PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
G06F 9/38

A1

(11) International Publication Number: **WO 95/08801**

(43) International Publication Date: 30 March 1995 (30.03.95)

(21) International Application Number: PCT/GB94/01793

(22) International Filing Date: 16 August 1994 (16.08.94)

(30) Priority Data:
9319662.4 23 September 1993 (23.09.93) GB

(71) Applicant *(for all designated States except US)*: ADVANCED RISC MACHINES LIMITED [GB/GB]; Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN (GB).

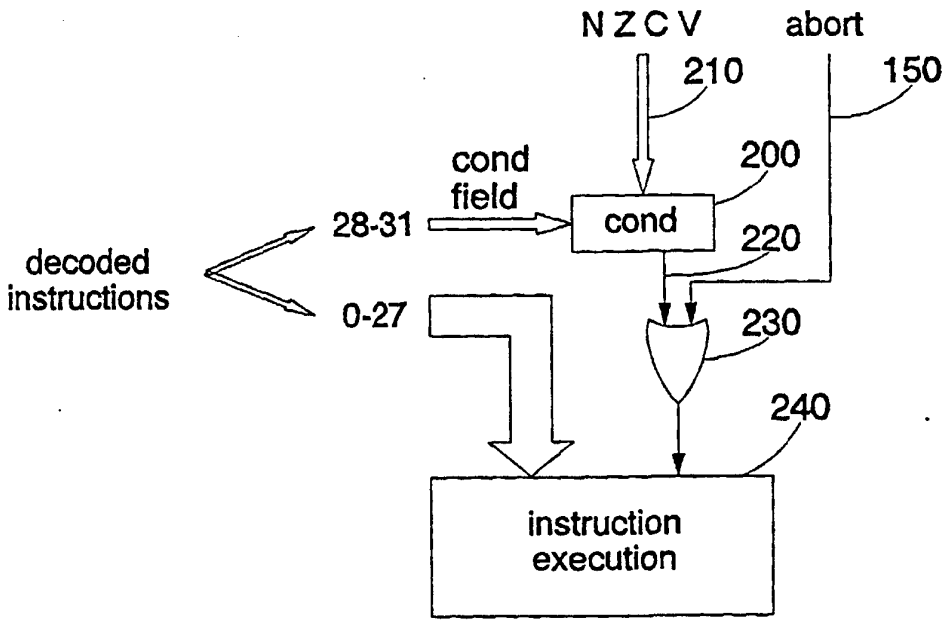
(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: JAGGAR, David, Vivian [NZ/GB]; 48 Mandrill Close, Cherry Hinton, Cambridge CB1 4TN (GB).

(74) Agent: TURNER, James, Arthur; D. Young & Co., 21 New Fetter Lane, London EC4A 1DA (GB).

(81) Designated States: CN, JP, KR, RU, US, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: EXECUTION OF DATA PROCESSING INSTRUCTIONS



(57) Abstract

Data processing apparatus in which successive data processing instructions are executed comprises: memory accessing means for accessing a data memory in response to one or more of the instructions, the memory accessing means comprising means for detecting whether each memory access is invalid; condition test means, responsive to a processing state of the apparatus generated by previously executed instructions and operable during execution of each instruction, for detecting whether that instruction should be executed; and conditional control means, responsive to the memory accessing means and to the condition test means, for preventing complete execution of a current instruction if either the memory accessing means detects that a memory access initiated by the preceding instruction is invalid or the condition test means detects that the current instruction should not be executed.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**GB**	United Kingdom	**MR**	Mauritania
AU	Australia	**GE**	Georgia	**MW**	Malawi
BB	Barbados	**GN**	Guinea	**NE**	Niger
BE	Belgium	**GR**	Greece	**NL**	Netherlands
BF	Burkina Faso	**HU**	Hungary	**NO**	Norway
BG	Bulgaria	**IE**	Ireland	**NZ**	New Zealand
BJ	Benin	**IT**	Italy	**PL**	Poland
BR	Brazil	**JP**	Japan	**PT**	Portugal
BY	Belarus	**KE**	Kenya	**RO**	Romania
CA	Canada	**KG**	Kyrgystan	**RU**	Russian Federation
CF	Central African Republic	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CG	Congo			**SE**	Sweden
CH	Switzerland	**KR**	Republic of Korea	**SI**	Slovenia
CI	Côte d'Ivoire	**KZ**	Kazakhstan	**SK**	Slovakia
CM	Cameroon	**LI**	Liechtenstein	**SN**	Senegal
CN	China	**LK**	Sri Lanka	**TD**	Chad
CS	Czechoslovakia	**LU**	Luxembourg	**TG**	Togo
CZ	Czech Republic	**LV**	Latvia	**TJ**	Tajikistan
DE	Germany	**MC**	Monaco	**TT**	Trinidad and Tobago
DK	Denmark	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**US**	United States of America
FI	Finland	**ML**	Mali	**UZ**	Uzbekistan
FR	France	**MN**	Mongolia	**VN**	Viet Nam
GA	Gabon				

EXECUTION OF DATA PROCESSING INSTRUCTIONS

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to the execution of data processing instructions.

