

コンピュータ アーキテクチャ I 第1回

塩谷 亮太 (shioya@ci.i.u-tokyo.ac.jp)

東京大学大学院情報理工学系研究科 創造情報学専攻

自己紹介

■ 塩谷 亮太（しおや りょうた）

- 東京大学大学院 情報理工学系研究科
創造情報学専攻 准教授
- お茶大には非常勤講師として来ています

■ 専門：

- コンピュータのハードウェア
 - 特に CPU と呼ばれる部分が専門です
- コンパイラとか OS などの基盤ソフトウェア

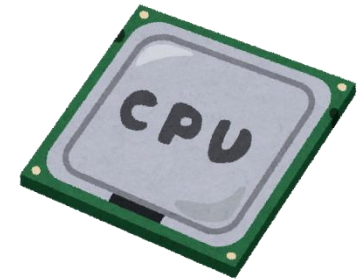
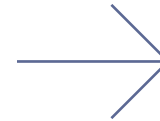
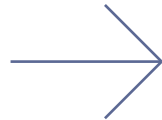
今年度特有の事情

- 昨年度まで担当されていた粕川先生から塩谷に急遽交代
- なので、正直いろいろ手探りです
 - 学部向けのアーキテクチャの講義を担当したことがない
 - 講義資料のストックもない
 - （講義資料を作るのはめっさ大変です
 - お茶大の事もよくわからない
- 色々と試行錯誤しながら進めると思います
 - よろしくお願いします

コンピュータ・アーキテクチャとは

- 「アーキテクチャ」
 - 建築そのものや, 建築における設計や様式のこと
- 「コンピュータ・アーキテクチャ」
 - コンピュータにおける設計や様式のこと
 - 情報分野では単に「アーキテクチャ」と呼ぶことも多い

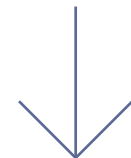
ソフトウェアとハードウェア



ソフトウェアに 書かれている命令に従って

ハードウェアは
計算して結果を出力

- コンピュータは
ソフトウェアとハードウェアからなる
- コンピュータが
「どうやって動いているのか？」
「どう作られているのか？」を
理解するのがこの講義の目的



講義の進め方：

基本的には去年の内容をそのままカバーする予定

■ 去年の内容：

- ノイマン型コンピュータ
- コンパイラや OS, 仮想マシンなどの階層化
- バスや CPU, ALU のコンピュータ構造, 命令
- IPC などの性能指標, RISC/CISC
- パイプライン化, スーパスカラ, 投機,
- 割り込み, DMA
- 論理回路, ALU, 遅延, カルノー図による最適化
- キャッシュ, 局所性, メモリの種類
- 仮想記憶, MMU, 浮動小数点数

講義資料と連絡先

- 講義資料は以下にアップロードします
 - <https://github.com/shioyadan/otya-computer-architecture-i>
- 連絡先：以下のメールまでお願いします
 - shioya@ci.i.u-tokyo.ac.jp
- これらの情報はオンラインのシラバスにも書いておいたので、必要な場合はそちらから

参考資料



参考資料 1

- アンドリュー・S. タネンバウム：
「構造化コンピュータ構成」
 - 去年までの講義はこの教科書の内容に沿っていた…らしい
 - コンピュータのハードから OS まで広くカバー
 - 20年前の出版なので、やや内容が古いかもしれない

参考資料 2

- デイビッド・A. パターソン, ジョン・L. ヘネシー :
「コンピュータの構成と設計」 第6 版
 - 通称「**パタヘネ**」
 - アーキテクチャの教科書的世界的な鉄板その1
 - こっちの方が基礎的

- ジョン・L. ヘネシー, デイビッド・A. パターソン
「コンピュータアーキテクチャ 定量的アプローチ」 第6 版
 - 通称「**ヘネパタ**」
 - アーキテクチャの教科書的世界的な鉄板その2
 - こっちの方が難しい

