Vol.2012-CE-117 No.9
2012/12/9

機械語教育用マイコンTeCとシリーズ化した教育用パソコンTaCのアーキテクチャ

重村　哲至[1,a)]　山田　健仁[1]

概要：オペレーティングシステムやコンパイラを学ぶ学生に，これらの実装例や実行結果を示すために使用する目的で，とてもシンプルな 16 ビットのパーソナルコンピュータ TaC を開発した．TaC のアーキテクチャは，機械語教育用に開発され 10 年の使用実績がある TeC のアーキテクチャと似せてある．TeC で機械語を学習した学生は，TaC のアーキテクチャを簡単に理解することができ，すぐに TaC のアセンブリ言語で記述されたオペレーティングシステムの内部や，コンパイラの出力を読んで理解することが可能である．TaC のアセンブリ言語でオペレーティングシステムの簡単なディスパッチャを記述し，その概要を TeC を用いて学習した知識で理解できることを確認した．また，コンパイラの出力も同様に理解可能なことと，コンパイラが出力するアセンブリ言語プログラムをコンパクトな機械語に変換できることを確認した．

1. はじめに

TeC(Tokuyama Educational Computer)[1], [2], [3], [4] は高専低学年の学生や高校生が，ノイマン型コンピュータの動作原理を理解するために使用する教材として開発された 8 ビットのシンプルなマイコンである．TeC を教材にコンピュータの機械語を勉強することができる．学生は機械語を学習するうちに，ノイマン型コンピュータの動作原理を体感的に理解することができる．既に 10 年の使用実績があり動作原理の理解について効果を確認済みである [5]．

しかし，TeC のアドレス空間は 8 ビットであるため，低学年の機械語演習には十分であるがオペレーティングシステム (以下では OS) や高級言語を使用した演習に使用することができない．そのため，低学年で学んだ TeC のアーキテクチャに関する知識を，その後の学習で十分に活用できない問題がある．

そこで，TeC のアーキテクチャを 16 ビットに拡張したシンプルなパーソナルコンピュータ TaC(Tokuyama Advanced Computer)[6] を開発した．TaC は，簡単な OS を搭載することや，高級言語で記述したプログラムを実行することが可能である．OS の構造を説明する際に実装例として使用したり，組込みシステムの設計・実装用の教材として利用できる．すでに，組込みシステム実装用の教材として 4 年の実績がある．

[1] 徳山工業高等専門学校
Tokuyama College of Technology
[a)] sigemura@tokuyama.ac.jp

図 1　TeC7
Fig. 1　Tokuyama Educational Computer (Ver.7).

本稿では，TeC に似せた 16 ビット・パーソナルコンピュータ TaC のアーキテクチャとハードウェア構成について報告する．

2. TeC

最新型の TeC(TeC7) の写真を**図 1** に示す．基板の左側に搭載された FPGA にコンピュータの機能が作り込んである．基板の右半分はユーザが 2 進数で操作するコンソールパネルである．

2.1 特徴

TeC はノイマン型コンピュータの動作原理を理解するための教材である．コンピュータの内部をプログラムの介在なしに，2 進数のまま観察・操作することができるコンソールパネルを装備している．学生はコンソールパネルから 2

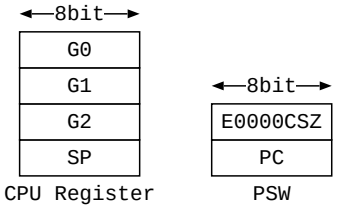

図 2 TeC の CPU レジスタと PSW
Fig. 2 CPU register and PSW of TeC.

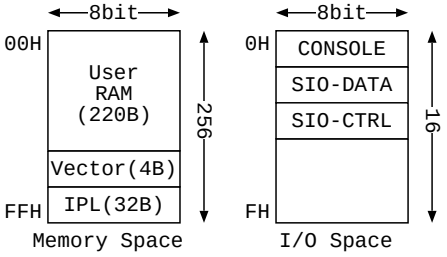

図 3 TeC アドレス空間
Fig. 3 TeC address space.

