情報工学実験２
命令実行フェーズ

学籍番号 045718D：翁長絵美

実地日：平成 17 年 10 月 31 日
提出日：平成 17 年 11 月 7 日

共同実験者
045751F：又吉　美奈

1　実験目的

機械語 (マシン語) 命令をフェーズ毎に実行させ、そのときのコンピュータ内部の状態を観測することにより、各フェーズでどのような処理が行われているかを調査し、機械語命令の実行の仕組みを理解することを目的とする。

2　実験概要

上記の実験目的を達成するため、まず実験 (1)〜(3) で計７つのプログラムを使って実際に１フェーズずつ KUE-CHIP2 を実行していく。そのときのプログラムカウンタや、アキュムレータなどが各フェーズごとにどのような働きをし、処理をしているのかを観測することにより機械語命令の実行の仕組みを理解する。また、実験 (4) で商 m ÷ n を求めるアセンブラプログラムを作り実行させることにより、より理解を深める。

3　実験結果

3.1　以下に示すプログラムを用いて、減算命令 (SUB) の実行フェーズを観測せよ。ただし、ACC には、予め 07H を格納しておき、(7-5) の計算過程を観測するものとする。

番地	機械語	アセンブラ言語	
00	00	NOP	← 3 フェーズ
01	A2	SUB ACC,05H	← 4 フェーズ
02	05		
03	0F	HLT	← 3 フェーズ

上記のアセンブラプログラムを実際に KUE-CHIP2 に入力し、実行フェーズを観測するため SP スイッチを押して SUB 命令の実行の様子をみた。SUB 命令の実行フェーズの様子を表１に示す。

表 1.SUB 命令の実行フェーズ表

フェーズ	LED		PC	FLAG	ACC	IX	DBi	DBo	MAR	IR
実行直前	P0 点灯		01	00	07	00	00	00	00	00
P0 実行後	P1 点灯		02	00	07	00	A2	A2	01	00
P1 実行後	P2 点灯		02	00	07	00	A2	A2	01	A2
P2 実行後	P3 点灯		03	00	07	00	05	02	02	A2
P3 実行後	P0 点灯		03	00	02	00	05	05	02	A2

3.2 下記の４つのプログラムについて、それぞれ２番目の命令の実行フェーズを観測し、表 2〜5 の実行フェーズ表を完成させよ。また、それぞれのアセンブラプログラムに対応する機械語プログラムを示せ。

1	アセンブラ言語
	NOP
	LD ACC,05H
	HLT

2	アセンブラ言語
	NOP
	LD ACC,[07H]
	HLT

3	アセンブラ言語
	NOP
	SCF
	HLT

4	アセンブラ言語
	NOP
	ADD ACC,05H
	HLT

上記のアセンブラプログラムに対応する機械語プログラムを以下に示す。これをもとに実際に KUE-CHIP2 に入力、実行しそれぞれ 2 番目の命令の実行フェーズを観測し、表 2〜5 の実行フェーズ表を完成させた。

1	番地	機械語
	00	00
	01	62
	02	05
	03	0F

2	番地	機械語
	00	00
	01	64
	02	07
	03	0F

3	番地	機械語
	00	00
	01	2F
	02	0F

4	番地	機械語
	00	00
	01	F2
	02	05
	03	0F

表 2.LD 命令 (即値アドレスモード) の実行フェーズ表 (プログラム 1)

フェーズ	LED		PC	FLAG	ACC	IX	DBi	DBo	MAR	IR
実行直前	P0 点灯		01	00	00	00	00	00	00	00
P0 実行後	P1 点灯		02	00	00	00	62	62	01	00
P1 実行後	P2 点灯		02	00	00	00	62	62	01	62
P2 実行後	P3 点灯		03	00	00	00	05	05	02	62
P3 実行後	P0 点灯		03	00	05	00	05	05	02	62

表 3.LD 命令 (絶対アドレスモード) の実行フェーズ表 (プログラム 2)

フェーズ	LED		PC	FLAG	ACC	IX	DBi	DBo	MAR	IR
実行直前	P0 点灯		01	00	00	00	00	00	00	00
P0 実行後	P1 点灯		02	00	00	00	64	64	01	00
P1 実行後	P2 点灯		02	00	00	00	64	64	01	64
P2 実行後	P3 点灯		03	00	00	00	07	07	02	64
P3 実行後	P4 点灯		03	00	00	00	FF	FF	07	64
P4 実行後	P0 点灯		03	00	FF	00	FF	FF	07	64