- Noam Nisan, Shimon Schocken :
「コンピュータシステムの理論と実装 —モダンなコンピュータの作り方」
 - 論理回路から OS, その上で動くアプリまでをカバー
 - 「NAND to Tetris」
 - NAND : もっとも基礎的な論理回路
 - Tetris : 有名なゲーム
 - 全体にコンパクトかつ演習形式でまとまっている

その他の参考資料

- 去年の粕川先生のコンピュータアーキテクチャの講義資料：
 - <http://www.kasukawa.net>
- 塩谷が大学院でやっている「先進計算機構成論」の講義資料：
 - <https://github.com/shioyadan/advanced-computer-organization>

前提とする知識

- 必修で1年生の時にやっていると聞いたので、以下の基本のきがわかっている前提で進めます
 - C言語のプログラミング
 - 論理回路
- （と言う予定なんですが、大丈夫でしょうか？

- 期末試験を行う予定です
 - 普段の課題も成績に入れる予定です
 - … が, 試行錯誤して色々変更になるかもしれません

もくじ

1. コンピュータ・アーキテクチャとは？
2. コンピュータの種類
3. ソフトウェアとの関係

コンピュータ・アーキテクチャとは（再）

- 「アーキテクチャ」
 - 建築そのものや, 建築における設計や様式のこと
- 「コンピュータ・アーキテクチャ」
 - コンピュータにおける設計や様式のこと
 - 情報分野では単に「アーキテクチャ」と呼ぶことも多い

広義と狭義のコンピュータ・アーキテクチャ

- 広義にはハードとソフトの双方を含めたコンピュータ全体の設計や様式をのことを言う（と思う）
 - この講義のタイトルはこちらの意味
- 狭義にはハードの設計や様式のことを言う
 - 一般にはこちらの意味で使うことの方が多い
 - 塩谷は口頭ではこっちの意味で使うことが多いと思う

コンピュータ・アーキテクチャ I と II

- 1 学期と 2 学期に, それぞれ I と II があります
 - 2 学期の II は東京農工大の山田先生が担当予定です
 - 山田先生は日本を代表する OS の先生です
- コンピュータ全体を,
 - アーキテクチャ I の講義ではハードウェア面から
アーキテクチャ II の講義ではソフトウェア面から 見ていきます

コンピュータに関わる分野の階層関係

上にあるものは、その直下にあるものを使って構築されている

アプリケーション・ソフトウェア
画像処理 / 音声認識 / 言語処理 / 機械制御
機械学習 / AI / WEB サービス / 暗号 ...

システム・ソフトウェア
OS / コンパイラ / インタプリタ

(広義の)
コンピュータ
アーキテクチャの範囲

(狭義の) コンピュータ・アーキテクチャ

論理回路

集積回路 / 半導体デバイス

コンピュータ・アーキテクチャ I では ハードウェア面からみた話題を中心に

この講義では
下側から見て
説明



アプリケーション・ソフトウェア
画像処理 / 音声認識 / 言語処理 / 機械制御
機械学習 / AI / WEB サービス / 暗号 ...

システム・ソフトウェア
OS / コンパイラ / インタプリタ

(狭義の) コンピュータ・アーキテクチャ

論理回路

集積回路 / 半導体デバイス

コンピュータ・アーキテクチャⅡでは ソフトウェア面からみた話題を中心に

アプリケーション・ソフトウェア
画像処理 / 音声認識 / 言語処理 / 機械制御
機械学習 / AI / WEB サービス / 暗号 ...


システム・ソフトウェア
OS / コンパイラ / インタプリタ

(狭義の) コンピュータ・アーキテクチャ

論理回路

集積回路 / 半導体デバイス

来学期の
コンピュータ
アーキテクチャⅡ
では上側から見て
説明



余談：応用系と基盤系

■ 応用系（アプリ系とも）：

- 基盤となる分野を
応用した分野
- 直接人が使う応用を
扱う
- 割と派手でわかりやすい

アプリケーション・ソフトウェア
画像処理 / 音声認識 / 言語処理 / 機械制御
機械学習 / AI / WEB サービス / 暗号 ...

