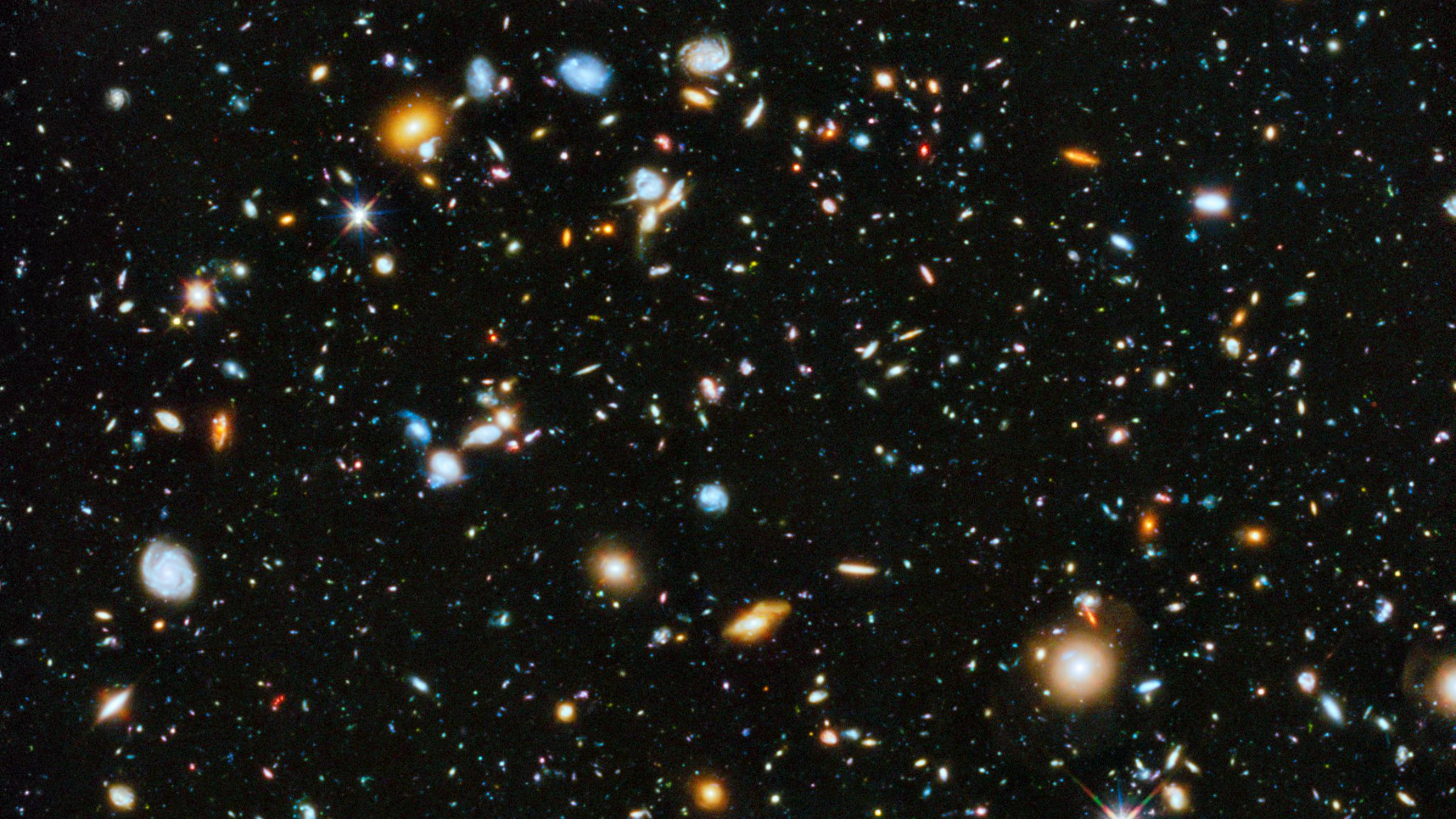
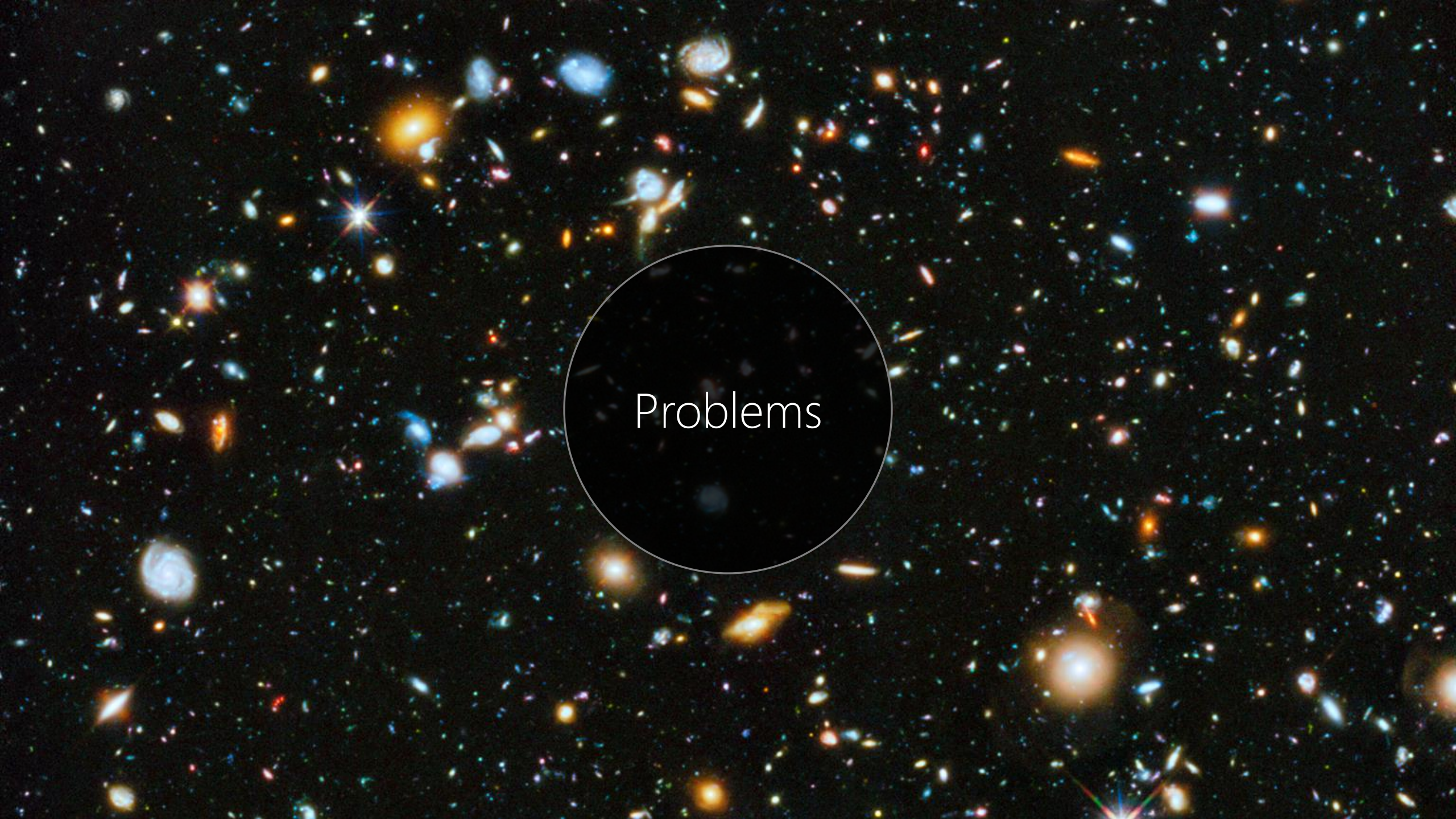
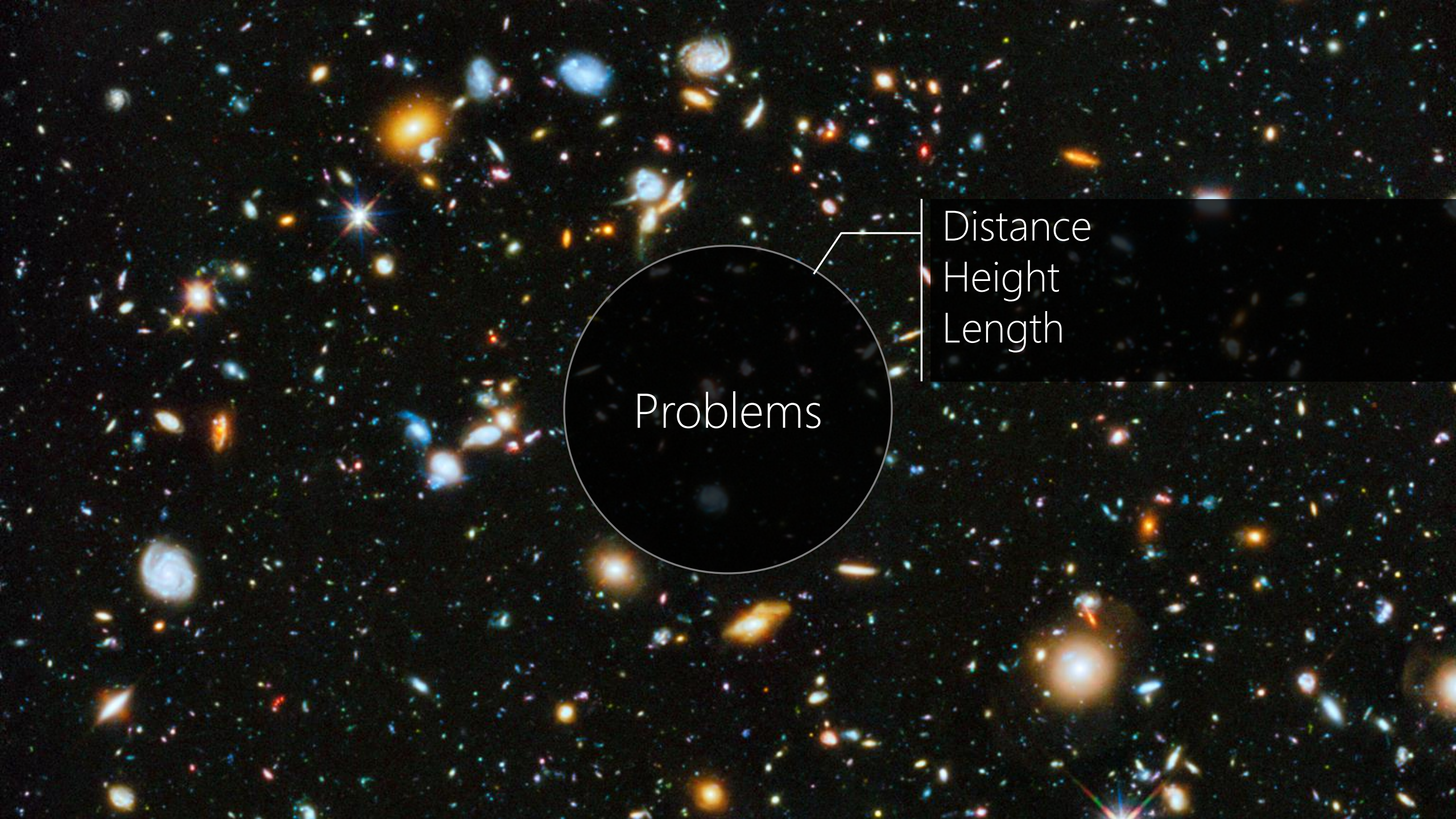
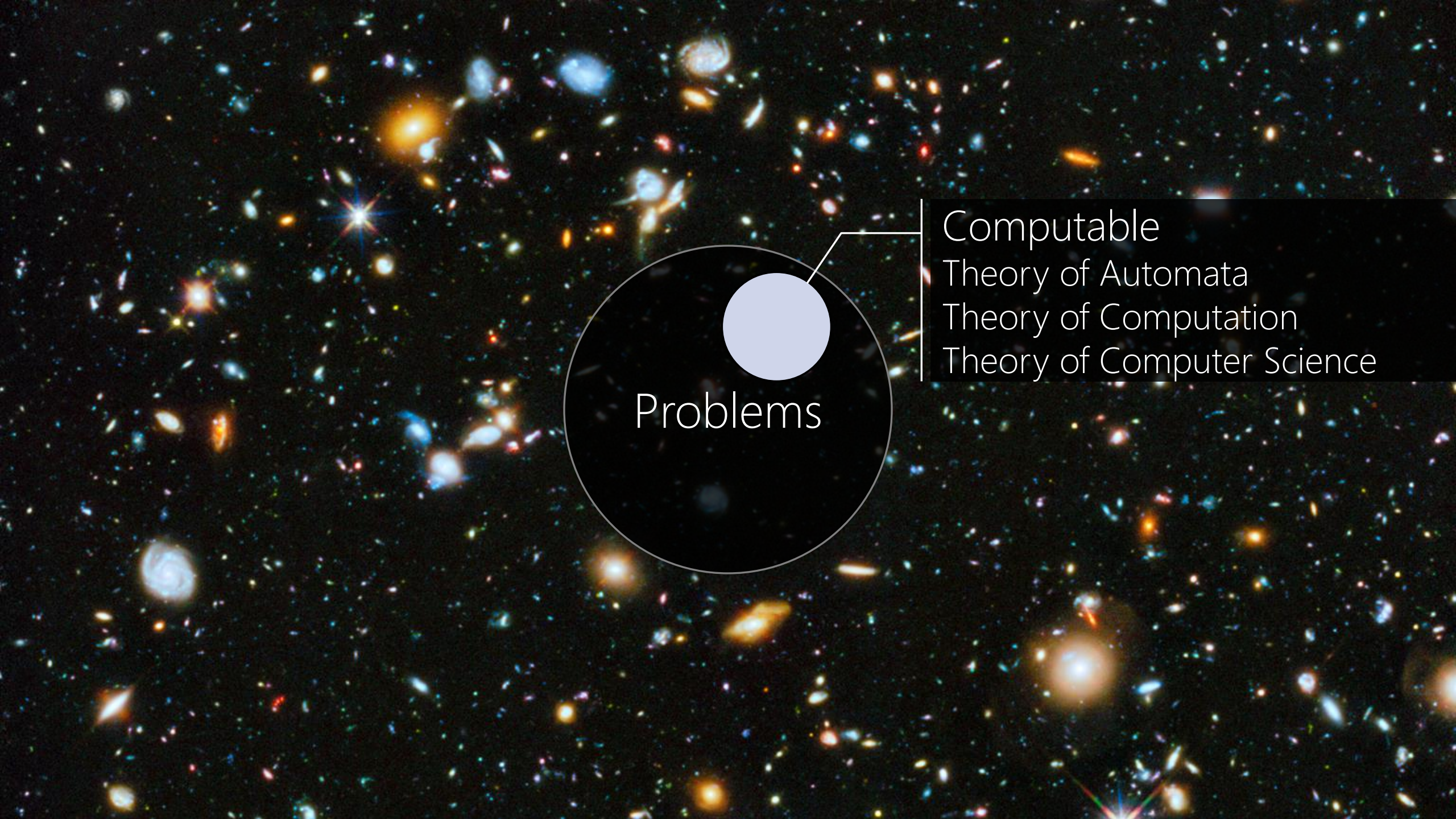
The Bonnie Situation, Pulp Fiction (1994), Quentin Tarantino

