(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년01월09일
(11) 등록번호 10-1349628
(24) 등록일자 2014년01월03일

(51) 국제특허분류(Int. Cl.)
G06F 9/45 (2006.01)
(21) 출원번호 10-2011-0130292
(22) 출원일자 2011년12월07일
심사청구일자 2011년12월07일
(65) 공개번호 10-2013-0063758
(43) 공개일자 2013년06월17일
(56) 선행기술조사문헌
JP2002024032 A
KR1020097011902 A
WO03065179 A2
US7219338 B2
전체 청구항 수 : 총 15 항

(73) 특허권자
한국과학기술연구원
서울특별시 성북구 화랑로14길 5 (하월곡동)
(72) 발명자
김수현
서울특별시 송파구 올림픽로 99, 114동 2102호 (잠실동, 잠실엘스)
강진구
전라북도 군산시 약전길 13 (중앙로2가)
(74) 대리인
김 순 영, 김영철

심사관 : 지정훈

(54) 발명의 명칭 **연산자를 이용한 중간 언어 변환 방법과 그를 위한 시스템 및 컴퓨터로 읽을 수 있는 기록매체**

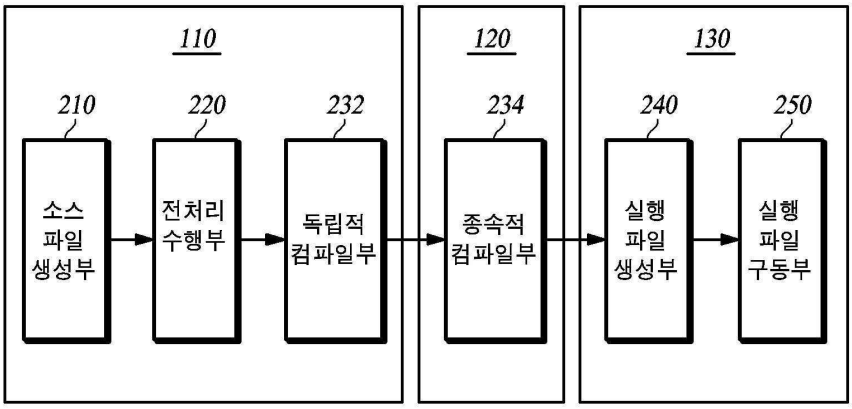
(57) 요 약

연산자를 이용한 중간 언어 변환 방법과 그를 위한 시스템 및 컴퓨터로 읽을 수 있는 기록매체를 개시한다.

C 언어(C-Language) 기반의 소스(Source) 파일을 생성하는 소스 파일 생성 과정; 상기 소스 파일을 중간 언어 코드로 변환하고, 상기 중간 언어 코드를 플랫폼(Platform)에 따라 컴파일(Compile)할 때 특정 지시자가 특정 연산자를 경유하도록 하는 컴파일 과정; 상기 컴파일된 상기 중간 언어 코드에 대한 실행 파일을 생성하는 실행 파일 생성 과정; 및 상기 실행 파일이 상기 플랫폼에서 구동되도록 하는 실행 파일 구동 과정을 포함하는 것을 특징으로 하는 연산자를 이용한 중간 언어 변환 방법을 제공한다.

본 실시예에 의하면, 다양한 정보기기의 사용으로 인해 다양한 플랫폼이 이용됨에 따라 특정 플랫폼에 의존적이지 않은 독립적인 중간 언어 코드로 변환한 후 각각의 플랫폼에 해당하는 종속적인 중간 언어 코드로 변환되도록 하는 데 이때, C 언어 기반의 특정 지시자가 독립적인 중간 언어 코드 상에 특정 연산자로 표현되도록 하며, 특정 연산자가 종속적인 중간 언어 코드 상에 상수로 표현되도록 하는 효과가 있다.

대 표 도 - 도4



특허청구의 범위

청구항 1

C 언어(C-Language) 기반의 소스(Source) 파일을 생성하는 소스 파일 생성 과정;

상기 소스 파일을 중간 언어 코드로 변환하고 상기 중간 언어 코드를 플랫폼(Platform)에 따라 컴파일(Compile)하는 과정으로서, 상기 소스 파일의 특정 지시자를 특정 연산자로 변환하는 과정 및 상기 특정 연산자를 상기 플랫폼에 따른 값을 갖는 상수로 변환하는 과정을 포함하는, 컴파일 과정;

상기 컴파일된 상기 중간 언어 코드에 대한 실행 파일을 생성하는 실행 파일 생성 과정; 및

상기 실행 파일이 상기 플랫폼에서 구동되도록 하는 실행 파일 구동 과정을 포함하는 것을 특징으로 하는 연산자를 이용한 중간 언어 변환 방법.

청구항 2

제 1 항에 있어서,

상기 컴파일 과정은,

상기 중간 언어 코드를 상기 플랫폼의 독립적인 중간 언어 코드로 컴파일하는 독립적 컴파일 과정; 및

상기 독립적인 중간 언어 코드를 상기 플랫폼의 종속적인 중간 언어 코드로 컴파일하는 종속적 컴파일 과정을 포함하는 것을 특징으로 하는 연산자를 이용한 중간 언어 변환 방법.

청구항 3

제 2 항에 있어서,

상기 독립적 컴파일 과정은,

상기 특정 지시자를 갖는 상기 소스 파일을 상기 중간 언어 코드로 변환한 후 상기 중간 언어 코드를 상기 플랫폼의 독립적인 중간 언어 코드로 컴파일할 때 상기 특정 지시자를 상기 특정 연산자로 변환하는 것을 특징으로 하는 연산자를 이용한 중간 언어 변환 방법.

청구항 4

제 2 항에 있어서,

상기 종속적 컴파일 과정은,

상기 독립적인 중간 언어 코드를 상기 플랫폼의 종속적인 중간 언어 코드로 컴파일할 때 상기 독립적인 중간 언어 코드에 포함된 상기 특정 연산자를 상기 플랫폼에 따른 값을 갖는 상수로 변환하는 것을 특징으로 하는 연산자를 이용한 중간 언어 변환 방법.

청구항 5

제 4 항에 있어서,

상기 종속적인 컴파일 과정은,

상기 독립적인 중간 언어 코드를 상기 종속적인 중간 언어 코드로 컴파일할 때 문법 오류 체크를 수행하며, 확인 결과, 상기 문법 오류 체크에 오류가 없는 경우 상기 실행 파일 생성 과정에서 상기 실행 파일을 생성하는

것을 특징으로 하는 연산자를 이용한 중간 언어 변환 방법.

청구항 6

제 1 항에 있어서,

상기 특정 지시자는 'sizeof ()'인 것을 특징으로 하는 연산자를 이용한 중간 언어 변환 방법.

청구항 7

제 1 항에 있어서,

상기 소스 파일에 대한 전처리(Preprocessing)를 수행하는 전처리 수행 과정

을 추가로 포함하는 것을 특징으로 하는 연산자를 이용한 중간 언어 변환 방법.

청구항 8

C 언어 기반의 소스 파일을 생성하는 소스 파일 생성부;

상기 소스 파일을 중간 언어 코드로 변환하고 상기 중간 언어 코드를 플랫폼에 따라 컴파일하되, 상기 중간 언어 코드를 컴파일할 때 상기 소스 파일의 특정 지시자를 특정 연산자로 변환하고 상기 특정 연산자를 상기 플랫폼에 따른 값을 갖는 상수로 변환하는 컴파일부;

상기 컴파일된 상기 중간 언어 코드에 대한 실행 파일을 생성하는 실행 파일 생성부; 및

상기 실행 파일이 상기 플랫폼에서 구동되도록 하는 실행 파일 구동부를 포함하는 것을 특징으로 하는 중간 언어 변환 시스템.

청구항 9

제 8 항에 있어서,

상기 컴파일부는,

상기 중간 언어 코드를 상기 플랫폼의 독립적인 중간 언어 코드로 컴파일하는 독립적 컴파일부; 및

상기 독립적인 중간 언어 코드를 상기 플랫폼의 종속적인 중간 언어 코드로 컴파일하는 종속적 컴파일부를 포함하는 것을 특징으로 하는 중간 언어 변환 시스템.

청구항 10

제 9 항에 있어서,

상기 독립적 컴파일부는,

상기 특정 지시자를 갖는 상기 소스 파일을 상기 중간 언어 코드로 변환한 후 상기 중간 언어 코드를 상기 플랫폼의 독립적인 중간 언어 코드로 컴파일할 때 상기 특정 지시자를 상기 특정 연산자로 변환하는 것을 특징으로 하는 중간 언어 변환 시스템.

청구항 11

제 9 항에 있어서,

상기 종속적 컴파일부는,

상기 독립적인 중간 언어 코드를 상기 플랫폼의 종속적인 중간 언어 코드로 컴파일할 때 상기 독립적인 중간 언어 코드에 포함된 상기 특정 연산자를 상기 플랫폼에 따른 값을 갖는 상수로 변환하는 것을 특징으로 하는 중간 언어 변환 시스템.

청구항 12

제 11 항에 있어서,

상기 종속적인 컴파일부는,

상기 독립적인 중간 언어 코드를 상기 종속적인 중간 언어 코드로 컴파일할 때 문법 오류 체크를 수행하며, 확인 결과, 상기 문법 오류 체크에 오류가 없는 경우 상기 실행 파일 생성부에서 상기 실행 파일을 생성하는 것을 특징으로 하는 중간 언어 변환 시스템.

청구항 13

제 8 항에 있어서,

상기 특정 지시자는 'sizeof ()'인 것을 특징으로 하는 중간 언어 변환 시스템.

청구항 14

제 8 항에 있어서,

상기 소스 파일에 대한 전처리를 수행하는 전처리 수행부를 추가로 포함하는 것을 특징으로 하는 중간 언어 변환 시스템.