■ 基盤系（システム系とも）：

- 応用となる分野の
基盤となる分野
- 直接人の目に触れない
下支えをする部分を扱う
- 割と地味でわかりにくい

システム・ソフトウェア
OS / コンパイラ / インタプリタ

(狭義の)
コンピュータ・アーキテクチャ

- (あくまで傾向なので、右の図に直接当てはまらない場合もある

余談：応用系と基盤系 処理の高速化を例とした場合の例

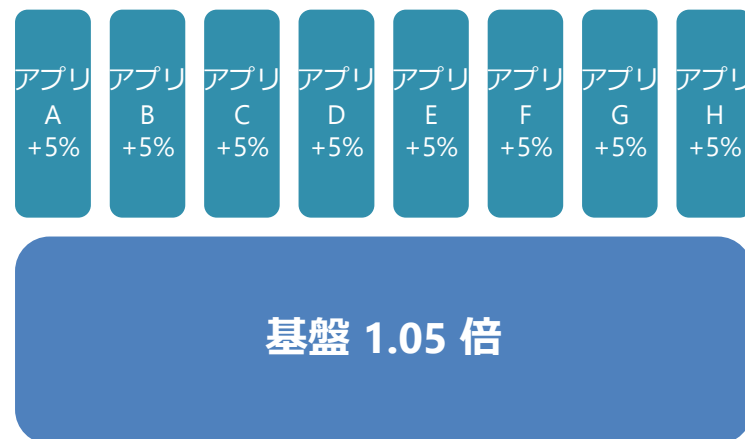
■ 応用系

- 効果は特定アプリケーションのみだが、改善量が大きい
- たとえば、あるアプリケーションを数倍～1桁速く出来れば凄い



■ 基盤系

- 効果は広く薄く
- 色々なアプリケーション一般で平均 5% 高速化できれば偉大な成果



余談：応用系と基盤系

- 応用系や基盤系の中にも，応用よりや基盤よりの部分がある
 - たとえば特定分野のアプリの中の基盤的な部分を作るとか
- 各人ごとに好みや相性がある
 - 研究室配属や将来やりたいことを考える時に意識してみると良いかも

もくじ

1. コンピュータ・アーキテクチャとは？
2. コンピュータの種類
3. ソフトウェアとの関係

コンピュータの種類

■ 類型

- 組み込みコンピュータ
- 車載コンピュータ
- スマホ, タブレット
- ゲーム機
- パソコン
- サーバー
- スーパーコンピュータ

■ 備考

- 典型的には下に行くほど強力かつ大きくて, 値段が高い
- これは学術的なきちんとした分類ではあまりない
 - なんかこう言う分類が一般にされているかもぐらい

組み込みコンピュータ

- 色々な機器に組み込まれて使用される小さいコンピュータ
 - それらの機器を制御するために使われる
- 身のまわりのありとあらゆる機器に組み込まれている
 - 冷蔵庫, 炊飯器, エアコン, 洗濯機, お掃除ロボット, 食洗機
 - 時計, 体重計, 体温計, プリンタ, シュレッダー
 - ヘッドセット, 空気清浄機, 加湿器 …
- 性能や記憶容量はかなり低い, 安い
 - 数十円～数百円ぐらい

車載コンピュータ

- 基本的には、組み込みコンピュータの一種
 - エンジンなどの制御は全部コンピュータで行われている
 - 1台の車には結構な数の小さなコンピュータがのっている
- 自動運転の処理はもっと高い性能が要求される
 - 自動運転の実現はかなり高度な計算が必要
 - 「スーパーコンピュータを背負って走っている」と言われることもあるぐらい

スマホ, タブレット

- スマホやタブレットは, いま最も高性能なコンピュータの1つ
 - いわゆるガラケーは組み込みコンピュータの一種だった
 - スマホでは高度なアプリを動かすためにどんどん高性能化していった
 - WEB ブラウザの高度化も影響
- 消費電力の縛りが大きいため, あえてゆっくり動かされている事もある
 - バッテリー駆動される
 - 冷却に制約がある (冷却ファンとか使えない)
- お値段は数万円から最近では10万円を超えることも

