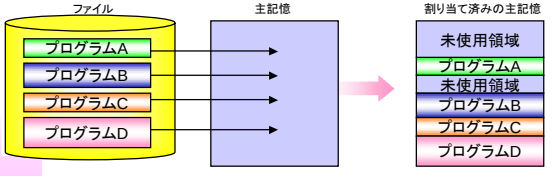
記憶管理

主記憶（メモリ）へのプログラムの割り当て

- プログラムは主記憶上で実行する必要がある
- プログラムやデータはファイルとして補助記憶装置に格納されている
- 複数プログラムを同時に主記憶上で実行する
- 主記憶より大きな領域を必要とするプログラムを実行する

プログラムを実行するには、プログラムを補助記憶装置から主記憶に読み込んだ（ロード）後に実行する。



主記憶の連続割り当て

主記憶をどのように複数のプログラムで共用するか

単一割り当て

すべてのプログラムを一つの領域に割り当てる

分割割り当て

主記憶を区画という単位で分割し、プログラムごとに区画を割り当て

固定区画方式：

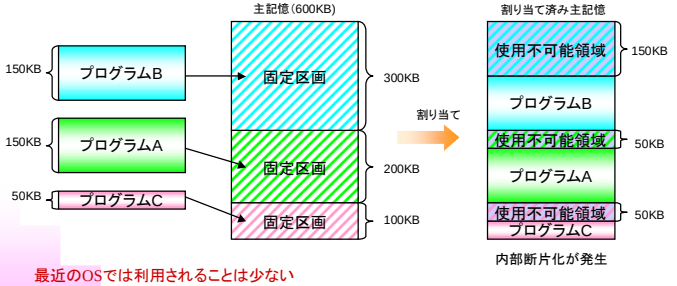
主記憶を固定した区画に分割し、そこに収まるプログラムをロード

可変区画方式：

各プログラムが必要とする領域サイズに主記憶を分割してロード

固定区画方式

- 各区画の大きさは予め決められ、変更はできない
- 各区画には高々一つのプログラムが使用可能
- 各プログラムはそれの必要領域よりも大きな区画に割り当てられる



最近のOSでは利用されることは少ない

断片化

外部断片化

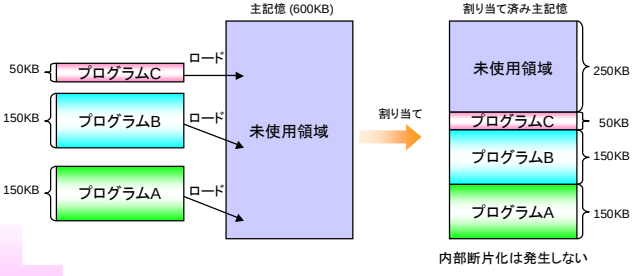
区画が未使用にも関わらず、その区画に合致するプログラムが存在せずに未使用になる

内部断片化

区画自体はプログラムに割り当てられているにも関わらず、区画内に未使用領域が存在する

可変区画方式

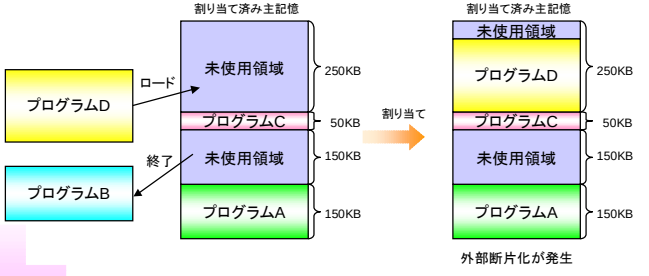
- 各区画の大きさは動的に変更可能
- 各区画には高々一つのプログラムが使用可能
- 各プログラムに必要な領域サイズの区画を生成し、割り当てる



内部断片化は発生しない

可変区画方式

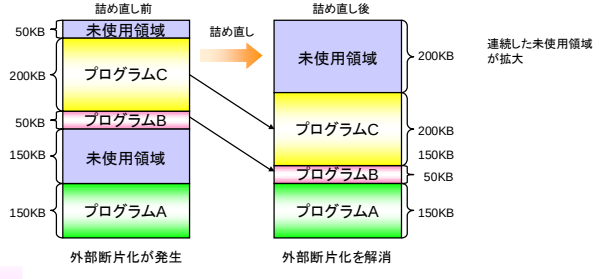
内部断片化は発生しないが、プログラムの終了やロードを繰り返すと利用可能な未使用領域が断片化し、大きなプログラムのロードが困難になる。



