

(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2022年3月31日(31.03.2022)



(10) 国際公開番号

WO 2022/064683 A1

(51) 国際特許分類:
G06F 8/52 (2018.01)

(21) 国際出願番号: PCT/JP2020/036585

(22) 国際出願日: 2020年9月28日(28.09.2020)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(71) 出願人:三菱電機株式会社(MITSUBISHI ELECTRIC CORPORATION) [JP/JP]; 〒1008310 東京都千代田区丸の内二丁目7番3号 Tokyo (JP).
国立大学法人 東京大学(THE UNIVERSITY OF TOKYO) [JP/JP]; 〒1138654 東京都文京区本郷七丁目3番1号 Tokyo (JP).

(72) 発明者:奥田 勝己(OKUDA Katsumi); 〒1008310 東京都千代田区丸の内二丁目7番3号 三菱電機株式会社内 Tokyo (JP). 千葉 滋(CHIBA

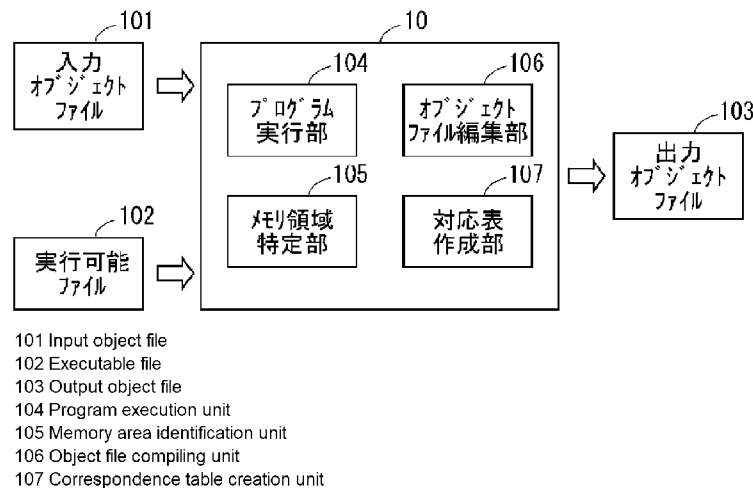
Shigeru); 〒1138654 東京都文京区本郷七丁目3番1号 国立大学法人東京大学内 Tokyo (JP).

(74) 代理人:吉竹 英俊, 外(YOSHITAKE Hidetoshi et al.); 〒5400001 大阪府大阪市中央区域見1丁目4番70号住友生命OBPプラザビル10階 Osaka (JP).

(81) 指定国(表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(54) Title: PROGRAM SIZE REDUCTION DEVICE, PROGRAM SIZE REDUCTION PROGRAM, AND PROGRAM SIZE REDUCTION METHOD

(54) 発明の名称: プログラムサイズ低減装置、プログラムサイズ低減プログラム、および、プログラムサイズ低減方法



(57) Abstract: The present invention reduces the size of a program while enabling reuse of the program. This program size reduction device comprises: a memory area identification unit that identifies, as a memory area, a first address of a first binary code to be accessed by execution of an executable file linked to an object file; a correspondence table creation unit that creates a correspondence table between a first address and an offset from the head of a section in a second binary code in the object file; and an object file compiling unit that deletes a second binary code for an offset that does not correspond to the first address on the basis of the correspondence table.

(84) 指定国(表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, RU, TJ, TM), ヨーロッパ (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

添付公開書類:

一 国際調査報告 (条約第21条(3))

(57) 要約: 再利用可能な状態で、プログラムサイズを低減する。プログラムサイズ低減装置は、オブジェクトファイルにリンクされた実行可能ファイルの実行によってアクセスされる第1のバイナリコードの第1のアドレスを、メモリ領域として特定するメモリ領域特定部と、第1のアドレスと、オブジェクトファイルにおける第2のバイナリコードのセクション先頭からのオフセットとの対応表を作成する対応表作成部と、対応表に基づいて、第1のアドレスに対応しないオフセットの第2のバイナリコードを削除するオブジェクトファイル編集部とを備える。

明 細 書

発明の名称：

プログラムサイズ低減装置、プログラムサイズ低減プログラム、および、
プログラムサイズ低減方法

技術分野

[0001] 本願明細書に開示される技術は、プログラムサイズ低減装置、プログラムサイズ低減プログラム、および、プログラムサイズ低減方法に関するものである。

背景技術

[0002] 特定の機能を実現するために機械などに組み込まれる組み込みシステムでは、一般にメモリの低減が重要である。組み込みシステムでは、少ないリソースでソフトウェアを実行する必要がある。よって、動作のために必要となるメモリのサイズが大きくなると、その分だけメモリモジュールの追加または高価なメモリ内蔵マイコンなどが必要となってしまう。このため、製品の製造コストを下げるためには、ソフトウェアの省メモリ化を行うことが必要である。

[0003] また、近年ではIoT (Internet of Things) への対応のため、組み込みシステムの高機能化が進んでいる。組み込みシステムがインターネットに接続される環境では、セキュリティー確保のための暗号化および復号、通信量を削減するための圧縮および展開、さらには、アプリケーション間の通信のためのプロトコル処理などが必要である。

[0004] このような高機能化された組み込みシステムを短期間で開発するためには、既存ライブラリを利用することが有効な手段である。既存ライブラリとしては、たとえば、COTS (Commercial Off the Shelf Software)、または、OSS (Open Source Software) などが考えられる。

[0005] しかしながら、これらの既存ライブラリは多用途向けに汎用化されている

場合が多く、ソフトウェアにおけるプログラムにライブラリをリンクさせた結果として、不要な機能までリンク結果に含まれる場合がある。このため、メモリに対応するプログラムのサイズが必要以上に大きくなり、組み込みシステムとして用いることが困難となる場合がある。

[0006] これらの汎用的な既存ライブラリの多くは、PCまたはサーバー向けのものであり、そのままでは組み込みシステムとして用いることは難しい。このため、既存ライブラリを組み込みシステムに対して用いる場合には、そのソースコードから不要な部分を削除するというチューニングが必要である。

[0007] ただし、人手によるチューニング作業は、単調で、かつ、誤りが生じがちである。また、COTSの場合はソースコードが提供されないため、人手によるチューニングが実質的には困難である。

[0008] ソースコードが存在しない場合にプログラムのサイズを削除する他の方法としては、たとえば、特許文献1に記載される方法がある。特許文献1に記載される方法では、プログラムを実行し、当該実行時に実行されなかった部分のバイナリコードを削除する。

先行技術文献

特許文献

[0009] 特許文献1：特開2000-035891号公報

発明の概要

発明が解決しようとする課題

[0010] しかしながら、特許文献1に記載される方法では、実行されたプログラムのバイナリコードを直接書き換えることとなるため、書き換えられたプログラムを他の用途で再利用することができないという問題があった。

[0011] 本願明細書に開示される技術は、以上に記載されたような問題を鑑みてなされたものであり、ソースコードが得られないライブラリである場合を含め、再利用可能な状態で、プログラムサイズを低減するための技術である。

課題を解決するための手段

- [0012] 本願明細書に開示される技術の第1の態様であるプログラムサイズ低減装置は、入力オブジェクトファイルにリンクされた実行可能ファイルの実行によってアクセスされる、前記実行可能ファイルにおける第1のバイナリコードの第1のアドレスを、メモリ領域として特定するためのメモリ領域特定部と、前記メモリ領域における前記第1のアドレスと、前記入力オブジェクトファイルにおける第2のバイナリコードのセクション先頭からの第1のオフセットとの対応表を作成するための対応表作成部と、前記対応表に基づいて、前記メモリ領域における前記第1のアドレスに対応しない前記第1のオフセットの前記第2のバイナリコードを削除する編集を行うためのオブジェクトファイル編集部とを備える。
- [0013] 本願明細書に開示される技術の第2の態様であるプログラムサイズ低減プログラムは、コンピュータにインストールされて実行されることによって、前記コンピュータに、入力オブジェクトファイルにリンクされた実行可能ファイルの実行によってアクセスされる、前記実行可能ファイルにおける第1のバイナリコードの第1のアドレスを、メモリ領域として特定させ、前記コンピュータに、前記メモリ領域における前記第1のアドレスと、前記入力オブジェクトファイルにおける第2のバイナリコードのセクション先頭からの第1のオフセットとの対応表に基づいて、前記メモリ領域における前記第1のアドレスに対応しない前記第1のオフセットの前記第2のバイナリコードを削除する編集を行わせる。
- [0014] 本願明細書に開示される技術の第3の態様であるプログラムサイズ低減方法は、入力オブジェクトファイルにリンクされた実行可能ファイルの実行によってアクセスされる、前記実行可能ファイルにおける第1のバイナリコードの第1のアドレスを、メモリ領域として特定し、前記メモリ領域における前記第1のアドレスと、前記入力オブジェクトファイルにおける第2のバイナリコードのセクション先頭からの第1のオフセットとの対応表に基づいて、前記メモリ領域における前記第1のアドレスに対応しない前記第1のオフセットの前記第2のバイナリコードを削除する編集を行う。

発明の効果

[0015] 本願明細書に開示される技術の少なくとも第1、2、3の態様によれば、実行可能ファイルを直接書き換えるのではなく、ソフトウェアを構築するときに際に用いられたオブジェクトファイルを書き換えることによってプログラムサイズを低減させることができる。そのため、書き換えられた出力オブジェクトファイルを他のプログラムでも再利用することができる。

[0016] また、本願明細書に開示される技術に関連する目的と、特徴と、局面と、利点とは、以下に示される詳細な説明と添付図面とによって、さらに明白となる。

図面の簡単な説明

[0017] [図1]実施の形態に関する、プログラムサイズ低減装置の構成の例を概念的に示す図である。

[図2]コンパイルした結果入力オブジェクトファイルを得ることができるC言語プログラムのソースコードの例を示す図である。

[図3]コンパイルおよびリンクして実行可能ファイルを得ることができるC言語プログラムのソースコードの例を示す図である。

[図4]ARMアーキテクチャ向けの実行可能ファイルをGNU `objdump` でディスアセンブルした結果を編集したものを示す図である。

[図5]プログラム実行部およびメモリ領域特定部の処理フローの例を示すフローチャートである。

[図6]オブジェクトファイル編集部および対応表作成部の処理フローの例を示すフローチャートである。

[図7]図2に示されたC言語プログラムをコンパイルして得られる入力オブジェクトファイルの、セクションヘッダの例を示す図である。

[図8]図2に示された入力オブジェクトファイルをディスアセンブルした結果を示す図である。

[図9]図8に示されたディスアセンブル結果における、`text` セクションから、不要部分が削除された結果を示す図である。

[図10]例として、図6のステップST603での、textセクション書き換えの結果が図9に示されるようになった場合の、textセクションのサイズを反映するセクションヘッダを示す図である。

[図11]図8の、textセクションを図9のように書き換えた場合の、書き換え前のリロケーションテーブルの例を示す図である。

[図12]図11に示されたりロケーションテーブルが適切に書き換えられた結果を示す図である。

[図13]図6のステップST602の詳細な処理フローを示すフローチャートである。

[図14]図6のステップST603の詳細な処理フローを示すフローチャートである。

[図15]上記の書き換え前のシンボルテーブルの例を示す図である。

[図16]上記の書き換え後のシンボルテーブルの例を示す図である。

[図17]図14のステップST1411の分岐命令書き換え処理の詳細な処理フローを示すフローチャートである。

[図18]図6のステップST606のリロケーションテーブル書き換え処理の詳細な処理フローを示すフローチャートである。

[図19]図1に例が示されるプログラムサイズ低減装置を実際に運用する場合のハードウェア構成を概略的に例示する図である。

[図20]図1に例が示されるプログラムサイズ低減装置を実際に運用する場合のハードウェア構成を概略的に例示する図である。

[図21]本実施の形態に関するプログラムサイズ低減装置の構成（機能部）の例を概念的に示す図である。

発明を実施するための形態

[0018] 以下、添付される図面を参照しながら実施の形態について説明する。以下の実施の形態では、技術の説明のために詳細な特徴なども示されるが、それらは例示であり、実施の形態が実施可能となるためにそれらすべてが必ずしも必須の特徴ではない。

[0019] なお、図面は概略的に示されるものであり、説明の便宜のため、適宜、構成の省略、または、構成の簡略化が図面においてなされるものである。また、異なる図面にそれぞれ示される構成などの大きさおよび位置の相互関係は、必ずしも正確に記載されるものではなく、適宜変更され得るものである。また、断面図ではない平面図などの図面においても、実施の形態の内容を理解することを容易にするために、ハッチングが付される場合がある。

