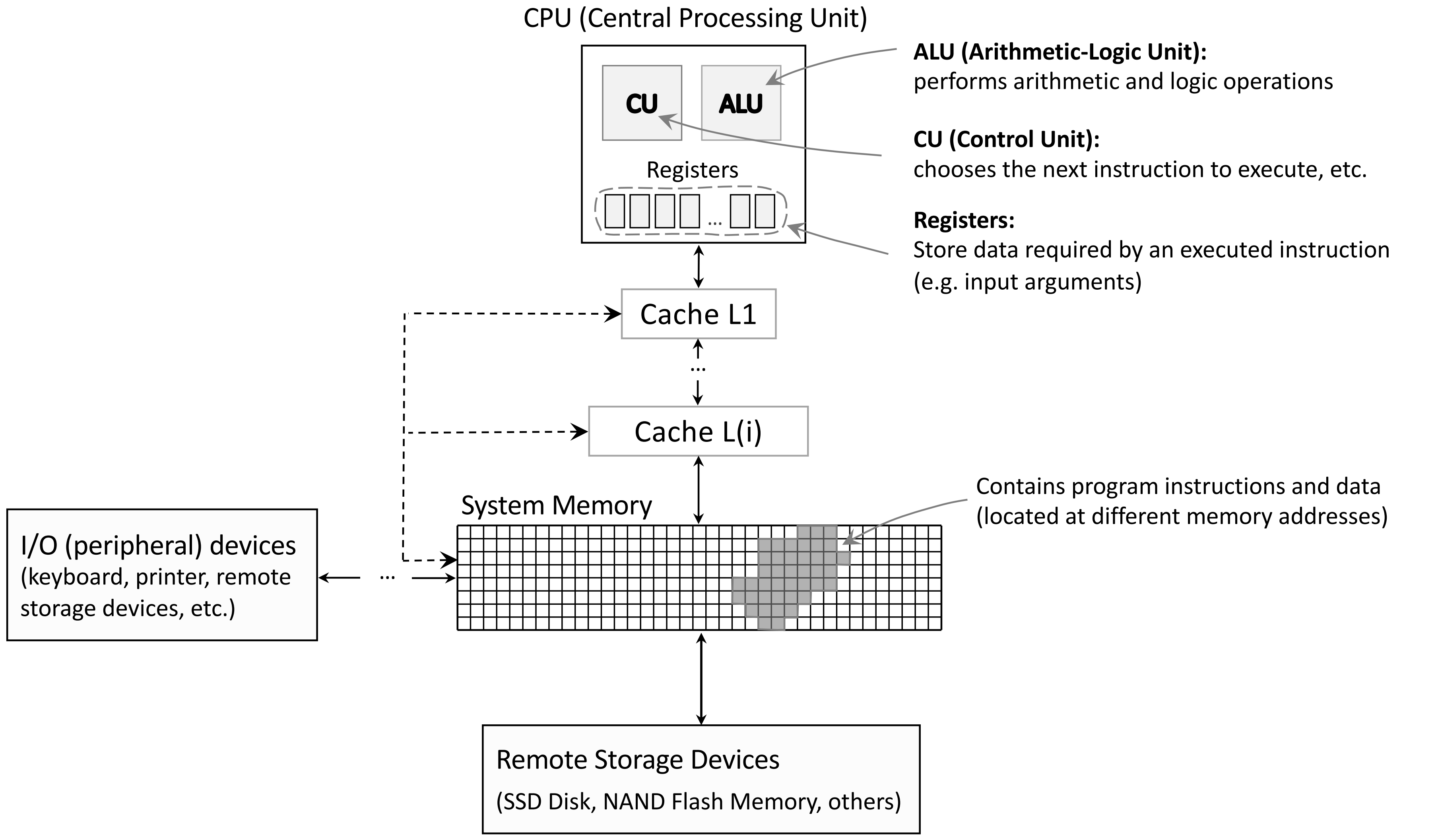
Computer Architecture
Tutorial 2

Introduction to Fundamental Ideas of Computer Architecture

Artem Burmyakov, Alexander Tormasov

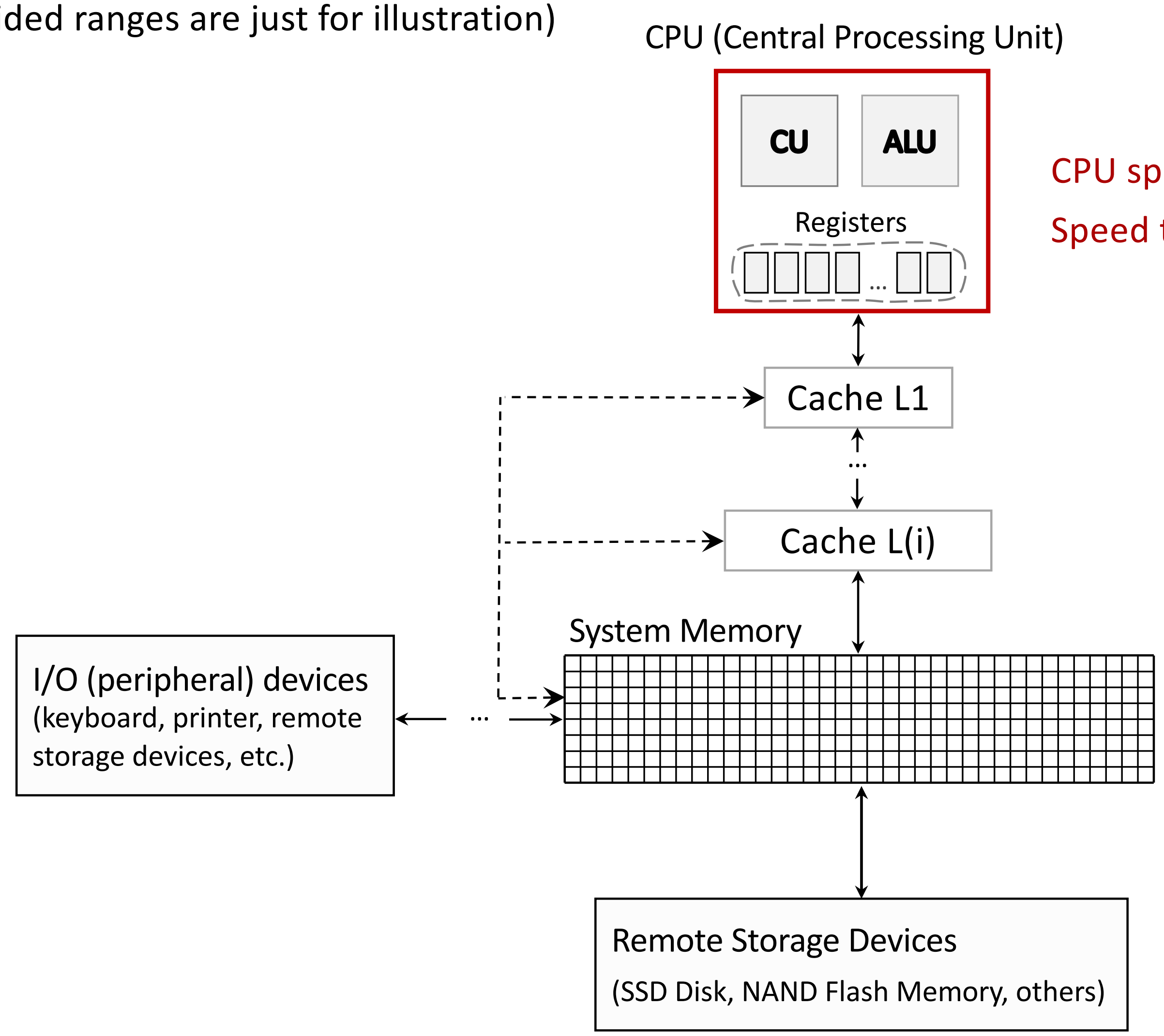
September 02, 2021

Recap of Key Computer Components


CPU (Central Processing Unit)
CU
ALU
Registers
...
ALU (Arithmetic-Logic Unit): performs arithmetic and logic operations
CU (Control Unit): chooses the next instruction to execute, etc.
Registers: Store data required by an executed instruction (e.g. input arguments)
Cache L1
...
Cache L(i)
System Memory
Contains program instructions and data (located at different memory addresses)
I/O (peripheral) devices (keyboard, printer, remote storage devices, etc.)
...
Remote Storage Devices (SSD Disk, NAND Flash Memory, others)

Some Performance Metrics

(provided ranges are just for illustration)

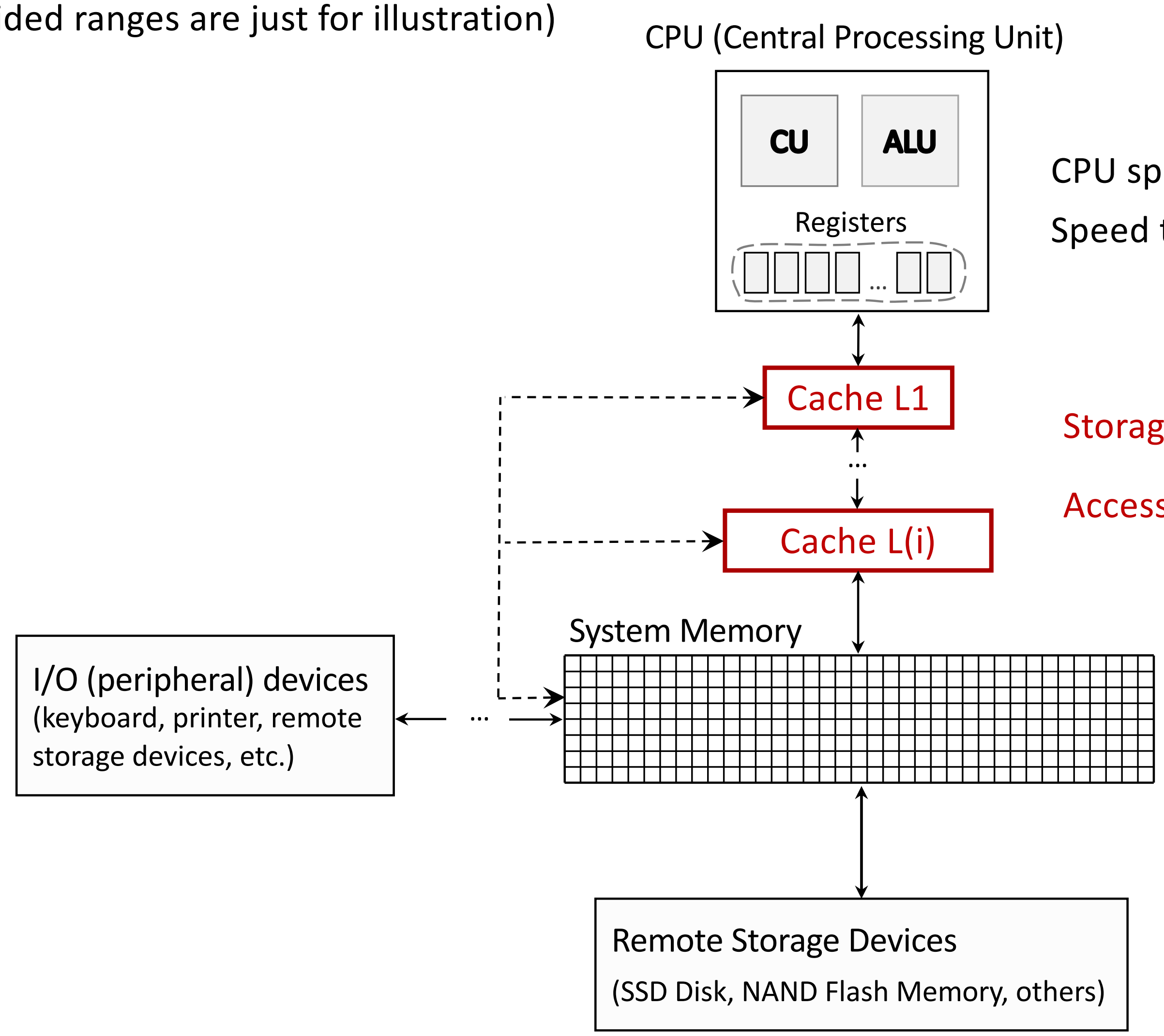


CPU (Central Processing Unit)
CU
ALU
Registers
...
Cache L1
...
Cache L(i)
System Memory
I/O (peripheral) devices
(keyboard, printer, remote storage devices, etc.)
...
Remote Storage Devices
(SSD Disk, NAND Flash Memory, others)

CPU speed (or frequency, ~2-3 GHz)

Speed turbo boost up to ~5 GHz (short time interval)

Some Performance Metrics
(provided ranges are just for illustration)


CPU (Central Processing Unit)
CU
ALU
Registers
...
Cache L1
...
Cache L(i)
System Memory
I/O (peripheral) devices
(keyboard, printer, remote storage devices, etc.)
...
Remote Storage Devices
(SSD Disk, NAND Flash Memory, others)

CPU speed (or frequency, ~2-3 GHz)

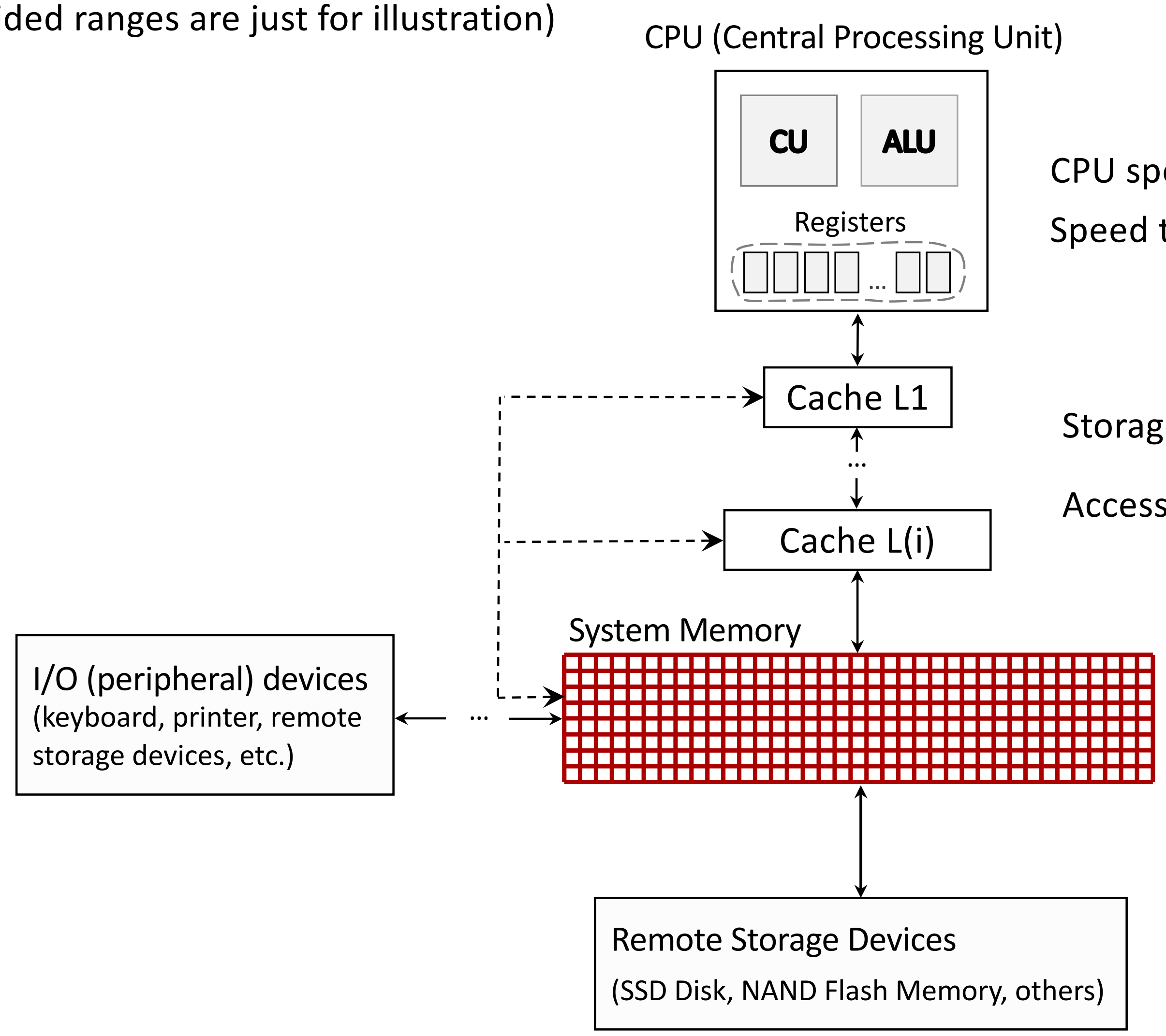
Speed turbo boost up to ~5 GHz (short time interval)

Storage capacity (up to ~15-20 MBs)

Access speed (bits/sec)

Some Performance Metrics

(provided ranges are just for illustration)


CPU (Central Processing Unit)
CU
ALU
Registers
...
Cache L1
...
Cache L(i)
System Memory
I/O (peripheral) devices (keyboard, printer, remote storage devices, etc.)
...
Remote Storage Devices
(SSD Disk, NAND Flash Memory, others)

CPU speed (or frequency, ~2-3 GHz)

Speed turbo boost up to ~5 GHz (short time interval)

Storage capacity (up to ~15-20 MBs)

