社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2007－ＳＬＤＭ－128（12）
2007／1／18

教育用マイコンCOMET IIのCコンパイラ開発

松田 健　佐藤 暁　森 健介　堤 利幸

明治大学理工学部情報科学科 〒214-8571 神奈川県川崎市多摩区東三田 1-1-1
E-mail: {kmatsuda,tsutsumi}@cs.meiji.ac.jp

あらまし：情報処理技術者試験で用いられている仮想計算機COMET IIとそのアセンブリ言語CASL IIは、シンプルで理解しやすい構造のため多くの教育機関でCPUやアセンブリ言語の学習に広く活用されている。しかし現在組込みソフトウェアの開発では、C言語でCPU制御プログラムを記述することが一般的である。しかし実用的なCOMET II用Cコンパイラはこれまでなかった。そこで本研究では実用レベルのCOMET II用Cコンパイラの開発を行い、C言語を用いたCOMET IIマイコンの制御を可能にした。

キーワード: Cコンパイラ,COMET II,CASL II

Development of C Compiler for Educational Microprocessor COMET II

Ken MATSUDA　Akira SATO　Kensuke MORI　and　Toshiyuki TSUTSUMI
School of Science and Technology, Meiji University
1-1-1 Higashi-mita, Tama-ku,Kawasaki Kanagawa, 214-8571 Japan
E-mail: {kmatsuda,tsutsumi}@cs.meiji.ac.jp

Abstract :A microprocessor COMET II and it's assembly language CASL II , which were used by Information-Technology Engineers Examination by the Ministry of Economy, Trade and Industry , have been widely applied to learn the CPU and the assembly language in a lot of educational institutions, because the structure of the COMET II is simple and understood easily. However, recently embedded engineers control CPUs using C language. There is not an actual C complier for the COMET II. So, we have developed the C compiler for the COMET II. As a result, the C compiler enables learners to control the COMET II using C language.

Keyword: C Compiler ,COMET II,CASL II

1. はじめに

経済産業省による情報処理技術者試験で用いられている仮想計算機COMET IIはシンプルで理解しやすい構造であり、そのアセンブリ言語CASL IIと共に、多くの教育機関でCPUやアセンブリ言語の学習に広く利用されている。また、COMET IIを教材としたマイクロプロセッサの設計教育環境[1]や人材育成用ASICマイコン教材[2]などが研究されている。これらの学習や教育ではCOMET IIの制御プログラムをCASL IIで記述している。

しかし、近年マイコンを用いた組込みソフトウェアの開発においてはC言語でマイコン制御プログラムを記述することが一般的であり、アセンブリ言語でマイコン制御プログラムを記述している教育機関との相違が生じている。そのため、C言語によるマイコン制御教育を行えることが重要になっている。

教育機関で広く使われているCOMET IIの制御をC言語で行うためにはCOMET II用のCコンパイラが必要である。しかし、これまでは簡単なプロトタイプレベルのCOMET II用Cコンパイラ[2]しか存在しなかった。

そこで、本研究では実用的なレベルの仕様を持ったCOMET II用のCコンパイラの開発を行った。開発したCコンパイラとCOMET II搭載FPGAボードを利用して実際にC言語でCOMET IIマイコンの制御が可能であることを確認する。

2. COMET II・CASL II

2.1. COMET II

COMET IIはCPUの本質的な機能を持ちながらも初心者が簡単に理解できるように非常にシンプルな構造となっているマイクロプロセッサである。1語は16ビットで構成され、主記憶は65536語の容量を持つ。内部レジスタは汎用レジスタ（GR0-GR7）、スタックポインタ（SP）、プログラムレジスタ（PR）、フラグレジスタ（FR）の4種類を持つ。この他に演算装置や制御装置がある（図1）。

