コンピュータと教育　1－7
（1988. 7. 14）

# ＦＯＲＴＲＡＮ　７７によるＣＡＳＬアセンブラの開発

平山　弘　　田口　幹
神奈川工科大学　機械システム工学科

計算機教育によく使われるＦＯＲＴＲＡＮ　７７言語を使いＣＡＳＬアセンブラの開発を行なった。このシステムは、単にアセンブラの教育用というだけでなくアセンブラの中身を見ることが出来るような構造にし、リンカー等の理解に役立つように作成した。アセンブラ、リンカー、シミュレターを含めて約４０００行位のプログラムであるが、ＦＯＲＴＲＡＮの言語上の制限から論理和、論理積、シフト演算等を実現するために不自然なプログラミングを必要とした。また、入力処理に関しては実現不可能な部分もあった。アセンブラ言語の目的である演算の高速化、入出力処理および割り込み処理の学習にはＣＡＳＬアセンブラは不向きであるように思われる。

# Development of the Assembly Language CASL by FORTRAN 77

Hiroshi HIRAYAMA and Kan TAGUCHI
Dept. of Mechanical Systems Engng. Kanagawa Institute of Technology
1030 Shimo-ogino Astugi Kanagawa-ken 243-02 Japan

The assembler CASL, linker and simulator of the COMET computer for education are developped by the programing language FORTRAN 77. The size of this program include linker and simulator is about 4000 lines in FORTRAN 77. The aims of the development are not only learning of an assembely language but also undestanding of the linker and simulator. In order to the restrictions of FORTRAN 77, we have to write long program, and there are the functions of the program we cannot impliment. We appoint the probems in the CASL assembler when we use it for executing the program fast and writing of the I/O control program.

## 1．まえがき

コンピュータといろいろな機械や測定器と接続し使いこなすためには、コンピュータを理解する必要がある。アセンブリ言語を習得することは、コンピュータを理解するために最も良い方法だと思われる。アセンブリ言語を修得するするためには、コンピューターのアーキテクチャー、メモリ、レジスタ等のハードウエアに関する知識、２進・１０進・１６進数についての知識、ＡＮＤ・ＯＲ・ＳＨＩＦＴなどの論理演算に関する知識などが要求される。これらの内容は計測機器等を使うためには必ず理解しておく必要のある事柄で、たとえ高級言語だけを使う場合でも必ず理解していなければならない事柄である。このために、コンピューター関連の教育では必ずといっていいほどアセンブリ言語教育が行われている。

アセンブリ言語の教育では、アセンブリ言語がコンピューター毎にその内容が異なるため、どのコンピューターを選ぶかが大きな問題となってくる。大型計算機を使うことも考えられるが、大型計算機は非常に複雑なオペレーティングシステムを持っているため、アセンブリ言語でプログラムを書くためにはある程度これを理解する必要がある。このため、初心者を対象とする教育の場合には、非常に多くの内容となり短期間では不可能であると思われる。

実際の応用で使われているマイクロコンピューターを使うことも考えられる。これらの物としては、インテル社の８０８０・８０８６、モトローラ社の６８０００、ザイログ社のＺ８０等が考えられる。いずれのマイクロコンピューターでも完全に使いこなすには、かなりの時間が必要である。時間がある程度あるならば、これらのマイクロコンピューターを使うのも一方法と思われる。

さらに、仮想コンピューターを使うことも考えられる。これらのものとして一番に考えられるのが、通産省の情報処理技術者の試験で使われている仮想コンピューターＣＯＭＥＴのアセンブリ言語ＣＡＳＬである。このコンピューターは非常に単純な構造を持ち、命令も３０命令と少なく、疑似命令も大変少ない。初心者にも容易に修得しやすいように思える。それ以前に使われていた仮想コンピューターＣＯＭＰ－Ｘと比較して、コンピューターのアーキテクチャーや命令が現実のコンピューターに近くなり、メモリーのセグメントがなくなり大変使い易くなっている。また、このコンピュータについては多くの参考書等が出版され、非常に勉強しやすい環境にもなっている。

