US 20020032845A1

(19) **United States**

(12) **Patent Application Publication**

(10) **Pub. No.: US 2002/0032845 A1**

(43) **Pub. Date: Mar. 14, 2002**

MCGREGOR et al.

(54) **ARRAY INDEXING WITH SEQUENTIAL ADDRESS GENARATOR FOR A MULTI-DIMENSIONAL ARRAY HAVING FIXED ADDRESS INDICES**

(76) Inventors: **DOUGLAS ROBERT MCGREGOR**, GLASGOW (GB); **WILLIAM PAUL COCKSHOTT**, GLASGOW (GB)

Correspondence Address:
ALSTON & BIRD LLP
BANK OF AMERICA PLAZA
101 SOUTH TRYON STREET, SUITE 4000
CHARLOTTE, NC 28280-4000 (US)

(*) Notice: This is a publication of a continued prosecution application (CPA) filed under 37 CFR 1.53(d).

(21) Appl. No.: **09/142,156**

(22) PCT Filed: **Mar. 19, 1997**

(86) PCT No.: **PCT/GB97/00764**

(30) **Foreign Application Priority Data**

Mar. 20, 1996 (GB) 9605853.2

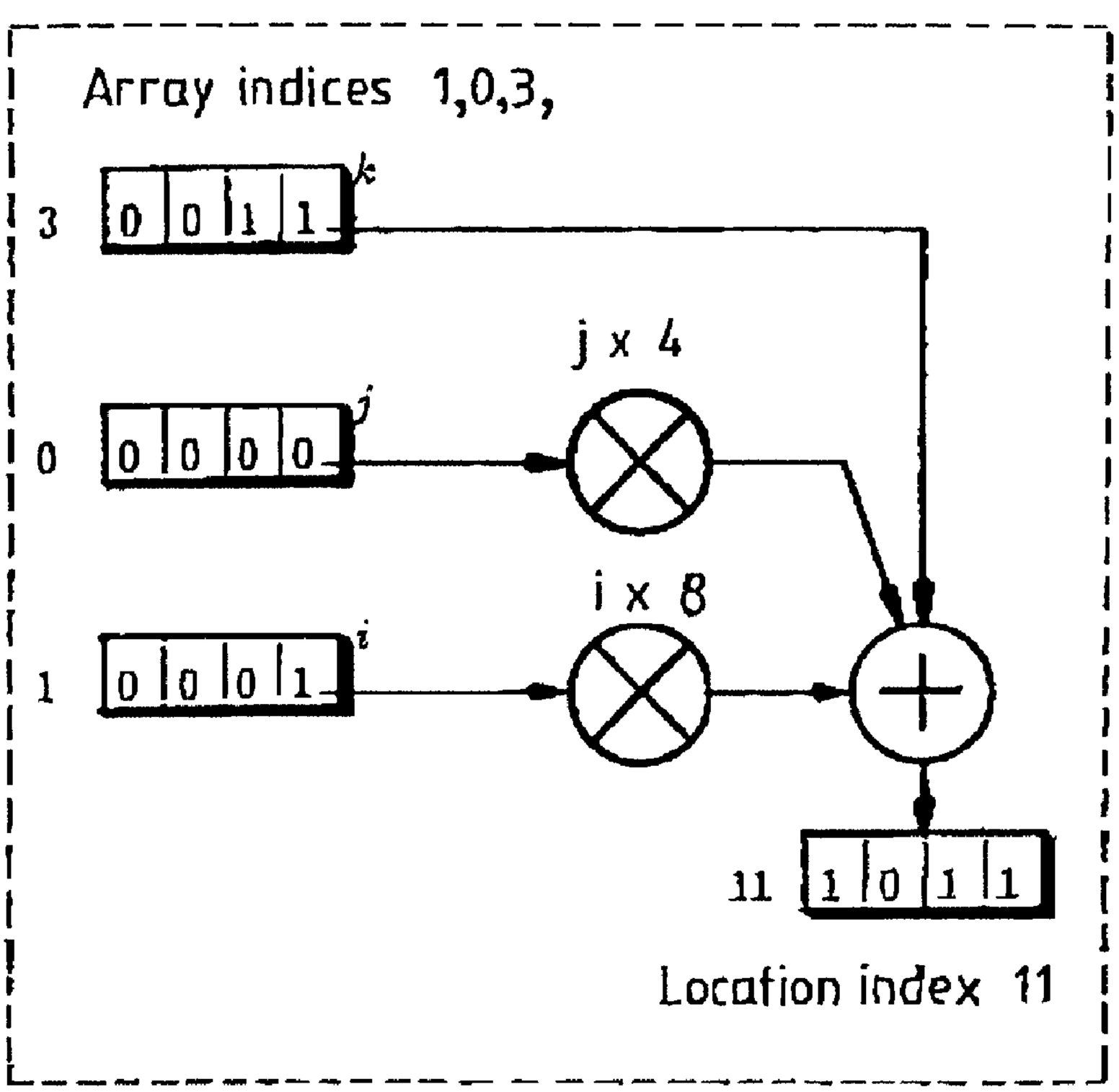
Publication Classification

(51) **Int. Cl.**[7] **G06F 12/00**

(52) **U.S. Cl.** **711/202**; 711/217; 711/209; 711/204

(57) **ABSTRACT**

A method and apparatus for sequentially generating a set of addresses, defined over a plurality of indices, for a multi-dimensional array stored in a memory for the condition where at least one of the address indices is fixed, is performed by simple addition, OR-ing and AND-ing. An accumulator or counter initially holds an arbitrary binary value composed of a set of binary indices corresponding to the address indices. This binary value is logically OR-ed with a first mask value having binary indices selected in value in relation to the fixed address indices. The resultant is logically AND-ed with a second mask value having binary indices selected in value in relation to the fixed address indices, and this operation produces a first address of the set. The same resultant is incremented and the incremented value is delivered to the accumulator for the cycle to be repeated.



