CPU Simulator

(Inspired by slides by Martin Bättig and Gian Ulli)

Example Program

- How would the processor execute this program?

```
0xb000002a
0xb4000007
0xd1000000
0xc9000000
0xe8000000
0x00000000
```

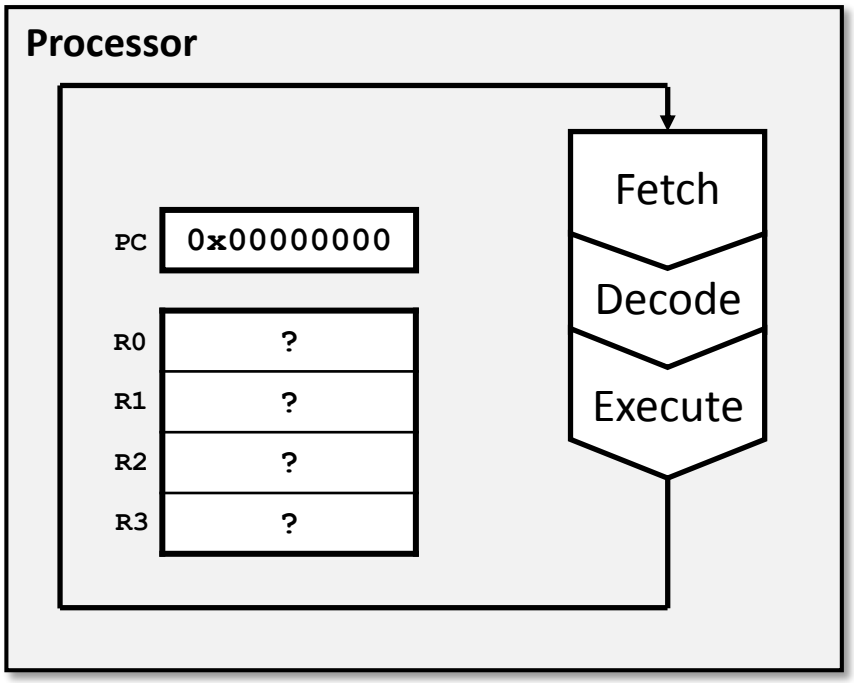
- Which means

```
mov r0, 42     ;load 42 into r0
mov r1,  7     ;load 7 into r1
st  r0, r1     ;store r0 into mem[r1]
ld  r2, r1     ;load mem[r1] into r2
out r2,  0     ;output r2 as number
hlt            ;stop
```

(; denote comments in assembly)

CPU Simulator

Initial State

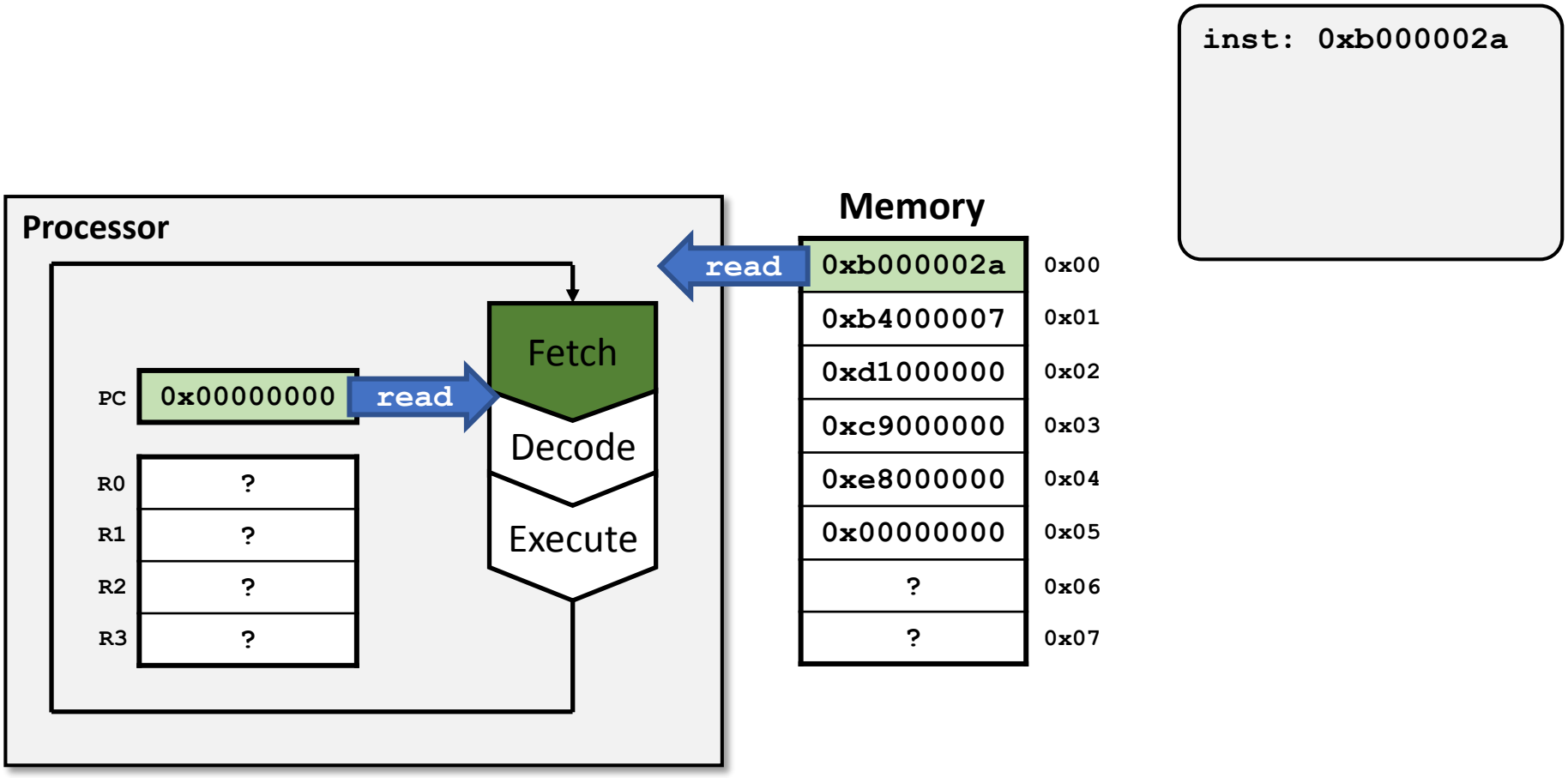


Memory

0xb000002a	0x00
0xb4000007	0x01
0xd1000000	0x02
0xc9000000	0x03
0xe8000000	0x04
0x00000000	0x05
?	0x06
?	0x07

