(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7035751号
(P7035751)

(45)発行日　令和4年3月15日(2022.3.15)

(24)登録日　令和4年3月7日(2022.3.7)

(51)国際特許分類

		FI		
G06F	*8/41 (2018.01)*	G06F	8/41	130
G06F	*9/38 (2006.01)*	G06F	9/38	370A
G06F	*9/312(2006.01)*	G06F	9/312	L
G06F	*17/16 (2006.01)*	G06F	9/312	W
		G06F	17/16	G

請求項の数　10　（全31頁）

(21)出願番号　特願2018-77108(P2018-77108)
(22)出願日　平成30年4月12日(2018.4.12)
(65)公開番号　特開2019-185486(P2019-185486A)
(43)公開日　令和1年10月24日(2019.10.24)
審査請求日　令和3年1月13日(2021.1.13)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100074099
弁理士　大菅 義之
(74)代理人　100133570
弁理士　▲徳▼永 民雄
(72)発明者　木村 茂
神奈川県川崎市中原区上小田中４丁目１番１号 富士通株式会社内
審査官　松崎 孝大

最終頁に続く

(54)【発明の名称】　コード変換装置、コード変換方法、及びコード変換プログラム

(57)【特許請求の範囲】
【請求項１】
複数の配列のデータ定義と、前記複数の配列に対する所定の演算と、前記所定の演算の演算結果を表す配列のデータ定義とを含む、第１コードを記憶する記憶部と、
前記第１コードに含まれる前記複数の配列のデータ定義と前記演算結果を表す配列のデータ定義とを、構造体配列のデータ定義に変換し、前記第１コードに含まれる前記所定の演算を、前記構造体配列に対する演算に変換する変換部と、
前記複数の配列各々の異なるデータに対して、前記構造体配列に対する演算を並列に実行する所定の命令を含む、第２コードを生成する生成部と、
を備えることを特徴とするコード変換装置。
【請求項２】
前記所定の命令は、複数のレジスタそれぞれに格納された配列のデータに対して、前記構造体配列に対する演算を実行し、演算結果を表す配列のデータを所定のレジスタに書き込む命令であり、
前記第２コードは、
メモリに連続して格納された複数の構造体配列のデータを読み出して、各構造体配列のデータに含まれる前記複数の配列のデータを、前記複数のレジスタにそれぞれ書き込む命令と、
前記所定のレジスタから前記演算結果を表す配列のデータを読み出して、前記メモリに格納された各構造体配列のデータに含まれる、前記演算結果を表す配列のデータの位置に書

き込む命令と、
をさらに含むことを特徴とする請求項１記載のコード変換装置。
【請求項３】
前記所定の命令は、複数のレジスタそれぞれに格納された配列のデータに対して、前記構造体配列に対する演算を実行し、演算結果を表す配列のデータを所定のレジスタに書き込む命令であり、
前記第２コードは、
メモリに連続して格納された複数の構造体配列のデータを読み出して、第１レジスタに連続して書き込む命令と、
前記第１レジスタから、各構造体配列のデータに含まれる同じ配列のデータの位置を指定して、指定した位置のデータを読み出し、前記複数のレジスタのうち同じレジスタに連続して書き込む命令と、
前記所定のレジスタから前記演算結果を表す配列のデータを読み出し、前記メモリに格納された各構造体配列のデータに含まれる、前記演算結果を表す配列のデータの位置を指定して、前記所定のレジスタから読み出したデータを前記メモリの指定した位置に書き込む命令と、
をさらに含むことを特徴とする請求項１記載のコード変換装置。
【請求項４】
前記変換部は、データ定義の変換を示すコンパイラオプションに従って、前記第１コードに含まれる配列の中から、前記複数の配列と前記演算結果を表す配列とを選択することを特徴とする請求項１乃至３のいずれか１項に記載のコード変換装置。
【請求項５】
前記第１コードは、前記複数の配列と前記演算結果を表す配列とを指定する制御文を含み、
前記変換部は、前記制御文に従って、前記第１コードに含まれる配列の中から、前記複数の配列と前記演算結果を表す配列とを選択することを特徴とする請求項１乃至３のいずれか１項に記載のコード変換装置。
【請求項６】
前記変換部は、前記第１コードに含まれる配列のアクセス頻度を示すプロファイル情報を用いて、前記第１コードに含まれる配列の中から、前記複数の配列と前記演算結果を表す配列とを選択することを特徴とする請求項１乃至３のいずれか１項に記載のコード変換装置。
【請求項７】
前記変換部は、前記第１コードに含まれるループ内における各配列の出現回数、又は前記ループ内で同じ添え字を有する配列のグループにおける各配列の出現回数のうち、少なくとも一方に基づいて、前記ループに含まれる配列の中から前記複数の配列を選択することを特徴とする請求項１乃至６のいずれか１項に記載のコード変換装置。
【請求項８】
前記第１コードに含まれる複数の配列のデータ定義は、配列構造体のデータ定義であることを特徴とする請求項１乃至７のいずれか１項に記載のコード変換装置。
【請求項９】
複数の配列のデータ定義と、前記複数の配列に対する所定の演算と、前記所定の演算の演算結果を表す配列のデータ定義とを含む、第１コードを記憶する記憶部を有するコンピュータが、
前記第１コードに含まれる前記複数の配列のデータ定義と前記演算結果を表す配列のデータ定義とを、構造体配列のデータ定義に変換し、
前記第１コードに含まれる前記所定の演算を、前記構造体配列に対する演算に変換し、
前記複数の配列各々の異なるデータに対して、前記構造体配列に対する演算を並列に実行する所定の命令を含む、第２コードを生成することを特徴とするコード変換方法。
【請求項１０】
複数の配列のデータ定義と、前記複数の配列に対する所定の演算と、前記所定の演算の演

算結果を表す配列のデータ定義とを含む、第１コードを記憶する記憶部を有するコンピュータに、
前記第１コードに含まれる前記複数の配列のデータ定義と前記演算結果を表す配列のデータ定義とを、構造体配列のデータ定義に変換させ、
前記第１コードに含まれる前記所定の演算を、前記構造体配列に対する演算に変換させ、
前記複数の配列各々の異なるデータに対して、前記構造体配列に対する演算を並列に実行する所定の命令を含む、第２コードを生成させることを特徴とするコード変換プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、コード変換装置、コード変換方法、及びコード変換プログラムに関する。
【背景技術】
【０００２】
従来のコンピュータにおいて、演算性能を向上させるために、複数のデータに対する演算を同じ演算器で並列に実行する、ＳＩＭＤ（Single Instruction Multiple Data）命令が利用されることが多くなっている。特に、高い演算性能が期待されるスーパーコンピュータ又はサーバ内で動作する演算処理装置において、ＳＩＭＤ命令が用意されている。演算処理装置は、プロセッサと呼ばれることもある。
【０００３】
ＳＩＭＤ命令をサポートしているプロセッサは、ＳＩＭＤ命令の実行時に、メモリからデータをレジスタに読み出し、そのレジスタを使用して、所定のＳＩＭＤ幅に含まれる複数のＳＩＭＤ要素を単位として並列に演算を行う。そして、プロセッサは、それらのＳＩＭＤ要素を単位として、演算結果をメモリに格納する。例えば、４個の要素に対して同時に同じ演算を行う場合、ＳＩＭＤ要素の個数（要素数）は４個である。
【０００４】
また、ソフトウェアで記述されるプログラムロジック、特に、繰り返し処理（ループ処理）を高速化するために、コンパイラが最適な命令展開を行うことが望まれる。ループ処理を高速化する技術として、ループアンロール、ソフトウェアパイプライン、ループマージ等、様々な方法が考案されている。
【０００５】
ＳＩＭＤ命令に関連して、実行効率が向上するように、異なるデータに対して同じ種類の演算を並列実行するように指示する特定命令を含むコードを生成する技術が知られている（例えば、特許文献１を参照）。
【先行技術文献】
【特許文献】
【０００６】
【文献】特開２０１３－２０６２９１号公報
【発明の概要】
【発明が解決しようとする課題】
【０００７】
ソースコードに含まれるループ内のＳＩＭＤ命令のオブジェクト展開は、コンパイラによって、ベクトルレジスタの水平方向に行われることが多いため、ＳＩＭＤ命令は、主として水平方向に対して適用される。一方、ベクトルレジスタの垂直方向に対しては、ＳＩＭＤ命令を適用しないか、又は、データの配置を水平方向に変換してから、ＳＩＭＤ命令を水平方向に対して適用することが考えられる。
【０００８】
以下では、ＳＩＭＤ命令のオブジェクト展開を指して、ＳＩＭＤ展開と記載することがある。また、ＳＩＭＤ命令によるＳＩＭＤ演算に使用されるベクトルレジスタを指して、ＳＩＭＤレジスタと記載することがある。
【０００９】

ＳＩＭＤ展開には、配列構造体（Structure of Arrays，ＳＯＡ）形式のデータ定義が適している。ＳＯＡ形式は、複数の連続する要素からなるデータ定義であり、ＳＯＡ形式のデータに含まれる複数の要素は、連続的にアクセスすることが容易である。

【００１０】

しかしながら、ＳＯＡ形式のデータ定義では、各配列のすべての要素がキャッシュメモリ内に収まるとは限らないため、局所性が高いアクセスに対するキャッシュ効率が低下する。

【００１１】

なお、かかる問題は、ＳＯＡ形式のデータに対してＳＩＭＤ命令を適用する場合に限らず、他のデータ定義に基づく配列の異なる要素に対して演算を並列に実行する場合においても生ずるものである。

【００１２】

１つの側面において、本発明は、アクセスの局所性が高い配列の異なる要素に対して並列に実行される演算の性能を向上させることを目的とする。

【課題を解決するための手段】

【００１３】

１つの案では、コード変換装置は、記憶部、変換部、及び生成部を含む。記憶部は、複数の配列のデータ定義と、それらの配列に対する所定の演算と、所定の演算の演算結果を表す配列のデータ定義とを含む、第１コードを記憶する。

【００１４】

変換部は、第１コードに含まれる複数の配列のデータ定義と演算結果を表す配列のデータ定義とを、構造体配列（Array of Structures，ＡＯＳ）のデータ定義に変換し、第１コードに含まれる所定の演算を、構造体配列に対する演算に変換する。生成部は、複数の配列各々の異なるデータに対して、構造体配列に対する演算を並列に実行する所定の命令を含む、第２コードを生成する。

【発明の効果】

【００１５】

１つの実施形態によれば、アクセスの局所性が高い配列の異なる要素に対して並列に実行される演算の性能を向上させることができる。

【図面の簡単な説明】

【００１６】

【図１】ＳＯＡ形式のデータに対するＳＩＭＤ展開を示す図である。
【図２】ＳＯＡ形式のデータに対するＳＩＭＤ演算を示す図である。
【図３】配列の格納領域を示す図である。
【図４】コード変換装置の機能的構成図である。
【図５】コード変換処理のフローチャートである。
【図６】コード変換装置の具体例を示す機能的構成図である。
【図７】ＡＯＳ形式のデータ定義及びＳＩＭＤ演算を示す図である。
【図８】ハイブリッドＡＯＳ形式のデータ定義を示す図である。
【図９】処理性能を示す図である。
【図１０】抽出方法Ｍ１を示す図である。
【図１１】抽出方法Ｍ２を示す図である。
【図１２】抽出方法Ｍ３を示す図である。
【図１３】ループ管理テーブルを示す図である。
【図１４】評価値テーブルを示す図である。
【図１５】１次元の配列に対する変換処理を示す図である。
【図１６】２次元の配列に対する変換処理を示す図である。
【図１７】既存の命令を用いるＳＩＭＤ展開を示す図である。
【図１８】ＡＯＳ専用命令を用いるＳＩＭＤ展開を示す図である。
【図１９】コード変換処理の具体例を示すフローチャートである。
【図２０】グループ番号設定処理のフローチャートである。