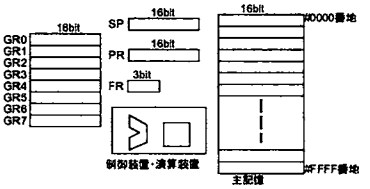

図1　COMET IIの構成

COMET II はプログラムとデータを主記憶に格納する。プログラムレジスタが示す番地から1命令ずつ取り出して実行する。COMET II の機械語命令は1語長命令と2語長命令が存在する（図 2）。オペコードは8bit であり、オペランドには1語長命令の場合は汎用レジスタを用いることができ、2 語長命令では汎用レジスタとアドレスを用いることができる。なお、2語長命令では指標レジスタ(GR1-GR7)を用いたアドレス修飾が可能である。

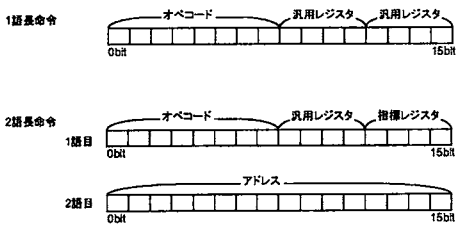

図 2　1語長命令と2語長命令のビット構成

表 1　COMET II 機械語命令

転送命令	ロード	1語長・2語長
	ストア	2語長
	ロードアドレス	2語長
算術・論理演算命令	加算（算術・論理）	2語長
	減算（算術・論理）	2語長
	論理演算（AND、OR、XOR）	2語長
比較命令	比較（算術・論理）	2語長
シフト演算命令	左シフト（算術・論理）	2語長
	右シフト（算術・論理）	2語長
分岐命令	分岐（無条件・正・負・非零・零・オーバーフロー）	2語長
スタック操作命令	プッシュ	2語長
	ポップ	1語長
コール・リターン命令	コール	2語長
	リターン	1語長
その他の命令	スーパーバイザーコール	2語長
	ノーオペレーション	1語長

表 1に示すとおり機械語命令は 28 種類あり、転送命令、算術・論理演算命令、シフト演算命令、分岐命令、スタック操作命令、コール・リターン命令に分類できる。転送命令、算術・論理演算命令、シフト演算命令では実行結果によりフラグレジスタが設定される。機械語命令の多くが2オペランド形式を採り、レジスター-レジスタ間またはレジスター主記憶間での処理が可能である。

2.2. アセンブリ言語 CASL II

CASL II は COMET II のアセンブリ言語である。CASL II の命令には COMET II の機械語命令、アセンブラ命令、マクロ命令が存在する。このうち COMET II の機械語命令とは表 1に示したとおりである。アセンブラ命令とは CASL II 用アセンブラへの指示を行う命令である。アセンブラ命令は、プログラムの先頭を定義する START 命令、プログラムの終わりを定義する END 命令、メモリ領域を確保する DS, DC 命令がある。マクロ命令は入出力に関する IN, OUT 命令がある。命令にはラベルを付けることが出来る。ラベルの長さは 8 文字以内で英大文字と数字の組み合わせで記述する。

3. C コンパイラ

開発した C コンパイラの構成を図 3に示す。C コンパイラは Windows 上で Cygwin[3]（Windows OS に移植された GNU 開発ツール群）がインストールされている環境で動作する。C コンパイラは C 言語ファイルを受け取り、最初にソースプログラムの前処理（コメントの除去、define 定義の置換、ヘッダファイルのインクルード等）を行う。前処理の結果を一時ファイルとして出力する。なお、前処理には GNU プロジェクトのプリプロセッサ(cpp[4])を利用している。

次に前処理済みの一時ファイルを C コンパイラのフロントエンドへの入力とする。フロントエンドでは字句解析・構文解析・意味解析・記号表管理を行う。フロントエンドが終了すると記号表ファイル（変数や関数の情報）および中間コードファイル（演算子や制御文の情報）を出力する。

C コンパイラのバックエンドでは記号表ファイルと中間コードファイルを受け取り、コード生成および最適化処理を行う。バックエンド処理が終了すると CASL II アセンブリファイルが出力される。