청구항 15

데이터 처리 기기에,

C 언어 기반의 소스 파일을 생성하는 소스 파일 생성 과정;

상기 소스 파일을 중간 언어 코드로 변환하고 상기 중간 언어 코드를 플랫폼에 따라 컴파일하는 과정으로서, 상기 소스 파일의 특정 지시자를 특정 연산자로 변환하는 과정 및 상기 특정 연산자를 상기 플랫폼에 따른 값을 갖는 상수로 변환하는 과정을 포함하는, 컴파일 과정;

상기 컴파일된 상기 중간 언어 코드에 대한 실행 파일을 생성하는 실행 파일 생성 과정; 및

상기 실행 파일이 상기 플랫폼에서 구동되도록 하는 실행 파일 구동 과정을 실현시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

명 세 서

기 술 분 야

[0001] 본 실시예는 연산자를 이용한 중간 언어 변환 방법과 그를 위한 시스템 및 컴퓨터로 읽을 수 있는 기록매체에 관한 것이다. 더욱 상세하게는, 다양한 정보기기의 사용으로 인해 다양한 플랫폼이 이용됨에 따라 특정 플랫폼에 의존적이지 않은 독립적인 중간 언어 코드로 변환한 후 각각의 플랫폼에 해당하는 종속적인 중간 언어 코드로 변환되도록 하는 데 이때, C 언어 기반의 특정 지시자가 독립적인 중간 언어 코드 상에 특정 연산자로 표현되도록 하며, 특정 연산자가 종속적인 중간 언어 코드 상에 상수로 표현되도록 하는 연산자를 이용한 중간 언어 변환 방법과 그를 위한 시스템 및 컴퓨터로 읽을 수 있는 기록매체에 관한 것이다.

배 경 기 술

[0002] 이 부분에 기술된 내용은 단순히 본 실시예에 대한 배경 정보를 제공할 뿐 종래기술을 구성하는 것은 아니다.

[0003] 프로그램은 프로그래밍 언어로 기입된다. 프로그래밍 언어는 컴퓨터에 의해 처리 및 실행될 수 있는 명령의 시퀀스를 정의하는데 사용될 수 있는 임의의 인공 언어이다. 그러나, 프로그래밍 언어는 프로그래밍 언어를 사용하여 표현된 소스 코드로부터 컴퓨터 또는 데이터-처리 시스템이 작업하는데 필요한 기계 코드로의, 컴파일러와 같은 또 다른 프로그램에 의한 번역(Translate) 절차를 필요로 하는 것으로 의미한다.

[0004] 이러한, 프로그래밍 언어는 전형적으로 3가지의 서로 다른 개념의 부류: 저-레벨 언어, 고-레벨 언어, 또는 중간-레벨 언어 중 하나에 속하는 것으로서 간주된다. 일반적인 저-레벨 프로그래밍 언어의 예로는 어셈블리 언어가 있으며, 고-레벨 프로그래밍 언어는 코볼(COBOL), 포트란(FORTRAN), 파스칼(Pascal)이 있으며, 중간 레벨 프로그래밍 언어로는 C 언어 및 C++ 언어가 있다.

[0005] C 언어는 기술된 중간 레벨 특징을 갖는, 고레벨 언어보다 더 기계-독립형인 어셈블리 언어인 것으로 많은 사람이 생각하는 프로그래밍 언어이다. C++ 언어는 기술된 중간 레벨 특징을 또한 포함하는 C 프로그래밍 언어의 객체-지향 버전이다. 여기서, 'C 언어'는 C 언어 및 그의 C++ 언어의 구현 모두를 지칭한다. C 언어는 중간 레벨 프로그래밍 언어와 관련된 가능한 논리적 실수, 특유의 위험, 및 비효율성의 구체적인 예를 보여주는데 이용될 수 있다.

[0006] 이러한, 종래의 C 언어 또는 C++ 언어 등의 프로그래밍 언어(Programming Language)로 작성된 애플리케이션(Application)들은, 애플리케이션이 사용될 플랫폼(Platform)에 맞게 중간 언어나 실행파일의 형태로 변환될 수 있는데, 최근들어 다양한 정보기기의 사용에 따른 다양한 플랫폼이 이용됨에 따라 각각의 플랫폼에 따라 특정 플랫폼에 의존적이지 않은 독립적인 중간 언어나 실행파일의 형태로 변환하는 기술이 필요한 실정이다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 공개특허공보 제10-2009-0089382호

발명의 내용

해결하려는 과제

[0008] 전술한 문제점을 해결하기 위해 본 실시예는, 다양한 정보기기의 사용으로 인해 다양한 플랫폼이 이용됨에 따라 특정 플랫폼에 의존적이지 않은 독립적인 중간 언어 코드로 변환한 후 각각의 플랫폼에 해당하는 종속적인 중간 언어 코드로 변환되도록 하는 연산자를 이용한 중간 언어 변환 방법과 그를 위한 시스템 및 컴퓨터로 읽을 수 있는 기록매체를 제공하는 데 주된 목적이 있다.

과제의 해결 수단

[0009] 전술한 목적을 달성하기 위해 본 실시예의 일 측면에 의하면, C 언어(C-Language) 기반의 소스(Source) 파일을 생성하는 소스 파일 생성 과정; 상기 소스 파일을 중간 언어 코드로 변환하고, 상기 중간 언어 코드를 플랫폼(Platform)에 따라 컴파일(Compile)할 때 특정 지시자가 특정 연산자를 경유하도록 하는 컴파일 과정; 상기 컴파일된 상기 중간 언어 코드에 대한 실행 파일을 생성하는 실행 파일 생성 과정; 및 상기 실행 파일이 상기 플랫폼에서 구동되도록 하는 실행 파일 구동 과정을 포함하는 것을 특징으로 하는 연산자를 이용한 중간 언어 변환 방법을 제공한다.

[0010] 또한, 본 실시예의 다른 측면에 의하면, C 언어 기반의 소스 파일을 생성하는 소스 파일 생성부; 상기 소스 파일을 중간 언어 코드로 변환하고, 상기 중간 언어 코드를 플랫폼에 따라 컴파일할 때 특정 지시자가 특정 연산자를 경유하도록 하는 컴파일부; 상기 컴파일된 상기 중간 언어 코드에 대한 실행 파일을 생성하는 실행 파일 생성부; 및 상기 실행 파일이 상기 플랫폼에서 구동되도록 하는 실행 파일 구동부를 포함하는 것을 특징으로 하는 중간 언어 변환 장치를 제공한다.

[0011] 또한, 본 실시예의 다른 측면에 의하면, 데이터 처리 기기에, C 언어 기반의 소스 파일을 생성하는 소스 파일

생성 과정; 상기 소스 파일을 중간 언어 코드로 변환하고, 상기 중간 언어 코드를 플랫폼에 따라 컴파일할 때 특정 지시자가 특정 연산자를 경유하도록 하는 컴파일 과정; 상기 컴파일된 상기 중간 언어 코드에 대한 실행 파일을 생성하는 실행 파일 생성 과정; 및 상기 실행 파일이 상기 플랫폼에서 구동되도록 하는 실행 파일 구동 과정을 실현시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체를 제공한다.

발명의 효과

[0012] 이상에서 설명한 바와 같이 본 실시예에 의하면, 다양한 정보기기의 사용에 따라 다양한 플랫폼이 이용됨에 따라 특정 플랫폼에 의존적이지 않은 독립적인 중간 언어 코드로 변환한 후 각각의 플랫폼에 해당하는 종속적인 중간 언어 코드로 변환되도록 하는 데 이때, C 언어 기반의 특정 지시자가 독립적인 중간 언어 코드 상에 특정 연산자로 표현되도록 하며, 특정 연산자가 종속적인 중간 언어 코드 상에 상수로 표현되도록 하는 효과가 있다.

[0013] 또한, 본 실시예에 의하면, 플랫폼에 따라 값을 가지는 C 언어 기반의 특정 지시자(sizeof())를 특정 연산자로 독립적인 중간 언어 코드로 표현 가능하게 함으로써, 독립적인 중간 코드의 이식성(Portability)을 향상시킬 수 있는 효과가 있다.

도면의 간단한 설명

[0014] 도 1은 본 실시예에 따른 중간 언어 변환 시스템을 개략적으로 나타낸 블럭 구성도,

도 2는 제 1 실시예에 따른 중간 언어 변환 시스템을 개략적으로 나타낸 블럭 구성도,

도 3은 제 2 실시예에 따른 중간 언어 변환 시스템을 개략적으로 나타낸 블럭 구성도,

도 4는 제 3 실시예에 따른 중간 언어 변환 시스템을 개략적으로 나타낸 블럭 구성도,

도 5는 제 4 실시예에 따른 중간 언어 변환 시스템을 개략적으로 나타낸 블럭 구성도,

도 6은 본 실시예에 따른 중간 언어 변환 방법을 설명하기 위한 순서도,

도 7은 본 실시예에 따른 중간 언어 변환 과정을 설명하기 위한 예시도이다.

발명을 실시하기 위한 구체적인 내용

[0015] 이하, 본 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0016] 도 1은 본 실시예에 따른 중간 언어 변환 시스템을 개략적으로 나타낸 블럭 구성도이다.

[0017] 본 실시예에 따른 중간 언어 변환 시스템은 개발자 단말기(110), 중간 언어 변환 장치(120) 및 플랫폼(130)을 포함한다. 본 실시예에서는 중간 언어 변환 시스템이 개발자 단말기(110), 중간 언어 변환 장치(120) 및 플랫폼(130)만을 포함하는 것으로 기재하고 있으나, 이는 본 발명의 일 실시예의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 본 발명의 일 실시예가 속하는 기술 분야에서 통상의 지식을 가진 자라면 본 발명의 일 실시예의 본질적인 특성에서 벗어나지 않는 범위에서 중간 언어 변환 시스템에 포함되는 구성 요소에 대하여 다양하게 수정 및 변형하여 적용 가능할 것이다.

[0018] 한편, 도 1에서의 전반적인 흐름을 설명하자면, 개발자 단말기(110)는 탑재된 개발자 툴을 이용하여 C 언어 기반의 소스 파일을 생성하며, 중간 언어 변환 장치(120)는 탑재된 컴파일러를 이용하여 소스 파일을 컴파일(Compile)하여 중간 언어 코드로 변환한 후 링커(Linker) 프로그램으로 중간 언어 코드를 연결시켜 실행 파일을 만든 후 만들어진 실행 파일이 플랫폼(130)에서 실행된다.

