(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
2006 年1 月5 日 (05.01.2006)

PCT

(10) 国際公開番号
WO 2006/001365 A1

(51) 国際特許分類[7]: H04N 5/92, G11B 20/10

(21) 国際出願番号: PCT/JP2005/011602

(22) 国際出願日: 2005 年6 月24 日 (24.06.2005)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(30) 優先権データ:
特願2004-190366 2004 年6 月28 日 (28.06.2004) JP

(71) 出願人 (米国を除く全ての指定国について): 松下電器産業株式会社 (MATSUSHITA ELECTRIC INDUSTRIAL CO.,LTD.) [JP/JP]; 〒5718501 大阪府門真市大字門真１００６番地 Osaka (JP).

(72) 発明者; および

(75) 発明者/出願人 (米国についてのみ): 芳賀 智之 (HAGA, Tomoyuki). 庄田 幸恵 (SHODA, Yukie). 佐藤 太一 (SATO, Taichi). 廣田 照人 (HIROTA, Teruto).

(74) 代理人: 中島 司朗, 外(NAKAJIMA, Shiro et al.); 〒5310072 大阪府大阪市北区豊崎三丁目２番１号淀川５番館６Ｆ Osaka (JP).

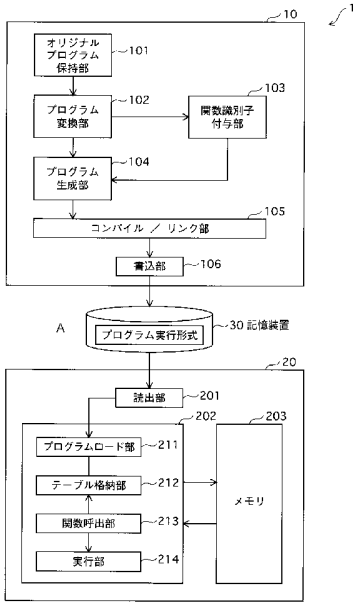
(81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ,

[続葉有]

(54) Title: PROGRAM CREATION DEVICE, PROGRAM TEST DEVICE, PROGRAM EXECUTION DEVICE, INFORMATION PROCESSING SYSTEM

(54) 発明の名称: プログラム生成装置、プログラムテスト装置、プログラム実行装置、及び情報処理システム



101 ORIGINAL PROGRAM HOLDING UNIT
102 PROGRAM TRANSLATION UNIT
103 FUNCTION IDENTIFIER ADDITION UNIT
104 PROGRAM CREATION UNIT
105 COMPILE/LINK UNIT
106 WRITE UNIT
30 STORAGE DEVICE
A PROGRAM EXECUTION SCHEME
201 READ UNIT
211 PROGRAM LOADING UNIT
212 TABLE LOADING UNIT
213 FUNCTION INVOKING UNIT
214 EXECUTION UNIT
203 MEMORY

(57) Abstract: An information processing system comprising a program creation device for creating an enciphered program difficult to analyze from the outside, and a program execution device for executing the program. The program creation device includes acquisition means for acquiring a first program for causing one or more instructions to be executed in a predetermined sequence so that one result may be obtained, creation means for creating a second program on the basis of the first program, and output means for outputting the second program. This second program changes the processing to be executed for the result, in accordance with current information decided at the time of executing the second program, and the obtained result is identical to that obtained with the first program, irrespective of the change in the processing according to the current information.

(57) 要約: 本発明は、外部からの解析が困難な難読化プログラムを生成するプログラム生成装置と、前記プログラムを実行するプログラム実行装置とから成り、前記プログラム生成装置は、ある１の結果を得ることができるように、１以上の命令を所定の順序で実行させる第１プログラムを取得する取得手段と、前記第１プログラムに基づき第２プログラムを生成する生成手段と、前記第２プログラムを出力する出力手段とを備え、前記第２プログラムは、当該第２プログラムの実行時に決定されるカレント情報に応じて、結果を得るために実行させる処理が変化し、かつ、得られる前記結果が、前記カレント情報に応じた処理の変化に関わらず、前記第１プログラムにより得られる結果と同一となるプログラムであることを特徴とする。

WO 2006/001365 A1

BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

添付公開書類:
— 国際調査報告書

２文字コード及び他の略語については、定期発行される各*PCT*ガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

明 細 書

プログラム生成装置、プログラムテスト装置、プログラム実行装置、及び情報処理システム

技術分野

[0001] 本発明は、コンピュータプログラムの保護に関し、特に、外部からの解析が困難なコンピュータプログラムを生成し、テストし、実行する技術に関する。

背景技術

[0002] コンピュータプログラムの不正利用を阻止する方法として、特許文献1では、実行時にアクセスされるアドレスの順序を監視し、正当なユーザのみ知り得る正しい順序でアクセスされていない場合に、プログラムの実行を阻止する技術が開示されている。これにより、コンピュータプログラムの無許可アクセスを検出、及び阻止することができる。

特許文献1:特開平2-45829号公報

非特許文献1:湯淺太一、安村通晃、中田登志之編『はじめての並列プログラミング』共立出版 1999

発明の開示

発明が解決しようとする課題

[0003] しかしながら、上記の技術では、毎回同じ順序でプログラムが実行されるため、不正なユーザがプログラムにブレークポイントを設定して、プログラムを解析することにより、プログラムを不正に利用する可能性がある。

そこで、本発明は上記の問題点に鑑みなされたものであって、外部からの解析が困難なコンピュータプログラムを生成するプログラム生成装置、及びこれを実行するプログラム実行装置を提供することを目的とする。

課題を解決するための手段

[0004] 上記の目的を達成するために、本発明は、ある1の結果を得ることができるように、1以上の命令を所定の順序で実行させる第1プログラムを取得する取得手段と、前記第1プログラムに基づき第2プログラムを生成する生成手段と、前記第2プログラムを

出力する出力手段とを備え、前記第2プログラムは、当該第2プログラムの実行時に決定されるカレント情報に応じて、結果を得るために実行させる処理が変化し、かつ、得られる前記結果が、前記カレント情報に応じた処理の変化に関わらず、前記第1プログラムにより得られる結果と同一となるプログラムであることを特徴とする。

[0005] 更に、本発明は、上記のプログラム生成装置により生成された前記第2プログラムを実行するプログラム実行装置であって、前記第2プログラムを取得する取得手段と、実行時に、前記カレント情報を指定する情報指定手段と、前記第2プログラムを実行する実行手段とを備えることを特徴とする。

発明の効果

[0006] 本発明のプログラム生成装置によると、実行時に、実行させる処理が変化する第2プログラムを生成するので、実行する毎に、同一の結果を得ることが出来、尚且つ実行順序や実行経路の異なる処理を実行させることができる。実行する毎に実行順序や実行経路が異なるので、不正なユーザがプログラムを解析することを困難にし、プログラムの不正利用を防止することができる。

[0007] また、本発明は、コンパイラやOSに依存しないので、不正解析への手掛かりを与えない。

ここで、前記生成手段は、前記第1プログラムに含まれる命令のうち、相互に依存関係にない命令である複数の並列命令を検出する検出部を備え、前記第1プログラムが実行させる処理のうち、前記複数の並列命令を前記所定の順序で実行させる処理に替えて、前記カレント情報に応じて変化する順序で実行させる前記第2プログラムを生成するように構成してもよい。

[0008] この構成によると、プログラム生成装置は、第1プログラムの並列性を検出することにより、出力結果が同一であるが実行順序が異なる処理を実行させる第2プログラムを生成することができる。

ここで、前記生成手段は、前記第1プログラムに含まれる1以上の命令であるオリジナル命令と異なる処理を行い、かつ、前記オリジナル命令と同一の結果を出力する1以上の命令からなる恒等命令を1以上生成する恒等命令生成部を備え、前記第1プログラムが実行させる処理のうち、前記オリジナル命令を実行させる処理に替えて、

前記カレント情報に応じて、前記オリジナル命令、又は生成された前記恒等命令のうちから1を選択させ、実行させる前記第2プログラムを生成するように構成してもよい。

[0009] この構成によると、プログラム生成装置は、第1プログラムに含まれる命令の恒等命令を生成することにより、出力結果が同一であるが実行経路が異なる処理を実行させる第2プログラムを生成することができる。

ここで、前記生成手段は、前記結果に影響を与えない命令であるダミー命令を生成するダミー命令生成部を備え、前記第1プログラムと同一の結果を得るために実行させる処理に加えて、前記カレント情報に応じて変化するタイミングで、前記ダミー命令を実行させる前記第2プログラムを生成するように構成してもよい。

[0010] この構成によると、プログラム生成装置は、出力結果に影響を与えないダミー命令を生成することにより、出力結果が同一であるが実行経路が異なる処理を実行させる第2プログラムを生成することができる。

ここで、前記生成手段は、前記第1プログラムに含まれる命令のうち、相互に依存関係にない命令である複数の並列命令を検出する検出部と、前記第1プログラムに含まれる1以上の命令であるオリジナル命令と異なる処理を行い、かつ、前記オリジナル命令と同一の結果を出力する1以上の命令からなる恒等命令を1以上生成する恒等命令生成部と、前記結果に影響を与えない命令であるダミー命令を生成するダミー命令生成部とを備え、前記第1プログラムが実行させる処理のうち、前記複数の並列命令を前記所定の順序で実行させる処理に替えて、前記カレント情報に応じて変化する順序で実行させ、前記第1プログラムが実行させる処理のうち、前記オリジナル命令を実行させる処理に替えて、前記カレント情報に応じて、前記オリジナル命令又は生成された前記恒等命令のうちから1を選択させ、実行させ、当該カレント情報に応じて変化するタイミングで、前記ダミー命令を実行させる前記第2プログラムを生成するように構成してもよい。

[0011] この構成によると、プログラム生成装置は、並列性を検出し、恒等命令を生成し、ダミー命令を生成することにより、出力結果が同一であるが実行順序及び実行経路の異なる処理を実行させる第2プログラムを生成することができる。

ここで、前記生成手段は、更に、前記第1プログラムに含まれる命令、恒等命令生

成部により生成された恒等命令、及びダミー命令生成部により生成されたダミー命令のそれぞれに、各命令を一意に識別する識別子を付与する識別子付与手段を備え、前記カレント情報に応じて生成される乱数列の各要素と一致する識別子により識別される命令を呼び出す前記第2プログラムを生成するように構成してもよい。

[0012] この構成によると、プログラム生成装置は、各命令に識別子を付与することにより、実行時に1の乱数列を生成させることで、実行すべき命令を呼び出す第2プログラムを生成することができる。

ここで、前記生成手段は、前記カレント情報に応じて前記乱数列を生成する乱数列生成アルゴリズムを含む前記第2プログラムを生成するように構成してもよい。

[0013] この構成によると、プログラム生成装置は、乱数列生成アルゴリズムを含む第2プログラムを生成することにより、実行時に生成される乱数の安全性を保障することができる。

ここで、前記生成手段は、前記カレント情報に応じて乱数列を生成する複数の乱数列生成アルゴリズムを含み、実行時に前記複数の乱数列生成アルゴリズムから1の乱数列生成アルゴリズムを選択し、選択した乱数列生成アルゴリズムに従い生成される乱数列の各要素と一致する識別子により識別される命令を呼び出す前記第2プログラムを生成するように構成してもよい。

[0014] この構成によると、プログラム生成装置は、複数個の乱数列生成アルゴリズムから、実行時に1のアルゴリズムが選択されるように第2プログラムを生成することにより、よりランダム性の高い処理を実行させる第2プログラムを生成することができる。

ここで、前記生成手段は、更に、前記第1プログラムに含まれる命令、恒等命令生成部により生成された恒等命令、及びダミー命令生成部により生成されたダミー命令のそれぞれを関数化し、複数の関数からなる関数データを生成する関数データ生成部と、前記関数データ生成部により生成されたそれぞれの関数に、各関数を一意に識別する識別子を付与する識別子付与部と、前記カレント情報に応じて乱数列を生成する乱数列生成アルゴリズムを含み、生成される乱数列の各要素と一致する識別子により識別される関数のうち、前記結果と同一の結果を得るための条件に従う関数の呼び出し、及び実行を指示する呼出プログラムを生成する呼出プログラム生成部と

を備え、前記関数データ、及び前記呼出プログラムを含む前記第2プログラムを生成するように構成してもよい。

[0015] この構成によると、プログラム生成装置は、各命令を関数化し、各関数に識別子を付与することにより、実行時に1の乱数を生成させることで、実行させる関数を呼び出す第2プログラムを生成することができる。

ここで、前記複数の関数は、それぞれ、並列グループ、シーケンシャルグループ、恒等グループ、及びダミーグループの何れか1以上のグループに属し、前記呼出プログラム生成部は、各関数が属するグループを判断することにより、前記条件に従う順序で、各関数を呼び出すことを指示する前記呼出プログラムを生成するように構成してもよい。

[0016] この構成によると、プログラム生成装置は、関数属性の概念を導入することにより、実行時には、関数属性を判断することにより各関数の実行順序や並列関係を判断し、実行させる関数を呼び出す第2プログラムを生成することができる。

ここで、前記プログラム生成装置は、更に、前記カレント情報の候補となる候補情報を複数個生成する候補情報生成手段を備え、前記出力手段は、前記第2プログラムと共に、適切な複数個の候補情報を出力するように構成してもよい。

[0017] この構成によると、プログラム生成装置は、予めカレント情報の候補となる候補情報を複数個生成し、実行時に決定されるカレント情報を限定することにより、実行させる処理のランダム性を確保しながら、実行されうる複数の処理について、正常に動作するか否かをテストすることも現実的に可能となる。

ここで、前記プログラム生成装置は、更に、生成された前記複数個の候補情報のそれぞれについて、各候補情報に応じて行う動作が、正しい動作であるか否か判断するテスト手段を備え、前記出力手段は、前記テスト手段により正しい動作を行うと判断された候補情報のみを抽出し、出力するように構成してもよい。

[0018] この構成によると、プログラム生成装置は、正常に動作すると保障された処理に対応する候補情報のみを出力するので、実行時においては、正常に動作する処理のみを実行させることができる。

ここで、前記テスト手段は、前記第1プログラムを実行して得られる第1値を取得する

第1取得部と、1の候補情報に応じた動作を行い得られる第2値を取得する第2取得部と、前記第1値と前記第2値とを比較し、両者が一致する場合に、前記1の候補情報に応じて行う動作が、正しい動作であると判断する比較判断部とを備えるように構成してもよい。

[0019] この構成によると、プログラム生成装置は、第1プログラムの実行結果を用いて、第2プログラムが実行させる処理が正常に動作するか否か、正確に判断することができる。

また、本発明は、コンピュータプログラムであって、実行時に決定されるカレント情報に応じて異なる動作を行い、かつ、前記動作のそれぞれによって得られる結果は、前記カレント情報に関わらず一定であることを特徴とする。

[0020] この構成によると、このコンピュータプログラムは、実行する毎に、同一の結果を得ることができ、尚且つ動作が異なる処理を実行させることができる。

また、本発明は、前記プログラム生成装置により生成された前記第2プログラムを実行するプログラム実行装置であって、前記第2プログラムを取得する取得手段と、実行時に、前記カレント情報を指定する情報指定手段と、前記第2プログラムを実行する実行手段とを備えることを特徴とする。

[0021] この構成によると、プログラム実行装置は、実行時に指定するカレント情報により異なる動作を行うので、不正な解析を困難にし、不正利用を防止することができる。

ここで、前記情報指定手段は、乱数を生成し、生成した乱数を前記カレント情報とするように構成してもよい。

この構成によると、プログラム実行装置は、実行時に乱数を発生させてカレント情報を指定するため、実行する処理をランダムに決定することができる。

[0022] ここで、前記取得手段は、前記カレント情報の候補となる複数の候補情報を取得し、前記情報指定手段は、前記複数の候補情報から1の候補情報を選択するように構成してもよい。

この構成によると、プログラム実行装置は、第2プログラムと共に取得する複数の候補情報から1の候補情報を選択すればよいので、乱数を発生させるなどの方法によりカレント情報を生成する必要がない。

[0023] ここで、前記情報指定手段は、乱数を生成し、生成した乱数を用いて前記複数の候補情報から1の候補情報を選択するように構成してもよい。

この構成によると、プログラム実行装置は、複数の候補情報から1の候補情報を選択する際に、乱数を発生させ、乱数を用いてランダムに1の候補情報を選択することができる。

[0024] ここで、前記第2プログラムを複数回実行する前記プログラム実行装置において、前記情報指定手段は、選択済みの候補情報を記憶する記憶部と、前記記憶部を参照することにより、未選択の候補情報を選択するよう制御する制御部とを備えるように構成してもよい。

この構成によると、プログラム実行装置は、複数の候補情報から1の候補情報を万偏無く選択することができ、これにより、ユーザによる不正な解析を困難にし、不正利用を防止することができる。

[0025] ここで、前記プログラム実行装置は、更に、前記第1プログラムを実行して得られる第1値を取得する第1値取得手段と、前記実行手段により実行された第2プログラムの実行結果である第2値と、前記第1値とを比較し、両者が一致するか否か判断する比較判断手段と、前記比較判断手段により、前記第1値と前記第2値とが一致すると判断された場合に、前記情報指定手段により指定された前記候補情報を記録する記録手段とを備えるように構成してもよい。

[0026] この構成によると、プログラム実行装置は、取得した複数の候補情報を、第1プログラムの実行結果に基づき正常動作するか否か正しく判断することができる。

また、プログラム実行装置は、取得した複数の候補情報について比較判断し、正常に動作した候補情報を記録しておくことにより、次に第2プログラムを実行するときから、記録している候補情報から1を選択し、選択した候補情報をカレント情報とすることで、正常動作が保障された処理を実行することができる。

[0027] ここで、前記プログラム実行装置は、更に、前記情報指定手段、前記実行手段、前記比較判断手段及び前記記録手段による処理を複数回行い、前記記録手段により記録された複数個の候補情報を含むリストを生成し、生成したリストを出力するように構成してもよい。

この構成によると、プログラム実行装置は、正常動作する候補情報を含むリストを他の装置へ出力することにより、他の装置に、正常動作が保障された第2プログラムを実行させることができる。

図面の簡単な説明

[0028] [図1]第1の実施形態である情報処理システム1の構成を示すシステム構成図であり、プログラム生成装置10及びプログラム実行装置20の機能的な構成を示す機能ブロック図である。

[図2]情報処理システム1全体の動作を示すフローチャートである。

[図3]オリジナルプログラムの具体例であるオリジナルプログラム110を示す図である。

[図4](a)オリジナルプログラム110に含まれる各処理の、依存関係を説明する図である。(b)オリジナルプログラム110の関数構造を示す図である。

[図5](a)プログラム変換部102が生成する恒等変換を説明する図である。(b)恒等変換が生成された後のオリジナルプログラム110の関数構造を示す図である。

[図6](a)プログラム変換部102が生成するダミー処理を説明する図である。(b)ダミー処理が生成された後のオリジナルプログラム110の関数構造を示す図である。

[図7]オリジナルプログラム110の各処理の変数を構造体に、各処理を関数に変換して生成した関数データ120を示す図である。

[図8]関数識別子テーブル130を示す図である。

[図9]プログラム生成部104が生成する関数呼出プログラム140を示す図である。

[図10]プログラム生成部104が生成する関数呼出順決定プログラム150を示す図である。

[図11]プログラム生成装置によるプログラム実行形式の生成処理の動作を示すフローチャートである。

[図12]関数識別子テーブル220を示す図である。

[図13]プログラム実行装置20によるプログラム実行処理の動作を示すフローチャートであり、図14に続く。

[図14]プログラム実行装置20によるプログラム実行処理の動作を示すフローチャート

であり、図13から続く。

[図15]線形合同法による乱数列の生成を説明する図である。

[図16]初期値bの値に応じた関数の実行順序を説明する図である。

[図17]関数構造250を示し、関数グループに基づき各関数の呼出順序を決定する方法について説明する図である。

[図18]第2の実施形態である情報処理システム2のシステム構成を示す図であり、プログラム生成装置10a及びプログラム実行装置20aの内部構成を機能的に示す機能ブロック図である。

[図19]情報処理システム2全体の動作を示すフローチャートである。

[図20]実行経路パターンリスト300のデータ構成を示す図である。

[図21]プログラム生成装置10aにおける実行経路パターンリスト生成処理の動作を示すフローチャートである。

[図22]プログラム実行装置20aにおけるプログラム実行処理の動作を示すフローチャートであり、図23へ続く。

[図23]プログラム実行装置20aにおけるプログラム実行処理の動作を示すフローチャートであり、図22から続く。

[図24]第3の実施形態である情報処理システム3のシステム構成を示す図であり、プログラムテスト装置40b及びプログラム実行装置20bの内部構成を機能的に示す機能ブロック図である。

[図25]情報処理システム3全体の動作を示すフローチャートである。

[図26]プログラムテスト装置40bにおける実行経路パターンリスト生成処理の動作を示すフローチャートである。

[図27](a)変形例として、プログラム生成装置が生成する関数呼出プログラム401を示す図である。(b)変形例として、プログラム生成装置が生成する関数呼出順決定プログラム402を示す図である。

[図28]変形例として、プログラム実行装置がセルオートマトンを用い関数識別子を決定する動作を示すフローチャートである。

[図29]セルオートマトンで出力される値について、具体例を用いて説明する図である

。

[図30]セルオートマトンで出力される値に応じた関数の実行順序を説明する図である

。

符号の説明

[0029] 1 情報処理システム

2 情報処理システム

3 情報処理システム

10 プログラム生成装置

10a プログラム生成装置

10b プログラム生成装置

20 プログラム実行装置

20a プログラム実行装置

20b プログラム実行装置

30 記憶装置

30a 記憶装置

30b 記憶装置

40a プログラムテスト装置

40b プログラムテスト装置

50b 記憶装置

101 オリジナルプログラム保持部

101a オリジナルプログラム保持部

102 プログラム変換部

102a プログラム変換部

103 関数識別子付与部

103a 関数識別子付与部

104 プログラム生成部

104a プログラム生成部

105 コンパイル／リンク部

105a コンパイル／リンク部

106 書込部

106a 書込部

107a 実行経路パターン生成部

201 読出部

201a 読出部

201b 読出部

202 プログラム実行部

202a プログラム実行部

202b プログラム実行部

203 メモリ

203a メモリ

203b メモリ

211 プログラムロード部

211a プログラムロード部

211b プログラムロード部

212 テーブル格納部

212a テーブル格納部

212b テーブル格納部

213 関数呼出部

213a 関数呼出部

213b 関数呼出部

214 実行部

214a 実行部

214b 実行部

401a 読出部

401b 読出部

402b プログラム実行部

403b メモリ

404b 実行経路パターン生成部

405b 書込部

411b プログラムロード部

412b テーブル格納部

413b 関数呼出部

414b 実行部

発明を実施するための最良の形態

[0030] 以下では、本発明の実施するための最良の形態について、図面を参照し詳細に説明する。

＜第1の実施形態＞

本発明に係る第1の実施形態として、情報処理システム1について、図面を参照して説明する。

[0031] 1. 全体の構成

図1は、情報処理システム1の構成を示す図である。情報処理システム1は、プログラム生成装置10、プログラム実行装置20、及び記憶装置30から構成される。

プログラム生成装置10及びプログラム実行装置20は、共にコンピュータシステムであり、記憶装置30は、例えば光ディスク装置である。

[0032] プログラム生成装置10は、オリジナルプログラムから実行形式のプログラムファイル（以下「プログラム実行形式」と呼称する）を生成し、生成したプログラム実行形式を記憶装置30に書き込む。プログラム実行装置20は、記憶装置30に記憶されているプログラム実行形式を読み出し実行する。

記憶装置30は、プログラム生成装置10が書き込むプログラム実行形式を記憶する。

[0033] 2. 全体の動作

図2は、情報処理システム1全体の動作を示すフローチャートである。

先ず、プログラム生成装置10が、オリジナルプログラムからプログラム実行形式を生成し（ステップS101）、生成したプログラム実行形式を記憶装置30に書き込む（ステ

ップS102)。記憶装置30は、プログラム実行形式を記憶する(ステップS103)。

[0034] プログラム実行装置20は、記憶装置30からプログラム実行形式を読み出す(ステップS104)。プログラム実行装置20は、初期値を設定し(ステップS105)、設定した初期値に応じて決定される順序で、プログラム実行形式の各処理を実行する(ステップS106)。

3. プログラム生成装置10の構成

ここでは、プログラム生成装置10の構成について詳細に説明する。

[0035] プログラム生成装置10は、図1に示す様に、オリジナルプログラム保持部101、プログラム変換部102、関数識別子付与部103、プログラム生成部104、コンパイル/リンク部105、及び書込部106から構成される。

プログラム生成装置10は、具体的には、マイクロプロセッサ、ROM、RAM、ハードディスクユニット等から構成されるコンピュータシステムである。ハードディスクユニット及びRAMには、コンピュータプログラムが記憶されており、マイクロプロセッサがコンピュータプログラムに従い動作することにより、プログラム生成装置10は、その機能を達成する。

[0036] (1)オリジナルプログラム保持部101

オリジナルプログラム保持部101は、図3に示すオリジナルプログラム110を保持している。同図に示す様に、オリジナルプログラム110は、命令701、702、703、704及び705の5つの命令から成り、命令701を(処理1)、命令702を(処理2)、命令703を(処理3)、命令704を(処理4)、命令705を(処理5)と呼称する。オリジナルプログラム110は、(処理1)から(処理4)まで、逐次計算し、(処理5)によりdの値を出力することを示している。

[0037] なお、オリジナルプログラム保持部101は、どのようにオリジナルプログラム110を取得するのか限定されない。ユーザにより入力されてもよいし、記録媒体などに格納されたオリジナルプログラム110を読み込んでもよい。また、図3のオリジナルプログラム110は、一例であって、オリジナルプログラム保持部101が保持するプログラムには、様々なプログラムが想定される。

[0038] (2)プログラム変換部102

プログラム変換部102は、並列性の検出、恒等変換の生成、ダミー処理の生成、及び関数化の処理を行う。

(並列性の検出)

