ハーバードアーキテクチャ

ハーバードアーキテクチャとは？

CPUの内部構造の作り方
の１つ
「ノイマン型」を少しはみ出す

復習：　「ノイマン式」

何だったか思い出してみよう

復習：　「ノイマン式」

何だったか思い出してみよう

1. プログラム(可変)内蔵方式
2. 逐次処理方式
3. 単一メモリ方式

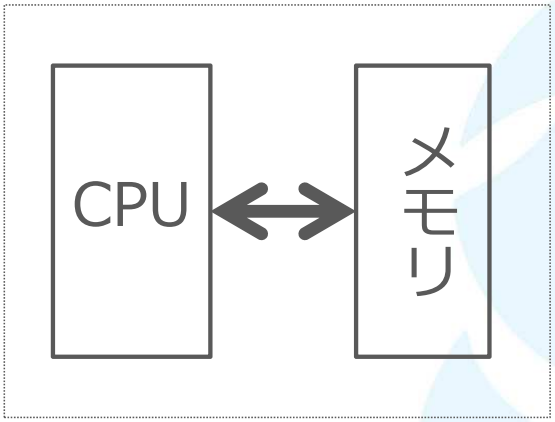
復習：　「ノイマン式」

何だったか思い出してみよう

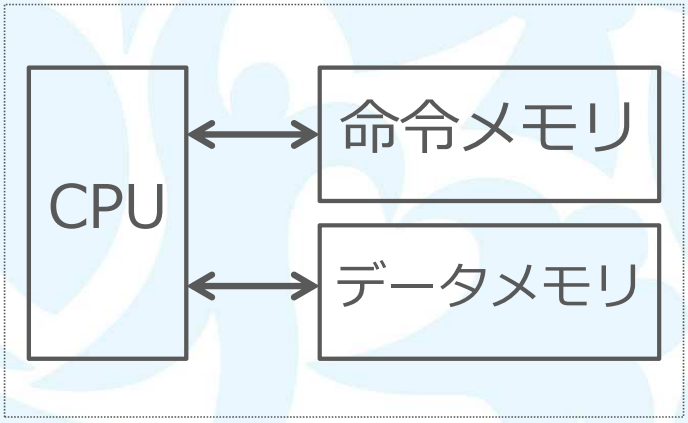
1. プログラム(可変)内蔵方式
2. 逐次処理方式
3. 単一メモリ方式 ⇦ これを はみ出す

