# 情報工学実験2
# コンピュータアーキテクチャと命令セット

035760A : 横田敏明

実験実施 : 2004 / 10 / 29
提出日 : 2004 / 11 / 5
共同実験 : 国吉貴文

# 1 実験目的

教育用ワンボードコンピュータ KUE-CHIP2 を用いて例題プログラムを実行することにより，KUE-CHIP2 の操作方法を習得．さらに，簡単な機械語 (マシン語) プログラムを作成し，機械語プログラムの構造を理解することを目的とする．

# 2 実験報告

## 2.1 弟 3 章 3.1 の例題プログラムの解読と C 言語への書き換え

以下のプログラムを打ち込む．

```
ADRS DATA  OPECODE          解説
 00   75   ST ACC,(03H)     ACC を (03H) に格納
 01   03
 02   C0   EOR ACC,ACC      ACC を初期化
 03   B5   ADD ACC,(03H)    ACC に (03H) を足して ACC に格納
 04   03
 05   AA   SUB IX,1         IX から 1 を引く
 06   01
 07   31   BNZ 03H          IX の値が正ならば，03 へ戻る
 08   03
 09   0F   HLT              終了
```

補足 : メモリにはプログラム領域とデータ領域がある．プログラム領域のデータは，命令をコード化された数字が格納される．例えば，

```
03  B5  ADD ACC,(03H)
```

は ACC(アキュムレータ) に (03H) の内容を足し，それを ACC に格納する意味である．プログラムはジャンプ命令などの制御命令がない限り，アドレス順に実行される．実行されるとプログラムカウンタが 1 づつ増加し，次の実行ではカウンタの番号のアドレスの命令を実行する．

### 2.1.1 C 言語への書き換え

```
#include <stdio.h>
int main()
{
  int acc,ix,add[10];
```

```
  printf("input ACC : ");
  scanf("%d",%acc);
  printf("\n input IX : ");
  scanf("%d",&ix);
  add[3] = acc;
  acc = 0;
  for(ix; ix>0 ;ix--){
    acc += add[3];
  }
  printf("\n%d \n",acc);
}
```

## 2.2 リファレンスマニュアル第 3 章 3.1 のプログラムを 0x20 番地から実行せよ.

プログラムを 0x20 番地から格納し，ADDRESS LED を 0x20 番地に合わせて実行すればよい.

## 2.3 KUE-CHIP2 の使用法について

以下の各項目について説明せよ．また，KUE-CHIP2 を使って，実際に操作せよ.

(a)ACC の内容を観測するには，どのように操作すればよいか.
解答 : SELsw を 0100 にすると，DATA LED に ACC の内容が表示される.

(b)IX の内容を観測するには，どのように操作すればよいか.
解答 : SELsw を 0101 にすれば表示される.

(c)PC の内容を観測するには，どのように操作すればよいか.
解答 : SELsw を 0010 にすれば表示される.

(d)ACC に 0x10 をセットするには，どのように操作すればよいか.
解答 : SELsw を 0100 に，DATAsw を 00010000 にして，SETsw を押す.

(e)IX に 0x10 をセットするには，どのように操作すればよいか.
解答 : SELsw を 0101 に，DATAsw を 0001000 にして，SETsw を押す.

(f)PC に 0x10 をセットするには，どのように操作すればよいか.

解答 ： SELsw を 0010 に，DATAsw を 00010000 にして，SETsw を押す．

(g)0x80 番地の内容を見るには，どのように操作すればよいか．
解答 ： ADDRESSsw を 80H(10000000) にして，SELsw を 0001 にし，IMCsw を CHECK に入れる．

(h)0x80 番地に 0x34 をセットするには，どのように操作すればよいか．
解答 ： ADDRESSsw を用い，80H(10000000) に指定した後，DATAsw を 0x34 にして，SETsw を押す．

(i)0x00 番地から 0x0F 番地茉での内容を連続的に見るには，どのように操作すればよいか．
解答 ： ADDRESS を 0x00 に指定し，ADRINC を押しては確認する作業を繰り返す．

(j)0x90 番地から 0x0F 番地までの内容を全て 0x11 にセットするには，どのように操作すればよいか．
解答 ： SELsw を 0000 にし，ADDRESS を 0x00 にあわせ，DATAsw を 00010001 にして SETsw を押す．その後 ADRINC を押して SETsw を押す作業を 0x0F まで繰り返す．

