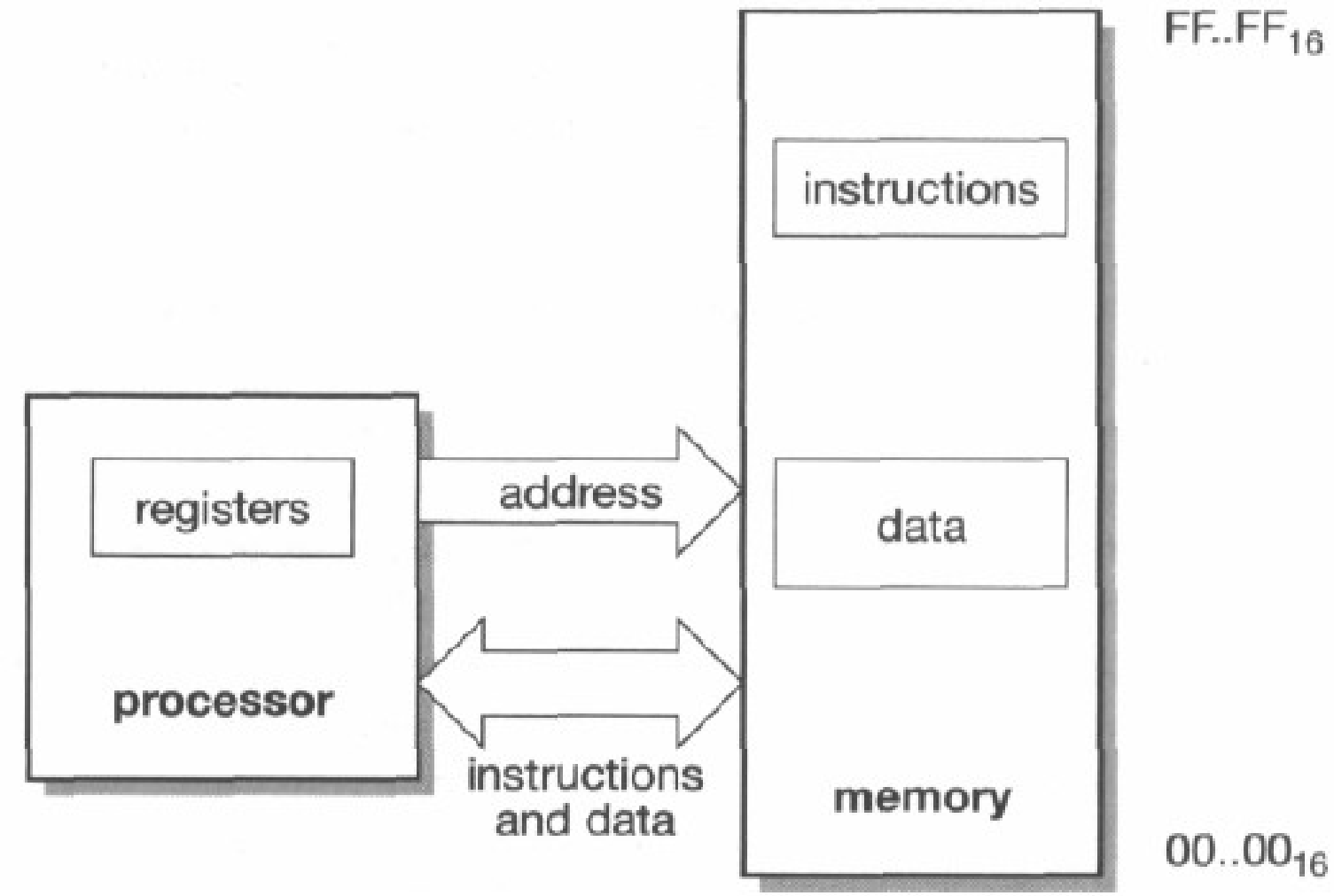
Set X3 to value of data memory addressed by value of X4 + X5

LDR X3, [X4, X5]



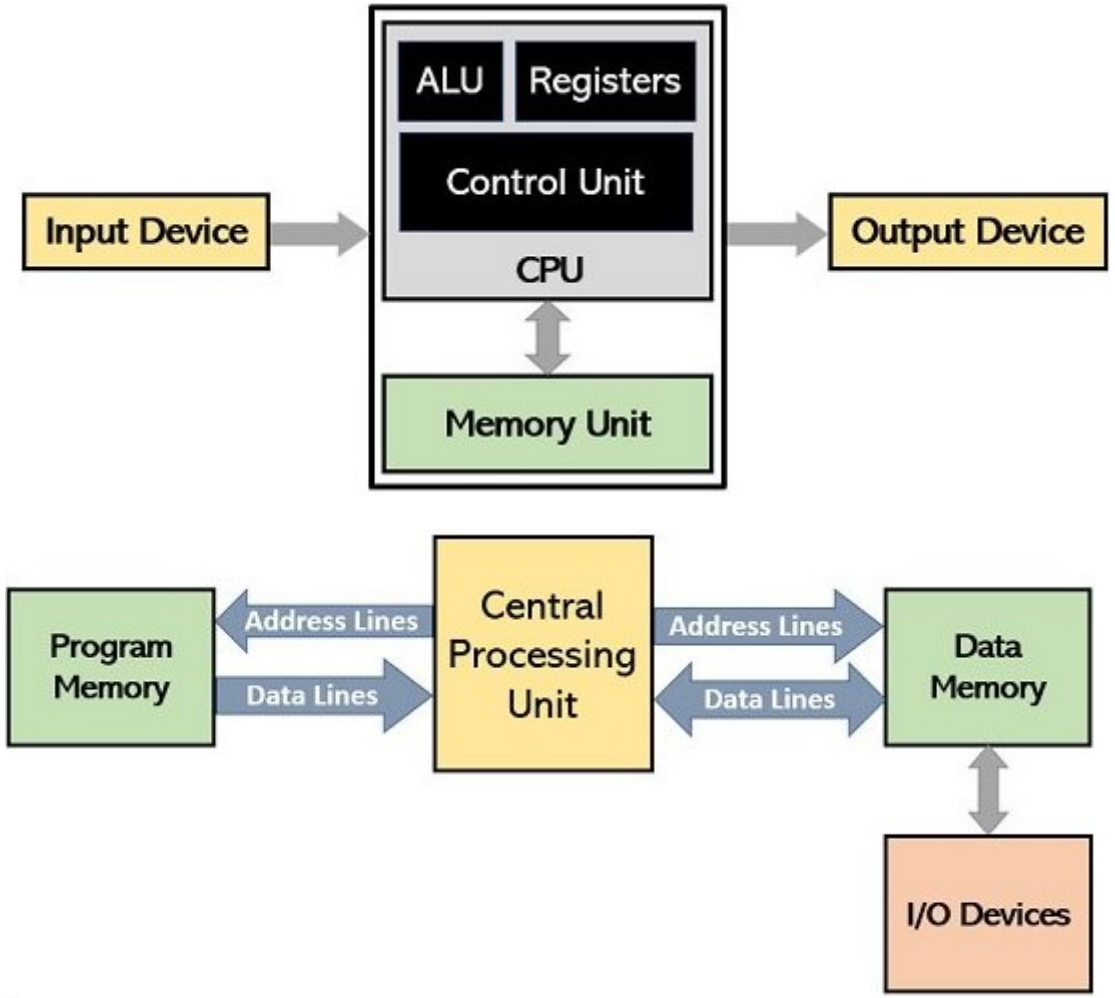
Set data memory addressed by X5 + 6 to value of X17

