FETCH REROUTING IN RESPONSE TO AN EXECUTION-BASED OPTIMIZATION PROFILE

Inventors: Bran Ferren, Beverly Hills, CA (US); W. Daniel Hills, Encino, CA (US); William Henry Mangione-Smith, Kirkland, WA (US); Nathan P. Myhrvold, Medina, WA (US); Clarence T. Tegreene, Bellevue, WA (US); Lowell L. Wood Jr., Livermore, CA (US)

Correspondence Address:
SEARTE LLC
CLARENCE T. TEGREEENE
1756 - 114TH AVE. S.E.
SUITE 110
BELLEVUE, WA 98004 (US)

Assignee: Searete LLC, a limited liability corporation of the State of Delaware

Related U.S. Application Data
Continuation-in-part of application No. 11/214,449, filed on Aug. 29, 2005.
Continuation-in-part of application No. 11/214,458, filed on Aug. 29, 2005.
Continuation-in-part of application No. 11/214,459, filed on Aug. 29, 2005.
Continuation-in-part of application No. 11/292,323, filed on Nov. 30, 2005.
Continuation-in-part of application No. 11/292,207, filed on Nov. 30, 2005.
Continuation-in-part of application No. 11/292,296, filed on Nov. 30, 2005.

Abstract

Embodiments include a device, and a method. In an embodiment, a device includes a processor operable to execute an instruction set, and an execution-optimization circuit. The execution circuit includes an execution circuit for receiving an identification of a first instruction to be fetched from the instruction set for execution by the processor, and for pointing to a second instruction of the instruction set of the processor to be fetched for execution by the processor if indicated by an execution-based optimization profile. The execution-based optimization profile being previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of at least a portion of the instruction set. The execution-optimization circuit may include at least one of a microengine, a micro-programmed circuit, and/or a hardwired circuit.

```
Collecting data corresponding to an execution of at least one instruction of an instruction set from a processor executing the at least one instruction of an instruction set.
Creating an execution-optimization information utilizing the collected data corresponding to the execution of at least one instruction of an instruction set.
Creating a data format modification usable in another execution of the at least one instruction of an instruction set.
Creating a data layout optimization usable in another execution of the at least one instruction of an instruction set.
```

End
Collecting data corresponding to an execution of at least one instruction of an instruction set from a processor executing the at least one instruction of an instruction set.

Creating an execution-optimization information utilizing the collected data corresponding to the execution of at least one instruction of the instruction set, and which is usable in another execution of the at least one instruction of an instruction set.
Collecting data corresponding to an execution of at least one instruction of an instruction set from a processor executing the at least one instruction of an instruction set.

312 Collecting data corresponding to an execution of at least one instruction of an instruction set from a processor actually executing the at least one instruction of an instruction set.

314 Collecting data corresponding to a runtime execution of at least one instruction of an instruction set from a processor executing the at least one instruction of an instruction set.

Creating an execution-optimization information utilizing the collected data corresponding to the execution of at least one instruction of the instruction set, and which is usable in another execution of the at least one instruction of an instruction set.
Collecting data corresponding to an execution of at least one instruction of an instruction set from a processor executing the at least one instruction of an instruction set.

Creating an execution-optimization information utilizing the collected data corresponding to the execution of at least one instruction of the instruction set, and which is usable in another execution of the at least one instruction of an instruction set.

- Creating a modification of the at least one instruction of the instruction set usable in another execution of the at least one instruction of an instruction set.
- Creating a branch predictor modification usable in another execution of the at least one instruction of an instruction set.
- Creating a data format modification usable in another execution of the at least one instruction of an instruction set.
- Creating a data layout optimization usable in another execution of the at least one instruction of an instruction set.
410 First circuit for collecting data corresponding to a runtime execution of at least one instruction of an instruction set from a communications link that is transparent to software executing on the processor and exposed to a processor having a processor instruction set that includes the instruction set.

420 Second circuit for creating an execution-optimization information utilizing the collected data corresponding to the execution of at least one instruction of the instruction set and which is usable in another execution of the at least one instruction of an instruction set.
Gathering data corresponding to an execution of at least one instruction of an instruction set by a processor and in a manner transparent to software executing on the processor.

Creating an execution-based optimization profile utilizing the gathered data and which is useable in a subsequent execution of the at least one instruction of the instruction set by the processor.
Gathering data corresponding to an execution of at least one instruction of an instruction set by a processor and in a manner transparent to software executing on the processor.

- **612** Gathering at least one of dynamic data and/or static data in a manner transparent to software executing on the processor and corresponding to an execution of at least one instruction of an instruction set by a processor.

- **614** Gathering data in a manner transparent to software executing on the processor and corresponding to a normal execution of at least one instruction of an instruction set by a processor.

- **616** Gathering data in a manner transparent to software executing on the processor and corresponding to a runtime execution of at least one instruction of an instruction set by a processor.

Creating an execution-based optimization profile utilizing the gathered data and which is useable a subsequent execution of the at least one instruction of the instruction set by the processor.

- **632** Creating an execution-based optimization profile utilizing the gathered data and which is operable to modify a subsequent execution of the at least one instruction of the instruction set by the processor.

End
Gathering data corresponding to an execution of at least one instruction of an instruction set by a processor and in a manner transparent to software executing on the processor.

Creating an execution-based optimization profile utilizing the gathered data and which is usable a subsequent execution of the at least of one instruction of the instruction set by the processor.

Changing an execution of the at least one instruction of the instruction set in response to the execution-based optimization profile.

Changing a movement of data with respect to the processor in response to the execution-based optimization profile.

Changing a format of data processable by the processor in response to the execution-based optimization profile.

Changing a movement of the at least one instruction of the instruction set toward a processor for execution in response to the execution-based optimization profile.

End
Gathering data corresponding to an execution of at least one instruction of an instruction set by a processor and in a manner transparent to software executing on the processor.

Creating an execution-based optimization profile utilizing the gathered data and which is useable in a subsequent execution of the at least one instruction of the instruction set by the processor.

Changing an execution of the at least one instruction of the instruction set in response to the execution-based optimization profile.

- Substituting at least one other instruction of the instruction set for execution by the processor in place of the at least one instruction of the instruction set in response to the execution-based optimization profile.
- Substituting at least one other instruction of the instruction set for the at least one instruction of the instruction set in a static program in response to the execution-based optimization profile.

End
FIG. 13

600
Start

610
Gathering data corresponding to an execution of at least one instruction of an instruction set by a processor and in a manner transparent to software executing on the processor.

630
Creating an execution-based optimization profile utilizing the gathered data and which is useable in a subsequent execution of the at least one instruction of the instruction set by the processor.

640
Changing an execution of the at least one instruction of the instruction set in response to the execution-based optimization profile.

654 Executing at least one other instruction of the instruction set in response to the execution-based optimization profile.

656 Executing at least one other instruction of the instruction set in response to the execution-based optimization profile and omitting an execution of the at least one instruction.

End
Gathering data corresponding to an execution of at least one instruction of an instruction set by a processor and in a manner transparent to software executing on the processor.

Creating an execution-based optimization profile utilizing the gathered data and which is useable in a subsequent execution of the at least one instruction of the instruction set by the processor.

Changing an execution of the at least one instruction of the instruction set in response to the execution-based optimization profile.

- Omitting an execution of at least one other instruction of the instruction set in response to the execution-based optimization profile.
- Omitting an execution of the at least one instruction of the instruction set in response to the execution-based optimization profile.

End
Gathering data corresponding to an execution of at least one instruction of an instruction set by a processor and in a manner transparent to software executing on the processor.

Creating an execution-based optimization profile utilizing the gathered data and which is useable in a subsequent execution of the at least one instruction of the instruction set by the processor.

Saving the execution-based optimization profile.

- Saving the execution-based optimization profile in an association with the at least one instruction of the instruction set.
- Saving the execution-based optimization profile in an associative cache with the at least one instruction of the instruction set.
FIG. 16

710 Means for gathering data in a manner transparent to software executing on a processor and corresponding to an execution of at least one machine instruction of an instruction set by the processor.

720 Means for creating an execution-based optimization profile utilizing the gathered data and which is useable in a subsequent execution of the at least one machine instruction of the instruction set by the processor.

722 Hardware-implemented means for creating an execution-based optimization profile utilizing the gathered data and which is useable in a subsequent execution of the at least one machine instruction of the instruction set by the processor.

724 Software-implemented means for creating an execution-based optimization profile utilizing the gathered data and which is useable in a subsequent execution of the at least one machine instruction of the instruction set by the processor.
Identifying an instruction to be fetched for execution by a first processor.

Altering an execution of the instruction to be fetched for execution in response to an execution-optimization information.

The execution-optimization information previously generated by a hardware device utilizing data corresponding to a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor.

End
FIG. 19

900 Start

910 Identifying an instruction to be fetched for execution by a first processor.

912 Identifying an instruction to be fetched from an instruction set of a static program for execution by a first processor.

920 Altering an execution of the instruction to be fetched for execution in response to an execution-optimization information.

922 Altering an execution of the instruction to be fetched from an instruction set of a static program for execution in response to an execution-optimization information.

930 The execution-optimization information previously generated by a hardware device utilizing data corresponding to a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor.

