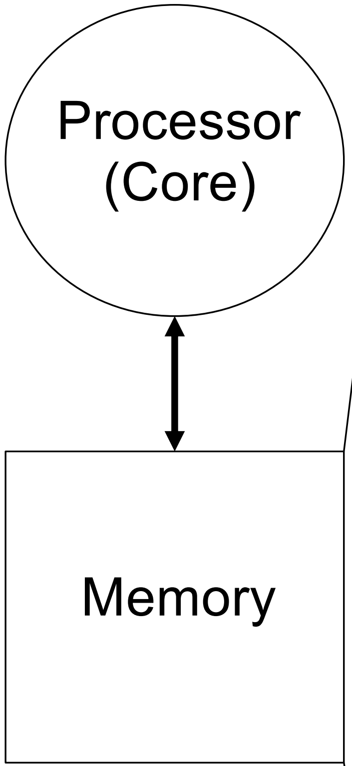
Optional Lecture 2

Basic Memory Hierarchy Concepts

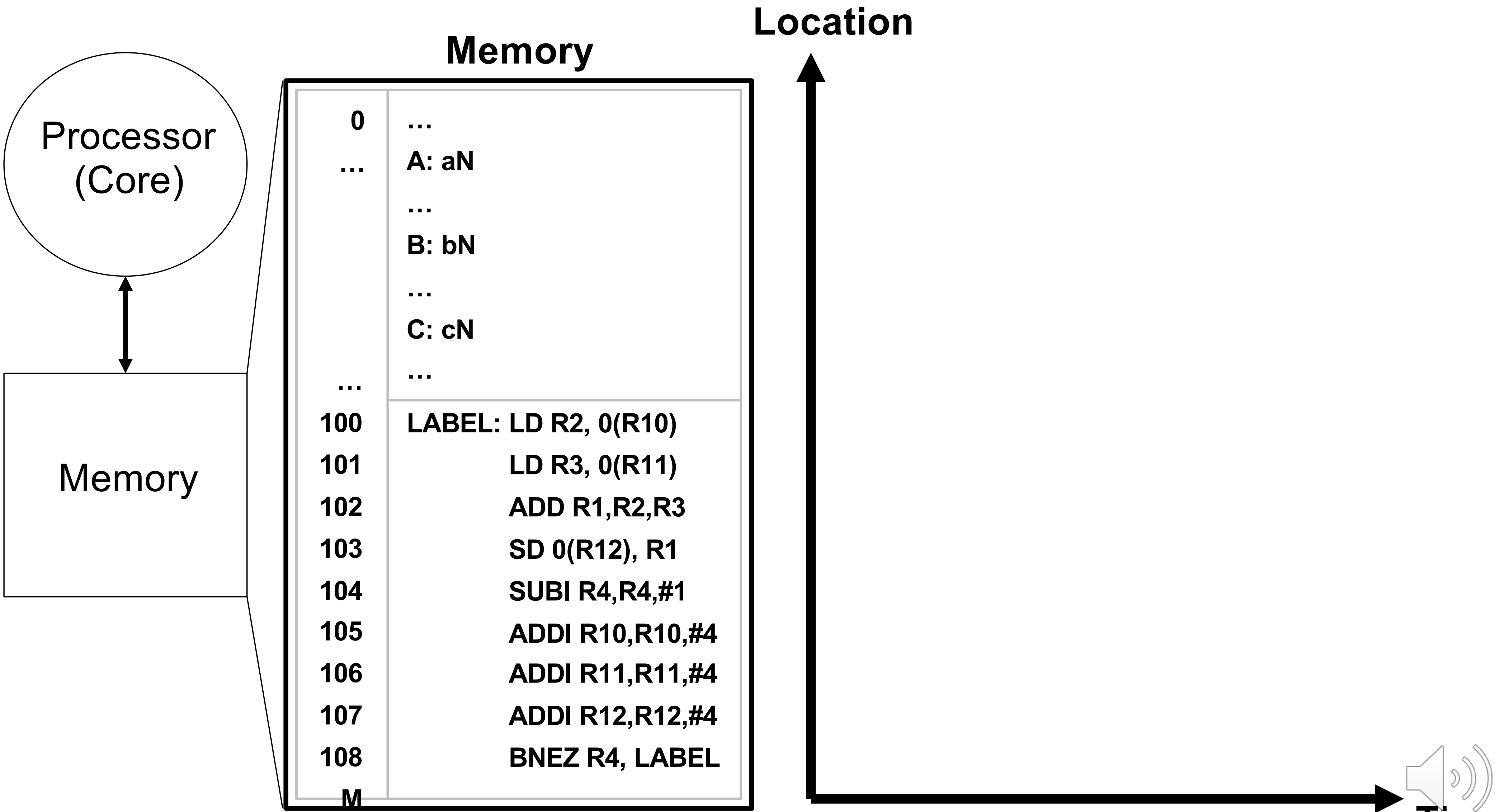
- The pyramid of memory levels (Ch. 4.2)
- Cache hierarchy (parts of Ch. 4.3)
  - ✓ Cache mapping and organization (4.3.1)
  - ✓ Replacement policies (4.3.2)
  - ✓ Write policies (4.3.3)
  - ✓ Cache hierarchy performance (4.3.4)

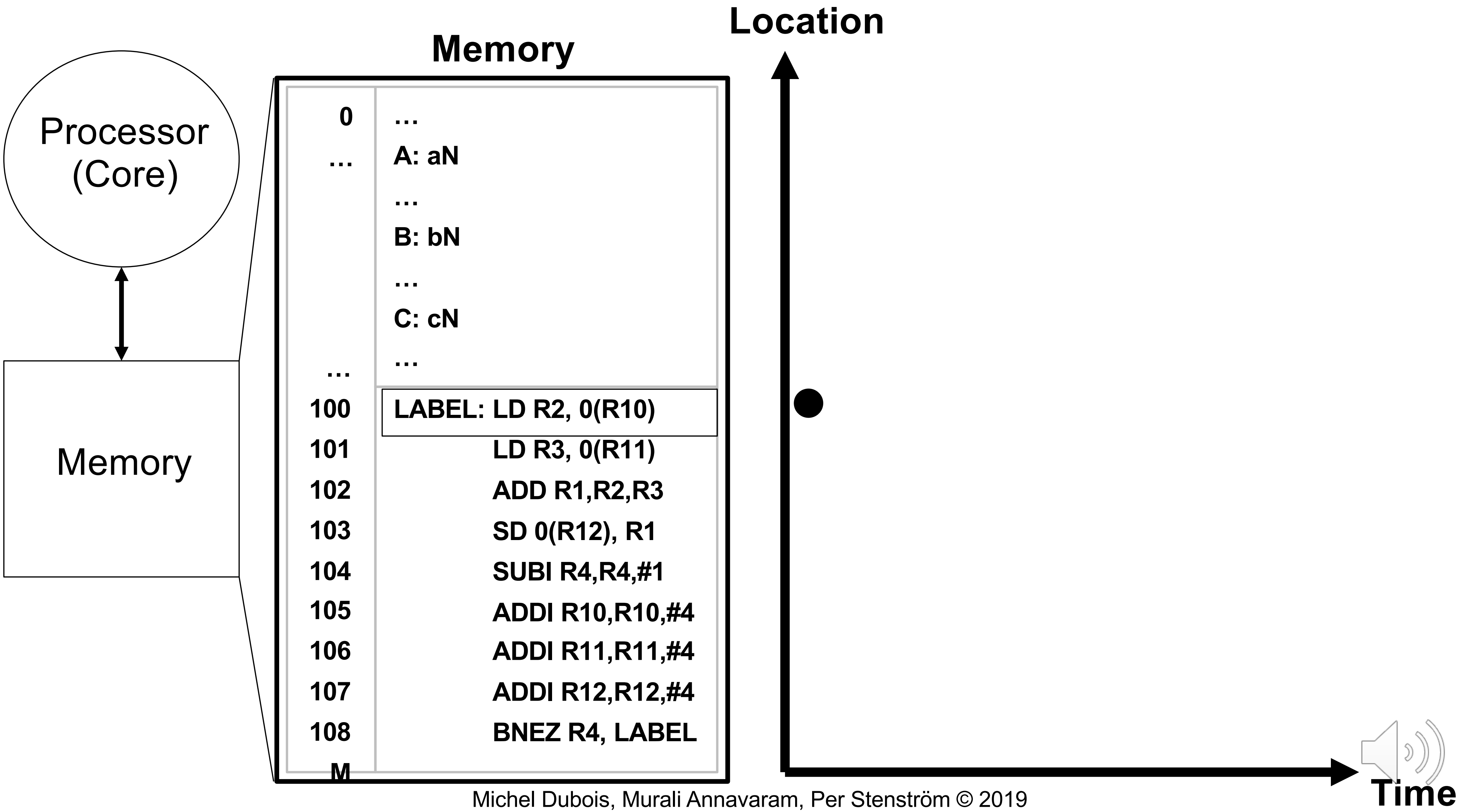
The Locality Principle
(Ch 4.2)

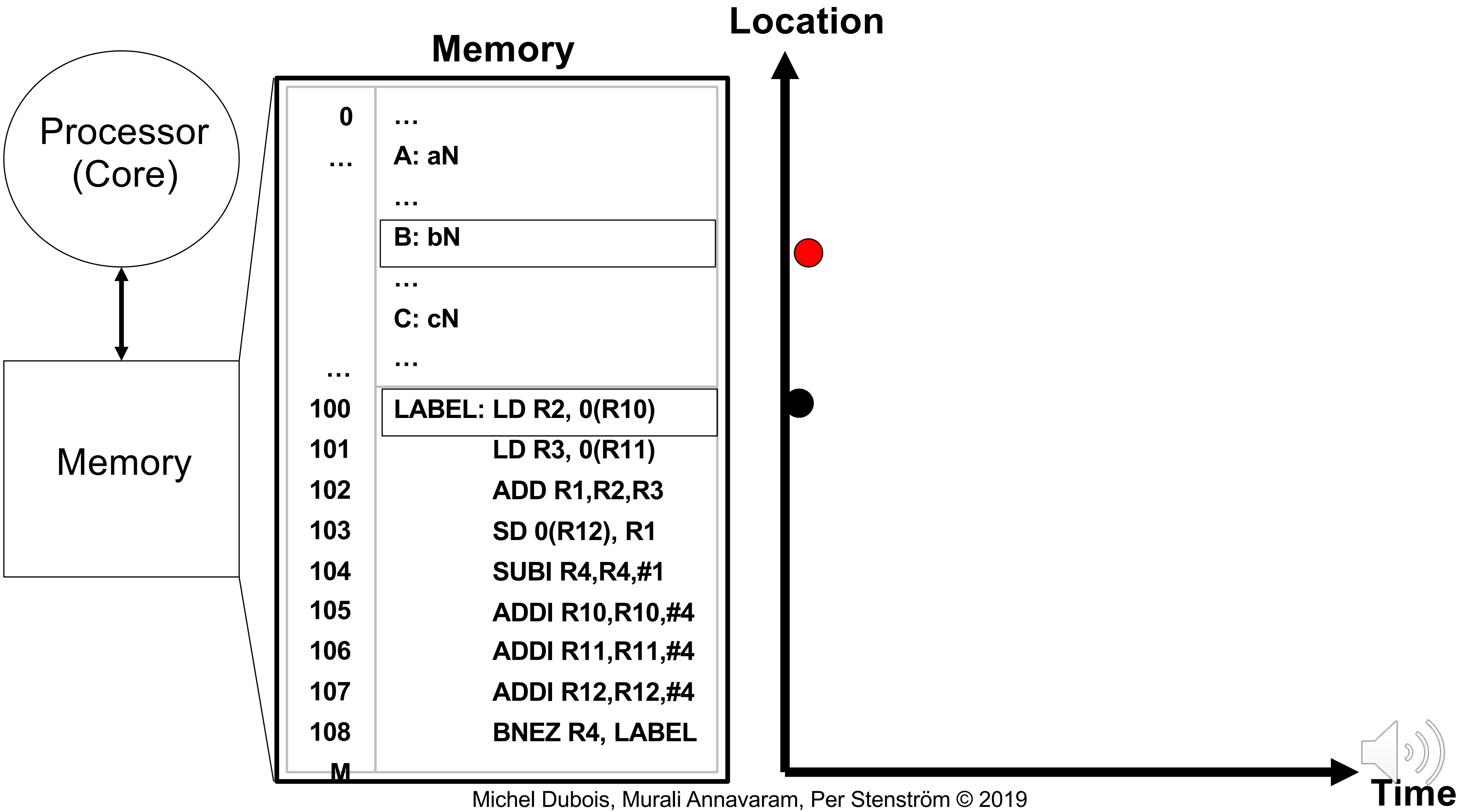


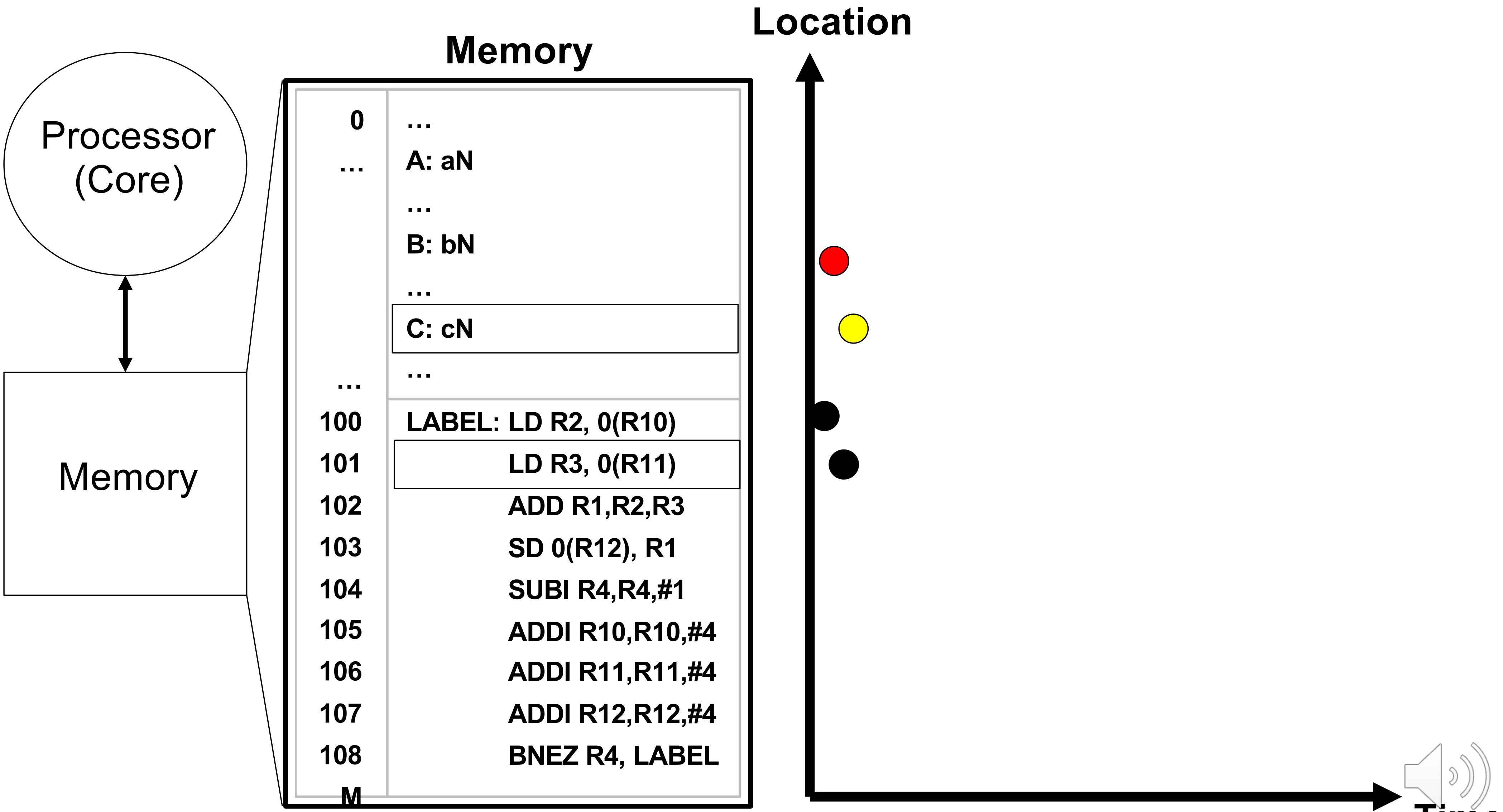
Memory

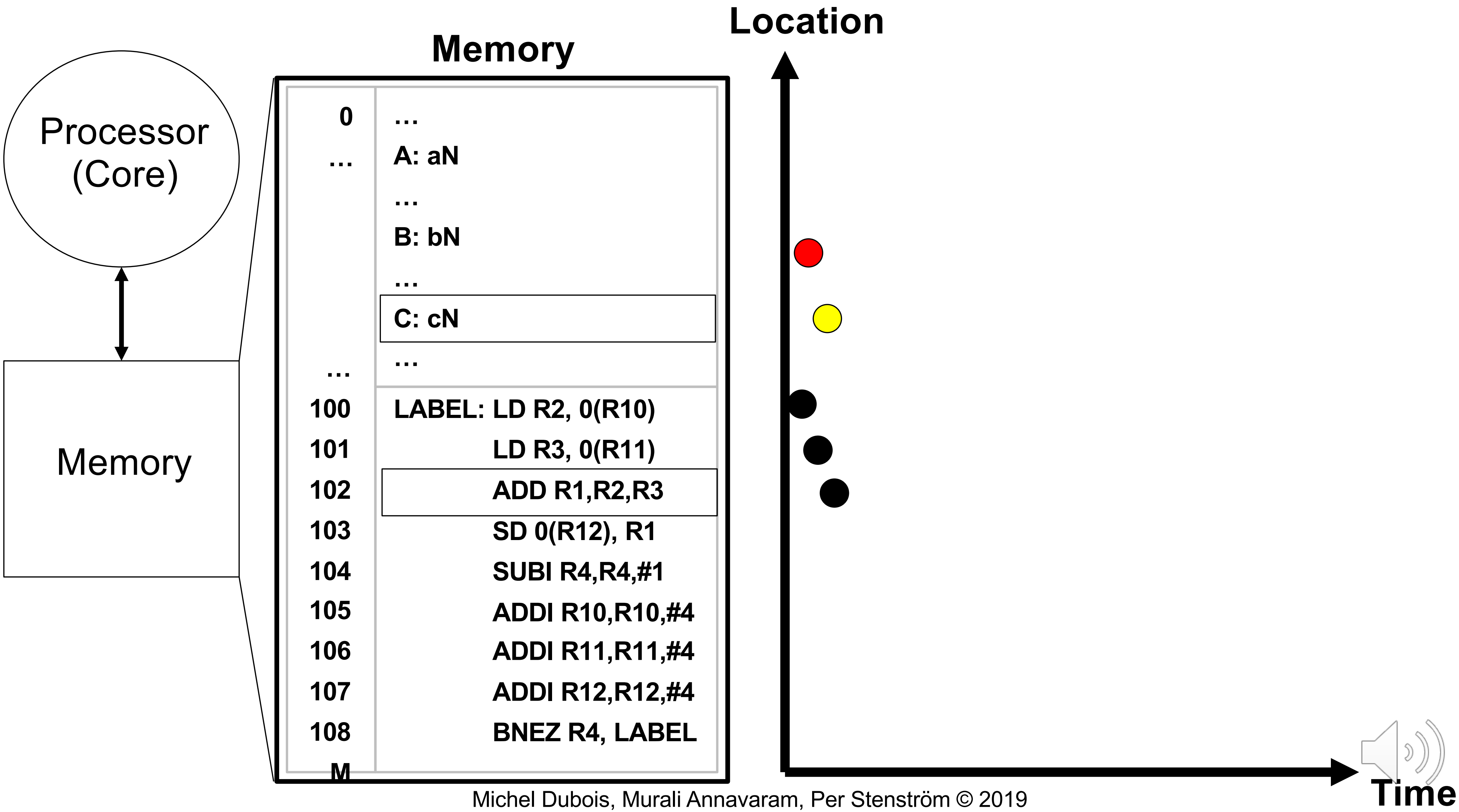
0	…
…	A: aN
	…
	B: bN
	…
	C: cN
…	…
100	LABEL: LD R2, 0(R10)
101	LD R3, 0(R11)
102	ADD R1,R2,R3
103	SD 0(R12), R1
104	SUBI R4,R4,#1
105	ADDI R10,R10,#4
106	ADDI R11,R11,#4
107	ADDI R12,R12,#4
108	BNEZ R4, LABEL
M	


Processor (Core)
Memory
Memory
0 …
… A: aN
…
B: bN
…
C: cN
… …
100 LABEL: LD R2, 0(R10)
101 LD R3, 0(R11)
102 ADD R1,R2,R3
103 SD 0(R12), R1
104 SUBI R4,R4,#1
105 ADDI R10,R10,#4
106 ADDI R11,R11,#4
107 ADDI R12,R12,#4
108 BNEZ R4, LABEL
M
Location
Time

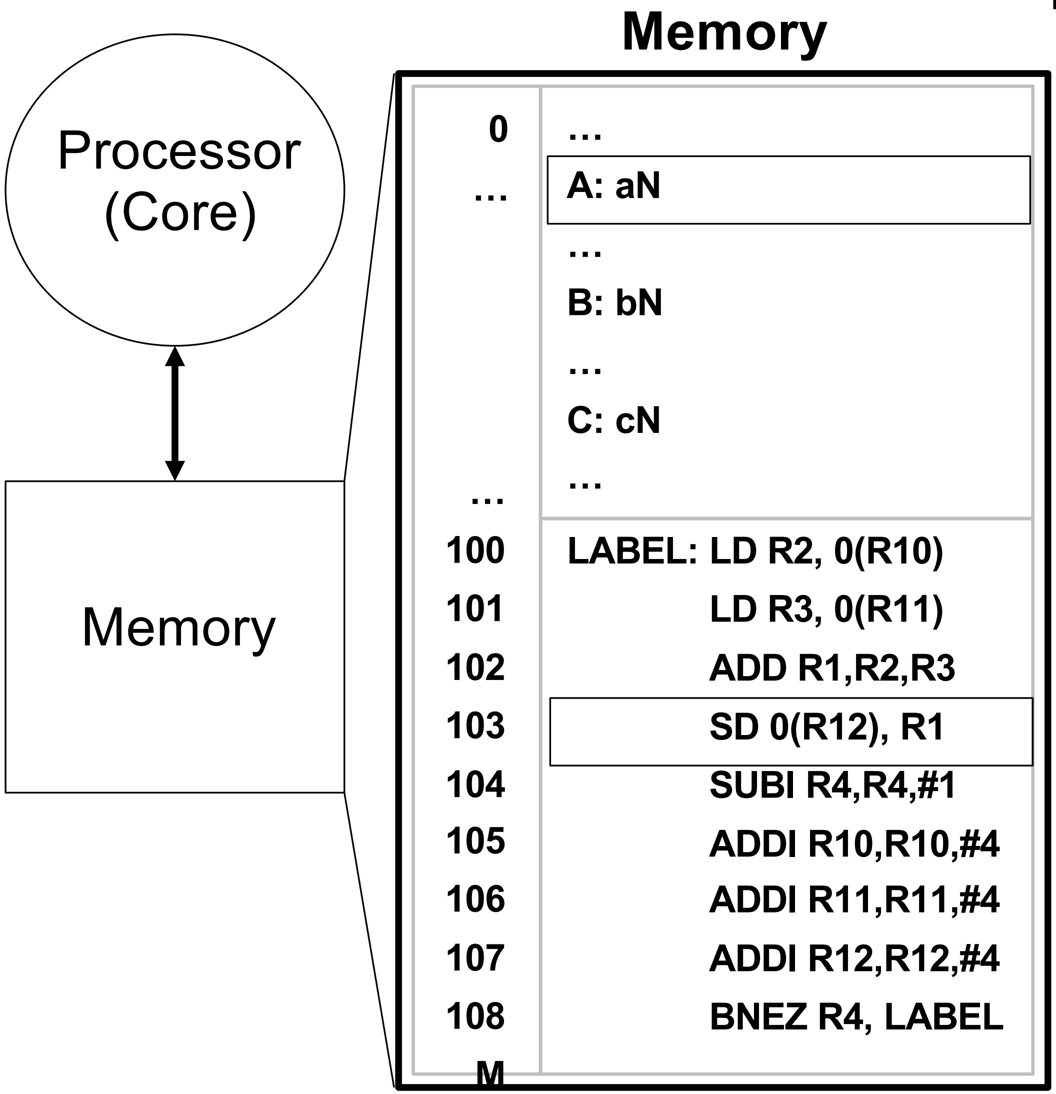

Processor (Core)
Memory
Memory
0
…
…
A: aN
…
B: bN
…
C: cN
…
100 LABEL: LD R2, 0(R10)
101 LD R3, 0(R11)
102 ADD R1,R2,R3
103 SD 0(R12), R1
104 SUBI R4,R4,#1
105 ADDI R10,R10,#4
106 ADDI R11,R11,#4
107 ADDI R12,R12,#4
108 BNEZ R4, LABEL
M
Location
Time

