GB 2292822A

(12) UK Patent Application (19) GB (11) 2 292 822 (13) A

(43) Date of A Publication 06.03.1996

(21) Application No 9513272.6

(22) Date of Filing 29.06.1995

(30) Priority Data
(31) 08298861 (32) 31.08.1994 (33) US

(51) INT CL[6]
G06F 12/08

(52) UK CL (Edition O)
G4A AMC

(56) Documents Cited
EP 0466265 A1 EP 0284751 A2 WO 93/09497 A2
US 4905141 A

(58) Field of Search
UK CL (Edition N) G4A AMC
INT CL[6] G06F 12/08
ONLINE : WPI, INSPEC

(71) Applicant(s)
Hewlett-Packard Company

(Incorporated in USA - California)

3000 Hanover Street, Palo Alto, California 94304, United States of America

(72) Inventor(s)
Rajendra Kumar
Paul G Emerson

(74) Agent and/or Address for Service
Williams,Powell & Associates
34 Tavistock Street, LONDON, WC2E 7PB, United Kingdom

(54) Partitioned cache memory

(57) A cache memory (10) is partitioned during write operations into several regions (12, 14) in a programmable or adaptive manner. Each region is used to store a different type of data. The partitioning of the cache memory has no effect during read operations. The partitioning can be achieved dynamically for unproved optimization, being particularly suited for an instruction cache wherein one of the regions stores sequential instructions and another region stores branch target instructions. It is also useful for a data cache.