(k)0x90 番地から 0x9F 番地までの内容を連続的に見るには，どのように操作すればよいか．
解答 ： SELsw を MAR(1000) に設定し，DATAsw を 10010000 にして SETsw を押す．さらに SELsw を 0000 に戻し，内容を確認する．ADRINC を押して内容を確認することを繰り返す．

(l)0x90 番地から 0x9F 番地までの内容を全て 0x11 にセットするには，どのように操作すればよいか．
解答 ： SELsw を 1000 に，DATAsw を 10010000 にして SETsw をおす．SELsw を 0000 に戻し，DATAsw を 00010001 に設定し，SETsw を押す．ADRINC を押して SETsw を押すことを繰り返す．

(m) プログラムを 0x00 番地から実行するには，どのように操作すればよいか．
解答 ： ADDRESS を 0x00 に合わせ，SSsw を押す．

(n) プログラムを 0x20 番地から実行するには，どのように操作すればよいか．

解答 : ADDRESS を 0x20 にあわせ，SSsw を押す.

(o) プログラムを 1 命令づつ実行するには，どのように操作すればよいか.
解答 : SIsw を押す.

(p) プログラムを 1 フェーズずつ実行するには，どのように操作すればよいか.
解答 : SPsw を押す.

## 2.4 リファレンスマニュアル第 4 章 4.1 の例題プログラムを KUE-CHIP2 に入力し，実行せよ．また，このプログラムを解析し，C 言語によるプログラムに書き換えよ.

以下に例題プログラムを示す.

```
00 : 6C 80  LD IX,[N]      0x80 の内容を IX に代入.
02 : 62 00  LD ACC,0       ACC を初期化.
04 : B1     ADD ACC,IX     ACC = ACC + IX
05 : AA 01  SUB IX,1       IX = IX - 1
07 : 33 04  BP             04 に移動.
09 : 74 81  ST ACC,[SUM]
0B : OF     HLT            終了.
```

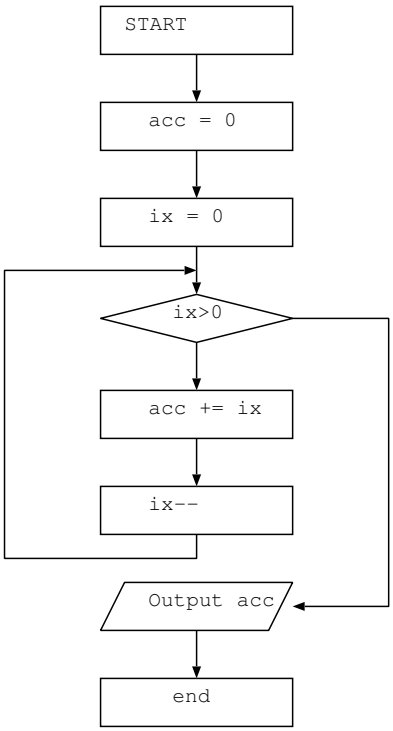


図 1: C 言語によるプログラムのフローチャート

```
#include <stdio.h>

int main()
{
  int acc,ix;
  acc = 0;
  for(ix=10; ix > 0; ix--)
    acc += ix;
  printf("%d\n",acc);
}
```

## 2.5 学籍番号 6 個の数を足してその合計を求めるプログラムを各自で作成し，実行せよ．ただし，学籍番号の 6 個の数は，データ領域の 0x00 番地 (0x100 番地) から 0x05 番地にあらかじめ格納しておくこと．また，このプログラムを解析し，C 言語によるプログラムに書き換えなさい

| ADRS | DATA | OPECODE | 解説 |
|---|---|---|---|
| 00 | 62 00 | LD ACC,0 | ACC = 0 |
| 02 | B5 00 | ADD ACC,(00H) | ACC に 1x00 の内容を足す |
| 04 | B5 01 | ADD ACC,(01H) | ACC に 1x01 の内容を足す |
| 06 | B5 02 | ADD ACC,(02H) | ACC に 1x02 の内容を足す |
| 08 | B5 03 | ADD ACC,(03H) | ACC に 1x03 の内容を足す |
| 0B | B5 04 | ADD ACC,(04H) | ACC に 1x04 の内容を足す |
| 0D | B5 05 | ADD ACC,(05H) | ACC に 1x05 の内容を足す |
| 0E | 0F | HLT | 終了 |

