東邦大学

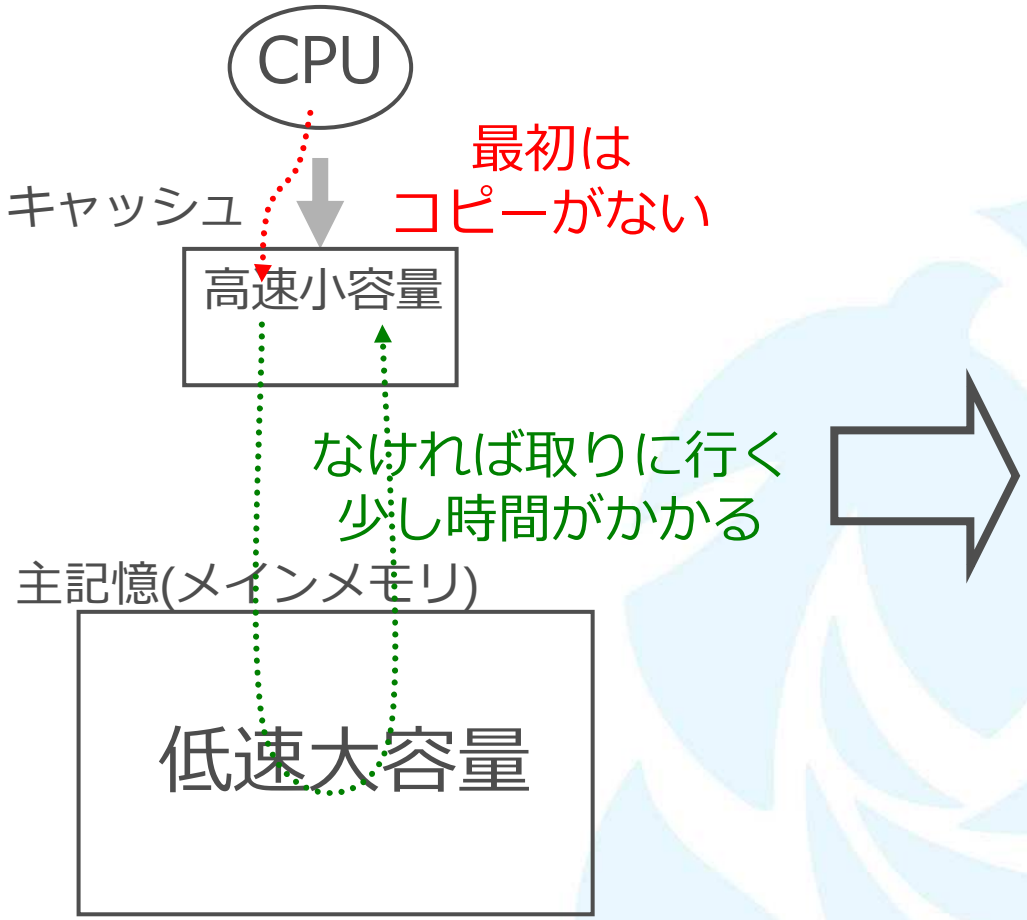
生命（いのち）の科学で未来をつなぐ

キャッシュの追い出し

まず復習：
キャッシュの考え方

東邦大学

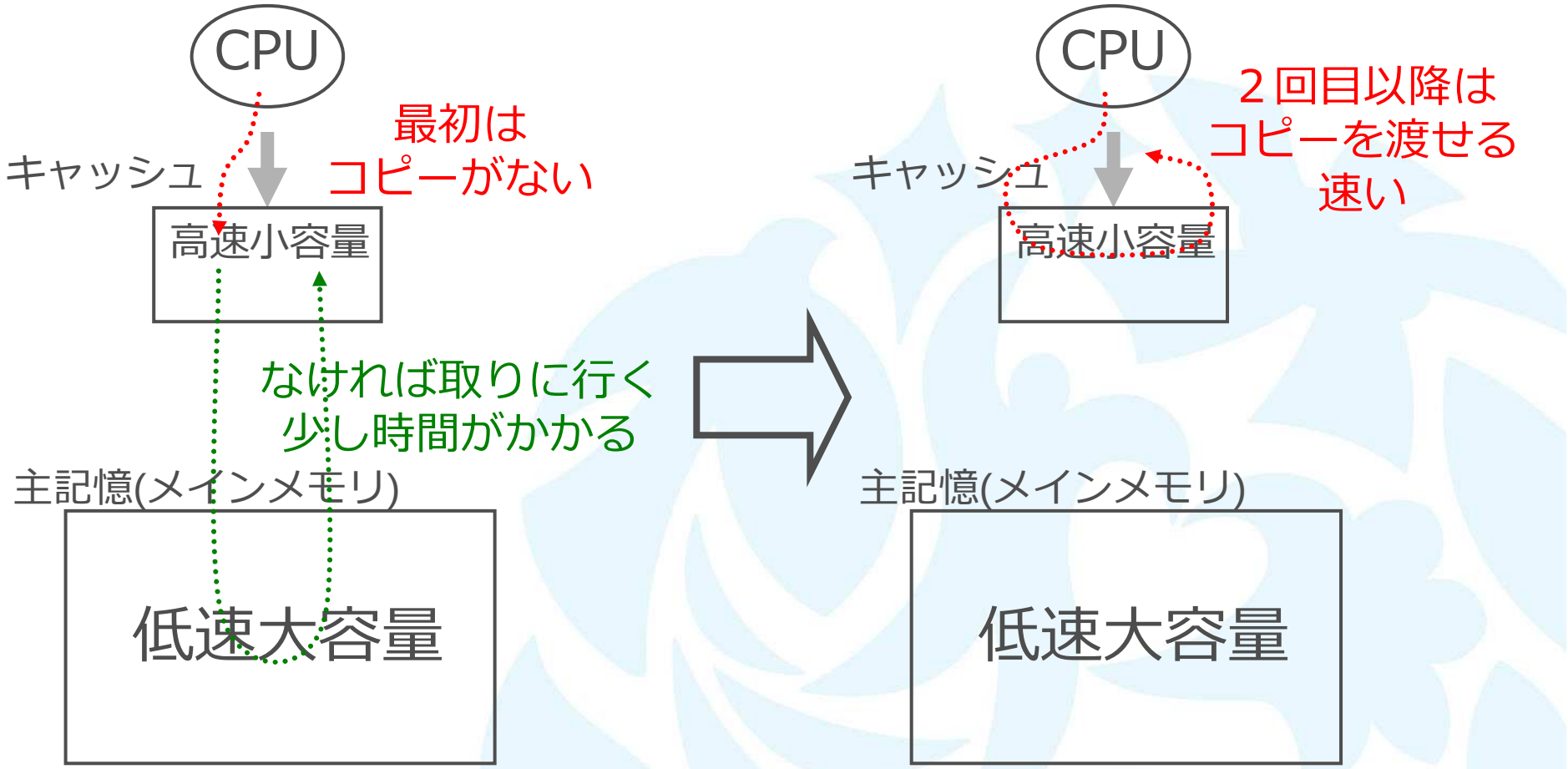
キャッシュの考え方



初めてのアクセス

東邦大学

キャッシュの考え方



さてそれで

２回目以降のアクセス

東邦大学

どんどんキャッシュにコピーを作ると

CPU が次々に (いろいろな場所に) アクセス

東邦大学

どんどんキャッシュにコピーを作ると

CPU が次々に (いろいろな場所に) アクセス

⇩

キャッシュには、いろいろな場所のコピーが
　たまってゆく

⇩

どんどんキャッシュにコピーを作ると

CPU が次々に (いろいろな場所に) アクセス

⇩

キャッシュには、いろいろな場所のコピーが
　たまってゆく

⇩

そのうちに、キャッシュが満杯になる

⇩

どんどんキャッシュにコピーを作ると

CPU が次々に (いろいろな場所に) アクセス

⇩

キャッシュには、いろいろな場所のコピーが
　たまってゆく

⇩

そのうちに、キャッシュが満杯になる

⇩

どうしよう？

新しい要求を満たすために、
場所を空ける必要がある

東邦大学

新しい要求を満たすために、
場所を空ける必要がある

追い出す

東邦大学

新しい要求を満たすために、
場所を空ける必要がある

追い出す

誰を？

東邦大学

誰を追い出すか？

１）誰でもいいから適当に

２）古いものから順番に

３）最近使っていないものを優先的に

４）将来使わないものを優先的に

など考えられる