Description of the Prior Art

Some data processors comprise a central processor unit (CPU) which is able, under the control of a currently executed data processing instruction, to access data stored in a random access memory (RAM) via an intermediate memory management unit. A previously proposed example of such a data processor is the ARM6 processor, described in the "ARM6 Data Sheet" published by Advanced Risc Machines Limited, 1993.

During a memory access, the memory management unit may generate an abort signal indicating that the current memory access cannot be completed. Abort signals may be generated for a number of reasons. In one example, an aborted memory access can occur in a data processing system employing virtual memory in which data are swapped between a RAM and slower disk storage to give the illusion that the addressable memory space is greater than the amount of RAM provided. In such a system, if data corresponding to a required virtual address are currently held in the disk storage rather than the RAM, there will be a delay before those data are accessible, during which delay the data have to be transferred from the disk storage into the RAM. In this case, the current memory access is aborted, and an attempt is made later to access those data.

The abort signal supplied from the memory management unit is generated too late to stop execution of the instruction which initiated the failed memory access, but can instead be used to cancel execution of the following data processing instruction, i.e. the data processing instruction after the one which initiated the failed memory access. This is useful because subsequent instructions may rely on the memory access having been successful.

The use of the abort signal to cancel execution of the instruction immediately following the instruction which initiated the failed memory access places stringent requirements on the timing of the

abort signal. Alternatively, a complex mechanism must be provided to 'undo' the results of the execution of the immediately following instruction, after execution of that instruction has been completed.

Figure 1 of the accompanying drawings is a schematic timing diagram illustrating the timing requirements of the abort signal during a data write operation (in which data are written to RAM) by the previously proposed data processor referred to above.

Referring to Figure 1, a clock signal 10 controls the execution of data processing instructions by the data processor. When a data write operation is initiated, a memory address 20 is supplied by the data processor to a memory management unit, and one half-cycle of the clock signal later, the data 30 to be written to that address are output by the data processor.

If the memory management unit detects that the memory address 20 is invalid (for example, because data corresponding to that address are currently held in disk storage in a virtual memory system), an abort signal 40 is generated by the memory management unit and supplied to the data processor.

The next instruction after a data write instruction is executed straight away, since there is no need (under normal circumstances) to await a response from the memory management unit after the data to be written have been placed on the data bus. Accordingly, in order for the abort signal to arrive in time to cancel execution of the immediately following instruction, the previously proposed data processor referred to above requires the abort signal to be valid one half-cycle of the clock signal before the data to be written are output by the data processor.

In practice, this timing constraint is difficult to achieve, and requires particularly fast operation of the memory management unit (with a correspondingly high power consumption by that unit).

The previously proposed data processor referred to above also provides conditional execution of its entire instruction set. This is achieved by comparing the current state of up to four processing flags with respective states defined by a condition code included in each instruction. This comparison takes place concurrently with execution of the instruction. Complete execution of an instruction is then prevented if the state of the processing flags does not match that

specified by the condition code.

SUMMARY OF THE INVENTION

This invention provides a data processing apparatus in which successive data processing instructions are executed, the apparatus comprising: memory accessing means for accessing a data memory in response to one or more of the instructions, the memory accessing means comprising means for detecting whether each memory access is invalid; condition test means, responsive to a processing state of the apparatus generated by previously executed instructions and operable during execution of each instruction, for detecting whether that instruction should be executed; and conditional control means, responsive to the memory accessing means and to the condition test means, for preventing complete execution of a current instruction if either the memory accessing means detects that a memory access initiated by the preceding instruction is invalid or the condition test means detects that the current instruction should not be executed.

In a data processing apparatus according to the invention, a fully conditional instruction set is employed, and the mechanism for conditionally preventing complete execution of each instruction is also used to handle memory aborts. Thus, memory abort signals relating to a memory access initiated by the preceding instruction can be received at the same time, during execution of each instruction, as the detection by the condition test means of whether that instruction should be executed. This can allow memory aborts to be processed at a later time for each instruction than that allowed for the previously proposed data processor described above.