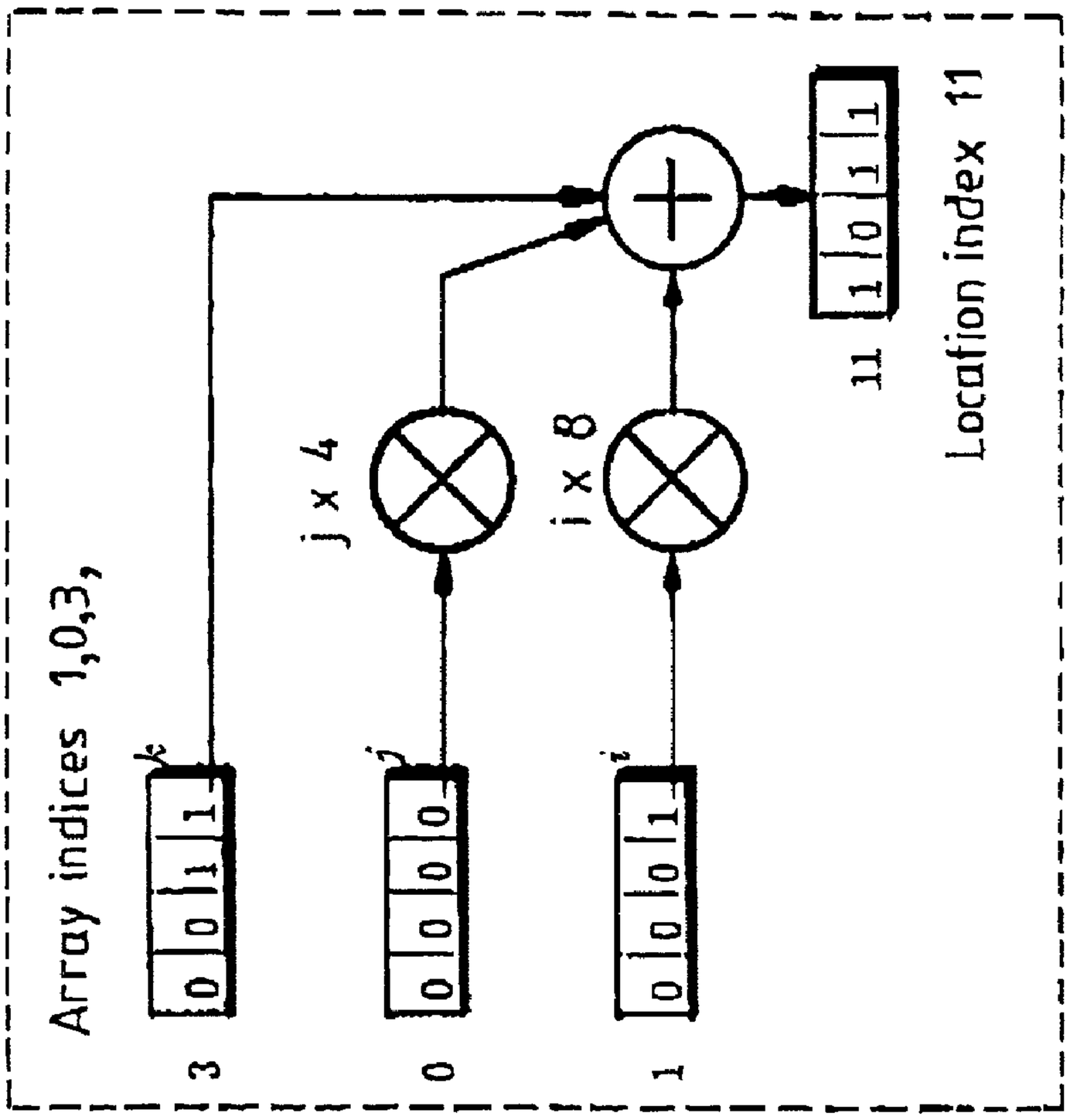
