Evolution of Map

From Go-ing Simple to Map-tastic

George Sereda | 12.03.2025

AGENDA

- Introduction
- First map
- Classic map
- Swiss Tables map
- Questions

Meet your speaker

George Sereda

Principal Golang Engineer at Mediahuis

DUMPERT MEDIAHUIS

Naive solution

map [user] = views

i	user
0	
1	
2	
3	

i	views
0	
1	
2	
3	

Naive solution

map [user] = views

i	user
0	alice
1	bob
2	alfons
3	

i	views
0	11
1	4
2	7
3	

Naive solution

map [user] = views

map[“bob”] += 1

i	key
0	alice
1	bob
2	alfons
3	

i	value
0	11
1	4
2	7
3	

Linear **scan**

map [user] = views

map[“bob”] += 1

equal (“bob”, “alice”) => false

i	key
0	alice
1	bob
2	alfons
3	

i	value
0	11
1	4
2	7
3	

Linear **scan**

map [user] = views

```
map[“bob”] += 1

equal (“bob”, “bob”) => true
```

i	key
0	alice
1	bob
2	alfons
3	

i	value
0	11
1	4
2	7
3	

Linear **scan**

map [user] = views

map[“bob”] += 1

equal (“bob”, “bob”) => true

index = 1

value[index] = 4 + 1

i	key
0	alice
1	bob