外部断片化が発生

メモリ詰め直し

外部断片化を解消するために、プログラムを主記憶の連続した領域に詰めて、一つの大きな空き領域を作る（compaction）



- 詰め直し処理自体が多くのCPU時間を消費
- 詰め直し中は他の処理は実行は困難

空き領域管理

可変区画方式における空き領域管理

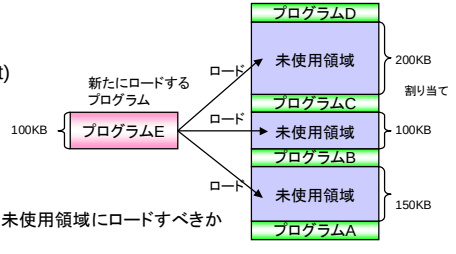
- 未使用領域が主記憶内に分散化する
- 新たなプログラムのロードに必要な未使用領域に割り当てる

空き領域（未使用領域）の割り当て方：

(A) 初適合(first-fit)

(B) 最適適合(best-fit)

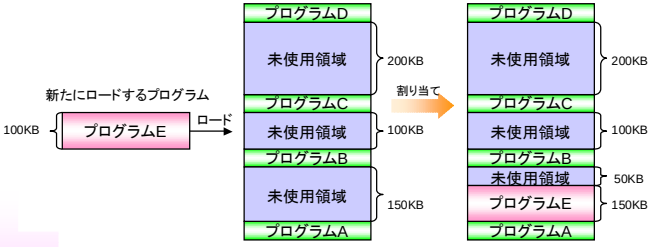
(C) 最悪適合（worst-fit)



どの未使用領域にロードすべきか

空き領域管理(初適合)

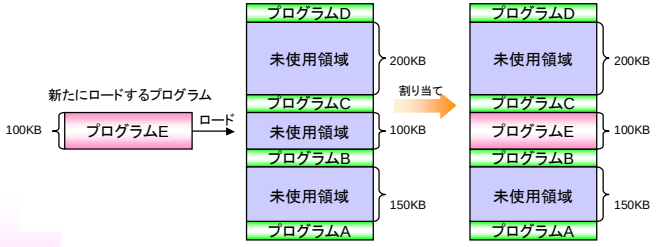
最初に見つかった十分な大きさの未使用(空き)領域を割り当てる
○割り当て領域を見つける(探索)が速い、実装が簡単
×小さなプログラムが大きな未使用領域に割り当てられる可能性がある



Ichiro Satoh

空き領域管理(最適適合)

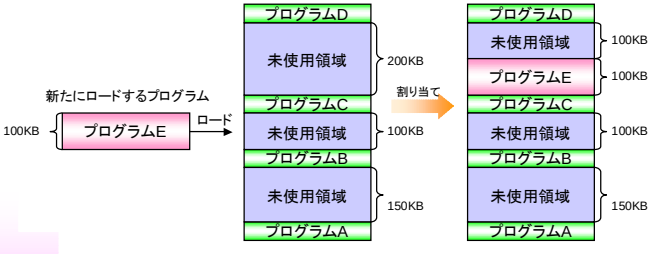
十分な大きさをもつ最小の未使用(空き)領域を割り当てる
○割り当て領域を見つける(探索)が遅い、実装が難しい
×小さな未使用領域が発生する可能性がある



Ichiro Satoh

空き領域管理(最悪適合)

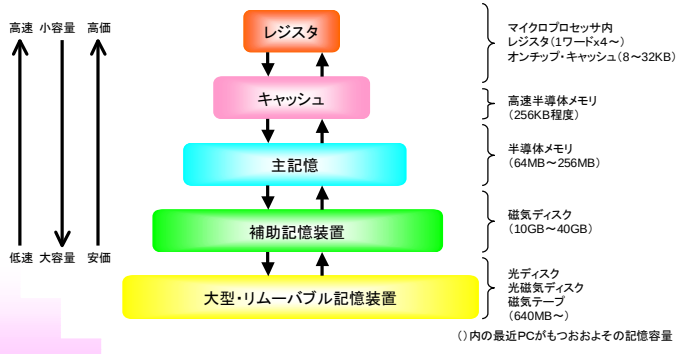
最大の未使用領域(空き)領域を割り当てる
○大きな未使用領域を作り出す
×領域の利用効率が悪化