[0019] 한편, 도 1에서의 제 2 실시예에 따른 전반적인 흐름을 설명하자면, 개발자 단말기(110)는 개발자의 조작 또는 명령에 의해 작성된 C 언어 소스 코드를 읽어 들이고 전처리 과정을 수행한 후 플랫폼의 독립적인 중간 언어 코드를 생성한다. 이에, 중간 언어 변환 장치(120)는 개발자 단말기(110)로부터 생성된 플랫폼 독립적인 중간 언어 코드를 읽어 들이고 이를 특정 플랫폼에 맞게 플랫폼 종속적인 중간 언어 코드로 변환한다. 또한, 중간 언어 변환 장치(120)는 플랫폼의 종속적인 중간 언어 코드를 플랫폼(130)에 맞는 실행 파일로 변환한다. 이에, 각 플랫폼(130)은 중간 언어 변환 장치(120)로부터 실행파일을 전송받아 구동한다.

[0020] 한편, 도 1에서의 제 3 실시예에 따른 전반적인 흐름을 설명하자면, 개발자 단말기(110)는 개발자의 조작 또는 명령에 의해 작성된 C 언어 소스 코드를 읽어 들이고 전처리 과정을 수행한 후 플랫폼의 독립적인 중간 언어 코드를 생성한다. 이에, 중간 언어 변환 장치(120)는 개발자 단말기(110)로부터 생성된 플랫폼 독립적인 중간 언

어 코드를 읽어 들이고 이를 특정 플랫폼에 맞게 플랫폼 독립적인 중간 언어 코드로 변환한다. 이에, 각 플랫폼(130)은 중간 언어 변환 장치(120)로부터 플랫폼의 종속적인 중간 언어 코드를 전송받고 이를 플랫폼(130)에 맞게 실행파일로 변환 후 구동한다.

[0021] 한편, 도 1에서의 제 4 실시예에 따른 전반적인 흐름을 설명하자면, 개발자 단말기(110)는 개발자의 조작 또는 명령에 의해 작성된 C 언어 소스 코드를 읽어 들이고 전처리 과정을 수행한 후 플랫폼의 독립적인 중간 언어 코드를 생성한다. 또한, 개발자 단말기(110)는 이를 특정 플랫폼에 맞게 플랫폼의 독립적인 중간 언어 코드로 변환한 후 플랫폼에 맞게 실행파일로 변환한다. 이에, 각 플랫폼(130)은 개발자 단말기(110)로부터 실행파일을 전송받아 구동한다.

[0022] 이하에서 개발자 단말기(110), 중간 언어 변환 장치(120) 및 플랫폼(130)에 대해 각각 설명하도록 한다.

[0023] 개발자 단말기(110)는 태블릿 PC(Tablet PC), 랩톱(Laptop), 개인용 컴퓨터(PC: Personal Computer), 스마트폰(Smart Phone), 개인휴대용 정보단말기(PDA: Personal Digital Assistant) 및 이동통신 단말기(Mobile Communication Terminal) 등 중 어느 하나일 수 있으며, 사용자의 조작 또는 명령에 의해 C 언어 기반의 소스 파일을 생성할 수 있으며, 사용자의 키 조작에 따라 통신망을 경유하여 각종 데이터를 송수신할 수 있는 단말기를 말한다.

[0024] 즉, 개발자 단말기(110)는 C 언어 기반의 소스 파일을 생성할 수 있는 프로그램(예컨대, 개발자 툴) 또는 프로토콜을 저장하기 위한 메모리, 해당 프로그램을 실행하여 연산 및 제어하기 위한 마이크로프로세서 등을 구비하고 있는 단말기를 의미한다. 또한, 개발자 단말기(110) 중간 언어 변환 장치(120)와 통신할 수 있는 단말기이며, 중간 언어 변환 장치(120)와 서버-클라이언트 통신이 가능하다면 그 어떠한 단말기도 가능하다. 한편, 개발자 단말기(110)는 통신망을 통하여 C 언어 기반의 소스 파일을 생성할 수 있는 클라우드 컴퓨팅(Cloud Computing)을 지원하는 클라우드 컴퓨팅 단말기가 될 수 있다.

[0025] 본 실시예에 기재된 'C 언어'란 프로그램을 만드는 프로그램으로서, 타 기종간의 이식성이 높고 고정적인 형식이 없으며, 퀵 베이직처럼 퀵C, 터보C 등의 처리계가 있다. 또한, 오브젝트 지향적인 요소가 첨가된 C++이 개발되었으며, 시스템의 저급 부분을 제어할 수 있어 사용하기 쉽고 능률적인 프로그램을 작성할 수 있는 프로그래밍 언어이다.

[0026] 개발자 단말기(110)는 사용자의 조작 또는 명령에 의해 개발자 툴을 구동하며, 개발자 툴을 통해 C 언어 기반의 소스 파일을 생성할 수 있다. 한편, 도 1에서는 개발자 단말기(110), 중간 언어 변환 장치(120) 및 플랫폼(130)이 각각 별도의 장치로 구현된 것으로 기재하고 있으나, 실제 실시예의 구현에 있어서, 개발자 단말기(110), 중간 언어 변환 장치(120) 및 플랫폼(130)을 모두 포함하는 형태의 자립형(Stand Alone) 장치로 구현될 수 있을 것이다.

[0027] 이러한, 개발자 단말기(110)는 애플리케이션 개발자 측의 플랫폼으로서, 개발자 단말기(110)에서 개발자 툴을 이용하여 소스 코드(Source Code)의 형태의 소스 파일을 생성할 수 있다. 여기서, 'C 언어 기반의 프로그래밍 언어'란 C 언어 자체, C 언어를 확장한 객체 지향형 언어인 C++언어, C 프로그래밍 언어에 스몰토크(Smalltalk) 스타일의 메시지 구문을 추가한 객체 지향형 언어인 오브젝티브-C(Objective-C) 언어, 또는 C/C++언어에 기반하여 이를 수정 또는 변형하여 구성되는 임의의 프로그래밍 언어를 지칭한다.

[0028] 또한, 본 실시예에 기재된 '중간 언어 코드'란 플랫폼(130)의 CPU 종류(예컨대, x86, x64, ARM 등) 또는 운영체제의 종류(예컨대, 윈도우(Windows), 리눅스(Linux), OSX, iOS, 안드로이드(Android), 윈도우 모바일, 윈도우 폰 등)와 같은 플랫폼의 특성에 의존하지 않으며, 플랫폼에 대해 독립적인 형태의 코드를 의미한다. 즉, 중간 언어 코드는 중간 언어 변환 장치(120)에 의해 추후 컴파일 시에 특정 플랫폼에 대응되는 바이너리(Binary) 코드로 변환하여 실행하는 것이 가능하다. 예를 들어, 중간 언어 코드는 소정 규격의 가상 머신(Virtual Machine)에 의하여 실행될 수 있다.

[0029] 제 1 실시예에 따른 개발자 단말기(110)는 개발자의 조작 또는 명령에 의해 작성된 C 언어 소스 코드를 중간 언어 변환 장치(120)로 전송한다. 한편, 제 2 실시예에 따른 개발자 단말기(110)는 개발자의 조작 또는 명령에 의해 작성된 C 언어 소스 코드를 읽어 들이고 전처리 과정을 수행한 후 플랫폼의 독립적인 중간 언어 코드를 생성한다.

[0030] 제 3 실시예에 따른 개발자 단말기(110)는 개발자의 조작 또는 명령에 의해 작성된 C 언어 소스 코드를 읽어 들이고 전처리 과정을 수행한 후 플랫폼의 독립적인 중간 언어 코드를 생성한다. 한편, 제 4 실시예에 따른 개발자 단말기(110)는 개발자의 조작 또는 명령에 의해 작성된 C 언어 소스 코드를 읽어 들이고 전처리 과정을 수행

한 후 플랫폼의 독립적인 중간 언어 코드를 생성한다. 또한, 제 4 실시예에 따른 개발자 단말기(110)는 이를 특정 플랫폼에 맞게 플랫폼의 독립적인 중간 언어 코드로 변환한 후 플랫폼에 맞게 실행파일로 변환한다.

[0031] 한편, 제 1 실시예에 따른 중간 언어 변환 장치(120)는 C 언어 기반의 프로그래밍 언어와 플랫폼의 독립적인 중간 언어 사이의 변환 규칙이 정의되어 있는 매핑 정보를 저장할 수 있다. 예를 들어, 중간 언어 변환 장치(120)는 C 언어의 각 지시자를 이에 대응되는 중간 언어 코드와 매칭시킨 테이블(Table)의 형태의 매핑 정보를 저장할 수 있다. 즉, 중간 언어 변환 장치(120)는 이러한 매핑 정보를 이용하여, C 언어 기반의 소스 파일을 중간 언어 코드로 변환할 수 있다.

[0032] 예를 들어, C 언어에서 사용하는 "int" 라는 타입(Type) 지시자는 16비트(Bit) 머신 플랫폼에서는 16비트 크기를 의미하는 반면 32 비트 머신 플랫폼에서는 32 비트 크기를 의미한다. 이러한 이유 때문에 C 언어를 이용하여 애플리케이션을 작성하는 경우 애플리케이션이 사용될 플랫폼의 사양에 맞추어 "int" 지시자를 사용해야 한다. 본 실시예들에서는 C 언어의 "int" 지시자가 이에 대응되는 중간 언어 코드로 변환되어 배포되며, 추후 중간 언어 코드의 컴파일시에 플랫폼이 16비트 머신인지 또는 32비트 머신인지에 따라 중간 언어 코드를 적절한 비트 크기를 의미하는 바이너리 코드로 컴파일할 수 있다.

