東邦大学

生命（いのち）の科学で未来をつなぐ

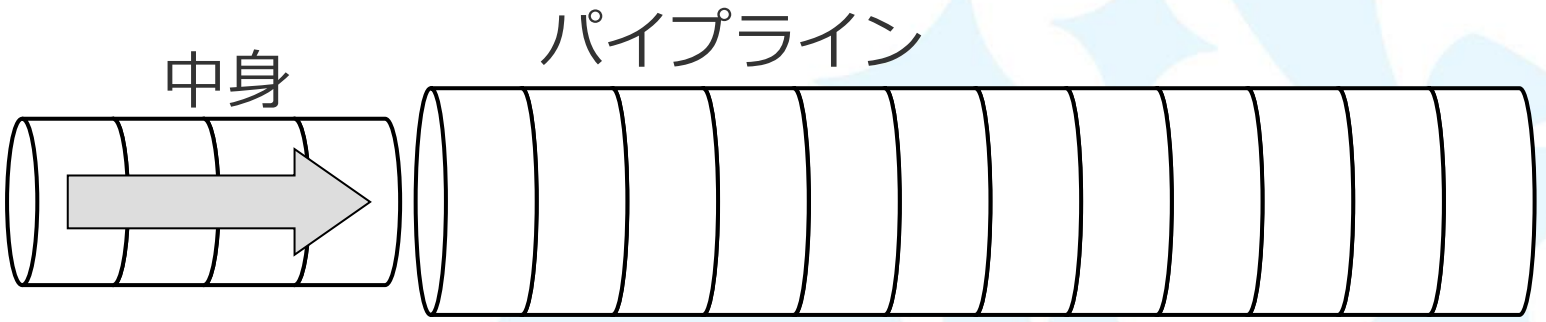
パイプライン

同じ原理を使って
２つのストーリーが

東邦大学

まず原理　～　パイプラインとは

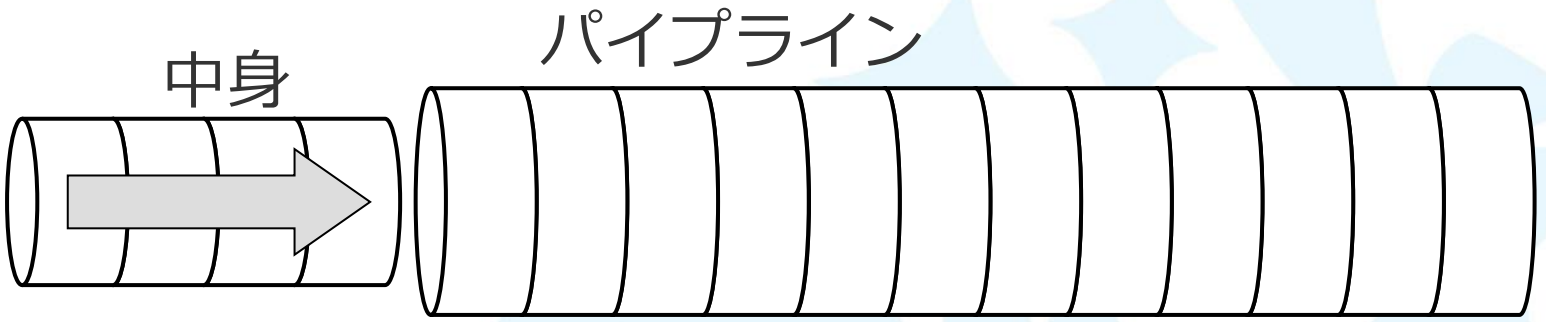
まず原理　～　パイプラインとは



中身をパイプに押し込む
段々に先に進む

東邦大学

まず原理　～　パイプラインとは



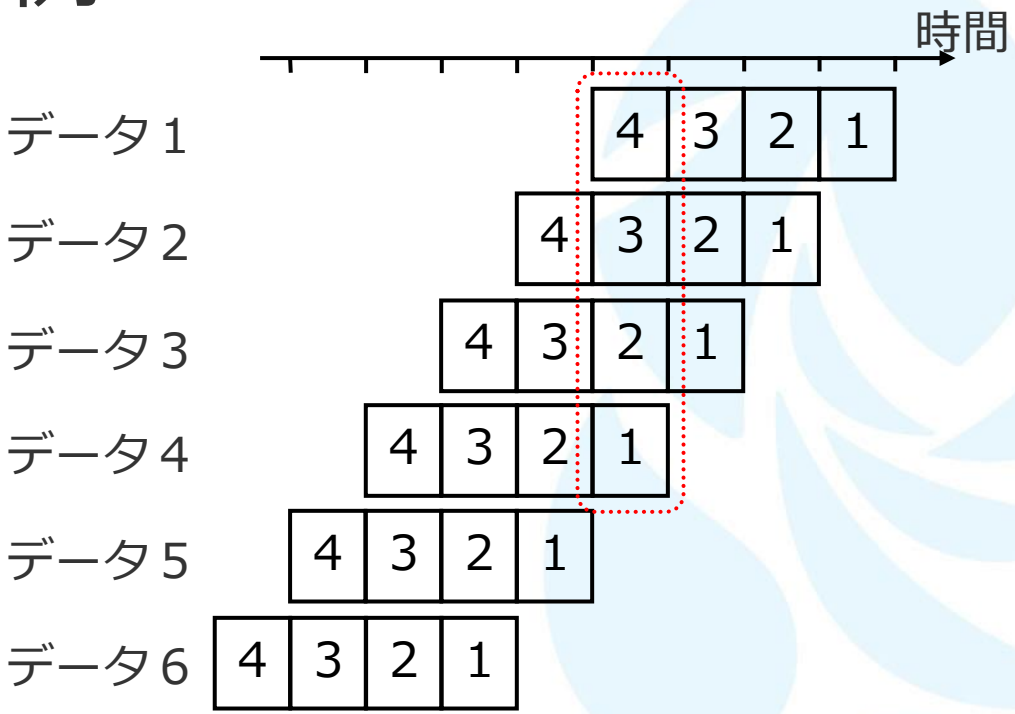