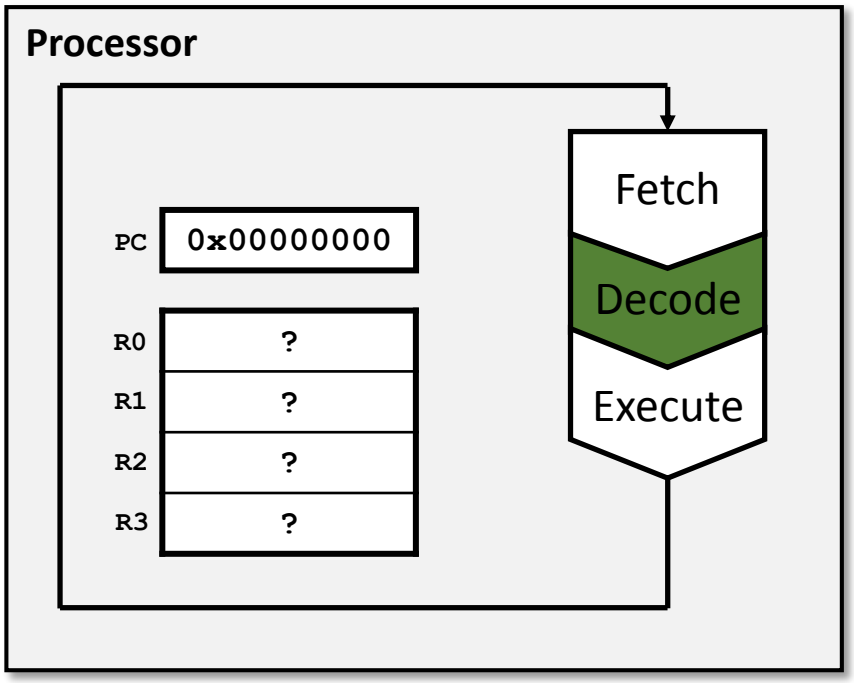
For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator



For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator



Memory

0xb000002a	0x00
0xb4000007	0x01
0xd1000000	0x02
0xc9000000	0x03
0xe8000000	0x04
0x00000000	0x05
?	0x06
?	0x07

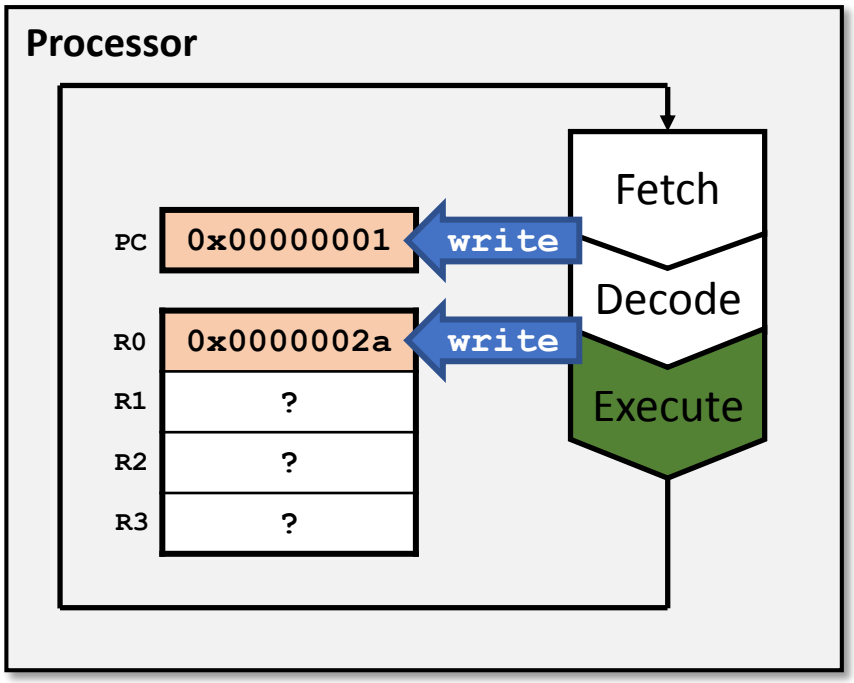
```
inst: 0xb000002a
Opcode: 0xb [mov]
Op A: 0x0
Op B: 0x0
Op C: 0x00002a
```

For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator

`mov`: Load the (unsigned) constant `c` into register `reg[A]`.

```
inst: 0xb000002a
Opcode: 0xb [mov]
Op A: 0x0
Op B: 0x0
Op C: 0x00002a
```



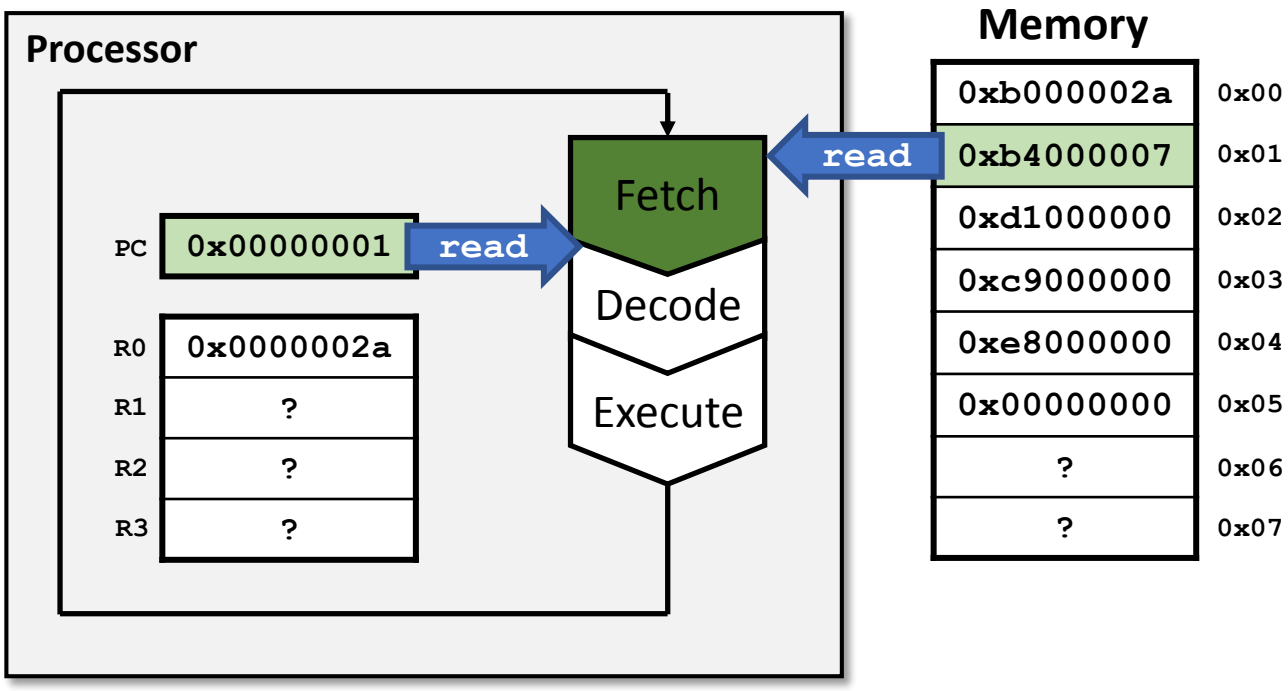
Memory

0xb000002a	0x00
0xb4000007	0x01
0xd1000000	0x02
0xc9000000	0x03
0xe8000000	0x04
0x00000000	0x05
?	0x06
?	0x07

For simplicity: Memory with just 8 adresses instead of 2^{16}.

