

機械語序論

第1回

計算機の仕組みと機械語

佐藤

機械語序論

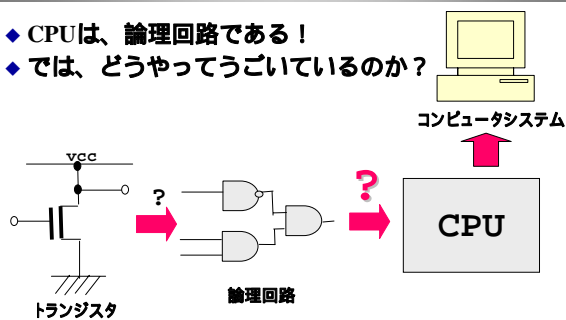
この講義の目的

- ◆ この講義の目的は、機械語すなわちアセンブリ言語のプログラミングを通じて、コンピュータについての基礎的な仕組みを理解することです。
 - アセンブリ言語をすなわち機械語を直接プログラミングする機会はそれほど多くはありませんが、コンピュータがどのように動作しているかについて理解することは情報科学、コンピュータ科学を学ぶ上でもっとも基本的な事柄の一つです。
- ◆ 4ビットマイコンを理解しよう。

機械語序論

CPUと論理回路

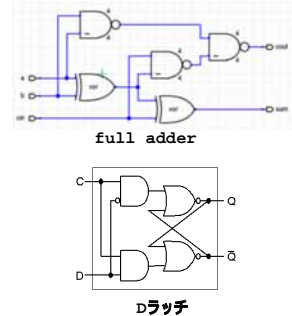
- ◆ CPUは、論理回路である！
- ◆ では、どうやってうごいているのか？



機械語序論

論理回路

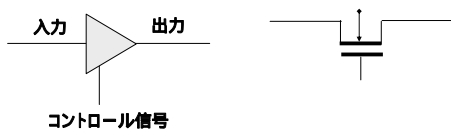
- ◆ 組み合わせ回路
 - 入力により、出力が決まる
 - 論理素子AND, OR, NOTの組み合わせ
 - 加算器
- ◆ 順序回路
 - 現在の出力が過去の入力の状態によって決まる回路
 - フリップフロップ
 - レジスタ、カウンタ、...
 - メモリ（電荷素子）



機械語序論

論理回路

- ◆ バススイッチ（ゲート）
 - コントロール信号によって、出力を遮断する。



機械語序論

論理回路とコンピュータ

- ◆ コンピュータは、0と1の2進数で動作している。
- ◆ 0と1とは、電圧が高い、低い、あるいは、メモリでは電気がたまっている、たまっていない、といった2つの物理的な状態で表現されている。
- ◆ 例：加算器

コンピュータの基本的な構成

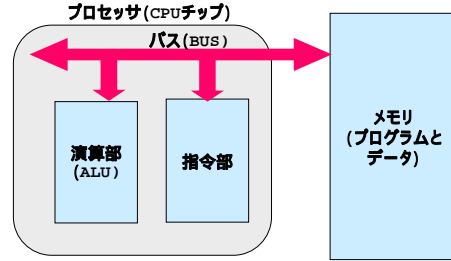
- ◆ コンピュータのもっとも基本的な要素は、メモリとプロセッサ (CPU) である。
 - メモリはプログラムやデータを格納する場所
 - プロセッサはそのメモリからプログラムやデータを読み出して、プログラムを実行しています。
 - ・ 指令する部分：プログラムを解釈 (?) して指令する
 - ・ 演算する部分：足し算や掛け算をする部分
- ◆ プログラムとデータをメモリに置いて、プロセッサがメモリから読み出して実行する方式を、ストアードプログラム方式という。

現在のコンピュータのもっとも重要な基本的な概念

プログラムを実行するプログラムがつくれる システム

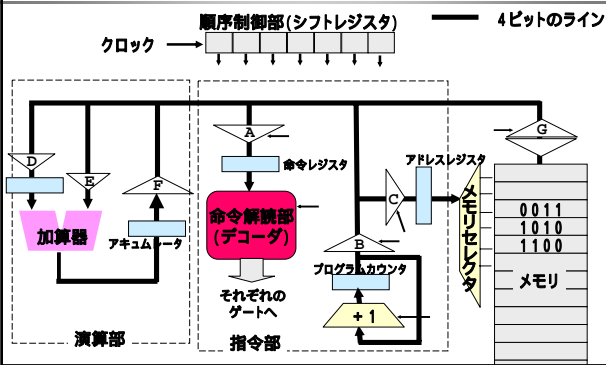
コンピュータの基本的な構成

- ◆ 記憶 (メモリ)、指令、演算



簡単な4ビットプロセッサ

レジスタ (rectangle), ゲート (triangle)



