Systemnahe Programmierung in C

16 μC-Systemarchitektur – Prozessor

J. Kleinöder, D. Lohmann, V. Sieh

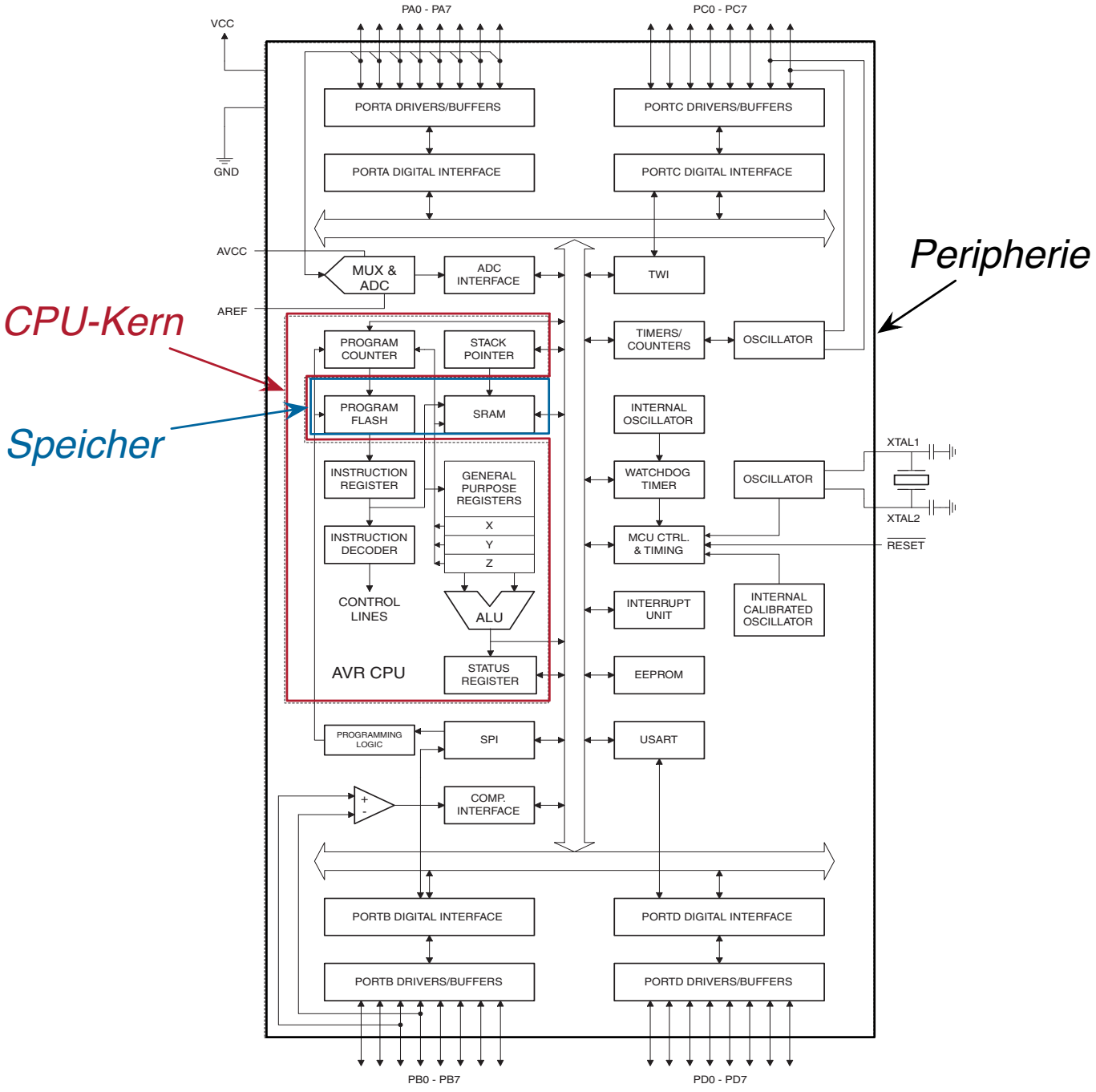
Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Erlangen-Nürnberg