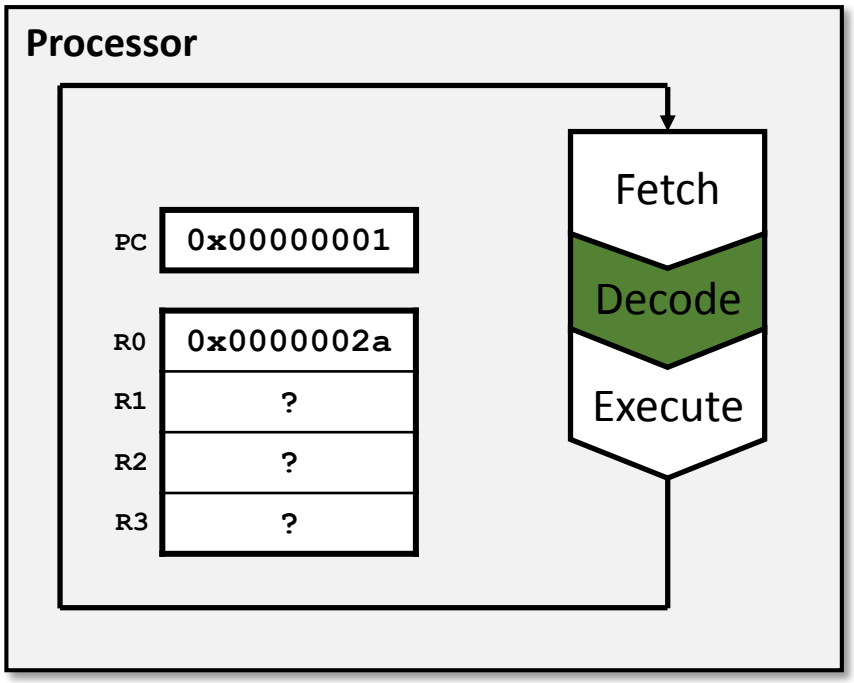
CPU Simulator

```
inst: 0xb4000007
```



For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator



Memory

0xb000002a	0x00
0xb4000007	0x01
0xd1000000	0x02
0xc9000000	0x03
0xe8000000	0x04
0x00000000	0x05
?	0x06
?	0x07

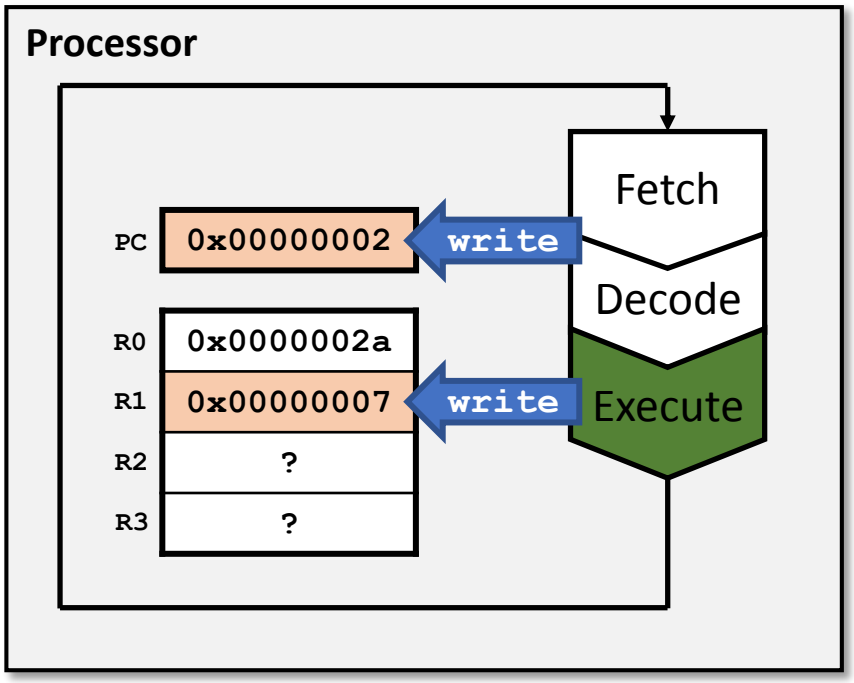
```
inst: 0xb4000007
Opcode: 0xb [mov]
Op A: 0x1
Op B: 0x0
Op C: 0x000007
```

For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator

`mov`: Load the (unsigned) constant `c` into register `reg[A]`.

```
inst: 0xb4000007
Opcode: 0xb [mov]
Op A: 0x1
Op B: 0x0
Op C: 0x000007
```