表 4.SCF 命令の実行フェーズ表 (プログラム 3)

フェーズ	LED		PC	FLAG	ACC	IX	DBi	DBo	MAR	IR
実行直前	P0 点灯		01	00	00	00	00	00	00	00
P0 実行後	P1 点灯		02	00	00	00	2F	2F	01	00
P1 実行後	P2 点灯		02	00	00	00	2F	2F	01	2F
P2 実行後	P0 点灯		02	08	00	00	2F	2F	01	2F

表 5.AND 命令の実行フェーズ表 (プログラム 4)

フェーズ	LED		PC	FLAG	ACC	IX	DBi	DBo	MAR	IR
実行直前	P0 点灯		01	00	00	00	00	00	00	00
P0 実行後	P1 点灯		02	00	00	00	F2	F2	01	00
P1 実行後	P2 点灯		02	00	00	00	F2	F2	01	F2
P2 実行後	P3 点灯		03	00	00	00	05	Fb	02	F2
P3 実行後	P0 点灯		03	02	00	00	05	05	02	F2

3.3 下記のプログラムにおいて、IX=2 とした場合および IX=1 とした場合のそれぞれについて、BZ 命令の実行フェーズを観測し、表 6,7 の実行フェーズ表を完成させよ。なお下記のプログラムにおいて、jp はラベルであり、BZ 命令の分岐先のアドレスを表すものとする。機械語プログラム作成時に、各自で適当な値を jp に設定すること。

アセンブラ言語
SUB IX,01H
BZ jp
HLT
jp: HLT

上記のアセンブラプログラムに対応する機械語プログラムを作り、入力・実行した。以下は、実際に入力した機械語プログラムと IX にぞれぞれ 2 と 1 を入力し BZ 命令の実行フェーズを観測した結果である。(表 6,7)

番地	機械語
00	AA
01	01
02	39
03	05
04	0F
05	0F

表 6.BZ 命令 (分岐条件不成立時) の実行フェーズ表 (IX=2)

フェーズ	LED		PC	FLAG	ACC	IX	DBi	DBo	MAR	IR
実行直前	P0 点灯		02	00	00	01	01	01	01	AA
P0 実行後	P1 点灯		03	00	00	01	39	39	02	AA
P1 実行後	P2 点灯		03	00	00	01	39	39	02	39
P2 実行後	P3 点灯		04	00	00	01	05	05	03	39
P3 実行後	P0 点灯		04	00	00	01	05	05	03	39

表 7.BZ 命令 (分岐条件成立時) の実行フェーズ表 (IX=1)

フェーズ	LED		PC	FLAG	ACC	IX	DBi	DBo	MAR	IR
実行直前	P0 点灯		02	01	00	00	01	01	01	AA
P0 実行後	P1 点灯		03	01	00	00	39	39	02	AA
P1 実行後	P2 点灯		03	01	00	00	39	39	02	39
P2 実行後	P3 点灯		04	01	00	00	05	05	03	39
P3 実行後	P0 点灯		05	01	00	00	05	05	03	39

3.4 8ビットの2進数m(データ領域の0x00番地に格納)、n(データ領域の0x01番地に格納) に対し、商 m ÷ n(少数点以下は不要) を求めるアセンブラプログラムを作成し、下記の場合の動作を確認しなさい。ただし、n=0 のときの商を0xFF とすること。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。

(a) m>n で、n が m を割り切れる場合の動作

(b) m>n で、n が m を割り切れない場合の動作

(c) m<n の場合の動作

(d) m=n の場合の動作

(e) n=0 の場合の動作

IX を m、ACC を n としてフローチャートをまず作った。(図１)

最初に IX,ACC を 0 にして初期化を行い、そのあとに ACC に (01H) 番地の値を入れる。これはまず、n=0 かどうかを調べるためであり、分岐を使ってもし ACC が０なら n=0 ということで結果の値を FF で返して終了する。n=0 でなければそのまま次に進み、IX つまり m に (00H) 番地の値を入れる。次の過程で ACC を０にしている理由は、ACC に m ÷ n の答えをいれようとしたためである。(01H) 番地に n の値が入っているので、ACC の値を 0 にしても支障はない。次に IX=IX-(01H) として実際には除算ではなく減算をし、減算したら ACC に 1 ずつ値を入れていく。$IX \geq 0$ になるまでこれを繰り返す。分岐の前に IX=IX+0 を入れたのは、分岐命令は直前の演算の結果を処理するので、そのための帳じり合わせである。$IX \leq 0$ になると ACC の値を-1 して終了する。これは、マイナスになったときも ACC の値に+1 されるため、その余分な１を引いている。