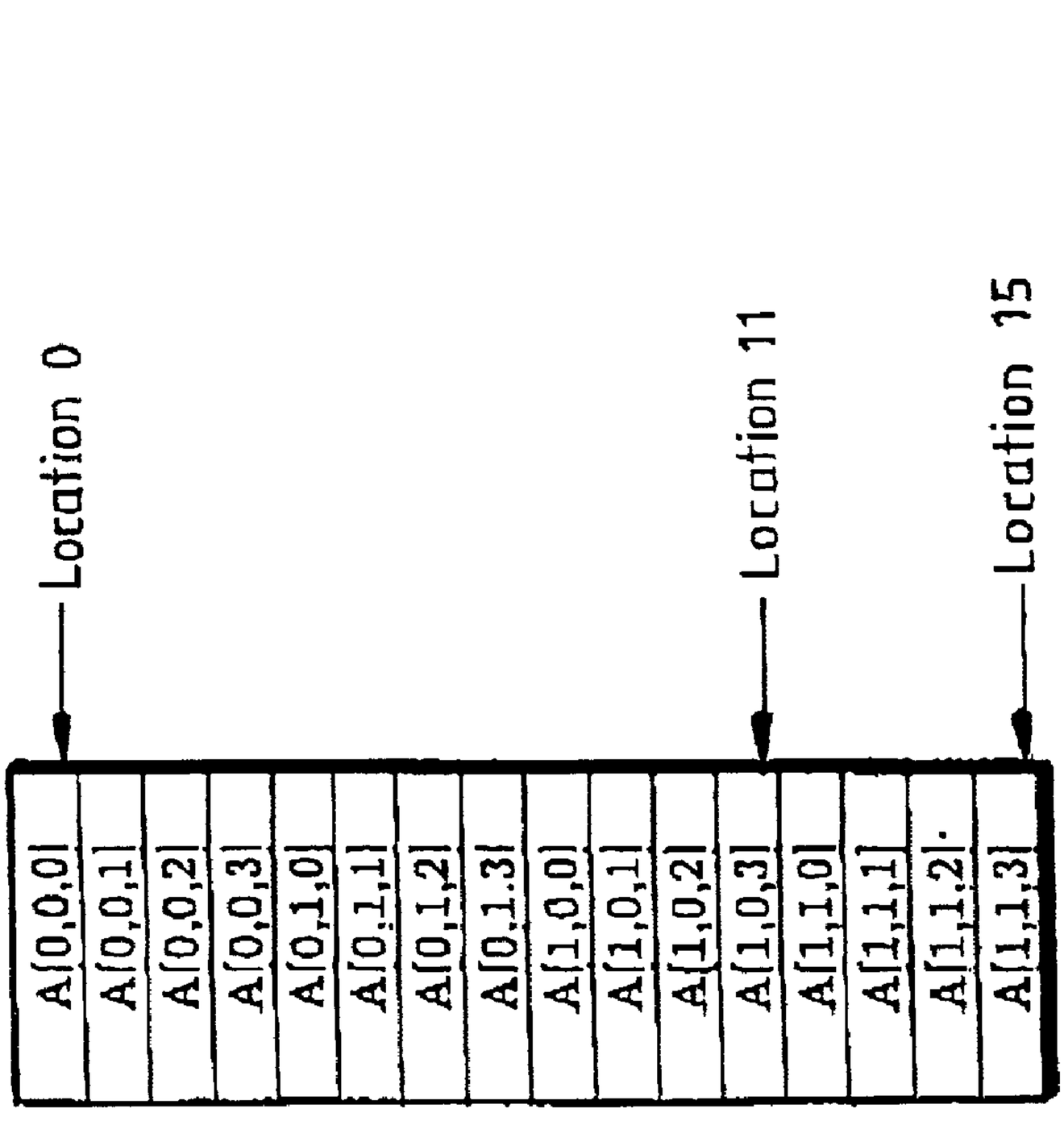


