# コンピュータアーキテクチャと命令セット

学籍番号 045713C:大城和也

実験実施日:平成 17 年 10 月 17 日 (月)
提出日:平成 17 年 10 月 24 日 (月)

共同実験者
045709E:上原直久

# 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事により，KUE-CHIP2 の操作方法を習得する．また，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事を目的とする．

# 2　概要

本実験は実験 (1) で KUE-CHIP2 の操作方法を学ぶためにリファレンスマニュアルを読み，スイッチやボタンの意味や扱い方を覚え，実験 (2) で実際にマニュアルに従って例題プログラムを動かしてみる．実験 (3) では，先の実験で扱ったプログラムを解読して、C 言語に書き換えてみる．次に実験 (4) では実験 (1) で学んだ事柄を使い、指示された簡単な操作の方法を模索する．最後の実験 (5) ではこれらのまとめとして格納された値を足す簡単なプログラムをアセンブラで考え、機械語に直し，KUE-CHIP2 に組み込み，その動作を確認する．

# 3　実験結果

## 3.1　「KUE-CHIP2 教育用ボード・リファレンスマニュアル」(以下、マニュアル) の第 1 章及び、第 2 章を読み，操作方法を調べよ．(報告の必要なし)

本実験の結果については．報告を省略する．

## 3.2　マニュアルの第 3 章の 3.2(P18) 及び 3.3(P24) の内容に従って例題プログラムを入力し，実行せよ．(報告の必要なし)

本実験の結果については，報告を省略する．

## 3.3　マニュアルの第 3 章 3.1(P17) の例題プログラムを解読し，C 言語によるプログラムに書き換えよ．

---

```
アドレス 機械語  アセンブラ
   00      75    ST  ACC,(03H)
   01      03
```

```
02    C0    EOR ACC,ACC
03    B5    ADD ACC,(03H)
04    03
05    AA    SUB IX,1
06    01
07    31    BNZ 03H
08    03
09    0F    HLT
```

---

上記のアセンブラプログラムは5×4を足し算を行う事で実現したものである．プログラムを動かすとACCには14が格納されており，IXは00，PCには0Aが格納されていた．プログラムの具体的な流れは次のようになっている．(ちなみにこのプログラムは初期値としてACCに5，IXに4が格納されているものとする．)

**1.** ACCの値を03H番地に代入する．

**2.** ACCに自分自身の排他的論理和をビット演算したものを代入．(つまりは0になる)

**3.** ACCにデータレジスタの03Hの値を足す．

**4.** IXの値を1減らす．

**5.** BNZはBranch on Not Zeroの略．つまりは「0でなければ」という意味．BNZは直前の命令の値を評価しており，今回の場合はIXが対象となっている．値が0でなかった場合，後に書かれているプログラムのアドレスに移る．今回は03Hのアドレス．これで繰り返し文となっている．

**6.** 上記の繰り返しにより03H 06Hの操作を4度行い，IXが0となった後，HLT命令によりプログラムは終了する．

次にこの流れを踏まえてC言語に書き換える．下にあるのがC言語に直したものである．このプログラムではアセンブラでのACC，IX，03H番地をそれぞれint型のACC，ix，aという変数に置き換えている．BNZでの繰り返し判定はdo-while文を用いて表現している．

それぞれのフローチャートを図1と図2に示す．

---

```
#include<stdio.h>

int main(){
  int a, ix, ACC;                /* 変数の宣言 */
```

```
  a = 0;                          /* a の初期化 */
  ix  = 4;                        /* ix の初期化 (4 を代入) */
  ACC = 5;                        /* ACC の初期化 (5 を代入) */

  a  = ACC;                       /* a に ACC を代入 (a = 5) */
  ACC= ACC^ACC;                   /* ACC の排他的論理和をとる (ACC = 0) */

  do{
    ACC += a;                     /* ACC = ACC+a と同じ */
    ix--;                         /* ix のデクリメント */
  }while(ix != 0);                /* ix が 0 じゃなければ繰り返す */

  printf("合計は%x\n", ACC);     /* ACC の値を 16 進数で画面に表示 */
}
```