ゲーム機

- ゲーム機もかなり高性能な方
 - ゲーム機も大昔は組み込みコンピュータの一種だった
 - どんどん高性能化していった
- パソコンやスマホと基本的な構造はあまり変わらない
 - PS5, XBOX ONE は PC と, Switch は スマホとかなり似てる
- お値段は数万円ぐらい

パソコン, サーバー

- 最も高性能なコンピュータの1つ
- スマホやタブレットと比べると電力の縛りがかなり薄い
 - コンセントから電源が供給できる
 - 冷却ファンが使える
- パソコンとサーバーはほぼ同じ構造
 - アーキテクチャはまったく同じと言って良いほど
- お値段は
 - パソコンなら数万から数十万円ぐらい
 - サーバーは数百万円するものもある

スーパーコンピュータ

- スーパーコンピュータはパソコンやサーバー、スマホの上位互換ではない
 - それぞれ目的とする「速さ」が実は違う
 - パソコンやサーバーとはアーキテクチャがかなり違うことが多い

目的とする「速さ」の違い

■ スーパーコンピュータ

- いかにより多くの計算量をさばくかを重視
 - 科学技術計算（気象のシミュレーションとか）が主な対象
 - 地球を細かく区切ってシミュレーションとか
- 車で言うとダンプカーみたいなもの
 - 時間あたりで大量の物資を運べるが、速度は速いわけではない

■ パソコンやスマホ

- 量は多くないが、いかに速く計算を終わらせる事を重視
 - ブラウザとかのアプリケーションが対象
 - タップしたらすぐに反応するようにとか
- 車でいうとレース用車のようなもの
 - 速度は速いが、多くの量を運べるわけではない

スーパーコンピュータ

- 典型的には数千台とかのコンピュータを大量に束ねたものになっている
 - 電力を供給するために専用の変電所を建てたりすることもある
- お値段は桁違いに高い
 - 富岳の場合1300 億円
 - （以下は、富岳が設置されている神戸の理研に行った時の写真



余談：計算科学センター駅

- 2021年に「京コンピュータ前駅」から「計算科学センター駅」に変更
 - 京はここに設置されていた先代のスーパーコンピュータ
 - 神戸のポートアイランドにある
 - 三ノ宮からは（がんばれば）歩いて行ける



コンピュータの種類とアーキテクチャ

- さまざまなコンピュータが存在
 - 値段も数十円から数千億円まで
 - それぞれの応用に合わせたアーキテクチャがある
- 色々なコンピュータで何がどう違うのかを理解するのもこの講義の目的
- みんな基本的には「ノイマン型アーキテクチャ」に従う
 - John von Neumann (1903年 - 1957年)
 - (写真は Wikipedia より)
 - 次回以降で説明



もくじ

1. コンピュータ・アーキテクチャとは？
2. コンピュータの種類
3. ソフトウェアとの関係

なんでアーキテクチャを学ぶのか？

■ ありえる疑問：

- 将来ソフトを書くことがあっても、別にハードは作らないのでは？
- 別にハードの事を理解していなくても問題ないのでは？

ソフトについて、この講義でわかること

- ソフトはどのようにハード上で実行されるのか？
 - C や Python など書かれたソフトは、
どのように翻訳されて実行されるのか
- ソフトとハードのインターフェースとは何か？
 - どのような理由でどう階層が切られているのか
 - ソフトはハードにどのような指示を出しているのか
- ソフトの性能や特性はどのように決まるのか？
 - 速いとか遅いとかは、どのようにって決まるのか
 - 高いとか安いとか、電気を食うとか食わないとか…

