OPIC
Office de la propriété intellectuelle du Canada

CIPO
Canadian Intellectual Property Office

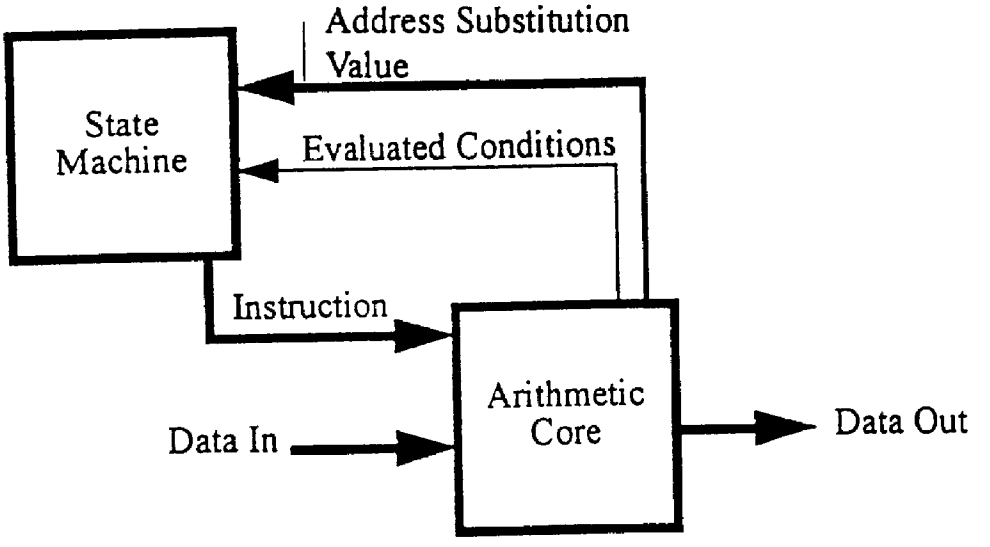
(12) (19) (CA) **Demande-Application**

(21) (A1) **2,244,810**
(22) 1995/03/23
(43) 1995/09/25
(62) 2,145,379
(22) 1995/03/23

(72) ROBBINS, William Philip, GB
(72) WISE, Adrian Philip, GB
(71) Discovision Associates, US

(51) Int.Cl.[6] G11C 8/00, G06F 12/02
(30) 1994/03/24 (9405914.4) GB
(30) 1994/07/29 (9415387.1) GB
(30) 1995/02/28 (9503964.0) GB

(54) **METHODE D'ACCES MEMOIRE**
(54) **METHOD FOR ADDRESSING MEMORY**



(57) L'invention est constituée par une méthode et un dispositif d'accès à une mémoire. Dans l'une des concrétisations de l'invention, est divulguée une procédure d'obtention de mots de longueur fixe ayant recours à un nombre de bits fixe pour avoir accès à des données de longueur variable, ainsi qu'à un champ de définition de la longueur et à un champ d'adresse. De plus, une procédure d'accès mémoire utilisant un mot de longueur fixe et un nombre de bits fixe pour avoir accès aux données, ainsi qu'un champ de substitution et un champ d'adresse, est discutée. Un dispositif d'adressage mémoire incluant un automate fini et un noyau arithmétique est également divulgué.

(57) A method and apparatus for addressing memory is disclosed. In one embodiment, a procedure for providing a word with fixed width, having a fixed number of bits to be used for addressing variable width data, and having a width defining field and address field, is disclosed. In addition, a procedure for addressing memory with a fixed width word, having a fixed number of bits, to be used for addressing data and having a substitution field and an address field, is discussed. Also, an apparatus for addressing memory, including a state machine and an arithmetic core is disclosed.

ABSTRACT

A method and apparatus for addressing memory is disclosed. In one embodiment, a procedure for providing a word with fixed width, having a fixed number of bits to be used for addressing variable width data, and having a width defining field and address field, is disclosed. In addition, a procedure for addressing memory with a fixed width word, having a fixed number of bits, to be used for addressing data and having a substitution field and an address field, is discussed. Also, an apparatus for addressing memory, including a state machine and an arithmetic core is disclosed.

Background of the Invention

This invention relates generally to a method and apparatus for addressing memory and, in particular, to using a fixed width word having a fixed number of bits to be used for addressing variable width data and address substitution.

