コンピュータと教育 51－1
（1999. 2. 23）

構造化チャートを利用したアセンブリ言語教育の提案

神村 伸一
東北科学技術短期大学 情報工学科
kami@cc.cstt.ac.jp

構造化チャートの一つである PAD(Problem Analysis Diagram)を利用してチャートで表現した論理記述から機械的にアセンブリ言語のプログラムを生成する教育方法について述べる。アセンブリ言語のプログラミングに必要な基本制御構造とその制御構造を実現するアセンブリ言語の命令パターンを定義したコーディング規則を準備する。学生は演習プログラムを PAD で記述した後、PAD とコーディング規則を対比させながら機械的にアセンブリプログラムを書き出すことができる。この結果、高級言語のプログラミングと同様の感覚でアセンブリ・プログラミングが可能となる。

Applied Structured Chart(PAD) to Education of Assembly Language for Microprocessor

Shinichi Kamimura
Department of Information Engineering
College of Science and Technology TOHOKU

It is stated about the education of the assembly language which PAD (Problem Analysis Diagram) of the structured chart was applied to. Assembly source program is formed mechanically from the logic description that it was expressed in the chart. The student can make a list of Assembly source program mechanically with making it compare the coding rule with PAD after the practice programme is described with PAD. Assembly programming became possible by the same sense as programming of the high level language as this result.

1．背景

大学の情報系学科において、学生にアセンブリ言語のプログラミングを通してコンピュータの仕組みを理解させることは大変重要である。情報処理学会の情報処理教育カリキュラム調査委員会 J97-WG が策定しまとめた「大学の理工系学部情報系学科のためのコンピュータサイエンス教育カリキュラム J97」[1]においても、学部科目 U-4「コンピューターアーキテクチャ」の中で、アセンブリ言語のプログラミングを実験及び演習として次のように提案している。

アセンブリプログラミング：身近にあるコンピュータ（8 ビットマイクロプロセッサやボードコンピュータなど）のマシン語によって簡単なプログラムを作成し、ハードウェア機構について理解する（以上、抜粋）。

しかしながらアセンブリ言語のプログラミングをゼロから学ぶことは、BASIC や FORTRAN、C 等の高級言語を学ぶことと比較すると容易ではない。なぜアセンブリ言語習得が難しいのか、一つはアセンブリ言語の命令・マシン語は CPU アーキテクチャと密接に関連しており、CPU のリソースであるプログラムカウンタやアキュムレータ、汎用レジスタ、状態フラグ等を直接または間接的に操作する。中でも状態フラグは命令を実行する毎に、操作したデータや結果に基づいて影響を受け、プログラムの要となる条件分岐命令を使いこなすためにも十分な理解が必要である。つまりアセンブリ言語でプログラムを作成するためには、プログラミング・モデルで定義されている CPU のリソースの理解と命令の機能（多くが単機能）および実行に伴って変化する状態フラグの意味など、プログラムの論理記述とハードウェア動作の両方を同時にイメージして組む必要があり、学習者とっては大きな負担となる。

またアセンブリ言語を学ぶ以前に高級言語を使ったプログラミングを経験している場合が多いことから、既修の高級言語プログラミングの影響もあると見ている。本学のケースを例に考えてみる。本学情報工学科の学生は 1 年次前期には C、後期は FORTRAN、2 年前期には PASCAL といった高級言語プログラミングを履修する。2 年次後期に情報工学実験においてマイコンキットを利用したアセンブリ言語プログラミングを履修するカリキュラムになっている。このため高級言語プログラムのようなブラックボックス的な論理記述の表現は慣れているが、アセンブリプログラムのようにホワイトボックス的な論理記述には不慣れである。高級言語なら 1 行で表現できる単純な代入文でもアセンブリプログラムでは複数個のマシン語を並べなければならないし、高級言語なら簡単に記述することができた基本制御構造 if～then～else や while～do 等も、アセンブリプログラムではマシン語を組み合わせて制御構造を実現しなければならない。それ以上にアセンブリ言語命令には高級言語における if や while といった制御構造そのものを表す命令語がないので、多くの学生はアセンブリプログラムでは構造化技法が無縁であると考えている。ほとんどの学生は１年半高級言語プログラミング感覚に慣れて戸惑いを感じている様子が伺えた。事実、学生にアセンブリ言語を学ぶ上で苦労したことは何か尋ねてみると、この慣れを払拭するのに苦労したという答えが多かった。以上のように学習者がアセンブリ言語プログラムを学ぶにあたって、大きな負荷となる二つの要因を可能な限り低減する教育方法を検討した。その結果、構造化チャートの一つである PAD を利用した教育方法を開発したので提案する。

２．教育方法の狙い

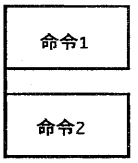
今回、次のような項目を満たすためにアセンブリ言語教育に構造化チャート PAD を導入した。