---

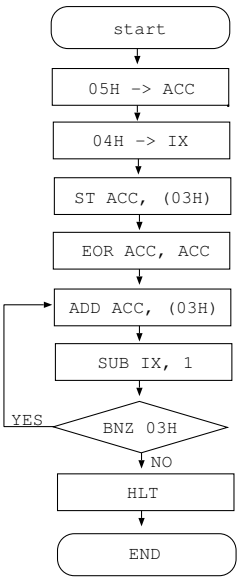


図 1: アセンブラフローチャート

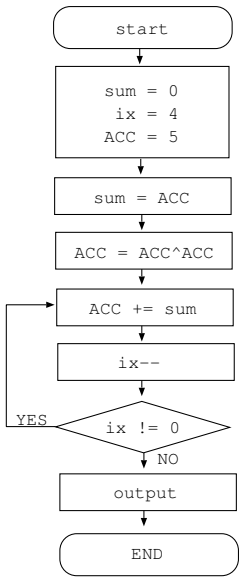


図 2: C 言語フローチャート

## 3.4 以下の項目に付いて，その操作方法を調べ，実際に KUE-CHIP2 を操作し，動作を確認せよ．

**(a)** ACC の内容を見る方法

ACC の内容を見るには SEL スイッチを $(0100)_2$ にすれば良い．表示は LED で確認する．

**(b)** IX の内容を見る方法

先ほどと同様に SEL スイッチを操作する．IX の場合は $(0101)_2$ にする．

**(c)** PC の内容を見る方法

これも同様に SEL スイッチを $(0010)_2$ にすることで内容を確認できる．

**(d)** ACC に 0x10 をセットする方法

まず，SEL を $(0100)_2$ にし，DATA スイッチを $(00010000)_2$ として，SET ボタンを押す．

**(e)** IX に 0x10 をセットする方法

SEL は $(0101)_2$ で，DATA スイッチを $(00010000)_2$ にして，SET ボタンを押す．

**(f)** PC に 0x10 をセットする方法

SEL は $(0010)_2$ で，DATA スイッチを $(00010000)_2$ にして，SET ボタンを押す．

**(g)** 0x80 番地の内容を見る方法

IMC スイッチを Check に入れる．ADRESS8 0(9 ビットディップ SW) を $(10000000)_2$ にすることで表示される．

**(h)** 0x80 番地に 0x34 をセットする方法

先ほどのように 0x80 番地にアクセスした後，DATA スイッチを $(00110100)_2$ にして SET ボタンを押す．

**(i)** 0x00 番地から 0x0F 番地までの内容を連続的に見る方法

SEL を $(1000)_2$ にして MAR を表示する．DATA スイッチの値を $(00000000)_2$ にして SET ボタンを押す．その後，観測したいものを SEL で表示し，ADRINC ボタンを押して次のアドレスの内容が表示できるので，アドレスが 0x0F になるまで繰り返す．

**(j)** 0x90 番地から 0x9F 番地までの内容を連続的に見る方法

先ほどと同様に SEL を MAR にし，DATA スイッチを $(10010000)_2$ にして SET ボタンを押す，その後，ADRINC ボタンを使って次の値を見ていく．

**(k)** 0x00 番地から 0x0F 番地までの内容を全て 0x11 にセットする方法

(i) と同じようにして，MAR を設定する．SEL を変更する．次に DATA スイッチを $(00010001)_2$ にして SET ボタンを押してセットし，ADRINC ボタンを押して次のアドレスに移り，また SET ボタンを押す，これを 0x0F まで続ける．

**(l)** 0x90 番地から 0x9F 番地までの内容を全て 0x11 にセットする方法

(j) と同じようにして，MAR を設定する．次に DATA スイッチを $(00010001)_2$ にして SET ボタンを押してセットし，ADRINC ボタンを押して次のアドレスに移り，また SET ボタンを押す，これを 0x9F まで続ければよい．

**(m)** プログラムを 0x00 番地から実行する方法

SEL を $(10000000)_2$ にして DATA スイッチを $(00000000)_2$ にし SET を押す．その後，SS ボタンを押すとプログラムが HALT 命令が来るまで実行する．

**(n)** プログラムを 0x20 番地から実行する方法

SEL を $(10000000)_2$ にして DATA スイッチを $(00100000)_2$ にし SET を押す．その後，SS ボタンを押してプログラムを起動させる．