オリジナルプログラム110は、図3に示したように、(処理1)から(処理5)までをこの順序で実行する逐次プログラムである。逐次プログラムには、計算結果を得る上で、本質的でない記述上の順序関係が存在する。記述上の順序関係にある処理は、実行順序を入れ替えても逐次実行と同じ計算結果を得ることができる。また、これらの処理を複数のプロセッサに分割して並列実行することもできる。逐次プログラムから、記述上の順序関係にある処理を検出することを、並列性の検出という。

[0039] プログラム変換部102は、オリジナルプログラム110から、以下に示す様に並列性を検出する。

プログラム変換部102は、並列性を検出するために、オリジナルプログラム110の各処理を先頭から解析して、データ依存関係の有無を調べる。

図4(a)に示す様に、命令1001、即ち(処理2)は、(処理1)で代入された変数aの値を参照しているので、(処理1)にフロー依存している。ここで、フロー依存とは、例えば、(処理1)を実行したあとでなければ、(処理2)を実行できない関係をいう。命令1002、即ち(処理3)は、(処理1)で代入された変数aの値を参照しているので、(処理1)にフロー依存している。命令1003、即ち(処理4)は、(処理2)で代入された変数bの値、及び(処理3)で代入された変数cの値を参照しているので、(処理2)及び(処理3)にフロー依存している。

[0040] オリジナルプログラム110に含まれる各処理の内、データ依存関係が無いのは、(処理2)と(処理3)である。従って、オリジナルプログラム110で並列性が検出されるのは、(処理2)及び(処理3)である。なお、並列性の検出は、公知技術である並列化コンパイラで実現可能である。

図4(b)は、オリジナルプログラム110の記述上の構造ではなく、本質的な関数構造を示す図である。オリジナルプログラム110は、(処理1)が実行された後、(処理2)又は(処理3)が実行される。(処理2)及び(処理3)は並列性があるので、実行順序は問わないが、(処理2)及び(処理3)が共に実行された後、(処理4)が実行される。

[0041] プログラム変換部102は、並列性を検出すると、各処理をグループに分類する。ここでは、処理1をグループ1、処理2をグループ2、処理3をグループ3、及び処理4をグループ4とする。

(恒等変換の生成)

プログラム変換部102は、グループ1からグループ4までの内、1以上のグループに於いて、恒等変換を生成する。ここで恒等変換とは、計算結果が同じになるような異なる処理を示す。具体的には、プログラム変換部102は、使用する演算子を変えたり、定数、変数を追加したりして、恒等変換を生成する。

[0042] ここでは、具体例として、プログラム変換部102は、図5(a)に示す様に、グループ3に於いて(処理3)の恒等変換である(処理3´)を生成する。(処理3´)は、aを左へ2ビットシフトする処理であり、(処理3)の a＊4と計算結果は同値である。

図5(b)は、上記の様に(処理3´)を生成した後のオリジナルプログラム110の関数構造を示す図である。同図に示す様に、(処理3´)を生成した後のオリジナルプログラム110は、(処理1)が実行された後、(処理2)又はグループ3に属する何れかの処理が実行される。(処理2)、及びグループ3に属する何れかの処理が実行された後、(処理4)が実行される。ここで、グループ3に属する(処理3)及び(処理3´)は、何れか一方のみを実行すればよい。また、(処理2)、及びグループ3に属する何れかの処理は、実行の順序は問わない。

[0043] (ダミー処理の生成)

プログラム変換部102は、グループ1からグループ4までの内、1以上のグループに於いてダミー処理を生成する。ここでダミー処理とは、オリジナルプログラム110を複雑化するために、オリジナルプログラム110に挿入される処理であって、オリジナルプログラム110から出力される結果に影響を与えない処理を示す。

[0044] ここでは、具体例として、プログラム変換部102は、図6(a)に示す様に、グループ2に於いて、(処理2)のダミー処理として(処理2´)を生成する。(処理2´)のd＝3は、(処理2)のb＝a＋2とは無関係の処理である。なお、ダミー処理の生成は、オリジナルプログラム110全体としての結果に影響を与えない処理であるため、必ずしもグループ単位で生成される必要はない。

[0045] 図6(b)は、上記の様に(処理2´)を生成した後のオリジナルプログラム110の関数構造を示す図である。同図に示す様に、(処理2´)を生成した後のオリジナルプログラム110は、(処理1)が実行された後、グループ2から(処理2)を含む1以上の処理、又は、グループ3に属する(処理3)又は(処理3´)の内、何れか一方の処理を実行する。(処理2)、及び(処理3)又は(処理3´)を実行した後、(処理4)を実行する。ここで、ダミー処理である(処理2´)は、実行してもよいし、実行しなくてもよい。また、(処理2)、(処理2´)、及び(処理3)又は(処理3´)は、実行の順序は問わない。

[0046] (関数化)

プログラム変換部102は、(処理1)、(処理2)、(処理2´)、(処理3)、(処理3´)及び(処理4)の各処理を、それぞれ関数化して関数データ120を生成する。

図7は、関数データ120を示す図である。ここでは、具体例として、プログラム変換部102は、引数の型を統一するために、使用する変数を、「struct val_t」という一つの構造体1011にし、(処理1)を、「funcA」、(処理2)を、「funcB」、(処理3´)を、「funcC1」、(処理3)を、「funcC」、(処理4)を、「funcD」、(処理2´)を、「funcDummy」という関数に変換し、データ1012を生成する。

[0047] プログラム変換部102は、生成した関数データ120と、関数構造とをプログラム生成部104へ渡す。関数構造は、各関数の処理順序を決定するために用いられるデータであって、例えば図17に示す関数構造250がデータ化されて生成されたものである。ここでは、プログラム生成装置10が関数構造をどの様なデータで持つのか限定しない。

(3)関数識別子付与部103

関数識別子付与部103は、プログラム変換部102が生成した関数データ120に含まれる各関数に、各関数をそれぞれ一意に識別する関数識別子を生成する。関数識別子付与部103は、生成した関数識別子を各関数の関数名に対応付け、関数識別子テーブル130を生成する。なお、関数識別子はランダムな値でよい。

[0048] 図8は、関数識別子テーブル130を示す図である。図8に依ると、funcAの関数識別子は「0」、funcBの関数識別子は「1」、funcCの関数識別子は「5」、funcC1の関数識別子は「2」、funcDの関数識別子は「4」、funcDummyの関数識別子は「3」である。

関数識別子付与部103は、生成した関数識別子テーブル130を、プログラム生成部104へ渡す。

[0049] (4)プログラム生成部104

プログラム生成部104は、図9に示す関数呼出プログラム140、及び図10に示す関数呼出順決定プログラム150を生成する。

(関数呼出プログラム140の生成)

プログラム生成部104は、予め関数呼出プログラム生成情報を記憶している。関数呼出プログラム生成情報は、図9に示す関数呼出プログラム140の内、main( )、143の「val->s=rand( );」、及び、144の「while(!FuncSelector(&val));」が記述されたデータである。143の「val->s=rand( );」は、sの値を、実行時に生成される乱数の値に設定することを示し、144の「while(!FuncSelector(&val));」は、FuncSelectorが1を返すまで、val->nを更新して、ループさせることを示す。

[0050] なお、この関数呼出プログラム生成情報は、一例であって、この形に限定する必要はない。以下、関数呼出プログラム140の生成について説明する。

プログラム生成部104は、プログラム変換部102から受け取る関数構造から、初めに実行される関数がfuncAであり、funcAを識別する関数識別子が、関数識別子付与部103から受け取る関数識別子テーブル130より、「0」であることを判断し、関数呼出プログラム生成情報に、142の「val-> n=0;」を書き込む。更に、プログラム生成部104は、関数構造から、最後に実行される関数がfuncDであることを判断し、関数呼出プログラム生成情報に、145の「printf("%d¥n",a->d);」を書き込み、メイン関数141を生成する。ここで、145は、while文によるループが終了すると、dの値を出力することを示す。

[0051] 更に、プログラム生成部104は、関数呼出プログラム生成情報に関数データ120を書き込み、図9に示す関数呼出プログラム140を生成する。

(関数呼出順決定プログラム150の生成)

プログラム生成部104は、予め関数呼出順決定プログラム生成情報を記憶している。関数呼出順決定プログラム生成情報は、図10に示す関数呼出順決定プログラム150の内、「FuncSelector( )」、「int x」、152の「x=(a->n*13+a->s)%8;」、「a->n=x」及び

153内の「switch(x)」が記述されたデータである。なお、この関数呼出順決定プログラム生成情報は、一例であって、この形に限定する必要はない。

[0052] ここで、152は、以下の(式1)を示している。

$$X_{n+1} = (X_n * a + b) \bmod m \quad a=13、m=8 \quad \cdots \quad (式1)$$

(式1)は、線形合同法と呼ばれる擬似乱数生成のアルゴリズムであって、適当なX_0を与えることにより、0以上m未満の乱数を要素とする乱数列$\{X_0、X_1、X_2、X_3、\cdots\}$を生成する漸化式である、更に、(式1)は、定数項bを変化させることにより、異なる乱数列を生成する漸化式である。本明細書においては、定数項bを「初期値b」と呼称する。以下、関数呼出順決定プログラム150の生成について説明する。

[0053] プログラム生成部104は、「FuncSelector( )」に関数名を書き込み「FuncSelector(struct val_t*a)」とする。

プログラム生成部104は、関数構造から各関数の実行順序を判断し、funcAが処理されたか否かを記憶する変数「f_a」、funcBが処理されたか否かを記憶する変数「f_b」、及び、funcC又はfuncC1が、処理されたか否かを記憶する変数「f_c」を関数呼出順決定プログラム生成情報に書き込む。funcCとfuncC1とは、恒等変換であり、何れか一方が処理されればよいので、同一の変数「f_c」を用いる。各変数は、「0」又は「1」の値を取り、「0」は、各関数が未処理であることを示し、「1」は、各関数が処理済であることを示す。具体的に、ここでは各変数の初期状態として、151に示す様に「static int f_a=f_b=f_c=0;」、即ち、何れの関数も未処理であることを示す変数を書き込む。

[0054] また、プログラム生成部104は、関数識別子テーブル130、及び関数構造から、153のswitch文を作る。

プログラム生成部104は、関数識別子テーブル130から関数識別子が「0」である関数がfuncAであることを判断し、153のswitch文に、

case0:if(f_a==0){f_a=1;funcA(a);return(0);}を書き込む。

[0055] case0は、152で生成されるxの値が「0」のとき実行される処理を示す。x=0のとき、151で記憶している変数f_aが「0」の場合、即ちfuncAが未処理の場合に、funcAを処理し、変数f_aを「1」に替えて、関数呼出プログラム140にリターン値「0」を返す。

次に、プログラム生成部は、関数識別子テーブル130から関数識別子が「1」である

関数がfuncBであることを判断し、153のswitch文に、

case1:if(f_b==0){f_b=1;funcB(a);return(0);}を書き込む。

[0056] case1は、152で生成されるxの値が「1」のとき実行される処理を示す。x＝1のとき、151で記憶している変数f_bが「0」の場合、即ちfuncBが未処理の場合に、funcBを処理し、変数f_bを「1」に替えて、関数呼出プログラム140にリターン値「0」を返す。

次に、プログラム生成部は、関数識別子テーブル130から関数識別子が「2」である関数がfuncC1であることを判断し、153のswitch文に、

case2:if(f_c==0){f_c=1;funcC1(a);return(0);}を書き込む。

[0057] case2は、152で生成されるxの値が「2」のとき実行される処理を示す。x＝2のとき、151で記憶している変数f_cが「0」の場合、即ちfuncC及びfuncC1が未処理の場合に、funcC1を処理し、変数f_cを「1」に替えて、関数呼出プログラム140にリターン値「0」を返す。

次に、プログラム生成部は、関数識別子テーブル130から関数識別子が「3」である関数がfuncDummyであることを判断し、153のswitch文に、

case3:funcDummy(a);return(0);を書き込む。

[0058] case3は、152で生成されるxの値が「3」のとき実行される処理を示す。x＝3のとき、ダミー処理であるfuncDummyを実行し、リターン値「0」を返す。

次に、プログラム生成部は、関数識別子テーブル130から関数識別子が「4」である関数がfuncDであることを判断し、153のswitch文に、

case4:if(f_b==1 && f_c==1){funcD(a);return(1);}を書き込む。

[0059] case4は、152で生成されるxの値が「4」のとき実行される処理を示す。x＝4のとき、f_b及びf_cが共に「1」、即ちfuncB、及びfuncC又はfuncC1が処理済である場合に、funcDを処理し、関数呼出プログラム140にリターン値「1」を返す。

次に、プログラム生成部は、関数識別子テーブル130から関数識別子が「5」である関数がfuncCであることを判断し、153のswitch文に、

case5:if(f_c==0){f_c=1;funcC(a);return(0);}を書き込む。

[0060] case5は、152で生成されるxの値が「5」のとき実行される処理を示す。x＝5のとき、151で記憶している変数f_cが「0」の場合、即ちfuncC及びfuncC1が未処理の場合に

、funcCを処理し、変数f_cを「1」に替えて、関数呼出プログラム140にリターン値「0」を返す。

プログラム生成部104は、関数識別子テーブル130に含まれる全ての関数識別子についてcaseを生成すると、関数呼出順決定プログラム生成情報に、default:return(0)を書き込む。defaultは、case0からcase5まで全てのcaseが満たされなかった場合の処理を示す。即ち152で生成されるxの値が0、1、2、3、4、及び5の何れでもない場合に、関数呼出プログラム140にリターン値「0」を返す。

[0061] プログラム生成部104は、関数識別子付与部103から受け取った関数識別子テーブル130、生成した関数呼出プログラム140、及び生成した関数呼出順決定プログラム150を、コンパイル／リンク部105へ渡す。

(5)コンパイル／リンク部105

コンパイル／リンク部105は、関数識別子テーブル130、関数呼出プログラム140、及び関数呼出順決定プログラム150を、コンパイル、及びリンクして、プログラム実行形式を生成する。生成されたプログラム実行形式は、書込部106を介して記憶装置30に書き込まれる。

[0062] (6)書込部106

書込部106は、記憶装置30に対応したドライバであって、コンパイル／リンク部105によりコンパイル、及びリンクされて生成されたプログラム実行形式を、記憶装置30に書き込む。

4. プログラム生成装置10の動作

ここでは、図11に示すフローチャートを用いて、プログラム生成装置10におけるプログラム実行形式生成の動作について説明する。なお、ここで説明する動作は、図2のステップS101の詳細である。

[0063] 先ず、プログラム変換部102は、オリジナルプログラム保持部101が保持するオリジナルプログラム110を読み出し、読み出したオリジナルプログラム110に含まれる各処理を解析し、各処理の依存関係を調べる(ステップS200)。プログラム変換部102は、各処理の依存関係を調べた結果から、並列処理が可能な部分を抽出する(ステップS201)。プログラム変換部102は、各処理をグループに分け、1以上のグループ

に於いて、恒等変換を生成する(ステップS202)。具体的には、プログラム変換部102は、グループ3に於いて、(処理3)の恒等変換である(処理3´)を生成する。

[0064] 次に、プログラム変換部102は、1以上のグループに於いて、ダミー処理を生成する(ステップS203)。具体的には、プログラム変換部102は、グループ2に於いて、(処理2)のダミー処理として(処理2´)を生成する。

次に、プログラム変換部102は、オリジナルプログラム110に含まれる(処理1)から(処理4)、恒等変換(処理3´)、及び、ダミー処理(処理2´)の変数を一つの構造体にして、各処理を関数化して図7に示した関数データ120を生成する(ステップS204)。

[0065] 次に、関数識別子付与部103は、プログラム変換部102で変換した各関数に、各関数を一意に識別する関数識別子を付与し(ステップS205)、図8に示した、関数名と関数識別子とから成る関数識別子テーブル130を生成する(ステップS206)。

次に、プログラム生成部104は、関数呼出プログラム生成情報を読み出し(ステップS207)、読み出した関数呼出プログラム生成情報に、「int n=0」を書き込む(ステップS208)。次に、プログラム生成部104は、関数呼出プログラム生成情報に、「printf("%d¥n",a->d)」を書き込む(ステップS209)。続いて、プログラム生成部104は、関数呼出プログラム生成情報に、ステップS204で生成した関数データ120(図7)を含めて、図9に示した関数呼出プログラム140を生成する(ステップS210)。

[0066] 次に、プログラム生成部104は、関数呼出順決定プログラム生成情報を読み出し(ステップS211)、読み出した関数呼出順決定プログラム生成情報の「FuncSelector( )」に関数名を書き込み、「FuncSelector(struct val_t*a)とする(ステップS212)。次に、プログラム生成部104は、関数呼出順決定プログラム生成情報に、各関数の呼出状態を記憶する変数、及びその初期状態である「static int f_a=f_b=f_c=0」を書き込む(ステップS213)。

[0067] 続いて、プログラム生成部104は、関数識別子テーブル130に含まれる全ての関数識別子について、caseを生成する(ステップS214)。具体的には、関数識別子テーブル130には、0から5までの関数識別子とそれに対応する関数名が記述されており、プログラム生成部104は、図10に示す、case0からcase5までを生成する。プログ

ラム生成部104は、全ての関数識別子に対応するcaseを生成すると、「default:return(0)」を生成する(ステップS215)。

[0068] プログラム生成部104は、ステップS214及びステップS215で生成されたcaseとdefaultとを含めてswitch文を完成させ、関数呼出順決定プログラム生成情報にswitch文を含め、図10に示した関数呼出順決定プログラム150を生成する(ステップS216)。

次に、コンパイル／リンク部105は、関数識別子テーブル130、関数呼出プログラム140、及び関数呼出順決定プログラム150をコンパイル、及びリンクしてプログラム実行形式を生成する(ステップS217)。

[0069] 5. プログラム実行装置20の構成

ここでは、プログラム実行装置20の構成について説明する。

図1に示す様に、プログラム実行装置20は、読出部201、プログラム実行部202、及びメモリ203から構成される。

プログラム実行装置20は、具体的には、マイクロプロセッサ、ROM、RAM、ハードディスクユニット等から構成されるコンピュータシステムである。

[0070] (1)読出部201

読出部201は、記憶装置30に対応したドライバであって、記憶装置30が挿入された状態で、プログラム実行部202からの指示により、記憶装置30に格納されているプログラム実行形式を読み出し、プログラム実行部202に出力する。

(2)プログラム実行部202

プログラム実行部202は、マイクロプロセッサ、作業用のメモリ、関数呼出プログラム140、関数呼出順決定プログラム150などにより実現される。

[0071] プログラム実行部202は、読出部201からプログラム実行形式を受け取り、メモリ203に格納する。また、プログラム実行部202は、メモリ203からプログラム実行形式を呼び出して実行する。

図1のプログラム実行部202の内部は、プログラム実行部202の機能的な構成を示している。同図に示す様に、プログラム実行部202は、プログラムロード部211、テーブル格納部212、関数呼出部213、及び実行部214から構成される。以下では、こ

れらの機能的な構成に基づいて、プログラム実行部202について説明する。

[0072] プログラムロード部211は、記憶装置30からのプログラム実行形式の読み出しを読出部201に指示し、読出部201からプログラム実行形式を受け取る。プログラムロード部211は、受け取ったプログラム実行形式の内、関数呼出プログラム140及び関数呼出順決定プログラム150を、メモリ203にロードする。

また、プログラムロード部211は、メモリ203にロードした関数呼出プログラム140に含まれる各関数の位置を指すアドレスを、読出部201から受け取った関数識別子テーブル130に追加して、図12に示す関数識別子テーブル220を生成する。プログラムロード部211は、生成した関数識別子テーブル220を、テーブル格納部212に格納する。

[0073] テーブル格納部212は、プログラムロード部211から関数識別子テーブル220を受け取り、保持する。関数識別子テーブル220は、図12に示す様に、関数毎に、関数名、関数識別子、及びメモリ上の位置を対応付けている。例えば、関数識別子「0」により識別される関数「funcA」が格納されているメモリ203上の位置を指すアドレスは、「0x07e9500」である。

[0074] 関数呼出部213は、呼出状態保持部を備え、以下の情報を保持する。

(a) funcA、funcB、及び、funcC又はfuncC1が実行部214により実行されたか否かを示す情報。具体的に、関数呼出部213は、図10に示した関数呼出順決定プログラム150の151で示される変数f_a、f_b、及びf_cにより、各関数が実行されたか否かが分かる。

[0075] (b) 関数の並列性を示す情報。具体的に、関数呼出部213は、関数呼出順決定プログラム150の151及び153から、funcBとfuncCとに並列性があることが分かる。

(c) 関数構造を示す情報。具体的に、関数呼出部213は、関数呼出プログラム140の142で示される$X_0=0$、及び、関数呼出順決定プログラム150の153から、funcAが最初に実行され、funcB、及びfuncC又はfuncC1が実行された後、funcDが最後に実行されることが分かる。

[0076] 関数呼出部213は、実行すべき関数の関数識別子を求め、関数識別子テーブル220から対応するアドレスを取得する。次に、関数呼出部213は、取得したアドレスに

基づきメモリ203から関数を呼び出し、実行部214に渡す。

具体的には、関数呼出部213は、X_0＝0であることから、先ず関数識別子が0であるfuncAをメモリ203から呼び出して実行部214に渡す。

[0077] 関数呼出部213は、乱数を生成し、生成した乱数を(式1)のbに代入し、更にX_0＝0を代入し、X_1を得る。関数呼出部213は、関数識別子がX_1である関数を実行すべきか否か、呼出状態保持部の各情報から判断する。実行すべきであると判断する場合に、関数識別子テーブル220から対応するアドレスを取得して、メモリ203から関数を呼び出し、実行部214に渡す。関数を実行部214に渡すと、関数呼出部213は、呼出状態保持部が保持している実行済みであるか否かを示す変数を「実行済み」を示す状態に変える。

[0078] 続いて、関数呼出部213は、(式1)にX_1を代入してX_2を得る。同様に、関数呼出部213は、関数識別子がX_2である関数を実行すべきか否か、呼出状態保持部の各情報から判断する。実行すべきであると判断する場合に、関数識別子テーブル220から対応するアドレスを取得して、メモリ203から関数を呼び出し、実行部214に渡す。関数を実行部214に渡すと、関数呼出部213は、呼出状態保持部が保持している実行済みであるか否かを示す変数を「実行済み」を示す状態に変える。

[0079] 同様にして、関数呼出部213は、(式1)に順にX_nを代入してX_{n+1}を得る。関数呼出部213は、funcDが呼び出され実行されるまで、上記の処理を続ける。funcDが実行されると、関数呼出部213は処理を終了する。

なお、関数呼出部213は、得られたX_nと一致する関数識別子が関数識別子テーブル220に無い場合は、X_nを(式1)に代入して、X_{n+1}を得た後、X_nを破棄する。

[0080] 実行部214は、関数呼出部213から受け取る関数を実行する。

(3)メモリ203

メモリ203は、具体的には、RAM及びROMから構成され、プログラムロード部211により書き込まれる関数呼出プログラム140、及び関数呼出順決定プログラム150を記憶する。

[0081] 6. プログラム実行装置20の動作

ここでは、図13及び図14に示すフローチャートを用いて、プログラム実行装置20の

動作について説明する。

先ず、プログラムロード部211は、読出部201を介して記憶装置30からプログラム実行形式を読み出し、関数呼出プログラム140及び関数呼出順決定プログラム150を、メモリ203にロードする(ステップS301)。更に、プログラムロード部211は、読出部201から関数識別子テーブル130を受け取り、各関数のメモリ203上の位置を示すアドレスを格納し(ステップS302)、関数識別子テーブル220を生成する。プログラムロード部211は、生成した関数識別子テーブル220を、テーブル格納部212に格納する。

[0082] 関数呼出部213は、乱数を生成し(ステップS303)、nを0に設定する(ステップS304)。関数呼出部213は、初期値bとして、ステップS303で生成した乱数を設定し(ステップS305)、

$X_{n+1} = (X_n * 13 + b) \bmod 8$ ・・・ (式1)

$X_0 = 0$ ・・・ (式2)

から、X_nを得る(ステップS306)。ここで、(式2)の$X_0 = 0$は、関数呼出プログラム140により与えられる。

[0083] 関数呼出部213は、テーブル格納部212に格納されている関数識別子テーブル220から、関数識別子が、ステップS306で得たX_nと同一である関数名を取得する(ステップS307)。例えば、関数呼出部213は、ステップS306で得たX_nが、$X_n = 0$の場合、関数識別子テーブル220から関数名funcAを取得し、ステップS306で得たX_nが、$X_n = 3$の場合は、関数識別子テーブル220から、関数名funcDummyを取得する。

[0084] 関数呼出部213は、ステップS307で取得した関数名が、funcDummyである場合(ステップS308でYES)、ステップS312へ進み処理を続け、関数名がfuncDummyでない場合(ステップS308でNO)、呼出状態を確認する(ステップS309)。具体的には、関数呼出部213は、ステップS307で取得した関数名と同一の関数名を、内部の呼出状態保持部が保持するか否か判断する。

[0085] 呼出状態保持部が、同一の関数名を保持する場合、即ち、ステップS307で取得した関数名の関数が既に処理済みである場合(ステップS310でYES)、ステップS316に進み処理を続ける。呼出状態保持部が、同一の関数名を保持しない場合、即ち

、ステップS307で取得した関数名の関数が未だ処理されていない場合(ステップS310でNO)、呼出状態保持部の関数構造を確認する。

[0086] 例えば、funcDは、funcB、及びfuncC又はfuncC1が処理済みでないと、処理されない。

ステップS307で取得した関数名の処理順序が正しくない場合(ステップS311でNO)、ステップS317へ進み処理を続ける。処理順序が正しい場合(ステップS311でYES)、関数呼出部213は、テーブル格納部212に格納されている関数識別子テーブル220から、ステップS307で取得した関数名、即ち、ステップS306で取得したX_nと同一の関数識別子に対応する関数のメモリ203上の位置を示すアドレスを取得する(ステップS312)。例えば、$X_n=0$であり、関数名がfuncAである場合、関数呼出部213は、アドレス「0x07e9500」を取得する。

