



УКРАЇНА

(19) UA (11) 55438 (13) C2

(51) 7 G06F9/42, G06F12/14

МІНІСТЕРСТВО ОСВІТИ
І НАУКИ УКРАЇНИДЕРЖАВНИЙ ДЕПАРТАМЕНТ
ІНТЕЛЕКТУАЛЬНОЇ
ВЛАСНОСТІОПИС
ДО ПАТЕНТУ НА ВІНАХІД

(54) СПОСІБ КОНТРОЛЮ ПРИПИСАНОГО ВИКОНАННЯ ПРОГРАМ

1

2

(21) 99074040

(22) 15 01 1998

(24) 15 04 2003

(86) PCT/DE98/00133, 15 01 1998

(31) 197 01 166 7

(32) 15 01 1997

(33) DE

(46) 15 04 2003, Бюл. № 4, 2003 р

(72) Бальдішвайлер Міхаель, DE, Пфаб Штефан, DE

(73) СІМЕНС АКЦІЕНГЕЗЕЛЬШАФТ, DE

(56) Uwe Wildner "Compiler Assisted Self-Checking of Struktural Integrity Using Address Hashing", Proceeding of 2nd European Dependable Computing Conference, EDCC-2, Taormina, Italy, 2 -4 10 1996, ISBN 3-540-61772-8, 1996, Berlin, Springer-Verlag, Seiten 151-177, XP002068242

DE 19614904, М кл. G06F 12/14, 1997

EP 0011136, М кл. G06F 13/00, G11C 7/00, 1980

US 5274817, М кл. G06F 7/00, G06F 15/00, 1993

EP 0010186, М кл. G06F 13/00, 1980

(57) 1 Спосіб контролю приписаного виконання програм, при якому запобігається перезапис адрес повернення, записаних для наступного використання, та/або використання як адрес повернення помилково записаних адрес повернення, або адрес повернення, щодо яких було здійснено перезапис, який відрізняється тим, що здійснюють захист від перезапису адрес повернення та захист від використання адрес повернення з оцінкою інформації захисту, яка генерується під час запису адрес повернення і записується у запам'ятовувальний пристрій захисту, і тим, що як запам'ятовувальний пристрій захисту використовують запам'ятовувальний пристрій, який не можна цілеспрямовано примусити спрацювати ззовні, з-за меж системи, яка виконує програму, що підлягає контролю

2 Спосіб за п. 1, який відрізняється тим, що як запам'ятовувальний пристрій захисту використовують запам'ятовувальний пристрій, передбачений у мікропроцесорі, мікроконтролері або сигнальному процесорі, що виконує програму

3 Спосіб за п. 1 або 2, який відрізняється тим, що як інформацію захисту використовують адресу повернення або дані, що відображають або характеризують цю адресу

4 Спосіб за п. 3, який відрізняється тим, що за-

писані дані перед їхнім використанням як адрес повернення зставляють з підпорядкованою інформацією захисту, причому використання даних як адрес повернення дозволяється лише у тому разі, якщо встановлено, що інформація захисту репрезентує або характеризує записані дані

5 Спосіб за п. 3 або 4, який відрізняється тим, що тоді, коли у разі запиту даних, що репрезентують адресу повернення, встановлюється, що інформація захисту не репрезентує або не характеризує дані, зазначені у відповіді на запит, ініціюють негайне припинення процесу виконання програми та/або повернення у вихідний стан системи, яка виконує програму, та/або тривогу, та/або стирання даних, що стосуються безпеки

6 Спосіб за п. 1 або 2, який відрізняється тим, що як інформацію захисту записують коди захисту від запису, що сигналізують про наявність захисту від запису

7 Спосіб за п. 6, який відрізняється тим, що попереджується перезапис даних в області запам'ятовувального пристрою, про наявність захисту від запису до яких сигналізують коди захисту від запису