Summary of the Invention

In accordance with the present invention, there is provided a procedure for addressing memory characterized by: providing a fixed width word having a predetermined fixed number of bits to be used for addressing data; defining the fixed width word with an address field and a substitution field; defining the address field with a plurality of bits defining the address of the data; defining a variable width substitution field with at least one substitution bit; the substitution field has at least one bit to serve as a termination marker between the address field and the substitution field; using the substitution field to indicate substituted bits from a separate addressing source; and maintaining a fixed width word for addressing variable width data while inversely varying the width of the address field and the width of the substitution field.

Brief Description of the Drawings

Figure 1 is a block diagram of the Microcodable state machine.

Figure 2 is a block diagram of the Arithmetic Core.

Figure 3 is a functional diagram of the Register File.

Figure 4 is a block diagram of data flow in the Register File.

Figure 5 is a block diagram of Register File address decoding.

Figure 6 is a fixed width word to be used for addressing, having a width defining field and an address field.

Figure 7 is a fixed width word to be used for addressing, having an address field, a substitution field and a substitution indicator.

Figure 8 is an example of a 13 bit word to be used to address 8 bit data in a 64 x 32 RAM.

Figure 9 is an example of a fixed width word having fields.

Detailed Description of the Invention for Memory Addressing

A method and apparatus for addressing memory is described herein. In particular, the process calls for using a fixed width word for addressing variable width data. In various forms of the embodiment, the fixed width word may contain a width defining field, an address field, or a substitution field. The length of the fixed width word is

predetermined by the amount of memory to be addressed. The apparatus form of the present invention includes a microcodable state machine with an arithmetic core.

The microcodable state machine is intended to be used to solve design problems where there is a need for versatile and/or complicated calculations. Examples of such designs are: address generation, stream parsing and decoding or filter tap coefficient calculations. The addressing must cope with two different features, variable length addresses to access varying width portions of words and address substitution. In the present invention, a RAM having a 64 x 32 bit configuration can be addressed in partial words having 64 x 32 bit, 128 x 16 bit, 256 x 8 bit, 512 x 4 bit, 1024 x 2 bit or 2048 x 1 bit formats.

VARIABLE LENGTH FIELDS WITHIN A FIXED WIDTH WORD

In many applications it is useful to define variable portions of a word (to be known as fields) for actions such as substitution, variable width data addressing, or the constriction of other parts of the word. The conventional method for this would be to have an additional word (or words) to specify the width of the field (or fields) within the

word. Below a method for encoding this information within the word itself will be described. This method has the advantages of saving bits in the overall definition of the word, simplifying decoding of the encoded word and providing a more intuitive view of what has been encoded. This encoding method is applicable if the variable width fields are most or least significant bit justified within the word.

Table 1-1 shows two examples of variable width fields (marked "F") least significant bit justified defined within an eight bit word, "w" marks other potential fields of these words.

Table 1-1

Bit number (hex)	7	6	5	4	3	2	1	0
Fixed word	w	w	w	F	F	F	F	F
	w	w	w	w	w	w	F	F

Table 1-2 shows the conventional method of encoding the fields shown in Table 1-1 by the addition of enough bits to specify the maximum width of the field in binary. (Bits marked "x" are "don't care" - a term of art). Too much space is taken up with this method.

Table 1-2

Bit number (hex)	7	6	5	4	3	2	1	0	Field Define		
Fixed word	w	w	w	x	x	x	x	x	1	0	1
	w	w	w	w	w	w	x	x	0	1	0

Table 1-3 shows the encoding of the fields shown in Table 1-1 using the new method. This method defines the field by using a continuation marker and a termination marker. In this case the continuation marker is "1" and the termination marker is "0". The field is defined as all continuation markers from the justified end of the field (least significant in this case) until and including the termination marker. It is shown in Table 1-3 that to encode the field, the space taken by a termination marker must be added

to the fixed width word at the start of the field, this allows the definition of a zero length field by the additional space containing a termination marker.

Table 1-3

Bit number (hex)	7	6	5	4	3	2	1	0	
Fixed word	w	w	w	0	1	1	1	1	1
Continuation marker = 1; Termination marker = 0.	w	w	w	w	w	w	0	1	1

It can be seen that the advantages of this encoding method are:

1. A reduction in the number of bits needed in the encoding.
2. A simplification in the decoding required since the need for a "x to 1 of 2^x" decode of the "field define" shown in Table 1-2 that would normally be needed is inherent in the encoding which is already in the form of 1 of 2^x.
3. The encoding is in a more intuitive form allowing the field defined to be easily identified.

The use of this encoding can be widened by stating that the termination marker and the continuation marker can be reversed to make the encoding of Table 1-3 resemble that of Table 1-4. In addition, the use of "1" or "0" is used interchangeably throughout this application.