バックエンド終了後の処理はリンカとアセンブラが行う。リンカは、出力された CASL II アセンブリファイルとライブラリファイルの結合を行う。アセンブラは CASL II アセンブリファイルを COMET II 機械語ファイルへと変換する。COMET II の機械語ファイルは COMET II 搭載 FPGA ボードにロード可能なインテル形式の Hex ファイルフォーマットとした。

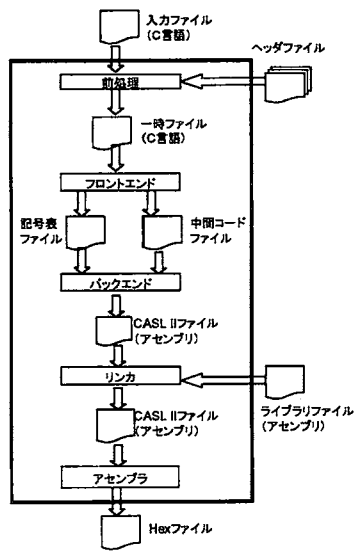

図 3　C コンパイラの構成

本研究で開発した C コンパイラは、マイコン教育機関において初心者が C 言語で簡単にマイコン制御を行い学習する為の利用を想定しているが、十分実用的なレベルの C 言語仕様を持っている。

変数の基本型は 16bit の int 型とし、変数の型修飾は auto,static, register,const,signed,unsigned に対応している。char,double,float と型修飾 short,long および文字列の扱いは現在未対応である。

制御文、配列、ポインタ、関数再帰に対応しているが、構造体、列挙体、共用体は対応していない。これらの機能は組込みソフトウェアの開発でも多用されているため、次期バージョンでは対応予定である。演算子は構造体と共用対に関する演算子以外はすべて対応している。

3.1 フロントエンド

開発したCコンパイラのフロントエンドはC言語ファイルを入力として記号表ファイルと中間コードファイルを出力する。フロントエンドは字句解析プログラム、構文解析プログラム、意味解析・記号表管理プログラムで構成される。図 4にフロントエンドの構成を示す。

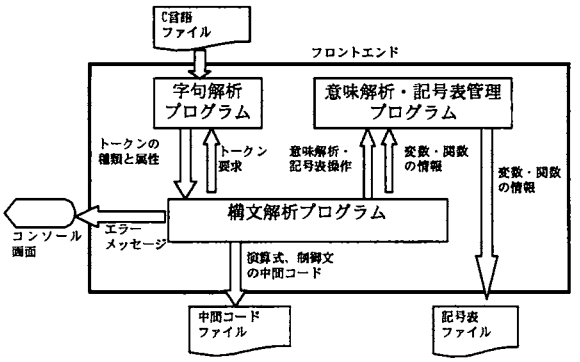

図 4 フロントエンドの構成

字句解析プログラムはソースプログラムをトークン（字句）に分解する。構文解析プログラムはソースプログラムを解析して文法の構造（構文）を明らかにし、演算式と制御文の内容を中間コードファイルに出力する。構文解析中にプログラムが正しい文法で記述されていない場合や意味的な間違いを発見したらエラーメッセージをコンソール画面に出力する。意味解析・記号表管理プログラムは変数や関数の使い方が正しいか確認し、宣言された変数と関数の情報を記号表ファイルに出力する。

3.1.1. 字句解析

字句解析プログラムは構文解析プログラムから呼び出され、C プログラム中のトークン（字句）を解析し、トークンの種類と属性を構文解析プログラムに返す。トークンの種類は ANSI-C の規格を参考にして予約語・識別子・定数・文字列リテラル・演算子・区切記号の 6 種類に分類した。トークンの属性とはそのトークンを補足する情報である。

本研究では字句解析プログラムは GNU プロジェクトの flex[5] ツールを用いて自動生成を行った。flex とはトークン定義を記述した入力ファイルを受け取り、そのトークンを解析する字句解析プログラムを出力するツールである。flex に入力するトークン定義は正規表現であり、flex が出力する字句解析プログラムはC言語である。