8 Спосіб за п. 6 або 7, який відрізняється тим, що тоді, коли у разі спроби перезапису даних в області запам'ятовувального пристрою встановлюється, що підпорядкована інформація захисту сигналізує про наявність захисту від запису, ініціюють негайне припинення процесу виконання програми та/або повернення системи, що виконує програму, до вихідного стану, та/або тривогу, та/або стирання даних, що стосуються безпеки

9 Спосіб за будь-яким з пп. 6 - 8, який відрізняється тим, що перед використанням записаних даних як адреси повернення опитують підпорядковану інформацію захисту, причому використання даних як адрес повернення дозволяється лише у тому разі, якщо встановлюється, що інформація захисту сигналізує про наявність захисту від запису

10 Спосіб за п. 9, який відрізняється тим, що тоді, коли при запиті даних, які репрезентують адреси повернення, встановлюється, що підпорядкована інформація захисту не сигналізує про наявність захисту від запису, ініціюють негайне припинення процесу виконання програми та/або переведення до вихідного стану системи, що ви-

(13) C2

(11) 55438

(19) UA

конус програму, та/або тривогу, та/або стирання

даних, що стосуються безпеки

Цей винахід стосується способу згідно з обмежувальною частиною п. 1 формули винаходу, тобто способу контролю приписаного виконання програм математичного забезпечення.

Приписане виконання програм математичного забезпечення стає дедалі все більш реальним завданням завдяки вживанню різноманітних заходів.

Якщо раніше причиною порушення належного виконання програм математичного забезпечення були насамперед ненадійне апаратне забезпечення і помилки у програмуванні, сьогодні більш важливу роль відіграє цілеспрямоване маніпулювання процесом виконання програм.

Шляхом цілеспрямованого маніпулювання процесом виконання програм можна, наприклад, пропускати певні частини програм, завдяки чому створюється можливість виключення етапів перевірки для визначення повноважень щодо доступу.

Це може створити значні проблеми, наприклад, у галузі застосування чіп-карток, але, якщо розглядати проблему ширше, то не тільки для них, оскільки, з одного боку, вони набувають поширеного застосування, насамперед у галузях, пов'язаних з питаннями захисту (наприклад, для контролю доступу, фінансових операцій тощо), а з іншого боку, оскільки їх, природно, не можна постійно контролювати чи перевіряти, вони можуть легко стати предметом спроб маніпулювання.

У зв'язку з великою кількістю вже передбачених заходів безпеки ймовірність того, що спроба маніпуляції, метою якої є зловживання, виявиться вдалою, дуже мала. Проте, взагалі ймовірність цього виключити не можна.

Тому в основу цього винаходу було покладено завдання розробити спосіб контролю приписаного виконання програм математичного забезпечення, за допомогою якого можна, зокрема, надійно виключити можливість цілеспрямованого маніпулювання процесом виконання програми.

Це завдання згідно з винаходом вирішується за допомогою заходів, зазначених у відрізняльній частині п. 1 формули винаходу.

Відповідно до цього, попереджується можливість перезапису на місце адрес повернення, записаних у запам'ятовувальний пристрій для наступного використання, та/або використання як адрес повернення помилково записаних або перезаписаних адрес повернення.

На практиці реалізувати ці етапи завдання можна, вживаючи найрізноманітніших заходів. У найпростішому випадку у разі виклику функції, що потребує запису адреси повернення у пам'ять, або в аналогічних випадках записуються не лише адреси повернення, але й додаткова інформація захисту, що дозволяє зробити висновок про те, чи будуть ще потрібні записані адреси повернення і тому на їхнє місце не можна здійснювати перезапис, та/або чи є записана адреса повернення такою, що була записана у пам'ять спочатку, чи такою, яку треба записати.

У першому випадку, тобто у разі реалізації за-

хисту від запису інформації на місце адрес повернення, інформація захисту може складатись, наприклад, з коду захисту від запису, наприклад, біта захисту від запису або аналогічного, що додається під час запису адреси повернення і після використання записаної адреси повернення як адреси повернення стирається.

У другому випадку, тобто у разі реалізації захисту від використання адрес повернення, інформація захисту може складатись, наприклад, лише з адреси повернення або з даних, які іншим чином репрезентують або характеризують адресу повернення.