STR X17, [X5, #6]



The state in a stored-program digital computer.

Two Principle Processing Architectures

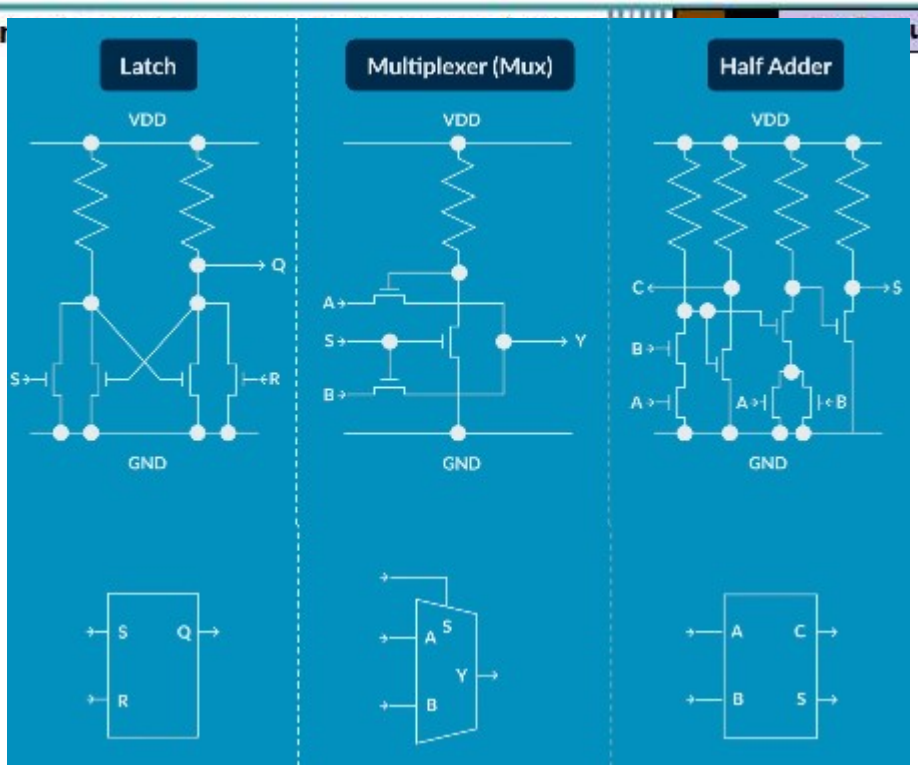


When designing a digital system

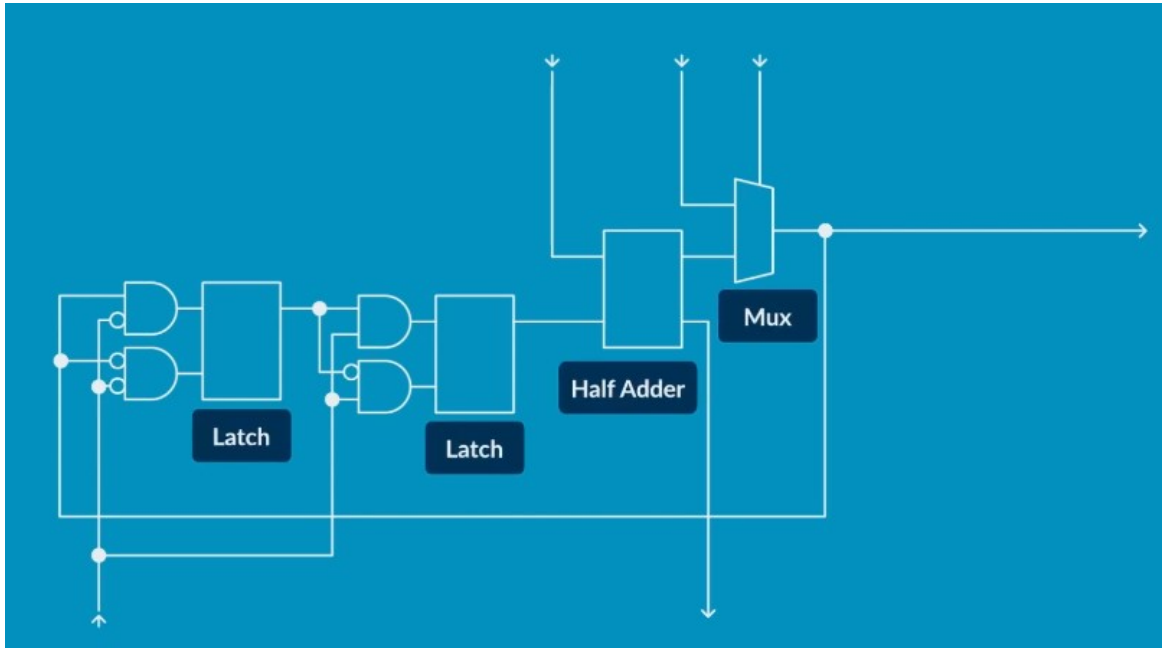
We must keep everything synchronized to control behavior.

- To do this, we use a "Clock Signal," which is a wire in the circuit whose signal cycles between zero and one. We measure the rate in Hertz.
- For example, if you hear a processor has a clock speed of two Gigahertz,it means 2 billion ones and zeroes per second.
- The maximum speed of the clock signal is determined by the longest, and therefore slowest, path in the circuit between 2 clocked flip-flops.
- This is referred to as the "Critical Path." The signal must have time to propagate all the way along the critical path before the clock cycle completes.