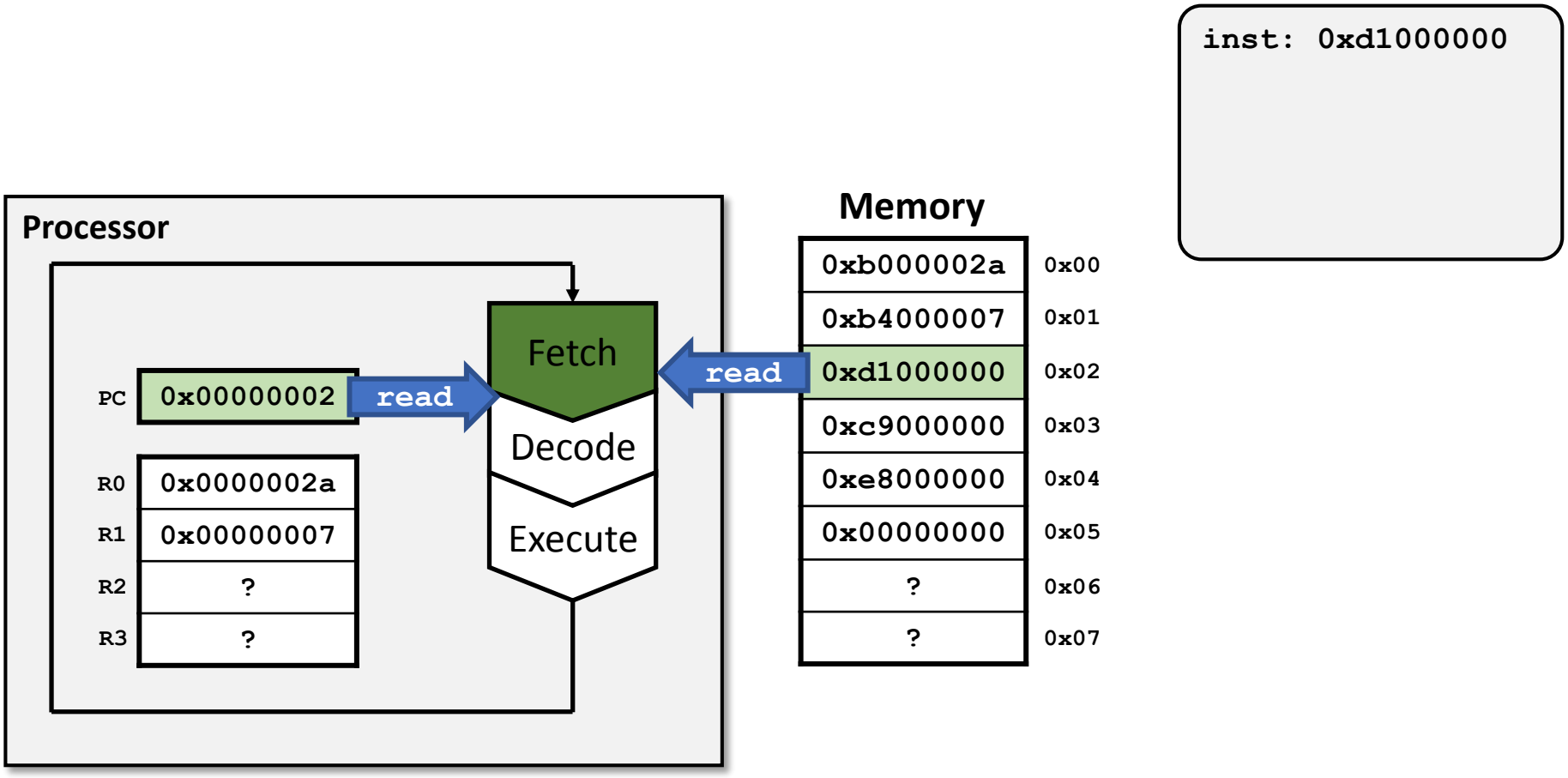


Memory

0xb000002a	0x00
0xb4000007	0x01
0xd1000000	0x02
0xc9000000	0x03
0xe8000000	0x04
0x00000000	0x05
?	0x06
?	0x07

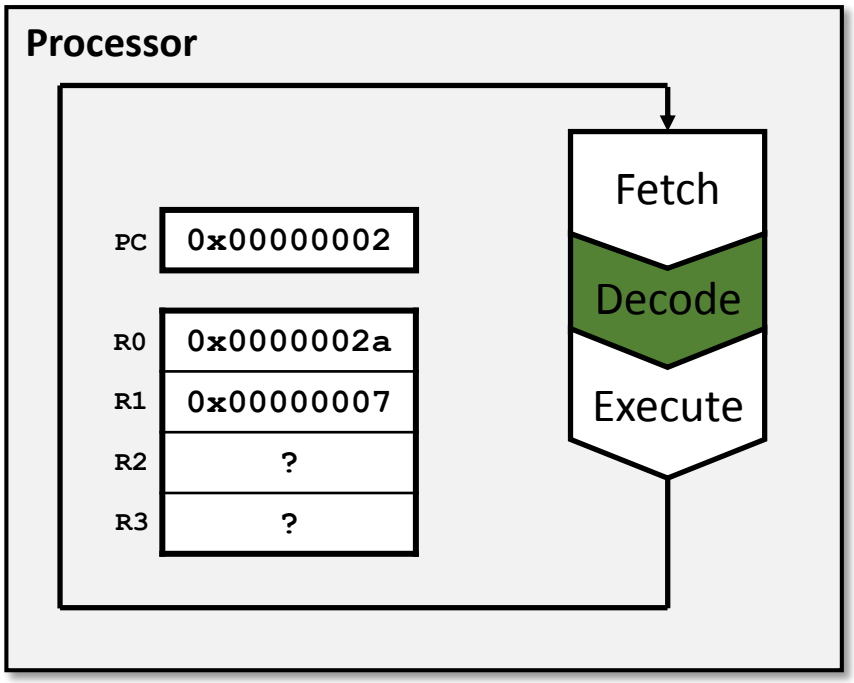
For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator



For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator



Memory

0xb000002a	0x00
0xb4000007	0x01
0xd1000000	0x02
0xc9000000	0x03
0xe8000000	0x04
0x00000000	0x05
?	0x06
?	0x07

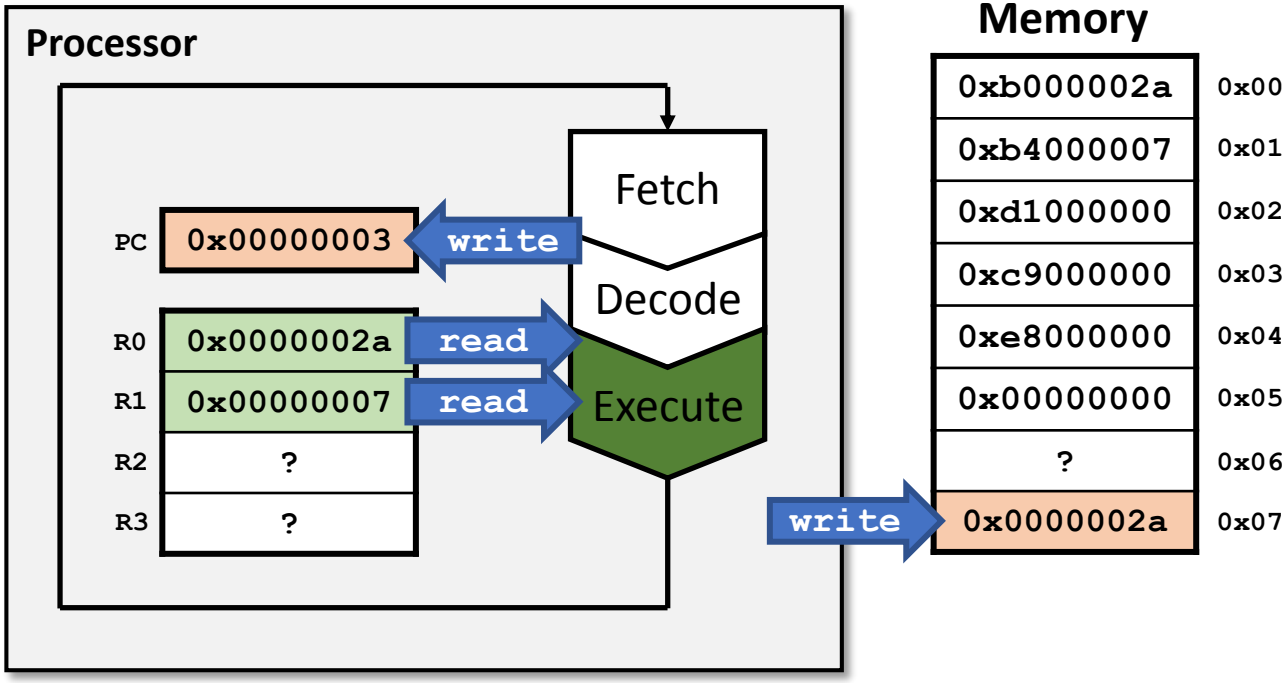
```
inst: 0xd1000000
Opcode: 0xd [st]
Op A: 0x0
Op B: 0x1
Op C: 0x000000
```