[0033] 한편, 제 1 실시예에 따른 중간 언어 변환 장치(120)는 개발자 단말기(110)로부터 소스 파일을 수신(예컨대, 통신망 등을 이용하여)할 수 있으며, 플랫폼(130)과도 통신(예컨대, 통신망 등을 이용하여)할 수 있다. 이때, 개발자 단말기(110), 중간 언어 변환 장치(120) 및 플랫폼(130) 간에 통신 방식은 특정 방식으로 한정되지 않으며, 플랫폼(130)은 동일한 처리장치 또는 동일한 운영체제를 사용할 수도 있으나, 서로 상이한 처리장치 또는 운영체제를 사용할 수도 있다. 예를 들어, 중간 언어 변환 장치(120)가 자동으로 또는 플랫폼(130)의 다운로드(Download) 요청에 의해 하나의 플랫폼(130)에 애플리케이션을 전송하고자 하는 경우, 중간 언어 변환 장치(120)는 중간 언어 코드로 된 애플리케이션을 해당 플랫폼(130)에 대응되는 형태의 바이너리 코드로 변환할 수 있다. 이때, 변환된 바이너리 코드는 플랫폼(130)에서 실행 가능한 실행파일의 형태일 수 있다. 예컨대 플랫폼(130)이 윈도우(Windows) 운영체제를 이용하는 경우 바이너리 코드는 "exe" 확장자를 갖는 실행 파일일 수 있다.

[0034] 또한, 제 1 실시예에 따른 중간 언어 변환 장치(120)는 중간 언어 코드를 다양한 형태의 실행 파일로 생성하기 위한 중간 언어 해석부를 포함할 수 있다. 이러한, 중간 언어 해석부는 플랫폼(130)의 독립적인 중간 언어 코드를 컴파일하여 각 플랫폼에 대해 최적화된 형태의 바이너리 코드로 변환할 수 있다. 이를 위하여, 중간 언어 해석부는 중간 언어 코드를 공지된 플랫폼의 유형별 바이너리 코드로 변환하기 위한 수단을 포함할 수 있다. 예컨대 가상머신 컴파일러(VM Compiler) 등이 이에 해당될 수 있다.

[0035] 한편, 제 1 실시예에 따른 중간 언어 변환 장치(120)는 중간 언어 변환 장치(120)에 접속한 플랫폼(130)의 프로파일(Profile)(예컨대, CPU 또는 운영 체제의 종류 등)이 전송된다. 따라서, 중간 언어 변환 장치(120)는 실행파일을 전송할 해당 플랫폼(130)의 프로파일을 이용하여 코드 사이즈 최적화 및 변수 처리 등을 수행함으로써, 중간 언어 코드를 플랫폼(130)에 최적화된 바이너리 코드(예컨대, 실행파일)로 변환할 수 있다.

[0036] 제 1 실시예에 따른 중간 언어 변환 장치(120)는 C 언어 기반의 소스 파일을 생성하거나, 개발자 단말기(110)로부터 C 언어 기반의 소스 파일을 수신한다. 소스 파일에 대한 전처리를 수행하며, 전처리가 수행된 소스 파일을 중간 언어 코드로 변환하고, 중간 언어 코드를 플랫폼(130)에 따라 컴파일할 때 특정 지시자가 특정 연산자를 경유하도록 하며, 컴파일된 중간 언어 코드에 대한 실행 파일을 생성하며, 실행 파일이 플랫폼(130)에서 구동되도록 한다.

[0037] 제 1 실시예에 따른 중간 언어 변환 장치(120)에서의 전처리 과정에 대해 설명하자면, 중간 언어 변환 장치(120)는 소스 파일에 대한 전처리를 수행할 수 있는데, '전처리'란 컴파일 이전에 특정한 부분을 치환하는 작업을 의미한다. 예컨대, '#'으로 정의된 라인을 전처리 구문이라 하며, 전처리 구문 끝에는 명령의 끝을 의미하는 세미콜론(;)을 붙이지 않는다.

[0038] 한편, 라이브러리는 소프트웨어를 만들 때 쓰이는 클래스나 서브루틴들의 모임으로서, 누구나 가져다 쓸 수 있도록 만들어져 있는 기본적으로 제공되는 함수를 포함한다. 이러한, 라이브러리는 '정적 라이브러리'와 '동적 라이브러리'를 포함하는데, 정적 라이브러리는 컴파일러가 소스 파일을 컴파일할 때 참조되는 프로그램 모듈이다. 즉, 정적 라이브러리(Statically-Linked Library)는 루틴(Routine) 외부 함수와 변수들의 집합으로, 컴파일러, 링커, 바인더 등에 의해 목표된 애플리케이션으로 복사되어 오브젝트 파일과 독립적으로 실행할 수 있는 실행 파일을 생성하는데 사용된다. 예컨대, 윈도우의 '.LIB' 파일과 같이 '.a'의 확장자를 갖고 있다. 한편, 동

적 라이브러리는 프로그램 수행 도중 해당 모듈이 필요할 때 불러오는 프로그램 모듈이다. 즉, 윈도우에서는 주로 'DLL' 확장자를 가지며, 리눅스에서는 주로 'SO' 확장자를 가진다.

[0039] 제 1 실시예에 따른 중간 언어 변환 장치(120)에서의 컴파일 과정에 대해 설명하자면, 중간 언어 변환 장치(120)는 전처리가 수행된 소스 파일을 플랫폼(130)의 독립적인 중간 언어 코드로 컴파일하는 독립적 컴파일 과정을 거치고, 독립적인 중간 언어 코드를 플랫폼(130)의 종속적인 중간 언어 코드로 컴파일하는 종속적 컴파일 과정을 거친다.

[0040] 본 실시예에 기재된, '컴파일'이란 사람이 이해할 수 있는 컴퓨터언어를 기계가 이해할 수 있는 기계어로 통역해주는 과정을 말한다. 일반적으로 컴퓨터언어 프로그램이라 부르는 터보C, 볼랜드C++, 비주얼베이직 등이 컴파일러 프로그램으로, 컴퓨터언어 프로그램을 이용해서 컴퓨터 프로그램을 만든다. 이러한, 컴파일 과정은 '컴파일러'를 이용하여 수행할 수 있는데, 컴파일러는 소스 파일을 기계어로 번역해주는 프로그램을 말한다. 이러한, C 언어 컴파일러는 C 언어 문법에 맞추어서 쓴 명령어들을 컴퓨터가 이해할 수 있는 기계어로 번역해주는 프로그램이다. 즉, 컴파일러는 사람들이 알아볼 수 있도록 숫자와 문자로 명령을 기록한 내용을 기계가 이해할 수 있는 언어인 기계어로 번역해준다.

[0041] 이때, 제 1 실시예에 따른 중간 언어 변환 장치(120)는 독립적 컴파일 과정을 수행하기 위해 특정 지시자를 갖는 소스 파일을 중간 언어 코드로 변환한 후 중간 언어 코드를 플랫폼(130)의 독립적인 중간 언어 코드로 컴파일할 때 특정 지시자를 특정 연산자로 변환한다. 또한, 제 1 실시예에 따른 중간 언어 변환 장치(120)는 종속적 컴파일 과정을 수행하기 위해 독립적인 중간 언어 코드를 플랫폼의 종속적인 중간 언어 코드로 컴파일할 때 독립적인 중간 언어 코드에 포함된 특정 연산자를 플랫폼(130)에 따른 값을 갖는 상수로 변환한다. 여기서, '상수'는 한번 값을 저장하면 변경이 불가능한 값을 말하며, 변수와 마찬가지로 상수도 값을 저장할 수 있는 메모리 공간을 가지나, 변수와 다르게 데이터의 변경은 불가능하다. 한편, 중간 언어 변환 장치(120)는 종속적인 컴파일 과정을 수행할 때 독립적인 중간 언어 코드를 종속적인 중간 언어 코드로 컴파일할 때 문법 오류 체크를 수행하며, 확인 결과, 문법 오류 체크에 오류가 없는 경우 실행 파일 생성 과정에서 실행 파일을 생성한다. 이때, 특정 지시자는 'sizeof ()'를 말하며, 'sizeof ()'란 피연자의 크기를 바이트 단위로 계산해서 변환하는 연산자를 말한다.

[0042] 한편, 제 1 실시예에 따른 중간 언어 변환 장치(120)가 적용될 여지가 많다고 여겨지는 분야에 대해 설명하자면, 중간 언어 변환 장치(120)는 일종에 '애플리케이션 스토어'에 적용될 수 있다. 즉, 중간 언어 변환 장치(120)가 적용된 애플리케이션 스토어는 개발자 단말기(110)로부터 소스 파일만을 수신한 후 이를 중간 언어 코드로 변환하고, 해당 애플리케이션 스토어에 접속한 플랫폼(130)에 해당하는 형태로 컴파일한 후 해당 플랫폼(130)으로 해당 애플리케이션을 전송하여 수익을 창출할 수 있을 것이다.

[0043] 한편, 제 2 실시예에 따른 중간 언어 변환 장치(120)는 개발자 단말기(110)로부터 생성된 플랫폼 독립적인 중간 언어 코드를 읽어 들이고 이를 특정 플랫폼에 맞게 플랫폼 종속적인 중간 언어 코드로 변환한다. 또한, 중간 언어 변환 장치(120)는 플랫폼의 종속적인 중간 언어 코드를 플랫폼(130)에 맞는 실행 파일로 변환한다. 한편, 제 3 실시예에 따른 중간 언어 변환 장치(120)는 개발자 단말기(110)로부터 생성된 플랫폼 독립적인 중간 언어 코드를 읽어 들이고 이를 특정 플랫폼에 맞게 플랫폼 독립적인 중간 언어 코드로 변환한다.