Table 1-4

Bit number (hex)	7	6	5	4	3	2	1	0	
Fixed word	w	w	w	1	0	0	0	0	0
Continuation marker = 1; Termination marker = 0.	w	w	w	w	w	w	1	0	0

Fields may also be most significant bit justified such as shown in Table 1-5. These are encoded in a similar way to least significant bit justified fields simply that the field reaches from the most significant bit (hereinafter "MSB") towards the least significant bit (hereinafter "LSB") up to and including the first termination marker. The encoding of the fields shown in Table 1-5 are shown in Table 1-6.

Table 1-5

Bit number (hex)	7	6	5	4	3	2	1	0
Fixed word	F	F	F	F	F	w	w	w
	F	F	w	w	w	w	w	w

Table 1-6

Bit number (hex)		7	6	5	4	3	2	1	0
Fixed word	1	1	1	1	1	0	w	w	w
Continuation marker = 1; Termination marker = 0.	1	1	0	w	w	w	w	w	w

Finally, fields may be encoded from the least significant and most significant ends of the word simultaneously. For example, the two fields shown in Table 1-7 may be encoded as in Table 1-8, with the addition of just one bit for each field for the reason explained earlier.

Table 1-7

Bit number (hex)	7	6	5	4	3	2	1	0
Fixed word	F	F	F	F	w	w	F	F
	w	w	w	w	F	F	F	F

Table 1-8

Bit number (hex)		7	6	5	4	3	2	1	0	
Fixed word	1	1	1	1	0	w	w	0	1	1
Continuation marker = 1; Termination marker = 0.	0	w	w	w	w	0	1	1	1	1

In Figure 9 the general above concept is illustrated. An address field, not necessarily used to address data has a field having a termination marker and a continuation marker. In this case the field is least significant bit justified.

USING FIXED WIDTH WORD WITH VARIABLE LENGTH FIELDS TO PERFORM ADDRESS SUBSTITUTION.

There are situations in which it is useful to substitute part of a memory address by another value. In this way it is possible to construct a data dependent address. The encoding method used in Claim 1 can be applied to the addresses of a memory to specify what portion of the address is to be substituted. If a least significant bit justified variable length field is used in this address, a substitution field can be defined. For example, a 12 bit address 0baaaaaaaaaaa encoded to have its five least significant bit substituted by the 12 bit value 0bccccccccccc would be 0baaaaaaa011111 and produce the address 0baaaaaaaccccc. Table 1-9 shows the encoding for substitution into a 12 bit address.

Table 1-9: Address substitution

No. Bits Substituted	B	A	9	8	7	6	5	4	3	2	1	0	
0	a	a	a	a	a	a	a	a	a	a	a	a	0
1	a	a	a	a	a	a	a	a	a	a	a	0	1
2	a	a	a	a	a	a	a	a	a	a	0	1	1
3	a	a	a	a	a	a	a	a	a	0	1	1	1
4	a	a	a	a	a	a	a	a	0	1	1	1	1
5	a	a	a	a	a	a	a	0	1	1	1	1	1
6	a	a	a	a	a	a	0	1	1	1	1	1	1
7	a	a	a	a	a	0	1	1	1	1	1	1	1
8	a	a	a	a	0	1	1	1	1	1	1	1	1
9	a	a	a	0	1	1	1	1	1	1	1	1	1
10	a	a	0	1	1	1	1	1	1	1	1	1	1
11	a	0	1	1	1	1	1	1	1	1	1	1	1
12	0	1	1	1	1	1	1	1	1	1	1	1	1

In Figure 7, a fixed width word for addressing having an address field with an

optional substitution indicator. As mentioned above, the substitution field has a variable size and will function to substitute an outside addressing source for a variable amount of address bits "a". The substitution occurs in place of the termination marker bit(s) "y" and continuation marker bit(s) "x".

The termination marker functions to inform the address decoding circuit where the substitution stops. The continuation marker pads the fixed width word.

If substitution is always to be used, then there is no need for an indicator. However, a substitution indicator allows optimal use of substitution.

ADDRESSING VARIABLE WIDTH DATA WITH A FIXED WIDTH WORD.

One embodiment of the present invention is for addressing a memory which can be accessed at its full width or in 2^n widths up to its full width (these smaller words are called partial words). It will be shown how the variable field encoding can be used to address this memory and to index those addresses into the memory.

