面積削減を目的としたデータ圧縮手法

門前, 淳
九州大学大学院システム情報科学府

大隈, 孝憲
九州大学大学院システム情報科学府

安浦, 寛人
九州大学大学院システム情報科学府

出版情報：電子情報通信学会技術研究報告ICD2001, pp.17-21, 2002-03. 電子情報通信学会ICD研究会
バージョン：
権利関係：

面積削減を目的としたデータ圧縮手法

門前　淳[†]　大隈　孝憲[†]　安浦　寛人[†]

† 九州大学 大学院 システム情報科学府
〒 816-8580 福岡県春日市春日公園 6-1
E-mail: {monzen, okuma, yasuura}@c.csce.kyushu-u.ac.jp

あらまし　本稿では，組み込みシステムにおける命令 ROM の面積削減を目的としたオブジェクトコードの圧縮手法を提案する．文字列圧縮法である Byte Pair 符号化はアルゴリズムがシンプルであり，任意の場所から展開ができる性質を持つ．命令 ROM を圧縮するための要求をピックアップし，より効率的に命令 ROM を圧縮・展開できるよう Byte Pair 符号化を拡張した．Byte Pair 符号化を使ったオブジェクトコード圧縮の実験をし，ROM 面積の削減に効果があることを確認した．

キーワード　組み込みシステム，コード圧縮，ROM 面積削減

A Code Compression Technique for Chip Area Minimization

Atsushi MONZEN[†], Takanori OKUMA[†], and Hiroto YASUURA[†]

† Department of Computer Science and Communication Engineering, Kyushu University
6-1 Kasuga-Koen,a Kasuga-shi, Fukuoka 816-8580 Japan

Abstract　In this report, we propose an object-code compression technique for instruction ROMs in embedded systems. The Byte Pair Encoding ,which is a kind of dictionary compression methods, has a very simple encoding/decoding scheme and can be decoded in any positions. We surmmarize necessities of compressing ROMs and extend the Byte Pair Encoding to encode/decode more effectively. We demonstrate the code compression for actual programs by the Byte Pair Encoding on the reduction of ROM area.

Key words　Embedded System, Code Compression, ROM Area Minimization

1. はじめに

組み込みシステムは多機能化・高性能化を続けている．それに伴いアプリケーションの規模は増大し，オブジェクトコード・サイズは増加している．

オブジェクトコードのサイズを縮小するためにオブジェクトコードを圧縮し，命令の実行時に展開（伸張）する手法がある．現在までに多くの手法が提案されている [9].

オブジェクトコードを圧縮するタイミングがオブジェクトコードを生成する前では，プロセッサのアーキテクチャやコンパイラを変更する必要がある．一方，オブジェクトコードを生成した後では，プロセッサに命令を展開するデコーダを付加するがアーキテクチャやコンパイラの変更部分は小規模になる．

ターゲットは RISC プロセッサを使用する組み込みシステムの命令 ROM とする．ROM は製品として出荷したあとは書き換えられることはない．

圧縮率は以下で定義する．

$$\text{圧縮率}\,(\%) = \frac{\text{圧縮後のサイズ}}{\text{圧縮前のサイズ}} \times 100 \qquad (1)$$

本稿の構成は以下の通りである．2 章で関連研究を紹介し，命令 ROM 圧縮にはどのような条件が重要となるかをまとめる．3 章では，Byte Pair 符号化のアルゴリズムを紹介し，組み込みシステムに Byte Pair 符号化を適用する際の変更点や動作について説明する．4 章では Byte Pair 符号化をアプリケーションのオブジェクトコードに適用した実験とその考察を行う．5 章では今後の課題を述べる．