[0020] また、以下に示される説明では、同様の構成要素には同じ符号を付して図示し、それらの名称と機能とについても同様のものとする。したがって、それらについての詳細な説明を、重複を避けるために省略する場合がある。

[0021] また、以下に記載される説明において、ある構成要素を「備える」、「含む」または「有する」などと記載される場合、特に断らない限りは、他の構成要素の存在を除外する排他的な表現ではない。

[0022] また、以下に記載される説明において、「第1の」または「第2の」などの序数が用いられる場合があっても、これらの用語は、実施の形態の内容を理解することを容易にするために便宜上用いられるものであり、これらの序数によって生じ得る順序などに限定されるものではない。

[0023] <プログラムサイズ低減装置の概念的な構成について>

図21は、本実施の形態に関するプログラムサイズ低減装置の構成（機能部）の例を概念的に示す図である。

[0024] 図21に例示されるようにプログラムサイズ低減装置は、メモリ領域特定部3001と、対応表作成部3002と、オブジェクトファイル編集部3003とを備える。

[0025] メモリ領域特定部3001は、入力オブジェクトファイルにリンクされた実行可能ファイルの実行によってアクセスされる、実行可能ファイルにおける第1のバイナリコードの第1のアドレスを、メモリ領域として特定する。対応表作成部3002は、メモリ領域における第1のアドレスと、入力オブジェクトファイルにおける第2のバイナリコードのセクション先頭からの第1のオフセットとの対応表を作成する。オブジェクトファイル編集部300

3は、対応表に基づいて、メモリ領域における第1のアドレスに対応しない第1のオフセットの第2のバイナリコードを削除する編集を行う。

[0026] 以下の実施の形態において図面などを用いて示されるそれぞれの構成は、上記の図21に示された構成の例をさらに具体的に示すものである。

[0027] <実施の形態>

以下、本実施の形態に関するプログラムサイズ低減装置、プログラムサイズ低減プログラム、および、プログラムサイズ低減方法について説明する。

[0028] <プログラムサイズ低減装置の構成について>

図1は、本実施の形態に関するプログラムサイズ低減装置10の構成の例を概念的に示す図である。

[0029] 図1に例が示されるプログラムサイズ低減装置10は、軽量化対象である入力オブジェクトファイル101と、入力オブジェクトファイル101とリンクする実行可能ファイル102とを入力とし、実行可能ファイル102の実行に不要な部分が論理的に削除された軽量化済みの出力オブジェクトファイル103を出力する。

[0030] 図1に例が示されるプログラムサイズ低減装置10は、プログラム実行部104と、メモリ領域特定部105と、オブジェクトファイル編集部106と、対応表作成部107とを備える。

[0031] プログラム実行部104は、実行可能ファイル102をロードし、かつ、実行可能ファイル102を実行する。メモリ領域特定部105は、実行可能ファイル102の実行時にアクセスされるバイナリコードのアドレスを記録し、当該アドレス群を、実行可能ファイル102の実行時に必要となるメモリ領域として特定する。

[0032] オブジェクトファイル編集部106は、メモリ領域特定部105によって特定されたメモリ領域を用いて、入力オブジェクトファイル101から実行可能ファイル102を実行するために不要な部分を削除するとともに、入力オブジェクトファイル101のセクションヘッダテーブル、シンボルテーブルおよびリロケーションテーブルを書き換える。そして、オブジェクトファ

イル編集部 106 は、メモリ領域特定部 105 によって特定されたメモリ領域に対応しない入力オブジェクトファイル 101 中のバイナリコードおよびデータを削除することで、プログラムサイズを削減する。

[0033] このとき、入力オブジェクトファイル 101 を編集するためには、メモリ領域に対応する入力オブジェクトファイル 101 中の領域が既知である必要がある。そこで、オブジェクトファイル編集部 106 は、対応表作成部 107 によって作成された、メモリ領域におけるアドレスと、入力オブジェクトファイル 101 中のセクション先頭からのオフセットとの対応表を利用して、入力オブジェクトファイル 101 を編集する。

[0034] 図 2 は、コンパイルした結果入力オブジェクトファイル 101 を得ることができる C 言語プログラムのソースコードの例を示す図である。たとえば、図 2 のソースコードを ARM アーキテクチャ向けにコンパイルすると入力オブジェクトファイル 101 の 1 例を得られる。図 2 のソースコードから得られるオブジェクトファイルは、ライブラリとして利用することが想定される。なお、図 2 のソースコードは説明のために便宜上図示されたものであり、本実施の形態の実施において用いられなくてもよい。また、図 3 は、コンパイルおよびリンクして実行可能ファイル 102 を得ることができる C 言語プログラムのソースコードの例を示す図である。図 3 のソースコードをコンパイルして得られるオブジェクトファイルと、図 2 のソースコードをコンパイルして得られるオブジェクトファイルとをリンクすると、実行可能ファイル 102 の 1 例が得られる。

[0035] 以降の説明では、図 2 に例として示されたソースコードをコンパイルして得られた入力オブジェクトファイル 101 から、図 3 に例が示された実行可能ファイル 102 のバイナリコードによるプログラムを実行するために不要な部分を削除する場合を考える。

[0036] 図 2 に示される C 言語プログラムの関数 `calc` は、引数 `operation` で与えられる演算種類ごとに引数 `x` と `y` とを用いて四則演算のいずれかを行う。図 3 に示される C 言語プログラムの関数 `calc__average`

は、図2に示される関数 `calc` を用いて2数の平均を計算する。

[0037] 関数 `calc` での計算および関数 `calc__average` での計算では、加算および除算のみを用いる。このため、図3に示されたソースコードによるプログラムを実行するためには、図2の関数 `calc` における減算および乗算のソースコードに対応するバイナリコードは不要である。

[0038] プログラム実行部104の実現方法の1つは、命令セットシミュレータである。命令セットシミュレータは、実行可能ファイル102をロードし、さらに、プログラムのエントリーポイントからプログラムの実行を開始する。命令セットシミュレータの命令実行方式にはインタプリタ方式またはバイナリ変換方式などが存在するが、本実施の形態ではいずれの方式が採用されてもよい。

[0039] 図2では、命令セットシミュレータであるプログラム実行部104の動作手順が示される。

[0040] 図2では、プログラム実行部104は、実行可能ファイル102を実行し、必要アドレス情報として必要アドレス集合と命令先頭アドレス集合とを出力する。メモリ領域特定部105は、出力された必要アドレス情報に基づいて、メモリ領域を特定する。

[0041] 必要アドレス集合は、削除対象外のバイナリコードまたはデータのアドレスをバイトごとに保持する集合である。また、命令先頭アドレス集合は、実行された命令の先頭のアドレスを保持する集合である。命令先頭アドレス集合は、入力オブジェクトファイル101の編集処理において命令をデコードする際に用いられる。

[0042] 例として、図3に示されるC言語プログラムをコンパイルして得られる実行可能ファイル102を実行するために必要となる必要アドレス情報を考える。

[0043] 図4は、ARMアーキテクチャ向けの実行可能ファイル102を `GNU objdump` でディスアセンブルした結果を編集したものを示す図である。図4におけるそれぞれの行は、図2に示された入力オブジェクトファイル

101のそれぞれの行に対応している。

[0044] 図4において、実行可能ファイル102を実行するために必要となる図2中のソースコードのアドレスと対応する行の先頭には、xが示されている。

[0045] この場合、必要アドレス集合は、以下のように表される。

[0046] {整数 $n \mid 0x82b4 \leq n < 0x82e0$ または $0x82f8 \leq n < 0x8300$ }

また、命令先頭アドレス集合は、以下のように表される。

[0047] { $0x82b4, 0x82b8, 0x82bc, 0x82c0, 0x82c4, 0x82c8, 0x82cc, 0x82d0, 0x82d4, 0x82d8, 0x82dc, 0x82f8, 0x82fc$ }

メモリ領域特定部105は、出力されたこれらの必要アドレス情報に基づいて、メモリ領域を特定する。

[0048] 図5は、プログラム実行部104およびメモリ領域特定部105の処理フローの例を示すフローチャートである。

[0049] 図5に例が示されるように、ステップST501では、プログラム実行部104がループ処理を開始する。当該ループ処理は、ステップST509においてプログラムの終了条件が満たされるまで1命令ずつの実行を繰り返す処理である。

[0050] 次に、ステップST502では、プログラム実行部104が、プログラムカウンターが指す実行可能ファイル102のアドレスから命令をフェッチする。

[0051] 次に、ステップST503では、メモリ領域特定部105が、ステップST502でフェッチされた命令に対応するアドレスを、バイトごとに必要アドレス集合に加える。たとえば、 0×1000 番地から4バイトの命令がフェッチされた場合、 0×1000 番地、 0×1001 番地、 0×1002 番地および 0×1003 番地を必要アドレス集合に加える。

[0052] 次に、ステップST504では、メモリ領域特定部105が、ステップST502でプログラムカウンターが指す先頭のアドレスを、命令先頭アドレ

ス集合に追加する。たとえば、命令をフェッチされたアドレスが 0×1000 番地である場合、命令先頭アドレス集合に 0×1000 番地を加える。

[0053] 次に、ステップST505では、プログラム実行部104が、フェッチされた命令をデコードする。当該デコードでは、命令のオペコードに基づいて命令の種類を特定し、さらに、命令オペランドを対応する命令から取得する。

[0054] 次に、ステップST506では、プログラム実行部104が、ステップST505で取得された命令を実行する。

[0055] 次に、ステップST507では、プログラム実行部104が、ステップST506で実行された命令がコード領域へのメモリ読み出しを伴う命令であるか否かを判定する。

[0056] 命令がコード領域へのメモリ読み出しを伴うか否かは、実行可能ファイル102に含まれるセクションテーブルから判定することができる。たとえば、実行可能ファイル102における読み出し先（アクセス先）のアドレスXが、textセクションに含まれている場合、アドレスXはコード領域中のデータを指す。なお、ここで読み出されるデータは、命令ではなく、コード領域に埋め込まれた定数または分岐先アドレスである。

[0057] そして、ステップST506で実行された命令がコード領域へのメモリ読み出しを伴う命令である場合、すなわち、図5に例が示されるステップST507から分岐する「YES」に対応する場合には、図5に例が示されるステップST508へ進む。

[0058] 一方で、ステップST506で実行された命令がコード領域へのメモリ読み出しを伴う命令でない場合、すなわち、図5に例が示されるステップST507から分岐する「NO」に対応する場合には、図5に例が示されるステップST509へ進む。

[0059] ステップST508では、メモリ領域特定部105が、必要アドレス集合に要素を追加する。具体的には、ステップST502でフェッチされた命令に対応する先頭のアドレスをYとし、実行可能ファイル102における読み

出し先（アクセス先）のアドレスを X （ $X > Y$ ）とし、アクセスサイズを n とする場合、 Y 、 $Y + 1$ 、 $\dots$ 、 $X + n - 1$ を必要アドレス集合に加える。

[0060] たとえば、ステップST502でフェッチされた命令に対応する先頭のアドレスが図4に示された $0x82b8$ であり、読み出されたコード領域のデータが $0x82cc$ の4バイトデータである場合、 $X = 0x82cc$ 、 $Y = 0x82b8$ 、 $n = 4$ となり、必要アドレス集合には、 $0x82b8$ （ $= Y$ ）、 $0x82b9$ 、 $\dots$ 、 $0x82ce$ 、 $0x82cf$ （ $X + 3$ ）が追加される。

[0061] なお、ステップST507およびステップST508は、入力オブジェクトファイル101のコード領域中に命令以外のデータが埋め込まれた場合を扱うために必要となる付加的なステップである。そのため、コード領域に命令以外のデータが埋め込まれていない場合には、これらのステップは不要である。

[0062] ステップST509では、プログラム実行部104が、終了条件が満たされたか否かを判定する。ここで、終了条件は、ユーザーの対話的操作による終了、または、プログラムカウンタが特定のアドレスに到達した場合などであり、一般的な命令セットシミュレータの終了条件と同様である。

[0063] そして、終了条件が満たされた場合、すなわち、図5に例が示されるステップST509から分岐する「YES」に対応する場合には、動作を終了する。一方で、終了条件が満たされていない場合、すなわち、図5に例が示されるステップST509から分岐する「NO」に対応する場合には、図5に例が示されるステップST501に戻る。

[0064] 次に、オブジェクトファイル編集部106の詳細について説明する。オブジェクトファイル編集部106は、実行可能ファイル102と、入力オブジェクトファイル101と、必要アドレス情報とを入力とし、入力オブジェクトファイル101を書き換えて作成される不要箇所が削除済みである出力オブジェクトファイル103を出力する。