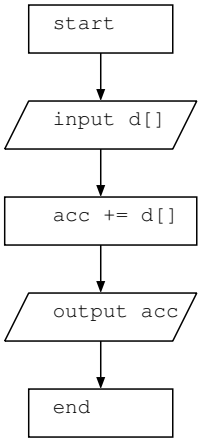


図 2: C 言語によるプログラムのフローチャート

```
#include <stdio.h>

int main()
{
  int d[6], acc=0,count;
  printf("input six numbers...\n");
  for(count=0;count<6;count++)
    scanf("%d",&d[count]);
  for(count=0;count<6;count++)
    acc = acc + d[count];
  printf("acc = %d\n",acc);
}
```

# 3　コンピュータは，入力装置，記憶装置，演算装置，出力装置，制御装置の 5 つの主要構成要素を持つ．コンピュータの主要構成要素について，以下の設問に答えよ．

## 3.1　(a) 入力装置

### 3.1.1　入力装置の役割を簡単に説明せよ．

解答：プログラムやデータを入力する装置．

### 3.1.2　入力装置の具体例を 3 つ以上挙げ，それぞれの特徴を簡単に説明せよ．

- キーボード：文字を入力することに特化した装置
- マウス：カーソルを操作することに特化した装置
- マイクロフォン：空気の振動を入力する装置
- カメラ：映像をリアルタイムで入力する装置

### 3.1.3　3, 自分のコンピュータ，或は自宅や大学にあるコンピュータの中から 1 台選択し，そのコンピュータに内蔵あるいは外付けされている入力装置を全て列挙せよ．

解答：自宅のパソコン NEC valuestar

- keybord
- マウス
- マイク
- 各種スキャナ
- 各種マイクロフォン

## 3.2　(b) 記憶装置

### 3.2.1　記憶装置の役割を簡単に説明せよ．

解答：入力されたプログラムやデータを記憶する．

#### 3.2.2 コンピュータは，主記憶装置であるメインメモリ以外に大容量の補助記憶装置を装備しているのが普通である．補助記憶装置の例を 5 つ以上挙げ，それぞれの特徴や違いなどを簡単に説明せよ．

- CD-R：約 700MB の記憶容量を持つ．
- DVD-R：約 4.7GB の記憶容量を持つ．
- ハードディスク：磁気ディスクを集積した大容量補助記憶装置
- MO：マグネティックオプティカル (光磁気) ディスク．230MB ほど
- フラッシュメモリ：揮発性メモリ．小型で持ち運びに有利．

#### 3.2.3 上記で選択したコンピュータに内蔵或は外付けされている記憶装置をすべて列挙し，それぞれの容量を調べて報告せよ．

解答：
内蔵メモリ：240MB RAM
内蔵ハードディスク：MAXTOR 4K060H3

### 3.3 (c) 出力装置

#### 3.3.1 出力装置の役割を簡単に説明せよ．

解答：処理結果を出力する．

#### 3.3.2 出力装置の具体例を 3 つ以上挙げ，それぞれの特徴を簡単に説明せよ．

- ディスプレイ：視覚情報として出力を表示する．
- プリンタ：紙に絵や文字などを印刷する装置．
- スピーカー：音声情報を出力する．

#### 3.3.3 上記で選択したコンピュータに内蔵あるいは外付けされている出力装置をすべて列挙せよ

.

- ディスプレイ：CRT ディスプレイ DV17D6
- プリンタ：CANON 560i
- スピーカ：YAMAHA YST-MS201

## 3.4 (d) プロセッサ (CPU) = 演算装置 + 制御装置

### 3.4.1 演算装置の役割を簡単に説明せよ.

解答：プログラムに基づく演算および処理を実行する.

### 3.4.2 制御装置の役割を簡単に説明せよ.

解答：入力，記憶，演算，出力の機能を順序よく動作させるための制御を行う.

### 3.4.3 上記で選択したコンピュータに搭載されている CPU を挙げ，その性能や特徴について報告せよ.

解答：
CPU : AMD Athlon Processor
clock : 896MHz
特徴：ペンティアムプロセッサよりややグレードを落とした兼価版で，性能よりもコストパフォーマンスに重点をおいたプロセッサ．ソケット形状はペンティアムと異なる.