東邦大学

誰を追い出すか？

名前が付いている

１）誰でもいいから適当に

２）古いものから順番に

３）最近使っていないものを優先的に

４）将来使わないものを優先的に

など考えられる

東邦大学

誰を追い出すか？

名前が付いている

1）誰でもいいから適当に

ランダム

2）古いものから順番に

FIFO、到着順

3）最近使っていないものを優先的に

LRU、最長未使用時間順

4）将来使わないものを優先的に

OPT、最適

など考えられる

東邦大学

誰を追い出すか？

１）誰でもいいから適当に
２）古いものから順番に
３）最近使っていないものを優先的に
４）将来使わないものを優先的に

ここでは、すべてを記憶する必要は無い
(そんな考え方があったな) と理解しよう

東邦大学

誰でもいいから適当に（ランダム）

キャッシュの中にあるブロックから
ランダムに選んで追い出す

⇒　仕組は簡単

誰でもいいから適当に（ランダム）

キャッシュの中にあるブロックから
ランダムに選んで追い出す

⇒　仕組は簡単

⇒　後で使うかもしれないブロックを
　　追い出してしまう可能性あり

古いものから順番に（FIFO、到着順）

FIFO ＝ First-In First-Out　先入れ先出し

キャッシュの中にあるブロック中で
一番古いものから追い出す

東邦大学

古いものから順番に（FIFO、到着順）

キャッシュの中にあるブロック中で
一番古いものから追い出す

⇒　取込んだ時刻を覚える必要あり

古いものから順番に（FIFO、到着順）

キャッシュの中にあるブロック中で
一番古いものから追い出す

⇒　取込んだ時刻を覚える必要あり

⇒　到着時刻が古いことと、後で使う
