

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

**特許第3672076号
(P3672076)**

(45) 発行日 平成17年7月13日(2005.7.13)

(24) 登録日 平成17年4月28日(2005.4.28)

(51) Int.Cl.⁷**G06F 7/02**

F I

G O 6 F 7/02

J

請求項の数 13 (全 50 頁)

(21) 出願番号	特願平11-269566	(73) 特許権者	000005821
(22) 出願日	平成11年9月22日(1999.9.22)		松下電器産業株式会社
(65) 公開番号	特開2001-92634(P2001-92634A)		大阪府門真市大字門真1006番地
(43) 公開日	平成13年4月6日(2001.4.6)	(74) 代理人	100078282
審査請求日	平成13年4月11日(2001.4.11)		弁理士 山本 秀策
		(72) 発明者	藤本 正一
			大阪府門真市大字門真1006番地 松下
			電器産業株式会社内
		(72) 発明者	松本 道弘
			大阪府門真市大字門真1006番地 松下
			電器産業株式会社内
		審査官	田中 友章

最終頁に続く

(54) 【発明の名称】 演算器および演算方法およびプロセッサ

(57) 【特許請求の範囲】

【請求項1】

2の補数表現による第1の値と2の補数表現による第2の値とを受け取り、前記第1の値の絶対値と前記第2の値の絶対値とのどちらが大きいかを表す情報を出力する演算器であって、

前記第1の値の符号が負である場合には前記第1の値を1の補数表現に変換して出力し、符号が正である場合には前記第1の値をそのまま出力する第1の1の補数絶対値演算手段と、

前記第2の値の符号が負である場合には前記第2の値を1の補数表現に変換して出力し、符号が正である場合には前記第2の値をそのまま出力する第2の1の補数絶対値演算手段と、

前記第1の1の補数絶対値演算手段および前記第2の1の補数絶対値演算手段から出力された前記第1の値と前記第2の値とに基づいて、前記第1の値の絶対値と前記第2の値の絶対値とのどちらが大きいかを表す指示情報を出力する指示情報出力手段と、

を備える演算器。

【請求項2】

前記指示情報に基づいて前記第1の値の絶対値と前記第2の値の絶対値とのうちの絶対値が大きい方の値を出力する選択手段を備える請求項1に記載の演算器。

【請求項3】

前記選択手段は、前記第1の値の絶対値と前記第2の値の絶対値とが等しく、前記第1の

10

20

値の符号と前記第2の値の符号とが異なる場合には、前記第1の値と前記第2の値のうちの符号が正である方の値を選択して出力する請求項2に記載の演算器。

【請求項4】

前記指示情報出力手段は、前記第1の1の補数絶対値演算手段より出力された前記第1の値から前記第2の1の補数絶対値演算手段より出力された前記第2の値を減じて出力する請求項1に記載の演算器。

【請求項5】

前記第2の1の補数絶対値演算手段が、前記第2の値の符号が負である場合には前記第2の値が1の補数表現に変換された値の各ビットを反転させ、符号が正である場合には前記第2の値そのままの値の各ビットを反転させて出力する反転手段を備え、

10

前記指示情報出力手段は、前記第1の1の補数絶対値演算手段から出力された前記第1の値と前記第2の1の補数絶対値演算手段から出力された前記第2の値との和を求め、この和に更に1を加えて出力する請求項1に記載の演算器。

【請求項6】

前記指示情報出力手段からの出力が0であるか否かを判断するゼロ値判断手段を備え、前記選択手段は、前記指示情報出力手段からの出力が負であるか、または、前記指示情報出力手段からの出力が0でありかつ前記第2の値の符号が負である場合に前記第2の値を選択して出力し、それ以外の場合に前記第1の値を選択して出力する請求項2または請求項3に記載の演算器。

【請求項7】

20

2の補数表現による第1の値の絶対値と2の補数表現による第2の値の絶対値とのどちらが大きいかを表す情報を求める演算方法であって、前記第1の値の符号が負である場合には前記第1の値を1の補数表現に変換し、符号が正である場合には前記第1の値をそのままにしておく第1の1の補数絶対値演算処理工程と、

前記第2の値の符号が負である場合には前記第2の値を1の補数表現に変換し、符号が正である場合には前記第2の値をそのままにしておく第2の1の補数絶対値演算処理工程と、

前記第1の1の補数絶対値演算処理工程および前記第2の1の補数絶対値演算処理工程において処理された前記第1の値と前記第2の値とに基づいて、前記第1の値の絶対値と前記第2の値の絶対値とのどちらが大きいかを表す指示情報を求める指示情報処理工程と、

30

を包含する演算方法。

【請求項8】

前記指示情報に基づいて、前記第1の値と前記第2の値のうちの絶対値が大きい方の値を選択する選択処理工程を包含する請求項7に記載の演算方法。

【請求項9】

前記選択処理工程において、前記第1の値の絶対値と前記第2の値の絶対値とが等しく、前記第1の値の符号と前記第2の値の符号とが異なる場合には、前記第1の値と前記第2の値のうちの符号が正である方の値を絶対値が大きい値として選択する請求項8に記載の演算方法。

【請求項10】

40

前記指示情報処理工程において、前記第1の1の補数絶対値演算処理工程で処理された前記第1の値から前記第2の1の補数絶対値演算処理工程で処理された前記第2の値を減じる請求項7に記載の演算方法。

【請求項11】

前記の2の補数絶対値演算処理工程において、前記第2の値の符号が負である場合には前記第2の値が1の補数表現に変換された値の各ビットを反転させ、符号が正である場合には前記第2の値そのままの値の各ビットを反転させ、

前記指示情報処理工程において、前記第1の1の補数絶対値演算処理工程で処理された前記第1の値と前記第2の1の補数絶対値演算処理工程で処理された前記第2の値との和を求め、この和に更に1を加える請求項7に記載の演算方法。

50

【請求項 1 2】

前記指示情報処理工程による処理結果が 0 であるか否かを判断するゼロ値判断処理工程を含み、

前記選択処理工程において、前記指示情報処理工程の処理結果が負であるか、または、前記指示情報処理工程の処理結果が 0 でありかつ前記第 2 の値の符号が負である場合に前記第 2 の値を絶対値が大きい値として選択し、それ以外の場合に前記第 1 の値を絶対値が大きい値として選択する請求項 8 または請求項 9 に記載の演算方法。

【請求項 1 3】

実行命令を記憶する実行命令記憶手段と、

実行命令記憶手段から読み出した前記実行命令をデコードした制御信号を出力するデコード手段と、 10

前記制御信号に基づいて演算を行う演算器と、

を備えたプロセッサであって、

前記演算器は、入力された 2 の補数表現による第 1 の値と、入力された 2 の補数表現による第 2 の値と、を前記制御信号に基づいて、

前記第 1 の値の符号もしくは前記第 2 の値の符号が負である場合には符号が負である値を 1 の補数表現に変換し、前記第 2 の値の符号もしくは前記第 2 の値の符号が正である場合には符号が正である値をそのままにする変換処理を行い、

前記変換処理後の第 1 の値と前記変換処理後の第 2 の値との大小を比較して、大小の違いがある場合には大きい方の前記変換処理を行う前の前記第 1 の値もしくは前記第 2 の値を 20
出力し、大小の違いがなく前記変換処理を行う前の前記第 1 の値の符号と前記第 2 の値の符号とが異なる場合には、前記変換処理を行う前の前記第 1 の値もしくは前記第 2 の値のうち符号が正の方の値を出力するプロセッサ。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、複数のデータ（値）のなかから最大の絶対値をもつデータを求めるために、2 つのデータのうち絶対値が大きいものを求める処理を実行する演算器および演算方法およびプロセッサに関する。特に、絶対値の比較演算が大きな負担となるデジタル化された音楽データなどの A V データの圧縮に使用する演算器および演算方法およびプロセッサに関 30
する。

【0002】

【従来の技術】

現在の A V システムでは信号をデジタル信号として取り扱うことが多い。デジタル信号は、アナログ信号に比べ周囲の雑音に強く、記憶媒体に記憶させたり、信号を伝送させた場合でも、コンパクトディスク（C D : C o m p a c t D i s c）のように極めて良好に信号を再生することができる。この様に優れた特質を持つデジタルデータを記憶媒体に記憶させる場合や信号伝送する場合には、より多くのデータを記憶したり伝送するためにデジタルデータの圧縮がなくてはならない技術となっている。この圧縮技術を標準化するために設立された団体として M P E G（M o v i n g P i c t u r e E x p e r t s 40
G r o u p）がある。M P E G により規格化された圧縮方式には、動画像の圧縮方式として M P E G - 1、M P E G - 2 などがあり、音声の圧縮方式として M P E G - A u d i o がある。

【0003】

M P E G - A u d i o では次のような技術を用いて音声の圧縮を行っている。まず、時間領域にあるオーディオデータを周波数領域の 3 2 個の均等な帯域に変換する。つぎに、各帯域内のデータを、波形と倍率とに分離する。波形は、最大振幅を 1. 0 に近づくように正規化し、サンプルとして符号化される。つまり、波形の正規化とは、波形の相似性を損なうことなく波形信号を表現する個々の値 A が、 $-1 \leq A < 1$ の範囲に収まるようにする 50
処理である。また、正規化時の倍率もスケール・ファクタとして符号化される。その後、

ヒトの聴覚特性を考慮して、ヒトが聴き取ることができる限界の量子化精度で好適なビットを割り当てることにより圧縮が行われる。

【0004】

また、周波数帯域毎に波形信号の大きさを一定のビット幅を用いて表現する必要があるが、最小のビット幅で無駄なく波形信号の大きさをすべて表現するためにも、波形信号の個々の値の中から絶対値が最大の値を求める必要がある。

【0005】

今日では、これらの処理はプログラマブルなプロセッサで行われることが多い。絶対値が最大の値を求めるために用いられる従来の一般的なプロセッサの構成を図7のブロック図に示す。このプロセッサは、プロセッサの制御を行う制御部1と、A L U 6などを有し演算を行うデータバス部2とを備えるものである。

10

【0006】

制御部1は、プログラムカウンタ3、実行命令メモリ4、デコーダ5から構成される。実行命令メモリ4には、実行命令がプログラムとして格納されている。プログラムカウンタ3は、次に実行すべき実行命令が記憶されている実行命令メモリ4上のアドレスを示しており、通常はデコードからの制御信号12によってインクリメントされ、分岐命令等が実行される場合は分岐先のアドレスをセットする。デコーダ5は、実行命令メモリから出力された実行命令とフラグ設定部9の値に基づいて、データバス部2に制御信号10を出力し、プログラムカウンタ3に制御信号12を出力する。

【0007】

20

データバス部2は、A L U 6、レジスタファイル7、データメモリ8、フラグ設定部9から構成される。A L U 6は、制御信号10に基づいた演算をレジスタファイル7から出力された値に対して行い、その結果をレジスタファイル7に格納する。レジスタファイル7は、複数のレジスタが集まったものであり、制御信号10によりA L U 6やデータメモリ8に内容を出力するレジスタが指定される。同じく制御信号10により、A L U 6、データメモリ8からの出力を書き込むレジスタが指定される。また、レジスタファイル7からアドレス信号11がデータメモリ8に出力されるが、制御信号10により指定されたレジスタファイル7内のレジスタの内容がアドレス信号11として出力する。

【0008】

レジスタファイル7には、A, B, C, D, Pの5つのレジスタがある。データメモリ8は、演算に使用されるデータを保持するためのメモリである。データメモリ8内のアドレスがアドレス信号11により指定されて、レジスタファイル7からの出力が書き込まれる。また、アドレス信号11により指定されたデータメモリ8内のアドレスの内容が読み出されてレジスタファイル7に書き込まれる。フラグ設定部9には、NフラグレジスタとZフラグレジスタからそれぞれ構成されるNフラグとZフラグがある。Nフラグには、A L U 6から出力された演算結果が負の場合に1がセットされ、それ以外の場合には0をセットする。ZフラグにはA L U 6から出力された演算結果がゼロの場合に1がセットされ、それ以外の場合には0をセットする。

30

【0009】

なお、A, B, C, D, Pの5つのレジスタのビット幅、データメモリ8のビット幅、A L U 6の入出力、演算精度のビット幅はすべて4ビットとする。

40

【0010】

このプロセッサは、次のような実行命令を備えている。

【0011】

「move a, b : レジスタbからレジスタaへのデータ転送」、「move i a, imm : 即値immをレジスタaにセット」、「cmp a, b : レジスタaとレジスタbとのデータ比較。A L U 6において減算が実行されるが、結果はレジスタに書き込まれない(a - b)」、「cmp i a, imm : レジスタiと即値immの比較。A L U 6において減算が実行されるが、結果はレジスタに書き込まれない(a - imm)」、「add i a, imm : レジスタaの値に即値immを足してレジスタaにセッ

50

ト」、「abs a, b:レジスタbの2の補数の絶対値の値をレジスタaにセット」、「jmp label:無条件にlabelにジャンプ」、「jmp(b) label: NフラグとZフラグが共に0の場合にlabelにジャンプ」、「jmp(e) label: Zフラグが1の場合にlabelにジャンプ」。

【0012】

絶対値が最も大きい値を求める処理を、従来の一般的なプロセッサで行う場合のプログラムを図13に示す。

【0013】

この例において、絶対値が最も大きい値を求める対象となるデータは、データメモリ8内の0番地から3番地までのメモリ領域上のアドレスに保持されている。

10

【0014】

ステップ(1)において、レジスタPに0(0番地)をセットする。ステップ(2)において、レジスタCに絶対値が最小である0をダミーデータとしてセットする。ステップ(3)では、レジスタPが示すメモリのアドレスの値をレジスタAにセットする。ステップ(4)では、レジスタAの値の絶対値演算の結果をレジスタBにセットする。ステップ(5)では、レジスタCの値の絶対値演算の結果をレジスタDにセットする。ステップ(6)では、レジスタD, Bの値を比較する。ステップ(7)では、ステップ(6)での比較結果が正の場合(フラグN=0, フラグZ=0)は、ステップ(9)に進む。それ以外の場合はステップ(8)に進む。ステップ(8)では、レジスタAの値をレジスタCにデータ転送する。ステップ(9)ではレジスタPの値が示すアドレスがデータ群の最終データのアドレスであるか、つまり3番地に等しいか比較する。ステップ(10)では、ステップ(9)での比較結果が等しい場合(フラグZ=1)に、ステップ(13)に進む。ステップ(11)はレジスタPの値を1増加させて次のデータをポイントする。ステップ(12)では、ステップ(3)に戻る。ステップ(13)では、処理を終了する。これらの処理によりデータメモリ8内の0番地から3番地に格納されているデータの中で、絶対値が最大であるデータをレジスタCに求めることができる。

20

【0015】

この中で、ステップ(4)～ステップ(6)が、2つの値の絶対値の差を求める絶対値比較演算であり、ステップ(7)～ステップ(8)で2つの値のうちでより大きい絶対値を持つ方を求めている。

30

【0016】

ここで図16に示す4つの具体的データから図13に示すプログラムによって絶対値が最大のデータが求まる様子を示す。

【0017】

ステップ(1)では、レジスタPに0をセットする。ステップ(2)では、レジスタCに0をセットする。

【0018】

繰返し1回目のステップ(3)では、レジスタAにアドレス{0}内に保持されているデータの値「3」をセットする。ステップ(4)では、レジスタAの値「3」の絶対値演算の結果「3」がレジスタBにセットされる。ステップ(5)では、レジスタCの値「0」の絶対値演算の結果「0」がレジスタDにセットされる。ステップ(6)では、レジスタDの値「0」とレジスタBの値「3」を比較する。比較(減算)結果が「0-3=-3」で、負なのでフラグN=1、フラグZ=0をセットする。ステップ(7)では、フラグN=1、フラグZ=0なので、ステップ(8)に進む。ステップ(8)では、レジスタAの値「3」をレジスタCにセットする。ステップ(9)ではレジスタPの値「0」と最終データのアドレス{3}を比較する。比較(減算)結果が「0-3=-3」で、負なので、フラグN=1、フラグZ=0をセットする。ステップ(10)では、フラグZ=0なのでステップ(11)に進む。ステップ(11)では、レジスタPの値を1増加した値「1」とする。ステップ(12)ではステップ(3)に戻る。

40

【0019】

50

繰返し2回目のステップ(3)では、レジスタAにアドレス{1}内に保持されているデータの値「-5」をセットする。ステップ(4)では、レジスタAの値「-5」の絶対値演算の結果「5」がレジスタBにセットされる。ステップ(5)では、レジスタCの値「3」の絶対値演算の結果「3」がレジスタDにセットされる。ステップ(6)では、レジスタDの値「3」とレジスタBの値「5」を比較する。比較(減算)結果が「 $3 - 5 = -2$ 」で、負なのでフラグN=1、フラグZ=0をセットする。ステップ(7)では、フラグN=1、フラグZ=0なので、ステップ(8)に進む。ステップ(8)では、レジスタAの値「-5」をレジスタCにセットする。ステップ(9)ではレジスタPの値「1」と最終データのアドレス{3}を比較する。比較(減算)結果が「 $1 - 3 = -2$ 」で、負なので、フラグN=1、フラグZ=0をセットする。ステップ(10)では、フラグZ=0
10
なのでステップ(11)に進む。ステップ(11)では、レジスタPの値を1増加した値「2」とする。ステップ(12)ではステップ(3)に戻る。

【0020】

繰返し3回目のステップ(3)では、レジスタAにアドレス{2}内に保持されているデータの値、「-2」をセットする。ステップ(4)では、レジスタAの値「-2」の絶対値演算の結果「2」がレジスタBにセットされる。ステップ(5)では、レジスタCの値「-5」の絶対値演算の結果「5」がレジスタDにセットされる。ステップ(6)では、レジスタDの値「5」とレジスタBの値「2」を比較する。比較(減算)結果が「 $5 - 2 = 3$ 」で、正なのでフラグN=0、フラグZ=0をセットする。ステップ(7)では、フラグN=0、フラグZ=0なので、ステップ(9)に進む。ステップ(9)ではレジスタ
20
Pの値「2」と最終データのアドレス{3}を比較する。比較(減算)結果が「 $2 - 3 = -1$ 」で、負なので、フラグN=1、フラグZ=0をセットする。ステップ(10)では、フラグZ=0なのでステップ(11)に進む。ステップ(11)では、レジスタPの値を1増加した値「3」とする。ステップ(12)ではステップ(3)に戻る。

【0021】

繰返し4回目のステップ(3)では、レジスタAにアドレス{3}内に保持されているデータの値「6」をセットする。ステップ(4)では、レジスタAの値「6」の絶対値演算の結果「6」がレジスタBにセットされる。ステップ(5)では、レジスタCの値「-5」の絶対値演算の結果「5」がレジスタDにセットされる。ステップ6では、レジスタDの値「5」とレジスタBの値「6」を比較する。比較(減算)結果が「 $5 - 6 = -1$ 」で
30
、負なのでフラグN=1、フラグZ=0をセットする。ステップ(7)では、フラグN=1、フラグZ=0なので、ステップ(8)に進む。ステップ(8)では、レジスタAの値「6」をレジスタCにセットする。ステップ(9)ではレジスタPの値「4」と最終データのアドレス{3}を比較する。比較(減算)結果が「 $3 - 3 = 0$ 」なので、フラグN=0、フラグZ=1をセットする。ステップ(10)では、フラグZ=1なのでステップ(13)に進む。ステップ(13)で処理を終了する。

