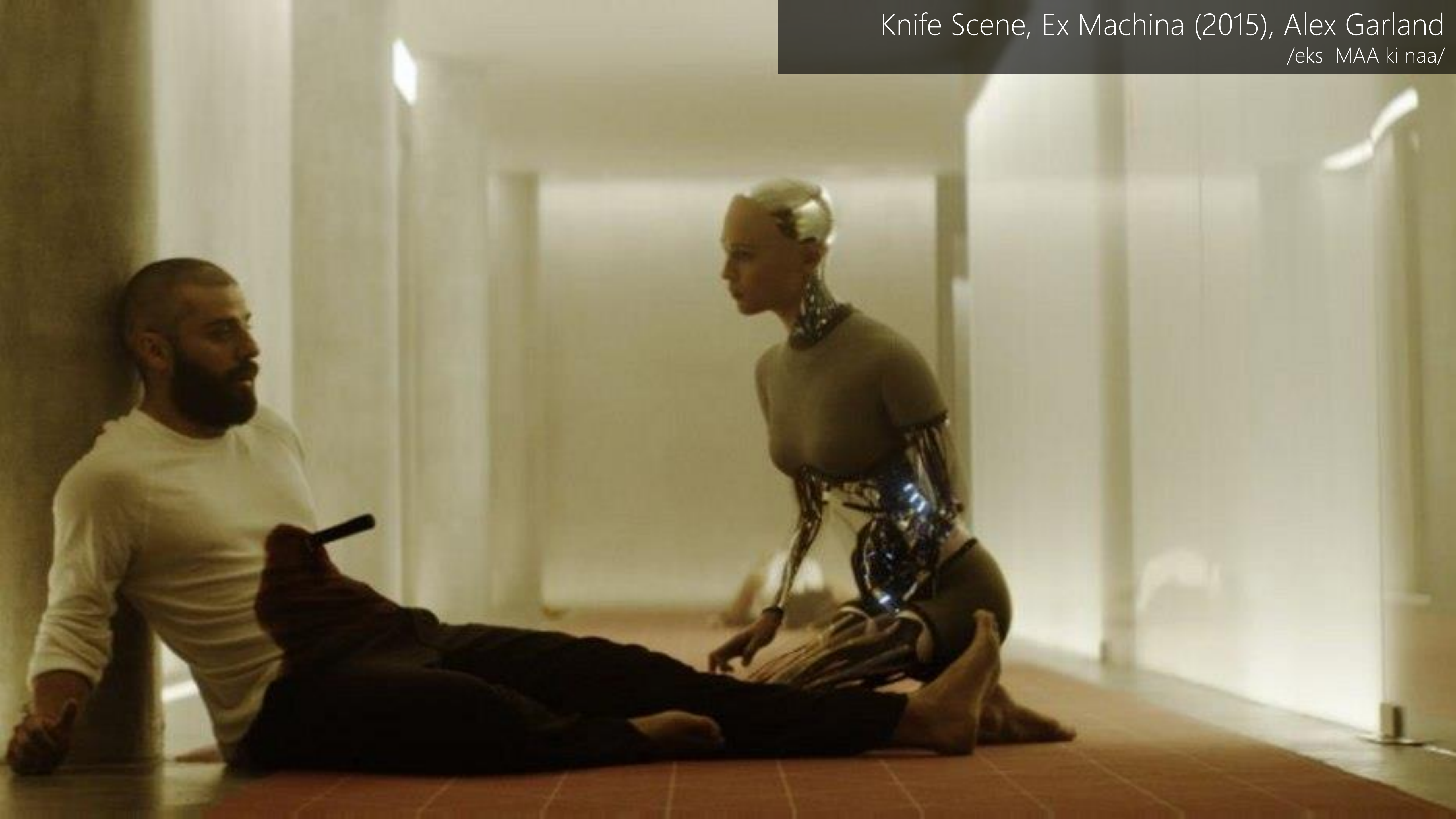
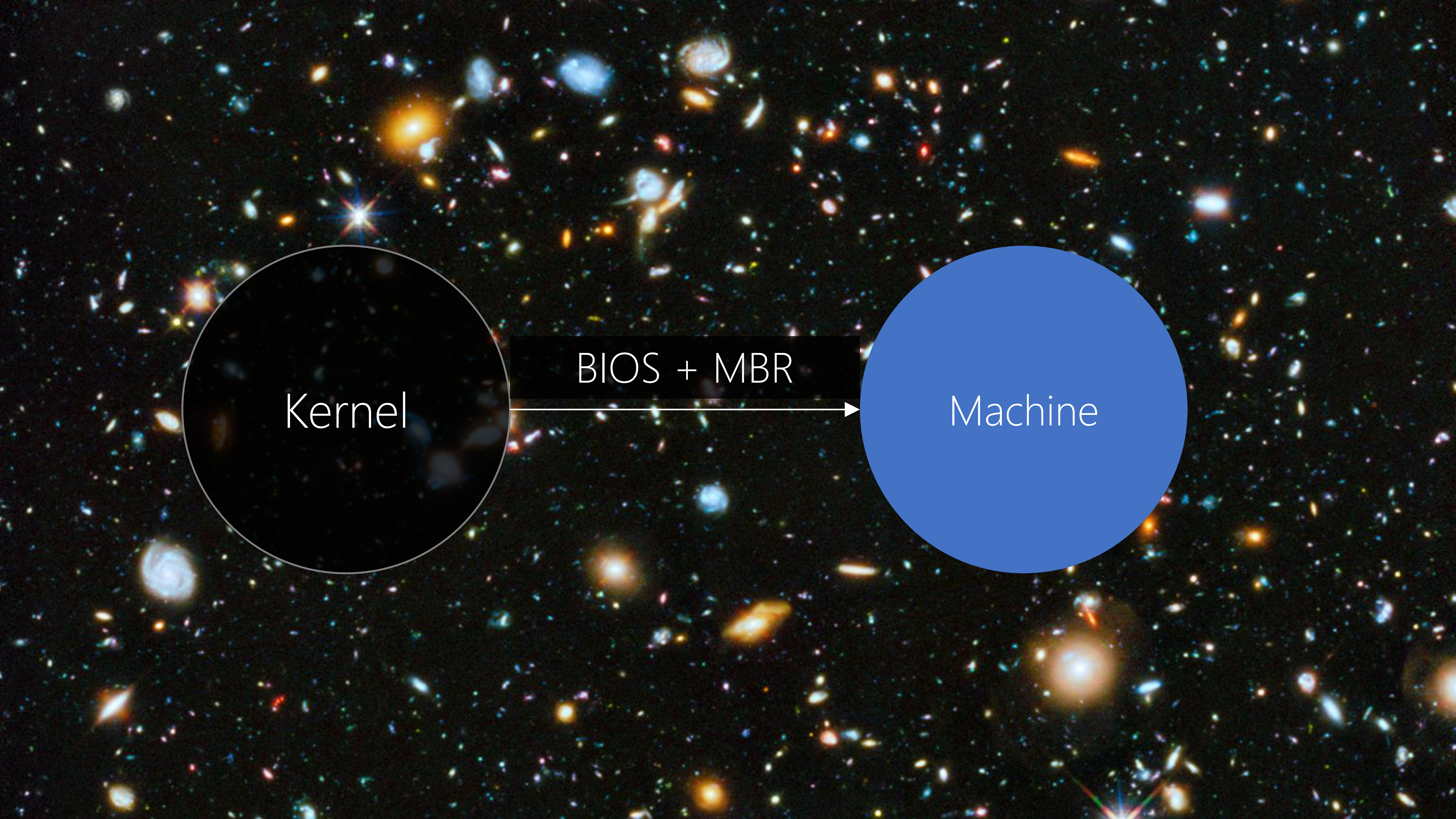
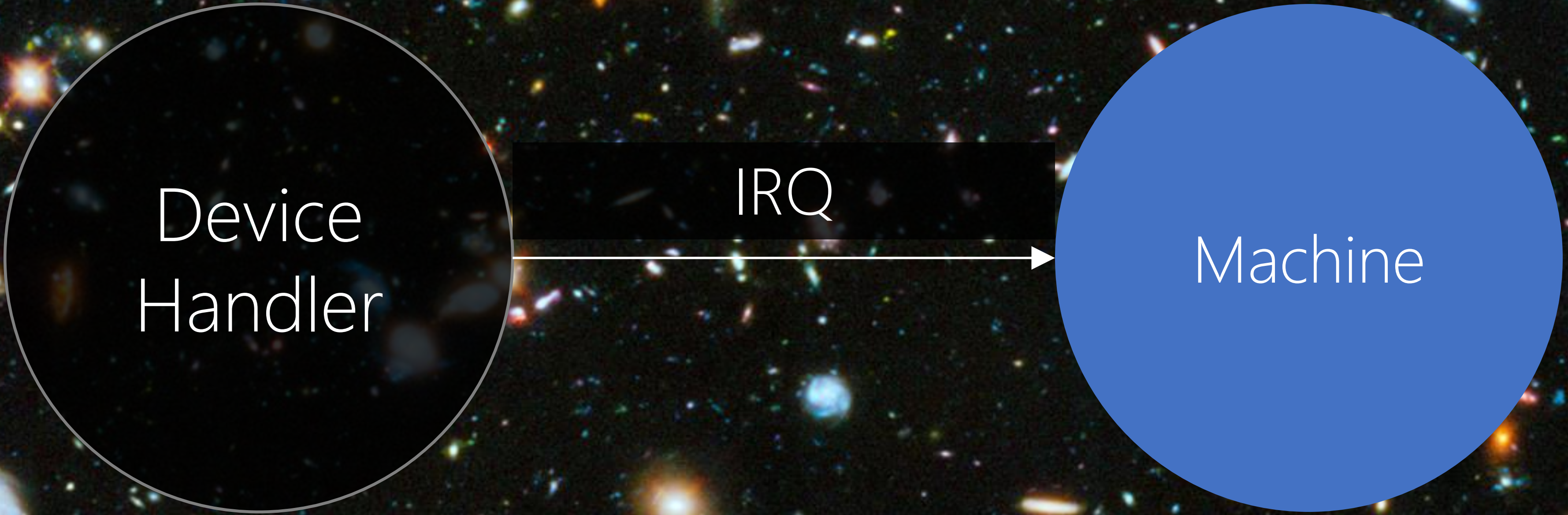
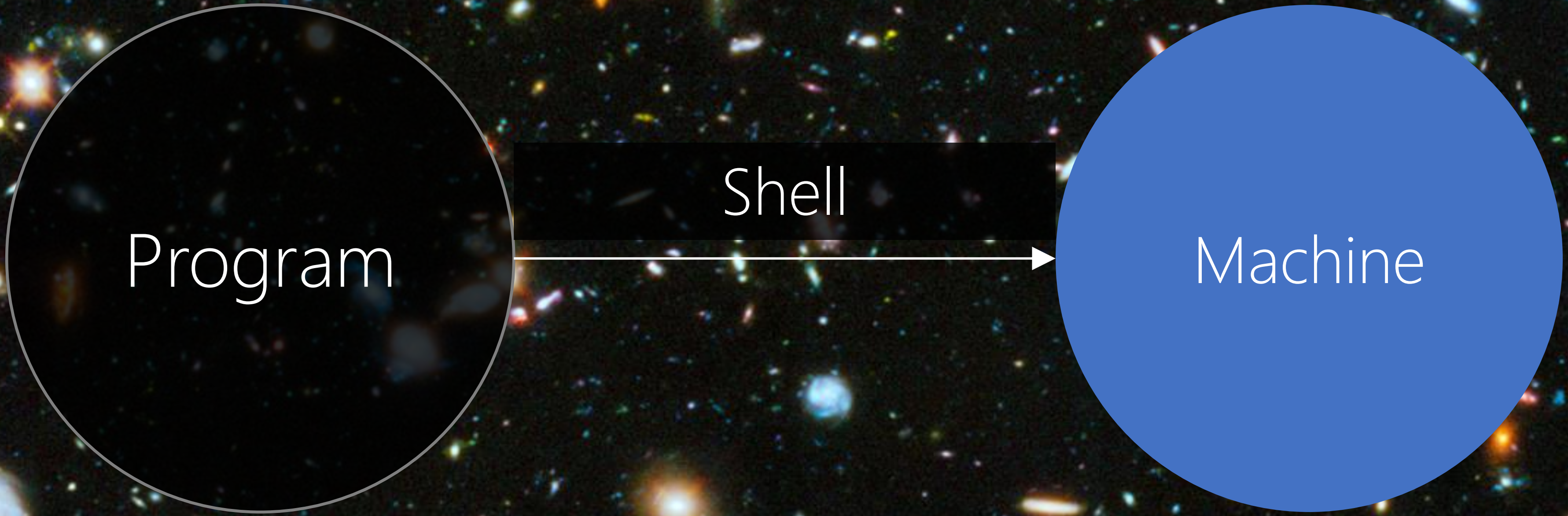


Knife Scene, Ex Machina (2015), Alex Garland

/eks MAA ki naa/


Kernel
BIOS + MBR
Machine


Device Handler
IRQ
Machine


Program
Shell
Machine

Ezekiel 25:17, Pulp Fiction (1994), Quentin Tarantino

Well, there's this passage I've got memorized; sorta fits this occasion.

Program → Process

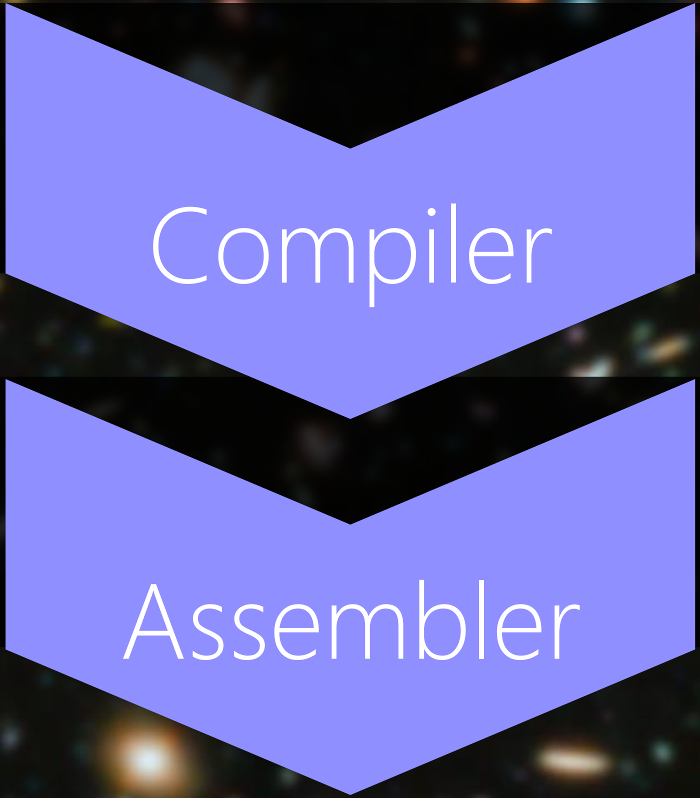
Hello World!

"Then the LORD God formed a man from the dust of the ground and breathed into his nostrils the breath of life, and the man became a living being." - Genesis 2:7

C

```
hfani@alpha:~$ vi hello.c
```

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



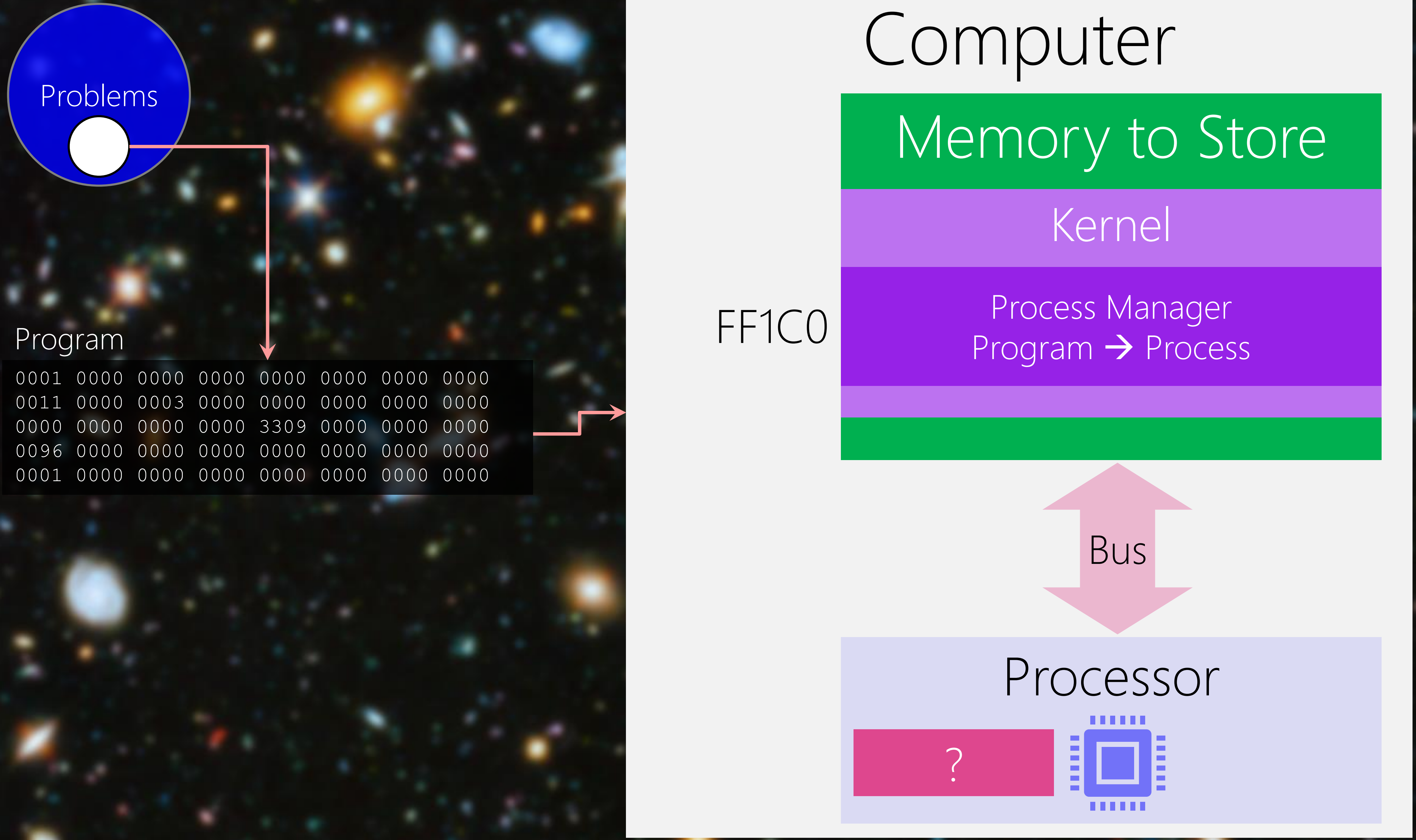
Assembly

```
<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
```

OP Code

```
hfani@alpha:~$ cc hello.c -o hello
```

```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
0001 0000 0000 0000 0000 0000 0000 0000
```

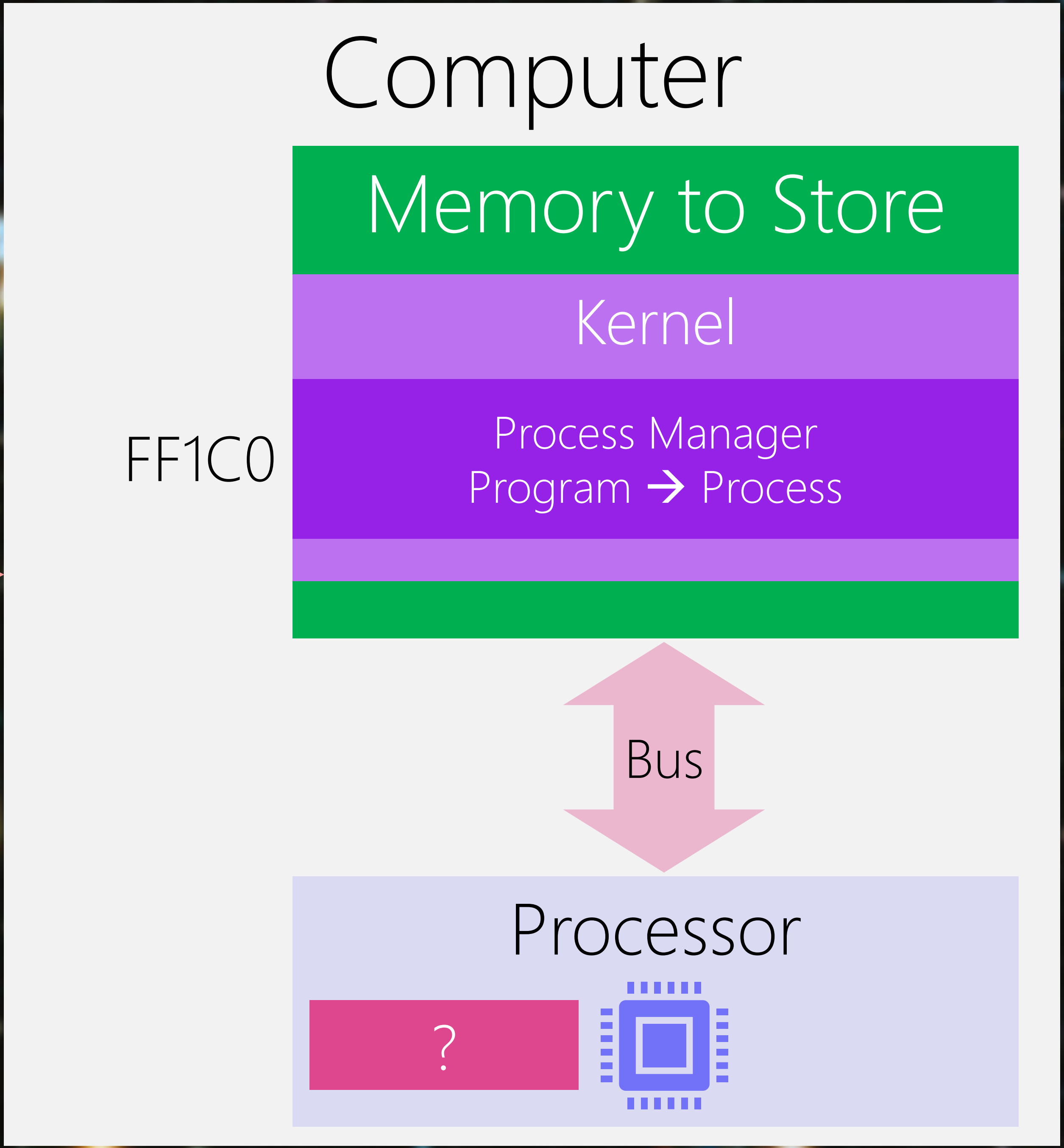


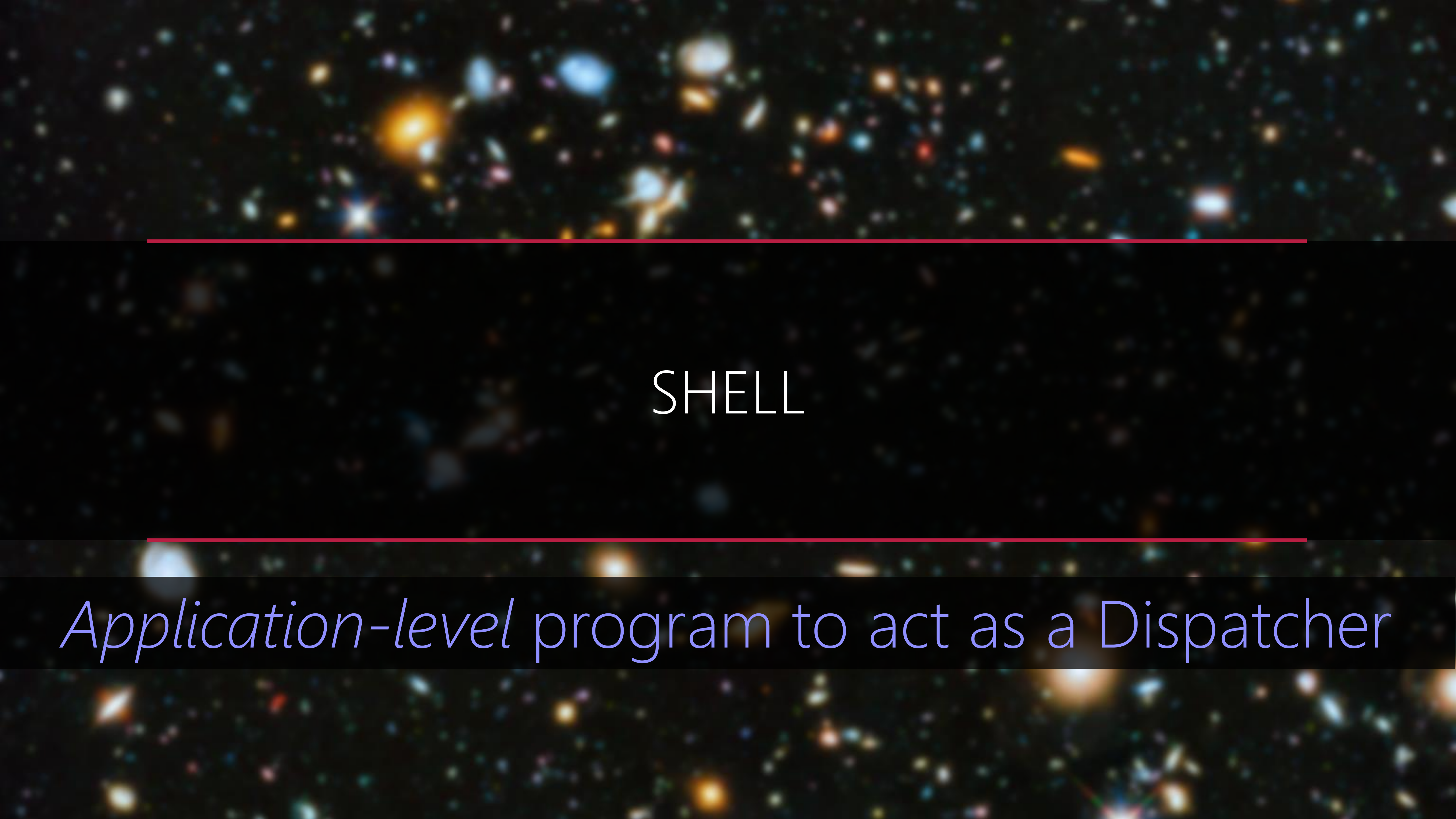
```
0001  0000  0000  0000  0000  0000  0000  0000
0011  0000  0003  0000  0000  0000  0000  0000
0000  0000  0000  0000  3309  0000  0000  0000
0096  0000  0000  0000  0000  0000  0000  0000
0001  0000  0000  0000  0000  0000  0000  0000
```

Program

```
0001  0000  0000  0000  0000  0000  0000  0000
0011  0000  0003  0000  0000  0000  0000  0000
0000  0000  0000  0000  3309  0000  0000  0000
0096  0000  0000  0000  0000  0000  0000  0000
0001  0000  0000  0000  0000  0000  0000  0000
```