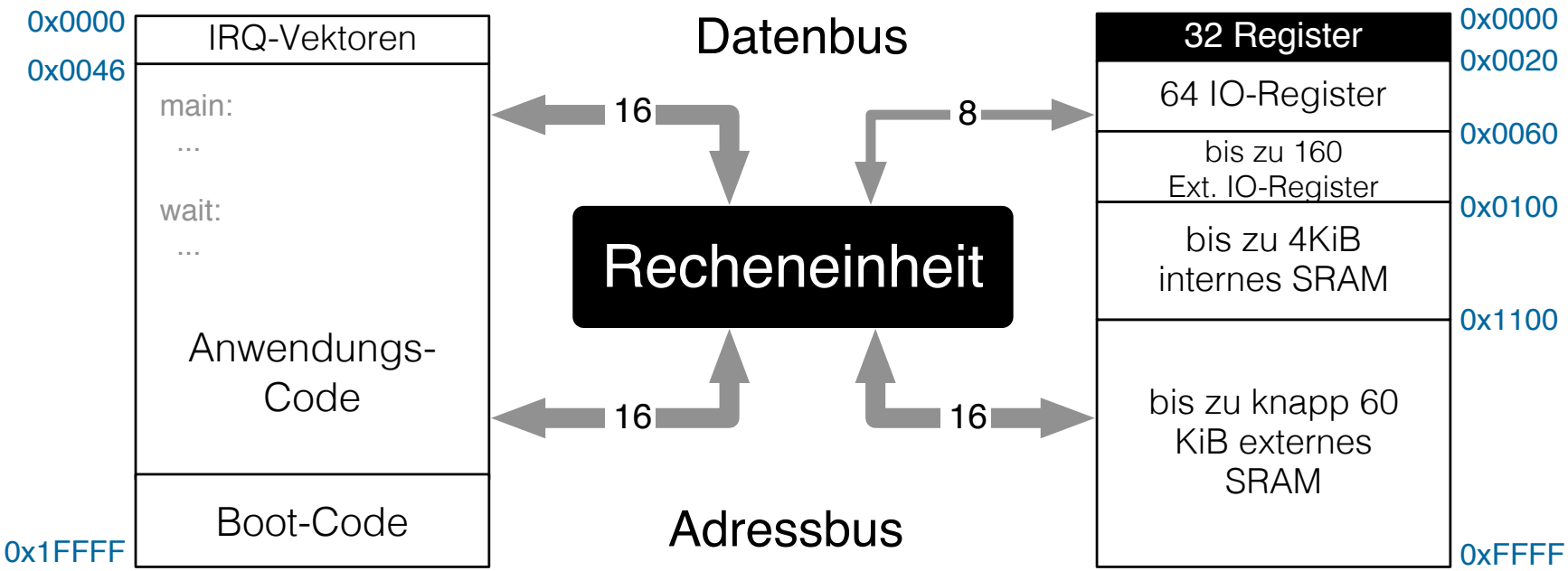
Sommersemester 2024

http://sys.cs.fau.de/lehre/ss24

Beispiel ATmega32: Blockschaltbild



Beispiel ATmega-Familie: CPU-Architektur

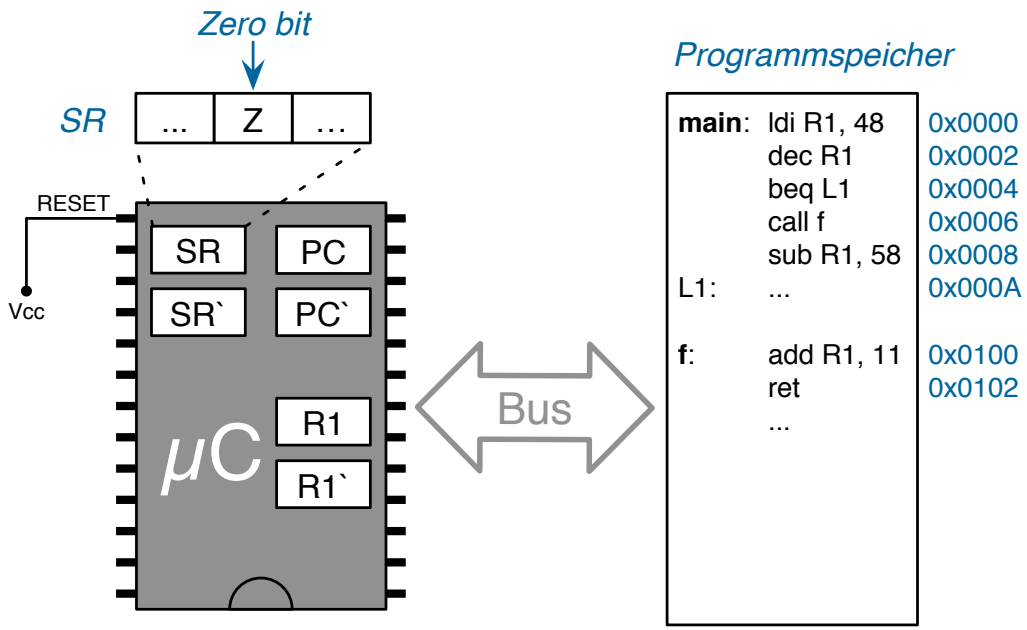


Programmadressraum (Flash)
1–128 KiB, 2-Byte-Elemente

Datenadressraum (SRAM)
0–64 KiB, 1-Byte-Elemente

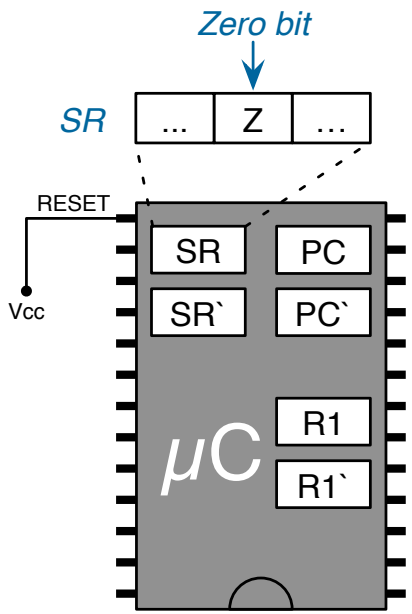
- Harvard-Architektur (getrennter Speicher für Code und Daten)
- Peripherie-Register sind in den Speicher eingeblendet
  $\rightsquigarrow$ ansprechbar wie globale Variablen

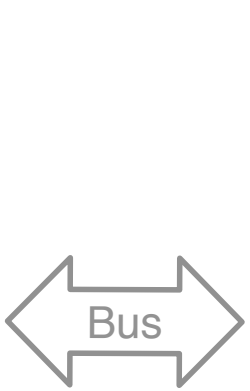
Wie arbeitet ein Prozessor?



- Hier am Beispiel eines sehr einfachen Pseudoprozessors
  - Nur ein Vielzweckregister (R1)
  - Programmzähler (PC) und Statusregister (SR) (+ „Schattenkopien“)
  - Kein Datenspeicher, kein Stapel $\rightsquigarrow$ Programm arbeitet nur auf Registern

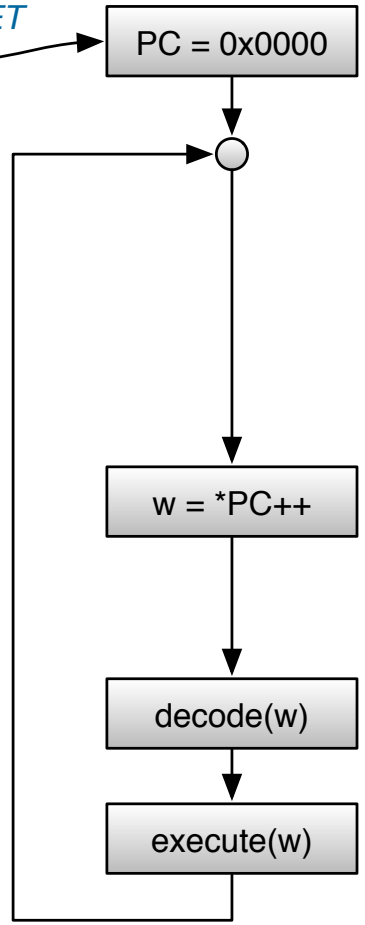
Wie arbeitet ein Prozessor?