End
FIG. 20

900 Start

Identifying an instruction to be fetched for execution by a first processor.

910

Altering an execution of the instruction to be fetched for execution in response to an execution-optimization information.

920

The execution-optimization information previously generated by a hardware device utilizing data corresponding to a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor.

932 The execution-optimization information having been previously generated by a hardware device utilizing data corresponding to a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor, the second processor being at least a substantially same processor as the first processor.

934 The execution-optimization information having been previously generated by a hardware device utilizing data corresponding to a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor, the second processor being at least a substantially different processor from the first processor.

930

End
FIG. 21

900
Start

Identifying an instruction to be fetched for execution by a first processor.

910

Altering an execution of the instruction to be fetched for execution in response to an execution-optimization information.

920

The execution-optimization information previously generated by a hardware device utilizing data corresponding to a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor.

930

The execution-optimization information having been previously generated by a hardware device utilizing data corresponding to a state of the second processor during a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor.

936

The execution-optimization information having been previously generated by a hardware device utilizing data corresponding to an instruction state during a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor.

940

The execution-optimization information having been previously generated by a hardware device utilizing data corresponding to a data relationship during a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor.

942

The execution-optimization information having been previously generated by a hardware device utilizing data corresponding to a data relationship during a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor.

944

End
1020. Means for altering an execution of the instruction from the instruction set of a program in response to an execution-optimization information.

1010. Means for identifying an instruction to be fetched from an instruction set of a program for execution by a first processor.

1030. The execution-optimization information having been generated by a hardware device utilizing data generated by a second processor, and which data corresponds to previous real execution the instruction to be fetched from the instruction set of a program that was transparent to software executing on the second processor.
In response to an identification of a first instruction to be fetched for execution by a processor, fetching a second instruction for execution by the processor if indicated by an execution-based optimization profile.

The execution-based optimization profile previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of at least a portion of the instruction set.
In response to an identification of a first instruction to be fetched for execution by the processor, fetching a second instruction for execution by a processor if indicated by an execution-based optimization profile.

1212 In response to an identification of a first instruction to be fetched from an instruction set of a static program for execution by the processor, fetching a second instruction for execution by the processor if indicated by an execution-based optimization profile.

1214 Fetching the second instruction from an instruction set of a static program for execution by the processor if indicated by an execution-based optimization profile.

1216 Fetching the second instruction from an instruction set of the processor for execution by the processor if indicated by an execution-based optimization profile.

The execution-based optimization profile previously derived by a hardware device utilizing data invisible to software executing on the processor and generated during a runtime execution of at least a portion of the instruction set.

End
In response to an identification of a first instruction to be fetched for execution by the processor, fetching a second instruction for execution by a processor if indicated by an execution-based optimization profile.

- **1218** Fetching the second instruction of the instruction set of a static program if indicated by an execution-based optimization profile linked to the first instruction.
- **1222** Fetching the second instruction for execution by the processor if indicated by an execution-based optimization profile linked to the first instruction.
- **1224** Fetching the second instruction for execution by the processor if indicated by an execution-based optimization profile saved in a cache operably coupled with the processor.

The execution-based optimization profile previously derived by a hardware device utilizing data invisible to software executing on the processor and generated during a runtime execution of at least a portion of the instruction set.

- **1232** The execution-based optimization profile being previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of the first instruction.

**End**
Identifying the first instruction to be fetched for execution by the processor.

In response to an identification of a first instruction to be fetched for execution by a processor, fetching a second instruction for execution by the processor if indicated by an execution-based optimization profile.

The execution-based optimization profile previously derived by a hardware device utilizing data invisible to software executing on the processor and generated during a runtime execution of at least a portion of the instruction set.

Otherwise, fetching the first instruction from the instruction set of a static program for execution by the processor.
1310 Means for selecting a first instruction to be fetched from an instruction set of a static program for execution by a processor.

1320 Means for routing the fetch of the first instruction to a second instruction of the instruction set of a static program if indicated by an execution-based optimization profile.

1330 The execution-based optimization profile having been derived from data invisible to software executing on the processor and generated during a historical execution of the static program.
FETCH REROUTING IN RESPONSE TO AN EXECUTION-BASED OPTIMIZATION PROFILE
CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application is related to, claims the earliest available effective filing date(s) from (e.g., claims earliest available priority dates for other than provisional patent applications; claims benefits under 35 USC § 119(e) for provisional patent applications), and incorporates by reference in its entirety all subject matter of the following listed application(s) (the "Related Applications") to the extent such subject matter is not inconsistent herewith; the present application also claims the earliest available effective filing date(s) from, and also incorporates by reference in its entirety all subject matter of any and all parent, grandparent, great-grandparent, etc. applications of the Related Application(s) to the extent such subject matter is not inconsistent herewith.

Related Applications


[0008] The United States Patent Office (USPTO) has published a notice to the effect that the USPTO’s computer programs require that patent applicants reference both a serial number and indicate whether an application is a continuation or continuation in part, Stephen G. Kunin, Benefit of Prior-Filed Application, USPTO Electronic Official Gazette, Mar. 18, 2003 at http://www.uspto.gov/web/offices/com/sol/og/2003/week 11/pathene.htm. The present applicant entity has provided a specific reference to the application(s) from which priority is being claimed as recited by statute. Applicant entity understands that the statute is unambiguous in its specific reference language and does not require either a serial number or any characterization such as “continuation” or “continuation-in-part.” Notwithstanding the foregoing, applicant entity understands that the USPTO’s computer programs have certain data entry requirements, and hence applicant entity is designating the present application as a continuation in part of its parent applications, but expressly points out that such designations are not to be construed in any way as any type of commentary and/or admission as to whether or not the present application contains any new matter in addition to the matter of its parent application(s).

SUMMARY

[0009] An embodiment provides a device. The device includes a processor operable to execute an instruction set, a communications link exposed to an execution-optimization synthesizer and to the processor, and an execution-optimization synthesizer. The execution-optimization synthesizer includes an execution-optimization synthesizer operable to collect data from the communications link that corresponds to an execution of at least one instruction of the instruction set, and generate an execution-optimization information utilizing the collected data from the communications link and corresponding to the execution of at least one instruction of the instruction set. In addition to the foregoing, other device embodiments are described in the claims, drawings, and text form a part of the present application.

[0010] Another embodiment provides a method. The method includes collecting data corresponding to an execution of at least one instruction of an instruction set from a processor executing the at least one instruction of an instruction set. The method also includes creating an execution-optimization information utilizing the collected data corresponding to the execution of at least one instruction of the instruction set and which is usable in another execution of the at least one instruction of an instruction set. In addition to the foregoing, other method embodiments are described in the claims, drawings, and text form a part of the present application.

[0011] A further embodiment provides a device. The device includes a first circuit for collecting data correspond-
ing to a runtime execution of at least one instruction of an instruction set from a communications link that is transparent to software and exposed to a processor having a processor instruction set that includes the instruction set. The device also includes a second circuit for creating an execution-optimization information utilizing the collected data corresponding to the execution of at least one instruction of the instruction set and which is usable in another execution of the at least one instruction of an instruction set. The circuit for creating the execution-optimization information may include the circuit for collecting data corresponding to an execution. In addition to the foregoing, other device embodiments are described in the claims, drawings, and text form a part of the present application.

[0012] An embodiment provides a device. The device includes a microengine operatively coupled with a processor having an instruction set. The microengine includes a microengine operable gather data in a manner transparent to software executing on the processor and corresponding to a runtime execution of at least a portion of the instruction set by the processor. The microengine is operable to create a runtime-based optimization profile utilizing the gathered dynamic data and which is usable in a subsequent execution of the at least a portion of the instruction set by the processor. The device may include the processor having an instruction set. The device may include a communications link exposed to the microengine. The device may include a communications link exposed to the microengine and transparent to software executing on the processor. The device may include a communications link operably coupled to the microengine and to the processor. In addition to the foregoing, other device embodiments are described in the claims, drawings, and text form a part of the present application.

[0013] Another embodiment provides a method implemented in a hardware device. The method includes gathering data corresponding to an execution of at least one instruction of an instruction set by a processor and in a manner transparent to software executing on the processor. The method includes creating an execution-based optimization profile utilizing the gathered data and which is usable in a subsequent execution of the at least one instruction of the instruction set by the processor. The method may include saving the execution-based optimization profile. The method may include saving the execution-based optimization profile in an association with the at least one instruction of the instruction set. In addition to the foregoing, other method embodiments are described in the claims, drawings, and text form a part of the present application.

[0014] A further embodiment provides a device. The device includes means for gathering data in a manner transparent to software executing on a processor and corresponding to an execution of at least one machine instruction of an instruction set by the processor. The device also includes means for creating an execution-based optimization profile utilizing the gathered data and which is usable in a subsequent execution of the at least one machine instruction of the instruction set by the processor. In addition to the foregoing, other device embodiments are described in the claims, drawings, and text form a part of the present application.

[0015] An embodiment provides a device. The device includes an information store operable to save an execution-optimization information, a first processor, and a hardware circuit. The hardware circuit includes a hardware circuit for altering an execution of a program by the first processor in response to the execution-optimization information. The execution-optimization information being created by a hardware device utilizing data collected from a second processor and corresponding to a previous runtime execution by the second processor of at least a portion of the program that was transparent to any software executing on the second processor. The first processor and the hardware circuit may be formed on a single chip. In addition to the foregoing, other device embodiments are described in the claims, drawings, and text form a part of the present application.