【0022】

以上の処理を行うことにより、レジスタCに絶対値が最大である値「6」を求めることができる。

【0023】

しかし、このように多くの計算を必要とする処理のために、従来のプロセッサにおいても専用の演算器を用いて高速化を図ってきた。

【0024】

図8は、専用の演算器を備えた従来のプロセッサである。図8の従来のプロセッサも図7のプロセッサと同じく制御部1とデータバス部2とを備えている。制御部1は図7のプロセッサのものと同じ働きをする。データバス部2では、図7のプロセッサのデータバス部2に加えて専用の演算器である2の補数絶対値減算器14を備えている。2の補数絶対値減算器14は、制御信号10により2の補数絶対値減算処理を、レジスタファイル7から出力された値に対して行う。

【0025】

10

20

30

40

50

A L U 6 , レジスタファイル 7、データメモリ 8 も図 7 のプロセッサと同じ働きをする。フラグ設定部 9 は図 7 のプロセッサと同じ働きに加え、2 の補数絶対値減算器 1 4 の出力の結果をも反映させたフラグ設定を行う。

【0026】

2 の補数絶対値減算器 1 4 を図 9 に示す。2 の補数絶対値減算器 1 4 は、第 1 の 2 の補数絶対値演算器 5 0 と第 2 の 2 の補数絶対値演算器 5 1 と減算器 2 2 を備える。第 1 の 2 補数絶対値演算器 5 0 は、反転器 5 2 と + 1 加算器 5 3 とセクタ 5 4 とを備える。反転器 5 2 は入力されたデータ (入力 A (6 9)) のビットを全て反転する。+ 1 加算器 5 3 は反転器 5 2 から出力された値に 1 を加算し、入力 A (6 9) に 2 の補数化を行う手順操作を施し、つまり、ビットを反転して 1 を加算することにより符号を反転させた符号反転後の値 6 0 を出力する。セクタ 5 4 は、入力 A (6 9) の符号が負の場合は符号反転後の値 6 0 を選択し、それ以外の場合は入力 A (6 9) を選択して出力する。これにより、入力 A (6 9) が負の値の場合は、絶対値の等しい正の値が、入力 A (6 9) が正の場合は入力 A (6 9) がそのまま第 1 の 2 補数絶対値演算器 5 0 (セクタ 5 4) から出力 (6 2) される。以上の処理を絶対値演算処理と呼ぶ。

10

【0027】

第 2 の 2 の補数絶対値演算器 5 1 も、第 1 の 2 の補数絶対値演算器 5 0 と同じく反転器 5 5 と + 1 加算器 5 6 とセクタ 5 7 を備える。反転器 5 5 は入力されたデータ (入力 B (7 0)) のビットを全て反転する。+ 1 加算器 5 6 は反転器 5 5 から出力された値に 1 を加算し、入力 B (7 0) に 2 の補数化を行う操作を施した符号反転後の値 6 1 を出力する。セクタ 5 7 では、入力 B (7 0) の符号が負の場合は符号反転後の値 6 1 を選択し、それ以外の場合は入力 B (7 0) を選択して出力される。これにより、入力 B (7 0) の符号が負の値の場合は、絶対値の等しい正の値が、入力 B (7 0) の符号が正の場合は入力 B (7 0) がそのまま第 2 の 2 補数絶対値演算器 5 1 (セクタ 5 7) から出力 (6 3) される。

20

【0028】

減算器 2 2 では、絶対値演算処理後の値 6 2 から、絶対値演算処理後の値 6 3 を減じて絶対値比較演算の結果 6 5 が出力される。

【0029】

図 8 のプロセッサも図 7 のプロセッサと同じく、A , B , C , D , P の 5 つのレジスタのビット幅、データメモリ 8 のビット幅、A L U 6 の入出力のビット幅はすべて 4 ビットである。また、2 の補数絶対値減算器 1 4 の入出力、演算精度のビット幅も 4 ビットである。

30

【0030】

図 8 のプロセッサは、図 7 のプロセッサと同じ実行命令に加え、2 の補数絶対値減算器 1 4 で実行される以下の実行命令を備える。

【0031】

「a b c m p a , b : レジスタ a とレジスタ b との絶対値の比較。結果はレジスタに書き込まれない。(| a | - | b |) 」

従来の一般的なプロセッサでの例と同じく、2 の補数絶対値減算器 1 4 を備えたプロセッサで、図 1 6 に示す 4 つのデータから絶対値が最大のデータを求める手順を、2 の補数絶対値減算器 1 4 を用いる「a b c m p」命令を中心に説明する。

40

【0032】

2 の補数絶対値減算器 1 4 を備えた従来のプロセッサでの絶対値が最大の値を求める処理のプログラムを図 1 4 に示す。ステップ (1) からステップ (3)、ステップ (7) からステップ (1 3) までは従来の一般的なプロセッサの場合と同じである。ステップ (1 4) は、「a b c m p」命令を使ってレジスタ C の値とレジスタ A の値の絶対値を比較するステップである。ステップ (1 4) では、従来の一般的なプロセッサで行う場合のステップ (4) からステップ (7) までの処理と同じ処理を行う。2 の補数絶対値減算器 1 4 の入力 A (6 9) にはレジスタ C の値、入力 B (7 0) にはレジスタ A の値が入力される。

50

【0033】

なお、データメモリ8に格納されているデータはMSB（4ビットのうちの最上位のビット位置）に符号ビットがある2の補数表現によるものとする。数値の符号はMSBの値が1の場合は負、0の場合は正である。また、数表記として2進数で表す場合はビット列の前に「0b」を付けるものとする。

【0034】

入力A、Bは4ビットなので表現が可能な値の範囲は-8～7（0b1000～0b0111）である。

【0035】

ステップ（1）、（2）での動作は、従来の一般的なプロセッサの場合と同じである。 10

【0036】

繰返し1回目のステップ（3）での動作は、従来の一般的なプロセッサの場合と同じである。

【0037】

ステップ（14）では、2の補数絶対値減算器14を使った演算が行われる。この時の各数値の状態を図17を用いて説明する。

【0038】

レジスタCの値「0（0b0000）」は入力A（69）に、レジスタAの値「3（0b0011）」は入力B（70）にそれぞれ入力される。

【0039】

第1の2の補数絶対値演算器50には、入力A（69）の値「0（0b0000）」が入力される。反転器52でビット反転されて、+1加算器53で1加算されるので、符号反転後の値60は「0（0b0000）」となる。セクタ54では、入力A（69）の値「0（0b0000）」の符号を判断して、入力A（69）を選択して絶対値演算処理後の値62として「0（0b0000）」を出力する。 20

【0040】

第2の2の補数絶対値演算器51には、入力B（70）の値「3（0b0011）」が入力される。反転器55でビット反転されて、+1加算器56で1加算されるので、符号反転後の値61は「-3（0b1101）」となる。セクタ57では、入力B（70）の値「3（0b0011）」の符号を判断して、入力B（70）を選択して絶対値演算処理後の値63として「3（0b0011）」を出力する。 30

【0041】

減算器22では、絶対値演算処理後の値62から絶対値演算処理後の値63を減算するので、絶対値比較演算の結果65として「-3（0b1101）」を出力する。

【0042】

フラグ設定部9では入力された値「-3（0b1101）」を判断して、Nフラグレジスタ=1、Zフラグレジスタ=0をセットする。

【0043】

ステップ（9）～（11）での動作は、従来の一般的なプロセッサの場合と同じである。

【0044】

繰返し2回目のステップ（3）は、従来の一般的なプロセッサの場合と同じである。 40

【0045】

ステップ（14）では、2の補数絶対値減算器14を使った演算が行われる。この時の各数値の状態を図18を用いて説明する。

【0046】

レジスタCの値「3（0b0011）」は入力A（69）に、レジスタAの値「-5（0b1011）」は入力B（70）にそれぞれ入力される。

【0047】

第1の2の補数絶対値演算器50には、入力A（69）の値「3（0b0011）」が入力される。反転器52でビット反転されて、+1加算器53で1加算されるので、符号反 50

転後の値60は「-3(0b1101)」となる。セクタ54では、入力A(69)の値「3(0b0011)」の符号を判断して、入力A(69)を選択して絶対値演算処理後の値62として「3(0b0011)」を出力する。

【0048】

第2の2の補数絶対値演算器51には、入力B(70)の値「-5(0b1011)」が入力される。反転器55でビット反転されて、+1加算器56で1加算されるので、符号反転後の値61は「5(0b0101)」となる。セクタ57では、入力B(70)の値「-5(0b1011)」の符号を判断して、符号反転後の値61を選択されて絶対値演算処理後の値63として「5(0b0101)」を出力する。

【0049】

減算器22では、絶対値演算処理後の値62から絶対値演算処理後の値63を減算するので、絶対値比較演算の結果65として「-2(0b1110)」を出力する。

【0050】

フラグ設定部9では入力された値「-2(0b1110)」を判断して、Nフラグレジスタ=1、Zフラグレジスタ=0をセットする。

【0051】

ステップ(7)～(12)での動作は、従来の一般的なプロセッサの場合と同じである。

【0052】

繰返し3回目のステップ(3)は、従来の一般的なプロセッサの場合と同じである。

【0053】

ステップ(14)では、2の補数絶対値減算器14を使った演算が行われる。この時の各数値の状態を図19を用いて説明する。

【0054】

レジスタCの値「-5(0b1011)」は入力A(69)に、レジスタAの値「-2(0b1110)」は入力B(70)にそれぞれ入力される。

【0055】

第1の2の補数絶対値演算器50には、入力A(69)の値「-5(0b1011)」が入力される。反転器52でビット反転されて、+1加算器53で1加算されるので、符号反転後の値60は「5(0b0101)」となる。セクタ54では、入力A(69)の値「-5(0b1011)」の符号を判断して、符号反転後の値60を選択して絶対値演算処理後の値62として「5(0b0101)」を出力する。

【0056】

第2の2の補数絶対値演算器51には、入力B(70)の値「-2(0b1110)」が入力される。反転器55でビット反転されて、+1加算器56で1加算されるので、符号反転後の値61は「2(0b0010)」となる。セクタ57では、入力B(70)の値「-2(0b1110)」の符号を判断して、符号反転後の値61を選択して絶対値演算処理後の値63として「2(0b0010)」を出力する。

【0057】

減算器22では、絶対値演算処理後の値62から絶対値演算処理後の値63を減算するので、絶対値比較演算の結果65として「3(0b0011)」を出力する。

【0058】

フラグ設定部9では入力された値「3(0b0011)」を判断して、Nフラグレジスタ=0、Zフラグレジスタ=0をセットする。

【0059】

ステップ(7)～(12)での動作は、従来の一般的なプロセッサの場合と同じである。

【0060】

繰返し4回目のステップ(3)は、従来の一般的なプロセッサの場合と同じである。

【0061】

ステップ(14)では、2の補数絶対値減算器14を使った演算が行われる。この時の各数値の状態を図20を用いて説明する。

10

20

30

40

50

【0062】

レジスタCの値「 -5 (0b1011)」は入力A (69) に、レジスタAの値「 6 (0b0110)」は入力B (70) にそれぞれ入力される。

【0063】

第1の2の補数絶対値演算器50には、入力A (69) の値「 -5 (0b1011)」が入力される。反転器52でビット反転されて、+1加算器53で1加算されるので、符号反転後の値60は「 5 (0b0101)」となる。セクタ54では、入力A (69) の値「 -5 (0b1011)」の符号を判断して、符号反転後の値60を選択して絶対値演算処理後の値62として「 5 (0b0101)」を出力する。

【0064】

第2の2の補数絶対値演算器51には、入力B (70) の値「 6 (0b0110)」が入力される。反転器55でビット反転されて、+1加算器56で1加算されるので、符号反転後の値61は「 -6 (0b1010)」となる。セクタ57では、入力B (70) の値「 6 (0b0110)」の符号を判断して、入力B (70) を選択して絶対値演算処理後の値63として「 6 (0b0110)」を出力する。

【0065】

減算器22では、絶対値演算処理後の値62から絶対値演算処理後の値63を減算するので、絶対値比較演算の結果65として「 -1 (0b1111)」を出力する。

【0066】

フラグ設定部9では入力された値「 -1 (0b1111)」を判断して、Nフラグレジスタ=1、Zフラグレジスタ=0をセットする。

【0067】

ステップ(7)～(12)での動作は、従来の一般的なプロセッサの場合と同じである。

【0068】

以上の処理を行うことにより、レジスタCに絶対値が最大である値「 6 」を求めることができる。

【0069】

従来のプロセッサでは、このように、2の補数絶対値減算器を用いて、演算処理の高速化を行っている。

【0070】**【発明が解決しようとする課題】**

しかし、従来のプロセッサで使用されている2の補数絶対値減算器では、所定のビット幅で表現が可能な最も絶対値が大きい負の値（ -8 から7までの値が表現が可能な4ビットでは「 -8 」）が入力されると、不具合を生じる場合がある。この例として、レジスタAに「 0 (0b0000)」、レジスタCに「 -8 (0b1000)」がそれぞれセットされている場合に、「a b c m p C, A」ステップ(14)が実行される動作を図21を用いて説明する。

【0071】

レジスタCの値「 -8 (0b1000)」は入力A (69) に、レジスタAの値「 0 (0b0000)」は入力B (70) にそれぞれ入力される。

【0072】

入力A (69) に「 -8 (0b1000)」が入力されると、反転器52により全てのビットが反転され「 $(0b0111)$ 」が反転器52から出力される。この値に+1加算器53で1が加算されると、オーバーフローを起こす。このために、+1加算器53からは、符号反転後の値60として「 -8 (0b1000)」を出力する。つまり、「 -8 (0b1000)」の符号を反転させて正の値（絶対値）が求まるべきであるが、オーバーフローのために負の符号のままとなっている。セクタ54では、入力A (69) の値「 -8 (0b1000)」の符号を判断して、符号反転後の（符号が正であるべき）値60を選択するが、「 -8 (0b1000)」が絶対値演算の結果62として出力する。

【0073】

10

20

30

40

50

入力B (70) に「0 (0b0000)」が入力される。反転器55でビット反転されて、+1加算器56で1加算されるので、符号反転後の値61は「0 (0b0000)」となる。セクタ57では、入力B (70) の値「0 (0b0000)」の符号を判断して、入力B (70) を選択して絶対値演算処理後の値63として「0 (0b0000)」を出力する。

【0074】

2の補数絶対値減算器14では、絶対値演算処理後の値62から絶対値演算処理後の値63を減算するので、絶対値比較演算の結果65として「-8 (0b1000)」を出力する。つまり、 $|-8| - |0|$ の結果が負となる。

【0075】

フラグ設定部9では入力された値「-8 (0b1000)」を判断して、Nフラグレジスタ=1、Zフラグレジスタ=0をセットする。

【0076】

ステップ(7)では、Nフラグレジスタ=1、Zフラグレジスタ=0なので、ステップ(8)に進み、ステップ(8)では、レジスタAの値「0 (0b0000)」がレジスタCにセットされる。

【0077】

「-8 (0b1000)」と「0 (0b0000)」では、「-8 (0b1000)」の方が絶対値がはるかに大きく、本来ならばレジスタCに「-8 (0b1000)」がセットされなければならないが、従来のプロセッサではオーバーフローの発生のために間違っ

【0078】

このため、2の補数絶対値減算器14の入力に、ビット表現が可能な範囲で絶対値が最大である負の値が入力されないようにする必要がある。この制限をハードウェアで行う場合は、新たな回路を追加しなければならないので回路の増加、引いては回路基板の面積の拡大を招く。また、上記の制限をソフトウェアで行う場合は、ステップ数が増加するため演算処理が遅くなる。

【0079】

さらに、2の補数絶対値減算器14を備えた従来のプロセッサでは、絶対値が同じ値があった場合、符号の正負に関わらず先に出現した値が、絶対値が最大の値としてレジスタCに残る。このため、データ群から絶対値が最大の数値を求める処理を行った結果、符号が負の値(例えば-4)が求まった場合に、求まった値と絶対値が同じで符号が正の値(例えば4)がデータ群のなかに存在したか否かは不明である。

【0080】

ここで、2の補数で表現された複数のデータ(数値)を、一定のビット幅で表現される値に対応させる場合を考える。このとき、複数のデータを最小のビット幅で無駄なく表現したい。そのためには、絶対値が最大となるデータAをデータ群から求め、 $-A \sim A$ の範囲にある0を含む $2A + 1$ 個の値を表現するために最低限必要なビット幅を用意すればよい。

【0081】

従来のプロセッサで、絶対値が最大であるデータを求める処理を行った結果、レジスタCに「-4 (0b1100)」が絶対値が最大のデータとして求まったとする。しかし、従来のプロセッサでは、調べたデータ群の中に「4 (0b0100)」が存在したか否かは不明である。このため、表現する必要があるデータ範囲を、「-4」から「4」の範囲と考えなければならない。この場合「-4」から「4」を表現するためには0を含む9個の値が必要であり、必要なビット幅は4ビット(0000~1000)となる。しかし、ここで仮に、絶対値が最大の値は「-4」であるが、データ群のなかに絶対値が同じ「4」は存在しないことが明確であれば、表現する必要があるデータ範囲を、「-4」から「3」とすることができる。この場合には、「-4」から「3」を表現するためには8個の値が必要であり、必要なビット幅は3ビット(000~111)となる。このように、従来

10

20

30

40

50

のプロセッサでは上記の処理を行った場合は、必要以上に無駄なビットを割り当ててしまう場合があった。

【0082】

また、従来の2の補数絶対値減算器14では、+1加算器が2つあるので+1加算に要する演算時間のために2の補数絶対値減算器14全体としての演算時間が長くなっている。さらに、+1加算器の2つ分のスペースのために、2の補数絶対値減算器14の大きさも大きくなっている。

【0083】

本発明は、このような実情に鑑みてなされたものであり、従来の技術では正しく扱うことができなかった所定のビット幅で表現が可能な範囲において絶対値が最大の負の値を正しく扱うことができ、複数のデータを一定のビット幅で表す値に対応させる場合に、より無駄のないビット幅を決定することができる演算方法および演算器およびプロセッサを提供することを目的とする。

10

【0084】

【課題を解決するための手段】

本発明の演算器は、2の補数表現による第1の値と2の補数表現による第2の値とを受け取り、前記第1の値の絶対値と前記第2の値の絶対値とのどちらが大きいかを表す情報を出力する演算器であって、前記第1の値の符号が負である場合には前記第1の値を1の補数表現に変換して出力し、符号が正である場合には前記第1の値をそのまま出力する第1の1の補数絶対値演算手段と、前記第2の値の符号が負である場合には前記第2の値を1の補数表現に変換して出力し、符号が正である場合には前記第2の値をそのまま出力する第2の1の補数絶対値演算手段と、前記第1の1の補数絶対値演算手段および前記第2の1の補数絶対値演算手段から出力された前記第1の値と前記第2の値とに基づいて、前記第1の値の絶対値と前記第2の値の絶対値とのどちらが大きいかを表す指示情報を出力する指示情報出力手段とを備えており、このことにより上述の目的が達成される。

20

【0085】

前記指示情報に基づいて前記第1の値の絶対値と前記第2の値の絶対値とのうちの絶対値が大きい方の値を出力する選択手段を備えてもよい。

【0086】

前記選択手段は、前記第1の値の絶対値と前記第2の値の絶対値とが等しく、前記第1の値の符号と前記第2の値の符号とが異なる場合には、前記第1の値と前記第2の値のうちの符号が正である方の値を選択して出力してもよい。