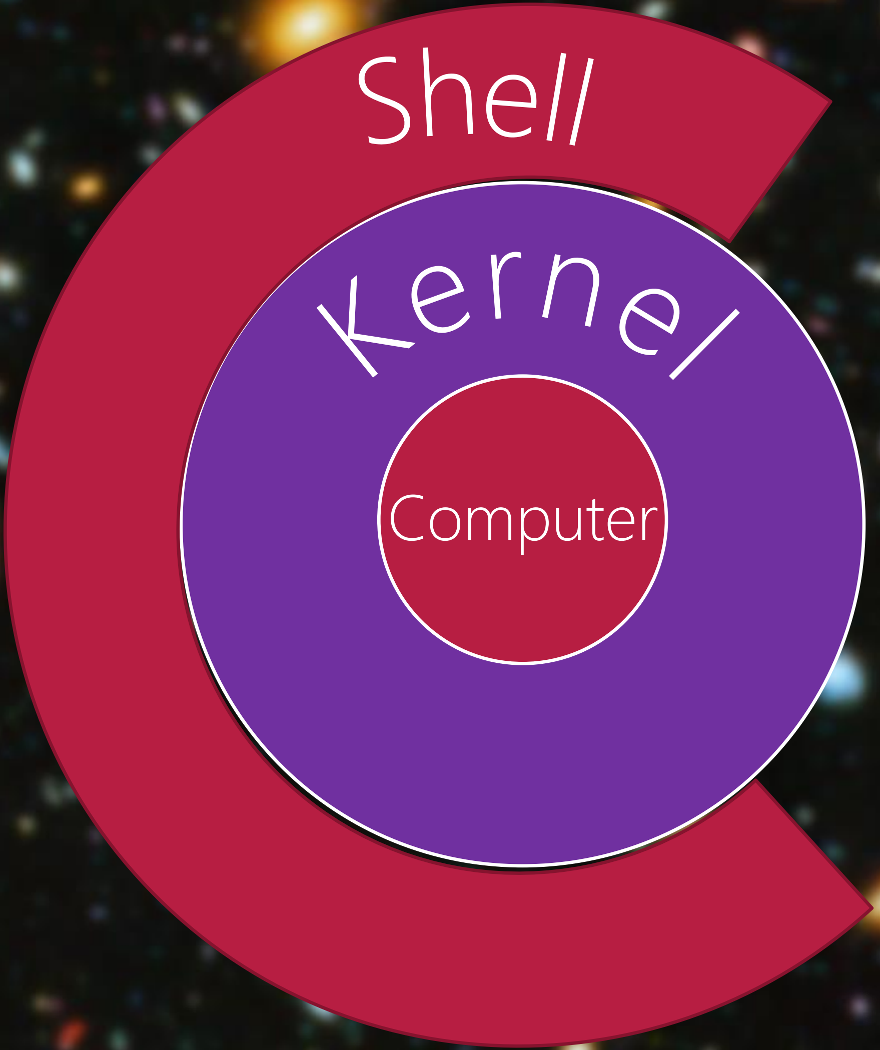
1. Locate the program: /home/hfani/…
2. Call Process Manager for loading the program
   IP = FF1C0
3. Point IP to the first line of the program
   IP = &first opcode

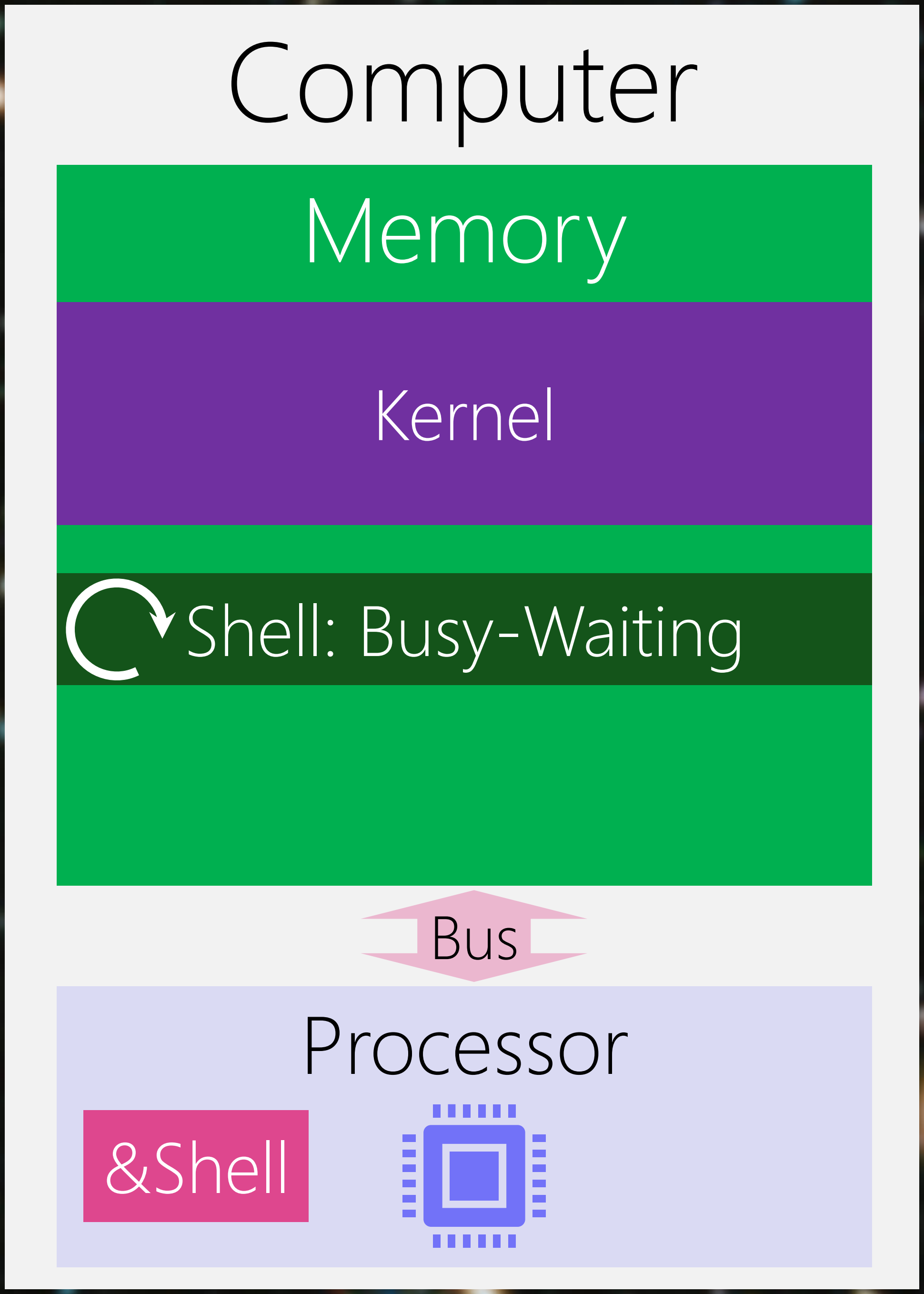


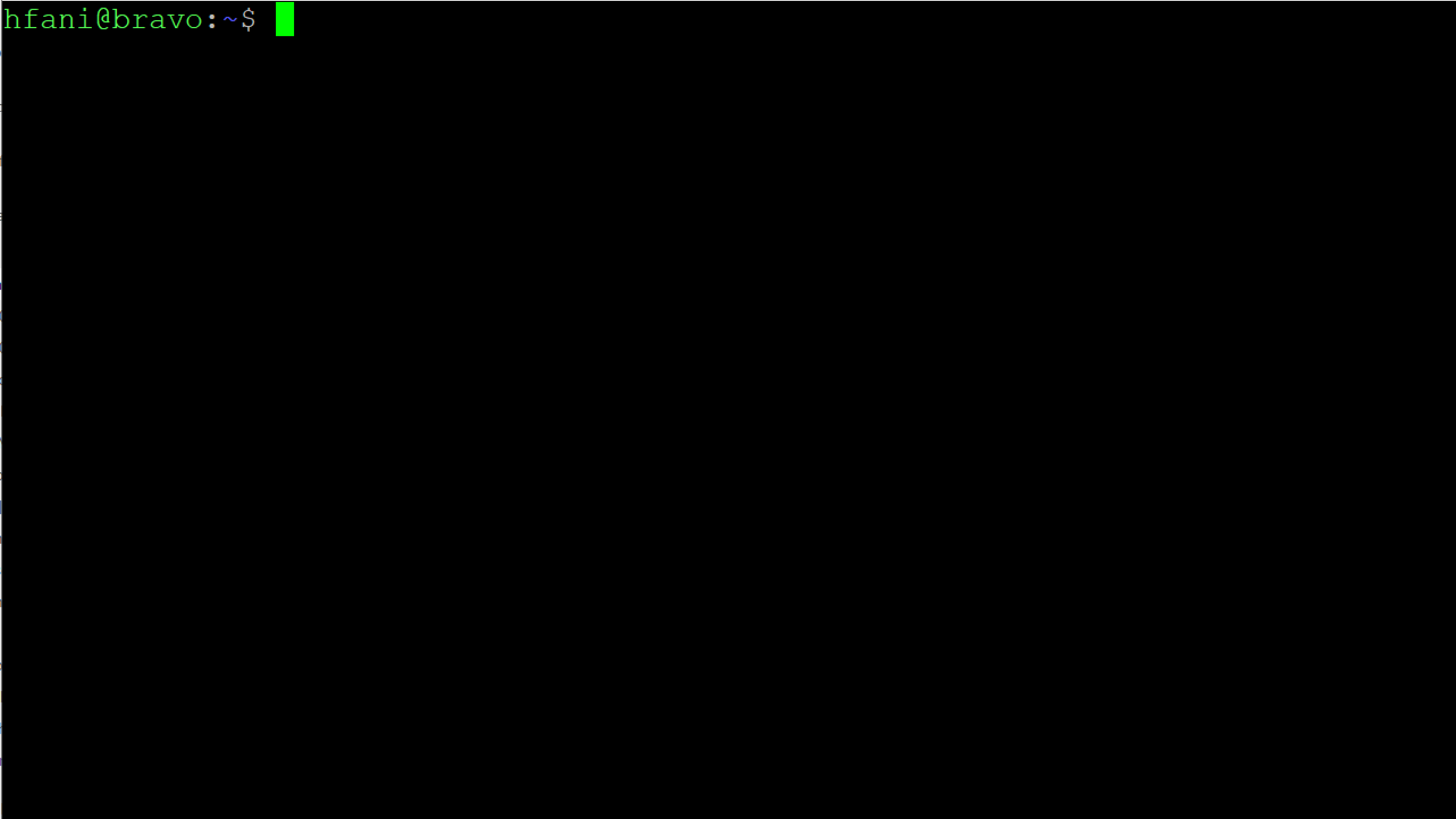


SHELL

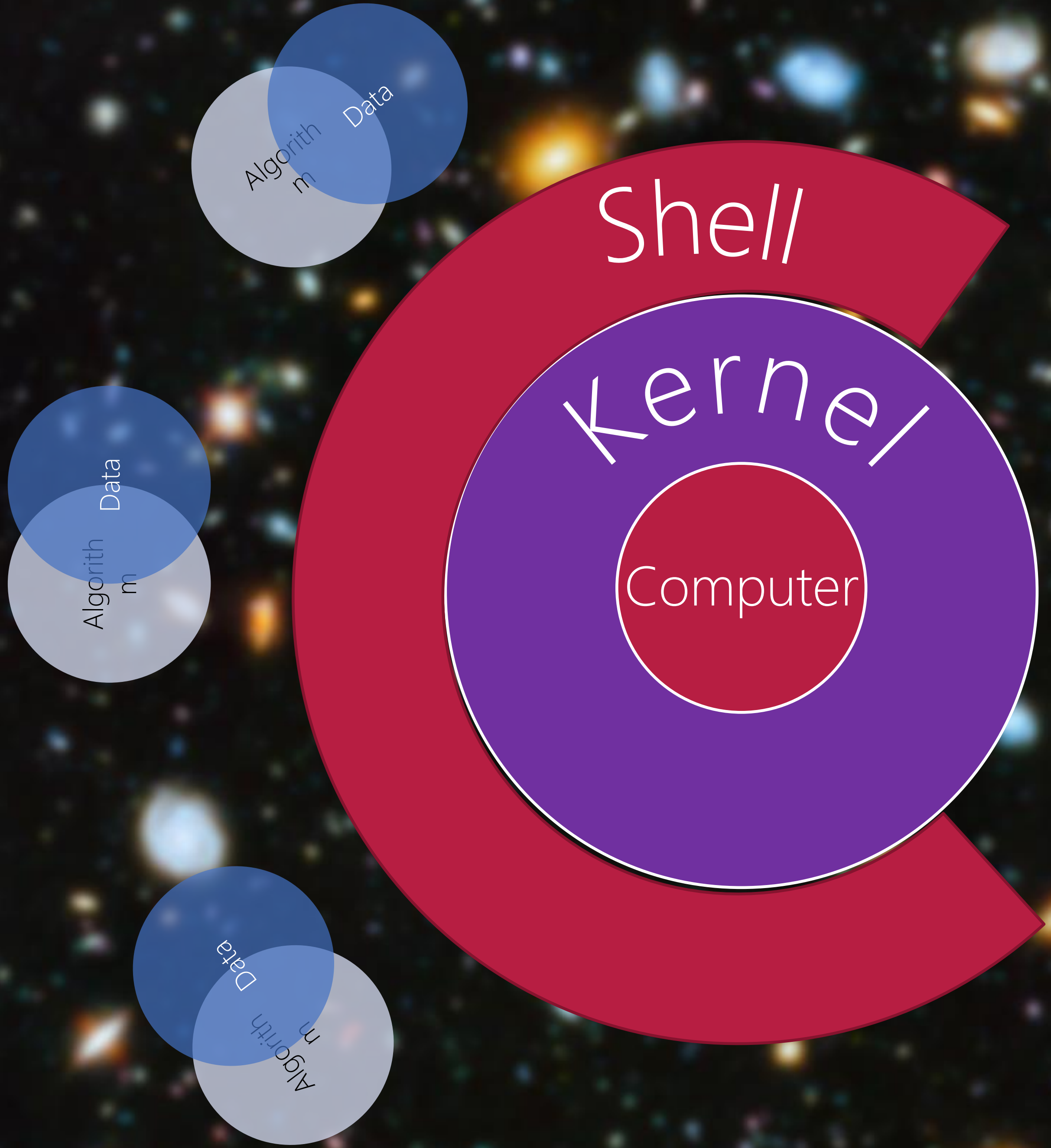
Application-level program to act as a Dispatcher