Згадана інформація захисту записується в область пам'яті, до якої переважно неможливий доступ ззовні, "нормальний" запис адрес повернення можна, як і раніше, здійснювати у так званій стек (стекову пам'ять) (Stack).

Якщо перед кожною спробою запису до стеку здійснюється перевірка щодо того, чи позначена за допомогою біту захисту від запису область, в яку треба здійснити запис, як така, що має захист від запису, у цьому разі можна попередити запис поверх даних, які потім мають бути використані як адреса повернення.

Якщо альтернативно або додатково перевіряється, чи відповідають дані, що мають бути використані як адреси повернення, адреси повернення, записані раніше або такі, що має бути записана, можна запобігти тому, щоб дані, які були змінені (тобто якими маніпулювали) після запису адреси повернення, були використані як адреса повернення.

В обох випадках для того, щоб попередити наступні спроби маніпулювання, можна припинити процес виконання актуальної програми та/або повернути до вихідного стану систему, що виконує програму, та/або ініціювати тривогу, та/або вжити інших заходів захисту.

Таким чином, можна забезпечити, щоб цілеспрямоване маніпулювання адресами повернення не могло б призвести до зміни приписаного виконання програми.

Отже, було розроблено спосіб для того, щоб, зокрема, можна було б надійно виключити можливість цілеспрямованого маніпулювання процесом виконання програми.

Інші ознаки винаходу наведені у залежних пунктах формули винаходу.

Суть винаходу пояснюється далі на прикладах реалізації за допомогою креслень. На них показано:

Фіг. 1 Фрагмент схеми системи для реалізації захисту від використання адреси повернення, та

Фіг. 2 Схема для пояснення способу захисту від запису поверх адрес повернення та модифікованого захисту від використання адрес повернення.

Систему, фрагмент якої наведений на фіг. 1, було розроблено для виконання програм математичного забезпечення, яка повністю або частково може бути інтегрована у мікропроцесор, мікрокон-

тропер, сигнальний процесор тощо

Показаний фрагмент є частиною системи для оперування адресами повернення

Адреси повернення слід записувати у пам'ять, наприклад, у тому разі, якщо програма, що підлягає виконанню, включає у себе виклик функції. У разі наявності виклику функції, наприклад, команди LCALL для мікроконтролера Intel 8051,

- у пам'яті програм виконується (адресний) перехід у місце, на якому записана програма для функції, яка підлягає виконанню на цей момент,

- виконується відповідна функціональна програма, та

- здійснюється повернення у місце у програмний пристрій пам'яті, з якого було здійснено перехід до функціональної програми

Остання згадана адреса, тобто адреса, за якою слід продовжувати виконання програми після виконання функціональної програми, є згаданою вище адресою повернення

Для того, щоб пристрій, який виконує програму математичного забезпечення, знав, у яке місце йому треба перейти після виконання функції, він потребує проміжного запису адреси повернення у пам'ять

Саме тепер слід зауважити, що виклики функцій не є єдиними подіями, коли потрібний запис адрес повернення у пам'ять. Лише для того, щоб навести кілька прикладів, можна згадати, що запис адрес повернення потрібен також у разі переривання процесу виконання програми (Interrupts) або у разі зміни завдань в експлуатаційних системах, призначених для виконання кількох завдань (системах Multi-Tasking)

Запис у пам'ять адрес повернення здійснюється зазвичай у так звані стеки або стекову пам'ять. Такий стек позначений на фіг 1 цифрою 1. Керування стеком 1 здійснюється від логічної схеми стеку 2. Логіка стеку 2, зокрема, генерує так званий укажчик стеку (Stack-Pointer), що вказує на ту окрему найближчу область стеку 1, в яку треба здійснити запис або з якої треба здійснити зчитування. Оскільки стеки та пов'язані з ними операції широко відомі, можна відмовитись від подальших пояснень

У стек 1 можна записувати не лише адреси повернення, але також будь-які інші дані (вміст регістрів, локальні змінні тощо). Проте, наведені пояснення стосуються, в основному, виключно операцій з адресами повернення