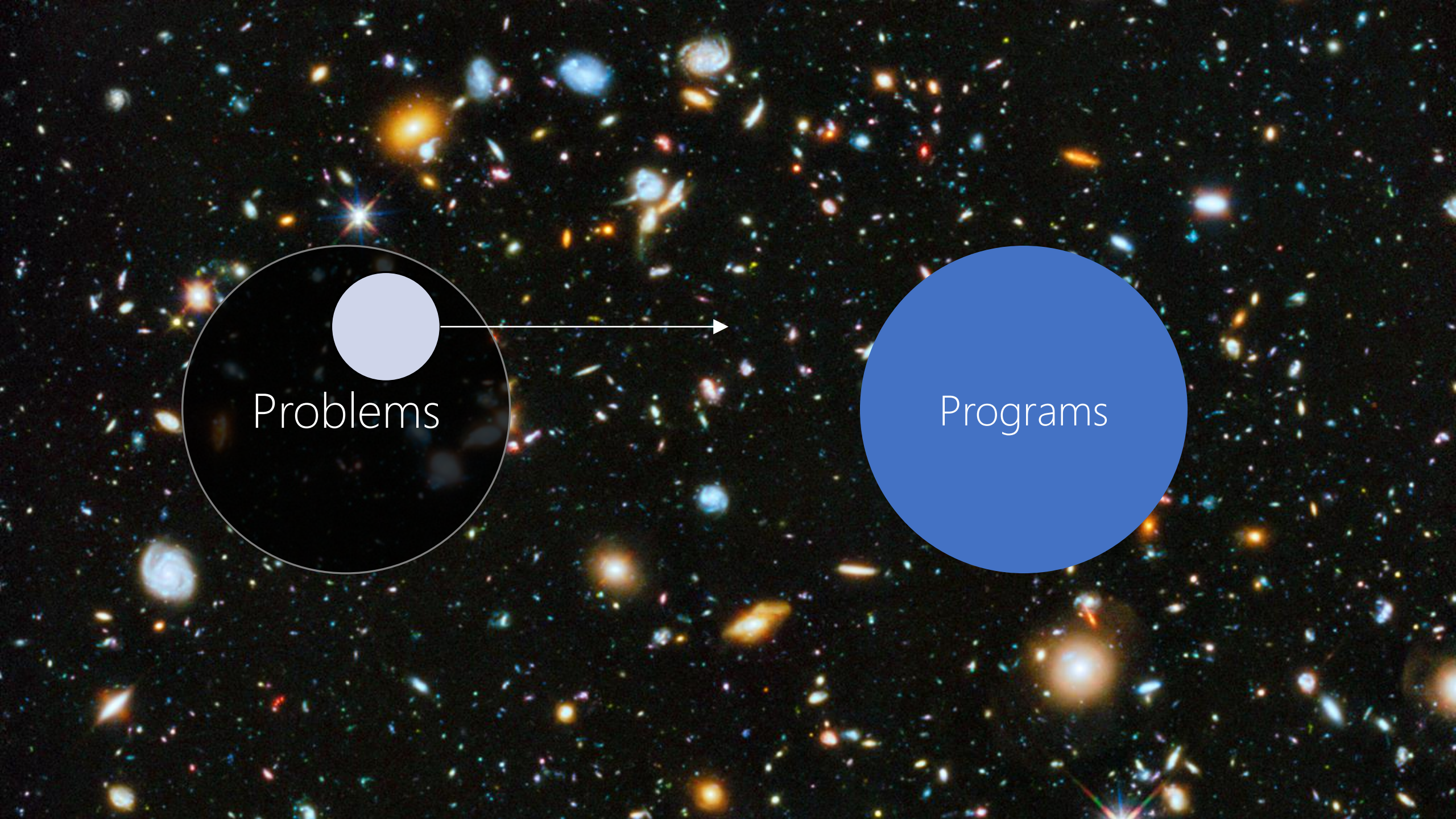


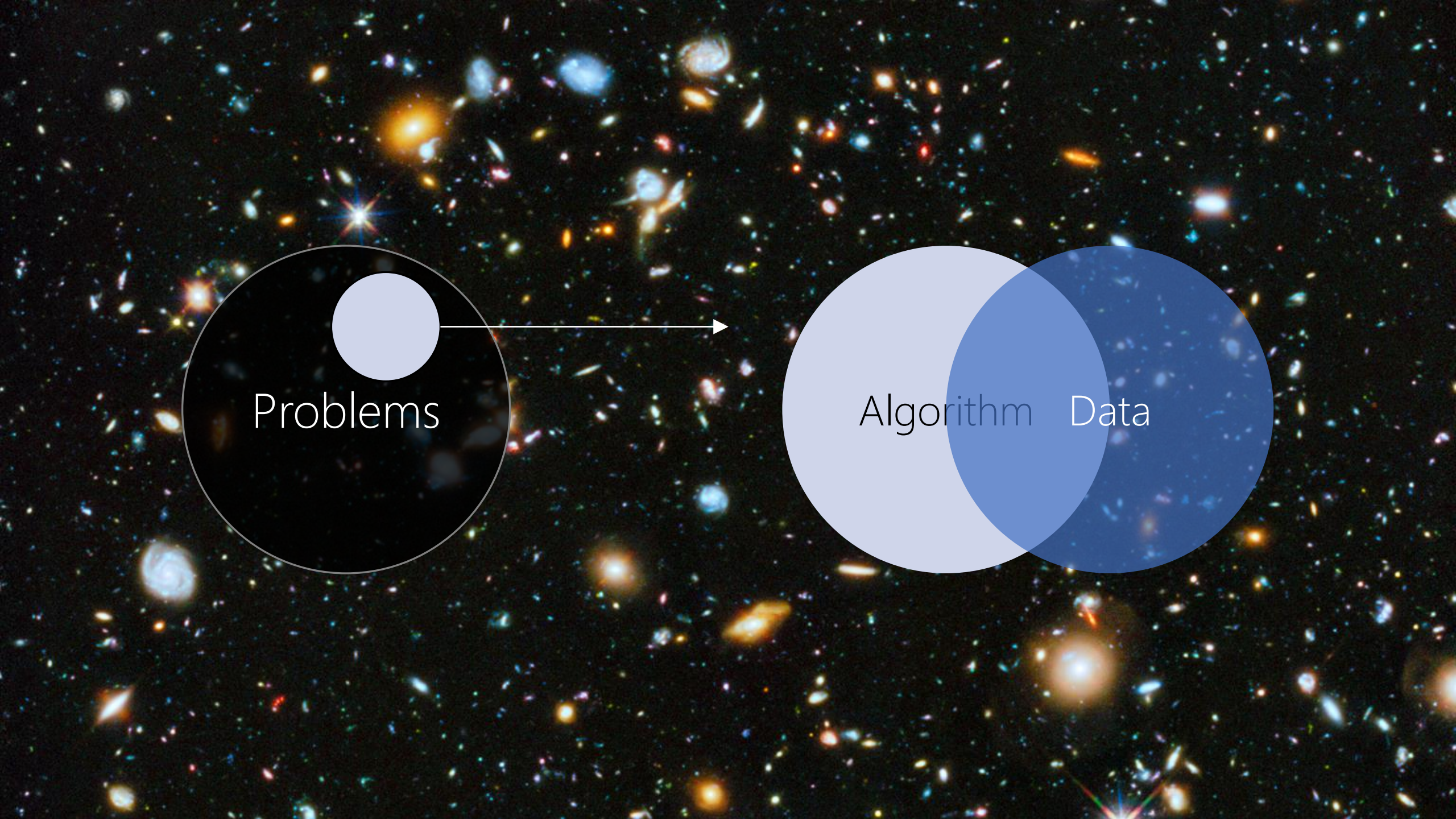

HUBBLE UNVEILS ITS MOST COLORFUL VIEW OF THE UNIVERSE
(ZOOM AND PAN)
BROWSE THE LINK BELOW AND PLAY!
https://hubblesite.org/contents/media/videos/2014/27/766-Video.html


Problems


Problems
Distance
Height
Length


Problems
Computable
Theory of Automata
Theory of Computation
Theory of Computer Science

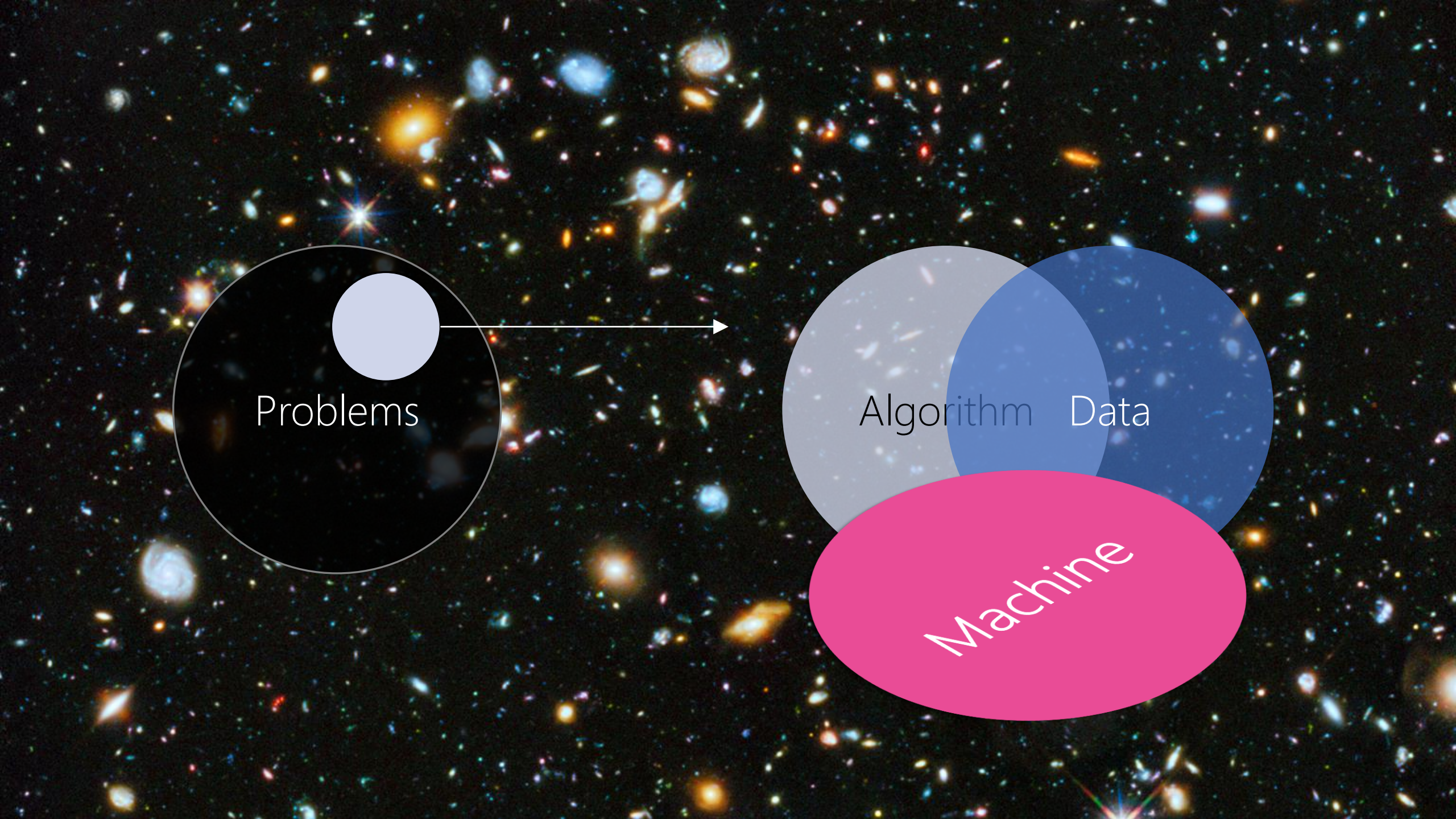

Problems
Programs


Problems
Algorithm
Data

Euclid – 300 BCE
Greatest Common Divisor

Archimedes – 212 BCE
Approximate π

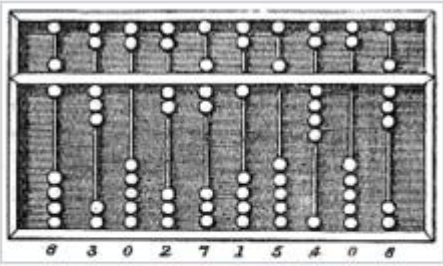
Khwārizmī – 850 AD
Linear and Quadratic Equations


Problems
Algorithm
Data
Machine

https://en.wikipedia.org/wiki/Computer#History

The Ishango bone, a bone tool dating back to prehistoric Africa.

2700 – 2300 BC

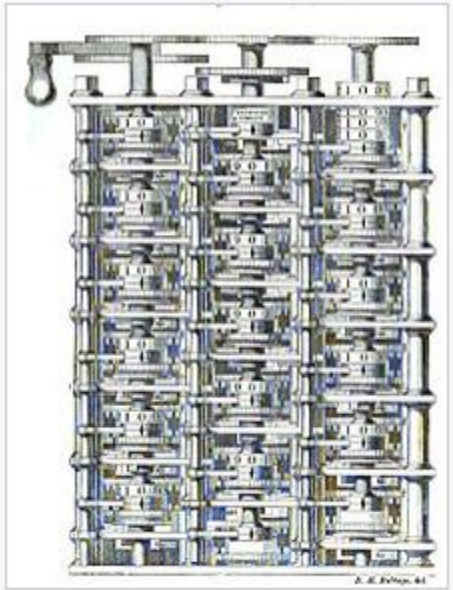


The Chinese suanpan (算盘). The number represented on this abacus is 6,302,715,408.

100 BC

The Antikythera mechanism, dating back to ancient Greece circa 150–100 BC, is an early analog computing device.

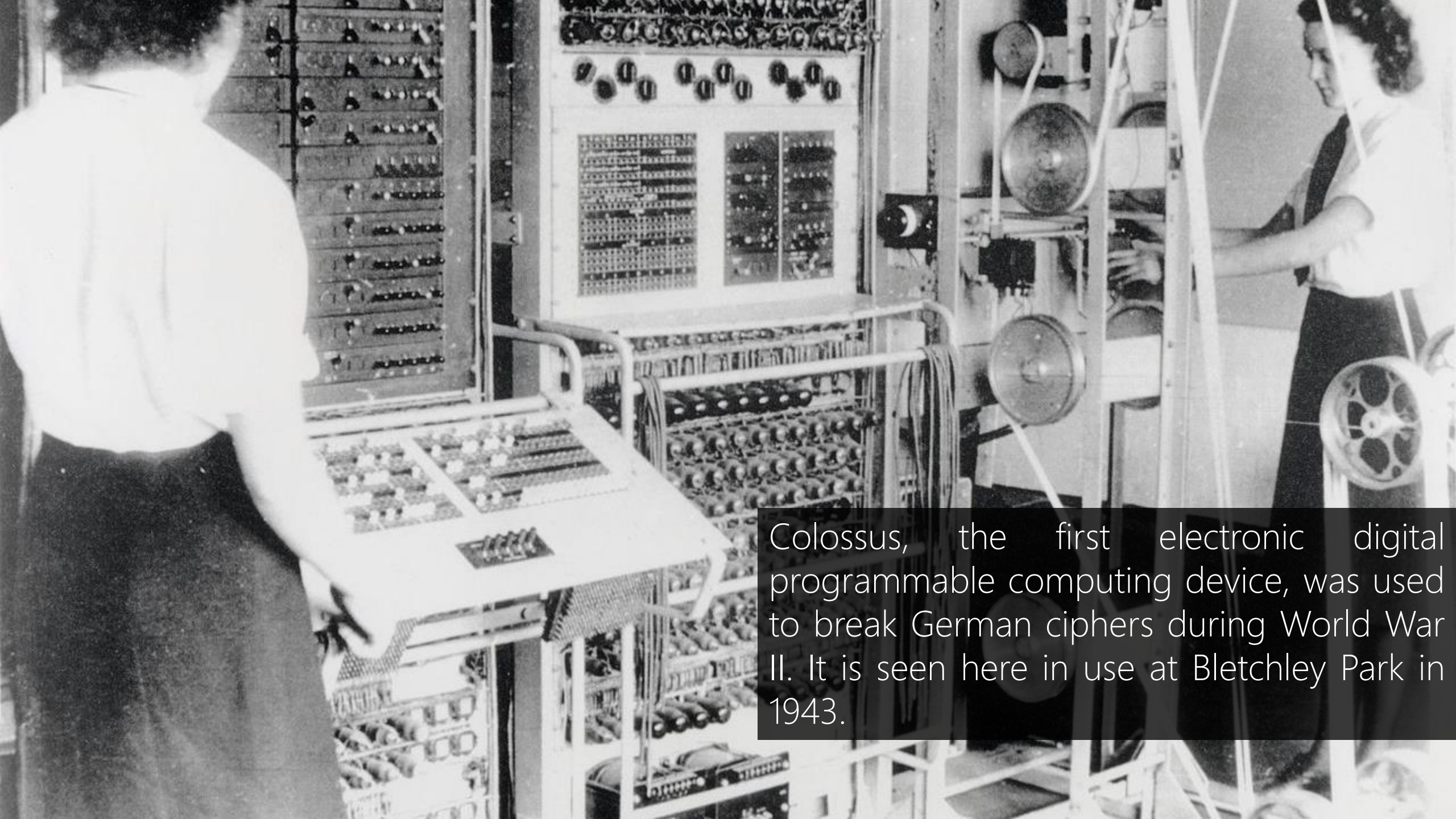
1833



A portion of Babbage's Difference engine.

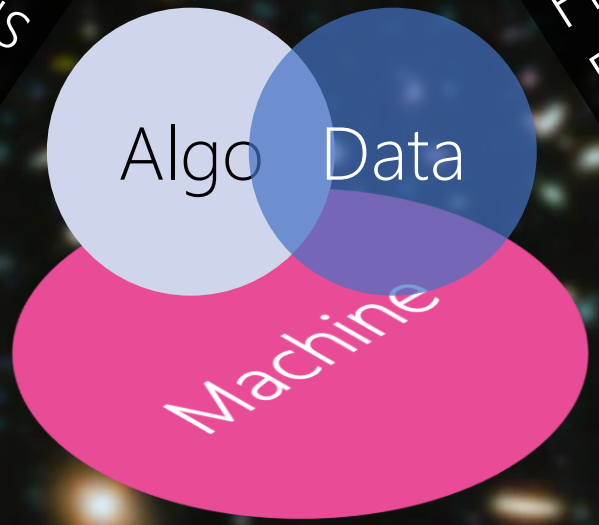
1941

Replica of Zuse's Z3, the first fully automatic, digital (electromechanical) computer.