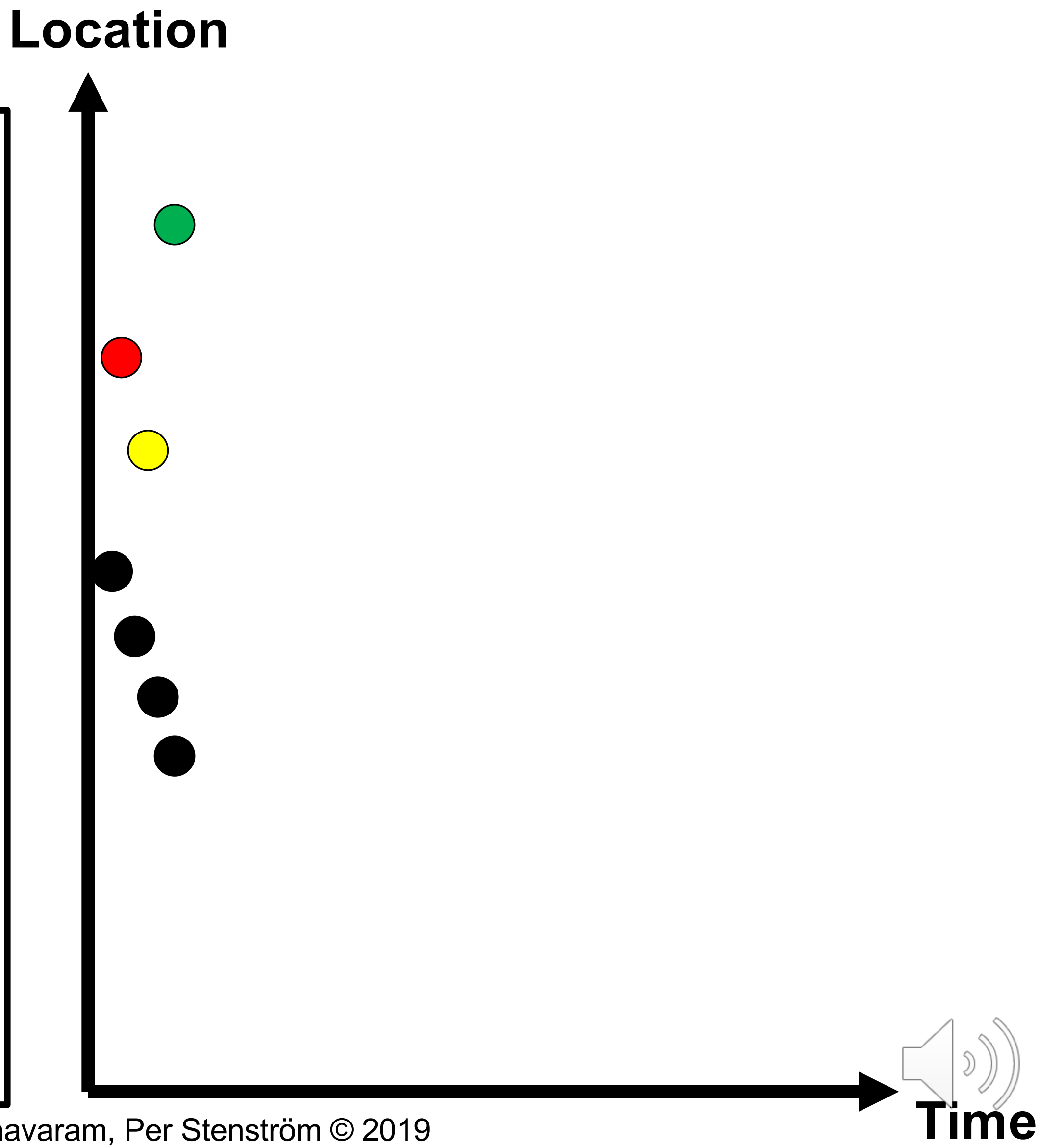

Processor
(Core)
Memory
Memory
0
…
A: aN
…
B: bN
…
C: cN
…
…
100 LABEL: LD R2, 0(R10)
101 LD R3, 0(R11)
102 ADD R1,R2,R3
103 SD 0(R12), R1
104 SUBI R4,R4,#1
105 ADDI R10,R10,#4
106 ADDI R11,R11,#4
107 ADDI R12,R12,#4
108 BNEZ R4, LABEL
M
Location
Time

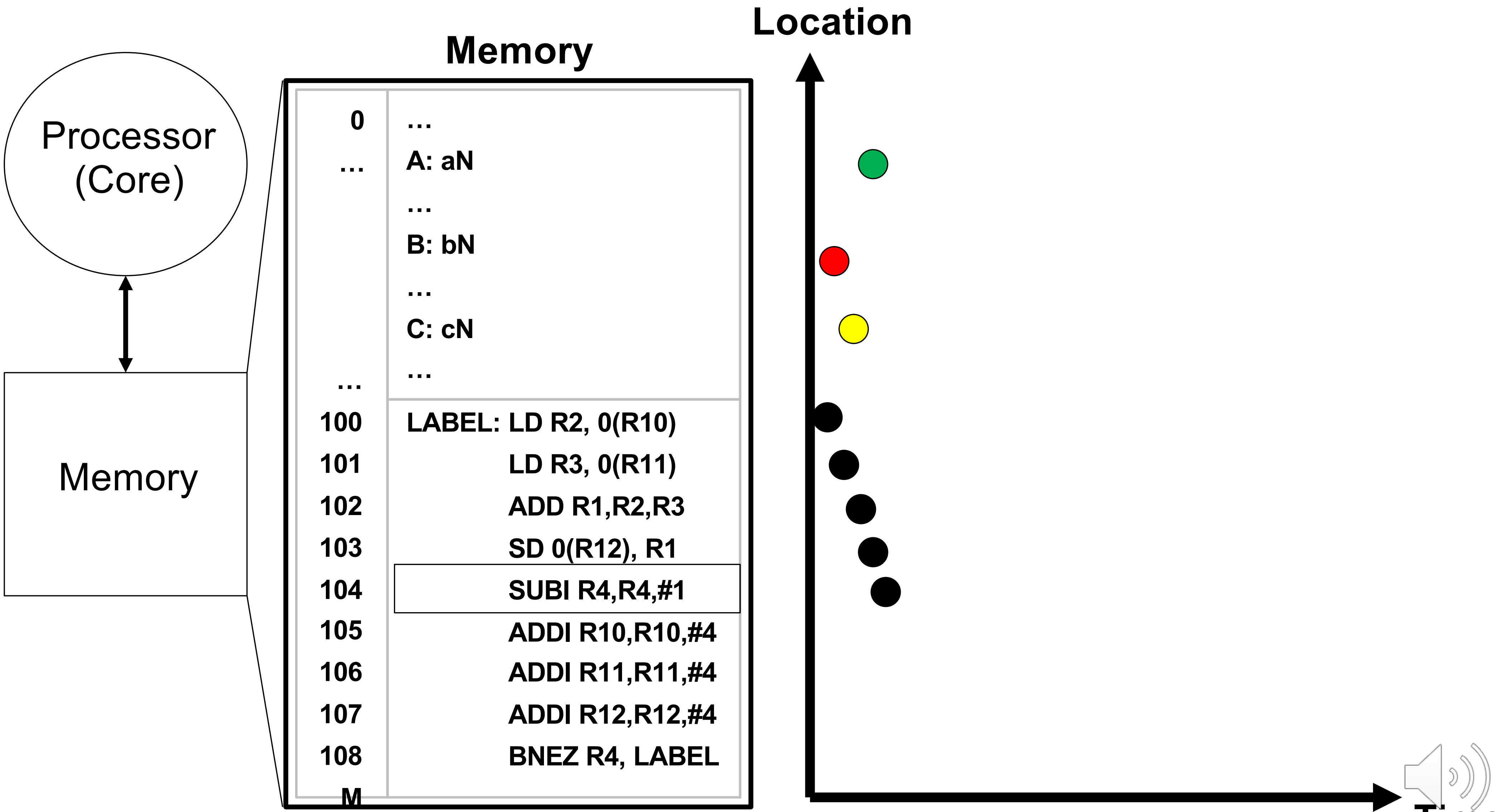

Processor (Core)
Memory
Memory
Location
Time
0
...
...
100
101
102
103
104
105
106
107
108
M
...
A: aN
...
B: bN
...
C: cN
...
LABEL: LD R2, 0(R10)
LD R3, 0(R11)
ADD R1,R2,R3
SD 0(R12), R1
SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4
BNEZ R4, LABEL

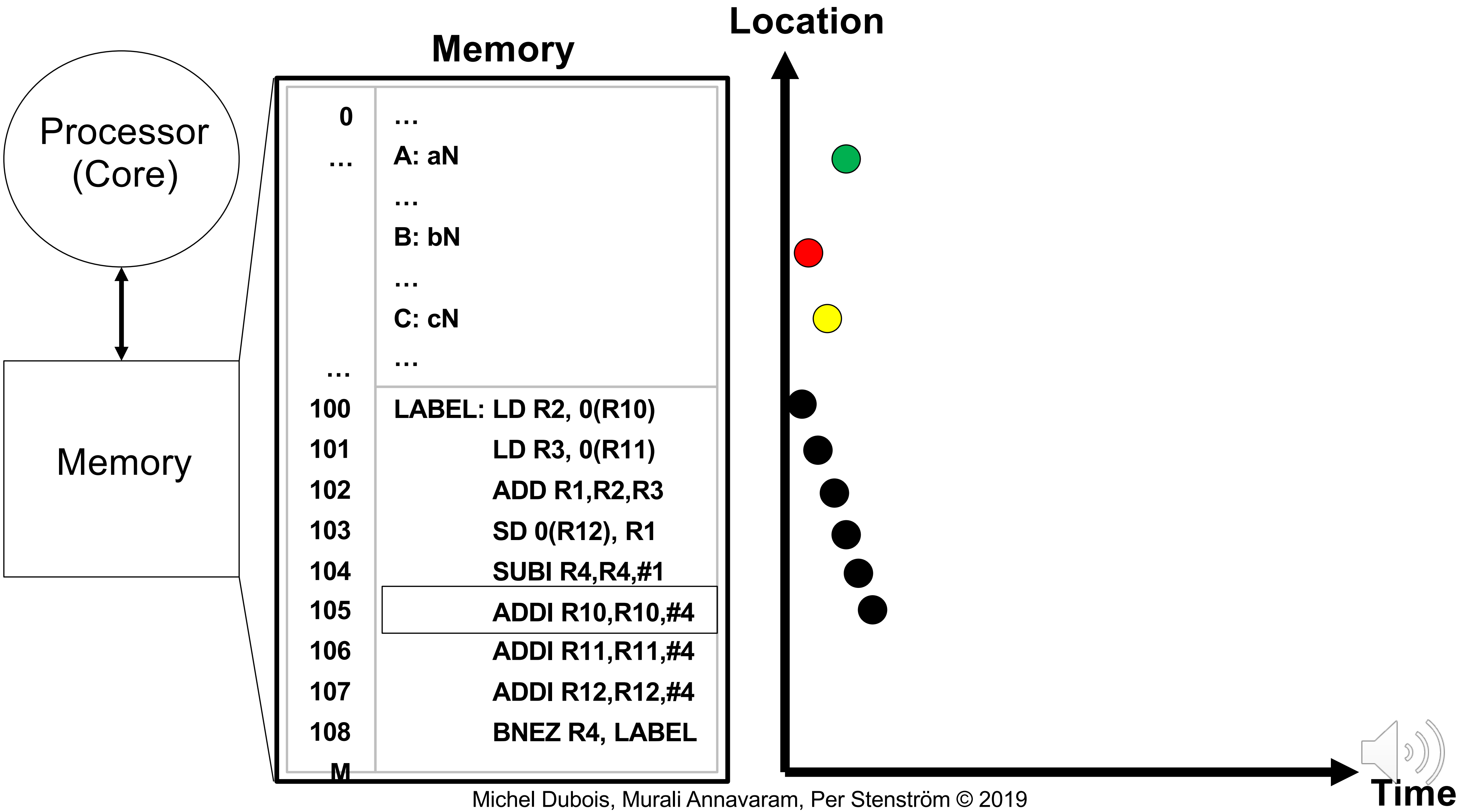


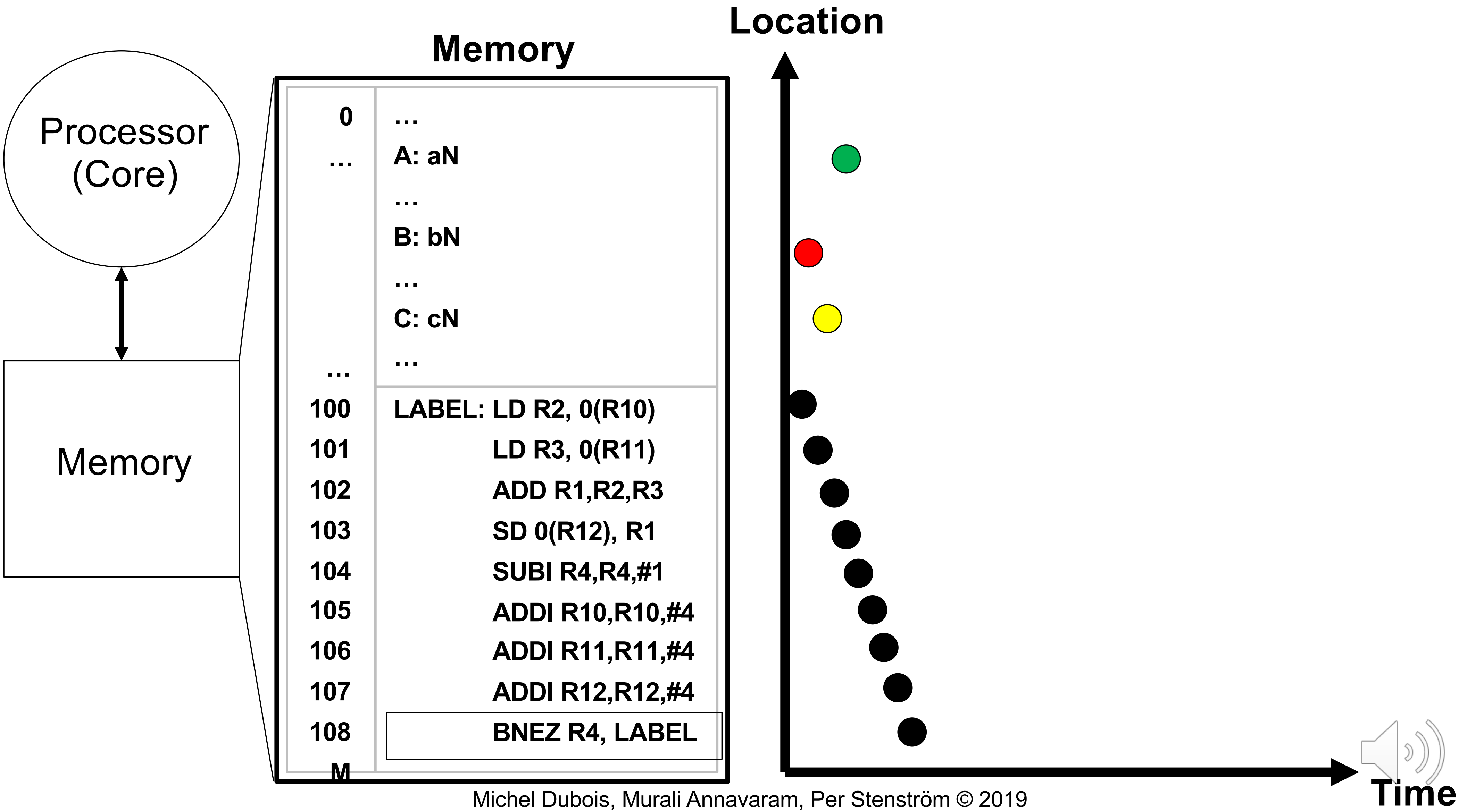
Processor (Core)
Memory
Memory
0 …
… A: aN
…
B: bN
…
C: cN
…
…
100 LABEL: LD R2, 0(R10)
101 LD R3, 0(R11)
102 ADD R1,R2,R3
103 SD 0(R12), R1
104 SUBI R4,R4,#1
105 ADDI R10,R10,#4
106 ADDI R11,R11,#4
107 ADDI R12,R12,#4
108 BNEZ R4, LABEL
M
Location
Time

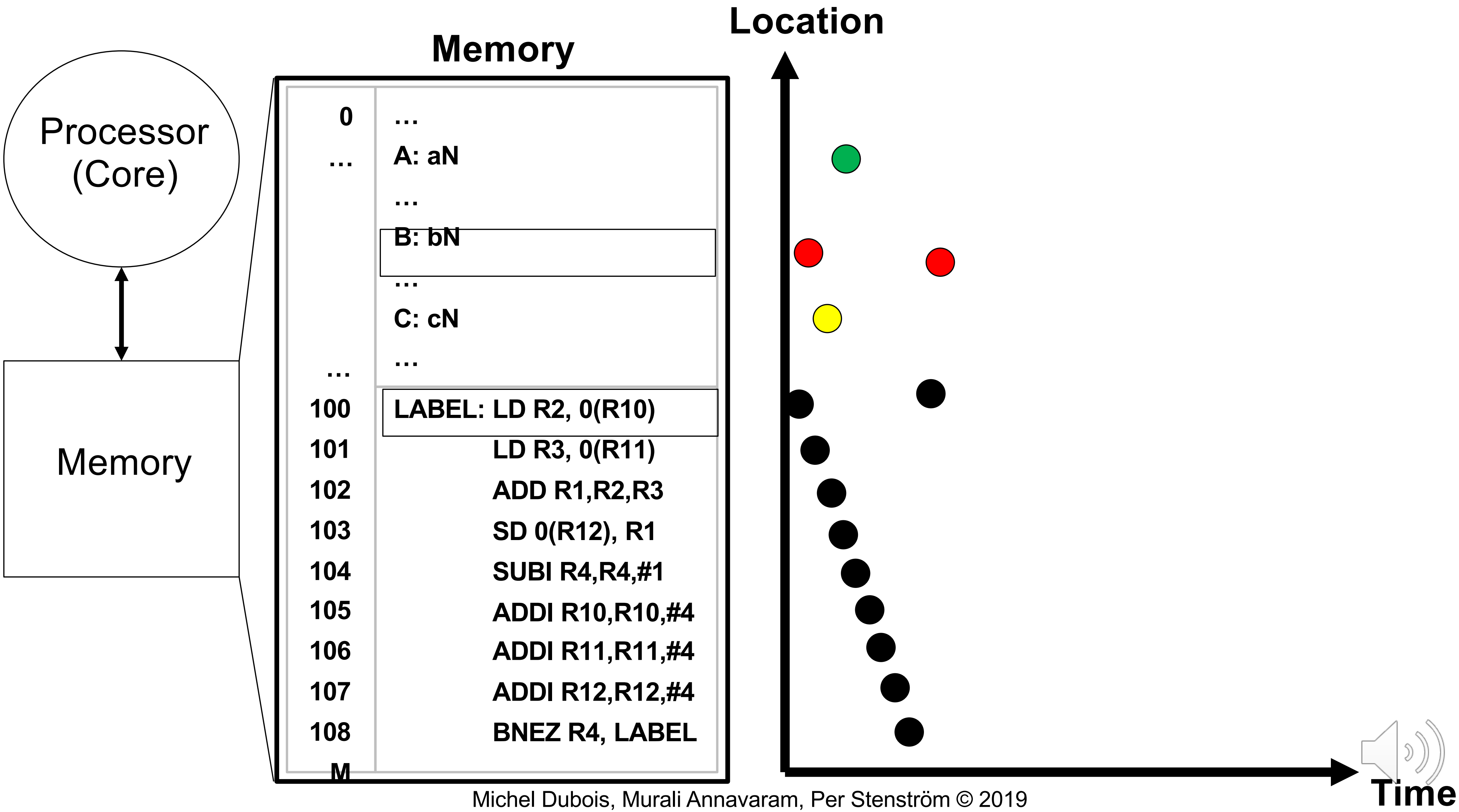

Processor (Core)
Memory
Memory
0 …
… A: aN
…
B: bN
…
C: cN
…
…
100 LABEL: LD R2, 0(R10)
101 LD R3, 0(R11)
102 ADD R1,R2,R3
103 SD 0(R12), R1
104 SUBI R4,R4,#1
105 ADDI R10,R10,#4
106 ADDI R11,R11,#4
107 ADDI R12,R12,#4
108 BNEZ R4, LABEL
M