ハーバードアーキテクチャとは

ノイマン型
アーキテクチャ

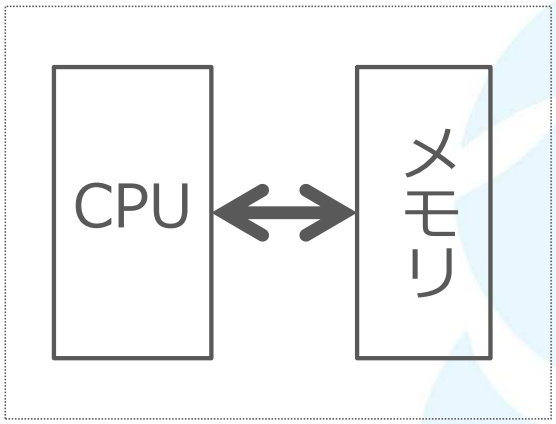


ハーバード
アーキテクチャ



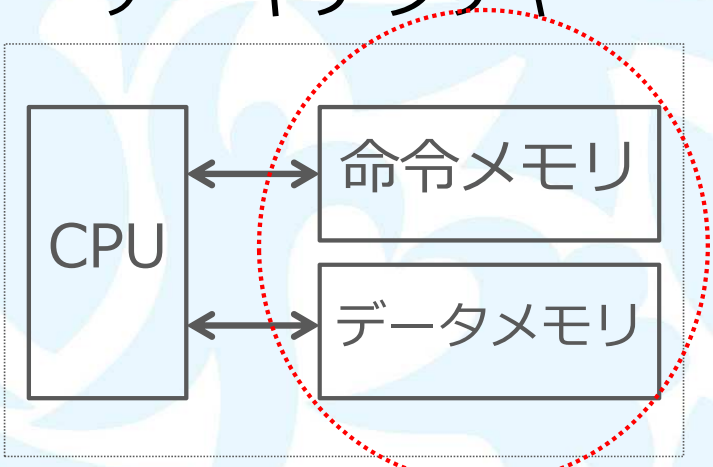
ハーバードアーキテクチャとは

ノイマン型
アーキテクチャ



単一メモリ

ハーバード
アーキテクチャ



２つのメモリ

ハーバードアーキテクチャの考え方

ノイマン型では１本だけなので
アクセスが混雑する　～　「隘路」

ハーバードアーキテクチャの考え方

ノイマン型では１本だけなので
　アクセスが混雑する　～　「隘路」

⇩

２本別にすればよい？

ハーバードアーキテクチャの考え方

ノイマン型では１本だけなので
　アクセスが混雑する　～　「隘路」

⇩

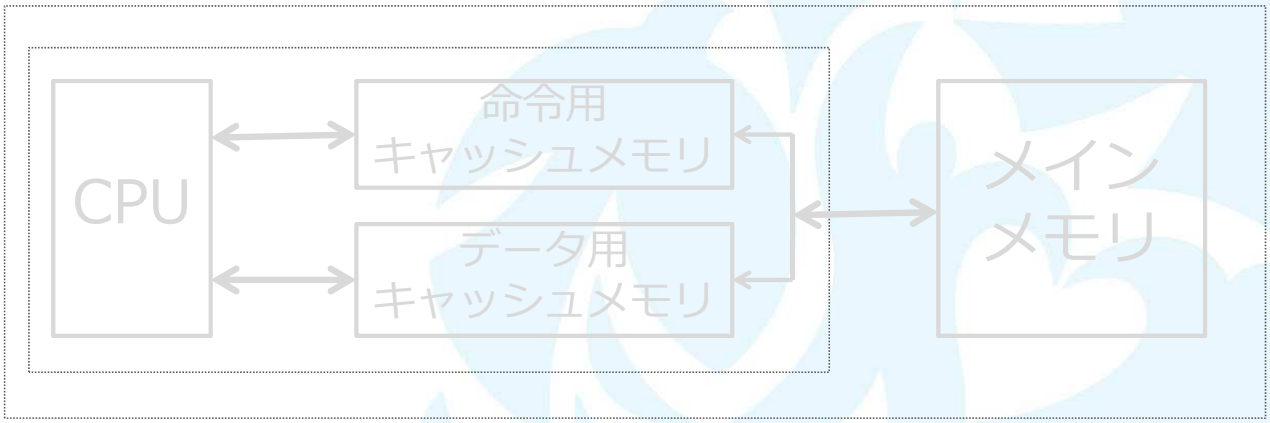
２本別にすればよい？

⇩

メモリを予め区分
　⇒　容量の柔軟性が失われる

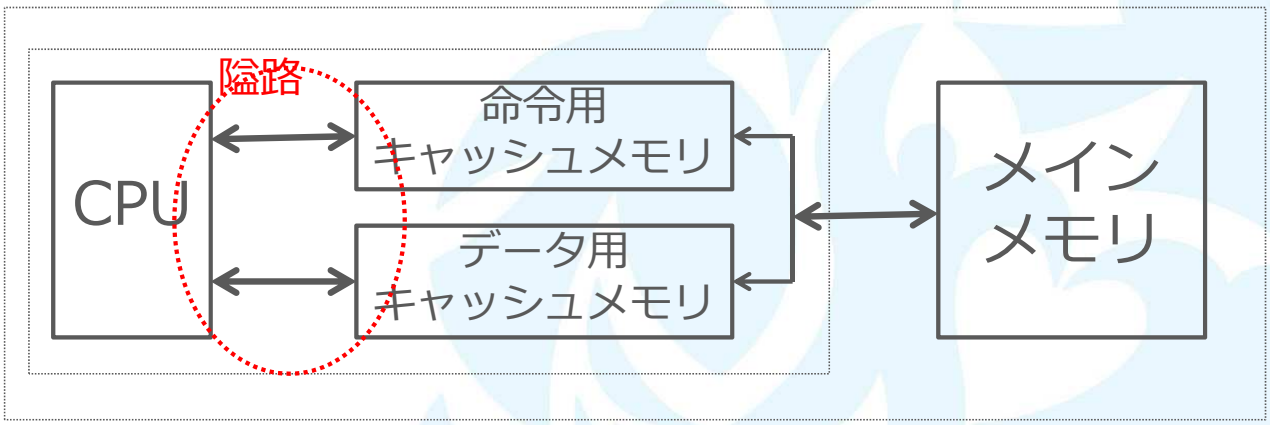
ハーバードアーキテクチャの考え方

現実には「キャッシュ」(後述)までの
経路を２本にするのが普通



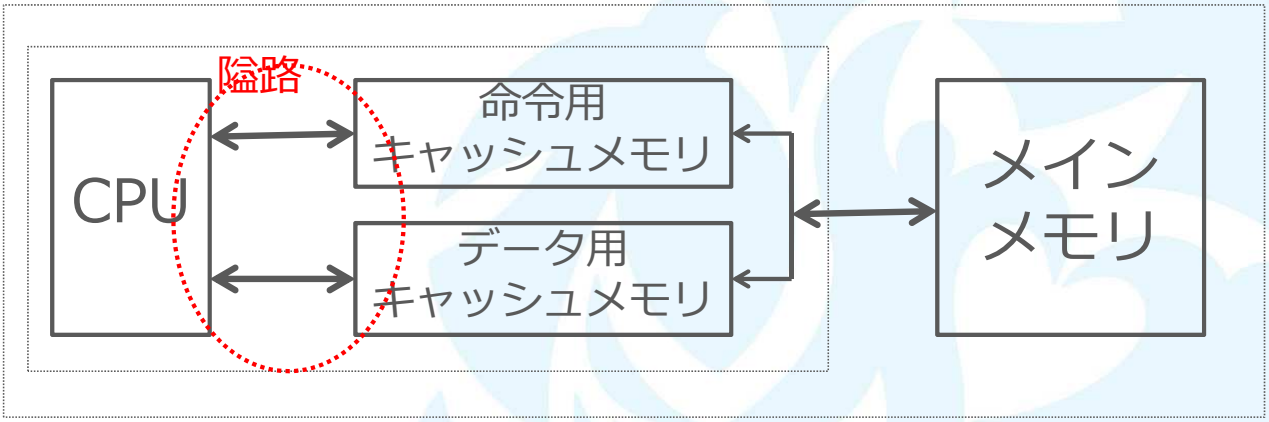
ハーバードアーキテクチャの考え方

現実には「キャッシュ」(後述)までの