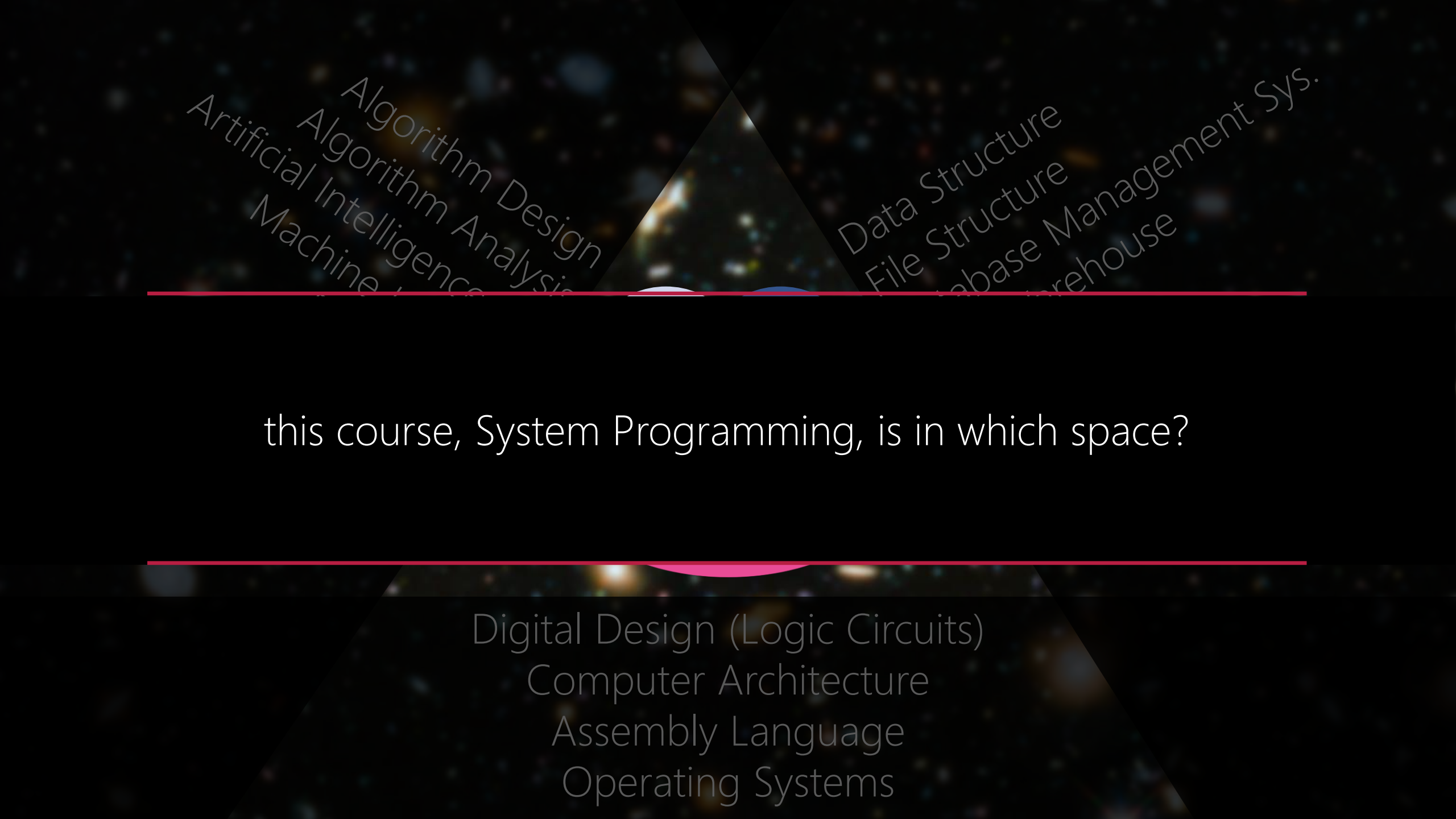
Colossus, the first electronic digital programmable computing device, was used to break German ciphers during World War II. It is seen here in use at Bletchley Park in 1943.

Algorithm Design
Algorithm Analysis
Artificial Intelligence (AI)
Machine Learning
Data Mining



Data Structure
File Structure
Database Management Sys.
Data Warehouse
Big Data
Cloud

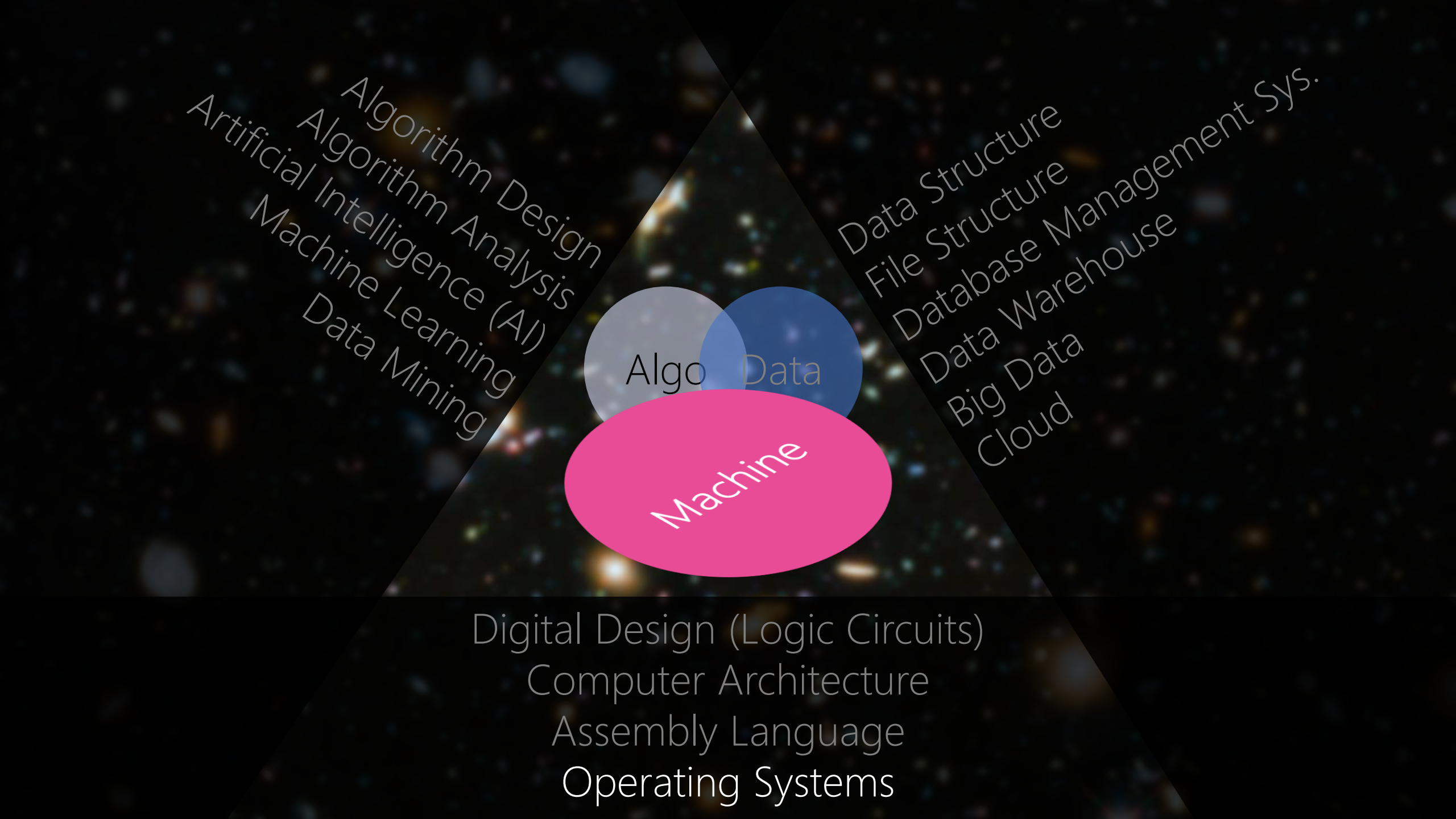
Digital Design (Logic Circuits)
Computer Architecture
Assembly Language
Operating Systems


Algorithm Design
Algorithm Analysis
Artificial Intelligence
Machine
Data Structure
File Structure
Management Sys.
this course, System Programming, is in which space?
Digital Design (Logic Circuits)
Computer Architecture
Assembly Language
Operating Systems

Algorithm Design
Algorithm Analysis
Artificial Intelligence (AI)
Machine Learning
Data Mining

Data Structure
File Structure
Database Management Sys.
Data Warehouse
Big Data
Cloud

Digital Design (Logic Circuits)
Computer Architecture
Assembly Language
Operating Systems


Algorithm Design
Algorithm Analysis
Artificial Intelligence (AI)
Machine Learning
Data Mining
Data Structure
File Structure
Database Management Sys.
Data Warehouse
Big Data
Cloud
Algo
Data
Machine
Digital Design (Logic Circuits)
Computer Architecture
Assembly Language
Operating Systems

John von Neumann

(/vɒn ˈnɔɪmən/)

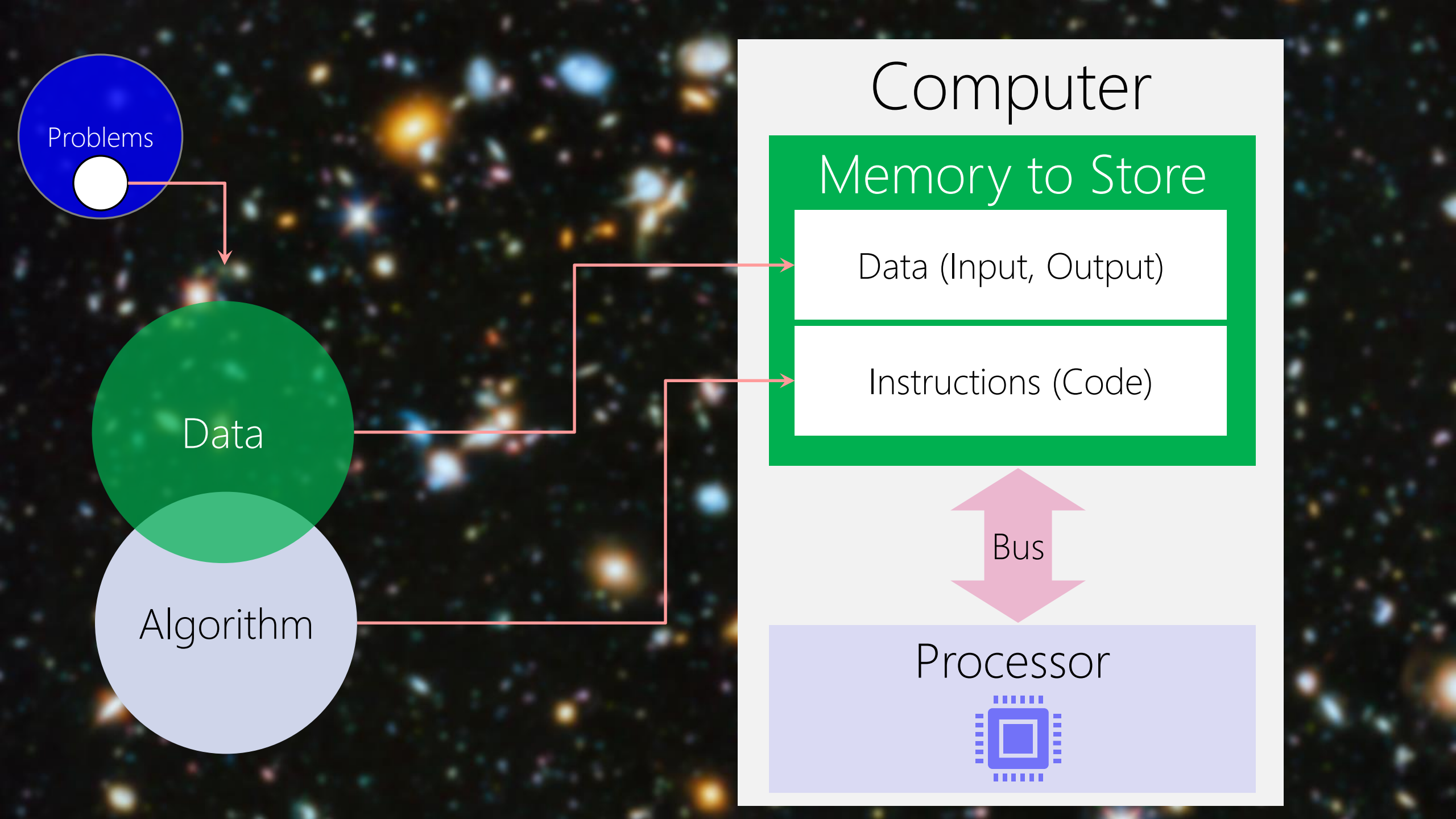
1903 –1957

Mathematician, Physicist, Computer Scientist, Engineer

Polymath

He integrated pure and applied sciences. He made major contributions to many fields, including:

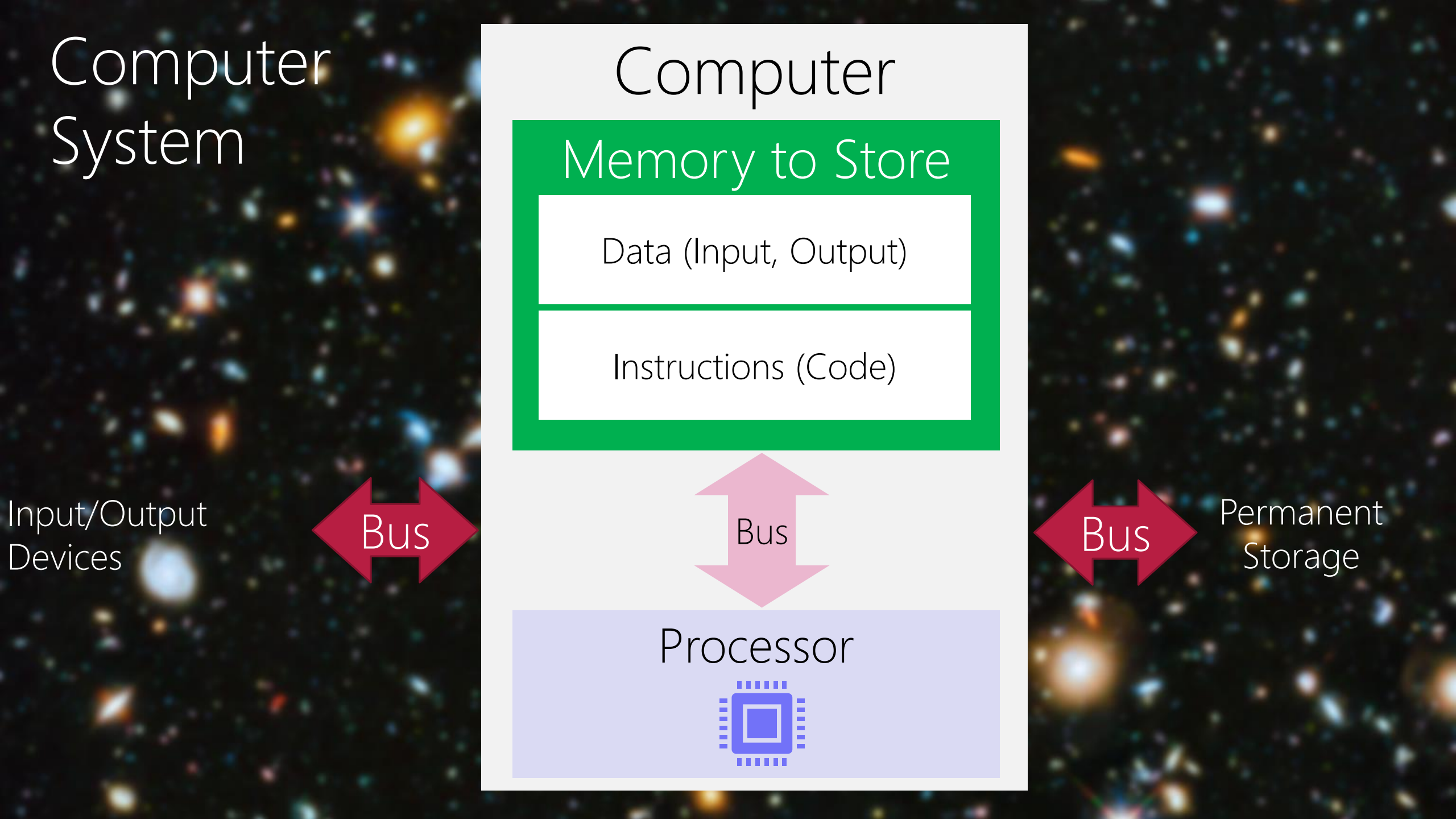
- Mathematics
- Physics
- Economics (game theory)
- Computing
- Statistics

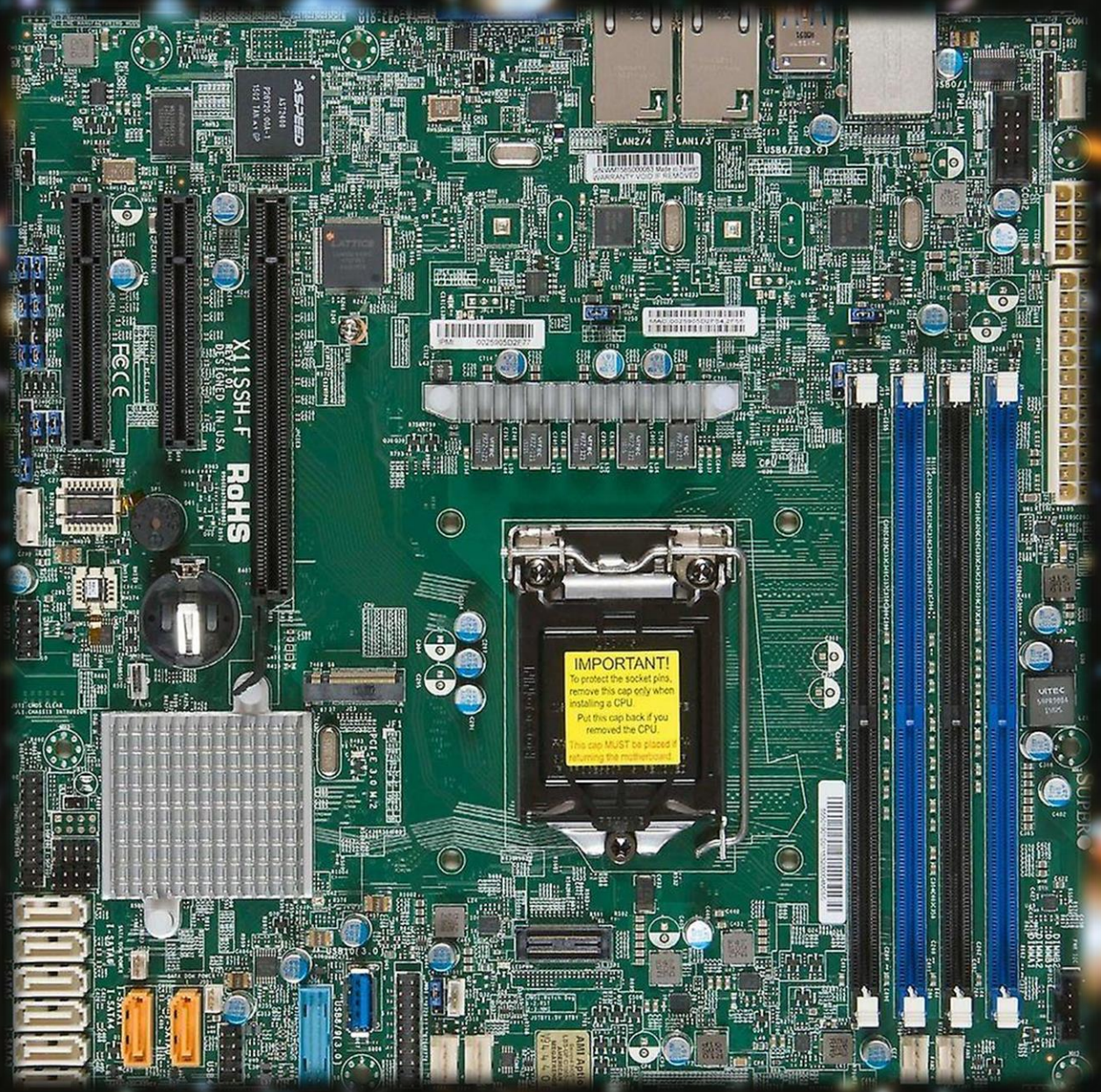

Problems
Data
Algorithm
Computer
Memory to Store
Data (Input, Output)
Instructions (Code)
Bus
Processor

