15.4 БУФЕРИЗАЦІЯ ОПЕРАЦІЙ ВВЕДЕННЯ-ВИВЕДЕННЯ

Припустимо, що процесу користувача необхідно виконати зчитування блоків даних завдовжки 512 байт кожний по одному з гнучкого диска. Дані будуть розміщені в область всередині адресного простору процесу користувача з віртуальною адресою від 1000 до 1511. Найпростіший шлях розв'язання цієї задачі – виконати команду введення-виведення і очікувати дані.

При генерації процесом команди введення-виведення він призупиняється і вивантажується на диск до початку її виконання. Далі процес чекає, коли буде виконана запрошена ним операція введення-виведення, яка, усвою чергу, чекає, коли процес буде повернений в основну пам'ять, оскільки місце в основній пам'яті для зчитування даних просто відсутнє. Для того щоб уникнути взаємоблокування, призначена для користувача пам'ять в операціях введення- виведення має бути заблокована в основній пам'яті відразу ж після видачі запиту на введення-виведення.

Те ж міркування застосовне і до операції виведення. Щоб зменшити накладні витрати і збільшити ефективність, іноді зручно виконувати читання даних заздалегідь, до реального запиту, а запис даних – трохи пізніше за реальний запит. Ця методика відома як ***буферизація.*** Розглянемо деякі схеми буферизації, підтримувані ОС для підвищення продуктивності (рис. 15.5).

Введення без буферу. При небуферованому введенні (рис. 15.5, *a)* пересилка здійснюється посимвольно. Після кожного символу відбувається переривання.

Одинарний буфер. Простим типом підтримки з боку ОС є одинарний буфер. У той момент, коли процес користувача виконує запит уведення- виведення, ОС призначає йому буфер в системній частині основної пам'яті (рис. 15.5, *b*).

Схема одинарного буфера для блочно-орієнтованих пристроїв може бути описана таким чином.

Спочатку здійснюється передача вхідних даних у системний буфер. Коли вона завершується, процес переміщає блок у простір користувача і негайно робить запит наступного блоку. Така процедура називається ***випереджаючим прочитанням,*** або ***застережливим введенням***. Вона виконується в припущенні, що цей блок з часом знадобиться.

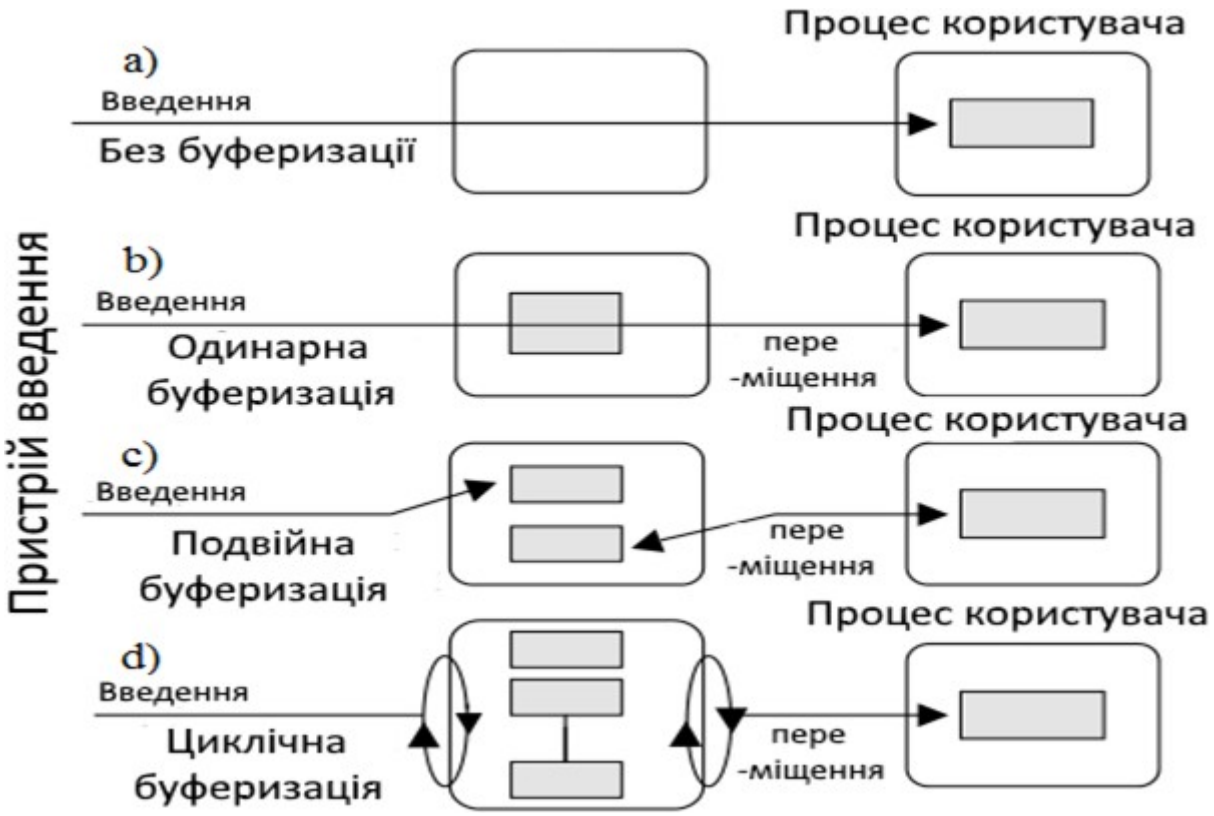

Рисунок 15.5 – Схеми буферизації введення-виведення

Такий підхід, у порівнянні з відсутністю буферизації, забезпечує підвищення швидкодії. Призначений для користувача процес може обробляти один блок даних в той час, коли відбувається прочитання наступного блоку. ОС при цьому може здійснювати вивантаження процесу, оскільки виконується операція зчитування даних в системну область, а не в пам'ять процесу користувача.