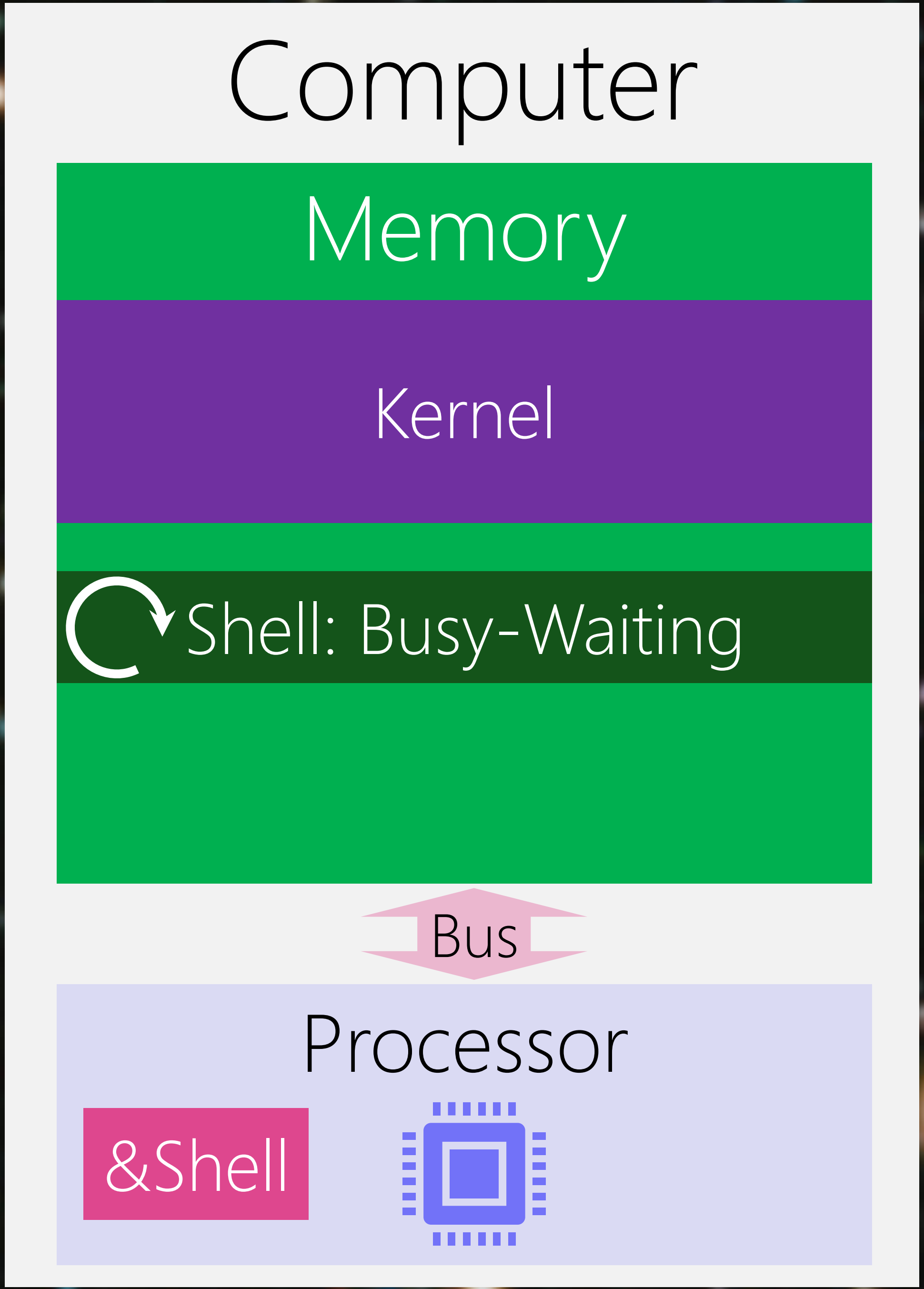

Shell
Kernel
Computer

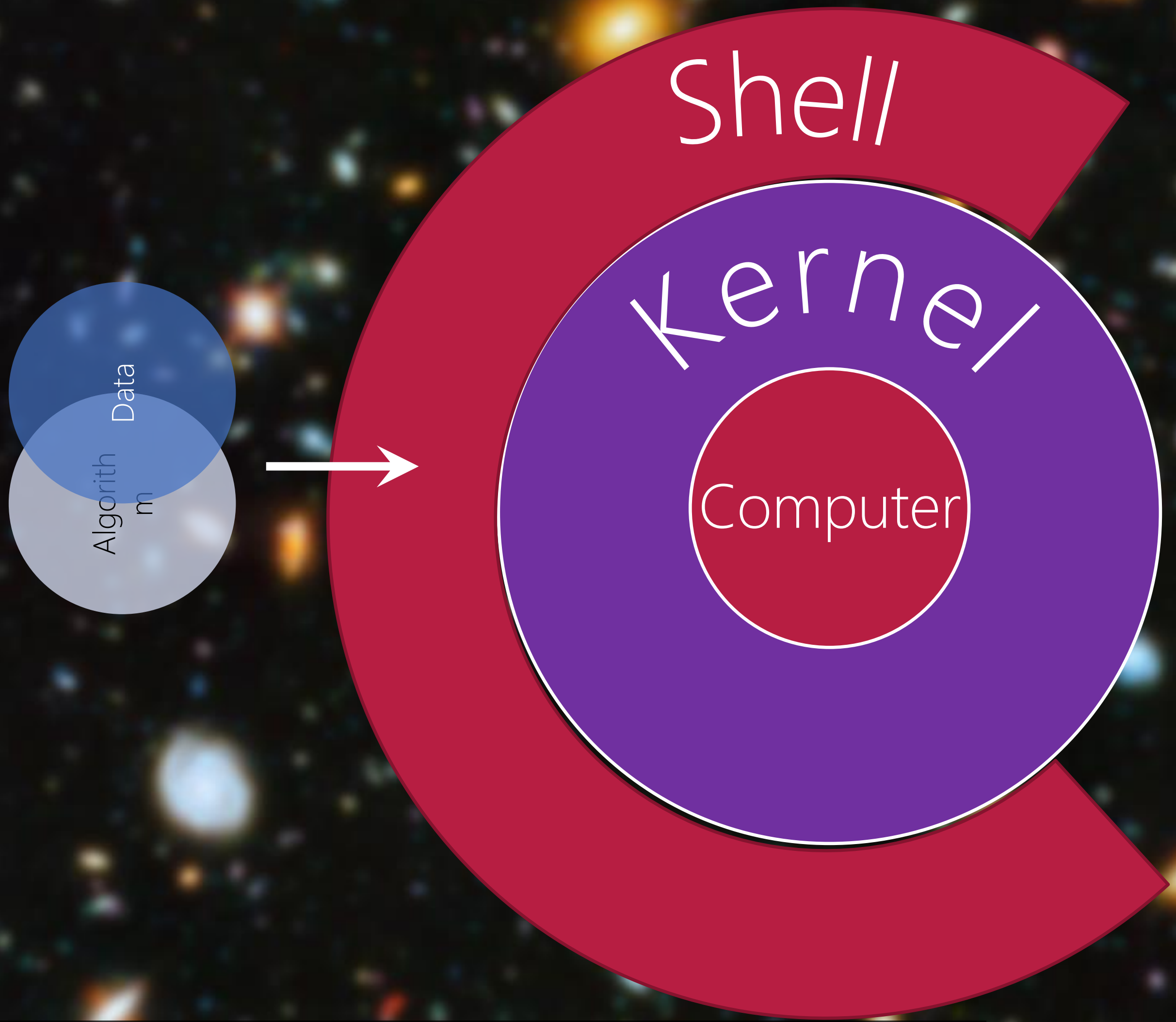

Computer
Memory
Kernel
Shell: Busy-Waiting
Bus
Processor
&Shell

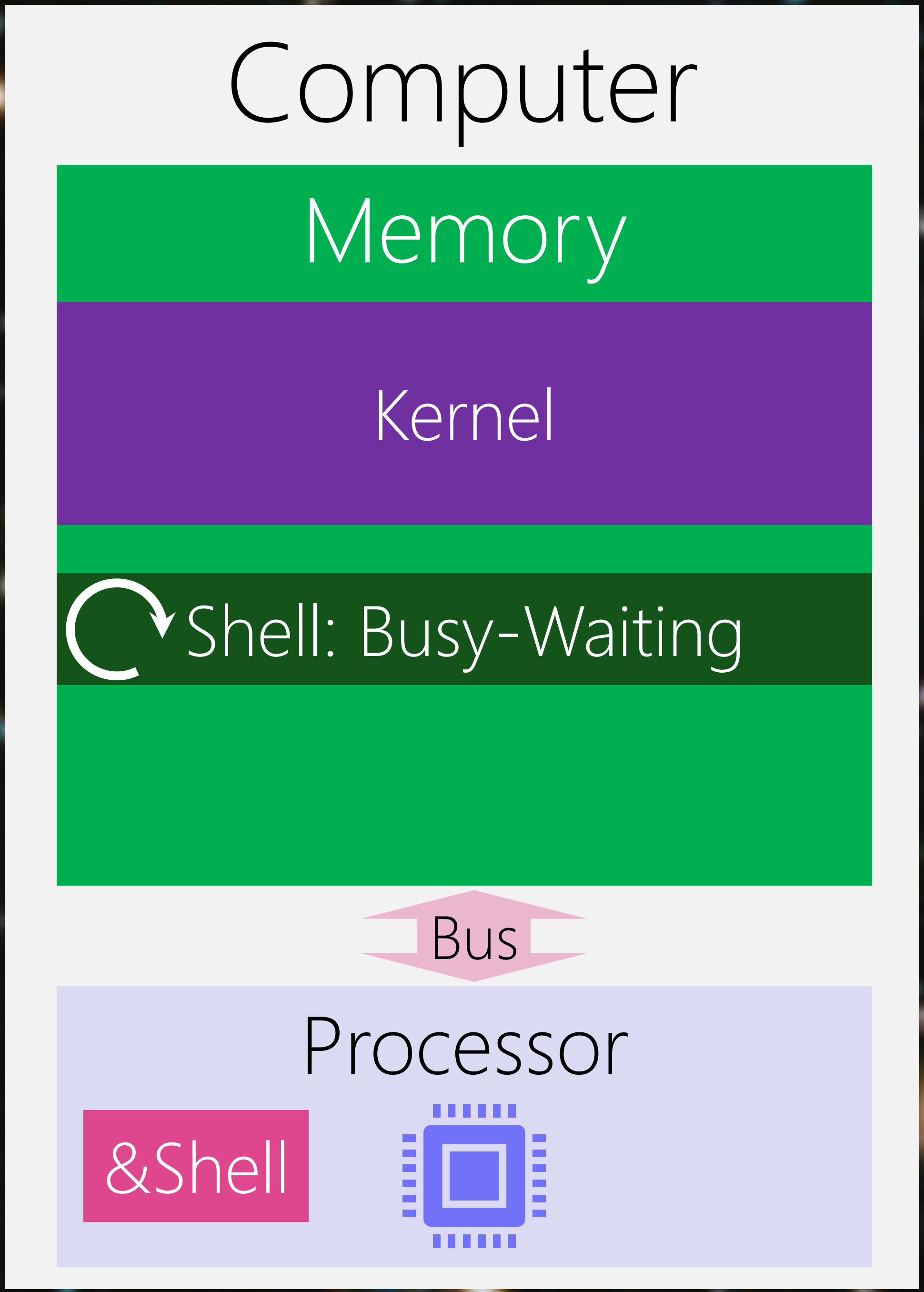


```
hfani@bravo:~$ 
```

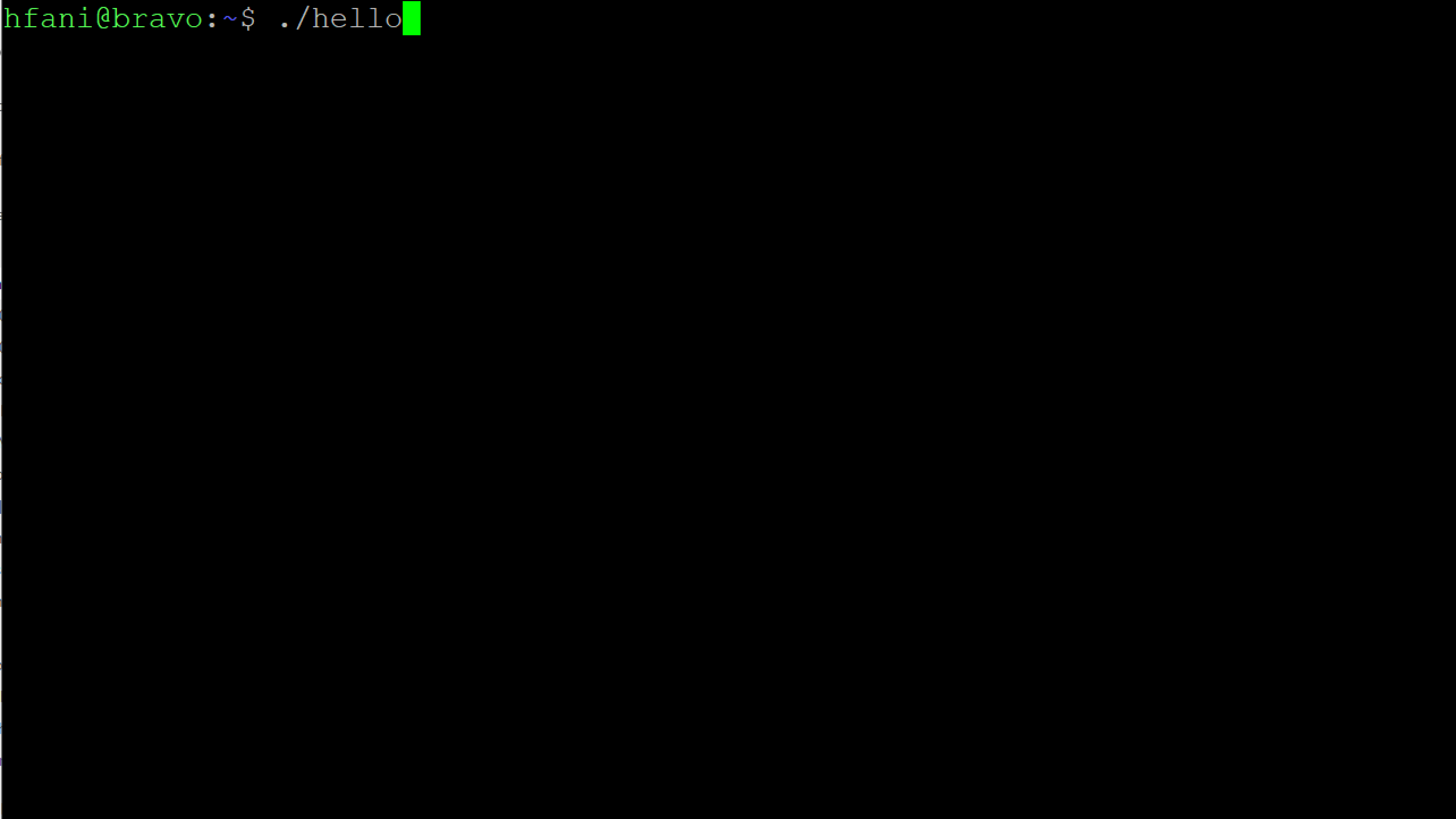

Algorithm
Data
Algorithm
Data
Algorithm
Data
Shell
Kernel
Computer


Computer
Memory
Kernel
Shell: Busy-Waiting
Bus
Processor
&Shell

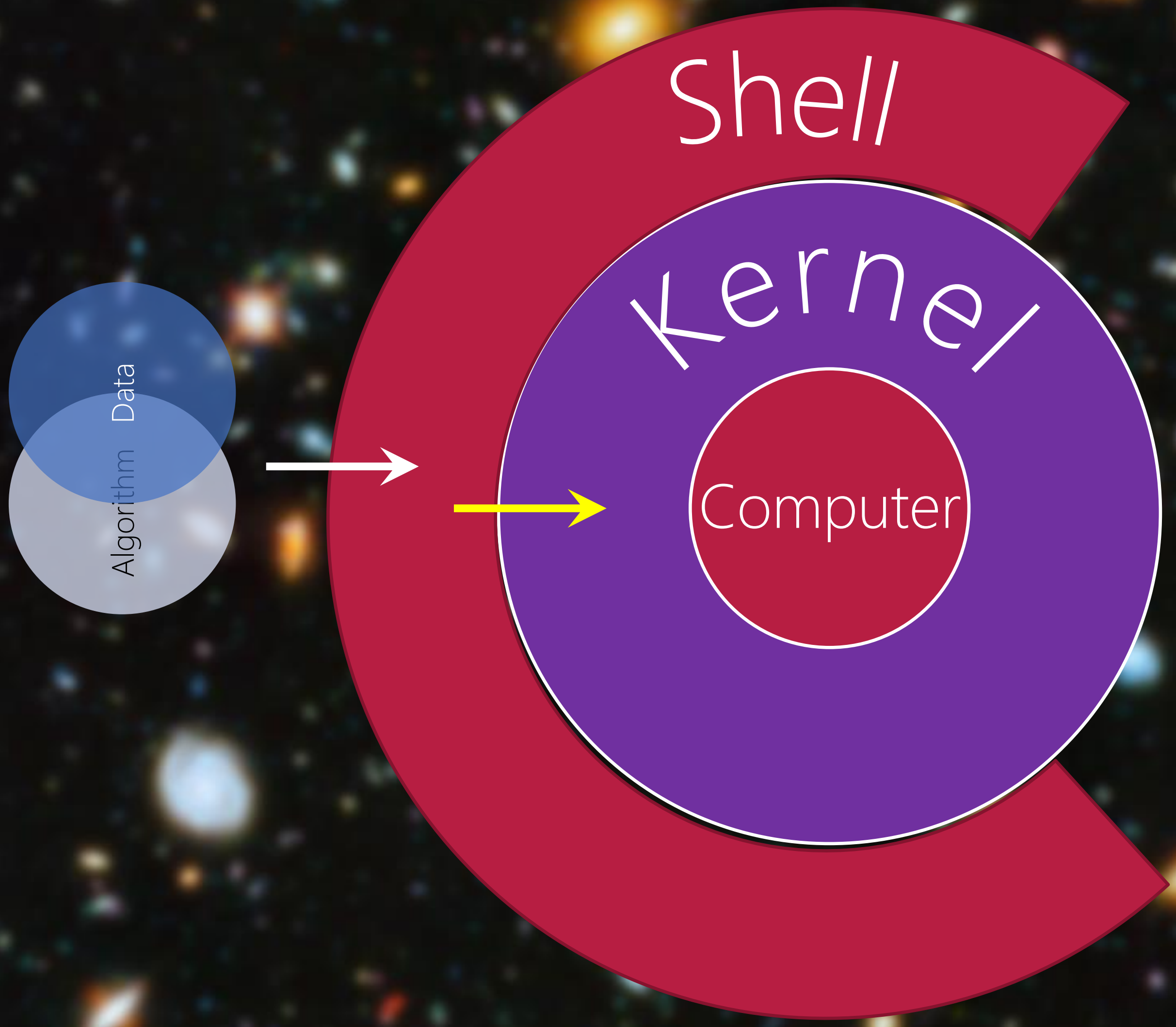


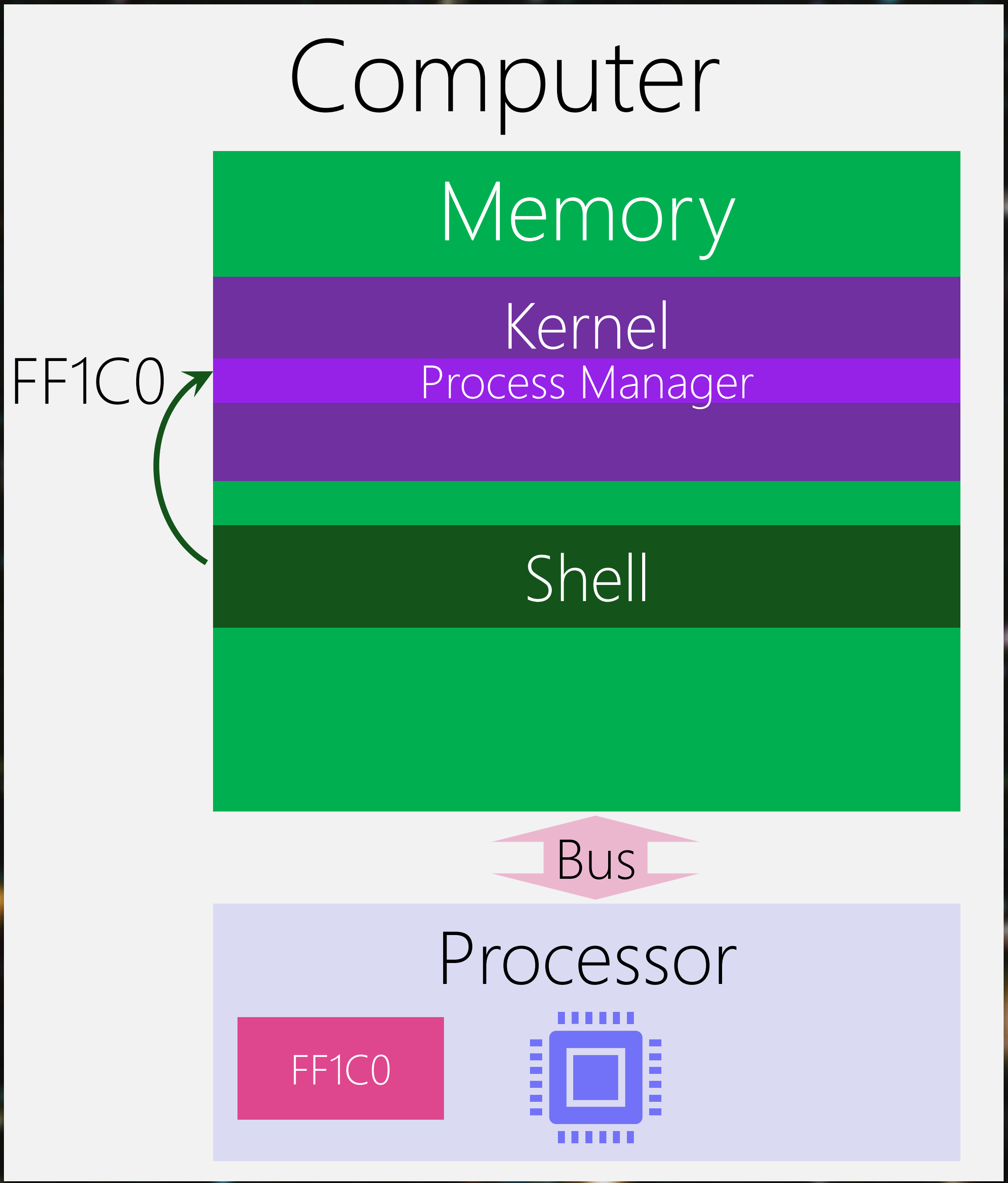


1. Locate the program: /home/hfani/...

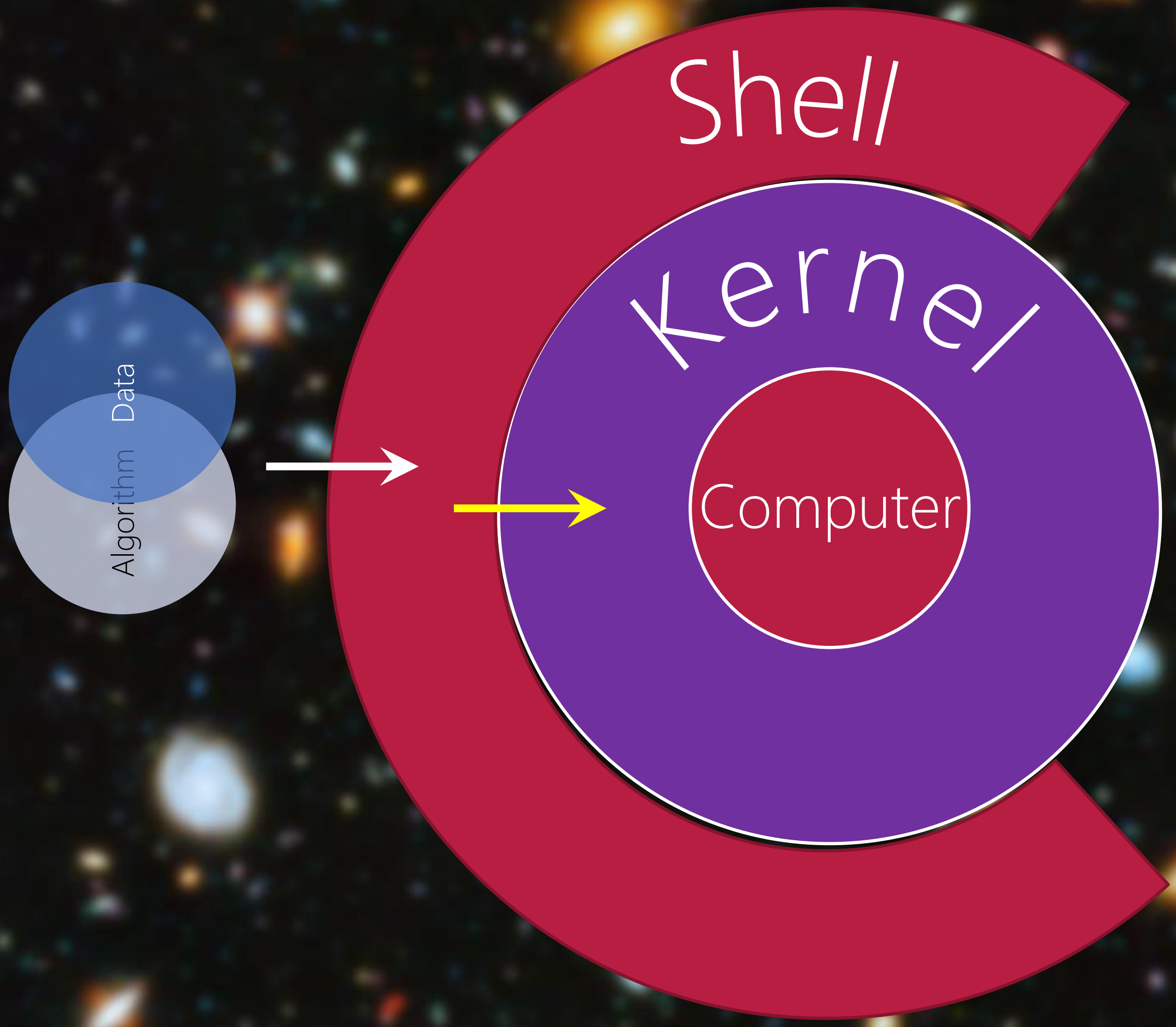


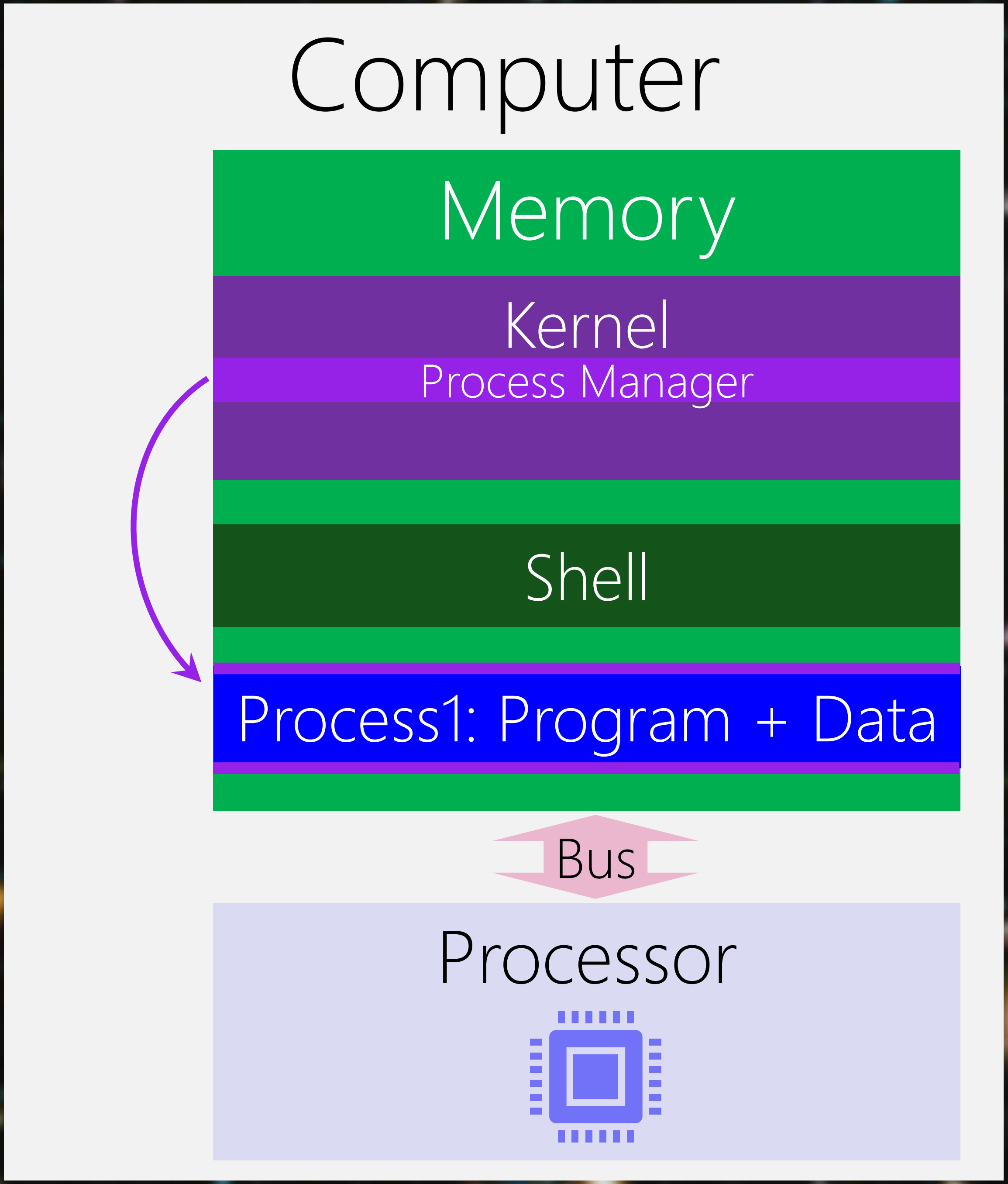
```
hfani@bravo:~$ ./hello
```

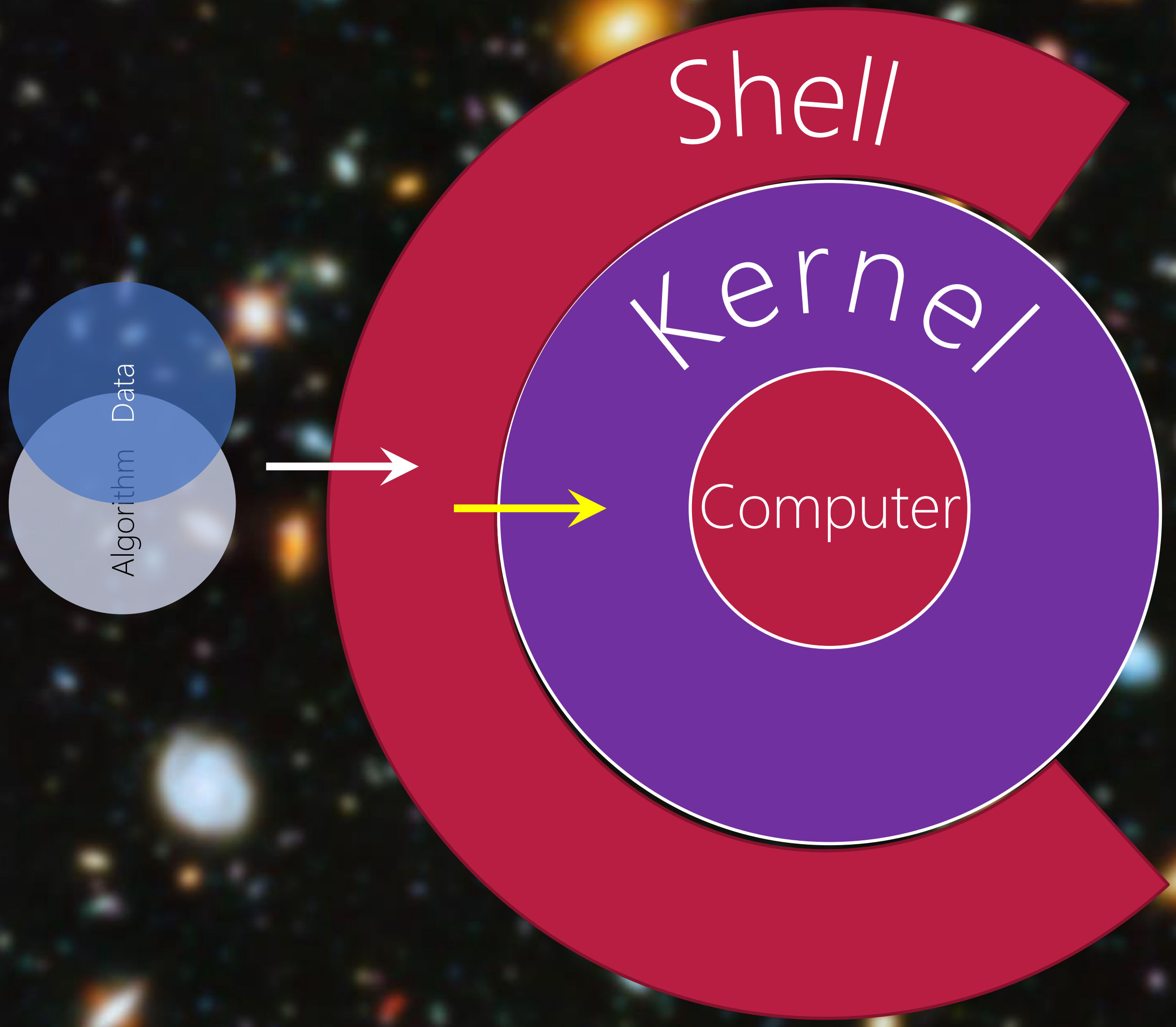


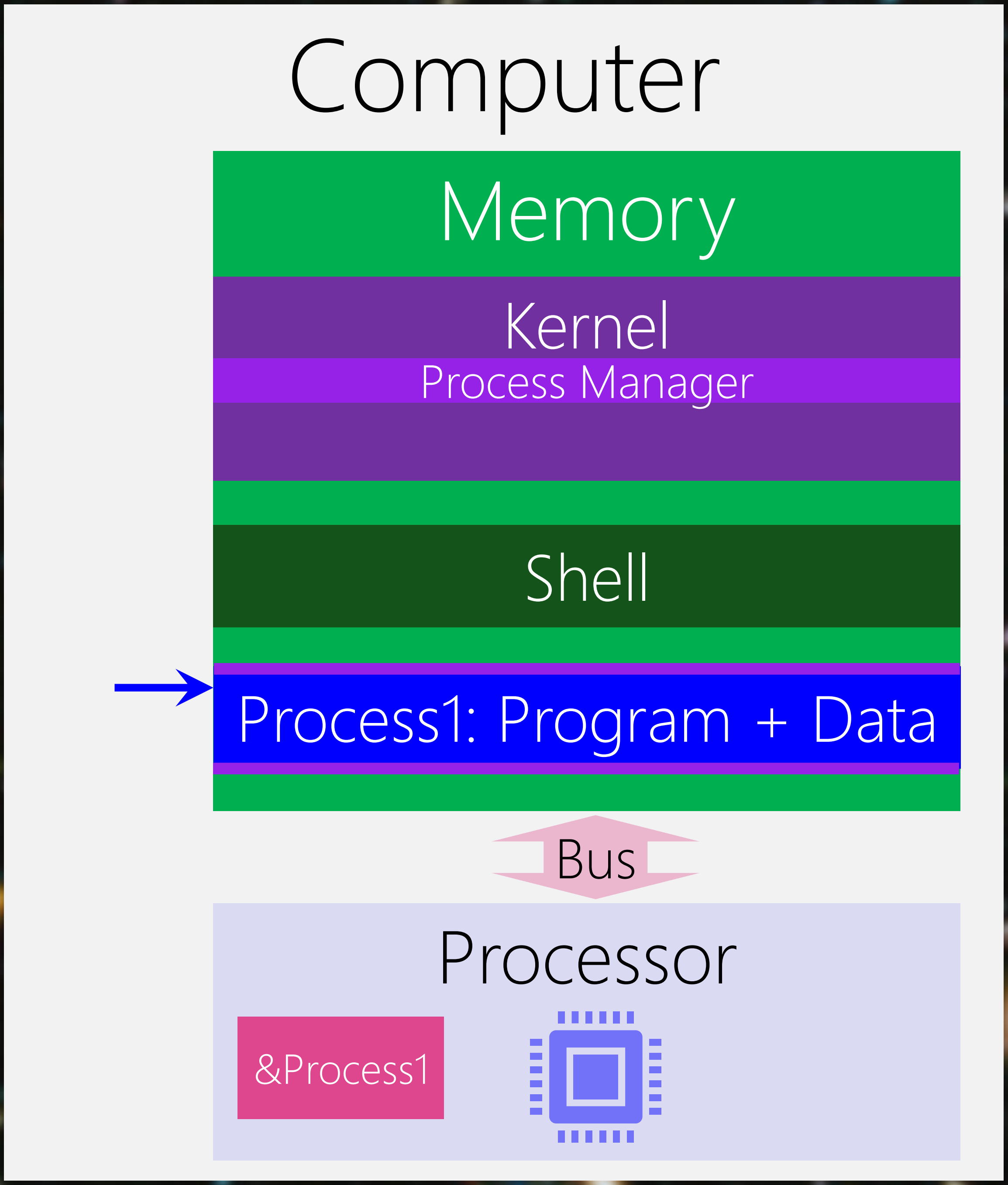


2. Call Process Manager for loading the program: IP = FF1C0

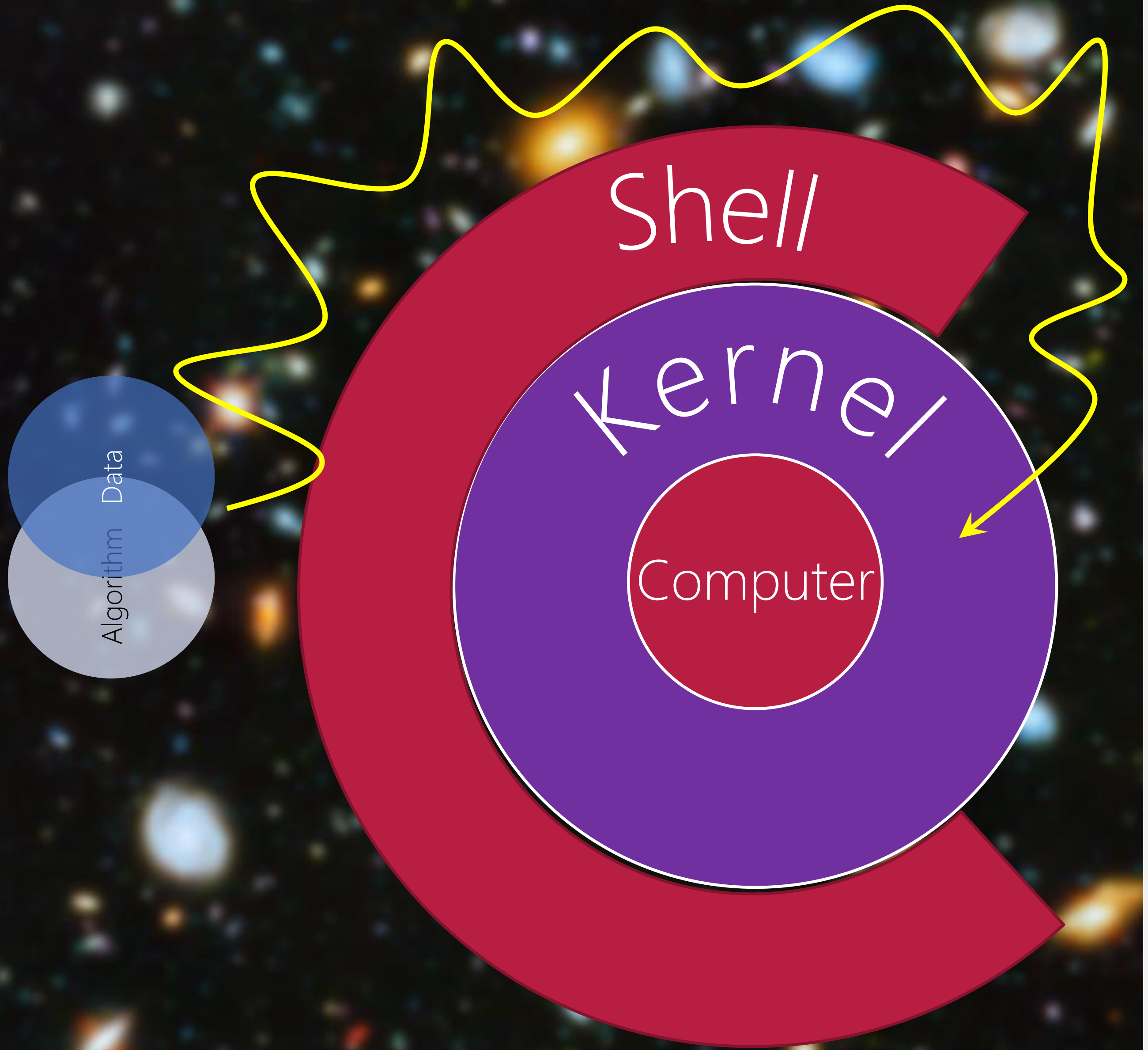

Data
Algorithm
Shell
Kernel
Computer

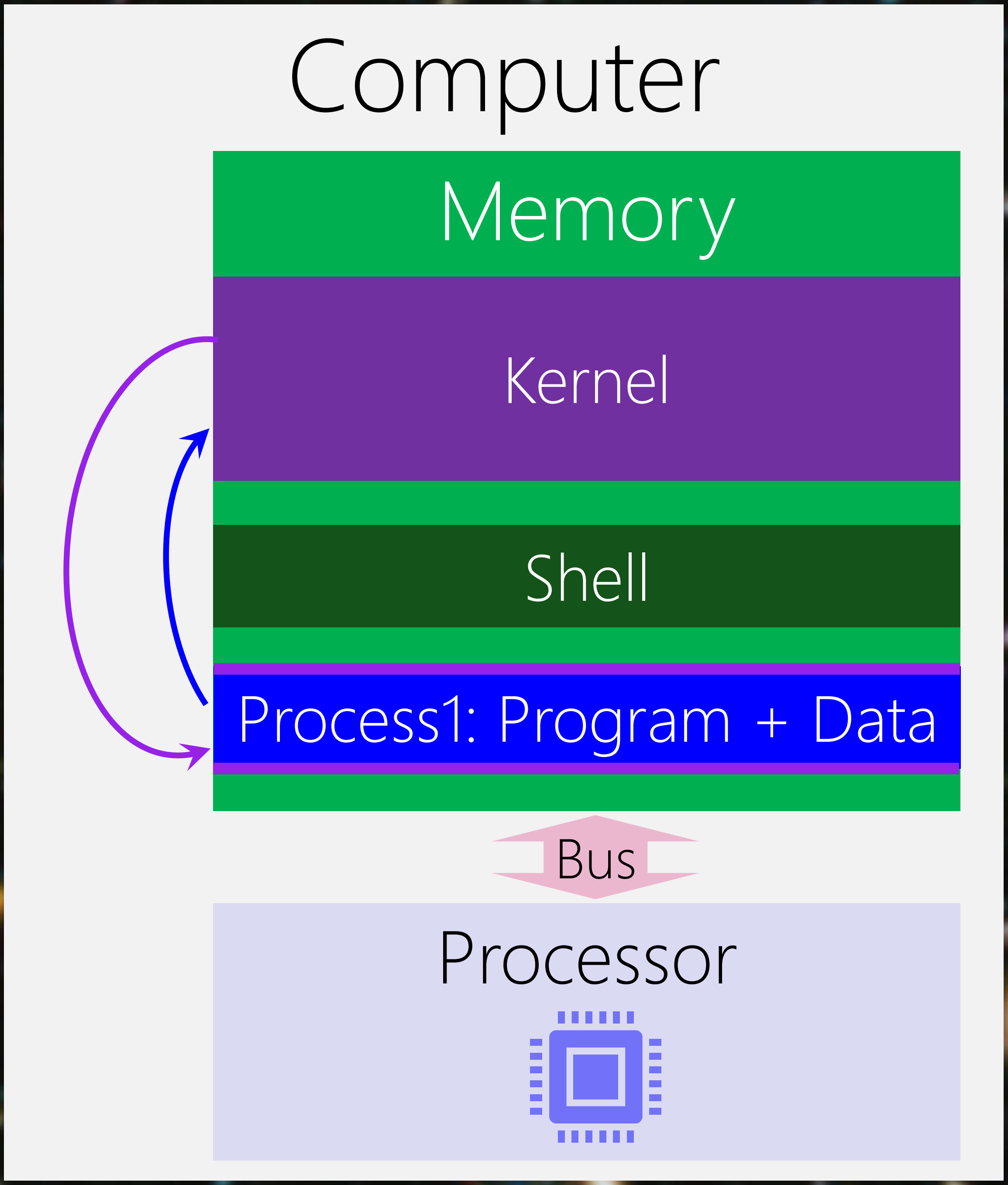

Computer
Memory
Kernel
Process Manager
Shell
Process1: Program + Data
Bus
Processor