このようなことから、仮想コンピューターＣＯＭＥＴを教育用として使ってみるため、ＣＯＭＥＴ用のアセンブラ、リンカーおよびシミュレータを作成した。そして、その評価を行なった。

多数のコンピューター関連雑誌で、ＣＡＳＬおよびＣＯＭＥＴのアセンブラ、シミュレータが発表されている。これらの中でソースプログラムとして発表されているものもあるが、いずれもパーソナルコンピューターのＢＡＳＩＣ言語で書かれたもので、かなりそのパーソナルコンピューターに依存するようなプログラムである。このため、これを大型コンピューター等で使うことは殆ど不可能に近い。大型コンピューターで使うことを考慮して、今回作成したアセンブラおよびシミュレータは、標準的なＦＯＲＴＲＡＮ７７で、ほぼすべて記述した。

## ２．　アセンブリ言語ＣＡＳＬシステムの構成

これらのプログラムを作成するのに、特に留意して点は、以下の３点である。

（１）　教育用として開発したものであるが、あまり極端簡略化を行なわないことである。なるべく通常のアセンブリ言語を使うような形式で使えるようにすることである。これらのプログラムも、オペレーティング・システムが準備しているコマンドのように使うことができるようにした。すでに存在するプログラムと共存させるためでもある。

このようにすれば、アセンブリ言語がコンピューターシステムの中でどの様な位置にあり、どの様な役割果たしているのかを理解しやすいと考えたからである。コンピューター・システムを理解することも重要だと思われるからである。アセンブリ言語だけを使うのであるならば、エディター、リンカー等をその中に含ませるほうが使いやすい。しかし、コンピューター・システムに直接触れるようにしたほうが、コンピューターの学習者にとってより有益だと思われたので、アセンブラ、リンカー、シミュレータを一つ一つ作成した。したがって、アセンブリ言語ＣＡＳＬシステムは

（Ａ）　ＣＡＳＬアセンブラ
（Ｂ）　ＣＡＳＬ用リンカー
（Ｃ）　ＣＯＭＥＴ用シミュレータ

の３つのプログラムで構成した。

（2） 通常のアセンブリ言語では見ることが出来ないような部分もなるべく見ることが出来るようにすることである。アセンブルの意味、オブジェクトコードの構造等を理解し易いように、オブジェクト・コードは文字型ファイル形式とした。ロードモジュールも簡単にプリンター等に出力させて内部を見ることが出来るように、同様に文字型ファイル形式とした。

このような形式にすれば、処理速度が落ち、ファイルの大きさが大きくなるなどの非効率な部分が出てくるが、このプログラムの使用目的から、大きなプログラムを作ることはないと考えられるので、分かりやすさのほうを優先させた。

（3） 大型コンピューターや他のコンピューターに容易に移植出来るように作成した点である。このために、プログラムはごく一部を除いて、すべてＦＯＲＴＲＡＮ７７の範囲で記述し作成した。

これらのプログラムの中でＦＯＲＴＲＡＮ７７から外れる部分は、次の３点である。

（A） コマンドラインを得るために、サブルーチンを使用している。

（B） 時間と日付を得るために、サブルーチンを使用している。

（C） タブコードとして、１６進数の９を使っている。ＡＳＣＩＩコードを使っていないコンピュータでは変更が必要である。

この他に、ＦＯＲＴＲＡＮ７７の規格外のことであるが、整数形変数は３２ビット以上の精度があるとしてプログラムを作成している。

このような、システム関係のプログラムにはＣ言語が向いていると言われている。このプログラム作成開始時点で、当大学の計算センターでは、Ｃ言語が使えなかったことと、ＦＯＲＴＲＡＮ言語に慣れていたため、ＦＯＲＴＲＡＮによって作成した。次の段階ではＣ言語で作成することも考えている。