経路を２本にするのが普通



ハーバードアーキテクチャの考え方

現実には「キャッシュ」(後述)までの
経路を２本にするのが普通



更に、命令とデータでアクセスの動きが違い、
キャッシュの振舞いが変わるので、別々がよい

ハーバードアーキテクチャのまとめ

ハーバードアーキテクチャとは

[　　]を[　　]分離

⇒ [　　]避けられる

⇒ 代償 [　　]

⇒ 現状 [　　]を分離

ハーバードアーキテクチャのまとめ

ハーバードアーキテクチャとは
メモリへの経路をデータと命令に分離

⇒ 避けられる

⇒ 代償

⇒ 現状 を分離

ハーバードアーキテクチャのまとめ

ハーバードアーキテクチャとは
メモリへの経路をデータと命令に分離

⇒ メモリアクセスの混雑が避けられる

⇒ 代償

⇒ 現状 を分離

ハーバードアーキテクチャのまとめ

ハーバードアーキテクチャとは
メモリへの経路をデータと命令に分離

⇒ メモリアクセスの混雑が避けられる

⇒ 代償　メモリを区分～柔軟性に欠ける

⇒ 現状　　を分離

ハーバードアーキテクチャのまとめ

ハーバードアーキテクチャとは
メモリへの経路をデータと命令に分離

⇒ メモリアクセスの隘路が避けられる

⇒ 代償　メモリを区分～柔軟性に欠ける

⇒ 現状　CPU～キャッシュまでを分離

ハーバードアーキテクチャの考え方について分かりましたか？

次へ
(マイクロプログラムへ)