説明

- ◆ 命令やデータのとおり道 (実際は電線) を「バス」という。
 - データのとおり道はデータバス、
 - メモリにどのデータを読み出すかを伝えるバスをアドレスバス
- ◆ メモリの横にある「番入」(セレクトという)に接続されているのがアドレスバスで、操作するメモリを指定してしま三角で止められているのがゲート。信号の流れを制御する。
- ◆ このゲートを制御する信号は、現在の命令 (0と1の組み合わせ) から、命令解釈部 (デコード) で作られる
- ◆ プロセッサの中にも一時的にデータを格納するメモリ (のようなもの) がある。レジスタと呼ばれる。そのいくつかはプログラムからは見えない (例えば、現在の命令を保持している命令レジスタや読み出すメモリの番地を保持しているアドレスレジスタなど)
- ◆ 実行するプログラムの番地を保持しているレジスタをプログラムカウンタという
- ◆ 演算の一時的な結果を保持するレジスタをアキュムレータと呼ぶことがある。実際のプロセッサではこのようなレジスタが複数ある。

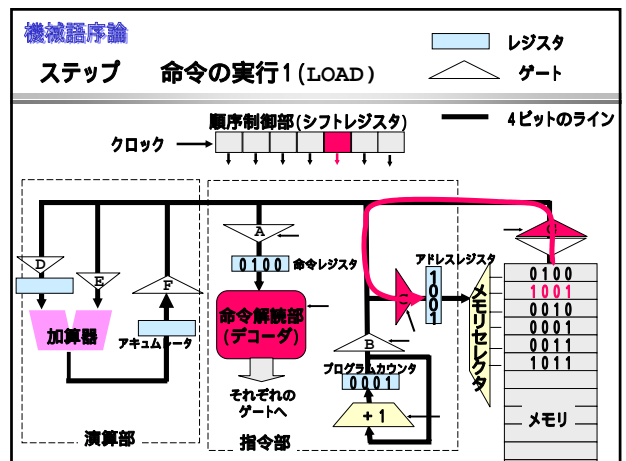
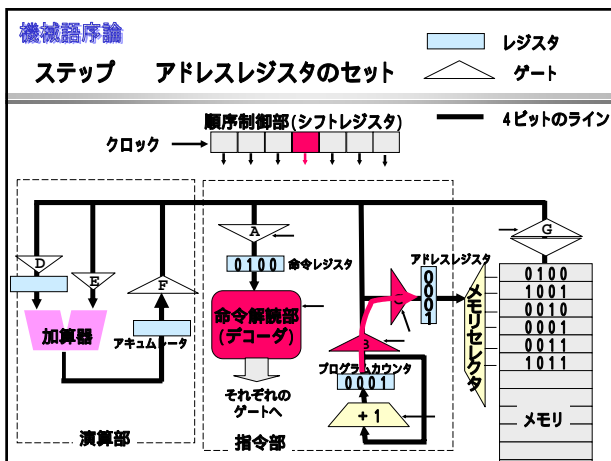
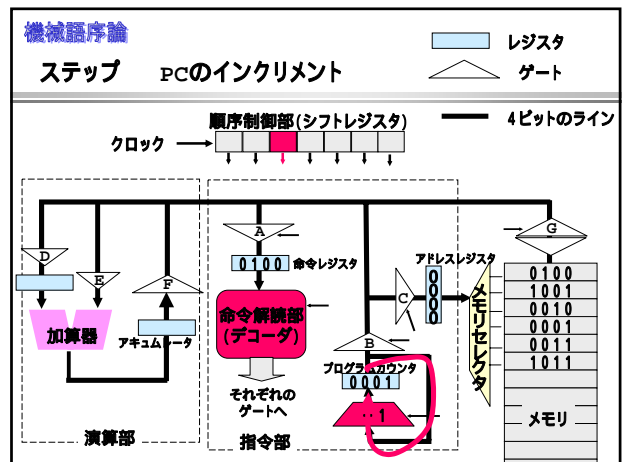
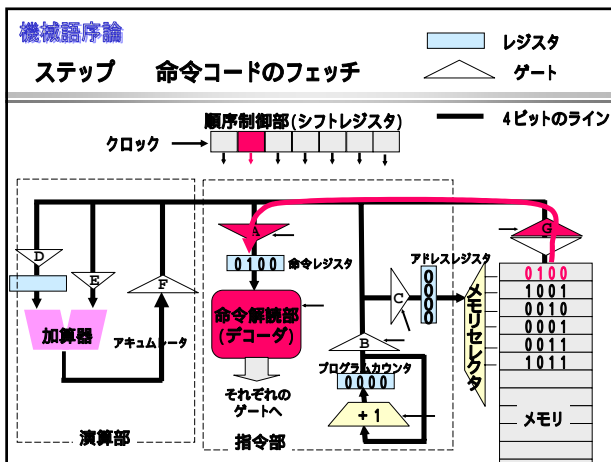
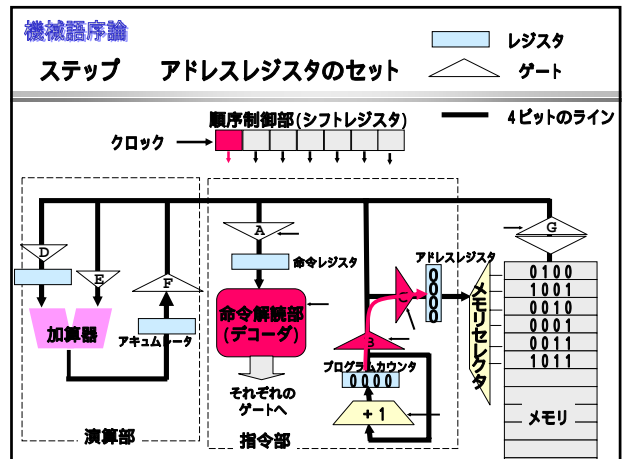
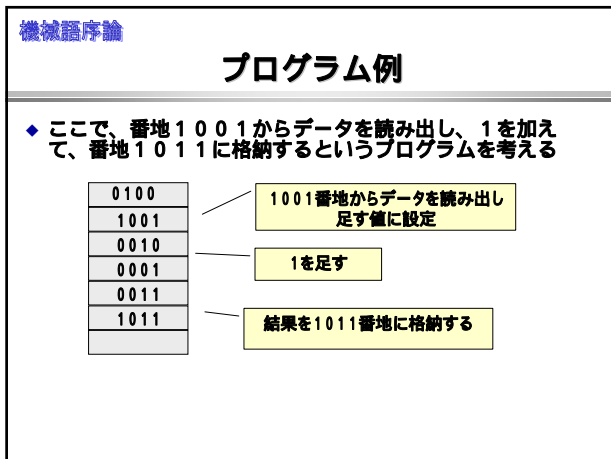
命令コード (機械語)