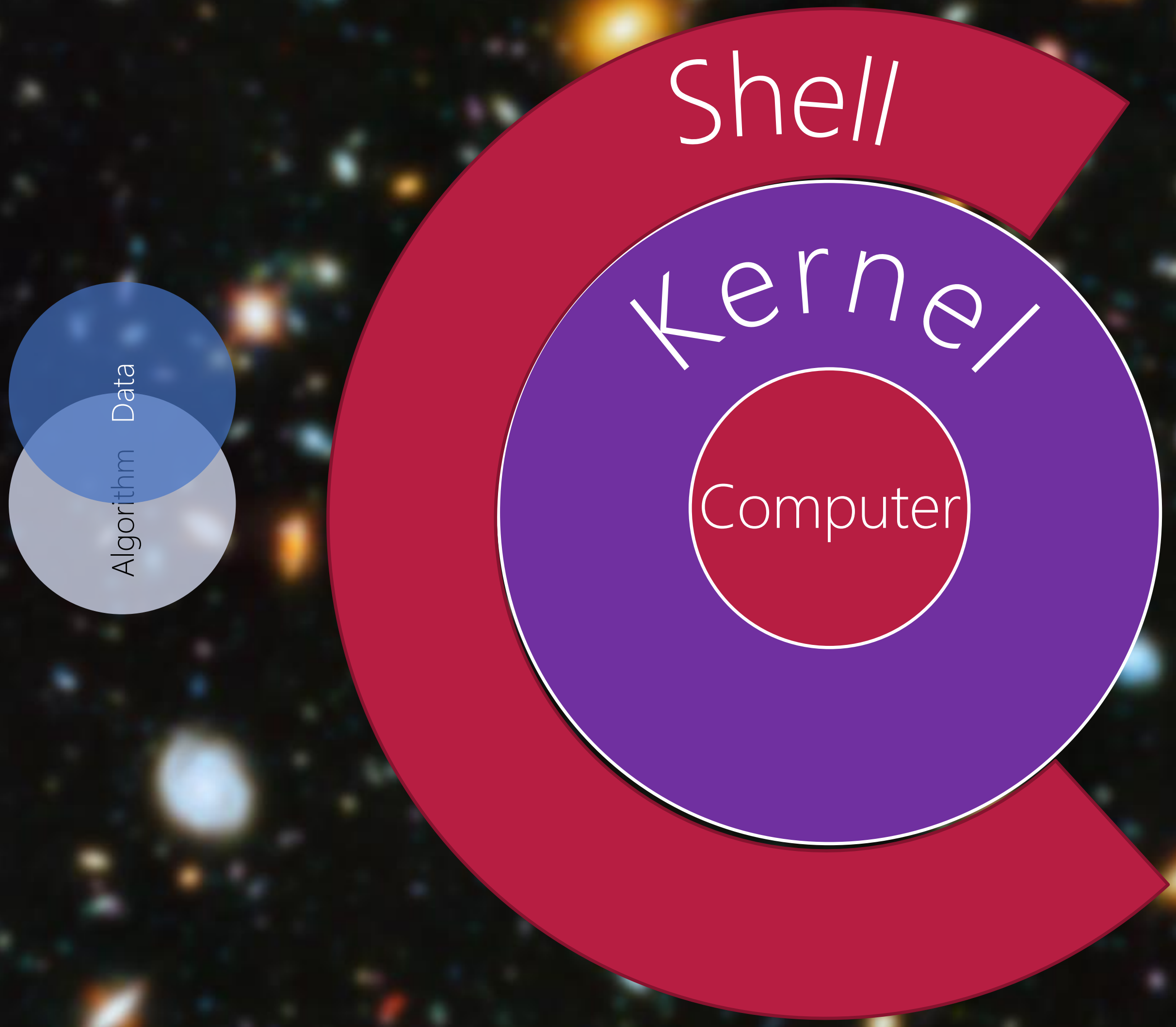


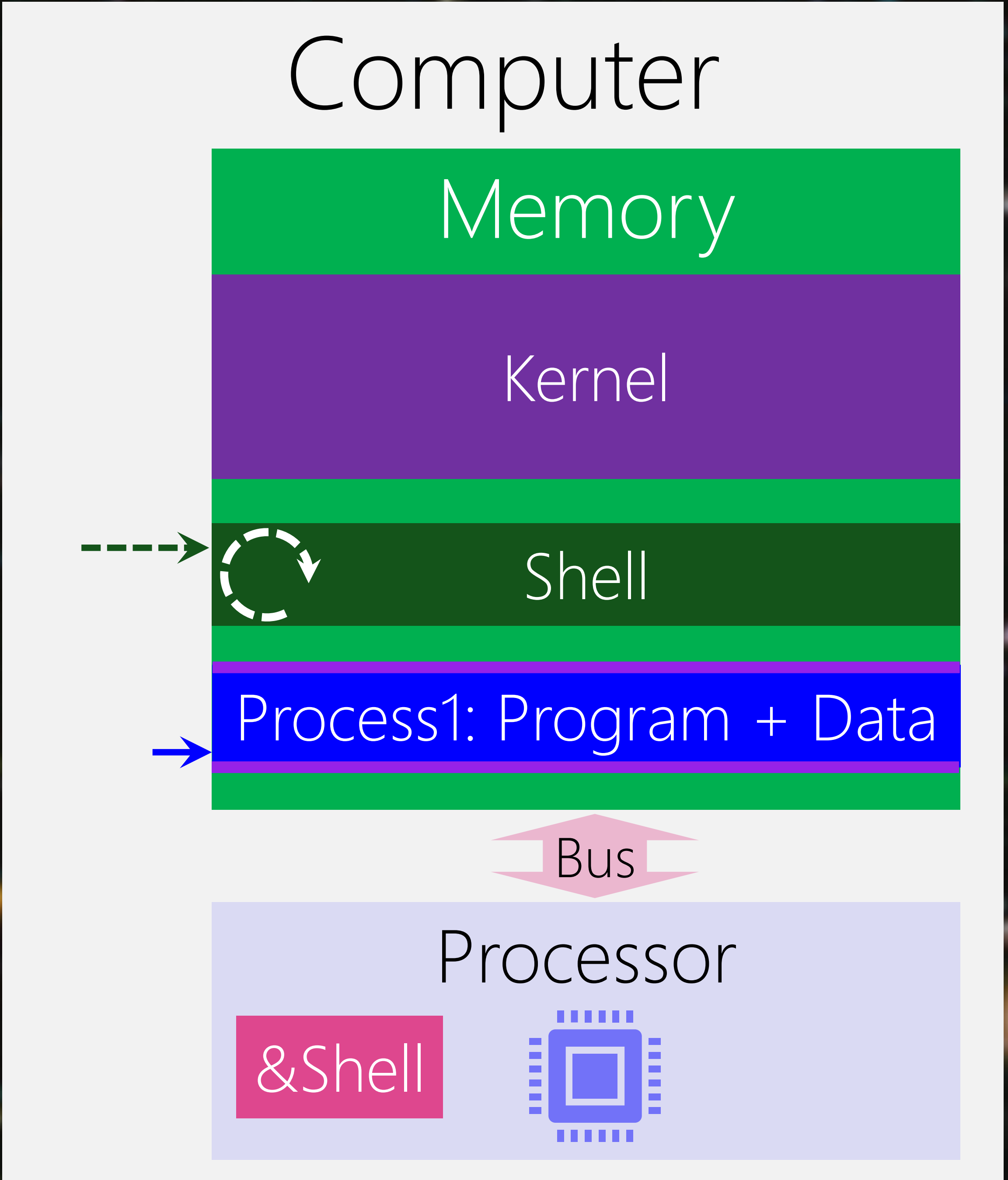


3. Point IP to the first line of the program: IP = &first opcode

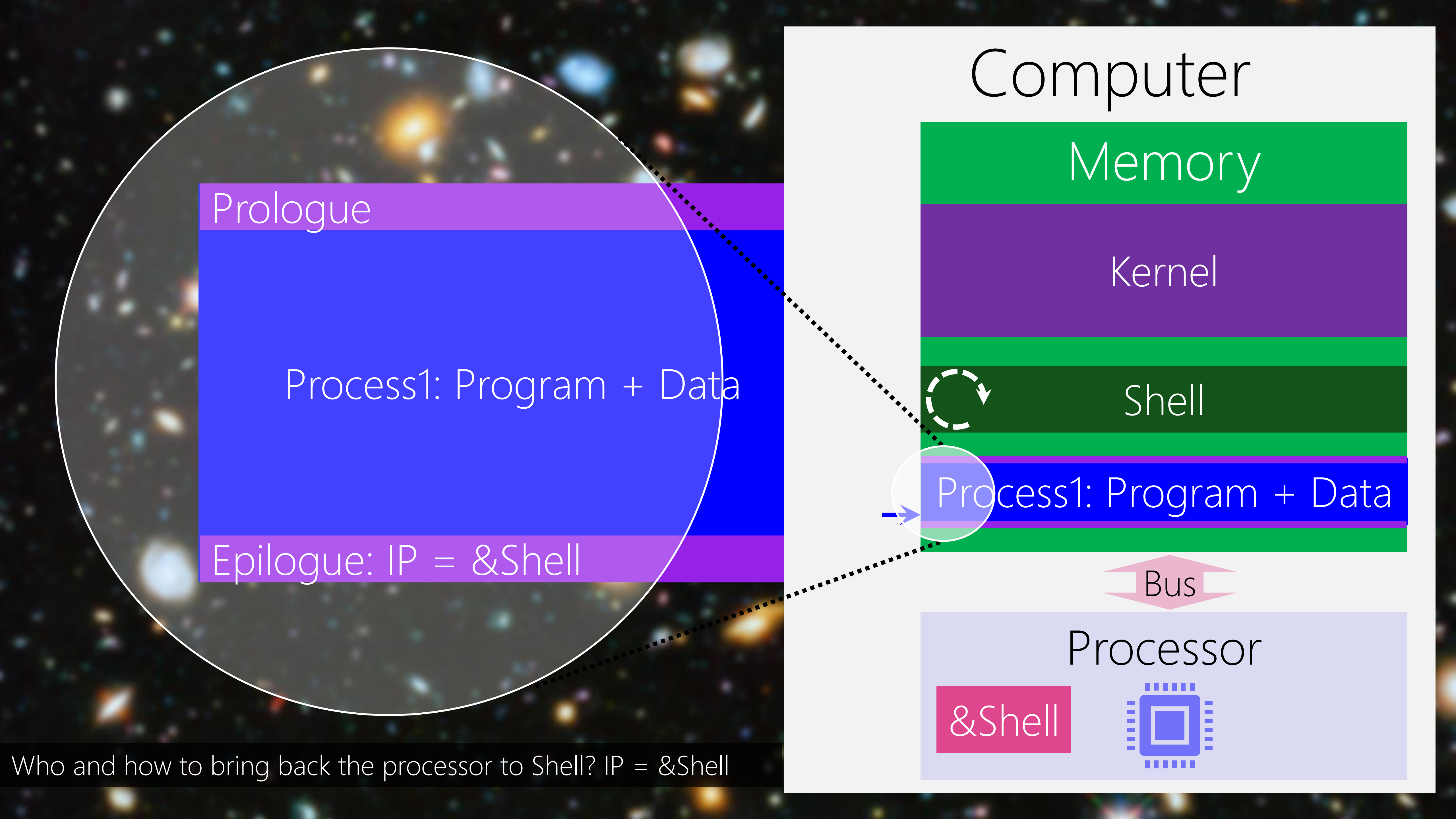

Data
Algorithm
Shell
Kernel
Computer


Computer
Memory
Kernel
Shell
Process1: Program + Data
Bus
Processor



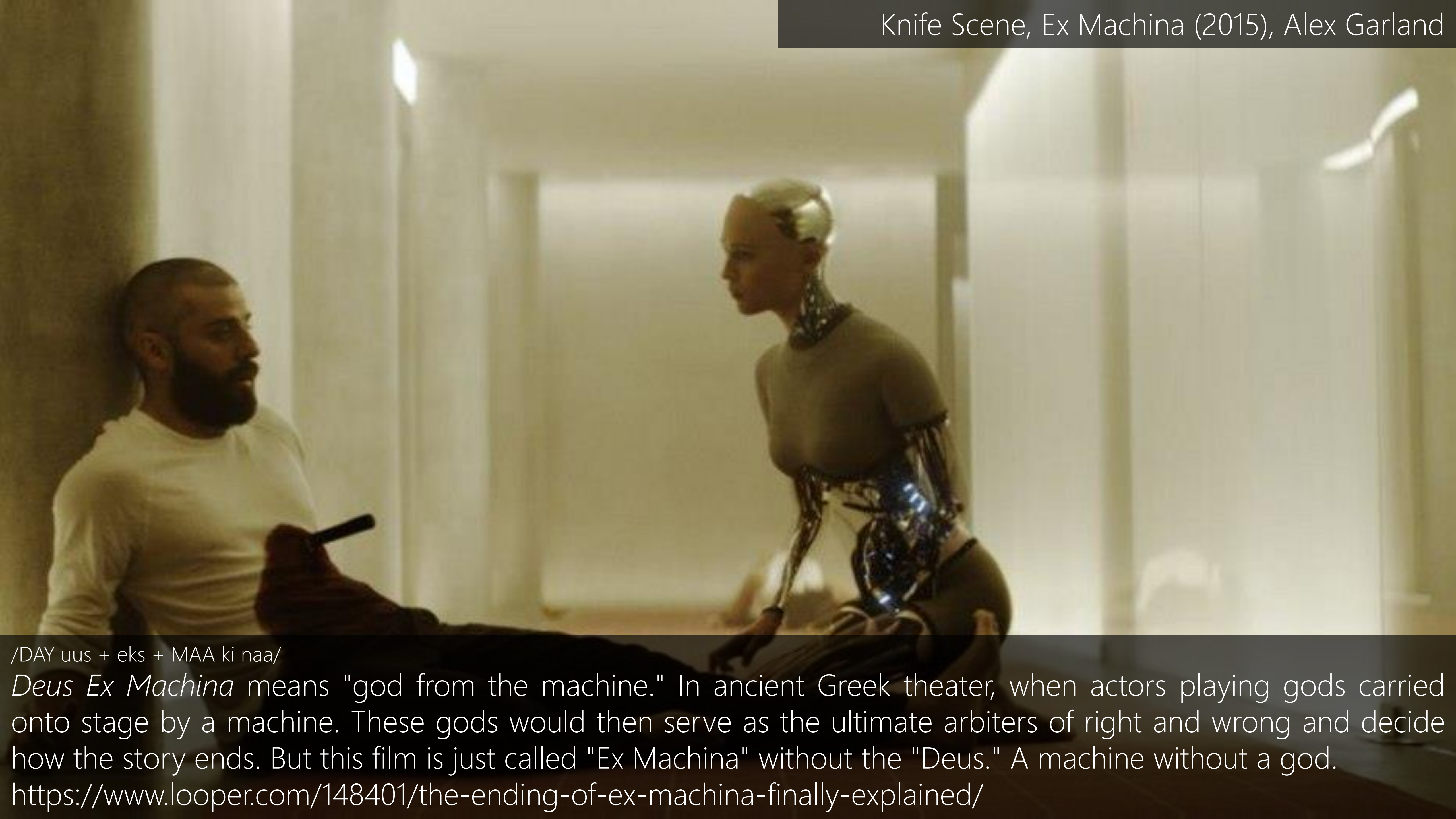


Who and how to brings back the processor to Shell? IP = &Shell



Who and how to bring back the processor to Shell? IP = &Shell

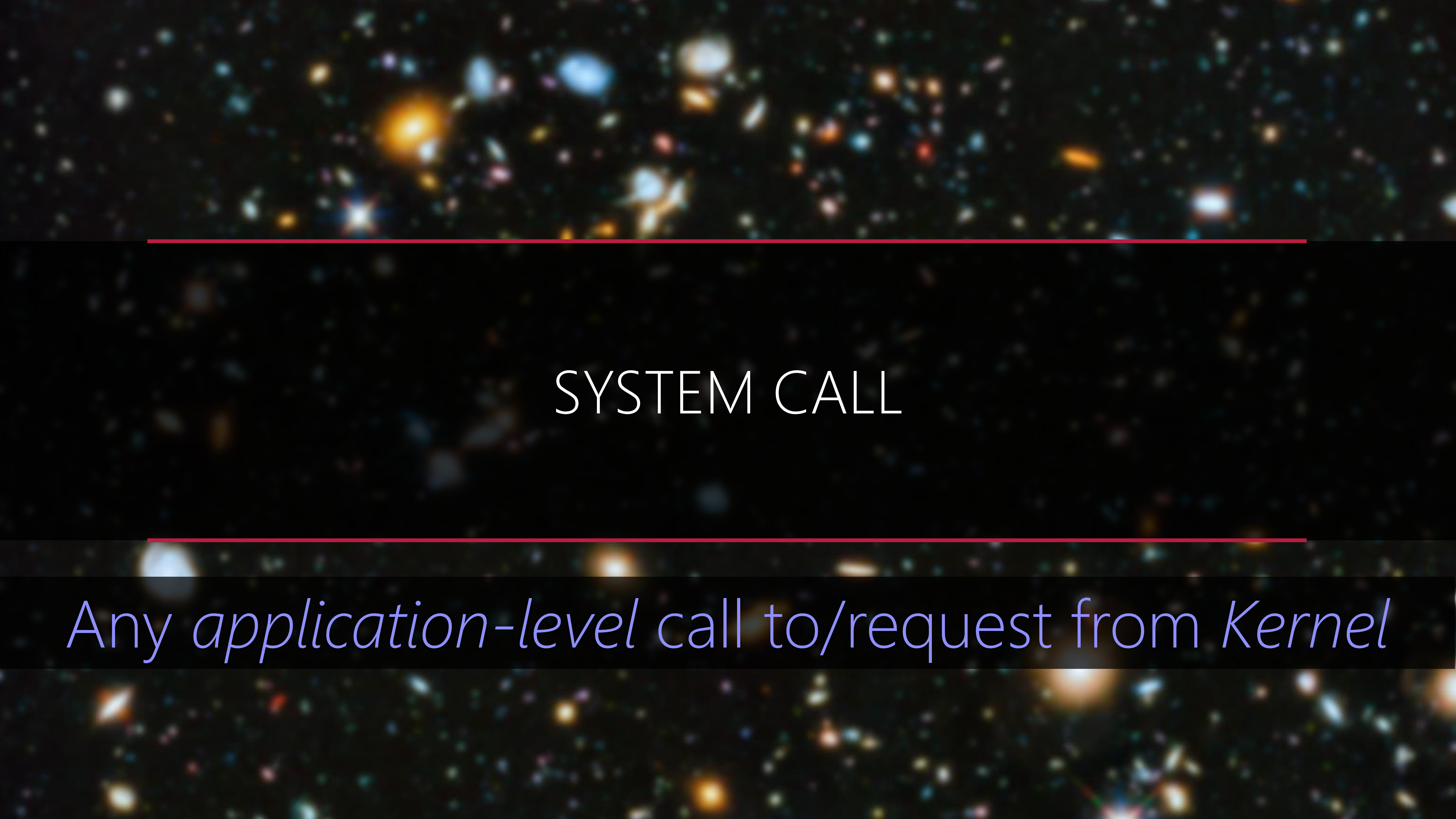
Knife Scene, Ex Machina (2015), Alex Garland



/DAY uus + eks + MAA ki naa/

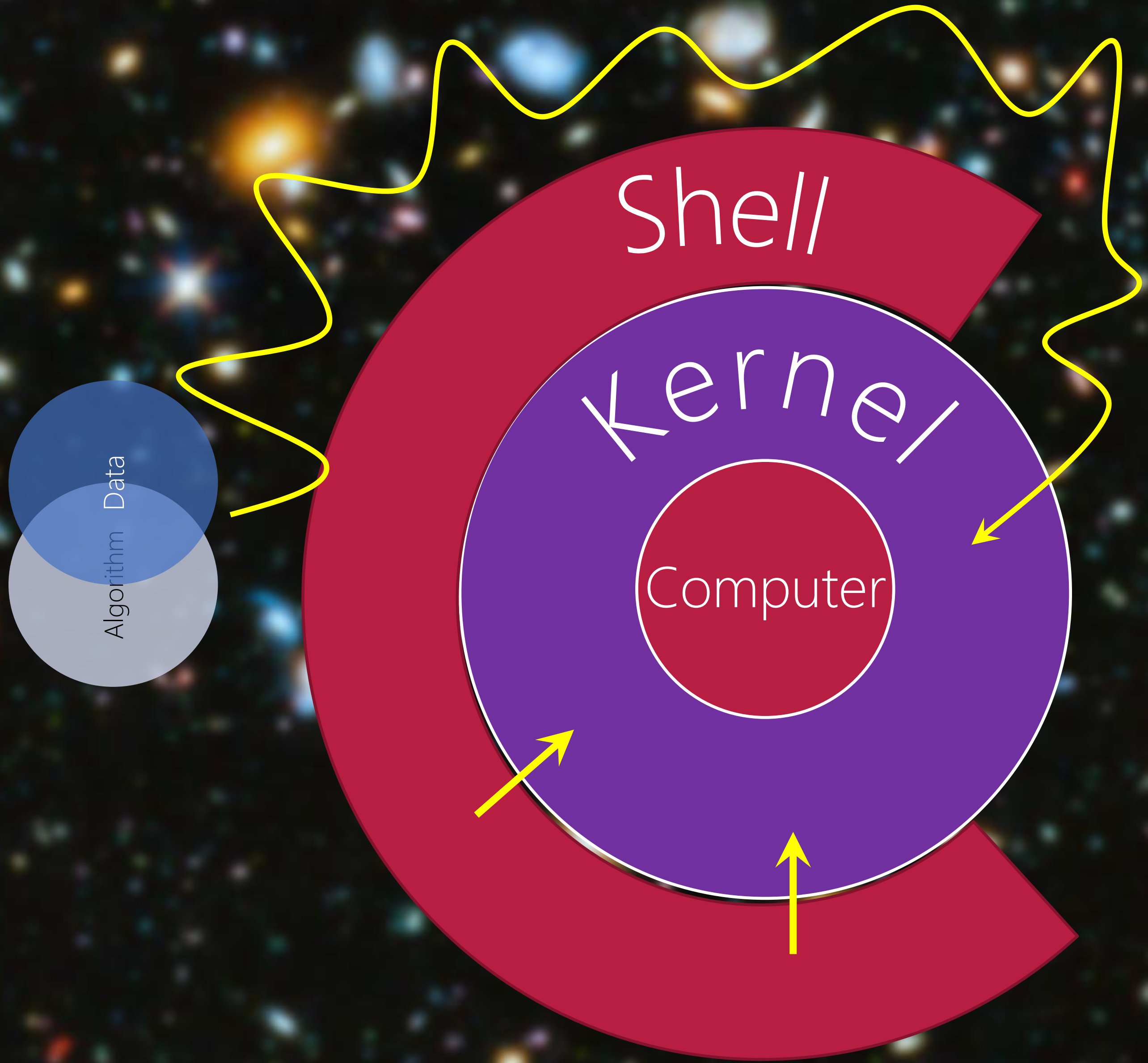
Deus Ex Machina means "god from the machine." In ancient Greek theater, when actors playing gods carried onto stage by a machine. These gods would then serve as the ultimate arbiters of right and wrong and decide how the story ends. But this film is just called "Ex Machina" without the "Deus." A machine without a god.

