知能情報実験II　:　第七回レポート

175751C 宮城孝明

平成 30 年 10 月 15 日

目 次

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), 実行の書く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ラ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

この実験では,KUE-CHIP2 を用いてアセンブラプログラミングを人の手で機械語プログラムに変換し, その結果をオシロスコープで確認した. 初めのプログラム結果では, オシロスコープの波形はノコギリ型を示す形で表示された. そして, 課題の波形では矩形波や山形波, 菱形波の 3 種類であった. これらのプログラムを組み立てるために, まずは頭の中のイメージを直接図にして描いてみたりなどした. そうすることで, 頭の中を整理していく. イメージが出来てきたら, プログラムを実際に書いていく. 出来上がったプログラムを直接機械語に翻訳していく. この 1 つ 1 つの工程をこなすことで, 作成者自身のプログラミング力が向上し, さらには目的の高水準言語とアセンブラ言語との違いをより明確的に把握することができる.

3　実験結果

1. 実験 (1) について

   本実験の結果に関しては, 報告を省略する.

2. 実験 (2),(3) の結果について

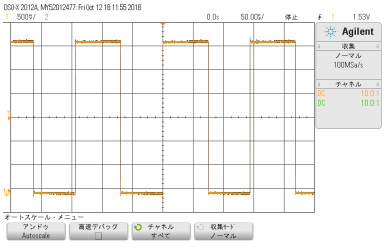

図 1: 矩形波の波形

Listing 1: 矩形波の波形

```
1  ADRS  DATA  OPECODE
2   10    C0    EOR  ACC,ACC
3   11    6D    LD  IX,(FF)
4   12    FF
5   13    10    OUT
6   14    AD    SUB  IX,01H
7   15    01
8   16    31    BNZ
9   17    13
10  18    65    LD  ACC,(FF)
11  19    FF
12  1A    6D    LD  IX,(FF)
13  1B    FF
14  1C    10    OUT
15  1D    AD    SUB  IX,01H
16  1E    01
17  1F    31    BNZ
18  20    1C
19  21    30    BA
20  22    10
```

それでは, 矩形波の波形プログラムを見ていこう. 初めに,ACC と ACC の XOR を使用して ACC の値を 0 にする. 次に, データ領域 (40H) に格納していた FF の値を変数 IX に代入する. そして,OUT を用いて ACC の値を表示していく.ACC の値は 0 のため表示では下に棒線が続くようになる. 次の処理で変数 IX の値を 1 ずつ減らしていく. そして,IX の値が 0 になるまで, 繰り返し処理を続けていく.IX の値が 0 になれば次の処理が開始される. ACC と IX はに,IX に代入したデータ領域 (40H) をまた格納する. そして,ACC の値を表示する.ACC の値は,FF のため表示では上に棒線が続くようになる. そして,IX を 1 ずつ減らしていく. そして,IX の値が 0 になるまで処理を続ける.IX が 0 になればまた, アドレス 10 に戻り再度繰り返し処理を開始していく.

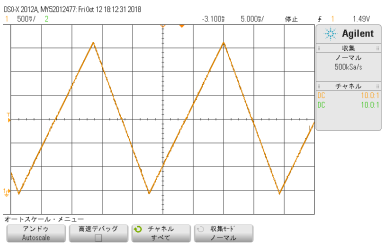

図 2: 山形波の波形

Listing 2: 山形波の波形

```
ADRS DATA OPECODE
 40   6D   LD IX,(FF)
 41   FF
 42   65   LD ACC,(FF)
 43   FF
 44   10   OUT
 45   A2   SUB ACC,01H
 46   01
 47   31   BNZ
 48   44
 49   10   OUT
 4A   B2   ADD ACC,01H
 4B   01
 4C   AA   SUB IX,01H
 4D   01
 4E   31   BNZ
 4F   49
 50   30   BA
 51   42
```

それでは, 山形波の波形プログラムを見ていこう. 初めに, データ領域 (40H) の値 FF を変数 IX に格納する. そして, 変数 ACC にもデータ領域 (40H) を格納する. そして,ACC の値を表示する. そして,ACC の値を 1 ずつ減らし ACC の値が 0 になるまで, 表示と減少を繰り返していく. これにより, ノコギリ波とは逆向きの波形を作ることができる. ACC が 0 になり次の処理を行う. 初めは ACC の値を表示し,0 になった ACC に 1 ずつ増やす処理を行う. そして, 初めに値を格納した IX を 1 ずつ減らしていく. そして,IX が 0 になるまで繰り返す. これにより, ノコギリ波と同じ波形を作ることができる. よって, 全体図としては, 山形波ができる. そして, また初めの処理に戻り, 繰り返していく.

4 考察

1. 実験 (1),(2),(3) の考察について

   今回の方法では,2 つ以上の同時出力が可能でない. 菱形型波形を作成時に何度か同時出力を試したが, 出来なかった. 菱形は, 上の点と下の点を条件分岐する前に行わないといけない. しかし, そのような方法を取ってしまうと, 次の処理に影響が出てしまう. よって, 今回課題として出ていた菱形波形の完成図には微細のズレがある. これにより, 同時出力をしないといけない処理がある波形 (例えば, 円や

二重線, 二次関数) は出力出来ない可能性がある. それ以外の波形は出力することは可能かもしれない.