進数で入力したプログラムを，CPU レジスタや PSW，主記憶等を 2 進数で観察しながら実行する．これを繰り返すうちに，ノイマン型コンピュータの主要な特徴である，2 進法，逐次実行，ストアードプログラム方式を体感的に理解することができる．

2.2 アーキテクチャ

TeC はコンソールパネルを最も重要な装置と位置付け，マイコンのアーキテクチャはコンソールパネルの都合に合わせて設計した．コンソールパネルの大きさと操作性の制約から，データもアドレスも 8 ビット構成のコンピュータになった．

2.2.1 CPU レジスタと PSW

図 2 に CPU レジスタと PSW の構成を示す．図中の E0000CSZ は，割込み許可（E），キャリー（C），符号（S），ゼロ（Z）の各フラグを意味する．コンソールパネルで 2 進数を表現するランプやスイッチの桁数が 8 であるので，CPU レジスタ等も全て 8 ビットの構成になっている．

2.2.2 メモリ空間と I/O 空間

図 3 にメモリ空間と I/O 空間の概略を示す．メモリ空間は 00H 番地から FFH 番地の 256 バイトである．その中に 4 バイトの割り込みベクタと，32 バイトの IPL ROM が配置される．そのため，ユーザ用のメモリは 220 バイトしか残らない．

I/O 空間は 0H 番地から FH 番地の 16 バイトである．ここに，シリアル入出力インタフェース (SIO) 等のポートが配置される．

2.2.3 命令セット

機械語命令は**表 1** に示す 29 種類，アドレッシングモードは 4 種類である．全ての転送命令と演算命令で全てのアドレッシングモードが使用可能な，直交性の高い設計になっている[*1]．

表 1 TeC 機械語命令
Table 1 TeC instruction set.

命令種類	具体的な命令
転送	CPU レジスタへ (LD)，メモリへ (ST)
演算	足算 (ADD)，引算 (SUB)，比較 (CMP)， 論理演算 (AND，OR，XOR)
シフト	左右，算術/論理 (SHLA，SHLL，SHRA，SHRL)
ジャンプ	無条件 (JMP)，条件 (JZ，JC，JM)， サブルーチン呼出し (CALL)，戻り (RET)
入出力	入力 (IN)，出力 (OUT)
スタック	一般 (PUSH，POP)，PSW(PUSHF，POPF)
割込み	禁止 (DI)，許可 (EI)，戻り (RETI)
その他	何もしない (NO)，停止 (HALT)

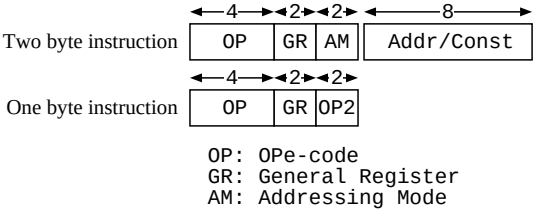

図 4 TeC 命令コード
Fig. 4 TeC instruction code.

図 4 に機械語コードの構成を示す．機械語命令には 2 バイトのものと 1 バイトのものがある．OP，OP2 フィールドは命令の種類を表すために使用される．GR フィールドは演算等の対象となる CPU レジスタの指定に使用される．AM フィールドはアドレッシングモードの指定に使用される．2 バイト命令では，第 2 バイトがアドレスや定数を格納するために使用される．

TeC では学生がハンドアセンブルによって作った機械語をコンソールパネルから 2 進数で打ち込むことを想定している．そのため，ハンドアセンブルがしやすいように単純で例外の少ない命令フォーマットになっている．

2.2.4 アドレッシングモード

2 バイト命令の AM フィールドはアドレッシングモードを指定するために使用する．AM フィールドが 2 ビットなので，ダイレクト，G1 インデクスド，G2 インデクスド，イミディエイトの 4 種類しか表現できない．G0 インデクスド，SP 相対は存在しない．

ニーモニックでは，アドレッシングモードを次の例のように表現する．（ADD 命令の場合を例にする．）

（1）ダイレクト: `ADD G0,A`
（2）G I インデクスド: `ADD G0,A,G1`
（3）イミディエイト: `ADD G0,#100`

[*1] ST 命令でイミディエイトは使用できない．

2.3 TeC の限界

TeC は割込み機構やシリアル入出力等も備えており，OS やコンピュータアーキテクチャを学ぶ際に必要な知識の多くを教えることに役立つ．また，220 バイトのプログラムメモリがあれば，機械語のステート数で時間を計りスピーカを駆動して音楽を鳴らす電子オルゴールのような，学生の演習でアセンブリ言語を用いて記述するには高度なプログラムも実行できる．低学年で行う機械語やアセンブリ言語の演習には十分である．

