CPUのしくみ、CPU命令とC言語プログラムの関係

～CPUシミュレータを用いた説明～

名古屋大学　浅井徹

コンピュータの構成要素

　CPU、メモリ、入出力装置

　バスで接続されている

プログラム、データはメモリ上に置かれる

（Linux, Windows などのOSが配置する）

プログラム、データはバスを介してやりとりされる

　→　サンプルを実行

CPU：制御装置、レジスタ、演算装置

メモリから命令を読み、対応する動作をする

　プログラムカウンタ（PC）が指す番地から読む

　命令を読むと原則としてPCは一つ進む

このシミュレータはプログラムがなくなると止まる

（本物は際限なく命令を実行する）

CPUが実行できる命令の種類・内容（命令セット）は限られていて、変化しない

（CPUの開発では、まず命令セットを設計し、その後その動作を実現する回路を設計する）

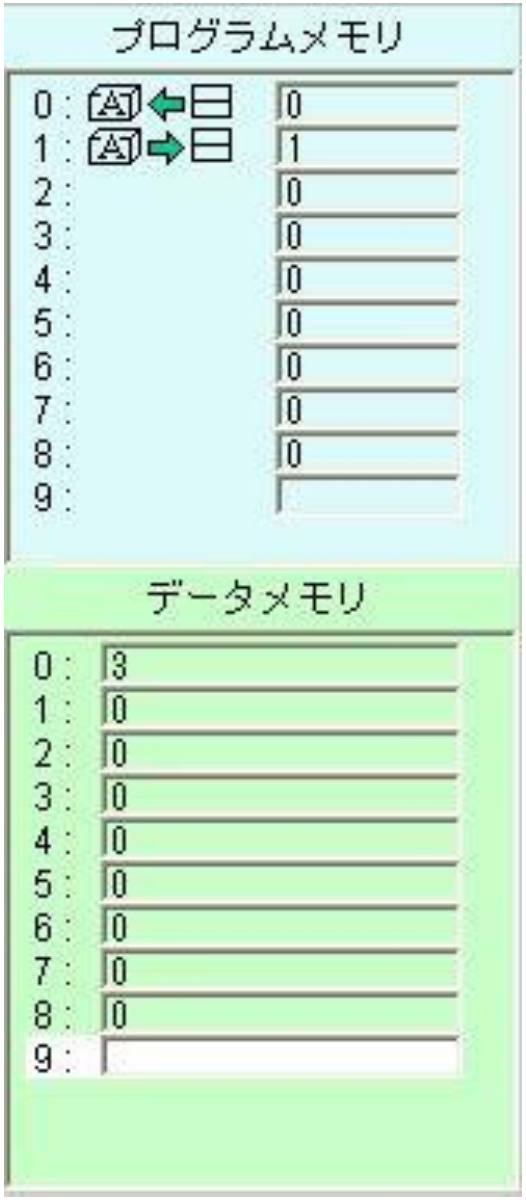
基本的にはデータの移動、演算、ジャンプ

1. 各命令の説明