[0087] 次に、関数呼出部213は、メモリ203における、ステップS312で取得したアドレスが示す位置から、関数を呼び出し(ステップS313)、実行部214に渡す。実行部214は、関数呼出部213から関数を受け取り、実行する(ステップS314)。

続いて、関数呼出部213は、実行された関数の関数名を、呼出状態保持部に処理済みの関数として記憶する(ステップS315)。

[0088] 関数呼出部213は、関数が全て処理されたことにより終了か否か判断し、終了の場合(ステップS316でYES)、処理を終了する。未処理の関数があり、終了でない場合(ステップS316でNO)、関数呼出部213は、nにn+1を設定して(ステップS317)、ステップS306に戻り処理を続ける。

上記のように、プログラム実行装置20は、生成した乱数を初期値bに設定し、

$X_{n+1}=(X_n*13+b)\bmod 8$及び

$X_0=0$、

から、初期値bに固有の乱数列$\{X_0、X_1、X_2、\cdots\}$を取得する。

[0089] 以下では、bに具体的な値を代入して説明する。図15に示したテーブル230は、関数呼出順決定プログラム150に含まれる(式1)、及び関数呼出プログラム140に含まれる(式2)とから、初期値bを変化させたときに得られる乱数列の具体例を示している。

231は、初期値b＝1を(式1)に代入した場合に生成される乱数列の一部を示す。同図に示す様に、b＝1の場合に生成される乱数列は、

{0, 1, 6, 7, 4, 5, 2, 3, 0, 1, 6, 7, 4, ・・・}

である。

[0090] 232は、初期値b＝3を(式1)に代入した場合に生成される乱数列の一部を示す。同図に示す様に、b＝3の場合に生成される乱数列は、

{0, 3, 2, 5, 4, 7, 6, 1, 0, 3, 2, 5, 4, ・・・}

である。

233は、初期値b＝5を(式1)に代入した場合に生成される乱数列の一部を示す。同図に示す様に、b＝5の場合に生成される乱数列は、

{0, 5, 6, 3, 4, 2, 1, 7, 0, 5, 6, 3, 4, ・・・}

である。

[0091] 234は、初期値b＝7を(式1)に代入した場合に生成される乱数列の一部を示す。同図に示す様に、b＝7の場合に生成される乱数列は、

{0, 7, 2, 1, 4, 3, 6, 5, 0, 7, 2, 1, 4, ・・・}

である。

図16に示したテーブル240は、初期値bを変化させたときに実行される関数の順序の具体例を示している。

[0092] 241は、初期値b＝1を(式1)に代入した場合に実行される関数の順序を示す。 b＝1の場合、関数呼出部213は、先ずX_0＝0を得る。関数呼出部213は、関数識別子テーブル220から関数識別子が0であるfuncAのアドレス「0x07e9500」を取得し、メモリ203から、取得したアドレスの位置にあるfuncAを呼び出して実行部214に渡す。実行部214は、funcAを実行する。

[0093] 次に、関数呼出部213は、X_1＝1を得る。関数呼出部213は、関数識別子テーブル220から関数識別子が1である関数がfuncBであることを判断し、呼出状態保持部が保持する情報から、未だfuncBが実行されていないことを判断する。また、呼出状態保持部が保持する情報から実行順序が正しいと判断する。関数呼出部213は、関数識別子テーブルからfuncBのアドレス「0x07e9800」を取得し、メモリ203から、取得

したアドレスの位置にあるfuncBを呼び出して実行部214に渡す。実行部214は、funcBを実行する。

[0094] 次に、関数呼出部213は、X_2＝6を得る。関数識別子が6である関数は存在しないので、関数呼出部213は、次にX_3＝7を得る。同様に、関数識別子が7である関数は、存在しないので、関数呼出部213は、次にX_4＝4を得る。

関数呼出部213は、関数識別子テーブル220から関数識別子が4である関数がfuncDであることを判断する。関数呼出部213は、呼出状態保持部が保持する情報から、funcC又はfuncC1が未だ実行されていないことから、実行順序が正しくないと判断する。

[0095] 次に、関数呼出部213は、X_5＝5を得る。関数呼出部213は、関数識別子テーブル220から関数識別子が5である関数がfuncCであることを判断し、呼出状態保持部が保持する情報から、未だfuncCが実行されていないことを判断する。また、呼出状態保持部が保持する情報から実行順序が正しいと判断する。関数呼出部213は、関数識別子テーブルからfuncCのアドレス「0x07e9900」を取得し、メモリ203から、取得したアドレスの位置にあるfuncCを呼び出して実行部214に渡す。実行部214は、funcCを実行する。

[0096] 次に、関数呼出部213は、X_6＝2を得る。関数呼出部213は、関数識別子テーブル220から関数識別子が2である関数がfuncC1であると判断し、呼出状態保持部が保持する情報から、fucnCが実行済みであることから、ここでは、funcC1は、実行しないことを判断する。

次に、関数呼出部213は、X_7＝3を得る。関数呼出部213は、関数識別子テーブル220から、関数識別子が3であるfuncDummyのアドレス「0x07eb050」を取得し、メモリ203から、取得したアドレスの位置にあるfuncDummyを呼び出して実行部214に渡す。実行部214は、funcDummyを実行する。

[0097] 次に、関数呼出部213は、X_8＝0を得る。関数呼出部213は、関数識別子が0であるfuncAは、既に実行済みであると判断する。

次に、関数呼出部213は、X_9＝1を得る。関数呼出部213は、関数識別子が1であるfuncBは、既に実行済みであると判断する。

次に、関数呼出部213は、X_{10}＝6を得る。関数識別子が6である関数は存在しないので、関数呼出部213は、次にX_{11}＝7を得る。同様に、関数識別子が7である関数は、存在しないので、関数呼出部213は、次にX_{12}＝4を得る。

[0098] 関数呼出部213は、関数識別子テーブル220から関数識別子が4である関数がfuncDであることを判断する。関数呼出部213は、呼出状態保持部が保持する情報から、funcB、及びfuncC又はfuncC1が実行済みであることから、実行順序が正しいと判断する。関数呼出部213は、関数識別子テーブル220から関数識別子が4であるfuncDのアドレス「0x07eb010」を取得し、メモリ203から、取得したアドレスの位置にあるfuncDを呼び出し、実行部214に渡す。実行部214は、funcDを実行する。

[0099] 関数呼出部213は、funcDが実行され全ての関数の処理が終了したと判断して処理を終える。

上記の様に、初期値b＝1の場合、funcA、funcB、funcC、funcDummy、funcDの順序で実行される。

242は、初期値b＝3を(式1)に代入した場合に実行される関数の順序を示す。 b＝1の場合、関数呼出部213は、先ずX_0＝0を得る。関数呼出部213は、関数識別子テーブル220から関数識別子が0であるfuncAのアドレス「0x07e9500」を取得し、メモリ203から、取得したアドレスの位置にあるfuncAを呼び出して実行部214に渡す。実行部214は、funcAを実行する。

[0100] 次に、関数呼出部213は、X_1＝3を得る。関数呼出部213は、関数呼出部213は、関数識別子テーブル220から、関数識別子が3であるfuncDummyのアドレスを取得し、メモリ203から、取得したアドレスの位置にあるfuncDummyを呼び出して実行部214に渡す。実行部214は、funcDummyを実行する。

次に、関数呼出部213は、X_2＝2を得る。関数呼出部213は、関数識別子テーブル220から関数識別子が2である関数がfuncC1であることを判断する。

関数呼出部213は、呼出状態保持部が保持する情報から、funcC1及びfuncC共に未だ実行されていないことを判断する。また、呼出状態保持部が保持する情報から実行順序が正しいと判断する。関数呼出部213は、関数識別子テーブルからfuncC1のアドレスを取得し、メモリ203から、取得したアドレスの位置にあるfuncC1を呼び出し

て実行部214に渡す。実行部214は、funcC1を実行する。

[0101] 次に、関数呼出部213は、X_3＝5を得る。関数呼出部213は、関数識別子テーブル220から関数識別子が5である関数がfuncCであると判断し、呼出状態保持部が保持する情報から、fucnC1が実行済みであることから、ここでは、funcCは、実行しないことを判断する。

次に、関数呼出部213は、X_4＝4を得る。関数呼出部213は、関数識別子テーブル220から関数識別子が4である関数がfuncDであることを判断する。関数呼出部213は、呼出状態保持部が保持する情報から、funcBが未だ実行されていないことから、実行順序が正しくないと判断する。

[0102] 次に、関数呼出部213は、X_5＝7を得る。関数識別子が7である関数は存在しないので、関数呼出部213は、次にX_6＝6を得る。同様に、関数識別子が6である関数は、存在しないので、関数呼出部213は、次にX_7＝1を得る。

関数呼出部213は、関数識別子テーブル220から関数識別子が1である関数がfuncBであることを判断し、呼出状態保持部が保持する情報から、未だfuncBが実行されていないことを判断する。また、呼出状態保持部が保持する情報から実行順序が正しいと判断する。関数呼出部213は、関数識別子テーブルからfuncBのアドレスを取得し、メモリ203から、取得したアドレスの位置にあるfuncBを呼び出して実行部214に渡す。実行部214は、funcBを実行する。

[0103] 次に、関数呼出部213は、X_8＝0を得る。関数呼出部213は、関数識別子が0であるfuncAは、既に実行済みであると判断する。

次に、関数呼出部213は、X_9＝3を得る。関数呼出部213は、関数呼出部213は、関数識別子テーブル220から、関数識別子が3であるfuncDummyのアドレスを取得し、メモリ203から、取得したアドレスの位置にあるfuncDummyを呼び出して実行部214に渡す。実行部214は、funcDummyを実行する。

[0104] 次に、関数呼出部213は、X_{10}＝2を得る。関数呼出部213は、関数識別子が2であるfuncC1は、既に実行済みであると判断する。

次に、関数呼出部213は、X_{11}＝5を得る。関数呼出部213は、関数識別子テーブル220から関数識別子が5である関数がfuncCであると判断し、呼出状態保持部が保

持する情報から、fucnC1が実行済みであることから、ここでは、funcCは、実行しないことを判断する。

[0105] 次に、関数呼出部213は、X_{12}＝4を得る。関数呼出部213は、関数識別子テーブル220から関数識別子が4である関数がfuncDであることを判断する。関数呼出部213は、呼出状態保持部が保持する情報から、funcB、及びfuncC又はfuncC1が実行済みであることから、実行順序が正しいと判断する。関数呼出部213は、関数識別子テーブル220から関数識別子が4であるfuncDのアドレスを取得し、メモリ203から、取得したアドレスの位置にあるfuncDを呼び出し、実行部214に渡す。実行部214は、funcDを実行する。

[0106] 関数呼出部213は、funcDが実行され全ての関数の処理が終了したと判断して処理を終える。

上記の様に、初期値b＝3の場合、funcA、funcDummy、funcC1、funcB、funcDummy、funcDの順序で実行される。

243は、初期値b＝5の場合に実行される関数の順序を示している。初期値b＝5の場合、funcA、funcC、funcDummy、funcB、funcDummy、funcDの順序で実行される。

[0107] 244は、初期値b＝7の場合に実行される関数の順序を示している。初期値b＝7の場合、funcA、funcC1、funcB、funcDの順序で実行される。

7. 関数の属性によるグループ化について

以上説明したように、プログラム生成装置10のプログラム変換部102は、オリジナルプログラムを解析し、解析結果である関数構造をプログラム生成部104へ通知する構成を有する。

[0108] ここで、プログラム生成装置10は、オリジナルプログラムを解析し、オリジナルプログラムに含まれる各関数の属性により関数をグループ化した関数グループに従い関数の実行順序を決定するようなプログラム実行形式を生成してもよい。

ここでは、図17に示した関数構造250を用いて、関数グループに従い関数の実行順序を決定する方法について説明する。同図に示す様に、関数構造250は、シーケンシャルグループ251、並列グループ252、及び恒等グループ254を含む。

[0109] シーケンシャルグループ251は、funcA261、並列グループ252、及びfuncD266か

ら成る。

並列グループ252は、グループ253及び恒等グループ254から成る。

恒等グループ254は、funcC264及びfuncC1265から成る。

プログラム生成装置10は、シーケンシャルグループ251について、261→252→266の順序にシーケンシャルに実行し、並列グループ252について、実行時にランダムに決定される順序でグループ253と恒等グループ254とを実行し、恒等グループ254について、funcC264又はfuncC1265の何れか一方を実行時にランダムに選択して実行し、funcDummy263について、実行するか又は実行しないか、及び、実行する場合に何回実行するか、実行時にランダムに決定して実行する関数呼出プログラムを生成する。

[0110] プログラム実行装置20は、プログラム生成装置10が生成した関数呼出プログラムを実行することにより、実行毎に異なる順序で各関数を実行する。

＜第2の実施形態＞

ここでは、本発明に係る第2の実施形態として、情報処理システム2について説明する。

[0111] 情報処理システム2は、プログラム実行形式の生成時に、複数の実行経路パターンを生成し、プログラム実行時には、生成された複数の実行経路パターンの内、一の実行経路パターンを選択して実行することを特徴とする。

1. 全体の構成

図18は、情報処理システム2の構成を示す構成図である。同図に示すように、情報処理システム2は、プログラム生成装置10a、プログラム実行装置20a、及び記憶装置30aから構成される。

[0112] プログラム生成装置10a及びプログラム実行装置20aは、第1の実施形態で開示したプログラム生成装置10及びプログラム実行装置20と同様、コンピュータシステムであり、記憶装置30aは、光ディスクである。

2. 全体の動作

ここでは、図19に示すフローチャートを用いて、情報処理システム2全体の動作について説明する。

[0113] プログラム生成装置10aは、オリジナルプログラムからプログラム実行形式を生成し（ステップS401）、その後、実行経路パターンリストを生成する（ステップS402）。プログラム生成装置10aは、生成したプログラム実行形式と実行経路パターンリストとを記憶装置30aに書き込む（ステップS403）。

記憶装置30aは、プログラム実行形式と実行経路パターンリストとを記憶する（ステップS103）。

[0114] プログラム実行装置20aは、記憶装置30aからプログラム実行形式と実行経路パターンリストとを読み出す（ステップS405）。プログラム実行装置20aは、実行経路パターンリストから初期値を選択し（ステップS406）、選択した初期値に応じて決定される順序で、プログラム実行形式の各処理を実行する（ステップS407）。

3. プログラム生成装置10aの構成

ここでは、図18を用いて、プログラム生成装置10aの構成について説明する。同図に示すように、プログラム生成装置10aは、オリジナルプログラム保持部101a、プログラム変換部102a、関数識別子付与部103a、プログラム生成部104a、コンパイル／リンク部105a、書込部106a、及び実行経路パターン生成部107aから構成される。

[0115] プログラム生成装置10aは、図1に示したプログラム生成装置10に、実行経路パターン生成部107aを追加した構成を有する。

オリジナルプログラム保持部101a、プログラム変換部102a、関数識別子付与部103a、プログラム生成部104a、コンパイル／リンク部105a、及び書込部106aは、オリジナルプログラム保持部101、プログラム変換部102、関数識別子付与部103、プログラム生成部104、コンパイル／リンク部105、及び書込部106と同様の機能を有する。

[0116] 即ち、オリジナルプログラム保持部101aは、図3に示したオリジナルプログラム110を保持しており、プログラム変換部102aは、オリジナルプログラム110を変換して、図7に示した関数データ120を生成する。関数識別子付与部103aは、関数データ120に含まれる各関数に、識別子を付与し、図8に示した関数識別子テーブル130を生成する。プログラム生成部104aは、図9に示した関数呼出プログラム140と、図10に示した関数呼出順決定プログラム150とを生成する。

[0117] 実行経路パターン生成部107aは、先に述べたように第2の実施形態のプログラム生成装置10aに特徴的な構成要素である。実行経路パターン生成部107aは、実行経路パターンリストを生成する。実行経路パターン生成部107aは、プログラム生成部104aにより生成された関数呼出プログラム140と関数呼出順決定プログラム150とを受け取り、受け取った関数呼出プログラム140と、関数呼出順決定プログラム150と、生成した実行経路パターンリストとをコンパイル／リンク部105aへ出力する。

[0118] ここで、図20は、実行経路パターンリスト300のデータ構成を示す図である。実行経路パターンリスト300は、実行経路パターン生成部107aが生成する実行経路パターンリストの具体例である。同図に示すように、実行経路パターンリスト300は、4つのパターン情報301、302、303、及び304から構成される。各パターン情報は、index、実行経路パターン値、及びフラグを含む。

[0119] 各indexに対応付けられた実行経路パターン値は、実行経路パターン生成部107aにより生成された乱数の値である。フラグは、0又は1に設定されており、0は、プログラム実行装置20aによるプログラム実行形式の実行に際し、対応付けられている実行経路パターン値が、また選択されていないことを示し、1は、対応付けられている実行経路パターン値が、すでに選択されたことを示す。

[0120] コンパイル／リンク部105aは、実行経路パターン生成部107aから、関数呼出プログラム140と、関数呼出順決定プログラム150と、実行経路パターンリストとを受け取る。コンパイル／リンク部105aは、受け取ったこれらのプログラム及びデータを、コンパイル及びリンクして、プログラム実行形式を生成する。コンパイル／リンク部105aは、生成したプログラム実行形式を書込部106aへ出力する。

[0121] 書込部106aは、コンパイル／リンク部105aから出力されるプログラム実行形式を、記憶装置30aに書き込む。

4. 実行経路パターンリスト生成の動作

ここでは、図21に示したフローチャートを用いて、プログラム生成装置10aによる実行経路パターンリスト生成処理の動作について説明する。

[0122] 先ず、プログラム生成装置10aの実行経路パターン生成部107aは、実行経路パターン数Nを決定する（ステップS411）。実行経路パターン数Nは、予め所定数が実行

経路パターン生成部107aの内部に保持されており、この所定数をNとして用いてもよいし、外部の入力装置からユーザの入力を受け付け、受け付けた値をNとして用いてもよい。ここで、実行経路パターン生成部107aが、図20に示した実行経路パターンリスト300を生成しようとする場合は、実行経路パターン数N＝4とする必要がある。

[0123] 次に、実行経路パターン生成部107aは、実行経路パターンリストを内部にセットする(ステップS412)。ここでセットされる実行経路パターンリストは、indexの列については、上から順に0から(N－1)までの数値が書き込まれており、フラグの列については、上から順に全て0が書き込まれており、実行経路パターン値の列については、まだ何れのデータも書き込まれていない空欄状態のリストである。

[0124] 次に、実行経路パターン生成部107aは、l＝0とし(ステップS413)、乱数rを生成する(ステップS414)。実行経路パターン生成部107aは、ステップS414で生成した乱数rが、既に実行経路パターンリストの実行経路パターン値に含まれるか否か判断する(ステップS415)。rが実行経路パターンリストに含まれる場合(ステップS415でYES)、ステップS414に戻って処理を続ける。rが実行経路パターンリストに含まれない場合(ステップS415でNO)、実行経路パターン生成部107aは、rを、実行経路パターンリストに書き込む(ステップS416)。

[0125] 次にl＝l＋1とし(ステップS417)、実行経路パターン生成部107aは、lとNとが一致するか否か判断する(ステップS418)。Nは、ステップS411で決定した実行経路パターン数である。

l≠Nの場合(ステップS418でNO)、実行経路パターン生成部107aは、ステップS414へ戻り処理を続ける。l＝Nの場合(ステップS418でYES)、即ち、実行経路パターン数Nと一致する数の実行経路パターン値を含む実行経路パターンリストが生成された場合に、実行経路パターン生成部107aは、生成された実行経路パターンリストを、コンパイル／リンク部105aへ出力する(ステップS419)。

[0126] 5. プログラム実行装置20aの構成

プログラム実行装置20aは、記憶装置30aからプログラム実行形式を読み出し、読み出したプログラム実行形式を実行する装置である。

図18に示すように、プログラム実行装置20aは、読出部201aとプログラム実行部2

02aとメモリ203aとから構成され、プログラム実行部202aは、プログラムロード部211a、テーブル格納部212a、関数呼出部213a、及び実行部214aから構成される。

[0127] 各構成要素の機能については、第1の実施形態におけるプログラム実行装置20と同様である。

第1の実施形態で開示したプログラム実行装置20は、プログラム実行時に、乱数を生成し、生成した乱数を初期値とし、初期値に応じて決定される順序で、プログラムを実行する構成を有していた。一方、本実施形態におけるプログラム実行装置20aは、プログラム実行時に、図20に示した実行経路パターンリストから1の実行経路パターン値を選択して、選択した実行経路パターン値を初期値として、初期値に応じて決定される順序で、プログラムを実行する。

[0128] 6. プログラム実行処理の動作

図22及び図23は、プログラム実行装置20aによるプログラム実行処理の動作を示すフローチャートである。なお、ここでは、具体例として、記憶装置30aには、関数識別子テーブル130(図8)、関数呼出プログラム140(図9)、関数呼出順決定プログラム150(図10)、及び実行経路パターンリスト300(図20)が格納されているものとする。

[0129] 先ず、プログラム実行装置20aの読出部201aは、記憶装置30aから関数呼出プログラム140及び関数呼出順決定プログラム150を読み出す。プログラムロード部211aは、読出部201aが読み出した関数呼出プログラム140及び関数呼出順決定プログラム150を、メモリ203aにロードする(ステップS431)。

更に、読出部201aは、記憶装置30aから関数識別子テーブル130を読み出す。プログラムロード部211aは、関数識別子テーブル130に、各関数のメモリ203a上の位置を示すアドレスを格納し(ステップS432)、関数識別子テーブル220(図12)を生成する。テーブル格納部212aは、生成された関数識別子テーブル220を、内部に格納する。

[0130] また、プログラム実行装置20aは、記憶装置30aから実行経路パターンリスト300を読み出し、読み出した実行経路パターンリスト300を、メモリ203aに格納する。

次に、プログラム実行部202aの関数呼出部213aは、実行経路パターンリスト300

に含まれる各indexに対応付けられたフラグの値を合計し、合計値をFとする(ステップS433)。関数呼出部213aは、Fと実行経路パターン値Nとが一致するか否か判断する(ステップS434)。ここで実行経路パターンリスト300の実行経路パターン数は、N＝4である。

[0131] F＝Nの場合(ステップS434でYES)、関数呼出部213aは、実行経路パターンリスト300のフラグ値を全て0にセットしなおす(ステップS435)。その後、ステップS433へ進み、処理を続ける。

F≠Nの場合(ステップS434でNO)、関数呼出部213aは、乱数Rを生成し(ステップS436)、R mod Nの値を、indexとして指定する(ステップS437)。R mod Nとは、RをNで割った余りである。関数呼出部213aは、メモリ203aに格納されている実行経路パターンリスト300から、指定したindexに対応付けられているフラグ値を読む(ステップS438)。

[0132] フラグ値が1の場合(ステップS439でNO)、関数呼出部213aは、ステップS436へ戻って処理を続ける。

フラグ値が0の場合(ステップS439でYES)、関数呼出部213aは、フラグ値を0から1に替えてセットし(ステップS440)、実行経路パターンリスト300から、指定したindexに対応付けられている実行経路パターン値を読み出し(ステップS441)、読み出した実行経路パターン値をrとする。

[0133] 関数呼出部213aは、実行経路パターン値rを初期値として、

$X_0 = 0$ 、及び

$X_{n+1} = (X_n * 13 + r) \bmod 8$

から、X_nを得る(ステップS442)。

関数呼出部213aは、テーブル格納部212aに格納されている関数識別子テーブル220から、関数識別子が、ステップS442で得たX_nと同一である関数名を取得する(ステップS451)。関数呼出部213aは、ステップS451で取得した関数名がfuncDummyである場合(ステップS452でYES)、ステップS456へ進み処理を続け、関数名がfuncDummyでない場合(ステップS452でNO)、呼出状態を確認する(ステップS453)。具体的には、関数呼出部213aは、ステップS451で取得した関数名と同一の関

数名を、内部の呼出状態保持部が保持するか否か判断する。ここで、呼出状態保持部は、第1の実施形態であるプログラム実行装置20の関数呼出部213が備える呼出状態保持部と同様の情報を保持しているものとする。

[0134] 呼出状態保持部が、同一の関数名を保持する場合、即ち、ステップS451で取得した関数名の関数が既に処理済みである場合(ステップS454でYES)、ステップS460に進み処理を続ける。呼出状態保持部が、同一の関数名を保持しない場合、即ち、ステップS451で取得した関数名の関数が未だ処理されていない場合(ステップS454でNO)、呼出状態保持部の関数構造を確認する。

[0135] 例えば、funcDは、funcB、及びfuncC又はfuncC1が処理済みでないと、処理されない。

ステップS451で取得した関数名の処理順序が正しくない場合(ステップS455でNO)、ステップS461へ進み処理を続ける。処理順序が正しい場合(ステップS455でYES)、関数呼出部213aは、テーブル格納部212aに格納されている関数識別子テーブル220から、ステップS451で取得した関数名、即ち、ステップS442で取得した X_n と同一の関数識別子に対応する関数のメモリ上の位置を示すアドレスを取得する(ステップS456)。

[0136] 次に、関数呼出部213aは、メモリ203aにおける、ステップS456で取得したアドレスが示す位置から、関数を呼び出し(ステップS457)、実行部214aに渡す。実行部214aは、関数呼出部213aから関数を受け取り、実行する(ステップS458)。続いて、関数呼出部213aは、実行された関数の関数名を、呼出状態保持部に処理済みの関数として記憶する(ステップS459)。

[0137] 関数呼出部213aは、実行すべき関数が全て処理されたことにより終了か否か判断し(ステップS460)、終了の場合(ステップS460でYES)、処理を終了する。未処理の関数があり、終了でない場合(ステップS460でNO)、関数呼出部213aは、nにn+1を設定して(ステップS461)、ステップS442に戻り処理を続ける。

＜第3の実施形態＞

ここでは、本発明に係る第3の実施形態として、情報処理システム3について説明する。

[0138] 情報処理システム3は、プログラム実行形式を生成し、生成したプログラム実行形式が正常に動作するか否かをテストし、正常に動作する実行経路パターンを複数個抽出し、プログラム実行時には、正しく動作する複数の実行経路パターンの内、一の実行経路パターンを選択して実行することを特徴とする。