【図２１】変換候補抽出処理のフローチャートである。
【図２２】変換対象選択処理のフローチャートである。
【図２３】中間コード生成処理のフローチャートである。
【図２４】機械語コード生成処理のフローチャートである。
【図２５】情報処理装置の構成図である。
【発明を実施するための形態】
【００１７】
以下、図面を参照しながら、実施形態を詳細に説明する。
図１は、ＦＯＲＴＲＡＮで記述されたＳＯＡ形式のデータに対する演算のＳＩＭＤ展開の例を示している。図１（ａ）は、１次元の配列Ａ、配列Ｂ、及び配列Ｃに対するＳＯＡ形式のデータ定義の例を示している。各配列の要素は、倍精度実数であり、各配列の要素数は、ｎ（ｎは２以上の整数）である。
【００１８】
図１（ｂ）は、配列Ａ、配列Ｂ、及び配列Ｃの格納領域の例を示している。Ａｉは、配列Ａのｉ番目（ｉ＝１～ｎ）の要素Ａ（ｉ）を表し、Ｂｉは、配列Ｂのｉ番目の要素Ｂ（ｉ）を表し、Ｃｉは、配列Ｃのｉ番目の要素Ｃ（ｉ）を表す。Ａ１～Ａｎからなるストリームは、領域１０１内に連続して格納され、Ｂ１～Ｂｎからなるストリームは、領域１０２内に連続して格納され、Ｃ１～Ｃｎからなるストリームは、領域１０３内に連続して格納される。この場合、３つのストリームを同時にアクセスすることが可能である。
【００１９】
図１（ｃ）は、配列Ａ及び配列Ｂに対する演算のループを含むソースコードの例を示している。この例では、ｎ＝１０２４であり、ｄｏループ内に、Ｃ（ｉ）＝Ａ（ｉ）＋Ｂ（ｉ）という演算が含まれている。
【００２０】
図１（ｄ）は、図１（ｃ）のループに対するＳＩＭＤ展開の例を示している。この例では、ＳＩＭＤ命令によって同時に処理できる要素数は４個であり、３つのストリームが連続域アクセスの対象となる。したがって、図１（ｂ）の領域１０１～領域１０３を用いて、各ストリームが４要素毎にアクセスされ、配列Ａの４個の要素と配列Ｂの４個の要素に対して並列に加算処理が行われる。そして、４個の加算結果が並列に領域１０３に格納される。
【００２１】
このように、ＳＯＡ形式のデータ定義によれば、各配列の複数の要素がメモリ内の連続する領域に格納される。一方、ＳＩＭＤ演算に使用するデータをメモリからＳＩＭＤレジスタに読み出す場合、連続して格納された複数の要素を、ＳＩＭＤレジスタに連続して読み出すのが一般的である。
【００２２】
図２は、ＳＩＭＤレジスタを利用して図１（ｃ）の処理を行うＳＩＭＤ演算の例を示している。メモリ１～メモリ３は、メモリ内の連続する領域を表す。メモリ１には、配列Ａの要素が格納されており、メモリ２には、配列Ｂの要素が格納されており、メモリ３には、配列Ｃの要素が格納されている。
【００２３】
各配列の要素のデータサイズは８バイトであり、ＳＩＭＤレジスタ１～ＳＩＭＤレジスタ３は、６４ビットのデータを８個格納することができる。したがって、ＳＩＭＤ命令によって同時に処理できる要素数は８個である。
【００２４】
まず、ＳＩＭＤロード命令により、メモリ１の先頭アドレスから順に、連続する８個の要素であるＡ１～Ａ８が読み出されて、ＳＩＭＤレジスタ１の水平方向に連続して書き込まれる。同時に、メモリ２の先頭アドレスから順に、連続する８個の要素であるＢ１～Ｂ８が読み出されて、ＳＩＭＤレジスタ２の水平方向に連続して書き込まれる。
【００２５】

次に、ＳＩＭＤレジスタ１及びＳＩＭＤレジスタ２の８個のデータに対して並列に加算処理が実行され、加算結果であるＣ１～Ｃ８がＳＩＭＤレジスタ３に書き込まれる。そして、ＳＩＭＤレジスタ３からＣ１～Ｃ８が連続して読み出され、メモリ３の先頭アドレスから順に書き込まれる。これにより、１回のループのＳＩＭＤ演算が完了し、次のループのＳＩＭＤ演算では、次の８個の要素について、同様の処理が繰り返される。

【００２６】

このように、ＳＩＭＤ演算の演算結果は、ＳＩＭＤレジスタ３内に連続して格納されているため、その演算結果をＳＯＡ形式のデータとしてメモリ３に格納すればよく、演算結果の並べ替えは不要である。

【００２７】

しかしながら、ＳＯＡ形式のデータ定義では、配列構造体に含まれる複数の配列の要素へのアクセスに際して、アクセスの局所性が低下するという問題がある。例えば、ビジネス系のアプリケーションプログラムでは、特定のデータの再利用率が高いことが多く、データに対するアクセスの局所性が高くなる。ＳＯＡ形式のデータの場合、各配列のすべての要素がキャッシュメモリ内に収まるとは限らないため、局所性が高いアクセスに対するキャッシュ効率が低下する。

【００２８】

この場合、ＳＯＡ形式のデータよりも、ＡＯＳ形式で定義された離散的なデータを扱う方が、キャッシュ効率が高くなり、処理性能が向上する。そこで、処理性能を向上させるために、ソースコードに含まれるＳＯＡ形式のデータ定義を、プログラマがＡＯＳ形式のデータ定義に変更する方法が考えられる。

【００２９】

図３は、ＡＯＳ形式で定義された配列Ａ、配列Ｂ、及び配列Ｃの格納領域の例を示している。Ａ１、Ｂ１、及びＣ１は、領域３０１内に連続して格納され、Ａ２、Ｂ２、及びＣ２は、領域３０２内に連続して格納され、Ａ３、Ｂ３、及びＣ３は、領域３０３内に連続して格納される。アプリケーションプログラムにおいて、特定のＡｉ、Ｂｉ、及びＣｉの組み合わせが高い頻度で再利用される場合、その組み合わせのデータがキャッシュメモリ内に留まることによって、キャッシュ効率が向上する。

【００３０】

ただし、従来のＳＩＭＤ展開では、連続する要素がＳＩＭＤレジスタの水平方向に格納されるため、ＡＯＳ形式のデータに対して、ＳＩＭＤレジスタを利用してＳＩＭＤ演算を行うことは困難である。

【００３１】

また、ＳＯＡ形式のデータ定義では、１回の処理で扱う領域が長いため、連続域アクセスによるページサイズオーバーに起因するＴＬＢ（Translation Lookaside Buffer）ミスが多発し、処理性能が劣化する可能性もある。さらに、連続するストリームに対するハードウェアプリフェッチ又はソフトウェアプリフェッチが冗長に発行された場合、処理性能がさらに劣化する。

【００３２】

図１及び図２の例では、配列Ａ、配列Ｂ、及び配列Ｃのストリーム毎に発行されるハードウェアプリフェッチ又はソフトウェアプリフェッチによって、ハードウェア資源が消費される。このため、キャッシュメモリへの冗長なデータ書き込みによるパイプラインに投入される命令数の増加、スケジューリングの阻害、バス幅の消費による転送速度の低下等、様々な性能劣化が発生する。

【００３３】

図４は、実施形態のコード変換装置の機能的構成例を示している。図４のコード変換装置４０１は、記憶部４１１、変換部４１２、及び生成部４１３を含む。記憶部４１１は、複数の配列のデータ定義と、それらの配列に対する所定の演算と、所定の演算の演算結果を表す配列のデータ定義とを含む、第１コード４２１を記憶する。

【００３４】

図５は、図４のコード変換装置４０１が行うコード変換処理の例を示すフローチャートである。まず、変換部４１２は、第１コード４２１に含まれる複数の配列のデータ定義と演算結果を表す配列のデータ定義とを、構造体配列のデータ定義に変換する（ステップ５０１）。次に、変換部４１２は、第１コード４２１に含まれる所定の演算を、構造体配列に対する演算に変換する（ステップ５０２）。そして、生成部４１３は、複数の配列各々の異なるデータに対して、構造体配列に対する演算を並列に実行する所定の命令を含む、第２コードを生成する（ステップ５０３）。

【００３５】

図４のコード変換装置４０１によれば、アクセスの局所性が高い配列の異なる要素に対して並列に実行される演算の性能を向上させることができる。

【００３６】

図６は、図４のコード変換装置４０１の具体例を示している。図６のコード変換装置４０１は、記憶部４１１、変換部４１２、生成部４１３、及び解析部６１１を含み、高級言語で記述されたソースコード６２１を機械語コード６２６に変換する。例えば、ソースコード６２１は、ＦＯＲＴＲＡＮ、Ｃ言語、Ｃ＋＋、ＬＩＳＰ等で記述されたコードであってもよい。コード変換装置４０１は、コンパイラ装置と呼ばれることもある。

【００３７】

記憶部４１１は、ソースコード６２１及び評価関数６２２を記憶する。ソースコード６２１は、図４の第１コード４２１に対応し、ソースプログラムと呼ばれることもある。評価関数６２２は、ソースコード６２１に含まれる配列の評価値を計算するために用いられ、計算された評価値は、配列がＡＯＳ形式のデータ定義に適している度合いを示す。

【００３８】

解析部６１１は、ユーザから入力される最適化指示に従って、ソースコード６２１を解析し、解析結果に基づいてループ管理テーブル６２３を生成して、記憶部４１１に格納する。ループ管理テーブル６２３は、ソースコード６２１に含まれるループ毎に、ループに含まれる各配列の次元数、出現回数、各次元の添え字等の属性を含むテーブルである。

【００３９】

変換部４１２は、ループ管理テーブル６２３を参照して、データ定義を変換する変換候補の配列を抽出する。そして、変換部４１２は、評価関数６２２を用いて、変換候補の各配列に対する評価値を計算し、計算した評価値を含む評価値テーブル６２４を生成して、記憶部４１１に格納する。

【００４０】

次に、変換部４１２は、評価値テーブル６２４を参照して、変換候補の配列の中から変換対象の配列を選択し、選択した配列のデータ定義を、ＡＯＳ形式のデータ定義に変換する。変換対象の配列には、所定の演算が適用される複数の配列と、それらの配列に対する所定の演算の演算結果を表す配列とが含まれる。変換対象の配列のデータ定義は、ＳＯＡ形式であってもよく、ＳＯＡ形式以外のデータ定義であってもよい。

【００４１】

次に、変換部４１２は、ソースコード６２１に含まれる所定の演算を、ＡＯＳ形式のデータに対する演算に変換し、変換後のデータ定義及び演算を含む中間コード６２５を生成して、記憶部４１１に格納する。

【００４２】

生成部４１３は、中間コード６２５に含まれる所定の演算に対するＳＩＭＤ展開を行うことで、中間コード６２５を最適化し、ＡＯＳ形式のデータの読み出し及び書き込みを行うＳＩＭＤ命令を含む、機械語コード６２６を生成して、記憶部４１１に格納する。機械語コード６２６は、第２コードに対応し、機械語プログラムと呼ばれることもある。

【００４３】

図６のコード変換装置４０１によれば、アクセスの局所性が高いデータに着目して、配列のデータ定義をＡＯＳ形式に変換することで、機械語コード６２６を実行するプロセッサのキャッシュ効率が高くなる。変換後のＡＯＳ形式のデータ定義では、複数の異なる配列

の要素が局所的に配置されるため、それらの要素のデータがまとまってキャッシュメモリ内に留まることが多い。したがって、配列に対するアクセスの局所性が高い場合、キャッシュミス率が低く抑えられ、キャッシュ効率が向上する。これにより、そのような配列に対するＳＩＭＤ演算の性能を向上させることができる。

【００４４】

また、ＳＯＡ形式のデータ定義をＡＯＳ形式に変換することで、１回の処理で扱う領域が短くなり、連続域アクセスによるページサイズオーバーに起因するＴＬＢミスが減少する。さらに、複数のストリームが１つのストリームにまとめられ、ストリーム数が削減されるため、ハードウェアプリフェッチ等によるハードウェア資源の消費を少なくすることができる。したがって、限られたハードウェア資源を効率的に活用することができ、プロセッサの処理性能が向上するとともに、消費電力も削減される。

【００４５】

生成部４１３は、ＡＯＳ形式のデータの読み出し及び書き込みを行うＳＩＭＤ命令として、ＳＩＭＤレジスタにアクセスする既存の命令を用いてもよく、新たに定義されるＡＯＳ専用命令を用いてもよい。ＡＯＳ専用命令としては、メモリからＡＯＳ形式のデータを読み出してＳＩＭＤレジスタに書き込む専用ロード命令と、ＳＩＭＤレジスタからデータを読み出してＡＯＳ形式でメモリに書き込む専用ストア命令とが定義される。

【００４６】

専用ロード命令は、メモリに連続して格納された複数の構造体配列のデータを読み出して、各構造体配列に含まれる複数の配列のデータを、複数のＳＩＭＤレジスタにそれぞれ書き込む命令である。

【００４７】

一方、専用ストア命令は、複数のＳＩＭＤレジスタから、各構造体配列に含まれる複数の配列のデータを読み出して、メモリに格納された各構造体配列の位置に書き込む命令である。したがって、専用ストア命令を実行することで、所定のＳＩＭＤレジスタから演算結果を表す配列のデータが読み出されて、メモリに格納された各構造体配列に含まれる、演算結果を表す配列の位置に書き込まれる。

