УДК 004.432.2

МОВА ПРОГРАМУВАННЯ АСЕМБЛЕР

Мокан Дмитро Володимирович,
здобувач вищої освіти спеціальності 122 «Комп'ютерні науки»
Миколаївський національний аграрний університет
м. Миколаїв, Україна

Анотація: *Асемблер є низькорівневою мовою програмування, яка безпосередньо взаємодіє з процесором комп'ютера. Він відіграє важливу роль у розробці системного програмного забезпечення, драйверів пристроїв та оптимізації критичних до продуктивності компонентів програм. Ця доповідь охоплює історію розвитку асемблера, його принципові особливості та відмінності від інших мов програмування, а також надає приклади коду з детальними поясненнями. Метою є надання глибокого розуміння цієї потужної мови низького рівня та її застосування в сучасній розробці програмного забезпечення.*

Ключові слова: *асемблер, низькорівнева мова програмування, машинний код, процесор, оптимізація, системне програмне забезпечення.*

Асемблер є однією з найдавніших і найпотужніших мов програмування, яка дозволяє безпосередньо взаємодіяти з апаратним забезпеченням комп'ютера на найнижчому рівні. Протягом десятиліть асемблер був основною мовою програмування для багатьох комп'ютерних систем, оскільки він забезпечував максимальний контроль над апаратним забезпеченням і оптимізацію продуктивності. Однак з розвитком високорівневих мов програмування, таких як C, C++ та Java, асемблер поступово відійшов на другий план для більшості загальних програмістських завдань, але все ще продовжує відігравати важливу роль у розробці системного програмного забезпечення, драйверів пристроїв та оптимізації критичних до продуктивності компонентів програм. Він також широко використовується в галузях, де необхідна максимальна продуктивність і безпосередній контроль над апаратним забезпеченням, наприклад, у вбудованих системах, іграх та мультимедійних додатках.

Асемблер є низькорівневою мовою програмування, що означає, що він безпосередньо взаємодіє з процесором комп'ютера на рівні машинного коду. Це надає йому кілька ключових переваг та особливостей:

1. Близькість до апаратного забезпечення: асемблер дозволяє програмістам безпосередньо маніпулювати регістрами, пам'яттю та іншими апаратними компонентами комп'ютера. Це забезпечує максимальний контроль і оптимізацію продуктивності, але вимагає глибокого розуміння архітектури процесора.
2. Висока продуктивність: оскільки асемблер безпосередньо перекладається в машинний код, він забезпечує найвищу можливу продуктивність у порівнянні з

високорівневими мовами програмування, які потребують додаткового шару абстракції та компіляції.

3. Мнемонічні коди: асемблер використовує мнемонічні коди, які є скороченнями для команд процесора. Це полегшує читання та написання коду в порівнянні з безпосереднім використанням машинного коду.

4. Залежність від архітектури: асемблер тісно пов'язаний з архітектурою процесора, для якого він створений. Це означає, що код асемблера, написаний для однієї архітектури, не буде працювати на іншій без перекладу.

5. Складність: асемблер вважається більш складним для вивчення та використання, ніж високорівневі мови програмування, оскільки він вимагає глибокого розуміння архітектури процесора та низькорівневих деталей.

На відміну від високорівневих мов програмування, таких як C, C++, Java або Python, асемблер не має вбудованих абстракцій та структур даних високого рівня. Натомість він оперує безпосередньо на рівні машинних інструкцій, регістрів і пам'яті. Це робить його більш складним для розробки великих і складних програм, але надзвичайно потужним для оптимізації та взаємодії з апаратним забезпеченням на найнижчому рівні.

Розглянемо декілька прикладів коду на асемблері для процесора x86 з детальними поясненнями:

1. Додавання двох чисел:

```
section .text
    global _start
_start:
    ; Завантажити значення в регістри
    mov eax, 5       ; eax = 5
    mov ebx, 10      ; ebx = 10
    ; Додати значення та зберегти результат
    add eax, ebx     ; eax = eax + ebx (eax = 15)
    ; Завершити програму
    mov eax, 1       ; sys_exit (1) для виходу
    xor ebx, ebx     ; вихідний код 0
    int 0x80         ; викликати системну функцію
```

У цьому прикладі ми завантажуємо значення 5 і 10 у регістри `eax` та `ebx` відповідно. Потім використовуємо інструкцію `add` для додавання значень цих регістрів, і результат зберігається в `eax`. Нарешті, ми виконуємо системний виклик для завершення програми.

2. Обчислення факторіалу числа:

```
section .text
    global _start
_start:
    ; Завантажити значення в регістри
    mov eax, 5       ; eax = 5 (число, факторіал якого потрібно обчислити)
    mov ebx, 1       ; ebx = 1 (акумулятор для факторіалу)
loop:
    ; Обчислити факторіал
```

```
mul ebx         ; eax = eax * ebx
dec eax         ; eax = eax - 1
jnz loop        ; якщо eax != 0, перейти до loop
; Результат факторіалу зберігається в ebx
mov eax, 1      ; sys_exit (1) для виходу
int 0x80        ; викликати системну функцію
```

У цьому прикладі ми завантажуємо число 5 у регістр `eax` і використовуємо регістр `ebx` як акумулятор для обчислення факторіалу. Ми використовуємо цикл `loop`, який множить `eax` на `ebx` (початкове значення `ebx` дорівнює 1), а потім зменшує `eax` на 1. Цикл продовжується, доки `eax` не стане рівним 0. Після виходу з циклу, результат факторіалу зберігається в `ebx`.

Ці приклади демонструють, як асемблер працює безпосередньо з регістрами та машинними інструкціями для виконання обчислень та управління потоком виконання програми. Незважаючи на свою простоту, асемблер залишається потужним інструментом для оптимізації та розробки системного програмного забезпечення.

Асемблер є унікальною низькорівневою мовою програмування, яка забезпечує безпосередню взаємодію з апаратним забезпеченням комп'ютера. Незважаючи на появу багатьох високорівневих мов, асемблер продовжує відігравати важливу роль у розробці системного програмного забезпечення, драйверів пристроїв та оптимізації критичних до продуктивності компонентів програм.

Список використаних джерел:

1. "Low Level Programming" by Jonathan Blow
https://en.wikipedia.org/wiki/Low-level_programming_language
2. "Real World Assembly Language" by Randall Hyde
https://hackaday.com/2020/08/25/assembly-language-for-real/

Abstract: *Assembler is a low-level programming language that directly interacts with the computer processor. It plays an important role in the development of system software, device drivers, and optimization of performance-critical application components. This talk covers the history of assembly language development, its fundamental features and differences from other programming languages, and provides code examples with detailed explanations. The goal is to provide a deep understanding of this powerful low-level language and its applications in modern software development.*

Keywords: *assembler, low-level programming language, machine code, processor, optimization, system software.*

Науковий керівник: Тищенко С.І.,
к.п.н., доцент,
доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій,
Миколаївський національний аграрний університет
м. Миколаїв, Україна