[0016] Another embodiment provides a method. The method includes identifying an instruction to be fetched for execution by a first processor, and altering an execution of the instruction to be fetched for execution in response to an execution-optimization information. The execution-optimization information previously generated by a hardware device utilizing data corresponding to a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor. In addition to the foregoing, other method embodiments are described in the claims, drawings, and text form a part of the present application.

[0017] A further embodiment provides a device. The device includes means for identifying an instruction to be fetched from an instruction set of a program for execution by a first processor. The device also includes means for altering an execution of the instruction from the instruction set of a program in response to an execution-optimization information. The execution-optimization information having been generated by a hardware device utilizing data generated by a second processor, and which data corresponds to a previous real execution the instruction to be fetched from the instruction set of a program that was transparent to software executing on the second processor. In addition to the foregoing, other device embodiments are described in the claims, drawings, and text form a part of the present application.

[0018] An embodiment provides a device. The device includes a processor operable to execute an instruction set, and an execution-optimization circuit. The execution circuit includes an execution circuit for receiving an identification of a first instruction to be fetched from the instruction set for execution by the processor, and for pointing to a second instruction of the instruction set of the processor to be fetched for execution by the processor if indicated by an execution-based optimization profile. The execution-based optimization profile being previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of at least a portion of the instruction set. The execution-optimization circuit may include at least one of a microengine, a micro-programmed circuit, and/or a hardwired circuit. In addition to the foregoing, other method embodiments are described in the claims, drawings, and text form a part of the present application.

[0019] Another embodiment provides a method transparent to software executing on a processor. The method including fetching a second instruction for execution by the processor if indicated by an execution-based optimization
profile in response to an identification of a first instruction to be fetched for execution by a processor. The execution-based optimization profile previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of at least a portion of the instruction set. The method may include identifying the first instruction to be fetched for execution by the processor.

A further embodiment provides a device. The device includes means for selecting a first instruction to be fetched from an instruction set of a static program for execution by a processor. The device also includes means for routing the fetch of the first instruction to a second instruction of the instruction set of a static program if indicated by an execution-based optimization profile. The execution-based optimization profile having been derived from data invisible to software and generated during a historical execution of the static program.

The foregoing is a summary and thus by necessity contains simplifications, generalizations and omissions of detail. Consequently, those skilled in the art will appreciate that the summary is illustrative only and is not intended to be in any way limiting. Other aspects, inventive features, and advantages of the devices and/or processes described herein, as defined by the claims, will become apparent by reference to the drawings and the following detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a partial view of an exemplary device in which embodiments may be implemented;

FIG. 2 illustrates a partial view of an exemplary device in which embodiments may be implemented;

FIG. 3 partially illustrates an association between optimization information and a program and/or data;

FIG. 4 illustrates an exemplary operational flow in which embodiments may be implemented;

FIG. 5 illustrates an alternative embodiment of the exemplary operational flow of FIG. 4;

FIG. 6 illustrates another alternative embodiment of the exemplary operational flow of FIG. 4;

FIG. 7 illustrates a partial view of an exemplary device in which embodiments may be implemented;

FIG. 8 illustrates a partial view of an exemplary device in which embodiments may be implemented;

FIG. 9 illustrates an exemplary operational flow implemented in a hardware device and in which embodiments may be implemented;

FIG. 10 illustrates an alternative embodiment of the exemplary operational flow of FIG. 9;

FIG. 11 illustrates another alternative embodiment of the exemplary operational flow of FIG. 9;

FIG. 12 illustrates a further alternative embodiment of the exemplary operational flow of FIGS. 9 and 11;

FIG. 13 illustrates an alternative embodiment of the exemplary operational flow of FIGS. 9 and 11;

FIG. 14 illustrates another alternative embodiment of the exemplary operational flow of FIGS. 9 and 11;

FIG. 15 illustrates another alternative embodiment of the exemplary operational flow of FIG. 9;

FIG. 16 illustrates a partial view of an exemplary device in which embodiments may be implemented;

FIG. 17 illustrates a partial view of an exemplary device in which embodiments may be implemented;

FIG. 18 illustrates an exemplary operational flow that may implement embodiments;

FIG. 19 illustrates an alternative embodiment of the exemplary operational flow of FIG. 18;

FIG. 20 illustrates an alternative embodiment of the exemplary operational flow of FIG. 18;

FIG. 21 illustrates an alternative embodiment of the exemplary operational flow of FIG. 18;

FIG. 22 illustrates a partial view of an exemplary device in which embodiments may be implemented;

FIG. 23 illustrates a partial view of an exemplary device in which embodiments may be implemented;

FIG. 24 illustrates an exemplary operational flow in which embodiments may be implemented;

FIG. 25 illustrates an alternative embodiment of the exemplary operational flow of FIG. 24;

FIG. 26 illustrates another alternative embodiment of the exemplary operational flow of FIG. 24;

FIG. 27 illustrates a further alternative embodiment of the exemplary operational flow of FIG. 24;

FIG. 28 illustrates an alternative embodiment of the exemplary operational flow of FIG. 24, and includes FIGS. 28A and 28B; and

FIG. 29 illustrates a device in which embodiments may be implemented.

DETAILED DESCRIPTION

In the following detailed description of exemplary embodiments, reference is made to the accompanying drawings, which form a part hereof. In the several figures, like referenced numerals identify like elements. The detailed description and the drawings illustrate exemplary embodiments. Other embodiments may be utilized, and other changes may be made, without departing from the spirit or scope of the subject matter presented here. The following detailed description is therefore not to be taken in a limiting sense, and the scope of the claimed subject matter is defined by the appended claims.

FIG. 1 illustrates an exemplary general-purpose computing system in which embodiments may be implemented, shown as a computing system environment 100. Components of the computing system environment 100 may include, but are not limited to, a computing device 110 having a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a
peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus, also known as Mezzanine bus.

[0053] The computing system environment 100 typically includes a variety of computer-readable media products. Computer-readable media may include any media that can be accessed by the computing device 110 and include both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer-readable media may include computer storage media and communications media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules, or other data. Computer storage media include, but are not limited to, random-access memory (RAM), read-only memory (ROM), electronically erasable programmable read-only memory (EEPROM), flash memory, or other memory technology, CD-ROM, digital versatile disks (DVD), or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage, or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by the computing device 110. Communications media typically embody computer-readable instructions, data structures, program modules, or other data in a modulated data signal such as a carrier wave or other transport mechanism and include any information delivery media. The term "modulated data signal" means a signal that has one more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communications media include wired media such as a wired network and a direct-wired connection and wireless media such as acoustic, RF, optical, and infrared media. Combinations of any of the above should also be included within the scope of computer-readable media.

[0054] The system memory 130 includes computer storage media in the form of volatile and nonvolatile memory such as ROM 131 and RAM 132. A basic input/output system (BIOS) 133, containing the basic routines that help to transfer information between elements within the computing device 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and program modules that are immediately accessible to or presently being operated on by processing unit 120. By way of example, and not limitation, FIG. 1 illustrates an operating system 134, application programs 135, other program modules 136, and program data 137. Often, the operating system 134 offers services to applications programs 135 by way of one or more application programming interfaces (APIs) (not shown). Because the operating system 134 incorporates these services, developers of applications programs 135 need not redevelop code to use the services. Examples of APIs provided by operating systems such as Microsoft’s "WINDOWS" are well known in the art.

[0055] The computing device 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media products. By way of example only, FIG. 1 illustrates a non-removable non-volatile memory interface (hard disk interface) 140 that reads from and writes to non-removable, non-volatile magnetic media, a magnetic disk drive 151 that reads from and writes to a removable, non-volatile magnetic disk 152, and an optical disk drive 155 that reads from and writes to a removable, non-volatile optical disk 156 such as a CD ROM. Other removable/nonremovable, volatile/non-volatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, DVDs, digital video tape, solid state RAM, and solid state ROM. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface, such as the interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable non-volatile memory interface, such as interface 150.

[0056] The drives and their associated computer storage media discussed above and illustrated in FIG. 1 provide storage of computer-readable instructions, data structures, program modules, and other data for the computing device 110. In FIG. 1, for example, hard disk drive 141, is illustrated as storing an operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as, or different from, the operating system 134, application programs 135, other program modules 136, and program data 137. The operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at minimum, they are different copies. A user may enter commands and information into the computing device 110 through input devices such as a microphone 163, keyboard 162, and pointing device 161, commonly referred to as a mouse, trackball, or touchpad. Other input devices (not shown) may include a joystick, game pad, satellite dish, and scanner. These and other input devices are often connected to the processing unit 120 through a user interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 195.