2. 足し算プログラム
3. 条件つき繰り返しプログラム
4. 関数呼び出しの原理

データ転送命令

メモリ→レジスタ

レジスタ→メモリ



演算命令

和 （A＋B→A）

差 （A－B→A）

演算はレジスタに入っている
データに対して行われ、演算
結果はレジスタに格納される

出力命令

メモリ→出力ポート

（データ転送の一種）

例えば画面への表示はビデオデバイスが接続されているポートにデータを出力することで行われる

ジャンプ命令

　Ａ＞０のときジャンプ

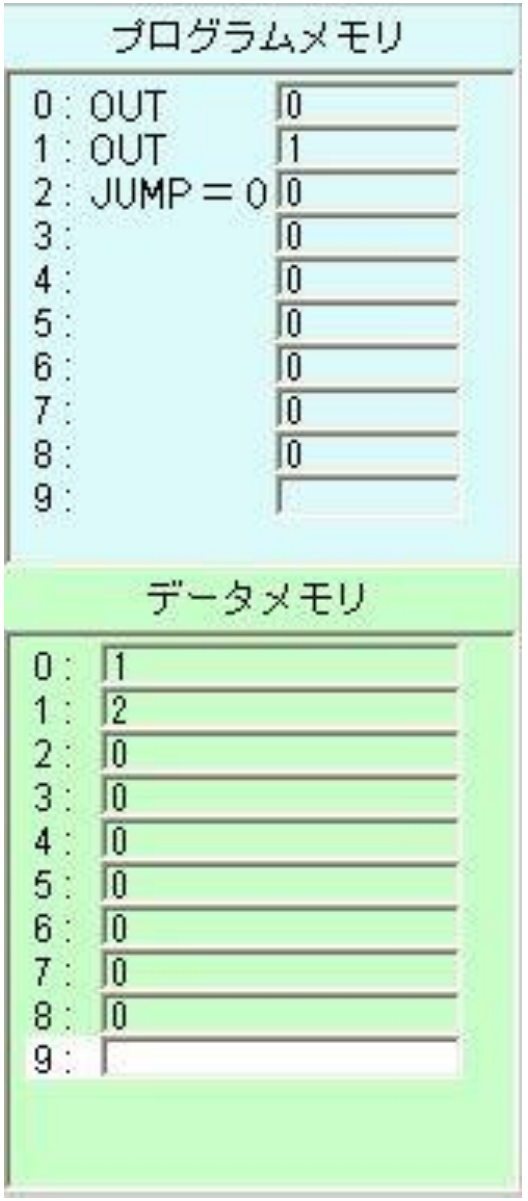
　Ａ＝０のときジャンプ

条件が成立すれば

　PCを指定した値に変更する

成立しなければ

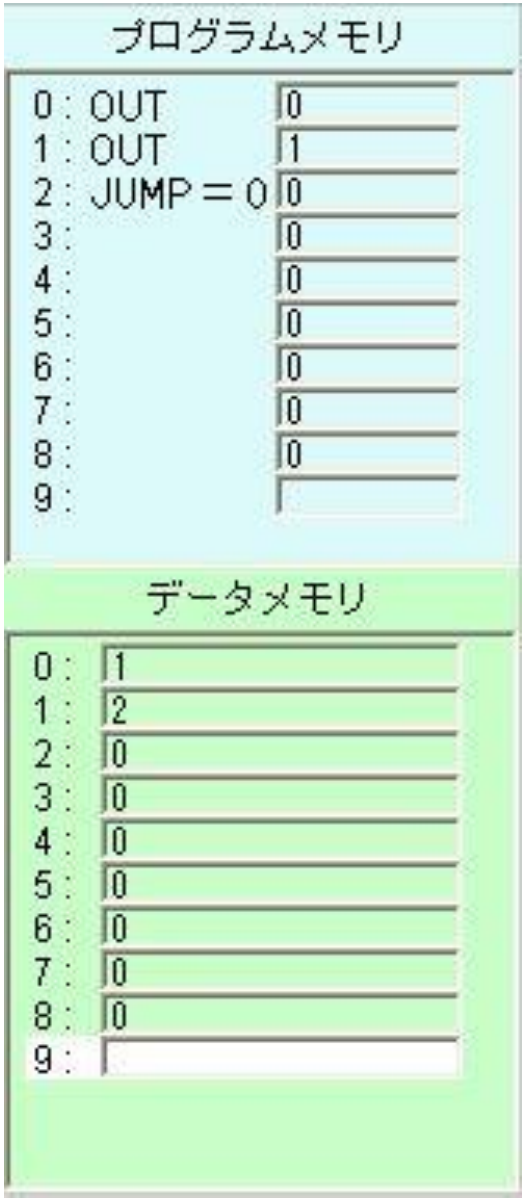
　何もしない



類似したCプログラム

```
int x=1, y=2;
while(1) {
  printf("%d",x);
  printf("%d",y);
}
```

