

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第6160259号
(P6160259)

(45) 発行日 平成29年7月12日 (2017. 7. 12)

(24) 登録日 平成29年6月23日 (2017. 6. 23)

(51) Int. Cl.

F 1

G 0 6 F 17/30 (2006.01)

G 0 6 F 17/30 3 5 0 A

G 0 6 F 17/30 1 7 0 A

請求項の数 5 (全 27 頁)

(21) 出願番号	特願2013-114677 (P2013-114677)	(73) 特許権者	000005223
(22) 出願日	平成25年5月30日 (2013. 5. 30)		富士通株式会社
(65) 公開番号	特開2014-235454 (P2014-235454A)		神奈川県川崎市中原区上小田中4丁目1番1号
(43) 公開日	平成26年12月15日 (2014. 12. 15)		
審査請求日	平成28年2月26日 (2016. 2. 26)	(74) 代理人	100104190
			弁理士 酒井 昭徳
		(72) 発明者	杉山 幸吉
			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
		(72) 発明者	佐野 貴之
			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
		(72) 発明者	伊藤 尚洋
			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

最終頁に続く

(54) 【発明の名称】 文字列探索方法、文字列探索装置および文字列探索プログラム

(57) 【特許請求の範囲】

【請求項 1】

コンピュータが、

探索文字列に対応する探索文字ビット列に含まれる 1 バイト の第 1 ビット列を、ビット列同士をバイト単位で一括比較する比較命令の処理単位を表す第 2 ビット長分第 1 レジスタに格納する処理と、

対象文字列に対応する対象文字ビット列に含まれる 前記第 2 ビット長の第 2 ビット列を第 2 レジスタに格納する処理と、

前記対象文字ビット列のうち前記第 2 ビット列の次の前記第 2 ビット長のビット列を第 3 レジスタに格納する処理と、

前記第 1 レジスタに格納されたビット列と、前記第 2 レジスタに格納されたビット列とをバイト単位で一括比較する第 1 比較と、

前記第 1 比較の比較結果に基づいて、前記探索文字ビット列のうち前記第 1 ビット列の次の 1 バイト の第 3 ビット列と、前記第 2 ビット列のうち前記第 1 ビット列と一致する箇所の次の 1 バイト の第 4 ビット列との比較と、

前記第 3 ビット列と前記第 4 ビット列とが一致する場合に、前記第 1 比較の比較結果に基づいて、前記第 2 レジスタに格納されたビット列を左シフトするとともにシフト元のビットをマスクし、前記第 3 レジスタに格納されたビット列を右シフトするとともにシフト元のビットをマスクし、前記第 2 レジスタに格納されたビット列と前記第 3 レジスタに格納されたビット列との論理和をとることにより、前記対象文字ビット列のうち前記第 1 ビ

ット列と一致する箇所から前記第2ビット長の第5ビット列を生成する処理と、

生成した前記第5ビット列と、前記探索文字ビット列のうち前記第1ビット列から前記第2ビット長の第6ビット列とをバイト単位で一括比較する第2比較と、

前記第2比較の比較結果に基づいて、前記対象文字列に前記探索文字列が含まれるか否かの判断と、

を実行することを特徴とする文字列探索方法。

【請求項2】

前記コンピュータが、

前記探索文字列および前記対象文字列が特定の文字コードのときは、前記第3ビット列と前記第4ビット列とが一致する場合、前記探索文字ビット列のうち前記第3ビット列の次の1バイトのビット列と、前記対象文字ビット列のうち前記第4ビット列の次の1バイトのビット列との比較を実行し、

前記第3ビット列の次のビット列と前記第4ビット列の次のビット列とが一致する場合に、前記第5ビット列の生成を実行することを特徴とする請求項1に記載の文字列探索方法。

【請求項3】

前記コンピュータが、

前記探索文字列および前記対象文字列がマルチバイト文字の場合、前記第1比較の比較結果に基づいて、前記第2ビット列のうち前記第1ビット列と一致する箇所がマルチバイト文字の先頭バイトか否かの判断を実行し、

前記第1ビット列と一致する箇所がマルチバイト文字の先頭バイトである場合に、前記第3ビット列と前記第4ビット列との比較を実行することを特徴とする請求項1または2に記載の文字列探索方法。

【請求項4】

探索文字列に対応する探索文字ビット列に含まれる1バイトの第1ビット列を、ビット列同士をバイト単位で一括比較する比較命令の処理単位を表す第2ビット長分第1レジスタに格納し、対象文字列に対応する対象文字ビット列に含まれる前記第2ビット長の第2ビット列を第2レジスタに格納し、前記対象文字ビット列のうち前記第2ビット列の次の前記第2ビット長のビット列を第3レジスタに格納する作成部と、

前記第1レジスタに格納されたビット列と、前記第2レジスタに格納されたビット列とをバイト単位で一括比較する第1の比較部と、

前記第1の比較部の比較結果に基づいて、前記探索文字ビット列のうち前記第1ビット列の次の1バイトの第3ビット列と、前記第2ビット列のうち前記第1ビット列と一致する箇所の次の1バイトの第4ビット列とを比較する第2の比較部と、

前記第3ビット列と前記第4ビット列とが一致する場合に、前記第1比較の比較結果に基づいて、前記第2レジスタに格納されたビット列を左シフトするとともにシフト元のビットをマスクし、前記第3レジスタに格納されたビット列を右シフトするとともにシフト元のビットをマスクし、前記第2レジスタに格納されたビット列と前記第3レジスタに格納されたビット列との論理和をとることにより、前記対象文字ビット列のうち前記第1ビット列と一致する箇所から前記第2ビット長の第5ビット列を生成する生成部と、

前記生成部によって生成された前記第5ビット列と、前記探索文字ビット列のうち前記第1ビット列から前記第2ビット長の第6ビット列とをバイト単位で一括比較する第3の比較部と、

前記第3の比較部の比較結果に基づいて、前記対象文字列に前記探索文字列が含まれるか否かを判断する判断部と、

を有することを特徴とする文字列探索装置。

【請求項5】

コンピュータに、

探索文字列に対応する探索文字ビット列に含まれる1バイトの第1ビット列を、ビット列同士をバイト単位で一括比較する比較命令の処理単位を表す第2ビット長分第1レジス

10

20

30

40

50

タに格納する処理と、

対象文字列に対応する対象文字ビット列に含まれる前記第2ビット長の第2ビット列を第2レジスタに格納する処理と、

前記対象文字ビット列のうち前記第2ビット列の次の前記第2ビット長のビット列を第3レジスタに格納する処理と、

前記第1レジスタに格納されたビット列と、前記第2レジスタに格納されたビット列とをバイト単位で一括比較する第1比較と、

前記第1比較の比較結果に基づいて、前記探索文字ビット列のうち前記第1ビット列の次の1バイトの第3ビット列と、前記第2ビット列のうち前記第1ビット列と一致する箇所の次の1バイトの第4ビット列との比較と、

10

前記第3ビット列と前記第4ビット列とが一致する場合に、前記第1比較の比較結果に基づいて、前記第2レジスタに格納されたビット列を左シフトするとともにシフト元のビットをマスクし、前記第3レジスタに格納されたビット列を右シフトするとともにシフト元のビットをマスクし、前記第2レジスタに格納されたビット列と前記第3レジスタに格納されたビット列との論理和をとることにより、前記対象文字ビット列のうち前記第1ビット列と一致する箇所から前記第2ビット長の第5ビット列を生成する処理と、

生成した前記第5ビット列と、前記探索文字ビット列のうち前記第1ビット列から前記第2ビット長の第6ビット列とをバイト単位で一括比較する第2比較と、

前記第2比較の比較結果に基づいて、前記対象文字列に前記探索文字列が含まれるか否かの判断と、

20

を実行させることを特徴とする文字列探索プログラム。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、文字列探索方法、文字列探索装置および文字列探索プログラムに関する。

【背景技術】

【0002】

従来、文字列を検索する技術として、ターゲット文字列に対して、パターン文字列を1バイト単位で比較するものがある（以下、「従来技術1」と称する）。例えば、「The quick brown fox」というターゲット文字列に対して、「queen」というパターン文字列を検索する場合、5バイト目の「q」、6バイト目の「u」を比較後、7バイト目の「i」と「e」との比較で検索結果が異なると判断することができる。

30

【0003】

また、ターゲット文字列に対して、状態遷移図を用いて、パターン文字列の照合を行う技術がある（以下、「従来技術2」と称する）。状態遷移図は、状態が遷移していく様子を表現する図である。状態遷移図を用いて検索を行うことにより、例えば、上記の例では、7バイト目の「i」と「e」の比較が終わった後に、次の検索開始位置が8バイト目になるよう制御することができる。

【0004】

また、複数のレジスタを用いて、文字列を検索する検索処理の一部をハードウェア処理により実行する技術がある。具体的には、従来装置は、第1レジスタに照合または被照合の文字列のデータのうち長いデータ長のデータを保持し、第2レジスタに短いデータをシフト自在に保持する。また、従来装置は、第2レジスタに保持した文字列の先頭の文字のデータを、第1レジスタに保持した文字列分第3レジスタに一旦保持し、第1レジスタ、第3レジスタに保持した文字列のデータの排他的論理和の演算結果を保持する。また、従来装置は、この演算結果により第2レジスタの文字列をシフトさせ、第1レジスタ、第2レジスタに保持したデータの排他的論理和の演算結果と第2レジスタの文字列のシフト位置により検索出力の可否またはさらなるシフトの可否を判定する（以下、「従来技術3」と称する）。

40

【先行技術文献】

50

【特許文献】

【0005】

【特許文献1】特開2002-312364号公報

【発明の概要】

【発明が解決しようとする課題】

【0006】

しかしながら、従来技術によれば、ターゲット文字列やパターン文字列が長くなると、文字列探索にかかる処理時間が増大するという問題がある。例えば、従来技術1では、文字列の差異が見つかった場合、1バイト後方の文字列から再度検索処理を行うことになり、文字列探索にかかる処理時間の増大を招く。また、従来技術2では、検索を行うたびに状態遷移図を作成することになり、状態遷移図の作成にかかる処理負荷が増大する。また、従来技術3では、パターン文字列自体を加工する場合、同一文字列の再検索時には再度シフトを実施することになり、文字列探索にかかる処理時間の増大を招く。

10

【0007】

一つの側面では、本発明は、文字列探索にかかる処理時間の短縮化を図る文字列探索方法、文字列探索装置および文字列探索プログラム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】

