(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 October 2010 (07.10.2010)

PCT

(10) International Publication Number
WO 2010/112970 A1

(51) International Patent Classification:
G06F 9/06 (2006.01) *G06F 9/34* (2006.01)
G06F 9/30 (2006.01)

(21) International Application Number:
PCT/IB2009/051345

(22) International Filing Date:
31 March 2009 (31.03.2009)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **FREESCALE SEMICONDUCTOR, INC.** [US/US]; 6501 William Cannon Drive West, Austin, Texas 78735 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **KRUECKEN, Joachim** [DE/DE]; Felicitas-Füss-Straße 29, 81827 Munich (DE).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

[Continued on next page]

(54) Title: DATA PROCESSING WITH VARIABLE OPERAND SIZE

(57) Abstract: A method of processing data comprising performing a sequence of operation instructions with variable operand size, wherein respective size codes for different source and destination operands are obtained and registered 406 separately from performing 408 the sequence of operation instructions, and the sequence of operation instructions is performed 408 using operand sizes defined by the registered size codes, the operation instructions of the sequence not themselves containing size codes.

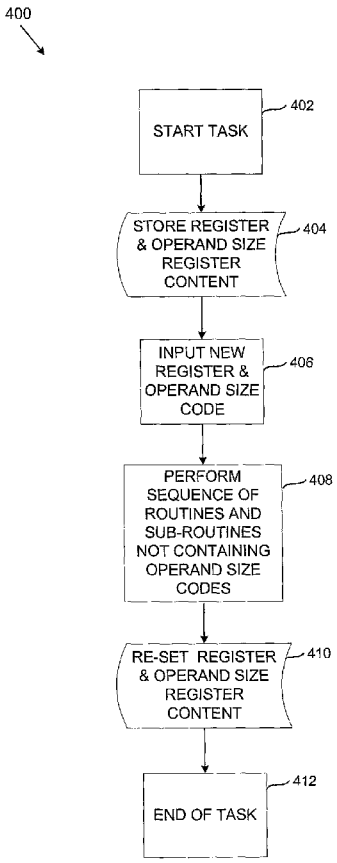

Fig. 4

WO 2010/112970 A1

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report (Art. 21(3))*

Title : DATA PROCESSING WITH VARIABLE OPERAND SIZE

Description

Field of the invention

This invention relates to a data processor and to a method of processing data with variable operand size.

Background of the invention

A data processor functions in response to machine code instructions. A machine code instruction set may have all instructions of the same length, or may have variable-length instructions. How the machine code instructions are organized depends largely on the specification of the machine code. Common to most organisations is the division of the instruction into one field, the opcode (operation code) which specifies the exact operation (for example "add") and other fields, which may give the type of the operands, their location, or their value directly (operands contained in an instruction are called immediate). The specification and format of the instructions are laid out in the instruction set architecture of the processor in question (which may be a general central processor unit ('CPU') or a more specialized processing unit). The location of an operand may be a register or a memory. The types of operations may include arithmetics, logical operations and data instructions including display or print characters for example, as well as special instructions (such as CPUID and others) and program control.

US patent specification 4,649,477 describes certain issues arising with instruction execution in a typical data processor, such as a microcontroller or a microprocessor. The instructions are partitioned into functional routines. These routines perform either a portion of the instruction or the entire instruction, depending on the instruction's complexity. An example is the fetch effective address routine which calculates an effective address and then fetches the sized operand from that address. To perform this operation, it is necessary to provide size information during the address calculation operation. The size is also needed by the bus controller to determine the size of the operand to be fetched from memory. These routines require that the instruction be decoded to reflect the size of the operand to be fetched. Similarly, the instruction operation routine which performs the operation on the operands for the instruction may require their size.

In a machine which performs operations on various sized operands that is controlled by a control structure, it is possible to provide different control sequences for each sized operation. This technique is inefficient, however, for while it is necessary to provide separate control for those operations larger than the size of the available resources, it would be preferable that all sizes less than or equal to those resources should share control sequences and allow secondary decoding of the instruction and provide the determination as to which size should occur.

If there is sharing of code sequences for multiple operations as well as multiple operand sizes, the constraints become more stringent. As an example, there are functional routines provided to evaluate an effective address and to fetch the sized operand. US patent specification

4,649,477 refers to data processors having a 16 bit data bus in which there are two different routines, one to fetch a long word operand and the other to fetch a byte or a word operand depending upon the size of the operation. These routines can be called in the execution of any instruction.