**(o)** プログラムを 1 命令ずつ実行する方法

SI ボタンを押すことで 1 命令ずつ実行することが可能である．

**(p)** プログラムを 1 フェーズずつ実行する方法

SP ボタンを押すことで 1 クロックフェーズだけ命令の実行が行える．

## 3.5 学籍番号の 6 個の数字を足してその合計を求めるプログラムを作成せよ．但し，6 個の数字はデータ領域の 0x00 番地から 0x05 番地にあらかじめ格納しておく物とする．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．

---

```
アドレス 機械語 アセンブラ
  00     65    LD  ACC,(00H)
  01     00
  02     B5    ADD ACC,(01H)
  03     01
  04     B5    ADD ACC,(02H)
```

```
  05     02
  06     B5    ADD ACC,(03H)
  07     03
  08     B5    ADD ACC,(04H)
  09     04
  10     B5    ADD ACC,(05H)
  11     05
  12     0F    HLT
```

---

このアセンブラプログラムはデータ領域の 0x00 番地から 0x05 番地にある値を全て足すものである．

プログラムの流れとしては図 3 のフローチャートを見ていただければわかるように 1 本線となっている．まず ACC にデータ領域の 00H 番地の値を代入する．次に ACC にデータ領域の 0x01 番地の値を足す．同じような動作を 0x02，0x03，0x04，0x05 番地にして続け，最後は HLT 命令となり終了するといった簡単なプログラムである．

先のプログラムを C 言語のプログラムに書き換えると次のようになる．データ領域における 0x00 から 0x05 番地のデータはそれぞれを a から f までの変数とし，ACC を sum という変数とした．sum には初期化の時点で a を代入し，その後 b から f の値を足す．最後に 16 進数で sum の値を表示させるものとなっている．C 言語プログラムのフローチャートを図 4 に示す．

---

```
int main(){
  int a, b, c, d, e, f, sum;     /* 変数の宣言 */
  a=0;                           /* それぞれのデータの代入 */
  b=4;
  c=5;
  d=7;
  e=1;
  f=3;

  sum =a;                        /* sum に a を代入 */
  sum+=b;                        /* それぞれを sum に足す */
  sum+=c;
  sum+=d;
  sum+=e;
  sum+=f;
```

```
  printf("合計は%x\n", sum);     /* 16 進数で sum を表示 */
}
```

---

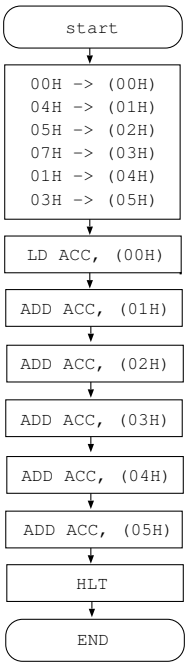


図 3: アセンブラフローチャート

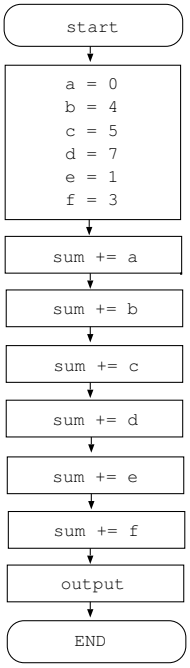


図 4: C 言語フローチャート

# 4 考察

## 4.1 実験 (3)，(5) の考察

アセンブラ言語とは機械語を多少分かりやすくしたもので機械語と 1 対 1 で対応しており，機械語にもっとも近い言語の事である．機械語とは CPU が直接解釈できる言語の事で，数字だけで表現されている．アセンブラ言語の利点は CPU と近い位置にあることでその CPU を最大限に生かした効率の良いプログラムを作る事が出来る事である．一方，アセンブラ言語は機械語自体の命令が少ないため，一見簡単なプログラムを作成するにも処理を多く書く必要が出てくる．今回の実験の場合でも乗算をするために 5 を 4 回足すという作業を行った．これが C 言語の場合だと 5*4 だけですむものであり，アセンブラ言語の欠点ともいえるだろう．もう 1 つの欠点は互換性がない事である．先も述べた事