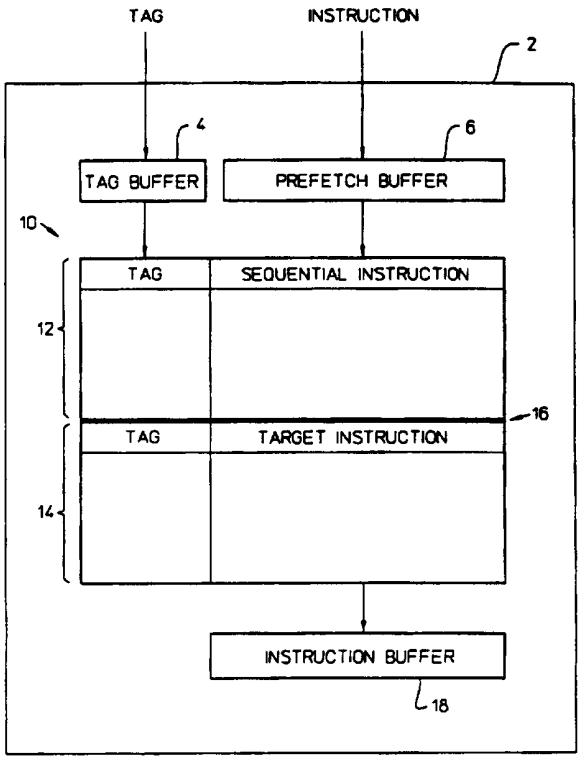

FIG. 1

GB 2 292 822 A

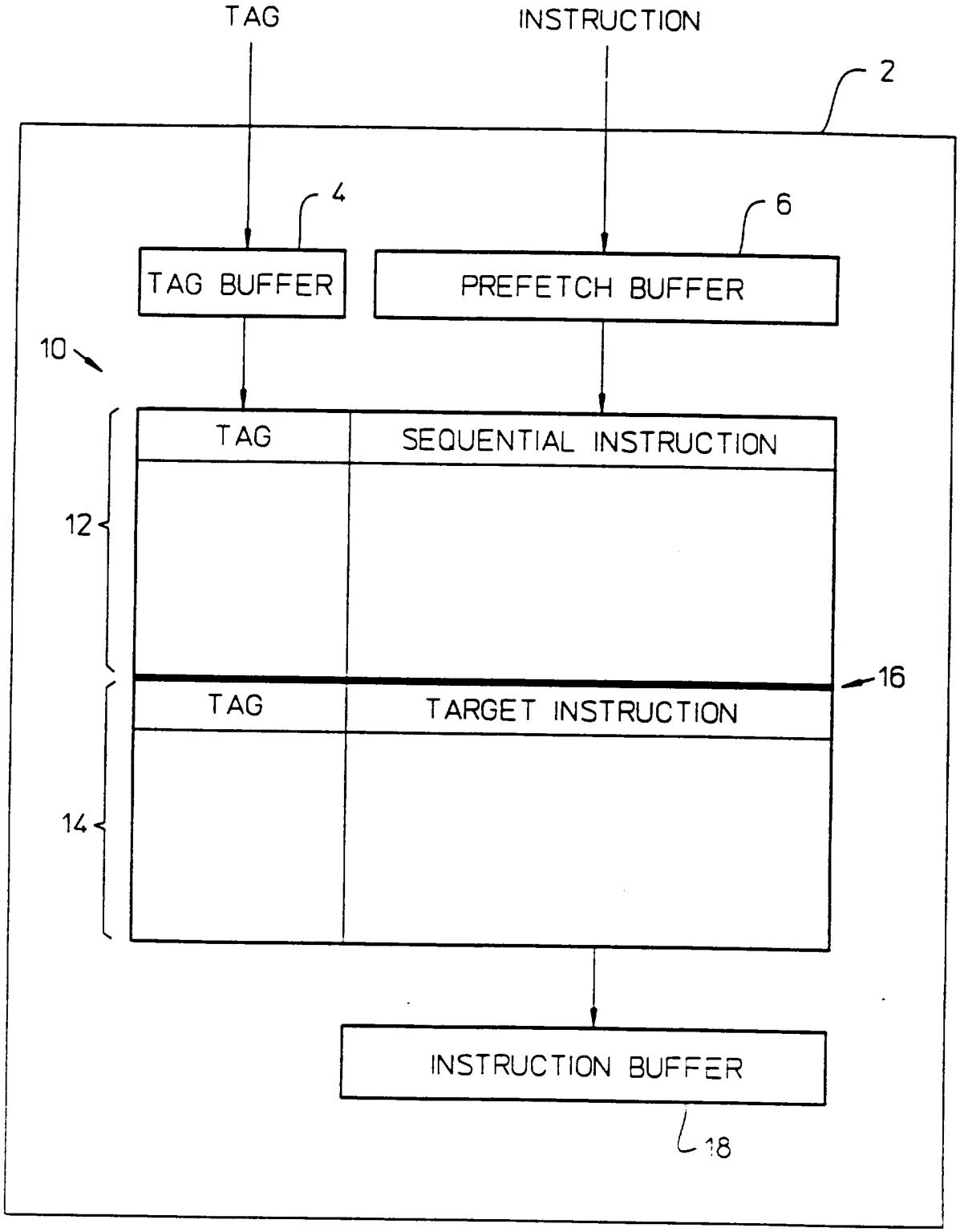
TAG
INSTRUCTION
2
4
6
TAG BUFFER
PREFETCH BUFFER
10
TAG
SEQUENTIAL INSTRUCTION
12
16
TAG
TARGET INSTRUCTION
14
INSTRUCTION BUFFER
18

FIG. 1

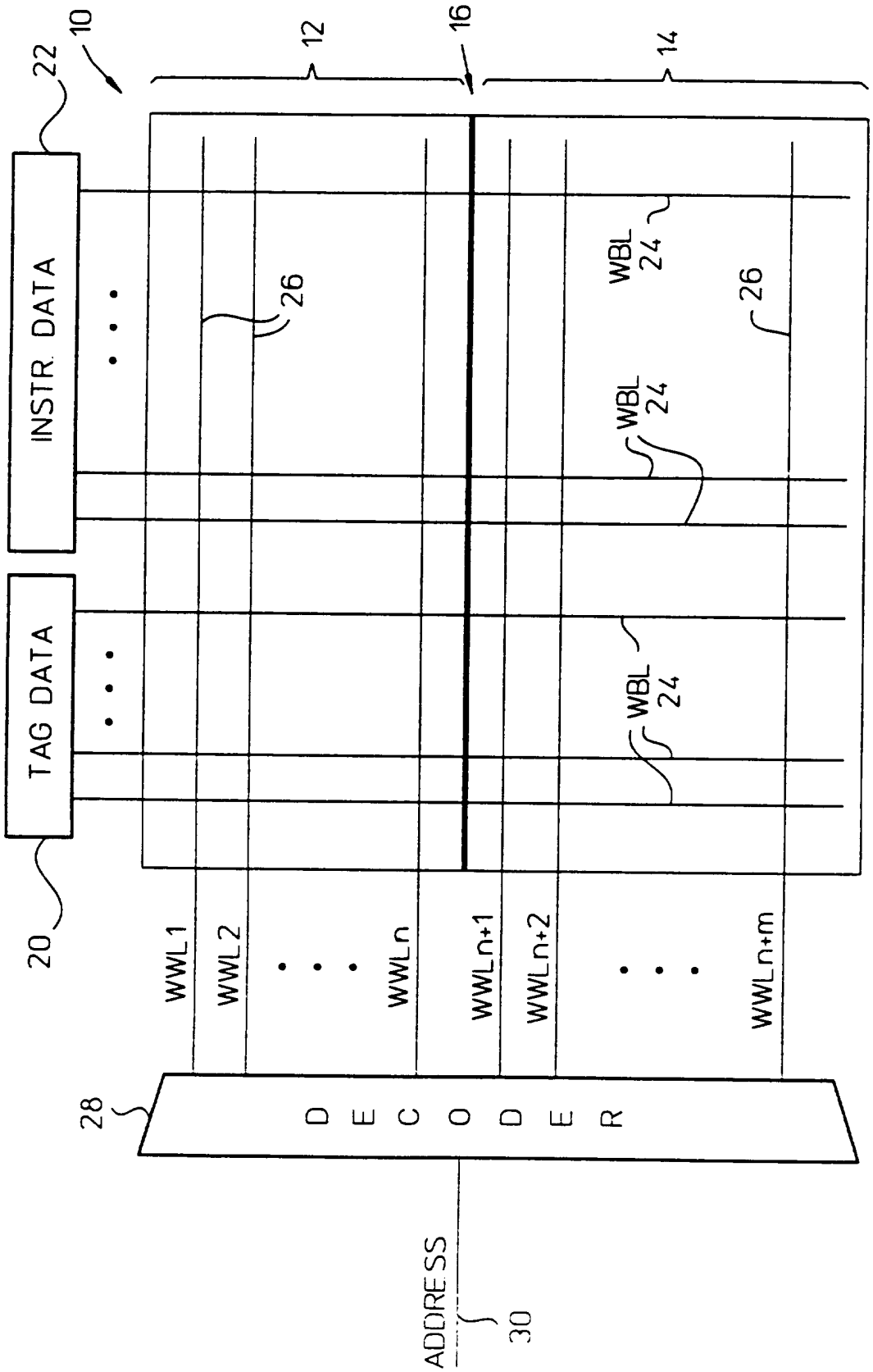
10
12
14
16
22
INSTR. DATA
TAG DATA
20
26
WBL
24
WWL1
WWL2
WWLn
WWLn+1
WWLn+2
WWLn+m
28
DECODER
ADDRESS
30