US patent specification 4,649,477 refers to data processors which perform operations on operands which have their size or length specified as part of the instruction, the operand and register size being controlled implicitly. The size of the operation normally reflects the size of the operand to be operated on for an instruction. There are three operand sizes handled in the examples of processors referred to, namely 8-, 16- and 32-bit operands to perform byte, word and long word operations, respectively. The control used to perform sized operations is derived from the instruction and affects the execution unit, condition codes in the condition code register, the data bus and the bus controller. Other operand sizes are also possible, notably 64-bit, for example.

US patent specification 4,649,477 proposes a data processor which has an instruction register for storing an instruction which selects the operation to be performed on the operand and the size of the operand, and a size controller for selectively enabling either the instruction register or another functional block, or a size selector to select the size of the operand so as to allow both implicit residual size control, where the size is calculated by one of several control units, and explicit size control, where the size is selected or forced by the size selector.

US patent specification 4,649,477 aims in this way to achieve a simplification in the internal state machine based on selecting the operand sizes by an internal size bus where different sized operands are needed. However, when employing explicit control it applies the same operand size to each register and does not enable dynamic switching of the register sizes, that is to say making the register size part of a task state.

Summary of the invention

The present invention provides a data processor and a method of processing data as described in the accompanying claims.

These and other aspects of the invention will be apparent from and elucidated with reference to the embodiments described hereinafter.

Brief description of the drawings

Further details, aspects and embodiments of the invention will be described, by way of example only, with reference to the drawings. Elements in the figures are illustrated for simplicity and clarity and have not necessarily been drawn to scale.

Figure 1 is a block schematic diagram of a data processor in accordance with one embodiment of the invention, given by way of example,

Figure 2 is a schematic diagram of registers in the data processor of Figure 1 in one configuration of usage,

Figure 3 is a schematic diagram of registers in the data processor of Figure 1 in another configuration of usage, and

Figure 4 is a flow chart of a method of processing data in accordance with one embodiment of the invention, given by way of example.

Detailed description of the preferred embodiments

Figure 1 shows schematically the general architecture of a typical data processor 100 to which one embodiment of the invention is applied.

In a data processor, such as a microprocessor for example, the effective size of general purpose registers such as a register bank 118 is often set as a static attribute for each register, for example 8-bit, 16-bit or 32-bit. Effective size means that only the least significant bits ('LSBs') selected by the size attributes are used by the computation and load/store instructions and some most significant bits ('MSBs') not covered by a register's attribute may be ignored. In prior data processors, in order to avoid having to provide different control sequences for each sized operation, operand size is coded as part of the instruction.

The instruction coding in a microcontroller typically requires one or more bits to code the operand size. In the above example, of 8-, 16-, 32-bit operand sizes, 3 codes out of 4 possible codes for 2 bit positions are required and consume effectively 1.5 bits of the operating instructions. In many 16-bit controllers, such as the S12-16 microprocessor of Freescale Semiconductor, Inc. for example, the majority of op-codes are only 8-Bit wide, so these 1.5 bits consume a major portion of the available code space. Also, for a 16-Bit op-code like on the ColdfireV1 or PowerPC microprocessors of Freescale Semiconductor, Inc., and even for 32-bit controllers such as ARM, it is a significant portion of the available code space, resulting in reduction of code density for the operational instructions.

Another effect is that parts of a typical application program in a multitasking system have different properties with respect to the operand sizes needed. For example, an interrupt handler mainly uses 8 or 16 Bit wide operands, while other tasks heavily using mathematical operations prefer 32-Bit operations. Many controllers today have a fixed set of fixed registers

This embodiment of the present invention provides a method of processing data with comprising performing a sequence of operation instructions with variable operand size. Respective size codes for different source and destination operands are obtained and registered separately from performing the sequence of operation instructions, and the sequence of operation instructions is performed using operand sizes defined by the registered size codes, the operation instructions of the sequence not themselves containing size codes.

This embodiment of the present invention also provides a data processor for performing a sequence of operation instructions with variable operand size. The data processor includes a size code register module 120 for receiving and registering respective size codes for different source and destination operands separately from performing the sequence of operation instructions, and a processing module 236 for performing the sequence of operation instructions using operand sizes defined by the size codes registered in the size code register module 120, the operation instructions of said sequence not themselves containing size codes.

The respective operand size attributes for each operand register of the register bank 118, in this example of 8-, 16-, 32-bit operand sizes coded by 2 bits per source/destination register, are contained in a special Register & Operand Size register ('ROSR') 120 and can be set or modified by code inserted by the compiler intermittently, as required by the task context, without the size code information having to form part of every routine or sub-routine. In this embodiment of the invention, obtaining and registering the respective size codes is performed before performing the instruction sequence and in fact can be performed dynamically at runtime of the sequence of operation instructions.