3.1.2. 構文解析

構文解析プログラムはソースプログラムが構文として正しいか確認し、演算式や制御文など実行対象となる言語要素を中間コードとして出力する。もし構文エラーを発見した場合はエラーメッセージをコンソール画面に出力する。

構文解析プログラムは GNU プロジェクトの bison[6] ツールを用いて自動生成を行った。bison とは構文定義が記述された入力ファイルを受け取り、目的の構文解析プログラムを出力するツールである。bison に入力する構文定義はバッカス-ナウア記法（BNF）で記述し、bison が出力する構文解析プログラムは C 言語である。本研究では ANSI-C の BNF[7] を参考にして、C コンパイラが扱うC言語サブセットに適応するように構文定義の変更を行った。

3.1.3. 意味解析・記号表管理

意味解析プログラムは構文解析プログラムから呼び出される。構文解析中に式や関数呼び出しを検出した時、その式や関数呼び出しの型や引数が正しく使われているか確認を行う。

記号表管理プログラムも構文解析プログラムから呼び出される。構文解析中に変数の宣言または関数の定義を検出した時、宣言された変数や関数の情報を記号表に登録する。解析が全て終わると記号表の内容を記号表ファイルに出力する。

3.1.4. 記号表ファイル

記号表ファイルにはプログラム中に宣言されたすべての変数と関数の情報が出力される（図 5）。

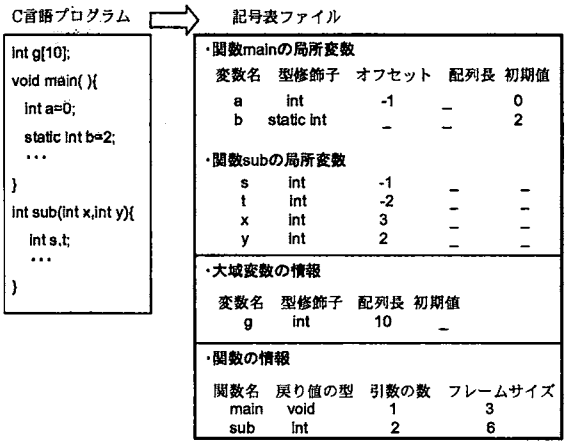

図 5 記号表ファイルの構成

ファイルの先頭から順に、局所変数の情報、大域変数の情報、関数の情報を記述する。変数の情報は変数名、型修飾子（記憶クラス・データ型・符号修飾子等）、オフセット、配列長（配列の場合）、初期値となる。オフセットとはその変数にアクセスする際に必要な基準点からの距離である。局所変数の情報は関数毎にまとめて出力す

る。関数の情報は、関数名、戻り値の型、引数の個数、フレームサイズである。フレームサイズとは関数の実行時に必要となるスタック領域の大きさである。

3.1.5. 中間コードファイル

構文解析中に演算式や制御文を検出したら出現順に中間コードとして中間コードファイルへ出力する。中間コードの形式は、演算式と制御文により異なる。演算式・制御文の中間コードは独自の形式とした。

図 6に C 言語を中間コードに変換する例を示す。図 6の例では演算式(<=,+)と制御文（while）の中間コードが出力されている。

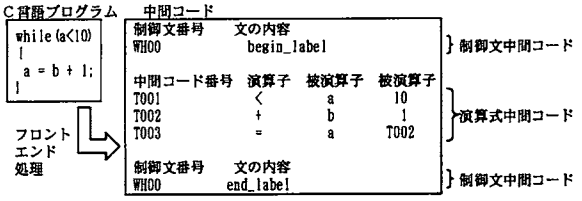

図 6 中間コードへの変換例

演算式の中間コードは、中間コード番号、演算子、被演算子を続けて記述する。中間コード番号は Txxx のように連続した番号が付けられる。演算子は (),[ ],++,--,&,*,+,-,~,!,/,%,+,-,<<,>>,<,<=,>,>=, ==,!=,^,|,&&,||,=,%=,&=,*=,+=,-=,/=,<<=,>>=,^=,~=の 36 種類である。被演算子は定数・変数・中間コード番号のいずれかが入る。被演算子の数は単項演算子の場合は1つで二項演算子の場合は2つである。