[0065] 図6は、オブジェクトファイル編集部106および対応表作成部107の

処理フローの例を示すフローチャートである。

- [0066] 図6に例が示されるように、ステップST601では、オブジェクトファイル編集部106がループ処理を開始する。当該ループ処理は、入力オブジェクトファイル101のセクションヘッダのすべてのエントリーに対して処理が行われるまで実行を繰り返す処理である。
- [0067] 図7は、図2に示されたC言語プログラムをコンパイルして得られる入力オブジェクトファイル101の、セクションヘッダの例を示す図である。図7は、GNU binutilsのreadelfコマンドで入力オブジェクトファイル101のセクションヘッダを出力させた結果を示している。
- [0068] 図7に示されるように、セクションヘッダのエントリーとして、.textまたは.dataなどが存在する。そして、それぞれのエントリーは、セクションごとのファイルオフセットなどを保持する。
- [0069] 図6のステップST602、図6のステップST603、図6のステップST604、図6のステップST605および図6のステップST606は、.textまたは.dataなどのセクションごとに実行される。
- [0070] 図6のステップST602では、対応表作成部107が、アドレスとオフセットとの対応表を作成する。当該対応表は、セクション中のバイトデータのオフセットごとに、実行可能ファイル102をロードした後のアドレスxとオフセットyとの対を保持する。
- [0071] ここで、図4に示された、実行可能ファイル102がディスアセンブルされた結果に対応する、アドレスとオフセットとの対応表を考える。
- [0072] 図8は、図2に示された入力オブジェクトファイル101をディスアセンブルした結果を示す図である。図8では、入力オブジェクトファイル101で削除することができない箇所には、行先頭のxが付されている。アドレスとオフセットとの対応表が保持するxとyとの対の一例として、図4に示されたアドレス0x82b4と、図8に示されたオフセット0x14とが挙げられる。
- [0073] これは、オフセット0x14は関数calcの、.textセクション先頭

からのオフセットであり、入力オブジェクトファイル101にリンクされた実行可能ファイル102では、0x82b4に関数calcの先頭が配置されていることを意味する。なお、図6のステップST602の詳細については、さらに後述する。

[0074] 図6のステップST603では、オブジェクトファイル編集部106が、必要アドレス情報と、アドレスとオフセットとの対応表とを用いて、セクションから不要データを削除してセクションの書き換えを行う。

[0075] 図9は、図8に示されたディスアSEMBル結果における、textセクションから、不要部分が削除された結果を示す図である。

[0076] 図8では、入力オブジェクトファイル101で削除することができない箇所には、行先頭のxが付されている。よって、図9において残っているコードは、図8において先頭にxが付記されている行に対応する。なお、図6のステップST603の詳細については、さらに後述する。

[0077] 図6のステップST604では、オブジェクトファイル編集部106が、セクションヘッダのエントリーを書き換える。ここで、書き換え対象となる部分は、セクションのサイズである。新しいセクションのサイズは、図6のステップST603で不要部分が削除された後のセクションのサイズである。

[0078] 図10は、例として、図6のステップST603での、textセクション書き換えの結果が図9に示されるようになった場合の、textセクションのサイズを反映するセクションヘッダを示す図である。図10において、textセクションのサイズは0x3Cバイトに変更されている。

[0079] 図6のステップST605では、オブジェクトファイル編集部106が、シンボルテーブルの書き換え処理を行う。

[0080] 図6のステップST606では、オブジェクトファイル編集部106が、リロケーションテーブルの書き換え処理を行う。具体的には、図6のステップST603で移動された（書き換えられた）箇所がリロケーション箇所（すなわち、リロケーションテーブルに格納されているエントリーのうちのい

ずれか)である場合に、リロケーション箇所を示すオフセットを新しいオフセットに更新する。また、図6のステップST603で削除された領域に対応するリロケーションエントリーを無効化する。

[0081] 図6のステップST607では、オブジェクトファイル編集部106が、入力オブジェクトファイル101のセクションヘッダのすべてのエントリーに対して処理が行われたか否かを判定する。

[0082] そして、入力オブジェクトファイル101のセクションヘッダのすべてのエントリーに対して処理が行われた場合、すなわち、図6に例が示されるステップST607から分岐する「YES」に対応する場合には、動作を終了する。一方で、処理が行われていないエントリーが残っている場合、すなわち、図6に例が示されるステップST607から分岐する「NO」に対応する場合には、図6に例が示されるステップST601に戻る。

[0083] 図11は、図8の、`text`セクションを図9のように書き換えた場合の、書き換え前のリロケーションテーブルの例を示す図である。また、図12は、図11に示されたリロケーションテーブルが適切に書き換えられた結果を示す図である。

[0084] 図12において、型が`R_ARM_NONE`に書き換えられたエントリーは、削除された領域に対応する。また、型が`R_ARM_NONE`以外であるエントリーは、それぞれオフセットが修正されている(太字参照)。

[0085] たとえば、図11における書き換え前のエントリー1101のオフセットは`0x3c`であるが、図12において対応する書き換え後のエントリー1201では、オフセットは`0x28`に書き換えられる。

[0086] これは、図8の、`text`セクションのオフセット`0x3c`に存在していたリロケーション対象箇所801が、図6のステップST603のセクション書き換えによって、図9の、`text`セクションのオフセット`0x28`に存在するリロケーション対象箇所901に移動したことに対応している。なお、図6のステップST606の詳細については、さらに後述する。

[0087] 図13は、図6のステップST602の詳細な処理フローを示すフローチ

ャートである。図 1 3 に例が示される場合では、対象セクション名を入力とする。

[0088] 図 1 3 に例が示されるように、ステップ S T 1 3 0 1 では、オブジェクトファイル編集部 1 0 6 がループ処理を開始する。当該ループ処理は、入力オブジェクトファイル 1 0 1 のシンボルテーブルのすべてのエントリーに対して処理が行われるまで実行を繰り返す処理である。

[0089] ステップ S T 1 3 0 2 では、オブジェクトファイル編集部 1 0 6 が、シンボルテーブルのエントリーのシンボルが対象セクションのシンボルであるか否かを判定する。

[0090] そして、シンボルテーブルのエントリーのシンボルが対象セクションのシンボルである場合、すなわち、図 1 3 に例が示されるステップ S T 1 3 0 2 から分岐する「Y E S」に対応する場合には、ステップ S T 1 3 0 3 へ進む。一方で、シンボルテーブルのエントリーのシンボルが対象セクションのシンボルでない場合、すなわち、図 1 3 に例が示されるステップ S T 1 3 0 2 から分岐する「N O」に対応する場合には、図 1 3 に例が示されるステップ S T 1 3 0 5 へ進む。

[0091] ステップ S T 1 3 0 3 では、オブジェクトファイル編集部 1 0 6 が、実行可能ファイル 1 0 2 のシンボルテーブルを検索し、現在対象にしているシンボルテーブルのエントリーのシンボル名に対応するアドレス x とサイズ n とを得る。

[0092] ステップ S T 1 3 0 4 では、オブジェクトファイル編集部 1 0 6 が、シンボルテーブルのエントリーが保持するオフセットを y として、アドレス x からアドレス $x + n - 1$ の領域と、オフセット y からオフセット $y + n - 1$ の領域とのバイトデータごとの対応を、アドレスとオフセットとの対応表に追加する。

[0093] ステップ S T 1 3 0 5 では、オブジェクトファイル編集部 1 0 6 が、入力オブジェクトファイル 1 0 1 のシンボルテーブルのすべてのエントリーに対して処理が行われたか否かを判定する。

[0094] そして、入力オブジェクトファイル 101 のシンボルテーブルのすべてのエントリーに対して処理が行われた場合、すなわち、図 13 に例が示されるステップ ST 1305 から分岐する「YES」に対応する場合には、動作を終了する。一方で、処理が行われていないエントリーが残っている場合、すなわち、図 13 に例が示されるステップ ST 1305 から分岐する「NO」に対応する場合には、図 13 に例が示されるステップ ST 1301 に戻る。

[0095] 図 14 は、図 6 のステップ ST 603 の詳細な処理フローを示すフローチャートである。図 14 に例が示される場合では、必要アドレス情報と、アドレスとオフセットとの対応表とを入力とし、対象セクションから不要なコードまたはデータを削除する。なお、不要部分の削除は、必要な部分のみを上から順に上書きコピーすることで行う。

[0096] 図 14 の処理フローでは出力として、セクション書き換え前後のデータの対応を示すオフセットとオフセットとの対応表を出力する。オフセットとオフセットとの対応表のエントリーはオフセット対 (i、j) であり、書き換え前にオフセット i にあったバイトデータが書き換え後にオフセット j に移動したことを保持する。

[0097] まず、図 14 のステップ ST 1401 では、オフセット j を 0 で初期化する。オフセット j は書き込み先のバイトオフセットを示し、適宜更新される。ここで、セクション先頭からオフセットを加えた値が書き込み先を示す。

[0098] 次に、ステップ ST 1402 では、オブジェクトファイル編集部 106 がループ処理を開始する。当該ループ処理は、オフセット i = 0 から i = セクションサイズ - 1 まで i の値を 1 ずつインクリメントしながら処理が行われる繰り返し処理である。ここで、オフセット i は、読み出し元のバイトオフセットを示す。

[0099] ステップ ST 1403 では、オブジェクトファイル編集部 106 が、アドレスとオフセットとの対応表から、オフセットの i に対応するアドレス x を取得する。

[0100] ステップ ST 1404 では、オブジェクトファイル編集部 106 が、アド

レスxが必要アドレス集合に含まれるか否かを判定する。

[0101] そして、アドレスxが必要アドレス集合に含まれる場合、すなわち、図14に例が示されるステップST1404から分岐する「YES」に対応する場合には、オフセットiのデータが必要であると判定し、図14に例が示されるステップST1405へ進む。一方で、アドレスxが必要アドレス集合に含まれない場合、すなわち、図14に例が示されるステップST1404から分岐する「NO」に対応する場合には、オフセットiのデータが必要でないと判定し、図14に例が示されるステップST1408へ進む。

[0102] ステップST1405では、オブジェクトファイル編集部106が、オフセットiのバイトデータをオフセットjの位置に上書きする（移動する）。

[0103] ステップST1406では、オブジェクトファイル編集部106が、オフセットとオフセットとの対応表にオフセット対(i、j)を追加する。

[0104] ステップST1407では、オブジェクトファイル編集部106が、jを1だけインクリメントする。

[0105] ステップST1408では、オブジェクトファイル編集部106が、オフセットiがシンボルの末尾に対応しているか否かを、入力オブジェクトファイル101のシンボルテーブルを検索することによって判定する。なお、シンボルテーブルに $k + s - 1 = i$ を満たすオフセットkとシンボルサイズsとを保持するエントリーが存在する場合、オフセットiはシンボルの末尾に対応している。

[0106] そして、オフセットiがシンボルの末尾に対応している場合、すなわち、図14に例が示されるステップST1408から分岐する「YES」に対応する場合には、ステップST1409へ進む。一方で、オフセットiがシンボルの末尾に対応していない場合、すなわち、図14に例が示されるステップST1408から分岐する「NO」に対応する場合には、図13に例が示されるステップST1410へ進む。

[0107] ステップST1409では、オブジェクトファイル編集部106が、ステップST1408における検索で得られたシンボルテーブルのエントリーの

オフセットとサイズとをそれぞれ書き換える。

[0108] 具体的には、シンボルテーブルのエントリーのオフセット k を用いて、オフセットとオフセットとの対応表からオフセット対 (k, l) を検索する。そして、オフセットの値を l で更新する。また、サイズは $j - l$ となる。なお、この場合 j は、移動後のシンボルの末尾のオフセット $+1$ の値を保持しており、 l は移動後のシンボル先頭のオフセットを保持している。

[0109] ステップ $ST1410$ では、オブジェクトファイル編集部 106 が、オフセット $i = 0$ から $i = \text{セクションサイズ} - 1$ までインクリメントされたか否かを判定する。

[0110] そして、オフセット $i = 0$ から $i = \text{セクションサイズ} - 1$ までインクリメントされた場合、すなわち、図 14 に例が示されるステップ $ST1410$ から分岐する「YES」に対応する場合には、図 14 に例が示されるステップ $ST1411$ へ進む。一方で、オフセット $i = 0$ から $i = \text{セクションサイズ} - 1$ までインクリメントされていない場合、すなわち、図 14 に例が示されるステップ $ST1410$ から分岐する「NO」に対応する場合には、図 14 に例が示されるステップ $ST1402$ に戻る。

[0111] ここで、シンボルテーブルのエントリーの更新の例として、.textセクションを図 8 に示されるものから図 9 に示されるものへ書き換えた場合の、シンボルテーブルのエントリーの書き換えを考える。