In a preferred embodiment, the apparatus comprises one or more processing flags for storing data indicative of a current processing state of the data processing apparatus; each instruction includes a condition code defining a state of the processing flags required for that instruction to be executed; and the condition test means is operable to compare the required state of the processing flags defined by the condition code in each instruction with the actual state of the processing flags. The condition codes may specify that a particular processing flags should be set to a particular logical state, or that the state of that processing flag has no influence on whether the current instruction should be executed. In an extreme case, a possible

condition code may specify that a particular instruction should be executed regardless of the state of any of the processing flags.

The processing flags could specify various features of the processing state of the apparatus. In a preferred embodiment the apparatus comprises four processing flags respectively denoting:

(i) whether a previous data processing operation of the apparatus generated a negative result;

(ii) whether a previous data processing operation of the apparatus generated a zero result;

(iii) whether a carry bit was set by a previous data processing operation of the apparatus; and

(iv) whether an arithmetic overflow occurred during a previous data processing operation of the apparatus.

Preferably the memory accessing means comprises: means for transmitting a memory address to the data memory; and means for subsequently transmitting data to the data memory or receiving data from the data memory.

In order that the conditional control means can conveniently be made responsive to both the memory accessing means and to the condition test means, it is preferred that: the memory accessing means is operable to generate an abort control signal to indicate that a memory access is invalid; the condition test means is operable to generate a condition failure control signal to indicate that the current instruction should not be executed; and the apparatus comprises means for combining the abort control signal and the condition failure control signal to generate a combined control signal for supply to the condition control means.

In an advantageously simple embodiment, the means for combining comprises a logical OR gate.

In a convenient embodiment, data processing operations of the apparatus are controlled by a clock signal.

Viewed from a second aspect this invention provides an integrated circuit comprising apparatus as defined above.

Viewed from a third aspect this invention provides a method of data processing in which successive data processing instructions are conditionally executed, the method comprising the steps of: accessing a data memory in response to one or more of the instructions; detecting

whether each memory access is invalid; detecting, during execution of each instruction, whether that instruction should be executed, in dependence on a processing state of the apparatus generated by previously executed instructions; and preventing complete execution of a current instruction if it is detected either that a memory access initiated by the preceding instruction is invalid or that the current instruction should not be executed.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will now be described by way of example with reference to the accompanying drawings, throughout which like parts are referred to by like references, and in which:

Figure 1 is a schematic timing diagram illustrating the timing requirements of an abort signal during a data write operation by a previously proposed data processor;

Figure 2 is a schematic block diagram of a data processing apparatus according to an embodiment of the invention;

Figure 3 is a schematic block diagram of a part of a central processing unit; and

Figure 4 is a schematic timing diagram illustrating the timing requirements of an abort signal during a data write operation by the data processing apparatus of Figure 2.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring now to Figure 2, a schematic block diagram of a data processing apparatus according to an embodiment of the invention is illustrated. The apparatus comprises a central processing unit (CPU) 100, a memory management unit (MMU) 110 and a random access memory (RAM) 120. The CPU 100 and the MMU 110 are linked by an address bus 130 for the exchange of memory addresses and a data bus 140 for the exchange of data. An abort control line 150 is also provided from the MMU 110 to the CPU 100 to carry an abort signal indicative of a failed or invalid memory access.

A clock signal is supplied to the CPU 100 and to the MMU 110 to control the operations of both of these units. Each data processing instruction is executed by the CPU 100 in a particular number of cycles of the clock signal (depending on the nature of that instruction) and memory access by the MMU 110 is performed in synchronism with the clock signal.

The MMU 110 operates under the control of the CPU 100 and the clock signal to access data stored in the RAM 120. Accordingly, the MMU is linked to the RAM by a plurality of address and data lines 160.

Figure 3 is a schematic block diagram of a part of the central processing unit 100. The CPU 100 employs instruction pipelining, to allow the processing and memory operations to be performed substantially continuously. Typically, while one data processing instruction is being executed, its successor is being decoded and a third instruction is being fetched from memory. This arrangement is referred to as a three-stage execution pipeline.

In the part of the CPU 100 illustrated in Figure 3, data processing instructions which have been fetched from memory and then decoded are passed for execution. The data processing instructions are 32-bit data words, of which bits 28 to 31 form a 4-bit condition field. The remaining bits (bits 0 to 27) define the operation to be performed in response to that instruction and, in some cases an operand on which the operation is to be performed.

The condition field (bits 28 to 31) is passed to a condition tester 200 which compares the bits of the condition field with 16 pre-defined condition codes. The 16 condition codes define the state of one or more of four processor flags 210, referred to as the N, Z, C and V flags. These flags represent a processing state of the CPU 100 generated by previously executed instructions.

