アセンブラプログラミング

学籍番号 045713C:大城和也

実験実施日:平成 17 年 10 月 24 日 (月)
提出日:平成 17 年 10 月 31 日 (月)

共同実験者
045709E：上原直久
045736B：知念栄作
045739G：友寄雄一朗

1　実験目的

アセンブラプログラムを実際に作成し，それえ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 上で実行する事を通して，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事を目的とする．

2　概要

本実験はまずブレッドボード上に D-A コンバータを作成し，KUE-CHIP2 に繋げて例として与えられたプログラムを入力し，オシロスコープでのこぎり派となっていることを確認する．次に提示された波形のでるアセンブラプログラムを作成する．提示されたプログラムとは矩形派，山形派，菱形波がでるものである．アセンブラプログラムを作った後には対応した機械語に直して KUE-CHIP2 に入力しその D-A コンバータに繋げその派系を確認する．なお，アセンブラプログラムを考え，機械語に直す際には KUE-CHIP2 教育用ボードリファレンスマニュアルの第 7 章を参考にするとよい．

3　実験結果

3.1　図 2.2(実験指導書，参照) 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事を，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)(実験指導書，参照) 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

前置きとして図 2，4，6 のオシロスコープの設定は cl=0.1v，MHG×10，A=10ms となっている (縦の 1 マスは 1[V] を，横の 1 マスは 10[ms] を表している)

3.2.1 矩形波プログラム

```
-----------------------------------------------------------------------
番地   機械語   アセンブラ言語       コメント
 00     C9       EOR IX, IX         IX の初期化
 01     C0       EOR ACC, ACC       ACC の初期化
 02     10       OUT                値の出力
 03     BA       ADD IX, 01H        IX に 1 を加える
 04     01
 05     31       BNZ 02H            IX の値が 0 でない場合に 02 番地へ
 06     02
 07     C2       EOR ACC,FFH        ACC と FF との排他的論理和
 08     FF
 09     30       BA 02H             常に 02 番地へ飛ぶ
 10     02
-----------------------------------------------------------------------
```

上記は矩形波のアセンブラプログラムである．このプログラムを動かすことで矩形波を出すことができる．流れとして図 1 のフローチャートの説明をする．

1. ACC と IX の初期化を排他的論理和を用いて行う．
2. OUT を用いて ACC の値を OBUF に出力．つまりはオシロスコープに表示．
3. IX の値を 1 つ増やす．
4. 直前の値が 0 でなかったら (ACC の値が 0 でないなら)02 番地に戻る (上記の 2 処理のループ)．0 であった場合はそのまま次の命令に進む．
5. ACC と FF の排他的論理和を取る．$(FF)_{16}$ は $(11111111)_2$ であるのでこれとの排他的論理和を取ることでビットの反転を行っている．
6. BA とは Branch All の略で常に指定された番地に飛ぶようになっている．今回の場合は 02 番地に戻っているため無限ループ状態になる．

今回，一定の時間同じ値を出すように IX を用いて IX の値が 1 つずつ増え，FF を超えて 00 に戻ったとき (オーバーフローしたとき) に，ACC の値をビット反転するようなっている．それにより，図 2 のような一定の幅で 0V と 3V を交互に変わるようになっている．

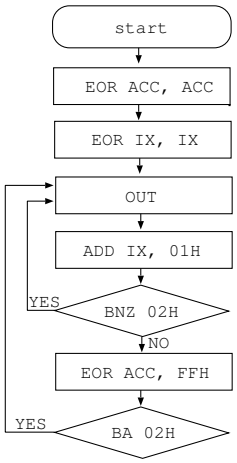

図 1: 矩形波フローチャート

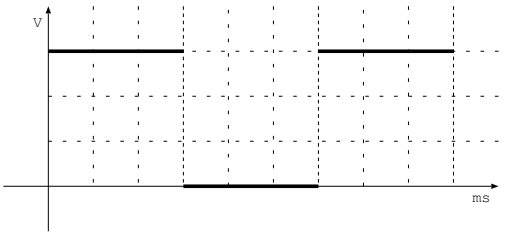

図 2: 矩形波

3.2.2 山形波プログラム

番地	機械語	アセンブラ言語	コメント
00	C0	EOR ACC, ACC	ACC の初期化
01	10	OUT	値の出力
02	B2	ADD ACC, 01H	ACC に 1 を加える
03	01		
04	31	BNZ 01H	ACC が 0 で無い場合 01 番地へ
05	01		
06	A2	SUB ACC, 01H	ACC から 1 を引く
07	01		
08	10	OUT	値の出力
09	31	BNZ 06H	ACC が 0 で無い場合 06 番地へ
10	06		
11	30	BA 02H	常に 02 番地へ飛ぶ
12	02		