[0044] 플랫폼(130)은 중앙처리장치의 아키텍처(Architecture) 및 운영체제(Operating System; OS) 중 적어도 하나 이상이 조합된 장치를 말한다. 즉, 중앙처리장치의 아키텍처는 ARM의 ARMv5, ARMv7, ARMv8, X86 또는 X64 등이 될 수 있으며, 운영체제는 윈도우, 리눅스, 윈도우 모바일, 안드로이드, OSX 또는 iOS 등이 될 수 있으나 반드시 이에 한정되는 것은 아니다. 이러한, 플랫폼(130)이란 하나 또는 복수 개의 처리장치(예컨대, CPU 등)를 포함하는 하드웨어, 하드웨어를 이용하여 동작하는 운영체제(Operating System; OS), 또는 하드웨어 및 운영체제 모두를 지칭한다. 예컨대 플랫폼(130)은 데스크탑 컴퓨터(Desktop Computer), 노트북(Notebook) 컴퓨터, 휴대전화(Cellular Phone) 등의 이동 장치(Mobile Device), PDA(Personal Digital Assistant) 등일 수 있으나 이에 한정되는 것은 아니다. 한편, 플랫폼(130)은 다양한 장치(제 1 플랫폼, 제 2 플랫폼 내지 제 N 플랫폼)를 포함할 수 있다.

[0045] 제 1 실시예 및 제 2 실시예에 따른 플랫폼(130)은 중간 언어 변환 장치(120)로부터 실행파일을 전송받아 구동한다. 한편, 제 3 실시예에 따른 플랫폼(130)은 중간 언어 변환 장치(120)로부터 플랫폼의 종속적인 중간 언어 코드를 전송받고 이를 플랫폼(130)에 맞게 실행파일로 변환 후 구동한다. 한편, 제 4 실시예에 따른 각 플랫폼(130)은 개발자 단말기(110)로부터 실행파일을 전송받아 구동한다.

[0046] 도 2는 본 실시예에 따른 중간 언어 변환 장치를 개략적으로 나타낸 블럭 구성도이다.

[0047] 제 1 실시예에 따른 중간 언어 변환 장치(120)는 전처리 수행부(220), 컴파일부(230), 실행 파일 생성부(240) 및 실행 파일 구동부(250)를 포함한다. 제 1 실시예에서는 중간 언어 변환 장치(120)가 소스 파일 생성부(210), 전처리 수행부(220), 컴파일부(230), 실행 파일 생성부(240) 만을 포함하는 것으로 기재하고 있으나, 이는 제 1 실시예의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 제 1 실시예가 속하는 기술 분야에서 통상의 지식을 가진 자라면 제 1 실시예의 본질적인 특성에서 벗어나지 않는 범위에서 중간 언어 변환 장치(120)에 포함되는 구성 요소에 대하여 다양하게 수정 및 변형하여 적용 가능할 것이다. 여기서, 소스 파일 생성부(210)는 개발자 단말기(110)에 포함될 수 있으며, 실행 파일 구동부(250)는 플랫폼(130)에 포함될 수 있다.

[0048] 개발자 단말기(110)에 포함된 소스 파일 생성부(210)는 C 언어 기반의 소스 파일을 생성한다. 이때, 중간 언어 변환 장치(120)에 포함된 전처리 수행부(220)는 소스 파일에 대한 전처리를 수행한다.

[0049] 중간 언어 변환 장치(120)에 포함된 컴파일부(230)는 전처리가 수행된 소스 파일을 중간 언어 코드로 변환하고, 중간 언어 코드를 플랫폼에 따라 컴파일할 때 특정 지시자가 특정 연산자를 경유하도록 한다. 여기서, 특정 지시자는 'sizeof ()'를 말한다. 이러한, 중간 언어 변환 장치(120)에 포함된 컴파일부(230)는 독립적 컴파일부(232)와 종속적 컴파일부(234)를 포함하는데, 이러한 독립적 컴파일부(232)와 종속적 컴파일부(234)에 대해 각각 설명하자면 다음과 같다. 독립적 컴파일부(232)는 중간 언어 코드를 플랫폼의 독립적인 중간 언어 코드로 컴파일한다. 또한, 독립적 컴파일부(232)는 특정 지시자를 갖는 소스 파일을 중간 언어 코드로 변환한 후 중간 언어 코드를 플랫폼(130)의 독립적인 중간 언어 코드로 컴파일할 때 특정 지시자를 특정 연산자로 변환한다. 한편, 종속적 컴파일부(234)는 독립적인 중간 언어 코드를 플랫폼(130)의 종속적인 중간 언어 코드로 컴파일한다. 또한, 종속적 컴파일부(234)는 독립적인 중간 언어 코드를 플랫폼(130)의 종속적인 중간 언어 코드로 컴파일할 때 독립적인 중간 언어 코드에 포함된 특정 연산자를 플랫폼(130)에 따른 값을 갖는 상수로 변환한다. 또한, 종속적인 컴파일부(230)는 독립적인 중간 언어 코드를 종속적인 중간 언어 코드로 컴파일할 때 문법 오류 체크를 수행하며, 확인 결과, 문법 오류 체크에 오류가 없는 경우 실행 파일 생성부(240)에서 실행 파일을 생성한다.

[0050] 중간 언어 변환 장치(120)에 포함된 실행 파일 생성부(240)는 컴파일된 중간 언어 코드에 대한 실행 파일을 생성한다. 또한, 플랫폼(130)에 포함된 실행 파일 구동부(250)는 실행 파일이 플랫폼(130)에서 구동되도록 한다.

[0051] 도 3은 제 2 실시예에 따른 중간 언어 변환 시스템을 개략적으로 나타낸 블럭 구성도이다.

[0052] 제 2 실시예에 따른 개발자 단말기(110)는 소스 파일 생성부(210), 전처리 수행부(220) 및 독립적 컴파일부(232)를 포함하고, 중간 언어 변환 장치(120)는 종속적 컴파일부(234) 및 실행 파일 생성부(240)를 포함하고, 플랫폼(130)은 실행 파일 구동부(250)를 포함한다. 제 2 실시예에서, 개발자 단말기(110), 중간 언어 변환 장치(120) 및 플랫폼(130)에 포함된 각 모듈은 제 2 실시예의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 제 2 실시예가 속하는 기술 분야에서 통상의 지식을 가진 자라면 본질적인 특성에서 벗어나지 않는 범위에서 다양하게 수정 및 변형하여 적용 가능할 것이다.

[0053] 개발자 단말기(110)에 포함된 소스 파일 생성부(210)는 C 언어 기반의 소스 파일을 생성한다. 개발자 단말기(110)에 포함된 전처리 수행부(220)는 소스 파일에 대한 전처리를 수행한다. 개발자 단말기(110)에 포함된 독립적 컴파일부(232)는 중간 언어 코드를 플랫폼의 독립적인 중간 언어 코드로 컴파일한다. 또한, 개발자 단말기(110)에 포함된 독립적 컴파일부(232)는 특정 지시자를 갖는 소스 파일을 중간 언어 코드로 변환한 후 중간 언어 코드를 플랫폼(130)의 독립적인 중간 언어 코드로 컴파일할 때 특정 지시자를 특정 연산자로 변환한다. 여기서, 특정 지시자는 'sizeof ()'를 말한다.

[0054] 한편, 중간 언어 변환 장치(120)에 포함된 종속적 컴파일부(234)는 독립적인 중간 언어 코드를 플랫폼(130)의 종속적인 중간 언어 코드로 컴파일한다. 또한, 중간 언어 변환 장치(120)에 포함된 종속적 컴파일부(234)는 독립적인 중간 언어 코드를 플랫폼(130)의 종속적인 중간 언어 코드로 컴파일할 때 독립적인 중간 언어 코드에 포함된 특정 연산자를 플랫폼(130)에 따른 값을 갖는 상수로 변환한다. 또한, 중간 언어 변환 장치(120)에 포함된 종속적인 컴파일부(230)는 독립적인 중간 언어 코드를 종속적인 중간 언어 코드로 컴파일할 때 문법 오류 체크를 수행하며, 확인 결과, 문법 오류 체크에 오류가 없는 경우 실행 파일 생성부(240)에서 실행 파일을 생성한다. 중간 언어 변환 장치(120)에 포함된 실행 파일 생성부(240)는 컴파일된 중간 언어 코드에 대한 실행 파일을 생성한다. 또한, 플랫폼(130)에 포함된 실행 파일 구동부(250)는 실행 파일이 플랫폼(130)에서 구동되도록 한다.

[0055] 도 4는 제 3 실시예에 따른 중간 언어 변환 시스템을 개략적으로 나타낸 블럭 구성도이다.

[0056] 제 3 실시예에 따른 개발자 단말기(110)는 소스 파일 생성부(210), 전처리 수행부(220) 및 독립적 컴파일부(232)를 포함하고, 중간 언어 변환 장치(120)는 종속적 컴파일부(234)를 포함하고, 플랫폼(130)은 실행 파일 생성부(240) 및 실행 파일 구동부(250)를 포함한다. 제 3 실시예에서, 개발자 단말기(110), 중간 언어 변환 장치(120) 및 플랫폼(130)에 포함된 각 모듈은 제 3 실시예의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 제 3 실시예가 속하는 기술 분야에서 통상의 지식을 가진 자라면 본질적인 특성에서 벗어나지 않는 범위에서 다양하게 수정 및 변형하여 적용 가능할 것이다.

[0057] 개발자 단말기(110)에 포함된 소스 파일 생성부(210)는 C 언어 기반의 소스 파일을 생성한다. 개발자 단말기(110)에 포함된 전처리 수행부(220)는 소스 파일에 대한 전처리를 수행한다. 개발자 단말기(110)에 포함된 독립적 컴파일부(232)는 중간 언어 코드를 플랫폼의 독립적인 중간 언어 코드로 컴파일한다. 또한, 개발자 단말기(110)에 포함된 독립적 컴파일부(232)는 특정 지시자를 갖는 소스 파일을 중간 언어 코드로 변환한 후 중간 언어 코드를 플랫폼(130)의 독립적인 중간 언어 코드로 컴파일할 때 특정 지시자를 특정 연산자로 변환한다. 여기서, 특정 지시자는 'sizeof ()'를 말한다.

