では、授業の本題に入ります

- まず、教科書の p.1～8（1章始め～1.1.4項）をよく読んでください。
  - ハードウェアとソフトウェアの機能分担
  - 「コンピュータアーキテクチャ」とは？

  がテーマです。

- 読み終えたら、次のページへ。
  - 教科書の内容を補足します。

ハードウェアとソフトウェア

- コンピュータはハードウェアとソフトウェアでできています。
  - 皆さん（TVゲーム世代）にとっては、ハードウェアとソフトウェアの違いなど分かり切っていますね。
  - ですから、その「違い」については説明しません。
    - 因みに、昔の人にとっては「ソフトウェア」は難解でしたので、かなり説明が必要でした。昔は、機械（装置）と言えば、ハードウェアだけでできていましたから。
  - しかし、そのような皆さんでも、ハードウェアとソフトウェアの「境界」が定まったものではないことは、意識していますか？

- つまり、どれだけのことをハードウェアで行うのか、また、どれだけのことをソフトウェアで行うのか、が決まっている訳ではない、ということです。
- 従って、コンピュータシステムを設計する場合には、そこから考え始めなければなりません。
  - その「境界」を定めた結果が、トレードオフ点ということになります。

- では、何によってそのトレードオフ点を決めるのでしょうか？

- それが教科書の図1.2(p.3)になります。
  - つまり、どれだけの範囲の仕事(専用性・汎用性の度合い)を、どれくらいの性能(高速性の度合い)で行いたいのか、によります。

    - 何でもできるようにしようとすると、大部分をソフトウェアで実現する(プログラムを書き換えることによって異なる機能が実現できるので)ことになり、ハードウェアで専用化するほどの性能は得られなくなります。

    - ソフトウェアで行うようなことをハードウェアで専用化すると、高速化できます。しかし、それしかできなくなり、用途が狭まります。

命令セット

後の講義で詳しく説明しますが、

- コンピュータハードウェアは「命令（instruction）」によって動作します。加算命令や乗算命令などの種類があり、これらを用いてコンピュータハードウェアに何をするべきかを指示します。
- この命令を複数個の0/1を組み合わせて表現したものが「命令語（instruction word）」です。
- したがって、プログラムとは、この命令語の集まりであり、コンピュータハードウェアはそれらの命令語を順番に実行します。

- すべての命令を定義したものを「命令セット（instruction set）」と呼びます。
  - このコンピュータハードウェアではどのような命令が使えるのか、したがって、どのような操作（演算）が行えるのか、を示すものになります。
  - プログラマ（ソフトウェア側）から見れば、命令として定義されていない操作（演算）は、そのコンピュータハードウェアに行わせることはできません。
  - つまり、このコンピュータハードウェアで何ができるのか（またはできないのか）、を示すものとも言えます。

- 命令を用いてプログラムを表現するために、コンピュータが理解できる言語という意味で、「マシン語/機械語（machine language）」といいます。
  - 「マシン命令」と日本語で言うときは曖昧です。machine languageの意味でも、instructionの意味でも、また、machine code（意味は命令語と同じ）の意味でも、用います。

- 一方、コンピュータハードウェアの設計者は、命令セット中の全ての命令を実行できるように、ハードウェアを設計・実装しなければなりません。

- したがって、マシン語が、ハードウェアとソフトウェアの境界となります。

- また、命令セットがコンピュータの種類を決めることになります。
  - 同じ命令セットのコンピュータでなら、同じマシン語のプログラムが実行できます。
  - 異なる命令セットのコンピュータの間ではそうはいきません。プログラムが、元々、C言語などのプログラミング言語（高級言語）で記述されているのなら、コンパイルし直さなければなりません。

ハードウェアとソフトウェア（続き）

- ハードウェアとソフトウェアのトレードオフを、教科書の図1.3（p.5）のセマンティックギャップの観点で考えてみましょう。
  - ユーザから見れば、解きたい問題が解ければそれで良いのですが、コンピュータシステムの設計者の立場では、その問題に対してハードウェアとソフトウェアの機能分担を考えなければなりません。

- マシン命令（instruction）の機能を高機能にすると、マシン語の境界線が上に上がります。
  - 高機能な命令を実行できる「複雑な」ハードウェアを設計しなければなりません。（要するに、ハードウェアで頑張る。）
  - その命令を実行すると、低機能な複数のマシン語で実行するよりも性能は上がるでしょう。
  - ハードウェアが高機能になった分、プログラム中の命令数は減るはずです。（ハードウェアで頑張った分、ソフトウェアは楽になる。）
  - しかし、その高性能な命令は、実際に使われる頻度は多くはないでしょう。つまり、何にでも使えるというわけではなく、専用性の度合いが高まることになります。
- マシン命令の機能を低機能にすると、上記の逆になります。