かどうかは、あまり関係ない

古いものから順番に（FIFO、到着順）

キャッシュの中にあるブロック中で
一番古いものから追い出す

⇒　取込んだ時刻を覚える必要あり

⇒　到着時刻が古いことと、後で使う
かどうかは、あまり関係ない

⇒　後で使うかもしれないブロックを
追い出してしまう可能性あり

東邦大学

最近使っていないものを優先的に（LRU）

Least Recently Used　最長未使用時間順

キャッシュの中にあるブロック中で
最後に使ったのが最も古いものを追出す

東邦大学

最近使っていないものを優先的に（LRU）

キャッシュの中にあるブロック中で
最後に使ったのが最も古いものを追出す

⇒　使った時刻を覚える必要あり

最近使っていないものを優先的に（LRU）

キャッシュの中にあるブロック中で
最後に使ったのが最も古いものを追出す

⇒　使った時刻を覚える必要あり

⇒　最近使われていないと、一般には
(近い) 将来も使われない ～ 経験則

最近使っていないものを優先的に（LRU）

キャッシュの中にあるブロック中で
最後に使ったのが最も古いものを追出す

⇒　使った時刻を覚える必要あり

⇒　最近使われていないと、一般には
(近い) 将来も使われない ～ 経験則

⇒　後で使うかもしれないブロックを
なるべく追い出さないでおく

東邦大学

将来使わないものを優先的に（OPT）

OPT ＝ Optimal　　最適

将来を含めて見渡して、
欲しいブロックが
最もよくキャッシュに残っている方法

将来使わないものを優先的に（OPT）

将来を含めて、欲しいブロックが
最もよくキャッシュに残っている方法

⇒　将来のことは分らない(予測不可)

