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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 October 2016 (20.10.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/167797 A1

(51) International Patent Classification:
G06F 9/45 (2006.01) *G06F 9/30* (2006.01)

(21) International Application Number:
PCT/US2015/026412

(22) International Filing Date:
17 April 2015 (17.04.2015)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) Inventors: **KEAVENY, Thomas A.**; 8000 Foothills Rd., Roseville, California 95747 (US). **GARRO, Diego Valverde**; UltraPark, La Aurora, Heredia Building 8, San Jose, 40101 (CR).

(74) Agents: **CHEN, Lawrence M.** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79 , Fort Collins, CO 80528 (US).

(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, BN, BR, BW, BY, BZ, CA, CH, CL, 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, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, 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, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

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

(54) Title: MORPHED INSTRUCTION ACCORDING TO CONFIGURATION UPDATE

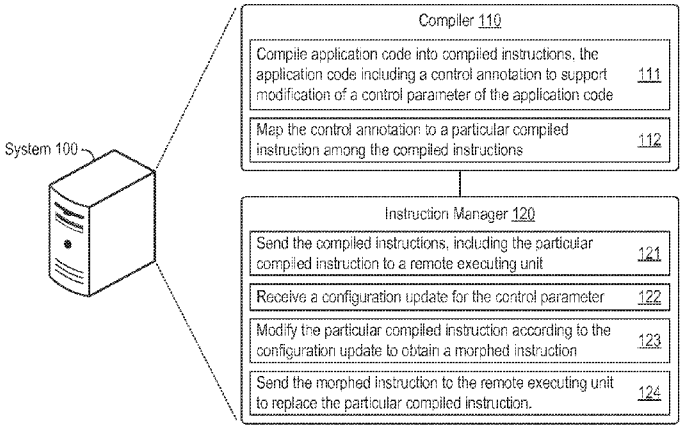

Figure 1

(57) Abstract: In some examples, a method includes sending compiled instructions generated from application code to a remote executing unit, the compiled instructions including a particular compiled instruction that maps to a control parameter for the application code. The method may also include receiving a configuration update specifying an updated value for the control parameter, modifying the particular compiled instruction according to the configuration update to obtain a morphed instruction, and sending the morphed instruction to the remote executing unit to replace the particular compiled instruction.

WO 2016/167797 A1

MORPHED INSTRUCTION ACCORDING TO CONFIGURATION UPDATE

BACKGROUND

[0001] With rapid advances in technology, computing systems are increasingly prevalent in society today. Computing systems execute and support applications that may communicate and process immense amounts of data, many times with performance constraints to meet the increasing demands of users. Increasing the efficiency, speed, and effectiveness of computing systems will further improve user experience.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Certain examples are described in the following detailed description and in reference to the drawings.

[0003] Figure 1 shows an example of a system that supports instruction morphing.

[0004] Figure 2 shows an example of control parameter identification and mapping.

[0005] Figure 3 shows an example of a control mapping that maps a control parameter to a particular instruction field of a compiled instruction.

[0006] Figure 4 shows an example of morphed instruction generation according to a configuration update.

[0007] Figure 5 shows an example of a morphed instruction generated using a control mapping.

[0008] Figure 6 shows an example of logic that a control system may implement to support instruction morphing.

[0009] Figure 7 shows an example of a device that supports instruction morphing.

DETAILED DESCRIPTION

[0010] The disclosure below provides systems, devices, circuitry, logic, and methods that support instruction morphing. As described in greater detail below, instruction morphing may include modifying a processor instruction (e.g., a machine instruction) to reflect an update to a control parameter or system setting. Instruction morphing may thus support real-time and on demand adjusting of system configurations, program behavior, control flow, debug hooks, and more, on an instruction-specific level.

[0011] In some examples, instruction morphing may be utilized by a control entity managing any number of execution units, e.g., processors or devices. Thus, a network control system may employ instruction morphing to modify individual instructions for execution by network devices (e.g., routers) in a communication system. As another example, a master processor may likewise modify individual instructions for execution by any number of slave processors managed by the master processor. By modifying an instruction itself with an updated control parameter, the control entity may reduce additional latency the executing unit may incur had the updated control parameter been implemented instead through, for example, a memory register access by the execution unit. As the morphed instruction itself includes the updated control parameter, the executing unit may incur reduced or no additional latency to effectuate the updated control parameter. Accordingly, instruction morphing may provide a clean, efficient, and scalable mechanism to propagate configuration changes in a system.

[0012] Figure 1 shows an example of a system 100 that supports instruction morphing. The system 100 may implement a control entity that is communicatively linked to an executing unit, which may be remote or local to the system 100. As examples, the system 100 may implement a network control entity controlling remote routing devices, a master processor controlling a slave processor, a programmable logic controller (PLC) controlling devices in an industrial environment, a production control device managing field level

devices in a distributed control system, or any other control entity that provides instructions to a managed executing unit for execution.

[0013] The system 100 in Figure 1 includes a compiler 110 and an instruction manager 120. The compiler 110, instruction manager 120, or both, may be implemented through any combination of hardware, circuitry, executable instructions stored on a machine-readable medium, or various other forms. As described in greater detail below, the compiler 110 and instruction manager 120 may support instruction morphing by linking control parameters annotated in application code to particular compiled instructions generated from the application code that implement the control parameter.

[0014] A control parameter may refer to any control point, value, instruction, setting, or other parameter that affects application behavior. As examples, the control parameter may take the form of a quantifiable value used to evaluate a control flow condition in the application code, a selector to determine a particular element among multiple elements, a pointer value, a comparison operator (e.g., less than, greater than, equal to), an instruction operation (e.g., add, subtract, decrement, increment, multiply, and more), loop parameters, an execution criterion or variable for test conditions in a control flow, a target address, label, or code location to branch execution flow to, or any other element that affects the execution of the application code.

[0015] For example, in Figure 1, the compiler 110 includes the modules 111 and 112, through which the compiler 110 may compile the application code into compiled instructions and map the control parameter to a particular compiled instruction among the compiled instructions. The application code may include a control annotation for a control parameter, the control annotation to support modification of the control parameter. In Figure 1, the instruction manager 120 includes the modules 121, 122, 123, and 124, through which the instruction manager 120 may send the compiled instructions, including the particular compiled instruction, to a remote executing unit; receive a configuration update for the control parameter; modify the particular compiled instruction among the compiled instructions according to the configuration update to obtain a morphed

instruction; and send the morphed instruction to the remote executing unit to replace the particular compiled instruction.