FIG. 2



The 3 diemensional array A[0..1,0..1,0..3]

FIG. 1

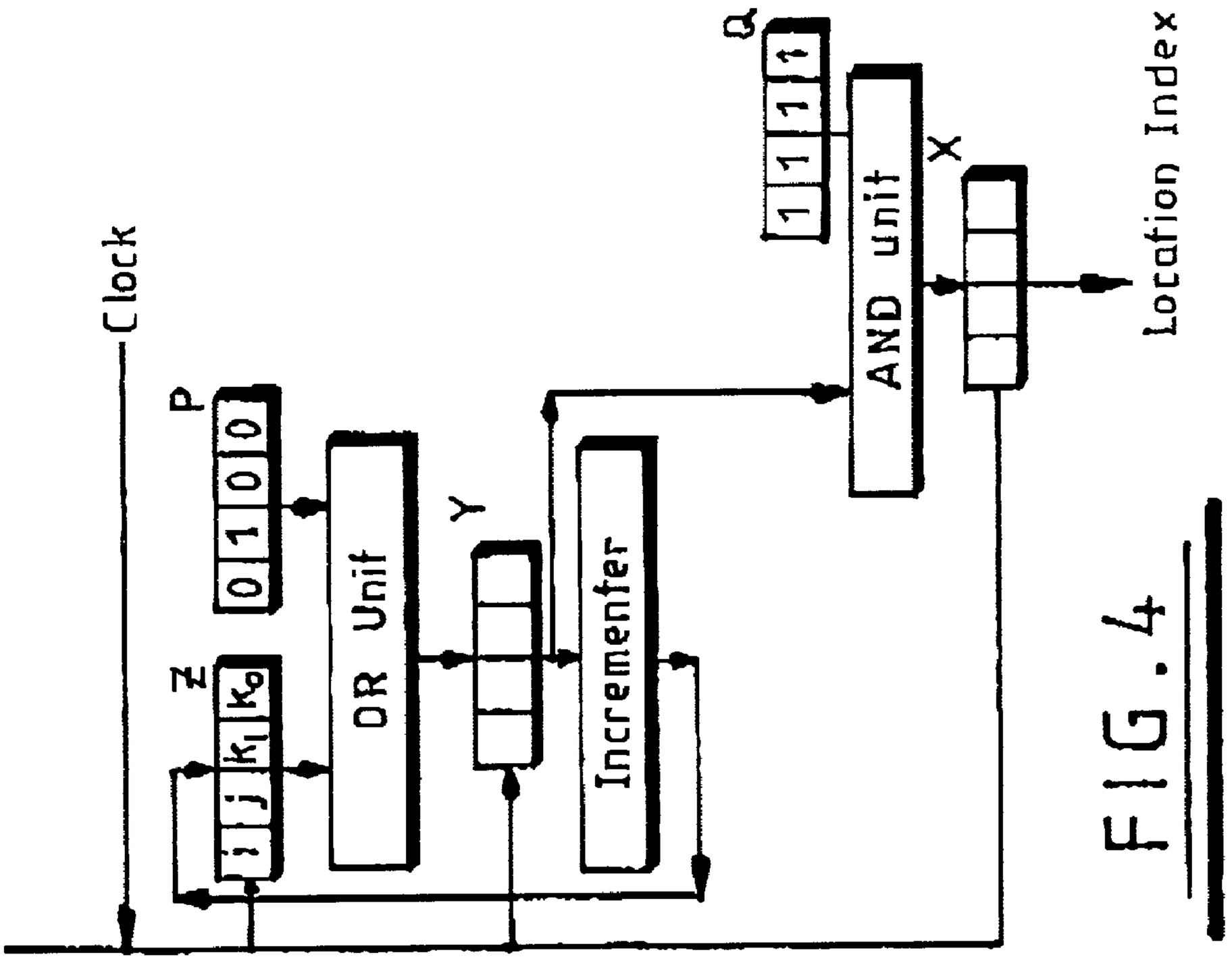

Clock
P
0 1 0 0
Z
i j k1 k0
OR Unit
Y
Incrementer
q
1 1 1 1
unit
AND
X
Location Index