The N flag denotes a negative result from the previous arithmetic operation of the CPU 100; the Z flag denotes a zero (equal) result from the previous CPU operation; the C flag denotes whether a carry bit was set during the previous CPU operation; and the V flag indicates an arithmetic overflow occurring during the previous CPU operation.

The relation between the four bits of the condition field and the conditions applied to the N, Z, C and V flags are shown in the list below:

```
0000 =  EQ  -  Z set (equal)
0001 =  NE  -  Z clear (not equal)
0010 =  CS  -  C set (unsigned higher or same)
0011 =  CC  -  C clear (unsigned lower)
0100 =  MI  -  N set (negative)
```

0101	=	PL	-	N clear (positive or zero)
0110	=	VS	-	V set (overflow)
0111	=	VC	-	V clear (no overflow)
1000	=	HI	-	C set and Z clear (unsigned higher)
1001	=	LS	-	C clear or Z set (unsigned lower or same)
1010	=	GE	-	N set and V set, or N clear and V clear (greater or equal)
1011	=	LT	-	N set and V clear, or N clear and V set (less than)
1100	=	GT	-	Z clear, and either N set and V set, or N clear and V clear (greater than)
1101	=	LE	-	Z set, or N set and V clear, or N clear and V set (less than or equal)
1110	=	AL	-	always
1111	=	NV	-	never

The condition tester 200 tests the state of the flags listed above, in dependence on which bits are set in the condition field of the current instruction. This comparison takes place during execution of the current instruction. The current instruction is allowed to complete its execution only if the appropriate flags are set to the states specified by the condition field.

If the always (AL) condition is specified, the instruction will be executed irrespective of the flags. The never (NV) condition code prevents execution of the instruction irrespective of the state of the flags 210.

The condition tester 200 generates an output signal 220 indicating whether the current instruction should be completely executed. The output signal 220 is combined with the abort signal supplied on the abort control line 150 from the MMU 110 to the CPU 100, using an OR-gate 230. The output of the OR-gate 230 is therefore set if either the condition tester 200 indicates that the current instruction should not be completely executed, or the abort signal is asserted by the MMU 110.

An instruction execution unit 240 receives bits 0 to 27 of each instruction, defining the operation to be performed and, in some cases,

an operand on which the operation is to be performed. During execution of the current instruction (i.e. after the propagation and processing delays of the condition tester 200 and the OR-gate 230), the instruction execution unit 240 receives the output of the OR-gate 230 indicating whether the current instruction should be completely executed. If the output of the OR-gate 230 indicates that the current instruction should not be completely executed, the instruction is cancelled without changing the state of any registers or memory locations associated with the apparatus.

The condition tester 200 and the instruction execution unit 240 may be of the same form as the corresponding components of the previously proposed ARM 6 processor referred to above.

The result of using the apparatus of Figure 3 is that the stringent timing of the abort signal is greatly relaxed, so that the abort signal relating to a memory access initiated by the preceding instruction may be set at a late stage during execution of each instruction.

If an abort signal is received as a result of a failed instruction fetch operation, then that instruction is simply discarded at a later stage in the three-stage instruction pipeline referred to above.

Data read operations are followed by a non-memory-accessing (internal) processor cycle, to allow time for the data which has been read from memory to be loaded into the appropriate processor register. Accordingly, if an abort signal is received as a result of a failed data read operation, the internal cycle following the read operation allows time for any data supplied from memory by the MMU 110 (which data may well be erroneous) to be ignored and not stored in the intended location (e.g. a processor register).

The timing relationship of the instruction execution and the receipt of the abort signal for a data write operation is illustrated in Figure 4, in which a clock signal 300 which controls instruction execution by the CPU 100 is illustrated along with the states of the address bus 130 and the data bus 140.

In order to initiate the data write operation an address 310 is placed on the address bus by the CPU 100. One half-cycle of the clock signal 300 later, data 320 are placed on the data bus 140 by the CPU

100. Once this has been done, the CPU 100 is able to execute the next instruction during the following cycle of the clock signal 300.

At the same time as the data 320 are placed on the data bus, if an abort signal is received (indicated in Figure 4 as 330) or if the condition tester 200 determines that the next instruction should not be executed (which is determined by the condition tester 200 during the half-cycle after the data has been placed on the data bus and is indicated as 340), execution of the next instruction is abandoned.