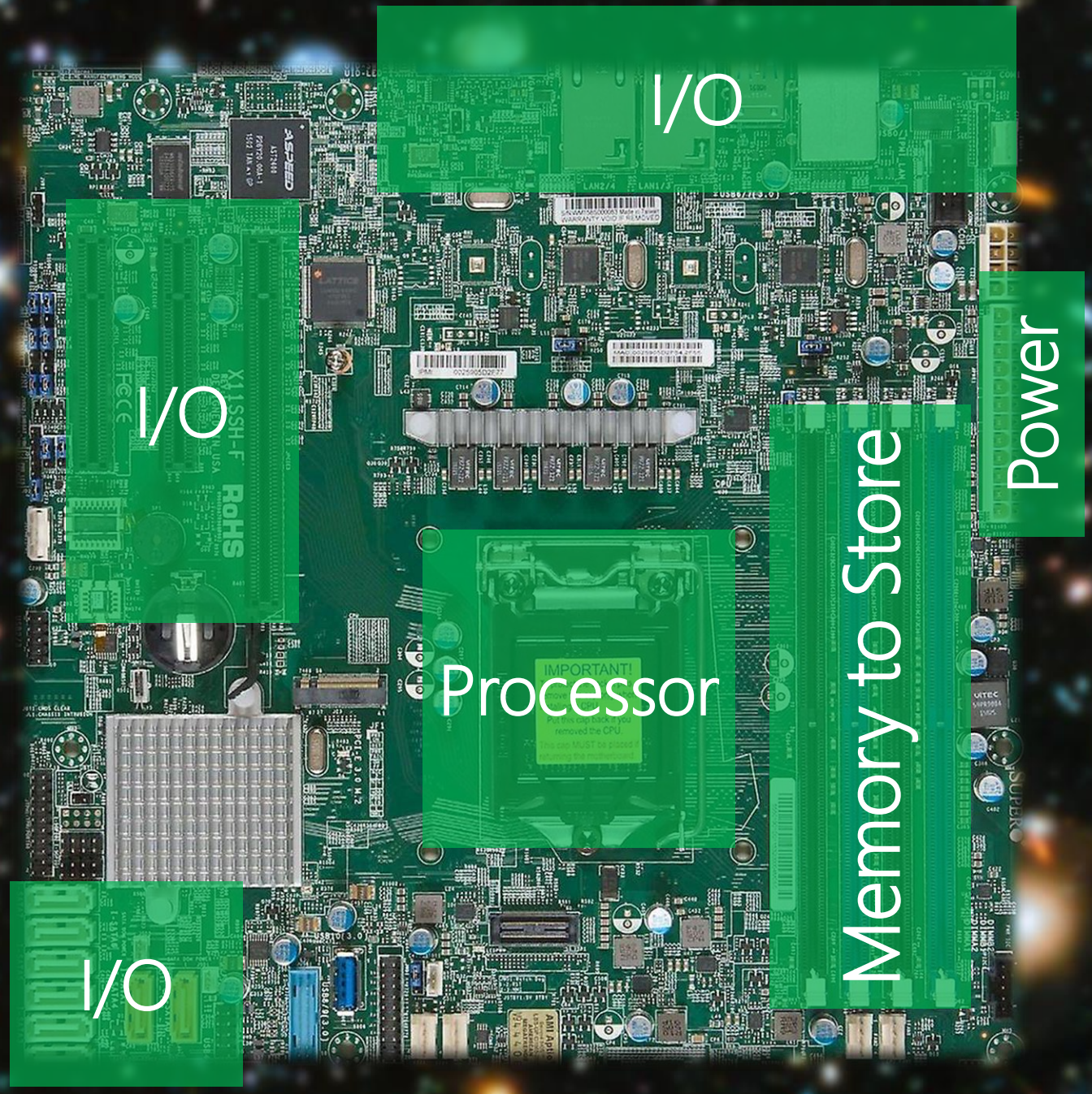
von NEUMANN ARCHITECTURE

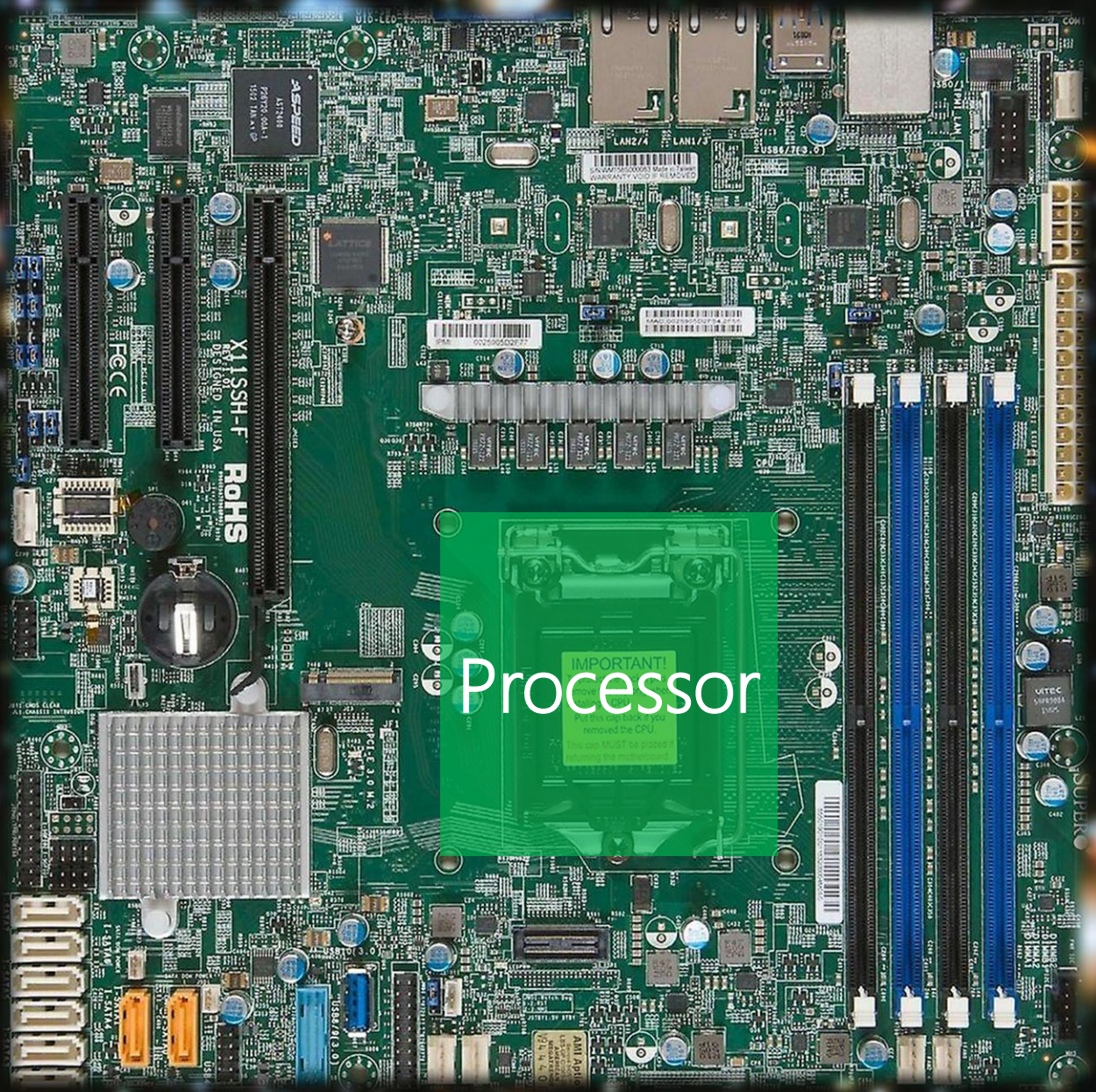
1. Data and instructions are all in the memory
2. The memory is addressable by location (regardless of what is stored in that location)
3. Instructions are executed sequentially unless the order is explicitly modified

Computer System



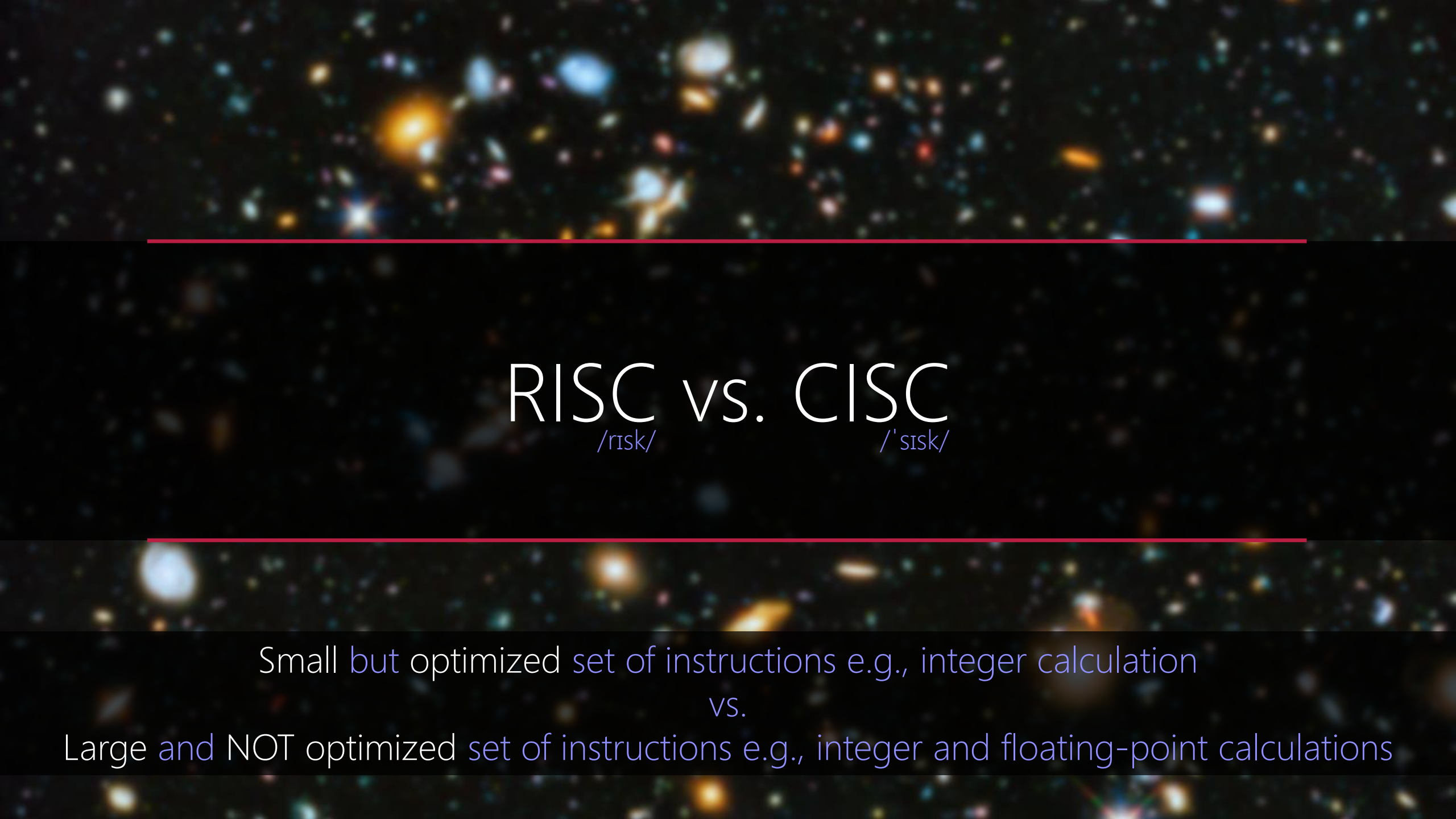

X11SSH-F
REV:1.01
DESIGNED IN USA
RoHS
IMPORTANT!
To protect the socket pins, remove this cap only when installing a CPU.
Put this cap back if you removed the CPU.
This cap MUST be placed if returning the motherboard.
LAN2/4
LAN1/3
USB6/7(3.0)
USB0/1(3.0)
USB4/5(3.0)
SUPER


I/O
I/O
Power
Memory to Store
Processor
I/O


Processor

PROCESSOR

1. Cannot instruct a processor to do whatever we want!
2. Processors have limitations.
   Some can only do addition,
   Some can do both addition and subtraction, but no division



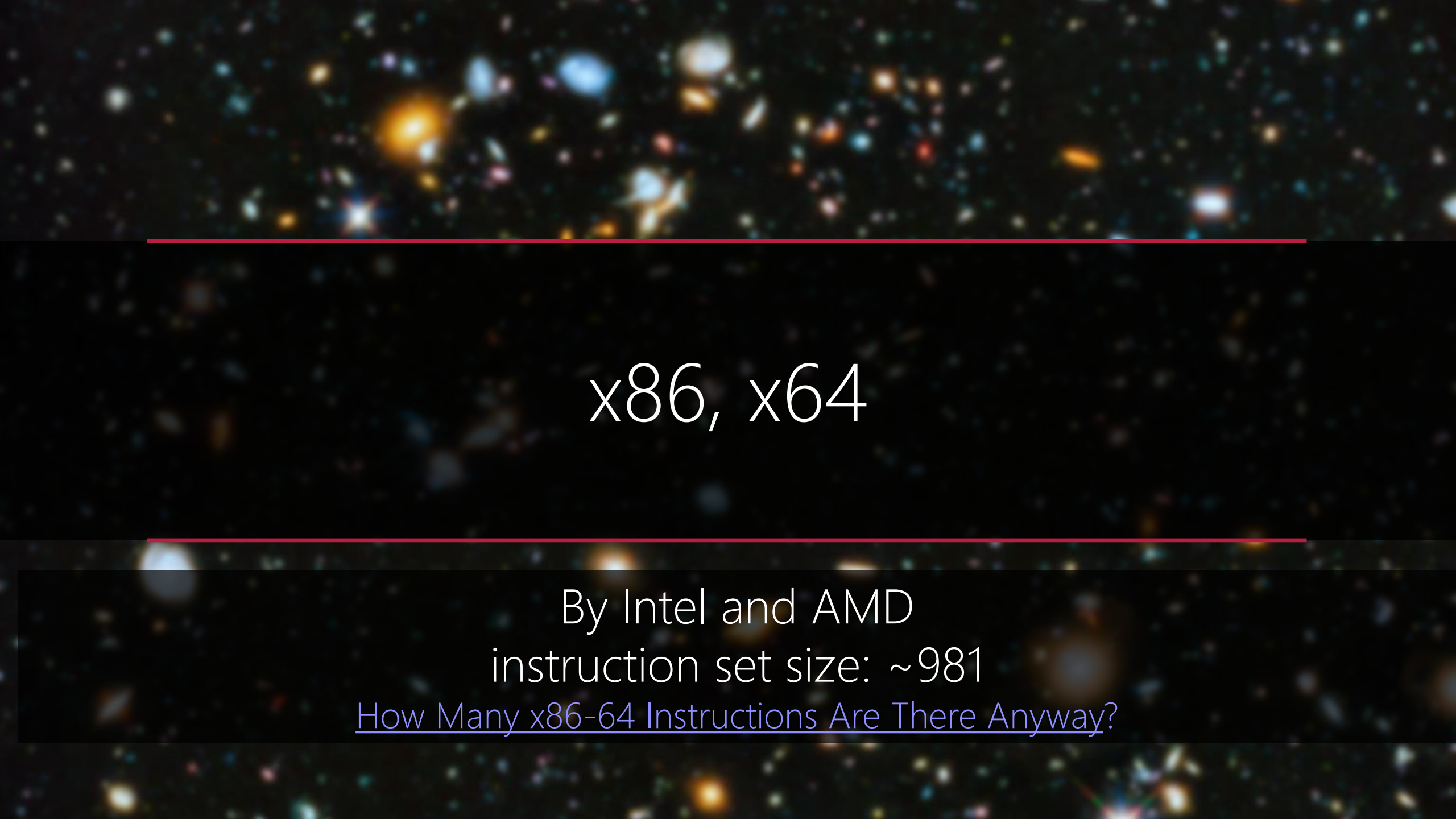
RISC vs. CISC

/rɪsk/ /ˈsɪsk/

Small but optimized set of instructions e.g., integer calculation

vs.

