情報工学実験２
アセンブラプログラミング

学籍番号 045718D：翁長絵美

実地日：平成 17 年 10 月 24 日
提出日：平成 17 年 10 月 31 日

共同実験者
045751F：又吉　美奈

1　実験目的

アセンブラプログラムを実際に作成し、それをハンドアセンブル (アセンブラプログラムを人手で機械語プログラムに直すこと) し、さらに KUE-CHIP2 上で実行することを通して、アセンブラプログラミングおよび機械語 (マシン語) プログラムについて理解することを目的とする。

2　実験概要

上記の実験目標を達成するために、KUE-CHIP2 とオシロスコープを用いて、実際に作成したプログラムの出力が、オシロスコープで矩形波、山形波、菱形波になるようにするという実験を行った。また、KUE-CHIP2 の出力ポートに D-A コンバータを使い波形を見た。

3　実験結果

3.1　図 2.2 の D-A コンバータをブレッドボード上に実現し、KUE-CHIP2 の出力ポートに接続せよ。また、リスト 2.1 のプログラムを KUE-CHIP2 に入力し、D-A コンバータのアナログ出力端子から図 2.3 に示したようなのこぎり波が出力されることを、オシロスコープを用いて確認せよ。

本実験の結果に関しては、報告を省略する。

3.2　図 2.4(a)〜(c) に示した各波形を出力するアセンブラプログラムを作成し、KUE-CHIP2 上で実行しなさい。なお、オシロスコープの設定および KUE-CHIP2 のクロック設定は、(1) ののこぎり波を表示させたときのままとし、変更しないこと。

まず、(a) の矩形波を出力するアセンブラプログラムを作る。
矩形波の特徴は、ある一定の時間同じ値を示し、時間が過ぎたら出力が 0 になる。そしてまた、一定の時間が来たらある一定の値を示す … というものである。私たちはプログラムを作る前にフローチャート (図 1) を考えてみた。

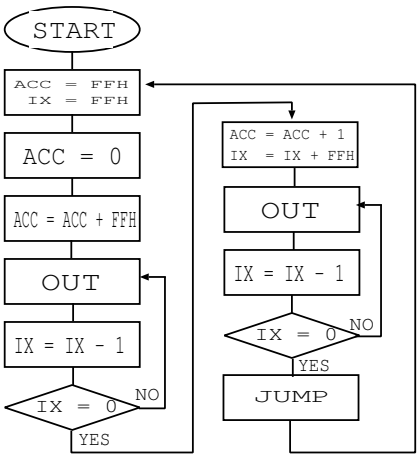

図 1: 矩形波のフローチャート

まず初期値として、ACC,IX に FF つまり $255_{(10)}$ を入れる。ACC を 0 にしてそこにダイレクトに FF($255_{(10)}$) を入れる。ACC を 0 にわざわざしたのは初期化を行うためである。それを表示すると、値は 255 が出てくる。それを IX を 1 ずつ減らし、IX が 0 になるまで繰り返すことによって、一定の時間同じ値が表示されることを実現した。IX が 0 になると、今度は ACC に 1 を足し、0 になった IX に再び FF($255_{(10)}$) を入れる。ACC に 1 を足すと桁が繰り上がってまた 0 になるためである。先ほどと同様に IX=0 になるまで繰り返して表示すると、一定の時間 (IX=O になるまでの時間)0 が出力される。IX=O になると、最初に戻るという考えである。

これをもとに、矩形波のアセンブラプログラムを作ってみた。(表 1) 結果は矩形波になった。(図 1)

表 1 :矩形波アセンブラプログラム

ADRS	DATA	OPECODE
00	62	jp:LD ACC,FFH
01	FF	
02	6A	LD IX,FFH
03	FF	
04	C0	EOR ACC,ACC
05	B2	ADD ACC,FFH
06	FF	
07	10	OUT
08	AA	SUB IX,01H
09	01	
0A	31	BNZ 07H
0B	07	
0C	B2	ADD ACC,01H
0D	01	
0E	6A	LD IX,FFH
0F	FF	
10	10	OUT
11	AA	SUB IX,01H
12	01	
13	31	BNZ 10H
14	10	
15	30	BA JP
16	00	

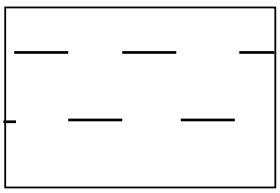

図 2: 矩形波の実行結果

次に (b) 山形波のアセンブラプログラムについて考えてみる。
山形波は 0 から順に 1 ずつ足されていき、最大値である $FF255_{(10)}$ まできたら、今度は順に 1 ずつ引かれているのが特徴である。実験 (1)、(2) の (a) 矩形波を応用してフローチャートを考えた。(図 3)