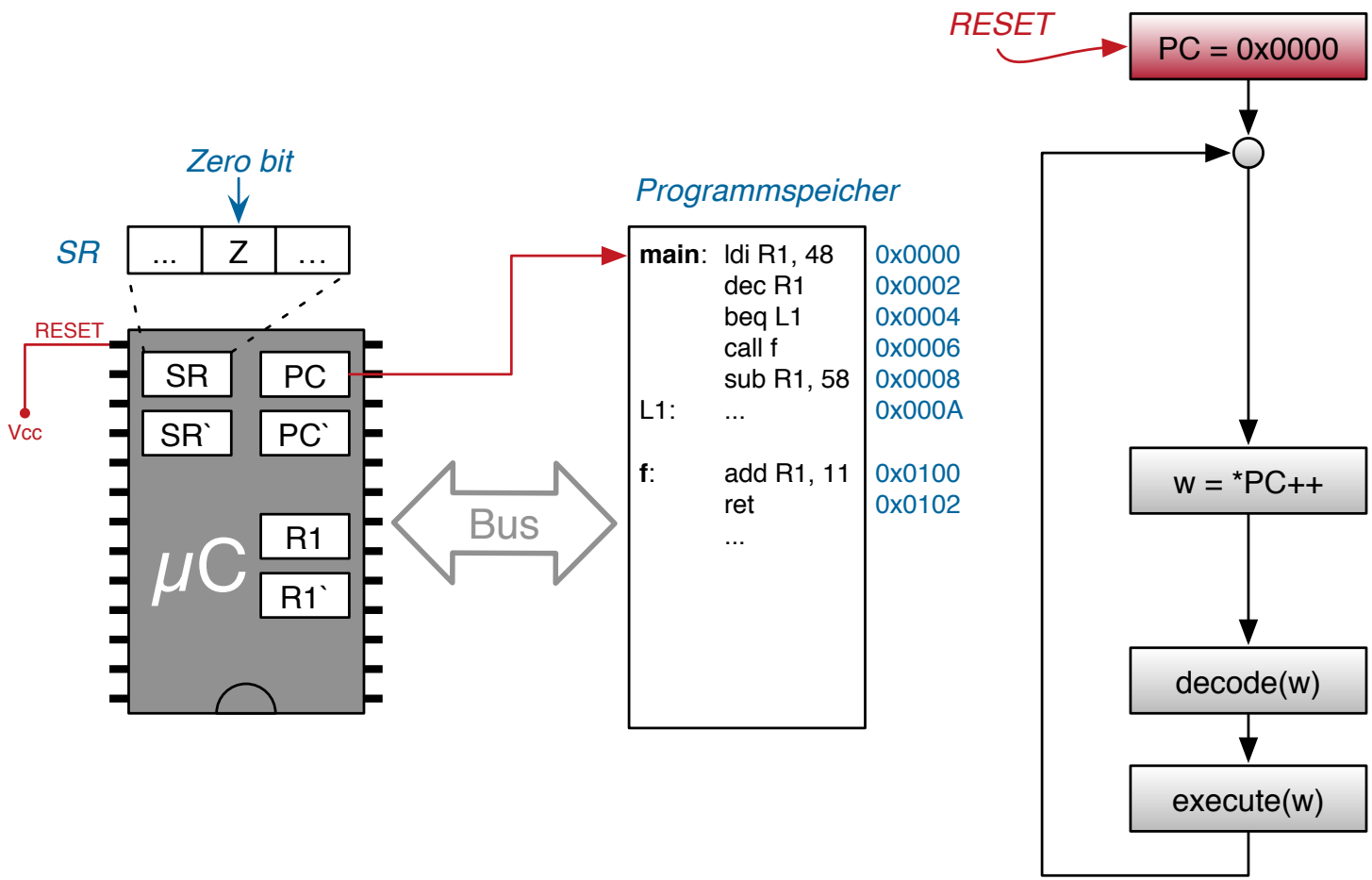


Programmspeicher

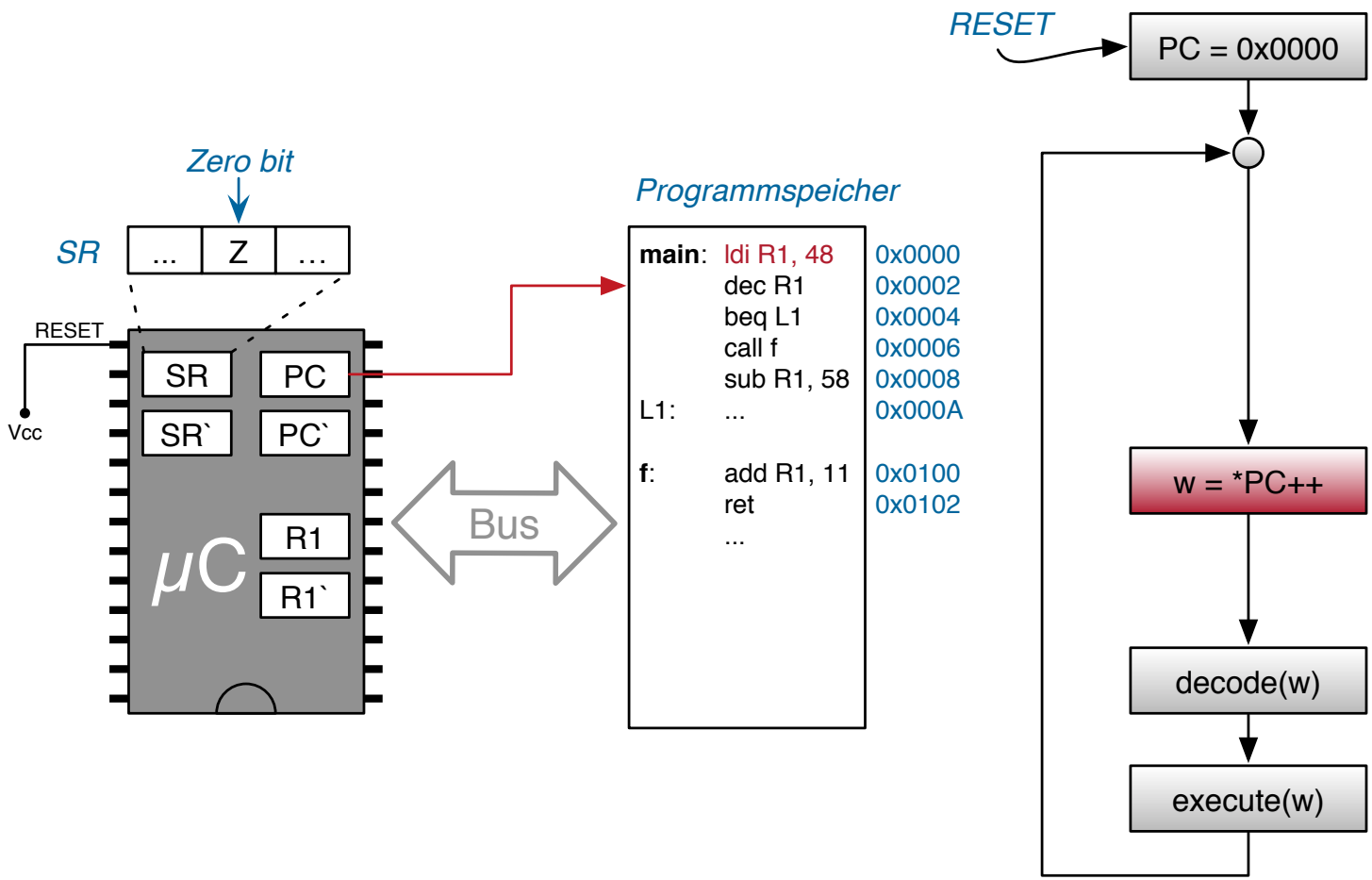
```
main:  ldi R1, 48      0x0000
       dec R1          0x0002
       beq L1          0x0004
       call f          0x0006
       sub R1, 58      0x0008
L1:    ...             0x000A

f:     add R1, 11      0x0100
       ret             0x0102
       ...
```



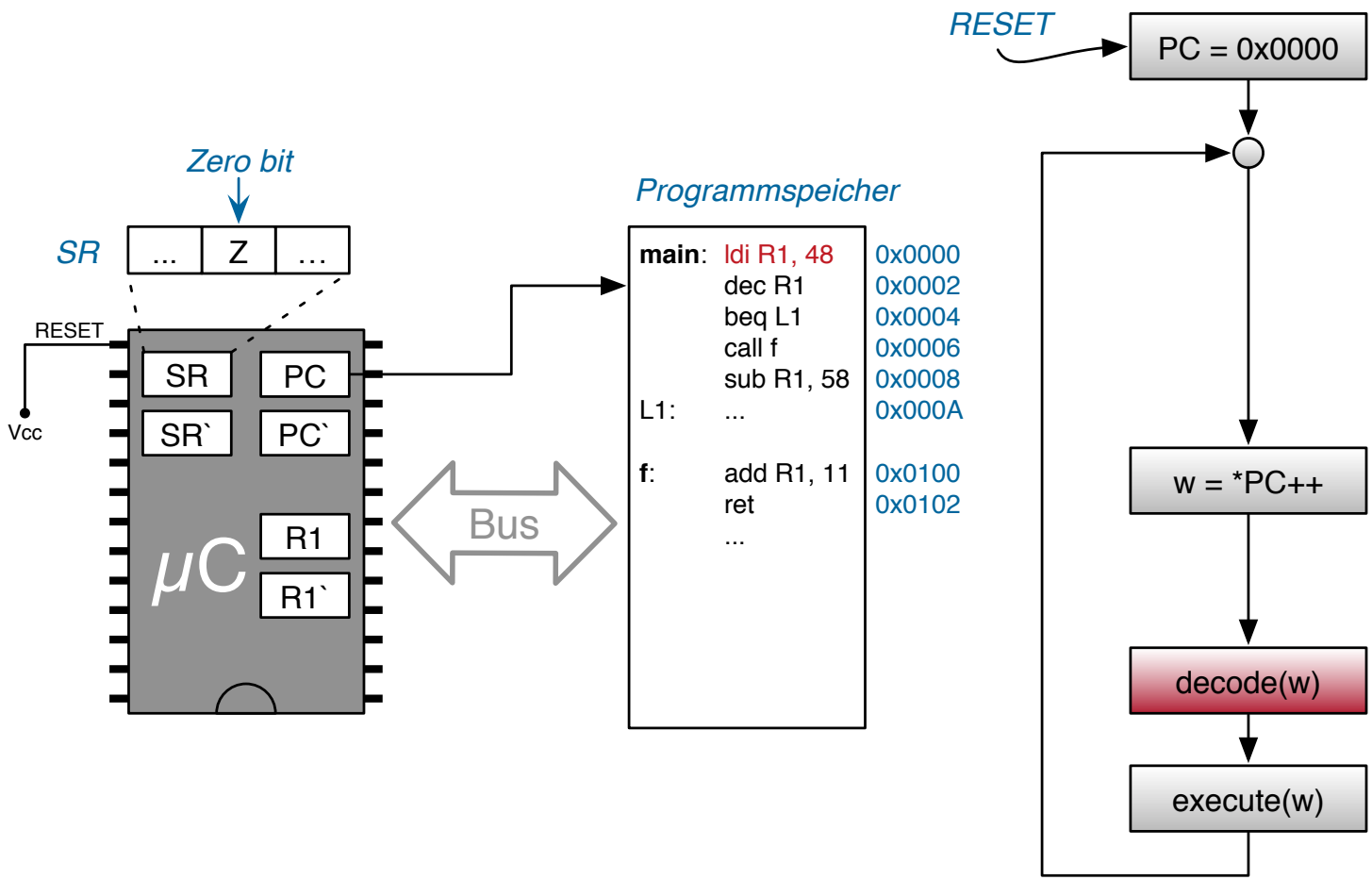
Wie arbeitet ein Prozessor?