上記のアセンブラプログラムは山形派を出すプログラムである．山形派を出すプログラムの考え方としては例題ののこぎり波を初めに行い，次に値が FF まで到達したときに値を減らすようなプログラムにしればいい．

よってプログラムの前半はのこぎり波のプログラムを引用し，ACC の値が $(FF)_{16}$ にまで到達したときに Branch で分岐できるようにし，次は値を減らすものを作成しればよい．よって上記のようなプログラムとなる．

下記にこのプログラムのフローチャート (図 3) の説明をする.

1. まず初めに ACC の初期化を行う．ACC の排他的論理和を扱う.
2. OUT で ACC の値を出力させる.
3. ACC に 1 を足す.
4. 前の ACC の値が 0 でなかったら 02 番地の命令に移り，0 であった場合には次の命令に移る.
5. 先のループを抜けた後，ACC の値を 1 減らす.
6. OUT で ACC の値を出力させる.
7. ACC の値が 0 でなかった場合は 06 番地の命令に移り，0 であった場合には次の命令に移る.
8. BA を使って 02 番地の命令に飛ぶ.

これは前半の OUT から BNZ までで山形の右上がりを，後半の SUB から BNZ までが右下がりを作り出し，それをループさせることで図 4 のような結果を作り出している.

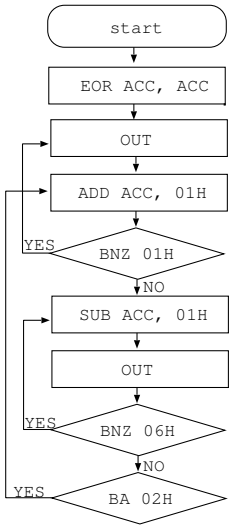

図 3: 山形波フローチャート

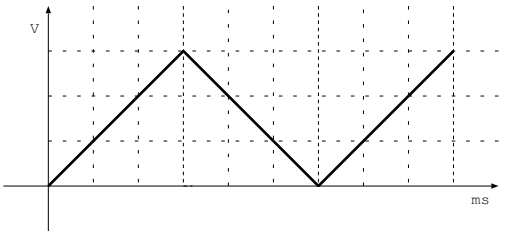

図 4: 山形波

3.2.3 菱形波プログラム

番地	機械語	アセンブラ言語	コメント
00	C0	EOR ACC, ACC	ACC の初期化
01	10	OUT	値を出力
02	B2	ADD ACC, 01H	ACC に 1 を加える
03	01		
04	C2	EOR ACC, FFH	ACC と FF との排他的論理和
05	FF		
06	10	OUT	値を出力
07	C2	EOR ACC, FFH	ACC と FF との排他的論理和
08	FF		
09	30	BA 01H	常に 01 番地へ飛ぶ
10	01		

上記のプログラムは菱形波を出すプログラムである．菱形波を出すプログラムの考え方としては，まず，1 つの座標に 2 つの値を出すことはできないので上と下を交互に描くようにして菱形を出力することを念頭に置く．そして菱形ということは 2 つの座標は線対称ということなので座標は反転することで次の座標が表せることとなる．はじめはそのまま出力し，次は反転して出力，その次の座標はまた反転させて 1 を足して出力すればよく，さらにその次はそれをさらに反転して出力．これを繰り返せば菱形波ができる，それらを考えて上記のプログラムを作成した．

下記にこのプログラムのフローチャート (図 5 参照) の説明をする．

1. まず，ACC の初期化．ACC 自身との排他的論理和をとる．
2. OUT で ACC の値を出力．
3. ACC の値に 1 を加える
4. ACC と $(FF)_{16}$ の値の排他的論理和をとる．これでビットの反転を行っている．
5. OUT で先ほど反転した ACC の値を出力．
6. ACC と $(FF)_{16}$ でさらに反転．
7. BA により 01 番地の命令に戻る．これにより無限ループを行う．