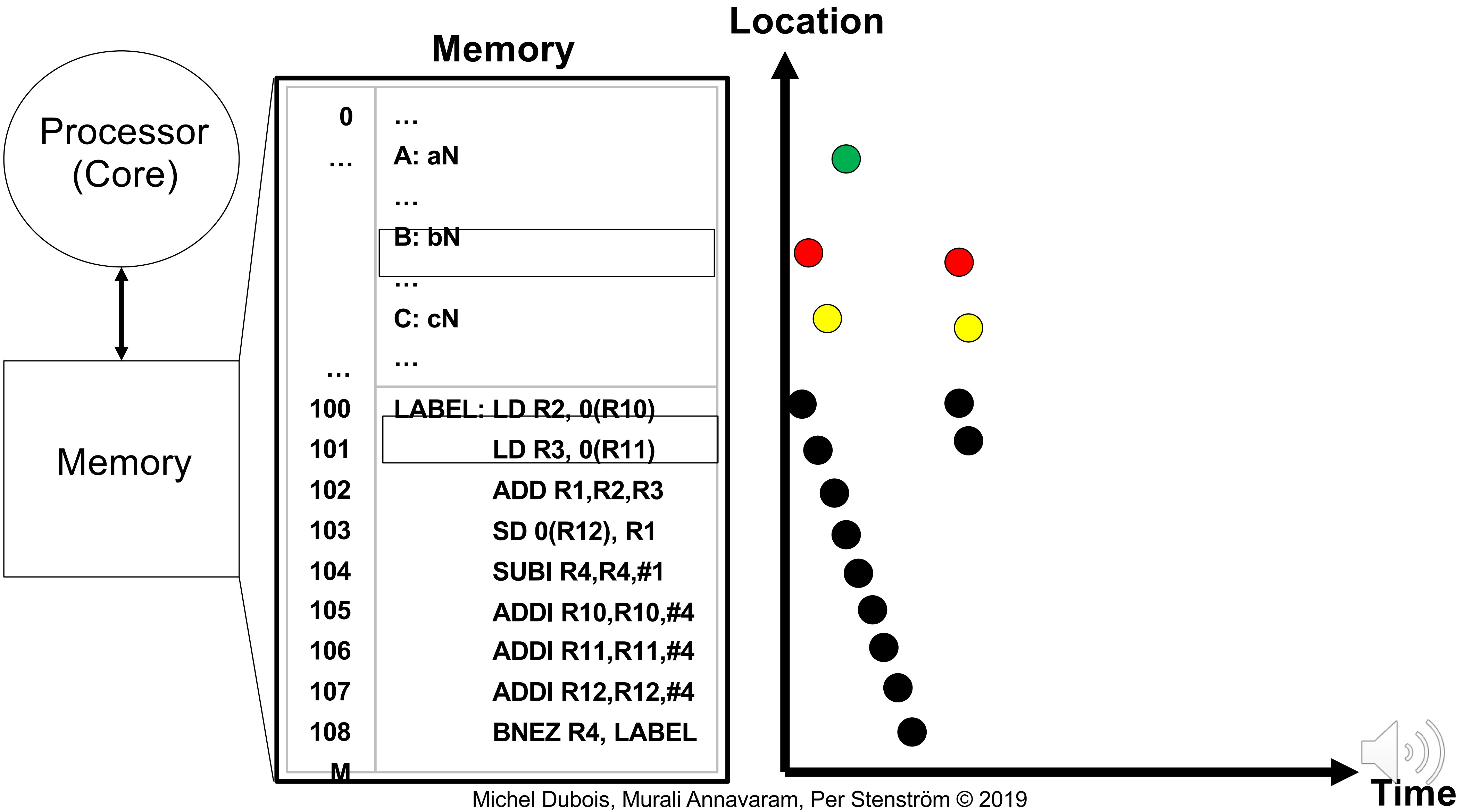

Location
Time

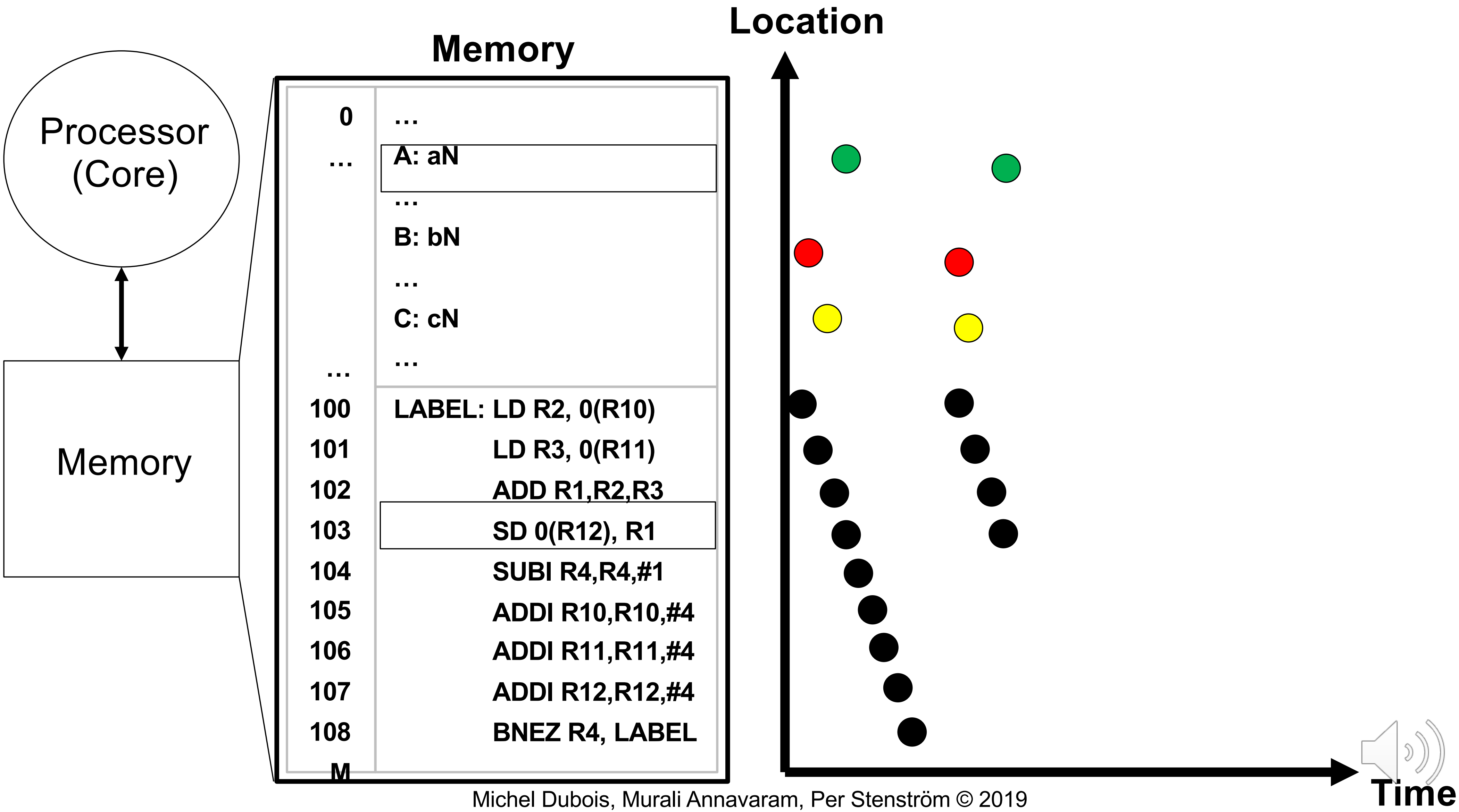

Processor (Core)
Memory
Memory
0
...
...
...
A: aN
...
B: bN
...
C: cN
...
100 LABEL: LD R2, 0(R10)
101 LD R3, 0(R11)
102 ADD R1,R2,R3
103 SD 0(R12), R1
104 SUBI R4,R4,#1
105 ADDI R10,R10,#4
106 ADDI R11,R11,#4
107 ADDI R12,R12,#4
108 BNEZ R4, LABEL
M
Location
Time


Processor (Core)
Memory
Memory
0
...
...
100
101
102
103
104
105
106
107
108
M
...
A: aN
...
B: bN
...
C: cN
...
LABEL: LD R2, 0(R10)
LD R3, 0(R11)
ADD R1,R2,R3
SD 0(R12), R1
SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4
BNEZ R4, LABEL
Location
Time

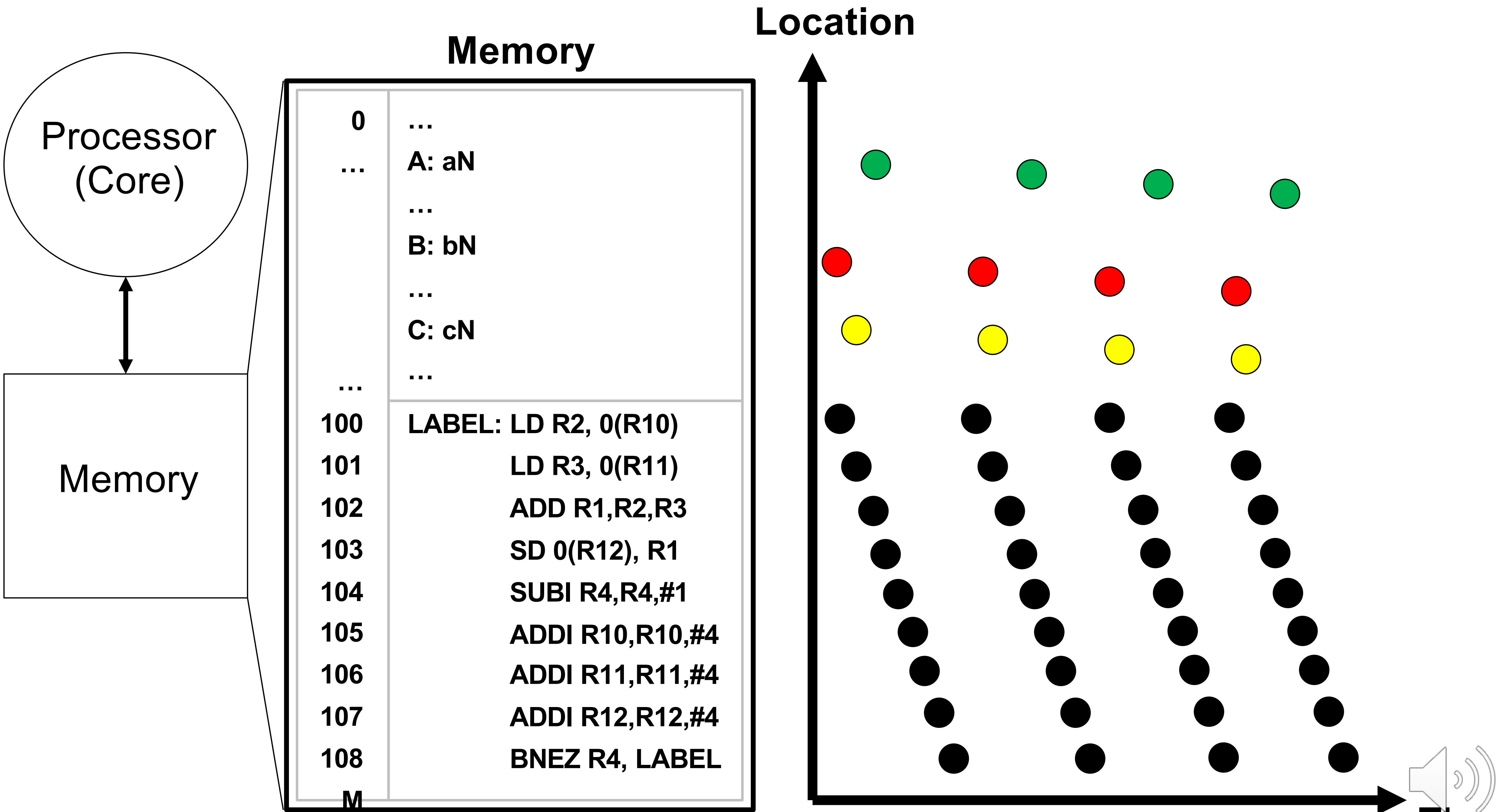

Processor (Core)
Memory
Memory
0
...
...
...
A: aN
...
B: bN
...
C: cN
...
100 LABEL: LD R2, 0(R10)
101 LD R3, 0(R11)
102 ADD R1,R2,R3
103 SD 0(R12), R1
104 SUBI R4,R4,#1
105 ADDI R10,R10,#4
106 ADDI R11,R11,#4
107 ADDI R12,R12,#4
108 BNEZ R4, LABEL
M
Location
Time


Processor (Core)
Memory
Memory
0
...
...
...
A: aN
...
B: bN
...
C: cN
...
100
101
102
103
104
105
106
107
108
M
LABEL: LD R2, 0(R10)
LD R3, 0(R11)
ADD R1,R2,R3
SD 0(R12), R1
SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4
BNEZ R4, LABEL
Location
Time


Processor (Core)
Memory
Memory
0
...
...
...
A: aN
...
B: bN
...
C: cN
...
100 LABEL: LD R2, 0(R10)
101 LD R3, 0(R11)
102 ADD R1,R2,R3
103 SD 0(R12), R1
104 SUBI R4,R4,#1
105 ADDI R10,R10,#4
106 ADDI R11,R11,#4
107 ADDI R12,R12,#4
108 BNEZ R4, LABEL
M
Location
Time

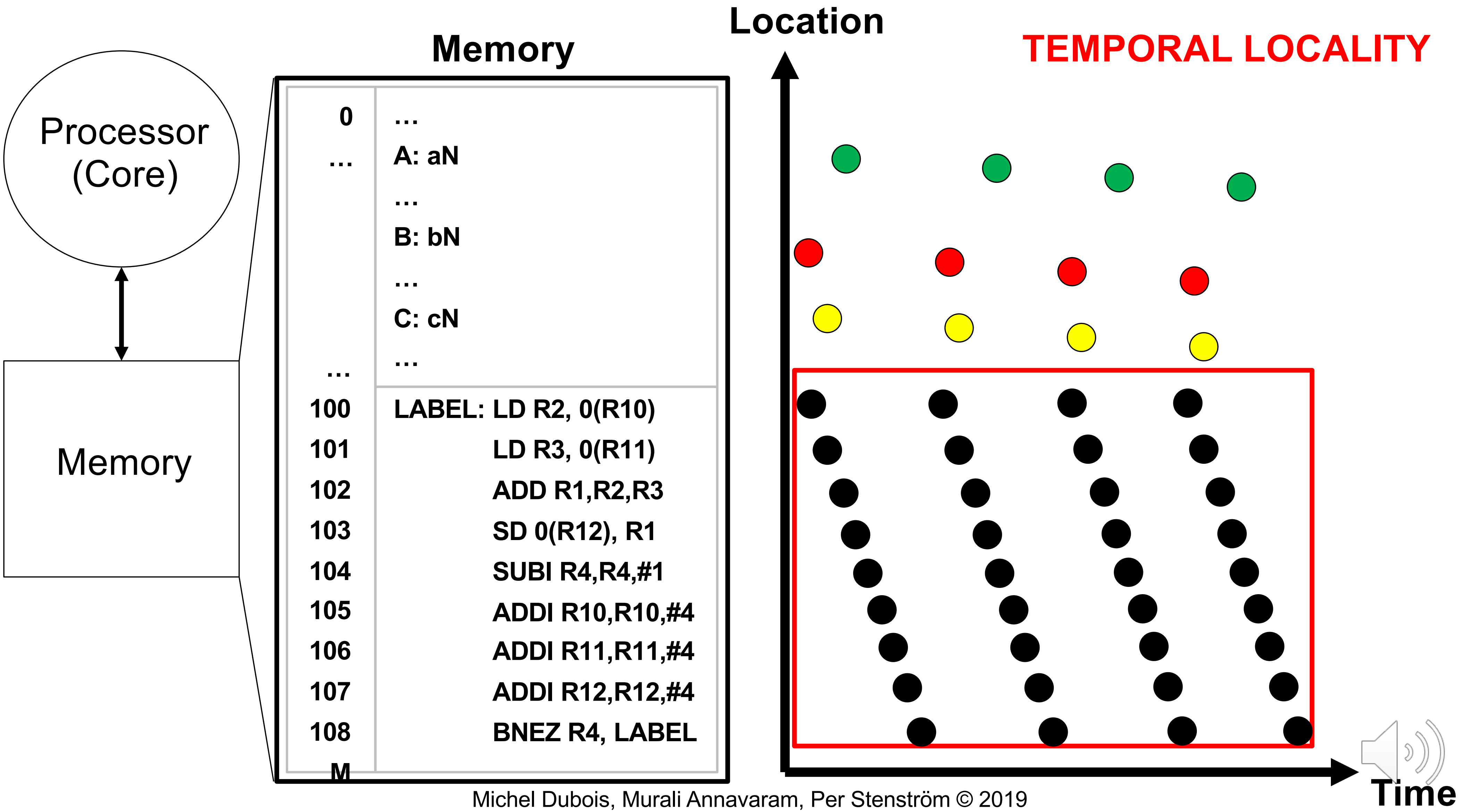


Processor
(Core)
Memory
Memory
0
…
…
A: aN
…
B: bN
…
C: cN
…
100
101
102
103
104
105
106
107
108
M
LABEL: LD R2, 0(R10)
LD R3, 0(R11)
ADD R1,R2,R3
SD 0(R12), R1
SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4
BNEZ R4, LABEL
Location
Time

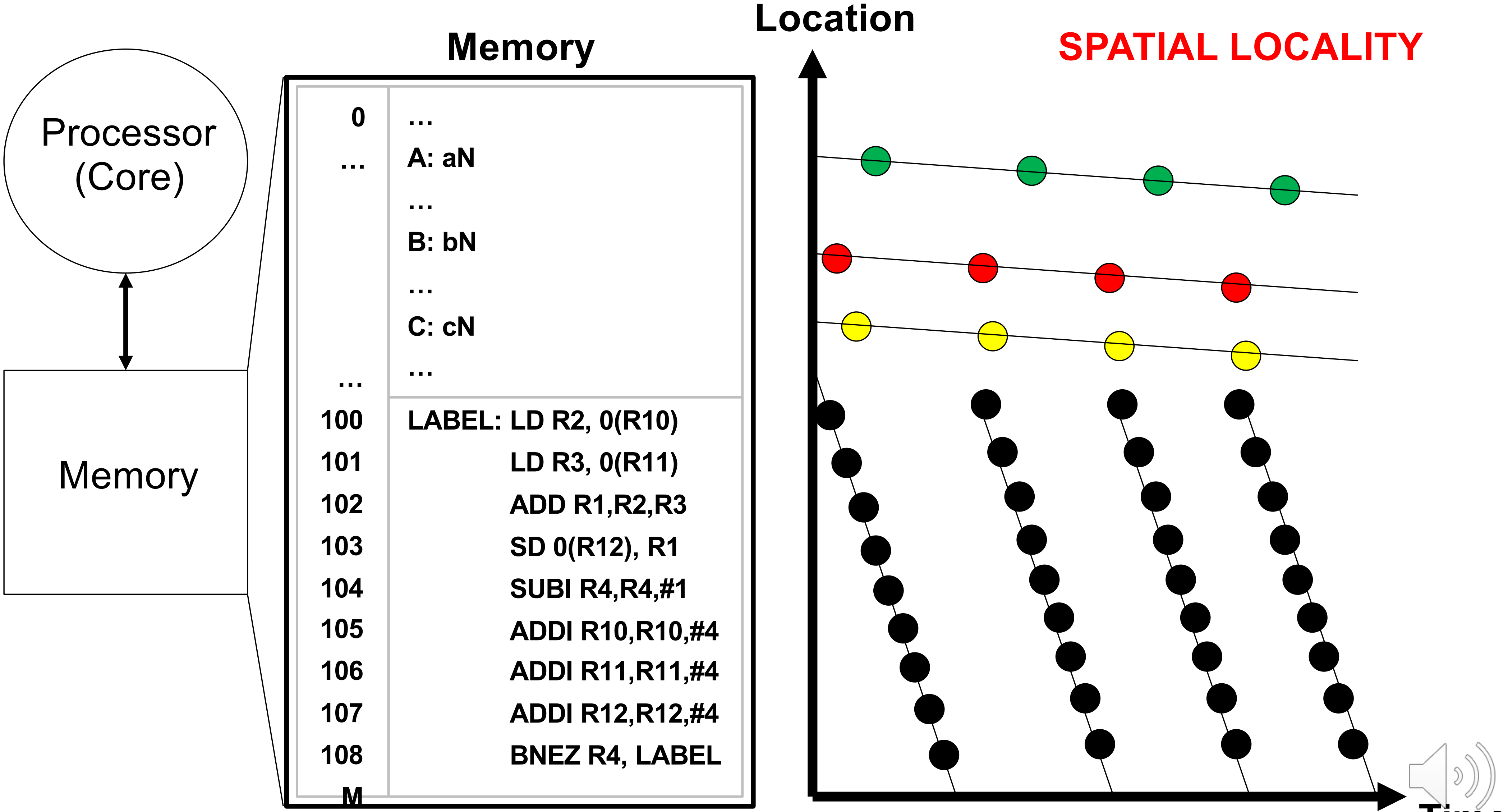


Processor (Core)
Memory
Memory
0 …
… A: aN
…
B: bN
…
C: cN
…
…
100 LABEL: LD R2, 0(R10)
101 LD R3, 0(R11)
102 ADD R1,R2,R3
103 SD 0(R12), R1
104 SUBI R4,R4,#1
105 ADDI R10,R10,#4
106 ADDI R11,R11,#4
107 ADDI R12,R12,#4
108 BNEZ R4, LABEL
M
Location
Time

TEMPORAL LOCALITY



SPATIAL LOCALITY



Cache Mapping Policies and Organizations (Ch 4.3.1)

A Simple Cache Design

Instruction memory

0	LABEL: LD R2,0(R10)
4	LD R3,0(R11)
8	ADD R1,R2,R3
12	SD 0(R12),R1
16	SUBI R4,R4,#1
20	ADDI R10,R10,#4
24	ADDI R11,R11,#4
28	ADDI R12,R12,#4
32	BNEZ R4,LABEL
36	
40	
44	
48	
…	
…	
124	
127	

Instruction memory

0	LABEL: LD R2,0(R10)
4	LD R3,0(R11)
8	ADD R1,R2,R3
12	SD 0(R12),R1
16	SUBI R4,R4,#1
20	ADDI R10,R10,#4
24	ADDI R11,R11,#4
28	ADDI R12,R12,#4
32	BNEZ R4,LABEL
36	
40	
44	
48	
…	
…	
124	
127	

Memory blocks – 16 bytes each

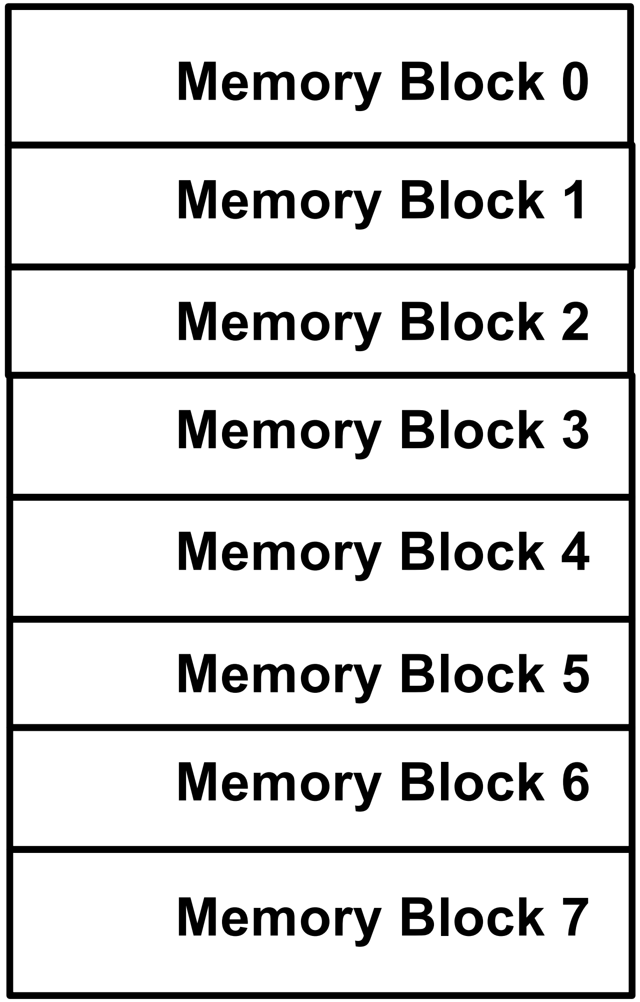
Question:
How many blocks does the memory contain?