中身をパイプに押し込む
段々に先に進む

パイプは同じ長さのステージに分かれる
中身は１単位時間ごとに１ステージ進む

まず原理　～　パイプラインとは

パイプは同じ長さのステージに分かれる
中身は１単位時間ごとに１ステージ進む

例



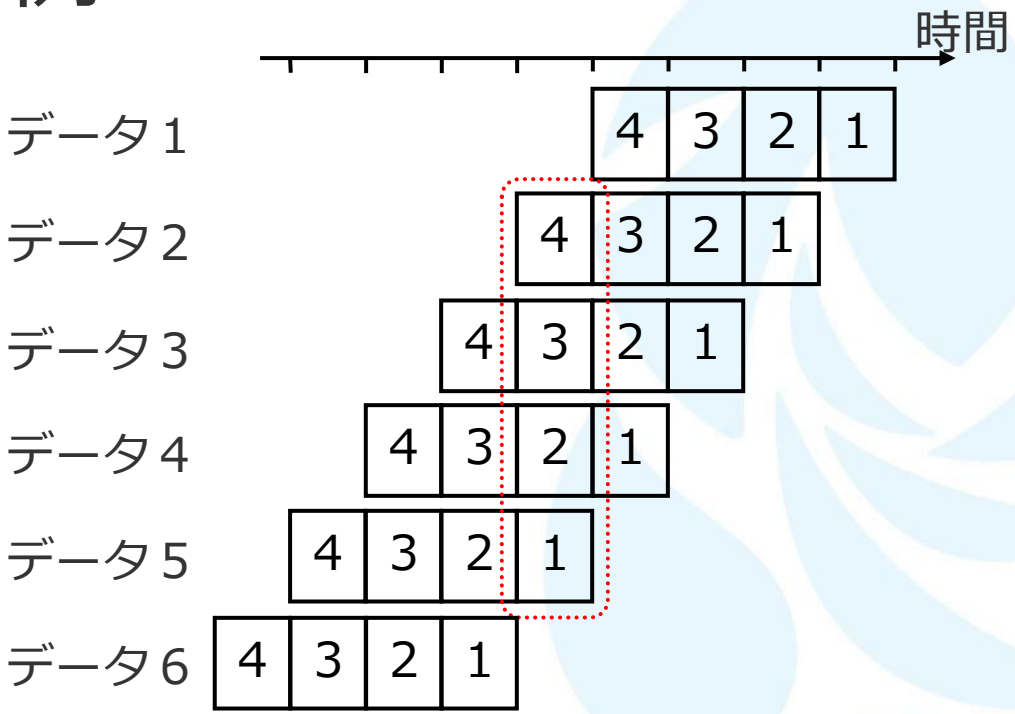
到着するデータは４段で処理される（１～４）
左図時刻では、
　データ１はステージ４
　データ２はステージ３
　データ３はステージ２
　データ４はステージ１
で処理されている