- 実際、世の中には何種類かのコンピュータが存在します（売られています）。
  - 同じ種類の中では、製品価格に応じて性能の序列もはっきりしています。
- そして、どの種類が一番優れているか、がはっきりと決着がつきません。
- どうしてこんな状況になっているのでしょう？
  - 一種類に決まっていると、使う方も楽なのですが。。。
- それはトレードオフ点が異なっているからです。
  - ある目的にはコンピュータAが最も優れていて、別の目的にはコンピュータBが最も優れていて、コンピュータCはどの目的でも1番にはなれないけれど、平均的には最も優れている、というような状況が起こり得るのです。どれか1つには決められませんね。

- ですから、コンピュータシステムを設計する際には、そのトレードオフ点を定めることが重要で、また、その解も1つとは限らないのです。
  - コンピュータシステムの設計においては、ハードウェアとソフトウェアのトレードオフだけでなく、より細かくは、もっと多くのいろいろなトレードオフを考慮する必要があり、そのそれぞれについてトレードオフ点を適切に決定しなければなりません。

コンピュータアーキテクチャ

- コンピュータを建造物のように見立てて、それをどう設計するか、ということで「コンピュータアーキテクチャ（Computer Architecture）」という言葉が使われ始めました。

- この言葉は、IBMの大型コンピュータ「System 360」を設計する際に、その設計者であるアムダール（Amdahl）氏が最初に用いました。

- その意味（定義）は以下の通りです。

> 物理的機構（ハードウェア実現方式）とは無関係に、プログラマからコンピュータハードウェアを見たときの論理的な仕様

- ちょっと何を言っているのかよくわかりませんね。
- でも、これは、当時としては画期的な、重要な考えを表すためのものなのです。
  - 今では当たり前になっていますが。
- だからこそ、わざわざ新しい言葉を作って表現したのです。

- 当時は、同じメーカーでも、異なる種類のコンピュータを複数製造していました。
  - 性能の違いはハードウェアの違い、ハードウェアが違うということは、命令セットも違う、というわけです。
    - 性能の違いは価格の違いでもあります。
    - 新発売のコンピュータは性能も高いが、命令セットも違う。
  - 従って、3～4年経ってコンピュータを買い換えたら、それまでのソフトはすべて使えない、それが嫌なら、性能も陳腐化した古いコンピュータをいつまでも使い続ける、ということになるのです。
    - これが皆さんのパソコンや家庭用ゲーム機で起こっていたら、我慢できますか？　新しいゲームソフトで遊びたかったら、ゲーム機本体も買い換えなさい、という状況が頻繁に（例えば、毎年、とか）発生したらたまりませんよね。

- 「そんなことはもう止めにしよう！」というのがIBM System 360を設計するときの基本的な考え方の1つだったのです。
  - 「プログラマからコンピュータハードウェアを見たときの論理的な仕様」というのは、簡単に言えば命令セットのことを指します。
  - それが、「物理的機構（ハードウェア実現方式）とは無関係」だというのです。
  - つまり、性能の差はハードウェアの構成方式によって生じるのですが、安価で低性能なコンピュータも、高価で高性能なコンピュータも、命令セットは同じにしましょう、と決めたのです。
    - このことを「ファミリ」という概念で提唱しました。
    - これにより、コンピュータを買い替えてもそれまでのソフトがそのまま使用できるようにもなりました。
      - 逆に、命令セットを設計する際には、長い将来に渡って使い続けられることを念頭において設計しなければならなくなりました。

- しかし、ここでちょっと待ってください。こんな説明で納得してはいけません。もう一度考え直してみましょう。
  - 命令セットはハードウェア構成と密接に関連します。
  - 性能はハードウェア構成がもとで決まります。
  - それならば、性能が異なるのに命令セットが同じ、ということなどあり得ない話です。

  - だからこそ、まず、命令セットとハードウェア構成との関係を断ち切る必要がありました。
    - 定義では、「物理的機構（ハードウェア実現方式）とは無関係に」と言っています。
  - そして、その断ち切った関係を穴埋めする（つまり、安価なハードウェアでも高価なハードウェアでも同じ命令セットを実現できるようにするための）技術が新たに必要となったのです。
    - これがコンピュータの仮想化の始まりでもありました。
    - 詳しくは、後の講義で解説します。

- ということで、その頃は「どんな命令セットにするか」がコンピュータアーキテクチャの主要な問題でした。
  - 現在は、それを命令セットアーキテクチャ（ISA: Instruction Set Architecture）と呼んでいます。
- 現在は、もっと広い意味で、コンピュータを設計する際の設計思想を指す言葉として用いられています。
  - 当時も、ISAが主要問題でしたが、その元は、ファミリの概念（思想）でしたし。。。
  - また、現在ではソフトウェアの分野でも、その設計思想という意味で「アーキテクチャ」という言葉が使用されています。

教科書の1.2節（p.8～28）は、軽く目を通しておいてください。

- 高校までの歴史の科目のような捉え方で勉強する必要はありませんが、技術的な進化の「内容」を押さえておくことは重要です（本質的な「内容」以外の部分はどうでもよいでしょう）。
- 現在でも普通に使用される用語/術語がたくさん出てきますので、覚えておいてください。