For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator

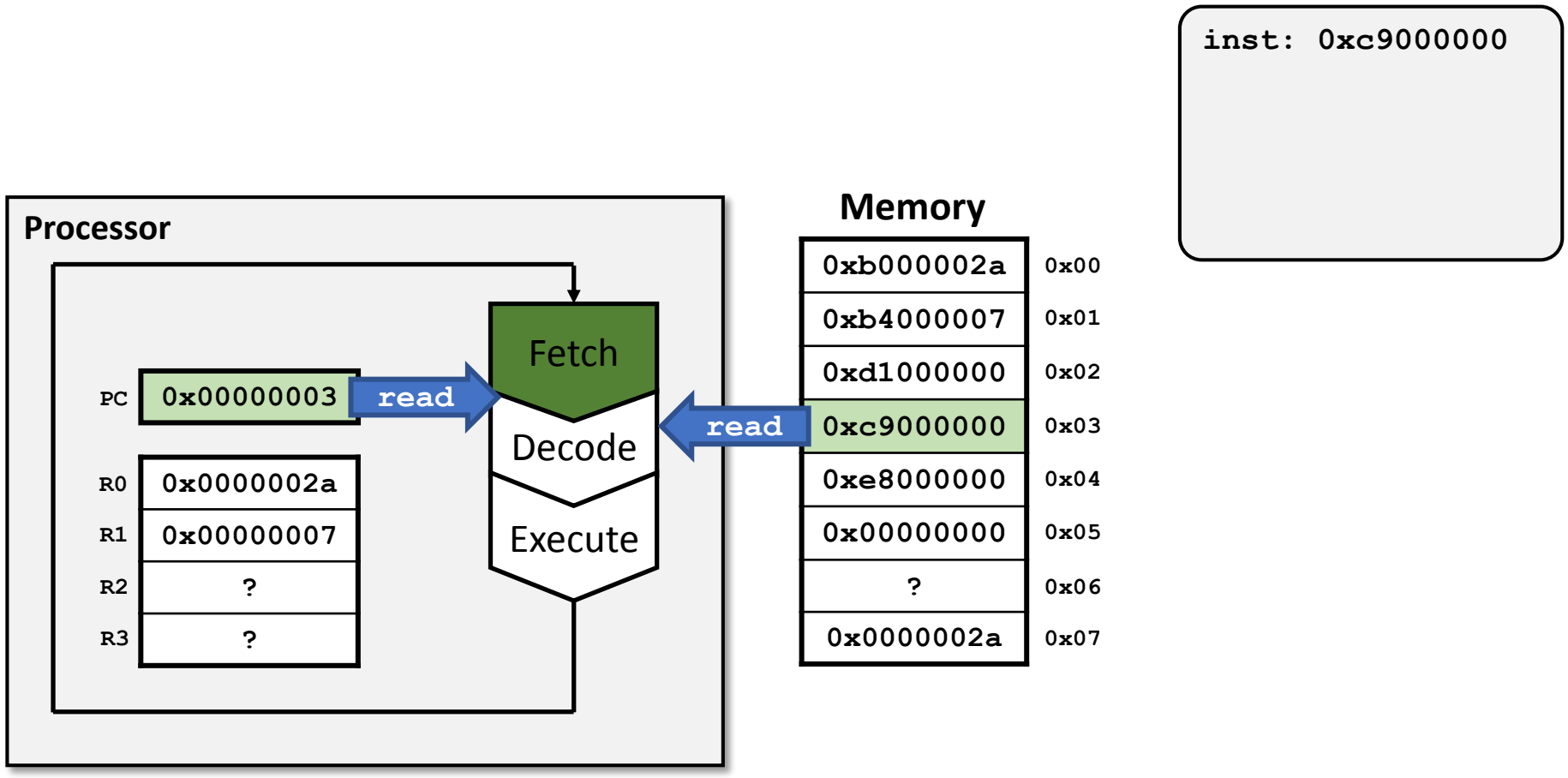
`st:` Store the value of `reg[A]` into `mem[reg[B]]`.

```
inst: 0xd1000000
Opcode: 0xd [st]
Op A: 0x0
Op B: 0x1
Op C: 0x000000
```



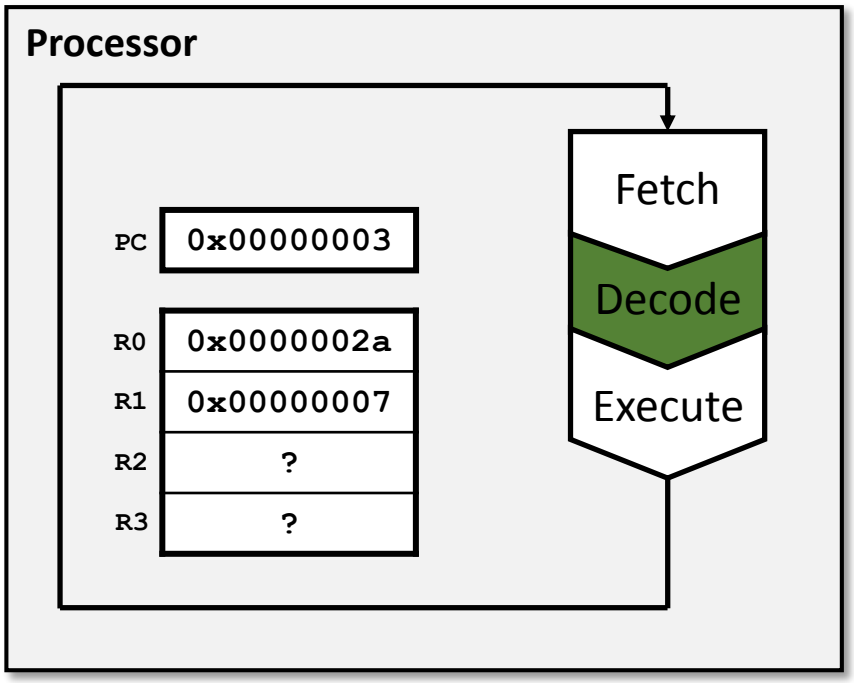
For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator



For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator



Memory

0xb000002a	0x00
0xb4000007	0x01
0xd1000000	0x02
0xc9000000	0x03
0xe8000000	0x04
0x00000000	0x05
?	0x06
0x0000002a	0x07