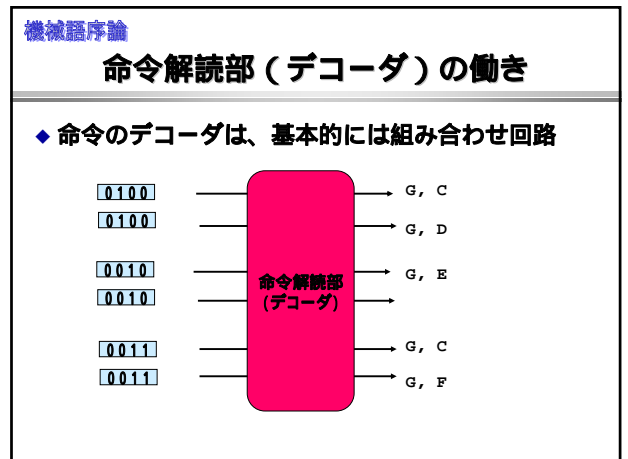
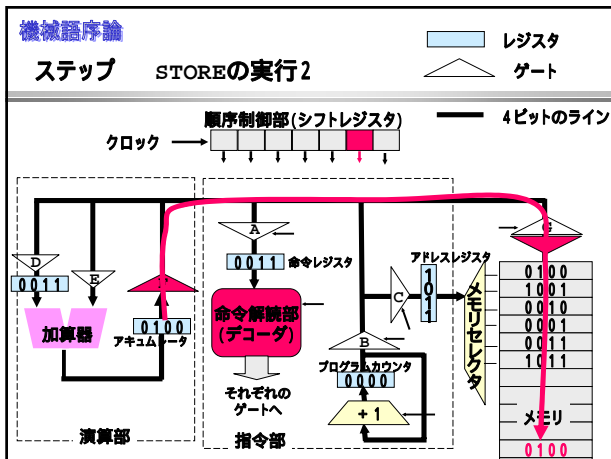
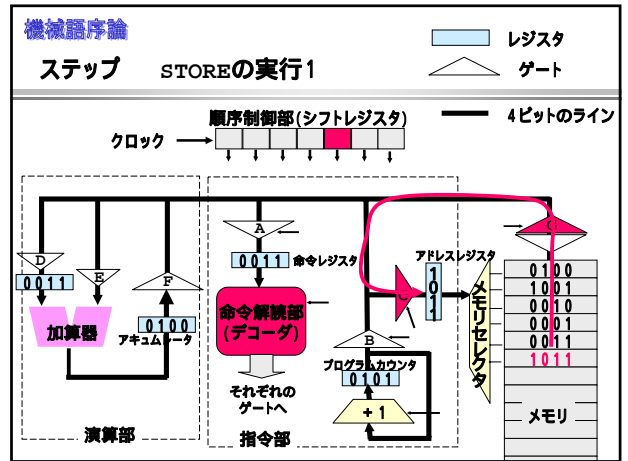
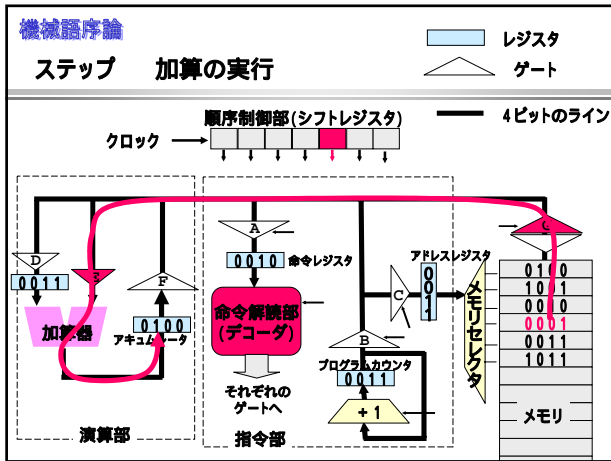
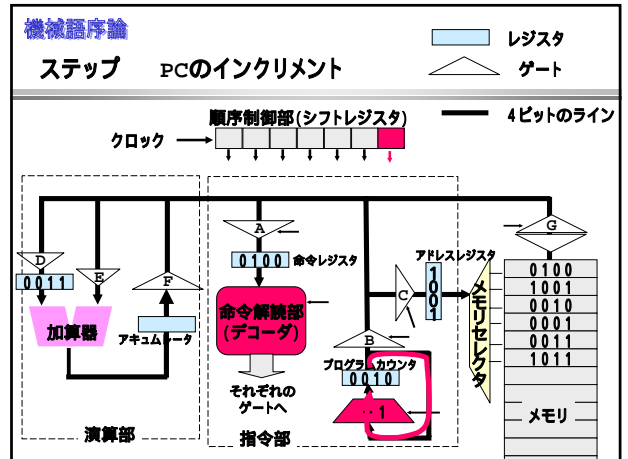
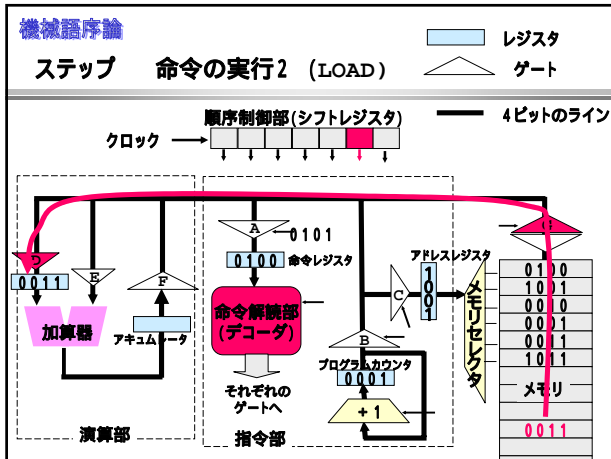
- ◆ メモリ上にあるプログラムのそれぞれの命令は、動作とその対象からなる。
 - 動作を指定するのが、命令コード (オペコード)
 - 対象をオペランドという
 - 実際のマシンではオペランドのない命令もある
- ◆ このマシンでは、2ワードであらわす

