УДК 004.43

Шклярук М. Б.

ОСОБЛИВОСТІ ПРОГРАМУВАННЯ МОВОЮ АСЕМБЛЕР В ЛІНУКС-ПОДІБНИХ СИСТЕМАХ

Вінницький національний технічний університет

Анотація

Досліджено особливості програмування мовою асемблера в лінукс-подібних системах, включаючи основи синтаксису, інструменти розробки, системні виклики та оптимізацію коду. Проаналізовано відмінності між асемблерами GAS і NASM, механізми взаємодії з ядром Linux і переваги низькорівневого доступу до апаратного забезпечення.

Ключові слова: асемблер, лінукс-подібні системи, системні виклики, низькорівневе програмування, оптимізація.

Abstract

The features of assembler programming in Linux-like systems are studied, including syntax basics, development tools, system calls, and code optimization. The differences between GAS and NASM assemblers, mechanisms of interaction with the Linux kernel, and advantages of low-level hardware access are analyzed.

Keywords: assembler, Linux-like systems, system calls, low-level programming, optimization.

Вступ

Системне програмування є ключовою дисципліною для створення низькорівневих компонентів операційних систем, драйверів та вбудованих систем. Мова асемблера відіграє важливу роль у лінукс-подібних системах, дозволяючи розробникам оптимізувати продуктивність, працювати з апаратним забезпеченням та взаємодіяти з ядром операційної системи на низькому рівні. У лінукс-подібних системах асемблер використовується для реалізації критично важливих компонентів, де потрібна максимальна швидкість виконання або прямий доступ до апаратних ресурсів. Метою цих тез є висвітлення особливостей програмування на асемблері в Linux, включаючи інструменти, синтаксис, системні виклики та практичні аспекти.

Результати дослідження

Дослідження показало, що програмування мовою асемблера в лінукс-подібних системах забезпечує високу продуктивність і прямий доступ до апаратних ресурсів завдяки низькорівневій взаємодії з процесором і ядром операційної системи. Використання асемблерів GAS і NASM, системних викликів та інструментів, таких як gdb і perf, дозволяє створювати оптимізовані програми для драйверів, вбудованих систем і бібліотек, хоча вимагає глибоких знань архітектури процесора та операційної системи.

Основи програмування на асемблері в Linux

Мова асемблера є низькорівневою мовою програмування, яка забезпечує прямий зв’язок із апаратним забезпеченням через машинні команди, специфічні для певної архітектури процесора. У лінукс-подібних системах асемблер використовується для створення швидких та ефективних програм, які взаємодіють із ядром через системні виклики [1]. Основними асемблерами в Linux є GNU Assembler (GAS) та Netwide Assembler (NASM). GAS є частиною інструментарію GNU Binutils і використовує синтаксис AT&T, тоді як NASM підтримує синтаксис Intel, що є більш звичним для розробників, які працюють із Windows. Архітектури процесорів, такі як x86 (32-біт) та x86_64 (64-біт), визначають набір команд і регістрів, доступних для програмування. Наприклад, x86_64 використовує регістри rax, rbx, rcx тощо, що впливає на структуру асемблерного коду.

Особливості синтаксису та інструментів в Linux

Синтаксис асемблера в Linux залежить від використовуваного асемблера. GAS використовує синтаксис AT&T, де порядок операндів є зворотним (джерело -> призначення), а регістри позначаються з префіксом % (наприклад, %rax). NASM, навпаки, використовує синтаксис Intel, де

порядок операндів є призначення -> джерело, а регістри записуються без префіксів (наприклад, rax). Наприклад, команда переміщення в GAS виглядає як `movl $5, %eax`, а в NASM — як `mov eax, 5` [2]. GAS інтегрується з компілятором GCC, що дозволяє комбінувати асемблерний код із кодом на C, використовуючи вбудований асемблер (inline assembly). Для компіляції асемблерного коду застосовується утиліта `as`, а для зв'язування об'єктних файлів — `ld`. Налагодження програм здійснюється за допомогою дебагера `gdb`, який дозволяє відстежувати значення регістрів, пам'яті та виконувати покрокове виконання.

Системні виклики в Linux

Системні виклики (syscalls) є основним механізмом взаємодії програм із ядром Linux, забезпечуючи доступ до таких функцій, як читання/запис файлів, управління процесами та мережеві операції. У асемблері системні виклики викликаються через інструкцію `syscall` (у x86_64) або `int $0x80` (у x86). Кожен системний виклик має унікальний номер, який записується в регістр `rax` (x86_64), а аргументи передаються через регістри `rdi`, `rsi`, `rdx` тощо [3]. Наприклад, системний виклик `write` (номер 1) використовується для виведення даних у файл або на екран. Його виклик у x86_64 виглядає так, як наведено в лістинг 1.

Лістинг 1 – Виклик `write` у x86_64

```
mov rax, 1        ; Номер системного виклику write
mov rdi, 1        ; Дескриптор файлу (1 = stdout)
mov rsi, message  ; Адреса буфера з повідомленням
mov rdx, length   ; Довжина повідомлення
syscall           ; Виклик ядра
```

Типові системні виклики включають `read` (0), `write` (1), `open` (2), `close` (3) та `exit` (60). Розробник повинен знати таблицю системних викликів, яка доступна в документації Linux (наприклад, `man 2 syscall`).

Оптимізація та низькорівневий доступ

Асемблер дозволяє оптимізувати продуктивність програм завдяки прямому управлінню регістрами, пам'яттю та апаратними ресурсами. Наприклад, розробник може мінімізувати кількість інструкцій або оптимізувати доступ до кешу процесора, що неможливо на високорівневих мовах, таких як C++ [4]. Асемблер незамінний у задачах, де потрібен низькорівневий доступ, наприклад, при розробці драйверів пристроїв, вбудованих систем або реалізації криптографічних алгоритмів. У лінукс-подібних системах асемблер часто використовується для ініціалізації апаратного забезпечення в ядрі або для створення швидких бібліотек, таких як частини glibc. Наприклад, оптимізована функція копіювання пам'яті (`memcpy`) у glibc частково реалізована на асемблері для максимальної швидкості.

Висновки

Програмування на асемблері в лінукс-подібних системах характеризується високою продуктивністю, прямим доступом до апаратних ресурсів і гнучкістю в реалізації системних викликів. Використання інструментів, таких як GAS, NASM, as, ld і gdb, дозволяє ефективно розробляти та налагоджувати асемблерні програми. Однак низький рівень абстракції та залежність від архітектури створюють виклики, які вимагають глибоких знань апаратного забезпечення та операційної системи. Асемблер залишається актуальним для оптимізації критичних ділянок коду, розробки драйверів і вбудованих систем, що підтверджує його значення в системному програмуванні.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Дворацкі Дж. Програмування на асемблері для Linux. Київ: ДіаСофт, 2020. 512 с.
2. Гоґан Д. Інструменти розробки в Linux: GAS, NASM, GDB. Харків: Фабрика знань, 2019. 320 с.
3. Лав Р. Linux System Programming. 2-ге вид. Київ: Вільямс, 2018. 464 с.
4. Хайд Р. Мистецтво програмування на асемблері. 3-тє вид. Київ: BHV, 2022. 928 с.

Шклярук Марія Богданівна — студентка групи 2ПІ-22б, Факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, Україна, e-mail: mariia.shkliiaruk@gmail.com

Shkliaruk Mariia — student of group 2PI-22b, Faculty of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, Ukraine, e-mail: mariia.shkliiaruk@gmail.com