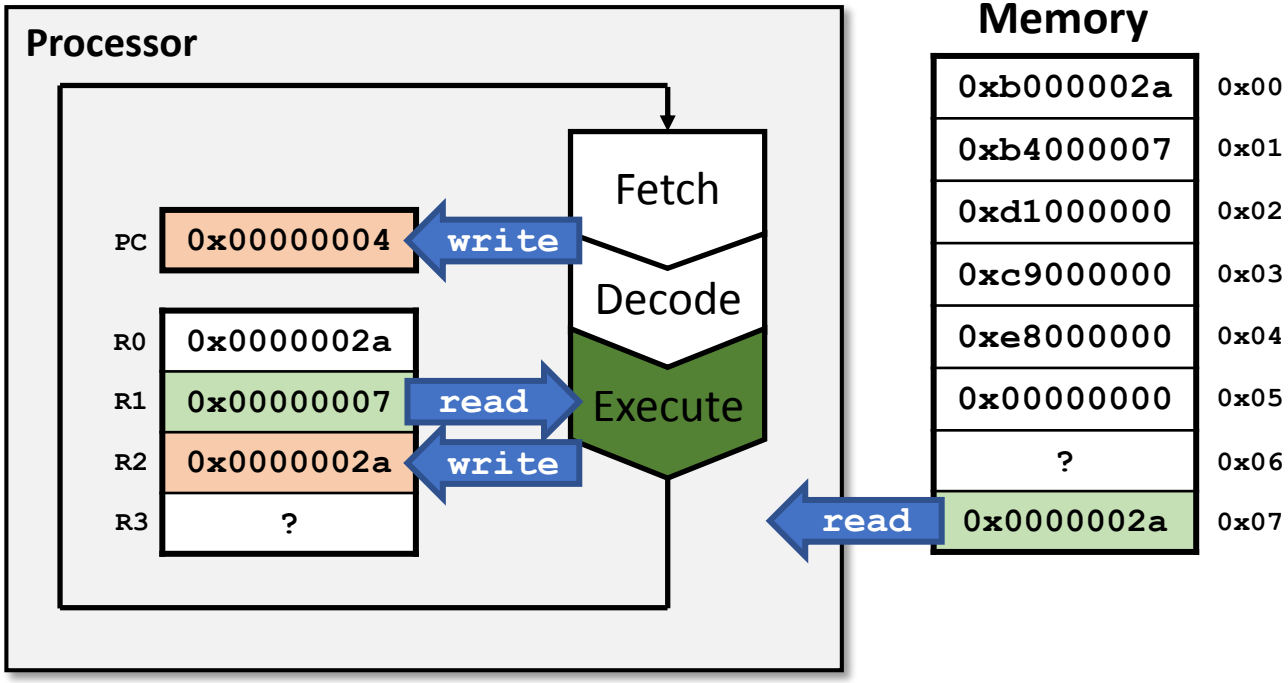
```
inst: 0xc9000000
Opcode: 0xc [ld]
Op A: 0x2
Op B: 0x1
Op C: 0x000000
```

For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator

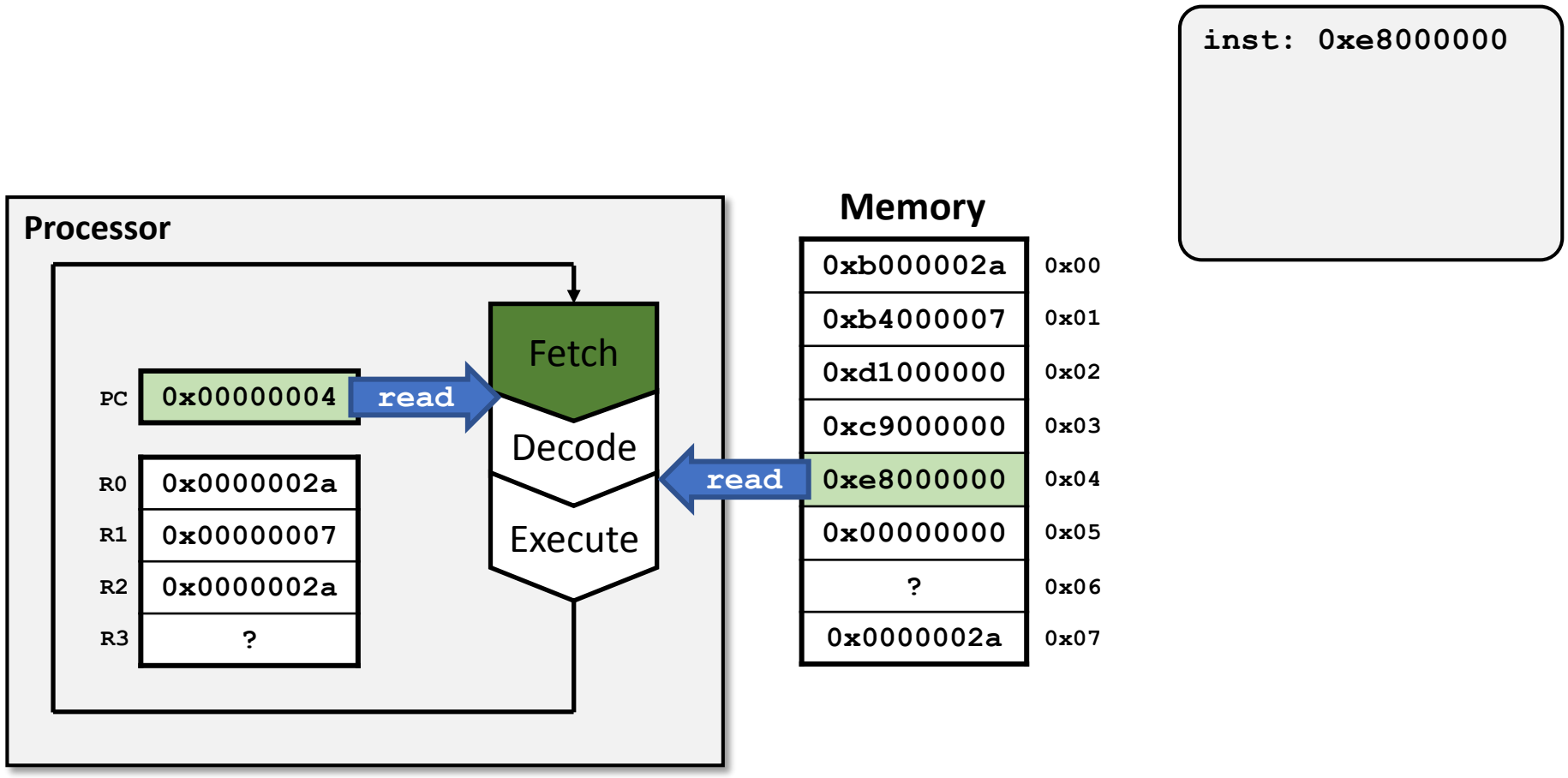
`ld`: Load the value stored at `mem[reg[B]]` into `reg[A]`.

```
inst: 0xc9000000
Opcode: 0xc [ld]
Op A: 0x2
Op B: 0x1
Op C: 0x000000
```



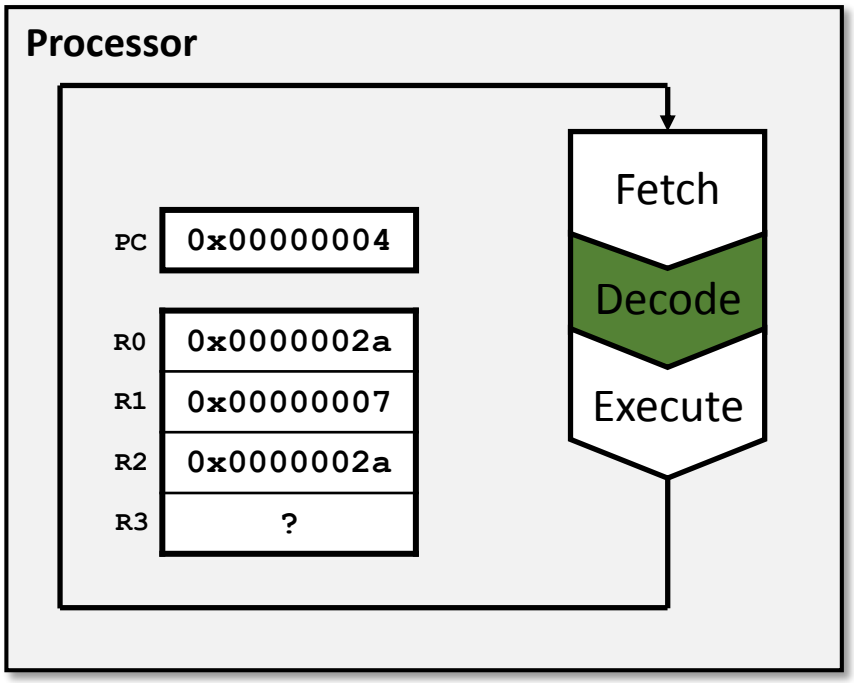
For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator



For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator



Memory

0xb000002a	0x00
0xb4000007	0x01
0xd1000000	0x02
0xc9000000	0x03
0xe8000000	0x04
0x00000000	0x05
?	0x06
0x0000002a	0x07

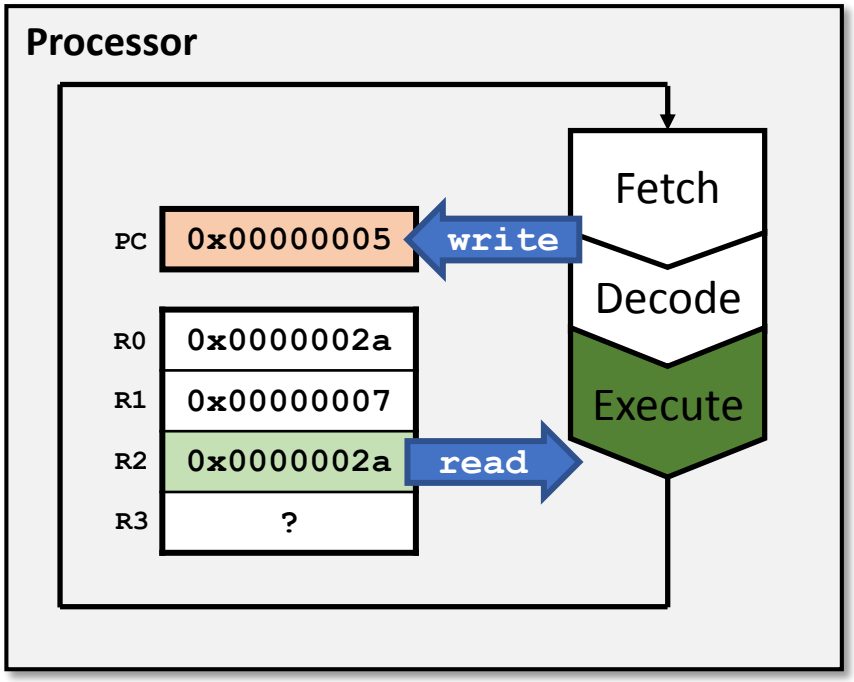
```
inst: 0xe8000000
Opcode: 0xe [out]
Op A: 0x2
Op B: 0x0
Op C: 0x000000
```

For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator

`out:` Write the value stored at `reg[A]` to standard out `std::cout`. If `c = 0` then write as 32-bit unsigned decimal integer else write as character.

```
inst: 0xe8000000
Opcode: 0xe [out]
Op A: 0x2
Op B: 0x0
Op C: 0x000000
```



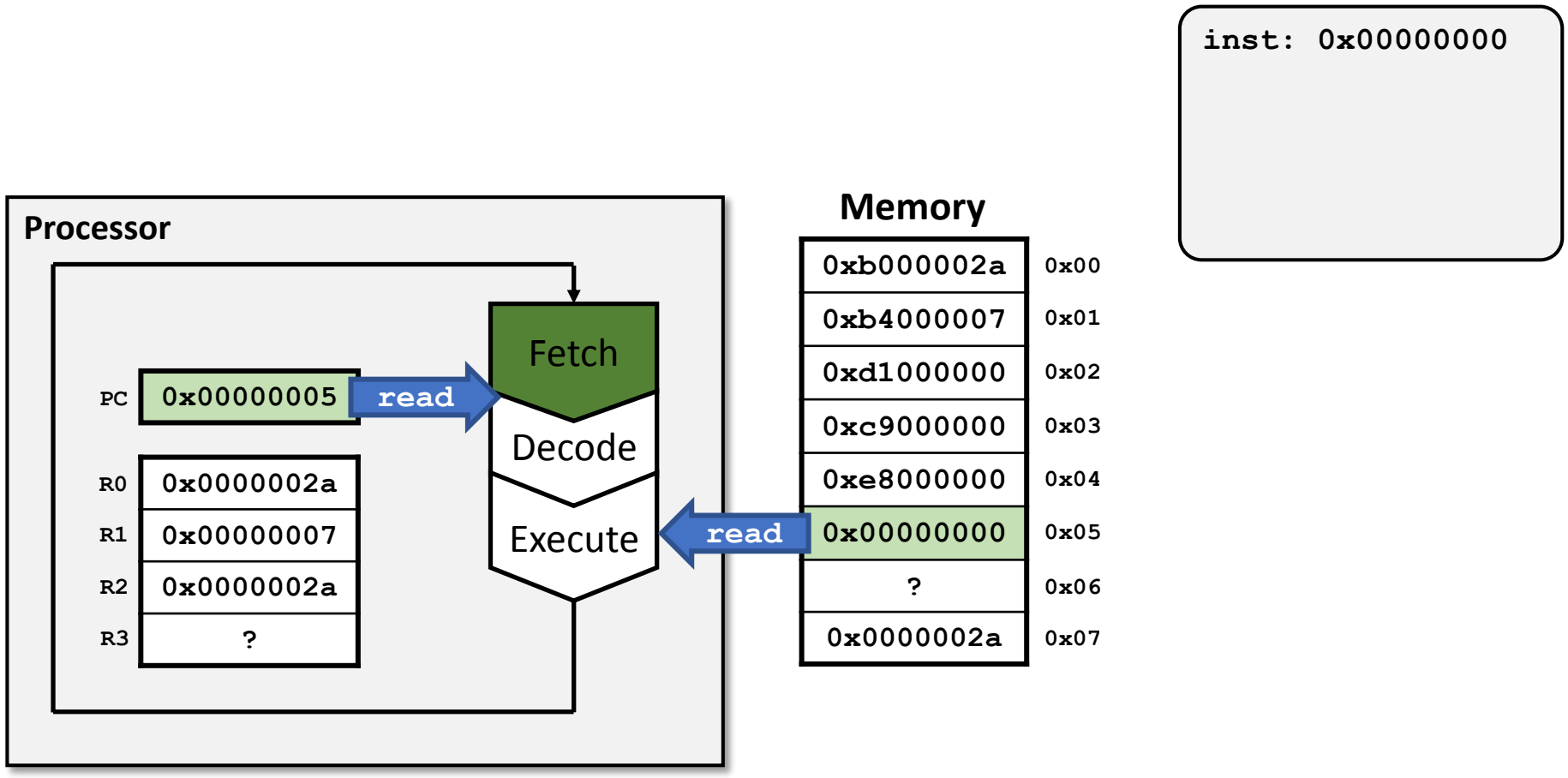
Memory

0xb000002a	0x00
0xb4000007	0x01
0xd1000000	0x02
0xc9000000	0x03
0xe8000000	0x04
0x00000000	0x05
?	0x06
0x0000002a	0x07