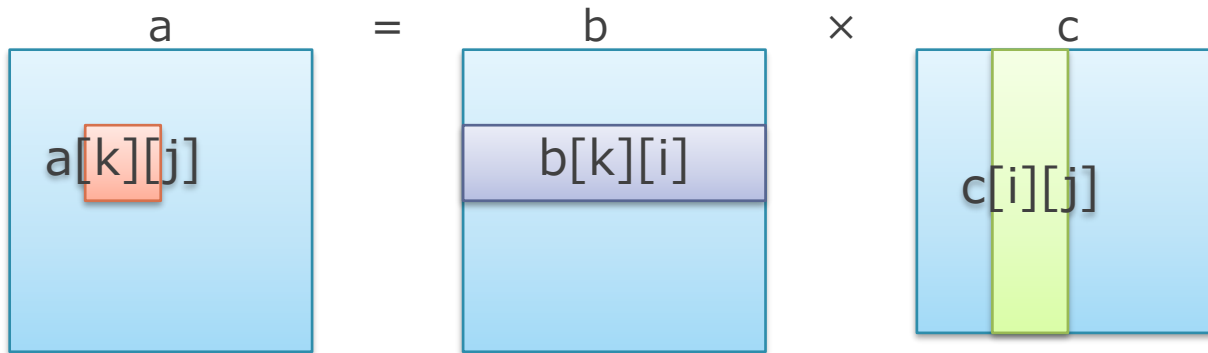
例：行列積の処理時間について考える

```
for (int k = 0; k < SIZE; k++) {  
    for (int j = 0; j < SIZE; j++) {  
        for (int i = 0; i < SIZE; i++) {  
            a[k][j] += b[k][i] * c[i][j];  
        }  
    }  
}
```

- 三重ループとして実現できる
 - このプログラムの性能についてしてみる

行列積のプログラムの構造

(ここは今日は別にきちんと理解しないでも良い)



```
for (int k = 0; k < SIZE; k++) {  
    for (int j = 0; j < SIZE; j++) {  
        for (int i = 0; i < SIZE; i++) {  
            a[k][j] += b[k][i] * c[i][j];  
        }  
    }  
}
```

- $a[k][j]$ は, b の k 行目 (紫) と, c の j 列目 (緑) の各要素を乗算して累積することにより求まる
 - 一番内側の i はこの各要素を参照するために回る

重要なポイント

```
for (int k = 0; k < SIZE; k++) {  
    for (int j = 0; j < SIZE; j++) {  
        for (int i = 0; i < SIZE; i++) {  
            a[k][j] += b[k][i] * c[i][j];  
        }  
    }  
}
```

- (とりあえず三重ループでひたすら乗算と加算をしている事がわかってればよい
- このプログラムをよく見ると,
 - `a[k][j] +=` の部分の加算の順序は自由に入れ替え可能
 - 加算はどのような順序でやってもよい
 - **なので, ループの外側と内側を入れ替えても, 結果は同じ**

行列積の実行時間の差

// 1100 秒かかる

```
for (int i = 0; i < SIZE; i++) {  
    for (int j = 0; j < SIZE; j++) {  
        for (int k = 0; k < SIZE; k++) {  
            a[k][j] += b[k][i] * c[i][j];  
        }  
    }  
}
```

// 20 秒で終わる

```
for (int i = 0; i < SIZE; i++) {  
    for (int k = 0; k < SIZE; k++) {  
        for (int j = 0; j < SIZE; j++) {  
            a[k][j] += b[k][i] * c[i][j];  
        }  
    }  
}
```

- 上記のプログラムは、ループの順序を入れ替えただけ
 - したがって、左右で結果は変わらない
- しかし、実行時間は大きく異なる
 - 1100 秒 vs. 20 秒
 - この理由を知るためにはアーキテクチャの知識が必要

なんでアーキテクチャを学ぶのか？

- ハードを作る人，いじる人にはそりゃ当然に重要
- ソフトしか作らない人にも重要
 - 処理速度やコストの問題は常につきまとう
 - アプリのバグの原因は，その下にあるハードや OS，コンパイラの仕組みを理解してないとわからい事が多い
 - これらの事を理解するためにはハードや OS などの理解が必要
 - 一流になるか二流になるかの違いになりえる

まとめ

■ 今日の内容

- コンピュータ・アーキテクチャとは？
- コンピュータの種類
- ソフトウェアとの関係

■ 正直いろいろ手探りで

- 色々と試行錯誤しながら進めると思います
- よろしくをお願いします

- 感想や質問を投稿してください
 - Moodle の「感想や質問」のところからお願いします
 - 締め切り：来週の講義開始時まで