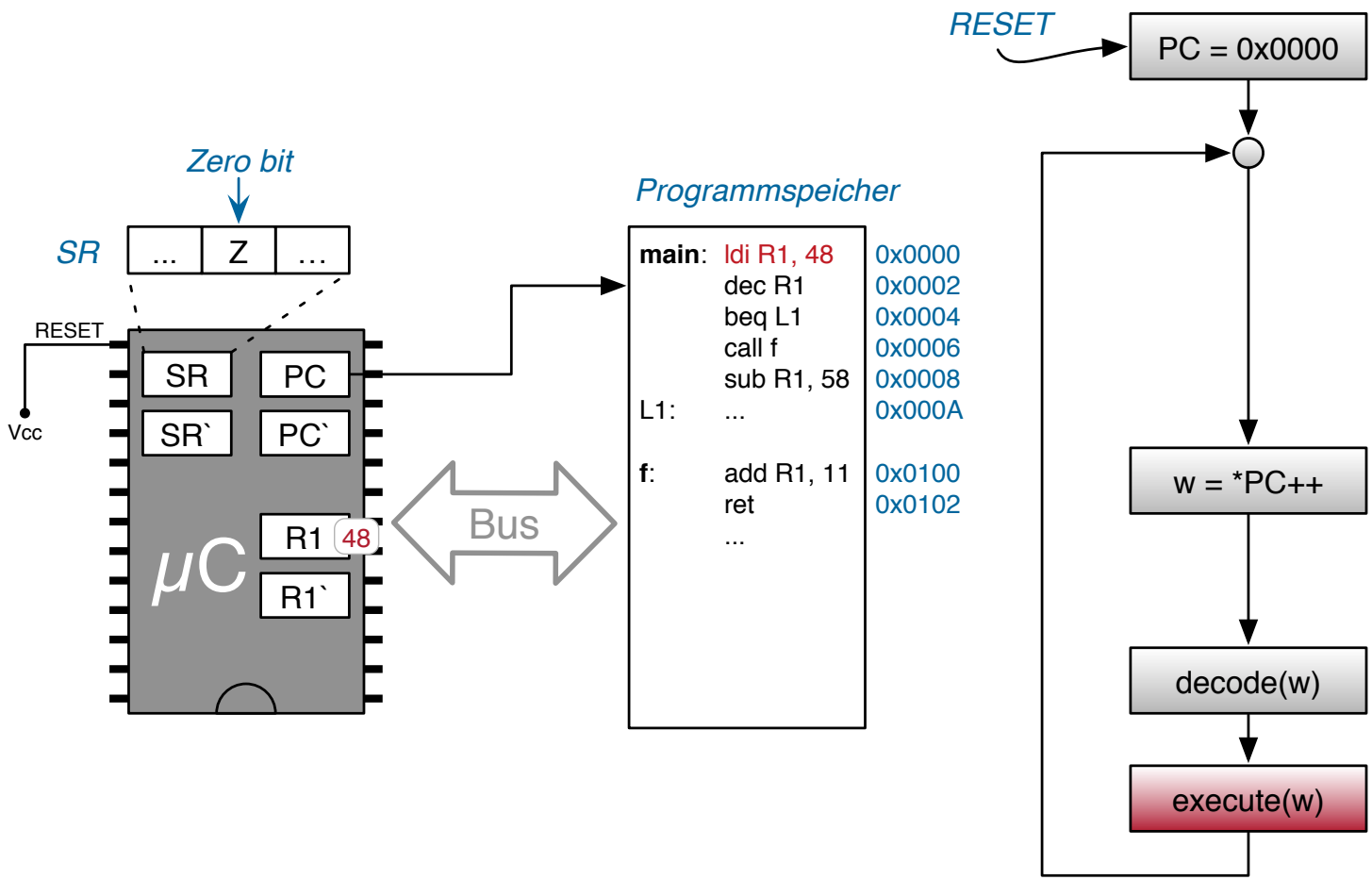
Wie arbeitet ein Prozessor?



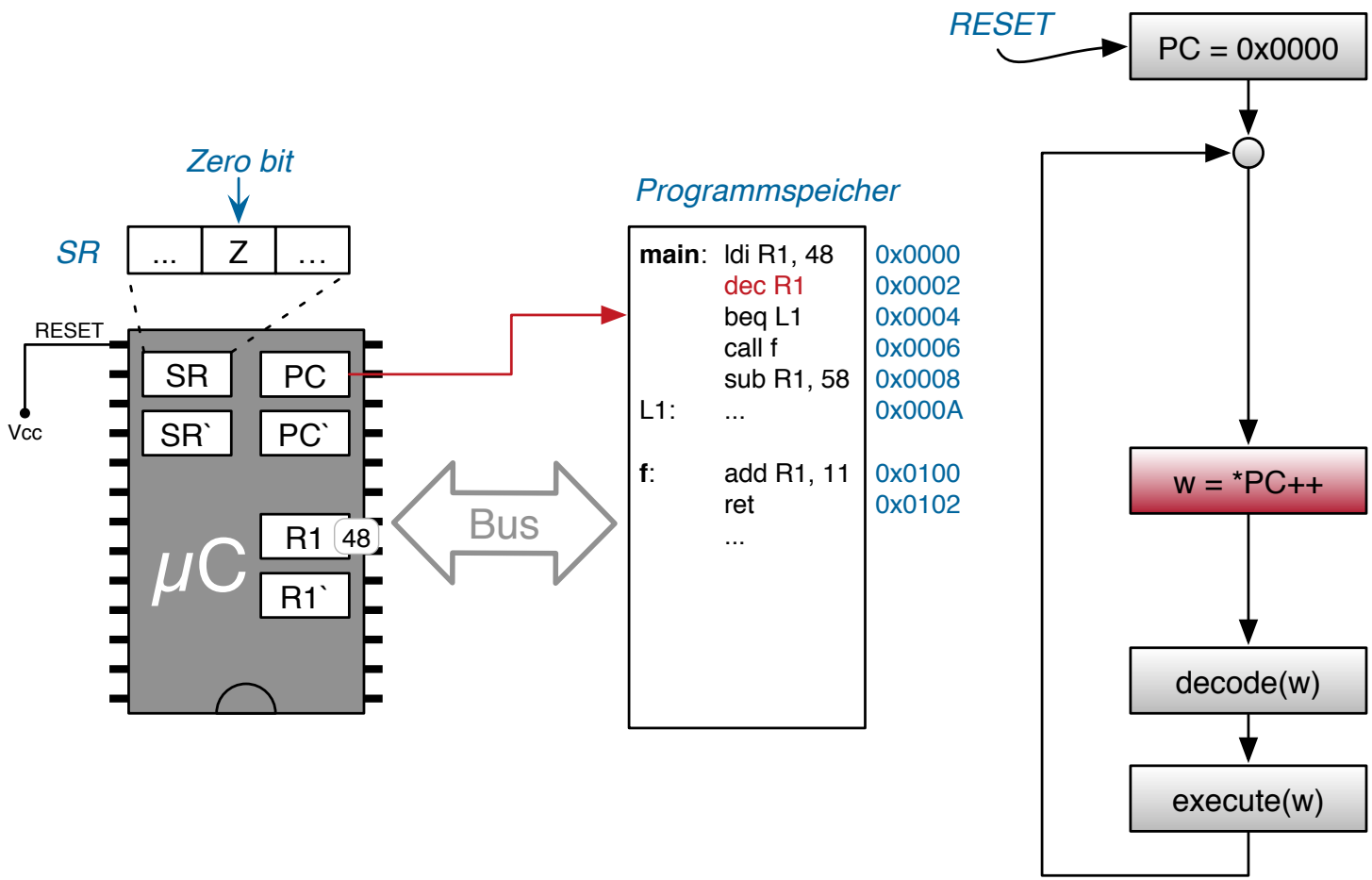
Wie arbeitet ein Prozessor?