Answer:
128/16 = 8

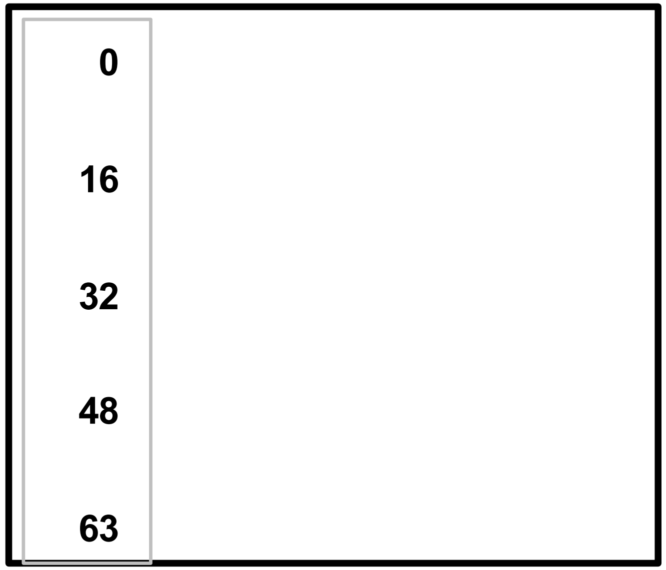
Instruction memory

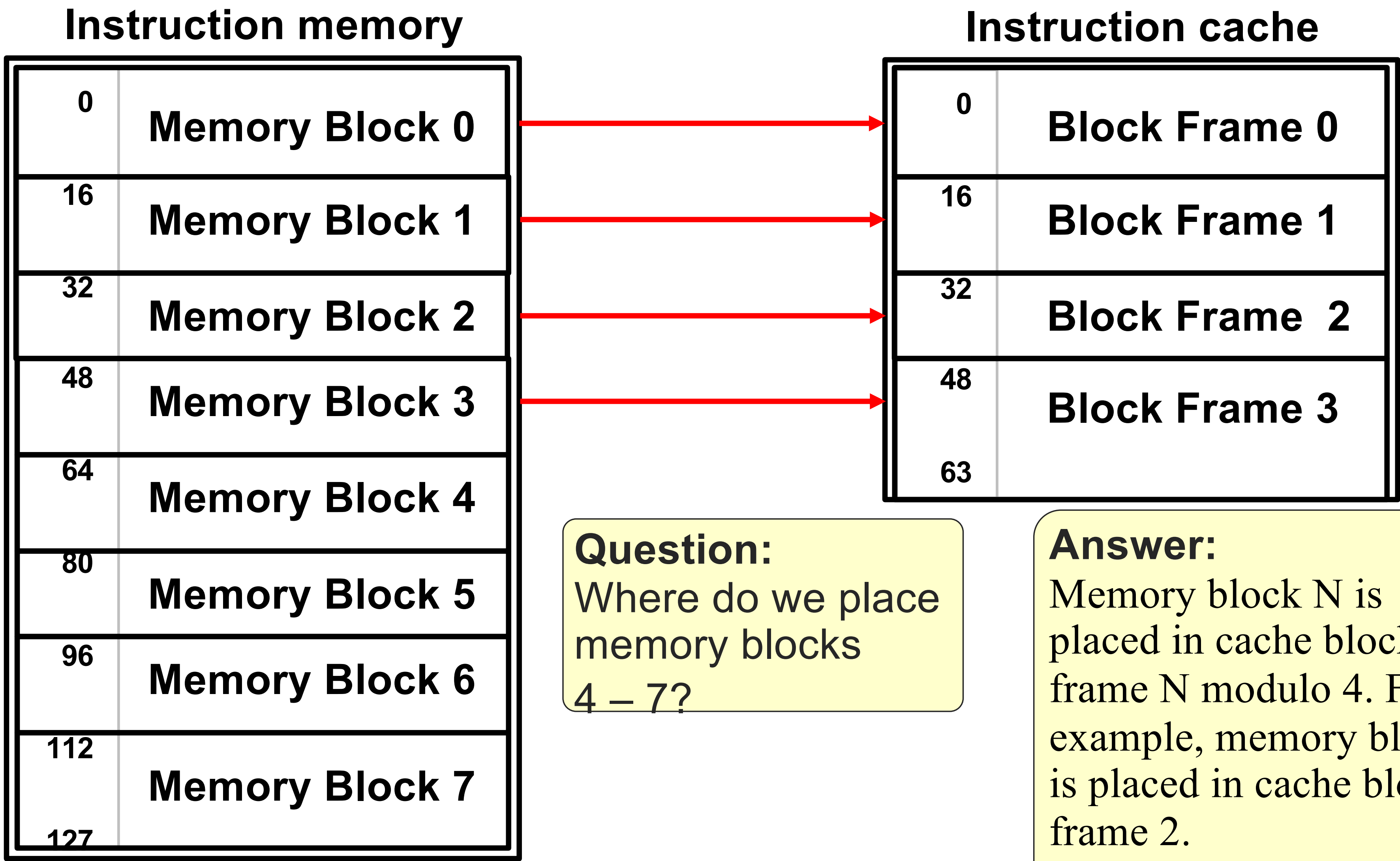
0	Memory Block 0
16	Memory Block 1
32	Memory Block 2
48	Memory Block 3
64	Memory Block 4
80	Memory Block 5
96	Memory Block 6
112 127	Memory Block 7

Instruction memory



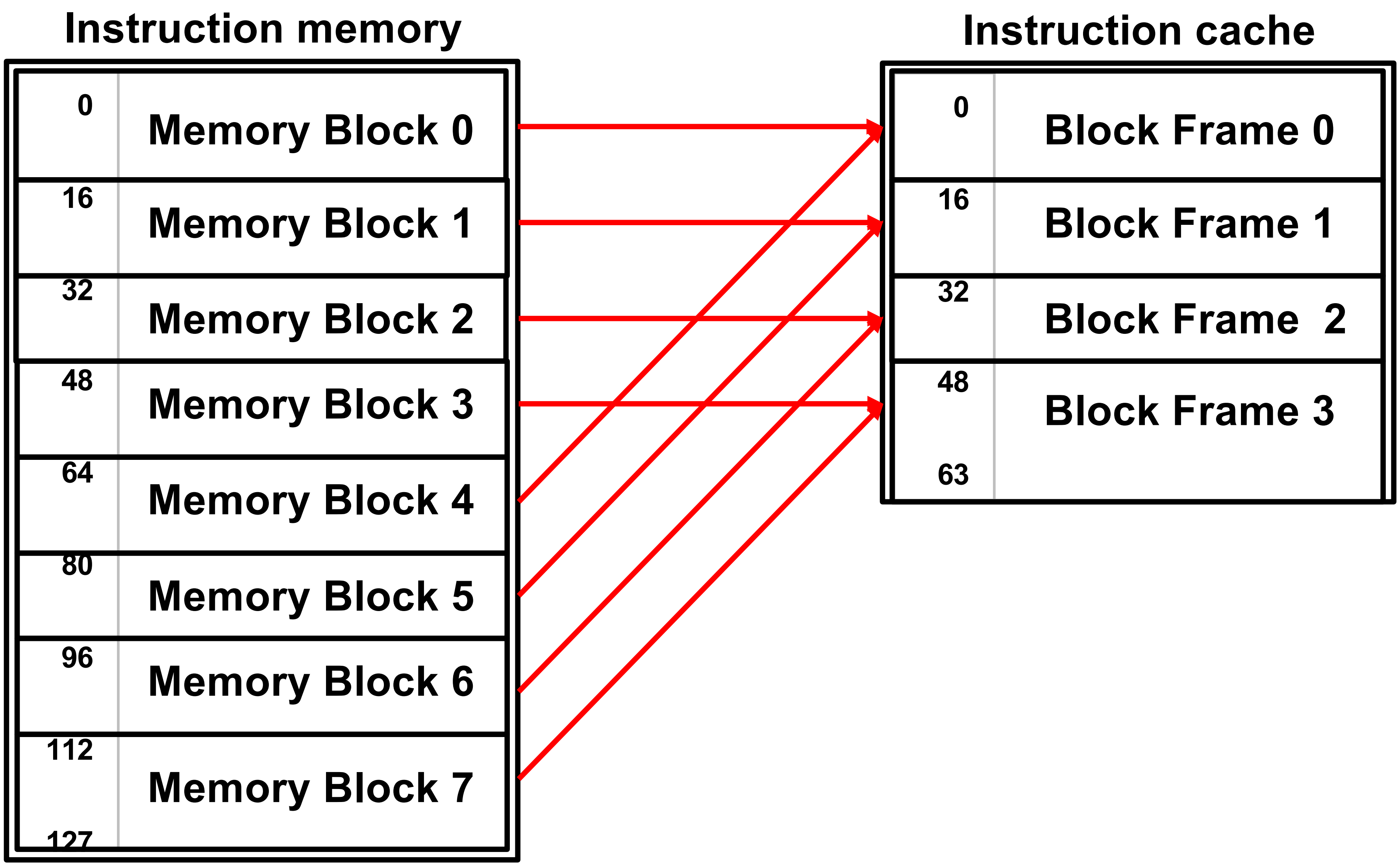
Instruction cache

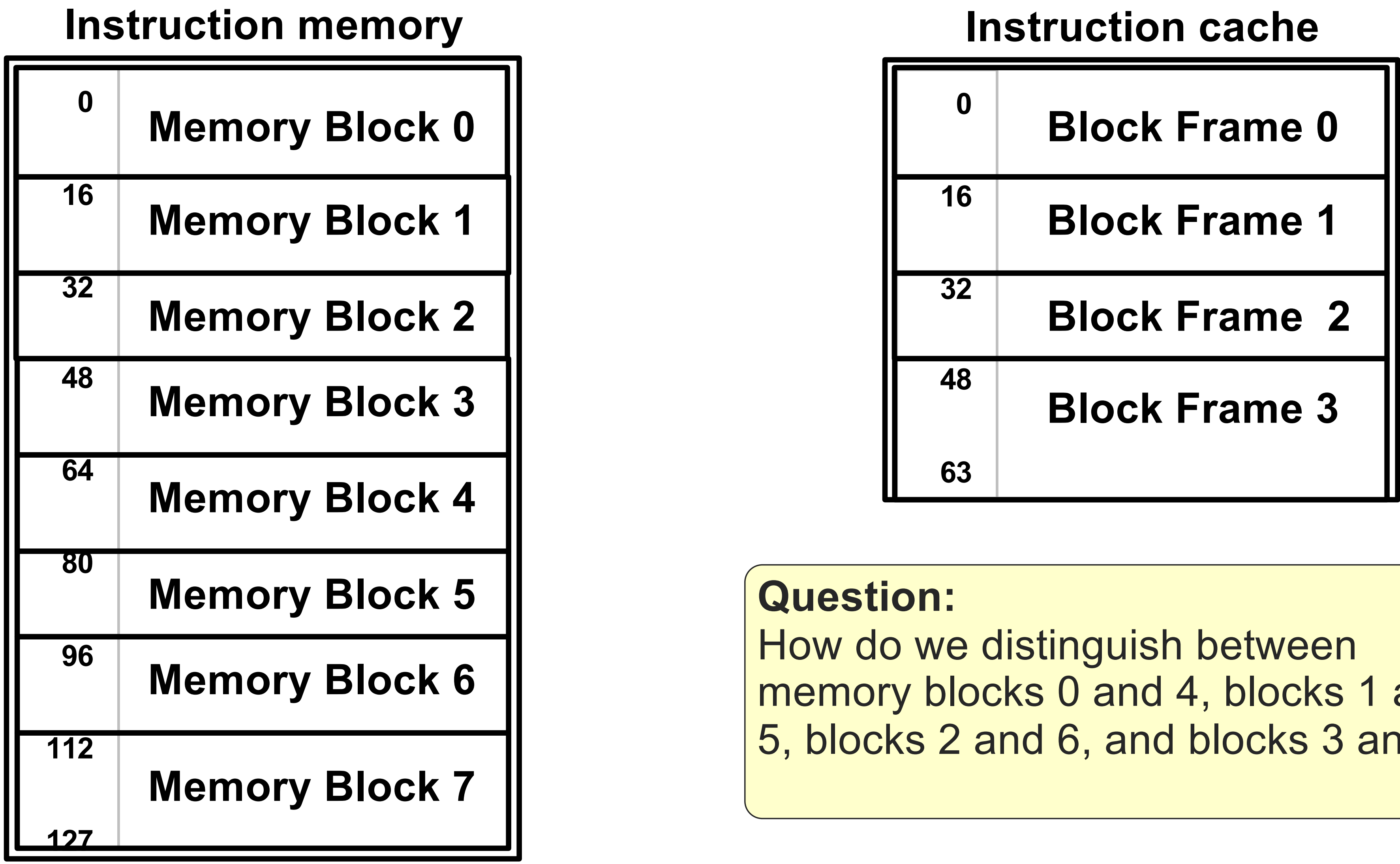




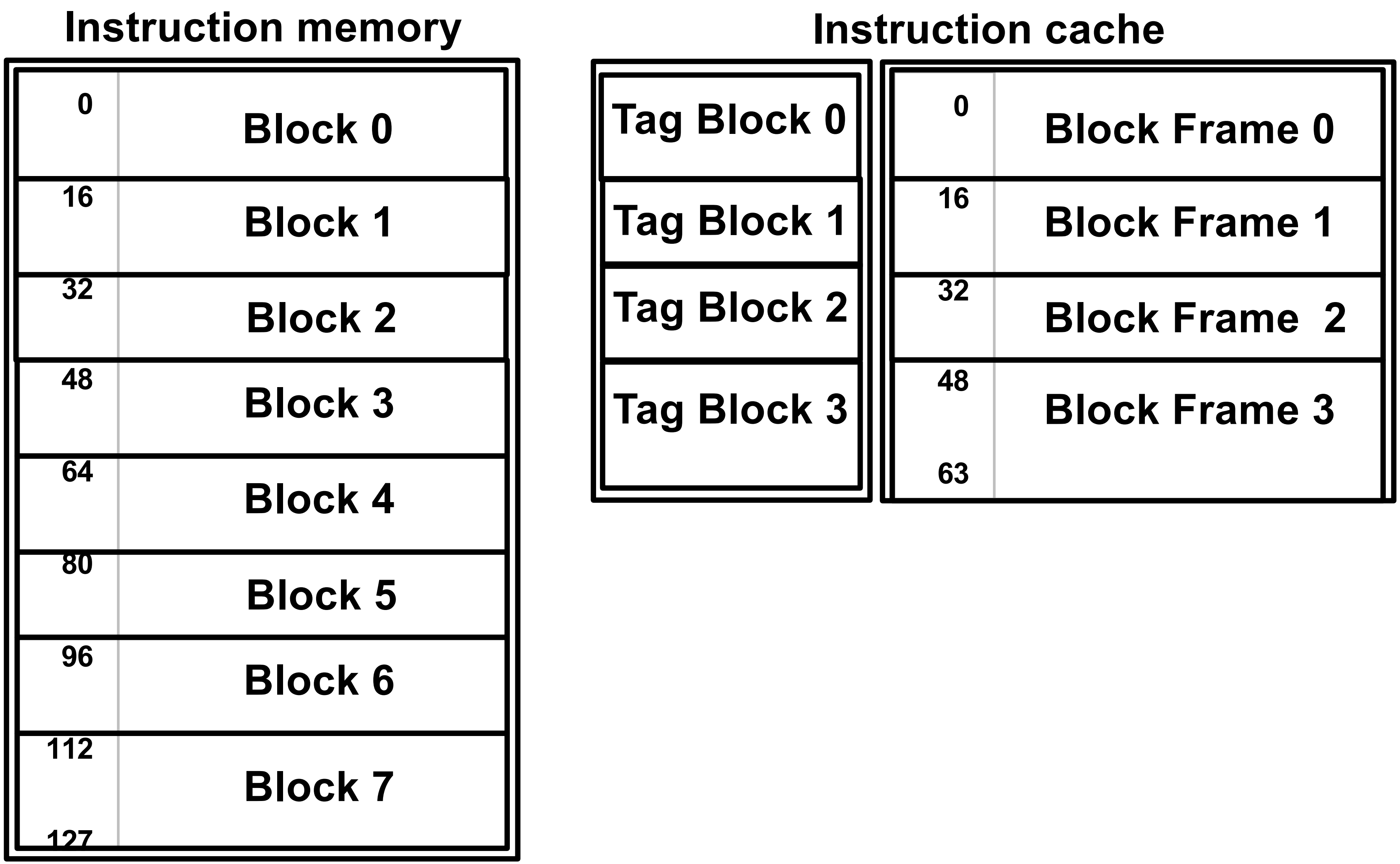
Question:
Where do we place memory blocks 4 – 7?

Answer:
Memory block N is placed in cache block frame N modulo 4. For example, memory block 6 is placed in cache block frame 2.


Instruction memory
0
Memory Block 0
16
Memory Block 1
32
Memory Block 2
48
Memory Block 3
64
Memory Block 4
80
Memory Block 5
96
Memory Block 6
112
Memory Block 7
127
Instruction cache
0
Block Frame 0
16
Block Frame 1
32
Block Frame 2
48
Block Frame 3
63



Question:
How do we distinguish between memory blocks 0 and 4, blocks 1 and 5, blocks 2 and 6, and blocks 3 and 7?


Instruction memory
0
Block 0
16
Block 1
32
Block 2
48
Block 3
64
Block 4
80
Block 5
96
Block 6
112
Block 7
127
Instruction cache
Tag Block 0
Tag Block 1
Tag Block 2
Tag Block 3
0
Block Frame 0
16
Block Frame 1
32
Block Frame 2
48
Block Frame 3
63

Block address: 3 bits

Block 0: 000
Block 1: 001
Block 2: 010
Block 3: 011
Block 4: 100
Block 5: 101
Block 6: 110
Block 7: 111

The Tag is a single bit

$$\frac{\text{Memory Size}}{\text{Cache Size}} = N \Rightarrow \log_2 N \text{ tag bits}$$

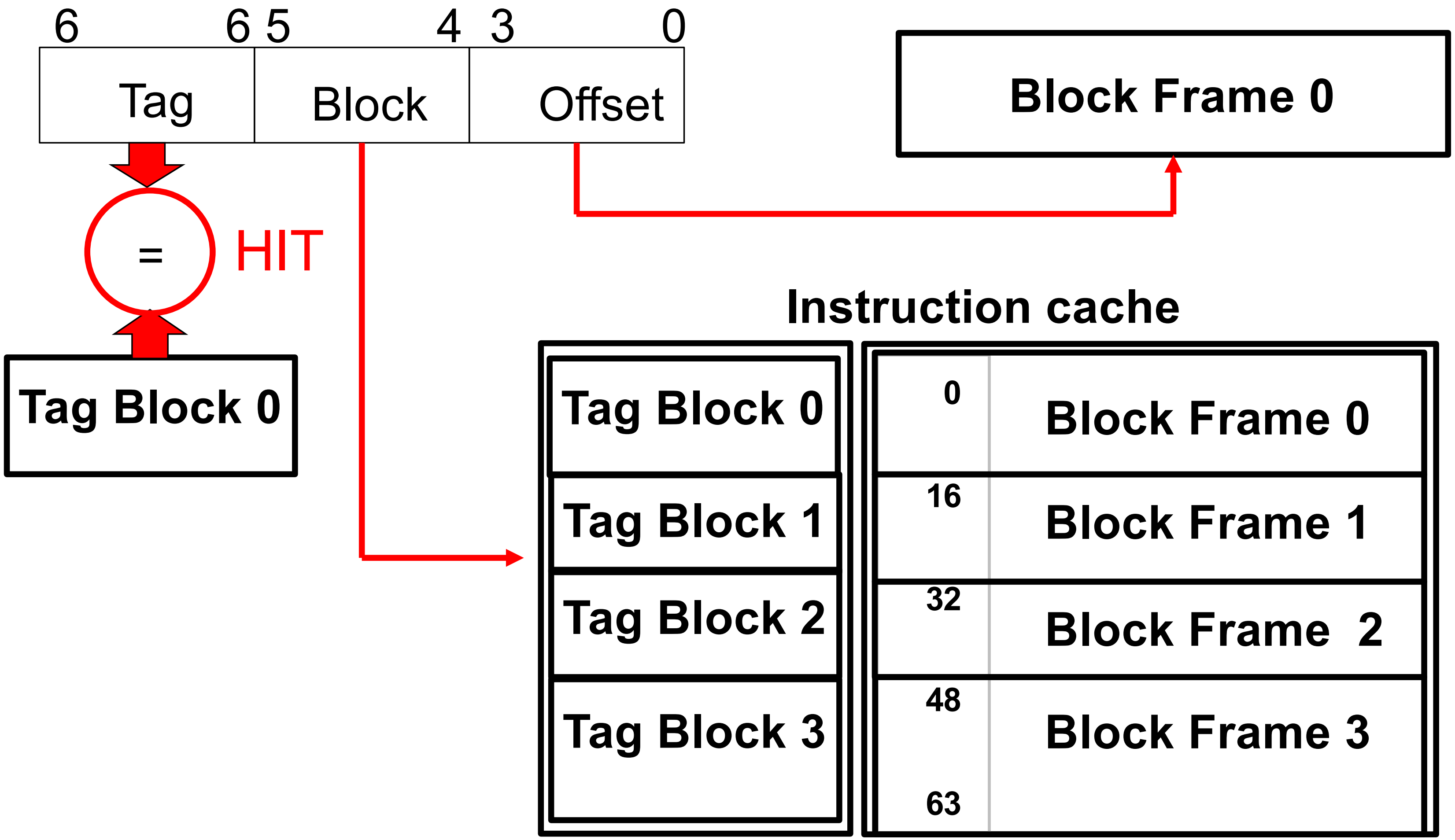
Question:
How many tag bits are needed if the memory is 64 Gbytes and the cache is 1 Mbytes?

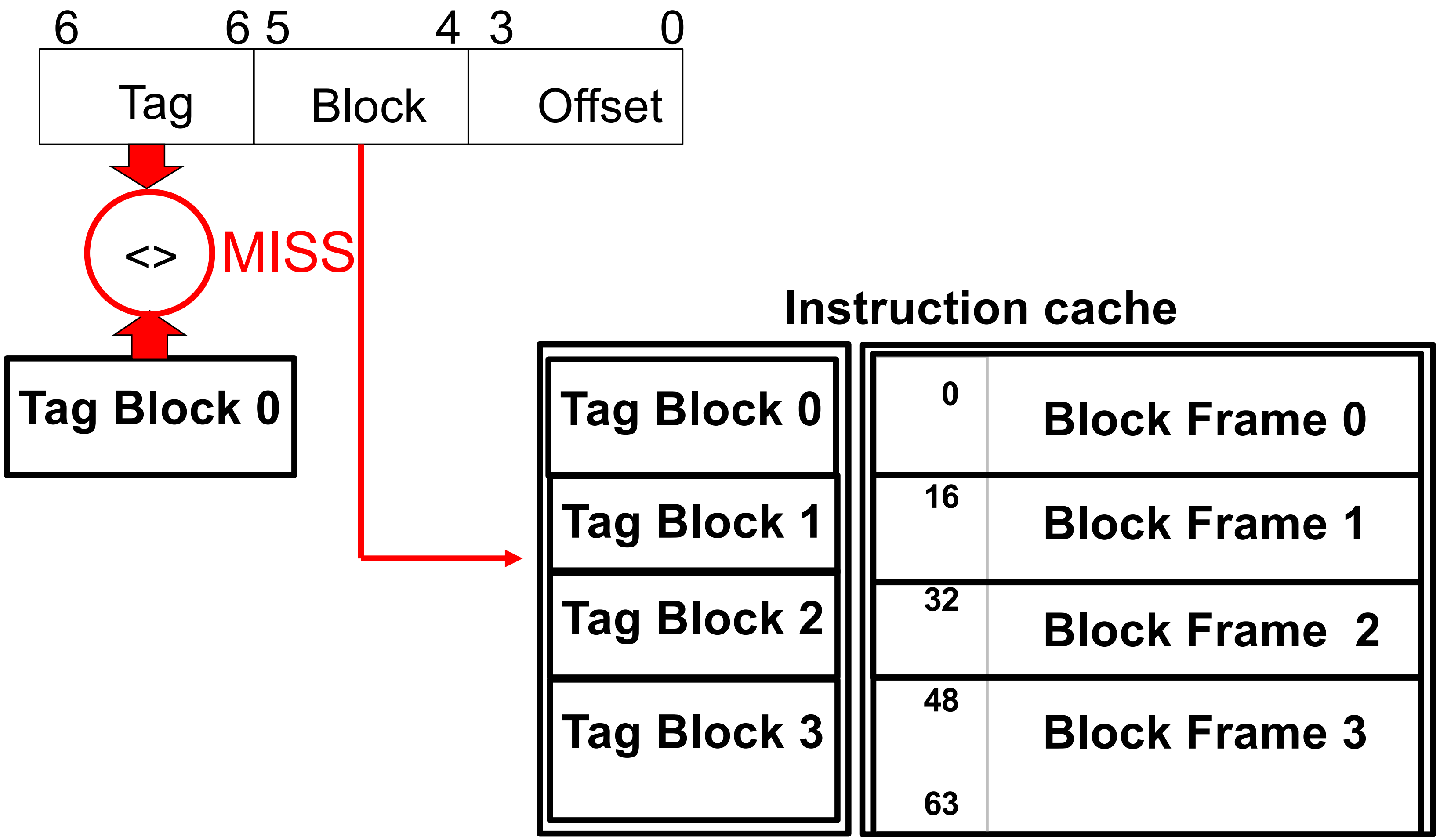
Answer:
Number of memory blocks that map to the same block frame is

64 Gbytes/1 Mbytes = 64 x 1024.

$\log_2(64 \times 1024) = 16$ bits.

