Title (en)

METHOD OF CONSTRUCTING A CONSTANT-FOLDING MECHANISM IN A MULTILANGUAGE OPTIMIZING COMPILER.

Title (de)

METHODE ZUR GENERIERUNG EINER KONSTANTEN FALTROUTINE IN EINEM MEHRSPRACHENCOMPILER.

Title (fr)

METHODE DE CONSTRUCTION D'UN MECANISME A REPLIEMENT CONSTANT DANS UN PROGRAMME COMPILATEUR OPTIMISEUR MULTILANGAGE.

Publication

EP 0532731 A1 19930324 (EN)

Application

EP 92908683 A 19920218

Priority

- US 66246191 A 19910227
- US 66246491 A 19910227
- US 66247791 A 19910227
- US 66248391 A 19910227
- US 66272591 A 19910227

Abstract (en)

[origin: WO9215945A1] A compiler framework uses a generic "shell" or control and sequencing mechanism, and a generic back end (where the code generator is target-specific). The generic back end includes the functions of optimization, register and memory allocation, and code generation. The shell may be executed on various host computers, and the code generation function of the back end may be targeted for any of a number of computer architectures. A front end is tailored for each different source language, such as Cobol, Fortran, Pascal, C, C++, Ada, etc. The front end scans and parses the source code modules, and generates from them an intermediate language ("IL") representation of the programs expressed in the source code. This IL is constructed to represent any of the source code languages in a universal manner, so the interface between the front end and back end is of a standard format, and need not be rewritten for each language-specific front end. The IL representation generated by the front end is based upon a tuple as the elemental unit, where each tuple represents a single operation to be performed, such as a load, a store, an add, a label, a branch, etc. A data structure is created by the front end for each tuple, with fields for various necessary information. One feature of the invention is a mechanism for representating effects and dependencies in the interface between front end and back end; a tuple has an effect if it writes to memory, and has a dependency if it reads from a location which some other node may write to. A mechanism independent of source language is provided for describing the effects of program execution. Another feature is the use in the optimization part of the compiler of a method for analyzing induction variables, where the improvement is to use the side effects sets used to construct IDEF sets. Another feature is a mechanism for "folding constants" (referred to as K-folding or a KFOLD routine), included as one of the optimizations. A further feature is the type definition mechanism, referred to as the TD module, which provides mechanisms used by the front end and the compiler of the back end in constructing program type information to be incorporated in an object module for use by a linker or debugger. Another feature is a method for doing code generation using code templates in a multipass manner.

Abstract (fr)

Programme compilateur "framework" utilisant un système expert générique ou un mécanisme de régulation et de classement, et un processeur dorsal générique (dans lequel le générateur de code a un récepteur spécifique). Le processeur dorsal générique regroupe les fonctions d'optimisation, d'allocation de registre et de mémoire et de génération de code. Le système expert générique peut être réalisé sur divers ordinateurs centraux et la fonction génération de code du processeur dorsal peut être ciblée en fonction de l'une quelconque de plusieurs architectures d'ordinateur. Un processeur frontal est adapté à chacun des différents langages source, tels que Cobolt, le Fortran, le Pascal, C, C++, Ada, etc. Ce processeur frontal balaie et analyse les modules codes sources puis élabore à partir de ceux-ci une représentation en langage intermédiaire ("IL") des programmes exprimés dans le code source. Ce langage IL est conçu de manière à représenter l'un quelconque des langages code source de manière universelle, de telle sorte que l'interface entre le processeur frontal et le processeur dorsal se présente sous une forme standard et n'a pas besoin d'être réécrit pour chacun des processeurs frontaux à langage spécifique. La représentation IL produite par le processeur frontal est basée sur un nuplet en tant qu'élément de base, dans lequel chaque nuplet représente une opération unique à effectuer -charge, mise en mémoire, addition, étiquettage, branchement, etc. Une structure de données est élaborée par le processeur frontal pour chaque nuplet, avec des champs pour les diverses informations nécessaires. Une des caractéristiques de l'invention réside dans un mécanisme de représntation des effets et des dépendances dans l'interface entre le processeur frontal et le processeur dorsal; un nuplet produit un effet s'il écrit dans la mémoire et présente une dépendance s'il lit à partir d'un emplacement vers lequel un autre noeud peut écrire. Un mécanisme indépendant du langage source est prévu pour décrire les effets de l'exécution