Ichiro Satoh

記憶装置の階層

記憶装置の容量と高速性はトレードオフの関係



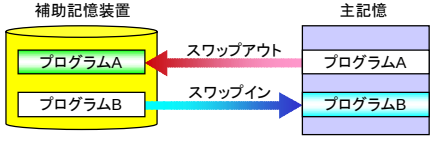
( )内の最近PCがもつおおよその記憶容量

Ichiro Satoh

スワップ

主記憶には限りがあり、実行したいすべてのプログラムを主記憶にロードできるとは限らない。

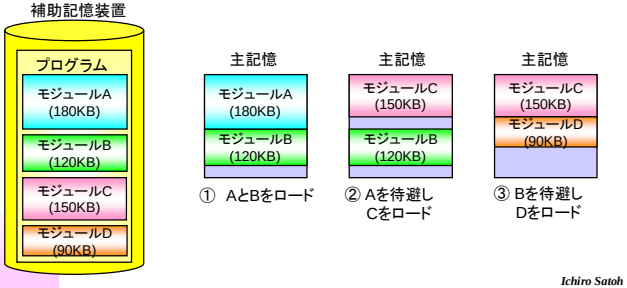
- プログラムが待ち状態にあるときは補助記憶装置にそのプログラムを待避する（スワップアウト）
- 補助記憶装置に待避されている実行可能なプログラムを主記憶に読み込む（スワップイン）



見かけの主記憶サイズが多くなる反面、スワップ処理に大きなコストを必要とする

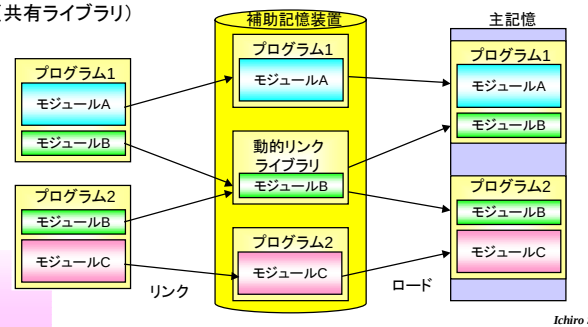
オーバレイ

プログラムの実行ではプログラム全体を主記憶にロードする必要はない
これから実行しようとする部分だけをロードすれば十分
例：　モジュールA～Dからなるプログラム
　　モジュールの実行順：A→B→C→D



動的リンク

プログラムを構成するモジュールをリンクして一つのプログラムとせず、
　実行時にリンク・ロードする
動的リンク用のモジュールは他のプログラムからも利用可能
（共有ライブラリ）



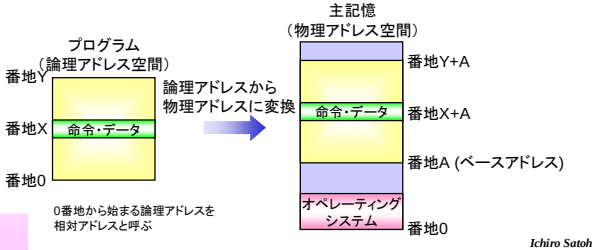
アドレス

物理アドレス

- プログラム中の命令やデータの主記憶上の場所を識別するアドレス

論理アドレス

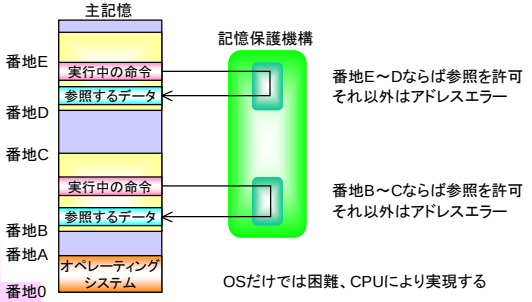
- プログラム中の命令やデータの配置場所を識別するアドレス
- プログラム実行時には物理アドレスに変換される（アドレスマッピング）