[0016] Some instruction morphing examples are present next through Figures 2-5. Figure 2 shows an example of control parameter identification and mapping. In particular, Figure 2 shows an example of how the compiler 110 and instruction manager 120 may identify a control parameter that can be updated through instruction morphing and map the control parameter to a particular compiled instruction.

[0017] The compiler 110 may access application code 210. The application code 210 may include source code for an application or function to be executed by an executing unit. In particular, the application code 210 may include a control annotation 211 through which the compiler 110 may recognize a control point in the application code 210 that is modifiable through instruction morphing. The control annotation 211 may be set at any point in the application code 210, for example by a developer, through any automated control detection methods, or in other ways. The compiler 110 may parse the application code 210 to identify the control annotation 211.

[0018] In some examples, the compiler 110 identifies the control annotation 211 for a control parameter according to an annotation syntax. As one example annotation syntax, the compiler 110 may recognize control annotations in the application code 210 according to the following format:

```
morph_control <name> (DEFAULT)
```

In the example annotation syntax above, the “morph_control” string may signal the presence and start of a control annotation in the application code 210. The “name” field may specify a control parameter name that identifies a particular control parameter and the “DEFAULT” field may specify a default value for the control parameter. While example fields and syntax format are provided in the example above, the compiler 110 may be configured to identify a control annotation according to any variety of specific syntax implementations.

[0019] To provide a continuing illustration, the application code 210 may include or implement the following code:

```
#define DEFAULT_MAX_ATTEMPTS 10
```

```
while ( counter-- > morph_control FN_max_attempts
(DEFAULT_MAX_ATTEMPTS)) {
    result = issue_request (parameters);
    if (result.valid) break;
}
```

In this FN_max_attempts illustration, the compiler 110 may identify a control annotation within a particular portion of the application code 210 above for the control parameter named FN_max_attempts. The compiler 110 may also determine that the control parameter FN_max_attempts has a default value of 10 through the control annotation (e.g., as set through the variable DEFAULT_MAX_ATTEMPTS).

[0020] The compiler 110 may identify any number of control annotations in the application code 210. To provide another control annotation example, the control annotation 211 may annotate a control parameter in the form of a comparison operator, for example as the annotation "morph_control FN_comparison (GREATER THAN)". The compiler 110 may identify the default operator for control parameter FN_comparison as a greater than operation, and later morphing of a compiled instruction could update the FN_comparison control parameter to a less than operation, greater than or equal to operation, or any other operation. Control annotations in the application code 210 may identify control parameters of various other forms as well.

[0021] The compiler 110 may compile the application code 210 to generate the compiled instructions 221. As part of or as an addition to compiling the application code 210, the compiler 110 may map a control parameter indicated through a control annotation to a particular compiled instruction. To illustrate with respect to Figure 2A, the compiler 110 may map the control parameter specified in the control annotation 211 to one or more of the compiled instructions 221 generated from the application code 210, including the particular compiled instruction among the compiled instructions 221 that is shaded in Figure 2A.

[0022] The compiler 110 may determine the particular compiled instruction(s) to map the control parameter according to a location of the control annotation 211 within the application code 210. For instance, the compiler 110 may track which compiled instruction(s) that are generated from compiling the particular portion (e.g., line or instruction) of application code where the control annotation 211 occurs. In the illustration above, the compiler 110 may compile the code portion for the "while" expression, specifically the code for "while (counter-- > morph_control FN_max_attempts (DEFAULT_MAX_ATTEMPTS)". The compiled instruction(s) resulting from this portion of the application code 210 may be flagged by the compiler 110 for mapping to the control parameter named FN_max_attempts. In generating the compiled instruction that includes or implements the control parameter, the compiler 110 may set a value for control value as the default value specified in the control annotation 211.

[0023] The compiler 110 may map the control parameter indicated by a control annotation to a particular instruction field of a compiled instruction. In that regard, the compiler 110 may map the control parameter to a particular operand or other distinct instruction field of a compiled instruction. As one example, the control parameter FN_max_attempts may be set as a scalar value in an operand field of a branch instruction compiled from the application code 210. The compiler 110 may map the control parameter FN_max_attempts to the particular instruction operand field of the particular branch instruction generated through the compilation process. In other examples, the compiler 110 may map a control parameter to an operation field or address field of a compiled instruction, depending on the particular control parameter being mapped.

[0024] The compiler 110 may track mappings of control parameters to instruction fields of compiled instructions through a control mapping 222. A control mapping 222 may take the form of any data structure that maps a control parameter to a particular compiled instruction or instruction field thereof. The compiler 110 may generate the control mapping 222 as, for example, a table or database, a list of control parameter to instruction field mappings, or in any other data form. In some examples, the control mapping 222 indicates a particular compiled instruction or particular instruction field of the particular

compiled instruction as an offset value, e.g., a number of offset bytes from a particular data address or position. An example of a control mapping is also presented in Figure 3, which is discussed next.

[0025] Figure 3 shows an example of a control mapping 310 that maps a control parameter to a particular instruction field of a compiled instruction. The specific example shown in Figure 3 continues the example code and control annotation discussed above for the control parameter FN_max_attempts. As seen in Figure 3, the control mapping 310 includes a control parameter name field to identify the relevant control parameter, which in Figure 3 is FN_max_attempts.

[0026] The control mapping 310 also includes an instruction field mapping field, which may indicate a particular instruction field of compiled instruction that the control parameter maps to. Figure 3 illustrates conceptually how the control parameter FN_max_attempts maps to the "Op A" field of the original compiled instruction 312, as the control annotation in application code for FN_max_attempts may correspond to the original compiled instruction 312. Continuing the illustration above, the compiler 110 may generate a branch-greater-than machine instruction to compile the "while (counter-- > morph_control FN_max_attempts (DEFAULT_MAX_ATTEMPTS)" portion of code from the application code. Accordingly, the compiler 110 may populate the control mapping 310 to indicate the mapping of the control parameter FN_max_attempts to the "Op A" field of the original compiled instruction 312. The compiler 110 may specify the instruction field mapping to map to the particular instruction field through, for example, an offset value, a memory address, an instruction and field indicator, or through any other data specifying the compiled instruction and instruction field therein.