2	alfons
3	

i	value
0	11
1	5
2	7
3	

Weekly snapshot 2009-12-09

First public release, included map

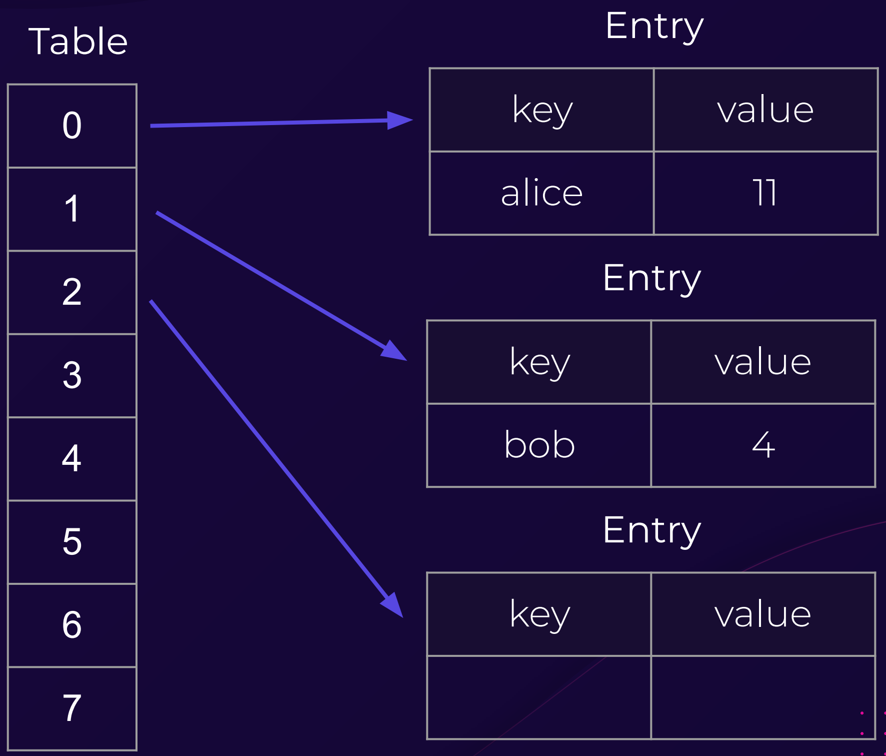
First golang **Map**

map [user] = views

Entry

key	value
alice	11

First golang **Map**



Table

0
1
2
3
4
5
6
7

Entry

key	value
alice	11

Entry

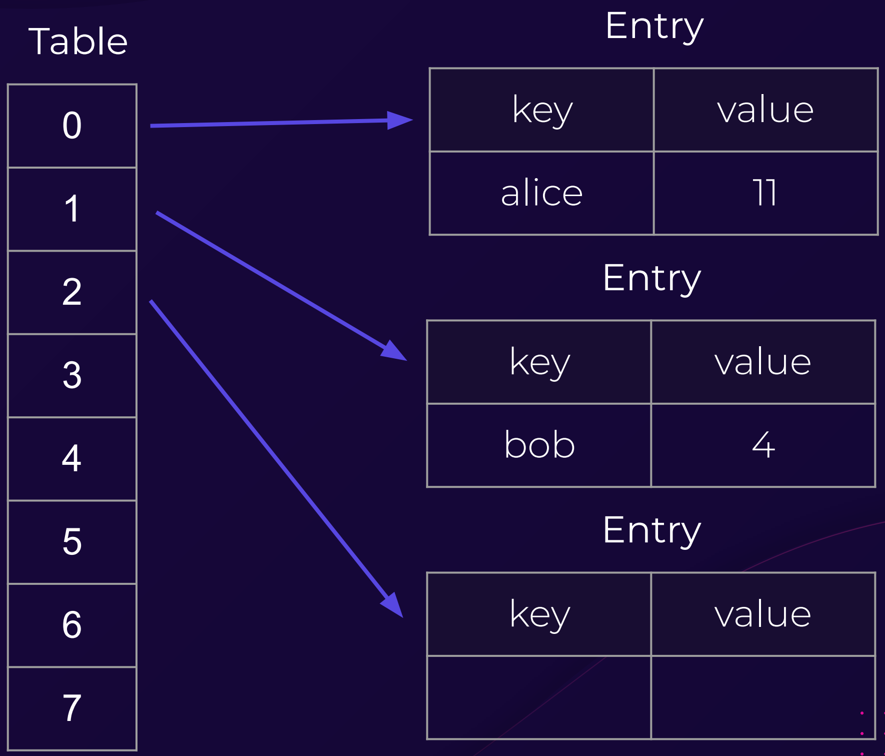
key	value
bob	4

Entry

key	value

First golang **Map**

map [billy] = 1



Table

0
1
2
3
4
5
6
7

Entry

key	value
alice	11

Entry

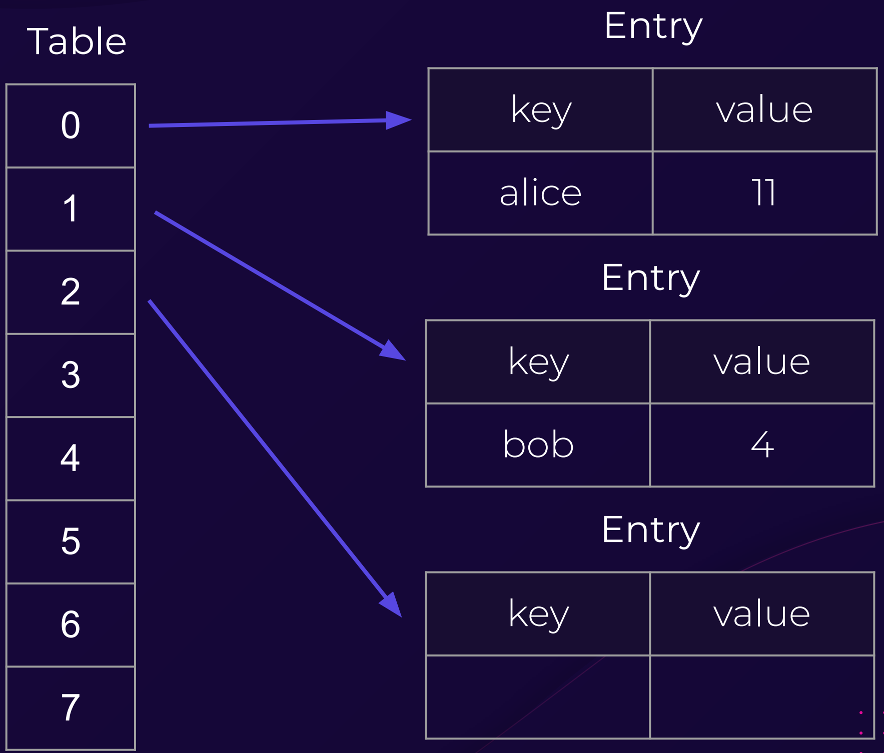
key	value
bob	4

Entry

key	value

First golang **Map**

key: billy



Table

0
1
2
3
4
5
6
7

Entry

key	value
alice	11

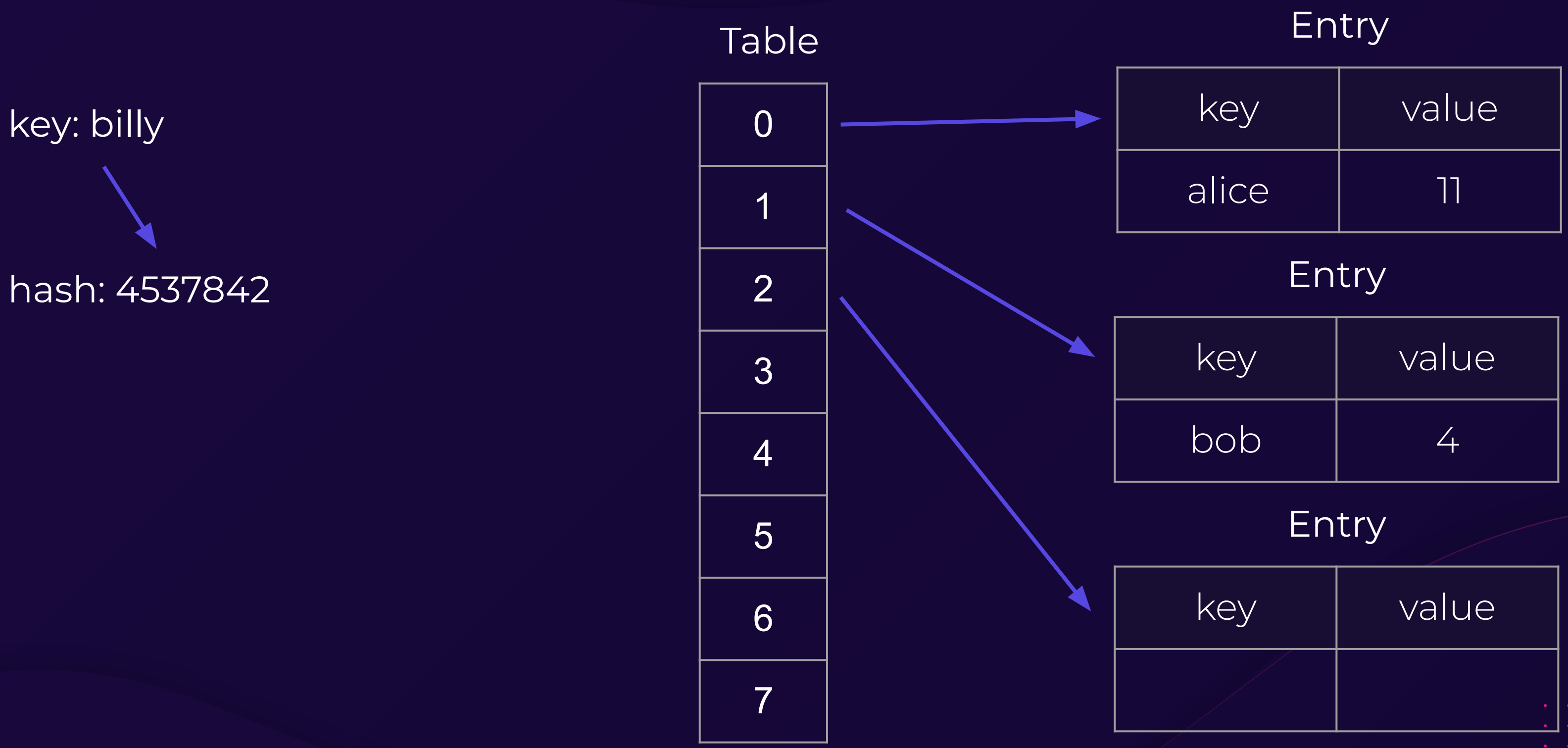
Entry

key	value
bob	4

Entry

key	value

First golang **Map**



key: billy

hash: 4537842

Table

0
1
2
3
4
5
6
7

Entry

key	value
alice	11

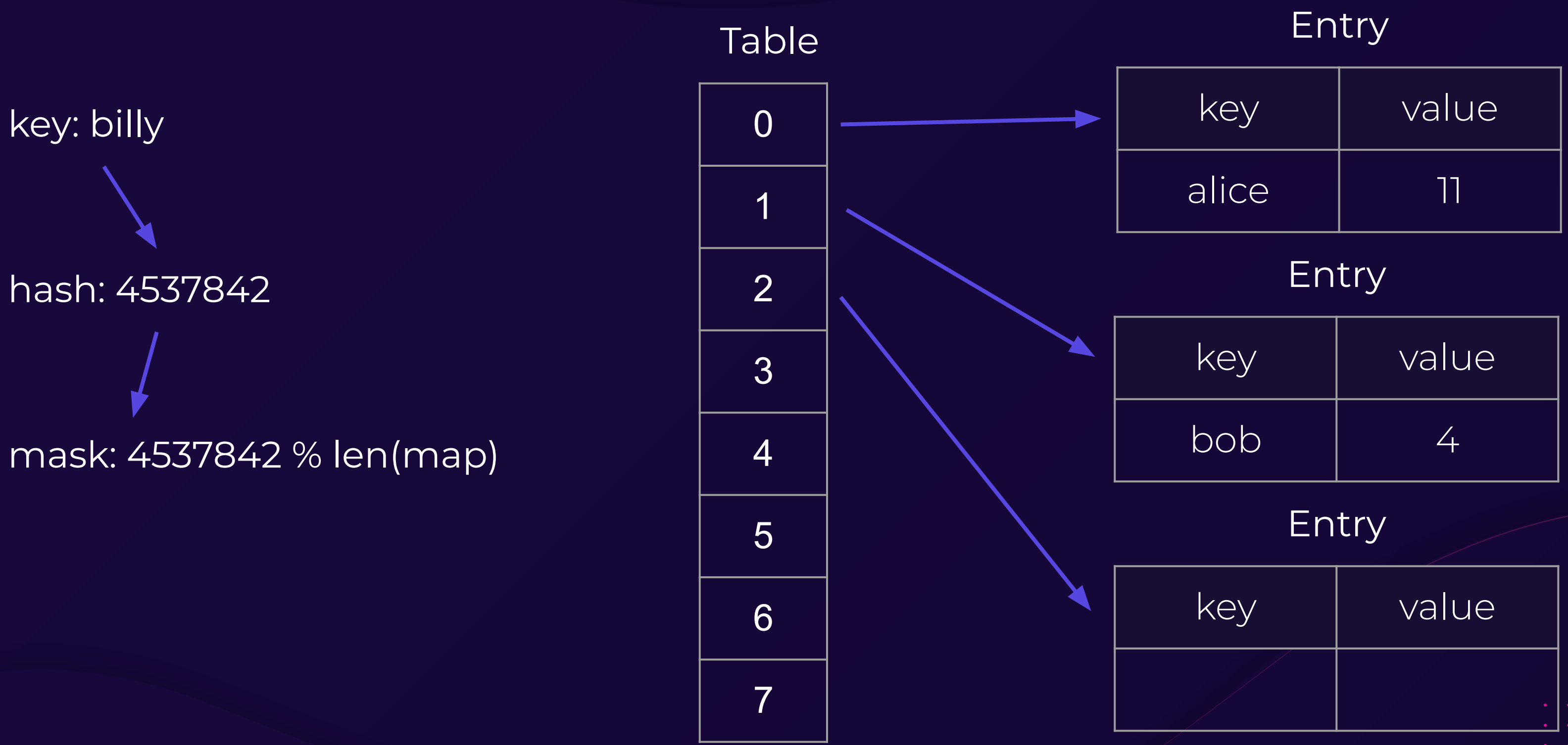
Entry

key	value
bob	4

Entry

key	value

First golang **Map**



key: billy

hash: 4537842

mask: 4537842 % len(map)