【0008】

本発明の一側面によれば、探索文字列に対応する探索文字ビット列に含まれる第1ビット長の第1ビット列を、対象文字列に対応する対象文字ビット列に含まれる第2ビット長の第2ビット列と前記第1ビット長単位で比較する第1比較と、前記第1比較の比較結果に基づいて、前記探索文字ビット列のうち前記第1ビット列の次の前記第1ビット長の第3ビット列と、前記第2ビット列のうち前記第1ビット列と一致する箇所次の前記第1ビット長の第4ビット列との比較と、前記第3ビット列と前記第4ビット列とが一致する場合に、前記対象文字ビット列のうち前記第1ビット列と一致する箇所から前記第2ビット長の第5ビット列の生成と、生成した前記第5ビット列と、前記探索文字ビット列のうち前記第1ビット列から前記第2ビット長の第6ビット列とを前記第1ビット長単位で比較する第2比較と、前記第2比較の比較結果に基づいて、前記対象文字列に前記探索文字列が含まれるか否かの判断と、を実行する文字列探索方法、文字列探索装置および文字列探索プログラムが提案される。

20

30

【発明の効果】

【0009】

本発明の一態様によれば、文字列探索にかかる処理時間の短縮化を図ることができるという効果を奏する。

【図面の簡単な説明】

【0010】

【図1】図1は、実施の形態にかかる文字列探索方法の一実施例を示す説明図（その1）である。

【図2】図2は、実施の形態にかかる文字列探索方法の一実施例を示す説明図（その2）である。

40

【図3】図3は、システム300のシステム構成例を示す説明図である。

【図4】図4は、データベースサーバ301のハードウェア構成例を示すブロック図である。

【図5】図5は、データベースサーバ301の機能的構成例を示すブロック図である。

【図6】図6は、文字コードがSJISの場合の文字列探索処理例を示す説明図（その1）である。

【図7】図7は、文字コードがSJISの場合の文字列探索処理例を示す説明図（その2）である。

【図8】図8は、文字コードがSJISの場合の文字列探索処理例を示す説明図（その3）である。

50

【図 9】図 9 は、データベースサーバ 301 の文字列探索処理手順の一例を示すフローチャート（その 1）である。

【図 10】図 10 は、データベースサーバ 301 の文字列探索処理手順の一例を示すフローチャート（その 2）である。

【発明を実施するための形態】

【0011】

以下に図面を参照して、本発明にかかる文字列探索方法、文字列探索装置および文字列探索プログラムの実施の形態を詳細に説明する。

【0012】

（文字列探索方法の一実施例）

図 1 および図 2 は、実施の形態にかかる文字列探索方法の一実施例を示す説明図である。図 1 において、文字列探索装置 101 は、対象文字列から探索文字列を探索するコンピュータである。探索文字列は、探索する文字列である。対象文字列は、探索先の文字列である。

【0013】

ここで、対象文字列から探索文字列を探索する技術として、特定の比較命令（例えば、SIMD Compare）により、探索文字列の先頭文字と対象文字列とを一括比較して、対象文字列から探索文字列の先頭文字を検出し、探索文字列との文字列マッチングを行うことが考えられる。すなわち、対象文字列から探索文字列の先頭文字と一致する箇所を検出するという一次スクリーニングを行った後に、より詳細な文字列探索を行うことが

【0014】

SIMD (Single Instruction Multiple Data) 命令とは、1 命令で複数のデータを処理するものである。ただし、SIMD 命令を用いる場合には、8 バイトのバウンダリを守らなければならない。このため、探索文字列の先頭文字と対象文字列とを一括比較した比較結果に応じて、例えば、レジスタ内の対象文字列のデータを、Shift、Mask した上で Or をとり、SIMD 命令にて探索文字列のデータと比較できる形に加工する必要がある。

【0015】

この際、特定のビット操作命令（例えば、Shift-Mask-Or）を用いて、レジスタ内の対象文字列のデータを加工することにより、Shift、Mask、Or をそれぞれ独立した命令として実行する場合に比べて、データ加工にかかる処理時間を短縮することができる。

【0016】

しかし、偶然先頭文字のみが一致したものについても、個別に Shift-Mask-Or と SIMD Compare を行うことは、先頭文字が出現頻度の多い文字である場合、詳細探索を行って外れることによるペナルティが大きくなる。また、文字には、マルチバイト文字と呼ばれる 1 文字を複数バイトで表すものがある。

【0017】

マルチバイト文字においては、文字が完全に一致しなくても、ビット列が一致する組み合わせが多数存在し、SIMD Compare によるスクリーニングだけでは、先頭文字さえも一致しない余分な候補に対して、個別に Shift-Mask-Or と SIMD Compare を行うことになる場合がある。

【0018】

マルチバイト文字の文字コードとしては、例えば、SJIS (Shift_JIS) や UTF-8 (Unicode Transformation Format-8) などがある。一方、1 文字を 1 バイトで表すシングルバイト文字の文字コードとしては、例えば、ASCII (American Standard Code for Information Interchange) がある。

【0019】

ここで、ペナルティが大きくなる場合の具体例として、文字コードがS J I Sの場合を例に挙げて説明する。S J I Sは、1文字を2バイトで表す文字コードである。また、対象文字列を「デジタルテレビの普及」とし、探索文字列を「テレビ」とする。対象文字列「デジタルテレビの普及」および探索文字列「テレビ」をS J I Sで表すと以下のようになる。ただし、各文字の右側の括弧内の数字が各文字を表すデータ（0 x ~の~の部分）である。

【0020】

[対象文字列] デ（83 66）ジ（83 57）タ（83 5E）ル（83 8B）ー（81 7C）テ（83 65）レ（83 8C）ビ（83 72）の（82 CC）普（95 81）及（8B 79）

10

【0021】

[探索文字列] テ（83 65）レ（83 8C）ビ（83 72）

【0022】

S J I Sではカタカナの先頭の1バイトは「83」となるため、探索文字列の先頭1バイトだけの確認では、対象文字列に含まれるすべてのカタカナがヒットしてしまう。ヒットする度に、対象文字列を8バイトの文字列として加工し、SIMD命令により8バイトでの比較を実施すると、文字列探索にかかる処理時間の増大を招いてしまう。

【0023】

そこで、本実施の形態では、文字列探索装置101は、例えば、SIMD命令を用いて探索文字列の先頭バイトと一致する箇所を対象文字列から検出し、その次のバイトも探索文字列と一致すれば、SIMD命令で比較可能にビット列を加工して詳細探索を行う。これにより、無駄な詳細探索を抑制して文字列探索にかかる処理時間の短縮化を図る。

20

【0024】

以下、文字列探索装置101の文字列探索処理例について説明する。ここでは、文字コードを「ASCII」とし、探索文字列を「quick」とし、対象文字列を「The quick brown fox jumps over the lazy dog」とする。

【0025】

(1) 文字列探索装置101は、探索文字列を表すビット列に含まれる第1ビット長11のビット列b1を、レジスタR1に第2ビット長12分格納する。ここで、第1ビット長11は、特定の比較命令（例えば、SIMD Compare）の処理単位となるビット数である。第1ビット長11は、例えば、1バイト（8ビット）、2バイト、3バイトなどである。

30

【0026】

また、第2ビット長12は、第1ビット長11のX倍のビット長であり（X：2以上の自然数）、特定の比較命令（例えば、SIMD Compare）において1回で処理されるデータのビット長である。第2ビット長12は、例えば、8バイト、16バイト、32バイトなどである。以下の説明では、第1ビット長11を「1バイト」とし、第2ビット長12を「8バイト」とする。

【0027】

具体的には、例えば、文字列探索装置101は、探索文字列を表すビット列の先頭から1バイト分のビット列b1をレジスタR1に8バイト分格納する。図1の例では、探索文字列の先頭文字「q」を表すビット列b1「01110001」がレジスタR1に8個格納されている。

40

【0028】

また、文字列探索装置101は、対象文字列を表すビット列の先頭から8バイト分のビット列b2をレジスタR2に格納する。また、文字列探索装置101は、対象文字列を表すビット列のうちビット列b2の次の8バイト分のビット列b3をレジスタR3に格納する。

【0029】

50

なお、対象文字列を表すビット列の先頭から 8 バイト分のビット列がない場合は、対象文字列を表すビット列の先頭から末尾までのビット列がレジスタ R 2 に格納される。また、対象文字列を表すビット列のうちビット列 b 2 の次の 8 バイト分のビット列がない場合は、対象文字列を表すビット列のうちビット列 b 2 の次ビットから末尾までのビット列がレジスタ R 3 に格納される。

【0030】

図 1 の例では、対象文字列の先頭から 8 文字の文字列「The(br)quic」を表すビット列 b 2 がレジスタ R 2 に格納されている。なお、(br) は、ブランクを表す。また、対象文字列の 9 文字目から 8 文字の文字列「k(br)brown(br)」を表すビット列 b 3 がレジスタ R 3 に格納されている。

10

【0031】

また、文字列探索装置 101 は、探索文字列を表すビット列の先頭から 8 バイト分のビット列 b 4 をレジスタ R 4 に格納する。なお、探索文字列を表すビット列の先頭から 8 バイト分のビット列がない場合は、探索文字列を表すビット列の先頭から末尾までのビット列 b 4 がレジスタ R 4 に格納される。

【0032】

図 1 の例では、探索文字列の先頭から 5 文字の文字列「quick」を表すビット列 b 4 がレジスタ R 4 に格納されている。また、各レジスタ R に記憶される各ビットの初期値は「0」である。このため、レジスタ R 4 の 6 バイト以降の各ビットの値は「0」となっている。

20

【0033】

(2) 文字列探索装置 101 は、探索文字列を表すビット列に含まれる 1 バイト分のビット列 b 1 を、対象文字列を表すビット列に含まれる 8 バイト分のビット列 b 2 とバイト単位で一括比較する第 1 比較命令を実行する。具体的には、例えば、文字列探索装置 101 は、SIMD Compare を実行することにより、レジスタ R 1 に格納されたビット列とレジスタ R 2 に格納されたビット列とをバイト単位で一括比較する。SIMD Compare によれば、1 回の命令で複数個のデータ同士 (8 ビットの符号なし整数) を一括比較することができる。

【0034】

図 2 の例では、第 1 の比較命令の比較結果を表すビット列 b 5 がレジスタ R 5 に格納されている。具体的には、ビット列 b 5 は、第 1 の比較命令のバイト単位の比較結果を表す。ここでは、「1」の場合、レジスタ R 1, R 2 のバイト単位のデータ同士が一致したことを示し、「0」の場合、レジスタ R 1, R 2 のバイト単位のデータ同士が不一致であることを示す。なお、レジスタ R 内の初期値の部分については、例えば、データ同士の比較が行われず、あるいは、比較結果が不一致となる。

30

【0035】

ビット列 b 5 によれば、対象文字列を表すビット列における探索文字列の先頭文字を表すビット列 b 1 の出現位置を特定することができる。ここでは、対象文字列を表すビット列の 5 バイト目に、探索文字列の先頭文字「q」を表すビット列 b 1 があることがわかる。