【００４８】

専用ロード命令は、ＳＩＭＤレジスタの垂直方向に対して、メモリ内の要素を直接展開するために用いられ、専用ストア命令は、ＳＩＭＤレジスタの垂直方向に展開された要素を抽出して、メモリ内に格納するために用いられる。これらのＡＯＳ専用命令を用いることで、従来の水平方向のＳＩＭＤ展開に代えて、垂直方向のＳＩＭＤ展開を効率良く行うことができる。プロセッサは、専用ロード命令及び専用ストア命令をサポートすることで、ＡＯＳ形式のデータに対する処理性能がさらに向上する。

【００４９】

図７は、ＡＯＳ形式のデータ定義及びＳＩＭＤ演算の例を示している。図７（ａ）は、１次元の配列Ａ、配列Ｂ、及び配列Ｃに対するＡＯＳ形式のデータ定義の例を示している。構造体ｓｔｒｕｃｔは、配列Ａ、配列Ｂ、及び配列Ｃの要素からなり、各配列の要素は、倍精度実数であり、各配列の要素数は１０２４個である。構造体配列Ｓｔは、構造体ｓｔｒｕｃｔの配列であり、構造体配列Ｓｔの要素数も１０２４個である。この場合、構造体配列Ｓｔのｉ番目（ｉ＝１～１０２４）の要素に含まれるＡｉ、Ｂｉ、及びＣｉのデータが、連続してメモリ７０１に格納され、連続域アクセスの対象となる。

【００５０】

図７（ｂ）は、配列Ａ及び配列Ｂに対する演算のループを構造体配列Ｓｔを用いて記述したソースコードの例を示している。この例では、ｄｏループ内に、Ｓｔ（ｉ）％Ｃ＝Ｓｔ（ｉ）％Ａ＋Ｓｔ（ｉ）％Ｂという演算が含まれている。Ｓｔ（ｉ）％Ａは、構造体配列Ｓｔのｉ番目の要素に含まれる配列Ａの要素を表し、Ｓｔ（ｉ）％Ｂは、構造体配列Ｓｔのｉ番目の要素に含まれる配列Ｂの要素を表し、Ｓｔ（ｉ）％Ｃは、構造体配列Ｓｔのｉ番目の要素に含まれる配列Ｃの要素を表す。

【００５１】

図７（ｃ）は、図７（ｂ）のループに対するＳＩＭＤ展開の例を示している。この例では、ＳＩＭＤ命令によって同時に処理できる要素数は４個であり、メモリ７０１に格納されている１つのストリームのみが、連続域アクセスの対象となる。

【００５２】

したがって、図７（ａ）のメモリ７０１の先頭アドレスから順に、構造体配列Ｓｔの４個の要素に対応するＡ１～Ｃ４の１２個のデータが読み出され、ＳＩＭＤレジスタ１～ＳＩＭＤレジスタ３の垂直方向に順に書き込まれる。そして、ＳＩＭＤレジスタ１に格納された配列Ａの４個の要素と、ＳＩＭＤレジスタ２に格納された配列Ｂの４個の要素に対して、並列に加算処理が行われ、４個の加算結果が並列にＳＩＭＤレジスタ３に書き込まれる。

【００５３】

同時にアクセスされるＡｉ、Ｂｉ、及びＣｉのデータは、メモリ７０１内で互いに近接して格納されているため、キャッシュメモリ内においても互いに近接して配置される。したがって、ＳＯＡ形式の場合とは異なり、Ａｉ、Ｂｉ、及びＣｉのうち一部のデータがキャッシュメモリから欠落する可能性は低くなる。

【００５４】

データのキャッシングを行うことで、次の処理ステップで使用されるまで、データをキャッシュメモリ内に留めておくことができる。これにより、次のアクセス時におけるアクセスコストが削減される。しかし、より大きなサイズのデータセットの場合、すべてのデータがキャッシュメモリに収まりきらず、次に使用される前にデータが書き換えられてしまうこともある。したがって、同時に使用される頻度の高い配列だけを、ＡＯＳ形式の構造体のメンバに加えることが効果的である。

【００５５】

また、ＡＯＳ形式のデータの特性であるデータの隣接性を確保しつつ、ＳＯＡ形式のデータのロード順序をサポートする、ハイブリッドＡＯＳ形式をデータ定義として用いることもできる。

【００５６】

図８は、ハイブリッドＡＯＳ形式のデータ定義の例を示している。構造体Ｈｙｂｒｉｄ＿ｓｔｒｕｃｔは、配列Ａの８個の要素、配列Ｂの８個の要素、及び配列Ｃの８個の要素からなり、各配列の要素は、倍精度実数であり、各配列の要素数は１０２４個である。構造体配列Ｓｔは、構造体Ｈｙｂｒｉｄ＿ｓｔｒｕｃｔの配列であり、構造体配列Ｓｔの要素数は１２８個である。

【００５７】

この場合、構造体配列Ｓｔのｉ番目（ｉ＝１～１２８）の要素は、Ａ（８＊（ｉ－１）＋１）～Ａ（８＊ｉ）、Ｂ（８＊（ｉ－１）＋１）～Ｂ（８＊ｉ）、及びＣ（８＊（ｉ－１）＋１）～Ｃ（８＊ｉ）からなる。これらの２４個のデータが連続してメモリ８０１に格納され、連続域アクセスの対象となる。したがって、構造体配列Ｓｔの１つの添え字によって、２４個のデータをアクセス対象として指定することができる。

【００５８】

図８のハイブリッドＡＯＳ形式のデータ定義によれば、図２のＳＩＭＤ演算の場合と同様に、プロセッサは、配列Ａ、配列Ｂ、及び配列Ｃのいずれからも８個のデータを同時にロードすることができる。この場合、配列Ａ、配列Ｂ、及び配列Ｃの同じ添え字を有する要素同士が隣接していなくても、それらの要素は十分に近い位置に存在するため、通常は同じメモリページに格納されると考えられる。

【００５９】

したがって、複数の異なる配列の要素が局所的に配置されるため、キャッシュ効率が向上する可能性が高くなる。また、１回の処理で扱う領域が短くなるため、連続域アクセスによるページサイズオーバーに起因するＴＬＢミスが減少する可能性も高くなる。

【００６０】

このように、コード変換装置４０１は、プログラムが扱うデータの特性に応じて、配列のデータ定義をＡＯＳ形式又はハイブリッドＡＯＳ形式に変換することができる。データ特

性とデータに対する処理に応じてデータ定義を切り換えることで、プログラム全体を最適化することが可能になる。

【００６１】

図９は、プログラム特性とデータ定義の組み合わせに応じた処理性能の例を示している。○印は、処理の高速化が可能であることを示し、△印は、処理の高速化が部分的に可能であることを示し、×印は、処理の高速化が不可能であることを示す。

【００６２】

データに対するアクセスの局所性が存在しない場合、ＳＯＡ形式のデータ定義の方が、ＡＯＳ形式のデータ定義よりも処理性能が高くなる。一方、データに対するアクセスの局所性が存在する場合、ＡＯＳ形式のデータ定義の方が、ＳＯＡ形式のデータ定義よりも処理性能が高くなる。

【００６３】

したがって、データに対するアクセスの局所性が存在する場合、ソースコード６２１に含まれるＳＯＡ形式のデータ定義をＡＯＳ形式のデータ定義に変換し、所定の演算をＡＯＳ形式のデータに対する演算に変換することで、処理性能の大幅な向上が期待できる。

【００６４】

ハードウェア資源の負荷としては、ＨＰＦ（Hardware Prefetch）による負荷と、ＴＬＢミスによる負荷とが考慮される。ＳＯＡ形式のデータ定義では、ＨＰＦによる負荷及びＴＬＢミスによる負荷がともに大きいため、処理性能が低くなる。一方、ＡＯＳ形式のデータ定義では、ＨＰＦによる負荷は存在せず、ＴＬＢミスによる負荷も小さいため、処理性能は高くなる。

【００６５】

したがって、ソースコード６２１に含まれるＳＯＡ形式のデータ定義をＡＯＳ形式のデータ定義に変換し、所定の演算をＡＯＳ形式のデータに対する演算に変換することで、ハードウェア資源の負荷を削減して、処理性能をさらに向上させることができる。

【００６６】

プロセッサのＳＩＭＤサポート状況において、水平方向のＳＩＭＤ命令がサポートされている場合、ＳＯＡ形式のデータ定義の方が、ＡＯＳ形式のデータ定義よりも処理性能が高くなる。ＡＯＳ形式のデータ定義では、データをＳＩＭＤレジスタに格納した後、水平方向に並べ替える処理が発生するため、ＳＩＭＤ演算による効果が相殺され、逆に性能が劣化する場合もある。

【００６７】

一方、垂直方向のＳＩＭＤ命令がサポートされている場合、ＡＯＳ形式のデータ定義の方が、ＳＯＡ形式のデータ定義よりも処理性能が高くなる。特に、ＡＯＳ専用命令がサポートされている場合、ＡＯＳ形式のデータに対する処理性能がさらに向上する。

【００６８】

変換部４１２は、以下のいずれかの抽出方法を用いて、ソースコード６２１に含まれる配列の中から、変換候補の配列を抽出することができる。

【００６９】

Ｍ１：ソースコード６２１の静的解析に基づく変換候補の抽出
Ｍ２：ソースコード６２１に記述された制御文に基づく変換候補の抽出
Ｍ３：プロファイル情報に基づく変換候補の抽出

【００７０】

抽出方法Ｍ１を採用した場合、ユーザは、データ定義の変換を示すコンパイラオプションを指定し、変換部４１２は、指定されたコンパイラオプションに従って、変換候補の配列を抽出する。例えば、データ定義の変換を示すコンパイラオプションとしては、以下のようなものが用いられる。

【００７１】

－ＫＡｏｓ：すべての配列の中から変換候補を自動的に抽出するコンパイラオプション
－ＫＡｏｓ（Ａ，Ｂ）：ユーザが明示的に指定した配列名を有する配列を、変換候補とし

て抽出するコンパイラオプション
【００７２】
－ＫＡｏｓ（Ａ，Ｂ）のＡ及びＢは、ユーザが指定した配列名を表す。この場合、配列名Ａ及び配列名Ｂを有する配列のうち、次元数と要素数が共通する配列が変換候補として抽出される。コンパイラオプションの名称として、－ＫＡｏｓ以外の名称を用いても構わない。
【００７３】
－ＫＡｏｓを指定した場合、ユーザが配列を明示的に指定しなくても、変換候補を自動的に抽出することができる。一方、－ＫＡｏｓ（Ａ，Ｂ）を指定した場合、ユーザが明示的に指定した配列を、変換候補として抽出することができる。
【００７４】
図１０は、抽出方法Ｍ１の例を示している。－ＫＡｏｓ（Ａ，Ｂ）が指定された場合、ＤＯループから１次元の配列Ａ及び配列Ｂが変換候補として抽出される。そして、配列Ａ及び配列Ｂが変換対象として選択された場合、それらの配列のデータ定義が構造体配列Ｓｔのデータ定義に変換され、ＤＯループ内のＡ（ｉ）及びＢ（ｉ）が、Ｓｔ（ｉ）％Ａ及びＳｔ（ｉ）％Ｂにそれぞれ変換される。抽出される配列の次元数は、２次元以上であってもよい。
【００７５】
抽出方法Ｍ２を採用した場合、ユーザは、変換候補の配列を明示的に指定する制御文をソースコード６２１に記述し、変換部４１２は、その制御文に従って変換候補の配列を抽出する。例えば、制御文としては、ＦＯＲＴＲＡＮにおけるＯＣＬ（Object Constraint Language）文、Ｃ言語における＃ｐｒａｇｍａ等を用いることができ、制御文には、配列名、次元数、及び要素数を記述することができる。制御文を用いることで、ユーザが明示的に指定した配列を、変換候補として抽出することができる。
【００７６】
図１１は、抽出方法Ｍ２の例を示している。ＯＣＬ文“！ｏｃｌ　ＡＯＳ（Ａ，Ｂ，Ｃ）”によって、配列Ａ、配列Ｂ、及び配列Ｃが変換候補として指定された場合、ＤＯループから２次元の配列Ａ、配列Ｂ、及び配列Ｃが抽出される。そして、配列Ａ、配列Ｂ、及び配列Ｃが変換対象として選択された場合、それらの配列のデータ定義が構造体配列Ｓｔのデータ定義に変換される。さらに、ＤＯループ内のＡ（ｉ，ｊ）、Ｂ（ｉ，ｊ）、及びＣ（ｉ，ｊ）が、Ｓｔ（ｉ，ｊ）％Ａ、Ｓｔ（ｉ，ｊ）％Ｂ、及びＳｔ（ｉ，ｊ）％Ｃにそれぞれ変換される。抽出される配列の次元数は、１次元であってもよく、３次元以上であってもよい。
【００７７】
抽出方法Ｍ３を採用した場合、変換部４１２は、ソースコード６２１に含まれるループ処理における各配列のアクセス頻度を示すプロファイル情報を取得する。そして、変換部４１２は、取得したプロファイル情報を用いて、同時にアクセスされる頻度が高い複数の配列を、変換候補として抽出する。同じループ内の複数の配列に対するアクセスがともに高頻度で行われている場合、これらの配列のデータ定義をＡＯＳ形式に変換することで、それらの配列のデータが同じ期間にキャッシュメモリに留まる可能性が高くなる。
【００７８】
図１２は、抽出方法Ｍ３の例を示している。プロファイル情報１２０１は、高コストのループ処理において、配列Ａ及び配列Ｃが同時にアクセスされる頻度が高く、配列Ｂは他の配列と同時にアクセスされないことを示している。この場合、ＤＯループから１次元の配列Ａ及び配列Ｃが変換候補として抽出される。そして、配列Ａ及び配列Ｃが変換対象として選択された場合、それらの配列のデータ定義が構造体配列Ｓｔのデータ定義に変換され、ＤＯループ内のＡ（ｉ）及びＣ（ｉ）が、Ｓｔ（ｉ）％Ａ及びＳｔ（ｉ）％Ｃにそれぞれ変換される。抽出される配列の次元数は、２次元以上であってもよい。
【００７９】
図１３は、ソースコード６２１に含まれるループのループ管理テーブル６２３の例を示し