東邦大学

まず原理　～　パイプラインとは

パイプは同じ長さのステージに分かれる
中身は１単位時間ごとに１ステージ進む

例

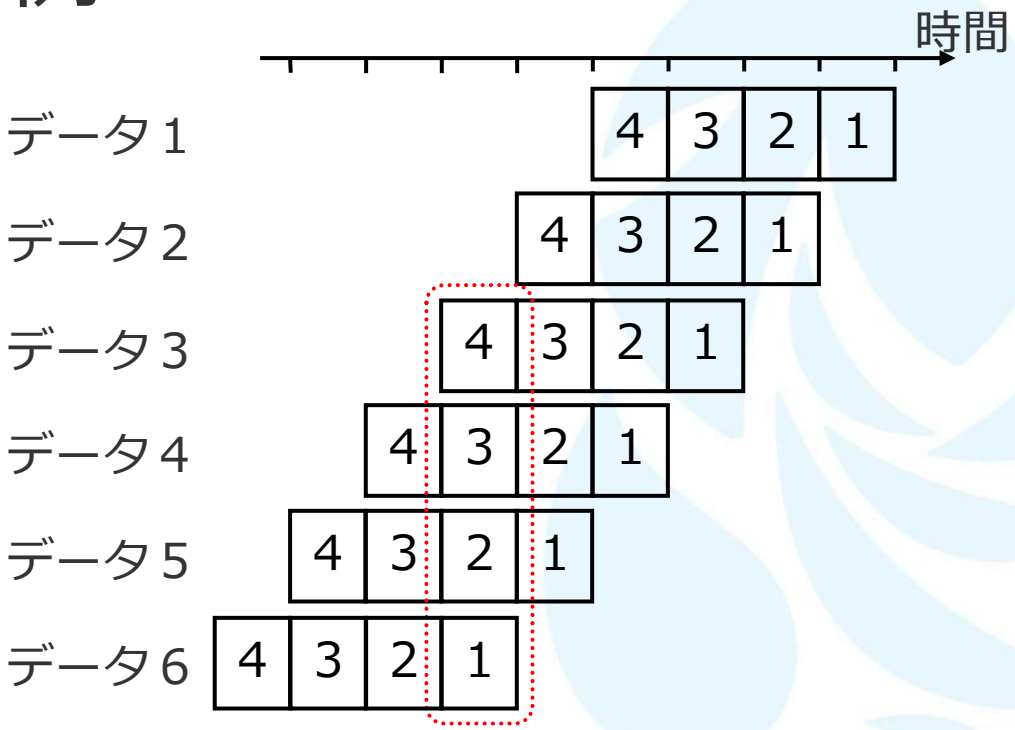


到着するデータは４段で処理される（１～４）
左図時刻では、
　データ２はステージ４
　データ３はステージ３
　データ４はステージ２
　データ５はステージ１
で処理されている

まず原理　～　パイプラインとは

パイプは同じ長さのステージに分かれる
中身は１単位時間ごとに１ステージ進む

例



到着するデータは４段で処理される（１～４）
左図時刻では、
　データ３はステージ４
　データ４はステージ３
　データ５はステージ２
　データ６はステージ１
で処理されている

