情報工学実験II

実験2-アセンブラプログラミング-

レポート作成者: 055702B　池野谷克俊
共同実験者: 055722G　小林佑亮

提出日 2006年 11月 6日 月曜日

1　実験目的

アセンブラプログラムの作成, 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), 実行の各作業を実際に行うことにより, アセンブラプログラミングの流れを習得する. また, コンパイラとアセンブラの違いや, 高水準言語とアセンブリ言語の違いについて理解することを目的とする.

2　実験概要

まず, 実験をする前に D-A(デジタル-アナログ) コンバータをブレッドボード上に作成し回路がうまく組めているかを確かめた. それから例題として, のこぎり波を出力するプログラムを KUE-CHIP2 で作成し, オシロスコープに波形を出力させた.
ここまで準備してから, 矩形波, 山形波, 菱形波を出力するプログラムを組み, それぞれうまくできているかをオシロスコープに出力させて確かめた.

3　実験結果

- 実験 (1) の結果について

  本実験の結果に関しては, 報告を省略する.

- 実験 (2) の結果について

  (a) 矩形波

  　アセンブラプログラム

アドレス	マシン語	アセンブリ言語
00	6A FFH	LD IX FFH
02	62 FFH	LD ACC FFH
04	10	OUT
05	AA 01H	SUB IX 01H
07	31 04H	BNZ 04H
09	6A FFH	LD IX FFH
0B	62 00H	LD ACC 00H
0D	10	OUT
0E	AA 01H	SUB IX 01H
10	31 0DH	BNZ 0DH
12	30 00H	BA 00H

実行結果

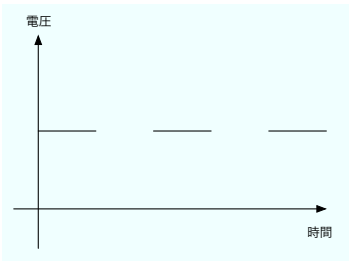


図 1: 矩形波

プログラム動作

IX と ACC に FF を代入し,OUT(オシロスコープに ACC の値を出力) をする. その後,IX をデクリメントして OUT するということを IX が 0 で無い限り繰り返す. この命令によって, しばらく FF が OUT される. それから, 再度 IX に Ff を代入し,ACC に 00 を代入する. その後,IX をデクリメントして OUT するということを IX が 0 で無い限り繰り返す. この命令によって, しばらく 00 が OUT される. 最後にプログラムの最初に無条件でループさせれば, 矩形波ができる.

(b) 山形波

アセンブラプログラム

アドレス	マシン語	アセンブリ言語
00	62 00H	LD ACC 00H
02	10	OUT
03	B2 01H	ADD ACC 01H
05	31 02H	BNZ 02H
07	C2 FFH	EOR ACC FFH
09	10	OUT
0A	A2 01H	SUB ACC 01H
0C	31 09H	BNZ 09H
0E	30 00H	BA 00H

実行結果

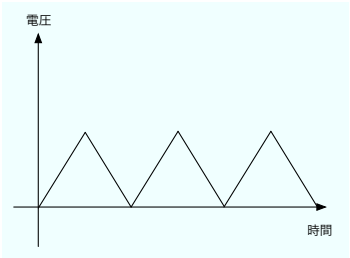


図 2: 山形波

プログラム動作

ACC に 00 を代入して,OUT する. その後,ACC に 1 を足し,ACC が 0 じゃない限り (ACC がオーバーフローして 0 になるまで), アドレス 02H 番地に戻り繰り返す. この命令で 00 から FF までが順番に OUT される. それから,ACC(ここでは 00) と FF を XOR し (ACC を NOT をしたことになる), ACC を FF にしてから,OUT をする. その後,ACC から 1 を引き ACC が 0 じゃない限りアドレス 09H 番地に戻り繰り返す. この命令で FF から 00 までが順番に OUT される. 最後にプリグラムの最初に無条件でループさせれば, 山形波ができる.