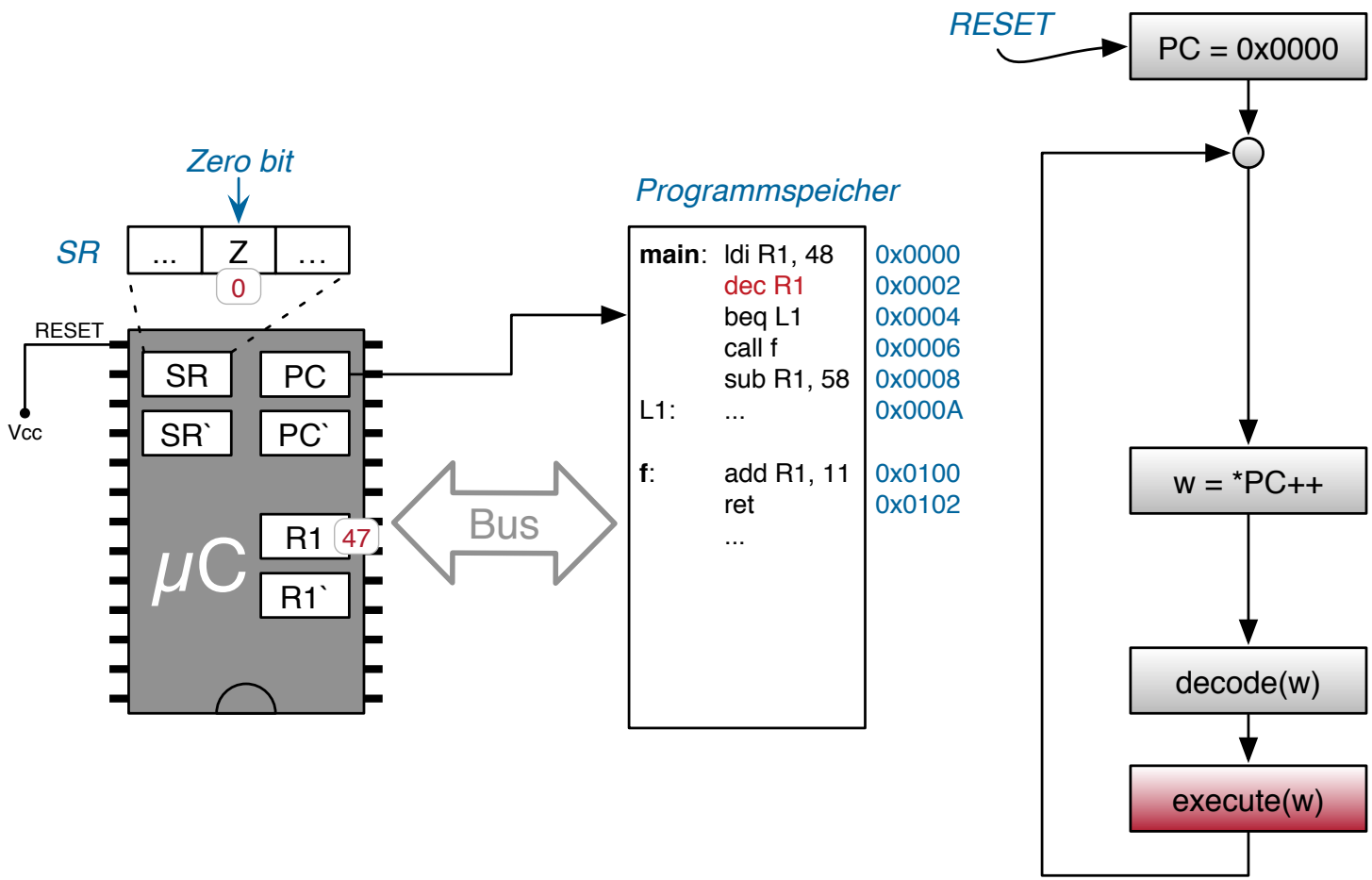
Wie arbeitet ein Prozessor?



Wie arbeitet ein Prozessor?

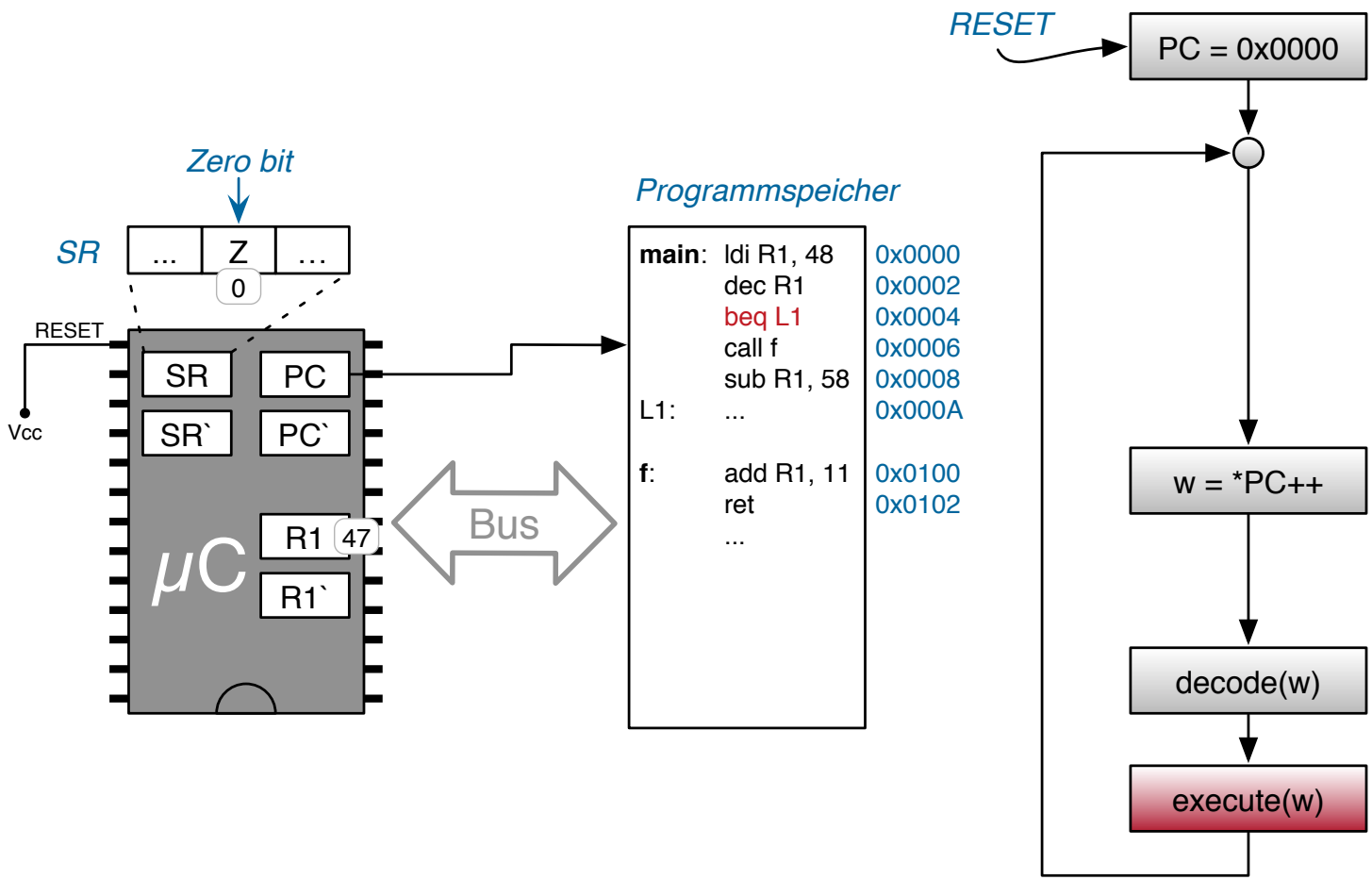


Wie arbeitet ein Prozessor?