The ROSR 120 specifies the size of the data registers of the bank 118 and hence at the same time the size of the operand. Instead of having the operand size in each instruction of the sequence, the operand size is associated with the register. The operand field of each instruction has the address information (register name or number) identifying the register to operate on, but no size code for the register nor the operand. Using this address information, the load store unit then picks the size from the ROSR 120, as if it were a look-up table. In this way effective sizes for memory or register and for operands of load/store and computation instructions are implicitly defined by the selection of their source or destination register. This eliminates the need to have multiple versions of the instruction routines and sub-routines for different operand sizes.

In typical programs general purpose registers are used continuously with a certain operand size (for example 8-bit, 16-bit or 32-bit) for longer sequences of instructions (10s to 100s). Specifying this size again and again throughout such a sequence means that instruction words carry redundant information. By setting the size codes in the ROSR 120 intermittently, notably at the beginning of such sequences, a single instruction within the code stream sets the register and operand sizes statically for the whole sequence.

The ROSR holds the size information and can be reconfigured on the fly. In addition to improving the code size and code density, the controller performance may be increased, since only as much information is loaded and stored as needed. This means, for example, that 32 bits are not loaded and operated wastefully when only 8 or 16 bits are required. This has a positive effect on energy consumption also.

In US patent specification 4,649,477 both implicit residual size control and explicit size control are allowed. However, even when employing explicit size control, in the system of US patent specification 4,649,477, all the instructions still contain size codes for the operand and register sizes, which the present invention sets only intermittently, releasing code bits for operators and operands.

In more detail, the data processor 100 is shown in Figure 1 as comprising a controller 112 which controls the operation of the different modules of the data processor over a control line 130. In this example, the data processor includes also a shifter control module 122, an arithmetic and arithmetic/logic module 124, a bus controller126 and a condition code logic module 128 controlled by the controller 112 over the line 130. The bus controller receives instructions from an external resource 136, including a compiler/decompiler, to which it also transmits data.

Register and operand size code, inserted by a compiler for example, is obtained and received and stored in the ROSR 120 and provided to a size multiplexer 114. In response to a control signal from the controller 112, the size multiplexer 114 provides register and operand size information over a size bus 116 to the other modules of the data processor.

The bus controller 126 communicates the instructions from the external resource to the register bank 118 the shifter control module and the arithmetic/logic module 124. The arithmetic/logic module 124 performs a sequence of operations on the operand in dependence on the operand size defined by the information from the size multiplexer 114.

Figures 2 and 3 show the elements closely associated with the register and operand size information in more detail. In this example the register bank 118 includes eight registers 202 to 216 of variable effective size, up to 32-bit. The ROSR 120 comprises the same number of size code register elements 220 to 234 as the number of registers in the register bank 118. In this example, each of the ROSR register elements 220 to 234 is a 2-bit element, enabling up to four size codes to be registered for the corresponding one of the registers 202 to 216. The data processing module of the data processor, shown schematically at 236, performs the operation routines and sub-routines in a sequence of operation instructions longer than the operands. The processing module 236 includes system memory 238, a load/store unit 240 which transfers data between the register bank 118 and the memory and the system memory 238. The processing module 236 also includes an instruction decoder 242 for receiving and decoding the operation routines and sub-routines in the sequence of operation instructions. It will be appreciated that the structure shown is simplified for clarity and typically the data processor structure would be more complex and that elements shown in Figure 1 may also form parts of elements shown in Figures 2 and 3.

In use, the effective sizes of each of the registers 202 to 216 and hence the operand sizes are set at the start of an operating instruction sequence by the 2-bit codes registered in the ROSR elements 220 to 234. The register and operand size is set as part of the task context, so the compiler can choose the register layout as a function of the type of task to optimize it to the attributes of a task or even a subroutine. This means routines which need a lot of 32-Bit operands can be chosen to have a larger amount of 32-Bit registers, while others doing character processing, for example, on a byte basis can be chosen to have a larger amount of 8-Bit operands.