# 4 コンピュータのプロセッサ内部では，レジスタと呼ばれる記憶装置が使用されている．以下に示す各レジスタの役割を調査し，説明せよ．また，以下の各レジスタのうち，KUE-CHIP2 に無いものを挙げよ.

## 4.1 アキュムレータ

レジスタの一種．加算などの処理で，変数を 3 つ使う方法よりもアキュムレータというレジスタを用いて計算したほうが効率がよい．変数のうちの他方をアキュムレータの情報を暗黙のうちに指定し，結果をアキュムレータに入れることで，アドレスごとにビットを使う必要をなくしている.

## 4.2 インデクスレジスタ

指定されたアドレス部の内容と，インデクスレジスタと呼ばれるレジスタで指定された内容を実効アドレスとする方式の場合に使用される.

### 4.3 プログラムカウンタ

プログラムの格納されているアドレスを指定する．プログラムカウンタの指定するアドレスが実行されるアドレスとなるので，命令によってプログラムカウンタの内容が変更された場合はジャンプしたり誤作動を起こすこともある．

### 4.4 メモリアドレスレジスタ

実行するべき命令をメインメモリから取り出すとき，または，データをメインメモリから読み出すとき，もしくは，データをメインメモリに格納するときに使用する．

### 4.5 命令レジスタ

現在実行している命令を格納しておく場所．

### 4.6 フラグレジスタ

汎用的にある状態を保存するレジスタ．

### 4.7 スタックポインタ

LIFO 型のデータのアドレスを格納する．

### 4.8 汎用レジスタ

プログラム中で一時的に使われるデータの記憶場所を格納．

### 4.9 KUE-CHIP2 に無いもの

フラグレジスタ，汎用レジスタ

**5　機械語の命令は，表 1.1 に示したように，一般にデータ転送命令，演算命令，制御命令，入出力命令，特殊命令の 5 種類に分類される．KUE-CHIP2 の命令を，これら 5 種類の命令に分類し，各分類ごとにアセンブリ言語と機械語の対応表を書け．また，各命令の機能を簡単に説明せよ．**

| 機能 | アセンブリ言語 | 機械語 | 機能 |
|---|---|---|---|
| データ転送命令 | ST / LD | 75 / 6C | (A) → B / (B) → A |
| 演算命令 | EOR / ADD | C0 / B1 | (A) + (B) → A / (A) + (B) → A |
| 制御命令 | BN2 / HLT | 31 / 0F | 1 でなければ / 停止 |
| 入出力命令 | OUT / IN | 03 / 0B | (ACC) → OBUF / (IBUF) → ACC |
| 特殊命令 | JP | CA | 指定番地へジャンプ |

表 1:機能対応表

# 6　考察

今回の実験で使用したコンピュータは，学習用の単純なコンピュータである．クロック周波数を調節して動作を確認できる機能があった．計算機の重要な機能は，演算と記憶だと思われる．演算は人間の計算速度よりはるかに高速で正確に行われ，記憶する機能があるからこそ大きな計算量を実現できるからである．命令に対する演算と制御は最大の自由度を持たせるように設定しなければならない．例えば，命令を繰り返したり，ジャンプする機能を実現できなければ，複雑で機械的な計算量の多い処理が全くできなくなってしまうからである．このような意味で，KUE-CHIP2 は最低限の機能を幅広く装備していると考えた．高級言語はプログラムを作成する上で便利なツールとなっているが，性能の低いプロセッサと領域の狭いメモリを使わざるを得なかった時代では，アセンブラ言語や機械語はとても身近なものであったと感じた．そして，低級言語でもあらゆるプログラムを作成することは可能であると理解した．ただし，高級言語でプログラムを作成するほうがはるかに開発効率がよく，プロセッサの性能が目覚しく発展した今となっては，アセンブラなどによる高速化はそれほど意識するほどのことでもない．しかし，高級言語で記述するよりも，低級言語で記述したプログラムのほうが一般的には高速であるといわれている．C 言語などでは，コンパイルの段階で余分なメモリ領域の確保を行うことが避けられない，などの理由からである．このため，高速化を重要視するプログラムなどでは今でもアセンブラ言語が使われている．例として，ファイナルファンタジー 8(旧スクウェア) というゲームソフトは，高速化を図るためアセンブラ言語で記述されている．

# 7　参考文献


- 実験 2 付録リファレンスマニュアル
- 計算機アーキテクチャ
- the playstation 1998/4 月号