東邦大学

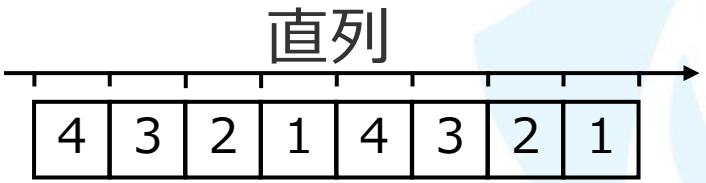
まず原理　～　パイプラインとは

このように、パイプラインでは
　それぞれのステージが並行して動作する

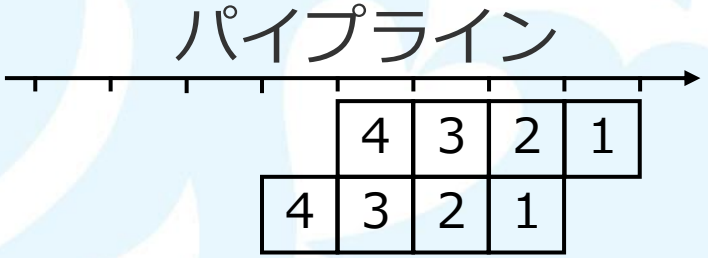
まず原理　～　パイプラインとは

このように、パイプラインでは
　それぞれのステージが並行して動作する

⇒　処理が早くなる



2データで8クロック



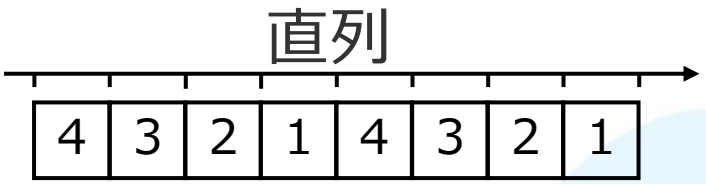
2データで5クロック

どれだけ高速化するか

Sステージ、Nデータの所要時間は？

どれだけ高速化するか

Sステージ、Nデータの所要時間は？



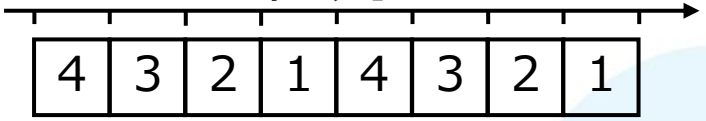
NデータでN×Sクロック

東邦大学

どれだけ高速化するか

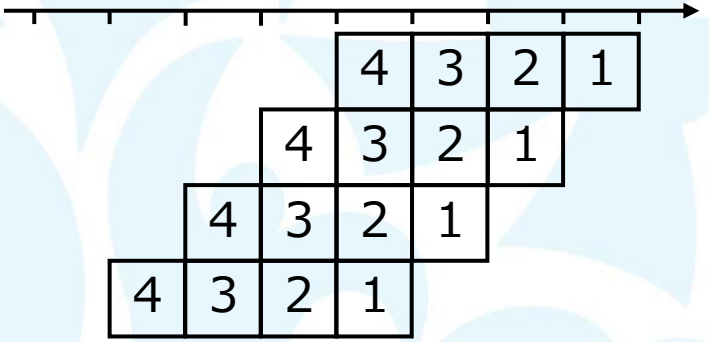
Sステージ、Nデータの所要時間は？

直列



NデータでN×Sクロック

パイプライン

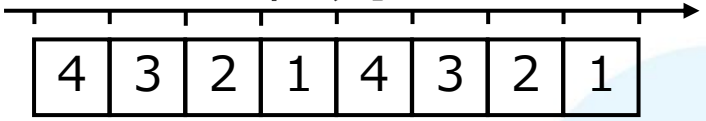


NデータでN+S-1クロック

どれだけ高速化するか

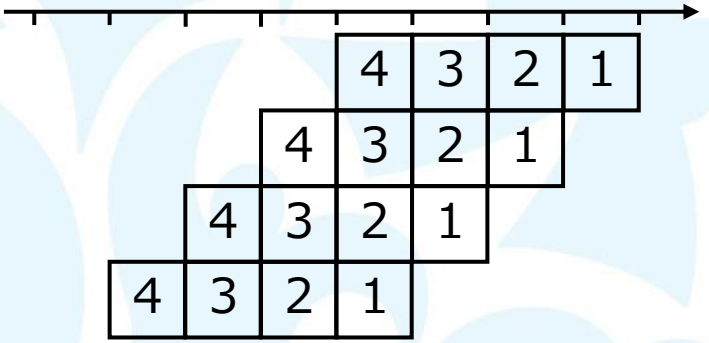
Sステージ、Nデータの所要時間は？

直列



NデータでN×Sクロック

パイプライン



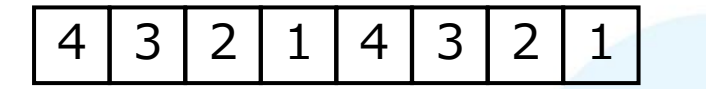
NデータでN+S-1クロック

加速率＝T(直列)/T(パイプライン)＝

どれだけ高速化するか

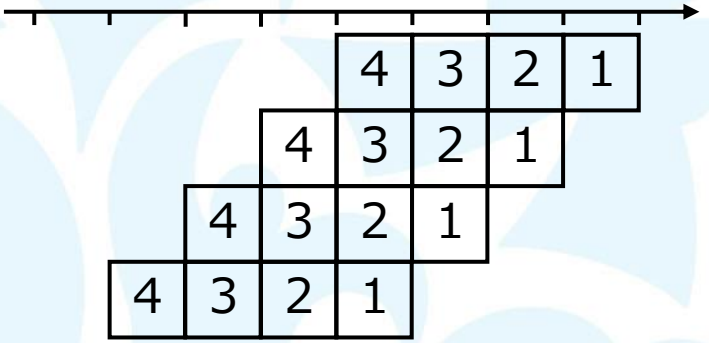
Sステージ、Nデータの所要時間は？

直列



NデータでN×Sクロック

パイプライン



NデータでN+S-1クロック

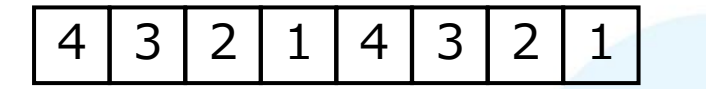
加速率＝T(直列)/T(パイプライン)＝(N×S)/(N+S-1)