演算子の中間コード形式は3つ組の表記[8]を採用している。これは、COMET II の機械語命令に2オペランド形式が多く、2オペランド形式が3つ組み表記に近いためである。これによりバックエンドでのコード生成が容易になる。

制御文の中間コードは、制御文番号、文の内容が記述される。制御文は for,while,switch,if,return に対応している。制御文番号は制御文を表す記号（FO,WH,SW,IF,RE）と番号で記述される。制御文番号に続く文の内容はそれぞれの制御文によって異なる。先の例の begin_label は while 文の先頭位置であることを表す。

3.2. バックエンド

バックエンドはフロントエンドの生成物である記号表ファイルと中間コードファイルを用いて CASL II のアセンブリファイルを出力する。図 7に示すようにバックエンドではまずコード生成を行い、次に最適化の処理を行う。

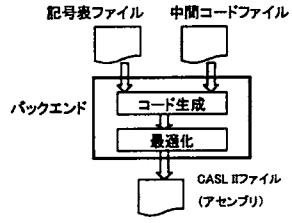

図 7 バックエンドの構成

3.2.1. コード生成

コード生成処理は中間コードを CASL II アセンブリコードへ変換を行う。中間コードファイルを1行単位で読み込み、逐一 CASL II コードへと変換する。その際、変数や関数の情報が必要な場合は記号表ファイルを参照する。中間コードを CASL II コードへ変換する例を図 8に示す。

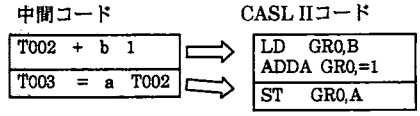

図 8 コード生成例

この例では加算演算子と代入演算子を使っているため、中間コードと CASL II のコード量に差はほとんど無いが、乗除算演算などは CASL II に直接対応する機械語命令が無いため変換した CASL II のコード量は多くなる。

3.2.2. メモリ割り当て

Cコンパイラが割り当てる COMET II の主記憶領域はコード領域、静的変数領域、スタック領域の3領域に分類できる（図 9）。

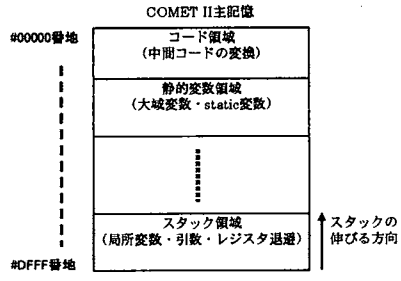

図 9 メモリ割り当て

中間コードを変換した CASL II コードは0番地から始まるコード領域に配置される。コード領域に続いて静的変数領域が配置される。静的変数領域は大域変数および static 宣言された局所変数を CASL II の DS, DC 命令を用いて領域を確保する。

スタック領域はプログラムの実行時に領域が動的に変化する。ス

タック領域はスタックポインタの変化とともに、静的変数領域の方向へ伸びていく。スタック領域の最上位番地（スタックポインタの初期番地）はCOMET IIの実行環境によって異なり、本研究で用いたCOMET IIボードでは#DFFF番地となっている。

3.2.3. 関数フレーム

関数の実行時に必要な情報（局所変数、引数、戻り番地）はスタック領域に動的に配置する。この領域を関数フレームと呼ぶ（図10）。なお、フレームポインタ（FP）はCOMET IIでは専用のレジスタが定義されていないためGR7をフレームポインタとして扱う。

関数の実行中は関数フレームがスタック領域に確保され、関数の実行を終えると関数フレームは開放される。したがって関数の実行前と実行後ではスタック領域は同じ状態になる。なお、関数実行中の局所変数と引数はフレームポインタからのオフセットを用いてアクセスする。