The following example below illustrates a potential usage of the ROSR 120 for two different types of task

```
char_fun:     ; Entry into a function requiring a lot of character processing
push   rosr   ; store the actual size information on Stack for future retrieval
mov    #sz66668888,rosr    ; select e.g. four 16-Bit wide and four 8-Bit wide
registers
ld     d0,char1      ; now loads a byte from the memory [1]
  ...
(many character processing operations, heavy use of 8-Bit data)
  ...
pull   rosr   ; restore the previous size information
rts           ; return from subroutine
-------------------- other function
math_fun:     ; Entry to a to a function requiring a lot of 32-Bit processing
push   rosr   ; store the actual size information on Stack for future retrieve
```

```
mov    #sz66662222,rosr    ; select e.g. four 16-Bit wide and four 32-Bit wide
registers
ld     d0,float      ; now loads a 32-Bit word from the memory [2]
  ...
(many arithmetic processing operations, heavy use of 16-Bit data)
  ...
pull   rosr   ; restore the previous size information
rts           ; return from subroutine
```

In function [2] the same instruction code as in [1] now loads a 32-Bit value instead of an 8-Bit one. The change of size code can be input by the compiler when a sequence of instructions can most appropriately be performed with a different register and operand size. Restoring the previous size code enables the size to be correct for the calling context even though the compiler is unaware of the context.

Since also Store instructions are required with the same attribute, size coding in the instruction code would consume 48 combinations, which would represent about 19% of the available combinations for an 8-Bit wide instruction code, whereas with this embodiment of the present invention, in the example given, size coding consumes 16 combinations equivalent to about 6% of the available combinations. The combinations freed up can be used to code more instructions in the compact 8-Bit format.

Figure 4 shows a simplified flow chart of a method 400 of processing data in accordance with an example of an embodiment of the present invention. The method starts a task at 402. At 404, the current content of the ROSR 120 is stored for later re-set at 410. The set of operand size codes are loaded into the ROSR 120 at 406. A sequence of instructions which do not contain operand size information are performed at 408. At 410 the content of the ROSR 120 is re-set so that the calling task can continue or resume in the same context. The task ends at 412.

The methods of embodiments of the invention may also be implemented partially or wholly in a computer program including code portions for performing steps of the method when run on a programmable apparatus, such as a computer system, or enabling a programmable apparatus to perform functions of a device or system according to embodiments of the invention. The computer program may for instance include one or more of: a subroutine, a function, a procedure, an object method, an object implementation, an executable application, an applet, a servlet, a source code, an object code, a shared library/dynamic load library and/or other sequence of instructions designed for execution on a computer system. The computer program may be provided on a data carrier, such as a CD-ROM or other storage device, containing data loadable in a memory of a computer system, the data representing the computer program. The data carrier may further be a data connection, such as a telephone cable or a wireless connection. The description of the information processing architecture has been simplified for purposes of illustration, and it is just one of many different types of appropriate architectures that may be used in embodiments of the invention. It will be appreciated that the boundaries between logic blocks are merely illustrative and that alternative embodiments may merge logic blocks or circuit elements or impose an alternate decomposition of functionality upon various logic blocks or circuit elements.

In the foregoing specification, the invention has been described with reference to specific examples of embodiments of the invention. It will, however, be evident that various modifications and changes may be made therein without departing from the broader spirit and scope of the invention as set forth in the appended claims. For example, the connections may be any type of connection suitable to transfer signals from or to the respective nodes, units or devices, for example via intermediate devices. Accordingly, unless implied or stated otherwise the connections may for example be direct connections or indirect connections.

Where the apparatus implementing the present invention is composed of electronic components and circuits known to those skilled in the art, circuit details have not been explained to any greater extent than that considered necessary for the understanding and appreciation of the underlying concepts of the present invention.

Where the context admits, the terms “front,” “back,” “top,” “bottom,” “over,” “under” and the like in the description and in the claims, if any, are used for descriptive purposes and not necessarily for describing permanent relative positions. It is understood that the terms so used are interchangeable under appropriate circumstances such that the embodiments of the invention described herein are, for example, capable of operation in other orientations than those illustrated or otherwise described herein.

Where the context admits, illustrated hardware elements may be circuitry located on a single integrated circuit or within a same device or may include a plurality of separate integrated circuits or separate devices interconnected with each other. Also, hardware elements in an embodiment of the invention may be replaced by software or code representations in an embodiment of the invention.

Furthermore, it will be appreciated that boundaries described and shown between the functionality of circuit elements and/or operations in an embodiment of the invention are merely illustrative. The functionality of multiple operations may be combined into a single operation, and/or the functionality of a single operation may be distributed in additional operations. Moreover, alternative embodiments may include multiple instances of a particular operation, and the order of operations may be altered in various other embodiments.

In the claims, any reference signs placed between parentheses shall not be construed as limiting the claim. Where the context admits, terms such as “first” and “second” are used to distinguish arbitrarily between the elements such terms describe and these terms are not necessarily intended to indicate temporal or other prioritization of such elements.