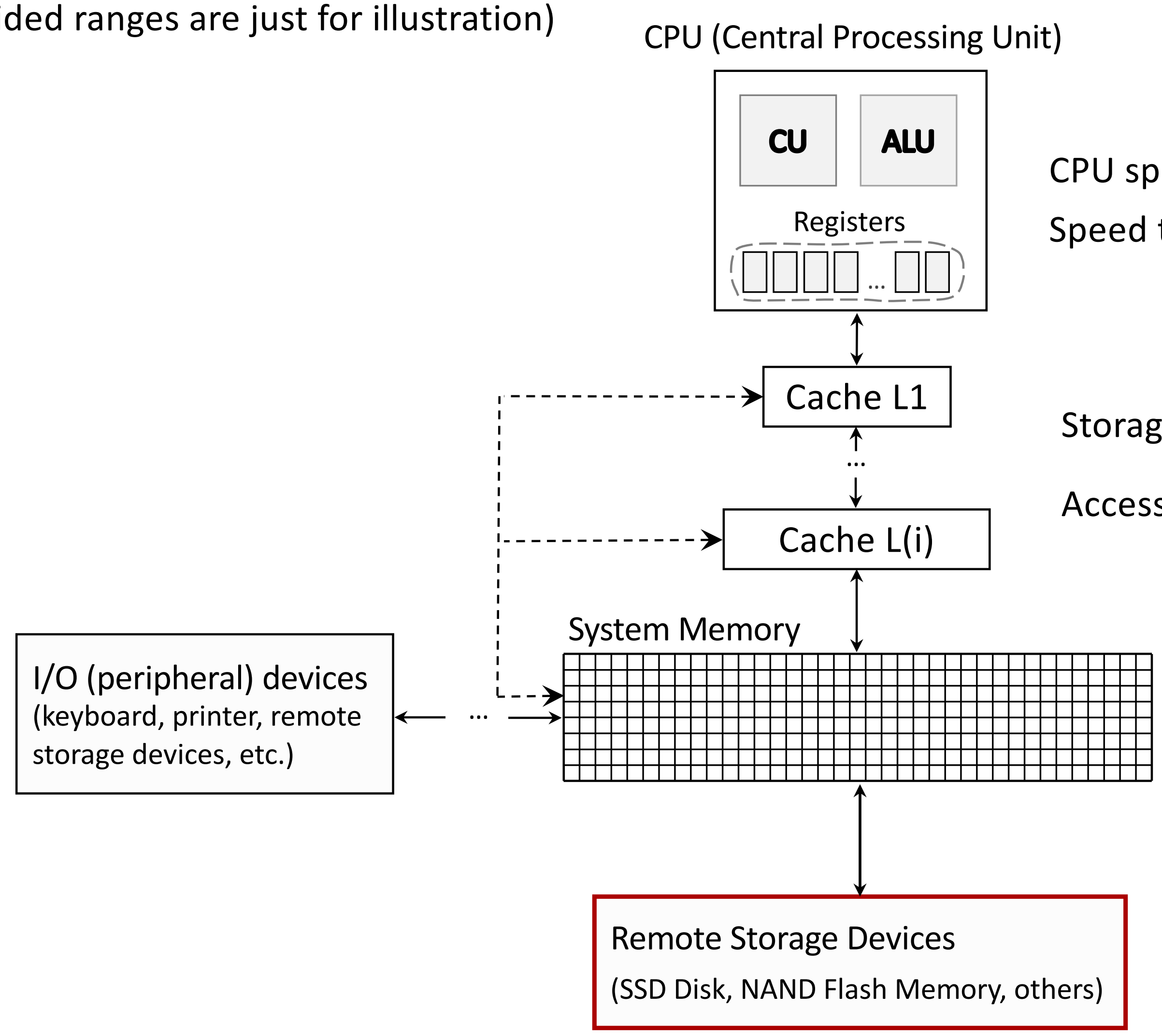
Access speed (bits/sec or MHz)

Storage capacity (up to ~16-64 GBs)

Access speed (~10-20 GBs/sec)

Some Performance Metrics

(provided ranges are just for illustration)


CPU (Central Processing Unit)
CU
ALU
Registers
Cache L1
Cache L(i)
System Memory
I/O (peripheral) devices (keyboard, printer, remote storage devices, etc.)
Remote Storage Devices (SSD Disk, NAND Flash Memory, others)

CPU speed (or frequency, ~2-3 GHz)

Speed turbo boost up to ~5 GHz (short time interval)

Storage capacity (up to ~15-20 MBs)

Access speed (bits/sec or MHz)

Storage capacity (up to ~16-64 GBs)

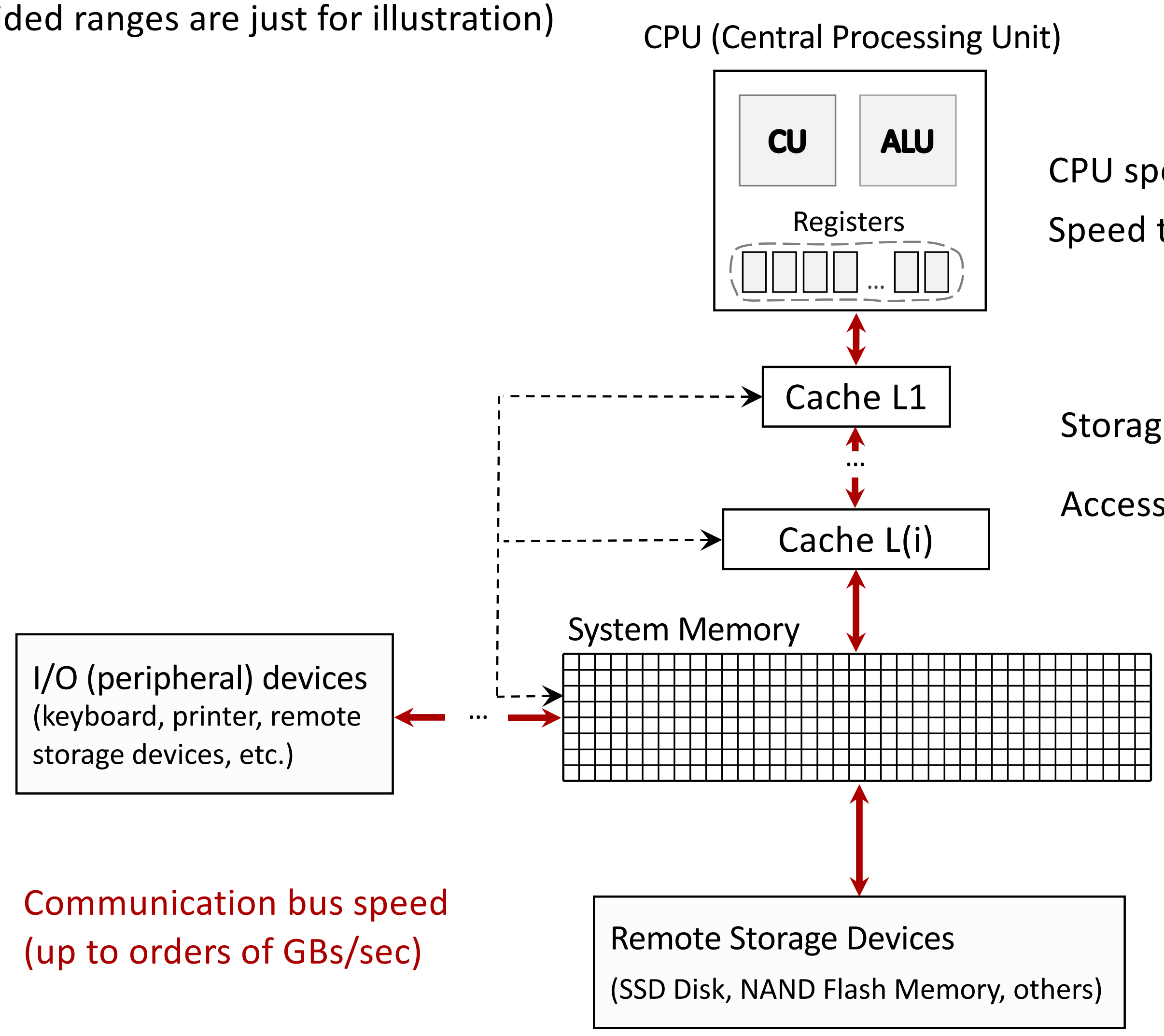
Access speed (~10-20 GBs/sec)

Storage capacity (~TBs)

Access speed (50-300 MB/s)

Some Performance Metrics

(provided ranges are just for illustration)


CPU (Central Processing Unit)
CU
ALU
Registers
...
Cache L1
...
Cache L(i)
System Memory
I/O (peripheral) devices (keyboard, printer, remote storage devices, etc.)
...
Remote Storage Devices
(SSD Disk, NAND Flash Memory, others)

CPU speed (or frequency, ~2-3 GHz)

Speed turbo boost up to ~5 GHz (short time interval)

Storage capacity (up to ~15-20 MBs)

Access speed (bits/sec or MHz)

Storage capacity (up to ~16-64 GBs)

Access speed (~10-20 GBs/sec)

Communication bus speed (up to orders of GBs/sec)

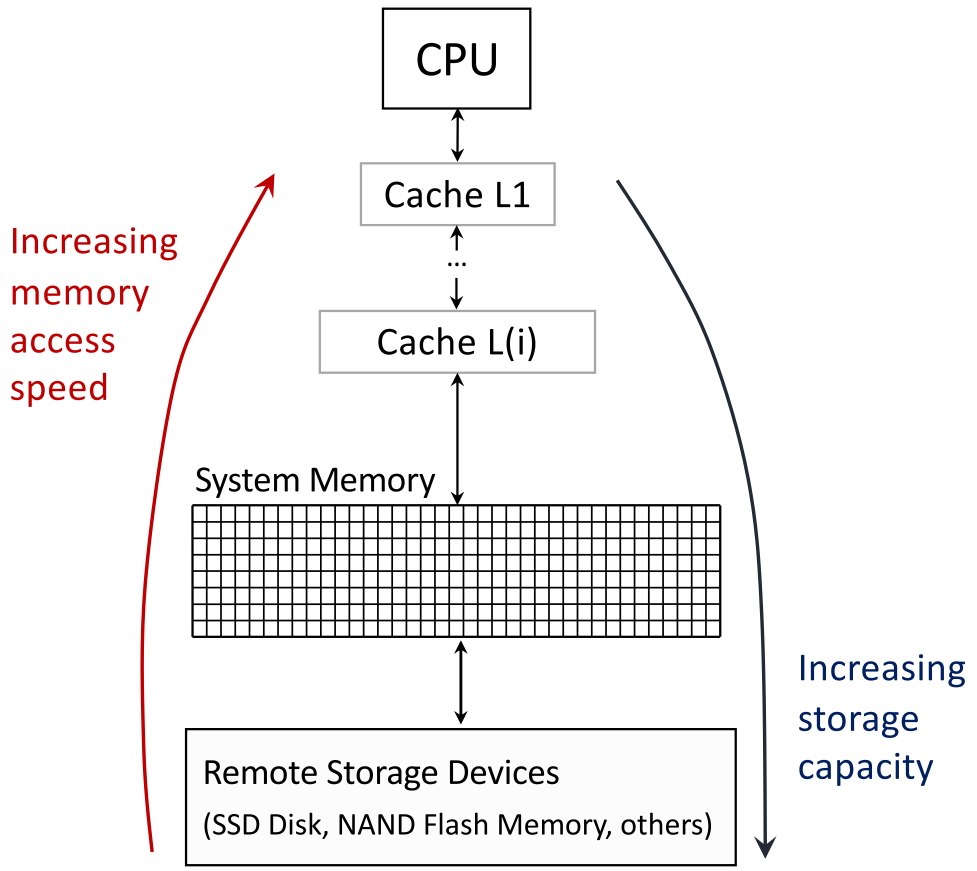
Storage capacity (~TBs)

Access speed (50-300 MB/s)

Fundamental (“Great”) Ideas of Computer Architecture

Fundamental (“Great”) Ideas of Computer Architecture

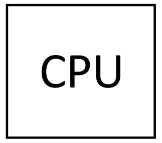
1. Hierarchy of memories

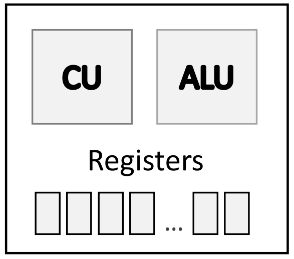

CPU
Cache L1
...
Cache L(i)
System Memory
Remote Storage Devices
(SSD Disk, NAND Flash Memory, others)
Increasing memory access speed
Increasing storage capacity

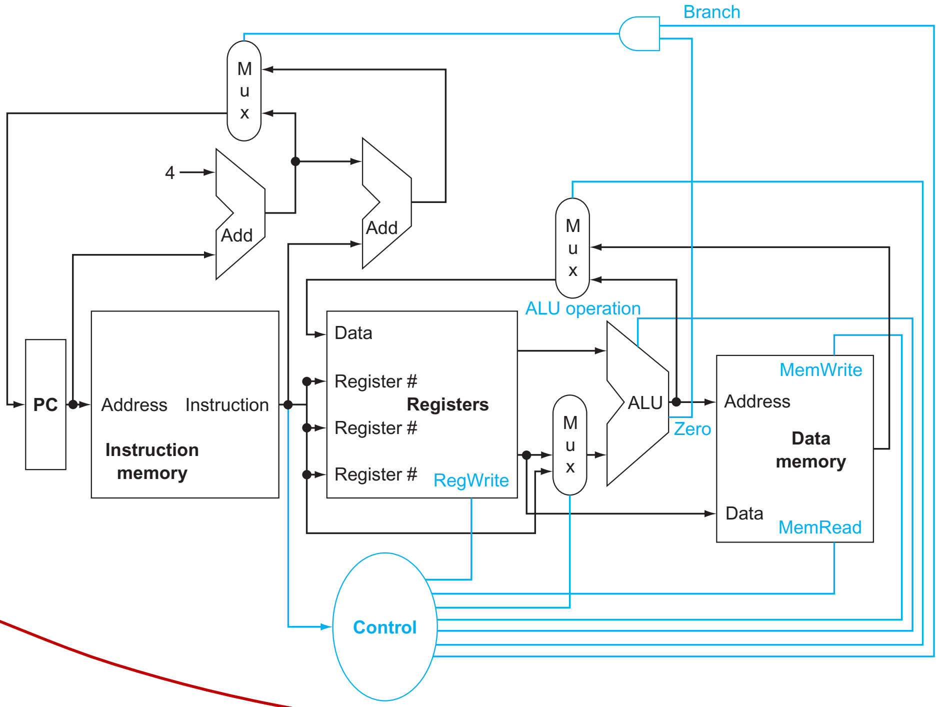
Fundamental (“Great”) Ideas of Computer Architecture

1. Hierarchy of memories
2. Use abstraction to simplify design

Example: CPU representation at different abstraction levels







…

An increasing level of details in the CPU abstraction

Fundamental (“Great”) Ideas of Computer Architecture

1. Hierarchy of memories
2. Use abstraction to simplify design
3. Design for Moore’s law

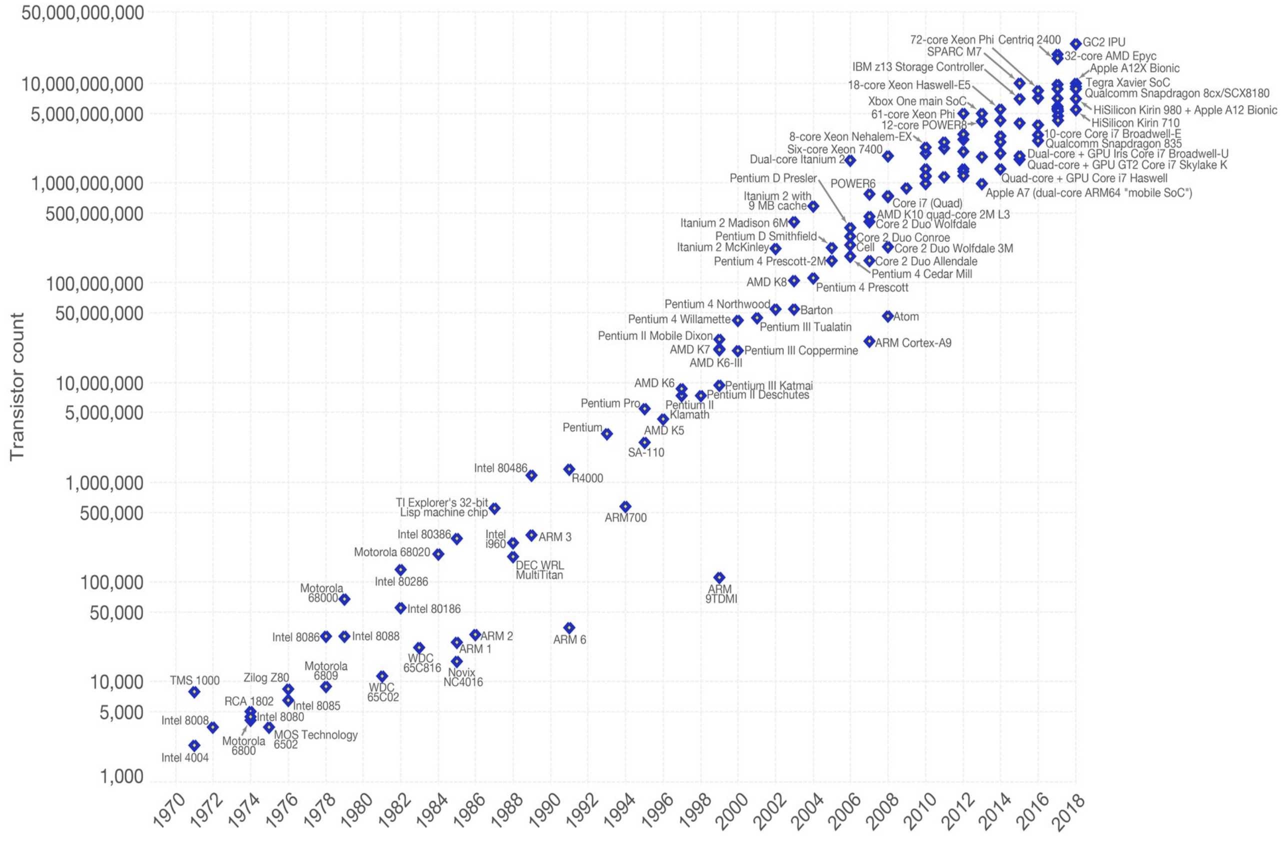
The number of transistors on a CPU chip doubles every 18-24 months (and thus, the CPU speed increases);

This observation is useful for long term projects: By the time you complete your project, the CPU speed might increase significantly