30

【0087】

前記指示情報出力手段は、前記第1の1の補数絶対値演算手段より出力された前記第1の値から前記第2の1の補数絶対値演算手段より出力された前記第2の値を減じて出力してもよい。

【0088】

前記第2の1の補数絶対値演算手段が、前記第2の値の符号が負である場合には前記第2の値が1の補数表現に変換された値の各ビットを反転させ、符号が正である場合には前記第2の値そのままの値の各ビットを反転させて出力する反転手段を備え、前記指示情報出力手段は、前記第1の1の補数絶対値演算手段から出力された前記第1の値と前記第2の1の補数絶対値演算手段から出力された前記第2の値との和を求め、この和に更に1を加えて出力してもよい。

40

【0089】

前記指示情報出力手段からの出力が0であるか否かを判断するゼロ値判断手段を備え、前記選択手段は、前記指示情報出力手段からの出力が負であるか、または、前記指示情報出力手段からの出力が0でありかつ前記第2の値の符号が負である場合に前記第2の値を選択して出力し、それ以外の場合に前記第1の値を選択して出力してもよい。

【0090】

本発明の演算方法は、2の補数表現による第1の値の絶対値と2の補数表現による第2の

50

値の絶対値とのどちらが大きいかを表す情報を求める演算方法であって、前記第1の値の符号が負である場合には前記第1の値を1の補数表現に変換し、符号が正である場合には前記第1の値をそのままにしておく第1の1の補数絶対値演算処理工程と、前記第2の値の符号が負である場合には前記第2の値を1の補数表現に変換し、符号が正である場合には前記第2の値をそのままにしておく第2の1の補数絶対値演算処理工程と、前記第1の1の補数絶対値演算処理工程および前記第2の1の補数絶対値演算処理工程において処理された前記第1の値と前記第2の値とに基づいて、前記第1の値の絶対値と前記第2の値の絶対値とのどちらが大きいかを表す指示情報を求める指示情報処理工程とを包含しており、これにより上述の目的が達成される。

【0091】

10

前記指示情報に基づいて、前記第1の値と前記第2の値のうちの絶対値が大きい方の値を選択する選択処理工程を包含してもよい。

【0092】

前記選択処理工程において、前記第1の値の絶対値と前記第2の値の絶対値とが等しく、前記第1の値の符号と前記第2の値の符号とが異なる場合には、前記第1の値と前記第2の値のうちの符号が正である方の値を絶対値が大きい値として選択してもよい。

【0093】

前記指示情報処理工程において、前記第1の1の補数絶対値演算処理工程で処理された前記第1の値から前記第2の1の補数絶対値演算処理工程で処理された前記第2の値を減じてもよい。

20

【0094】

前記の2の補数絶対値演算処理工程において、前記第2の値の符号が負である場合には前記第2の値が1の補数表現に変換された値の各ビットを反転させ、符号が正である場合には前記第2の値そのままの値の各ビットを反転させ、前記指示情報処理工程において、前記第1の1の補数絶対値演算処理工程で処理された前記第1の値と前記第2の1の補数絶対値演算処理工程で処理された前記第2の値との和を求め、この和に更に1を加えてもよい。

【0095】

前記指示情報処理工程による処理結果が0であるか否かを判断するゼロ値判断処理工程を含み、前記選択処理工程において、前記指示情報処理工程の処理結果が負であるか、または、前記指示情報処理工程の処理結果が0でありかつ前記第2の値の符号が負である場合に前記第2の値を絶対値が大きい値として選択し、それ以外の場合に前記第1の値を絶対値が大きい値として選択してもよい。

30

【0096】

本発明のプロセッサは、実行命令を記憶する実行命令記憶手段と、実行命令記憶手段から読み出した前記実行命令をデコードした制御信号を出力するデコード手段と、前記制御信号に基づいて演算を行う演算器と、を備えるプロセッサであって、前記演算装置は、入力された2の補数表現による第1の値と、入力された2の補数表現による第2の値と、を前記制御信号に基づいて、前記第1の値の符号もしくは前記第2の値の符号が負である場合には符号が負である値を1の補数表現に変換し、前記第2の値の符号もしくは前記第2の値の符号が正である場合には符号が正である値をそのままにする変換処理を行い、前記変換処理後の第1の値と前記変換処理後の第2の値との大小を比較して、大小の違いがある場合には大きい方の前記変換処理を行う前の前記第1の値もしくは前記第2の値を出力し、大小の違いがなく前記変換処理を行う前の前記第1の値と前記第2の値の符号が異なる場合には、前記変換処理を行う前の前記第1の値もしくは前記第2の値のうち符号が正の方の値を出力し、これにより上述の目的が達成される。

40

【0097】**【発明の実施の形態】**

以下、本発明の好ましい実施形態を図面を用いて説明する。なお、実施の形態の説明における数表記として、2進数で表す場合はビット列の前に「0b」を付けるものとし、その

50

ビット列が値を示す場合には、「0b」の前にその値を示すものとする。また、値の符号はMSBの値（最上位のビット）が1の場合は負であり、0の場合は正である。例えば、10進数での「5」を4ビットの2の補数の2進数表記で表すと「5（0b0101）」であり、1の補数の2進数表記で表しても「5（0b0101）」である。また、10進数での「-5」を4ビットの2の補数の2進数表記で表すと「-5（0b1011）」であり、1の補数の2進数表記で表すと「-5（0b1010）」である。

【0098】

（第1の実施の形態）

第1の実施の形態は、本発明に関わる演算器および演算方法およびプロセッサにおける技術をプロセッサに適用したものである。図1に示すように、プロセッサは、プロセッサの制御を行う制御部1と演算を行うデータバス部2とを備えるものである。

10

【0099】

制御部1は、プログラムカウンタ3、実行命令メモリ4、デコーダ5により構成される。プログラムカウンタ3、実行命令メモリ4、デコーダ5は、図8に示す従来のプロセッサと同じ構成なので説明を省略する。

【0100】

データバス部2は、ALU6、演算器としての1の補数絶対値減算器13、レジスタファイル7、データメモリ8、フラグ設定部9により、構成される。ALU6、レジスタファイル7、データメモリ8は、図8に示した従来のプロセッサと同じ構成なので、説明を省略する。フラグ設定部9は、図8に示した従来のプロセッサと同じくNフラグレジスタとZフラグレジスタを備え、フラグ設定の条件にはALU6の演算結果と共に1の補数絶対値減算器13からの出力の結果も反映される。

20

【0101】

1の補数絶対値減算器13は、制御信号10により指定されるレジスタファイル7内の2つのレジスタのそれぞれが保持する2つの値について1の補数表現を用いた絶対値の比較演算を行い、その比較結果を2つの値の絶対値のうちのどちらが大きいかを表す指示情報として出力する。

【0102】

図2に示す1の補数絶対値減算器13は、本発明に係る演算器である。図2に示すように、1の補数絶対値減算器13は、第1の1の補数絶対値演算手段としての第1の1の補数絶対値演算器20と、第2の1の補数絶対値演算手段としての第2の1の補数絶対値演算器21と、指示情報出力手段としての減算器22を備える。

30

【0103】

第1の1の補数絶対値演算器20は、入力A（69）に対して1の補数表現化の手順操作を用いた絶対値演算処理を行い絶対値演算処理後の値62を出力する。第2の1の補数絶対値演算器21は、入力B（70）に対して1の補数表現化の手順操作を用いた絶対値演算処理を行い絶対値演算処理後の値63を出力する。つまり、第1の1の補数絶対値演算器20および第2の1の補数絶対値演算器21は、入力された値の符号が負の場合に、従来の2の補数表現化の手順操作に換えて1の補数表現化の手順操作により、入力された値の符号を反転させて出力する。減算器22は、絶対値演算処理後の値62から絶対値演算処理後の値63を減算し、絶対値比較演算の結果（指示情報）65を出力する。

40

【0104】

第1の1の補数絶対値演算器20は、反転器23とセレクタ24を備える。反転器23は、入力されたビットを全て反転（1の補数表現化手順操作）して符号反転後の値60を出力する。セレクタ24は、入力A（69）の符号が負の場合は符号反転後の値60を選択し、それ以外の場合は入力A（69）を選択して、絶対値演算処理後の値62として出力する。

【0105】

第2の1の補数絶対値演算器21は、反転器25とセレクタ26を備える。反転器25は、入力されたビットを全て反転（1の補数表現化手順操作）して符号反転後の値60を出

50

力する。セレクタ26では、入力B(70)の符号が負の場合は符号反転後の値61を選択し、それ以外の場合は入力B(70)を選択して、絶対値演算処理後の値63として出力する。

【0106】

図1のプロセッサも図8のプロセッサと同じく、A、B、C、D、Pの5つのレジスタのビット幅、データメモリ8のビット幅、ALU6の入出力のビット幅はすべて4ビットとする。また、1の補数絶対値減算器13の入出力、演算精度のビット幅も4ビットとする。

【0107】

第1の実施の形態のプロセッサは、図8の従来のプロセッサと同じ実行命令を備える。ただし、「abcmp」命令による処理を実行する演算器は図2に示す1の補数絶対値減算器13である。よって、「abcmp」命令では、1の補数表現化された数値による絶対値の比較が行われる。図14に示す、第1の実施の形態のプロセッサで行われる演算処理を「abcmp」命令を使用せず、「abcmp」命令を複数のステップを用いて詳細に表現すると、図51に示すプログラムのようになる。このプログラムにおいて、「abs 1 a, b」命令は、レジスタbの値を1の補数表現化による絶対値にしてレジスタaにセットする命令であり、「jmp (be) label」命令は、Nフラグが0の場合にlabelにジャンプする命令である。また、ステップ(16)、ステップ(17)、ステップ(6)の3ステップが「abcmp」命令に相当する。

【0108】

次に、第1の実施の形態によるプロセッサの動作を説明する。

【0109】

(動作例1)

第1の実施の形態のプロセッサで、図16に示す4つのデータの中から絶対値が最大であるデータを求める動作を、1の補数絶対値減算器13を使用する「abcmp」命令を中心に説明する。

【0110】

第1の実施の形態のプロセッサで絶対値が最大であるデータを求める処理のプログラムは、図8に示した従来のプロセッサと同じ図14のプログラムである。

【0111】

ステップ(1)、(2)での動作は、従来の一般的なプロセッサの場合と同じである。

【0112】

繰返し1回目のステップ(3)での動作は、従来の一般的なプロセッサの場合と同じである。

【0113】

ステップ(14)では、1の補数絶対値減算器13を使った演算が行われる。この時の動作を図22を用いて説明する。

【0114】

レジスタCの値「0(0b0000)」は入力A(69)に、レジスタAの値「3(0b0011)」は入力B(70)にそれぞれ入力される。

【0115】

第1の1の補数絶対値演算器20には、入力A(69)の値「0(0b0000)」が入力される。反転器23で全てのビットが反転されるので、符号反転後の値60は「(0b1111)」となる。セレクタ24では、入力A(69)の値「0(0b0000)」の符号を判断して、入力A(69)を選択して絶対値演算処理後の値62として「(0b0000)」を出力する。

【0116】

第2の1の補数絶対値演算器21には、入力B(70)の値「3(0b0011)」が入力される。反転器25で全てのビットが反転されるので、符号反転後の値61は「(0b1100)」となる。セレクタ26では、入力B(70)の値「3(0b0011)」の

10

20

30

40

50

符号を判断して、入力 B (70) を選択して絶対値演算処理後の値 63 として「(0b0011)」を出力する。

【0117】

減算器 22 では、絶対値演算処理後の値 62 から絶対値演算処理後の値 63 を減算するので、絶対値比較演算の結果 (指示情報) 65 として「(0b1101)」を出力する。

【0118】

フラグ設定部 9 では入力された値「(0b1101)」を判断して、N フラグレジスタ = 1、Z フラグレジスタ = 0 をセットする。

【0119】

ステップ (7) ~ (12) での動作は、従来の一般的なプロセッサの場合と同じである。 10

【0120】

繰返し 2 回目のステップ (3) は、従来の一般的なプロセッサの場合と同じである。

【0121】

ステップ (14) では、1 の補数絶対値減算器 13 を使った演算が行われる。この時の動作を図 23 を用いて説明する。

【0122】

レジスタ C の値「3 (0b0011)」は入力 A (69) に、レジスタ A の値「-5 (0b1011)」は入力 B (70) にそれぞれ入力される。

【0123】

第 1 の 1 の補数絶対値演算器 20 には、入力 A (69) の値「3 (0b0011)」が入力される。反転器 23 で全てのビットが反転されるので、符号反転後の値 60 は「(0b1100)」となる。セレクト 24 では、入力 A (69) の値「3 (0b0011)」の符号を判断して、入力 A (69) を選択して絶対値演算処理後の値 62 として「(0b0011)」を出力する。 20

【0124】

第 2 の 1 の補数絶対値演算器 21 には、入力 B (70) の値「-5 (0b1011)」が入力される。反転器 25 で全てのビットが反転されるので、符号反転後の値 61 は「(0b0100)」となる。セレクト 26 では、入力 B (70) の値「-5 (0b1011)」の符号を判断して、符号反転後の値 61 を選択して絶対値演算処理後の値 63 として「(0b0100)」を出力する。 30

【0125】

減算器 22 では、絶対値演算処理後の値 62 から絶対値演算処理後の値 63 を減算するので、絶対値比較演算の結果 65 として「(0b1111)」を出力する。

【0126】

フラグ設定部 9 では入力された値「(0b1111)」を判断して、N フラグレジスタ = 1、Z フラグレジスタ = 0 をセットする。

【0127】

ステップ (7) ~ (12) での動作は、従来の一般的なプロセッサの場合と同じである。繰返し 3 回目のステップ (3) は、従来の一般的なプロセッサの場合と同じである。

【0128】

ステップ (14) では、1 の補数絶対値減算器 13 を使った演算が行われる。この時の動作を図 24 を用いて説明する。 40

【0129】

レジスタ C の値「-5 (0b1011)」は入力 A (69) に、レジスタ A の値「-2 (0b1110)」は入力 B (70) にそれぞれ入力される。

【0130】

第 1 の 1 の補数絶対値演算器 20 には、入力 A (69) の値「-5 (0b1011)」が入力される。反転器 23 で全てのビットが反転されるので、符号反転後の値 60 は「(0b0100)」となる。セレクト 24 では、入力 A (69) の値「-5 (0b1011)」の符号を判断して、符号反転後の値 60 を選択して絶対値演算処理後の値 62 として「 50

「(0b0100)」を出力する。

【0131】

第2の1の補数絶対値演算器21には、入力B(70)の値「-2(0b1110)」が入力される。反転器25で全てのビットが反転されるので、符号反転後の値61は「(0b0001)」となる。セクタ26では、入力B(70)の値「-2(0b1110)」の符号を判断して、符号反転後の値61を選択して絶対値演算処理後の値63として「(0b0001)」を出力する。

【0132】

減算器22では、絶対値演算処理後の値62から絶対値演算処理後の値63を減算するので、絶対値比較演算の結果65として「(0b0011)」を出力する。

10

【0133】

フラグ設定部9では入力された値「(0b0011)」を判断して、Nフラグレジスタ=0、Zフラグレジスタ=0をセットする。

【0134】

ステップ(7)～(12)での動作は、従来一般的なプロセッサの場合と同じである。繰返し4回目のステップ(3)は、従来一般的なプロセッサの場合と同じである。

【0135】

ステップ(14)では、1の補数絶対値減算器13を使った演算が行われる。この時の動作を図25を用いて説明する。

【0136】

レジスタCの値「-5(0b1011)」は入力A(69)に、レジスタAの値「6(0b0110)」は入力B(70)にそれぞれ入力される。

20

【0137】

第1の1の補数絶対値演算器20には、入力A(69)の値「-5(0b1011)」が入力される。反転器23で全てのビットが反転されるので、符号反転後の値60は「(0b0100)」となる。セクタ24では、入力A(69)の値「-5(0b1011)」の符号を判断して、符号反転後の値60を選択して絶対値演算処理後の値62として「(0b0100)」を出力する。

【0138】

第2の1の補数絶対値演算器21には、入力B(70)の値「6(0b0110)」が入力される。反転器25で全てのビットが反転されるので、符号反転後の値61は「(0b1001)」となる。セクタ26では、入力B(70)の値「6(0b0110)」の符号を判断して、入力B(70)を選択して絶対値演算処理後の値63として「(0b0110)」を出力する。

30

【0139】

減算器22では、絶対値演算処理後の値62から絶対値演算処理後の値63を減算するので、絶対値比較演算の結果65として「(0b1110)」を出力する。

【0140】

フラグ設定部9では入力された値「(0b1110)」を判断して、Nフラグレジスタ=1、Zフラグレジスタ=0をセットする。

40

【0141】

ステップ(7)～(12)での動作は、従来一般的なプロセッサの場合と同じである。

【0142】

以上の処理を行うことにより、レジスタCに絶対値が最大である値「6」を求めることができる。このように、従来技術において符号の反転処理に用いていた2の補数表現化手順操作に換えて1の補数表現化手順操作を用いることによっても絶対値が最大の値を求めることができる。

【0143】

(動作例2)

第1の施例の形態のプロセッサで、レジスタAに「0(0b0000)」、レジスタCに

50

4ビットの2の補数表現で表現が可能な負の値で絶対値が最大である「 -8 (0b1000)」がセットされている場合に、「a b c m p C, A」ステップ(14)が実行される場合を図26を用いて説明する。レジスタAに「 0 (0b0000)」、レジスタCに「 -8 (0b1000)」というデータの設定は、前述の従来のプロセッサではオーバーフローが発生して正しい演算を行うことが不可能なデータケースである。

【0144】

レジスタCの値「 -8 (0b1000)」は入力A(69)に、レジスタAの値「 0 (0b0000)」は入力B(70)にそれぞれ入力される。

【0145】

第1の1の補数絶対値演算器20には、入力A(69)の値「 -8 (0b1000)」が 10
入力される。反転器23で全てのビットが反転されるので、符号反転後の値60は「(0b0111)」となる。セクタ24では、入力A(69)の値「 -8 (0b1000)」の符号を判断して、符号反転後の値60を選択して絶対値演算処理後の値62として「(0b0111)」を出力する。このように、従来のプロセッサと異なりオーバーフローが発生せず、正常に符号が反転される。

【0146】

第2の1の補数絶対値演算器21には、入力B(70)の値「 0 (0b0000)」が入 20
力される。反転器25で全てのビットが反転されるので、符号反転後の値61は「(0b1111)」となる。セクタ26では、入力B(70)の値「 0 (0b0000)」の符号を判断して、入力B(70)を選択して絶対値演算処理後の値63として「(0b0000)」を出力する。

【0147】

減算器22では、絶対値演算処理後の値62から絶対値演算処理後の値63を減算するので、絶対値比較演算の結果65として「(0b0111)」を出力する。すなわち、 $|-8| - |0|$ の減算結果が正となり、 $|-8|$ と $|0|$ との大小比較が正しく行われる。

【0148】

フラグ設定部9では入力された値「(0b0111)」を判断して、Nフラグレジスタ=0、Zフラグレジスタ=0をセットする。

【0149】

ステップ(7)では、Nフラグレジスタ=0、Zフラグレジスタ=0なので、ステップ(30
9)に進み、レジスタAの値はレジスタCにセットされない。

【0150】

このように第1の実施の形態のプロセッサにおいては「 -8 (0b1000)」と「 0 (0b0000)」との絶対値の大小比較において、絶対値が大きい「 -8 (0b1000)」を正しく出力することができる。