[0057] The computing system environment 100 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device, or other common network node, and typically includes many or all of the elements described above relative to the computing device 110, although only a memory storage device 181 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks such as a personal area network (PAN) (not shown). Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.
When used in a LAN networking environment, the computing system environment 100 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computing device 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or via another appropriate mechanism. In a networked environment, program modules depicted relative to the computing device 110, or portions thereof, may be stored in a remote memory storage device. By way of example, and not limitation, FIG. 1 illustrates remote application programs 185 as residing on computer storage medium 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

FIG. 1 is intended to provide a brief, general description of an illustrative and/or suitable exemplary environment in which embodiments may be implemented. An exemplary system may include the computing system environment 100 of FIG. 1. FIG. 1 is an example of a suitable environment and is not intended to suggest any limitation as to the structure, scope of use, or functionality of an embodiment. A particular environment should not be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in an exemplary operating environment. For example, in certain instances, one or more elements of an environment may be deemed not necessary and omitted. In other instances, one or more other elements may be deemed necessary and added.

In the description that follows, certain embodiments may be described with reference to acts and symbolic representations of operations that are performed by one or more computing devices, such as the computing device 110 of FIG. 1. As such, it will be understood that such acts and operations, which are at times referred to as being computer-executed, include the manipulation by the processing unit of the computer of electrical signals representing data in a structured form. This manipulation transforms the data or maintains them at locations in the memory system of the computer, which reconfigures or otherwise alters the operation of the computer in a manner well understood by those skilled in the art. The data structures in which data is maintained are physical locations of the memory that have particular properties defined by the format of the data. However, while an embodiment is being described in the foregoing context, it is not meant to be limiting as those of skill in the art will appreciate that the acts and operations described hereinafter may also be implemented in hardware.

Embodiments may be implemented with numerous other general-purpose or special-purpose computing devices and computing system environments or configurations. Examples of well-known computing systems, environments, and configurations that may be suitable for use with an embodiment include, but are not limited to, personal computers, handheld or laptop devices, personal digital assistants, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network, minicomputers, server computers, game server computers, web server computers, mainframe computers, and distributed computing environments that include any of the above systems or devices.

Embodiments may be described in a general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc., that perform particular tasks or implement particular abstract data types. An embodiment may also be practiced in a distributed computing environment where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

FIG. 2 illustrates a partial view of an exemplar device 200 in which embodiments may be implemented. The device includes a processor 210, an execution-optimization synthesizer 250, and a communications link 240 exposed to the execution-optimization synthesizer and to the processor. The processor may include any processing unit, and may be described as a central processing unit that controls operation of a computer, such as for example, the processing unit 120 described in conjunction with FIG. 1. The device may also include a hardware resource 220 interconnected with the processor. The hardware resource may be any hardware resource associated and/or interconnected with the processor. In an embodiment, the hardware resource may include one or more caches, illustrated as a cache A (222), a cache B (224), and through a cache N (226). Also, the hardware resource may include a branch predictor (not shown). In another embodiment, the hardware resource 220 may include any other resource associated with the processor, illustrated as other on-chip resource 228. In a further embodiment, the hardware resource includes an off-chip resource, illustrated as an off-chip resource 229. For example, the cache A (222) may be an on-chip L1 cache and the off-chip resource 229 may be an off-chip cache, such as an off-chip L2 cache.

The processor 210 includes a processor operable to execute an instruction set. In an embodiment, the instruction set may include a collection of instructions that the processor can execute. In a further embodiment, the instruction set may include an instruction set architecture of the processor. In another embodiment, the instruction set may include a group of machine instructions and/or computer instructions that the processor can execute. In a further embodiment, the instruction set may be interpreted by the processor. In further embodiment, the instruction set may include a high-level language, an assembly language, and/or a machine code that the processor can execute, with or without a compiling and/or a translation. In an embodiment, an instruction of the instruction set may include a functional instruction, a branching instruction, a memory instruction, and/or other instruction that may be executed by a processor. In another embodiment, an instruction of the instruction set may include a statement or a portion of a statement in a program. In a further embodiment, an instruction group includes at least two statements from a program. A program may include any type of a program, from several lines of instructions, to an application, and to an operating system. In an embodiment, an instruction of the instruction set may include a decoded instruction, a translated instruction, a portion of a translated instruction, and/or a micro-operation.
In a further embodiment, an instruction of the instruction set may include an instruction block, a basic block, a functional block, and/or an instruction module of the instruction set.

[0065] The execution-optimization synthesizer 250 includes an execution-optimization synthesizer operable to collect data from the communications link that corresponds to an execution of at least one instruction of the instruction set. In an embodiment, the data may include certain data items, such as datum, byte, bit, and/or a block that are associated together. The execution-optimization synthesizer is also operable to generate an execution-optimization information utilizing the collected data from the communications link and corresponding to the execution of at least one instruction of the instruction set.

[0066] In an embodiment, the communications link 240 may include at least one of a signal-bearing medium, digital-signal-bearing medium, a light propagation medium, a light propagation medium, an optical fiber, a light guide, a computer readable storage medium, a hardware register, a bus, a memory local to the processor, an interconnection structure, and/or a digital-signal conductor. For example, a computer readable storage medium may be a memory and/or a memory system directly accessible by the processor and the execution-optimization synthesizer. By way of further example, a digital-signal conductor may include any digital signal conducting structure configured to at least transfer digital signals from the processor to the execution-optimization synthesizer. In another embodiment, the communications link includes a signal-bearing medium exposed only to an execution-optimization synthesizer and the processor. In a further embodiment, the communications link includes a signal-bearing medium exposed to an execution-optimization synthesizer and the processor, and transparent to software executing on the processor. In another embodiment, the communications link includes a signal-bearing medium exposed to an execution-optimization synthesizer, to the processor, and to software.

[0067] In an embodiment, the processor 210 and the communications link 240 reside on a single chip, illustrated as a single chip 201. In another embodiment, the processor and the execution-optimization synthesizer 250 reside on a single chip, illustrated as the single chip 201. In a further embodiment, the processor, communications link, and the execution-optimization synthesizer are formed on a single chip, illustrated as the single chip 201.

[0068] In an embodiment, the execution-optimization synthesizer 250 includes a hardware implemented execution-optimization synthesizer. In another embodiment, the execution-optimization synthesizer includes a microengine implemented execution-optimization synthesizer.

[0069] In a further embodiment, the execution-optimization synthesizer 250 operable to collect data from the communications link that corresponds to an execution of at least one instruction of the instruction set includes an execution-optimization synthesizer operable to collect dynamic data from the communications link that corresponds to a runtime execution of at least one instruction of the instruction set. In an embodiment, the data collected by the execution-optimization synthesizer includes at least one of an interpreted instruction of the instruction set, a translated instruction of the instruction set, a decoded instruction of the instruction set, a micro-operation corresponding to at least a portion of the at least one instruction of the instruction set, data correlating to the execution of the at least one instruction of the instruction set, a movement of data correlating to an execution of the at least one instruction of the instruction set, a result of an execution of the at least one instruction of the instruction set, a branch outcome of an execution of the at least one instruction of the instruction set, an exception correlating to an execution of the at least one instruction of the instruction set, a store-to-load dependency correlating an execution of the at least one instruction of the instruction set, a predicted value correlating to an execution of the at least one instruction of the instruction set, and/or a relationship between at least two instructions of the instruction set.

[0070] In an embodiment, the execution-optimization synthesizer 250 operable to collect data from the communications link that corresponds to an execution of at least one instruction of the instruction set includes an execution-optimization synthesizer operable to collect at least one of data transparent to a user, data visible to a user, data transparent to software executing on the processor, data visible to software executing on the processor, and/or data exposed for user manipulation.

[0071] In another embodiment, the execution-optimization synthesizer 250 operable to generate an execution-optimization information utilizing the collected data includes an execution-optimization synthesizer operable to generate an optimization information that is at least one of responsive to the collected data, derived from the collected data, associated with the collected data, and/or using the collected data. In a further embodiment, the execution-optimization synthesizer operable to generate an execution-optimization information corresponding to the execution of at least one instruction of the instruction set includes an execution-optimization synthesizer operable to generate at least one of an execution-environment optimization information, a processor-environment optimization information, a data-environment optimization information, and/or a metadata reporting an execution environment. For example, an execution-environment optimization information may include an indication that an identified micro-op is used frequently and may be advantageously saved in a memory close to the processor 210. Another execution-environment optimization may include one or more versions of the at least one instruction of the instruction set that provides some expected benefit over the original at least one instruction of the instruction set. A memory management system serving the processor may cause one of the versions to be executed transparently instead of the original at least one instruction of the instruction set, such as through a translation lookaside buffer. By way of further example, metadata reporting an execution environment may include tracking information with respect to data objects. For example, certain access predictors may work well with certain data objects, or some objects may not appear to be co-resident in the cache, or may be highly co-resident, or certain pointers in object-oriented systems typically point to specific object types, or specific value predictors have worked well with some data in the past.

[0072] In other embodiments, the execution-optimization synthesizer 250 operable to generate an execution-optimization information utilizing the collected data may include optimizing data handling, which may be by a data class. In
Some instances, a data class may include certain data items (datum, byte, bit, a block, a page) that are used once and never again. In other instances, a data class may include certain data items are used constantly but never written and/or infrequently written. In further data classes, certain data items may be constantly read and written to, or other data items may be often being written but never read. The execution-optimization synthesizer 250 operable to generate an execution-optimization information may predict how a data class will likely be used in the future and/or saves the data items in a manner and/or a location that substantially optimizes utilization of the data items by an instruction group and/or storage of the data items by the computing device. Any suitable type of predictive algorithm providing meaningful results may be used, including a predictive algorithm based on a Bayesian method, and/or a learning algorithm. The prediction may be written to a ghost page associated with a piece of data. A prediction may be straightforward if it is known that the data piece will never be written or read. Each data item will expose what its peculiar flavor is. This may be implemented down to the size of a single cache line, or even below the cache line.