Note:
There is an ongoing discussion, if Moore’s law still holds

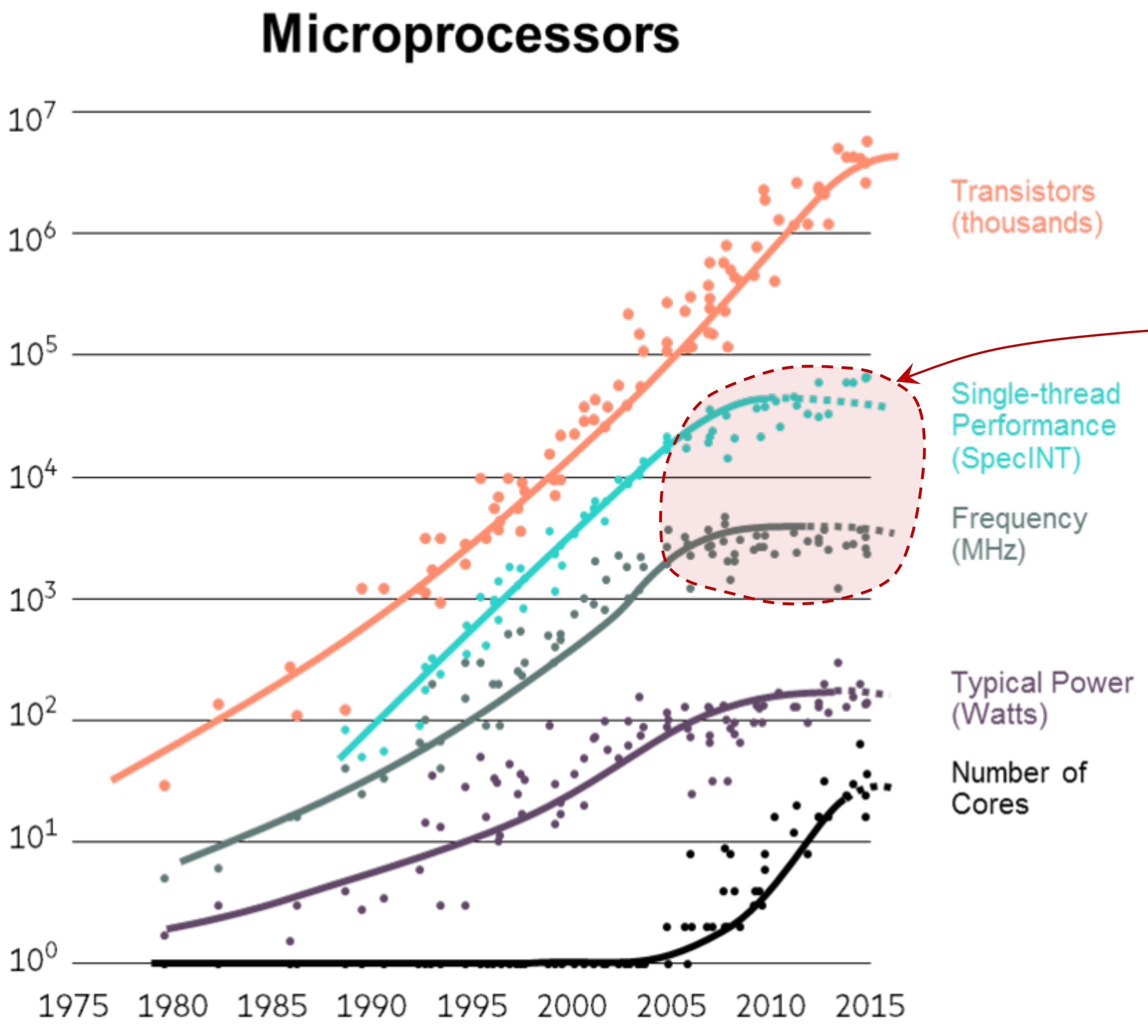
Moore’s Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

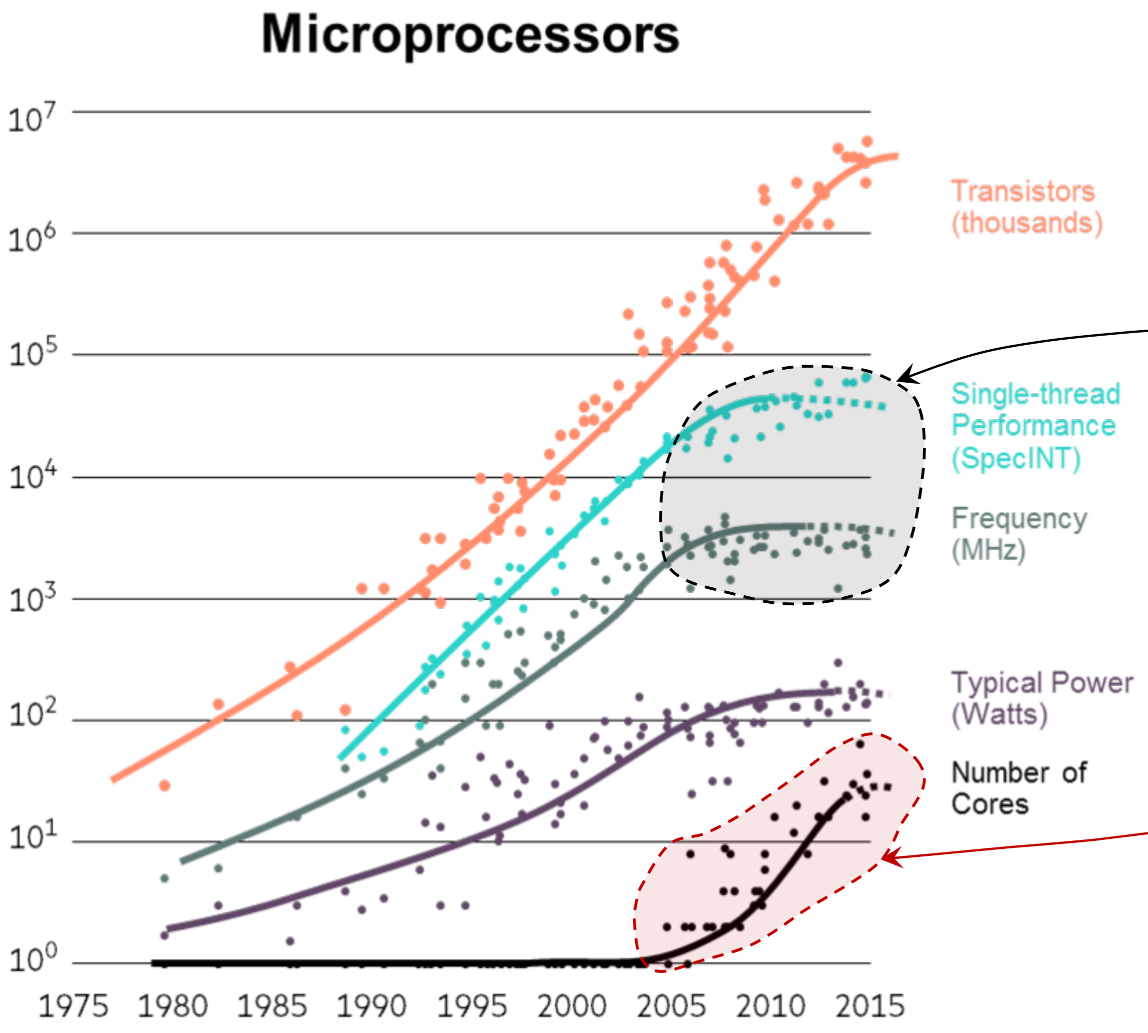
Moore’s Law Stagnation



CPU speed and single-thread performance seem not to increase since ~2008

Image is taken from https://www.quora.com/Why-is-Moores-law-no-longer-valid

Moore’s Law Stagnation

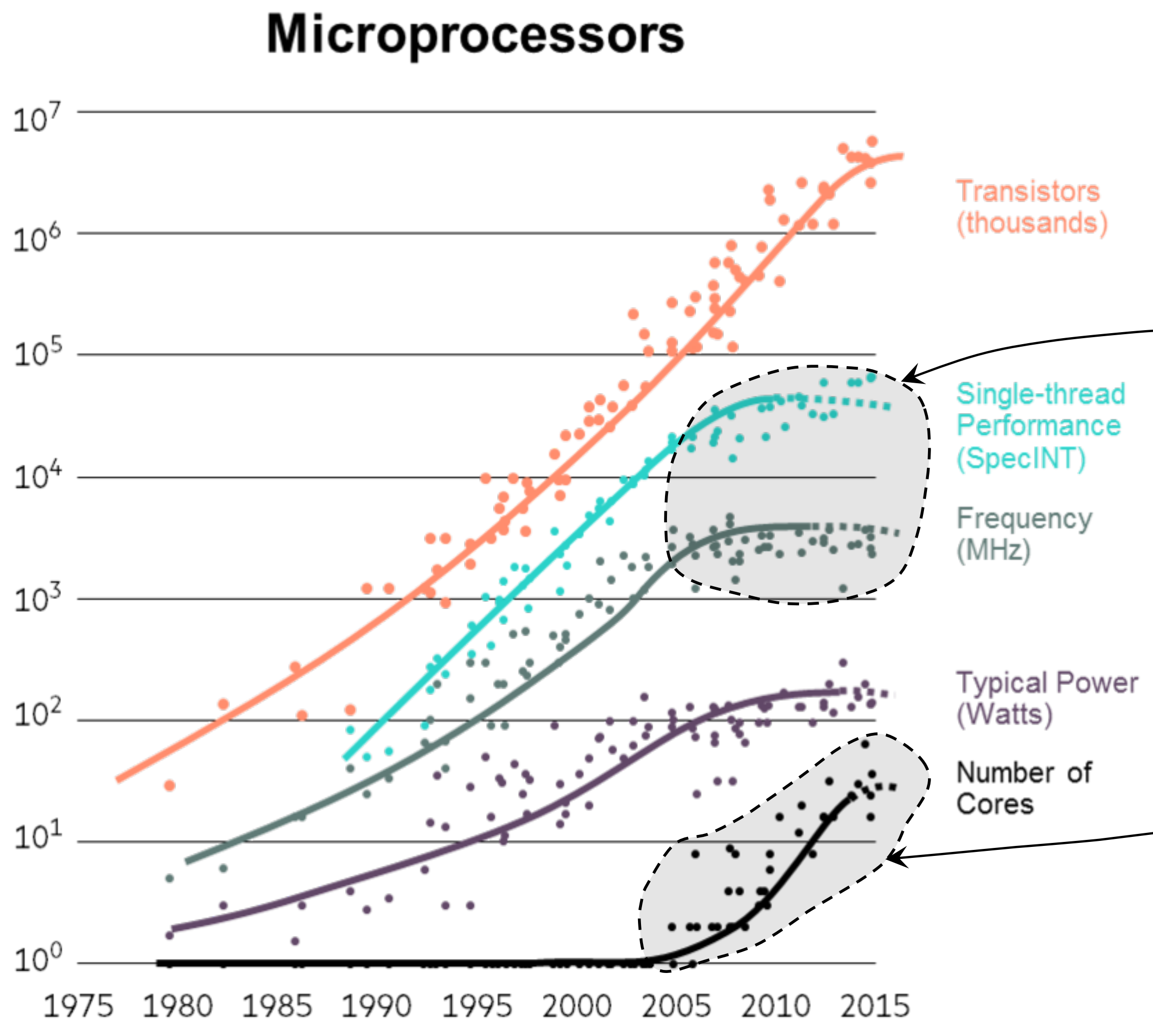


CPU speed and single-thread performance seem not to increase since ~2008

The number of cores increases instead

Image is taken from https://www.quora.com/Why-is-Moores-law-no-longer-valid

Moore’s Law Stagnation



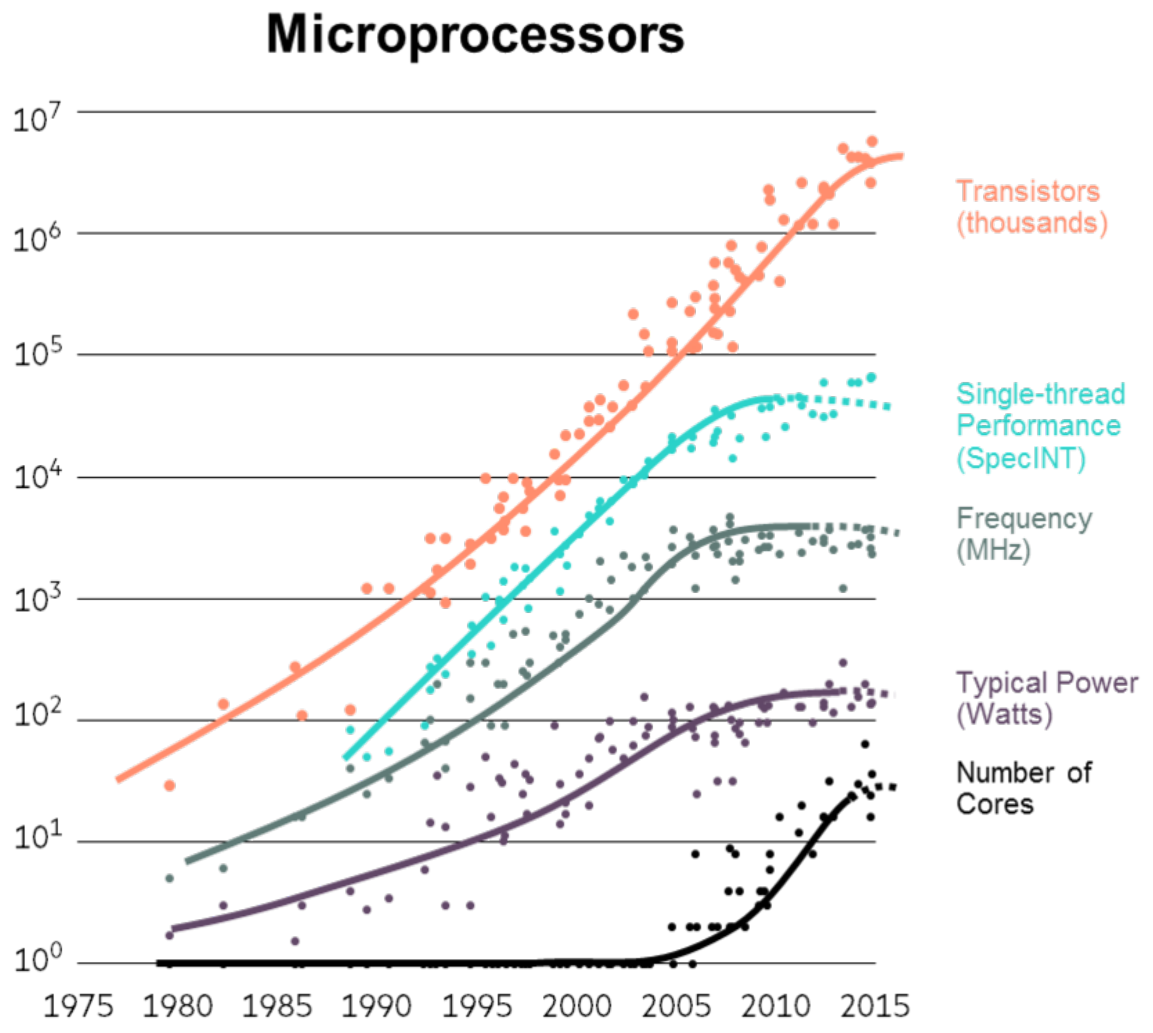
CPU speed and single-thread performance seem not to increase since ~2008

The number of cores increases instead

Nowadays, the increase of performance is achieved through the number of CPUs (or CPU cores), rather than a further increase of CPU speed

Image is taken from https://www.quora.com/Why-is-Moores-law-no-longer-valid

Moore’s Law Stagnation

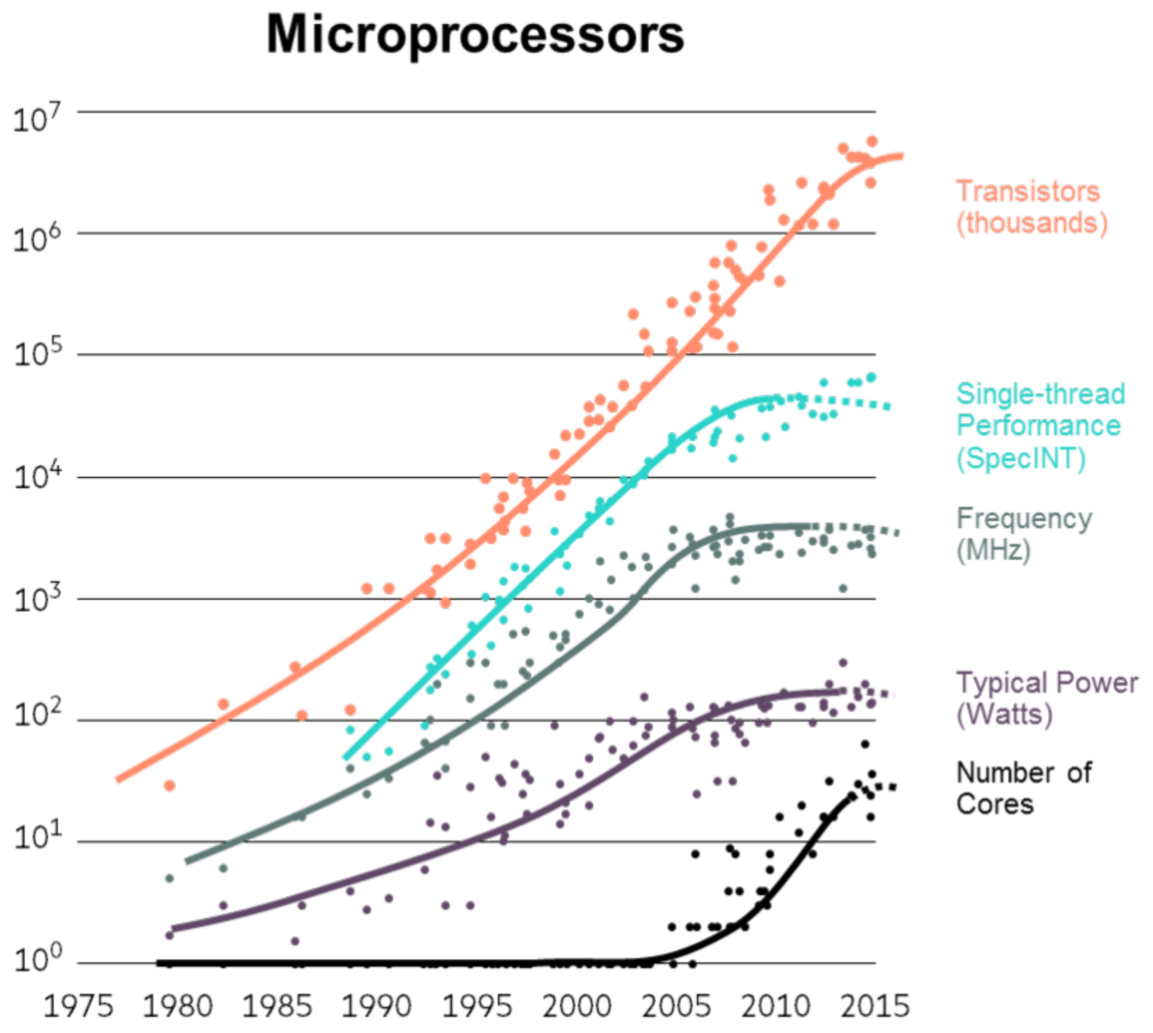


Heat dissipation problem –
the key reason for the CPU speed stagnation:

a higher clock rate –>
a higher consumed power ->
a higher power loss ->
CPU overheating

