東邦大学

生命の科学で未来をつなぐ

デマンドページングの性能４
局所性とミス率の例

東邦大学

参照の局所性はかなりよく成り立つ

- メモリの参照は、命令か、データ (オペランド) か
- 命令はたいていはアドレス順に実行される
  - 後述する理由で、局所性が成り立つことが多い
- データは近傍がアクセスされるとは限らないが
  - 変数は、メモリ内ではたいてい近傍に配置される
  - 変数同士が近傍なら、参照は少数ページ内に留まるはず
  - そうでない配置もあり得る（後述）
    ⇒　局所性は成り立たず、アクセスが遅くなる

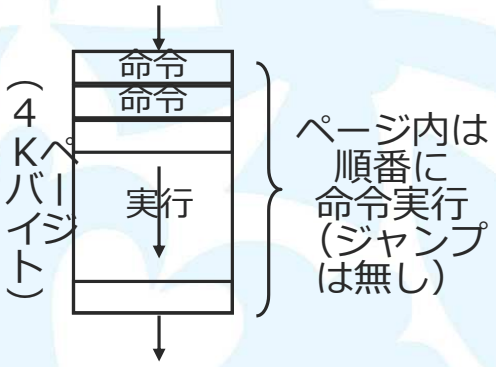
東邦大学

命令の読出しアクセス①

- 命令の読出しは、メモリ上で順番に参照

命令の読出しアクセス①

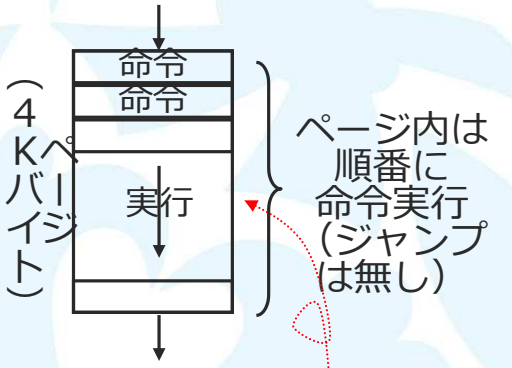
- 命令の読出しは、メモリ上で順番に参照
  - ページ先頭の命令を読む
  - 次の命令も同じページ
  - 順番に続けて同じページ



東邦大学

命令の読出しアクセス①

- 命令の読出しは、メモリ上で順番に参照
  - ページ先頭の命令を読む
  - 次の命令も同じページ
  - 順番に続けて同じページ

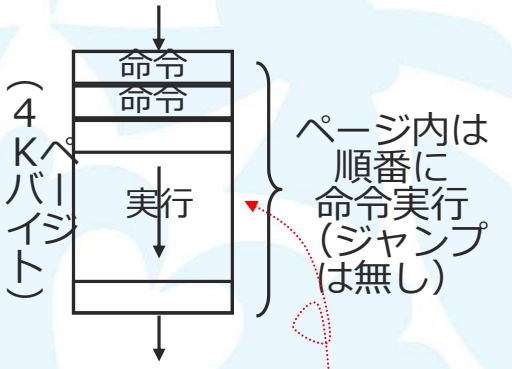


このページ内の命令を実行している間は新たなページフォルトは発生しない

命令の読出しアクセス①

- 命令の読出しは、メモリ上で順番に参照
  - ページ先頭の命令を読む
  - 次の命令も同じページ
  - 順番に続けて同じページ

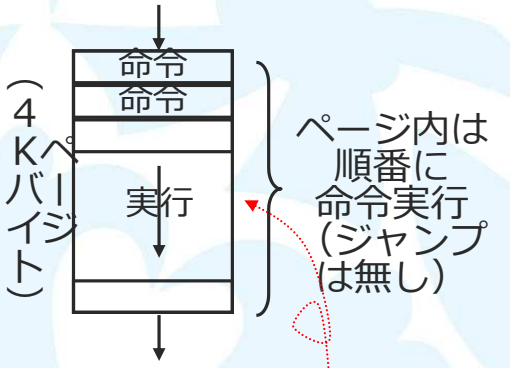
- 例で計算
  - １ページ＝4KB、１命令＝4B
  - 順番どおりアクセス（ジャンプ無）
  - 1024命令進むと次ページ