Digital Logic Systems

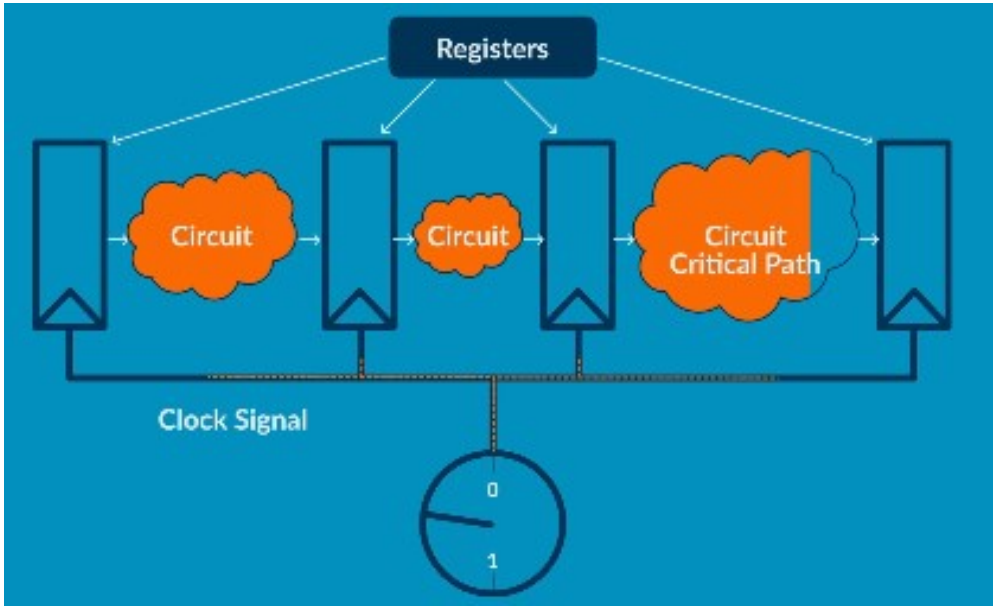


Synchronised



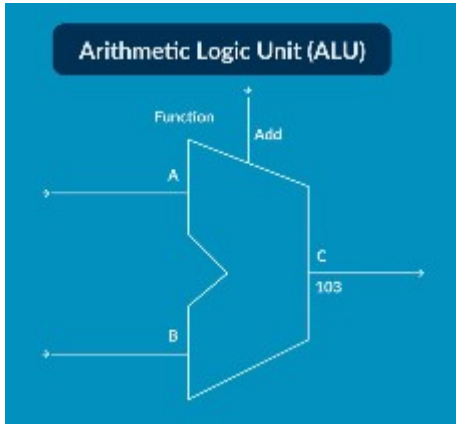
Critical path

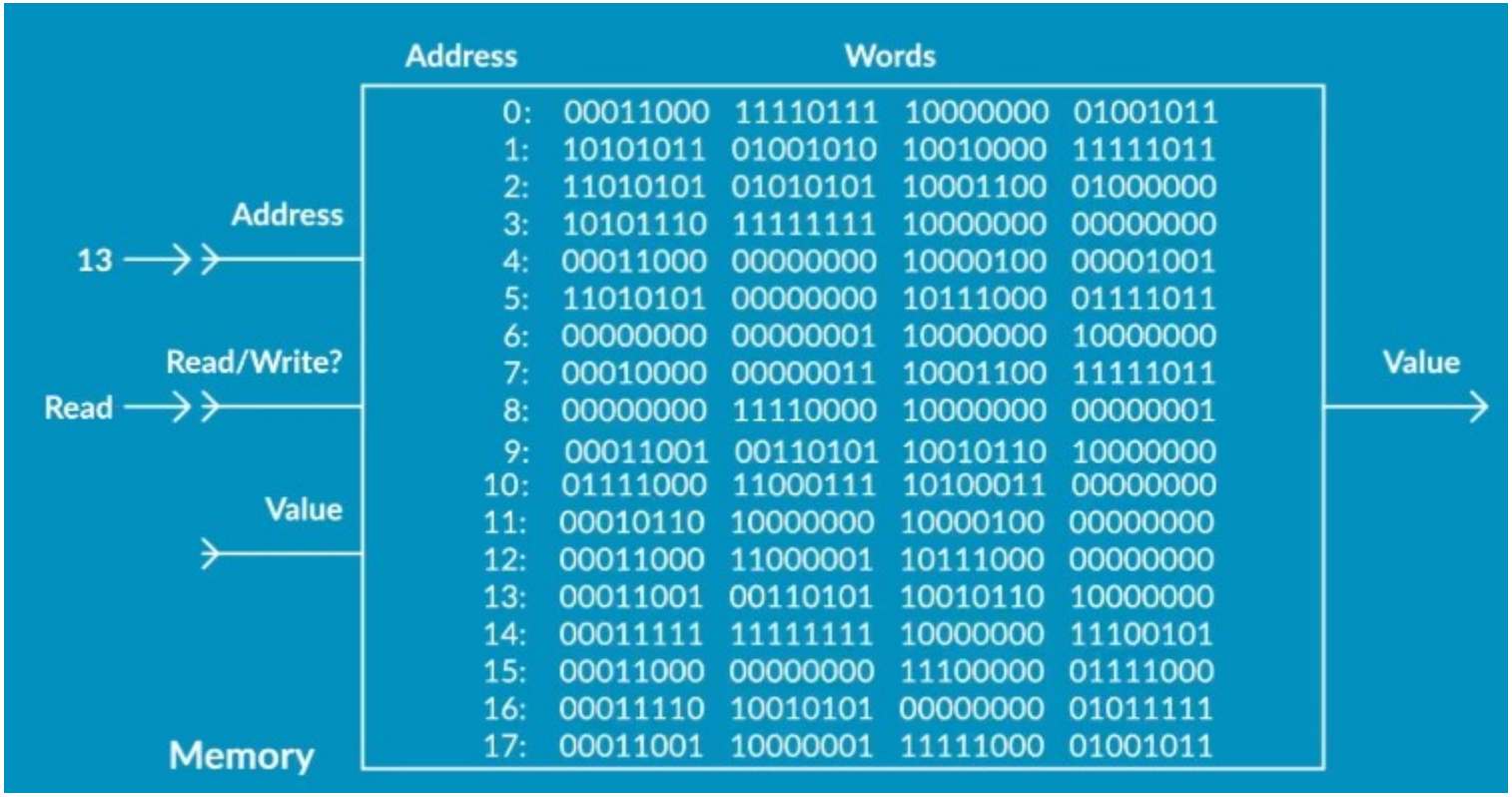
- The maximum speed of the clock signal is determined by the longest, and therefore slowest, path in the circuit between 2 clocked flip-flops