FIG. 2

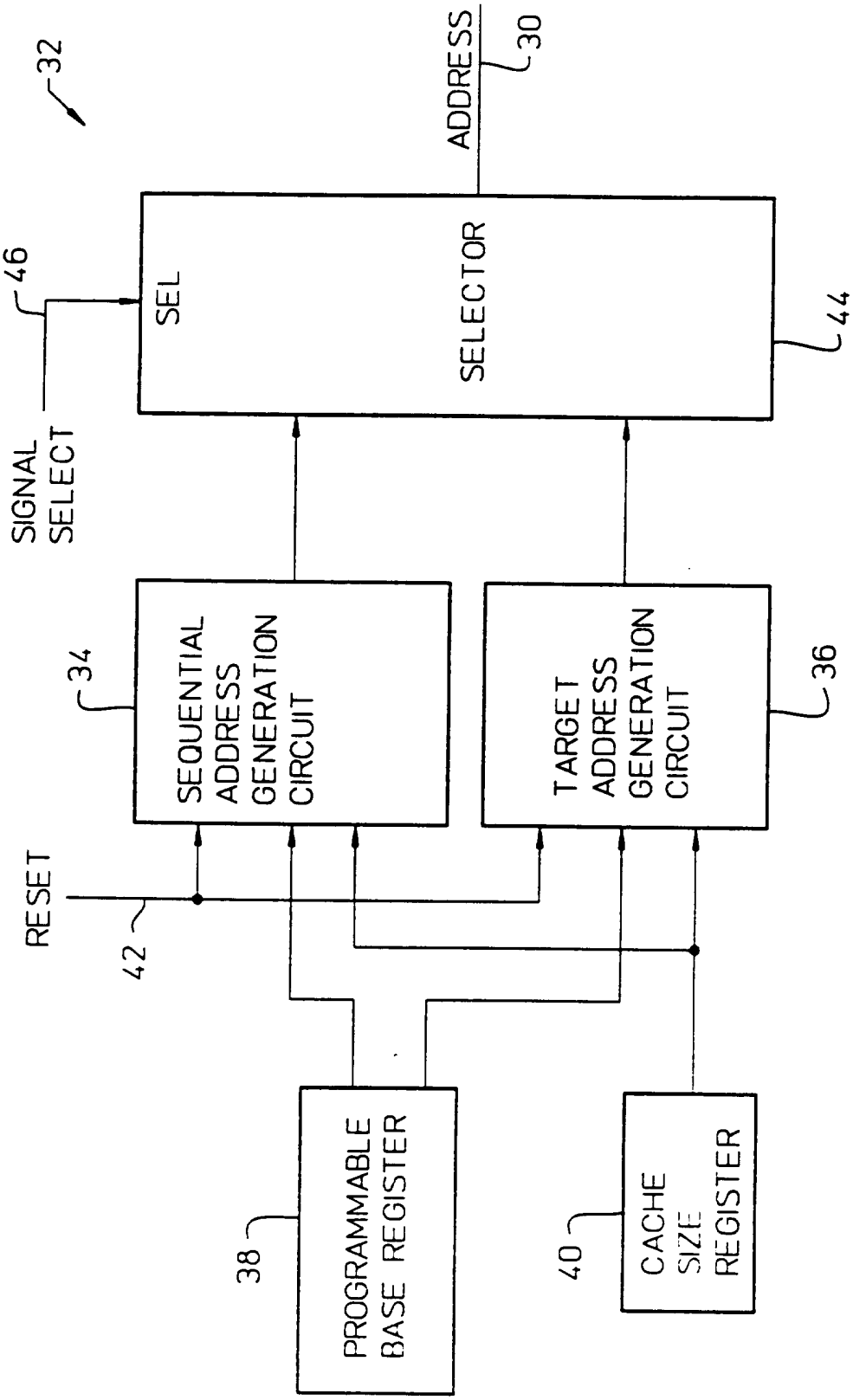
32
ADDRESS
30
46
SEL
SELECTOR
44
SIGNAL
SELECT
34
SEQUENTIAL
ADDRESS
GENERATION
CIRCUIT
36
TARGET
ADDRESS
GENERATION
CIRCUIT
RESET
42
38
PROGRAMMABLE
BASE REGISTER
40
CACHE
SIZE
REGISTER