$\frac{\text{Memory Size}}{\text{Cache Size}} = N \Rightarrow \log_2 N$ tag bits


6
6 5
4 3
0
Tag
Block
Offset
Block Frame 0
=
HIT
Tag Block 0
Instruction cache
Tag Block 0
Tag Block 1
Tag Block 2
Tag Block 3
0
Block Frame 0
16
Block Frame 1
32
Block Frame 2
48
Block Frame 3
63


6
6 5
4 3
0
Tag
Block
Offset
<>
MISS
Tag Block 0
Instruction cache
Tag Block 0
Tag Block 1
Tag Block 2
Tag Block 3
0
Block Frame 0
16
Block Frame 1
32
Block Frame 2
48
Block Frame 3
63

Memory address ($\log_2 128 = 7$ bits):

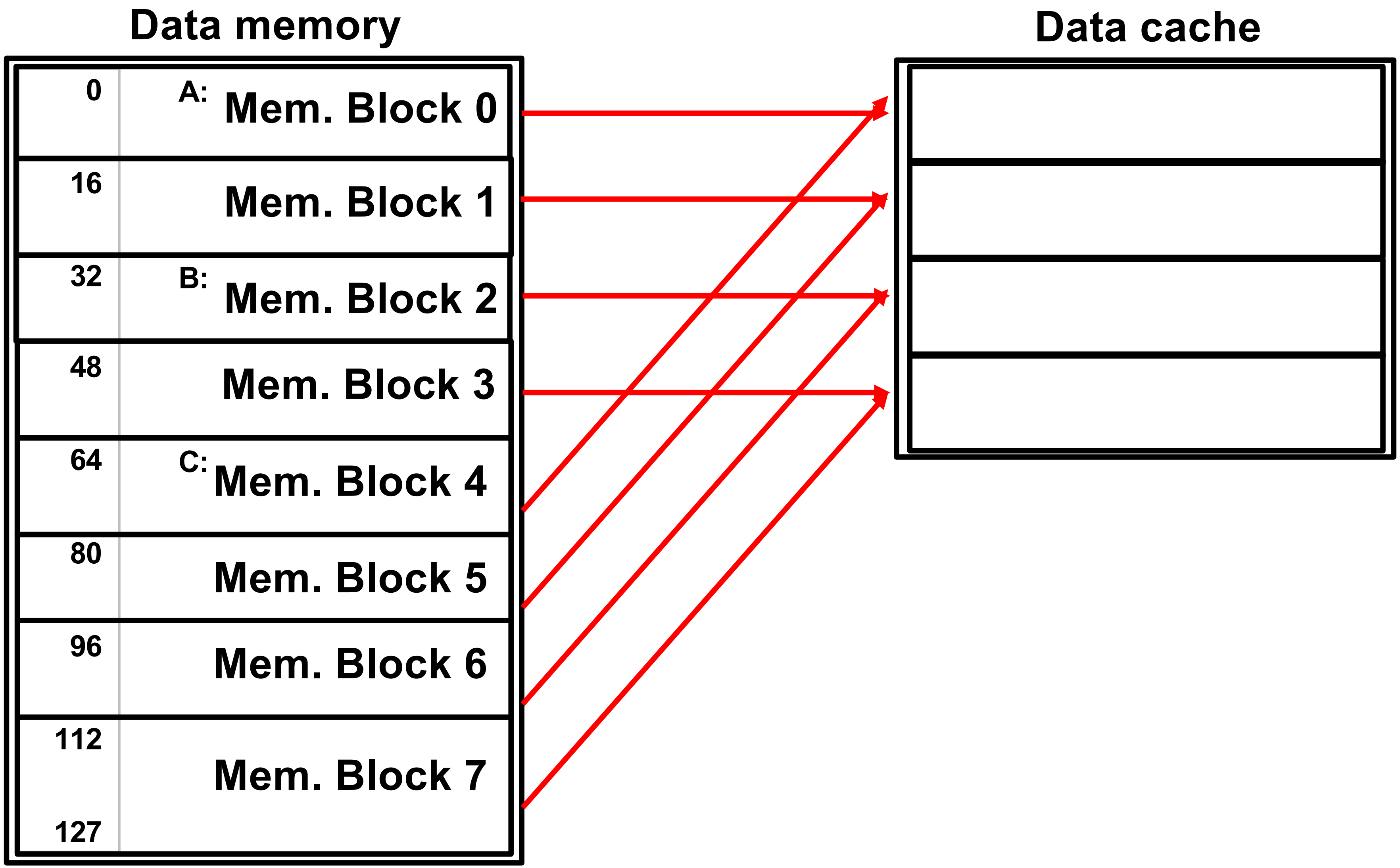
6 6	5 4	3 0
Tag	Block	Offset

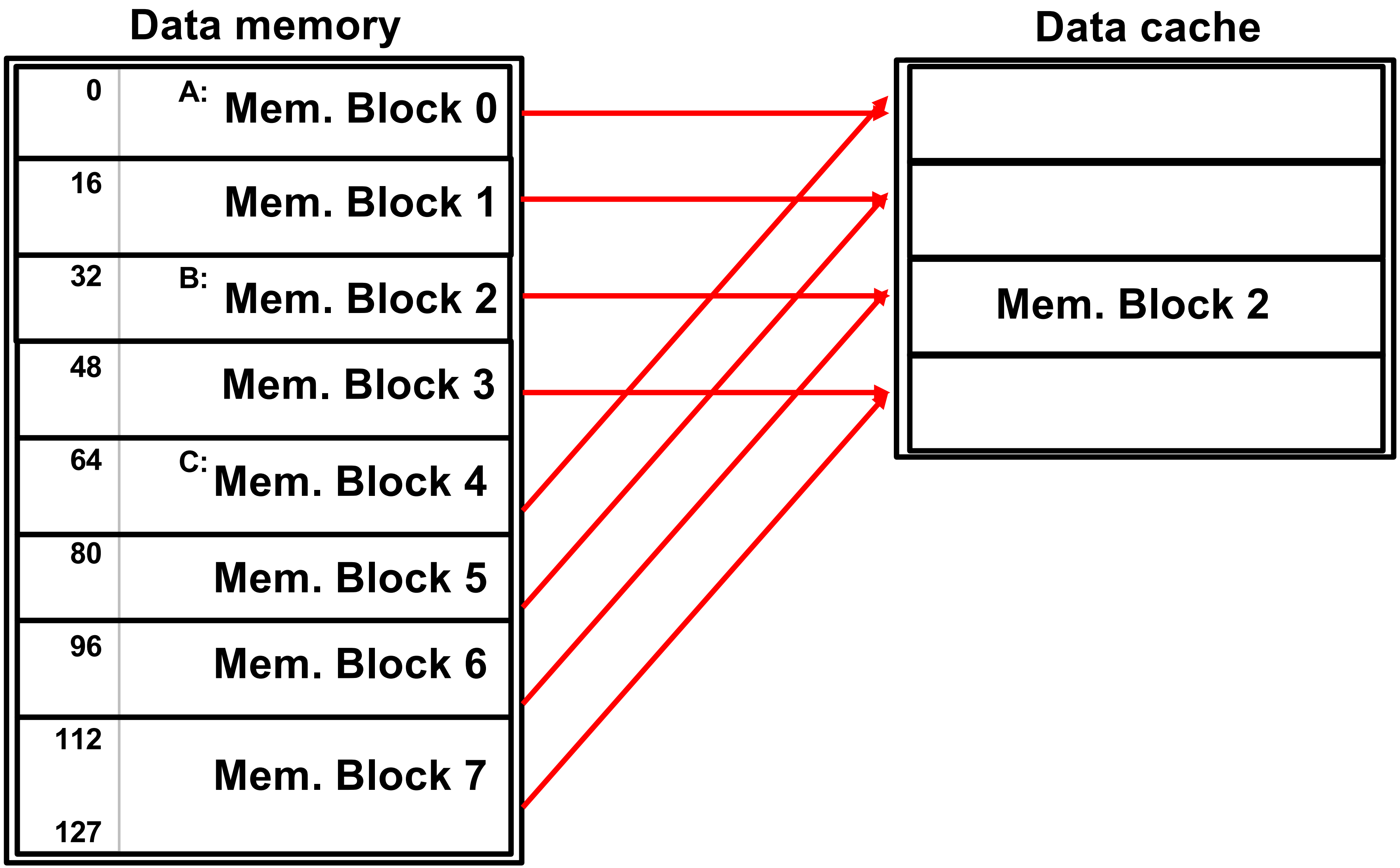
Question:
How many bits would be required for Tag, Block and offset for a memory of 1024 bytes divided into blocks of 32 bytes and a cache containing 4 blocks?

Answer:

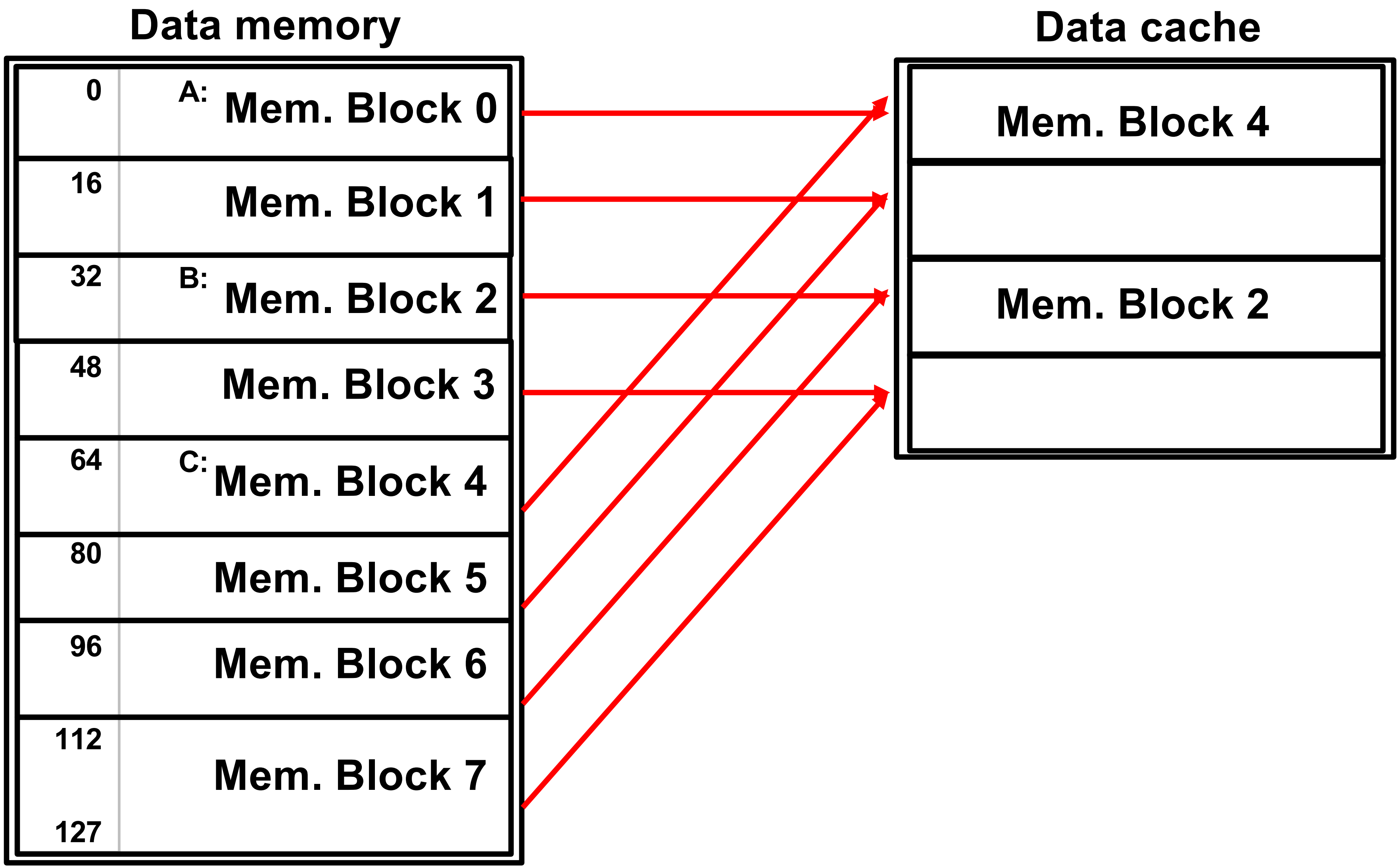
- Tag bits: $\log_2$(Memory size/Cache size) = $\log_2(1024/(4x32))$= 3 bits
- Block bits: 4 blocks, so 2 bits
- Offset bits: 32 bytes, so 5 bits
- Memory address: 10 bits

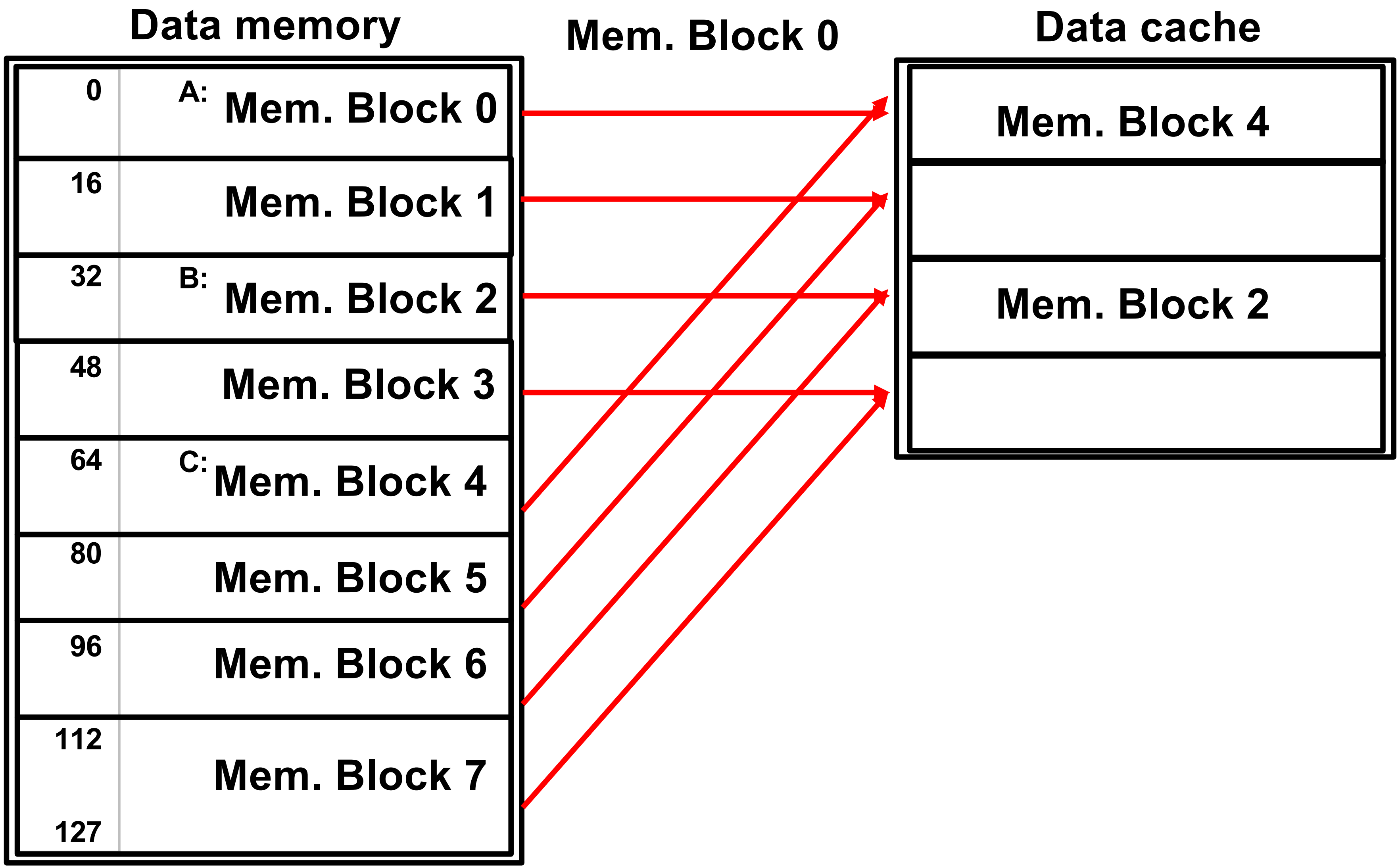
Set-Associative Caches

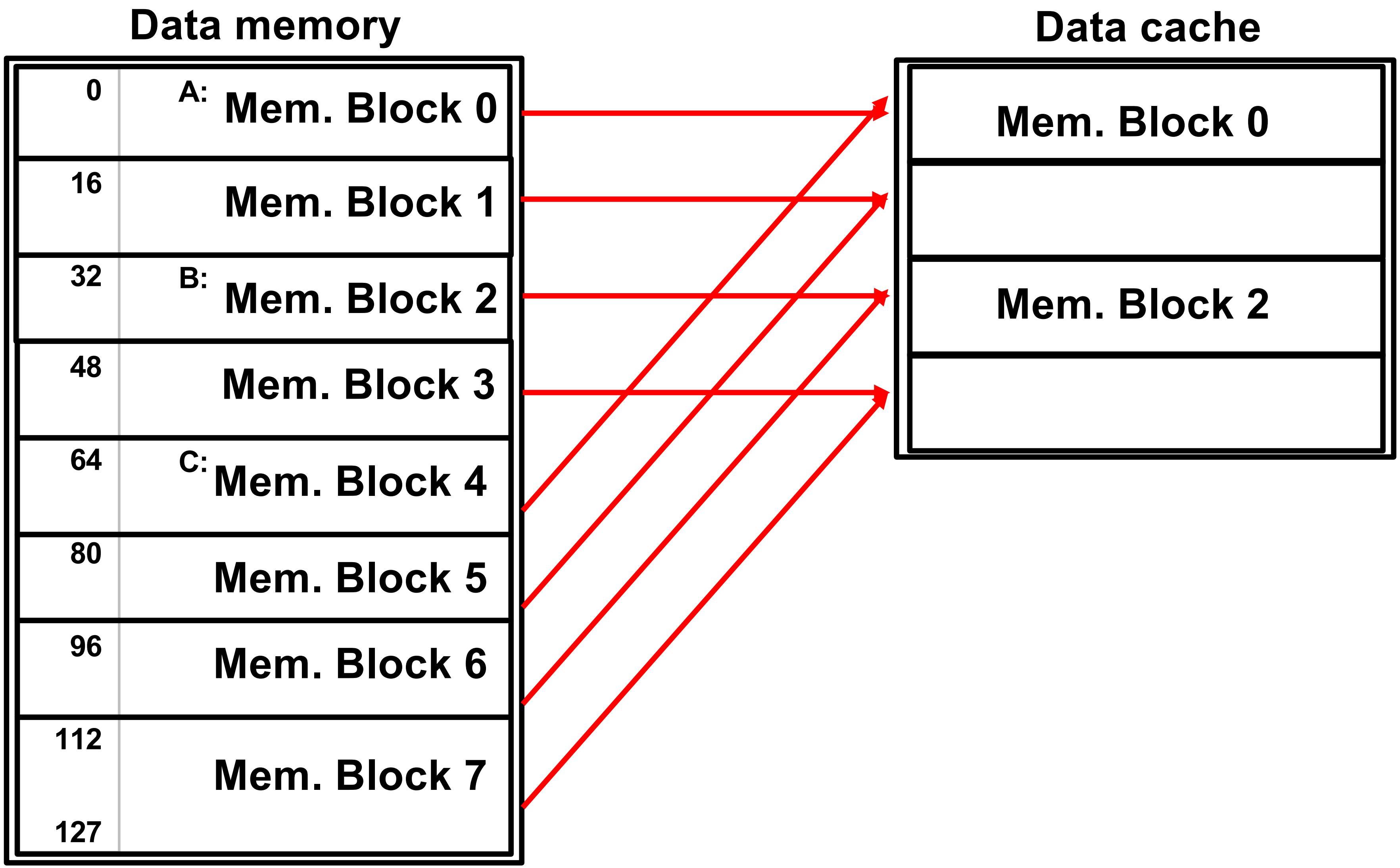

Data memory
Data cache
0
A:
Mem. Block 0
16
Mem. Block 1
32
B:
Mem. Block 2
48
Mem. Block 3
64
C:
Mem. Block 4
80
Mem. Block 5
96
Mem. Block 6
112
Mem. Block 7
127