ている。図１３（ａ）は、ＤＯループの例を示しており、図１３（ｂ）は、解析部６１１が図１３（ａ）のＤＯループを解析して生成したループ管理テーブル６２３の例を示している。図１３（ｂ）のループ管理テーブル６２３は、以下の項目を含む。

【００８０】

変数：ループ内における配列の記述（添え字を含む）
配列名：配列の名称（添え字を含まない）
次元数：配列の次元数
出現回数：ループ内における配列の記述回数
ｐ次元（ｐ＝１～Ｐ）：配列のｐ番目の添え字（定数を含む）
グループ番号：同じ添え字を有する配列のグループを示す識別情報

【００８１】

変換部４１２は、ループ管理テーブル６２３を参照して、同じ添え字を有する複数の配列を検索し、それらの配列に同じグループ番号を付与する。図１３（ｂ）のループ管理テーブル６２３の場合、同じ添え字を有する配列のグループとして、以下のグループが抽出される。

【００８２】

｛Ａ（ｉ），Ｂ（ｉ）｝
｛ＡＡ（ｉ，ｊ），ＢＢ（ｉ，ｊ）｝
｛ＣＣ（ｘ，ｙ），ＤＤ（ｘ，ｙ）｝
｛α（Ｗ（ｉ））），β（Ｗ（ｉ））｝

【００８３】

配列α及び配列βの添え字は、間接参照を示すＷ（ｉ）であり、同じ添え字とみなすことができるため、これらの配列は同じグループに分類される。複数の配列の間で、間接参照を示す添え字が異なっている場合であっても、プロファイル情報等から、実行時にそれらの添え字が等しいと判断できる場合は、それらの配列が同じグループに分類される。

【００８４】

図１３（ａ）のＤＯループでは、配列のデータ領域として静的領域が用いられているが、データ領域が動的に獲得される配列についても、ループ管理テーブル６２３に登録することが可能である。

【００８５】

変換部４１２は、ループ管理テーブル６２３に含まれる配列の中から、抽出方法Ｍ１～抽出方法Ｍ３のいずれかを用いて、変換候補の配列を抽出する。抽出方法Ｍ１において、コンパイラオプション－ＫＡｏｓが指定された場合、変換部４１２は、いずれかのグループに分類された配列を、変換候補として抽出する。

【００８６】

抽出方法Ｍ１又は抽出方法Ｍ２において、コンパイラオプション又は制御文により複数の配列が明示的に指定された場合、変換部４１２は、指定された複数の配列が、ループ管理テーブル６２３内で同じグループ番号を有するか否かをチェックする。そして、指定された複数の配列が同じグループ番号を有する場合、変換部４１２は、それらの配列を変換候補に決定する。指定された複数の配列が同じグループ番号を有さない場合、変換部４１２は、それらの配列を変換候補から除外する。

【００８７】

抽出方法Ｍ３において、プロファイル情報が指定された場合、変換部４１２は、指定されたプロファイル情報を用いて、同時にアクセスされる頻度が高い複数の配列を、変換候補として抽出する。

【００８８】

配列名Ｑを有する配列の評価関数６２２としては、例えば、次式の評価関数Ｅ（Ｑ）を用いることができる。

Ｅ（Ｑ）＝（Ｃ（Ｑ）／Ｓ）＊ｗ１＋（Ｍ（Ｑ）／Ｇ（Ｑ））＊ｗ２　　（１）

【００８９】

式（１）のＳは、ループに含まれる配列の総数を表し、Ｃ（Ｑ）は、ループ内における配列Ｑの出現回数を表す。したがって、Ｃ（Ｑ）／Ｓは、ループ内における配列Ｑの割合（出現率）を表す。

【００９０】

Ｇ（Ｑ）は、配列Ｑと同じグループ番号を有する配列の総数を表し、Ｍ（Ｑ）は、そのグループ番号が示すグループ内における配列Ｑの出現回数を表す。したがって、Ｍ（Ｑ）／Ｇ（Ｑ）は、グループ内における配列Ｑの割合（一致率）を表す。

【００９１】

出現率が高い配列ほど、アクセスされる頻度が高いため、データ定義をＡＯＳ形式に変更することが効果的である。同様に、一致率が高い配列ほど、アクセスされる頻度が高いため、データ定義をＡＯＳ形式に変更することが効果的である。ｗ１は、出現率に対する重み係数を表し、ｗ２は、一致率に対する重み係数を表す。

【００９２】

変換部４１２は、ループ管理テーブル６２３に登録された各配列の出現回数を用いて、評価関数Ｅ（Ｑ）の値（評価値）を計算する。例えば、ｗ１＝１、ｗ２＝２とすると、図１３（ｂ）のループ管理テーブル６２３から、配列名Ａを有する配列の評価値が、次のようにして計算される。

【００９３】

Ｓ＝１４：ループに含まれる配列の総数は、下記の１４個である。
Ａ（ｉ），Ｂ（ｉ），ＡＡ（ｉ，ｊ），ＢＢ（ｉ，ｊ），ＡＡ（ｉ＋１，ｊ），
ＡＡ（ｘ，ｚ），ＢＢ（ｉ＋２，ｊ），ＢＢ（ｚ，ｃ），ＣＣ（ｘ，ｙ），
Ａ（ｉ＋２），ＡＡ（ｉ，ｊ），ＤＤ（ｘ，ｙ），α（Ｗ（ｉ）），β（Ｗ（ｉ））

【００９４】

Ｃ（Ａ）＝２：配列Ａの出現回数は、Ａ（ｉ）及びＡ（ｉ＋２）の２回である。
Ｇ（Ａ）＝２：配列Ａと同じグループ番号“１”を有する配列の総数は、Ａ（ｉ）及びＢ（ｉ）の２個である。
Ｍ（Ａ）＝１：グループ番号“１”が示すグループ内における配列Ａの出現回数は、Ａ（ｉ）の１回である。
Ｅ（Ａ）＝（２／１４）＊１＋（１／２）＊２＝１．１４　　（２）

【００９５】

図１４は、変換候補の配列に対する評価値テーブル６２４の例を示している。この例では、グループ番号“１”～グループ番号“４”の４個のグループに属するすべての配列が、変換候補として抽出されている。図１４の評価値テーブル６２４は、グループ番号、配列名Ｑ、Ｃ（Ｑ）、ｗ１、Ｍ（Ｑ）、Ｇ（Ｑ）、ｗ２、評価値、及び評価値合計を含む。評価値は、式（１）の評価関数Ｅ（Ｑ）を用いて計算された各配列の評価値を表し、評価値合計は、同じグループに属する配列の評価値の総和を表す。

【００９６】

変換部４１２は、評価値テーブル６２４を参照して、評価値合計が閾値よりも大きなグループに属する配列を、変換対象として選択する。さらに、変換部４１２は、各グループに属する配列の中から、評価値が閾値よりも大きな配列を変換対象として選択してもよい。これらの閾値は、事前に設定された所定値であってもよく、ユーザにより指定された値であってもよい。

【００９７】

例えば、評価値合計の閾値が２．２である場合、図１４のグループ番号“１”のグループに属するＡ（ｉ）及びＢ（ｉ）と、グループ番号“２”のグループに属するＡＡ（ｉ，ｊ）及びＢＢ（ｉ，ｊ）が、変換対象として選択される。

【００９８】

変換部４１２は、ループ管理テーブル６２３とは別に評価値テーブル６２４を生成する代わりに、評価値テーブル６２４の項目をループ管理テーブル６２３に追加して、２つのテーブルを統合してもよい。

【００９９】
評価関数Ｅ（Ｑ）は、配列Ｑの出現率又は配列Ｑの一致率のうち、いずれか一方のみを含む関数であっても構わない。さらに、評価関数Ｅ（Ｑ）は、出現率及び一致率以外の属性を含んでいても構わない。例えば、プロファイル情報から得られるプログラムの実行時の情報を、評価関数Ｅ（Ｑ）の属性として用いることができる。このような情報としては、各配列のアクセス回数、キャッシュミス等のプロセッサイベント情報、プロセッサが取得した実測値（経験値）又は論理値が挙げられる。
【０１００】
コード変換装置４０１は、事前に記憶している評価関数Ｅ（Ｑ）を用いる代わりに、コンパイラのパラメータとして外部から与えられた評価関数Ｅ（Ｑ）を用いて、評価値を計算することもできる。さらに、コード変換装置４０１は、人工知能の機械学習によって、プログラムの実行時に取得した情報を学習データとしてフィードバックし、自動的に評価関数Ｅ（Ｑ）を生成することも可能である。
【０１０１】
変換部４１２は、変換対象として選択した配列の構造体配列を、ソースコード６２１に追加する。例えば、配列Ａ及び配列Ｂが変換対象として選択された場合、変換部４１２は、配列Ａ及び配列Ｂの構造体ｓｔｒｕｃｔを定義し、その構造体の配列として、任意の配列名の構造体配列を定義する。そして、変換部４１２は、それらのデータ定義を、ソースコード６２１中のデータ記述部に追加する。
【０１０２】
ｔｙｐｅ　ｓｔｒｕｃｔ
属性１：：Ａ
属性２：：Ｂ
ｅｎｄ　ｔｙｐｅ　ｓｔｒｕｃｔ
ｔｙｐｅ（ｓｔｒｕｃｔ）：：Ｓｔ（ｎ）
この例では、配列名Ｓｔを有する構造体配列が定義されている。構造体配列Ｓｔの要素数ｎは、配列Ａ及び配列Ｂの要素数と同じである。
【０１０３】
次に、変換部４１２は、ソースコード６２１に含まれる配列Ａ及び配列Ｂの記述を、構造体配列Ｓｔの配列名を用いた記述に置き換えて、中間コード６２５を生成する。
Ａ（）→Ｓｔ（）％Ａ
Ｂ（）→Ｓｔ（）％Ｂ
【０１０４】
図１５は、１次元の配列Ａ、配列Ｂ、及び配列Ｃに対するデータ定義及び演算の変換処理の例を示している。ソースコード６２１に１次元の配列Ａ、配列Ｂ、及び配列Ｃが含まれている場合、それらの配列の要素からなる構造体ｓｔｒｕｃｔが定義され、その構造体の配列として、構造体配列Ｓｔが定義される。構造体配列Ｓｔの要素数は１０２４個である。そして、ＤＯループ内のＣ（ｉ）＝．．．Ａ（ｉ）＋Ｂ（ｉ）．．．という演算が、Ｓｔ（ｉ）％Ｃ＝．．．Ｓｔ（ｉ）％Ａ＋Ｓｔ（ｉ）％Ｂ．．．という演算に置き換えられる。
【０１０５】
図１６は、２次元の配列Ａ、配列Ｂ、及び配列Ｃに対するデータ定義及び演算の変換処理の例を示している。ソースコード６２１に２次元の配列Ａ、配列Ｂ、及び配列Ｃが含まれている場合、それらの配列の要素からなる構造体ｓｔｒｕｃｔが定義され、その構造体の配列として、構造体配列Ｓｔが定義される。構造体配列Ｓｔの添え字ｉ及びｊは、１～１０２４の範囲の整数である。そして、ＤＯループ内のＣ（ｉ，ｊ）＝Ａ（ｉ，ｊ）＋Ｂ（ｉ，ｊ）という演算が、Ｓｔ（ｉ，ｊ）％Ｃ＝Ｓｔ（ｉ，ｊ）％Ａ＋Ｓｔ（ｉ，ｊ）％Ｂという演算に置き換えられる。
【０１０６】
変換部４１２は、配列Ａ及び配列Ｂの配列名の変更を、ソースコード６２１に対して行う