東邦大学

どれだけ高速化するか

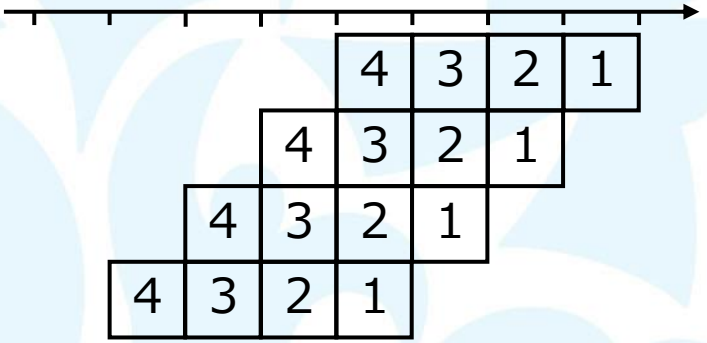
Sステージ、Nデータの所要時間は？

直列



NデータでN×Sクロック

パイプライン



NデータでN+S-1クロック

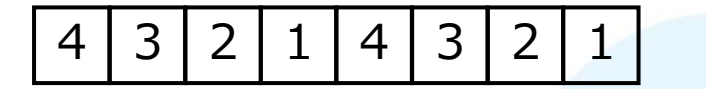
加速率＝T(直列)/T(パイプライン)＝(N×S)/(N+S-1)

データが無限に続くとすると

どれだけ高速化するか

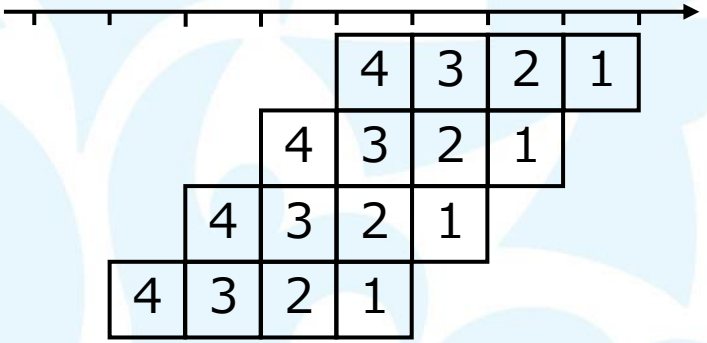
Sステージ、Nデータの所要時間は？

直列



NデータでN×Sクロック

パイプライン



NデータでN+S-1クロック

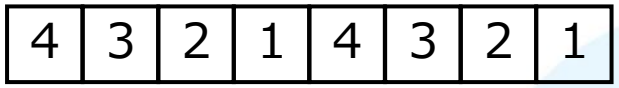
加速率＝T(直列)/T(パイプライン)＝(N×S)/(N+S-1)

データが無限に続くとすると（分母・子をNで割って）

どれだけ高速化するか

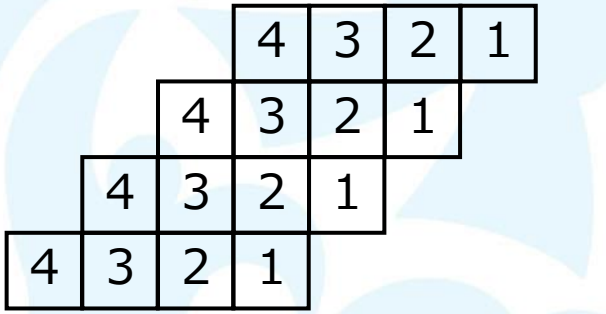
Sステージ、Nデータの所要時間は？

直列



NデータでN×Sクロック

パイプライン



NデータでN+S-1クロック

加速率＝T(直列)/T(パイプライン)＝(N×S)/(N+S-1)

データが無限に続くとすると（分母・子をNで割って）

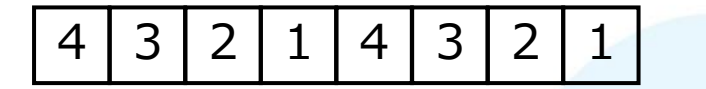
加速率＝ $\lim_{(N\to\infty)} S/(1+(S-1)/N)=S$ ⇒ S倍

東邦大学

どれだけ高速化するか

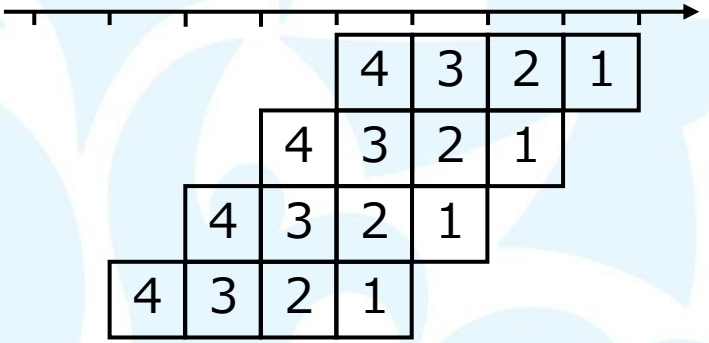
Sステージ、Nデータの所要時間は？

直列



NデータでN×Sクロック

パイプライン



NデータでN+S-1クロック

加速率＝T(直列)/T(パイプライン)＝(N×S)/(N+S-1)