Table

0
1
2
3
4
5
6
7

Entry

key	value
alice	11

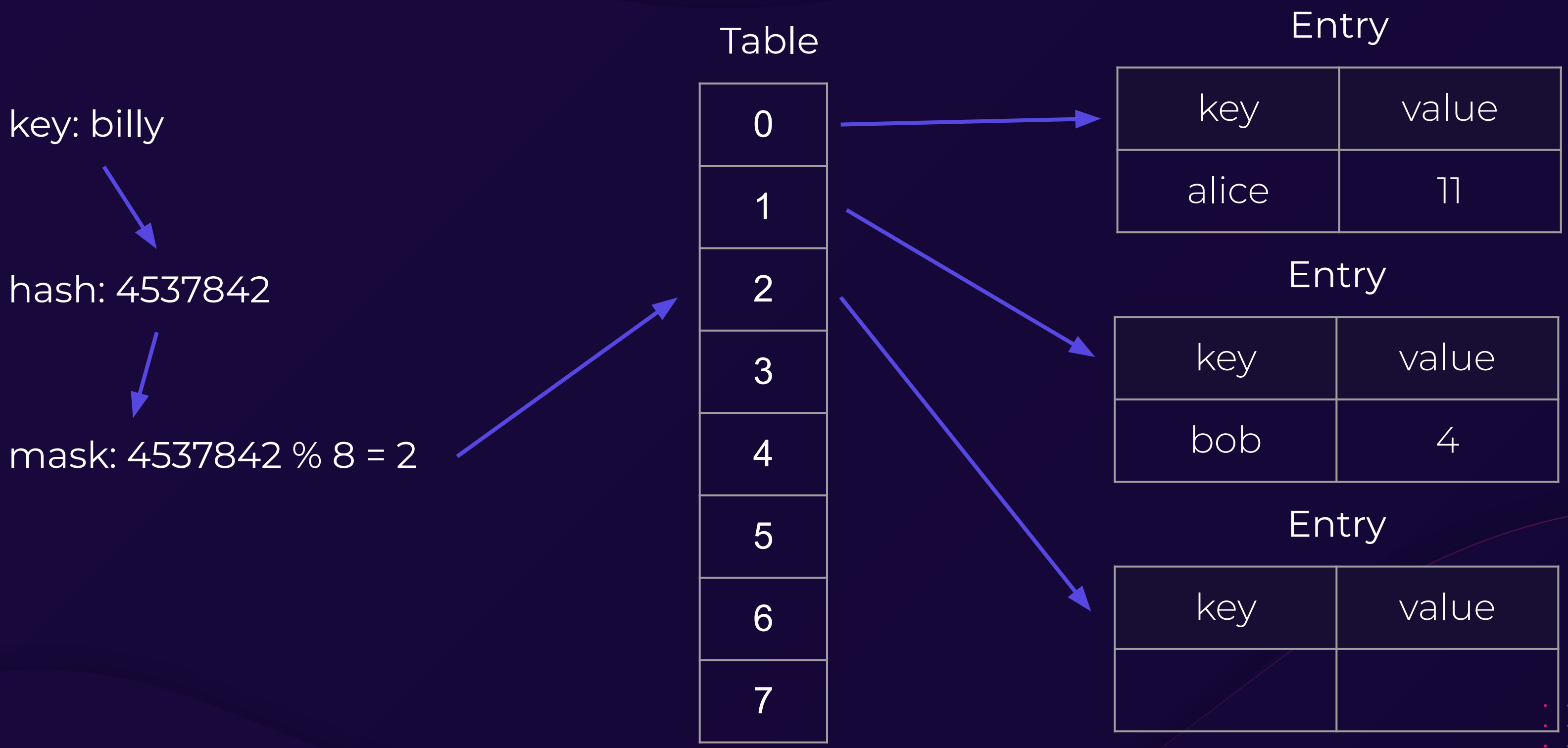
Entry

key	value
bob	4

Entry

key	value

First golang **Map**



key: billy

hash: 4537842

mask: 4537842 % 8 = 2

Table

0
1
2
3
4
5
6
7

Entry

key	value
alice	11

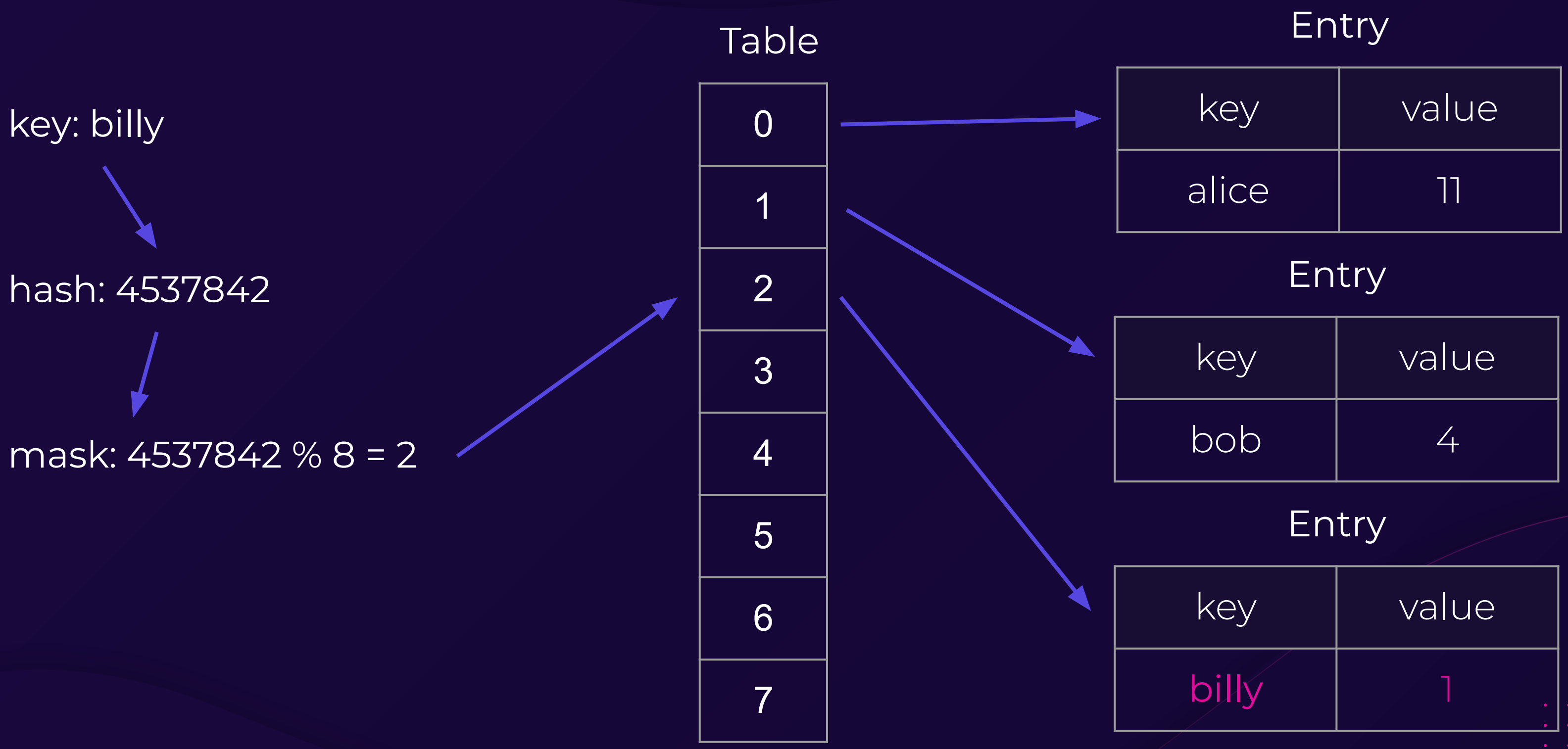
Entry

key	value
bob	4

Entry

key	value

First golang **Map**



key: billy

hash: 4537842

mask: 4537842 % 8 = 2

Table

0
1
2
3
4
5
6
7

Entry

key	value
alice	11