FIG. 4

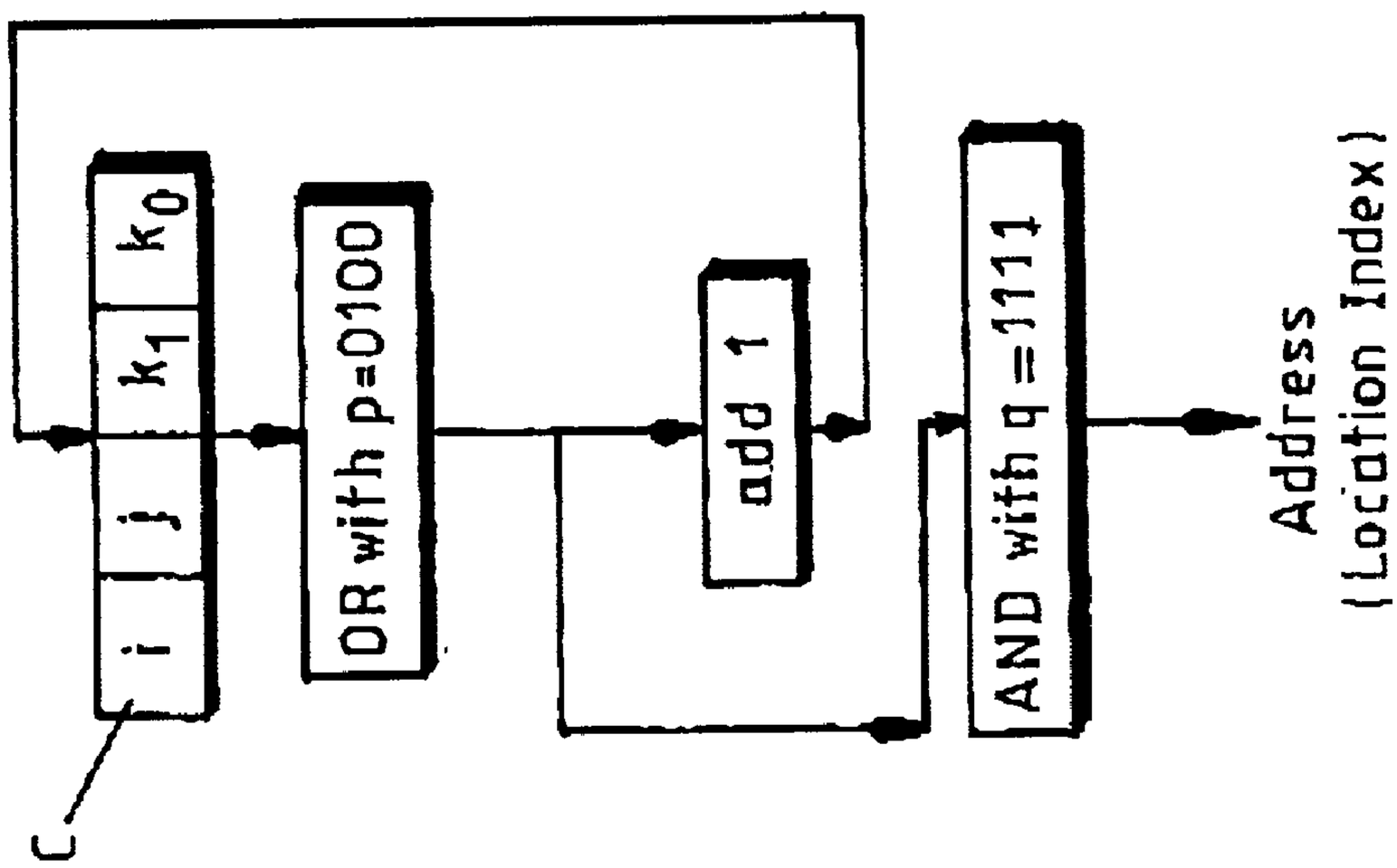

i j k1 k0
OR with p=0100
add 1
AND with q=1111
Address
(Location Index)