データが無限に続くとすると（分母・子をNで割って）

加速率＝ $\lim_{(N\to\infty)}$ S/(1+(S-1)/N)＝S　⇒　S倍

S（ステージ数）倍　速くなる

東邦大学

実際の使われ方　２つ

実際の使われ方　２つ

使い方１）　命令パイプライン

使い方２）　データパイプライン

命令パイプライン

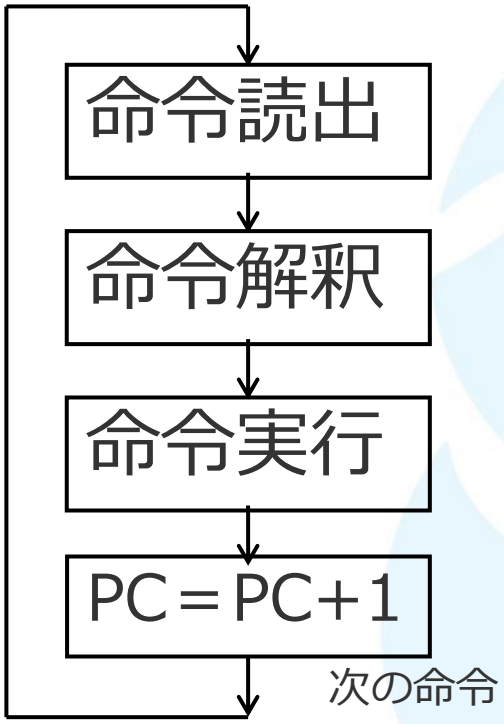
１つ１つの命令の「実行サイクル」？

思い出してみよう

命令パイプライン

１つ１つの命令の「実行サイクル」？

思い出してみよう



１つの命令は４つのステップで実行される

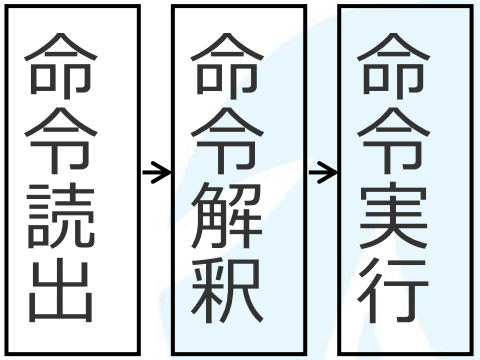
これをぐるぐる回りながら
次々と命令を順番に実行する

命令パイプライン

で

命令パイプライン

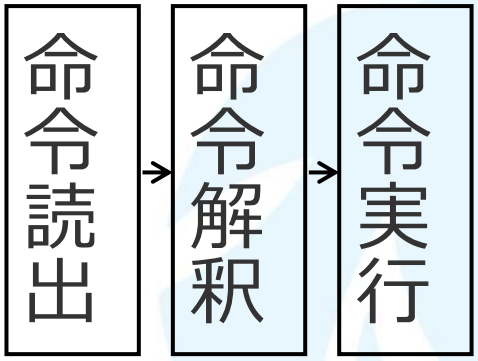
これらのステップをステージとして
パイプラインを作れる

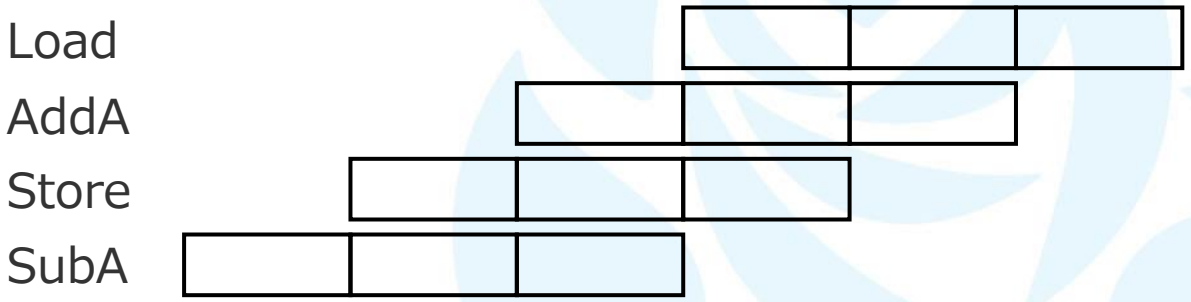


PC←PC+1は
高速なので
読出ステージ
に含める

命令パイプライン

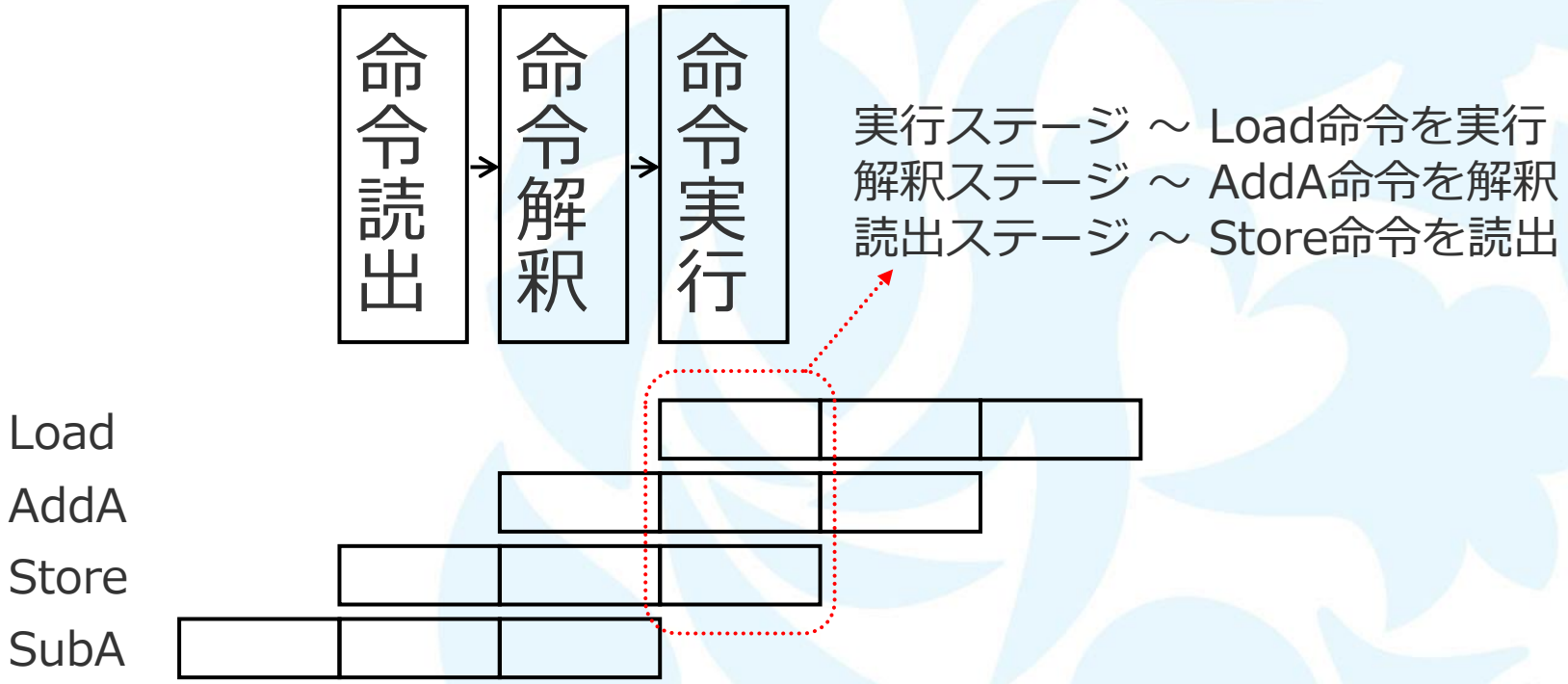
これらのステップをステージとして パイプラインを作れる





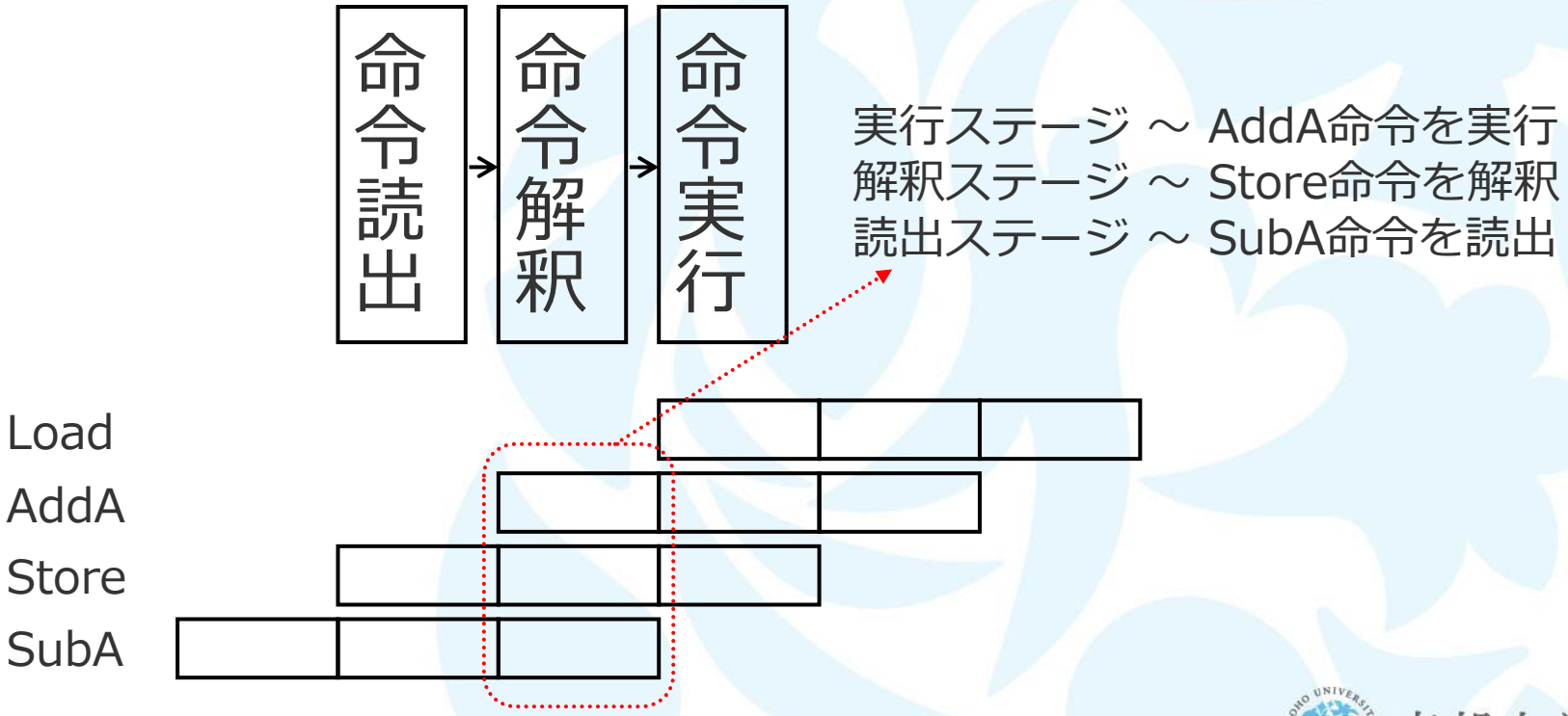
命令パイプライン

これらのステップをステージとして パイプラインを作れる