Large and NOT optimized set of instructions e.g., integer and floating-point calculations

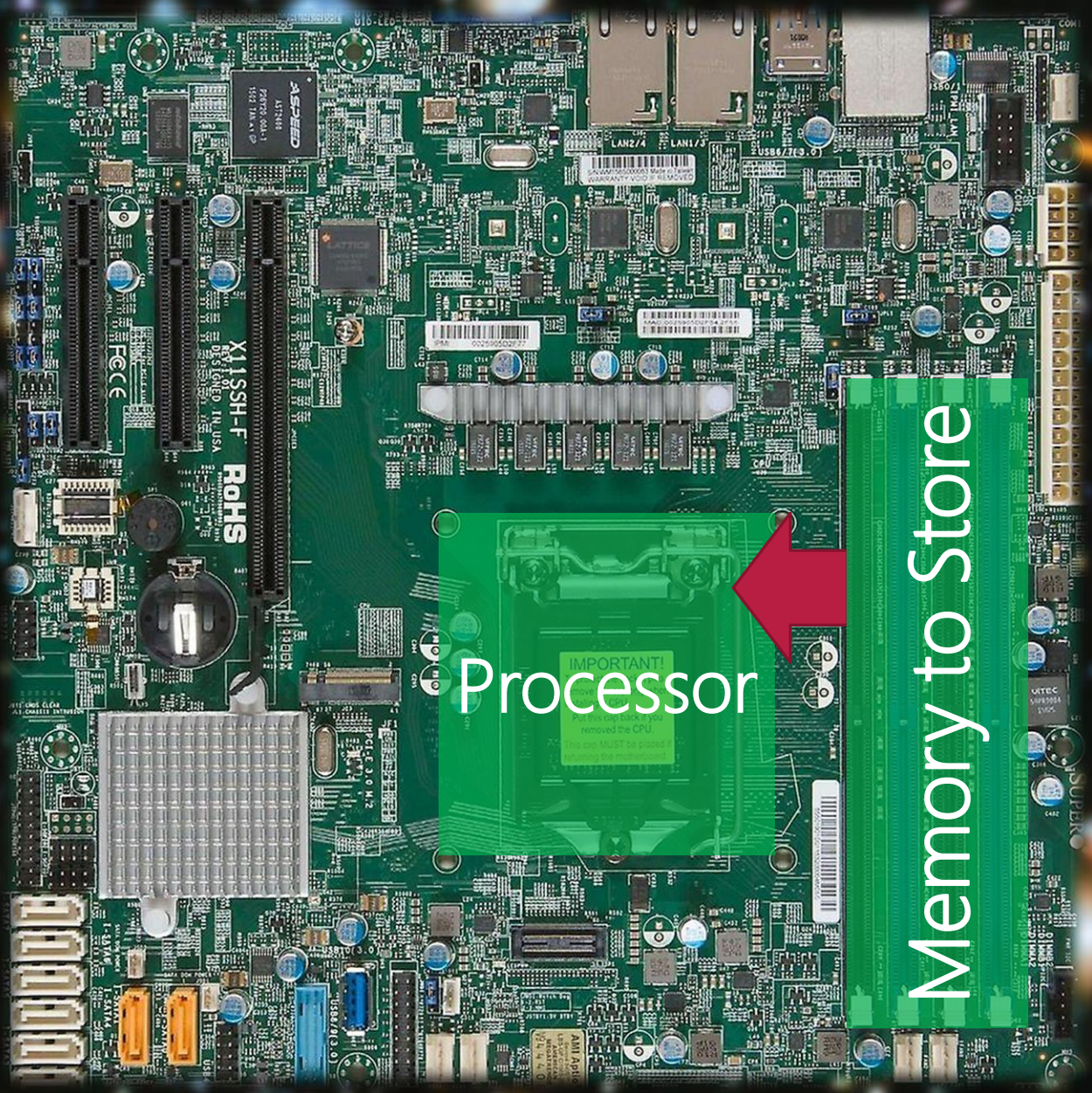


x86, x64

By Intel and AMD

instruction set size: ~981

How Many x86-64 Instructions Are There Anyway?


Processor
Memory to Store

Instruction Decoder

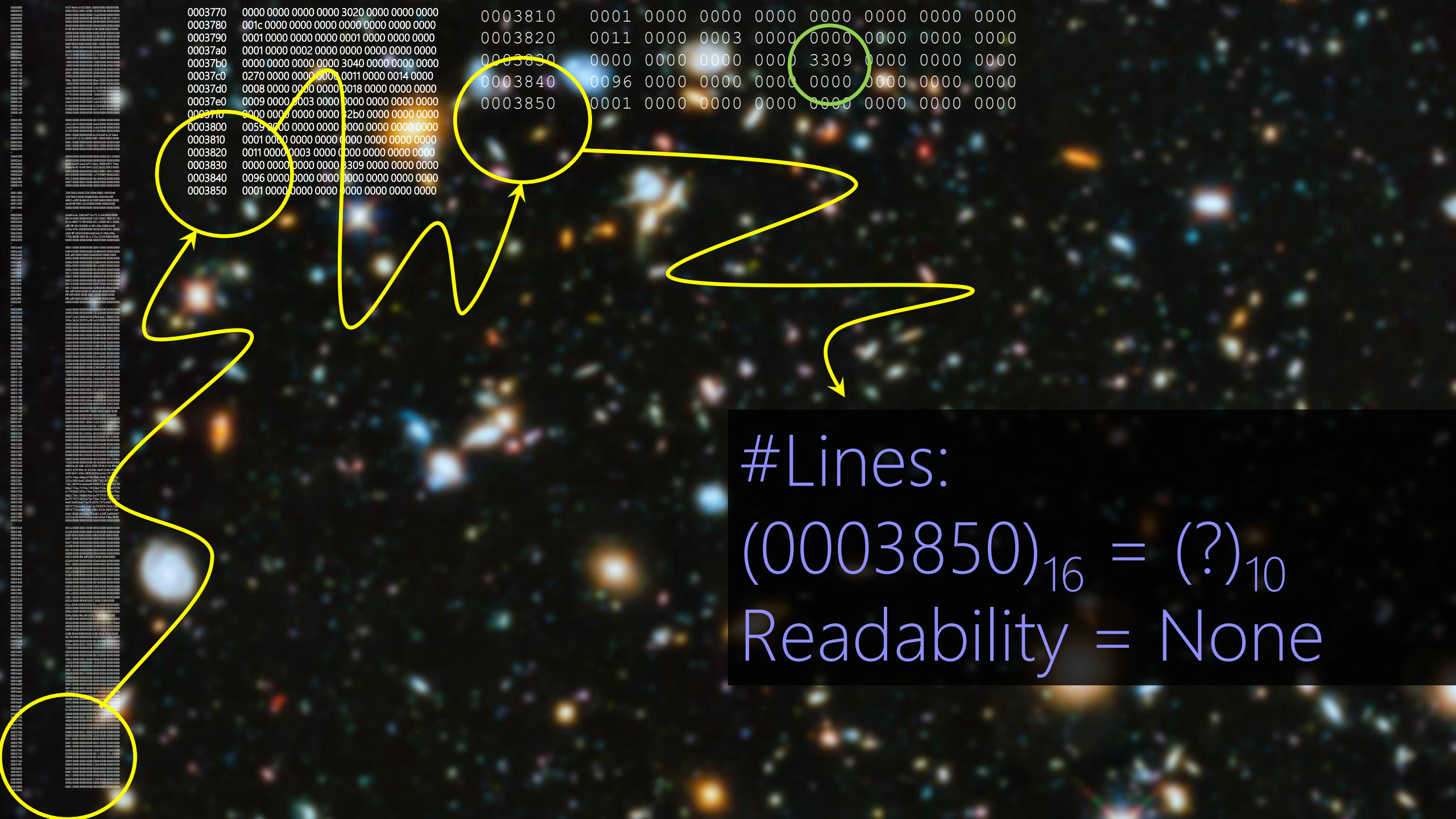
Operation Code (OP Code)	01001000 XXXX YYYY ZZZZ
What to do?	1) Fetch the first operand from memory at XXXX address 2) Store the first operand inside somewhere (AX) 3) Fetch the second operand from memory at YYYY address 4) Store the first operand inside somewhere else (BX) 5) Use the n-bit Adder to add AX and BX 6) Store the result inside somewhere else (CX) 7) Push CX to memory at ZZZZ address

extremely simplified version*!*

Hello world!


0003770 0000 0000 0000 0000 3020 0000 0000 0000
0003780 001c 0000 0000 0000 0000 0000 0000 0000
0003790 0001 0000 0000 0000 0001 0000 0000 0000
00037a0 0001 0000 0002 0000 0000 0000 0000 0000
00037b0 0000 0000 0000 0000 3040 0000 0000 0000
00037c0 0270 0000 0000 0000 0011 0000 0014 0000
00037d0 0008 0000 0000 0000 0018 0000 0000 0000
00037e0 0009 0000 0003 0000 0000 0000 0000 0000
00037f0 0000 0000 0000 0000 32b0 0000 0000 0000
0003800 0059 0000 0000 0000 0000 0000 0000 0000
0003810 0001 0000 0000 0000 0000 0000 0000 0000
0003820 0011 0000 0003 0000 0000 0000 0000 0000
0003830 0000 0000 0000 0000 3309 0000 0000 0000
0003840 0096 0000 0000 0000 0000 0000 0000 0000
0003850 0001 0000 0000 0000 0000 0000 0000 0000


0003770 0000 0000 0000 0000 3020 0000 0000 0000
0003780 001c 0000 0000 0000 0000 0000 0000 0000
0003790 0001 0000 0000 0000 0001 0000 0000 0000
00037a0 0001 0000 0002 0000 0000 0000 0000 0000
00037b0 0000 0000 0000 0000 3040 0000 0000 0000
00037c0 0270 0000 0000 0000 0011 0000 0014 0000
00037d0 0008 0000 0000 0000 0018 0000 0000 0000
00037e0 0009 0000 0003 0000 0000 0000 0000 0000
0003800 0059 0000 0000 0000 0000 0000 0000 0000
0003810 0001 0000 0000 0000 0000 0000 0000 0000
0003820 0011 0000 0003 0000 0000 0000 0000 0000
0003830 0000 0000 0000 0000 3309 0000 0000 0000
0003840 0096 0000 0000 0000 0000 0000 0000 0000
0003850 0001 0000 0000 0000 0000 0000 0000 0000
0003810 0001 0000 0000 0000 0000 0000 0000 0000
0003820 0011 0000 0003 0000 0000 0000 0000 0000
0003830 0000 0000 0000 0000 3309 0000 0000 0000
0003840 0096 0000 0000 0000 0000 0000 0000 0000
0003850 0001 0000 0000 0000 0000 0000 0000 0000


0003770 0000 0000 0000 0000 3020 0000 0000 0000
0003780 001c 0000 0000 0000 0000 0000 0000 0000
0003790 0001 0000 0000 0000 0001 0000 0000 0000
00037a0 0001 0000 0002 0000 0000 0000 0000 0000
00037b0 0000 0000 0000 0000 3040 0000 0000 0000
00037c0 0270 0000 0000 0000 0011 0000 0014 0000
00037d0 0008 0000 0000 0000 0018 0000 0000 0000
00037e0 0009 0000 0003 0000 0000 0000 0000 0000
0003800 0059 0000 0000 0000 0000 0000 0000 0000
0003810 0001 0000 0000 0000 0000 0000 0000 0000
0003820 0011 0000 0003 0000 0000 0000 0000 0000
0003830 0000 0000 0000 0000 3309 0000 0000 0000
0003840 0096 0000 0000 0000 0000 0000 0000 0000
0003850 0001 0000 0000 0000 0000 0000 0000 0000
#Lines:
$(0003850)_{16} = (?)_{10}$
Readability = None

ASSEMBLY

Instruction Decoder

Assembler

Assembly	`ADD [XXXX], [YYYY], [ZZZZ]`
Operation Code (OP Code)	`01001000 XXXX YYYY ZZZZ`
What to do?	1) Fetch the first operand from memory at XXXX address 2) Store the first operand inside somewhere (AX) 3) Fetch the second operand from memory at YYYY address 4) Store the first operand inside somewhere else (BX) 5) Use the n-bit Adder to add AX and BX 6) Store the result inside somewhere else (CX) 7) Push CX to memory at ZZZZ address

extremely simplified version*!*

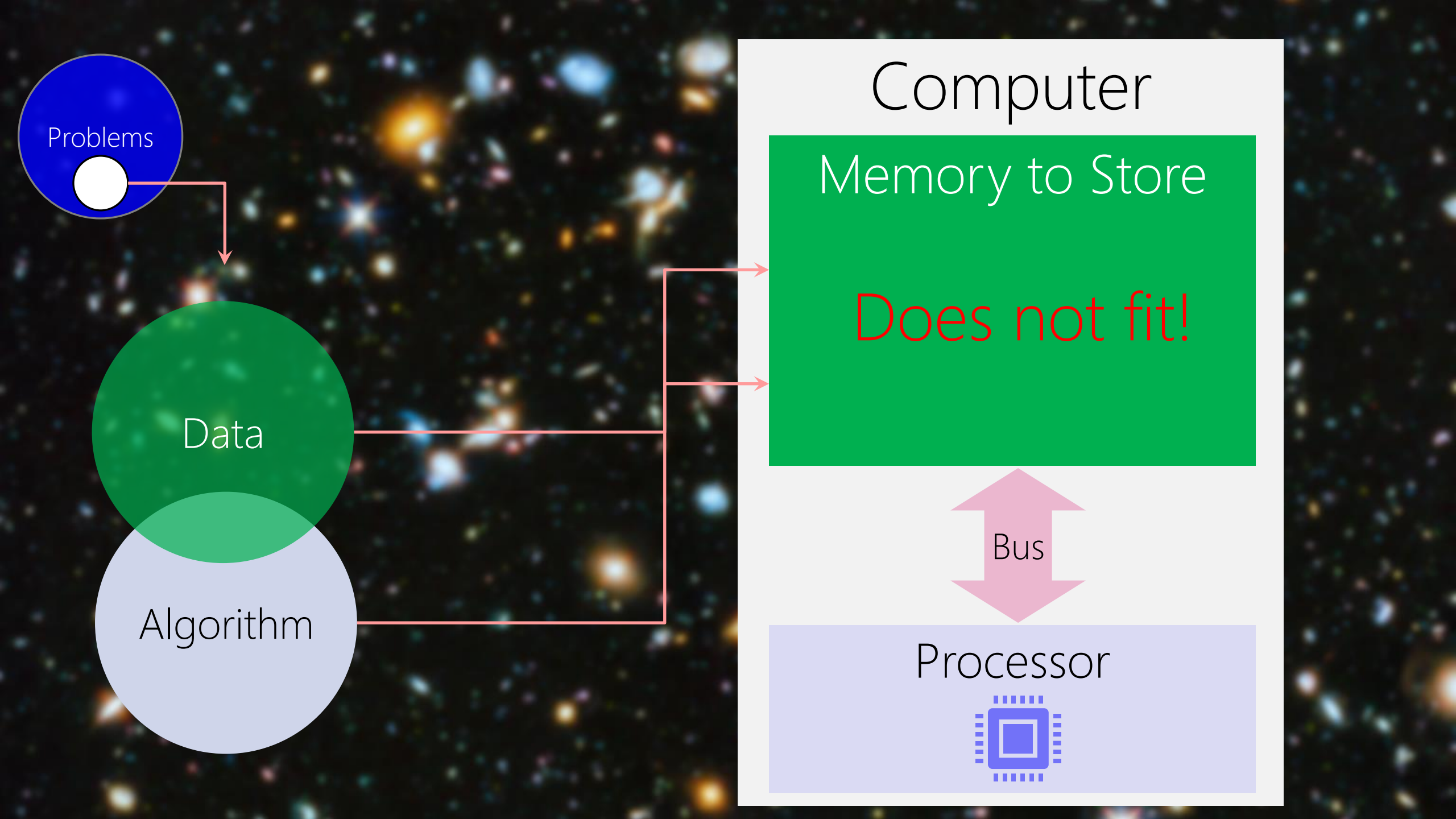
Hello world!

