エディタ・アセンブラ・デバッガを統合化した開発ツール

4E-3

藤原　久永　　平松　隆志

(岡山県工業技術センター)

1．はじめに

プログラム開発の過程で使われる各種のツール(エディタ、アセンブラ、コンパイラ、デバッガ等)は通常機能単位で独立したものとなっている。このようになっていることの長所としては、少ないメモリーで動作する、汎用的に使用できる、自分の好みに合ったものが選んで使える等が考えられる。しかし補助記憶として動作の遅いフロッピーディスクを使用している場合はプログラムのローディングならびにデータの受渡しに要するオーバーヘッドが大きい、といった欠点もある。最近はメモリーが非常に安価になって来ているため少ないメモリーで動作することは長所でも何でもなくなってきており、むしろ大容量のメモリーを実装し、それをＲＡＭディスクとして使用して動作の遅いフロッピーディスクの穴を埋めることが普通に行われている。しかしこの場合でも、これらのツール間でのデータの受渡し方が単一的なものに限られ複合的な機能を実現できないためコマンドの投入が煩雑になりその分効率が悪くなる、といったことは避けられない。

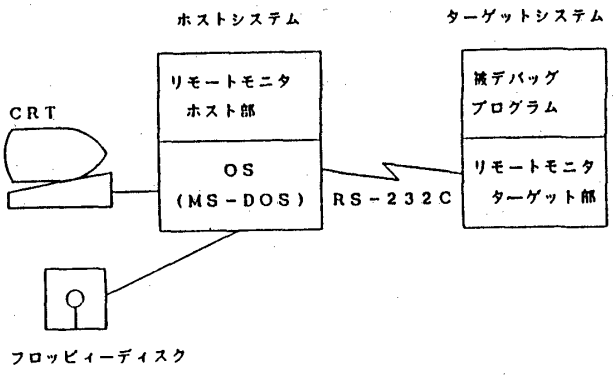
特にデバッグの過程と編集の過程とには密接な関係があるため、デバッガ自身が編集機能、アセンブル機能を持っていればプログラム作成の効率が良くなることは容易に想像できる。このような観点のもとに編集機能・アセンブル機能さらにはコマンド履歴機能を持ったデバッガを作成した。

2．デバッガの構成

演者らが対象としているシステムは小規模の組み込みシステムである。組み込みシステムのデバッグにはインサーキットエミュレータ（ＩＣＥ）がよく使われる。しかしここではＩＣＥは使用せず、ＩＣＥに代わるものとして図．１に示す様なクロスのデバッグシステムを作成した。これはターゲットシステムに置かれたプログラムとホストシステムに置かれたプログラムがＲＳ－２３２Ｃを介して協調して動作するデバッグモニタであり、リモートモニタと呼んでいる。ホスト側に置かれたプログラムがデバッグに必要なほとんどの処理を行う。ターゲット側に置かれたプログラムはホスト側から与えられた指示に従ってメモリーの読み出し、メモリーへの書き込みなど基本的な動作を行い、ＲＯＭ化して登載されている。言うまでもなくターゲット側のプログラムは対象とするシステム毎に書き換えなければならないが、この負担をできるだけ軽くするためターゲット側のプログラムは最低限必要な機能だけを持った小さなプログラムとなるようにしてある。デバッグの機能としてはメモリー・レジスターの参照・変更、逆アセンブル、アセンブル、トレース（ただしソフトウェアによるもの）、任意のアドレスからのプログラムの実行、ブレイクポイントの設定など標準的な機能は備えており、シンボルも使用できるようにしてある。なおホスト側のプログラムはPC-9801のMS-DOS上に作成されており、ターゲットのＣＰＵとしてはZ-80用のものと6301/3用のものがある。ターゲットのＣＰＵとしてこれらを選んだ理由はZ-80は最もよく使われる８ビットＣＰＵとしてこれを無視できなかったため、6301/3は演者らが対象にしようとするシステムにはこれ位が最も適していると思われるためである。

3．統合化への拡張

コンパイラとエディタを統合化したシステムとしてはMS-DOSやCP/Mといったパソコン上で動くTURBO-Pascalが有名である。TURBO-Pascalでは、プログラムの編集、コンパイル、実行のサイクルにおいて動作の遅いフロッピーディスクへのアクセスを行わないですみ、コンパイル中や実行時