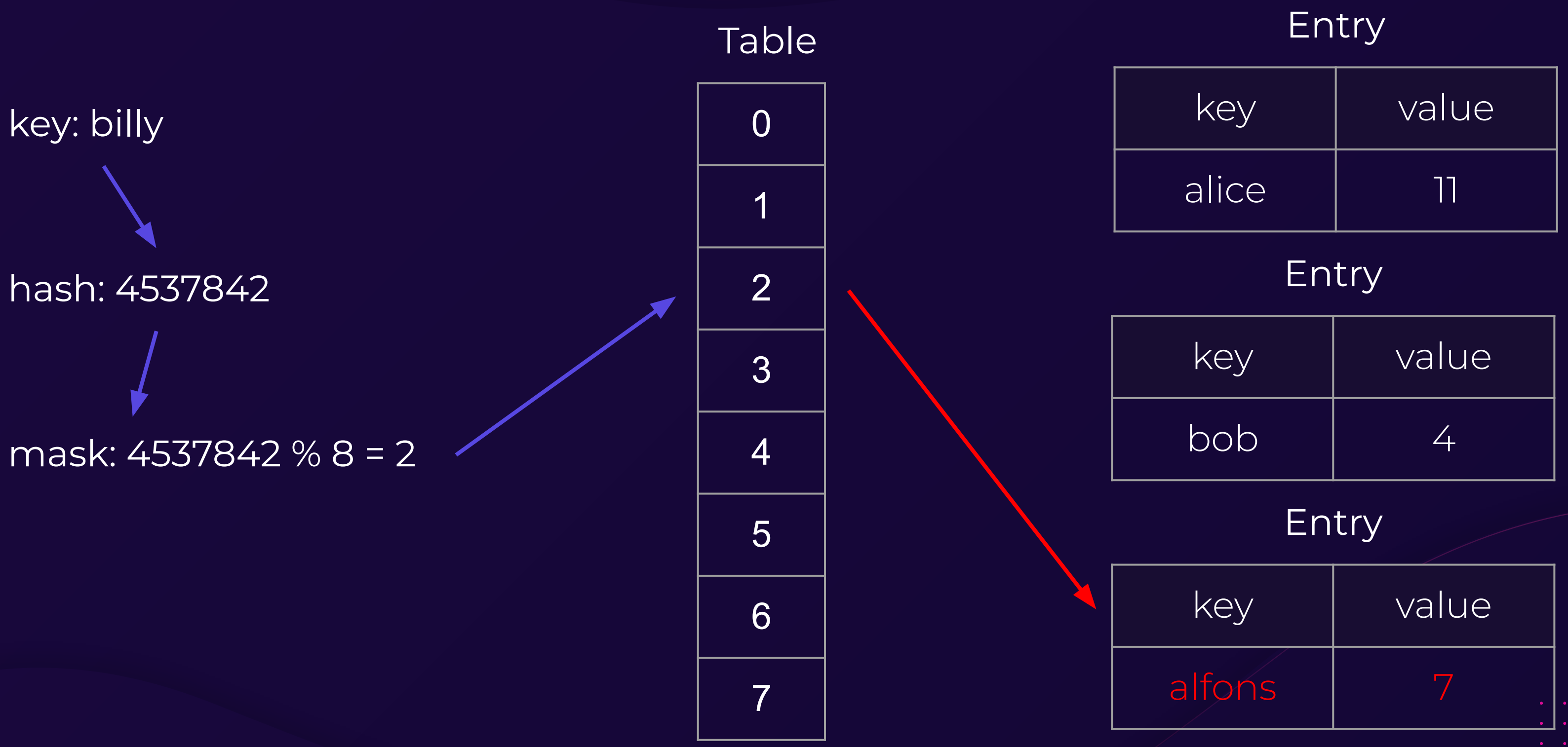
Entry

key	value
bob	4

Entry

key	value
billy	1

Collusion



key: billy

hash: 4537842

mask: 4537842 % 8 = 2

Table

0
1
2
3
4
5
6
7

Entry

key	value
alice	11

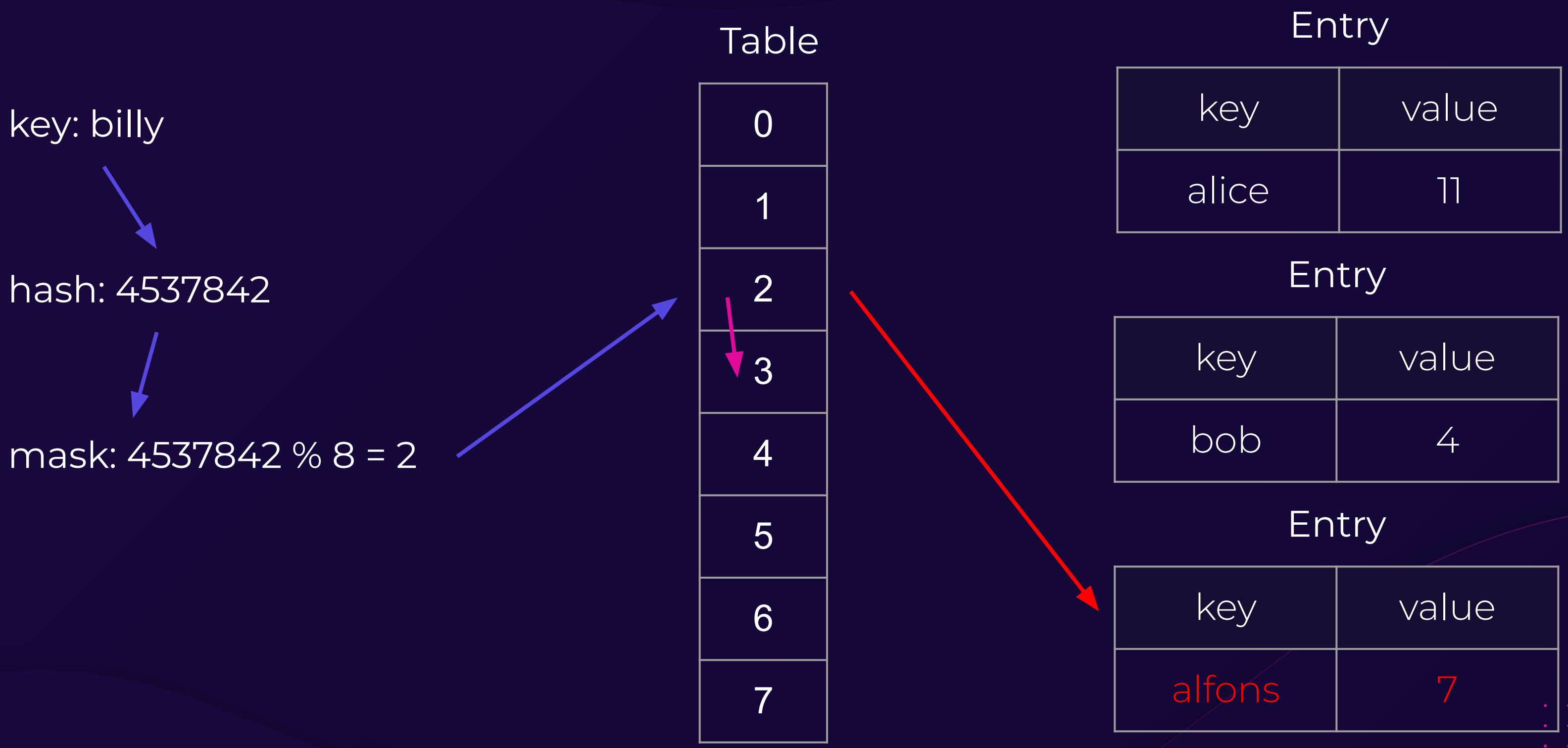
Entry

key	value
bob	4

Entry

key	value
alfons	7

Linear probing



key: billy

hash: 4537842

mask: 4537842 % 8 = 2

Table

0
1
2
3
4
5
6
7

Entry

key	value
alice	11

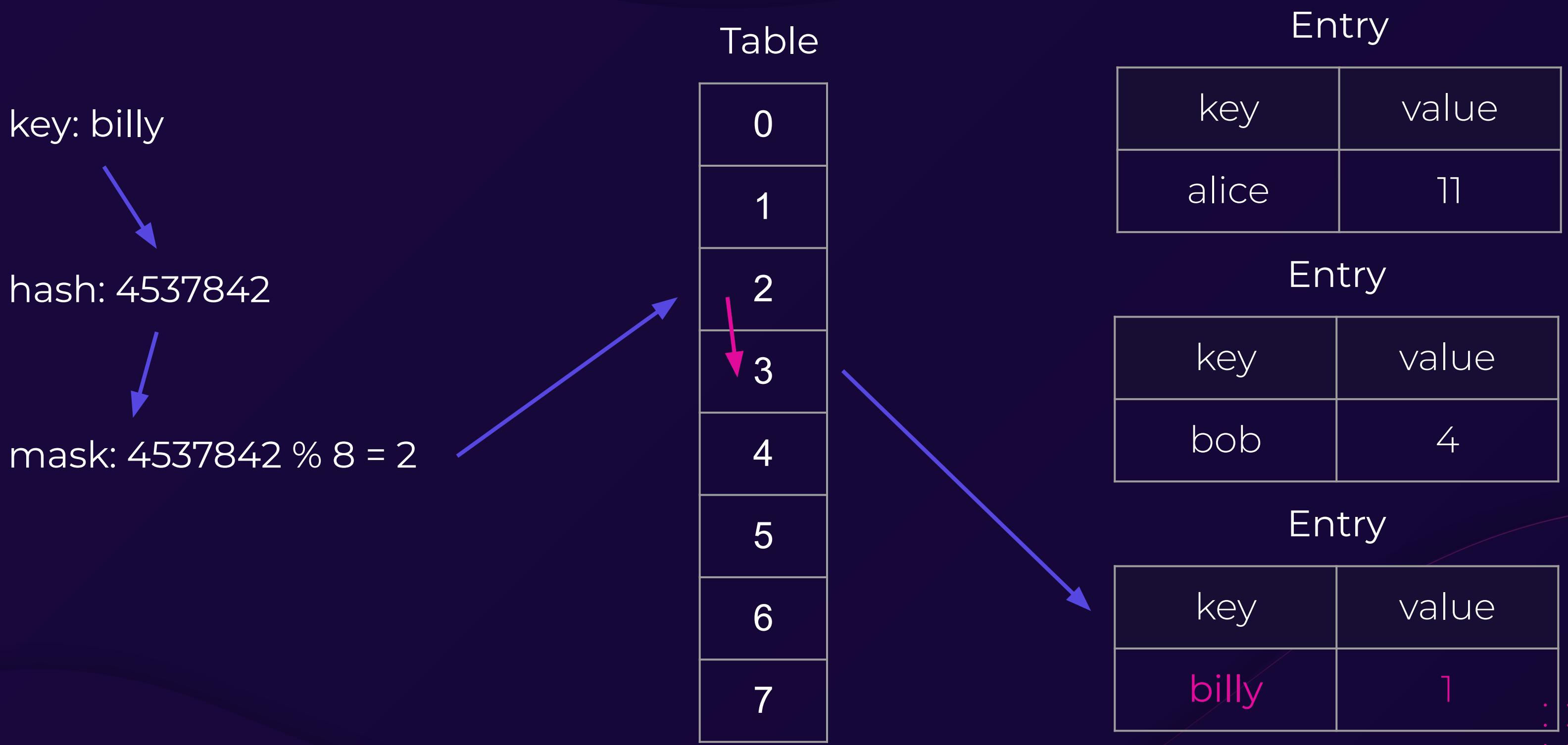
Entry

key	value
bob	4

Entry

key	value