Claims

1. A method of processing data comprising performing a sequence of operation instructions with variable operand size,
wherein respective size codes for different source and destination operands are obtained and registered (406) separately from performing (408) said sequence of operation instructions, and said sequence of operation instructions is performed (408) using operand sizes defined by the registered size codes, the operation instructions of said sequence not themselves containing size codes.

2. A method of processing data as claimed in claim 1, wherein said operands are the contents of one or more of a set of registers, said size codes defining effective sizes of respective registers of said set.

3. A method of processing data as claimed in claim 1 or 2, wherein said size codes are suitable for defining said operand sizes as selected ones of 8-bit, 16-bit and 32-bit.

4. A method of processing data as claimed in any preceding claim wherein obtaining and registering said respective size codes (406) is performed before performing said sequence of operation instructions (408).

5. A method of processing data as claimed in claim 4, wherein obtaining and registering said respective size codes (4086) is performed dynamically at runtime of said sequence of operation instructions.

6. A method of processing data as claimed in any preceding claim and including storing (404) initial values of said respective size codes before obtaining and registering (406) new values of said respective size codes, and re-setting (410) said stored initial values of said respective size codes after performing said sequence of operation instructions.

7. A data processor for performing a sequence of operation instructions with variable operand size,
said data processor including a size code register module (120) for receiving and registering respective size codes for different source and destination operands separately from performing said sequence of operation instructions, and a processing module (236) for performing said sequence of operation instructions using operand sizes defined by the size codes registered in said size code register module (120), the operation instructions of said sequence not themselves containing size codes.

8. A data processor as claimed in claim 7 and including a set of operand registers (202-216) for containing said operands, said size codes being arranged to define effective sizes of respective registers of said set.

9. A data processor as claimed in claim 7 or 8, wherein said operand registers (202-216) are configurable with effective sizes as selected ones of 8-bit, 16-bit and 32-bit, and said size codes in said size code register module (120) are arranged to define said selected ones of said effective sizes.

10. A data processor as claimed in any of claims 7 to 9, wherein receiving and registering said respective size codes is arranged to be performed before performing said sequence of operation instructions.

11. A data processor as claimed in any of claims 7 to 10, wherein receiving and registering said respective size codes is arranged to be performed dynamically at runtime of said sequence of operation instructions.

12. A data processor as claimed in any of claims 7 to 11, wherein said size code register module (120) is arranged to store initial values of said respective size codes before receiving and registering new values of said respective size codes, and to re-set said stored initial values of said respective size codes after said processing module performs said sequence of operation instructions.