Although illustrative embodiments of the invention have been described in detail herein with reference to the accompanying drawings, it is to be understood that the invention is not limited to those precise embodiments, and that various changes and modifications can be effected therein by one skilled in the art without departing from the scope of the invention as defined by the appended claims.

CLAIMS

1. Data processing apparatus in which successive data processing instructions are executed, said apparatus comprising:

memory accessing means for accessing a data memory in response to one or more of said instructions, said memory accessing means comprising means for detecting whether each memory access is invalid;

condition test means, responsive to a processing state of said apparatus generated by previously executed instructions and operable during execution of each instruction, for detecting whether that instruction should be executed; and

conditional control means, responsive to said memory accessing means and to said condition test means, for preventing complete execution of a current instruction if either said memory accessing means detects that a memory access initiated by a preceding instruction is invalid or said condition test means detects that said current instruction should not be executed.

2. Apparatus according to claim 1, in which:

said apparatus comprises one or more processing flags for storing data indicative of a current processing state of said data processing apparatus;

each instruction includes a condition code defining a state of said processing flags required for that instruction to be executed;

said condition test means is operable to compare a required state of the processing flags defined by said condition code in each instruction with an actual state of said processing flags.

3. Apparatus according to claim 2, comprising four processing flags respectively denoting:

(i) whether a previous data processing operation of said apparatus generated a negative result;

(ii) whether a previous data processing operation of said apparatus generated a zero result;

(iii) whether a carry bit was set by a previous data processing operation of said apparatus; and

(iv) whether an arithmetic overflow occurred during a previous

data processing operation of said apparatus.

4. Apparatus according to any one of claims 1 to 3, in which said memory accessing means comprises:

means for transmitting a memory address to said data memory; and

means for subsequently transmitting data to said data memory or receiving data from said data memory.

5. Apparatus according to any one of the preceding claims, in which:

said memory accessing means is operable to generate an abort control signal to indicate that a memory access is invalid;

said condition test means is operable to generate a condition failure control signal to indicate that said current instruction should not be executed; and

said apparatus comprises means for combining said abort control signal and said condition failure control signal to generate a combined control signal for supply to said condition control means.

6. Apparatus according to claim 5, in which said means for combining comprises a logical OR gate.

7. Apparatus according to any one of the preceding claims, in which data processing operations of the apparatus are controlled by a clock signal.

8. An integrated circuit comprising apparatus according to any one of the preceding claims.

9. A method of data processing in which successive data processing instructions are conditionally executed, said method comprising the steps of:

accessing a data memory in response to one or more of said instructions;

detecting whether each memory access is invalid;

detecting, during execution of each instruction, whether that instruction should be executed, in dependence on a processing state of said apparatus generated by previously executed instructions; and

preventing complete execution of a current instruction if it is detected either that a memory access initiated by a preceding instruction is invalid or that said current instruction should not be executed.