## 3 プログラムの概要

### （1）プログラムの起動法

プログラムの起動は、オペレーティングシステムのコマンドを実行するような形式とした。アセンブラを例にとり、プログラムの実行開始方法を説明する。

アセンブラは、コマンドの入力行からアセンブルの指示を与える。ソースファイル名は必ず指定しなければならない。オブジェクト・ファイル名やリスト・ファイル名もここで指示する。この場合、オブジェクト・ファイル名やリスト・ファイル名は省略出来る様にもなっている。ＭＳ－ＤＯＳ上では次のようになる。

```
A>CASL SOURCE
```

この場合、”ＳＯＵＲＣＥ”という名前のファイルをアセンブルせよという意味になる。実際にアセンブルされるファイル名は、ＭＳ－ＤＯＳをはじめ、ＵＮＩＸ等のオペレティング・システムで行なわれるように、”ＳＯＵＲＣＥ”の名前に拡張子が付加された名前が使われる。オブジェクト・ファイルおよびリスト・ファイル名の指定も、この名前に違った拡張子を付加した名前を持つファイルが自動的に指定される。

ＣＡＳＬでは、拡張子が省略された場合、次のような拡張子を使うように作られている。

| | | |
|---|---|---|
| ソース・ファイル | ―― | ＣＡＳ |
| オブジェクト・ファイル | ―― | ＣＯＢ |
| リスト・ファイル | ―― | ＬＩＳ |

基本的な使い方をする場合は、コマンドにソース・ファイル名だけの形だけで十分であるが、高度な使い方をするために、スイッチと呼ばれるオプション機能がある。これは次のような形式を持っているものである。

／＜キーの名前＞［＝＜キーの値＞］

＜キーの名前＞は機能を変更するために使われる名前である。＜キーの値＞はその変更のために値が必要な場合に指定する部分である。［］で囲まれた部分は省略可能な部分を意味する。キーの値がないような例としては次のようなものがある。オブジェクト・コードを出力しないことを指定する場合、

```
/NOOBJECT
```

のように指定する。キーの値が必要な例として、リスト・ファイルを”Ａ．ＬＳＴ”に変更する事を指定場合、

```
/LIST=A.LST
```

と指定する。これらの機能変更を行う場合、アセンブリ命令は次のように指定される。

```
A>CASL/NOOBJECT/LIST=A.LST SOURCE
```

ファイル名の指定にも、省略形を許すようにした。拡張子を省略した場合、自動的に付加するようになっている。

付加する拡張子は、ファイル名が省略された時に付加されるものと同じである。

（2） アセンブリ言語CASL

CASLのオブジェクト・コードは参考資料に載っているコードをそのまま使うことにした。他のCASLアセンブラでもこのようになっているので、互換性が良いと思われるのでこのプログラムでもそのまま使うことにした。

マクロ命令に対しては、参考資料の中である程度例示されているが、CASLアセンブリ言語毎にその実現方法は異なっている。ここで作成したCASLでは、マクロ命令も普通の命令と同様な命令として扱うことにした。INおよびOUT命令は4語使う命令とし、第1語には命令コードを入れ、第2語には第1オペランドの値を入れる。第3語には使用しないで、第4語に第2オペランドの値を入れるようにした。INおよびOUT命令の第3語目は、命令はすべて2語であるというCASLの仕様から2語の倍数に調整するために、ダミーに付加したものである。EXIT命令は2語の長さを持つ通常の命令のように設定した。EXITはオペランドがないので、第2語には何も入れないことになる。この様に設定することは、ハードウエア的には、存在しない命令を実行しようとして、割り込みが起こり、その割り込みルーチンで、IN、OUT，EXITのような機能が実行されると考えられる。

アセンブルの例として、CASLの仕様書にもあるCASLプログラムのアセンブル・リストを図1に示した。

（3） リンカーCLINKおよび実行モジュール

アセンブリ言語CASLではリンカーについて、何も規定していない。CASLではメインプログラムという概念がない。すべてサブルーチンである。このため、リンクを行なうためには、メインプログラムを指定しなければならない。CLINKでは、この指定をリンカーのオペランドで指定することにした。最初のオペランドで指定したモジュールがメインルーチンとなる。