[0112] 図 15 は、上記の書き換え前のシンボルテーブルの例を示す図である。また、図 16 は、上記の書き換え後のシンボルテーブルの例を示す図である。

[0113] 図 15 に例が示される書き換え前のシンボルテーブルのエントリー 1501 は、関数 $calc$ に対応している。関数 $calc$ のアドレスは、図 8 から図 9 への書き換えで示されたように、オフセット 0×14 からオフセット 0×00 へ移動している。このため、図 16 に示される書き換え後のエントリー 1601 では、シンボルの値は 0×0 に書き換えられている。また、対応するシンボルのサイズも、 84 バイトから 60 バイトに書き換えられている。この書き換えは、 $i = 67$ の場合、すなわち、関数 $calc$ の末尾の場合

に、ステップS T 1 4 0 9において書き換えられる。

[0114] この際、シンボルテーブルの検索によってオフセット $k = 0 \times 14$ 、すなわち、c a l c の先頭が得られる。次に k を用いてオフセットとオフセットとの対応表を検索すると ($k = 0 \times 14$ 、 $l = 0$) のペアが得られる。図 16 に示される書き換え後のエントリー 1 6 0 1 のシンボルの値 0×0 は、この場合に得られる l の値である。また、この際の j は $0 \times 3C$ であり、 $j - l$ 、すなわち、 $0 \times 3C - 0 \times 0 = 0 \times 3C$ (10進数で60) が新しいシンボルのサイズとなる。

[0115] 書き換え対象がコード領域に対応するセクションである場合、ステップS T 1 4 0 2 からステップS T 1 4 1 0 における書き換えによって、分岐命令の分岐先が移動する場合がある。ステップS T 1 4 1 1 では、オブジェクトファイル編集部 1 0 6 が、分岐命令の分岐先を移動後のものに書き換える。

[0116] 図 1 7 は、図 1 4 のステップS T 1 4 1 1 の分岐命令書き換え処理の詳細な処理フローを示すフローチャートである。図 1 7 に例が示される場合では、セクションと分岐命令先頭アドレス集合とを入力とし、当該セクションに存在する分岐命令のうち分岐先が変更された命令について分岐先を書き換える。

[0117] 図 1 7 に例が示されるように、ステップS T 1 7 0 1 では、オブジェクトファイル編集部 1 0 6 がループ処理を開始する。当該ループ処理は、オフセットとオフセットとの対応表のすべてのエントリーのオフセット対 (i 、 j) に対して処理が行われるまで実行を繰り返す処理である。

[0118] ステップS T 1 7 0 2 では、オブジェクトファイル編集部 1 0 6 が、書き換え前のオフセット i を用いてアドレスとオフセットとの対応表からアドレス x を取得する。

[0119] ステップS T 1 7 0 3 では、オブジェクトファイル編集部 1 0 6 が、アドレス x が命令先頭アドレス集合に含まれるか否かを判定する。

[0120] そして、アドレス x が命令先頭アドレス集合に含まれる場合、すなわち、図 1 7 に例が示されるステップS T 1 7 0 3 から分岐する「YES」に対応

する場合には、書き換え後のオフセット j が実行された命令の先頭のアドレスであると判定し、図 17 に例が示されるステップ ST 1704 へ進む。一方で、アドレス x が命令先頭アドレス集合に含まれていない場合、すなわち、図 17 に例が示されるステップ ST 1703 から分岐する「NO」に対応する場合には、書き換え後のオフセット j が実行された命令の先頭のアドレスではないと判定し、図 17 に例が示されるステップ ST 1709 へ進む。

[0121] ステップ ST 1704 では、オブジェクトファイル編集部 106 が、オフセット j のデータを命令としてデコードする。

[0122] ステップ ST 1705 では、オブジェクトファイル編集部 106 が、ステップ ST 1704 におけるデコードの結果に基づいて、オフセット j の命令が直接分岐命令であるか否かを判定する。ここで、直接分岐命令は、レジスタ間接分岐命令以外の分岐命令である。

[0123] そして、オフセット j の命令が直接分岐命令である場合、すなわち、図 17 に例が示されるステップ ST 1705 から分岐する「YES」に対応する場合には、図 17 に例が示されるステップ ST 1704 へ進む。一方で、オフセット j の命令が直接分岐命令ではない場合、すなわち、図 17 に例が示されるステップ ST 1705 から分岐する「NO」に対応する場合には、図 17 に例が示されるステップ ST 1709 へ進む。

[0124] ステップ ST 1706 では、オブジェクトファイル編集部 106 が、分岐命令の分岐先アドレスを計算する。直接分岐の例としては PC 相対の分岐を考える。PC 相対では、オフセット j からの相対アドレス k (k は正または負の整数) が命令フィールドに格納されている。ステップ ST 1706 では、オブジェクトファイル編集部 106 が、 $i + k$ を書き換え前のオフセットとして用いて、オフセットとオフセットとの対応表を検索する。

[0125] ステップ ST 1707 では、オブジェクトファイル編集部 106 が、上記の検索が成功したか否か、すなわち、オフセット対 ($i + k$, m) が得られたか否かを判定する。

[0126] そして、上記の検索が成功した場合、すなわち、図 17 に例が示されるス

テップST1707から分岐する「YES」に対応する場合には、図17に例が示されるステップST1708へ進む。一方で、上記の検索が成功しなかった場合、すなわち、図17に例が示されるステップST1707から分岐する「NO」に対応する場合には、図17に例が示されるステップST1709へ進む。

[0127] ステップST1708では、オブジェクトファイル編集部106が、得られたオフセット対($i+k$, m)に基づいて、 $m-i$ を命令フィールドに書き込む。すなわち、分岐先のオフセットを書き換える。

[0128] なお、ここではPC相対アドレスに使われるベースアドレスが分岐命令のアドレスである場合が示されたが、ベースアドレスが分岐命令の次の命令のアドレスの場合も、同様の考え方で格納すべき値を計算することができる。

[0129] ステップST1709では、オブジェクトファイル編集部106が、オフセットとオフセットとの対応表のすべてのエントリーのオフセット対(i , j)に対して処理が行われたか否かを判定する。

[0130] そして、オフセットとオフセットとの対応表のすべてのエントリーのオフセット対(i , j)に対して処理が行われた場合、すなわち、図17に例が示されるステップST1709から分岐する「YES」に対応する場合には、動作を終了する。一方で、処理が行われていないエントリーのオフセット対が残っている場合、すなわち、図17に例が示されるステップST1709から分岐する「NO」に対応する場合には、図17に例が示されるステップST1701に戻る。

[0131] 図18は、図6のステップST606のリロケーションテーブル書き換え処理の詳細な処理フローを示すフローチャートである。図18に例が示される場合では、入力が入力オブジェクトファイル101、セクション名、オフセットとオフセットとの対応表であり、出力はリロケーションテーブルが修正された出力オブジェクトファイル103である。

[0132] 図18に例が示されるように、ステップST1801では、オブジェクトファイル編集部106がループ処理を開始する。当該ループ処理は、入力さ

れたセクション名に対応するリロケーションテーブルのすべてのエントリーに対して処理が行われるまで実行を繰り返す処理である。入力されたセクション名が、`text`の場合、ELFファイルでは、リロケーションテーブルは、`text.reloc`を名前とするセクションに格納されている。

[0133] ステップST1802では、オブジェクトファイル編集部106が、リロケーションエントリーのオフセット*i*を書き換え前のオフセットとして、オフセットとオフセットとの対応表からオフセット対を検索する。

[0134] ステップST1803では、オブジェクトファイル編集部106が、上記の検索が成功したか否か、すなわち、オフセット対(*i*、*j*)が得られたか否かを判定する。

[0135] そして、上記の検索が成功した場合、すなわち、図18に例が示されるステップST1803から分岐する「YES」に対応する場合には、図18に例が示されるステップST1804へ進む。一方で、上記の検索が成功しなかった場合、すなわち、図18に例が示されるステップST1803から分岐する「NO」に対応する場合には、図18に例が示されるステップST1805へ進む。

[0136] ステップST1804では、オブジェクトファイル編集部106が、リロケーションテーブルのエントリーに格納されたオフセット*i*を、書き換え後のオフセット*j*で上書きする。

[0137] ステップST1805では、オブジェクトファイル編集部106が、入力されたセクション名に対応するリロケーションテーブルのすべてのエントリーに対して処理が行われたか否かを判定する。

[0138] そして、入力されたセクション名に対応するリロケーションテーブルのすべてのエントリーに対して処理が行われた場合、すなわち、図18に例が示されるステップST1805から分岐する「YES」に対応する場合には、動作を終了する。一方で、処理が行われていないエントリーが残っている場合、すなわち、図18に例が示されるステップST1805から分岐する「NO」に対応する場合には、図18に例が示されるステップST1801に

戻る。

[0139] <プログラムサイズ低減装置のハードウェア構成について>

図19および図20は、図1に例が示されるプログラムサイズ低減装置を実際に運用する場合のハードウェア構成を概略的に例示する図である。

[0140] なお、図19および図20に例示されるハードウェア構成は、図1に例示される構成とは数などが整合しない場合があるが、これは図1に例示される構成が概念的な単位を示すものであることに起因する。

[0141] よって、少なくとも、図1に例示される1つの構成が、図19および図20に例示される複数のハードウェア構成から成る場合と、図1に例示される1つの構成が、図19および図20に例示されるハードウェア構成の一部に対応する場合と、さらには、図1に例示される複数の構成が、図19および図20に例示される1つのハードウェア構成に備えられる場合とが想定され得る。

[0142] 図19では、図1中のプログラム実行部104、メモリ領域特定部105、オブジェクトファイル編集部106および対応表作成部107を実現するためのハードウェア構成として、演算を行う処理回路1102Aと、情報を記憶することができる記憶装置1103とが示される。

[0143] 図20では、図1中のプログラム実行部104、メモリ領域特定部105、オブジェクトファイル編集部106および対応表作成部107を実現するためのハードウェア構成として、演算を行う処理回路1102Bが示される。

[0144] アドレスとオフセットとの対応表などを記憶する記憶部（図示せず）は、記憶装置1103または別の記憶装置（ここでは、図示しない）によって実現される。

[0145] 記憶装置1103は、たとえば、ハードディスクドライブ（Hard disk drive、すなわち、HDD）、ランダムアクセスメモリ（random access memory、すなわち、RAM）、リードオンリーメモリ（read only memory、すなわち、ROM）、フ

ラッシュメモリ、erasable programmable read only memory (EPROM) および electrically erasable programmable read-only memory (EEPROM) などの、揮発性または不揮発性の半導体メモリ、磁気ディスク、フレキシブルディスク、光ディスク、コンパクトディスク、ミニディスクまたはDVDなどを含むメモリ（記憶媒体）、または、今後使用されるあらゆる記憶媒体であってもよい。

[0146] 処理回路1102Aは、記憶装置1103、外部のCD-ROM、外部のDVD-ROM、または、外部のフラッシュメモリなどに格納されたプログラムを実行するものであってもよい。すなわち、たとえば、中央演算処理装置 (central processing unit、すなわち、CPU)、マイクロプロセッサ、マイクロコンピュータ、デジタルシグナルプロセッサ (digital signal processor、すなわち、DSP) であってもよい。

[0147] 処理回路1102Aが記憶装置1103、外部のCD-ROM、外部のDVD-ROM、または、外部のフラッシュメモリなどに格納されたプログラムを実行するものである場合、プログラム実行部104、メモリ領域特定部105、オブジェクトファイル編集部106および対応表作成部107は、記憶装置1103に格納されたプログラムが処理回路1102Aによって実行されるソフトウェア、ファームウェアまたはソフトウェアとファームウェアとの組み合わせにより実現される。なお、プログラム実行部104、メモリ領域特定部105、オブジェクトファイル編集部106および対応表作成部107の機能は、たとえば、複数の処理回路が連携することによって実現されてもよい。

[0148] ソフトウェアおよびファームウェアはプログラムとして記述され、記憶装置1103に記憶されるものであってもよい。その場合、処理回路1102Aは、記憶装置1103に格納されたプログラムを読み出して実行することによって、上記の機能を実現する。すなわち、記憶装置1103は、処理回

路 1102A に実行されることによって、上記の機能が結果的に実現されるプログラムを記憶するものであってもよい。

[0149] また、処理回路 1102B は、専用のハードウェアであってもよい。すなわち、たとえば、単回路、複合回路、プログラム化されたプロセッサ、並列プログラム化されたプロセッサ、集積回路 (application specific integrated circuit、すなわち、ASIC)、field-programmable gate array (FPGA) またはこれらを組み合わせた回路であってもよい。