[0027] The compiler 110 may map a particular control parameter to multiple compiled instructions. Such a scenario may occur when the application code includes multiple control annotations for a particular control parameter. Accordingly, the instruction field mappings field of the control mapping 310 may map the control parameter to the respective instruction field of multiple compiled instructions, such as through multiple offset values. In doing so, the compiler

110 may support morphing of multiple compiled instructions based on the same configuration update.

[0028] Returning to Figure 2, the instruction manager 120 may receive the compiled instructions 221 and the control mappings 222 from the compiler 110. The instruction manager 120 may then send the compiled instructions 221 to an executing unit for execution, and the executing unit may include a processor that supports the instruction set architecture of the compiled instructions 221. The compiled instructions 221 may include specific (e.g., default) values for the control parameters as implemented through the compiled instructions 221, which may be subject to later morphing as illustrated through Figure 4.

[0029] Turning to Figure 4, Figure 4 shows an example of morphed instruction generation according to a configuration update. In Figure 4, the instruction manager 120 receives a configuration update 410. The configuration update 410 may be received by the instruction manager 120 after sending the compiled instructions 221 to an executing unit. The configuration update 410 may specify a change to a control parameter for the application code. In the continuing FN_max_attempts illustration, the configuration update 410 may indicate a change to the value of the FN_max_attempts control parameter from 10 to 20, for example.

[0030] In response to receiving the configuration update 410, the instruction manager 120 may identify a control parameter specified by the configuration update 410 and morph any number of compiled instructions impacted by the configuration update. To do so, the instruction manager 120 may access a control mapping for the control parameter to identify a particular compiled instruction impacted by the configuration update 410, including the particular instruction field that the control parameter maps to. An example of this process is illustrated in Figure 5, which is discussed next.

[0031] Figure 5 shows an example of a morphed instruction generated using a control mapping. Figure 5 continues the FN_max_attempts illustration and with reference to the control mapping 310 described in Figure 3. As seen in Figure 5 and discussed above, the control mapping 310 maps the control parameter FN_max_attempts to the “Op A” instruction field of the original compiled

instruction 312. In response to receiving a configuration update 410 for the FN_max_attempts control parameter, the instruction manager 120 may retrieve the control mapping 310 and identify the original compiled instruction 312 as being affected by the configuration update 410.

[0032] The instruction manager 120 may morph (e.g. modify) the original compiled instruction 312 to obtain a morphed instruction 412. As the configuration update 410 may specify an updated value for the control parameter of the application code, the instruction manager 120 may modify the original compiled instruction 312 by updating the mapped instruction field of the particular compiled instruction to the updated value. In the example shown in Figure 5, the instruction manager 120 modifies the "Op A" instruction field value from 10 (in the original compiled instruction 312) to the updated value of 20 as specified in the configuration update 410. Accordingly, the instruction manager 120 may generate the morphed instruction 412 that includes the updated value of 20 in the "Op A" instruction field, implementing the control parameter change specified by the configuration update 410. For control parameters that map to multiple compiled instructions, the instruction manager 120 may generate multiple morphed instructions by modifying the respective instruction field value of the multiple compiled instructions according to the configuration update 410.

[0033] Returning to Figure 4, the instruction manager 120 may send the morphed instruction 412 to the executing unit to replace a previously sent compiled instruction. As the instruction manager 120 may morph a particular compiled instruction impacted by the configuration update 410 without altering other compiled instructions, the instruction manager 120 may send the morphed instruction 412 to the executing unit without sending the entirety of the compiled instructions 221. That is, the instruction manager 120 may selectively send the morphed instruction 412 without sending others among the compiled instructions 221, e.g., those that remain unchanged by the configuration update 410.

[0034] In doing so, the instruction manager 120 may reduce the amount of instructions that the executing unit must reload to effectuate the configuration update 410. Instead of having to reload the entire application code through the

compiled instructions 221, the executing unit may instead replace a particular compiled instruction with the morphed instruction 412, while unaffected instructions remain intact and unchanged. Instruction morphing may thus provide a graceful, efficient, and flexible mechanism to update the control parameter on an instruction-level basis.

[0035] Figure 6 shows an example of logic 600 that a control system may implement to support instruction morphing. For example, the system 100 may implement the logic 600 as part of the instruction manager 120, the compiler 110, or a combination of both. The control system may implement the logic 600 as hardware, firmware, executable instructions stored on a machine-readable medium, or any combination thereof. The control system may execute the logic 600 as a method or process to support instruction morphing.

[0036] By implementing or executing the logic 600, the control system may send compiled instructions generated from application code to a remote executing unit (602). The compiled instructions may include a particular compiled instruction that maps to a control parameter for the application code. The control system may receive a configuration update specifying an updated value for the control parameter for the application code (604). In response to receiving the configuration update, the control system may modify the particular compiled instruction according to the configuration update to obtain a morphed instruction (606). Then, the control system may send the morphed instruction to the remote executing unit to replace the particular compiled instruction (608).

[0037] In some examples, the control system may identify a control annotation in the application code for the control parameter. The control annotation may occur at a particular point (e.g., line) in the application code. Thus, the control system may map the control parameter to a particular compiled instruction instructions that correspond to the location of the control annotation. In particular, the control system may and map the control parameter to a particular instruction field that implements the control parameter, such as a particular instruction operand storing the value of the control parameter. Thus, the control system may modify the particular compiled instruction according to the

configuration update by setting the particular instruction field of the particular compiled instruction to the updated value specified by the configuration update.

[0038] The configuration update may be set in various ways. In some examples, the control system receives the configuration update as a user-specified change to the control parameter, e.g., through a user interface or user dashboard. In other examples, the configuration updates may be automated according to any number of update criteria, examples of which include time of day, resource bandwidth, consumption, or availability thresholds, environmental triggers, and more.

[0039] The control system may send the morphed instruction to an executing unit logically or physically remote to the control system. The control system may manage multiple executing units, and thus the control system may send the morphed instruction to multiple remote executing units. In some examples, the control system sends the morphed instruction during an initialization state of the remote executing unit, e.g., when a remote device receiving instructions from the control system starts a boot sequence or initially powers up. In some examples, the control system sends the morphed instruction during a run-time state of the executing unit. As such, the control system may utilize instruction morphing to implement real-time configuration changes to the remote executing unit without disrupting real-time processing by the executing unit. For example, in a communication network scenario, a network control device may utilize instruction morphing to propagate updated routing parameters to managed network devices to meet in-line processing requirements of the managed network devices.