FIG. 3

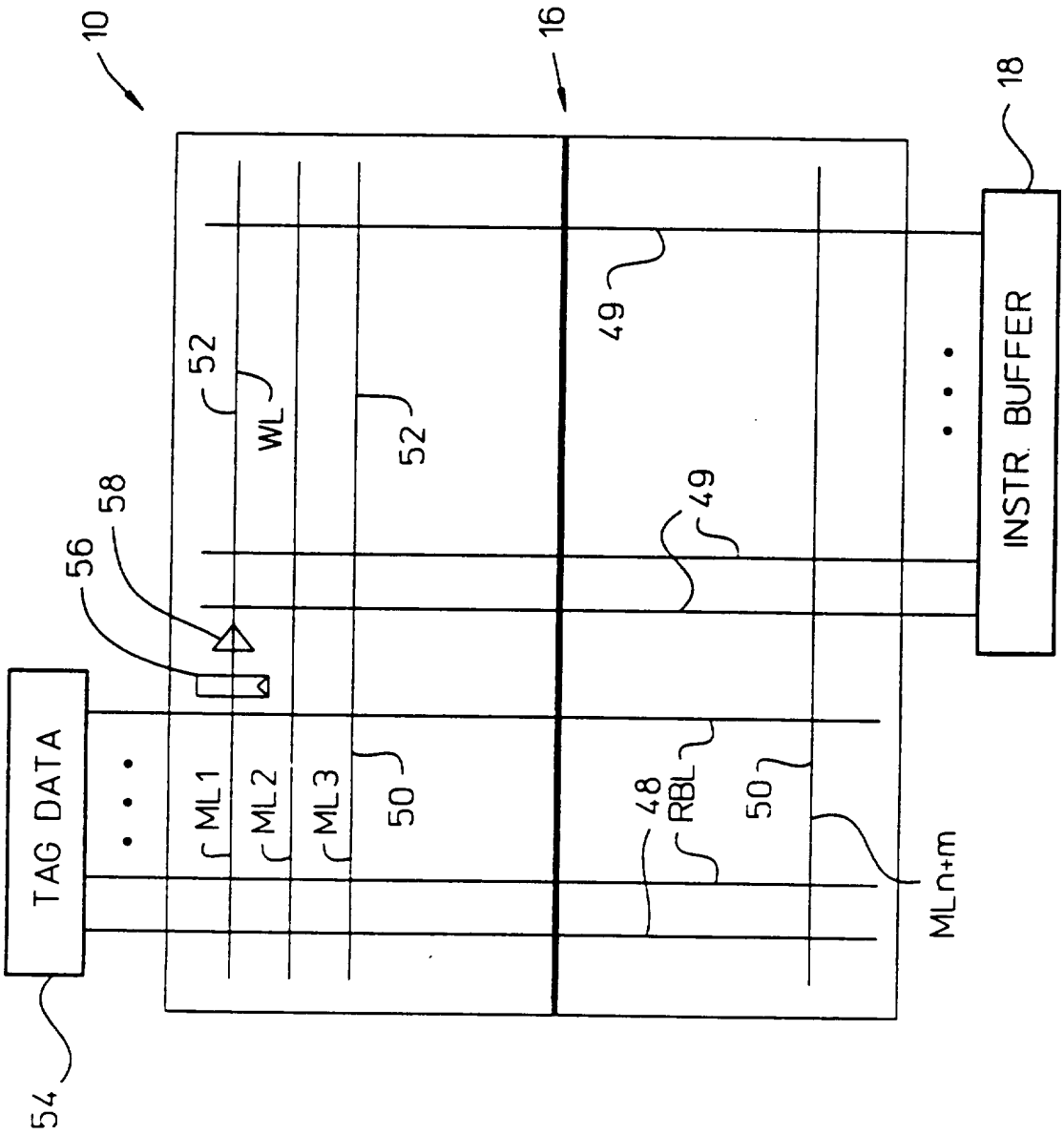
10
16
18
TAG DATA
INSTR. BUFFER
ML1
ML2
ML3
WL
RBL
MLn+m
48
49
50
52
54
56
58