[0150] 処理回路 1102B が専用のハードウェアである場合、プログラム実行部 104、メモリ領域特定部 105、オブジェクトファイル編集部 106 および対応表作成部 107 は、処理回路 1102B が動作することにより実現される。なお、プログラム実行部 104、メモリ領域特定部 105、オブジェクトファイル編集部 106 および対応表作成部 107 の機能は、別々の回路で実現されてもよいし、単一の回路で実現されてもよい。

[0151] なお、プログラム実行部 104、メモリ領域特定部 105、オブジェクトファイル編集部 106 および対応表作成部 107 の機能は、一部が記憶装置 1103 に格納されたプログラムを実行するものである処理回路 1102A において実現され、一部が専用のハードウェアである処理回路 1102B において実現されてもよい。

[0152] <以上に記載された実施の形態によって生じる効果について>

次に、以上に記載された実施の形態によって生じる効果の例を示す。なお、以下の説明においては、以上に記載された実施の形態に例が示された具体的な構成に基づいて当該効果が記載されるが、同様の効果が生じる範囲で、本願明細書に例が示される他の具体的な構成と置き換えられてもよい。

[0153] 以上に記載された実施の形態によれば、プログラムサイズ低減装置は、メモリ領域特定部 105 と、対応表作成部 107 と、オブジェクトファイル編集部 106 とを備える。メモリ領域特定部 105 は、入力オブジェクトファイル 101 にリンクされた実行可能ファイル 102 の実行によってアクセス

される、実行可能ファイル102における第1のソースコードの第1のアドレス（すなわち、実行可能ファイル102の実行時に必要となる実行可能ファイル102のソースコードのアドレス）を、メモリ領域として特定する。対応表作成部107は、メモリ領域における第1のアドレスと、入力オブジェクトファイル101における第2のソースコードのセクション先頭からの第1のオフセットとの対応表を作成する。オブジェクトファイル編集部106は、対応表に基づいて、メモリ領域における第1のアドレスに対応しない第1のオフセットの第2のソースコード（すなわち、実行可能ファイル102の実行時に必要でない実行可能ファイル102のソースコードに対応するオフセットを有する、入力オブジェクトファイル101のソースコード）を削除する編集を行う。

[0154] また、以上に記載された実施の形態によれば、プログラムサイズ低減装置は、プログラムを実行する処理回路1102Aと、実行されるプログラムを記憶する記憶装置1103とを備える。そして、処理回路1102Aがプログラムを実行することによって、以下の動作が実現される。

[0155] すなわち、入力オブジェクトファイル101にリンクされた実行可能ファイル102の実行によってアクセスされる、実行可能ファイル102における第1のバイナリコードの第1のアドレスが、メモリ領域として特定される。そして、メモリ領域における第1のアドレスと、入力オブジェクトファイル101における第2のバイナリコードのセクション先頭からの第1のオフセットとの対応表に基づいて、メモリ領域における第1のアドレスに対応しない第1のオフセットの第2のバイナリコードが削除される。

[0156] また、以上に記載された実施の形態によれば、プログラムサイズ低減装置は、専用のハードウェアである処理回路1102Bを備える。そして、専用のハードウェアである処理回路1102Bは、以下の動作を行う。

[0157] すなわち、専用のハードウェアである処理回路1102Bは、入力オブジェクトファイル101にリンクされた実行可能ファイル102の実行によってアクセスされる、実行可能ファイル102における第1のバイナリコード

の第1のアドレスを、メモリ領域として特定する。そして、メモリ領域における第1のアドレスと、入力オブジェクトファイル101における第2のバイナリコードのセクション先頭からの第1のオフセットとの対応表に基づいて、メモリ領域における第1のアドレスに対応しない第1のオフセットの第2のバイナリコードを削除する編集を行う。

[0158] このような構成によれば、実行可能ファイル102を直接書き換えるのではなく、ソフトウェアを構築するとき際に用いられた入力オブジェクトファイル101を書き換えることによってプログラムサイズを低減させることができる。そのため、書き換えられた出力オブジェクトファイル103を他のプログラムでも再利用することができる。入力オブジェクトファイル101から一旦特定の機能を抽出して出力オブジェクトファイル103を作成すると、当該機能のみを有する出力オブジェクトファイル103を他のプログラムに再利用することができる。たとえば、セキュリティーライブラリであるOpenSSLをリンクし、AES-GCM-256の暗号化テストベクタを実行する実行可能ファイル102を入力として与えると、暗号アルゴリズムがAES、暗号モードがGCM、256バイトの鍵長に特化した機能を有する軽量なライブラリを得ることができる。

[0159] なお、上記の構成に本願明細書に例が示された他の構成を適宜追加した場合、すなわち、上記の構成としては言及されなかった本願明細書中の他の構成が適宜追加された場合であっても、同様の効果を生じさせることができる。

[0160] また、以上に記載された実施の形態によれば、第1のバイナリコードに対応する命令がコード領域へのメモリ読み出しを伴う命令である。そして、メモリ領域特定部105は、少なくとも、第1のバイナリコードの第1のアドレス（すなわち、実行可能ファイル102の実行時に必要となる実行可能ファイル102のバイナリコードのアドレス）と、命令による読み出し先の第2のアドレス（たとえば、アクセス先のアドレスX）とをメモリ領域として特定する。このような構成によれば、コード領域に命令以外のデータが埋め

込まれている場合でも、削除することができないアドレス（すなわち、実行可能ファイル 102 の実行時に必要となる実行可能ファイル 102 のバイナリコードのアドレス）を、メモリ領域として特定することができる。

[0161] また、以上に記載された実施の形態によれば、メモリ領域特定部 105 は、第 1 のバイナリコードの第 1 のアドレスを含む必要アドレス集合と、第 1 のバイナリコードに対応する命令の先頭のアドレスを含む命令先頭アドレス集合とを用いて、メモリ領域を特定する。このような構成によれば、実行可能ファイル 102 におけるバイナリコードに対応する命令が可変長の命令である場合でも、適切にメモリ領域を特定することができる。たとえば、ARM の Thumb 命令セットまたは PowerPC の VLE などの命令セットにも対応して、メモリ領域を特定することができる。

[0162] また、以上に記載された実施の形態によれば、オブジェクトファイル編集部 106 は、メモリ領域における第 1 のアドレスに対応しない第 1 のオフセットの第 2 のバイナリコードに対応する命令が直接分岐命令である場合、直接分岐命令による分岐先の第 2 のオフセットを書き換える編集を行う。このような構成によれば、プログラムサイズの低減によって直接分岐命令の分岐先のオフセットが移動する場合であっても、再利用可能な状態で入力オブジェクトファイル 101 を編集することができる。すなわち、直接分岐命令を含む入力オブジェクトファイル 101 であっても、適切にプログラムサイズを低減することができる。

[0163] また、以上に記載された実施の形態によれば、オブジェクトファイル編集部 106 は、メモリ領域における第 1 のアドレスに対応しない第 1 のオフセットの第 2 のバイナリコードの削除に伴ってリロケーション箇所が書き換えられた場合、リロケーションテーブルにおけるリロケーション箇所を示す第 3 のオフセットを書き換える編集を行う。このような構成によれば、プログラムサイズを低減する際にリロケーション箇所のバイナリコードまたはデータを削除せずに残す場合であっても、再利用可能な状態で入力オブジェクトファイル 101 を編集することができる。

- [0164] 以上に記載された実施の形態によれば、プログラムサイズ低減プログラムは、コンピュータ（たとえば、処理回路1102Aまたは処理回路1102B）にインストールされて実行されることによって、コンピュータに、入力オブジェクトファイル101にリンクされた実行可能ファイル102の実行によってアクセスされる、実行可能ファイル102における第1のバイナリコードの第1のアドレスを、メモリ領域として特定させる。そして、コンピュータに、メモリ領域における第1のアドレスと、入力オブジェクトファイル101における第2のバイナリコードのセクション先頭からの第1のオフセットとの対応表に基づいて、メモリ領域における第1のアドレスに対応しない第1のオフセットの第2のバイナリコードを削除する編集を行わせる。
- [0165] このような構成によれば、実行可能ファイル102を直接書き換えるのではなく、ソフトウェアを構築するとき際に用いられた入力オブジェクトファイル101を書き換えることによってプログラムサイズを低減させることができる。そのため、書き換えられた出力オブジェクトファイル103を他のプログラムでも再利用することができる。
- [0166] なお、本願明細書に例が示される他の構成のうちの少なくとも1つを、上記の構成に適宜追加した場合、すなわち、上記の構成としては言及されなかった本願明細書に例が示される他の構成が適宜追加された場合であっても、同様の効果を生じさせることができる。
- [0167] また、上記のプログラムは、磁気ディスク、フレキシブルディスク、光ディスク、コンパクトディスク、ブルーレイ（登録商標）ディスクまたはDVDなどのコンピュータが読取可能な可搬記録媒体に記憶されていてもよい。そして、上記の機能を実現するプログラムが記憶された可搬記録媒体が商業的に流通されてもよい。
- [0168] 以上に記載された実施の形態によれば、プログラムサイズ低減方法において、入力オブジェクトファイル101にリンクされた実行可能ファイル102の実行によってアクセスされる、実行可能ファイル102における第1のバイナリコードの第1のアドレスを、メモリ領域として特定する。そして、

メモリ領域における第1のアドレスと、入力オブジェクトファイル101における第2のバイナリコードのセクション先頭からの第1のオフセットとの対応表に基づいて、メモリ領域における第1のアドレスに対応しない第1のオフセットの第2のバイナリコードを削除する。

[0169] このような構成によれば、実行可能ファイル102を直接書き換えるのではなく、ソフトウェアを構築するとき際に用いられた入力オブジェクトファイル101を書き換えることによってプログラムサイズを低減させることができる。そのため、書き換えられた出力オブジェクトファイル103を他のプログラムでも再利用することができる。

[0170] なお、特段の制限がない場合には、それぞれの処理が行われる順序は変更することができる。

[0171] また、上記の構成に本願明細書に例が示された他の構成を適宜追加した場合、すなわち、上記の構成としては言及されなかった本願明細書中の他の構成が適宜追加された場合であっても、同様の効果を生じさせることができる。

[0172] <以上に記載された実施の形態の変形例について>

以上に記載された実施の形態では、それぞれの構成要素の寸法、形状、相対的配置関係または実施の条件などについても記載する場合があるが、これらはすべての局面においてひとつの例であって、限定的なものではないものとする。

[0173] したがって、例が示されていない無数の変形例、および、均等物が、本願明細書に開示される技術の範囲内において想定される。たとえば、少なくとも1つの構成要素を変形する場合、追加する場合または省略する場合が含まれるものとする。

[0174] また、矛盾が生じない限り、以上に記載された実施の形態において「1つ」備えられるものとして記載された構成要素は、「1つ以上」備えられていてもよいものとする。

[0175] さらに、以上に記載された実施の形態におけるそれぞれの構成要素は概念

的な単位であって、本願明細書に開示される技術の範囲内には、1つの構成要素が複数の構造物から成る場合と、1つの構成要素がある構造物の一部に対応する場合と、さらには、複数の構成要素が1つの構造物に備えられる場合とを含むものとする。

[0176] また、以上に記載された実施の形態におけるそれぞれの構成要素には、同一の機能を発揮する限り、他の構造または形状を有する構造物が含まれるものとする。

[0177] また、本願明細書における説明は、本技術に関連するすべての目的のために参照され、いずれも、従来技術であると認めるものではない。

[0178] また、以上に記載された実施の形態で記載されたそれぞれの構成要素は、ソフトウェアまたはファームウェアとしても、それと対応するハードウェアとしても想定され、その双方の概念において、それぞれの構成要素は「部」または「処理回路」(c i r c u i t r y)などと称される。

符号の説明

[0179] 10 プログラムサイズ低減装置、101 入力オブジェクトファイル、102 実行可能ファイル、103 出力オブジェクトファイル、104 プログラム実行部、105, 3001 メモリ領域特定部、106, 3003 オブジェクトファイル編集部、107, 3002 対応表作成部、801, 901 リロケーション対象箇所、1102A, 1102B 処理回路、1103 記憶装置。

請求の範囲

【請求項1】 入力オブジェクトファイルにリンクされた実行可能ファイルの実行によってアクセスされる、前記実行可能ファイルにおける第1のバイナリコードの第1のアドレスを、メモリ領域として特定するためのメモリ領域特定部と、

前記メモリ領域における前記第1のアドレスと、前記入力オブジェクトファイルにおける第2のバイナリコードのセクション先頭からの第1のオフセットとの対応表を作成するための対応表作成部と、

前記対応表に基づいて、前記メモリ領域における前記第1のアドレスに対応しない前記第1のオフセットの前記第2のバイナリコードを削除する編集を行うためのオブジェクトファイル編集部とを備える、プログラムサイズ低減装置。