alfons	7

Linear probing



key: billy

hash: 4537842

mask: 4537842 % 8 = 2

Table

0
1
2
3
4
5
6
7

Entry

key	value
alice	11

Entry

key	value
bob	4

Entry

key	value
billy	1

Go 1.1

2013-05-13

Traditional hashmap

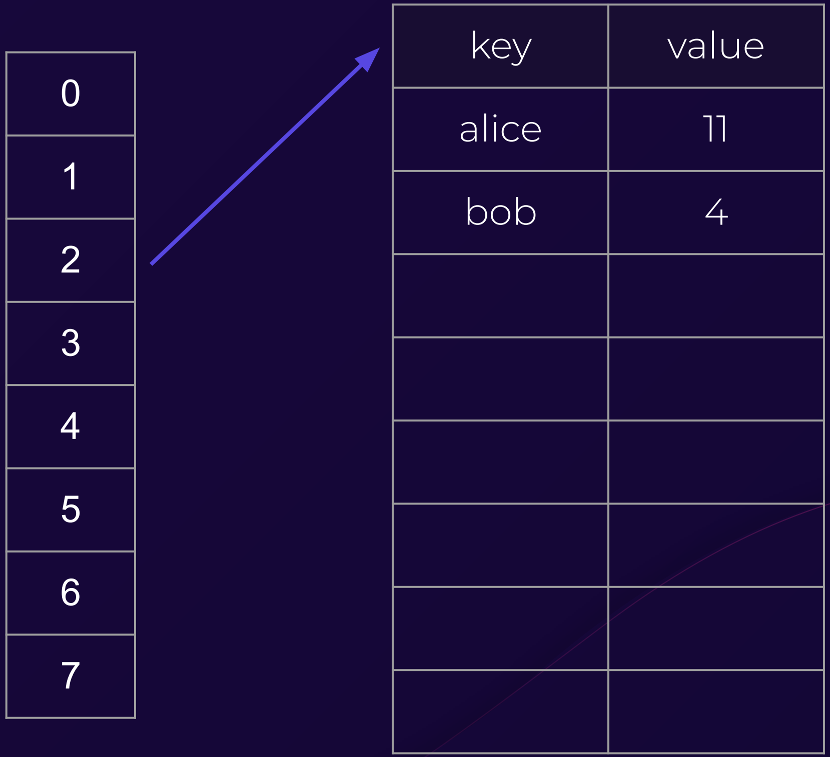
Old, but gold

Bucket

always 8 elements

key	value
alice	11
bob	4

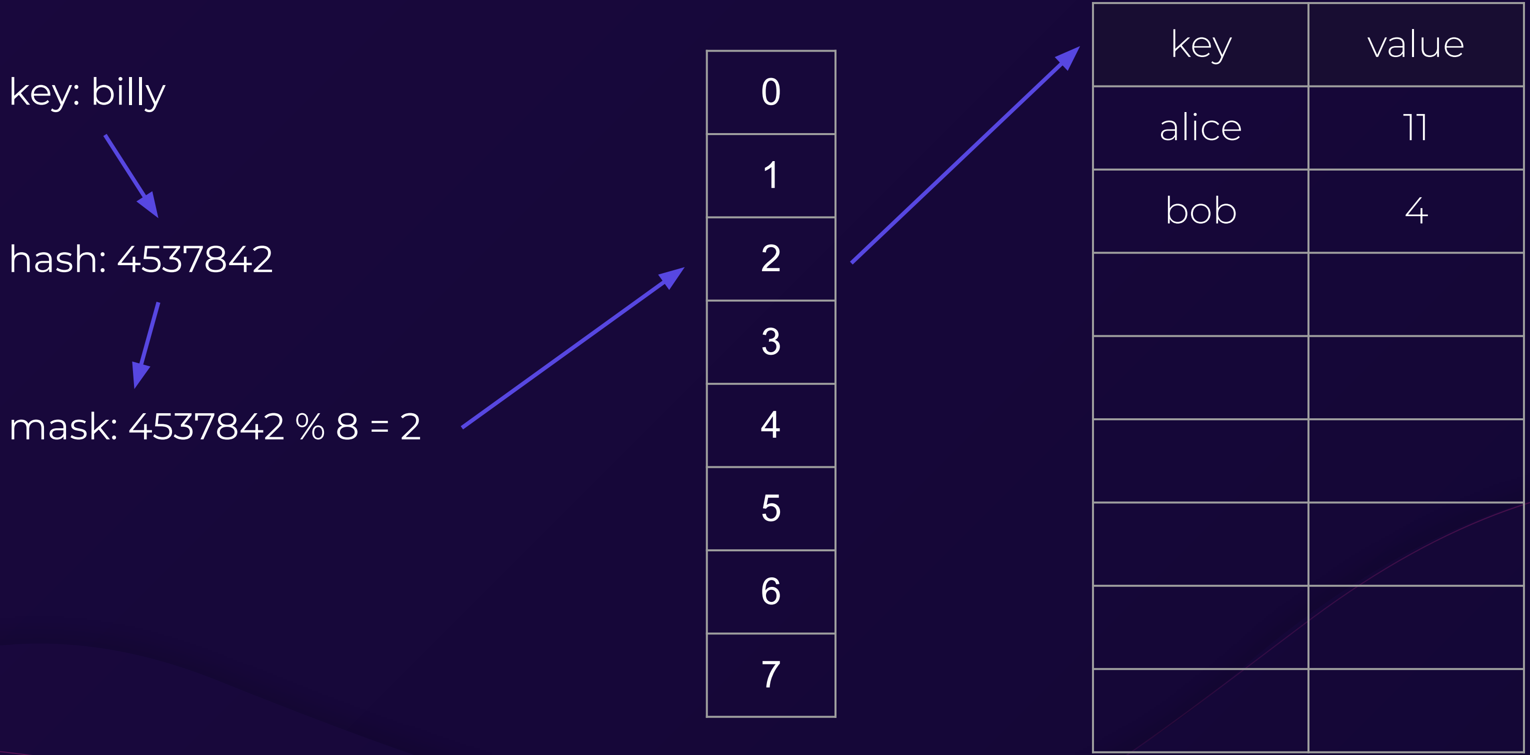
Classic **Hashmap**



0
1
2
3
4
5
6
7

key	value
alice	11
bob	4

Classic **Hashmap**



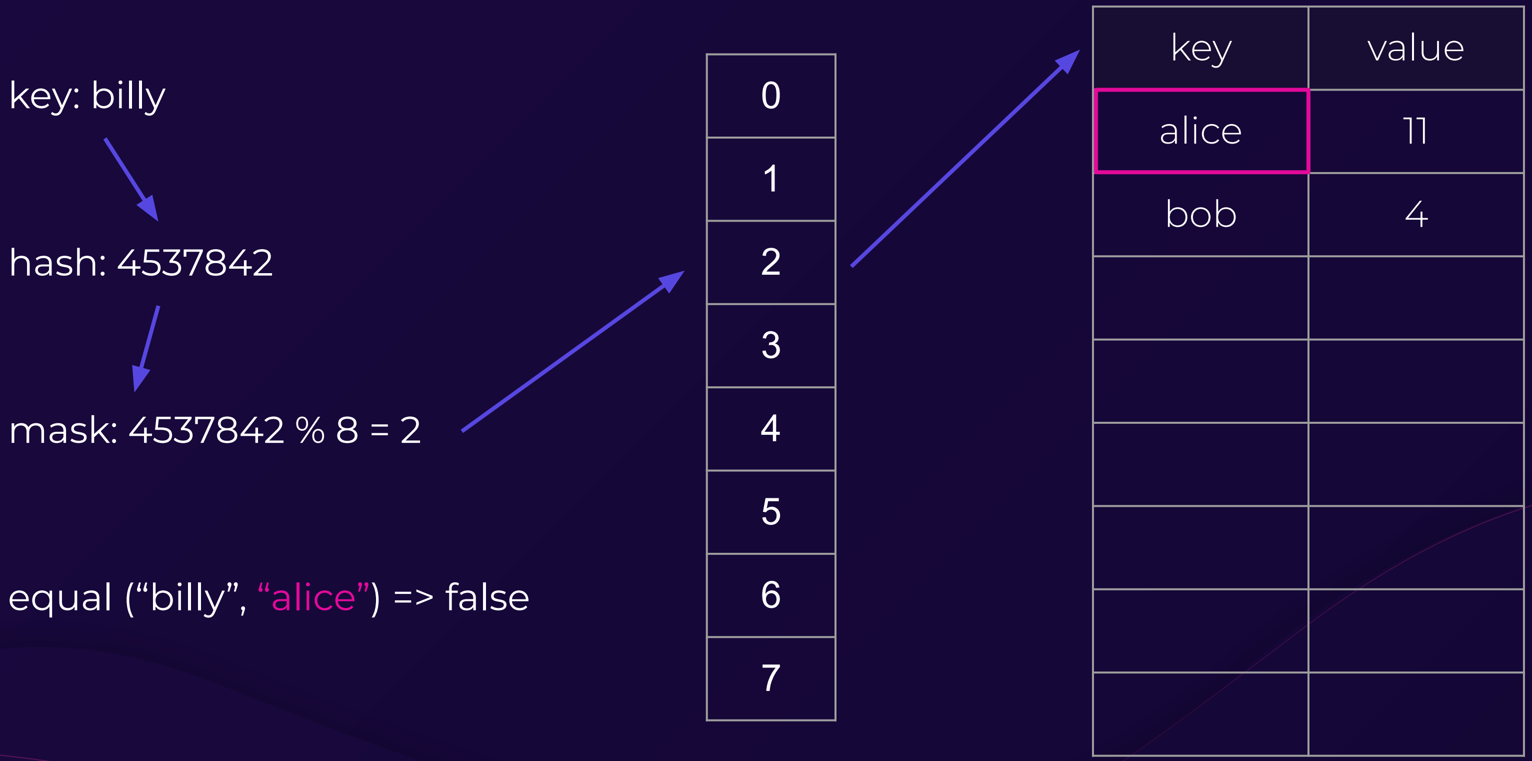