マイクロプログラムアーキテクチャ

マイクロプログラムアーキテクチャとは？

CPUの内部構造の作り方
の１つ

作り方の工夫

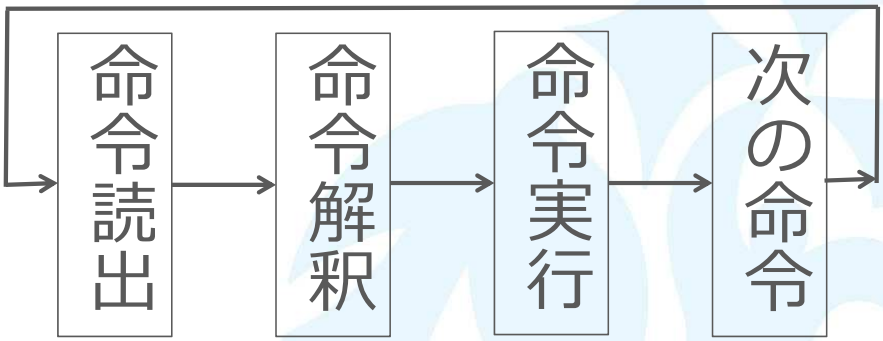
マイクロプログラムの考え方

１つの命令の実行は４つのステップ

何だったかな？

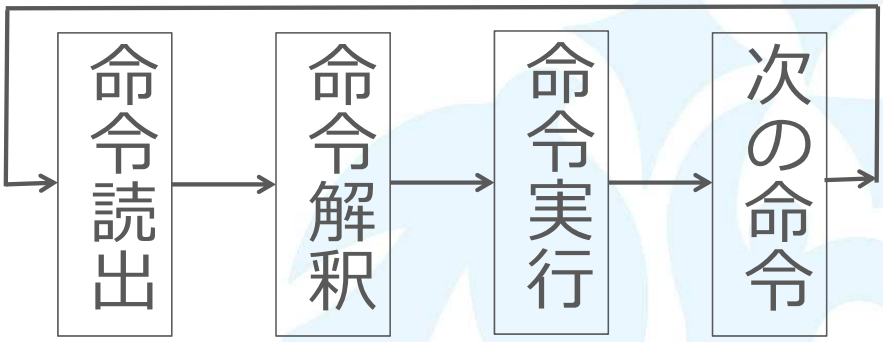
マイクロプログラムの考え方

１つの命令の実行は４つのステップ



マイクロプログラムの考え方

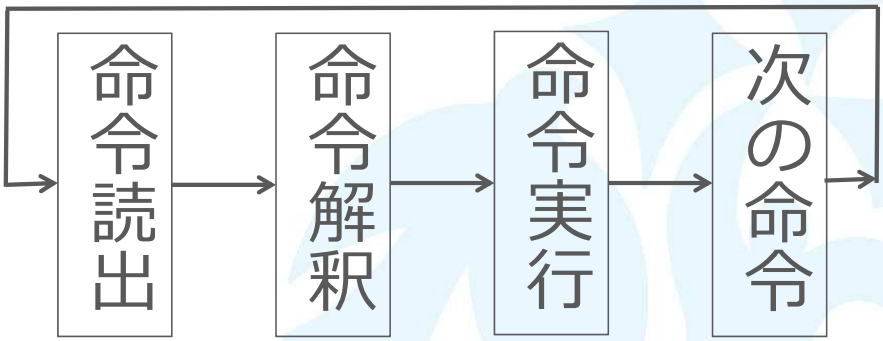
１つの命令の実行は４つのステップ



４つのステップの制御を
ハード(論理回路)で実現してもいいけれど
プログラムもどきで制御してもいいよね

マイクロプログラムの考え方

１つの命令の実行は４つのステップ

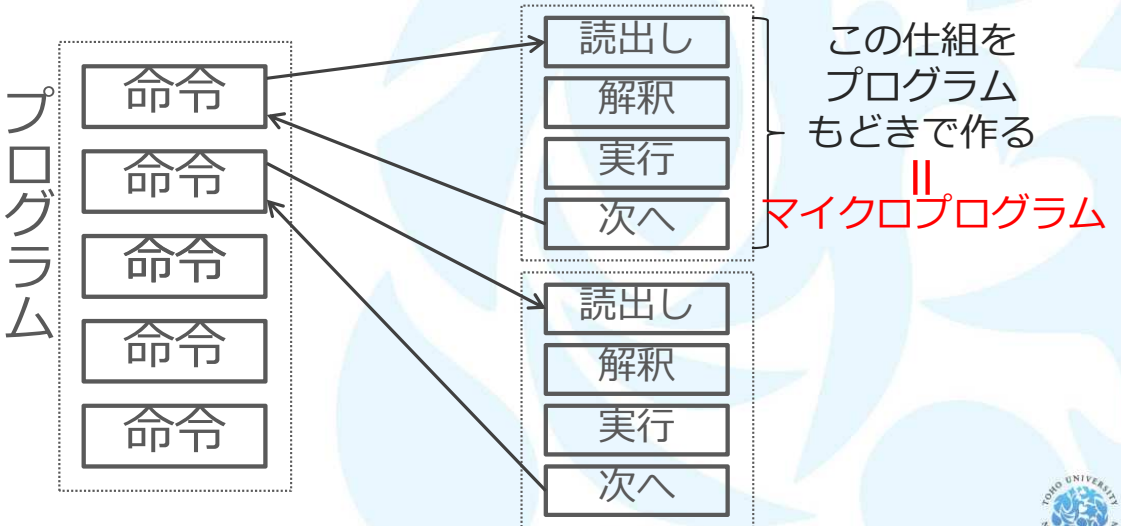


４つのステップの制御を
ハード(論理回路)で実現してもいいけれど
プログラムもどきで制御してもいい？

マイクロプログラムの考え方

１つの命令の実行の４ステップの制御を
プログラムもどきで行う

マイクロプログラムと呼ぶ



マイクロプログラムの考え方

言ってしまえば

非常に高速で単純なコンピュータがあって
それが１つ１つの命令を読出し・解釈・実行
しているモデル

マイクロプログラムの考え方

マイクロプログラム　ハード(布線論理)