だがアセンブラは機械語と 1 対 1 の関係にある．つまりはその動作は完全に CPU に依存するのである．よって Windows で作られたプログラムを Mac で動かしたりする事が出来ない．C 言語などの高級言語はソフトウェアによってその内容がコンパイルされ機械語になるので互換性があるのである．

## 4.2 実験 (4) について

実験 (4) で上げた以外に操作方法を下に列挙する．

**(g),(h)** SEL を MAR(1000) にセットして，DATA スイッチを $(10000000)_2$ にして SET ボタンを押す事で 0x08 番地にアクセスでき，またそこから SEL を変更し，DATA スイッチを $(00110100)_2$ とし，SET ボタンを押す事で 0x34 をセットする事が可能．

**(i),(k)** 操作として違う点は 0x00 番地に戻る際に RESET ボタンを使うことで戻る事．RESET をつかうと 0x00 番地に戻れた．あとの操作は同じ．他に考えるならアドレスを 1 つずつ ADRESS8 0(IMC=Check としておく) で変更する事もできるが実用性はないと思われる．

**(j),(l)** ADRINC，あるいは ADRDEC を使って 0x09 番地まで辿り着くようにする．後は同じ様に．これも実用性はほとんど無いと思われる．

## 4.3 その他

今回，実験 (5) において隣のグループとプログラムが終了する速度を比べた．隣グループのプログラムは実験 (3) のように繰り返し文を扱うプログラムであった．CLK を 12(1Hz) として比べたとき，私たちのグループの方が遥かに早く終了した．これは少し意外だった．

# 5 調査課題

## 5.1 コンピュータ主要構成要素に関して

### 5.1.1 自分の PC のスペック

私の ibook のスペックを表 5.1.1 に示す．

表 1: スペック

| | |
|---|---|
| メーカ名 | Apple |
| 商品名 | ibook G4 |
| 型番 | UV4109Y2PGZ |

### 5.1.2 入力装置について

1. 入力装置の役割とはユーザがコンピュータへ情報を与える事である．
2. 入力装置の具体例としてキーボード，マウス，イメージスキャナなどが挙げられる．キーボードはキーを打つ事により文字や記号を入力する事ができ，マウスはその名の通りねずみに似た形をしており，卓上を滑らせて裏のボールを回転させて画面上の位置を指示するものである．イメージスキャナは紙や写真といった媒体を画像データとしてコンピュータに読み込むものである．
3. 自分の ibook には表 3 のような入力装置が内蔵されている．

表 2: 内蔵入力装置

| |
|---|
| キーボード (keyboard) |
| スライドパッド (slide pad) |
| 無指向性マイクロフォン |

### 5.1.3 記憶装置について

1. 記憶装置の役割はプログラムやデータを記憶する事である．
2. 補助記憶の例として磁気ディスク装置，フロッピーディスク装置，光磁気ディスク装置，光ディスク装置，半導体メモリ装置が挙げられる．

   磁気ディスク装置とは一般的にハードディスクと呼ばれているものであり，磁気ディスクは磁性体を薄い円盤に塗ったもので複数重なった状態になっている．大容量化が進んでいる．

   フロッピーディスクとは磁性体を塗布した薄い円盤を備えており，容量は小さく，読み書きの速度も速くはない．最近は使われる事は少なくなってきている．

光磁気ディスク装置とはレーザ光線と磁気を利用して読み書きを行う装置のことで，MO と呼ばれており，容量は 128MB，230MB，540MB，640MB，1.3GB，2.3GB などがあり，640MB が普及している．

光ディスク装置とはレーザ光線でデータを読み書きする装置で CD，DVD，PD などが挙げられる．CD は音楽の記憶装置として用いられたりなどと現在広く普及している媒体といえる．

半導体メモリ装置とは電気的に記録する装置の事で，メモリースティックや SD メモリーカードフラッシュメモリーなどが挙げられる．これらの補助記憶装置は不揮発性だが，同じ半導体メモリの DRAM や SRAM などの主記憶装置におけるものは揮発性となっている．

3. 私の ibook には表 3 のような記憶装置が内蔵されている．

表 3: 記憶装置

| 装置名 | 容量 |
|---|---|
| メインメモリ | 640MB |
| 2 次キャッシュ | 245KB |
| 磁気ディスク装置 | 27.94GB |
| コンボドライブ | |