図．１

Programming Tool Synthesizing Editor,Assembler and Debugger

Hisanaga FUJIWARA　Takashi HIRAMATSU

Industrial Technology Center of Okayama Prefecture

にエラーが発見された場合はただちにソースプログラムのその部分へカーソルをもっていってプログラムを修正することができるようになっているため開発効率は非常によくなっている。しかしこのようなことが有効になるためには実行中のプログラムがシステム領域を破壊せず、しかもプログラムが暴走しないかまたは暴走してもシステムに復帰する手段があるという条件が必要になる。同じ考えをアセンブリ言語に適用しようとしても、アセンブリ言語で作られたプログラムの場合は上記のことがよく起こるため単にエディタとアセンブラを統合化しただけでは不十分である。しかしＩＣＥのようなデバッガに組み込めば上記のような問題は避けることができる。実際アセンブリ言語でプログラミングしている場合に時間がかかるのはデバッグの過程であるからデバッガに統合化してしまうのが合理的であると思われる。

エディタ・アセンブラをデバッガに組み込む他にデバッグを効率的に行うためにデバッガに備わっていた方がよいと思われる機能として、マクロコマンドを実行できる機能、打ち込んだコマンド群が後で確認できる機能などが考えられる。特にデバッグの中期以降の段階では似たオペレーションの繰り返しが多くなるため前者の実現は必須である。これらをどのように実現すべきか、いろいろな方法が考えられるが、ここでは打ち込んだコマンドをコマンド履歴として記憶しておき、これを利用してコマンドの再実行が可能となるようにし、エディタの編集対象はこのコマンド履歴に対して行うようにした。さらにこのコマンド履歴はファイルとしてディスクにセーブしたりディスクからロードしてくることも可能になっている。このようにコマンド履歴を利用すればアセンブラは別に設けなくても、拡張する前のデバッガに備わっているワンラインのアセンブラが利用できることがわかる。ただしこのワンラインのアセンブラをそのまま使用したのではラベルが使用できないのでラベルが使用できるように手を加えなければならない。当初はプログラムを簡単にするために２パスのアセンブラとしていたが、２パスの場合ラベルのアドレスを決定するためにソースプログラム全体を読まなければならないため部分的にアセンブルを行うとコード生成を誤ることがある。そこで１パスのままでラベルを使用できるようにして部分的にアセンブルを行うことも可能となるようにした。またマクロ機能は実現されていない。なおアセンブル時間の大半はターゲット側へのコード転送時間で占められている。

エディタは先にも述べた通りコマンド履歴を編集対象とするスクリーンエディタとしてある。構造エディタと言える程のものではないが（そもそもアセンブリ言語のプログラムには構造といえるようなものがない）アセンブリ言語のプログラムは入力段階でラベル参照に関しないエラーに限ってエラーチェックを行うことも可能となるようにしてある。その際コメントを入れることも可能である。

４．実際の使用感

本ツールの特徴はアセンブリ言語のソースプログラムとコマンド履歴が同じレベルに置かれていることにある。このため外部で高水準言語などを使って作られたプログラムでもヘキサファイルさえあれば問題無くデバッグできる（ただしデバッガ中からエディタを起動してプログラムを修正することはできなくなる）。また長いプログラムは開始アドレスさえユーザが管理すれば部分的にアセンブルすることも可能である。しかしソースプログラムとコマンド履歴が同じレベルに置かれていることはユーザの立場からすればあまり使い勝手良いものではないようである。また、デバッガとエディタの結び付きが弱くなっており、デバッガから得られた情報がソースプログラムにまで反映されず、人手に頼らざるを得なくなっているのも問題である。

このあたりをどうするかは今後使用経験を深める中で検討していきたい。

５．おわりに

大規模なプログラム開発には大規模なプログラム開発のための方法やツールがあるべきであり、小規模なプログラム開発には小規模なプログラム開発のための方法やツールがあるべきである。

演者らは小規模の組み込み用プログラムを手早く作ってしまえるようなシステムを目的としており、今回作成したツールによってそうした目標のいくらかは達成せられたと思われる。今後は組み込み目的に適した高水準言語の検討と、それへの対応を図っていく予定である。

謝辞

本システムの作成に関して御指導・御助言下さいました電子技術総合研究所の真野芳久博士、東京都立工業技術センターの坂巻佳寿美氏、および本研究の機会を与えて下さいました南俊基前所長、川崎仁士部長に感謝致します。