[0058] 한편, 중간 언어 변환 장치(120)에 포함된 종속적 컴파일부(234)는 독립적인 중간 언어 코드를 플랫폼(130)의 종속적인 중간 언어 코드로 컴파일한다. 또한, 중간 언어 변환 장치(120)에 포함된 종속적 컴파일부(234)는 독립적인 중간 언어 코드를 플랫폼(130)의 종속적인 중간 언어 코드로 컴파일할 때 독립적인 중간 언어 코드에 포함된 특정 연산자를 플랫폼(130)에 따른 값을 갖는 상수로 변환한다. 또한, 중간 언어 변환 장치(120)에 포함된 종속적인 컴파일부(230)는 독립적인 중간 언어 코드를 종속적인 중간 언어 코드로 컴파일할 때 문법 오류 체크를 수행하며, 확인 결과, 문법 오류 체크에 오류가 없는 경우 실행 파일 생성부(240)에서 실행 파일을 생성한다.

[0059] 또한, 플랫폼(130)에 포함된 실행 파일 생성부(240)는 컴파일된 중간 언어 코드에 대한 실행 파일을 생성한다. 플랫폼(130)에 포함된 실행 파일 구동부(250)는 실행 파일이 플랫폼(130)에서 구동되도록 한다.

[0060] 도 5는 제 4 실시예에 따른 중간 언어 변환 시스템을 개략적으로 나타낸 블럭 구성도이다.

[0061] 제 4 실시예에 따른 개발자 단말기(110)는 소스 파일 생성부(210), 전처리 수행부(220), 독립적 컴파일부(232), 종속적 컴파일부(234) 및 실행 파일 생성부(240)를 포함하고, 플랫폼(130)은 실행 파일 구동부(250)를 포함한다. 제 4 실시예에서, 개발자 단말기(110) 및 플랫폼(130)에 포함된 각 모듈은 제 4 실시예의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 제 4 실시예가 속하는 기술 분야에서 통상의 지식을 가진 자라면 본질적인 특성에서 벗어나지 않는 범위에서 다양하게 수정 및 변형하여 적용 가능할 것이다.

[0062] 개발자 단말기(110)에 포함된 소스 파일 생성부(210)는 C 언어 기반의 소스 파일을 생성한다. 개발자 단말기(110)에 포함된 전처리 수행부(220)는 소스 파일에 대한 전처리를 수행한다. 개발자 단말기(110)에 포함된 독립적 컴파일부(232)는 중간 언어 코드를 플랫폼의 독립적인 중간 언어 코드로 컴파일한다. 또한, 개발자 단말기(110)에 포함된 독립적 컴파일부(232)는 특정 지시자를 갖는 소스 파일을 중간 언어 코드로 변환한 후 중간 언어 코드를 플랫폼(130)의 독립적인 중간 언어 코드로 컴파일할 때 특정 지시자를 특정 연산자로 변환한다. 여기서, 특정 지시자는 'sizeof ()'를 말한다.

[0063] 한편, 개발자 단말기(110)에 포함된 종속적 컴파일부(234)는 독립적인 중간 언어 코드를 플랫폼(130)의 종속적인 중간 언어 코드로 컴파일한다. 또한, 개발자 단말기(110)에 포함된 종속적 컴파일부(234)는 독립적인 중간 언어 코드를 플랫폼(130)의 종속적인 중간 언어 코드로 컴파일할 때 독립적인 중간 언어 코드에 포함된 특정 연산자를 플랫폼(130)에 따른 값을 갖는 상수로 변환한다. 또한, 개발자 단말기(110)에 포함된 종속적인 컴파일부(230)는 독립적인 중간 언어 코드를 종속적인 중간 언어 코드로 컴파일할 때 문법 오류 체크를 수행하며, 확인 결과, 문법 오류 체크에 오류가 없는 경우 개발자 단말기(110)에 포함된 실행 파일 생성부(240)에서 실행 파일을 생성한다. 개발자 단말기(110)에 포함된 실행 파일 생성부(240)는 컴파일된 중간 언어 코드에 대한 실행 파일을 생성한다. 한편, 플랫폼(130)에 포함된 실행 파일 구동부(250)는 실행 파일이 플랫폼(130)에서 구동되도록 한다.

[0064] 도 6은 본 실시예에 따른 중간 언어 변환 방법을 설명하기 위한 순서도이다.

[0065] 소스 파일 생성부(210)는 C 언어 기반의 소스 파일을 생성한다(S610). 단계 S610의 소스 파일 생성부(210)는 제 1 실시예 내지 제 4 실시예에 따라 개발자 단말기(110)에 포함될 수 있다. 전처리 수행부(220)는 소스 파일에

대한 전처리를 수행한다(S620). 단계 S620에서의 전처리 수행부(220)는 제 1 실시예에 따라 중간 언어 변환 장치(120)에 포함될 수 있고, 제 2 실시예 내지 제 4 실시예에 따라 개발자 단말기(110)에 포함될 수 있다.

[0066] 독립적 컴파일부(232) 및 종속적 컴파일부(234)는 소스 파일을 중간 언어 코드로 변환하고, 중간 언어 코드를 플랫폼(130)에 따라 컴파일할 때 특정 지시자가 특정 연산자를 경유하도록 한다(S630). 단계 S630에서의 독립적 컴파일부(232)는 제 1 실시예에 따라 중간 언어 변환 장치(120)에 포함될 수 있고, 제 2 실시예 내지 제 4 실시예에 따라 개발자 단말기(110)에 포함될 수 있다. 한편, 종속적 컴파일부(234)는 제 1 실시예 내지 제 3 실시예에 따라 중간 언어 변환 장치(120)에 포함될 수 있고, 제 4 실시예에 따라 개발자 단말기(110)에 포함될 수 있다.

[0067] 단계 S630에서 독립적 컴파일부(232)는 플랫폼(130)에 독립적인 중간 언어 코드로 변환할 수 있다. 이러한, 중간 언어 코드로의 변환은 미리 저장된 C 언어 기반의 프로그래밍 언어와 중간 언어 코드 사이의 매칭 정보를 이용하여 수행될 수 있다. 그 결과 애플리케이션의 소스 파일을, 특정 CPU 또는 운영 체제에 의존적이지 않은 중간 언어 코드로 변환할 수 있는 것이다. 한편, 단계 S630에서 독립적 컴파일부(232)에서 전처리가 수행된 소스 파일을 플랫폼(130)의 독립적인 중간 언어 코드로 컴파일하는 독립적 컴파일 과정을 거치고, 종속적 컴파일부(234)에서 독립적인 중간 언어 코드를 플랫폼(130)의 종속적인 중간 언어 코드로 컴파일하는 종속적 컴파일 과정을 거친다. 이때, 단계 S630에서, 독립적 컴파일부(232)는 독립적 컴파일 과정을 수행하기 위해 특정 지시자를 갖는 소스 파일을 중간 언어 코드로 변환한 후 중간 언어 코드를 플랫폼(130)의 독립적인 중간 언어 코드로 컴파일할 때 특정 지시자를 특정 연산자로 변환한다.

[0068] 또한, 종속적 컴파일부(234)는 종속적 컴파일 과정을 수행하기 위해 독립적인 중간 언어 코드를 플랫폼의 종속적인 중간 언어 코드로 컴파일할 때 독립적인 중간 언어 코드에 포함된 특정 연산자를 플랫폼(130)에 따른 값을 갖는 상수로 변환한다. 여기서, '상수'는 한번 값을 저장하면 변경이 불가능한 값을 말하며, 변수와 마찬가지로 상수도 값을 저장할 수 있는 메모리 공간을 가지나, 변수와 다르게 데이터의 변경은 불가능하다. 또한, 단계 S630에서 종속적 컴파일부(234)는 종속적인 컴파일 과정을 수행할 때 독립적인 중간 언어 코드를 종속적인 중간 언어 코드로 컴파일할 때 문법 오류 체크를 수행하며, 확인 결과, 문법 오류 체크에 오류가 없는 경우 실행 파일 생성 과정에서 실행 파일을 생성한다. 이때, 특정 지시자는 'sizeof ()'를 말하며, 'sizeof ()'란 피연자의 크기를 바이트 단위로 계산해서 변환하는 연산자를 말한다.

[0069] 실행 파일 생성부(240)는 컴파일된 중간 언어 코드에 대한 실행 파일을 생성한다(S640). 단계 S640에서 중간 언어 변환 장치(120)는 자동으로 또는 하나 이상의 플랫폼으로부터의 다운로드 요청에 의해, 중간 언어 코드의 형태로 되어 있는 애플리케이션을 해당 플랫폼에 대응되는 실행 파일을 생성할 수 있다. 단계 S640에서의 실행 파일 생성부(240)는 제 1 실시예 및 제 2 실시예에 따라 중간 언어 변환 장치(120)에 포함될 수 있고, 제 3 실시예에 따라 플랫폼(130)에 포함될 수 있고, 제 4 실시예에 따라 개발자 단말기(110)에 포함될 수 있다.

[0070] 실행 파일 구동부(250)는 실행 파일이 플랫폼(130)에서 구동되도록 한다(S650). 단계 S650에서, 실행 파일 구동부(250)는 하나 이상의 동일하거나 또는 서로 상이한 종류의 플랫폼(130)과 통신할 수 있으며, 플랫폼(130)의 CPU 또는 운영체제 정보 등과 같은 플랫폼의 프로파일을 수신할 수 있다. 즉, 실행 파일 구동부(250)에서 중간 언어 코드가 플랫폼(130)에 대응되는 실행 파일의 형태로 변환되므로, 플랫폼(130)에서는 애플리케이션을 플랫폼(130)에 최적화된 실행파일 등의 형태로 다운로드 또는 실행할 수 있다. 단계 S650에서의 실행 파일 구동부(250)는 제 1 실시예 및 제 4 실시예에 따라 플랫폼(130)에 포함될 수 있다.

[0071] 도 6에서는 단계 S610 내지 단계 S650을 순차적으로 실행하는 것으로 기재하고 있으나, 이는 본 실시예의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 본 실시예가 속하는 기술 분야에서 통상의 지식을 가진 자라면 본 실시예의 본질적인 특성에서 벗어나지 않는 범위에서 도 3에 기재된 순서를 변경하여 실행하거나 단계 S610 내지 단계 S650 중 하나 이상의 단계를 병렬적으로 실행하는 것으로 다양하게 수정 및 변형하여 적용 가능할 것이므로, 도 3은 시계열적인 순서로 한정되는 것은 아니다.