[0040] Figure 7 shows an example of a device 700 that supports instruction morphing. The device 700 may include a processor 710. The processor 710 may include a central processing unit (CPU), microprocessor, and/or any hardware device suitable for executing instructions stored on a machine-readable medium.

[0041] The device 700 may include a machine-readable medium 720. The machine-readable medium 720 may be any electronic, magnetic, optical, or other physical storage device that stores executable instructions, such as the

instruction morphing instructions 722 shown in Figure 7. Thus, the machine-readable medium 720 may be, for example, Random Access Memory (RAM), an Electrically-Erasable Programmable Read-Only Memory (EEPROM), a storage drive, an optical disk, and the like.

[0042] The device 700 may execute instructions stored on the machine-readable medium 720 through the processor 710. Executing the instructions may cause the device 700 to perform any combination of the features described herein, including features with respect to the compiler 110 and the instruction manager 120. To illustrate, executing the instruction morphing instructions 722 may cause the device 700 to map a control annotation in application code to a particular compiled instruction among compiled instructions generated from the application code, the control annotation identifying a control parameter for the application code; send the compiled instructions generated from the application code, including the particular compiled instruction, to a remote executing unit; receive a configuration update specifying a change to the control parameter; modify the particular compiled instruction according to the configuration update to obtain a morphed instruction 724, the morphed instruction 724 implementing the change to the control parameter; and send the morphed instruction 724 to the remote executing unit to replace the particular compiled instruction.

[0043] In some examples, executing the instruction morphing instructions 722 may further cause the device 700 to map the control parameter to a particular instruction field of the particular compiled instruction. As another example, executing the instruction morphing instructions 722 may further cause the device 700 to identify the control parameter in the application code by identifying a control annotation for the control parameter in the application code. As yet another example, executing the instruction morphing instructions 722 may further cause the device 700 to send the morphed instruction to multiple remote executing units.

[0044] The methods, devices, systems, and logic described above, including the compiler 110, the instruction manager 120, or both, may be implemented in many different ways in many different combinations of hardware, executable instructions stored on a machine-readable medium, or both. For example, the

compiler 110, the instruction manager 120, or both, may include circuitry in a controller, a microprocessor, or an application specific integrated circuit (ASIC), or may be implemented with discrete logic or components, or a combination of other types of analog or digital circuitry, combined on a single integrated circuit or distributed among multiple integrated circuits. A product, such as a computer program product, may include a storage medium and machine readable instructions stored on the medium, which when executed in an endpoint, computer system, or other device, cause the device to perform operations according to any of the description above.

[0045] The processing capability of the systems, devices, and circuitry described herein, including the compiler 110 and the instruction manager 120, may be distributed among multiple system components, such as among multiple processors and memories, optionally including multiple distributed processing systems. Parameters, databases, and other data structures may be separately stored and managed, may be incorporated into a single memory or database, may be logically and physically organized in many different ways, and may implemented in many ways, including data structures such as linked lists, hash tables, or implicit storage mechanisms. Programs may be parts (e.g., subroutines) of a single program, separate programs, distributed across several memories and processors, or implemented in many different ways, such as in a library, such as a shared library (e.g., a dynamic link library (DLL)). The DLL, for example, may store code that performs any of the system processing described above.

[0046] While various examples have been described above, many more implementations are possible.

CLAIMS

1. A system comprising:

a compiler to:

compile application code into compiled instructions, the application code including a control annotation for a control parameter, the control annotation to support modification of the control parameter; and

map the control parameter to a particular compiled instruction among the compiled instructions; and

an instruction manager to:

send the compiled instructions, including the particular compiled instruction, to a remote executing unit;

receive a configuration update for the control parameter;

modify the particular compiled instruction among the compiled instructions according to the configuration update to obtain a morphed instruction; and

send the morphed instruction to the remote executing unit to replace the particular compiled instruction.

2. The system of claim 1, wherein the compiler is to map the control annotation to a particular compiled instruction field of the particular compiled instruction.

3. The system of claim 2, wherein the configuration update specifies an updated value for the control parameter of the application code; and

wherein the instruction manager is to modify the particular compiled instruction by updating the particular field of the particular compiled instruction to the updated value.

4. The system of claim 1, wherein the compiler is further to map the control parameter to multiple compiled instructions; and

wherein instruction manager is further to:

determine that the configuration update for the control parameter affects multiple compiled instructions, including the particular compiled instruction; and

modify the multiple compiled instructions according to the configuration update to obtain multiple morphed instructions.

5. The system of claim 1, wherein the instruction manager is to send the morphed instruction to the remote executing unit to replace the particular compiled instruction without sending other compiled instructions among the compiled instructions.

6. A method comprising:

sending compiled instructions generated from application code to a remote executing unit, the compiled instructions including a particular compiled instruction that maps to a control parameter for the application code;

receiving a configuration update specifying an updated value for the control parameter;

modifying the particular compiled instruction according to the configuration update to obtain a morphed instruction; and

sending the morphed instruction to the remote executing unit to replace the particular compiled instruction.

7. The method of claim 6, further comprising:

identifying a control annotation in the application code for the control parameter; and

mapping the control parameter to a particular instruction field of the particular compiled instruction.

8. The method of claim 7, wherein modifying the particular compiled instruction comprises setting the particular instruction field of the particular compiled instruction to the updated value specified by the configuration update.

9. The method of claim 6, wherein receiving the configuration update comprises receiving a user-specified change to the control parameter.

10. The method of claim 6, wherein sending the morphed instruction comprises sending the morphed instruction during a run-time state of the executing unit.

11. The method of claim 6, comprising sending the morphed instruction to multiple executing units.

12. A non-transitory machine-readable medium storing executable instructions to:

map a control parameter in application code to a particular compiled instruction among compiled instructions generated from the application code;

send the compiled instructions generated from the application code, including the particular compiled instruction, to a remote executing unit;

receive a configuration update specifying a change to the control parameter;

modify the particular compiled instruction according to the configuration update to obtain a morphed instruction, the morphed instruction implementing the change to the control parameter; and

send the morphed instruction to the remote executing unit to replace the particular compiled instruction.