1) 学習者は構造化技法に従いアルゴリズムを考えることができるようにする。構造化技法の連接、選択、反復の基本制御構造を用意する。チャートで表現した論理構造がそのままアセンブリ・プログラムへ反映されること。つまり最終的に構造化されたアセンブリ・プログラムが自然に完成すること。アセンブリプログラムにおいても構造化されたプログラムを記述する事ができるという認識が養えるようにする。
2) 学生が PAD を作成した後、手作業ではあるが、PAD から機械的にアセンブリ言語へ変換できるコーディング規則(変換規則)を作成する。当面は CPU 内部の状態フラグ等を理解できなくてもブラックボックス的にアセンブリプログラムが作れる仕組みを用意し、アセンブリ言語そのものに慣れることで学びはじめの負荷を低減する。
3) アルゴリズムを考える場合は大枠から詳細へ思考を進める、いわゆるトップダウンアプローチの考え方が自然に身につくようにする。

3．PAD-アセンブリ言語変換規則

前述した条件を踏まえ、次のような PAD-アセンブリ言語変換規則（コーディング規則 - Z80 版)を作成したので説明する。

1）連接 は2つ以上の連続した 処理 と定義する。処理の最小単位は1つの命令で、複数命令の並びも1つの 処理 としてマクロ的に扱ってもよい。




```
命令1
命令2
```

図1. 連接

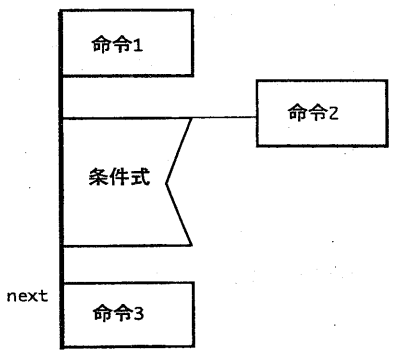
2）選択 と 反復 で必要になる条件式は アキュムレータ A と任意の値 x との比較 と定義して、表1のような条件式から条件分岐命令の分岐記号へ変換する表を準備した。

表1.条件式と展開する分岐記号

条件式	分岐記号1	分岐記号2
A＝x	NZ	－
A≠x	Z	－
A＞x	C	Z
A＜x	NC	Z

表1の分岐記号1と分岐記号2は 選択 および 反復 の PAD から生成されるアセンブリ言語命令群の中の条件分岐命令の第1オペランドへ記述する分岐記号のことで、NZ は Z フラグが 0 なら分岐、Z は Z フラグが 1 なら分岐、C はキャリーフラグが 1 なら分岐、NC はキャリーフラグが 0 なら分岐、－は必要なし(つまり2番目の条件分岐命令は不要)を意味する。

選択 は図 2. 単純型(if～then)と、図 3. 二分岐型(if～then～else)を用意した。この図は、左側の PAD の記述から表1を利用して条件式から分岐記号を求め、右側のようなアセンブリプログラムへ展開する規則を表している。なお図中の next はラベルを表し、命令3が置かれている番地である。




```
命令1
CP  x
JP  分岐記号1、next
(JP  分岐記号2、next)
命令2
next:命令3
```

図2. 単純型選択（if～then）

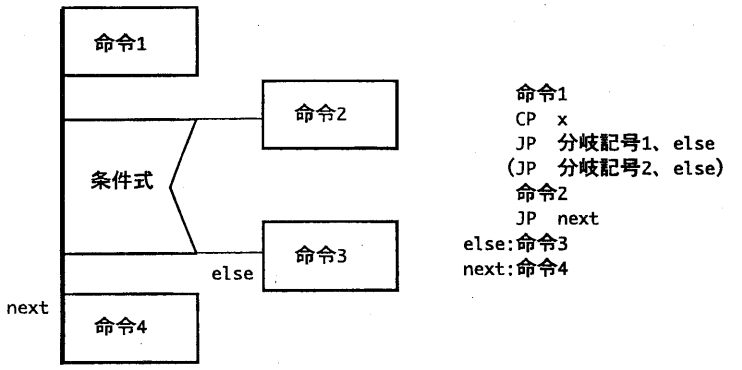


図 3. 二分岐型選択（if～then～else）

反復 は図 4. 前判定反復(while～do)、図 5. 後判定反復(do～until)、図 6. 問題向き反復(for～)を用意した。

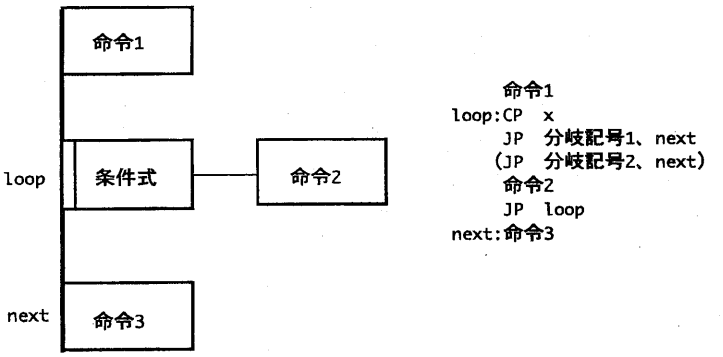


図 4. 前判定反復(while～do)