代わりに、中間コード６２５に対して行ってもよい。
【０１０７】
次に、生成部４１３は、既存の命令又はＡＯＳ専用命令を用いて、中間コード６２５に対するＳＩＭＤ展開を行うことで、機械語コード６２６を生成する。
【０１０８】
図１７は、既存の命令を用いるＳＩＭＤ展開の例を示している。メモリ１７０１には、配列Ａ、配列Ｂ、及び配列Ｃに対するＡＯＳ形式のデータが連続して格納されており、各配列の要素のデータサイズは８バイトである。ｒｅｇ１、ｒｅｇ４、ｒｅｇ５、及びｒｅｇ６は、ＳＩＭＤレジスタであり、６４ビットのデータを複数個格納することができる。
【０１０９】
図１７（ａ）は、ロード命令の例を示している。プロセッサは、ロード命令ｌｄ３　ｒｅｇ１，ａｄｄｒを実行することで、メモリ１７０１からＡＯＳ形式のデータＡ１～Ｃ３を読み出して、ｒｅｇ１の水平方向に連続して書き込む。
【０１１０】
図１７（ｂ）は、配列Ａに対するｓｅｌｅｃｔ命令の例を示している。まず、プロセッサは、命令ｍｏｖ　ｒｅｇｘ，（０ｘ６＆０ｘ３＆０ｘ０）を実行することで、ｒｅｇ１内における配列Ａの要素Ａ１～Ａ３の位置を示す要素番号０、３、及び６を、レジスタｒｅｇｘに書き込む。
【０１１１】
次に、プロセッサは、ｓｅｌｅｃｔ命令ｓｅｌｅｃｔ　ｒｅｇ４，ｒｅｇ１，ｒｅｇｘ，３を実行することで、ｒｅｇｘ内の３個の要素番号が示す３個の要素を、ｒｅｇ１から読み出して、ｒｅｇ４の水平方向に連続して書き込む。このｓｅｌｅｃｔ命令は、ｒｅｇ１から、各構造体配列に含まれる配列Ａのデータの位置を指定して、指定した位置のデータを読み出し、ｒｅｇ４に連続して書き込む命令である。
【０１１２】
図１７（ｃ）は、配列Ｂに対するｓｅｌｅｃｔ命令の例を示している。まず、プロセッサは、命令ｍｏｖ　ｒｅｇｘ，（０ｘ７＆０ｘ４＆０ｘ１）を実行することで、ｒｅｇ１内における配列Ｂの要素Ｂ１～Ｂ３の位置を示す要素番号１、４、及び７を、レジスタｒｅｇｘに書き込む。
【０１１３】
次に、プロセッサは、ｓｅｌｅｃｔ命令ｓｅｌｅｃｔ　ｒｅｇ５，ｒｅｇ１，ｒｅｇｘ，３を実行することで、ｒｅｇｘ内の３個の要素番号が示す３個の要素を、ｒｅｇ１から読み出して、ｒｅｇ５の水平方向に連続して書き込む。このｓｅｌｅｃｔ命令は、ｒｅｇ１から、各構造体配列に含まれる配列Ｂのデータの位置を指定して、指定した位置のデータを読み出し、ｒｅｇ５に連続して書き込む命令である。
【０１１４】
プロセッサは、配列Ａの他の要素に対しても、図１７（ａ）と同様のｓｅｌｅｃｔ命令を実行することで、Ａ１～Ａ８をレジスタ４に書き込むことができる。また、プロセッサは、配列Ｂの他の要素に対しても、図１７（ｂ）と同様のｓｅｌｅｃｔ命令を実行することで、Ｂ１～Ｂ８をレジスタ５に書き込むことができる。
【０１１５】
図１７（ｄ）は、ｒｅｇ４～ｒｅｇ６を用いたＳＩＭＤ命令の例を示している。プロセッサは、ＳＩＭＤ命令ＡＤＤ　ｒｅｇ６，ｒｅｇ４，ｒｅｇ５を実行することで、ｒｅｇ４に格納されたＡｉ（ｉ＝１～８）と、ｒｅｇ５に格納されたＢｉとを並列に加算して、加算結果Ｃｉをｒｅｇ６に書き込む。
【０１１６】
図１７（ｅ）は、ｓｃａｔｔｅｒ命令の例を示している。まず、プロセッサは、ロード命令ｌｄｒ　ｘ１，＆Ａ１を実行することで、メモリ１７０１内における加算結果Ｃｉ（ｉ＝１～８）の格納先先頭アドレスを計算して、不図示のレジスタｘ１に書き込む。例えば、Ｃ１、Ｃ２、及びＣ３の格納先先頭アドレスは、それぞれ、１６、４０、及び６４であ

る。
【０１１７】
次に、プロセッサは、命令ｍｏｖ　ｒｅｇｘ，（６４，４０，１６）を実行することで、レジスタｘ１に格納された、Ｃ１～Ｃ３の格納先先頭アドレスを、ｒｅｇｘに書き込む。プロセッサは、配列Ｃの他の要素に対しても同様の命令を実行することで、Ｃ１～Ｃ８の格納先先頭アドレスをｒｅｇｘに書き込むことができる。
【０１１８】
次に、プロセッサは、ｓｃａｔｔｅｒ命令ｓｃａｔｔｅｒ　ｒｅｇ６，（ｘ１，ｒｅｇｘ），３を実行することで、ｒｅｇ６から３個の要素Ｃ１～Ｃ３を読み出して、ｒｅｇｘ内の３個の要素が示すメモリ１７０１のアドレスに書き込む。このｓｃａｔｔｅｒ命令は、ｒｅｇ６から配列Ｃのデータを読み出し、メモリ１７０１に格納された各構造体配列に含まれる配列Ｃのデータの位置を指定して、読み出したデータを指定した位置に書き込む命令である。
【０１１９】
プロセッサは、配列Ｃの他の要素に対しても同様の命令を実行することで、Ｃ１～Ｃ８をメモリ１７０１に書き込むことができる。
【０１２０】
図１７の既存の命令を用いるＳＩＭＤ展開によれば、ＡＯＳ専用命令が定義されていない場合であっても、ＡＯＳ形式のデータに対するＳＩＭＤ演算を実行することが可能になる。
【０１２１】
図１８は、ＡＯＳ専用命令を用いるＳＩＭＤ展開の例を示している。ｒｅｇ１～ｒｅｇ３は、ＳＩＭＤレジスタであり、６４ビットのデータを８個格納することができる。
【０１２２】
図１８（ａ）は、専用ロード命令の例を示している。プロセッサは、専用ロード命令ｌｄ３　ｒｅｇ１，ｒｅｇ２，ｒｅｇ３，ａｄｄｒを実行することで、メモリ１７０１からＡＯＳ形式のデータＡ１～Ａ８を読み出して、ｒｅｇ１～ｒｅｇ３の垂直方向に順に書き込む。このように、専用ロード命令を用いることで、ＡＯＳ形式のデータを１命令でメモリ１７０１からｒｅｇ１～ｒｅｇ３へロードすることができる。
【０１２３】
図１８（ｂ）は、ｒｅｇ１～ｒｅｇ３を用いたＳＩＭＤ命令の例を示している。プロセッサは、ＳＩＭＤ命令ＡＤＤ　ｒｅｇ３，ｒｅｇ１，ｒｅｇ２を実行することで、ｒｅｇ１に格納されたＡｉ（ｉ＝１～８）と、ｒｅｇ２に格納されたＢｉとを並列に加算して、加算結果Ｃｉをｒｅｇ３に書き込む。
【０１２４】
図１８（ｃ）は、専用ストア命令の例を示している。まず、プロセッサは、命令ｌｄｒｘ１，＆Ａ１を実行することで、メモリ１７０１内におけるＡｉ、Ｂｉ、及びＣｉ（ｉ＝１～８）の格納先先頭アドレスを計算して、不図示のレジスタｘ１に書き込む。
【０１２５】
次に、プロセッサは、専用ストア命令ｓｔ３　ｒｅｇ１，ｒｅｇ２，ｒｅｇ３，（ｘ１）を実行することで、ｒｅｇ１～ｒｅｇ３から、Ａｉ、Ｂｉ、及びＣｉを垂直方向に順に読み出して、ＡＯＳ形式でメモリ１７０１に書き込む。ｒｅｇ１内のＡｉとｒｅｇ２内のＢｉは、メモリ１７０１内に既に格納されているが、専用ストア命令はｒｅｇ１～ｒｅｇ３を読み出す命令であるため、ｒｅｇ３内のＣｉとともにメモリ１７０１に書き込まれる。このように、専用ストア命令を用いることで、ＡＯＳ形式のデータを１命令でｒｅｇ１～ｒｅｇ３からメモリ１７０１に格納することができる。
【０１２６】
図１８のＡＯＳ専用命令を用いるＳＩＭＤ展開によれば、図１７の既存の命令を用いるＳＩＭＤ展開と比較して、ＳＩＭＤ演算のための命令の個数が少ないため、ＳＩＭＤ演算を効率良く実行することが可能になる。
【０１２７】

次に、図１９から図２４までを参照しながら、図６のコード変換装置４０１が行うコード変換処理の手順について説明する。

【０１２８】

図１９は、コード変換処理の具体例を示すフローチャートである。まず、ユーザは、変換候補の抽出方法として、抽出方法Ｍ１～抽出方法Ｍ３のいずれかを指定し、最適化指示を入力する（ステップ１９０１）。

【０１２９】

ユーザが抽出方法Ｍ１を指定した場合、コード変換装置４０１は、指定されたコンパイラオプションに従って、変換候補を抽出する。ユーザが抽出方法Ｍ２を指定した場合、コード変換装置４０１は、ソースコード６２１に記述された制御文に従って、変換候補を抽出する。ユーザが抽出方法Ｍ３を指定した場合、コード変換装置４０１は、指定されたプロファイル情報を用いて、変換候補を抽出する。

【０１３０】

次に、解析部６１１は、ソースコード６２１を解析し、解析結果に基づいてループ管理テーブル６２３を生成する（ステップ１９０２）。

【０１３１】

次に、変換部４１２は、ループ管理テーブル６２３にグループ番号を設定し（ステップ１９０３）、ループ管理テーブル６２３を参照して、変換候補の配列を抽出する（ステップ１９０４）。そして、変換部４１２は、変換候補の配列の中から変換対象の配列を選択し（ステップ１９０５）、変換対象の配列に対する変換処理を行って、中間コード６２５を生成する（ステップ１９０６）。

【０１３２】

次に、生成部４１３は、中間コード６２５に対するＳＩＭＤ展開を行って、機械語コード６２６を生成する（ステップ１９０７）。

【０１３３】

図２０は、図１９のステップ１９０３におけるグループ番号設定処理の例を示すフローチャートである。まず、変換部４１２は、ループ管理テーブル６２３から同じ次元数を有する変数を抽出する（ステップ２００１）。例えば、図１３（ｂ）のループ管理テーブル６２３の場合、同じ次元数を有する変数として、以下の変数が抽出される。

【０１３４】

１次元の変数：
Ａ（ｉ），Ａ（ｉ＋２），Ｂ（ｉ），α（Ｗ（ｉ）），β（Ｗ（ｉ））
２次元の変数：
ＡＡ（ｉ，ｊ），ＡＡ（ｉ＋１，ｊ），ＡＡ（ｘ，ｚ），ＢＢ（ｉ，ｊ），
ＢＢ（ｉ＋２，ｊ），ＢＢ（ｚ，ｃ），ＣＣ（ｘ，ｙ），ＤＤ（ｘ，ｙ）

【０１３５】

次に、変換部４１２は、同じ次元数を有する変数の中から、同じ添え字を有する変数を抽出し、抽出した変数のグループを生成する（ステップ２００２）。これにより、以下の４個のグループが生成される。