40

【0036】

(3) 文字列探索装置 101 は、第 1 比較命令の比較結果に基づいて、探索文字列を表すビット列のうちビット列 b 1 の次の 1 バイト分のビット列 b 6 と、ビット列 b 2 のうちビット列 b 1 と一致する箇所次の 1 バイト分のビット列 b 7 とを比較する。ここで、ビット列 b 6 は、探索文字列の 2 文字目の文字「u」を表すビット列である。また、ビット列 b 7 は、対象文字列のうち、探索文字列の先頭文字「q」と一致する 5 文字目の文字「q」の次の文字「u」を表すビット列である。

【0037】

すなわち、文字列探索装置 101 は、探索文字列を表すビット列の先頭バイトの次のバイトも対象文字列を表すビット列にヒットしているか否かを確認する。図 1 の例では、ビ

50

ット列b6「01110101」と、ビット列b7「01110101」とが一致する。これにより、探索文字列の先頭文字の次の文字も対象文字列にヒットしていることがわかる。

【0038】

(4) 文字列探索装置101は、ビット列b6とビット列b7とが一致する場合に、対象文字列を表すビット列のうちビット列b1と一致する箇所から8バイト分のビット列b8を生成するビット操作命令を実行する。ここで、ビット操作命令とは、例えば、Shift-Mask-Orである。

【0039】

Shift-Mask-Orは、2つのレジスタに対して「Shift、Mask、Or」を1つの命令で実行するものである。Shiftは、ビット列を左または右にずらす(シフト)操作である。Maskは、あるビットをオン(1)にしたりオフ(0)にしたりする操作である。Orは、2つの同じ長さのビット列に対して、同じ位置のビットごとに論理和(Or)をとって同じ長さのビット列を出力する操作である。

【0040】

具体的には、例えば、まず、文字列探索装置101は、ビット列b1と一致する箇所が先頭にくるようにレジスタR2を左シフトする。より具体的には、例えば、文字列探索装置101は、第1の比較命令の比較結果を表すビット列b5の先頭から「1」が出現するまでに「0」が出現した数分、レジスタR2を左シフトする。図2の例では、レジスタR2が4バイト左シフトされている。そして、文字列探索装置101は、レジスタR2のシフト元のビット(空いたバイト)を「0」でマスクする。図2の例では、レジスタR2の右側4バイトのビット列が「0」でマスクされている。

【0041】

また、文字列探索装置101は、レジスタサイズ(8バイト)から、ビット列b5の先頭から「1」が出現するまでに「0」が出現した数分引いた値で、レジスタR3を右シフトする。図2の例では、レジスタR3が4バイト右シフトされている。そして、文字列探索装置101は、レジスタR3のシフト元のビット(空いたバイト)を「0」でマスクする。図2の例では、レジスタR3の左側4バイトのビット列が「0」でマスクされている。

【0042】

そして、文字列探索装置101は、レジスタR2のビット列とレジスタR3のビット列とをOr命令で結合したビット列b8をレジスタR6に格納する。これにより、対象文字列を表すビット列のうち、探索文字列の先頭文字と一致する「q」から始まる8バイトのビット列b8を生成することができる。

【0043】

(5) 文字列探索装置101は、生成したビット列b8と、探索文字列を表すビット列のうち先頭から8バイト分のビット列b4とをバイト単位で一括比較する第2比較命令を実行する。具体的には、例えば、文字列探索装置101は、SIMD Compareを実行することにより、レジスタR6に格納されたビット列b8と、レジスタR4に格納されたビット列b4とを1バイト単位で一括比較する。

【0044】

図2の例では、第2の比較命令の比較結果を表すビット列b9がレジスタR7に格納されている。具体的には、ビット列b9は、第2の比較命令のバイト単位の比較結果を表す。ここでは、「1」の場合、レジスタR4、R6のバイト単位のデータ同士が一致したことを示し、「0」の場合、レジスタR4、R6のバイト単位のデータ同士が不一致であることを示す。

【0045】

(6) 文字列探索装置101は、第2比較命令の比較結果に基づいて、対象文字列に探索文字列が含まれるか否かを判断する。図2の例では、ビット列b9の先頭から「1」が5個以上続いており、比較して一致した文字数が5文字以上連続していることを示してい

10

20

30

40

50

る。また、探索文字列は5文字である。この場合、文字列探索装置101は、対象文字列に探索文字列が含まれると判断する。

【0046】

このように、文字列探索装置101によれば、特定の比較命令（例えば、SIMD Compare）を用いて一次スクリーニングを高速に行って、特定のビット操作命令（例えば、Shift-Mask-Or）および特定の比較命令を用いた文字列の詳細探索を効率的に行うことができる。

【0047】

これにより、無駄な詳細探索を抑制して文字列探索にかかる処理時間の短縮化を図ることができる。具体的には、例えば、シングルスバイト文字の場合は、偶然先頭文字のみが一致したものに対して、Shift-Mask-OrとSIMD Compareを行う無駄な処理を削減することができる。また、例えば、マルチバイト文字の場合は、先頭文字さえも一致しない候補に対して、個別にShift-Mask-OrとSIMD Compareを行う無駄な処理を削減することができる。

【0048】

（システム300のシステム構成例）

次に、実施の形態にかかるシステム300のシステム構成例について説明する。ここでは、図1に示した文字列探索装置101をシステム300内のデータベースサーバ301に適用した場合を例に挙げて説明する。

【0049】

図3は、システム300のシステム構成例を示す説明図である。図3において、システム300は、データベースサーバ301と、クライアント装置302と、を含む。システム300において、データベースサーバ301およびクライアント装置302は、有線または無線のネットワーク310を介して接続される。ネットワーク310は、例えば、LAN（Local Area Network）、WAN（Wide Area Network）、移動体通信網、インターネットなどである。

【0050】

データベースサーバ301は、データベース320およびログDB（データベース）330にアクセス可能なコンピュータである。データベース320は、複数のアプリケーションやユーザによって共有されるデータを記憶する。ログDB330は、テンポラリログやアーカイブログを記録する。テンポラリログは、データベース320の更新履歴を示すログである。アーカイブログは、データベース320を復旧する際に使用されるログである。

【0051】

データベースサーバ301は、クライアント装置302からデータの検索要求を受信すると、データベース320からデータをメモリ（例えば、後述の図4に示すメモリ402）に読み込む。そして、データベースサーバ301は、メモリに読み込まれたデータに対して検索を行い、その検索結果をクライアント装置302に送信する。

【0052】

より具体的には、例えば、データベースサーバ301は、不図示の受付部、データ操作部、レコード操作部、バッファ管理部、トランザクション管理部、更新管理部、ログ管理部等を有する。受付部は、クライアント装置302からのデータの検索要求を受け付ける。データの検索要求は、例えば、データベース320に対するSQL（Structured Query Language）操作である。

【0053】

データ操作部は、アクセス手順を作成し、トランザクション管理部、レコード操作部にデータアクセスを依頼する。レコード操作部は、レコードと呼ばれるデータの集まりの検索・更新を行う。なお、レコード操作部は、例えば、後述の図5に示すデータベースサーバ301の機能部に相当する。

【0054】

バッファ管理部は、データベースアクセスのメモリ上で管理するバッファアクセスを制御する。トランザクション管理部／更新管理部は、データベース320の変更の有効、無効の制御を行う。ログ管理部は、テンポラリログファイル、アーカイブログファイルのI/Oを管理する。

【0055】

クライアント装置302は、システム300のユーザが使用するコンピュータである。具体的には、例えば、クライアント装置302は、PC(Personal Computer)、ノートPC、スマートフォン、携帯電話機、タブレット型PCなどである。

【0056】

(データベースサーバ301のハードウェア構成例)

10

図4は、データベースサーバ301のハードウェア構成例を示すブロック図である。図4において、データベースサーバ301は、CPU(Central Processing Unit)401と、メモリ402と、I/F(Interface)403と、磁気ディスクドライブ404と、磁気ディスク405と、を有する。また、各構成部は、バス400によってそれぞれ接続される。

【0057】

ここで、CPU401は、データベースサーバ301の全体の制御を司る。メモリ402は、例えば、ROM(Read Only Memory)、RAM(Random Access Memory)およびフラッシュROMなどを有する。具体的には、例えば、フラッシュROMやROMが各種プログラムを記憶し、RAMがCPU401のワークエリアとして使用される。メモリ402に記憶されるプログラムは、CPU401にロードされることで、コーディングされている処理をCPU401に実行させる。

20

【0058】

I/F403は、通信回線を通じてネットワーク310に接続され、ネットワーク310を介して他のコンピュータ(例えば、図3に示したクライアント装置302)に接続される。そして、I/F403は、ネットワーク310と内部のインターフェースを司り、他のコンピュータからのデータの入出力を制御する。I/F403には、例えば、モデムやLANアダプタなどを採用することができる。

【0059】

磁気ディスクドライブ404は、CPU401の制御にしたがって磁気ディスク405に対するデータのリード/ライトを制御する。磁気ディスク405は、磁気ディスクドライブ404の制御で書き込まれたデータを記憶する。

30

【0060】

なお、データベースサーバ301は、上述した構成部のほか、例えば、SSD(Solid State Drive)、キーボード、マウス、ディスプレイなどを有することにしてもよい。また、図3に示したクライアント装置302についても、上述したデータベースサーバ301と同様のハードウェア構成例により実現することができる。

【0061】

(データベースサーバ301の機能的構成例)

図5は、データベースサーバ301の機能的構成例を示すブロック図である。図5において、データベースサーバ301は、取得部501と、作成部502と、第1の比較部503と、第2の比較部504と、生成部505と、第3の比較部506と、判断部507と、出力部508と、を含む構成である。各機能部は、具体的には、例えば、図4に示したメモリ402、磁気ディスク405などの記憶装置に記憶されたプログラムをCPU401に実行させることにより、または、I/F403により、その機能を実現する。また、各機能部は、例えば、ハードウェアにより実現されてもよい。具体的には、例えば、各機能部は、論理積回路であるAND、否定論理回路であるINVERTER、論理和回路であるOR、論理和否定回路であるNORや、ラッチ回路であるFF(Flip Flop)などの素子によって形成されてもよい。各機能部の処理結果は、例えば、メモリ402、磁気ディスク405などの記憶装置に記憶される。

40

50

【0062】

取得部501は、探索文字列を表すビット列を取得する機能を有する。具体的には、例えば、取得部501は、クライアント装置302からのデータの検索要求から、探索文字列を表すビット列を取得する。また、取得部501は、例えば、不図示のキーボードやマウスを用いたユーザの操作入力により、探索文字列を表すビット列を取得することにしてもよい。