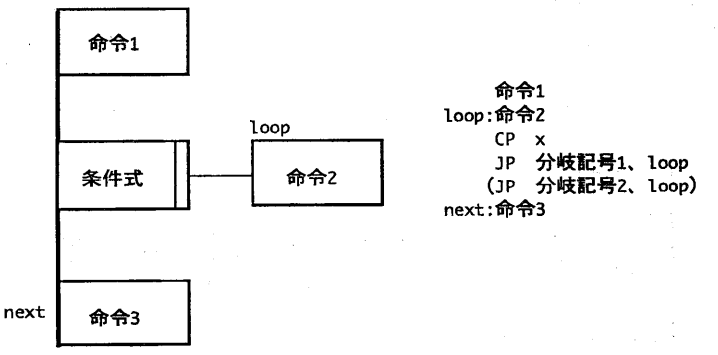


図 5. 後判定反復(do～until)

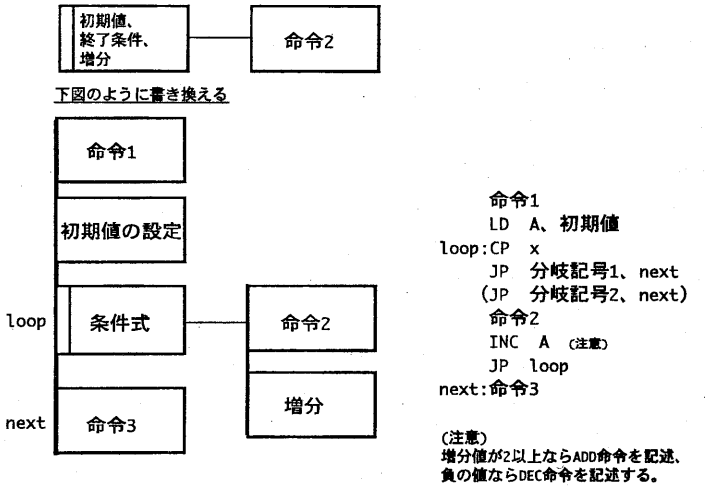


図 6. 問題向き反復(for～)

3）コーディング規則を用いた変換手順

実際に説明してきたコーディング規則を使った変換例を図 7 で説明する。

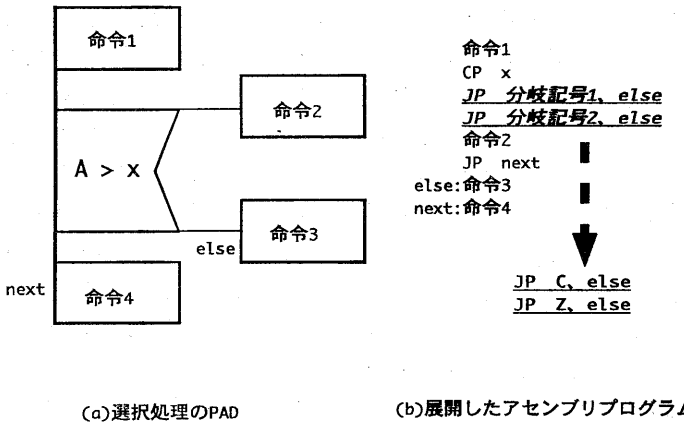


図 7. コーディング規則を用いた変換例

図 7(a)の PAD は一筆書きの要領で 命令 1 から 命令 4 に向かって矢印の順番にアセンブリ・プログラムへコーディングしていく。 選択箱 は PAD-アセンブリ言語-変換規則に従い変換すると、図 7(b)のようなアセンブリ・プログラムに展開される。ここで 選択 の条件式は A>x なので、表 1 条件式と展開する分岐記号 から条件分岐命令は 2 個必要であることがわかる。ここでは条件記号 1 と条件記号 2 を持つ条件分岐命令を連続して並べ、あとは変換規則にしたがって 命令 3、命令 4 を順番にコーディングすればよい。この例で 命令 1、 命令 2、 命令 3、 命令 4 は必ずしも一つの命令である必要はなく、複数の命令からなる 処理 でもよい。つまりトップダウンの考え方で、はじめは 処理 として選択処理をコーディングし、後から処理の中をさらに細かく展開するだけである。