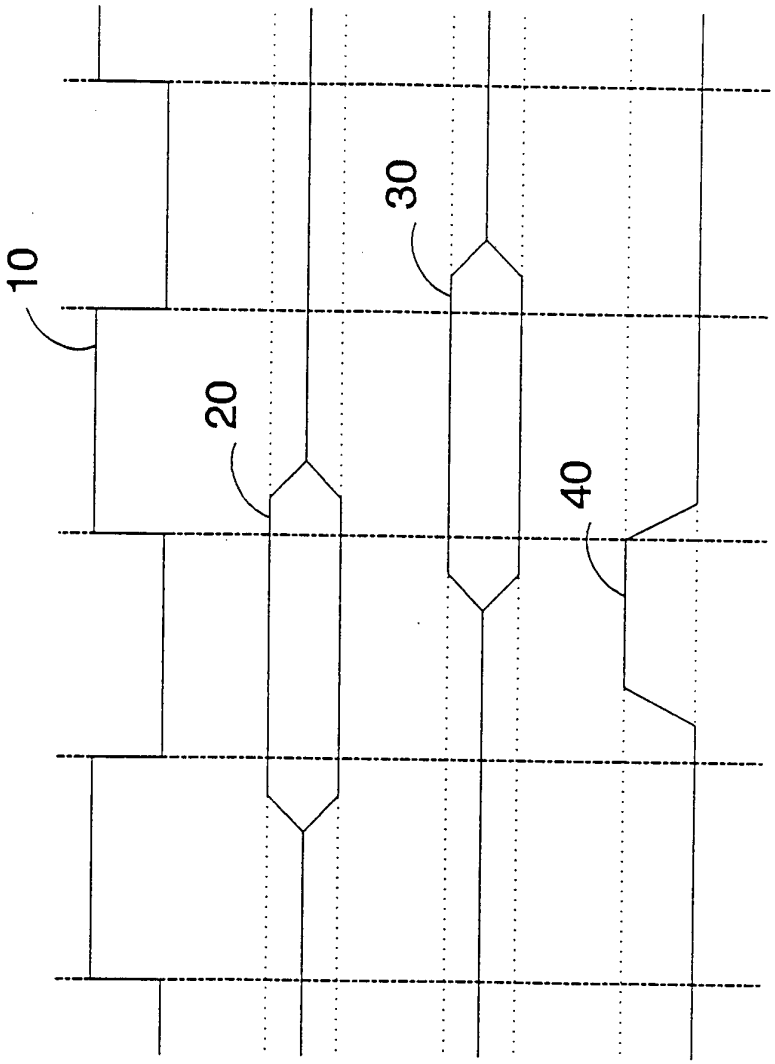


Fig. 1

1/4

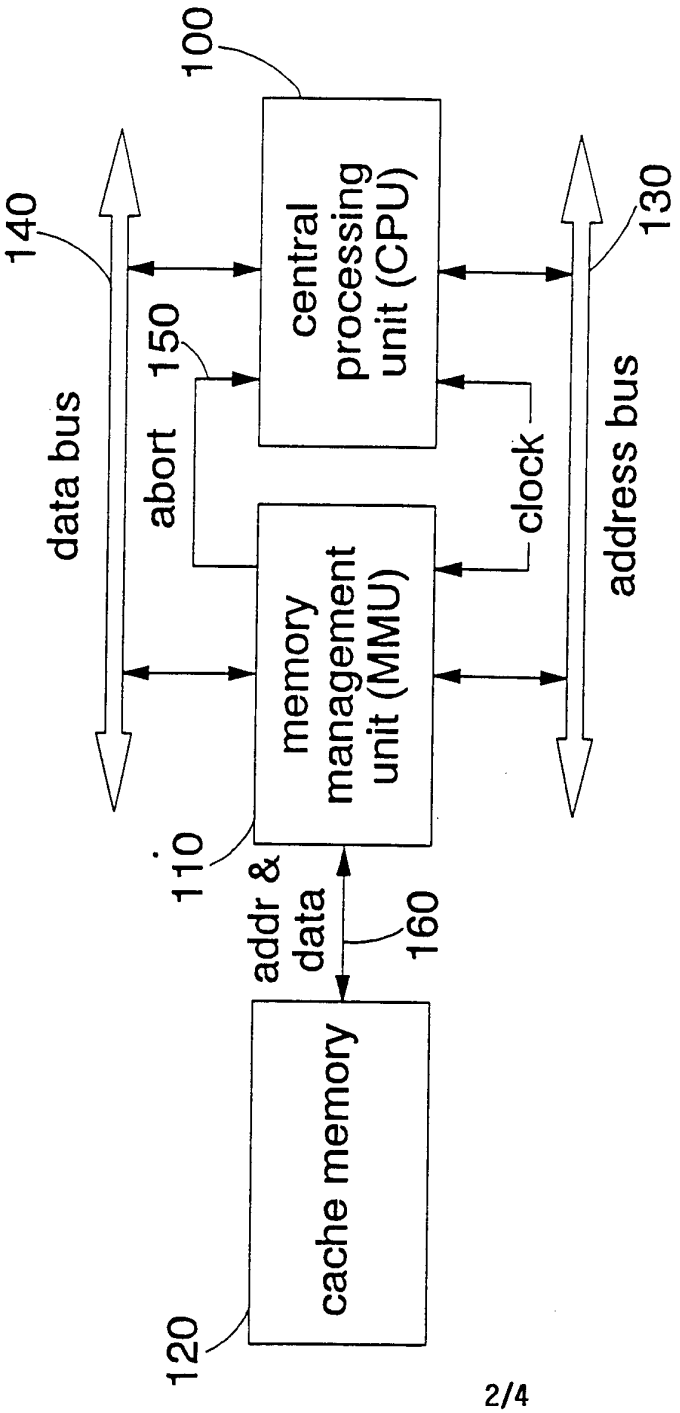

100
central processing unit (CPU)
140
data bus
150
abort
clock
130
address bus
memory management unit (MMU)
110
addr & data
160
cache memory
120