記憶保護

OSと複数のプログラムが同時に主記憶におかれるとき

- OS及びプログラムの主記憶領域を他のプログラムから読み書きできないようにする（記憶保護）

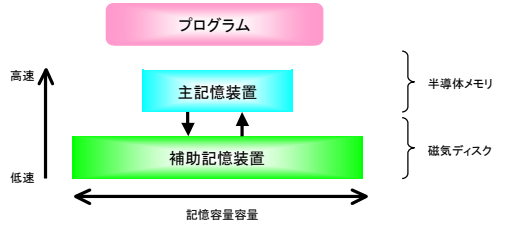


OSだけでは困難、CPUにより実現する

記憶空間

主記憶装置は小容量かつ高価

- プログラム実行に必要な記憶空間サイズよりも主記憶装置容量が小さいことがある



既存OSの仮想記憶

パーソナルコンピュータ及びワークステーション用OSの多くは仮想記憶機構を提供

例：標準的なパーソナルコンピュータ

主記憶：64MB

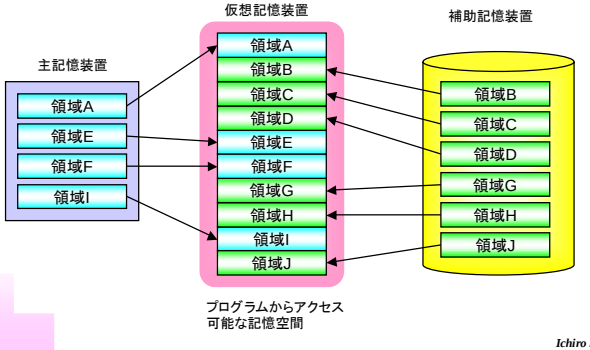
メモリ消費量：

Windows98	32～40MB
Internet Explorer	3～10MB
MS Word2000	8～12MB
MS Excel2000	6～10MB

注意：実際のメモリ消費量は使用条件に依存する

仮想記憶

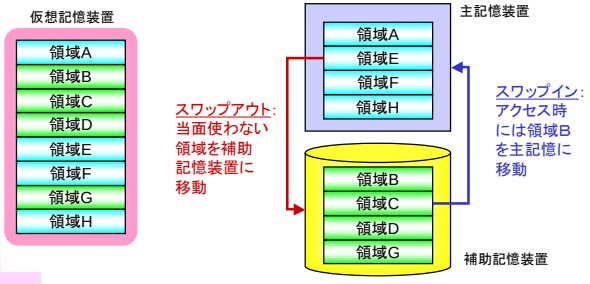
主記憶装置（メモリ）と補助記憶装置（ハードディスク）の両方を用いて、プログラムに主記憶の大きさにとらわれない仮想的な記憶装置



仮想記憶の原理

プログラムが利用する命令やデータは主記憶中になければならない

- 利用頻度の高い領域を主記憶装置に配置
- 利用頻度の低い領域を補助記憶装置に配置

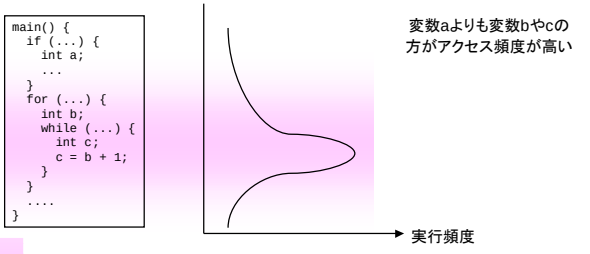


局所性原理

経験則

プログラムの10%の部分がアクセス回数の90%を占める。

- 頻繁に使うデータは決まっている
- プログラム実行の大部分はループ内の実行



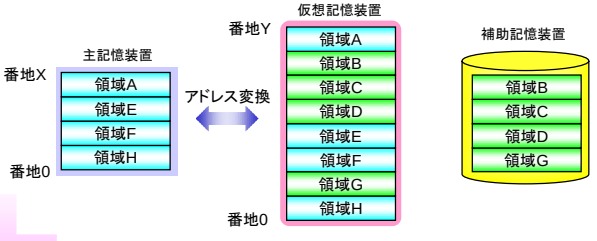
仮想アドレス

物理アドレス

- 主記憶中のプログラム中の命令やデータの場所を識別

仮想アドレス

- 仮想記憶中のプログラム中の命令やデータの場所を識別



仮想アドレス空間

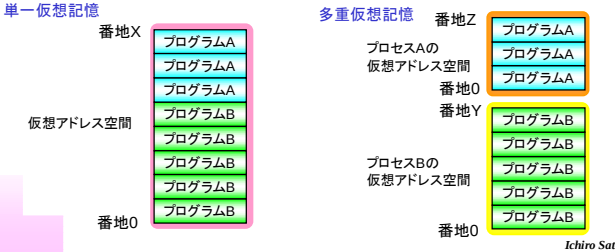
仮想記憶装置のアドレス方法

単一仮想記憶:

一つの仮想アドレス空間上にすべてのプログラムを配置

多重仮想記憶:

プロセスごとに仮想アドレス空間を容易



仮想記憶方式

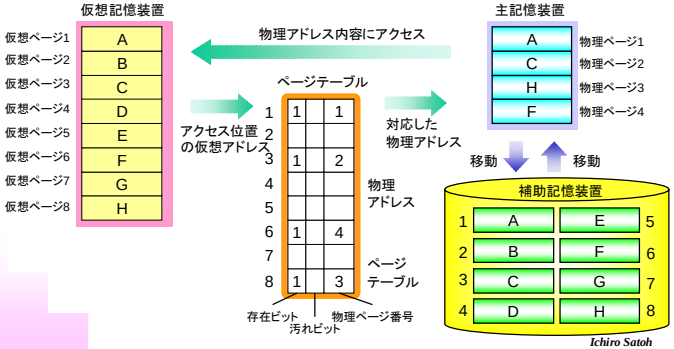
仮想記憶はプログラムをあるブロックサイズで分割して、そのブロックで主記憶と補助記憶にプログラムまたはデータを割り当てる

- セグメンテーション:
  プログラムの構造に従ってブロック(セグメントと呼ぶ)のサイズ設定して記憶領域を割り当てる。ブロックサイズは一定ではない
- ページング:
  OSにより定められたサイズのブロック(ページと呼ぶ)で記憶領域を割り当てる

Ichiro Satoh

ページング

ページテーブルにより仮想アドレスに対応したページを発見・アクセス



Ichiro Satoh

ページング

主記憶上にある仮想アドレスへのアクセスが起きたとき
①仮想アドレスに対応した主記憶上のページを発見
②主記憶上のページにアクセス

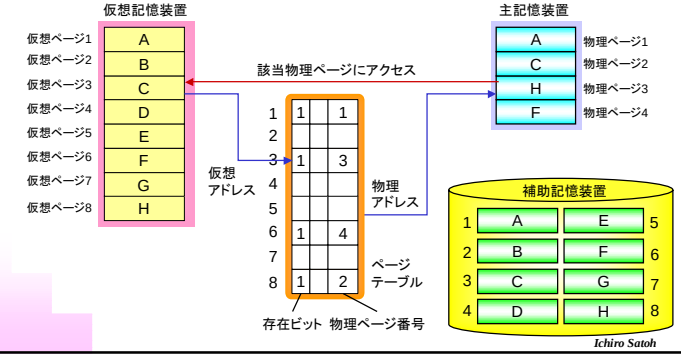
主記憶上にない仮想アドレスへのアクセスが起きたとき
①仮想アドレスに対応した補助記憶上のページを発見
②発見したページ内容を主記憶の空きスペースにコピー
③主記憶上のページにアクセス

空きスペースが内場合
①当面使いそうになり物理ページを選択
②選択したページ内容を補助記憶装置に移動させて空きスペースを作る
③仮想アドレスに対応した補助記憶上のページを発見
④発見したページ内容を主記憶の空きスペースにコピー
⑤主記憶上のページにアクセス

Ichiro Satoh

ページング(ヒット時)