⇒　現実には、この方式は作れない

⇒　こう追い出したら理想的という方式

実感が湧かないので例を

実感が湧かないので例を

下記の順番でブロックをアクセスする

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5

東邦大学

実感が湧かないので例を

下記の順でメモリ (ブロック) をアクセスする

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5

キャッシュは４ブロック置けるとする

ブロックの置き場所は自由に選べる方式（フルアソシアティブ方式）とする

東邦大学

実感が湧かないので例を

下記の順でメモリ (ブロック) をアクセスする

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5

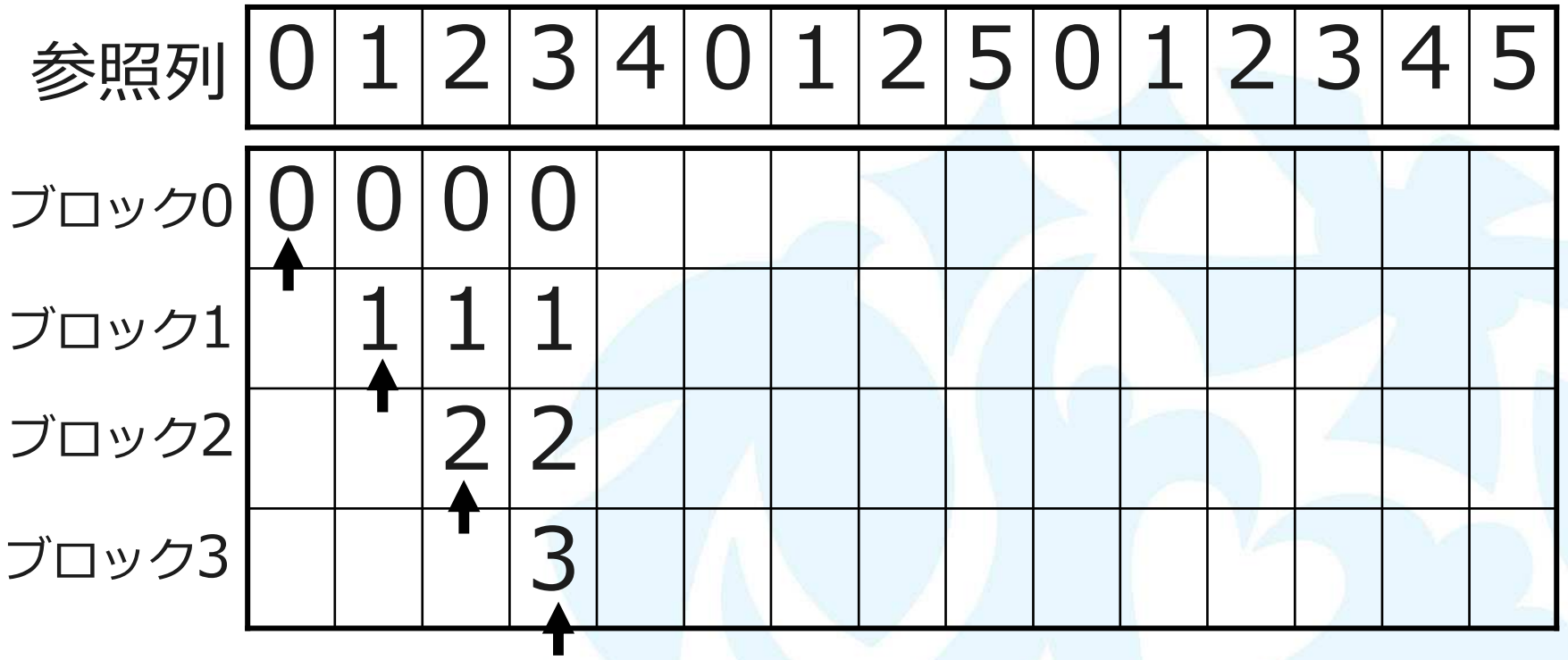
キャッシュは４ブロック置けるとする

ブロックの置き場所は自由に選べる方式（フルアソシアティブ方式）とする

５つ目で溢れるが誰を追い出すか

東邦大学

FIFO（古いブロックから追出し）



0, 1, 2, 3 と
空いているブロックへ入れた

FIFO（古いブロックから追出し）

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	4										
ブロック1		1	1	1	1										
ブロック2			2	2	2										
ブロック3				3	3										

4を入れたいが、すでに満杯で入らない
最も古いのは0なので追出して、4を入れた

FIFO（古いブロックから追出し）

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	4	4									
ブロック1		1	1	1	1	0									
ブロック2			2	2	2	2									
ブロック3				3	3	3									

0を入れたいが、すでに満杯で入らない
最も古いのは1なので追出して、0を入れた

FIFO（古いブロックから追出し）

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	4	4	4								
ブロック1		1	1	1	1	0	0								
ブロック2			2	2	2	2	1								
ブロック3				3	3	3	3								

1を入れたいが、すでに満杯で入らない
最も古いのは2なので追出して、1を入れた

FIFO（古いブロックから追出し）

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	4	4	4	4							
ブロック1		1	1	1	1	0	0	0							
ブロック2			2	2	2	2	1	1							
ブロック3				3	3	3	3	2							

2を入れたいが、すでに満杯で入らない
最も古いのは3なので追出して、2を入れた

FIFO（古いブロックから追出し）

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	4	4	4	4	5						
ブロック1		1	1	1	1	0	0	0	0						
ブロック2			2	2	2	2	1	1	1						
ブロック3				3	3	3	3	2	2						

5を入れたいが、すでに満杯で入らない
最も古いのは4なので追出して、5を入れた

FIFO（古いブロックから追出し）

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	4	4	4	4	5	5	5	5			
ブロック1		1	1	1	1	0	0	0	0	0	0	0			
ブロック2			2	2	2	2	1	1	1	1	1	1			
ブロック3				3	3	3	3	2	2	2	2	2			