2. 命令 ROM 圧縮

2.1 関連研究

データ圧縮手法の分類には，統計的圧縮法と辞書式圧縮法がある．

統計的圧縮法とは，ビット・パタンの出現頻度をもとに，より出現頻度が高いパタンにより少ないビット・パタンに置換することで圧縮を行う．手法 [1] はオブジェクトコードを圧縮する際に，Huffman 符号化を用いる．キャッシュラインサイズごとにオブジェクトコードを圧縮し，プロセッサはキャッシュに展開された命令を実行する．統計的圧縮法は圧縮後のサイズが 1bit 単位で変化するため，メモリにアラインメントを揃えてデータを配置したり，1 命令をフェッチしたりする操作が複雑になる．圧縮され小さくなったオブジェクトコードと圧縮前の命令のアドレスを対応させるため，LAT（Line Address Table）を用いる．

辞書式圧縮法とは，複数のビット・パタンからなるパタン列を 1 つのパタンに置換することで圧縮を行う．手法 [2] では，1 命令をサブルーチンに置き換えるソフトウェアだけによる手法と，ハードウェアを変更して 1 命令を命令列へのポインタに置き換える手法の 2 つを示している．手法 [3] では手法 [2] に加え，1 命令ごとにタグを付け 1 命令を可変長にしている．手法 [4] では，使用する命令の種類 m に対し $\log(m)$ の辞書を作り置き換える．すべての 1 命令のサイズを一定値 $\log(m)$ に圧縮し，アドレスは圧縮の前後で変化しない．手法 [5] では，命令のオペランド部分を辞書に登録することですべての 1 命令を一定のサイズに圧縮する．

命令セットを変更してオブジェクトコードのサイズを圧縮した手法もある．Thumb [6] と MIPS16 [7] はそれぞれ ARM と MIPS-III アーキテクチャのサブセットから成る．32-bit の命令セットから 16-bit の命令セットに変更することでオブジェクトコードの効率を上げている．

2.2 ROM 圧縮への要求

ROM 圧縮手法に対して，以下の要求があげられる．これらにはトレードオフの関係にある要求がある．

- チップ全体での面積を削減する
- チップ全体での消費電力を削減する
- 実行パフォーマンスを低下させない
- アーキテクチャへの影響が少ない

面積について，命令へ展開するためのハードウェアを付加する場合は，チップの面積は削減された ROM と付加された部分とのトータルの面積で評価する必要がある．ROM 上の命令の出現頻度と実行される命令の頻度は違う．消費電力について，実行される命令に対して展開処理を含めて消費電力を計る必要がある．

上記の要求に対し以下の考えるべきポイントがあげられる．

- 圧縮する処理はコンパイルの先か後か
- 圧縮アルゴリズムは何か
- 展開された命令はどこに格納されるか
- アプリケーションへの圧縮の影響はないか
- 分岐命令後にすぐに命令を実行できるか
- アプリケーションによって圧縮率が大きく変化するかどうか

3. 命令 ROM 圧縮のための Byte Pair 符号化

2.1 章の各手法から，統計的圧縮法はバイトまたはワードのアラインメントを維持することが難しい特徴があげられる．また，辞書式圧縮法は命令列のエントリを増やすことが難しい特徴があげられる．以下に示す Byte Pair 符号化 [8] はこれらの問題への 1 つの解決手段となる．尚これ以降で説明をする Byte Pair 符号化を利用した手法は，手法 [3] において，タグのサイズを 1，置き換える文字列を 2 にした場合にあてはまる．

3.1 Byte Pair 符号化

Byte Pair 符号化は以下の操作を繰り返すことで圧縮をする（図 1）．但しテキストは，文字を構成要素とする集合とする．

『最も出現頻度が高い文字の対をテキストから探し，その文字の対をテキスト中で使用していない文字に置き換える.』

Given: テキスト T, where
T 中に最も多く出現する 2 文字の文字列 b_i,
$[b_i]$ はその出現回数,
T 中に現れない文字 u_j,
j はその個数 .