2. その他の考察について

アセンブラ言語の OUT が出力する変数は ACC だけである. そのため, 変数 1 つだけでどうにか出力をやりくりした. 変数が限られたプログラムはしたことがなく, とても苦労した. いつも使用している言語では多くの変数が利用でき, どれだけ楽にプログラムができていたのかがよくわかった. この実験を通して, 普段のプログラム作成時には, 変数をもっと意識して作っていくことが必要であると認識を改めるようにしないといけない.

5 調査課題

1. パイプライン処理について

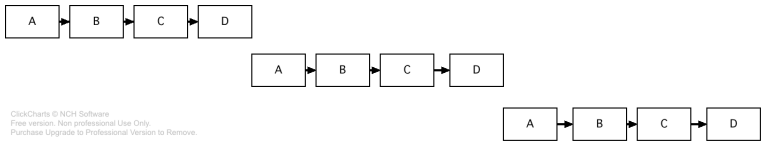

図 3: 通常の逐次制御方式

まず, 図 3 を見て欲しい.A は命令の取り出し,B は命令の解読,C は対象データ読み出し,D は命令の実行で分けられたステージを 3 回処理している. 通常の逐次処理は,1 つの命令が終わするまで次の命令を実行できない. このような流れ作業のため, 非効率に回っていることがよくわかる.

そのため, 全体の処理を効率に行うために使われる方法としてパイプライン処理がある.

図 4 を見てみる.

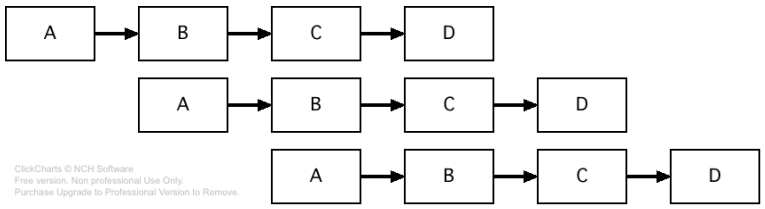

図 4: パイプライン処理

この処理は, 各ステージが終了したあとに, 次の命令が始まるようになっている. 例えば,1 番目の命令のステージ A の処理が始まり, 終了したら 2 番目の命令のステージ A も始まる.1 番目の命令の ステージ B が終了したら 3 番目の命令のステージ A の処理が始まる. このように処理していくことで, 通常の逐次処理よりも早く処理を終わらせることができる. 例えば, ステージ 1 つの処理に 1 クロックかかるとしたら, 通常の逐次制御方式には 12 クロック, パイプライン処理は 6 クロックと半分の時間で処理する.

この処理の欠点としては, 命令を先読みして行なっているため, 条件分岐の命令が出てきた場合先読みした部分が無駄になってしまうことがある. これを分岐ハザードと呼ぶ.

2. アセンブラ, コンパイラ, インタプリタについて

言語プロセッサは, ソースプログラムをパソコンが理解できる機械語に変換してくれるプログラムである.

アセンブラは, 機械語と 1 対 1 に対応している言語である. 初期の頃のコンピュータによく使用されていた. 現在でも, ハードウェアを制御するマイクロコンピュータの世界で活躍している. うまく利用できればコンピュータの性能をフル活用ができ, メモリ容量を最小に抑えることもできる. 小規模のシステムで利用されている.

代表的な言語としては, アセンブラ言語である.

コンパイラは, 人間が書いたソースコードを一括で実行可能な形式に変換する言語である. 一度で実行可能な形式に変換するため, 実行速度はインタプリタ言語よりも速い. コンパイラによっては, 実行効率が悪いものになる. メモリ容量は, アセンブラよりも多くなる. 開発効率が高いため, 大規模な開発に向いている.

代表的な言語としては,C 言語,Java 言語などが取り上げられる.

インタプリタは, ソースプログラムを機械語に変換せずそのまま,1 行ずつ翻訳し実行を行う. そのため, 実行終了にはコンパイラ言語と比べ, とても時間がかかる. しかし, インタプリタ言語はプログラムが全て完成していなくても実行可能であり, デバックが容易である.
代表的な言語としては,python,PHP などが上げられる.
高水準言語は, 人間の言葉に近い言語で人間が理解しやすい. メモリや入出力の制御を行わなくてよい.

低水準言語は, コンピュータが理解しやすくて, メモリや入出力装置などの周辺機器の制御ができるため, 高レベルな処理が可能である.

6　感想

今回は, 実験の課題 3 つが厳しくすぐに終わらせることができず, さらに追加課題も終わらせることができなかった. 自分の考えすぎる点も問題だと感じた. プログラミングは, 試行錯誤が大事であり, 失敗しても何度も繰り返し, 成功できるまでやり続けることが大切だと改めて認識した.

7　参考文献

参考文献

[1] 基本情報技術者; きたみりゅうじ, キタミ式　イラスト IT 塾　平成 30 年度　基本情報処理技術者試験基本情報技術者 2018

[2] processor; http://www.itlicense.com/Program_language/Language_processor.html

[3] picfun; http://www.picfun.com/prog04.html

[4] wcnixwu6yec; https://wcnixwu6yec.wiki.fc2.com/wiki/高水準言語と低水準言語