CLINKでは、アセンブラと同様にファイルの拡張子を省略出来るようになっている。CLINKで使われる拡張子は、つぎの3つである。

| | | |
|---|---|---|
| オブジェクト・ファイル | ---- | COB |
| 実行モジュール・ファイル | ---- | CEX |
| マップ・ファイル | ---- | CMA |

これを利用して、次のような形で指定できる。

```
A>CLINK A B C
```

これは、A．COB，B．COBおよびC．COBをリンクし、実行モジュールを作成し、A．CEXに出力することを意味する。

（4） シミュレータ CSIM

仮想コンピューターCOMETのシミュレータである。これは、アセンブリ言語でプログラムを開発するときに使われるデバッガーとほぼ同じ内容を持つものである。このプログラムでは、作成したプログラムを実行するだけでなく、そのプログラムの細かい部分も検証が出来るように作られている。このシミュレータは、11個のサブコマンドをもっている。以下にそれを示す。このサブコマンドもCASLのキーと同様な省略形を持っている。下線部分が最小限指定しなければならない部分である。

① REGISTER　レジスターの内容を表示する。
② UNASSEMBLE　逆アセンブルする。
③ DUMP　16進ダンプする。
④ ASSIGN　メモリの内容を変更する。
⑤ LOAD　プログラムをメモリにロードする。
⑥ GO　プログラムの実行を開始する。
⑦ TRACE　レジスタの内容を表示しながら実行する。
⑧ BREAK　ブレークポイントの設定、解除を行なう。
⑨ LOG　ログファイルの開閉を行なう。
⑩ STACK　スタックの内容を表示する。
⑪ QUIT　シミュレータを終らせる。

シミュレータ実行のようすを図2に示した。このプログラムはスタックオバーフローによって、ストップするプログラムである。この計算では7番目のフィボナッチ数を求めている。

## 4. FORTRANで開発する上での問題点

FORTRAN 77は文字型が導入され非数値処理にも使えるような言語となった。これを利用してアセンブラを開発した。この開発で問題となるのは次の5点である。

### (1) 論理演算

最初の大きな問題は、AND・OR・SHIFTなどの論理演算のプログラムである。このような演算は多くの計算機で拡張機能として使えるが、その使い方及びその関数の名前は計算機毎に異なっている。このため、このビット演算関数を使った場合、他の機種と互換性を保つことが出来なくなる。ここのプログラムではどの計算機でも使えるように、FORTRANで1ビット毎に分解し、1ビット毎に演算を行った。CASLは16ビットのコンピュータなので、このような分解を可能にするためには16ビットを越える精度で計算を行う必要がある。このために、全ての16ビット演算を計算機の中では32ビットの精度で計算されている。このためこの部分は、非常に低速となる。

このアセンブラを使って作るプログラムは小さなプログラムが多いので、使用上問題となるほど遅くなることはない。もし非常におそく感じるような場合には、他の機械との互換性はなくなるが、機械毎にもっているその機械特有のビット演算関数を使うように変えなければならない。

### (2) カレンダー関数

アセンブラを含め、FORTRAN等のリストに年月日及び時間を表示するのが普通である。このため年月日及び時間を取り出すサブルーチンが必要である。このような関数は多くのFORTRANで持っているが、ビット演算関数と同じようにその使用法、名前は違っている。これは、ビット演算関数と違い共通にする方法がないので、その計算機毎に違うことになる。
この部分は互換性がなくなることになる。

### (4) コマンドライン

コマンドラインを取り出す機能は、このアセンブラを作るために絶対に必要な訳ではないが、この機能があると非常に便利である。最近のコンピュータではパーソナルコンピュータをはじめとして大抵このような機能を持っている。大型コンピュータでもこのような機能が実現出来るはずであるが、このような機能を使うのは一般に新しいコンピュータと比較して難かしい。

### (5) 実現不可能な機能

FORTRAN 77では実現出来ない機能があった。これはCASLのIN命令である。文字を読み込んだ時、FORTRANではその文字の長さは知ることは出来ないからである。ここで実現したCASLでは、最後に続く空白は文字の長さに加えないようにしてある。この部分がCASLの仕様と食い違う唯一の点である。