FIG. 3

ARRAY INDEXING WITH SEQUENTIAL ADDRESS GENARATOR FOR A MULTI-DIMENSIONAL ARRAY HAVING FIXED ADDRESS INDICES

[0001] The present invention relates to array indexing in a computer and in particular to a method of and apparatus for sequentially generating a set of addresses, defined over a plurality of indices, for a multi-dimensional array stored in a memory, wherein at least one of the address indices is fixed.

[0002] In accessing data stored in a digital memory, it is often desirable or necessary to step through the entries of a multi-dimensional array, holding one or more indices of the address constant whilst doing so. To illustrate this, reference is made to **FIG. 1** which shows a typical example of how a multi-dimensional array A[i,j,k] where i=0 to 1, j=0 to 1, k=0 to 3 may be stored in a computer's memory. The array is logically a sequence of elements each of which is identified with a unique combination of subscripts or indices (i,j,k) but is physically stored in a one dimensional array of locations in the memory. In order to enable hardware to access the one dimensional memory array it is necessary to translate logical addresses composed of the appropriate indices (i,j,k) to location addresses which can directly access the one dimensional array.

[0003] It is therefore the case that associated with each set of indices in the multidimensional array there will be a number which, when used as an index to the underlying one dimensional array, will define the corresponding position in the one dimensional array. Thus, in the example shown in **FIG. 1**, element A[**1,0,3**] will be in decimal location **11** (i.e. 1011 in binary) in the underlying one dimensional array.

[0004] In the above example, location **11** is typically obtained from the array indices [**1,0,3**] as shown in **FIG. 2**. The indices [**1,0,3**] i.e. i,j,k are held in respective distinct registers (i,j,k) and are then multiplied by appropriate constants (8, 4 and 1). The results are then added together to form the location number (1011) in binary.

[0005] Consider now that one wishes to step through the array A[i,j,k], holding the value of j constant at 1 but going through all legitimate combinations of i and k. This involves accessing the set of array elements:

[0006] A[**0,1,0**], A[**0,1,1**], A[**0,1,2**], A[**0,1,3**],

[0007] A[**1,1,0**], A[**1,1,1**], A[**1,1,2**], A[**1,1,3**]

[0008] and the corresponding locations: 4, 5, 6, 7, 12, 13, 14, 15

[0009] The normal process by which this would be done would be to:

[0010] 1. reserve three computer registers for i, j, k;

[0011] 2. initialise them to the values i=0, j=1, k=0;

[0012] 3. set up a pair of nested loops in a computer program so that

[0013] (a) the outer loop increments i through all its allowed values

[0014] (b) the inner loop increments k through all its allowed values and

[0015] (c) the kernel of the inner loop uses the process shown in **FIG. 2** to compute the location numbers from the indices of each array element.

[0016] The disadvantage of this process is that it involves multiplication or shift operations and nested loops, all of which are relatively slow operations to perform using computer hardware.

[0017] It is an object of the present invention to overcome or at least mitigate the above mentioned disadvantages of array address generating methods.

[0018] According to a first aspect of the present invention there is provided a method of sequentially generating a set of addresses, defined over a plurality of indices, for a multi-dimensional array stored in a memory, wherein at least one of the address indices is fixed at an actual value, the method comprising the steps of:

[0019] (a) providing a first binary address having address indices of arbitrary value;

[0020] (b) replacing all of the bit values of the or each fixed index in the first binary address with 1's;

[0021] (c) incrementing the resulting binary value, wherein any carry value(s) generated propagate(s) across the or each fixed index;

[0022] (d) replacing the bit values of the or each fixed index in the resulting binary value with the actual values to which each fixed index is set to provide a first address for the set; and

[0023] (e) repeating steps (b) to (d), using the preceding result of step (c) as a new first binary address value, is until the required address set has been generated.