w: **dec** <R>
R –= 1
if(R == 0) Z = 1
else Z = 0

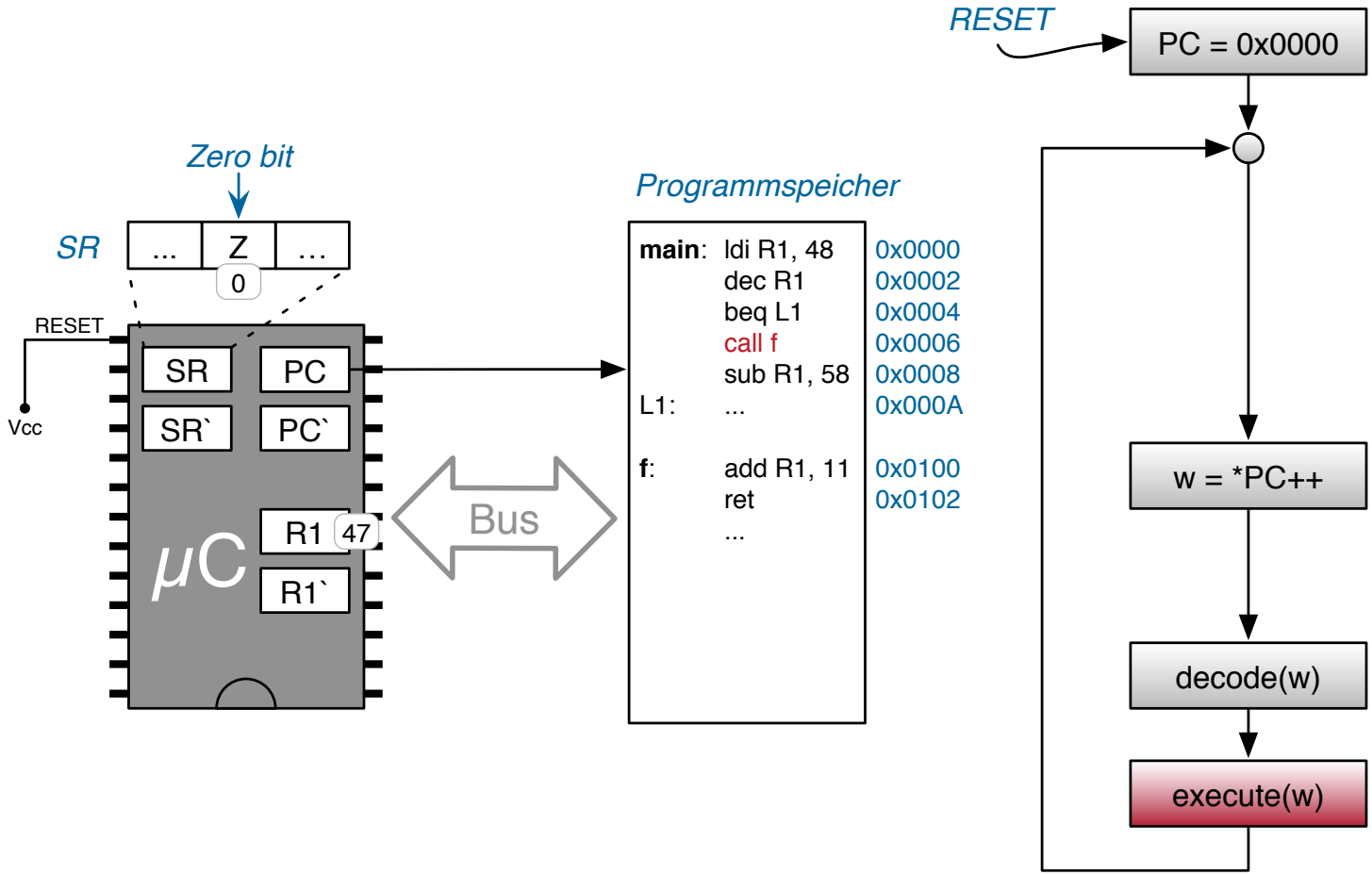
Wie arbeitet ein Prozessor?



w: **dec** <R>
R –= 1
if(R == 0) Z = 1
else Z = 0

w: **beq** <lab>
if(Z) PC = lab

Wie arbeitet ein Prozessor?

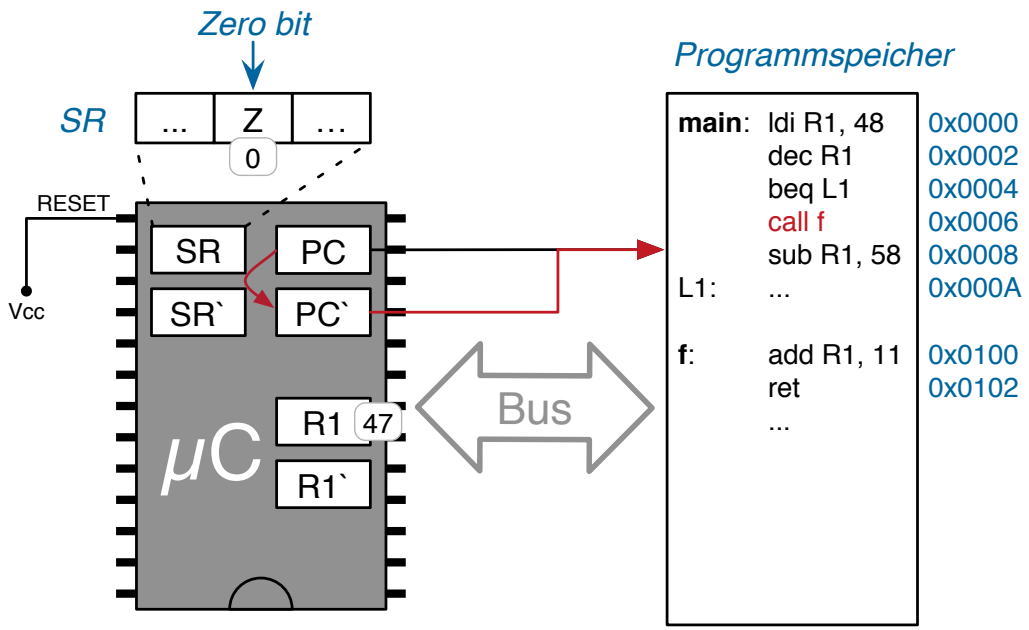


w: **dec** <R>
R –= 1
if(R == 0) Z = 1
else Z = 0

w: **beq** <lab>
if(Z) PC = lab

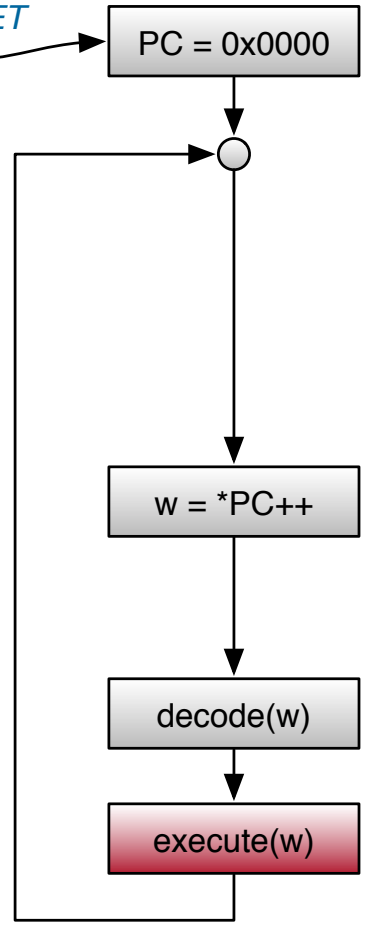
w: **call** <func>
PC` = PC
PC = func

Wie arbeitet ein Prozessor?