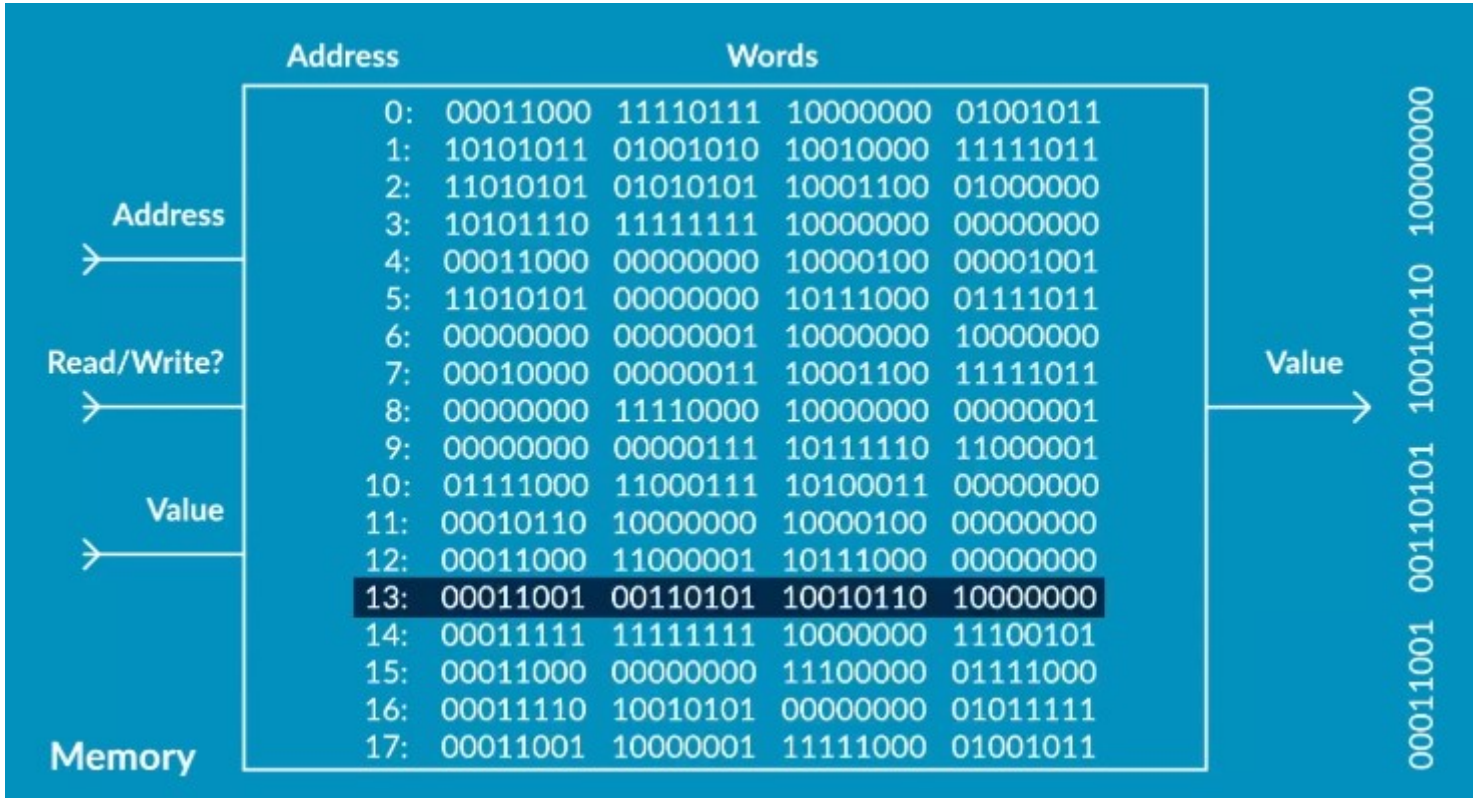
Processing needs Memory

- In addition to logic, a microprocessor needs memory.
- Memory is organized as arrays of memory cells that are able to store many "Words" of data.
- A specific word, commonly 32 bits in length, can be accessed by specifying its "Address."
- Each address is a number that indicates the location in the memory that should be read or written.
- Memory cells range from hundreds of bits to millions of bits in size, but larger ones are slower to access, as signals and their long internal wires take longer to propagate.
- For that reason, almost all microprocessors include at least two types for storing data: a big slow "Data Memory," and a small fast memory called a "Register File."
- In reality, the "Data Memory" may be implemented using many different sizes of memory, as well as storing data in memory, we also use some memory to store the instructions.
- We need a way to keep track of which instruction we will fetch next, so we have a "Program Counter."