繰り返しはジャンプを使って
実現される



命令セット

基本的にはデータの移動、演算、ジャンプ

このシミュレータはかなり簡略化したもの

実際のCPUでも基本的な枠組みはほぼ同じ

例：H8マイコン

1. 各命令の説明
2. 足し算プログラム
3. 条件つき繰り返しプログラム
4. 関数呼び出しの原理

足し算プログラム

メモリ0とメモリ1の和を

メモリ2に格納し、さらに

ポートにも出力する

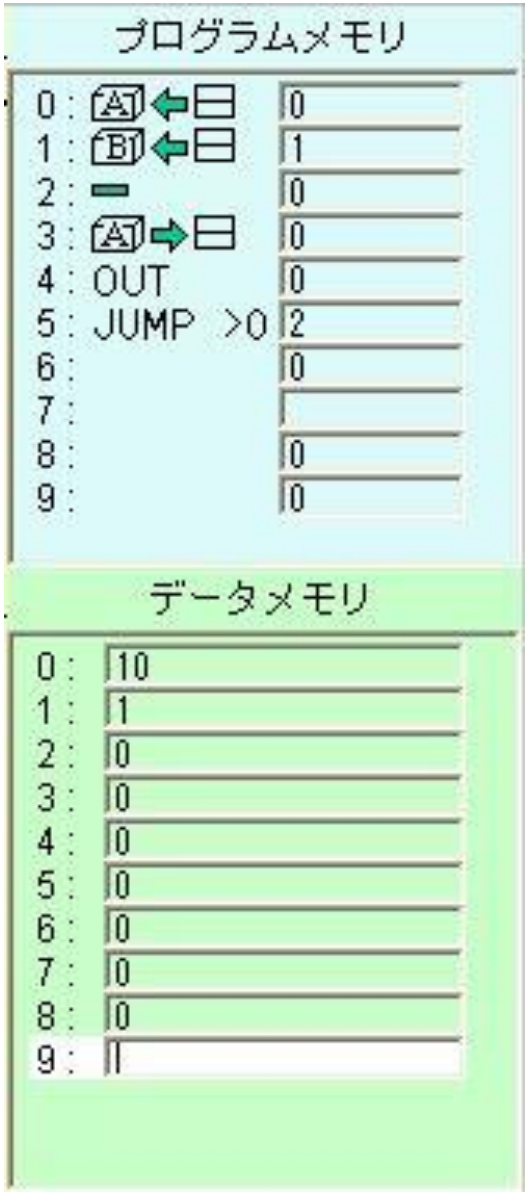
類似したCプログラム

```
int x=3, y=2;
z=x+y;
printf("%d",z);
```

条件つき繰り返しプログラム

類似したCプログラム

```
int x=10, y=1;
do {
   x=x-y;
   printf("%d",x);
} while (x > 0);
```



関数呼び出しの原理

類似したCプログラム

前半

```
int x=4, y;
printf("%d",x);
y=func(x);
printf("%d",y);
```

後半

```
int func(int z)
{
    z=z+1;
    return z;
}
```

関数呼び出しもジャンプを
用いて実現される

CPUはメモリ上の命令を次々実行しているだけ

一連の命令とデータを集またものがプログラム

目的にかなった処理をさせるためには、そのような動作をするように命令を組み合わせ、並べておく必要がある

プログラミングの本質は組み合わせ

CPU命令レベル（低級言語）でプログラムを書くのはかなりの労力を伴う

C言語などの高級言語は、組み合わせをやりやすくするためのもの

コンパイルは高級言語のプログラムを一連のCPU命令の組み合わせに変換する

CPUの命令もデータ（要は数値）

数値がどの命令を意味するのかはCPUによって異なる

CPUが異なればプログラムは正しく（意図した通りには）動かない

通常、同じ系列のCPUは命令の互換性を確保するように設計されている（ソフトウェアコンパチブル）

あるコンピュータ上に異なる種類のコンピュータをプログラムで仮想的に実現することもできる（エミュレータ）