To access a 64 x 32 bit Register file in widths of 32, 16, 8, 4, 2, and 1 bit requires different lengths of address. There are twice as many 16 bit locations as 32 bit locations and thirty-two times more 1 bit locations than 32 bit locations. Additionally, up to eight bits of this address can be substituted by an index register. Thus, a variable amount of information must be coded into a fixed number of microcode bits. One method would be to have a three bit field for the width and for the number of LSB's to be substituted and 12 bits for the address, giving a microcode word of 18 bits. However, a better method is to use a most significant justified variable length field to constrict the address its width can be defined and, thus, the width of the access can be defined. For example, a six bit address indicates a 32 bit access while a 12 bit address indicates a 1 bit access. This is illustrated in Table 1-10 where continuation marker is "0"; termination marker is "1". It can be seen how the variable width field constricts the address "a..a" so defining its width and so the access width. The general case of a fixed width word for addressing is shown in Figure 6.

Table 1-10: Variable width addressing

Data Width		A	9	8	7	6	5	4	3	2	1	0
1	1	a	a	a	a	a	a	a	a	a	a	a
2	0	1	a	a	a	a	a	a	a	a	a	a
4	0	0	1	a	a	a	a	a	a	a	a	a
8	0	0	0	1	a	a	a	a	a	a	a	a
16	0	0	0	0	1	a	a	a	a	a	a	a
32	0	0	0	0	0	1	a	a	a	a	a	a

To allow indexing of address locations portions of the addresses "a..a" can be substituted by an alternative value. The substitution portion (or field) of the address can be defined by a least significant bit justified variable length field (the continuation marker "1"; termination marker "0") that is super imposed on top of those shown in Table 1-10. Using an address of an eight bit word an example Table 1-11 shows how to define the number of the least significant bits to be substituted. The least significant bit added is the substitution indicator (marked "w"). The general case of a Fixed width word for substitution is shown in Figure 8.

Table 1-11: Address substitution

Bits to be substituted		A	9	8	7	6	5	4	3	2	1	0	w
0	0	0	0	1	a	a	a	a	a	a	a	a	0
1	0	0	0	1	a	a	a	a	a	a	a	0	1
2	0	0	0	1	a	a	a	a	a	a	0	1	1
3	0	0	0	1	a	a	a	a	a	0	1	1	1
4	0	0	0	1	a	a	a	a	0	1	1	1	1
5	0	0	0	1	a	a	a	0	1	1	1	1	1
6	0	0	0	1	a	a	0	1	1	1	1	1	1
7	0	0	0	1	a	0	1	1	1	1	1	1	1
8	0	0	0	1	0	1	1	1	1	1	1	1	1

In effect, the substitute code is superimposed on top of the address that is already coded.

From this coding, it can be seen that there are illegal addresses, most obviously 0x0000 and 0x3fff, and in this case a "0" must be in the bottom 9 bits to prevent substituting more than 8 bits and a "1" in the top 6 bits to specify an allowable access width. If one of these errors is detected the access is undefined, but the Register file contents will not be affected.

The apparatus for addressing and a method for accessing partial words in a Register file will be discussed below.

The conventional memory circuitry dictates that the memory must always be accessed at its full width. To achieve variable width accesses, a full (32 bit) width word is read. This full word is rotated until the partial word accessed is justified in the LSB. The upper parts of the word are extended to the full width and then output. Extending may encompass padding with zeros or ones, sign extending, using the sign bit of a sign-magnitude number as the new MSB or any similar conventional method. Extending is dependent on the mode of operation. When the partial word is input to be written back into the memory it is multiplexed back into the rotated full word, which is then rotated back and written into the array. Figure 3 shows these steps for the access of a 4 bit partial word in the fourth four bit word of the 32 bit word.

To access or read partial words, such as the highlighted four bit word in row "1" of Figure 3, the full width word must be rotated to place the partial word at the LSB, as shown in row "2". As shown in row "3", the four bit word is extended to create a full 32 bit word. This word can now be accessed.

A full width word that has been selected to be written back is truncated to the width of the original partial word which is multiplexed into the word shown in row "2" at the LSB position, this is shown in row "4". The resulting word is rotated back in its original significance in the read word, this is shown in row "5". This full word can now be written back into the Register file.

The list below summarizes the steps numbered in the Figure 3.

1. Full word read from memory

2. 12 bit rotate right puts partial word into the LSB
3. Extending to full word, then passed to output
4. The inputted partial word is multiplexed into rotated full word from (2)
5. 12 bit rotate left puts full word back to original to be written

The above accesses suggest the data flow structure of the memory that is shown in Figure 4. The numbers in the structure refer to the text above and to Figure 3.