0, 1, 2を参照するが、既にキャッシュ内にコピーがあるので、それをそのまま使う

FIFO（古いブロックから追出し）

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	4	4	4	4	5	5	5	5	5		
ブロック1		1	1	1	1	0	0	0	0	0	0	0	3		
ブロック2			2	2	2	2	1	1	1	1	1	1	1		
ブロック3				3	3	3	3	2	2	2	2	2	2		

3を入れたいが、すでに満杯で入らない
最も古いのは0なので追出して、3を入れた

FIFO（古いブロックから追出し）

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	4	4	4	4	5	5	5	5	5	5	
ブロック1		1	1	1	1	0	0	0	0	0	0	0	3	3	
ブロック2			2	2	2	2	1	1	1	1	1	1	1	4	
ブロック3				3	3	3	3	2	2	2	2	2	2	2	

4を入れたいが、すでに満杯で入らない
最も古いのは1なので追出して、4を入れた

FIFO（古いブロックから追出し）

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	4	4	4	4	5	5	5	5	5	5	5
ブロック1		1	1	1	1	0	0	0	0	0	0	0	3	3	3
ブロック2			2	2	2	2	1	1	1	1	1	1	1	4	4
ブロック3				3	3	3	3	2	2	2	2	2	2	2	2

5を参照するが、既にキャッシュ内に
コピーがあるので、それをそのまま使う

FIFO（古いブロックから追出し）

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	4	4	4	4	5	5	5	5	5	5	5
ブロック1		1	1	1	1	0	0	0	0	0	0	0	3	3	3
ブロック2			2	2	2	2	1	1	1	1	1	1	1	4	4
ブロック3				3	3	3	3	2	2	2	2	2	2	2	2