https://www.looper.com/148401/the-ending-of-ex-machina-finally-explained/



SYSTEM CALL

Any *application-level* call to/request from *Kernel*

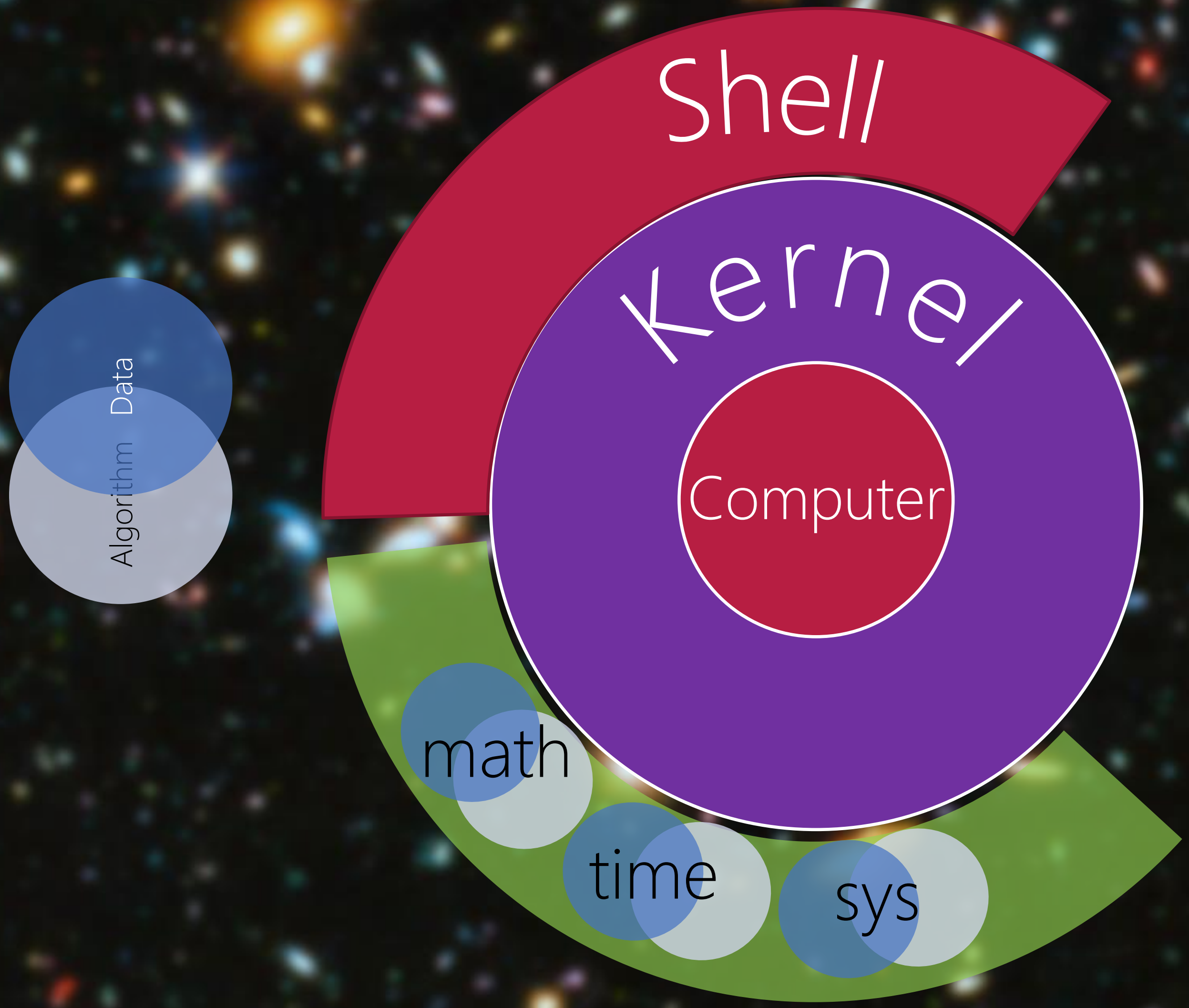

Shell
Kernel
Computer
Data
Algorithm

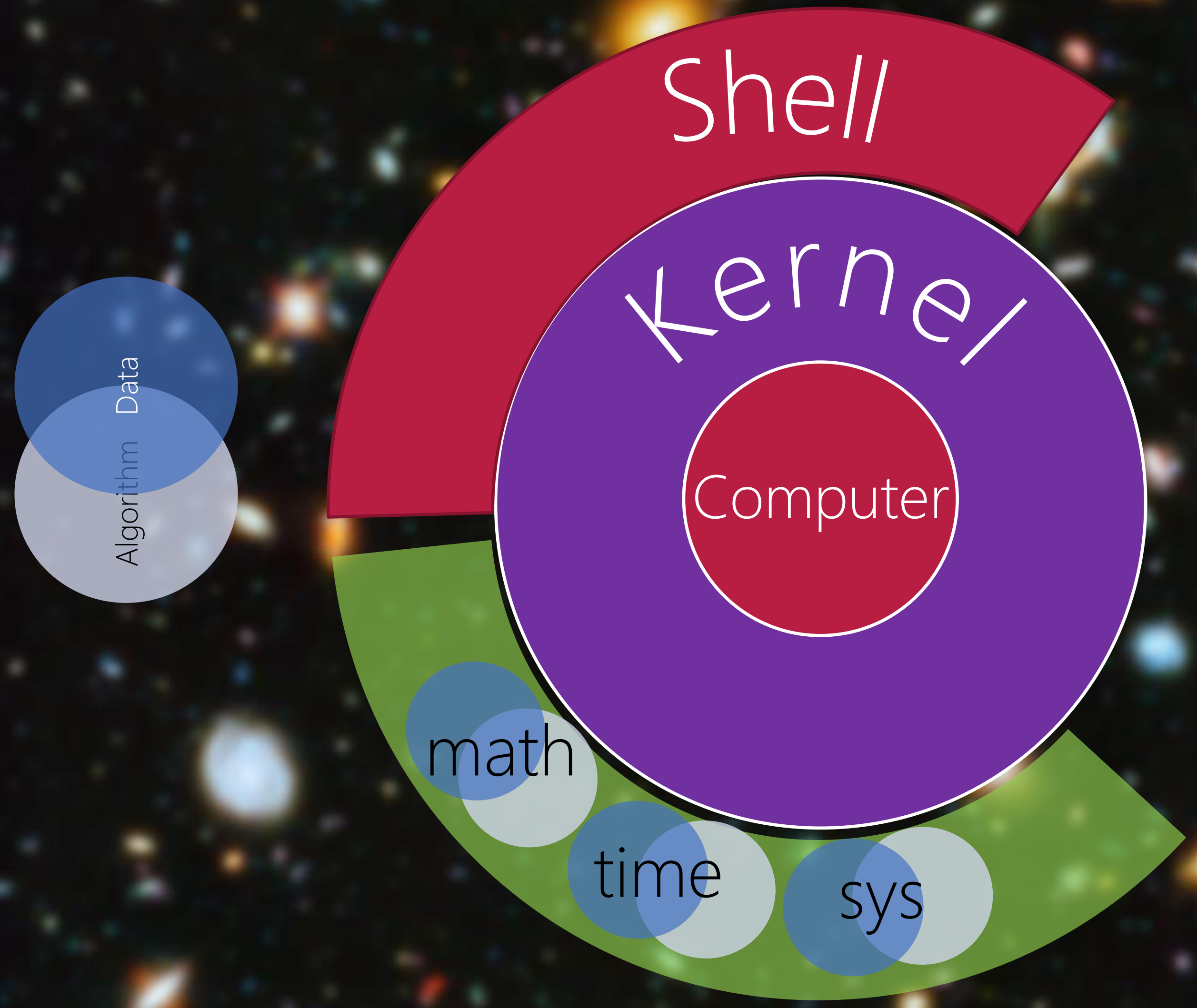
SYSTEM CALL
vs.
Interrupt Request Handler

Both are calls to Kernel, but what is the difference?

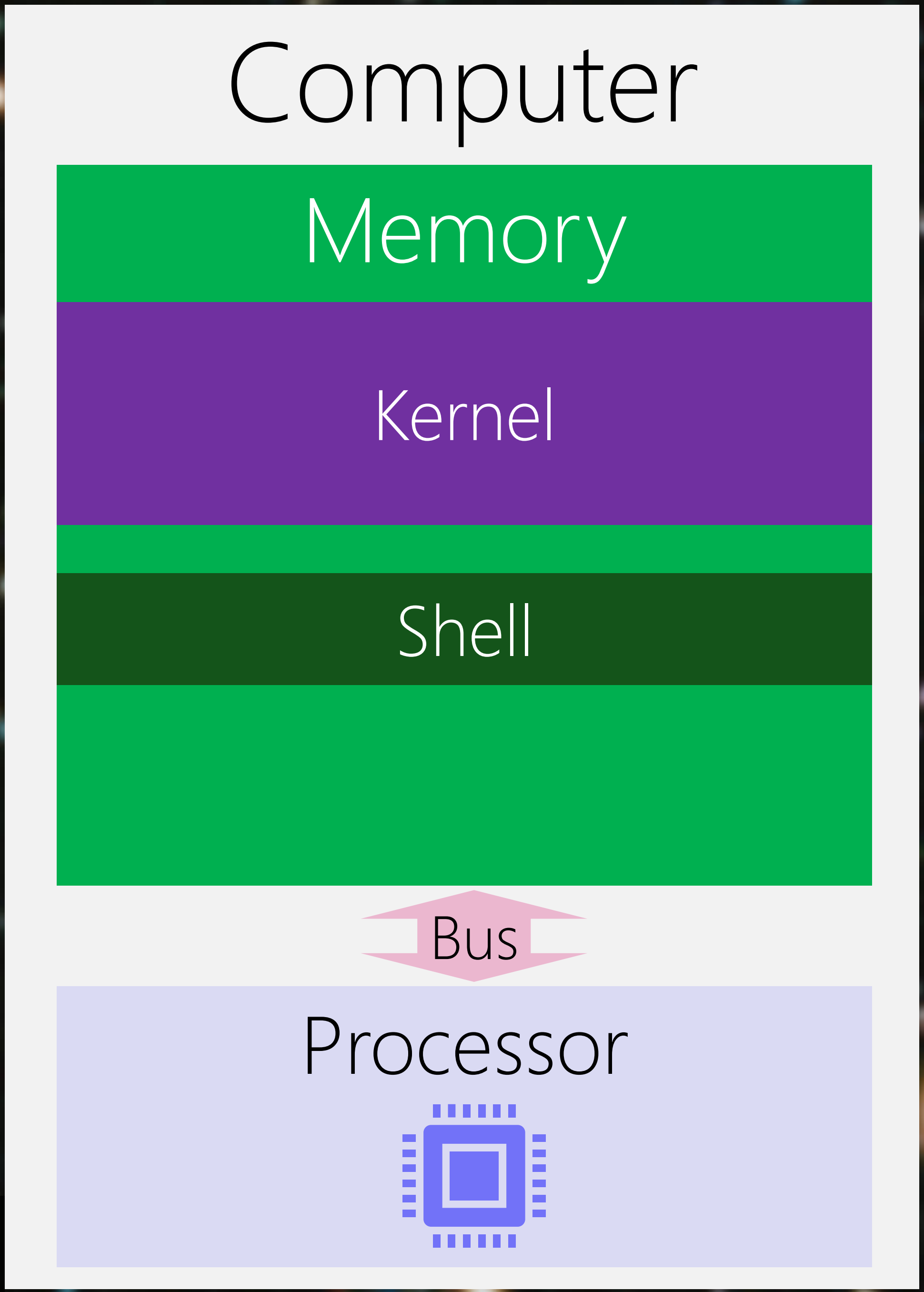
Library Routines

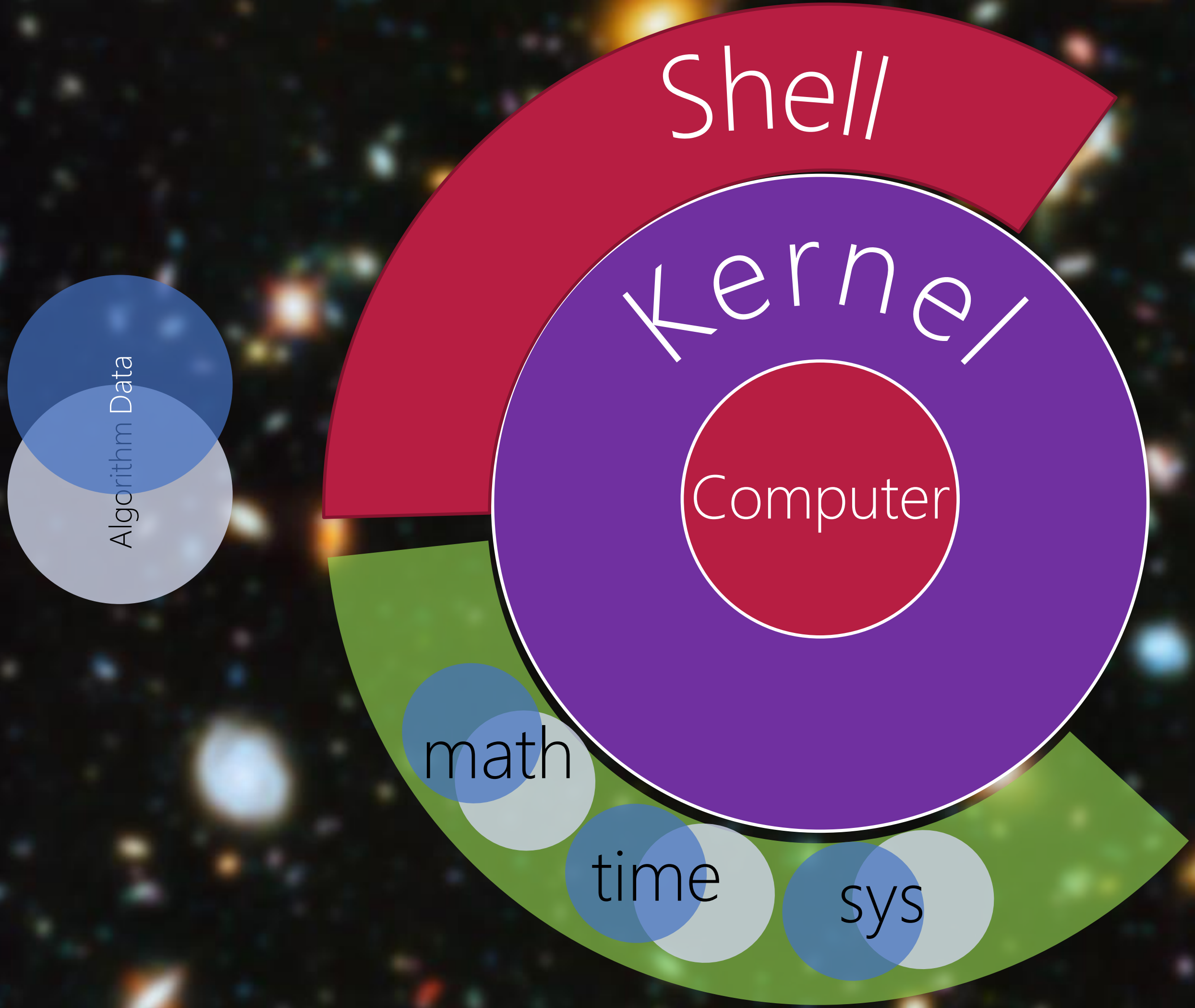
Library of common applications/functions
time, math, limit, sys, …


Data
Algorithm
Shell
Kernel
Computer
math
time
sys

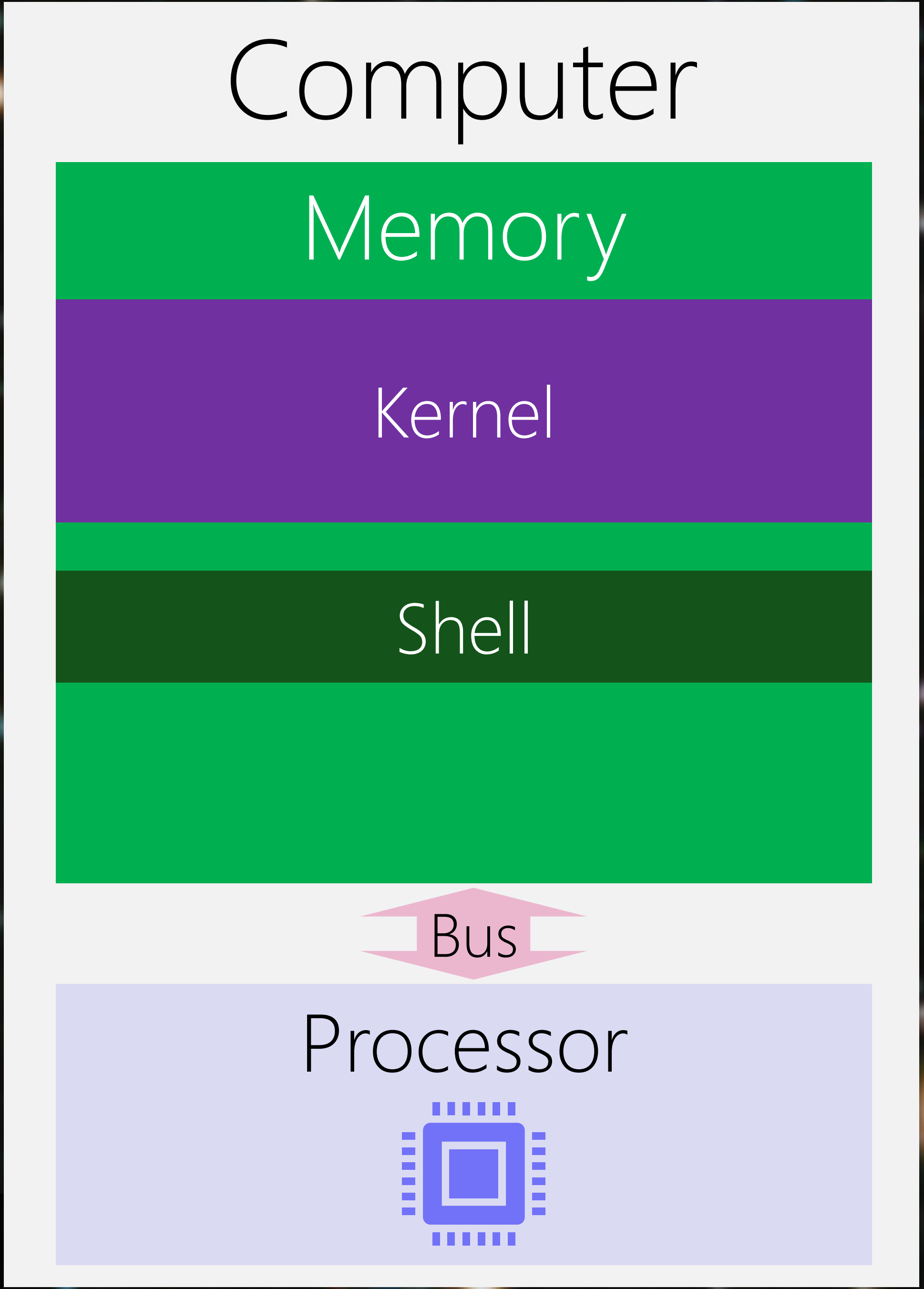


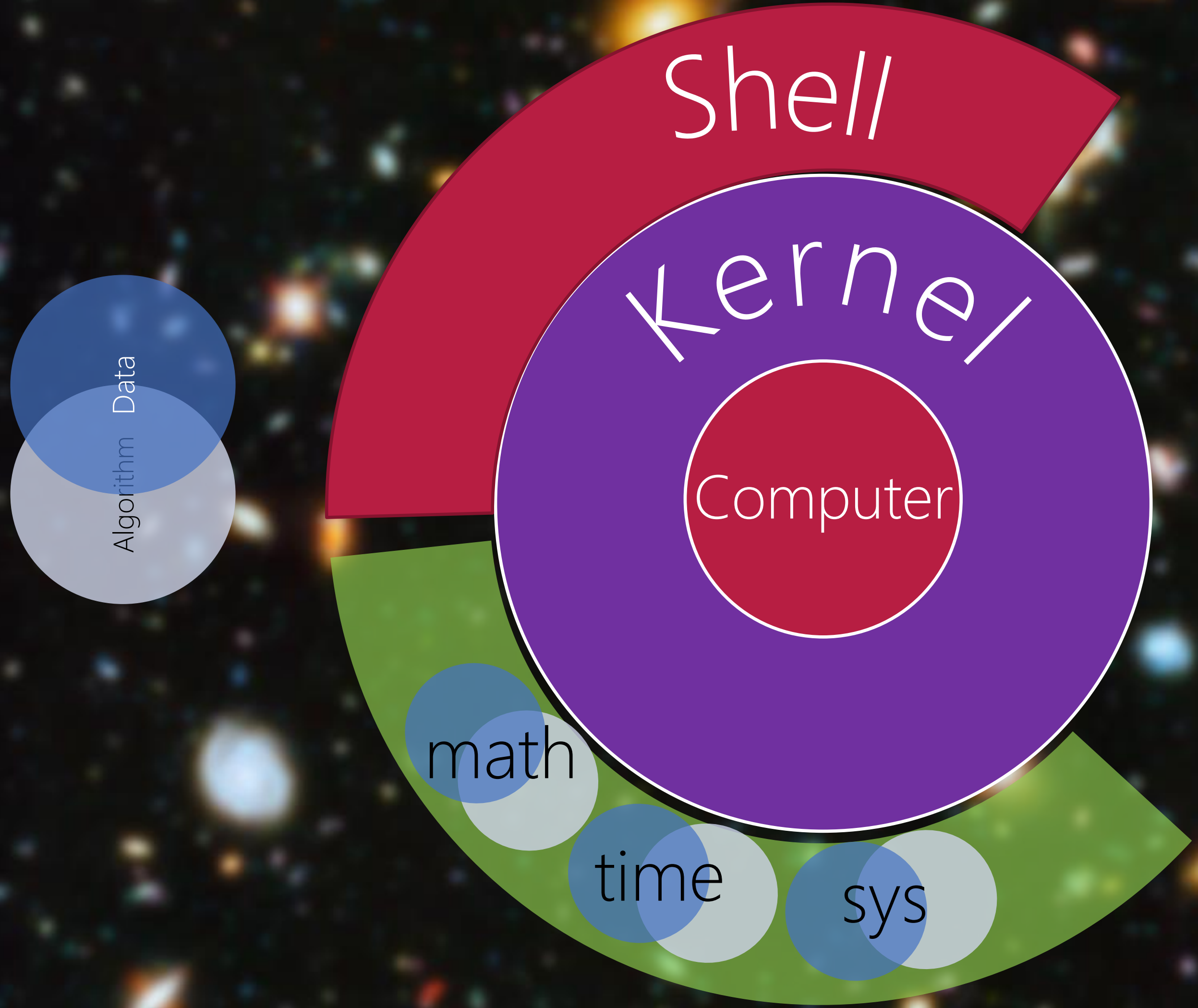
Library routines are NOT in memory!

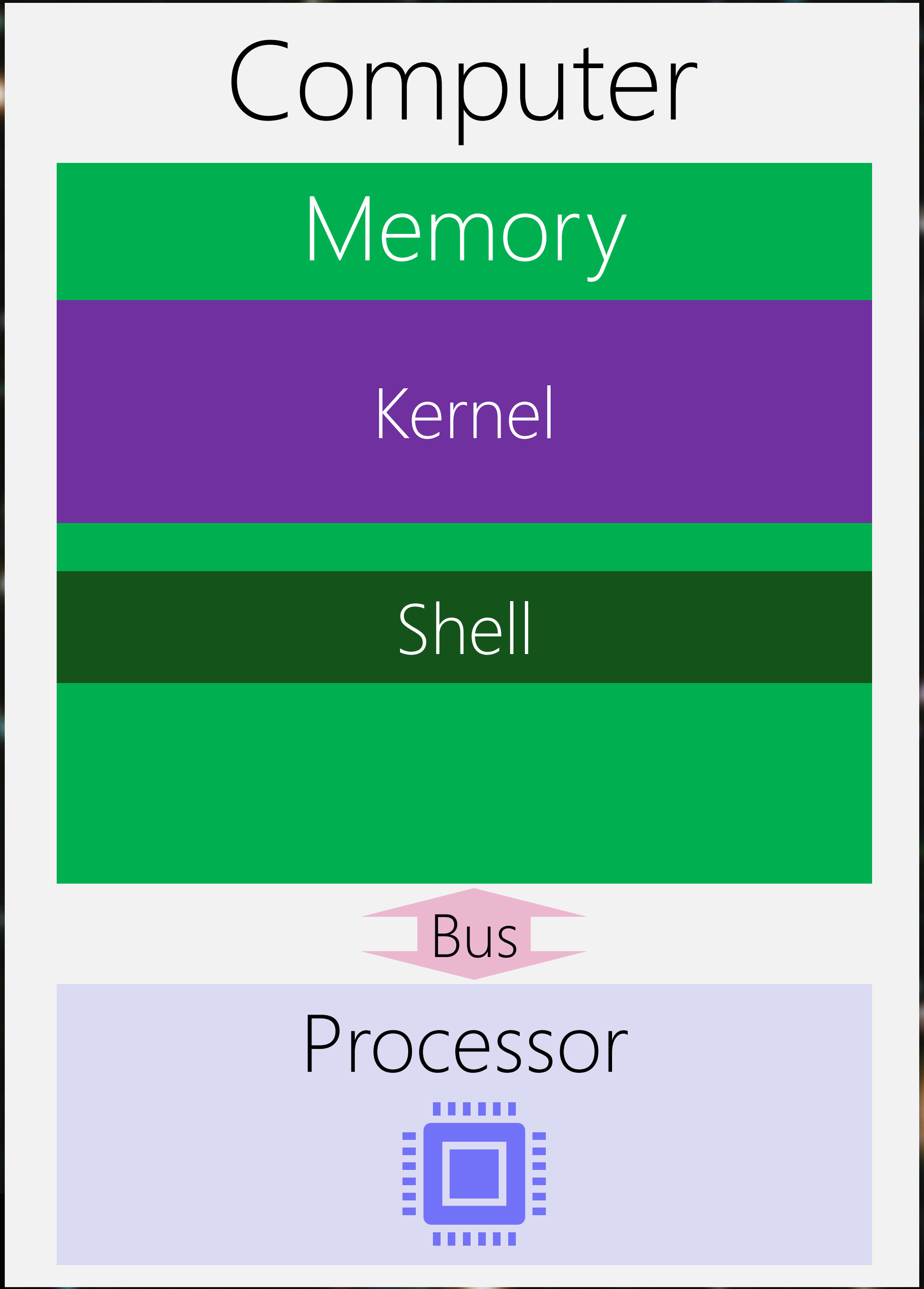




They are all in OP Code!







How to link (include vs. call) to library routines?

STATIC LINK

At compile time, using linker include!