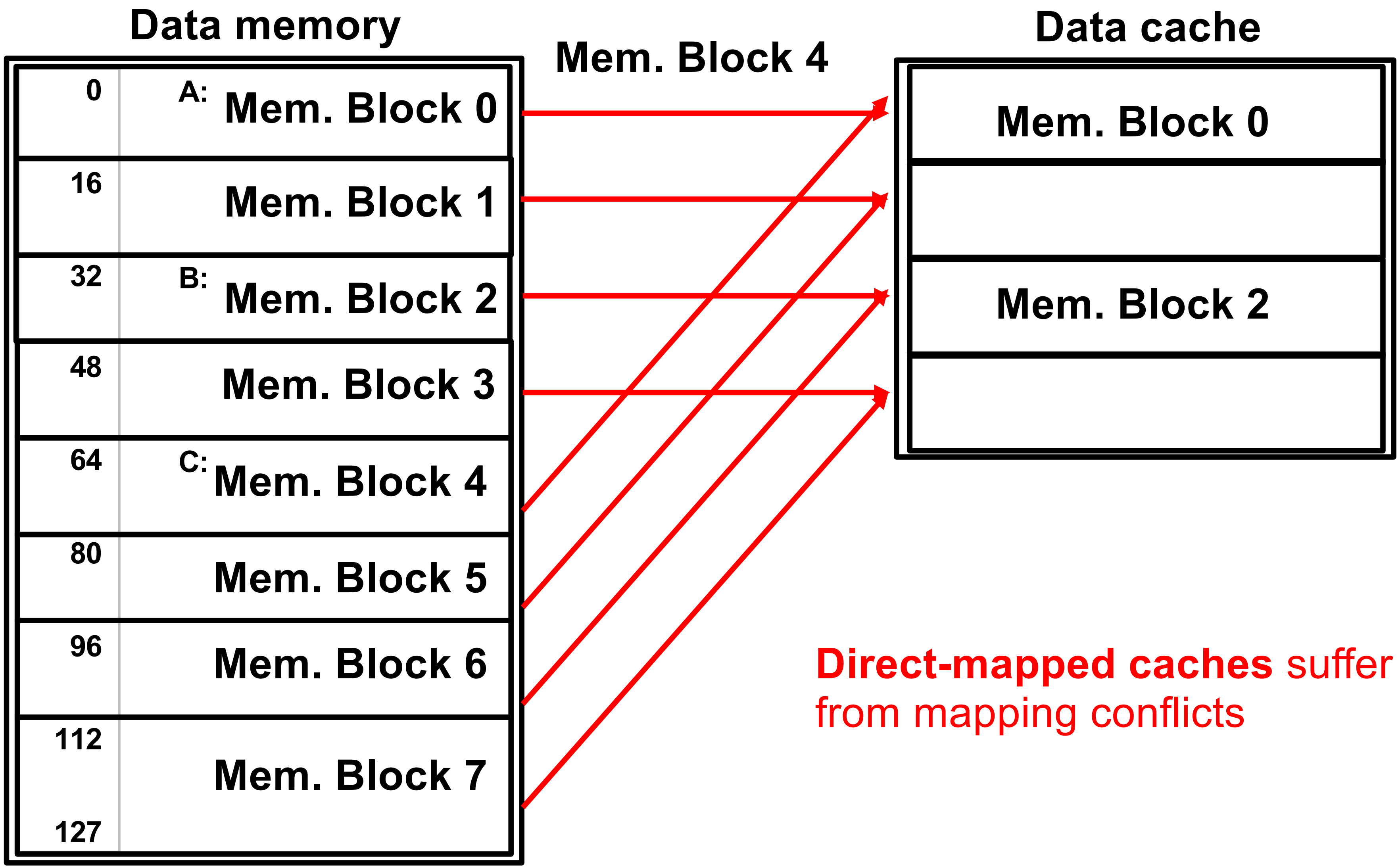


Data memory
Data cache
0
A:
Mem. Block 0
16
Mem. Block 1
32
B:
Mem. Block 2
48
Mem. Block 3
64
C:
Mem. Block 4
80
Mem. Block 5
96
Mem. Block 6
112
Mem. Block 7
127
Mem. Block 2

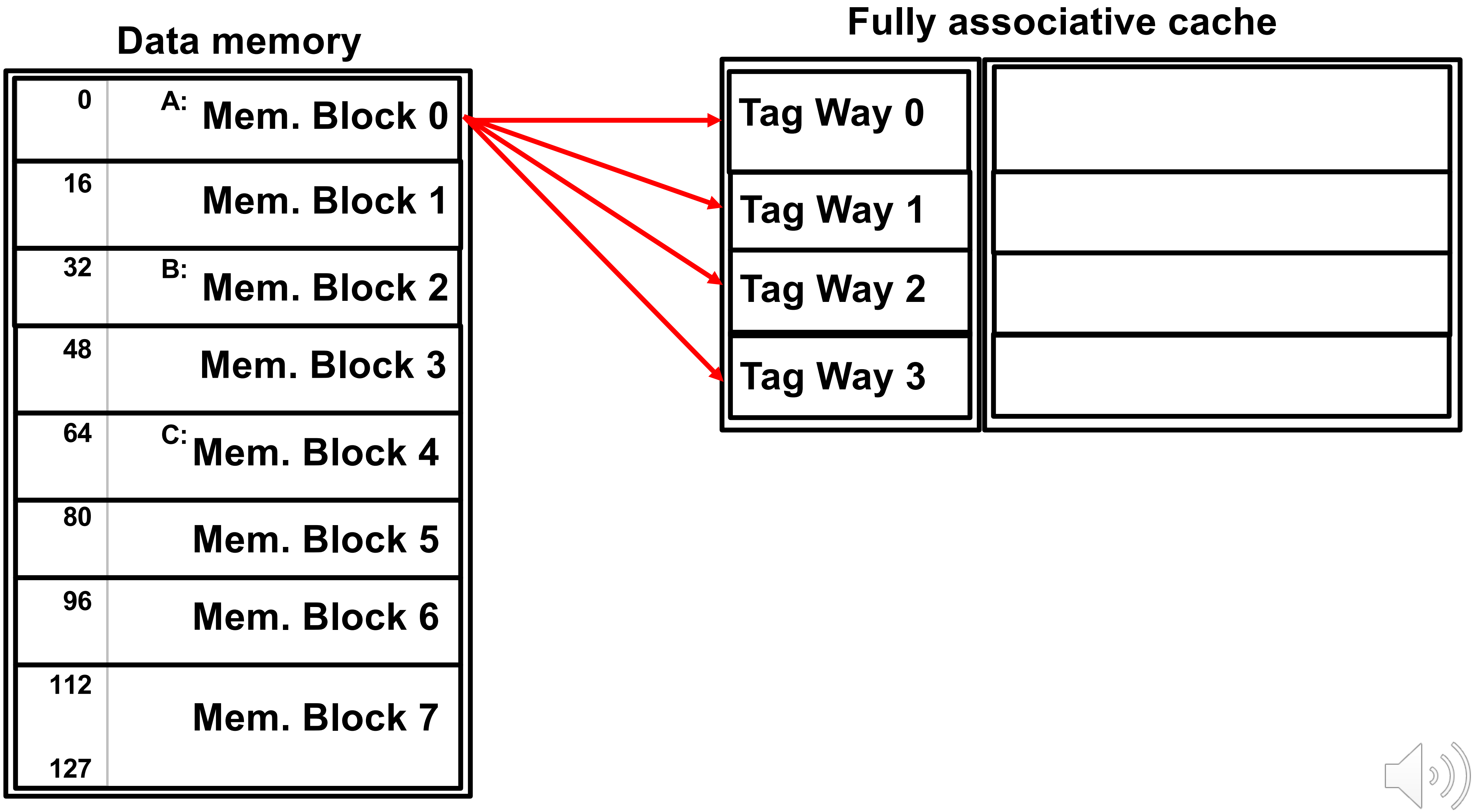

Data memory
Data cache
0
A:
Mem. Block 0
16
Mem. Block 1
32
B:
Mem. Block 2
48
Mem. Block 3
64
C:
Mem. Block 4
80
Mem. Block 5
96
Mem. Block 6
112
Mem. Block 7
127
Mem. Block 4
Mem. Block 2


Data memory
Mem. Block 0
Data cache
0
A:
Mem. Block 0
16
Mem. Block 1
32
B:
Mem. Block 2
48
Mem. Block 3
64
C:
Mem. Block 4
80
Mem. Block 5
96
Mem. Block 6
112
Mem. Block 7
127
Mem. Block 4
Mem. Block 2

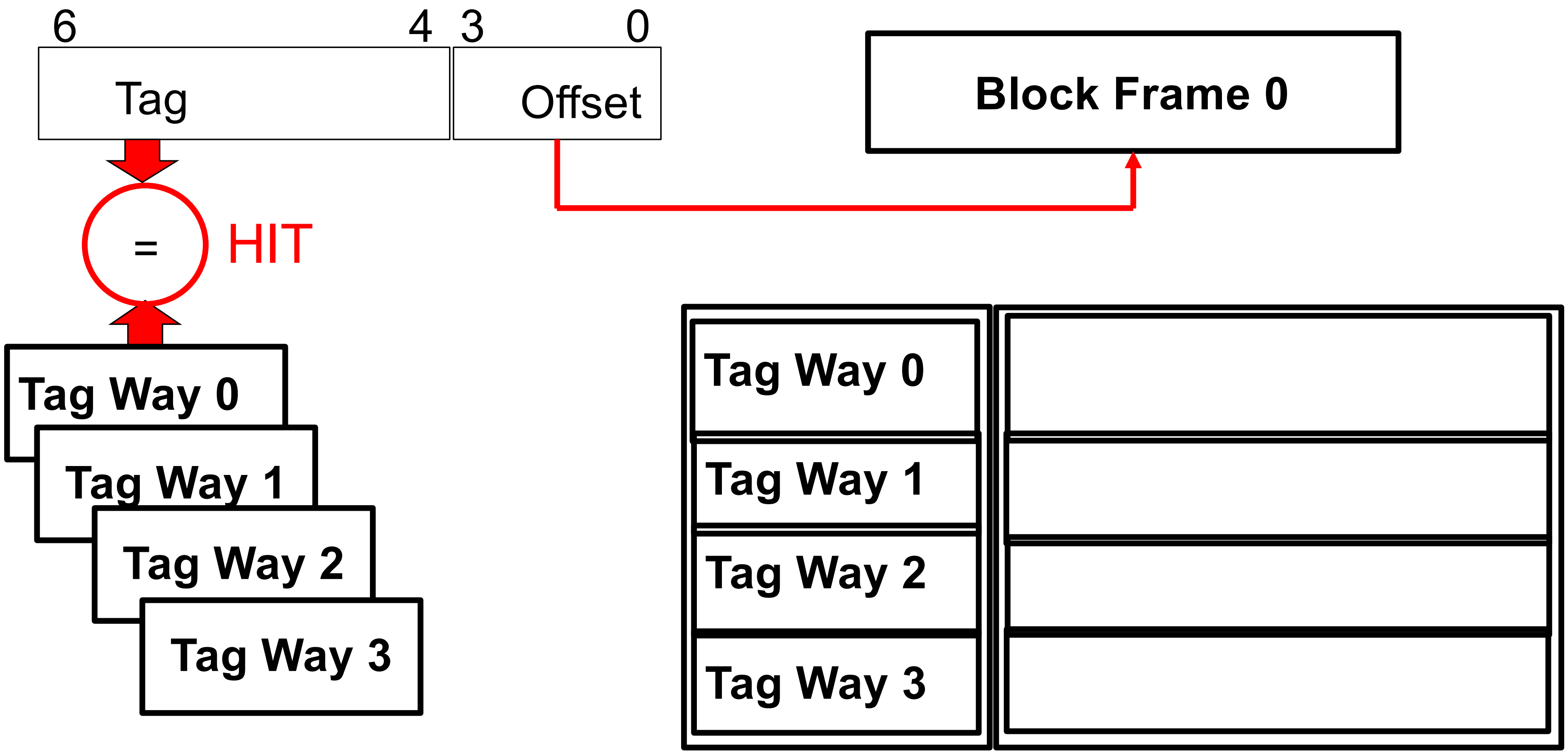

Data memory
Data cache
0
A:
Mem. Block 0
16
Mem. Block 1
32
B:
Mem. Block 2
48
Mem. Block 3
64
C:
Mem. Block 4
80
Mem. Block 5
96
Mem. Block 6
112
Mem. Block 7
127
Mem. Block 0
Mem. Block 2



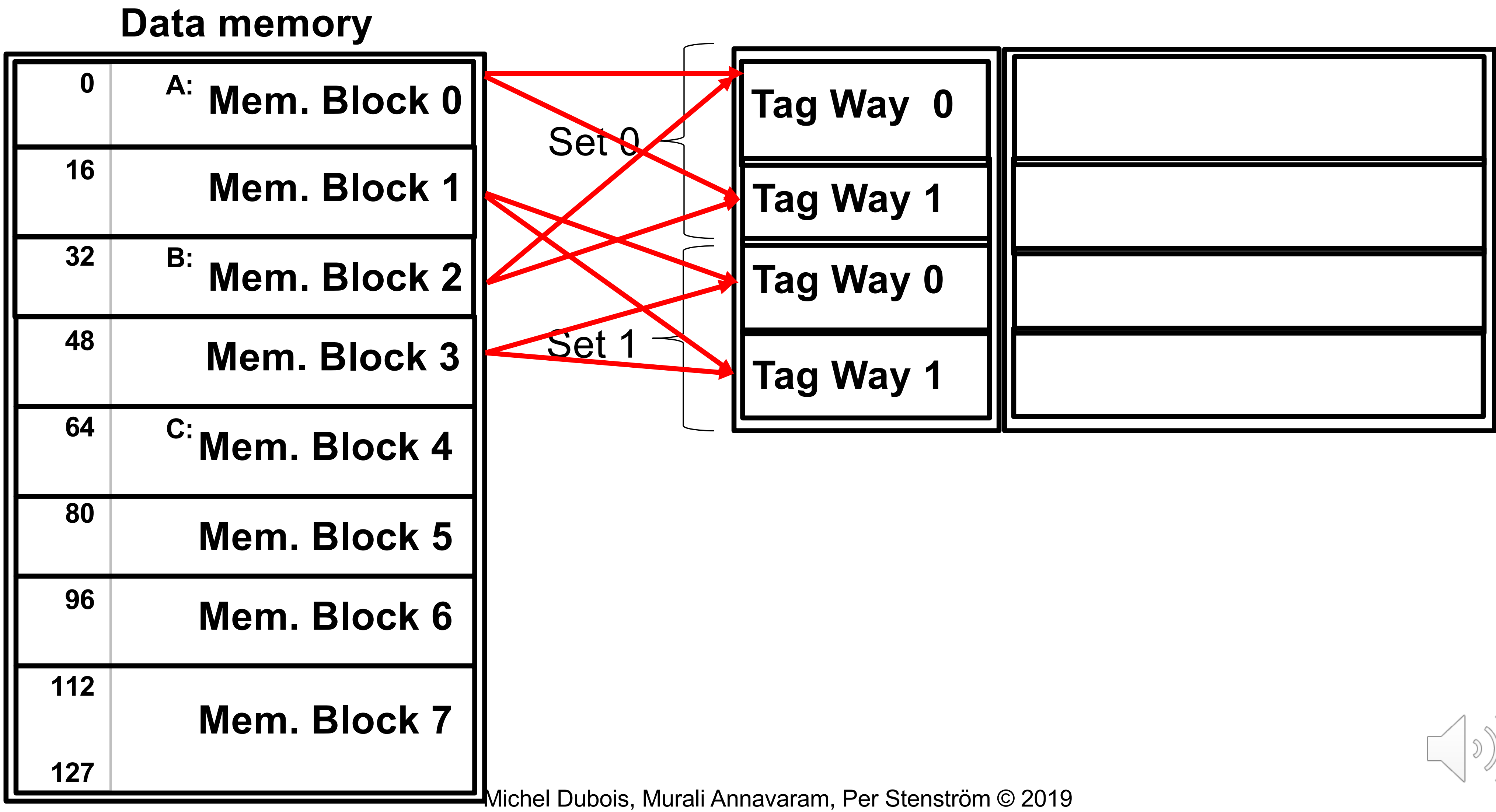
Direct-mapped caches suffer from mapping conflicts


Data memory
Fully associative cache
0
A:
Mem. Block 0
16
Mem. Block 1
32
B:
Mem. Block 2
48
Mem. Block 3
64
C:
Mem. Block 4
80
Mem. Block 5
96
Mem. Block 6
112
Mem. Block 7
127
Tag Way 0
Tag Way 1
Tag Way 2
Tag Way 3

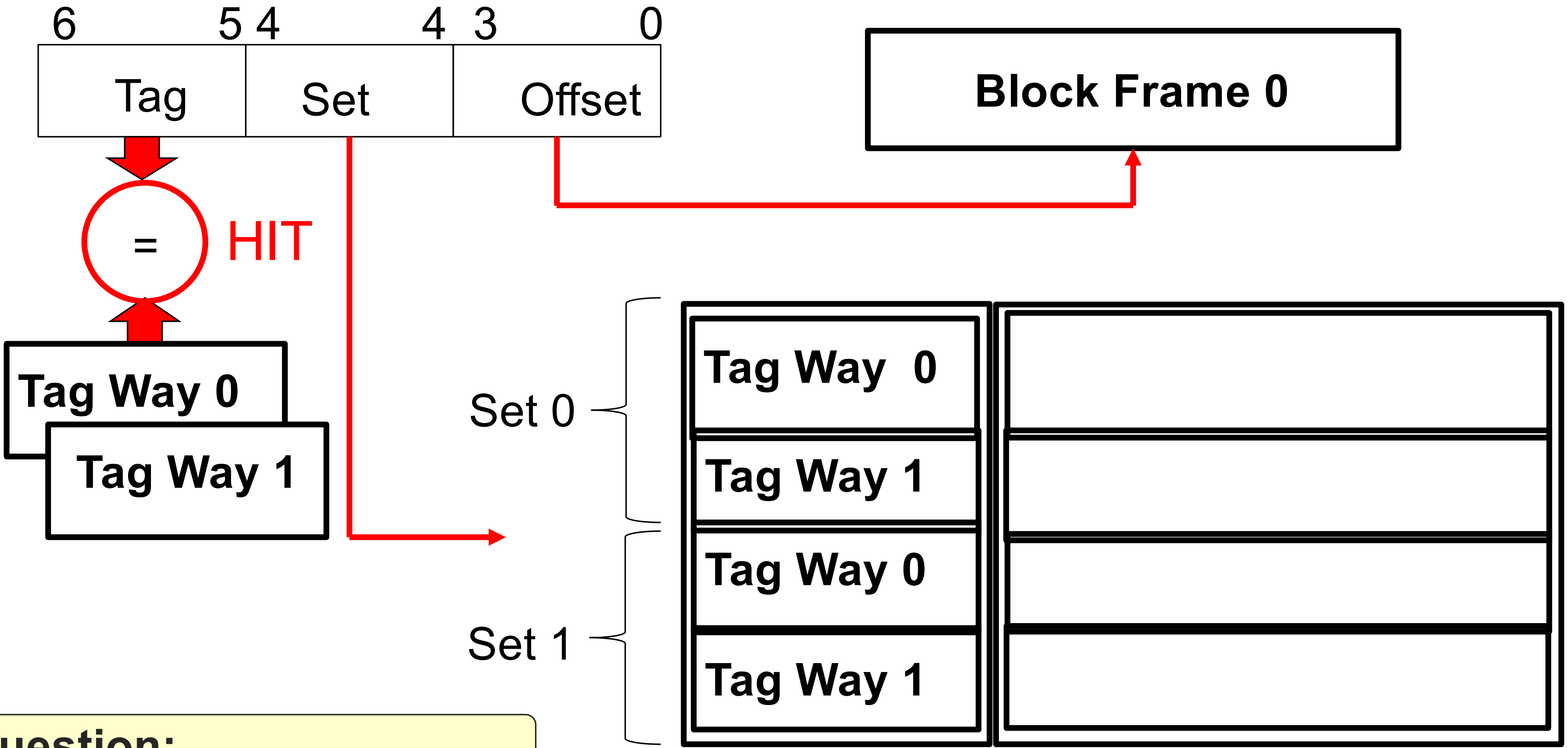
FULLY ASSOCIATIVE CACHE



2-WAY ASSOCIATIVE CACHE



2-WAY ASSOCIATIVE CACHE



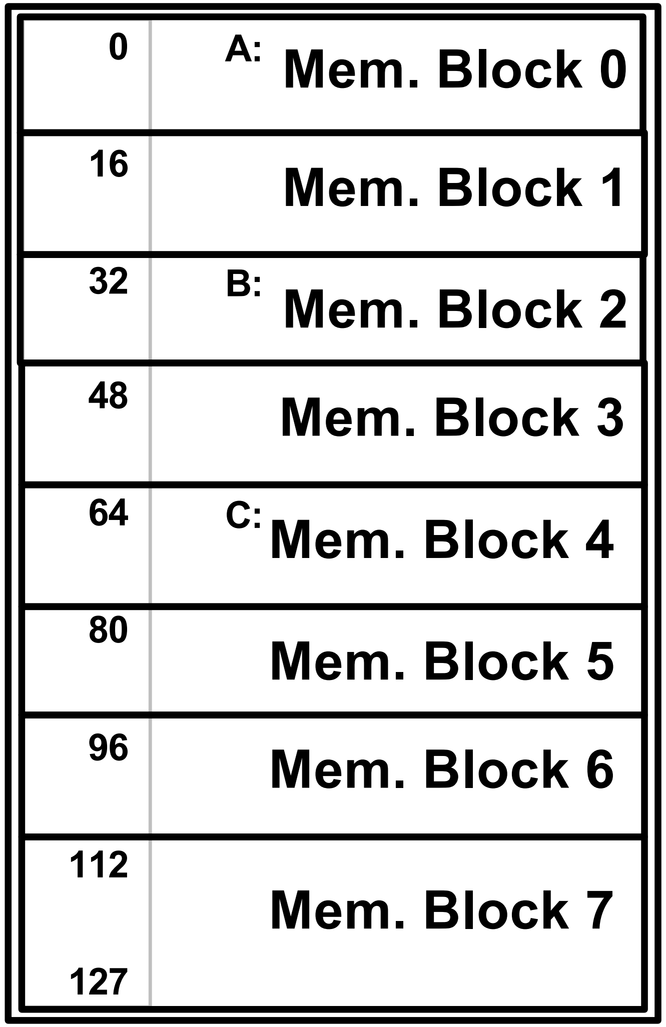
Question:
Why is the tag field 2 bits?

Answer:

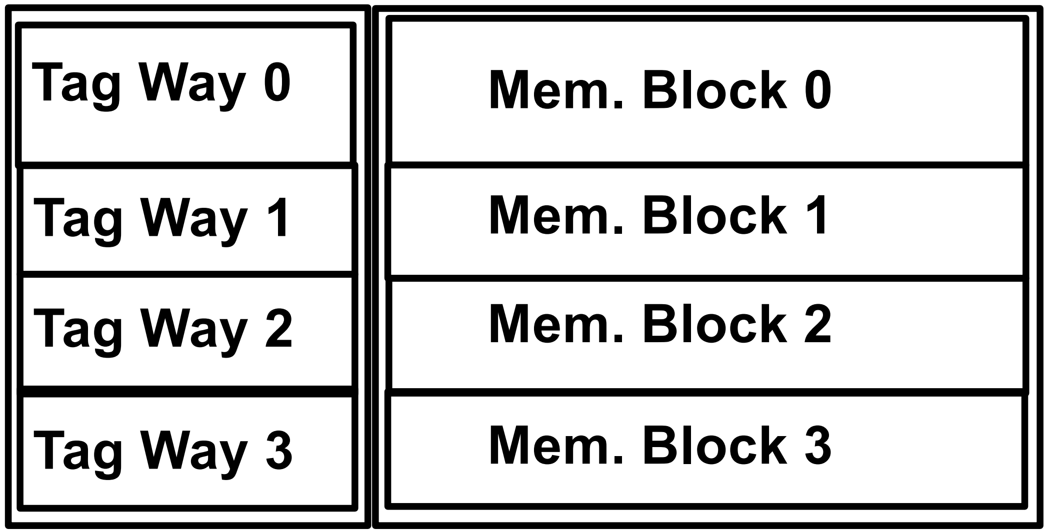
We have to determine how many blocks that can map to a given block frame. There are eight memory blocks and half of them will be mapped to Set 0 and half to Set 1. So there are four memory blocks that can be mapped to a given block frame.