このページ内の命令を実行している間は新たなページフォルトは発生しない

命令の読出しアクセス①

- 命令の読出しは、メモリ上で順番に参照
  - ページ先頭の命令を読む
  - 次の命令も同じページ
  - 順番に続けて同じページ
- 例で計算
  - １ページ＝4KB、１命令＝4B
  - 順番どおりアクセス（ジャンプ無）
  - 1024命令進むと次ページ
    ⇒ 1024回に１回フォールト

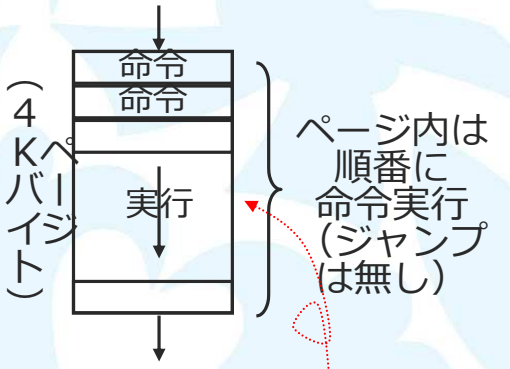


このページ内の命令を実行している間は新たなページフォルトは発生しない

命令の読出しアクセス①

- 命令の読出しは、メモリ上で順番に参照
  - ページ先頭の命令を読む
  - 次の命令も同じページ
  - 順番に続けて同じページ
- 例で計算
  - １ページ＝4KB、１命令＝4B
  - 順番どおりアクセス（ジャンプ無）
  - 1024命令進むと次ページ
    ⇒ 1024回に１回フォールト

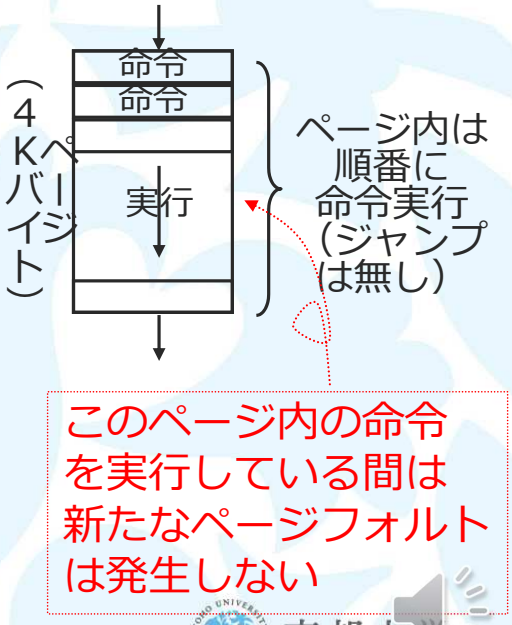
    ⇒ ミス率＝10^{-3}



このページ内の命令を実行している間は新たなページフォルトは発生しない

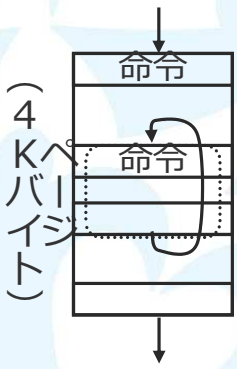
命令の読出しアクセス①

- 命令の読出しは、メモリ上で順番に参照
  - ページ先頭の命令を読む
  - 次の命令も同じページ
  - 順番に続けて同じページ
- 例で計算
  - １ページ＝4KB、１命令＝4B
  - 順番どおりアクセス（ジャンプ無）
  - 1024命令進むと次ページ
    - ⇒ 1024回に１回フォールト
    - ⇒ ミス率＝10^{-3}
- 更に