C

Compiler

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

Assembly

Assembler

```
<printf@plt>:
jmpq        *0x3002(%rip)
pushq       $0x0
jmpq        401000 <.plt>
<main>:
push        %rbp
…
```

OP Code
stdio.h

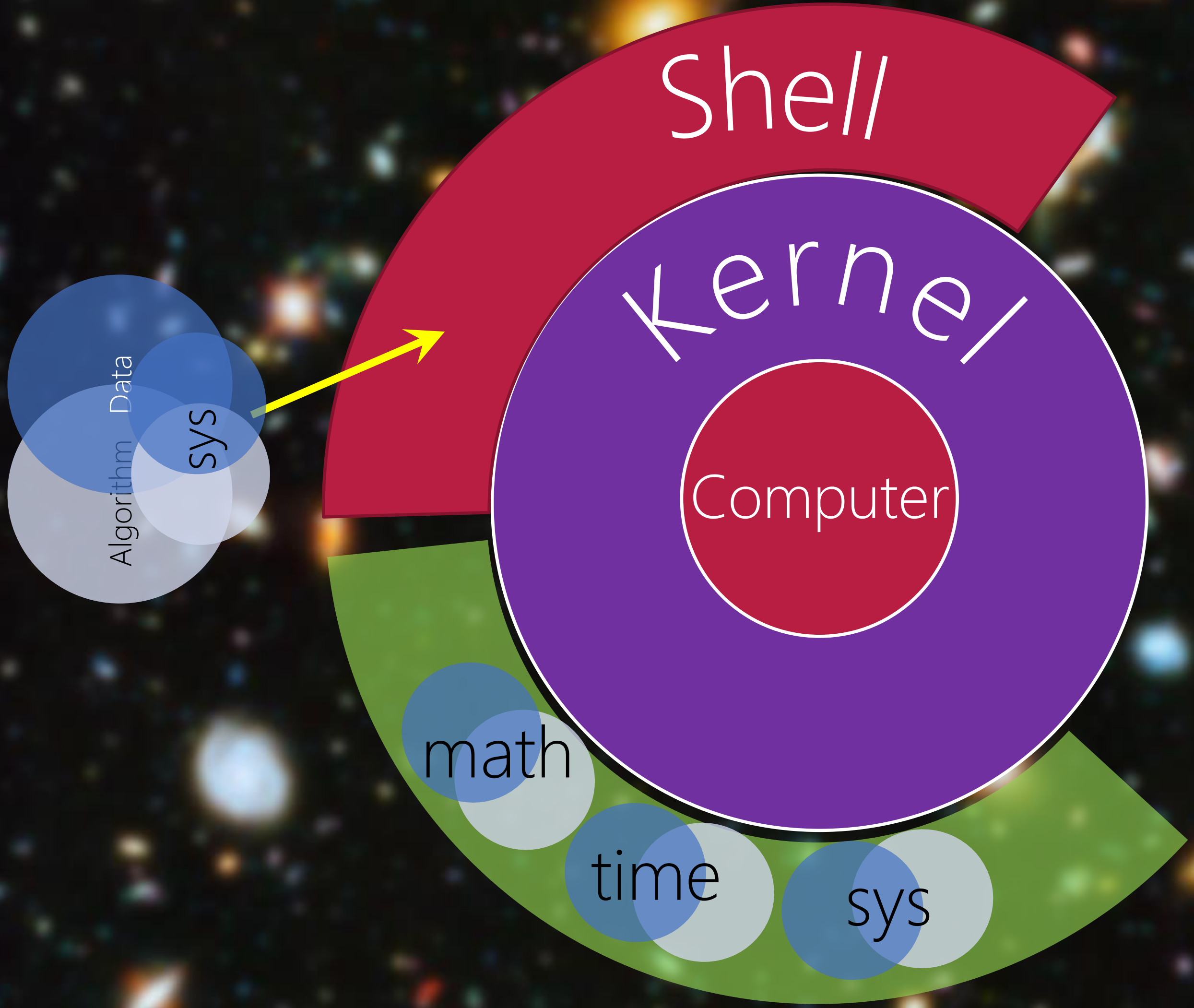
```
0001 0000 0000 0010 0000 0000 0000 1100
00f1 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0110 0111 0000
0046 0000 0220 0000 0000 0000 0000 2200
0111 0000 0000 0000 e432 0000 0000 0000
```

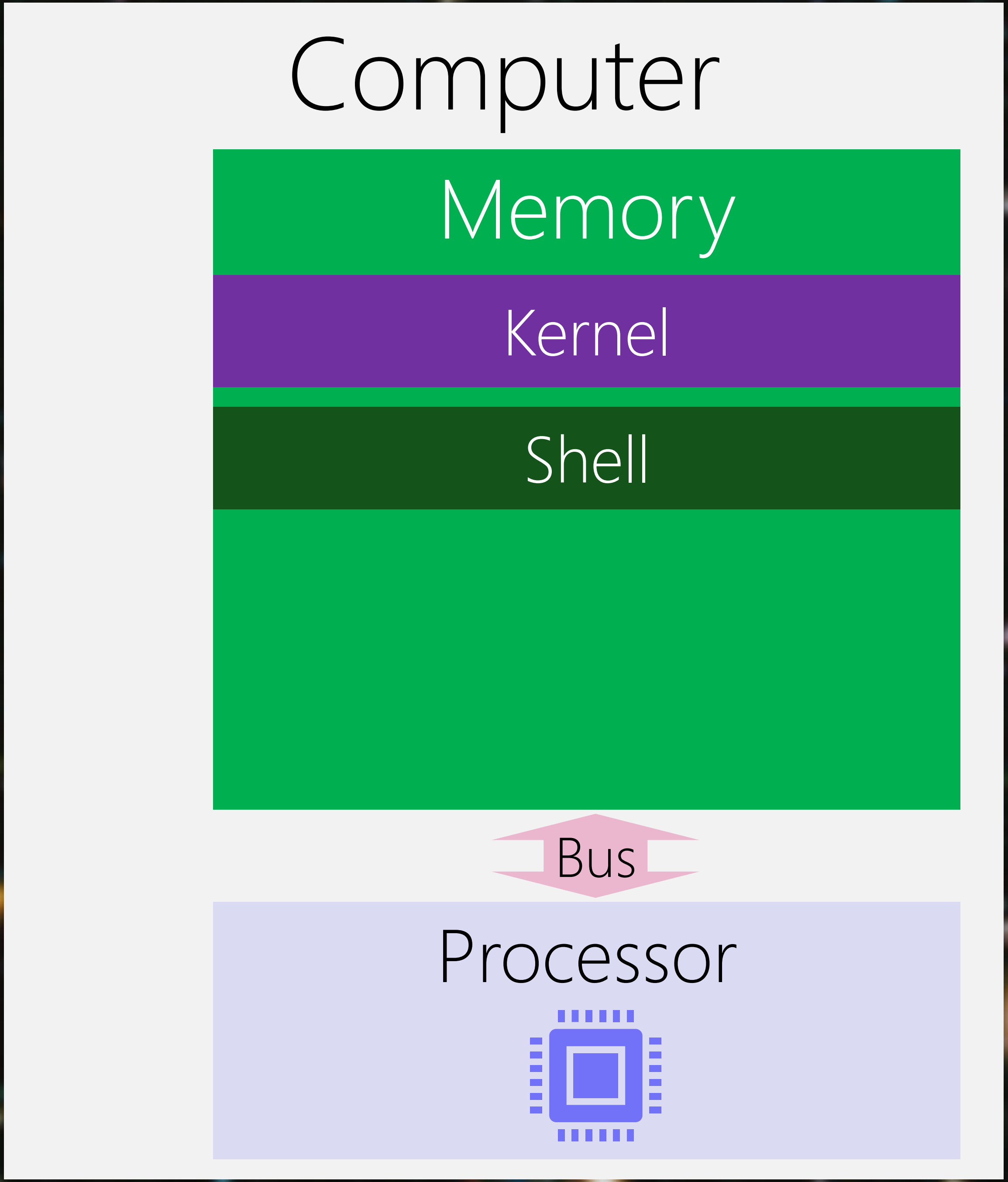
OP Code

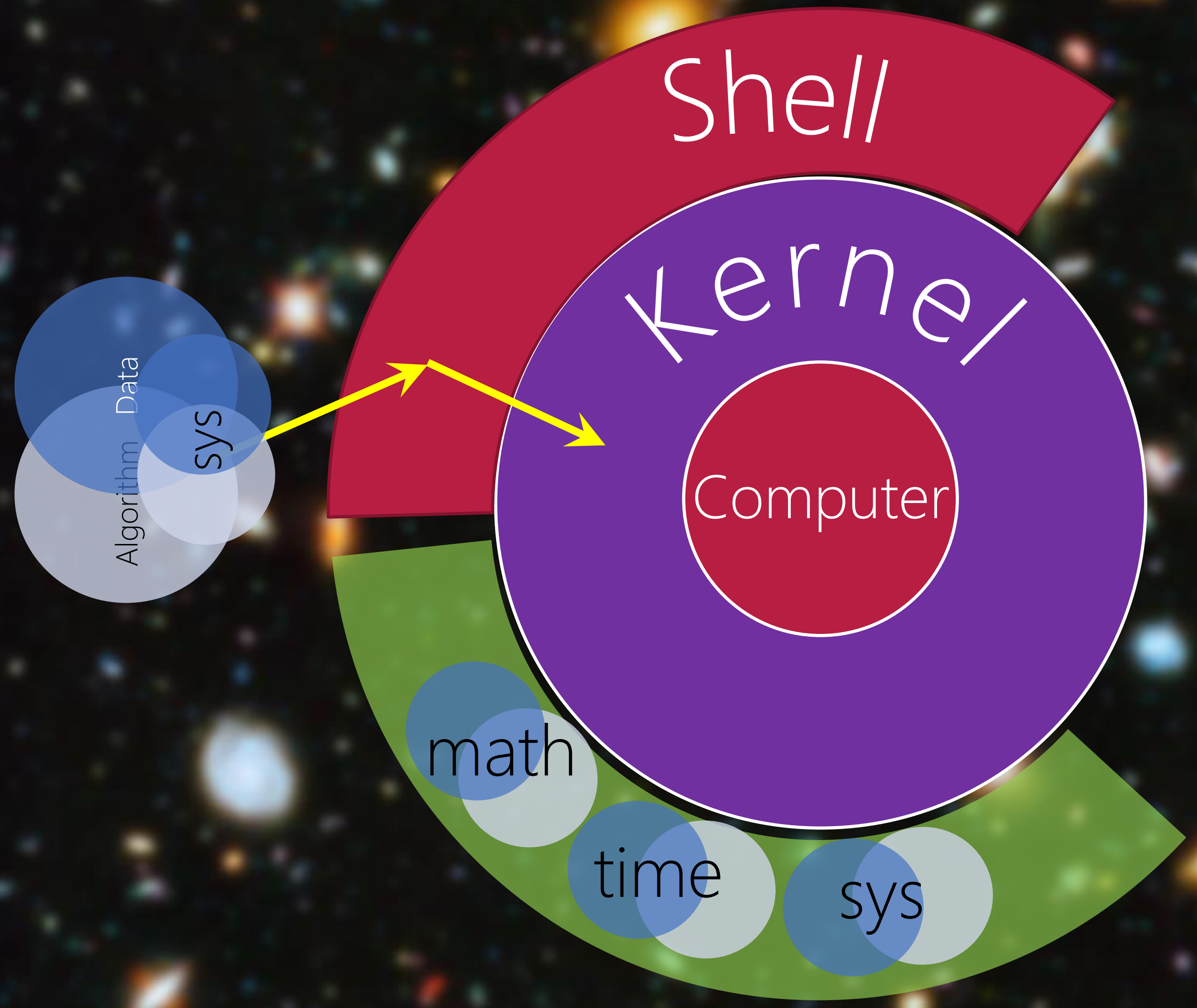
```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
0001 0000 0000 0000 0000 0000 0000 0000
```

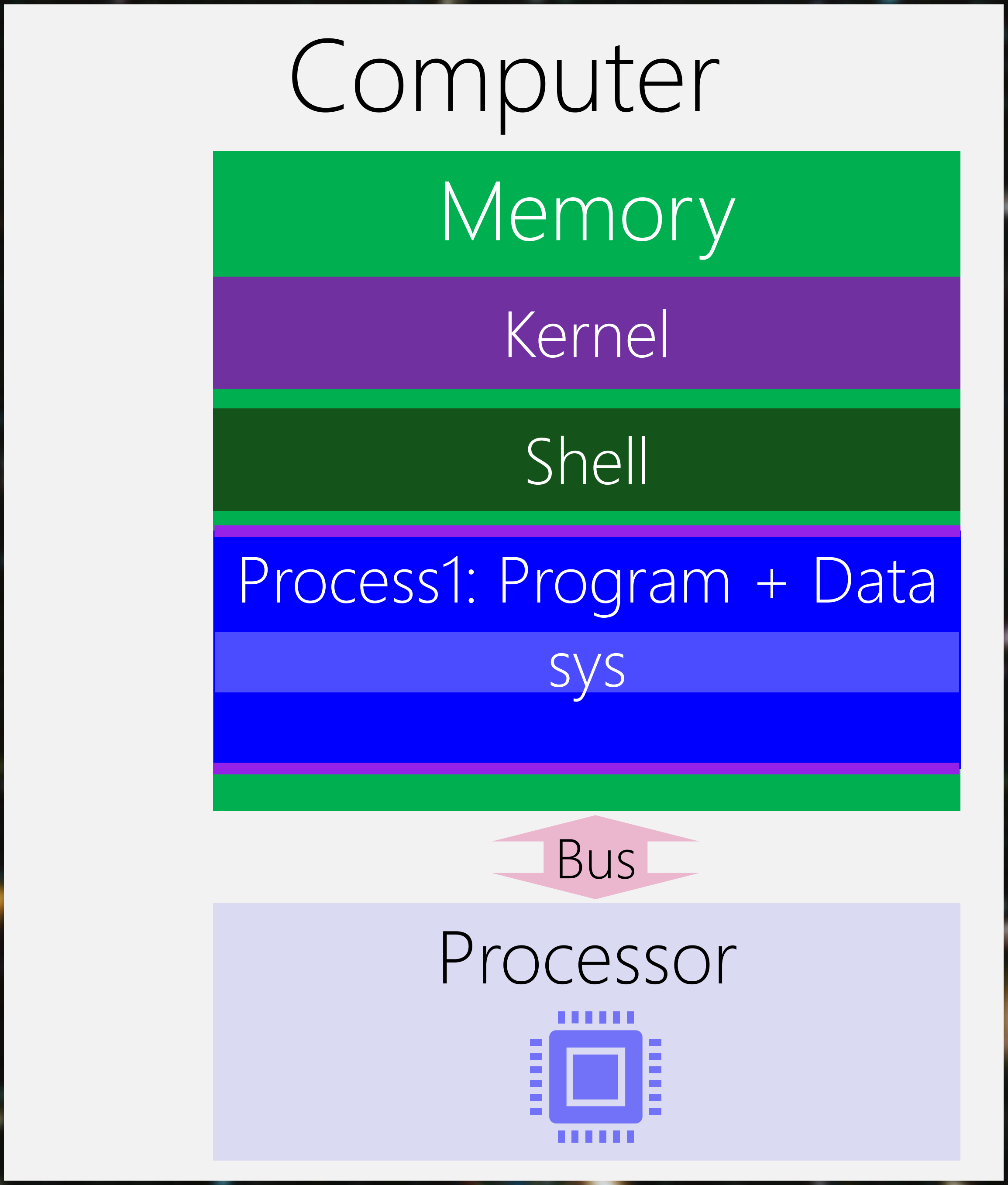
Linker

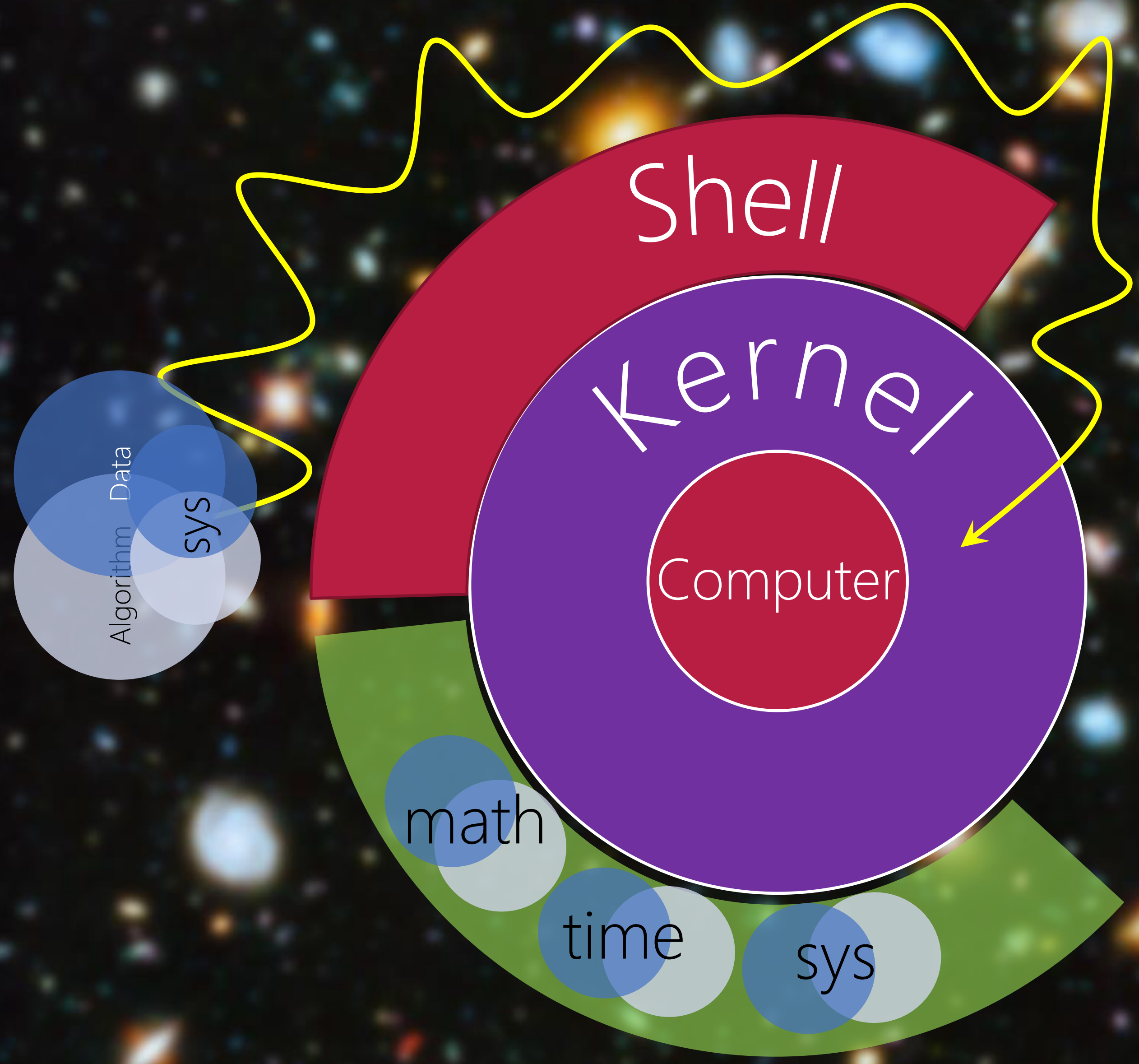
```
0001 0000 0000 0010 0000 0000 0000 1100
00f1 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0110 0111 0000
0046 0000 0220 0000 0000 0000 0000 2200
0111 0000 0000 0000 e432 0000 0000 0000
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
0001 0000 0000 0000 0000 0000 0000 0000
```

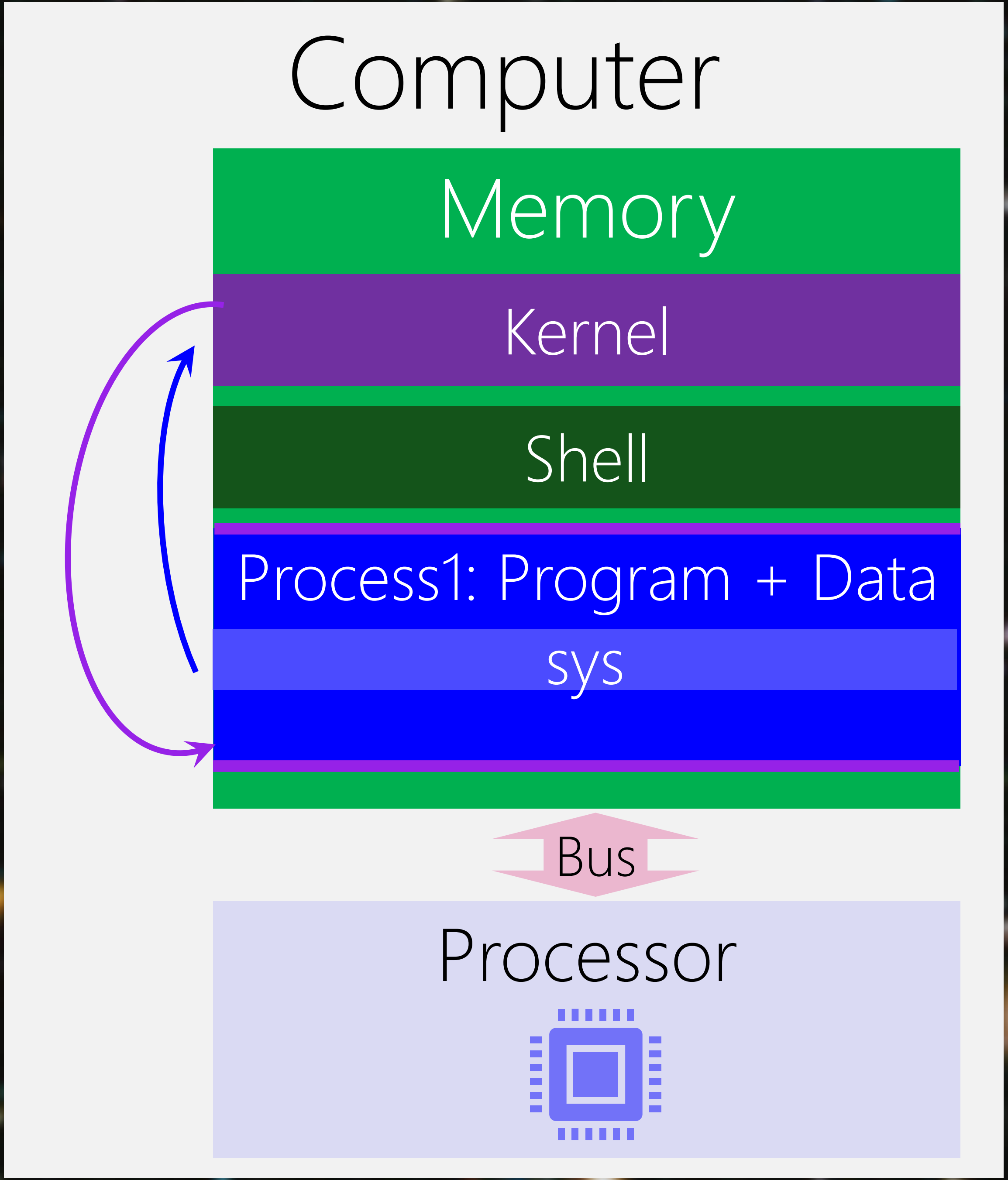

Shell
Kernel
Computer
Data
sys
Algorithm
math
time
sys


Computer
Memory
Kernel
Shell
Bus
Processor


Shell
Kernel
Computer
Data
Algorithm
sys
math
time
sys


Computer
Memory
Kernel
Shell
Process1: Program + Data
sys
Bus
Processor


Shell
Kernel
Computer
Data
Algorithm
sys
math
time
sys

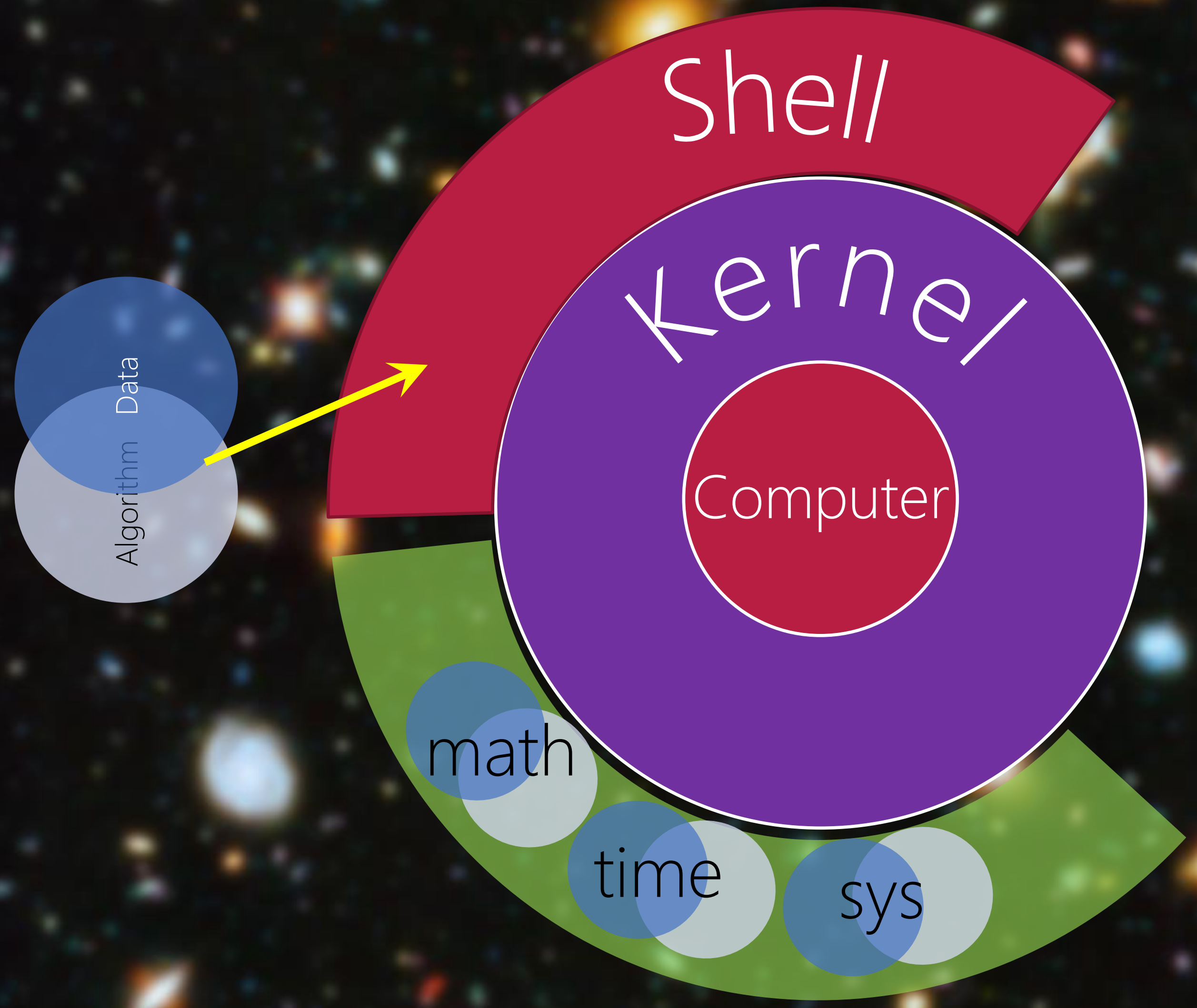

Computer
Memory
Kernel
Shell
Process1: Program + Data
sys
Bus
Processor

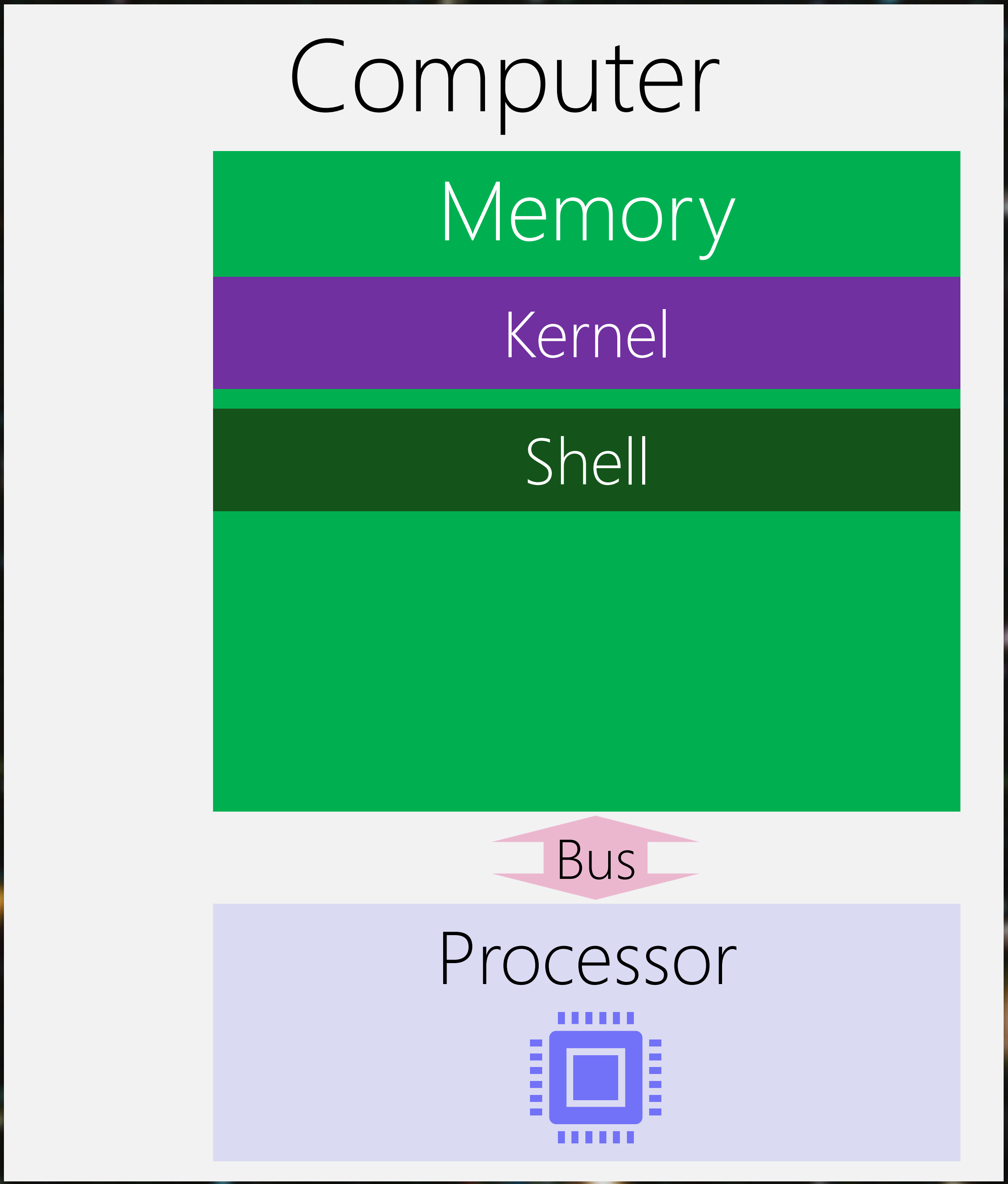
STATIC LINK

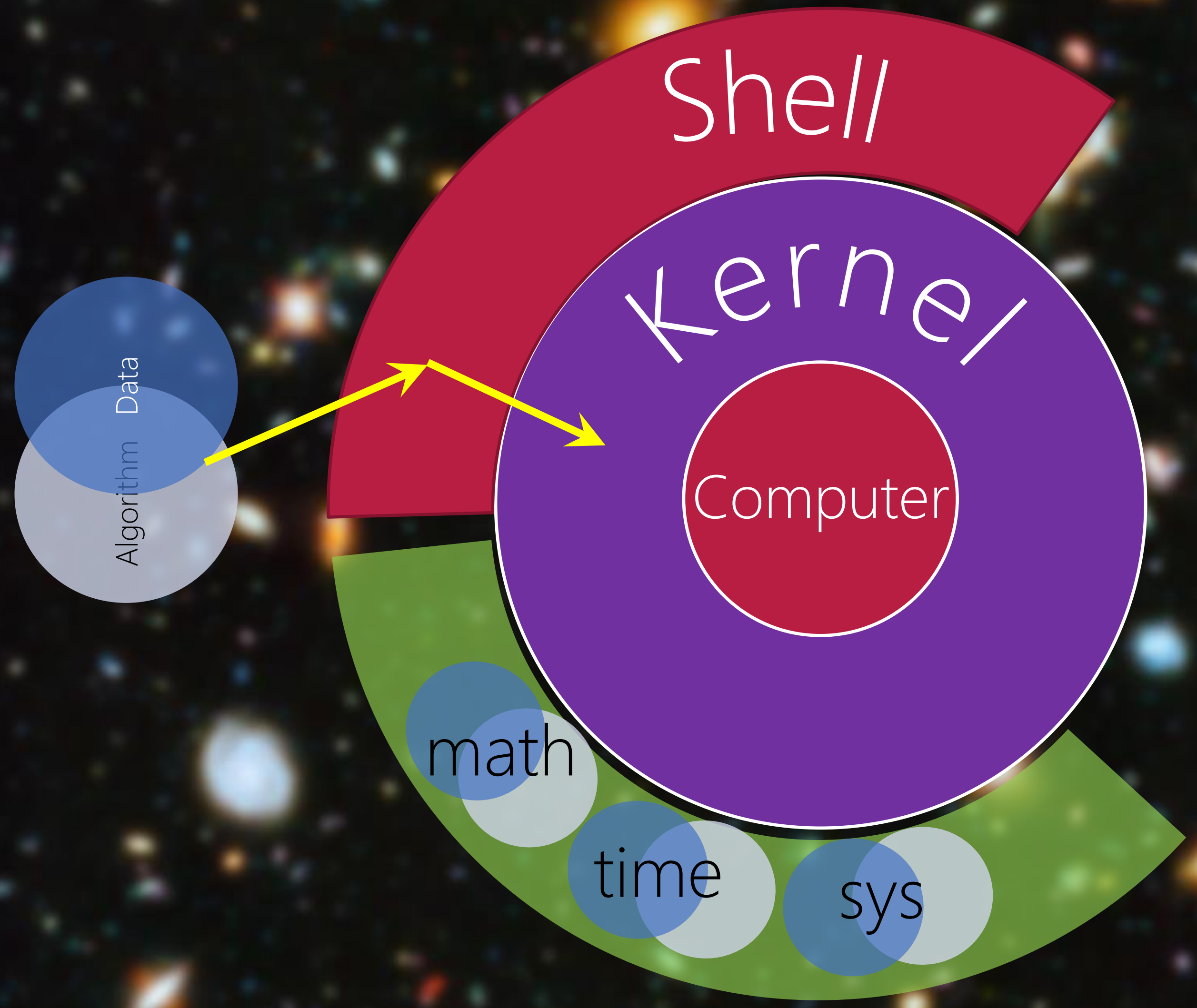
From Kernel's POV, everything is the same!