しかし，高級言語を用いた場合は，記述量あたりの生成される機械語プログラムの量が多い．学生が記述したプログラムでも簡単に 220 バイトを超えてしまい学生の演習でも実用にならない．更に，OS を実装した実例を見せたり，コンパイラが出力したコードを実行して見せたりするためにもメモリが不足する．

3. TaC

TaC の開発は 2005 年から開始し，FPGA(Xilinx 社 Spartan-3) のトレーニングボードを流用して実用化した．並行して 2006 年から高級言語 C--[7] の定義とクロス開発環境の整備を進め，2009 年度からは TaC 上で C--言語を用い MP3 プレーヤアプリケーションを作成する演習テーマを授業に取り入れた [8]．

現在の TaC は，2011 年に設計し直した新しい TeC(TeC7:図 1) に寄生している．TeC7 では Xilinx 社の最新の FPGA(Spartan-6)[9] を使用している．Spartan-6(XC6SL9) は，従来 TeC で使用していた Spartan-2(XC2S30) と比較して約 10 倍のロジックセルを搭載している．

TeC だけでは使い切れなかったロジックセルを使用し，同じ FPGA に TaC を同居させることが可能になった．ボード上のジャンパーの設定により TeC と TaC を切り替える．TaC の主記憶 (約 64 キロバイト) も FPGA 内部のブロック RAM で賄うことができた．基板には，TaC モードで使用する PS/2，VGA，μSD 等のコネクタが追加されている．

3.1 特徴

TaC は高級言語で記述したアプリケーションや OS が動作可能なコンピュータである．キーボード，ディスプレイやメモリカードを接続し，簡単なパーソナルコンピュータとして使用できる．また，実行モードを持ちメモリ保護等の機能を持った OS を実行可能である．よく似たシステムとして MieruPC[10] がある．MieruPC と比較すると TaC の方が「(1) より小規模である」，「(2)TeC とシリーズ化をしている」，「(3) セルフ開発環境の実装を目指している」等の違いがある．

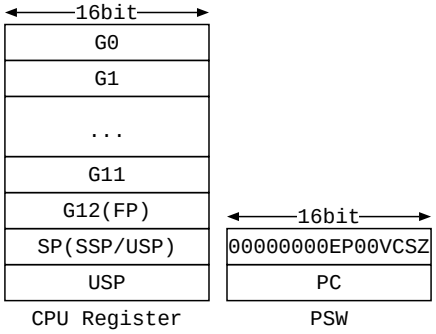

図 5 TaC の CPU レジスタと PSW
Fig. 5 CPU register and PSW of TaC.

3.2 アーキテクチャ

TaC アーキテクチャは，TeC と似せたままアドレス空間を広くすることを目指した．TeC で 8 ビットデータ，8 ビットアドレスであったものを，16 ビットデータ，16 ビットアドレスに変更することから設計を開始した．

3.2.1 CPU レジスタと PSW

図 5 に CPU レジスタと PSW の構成を示す．レジスタは全て 16 ビットに変更された．PSW には，特権モード (P)，オーバーフロー (V) の二つのフラグが追加された．

CPU レジスタは 15 本になった．G12 はフレームポインタ (FP) 用の特別な汎用レジスタである．スタックポインタ (SP) はスーパバイザ用の SSP とユーザ用の USP の 2 本になった．どちらのスタックポインタが使用されるかは CPU の実行モードにより切り換わる．USP は OS が初期値を与えるために，特権モードでもアクセス可能である．

3.2.2 メモリ空間と I/O 空間

図 6 にメモリ空間と I/O 空間の概略を示す．メモリ空間は 0000H 番地から FFFFH 番地の 64 キロバイトである．バイト単位でアドレス付けされているが 16 ビット単位のアクセスが基本である．16 ビットデータの上位 8 ビットを偶数番地に，下位 8 ビットを奇数番地に格納する．バイト単位のアクセスは特殊なアドレッシングモードでのみ可能である．