The memory address must be decoded to control the above structure. It should be recognized that the MSB of any width of address is at the same significance with reference to the memory. The top six bits of a decoded address are a 32 bit word address, the remainder is a bit address. Therefore, the stage of decoding (in parallel with the substitution) is to decode the address width defining variable field by detecting the position of the most significant termination marker. This allows the address to be MSB justified (shifting in zeros at the LSB). The top six bits can be used directly as a 32 bit word row address the memory. The bottom five bits can be used to directly control both barrel shifters (as seen in Figure 4), because for example an original 32 bit address will always have a shift of 0b00000 (these having been shifted when the address was MSB justified), similarly a 16 bit address can have a shift of 0bx0000 i.e. 0 or 16 bit shift and a 1 bit address can have a shift of 0bxxxxx i.e. 0 to 31 bit shifts. The extender and input multiplexer are controller by the access width decode to mask out the output words and multiplex the input words to an appropriate significance respectively. The block diagram of the decode is shown in Figure 5. It can be seen that the decode of the two variable width fields for width and substitution can be done in parallel and independently.

Figure 8 represents an example of fixed width word 13 bits long for addressing variable width data and substitution as shown in the bottom two rows. For these examples an eight bit word would have been addressed at location 0b1101ssss, where "ssss" is substituted from another address source.

MICROCODABLE STATE MACHINE STRUCTURE

The substitution into a memory address and variable width accessing of a

memory have been brought together in the implementation of a microcodable state machine the structure of which is shown in Figure 1. The structure is one of a state machine controlling an arithmetic core by way of a wide word of control signals called a microcode instruction. The arithmetic core in turn passes status flags and some data to the state machine.

The state machine consists of a memory containing a list of the microcode instructions. As with conventional microcodable state machines, it is capable of either proceeding through the list of microcode instructions contiguously or any instructions can jump to any other. The jump address is in the form of Figure 7. The value substituted comes from the arithmetic core as shown in Figures 1 and 2. This allows the construction of "jump tables" within the microcode programs. Thus if a jump is made with 3 bits substituted, for example, there are a possible eight contiguous locations that may be jumped to dependent on the value from the arithmetic core, it has so become a programmable jump.

ARITHMETIC CORE

The arithmetic core, as shown in Figure 2, is composed of a memory called a Register file, an Arithmetic and Logic unit (ALU), an input port and output port. These components are connected by busing and multiplexers. As previously stated, these and the multiplexers defining their connections are entirely controlled by the microcode instruction issued by the state machine. The ALU and ports are conventional, but the Register file is a memory that allows variable width indexed accesses to it. The address to the Register file is coded directly into the microcode instruction.

The advantages of using this method of addressing to the Register file are firstly that many locations in an application do not need to be the full width of the memory (32 bits in this case). Whilst it will cause no effect on the operation of the device to use a full width location it is very wasteful of memory locations. Minimizing the number of memory locations used will minimize the space used by the memory, therefore minimize the capacitive loading in the Register file and so maximize the speed of the Register file. Secondly, the indexing combined with the variable width of memory accessing allows the stepping through of locations of variable width. In the

one bit case, this allows an elegant implementation of long division and multiplication.

In summary of the above, a procedure for addressing memory having the following steps is disclosed: providing a fixed width word having a predetermined fixed number of bits to be used for addressing variable width data, defining the fixed width word with a width defining field and an address field proving the width defining field with at least one bit to serve as the termination marker, defining the address field with a plurality of bits defining the address of the data, varying the size of bits in the address field in inverse relation to the size of the variable width data, varying the number of bits in the width defining field in direct relation to the size of the variable width data and maintaining a fixed width word for addressing variable width data while varying the width of the width defining field and the address field. In addition, a procedure for addressing memory having the following steps is disclosed: providing a fixed width word having a predetermined fixed number of bits to be used for addressing data, defining the fixed width word with an address field and a substitution field, defining the address field with a plurality of bits defining the address of the data, defining a variable width substitution field with a least one substitution bit, the substitution field has at least one bit to serve as a termination marker between the address field and the substitution field, using the substitution field to indicate substituted bits from a separate addressing source and maintaining a fixed width word for addressing variable width data while inversely varying the width of the address field and the width of the substitution field. In addition, a process for addressing variable width data in a memory having the following steps providing a memory having words of predetermined width and composed of partial words rotating the partial word to be accessed to a least significant bit justification, extending remaining part of the word so that the accessed word will be recognized as the partial word, restoring the remaining part of the word and rotating the word until the partial word is restored to its original position.