Якщо, наприклад, для команди LCALL, у стек 1 треба записати адресу повернення, цей процес ініціюється за допомогою логіки стеку 2

У системі, що розглядається, на відміну від звичайних систем, адреси повернення, що записані або мають бути записані у стек 1, додатково записуються до іншого запам'ятовувального пристрою як інформація захисту. Цей запам'ятовувальний пристрій, далі - запам'ятовувальний пристрій захисту, позначений на фіг 1 цифрою 3. Запам'ятовувальному пристрою захисту 3 підпорядкована логічна схема 4 запам'ятовувального пристрою захисту, за допомогою якої здійснюється керування цим запам'ятовувальним пристроєм захисту, аналогічно тому, як керування стеком 1 здійснюється за допомогою логіки стеку 2

Запам'ятовувальний пристрій захисту 3, на відміну від стеку 1, не можна примусити спрацьовувати ззовні, з-за меж системи, що виконує програму математичного забезпечення. Тобто, записаними у ньому даними не можна маніпулювати, або у будь-якому випадку не можна маніпулювати з припустимими витратами

Логіка 4 запам'ятовувального пристрою захисту у наведеному прикладі, на відміну від логіки стеку 2, спрацьовує лише у тому разі, якщо треба записати або зчитати адресу повернення. Проте, альтернативно є природним, що можна передбачити можливість спрацьовування логіки 4 запам'ятовувального пристрою захисту також у разі інших подій (крім процесів запису та зчитування із запуском ззовні)

Якщо процес виконання програми дійшов до моменту, коли слід здійснити перехід до записаної раніше адреси повернення, тобто, наприклад, у разі надходження команди RET, потрібну адресу повернення можна забезпечити шляхом відповідного зчитування стеку 1. Проте, перед застосуванням одержаних при цьому даних як адреси повернення здійснюється перевірка того, чи є ідентичними отримані дані і адреса повернення, записана у запам'ятовувальний пристрій захисту 3 як інформація захисту

Для цього передбачений компаратор 5, що приймає дані стеку 1 та запам'ятовувального пристрою захисту 3, які необхідно зіставити, і виконує цей процес порівняння

Якщо у результаті проведеного у компараторі 5 процесу порівняння виявиться, що зіставлені дані є ідентичними, це означає, що дані, одержані зі стеку 1, відповідають адресі повернення, яку треба записати спочатку, тобто, ними не маніпулювали, вони не стали неправильними внаслідок помилки в апаратному та/або математичному забезпеченні, та не були записані у невірне місце чи зчитані з невірного місця. Тому дані, записані у стек 1, можна вважати автентичними адресами повернення та використовувати їх як такі. Цей висновок можна зробити тому, що, як вже було згадано вище, цілеспрямований вплив на вміст запам'ятовувального пристрою захисту 3 практично неможливий

Якщо у процесі порівняння, проведеному у компараторі 5, виявиться, що зіставлені дані не є ідентичними, це означає, що дані, одержані зі стеку 1, з великою ймовірністю зазнали маніпулювання або стали неправильними внаслідок помилки в апаратному чи програмному забезпеченні, або були записані у невірне місце чи зчитані з невірного місця. Незалежно від причини відсутності ідентичності, записані у стек 1 дані не можна використовувати як адресу повернення, оскільки наслідком цього було б відхилення від приписаного виконання програми. Компаратор 5 у цьому разі генерує сигнал NMI, який подається на логіку NMI 6. Логічна схема NMI 6 ініціює негайне припинення процесу виконання програми та/або повернення у вихідний стан системи, що обробляє програму математичного забезпечення, та/або ініціює тривогу та/або стирання даних, що стосуються безпеки

Компаратор 5 у наведеному прикладі активізу-

ється командами, що, наприклад, як команда RET, мають своїм наслідком зчитування із стеку 1 даних, які слід інтерпретувати як адреси повернення. Решту часу компаратор 5 є пасивним.