key: billy

hash: 4537842

mask: 4537842 % 8 = 2

key	value
alice	11
bob	4

Classic **Hashmap**



key: billy

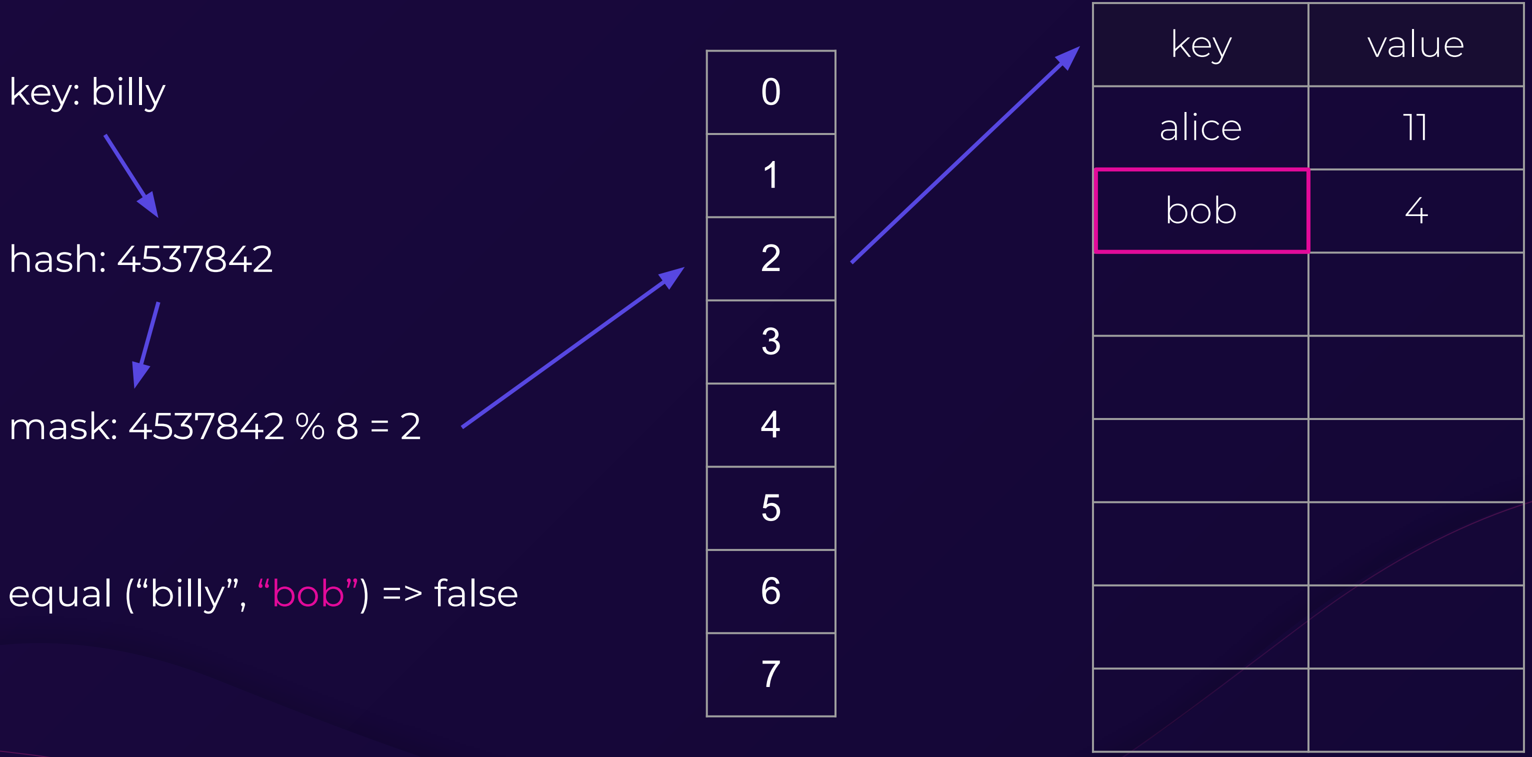
hash: 4537842

mask: 4537842 % 8 = 2

equal (“billy”, “alice”) => false

key	value
alice	11
bob	4

Classic **Hashmap**



key: billy

hash: 4537842

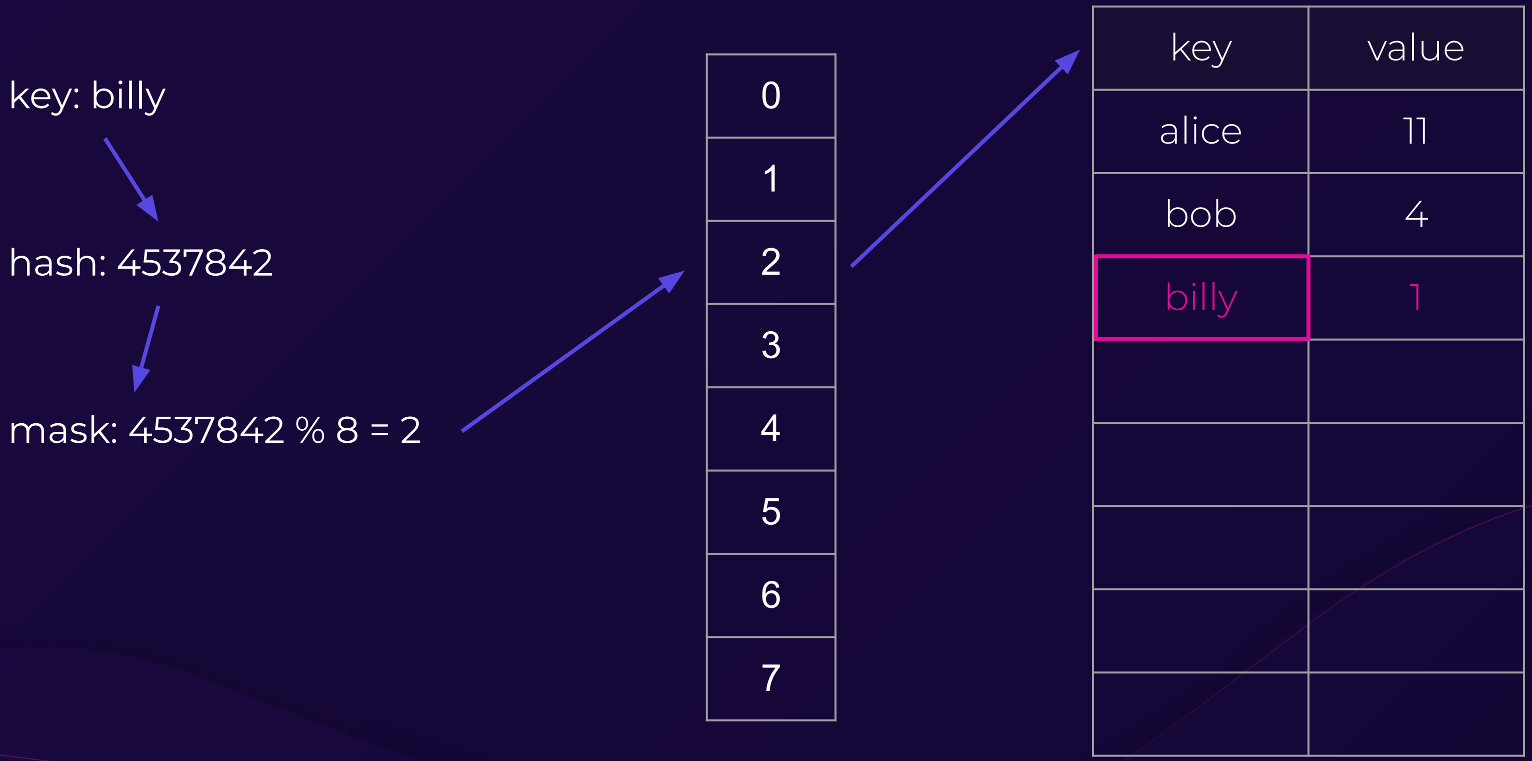
mask: 4537842 % 8 = 2

equal (“billy”, “bob”) => false

0
1
2
3
4
5
6
7

key	value
alice	11
bob	4

Classic **Hashmap**