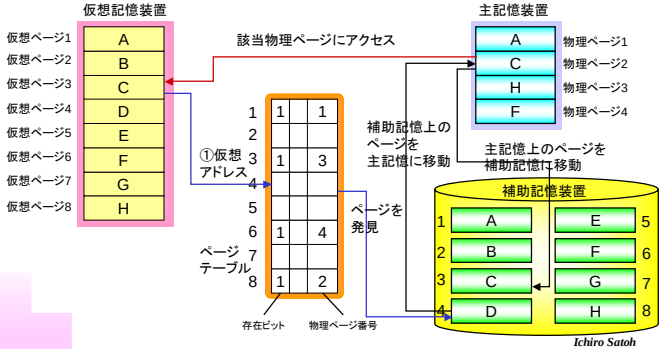
アクセスしたい仮想アドレスに対応したページが主記憶装置にあるとき



Ichiro Satoh

ページング（ミス時）

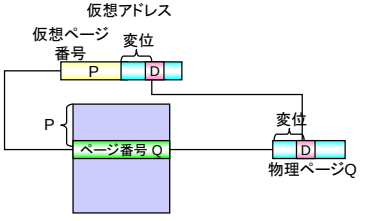
アクセスしたい仮想アドレスに対応したページが主記憶装置にないとき



アドレス変換

仮想アドレスから物理ページ番号と物理内ページ位置を計算
（ページサイズをN、仮想アドレスをLとする）

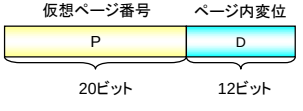
- 仮想ページ番号P
  P = L div N
- 物理ページ内ページ位置（変位）D
  D = L mod N



アドレス変換例

例：ページサイズ4KB、仮想アドレス空間が2^{32}とする

仮想アドレスは32bit長、上位ビット20が仮想ページ番号、下位12ビットがページ内変位

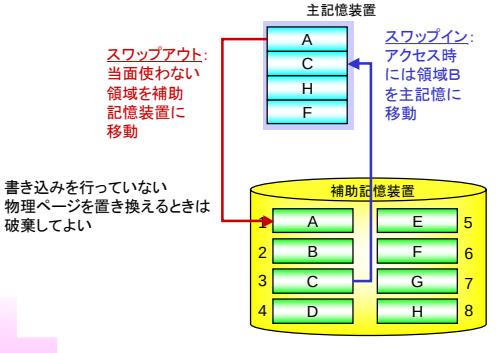


ページテーブル内のPエントリが示す物理ページ番号が4のとき

物理アドレス = 4KB × 4 + D

ページ置き換え

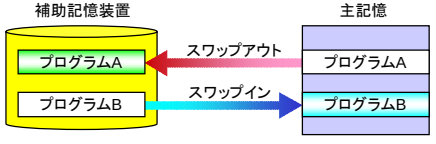
当面アクセスされない主記憶上の物理ページを補助記憶に移動



スワップ

主記憶には限りがあり、実行したいすべてのプログラムを主記憶にロードできるとは限らない。

- プログラムが待ち状態にあるときは補助記憶装置にそのプログラムを待避する（スワップアウト）
- 補助記憶装置に待避されている実行可能なプログラムを主記憶に読み込む（スワップイン）



見かけの主記憶サイズが多くなる反面、スワップ処理に大きなコストを必要とする

ページ置き換えアルゴリズム

ページ置き換えは実行コストが大きい
　理由：補助記憶装置への読み（書き）が必要

当面アクセスされないページを発見するのは困難

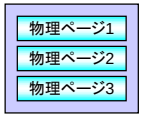
擬似的な方法で当該ページを発見
(A) FIFO (First In First Out)
　主記憶にロードされたのが一番古いものを置き換える

(B) LRU (Least Recently Used)
　最も長い時間アクセスされていないページを置き換える

FIFOによるページ置き換え

3つの物理ページからなる主記憶へのアクセス

主記憶装置



主記憶へのアクセス順序（仮想記憶上のページ番号）

0, 1, 2, 3, 0, 1, 4, 0, 1, 2, 3, 4

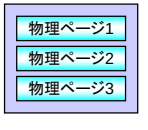
主記憶にロードされたのが一番古いものを置き換える

参照ページ	0	1	2	3	0	1	4	0	1	2	3	4
物理ページ1	0	0	0	1	2	3	0	0	0	1	4	4
物理ページ2		1	1	2	3	0	1	1	1	4	2	2
物理ページ3			2	3	0	1	4	4	4	2	3	3