Arithmetic Logic Unit (ALU)
Function
Add
A
B
C
103

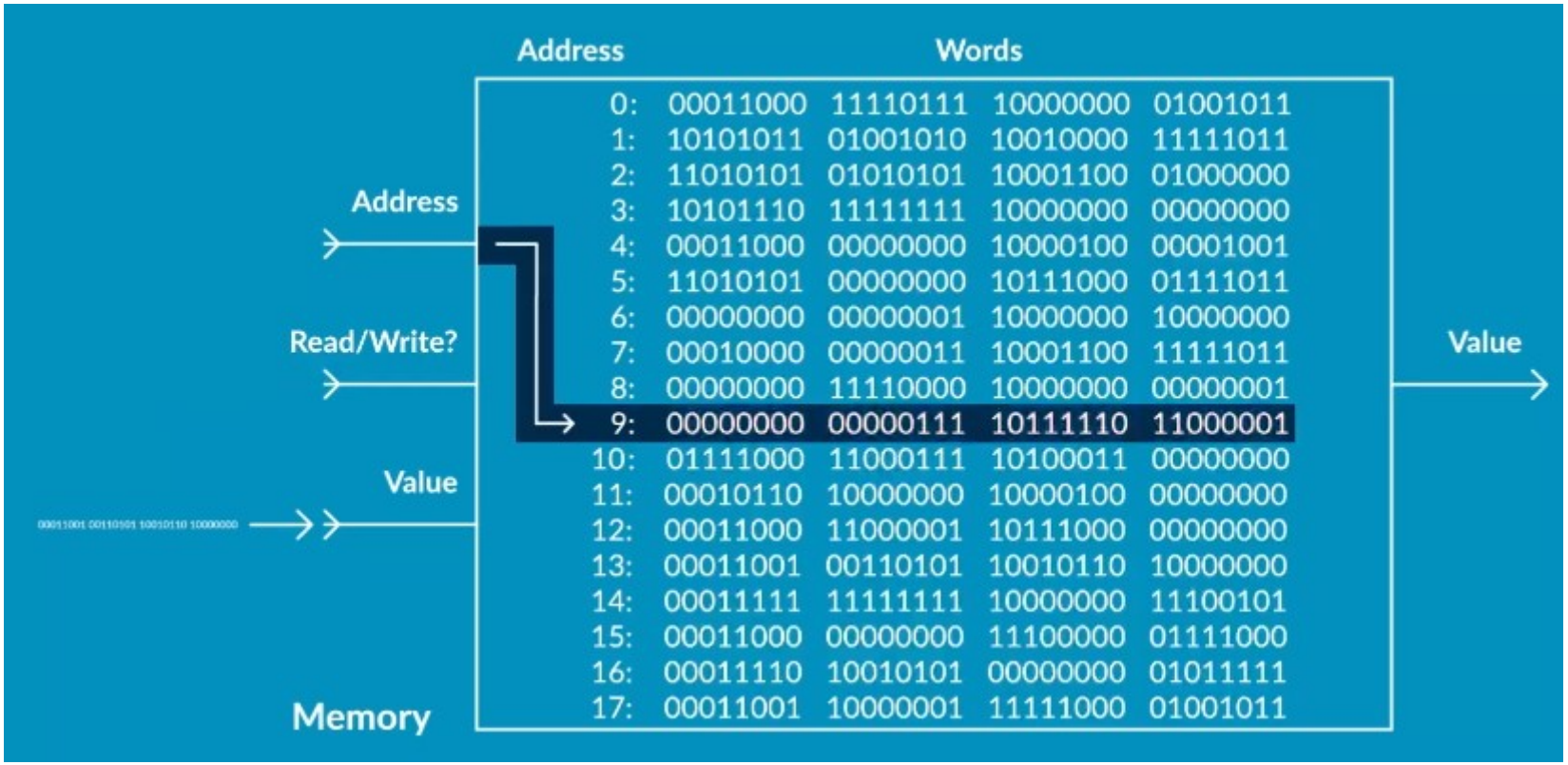

Address
Words
0: 00011000 11110111 10000000 01001011
1: 10101011 01001010 10010000 11111011
2: 11010101 01010101 10001100 01000000
3: 10101110 11111111 10000000 00000000
4: 00011000 00000000 10000100 00001001
5: 11010101 00000000 10111000 01111011
6: 00000000 00000001 10000000 10000000
7: 00010000 00000011 10001100 11111011
8: 00000000 11110000 10000000 00000001
9: 00011001 00110101 10010110 10000000
10: 01111000 11000111 10100011 00000000
11: 00010110 10000000 10000100 00000000
12: 00011000 11000001 10111000 00000000
13: 00011001 00110101 10010110 10000000
14: 00011111 11111111 10000000 11100101
15: 00011000 00000000 11100000 01111000
16: 00011110 10010101 00000000 01011111
17: 00011001 10000001 11111000 01001011
Address
13
Read/Write?
Read
Value
Value
Memory

Read


Address
Words
0: 00011000 11110111 10000000 01001011
1: 10101011 01001010 10010000 11111011
2: 11010101 01010101 10001100 01000000
3: 10101110 11111111 10000000 00000000
4: 00011000 00000000 10000100 00001001
5: 11010101 00000000 10111000 01111011
6: 00000000 00000001 10000000 10000000
7: 00010000 00000011 10001100 11111011
8: 00000000 11110000 10000000 00000001
9: 00000000 00000111 10111110 11000001
10: 01111000 11000111 10100011 00000000
11: 00010110 10000000 10000100 00000000
12: 00011000 11000001 10111000 00000000
13: 00011001 00110101 10010110 10000000
14: 00011111 11111111 10000000 11100101
15: 00011000 00000000 11100000 01111000
16: 00011110 10010101 00000000 01011111
17: 00011001 10000001 11111000 01001011
Address
Read/Write?
Value
Memory
Value
00011001 00110101 10010110 10000000

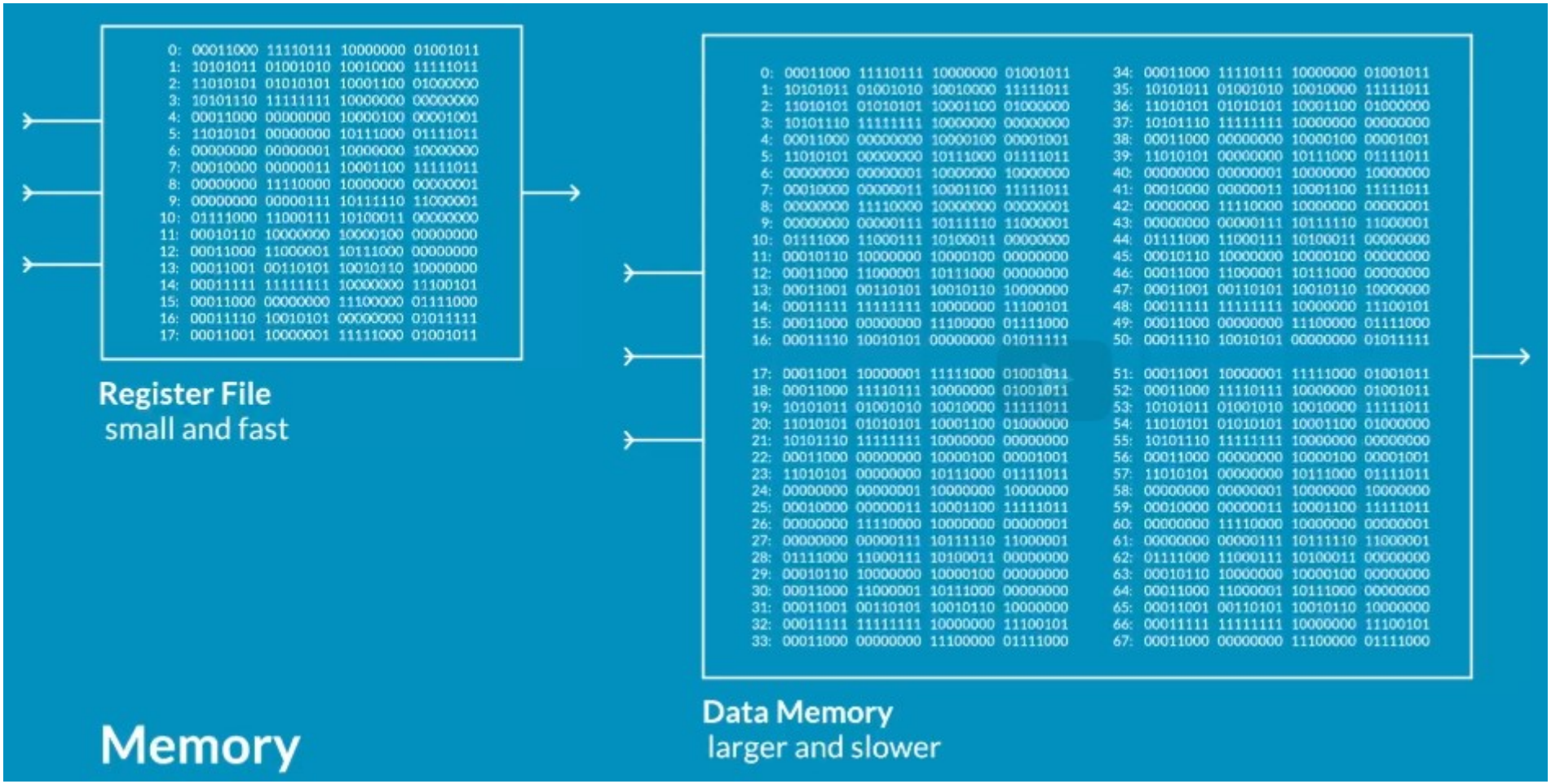
- larger memory ones are slower to access,
- as signals and their long internal wires take longer to propagate.
- For that reason, almost all microprocessors include at least two types for storing data:
- a big slow "Data Memory," and a small fast memory called a "Register File."
- In reality, the "Data Memory" may be implemented using many different sizes of memory,