За допомогою цих заходів можна досягти того, щоб програму математичного забезпечення, яка підлягає виконанню, можна було б виконувати лише у тому разі, якщо і поки не виявляються будь-які помилки в адресі повернення.

Приклад, що пояснюється за допомогою фіг 1, можна розглядати як варіант практичної реалізації способу захисту від використання адрес повернення, який спрацьовує у разі необхідності.

Хоча це не пояснюється більш докладно на прикладах, при цьому не є безумовно необхідним, щоб інформація захисту, підпорядкована відповідним адресам повернення, сама була адресою повернення. Як альтернативне рішення, можна передбачити, щоб тільки певні частини адреси повернення або дані, що будь-яким іншим чином репрезентують або характеризують адреси повернення, застосовувались як інформація захисту. У цьому разі компаратор 5 у разі необхідності слід було б, природно, замінити пристроєм для порівняння, зміненим у відповідності до конкретних обставин.

Альтернативним варіантом потрібного способу контролю процесу виконання програми є спосіб захисту від запису даних поверх адрес повернення, який пояснюється далі за допомогою фіг 2.

На фіг 2 показаний, зокрема, запам'ятовувальний пристрій 11, що складається з першої області пам'яті у вигляді стеку 11а, та другої області пам'яті у вигляді запам'ятовувального пристрою захисту 11b.

Стек 11а відповідає показаному на фіг 1 стеку 1. Запам'ятовувальний пристрій захисту 11b приблизно відповідає показаному на фіг 1 запам'ятовувальному пристрою захисту 3, проте, запам'ятовувальний пристрій захисту 11b описується іншою інформацією захисту на відміну від тієї, що використовується у запам'ятовувальному пристрої захисту 3.

Як і запам'ятовувальний пристрій захисту 3, запам'ятовувальний пристрій захисту 11b не може ззовні примусити спрацьовувати пристрій, що виконує програму математичного забезпечення, яку слід контролювати. Тобто, у цьому варіанті контролю процесу виконання програми інформацією, яка записана у запам'ятовувальному пристрої захисту, не можна маніпулювати таким чином, щоб витрати на це були припустимими.

Інформація, записана у запам'ятовувальний пристрій захисту 11b, складається з біту захисту від запису, якому приписане значення "1", якщо запис поверх даних у підпорядкованій області стеку не дозволений, або "0", якщо запис поверх даних у підпорядкованій області стеку дозволений.

Якщо, наприклад, як у разі надходження команди LCALL, треба записати адресу повернення у стек 11а, це ініціюється звичайним способом.

Одночасно в області запам'ятовувального пристрою захисту 11b, підпорядкованій області стеку, у якій була записана адреса повернення, записується "1", на ознаку того, що не можна здійснювати запис поверх даних у відповідній області

стеку.

Хай у наведеному прикладі стек 11а розділений на області, кожна з яких охоплює 8 біт відповідно, причому кожний з областей місткістю 8 біт підпорядкований біт захисту від запису, записаний у запам'ятовувальному пристрої захисту 11b. Якщо виходить з того, що робота ведеться з адресами довжиною 16 біт, для запису адреси повернення потрібні дві області стеку.

Як показано на фіг 2, адреса повернення записується у стек 11а. Адреса повернення складається з частини, що охоплює вісім старших біт (PCH), та частини, що охоплює 8 молодших біт (PCL). Областям стеку, як таким, що містять PCH, так і таким, що містять PCL, підпорядкована "1" як інформація захисту чи біт захисту від запису.

Відповідній інформації захисту чи біту захисту від запису знову приписується значення "0", якщо дані, записані у підпорядкованій області стеку, були використані як адреси повернення.

Перед кожною спробою запису даних до стеку (запису поверх записаних у ньому даних) інформація захисту (підпорядкований біт захисту від запису), підпорядковані області стеку, у яку треба здійснити запис, оцінюються з метою визначення, чи дозволений запис у стек на цьому місці.