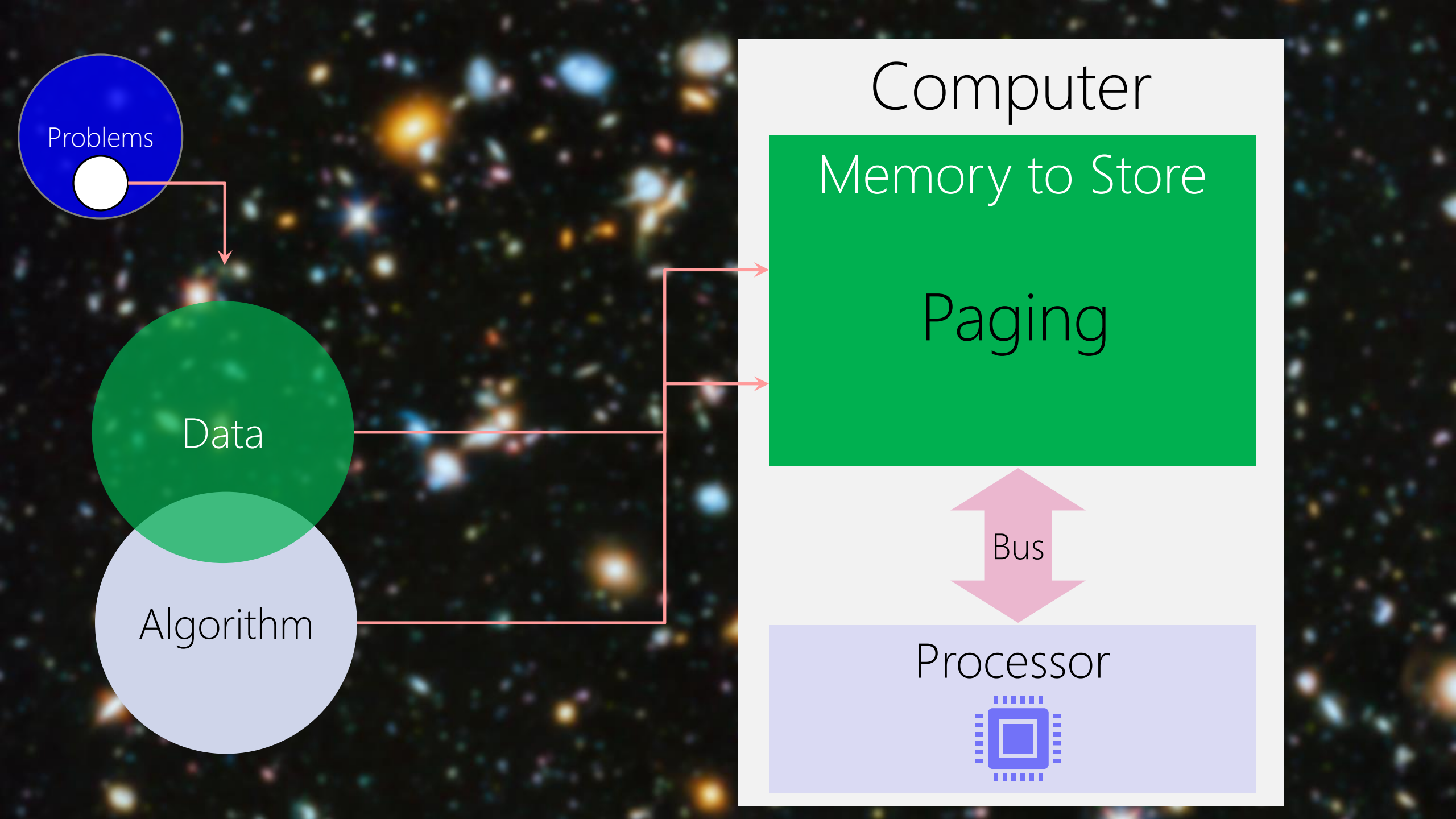
```
<.plt>:
pushq   0x3002(%rip)
jmpq    *0x3004(%rip)
nopl    0x0(%rax)
<printf@plt>:
jmpq    *0x3002(%rip)
pushq   $0x0
jmpq    401000 <.plt>
<main>:
push    %rbp
mov     %rsp,%rbp
lea     0xfd5(%rip),%rdi
mov     $0x0,%eax
callq   401010 <printf@plt>
nop
pop     %rbp
retq
```

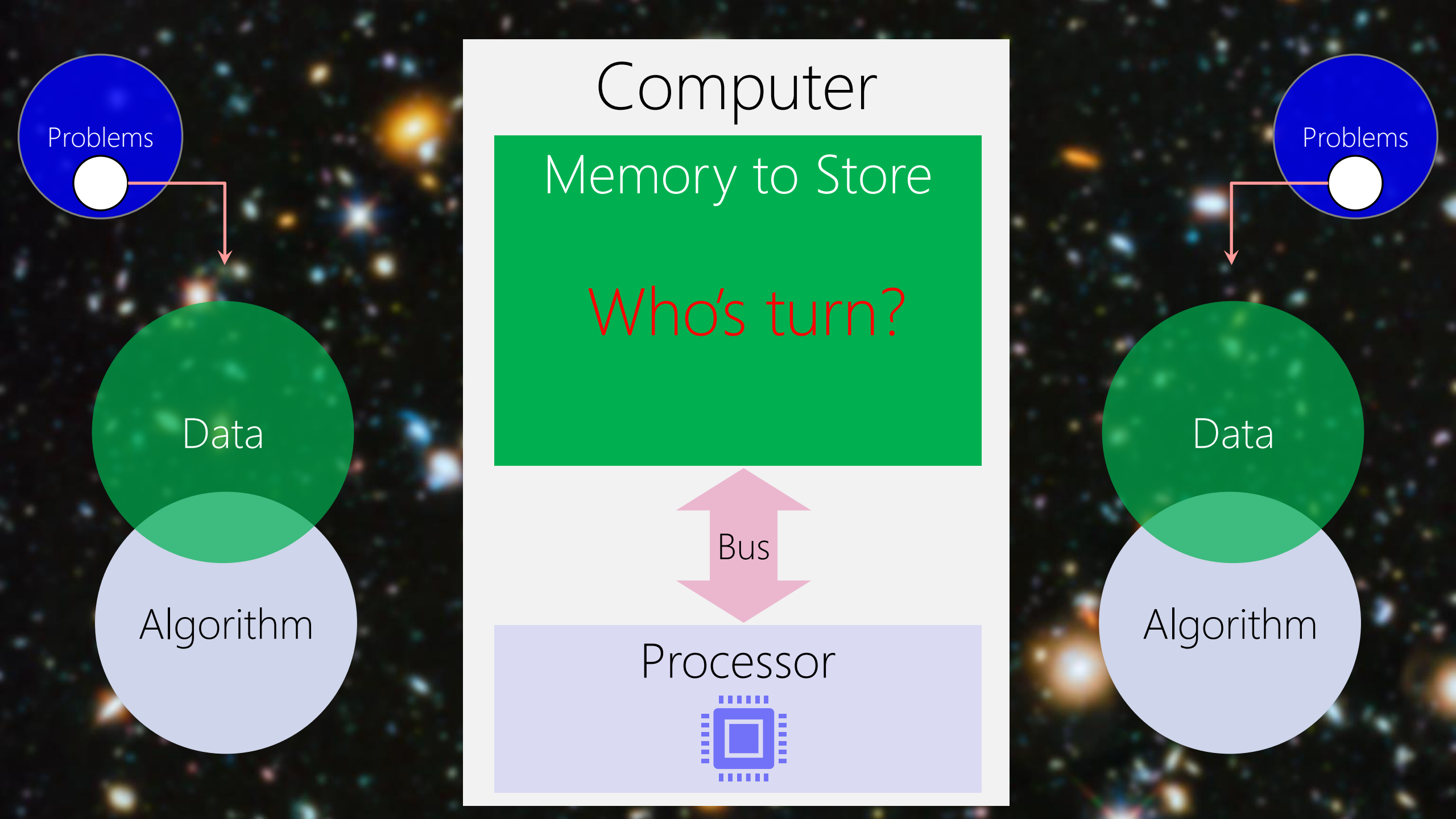
#Lines: 14-17 vs. 14416 opcodes

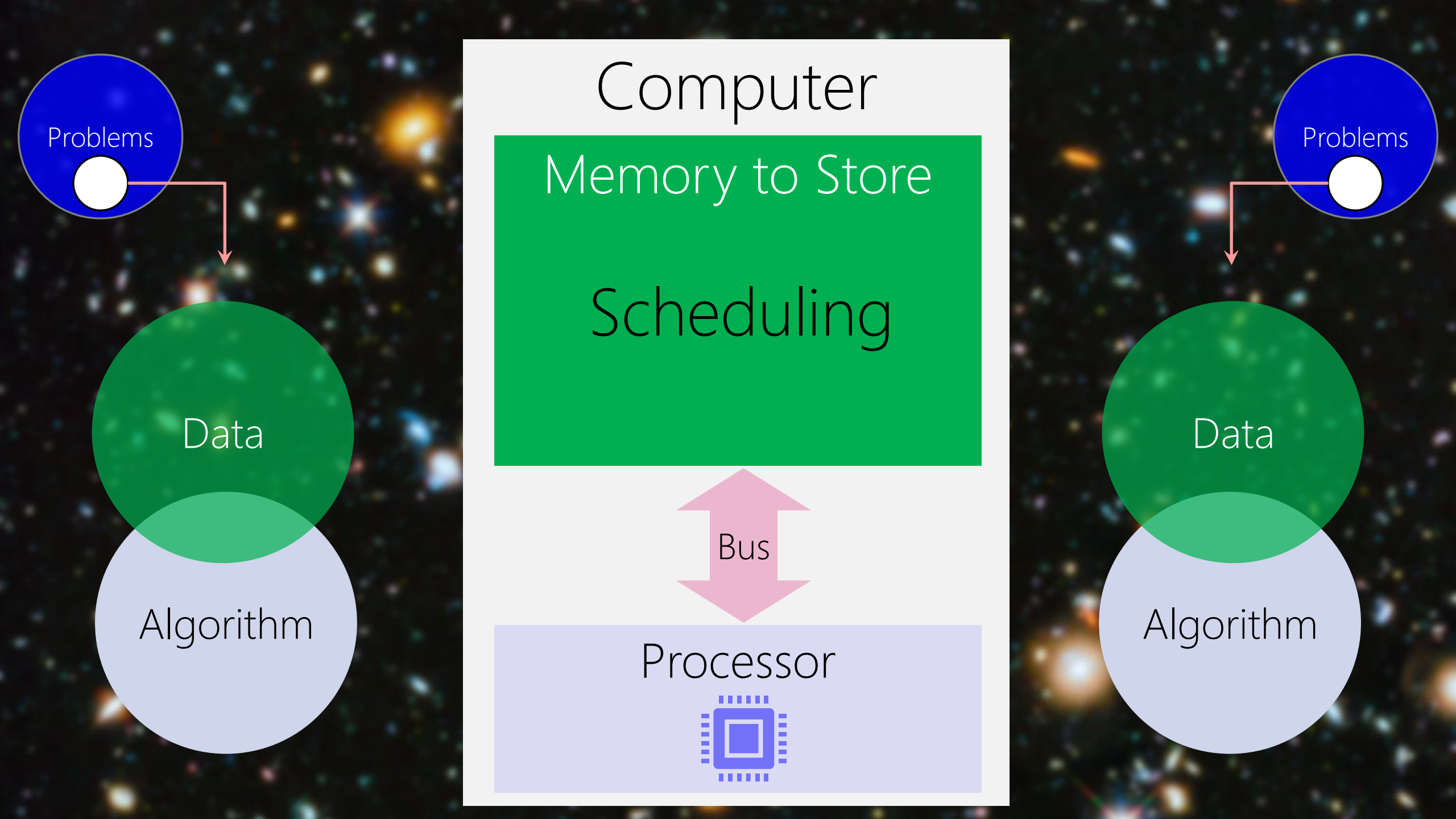
Readability: Fair

Dennis MacAlistair Ritchie
Kenneth Lane Thompson
AT&T Bell Lab, 1972, GE 645


Problems
Data
Algorithm
Computer
Memory to Store
Does not fit!
Bus
Processor

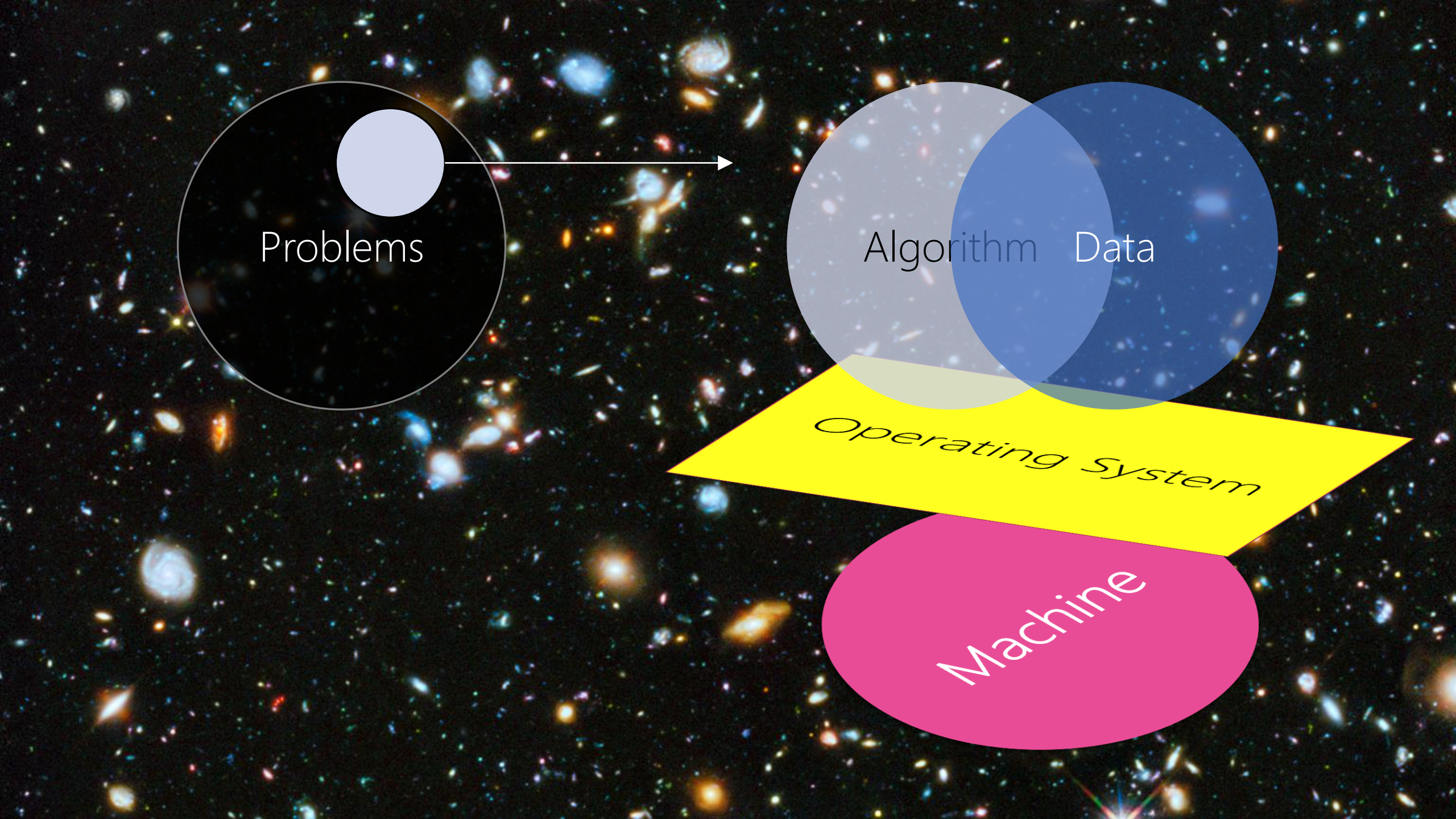

Problems
Data
Algorithm
Computer
Memory to Store
Paging
Bus
Processor

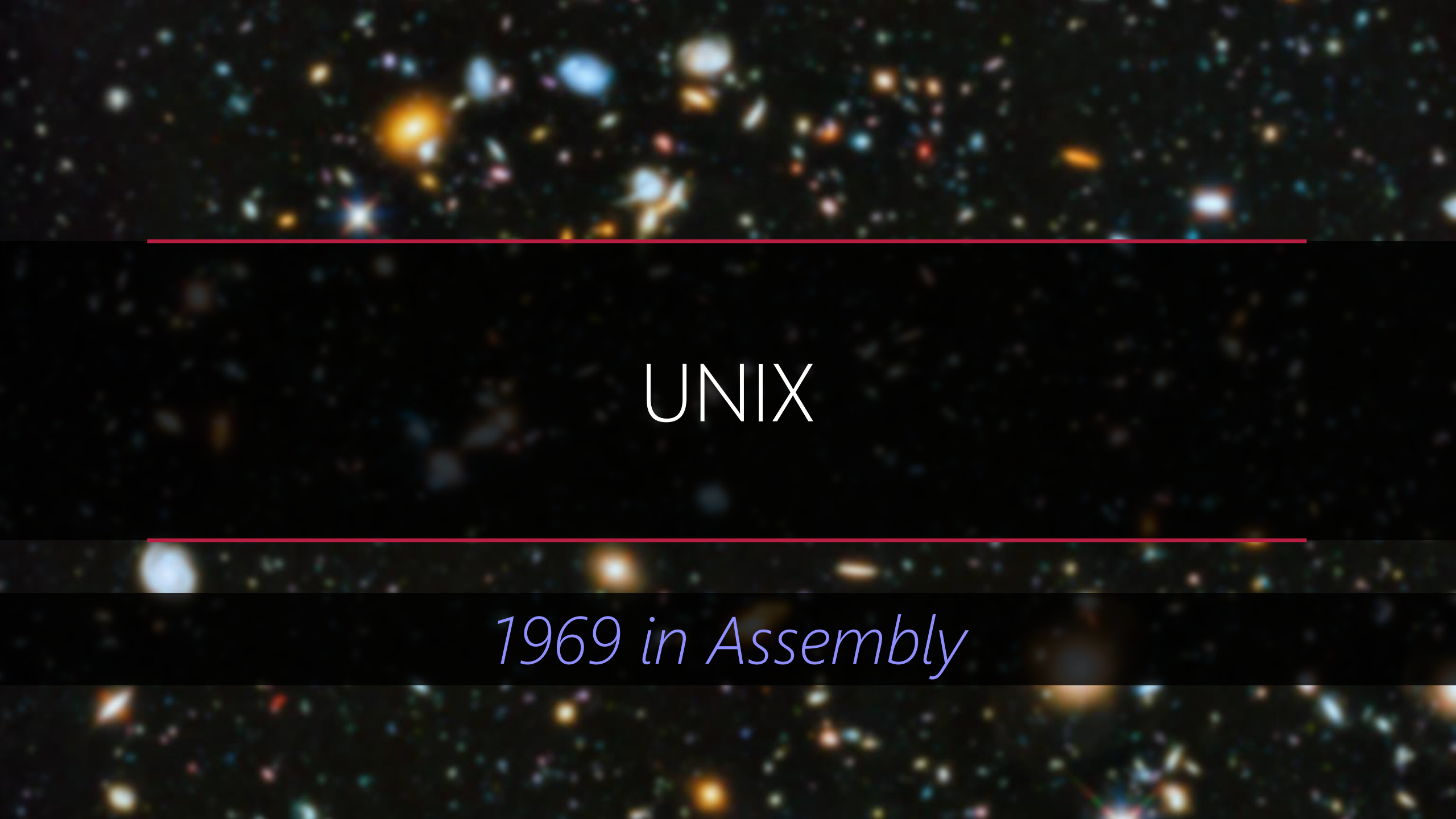

Problems
Data
Algorithm
Computer
Memory to Store
Who's turn?
Bus
Processor
Problems
Data
Algorithm