(c) 菱形波

アドレス	マシン語	アセンブリ言語
00	62 00H	LD ACC 00H
02	10	OUT
03	C2 FFH	EOR ACC FFH
05	10	OUT
06	C2 FFH	EOR ACC FFH
08	B2 01H	ADD ACC 01H
0A	30 02H	BA 02H

実行結果

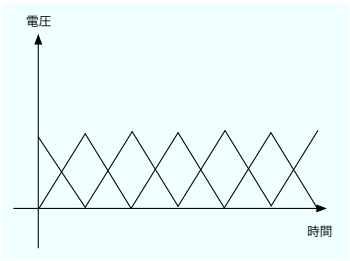


図 3: 菱形波

菱形波

まず,ACC に 00 を代入し OUT をする. 次に ACC と FF を EOR し (ACC を NOT していることになる) OUT をする. さらに再度 ACC と FF を EOR して,ACC に 1 を足す. 最後に無条件でアドレス 02H 番地にループさせれば,00 → FF → 01 → FE → 02 → FD ··· という風に OUT され菱形波ができる.

4 考察

- 実験 (2) の考察について
  - 今回の方法によって出力可能な波形と出力不可能な波形について考察せよ.

    今回の方法では, 出力が 1 つしかないので一度に 2 つ以上の出力が必要な波形は出力不可能である. 今回実験した菱形波は点を微妙にずらして菱形に見せているだけで二点を出力しているわけではない. また, 直線で描かれた波形であれば, 出力可能なものが多い.

- その他の考察について

  – 本実験を通して得られた新たな知見について詳しく説明せよ.

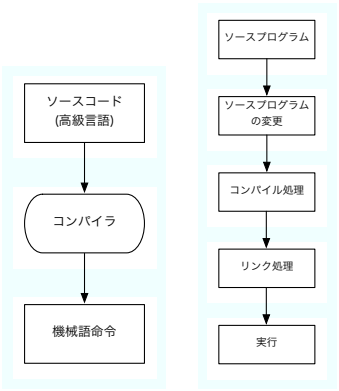
    EOR を用いて,NOT と同じ働きをする命令を作成できる.(今回の場合 FFH と EOR すれば NOT をしていることになる.)
    また, 今回のプログラムで BNZ の部分を BZP にするとうまくいかなかった. BP でもうまくいかなかったことから,Positive の部分がうまく働いていないということが推測されるが, 原因は不明である.

5 調査課題

(a) プログラムは, そのプログラム言語の規則に従って記述される. この段階のプログラムをソースプログラム (原始プログラム) と呼び, ソースプログラムを処理するためのソフトウェアを言語プロセッサ (言語処理系) と呼ぶ. 先に説明したアセンブラやコンパイラは, 言語プロセッサの一種である. 言語プロセッサには, この他, 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.

- コンパイラ

  コンパイラ



  コンパイラ（compiler）とは, プログラミング言語で書かれたプログラムを, コンピュータが直接実行可能な機械語のプログラムに変

換するソフトウェアである. コンパイラ型言語は, ソース・プログラムを、始めから終わりまですべて実行可能形式（通常は機械語コード）に変換してから実行するため, プログラムの変更 → コンパイル処理 → リンク処理 → 実行という手間がかかる. コンパイラは, 高速かつコンパクトなオブジェクトを生成する. コンパイラでは多くの場合、ソースコードの言語は、人間向けの簡潔な言語（高級言語）であり, オブジェクトコードはコンピュータが直接実行可能な機械語である。コンパイラで処理される代表的な言語として,C 言語などがある.