0000H 番地からの 56 キロバイトが自由に使用できる領域である．ここに OS とユーザプログラムを格納する．E000H 番地からの 4 キロバイトの奇数番地にキャラクタ V-RAM が配置される．偶数番地は使用されない．F000H 番地からの 4064 バイトに，メモリカードの FAT16 ファイルシステムから OS カーネルファイルを読み込む IPL を配置した．FFE0H 番地からの 32 バイトは，16 種類の割込・例外に対応するベクタである．

I/O 空間は 00H 番地から FFH 番地の 256 バイトである．メモリ空間と同様にバイト単位のアドレス付けがされている．ここに，タイマー，シリアル入出力，PS/2 キーボード等のインタフェース回路が配置される．

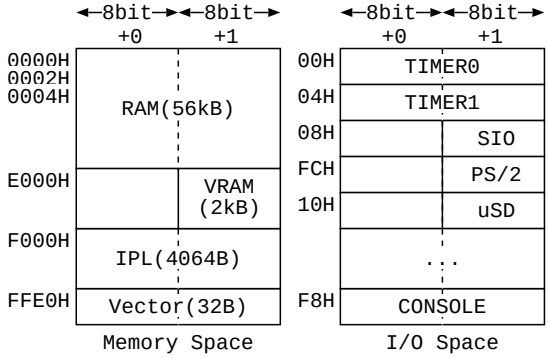

図 6 TaC アドレス空間
Fig. 6 TaC address space.

表 2 TaC 機械語命令
Table 2 TaC instruction set.

命令種類	具体的な命令
転送	CPU レジスタへ (LD)，メモリへ (ST)
演算	足算 (ADD)，引算 (SUB)，比較 (CMP)， 論理演算 (AND，OR，XOR)， 掛算 (MUL，MULL)，割算 (DIV，DIVL)， 余り (MOD)，アドレス足算 (ADDS)， シフト (SHLA，SHLL，SHRA，SHRL)
ジャンプ	無条件 (JMP)，条件 (JZ，JC，JM，JO， JGT，JGE，JLE，JLT，JNZ，JNC，JNM， JNO，JHI，JLS) サブルーチン呼出し (CALL)，戻り (RET)
入出力	入力 (IN)，出力 (OUT)
スタック	一般 (PUSH，POP)
割込み	禁止 (DI)，許可 (EI)，戻り (RETI)
その他	割出し (SVC)，何もしない (NO)，停止 (HALT)

3.2.3 命令セット

機械語命令は**表 2** に示す 45 種類である．TeC と比較して，掛算割算命令，アドレス計算用の足算命令，システムコール用の割出し命令を追加した．C--言語の比較演算を効率良く実行できるように，条件ジャンプ命令のレパートリーを 3 種類から 14 種類に増やした．TeC のシフト命令は 1 ビットシフト専用だったが，C--言語のシフト演算も効率良く実行できるように TaC では第 2 オペランドでシフトビット数を指定できるように変更した．

図 7 に機械語コードの典型的な構成を示す．命令コードには 2 ワードのものと 1 ワードのものがある．TeC では演算命令は必ず 2 バイトであったが，TaC ではアドレッシングモードにより 2 ワードの場合と 1 ワードの場合がある．OP，OP2 フィールドが命令の種類を表す．AM フィールドは 8 種類のアドレッシングモードを表現する．GR フィールドは演算対象の CPU レジスタを表現する．GR フィールドが 4 ビットになったので CPU レジスタの数を多くすることができた．XR フィールドはインデクスレジスタを指定するフィールドである[*2]．

[*2] 必ずしもインデクスレジスタではなく，ソースレジスタだったりベースレジスタだったりすることもあるが，本稿では便宜上インデクスレジスタと呼ぶ．

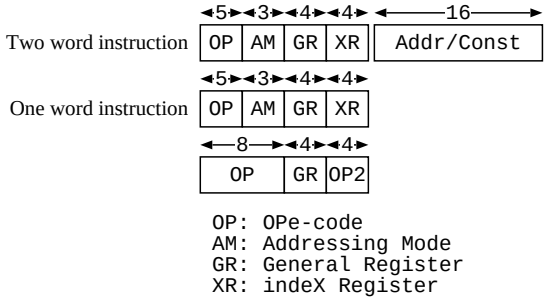

図 7 TaC 命令コード
Fig. 7 TaC instruction code.