[0073] In further embodiments, the execution-optimization synthesizer 250 operable to generate an execution-optimization information utilizing the collected data may provide storage mobility for data items that are associated together in a substantial disequilibrium based upon a shared fate, a shared nature, an entanglement to a page and/or line of similarly handled data. The data item may include one or more extra bits (tag) on end of a data item that may indicate its size, nature (written but never read, read but never written, read once in the life of the program, used by at least two threads). In a further embodiment, an indicator may say which code relates with to the data item. This may be used for doing storage assignment. For example, if the data item includes a semaphore that is used across multiple threads, that should be known and the data item managed accordingly. Most data is associated with a particular body of code and assigned to a storage unit together. By watching that, these assignments can be done together between the L-cache and the D-cache.

[0074] In an embodiment, the execution-optimization synthesizer 250 further includes an execution-optimization synthesizer operable to save the optimization information. The optimization information may be saved close to the processor 210, for example in an on-chip resource such as the cache A (222), or in the off-chip resource 229, such as a system memory or storage medium. In another embodiment, the execution-optimization synthesizer further includes an execution-optimization synthesizer operable to save the optimization information in an association with the at least one instruction of the instruction set.

[0075] In an embodiment, the device 200 includes a computing device, such as for example, the computing device 110 of the computing system environment 100 of FIG. 1. In a further embodiment, the computing device includes at least one of desktop computing device, a laptop-computing device, a portable computing device, and/or a supercomputing device.

[0076] FIG. 3 partially illustrates an association between optimization information and a program and/or data. An instruction set architecture is illustrated as an instruction set architecture 265, and related compiled programs are illustrated as an operating system 272 and an application program 276. The application program 276 may be a compiled application program or a compilable application program. Also illustrated is a data set 274.

[0077] The execution-optimization information generated by the execution-optimization synthesizer 250 may be associated with the at least one instruction of the instruction set of a program, an application, and/or a module that includes the at least one instruction. In the case of data, the execution-optimization information generated by the execution-optimization synthesizer may be associated with data received for processing by the execution, data produced by the execution, the at least one instruction of the instruction set that processed the data, and/or other related matter. FIG. 3 illustrates certain embodiments of an association of the execution-optimization information with the at least one instruction of the instruction set. The ghost pages 282 that include the execution-optimization information pertaining to the operating system 272 may be virtually and/or physically associated in an information storage with the operating system. The information storage may include a non-volatile memory structure. For example, the ghost pages may be saved in the same file as the operating system. When the operating system is loaded into system memory, the ghost pages may remain in the information storage, or may be, such as for example, also loaded into system memory, or loaded into an inboard memory. In another embodiment, an execution-optimization information 284 pertaining to a data set 274 is associated in an information storage with the data set. In a further embodiment, an execution-optimization profile 286 is associated in an information storage with an application 276.

[0078] In an embodiment, a ghost page of the ghost pages 282 containing the execution-optimization information may be associated with a selected page of a program or data whose content corresponds to the generation of the execution-optimization information, such as for example, a selected page containing the instruction of the operating system 272, a selected page containing the data of the data set 274, and/or a selected page containing the application program 276. By way of further example, data in a ghost page of the ghost pages 282 may indicate that a branch instruction on an identified line of an associated selected page of an application should not be taken. In another embodiment, a file containing the execution-optimization information 284 may be associated with a file containing the data set.

[0079] The illustrated embodiments of the ghost page 282, the execution-optimization information 284, and the execution-optimization profile 286 respectively associated with the operating system 272, the data 274, and the application program 276 are intended only to be illustrative and are not limiting. In another embodiment for example, the ghost pages 282 may be associated with the application 276, or the data set 274.

[0080] FIG. 4 illustrates an exemplary operational flow 300 in which embodiments may be implemented. In an embodiment, the operational flow may be implemented in the computing system environment 100 of FIG. 1 and/or the device 200 of FIG. 2. After a start operation, the operational flow moves to an accumulation operation 310. The accu-
mulation operation collects data corresponding to an execution of at least one instruction of an instruction set from a processor executing the at least one instruction of an instruction set. An enhancement operation 330 creates an execution-optimization information utilizing the collected data corresponding to the execution of at least one instruction of the instruction set and which is usable in another execution of the at least one instruction of an instruction set. The operational flow then moves to an end operation.

[0081] FIG. 5 illustrates an alternative embodiment of the exemplary operational flow 300 of FIG. 4. The accumulation operation 310 may include at least one additional operation. The at least one additional operation may include an operation 312 and/or an operation 314. The operation 312 collects data corresponding to an execution of at least one instruction of an instruction set from a processor actually executing that at least one instruction of an instruction set. The operation 314 collects data corresponding to a runtime execution of at least one instruction of an instruction set from a processor executing the at least one instruction of an instruction set.

[0082] FIG. 6 illustrates another alternative embodiment of the exemplary operational flow 300 of FIG. 4. The enhancement operation 330 may include at least one additional operation. The at least one additional operation may include an operation 332, an operation 334, an operation 336, and/or an operation 338. The operation 332 creates a modification of the at least one instruction of the instruction set usable in another execution of the at least one instruction of an instruction set. For example, the creating a modification of the at least one instruction may include creating one or more versions of the instruction wherein each version may provide some expected benefit over the original version, or saving a decoded version of a frequently fetched at least one instruction to save a future decoding of the frequently fetched instruction. The operation 334 creates a branch prediction modification usable in another execution of the at least one instruction of an instruction set. The operation 336 creates a data format modification usable in another execution of the at least one instruction of an instruction set. The operation 338 creates a data layout optimization usable in another execution of the at least one instruction of an instruction set. The device may include a communications link 540 exposed to the microengine. In another embodiment, the device may include the communications link 540 exposed to the microengine and the processor executing on the processor. In a further embodiment, the device may include the communications link operably coupled to the microengine and the processor. In another embodiment, the communications link may include an interconnection structure.

[0083] FIG. 7 illustrates a partial view of an exemplary device 400 in which embodiments may be implemented. The device includes a first circuit 410 for collecting data corresponding to a runtime execution of at least one instruction of an instruction set from a communications link that is transparent to software executing on the processor and exposed to a processor having a processor instruction set that includes the instruction set. The device also includes a second circuit 420 for creating an execution-optimization information utilizing the collected data corresponding to the execution of at least one instruction of the instruction set and which is usable in another execution of the at least one instruction of an instruction set. In an embodiment, the second circuit for creating the execution-optimization information includes the first circuit for collecting data corresponding to an execution.

[0084] FIG. 8 illustrates a partial view of an exemplary device 500 in which embodiments may be implemented. The device includes a microengine 550 operatively coupled with a processor 510 having an instruction set. The processor may include any processor, such as for example, the processing unit 120 described in conjunction with FIG. 1. The processor may be described as a central processing unit that controls operation of a computer. In an embodiment, the device may include an internal bus 530 providing a parallel data transfer path between the processor and the hardware resource 220.

[0085] The microengine 550 includes a microengine operable to gather data in a manner transparent to software executing on the processor 510 and corresponding to a runtime execution of at least a portion of the instruction set by the processor. The microengine is also operable to create a runtime-based optimization profile utilizing the gathered dynamic data and which is usable in a subsequent execution of the at least a portion of the instruction set by the processor.

[0086] In an embodiment, the microengine 550 may include a microengine operable to gather at least one of dynamic data and/or static data in a manner transparent to software executing on the processor and corresponding to a runtime execution of at least a portion of the instruction set by the processor 510.

[0087] In another embodiment, the device 500 may further include the processor 510 having an instruction set. In a further embodiment, the processor and the microengine 550 are formed on a chip, illustrated as a single chip 501. In an embodiment, the device may further include a communication link 540 exposed to the microengine. In another embodiment, the device may include the communication link 540 exposed to the microengine and transparent to software executing on the processor. In a further embodiment, the device may include the communication link operably coupled to the microengine and to the processor. In another embodiment, the communication link may include an interconnection structure.

[0088] FIG. 9 illustrates an exemplary operational flow 600 implemented in a hardware device and in which embodiments may be implemented. In an embodiment, the operational flow may be implemented in the computing system environment 100 of FIG. 1, and/or the device 500 of FIG. 8. After a start operation, the operational flow moves to a harvesting operation 610. The harvesting operation gathers data corresponding to an execution of at least one instruction of an instruction set by a processor and in a manner transparent to software executing on the processor. An improvement operation 630 creates an execution-based optimization profile utilizing the gathered data and which is usable in a subsequent execution of the at least one instruction of the instruction set by the processor. In an embodiment, the execution-based optimization profile may enhance a future execution of the at least one instruction by increasing an efficiency of the execution, reducing cache misses, reducing exceptions, reducing storage used, and/or reducing energy consumed. The operational flow then proceeds to an end operation.