Problems
Data
Algorithm
Computer
Memory to Store
Scheduling
Bus
Processor
Problems
Data
Algorithm

Operating System

A program for programs!
System-level Program


Problems
Algorithm
Data
operating system
Machine



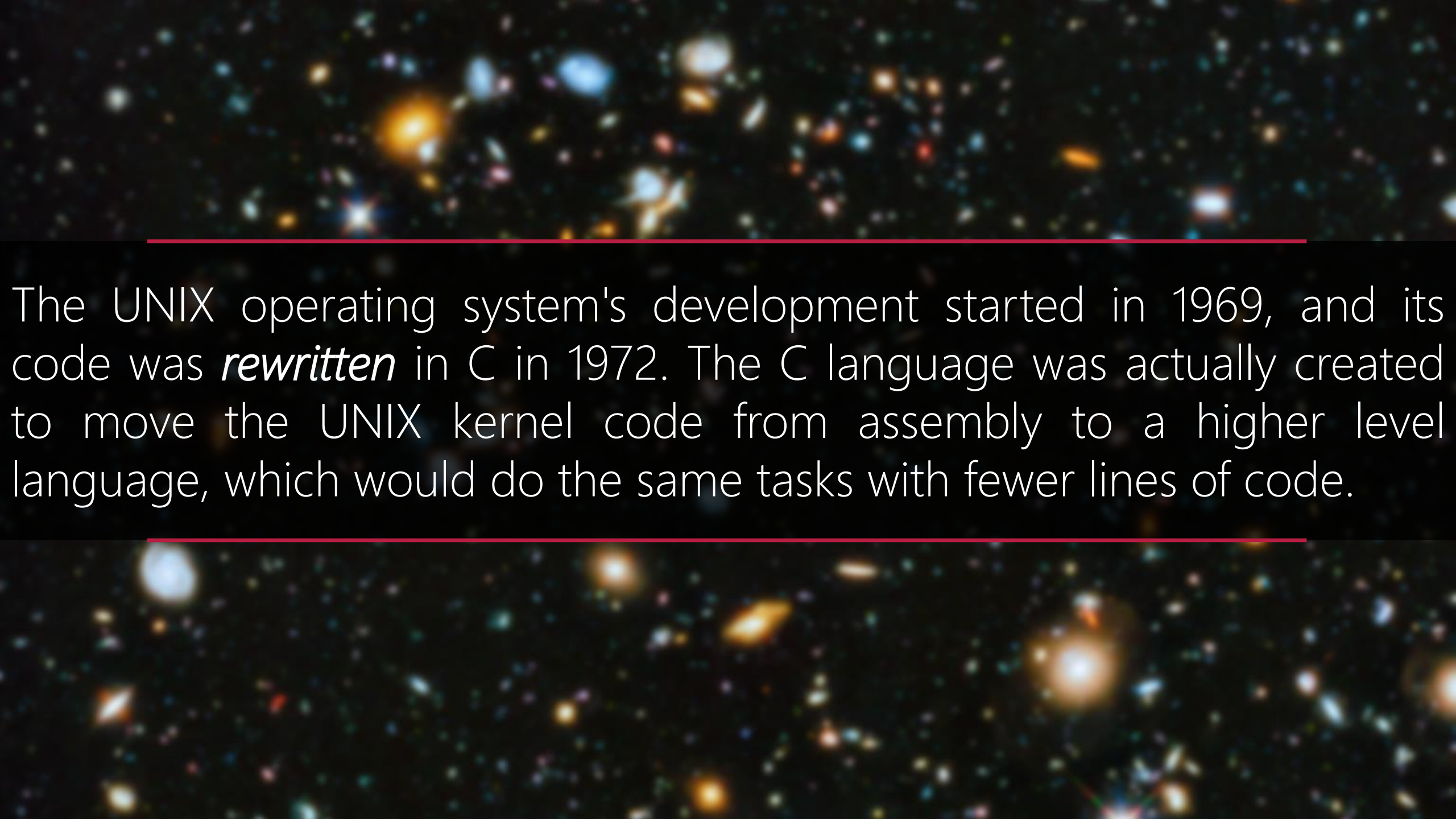
UNIX

1969 in Assembly


New Language

C for UNIX

System-level Programing



The UNIX operating system's development started in 1969, and its code was ***rewritten*** in C in 1972. The C language was actually created to move the UNIX kernel code from assembly to a higher level language, which would do the same tasks with fewer lines of code.

Instruction Decoder

Compiler

Assembler

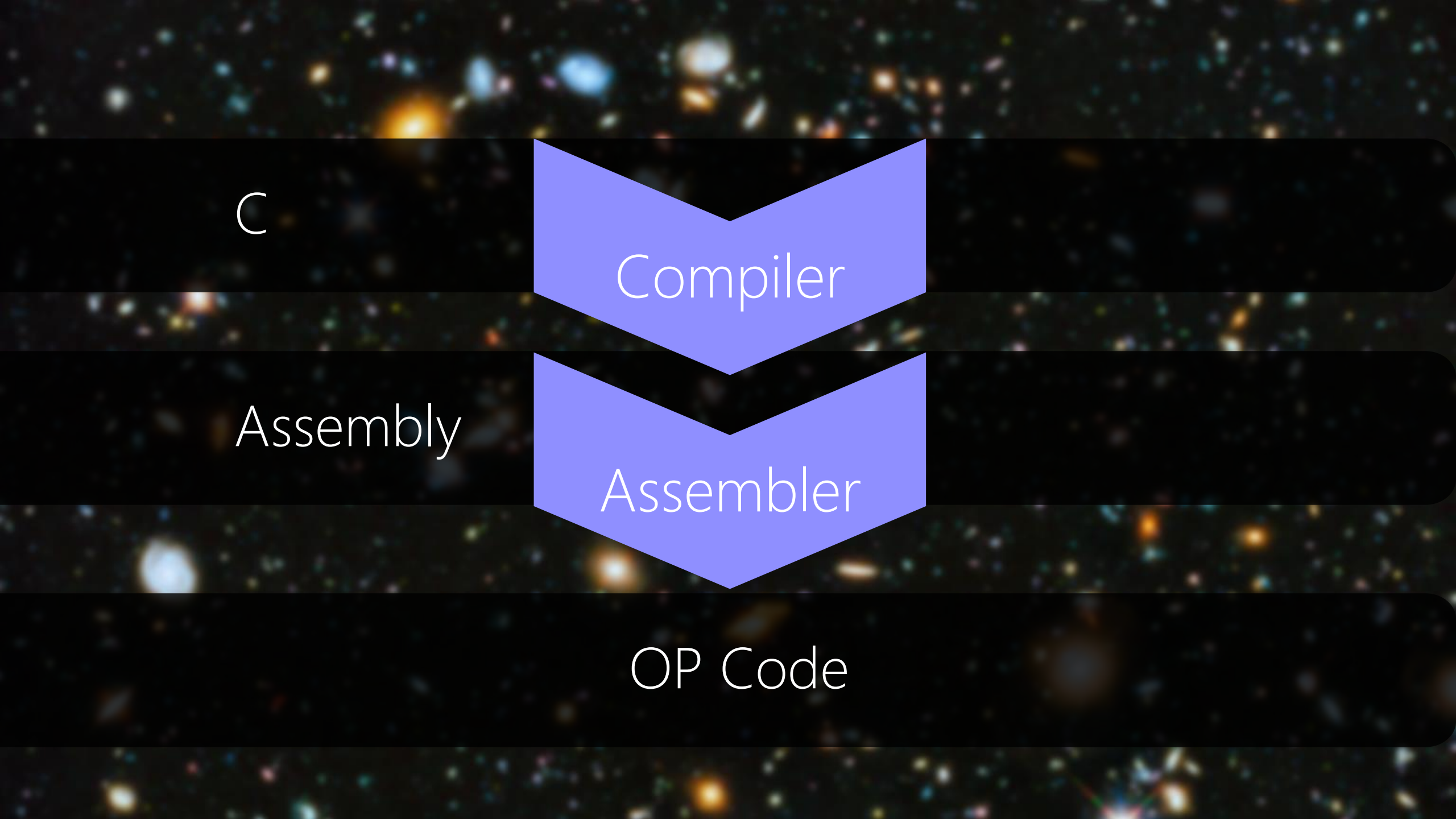
C	`C = a + b`
Assembly	`ADD [XXXX], [YYYY], [ZZZZ]`
Operation Code (OP Code)	`01001000 XXXX YYYY ZZZZ`
What to do?	1) Fetch the first operand from memory at XXXX address 2) Store the first operand inside somewhere (AX) 3) Fetch the second operand from memory at YYYY address 4) Store the first operand inside somewhere else (BX) 5) Use the n-bit Adder to add AX and BX 6) Store the result inside somewhere else (CX) 7) Push CX to memory at ZZZZ address

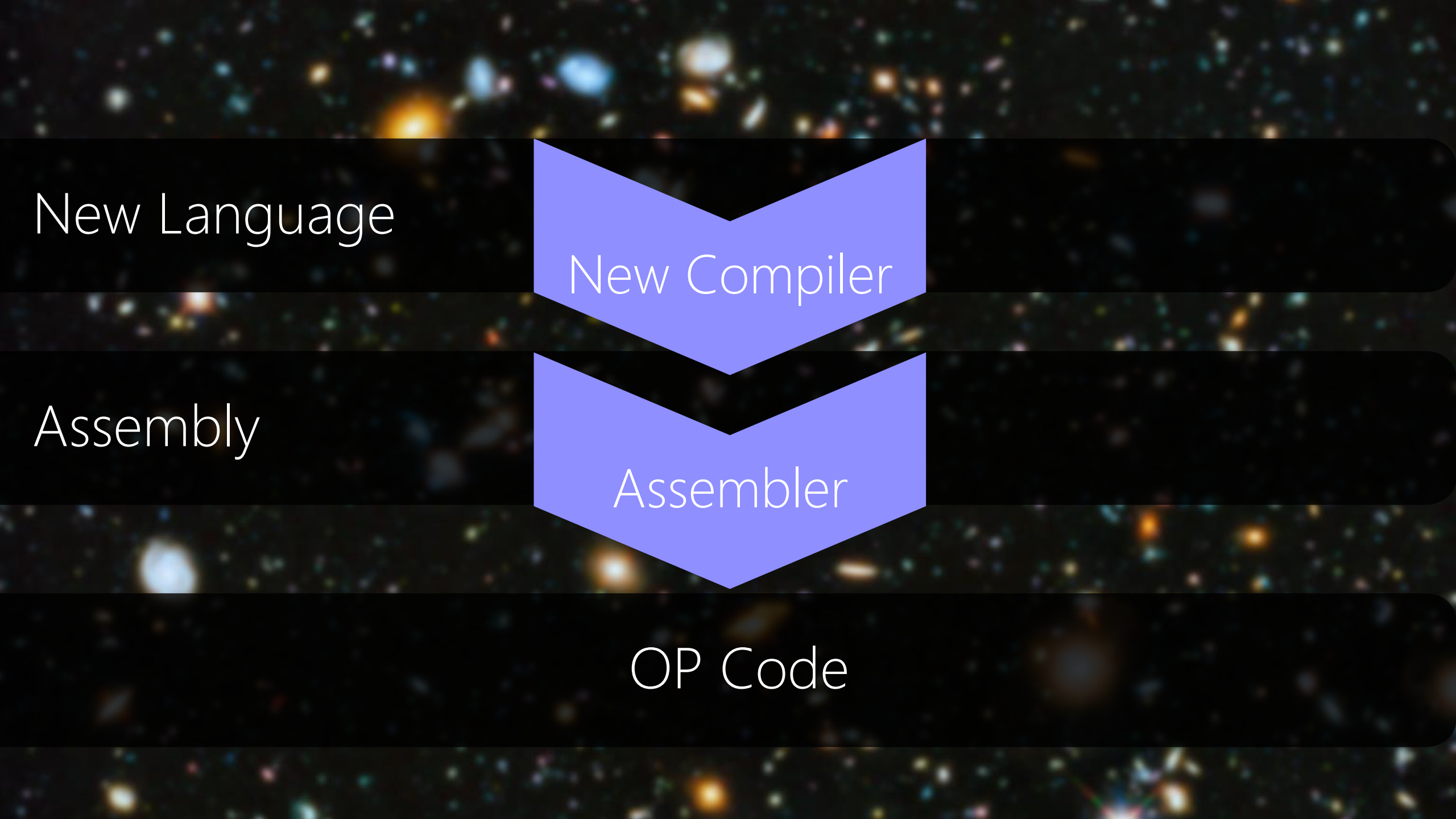
extremely simplified version*!*

```
#include <stdio.h>
void main(){
     printf("Hello world!");
}
```

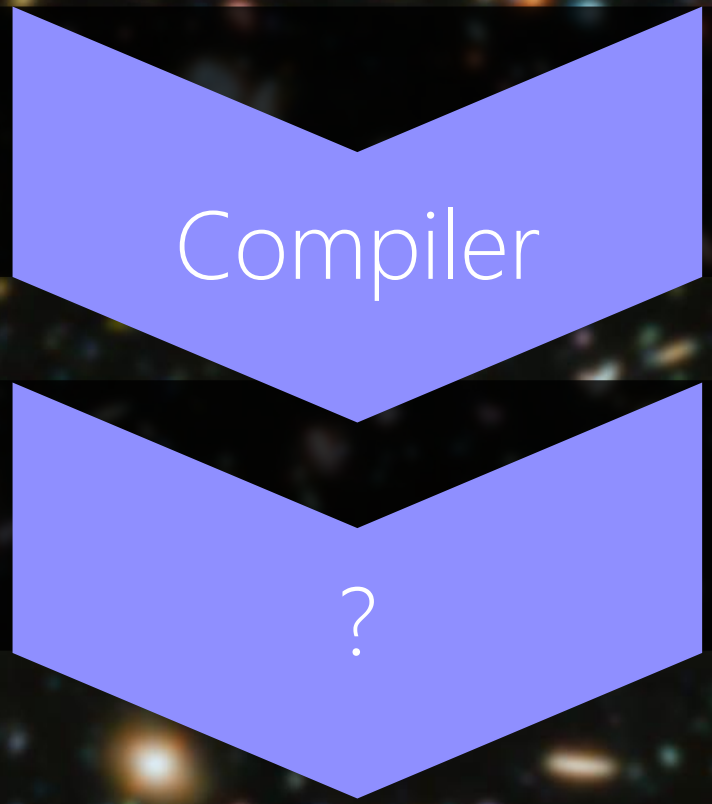
#Lines: 4 vs. 14 Assembly

Readability: Good!


C
Compiler
Assembly
Assembler
OP Code


New Language
New Compiler
Assembly
Assembler
OP Code

C



Intel → Apple

?