【0063】

また、取得部501は、対象文字列を表すビット列を取得する機能を有する。具体的には、例えば、取得部501は、データベース320から対象文字列を表すビット列を取得する。より具体的には、例えば、取得部501は、データベース320からメモリ402上のデータベースバッファ（不図示）に読み込まれたデータから、対象文字列を表すビット列を取得する。

10

【0064】

作成部502は、探索文字列を表すビット列に含まれる1バイト分（第1ビット長11に相当）のビット列b1をレジスタRに8バイト分（第2ビット長12に相当）格納する機能を有する。具体的には、例えば、作成部502は、探索文字列を表すビット列の先頭（または、末尾）から1バイト分のビット列b1をレジスタR1に8バイト分格納する。

【0065】

また、作成部502は、対象文字列を表すビット列に含まれる8バイト分のビット列b2、b3をそれぞれ異なるレジスタRに格納する機能を有する。ビット列b2、b3は、対象文字列を表すビット列に含まれる連続するビット列である。具体的には、例えば、作成部502は、対象文字列を表すビット列の先頭から8バイト分のビット列b2をレジスタR2に格納する。また、作成部502は、例えば、対象文字列を表すビット列のうちビット列b2の次の8バイト分のビット列b3をレジスタR3に格納する。

20

【0066】

また、作成部502は、探索文字列を表すビット列に含まれる8バイト分のビット列b4をレジスタRに格納する機能を有する。ビット列b4の先頭1バイトは、ビット列b1に対応する。具体的には、例えば、作成部502は、探索文字列を表すビット列の先頭から8バイト分のビット列b4をレジスタR4に格納する。

【0067】

第1の比較部503は、探索文字列を表すビット列に含まれる1バイト分のビット列b1を、対象文字列を表すビット列に含まれる8バイト分のビット列b2とバイト単位で一括比較する機能を有する。具体的には、例えば、第1の比較部503は、特定の比較命令（例えば、SIMD Compare）を実行することにより、レジスタR1のビット列（8バイト分のビット列b1）とレジスタR2のビット列b2とをバイト単位で一括比較する。

30

【0068】

第1の比較部503の比較結果は、例えば、レジスタR5に記憶される。具体的には、例えば、第1の比較部503の比較結果は、レジスタR1、R2間のバイト単位の比較結果を表す1バイトのビット列b5である。ここでは、バイト単位の比較結果が不一致だった場合は「0」が設定され、バイト単位の比較結果が一致の場合は「1」が設定される。

40

【0069】

第2の比較部504は、探索文字列を表すビット列のうちビット列b1の次の1バイト分のビット列b6と、対象文字列を表すビット列のうちビット列b1と一致する箇所の次の1バイト分のビット列b7とを比較する機能を有する。具体的には、例えば、第2の比較部504は、レジスタR4に格納されたビット列b4のうちのビット列b1の次の1バイト分のビット列b6と、レジスタR2に格納されたビット列b2のうちのビット列b1と一致する箇所の次の1バイト分のビット列b7とを比較する。これにより、SJISなどの1文字を2バイトで表すマルチバイト文字であっても、第2の比較部504の比較結果から、対象文字列に探索文字列の文字（例えば、先頭文字）と一致する文字があるか否

50

かを判断することができる。

【0070】

なお、対象文字列を表すビット列のうちビット列b 1と一致する箇所は、例えば、第1の比較部503の比較結果を表すビット列b 5から特定することができる。例えば、ビット列b 5が「00001000」の場合、第2の比較部504は、対象文字列を表すビット列に含まれるビット列b 2の5バイト目にビット列b 1と一致する箇所があることを特定することができる。

【0071】

また、対象文字列を表すビット列に含まれるビット列b 2のうちビット列b 1と一致する箇所がない場合がある。この場合、作成部502は、例えば、対象文字列を表すビット列のうちのビット列b 2の次の8バイト分のビット列を、新たなビット列b 2に設定してレジスタR 2に格納する。また、作成部502は、対象文字列を表すビット列のうちの新たなビット列b 2の次の8バイト分のビット列を新たなビット列b 3に設定してレジスタR 3に格納する。

10

【0072】

なお、作成部502は、対象文字列を表すビット列のうちのビット列b 3の次の8バイト分のビット列をレジスタR 2に格納することにしてもよい。この場合、ビット列b 1の比較対象は、レジスタR 3のビット列となる。すなわち、ビット列b 1の比較対象を格納するレジスタRを、レジスタR 2、R 3間で切り替えることにしてもよい。

【0073】

20

生成部505は、対象文字列を表すビット列のうちのビット列b 1と一致する箇所から8バイト分のビット列b 8を生成するビット操作命令を実行する機能を有する。具体的には、例えば、生成部505は、ビット列b 6とビット列b 7とが一致する場合に、Shift-Mask-Orを実行することにより、対象文字列を表すビット列のうちのビット列b 1と一致する箇所から8バイト分のビット列b 8を生成する。

【0074】

より詳細に説明すると、まず、生成部505は、第1の比較部503の比較結果に基づいて、レジスタR 2に格納されたビット列b 2を左シフトする。具体的には、例えば、生成部505は、第1の比較命令の比較結果を表すビット列b 5の先頭から「1」が出現するまでに「0」が出現した数分、レジスタR 2に格納されたビット列b 2を左シフトする。そして、生成部505は、レジスタR 2のシフト元のビット（空いたバイト）を「0」でマスクする。

30

【0075】

次に、生成部505は、第1の比較部503の比較結果に基づいて、レジスタR 3に格納されたビット列b 3を右シフトする。具体的には、例えば、生成部505は、レジスタサイズ（8バイト）から、ビット列b 5の先頭から「1」が出現するまでに「0」が出現した数分引いた値で、レジスタR 3に格納されたビット列b 3を右シフトする。そして、生成部505は、レジスタR 3のシフト元のビット（空いたバイト）を「0」でマスクする。

【0076】

40

次に、生成部505は、レジスタR 2のビット列とレジスタR 3のビット列とをOr命令で結合したビット列b 8をレジスタR 6に格納する。これにより、対象文字列を表すビット列のうちの、ビット列b 1と一致する箇所から始まる8バイトのビット列b 8を生成することができる。

【0077】

また、生成部505は、ビット列b 6とビット列b 7とが不一致の場合、ビット操作命令を実行しないことにしてもよい。これにより、例えば、シングルバイト文字においては、偶然先頭文字のみが一致したものに対して、Shift-Mask-OrとSIMD Compareを行う無駄な処理を削減することができる。また、例えば、SJISなどのマルチバイト文字においては、先頭文字さえも一致しない候補に対して、個別にShi

50

f t - M a s k - O r と S I M D C o m p a r e を行う無駄な処理を削減することができる。

【0078】

第3の比較部506は、生成部505によって生成されたビット列b8と、探索文字列を表すビット列のうちビット列b1から8バイト分のビット列b4とをバイト単位で一括比較する機能を有する。具体的には、例えば、第3の比較部506は、特定の比較命令（例えば、S I M D C o m p a r e）を実行することにより、レジスタR6のビット列b8とレジスタR4のビット列b4とをバイト単位で一括比較する。

【0079】

第3の比較部506の比較結果は、例えば、レジスタR7に記憶される。具体的には、例えば、第3の比較部506の比較結果は、レジスタR4、R6間のバイト単位の比較結果を表す1バイトのビット列b9である。ここでは、バイト単位の比較結果が不一致だった場合は「0」が設定され、バイト単位の比較結果が一致の場合は「1」が設定される。

【0080】

また、第2の比較部504は、探索文字列および対象文字列がマルチバイト文字の場合、対象文字列を表すビット列に含まれるビット列b2のうちビット列b1と一致する箇所がマルチバイト文字の先頭バイトか否かを判断することにしてもよい。具体的には、例えば、第2の比較部504は、ビット列b1と一致する箇所が、対象文字列のある文字を表すビット列のうちの先頭1バイトのビット列か否かを判断する。

【0081】

ビット列b1と一致する箇所がマルチバイト文字の先頭バイトでない場合は、探索文字列の文字（例えば、先頭文字）と一致しているとはいえない。このため、第2の比較部504は、ビット列b1と一致する箇所がマルチバイト文字の先頭バイトである場合に、ビット列b6とビット列b7とを比較することにしてもよい。

【0082】

また、ビット列b1と一致する箇所がマルチバイト文字の先頭バイトでない場合、作成部502は、例えば、対象文字列を表すビット列のうちのビット列b2の次の8バイト分のビット列を、新たなビット列b2に設定してレジスタR2に格納することにしてもよい。また、作成部502は、対象文字列を表すビット列のうちの新たなビット列b2の次の8バイト分のビット列を新たなビット列b3に設定してレジスタR3に格納することにしてもよい。なお、探索文字列および対象文字列がマルチバイト文字の場合のデータベースサーバ301の文字列探索処理例については、図6～図8を用いて後述する。

【0083】

また、第2の比較部504は、探索文字列を表すビット列のうちビット列b6の次の1バイト分のビット列b10と、対象文字列を表すビット列のうちビット列b7の次の1バイト分のビット列b11とを比較することにしてもよい。具体的には、例えば、第2の比較部504は、探索文字列および対象文字列が1文字を3バイトで表すマルチバイト文字のときに、ビット列b6とビット列b7とが一致する場合、ビット列b10とビット列b11とを比較することにしてもよい。

【0084】

一例として、対象文字列「デジタルテレビの普及」および探索文字列「テレビ」を、1文字を3バイトで表すU T F - 8で表すと以下ようになる。

【0085】

[対象文字列] デ (E 3 8 3 8 7) ジ (E 3 8 2 B 8) タ (E 3 8 2 B F)
ル (E 3 8 3 A B) テ (E 3 8 3 8 6) レ (E 3 8 3 A C) ビ (E 3 8 3
9 3) の (E 3 8 1 A E) 普 (E 6 9 9 A E) 及 (E 5 8 F 8 A)

【0086】

[探索文字列] テ (E 3 8 3 8 6) レ (E 3 8 3 A C) ビ (E 3 8 3 9 3)

【0087】

この例では、対象文字列を表すビット列のうち、探索文字列の先頭文字「テ」の2バイ

10

20

30

40

50

ト目まで一致する部分が多くなっている。このような場合に、対象文字列を8バイトの文字列として加工し、SIMD命令により8バイトでの比較を実施すると、文字列探索にかかる処理時間の増大を招く恐れがある。

【0088】

このため、第2の比較部504は、文字コードがUTF-8などの場合、さらに、ビット列b6の次の1バイト分のビット列b10と、ビット列b7の次の1バイト分のビット列b11とを比較する。これにより、UTF-8などの1文字を3バイトで表すマルチバイト文字であっても、対象文字列に探索文字列の文字（例えば、先頭文字）と一致する文字があるか否かを判断することができる。