出力の際には図 6 を見てくれれば分かるが，縦のボルトが 3[V] に達するのに今までと比べて 2 倍の時間がかかっている．これは 1 つの値を出したときに

反転して 2 つの出力をして 1 を加えるという動作を行うためであると考えられる．

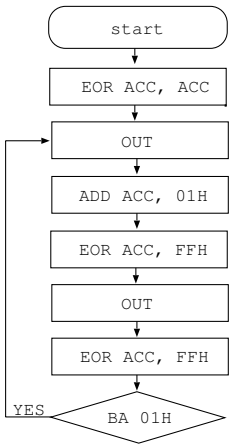

図 5: 菱形波フローチャート

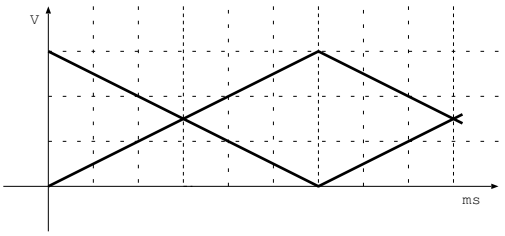

図 6: 菱形波

4 考察

4.1 実験 (2) について

実験 (2) は 3 つの波形を作成する実験であった．実験を行った全部のプログラムの振幅は (約)3.3[V] となった．全てのプログラムが同じ振幅となったのは 3 つのプログラムは全部，値として $(00000000)_2 \sim (11111111)_2$ の値を取るからであり，3.3[V] となっているのは実験指導書にも書いていた通り，D-A コンバータはディジタル入力 $(11111111)_2$ に対して (約)3.3[V] を出力するためだと考えられる．

上記と同じように 3 つの波形は波長が同じなため周波数もほとんど変わらないと思われる．周波数は 1 秒で繰り返す波の数なのでその処理の早さによって変化する．

今回，私のグループが作成したプログラムはどれも基本的には OUT で値が出力されるのには機械語で 4 行かかっている．さらに，どのプログラムも一周期に描写する値は 256 個である．これにより，同じ周波数になっていると考えられる．よって，周波数を他と変えるとするなら機械語にて 8 行で OUT を 1 つにすることで周期は半分に．また，矩形波のプログラムにて IX を設定し，繰り返しを 128 回の半分にすることで 2 倍の周波数がでるであろう．

また，周波数及び波長を大きく変化させるのはクロックである．クロックはそもそも処理を行うためのど話した処理の早さに大きく関わるからである．

4.2 その他

今回，のこぎり波を出力する際に D-A コンバータの最下位ビットにプローブを付けてみた所その波形は小刻みな矩形波となった．さらにその次のビットを表す所にプローブを付けた所その波形は先ほどの 2 倍の波長の矩形波となった．もう 1 つ次のビットはさらにその倍の波長となった．のこぎり波はビット的に考えれば 1 ずつ足されるのでその値は最下位ビットでは 0 と 1 が毎回交互に変わり，一方最上位ビットでは長い時間をかけて 0 と 1 が変わる．今回のこのことにより波形の面でもそのことが分かった．

5 調査課題

5.1 言語プロセッサの種類，その特徴について

言語プロセッサとは簡単に言えばソースを機械語に直す翻訳機のようなものである．言語プロセッサにはアセンブラ，コンパイラ，インタプリタ，ジェネレータなどがある．

5.1.1 アセンブラ (assembler)

アセンブラはアセンブラ言語で書かれたプログラムを翻訳する言語プロセッサである．以下の図 7 はアセンブラの変換の様子を表している．

図 7: アセンブラ

5.1.2 コンパイラ (compiler)

コンパイラは高水準言語 (C，FORTRAN，JAVA，COBOL など) のソースを一括してオブジェクト (機械語) に変換する言語プロセッサである．コンパイラを使う言語のプログラムを実行する際には図 8 の処理の後リンカによってライブラリや別のオブジェクトと連結させたりした後，実行する形となる．

図 8: コンパイラ

5.1.3 インタプリタ (interpreter)