Algorithm Byte Pair 符号化
  while ($[b_i] > 3$)
    if ($j > 0$)
      T 中の b_i を u_j に置換
      辞書に (u_j,b_i) を登録
      $j = j - 1$
    else
      exit while
    end if
  end while
  Output 圧縮したテキスト，辞書
end Algorithm

図 1　Byte Pair 符号化のアルゴリズム

圧縮の繰り返し処理は，テキスト中で使用していない文字の数を上限とする．もしテキスト中ですべての種類の文字が出現した場合は圧縮できない．辞書に置き換えた文字をエントリとして，置き換えた文字と置き換えられた文字列を対応づけた情報を格納する．具体例（図 2）を使い圧縮処理を説明する．与えられたテキスト abcabdcab 中で ab が出現回数 3 で最もよく現れている．この ab を文字 X で置き換えると図 2 中段の文字列 XcXdcX を得る．その際，X と ab を対応づけた情報を辞書に登録する．同様に，cX を Y に置き換え，対応づけた情報を辞書に登録する．このように 9 文字あったテキストは 4 文字と 2 つのエントリの辞書に圧縮された．

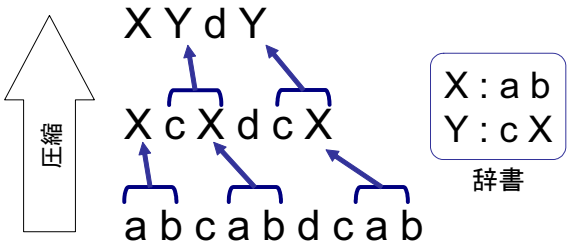

図 2　Byte Pair 符号化の圧縮例

Byte Pair 符号化は辞書式圧縮法に分類される．圧縮の際に固定長（2 文字）を固定長（1 文字）に変換するため，他の辞書式圧縮法と比べより短い文字列を辞書に登録できる．しかし，短い文字列への置換は，展開の差に繰り返し処理をより多く必要とする．Byte Pair 符号化は圧縮には時間がかかるが，展開はシンプルな処理になる性質がある．

圧縮されたテキストを展開する処理は，置き換えた文字を辞書を使い対応する文字列に置き換えることで行われる．展開処理はテキスト上の前後に依存せず，辞書があればどこからでも展開できる性質を持つ．具体例（図 3）を使い説明をする．3 文字分の情報を持つ Y を展開するには，Y を辞書により cX に展開する．同様に，X を ab に展開し，結果として Y は cab という 3 つの文字列に展開する．展開の際に Y は隣り合う文字 X と d に影響されず，辞書の情報のみを使っている．

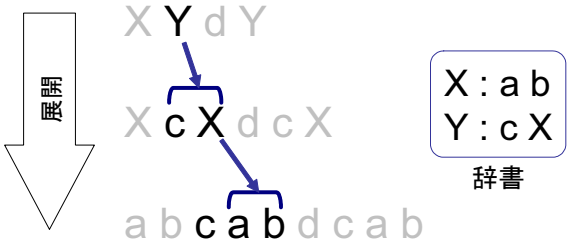

図 3　Byte Pair 符号化の部分的な展開例

3.2 圧縮した命令の展開

Byte Pair 符号化で圧縮した命令を展開するには以下の操作が必要になる．展開処理をハードウェア（デコーダ）により行い，プロセッサ・コアと命令 ROM の間で処理をする（図 4）．