【0089】

また、生成部505は、ビット列b10とビット列b11とが一致する場合に、対象文字列を表すビット列のうちビット列b1と一致する箇所から8バイト分のビット列b8を生成するビット操作命令を実行することにしてもよい。すなわち、生成部505は、文字コードがUTF-8などのときは、ビット列b10とビット列b11とが一致する場合にビット操作命令を実行することにしてもよい。

【0090】

これにより、UTF-8などのマルチバイト文字においては、先頭文字さえも一致しない候補に対して、個別にShift-Mask-OrとSIMD Compareを行う無駄な処理を削減することができる。また、例えば、シングルバイト文字においても、ビット列b10とビット列b11とが一致する場合にビット操作命令を実行することで、偶然先頭2文字のみが一致したものに対して、Shift-Mask-OrとSIMD Compareを行う無駄な処理を削減することができる。

【0091】

判断部507は、第3の比較部506の比較結果に基づいて、対象文字列に探索文字列が含まれるか否かを判断する機能を有する。具体的には、例えば、判断部507は、レジスタR7に格納されたビット列b9のうち連続する1の数が探索文字列の長さL以上の場合に、対象文字列に探索文字列が含まれると判断する。

【0092】

ここで、レジスタR7に格納されたビット列b9のうち連続する1の数は、レジスタR6のビット列b8とレジスタR4のビット列b4とを比較して連続して一致した数（バイト数）である。以下の説明では、レジスタR7に格納されたビット列b9のうち連続する1の数を「比較して一致した数CN」と表記する場合がある。

【0093】

また、探索文字列の長さLは、探索文字列の文字コードに応じて決まる。例えば、1文字を1バイトで表すシングルバイト文字の場合、探索文字列の文字数をm文字とすると、探索文字列の長さLは、「 $L=m$ 」となる。また、1文字を2バイトで表すマルチバイト文字の場合、探索文字列の文字数をm文字とすると、探索文字列の長さLは、「 $L=2m$ 」となる。また、1文字を3バイトで表すマルチバイト文字の場合、探索文字列の文字数をm文字とすると、探索文字列の長さLは、「 $L=3m$ 」となる。

【0094】

また、判断部507は、比較して一致した数CNが探索文字列の長さL未満の場合、探索文字列の長さLが8nバイトより大きいかなんかを判断することにしてもよい。なお、「8n」の「8」は、CPU401の処理単位である8バイト（第2ビット長12に相当）を表す。また、「8n」の「n」は、第3の比較部506の比較処理（例えば、SIMD Compare）を繰り返したループ数である（ $n=1, 2, \dots$ ）。

【0095】

ここで、探索文字列の長さLが8nバイト以下の場合、判断部507は、対象文字列を表すビット列のうち探索文字列を表すビット列と比較していない残余のビット列があるかなんかを判断する。そして、探索文字列を表すビット列と比較していない残余のビット列がない場合、判断部507は、対象文字列に探索文字列が含まれないと判断する。

【0096】

一方、探索文字列を表すビット列と比較していない残余のビット列がある場合、作成部502は、例えば、対象文字列を表すビット列のうちのビット列b2の次の8バイト分のビット列を、新たなビット列b2に設定してレジスタR2に格納する。また、作成部502は、例えば、対象文字列を表すビット列のうちの新たなビット列b2の次の8バイト分のビット列を新たなビット列b3に設定してレジスタR3に格納する。これにより、対象文字列を表すビット列のうち探索文字列を表すビット列と比較していない残余のビット列を、探索文字列を表すビット列と比較することができる。

【0097】

また、探索文字列の長さLが8nバイトより大きい場合、第3の比較部506は、対象文字列を表すビット列のうちビット列b8の次の8バイト分のビット列と、探索文字列を表すビット列のうちビット列b4の次の8バイト分のビット列とを比較する。すなわち、探索文字列の長さLが8nバイトより大きい場合は、1回の比較処理により探索文字列のすべてを対象文字列と比較できない。このため、第3の比較部506は、探索文字列を表すビット列のうち対象文字列を表すビット列と比較していない残余のビット列を、対象文字列を表すビット列と比較する。

【0098】

具体的には、例えば、第3の比較部506は、対象文字列を表すビット列のうちのビット列b8の次の8バイト分のビット列を、新たなビット列b8に設定してレジスタR6に格納する。また、第3の比較部506は、探索文字列を表すビット列のうちのビット列b4の次の8バイト分のビット列を新たなビット列b4に設定してレジスタR4に格納する。

【0099】

そして、第3の比較部506は、特定の比較命令（例えば、SIMD Compare）を実行することにより、レジスタR6のビット列b8とレジスタR4のビット列b4とをバイト単位で一括比較する。これにより、探索文字列を表すビット列のうち対象文字列を表すビット列と比較していない残余のビット列を、対象文字列を表すビット列と比較することができる。

【0100】

出力部508は、判断部507によって判断された判断結果を出力する機能を有する。ここで、判断結果は、対象文字列に探索文字列が含まれるか否かを示す情報である。例えば、対象文字列に探索文字列が含まれる場合、データベースサーバ301は、クライアント装置302からの検索要求に対する検索結果として、対象文字列を含むデータをクライアント装置302に送信する。また、対象文字列に探索文字列が含まれない場合は、例えば、データベースサーバ301は、検索要求に対するデータが検索されなかったことを示す検索結果をクライアント装置302に送信する。

【0101】

（マルチバイト文字の場合の文字列探索処理例）

次に、図6～図8を用いて、探索文字列および対象文字列がマルチバイト文字の場合のデータベースサーバ301の文字列探索処理例について説明する。ここでは、マルチバイト文字の一例として、文字コードがSJISの場合を例に挙げて説明する。

【0102】

図6～図8は、文字コードがSJISの場合の文字列探索処理例を示す説明図である。図6において、対象文字列は「デジタルテレビの普及」であり、22バイトのデータである。また、探索文字列は「テレビ」であり、6バイトのデータである。

【0103】

（1）作成部502は、探索文字列を表すビット列の先頭から1バイト分のビット列b1をレジスタR1に8バイト分格納する。図6の例では、探索文字列の先頭文字「テ」を表す2バイトのビット列のうちの先頭バイトのビット列b1「10000011」がレジスタR1に8個格納されている。

10

20

30

40

50

【0104】

また、作成部502は、対象文字列を表すビット列のうち、先頭から8バイト分のビット列b2をレジスタR2に格納するとともに、ビット列b2の次の8バイト分のビット列b3をレジスタR3に格納する。図6の例では、対象文字列のうち、先頭から4文字の文字列「デジタル」を表すビット列b2がレジスタR2に格納され、5文字目から4文字の文字列「ーテレビ」を表すビット列b3がレジスタR3に格納されている。

【0105】

また、作成部502は、探索文字列を表すビット列の先頭から8バイト分のビット列b4をレジスタR4に格納する。図6の例では、探索文字列の先頭から3文字の文字列「テレビ」を表すビット列b4がレジスタR4に格納されている。

10

【0106】

(2) 第1の比較部503は、SIMD Compareを実行することにより、レジスタR1のビット列(8バイト分のビット列b1)とレジスタR2のビット列(ビット列b2)とをバイト単位で一括比較して、比較結果をレジスタR5に格納する。図7の例では、第1の比較部503の比較結果を表すビット列b5「10101010」がレジスタR5に格納されている。このビット列b5によれば、1, 3, 5, 7バイト目に“83(ビット列b1)”があることを特定することができる。

【0107】

(3) 第2の比較部504は、対象文字列を表すビット列に含まれるビット列b2のうちビット列b1と一致する箇所がマルチバイト文字の先頭バイトか否かを判断する。具体的には、例えば、第2の比較部504は、ビット列b2のうちビット列b1と一致する箇所(1, 3, 5, 7バイト目)が奇数バイト目か否かを判断する。図7の例では、ビット列b1と一致するすべての箇所が先頭バイトとなっている。

20

【0108】

次に、第2の比較部504は、探索文字列を表すビット列のうちビット列b1の次の1バイト分のビット列b6と、対象文字列を表すビット列のうちビット列b1と一致する箇所の次の1バイト分のビット列b7とを比較する。図6の例では、ビット列b6は「01100101」である。また、ビット列b7は、ビット列b2の2, 4, 6, 8バイト目のいずれかのビット列であり、いずれのビット列もビット列b6と一致しない。

【0109】

この場合、作成部502は、対象文字列を表すビット列のうちのビット列b3の次の8バイト分のビット列(ここでは、「ビット列b2'」と表記する)をレジスタR2に格納する。

30

【0110】

(4) 第1の比較部503は、SIMD Compareを実行することにより、レジスタR1のビット列(8バイト分のビット列b1)とレジスタR3のビット列b3とをバイト単位で一括比較して、比較結果をレジスタR5に格納する。図7の例では、第1の比較部503の比較結果を表すビット列b5「00101010」がレジスタR5に格納されている。このビット列b5によれば、3, 5, 7バイト目に“83(ビット列b1)”があることを特定することができる。

40

【0111】

(5) 第2の比較部504は、レジスタR3のビット列b3のうちビット列b1と一致する箇所がマルチバイト文字の先頭バイトか否かを判断する。図6の例では、レジスタR3のビット列b3のうちビット列b1と一致するすべての箇所が先頭バイトとなっている。

【0112】

次に、第2の比較部504は、探索文字列を表すビット列のうちビット列b1の次の1バイト分のビット列b6と、レジスタR3のビット列b3のうちビット列b1と一致する箇所の次の1バイト分のビット列b7とを比較する。図6の例では、ビット列b7は、レジスタR3のビット列b3の4, 6, 8バイト目のいずれかのビット列であり、4バイト

50

目のビット列がビット列b 6と一致する。

【0113】

(6) 生成部505は、レジスタR2, R3に対して、Shift-Mask-Orを実行することにより、対象文字列を表すビット列のうちビット列b 1と一致する箇所から8バイト分のビット列b 8を生成して、レジスタR6に格納する。図8の例では、対象文字列を表すビット列のうち、ビット列b 1と一致する箇所から始まる8バイトのビット列b 8が生成されている。

【0114】

(7) 第3の比較部506は、SIMD Compareを実行することにより、レジスタR6のビット列(ビット列b 8)と、レジスタR4のビット列(ビット列b 4)とを1バイト単位で一括比較して、比較結果をレジスタR7に格納する。図8の例では、第3の比較部506の比較結果を表すビット列b 9「11111100」がレジスタR7に格納されている。

【0115】

(8) 判断部507は、レジスタR7のビット列b 9のうち連続する1の数が探索文字列の長さL以上の場合に、対象文字列に探索文字列が含まれると判断する。ここでは、探索文字列の長さLは、「6」である。また、ビット列b 9のうち連続する1の数は、「6」である。このため、判断部507は、対象文字列に探索文字列が含まれると判断する。