13. The non-transitory machine-readable medium of claim 12, wherein the executable instructions are further to map the control parameter to a particular instruction field of the particular compiled instruction.

14. The non-transitory machine-readable medium of claim 12, wherein the executable instructions are further to identify the control parameter in the application code by identifying a control annotation for the control parameter in the application code.

15. The non-transitory machine-readable medium of claim 12, wherein the executable instructions are to send the morphed instruction to multiple remote executing units.

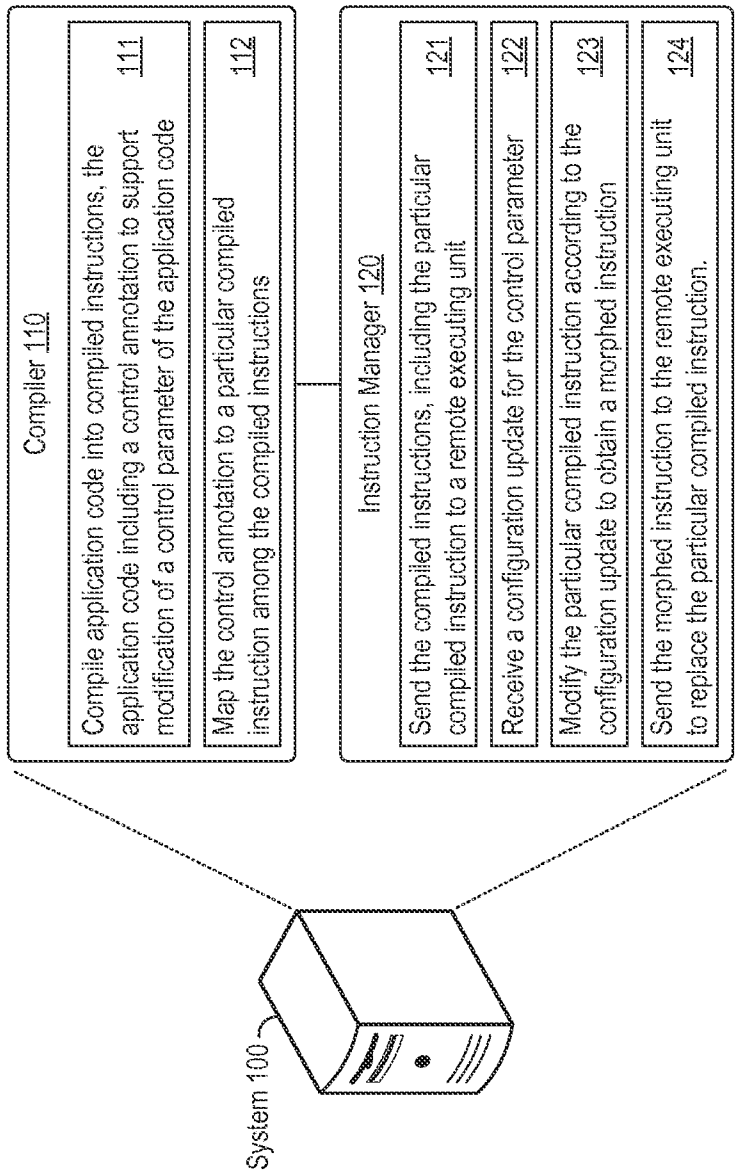
Compiler 110
Compile application code into compiled instructions, the application code including a control annotation to support modification of a control parameter of the application code
111
Map the control annotation to a particular compiled instruction among the compiled instructions
112
Instruction Manager 120
Send the compiled instructions, including the particular compiled instruction to a remote executing unit
121
Receive a configuration update for the control parameter
122
Modify the particular compiled instruction according to the configuration update to obtain a morphed instruction
123
Send the morphed instruction to the remote executing unit to replace the particular compiled instruction.
124
System 100

Figure 1

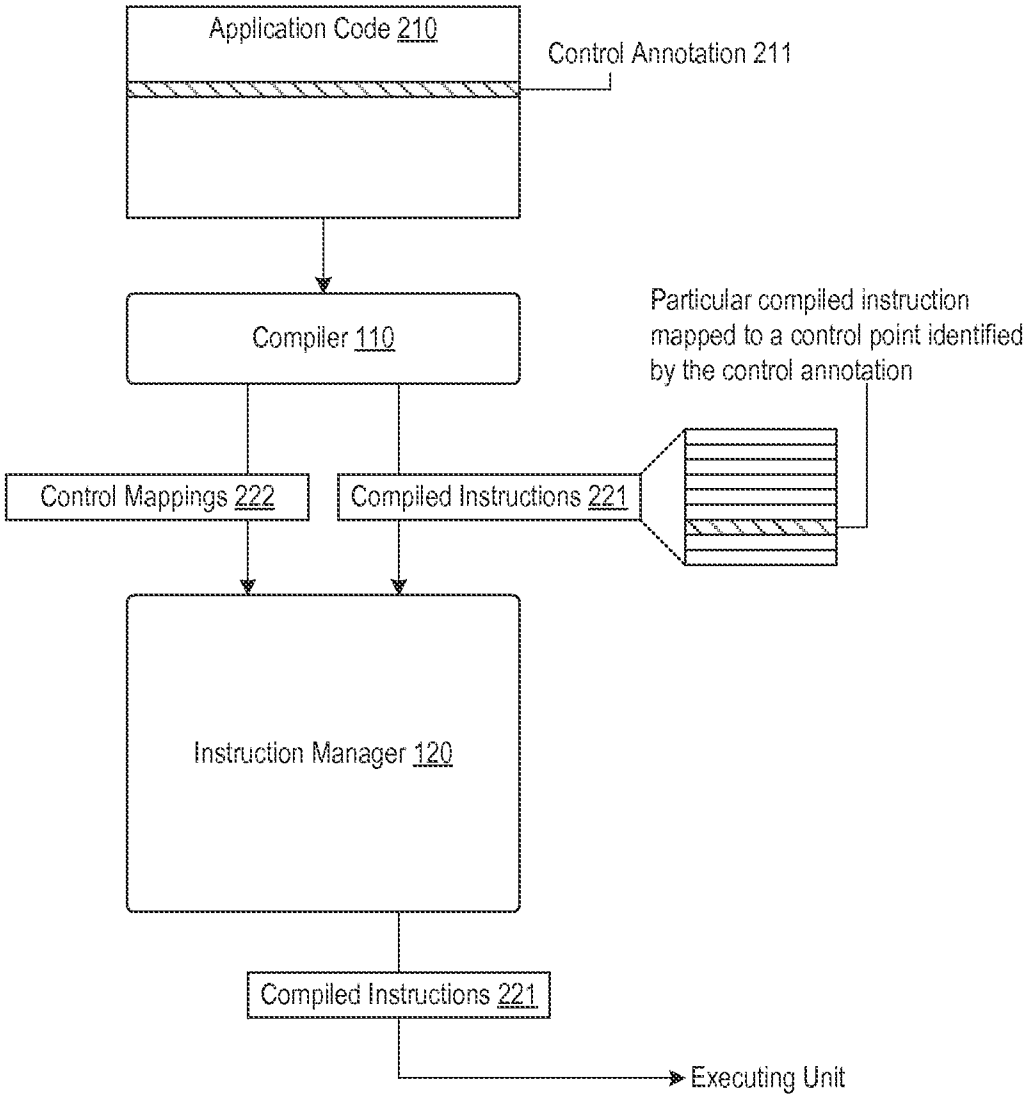
Application Code 210
Control Annotation 211
Compiler 110
Particular compiled instruction mapped to a control point identified by the control annotation
Control Mappings 222
Compiled Instructions 221
Instruction Manager 120
Compiled Instructions 221
Executing Unit

