目次

[**マイクロコンピュータ基礎(1)** 3](#_Toc511945771)

[**目的** 3](#_Toc511945772)

[**原理** 3](#_Toc511945773)

[**CPU(中央処理装置)について** 3](#_Toc511945774)

[**ノイマン型コンピュータについて** 4](#_Toc511945775)

[**実際にノイマン型コンピュータを動作させることについて** 4](#_Toc511945776)

[**スロースキャン・コンピュータについて** 5](#_Toc511945777)

[**実験に係る説明** 6](#_Toc511945778)

[**準備** 6](#_Toc511945779)

[**メモリへ書き込む** 6](#_Toc511945780)

[**メモリの中を見る** 7](#_Toc511945781)

[**連続したアドレスに順番にデータを書き込む** 7](#_Toc511945782)

[**プログラムの実行** 8](#_Toc511945783)

[**負の数の表現** 8](#_Toc511945784)

[**フラグについて** 8](#_Toc511945785)

[**命令の構成** 9](#_Toc511945786)

[**実験1** 11](#_Toc511945787)

[**内容** 11](#_Toc511945788)

[**方法** 11](#_Toc511945789)

[**結果** 12](#_Toc511945790)

[**実験2** 12](#_Toc511945791)

[**内容** 12](#_Toc511945792)

[**方法** 12](#_Toc511945793)

[**結果** 13](#_Toc511945794)

[**実験3** 13](#_Toc511945795)

[**内容** 13](#_Toc511945796)

[**方法** 13](#_Toc511945797)

[**結果** 13](#_Toc511945798)

[**実験4** 13](#_Toc511945799)

[**内容** 13](#_Toc511945800)

[**方法** 14](#_Toc511945801)

[**結果** 14](#_Toc511945802)

[**実験5** 14](#_Toc511945803)

[**内容** 14](#_Toc511945804)

[**方法** 15](#_Toc511945805)

[**結果** 15](#_Toc511945806)

[**実験6** 15](#_Toc511945807)

[**内容** 15](#_Toc511945808)

[**方法** 15](#_Toc511945809)

[**結果** 15](#_Toc511945810)

[**実験7** 16](#_Toc511945811)

[**内容** 16](#_Toc511945812)

[**方法** 16](#_Toc511945813)

[**結果** 16](#_Toc511945814)

[**実験8** 16](#_Toc511945815)

[**内容** 17](#_Toc511945816)

[**方法** 17](#_Toc511945817)

[**結果** 17](#_Toc511945818)

[**実験9** 17](#_Toc511945819)

[**内容** 17](#_Toc511945820)

[**方法** 17](#_Toc511945821)

[**結果** 17](#_Toc511945822)

[**実験10** 18](#_Toc511945823)

[**内容** 18](#_Toc511945824)

[**方法** 18](#_Toc511945825)

[**結果** 19](#_Toc511945826)

[**考察** 19](#_Toc511945827)

[**フォン・ノイマン式コンピュータの基本的な構成要素について図と文章を用いて説明しなさい** 19](#_Toc511945828)

[**実験で作成したプログラムの中には，同じ機能を果たすために複数の異なるプログラムが考えられるものがある．この例をあげてどのように異なったプログラムが考えられるかを説明しなさい．** 20](#_Toc511945829)

[**実験3や実験4のプログラムにおいて停止命令を入れ忘れるとどのような問題が生じ得るかを考えて報告しなさい** 20](#_Toc511945830)

# マイクロコンピュータ基礎(1)

**目的**

マイクロコンピュータ基礎の実験では機械語によるプログラミングを通してマイクロコンピュータの基本動作について理解する．  
第一回ではスロースキャン・コンピュータという実習機器によってコンピュータの中心であるCPUの素振りを観察する．

**原理**

**CPU(中央処理装置)について**

コンピュータは2値情報を記憶・処理する．  
この2値は電圧の高低で判断しており，人間はこれを0と1を利用した2進数として扱う．  
CPUとMainMemoryはバス(共有の信号線の集まり)で接続されている．  
MainMemoryから読み出された命令(2進数)はバスを通りCPUに取り込まれる．  
コンピュータが処理できる2進数を機械語と言う．  
CPUはMainMemoryから逐次命令を読み出して実行する