1. 全体の構成

図24は、情報処理システム3の構成を示す図である。同図に示すように、情報処理システム3は、プログラム生成装置10b、プログラム実行装置20b、記憶装置30b、プログラムテスト装置40b、及び記憶装置50bから構成される。

[0139] プログラム生成装置10b、プログラム実行装置20b、及びプログラムテスト装置40bは、コンピュータシステムである。また、記憶装置30b及び50bは、光ディスクである。

プログラム生成装置10bは、第1の実施形態で開示したプログラム生成装置10(図1)と同一の構成及び機能を有する。プログラム実行装置20bは、第2の実施形態で開示したプログラム実行装置20aと同一の構成及び機能を有する。プログラムテスト装置40bは、第3の実施形態に特有の構成要素である。

[0140] 2. 全体の動作

図25は、情報処理システム3全体の動作を示すフローチャートである。

プログラム生成装置10bは、オリジナルプログラムからプログラム実行形式を生成する(ステップS501)。プログラム生成装置10bは、記憶装置30bを介して、生成したプログラム実行形式をプログラムテスト装置40aへ渡す(ステップS502)。

[0141] プログラムテスト装置40bは、記憶装置30bから、プログラム実行形式を読み出し、実行する。プログラムテスト装置40bは、正常動作する実行経路パターンを複数個選択し(ステップS503)、実行経路パターンリストを生成する(ステップS504)。プログラムテスト装置40bは、記憶装置50bを介して、プログラム実行形式と実行経路パターンリストとをプログラム実行装置20bへ渡す(ステップS505)。

[0142] プログラム実行装置20bは、記憶装置50bからプログラム実行形式と実行経路パターンリストとを読み出す。プログラム実行装置20bは、実行経路パターンリストから初期値を選択し(ステップS506)、選択した初期値に応じて決定される順序で、プログ

ラム実行形式の各処理を実行する(ステップS507)。

3. プログラム生成装置10bの構成及び動作

プログラム生成装置10bは、第1の実施形態で開示したプログラム生成装置10(図1)と同様の構成を有するため、プログラム生成装置10bの内部構成については図示していない。

[0143] 即ち、プログラム生成装置10bは、オリジナルプログラム保持部、プログラム変換部、関数識別子付与部、プログラム変換部、コンパイル/リンク部、及び書込部から構成される。

オリジナルプログラム保持部は、図3に示したオリジナルプログラム110とオリジナルプログラム110の実行結果とを保持している。オリジナルプログラム保持部は、オリジナルプログラム110をプログラム変換部へ渡し、オリジナルプログラム110の実行結果を書込部へ渡す。

[0144] プログラム変換部は、オリジナルプログラム110を変換して、図7に示した関数データ120を生成する。関数識別子付与部は、関数データ120に含まれる各関数に、識別子を付与し、図8に示した関数識別子テーブル130を生成する。プログラム生成部は、図9に示した関数呼出プログラム140と、図10に示した関数呼出順決定プログラム150を生成する。

[0145] コンパイル/リンク部は、プログラム生成部から関数呼出プログラム140及び関数呼出順決定プログラム150を受け取る。コンパイル/リンク部は、受け取ったこれらのプログラム及びデータを、コンパイル及びリンクして、プログラム実行形式を生成する。コンパイル/リンク部は、生成したプログラム実行形式を書込部へ出力する。

書込部は、コンパイル/リンク部から出力されるプログラム実行形式を、記憶装置30bに書き込む。また、書込部は、オリジナルプログラム保持部から出力されるオリジナルプログラム110の実行結果を、記憶装置30bに書き込む。

[0146] なお、プログラム生成装置10bによる、プログラム実行形式生成処理の動作は、図11に示したプログラム生成装置10による動作と同様であるため、説明を省略する。

4. プログラムテスト装置40bの構成

プログラムテスト装置40bは、図24に示すように、読出部401b、プログラム実行部4

02b、メモリ403b、実行経路パターン生成部404b、及び書込部405bから構成され、プログラム実行部402bは、更に、プログラムロード部411b、テーブル格納部412b、関数呼出部413b、及び実行部414bから構成される。

[0147] 即ち、プログラムテスト装置40bは、第1の実施形態で開示したプログラム実行装置20（図1）に、実行経路パターン生成部404bと書込部405bとを追加した構成を有する。

読出部401bは、記憶装置30bからプログラム実行形式とオリジナルプログラム110の実行結果とを読み出し、実行結果を実行経路パターン生成部404bへ渡し、プログラム実行形式をプログラムロード部411bへ渡す。

[0148] プログラムロード部411bは、読出部401bからプログラム実行形式を受け取り、受け取ったプログラム実行形式の内、関数呼出プログラム140及び関数呼出順決定プログラム150を、メモリ403bにロードする。

また、プログラムロード部411bは、メモリ403bにロードした関数呼出プログラム140に含まれる各関数の位置を指すアドレスを、関数識別子テーブル130に追加して、図12に示す関数識別子テーブル220を生成する。プログラムロード部411bは、生成した関数識別子テーブル220を、テーブル格納部412bに格納する。

[0149] テーブル格納部412bは、プログラムロード部411bから関数識別子テーブル220を受け取り、保持する。

関数呼出部413bは、第1及び第2の実施形態と同様に、呼出状態保持部を備え上記実施形態で説明した情報を保持する。

関数呼出部213は、実行経路パターン生成部404bから初期値rを受け取り、受け取った初期値rに応じた順序でメモリ403bから関数を呼び出し、実行部414bに渡す。

[0150] 実行部414bは、関数呼出部413bから受け取る関数を実行する。更に、実行部414bは、プログラムの実行結果を、実行経路パターン生成部404bへ渡す。

実行経路パターン生成部404bは、第2の実施形態で開示したプログラム生成装置10aの構成要素である実行経路パターン生成部107aと同様の機能を有する。即ち、実行経路パターン生成部404bは、実行経路パターンリストを生成する。

[0151] ここで、実行経路パターン生成部404bと実行経路パターン生成部107aとの相違点は、実行経路パターン生成部404bは、生成した実行経路パターンに基づきプログラム実行形式を実行した結果と、オリジナルプログラム110の実行結果とを比較し、両者が等しいと判断された実行経路パターンのみから成るリストを生成する点である。これに対し、実行経路パターン生成部107aは、複数の実行経路パターンを生成するにとどまり、生成した実行経路パターンのそれぞれが、オリジナルプログラム110の実行結果と等しいか否かについての検証は行っていない。

[0152] 実行経路パターン生成部404bは、生成した実行経路パターンリストを書込部405bへ渡す。

書込部405bは、実行経路パターン生成部404bにより生成された実行経路パターンリストと、メモリ403bに記憶していたプログラム実行形式とを記憶装置50bに書き込む。

[0153] メモリ403bは、具体的には、RAM及びROMから構成され、プログラムロード部411bにより書き込まれる関数呼出プログラム140、及び関数呼出順決定プログラム150を記憶する。

5. 実行経路パターンリスト生成処理の動作

ここでは、図26に示すフローチャートを用いて、プログラムテスト装置40bによる、実行経路パターンリスト生成処理の動作について説明する。

[0154] 実行経路パターン生成部404bは、実行経路パターン数Nを決定する(ステップS511)。また、実行経路パターン生成部404bは、読出部401bを介して記憶装置30bからオリジナルプログラム110の実行結果Val1を取得する(ステップS512)。

次に、実行経路パターン生成部404bは、実行経路パターンリストを内部にセットする(ステップS513)。ここでセットされる実行経路パターンリストは、indexの列については、上から順に0から(N－1)までの数値が書き込まれており、フラグの列については、上から順に全て0が書き込まれており、実行経路パターン値の列については、まだ何れのデータも書き込まれていない空欄状態のリストである。

[0155] 次に、実行経路パターン生成部404bは、l＝0とし(ステップS514)、乱数rを生成する(ステップS515)。実行経路パターン生成部107aは、ステップS515で生成した

乱数rが、既に実行経路パターンリストの実行経路パターン値に含まれるか否か判断する(ステップS516)。rが実行経路パターンリストに含まれる場合(ステップS516でYES)、ステップS515に戻って処理を続ける。rが実行経路パターンリストに含まれない場合(ステップS516でNO)、rを初期値とし、プログラム実行部402bによりプログラム実行形式の実行処理を行う(ステップS518)。

[0156] なお、ステップS518のプログラム実行処理の動作は、図22及び図23に示したフローチャートの、ステップS441からステップS460までの動作と同一であるので、詳細な説明は省略する。

実行経路パターン生成部404bは、実行部414bからプログラム実行形式の出力結果Val2を受け取る(ステップS518)。実行経路パターン生成部404bは、オリジナルプログラム110の実行結果Val1と、初期値rとして実行したプログラム実行形式の実行結果Val2とを比較する(ステップS519)。

[0157] Val1≠Val2のとき(ステップS519でNO)、実行経路パターン生成部404bは、ステップS515に戻り処理を続ける。

Val1=Val2のとき(ステップS519でYES)、実行経路パターン生成部404bは、rを実行経路パターンリストに書き込む(ステップS520)。

次にl=l+1とし(ステップS521)、実行経路パターン生成部404bは、lとNとが一致するか否か判断する(ステップS522)。Nは、ステップS511で決定した実行経路パターン数である。

[0158] l≠Nの場合(ステップS522でNO)、実行経路パターン生成部404bは、ステップS515へ戻り処理を続ける。l=Nの場合(ステップS522でYES)、即ち、実行経路パターン数Nと一致する数の実行経路パターン値を含む実行経路パターンリストが生成された場合に、実行経路パターン生成部404bは、生成された実行経路パターンリストを書込部405bへ出力する。

[0159] 6. プログラム実行装置20bの構成及び動作

プログラム実行装置20bは、図24に示すように、読出部201b、プログラム実行部202b、及びメモリ203bから構成され、プログラム実行部202bは、プログラムロード部211b、テーブル格納部212b、関数呼出部213b、及び実行部214bから構成される。

[0160] プログラム実行装置20bの各構成要素は、第2の実施形態で開示したプログラム実行装置20aの各構成要素と同様の機能を有する。即ち、プログラム実行装置20bは、プログラム実行時に、メモリ203bに格納されている実行経路パターンリストから1の実行経路パターン値を選択し、選択した実行経路パターン値を初期値として、初期値に応じて決定される順序で、プログラムを実行する。

[0161] ＜その他の変形例＞

なお、本発明を上記の実施形態に基づき説明してきたが、本発明は上記の実施形態に限定されないのは勿論であり、以下の様な場合も本発明に含まれる。

(1) 上記の実施形態では、プログラム生成装置は、各命令を関数化し、各関数を線形合同法の式とswitch文とにより呼び出すプログラムを生成したが、本発明においてプログラム生成装置が生成するプログラムは、この構成に限定されない。例えば、プログラム生成装置が、図27(a)及び(b)に示すように、各命令を関数化せずswitch文とgoto文とを用い、各命令を実行させるプログラムを生成する場合も本発明に含まれる。

[0162] 図27(a)に示すプログラム401は、図9に示した関数呼出プログラム140から、funcA、funcB、funcC1、funcC、funcD、及びfuncDummyを削除したものである。そして、図27(b)に示すプログラム402により、呼出順を決定し、goto文でジャンプし、命令を実行する。

(2) 上記の実施形態では、関数の呼出順序を決定する方法として擬似乱数を用いているが、本発明において関数の呼出順序を決定する方法は、擬似乱数に限定されないのは勿論であり、ある初期値を与えることにより、それ以降の状態がランダムに決まる方法であれば、他の方法を用いてもよい。

[0163] 関数の呼出順序を決定する方法として、例えばカオス関数を用いてもよい。カオス関数は、ある時点での状態(初期値)が決まればその後の状態が原理的に全て決定されるという決定論的方式に従っているにも関わらず、非常に複雑、不規則、且つ不安定な振る舞いをして、次の状態の予測が不可能であるという性質を有する。

また、関数の呼出順序を決定する方法として、例えばセルオートマトンを用いてもよい。セルオートマトンは、離散的な状態が、ある決まった規則に従い離散時間で変化

する数理モデルであって、初期値、及び近傍のn－1の状態から、nの状態が決まるという性質を有する。

[0164] ここでは、図28、図29、及び図30を参照し、セルオートマトンを用いて、呼び出す関数の関数識別子を順次決定し、決定された関数識別子の順序に応じて、呼出順序を決定する方法について説明する。

図28は、セルオートマトンにより関数識別子を決定する手順を示したフローチャートである。

[0165] 先ず、n＝0とし(ステップS601)、(a_n^1, a_n^2, a_n^3)の初期値(a_0^1, a_0^2, a_0^3)s.t.a_i^jmod8を設定する(ステップS602)。

ここで、(a_n^1, a_n^2, a_n^3)は、初期値(a_0^1, a_0^2, a_0^3)からn経過後の状態を示している。

n＝0の場合(ステップS603でYES)、ステップS605へ進む。n≠0の場合(ステップS603でNO)、$a_n^i = a_{n-1}^i + a_{n-1}^{i+1}$、及び$a_n^3 = a_{n-1}^3$から、$(a_n^1, a_n^2, a_n^3)$を算出する(ステップS604)。

[0166] ステップS604で算出されたa_n^1を関数識別子とする(ステップS605)。処理すべき全ての関数が呼び出され、終了の場合(ステップS606でYES)、セルオートマトンによる関数識別子決定処理を終了する。

終了でない場合(ステップS606でNO)、nをn＋1として(ステップS607)、ステップS603へ進み処理を続ける。

[0167] 図29に示したテーブル500は、初期値を$a_0^1 = 1$、$a_0^2 = 0$、及び$a_0^3 = 1$として図28のフローチャートに従い、n＝1からn＝20までの各要素(a_n^1, a_n^2, a_n^3)を算出し、算出された各要素をまとめたテーブルである。

図30に示したテーブル600は、図29のテーブル500のように各要素が算出され、a_n^1を関数識別子とした場合に実行される関数の順序の具体例を示している。同図によると、初期値を$a_0^1 = 1$、$a_0^2 = 0$、及び$a_0^3 = 1$としてセルオートマトンを用いた場合、関数は、funcA、funcC、funcDummy、funcB、funcDの順序で実行される。

[0168] (3)第3の実施形態においては、プログラムテスト装置は正常動作するパターンのみを抽出し、プログラム実行装置は正常動作するパターンのみを記憶する構成を有するが、本発明においてはこの構成は必須ではない。例えば、プログラムテスト装置

は異常動作するパターンのみを抽出し、プログラム実行装置は異常動作するパターンのリストを記憶しておく構成であっても本発明に含まれる。この場合、プログラム実行装置は、実行時に乱数を生成し、生成した乱数が異常動作するパターンのリストに含まれているか否か判断し、含まれていないと判断した場合には、生成した乱数を初期値として用いて関数識別子を決定する。生成した乱数が異常動作するパターンのリストに含まれていると判断した場合には、プログラム実行装置は、異なる乱数を生成する。プログラム実行装置は、リストに含まれていない乱数を生成するまで、乱数の生成を続ける。

[0169] (4)第3の実施形態では、プログラム生成装置10bとプログラムテスト装置40bとが独立した2の装置で実現される構成を有するが、この構成は必須ではなく、プログラム生成装置10bとプログラムテスト装置40bとの機能を備えた1の装置で実現される構成であっても本発明に含まれる。

(5)オリジナルプログラムに分岐がある場合には、プログラム生成装置は、定数や変数を追加する冗長化処理を行い、内部に分岐のないプログラムに変形するように構成してもよい。また、オリジナルプログラムを複雑化するために、冗長化処理を行ってもよい。

[0170] (6)上記実施形態では、実行時に呼出順序を決定して呼び出される単位を関数としているが、本発明において、呼出順序を決定して呼び出される単位は、関数に限定されない。並列処理が可能な任意の大きさの処理ブロックであっても本発明に含まれる。

(7)上記第1の実施形態では、実行時に、プログラム実行装置20が乱数を生成し、生成した乱数を、擬似乱数生成アルゴリズムである線形合同法の初期値bに代入する構成を有しているが、初期値bは、プログラム実行装置20が生成する乱数に限定されず、実行ごとに異なるランダムな値であればよい。

[0171] 例えばプログラム実行装置20が実行時の時計を読み、それに基づきbを設定してもよいし、実行時にある特定の記憶領域が保持する値を読み、それに基づきbを設定してもよい。

(8)関数呼出順決定プログラム150に、関数の呼出順序を決定する方法を複数含

めてもよい。この場合、実行時にプログラム実行装置で生成される乱数に応じて前記複数の関数呼出順序決定方法のうち、1の方法を選択するように構成してもよい。

[0172] (9)上記実施の形態では、オリジナルプログラムの全体を関数化して、関数呼出プログラム140、及び関数呼出順決定プログラム150を生成して、全ての関数をこれらのプログラムを用いて呼び出す構成を有しているが、上記技術を、オリジナルプログラムの内の並列処理が可能な部分にのみ適用した場合も本発明に含まれる。この場合、関数呼出順決定プログラム150の呼出状態を保持する151が不要になる。

[0173] (10)プログラム生成装置は、擬似乱数を生成する式を含む関数呼出順決定プログラム150を生成する代わりに、アドレスを生成する式を含む関数呼出順決定プログラム150を生成するように構成してもよい。この場合、プログラム実行装置は、関数識別子テーブル220を生成、保持する必要がなくなり、関数呼出時には、オフセットを用いてもよい。

[0174] (11)上記実施形態においては、プログラム生成装置、プログラムテスト装置、及びプログラム実行装置間でプログラムを受け渡しする際に、光ディスクである記憶装置を用いたが、本発明は、光ディスクのような記憶装置でプログラムを受け渡しする構成に限定されず、その他の方法を用いてもよい。例えば、ネットワークを介してプログラムを送受信する構成であってもよい。

[0175] (12)上記第3の実施形態では、プログラムテスト装置40bは、ある初期値に基づきプログラムを実行した場合に、出力値とオリジナルプログラムの実行結果とが一致するか否か判断し、両者が一致すると判断された場合を「正常動作」とみなし、両者が一致しないと判断された場合を「異常動作」とみなす構成を有しているが、本発明において、プログラムが正常動作するか又は異常動作するかの判断方法は、これに限定されない。

[0176] 例えば、プログラムテスト装置は、各初期値に基づきプログラムを実行し、実行開始から実行終了までの時間である実行時間を測定する。プログラムテスト装置は、測定した実行時間が所定時間以上であれは異常動作とみなし、実行時間が所定時間以下であれば正常動作とみなすように構成してもよい。

また、上記実施形態で示した方法と、実行結果を測定する方法とを併用し、正常動

作及び異常動作の判断を行う構成であってもよい。

[0177] (13)上記第2及び第3の実施形態において、プログラム実行装置は、複数の初期値(実行経路パターンリスト)を記憶しておき、実行時に選択した初期値から乱数列を生成する構成を有するが、本発明においては、プログラム実行装置は、予め複数の初期値から複数の乱数列を生成しておき、複数の初期値に代えて、複数の乱数列を記憶しておき、実行時にランダムに乱数列を選択する構成であってもよい。

[0178] 更に、上記のように複数の乱数列を記憶しておく場合には、プログラム実行装置が乱数列を生成する構成ではなく、第2の実施形態においてはプログラム生成装置が、第3の実施形態においてはプログラムテスト装置が乱数列を生成する構成であってもよい。

(14)上記第2及び第3の実施形態において、プログラム実行装置は、実行経路パターンリストから、ランダムに初期値を選択する構成を有するが、本発明において、この構成は必須ではなく、リストから昇順で、又は降順で初期値を選択する構成であってもよい。

[0179] 更に、上記(13)にように、プログラム実行装置が複数の乱数列を記憶している場合であっても、同様に、ランダムに乱数列を選択する構成は、必須ではない。

(15)上記実施形態において、プログラム生成装置は、予めオリジナルプログラムを内部に記憶している構成を有するが、本発明においてこの構成は必須ではなく、プログラム生成装置が、外部からオリジナルプログラムを取得し、取得したオリジナルプログラムから、難読化プログラムであるプログラム実行形式を生成する構成であっても本発明に含まれる。

[0180] (16)本発明は、上記に示す方法であるとしてもよい。また、これらの方法をコンピュータにより実現するコンピュータプログラムであるとしてもよいし、前記コンピュータプログラムからなるデジタル信号であるとしてもよい。

また、本発明は、前記コンピュータプログラム又は前記デジタル信号をコンピュータ読み取り可能な記録媒体、例えば、フレキシブルディスク、ハードディスク、CD-ROM、MO、DVD、DVD-ROM、DVD-RAM、BD(Blu-ray Disc)、半導体メモリなど、に記録したものとしてもよい。また、これらの記録媒体に記録されている前記コ

ンピュータプログラム又は前記デジタル信号であるとしてもよい。

[0181] また、本発明は、前記コンピュータプログラム又は前記デジタル信号を、電気通信回線、無線又は有線通信回線、インターネットを代表とするネットワーク等を経由して伝送するものとしてもよい。

また、本発明は、マイクロプロセッサとメモリとを備えたコンピュータシステムであって、前記メモリは、上記コンピュータプログラムを記憶しており、前記マイクロプロセッサは、前記コンピュータプログラムに従って動作するとしてもよい。

[0182] また、前記プログラム又は前記デジタル信号を前記記録媒体に記録して移送することにより、又は前記プログラム又は前記デジタル信号を前記ネットワーク等を経由して移送することにより、独立した他のコンピュータシステムにより実施するとしてもよい。

(17)また本発明は、上記実施形態におけるプログラム生成装置10、10a、及び10b、プログラム実行装置20、20a、及び20b、並びにプログラムテスト装置40bの機能ブロックの一部又は全てが集積回路であるLSIとして実現される場合も本発明に含まれる。これらは個別に1チップ化されても良いし、一部又は全てを含むように1チップ化されてもよい。ここでは、LSIとしたが、集積度の違いにより、IC、システムLSI、スーパーLSI、ウルトラLSIと呼称されることもある。

[0183] また、集積回路化の手法はLSIに限るものではなく、専用回路又は汎用プロセッサで実現してもよい。LSI製造後に、プログラムすることが可能なFPGA(Field Programmable Gate Array)やLSI内部の回路セルの接続や設定を再構成可能なリコンフィギュラブル・プロセッサを利用しても良い。

更には、半導体技術の進歩又は派生する別技術によりLSIに置き換わる集積回路化の技術が登場すれば、当然その技術を用いて機能ブロックの集積化を行ってもよい。バイオ技術の適応などが可能性として有り得る。

[0184] (18)上記実施形態及び上記変形例をそれぞれ組み合わせる構成も本発明に含まれる。

産業上の利用可能性

[0185] 本発明は、正当な権利を有する者以外のユーザに解析されたくないプログラムを搭

載した装置を製造、販売する産業において、ユーザに解析されにくいプログラムの実行形式を生成する仕組みとして利用することができる。

請求の範囲

[1] ある1の結果を得ることができるように、1以上の命令を所定の順序で実行させる第1プログラムを取得する取得手段と、

前記第1プログラムに基づき第2プログラムを生成する生成手段と、

前記第2プログラムを出力する出力手段とを備え、

前記第2プログラムは、

当該第2プログラムの実行時に決定されるカレント情報に応じて、結果を得るために実行させる処理が変化し、かつ、得られる前記結果が、前記カレント情報に応じた処理の変化に関わらず、前記第1プログラムにより得られる結果と同一となるプログラムである

ことを特徴とするプログラム生成装置。

[2] 前記生成手段は、

前記第1プログラムに含まれる命令のうち、相互に依存関係にない命令である複数の並列命令を検出する検出部を備え、

前記第1プログラムが実行させる処理のうち、前記複数の並列命令を前記所定の順序で実行させる処理に替えて、前記カレント情報に応じて変化する順序で実行させる前記第2プログラムを生成する

ことを特徴とする請求項1に記載のプログラム生成装置。

[3] 前記生成手段は、

前記第1プログラムに含まれる1以上の命令であるオリジナル命令と異なる処理を行い、かつ、前記オリジナル命令と同一の結果を出力する1以上の命令からなる恒等命令を1以上生成する恒等命令生成部を備え、

前記第1プログラムが実行させる処理のうち、前記オリジナル命令を実行させる処理に替えて、前記カレント情報に応じて、前記オリジナル命令、又は生成された前記恒等命令のうちから1を選択させ、実行させる前記第2プログラムを生成する

ことを特徴とする請求項1に記載のプログラム生成装置。

[4] 前記生成手段は、

前記結果に影響を与えない命令であるダミー命令を生成するダミー命令生成部を

備え、

前記第1プログラムと同一の結果を得るために実行させる処理に加えて、前記カレント情報に応じて変化するタイミングで、前記ダミー命令を実行させる前記第2プログラムを生成する

ことを特徴とする請求項1に記載のプログラム生成装置。

[5] 前記生成手段は、

前記第1プログラムに含まれる命令のうち、相互に依存関係にない命令である複数の並列命令を検出する検出部と、

前記第1プログラムに含まれる1以上の命令であるオリジナル命令と異なる処理を行い、かつ、前記オリジナル命令と同一の結果を出力する1以上の命令からなる恒等命令を1以上生成する恒等命令生成部と、

前記結果に影響を与えない命令であるダミー命令を生成するダミー命令生成部とを備え、

前記第1プログラムが実行させる処理のうち、前記複数の並列命令を前記所定の順序で実行させる処理に替えて、前記カレント情報に応じて変化する順序で実行させ、

前記第1プログラムが実行させる処理のうち、前記オリジナル命令を実行させる処理に替えて、前記カレント情報に応じて、前記オリジナル命令又は生成された前記恒等命令のうちから1を選択させ、実行させ、

当該カレント情報に応じて変化するタイミングで、前記ダミー命令を実行させる前記第2プログラムを生成する

ことを特徴とする請求項1に記載のプログラム生成装置。

[6] 前記生成手段は、更に、

前記第1プログラムに含まれる命令、恒等命令生成部により生成された恒等命令、及びダミー命令生成部により生成されたダミー命令のそれぞれに、各命令を一意に識別する識別子を付与する識別子付与手段を備え、