【請求項2】 請求項1に記載のプログラムサイズ低減装置であり、

前記第1のバイナリコードに対応する命令がコード領域へのメモリ読み出しを伴う命令であり、

前記メモリ領域特定部は、少なくとも、前記第1のバイナリコードの前記第1のアドレスと、前記命令による読み出し先の第2のアドレスとを前記メモリ領域として特定する、

プログラムサイズ低減装置。

【請求項3】 請求項1または2に記載のプログラムサイズ低減装置であり、

前記メモリ領域特定部は、前記第1のバイナリコードの前記第1のアドレスを含む必要アドレス集合と、前記第1のバイナリコードに対応する命令の先頭のアドレスを含む命令先頭アドレス集合とを用いて、前記メモリ領域を特定する、

プログラムサイズ低減装置。

【請求項4】 請求項1から3のうちのいずれか1つに記載のプログラムサイズ低減装置であり、

前記オブジェクトファイル編集部は、前記メモリ領域における前記

第1のアドレスに対応しない前記第1のオフセットの前記第2のバイナリコードに対応する命令が直接分岐命令である場合、前記直接分岐命令による分岐先の第2のオフセットを書き換える編集を行う、

プログラムサイズ低減装置。

[請求項5] 請求項1から4のうちのいずれか1つに記載のプログラムサイズ低減装置であり、

前記オブジェクトファイル編集部は、前記メモリ領域における前記第1のアドレスに対応しない前記第1のオフセットの前記第2のバイナリコードの削除に伴ってリロケーション箇所が書き換えられた場合、リロケーションテーブルにおける前記リロケーション箇所を示す第3のオフセットを書き換える編集を行う、

プログラムサイズ低減装置。

[請求項6] コンピュータにインストールされて実行されることによって、

前記コンピュータに、入力オブジェクトファイルにリンクされた実行可能ファイルの実行によってアクセスされる、前記実行可能ファイルにおける第1のバイナリコードの第1のアドレスを、メモリ領域として特定させ、

前記コンピュータに、前記メモリ領域における前記第1のアドレスと、前記入力オブジェクトファイルにおける第2のバイナリコードのセクション先頭からの第1のオフセットとの対応表に基づいて、前記メモリ領域における前記第1のアドレスに対応しない前記第1のオフセットの前記第2のバイナリコードを削除する編集を行わせる、

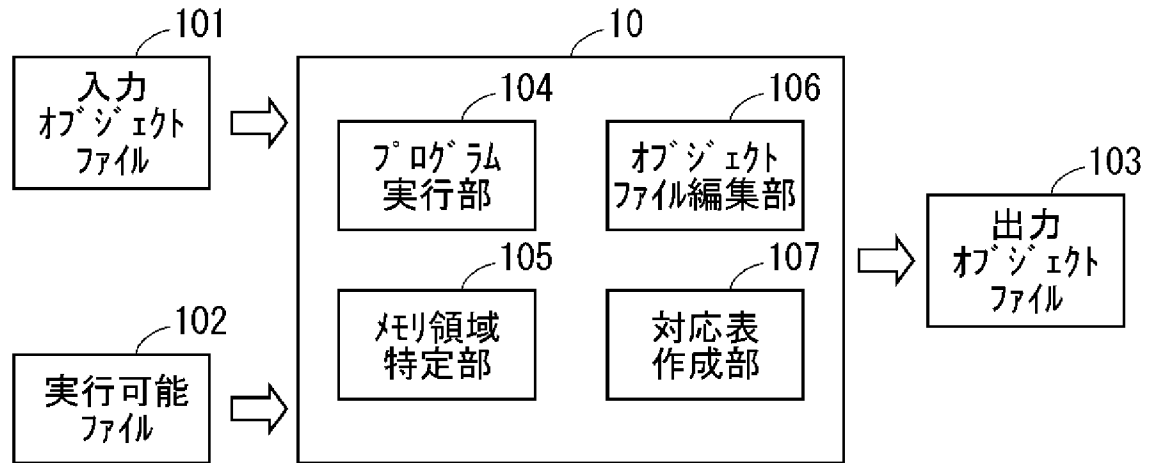
プログラムサイズ低減プログラム。

[請求項7] 入力オブジェクトファイルにリンクされた実行可能ファイルの実行によってアクセスされる、前記実行可能ファイルにおける第1のバイナリコードの第1のアドレスを、メモリ領域として特定し、

前記メモリ領域における前記第1のアドレスと、前記入力オブジェクトファイルにおける第2のバイナリコードのセクション先頭からの

第 1 のオフセットとの対応表に基づいて、前記メモリ領域における前記第 1 のアドレスに対応しない前記第 1 のオフセットの前記第 2 のバイナリコードを削除する編集を行う、
プログラムサイズ低減方法。

[図1]



[図2]

```
#include "calc.h"

int mod(int x, int y) { return x % y; }

int calc(int operation, int x, int y)
{
    int result = 0;

    switch (operation) {
        case ADD:
            result = x + y;
            break;
        case SUB:
            result = x - y;
            break;
        case MUL:
            result = x * y;
            break;
        case DIV:
            result = x / y;
        default:
            break;
    }

    return result;
}
```

[図3]

```
#include "calc.h"

int calc_average(int x, int y)
{
    int sum = calc(ADD, x, y);
    int average = calc(DIV, sum , 2);
    return average;
}

int main(void)
{
    calc_average(1, 99);
}
```

[図4]

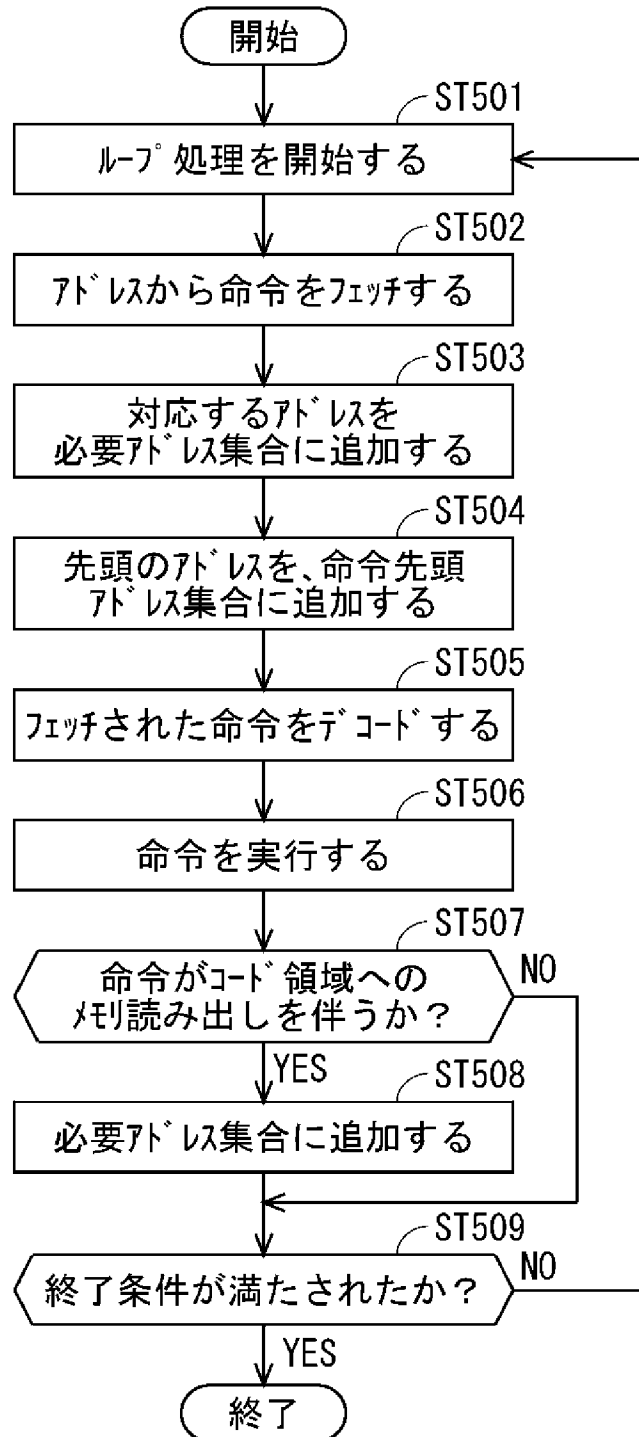
```

000082a0 <mod>:
  82a0: e92d4010  push
  82a4: eb000061  bl
  82a8: e8bd4010  pop
  82ac: e1a00001  mov
  82b0: e12fff1e  bx

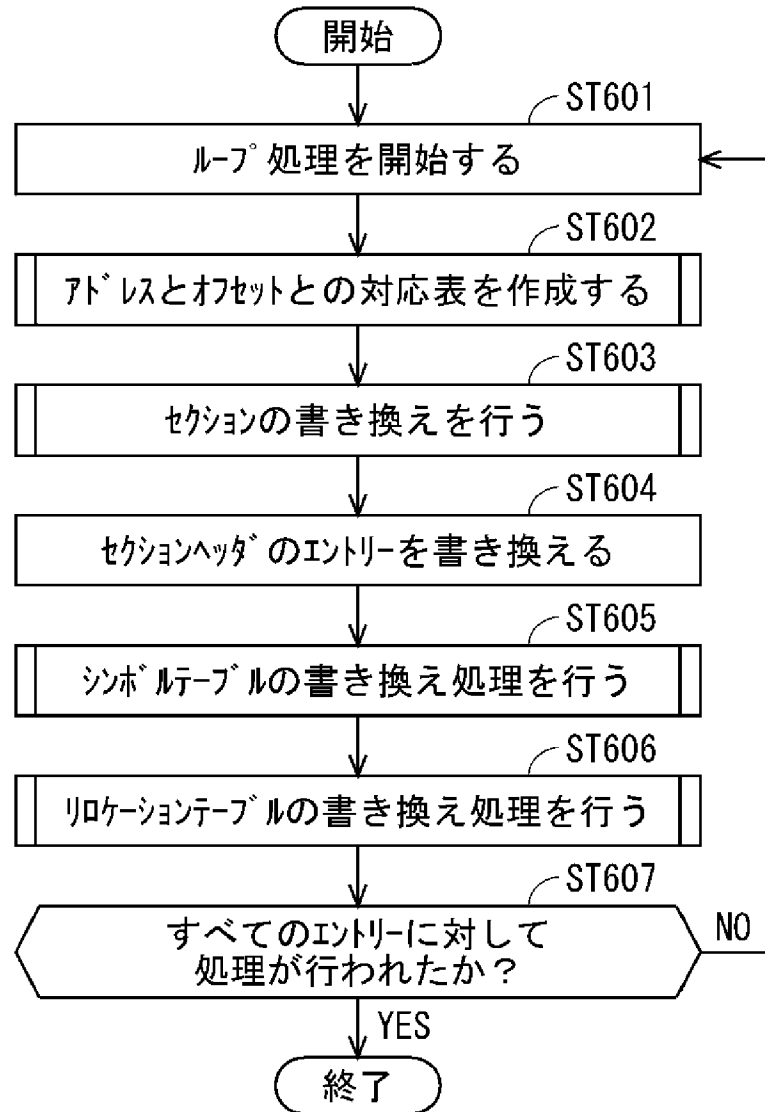
000082b4 <calc>:
x  82b4: e3500003  cmp
x  82b8: 979ff100  ldrls
x  82bc: ea00000f  b
x  82c0: 000082f8  .word
x  82c4: 000082f0  .word
x  82c8: 000082e8  .word
x  82cc: 000082d0  .word
x  82d0: e92d4010  push
x  82d4: e1a00001  mov
x  82d8: e1a01002  mov
x  82dc: eb000009  bl
  82e0: e8bd4010  pop
  82e4: e12fff1e  bx
  82e8: e0000192  mul
  82ec: e12fff1e  bx
  82f0: e0410002  sub
  82f4: e12fff1e  bx
x  82f8: e0810002  add
x  82fc: e12fff1e  bx
  8300: e3a00000  mov
  8304: e12fff1e  bx
  {r4, lr}
  8430 <__aeabi_idivmod>
  {r4, lr}
  r0, r1
  lr
  r0, #3
  pc, [pc, r0, lsl #2]
  8300 <calc+0x4c>
  0x000082f8
  0x000082f0
  0x000082e8
  0x000082d0
  {r4, lr}
  r0, r1
  r1, r2
  8308 <__aeabi_idiv>
  {r4, lr}
  lr
  r0, r2, r1
  lr
  r0, r1, r2
  lr
  r0, r1, r2
  lr
  r0, #0
  lr

```

[図5]



[図6]



[図7]