0001	X	xを足される数に設定しなさい (LOAD)
0010	Y	設定されている足される数に yを足しなさい (ADD)
0011	Z	zのアドレスに結果を入れなさい (STORE)
0100	W	w番地の中身を、足される数に設定しなさい (LOAD)

どのように実行されるか

- ◆ クロック信号が入力されると、順序制御部から から までの信号が順番に送られる
 - プログラムカウンタをアドレスレジスタに設定
 - 命令コードのフェッチ
 - プログラムカウンタを1つあげる (オペランドを読む準備)
 - プログラムカウンタをアドレスレジスタに設定
 - 命令コードの解釈と実行1
 - 命令コードの解釈と実行2
 - プログラムカウンタを1つあげる (次の命令を読む準備)
- ◆ これが、コンピュータの速度を決定する
- ◆ この単純なプロセッサでは、以外と同じパターンでゲートを開け閉めするが、だけは特定の命令から、デコード部で生成された信号でゲートを開け閉めする
 - このデコード部は組み合わせ回路である!





機械語序論

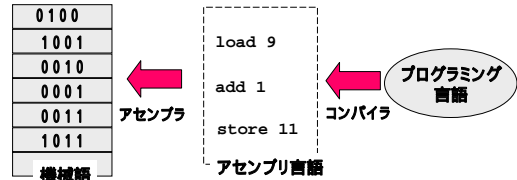
本当のマイクロプロセッサでは...

- ◆ 制御を変えるにはどうすればいい? (jump命令、条件分岐)
- ◆ 実際には、各ステップで同時に実行できるところがある。
 - たとえば、実行とプログラムカウンタのインクリメントを同時に実行
- ◆ プロセッサの実行フェーズ
 - 命令フェッチ (IF: instruction fetch)
 - 命令デコード (ID: instruction decode)
 - 命令実行 (EX: execute)
 - 結果の書き込み(WB: write back)
 - ...
- ◆ レジスタがたくさんある。
 - 汎用レジスタ(整数とアドレス)
 - 浮動小数点レジスタ
- ◆ キャッシュメモリがある。
- ◆ オペレーティングシステムのいろいろな機能
 - 仮想記憶、割り込み、入出力

機械語序論

アセンブリ言語とコンパイラ

- ◆ マシン語をそのものでは01のパターン、つまり数字ですので、これをつかってプログラミングするのは人間にとって非常に面倒な作業になる
- ◆ 基本的に、マシン語に1対1に対応するように記号を使って表記したのがアセンブリ言語
- ◆ オPCODEを表す記号をニーモニック という
- ◆ コンパイラは、プログラミング言語をアセンブラ言語に翻訳する。



機械語序論

- ◆ アセンブリ言語で記述されたプログラムは、アセンブラによって、機械語に翻訳(変換)されます。
- ◆ アセンブリ言語は基本的に機械語と1対1なので、アセンブリ言語で書かれたプログラムは機械語で書かれたプログラムとは同じものとして考えることができます。
- ◆ Cコンパイラでは、-S オプションをつけてコンパイルすると、.sというファイルができるはずですが、どんなコードができていのかをみてください。
 - % cc -S t.c

機械語序論

x86 (IA32) プロセッサ

- ◆ この講義では、インテルのx86ファミリーと呼ばれるプロセッサを題材に進めていきます。
 - このプロセッサは学類の計算機で使われているプロセッサであり、普通のPCに使われているプロセッサでもあります。
 - x86プロセッサはいろいろな種類がありましたが、この講義では80586移行のいわゆるPentiumプロセッサファミリーを対象とします。

機械語序論

x86 (IA32) プロセッサの特徴

- ◆ CISC (Complex instruction set computer) であるが、内部的に簡単な命令に分解して実行する機能を持ち、非常に高速化されている。
- ◆ 論理的なアドレス空間は32ビット(最近64)
- ◆ レジスタは、PC(プログラムカウンタ)のほか、32ビットの汎用レジスタとして、eax, ebx, ecx, edx, esi, edi, esp, ebpの8個のレジスタがある。このうち、espは、スタックポインタ、ebpはベースレジスタと名づけられ、ソフトウェア的につかい方が決まっている。
- ◆ メモリアクセスする場合には、豊富なメモリアクセスモードが使える。
- ◆ 浮動小数点レジスタは8個で、スタック状に使う。

機械語序論

- ◆ このプロセッサは歴史的な経緯を引きずっており、非常に複雑な命令セットになっていますが、この講義では、必要な部分のみを使ってプログラミングすることになります。
- ◆ 詳しく知りたい方は、Intelから出されている「Pentium ファミリー デベロッパーズマニュアル(下)アーキテクチャとプログラミングマニュアル」などを参照してください。

次回

- ◆ アセンブリ言語のプログラミング環境について解説します。
 - アセンブリ言語のプログラムの実行の確認やデバックのために、gnu debugger、gdbを使いますので、その使い方についても解説します。