FIG. 4

INSTRUCTION CACHE

The present invention relates to an instruction cache and, more particularly, to a cache memory and to a method of operating a cache.

Most microprocessors have on-chip instruction caches to facilitate faster access to data held in memory. Typically, these caches are relatively small in size due to their area requirement and also due to cycle time constraints. Consequently, it is important that these caches be used efficiently and effectively; otherwise, performance of the microprocessor suffers.

Besides a processing system, a modern microprocessor typically has a data cache and an instruction cache. The data cache stores data which is likely to be imminently needed by the processing system (e.g., functional units or central processing unit). The instruction cache stores instructions which have been prefetched prior to their probable execution. Instruction caches are used to store two different types of instructions, namely sequential instructions and branch target instructions. Sequential instructions make up the bulk of the instructions in most programming. Branch target instructions also occur in most programming due, for example, to the presence of loops within programs.

Known techniques for implementing instruction caches have normally used two separate and distinct caches, one instruction cache storing the

sequential instructions, and another instruction cache storing the branch target instructions. This conventional approach, however, is expensive to implement because additional die area as well as buses between the two caches are required. Further, this approach may also introduce extra delays. Another known approach places both sequential and branch target instructions into the same instruction cache. This conventional approach has a serious pollution in that one kind of instruction can displace the other type, thereby potentially inducing significant delays.

The present invention seeks to provide an improved cache.

According to an aspect of the present invention there is provided a cache memory comprising:

a single array of memory cells for storing data, said array including at least write word lines, write bit lines and read bit lines; and

partition circuitry disposed to partition said array into a first storage region and a second storage region,

wherein during a write operation to said array the first and second storage regions operate as separate cache memories, yet during a read operation said array operates as a single cache memory without regard to the first and second storage regions.

According to another aspect of the present invention there is provided a cache memory device operative to receive an address and to output data from a memory storage area, said memory storage area being partitioned during write operations into a plurality of regions in a programmable or adaptive manner.

According to another aspect of the present invention there is provided a method of operating a cache memory comprising the steps of:

(a) partitioning the cache memory into at least first and second storage regions;

(b) receiving data to be written to the cache memory;

(c) if the data received is of the first type of data, storing the data of

the first storage region; and

(d) if the data received is of the second type of data, storing the data to the second storage region.

The invention can provide an instruction cache which is able to store both types of instructions in a flexible and efficient manner.

Broadly speaking, the preferred embodiment provides a cache memory which is partitioned during write operations into several regions in a programmable or adaptive manner. The partitioning of the cache memory has no effect during read operations. Each region in the cache memory stores a different type of data. As an instruction cache, one of the regions would store sequential instructions and another would store branch target instructions. As a data cache, one of the regions would function as a data cache memory and another region would function as a victim cache and/or a prefetch buffer.

Preferably the cache memory includes a single array of memory cells for storing data and partition circuitry. The array of memory cells includes at least write word lines, write bit lines and read bit lines. The partition circuitry operates to partition the array into a first storage region and a second region. During a write operation to the array, the first and second storage regions operate as separate cache memories, yet during a read operation, they operate as a single cache memory without

regard to the first and second storage regions. It is also possible to dynamically partition the array.

A preferred method operates a cache memory to store at least a first type of data and a second type of data. The cache memory is partitioned into at least first and second storage regions. Thereafter, when data to be written to the cache memory is received, the method determines which of the storage regions of the cache memory to store the data in. If the data received is of the first type of data, the data is stored to the first storage region. On the other hand, if the data received is of the second type of data, the data is stored to the second storage region. The partitioning, for example, can be performed statically at power-on or dynamically by program instructions contained in a computer program being executed by a processing system including the cache memory.