[0089] FIG. 10 illustrates an alternative embodiment of the exemplary operational flow 600 of FIG. 9. The harvesting operation 610 may include at least one additional operation. The at least one additional operation may include an operation 612, an operation 614, and/or an operation 616. The operation 612 gathers at least one of dynamic data...
and/or static data in a manner transparent to software executing on the processor and corresponding to an execution of at least one instruction of an instruction set by a processor. The operation 614 gathers data in a manner transparent to software executing on the processor and corresponding to a normal execution of at least one instruction of an instruction set by a processor. The operation 616 gathers data in a manner transparent to software executing on the processor and corresponding to a runtime execution of at least one instruction of an instruction set by a processor. The improvement operation 630 may include at least one additional operation, such as an operation 632. The operation 632 creates an execution-based optimization profile utilizing the gathered data and which is operable to modify a subsequent execution of the at least one instruction of the instruction set by the processor.

[0090] FIG. 11 illustrates another alternative embodiment of the exemplary operational flow 600 of FIG. 9. The operational flow may include at least one additional operation. The at least one additional operation may include a modification operation 640. The modification operation changes an execution of the at least one instruction of the instruction set in response to the execution-based optimization profile.

[0091] The modification operation 640 may include at least one additional operation. The at least one additional operation may include an operation 642, an operation 644, and/or an operation 646. The operation 642 changes a movement of data with respect to the processor in response to the execution-based optimization profile. For example, changing a movement of data may include changing a movement of data toward and/or away from the processor. For example, frequently read data may be stored in a memory close to the processor and infrequently read data may be stored in a memory far from the processor. By way of further example, frequently written or rewritten data may be stored in a memory close to the processor and infrequently read data may be stored in a memory far from the processor. The operation 644 changes a format of data processable by the processor in response to the execution-based optimization profile. For example, the operation 644 may save data translated from one format to another, such as from big-endian to little-endian, or floating-point formats. The operation 646 changes a movement of the at least one instruction of the instruction set toward a processor for execution in response to the execution-based optimization profile.

[0092] FIG. 12 illustrates a further alternative embodiment of the exemplary operational flow 600 of FIGS. 9 and 11. The modification operation 640 may include at least one additional operation. The at least one additional operation may include an operation 648, and/or an operation 652. The operation 648 substitutes at least one other instruction of the instruction set for execution by the processor in place of the at least one instruction of the instruction set in response to the execution-based optimization profile. The operation 652 substitutes at least one other instruction of the instruction set for the at least one instruction of the instruction set in a static program in response to the execution-based optimization profile.

[0093] FIG. 13 illustrates an alternative embodiment of the exemplary operational flow 600 of FIGS. 9 and 11. The modification operation 640 may include at least one additional operation. The at least one additional operation may include an operation 654. The operation 654 executes at least one other instruction of the instruction set in response to the execution-based optimization profile. The operation 654 may include at least one additional operation, such as an operation 656. The operation 656 executes at least one other instruction of the instruction set in response to the execution-based optimization profile and emits an execution of the at least one instruction.

[0094] FIG. 14 illustrates another alternative embodiment of the exemplary operational flow 600 of FIGS. 9 and 11. The modification operation 640 may include at least one additional operation. The at least one additional operation may include an operation 658, and/or an operation 662. The operation 658 emits an execution of at least one other instruction of the instruction set in response to the execution-based optimization profile. The operation 662 emits an execution of the at least one instruction of the instruction set in response to the execution-based optimization profile.

[0095] FIG. 15 illustrates another alternative embodiment of the exemplary operational flow 600 of FIG. 9. The operational flow may include at least one additional operation, such as the operation 670. The operation 670 saves the execution-based optimization profile in an association with the at least one instruction of the instruction set. The operation 672 may include at least one additional operation, such as the operation 674. The operation 674 saves the execution-based optimization profile in an associative cache with the at least one instruction of the instruction set.

[0096] FIG. 16 illustrates a partial view of an exemplary device 700 in which embodiments may be implemented. The device includes means 710 for gathering data in a manner transparent to software executing on the processor and corresponding to an execution of at least one machine instruction of an instruction set by the processor. The device includes means 720 for creating an execution-based optimization profile utilizing the gathered data and which is useable in a subsequent execution of the at least one machine instruction of the instruction set by the processor.

[0097] In an alternative embodiment the means 710 includes hardware-implemented means 712 for gathering data in a manner transparent to software executing on a processor and corresponding to an execution of at least one machine instruction of an instruction set by the processor. In another alternative embodiment, the means 720 may include at least one additional means. The at least one additional means may include hardware-implemented means 722 for creating an execution-based optimization profile utilizing the gathered data and which is useable in a subsequent execution of the at least one machine instruction of the instruction set by the processor. The at least one additional means may include software-implemented means 724 for creating an execution-based optimization profile utilizing the gathered data and which is useable in a subsequent execution of the at least one machine instruction of the instruction set by the processor.

[0098] FIG. 17 illustrates a partial view of an exemplary device 800 in which embodiments may be implemented. The
device includes an information store 840 operable to save an execution-optimization information 842, a first processor 810, and a hardware circuit 850. The hardware circuit includes a circuit for altering an execution of a program by the first processor in response to the execution-optimization information. The execution-optimization information includes execution-optimization information created by a hardware device utilizing data collected from a second processor (not shown). The collected data corresponding to a previous runtime execution by the second processor are at least a portion of the program that was transparent to any software executing on the second processor.

[0099] In an embodiment, the execution-optimization information 842 may include the execution-optimization information generated by the execution-optimization synthesizer 250 of FIG. 2. In another embodiment, the execution-optimization information may include at least one of the ghost pages 272, the execution-optimization information 274, and/or the execution-optimization profile 276 described in conjunction with FIGS. 2 and 3. In an alternative embodiment, the first processor 810 includes a first processor operable to execute an instruction set and operably coupled to the information store 840. In another embodiment, the hardware circuit for altering an execution of a program includes a hardware circuit for altering an execution of a program and operably coupled to the information store. In a further embodiment, the hardware circuit includes a hardware circuit operably coupled to the processor.

[0100] In an embodiment, the hardware circuit 850 includes a hardware circuit for copying the execution-optimization information from the information store to at least one page of the program and operably coupled to the first processor. For example, the memory operably coupled to the first processor may include the hardware resource 220, such as the on-chip cache B 224, or the off-chip resource 229, such as an off-chip cache or an on-board memory or an outboard storage.

[0101] In a further embodiment, the hardware circuit 850 for altering an execution of a program by the first processor 810 in response to the execution-optimization information includes a hardware circuit for causing an alteration of an execution of at least one instruction of an instruction set of a static program by the first processor in response to the execution-optimization information. In another embodiment, the hardware circuit for altering an execution of a program by the first processor in response to the execution-optimization information includes a hardware circuit for altering an execution of at least one instruction of an instruction set of a dynamic program by the first processor in response to the execution-optimization information. In an embodiment, the hardware circuit for altering an execution of a program by the first processor in response to the execution-optimization information includes a hardware circuit for altering an execution of at least one instruction of an instruction set of a program by the first processor in response to the execution-optimization information. In another embodiment, the hardware circuit for altering an execution of a program by the first processor in response to the execution-optimization information includes a hardware circuit for altering a movement of at least one instruction of the program toward the first processor in response to the execution-optimization information.

[0102] In some instances, the altering an execution of a program by the first processor in response to the execution-optimization information may include directly altering an execution of a program by the first processor in response to the execution-optimization information. In other instances, the altering an execution of a program by the first processor in response to the execution-optimization information may include causing an alteration of an execution of a program by the first processor in response to the execution-optimization information. The execution-optimization information

[0103] In an embodiment, the execution-optimization information includes execution-optimization information created by a hardware device (not shown) utilizing data collected from a second processor (not shown) that at least substantially a same processor as the first processor 810. For example, the execution-optimization information used to alter a current execution of a program by the first processor 810 may have been created during a prior execution of the program by the first processor. In another embodiment, the execution-optimization information includes execution-optimization information created by a hardware device utilizing data collected from a second processor that is at least a substantially different processor from the first processor. For example, the execution-optimization information used to alter a current execution of a program by the first processor may have been created during a prior execution of the program by a completely different second processor, which may be a processor running in a completely different computing device.

[0104] In an embodiment, the information store includes at least a portion of a cache. In another embodiment, the information store includes at least one of an L-cache or a D-cache. In a further embodiment, the information store includes at least one of a volatile memory or a non-volatile memory. In a further embodiment, the information store includes a computer readable medium. In another embodiment, the information store may include a non-volatile outboard storage, such as magnetic disk storage.

[0105] In another embodiment, the first processor 810 and the hardware circuit 850 are formed on a single chip, illustrated as a single chip 801. In a further embodiment, the first processor 810 and the information store 840 are formed on a single chip, illustrated as a single chip 801.

[0106] FIG. 18 illustrates an exemplary operational flow 900 that may implement embodiments. In an embodiment, the operational flow may be implemented in the computing system environment 100 of FIG. 1, and/or the device 800 of FIG. 17. After a start operation, the operational flow moves to an instruction determination operation 910. The instruction determination operation identifies an instruction to be fetched for execution by a first processor. An optimization operation 920 alters an execution of the instruction to be fetched for execution in response to an execution-optimization information. The execution-optimization information
was previously generated by a hardware device utilizing data corresponding to a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor. The flow then moves to an end operation.