【0151】

(動作例3)

第1の実施の形態のプロセッサで、レジスタAに「 -4 (0b1100)」、レジスタCに「 3 (0b0011)」がセットされている場合に、ステップ(14)「a b c m p C, A」が実行される場合を説明する。 40

【0152】

レジスタCの値「 3 (0b0011)」は入力A(69)に、レジスタAの値「 -4 (0b1100)」は入力B(70)にそれぞれ入力される。

【0153】

第1の1の補数絶対値演算器20には、入力A(69)の値「 3 (0b0011)」が入 50
力される。反転器23で全てのビットが反転されるので、符号反転後の値60は「(0b1100)」となる。セクタ24では、入力A(69)の値「 3 (0b0011)」の符号を判断して、入力A(69)を選択して絶対値演算処理後の値62として「(0b0011)」を出力する。

【0154】

第2の1の補数絶対値演算器21には、入力B(70)の値「-4(0b1100)」が入力される。反転器25で全てのビットが反転されるので、符号反転後の値61は「(0b0011)」となる。セレクト26では、入力B(70)の値「-4(0b1100)」の符号を判断して、符号反転後の値61を選択して絶対値演算処理後の値63として「(0b0011)」を出力する。

【0155】

減算器22では、絶対値演算処理後の値62から絶対値演算処理後の値63を減算するので、絶対値比較演算の結果65として「(0b0000)」を出力する。

【0156】

フラグ設定部9では入力された値「(0b0000)」を判断して、Nフラグレジスタ= 10
0、Zフラグレジスタ=1をセットする。

【0157】

ステップ(7)では、Nフラグレジスタ=0、Zフラグレジスタ=1なので、ステップ(9)に進み、レジスタAの値「-4(0b1100)」はレジスタCにセットされず、レジスタCの値は絶対値が小さい「3(0b0011)」のままとなる。

【0158】

このように第1の実施の形態のプロセッサでは、ステップ(14)「abc mp C, A」が実行される場合に、レジスタAの値が負の値(例えば-4)で、レジスタBの値がレジスタAの値の絶対値の1小さい数の正の値(例えば3)の場合、減算結果が0となり正しく絶対値の大小判定が行われない。そのため、本来求めるべき絶対値が最大の値と比べ 20
1小さい値が、最大値として求められる場合があり得る。しかし、音声データは通常16ビット幅で扱われ、16ビットで表現しうる値の範囲は-32768~32767である。そのため、たとえ求める絶対値が最大の値が1違う場合にも、16ビットのフルレンジから見ると誤差は0.003%程度であり、無視することができる誤差である。

【0159】

以上の動作例に示したように第1の実施の形態のプロセッサでは、無視できるほどの誤差で絶対値が最大の値を求めることができ、2の補数表現で表現が可能な負の値で絶対値が最大である値が入力されても正しく演算することができる。また、1の補数絶対値減算器13には、+1加算器が使用されていないので、1の補数絶対値減算器13の小型化、引いてはプロセッサの小型化を図ることができる。さらに、従来は1度の絶対値比較演算処理において2度行われていた+1加算の処理を行わないので、複数のデータから絶対値が 30
最大となる値を求めるための時間を短縮することができる。

【0160】

(第2の実施の形態)

第2の実施の形態のプロセッサは、第1の実施の形態のプロセッサに用いた1の補数絶対値減算器13の構成を、図2に示すものに変えて、図3に示すような構成としたものである。プロセッサの他の部分の構成については図1に示す第1の実施の形態の構成と同じである。

【0161】

図3に示す1の補数絶対値減算器13は、本発明に係る演算器である。図3に示す1の補数絶対値減算器13は、1の補数絶対値演算手段としての第1の1の補数絶対値演算器20と、第2の1の補数絶対値演算手段および反転手段としての1の補数絶対値反転演算器27と、指示情報出力手段としての加算器28を備える。 40

【0162】

1の補数絶対値減算器13は、制御信号10により指定されるレジスタファイル7内の2つのレジスタのそれぞれが保持する2つの値について1の補数表現を用いた絶対値の比較演算を行い、その比較結果を2つの値の絶対値のうちのどちらが大きいかを表す指示情報として出力する。

【0163】

第1の1の補数絶対値演算器20は、第1の実施の形態の第1の1の補数絶対値演算器2 50

0と同じ構成である。1の補数絶対値反転演算器27は、入力B(70)の符号が負の場合には入力B(70)に対して1の補数表現化の手順操作を施す。さらに、入力B(70)もしくは入力B(70)を1の補数表現化したものの全てのビットを反転して絶対値反転処理後の値64として出力する。

【0164】

1の補数絶対値反転演算器27は、反転器29、反転器31とセレクタ30を備える。反転器29は、入力された全てのビットを反転して、符号反転後の値61を出力する。セレクタ30は、入力B(70)の符号が負の場合は、符号反転後の値61を選択し、それ以外の場合は入力B(70)を選択して、絶対値演算処理後の値63を出力する。反転器31は、絶対値演算処理後の値63の全てのビットを反転して、絶対値反転処理後の値64

10

【0165】

加算器28は、絶対値演算処理後の値62から絶対値反転処理後の値64に「1」を加算して、絶対値比較演算の結果(指示情報)65を出力する。

【0166】

つまり、第2の実施の形態のプロセッサでは、第1の実施の形態のプロセッサのように2つの値の絶対値の減算処理により絶対値の大小比較を行うのではなく、減算値の全てのビットを反転したものを被減算値に加え、これにさらに「1」を加えるという加算処理によって絶対値の大小比較を実現している。

【0167】

20

第2の実施の形態のプロセッサは、図1に示す第1の実施の形態のプロセッサと同じ実行命令を備える。ただし、「abc mp」命令による処理を実行する演算器は図3に示す1の補数絶対値減算器13である

次に、第2の実施の形態によるプロセッサの動作を説明する。

【0168】

(動作例1)

第2の実施の形態のプロセッサで、図16に示す4つのデータから絶対値が最大であるデータを求める動作を、図3に示す1の補数絶対値減算器13で実行される「abc mp」命令を中心に説明する。

【0169】

30

第2の実施の形態のプロセッサでの絶対値が最大のデータを求める処理のプログラムは、第1の実施の形態に示したプロセッサと同じく図8に示すプログラムである。

【0170】

ステップ(1)、(2)での動作は、従来の一般的なプロセッサの場合と同じである。

【0171】

繰返し1回目のステップ(3)での動作は、従来の一般的なプロセッサの場合と同じである。

【0172】

ステップ(14)では、1の補数絶対値減算器13を使った演算が行われる。この時の動作を図28を用いて説明する。

40

【0173】

レジスタCの値「0(0b0000)」は入力A(69)に、レジスタAの値「3(0b0011)」は入力B(70)にそれぞれ入力される。

【0174】

第1の1の補数絶対値演算器20には、入力A(69)の値「0(0b0000)」が入力される。反転器23で全てのビットが反転されるので、符号反転後の値60は「(0b1111)」となる。セレクタ24では、入力A(69)の値「0(0b0000)」の符号を判断して、入力A(69)を選択して絶対値演算処理後の値62として「(0b0000)」を出力する。

【0175】

50

1の補数絶対値反転演算器27には、入力B(70)の値「3(0b0011)」が入力される。反転器29で全てのビットが反転されるので、符号反転後の値61は「(0b1100)」となる。セクタ26では、入力B(70)の値「3(0b0011)」の符号を判断して、入力B(70)を選択して絶対値演算処理後の値63として「(0b0011)」を出力する。反転器31では、絶対値演算処理後の値63をビット反転し、絶対値反転処理後の値64として「(0b1100)」を出力する。

【0176】

加算器28では、絶対値演算処理後の値62と絶対値反転処理後の値64と「1」を加算するので、絶対値比較演算の結果65として「(0b1101)」を出力する。

【0177】

フラグ設定部9では入力された値「(0b1101)」を判断して、Nフラグレジスタ=1、Zフラグレジスタ=0をセットする。

【0178】

ステップ(7)～(12)での動作は、従来の一般的なプロセッサの場合と同じである。

【0179】

繰返し2回目のステップ(3)は、従来の一般的なプロセッサの場合と同じである。

【0180】

ステップ(14)では、1の補数絶対値減算器13を使った演算が行われる。この時の動作を図29を用いて説明する。

【0181】

レジスタCの値「3(0b0011)」は入力A(69)に、レジスタAの値「-5(0b1011)」は入力B(70)にそれぞれ入力される。

【0182】

第1の1の補数絶対値演算器20には、入力A(69)の値「3(0b0011)」が入力される。反転器23で全てのビットが反転されるので、符号反転後の値60は「(0b1100)」となる。セクタ24では、入力A(69)の値「3(0b0011)」の符号を判断して、入力A(69)を選択して絶対値演算処理後の値62として「(0b0011)」を出力する。

【0183】

1の補数絶対値反転演算器27には、入力B(70)の値「-5(0b1011)」が入力される。反転器25で全てのビットが反転されるので、符号反転後の値61は「(0b0100)」となる。セクタ26では、入力B(70)の値「-5(0b1011)」の符号を判断して、符号反転後の値61を選択して絶対値演算処理後の値63として「(0b0100)」を出力する。反転器31では、絶対値演算処理後の値63をビット反転し、絶対値反転処理後の値64として「(0b1011)」を出力する。

【0184】

加算器28では、絶対値演算処理後の値62と絶対値反転処理後の値64と「1」を加算するので、絶対値比較演算の結果65として「(0b1111)」を出力する。

【0185】

フラグ設定部9では入力された値「(0b1110)」を判断して、Nフラグレジスタ=1、Zフラグレジスタ=0をセットする。

【0186】

ステップ(7)～(12)での動作は、従来の一般的なプロセッサの場合と同じである。

繰返し3回目のステップ(3)は、従来の一般的なプロセッサの場合と同じである。

【0187】

ステップ(14)では、1の補数絶対値減算器13を使った演算が行われる。この時の動作を図30を用いて説明する。

【0188】

レジスタCの値「-5(0b1011)」は入力A(69)に、レジスタAの値「-2(0b1110)」は入力B(70)にそれぞれ入力される。

10

20

30

40

50

【0189】

第1の1の補数絶対値演算器20には、入力A(69)の値「-5(0b1011)」が
入力される。反転器23で全てのビットが反転されるので、符号反転後の値60は「(0
b0100)」となる。セレクタ24では、入力A(69)の値「-5(0b1011)
」の符号を判断して、符号反転後の値60を選択して絶対値演算処理後の値62として「
(0b0100)」を出力する。

【0190】

1の補数絶対値反転演算器27には、入力B(70)の値「-2(0b1110)」が入
力される。反転器25で全てのビットが反転されるので、符号反転後の値61は「(0b
0001)」となる。セレクタ26では、入力B(70)の値「-2(0b1110)」
の符号を判断して、符号反転後の値61を選択して絶対値演算処理後の値63として「(0
b0001)」を出力する。反転器31では、絶対値演算処理後の値63をビット反転
し、絶対値反転処理後の値64として「(0b1110)」を出力する。

10

【0191】

加算器28では、絶対値演算処理後の値62と絶対値反転処理後の値64と「1」を加算
するので、絶対値比較演算の結果65として「(0b0011)」を出力する。

【0192】

フラグ設定部9では入力された値「(0b0011)」を判断して、Nフラグレジスタ=
0、Zフラグレジスタ=0をセットする。

【0193】

ステップ(7)～(12)での動作は、従来の一般的なプロセッサの場合と同じである。
繰返し4回目のステップ(3)は、従来の一般的なプロセッサの場合と同じである。

20

【0194】

ステップ(14)では、1の補数絶対値減算器13を使った演算が行われる。この時の動
作を図31を用いて説明する。

【0195】

レジスタCの値「-5(0b1011)」は入力A(69)に、レジスタAの値「6(0
b0110)」は入力B(70)にそれぞれ入力される。

【0196】

第1の1の補数絶対値演算器20には、入力A(69)の値「-5(0b1011)」が
入力される。反転器23で全てのビットが反転されるので、符号反転後の値60は「(0
b0100)」となる。セレクタ24では、入力A(69)の値「-5(0b1011)
」の符号を判断して、符号反転後の値60を選択して絶対値演算処理後の値62として「
(0b0100)」を出力する。

30

【0197】

1の補数絶対値反転演算器27には、入力B(70)の値「6(0b0110)」が入
力される。反転器25で全てのビットが反転されるので、符号反転後の値61は「(0b1
001)」となる。セレクタ26では、入力B(70)の値「6(0b0110)」の符
号を判断して、入力B(70)を選択して絶対値演算処理後の値63として「(0b01
10)」を出力する。反転器31では、絶対値演算処理後の値63をビット反転し、絶対
値反転処理後の値64として「(0b1001)」を出力する。

40

【0198】

加算器28では、絶対値演算処理後の値62と絶対値反転処理後の値64と「1」を加算
するので、絶対値比較演算の結果65として「(0b1110)」を出力する。

【0199】

フラグ設定部9では入力された値「(0b1110)」を判断して、Nフラグレジスタ=
1、Zフラグレジスタ=0をセットする。

【0200】

ステップ(7)～(12)での動作は、従来の一般的なプロセッサの場合と同じである。

【0201】

50

以上の処理を行うことにより、減算処理に換えて加算処理によってレジスタCに絶対値が最大である値「6」を求めることができる。

【0202】

(動作例2)

第2の実施の形態のプロセッサで、レジスタAに「0 (0b0000)」、レジスタCに4ビットの2の補数表現で表現が可能な範囲で絶対値が最大の負の値「-8 (0b1000)」がセットされている場合に、「a b c m p C, A」ステップ(14)が実行される場合を図32を用いて説明する。レジスタAに「0 (0b0000)」、レジスタCに「-8 (0b1000)」というデータの設定は、前述の従来のプロセッサではオーバーフローを起こして正しい演算を行うことが不可能なデータケースである。

10

【0203】

レジスタCの値「-8 (0b1000)」は入力A(69)に、レジスタAの値「0 (0b0000)」は入力B(70)にそれぞれ入力される。

【0204】

第1の1の補数絶対値演算器20には、入力A(69)の値「-8 (0b1000)」が入力される。反転器23で全てのビットが反転されるので、符号反転後の値60は「(0b0111)」となる。セクタ24では、入力A(69)の値「-8 (0b1000)」の符号を判断して、符号反転後の値60を選択して絶対値演算処理後の値62として「(0b0111)」を出力する。オーバーフローを起こさず正常に符号が反転されている。

【0205】

20

1の補数絶対値反転演算器27には、入力B(70)の値「0 (0b0000)」が入力される。反転器25で全てのビットが反転されるので、符号反転後の値61は「(0b1111)」となる。セクタ26では、入力B(70)の値「0 (0b0000)」の符号を判断して、入力B(70)を選択して絶対値演算処理後の値63として「(0b0000)」を出力する。反転器31では、絶対値演算処理後の値63をビット反転し、絶対値反転処理後の値64として「(0b1111)」を出力する。

【0206】

加算器28では、絶対値演算処理後の値62と絶対値反転処理後の値64と「1」を加算するので、絶対値比較演算の結果65として「(0b0111)」を出力する。つまり、 $|-8| - |0|$ の演算結果が正となり、絶対値の比較が正常に行われる。

30

【0207】

フラグ設定部9では入力された値「(0b0111)」を判断して、Nフラグレジスタ=0、Zフラグレジスタ=0をセットする。

【0208】

ステップ(7)では、Nフラグレジスタ=0、Zフラグレジスタ=0なので、ステップ(9)に進み、レジスタAの値はレジスタCにセットされない。

【0209】

このように第2の実施の形態のプロセッサにおいては「-8 (0b1000)」と「0 (0b0000)」では、絶対値が大きい「-8 (0b1000)」を絶対値の最大値として出力することができる。

40

【0210】

(動作例3)

第2の実施の形態のプロセッサで、レジスタAに「-4 (0b1100)」、レジスタCに「3 (0b0011)」がセットされている場合に、ステップ(14)「a b c m p C, A」が実行される場合を図33を用いて説明する。

【0211】

レジスタCの値「3 (0b0011)」は入力A(69)に、レジスタAの値「-4 (0b1100)」は入力B(70)にそれぞれ入力される。

【0212】

第1の1の補数絶対値演算器20には、入力A(69)の値「3 (0b0011)」が入

50

力される。反転器 23 で全てのビットが反転されるので、符号反転後の値 60 は「(0b1100)」となる。セレクタ 24 では、入力 A (69) の値「3 (0b0011)」の符号を判断して、入力 A (69) を選択して絶対値演算処理後の値 62 として「(0b0011)」を出力する。

【0213】

1 の補数絶対値反転演算器 27 には、入力 B (70) の値「-4 (0b1100)」が入力される。反転器 25 で全てのビットが反転されるので、符号反転後の値 61 は「(0b0011)」となる。セレクタ 26 では、入力 B (70) の値「-4 (0b1100)」の符号を判断して、符号反転後の値 61 を選択して絶対値演算処理後の値 63 として「(0b0011)」を出力する。反転器 31 では、絶対値演算処理後の値 63 をビット反転し、絶対値反転処理後の値 64 として「(0b1100)」を出力する。

10

【0214】

加算器 28 では、絶対値演算処理後の値 62 と絶対値反転処理後の値 64 と「1」を加算するので、絶対値比較演算の結果 65 として「(0b0000)」を出力する。

【0215】

フラグ設定部 9 では入力された値「(0b0000)」を判断して、N フラグレジスタ = 0、Z フラグレジスタ = 1 をセットする。

【0216】

ステップ (7) では、N フラグレジスタ = 0、Z フラグレジスタ = 1 なので、ステップ (9) に進み、レジスタ A の値「-4 (0b1100)」はレジスタ C にセットされず、レジスタ C の値は絶対値が小さい「3 (0b0011)」のままとなる。

20

【0217】

このように第 2 の実施の形態のプロセッサでは、ステップ (14) 「abc mp C, A」が実行される場合に、レジスタ A の値が負 (例えば -4) で、レジスタ B の値がレジスタ A の値の絶対値より 1 小さい数の正の値 (例えば 3) の場合に、正しく絶対値の大小比較が行われない。そのため、本来選ばれるべき絶対値が最大の値に比べ 1 小さい値が最大の絶対値をもつ値として求められる場合がある。しかし、音声データは通常 16 ビットで扱われ、2 の補数表現で表現できる範囲は -32768 ~ 32767 である。そのため、たとえ絶対値が 1 違っていたとしても 16 ビットのフルレンジから見ると誤差は 0.003 % 程度であり、無視することができる誤差である。

30

【0218】

以上の動作例に示したように第 2 の実施の形態のプロセッサでは、無視できるほどの誤差で絶対値が最大の値を求めることができ、所定のビットの 2 の補数表現で表現が可能な負の値で絶対値が最大である値も正しく演算することができる。また、1 の補数絶対値減算器 13 には、使用されている +1 加算機は 1 つなので、1 の補数絶対値減算器 13 の小型化、引いてはプロセッサの小型化を図ることができる。さらに、従来は 1 組の絶対値比較演算処理において 2 度行われていた +1 加算の処理が 1 度で済むので、複数のデータから絶対値が最大となる値を求めるための時間を短縮することができる。

【0219】

(第 3 の実施の形態)

40

第 3 の実施の形態は、本発明に関わる演算器および演算方法およびプロセッサにおける技術をプロセッサに適用したものであり、絶対値の大小比較および大小比較に基づく入力値の 2 者択一までの処理をハード化すると共に、これを実行する実行命令を備えている。