【０１３６】

｛Ａ（ｉ），Ｂ（ｉ）｝
｛ＡＡ（ｉ，ｊ），ＢＢ（ｉ，ｊ）｝
｛ＣＣ（ｘ，ｙ），ＤＤ（ｘ，ｙ）｝
｛α（Ｗ（ｉ）），β（Ｗ（ｉ））｝

【０１３７】

次に、変換部４１２は、各グループにグループ番号を設定する（ステップ２００３）。これにより、以下のようなグループ番号が設定される。

【０１３８】

グループ番号“１”：｛Ａ（ｉ），Ｂ（ｉ）｝
グループ番号“２”：｛ＡＡ（ｉ，ｊ），ＢＢ（ｉ，ｊ）｝

グループ番号“３”：｛ＣＣ（ｘ，ｙ），ＤＤ（ｘ，ｙ）｝
グループ番号“４”：｛α（Ｗ（ｉ））），β（Ｗ（ｉ））｝
【０１３９】
図２１は、図１９のステップ１９０４における変換候補抽出処理の例を示すフローチャートである。まず、変換部４１２は、コンパイラオプション又は制御文により複数の配列が明示的に指定されているか否かをチェックする（ステップ２１０１）。
【０１４０】
複数の配列が明示的に指定されている場合（ステップ２１０１，ＹＥＳ）、変換部４１２は、指定された複数の配列が、ループ管理テーブル６２３内で同じグループ番号を有するか否かをチェックする（ステップ２１０２）。指定された複数の配列が同じグループ番号を有する場合（ステップ２１０２，ＹＥＳ）、変換部４１２は、それらの配列を変換候補に決定する（ステップ２１０３）。一方、指定された複数の配列が同じグループ番号を有さない場合、変換部４１２は、それらの配列を変換候補から除外する（ステップ２１０４）。
【０１４１】
複数の配列が明示的に指定されていない場合（ステップ２１０１，ＮＯ）、変換部４１２は、生成されたグループの中から変換候補を抽出する。例えば、コンパイラオプション－ＫＡｏｓが指定されている場合、いずれかのグループに分類されたすべての配列が、変換候補として抽出される。また、プロファイル情報が指定されている場合、同時にアクセスされる頻度が高い複数の配列が、変換候補として抽出される。
【０１４２】
図２２は、図１９のステップ１９０５における変換対象選択処理の例を示すフローチャートである。まず、変換部４１２は、評価関数６２２を用いて、変換候補の各配列に対する評価値を計算し、計算した評価値を含む評価値テーブル６２４を生成する（ステップ２２０１）。そして、変換部４１２は、評価値テーブル６２４を参照して、評価値合計が閾値よりも大きなグループに属する配列を、変換対象として選択する（ステップ２２０２）。
【０１４３】
図２３は、図１９のステップ１９０６における中間コード生成処理の例を示すフローチャートである。まず、変換部４１２は、ソースコード６２１に含まれる変換対象の配列のデータ定義を、グループ毎にＡＯＳ形式のデータ定義に変更し、構造体配列を定義する（ステップ２３０１）。構造体配列の添え字及び要素数としては、変更前の各配列の添え字及び要素数が用いられる。
【０１４４】
次に、変換部４１２は、ソースコード６２１に含まれる変換対象の配列の記述を、構造体配列を用いた記述に変更する（ステップ２３０２）。そして、変換部４１２は、ＡＯＳ形式のデータ定義及び構造体配列の記述を含むコードをコンパイルすることで、中間コード６２５を生成する（ステップ２３０３）。
【０１４５】
図２４は、図１９のステップ１９０７における機械語コード生成処理の例を示すフローチャートである。まず、生成部４１３は、専用ロード命令及び専用ストア命令を含む、ＡＯＳ専用命令が定義されているか否かをチェックする（ステップ２４０１）。
【０１４６】
ＡＯＳ専用命令が定義されている場合（ステップ２４０１，ＹＥＳ）、生成部４１３は、専用ロード命令を機械語コード６２６に記述する（ステップ２４０２）。一方、ＡＯＳ専用命令が定義されていない場合（ステップ２４０１，ＮＯ）、生成部４１３は、既存の命令を組み合わせて、メモリからＡＯＳ形式のデータを読み出してＳＩＭＤレジスタに書き込む処理を、機械語コード６２６に記述する（ステップ２４０３）。
【０１４７】
次に、生成部４１３は、ＳＩＭＤレジスタを用いるＳＩＭＤ命令を機械語コード６２６に記述し（ステップ２４０４）、ＡＯＳ専用命令が定義されているか否かをチェックする（

ステップ２４０５）。
【０１４８】
ＡＯＳ専用命令が定義されている場合（ステップ２４０５，ＹＥＳ）、生成部４１３は、専用ストア命令を機械語コード６２６に記述する（ステップ２４０６）。一方、ＡＯＳ専用命令が定義されていない場合（ステップ２４０５，ＮＯ）、生成部４１３は、既存の命令を組み合わせて、ＳＩＭＤレジスタからＡＯＳ形式のデータを読み出してメモリに書き込む処理を、機械語コード６２６に記述する（ステップ２４０７）。
【０１４９】
そして、生成部４１３は、専用ロード命令又は既存の命令と、ＳＩＭＤ命令と、専用ストア命令又は既存の命令とを含む、機械語コード６２６を生成する（ステップ２４０８）。
【０１５０】
図４及び図６のコード変換装置４０１の構成は一例に過ぎず、コード変換装置４０１の用途又は条件に応じて、一部の構成要素を省略又は変更してもよい。例えば、図６のコード変換装置４０１において、ループ管理テーブル６２３が外部の装置によって生成される場合は、解析部６１１を省略することができる。
【０１５１】
図５及び図１９～図２４のフローチャートは一例に過ぎず、コード変換装置４０１の構成又は条件に応じて一部の処理を省略又は変更してもよい。例えば、ループ管理テーブル６２３が外部の装置によって生成される場合は、図１９のステップ１９０２の処理を省略することができる。コンパイラオプション又は制御文により複数の配列が明示的に指定されていない場合は、図１９のステップ１９０４の処理を省略することができる。
【０１５２】
図１～図３に示したＳＯＡ形式のデータ定義及びＳＩＭＤ展開は一例に過ぎず、ＳＯＡ形式のデータ定義及びＳＩＭＤ展開は、ソースコード６２１の言語と、ソースコード６２１に含まれる配列及び演算の種類とに応じて変化する。
【０１５３】
図７、図１０～図１２、及び図１５～図１８に示したＡＯＳ形式のデータ定義及びＳＩＭＤ展開は一例に過ぎず、ＡＯＳ形式のデータ及びＳＩＭＤ展開は、ソースコード６２１の言語と、ソースコード６２１に含まれる配列及び演算の種類とに応じて変化する。ソースコード６２１に含まれる演算は、加算、減算、乗算、除算等の複数の演算の組み合わせであってもよい。
【０１５４】
図８のハイブリッドＡＯＳ形式のデータ定義は一例に過ぎず、ハイブリッドＡＯＳ形式のデータ定義は、ソースコード６２１の言語と、ソースコード６２１に含まれる配列及び演算の種類とに応じて変化する。図９の処理性能は一例に過ぎず、処理性能は、ソースコード６２１に応じて変化する。
【０１５５】
図１３のループ管理テーブル６２３及び図１４の評価値テーブル６２４は一例に過ぎず、ループ管理テーブル６２３及び評価値テーブル６２４は、ソースコード６２１に含まれる配列の種類及び個数に応じて変化する。コード変換装置４０１の構成又は条件に応じて、ループ管理テーブル６２３及び評価値テーブル６２４の一部の項目を省略又は変更してもよい。
【０１５６】
式（１）の評価関数Ｅ（Ｑ）は一例に過ぎず、別の評価関数Ｅ（Ｑ）を用いて配列の評価値を計算してもよい。
【０１５７】
図２５は、図４及び図６のコード変換装置４０１として用いられる情報処理装置（コンピュータ）のハードウェア構成例を示している。図２５の情報処理装置は、ＣＰＵ（Central Processing Unit）２５０１、メモリ２５０２、入力装置２５０３、出力装置２５０４、補助記憶装置２５０５、媒体駆動装置２５０６、及びネットワーク接続装置２５０７を含

む。これらの構成要素はバス２５０８により互いに接続されている。
【０１５８】
メモリ２５０２は、例えば、ＲＯＭ（Read Only Memory）、ＲＡＭ（Random Access Memory）、フラッシュメモリ等の半導体メモリであり、処理に用いられるプログラム及びデータを格納する。メモリ２５０２は、図４及び図６の記憶部４１１として用いることができる。
【０１５９】
ＣＰＵ２５０１（プロセッサ）は、例えば、メモリ２５０２を利用してプログラムを実行することにより、図４及び図６の変換部４１２及び生成部４１３として動作する。ＣＰＵ２５０１は、メモリ２５０２を利用してプログラムを実行することにより、図６の解析部６１１としても動作する。
【０１６０】
入力装置２５０３は、例えば、キーボード、ポインティングデバイス等であり、オペレータ又はユーザからの指示又は情報の入力に用いられる。出力装置２５０４は、例えば、表示装置、プリンタ、スピーカ等であり、オペレータ又はユーザへの問い合わせ又は指示、及び処理結果の出力に用いられる。
【０１６１】
補助記憶装置２５０５は、例えば、磁気ディスク装置、光ディスク装置、光磁気ディスク装置、テープ装置等である。補助記憶装置２５０５は、ハードディスクドライブであってもよい。情報処理装置は、補助記憶装置２５０５にプログラム及びデータを格納しておき、それらをメモリ２５０２にロードして使用することができる。
【０１６２】
媒体駆動装置２５０６は、可搬型記録媒体２５０９を駆動し、その記録内容にアクセスする。可搬型記録媒体２５０９は、メモリデバイス、フレキシブルディスク、光ディスク、光磁気ディスク等である。可搬型記録媒体２５０９は、ＣＤ－ＲＯＭ（Compact Disk Read Only Memory）、ＤＶＤ（Digital Versatile Disk）、ＵＳＢ（Universal Serial Bus）メモリ等であってもよい。オペレータ又はユーザは、この可搬型記録媒体２５０９にプログラム及びデータを格納しておき、それらをメモリ２５０２にロードして使用することができる。
【０１６３】
このように、処理に用いられるプログラム及びデータを格納するコンピュータ読み取り可能な記録媒体は、メモリ２５０２、補助記憶装置２５０５、又は可搬型記録媒体２５０９のような、物理的な（非一時的な）記録媒体である。
【０１６４】
ネットワーク接続装置２５０７は、ＬＡＮ（Local Area Network）、ＷＡＮ（Wide Area Network）等の通信ネットワークに接続され、通信に伴うデータ変換を行う通信インタフェース回路である。情報処理装置は、プログラム及びデータを外部の装置からネットワーク接続装置２５０７を介して受信し、それらをメモリ２５０２にロードして使用することができる。
【０１６５】
なお、情報処理装置が図２５のすべての構成要素を含む必要はなく、用途又は条件に応じて一部の構成要素を省略することも可能である。例えば、ユーザ又はオペレータとのインタフェースが不要である場合は、入力装置２５０３及び出力装置２５０４を省略してもよい。また、可搬型記録媒体２５０９又は通信ネットワークを使用しない場合は、媒体駆動装置２５０６又はネットワーク接続装置２５０７を省略してもよい。
【０１６６】
開示の実施形態とその利点について詳しく説明したが、当業者は、特許請求の範囲に明確に記載した本発明の範囲から逸脱することなく、様々な変更、追加、省略をすることができるであろう。
【０１６７】

図１乃至図２５を参照しながら説明した実施形態に関し、さらに以下の付記を開示する。
（付記１）
複数の配列のデータ定義と、前記複数の配列に対する所定の演算と、前記所定の演算の演算結果を表す配列のデータ定義とを含む、第１コードを記憶する記憶部と、
前記第１コードに含まれる前記複数の配列のデータ定義と前記演算結果を表す配列のデータ定義とを、構造体配列のデータ定義に変換し、前記第１コードに含まれる前記所定の演算を、前記構造体配列に対する演算に変換する変換部と、
前記複数の配列各々の異なるデータに対して、前記構造体配列に対する演算を並列に実行する所定の命令を含む、第２コードを生成する生成部と、
を備えることを特徴とするコード変換装置。
（付記２）
前記所定の命令は、複数のレジスタそれぞれに格納された配列のデータに対して、前記構造体配列に対する演算を実行し、演算結果を表す配列のデータを所定のレジスタに書き込む命令であり、
前記第２コードは、
メモリに連続して格納された複数の構造体配列のデータを読み出して、各構造体配列のデータに含まれる前記複数の配列のデータを、前記複数のレジスタにそれぞれ書き込む命令と、
前記所定のレジスタから前記演算結果を表す配列のデータを読み出して、前記メモリに格納された各構造体配列のデータに含まれる、前記演算結果を表す配列のデータの位置に書き込む命令と、
をさらに含むことを特徴とする付記１記載のコード変換装置。
（付記３）
前記所定の命令は、複数のレジスタそれぞれに格納された配列のデータに対して、前記構造体配列に対する演算を実行し、演算結果を表す配列のデータを所定のレジスタに書き込む命令であり、
前記第２コードは、
メモリに連続して格納された複数の構造体配列のデータを読み出して、第１レジスタに連続して書き込む命令と、
前記第１レジスタから、各構造体配列のデータに含まれる同じ配列のデータの位置を指定して、指定した位置のデータを読み出し、前記複数のレジスタのうち同じレジスタに連続して書き込む命令と、
前記所定のレジスタから前記演算結果を表す配列のデータを読み出し、前記メモリに格納された各構造体配列のデータに含まれる、前記演算結果を表す配列のデータの位置を指定して、前記所定のレジスタから読み出したデータを前記メモリの指定した位置に書き込む命令と、
をさらに含むことを特徴とする付記１記載のコード変換装置。
（付記４）
前記変換部は、データ定義の変換を示すコンパイラオプションに従って、前記第１コードに含まれる配列の中から、前記複数の配列と前記演算結果を表す配列とを選択することを特徴とする付記１乃至３のいずれか１項に記載のコード変換装置。
（付記５）
前記第１コードは、前記複数の配列と前記演算結果を表す配列とを指定する制御文を含み、
前記変換部は、前記制御文に従って、前記第１コードに含まれる配列の中から、前記複数の配列と前記演算結果を表す配列とを選択することを特徴とする付記１乃至３のいずれか１項に記載のコード変換装置。
（付記６）
前記変換部は、前記第１コードに含まれる配列のアクセス頻度を示すプロファイル情報を用いて、前記第１コードに含まれる配列の中から、前記複数の配列と前記演算結果を表す配列とを選択することを特徴とする付記１乃至３のいずれか１項に記載のコード変換装置。