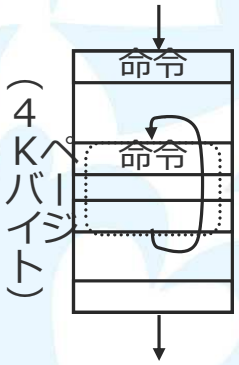
命令の読出しアクセス②

- ループがあると



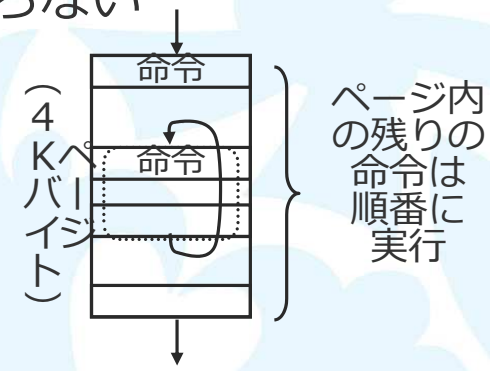
命令の読出しアクセス②

- ループがあると命令の読出しのミス率は低下



命令の読出しアクセス②

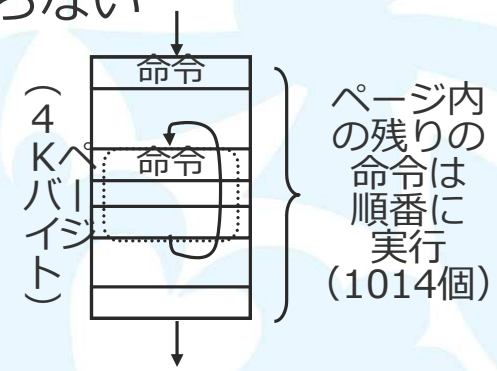
- ループがあると命令の読出しのミス率は低下
  - ループが回っている間はそのページに留まるので、フォールトは起こらない



東邦大学

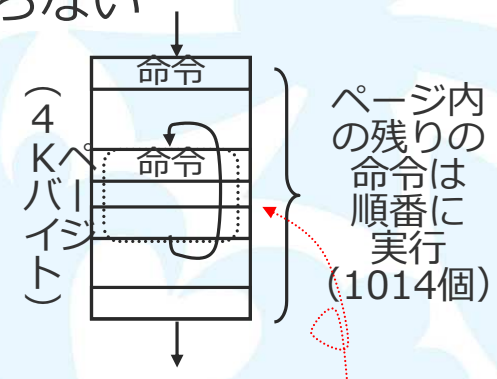
命令の読出しアクセス②

- ループがあると命令の読出しのミス率は低下
  - ループが回っている間はそのページに留まるので、フォールトは起こらない
  - ループが複数ページでもそのページさえ置いておけばフォールトは起こらない



命令の読出しアクセス②

- ループがあると命令の読出しのミス率は低下
  - ループが回っている間はそのページに留まるので、フォールトは起こらない
  - ループが複数ページでもそのページさえ置いておけばフォールトは起こらない
- 例で計算
  - ページ途中に10命令からなるループがあり、10000回まわる

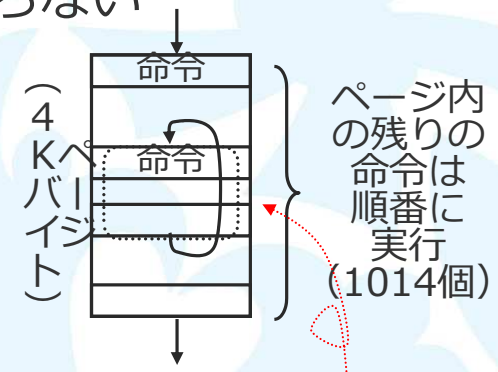


10命令からなるループを10000回まわると
10×10000命令実行

東邦大学

命令の読出しアクセス②

- ループがあると命令の読出しのミス率は低下
  - ループが回っている間はそのページに留まるので、フォールトは起こらない
  - ループが複数ページでもそのページさえ置いておけばフォールトは起こらない
- 例で計算
  - ページ途中に10命令からなるループがあり、10000回まわる
  - 残りの1014命令は１回ずつ実行

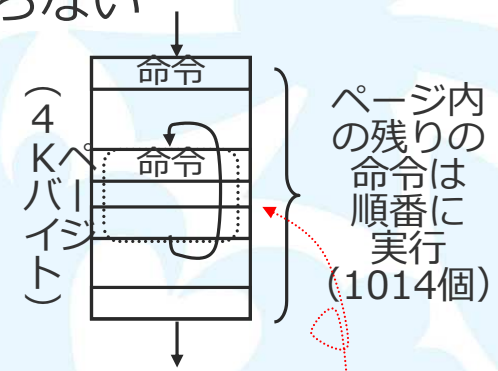


10命令からなるループを10000回まわると10×10000命令実行

命令の読出しアクセス②

- ループがあると命令の読出しのミス率は低下
  - ループが回っている間はそのページに留まるので、フォールトは起こらない
  - ループが複数ページでもそのページさえ置いておけばフォールトは起こらない
- 例で計算
  - ページ途中に10命令からなるループがあり、10000回まわる
  - 残りの1014命令は１回ずつ実行