AM フィールドと XR フィールドが独立したので全ての CPU レジスタをインデクスレジスタとして使用できるようになった．これによって，高級言語のコンパイラが作りやすくなった．

3.2.4 アドレッシングモード

転送命令と演算命令は，アドレッシングモードと組合せて使用できる．AM フィールドが 3 ビットあるので 8 種類のアドレッシングモードを持つことができる．XR フィールドが独立したこともあり，TeC と比較して非常に強力な機能をもつようになった．

このように大幅な変更を行うと TeC で学んだ知識を活かすことができなくなる危惧がある．しかし，64 キロバイトのメモリ空間に OS とアプリケーションを格納するためにはメモリの節約が肝要である．なるべく多くの命令が 1 ワードになるように，多くのアドレッシングモードを追加することにした．

以下にアドレッシングモードの一覧を示す．この中で，(1)，(2)，(3) が 2 ワード命令，他は 1 ワード命令である．実効アドレスは (8) 以外では偶数である必要がある．

(1) **ダイレクト**：命令コードの第 2 ワードの値が実効アドレスを表現する．
(2) **インデクスド**：命令コードの第 2 ワードの値と，XR フィールドで表現されるインデクスレジスタ値の和が実効アドレスを表現する．
(3) **イミディエイト**：命令コードの第 2 ワードがソースオペランドになる．
(4) **G12(FP) 相対**：XR フィールの値を 4 ビット符号付き定数 (C) とみなし，$C \times 2 + FP$ が実効アドレスを表現する．C--言語が，関数のローカル変数や仮引数をアクセスするために使用する．
(5) **レジスタ・レジスタ**：XR フィールドで表現されるレジスタがソースオペランドになる．
(6) **ショート・イミディエイト**：XR フィールの値を 4 ビット符号付き定数 (C) とみなし，それ

を 16 ビットに符号拡張した値がソースオペランドになる．

(7) **レジスタ・インダイレクト：**XR フィールドで表現されるレジスタの値が実効アドレスになる．

(8) **バイト・レジスタ・インダイレクト：**XR フィールドで表現されるレジスタの値が実効アドレスになる．実効アドレスの内容は 8 ビット単位でアクセスされる．ST 命令では CPU レジスタの下位 8 ビットがメモリに書き込まれ，LD 命令では 00H を上位 8 ビット，メモリ値を下位 8 ビットにした 16 ビットデータが CPU レジスタに読み込まれる．演算命令でも LD 命令と同様な 16 ビットデータが演算に使用される．C--言語で文字型配列のアクセスに使用される．

3.3 TeC との互換性

TaC は，TeC と比較してかなり複雑な命令セットアーキテクチャを持つことになった．このままでは当初目的とした「TeC で学習した知識を活かすこと」ができなくなる．

3.3.1 追加命令

新たに追加された命令は，乗除算命令 5 種類，アドレス足算命令，条件ジャンプ命令 11 種類，SVC 命令である．増加したのは主に乗除算命令と条件ジャンプ命令であり，不自然な命令は少なく，学生は便利な命令が増えたと考えるだけで受け入れることができる．

3.3.2 追加アドレッシングモード

アドレッシングモードの増加は問題である．3.2.4 で述べたように複雑なモードをたくさん持つようになった．しかし，追加された中で「(5) レジスタ・レジスタ」と「(8) バイト・レジスタ・インダイレクト」を除くアドレッシングモードは，存在をユーザから隠すことが可能である．アセンブラが自動的にアドレッシングモードを選択することにより，ソース・プログラムでは従来のアドレッシングモードを使用しているように見せることができる．

TeC はコンソールパネルから学生が機械語を 2 進数で打ち込むことを想定していた．それに対し TaC は，OS やコンパイラの学習や高級言語を使用したアプリケーションの開発用であるので，機械語命令の細部をアセンブラが隠しても支障は少ないと考えられる．次に，アドレッシングモードの自動選択例を示す．

```
ADD G0,0,G1     : レジスタ・インダイレクト
ADD G0,2,G1     : インデクスド (2 ワード)
ADD G0,-16,G12  : G12(FP) 相対
ADD G0,14,G12   : G12(FP) 相対
ADD G0,-18,G12  : インデクスド (2 ワード)
ADD G0,16,G12   : インデクスド (2 ワード)
ADD G0,#-8      : ショート・イミディエイト
ADD G0,#7       : ショート・イミディエイト
ADD G0,#-9      : イミディエイト (2 ワード)
ADD G0,#8       : イミディエイト (2 ワード)
```