Fig. 2

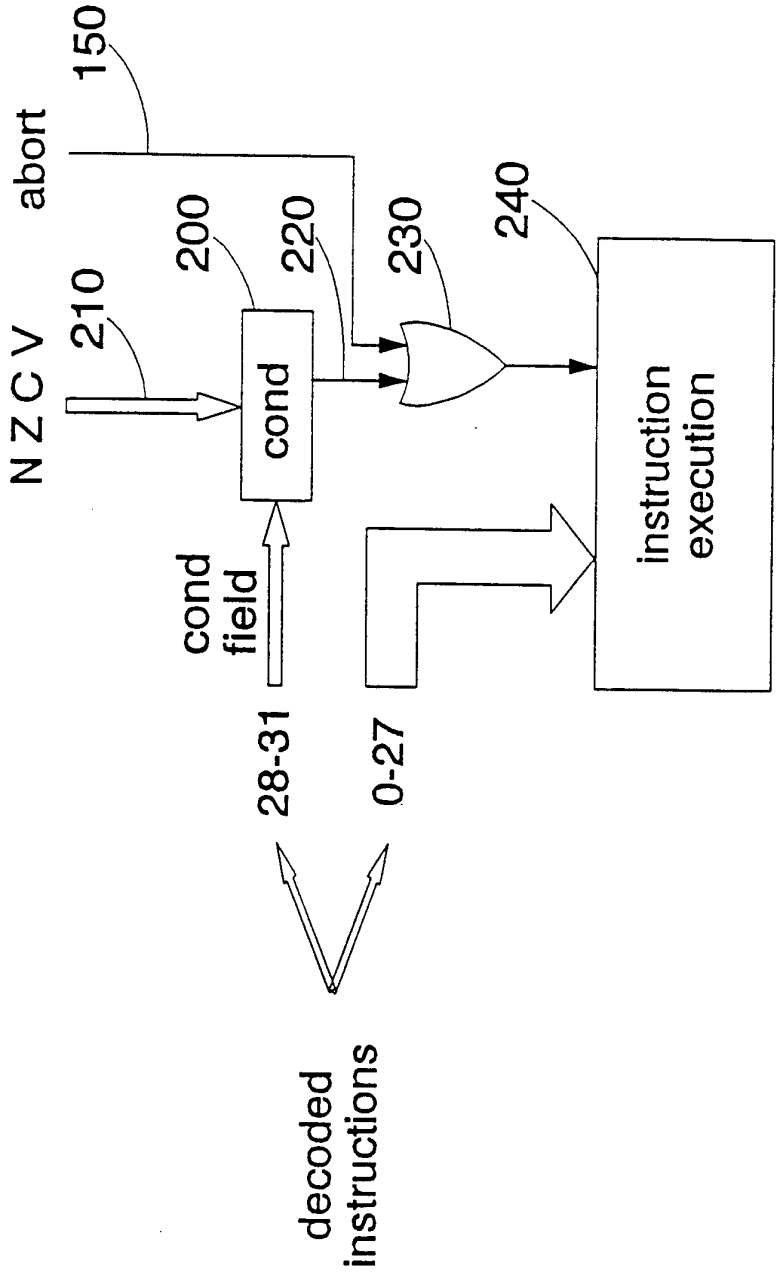

150
abort
200
220
230
240
210
N Z C V
cond
instruction execution
cond field
28-31
0-27
decoded instructions