[0072] 전술한 바와 같이 도 6에 기재된 본 실시예에 따른 중간 언어 변환 방법은 프로그램으로 구현되고 컴퓨터로 읽을 수 있는 기록매체에 기록될 수 있다. 본 실시예에 따른 중간 언어 변환 방법을 구현하기 위한 프로그램이 기록되고 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록장치를 포함한다. 이러한 컴퓨터가 읽을 수 있는 기록매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피디스크, 광 데이터 저장장치 등이 있으며, 또한 캐리어 웨이브(예를 들어, 인터넷을 통한 전송)의 형태로 구현되는 것도 포함한다. 또한 컴퓨터가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어, 분산방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수도 있다. 또한, 본 실시예를 구현하

기 위한 기능적인(Functional) 프로그램, 코드 및 코드 세그먼트들은 본 실시예가 속하는 기술분야의 프로그래머들에 의해 용이하게 추론될 수 있을 것이다.

[0073] 도 7은 본 실시예에 따른 중간 언어 변환 과정을 설명하기 위한 예시도이다.

[0074] 도 7의 (A)에 도시된 바와 같이, C 언어 기반의 'sizeof()'는 각각의 플랫폼에 따라 그 값이 달라질 수 있는데, 일반적으로 컴파일 타임에 따라 해당 값이 결정되기 때문에 중간 언어 코드로 변환 시 '상수'로 표현된다. 이는 C 언어 기반의 소스 파일을 독립적인 중간 언어 코드로 변환하는데 문제점으로 작용한다. 즉, 도 7의 (A)에 도시된 바와 같이, 일반적인 컴파일러를 이용하여 sizeof()를 컴파일하는 과정에서 '상수'로 변환(예컨대 'sizeof(a)→4')하고, 이러한 '상수'로 인해 발생할 수 있는 문법 에러들을 컴파일 과정에서 체크한 후 에러가 발생되지 않는 경우 중간 언어 코드로 생성한다. 하지만, 이는 C 언어로부터 특정 플랫폼에 의존적이지 않은 독립적인 중간 언어 코드를 생성하는데 문제점으로 작용할 수 있다.

[0075] 한편, 도 7의 (B)에 도시된 바와 같이, 플랫폼에 따라 값이 다른 값을 갖는 'sizeof()'를 컴파일 과정에서 '상수'가 아닌 '특정 연산자'로 표현한다. 또한, 컴파일 과정에서 관련된 문법 체크 없이 '특정 연산자'가 컴파일된 플랫폼의 독립적인 중간 언어 코드 내에 그대로 유지(예컨대, 'sizeof(a)→sizeof(a)')되며 플랫폼의 독립적인 중간 언어 코드가 플랫폼의 종속적인 중간 언어 코드로 변환될 때 '상수'로 변환(예컨대, 'sizeof(a)→4')된다. 이러한 '상수'로 변환된 후 해당 '상수'로 인해 문법적인 에러가 발생하였는지를 체크하는 과정을 통해 일반적인 컴파일러와 같이 문법 체크 기능을 수행할 수 있다.

[0076] 이상의 설명은 본 실시예의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 본 실시예가 속하는 기술 분야에서 통상의 지식을 가진 자라면 본 실시예의 본질적인 특성에서 벗어나지 않는 범위에서 다양한 수정 및 변형이 가능할 것이다. 따라서, 본 실시예들은 본 실시예의 기술 사상을 한정하기 위한 것이 아니라 설명하기 위한 것이고, 이러한 실시예에 의하여 본 실시예의 기술 사상의 범위가 한정되는 것은 아니다. 본 실시예의 보호 범위는 아래의 청구범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술 사상은 본 실시예의 권리범위에 포함되는 것으로 해석되어야 할 것이다.

부호의 설명

[0077] 110: 개발자 단말기 120: 중간 언어 변환 장치

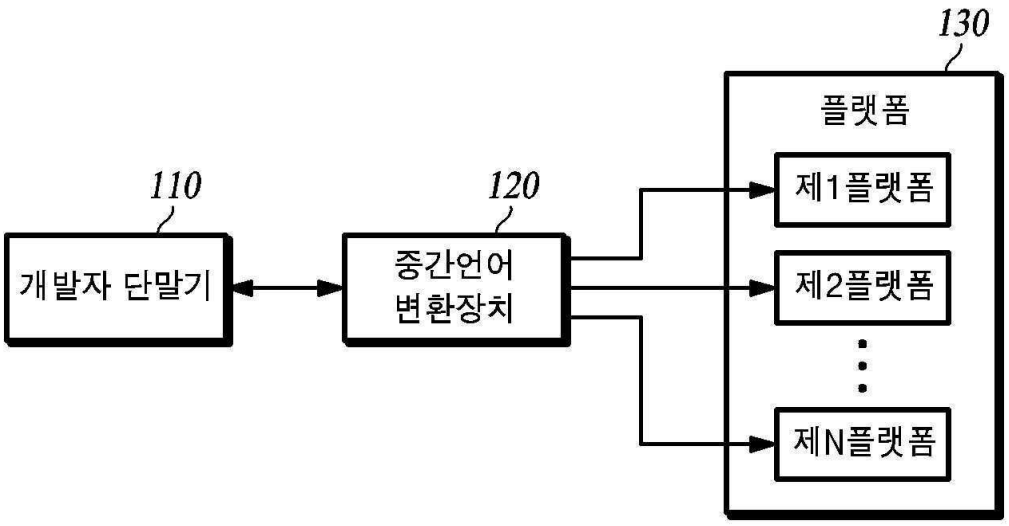
130: 플랫폼 210: 소스 파일 생성부

220: 전처리 수행부 230: 컴파일부

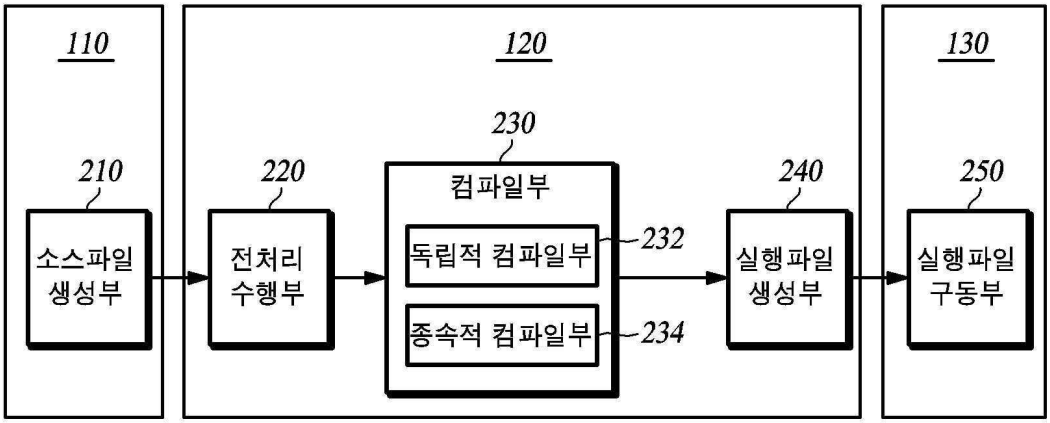
240: 실행 파일 생성부 250: 실행 파일 구동부

도면

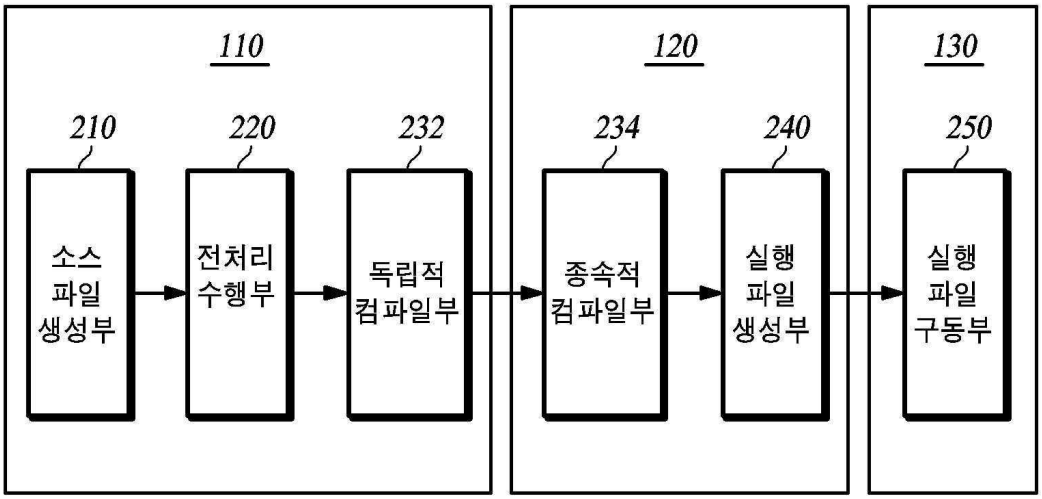
도면1



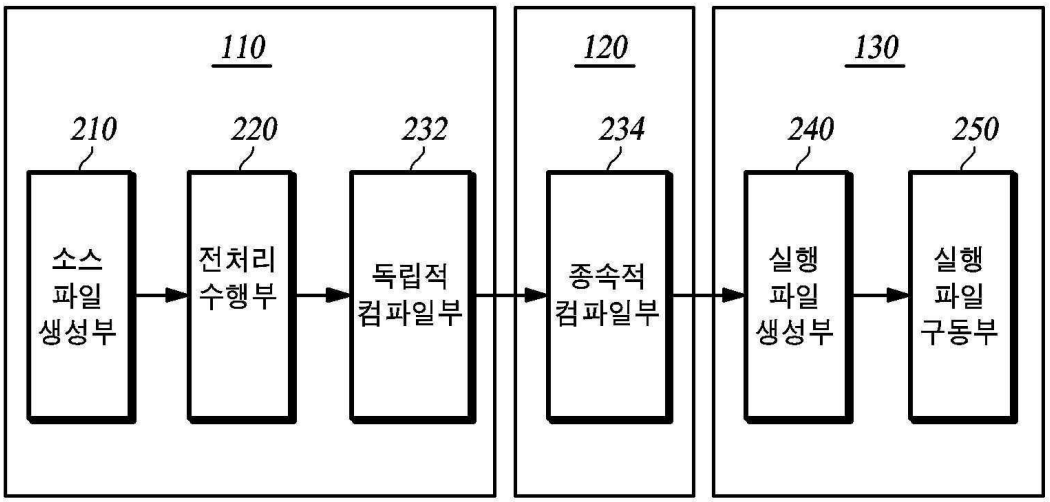
도면2


110
120
130
230
210
220
컴파일부
240
250
소스파일
생성부
전처리
수행부
독립적 컴파일부
232
실행파일
생성부
실행파일
구동부
종속적 컴파일부
234