#### 5.1.4 出力装置について

1. 出力装置の役割はコンピュータからユーザへと情報を与える事である．

2. 代表的な出力装置としてディスプレイ，プリンタ，音声出力装置などが挙げられる．

   ディスプレイとはコンピュータ内のデータを画面に映し出す装置であり，CRT ディスプレイ，液晶ディスプレイ，プラズマディスプレイなどがある．

   プリンタは，コンピュータで作成したデータを紙などに印刷する装置の事であり，インパクト式，ノンインパクト式，シリアル，ラインなど様々な種類がある．

   音声出力装置とはコンピュータ内の音楽データや音声をだすスピーカなどの装置の事である．

3. 私の ibook には表 3 の出力装置が内蔵されている．

表 4: 出力装置

| 装置名 |
|---|
| LCD ディスプレイ |
| スピーカー |
| プリンタ |

#### 5.1.5 プロセッサについて

1. 演算装置は，算術演算装置と論理演算装置があり，それぞれ四則演算，論理演算を行う回路である．演算装置には演算結果を保持するためのアキュムレータやフラグレジスタがある．

2. 制御装置は，コンピュータを動作させるプログラムを解析し，他の装置に制御信号を送る働きをする．

3. 私の ibook のプロセッサの能力は表 3 の様になっている．

表 5: CPU

| CPU タイプ | PowerPC G4 (3.3) |
|---|---|
| CPU 速度 | 800MHz |
| 二次キャッシュ | 256KB |

### 5.2 コンピュータには CPU から近い順に，レジスタ，キャッシュ，メインメモリ，ハードディスクなどの記憶装置が配置されている．これらの各記憶装置の役割と多種多様な記憶装置が用いられていることについて

**レジスタ** レジスタとは CPU の内部にあり，演算や実行状態の保持に CPU が直接使用する記憶域の事である．レジスタにはアキュムレータ，スタックレジスタ，プログラムカウンタ，割り込みレジスタなどの役割が決められているものがあり，特に決められていないものは汎用レジスタと呼ばれている．レジスタの動作速度はきわめて速く高速だが容量は非常に小さい．

**キャッシュ** キャッシュとはメインメモリとプロセッサとの速度差を緩和するための記憶装置で使用頻度の高いデータなどを保持している．速度はメインメモリより高速だが，面積辺りの記憶容量が少なく，コストは高い．

**メインメモリ** コンピュータ内部でデータやプログラムを一時的に記憶しておくもので主記憶装置とも呼ばれる．電源を切ると内容が失われる揮発性である．

**ハードディスク** 代表的な記憶装置で速度はメインメモリと比べ遅いが，容量は大きく，単位当たりのコストは安い．

これらの多種多様な記憶装置が用いられている理由としては，レジスタやキャッシュなどの高速な記憶装置は容量が小さい割にコストが高い．一方，メインメモリやハードディスクなどはこれらに比べると速度は劣るものの容量は大きくコストは安くつく．キャッシュやメインメモリはそれぞれ間を挟む記憶装置の緩和に役立っている．これらを踏まえて総合的にコスト，スペックのバランスを保つためにこのようになっていると考えられる．

# 6 感想

とうとう，実験 2 が始まりました．毎回思う事ですが，どうして自分は早めに始めるという事が出来ないのでしょうか？早めに終らせれば週末をしっかり過ごす事が出来るのに … 。と、バカな愚痴はここまでにして．簡潔な感想としては面白かったです本実験、KUE-CHIP2 といういわば小さなパソコンの扱いに慣れると言った物でした．はじめはさっぱりでしたが使ってみると少しずつわかってきて最後の課題のときは結構熱中していました．次回も楽しめたらいいと思います．次のレポートは早めに仕上げて手直しする余裕ができればいいと思います．

# 参考文献

[1] 基本情報技術者合格教本，技術評論社

[2] サクセスガイドハードウェア，一橋出版，著者：安藤 明之

[3] http://e-words.jp/ ”IT 用語辞典 e-Words”

[4] KUE-CHIP2 教育用ボードリファレンスマニュアル (抜粋)