図１のフローチャートをもとに以下に m ÷ n のアセンブラプログラムを作った。

番地	機械語	アセンブラ言語
00	C9	EOR IX,IX
01	C0	EOR ACC,ACC
02	95	ADD ACC,(01H)
03	01	
04	39	BZ
05	15	
06	6D	LD IX,(00H)
07	00	
08	C0	EOR ACC,ACC
09	AD	SUB IX,(01H)
0A	01	
0B	B2	ADD ACC,01H
0C	01	
0D	BA	ADD IX,00H
0E	00	
0F	32	BZP
10	09	
11	A2	SUB ACC,01H
12	01	
13	30	BA
14	17	
15	B2	ADD ACC,FFH
16	FF	
17	0F	HLT

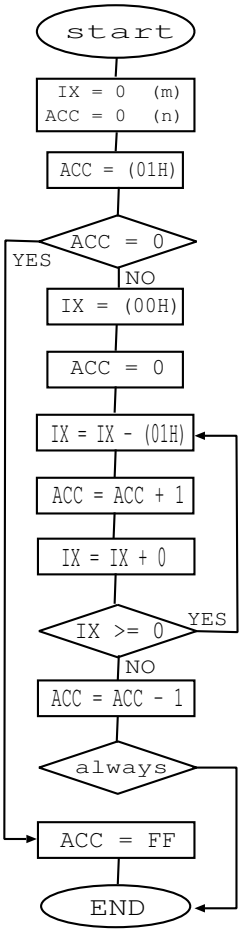

図 1: m ÷ n のフローチャート

IX(m) には初期値として 0A を入れ、ACC(n) にはそれぞれ (a)〜(e) の順番に 02,03,0F,0A,00 を入れて実行してみた。結果は

	m	n	結果
(a)	0A	02	05
(b)	0A	03	01
(c)	0A	0f	00
(d)	0A	0A	01
(e)	0A	00	FF

というように正しく表示された。
このアセンブラプログラムと同様の動作をするC言語のプログラムを作った。

```
#include<stdio.h>
main()
{ int m,n;
  int kotae = 0;
  char linem[100];   /* m の入力データ */
  char linen[100];   /* n の入力データ */

  printf("m の値を入力して下さい。 m = ");
  fgets(linem, sizeof(linem), stdin);
  sscanf(linem, "%d", &m);

  printf("n の値を入力して下さい。 n = ");
  fgets(linen, sizeof(linen), stdin);
  sscanf(linen,"%d", &n);

if(n == 0)                    /* n=0 の時の処理 */
  printf("m / n = FF \n");

else{
while(m >= 0){
  m = m - n;
  kotae = kotae + 1;
}
      kotae = kotae - 1;
      printf("m / n = %d\n",kotae);
}
}
```

実行結果は以下のようになり、アセンブラプログラムと同じ動作をしていることが確認取れた。

```
[Emi-ONAGA:~/kadai/experiment2/report3] j04018% ./warizan
m の値を入力して下さい。 m = 10
n の値を入力して下さい。 n = 2
m / n = 5
[Emi-ONAGA:~/kadai/experiment2/report3] j04018% ./warizan
m の値を入力して下さい。 m = 10
n の値を入力して下さい。 n = 3
m / n = 3
[Emi-ONAGA:~/kadai/experiment2/report3] j04018% ./warizan
m の値を入力して下さい。 m = 10
n の値を入力して下さい。 n = 16
m / n = 0
[Emi-ONAGA:~/kadai/experiment2/report3] j04018% ./warizan
m の値を入力して下さい。 m = 10
n の値を入力して下さい。 n = 10
m / n = 1
[Emi-ONAGA:~/kadai/experiment2/report3] j04018% ./warizan
m の値を入力して下さい。 m = 10
n の値を入力して下さい。 n = 0
m / n = FF
```

C 言語では好きな値を入れるために fgets 関数を使い、sscanf 関数を使って数値に変換した。if 文を使い、n = 0 のときは、FF を表示するようにし、n = 0 でないときは while 文を使って、$m \geq 0$ の間 m - n を繰り返させ、その度 kotae も 1 ずつ足していった。$m \leq 0$ になるとループを抜けて、kotae の余分な 1 を引いたあと kakoe の値を出力させるようにした。図 2 にフローチャートを示す。