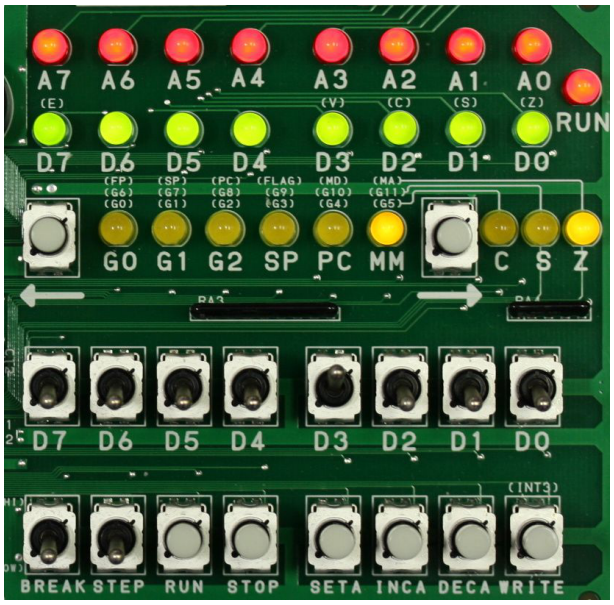

図 8 TaC のコンソールパネル
Fig. 8 TaC console panel.

ニーモニックでは TeC でも使用したインデクスドとイミディエイトの 2 種類のアドレッシングモードしか使用していないように見えるが，実際は 5 種類が使用されている．アドレスや定数が 4 ビットで表現できる場合は，新しく追加された 1 ワード命令が自動的に選択される．アドレスや定数にラベルを使用すると自動選択が難しくなるが，C--コンパイラが出力するアセンブリコードでは 1 ワード命令が選択できる可能性があり，かつ，ラベルが用いられる例は少ない．

3.4 コンソールパネル

TeC7 を TaC モードで使用する場合もコンソールパネル (**図 8**) を使用できるようにした．これにより，OS の内部までステップ実行することができる．

8 ビット用のコンソールパネルを 16 ビットの TaC で使用する．アドレスランプ (最上段の 8 個の LED) とデータランプ (上から 2 段目の 8 個の LED) を合わせて 16 ビットデータを表示するようにした．

何を表示中であるかを G0〜MM のランプと C，S，Z のランプの組合せで表現する．CPU レジスタが増加したため，また，アドレスとデータを同時に表示できなくなったため，表示されるものの種類が格段に増加した．そこで，ランプの組合せを用いて表現することにした．これらのランプの上部に TaC モードでの意味をシルク印刷してある．

データの入力には下から 2 段目の 8 個のスイッチを用いる．データの書込操作をする度にランプ表示されたデータ

が 8 ビットシフトされ，下位 8 ビットに新しい値がスイッチから入力される．

4. 実装結果

まず，OS の授業で使用する場合を想定し，TaC アーキテクチャ用のディスパッチャを記述し，TeC の知識で理解できるか検討した．次に，コンパイラの授業で使用する場合を想定し，C--コンパイラが出力するアセンブリ言語プログラムを TeC の知識で理解できるか検討した．また，コンパイラの出力では効率の良い 1 ワード命令が多く用いられるかも調べた．

その結果，CPU レジスタが増設されたこと，レジスタ・レジスタのアドレッシングモードが追加されたことを学生が理解していれば，必要に応じて僅かの新命令を教えるだけでこれらのアセンブリ言語プログラムが理解できることが分かった．また，C--コンパイラの出力では 1 ワード命令が多用され，プログラムサイズが大幅に縮小されることが確認できた．