Якщо підпорядкованій інформації захисту або підпорядкованому біту захисту від запису приписується значення "1", запис на це місце у стек не дозволений, спроба запису розпізнається як спроба маніпулювання або помилка у апаратному чи математичному забезпеченні. Навпаки, якщо підпорядкованій інформації захисту чи підпорядкованому біту захисту від запису приписується значення "0", запис у стек на цьому місці дозволений.

Рішення щодо припустимості запису у певну область стеку перевіряється за допомогою схеми перевірки наявності захисту від запису, яка у наведеному прикладі реалізована у вигляді ланки "TA" 12. На ланку "TA" 12 як вхідні сигнали подаються біт захисту від запису, підпорядкований області стеку, у яку треба здійснити запис, та сигнал Write_Stack (запис до стеку), який сигналізує про бажання здійснити запис, причому сигнал Write_Stack може набувати значення "1", якщо бажано здійснити запис, або "0", якщо запис не потрібний. Вихідний сигнал Valid_Write ланки "TA" 12 тоді свідчить про те, чи дозволений навмисний запис у відповідну область стеку (Valid_Write = "0"), чи не дозволений (Valid_Write = "1"). Вихідний сигнал Valid_Write ланки "TA" 12, як і вихідний сигнал NMI NMT компаратора 5 на фіг 1, можна використовувати для того, щоб негайно припинити процес виконання програми та/або перевести у вихідний стан систему, яка виконує програму, та/або ініціювати тривогу, та/або стерти дані, що стосуються безпеки.

Додатково до цього захисту від запису поверх адрес повернення у систему згідно з фіг 2 можна інтегрувати захист від застосування адрес повернення, модифікований у порівнянні з варіантом виконання згідно з фіг 1. Цей додатковий механізм захисту полягає у тому, що дані, зчитані зі стеку 11а, перед їхнім використанням як адрес повернення перевіряються щодо того, чи вони взагалі репрезентують адресу повернення. Цей висновок

можна зробити на підставі інформації захисту чи біту захисту від запису, підпорядкованих відповідним областям стеку. Лише у тому разі, якщо інформація захисту чи біт захисту від запису, підпорядковані області стеку, з якої слід здійснити зчитування, мають значення "1", дані, записані у відповідній області стеку, дійсно репрезентують адресу повернення. Тому умовою для цього є, природно, те, що даним, які репрезентують інформацію захисту чи біт захисту від запису виключно для адрес повернення, тобто, наприклад, додатково до команди CALL або аналогічним, записаним у стек 11а, приписується значення "1".

Цей додатковий механізм захисту згідно з фіг 2 реалізується за допомогою ланки "ТА" 13. У ланці "ТА" 13 як вхідні сигнали використовуються біт захисту від запису, підпорядкований області стеку, з якої слід здійснити зчитування, та сигнал Read_Stack, що сигналізує про мету використання даних, що зчитуються, причому сигнал Read_Stack може мати значення "1", якщо, наприклад, як у разі надходження команди RET, на меті є використання її як адреси повернення, або "0", якщо передбачається інший варіант її використання. Вихідний сигнал Valid_Read ланки "ТА" 13 у цьому разі свідчить про те, чи дозволяється використовувати дані, що запитуються, як адреси повернення (Valid_Read = "1"), або таке використання заборонено (Valid_Read = "0"). Якщо під час запиту щодо адрес повернення використання опитуваних зі стеку 11а даних як адрес повернення слід вважати неприпустимим, тому що Valid_Read = "0", це можна інтерпретувати як спробу маніпулювання або помилку в апаратному чи програмному забезпеченні, і використати як привід для введення у дію належних заходів для захисту. Ці заходи можуть полягати, зокрема, у негайному припиненні процесу виконання програми та/або переходу до вихідного стану системи, що опрацьовує програму, та/або ініціювання тривоги, та/або стирання даних,