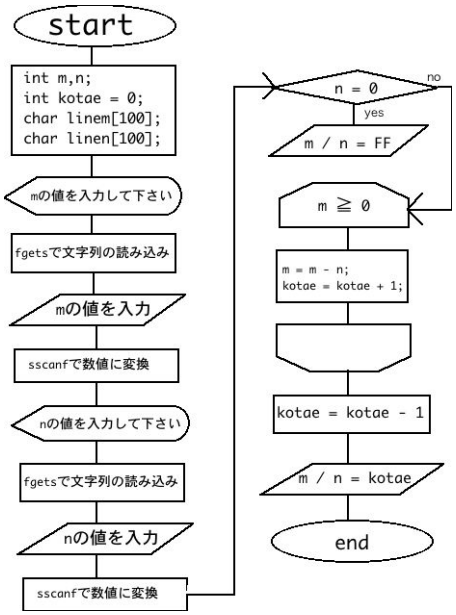

図 2: m ÷ n の C 言語プログラムのフローチャート

4 考察

- 実験 (1) について

実験 (1) では SUB 命令の実行フェーズを観測した。SUB 命令は 4 フェーズである。

P0　PC の値を MAR に格納し、PC を 1 足す。
P1　命令の内容をデータバスを使って IR に格納する。
P2　P0 と同様に PC の値を MAR に格納し、PC を足している。また、DBi(インプット・データバス) で 07 を読み込み、DBo(アウトプット・データバス) で 02 になっていることから、P2 実行の時点で減算がなされたと考えられる。
P3　ACC に値が表示され SUB 命令は終了する。

- 実験 (2) について

実験 (2) での 4 つのプログラムのそれぞれ 2 番目の命令の実行フェーズについて考える。

まず、１つめのプログラムの LD ACC, 05H について

P0　　PC の値を MAR に格納し、PC に 1 を足す。
P1　　命令を DBi、DBo を通して IR に格納する。
P2　　IR に命令が入ったので PC が指している 02 に行く
P3　　DBi,DBo を通して、ACC に 05 が格納された。

２つめのプログラムの LD ACC,[07H] について

P0　　PC の値を MAR に格納し、PC に 1 を足す
P1　　命令を DBi,DBo を通して IR に格納する。
P2　　命令が実行され、DBi,DBo に 07 が入る
P3　　MAR に 07 が格納され、その値が DBi に入る。
P4　　ここで、DBo を通して ACC に値が格納される

３つめのプログラム SCF について

P0　　PC の値を MAR に格納し、PC に 1 を足す
P1　　命令を DBi,DBo を通して IR に格納する
P2　　P1 で IR に格納された命令を実行。FLAG に 08 を入れる。

4 つめのプログラム AND ACC,05H について

P0　　PC の値を MAR に格納し、PC に 1 を足す
P1　　命令を DBi,DBo を通し、IR に格納
P2　　PC の値を MAR に格納し、PC に 1 を足す。IR を読み込み DBi を通して 05 を持ってくる。
P3　　DBo が 05 となっていることから AND 命令は終了したと考えられる。

- 実験 (3) について

表６:BZ 命令 (分岐条件不成立時) について
実行直前　　IX、DBi、DBo には前命令の SUB で使われた 01 が入っている
　　P0　　PC の値を MAR に格納し、PC に１を足す。また、PC に示

してあった 02 の番地の命令を DBi に格納

P1　　DBi を通して IR に命令を格納する

P2　　PC に 1 を足して、ここで分岐の処理をする。

P3　　FLAG が 00 なのでそのまま PC は 04 を指す。

表 7 :BZ 命令 (分岐条件成立時) について

実行直前　　SUB 命令の実行が終わり、FLAG には 01 が格納される。

P0　　PC の値を MAR に格納し、PC に 1 を足す。また、PC に示してあった 02 番地の命令を DBi,DBo に格納

P1　　DBi、DBo を通して IR に命令を格納

P2　　PC に 1 を足す。また、DBi,DBo の値が 05 になっていることから、ここで分岐の決定がなされていることがわかる。

P3　　FLAG が 01 なので PC は 05 になる。

これら表 1〜表 7 を見ると、KUE-CHIP2 のフェーズの分け方は妥当ではないと考える。その理由は、P0 実行後の処理が無駄に思われるためである。解決策としては、P0 実行後 IR に命令の値がくるようにすることだと思う。

- 実験 (4) について

アセンブラプログラムと C 言語プログラムの違いは、分岐の使い方だと思われる。アセンブラプログラムには while 文のようなものはなく、if 文のような働きしかなく、if 文で繰り返し処理を行っている。その点、C 言語プログラムは様々な処理が行える。