key: billy

hash: 4537842

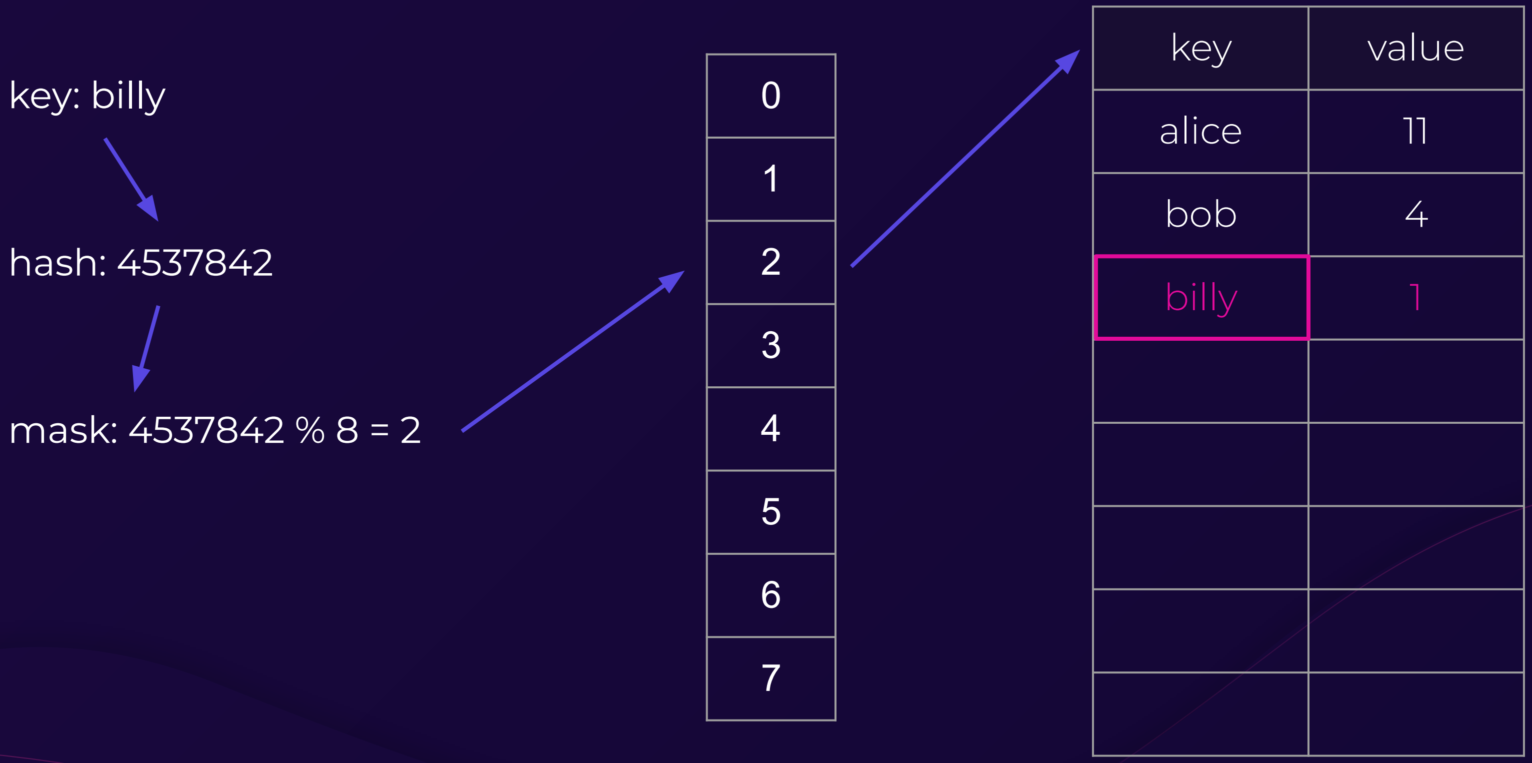
mask: 4537842 % 8 = 2

0
1
2
3
4
5
6
7

key	value
alice	11
bob	4
billy	1

Still **Linear Scan**

max 8 comparisons



key: billy

hash: 4537842

mask: 4537842 % 8 = 2

0
1
2
3
4
5
6
7

key	value
alice	11
bob	4
billy	1

20-50%

Operation optimization

Go 1.24

2025-02-11

🔥 SWISS tables 🔥

designed in 2017

Swiss tables

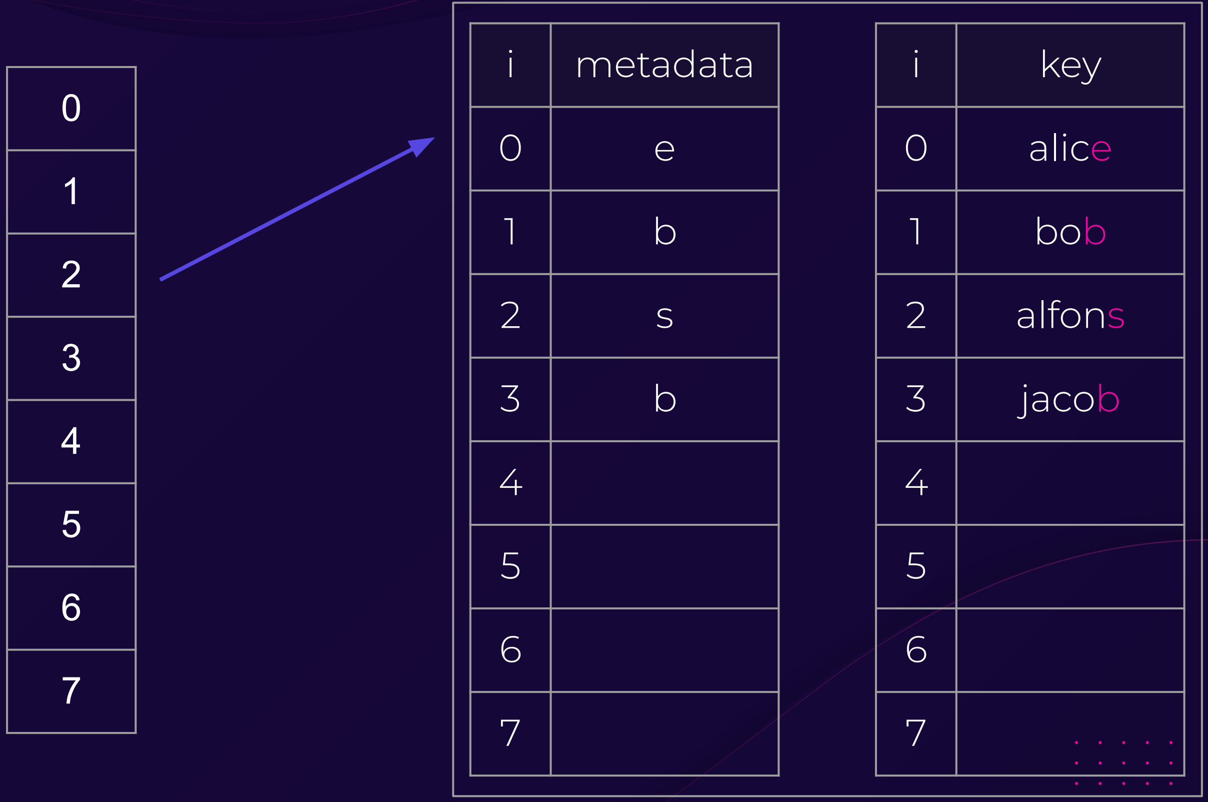
i	key
0	alice
1	bob
2	alfons
3	jacob
4	
5	
6	
7	