Programmspeicher

```
main:  ldi R1, 48      0x0000
       dec R1          0x0002
       beq L1          0x0004
       call f          0x0006
       sub R1, 58      0x0008
L1:    ...             0x000A

f:     add R1, 11      0x0100
       ret             0x0102
       ...
```

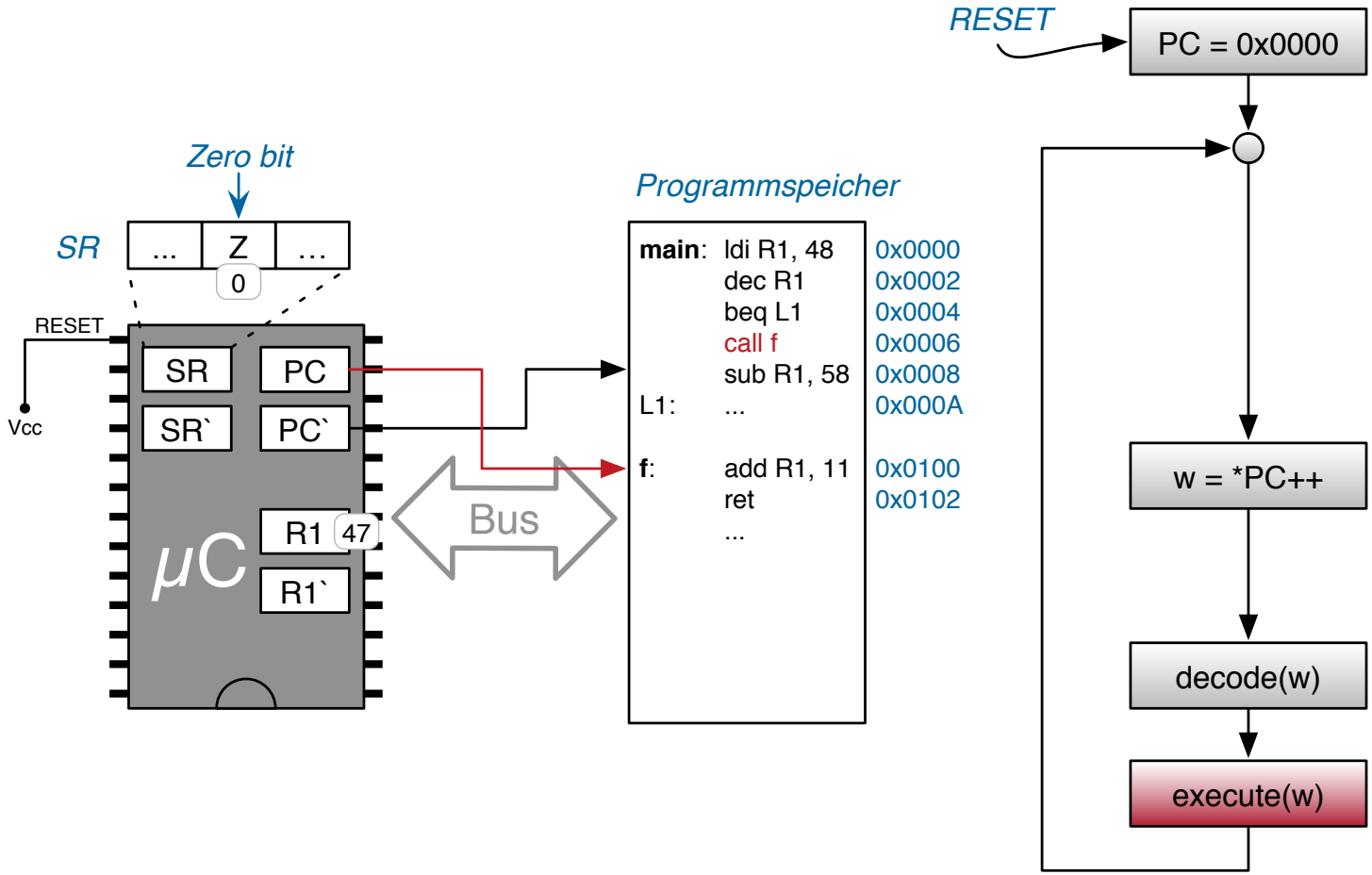


w: **dec** <R>
R –= 1
if(R == 0) Z = 1
else Z = 0

w: **beq** <lab>
if(Z) PC = lab

w: **call** <func>
PC` = PC
PC = func

Wie arbeitet ein Prozessor?

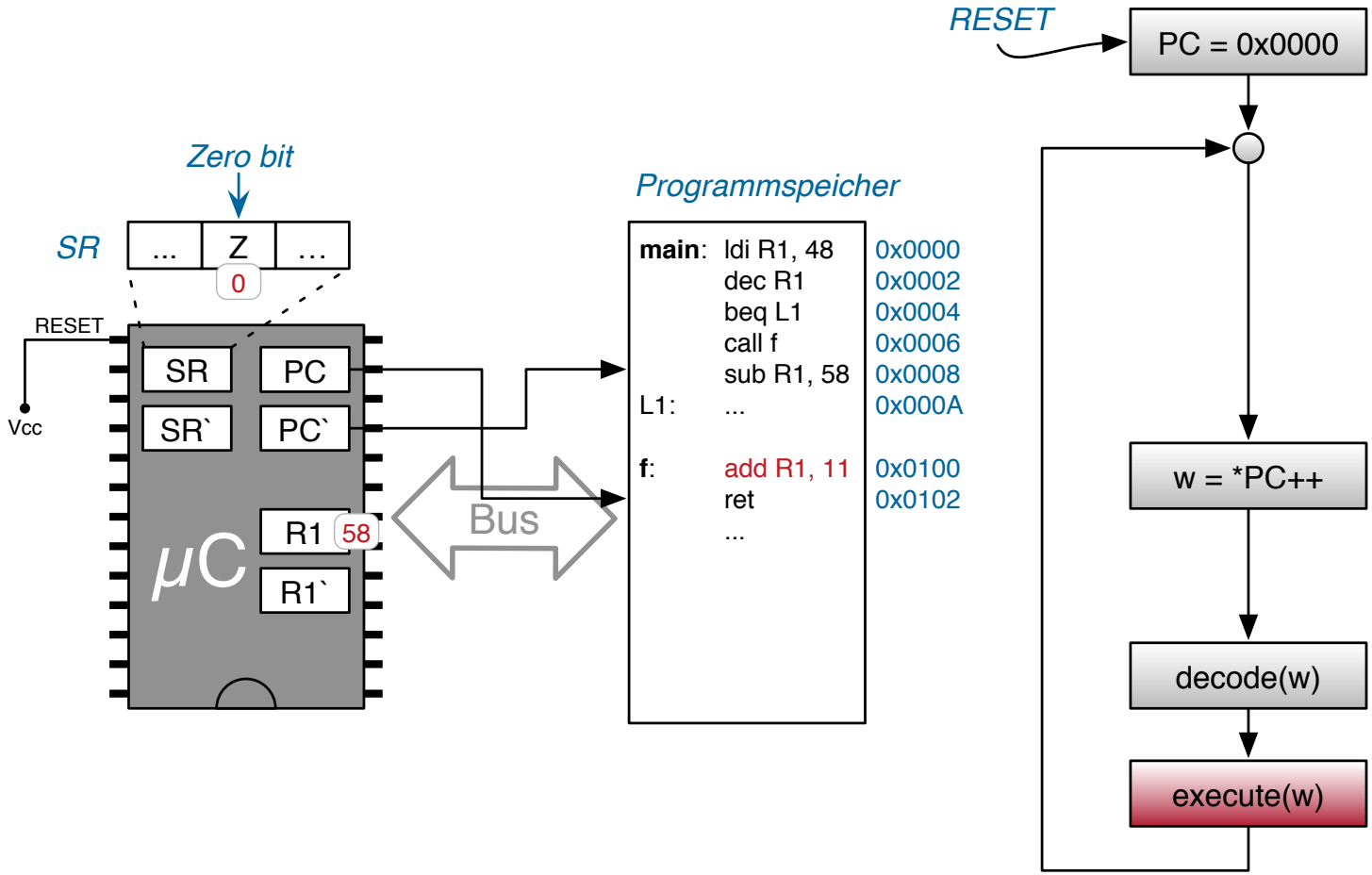


w: **dec** <R>
R –= 1
if(R == 0) Z = 1
else Z = 0

w: **beq** <lab>
if(Z) PC = lab

w: **call** <func>
PC` = PC
PC = func

Wie arbeitet ein Prozessor?