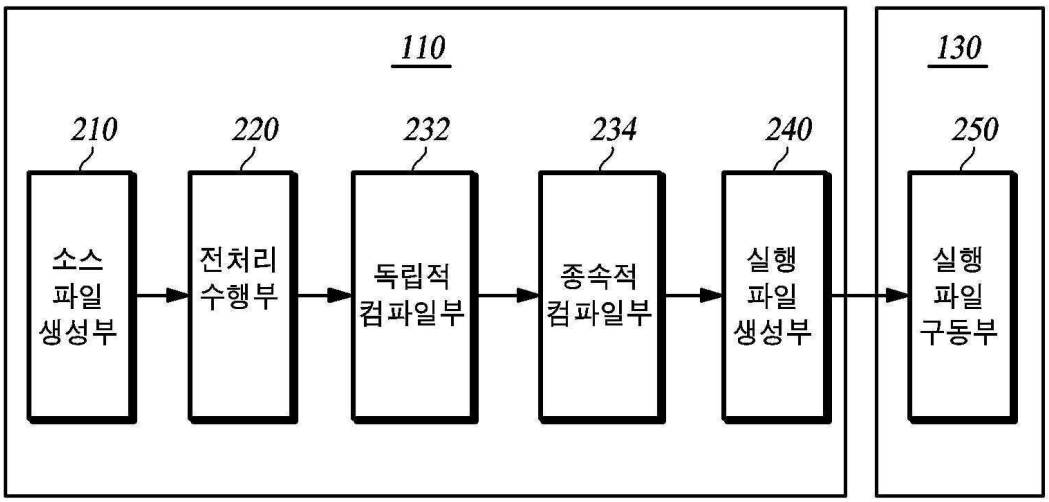
도면3


110
120
130
210
220
232
234
240
250
소스
파일
생성부
전처리
수행부
독립적
컴파일부
종속적
컴파일부
실행
파일
생성부
실행
파일
구동부

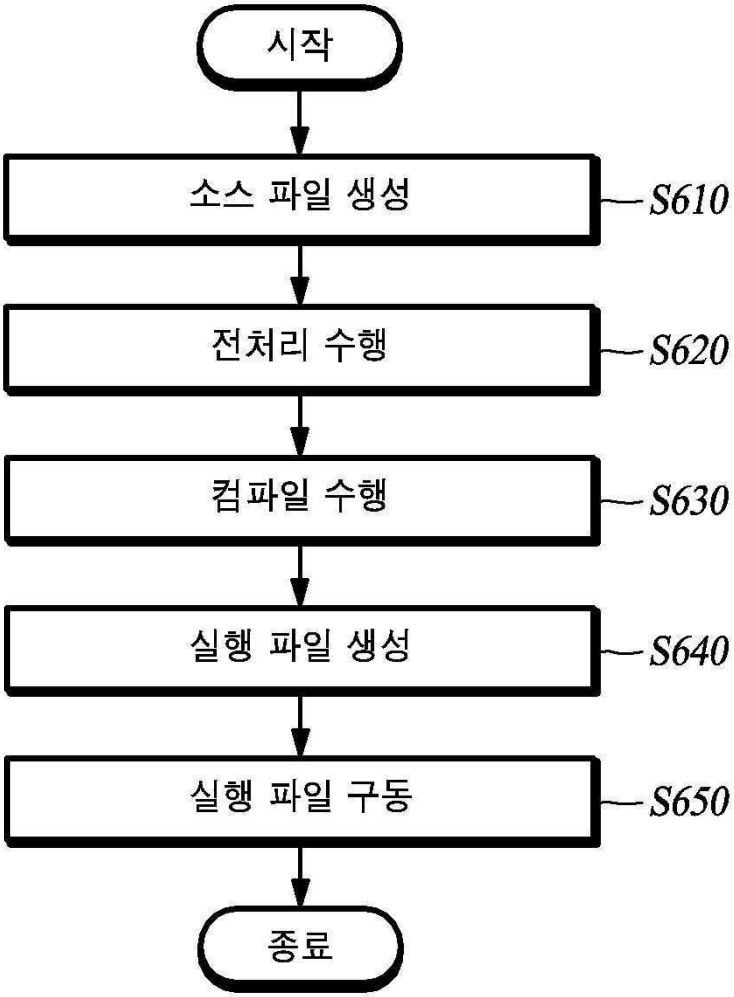
도면4


110
120
130
210
220
232
234
240
250
소스
파일
생성부
전처리
수행부
독립적
컴파일부
종속적
컴파일부
실행
파일
생성부
실행
파일
구동부

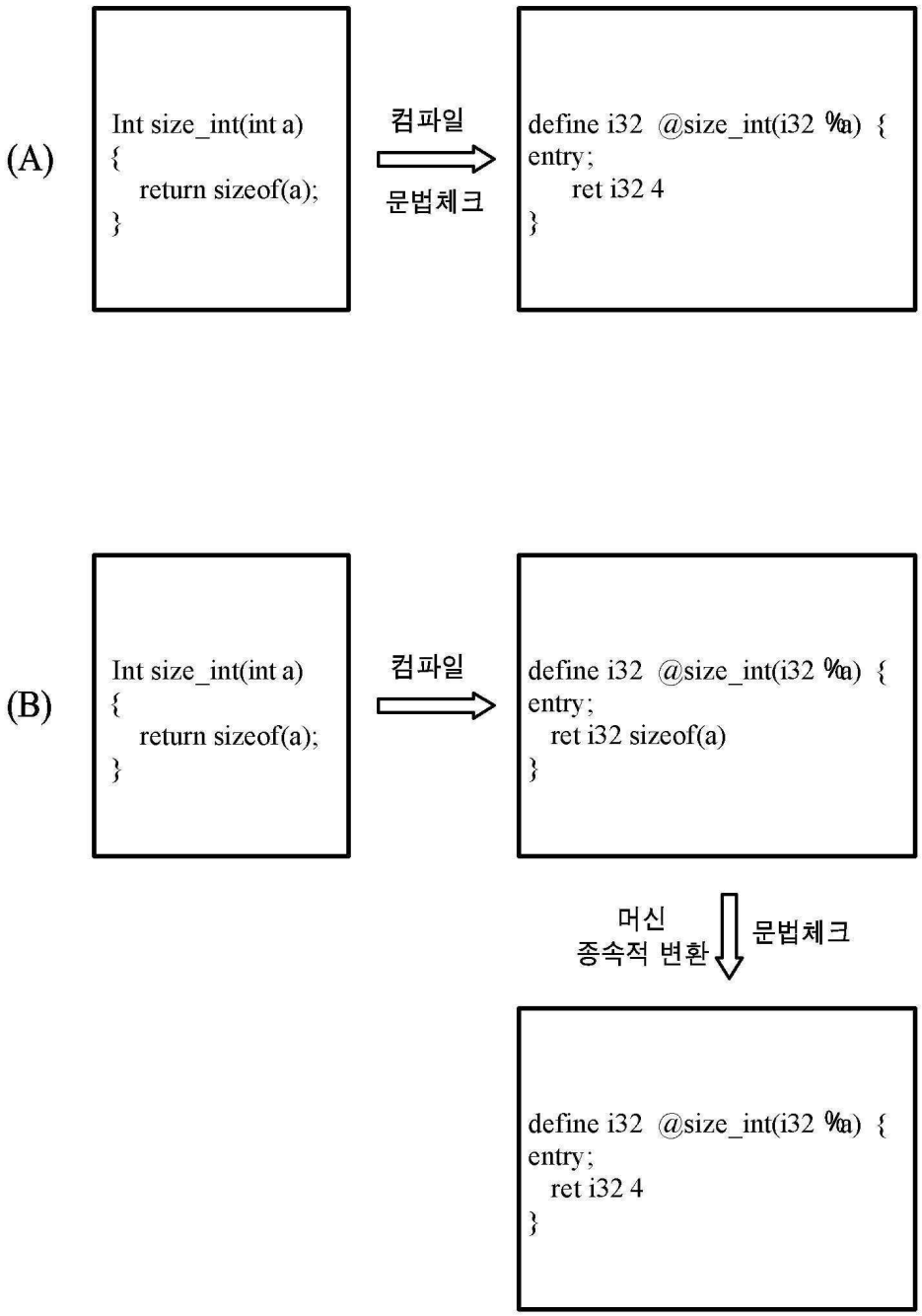
도면5


110
130
210
220
232
234
240
250
소스 파일 생성부
전처리 수행부
독립적 컴파일부
종속적 컴파일부
실행 파일 생성부
실행 파일 구동부

도면6


시작
소스 파일 생성
S610
전처리 수행
S620
컴파일 수행
S630
실행 파일 생성
S640
실행 파일 구동
S650
종료

도면7


(A)
Int size_int(int a)
{
return sizeof(a);
}
컴파일
문법체크
define i32 @size_int(i32 %a) {
entry;
ret i32 4
}
(B)
Int size_int(int a)
{
return sizeof(a);
}
컴파일
define i32 @size_int(i32 %a) {
entry;
ret i32 sizeof(a)
}
머신
종속적 변환
문법체크
define i32 @size_int(i32 %a) {
entry;
ret i32 4
}