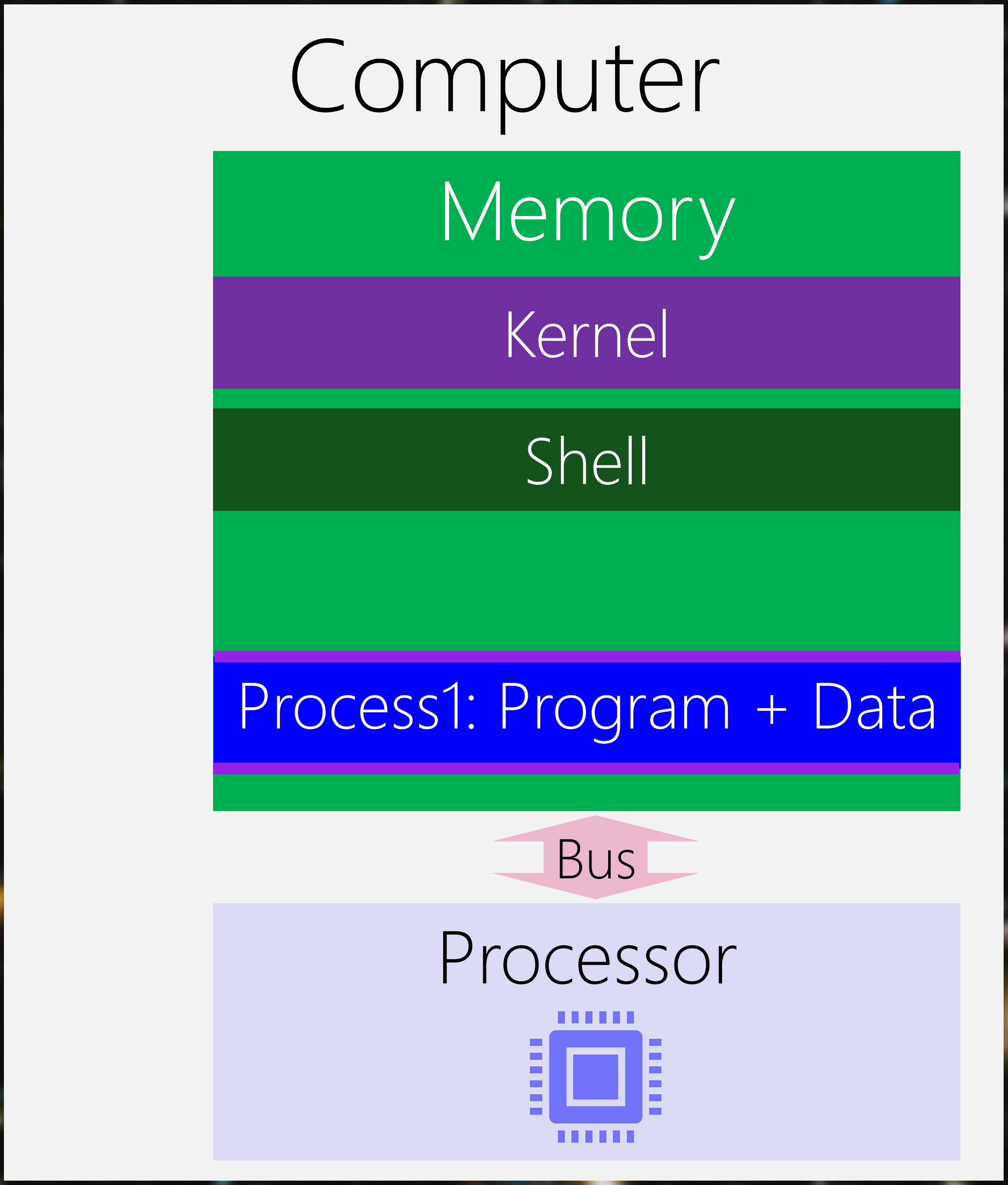
DYNAMIC LINK

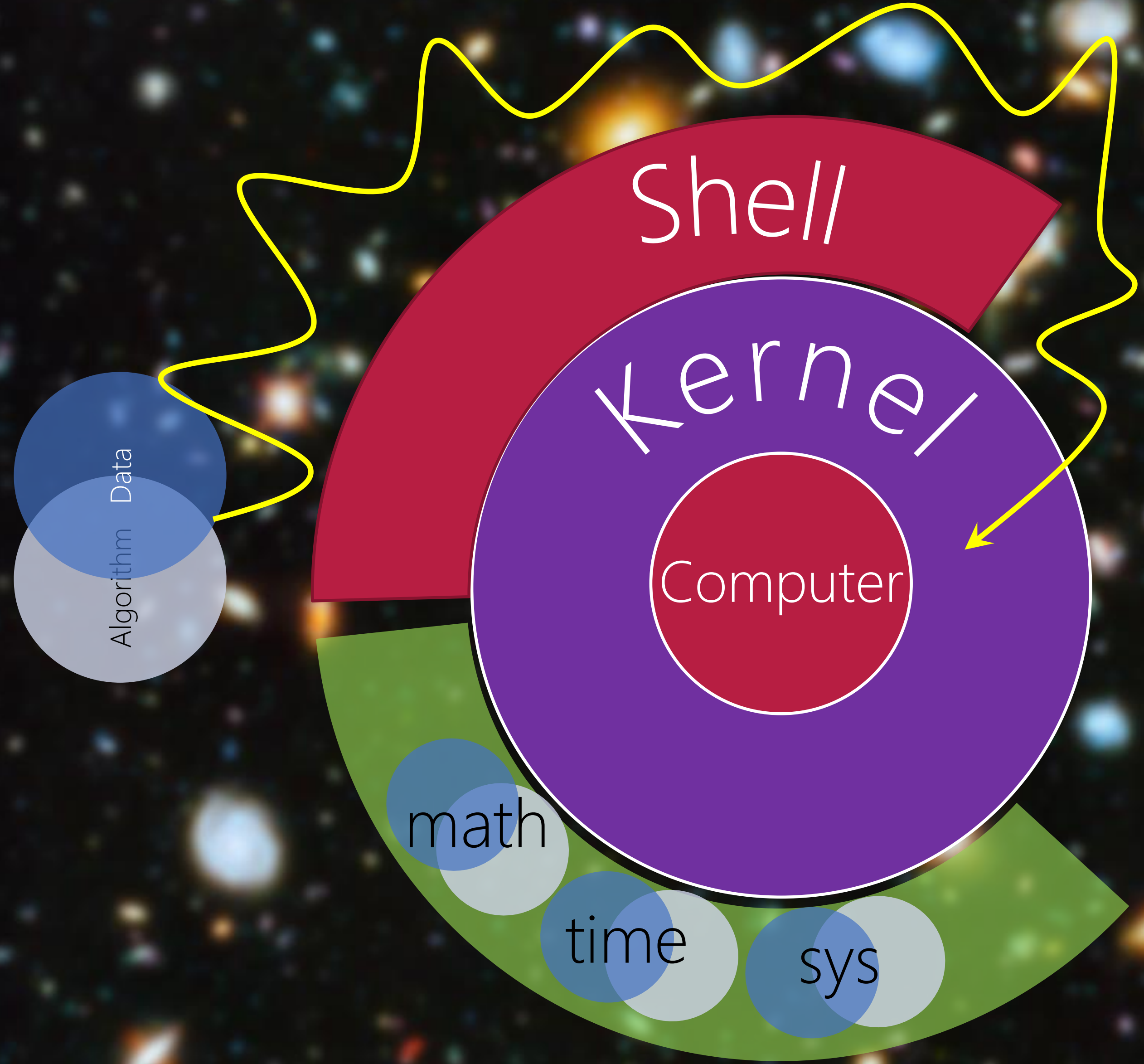
At run time, using kernel call!