An embodiment of the present invention is described below, by way of example only, with reference to the accompanying drawings, in which:

FIG.1 is a block diagram of an instruction unit in accordance with a first embodiment of the invention;

FIG.2 is a block diagram illustrating the first embodiment during a write operation;

FIG.3 is a block diagram of an embodiment of an address circuit; and

FIG.4 is a block diagram illustrating the first embodiment during a read operation.

Fig.1 is a block diagram of an instruction unit 2 the overall object of which (during a write operation) is to receive an instruction together with its associated tag (address tag) and to store the instruction and tag in an instruction cache. Typically, with high performance computers, the instruction and tag would be obtained from main memory or a level 2 cache memory in a prefetched manner. The instruction unit 2 thereafter (during a read operation) supplies the appropriate instructions to one or more functional units of a processing system. The functional units then carry-out the instructions by performing various operations.

The instruction unit 2 includes a tag buffer 4 and a prefetch buffer 6. The prefetch buffer 6 receives instructions from an instruction stream within a program being (or about to be) executed and stores the instructions in the prefetch buffer 6. The instructions may or may not be decoded depending on the design of the processing system. The tag buffer 4 stores tags corresponding to the instructions stored in the prefetch buffer 6. As noted above, the instruction and tag received by the instruction unit 2 may come from a lower level (e.g., level 2) cache memory or main memory.

The instruction unit 2 also includes a cache memory 10. The cache memory 10 is a single cache memory, that is, contains a single memory storage array. The cache memory 10 is partitioned horizontally into a first storage region 12 and a second storage region 14. During a write operation to the cache memory 10, a sequential instruction and its associated tag would be stored to the first storage region 12 via the prefetch buffer 6 and the tag buffer 4, respectively. Alternatively, if the instruction to be written to the cache memory 10 is a branch target instruction, then the instruction and its associated tag from the prefetch buffer 6 and the tag buffer 4, respectfully, would be stored to the second storage region 14 of the cache memory 10.

The partitioning of the cache memory 10 is achieved by effectively placing a divider within the instruction cache 10 to denote a division line 16 of the partitioning. However, as will be explained further below, the division line 16 is not physically present but achieved through addressing schemes.

During a read operation, an instruction is read from either the first storage region 12 or the second storage region 14 without regard to the partitioning at the division line 16. That is, whichever instruction within the cache memory 10 has a tag which matches the tag associated with the instruction to be read is supplied to an instruction buffer 18. Thereafter, the instruction buffer 18 supplies the instruction to the appropriate functional unit of the processing system.

FIG.2 is a block diagram illustrating the first embodiment during a write operation. In general, during a write operation, tag data 20 and instruction data 22 (provided via buffers 4 and 6) are placed on write bit lines (WBL) 24. The cache memory 10 also utilizes write word lines (WWL) 26 during the write operation. A memory cell is present at each intersection of a WBL and a WWL within the cache memory 10. A

decoder 28 receives an address 30 which selects one of several write word lines (WWL) 26. The selected WWL is then activated so as to store the tag data 20 and the instruction data 22 into the cache memory 10 at the memory cells associated with the selected WWL.

To implement the units, the division line 16 operates to partition the cache memory 10 into the first storage region 12 and the second storage region 14. Accordingly, the decoder 28 operates such that it will select one of write word lines WWL1-WWLn as the selected WWL when the instruction to be written to the cache memory 10 is a sequential instruction. This causes the sequential instruction to be stored in the first storage region 12 of the cache memory 10. On the other hand, the decoder 28 will operate to select one of write word lines WWLn+1 to WWLn+m as the selected WWL when the instruction to be written to the cache memory 10 is a branch target instruction.

In this embodiment, the address 30 supplied to the decoder 28 is controlled so that the decoder 28 selects the appropriate write word line (WWL) 26 based on the type of instruction to be stored to the cache memory 10.

FIG. 3 is a block diagram of an embodiment of an address circuit 32 The address circuit 32 includes a sequential address generation circuit 34 which produces a sequential address, and a target address generation circuit 36 which produces a target address. Both the address generation circuits 34, 36 receive a base number from a programmable base register 38, a cache size from a cache size register 40, and a reset signal 42. The base value stored in the programmable base register 38 is a value indicating where the cache memory 10 is to be partitioned. If the cache memory 10 is to be partitioned into more than two regions, then the programmable base register will contain several values

which indicate where partitions are to occur. The base value(s) may be stored into the programmable base register 38 at power-on, or alternatively by software techniques. The advantage of the software techniques is that the programmable base register 38 may be changed dynamically. As an example, the programmable base register 38 could be addressable and a special load instruction could be executed to write a base value into the programmable base register 38.