**ノイマン型コンピュータについて**

ノイマン型コンピュータの機能は4つに分かれる．

1. Fetch  
   命令を取り出して読み込む．
2. Decode  
   命令を解読・解釈する．
3. Execute  
   解読・解釈した命令を実行する．
4. Repeat  
   命令が停止命令でなければ繰り返す．

1~4の動作の流れを命令実行サイクル(Instruction Execution Cycle)と言う．

**実際にノイマン型コンピュータを動作させることについて**

実際にノイマン型コンピュータを動作させる場合の手順は4つに要約できる．

1. アルゴリズムを検討する．
2. アルゴリズムをプログラムする．
3. プログラムをマイクロコンピュータに書き込む．
4. プログラムを指定して実行する．

書き込む際には目盛りの場所を指定する必要がある．  
今回のスロースキャン・コンピュータの場合は5ビットに2進数を利用し，00000~11111までを指定できる

**スロースキャン・コンピュータについて**

* 構成
  + CPU
    - 制御装置
    - 命令デコーダ
    - 算術論理演算装置
    - 各種レジスタ
  + バス
    - 制御バス
      * どの機器がデータを送受信するかという信号と  
        読み込み操作か書き込み操作かを通知するための信号が通る．
    - アドレスバス
      * 読み書きの対象となるアドレスが通る．
    - データバス
      * 実際のデータが送られる．
  + メモリ
    - 32バイト
    - アドレスは00000~11111で指定する
  + I/Oポート
    - 入力装置
    - 出力装置
      * 省略されている．
      * 出力レジスタの値はLEDによって確認する．
* 観察  
  各部のLEDで状態を観察でき，発光していれば1，していなければ0である．
  + CPUの状態を観察する
    - PROGRAM COUNTER
      * 右上のあたり
    - INSTRUCTION
      * PROGRAM COUNTERの下のあたり
    - DECODER
      * INSTRUCTIONの下のあたり
    - ACCUMULATOR
      * DECODERの下のあたり
  + メモリの内容を知る
    - ADDRESS
      * 左上のあたり
      * CPUが読み書きするメモリの番地が一時的に格納される場所．
    - MEMORY
      * ADDRESSの下のあたり
      * 指定されたアドレスから読み出される，書き込まれるデータが格納される場所
  + 命令サイクルを知る
    - HALT
    - FETCH
    - EXE
* 操作
  + 各種スイッチ
    - 盤面の下にある

**実験に係る説明**

**準備**

1. スロースキャン・コンピュータの電源コンセントを接続する．
2. RUN/STOPをSTOPに設定する．
3. 電源をONにする．
4. 正常に電源が投入されるとPOWERランプが点灯する．

**メモリへ書き込む**

**アドレスの指定**

1. ADRS/DATA(盤面の下の真ん中のあたりにあるスイッチ)をADRSに設定する．  
   2. これはスロースキャン・コンピュータに番地を指定することを教えている．
2. データスイッチ(盤面の下の左よりのあたりにある8つのスイッチ)の右側5桁を利用して上:1，下:0としてアドレスを設定する．
3. WRITEスイッチ(盤面の下の真ん中よりちょっとだけ右側にあるSTOP/RUNの左側にあるスイッチ)を押す．
4. 指定した番地の情報がINPUTに取り込まれた後，PROGRAM COUNTERに送られる．
5. 指定した番地の現在の内容がOUTPUTSに表示される(表示内容は機器によって違う)．

**データを書き込む**

1. ADRS/DATAスイッチをDATAに設定する．
2. データスイッチを書き込む内容になるように設定する．
3. WRITEスイッチを押す．
4. データスイッチの内容がINPUTに取り込まれた後，見えない所でデータバスを通ってデータがメモリに転送される．
5. 書き込みが完了するとPROGRAM COUNTERが1だけ増加する
6. 新しい番地の内容がOUTPUTSに表示される．

**メモリの中を見る**

アドレスを指定する操作を行うとそのアドレスの内容をOUTPUTに表示させることができる．

**連続したアドレスに順番にデータを書き込む**

データを書き込む操作を繰り返すことで番地が1ずつ進み，連続したアドレスに順番にデータを書き込むことができる．

**プログラムの実行**

* 自動実行

1. AUTO/STEPスイッチをAUTOに設定する．
2. HIGH/LOWスイッチをLOWに設定する．
3. RUN/STOPスイチをRUNに設定する  
   4. すると，00000番地から順番にプログラムが実行される  
   5. いろいろなランプが点灯，消灯する
4. HALRランプが点灯するとプログラムの実行が停止する．
5. RUN/STOPスイッチをSTOPに設定する．