セクションヘッダ:									
[番]	名前	タイプ	アドレス	オフセット	サイズ	ES	Flg	Lk	Inf Al
[0]		NULL	00000000	000000	0000000	00		0	0 0
[1]	.text	PROGBITS	00000000	000034	000068	00	AX	0	0 4
[2]	.rel.text	REL	00000000	0001f0	000060	08	I	8	1 4
[3]	.data	PROGBITS	00000000	00009c	000000	00	WA	0	0 1
[4]	.bss	NOBITS	00000000	00009c	000000	00	WA	0	0 1
[5]	.comment	PROGBITS	00000000	00009c	000012	01	MS	0	0 1
[6]	.ARM.attributes	ARM_ATTRIBUTES	00000000	0000ae	000030	00		0	0 1
[7]	.shstrtab	STRTAB	00000000	000250	000049	00		0	0 1
[8]	.symtab	SYMTAB	00000000	0000e0	0000e0	10		9	10 4
[9]	.strtab	STRTAB	00000000	0001c0	000030	00		0	0 1

[図8]

calc.o: file format elf32-littlearm

Disassembly of section .text:

00000000 <mod>:

0:	e92d4010	push	{r4, lr}
4:	ebfffffe	bl	0 <__aeabi_idivmod>
8:	e8bd4010	pop	{r4, lr}
c:	e1a00001	mov	r0, r1
10:	e12fff1e	bx	lr

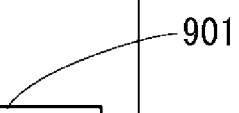
00000014 <calc>:

x 14:	e3500003	cmp	r0, #3
x 18:	979ff100	ldrls	pc, [pc, r0, lsl #2]
x 1c:	ea00000f	b	60 <calc+0x4c>
x 20:	00000058	.word	0x00000058
x 24:	00000050	.word	0x00000050
x 28:	00000048	.word	0x00000048
x 2c:	00000030	.word	0x00000030
x 30:	e92d4010	push	{r4, lr}
x 34:	e1a00001	mov	r0, r1
x 38:	e1a01002	mov	r1, r2
x 3c:	ebfffffe	bl	0 <__aeabi_idiv>
x 40:	e8bd4010	pop	{r4, lr}
x 44:	e12fff1e	bx	lr
48:	e0000192	mul	r0, r2, r1
4c:	e12fff1e	bx	lr
50:	e0410002	sub	r0, r1, r2
54:	e12fff1e	bx	lr
x 58:	e0810002	add	r0, r1, r2
x 5c:	e12fff1e	bx	lr
60:	e3a00000	mov	r0, #0
64:	e12fff1e	bx	lr

801

[図9]

00:	e3500003	cmp	r0, #3
04:	979ff100	ldrls	pc, [pc, r0, lsl #2]
08:	ea00000f	b	60 <calc+0x4c>
0c:	00000058	.word	0x00000058
10:	00000050	.word	0x00000050
14:	00000048	.word	0x00000048
18:	00000030	.word	0x00000030
1c:	e92d4010	push	{r4, lr}
20:	e1a00001	mov	r0, r1
24:	e1a01002	mov	r1, r2
28:	ebfffffe	bl	0 <__aeabi_idiv>
2c:	e8bd4010	pop	{r4, lr}
30:	e12fff1e	bx	lr
34:	e0810002	add	r0, r1, r2
38:	e12fff1e	bx	lr

A curved line originates from the label '901' on the right side of the page and points to the instruction at address 28: 'ebfffffe bl 0 <__aeabi_idiv>'. This instruction is highlighted with a rectangular box.

[図10]

セクションヘッダ:			タイプ		アドレス		オフセット		サイズ		ES	Flg	Lk	Inf	Al
[番]	名前														
[0]			NULL		00000000	00000000	00000000	00000000	00000000	00	00		0	0	0
[1]	.text		PROGBITS		00000000	000034	00003c	00	AX	00	00	00	0	0	4
[2]	.rel.text		REL		00000000	0001f0	000060	08	I	8	08	00	1	1	4
[3]	.data		PROGBITS		00000000	00009c	000000	00	WA	00	00	00	0	0	1
[4]	.bss		NOBITS		00000000	00009c	000000	00	WA	00	00	00	0	0	1
[5]	.comment		PROGBITS		00000000	00009c	000012	01	MS	00	00	00	0	0	1
[6]	.ARM.attributes		ARM_ATTRIBUTES		00000000	0000ae	000030	00		00	00	00	0	0	1
[7]	.shstrtab		STRTAB		00000000	000250	000049	00		00	00	00	0	0	1
[8]	.symtab		SYMTAB		00000000	0000e0	0000e0	10		10	10	10	9	10	4
[9]	.strtab		STRTAB		00000000	0001c0	000030	00		00	00	00	0	0	1

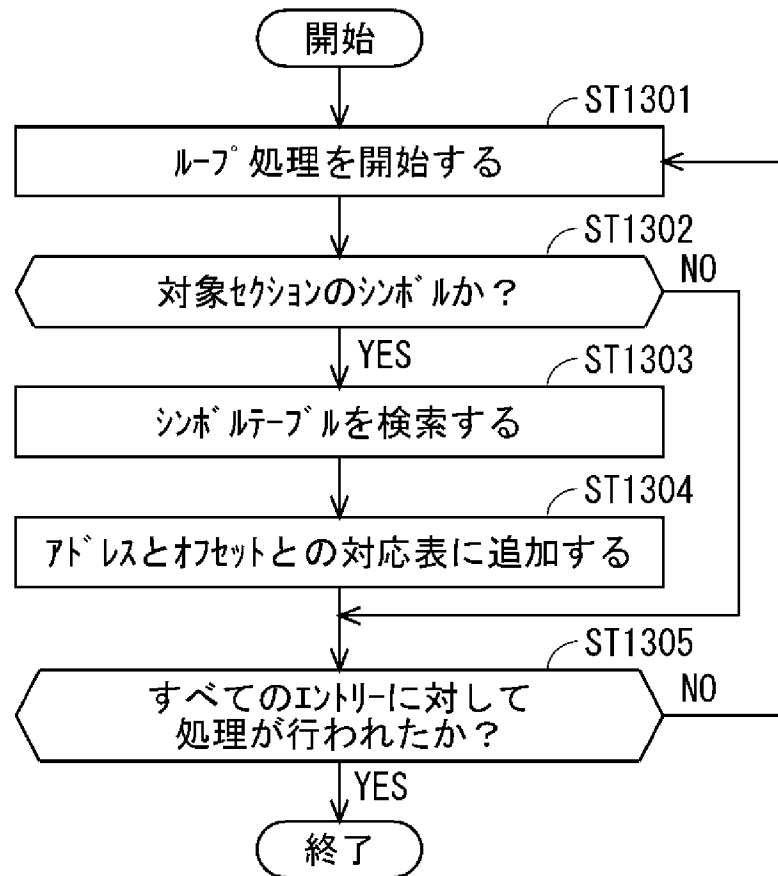
[図11]

オフセット	情報	型	シンボル値	シンボル名
00000004	00000a1c	R_ARM_CALL	00000000	__aeabi_idivmod
00000010	00000028	R_ARM_V4BX		
00000020	00000202	R_ARM_ABS32	00000000	.text
00000024	00000202	R_ARM_ABS32	00000000	.text
00000028	00000202	R_ARM_ABS32	00000000	.text
0000002c	00000202	R_ARM_ABS32	00000000	.text
0000003c	00000c1c	R_ARM_CALL	00000000	__aeabi_idiv
00000044	00000028	R_ARM_V4BX		
0000004c	00000028	R_ARM_V4BX		
00000054	00000028	R_ARM_V4BX		
0000005c	00000028	R_ARM_V4BX		
00000064	00000028	R_ARM_V4BX		

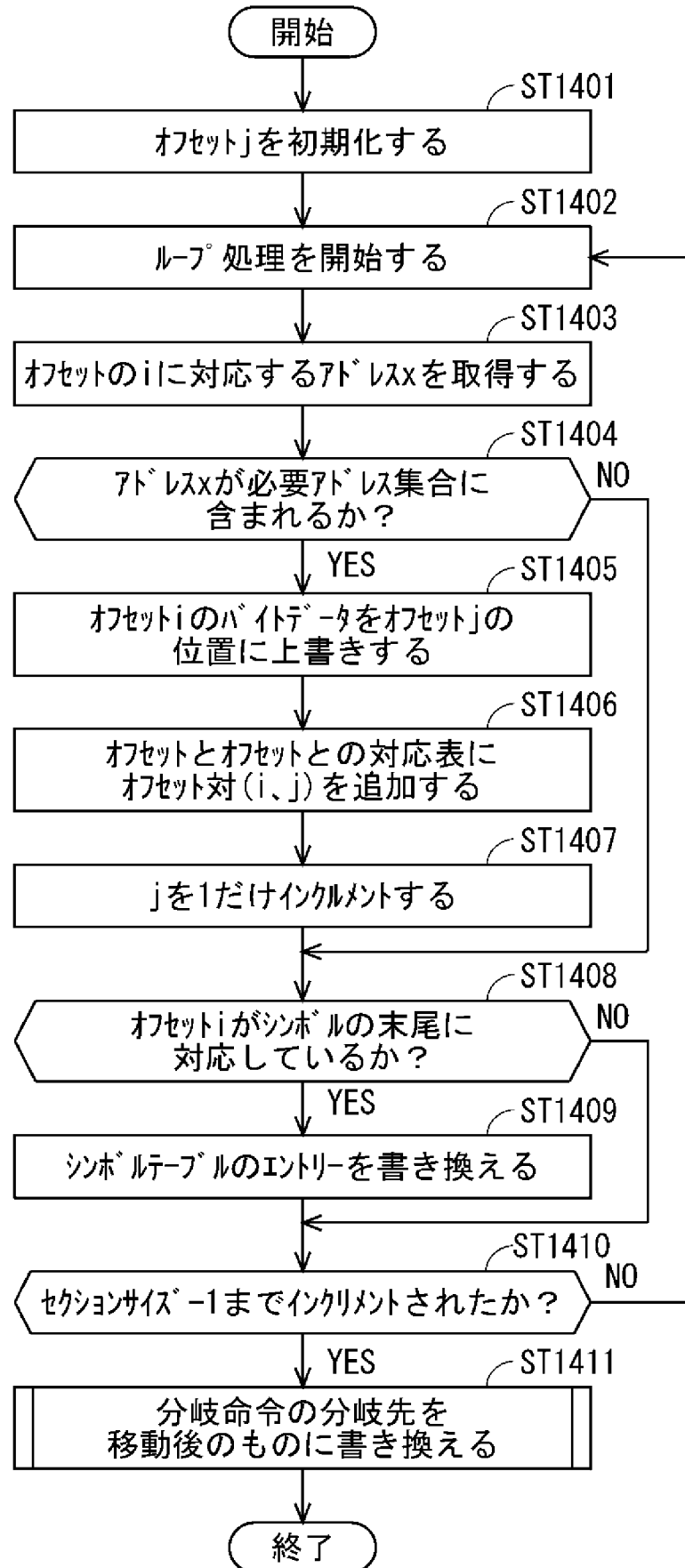
[図12]

オフセット	情報	型	シンボル値	シンボル名
00000004	0000a1c	R_ARM_NONE	00000000	__aeabi_idivmod
00000010	00000028	R_ARM_NONE		
0000000c	00000202	R_ARM_ABS32	00000000	.text
00000024	00000202	R_ARM_NONE	00000000	.text
00000028	00000202	R_ARM_NONE	00000000	.text
00000018	00000202	R_ARM_ABS32	00000000	.text ¹²⁰¹
00000028	0000c1c	R_ARM_CALL	00000000	__aeabi_idiv
00000030	00000028	R_ARM_V4BX		
0000004c	00000028	R_ARM_NONE		
00000054	00000028	R_ARM_NONE		
00000038	00000028	R_ARM_V4BX		
00000064	00000028	R_ARM_NONE		

[図13]



[図14]



[図15]