4.1 ディスパッチャ

簡易的な組込み用の OS のディスパッチャを TaC のアセンブリ言語で記述した例を**図 9** に示す．現在のプロセスは dsptch をコールする．この時，PC は現在のプロセスのスタックに保存される．その後，0000 番地の LD 命令で CPU のフラグの値を G0 レジスタに値をコピーし G0 レジスタを PUSH することで，PSW のスタックへの保存が完了する．次に CPU レジスタを順番にスタックに保存し，最後に現在のプロセスコントロールブロックに SP を保存する (0018 番地〜001C 番地)．保存されない CPU レジスタは C--の呼出し規則で保存が不要なものである．ここまでで現在のプロセスのコンテキストの保存が完了する．

start からのプログラムは新しいプロセスを選択しコンテキストを復元する．001E 番地の CALL 命令で呼出す selProc 関数が次に実行するプロセスを選択し，プロセスコントロールブロックのアドレスを curProc にセットする．次に，プロセスコントロールブロックから SP を復元し，スタックから CPU レジスタの値を POP する．最後に RETI 命令を用いて PSW を復元する．

0000 番地の FLAG を扱う LD 命令を除くと，レジスタ数が多くなったことさえ知っていれば TeC の知識でプログラムを読むことができる．

4.2 コンパイラの出力

C--言語で記述したプログラムの例を**図 10** に，そのプログラムを C--コンパイラに通した出力を**図 11** に示す．このプログラムは引数で渡された配列の内容の合計を求めて返すものである．関数引数はスタックに積んで渡し，関数値は G0 レジスタに入れて返す．

Addr	Code		Label	Nemonic	
0000	1400		dsptch	LD	G0,FLAG
0002	C000			PUSH	G0
0004	C0C0			PUSH	FP
0006	C030			PUSH	G3
0008	C040			PUSH	G4
000A	C050			PUSH	G5
000C	C060			PUSH	G6
000E	C070			PUSH	G7
0010	C080			PUSH	G8
0012	C090			PUSH	G9
0014	C0A0			PUSH	G10
0016	C0B0			PUSH	G11
0018	0800	0000		LD	G0,curProc
001C	16D0			ST	SP,0,G0
001E	A800	0000	start	CALL	selProc
0022	0800	0000		LD	G0,curProc
0026	0ED0			LD	SP,0,G0
0028	C4B0			POP	G11
002A	C4A0			POP	G10
002C	C490			POP	G9
002E	C480			POP	G8
0030	C470			POP	G7
0032	C460			POP	G6
0034	C450			POP	G5
0036	C440			POP	G4
0038	C430			POP	G3
003A	C4C0			POP	FP
003C	D400			RETI	

図 9 ディスパッチャ
Fig. 9 Dispatcher.

```
int sum(int[] n, int m) {
  int s = 0;
  for(int i=0;i<m;i=i+1)
    s = s + n[i];
  return s;
}
```

図 10 C--プログラムの例
Fig. 10 Example of C-- program.

コンパイラの出力では，引数のアクセスに FP 相対モードのアドレッシングが使用されているのが分かる．ローカル変数は G3，G4 に割当てられている．ADDS 命令は配列要素のアドレスを計算するための命令である．0014 番地の例では G0 + G4 × 2 を計算し結果を G0 に代入する．

ADDS 命令，JGE 命令，レジスタ・レジスタモードのアドレッシングは TeC に無いのでこれらに慣れる必要があるが，他は TeC の知識で理解できる命令ばかりである．また，000E，0022 番地のジャンプ命令以外は全てが 1 ワード命令である．1 ワード命令を用いてプログラムサイズを縮小することに成功している．

Addr	Code		Label	Nemonic	
0000	C0C0		sum	PUSH	FP
0002	0CCD			LD	FP,SP
0004	C030			PUSH	G3
0006	C040			PUSH	G4
0008	0D30			LD	G3,#0
000A	0D40			LD	G4,#0
000C	2B43		.L1	CMP	G4,6,FP
000E	A050	0026		JGE	.L2
0012	0B02			LD	G0,4,FP
0014	4C04			ADDS	G0,G4
0016	0E00			LD	G0,0,G0
0018	1C03			ADD	G0,G3
001A	0C30			LD	G3,G0
001C	0C04			LD	G0,G4
001E	1D01			ADD	G0,#1
0020	0C40			LD	G4,G0
0022	A0F0	000C		JMP	.L1
0026	0C03		.L2	LD	G0,G3
0028	C440			POP	G4
002A	C430			POP	G3
002C	C4C0			POP	FP
002E	D000			RET	

図 11 コンパイラの出力
Fig. 11 Output of compiler.