C

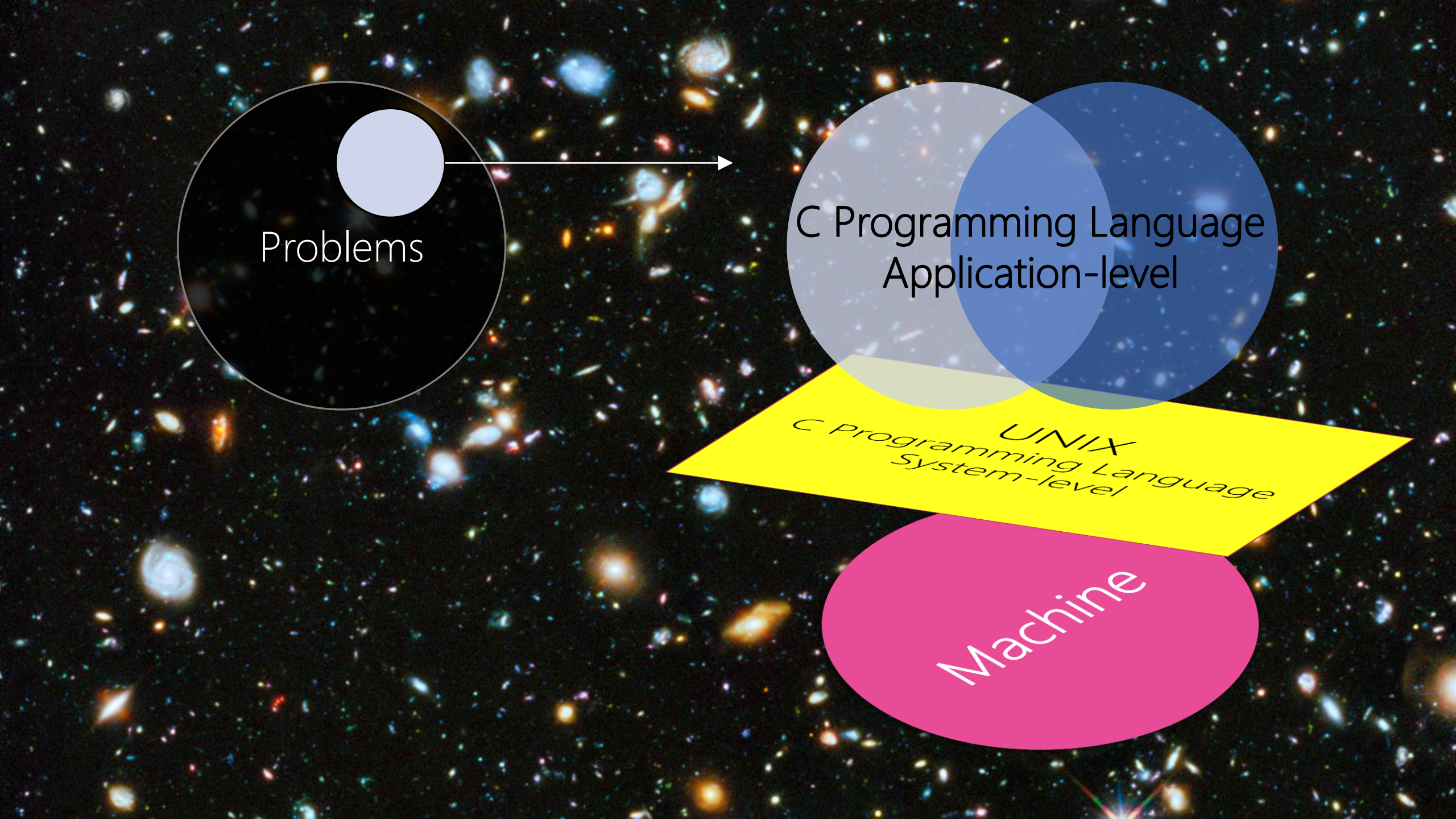
Compiler

Intel → Apple

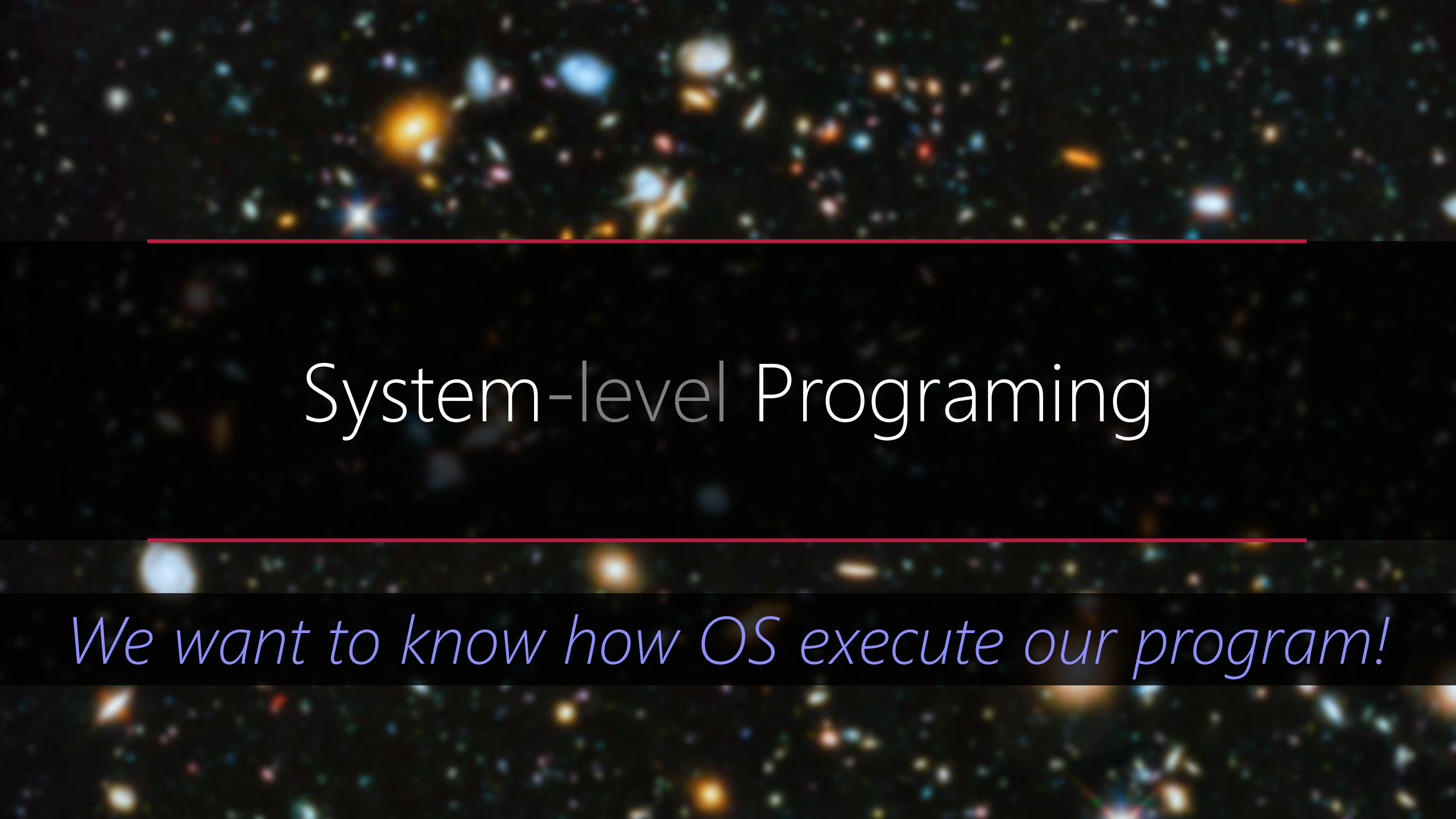
New Assembler

OP Code for Apple


C for ALL
Application-level Programing

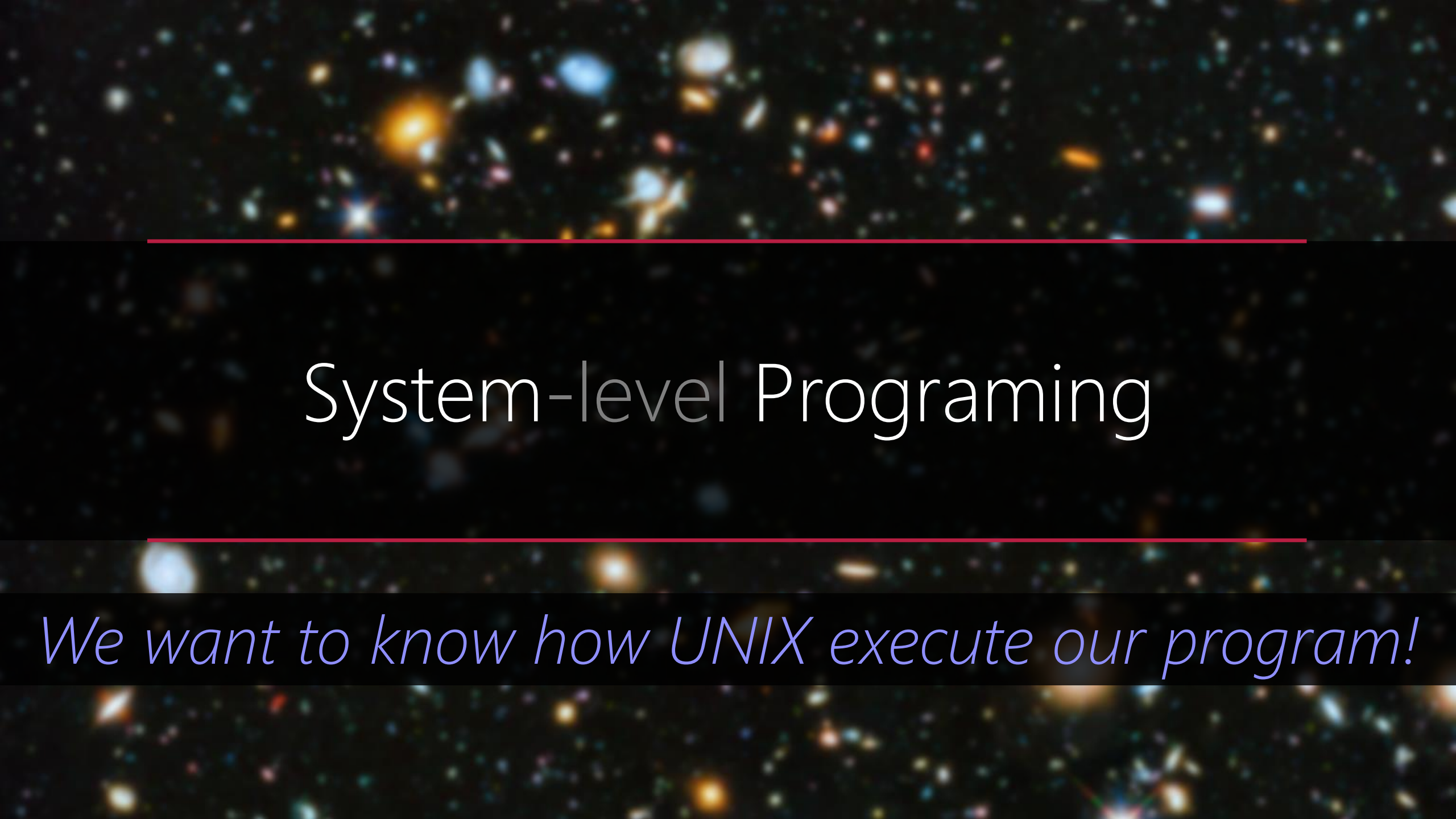

Problems
C Programming Language
Application-level
UNIX
C Programming Language
System-level
Machine

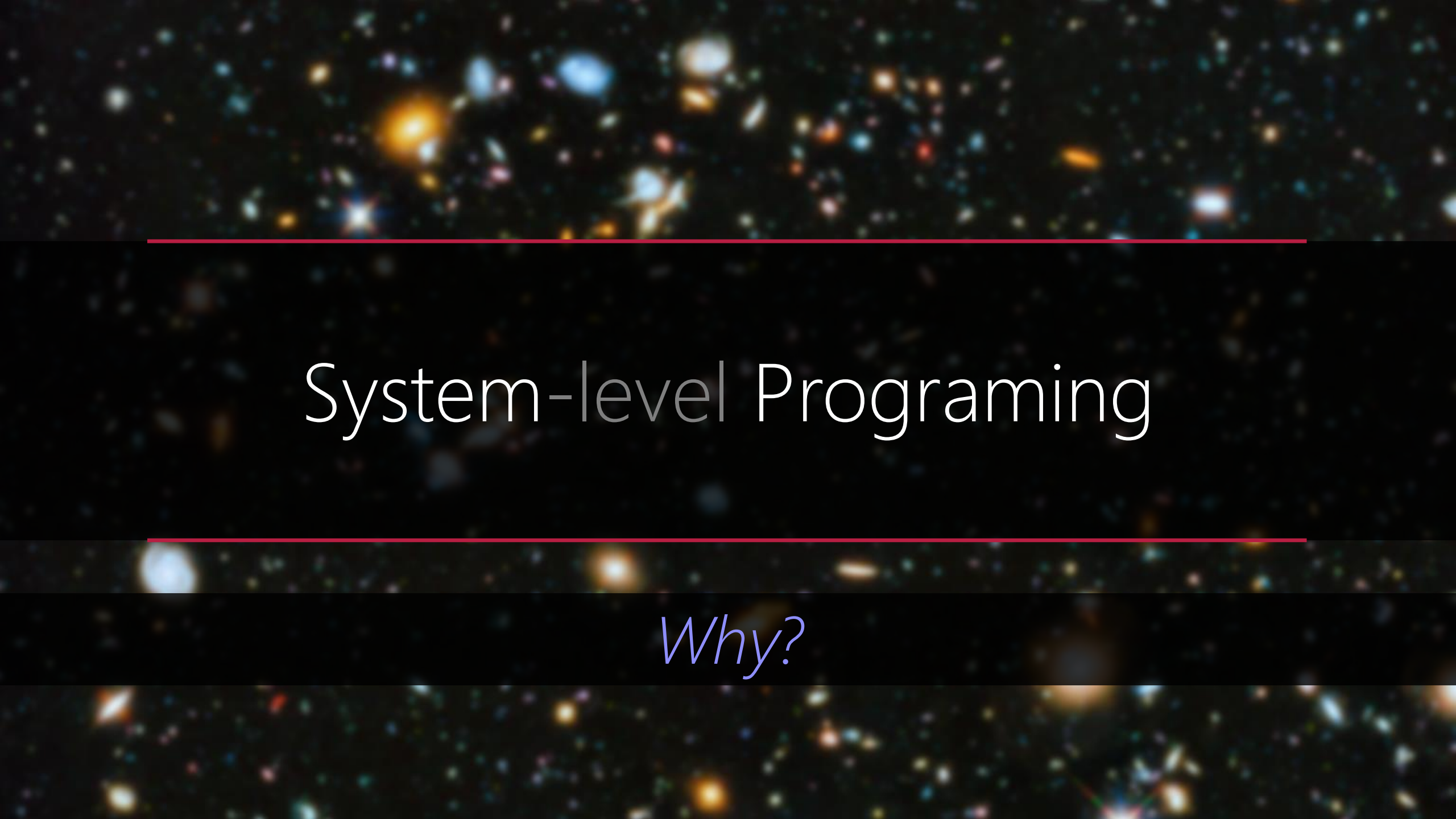

System-level Programing
We are not system-level programmer using C
We are still application-level programmer using C



System-level Programing

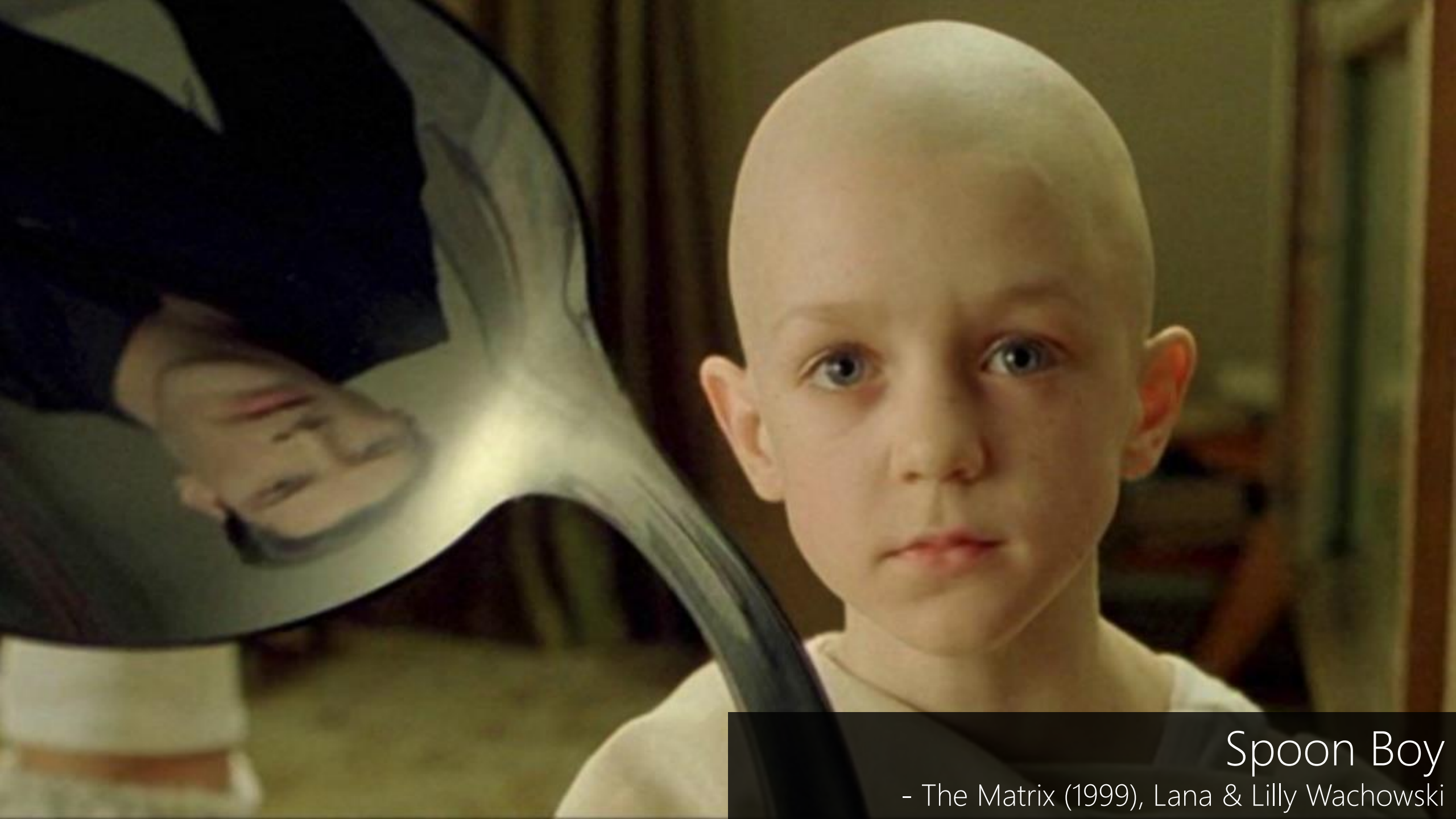
We want to know how OS execute our program!


System-level Programing
We want to know how UNIX execute our program!



System-level Programing

Why?



Spoon Boy

- The Matrix (1999), Lana & Lilly Wachowski