Image is taken from https://www.quora.com/Why-is-Moores-law-no-longer-valid

Moore’s Law Stagnation



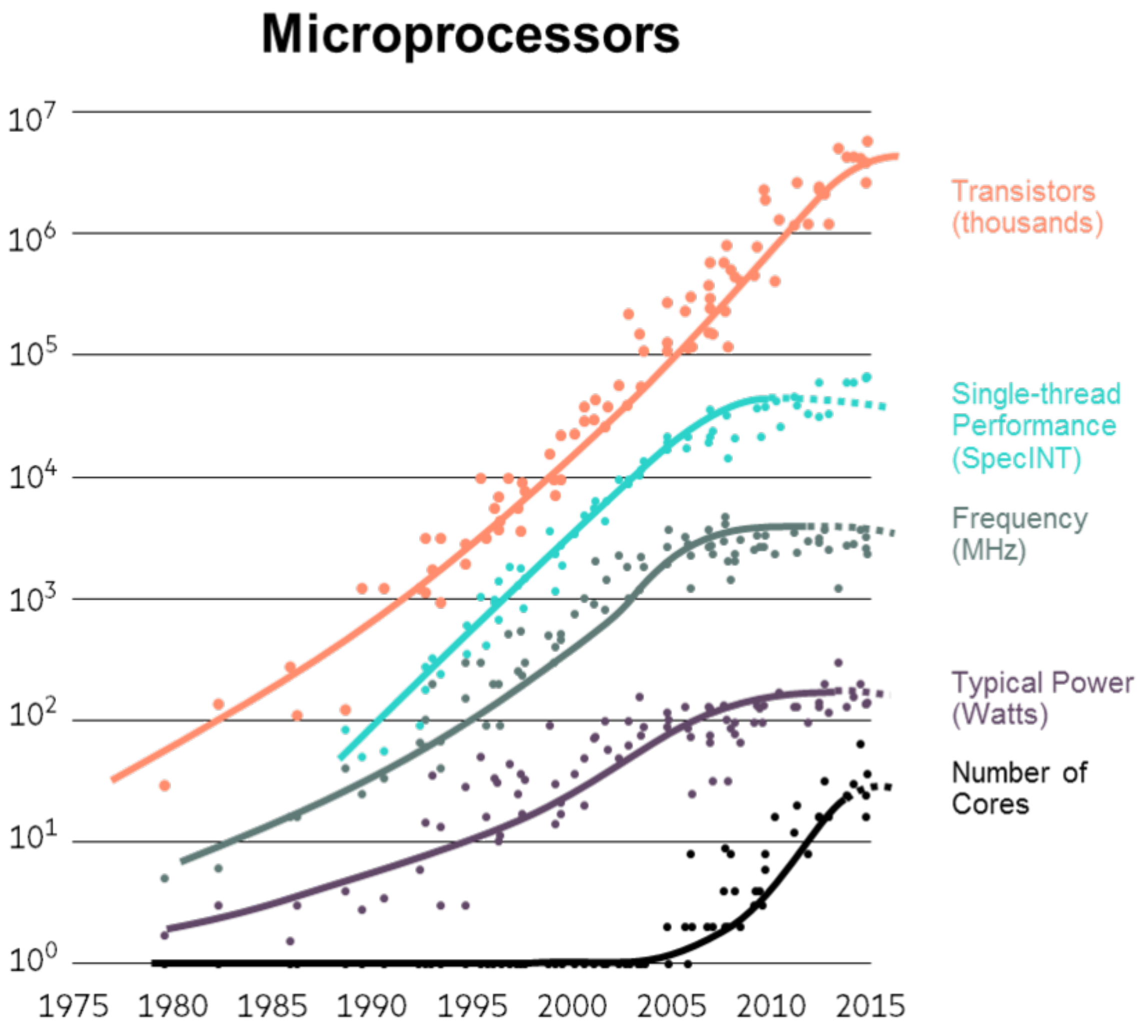
Heat dissipation problem –
the key reason for the CPU speed stagnation:

a higher clock rate –>
a higher consumed power ->
a higher power loss ->
CPU overheating

The limitation of the speed of light – another problem:
signals in a CPU already travel nearly with a speed of light, between input and output pins;

Image is taken from https://www.quora.com/Why-is-Moores-law-no-longer-valid

Moore’s Law Stagnation



Heat dissipation problem –
the key reason for the CPU speed stagnation:

a higher clock rate –>
a higher consumed power ->
a higher power loss ->
CPU overheating

The limitation of the speed of light – another problem:
signals in a CPU already travel nearly with a speed of light, between input and output pins;

There is a strong need for multiprocessor systems;

Multiprocessor systems however lead to various concurrency problems, such as race conditions

Image is taken from https://www.quora.com/Why-is-Moores-law-no-longer-valid

Fundamental (“Great”) Ideas of Computer Architecture

1. Hierarchy of memories
2. Use abstraction to simplify design
3. Design for Moore’s law
4. Performance via parallelism

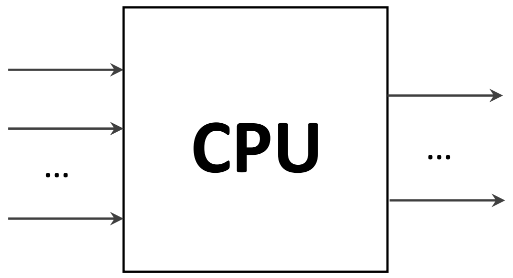
Fundamental ("Great") Ideas of Computer Architecture

4. Performance via parallelism

Computer program
(a set of instructions to be executed);
loaded into system memory

Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5
Instruction 6
Instruction 7
…

Single processor (or uniprocessor) system:



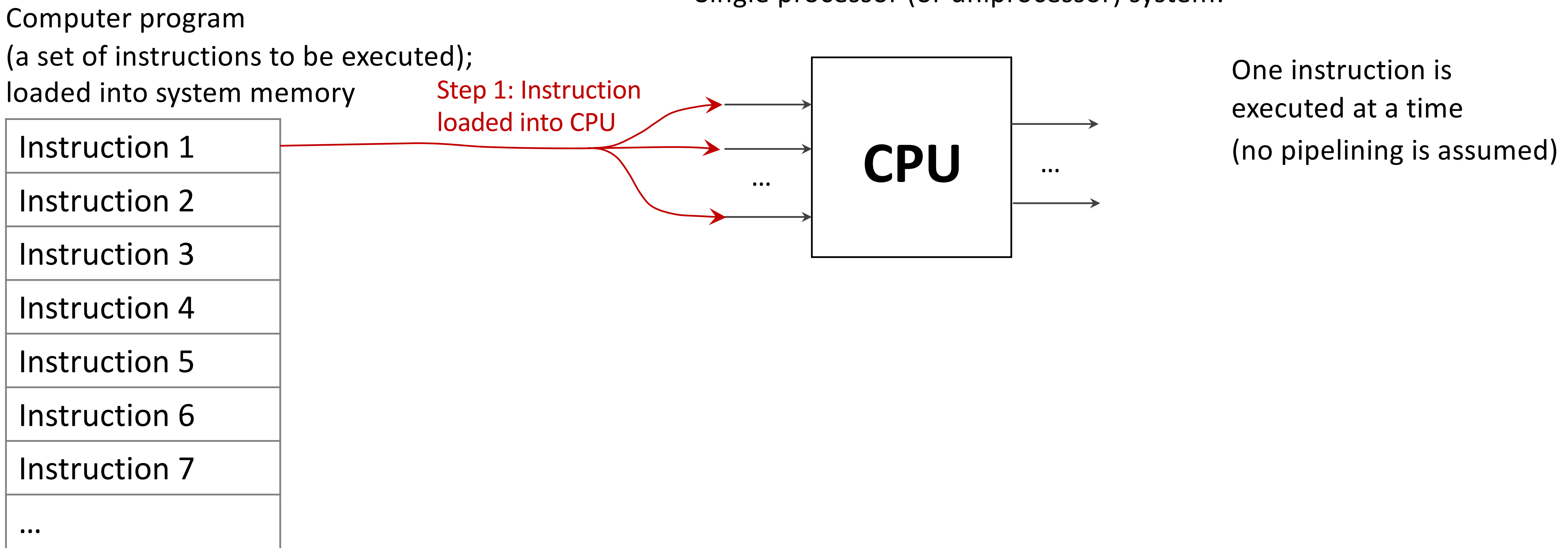
One instruction is
executed at a time
(no pipelining is assumed)

Assumptions:
- Instructions are executed sequentially;
- CPU executes one instruction at a time (not always a case)

Fundamental ("Great") Ideas of Computer Architecture

4. Performance via parallelism

Single processor (or uniprocessor) system:



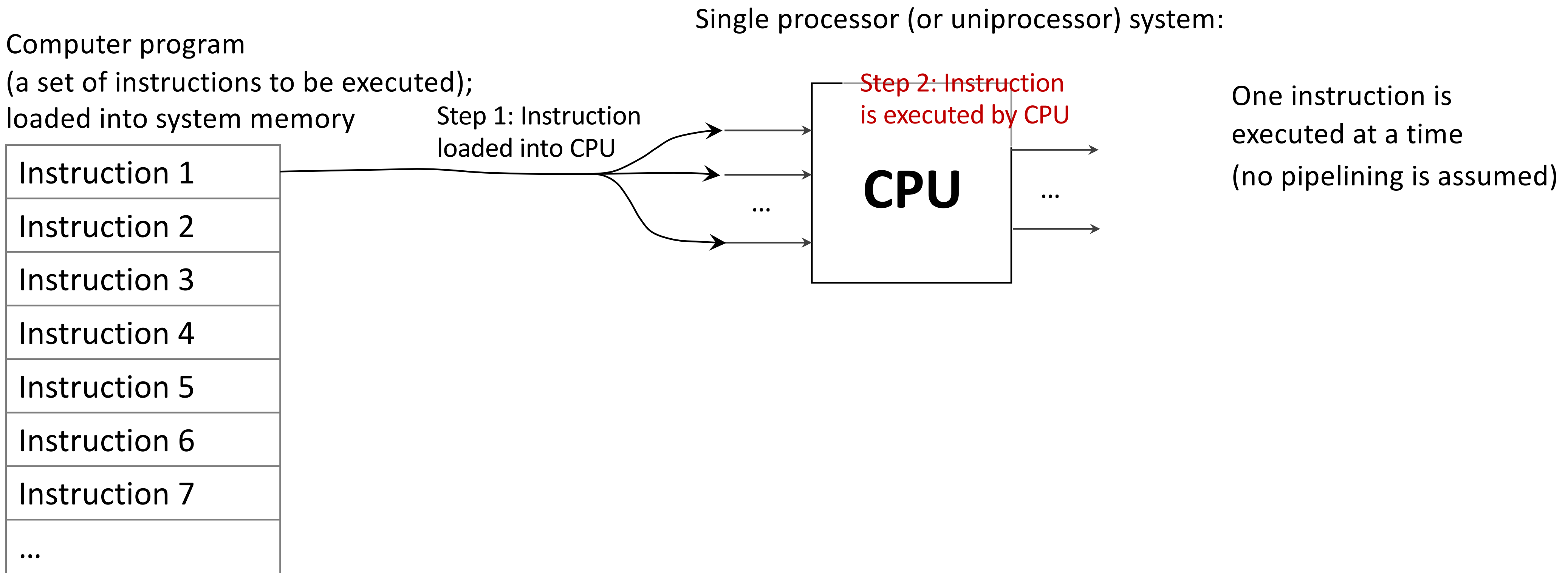
Computer program
(a set of instructions to be executed);
loaded into system memory
Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5
Instruction 6
Instruction 7
…
Step 1: Instruction loaded into CPU
…
CPU
…
One instruction is executed at a time
(no pipelining is assumed)

Assumptions:

- Instructions are executed sequentially;
- CPU executes one instruction at a time (not always a case)

Fundamental ("Great") Ideas of Computer Architecture

4. Performance via parallelism

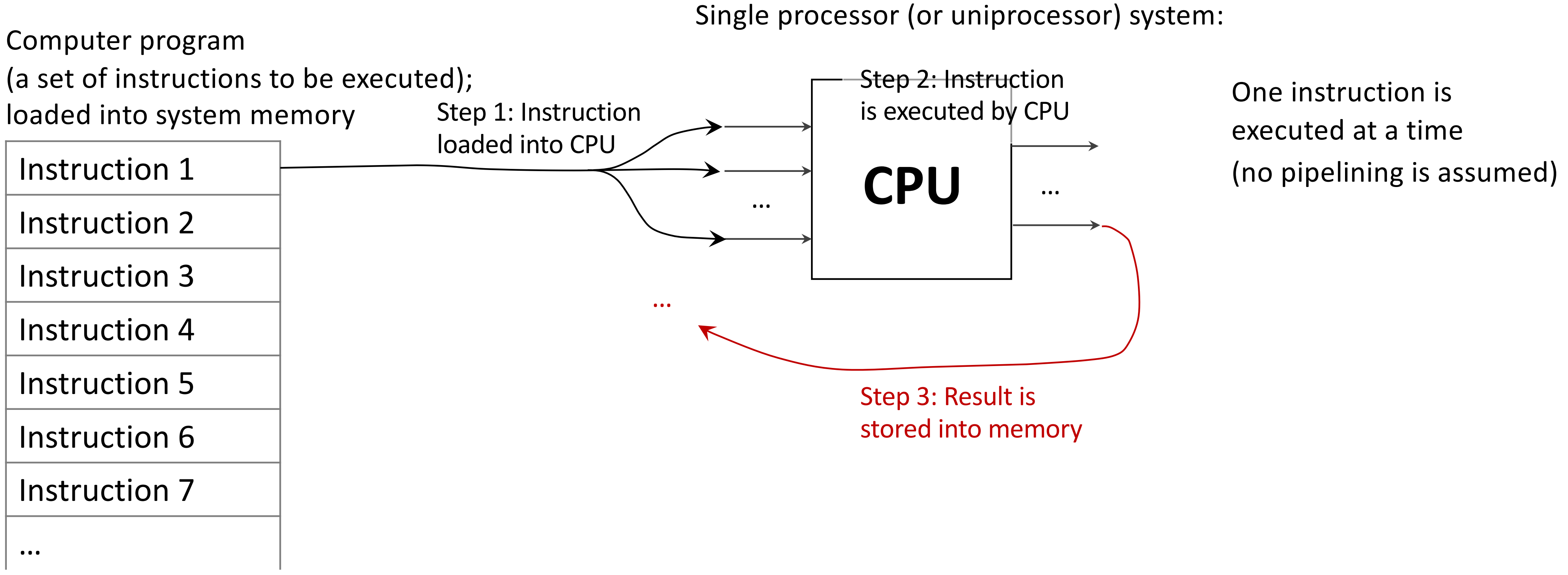

Single processor (or uniprocessor) system:
Computer program
(a set of instructions to be executed);
loaded into system memory
Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5
Instruction 6
Instruction 7
…
Step 1: Instruction loaded into CPU
…
Step 2: Instruction is executed by CPU
CPU
…
One instruction is executed at a time
(no pipelining is assumed)

Assumptions:

- Instructions are executed sequentially;
- CPU executes one instruction at a time (not always a case)

Fundamental ("Great") Ideas of Computer Architecture

4. Performance via parallelism

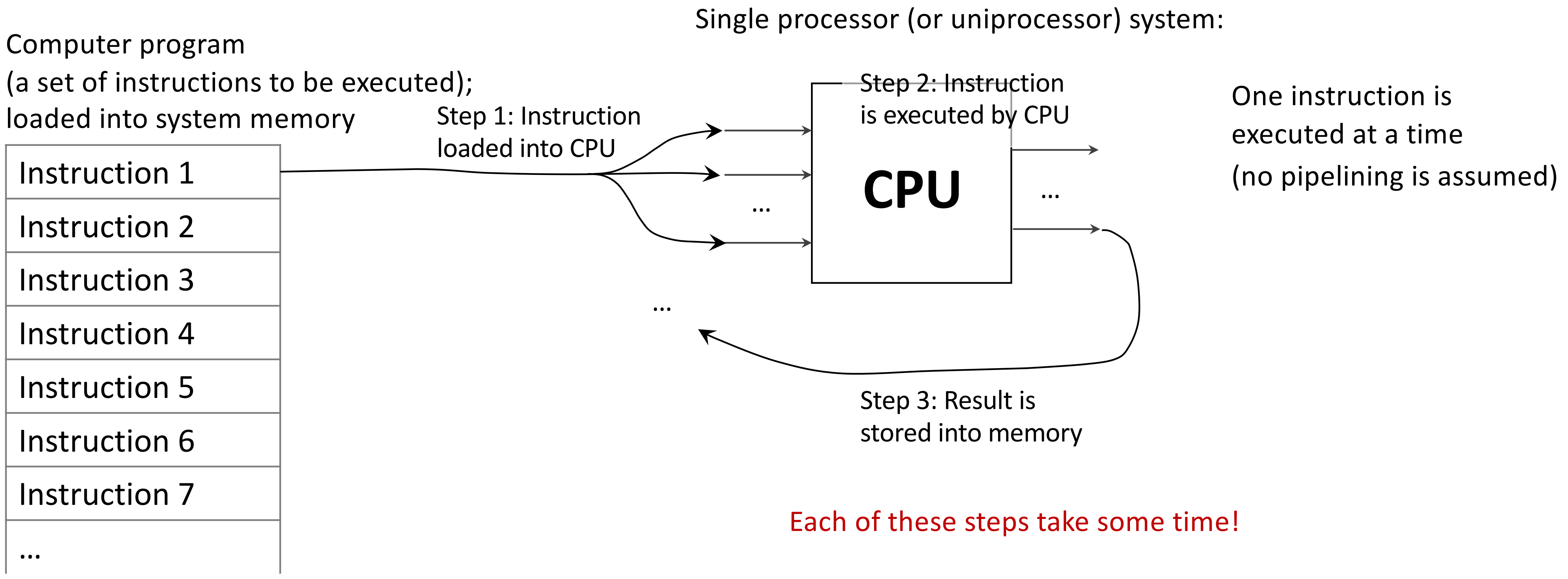


Assumptions:

- Instructions are executed sequentially;
- CPU executes one instruction at a time (not always a case)

Fundamental ("Great") Ideas of Computer Architecture

4. Performance via parallelism



Single processor (or uniprocessor) system:
Computer program
(a set of instructions to be executed);
loaded into system memory
Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5
Instruction 6
Instruction 7
…
Step 1: Instruction loaded into CPU
Step 2: Instruction is executed by CPU
CPU
Step 3: Result is stored into memory
One instruction is executed at a time
(no pipelining is assumed)

Each of these steps take some time!