* ステップ実行

1. AUTO/STEPスイッチをSTEPに設定する．
2. HIGH/LOWスイッチをLOWに設定する．
3. RUN/STOPスイッチをRUNに知っていする．  
   4. すると，CLOCKランプが点灯しFETCHランプが点滅する．
4. FET/EXEスイッチを一度押す  
   6. すると，FETCHランプが点灯して命令の読み込みと解読が行われる．  
   7. EXEランプが点滅する．
5. 再びFET/EXEスイッチを押すと1つ目の命令が実行される．
6. 1つ目の命令を実行し終えるとFETCHランプが点灯する．
7. 再びFET/EXEスイッチを押すと次の命令が実行される．
8. 停止命令が実行されるとHALTランプが点灯する．
9. RUN/STOPスイッチSTOPに設定する．

**負の数の表現**

2の補数表現によって表現する

**フラグについて**

CY：ALUでの結果に桁上り，桁下がりがあった際に光る  
Z：ALUでの結果が0のとき光る  
OF：ALUでの結果がオーバーフロー，ボローした際に光る

**命令の構成**

CPUは2進数によって表される機械語命令を解釈・実行する．  
スロースキャン・コンピュータの機械語命令は全て8ビットである．  
上位3ビットは命令の種類を表す命令コード(オペコードと言う)で，下位の5ビットは演算や転送などの対象となるアドレスでこれをオペランドと言う．

* ロード命令  
  オペランドで指定した番地の内容をACCUMULATORにコピーする命令である．  
  コピーなので指定されたアドレスの内容はそのまま残る．

オペコード：011  
オペランド：コピー元の番地

011オペランド

* ストア命令  
  ACCUMULATORの内容をオペランドで指定した番地に転送(コピー)する．  
  コピーなのでACCUMULATORの内容はそのまま残る．

オペコード：100  
オペランド：コピー先の番地

100オペランド

* 入力命令  
  Inputの内容をオペランドで指定した番地に転送する

オペコード：101  
オペランド：転送先の番地

101オペランド

* 出力命令  
  オペランドで指定した番地をOutputに転送する

オペコード：110  
オペランド：転送元の番地

110オペランド

* 加算命令  
  アキュムレータの内容＋オペランドで指定された番地の値＝アキュムレータに表示される．

オペコード：001  
オペランド：加算する値の格納されたメモリの番地

001オペランド

* 減算命令  
  アキュムレータの内容ーオペランドで指定された番地の値＝アキュムレータに表示される

オペコード：010  
オペランド：減算する値の格納されたメモリの値

010オペランド

* ジャンプ命令  
  プログラムの制御をオペランドで指定されたメモリまでジャンプさせる

オペコード：000  
オペランド：ジャンプ先メモリの番地

000オペランド

* スキップ命令  
  直後のプログラムを飛ばす処理をさせる命令．  
  →プログラム・カウンタが2増える
* 11111000
  + CY==1 then jump
* 11111001
  + CY==0 then jump
* 11111010
  + Z==1 then jump
* 11111011
  + Z==0 then jump
* 11111100
  + OF==1 then jump
* 11111101
  + OF==0 then jump
* 1111110
  + 何もしない
* 11111111
  + 停止命令

これらはすべて特殊命令である．

**実験1**

**内容**

PROGRAM COUNTERが01011になっている．  
01011番地に"11110000"を書き込む．  
01100番地に"00001111"を書き込む．

**方法**

アドレスの指定の操作とデータを書き込む操作をそれぞれの番地に対して行う．

* 01011番地に"11110000"を書き込む．  
  アドレスを01011に指定し，書き込むデータを11110000とする．
* 01100番地に"00001111"を書き込む．  
  アドレスを01100に指定し，書き込むデータを00001111とする．

**結果**

結果を確認するためにテキストM1-7の5.3にあるメモリ内容の読み出しをする．

1. アドレスを指定する操作を行う．
2. OUTPUTに表示される内容を確認する．
3. READスイッチを押す．
4. PROGRAM COUNTERが点滅する．(更新されないが読み込みモードに移行している)
5. 再びREADにスイッチを押す．
6. PROGRAM COUNTERが1だけ更新され，次の番地の内容がOUTPUTにに表示される．その内容を確認する．
7. 再びREADスイッチを押してその次の番地の内容を確認する