5　調査課題

下記の各項目について調査し、その結果を報告せよ。

(a) プロセッサ (CPU) の性能を表す指標に関して、以下の設問に答えよ。

① IPC とはどのような指標か調査し、説明せよ。また、IPC の他にも、プロセッサ (CPU) の性能を表す指標はたくさんある。IPC 以外の指標について調査し、５つ以上挙げて、それぞれの指標について詳しく説明せよ。

IPC とは、１クロックあたりに実行可能な命令数のことで、「実行命令数 ÷ 所用クロック数」で計算される。CPU の処理性能は CPU の処理性能=動作周波数× IPC で表される。

IPC 以外に指標は、MIPS(Mega Instructions Per Second)、MFLOPS(Mega FLOating point operations Per Second)、サイクルタイム (Cycle time)、ベースクロック (FSB)、周波数 (Hz) などが挙げられる。

MIPS：CPU が 1 秒間に実行可能な命令数を 100 万を単位に表したもの。1 MIPS は 1 秒間に 100 万命令を実行する。

MFOPS：CPU が 1 秒間に実行できる浮動小数点演算の回数を表したもの。スーパーコンピュータなどで用いられている。

サイクルタイム：CPU の命令実行よりも読み書きのためのデータの出し入れの時間を指標にする。CPU が命令を出してから次の読み取り命令が出せるまでの時間を表す。

ベースクロック：CPU とメモリがデータ通信を行う部分で、その速度を周波数で表す。数値が高いほど性能は高くなる。

周波数：CPU が 1 秒間に何回計算を行うかを表す。単位は主に'MHz'や'GHz' であり、値が大きいほど性能は高くなるが、発熱量も　大きくなる。

② IPC が 1 の CPU を載せたコンピュータ A と IPC が 2 の CPU を載せたコンピュータ B があり、両方のコンピュータで同じプログラムを同時に実行した。その結果、コンピュータ B の方が IPC が大きいにも関わらず、コンピュータ A の方が先に処理を終了した。この理由について考察せよ。

コンピュータの CPU の処理性能は動作周波数と IPC で表され、CPU の処理性能 = 動作周波数 × IPC という式が成り立つ。よってこの場合、コンピュータ B よりもコンピュータ A の方が動作周波数が大きかったため、このこうな結果になったと考えられる。

(b) パイプライン・アーキテクチャとはどのようなアーキテクチャか調査し、図表等を用いて分かりやすく説明せよ。また、パイプライン・アーキテクチャを採用した場合の利点と欠点についても詳しく説明せよ。

各命令処理は次の４つのステップを踏んで実行される。

1	命令フェッチ (F)
2	命令デコード (D)
3	演算実行 (E)
4	結果の格納 (W)

命令は通常この４つのステップを完全に終了しないと次の命令を実行することはできない。そこで効率をあげるため、この４つの各ステップをステージとして各工程にわける。処理後で使われていない工程を次の命令にあてることですべての工程を同時に実行することができ処理を早くすることが可能になる。図３はパイプラインで行われる処理のイメージである。

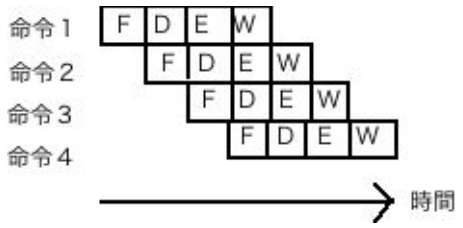

図 3: パイプラインのイメージ

実際のハードウェアでは、ステージの間がただの結線だと、前のステージの処理の影響が後のステージの処理に及ぶため、正しいパイプライン動作が行われないことがわかる。これを防ぐために各ステージの間にレジスタを設ければよい。

6　感想

もう少し余裕を持ってレポートを書けば良かった。理由は、調査課題をもっとちゃんと調べたかったからである。第１回、２回、３回とアセンブラプログラムを勉強して、フローチャートの便利さや機械語の作り方 (考え方) をより理解することができて良かったと思う。

参考文献

[1] tex の書き方
http://oku.edu.mie-u.ac.jp/　okumura/texwiki/?PDF　の 作 り 方
http://www002.upp.so-net.ne.jp/latex/

[2] IPC について
http://originalpc.hp.infoseek.co.jp/cpu.html
http://www.geocities.jp/mickey_son/hardware/processor/processor.htm
2003 年版　基本情報技術者　実践問題　　加藤昭:著　オーム社

[3] パイプライン・アーキテクチャについて
コンピュータアーキテクチャ　　坂井 修一:著　コロナ社