Assumptions:

- Instructions are executed sequentially;
- CPU executes one instruction at a time (not always a case)

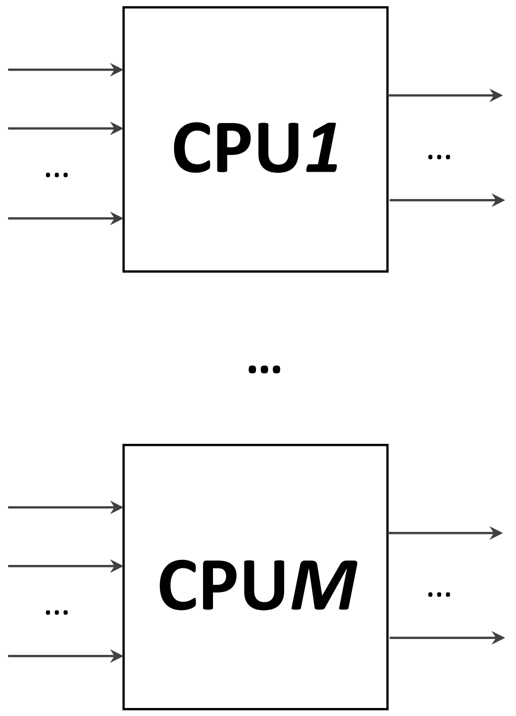
Fundamental ("Great") Ideas of Computer Architecture

4. Performance via parallelism

Computer program
(a set of instructions to be executed);
loaded into system memory

Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5
Instruction 6
Instruction 7
…

Multiprocessor (or multicore) system:



M instructions can be executed at a time, simultaneously

Assumptions:

- Instructions are executed sequentially;
- CPU executes one instruction at a time (not always a case)

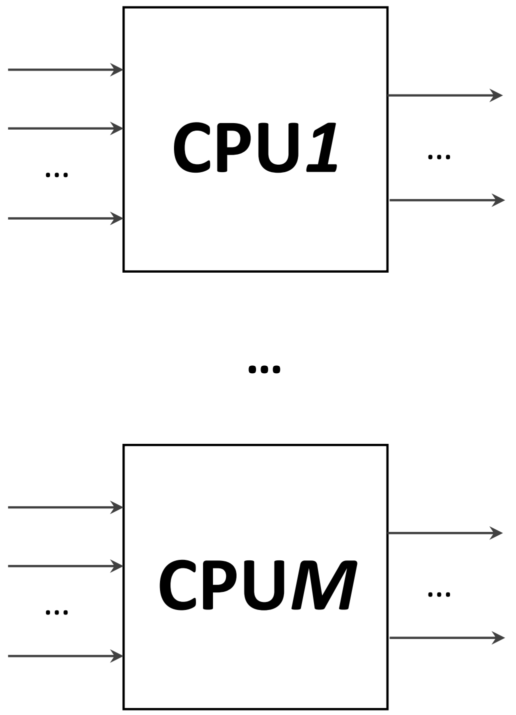
Fundamental ("Great") Ideas of Computer Architecture

4. Performance via parallelism

Computer program
(a set of instructions to be executed);
loaded into system memory

Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5
Instruction 6
Instruction 7
…

Multiprocessor (or multicore) system:



M instructions can be executed at a time, simultaneously

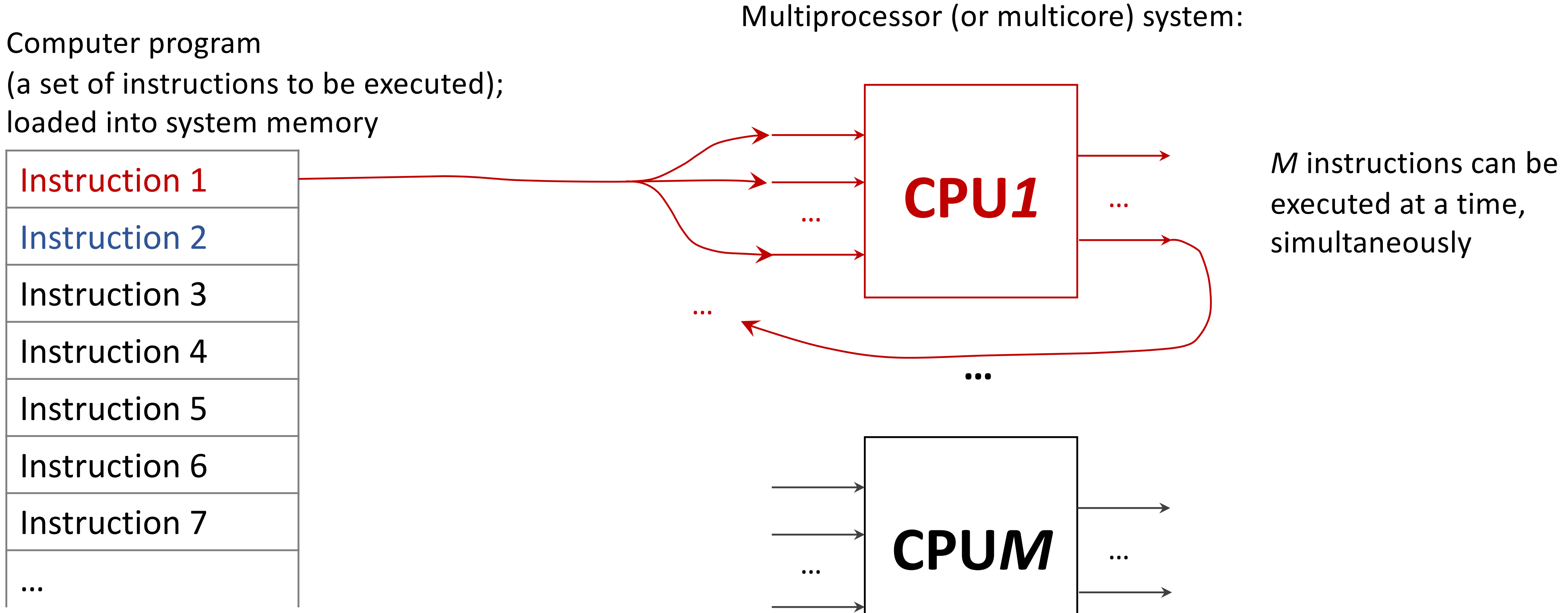
Assumptions:

- Instructions are executed sequentially;
- CPU executes one instruction at a time (not always a case)

Fundamental ("Great") Ideas of Computer Architecture

4. Performance via parallelism

Computer program
(a set of instructions to be executed);
loaded into system memory



M instructions can be executed at a time, simultaneously

Assumptions:

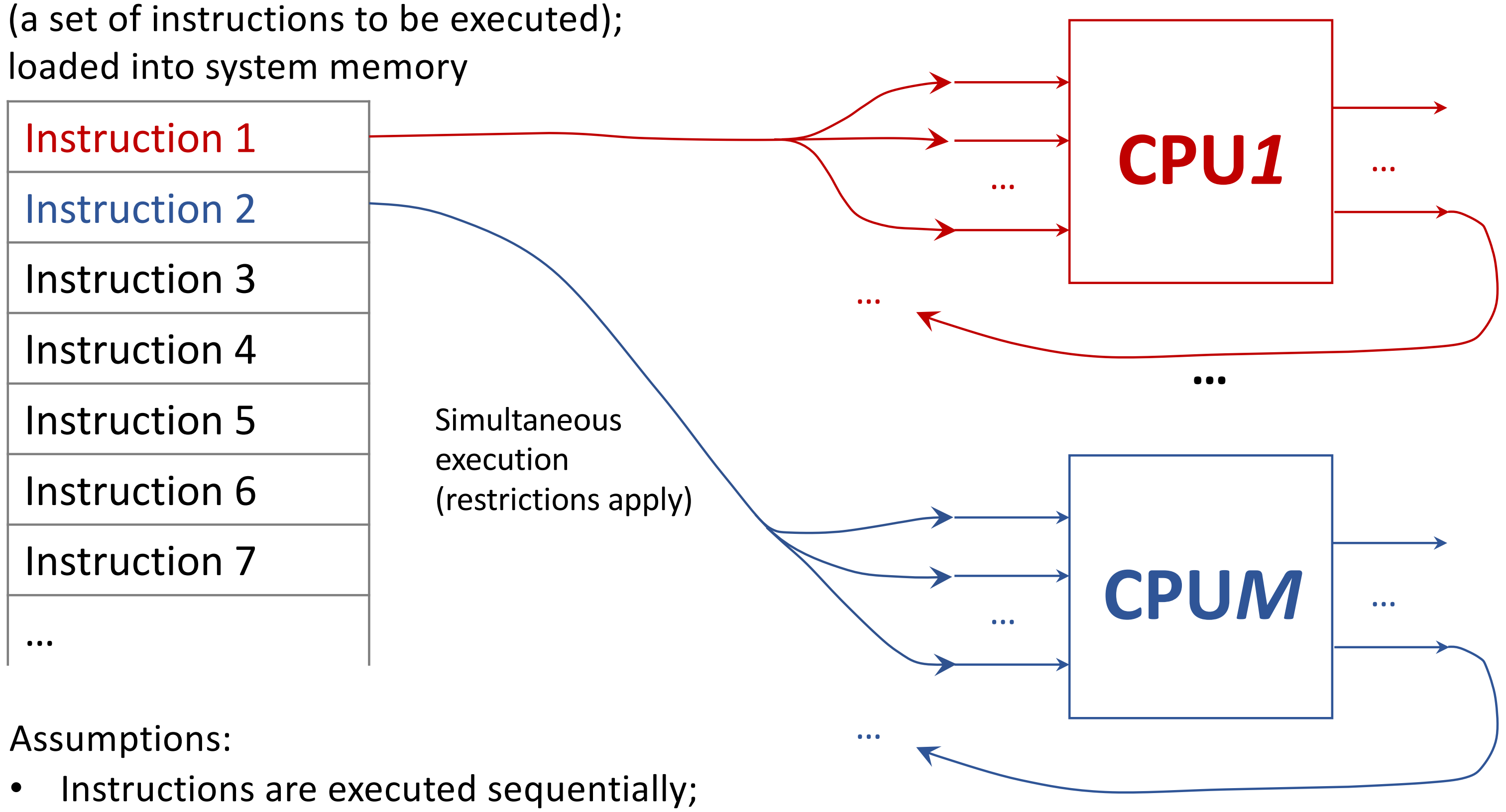
- Instructions are executed sequentially;
- CPU executes one instruction at a time (not always a case)

Fundamental ("Great") Ideas of Computer Architecture

4. Performance via parallelism

Computer program
(a set of instructions to be executed);
loaded into system memory

Multiprocessor (or multicore) system:



M instructions can be executed at a time, simultaneously

Multiprocessing aims at speeding-up program execution, or simultaneous execution of multiple programs

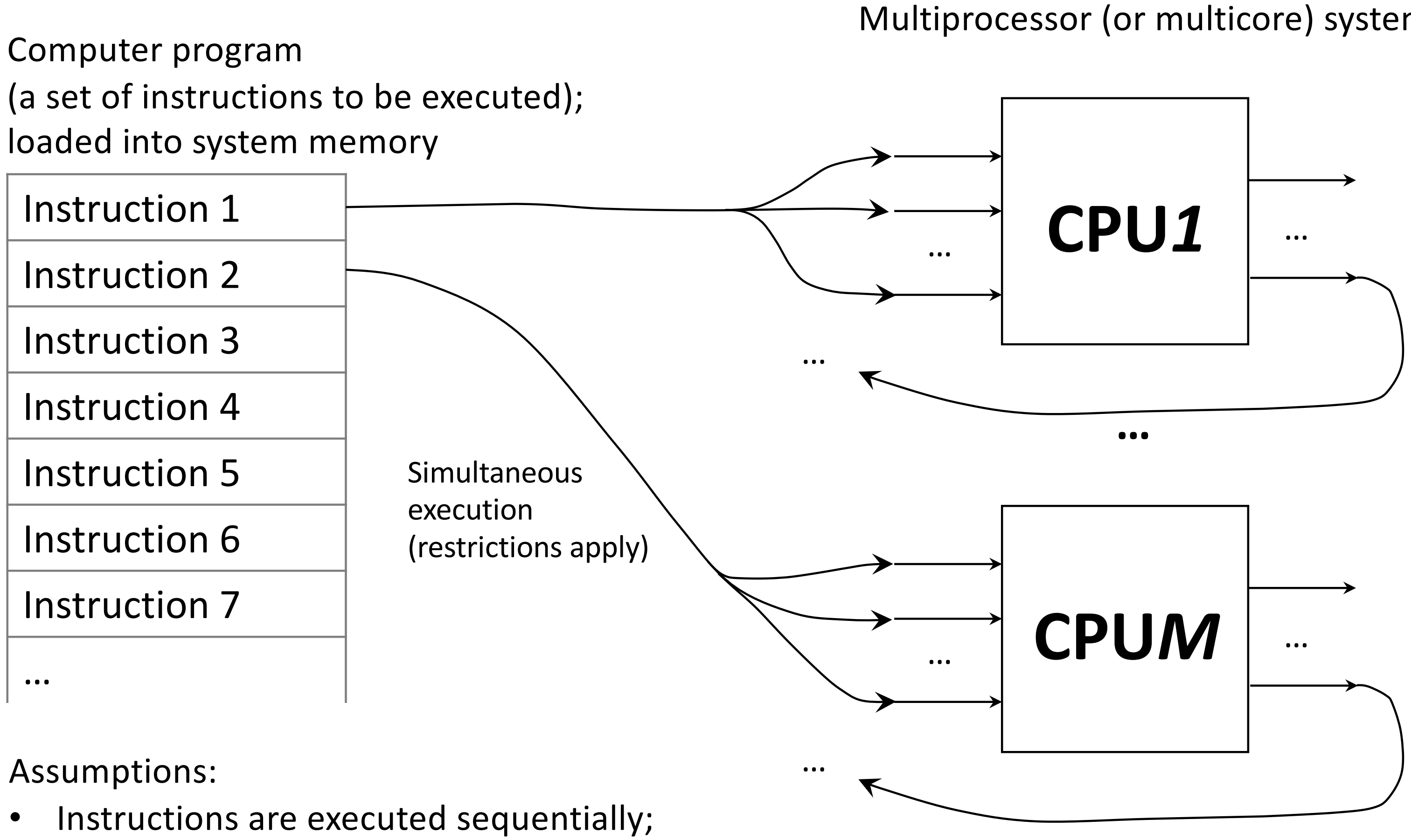
Assumptions:

- Instructions are executed sequentially;
- CPU executes one instruction at a time (not always a case)

Fundamental ("Great") Ideas of Computer Architecture

4. Performance via parallelism

Computer program
(a set of instructions to be executed);
loaded into system memory



M instructions can be executed at a time, simultaneously

Multiprocessing aims at speeding-up program execution, or simultaneous execution of multiple programs

The major problem:

Some instructions depend on the execution result of previous instructions, and, thus, have to wait for their completion

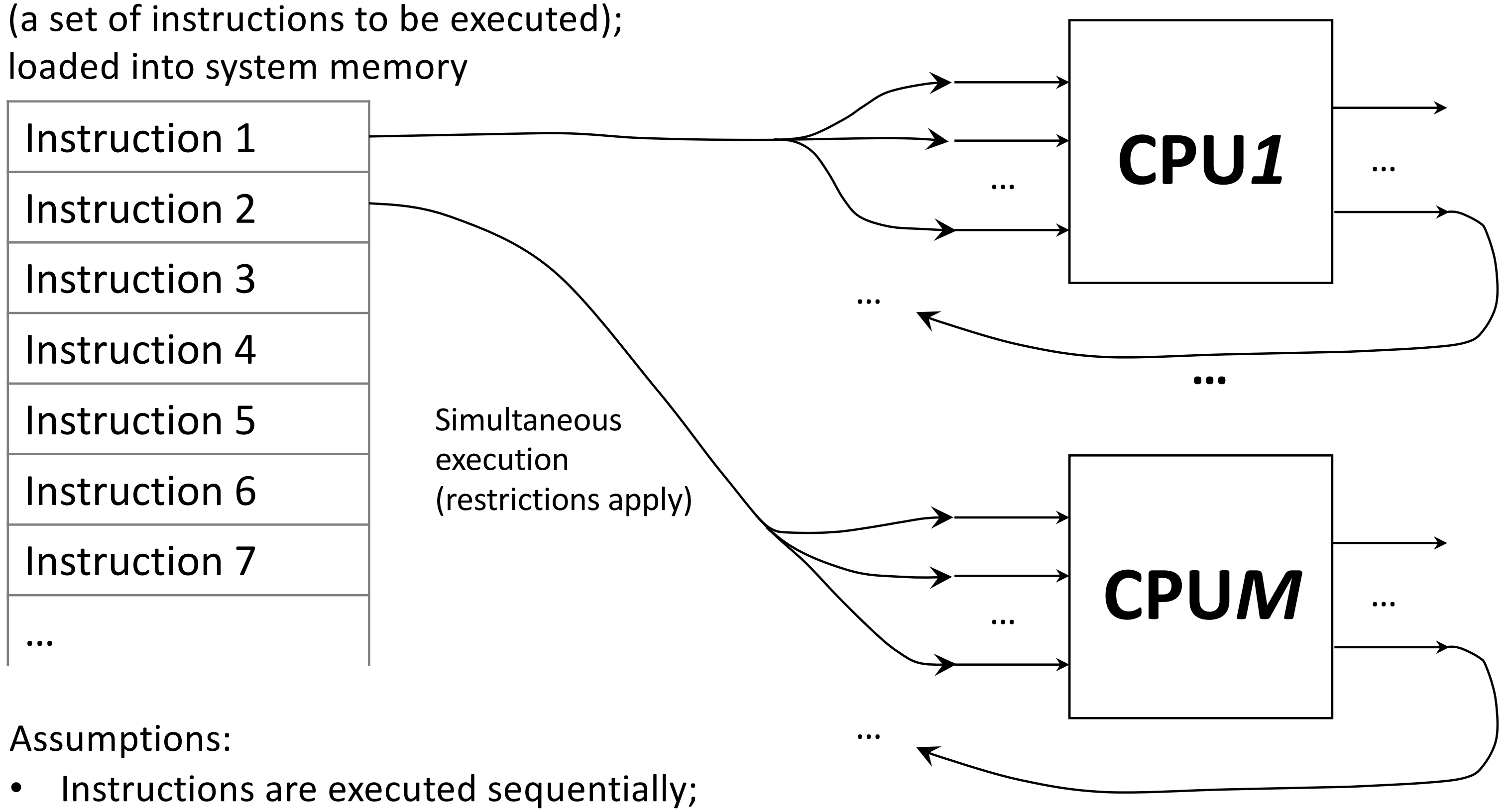
Assumptions:

- Instructions are executed sequentially;
- CPU executes one instruction at a time (not always a case)

Fundamental ("Great") Ideas of Computer Architecture

4. Performance via parallelism

Computer program
(a set of instructions to be executed);
loaded into system memory



M instructions can be executed at a time, simultaneously

Multiprocessing aims at speeding-up program execution, or simultaneous execution of multiple programs

Many problems remain open;

These problems got a lot of attention after Moore's law "slow down"

Assumptions:

- Instructions are executed sequentially;
- CPU executes one instruction at a time (not always a case)

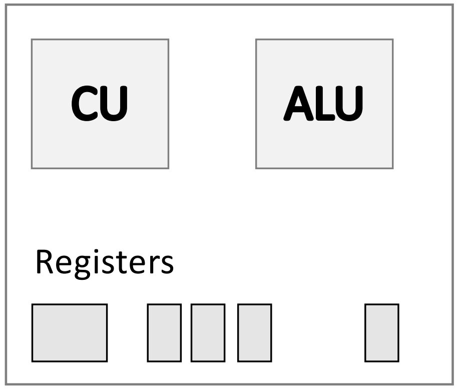
Fundamental (“Great”) Ideas of Computer Architecture

1. Hierarchy of memories
2. Use abstraction to simplify design
3. Design for Moore’s law
4. Performance via parallelism
5. Performance via pipelining

Fundamental (“Great”) Ideas of Computer Architecture

5. Performance via pipelining

Our custom CPU



Instruction Set, supported by our CPU:

Operation	Operation Code
Logical “OR”	0
Logical “AND”	1

Instruction Format:

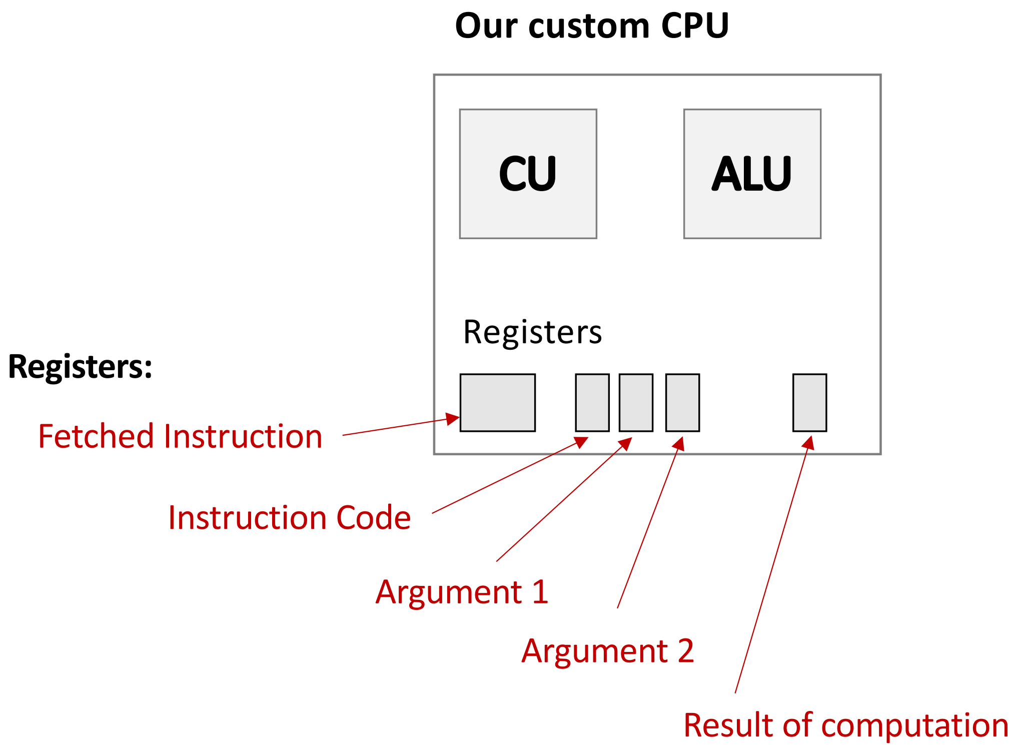
Operation Code	Argument 1	Argument 2

Sample Instruction:

0	1	0

Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining



Instruction Set, supported by our CPU:

Operation	Operation Code
Logical "OR"	0
Logical "AND"	1

Instruction Format:

Operation Code	Argument 1	Argument 2

Sample Instruction:

0	1	0

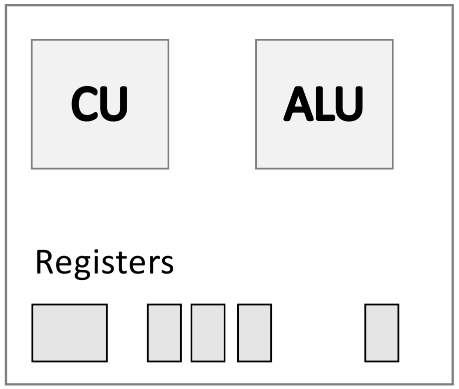
Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining

Instructions in System Memory

Instruction 1: **010**
Instruction 2
Instruction 3
Instruction 4
Instruction 5
…

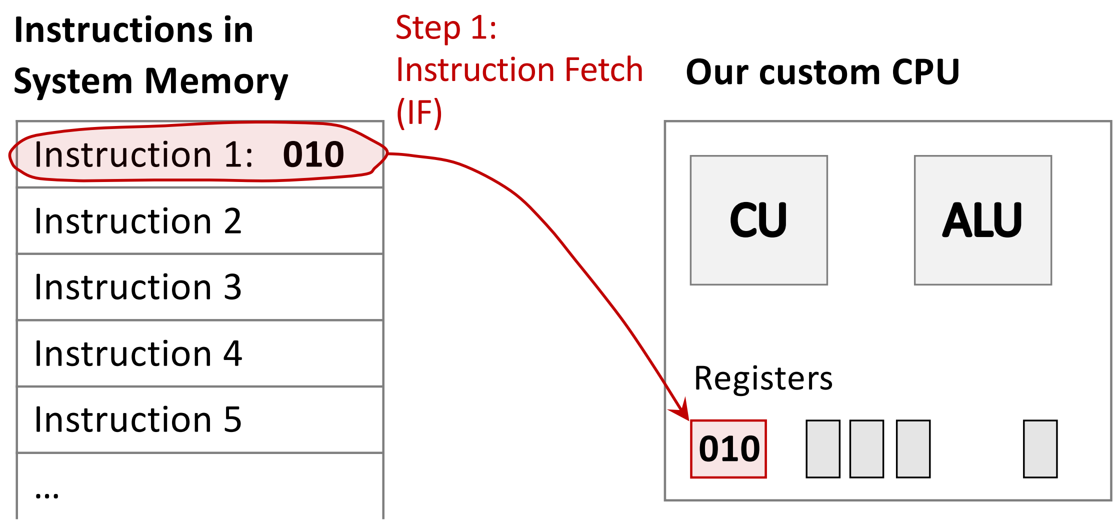
Our custom CPU



Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining

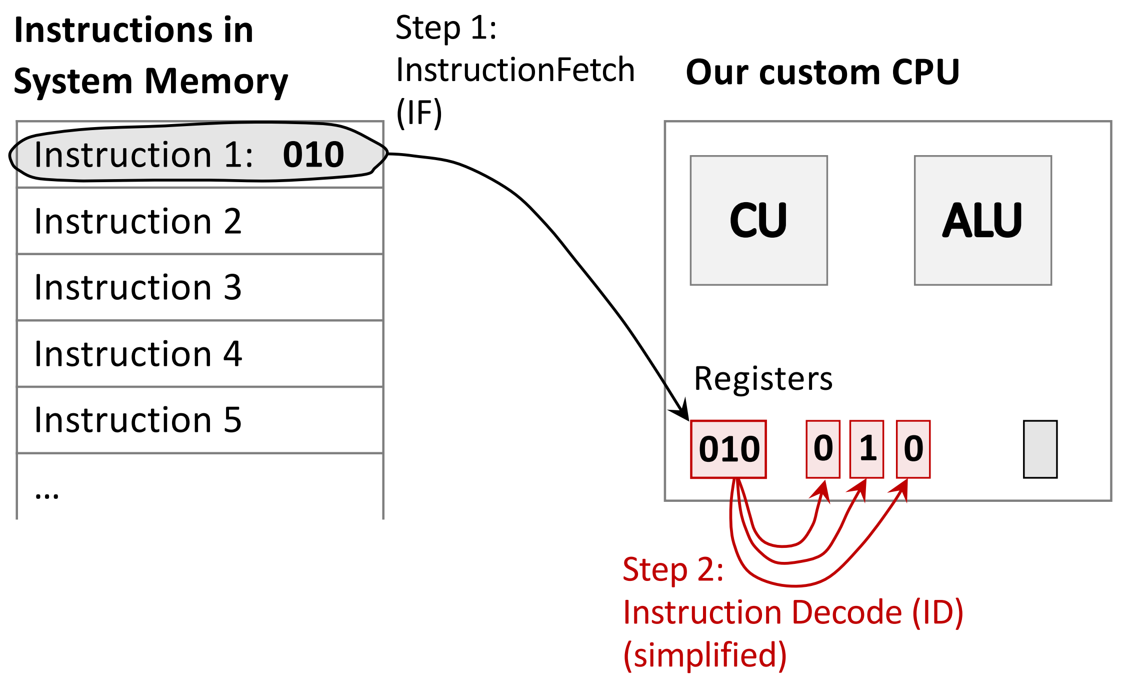
Each instruction is executed in several steps:



Fundamental (“Great”) Ideas of Computer Architecture

5. Performance via pipelining

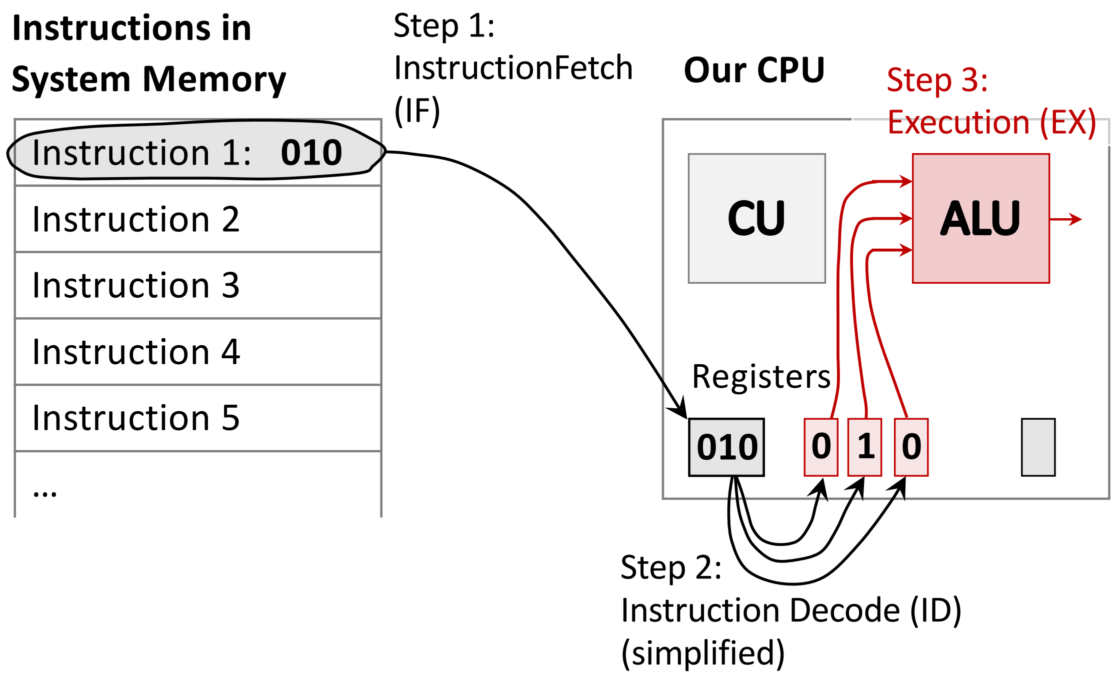
Each instruction is executed in several steps:



Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining

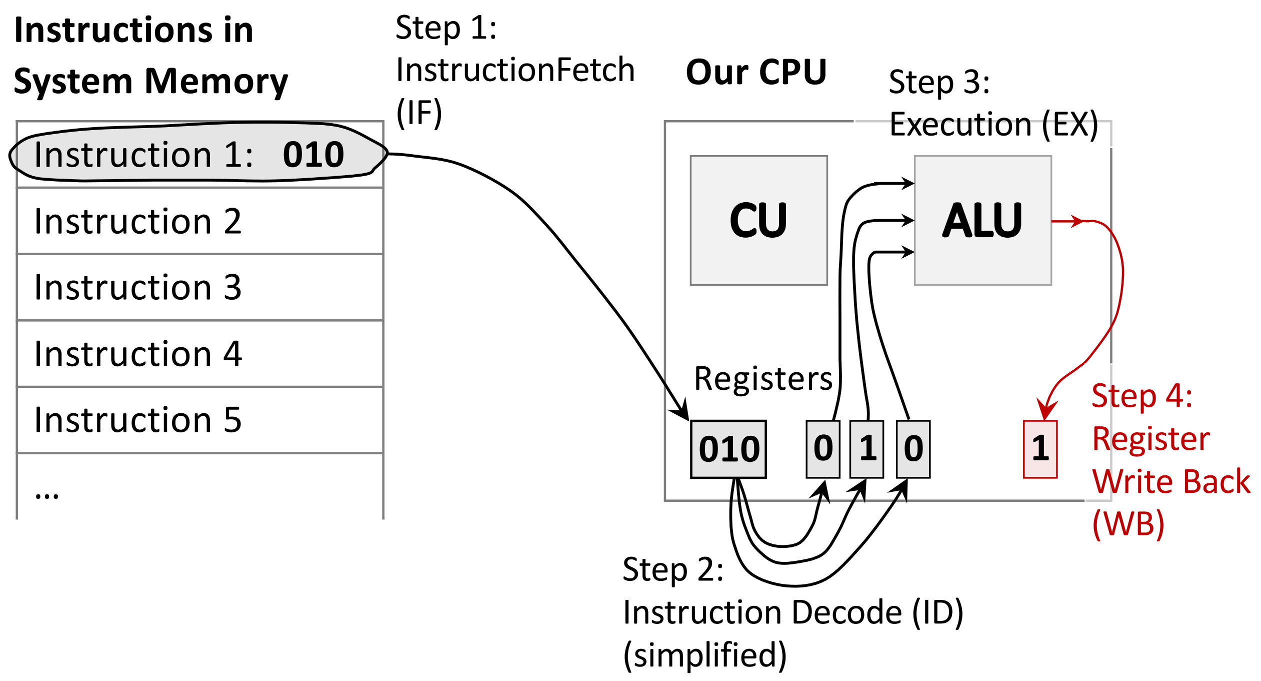
Each instruction is executed in several steps:



Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining

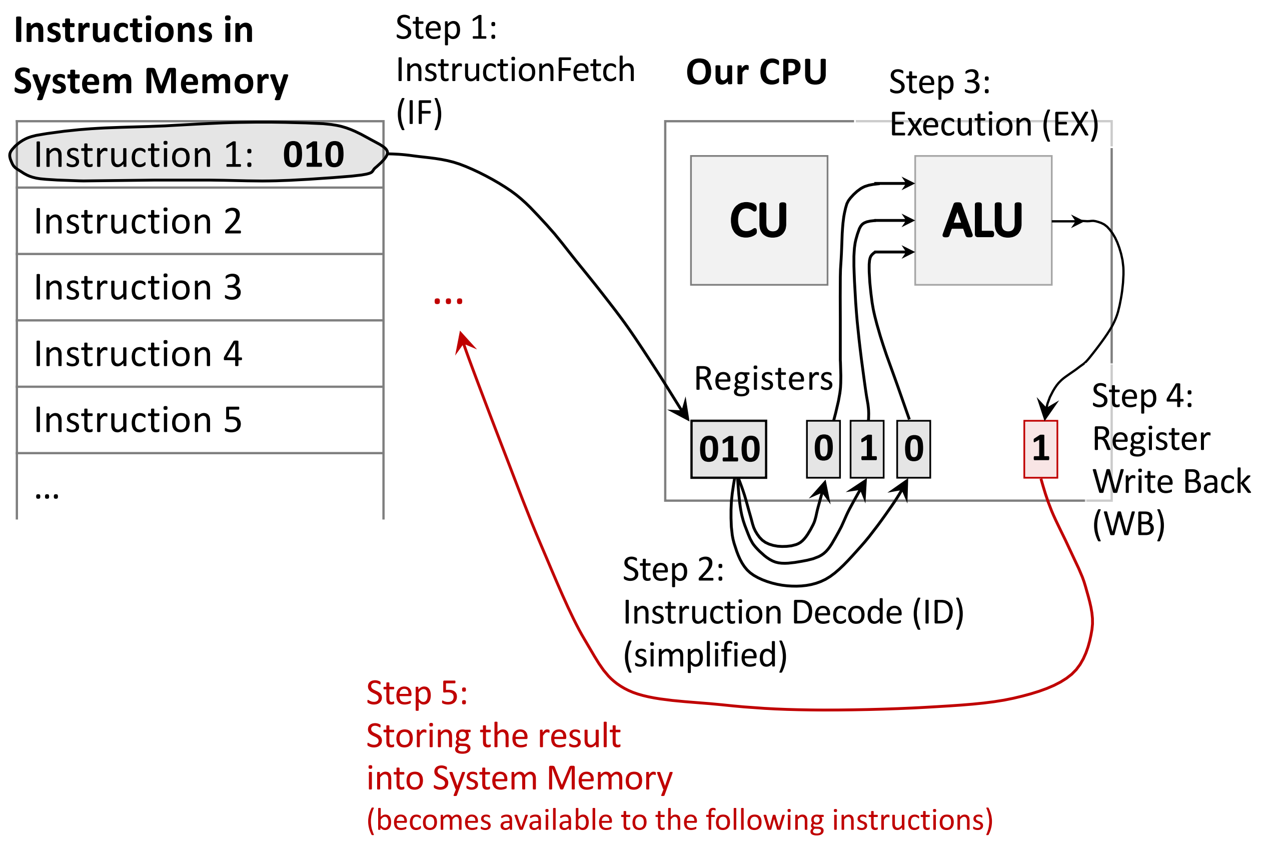
Each instruction is executed in several steps:


Instructions in System Memory
Instruction 1: 010
Instruction 2
Instruction 3
Instruction 4
Instruction 5
…
Step 1: InstructionFetch (IF)
Our CPU
CU
ALU
Step 3: Execution (EX)
Registers
010
0
1
0
1
Step 4: Register Write Back (WB)
Step 2: Instruction Decode (ID) (simplified)

Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining

Each instruction is executed in several steps:

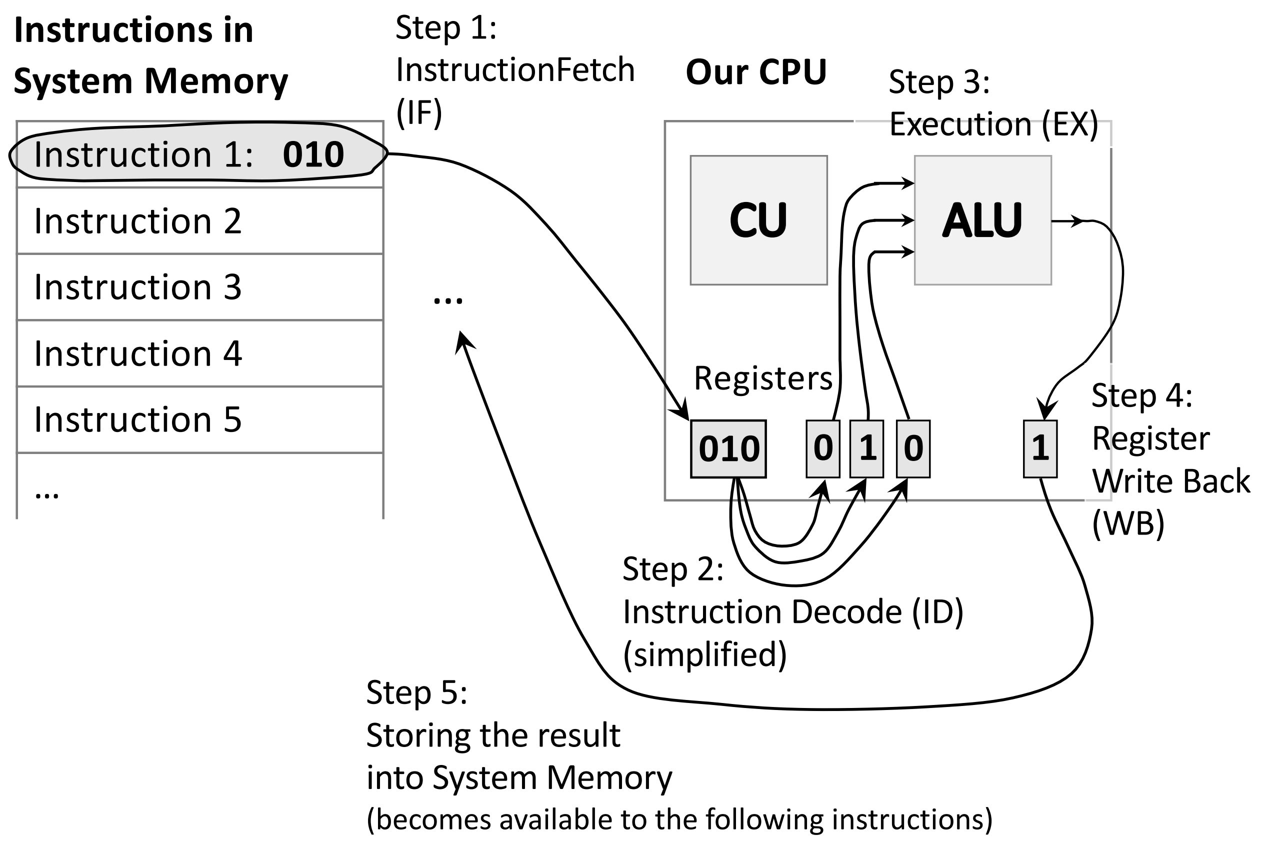


Instructions in System Memory
Instruction 1: 010
Instruction 2
Instruction 3
Instruction 4
Instruction 5
...
Step 1: InstructionFetch (IF)
...
Our CPU
Step 3: Execution (EX)
CU
ALU
Registers
010
0
1
0
1
Step 4: Register Write Back (WB)
Step 2: Instruction Decode (ID) (simplified)
Step 5: Storing the result into System Memory
(becomes available to the following instructions)

Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining

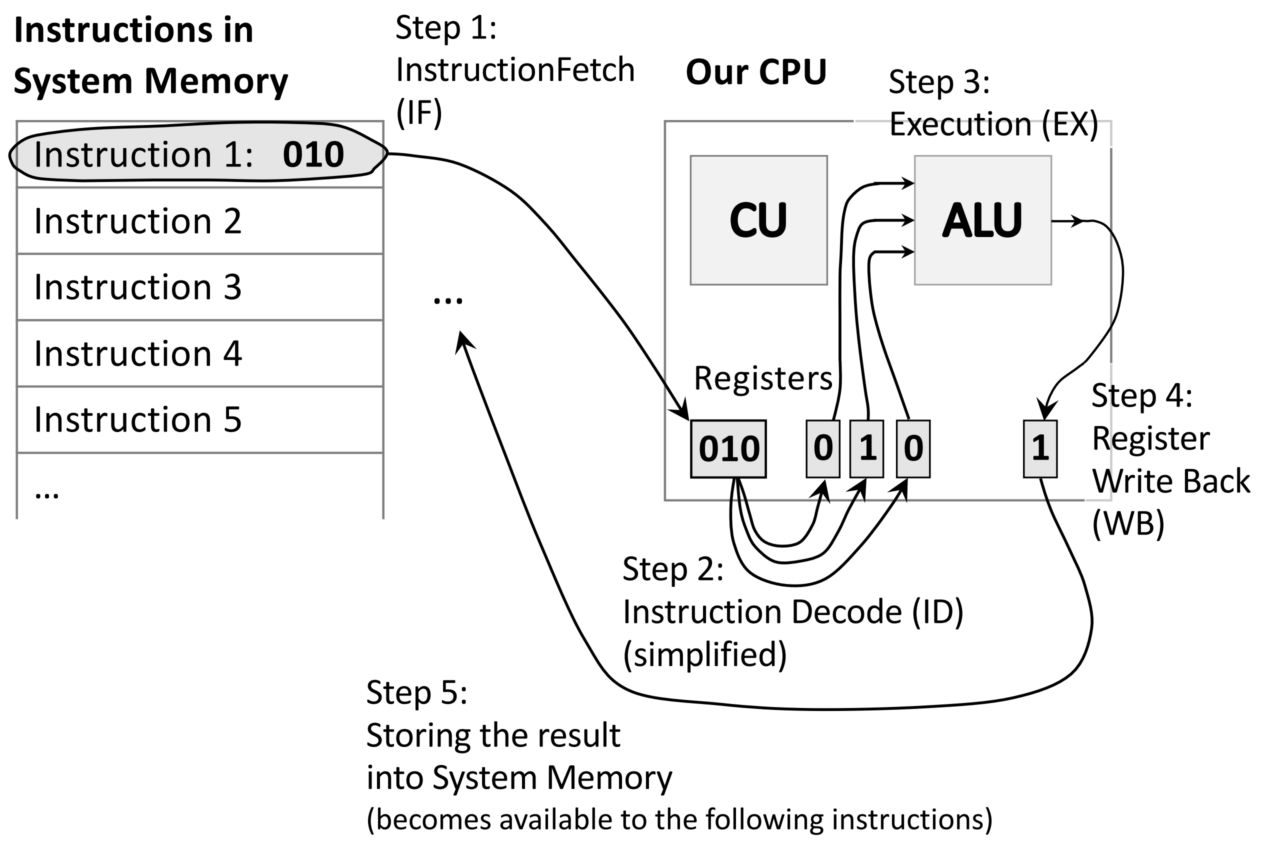
Each instruction is executed in several steps:


Instructions in System Memory
Instruction 1: 010
Instruction 2
Instruction 3
Instruction 4
Instruction 5
...
Step 1: InstructionFetch (IF)
...
Our CPU
CU
ALU
Step 3: Execution (EX)
Registers
010
0
1
0
1
Step 4: Register Write Back (WB)
Step 2: Instruction Decode (ID) (simplified)
Step 5: Storing the result into System Memory (becomes available to the following instructions)

Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining

Each instruction is executed in several steps:



Instructions in System Memory
Instruction 1: 010
Instruction 2
Instruction 3
Instruction 4
Instruction 5
...
Step 1: InstructionFetch (IF)
Our CPU
CU
ALU
Step 3: Execution (EX)
Registers
010
0
1
0
1
Step 4: Register Write Back (WB)
Step 2: Instruction Decode (ID) (simplified)
Step 5: Storing the result into System Memory (becomes available to the following instructions)
...

Idea of pipelining:

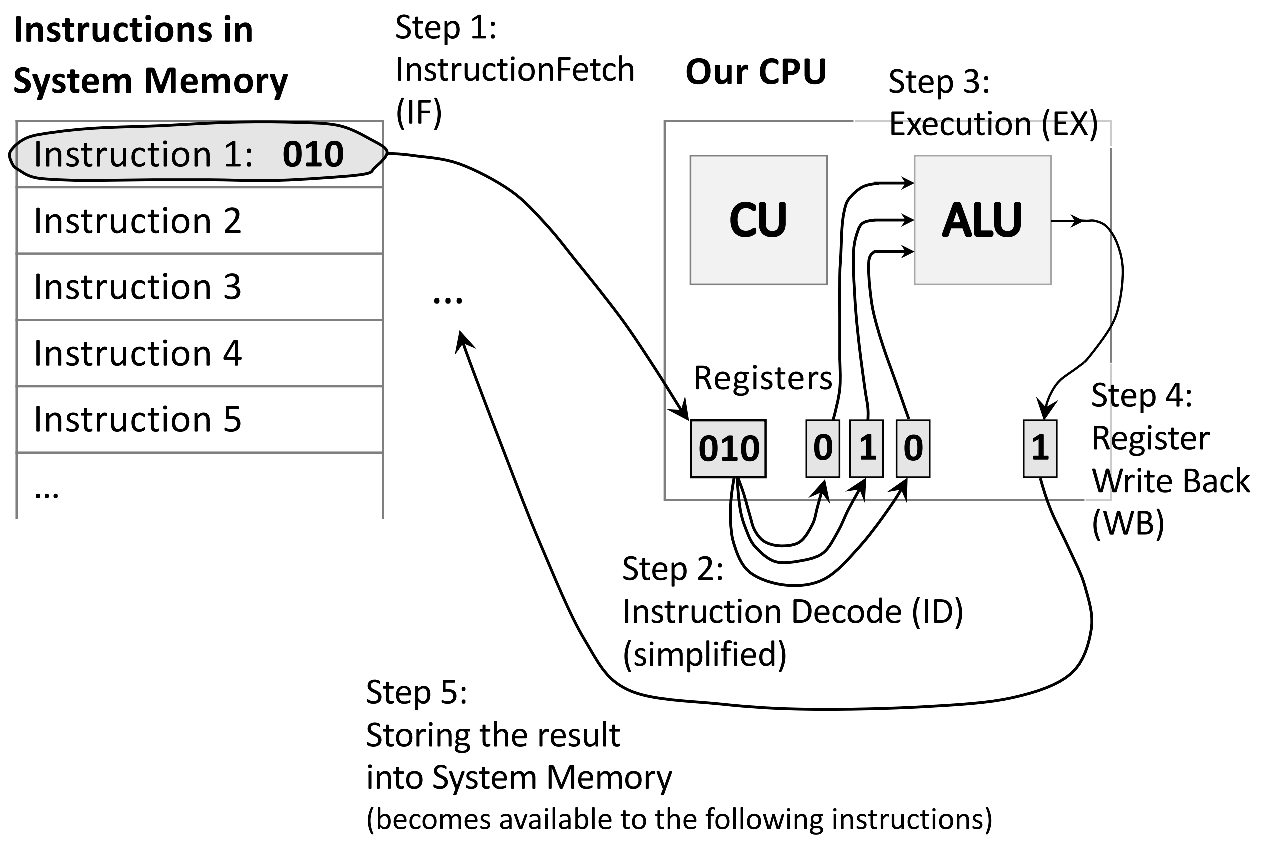
CPU might execute several instructions simultaneously, but each instruction at a different execution stage

Instr. No. \ Clock cycle	1	2	3	4	5	6	7
1							
2							
3							
4							
5							

Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining

Each instruction is executed in several steps:



Instructions in System Memory
Instruction 1: 010
Instruction 2
Instruction 3
Instruction 4
Instruction 5
…
Step 1: InstructionFetch (IF)
…
Our CPU
CU
ALU
Step 3: Execution (EX)
Registers
010
0
1
0
1
Step 4: Register Write Back (WB)
Step 2: Instruction Decode (ID) (simplified)
Step 5: Storing the result into System Memory (becomes available to the following instructions)

Idea of pipelining:

CPU might execute several instructions simultaneously, but each instruction at a different execution stage

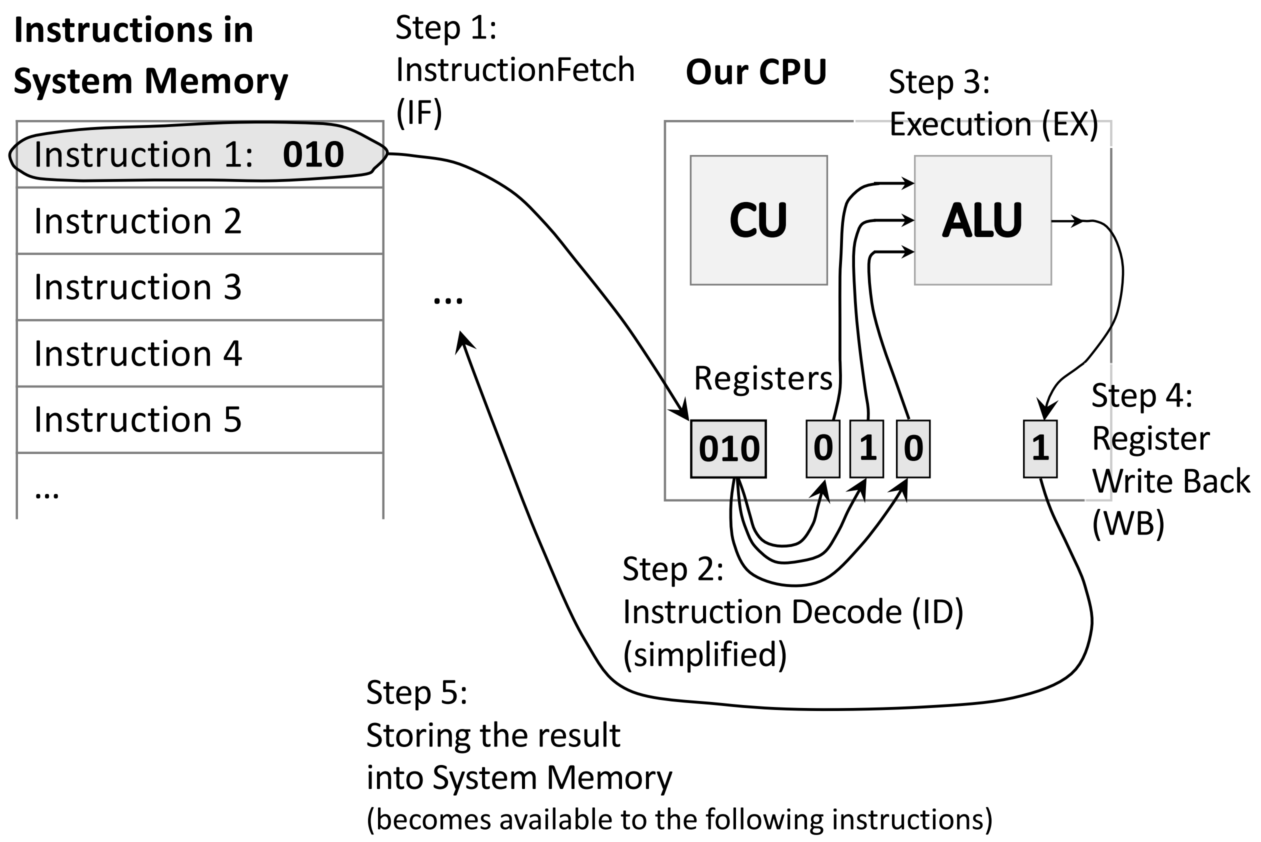
Instr. No. \ Clock cycle	1	2	3	4	5	6	7
1	IF						
2							
3							
4							
5							

IF: Instruction Fetch

Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining

Each instruction is executed in several steps:



Instructions in System Memory
Instruction 1: 010
Instruction 2
Instruction 3
Instruction 4
Instruction 5
...
Step 1: InstructionFetch (IF)
...
Our CPU
CU
ALU
Step 3: Execution (EX)
Registers
010
0
1
0
1
Step 4: Register Write Back (WB)
Step 2: Instruction Decode (ID) (simplified)
Step 5: Storing the result into System Memory
(becomes available to the following instructions)

Idea of pipelining:

CPU might execute several instructions simultaneously, but each instruction at a different execution stage