命令パイプライン

これらのステップをステージとして パイプラインを作れる



命令パイプライン

このように１つの命令の実行を
複数のステージに分割して
パイプライン化する
⇒　命令パイプライン

命令パイプライン

このように１つの命令の実行を
　複数のステージに分割して
　　パイプライン化する
　　⇒　命令パイプライン

ステージ数を増やせば、並列度が上がり
　高速化できる

　　Intel Core i7では16段程度まで増やしている

データパイプライン

同じ処理をしたいデータが
たくさんあるときに、パイプライン化

どういうときだろうか？

データパイプライン

同じ処理をしたいデータが
　たくさんあるときに、パイプライン化

ベクトルや行列の計算
　物理的なシミュレーション計算など

データパイプライン

同じ処理をしたいデータが
　たくさんあるときに、パイプライン化

ベクトルや行列の計算
　物理的なシミュレーション計算など
　画像処理・生成など

データパイプライン

同じ処理をしたいデータが
　たくさんあるときに、パイプライン化

ベクトルや行列の計算
　物理的なシミュレーション計算など
　画像処理・生成など

物理的なシミュレーション計算など　⇒
　すべての点に対して同じ計算をする（ベクトル・行列）
　昔のスパコンは「ベクトル計算機」であった
　　例：　CDC-6600 (1964)、Cray-1 (1976)

データパイプライン

同じ処理をしたいデータが
　たくさんあるときに、パイプライン化

ベクトルや行列の計算
　物理的なシミュレーション計算など