結果は下の表1の通り

表1

| **01011** | **01100** |
| --- | --- |
| 11110000 | 00001111 |

**実験2**

**内容**

00011番地のデータを読み出す

**方法**

adrsで番地00011を指定して読み出す

**結果**

| **Input** | **Output** |
| --- | --- |
| 00011 | 01000001 |

**実験3**

**内容**

ロード命令を使用するプログラムを作成・実行．

011オペランド

**方法**

実験書のプログラムを書き込む．  
プログラムを自動実行した後00101番地に書き込んだデータがACCUMULATORに転送されていることを確認する．  
RUN/STOPスイッチをSTOPに設定することを忘れないこと．

**結果**

アキュムレータ： 00110011

**実験4**

**内容**

ストア命令を仕様するプログラムの作成・実行

100オペランド

**方法**

実験書のプログラムを書き込み，メモリ内容を読みだして正しく書き込めているかを確認する間違いがあれば修正すること．  
プログラムを自動実行する．  
その後，00110の番地内容を読み出して，00101番地の内容と同じになっているかを確認する

プログラムをステップ実行する  
以下の2点を確認する

* ストア命令を読み込んで解読した際，DECODERの中で命令を表すSTRが点灯するか？
* ストア命令を実行するときACCUMULATORの内容がメモリレジスタを通ってメモリに書き込まれているか？

**結果**

00110番地： 00110011  
00101番地： 00110011

**STEP実行に関して**

DECODERでSTRを示すランプが光った．  
メモリのところが 00110011 に光った．

**実験5**

**内容**

入出力命令を使用したプログラムの作成・実行

**方法**

実験書のプログラムを書き込み，メモリ内容を読み出して内容を確認．  
その後自動実行する．  
↓  
Inputに先程とは違う値を入力する

**結果**

Input: 00110011  
Output: 00110011

**Inputの値を変えた場合**

私と川上の組ではInputを 11001100に変更した．  
Outputは 11001100 となり意図したプログラム通りに動いていることが分かった．

**実験6**

**内容**

加算命令を使用したプログラムの作成・実行

**方法**

実験書のプログラムを書き込む．  
念の為メモリを読み出して内容を確認し自動実行．  
この時点で二人共操作に慣れてきてプログラムの入力ミスが無くなった．

**結果**

アキュムレータ： 00000100  
この値は00101番地に格納していたアキュムレータに転送される予定の値である 00000011 と，  
00110番地に格納されていた 00000001 の加算結果である．

**ステップ実行した場合**

a. ADDが点灯したことを確認できた  
b. 00110番地の内容がメモリのところに表示された： 00000001  
c. ALUに加算結果( 00000100 )が表示された後，アキュムレータにも 00000100 と表示された．

**実験7**

**内容**

実験6のプログラムを減算するように変更する．

**方法**

00001番地に書き込む内容を 00100110 から 01000110 に変更した後，  
自動実行する．

**結果**

アキュムレータ： 00000010  
この値は00101番地に格納していたアキュムレータに転送される予定の値である 00000011 と，  
00110番地に格納されていた 00000001 の減算結果である．  
また動作の途中でDECODERのSUBが光ったことも確認できた

**実験8**

**内容**

ジャンプ命令を使用したプログラムの作成・実行

**方法**

実験書のプログラムを書き込み自動実行する

**結果**

DECODERのADDとJPが繰り返された．  
無限ループ(ジャンプ)するプログラムなので見飽きた頃に停止させた．

**実験9**

**内容**

01000番地と01001番地に格納されたデータを比較し，大きい方を表示するプログラムの作成・実行

**方法**

実験書にあるプログラムの書き込み，読み出して確認，自動実行．

**結果**

CY==0であったことが確認できた  
JPが光らなかったのでスキップ命令00100番地の命令を実行されたことも確認できた．(EXTが光った)  
OUTPUT: 00100101