Figure 2

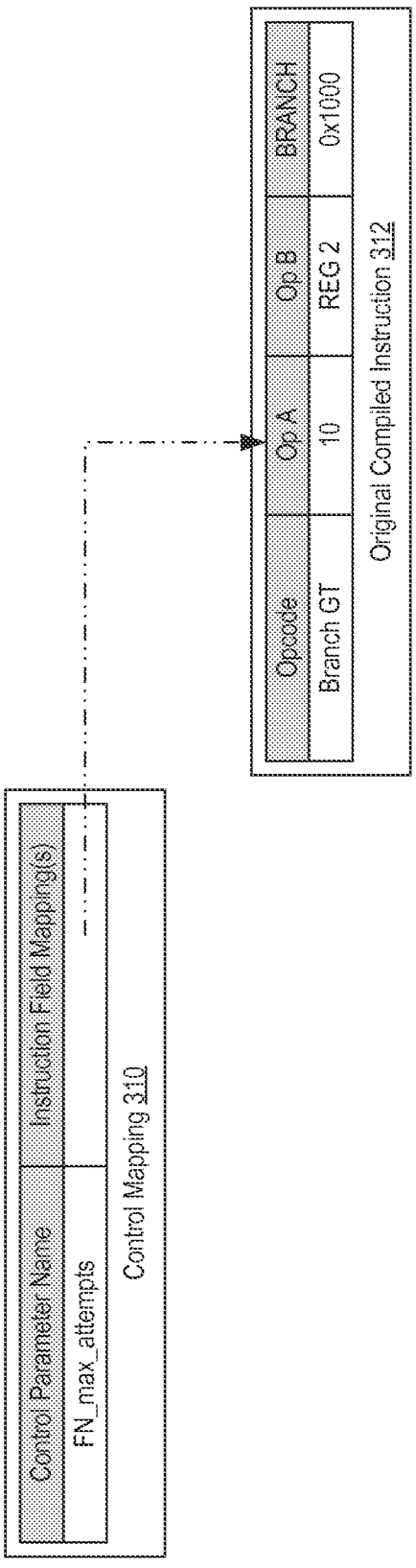
Control Parameter Name
Instruction Field Mapping(s)
FN_max_attempts
Control Mapping 310
Opcode
Op A
Op B
BRANCH
Branch GT
10
REG 2
0x1000
Original Compiled Instruction 312

Figure 3

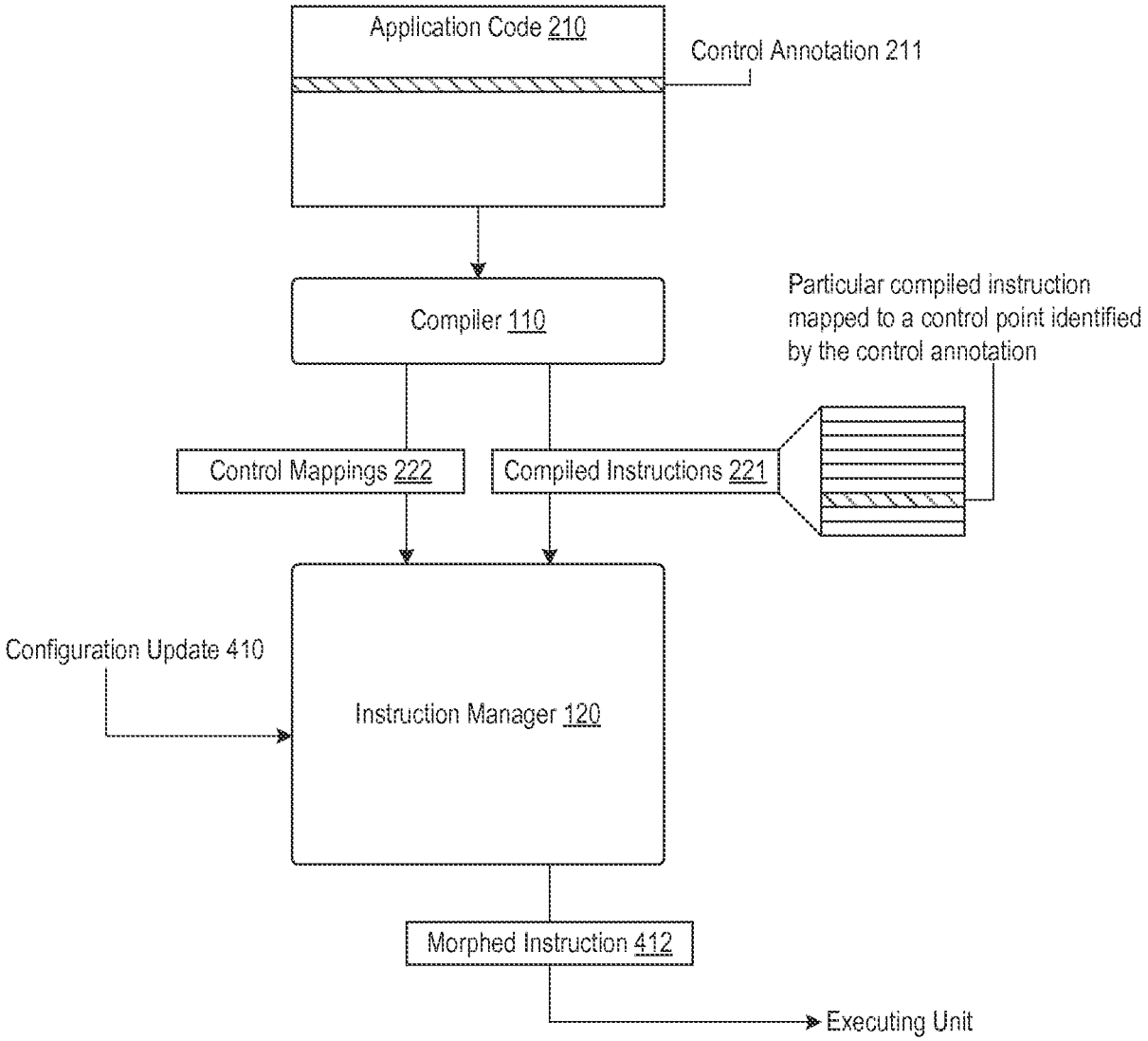
Application Code 210
Control Annotation 211
Compiler 110
Particular compiled instruction mapped to a control point identified by the control annotation
Control Mappings 222
Compiled Instructions 221
Configuration Update 410
Instruction Manager 120
Morphed Instruction 412
Executing Unit