[0024] In an embodiment of the above first aspect of the invention, the method comprises the steps of:

[0025] (i) providing an accumulator and storing in the accumulator a binary value composed of a set of binary indices corresponding to the address indices;

[0026] (ii) performing a logical OR operation between the binary value stored in the accumulator and a first mask binary value, where the first mask valve is also composed of a set of binary indices with the or each index, corresponding to the or each fixed address index, being composed entirely of 1's and the other indices being composed entirely of 0's;

[0027] (iii) performing a logical AND operation between the result of the OR operation in step (ii) and a second mask binary value composed of a set of binary indices, where the or each index, corresponding to the or each fixed address index, contains the corresponding binary value of the fixed address index, and the other indices are composed entirely of 1's, wherein the result of the AND operation provides a first address for the set;

[0028] (iv) incrementing the result of the OR operation performed in step (ii) and storing the result in the accumulator; and

[0029] (v) repeating steps (ii) to (iv) until the required address set has been generated.

[0030] The present invention makes it possible, by suitable use of mask bits, to so arrange the inputs to a conventional computer adder that the operations required to step through a multi-dimensional array can be performed by simple addition, ORing and ANDing

[0031] The binary value initially stored in the accumulator may be any arbitrary value although it would normally be initialised to zero.

[0032] According to a second aspect of the present invention there is provided a digital computer memory comprising at least one digital data storage device and address generating means arranged to generate addresses according to the method of the above first aspect of the present invention for the purpose of accessing data stored in the or each data storage device.

[0033] According to a third aspect of the present invention there is provided apparatus for carrying out the method of the above first aspect of the invention and comprising:

[0034] a first register arranged to provide said accumulator;

[0035] a second register arranged to store the first mask binary value;

[0036] an OR logic unit coupled to said first and second registers for performing the OR operation of step (ii);

[0037] a third register coupled to the OR unit and arranged to store the result of said OR operation;

[0038] an incrementing unit coupled to said first and third registers and arranged to perform step (iv);

[0039] a fourth register arranged to store said second mask value; and

[0040] an AND logic unit coupled to said third and fourth registers and arranged to perform the AND operation of step (iii).

[0041] For a better understanding of the present invention and in order to show how the same may be carried into effect reference will now be made, by way of example, to the accompanying drawings, in which:

[0042] **FIG. 1** illustrates the mapping of a logical 3-dimensional array onto a physical 1-dimensional memory space;

[0043] **FIG. 2** shows schematically a conventional process for generating the location in the 1-dimensional memory space of **FIG. 1** corresponding to a 3-dimensional array address;

[0044] **FIG. 3** shows a flow diagram illustrating the generation of multi-dimensional array addresses according to an embodiment of the invention;

[0045] **FIG. 4** shows schematically a system for carrying out the process of **FIG. 3**.

[0046] Considering the array A shown in **FIG. 1**, instead of maintaining three registers (as per **FIG. 2**) corresponding to the dimensions of the array to be traversed, an embodiment of the present invention requires (as per **FIG. 3**) only a single counting register C and two mask words, p and q, The following steps are then followed:

[0047] **1**. a boolean OR operation is performed between the consents of register C and the mask word p;

[0048] **2**. the result of the first step is then incremented and returned to register C;

[0049] **3**. the mask q is 'ANDed' with the result of the first step and the result used as the location (in binary) of the current array element.

[0050] For this technique to work, the mask words p, q must be prepared so that p holds 1's in the positions corresponding to the dimensions of the array that are being held fixed and 0's elsewhere, whilst the mask q contains:

[0051] 1's in the positions corresponding to the dimensions of the array that are being stepped through;

[0052] a pattern of 1's and 0's in the positions corresponding to the dimensions of the array that are fixed, so that the patterns of 1's and 0's correspond to the appropriately multiplied constant indices of these dimensions;

[0053] the value 0 in all other bit positions.

[0054] The effect of the 'ORing' with p is to create a number with strings of 1's between the fields that correspond to the dimensions being stepped through, These strings of 1's have the effect of allowing carry bits to propagate through them to the fields in the words corresponding to the next varying dimension.

[0055] As is illustrated in **FIG. 3** the register C is a four-bit register with the two least significant bits holding index k (where k=0 to 3) and the two most significant bits holding indices i and j (where i, j=0 to 1) and it is required to step through all possible addresses with j held constant at 1. The register C is initialised to 0000, or any other suitable value, before commencing address generation by applying steps **1**, **2** and **3** above, whilst the mask words p and q are set to 0100 and 1111 respectively.