THE EMBODIMENTS OF THE INVENTION IN WHICH AN EXCLUSIVE PROPERTY OR PRIVILEGE IS CLAIMED ARE DEFINED AS FOLLOWS:

1. A procedure for addressing memory characterized by:

providing a fixed width word having a predetermined fixed number of bits to be used for addressing data;

defining the fixed width word with an address field and a substitution field;

defining the address field with a plurality of bits defining the address of the data;

defining a variable width substitution field with at least one substitution bit;

the substitution field has at least one bit to serve as a termination marker between the address field and the substitution field;

using the substitution field to indicate substituted bits from a separate addressing source; and

maintaining a fixed width word for addressing variable width data while inversely varying the width of the address field and the width of the substitution field.

SMART & BIGGAR

OTTAWA, CANADA

PATENT AGENTS

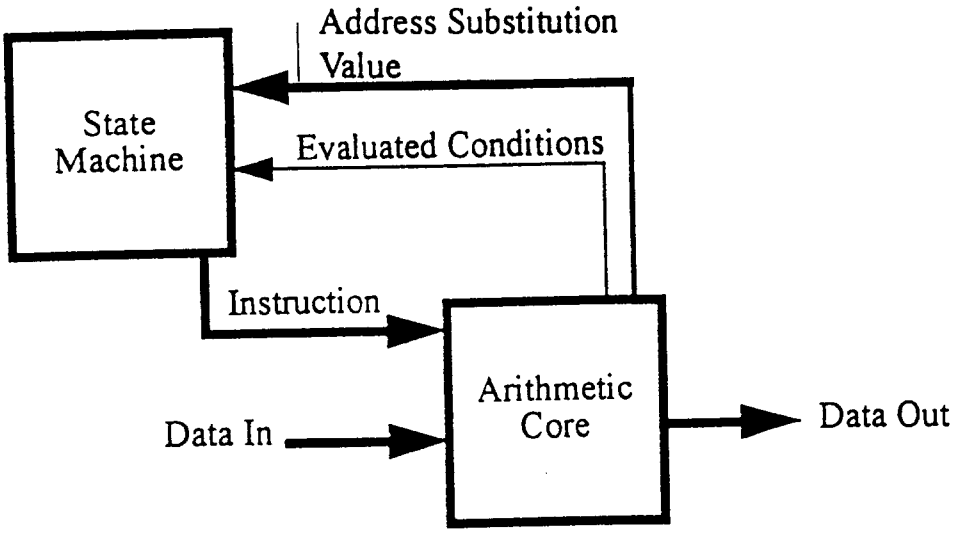


FIG. 1

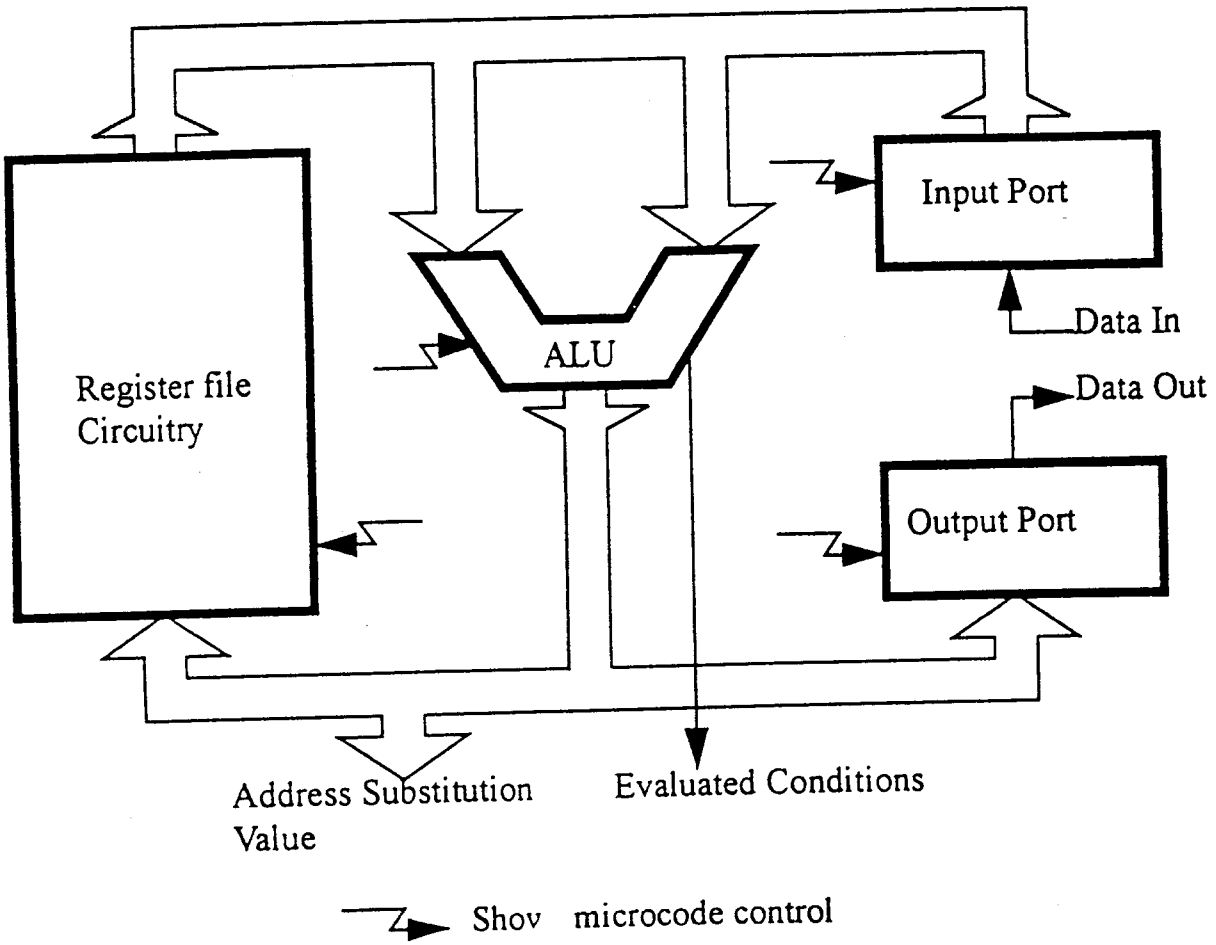


FIG. 2

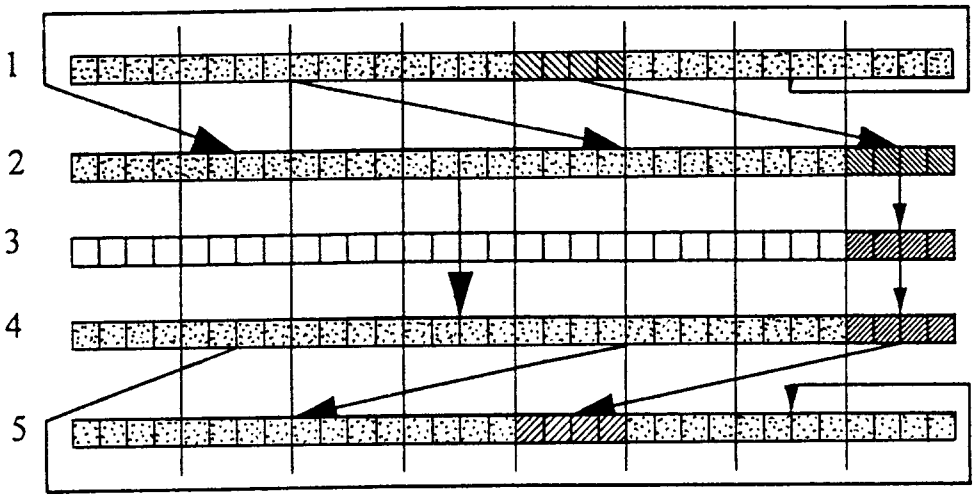


FIG.3

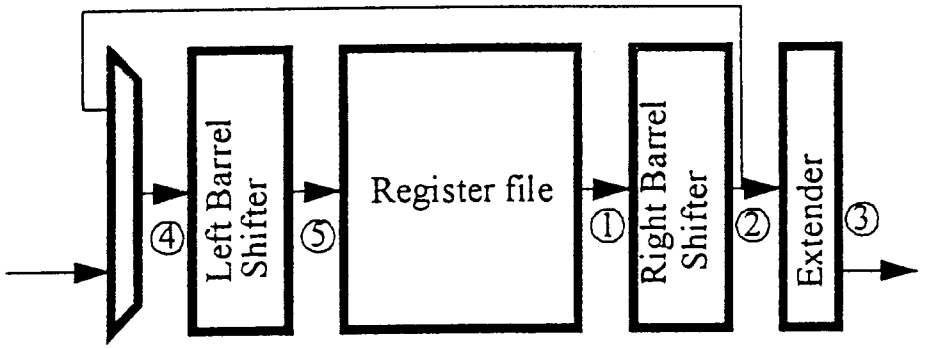


FIG.4

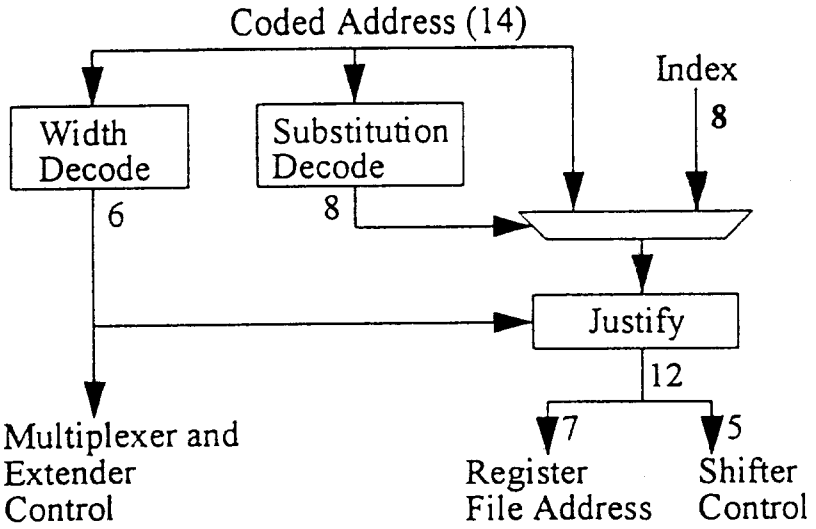


FIG.5

Fixed Width Word for Addressing		
Width Defining Field		Address Field
Continuation markers	Termination marker	
uu.....uu	vv.....vv	aa.....aa

FIG.6

Fixed Width Word for Addressing			
Address Field			Substitution Indicator