Figure 4

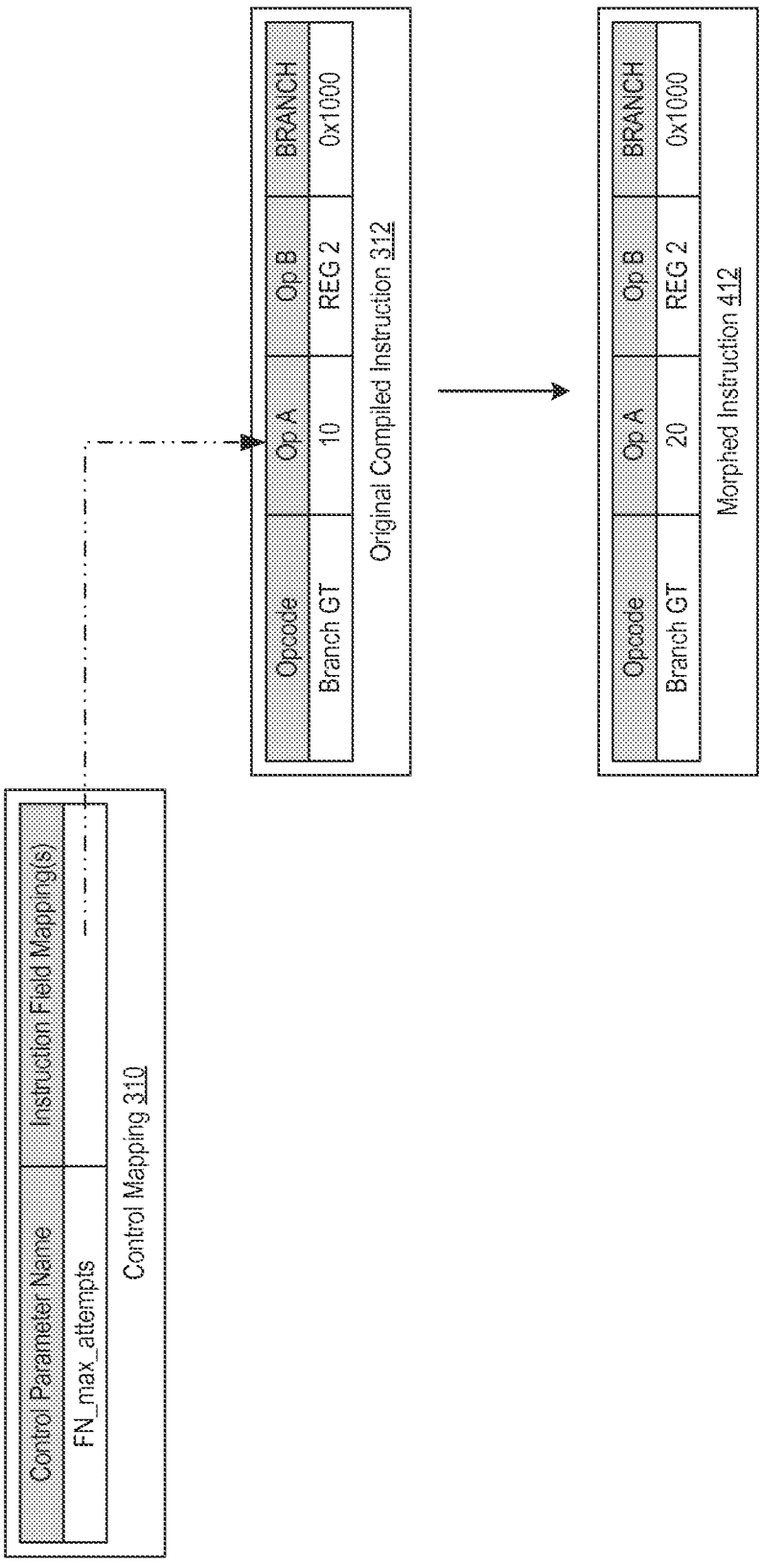
Control Parameter Name
Instruction Field Mapping(s)
FN_max_attempts
Control Mapping 310
Opcode
Op A
Op B
BRANCH
Branch GT
10
REG 2
0x1000
Original Compiled Instruction 312
Opcode
Op A
Op B
BRANCH
Branch GT
20
REG 2
0x1000
Morphed Instruction 412