- アセンブラ

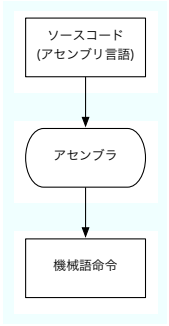
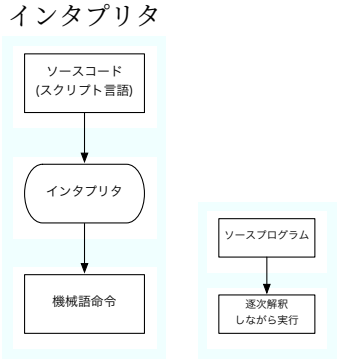


図 4: アセンブラ

アセンブリ言語を機械語に変換する事をアセンブルすると言い, それを行うプログラムの事をアセンブラ (assembler) と言う.

- インタプリタ



インタプリタ（interpreter）とは, プログラミング言語で書かれた

ソースコードを逐次解釈しながら実行するソフトウェアである. インタプリタの中には、実行直前に一度ソースコードを中間的なコード（中間言語）に変換し、それを逐次解釈するものもある. インタープリタ型言語の大きな特徴は、実行時にプログラム・コードの解釈を行うため、ソース・プログラムを記述してすぐに実行できるということである. コンパイラ方式と比較して, 会話的な応答性に優れる（プログラムを作成/変更してから実行するまでの手間がない）. 単純な実装では実行に時間がかかる. しかし動的に最適化を施すことができるので一概にコンパイラより遅いとは言えない.
インタプリタで処理される代表的な言語として, スクリプト言語などがある.

コンパイラとインタプリタを比較してみた.

言語プロセッサ	開発	規模
コンパイラ	しにくい	大規模向け
インタプリタ	しやすい	小規模向け

- ジェネレータ

  あらかじめプログラムの骨組みができており, 利用者が入力データや処理結果の内容と形式, および処理条件などを一定の書式の各欄に記入, 入力すると, 自動的に必要なプログラムを作成してくれるプログラム. ジェネレータで処理される代表的な言語として, パラメタがある.

(b) コンピュータのプロセッサ (CPU) 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.

1. アキュムレータ (ACC)

   演算結果を格納したり, データを一時的に保存するレジスタである. アキュムレータには足し算, 引き算などの基本的な演算を行なう回路が付属してあり, アキュムレータのデータとデータバス上のデータを演算した結果をアキュムレータに保存することができるようになっている。また、データバスからデータを取り込んだり、データバスにデータを出したりすることができるようになっている。

2. インデックスレジスタ (IX)

   アドレスレジスタの一種で, アドレス計算用の加算回路がついている. メインメモリのアドレスのオフセットを格納するレジスタである.

3. プログラムカウンタ (PC)

   次に実行するべき命令が格納されているメインメモリ上のアドレスを指し示すレジスタ. 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)

   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. この内容をアドレスバスに出す事により、メインメモリからデータを読み出す.

5. 命令レジスタ (IR)

   CPU の実行ユニットの一部であり, 現在実行中の命令を格納する.

6. フラグレジスタ (FR)

   フラグレジスタとは、CPU の演算状態を示す値であるステータスレジスタの一種. フラグレジスタは,CPU が特定の命令を実行した後に自動的に付与される.

7. スタックポインタ (SP)

   スタック上で最も最近参照された位置を指している. スタックのサイズがゼロのとき, スタックポインタはスタックの基点を指す.

8. 汎用レジスタ

   アキュムレータとしてもアドレスレジスタとしても, どちらにも自由に使えるレジスタ. データの計算をする回路とアドレスの計算

をする回路を一緒にしてしまう事ができる上, 命令の種類を減らす事ができるため, ほとんどの CPU は汎用レジスタを持っている.

以上のレジスタのうち,KUE-CHIP2 に無いものは, スタックポインタと汎用レジスタである.

6 感想

今回の実験では BNZ の部分を BZP にするとうまくいかないという原因不明のことが起こり, それに時間をかなりとられ, 苦戦してしまった. 次回はもっとスムーズに実験を進めていきたい.

参考文献

[1] IT 用語辞典
http://e-words.jp/