Write


Address
Words
0: 00011000 11110111 10000000 01001011
1: 10101011 01001010 10010000 11111011
2: 11010101 01010101 10001100 01000000
3: 10101110 11111111 10000000 00000000
4: 00011000 00000000 10000100 00001001
5: 11010101 00000000 10111000 01111011
6: 00000000 00000001 10000000 10000000
7: 00010000 00000011 10001100 11111011
8: 00000000 11110000 10000000 00000001
9: 00000000 00000111 10111110 11000001
10: 01111000 11000111 10100011 00000000
11: 00010110 10000000 10000100 00000000
12: 00011000 11000001 10111000 00000000
13: 00011001 00110101 10010110 10000000
14: 00011111 11111111 10000000 11100101
15: 00011000 00000000 11100000 01111000
16: 00011110 10010101 00000000 01011111
17: 00011001 10000001 11111000 01001011
Address
Read/Write?
Value
Value
Memory

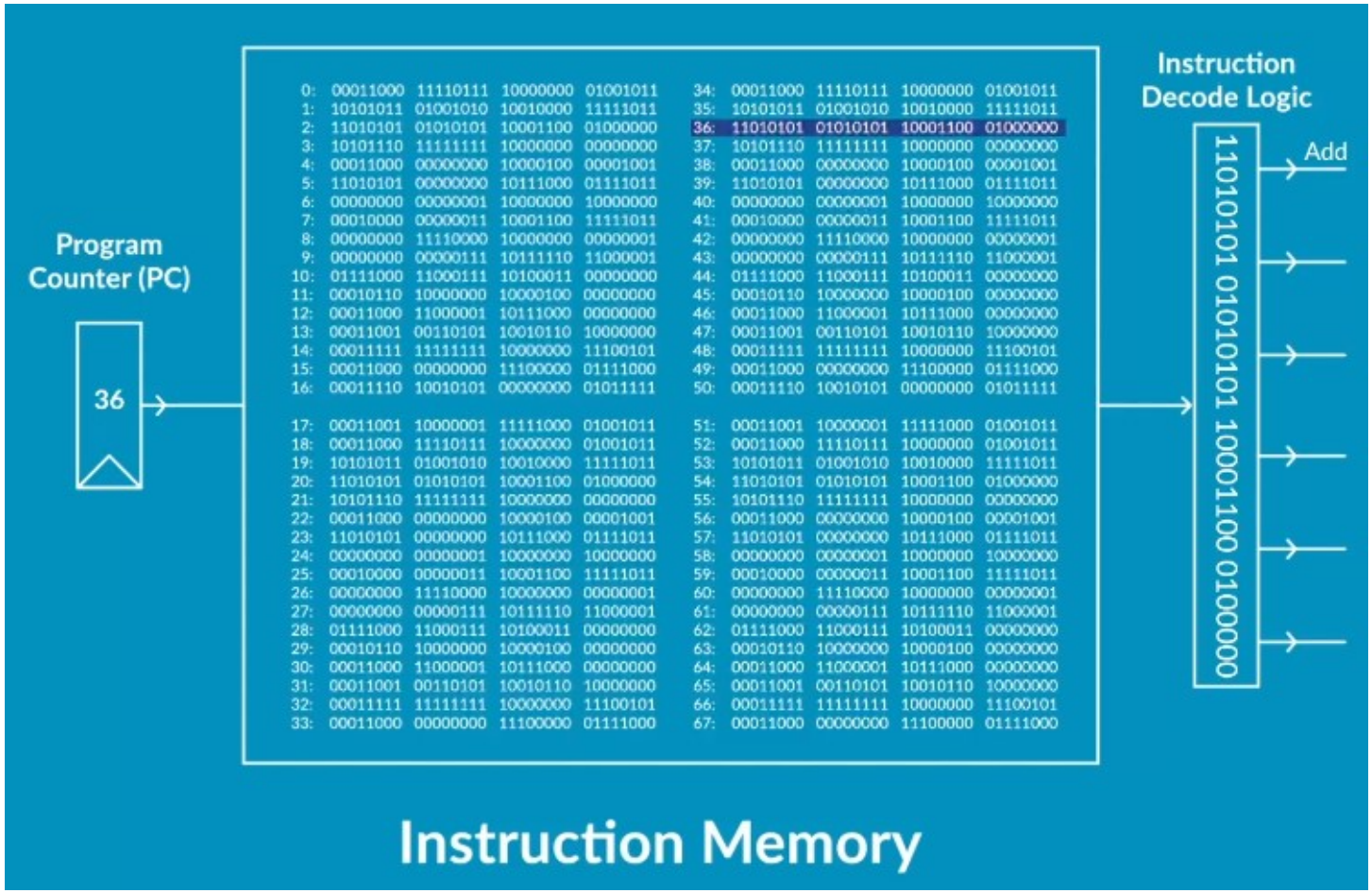
Computing Memory

- We need Register File, Data Memory, Instruction Memory.