IPC 1-7

G06F 9/45

IPC 8 full level

G06F 9/45 (2006.01)

IPC 8 main group level

G06F (2006.01)

CPC (source: EP)

G06F 8/433 (2013.01); **G06F 8/437** (2013.01); **G06F 8/443** (2013.01); **G06F 8/447** (2013.01)

Citation (search report)

See references of WO 9215941A1

Designated contracting state (EPC)

AT BE CH DE DK ES FR GB GR IT LI NL

DOCDB simple family (publication)

WO 9215945 A1 19920917; AU 1420492 A 19921006; AU 1429292 A 19921006; AU 1439792 A 19921006; AU 1442292 A 19921006; AU 1569892 A 19921006; AU 653799 B2 19941013; AU 658399 B2 19950413; AU 663310 B2 19951005; AU 663311 B2 19951005; AU 663493 B2 19951012; CA 2081449 A1 19920828; CA 2081449 C 19990413; CA 2081473 A1 19920828; CA 2081473 C 19990406; CA 2081475 A1 19920828; CA 2081475 C 19980505; CA 2081476 A1 19920828; CA 2081477 A1 19920828; CA 2081477 C 19920828; DE 69225281 D1 19980604; DE 69225281 T2 19990107; EP 0526621 A1 19930210; EP 0526622 A1 19930210; EP 0528008 A1 19930224; EP 0529049 A1 19930303; EP 0529049 B1 19980429; EP 0532731 A1 19930324; FI 924845 A0 19921026; FI 924845 A 19921026;

FI 924846 A0 19921026; FI 924846 A 19921026; JP H06501579 A 19940217; JP H06501580 A 19940217; JP H06501581 A 19940217; JP H06501582 A 19940217; JP H06501583 A 19940217; JP H0762825 B2 19950705; JP H0762826 B2 19950705; JP H0769832 B2 19950731; JP H0769833 B2 19950731; JP H0769834 B2 19950731; KR 950006607 B1 19950619; KR 950006608 B1 19950619; KR 950006609 B1 19950619; KR 960003050 B1 19960304; KR 960003138 B1 19960305; NO 924114 D0 19921023; NO 924114 L 19921218; NO 924115 D0 19921023; NO 924115 L 19921218; WO 9215941 A1 19920917; WO 9215942 A1 19920917; WO 9215943 A1 19920917; WO 9215944 A1 19920917

DOCDB simple family (application)

US 9201309 W 19920218; AU 1420492 A 19920218; AU 1429292 A 19920218; AU 1439792 A 19920218; AU 1442292 A 19920218; AU 1569892 A 19920218; CA 2081449 A 19920218; CA 2081473 A 19920218; CA 2081475 A 19920218; CA 2081476 A 19920218; CA 2081477 A 19920218; DE 69225281 T 19920218; EP 92906822 A 19920218; EP 92907024 A 19920218; EP 92907163 A 19920218; EP 92907267 A 19920218; EP 92908683 A 19920218; FI 924845 A 19921026; FI 924846 A 19921026; JP 50668792 A 19920218; JP 50669092 A 19920218; JP 50677392 A 19920218; JP 50706792 A 19920218; JP 50781492 A 19920218; KR 920702690 A 19921027; KR 920702691 A 19921027; KR 920702692 A 19921027; KR 920702693 A 19921027; KR 920702694 A 19921027; NO 924114 A 19921023; NO 924115 A 19921023; US 9201252 W 19920218; US 9201278 W 19920218; US 9201284 W 19920218; US 9201290 W 19920218