Replacement Policies
(Ch 4.3.2)

Data memory



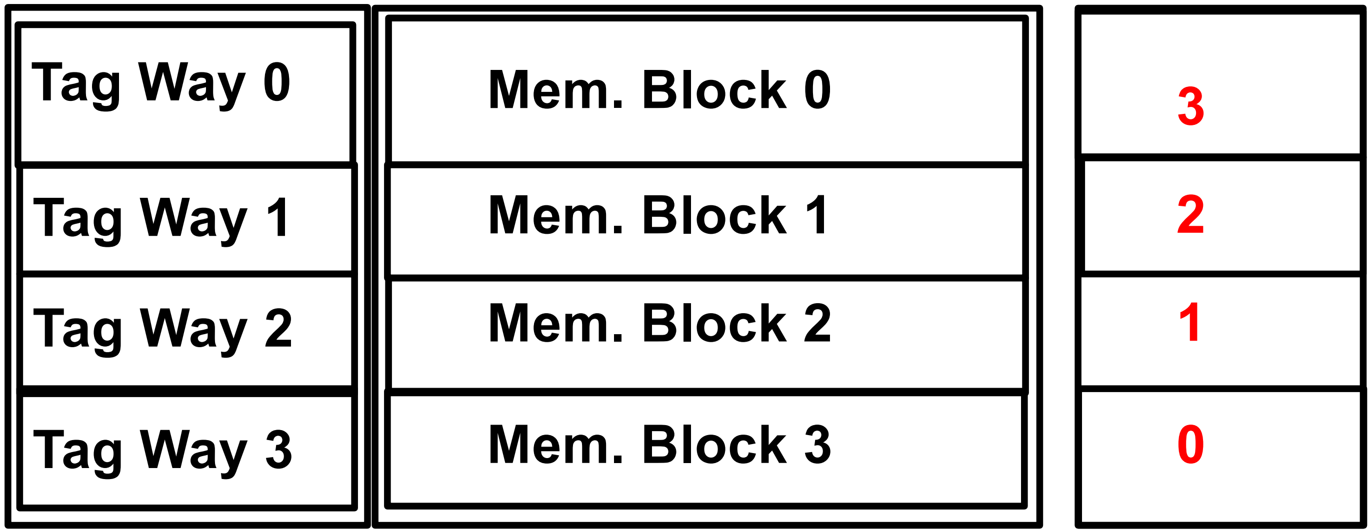
Fully associative cache



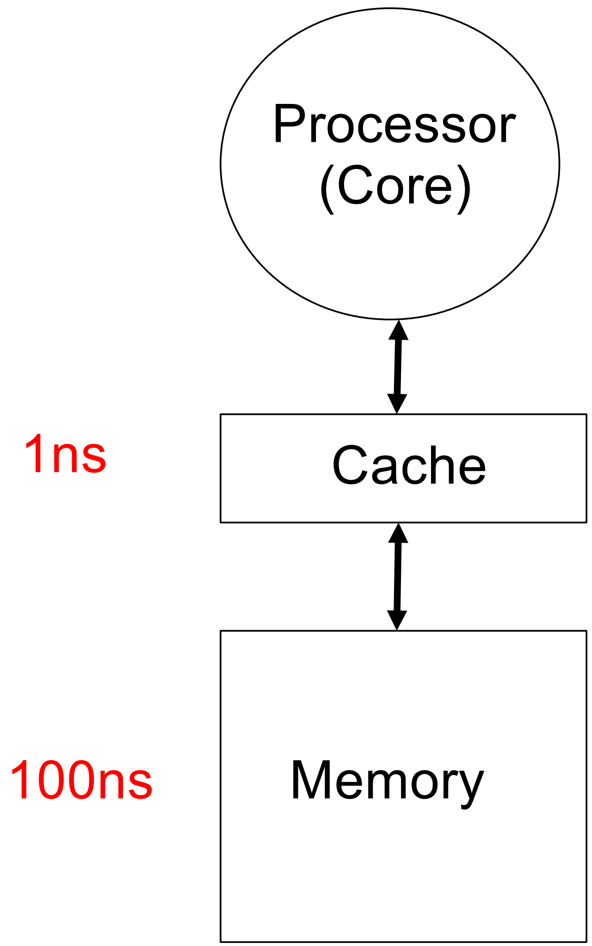
The Cache Replacement Algorithm determines which block should be replaced.

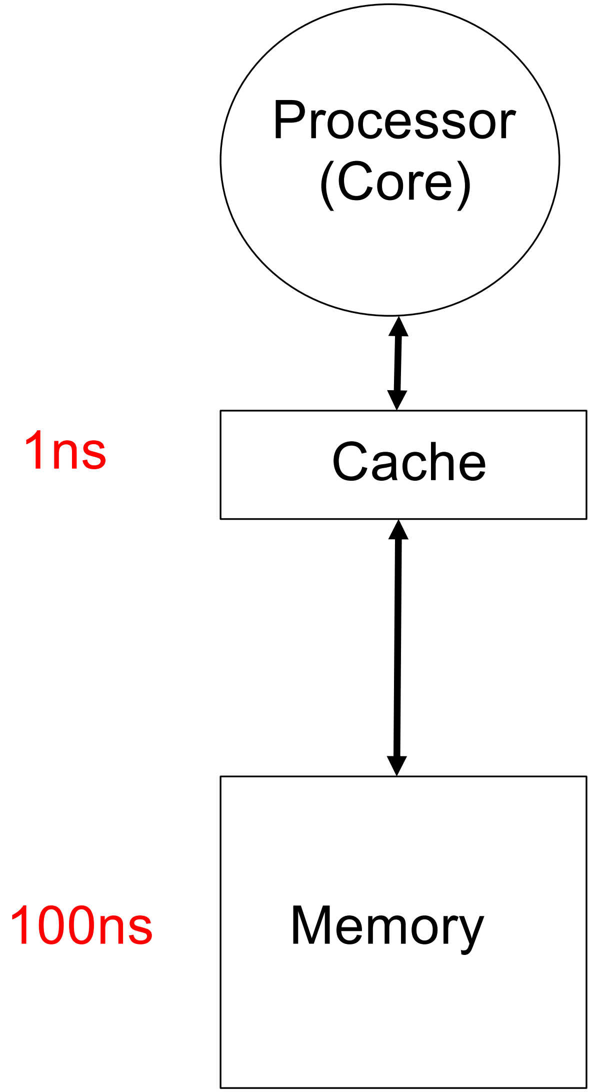
Least Recently Used (LRU)

Fully associative cache



Write Policies
(Ch 4.3.3)

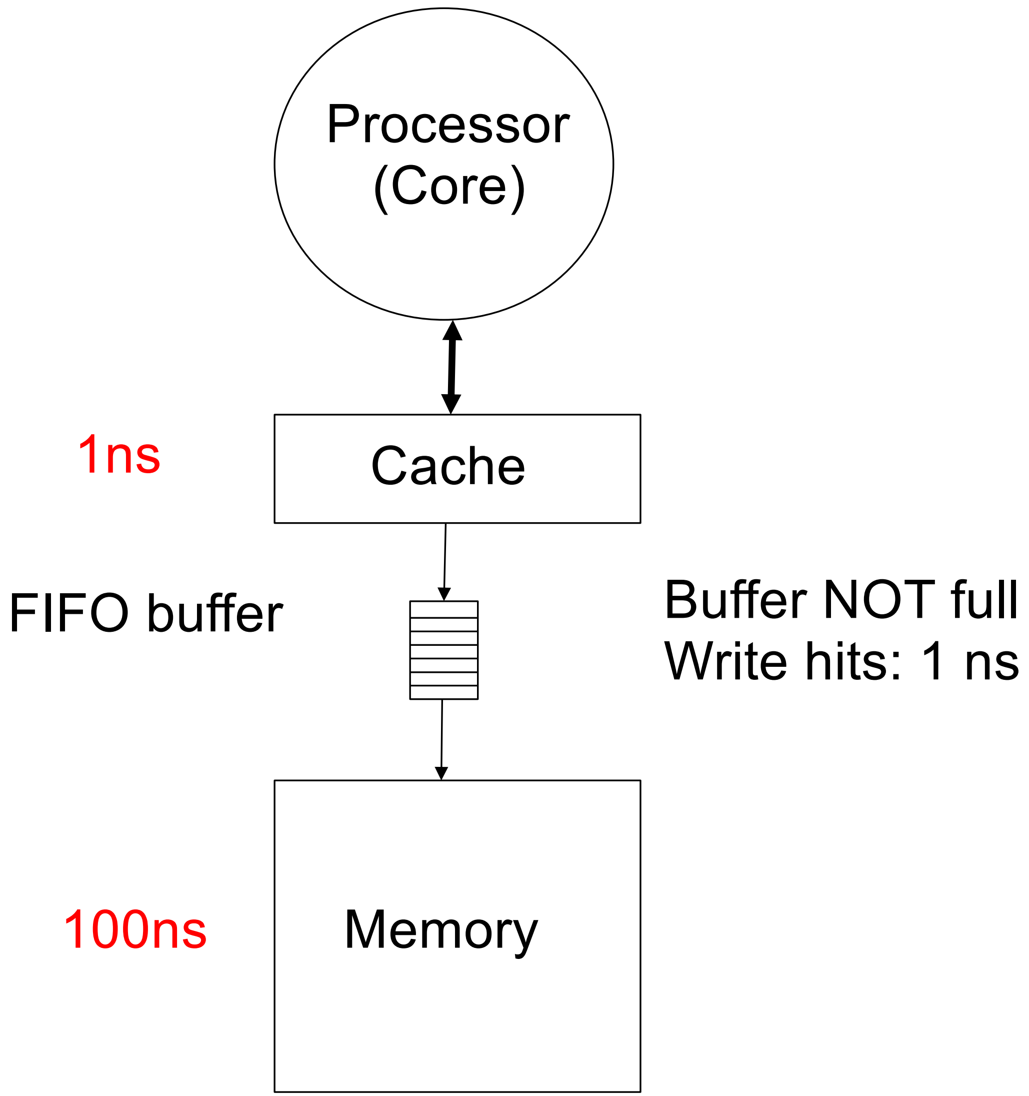

Processor
(Core)
1ns
Cache
100ns
Memory

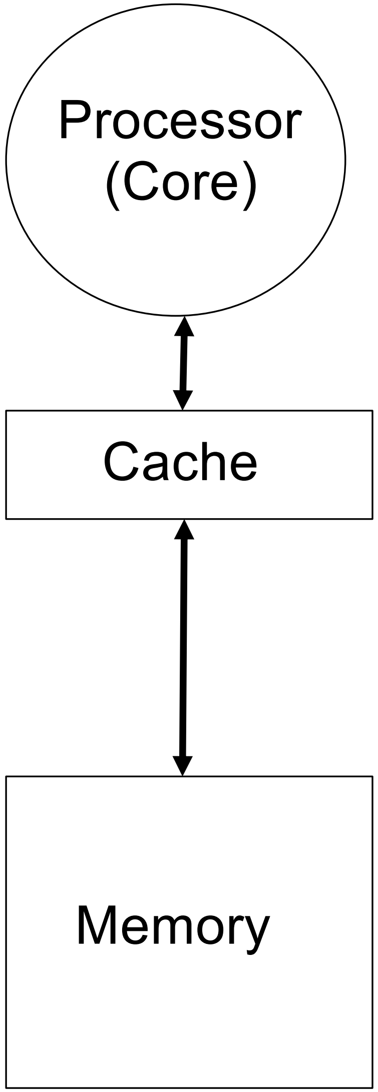


Read hits: 1ns
Read miss: 100 ns

Write hits: 100 ns
Write miss: 100 ns

THE WRITE-THROUGH POLICY


Processor
(Core)
1ns
Cache
FIFO buffer
Buffer NOT full
Write hits: 1 ns
100ns
Memory



1ns

100ns

Read hits: 1ns
Read miss: 100 ns

Write hits: 1 ns
Write miss: 100 ns

THE WRITE-BACK POLICY

NEED TO KEEP TRACK OF MODIFIED BLOCKS

Cache Performance
(Ch 4.3.4)

Instruction memory

0	LABEL: LD R2,0(R10)
4	LD R3,0(R11)
8	ADD R1,R2,R3
12	SD 0(R12),R1
16	SUBI R4,R4,#1
20	ADDI R10,R10,#4
24	ADDI R11,R11,#4
28	ADDI R12,R12,#4
32	BNEZ R4,LABEL
36	
40	
44	
48	
…	
…	
124	
127	

Direct-mapped Instruction cache

Block frame 0
Block frame 1
Block frame 2
Block frame 3

Instruction memory

0	LABEL: LD R2,0(R10)
4	LD R3,0(R11)
8	ADD R1,R2,R3
12	SD 0(R12),R1
16	SUBI R4,R4,#1
20	ADDI R10,R10,#4
24	ADDI R11,R11,#4
28	ADDI R12,R12,#4
32	BNEZ R4,LABEL
36	
40	
44	
48	
…	
…	
124	
127	

Direct-mapped Instruction cache

LABEL: LD R2,0(R10)
LD R3,0(R11)
ADD R1,R2,R3
SD 0(R12),R1

Block frame 1
Block frame 2
Block frame 3

1 MISS
3 HITS

Instruction memory

0	LABEL: LD R2,0(R10)
4	LD R3,0(R11)
8	ADD R1,R2,R3
12	SD 0(R12),R1
16	SUBI R4,R4,#1
20	ADDI R10,R10,#4
24	ADDI R11,R11,#4
28	ADDI R12,R12,#4
32	BNEZ R4,LABEL
36	
40	
44	
48	
…	
…	
124	
127	

Direct-mapped Instruction cache

LABEL: LD R2,0(R10)
LD R3,0(R11)
ADD R1,R2,R3
SD 0(R12),R1

SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4

Block frame 2
Block frame 3

1 MISS
3 HITS

Instruction memory

0	LABEL: LD R2,0(R10)
4	LD R3,0(R11)
8	ADD R1,R2,R3
12	SD 0(R12),R1
16	SUBI R4,R4,#1
20	ADDI R10,R10,#4
24	ADDI R11,R11,#4
28	ADDI R12,R12,#4
32	BNEZ R4,LABEL
36	
40	
44	
48	
…	
…	
124	
127	

Direct-mapped Instruction cache

LABEL: LD R2,0(R10)
LD R3,0(R11)
ADD R1,R2,R3
SD 0(R12),R1

SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4

BNEZ R4,LABEL
Block frame 3

1 MISS
0 HITS

Instruction memory

0	LABEL: LD R2,0(R10)
4	LD R3,0(R11)
8	ADD R1,R2,R3
12	SD 0(R12),R1
16	SUBI R4,R4,#1
20	ADDI R10,R10,#4
24	ADDI R11,R11,#4
28	ADDI R12,R12,#4
32	BNEZ R4,LABEL
36	
40	
44	
48	
…	
…	
124	
127	

Direct-mapped Instruction cache

LABEL: LD R2,0(R10)
LD R3,0(R11)
ADD R1,R2,R3
SD 0(R12),R1

SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4

BNEZ R4,LABEL
Block frame 3

4 HITS

Instruction memory

0	LABEL: LD R2,0(R10)
4	LD R3,0(R11)
8	ADD R1,R2,R3
12	SD 0(R12),R1
16	SUBI R4,R4,#1
20	ADDI R10,R10,#4
24	ADDI R11,R11,#4
28	ADDI R12,R12,#4
32	BNEZ R4,LABEL
36	
40	
44	
48	
…	
…	
124	
127	

Direct-mapped Instruction cache

LABEL: LD R2,0(R10)
LD R3,0(R11)
ADD R1,R2,R3
SD 0(R12),R1

SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4

BNEZ R4,LABEL
Block 3

4 HITS

Instruction memory

0	LABEL: LD R2,0(R10)
4	LD R3,0(R11)
8	ADD R1,R2,R3
12	SD 0(R12),R1
16	SUBI R4,R4,#1
20	ADDI R10,R10,#4
24	ADDI R11,R11,#4
28	ADDI R12,R12,#4
32	BNEZ R4,LABEL
36	
40	
44	
48	
…	
…	
124	
127	

Direct-mapped Instruction cache

LABEL: LD R2,0(R10)
LD R3,0(R11)
ADD R1,R2,R3
SD 0(R12),R1

SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4

BNEZ R4,LABEL

Block frame 3

1 HIT

Number of instruction fetches: 100 x 9 = 900
Number of MISSES: 3
Number of HITS: 897

Data memory

0	A:	Block 0
16		Block 1
32	B:	Block 2
48		Block 3
64	C:	Block 4
80		Block 5
96		Block 6
112 – 127		Block 7

4-Way Set Associative Data cache (4 Blocks)

Question:
Why is a 4-way associative cache with this configuration equivalent with a fully associative cache?

Answer:
Because the cache is configured with four blocks only.

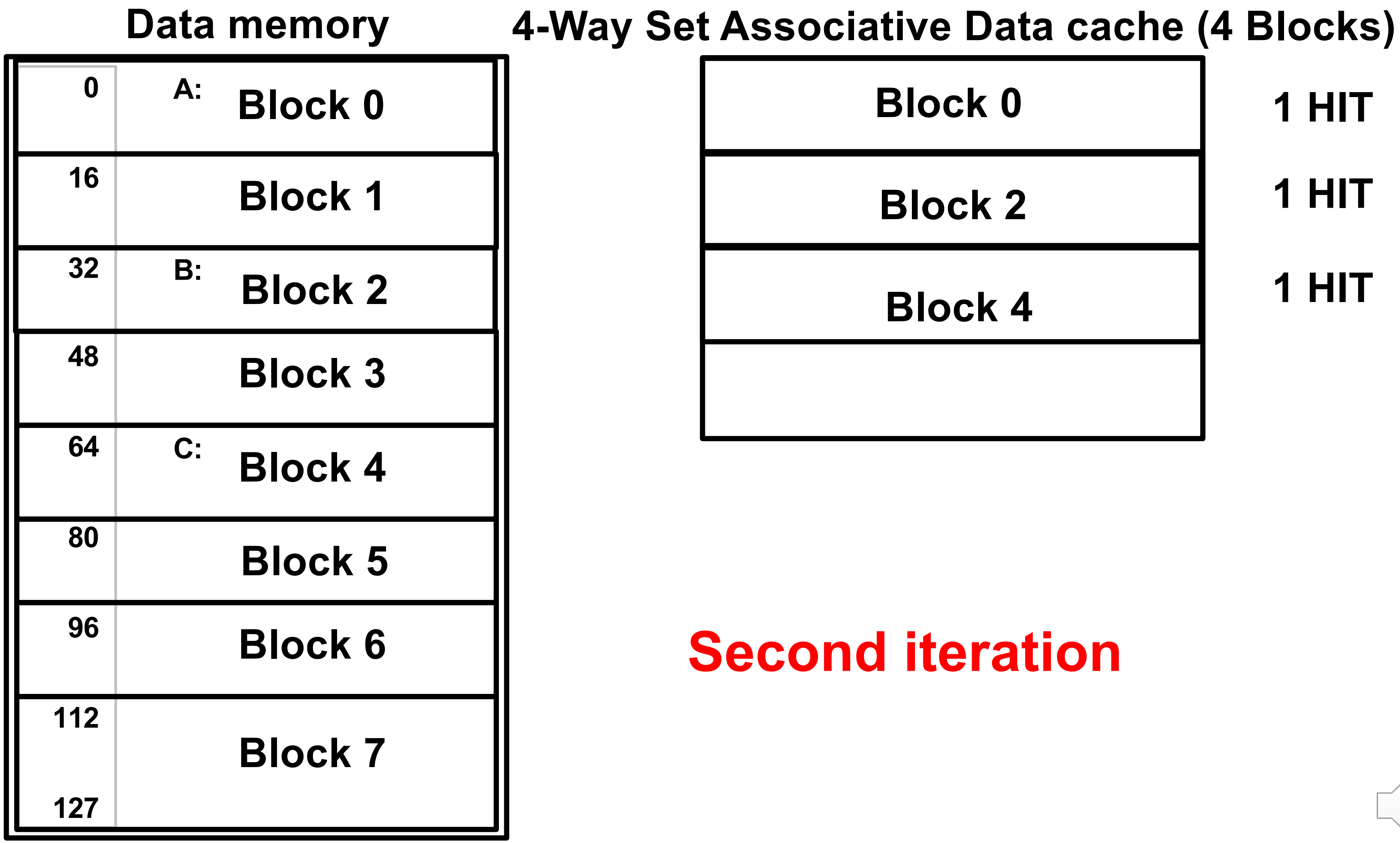
Data memory

0	A:	Block 0
16		Block 1
32	B:	Block 2
48		Block 3
64	C:	Block 4
80		Block 5
96		Block 6
112 127		Block 7

4-Way Set Associative Data cache (4 Blocks)

Block 0	1 MISS
Block 2	1 MISS
Block 4	1 MISS

First iteration


Data memory
4-Way Set Associative Data cache (4 Blocks)
0
A:
Block 0
16
Block 1
32
B:
Block 2
48
Block 3
64
C:
Block 4
80
Block 5
96
Block 6
112
Block 7
127
Block 0
1 HIT
Block 2
1 HIT
Block 4
1 HIT
Second iteration