　画像処理・生成など

画像処理・生成など　⇒
　最近のPCやグラフィックアダプタは画像処理プロセッサ(GPU) を積んでいるが、これらはポリゴンやピクセルごとに同一処理をするパイプライン処理が主体になっている

パイプラインの用途のまとめ

命令パイプライン

データパイプライン

パイプラインの用途のまとめ

命令パイプライン

・１つ１つの命令実行を、パイプラインに組む
・読出・解釈・実行（更にここを多段にする）
・PC用を含む広範なプロセッサで実用している

データパイプライン

・多数のデータに対し同じ処理を行う時に有効
・ベクトル・行列のような数値計算（物理シミュレーションなど）や
グラフィック処理（画像の生成や処理）・マルチメディア処理（画像・音声など）
が中心

命令パイプラインの追加情報

- Intel NetBurstアーキテクチャ (Pentium4, 2000年)
  - PentiumⅢが10段 ⇒ Pen 4で20段 ⇒ Prescottで31段
  - 「スーパーパイプライン」
  - （単純なステップ×短クロック）⇒　高速化を狙う
    ⇒（電力・発熱大＋ハザード大）で結局3GHz迄で中止
- その後Core2アーキテクチャで14段
  Nehalem(第１世代Core i) 16段?
  Sandy Bridge（第２世代Core i）　16段?
  Haswell も ほぼ踏襲らしい（14段?16段?）
- 興味があればIntel x86系のアーキテクチャの進化について調べてみよう

パイプラインの考え方が
分かりましたか？

東邦大学