【0116】

このように、データベースサーバ301によれば、SJISなどのマルチバイト文字において、先頭文字が一致する候補に対して、Shift-Mask-OrとSIMD Compareを行って、詳細探索を行うことができる。

【0117】

(データベースサーバ301の文字列探索処理手順)

次に、データベースサーバ301の文字列探索処理手順について説明する。ここでは、探索文字列および対象文字列がマルチバイト文字の場合を例に挙げて説明する。

【0118】

図9および図10は、データベースサーバ301の文字列探索処理手順の一例を示すフローチャートである。図9のフローチャートにおいて、まず、データベースサーバ301は、探索文字列を表すビット列の先頭1バイトのビット列をレジスタR1に8バイト分格納する(ステップS901)。

【0119】

次に、データベースサーバ301は、対象文字列を表すビット列のうちの未処理のビット列の先頭から8バイト分のビット列をレジスタR2に格納する(ステップS902)。なお、未処理のビット列とは、対象文字列を表すビット列のうち、レジスタR1のビット列と比較していない残余のビット列である。

【0120】

次に、データベースサーバ301は、対象文字列を表すビット列のうちのレジスタR2のビット列の次の8バイト分のビット列をレジスタR3に格納する(ステップS903)。次に、データベースサーバ301は、探索文字列を表すビット列の先頭から8バイト分のビット列をレジスタR4に格納する(ステップS904)。

【0121】

そして、データベースサーバ301は、SIMD Compareを実行することにより、レジスタR1のビット列とレジスタR2のビット列とをバイト単位で一括比較して、比較結果をレジスタR5に格納する(ステップS905)。次に、データベースサーバ301は、比較結果に基づいて、レジスタR2のビット列のうち、探索文字列の先頭1バイトのビット列と一致する箇所があるか否かを判断する(ステップS906)。

【0122】

ここで、探索文字列の先頭1バイトのビット列と一致する箇所がある場合(ステップS906: Yes)、データベースサーバ301は、レジスタR2のビット列のうちの探索

10

20

30

40

50

文字列の先頭 1 バイトのビット列と一致する箇所がマルチバイト文字の先頭バイトか否かを判断する（ステップ S 9 0 7）。

【0 1 2 3】

ここで、マルチバイト文字の先頭バイトの場合（ステップ S 9 0 7：Y e s）、データベースサーバ 3 0 1 は、探索文字列を表すビット列のうち先頭 1 バイトのビット列の次の 1 バイト分のビット列と、対象文字列を表すビット列のうちの探索文字列の先頭 1 バイトのビット列と一致する箇所の次の 1 バイト分のビット列とが一致するか否かを判断する（ステップ S 9 0 8）。

【0 1 2 4】

ここで、ビット列が一致する場合（ステップ S 9 0 8：Y e s）、データベースサーバ 3 0 1 は、レジスタ R 2，R 3 に対して、S h i f t - M a s k - O r を実行することにより、対象文字列を表すビット列のうちの探索文字列の先頭 1 バイトのビット列と一致する箇所から 8 バイト分のビット列を生成して、レジスタ R 6 に格納する（ステップ S 9 0 9）。

【0 1 2 5】

そして、データベースサーバ 3 0 1 は、S I M D C o m p a r e を実行することにより、レジスタ R 6 のビット列とレジスタ R 4 のビット列とをバイト単位で一括比較して、比較結果をレジスタ R 7 に格納し（ステップ S 9 1 0）、図 1 0 に示すステップ S 1 0 0 1 に移行する。

【0 1 2 6】

また、ステップ S 9 0 7 において、マルチバイト文字の先頭バイトではない場合（ステップ S 9 0 7：N o）、データベースサーバ 3 0 1 は、レジスタ R 5 に格納された比較結果を表すビット列のうちの最初の「1」を「0」にして（ステップ S 9 1 1）、ステップ S 9 0 6 に戻る。また、ステップ S 9 0 8 において、ビット列が不一致の場合（ステップ S 9 0 8：N o）、データベースサーバ 3 0 1 は、ステップ S 9 1 1 に移行する。

【0 1 2 7】

また、ステップ S 9 0 6 において、探索文字列の先頭 1 バイトのビット列と一致する箇所がない場合（ステップ S 9 0 6：N o）、データベースサーバ 3 0 1 は、対象文字列を表すビット列のうち未処理のビット列があるか否かを判断する（ステップ S 9 1 2）。ここで、未処理のビット列がある場合（ステップ S 9 1 2：Y e s）、データベースサーバ 3 0 1 は、ステップ S 9 0 2 に戻る。

【0 1 2 8】

一方、未処理のビット列がない場合（ステップ S 9 1 2：N o）、データベースサーバ 3 0 1 は、対象文字列に探索文字列が含まれないと判断して（ステップ S 9 1 3）、本フローチャートによる一連の処理を終了する。

【0 1 2 9】

図 1 0 のフローチャートにおいて、まず、データベースサーバ 3 0 1 は、レジスタ R 7 に格納された比較結果を表すビット列に基づいて、比較して一致した数 C N が探索文字列の長さ L 以上か否かを判断する（ステップ S 1 0 0 1）。ここで、比較して一致した数 C N が探索文字列の長さ L 以上の場合（ステップ S 1 0 0 1：Y e s）、データベースサーバ 3 0 1 は、対象文字列に探索文字列が含まれると判断して（ステップ S 1 0 0 2）、本フローチャートによる一連の処理を終了する。

【0 1 3 0】

一方、比較して一致した数 C N が探索文字列の長さ L 未満の場合（ステップ S 1 0 0 1：N o）、データベースサーバ 3 0 1 は、探索文字列の長さ L が 8 n バイトより大きいかな否かを判断する（ステップ S 1 0 0 3）。なお、「n」の初期値は「n = 1」である。

【0 1 3 1】

ここで、探索文字列の長さ L が 8 n バイトより大きい場合（ステップ S 1 0 0 3：Y e s）、データベースサーバ 3 0 1 は、対象文字列を表すビット列のうちのレジスタ R 6 のビット列の次の 8 バイト分のビット列をレジスタ R 6 に格納する（ステップ S 1 0 0 4）

10

20

30

40

50

。次に、データベースサーバ301は、探索文字列を表すビット列のうちのレジスタR4のビット列の次の8バイト分のビット列をレジスタR4に格納する（ステップS1005）。

【0132】

そして、データベースサーバ301は、SIMD Compareを実行することにより、レジスタR6のビット列とレジスタR4のビット列とをバイト単位で一括比較して、比較結果をレジスタR7に格納する（ステップS1006）。ただし、比較結果は、例えば、レジスタR7の未使用部分の先頭から順次格納される。

【0133】

次に、データベースサーバ301は、「n」をインクリメントして（ステップS1007）、ステップS1001に戻る。また、ステップS1003において、探索文字列の長さLが8nバイト以下の場合（ステップS1003：No）、データベースサーバ301は、「n」を「n=1」で初期化する（ステップS1008）。

【0134】

そして、データベースサーバ301は、レジスタR5に格納された比較結果を表すビット列のうちの最初の「1」を「0」にして（ステップS1009）、図9に示したステップS906に戻る。

【0135】

これにより、SIMD Compareを用いて一次スクリーニングを行った後で、Shift-Mask-Orを用いた詳細探索を行うことができ、対象文字列に探索文字列が含まれるか否かを効率的に判断することができる。なお、1文字を3バイトで表すマルチバイト文字の場合は、図9に示したステップS908において、さらに、次の1バイト分のビット列の一致判定を行うことにしてもよい。

【0136】

以上説明したように、実施の形態にかかるデータベースサーバ301によれば、特定の比較命令（例えば、SIMD Compare）により、探索文字列の先頭から1バイト分のビット列b1を、対象文字列を表すビット列に含まれる8バイト分（例えば、先頭から8バイト分）のビット列b2とバイト単位で一括比較することができる。これにより、ビット列同士を高速に比較して、対象文字列を表すビット列から探索文字列の先頭バイトの出現位置を検出することができる。例えば、ASCIIなどの1文字を1バイトで表すシングルバイト文字の場合、探索文字列の先頭文字が対象文字列にヒットしたか否かを判断することができる。

【0137】

また、データベースサーバ301によれば、探索文字列のビット列b1の次の1バイト分のビット列b6と、対象文字列のビット列b1と一致する箇所の次の1バイト分のビット列b7とを比較することができる。これにより、SJISなどの1文字を2バイトで表すマルチバイト文字であっても、探索文字列の先頭文字が対象文字列にヒットしたか否かを判断することができる。また、シングルバイト文字であれば、探索文字列の先頭文字の次の文字も対象文字列にヒットしているか否かを判断することができる。

【0138】

また、データベースサーバ301によれば、ビット列b6とビット列b7とが一致する場合に、特定のビット操作命令（例えば、Shift-Mask-Or）により、対象文字列を表すビット列のうちビット列b1と一致する箇所から8バイト分のビット列b8を生成することができる。また、データベースサーバ301によれば、特定の比較命令（例えば、SIMD Compare）により、生成したビット列b8と、探索文字列の先頭から8バイト分のビット列b4とをバイト単位で一括比較することができる。

【0139】

これにより、ビット列同士を高速に比較して、対象文字列を表すビット列から、探索文字列の先頭から8バイト分のビット列を検出することができる。また、ビット列b6とビット列b7とが不一致の場合は特定のビット操作命令が実行されないため、文字列探索に

10

20

30

40

50

かかる処理時間の増大を抑制することができる。例えば、シングルスバイト文字の場合は、偶然先頭文字のみが一致したものに対して、Shift-Mask-OrとSIMD Compareを行う無駄な処理を削減することができる。また、例えば、マルチバイト文字の場合は、先頭文字さえも一致しない候補に対して、個別にShift-Mask-OrとSIMD Compareを行う無駄な処理を削減することができる。

【0140】

また、データベースサーバ301によれば、ビット列b8とビット列b4とをバイト単位で比較した比較結果に基づいて、対象文字列に探索文字列が含まれるか否かを判断することができる。例えば、データベースサーバ301は、ビット列b8とビット列b4とを比較した比較結果を表すビット列b9のうち連続する1の数が探索文字列の長さL以上の場合に、対象文字列に探索文字列が含まれると判断することができる。

10

【0141】

これにより、探索文字列を含む対象文字列を検索することができる。この結果、例えば、データベースサーバ301は、クライアント装置302からのデータの検索要求に対する検索結果として、対象文字列を含むデータをクライアント装置302に送信することができる。