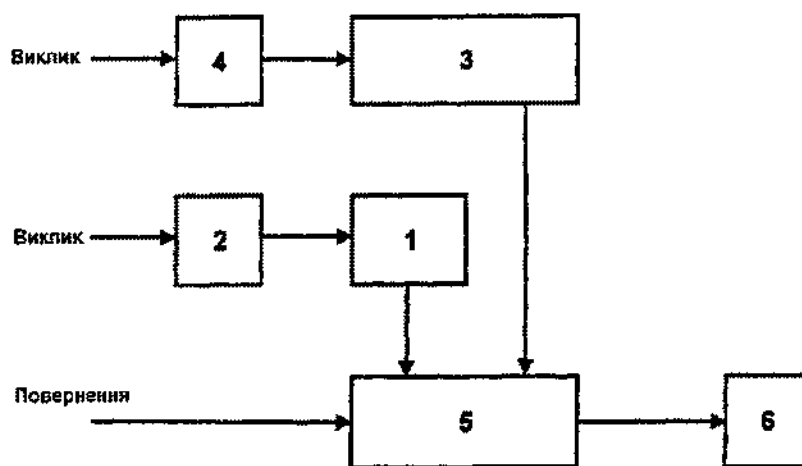
що стосуються безпеки.

З наведеного вище опису випливає, що інформація, яка має захист від доступу ззовні, складається з біту захисту від запису. Зрозуміло, що замість нього можна використовувати код, що складається з певної кількості біт з будь-якими значеннями, завдяки чому можна піддавати спеціальній обробці не тільки адреси повернення, але й будь-які інші дані, які треба захистити від маніпуляцій чи помилок.

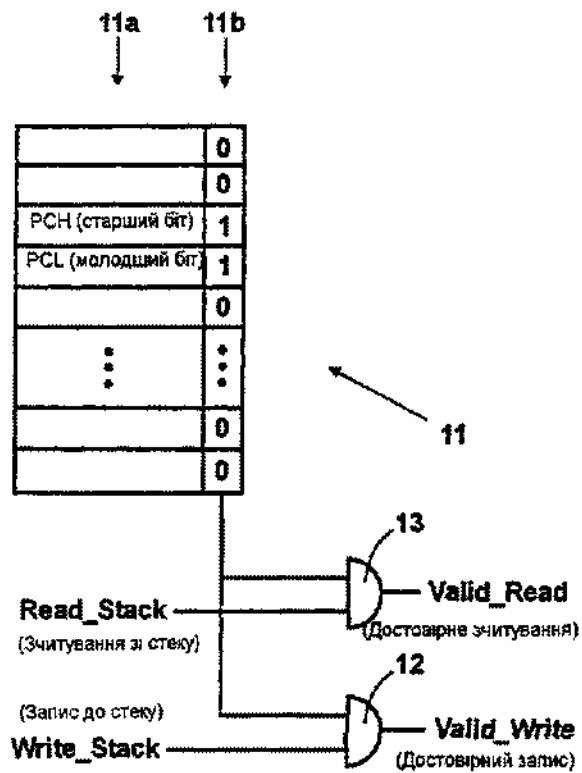
Запам'ятовувальний пристрій захисту, у якому записується різна інформація захисту, як вже неодноразово було сказано вище, представляє собою запам'ятовувальний пристрій, який ззовні не можна примусити спрацювати. Він розміщується переважно у пристрої, що виконує програму математичного забезпечення, яку слід захистити, тобто зазвичай мікропроцесор, мікроконтролер або сигнальний процесор, у цьому разі він особливо надійно захищений від доступу ззовні. Реалізація запам'ятовувального пристрою, який не можна примусити спрацювати ззовні, у мікропроцесорі, мікроконтролері або сигнальному процесорі (наприклад, у формі Hidden або Shadow Stack), є досить простою. Для цього необхідно лише відповідно модифікувати математичне забезпечення Kernel.

Застосування одного з наведених вище способів не потребує жодних додаткових змін в апаратному чи математичному забезпеченні. Зокрема, стек можна використовувати далі, як і раніше.

Таким чином, було розроблено спосіб для контролю приписаного виконання програм математичного забезпечення, завдяки чому просто і без змін математичного забезпечення, зокрема, дуже надійно можна запобігти цілеспрямованому маніпулюванню процесом виконання програм, а також частково виключити помилки в апаратному та математичному забезпеченні.



Фіг.1



Фіг.2