4. 実践結果について

過去に一度、本学情報工学科2年次後期に開講している 情報工学実験3 において、コーディング規則を用いたアセンブリ言語教育へ導入を試みた。 情報工学実験3は市販されている教育用 Z80 マイコンキットを利用してマイコンのアセンブリプログラミングと CPU の基本原理およびマイコンの入出力インタフェースの実験を行う。実験は3コマ（270分）×15回実施した。

実験課題の内容はアキュムレータの加減算、転送、ブロック転送、カウンター、大小文字変換、テーブル処理、LED点滅制御、スイッチ入力などである。また実験はおおよそ次のような手順で進められた。

1) 学生はPADを使用し実験課題のアルゴリズムを検討する。
2) PADが完成したらPAD-アセンブリ言語変換規則（コーディング規則）を用いて、PADから機械的にアセンブリプログラムへ展開しコーディングシートへ記述する。
3) アセンブリプログラムを別途準備しているZ80ニーモニック表を用いてハンドアセンブル作業をおこないマシン語を生成する。
4) Z80マイコンキット上のモニタプログラムを用いて、アセンブルが完了したマシン語プログラムをプログラムメモリへ1バイトずつ書き込む。
5) プログラムを実行して動作を確認する。予想した動作と異なれば1)へ戻りアルゴリズムを再検討する。

5. 実践結果の検討

1) 学生は PAD を使い高級言語の感覚でアルゴリズムを考え、完成した PAD を先に述べたコーディング規則に従い機械的にアセンブリ言語へ変換することで、構造化れたアセンブリプログラムを完成させることができるようになった。当初、多くの学生は「アセンブリプログラムは難しい」という思い込みを持っていたが、次第に「めんどうだが、けして難しいものではない」という雰囲気に変わってきた。これは学生の実験に臨む態度や提出された実験報告書からも垣間見ることができる。本研究の目的であったアセンブリ言語に慣れさせることで、アセンブリプログラムを初めて学ぶときの負荷を低減することができたと判断できる。またPAD自体が構造化に反する論理（流れを乱す不適切なGOTO文の使用）を描けないので、学生は自然に構造化されたプログラムとはどういうものかを学ぶことができたと考える。つまり構造化は高級言語プログラミングに限った手法ではなく、論理記述を検討するときに意識すべきものであることを実感させることができたと考える。

2) この方法を用いるとコーディング規則によりブラックボックス化してある CPU リソース、具体的には状態フラグを理解しなくても、ある程度のアセンブリプログラムを作ることができる。このためコーディング規則を利用したアセンブリプログラミングだけで終わってしまうと、本来の教育目的であるマイコンの仕組みを理解することが疎かになる恐れがある。これを避けるため適時に CPU リソースの解説とその機能を利用するような課題を与えることが必要となる。

3) コーディング規則を利用したアセンブリプログラミングは、常にアキュムレータを制御パラメータ（条件式）として使っている。このため他の目的でアキュムレータを使用する場合は、復帰・退避の処理が必要となり、プログラムが煩雑になる、冗長な命令群を生成する等が起こる。これは仕組み上、避けられないのだが、アセンブリ言語に慣れ親しむことが第一優先なので、前述 2) と同様、徐々に汎用レジスタの使い分けや高機能命令を理解すればよいと考える。むしろ学生にプログラムの煩雑さや冗長な命令の並びに気付かせることが重要だと考えている。

5．まとめ

今回、構造化チャート PAD を利用したアセンブリ言語教育の新たな方法を開発した。学生はアセンブリ言語の学び始めにおいて、CPU のリソースを殆ど意識せずに高級言語の感覚でアセンブリプログラムを作ることが可能となる。この方法をアセンブリ言語の学び始めにおいて上手に適用すれば、学生が抱いているアセンブリプログラムに対する不慣れからくる負荷を低減することができることが判った。また構造化プログラミングは高級言語やその特定の命令文に依存するものではなく、論理記述での表現方法であることを実感させることもできた。今後、5. 実践結果の検討 3) で述べたような本方法のデメリット面を教材として活かし、アセンブリプログラミングに慣れた以降、どのように CPU リソースやマイコン内部の仕組み等の理解を深めていくか検討したい。

[参考文献]

[1] 情報処理学会：大学の理工系学部情報系学科のためのコンピュータサイエンス教育カリキュラム J97、1997

[2] 情報処理学会：大学等の情報専門学科における情報処理教育の実態に関する調査研究-平成 9 年度報告書、1998-

[3] 神村、石田：PAD を導入したアセンブリ言語教育の試み、東北科学技術短期大学研究紀要第 2 巻、PP53-63、1995

[4] ComputerToday1985/5 別冊「PAD-構造化プログラム開発技法」サイエンス社

[5] 日立ソフトウェアエンジニアリング編「やさしい PAD 入門」オーム社