**[0107]** FIG. 19 illustrates an alternative embodiment of the exemplary operational flow 900 of FIG. 18. The instruction determination operation 910 may include at least one additional operation, such as an operation 912. The operation 912 identifies an instruction to be fetched from an instruction set of a static program for execution by a first processor. The optimization operation 920 may include at least one additional operation, illustrated as the operation 922. The operation 922 alters an execution of the instruction to be fetched from an instruction set of a static program for execution in response to an execution-optimization information.

**[0108]** FIG. 20 illustrates an alternative embodiment of the exemplary operational flow 900 of FIG. 18. The execution-optimization information 930 may include at least one additional embodiment. The at least one additional embodiment may include an execution-optimization information 932 and/or an execution-optimization information 934. The execution-optimization information 932 includes execution-optimization information having been previously generated by a hardware device utilizing data corresponding to a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor, the second processor being at least a substantially same processor as the first processor. The execution-optimization information 934 may include an execution-optimization information having been previously generated by a hardware device utilizing data corresponding to a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor, the second processor being at least a substantially different processor from the first processor. In an embodiment, the second processor may be a processor of a multiprocessor computing device that includes the first processor. In another embodiment, the second processor may be a processor of a second computing device that is a separate and distinct computing device from a first computing device that includes the first processor.

**[0109]** FIG. 21 illustrates an alternative embodiment of the exemplary operational flow 900 of FIG. 18. The execution-optimization information 930 may include at least one additional embodiment. The at least one additional embodiment may include an execution-optimization information 936, an execution-optimization information 938, and/or an execution-optimization information 942. The execution-optimization information 936 includes an execution-optimization information having been previously generated by a hardware device utilizing data corresponding to a state of the second processor during a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor. The execution-optimization information 938 includes an execution-optimization information having been previously generated by a hardware device utilizing data corresponding to an instruction state during a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor. The execution-optimization information 942 includes an execution-optimization information having been previously generated by a hardware device utilizing data corresponding to a data relationship during a real execution of the instruction to be fetched by a second processor that was transparent to software executing on the second processor.

**[0110]** FIG. 22 illustrates a partial view of an exemplary device 1000 in which embodiments may be implemented. The device includes means 1010 for identifying an instruction to be fetched from an instruction set of a program for execution by a first processor. The device also includes means 1020 for altering an execution of the instruction from the instruction set of a program in response to an execution-optimization information. The execution-optimization information 1030 having been generated by a hardware device utilizing data generated by a second processor, and which data corresponds to a previous real execution the instruction to be fetched from the instruction set of a program that was transparent to software executing on the second processor.

**[0111]** FIG. 23 illustrates a partial view of an exemplary device 1000 in which embodiments may be implemented. The device includes a processor 1010 operable to execute an instruction set, and an execution-optimization circuit 1150. The execution-optimization circuit includes an execution-optimization circuit for receiving an identification of a first instruction to be fetched from the instruction set for execution by the processor. The execution-optimization circuit also includes an execution-optimization circuit for pointing to a second instruction of the instruction set of the processor to be fetched for execution by the processor if indicated by an execution-based optimization profile saved in an information store. In an embodiment, the information store may include an information store 1140A close to the processor, such as on the same chip. In another embodiment, the information store may include an information store 11403 that is an off-processor-chip resource. The execution-based optimization profile being previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of at least a portion of the instruction set. The execution-based optimization profile may indicate that the second instruction be fetched in a direct and/or indirect manner. For example, the execution-based optimization profile may modify the next instruction address calculation causing the second instruction to be fetched in lieu of the first instruction. In another example, the execution-based optimization profile may signal and/or point the next instruction address calculation causing the second instruction to be fetched in lieu of the first instruction. In another example, the execution-based optimization profile may provide information usable in determining whether to fetch the second instruction in lieu of the first instruction.

**[0112]** In an embodiment, the execution-optimization circuit 1150 includes at least one of a microengine, a microprogrammed circuit, and/or a hardwired circuit. In another embodiment, the execution-optimization circuit includes an execution-optimization portion of a control unit of the processor. In another further embodiment, the processor and the execution-optimization circuit are formed on a chip, illustrated as the chip 1101.

**[0113]** In an embodiment, the execution-optimization circuit 1150 includes an execution-optimization circuit for receiving an identification of a first instruction to be fetched from the instruction set of a program for execution by the processor. The program may be a static program or a dynamic program. In another embodiment, the execution-optimization circuit includes an execution-optimization circuit for pointing to a second instruction of the instruction set of the processor to be fetched for execution by the processor.
if indicated by an execution-based optimization profile associated with the first instruction. In a further embodiment, the execution-optimization circuit includes an execution-optimization circuit for pointing to a second instruction of the instruction set of the processor to be fetched for execution by the processor if indicated by an execution-based optimization profile associated with another instruction of the instruction set. In an embodiment, the execution-based optimization profile includes the execution-based optimization profile being previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of at least a portion of an instruction set of a static program.

[0114] FIG. 24 illustrates an exemplary operational flow 1200 in which embodiments may be implemented. In an embodiment, the operational flow may be implemented in the computing system environment 100 of FIG. 1, and/or the device 1100 of FIG. 23. After a start operation, the operational flow moves to a substitution operation 1210. In response to an identification of a first instruction to be fetched for execution by a processor, the substitution operation fetches a second instruction for execution by the processor if indicated by an execution-based optimization profile. The execution-based optimization profile 1230 includes an execution-based optimization profile previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of at least a portion of the instruction set. The flow then proceeds to an end operation. The operational flow is transparent to software executing on the processor.

[0115] FIG. 25 illustrates an alternative embodiment of the exemplary operational flow 1200 of FIG. 24. The substitution operation 1210 may include at least one additional operation. The at least one additional operation may include an operation 1212, an operation 1214, and/or an operation 1216. In response to an identification of a first instruction to be fetched from an instruction set of a static program for execution by the processor, the operation 1212 fetches a second instruction for execution by the processor if indicated by an execution-based optimization profile. In a response to an identification of a first instruction to be fetched from an instruction set of a static program for execution by the processor, the operation 1214 fetches a second instruction from an instruction set of a static program for execution by the processor if indicated by an execution-based optimization profile. In response to an identification of a first instruction to be fetched from an instruction set of a static program for execution by the processor, the operation 1216 fetches the second instruction from an instruction set of the processor for execution by the processor if indicated by an execution-based optimization profile.

[0116] FIG. 26 illustrates another alternative embodiment of the exemplary operational flow 1200 of FIG. 24. The substitution operation 1210 may include at least one additional operation. The at least one additional operation may include an operation 1218, operation 1222, and/or an operation 1224. In response to an identification of a first instruction to be fetched from an instruction set of a static program for execution by the processor, the operation 1218 fetches the second instruction of the instruction set of a static program if indicated by an execution-based optimization profile linked to the first instruction. In response to an identification of a first instruction to be fetched from an instruction set of a static program for execution by the processor, the operation 1222 fetches a second instruction for execution by the processor if indicated by an execution-based optimization profile linked to the first instruction. In response to an identification of a first instruction to be fetched from an instruction set of a static program for execution by the processor, the operation 1224 fetches the second instruction for execution by the processor if indicated by an execution-based optimization profile saved in a cache operably coupled with the processor. The execution-based optimization profile 1230 may include at least one additional execution-based optimization profile, such as an execution-based optimization profile 1232. The execution-based optimization profile 1232 includes an execution-based optimization profile being previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of the first instruction.

[0117] FIG. 27 illustrates a further alternative embodiment of the exemplary operational flow 1200 of FIG. 24. The operational flow may include at least one additional operation. The at least one additional operation may include a fetch next instruction operation 1205, and/or an operation 1240. The fetch next instruction operation 1205 identifies the first instruction to be fetched for execution by the processor. If fetching a second instruction for execution by the processor is not indicated by the execution-based optimization profile, the operation 1240 fetches the first instruction from the instruction set of a static program for execution by the processor.

[0118] FIG. 28 illustrates an alternative embodiment of the exemplary operational flow 1200 of FIG. 24, and includes FIGS. 28A and 28B. FIG. 28A illustrates an instruction group 1250, which includes a basic block 100, a basic block 200, and a basic block 300. The execution sequence of the instruction group includes a jump from the last instruction of the basic block 100 to the first instruction of the basic block 200, a jump from the last instruction of the basic block 200 to the first instruction of the basic block 300, and a jump from the last instruction of the basic block 300 to the first instruction of the basic block 100 until a condition is met.

[0119] FIG. 28B illustrates a modification of the instruction group 1250 in response to a hardware-generated and historically-based execution-optimization strategy. For example, the optimization strategy may have been configured to respond to data collected from a processor indicating that a result produced by execution of the basic block 300 is never read. In an embodiment, the optimization strategy may include using at least one of the execution-optimization information described in conjunction with FIGS. 2, 4, 17, and 18; a runtime-based optimization profile described in conjunction with FIG. 8; and/or an execution-based optimization profile described in conjunction with FIGS. 9, 23, and 24. If indicated by the optimization strategy, the execution sequence of the instruction group includes a jump from the last instruction of the basic block 100 to the first instruction of the basic block 200, and jump from the last instruction of the basic block 200 to the first instruction of the basic block 100 until a condition is met. The execution of the basic block 300 is omitted or not executed.