Cコンパイラは関数の先頭に入口コードを、終了位置に出口コードを挿入する。入口コードでは汎用レジスタの退避と関数フレームの構成のための処理を行う。出口コードでは汎用レジスタの復帰と関数フレームの開放のための処理を行う。

関数フレームを構成する手順は、まず関数呼出元で引数をプッシュした後、関数呼び出しを行う。これにより引数と戻り番地がスタックに積まれる。次に呼出し先の関数の入口コードで呼出元関数のフレームポインタをプッシュした後、局所変数領域を確保する。関数フレームの開放はこの逆の手順となる。

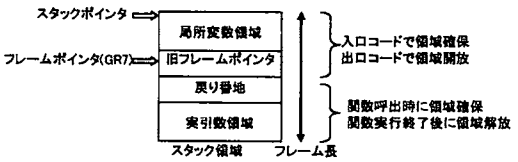

図 10　関数フレームの構造

3.2.4. レジスタ割り当て

Cコンパイラは汎用レジスタのうちGR0からGR3を中間コードの演算用として割り当てる。GR0は関数の戻り値の格納にも利用する。GR4からGR6まではregister宣言された変数用として割り当てる。register宣言された変数は最大3つまでが汎用レジスタに割り当てられる。同時に4つ以上の変数がregister宣言された場合は、4番目の変数以降はauto変数と同じ扱いとなる。GR7は関数のフレームポインタとして扱う。

3.2.5. 最適化

開発したCコンパイラはコード生成プログラムが出力したCASL IIアセンブリコードに対して最適化処理を行う。実装した最適化処理は、参照されないデータのメモリ格納の中止、参照されないラベルの削除、の2つである。一般的なCコンパイラの最適化処理は中間コードに対して処理（並べ替え、置換等）を行うが、開発したCコンパイラはアセンブリコードに対して処理（コード削減）を行う。

今回実装した最適化処理により、最適化前と比較して最大約10%のコード削減が可能となった。また、Cコンパイラの最適化処理がどのようなものかを学習できるように、コンパイラオプションによって最適化処理の有無を切り替え可能にした。

4. リンカ・アセンブラ

バックエンド終了後の処理はリンカとアセンブラが行う（図11）。リンカの役割はコンパイル処理が終わったCASL IIアセンブリファイルとライブラリ関数が含まれるライブラリファイルを結合することである。アセンブラはリンク済みのCASL IIアセンブリコードをCOMET IIボードにロード可能なHex形式に変換する。COMET IIの仕様ではその機械語については特に定められていないが、本研究では利用するCOMET IIボードの機械語と一致させた。

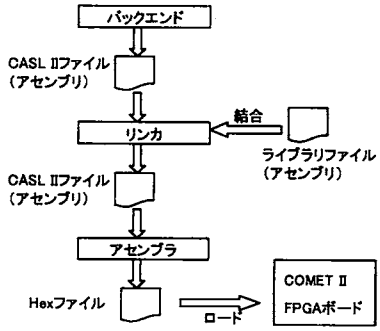

図 11　リンカとアセンブラの処理

5. COMET II ボード・ライブラリ関数

本研究で実際に制御するCOMET IIボードはアンドールシステムサポート株式会社[8]のET2というボードである（図12）。このET2ボードはCOMET IIのCPU機能のほかにLED・スイッチ・タイマ・ADコンバータなどの豊富な周辺デバイスを持つ。ET2ボードのすべての周辺デバイスはCOMET II主記憶のI/O領域と呼ばれる#FF00~#FFFF番地に接続されている。I/O領域に値の読み書きを行うことで入出力操作を行う。