LRUによるページ置き換え

3つの物理ページからなる主記憶へのアクセス

主記憶装置



主記憶へのアクセス順序（仮想記憶上のページ番号）

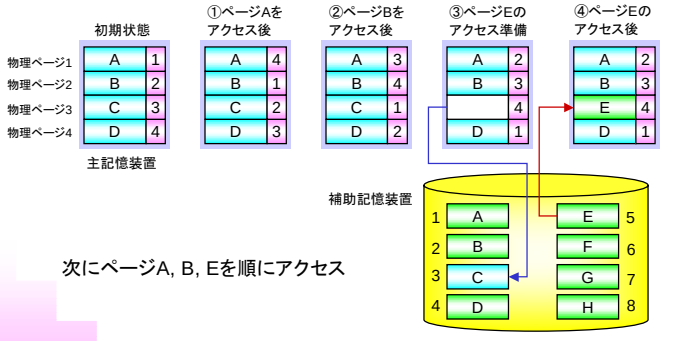
0, 1, 2, 3, 0, 1, 4, 0, 1, 2, 3, 4

最も長い時間アクセスされていないページを置き換える

参照ページ	0	1	2	3	0	1	4	0	1	2	3	4
物理ページ1	0	1	2	3	0	1	4	0	1	2	3	4
物理ページ2		0	1	2	3	0	1	4	0	1	2	3
物理ページ3			0	1	2	3	0	1	4	0	1	2

ページアクセス（LRU）

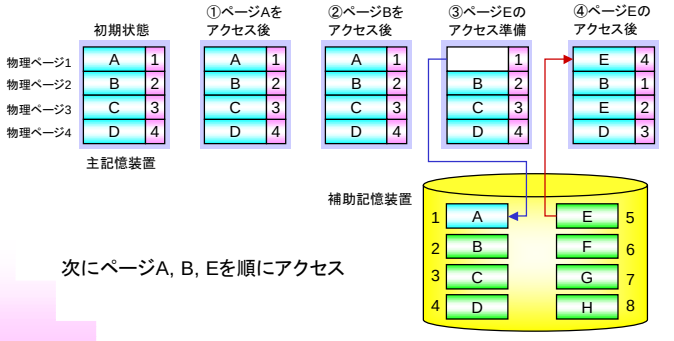
物理ページ(A～D)がA, B, C, Dが順にアクセスされたとするとき、



次にページA, B, Eを順にアクセス

ページアクセス（FIFO）

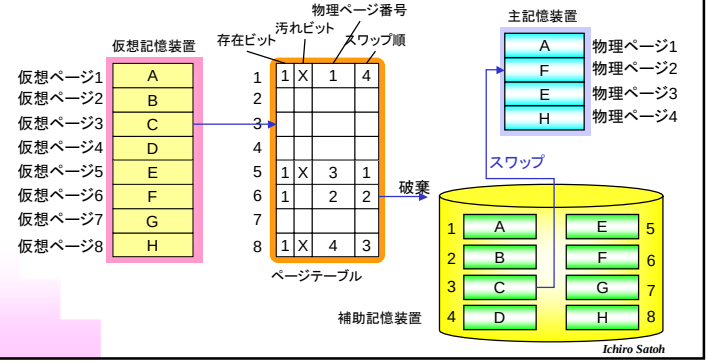
物理ページ(A～D)がA, B, C, Dが順にアクセスされたとするとき、

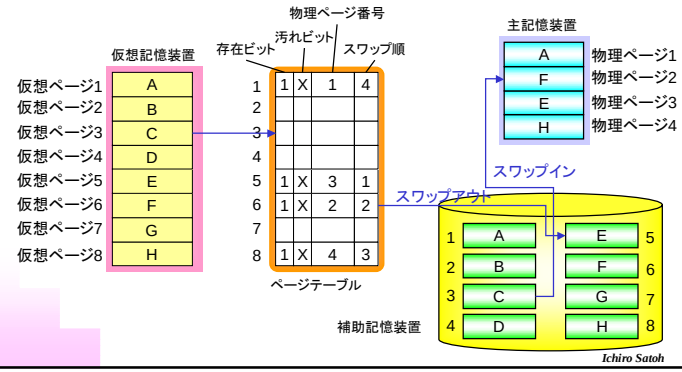


次にページA, B, Eを順にアクセス

スワップイン（空き有り）

内容に変更がない物理ページのスワップアウトせずに破棄し、
補助記憶上のアクセス対象ページをスワップイン



スワップイン（空きなし）

内容に変更がある物理ページのスワップアウト