【0142】

また、データベースサーバ301によれば、マルチバイト文字の場合は、対象文字列を表すビット列に含まれるビット列b2のうちのビット列b1と一致する箇所がマルチバイト文字の先頭バイトか否かを判断することができる。そして、データベースサーバ301によれば、ビット列b2のうちのビット列b1と一致する箇所がマルチバイト文字の先頭バイトである場合に、ビット列b6とビット列b7とを比較することができる。これにより、マルチバイト文字においては、先頭バイトさえも一致しない候補に対して、個別にShift-Mask-OrとSIMD Compareを行う無駄な処理を削減することができる。

20

【0143】

また、データベースサーバ301によれば、1文字を3バイトで表すマルチバイト文字の場合は、探索文字列のビット列b6の次の1バイト分のビット列b10と、対象文字列のビット列b7の次の1バイト分のビット列b11とを比較することができる。これにより、UTF-8などの1文字を3バイトで表すマルチバイト文字であっても、対象文字列に探索文字列の文字（例えば、先頭文字）と一致する文字があるか否かを判断することができる。

30

【0144】

また、データベースサーバ301によれば、ビット列b10とビット列b11とが一致する場合に、特定のビット操作命令により、対象文字列を表すビット列のうちビット列b1と一致する箇所から8バイト分のビット列b8を生成することができる。これにより、UTF-8などのマルチバイト文字において、先頭文字さえも一致しない候補に対して、個別にShift-Mask-OrとSIMD Compareを行う無駄な処理を削減することができる。

【0145】

これらのことから、データベースサーバ301によれば、SIMD Compare等の特定の比較命令を用いて一次スクリーニングを高速に行って、Shift-Mask-Or等の特定のビット操作命令を用いた詳細探索を効率的に行うことができる。これにより、文字列探索にかかる処理時間の短縮化を図ることができ、データ検索にかかる処理性能を向上させることができる。例えば、複数バイト単位N（例えば、8バイト）で文字列比較を行うことで、バイト単位で文字列比較を行う場合に比べて、検索コストを1/N程度に削減することができる。

40

【0146】

なお、本実施の形態で説明した文字列探索方法は、予め用意されたプログラムをパーソナル・コンピュータやワークステーション等のコンピュータで実行することにより実現す

50

ることができる。本文字列探索プログラムは、ハードディスク、フレキシブルディスク、CD-ROM、MO、DVD等のコンピュータで読み取り可能な記録媒体に記録され、コンピュータによって記録媒体から読み出されることによって実行される。また、本文字列探索プログラムは、インターネット等のネットワークを介して配布してもよい。

【0147】

上述した実施の形態に関し、さらに以下の付記を開示する。

【0148】

(付記1) コンピュータが、

探索文字列に対応する探索文字ビット列に含まれる第1ビット長の第1ビット列を、対象文字列に対応する対象文字ビット列に含まれる第2ビット長の第2ビット列と前記第1ビット長単位で比較する第1比較と、

10

前記第1比較の比較結果に基づいて、前記探索文字ビット列のうち前記第1ビット列の次の前記第1ビット長の第3ビット列と、前記第2ビット列のうち前記第1ビット列と一致する箇所の次の前記第1ビット長の第4ビット列との比較と、

前記第3ビット列と前記第4ビット列とが一致する場合に、前記対象文字ビット列のうち前記第1ビット列と一致する箇所から前記第2ビット長の第5ビット列の生成と、

生成した前記第5ビット列と、前記探索文字ビット列のうち前記第1ビット列から前記第2ビット長の第6ビット列とを前記第1ビット長単位で比較する第2比較と、

前記第2比較の比較結果に基づいて、前記対象文字列に前記探索文字列が含まれるか否かの判断と、

20

を実行することを特徴とする文字列探索方法。

【0149】

(付記2) 前記コンピュータが、

前記探索文字列および前記対象文字列が特定の文字コードのときは、前記第3ビット列と前記第4ビット列とが一致する場合、前記探索文字ビット列のうち前記第3ビット列の次の前記第1ビット長のビット列と、前記対象文字ビット列のうち前記第4ビット列の次の前記第1ビット長のビット列との比較を実行し、

前記第3ビット列の次のビット列と前記第4ビット列の次のビット列とが一致の場合に、前記第5ビット列の生成を実行することを特徴とする付記1に記載の文字列探索方法。

30

【0150】

(付記3) 前記第1ビット長は、1バイトであり、

前記第2ビット長は、ビット列同士をバイト単位で一括比較する比較命令の処理単位を表すビット長であることを特徴とする付記1または2に記載の文字列探索方法。

【0151】

(付記4) 前記コンピュータが、

前記探索文字列および前記対象文字列がマルチバイト文字の場合、前記第1比較の比較結果に基づいて、前記第2ビット列のうち前記第1ビット列と一致する箇所がマルチバイト文字の先頭バイトか否かの判断を実行し、

前記第1ビット列と一致する箇所がマルチバイト文字の先頭バイトである場合に、前記第3ビット列と前記第4ビット列との比較を実行することを特徴とする付記3に記載の文字列探索方法。

40

【0152】

(付記5) 前記コンピュータが、

前記第1ビット列を前記第2ビット長分第1レジスタに格納し、

前記第2ビット列を第2レジスタに格納する、処理を実行し、

前記第1レジスタに格納されたビット列と、前記第2レジスタに格納されたビット列とを前記第1ビット長単位で比較する第1比較を実行することを特徴とする付記1～4のいずれか一つに記載の文字列探索方法。

【0153】

50

(付記 6) 前記コンピュータが、

前記対象文字ビット列のうち前記第 2 ビット列の次の前記第 2 ビット長のビット列を第 3 レジスタに格納する処理を実行し、

前記第 1 比較の比較結果に基づいて、前記第 2 レジスタに格納されたビット列を左シフトするとともにシフト元のビットをマスクし、前記第 3 レジスタに格納されたビット列を右シフトするとともにシフト元のビットをマスクし、前記第 2 レジスタに格納されたビット列と前記第 3 レジスタに格納されたビット列との論理和をとることにより、前記第 5 ビット列の生成を実行することを特徴とする付記 5 に記載の文字列探索方法。

【0154】

(付記 7) 前記コンピュータが、

前記探索文字ビット列のうち前記第 1 ビット列から前記第 2 ビット長のビット列を第 4 レジスタに格納し、

生成した前記第 5 ビット列を第 5 レジスタに格納する、処理を実行し、

前記第 4 レジスタに格納されたビット列と、前記第 5 レジスタに格納されたビット列とを前記第 1 ビット長単位で比較する第 2 比較を実行することを特徴とする付記 6 に記載の文字列探索方法。

【0155】

(付記 8) 前記コンピュータが、

前記第 2 ビット列のうち前記第 1 ビット列と一致する箇所がない場合、前記対象文字ビット列のうちの前記第 2 ビット列の次の前記第 2 ビット長のビット列を、前記第 2 ビット列として前記第 2 レジスタに格納する処理を実行することを特徴とする付記 5～7 のいずれか一つに記載の文字列探索方法。

【0156】

(付記 9) 探索文字列に対応する探索文字ビット列に含まれる第 1 ビット長の第 1 ビット列を、対象文字列に対応する対象文字ビット列に含まれる第 2 ビット長の第 2 ビット列と前記第 1 ビット長単位で比較する第 1 の比較部と、

前記第 1 の比較部の比較結果に基づいて、前記探索文字ビット列のうち前記第 1 ビット列の次の前記第 1 ビット長の第 3 ビット列と、前記第 2 ビット列のうち前記第 1 ビット列と一致する箇所の次の前記第 1 ビット長の第 4 ビット列とを比較する第 2 の比較部と、

前記第 3 ビット列と前記第 4 ビット列とが一致する場合に、前記対象文字ビット列のうちの前記第 1 ビット列と一致する箇所から前記第 2 ビット長の第 5 ビット列を生成する生成部と、

前記生成部によって生成された前記第 5 ビット列と、前記探索文字ビット列のうち前記第 1 ビット列から前記第 2 ビット長の第 6 ビット列とを前記第 1 ビット長単位で比較する第 3 の比較部と、

前記第 3 の比較部の比較結果に基づいて、前記対象文字列に前記探索文字列が含まれるか否かを判断する判断部と、

を有することを特徴とする文字列探索装置。

【0157】

(付記 10) コンピュータに、

探索文字列に対応する探索文字ビット列に含まれる第 1 ビット長の第 1 ビット列を、対象文字列に対応する対象文字ビット列に含まれる第 2 ビット長の第 2 ビット列と前記第 1 ビット長単位で比較する第 1 比較と、

前記第 1 比較の比較結果に基づいて、前記探索文字ビット列のうち前記第 1 ビット列の次の前記第 1 ビット長の第 3 ビット列と、前記第 2 ビット列のうち前記第 1 ビット列と一致する箇所の次の前記第 1 ビット長の第 4 ビット列との比較と、

前記第 3 ビット列と前記第 4 ビット列とが一致する場合に、前記対象文字ビット列のうちの前記第 1 ビット列と一致する箇所から前記第 2 ビット長の第 5 ビット列の生成と、

生成した前記第 5 ビット列と、前記探索文字ビット列のうち前記第 1 ビット列から前記第 2 ビット長の第 6 ビット列とを前記第 1 ビット長単位で比較する第 2 比較と、

10

20

30

40

50

前記第2比較の比較結果に基づいて、前記対象文字列に前記探索文字列が含まれるか否かの判断と、

を実行させることを特徴とする文字列探索プログラム。

【0158】

(付記11) コンピュータに、

探索文字列に対応する探索文字ビット列に含まれる第1ビット長の第1ビット列を、対象文字列に対応する対象文字ビット列に含まれる第2ビット長の第2ビット列と前記第1ビット長単位で比較する第1比較と、

前記第1比較の比較結果に基づいて、前記探索文字ビット列のうち前記第1ビット列の次の前記第1ビット長の第3ビット列と、前記第2ビット列のうち前記第1ビット列と一致する箇所の次の前記第1ビット長の第4ビット列との比較と、