【0220】

図 4 は、第 3 の実施の形態のプロセッサを示す。図 4 に示すプロセッサは、プロセッサの制御を行う制御部 1 と演算を行うデータバス部 2 とを備えるものである。

【0221】

制御部 1 は、プログラムカウンタ 3、実行命令メモリ 4、デコーダ 5 により構成される。プログラムカウンタ 3、実行命令メモリ 4、デコーダ 5 は、図 7 に示した従来のプロセッサと同じ構成なので、説明を省略する。

50

【0222】

データバス部2は、ALU6、絶対値最大値演算器15、レジスタファイル7、データメモリ8、フラグ設定部9により構成される。ALU6、レジスタファイル7、データメモリ8、フラグ設定部9は、図7に示した従来の一般的なプロセッサと同じ構成なので、説明を省略する。

【0223】

本発明に係る演算器である絶対値最大値演算器15は、制御信号10により指定されたレジスタファイル7内の2つのレジスタに格納された値を入力とする。この2つの値について1の補数表現を用いて絶対値の比較演算を行うことにより、2つの値の絶対値のうちのどちらが大きいかを表す指示情報を求める。さらに、この指示情報に基づいて、2の値のうちの絶対値が大きい方の値を出力する。出力された値は、制御信号10により指定されたレジスタファイル7内のレジスタに書き込まれる。

10

【0224】

図5は、上記の絶対値最大値演算器15の構成を示した図である。図5に示すように、絶対値最大値演算器15は、第1の入力もしくは第1の入力を1の補数表現化した絶対値から第2の入力もしくは第2の入力を1の補数表現化した絶対値を減算する、第1の1の補数絶対値演算手段、第2の1の補数絶対値演算手段および指示情報出力手段としての1の補数絶対値減算器32と、選択手段としてのセレクタ33を備える。1の補数絶対値減算器32は、入力A(69)の値もしくは入力A(69)を1の補数表現化した絶対値から、入力B(70)の値もしくは入力B(70)を1の補数表現化した絶対値を減算し、その減算結果を絶対値比較演算の結果(指示情報)65として出力する。1の補数絶対値減算器32の構成は、第1の実施の形態または第2の実施の形態における1の補数絶対値減算器13と同じ構成なので説明を省略する。セレクタ33では、絶対値比較演算の結果65の符号が負の場合は、入力B(70)を選択し、負以外の場合は入力A(69)を選択して、選択した値を最大絶対値選択演算の結果68として出力する。

20

【0225】

図4のプロセッサも図7の従来の一般的なプロセッサと同じく、A、B、C、D、Pの5つのレジスタのビット幅、データメモリ8のビット幅、ALU6の入出力のビット幅はすべて4ビットとする。また、絶対値最大値演算器15の入出力、演算精度のビット幅も4ビットとする。

30

【0226】

第3の実施の形態のプロセッサは、図7のプロセッサと同じ実行命令に加え、絶対値最大値演算器15で実行される以下の実行命令を備える。

【0227】

「a b m a x a, b : レジスタaとレジスタbとのそれぞれに保持されている値の絶対値を比較し、絶対値が大きい方の値をレジスタaにセット」

次に、第3の実施の形態によるプロセッサの動作を説明する。

【0228】

(動作例1)

第3の実施の形態のプロセッサで、図16に示す4つのデータの中から絶対値が最大であるデータを求める動作を、絶対値最大値演算器15で処理が実行される「a b m a x」命令を中心に説明する。

40

【0229】

絶対値最大値演算器15を備えたプロセッサで絶対値が最大の値を求めるためのプログラムを図15に示す。ステップ(1)からステップ(3)、ステップ(9)からステップ(13)までは従来の一般的なプロセッサの場合と同じである。ステップ(15)は、「a b m a x」命令を使ってレジスタCの値とレジスタAの値の絶対値を比較し、絶対値が大きい方の値をレジスタCにセットするステップである。ステップ(15)では、従来の一般的なプロセッサで行う場合のステップ(4)からステップ(8)までの処理と同等の処理を行う。絶対値最大値演算器15の入力A(69)にはレジスタCの値、入力B(70)

50

）にはレジスタ A の値が入力される。

【0230】

ステップ（1）、（2）での動作は、従来の一般的なプロセッサの場合と同じである。

【0231】

繰返し1回目のステップ（3）での動作は、従来の一般的なプロセッサの場合と同じである。

【0232】

ステップ（15）では、絶対値最大値演算器15を使った演算が行われる。この時の動作を図34を用いて説明する。

【0233】

レジスタ C の値「0（0b0000）」は入力 A（69）に、レジスタ A の値「3（0b0011）」は入力 B（70）にそれぞれ入力される。

【0234】

第1の実施の形態の動作例1より、1の補数絶対値減算器32の出力である絶対値比較演算の結果65は「（0b1101）」となる。絶対値比較演算の結果65の符号は負なので、セクタ33では入力 B（70）を選択し、最大絶対値選択演算の結果68として「3（0b0011）」を出力する。そして、レジスタ C には「3（0b011）」をセットする。

【0235】

ステップ（9）～（12）での動作は、従来の一般的なプロセッサの場合と同じである。

【0236】

繰返し2回目のステップ（3）は、従来の一般的なプロセッサの場合と同じである。

【0237】

ステップ（15）では、絶対値最大値演算器15を使った演算が行われる。この時の動作を図35を用いて説明する。

【0238】

レジスタ C の値「3（0b0011）」は入力 A（69）に、レジスタ A の値「-5（0b1011）」は入力 B（70）にそれぞれ入力される。

【0239】

第1の実施の形態の動作例1より、1の補数絶対値減算器32の出力である絶対値比較演算の結果65は「（0b1111）」となる。絶対値比較演算の結果65の符号は負なので、セクタ33では入力 B（70）を選択し、最大絶対値選択演算の結果68として「-5（0b1011）」を出力する。そして、レジスタ C には「-5（0b1011）」をセットする。

【0240】

ステップ（9）～（12）での動作は、従来の一般的なプロセッサの場合と同じである。

繰返し3回目のステップ（3）は、従来の一般的なプロセッサの場合と同じである。

【0241】

ステップ（15）では、絶対値最大値演算器15を使った演算が行われる。この時の動作を図36を用いて説明する。

【0242】

レジスタ C の値「-5（0b1011）」は入力 A（69）に、レジスタ A の値「-2（0b1110）」は入力 B（70）にそれぞれ入力される。

【0243】

第1の実施の形態の動作例1より、1の補数絶対値減算器32の出力である絶対値比較演算の結果65は「（0b0011）」となる。絶対値比較演算の結果65の符号は負ではないので、セクタ33では入力 A（69）を選択し、最大絶対値選択演算の結果68として「-5（0b1011）」を出力する。そして、レジスタ C には「-5（0b1011）」をセットする。

【0244】

10

20

30

40

50

ステップ（９）～（１２）での動作は、従来の一般的なプロセッサの場合と同じである。
繰返し４回目のステップ（３）は、従来の一般的なプロセッサの場合と同じである。

【０２４５】

ステップ（１５）では、絶対値最大値演算器１５を使った演算が行われる。この時の動作を図３７を用いて説明する。

【０２４６】

レジスタＣの値「－５（０ｂ１０１１）」は入力Ａ（６９）に、レジスタＡの値「６（０ｂ０１１０）」は入力Ｂ（７０）にそれぞれ入力される。

【０２４７】

第１の実施の形態の動作例１より、１の補数絶対値減算器３２の出力である絶対値比較演算の結果６５は「（０ｂ１１１０）」となる。絶対値比較演算の結果６５の符号は負なので、セクタ３３では入力Ｂ（７０）を選択し、最大絶対値選択演算の結果６８として「６（０ｂ０１１０）」を出力する。そして、レジスタＣには「６（０ｂ０１１０）」をセットする。 10

【０２４８】

ステップ（７）～（１１）での動作は、従来の一般的なプロセッサの場合と同じである。

【０２４９】

以上の処理を行うことにより、レジスタＣに絶対値が最大である値「６」を求めることができる。

【０２５０】

20

（動作例２）

本実施例の形態３のプロセッサで、レジスタＡに「０（０ｂ００００）」、レジスタＣに４ビットの２の補数表現で表現が可能な負の値で絶対値が最大である「－８（０ｂ１０００）」がセットされている場合に、「a b m a x C, A」ステップ（１５）が実行される場合を図３８を用いて説明する。これは、従来のプロセッサでは正しく絶対値の比較演算を行うことができない場合である。

【０２５１】

レジスタＣの値「－８（０ｂ１０００）」は入力Ａ（６９）に、レジスタＡの値「０（０ｂ００００）」は入力Ｂ（７０）にそれぞれ入力される。

【０２５２】

30

第１の実施の形態の動作例２より、１の補数絶対値減算器３２の出力である絶対値比較演算の結果６５は「（０ｂ０１１１）」となる。絶対値比較演算の結果６５の符号は負でないので、セクタ３３では入力Ａ（６９）を選択し、最大絶対値選択演算の結果６８として「－８（０ｂ１０００）」を出力する。そして、レジスタＣには「－８（０ｂ１０００）」をセットする。

【０２５３】

このように第３の実施の形態のプロセッサにおいては「－８（０ｂ１０００）」と「０（０ｂ００００）」との絶対値比較において、「－８（０ｂ１０００）」を絶対値が大きい値として正しく出力することができる。

【０２５４】

40

（動作例３）

第３の実施の形態のプロセッサで、レジスタＡに「－４（０ｂ１１００）」、レジスタＣに「３（０ｂ００１１）」がセットされている場合に、「a b m a x C, A」ステップ（１５）が実行される場合を図３９を用いて説明する。

【０２５５】

レジスタＣの値「３（０ｂ００１１）」は入力Ａ（６９）に、レジスタＡの値「－４（０ｂ１１００）」は入力Ｂ（７０）にそれぞれ入力される。

【０２５６】

第１の実施の形態の動作例３より、１の補数絶対値減算器３２の出力である絶対値比較演算の結果６５は「（０ｂ００００）」となる。絶対値比較演算の結果６５の符号は負でな 50

いので、セクタ 33 では入力 A (69) を選択し、最大絶対値選択演算の結果 68 として「3 (0b0011)」を出力する。そして、レジスタ C には「3 (0b0011)」をセットする。

【0257】

このように第 3 の実施の形態のプロセッサでは、「a b c m p C, A」ステップ (15) が実行される場合に、レジスタ A の値が負 (例えば -4) で、レジスタ B の値がレジスタ A の値の絶対値の 1 小さい数の正の値 (例えば 3) の場合、正しい絶対値の大小比較が行われない。そのため、本来求められるべき絶対値が最大の値に比べ 1 小さい値が絶対値が最大の値として求められる場合がある。しかし、音声データは通常 16 ビットで扱われ、16 ビットで表現しうる 2 の補数表現の値の範囲は -32768 ~ 32767 である。そのため、たとえ絶対値が 1 違っていたとしても 16 ビットのフルレンジから見ると誤差は 0.003 % 程度であり、無視することができる誤差である。

10

【0258】

以上の動作例に示したように第 3 の実施の形態のプロセッサでは、無視できるほどの誤差で絶対値が最大の値を求めることができ、4 ビットの 2 の補数表現で表現が可能な負の値で絶対値が最大である値も正しく演算することができる。また、図 8 に示した従来のプロセッサのプログラムにおけるステップ (7) ~ ステップ (9) をハードで実現しているので、さらに高速に絶対値が最大の値を求めることができる。

【0259】

(第 4 の実施の形態)

20

第 4 の実施の形態のプロセッサは、図 5 に示した第 3 の実施の形態での絶対値最大値演算器 15 に変えて、図 6 に示すような構成の絶対値最大値演算器 15 を備えたものである。なお、他部分の構成は図 4 に示す第 3 の実施の形態の構成と同じである。

【0260】

図 6 に示す絶対値最大値演算器 15 は、本発明に係る演算器である。図 6 に示すように絶対値最大値演算器 15 は、第 1 の入力もしくは第 1 の入力を 1 の補数表現化した絶対値から第 2 の入力もしくは第 2 の入力を 1 の補数表現化した絶対値を減算する、第 1 の 1 の補数絶対値演算手段、第 2 の 1 の補数絶対値演算手段および指示情報出力手段としての 1 の補数絶対値減算器 32 と、ゼロ値判断手段としてのゼロ検出器 34 と、第 1 の入力または第 2 の入力のどちらかを選択して出力する選択手段 35 とを備える。

30

【0261】

絶対値最大値演算器 15 は、制御信号 10 により指定されたレジスタファイル 7 内の 2 つのレジスタに格納された値を入力とする。この 2 つの値について 1 の補数表現を用いて絶対値の比較演算を行うことにより、2 つの値の絶対値のうちのどちらが大きいかを表す指示情報を求める。さらに、この指示情報に基づいて、2 の値のうちの絶対値が大きい方の値を出力する。出力された値は、制御信号 10 により指定されたレジスタファイル 7 内のレジスタに書き込まれる。

【0262】

1 の補数絶対値減算器 32 は、入力 A (69) の値もしくは入力 A (69) を 1 の補数表現化した絶対値から、入力 B (70) の値もしくは入力 B (70) を 1 の補数表現化した絶対値を減算し、その結果を絶対値比較演算の結果 (指示情報) 65 として出力するものである。1 の補数絶対値減算器 32 の構成は、第 1 の実施の形態および第 2 の実施の形態における 1 の補数絶対値減算器 13 と同じ構成なので説明を省略する。

40

【0263】

ゼロ検出器 34 では、絶対値比較演算の結果 65 の値がゼロ (4 ビットの全てのビットがゼロ) 「0b0000」の場合に、ゼロ値判断結果 66 として「1」が出力され、それ以外の場合は「0」を出力する。

【0264】

出力値選択手段 35 では、選択信号生成手段 36 とセクタ 37 を備える。選択信号生成手段 36 は、絶対値比較演算の結果 65 の符号が負であるか、または、ゼロ値判断結果 6

50

6が「1」かつ入力B(70)の符号が負である場合に、選択信号67として「1」が出力され、それ以外の場合は「0」を出力する。セレクタ37では、選択信号67が「0」の場合に入力A(69)を選択し、選択信号67が「1」の場合に入力B(70)を選択し、最大絶対値選択演算の結果68として出力する。

【0265】

第4の実施の形態のプロセッサは、第3の実施の形態のプロセッサと同じ実行命令を備える。ただし、「abmax」命令による処理を行うのは演算器は図6に示す絶対値最大値演算器15である。

【0266】

次に、第4の実施の形態によるプロセッサの動作を説明する。

10

【0267】

(動作例1)

第4の実施の形態のプロセッサで、図16に示す4つのデータの中から絶対値が最大の値を求める動作を、絶対値最大値演算器15を使う「abmax」命令を中心に説明する。

【0268】

第4の実施の形態で使用するプログラムは第3の実施の形態に示した、図15と同じものである。

【0269】

繰返し1回目のステップ(3)での動作は、従来の一般的なプロセッサの場合と同じである。

20

【0270】

ステップ(15)では、絶対値最大値演算器15を使った演算が行われる。この時の動作を図40を用いて説明する。

【0271】

レジスタCの値「0(0b0000)」は入力A(69)に、レジスタAの値「3(0b00011)」は入力B(70)にそれぞれ入力される。

【0272】

第1の実施の形態の動作例1より、1の補数絶対値減算器32の出力である絶対値比較演算の結果65は「(0b1101)」となる。絶対値比較演算の結果65は「(0b0000)」ではないので、ゼロ値判断結果66は「0」である。選択信号生成手段36では、絶対値比較演算の結果65が負なので、選択信号67として「1」を出力する。セレクタ37では、選択信号67が「1」なので、入力B(70)を選択し、最大絶対値選択演算の結果68として、「3(0b0011)」を出力する。そして、レジスタCには「3(0b011)」をセットする。

30

【0273】

ステップ(9)～(12)での動作は、従来の一般的なプロセッサの場合と同じである。

【0274】

繰返し2回目のステップ(3)は、従来の一般的なプロセッサの場合と同じである。

【0275】

ステップ(15)では、絶対値最大値演算器15を使った演算が行われる。この時の動作を図41を用いて説明する。

40

【0276】

レジスタCの値「3(0b0011)」は入力A(69)に、レジスタAの値「-5(0b1011)」は入力B(70)にそれぞれ入力される。

【0277】

第1の実施の形態の動作例1より、1の補数絶対値減算器32の出力である絶対値比較演算の結果65は「(0b1111)」となる。絶対値比較演算の結果65は「(0b0000)」ではないので、ゼロ値判断結果66は「0」である。選択信号生成手段36では、絶対値比較演算の結果65が負なので、選択信号67として「1」を出力する。セレクタ37では、選択信号67が「1」なので、入力B(70)を選択し、最大絶対値選択演

50

算の結果 68 として、「-5 (0b1011)」を出力する。そして、レジスタ C には「-5 (0b1011)」をセットする。

【0278】

ステップ (9) ~ (12) での動作は、従来の一般的なプロセッサの場合と同じである。

【0279】

繰返し 3 回目のステップ (3) は、従来の一般的なプロセッサの場合と同じである。

【0280】

ステップ (15) では、絶対値最大値演算器 15 を使った演算が行われる。この時の動作を図 42 を用いて説明する。

【0281】

レジスタ C の値「-5 (0b1011)」は入力 A (69) に、レジスタ A の値「-2 (0b1110)」は入力 B (70) にそれぞれ入力される。

【0282】

第 1 の実施の形態の動作例 1 より、1 の補数絶対値減算器 32 の出力である絶対値比較演算の結果 65 は「(0b0011)」となる。絶対値比較演算の結果 65 は「(0b0000)」ではないので、ゼロ値判断結果 66 は「0」である。選択信号生成手段 36 では、絶対値比較演算の結果 65 が負でなく、ゼロ値判断結果 66 は「0」なので、選択信号 67 として「0」を出力する。セレクタ 37 では、選択信号 67 が「0」なので、入力 A (69) を選択し、最大絶対値選択演算の結果 68 として、「-5 (0b1011)」を出力する。そして、レジスタ C には「-5 (0b1011)」をセットする。

【0283】

繰返し 4 回目のステップ (3) は、従来の一般的なプロセッサの場合と同じである。

【0284】

ステップ (15) では、絶対値最大値演算器 15 を使った演算が行われる。この時の動作を図 43 を用いて説明する。

【0285】

レジスタ C の値「-5 (0b1011)」は入力 A (69) に、レジスタ A の値「6 (0b0110)」は入力 B (70) にそれぞれ入力される。

【0286】

第 1 の実施の形態の動作例 1 より、1 の補数絶対値減算器 32 の出力である絶対値比較演算の結果 65 は「(0b1110)」となる。絶対値比較演算の結果 65 は「(0b0000)」ではないので、ゼロ値判断結果 66 は「0」である。選択信号生成手段 36 では、絶対値比較演算の結果 65 が負なので、選択信号 67 として「1」を出力する。セレクタ 37 では、選択信号 67 が「1」なので、入力 B (70) を選択し、最大絶対値選択演算の結果 68 として、「6 (0b0110)」を出力する。そして、レジスタ C には「6 (0b0110)」をセットする。

【0287】

ステップ (9) ~ (12) での動作は、従来の一般的なプロセッサの場合と同じである。

【0288】

以上の処理を行うことにより、レジスタ C に絶対値の最大値である「6」を求めることができる。

【0289】

(動作例 2)