（1） フェッチする命令のアドレスを ROM 上のアドレスに変換する

（2） ROM 上の文字が圧縮されているかを判定する

（3） 文字が圧縮されている場合，辞書を参照し展開する

（4） 展開された文字を結合し，1 命令をプロセッサ・コアへ転送する

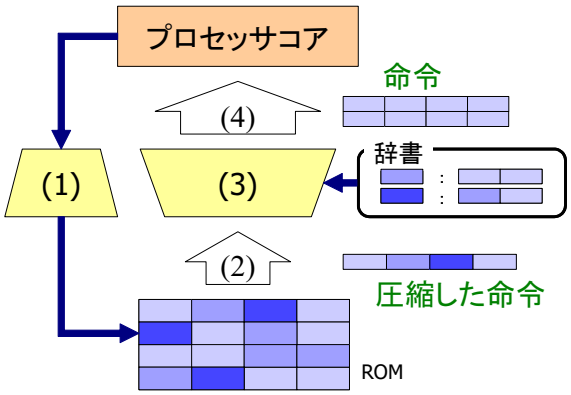

図 4　命令展開フロー

（1）～（4）の 4 つの処理は分岐後の場合はすべてを実行する必要がある．しかし，基本ブロック内では，連続した文字を処理するため（1）の処理は省ける．また，(2）～（4）の処理はパイプライン処理のようにオーバラップして同時実行することができる．

（1）において Byte Pair 符号化の特徴より圧縮されたテキストは任意の場所から展開できるため，分岐命令のアドレスが ROM のアドレスに変換できれば命令を実行できる．一定数の命令をブロックに分け，ブロックごとにアドレスの対応テーブルを作成する．

（2）において文字を読み込む際に文字と辞書のエントリを比較し，文字が圧縮されているかを判定しては処理に時間がかかる．そこで文字が圧縮されているかを判定するために，文字に 1-bit のタグを付加する．タグを付加することは，ハードウェアをシンプルにし，更に Byte Pair 符号化での圧縮を効率的にする．3.1 節で説明したように Byte Pair 符号化はテキスト中に出現しない文字を利用して圧縮を行う．ここで文字にタグを付加することにより，辞書のエントリとして使用する文字数を増やすことができる．

（4）においてデコーダは，一定のサイズごとに展開後の文字をまとめ，1 命令としての文字列をプロセッサ・コアに転送する（図 5）．

3.3　Byte Pair 符号化の効果

2.2 節における要求について考察する．

面積について，トータルの削減効果が期待できる．

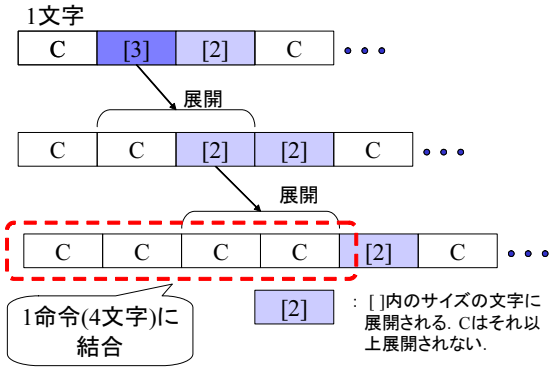

図 5　1 命令を展開する例

なぜならば展開処理はシンプルで繰り返しが多いため，デコーダ (辞書を含む) の面積は圧縮するオブジェクトコードが増加しても一定になることによる．

実行パフォーマンスについて，実行パフォーマンスは低下する．なぜならば命令をフェッチする際に命令を繰り返しにより展開するため，命令を定期的にフェッチできなくなることによる．しかし，繰り返しの回数を制限したり，繰り返しが連続してできるようにすれば，実行パフォーマンスの低下を一定以下に押さえることができる．

アーキテクチャへの変更について，プロセッサ・コアに変更を加える必要がある．具体的には，ROM へアクセスする際にアドレスを変換する必要があり，また命令を実行するまえに展開する必要がある，展開に要する時間を考慮しタイミングを調整する必要がある．

4.　実験・考察

SH4 プロセッサ [10] で動作するアプリケーションの実行ファイルに Byte Pair 符号化を適用する．

4.1　実験の概要

SH4 プロセッサで動作する Linux のアプリケーションの実行ファイルに Byte Pair 符号化を適用した．これらのオブジェクトコードは，SH の命令セットアーキテクチャの情報を含むため，組み込みシステムに使われるオブジェクトコードの代わりに実験を行った．

SH4 のアーキテクチャは以下の特徴を持つ．

- 16-bit 固定長の RISC タイプ命令セットを持つ
- 32-bit 内部データパスを持つ
- 5 段パイプラインで命令を実行する
- MMU を内蔵している
- PC 相対アドレッシングモードがある