## 5. アセンブリ言語CASLについて

CASLは情報処理技術者試験のためのアセンブラである。通常のコンピューターと比較してかなり単純になっている。このため、COMETコンピューターでは、レジスター間の演算が殆ど出来ない。今までアセンブラを利用してきた者にとって、非常に書きにくく思えた。アセンブラでプログラムを書く大きな目的の1つは、高速実行である。このとき、レジスタをなるべく多く使い高速化するように訓練されてきたからである。コンピューターもその様使われることを前提に設計されいるからである。

アセンブラ教育の目的の一つに、割り込み等の学習がある。COMETには割り込み関係の命令およびそれを実現するための構造はない。このような目的にCOMETを使うには、それらの命令を付加させる必要がある。

このような欠点はあるにしても、単純な構造を持っているため、そのアーキテクチャ等を理解することは容易で、アセンブラの最初の学習には大変良いように思える。割り込みの学習のために必要ならば、そのような命令を付加すれば良いからである。

### 参考文献


1) 情報処理セミナー編集部：情報処理試験のアセンブラ言語　CASL　新紀元社 (1987)
2) 小島　進：MC68000の使い方　オーム社 (1982)
3) Stephen P. Morse ( 内藤　祥雄訳) :
8080入門　システムソフト (1981)
4) 平山　弘：教育用アセンブラCASLの開発
幾徳工業大学研究報告　B理工編　VOL.12 (1988)

```
***  CASL LISTING ( KANAGAWA INSTITUTE OF TECHNOLOGY )   ***     PAGE: 01
                                                      DATE  : 88-05-24 14:33:31
                                                      SOURCE: FNUMB.CAS

  NO. ADDR   CODE             SOURCE
 0001                ;  フィボナッチ数を求めるプログラム
 0002                ;  入力: ＧＲ１   何番目のフィボナッチ数を求めたいかを
 0003                ;                 ＧＲ１出与える。
 0004                ;  出力: ＧＲ２   フィボナッチ数列の第ｎ番目項の値
 0005 0000           FNUMB   START
 0006 0000 4010001E          CPA     GR1,CONST3
 0007 0002 60000008          JPZ     NEXT
 0008 0004 1221FFFF          LEA     GR2,-1,GR1
 0009 0006 81000000          RET
 0010 0008 70010000  NEXT    PUSH    0,GR1
 0011 000A 1211FFFF          LEA     GR1,-1,GR1
 0012 000C 80000000          CALL    FNUMB
 0013 000E 70020000          PUSH    0,GR2
 0014 0010 1211FFFF          LEA     GR1,-1,GR1
 0015 0012 80000000          CALL    FNUMB
 0016 0014 1120001F          ST      GR2,WRK
 0017 0016 71200000          POP     GR2
 0018 0018 2020001F          ADD     GR2,WRK
 0019 001A 71100000          POP     GR1
 0020 001C 81000000          RET
 0021 001E 0003      CONST3  DC      3
 0022 001F 0000      WRK     DS      1
 0023 0020                   END

    *** LABEL ***

  NO.    NAME      ADDR (HEX)        ADDR (DEC)
 0001    FNUMB     0000              00000
 0002    NEXT      0008              00008
 0003    CONST3    001E              00030
 0004    WRK       001F              00031
 0005    $$$END    0020              00032
```

図１　ＣＡＳＬアセンブル・リスト

```
COMET>LOAD FNUMB
COMET>REG
 PC=0000 GR0=0000 GR1=0000 GR2=0000 GR3=0000 GR4(SP)=0FA1 FR=0
 CPA    GR1,001E
COMET>GR1=7
 GR1 = 0007(HEX)       00007(DEC)
COMET>G
 *** STACK OVER ***
 PC=001E GR0=0000 GR1=0007 GR2=0008 GR3=0000 GR4(SP)=0FA1 FR=0
 ?
COMET>GR2
 GR2 = 0008(HEX)       00008(DEC)
COMET>QUIT
```

図２　ＣＳＩＭの実行例