（付記７）
前記変換部は、前記第１コードに含まれるループ内における各配列の出現回数、又は前記ループ内で同じ添え字を有する配列のグループにおける各配列の出現回数のうち、少なくとも一方に基づいて、前記ループに含まれる配列の中から前記複数の配列を選択することを特徴とする付記１乃至６のいずれか１項に記載のコード変換装置。
（付記８）
前記第１コードに含まれる複数の配列のデータ定義は、配列構造体のデータ定義であることを特徴とする付記１乃至７のいずれか１項に記載のコード変換装置。
（付記９）
複数の配列のデータ定義と、前記複数の配列に対する所定の演算と、前記所定の演算の演算結果を表す配列のデータ定義とを含む、第１コードを記憶する記憶部を有するコンピュータが、
前記第１コードに含まれる前記複数の配列のデータ定義と前記演算結果を表す配列のデータ定義とを、構造体配列のデータ定義に変換し、
前記第１コードに含まれる前記所定の演算を、前記構造体配列に対する演算に変換し、
前記複数の配列各々の異なるデータに対して、前記構造体配列に対する演算を並列に実行する所定の命令を含む、第２コードを生成することを特徴とするコード変換方法。
（付記１０）
前記所定の命令は、複数のレジスタそれぞれに格納された配列のデータに対して、前記構造体配列に対する演算を実行し、演算結果を表す配列のデータを所定のレジスタに書き込む命令であり、
前記第２コードは、
メモリに連続して格納された複数の構造体配列のデータを読み出して、各構造体配列のデータに含まれる前記複数の配列のデータを、前記複数のレジスタにそれぞれ書き込む命令と、
前記所定のレジスタから前記演算結果を表す配列のデータを読み出して、前記メモリに格納された各構造体配列のデータに含まれる、前記演算結果を表す配列のデータの位置に書き込む命令と、
をさらに含むことを特徴とする付記９記載のコード変換方法。
（付記１１）
前記所定の命令は、複数のレジスタそれぞれに格納された配列のデータに対して、前記構造体配列に対する演算を実行し、演算結果を表す配列のデータを所定のレジスタに書き込む命令であり、
前記第２コードは、
メモリに連続して格納された複数の構造体配列のデータを読み出して、第１レジスタに連続して書き込む命令と、
前記第１レジスタから、各構造体配列のデータに含まれる同じ配列のデータの位置を指定して、指定した位置のデータを読み出し、前記複数のレジスタのうち同じレジスタに連続して書き込む命令と、
前記所定のレジスタから前記演算結果を表す配列のデータを読み出し、前記メモリに格納された各構造体配列のデータに含まれる、前記演算結果を表す配列のデータの位置を指定して、前記所定のレジスタから読み出したデータを前記メモリの指定した位置に書き込む命令と、
をさらに含むことを特徴とする付記９記載のコード変換方法。
（付記１２）
複数の配列のデータ定義と、前記複数の配列に対する所定の演算と、前記所定の演算の演算結果を表す配列のデータ定義とを含む、第１コードを記憶する記憶部を有するコンピュータに、
前記第１コードに含まれる前記複数の配列のデータ定義と前記演算結果を表す配列のデータ定義とを、構造体配列のデータ定義に変換させ、

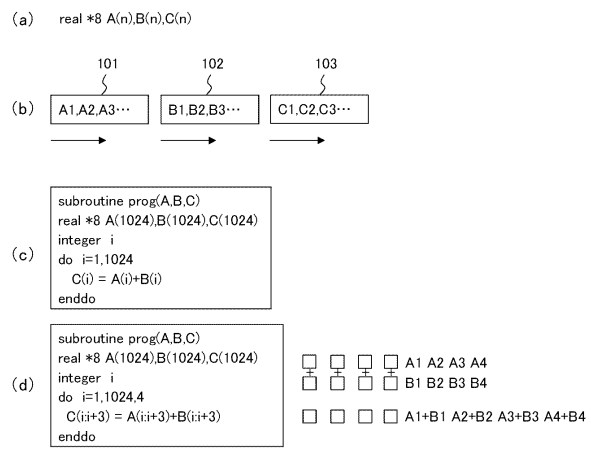
前記第１コードに含まれる前記所定の演算を、前記構造体配列に対する演算に変換させ、前記複数の配列各々の異なるデータに対して、前記構造体配列に対する演算を並列に実行する所定の命令を含む、第２コードを生成させることを特徴とするコード変換プログラム。
（付記１３）
前記所定の命令は、複数のレジスタそれぞれに格納された配列のデータに対して、前記構造体配列に対する演算を実行し、演算結果を表す配列のデータを所定のレジスタに書き込む命令であり、
前記第２コードは、
メモリに連続して格納された複数の構造体配列のデータを読み出して、各構造体配列のデータに含まれる前記複数の配列のデータを、前記複数のレジスタにそれぞれ書き込む命令と、
前記所定のレジスタから前記演算結果を表す配列のデータを読み出して、前記メモリに格納された各構造体配列のデータに含まれる、前記演算結果を表す配列のデータの位置に書き込む命令と、
をさらに含むことを特徴とする付記１２記載のコード変換プログラム。
（付記１４）
前記所定の命令は、複数のレジスタそれぞれに格納された配列のデータに対して、前記構造体配列に対する演算を実行し、演算結果を表す配列のデータを所定のレジスタに書き込む命令であり、
前記第２コードは、
メモリに連続して格納された複数の構造体配列のデータを読み出して、第１レジスタに連続して書き込む命令と、
前記第１レジスタから、各構造体配列のデータに含まれる同じ配列のデータの位置を指定して、指定した位置のデータを読み出し、前記複数のレジスタのうち同じレジスタに連続して書き込む命令と、
前記所定のレジスタから前記演算結果を表す配列のデータを読み出し、前記メモリに格納された各構造体配列のデータに含まれる、前記演算結果を表す配列のデータの位置を指定して、前記所定のレジスタから読み出したデータを前記メモリの指定した位置に書き込む命令と、
をさらに含むことを特徴とする付記１２記載のコード変換プログラム。
【符号の説明】
【０１６８】
１０１～１０３、３０１～３０３　領域
４０１　コード変換装置
４１１　記憶部
４１２　変換部
４１３　生成部
６１１　解析部
６２１　ソースコード
６２２　評価関数
６２３　ループ管理テーブル
６２４　評価値テーブル
６２５　中間コード
６２６　機械語コード
７０１、８０１、１７０１　メモリ
１２０１　プロファイル情報
２５０１　ＣＰＵ
２５０２　メモリ
２５０３　入力装置
２５０４　出力装置

２５０５　補助記憶装置
２５０６　媒体駆動装置
２５０７　ネットワーク接続装置
２５０８　バス
２５０９　可搬型記録媒体

【図面】

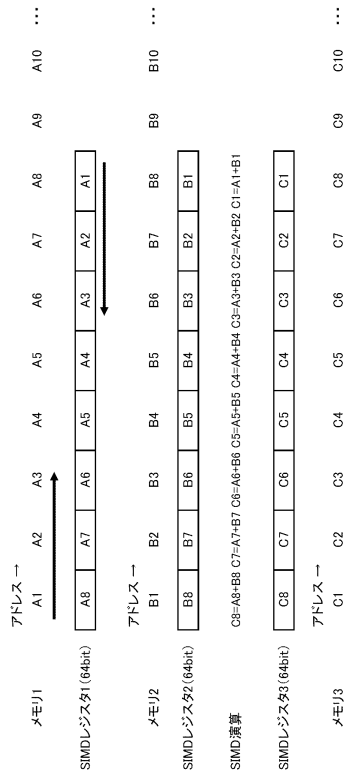
【図１】

SOA形式のデータに対するSIMD展開を示す図



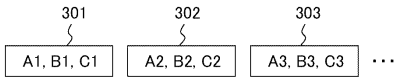
【図２】

SOA形式のデータに対するSIMD演算を示す図



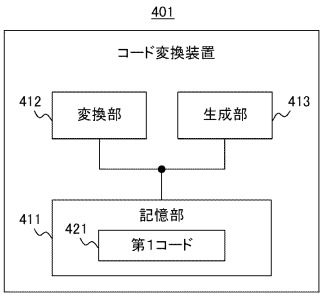
【図３】

配列の格納領域を示す図



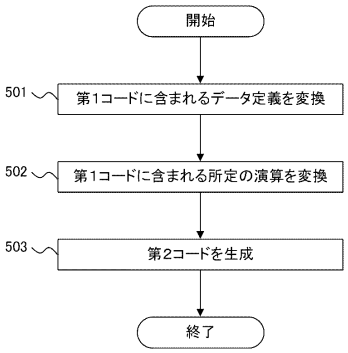
【図４】

コード変換装置の機能的構成図



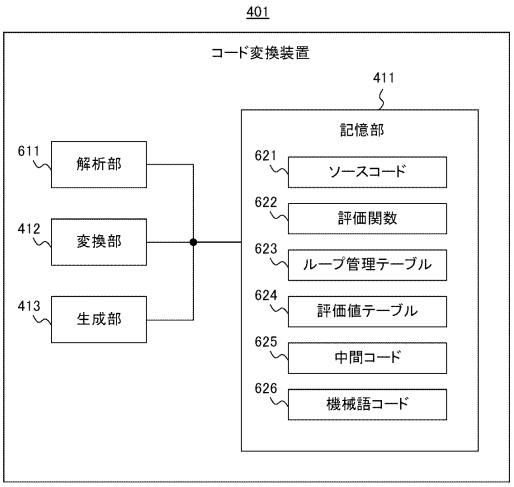
【図５】

コード変換処理のフローチャート



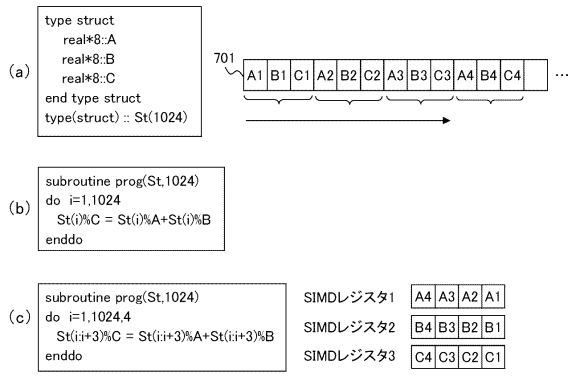
【図６】

コード変換装置の具体例を示す機能的構成図



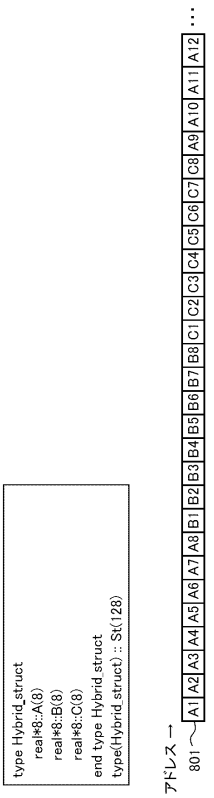
【図７】

AOS形式のデータ定義及びSIMD演算を示す図



【図８】

ハイブリッドAOS形式のデータ定義を示す図



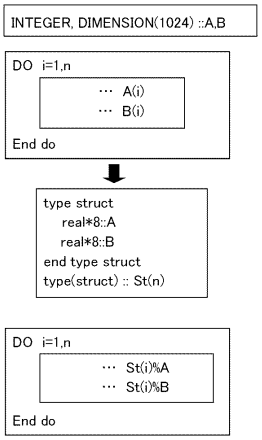
【図９】

処理性能を示す図

プログラム特性		データ定義	
		SOA	AOS
アクセスの局所性	なし	○	×
	あり	×	○
ハードウェア資源の負荷	HPF	×（大）	○（なし）
	TLBミス	×（大）	○（小）
プロセッサのSIMDサポート状況	水平方向	○	△
	垂直方向	×	○