⇒ フォールトは（1014+(10×10000)）個の命令読出しに対して１回　≒ 10^{-5}



10命令からなるループを10000回まわると10×10000命令実行

要するに命令読出しのミス率は

- ループがあると、読出しのミス率は極端に低下

要するに命令読出しのミス率は

- ループがあると、読出しのミス率は極端に低下
- 実測してみると、ミス率はかなり低くなる

東邦大学

要するに命令読出しのミス率は

- ループがあると、読出しのミス率は極端に低下
- 実測してみると、ミス率はかなり低くなる
  - 実際のプログラムでは「90-10の法則」まがいのことが成立つことが多い
    - «実行時間の90%はコードの10%を実行している»など

要するに命令読出しのミス率は

- ループがあると、読出しのミス率は極端に低下
- 実測してみると、ミス率はかなり低くなる
  - 実際のプログラムでは「90-10の法則」まがいのことが成立つことが多い
    - «実行時間の90%はコードの10%を実行している»など
    - （余談）80-20の法則は「パレートの法則」とも呼ばれている
      数値は厳密でない
      全体の数値の大部分は、全体を構成するうちの一部の要素が生み出しているとする説（経済学らしい）

東邦大学

要するに命令読出しのミス率は

- ループがあると、読出しのミス率は極端に低下
- 実測してみると、ミス率はかなり低くなる
  - 実際のプログラムでは「90-10の法則」まがいのことが成立つことが多い
    - «実行時間の90%はコードの10%を実行している»など
    - （余談）80-20の法則は「パレートの法則」とも呼ばれている
      数値は厳密でない
      全体の数値の大部分は、全体を構成するうちの一部の要素が生み出しているとする説（経済学らしい）
  - 言いだしっぺは、たくさんの実測データによって
    ミス率がかなり低いことを示し、
    だからデマンドページングが使えることを主張した

データのアクセス

- 他方、データのアクセスの場所は原則ランダム
  - 命令のように順番にアクセスするということはない

データのアクセス

- 他方、データのアクセスの場所は原則ランダム
  - 命令のように順番にアクセスするということはない
- しかし、一般にメモリ上では固めて配置される
  - コンパイラ等で変数・定数領域を固めて取る
  - スタック領域は１箇所で固まる

データのアクセス

- 他方、データのアクセスの場所は原則ランダム
  - 命令のように順番にアクセスするということはない
- しかし、一般にメモリ上では固めて配置される
  - コンパイラ等で変数・定数領域を固めて取る
  - スタック領域は１箇所で固まる
- もしアクセスのパターンが
  あちこちに散らばればページフォルトは多発する
  固まっていればフォルトは減少

データのアクセス

- 他方、データのアクセスの場所は原則ランダム
  - 命令のように順番にアクセスするということはない
- しかし、一般にメモリ上では固めて配置される
  - コンパイラ等で変数・定数領域を固めて取る
  - スタック領域は１箇所で固まる
- もしアクセスのパターンが
  あちこちに散らばればページフォルトは多発する
  固まっていればフォルトは減少
- 一般には、かなりの局所性が見られる（実測）
  （数～数十ページの範囲で）
- プログラムに依存（とんでもないプログラムも書ける）

（脱線）とんでもないプログラムを作る

- 試してみよう

（脱線）とんでもないプログラムを作る

- 試してみよう ～ 実験環境 ～ 差が見やすい環境は
  - Ｃ言語（Javaよりエラーが起こらない）で
  - UNIX/Linux環境（Windowsよりエラーが起こらない）

（脱線）とんでもないプログラムを作る

- 試してみよう ～ 実験環境 ～ 差が見やすい環境は
  - Ｃ言語（Javaよりエラーが起こらない）で
  - UNIX/Linux環境（Windowsよりエラーが起こらない）
- プログラム　　配列の要素のアクセス
  - 配列はメモリ上で連続して取られる
  - int A[1024×1024]を確保（要素数は加減せよ）
  - プログラム１では先頭から１つずつA[i]に値を代入
    プログラム２では先頭から1024飛びに値を代入
  - 同じ回数代入文を繰り返した時、かかる時間を測ってみよ
    intが1024飛び ⇒ 4096Bはページの大きさ
  - 時間測定はgettimeofday(ネット参照)を使うとよい

参照の局所性が
どうして思ったより大きいか
納得できましたか？

次へ