前記カレント情報に応じて生成される乱数列の各要素と一致する識別子により識別される命令を呼び出す前記第2プログラムを生成する

ことを特徴とする請求項5に記載のプログラム生成装置。

[7] 前記生成手段は、

前記カレント情報に応じて前記乱数列を生成する乱数列生成アルゴリズムを含む前記第2プログラムを生成する

ことを特徴とする請求項6に記載のプログラム生成装置。

[8] 前記生成手段は、

前記カレント情報に応じて乱数列を生成する複数の乱数列生成アルゴリズムを含み、実行時に前記複数の乱数列生成アルゴリズムから1の乱数列生成アルゴリズムを選択し、選択した乱数列生成アルゴリズムに従い生成される乱数列の各要素と一致する識別子により識別される命令を呼び出す前記第2プログラムを生成する

ことを特徴とする請求項6に記載のプログラム生成装置。

[9] 前記生成手段は、更に、

前記第1プログラムに含まれる命令、恒等命令生成部により生成された恒等命令、及びダミー命令生成部により生成されたダミー命令のそれぞれを関数化し、複数の関数からなる関数データを生成する関数データ生成部と、

前記関数データ生成部により生成されたそれぞれの関数に、各関数を一意に識別する識別子を付与する識別子付与部と、

前記カレント情報に応じて乱数列を生成する乱数列生成アルゴリズムを含み、生成される乱数列の各要素と一致する識別子により識別される関数のうち、前記結果と同一の結果を得るための条件に従う関数の呼び出し、及び実行を指示する呼出プログラムを生成する呼出プログラム生成部とを備え、

前記関数データ、及び前記呼出プログラムを含む前記第2プログラムを生成する

ことを特徴とする請求項5に記載のプログラム生成装置。

[10] 前記複数の関数は、それぞれ、並列グループ、シーケンシャルグループ、恒等グループ、及びダミーグループの何れか1以上のグループに属し、

前記呼出プログラム生成部は、

各関数が属するグループを判断することにより、前記条件に従う順序で、各関数を呼び出すことを指示する前記呼出プログラムを生成する

ことを特徴とする請求項9に記載のプログラム生成装置。

[11] 前記プログラム生成装置は、更に、

前記カレント情報の候補となる候補情報を複数個生成する候補情報生成手段を備え、

前記出力手段は、前記第2プログラムと共に、適切な複数個の候補情報を出力する

ことを特徴とする請求項1に記載のプログラム生成装置。

[12] 前記プログラム生成装置は、更に、

生成された前記複数個の候補情報のそれぞれについて、各候補情報に応じて行う動作が、正しい動作であるか否か判断するテスト手段を備え、

前記出力手段は、前記テスト手段により正しい動作を行うと判断された候補情報のみを抽出し、出力する

ことを特徴とする請求項11に記載のプログラム生成装置。

[13] 前記テスト手段は、

前記第1プログラムを実行して得られる第1値を取得する第1取得部と、

1の候補情報に応じた動作を行い得られる第2値を取得する第2取得部と、

前記第1値と前記第2値とを比較し、両者が一致する場合に、前記1の候補情報に応じて行う動作が、正しい動作であると判断する比較判断部とを備える

ことを特徴とする請求項12に記載のプログラム生成装置。

[14] 実行時に決定されるカレント情報に応じて異なる動作を行い、かつ、前記動作のそれぞれによって得られる結果は、前記カレント情報に関わらず一定である

ことを特徴とするコンピュータプログラム。

[15] 請求項1に記載のプログラム生成装置により生成された前記第2プログラムを実行するプログラム実行装置であって、

前記第2プログラムを取得する取得手段と、

実行時に、前記カレント情報を指定する情報指定手段と、

前記第2プログラムを実行する実行手段と

を備えることを特徴とするプログラム実行装置。

[16] 前記情報指定手段は、乱数を生成し、生成した乱数を前記カレント情報とする

ことを特徴とする請求項15に記載のプログラム実行装置。

[17] 前記取得手段は、前記カレント情報の候補となる複数の候補情報を取得し、

前記情報指定手段は、前記複数の候補情報から1の候補情報を選択する

ことを特徴とする請求項15に記載のプログラム実行装置。

[18] 前記情報指定手段は、乱数を生成し、生成した乱数を用いて前記複数の候補情報から1の候補情報を選択する

ことを特徴とする請求項17に記載のプログラム実行装置。

[19] 前記第2プログラムを複数回実行する前記プログラム実行装置において、

前記情報指定手段は、選択済みの候補情報を記憶する記憶部と、

前記記憶部を参照することにより、未選択の候補情報を選択するよう制御する制御部とを備える

ことを特徴とする請求項17に記載のプログラム実行装置。

[20] 前記プログラム実行装置は、更に、

前記第1プログラムを実行して得られる第1値を取得する第1値取得手段と、

前記実行手段により実行された第2プログラムの実行結果である第2値と、前記第1値とを比較し、両者が一致するか否か判断する比較判断手段と、

前記比較判断手段により、前記第1値と前記第2値とが一致すると判断された場合に、前記情報指定手段により指定された前記候補情報を記録する記録手段と

を備えることを特徴とする請求項15に記載のプログラム実行装置。

[21] 前記プログラム実行装置は、更に、

前記情報指定手段、前記実行手段、前記比較判断手段及び前記記録手段による処理を複数回行い、

前記記録手段により記録された複数個の候補情報を含むリストを生成し、生成したリストを出力する

ことを特徴とする請求項20に記載のプログラム実行装置。

[22] プログラム生成装置及びプログラム実行装置から構成される情報処理システムであって、

前記プログラム生成装置は、

ある1の結果を得ることができるように、1以上の命令を所定の順序で実行させる第1プログラムを取得する取得手段と、

前記第1プログラムに基づき第2プログラムを生成する生成手段と、

前記第2プログラムを出力する出力手段とを備え、

前記第2プログラムは、当該第2プログラムの実行時に決定されるカレント情報に応じて、結果を得るために実行させる処理が変化し、かつ、得られる前記結果が、前記カレント情報に応じた処理の変化に関わらず、前記第1プログラムにより得られる結果と同一となるプログラムであって、

前記プログラム実行装置は、

前記第2プログラムを取得する第2取得手段と、

前記カレント情報を指定する情報指定手段と、

前記第2プログラムを実行する実行手段とを備える

ことを特徴とする情報処理システム。

[23] 前記プログラム生成装置において、

前記生成手段は、前記カレント情報の候補となる複数個の候補情報を生成し、

前記出力手段は、前記第2プログラムと共に、前記複数個の候補情報を出力し、

前記プログラム実行装置において、

前記取得手段は、前記複数の候補情報を取得し、

前記情報指定手段は、前記複数の候補情報から1の候補情報を選択する

ことを特徴とする請求項22に記載の情報処理システム。

[24] プログラム生成装置で用いられるプログラム生成方法であって、

ある1の結果を得ることができるように、1以上の命令を所定の順序で実行させる第1プログラムを取得する取得ステップと、

前記第1プログラムに基づき第2プログラムを生成する生成ステップと、

前記第2プログラムを出力する出力ステップとを含み、

前記第2プログラムは、

当該第2プログラムの実行時に決定されるカレント情報に応じて、結果を得るために実行させる処理が変化し、かつ、得られる前記結果が、前記カレント情報に応じた処

理の変化に関わらず、前記第1プログラムにより得られる結果と同一となるプログラムである

ことを特徴とするプログラム生成方法。

[25] プログラム生成装置で用いられるコンピュータプログラムであって、

ある1の結果を得ることができるように、1以上の命令を所定の順序で実行させる第1プログラムを取得する取得ステップと、

前記第1プログラムに基づき第2プログラムを生成する生成ステップと、

前記第2プログラムを出力する出力ステップとを含み、

前記第2プログラムは、

当該第2プログラムの実行時に決定されるカレント情報に応じて、結果を得るために実行させる処理が変化し、かつ、得られる前記結果が、前記カレント情報に応じた処理の変化に関わらず、前記第1プログラムにより得られる結果と同一となるプログラムである

ことを特徴とするコンピュータプログラム。

[26] プログラム生成装置で用いられるコンピュータプログラムを記録しているコンピュータ読み取り可能な記録媒体であって、

前記コンピュータプログラムは、

ある1の結果を得ることができるように、1以上の命令を所定の順序で実行させる第1プログラムを取得する取得ステップと、

前記第1プログラムに基づき第2プログラムを生成する生成ステップと、

前記第2プログラムを出力する出力ステップとを含み、

前記第2プログラムは、

当該第2プログラムの実行時に決定されるカレント情報に応じて、結果を得るために実行させる処理が変化し、かつ、得られる前記結果が、前記カレント情報に応じた処理の変化に関わらず、前記第1プログラムにより得られる結果と同一となるプログラムである

ことを特徴とする記録媒体。

[27] ある1の結果を得ることができるように、1以上の命令を所定の順序で実行させる第

1プログラムを取得する取得手段と、

前記第1プログラムに基づき第2プログラムを生成する生成手段と、

前記第2プログラムを出力する出力手段とを備え、

前記第2プログラムは、

当該第2プログラムの実行時に決定されるカレント情報に応じて、結果を得るために実行させる処理が変化し、かつ、得られる前記結果が、前記カレント情報に応じた処理の変化に関わらず、前記第1プログラムにより得られる結果と同一となるプログラムである

ことを特徴とする集積回路。

[28] 請求項1に記載のプログラム生成装置により生成された前記第2プログラムを実行するプログラム実行装置で用いられるプログラム実行方法であって、

前記第2プログラムを取得する取得ステップと、

実行時に、前記カレント情報を指定する情報指定ステップと、

前記第2プログラムを実行する実行ステップと

を含むことを特徴とするプログラム実行方法。

[29] 請求項1に記載のプログラム生成装置により生成された前記第2プログラムを実行するプログラム実行装置で用いられるコンピュータプログラムであって、

前記第2プログラムを取得する取得ステップと、

実行時に、前記カレント情報を指定する情報指定ステップと、

前記第2プログラムを実行する実行ステップと

を含むことを特徴とするコンピュータプログラム。

[30] 請求項1に記載のプログラム生成装置により生成された前記第2プログラムを実行するプログラム実行装置で用いられるコンピュータプログラムを記録しているコンピュータ読み取り可能な記録媒体であって、

前記コンピュータプログラムは、

前記第2プログラムを取得する取得ステップと、

実行時に、前記カレント情報を指定する情報指定ステップと、

前記第2プログラムを実行する実行ステップと

を含むことを特徴とする記録媒体。

[31] 請求項1に記載のプログラム生成装置により生成された前記第2プログラムを実行する集積回路であって、

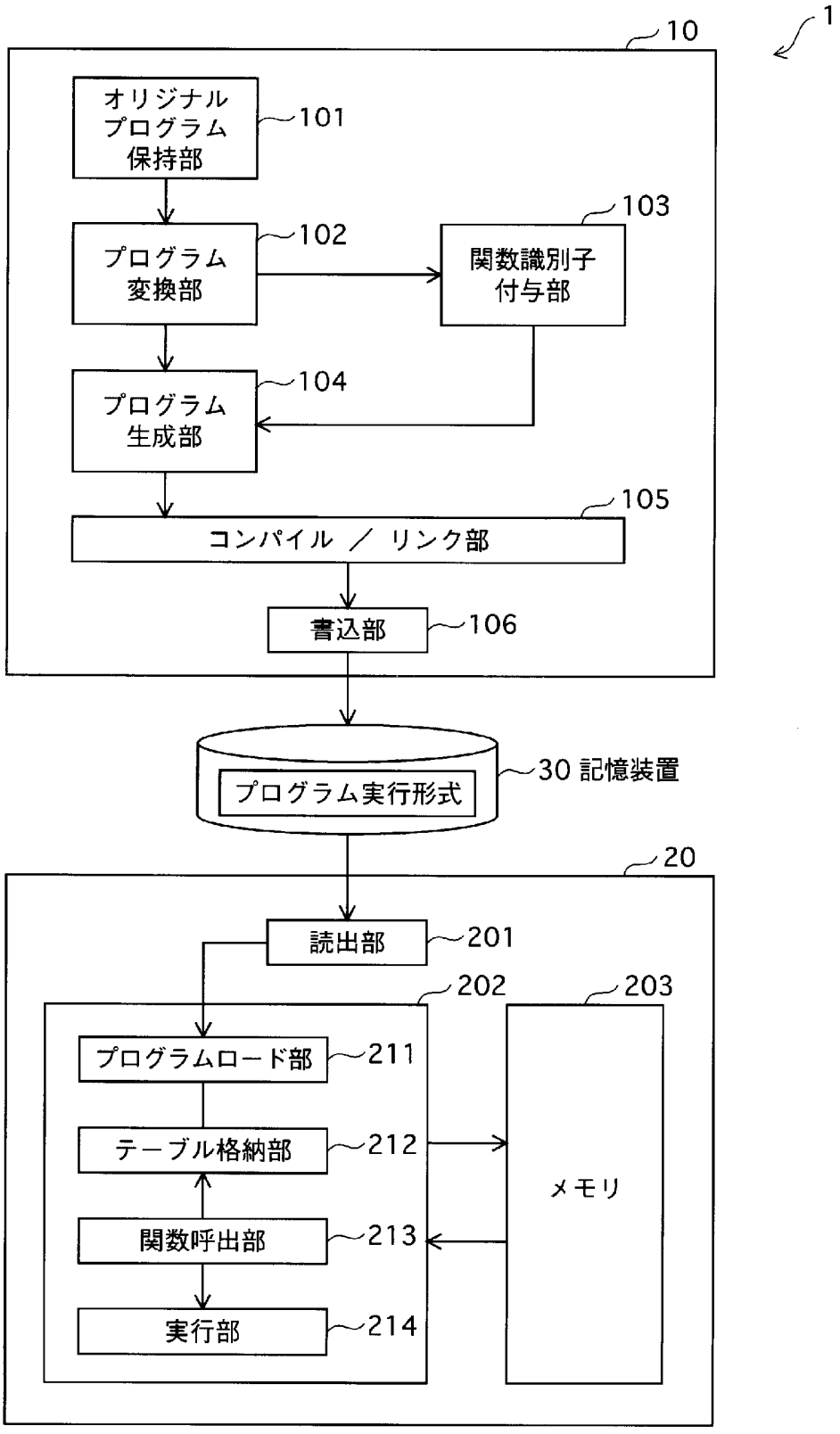
前記第2プログラムを取得する取得手段と、

実行時に、前記カレント情報を指定する情報指定手段と、

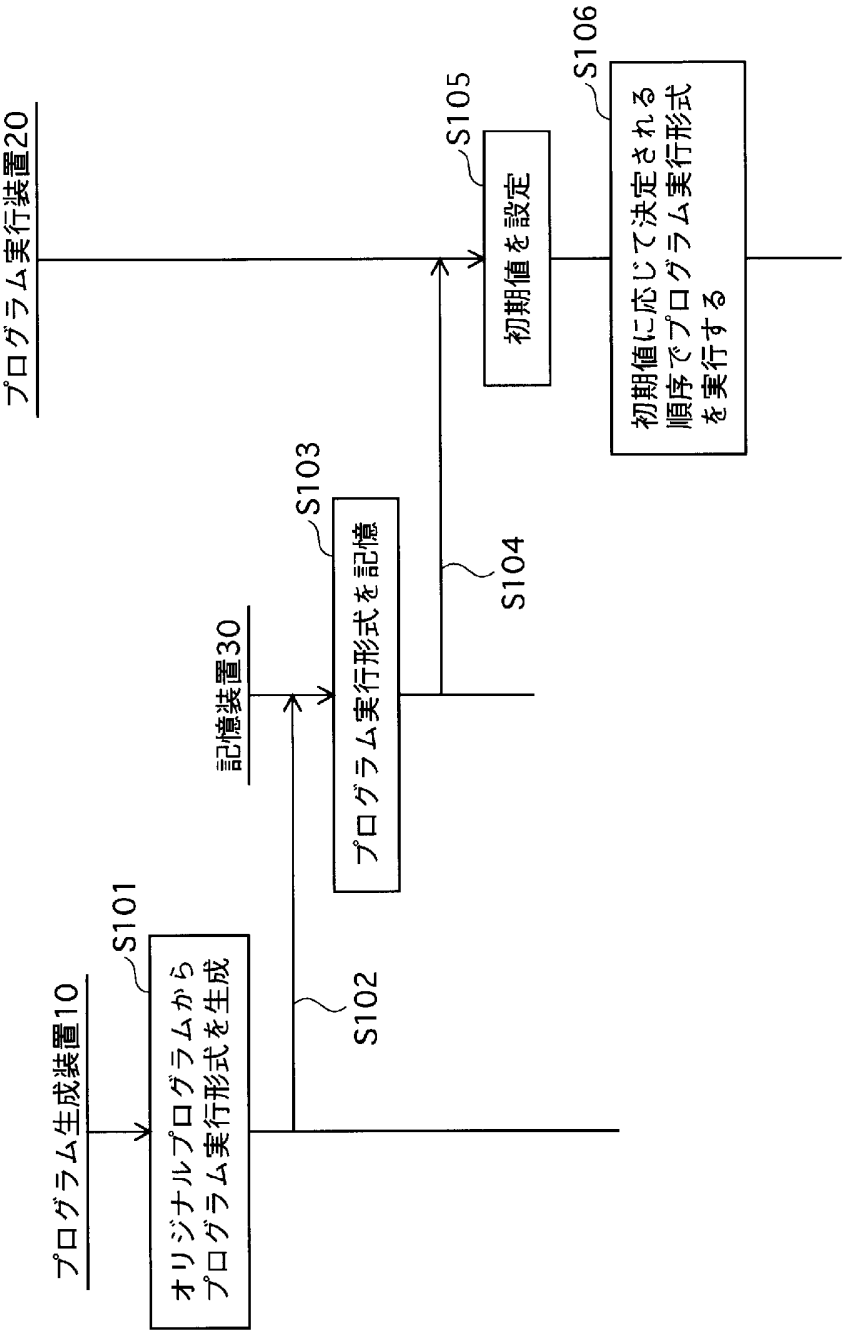
前記第2プログラムを実行する実行手段と

を備えることを特徴とする集積回路。

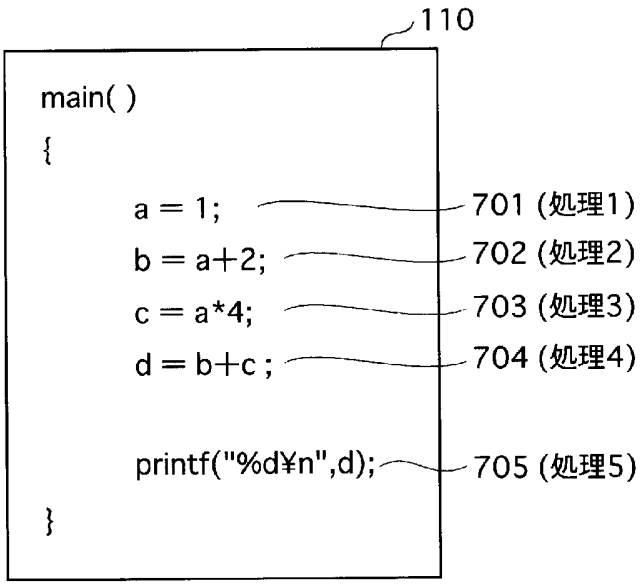
[図1]


1
10
オリジナル
プログラム
保持部
101
プログラム
変換部
102
関数識別子
付与部
103
プログラム
生成部
104
コンパイル ／ リンク部
105
書込部
106
プログラム実行形式
30 記憶装置
20
読出部
201
202
203
プログラムロード部
211
テーブル格納部
212
関数呼出部
213
実行部
214
メモリ

[図2]

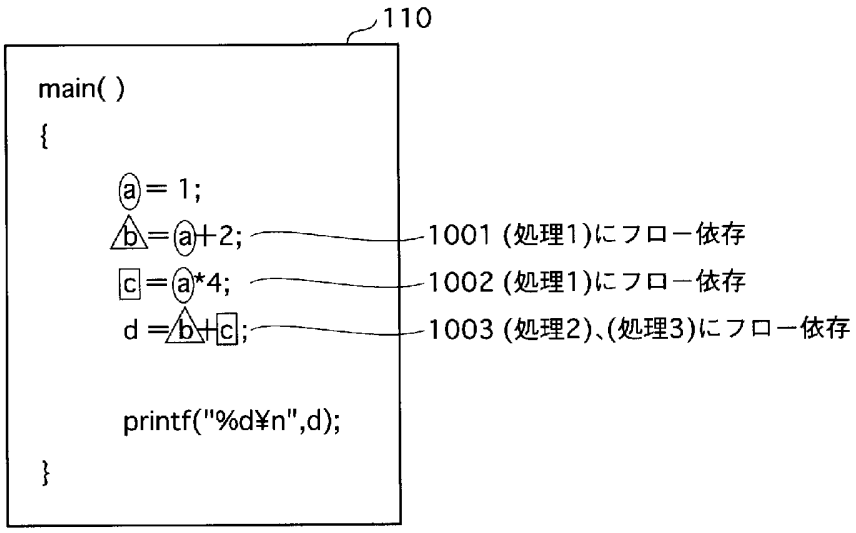

プログラム生成装置10
S101
オリジナルプログラムから
プログラム実行形式を生成
S102
記憶装置30
S103
プログラム実行形式を記憶
S104
プログラム実行装置20
S105
初期値を設定
S106
初期値に応じて決定される
順序でプログラム実行形式
を実行する

[図3]

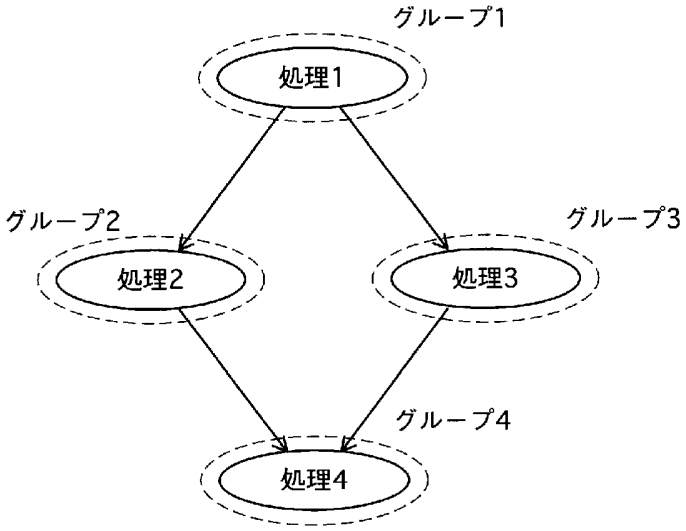

110
main( )
{
a = 1;
b = a+2;
c = a*4;
d = b+c ;
printf("%d¥n",d);
}
701 (処理1)
702 (処理2)
703 (処理3)
704 (処理4)
705 (処理5)

[図4]

(a)

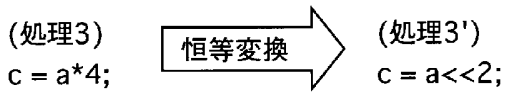

110
main( )
{
a = 1;
b = a + 2;
c = a * 4;
d = b + c;
printf("%d¥n",d);
}
1001 (処理1)にフロー依存
1002 (処理1)にフロー依存
1003 (処理2)、(処理3)にフロー依存

(b)

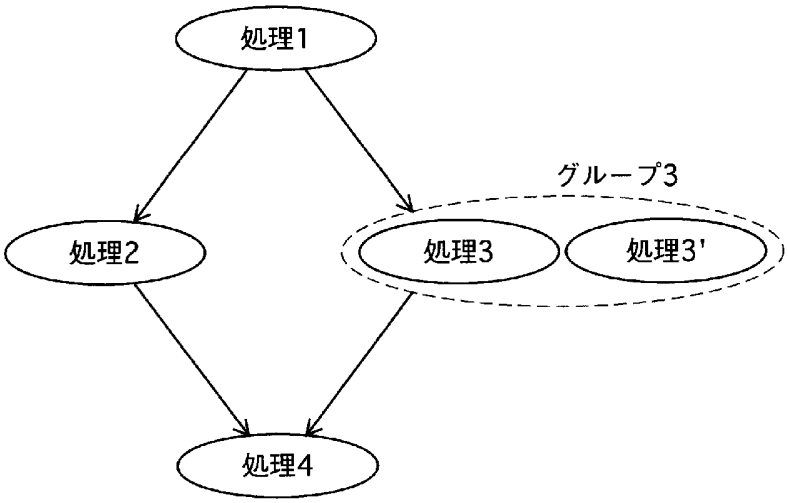

グループ1
処理1
グループ2
処理2
グループ3
処理3
グループ4
処理4

[図5]

(a)

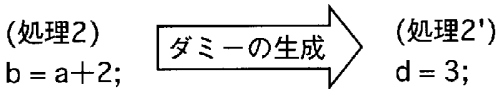


(b)

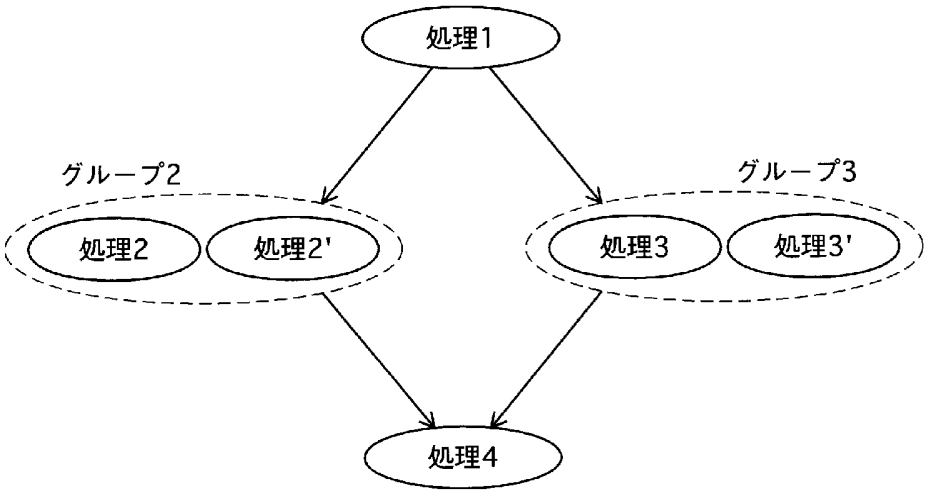


[図6]