第 4 の実施の形態のプロセッサで、レジスタ A に「0 (0b0000)」、レジスタ C に 4 ビットの 2 の補数表現で表現が可能範囲で絶対値が最大の負の値「-8 (0b1000)」がセットされている場合に、「abmax C, A」命令のステップ (15) が実行される場合を、図 44 を用いて説明する。これは、前記に示した従来のプロセッサではオーバーフローのために正しく絶対値の大小比較を行うことができない場合である。

【0290】

レジスタ C の値「-8 (0b1000)」は入力 A (69) に、レジスタ A の値「0 (0

10

20

30

40

50

b 0 0 0 0)」は入力 B (7 0) にそれぞれ入力される。

【0 2 9 1】

第 1 の実施の形態の動作例 2 より、1 の補数絶対値減算器 3 2 の出力である絶対値比較演算の結果 6 5 は「(0 b 0 1 1 1)」となる。絶対値比較演算の結果 6 5 は「(0 b 0 0 0 0)」ではないので、ゼロ値判断結果 6 6 は「0」である。選択信号生成手段 3 6 では、絶対値比較演算の結果 6 5 は負でなく、ゼロ値判断結果 6 6 も「0」なので、選択信号 6 7 として「0」を出力する。セクタ 3 7 では、選択信号 6 7 が「0」なので、入力 A (6 9) を選択し、最大絶対値選択演算の結果 6 8 として、「- 8 (0 b 1 0 0 0)」を出力する。そして、レジスタ C には「- 8 (0 b 1 0 0 0)」をセットする。

【0 2 9 2】

このように第 4 の実施の形態のプロセッサにおいては「- 8 (0 b 1 0 0 0)」と「0 (0 b 0 0 0 0)」では、「- 8 (0 b 1 0 0 0)」をより絶対値の大きい値として出力することができる。

【0 2 9 3】

(動作例 3)

第 4 の実施の形態のプロセッサで、レジスタ A に「- 4 (0 b 1 1 0 0)」、レジスタ C に「3 (0 b 0 0 1 1)」がセットされている場合に、「a b m a x C, A」ステップ (1 5) が実行される場合を図 4 5 を用いて説明する。これは、前述した第 1 の実施の形態から第 3 の実施の形態のプロセッサでは正しく絶対値の大小比較を行うことができない場合である。

【0 2 9 4】

レジスタ C の値「3 (0 b 0 0 1 1)」は入力 A (6 9) に、レジスタ A の値「- 4 (0 b 1 1 0 0)」は入力 B (7 0) にそれぞれ入力される。

【0 2 9 5】

第 1 の実施の形態の動作例 3 より、1 の補数絶対値減算器 3 2 の出力である絶対値比較演算の結果 6 5 は「(0 b 0 0 0 0)」となる。絶対値比較演算の結果 6 5 は「(0 b 0 0 0 0)」なので、ゼロ値判断結果 6 6 は「1」である。選択信号生成手段 3 6 では、絶対値比較演算の結果 6 5 は負でないが、ゼロ値判断結果 6 6 が「1」で、入力 B (7 0) の値も負なので、選択信号 6 7 として「1」を出力する。セクタ 3 7 では、選択信号 6 7 が「1」なので、入力 B (7 0) を選択し、最大絶対値選択演算の結果 6 8 として、「- 4 (0 b 1 1 0 0)」を出力する。そして、レジスタ C には「- 4 (0 b 1 1 0 0)」をセットする。

【0 2 9 6】

このように第 4 の実施の形態のプロセッサによれば、レジスタ A の値が負 (例えば - 4) で、レジスタ B の値がレジスタ A の値の絶対値の 1 小さい数の正の値 (例えば 3) の場合でも、正しく絶対値の大小比較を行うことができる。

【0 2 9 7】

(動作例 4)

第 4 の第 4 の実施の形態のプロセッサで、絶対値が等しく正負が逆である 2 つの入力値に対して絶対値の大小比較を行う場合の動作例を示す。

【0 2 9 8】

レジスタ A に「- 3 (0 b 0 0 1 1)」、レジスタ C に「3 (0 b 1 1 0 1)」がセットされている場合に、「a b m a x C, A」ステップ (1 5) が実行される場合を図 4 6 を用いて説明する。

【0 2 9 9】

レジスタ C の値「3 (0 b 0 0 1 1)」は入力 A (6 9) に、レジスタ A の値「- 3 (0 b 1 1 0 1)」は入力 B (7 0) にそれぞれ入力される。

【0 3 0 0】

入力 A (6 9) の値「(0 b 0 0 1 1)」から、入力 B (7 0) を 1 の補数表現化した絶対値「(0 b 0 0 1 0)」を減算するので、絶対値比較演算の結果 6 5 は「(0 b 0 0 0 0)」

10

20

30

40

50

1) 」となる。絶対値比較演算の結果65は「(0b0000)」ではないので、ゼロ値判断結果66は「0」である。選択信号生成手段36では、絶対値比較演算の結果65は負でなく、ゼロ値判断結果66も「0」あるので、選択信号67として「0」を出力する。セクタ37では、選択信号67が「0」なので、入力A(69)を選択し、最大絶対値選択演算の結果68として、「3(0b0011)」を出力する。そして、レジスタCには「3(0b0011)」をセットする。

【0301】

レジスタAに「3(0b1101)」、レジスタCに「-3(0b0011)」がセットされている場合に、「abmax C, A」ステップ(15)が実行される場合を図47を用いて説明する。

10

【0302】

レジスタCの値「-3(0b1101)」は入力A(69)に、レジスタAの値「3(0b0011)」は入力B(70)にそれぞれ入力される。

【0303】

入力A(69)を1の補数表現化した絶対値「(0b0010)」から、入力B(70)の値「(0b0011)」を減算するので、絶対値比較演算の結果65は「(0b1111)」となる。絶対値比較演算の結果65は「(0b0000)」ではないので、ゼロ値判断結果66は「0」である。選択信号生成手段36では、絶対値比較演算の結果65は負なので、選択信号67として「1」を出力する。セクタ37では、選択信号67が「1」なので、入力B(70)を選択し、最大絶対値選択演算の結果68として、「3(0b0011)」を出力する。そして、レジスタCには「3(0b0011)」をセットする。

20

【0304】

動作例4から分かるように、第4の実施の形態のプロセッサにおける絶対値最大値演算器では、絶対値が等しく、正負が逆の2つの値が入力された場合には必ず正の値の方を出力する。

【0305】

以上の動作例に示したように第4の実施の形態のプロセッサでは、誤差なしで絶対値が最大の値を求めることができ、4ビットの2の補数表現で表現が可能な負の値で絶対値が最大である値も正しく演算することができる。また、図8に示した従来のプロセッサのプログラムにおけるステップ(7)～ステップ(9)をハードで実行するので、より高速に絶対値が最大の値を求めることができる。

30

【0306】

さらに、絶対値が等しく正負が逆の2つの値を比較した場合には、正の値の方を必ず出力する。このため、図6に示す絶対値最大値演算器15を使って複数のデータの中の絶対値が最大の値を求めた結果、負の値(例えば-4)が求まった場合には、その求まった負の値と絶対値が等しい正の値(例えば4)は、絶対値が最大の値を求める対象となったデータ群の中には存在しないことが保証される。このことは次のような場合に有効である。

【0307】

例えば、第4の実施の形態のプロセッサで、複数のデータの中から絶対値が最大である値を求めた結果、「-4(0b1100)」が求まったとする。すると、その複数のデータの中に「4(0b0100)」は存在しないことが保証される。絶対値が最大である値を求める対象となった複数のデータの個々の値を、最小のビット幅で無駄なく表現する場合には、-4～3の8つの値を表すことができる3ビットのビット幅を与えれば良いことになる。仮に、データ群の中に「4(0b0100)」が存在しないことが保証されなければ、-4～4の9つの値を表すことが可能な4ビットのビット幅が必要となる。このように、第4の実施の形態のプロセッサは従来のプロセッサに比べてデータ群を無駄なく表現できるビット幅を高速に正しく求めることができる。

40

【0308】

第4の実施の形態のプロセッサを用いて絶対値最大値演算器15を使わずに絶対値が最大

50

の値を求めるプログラムを図52に示す。ただし、「abmax」命令は複数のステップに分けて詳細にしている。図52のプログラムにおいて、「abs1 a, b」命令は、レジスタbの値を1の補数表現化による絶対値にしてレジスタaにセットする命令であり、「jmp (be) label」命令は、Nフラグが0の場合にlabelにジャンプする命令であり、「jmp (ne) label」命令は、Zフラグが0の場合にlabelにジャンプする命令である。また、ステップ(20)において「jmp (be)」命令が条件判断に用いる演算結果は、ステップ(6)の演算結果が0の場合はステップ(19)の演算結果であり、ステップ(20)の演算結果が0でない場合はステップ(6)の演算結果である。ステップ(16)、ステップ(17)、ステップ(6)の3ステップが「abmax」命令に対応している。

10

【0309】

なお、第1の実施の形態および第2の実施の形態では1の補数絶対値演算器の構成として図2および図3の第1の1の補数絶対値演算器20や第2の1の補数絶対値演算器21のような構成を示したが、同じ機能が実現できるものであれば他の構成によって本発明の演算器またはプロセッサを実現してもよい。

【0310】

図2、図3に示したものの以外の1の補数絶対値演算器の構成例を図10に示す。図10に示す1の補数絶対値演算器は、入力の符号ビットと、入力の各ビットの排他論理和演算を行う排他論理和演算手段94で構成される。排他論理和演算手段94の制御信号としては入力データの符号ビットが入力される。

20

排他論理和演算手段94の構成の例としては図50に示すものが適用できる。図50に示す排他論理和演算手段94の入力は4ビット幅である。図50に示す排他論理和演算手段94は、入力の各ビットと制御信号との排他論理和演算を行う複数の排他論理和演算素子38からなる。

【0311】

次に図10に示す1の補数絶対値演算器の動作例を示す。例えば、「3(0b0011)」が入力されたとすると、符号ビットの値「0b0」と入力「3(0b0011)」のそれぞれのビットに対し排他論理和演算が行われて「(0b0011)」を出力する。この場合、排他論理和演算の前後で入力のビットは変化しない。また、「-5(0b1011)」が入力されたとすると、符号ビットの値「0b1」と入力「-5(0b1011)」のそれぞれのビットに対し排他論理和演算が行われて「(0b0100)」を出力する。この場合、排他論理和演算の前後で入力のビットが反転する。この例に示した通り、図10に示すような構成であっても、1の補数の絶対値演算器を構成することができる。

30

【0312】

なお、第2の実施の形態では1の補数絶対値反転演算器の構成として図3の1の補数絶対値反転演算器27のような構成を示したが、他の構成により本発明の演算器を実現してもよい。

【0313】

図3に示したものの以外の1の補数絶対値反転演算器の構成例を図11に示す。図11に示す1の補数の絶対値反転演算器は、入力の符号ビットと、入力の各ビットの排他論理和演算を行い、その結果をビット反転して出力する複数の排他論理和否定演算素子39で構成される。例えば、「3(0b0011)」が入力されたとすると、符号ビットの値「0b0」と入力「3(0b0011)」のそれぞれのビットに対し非排他論理和演算が行われて「(0b1100)」を出力する。また、「-5(0b1011)」が入力されたとすると、符号ビットの値「0b1」と入力「-5(0b1011)」のそれぞれのビットに対し非排他論理和演算が行われて「(0b1011)」を出力する。例に示した通り、図11のような構成であっても、1の補数絶対値反転演算器を構成できる。

40

【0314】

もう一つの1の補数絶対値反転演算器の構成例を図12に示す。図12に示す1の補数の絶対値反転演算器は、符号ビットを反転させる反転器40と、入力の符号ビットを反転し

50

た値と、入力各ビットの排他論理和演算を行う排他論理和演算手段94で構成される。排他論理和演算手段94の制御信号としては、反転器40の出力が入力される。排他論理和演算手段94の構成は図50に示したものと同一なので説明を省略する。

【0315】

次に図12に示す1の補数絶対値反転演算器の動作の例を説明する。例えば、「3(0b0011)」が入力されたとすると、符号ビットの値「0b0」をビット反転した値「0b1」と、入力「3(0b0011)」のそれぞれのビットに対し排他論理和演算が行われて「(0b1100)」を出力する。また、「-5(0b1011)」が入力されたとすると、符号ビットの値「0b1」をビット反転した値「0b0」と、入力「-5(0b1011)」のそれぞれのビットに対し排他論理和演算が行われて「(0b1011)」を出力する。例に示した通り、図12のような構成であっても、1の補数絶対値反転演算器を構成できる。

10

【0316】

これまでに説明した第1の実施の形態から第4の実施の形態のプロセッサおよび従来のプロセッサを用いて図53に示すデータから絶対値が最大のデータを求めた場合に、プロセッサが絶対値が最大のデータを最終的に決定するまでに選択するデータの変遷、および、求まる絶対値が最大のデータを図54に示す。この図に示すように、従来のプロセッサでは本来求まるべき値が「-8」であるのに対し、求まった値は「2」であり、真の値との差が非常に大きくなっている。これは、データの中に含まれている4ビット幅の2の補数で表現できる範囲で絶対値が最大である負の値「-8」を正しく演算できないためである。また、第1の実施の形態から第3の実施の形態のプロセッサでは、絶対値が最大の値として「7」が求まり、従来のプロセッサと比較して求まるべき値との誤差が遥かに小さくなっている。これらに対し、第4の実施の形態のプロセッサでは、絶対値が最大の値として本来求まるべき「-8」が正しく求まっている。このように、本発明の第1の実施の形態から第4の実施の形態のプロセッサによると、従来のプロセッサに比べ絶対値が最大の値を求める精度が格段に向上していることがわかる。

20

【0317】

(第5の実施の形態)

第5の実施の形態のプロセッサは、図7に示した従来の一般的なプロセッサでの通常のALUに変えて、図48に示すようなALUを備えるものである。なお、他の構成は図7に示す従来の一般的なプロセッサと同じである。これは即ち、従来のプロセッサが備えるALUに変更を加えて本発明の1の補数絶対値減算器の機能を持たせたプロセッサである。

30

【0318】

図48に示すようにALUは、排他論理和演算手段71と反転器72とセレクタ73と、排他論理和演算手段74と反転器75とセレクタ76と、加算器78と、セレクタ77を備えている。このALUは、通常の加算演算、減算演算に加え、制御信号10により指定されるレジスタファイル7内の2つのレジスタのそれぞれが保持する2つの値について1の補数表現を用いた絶対値の比較演算を行い、その比較結果を2つの値の絶対値のうちのどちらが大きいかを表す指示情報として出力する。

【0319】

排他論理和演算手段71は、図50に示すように複数の排他論理和演算子38を組み合わせ構成されている。排他論理和演算手段71は、入力として受け取った入力A(69)と、セレクタ73から受け取った1ビットの制御信号との排他論理和演算を行う。反転器72は、入力A(69)の符号のビットを反転したものを符号の反転信号84として出力する。セレクタ73は、制御信号A(79)に基づいて、入力A(69)の符号信号83、符号の反転信号84、固定値0信号85、固定値1信号86、からいずれかを選択して1ビットを出力する。これにより、排他論理和演算手段71は、セレクタ73から制御信号として入力A(69)の符号信号83を受け取った場合は、入力A(69)の絶対値を出力することになる。つまり、符号ビットが「1」で、負の場合は入力A(69)と「1111」との排他論理和演算が行われるため入力A(69)の各ビットが反転されて1

40

50

の補数表現化による絶対値が排他論理和演算手段17から出力され、符号ビットが「0」で、正の場合は入力A(69)と「0000」との排他論理和演算が行われて入力A(69)はそのままの値で排他論理和演算手段17から出力される。また、排他論理和演算手段17が、セクタ73から制御信号として入力A(69)の符号の反転信号84を受け取った場合は、入力A(69)の1の補数表現化による絶対値のビット反転を出力する。

【0320】

排他論理和演算手段74は、図50に示すように複数の排他論理和演算子38を組み合わせて構成されている。排他論理和演算手段74は、入力として受け取った入力B(70)の各ビットと、セクタ76から受け取った1ビットの制御信号との排他論理和演算を行う。反転器75は、入力B(70)の符号のビットを反転したものを符号の反転信号88として出力する。セクタ76は、制御信号B(80)に基づいて、入力B(70)の符号信号87、符号の反転信号88、固定値0信号89、固定値1信号90、からいずれかを選択して1ビットを出力する。

10

【0321】

セクタ77は、制御信号C(81)に基づいて、固定値0信号(91)か固定値1信号(92)かを選択して1ビットを出力する。

【0322】

加算器78は、排他論理和演算手段71からの出力と排他論理和演算手段74からの出力とセクタ77からの出力を加算して、出力82を出力する。

【0323】

20

なお、制御信号A(79)、制御信号B(80)、制御信号C(81)のそれぞれは、制御信号10として、デコーダ5(図7参照)からまとめて送信される。

【0324】

次に、第5の実施の形態におけるプロセッサのALUの動作を説明する。

【0325】

(動作例1)

制御信号A(79)に基づいてセクタ73から「符号信号83」が出力され、制御信号B(80)に基づいてセクタ76から「符号の反転信号88」が出力され、制御信号C(81)に基づいて「固定値1(92)」を出力する場合を説明する。この場合、プロセッサのALUは、入力A(69)から入力B(70)を減じる減算を行う。

30

【0326】

制御信号A(79)に基づいてセクタ73から「符号の信号83」が排他論理和演算手段71に入力されると、排他論理和演算手段71では入力A(69)の各ビットと入力A(69)の符号との排他論理和演算が行われる。そのため、排他論理和演算手段71の出力は入力A(69)の1の補数の絶対値となる。

【0327】

制御信号B(80)に基づいてセクタ76から「符号の反転信号88」が排他論理和演算手段74に入力されると、排他論理和演算手段74では入力B(70)と入力B(70)の符号との排他論理和演算の結果のビット反転が行われる。そのため、排他論理和演算手段74の出力は入力B(70)の1の補数化表現による絶対値のビット反転となる。

40

【0328】

加算器78では、入力A(69)の1の補数表現化による絶対値と、入力B(70)の1の補数表現化による絶対値のビット反転と、セクタ77からの出力「1」を加算することになる。これにより、加算器78は、入力A(69)から入力B(70)を減算した結果を出力82として出力する。よって、第5の実施の形態におけるプロセッサのALUは、第2の実施の形態に示した動作と同じく、入力A(69)と入力B(70)との1の補数絶対値減算を行うことができる。

【0329】

(動作例2)

制御信号A(79)に基づいてセクタ73から「固定値0(85)」が出力され、制御

50

信号 B (80) に基づいてセクタ 76 から「固定値 0 (89)」が出力され、制御信号 C (81) に基づいて「固定値 0 (91)」を出力する場合を説明する。この場合、プロセッサの ALU は、入力 A (69) に入力 B (70) を加える加算を行う。

【0330】

制御信号 A (79) に基づいて「固定値 0 (85)」が排他論理和演算手段 71 に入力されると、排他論理和演算手段 71 では入力 A (69) の各ビットと「0」との排他論理和演算が行われる。そのため、排他論理和演算手段 71 は、入力 A (69) をそのまま出力する。

【0331】

制御信号 B (80) に基づいて「固定値 0 (89)」が排他論理和演算手段 74 に入力されると、排他論理和演算手段 74 では入力 B (70) の各ビットと「0」との排他論理和演算が行われる。そのため、排他論理和演算手段 74 は、入力 B (70) をそのまま出力する。