4.3 実際的な例

TaC の IPL はメモリカード (μSD) の FAT16 ファイルシステムを解析し，ルートディレクトリから “KERNEL.BIN” ファイルを見つけ出し，主記憶に読み込んで実行するものである．メモリカードとは SPI を用いて接続される [11].

IPL は，メモリカード・ドライバ，最低限の FAT16 ファイルシステムを含んでおり，ソース・プログラムはアセンブリ言語約 100 行と C--言語約 650 行からなる．コンパイル後 IPL ROM に書き込む状態で，機械語プログラム 4000 バイト，初期化データ 26 バイト，非初期化データ 1072 バイトの大きさがあった．

このような高機能な IPL を 4 キロバイト程度の ROM に格納することができた．TaC のアーキテクチャは OS やコンパイラの実装例などを学生に示す目的に十分な実用性があると言える．

5. おわりに

TeC を用いて機械語を学習した学生が，OS やコンパイラの勉強をする際に教材として使用できるコンピュータ TaC を開発した．TaC は高級言語 C--で記述した OS やアプリケーションプログラムを実用的に実行できる．

TaC が実用的に使用できるように，メモリ使用効率が高い命令セットアーキテクチャを採用した．メモリ効率を高くするために 1 ワード命令を生成する多くのアドレッシングモードが追加されている．そのため，TaC の命令セットは複雑なものになってしまった．そのままでは TeC で学んだ知識を活かすことが難しい．

そこで，アドレッシングモードをアセンブラが自動的に選択するようにし，アセンブリ言語ソース・プログラムの段階では TeC とあまり差がないように見せることにした．

その結果，学生が TeC の知識を活かして，TaC 用に書かれたアセンブリ言語プログラムを理解することが可能になった．また，1 ワード命令を効率良く使用してプログラムサイズを縮小できるようになった．特に C--コンパイラが出力するコードでは 1 ワード命令が多用されるようになった．

謝辞 本研究は JSPS 科研費 22500833 の助成を受けたものである。

参考文献

[1] 重村哲至，山田健仁，新田貴之，力 規晃，原田徳彦，三木 幸：コンソールパネルを持つビデオカセットサイズの教育用マイコンとクロス開発環境，情報処理学会研究報告，コンピュータと教育，2004–CE–74，pp.25–32(2004).
[2] 重村哲至，守川和夫，力 規晃，新田貴之，原田耕治，山田健仁：教育用マイコンボードを用いた HDL 演習環境の実現，情報処理学会研究報告、コンピュータと教育，2005–CE–78，pp.43–48(2005).
[3] 重村哲至，山田健仁，新田貴之，力 規晃，原田徳彦，三木 幸，守川和夫：コンソールパネルを持つ教育用マイコンの開発と授業への適用事例，論文集「高専教育」，第 28 号，pp.77–82(2005).
[4] TeC 開発プロジェクト：教育用コンピュータ（TeC）のサイト，http://tec.tokuyama.ac.jp/TeC/
[5] 重村哲至, 古川達也, 相知政司, 林 敏浩：コンソールパネルを持つ機械語教育用マイコンの開発と授業への応用，情報処理学会論文誌，第 48 巻，第 9 号，pp.3318–3327(2007).
[6] 田中利康，重村哲至：OS を実装可能な教育用コンピュータの設計・開発，電気・情報関連学会中国支部第 56 回連合大会講演論文集，教育・研究 II，p.109(2005).
[7] 重村哲至，古川達也，相知政司，林 敏浩，土橋 塁：教育用システム記述言語 C--，教育システム情報学会研究報告，Vol.22，No.4，pp. 75-80(2007).
[8] 重村哲至，山田健仁，守川和夫，新田貴之，柳澤秀明，原田徳彦，宇野俊夫：教育用システム記述言語 C--の実用化，論文集「高専教育」，第 33 号，pp.821–826(2009).
[9] ザイリンクス株式会社：Spartan-6 FPGA ファミリ，http://japan.xilinx.com/products/silicon-devices/fpga/spartan-6/index.htm
[10] MieruPC 株式会社：MieruPC 株式会社，http://www.mierupc.com/
[11] 岡田浩人，横山智弘：マルチメディアカード & SD メモリカードの概要，Interface 増刊フラッシュ・メモリ・カードの徹底研究，pp.71–95，CQ 出版株式会社 (2006).