[0120] FIG. 29 illustrates a device 1300 in which embodiments may be implemented. The device includes means 1310 for selecting a first instruction to be fetched from an instruction set of a static program for execution by a processor. The device also includes means 1320 for routing the fetch of the first instruction to a second instruction of the instruction set of a static program if indicated by an execution-based optimization profile. The execution-based optimization profile 1330 includes an execution-based optimi-
The foregoing detailed description has set forth various embodiments of the devices and/or processes via the use of block diagrams, flow diagrams, operation diagrams, flowcharts, illustrations, and/or examples. Insofar as such block diagrams, operation diagrams, flowcharts, illustrations, and/or examples contain one or more functions and/or operations, it will be understood by those within the art that each function and/or operation within such block diagrams, operation diagrams, flowcharts, illustrations, or examples can be implemented, individually and/or collectively, by a wide range of hardware, software, firmware, or virtually any combination thereof.

In addition, those skilled in the art will appreciate that the mechanisms of the subject matter described herein are capable of being distributed as a program product in a variety of forms, and that an illustrative embodiment of the subject matter described herein applies equally regardless of the particular type of signal-bearing media used to actually carry out the distribution. Examples of a signal-bearing media include, but are not limited to, the following: recordable type media such as floppy disks, hard disk drives, CD ROMs, digital tape, and computer memory; and transmission type media such as digital and analog communication links using TDM or IP based communication links (e.g., packet links).

It will be understood by those within the art that, in general, terms used herein, and especially in the appended claims (e.g., bodies of the appended claims) are generally intended as “open” terms (e.g., the term “including” should be interpreted as “including but not limited to,” the term “having” should be interpreted as “having at least,” the term “includes” should be interpreted as “includes but is not limited to,” etc.). It will be further understood by those within the art that if a specific number of an introduced claim recitation is intended, such an intent will be explicitly recited in the claim, and in the absence of such recitation no such intent is present. For example, as an aid to understanding, the following appended claims may contain usage of the introductory phrases “at least one” and “one or more” to introduce claim recitations. However, the use of such phrases should not be construed to imply that the introduction of a claim recitation by the indefinite articles “a” or “an” limits any particular claim containing such introduced claim recitation to inventions containing only one such recitation, even when the same claim includes the introductory phrases “one or more” or “at least one” and indefinite articles such as “a” or “an” (e.g., “a” and/or “an” should typically be interpreted to mean “at least one” or “one or more”); the same holds true for the use of definite articles used to introduce claim recitations. In addition, even if a specific number of an introduced claim recitation is explicitly recited, those skilled in the art will recognize that such recitation should typically be interpreted to mean at least the recited number (e.g., the bare recitation of “two recitations,” without other modifiers, typically means at least two recitations, or two or more recitations). Furthermore, in those instances where a convention analogous to “at least one of A, B, and C, etc.” is used, in general such a construction is intended in the sense one having skill in the art would understand the convention (e.g., “a system having at least one of A, B, and C” would include but not be limited to systems that have A alone, B alone, C alone, A and B together, A and C together, B and C together, and/or A, B, and C together, etc.). In those instances where a convention analogous to “at least one of A, B, or C, etc.” is used, in general such a construction is intended in the sense one having skill in the art would understand the convention (e.g., “a system having at least one of A, B, or C” would include but not be limited to systems that have A alone, B alone, C alone, A and B together, A and C together, B and C together, and/or A, B, and C together, etc.).

The herein described aspects depict different components contained within, or connected with, different other components. It is to be understood that such depicted architectures are merely exemplary, and that in fact many other architectures can be implemented which achieve the same functionality. In a conceptual sense, any arrangement of components to achieve the same functionality is effectively “associated” such that the desired functionality is achieved. Hence, any two components herein combined to achieve a particular functionality can be seen as “associated with” each other such that the desired functionality is achieved, irrespective of architectures or intermedial components. Likewise, any two components so associated can also be viewed as being “operably connected,” or “operably coupled,” to each other to achieve the desired functionality. Any two components capable of being so associated can also be viewed as being “operably coupleable” to each other to achieve the desired functionality. Specific examples of operably coupleable include but are not limited to physically mateable and/or physically interacting components and/or wirelessly interactable and/or wirelessly interacting components.

While particular aspects of the present subject matter described herein have been shown and described, it will be apparent to those skilled in the art that, based upon the teachings herein, changes and modifications may be made without departing from this subject matter described herein and its broader aspects and, therefore, the appended claims are to encompass within their scope all such changes and modifications as are within the true spirit and scope of this subject matter described herein. Furthermore, it is to be understood that the invention is defined by the appended claims.

What is claimed is:

1. A device comprising:
   a processor operable to execute an instruction set;
   an execution-optimization circuit for
   receiving an identification of a first instruction to be fetched from the instruction set for execution by the processor; and
   pointing to a second instruction of the instruction set of the processor to be fetched for execution by the processor if indicated by an execution-based optimization profile,
   the execution-based optimization profile being previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of at least a portion of the instruction set.

2. The device of claim 1, wherein the execution-optimization circuit includes at least one of a microengine, a micro-programmed circuit, and/or a hardwired circuit.

3. The device of claim 1, wherein the execution-optimization circuit includes an execution-optimization portion of a control unit of the processor.
4. The device of claim 1, wherein the processor and the execution-optimization circuit are formed on a chip.

5. The device of claim 1, wherein the execution-optimization circuit for receiving an identification of a first instruction to be fetched from the instruction set for execution by the processor includes an execution-optimization circuit for receiving an identification of a first instruction to be fetched from the instruction set of a static program for execution by the processor.

6. The device of claim 1, wherein the execution-optimization circuit for pointing to a second instruction of the instruction set of the processor to be fetched for execution by the processor if indicated by an execution-based optimization profile includes

an execution-optimization circuit for pointing to a second instruction of the instruction set of the processor to be fetched for execution by the processor if indicated by an execution-based optimization profile associated with the first instruction.

7. The device of claim 1, wherein the execution-optimization circuit for pointing to a second instruction of the instruction set of the processor to be fetched for execution by the processor if indicated by an execution-based optimization profile includes

an execution-optimization circuit for pointing to a second instruction of the instruction set of the processor to be fetched for execution by the processor if indicated by another instruction of the instruction set.

8. The device of claim 1, wherein the execution-based optimization profile being previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of at least a portion of the instruction set includes

the execution-based optimization profile being previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of at least a portion of an instruction set of a static program.

9. A method transparent to software executing on a processor, the method comprising:

in response to an identification of a first instruction to be fetched for execution by a processor, fetching a second instruction for execution by the processor if indicated by an execution-based optimization profile,

the execution-based optimization profile previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of at least a portion of the instruction set.

10. The method of claim 9, further comprising:

identifying the first instruction to be fetched for execution by the processor.

11. The method of claim 9, wherein the in response to an identification of a first instruction to be fetched for execution by the processor, fetching a second instruction for execution by the processor if indicated by an execution-based optimization profile includes

in response to an identification of a first instruction to be fetched from an instruction set of a static program for execution by the processor, fetching the second instruction for execution by the processor if indicated by an execution-based optimization profile.

12. The method of claim 9, wherein the fetching a second instruction for execution by the processor if indicated by an execution-based optimization profile includes

fetching the second instruction from an instruction set of a static program for execution by the processor if indicated by an execution-based optimization profile.

13. The method of claim 9, wherein the fetching a second instruction for execution by the processor if indicated by an execution-based optimization profile includes

fetching the second instruction from an instruction set of the processor for execution by the processor if indicated by an execution-based optimization profile.

14. The method of claim 9, wherein the fetching a second instruction for execution by the processor if indicated by an execution-based optimization profile includes

fetching the second instruction from an instruction set of the processor for execution by the processor if indicated by an execution-based optimization profile linked to the first instruction.

15. The device of claim 9, wherein the fetching a second instruction for execution by the processor if indicated by an execution-based optimization profile includes

fetching the second instruction for execution by the processor if indicated by an execution-based optimization profile linked to the first instruction.

16. The device of claim 9, wherein the fetching a second instruction for execution by the processor if indicated by an execution-based optimization profile includes

fetching the second instruction for execution by the processor if indicated by an execution-based optimization profile saved in a cache operably coupled with the processor.

17. The device of claim 9, wherein the execution-based optimization profile previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of at least a portion of the instruction set includes

the execution-based optimization profile being previously derived by a hardware device utilizing data invisible to software and generated during a runtime execution of the first instruction.

18. The method of claim 9, further comprising:

otherwise, fetching the first instruction from the instruction set of a static program for execution by the processor.

19. A device comprising:

means for selecting a first instruction to be fetched from an instruction set of a static program for execution by a processor;

means for routing the fetch of the first instruction to a second instruction of the instruction set of a static program if indicated by an execution-based optimization profile;

the execution-based optimization profile having been derived from data invisible to software and generated during a historical execution of the static program.