	Substitution Field		
	Termination marker	Continuation markers	
aa.....aa	yy.....yy	xx.....xx	ww.....ww

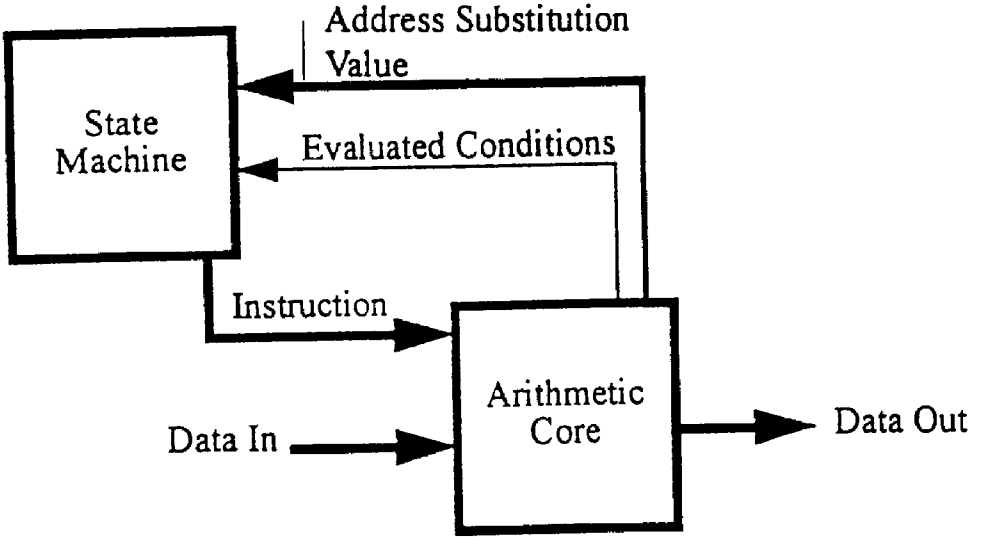
FIG.7

Fixed Width Word for Addressing					
Width Defining Field		Address Field			Substitution Indicator
			Substitution Field		
Continuation markers	Termination marker		Termination marker	Continuation markers	
uu.....uu	vv.....vv	aa.....aa	yy.....yy	xx.....xx	ww.....ww
000	1	1101	1	000	0
111	0	1101	0	111	1

FIG.8

Fixed Width Word with Variable Fields		
First Field	Second Field	
	Termination marker	Continuation markers
aa.....aa	yy.....yy	xx.....xx

FIG.9


Address Substitution
Value
State
Machine
Evaluated Conditions
Instruction
Arithmetic
Core
Data In
Data Out