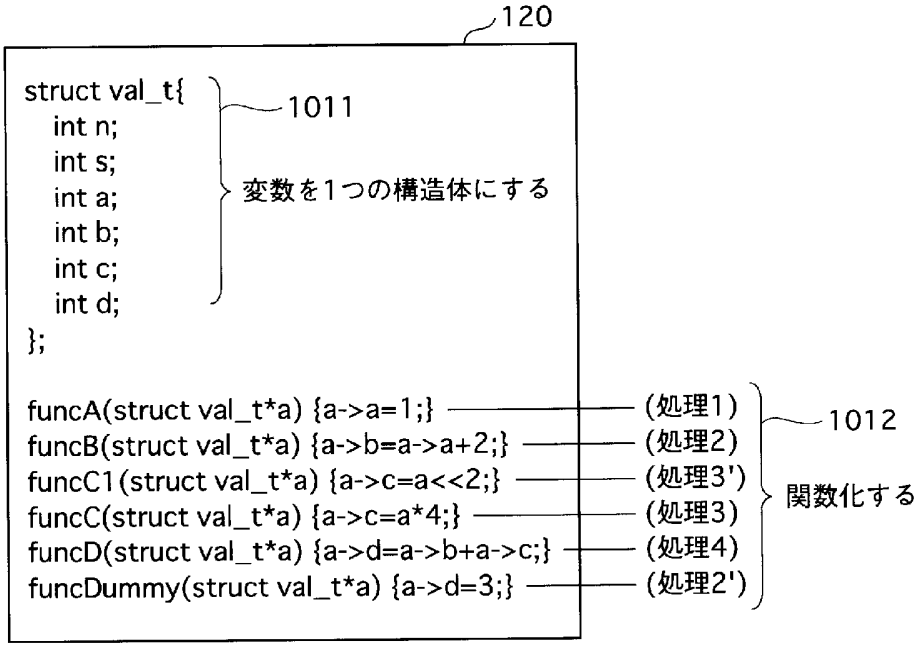
(a)



(b)



[図7]

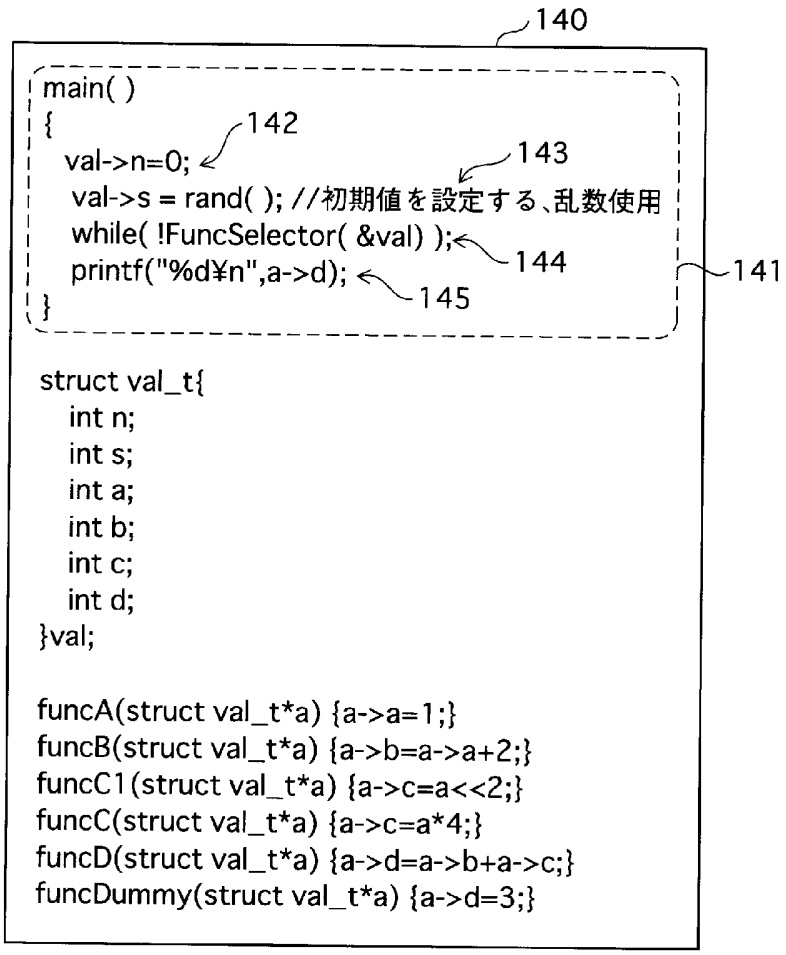

120
struct val_t{
int n;
int s;
int a;
int b;
int c;
int d;
};
1011
変数を1つの構造体にする
funcA(struct val_t*a) {a->a=1;}
funcB(struct val_t*a) {a->b=a->a+2;}
funcC1(struct val_t*a) {a->c=a<<2;}
funcC(struct val_t*a) {a->c=a*4;}
funcD(struct val_t*a) {a->d=a->b+a->c;}
funcDummy(struct val_t*a) {a->d=3;}
(処理1)
(処理2)
(処理3')
(処理3)
(処理4)
(処理2')
1012
関数化する

[図8]

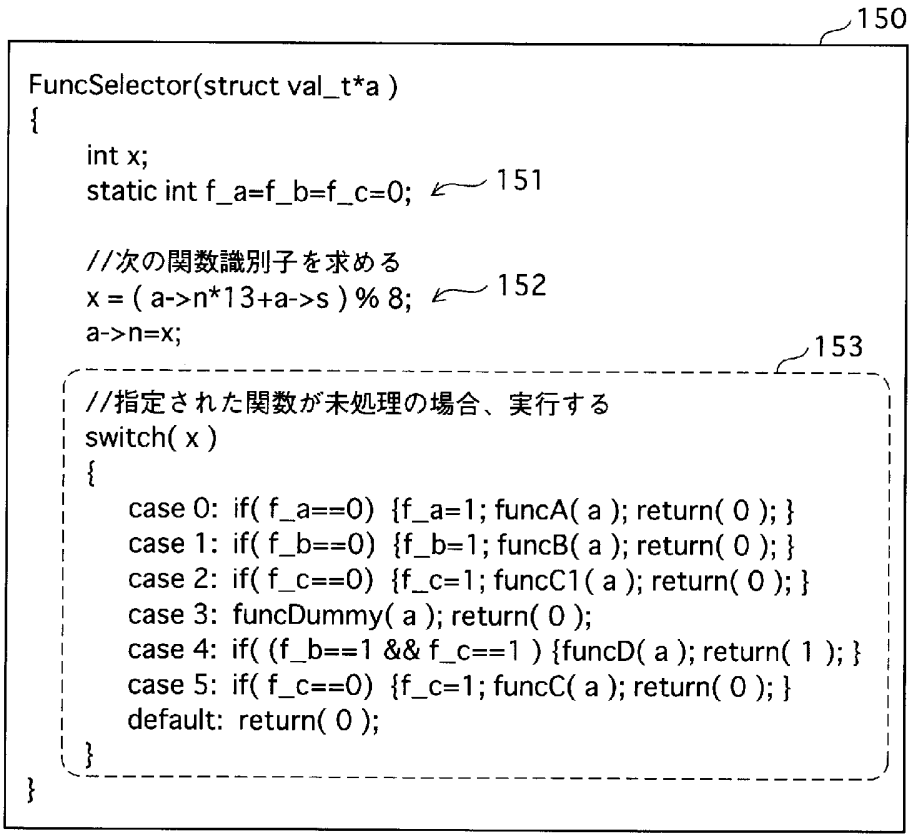
130

関数名	関数識別子
funcA	0
funcB	1
funcC	5
funcC1	2
funcD	4
funcDummy	3

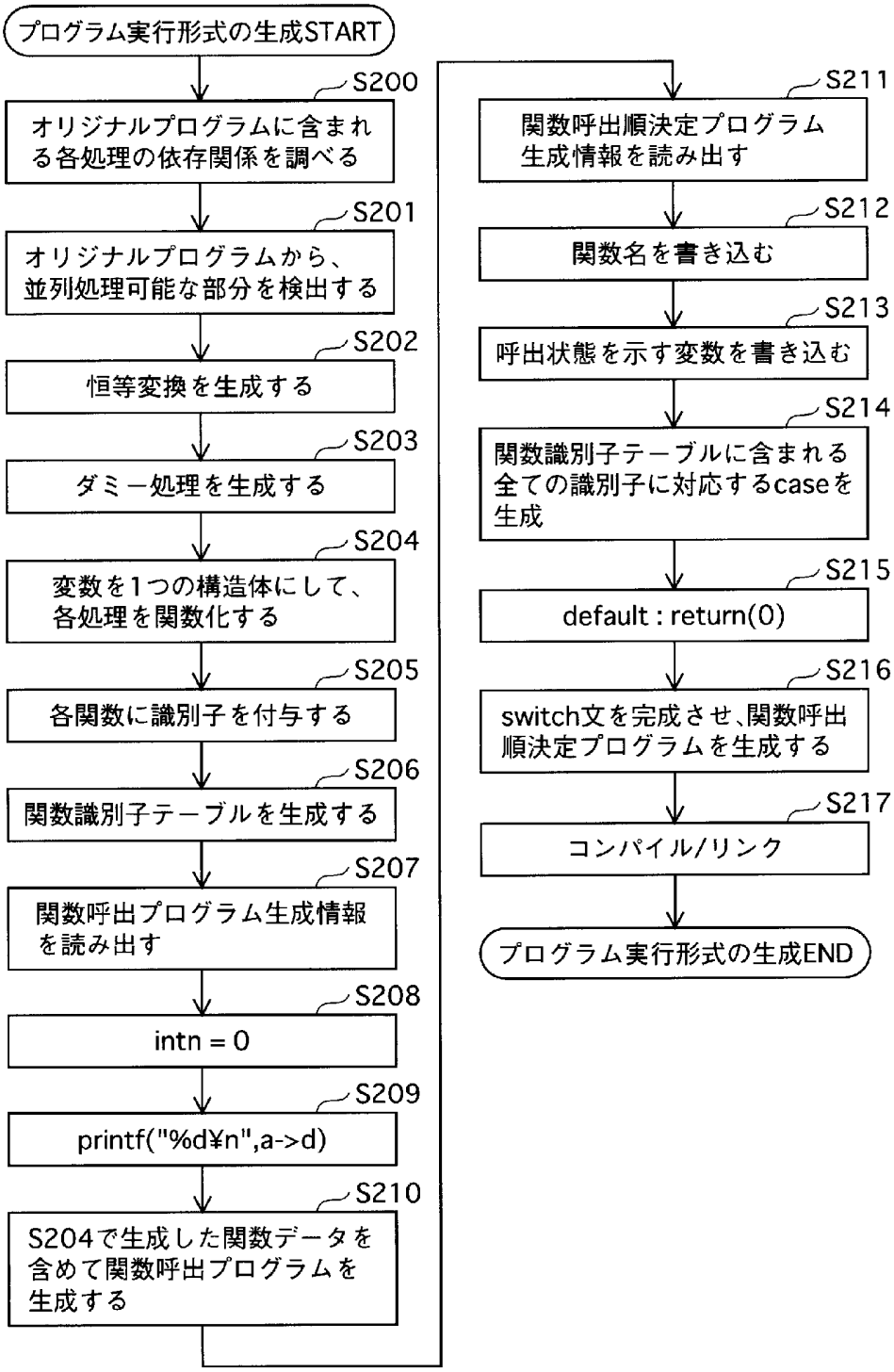
[図9]


140
main( )
{
142
val->n=0;
143
val->s = rand( ); //初期値を設定する、乱数使用
while(!FuncSelector(&val));
144
printf("%d¥n",a->d);
145
}
141
struct val_t{
int n;
int s;
int a;
int b;
int c;
int d;
}val;
funcA(struct val_t*a) {a->a=1;}
funcB(struct val_t*a) {a->b=a->a+2;}
funcC1(struct val_t*a) {a->c=a<<2;}
funcC(struct val_t*a) {a->c=a*4;}
funcD(struct val_t*a) {a->d=a->b+a->c;}
funcDummy(struct val_t*a) {a->d=3;}

[図10]


150
FuncSelector(struct val_t*a)
{
int x;
static int f_a=f_b=f_c=0;
151
//次の関数識別子を求める
x = (a->n*13+a->s) % 8;
152
a->n=x;
153
//指定された関数が未処理の場合、実行する
switch(x)
{
case 0: if(f_a==0) {f_a=1; funcA(a); return(0); }
case 1: if(f_b==0) {f_b=1; funcB(a); return(0); }
case 2: if(f_c==0) {f_c=1; funcC1(a); return(0); }
case 3: funcDummy(a); return(0);
case 4: if((f_b==1 && f_c==1) {funcD(a); return(1); }
case 5: if(f_c==0) {f_c=1; funcC(a); return(0); }
default: return(0);
}
}

[図11]

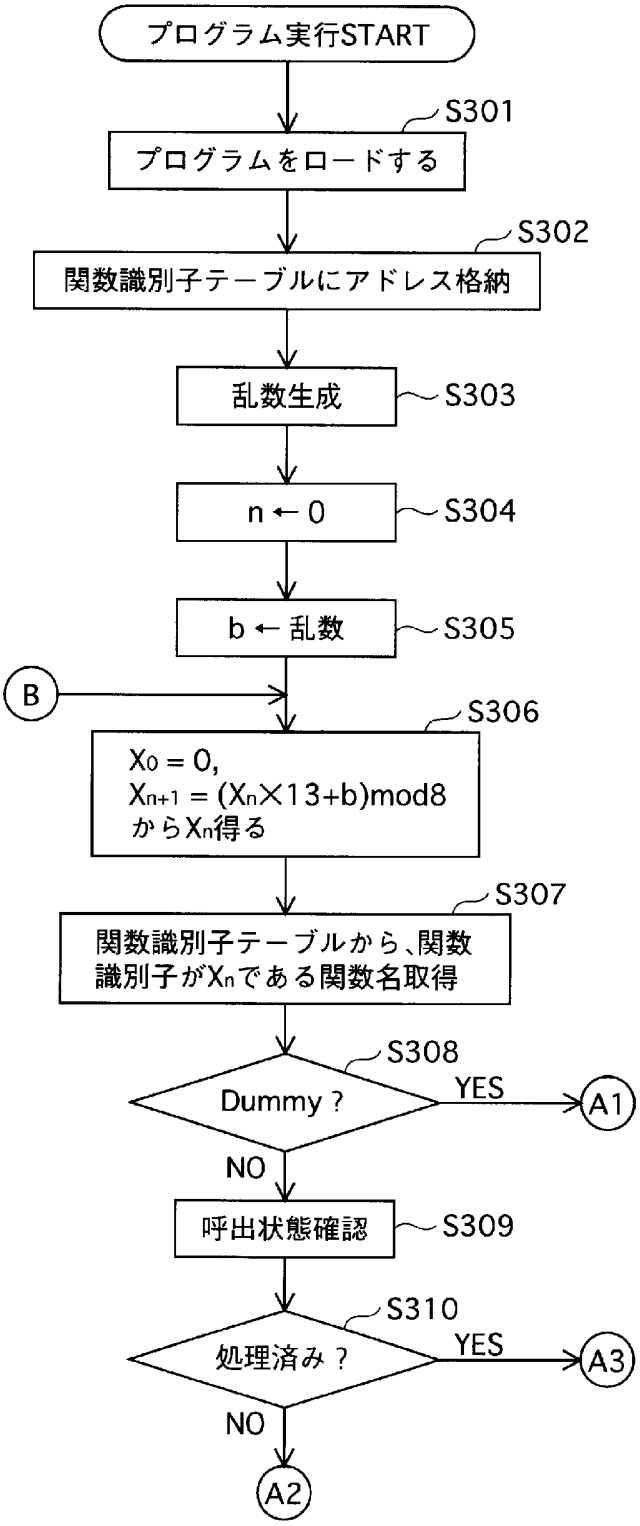

プログラム実行形式の生成START
S200
オリジナルプログラムに含まれる各処理の依存関係を調べる
S201
オリジナルプログラムから、並列処理可能な部分を検出する
S202
恒等変換を生成する
S203
ダミー処理を生成する
S204
変数を1つの構造体にして、各処理を関数化する
S205
各関数に識別子を付与する
S206
関数識別子テーブルを生成する
S207
関数呼出プログラム生成情報を読み出す
S208
intn = 0
S209
printf("%d¥n",a->d)
S210
S204で生成した関数データを含めて関数呼出プログラムを生成する
S211
関数呼出順決定プログラム生成情報を読み出す
S212
関数名を書き込む
S213
呼出状態を示す変数を書き込む
S214
関数識別子テーブルに含まれる全ての識別子に対応するcaseを生成
S215
default : return(0)
S216
switch文を完成させ、関数呼出順決定プログラムを生成する
S217
コンパイル/リンク
プログラム実行形式の生成END

[図12]

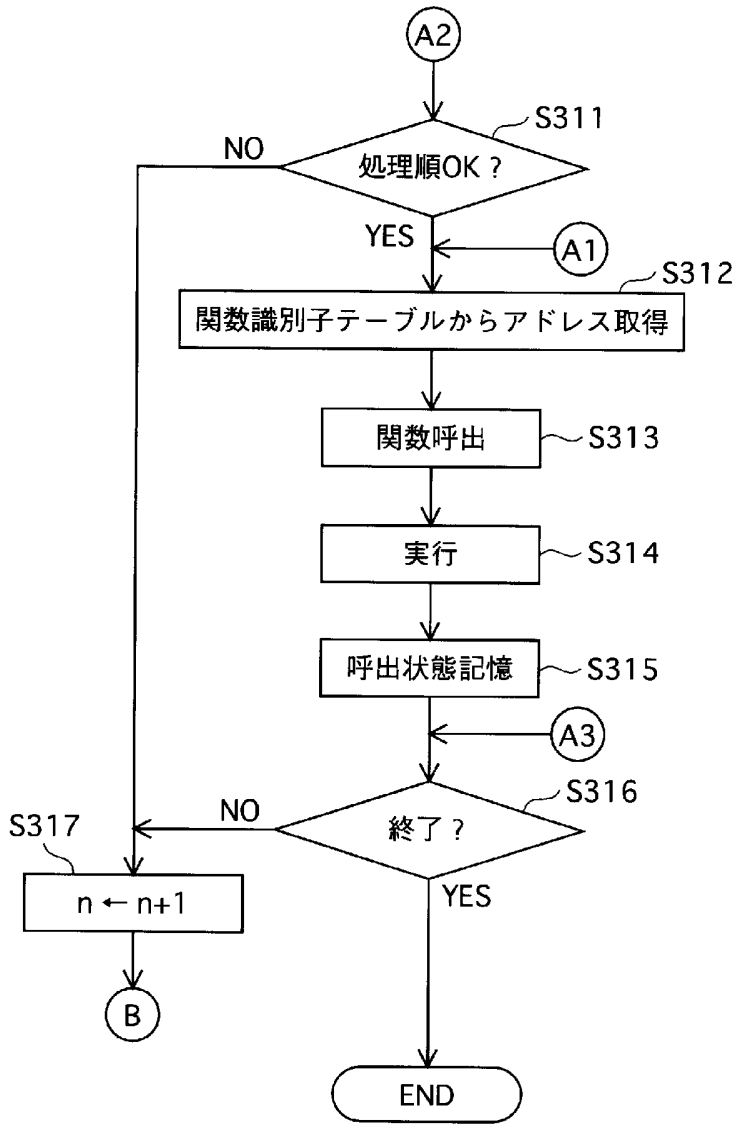
220

関数名	関数識別子	メモリ上の位置
funcA	0	0x07e9500
funcB	1	0x07e9800
funcC	5	0x07e9900
funcC1	2	0x07ea010
funcD	4	0x07eb010
funcDummy	3	0x07eb050

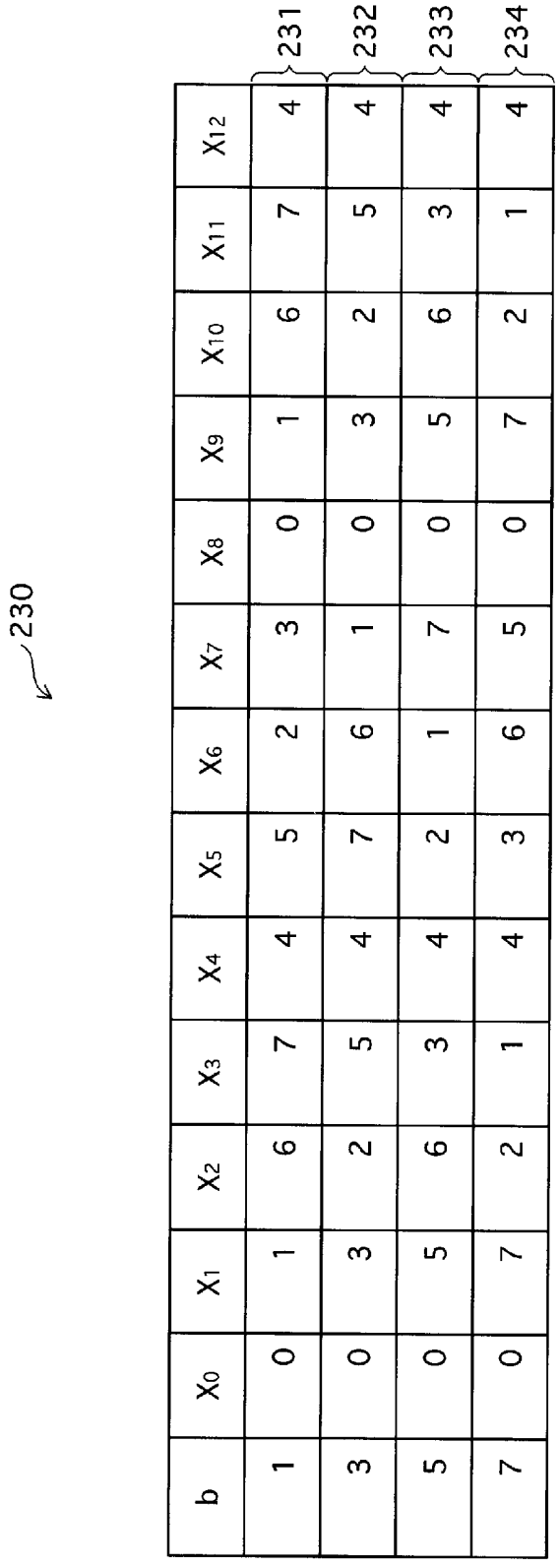
[図13]


プログラム実行START
S301
プログラムをロードする
S302
関数識別子テーブルにアドレス格納
乱数生成
S303
n ← 0
S304
b ← 乱数
S305
B
S306
X0 = 0,
Xn+1 = (Xn×13+b)mod8
からXn得る
S307
関数識別子テーブルから、関数識別子がXnである関数名取得
S308
Dummy ?
YES
A1
NO
呼出状態確認
S309
S310
処理済み ?
YES
A3
NO
A2

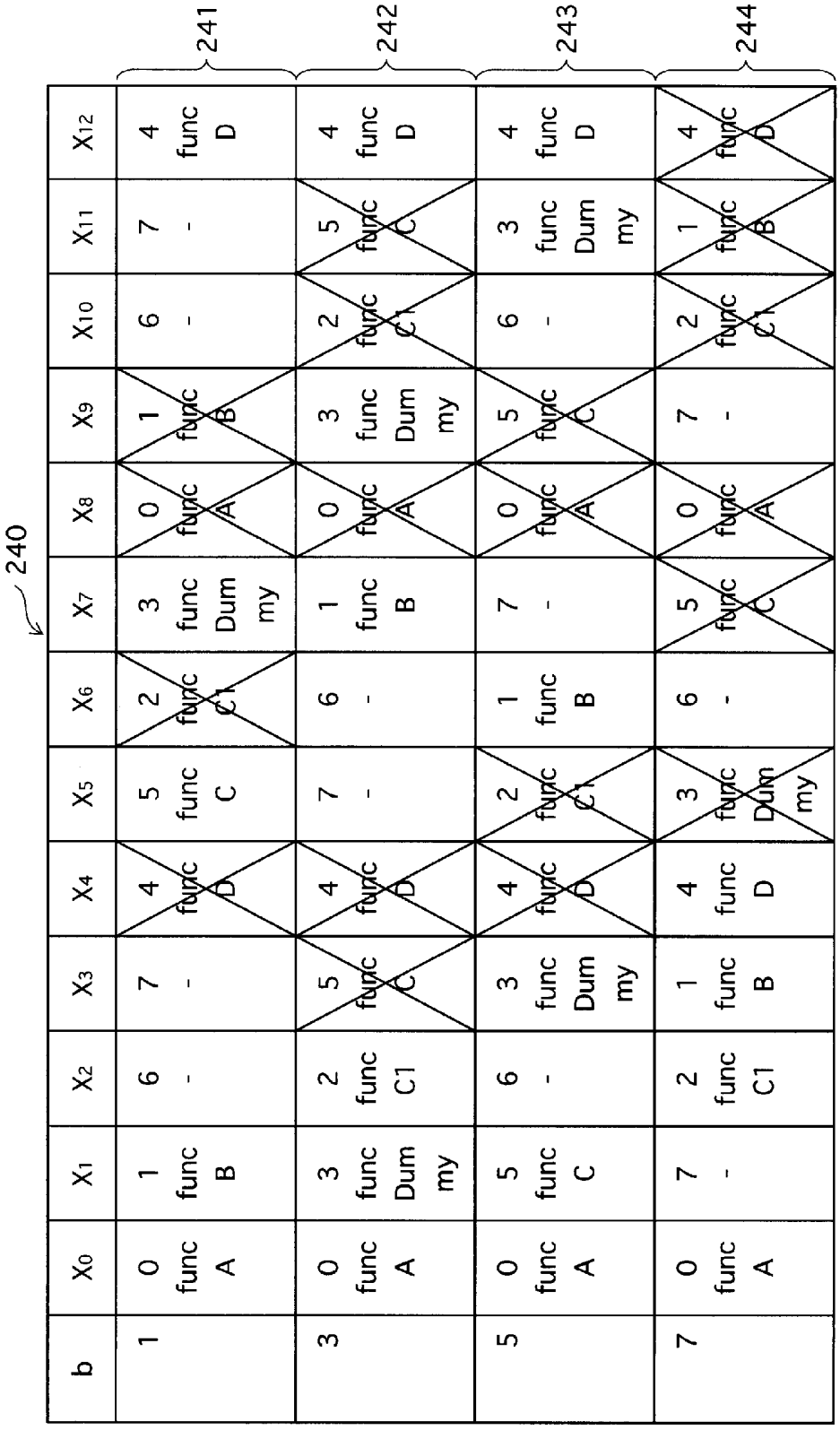
[図14]