10

【0332】

加算器 78 では、排他論理和演算手段 74 からの出力である入力 A (69) の値と、排他論理和演算手段 76 からの出力である入力 B (70) の値と、セクタ 77 からの出力である「0」とを加算する。よって、この場合、プロセッサの ALU は入力 A (69) と入力 B (70) との加算を行う。

【0333】

(動作例 3)

20

制御信号 A (79) に基づいてセクタ 73 から「固定値 0 (85)」が出力され、制御信号 B (80) に基づいてセクタ 76 から「固定値 1 (90)」が出力され、制御信号 C (81) に基づいてセクタ 77 から「固定値 1 (92)」を出力する場合を説明する。

【0334】

制御信号 A (79) に基づいて「固定値 0 (85)」が排他論理和演算手段 71 に入力されると、排他論理和演算手段 71 では、入力 A (69) の各ビットと「0」との排他論理和演算が行われる。そのため、排他論理和演算手段 71 は、入力 A (69) をそのまま出力する。

【0335】

30

制御信号 B (80) に基づいて「固定値 1 (90)」が排他論理和演算手段 76 に入力されると、排他論理和演算手段 74 では、入力 B (70) と「1」との排他論理和演算が行われる。そのため、排他論理和演算手段 74 は、入力 B (70) のビット反転を出力する。

【0336】

加算器 78 では、排他論理和演算手段 71 からの出力である入力 A (69) の値と、排他論理和演算手段 76 からの出力である入力 B (70) のビット反転の値と、セクタ 77 からの出力である「1」とを加算する。よって、この場合は、プロセッサの ALU は入力 A (69) から入力 B (70) を減じる減算を行う。

【0337】

40

以上の動作例に示したように、第 5 の実施の形態のプロセッサの ALU は、制御信号 A (79)、制御信号 B (80)、制御信号 C (81) の組み合わせにより、入力 A (69) と入力 B (70) とを用いた異なる種類の演算を行うことが可能である。制御信号 A (79)、制御信号 B (80)、制御信号 C (81) の組み合わせと、それによって得られる演算結果を動作例 1 ~ 3 に示した以外のものも含めて図 49 に示す。

【0338】

このように第 5 の実施の形態のプロセッサのように構成すれば、本発明に係る演算器を用いてプロセッサが本来備えている ALU の機能をも実現することが可能である。

【0339】

なお、第 5 の実施の形態に示した構成以外の構成により本発明に係る演算器を実現し、プ

50

ロセッサが本来備えている A L U の機能をも実現してもよい。

【0340】

なお、第1の実施の形態～第4の実施の形態では、演算器のビット幅が4ビットの場合を示したがそれ以外のビット幅を用いて同じ構成を実現してもよい。

【0341】

なお、第3の実施の形態および第4の実施の形態では、複数のデータの中から絶対値が最大となる値を求める場合の動作を示したが、セクタ33、セクタ37の選択条件を逆にすることにより、絶対値が最小となる値も求めることも可能である。

【0342】

さらに、本発明の演算方法をコンピュータプログラムで表し、これをコンピュータで読み取り可能な記録媒体に記録し、この記録媒体をコンピュータに読み込ませて本発明の演算方法を実現することも可能である。

10

【0343】

さらに、本発明の演算方法をプロセッサのマイクロコードで表し、これをプロセッサの実行命令として、本発明の演算方法を実現することも可能である。

【0344】

【発明の効果】

本発明の請求項1から請求項6のいずれかに記載の演算器は、絶対値の比較に1の補数表現化による絶対値を用いるので、所定のビット幅で2の補数を表現しうる範囲で最大の絶対値をもつ負の値を含む値についても正しく絶対値の大小比較を行うことができる。

20

【0345】

本発明の請求項3または請求項6に記載の演算器によれば、絶対値が同じで符号が異なる2つの値の場合には符号が正の値を絶対値が大きい値として出力する。よって、この演算器を用いて複数のデータの中から絶対値が最大の値を求めた結果として、負の値Aが求めた場合は、データ群の中には値Aと絶対値が同じで符号が逆、つまり正の値が含まれないことを保証することができる。正の値が含まれないことが保証されるので、データ群のそれぞれの値を一定のビット幅で表現するために必要なビット幅を無駄なく設定することができる。

【0346】

本発明の請求項6の演算器によれば、絶対値を比較する2数のうちの被減算値が負の値で、減算値が被減算値の絶対値より1小さい正の値である場合には、被減算値を絶対値が大きい値として出力する。よって1の補数表現化による絶対値の比較を用いて、誤差なく絶対値が最大の値を求めることができる。

30

【0347】

本発明の請求項7に記載の演算方法によれば、本発明の請求項1に記載の演算器を容易に実現することができるとともに、この発明の思想をもとにコンピュータプログラムなどに適用することもできる。

【0348】

本発明の請求項8に記載の演算方法によれば、本発明の請求項2に記載の演算器を容易に実現することができるとともに、この発明の思想をもとにコンピュータプログラムなどに適用することもできる。

40

【0349】

本発明の請求項9に記載の演算方法によれば、本発明の請求項3に記載の演算器を容易に実現することができるとともに、この発明の思想をもとにコンピュータプログラムなどに適用することもできる。

【0350】

本発明の請求項10に記載の演算方法によれば、本発明の請求項4に記載の演算器を容易に実現することができるとともに、この発明の思想をもとにコンピュータプログラムなどに適用することもできる。

【0351】

50

本発明の請求項 1 1 に記載の演算方法によれば、本発明の請求項 5 に記載の演算器を容易に実現することができるとともに、この発明の思想をもとにコンピュータプログラムなどに適用することもできる。

【0352】

本発明の請求項 1 2 に記載の演算方法によれば、本発明の請求項 6 に記載の演算器を容易に実現することができるとともに、この発明の思想をもとにコンピュータプログラムなどに適用することもできる。

【0353】

本発明の請求項 1 3 に記載のプロセッサによれば、プロセッサが備える実行命令により本発明の請求項 1 から請求項 5 のいずれかに記載の演算器を用いて 2 つの値の絶対値の大小比較や、複数のデータのなかから絶対値が最も大きな値を求めることができる。

10

【図面の簡単な説明】

【図 1】 本発明を適用した第 1 の実施の形態および第 2 の実施の形態のプロセッサの構成を示す図。

【図 2】 図 1 に示した 1 の補数絶対値減算器 1 3 の第 1 の実施の形態での構成を示す図。

【図 3】 図 1 に示した 1 の補数絶対値減算器 1 3 の第 2 の実施の形態での構成を示す図。

【図 4】 本発明を適用した第 3 の実施の形態および第 4 の実施の形態のプロセッサの構成を示す図。

20

【図 5】 図 4 に示した絶対値最大値演算器 1 5 の第 3 の実施の形態での構成を示す図。

【図 6】 図 4 に示した絶対値最大値演算器 1 5 の第 4 の実施の形態での構成を示す図。

【図 7】 従来一般的なプロセッサの構成を示す図。

【図 8】 従来の 2 の補数の絶対値減算器を備えたプロセッサの構成を示す図。

【図 9】 図 8 に示した 2 の補数絶対値減算器 1 4 の構成を示す図。

【図 10】 1 の補数の絶対値演算器の構成の例を示す図。

【図 11】 1 の補数の絶対値反転演算器の構成の例を示す図。

【図 12】 1 の補数の絶対値反転演算器の構成の例を示す図。

【図 13】 図 7 に示した従来一般的なプロセッサで絶対値が最大である値を求めるプログラムを示す図。

30

【図 14】 図 8 に示した従来プロセッサで絶対値が最大である値を求めるプログラムを示す図。

【図 15】 図 4 に示した第 3 の実施の形態および第 4 の実施の形態のプロセッサで絶対値の最大値を求めるプログラムを示す図。

【図 16】 絶対値が最大である値を求める対象となるデータが格納されているメモリを示す図。

【図 17】 従来プロセッサで処理を行った場合の繰返し 1 回目の動作における各数値を示す図。

【図 18】 従来プロセッサで処理を行った場合の繰返し 2 回目の動作における各数値を示す図。

40

【図 19】 従来プロセッサで処理を行った場合の繰返し 3 回目の動作における各数値を示す図。

【図 20】 従来プロセッサで処理を行った場合の繰返し 4 回目の動作における各数値を示す図。

【図 21】 従来プロセッサで絶対値が最大の負の値を処理した場合における各数値を示す図。

【図 22】 第 1 の実施の形態の動作例 1 での繰返し 1 回目の動作における各数値を図。

【図 23】 第 1 の実施の形態の動作例 1 での繰返し 2 回目の動作における各数値を示す図。

【図 24】 第 1 の実施の形態の動作例 1 での繰返し 3 回目の動作における各数値を示す

50

図。

【図 2 5】 第 1 の実施の形態の動作例 1 での繰返し 4 回目の動作における各数値を示す図。

【図 2 6】 第 1 の実施の形態の動作例 2 での動作における各数値を示す図。

【図 2 7】 第 1 の実施の形態の動作例 3 での動作における各数値を示す図。

【図 2 8】 第 2 の実施の形態の動作例 1 での繰返し 1 回目の動作における各数値を示す図。

【図 2 9】 第 2 の実施の形態の動作例 1 での繰返し 2 回目の動作における各数値を示す図。

【図 3 0】 第 2 の実施の形態の動作例 1 での繰返し 3 回目の動作における各数値を示す図。 10

【図 3 1】 第 2 の実施の形態の動作例 1 での繰返し 4 回目の動作における各数値を示す図。

【図 3 2】 第 2 の実施の形態の動作例 2 での動作における各数値を示す図。

【図 3 3】 第 2 の実施の形態の動作例 3 での動作における各数値を示す図。

【図 3 4】 第 3 の実施の形態の動作例 1 での繰返し 1 回目の動作における各数値を示す図。

【図 3 5】 第 3 の実施の形態の動作例 1 での繰返し 2 回目の動作における各数値を示す図。

【図 3 6】 第 3 の実施の形態の動作例 1 での繰返し 3 回目の動作における各数値を示す図。 20

【図 3 7】 第 3 の実施の形態の動作例 1 での繰返し 4 回目の動作における各数値を示す図。

【図 3 8】 第 3 の実施の形態の動作例 2 での動作における各数値を示す図。

【図 3 9】 第 3 の実施の形態の動作例 3 での動作における各数値を示す図。

【図 4 0】 第 4 の実施の形態の動作例 1 での繰返し 1 回目の動作における各数値を示す図。

【図 4 1】 第 4 の実施の形態の動作例 1 での繰返し 2 回目の動作における各数値を示す図。

【図 4 2】 第 4 の実施の形態の動作例 1 での繰返し 3 回目の動作における各数値を示す図。 30

【図 4 3】 第 4 の実施の形態の動作例 1 での繰返し 4 回目の動作における各数値を示す図。

【図 4 4】 第 4 の実施の形態の動作例 2 での動作における各数値を示す図。

【図 4 5】 第 4 の実施の形態の動作例 3 での動作における各数値を示す図。

【図 4 6】 第 4 の実施の形態の動作例 4 での動作における各数値を示す図。

【図 4 7】 第 4 の実施の形態の動作例 4 での動作における各数値を示す図。

【図 4 8】 第 5 の実施の形態のプロセッサの A L U の構成を示す図。

【図 4 9】 第 5 の実施の形態のプロセッサの A L U の制御信号と機能の組み合わせを示す図。 40

【図 5 0】 排他論理和演算手段 9 4 の構成を示す図。

【図 5 1】 第 1 の実施の形態のプロセッサで絶対値が最大である値を求めるプログラムを示す図。

【図 5 2】 第 4 の実施の形態のプロセッサで絶対値が最大である値を求めるプログラムを示す図。

【図 5 3】 絶対値が最大であるデータを求める対象となるデータを示す図。

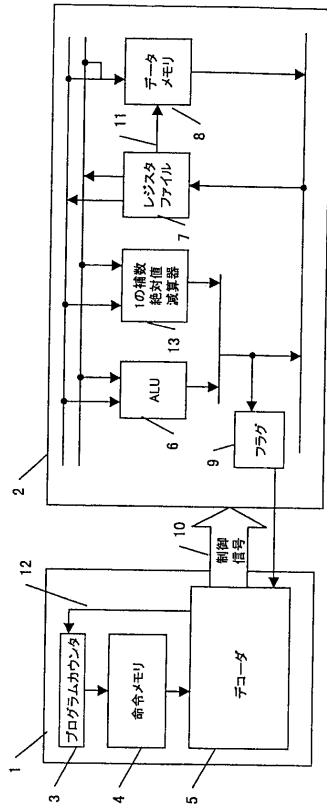
【図 5 4】 第 1 の実施の形態から第 4 の実施の形態のプロセッサおよび従来のそれぞれのプロセッサを用いて、図 5 3 に示すデータから絶対値が最大であるデータを求めた場合のデータの選択過程および最終的に求まるデータの違いを示す図。

【符号の説明】

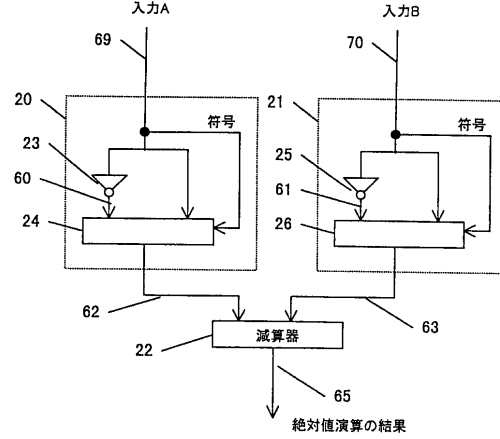
1	制御部	
2	データバス部	
3	プログラムカウンタ	
4	実行命令メモリ	
5	デコーダ	
6	A L U	
7	レジスタファイル	
8	データメモリ	
9	フラグ設定部	
10	制御信号	10
11	アドレス信号	
12	制御信号	
13	1の補数絶対値減算器	
14	2の補数絶対値減算器	
15	絶対値最大値演算器	
20	第1の1の補数絶対値演算器	
21	第2の1の補数絶対値演算器	
22	減算器（指示情報出力手段）	
23	反転器	
24	セレクタ	20
25	反転器	
26	セレクタ	
27	1の補数絶対値反転演算器	
28	加算器（指示情報出力手段）	
29	反転器	
30	セレクタ	
31	反転器	
32	1の補数絶対値減算器	
33	セレクタ	
34	ゼロ検出器	30
35	選択手段	
36	選択信号生成手段	
37	セレクタ	
38	排他論理和演算素子	
39	排他論理和否定演算素子	
40	反転器	
50	第1の2の補数絶対値演算器	
51	第2の2の補数絶対値演算器	
52	反転器	
53	+1加算器	40
54	セレクタ	
55	反転器	
56	+1加算器	
57	セレクタ	
60	符号反転後の値	
61	符号反転後の値	
62	絶対値演算処理後の値	
63	絶対値演算処理後の値	
64	絶対値反転処理後の値	
65	絶対値比較演算の結果（指示情報）	50

6 6	ゼロ値判断結果	
6 7	選択信号	
6 8	絶対値最大値演算の結果	
6 9	入力 A	
7 0	入力 B	
7 1	排他論理和演算手段	
7 2	反転器	
7 3	セレクタ	
7 4	排他論理和演算手段	
7 5	反転器	10
7 6	セレクタ	
7 7	セレクタ	
7 8	加算器	
7 9	制御信号 A	
8 0	制御信号 B	
8 1	制御信号 C	
8 3	符号	
8 4	符号の反転	
8 5	固定値 0	
8 6	固定値 1	20
8 7	符号	
8 8	符号の反転	
8 9	固定値 0	
9 0	固定値 1	
9 1	固定値 0	
9 2	固定値 1	
9 4	排他論理和演算手段	