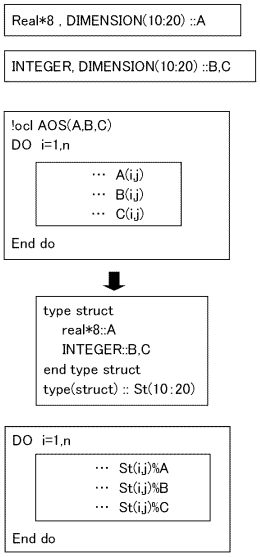
【図１０】

抽出方法M1を示す図



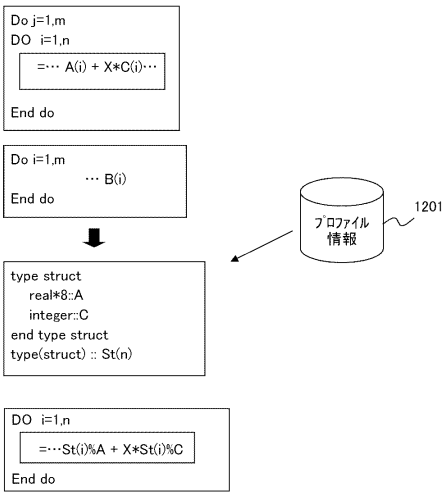
【図１１】

抽出方法M2を示す図



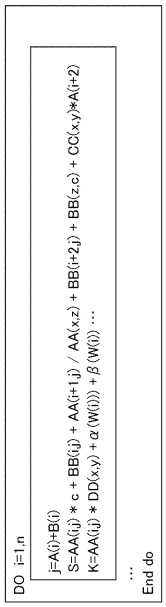
【図１２】

抽出方法M3を示す図



【図１３】

ループ管理テーブルを示す図



（a）

（b）

変数	配列名	次元数	出現回数	1次元	2次元	3次元	P次元	グループ番号
A(i)	A	1	1	i	–	–	…	1
A(i+2)	A	1	1	i+2	–	–	…	
B(i)	B	1	1	i	–	–	…	1
AA(i,j)	AA	2	2	i	j	–	…	2
AA(i+1,j)	AA	2	1	i+1	j	–	…	
AA(x,z)	AA	2	1	x	z	–	…	
BB(i,j)	BB	2	1	i	j	–	…	2
BB(i+2,j)	BB	2	1	i+2	j	–	…	
BB(z,c)	BB	2	1	z	c	–	…	
CC(x,y)	CC	2	1	x	y	–	…	3
DD(x,y)	DD	2	1	x	y	–	…	3
…	…	…	…	…	…	–	…	
α(W(i))	α	1	1	W(i)	–	–	…	4
β(W(i))	β	1	1	W(i)	–	–	…	4
…	…	…	…	…	…	–	…	

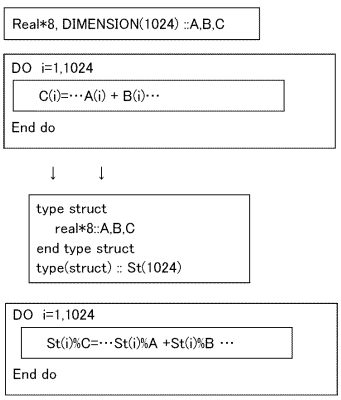
【図１４】

評価値テーブルを示す図

グループ番号	Q	C(Q)	w1	M(Q)	G(Q)	w2	評価値	評価値合計
1	A	2	1	1	2	2	1.14(2/14*1+1/2*2)	2.21
1	B	1	1	1	2	2	1.07(1/14*1+1/2*2)	2.21
2	AA	4	1	1	2	2	1.29(4/14*1+1/2*2)	2.5
2	BB	3	1	1	2	2	1.21(3/14*1+1/2*2)	2.5
3	CC	1	1	1	2	2	1.07(1/14*1+1/2*2)	2.14
3	DD	1	1	1	2	2	1.07(1/14*1+1/2*2)	2.14
4	α	1	1	1	2	2	1.07(1/14*1+1/2*2)	2.14
4	β	1	1	1	2	2	1.07(1/14*1+1/2*2)	2.14

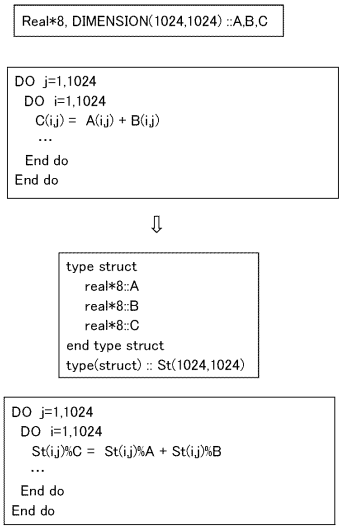
【図１５】

１次元の配列に対する変換処理を示す図



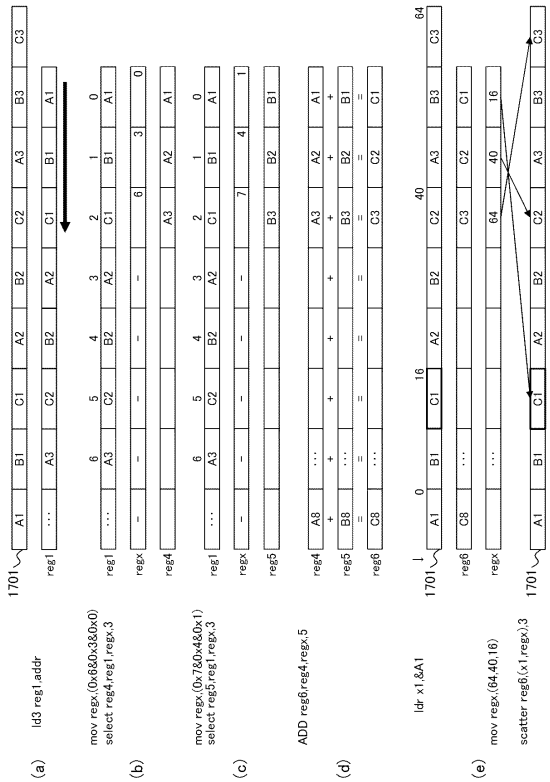
【図１６】

２次元の配列に対する変換処理を示す図



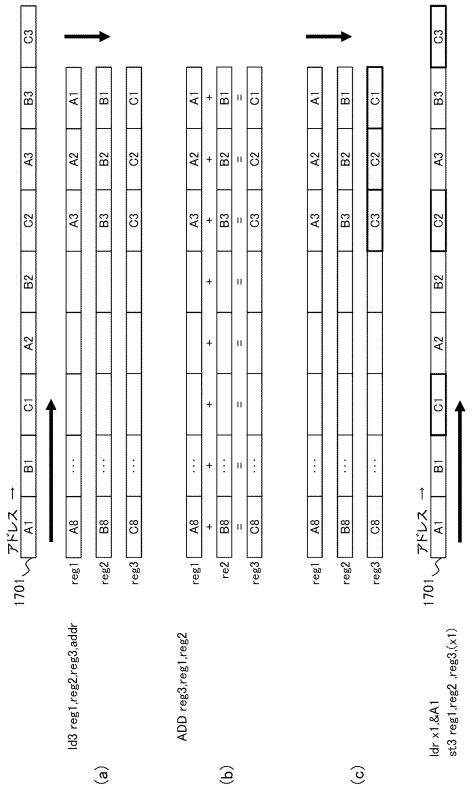
【図１７】

既存の命令を用いるSIMD展開を示す図



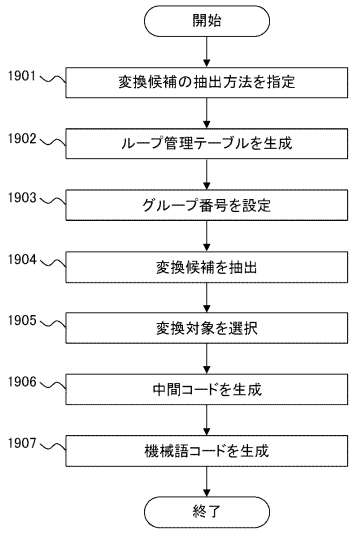
【図１８】

AOS専用命令を用いるSIMD展開を示す図



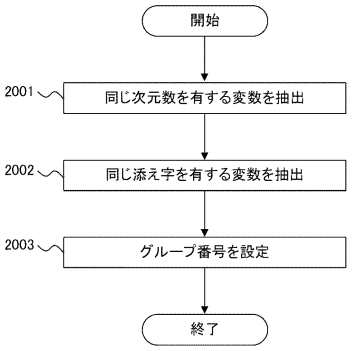
【図１９】

コード変換処理の具体例を示すフローチャート



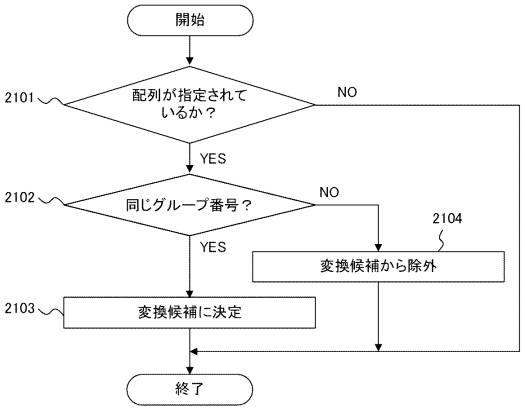
【図２０】

グループ番号設定処理のフローチャート



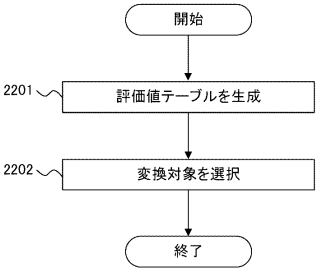
【図２１】

変換候補抽出処理のフローチャート



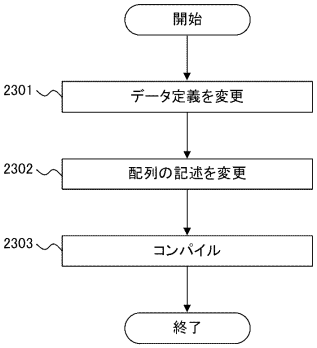
【図２２】

変換対象選択処理のフローチャート



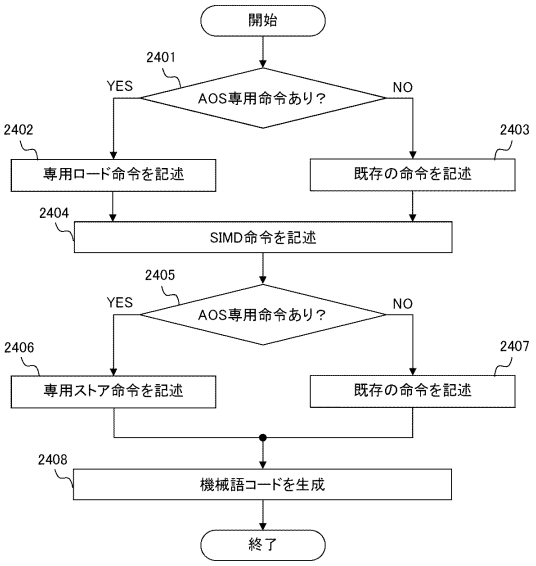
【図２３】

中間コード生成処理のフローチャート



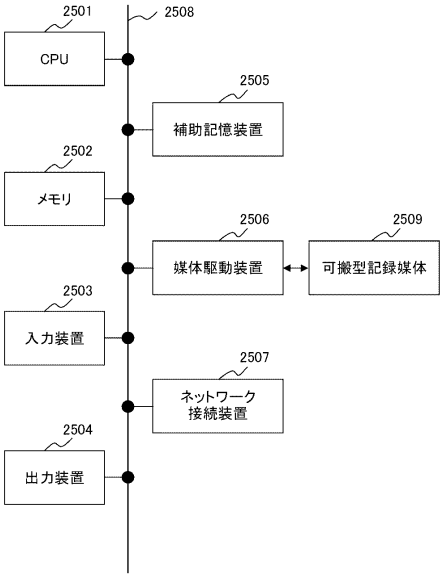
【図２４】

機械語コード生成処理のフローチャート



【図２５】

情報処理装置の構成図



フロントページの続き

(56)参考文献　特開平６－７５９８７（ＪＰ，Ａ）
特開平７－２８７０２（ＪＰ，Ａ）
特開２００５－１７４２９２（ＪＰ，Ａ）
特開２０１４－０３８６２４（ＪＰ，Ａ）
特開２０１５－２２５４２７（ＪＰ，Ａ）
特開２０１６－０８１１３５（ＪＰ，Ａ）
米国特許出願公開第２０１２／００８９７９２（ＵＳ，Ａ１）
米国特許出願公開第２０１５／０２９４４３５（ＵＳ，Ａ１）
米国特許出願公開第２０１７／０１７７３５６（ＵＳ，Ａ１）

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　８／４１
Ｇ０６Ｆ　９／３８
Ｇ０６Ｆ　９／３１２
Ｇ０６Ｆ　１７／１６