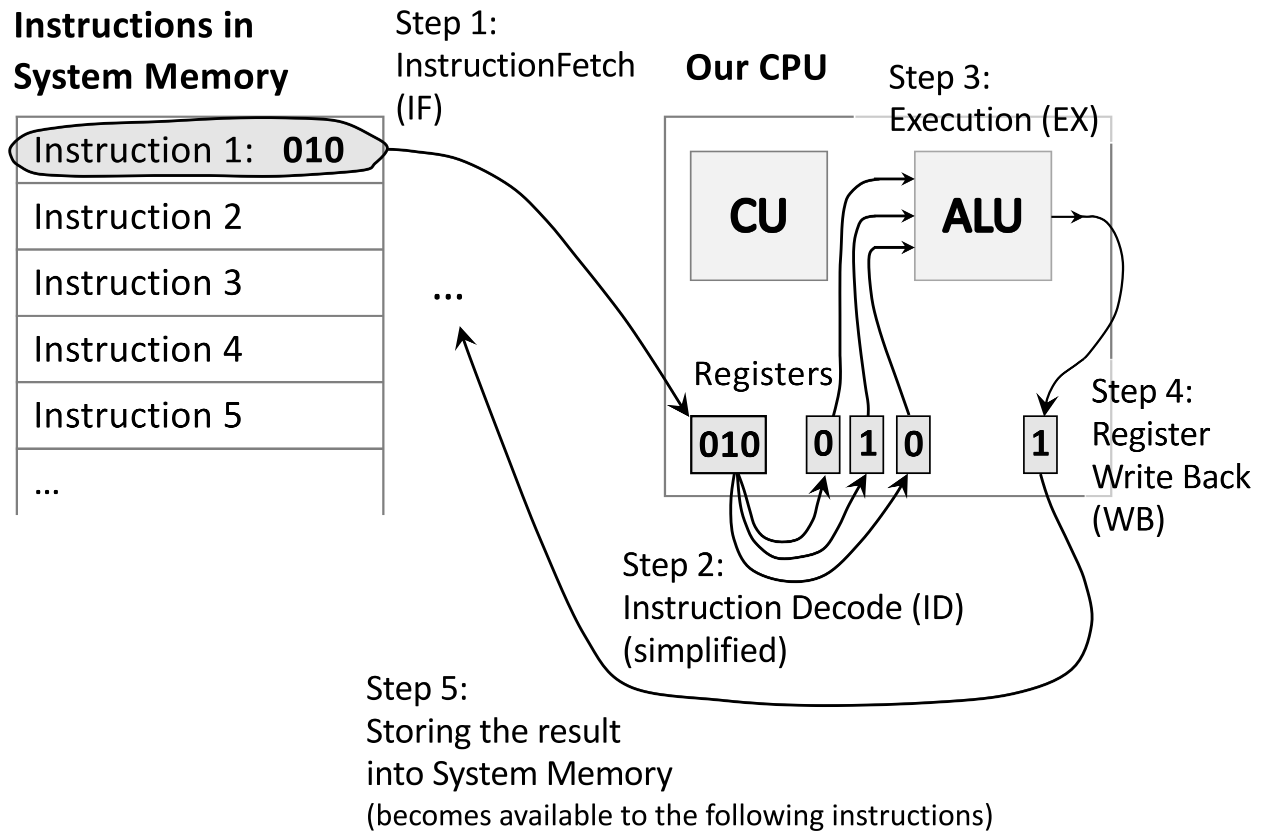
Instr. No. \ Clock cycle	1	2	3	4	5	6	7
1	IF	ID					
2		IF					
3							
4							
5							

IF: Instruction Fetch
ID: Instruction Decode

Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining

Each instruction is executed in several steps:


Instructions in System Memory
Instruction 1: 010
Instruction 2
Instruction 3
Instruction 4
Instruction 5
...
Step 1: InstructionFetch (IF)
Our CPU
CU
ALU
Step 3: Execution (EX)
Registers
010
0
1
0
1
Step 4: Register Write Back (WB)
Step 2: Instruction Decode (ID) (simplified)
Step 5: Storing the result into System Memory (becomes available to the following instructions)
...

Idea of pipelining:

CPU might execute several instructions simultaneously, but each instruction at a different execution stage

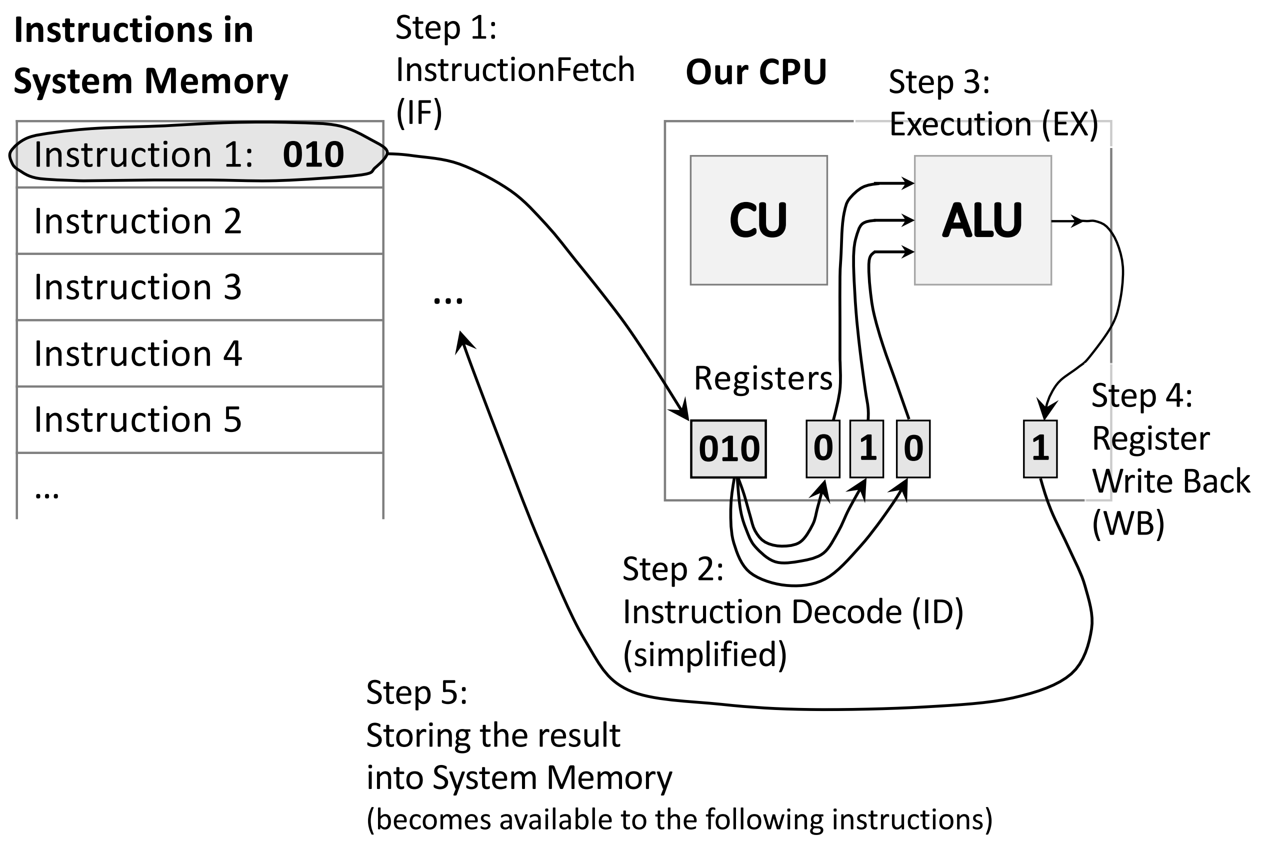
Instr. No. \ Clock cycle	1	2	3	4	5	6	7
1	IF	ID					
2		IF					
3							
4							
5							

IF: Instruction Fetch
ID: Instruction Decode

Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining

Each instruction is executed in several steps:



Instructions in System Memory
Instruction 1: 010
Instruction 2
Instruction 3
Instruction 4
Instruction 5
...
Step 1: InstructionFetch (IF)
...
Our CPU
CU
ALU
Step 3: Execution (EX)
Registers
010
0
1
0
1
Step 4: Register Write Back (WB)
Step 2: Instruction Decode (ID) (simplified)
Step 5: Storing the result into System Memory (becomes available to the following instructions)

Idea of pipelining:

CPU might execute several instructions simultaneously, but each instruction at a different execution stage

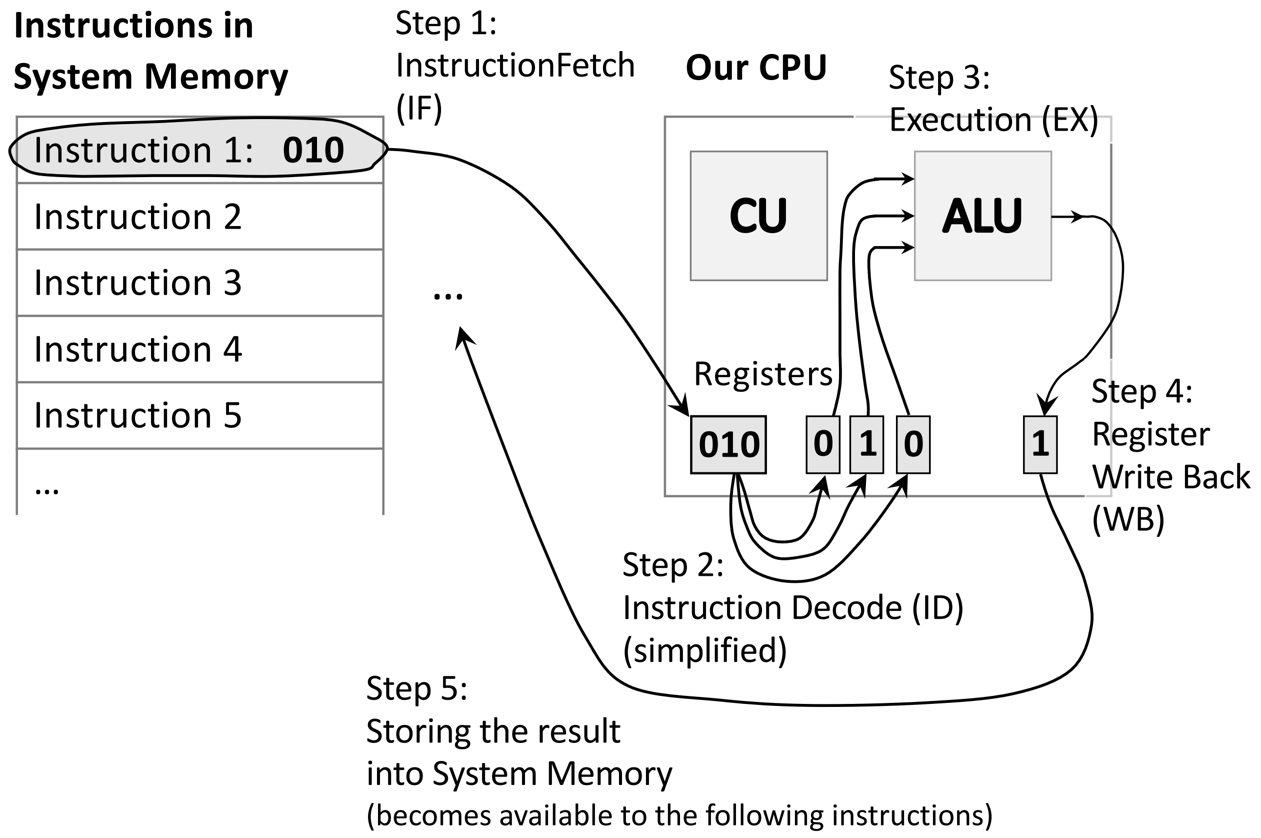
Instr. No. \ Clock cycle	1	2	3	4	5	6	7
1	IF	ID	EX				
2		IF	ID				
3			IF				
4							
5							

IF: Instruction Fetch
ID: Instruction Decode
EX: Execution

Fundamental ("Great") Ideas of Computer Architecture

5. Performance via pipelining

Each instruction is executed in several steps:



Instructions in System Memory
Instruction 1: 010
Instruction 2
Instruction 3
Instruction 4
Instruction 5
…
Step 1: InstructionFetch (IF)
Our CPU
Step 3: Execution (EX)
CU
ALU
…
Registers
010
0
1
0
1
Step 4: Register Write Back (WB)
Step 2: Instruction Decode (ID) (simplified)
Step 5: Storing the result into System Memory (becomes available to the following instructions)

Idea of pipelining:

CPU might execute several instructions simultaneously, but each instruction at a different execution stage

Instr. No. \ Clock cycle	1	2	3	4	5	6	7
1	**IF**	**ID**	**EX**	**MEM**	**WB**		
2		**IF**	**ID**	**EX**	**MEM**	**WB**	
3			**IF**	**ID**	**EX**	**MEM**	**WB**
4				**IF**	**ID**	**EX**	**MEM**
5					**IF**	**ID**	**EX**

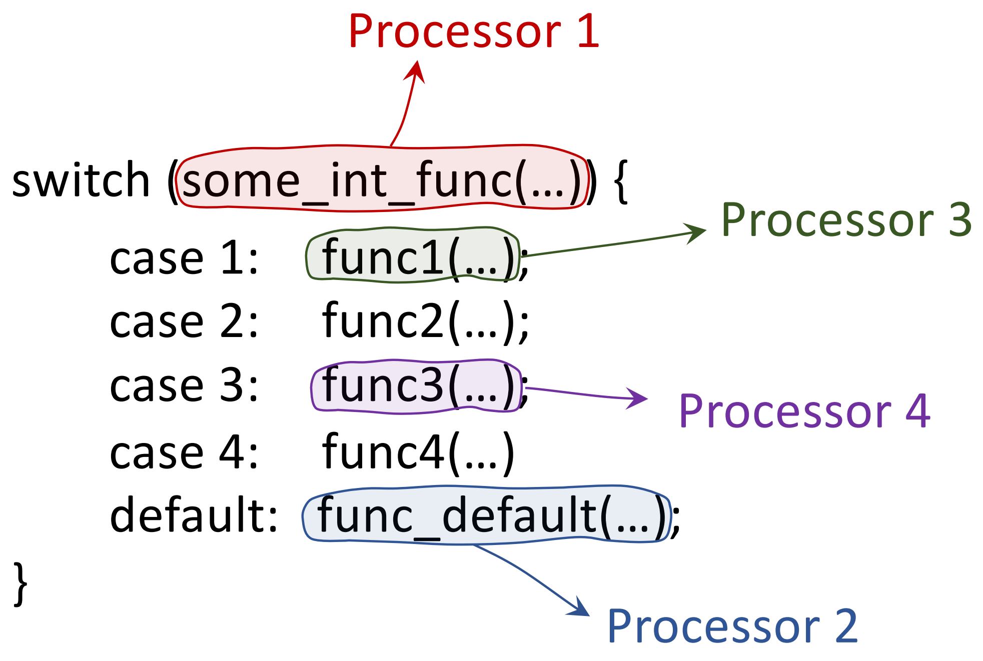
IF: Instruction Fetch
ID: Instruction Decode
EX: Execution
MEM: Memory access
WB: register Write Back

Fundamental ("Great") Ideas of Computer Architecture

1. Hierarchy of memories
2. Use abstraction to simplify design
3. Design for Moore's law
4. Performance via parallelism
5. Performance via pipelining
6. Performance via speculation (prediction)

C++ code sample example:

Assumptions:
- 4 processors are available;
- No dependencies between calling functions



```
switch (some_int_func(…)) {
        case 1:   func1(…);
        case 2:   func2(…);
        case 3:   func3(…);
        case 4:   func4(…)
        default:  func_default(…);
}
```

For this specific code, we try to predict ("speculate") 3 most possible cases to be executed

Fundamental (“Great”) Ideas of Computer Architecture

1. Hierarchy of memories
2. Use abstraction to simplify design
3. Design for Moore’s law
4. Performance via parallelism
5. Performance via pipelining
6. Performance via speculation (prediction)
7. Dependability (reliability) via redundancy

Some redundant (or “spare”) components are introduced (e.g. CPU or memory unit), to increase the reliability of a computer platform (e.g. in a spacecraft);

Fundamental (“Great”) Ideas of Computer Architecture

1. Hierarchy of memories
2. Use abstraction to simplify design
3. Design for Moore’s law
4. Performance via parallelism
5. Performance via pipelining
6. Performance via speculation (prediction)
7. Dependability (reliability) via redundancy
8. Make the common case fast

   The computer platform should be optimized for the most common use case expected

Fundamental ("Great") Ideas of Computer Architecture

1. Hierarchy of memories
2. Use abstraction to simplify design
3. Design for Moore's law
4. Performance via parallelism
5. Performance via pipelining
6. Performance via speculation (prediction)
7. Dependability (reliability) via redundancy
8. Make the common case fast
9. State machines

State machine – a set of states and transitions between them; a convenient way to model systems behaviour

A sample state machine

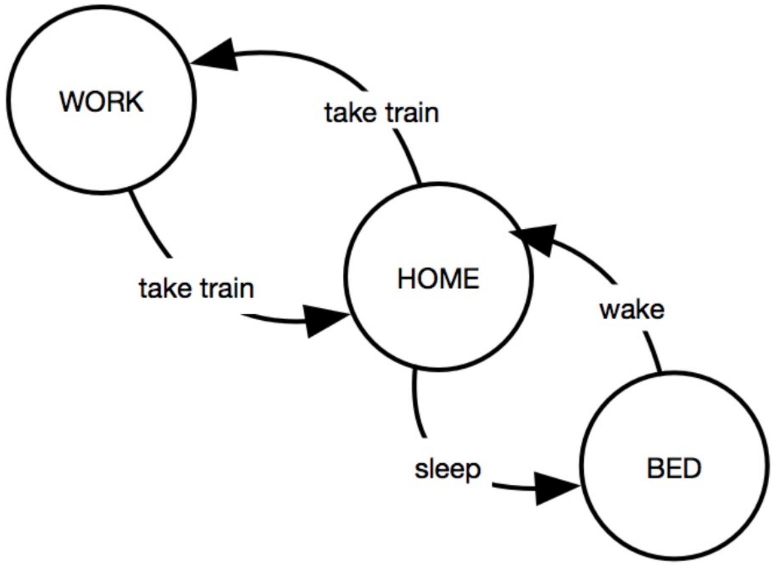


Image is taken from *https://dwarves.foundation/n/finite-state-machine/*

Fundamental (“Great”) Ideas of Computer Architecture

1. Hierarchy of memories
2. Use abstraction to simplify design
3. Design for Moore’s law
4. Performance via parallelism
5. Performance via pipelining
6. Performance via speculation (prediction)
7. Dependability (reliability) via redundancy
8. Make the common case fast
9. State machines