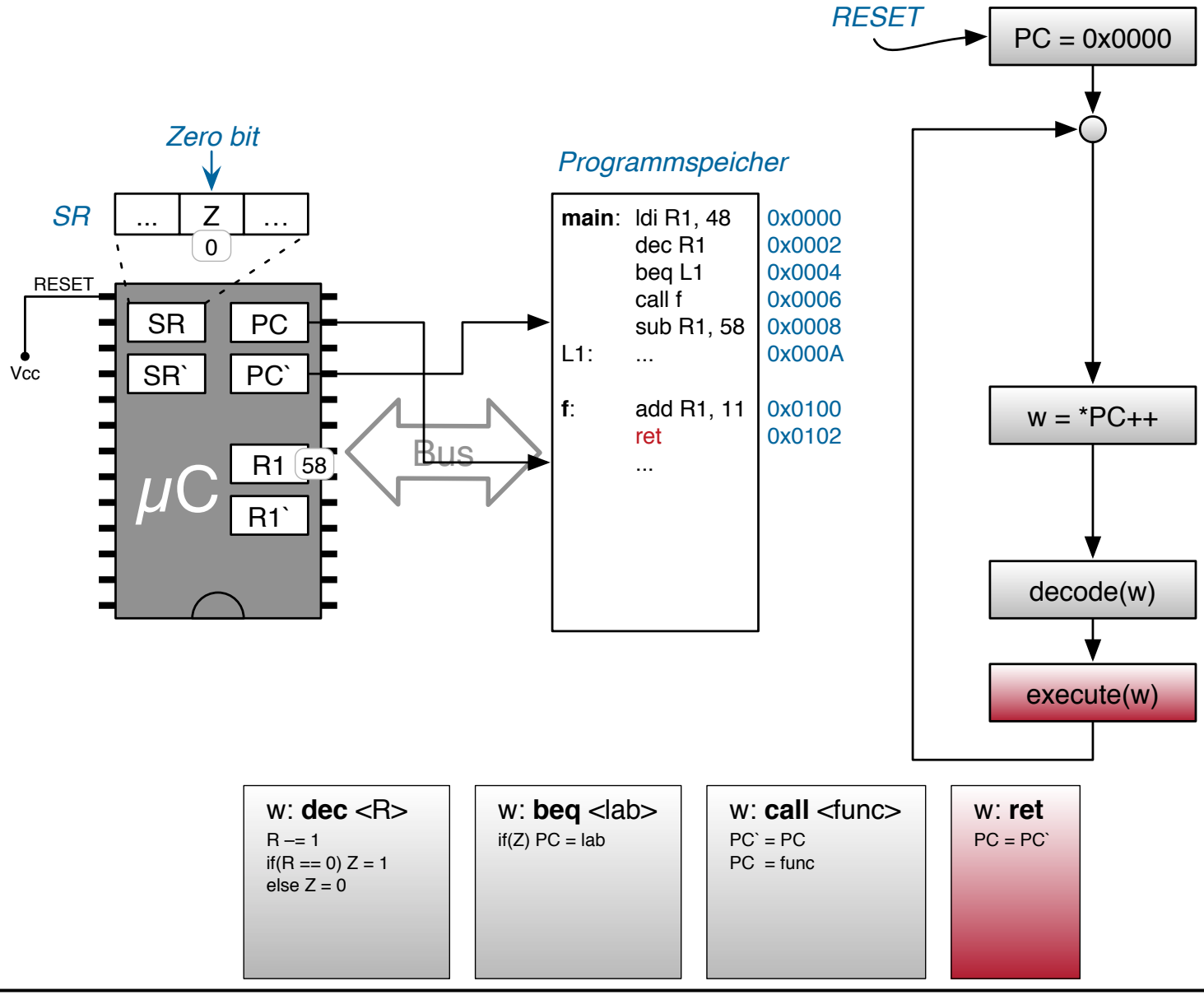


w: **dec** <R>
R –= 1
if(R == 0) Z = 1
else Z = 0

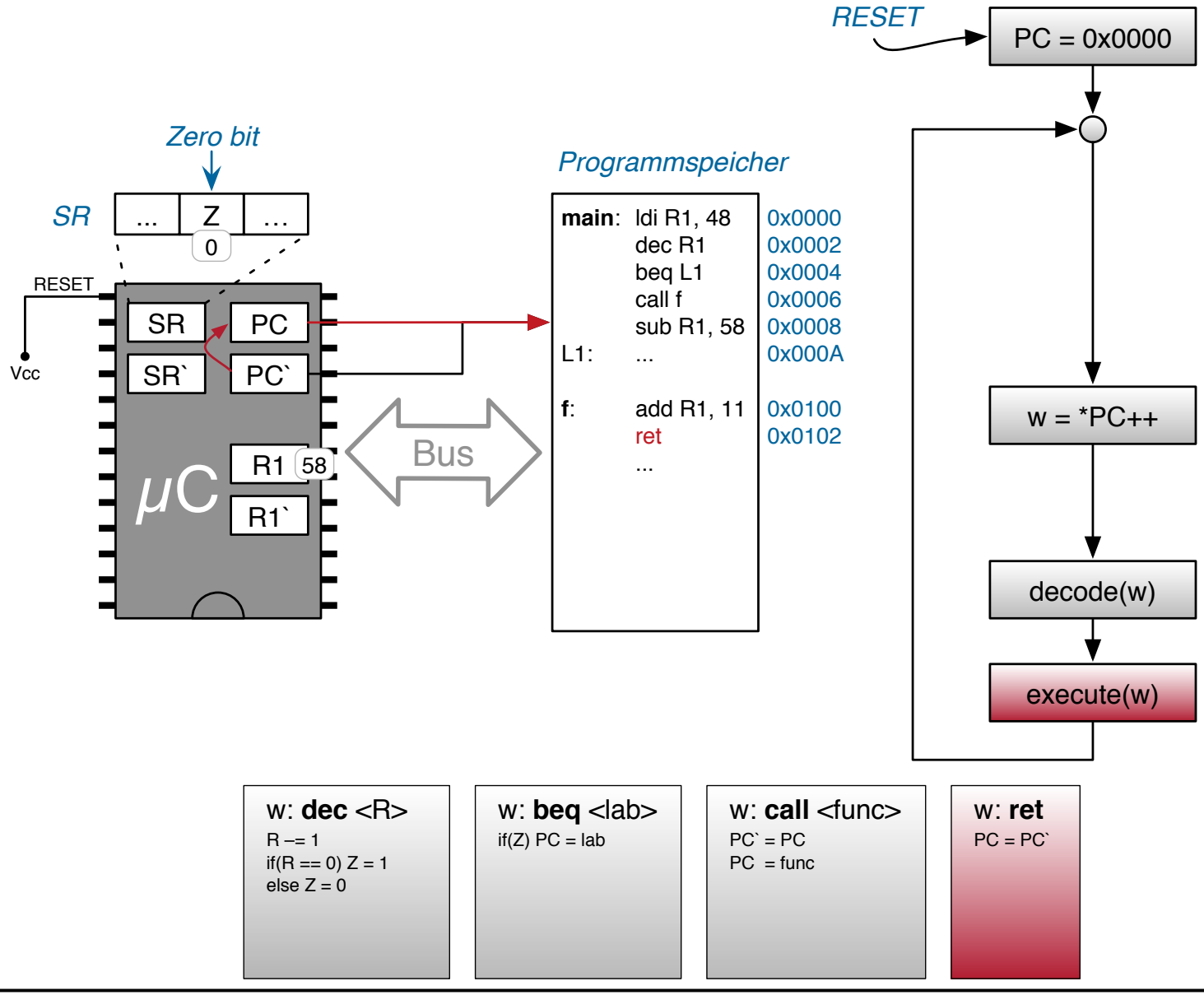
w: **beq** <lab>
if(Z) PC = lab

w: **call** <func>
PC` = PC
PC = func

Wie arbeitet ein Prozessor?



Wie arbeitet ein Prozessor?



Wie arbeitet ein Prozessor?