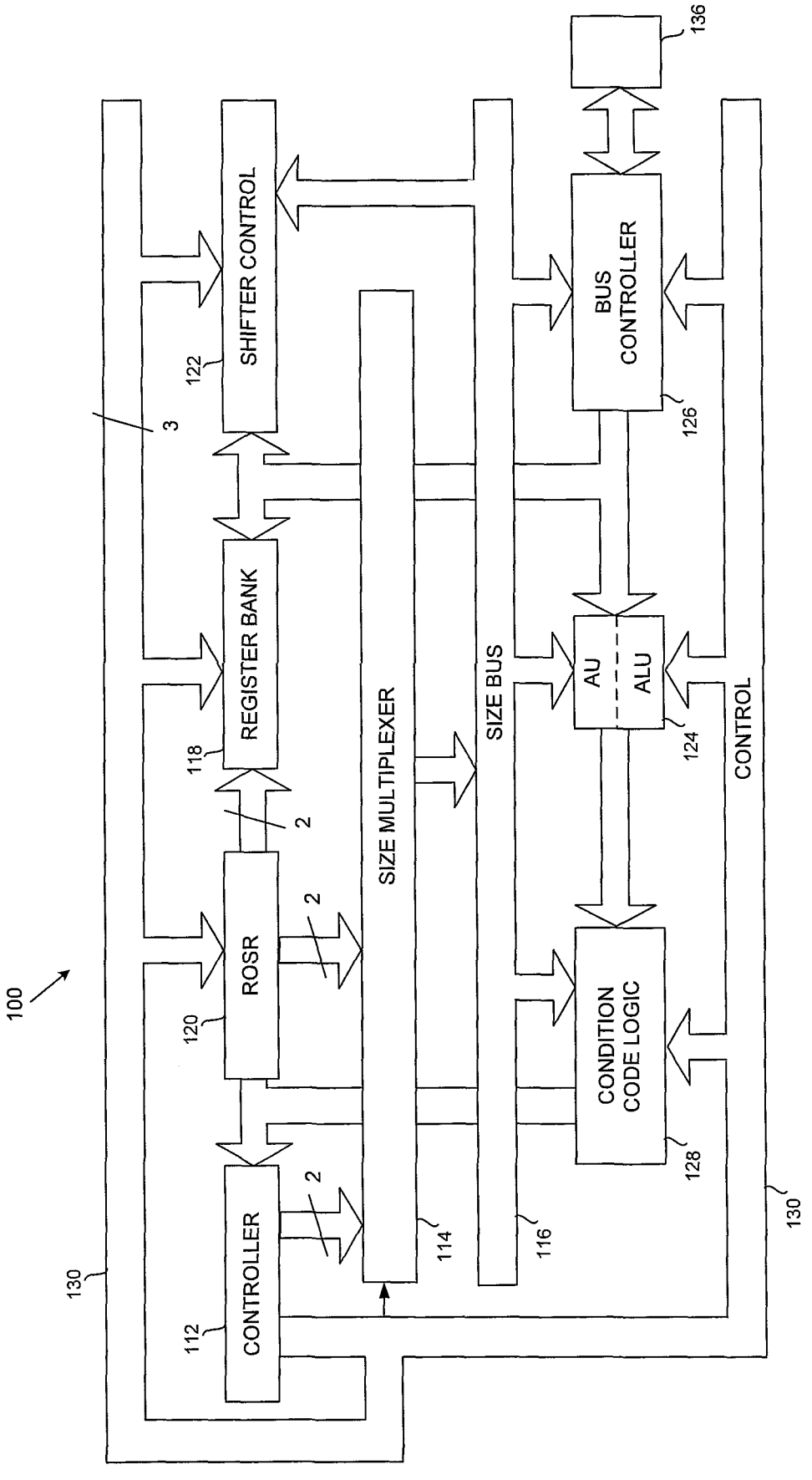
100
130
CONTROLLER
112
ROSR
120
REGISTER BANK
118
SHIFTER CONTROL
122
3
2
2
2
SIZE MULTIPLEXER
114
SIZE BUS
116
AU
ALU
124
BUS CONTROLLER
126
CONDITION CODE LOGIC
128
CONTROL
130
136

Fig. 1

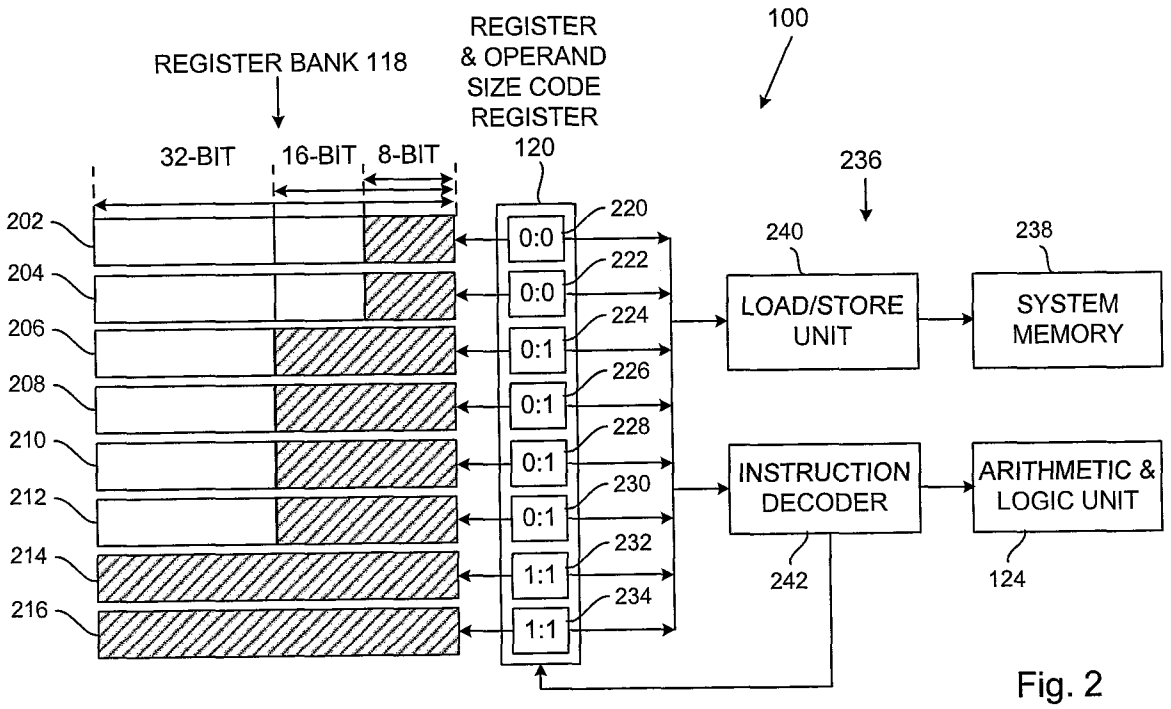
REGISTER BANK 118
REGISTER & OPERAND SIZE CODE REGISTER
120
100
32-BIT
16-BIT
8-BIT
202
204
206
208
210
212
214
216
0:0
0:0
0:1
0:1
0:1
0:1
1:1
1:1
220
222
224
226
228
230
232
234
236
240
238
LOAD/STORE UNIT
SYSTEM MEMORY
INSTRUCTION DECODER
ARITHMETIC & LOGIC UNIT
242
124