Program Counter (PC)

A Register to keep track of which instruction to be fetched next in process, it stores the memory address of the next instruction to be accessed



Memory Size Measure

Most modern machines are byte-addressable (8-bit)

If you have more specialized hardware, (embedded microcontrollers, etc) that are word addressable (16-bit, 32-bit), then you are correct that you would multiply

memory locations possible 2^n * (word-size in bits) / (8) = # of bytes.

Normally, a processor with a 32-bit architecture can only address 4 GiB of physical memory at any given time (2^{32} = 4294967296). Each byte of physical memory has its own address.

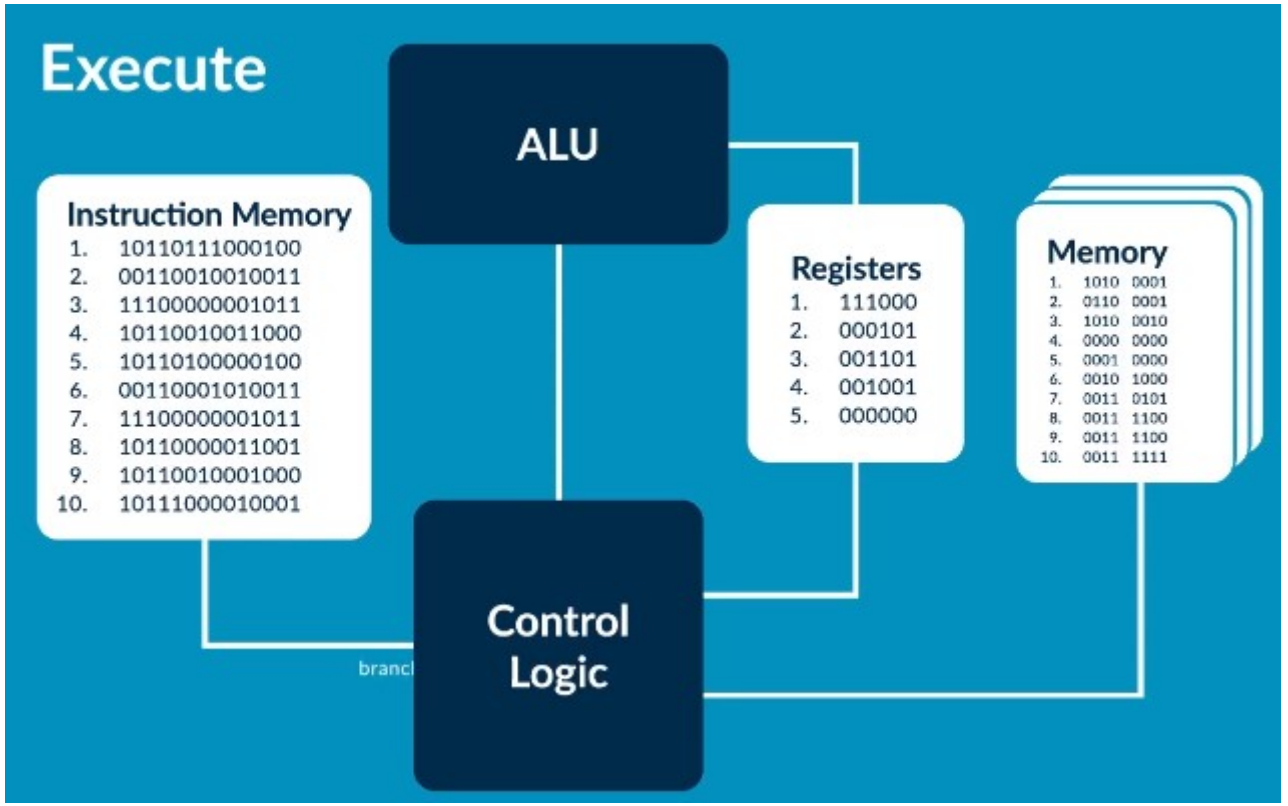
To determine the ammount of memory that can be addressed we need to know three things.

1. The size of the smallest addressable unit of memory. On pretty much all current general purpose computers this is the 8-bit byte. Computers do not generally address memory in bits.
2. The usable size the phsyical address. This may be the same as the data word size of the processor but it often isn't.
3. Whether any memory address ranges need to be used for things other than memory. Most systems place IO devices in the memory map reducing the ammount of space available for regular memory (sometimes significantly so).

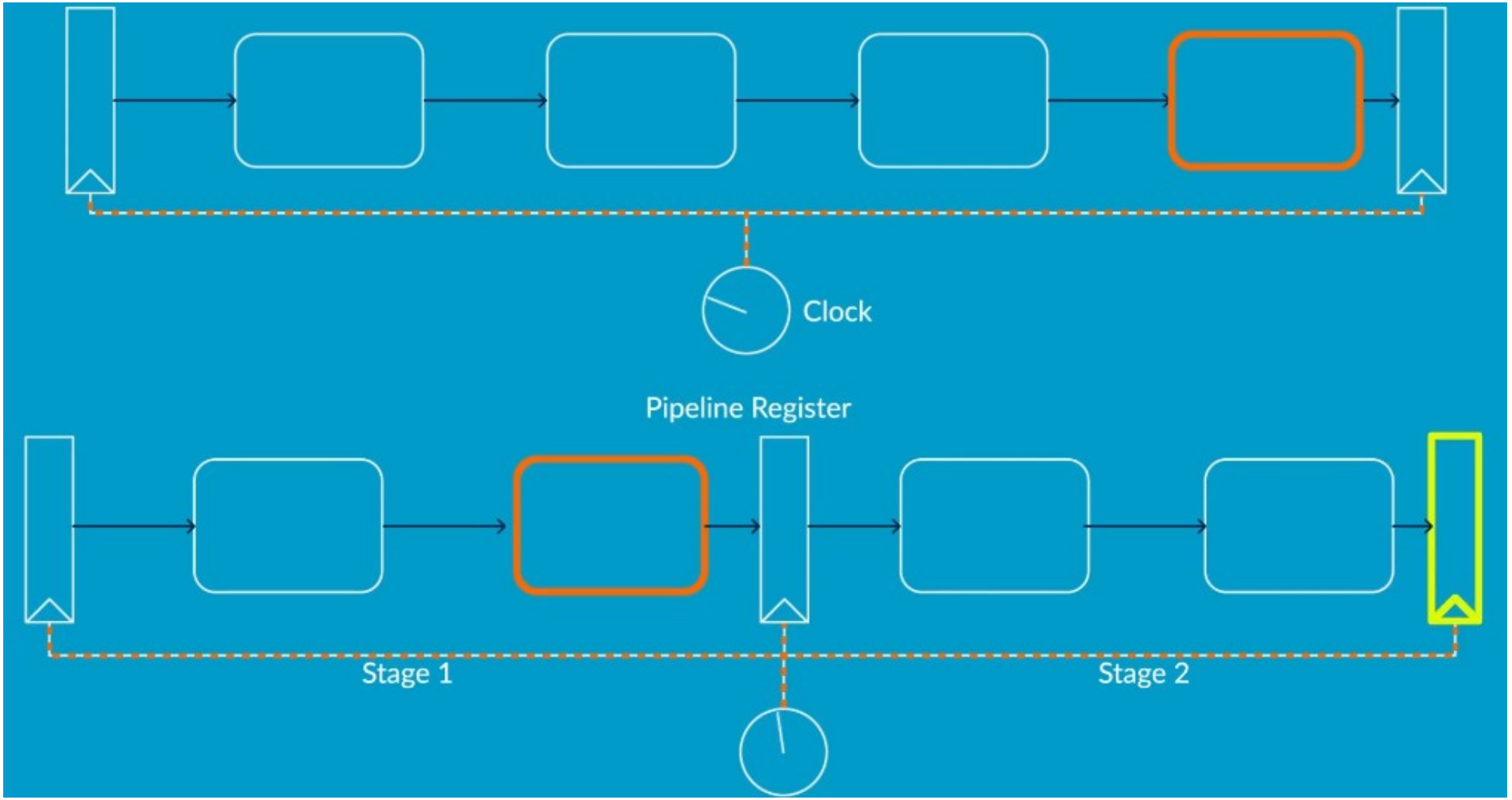
What Are the Basic Components of a Microprocessor