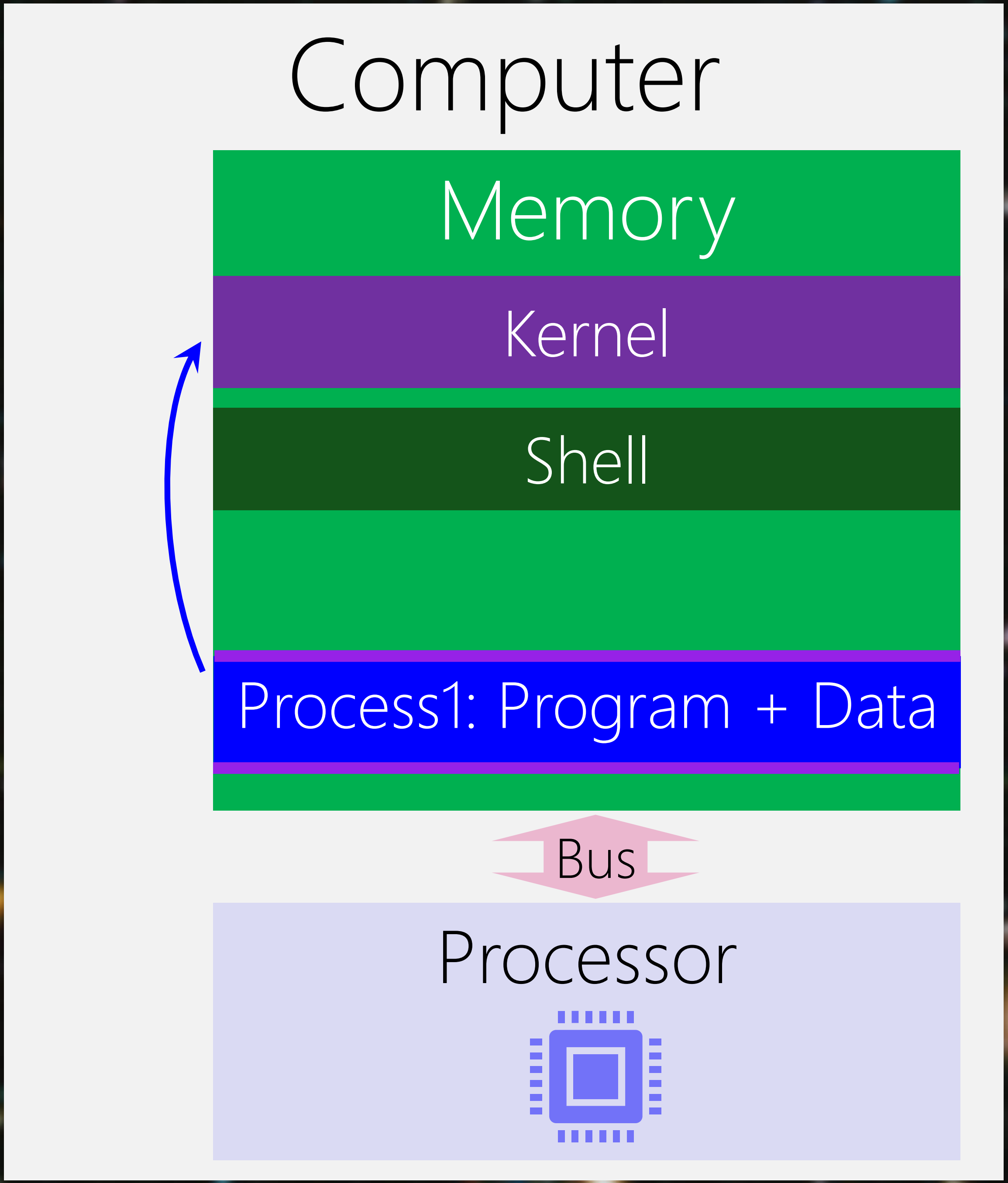

Shell
Kernel
Computer
Data
Algorithm
math
time
sys

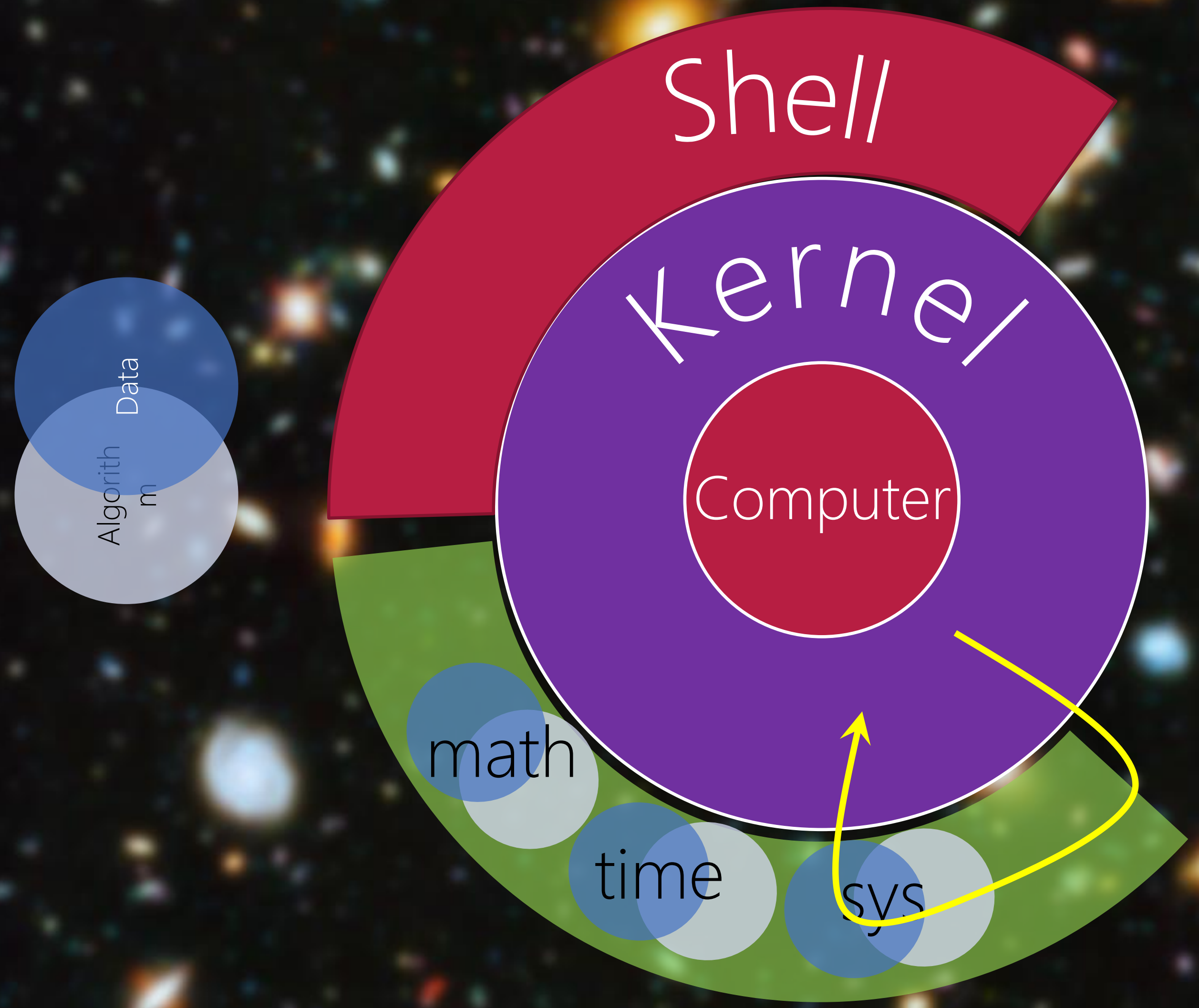

Computer
Memory
Kernel
Shell
Bus
Processor

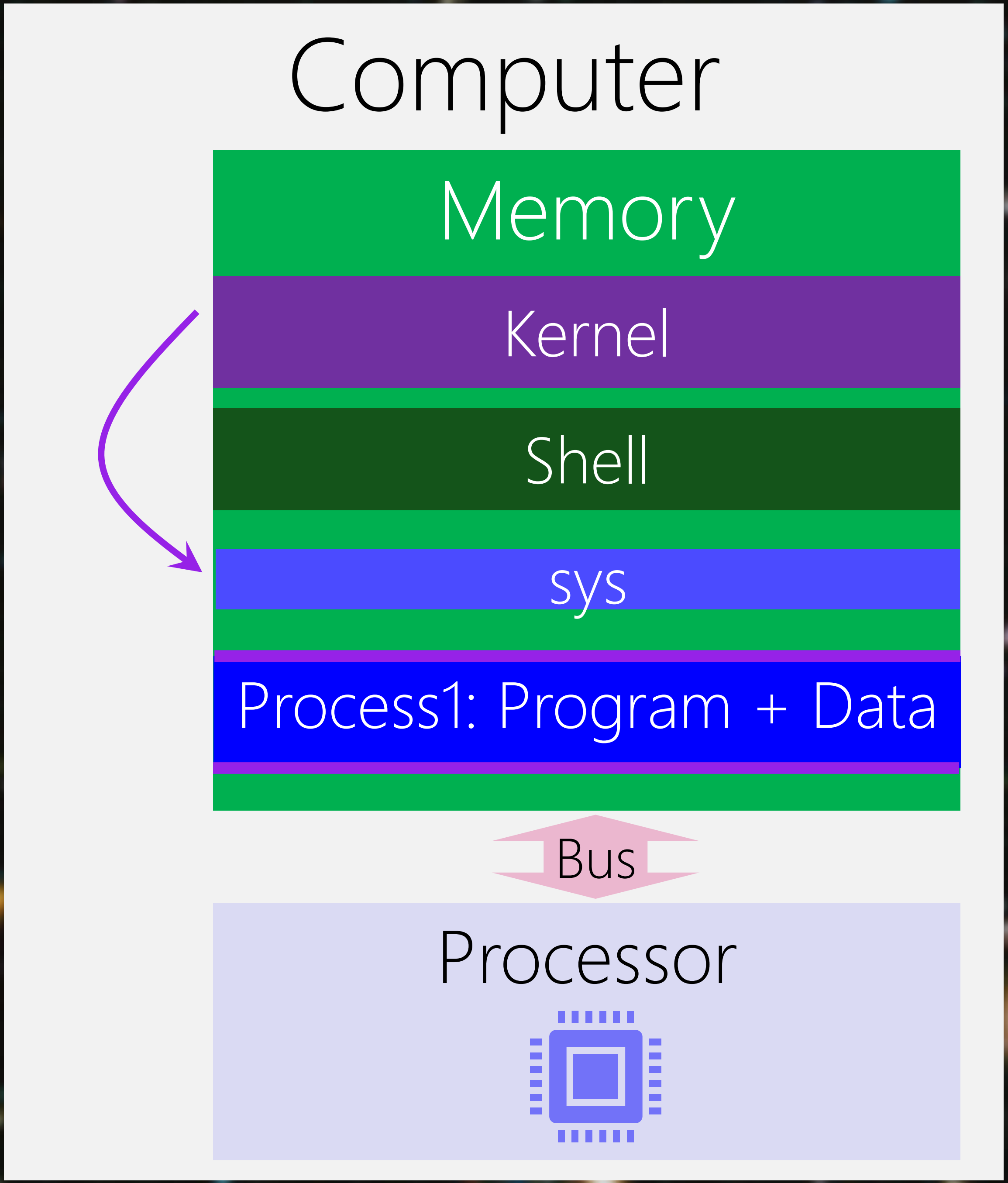

Shell
Kernel
Computer
Data
Algorithm
math
time
sys

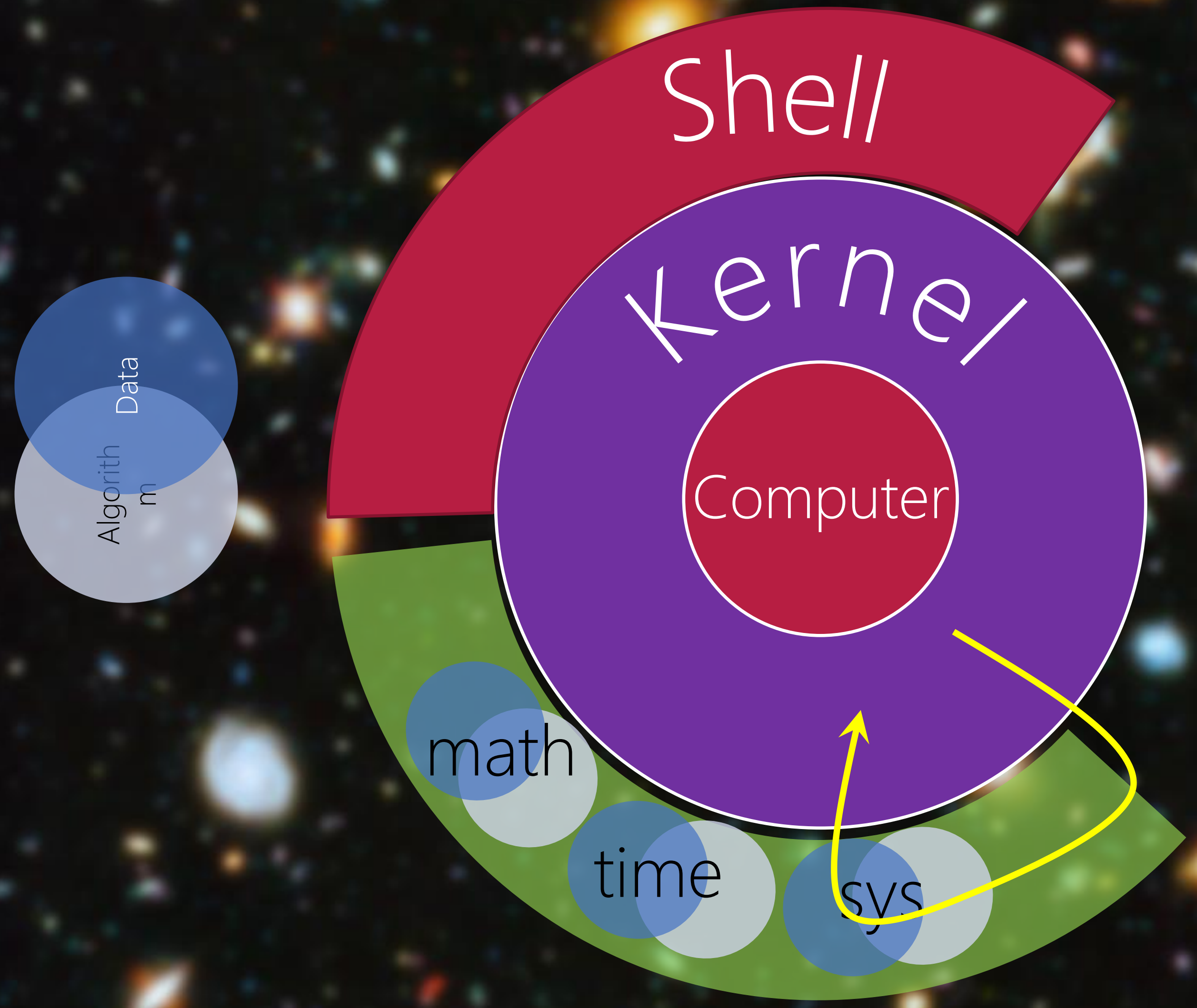

Computer
Memory
Kernel
Shell
Process1: Program + Data
Bus
Processor

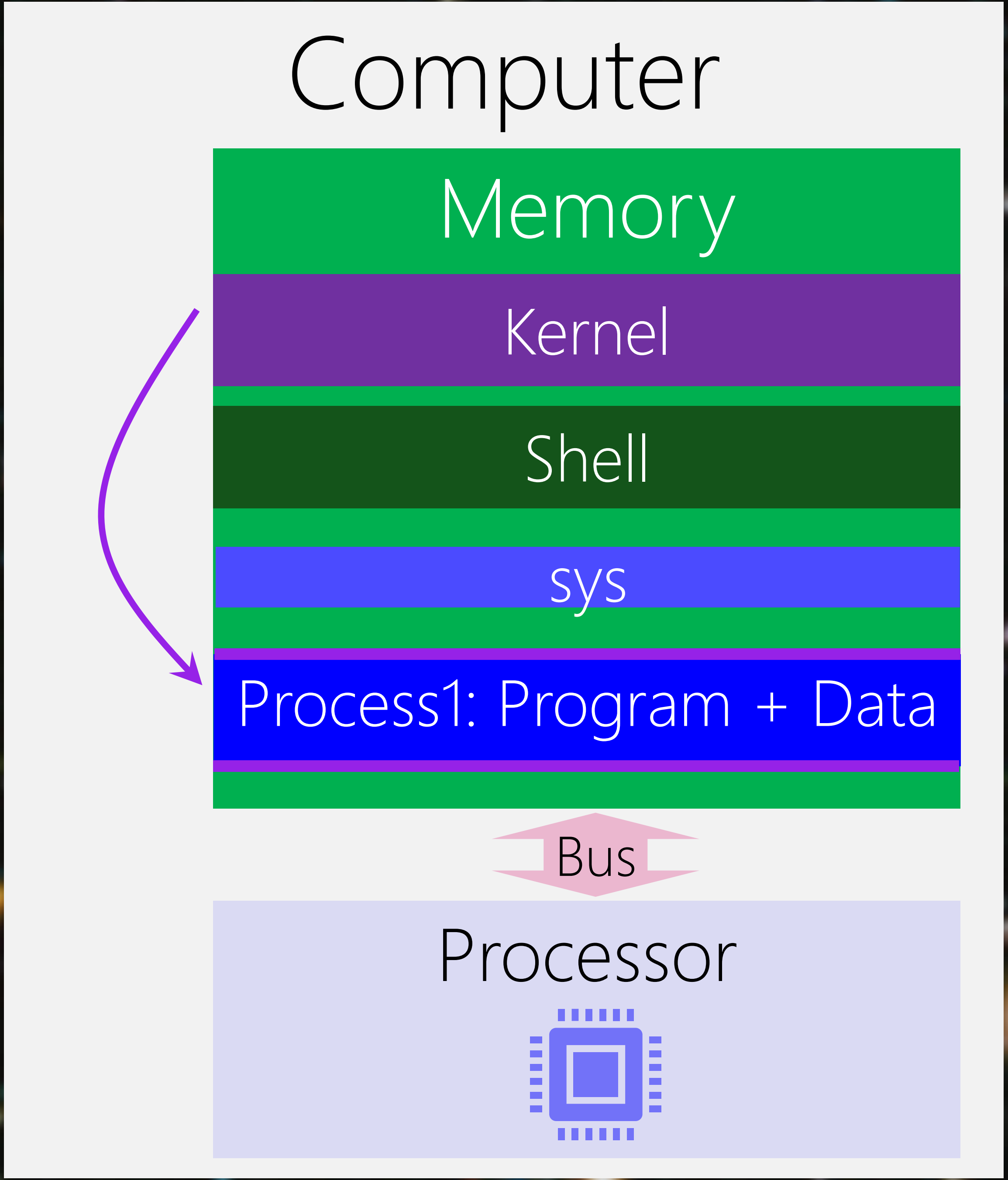

Shell
Kernel
Computer
Data
Algorithm
math
time
sys

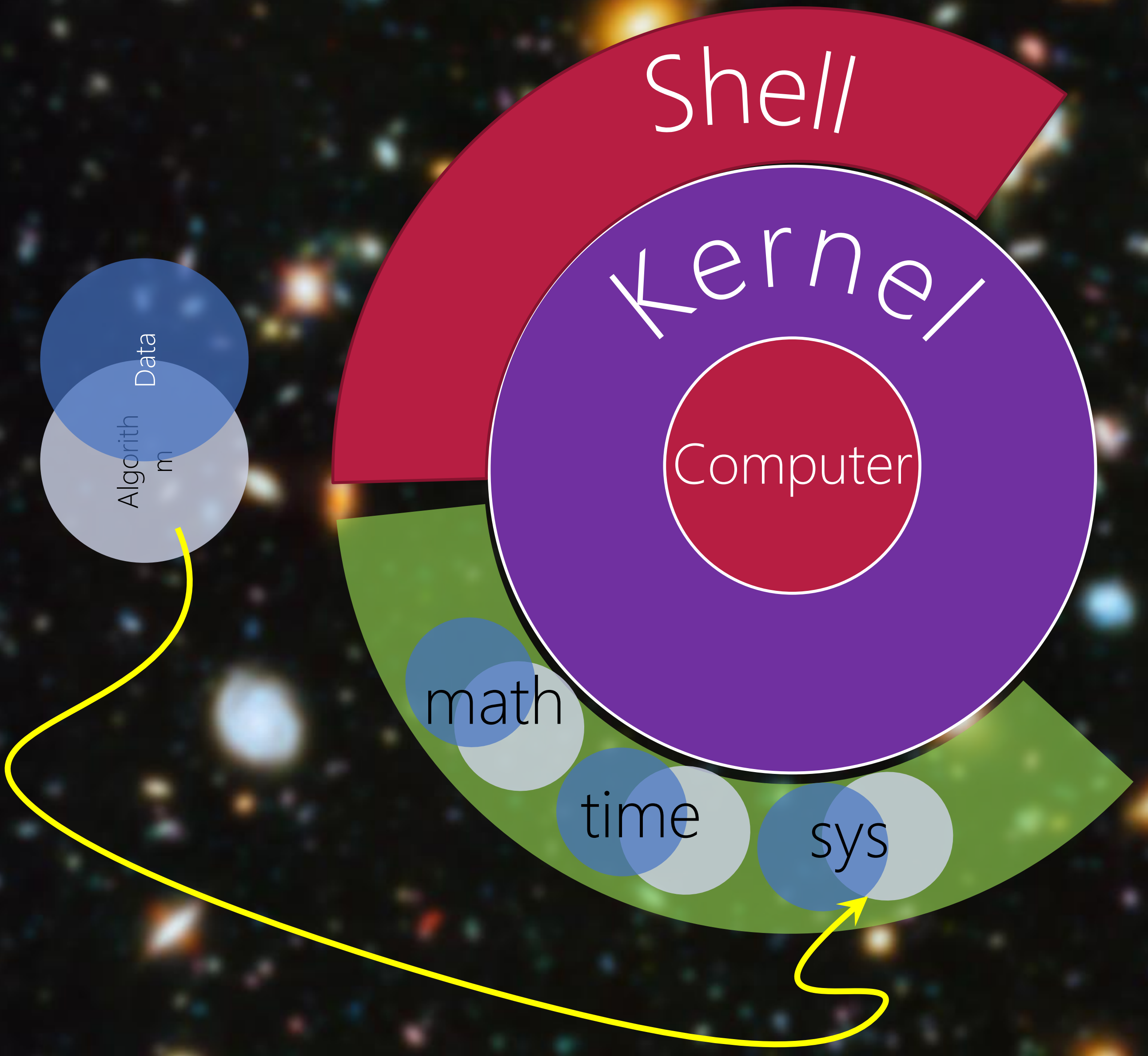

Computer
Memory
Kernel
Shell
Process1: Program + Data
Bus
Processor

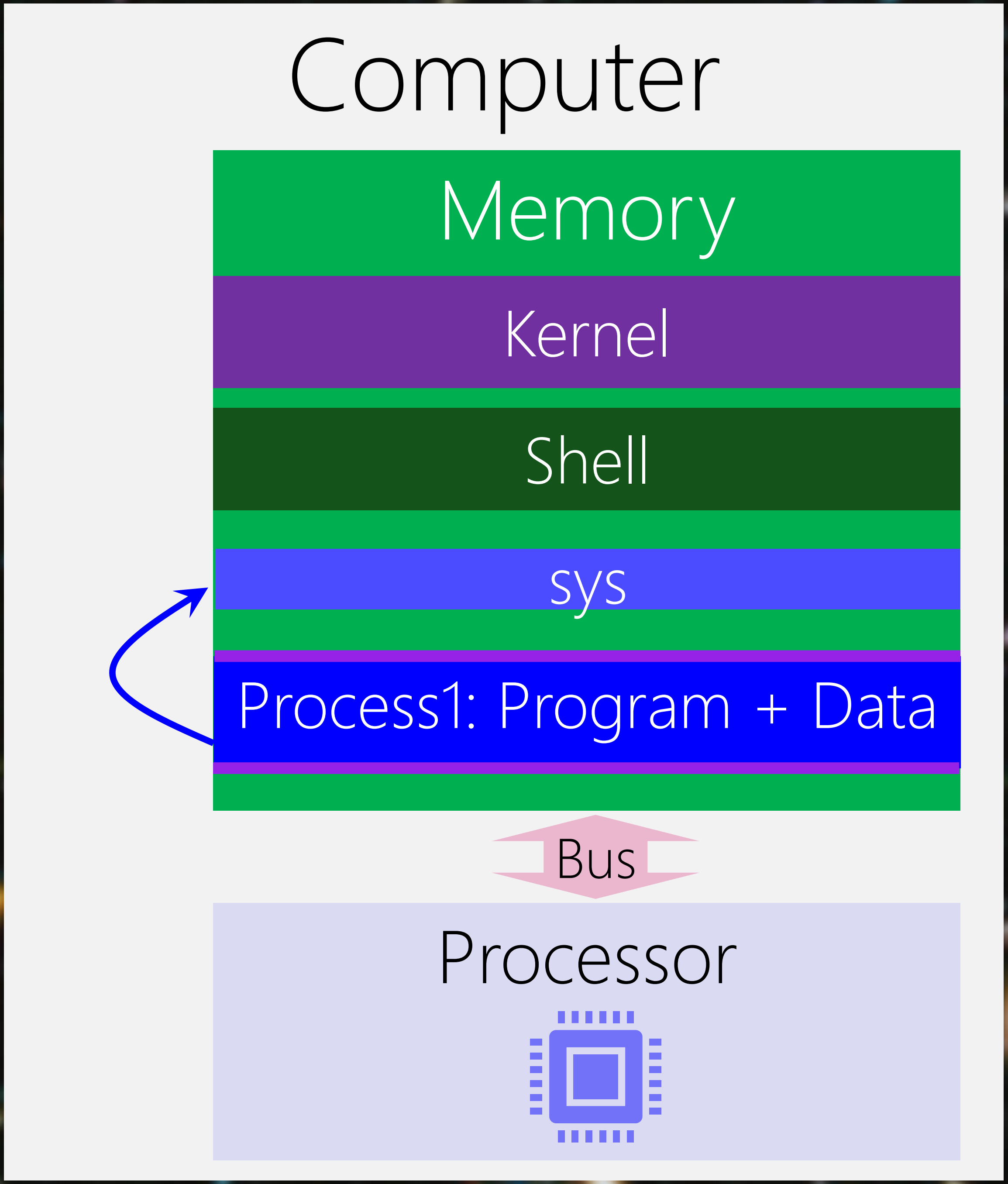

Data
Algorithm
Shell
Kernel
Computer
math
time
sys

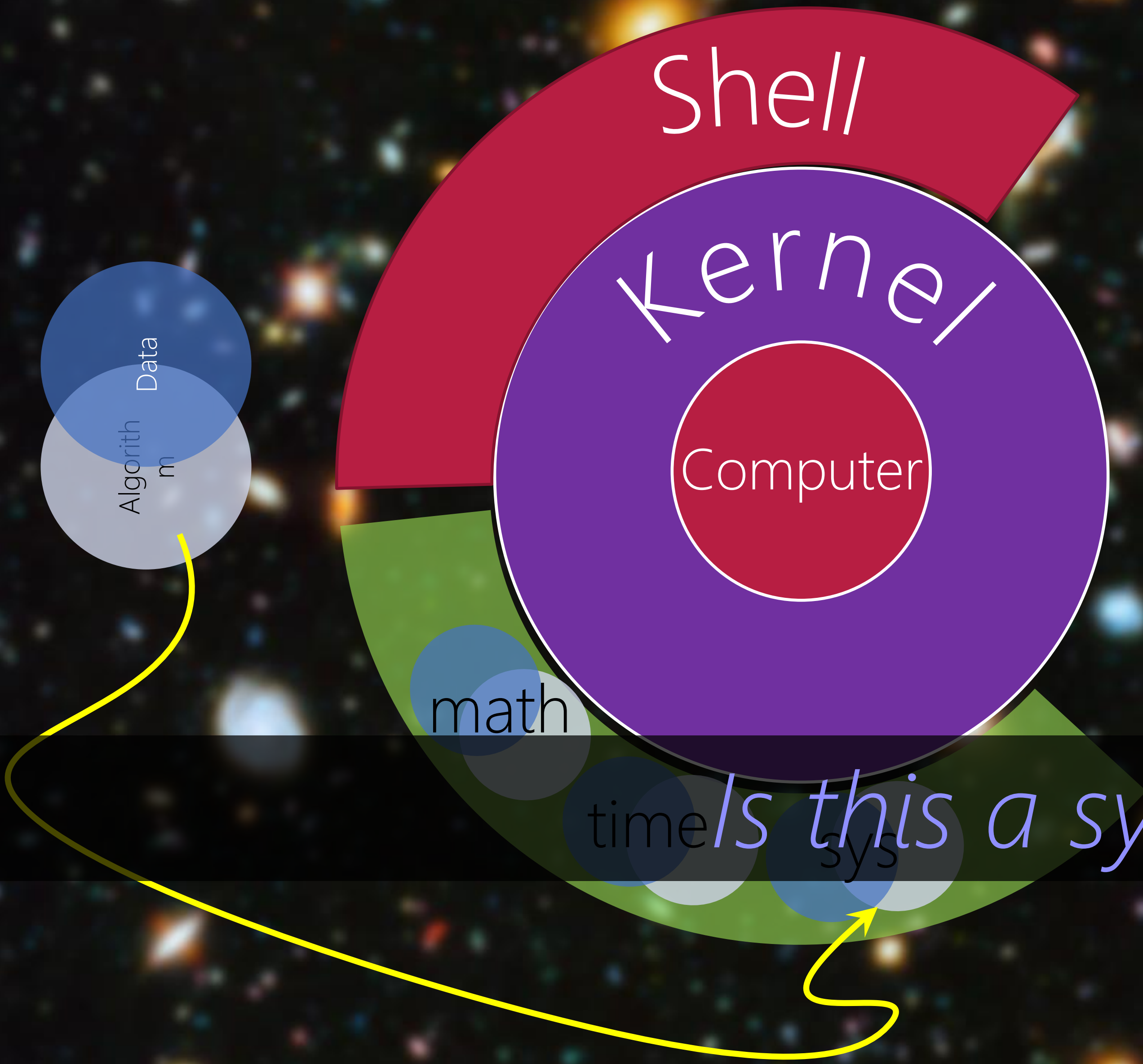

Computer
Memory
Kernel
Shell
sys
Process1: Program + Data
Bus
Processor

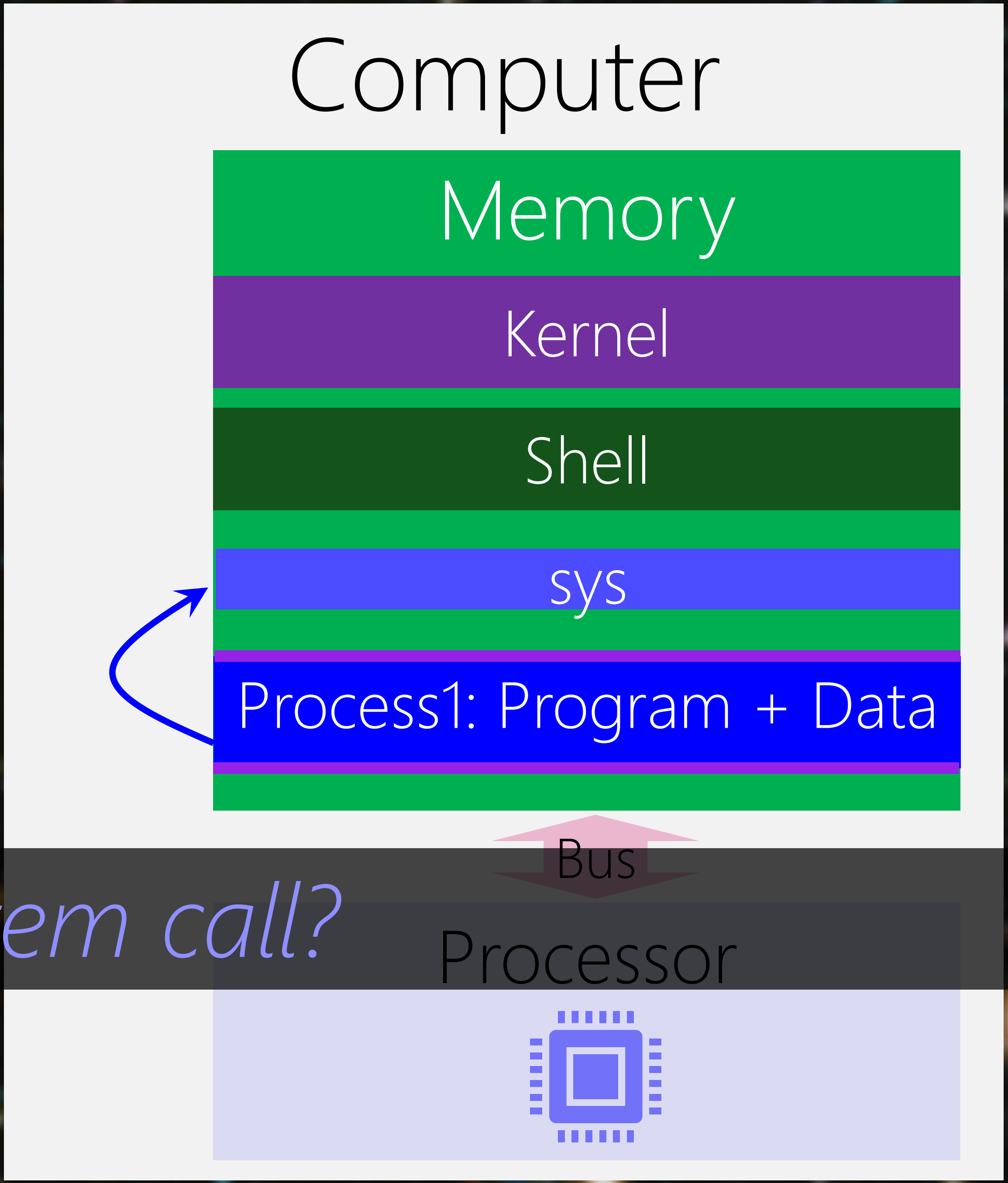

Data
Algorithm
Shell
Kernel
Computer
math
time
sys


Computer
Memory
Kernel
Shell
sys
Process1: Program + Data
Bus
Processor

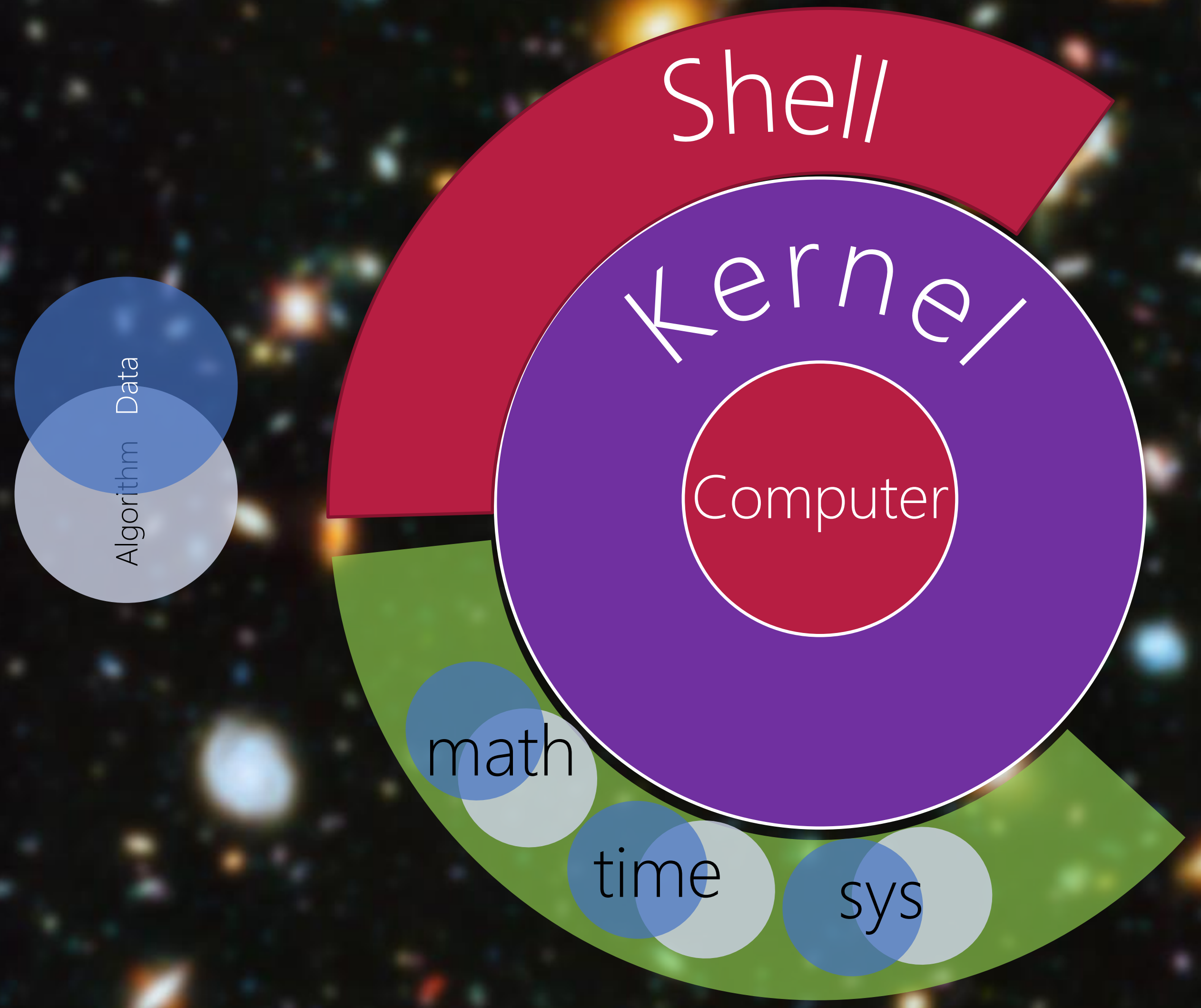

Shell
Kernel
Computer
Data
Algorithm
math
time
sys

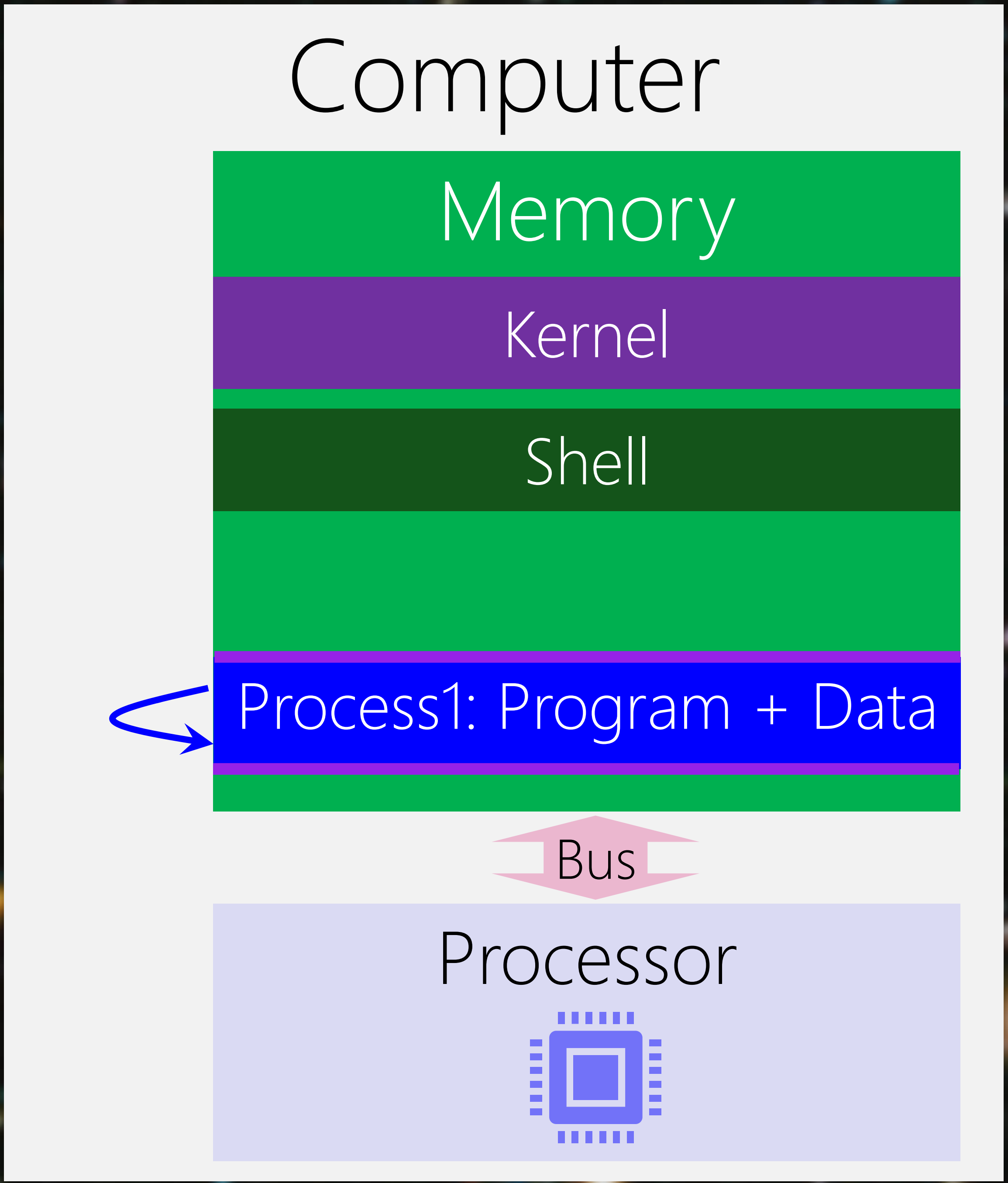

Computer
Memory
Kernel
Shell
sys
Process1: Program + Data
Bus
Processor

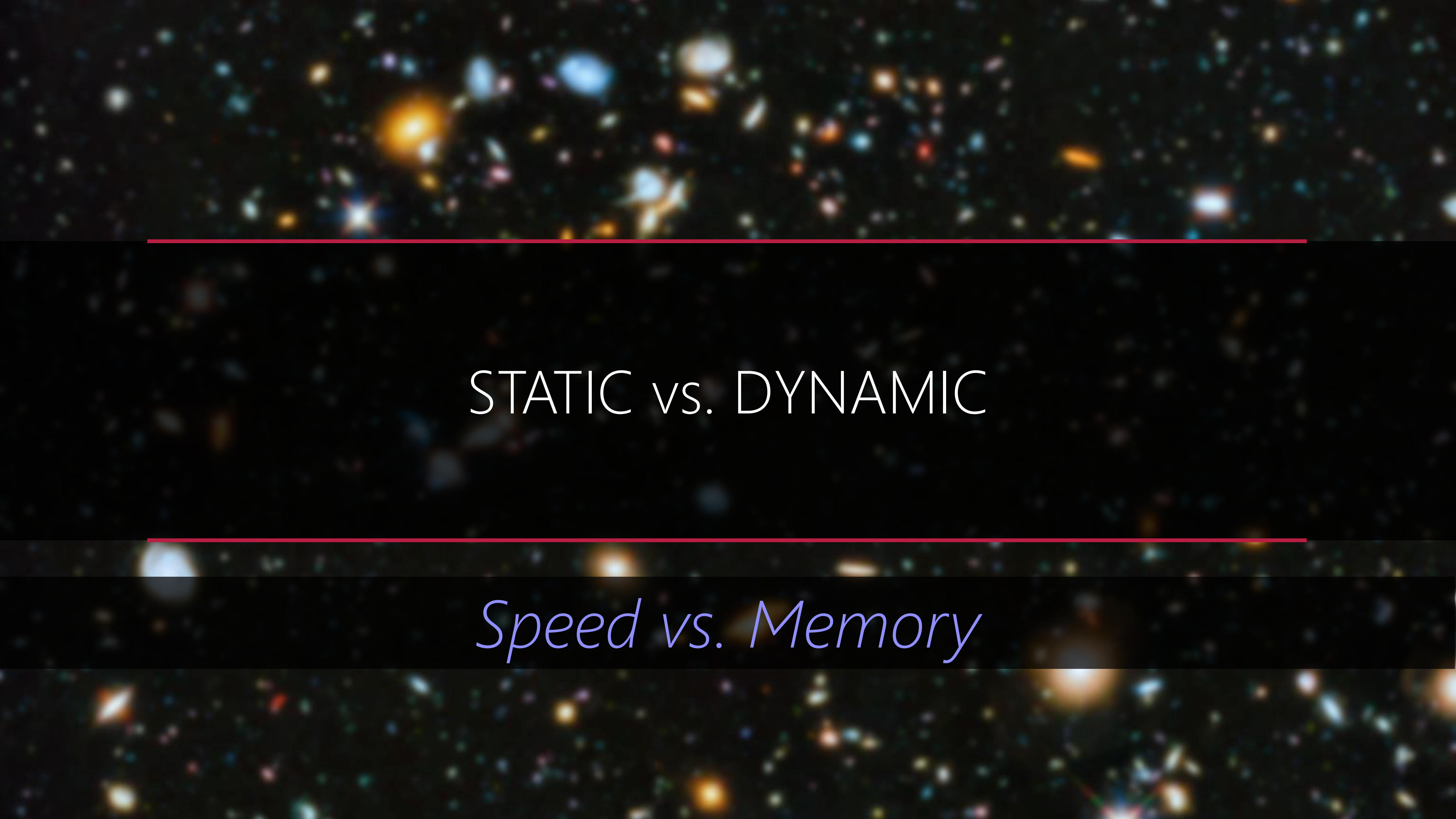




Is this a system call?


Data
Algorithm
Shell
Kernel
Computer
math
time
sys


Computer
Memory
Kernel
Shell
Process1: Program + Data
Bus
Processor



STATIC vs. DYNAMIC

Speed vs. Memory