Fig. 2

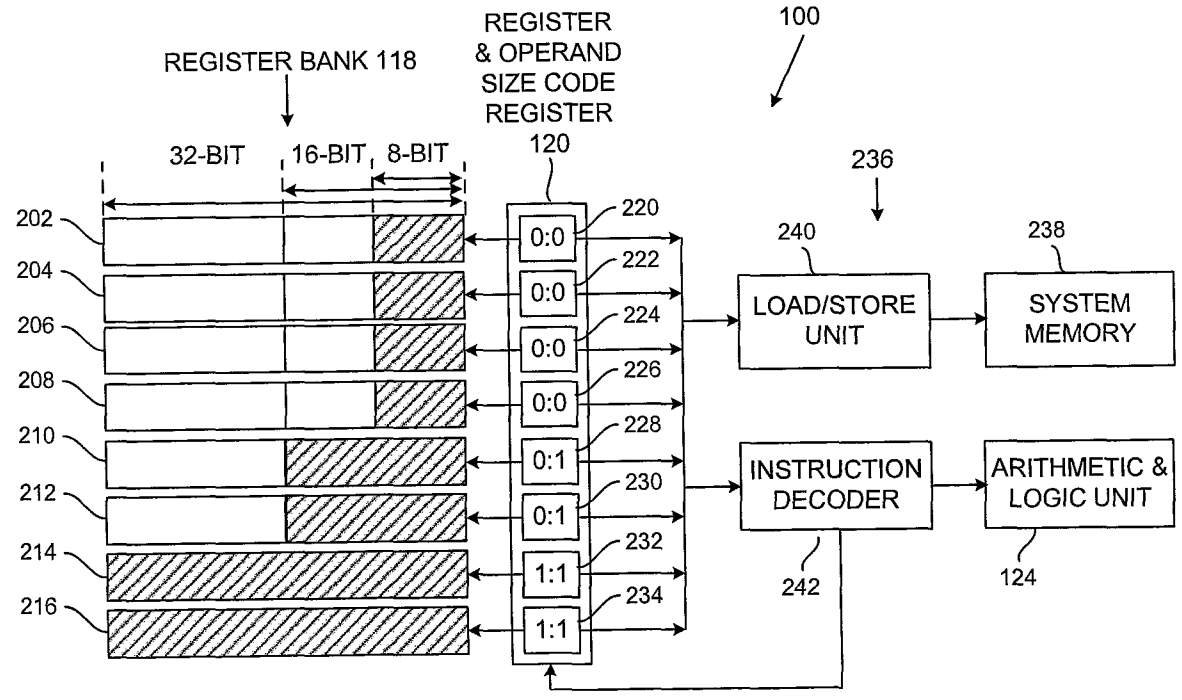
REGISTER BANK 118
REGISTER & OPERAND SIZE CODE REGISTER
120
100
32-BIT
16-BIT
8-BIT
202
204
206
208
210
212
214
216
0:0
0:0
0:0
0:0
0:1
0:1
1:1
1:1
220
222
224
226
228
230
232
234
236
240
238
LOAD/STORE UNIT
SYSTEM MEMORY
INSTRUCTION DECODER
ARITHMETIC & LOGIC UNIT
242
124