What Is the Fetch-Execute Cycle

- Fetch – Decode - Execute



Pipelining

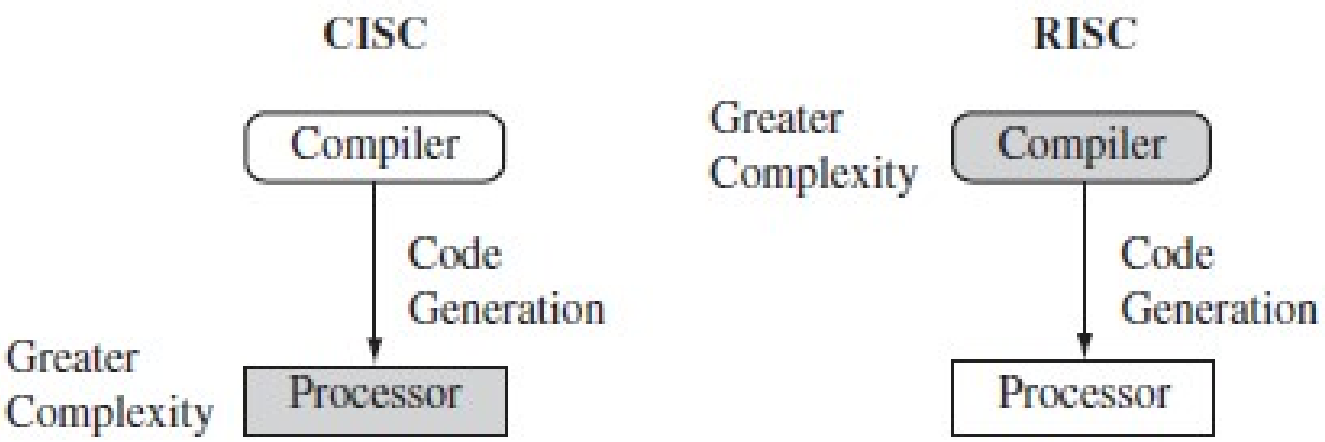


Simple Processor

A simple form of processor can be built from a few basic components:

- **A program counter (PC) register that is used to hold the address of the current** instruction; a single register called an **accumulator (ACC) that holds a data value while it is** worked upon;
- **An arithmetic-logic unit (ALU) that can perform a number of operations on** binary operands, such as add, subtract, increment, and so on;
- **An instruction register (IR) that holds the current instruction while it is executed;**
- **Control Logic Unite (CLU)**: instruction decode and control logic that employs the above components to achieve the desired results from each instruction.

- Reduced Instruction Set Computer (RISC)
- Complex Instruction Set Computer (CISC)
- RISC is a design philosophy aimed at delivering simple but powerful instructions that execute within a single cycle at a high clock speed.
- Reducing the complexity of instructions performed by the hardware because it is easier to provide greater flexibility and intelligence in software.
- A RISC design places greater demands on the compiler.



CISC vs. RISC. CISC emphasizes hardware complexity. RISC emphasizes compiler complexity.

CISC	*RISC*
Many instructions	Few instructions
Instructions have varying lengths	Instructions have fixed lengths
Instructions execute in varying times	Instructions execute in 1 or 2 bus cycles
Many instructions can access memory	Few instructions can access memory • Load from memory to a register • Store from register to memory
In one instruction, the processor can both • read memory and • write memory	No one instruction can both read and write memory in the same instruction
Fewer and more specialized registers. • some registers contain data, • others contain addresses	Many identical general purpose registers
Many different types of addressing modes	Limited number of addressing modes • register, • immediate, and • indexed.

What is ARM

- Acorn RISC Machine
- Advanced RISC Machines (ARM)
- The important concept of the *Reduced Instruction Set Computer (RISC)*
- System-on-Chip (SoC)
- The ARM core uses a RISC architecture.

ARM Features

The ARM architecture incorporated a number of features from the Berkeley RISC design, but a number of other features were rejected. Those that were used were:

- a load-store architecture;
- fixed-length 32-bit instructions;
- 3-address instruction formats.

RISC Features rejected

The RISC features that were rejected by the ARM designers were:

- Register windows.
- Delayed branches.
- Single-cycle execution of all instructions.

ARM development

The highlights of the last decade of ARM development include:

- The introduction of the novel compressed instruction format called 'Thumb‘ which reduces cost and power dissipation in small systems;
- Significant steps upwards in performance with the ARM9, ARM 10 and 'Strong- ARM' processor families;
- A state-of-the-art software development and debugging environment;
- A very wide range of embedded applications based around ARM processor cores.

The Reference:

[1] A., Sloss, et al., ARM System Developer's Guide.