Fig. 3

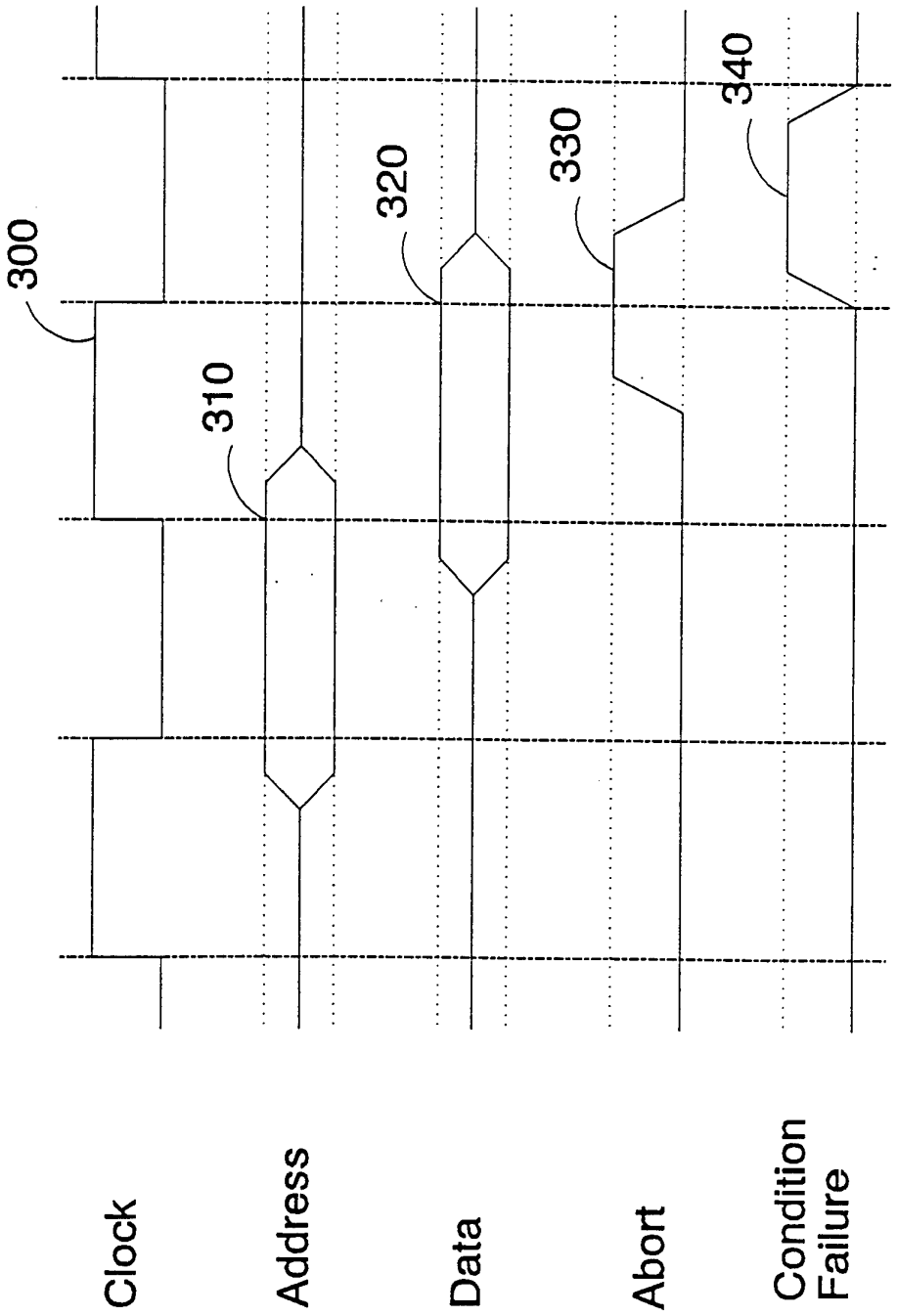

300
310
320
330
340
Clock
Address
Data
Abort
Condition Failure

Fig. 4

INTERNATIONAL SEARCH REPORT

Interr. ..al Application No
PCT/GB 94/01793

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/38

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	COMPCON SPRING '93 February 1993 , SAN FRANCISCO, CA, US pages 80 - 87 M. MULLER 'ARM6 a High Performance Low Power Consumption Macrocell' * page 84 section 3.3 * ---	1-4,7-9
Y	S. B. FURBER 'VLSI RISC Achitecture and Organization' 1989 , MARCEL DEKKER, INC. , NEW YORK, US. see page 229 - page 230 --- -/--	1-4,7-9

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 November 1994	17. 11. 94
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Authorized officer Daskalakis, T

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IBM TECHNICAL DISCLOSURE BULLETIN. vol. 25, no. 12 , May 1983 , NEW YORK US pages 6711 - 12 L. C. GARCIA ET AL. 'Storage Access-Exception Detection for Pipelined Execution Units' see the whole document	1,5,7,9
A	IBM TECHNICAL DISCLOSURE BULLETIN. vol. 27, no. 5 , October 1984 , NEW YORK US pages 2757 - 59 W. C. BRANTLEY ET AL. 'Exception Handling in a Highly Overlapped Machine' see page 2757, line 33 - line 36 see page 2758, line 9 - line 12	5,6
A	US,A,3 766 527 (BRILLEY) 16 October 1973	
A	US,A,4 385 365 (HASHIMOTO ET AL.) 24 May 1983	

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter ..al Application No
PCT/GB 94/01793

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
US-A-3766527	16-10-73	AU-B-	464291	21-08-75
		AU-A-	4699272	28-03-74
		BE-A-	789583	01-02-73
		CA-A-	954229	03-09-74
		CH-A-	560933	15-04-75
		DE-A-	2248296	05-04-73
		FR-A-	2158833	15-06-73
		GB-A-	1402585	13-08-75
		NL-A-	7213248	03-04-73
		SE-B-	393200	02-05-77
US-A-4385365	24-05-83	JP-C-	1094894	27-04-82
		JP-A-	54107645	23-08-79
		JP-B-	56040382	19-09-81