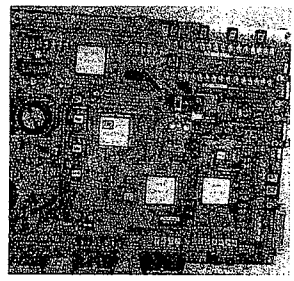

図 12　ET2ボードの写真

本研究では制御を行うET2ボード用のライブラリ関数を開発した。ライブラリ関数はすべてCASL IIアセンブリコードで記述してある。開発したライブラリ関数は、I/O領域に値を書込むwmem関数とI/O領域の値を読込むrmem関数の2種類である。それぞれ表2の形式でC言語から使用できる。wmem関数は引数に書込み先のアドレスと書込む値を指定する。rmem関数は読込み元のアドレスを指定すると値が返却される。

表2 開発したライブラリ関数

I/O領域書込み関数	void wmem(int adr, int val);
I/O領域読込み関数	int rmem(int adr);

6. C言語によるCOMET IIボードの制御実験

実際にC言語でCOMET IIボードの制御ができること実証するため、開発したCコンパイラを用いて実験を行った。記述したC言語の内容はエラトステネスの篩い法で100個の素数を求め、順番にET2ボード上の7セグメントLEDにダイナミック点灯で表示するものである。

今回実験を行った手順は、まずC言語で目的のプログラムの記述を行う。次に開発したCコンパイラでコンパイルを行い、最後に出力したhexファイルをボードへ転送して実行、という3段階のステップである。図13に実験で使用したC言語ファイルと中間コードファイル、リンク後のCASL IIアセンブリファイルおよびHexファイルの概要を示す。

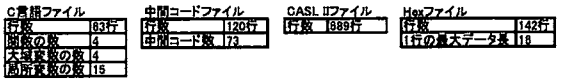

図13 Cコンパイラが処理するファイルの概要

7. まとめ

開発したCコンパイラは、マイコン制御で頻繁に利用されるC言語の文法や機能に対応し、Cコンパイラとして実用的なレベルである。また、開発したCコンパイラとCOMET IIボードを利用して、実際にC言語でCOMET IIの制御が可能であることの確認を行った。

本研究の成果として、現在までアセンブリ言語のみで行われているCOMET IIを用いたマイコン教育や学習において、C言語の扱いが可能になったことが挙げられる。C言語の扱いが可能になったことにより、アセンブリ言語の知識が無い初心者でもC言語でCOMET IIマイコンの制御が可能になる。また、学習者がC言語を用いた実際の組込みソフトウェアの開発手法が体験可能となる。

本研究の今後の課題として、Cコンパイラの複数入力ファイルや複数ライブラリファイルへの対応、構造体と列挙体および共用体の対応、char型の扱いと文字列への対応、double,floatの扱いと浮動小数点演算の対応、コード最適化処理の改良、レジスタ割り当ての工夫、などが挙げられる。

謝辞

今回COMET II用のCコンパイラ開発の端緒を与えてくださいましたアンドールシステムサポート株式会社様と宇賀神孝氏に深く感謝いたします。

文献

[1] 河合 一慶, 宮内 新, 荒井 秀一 "COMET-II 互換プロセッサによるCPU設計演習環境の開発" 第1回 情報科学技術フォーラム(FIT2002) 2002

[2] 岡崎保憲, 泉宏志, 村越英樹, 森久直, 坂巻佳寿美, 山本大介, 波多野八州夫, 村山彰一, 宇賀神孝 "高度情報化人材育成用ASICマイコン教材の開発" 東京都立科学技術大学紀要, 第15巻, pp.145-146 2001

[3] Cygwin Information and Installation
http://www.cygwin.com/

[4] The C Preprocessor
http://gcc.gnu.org/onlinedocs/cpp/

[5] flex: The Fast Lexical Analyzer
http://www.gnu.org/software/flex/

[6] Bison - GNU parser generator
http://www.gnu.org/software/bison/

[7] 「プログラミング言語C」第2版 B・W・カーニハン／D・M・リッチー著 石田晴久訳 共立出版 1989

[8] 「コンパイラ構成法」 原田賢一著 共立出版 1999

[9] アンドールシステムサポート株式会社
http://www.andor.jp/