Output: 42

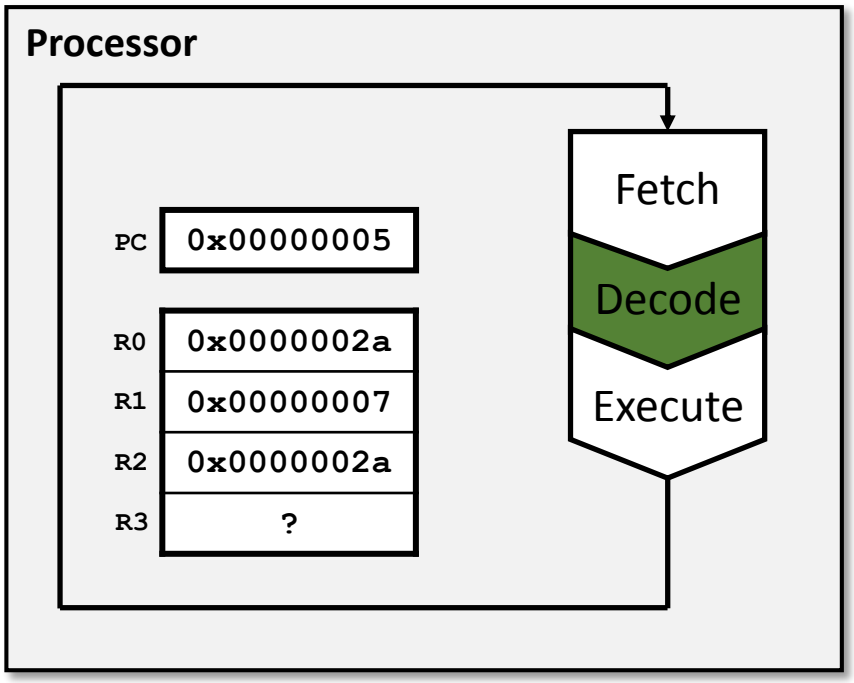
For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator



For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator



Memory

0xb000002a	0x00
0xb4000007	0x01
0xd1000000	0x02
0xc9000000	0x03
0xe8000000	0x04
0x00000000	0x05
?	0x06
0x0000002a	0x07

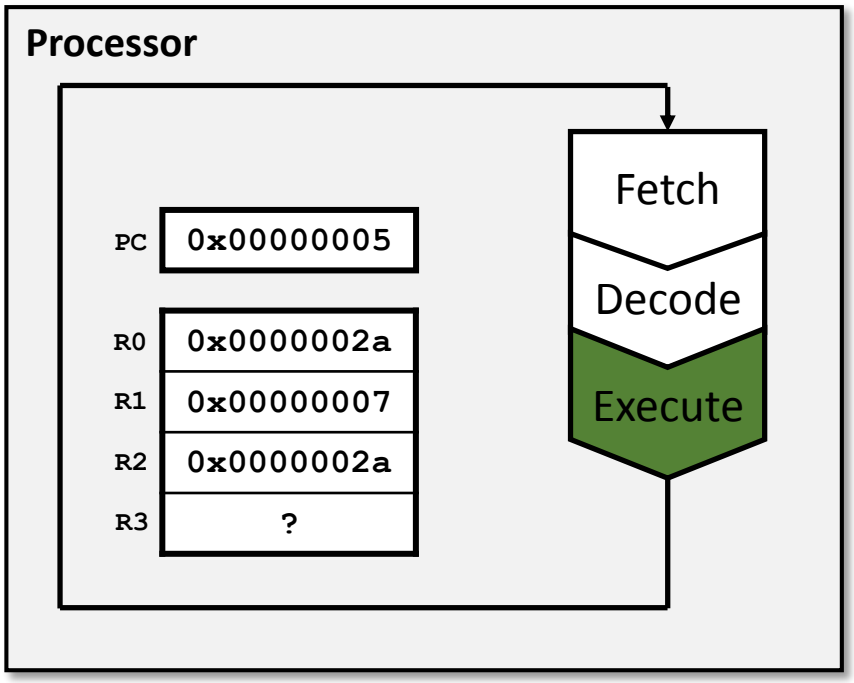
```
inst: 0x00000000
Opcode: 0x0 [hlt]
Op A: 0x0
Op B: 0x0
Op C: 0x000000
```

For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator

`hlt:` Halts the system.

```
inst: 0x00000000
Opcode: 0x0 [hlt]
Op A: 0x0
Op B: 0x0
Op C: 0x000000
```



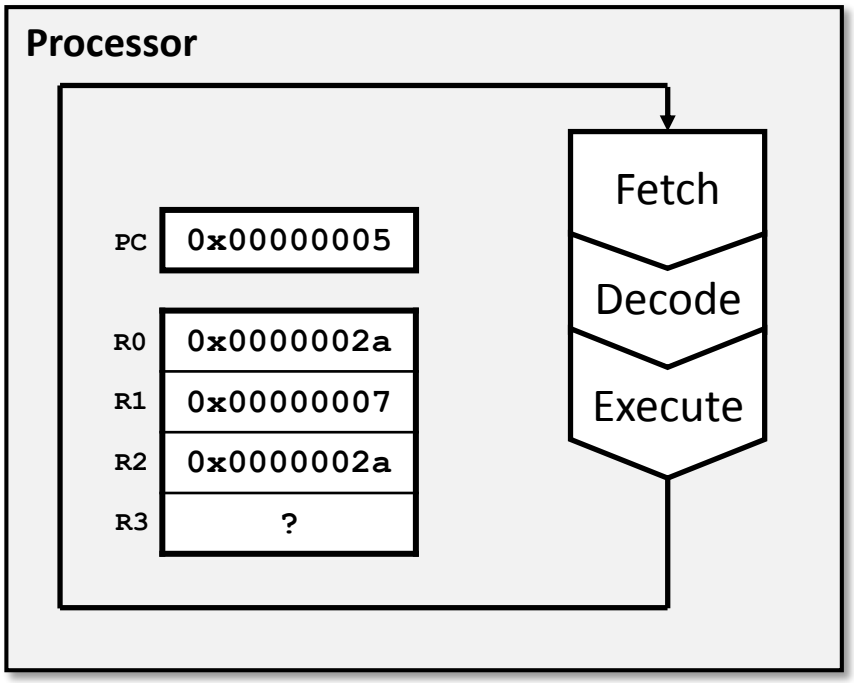
Memory

0xb000002a	0x00
0xb4000007	0x01
0xd1000000	0x02
0xc9000000	0x03
0xe8000000	0x04
0x00000000	0x05
?	0x06
0x0000002a	0x07

For simplicity: Memory with just 8 adresses instead of 2^{16}.

CPU Simulator

Simulation finished



Memory

0xb000002a	0x00
0xb4000007	0x01
0xd1000000	0x02
0xc9000000	0x03
0xe8000000	0x04
0x00000000	0x05
?	0x06
0x0000002a	0x07

For simplicity: Memory with just 8 adresses instead of 2^{16}.