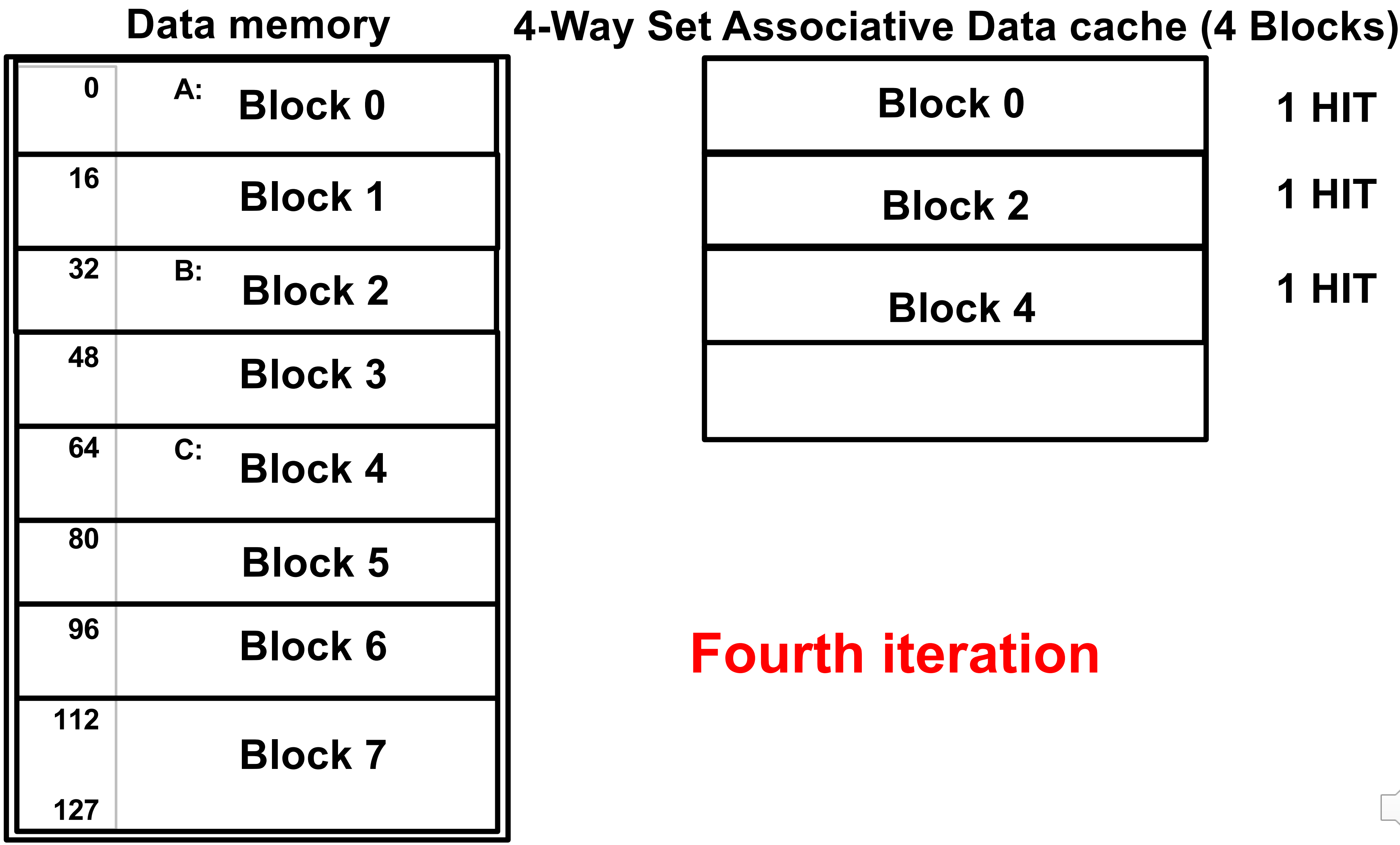
Data memory

0	A:	Block 0
16		Block 1
32	B:	Block 2
48		Block 3
64	C:	Block 4
80		Block 5
96		Block 6
112 127		Block 7

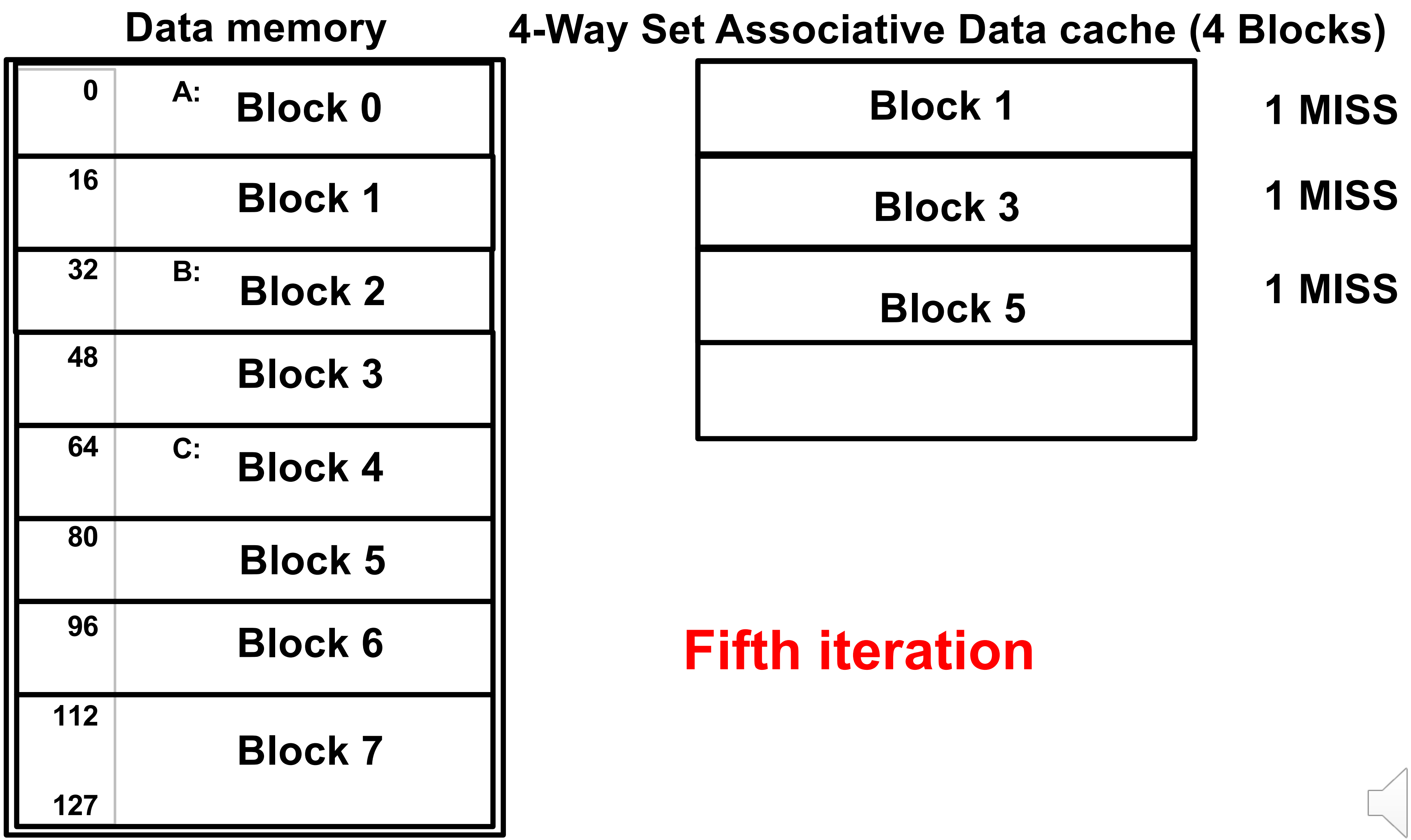
4-Way Set Associative Data cache (4 Blocks)

Block 0	1 HIT
Block 2	1 HIT
Block 4	1 HIT

Third iteration

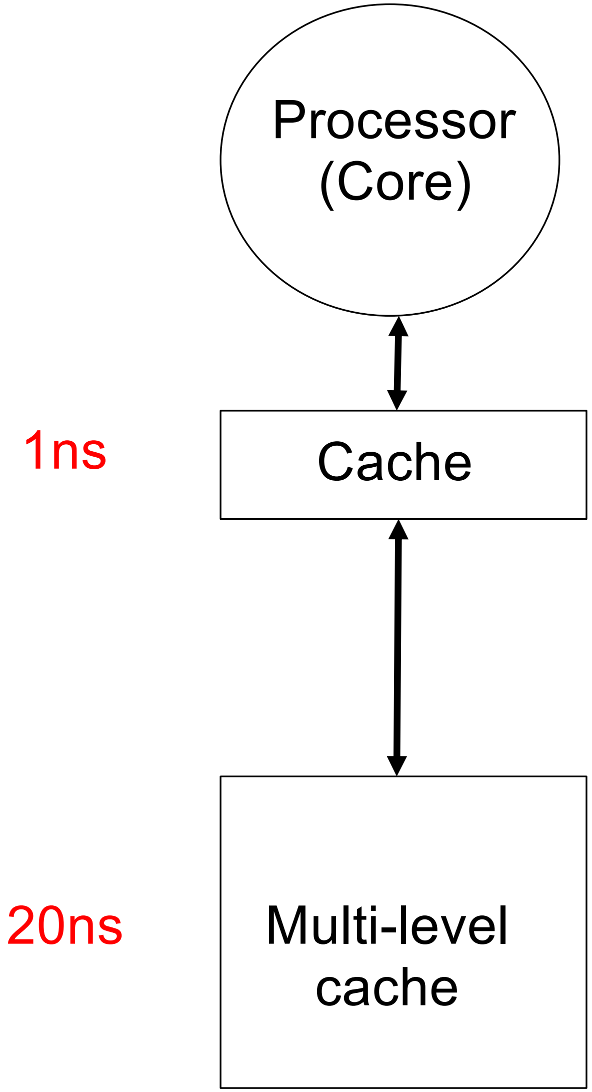


Fourth iteration



Fifth iteration

Number of data accesses: 100 x 3 = 300
Number of MISSES: 75
Number of HITS: 225



Read hits: 1ns
Read miss: 50 ns

Write hits: 1 ns
Write miss: 50 ns

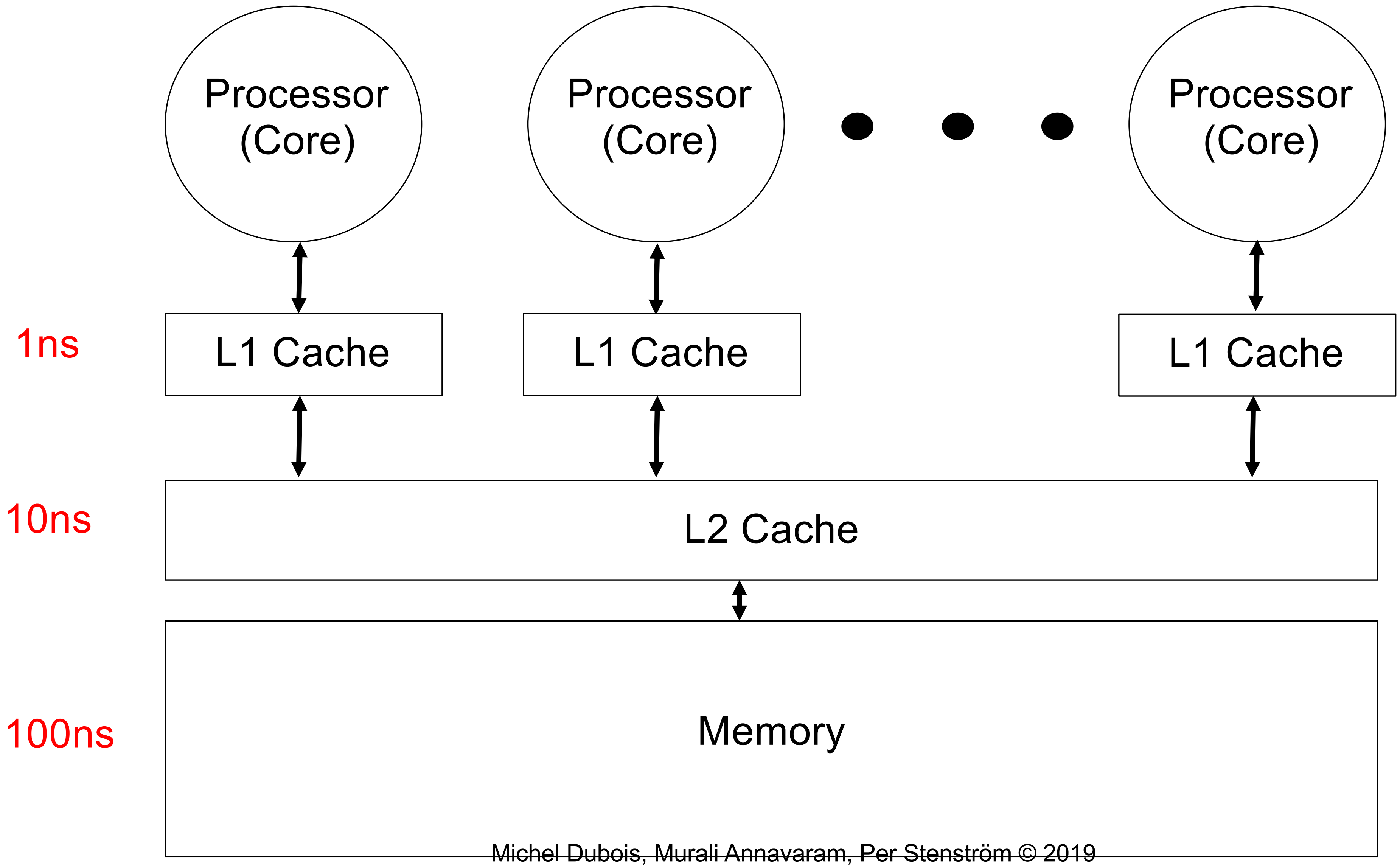
$$T = IC \times (CPI_0 + MPI \times MP) \times TPC$$

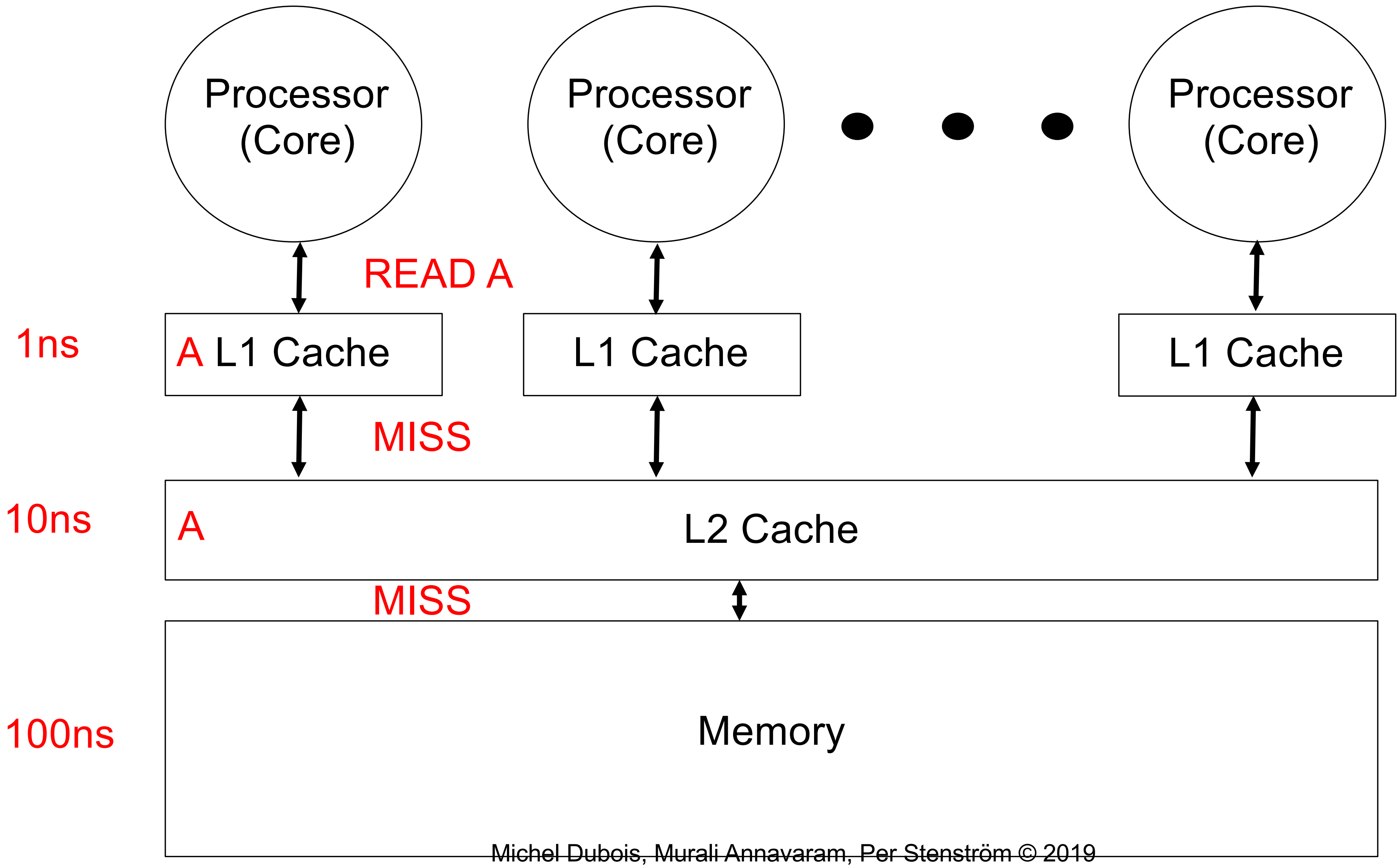
Total number of misses: 3+75 = 78
Total number of instructions: 900
Miss rate Per Instruction (MPI): 78/900 = 0.087
CPI =1 + 0.087 x 20 =2,74

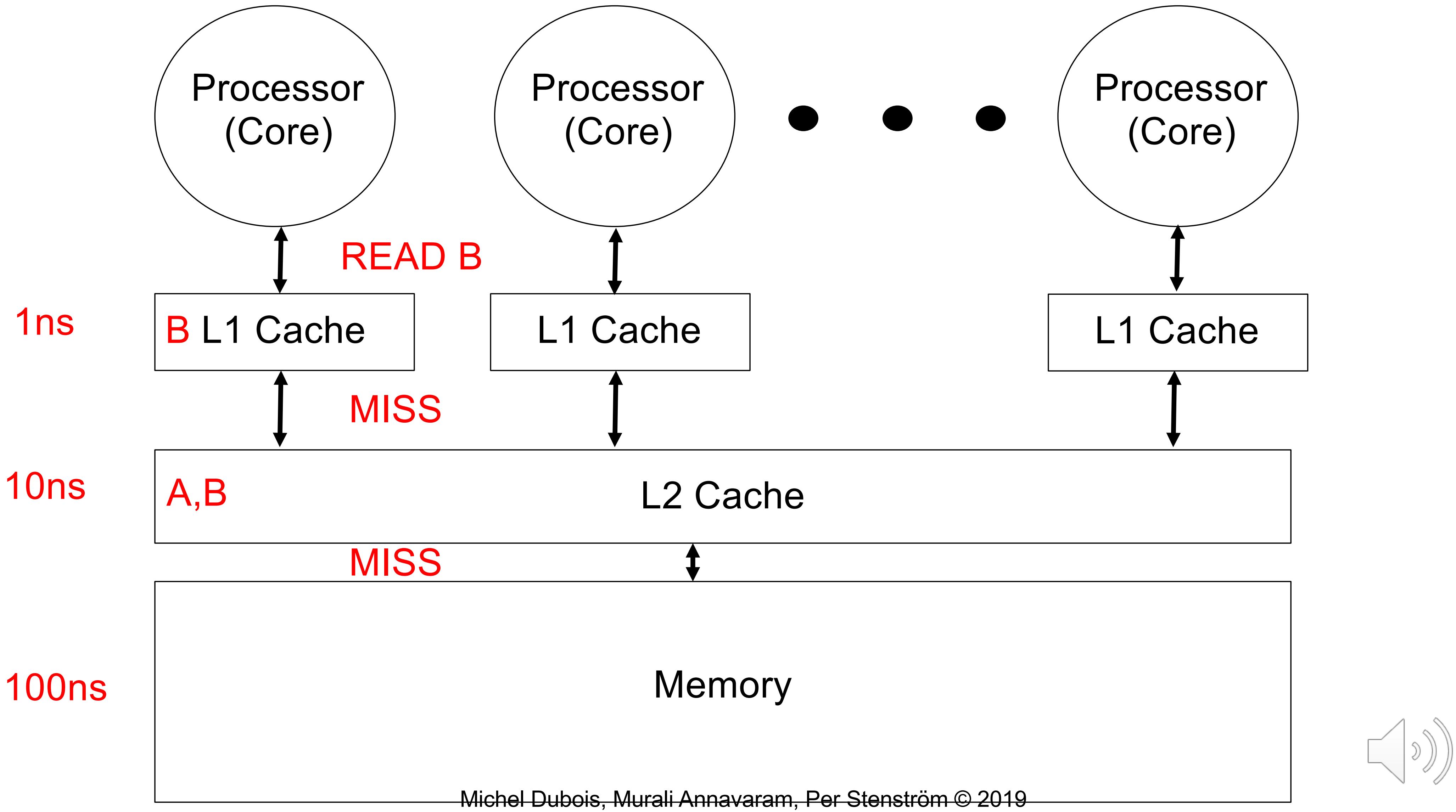
```
LABEL: LD R2, 0(R10)
       LD R3, 0(R11)
       UNUSED
       ADD R1,R2,R3
       SD 0(R12), R1
       SUBI R4,R4,#1
       ADDI R10,R10,#4
       ADDI R11,R11,#4
       ADDI R12,R12,#4
       BNEZ R4, LABEL
       UNUSED
       UNUSED
```

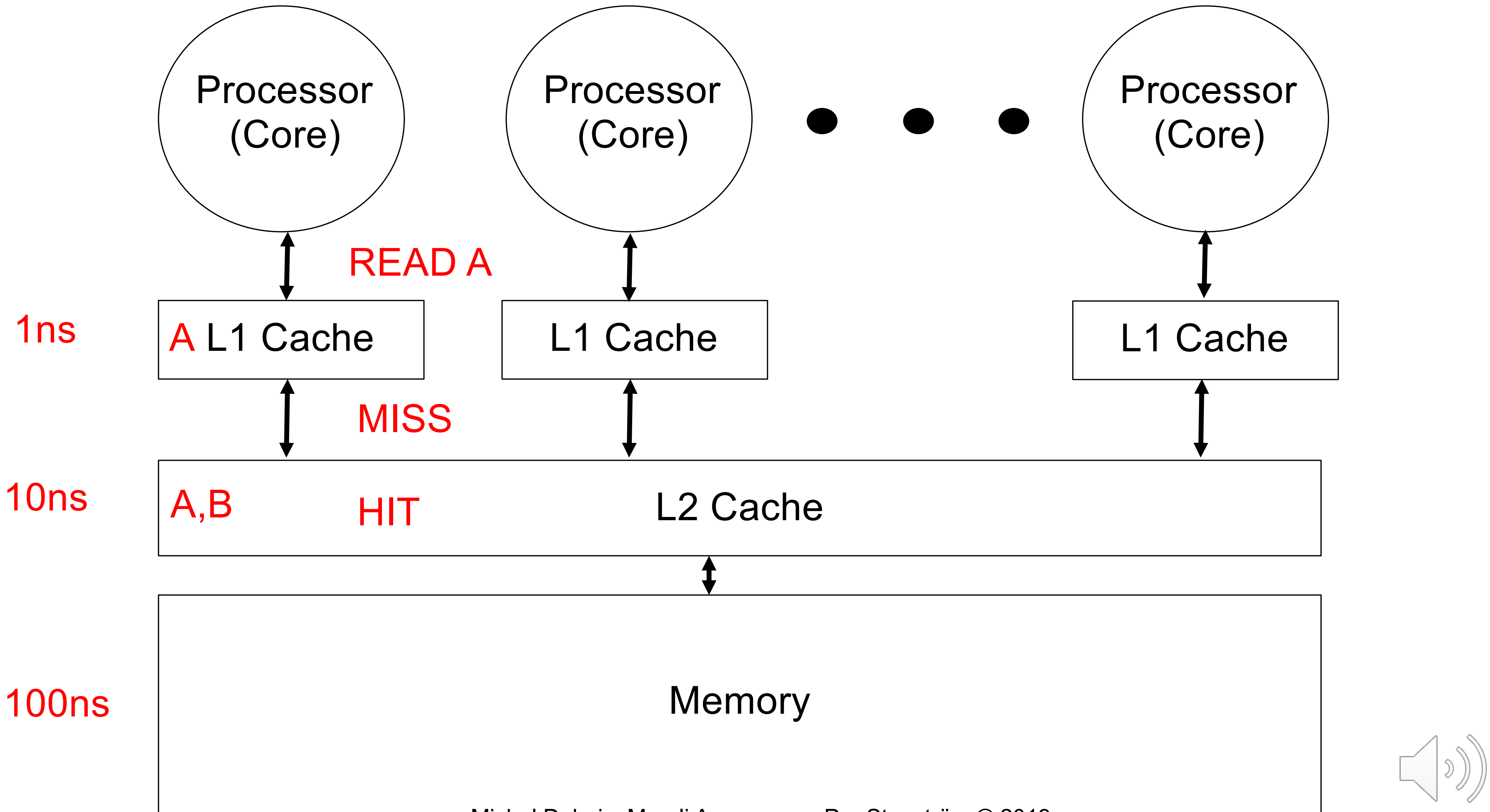
Type	Instruction Count (IC)	CPI	CPI x TPC [ns]
ALU	100 x 5 = 500	2.74	2.74
Load	100 x 2 = 200	3.74	3.74
Store	100 x 1 = 100	2.74	2.74
Control	100 x 1 = 100	4.74	4.74

T = 500 x 2.74 + 200 x 3.74 + 100 x 2.74 + 100 x 4.74 = 2.9 us


Processor (Core)
Processor (Core)
Processor (Core)
1ns
L1 Cache
L1 Cache
L1 Cache
10ns
L2 Cache
100ns
Memory


Processor (Core)
Processor (Core)
Processor (Core)
READ A
1ns
A L1 Cache
L1 Cache
L1 Cache
MISS
10ns
A
L2 Cache
MISS
100ns
Memory


Processor
(Core)
Processor
(Core)
Processor
(Core)
READ B
1ns
B L1 Cache
L1 Cache
L1 Cache
MISS
10ns
A,B
L2 Cache
MISS
100ns
Memory


Processor (Core)
Processor (Core)
Processor (Core)
READ A
1ns
A L1 Cache
L1 Cache
L1 Cache
MISS
10ns
A,B
HIT
L2 Cache
100ns
Memory

You should know by now

- The locality principle
- Cache mapping principles and cache organizations
- Replacement policies
- Write policies
- Cache hierarchy performance models