A2
S311
NO
処理順OK？
YES
A1
S312
関数識別子テーブルからアドレス取得
関数呼出
S313
実行
S314
呼出状態記憶
S315
A3
S316
S317
NO
終了？
n ← n+1
YES
B
END

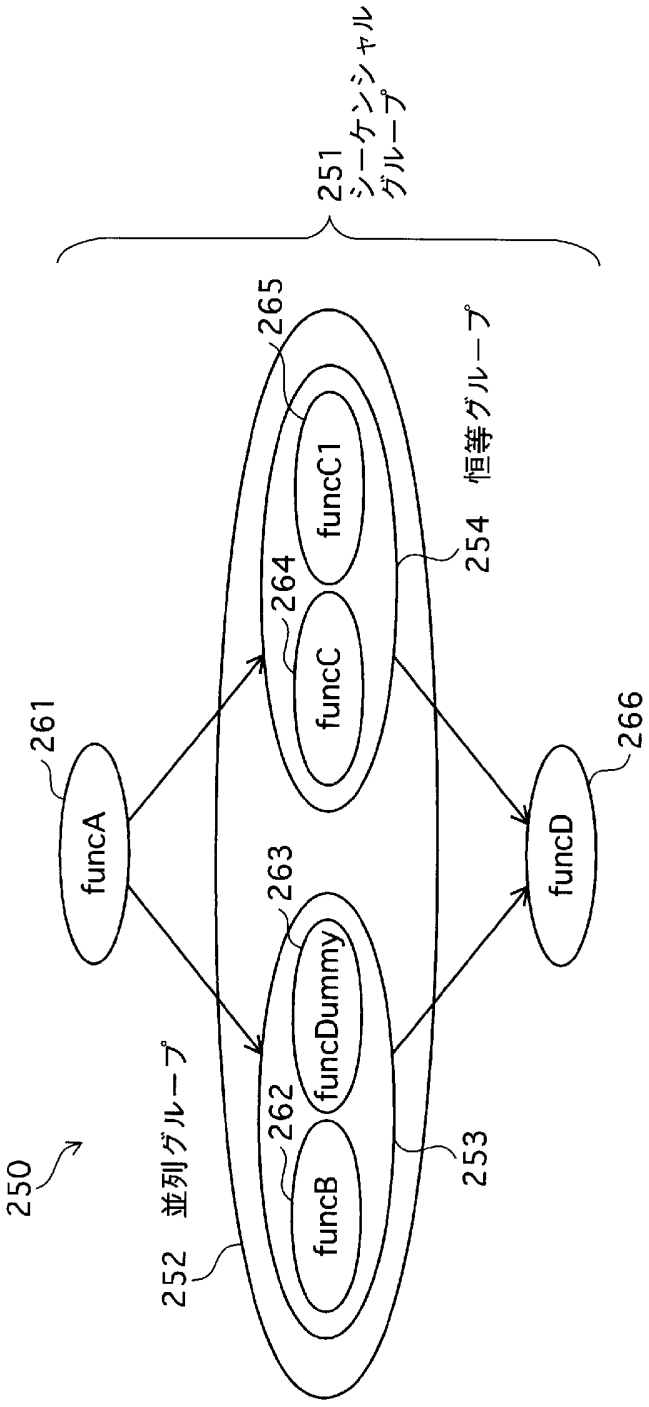
[図15]


230
231
232
233
234
b
X0
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
1 0 1 6 7 4 5 2 3 0 1 6 7 4
3 0 3 2 5 4 7 6 1 0 3 2 5 4
5 0 5 6 3 4 2 1 7 0 5 6 3 4
7 0 7 2 1 4 3 6 5 0 7 2 1 4
Xn+1 = (Xn*a+b)mod m ・・・ (式1), X0 = 0・・・ (式2)
a=13, m=8

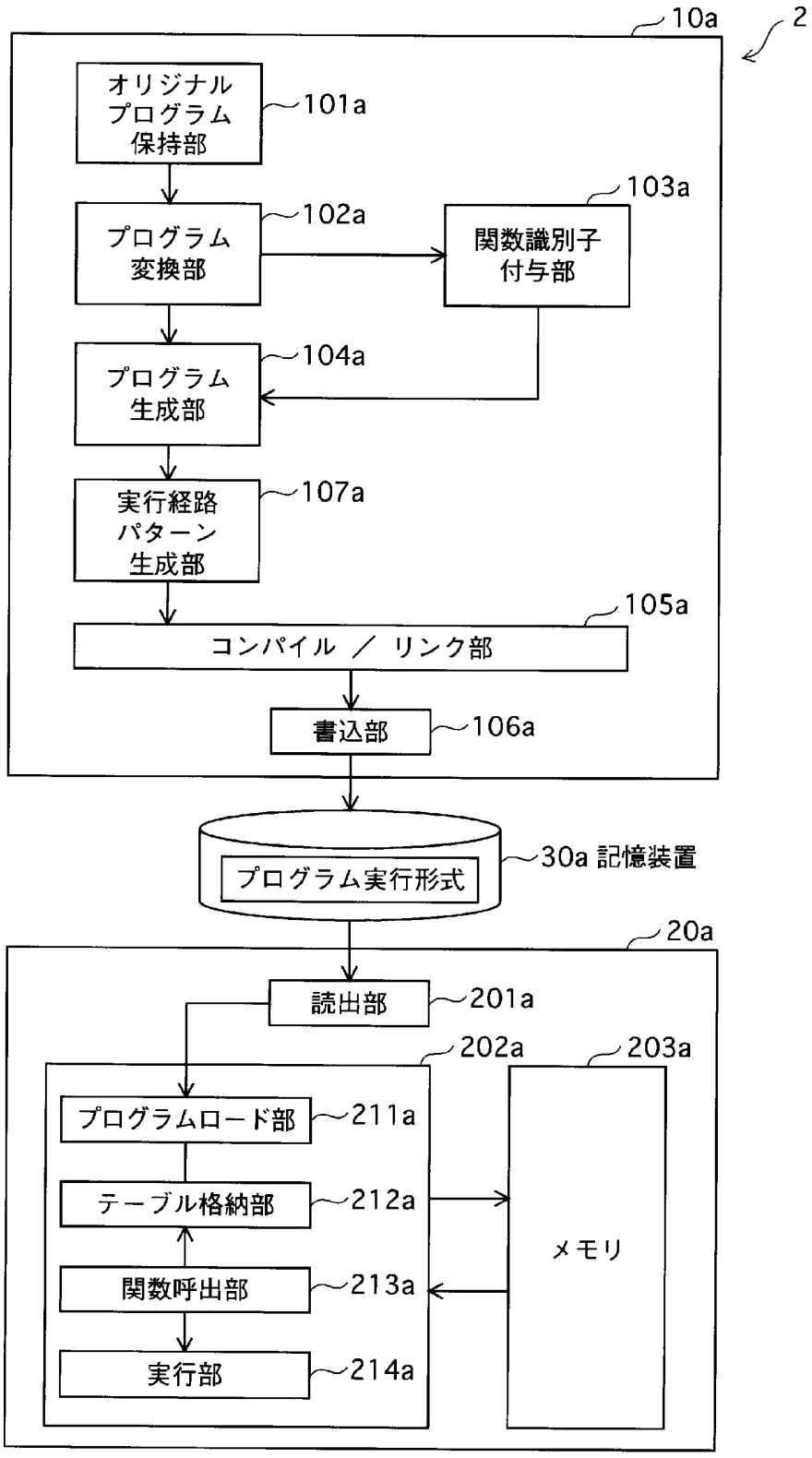
[図16]


240
241
242
243
244
b
X0
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
1
3
5
7
0 func A
1 func B
6 -
7 -
4 func D
5 func C
2 func C1
3 func Dummy

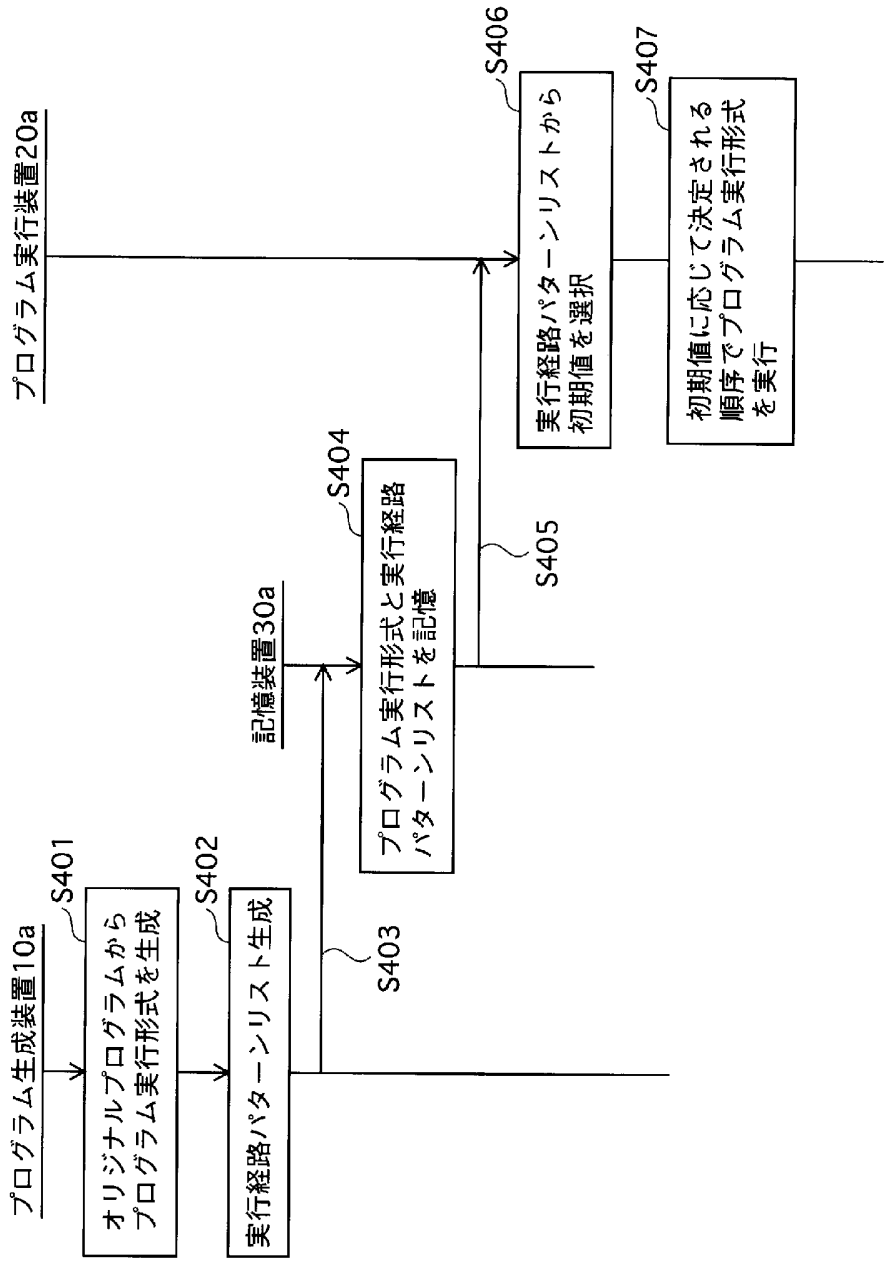
[図17]


250
251
シーケンシャル
グループ
252 並列グループ
253
254 恒等グループ
261
funcA
262
funcB
263
funcDummy
264
funcC
265
funcC1
266
funcD

[図18]


10a
2
オリジナル
プログラム
保持部
101a
プログラム
変換部
102a
103a
関数識別子
付与部
プログラム
生成部
104a
実行経路
パターン
生成部
107a
105a
コンパイル ／ リンク部
書込部
106a
プログラム実行形式
30a 記憶装置
20a
読出部
201a
202a
203a
プログラムロード部
211a
テーブル格納部
212a
メモリ
関数呼出部
213a
実行部
214a

[図19]

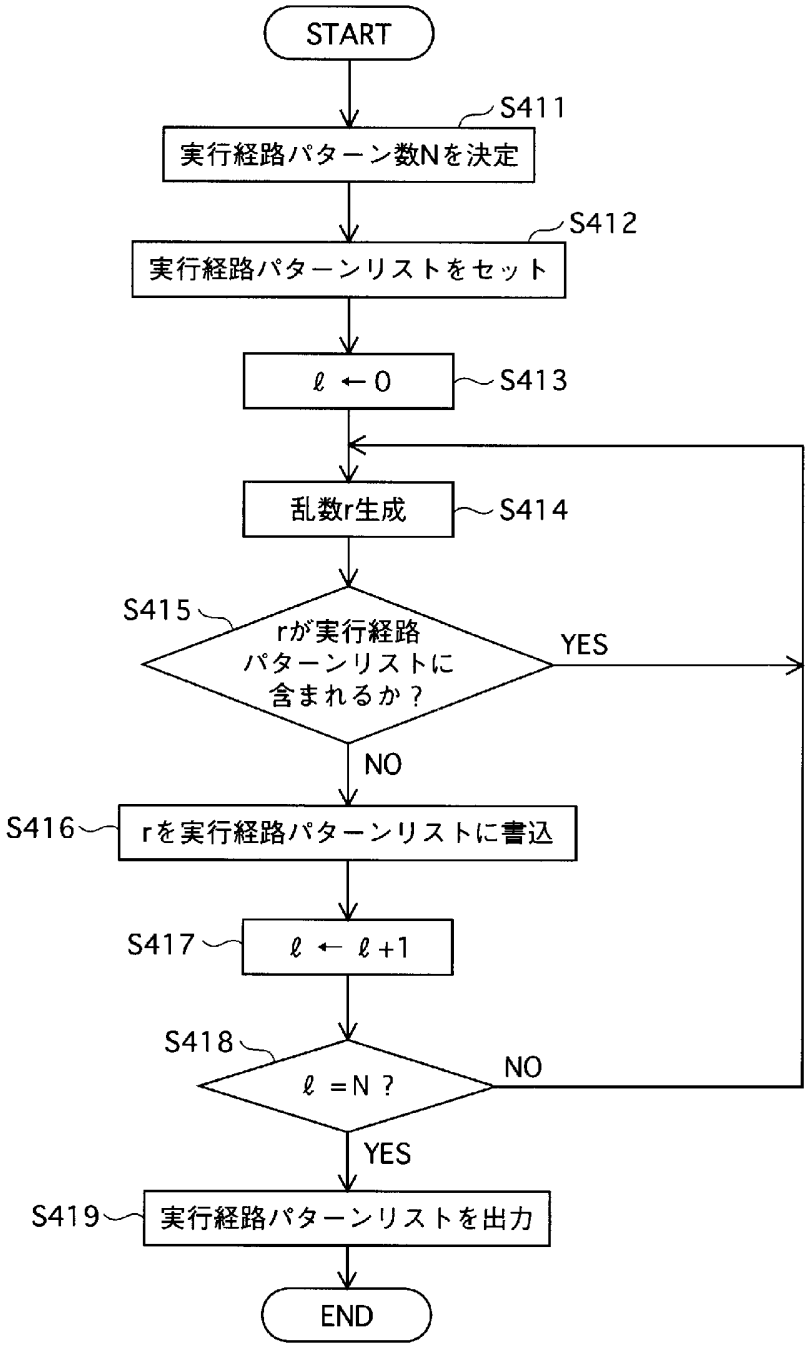

プログラム生成装置10a
S401
オリジナルプログラムから
プログラム実行形式を生成
S402
実行経路パターンリスト生成
S403
記憶装置30a
S404
プログラム実行形式と実行経路
パターンリストを記憶
S405
プログラム実行装置20a
S406
実行経路パターンリストから
初期値を選択
S407
初期値に応じて決定される
順序でプログラム実行形式
を実行

[図20]

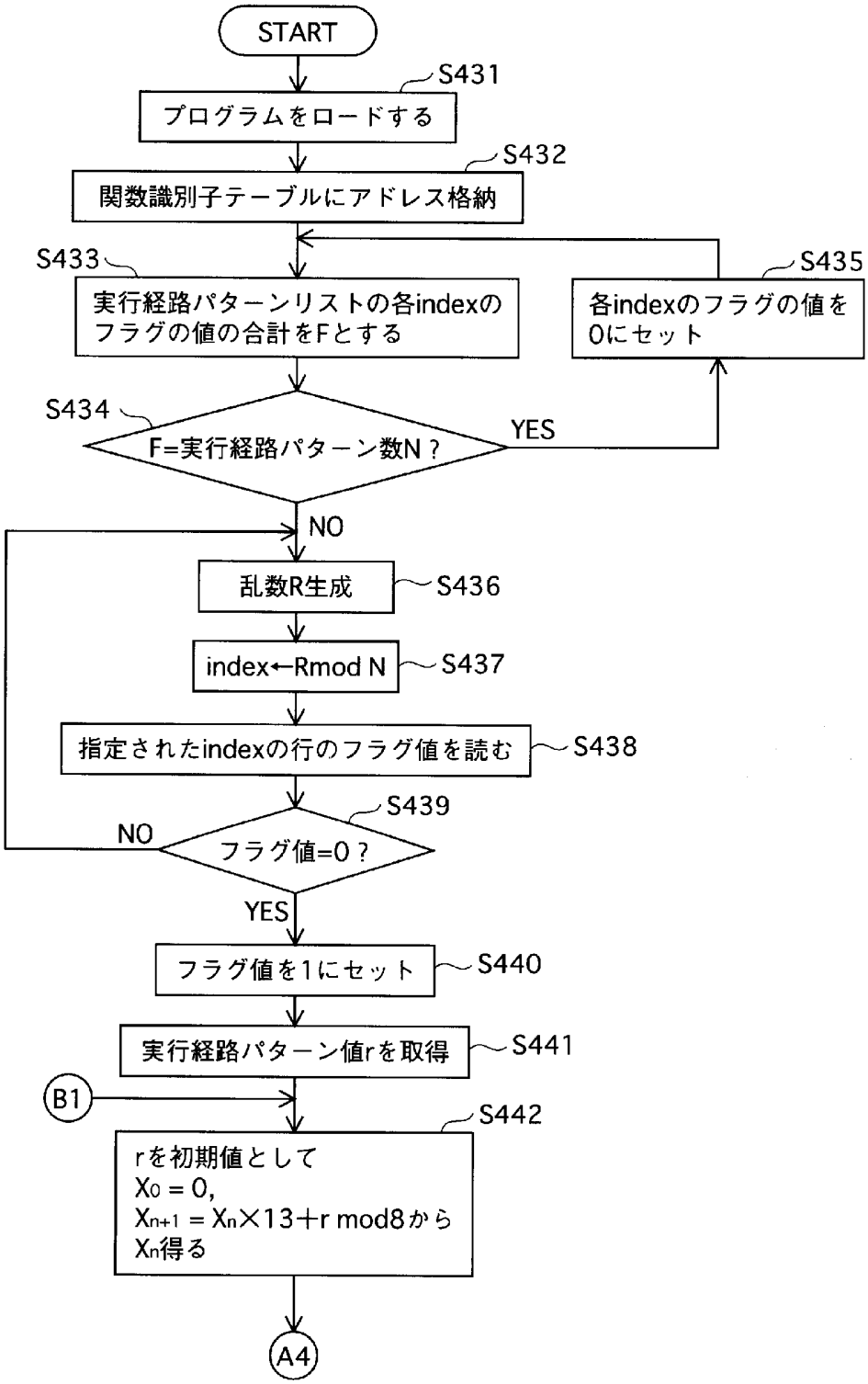
300 実行経路パターンリスト

index	実行経路パターン値	フラグ	
0	1	0	301
1	3	0	302
2	5	0	303
3	7	0	304

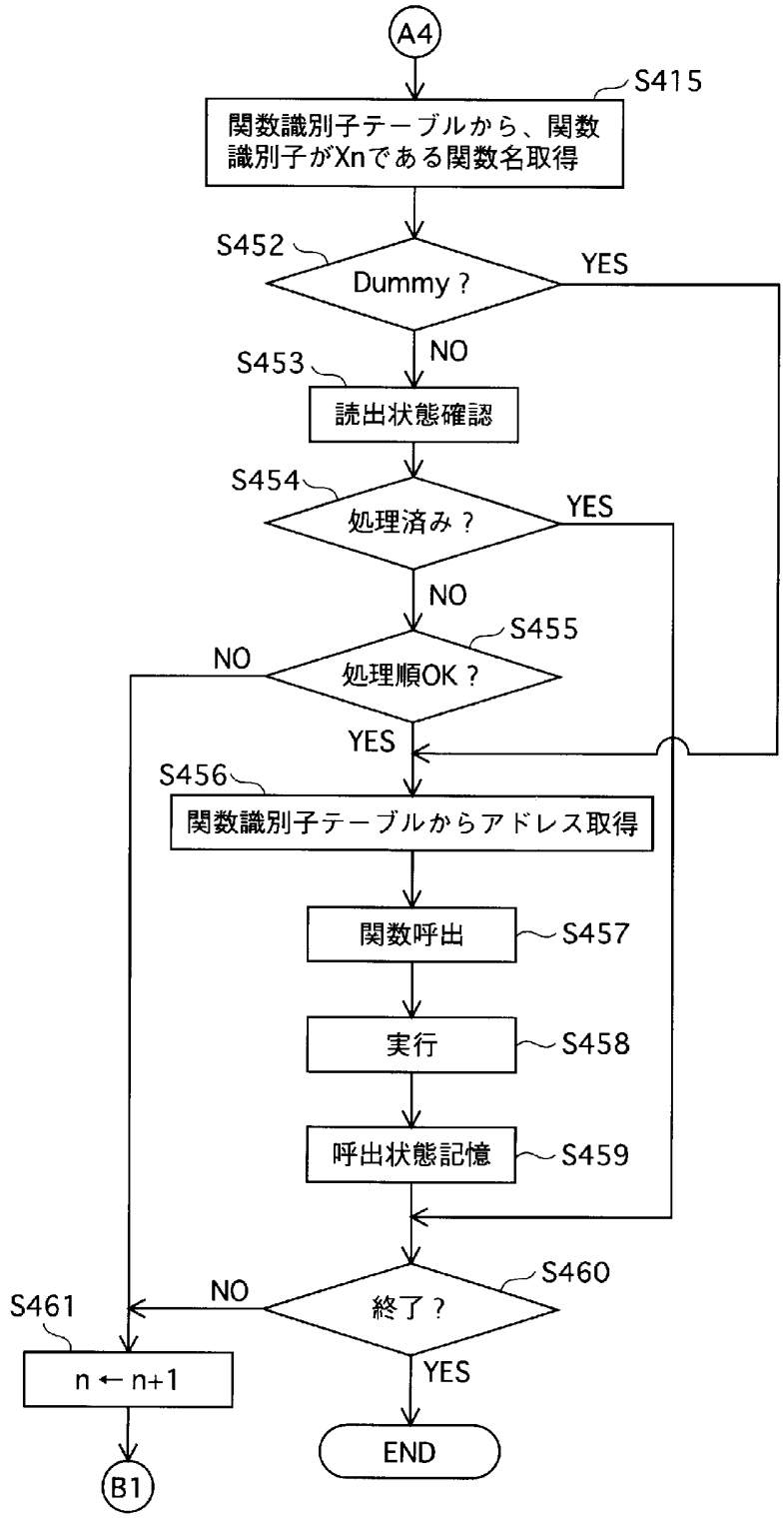
[図21]


START
S411
実行経路パターン数Nを決定
S412
実行経路パターンリストをセット
ℓ ← 0
S413
乱数r生成
S414
S415
rが実行経路
パターンリストに
含まれるか？
YES
NO
S416
rを実行経路パターンリストに書込
S417
ℓ ← ℓ +1
S418
ℓ =N ？
NO
YES
S419
実行経路パターンリストを出力
END