The address circuit 32 also includes a selector 44 which selects one of the sequential address and the target address based on a select signal 46. The select signal 46 can be provided by hardware (not shown) in the instruction unit 2 which can easily determined whether the prefetched instruction is, for example, a sequential instruction or a branch target instruction. For example, upon prefetching an instruction, the instruction unit 2 knows whether it is fetching a sequential instruction or a branch target instruction because some sort of prediction scheme is typically used to decide whether a branch will likely be taken or not. Hence, while prefetching an instruction, a target/sequential bit can be set and thereafter used to produce the select signal 46.

In effect, the base value stored to the programmable base register 38 corresponds to the value n shown in FIG. 2 and the cache size held in the cache size register 40 corresponds to the value n+m shown in FIG. 2. In the embodiment discussed above, where the cache memory 10 is partitioned into only two regions, the write word line WWLn+m would indicate the last word line for the second stored region 14. The value stored in the programmable base register 38 would indicate the value n which is the last write word line (WWLn) in the first storage region 12.

The address generation circuits 34 and 36 can be implemented with digital counters. Further, the techniques utilized to determine the particular

write word line to be activated within the region that is storing the data to be written can be determined by numerous known algorithms. One know algorithm is a random replacement algorithm. Using random replacement, the address generation circuits 34 and 36 would simply randomly generate an address within the region in question. Another known algorithm would be to search the region within the cache memory until an available write word line is found. The first write word line determined to be available would then used to store the data to be written. With this approach, the availability of a particular write word line is typically indicated by a extra bit associated with each word stored in the cache memory. This extra bit is preferably set or reset in a background mode of the processing system. In this case, the processing system could search each region within the cache memory until it finds a write word line which is available and then stores the data to be stored next at this location. If no available write word line is found within a predetermined period of time, then a write word line could be arbitrarily selected.

FIG. 4 is a block diagram of a cache memory during a read operation. The cache memory 10 includes read bit lines (RBL) 48 and 49, match lines 50, and write lines 52. During a read operation, tag data 54 is placed on certain of the read bit lines (RBL) 48. The tag data 54 is then compared with the tag stored at the memory cells associated with each of the match lines (ML) 50. At best, one of the match lines (ML) 50 would be activated upon finding a match between the stored tag and the supplied tag. The activated match line is driven high and latched into latch 56. The latch 56 then in turn separately drives its associated word line (WL) 52 via buffer 58. Although only one latch 56 and buffer 58 are shown in FIG. 4, it should be understood that each match line has a latch 56 and a buffer 58 associated therewith. The particular word line (WL) 52 which is driven

causes the data associated with the instruction stored to the memory cells associated with that word line to be placed on the read bit lines 49 which are associated with the stored instructions. The stored instruction is thereby supplied to the instruction buffer 18 over the read bit lines 49.

The processing system executes operations or instructions in accordance with a timing clock (CLK) having a period P. As an example, the period P is currently about four nanoseconds in high performance computers such as workstations. To accommodate such speeds, the comparing of the data tag 54 with the tag stored at the memory cells is preferably achieved by using content-addressable memory (CAM) for the memory cells containing the tag. The memory cells containing the instruction are preferably implemented by a random access memory (RAM). During a first half of the timing clock's cycle, the address evaluation occurs in the CAM, and the RAM storage is precharged. During the second half of the timing clock's cycle, the RAM is read and the CAM is precharged.

Although the embodiments discussed above refer to a cache memory partitioned into two regions, these can also be used to partition the cache memory into more than two partitions. For example, a cache memory can be partitioned into three regions, including a sequential instruction region, a branch target instruction region and a victim cache region. In this case, if a level 2 cache miss occurs, then line in the level 2 cache being replaced can be stored in the victim cache. Here, the write bit lines could be used to write back the cache line being replaced to the victim cache. If the sequential instruction region, the branch target instruction region and a victim cache region all use the same tags (e.g., all physical tags or all virtual tags), then all three regions can be referenced simultaneously during a one-cycle read operation. If all the regions do not use the same tags, then two cycles are required. In the first cycle the virtual tag can be compared

and address translation performed to obtain a physical tag. If a miss occurs in the first cycle, then in the second cycle, the region(s) using a physical tag are checked for a hit. If a hit occurs in the second cycle, then the cache line is read and possibly moved into another region of the cache to avoid the one-cycle penalty with future accesses to the same cache line.