Swiss tables

i	metadata
0	e
1	b
2	s
3	b
4	
5	
6	
7	

i	key
0	alice
1	bob
2	alfons
3	jacob
4	
5	
6	
7	

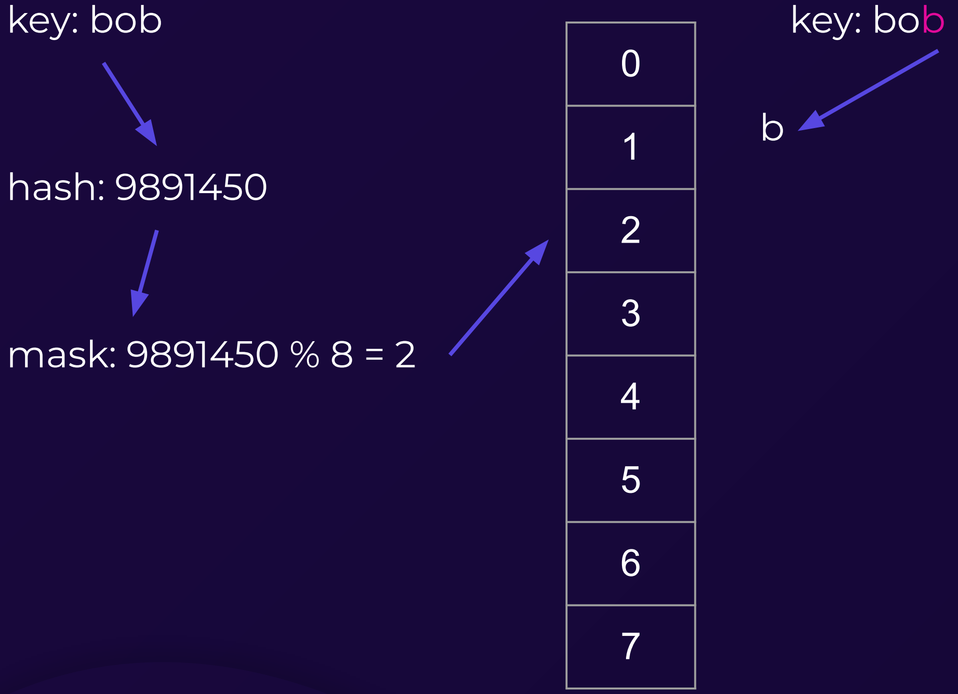
Swiss tables



i	metadata
0	e
1	b
2	s
3	b
4	
5	
6	
7	

i	key
0	alice
1	bob
2	alfons
3	jacob
4	
5	
6	
7	

Swiss tables



key: bob

hash: 9891450

mask: 9891450 % 8 = 2

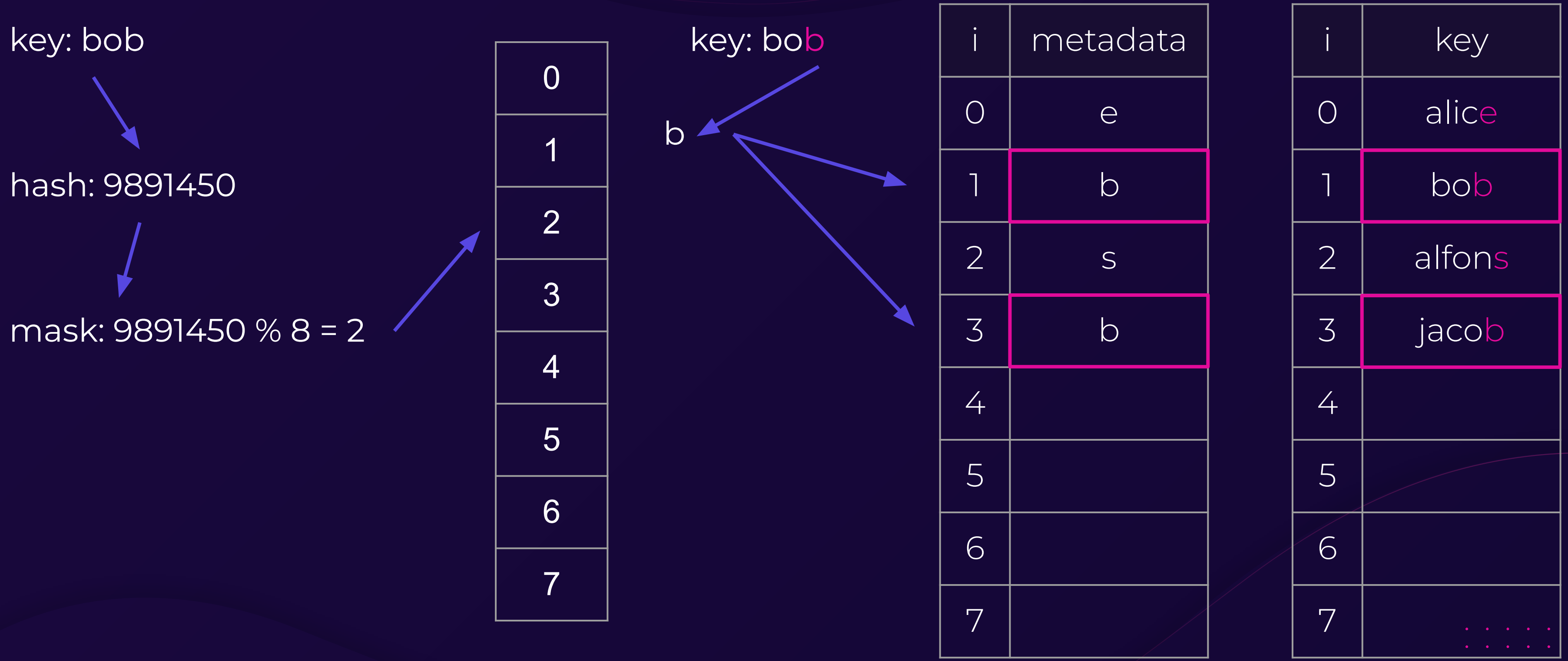
key: bob

b

i	metadata
0	e
1	b
2	s
3	b
4	
5	
6	
7	

i	key
0	alice
1	bob
2	alfons
3	jacob
4	
5	
6	
7	

Swiss tables



key: bob

hash: 9891450

mask: 9891450 % 8 = 2

0
1
2
3
4
5
6
7

key: bob

b

i	metadata
0	e
1	b
2	s
3	b
4	
5	
6	
7	

i	key
0	alice
1	bob
2	alfons
3	jacob
4	
5	
6	
7	

Swiss tables

key: bob

equal (“bob”, “bob”) => true

equal (“bob”, “jacob”) => false

i	metadata
0	e
1	b
2	s
3	b
4	
5	
6	
7	

i	key
0	alice
1	bob
2	alfons
3	jacob
4	
5	
6	
7	

Control word & Group

i	metadata
0	e
1	b
2	s
3	b
4	
5	
6	
7	

i	key
0	alice
1	bob
2	alfons
3	jacob
4	
5	
6	
7	

Control word & Group

	Group 0								Group 1							
Slot	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Key	56	32	21						78							

	64-bit control word 0								64-bit control word 1							
Slot	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
h2	23	89	50						47							

Control word & Group

	Group 0								Group 1							
Slot	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Key	56	32	21						78							

key: 32 → h1: 23894789

	64-bit control word 0								64-bit control word 1							
Slot	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
h2	23	89	50						47							

Control word & Group

	Group 0								Group 1							
Slot	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Key	56	32	21						78							

key: 32 → h1: 238947889 → h2: 89

	64-bit control word 0								64-bit control word 1							
Slot	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
h2	23	89	50						47							

Control word & Group

	64-bit control word 0							
Slot	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
h2	23	89	50					

Test word	89	89	89	89	89	89	89	89
Comparison	==	==	==	==	==	==	==	==
Control word	23	89	50	-	-	-	-	-
Result	0	1	0	0	0	0	0	0

SIMD

single instruction,
multiple data

60%

Operation optimization

Grow

Group-by-group

Thanks a lot!

George Sereda
12.03.2025

Additional reading:

01.

Go swiss table docu

https://go.dev/blog/swisstable

02.

SIMD

https://en.wikipedia.org/wiki/Single_instruction,_multiple_data

03.

Dave Chaney: old maps

https://dave.cheney.net/2018/05/29/how-the-go-runtime-implements-maps-efficiently-without-generics

04.

Abseil Swiss table

https://abseil.io/about/design/swisstables