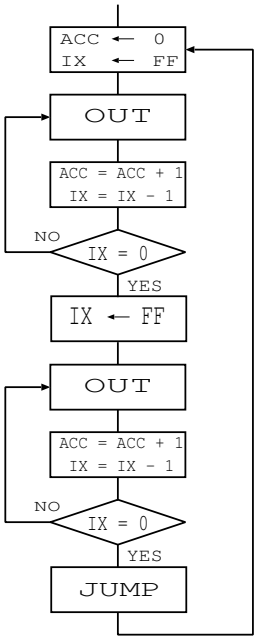

図 3: 山形波のフローチャート

まず初めに ACC=0,IX=FF と初期値を入れる。ACC を分岐を使って IX=0 になるまで 1 ずつ足していきながら表示することを繰り返す。IX=0 になったら、(a) 矩形型と同様に再び IX=FF にし、今度は ACC の値を 1 ずつ減らして行くようにする。IX=0 になったら初めに戻ることを繰り返せば山形波になる。　この考えのもと山形波のアセンブラプログラムを作った。(表 2) 結果はきちんと山形になった。(図 3)

表２:山形波アセンブラプログラム

ADRS	DATA	OPECODE
00	62	jp:LD ACC,00H
01	00	
02	6A	LD IX,FFH
03	FF	
04	10	OUT
05	B2	ADD ACC,01H
06	01	
07	AA	SUB IX,01H
08	01	
09	31	BNZ 04
0A	04	
0B	6A	LD IX,FFH
0C	FF	
0D	10	OUT
0E	A2	SUB ACC,01H
0F	01	
10	AA	SUB IX,01H
11	01	
12	31	BNZ 0D
13	0D	
14	30	BA
15	00	

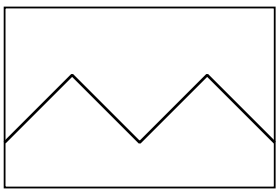

図 4: 山形波の実行結果

最後に (c) 菱形を出力するようなアセンブラプログラムについて考える。菱形の特徴は、出力の関係が $0 \rightarrow 255 \rightarrow 1 \rightarrow 254 \rightarrow 2 \rightarrow 253 \cdots$ と反転しつつ１ずつ数字が変わって行くことである。このことを踏まえ、菱形のフローチャート (図５) を作ってみた。

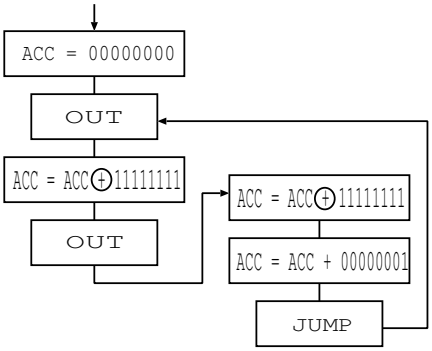

図 5: 菱形波のフローチャート

わかりやすいように値を 2 進数の 8 ビットで表す。
まず ACC を初期化し、表示する。現在 ACC は 00000000 である。次に 255 つまり 11111111 を表示させるためには EOR を使い、値を反転、表示させればよい。次は、00000001 の表示をさせたいので、また EOR を使って現在 11111111 になっている ACC に 11111111 と比較させて 00000000 に反転して 1 を足し、表示すればよい。これを繰り返せば菱形の波形ができるはずである。フローチャートをもとに実際にアセンブラプログラム (表３) を作り、実行してみた。結果は菱形波になった。(図６)

表３:菱形波アセンブラプログラム

ADRS	DATA	OPECODE
00	C0	EOR ACC,ACC
01	10	OUT
02	C2	EOR ACC,FFH
03	FF	
04	10	OUT
05	C2	EOR ACC,FFH
06	FF	
07	B2	ADD ACC,01H
08	01	
09	30	BA
0A	01	

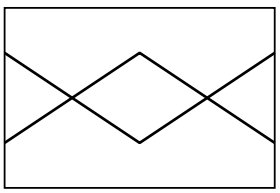

図 6: 菱形波の実行結果

4 考察

(1) 実験 (2) について

波形を出力するために D-A コンバータを用いた。D-A コンバータはデジタル信号をアナログ信号に変える回路である。ディジタル入力端子から'0' または'1' が入ってくると、その 2 進数をしての値に応じた電圧を出力する。