**01001番地と01000番地のデータを入れ替えた(大小逆転させた)場合**

CY==1になってJPが光った．  
OUTPUT: 00100101  
ちゃんとデータの大小によって条件分岐ができていることがわかる．

**実験10**

**内容**

実験9のプログラムを変更して小さい値を出力するプログラムの作成・実行

**方法**

実験9は条件分岐，if文を実装していると考えられる．  
↓  
if文の条件式

if(01000>01001){

01000番地の内容を表示

}else{

01001番地の内容を表示

}

↓  
これによって大きいほうが表示されている  
条件式を逆転させれば良い  
↓

if(01000<01001){

01000番地の内容を表示

}else{

01001番地の内容を表示

}

↓  
00010番地のコードを11111001から11111000に変更する

**結果**

01000：00100101  
01001：00100011  
OUTPUT：00100011

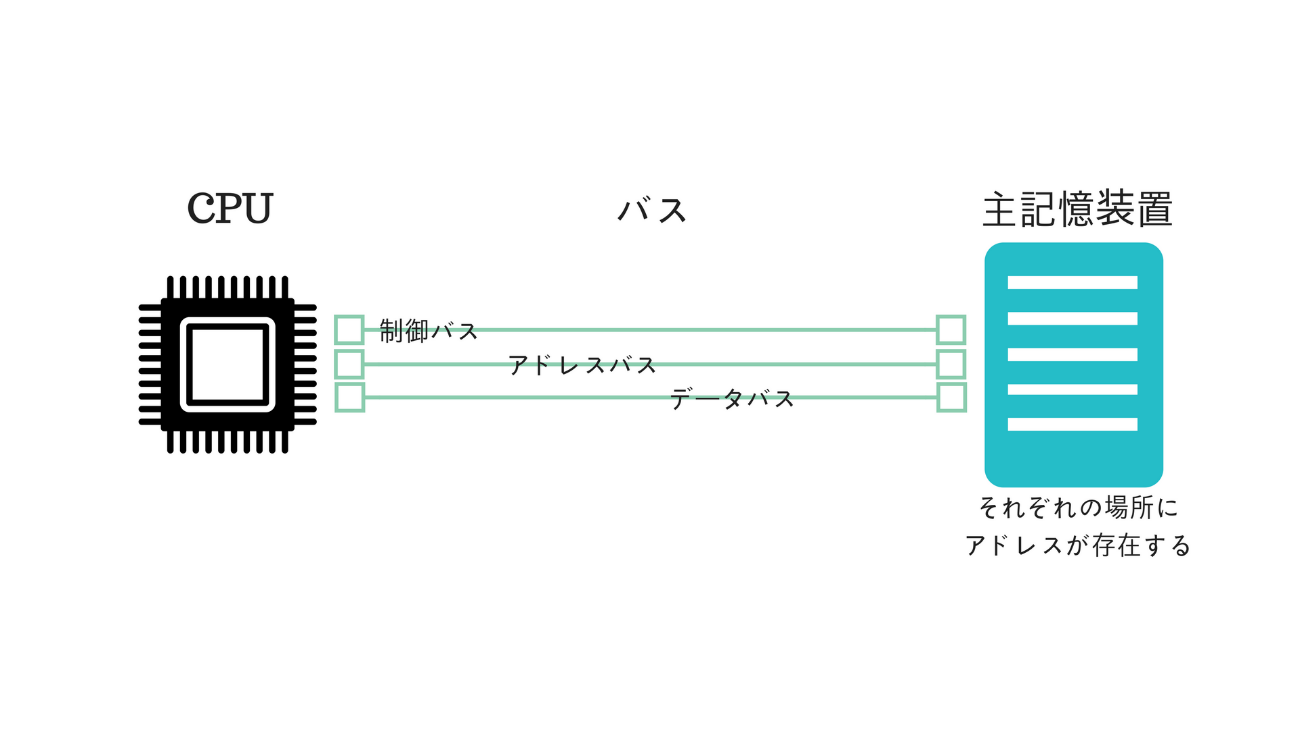
**01000と01001を入れ替えてみる**

01000：00100011  
01001：00100101  
OUTPUT：00100011

となり小さいほうが表示されているため成功したと考えられる．

**考察**

**フォン・ノイマン式コンピュータの基本的な構成要素について図と文章を用いて説明しなさい**

フォン・ノイマン式コンピュータではCPUと主記憶装置がバスによってつながっており，  
CPUが主記憶装置のデータや命令を処理することで動作する．  
考察1用の図↓

**実験で作成したプログラムの中には，同じ機能を果たすために複数の異なるプログラムが考えられるものがある．この例をあげてどのように異なったプログラムが考えられるかを説明しなさい．**

実験6を例に考えると加算命令のオペランドで指定した番地に2の補数表現によって表された負の数を格納しておくことで，減算命令と同様の振る舞いをさせることができる．

**実験3や実験4のプログラムにおいて停止命令を入れ忘れるとどのような問題が生じ得るかを考えて報告しなさい**

実験3の場合，00001の停止命令がなかったときそのまま進んで何が入ってるかわからないメモリの命令を実行してしまったり，  
何もなかったとしても，00101番地においてある00110011を加算命令として解釈してしまい，意図しない動作をする可能性がある．  
実験4の場合も同様のことが言える．