ハードは簡単　　ハードは複雑

マイクロプログラムの考え方

マイクロプログラム

ハードは簡単
変更も簡単

ハード(布線論理)

ハードは複雑
変更は作り直しで大変

マイクロプログラムの考え方

マイクロプログラム

ハードは簡単
変更も簡単
短時間で作れる

ハード(布線論理)

ハードは複雑
変更は作り直しで大変
作るのに時間がかかる

マイクロプログラムの考え方

マイクロプログラム　ハード(布線・回路)

ハードは簡単
変更も簡単
短時間で作れる
どうしても遅い

ハードは複雑
変更は作り直しで大変
作るのに時間がかかる
基本的に速い

マイクロプログラムの考え方

現実的には、組合せる考え方がある

シリーズ（ファミリー）に使われた

マイクロプログラムの考え方

現実的には、組合せる考え方がある

シリーズ（ファミリー）に使われた

高速だが高価な機種

布線(回路)による実現

ファミリー

ソフトは同じに使える

マイクロプログラム

低速だが安価な機種

マイクロ命令の形式

マイクロ命令の作り方は
２通り考えられた

マイクロ命令の形式

マイクロ命令の作り方は
２通り考えられた

水平型　　垂直型

命令の各ビットを
内部信号に対応させる

高速(内部処理が無い)
マイクロステップ数少ない
命令長が非常に長くなる

マイクロ命令の形式

マイクロ命令の作り方は
２通り考えられた

水平型　　垂直型

内部信号をエンコード

命令の各ビットを
内部信号に対応させる

命令長は短い
低速(デコード必要)
マイクロステップ数多い

高速(内部処理が無い)
マイクロステップ数少ない
命令長が非常に長くなる

マイクロ命令の形式

マイクロ命令の作り方は
２通り考えられた

水平型	垂直型
命令の各ビットを内部信号に対応させる	内部信号をエンコード
高速(内部処理が無い)	命令長は短い
マイクロステップ数少ない	低速(デコード必要)
命令長が非常に長くなる	マイクロステップ数多い

マイクロプログラムのまとめ

マイクロプログラムアーキテクチャは
[　　][　　]の実現法の
議論である

マイクロプログラムは[　　]と対比する

マイクロプログラムは
各命令を[　　]で実現

[　　]は
各命令を[　　]で実現

マイクロプログラムのまとめ

マイクロプログラムアーキテクチャは
　１つの命令の実行サイクルの実現法の
　　議論である
マイクロプログラムは　　と対比する
マイクロプログラムは
　各命令を　　で実現
　　は
　各命令を　　で実現

マイクロプログラムのまとめ

マイクロプログラムアーキテクチャは
　１つの命令の実行サイクルの実現法の
　　議論である
マイクロプログラムは布線論理と対比する
マイクロプログラムは
　各命令を　　で実現
　　は
　各命令を　　で実現

マイクロプログラムのまとめ

マイクロプログラムアーキテクチャは
　１つの命令の実行サイクルの実現法の
　　議論である
マイクロプログラムは布線論理と対比する
マイクロプログラムは
　各命令を複数のマイクロ命令で実現
　　は
　各命令を　　で実現

マイクロプログラムのまとめ

マイクロプログラムアーキテクチャは
　１つの命令の実行サイクルの実現法の
　　議論である
マイクロプログラムは布線論理と対比する
マイクロプログラムは
　各命令を複数のマイクロ命令で実現
布線論理は
　各命令を　　で実現

マイクロプログラムのまとめ

マイクロプログラムアーキテクチャは
　１つの命令の実行サイクルの実現法の
　　議論である

マイクロプログラムは布線論理と対比する

マイクロプログラムは
　各命令を複数のマイクロ命令で実現

布線論理は
　各命令をハードの回路で実現

マイクロプログラムアーキテクチャの考え方について分かりましたか？

次へ
(命令の実行性能)