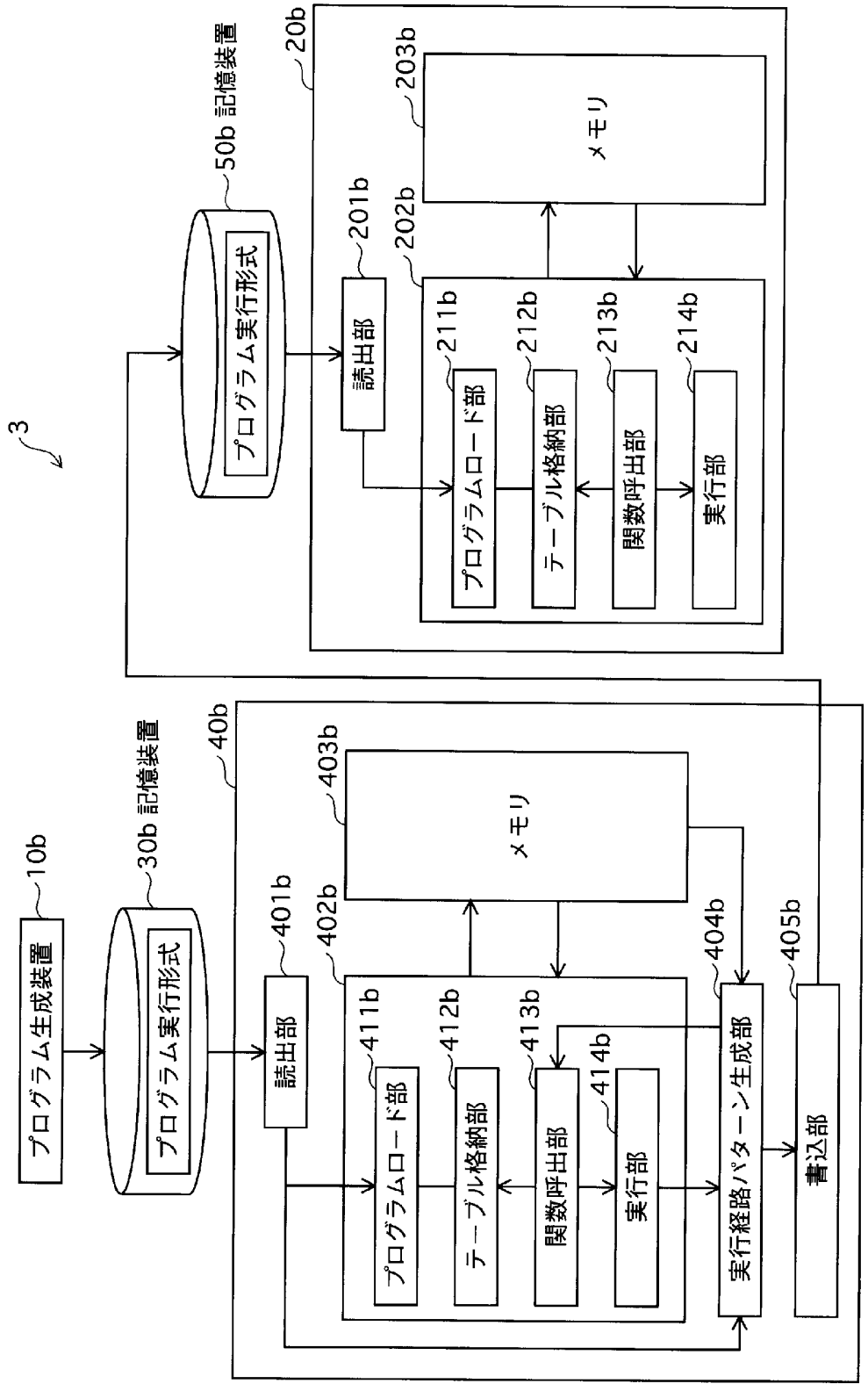
[図22]


START
S431
プログラムをロードする
S432
関数識別子テーブルにアドレス格納
S433
実行経路パターンリストの各indexのフラグの値の合計をFとする
S435
各indexのフラグの値を0にセット
S434
F=実行経路パターン数N？
YES
NO
乱数R生成
S436
index←Rmod N
S437
指定されたindexの行のフラグ値を読む
S438
S439
NO
フラグ値=0？
YES
フラグ値を1にセット
S440
実行経路パターン値rを取得
S441
B1
S442
rを初期値として
X0 = 0,
Xn+1 = Xn×13＋r mod8から
Xn得る
A4

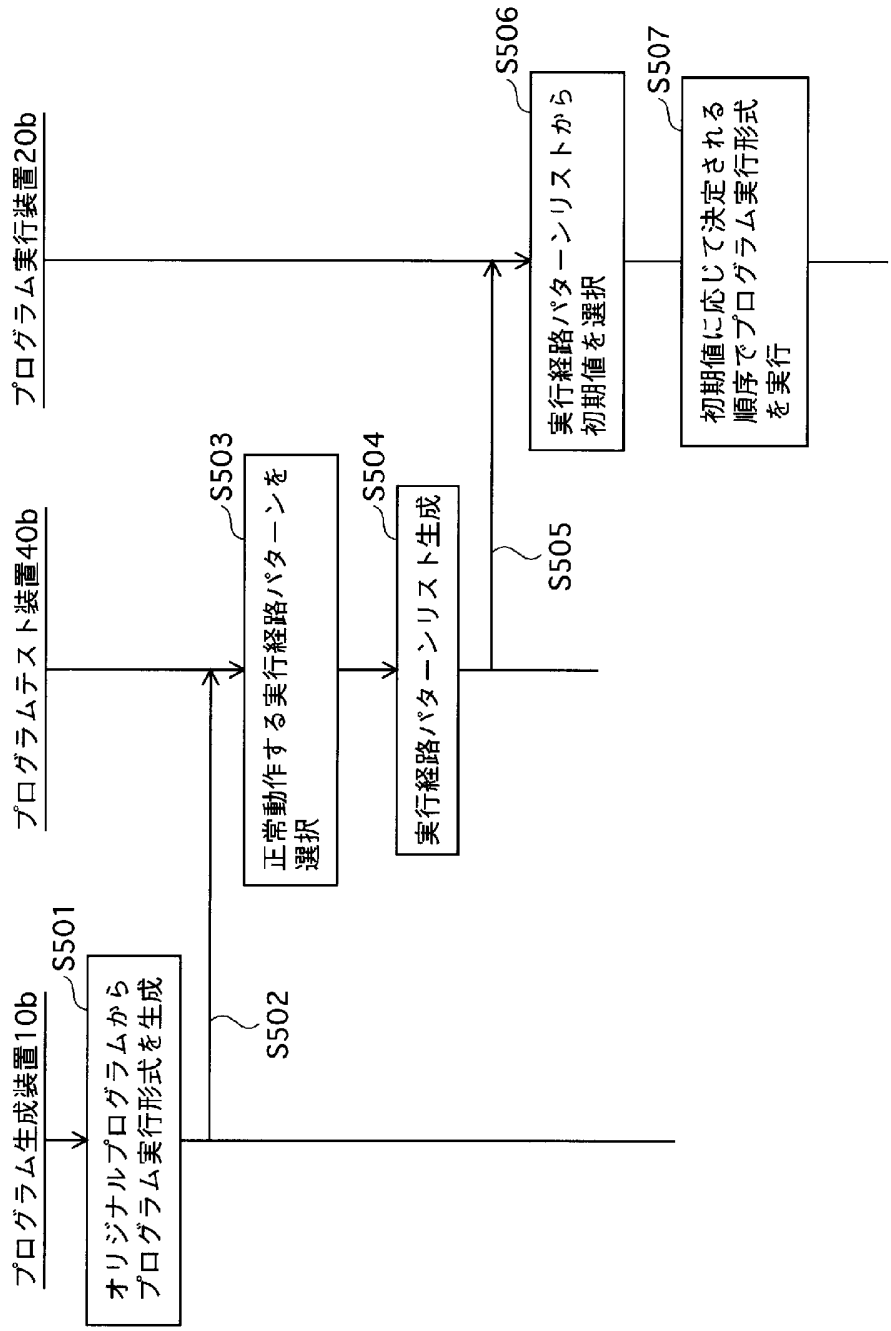
[図23]


A4
S415
関数識別子テーブルから、関数識別子がXnである関数名取得
S452
Dummy ?
YES
NO
S453
読出状態確認
S454
処理済み ?
YES
NO
S455
NO
処理順OK ?
YES
S456
関数識別子テーブルからアドレス取得
関数呼出
S457
実行
S458
呼出状態記憶
S459
S460
NO
終了 ?
S461
n ← n+1
YES
END
B1

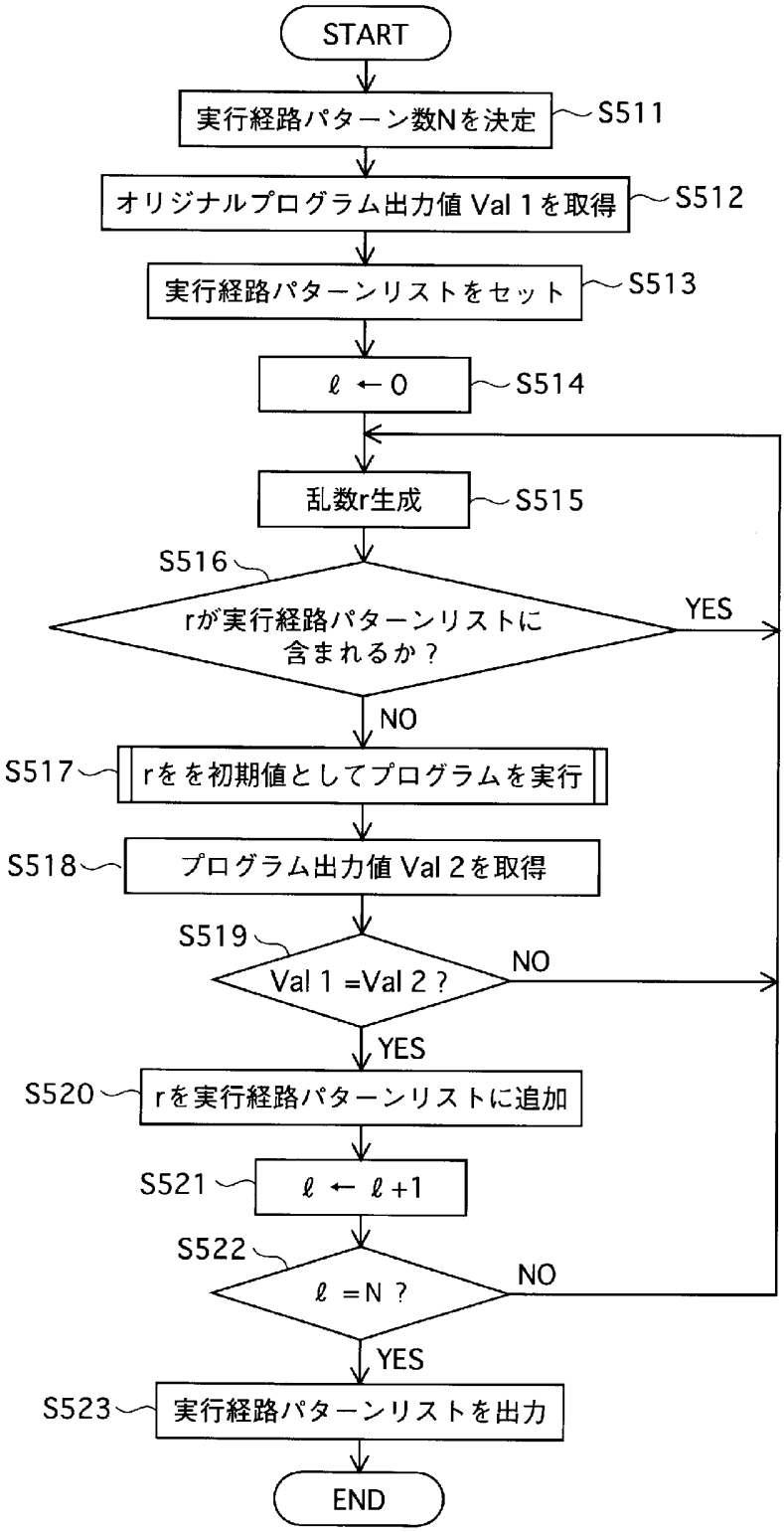
[図24]


3
10b
プログラム生成装置
30b 記憶装置
プログラム実行形式
40b
401b
読出部
402b
403b
メモリ
411b
プログラムロード部
412b
テーブル格納部
413b
関数呼出部
414b
実行部
404b
実行経路パターン生成部
405b
書込部
50b 記憶装置
プログラム実行形式
20b
201b
読出部
202b
203b
メモリ
211b
プログラムロード部
212b
テーブル格納部
213b
関数呼出部
214b
実行部

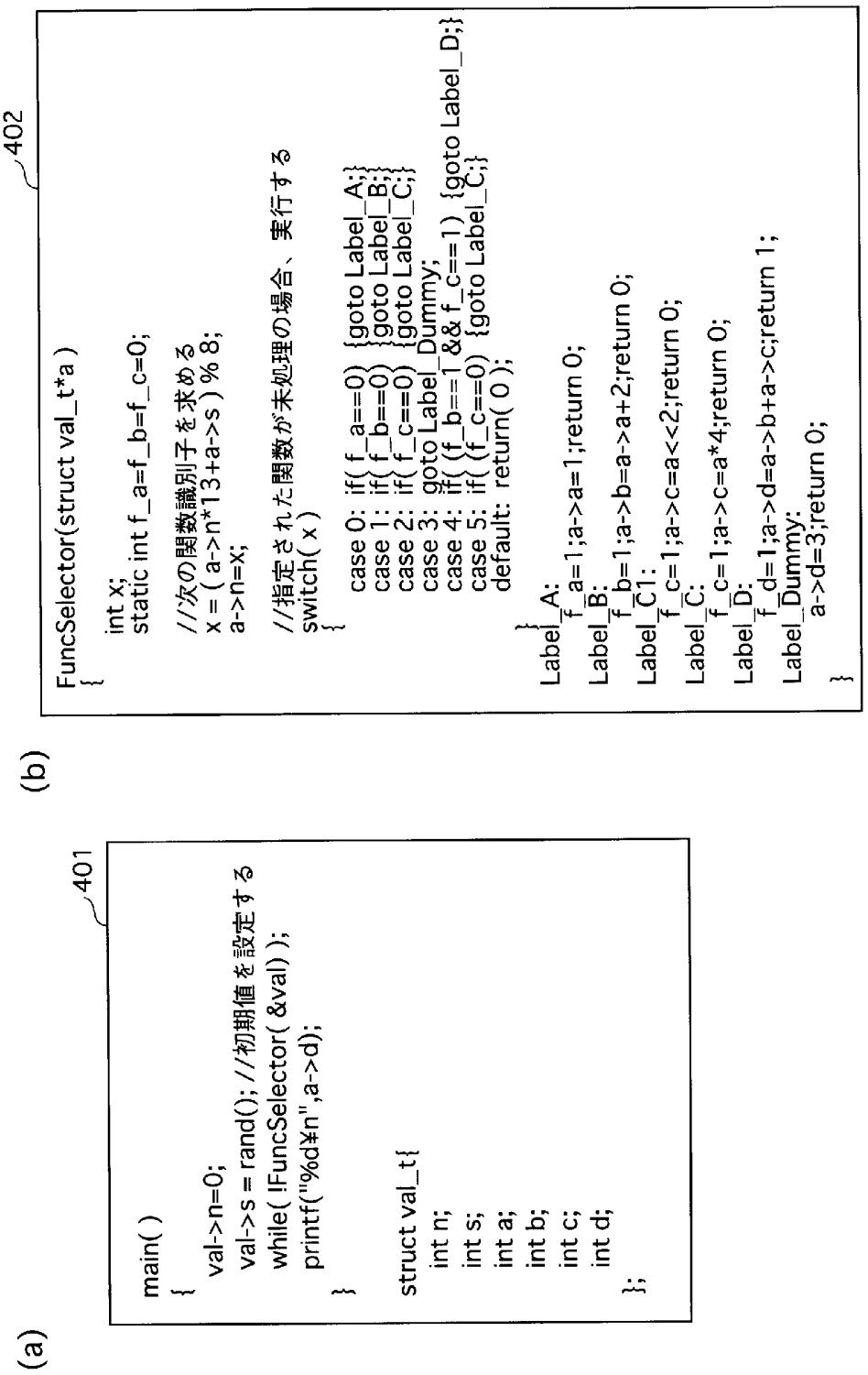
[図25]


プログラム生成装置10b
S501
オリジナルプログラムから
プログラム実行形式を生成
S502
プログラムテスト装置40b
S503
正常動作する実行経路パターンを
選択
S504
実行経路パターンリスト生成
S505
プログラム実行装置20b
S506
実行経路パターンリストから
初期値を選択
S507
初期値に応じて決定される
順序でプログラム実行形式
を実行

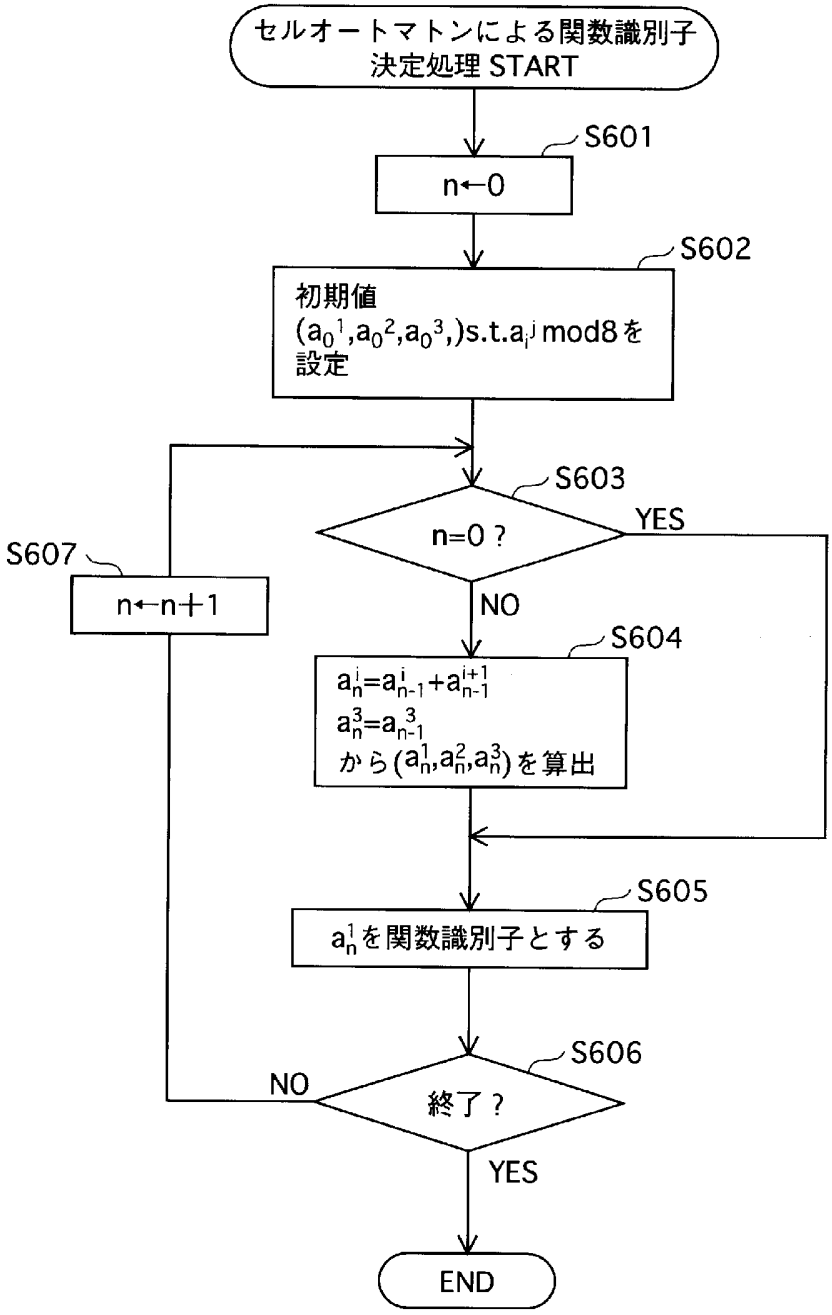
[図26]


START
実行経路パターン数Nを決定
S511
オリジナルプログラム出力値 Val 1を取得
S512
実行経路パターンリストをセット
S513
ℓ ← 0
S514
乱数r生成
S515
S516
rが実行経路パターンリストに含まれるか？
YES
NO
S517
rをを初期値としてプログラムを実行
S518
プログラム出力値 Val 2を取得
S519
Val 1 =Val 2 ?
NO
YES
S520
rを実行経路パターンリストに追加
S521
ℓ ← ℓ +1
S522
ℓ =N ?
NO
YES
S523
実行経路パターンリストを出力
END

[図27]


(a)
401
main( )
{
val->n=0;
val->s = rand(); //初期値を設定する
while(!FuncSelector(&val));
printf("%d¥n",a->d);
}
struct val_t{
int n;
int s;
int a;
int b;
int c;
int d;
};
(b)
402
FuncSelector(struct val_t*a)
{
int x;
static int f_a=f_b=f_c=0;
//次の関数識別子を求める
x = (a->n*13+a->s) % 8;
a->n=x;
//指定された関数が未処理の場合、実行する
switch(x)
{
case 0: if(f_a==0) {goto Label_A;}
case 1: if(f_b==0) {goto Label_B;}
case 2: if(f_c==0) {goto Label_C;}
case 3: goto Label_Dummy;
case 4: if((f_b==1 && f_c==1)) {goto Label_D;}
case 5: if((f_c==0)) {goto Label_C;}
default: return(0);
}
Label_A:
f_a=1;a->a=1;return 0;
Label_B:
f_b=1;a->b=a->a+2;return 0;
Label_C1:
f_c=1;a->c=a<<2;return 0;
Label_C:
f_c=1;a->c=a*4;return 0;
Label_D:
f_d=1;a->d=a->b+a->c;return 1;
Label_Dummy:
a->d=3;return 0;
}

[図28]

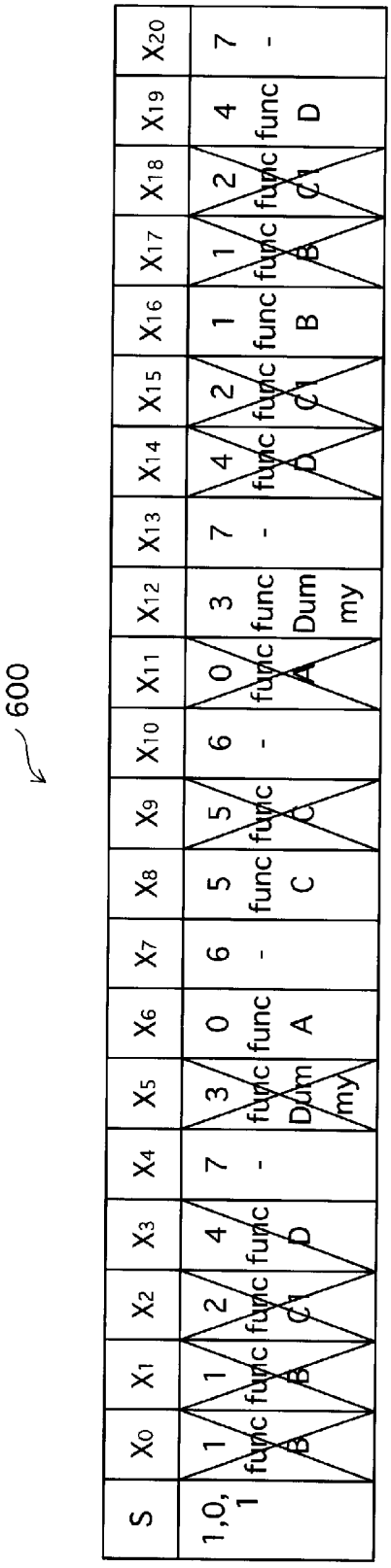

セルオートマトンによる関数識別子決定処理 START
S601
n←0
S602
初期値
$(a_0^1, a_0^2, a_0^3,)$s.t.a_i^j mod8を設定
S603
n=0？
YES
NO
S604
$a_n^i = a_{n-1}^i + a_{n-1}^{i+1}$
$a_n^3 = a_{n-1}^3$
から(a_n^1, a_n^2, a_n^3)を算出
S605
a_n^1を関数識別子とする
S606
終了？
NO
YES
END
S607
n←n+1

[図29]

500

n	a_n^1	a_n^2	a_n^3
0	1	0	1
1	1	1	1
2	2	2	1
3	4	3	1
4	7	4	1
5	3	5	1
6	0	6	1
7	6	7	1
8	5	0	1
9	5	1	1
10	6	2	1
11	0	3	1
12	3	4	1
13	7	5	1
14	4	6	1
15	2	7	1
16	1	0	1
17	1	1	1
18	2	2	1
19	4	3	1
20	7	4	1

[図30]


600
S
X0
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
X13
X14
X15
X16
X17
X18
X19
X20
1,0,1
1
func
B
1
func
B
2
func
C1
4
func
D
7
-
3
func
Dum
my
0
func
A
6
-
5
func
C
5
func
C
6
-
0
func
A
3
func
Dum
my
7
-
4
func
D
2
func
C1
1
func
B
1
func
B
2
func
C1
4
func
D
7
-

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2005/011602

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[7] G06F1/00

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)
Int.Cl[7] G06F1/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Jitsuyo Shinan Koho 1922-1996 Jitsuyo Shinan Toroku Koho 1996-2005
Kokai Jitsuyo Shinan Koho 1971-2005 Toroku Jitsuyo Shinan Koho 1994-2005

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.
X A	Akiteru KAMOSHIDA, Tsutomu MATSUMOTO, Shingo INOUE, "Tai-Tamper Software no Kosei Shuho ni Kansuru Kosatsu", The Institute of Electronics, Information and Communication Engineers Gijutsu Kenkyu Hokoku, Vol.97, No.461, 19 December, 1997 (19.12.97), pages 69 to 78	1-9,11, 14-19,22-31 10,12,13, 20,21

☐ 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
"T" 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	Date of mailing of the international search report
10 August, 2005 (10.08.05)	30 August, 2005 (30.08.05)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (January 2004)

国際調査報告

国際出願番号 PCT／JP2005／011602

A. 発明の属する分野の分類（国際特許分類（IPC））

Int.Cl.[7] G06F1/00

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））

Int.Cl.[7] G06F1/00

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	1971－2005年
日本国実用新案登録公報	1996－2005年
日本国登録実用新案公報	1994－2005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	関連する請求の範囲の番号
X	鴨志田昭輝，松本勉，井上信吾，耐タンパーソフトウェアの構成手法に関する考察，電子情報通信学会技術研究報告，vol.97，No.461，1997.12.19，p.69-p.78	1-9，11，14-19，22-31
A		10，12，13，20，21

☐ 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矛盾するものではなく、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 10．08．2005	国際調査報告の発送日 30.8.2005
国際調査機関の名称及びあて先 日本国特許庁（ISA／JP） 郵便番号100－8915 東京都千代田区霞が関三丁目4番3号	特許庁審査官（権限のある職員） 5S 3449 平井　誠 電話番号　03－3581－1101　内線　3546