10

前記第3ビット列と前記第4ビット列とが一致する場合に、前記対象文字ビット列のうち前記第1ビット列と一致する箇所から前記第2ビット長の第5ビット列の生成と、

生成した前記第5ビット列と、前記探索文字ビット列のうち前記第1ビット列から前記第2ビット長の第6ビット列とを前記第1ビット長単位で比較する第2比較と、

前記第2比較の比較結果に基づいて、前記対象文字列に前記探索文字列が含まれるか否かの判断と、

を実行させる文字列探索プログラムを記録したことを特徴とする前記コンピュータに読み取り可能な記録媒体。

【符号の説明】

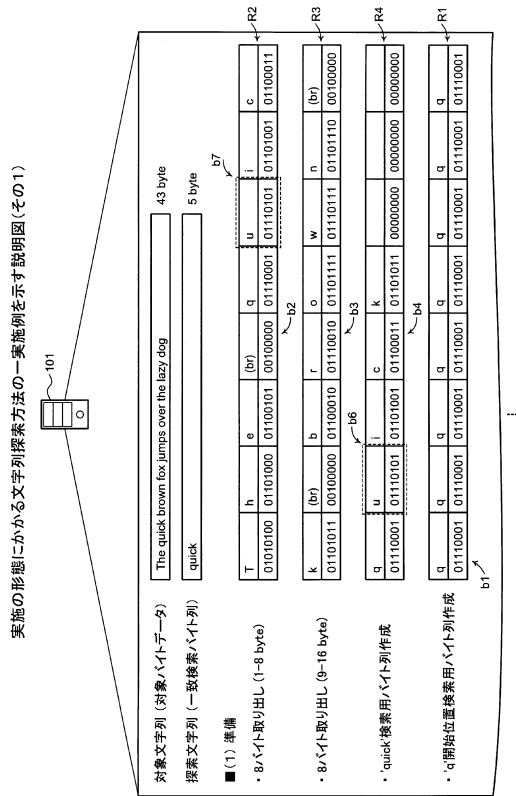
20

【0159】

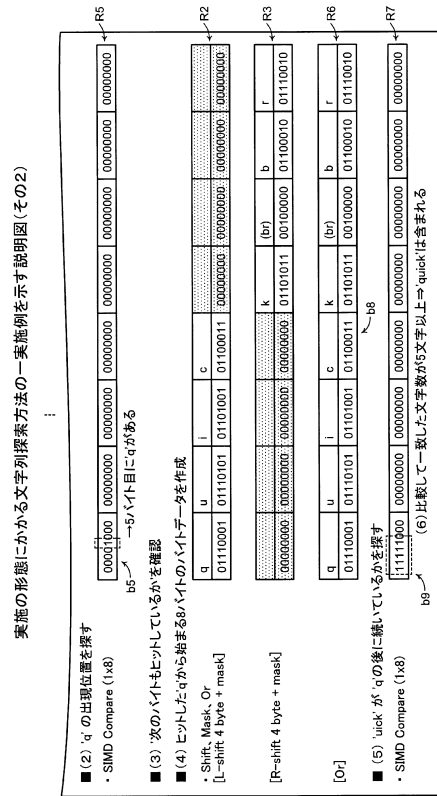
- 101 文字列探索装置
- 300 システム
- 301 データベースサーバ
- 302 クライアント装置
- 501 取得部
- 502 作成部
- 503 第1の比較部
- 504 第2の比較部
- 505 生成部
- 506 第3の比較部
- 507 判断部
- 508 出力部

30

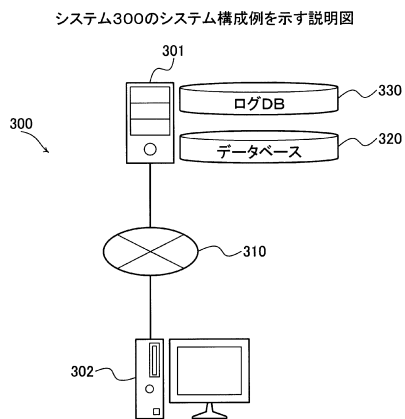
【図 1】



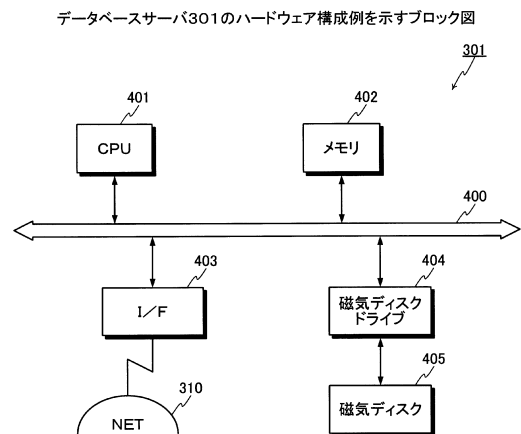
【図 2】



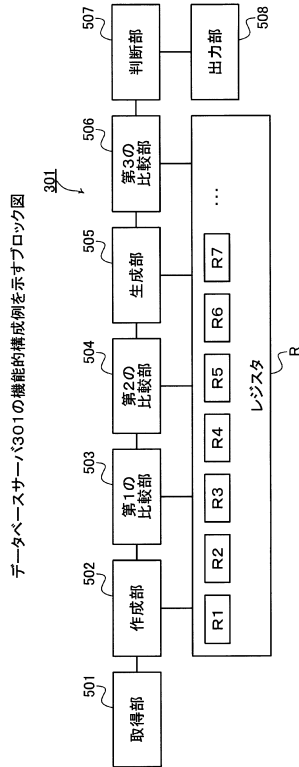
【図 3】



【图 4】

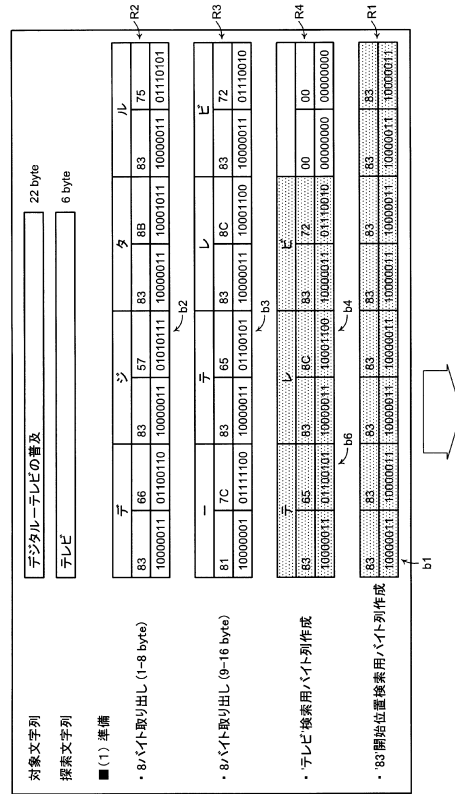


【図 5】



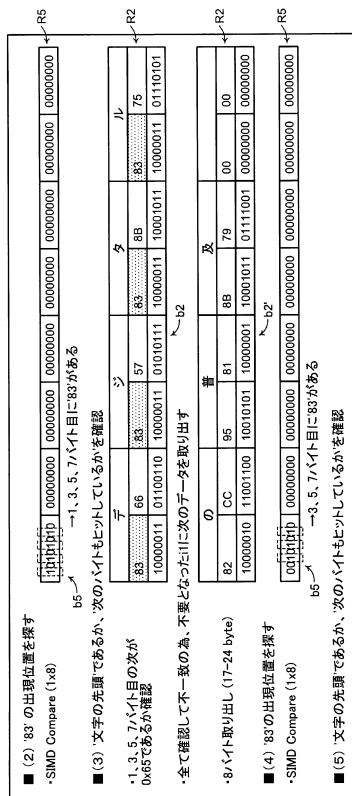
【図 6】

文字コードがSJISの場合の文字列探索処理例を示す説明図 (その1)



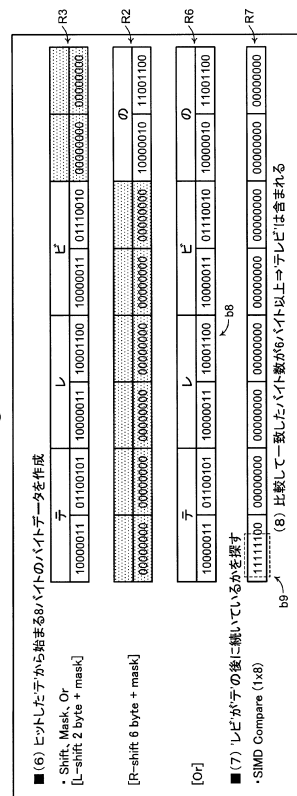
【図 7】

文字コードがSJISの場合の文字列探索処理例を示す説明図 (その2)

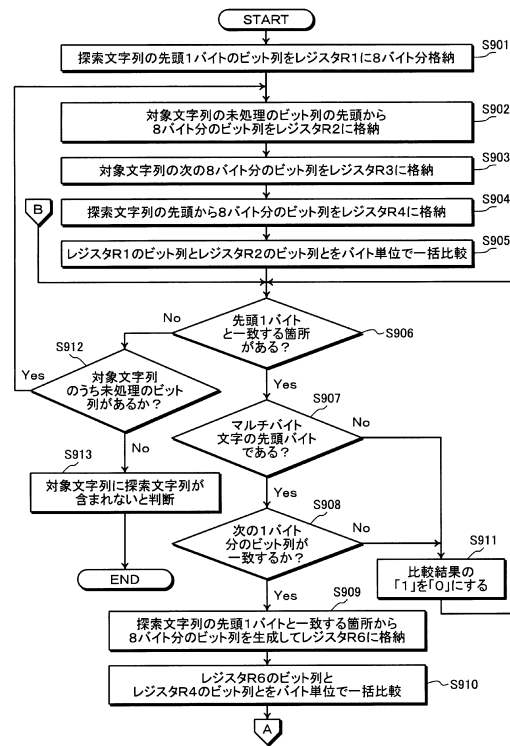


【図 8】

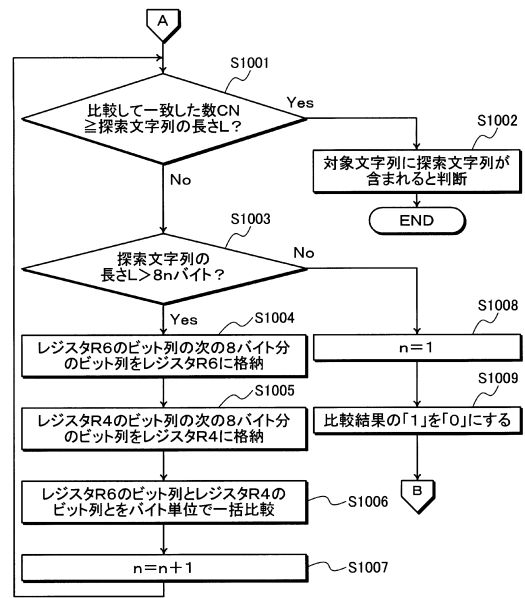
文字コードがSJISの場合の文字列探索処理例を示す説明図 (その3)



【図 9】

データベースサーバ301の文字列探索処理手順の一例を示すフローチャート
(その1)

【図 10】

データベースサーバ301の文字列探索処理手順の一例を示すフローチャート
(その2)

フロントページの続き

- (72)発明者 吉田 美樹雄
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
- (72)発明者 黒川 俊幸
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
- (72)発明者 吉永 玄德
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 小太刀 慶明

- (56)参考文献 特開2002-312364 (JP, A)
特開平01-224831 (JP, A)
特表2010-504594 (JP, A)

- (58)調査した分野(Int.Cl., DB名)
G06F 17/30