Dynamic or adaptive application of the memory while executing a software program is advantageous if the software contains loops within the program code. For example, a compiler would identify sections of the program code in which a loop is being carried out, and then indicate in the code an instruction which (when executed) would increase the size of the region in the cache memory allocated to the branch target instructions prior to such loop sections. Then, when the program code leaves the loop sections, the compiler would insert another instruction causing the size of the branch target instruction region within the cache memory to decrease. Such techniques would optimize utilization of the cache memory in accordance with the nature of the program being executed. The programmable base register 38 could be used to dynamically store the partition information.

The memory has been found to work well for both direct-mapped and fully-associative caches. Moreover, although it was primarily discussed above with reference to instruction cache embodiments, the memory can also be embodied in a data caches where it would be useful to partition different pieces of data for performance reasons. The method described can also be used to partition a cache memory into an instruction cache region and a data cache region.

It may be preferable to also vertically partition the instruction cache as described in co-pending British patent application No. (RJ/N3629), entitled "Instruction Cache", and filed the same day as this application. For example, the instruction cache 10

could be vertically partitioned to avoid the need for an additional stage in a pipeline to determine a next instruction address.

The disclosure in United States patent application no. 08/298,861, from which this application claims priority, and in the abstract accompanying this application are incorporated herein by reference.

CLAIMS

1. A cache memory comprising:

a single array of memory cells for storing data, said array including at least write word lines, write bit lines and read bit lines; and

partition circuitry disposed to partition said array into a first storage region and a second storage region,

wherein during a write operation to said array the first and second storage regions operate as separate cache memories, yet during a read operation said array operates as a single cache memory without regard to the first and second storage regions.

2. A cache memory as recited in claim 1, wherein said cache memory further comprises a base register for storing a base value, and

wherein said array is partitioned in accordance with the base value.

3. A cache memory as recited in claim 1 or 2, wherein said partition circuitry is operative to partition dynamically said array.

4. A cache memory as recited in claim 1,2 or 3, wherein said partition circuitry comprises:

a programmable base register for storing a base value;

a sequential address generation circuit, operatively connected to said base register, for producing a sequential address based in part on the base value;

a target address generation circuit, operatively connected to said base register, for producing a branch target address based in part on the base value;

a selector, operatively connected to said sequential address generation circuit and said target address generation circuit, for selecting one of the sequential address and the branch target address to be a selected address based

on a select signal; and

a decoder, operatively connected to said selector, for receiving the selected address and activating one of the write word lines associated therewith.

5. A cache memory device operative to receive an address and to output data from a memory storage area, said memory storage area being partitioned during write operations into a plurality of regions in a programmable or adaptive manner.

6. A cache memory device as recited in claim 5, wherein during read operations the partitioning of said memory storage area has no effect.

7. A cache memory device as recited in claim 5 or 6, wherein each of said regions stores a different type of data.

8. A cache memory device as recited in claim 5,6 or 7, wherein the plurality of regions includes at least a first storage region for storing sequential instructions and a second storage region for storing branch target instructions.

9. A cache memory device as recited in any one of claims 5 to 8, wherein the address circuit operates to partition said memory storage area in accordance with a base value.

10. A cache memory device as recited in claim 9, wherein said base value is stored in a programmable base register.

11. A cache memory device as recited in claim 9 or 10, wherein when said cache memory device is used in a computer system which executes a computer

program,

said base value is set by an instruction contained in the computer program being executed.

12. A method of operating a cache memory comprising the steps of:

(a) partitioning the cache memory into at least first and second storage regions;

(b) receiving data to be written to the cache memory;

(c) if the data received is of the first type of data, storing the data of the first storage region; and

(d) if the data received is of the second type of data, storing the data to the second storage region.

13. A method as recited in claim 12, wherein said partitioning dynamically alters the size of the partitions within the cache memory.

14. A method as recited in claim 13, wherein said partitioning is dynamically altered by a program instruction contained in a computer program being executed by a processing system including the cache memory.

15. A method as recited in claim 12,13 or 14, wherein said method comprises the step of:

(e) reading data from the cache memory without regard to said partitioning.

16. A cache memory substantially as hereinbefore described with reference to and as illustrated in the accompanying drawings.

17. A method of operating a cache memory substantially as hereinbefore described with reference to and illustrated in the ccompanying drawings.

The Patent Office

—

Application No: GB 9513272.6 **Examiner:** Mr S J Probert
Claims searched: 1-4 **Date of search:** 8 September 1995

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.N): G4A AMC

Int Cl (Ed.6): G06F 12/08

Other: Online : WPI, INSPEC

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
Y	EP 0466265 A1	(Philips) - See abstract	1-3
XY	EP 0284751 A2	(IBM) - See whole document	1-3
X	WO 93/09497 A2	(Unisys Corp) - See page 2 lines 14 to 28.	1
X	US 4905141	(Brenza) - See whole document	1

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.