この流れで、入れ替えを起こしたのは
○が付いているところ　＝　7回

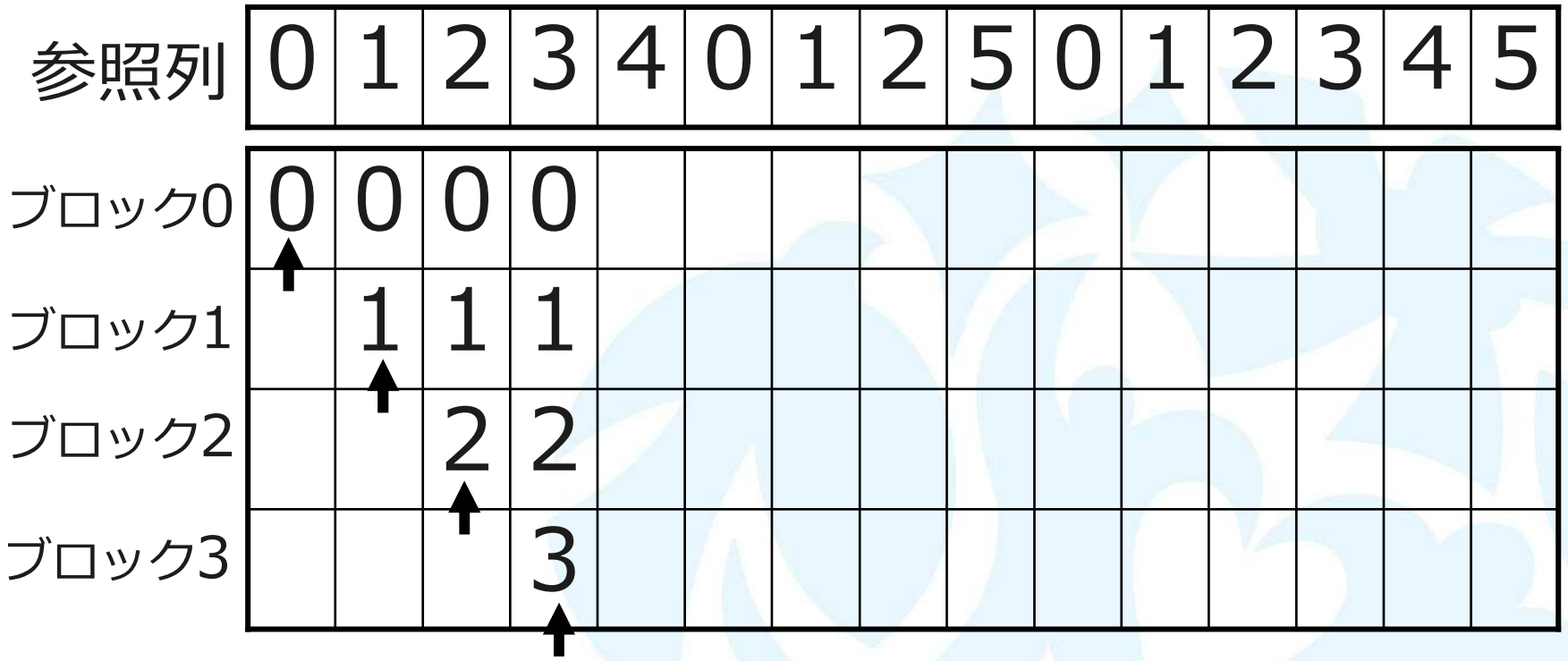
FIFO（古いブロックから追出し）

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	4	4	4	4	5	5	5	5	5	5	5
ブロック1		1	1	1	1	0	0	0	0	0	0	0	3	3	3
ブロック2			2	2	2	2	1	1	1	1	1	1	1	4	4
ブロック3				3	3	3	3	2	2	2	2	2	2	2	2

○が、入れ替えを起こした所　＝　7回

この0を残しておけば、ここで入替えず使えたはず

後で使うことも考えて追出す(OPT)と？



ここまでは FIFO と同じで、0, 1, 2, 3 と
空いているブロックへ入れた

東邦大学

後で使うことも考えて追出す(OPT)と？

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	0										
ブロック1		1	1	1	1										
ブロック2			2	2	2										
ブロック3				3	4										

すぐ次で 0, 1, 2 と使うことが分かっていれば
（使わない）3 を追い出すのが得策

後で使うことも考えて追出す(OPT)と？

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	0	0	0	0							
ブロック1		1	1	1	1	1	1	1							
ブロック2			2	2	2	2	2	2							
ブロック3				3	4	4	4	4							

0, 1, 2を参照するが、既にキャッシュ内に
コピーがあるので、それをそのまま使う

後で使うことも考えて追出す(OPT)と？

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	0	0	0	0	0						
ブロック1		1	1	1	1	1	1	1	1						
ブロック2			2	2	2	2	2	2	2						
ブロック3				3	4	4	4	4	5						

すぐ次で 0, 1, 2 と使うことが分かっていれば
（使わない）4 を追い出すのが得策

後で使うことも考えて追出す(OPT)と？

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	0	0	0	0	0	0	0	0			
ブロック1		1	1	1	1	1	1	1	1	1	1	1			
ブロック2			2	2	2	2	2	2	2	2	2	2			
ブロック3				3	4	4	4	4	5	5	5	5			

0, 1, 2を参照するが、既にキャッシュ内に
コピーがあるので、それをそのまま使う

後で使うことも考えて追出す(OPT)と？

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	3
ブロック1		1	1	1	1	1	1	1	1	1	1	1	1	4	4
ブロック2			2	2	2	2	2	2	2	2	2	2	2	2	2
ブロック3				3	4	4	4	4	5	5	5	5	5	5	5

すぐ次で 5 を使うことが分かっていれば
(使わない) 0, 1, 2 のどれかを追い出すのが得策

後で使うことも考えて追出す(OPT)と？

参照列	0	1	2	3	4	0	1	2	5	0	1	2	3	4	5
ブロック0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	3
ブロック1		1	1	1	1	1	1	1	1	1	1	1	1	4	4
ブロック2			2	2	2	2	2	2	2	2	2	2	2	2	2
ブロック3				3	4	4	4	4	5	5	5	5	5	5	5

○が、入れ替えを起こした所　＝　4回

このようにどれを追出すかによって
　入替えの回数が減って、速くなる

キャッシュの追出し方のまとめ

追出すブロックの選び方にはいろいろある

キャッシュの追出し方のまとめ

追出すブロックの選び方にはいろいろある

将来のことが分かれば、最適な選び方ができるが、分からないのでなるべくよい方法を考えるしかない

キャッシュの追出し方のまとめ

追出すブロックの選び方にはいろいろある

将来のことが分かれば、最適な選び方ができるが、分からないのでなるべくよい方法を考えるしかない

一般には、LRU (最近最も使われていないブロックから追出す）が良い

理由は、一般に「同じようなブロックが繰り返して使われる」傾向があるから

東邦大学

キャッシュの追出し方がいろいろ
あることを理解できましたか？

東邦大学