表 1　バイナリのサイズと圧縮率の関係
（単位表示のないものはすべて bit）

バイナリ (バイナリのサイズ)	文字のサイズ	圧縮したバイナリ	辞書	アドレス変換テーブル	圧縮率 (%)
openssl(9223952)	4	8355945	170	279240	93.6
	8	6725295	4626	279240	76.0
	16	4994413	667216	279240	64.4
	32	6993426	447018	279240	83.7
rpm(9176544)	4	7820035	170	277806	88.2
	8	6422733	4626	277806	73.1
	32	6835191	385968	277806	81.7

- メモリ領域に命令とデータが混在する

式 1 から ROM の圧縮率を評価する際，圧縮したオブジェクトコードとともに展開時に使用するアドレス変換の情報と文字の置き換えの情報のサイズも考慮する．式（1）において，圧縮後のサイズは以下になる．

圧縮後のサイズ
= 圧縮したオブジェクトコード
+辞書
+アドレス変換テーブル　(2)

本実験でにおいて辞書のサイズは置き換える文字と置き換えた文字列の bit 数に，またアドレス変換テーブルは命令 32-byte ごとに 24-bit のアドレスの対応情報を持つとして近似する．

4.2　実験結果と考察

圧縮率は文字のサイズが 16-bit のときに最高となった（表 1）．SH の命令長である 16-bit で圧縮率が高いのは，バイナリでは同じ命令，同じ命令列が頻繁に使われていると考えられる．

圧縮率はアプリケーションによりばらつきが見られる．圧縮の効果はアプリケーションにより違うが，最も効果がある文字のサイズは同じになった．

4.3　インプリメントへの考察

4.2 の結果より Byte Pair 符号化は文字のサイズが命令長と等しいときに高い圧縮率になる．メモリのアラインメントを保った状態にタグを付加し，アラインメントを変更することは避けた方がよい．文字とタグは別々にメモリ上に配置するか，またはタグは別のメモリや LUT に配置するとよい．

辞書に登録する際，展開処理が連続してできるように辞書のはじめは圧縮していない文字で始まるようにすることができる．

5.　おわりに

本稿では，Byte Pair 符号化を組み込みシステムのオブジェクトコードに適用し，ROM 面積削減への影響を調べた．

本手法をキャッシュやパイプライン，分岐命令といったプロセッサの様々な観点から動作や効果を調べる必要がある．本手法は SH 以外の RISC タイプアーキテクチャに適用できる．

文　献

[1] A.Wolfe and A.Chanin. “Executing Compressed Programs on An Embedded RISC Archtecture” Micro-25, 1992

[2] S.Y.Liao, S.Devadas and K.Keutzer. “Code Density Optimiziation for Enbedded DSP Processors Using Data Compression Techniques” ARVLSI, 1995

[3] C.Lefurgy, P.Bird, I.Chen and T.Mudge. “Improving Code Density Using Compression Techniques” Micro-30, 1997

[4] Y.Yoshida, B.Song, H.Okuhata,T.Onoye, and I.Shirakawa. “An Object Code Compression Approach to Embedded Processors” ISLPED97

[5] T.Okuma, H.Tomiyama, A.Inoue, E.Fajar and H.Yasuura. “Instruction Encoding Techniques for Area Minimization of Instruction ROM” ISSS, 1998

[6] Advanced RISC Machines Ltd. “An Intruduction to Themb” 1995

[7] K.Kissel. “MIPS16: High-density MIPS for the Embedded Market” Silicon Graphics MIPS Group, 1997

[8] P.Gage “A new algorithm for data compression” The C Users Journal,12(2),1994

[9] Charles Lefuragy. “Efficient Execution of Compressed Programs” http://www.eecs.umich.edu/~tnm/compress/compress-pubs.html The University of Michigan, 2000

[10] HITACHI “日立 SuperH RISC engine SH-4 プログラミングマニュアル” http://www.super-h.com/