インタプリタはソースコードを必要な所だけ機械語に変換しながら，実行する言語プロセッサである．代表的な言語に Perl や JavaScript などがある．実行時に変換を行うためプログラムの実行速度は遅いがプログラムの誤りなどは見つけやすくプログラム開発に要する時間は短くなる．よって C 言語や BASIC など開発のためにインタプリタとコンパイラも実装している言語もある．

図 9 はインタプリタの変換の様子を表している．

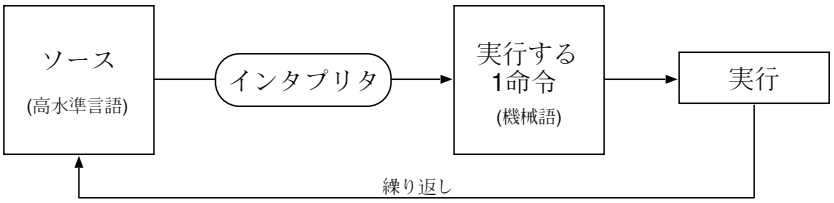

図 9: インタプリタ

5.1.4 ジェネレータ (generator)

ジェネレータはあらかじめプログラムの雛形が作られており，一定のパラメータを与えることにより自動的にオブジェクトに変換する言語プロセッサである．代表的なものに RPG があり，これは報告書作成向けの言語で適切なパラメータを記述することでアルゴリズムを気にせずにアプリケーションが作成が行える．

図 10 はジェネレータの変換の様子を表している．

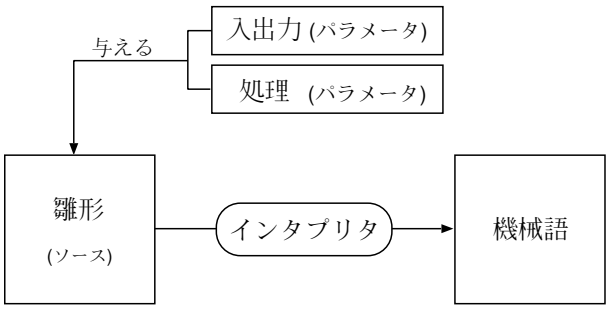

図 10: ジェネレータ

5.2 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内，KUE-CHIP2 に無いものを挙げよ．

(a) アキュムレータ (ACC)

ALU の一部であり，演算結果や，演算の対象となるデータ (オペランド) を一時的に保存するレジスタである．

(b) インデックスレジスタ (IX)

メインメモリのアドレスのオフセットを保持するレジスタである．

(c) プログラムカウンタ (PC)

次に実行するプログラムのアドレスを保持するレジスタである．

(d) メモリアドレスレジスタ (MAR)

それぞれの記憶装置の番地を指定するレジスタである．

(e) 命令レジスタ (IR)

命令レジスタとは主記憶から命令を呼び出した際にその命令を保持するためのレジスタである．

(f) フラグレジスタ (FR)

フラグレジスタには命令実行中の状態を保存するレジスタで，プログラムの分岐命令などに使われる．

(g) スタックポインタ (SP)

スタックポインタは CPU の動作状態を一時保存する場所のアドレスを保持する目的に用いられ，スタックの制御などを行うレジスタである．

(h) 汎用レジスタ

専用の用途が特に決められていないレジスタ．インデックスレジスタやアキュムレータの代用などに使用されることが多い．

これらのレジスタの中で KUE-CHIP2 に無いものはスタックポインタだけであると思われるが IX などは KUE-CHIP2 においては汎用レジスタのような役割のように思われた.

6 感想

今週は結構いっぱいいっぱいでした．前回の反省もかねて早めに始めたんですが OS の課題とも被ったり，従兄弟の結婚式があったりで結局仕上がったのは最終日に … 。まぁ、早めに始めててよかったです．後回しにしてたらきっと終ってませんから (笑)．実験自体の感想としてはなんか久しぶりにプログラムを考えたな〜って感じでしたね．実際，2 年の前期には自分で考えてプログラミングって感じの授業はあんまりありませんでしたから．けど，別にプログラミングに飢えてるってわけじゃないですよ！

参考文献

[1] 基本情報技術者合格教本, 技術評論社

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

[3] http://ja.wikipedia.org/wiki/メインページ “Wikipedeia”