D-A コンバータは出力が 1 つのため、同じ時間に 2 つの出力を表すことが出来ない。そのため、今回の方法によって大抵の波形は作れるが'正菱形' など正確な対称となる波形は作ることが出来ない。

(2) 本実験の考察について

今回の実験で言語プロセッサの種類や働きについて新たな知見を得られた。言語プロセッサについては、調査課題の (a) で記す。

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) 言語プロセッサには、アセンブラやコンパイラ、インタプリタとジェネレータがある。これら 4 種類の言語プロセッサにおいて行われる処理の特徴や違いについて図表等を用いて詳しく説明し、それぞれの言語プロセッサで処理される代表的な言語を挙げよ。

① アセンブラ

機械語と密接な関係にあるアセンブラ言語で書かれた原始プログラムを機械語に翻訳する。代表的な言語はアセンブラ言語。

② コンパイラ

高水準言語で書かれた原始プログラムを、機械語あるいは目的プログラムにまとめて翻訳する。代表的な言語は C,COBOL など。

③ ジェネレータ

パラメータを受け取って、処理目的に応じた機械語のプログラムを生成する。

④ インタプリタ

高水準言語で書かれた原始プログラムを、1 命令ずつ翻訳していきながら実行する。制作途中のプログラムでも出来ている部分だけを実行することが出来る。しかし、実行速度はコンパイラよりも遅い。代表的な言語は BASIC,LISP など。

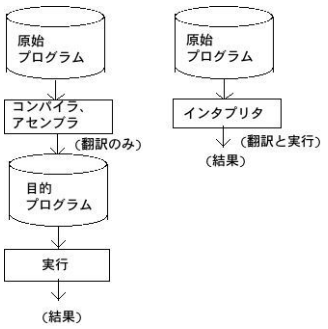

図 7: 翻訳プログラムと解釈実行プログラムのイメージ

(b) 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。

① アキュムレータ (ACC)

演算の際の被演算数を保持し、演算後は演算結果が格納されるレジスタ。

② インデックスレジスタ (IX)

メモリをアクセスする場合のアドレスを指定するときに用いるレジスタであり、この内容をアドレスバスに出すことにより、メインメモリからデータを読み出す。IX はアドレス計算用の加算回路がついており、実行アドレスの計算を行うことができる。

③ プログラムカウンタ (PC)

１つの命令が実行されたあと、次に実行されるべき命令のあるアドレスを記憶しておくレジスタ。１つの命令が実行される度に自動的にアドレスに「1」が加えられ、次に取り出すべきアドレスを保持する。分岐命令は、このプログラムカウンタに値を代入することで実現される。

④ メモリアドレスレジスタ (MAR)

データを書き込みや読み取りのためのアドレスを記憶する。

⑤ 命令レジスタ (IR)

記憶装置から取り出された命令を受け取り、それを実行するために一時記憶しておくレジスタ。

⑥ フラグレジスタ (FR)

フラグレジスタは演算命令を実行したときの実行結果の状態を保持する状態レジスタと CPU 自身の状態を示す制御レジスタに大

別できる。AレジスタとBレジスタの値を比較したときに，どちらの レジスタの値が大きいかなどの結果を覚えておくレジスタ。

⑦ スタックポインタ (SP)

　レジスタの値を一時的に保存したり、サブルーチンコールなどの戻りアドレスを保存するために使われるメモリ領域のアドレスを示すカウンタ。

⑧ 汎用レジスタ

　データの格納に用いるのが汎用レジスタであり、機能が限定されていないレジスタ。汎用レジスタにはアキュムレータ、ベース、カウンタ、データがあり、基本的に何に用いてもかまわないが、ベースにはメモリのアドレスを示すための領域などとされている。

この８つのうち KUE-CHIP2 にないものは、スタックポインタである。

6　感想

　今回の実験でフローチャートのすばらしさを実感した。
また、漠然としていた各レジスタの役割などが調査をしていくうちにわかってきたのでとても勉強になった。

参考文献

[1] CPU について
基本情報技術者試験　サクセスガイド　ハードウェア：安藤明之　一橋出版 http://ja.wikipedia.org/wiki/プログラムカウンタ

http://www.arch.cs.kumamoto-u.ac.jp/project/kite/kite2/regset.html#FR
http://wisdom.sakura.ne.jp/programming/asm/index.html