Fig. 3

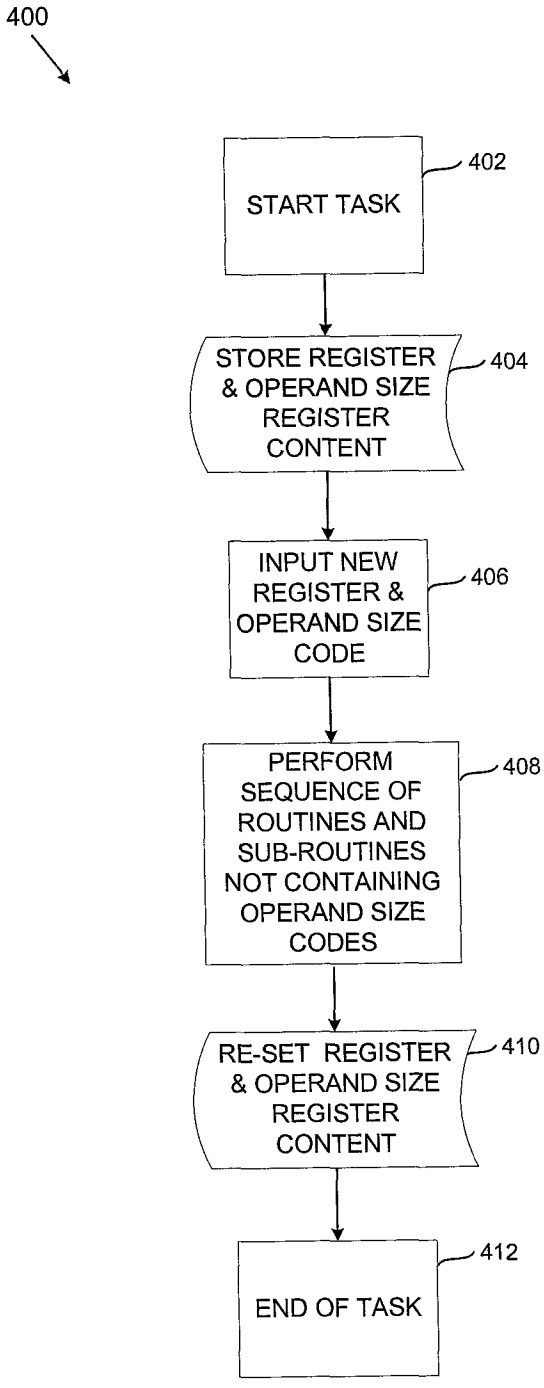
400
START TASK
402
STORE REGISTER & OPERAND SIZE REGISTER CONTENT
404
INPUT NEW REGISTER & OPERAND SIZE CODE
406
PERFORM SEQUENCE OF ROUTINES AND SUB-ROUTINES NOT CONTAINING OPERAND SIZE CODES
408
RE-SET REGISTER & OPERAND SIZE REGISTER CONTENT
410
END OF TASK
412

Fig. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB2009/051345

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/06(2006.01)i, G06F 9/30(2006.01)i, G06F 9/34(2006.01)i

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 : G06F

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

Korean Utility models and applications for Utility models since 1975

Japanese Utility models and applications for Utility models since 1975

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

eKOMPASS(KIPO internal) "variable size operand", "register", "effective size", "size code", "operation", "instruction"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 07191318 B2 (TARUN KUMAR TRIPATHY et al.) 13 March 2007 See abstract; column 2 lines 21-30, 39-44; claim 3.	1-12
A	US 2003-0172252 A1 (G.GLENN HENRY et al.) 11 September 2003 See abstract; paragraphs [0015]-[0016].	1-12
A	US 2005-0033940 A1 (MCGRATH KEVIN J.) 10 February 2005 See abstract; paragraphs [0009]-[0011]. claim 33.	1-12
A	US 2004-0181653 A1 (KEVIN J. MCGRATH et al.) 16 September 2004 See abstract; paragraphs [0007]-[0011].	1-12

☐ Further documents are listed in the continuation of Box C.

☒ See patent family 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 application or patent 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 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
15 DECEMBER 2009 (15.12.2009)	**16 DECEMBER 2009 (16.12.2009)**

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
Government Complex-Daejeon, 139 Seonsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

HAN, Seon Kyoung

Telephone No. 82-42-481-8523

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/IB2009/051345

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 07191318 B2	13.03.2007	US 2008-0022073 A1	24.01.2008
		US 07093099 B2	15.08.2006
		US 07254696 B2	07.08.2007
		US 07596634 B2	29.09.2009
		US 2004-0117496 A1	17.06.2004
		US 2004-0117599 A1	17.06.2004
		US 2004-0117600 A1	17.06.2004
		US 2004-0117602 A1	17.06.2004
US 2003-0172252 A1	11.09.2003	EP 1343075 A3	03.08.2005
		TW 282066 A	01.06.2007
		US 07395412 B2	01.07.2008
US 2005-0033940 A1	10.02.2005	US 06807622 B1	19.10.2004
		US 07284115 B2	16.10.2007
US 2004-0181653 A1	16.09.2004	US 06810476 B2	26.10.2004