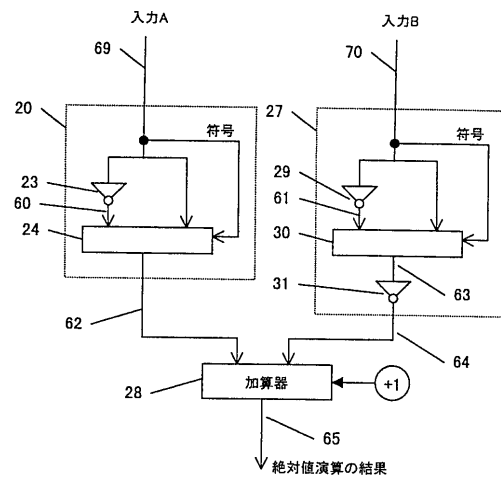
【図 1】



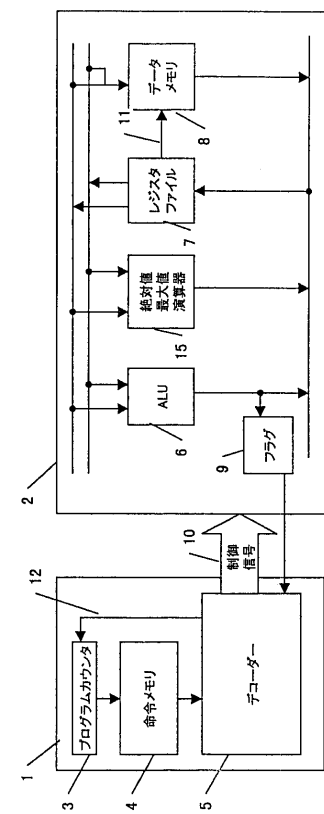
【図 2】



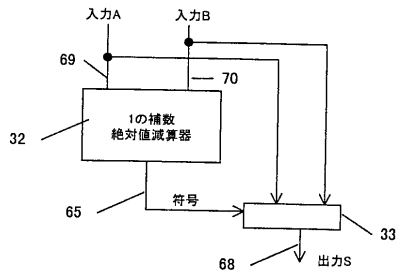
【図 3】



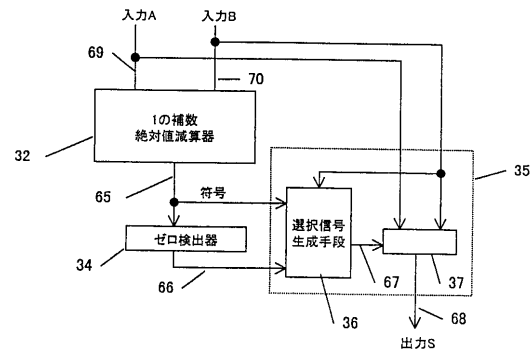
【図 4】



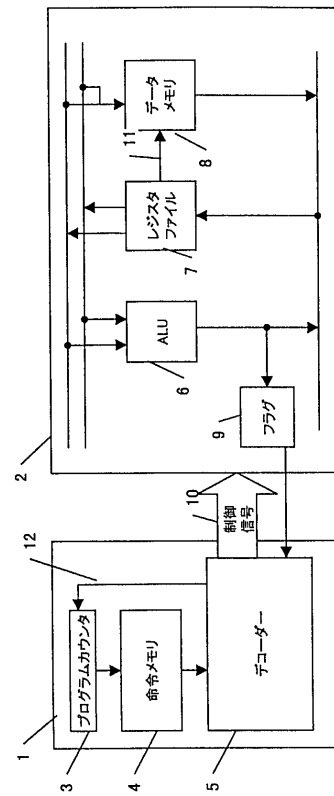
【図 5】



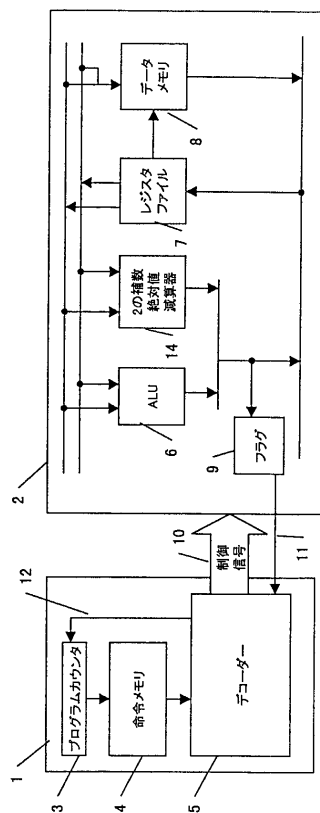
【図 6】



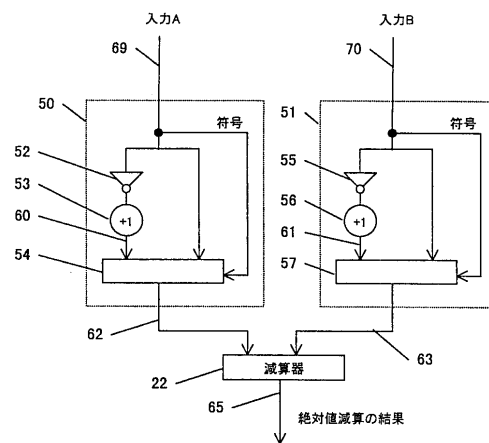
【図 7】



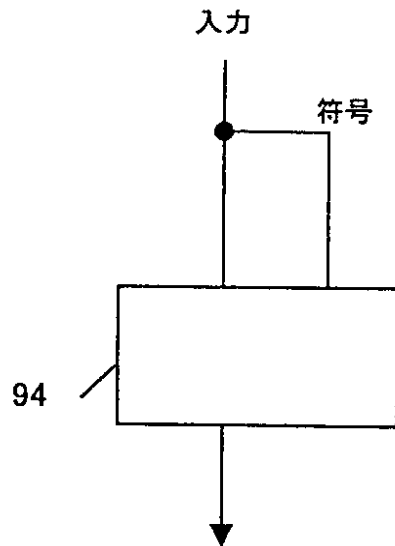
【図 8】



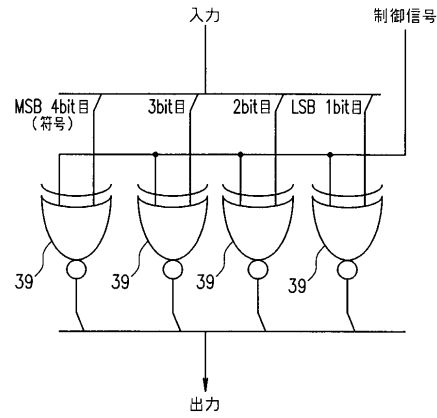
【図 9】



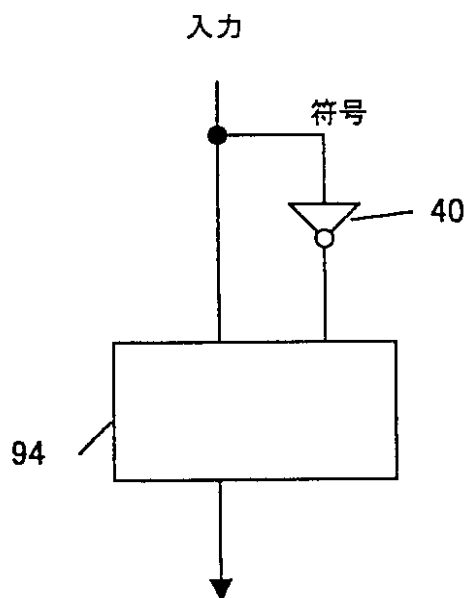
【図 10】



【図 11】



【図 12】



【図 13】

```

movei P, 0          - (1)
movei C, 0          - (2)
loop_begin: move A, M(P) - (3)
abs B, A            - (4)
abs D, C            - (5)
cmp D, B            - (6)
jmp(b) cmp_small    - (7)
move C, A           - (8)
cmp_small: cmpi P, 3 - (9)
jmp(e) end          - (10)
addi P, 1           - (11)
jmp loop_begin      - (12)
end: nop            - (13)

```

【図 14】

```

movei P, 0          - (1)
movei C, 0          - (2)
loop_begin: move A, M(P) - (3)
abcmp C, A          - (14)
jmp(b) cmp_small    - (7)
move C, A           - (8)
cmp_small: cmpi P, 3 - (9)
jmp(e) end          - (10)
addi P, 1           - (11)
jmp loop_begin      - (12)
end: nop            - (13)

```

【図 15】

```

        move    P, 0          - (1)
        move    C, 0          - (2)
loop_begin: move    A, M(P)    - (3)
        abmax   C, A          - (15)
        cmpi    P, 3          - (9)
        jmp(e)  end           - (10)
        addi    P, 1          - (11)
        jmp     loop_begin    - (12)
end:     nop                 - (13)

```

【図 16】

アドレス	データ
0	3
1	-5
2	-2
3	6

【図 17】

入力A 69	0 (0b0000)
符号反転後の値 60	0 (0b0000)
絶対値演算後の値 62	0 (0b0000)
入力B 70	3 (0b0011)
符号反転後の値 61	-3 (0b1101)
絶対値演算後の値 63	3 (0b0011)
絶対値減算の結果 65	-3 (0b1101)

【図 18】

入力A 69	3 (0b0011)
符号反転後の値 60	-3 (0b1101)
絶対値演算後の値 62	3 (0b0011)
入力B 70	-5 (0b1011)
符号反転後の値 61	5 (0b0101)
絶対値演算後の値 63	5 (0b0101)
絶対値減算の結果 65	-2 (0b1110)

【図 19】

入力A 69	-5 (0b1011)
符号反転後の値 60	5 (0b0101)
絶対値演算後の値 62	5 (0b0101)
入力B 70	-2 (0b1110)
符号反転後の値 61	2 (0b0010)
絶対値演算後の値 63	2 (0b0010)
絶対値減算の結果 65	3 (0b0011)

【図 21】

入力A 69	-8 (0b1000)
符号反転後の値 60	-8 (0b1000)
絶対値演算後の値 62	-8 (0b1000)
入力B 70	0 (0b0000)
符号反転後の値 61	0 (0b0000)
絶対値演算後の値 63	0 (0b0000)
絶対値減算の結果 65	-8 (0b1000)

【図 20】

入力A 69	-5 (0b1011)
符号反転後の値 60	5 (0b0101)
絶対値演算後の値 62	5 (0b0101)
入力B 70	6 (0b0110)
符号反転後の値 61	-6 (0b1010)
絶対値演算後の値 63	6 (0b0110)
絶対値減算の結果 65	-1 (0b1111)

【図 22】

入力A 69	0 (0b0000)
符号反転後の値 60	(0b1111)
絶対値演算後の値 62	(0b0000)
入力B 70	3 (0b0011)
符号反転後の値 61	(0b1100)
絶対値演算後の値 63	(0b0011)
絶対値減算の結果 65	(0b1101)

【図 2 3】

入力A 69	3 (0b0011)
符号反転後の値 60	(0b1100)
絶対値演算後の値 62	(0b0011)
入力B 70	-5 (0b1011)
符号反転後の値 61	(0b0100)
絶対値演算後の値 63	(0b0100)
絶対値減算の結果 65	(0b1111)

【図 2 4】

入力A 69	-5 (0b1011)
符号反転後の値 60	(0b0100)
絶対値演算後の値 62	(0b0100)
入力B 70	-2 (0b1110)
符号反転後の値 61	(0b0001)
絶対値演算後の値 63	(0b0001)
絶対値減算の結果 65	(0b0011)

【図 2 5】

入力A 69	-5 (0b1011)
符号反転後の値 60	(0b0100)
絶対値演算後の値 62	(0b0100)
入力B 70	6 (0b0110)
符号反転後の値 61	(0b1001)
絶対値演算後の値 63	(0b0110)
絶対値減算の結果 65	(0b1110)

【図 2 6】

入力A 69	-8 (0b1000)
符号反転後の値 60	(0b0111)
絶対値演算後の値 62	(0b0111)
入力B 70	0 (0b0000)
符号反転後の値 61	(0b1111)
絶対値演算後の値 63	(0b0000)
絶対値減算の結果 65	(0b0111)

【図 2 7】

入力A 69	3 (0b0011)
符号反転後の値 60	(0b1100)
絶対値演算後の値 62	(0b0011)
入力B 70	-4 (0b1100)
符号反転後の値 61	(0b0011)
絶対値演算後の値 63	(0b0011)
絶対値減算の結果 65	(0b0000)

【図 2 9】

入力A 69	3 (0b0011)
符号反転後の値 60	(0b1100)
絶対値演算後の値 62	(0b0011)
入力B 70	-5 (0b1011)
符号反転後の値 61	(0b0100)
絶対値演算後の値 63	(0b0100)
絶対値反転後の値 64	(0b1011)
絶対値減算の結果 65	(0b1111)

【図 2 8】

入力A 69	0 (0b0000)
符号反転後の値 60	(0b1111)
絶対値演算後の値 62	(0b0000)
入力B 70	3 (0b0011)
符号反転後の値 61	(0b1100)
絶対値演算後の値 63	(0b0011)
絶対値反転後の値 64	(0b1100)
絶対値減算の結果 65	(0b1101)

【図 3 0】

入力A 69	-5 (0b1011)
符号反転後の値 60	(0b0100)
絶対値演算後の値 62	(0b0100)
入力B 70	-2 (0b1110)
符号反転後の値 61	(0b0001)
絶対値演算後の値 63	(0b0001)
絶対値反転後の値 64	(0b1110)
絶対値減算の結果 65	(0b0011)

【図 3 1】

入力A 69	-5 (0b1011)
符号反転後の値 60	(0b0100)
絶対値演算後の値 62	(0b0100)
入力B 70	6 (0b0110)
符号反転後の値 61	(0b1001)
絶対値演算後の値 63	(0b0110)
絶対値反転後の値 64	(0b1001)
絶対値減算の結果 65	(0b1110)

【図 3 2】

入力A 69	-8 (0b1000)
符号反転後の値 60	(0b0111)
絶対値演算後の値 62	(0b0111)
入力B 70	0 (0b0000)
符号反転後の値 61	(0b1111)
絶対値演算後の値 63	(0b0000)
絶対値反転後の値 64	(0b1111)
絶対値減算の結果 65	(0b0111)

【図 3 3】

入力A 69	3 (0b0011)
符号反転後の値 60	(0b1100)
絶対値演算後の値 62	(0b0011)
入力B 70	-4 (0b1100)
符号反転後の値 61	(0b0011)
絶対値演算後の値 63	(0b0011)
絶対値反転後の値 64	(0b1100)
絶対値減算の結果 65	(0b0000)

【図 3 4】

入力A 69	0 (0b0000)
入力B 70	3 (0b0011)
絶対値減算の結果 65	(0b1101)
絶対値最大値演算の結果 68	3 (0b0011)

【図 3 5】

入力A 69	3 (0b0011)
入力B 70	-5 (0b1011)
絶対値減算の結果 65	(0b1111)
絶対値最大値演算の結果 68	-5 (0b1011)

【図 3 6】

入力A 69	-5 (0b1011)
入力B 70	-2 (0b1110)
絶対値減算の結果 65	(0b0011)
絶対値最大値演算の結果 68	-5 (0b1011)

【図 3 7】

入力A 69	-5 (0b1011)
入力B 70	6 (0b0110)
絶対値減算の結果 65	(0b1110)
絶対値最大値演算の結果 68	6 (0b0110)

【図 3 8】

入力A 69	-8 (0b1000)
入力B 70	0 (0b0000)
絶対値減算の結果 65	(0b0111)
絶対値最大値演算の結果 68	-8 (0b1000)

【図 3 9】

入力A 69	3 (0b0011)
入力B 70	-4 (0b1100)
絶対値減算の結果 65	(0b0000)
絶対値最大値演算の結果 68	3 (0b0011)

【図 4 0】

入力A 69	0 (0b0000)
入力B 70	3 (0b0011)
絶対値減算の結果 65	(0b1101)
ゼロ検出結果 66	0
選択信号 67	1
絶対値最大値演算の結果 68	3 (0b0011)

【図 4 1】

入力A 69	3 (0b0011)
入力B 70	-5 (0b1011)
絶対値減算の結果 65	(0b1111)
ゼロ検出結果 66	0
選択信号 67	1
絶対値最大値演算の結果 68	-5 (0b1011)

【図 4 2】

入力A 69	-5 (0b1011)
入力B 70	-2 (0b1110)
絶対値減算の結果 65	(0b0011)
ゼロ検出結果 66	0
選択信号 67	0
絶対値最大値演算の結果 68	-5 (0b1011)

【図 4 3】

入力A 69	-5 (0b1011)
入力B 70	6 (0b0110)
絶対値減算の結果 65	(0b1110)
ゼロ検出結果 66	0
選択信号 67	1
絶対値最大値演算の結果 68	6 (0b0110)

【図 4 4】

入力A 69	-8 (0b1000)
入力B 70	0 (0b0000)
絶対値減算の結果 65	(0b0111)
ゼロ検出結果 66	0
選択信号 67	0
絶対値最大値演算の結果 68	-8 (0b1000)

【図 4 5】

入力A 69	3 (0b0011)
入力B 70	-4 (0b1100)
絶対値減算の結果 65	(0b0000)
ゼロ検出結果 66	1
選択信号 67	1
絶対値最大値演算の結果 68	-4 (0b1100)

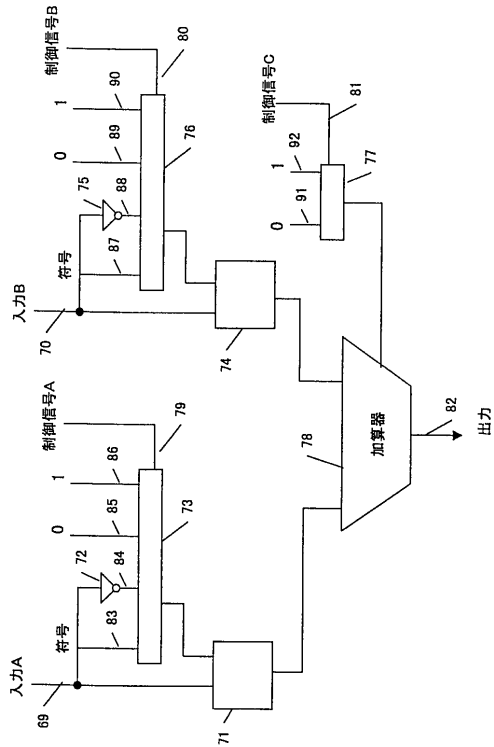
【図 4 6】

入力A 69	3 (0b0011)
入力B 70	-3(0b1101)
絶対値減算の結果 65	(0b0001)
ゼロ検出結果 66	0
選択信号 67	0
絶対値最大値演算の結果 68	3 (0b0011)

【図 4 7】

入力A 69	-3(0b1101)
入力B 70	3 (0b0011)
絶対値減算の結果 65	(0b1111)
ゼロ検出結果 66	0
選択信号 67	1
絶対値最大値演算の結果 68	3 (0b0011)

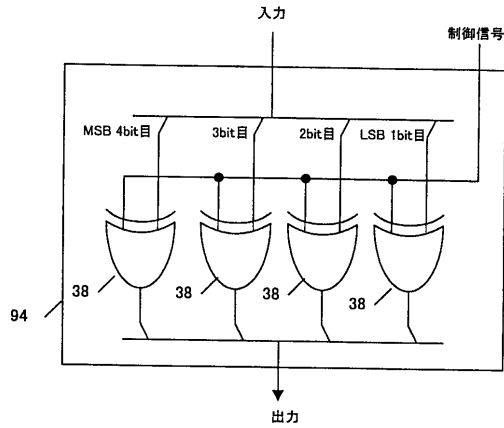
【図 4 8】



【図 4 9】

出力 82	
制御信号A 79	制御信号B 80
符号 83	符号の反転 88
符号の反転 84	符号 87
符号 83	符号 87
固定値0 85	固定値0 89
固定値0 85	固定値1 90
固定値1 86	固定値0 89
1の補数の絶対値減算 $ A - B $	1の補数の絶対値減算 $ B - A $
1の補数の絶対値減算 $ B - A $	1の補数の絶対値加算 $ A + B $
加算 $A + B$	減算 $A - B$
減算 $A - B$	減算 $B - A$

【図 50】



【図 51】

```

movei P, 0      - (1)
movei C, 0      - (2)
loop_begin: move A, M(P)  - (3)
abs1 D, C       - (16)
abs1 B, A       - (17)
cmp D, B        - (6)
jmp(be) cmp_small - (7)
move C, A       - (8)
cmp_small: cmpi P, 3    - (9)
jmp(e) end      - (10)
addi P, 1       - (11)
jmp loop_begin  - (12)
end: nop        - (13)

```

【図 52】

```

movei P, 0      - (1)
movei C, 0      - (2)
loop_begin: move A, M(P)  - (3)
abs1 D, C       - (16)
abs1 B, A       - (17)
cmp D, B        - (6)
jmp(ne) not_equal - (18)
cmpi A, 0       - (19)
not_equal: jmp(be) cmp_end - (20)
move C, A       - (8)
cmp_end: cmpi P, 3    - (9)
jmp(e) end      - (10)
addi P, 1       - (11)
jmp loop_begin  - (12)
end: nop        - (13)

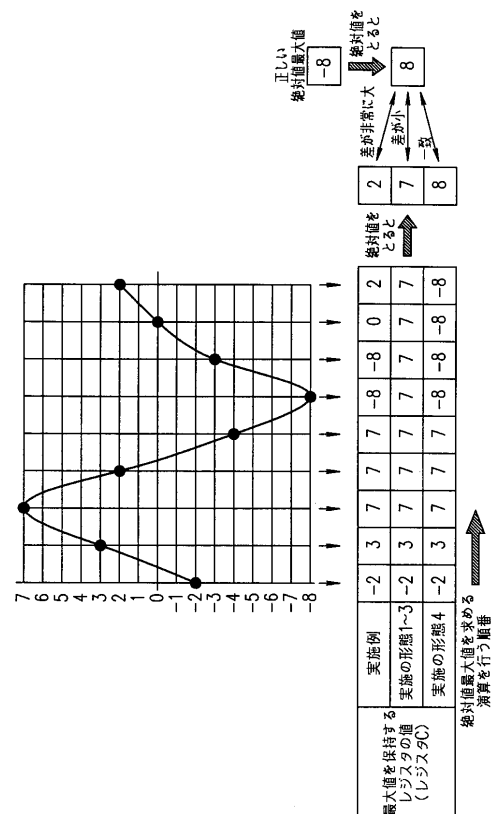
```

ステップ(20)のjmpが条件判断に使う演算結果
 ステップ(6)の結果が等しい時: ステップ(19)の演算結果
 ステップ(6)の結果が等しくなかった時: ステップ(8)の演算結果

【図 53】

-2
3
7
2
-4
-8
-3
0
2

【図 54】



フロントページの続き

- (56)参考文献 特開平10-161850 (JP, A)
特開平03-085823 (JP, A)
特開昭61-165127 (JP, A)

- (58)調査した分野(Int.Cl.⁷, DB名)
G06F 7/02