[0056] One possible implementation of this technique, suitable for high speed computers, is shown in **FIG. 4**. The apparatus contains three registers X,Y,Z that are loaded with new data every clock cycle. For reasons of simplicity all registers are shown having 4 bits (2 bits allocated to the first, least significant index k, and 1 bit allocated to each of the other two indices i, j), but the technique applies equally well to longer or shorter registers. During each clock cycle, the apparatus will store in register X the address (in binary) of the next location in the 3-dimensional array to be accessed.

[0057] Two auxiliary registers P, Q are loaded at the start of the process with the masks p, q described above. Each cycle, register Y is loaded with the result of 'ORing' the contents of registers Z and P, and register Z is subsequently loaded with [1+Z] Register X is loaded with the result of 'ANDing' the contents of registers Y and Q. Thus, without any circuitry more complex than an incrementer, a new array location (in binary) can be computed each cycle.

[0058] An alternative software implementation involves executing a simple loop on a conventional computer processor such as is set out, for example, below. Assume in what follows that register SI holds the mask p, register DX holds the mask q and that register CX holds a count of the number of elements to visit.

```
start: and AX, DX       ; location now in AX
       call uselocation ; use the location to fetch and
                          operate on the array element
       mov AX, BX       ; recover previous value
                          with propagate bits
       inc AX           ; increment propagating the
                          carries
       or AX, SI        ; set the propagate bits again
       mov BX, AX       ; save in BX
       loop start       ; decrement CX and jump to start
                          if nonzero
```

[0059] Again the loop requires no slow multiply or shift instructions, and only a single loop is required.

[0060] The embodiment described above is generally applicable to arrays having array indices each of which extends from 0 to 2^{n-1}. This enables the generation of a carry bit, for propagation to the next variable index, when the maximum value in the preceding variable index is exceeded.

1. A method of sequentially generating a set of addresses, defined over a plurality of indices, for a multi-dimensional array stored in a memory, wherein at least one of the address indices is fixed at an actual value, the method comprising the steps of:

(a) providing a first binary address having address indices of arbitrary value;

(b) replacing all of the bit values of the or each fixed index in the first binary address with 1's;

(c) incrementing the resulting binary value, wherein any carry values) generated propagate(s) across the or each fixed index;

(d) replacing the bit values of the or each fixed index in the resulting binary value with the actual values to which each fixed index is set to provide a first address for the set; and

(e) repeating steps (b) to (d), using the preceding result of step (c) as a new first binary address value, until the required address set has been generated.

2. A method of sequentially generating a set of addresses, defined over a plurality of indices, for a multi-dimensional array stored in a memory, wherein at least one of the address indices is fixed at an actual value. The method comprising the steps of:

(i) providing an accumulator and storing in the accumulator a binary value composed of a set of binary indices corresponding to the address indices;

(ii) performing a logical OR operation between the binary value stored in the accumulator and a first mask binary value, where the first mask value is also composed of a set of binary indices with the or each index, corresponding to the or each fixed address index, being composed entirely of 1's and the other indices being composed entirely of 0's;

(iii) performing a logical AND operation between the result of the OR operation in step (ii) and a second mask binary value composed of a set of binary indices, where the or each index, corresponding to the or each fixed address index, contains the corresponding binary value of the fixed address index, and the other indices are composed entirely of 1's, wherein the result of the AND operation provides a first address for the set;

(iv) incrementing the result of the OR operation performed in step (ii) and storing the result in the accumulator; and

(v) repeating steps (ii) to (iv) until the required address set has been generated.

3. A method as claimed in claim 2, wherein the binary value initially stored in the accumulator is any arbitrary value.

4. A method as claimed in claim 3, wherein the arbitrary value is zero.

5. Apparatus for carrying out the method of any one of claims **2-4**, said apparatus comprising:

a first register arranged to provide said accumulator;

a second register arranged to store the first mask binary valuer;

an OR logic unit coupled to said first and second registers for performing the OR operation of step (ii);

a third register coupled to the OR unit and arranged to store the result of said OR operation;

an incrementing unit coupled to said first and third registers and arranged to perform step (iv);

a fourth register arranged to store said second mask value; and

an AND logic unit coupled to said third and fourth registers and arranged to perform the AND operation of step (iii).

6. A digital computer memory comprising at least one digital data storage device and address generating means arranged to generate addresses according to the method of claim 1 for the purpose of accessing data stored in the or each data storage device.

* * * * *