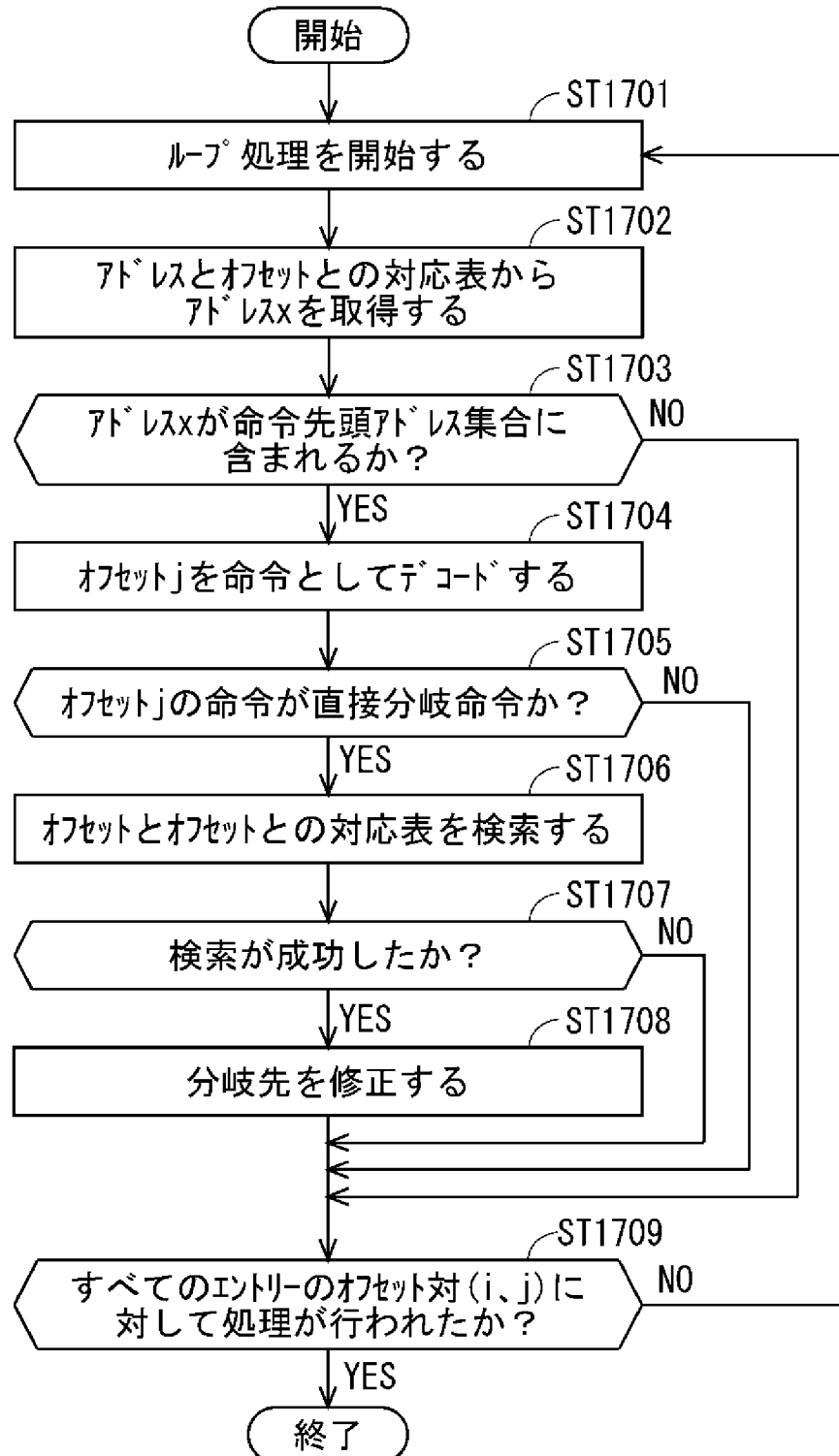
番号:	値	サイズ	タイプ	Bind	Vis	牽引名
0:	000000000	0	NOTYPE	LOCAL	DEFAULT	UND
1:	000000000	0	FILE	LOCAL	DEFAULT	ABS calc.c
2:	000000000	0	SECTION	LOCAL	DEFAULT	1
3:	000000000	0	SECTION	LOCAL	DEFAULT	3
4:	000000000	0	SECTION	LOCAL	DEFAULT	4
5:	000000000	0	NOTYPE	LOCAL	DEFAULT	1 \$a
6:	000000020	0	NOTYPE	LOCAL	DEFAULT	1 \$d
7:	000000030	0	NOTYPE	LOCAL	DEFAULT	1 \$a
8:	000000000	0	SECTION	LOCAL	DEFAULT	5
9:	000000000	0	SECTION	LOCAL	DEFAULT	6
10:	000000000	0	NOTYPE	GLOBAL	DEFAULT	UND __aeabi_idivmod
11:	000000000	20	FUNC	GLOBAL	DEFAULT	1 mod
12:	000000000	0	NOTYPE	GLOBAL	DEFAULT	UND __aeabi_idiv
13:	000000014	84	FUNC	GLOBAL	DEFAULT	1 calc ~1501

[図16]

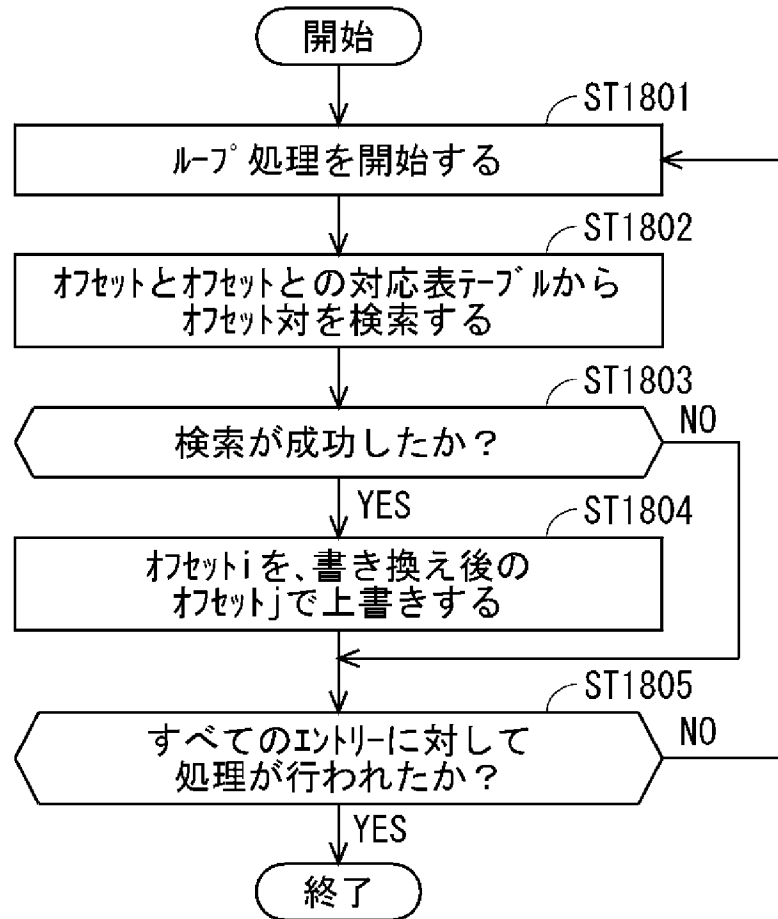
番号:	値	サイズ	タイプ	Bind	Vis	牽引名
0:	000000000	0	NOTYPE	LOCAL	DEFAULT	UND
1:	000000000	0	FILE	LOCAL	DEFAULT	ABS calc.c
2:	000000000	0	SECTION	LOCAL	DEFAULT	1
3:	000000000	0	SECTION	LOCAL	DEFAULT	3
4:	000000000	0	SECTION	LOCAL	DEFAULT	4
5:	000000000	0	NOTYPE	LOCAL	DEFAULT	1 \$a
6:	000000020	0	NOTYPE	LOCAL	DEFAULT	1 \$d
7:	000000030	0	NOTYPE	LOCAL	DEFAULT	1 \$a
8:	000000000	0	SECTION	LOCAL	DEFAULT	5
9:	000000000	0	SECTION	LOCAL	DEFAULT	6
10:	000000000	0	NOTYPE	GLOBAL	DEFAULT	UND __aeabi_idivmod
11:	000000000	20	FUNC	GLOBAL	DEFAULT	1 mod
12:	000000000	0	NOTYPE	GLOBAL	DEFAULT	UND __aeabi_idiv
13:	000000000	60	FUNC	GLOBAL	DEFAULT	1 calc

1601

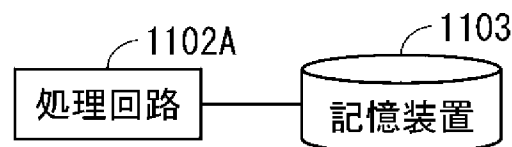
[図17]



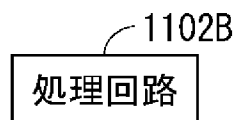
[図18]



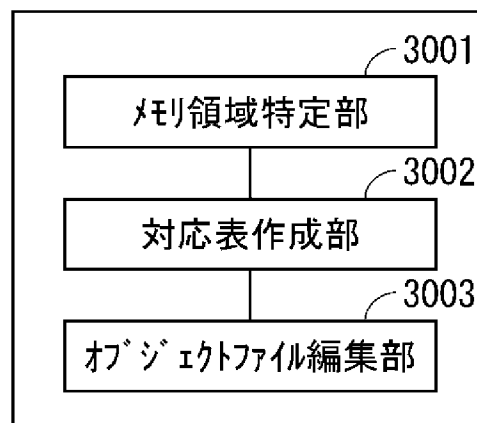
[図19]



[図20]



[図21]



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2020/036585

A. CLASSIFICATION OF SUBJECT MATTER

G06F 8/52 (2018.01) i

FI: G06F8/52

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F8/00-8/77

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Published examined utility model applications of Japan	1922-1996
Published unexamined utility model applications of Japan	1971-2020
Registered utility model specifications of Japan	1996-2020
Published registered utility model applications of Japan	1994-2020

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	PIETREK, Matt, LIBCTINY.LIB で, EXE, DLL ファイルのサイズを小さくする, MSDN Magazine, 18 February 2001, 2001 February no. 11, pp. 93-99, entire text, non-official translation ("Reduce the size of EXE and DLL files with LIBCTINY.LIB")	1-7
A	JP 2012-141887 A (CANON INC.) 26 July 2012 (2012-07-26) entire text, all drawings	1-7
A	US 9904527 B1 (AMAZON TECHNOLOGIES, INC.) 27 February 2018 (2018-02-27) entire text, all drawings	1-7



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"I" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
11 December 2020 (11.12.2020)Date of mailing of the international search report
22 December 2020 (22.12.2020)Name and mailing address of the ISA/
Japan Patent Office
3-4-3, Kasumigaseki, Chiyoda-ku,
Tokyo 100-8915, Japan

Authorized officer

Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/JP2020/036585

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
JP 2012-141887 A	26 Jul. 2012	(Family: none)	
US 9904527 B1	27 Feb. 2018	(Family: none)	

A. 発明の属する分野の分類（国際特許分類（IPC）） G06F 8/52(2018.01)i FI: G06F8/52		
B. 調査を行った分野		
調査を行った最小限資料（国際特許分類（IPC）） G06F8/00-8/77		
最小限資料以外の資料で調査を行った分野に含まれるもの 日本国実用新案公報 1922-1996年 日本国公開実用新案公報 1971-2020年 日本国実用新案登録公報 1996-2020年 日本国登録実用新案公報 1994-2020年		
国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）		
C. 関連すると認められる文献		
引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求項の番号
A	PIETREK, Matt, LIBCTINY.LIBで、EXE, DLLファイルのサイズを小さくする, MSDN Magazine, 2001.02.18, 2001 February No.11, pp.93-99 全文	1-7
A	JP 2012-141887 A (キャノン株式会社) 26.07.2012 (2012-07-26) 全文, 全図	1-7
A	US 9904527 B1 (AMAZON TECHNOLOGIES, INC.) 27.02.2018 (2018-02-27) 全文, 全図	1-7
☐ C欄の続きにも文献が列挙されている。 ☒ パテントファミリーに関する別紙を参照。		
* 引用文献のカテゴリー “A” 特に関連のある文献ではなく、一般的な技術水準を示すもの “E” 国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの “L” 優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す） “O” 口頭による開示、使用、展示等に言及する文献 “P” 国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献 “T” 国際出願日又は優先日後に公表された文献であって出願と抵触するものではなく、発明の原理又は理論の理解のために引用するもの “X” 特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの “Y” 特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの “&” 同一パテントファミリー文献		
国際調査を完了した日 11.12.2020		国際調査報告の発送日 22.12.2020
名称及びあて先 日本国特許庁(ISA/JP) 〒100-8915 日本国 東京都千代田区霞が関三丁目4番3号		権限のある職員（特許庁審査官） 杉浦 孝光 5B 5287 電話番号 03-3581-1101 内線 3545

国際調査報告
パテントファミリーに関する情報

国際出願番号

PCT/JP2020/036585

引用文献			公表日	パテントファミリー文献	公表日
JP	2012-141887	A	26.07.2012	(ファミリーなし)	
US	9904527	B1	27.02.2018	(ファミリーなし)	