Figure 5

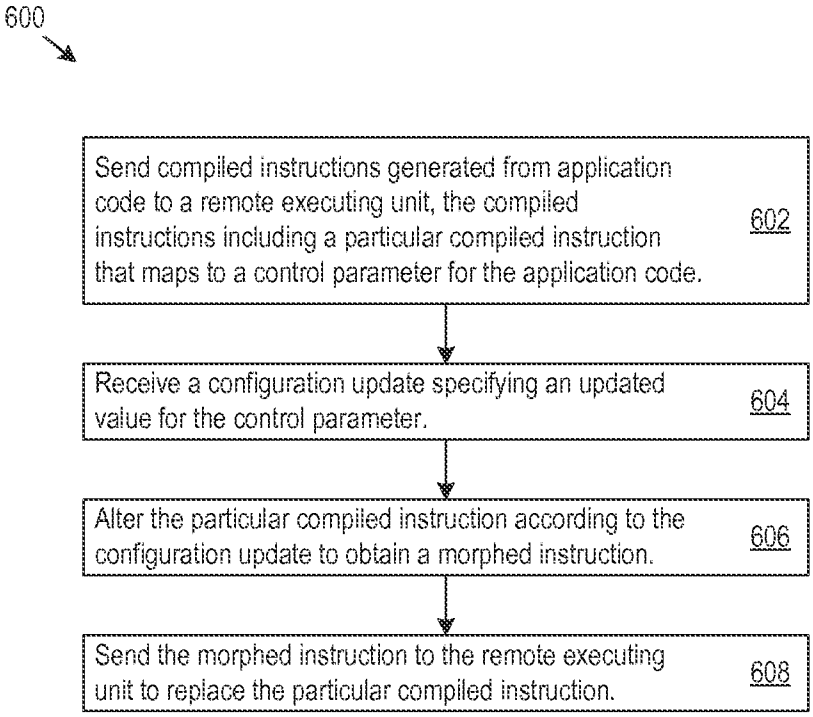
600
Send compiled instructions generated from application code to a remote executing unit, the compiled instructions including a particular compiled instruction that maps to a control parameter for the application code.
602
Receive a configuration update specifying an updated value for the control parameter.
604
Alter the particular compiled instruction according to the configuration update to obtain a morphed instruction.
606
Send the morphed instruction to the remote executing unit to replace the particular compiled instruction.
608

Figure 6

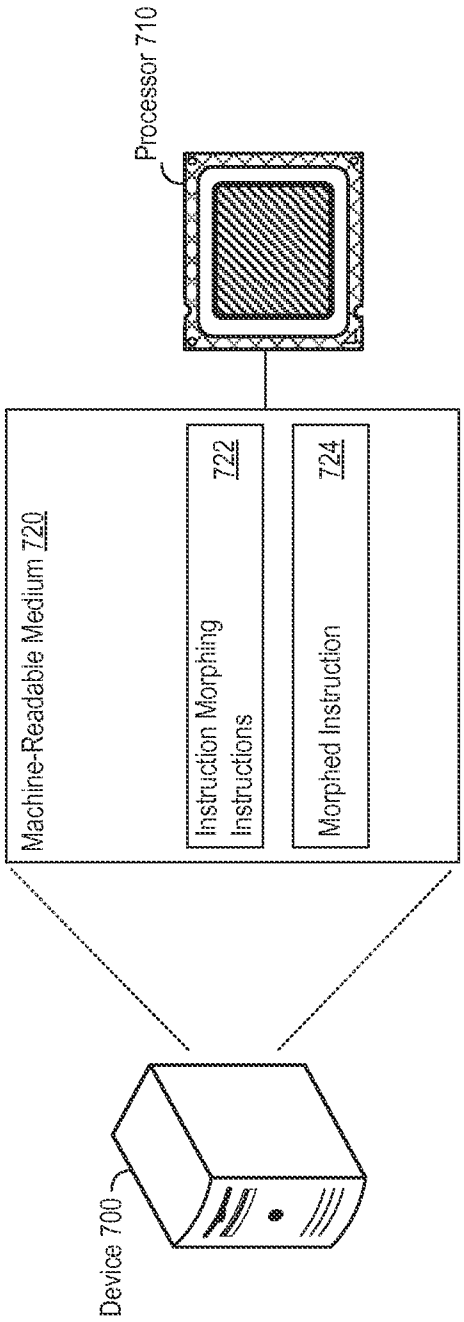
Processor 710
Machine-Readable Medium 720
Instruction Morphing Instructions 722
Morphed Instruction 724
Device 700

Figure 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/026412

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/45(2006.01)i, G06F 9/30(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)
G06F 9/45; G06F 9/30; G06F 9/445; G06F 9/44; G06F 9/318

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
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: instruction, modify, replace, configuration update

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2011-0238961 A1 (BANNING, J. et al.) 29 September 2011 See abstract; paragraphs [0021]-[0031]; and figs. 1 and 2.	1-15
A	US 2002-0073398 A1 (TINKER, J. L.) 13 June 2002 See abstract; paragraphs [0049]-[0054]; and fig. 4.	1-15
A	US 2004-0107416 A1 (BUBAN, G. J. et al.) 03 June 2004 See abstract; paragraphs [0052]-[0065]; and fig. 2.	1-15
A	US 2012-0179897 A1 (CAMPBELL, N. A.) 12 July 2012 See abstract; paragraphs [0017]-[0024]; and fig. 4.	1-15
A	US 06031992 A (CMELIK, R. F. et al.) 29 February 2000 See abstract; col. 14, line 29 - col. 15, line 39; and fig. 2.	1-15

☐ 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 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
11 January 2016 (11.01.2016)	**11 January 2016 (11.01.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
YU, Jintae
Telephone No. +82-42-481-8530

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2015/026412

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2011-0238961 A1	29/09/2011	US 2010-138638 A1	03/06/2010
		US 7698539 B1	13/04/2010
		US 7984277 B2	19/07/2011
		US 8549266 B2	01/10/2013
US 2002-0073398 A1	13/06/2002	CA 2292123 A1	14/06/2000
		CA 2292123 C	08/08/2006
		EP 1014263 A2	28/06/2000
		EP 1014263 A3	18/06/2003
		JP 2000-181725 A	30/06/2000
		US 2003-0212983 A1	13/11/2003
		US 6948164 B2	20/09/2005
US 2004-0107416 A1	03/06/2004	US 7784044 B2	24/08/2010
US 2012-0179897 A1	12/07/2012	None	
US 06031992 A	29/02/2000	None	