Проте така технологія ускладнює функціонування ОС, яка повинна стежити за виділенням системних буферів. Впливає буферизація і на схему підкачування. Коли операція введення-виведення працює з тим же диском, який використовується і для свопінгу, втрачається сенс в організації черги операцій запису. Вивантаження процесу і звільнення основної пам'яті не почнеться до тих пір, поки не завершиться запрошена операція введення-виведення – а тоді вивантаження процесу не матиме сенсу.

Подвійна буферизація в ядрі. Удосконалити схему одинарної буферизації можна шляхом використання двох системних буферів (рис. 15.4, *c)*. Тепер процес виконує передачу даних в один буфер (чи зчитування з нього), тоді як ОС звільняє (чи заповнює) інший. Ця технологія відома як ***подвійна буферизація*** або ***змінний буфер***.

Циклічний буфер. Схема подвійного буфера покликана вирівняти потік даних між пристроями введення-виведення і процесом. Якщо нас цікавить продуктивність деякого процесу, то в першу чергу вимагається, щоб операції введення-виведення не гальмували його роботи. Подвійна буферизація може виявитися недостатньою, якщо процес часто виконує введення або виведення. Частенько в такому разі розв'язати проблему допомагає нарощування буферів.

При використанні великої кількості буферів, яка складається більше ніж з двох, схема іменується ***циклічною буферизацією*** (рис. 15.4, *d*).

Проте при такій роботі запис або читання великої кількості інформації з адресного простору введення-виведення призводять до великої кількості операцій введення-виведення, які повинен виконувати процесор. Для звільнення процесора від операцій послідовного виведення даних з оперативної пам'яті або послідовного введення в неї був запропонований механізм прямого доступу зовнішніх пристроїв до пам'яті – ПДП або Direct Memory Access – **DMA**.

Контролер містить декілька регістрів: регістр адреси пам'яті, лічильник байтів і управляючі регістри (порт введення-виведення, читання або запис).

Для того щоб будь-який пристрій, окрім процесора, міг записати інформацію в пам'ять або прочитати її з пам'яті, необхідно, щоб цей пристрій міг забрати у процесора управління локальною магістраллю для виставляння відповідних сигналів на шини адреси, даних і управління. Для централізації ці обов'язки покладаються не на кожен пристрій окремо, а на спеціальний контролер – контролер прямого доступу до пам'яті.

Контролер прямого доступу до пам'яті має декілька спарених ліній – каналів DMA, які можуть підключатися до різних пристроїв. Перед початком використання прямого доступу до пам'яті цей контролер необхідно **запрограмувати**, записавши в його порти інформацію про те, який канал або канали передбачається задіяти, які операції вони здійснюватимуть, яка адреса пам'яті є початковою для передачі інформації і яка кількість інформації має бути передана.

Отримавши по одній з ліній (каналів DMA), сигнал запиту на передачу даних від зовнішнього пристрою, контролер по шині управління повідомляє процесор про бажання взяти на себе управління локальною магістраллю. Процесор, можливо, через деякий час, необхідний для завершення його дій з магістраллю, передає управління

нею контролеру DMA, сповістивши його спеціальним сигналом. Контролер DMA виставляє на адресну шину адреси пам'яті для передачі чергової порції інформації і другою лінією каналу прямого доступу до пам'яті повідомляє пристрій про готовність магістралі до передачі даних.

Після цього, використовуючи шину даних і шину управління, контролер DMA, пристрій введення-виведення і пам'ять здійснюють процес обміну інформацією. Потім контролер прямого доступу до пам'яті сповіщає процесор про свою відмову від управління магістраллю, і той берет керівні функції на себе. При передачі великої кількості даних увесь процес повторюється циклічнно.

Роботу з контролером DMA можна простежити за схемою, показаною на рис.15.6:

1. Процесор програмує контролер (які дані і куди перемістити).
2. Процесор дає команду дисковому контролеру прочитати дані в буфер.
3. Зчитуються дані в буфер, контролер диска перевіряє їх контрольну суму.
4. Контролер DMA посилає запит на читання дисковому контролеру.
5. Контролер диску поставляє дані на шину, адрес пам'яті вже знаходиться на шині, відбувається запис даних в пам'ять.
6. Коли запис закінчений, контролер диска посилає підтвердження DMA контролеру.
7. DMA контролер збільшує використовувану адресу і зменшує значення лічильника байтів.
8. Все повторюється з пункту 4, поки значення лічильника не стане рівним нулю.
9. Контролер DMA ініціює переривання.

При прямому доступі до пам'яті процесор і контролер DMA по черзі управляють локальною магістраллю. Це, звичайно, дещо знижує продуктивність процесора, оскільки при виконанні деяких команд або при читанні чергової порції команд у внутрішній кеш він повинен чекати звільнення магістралі, але в цілому продуктивність обчислювальної системи істотно росте.

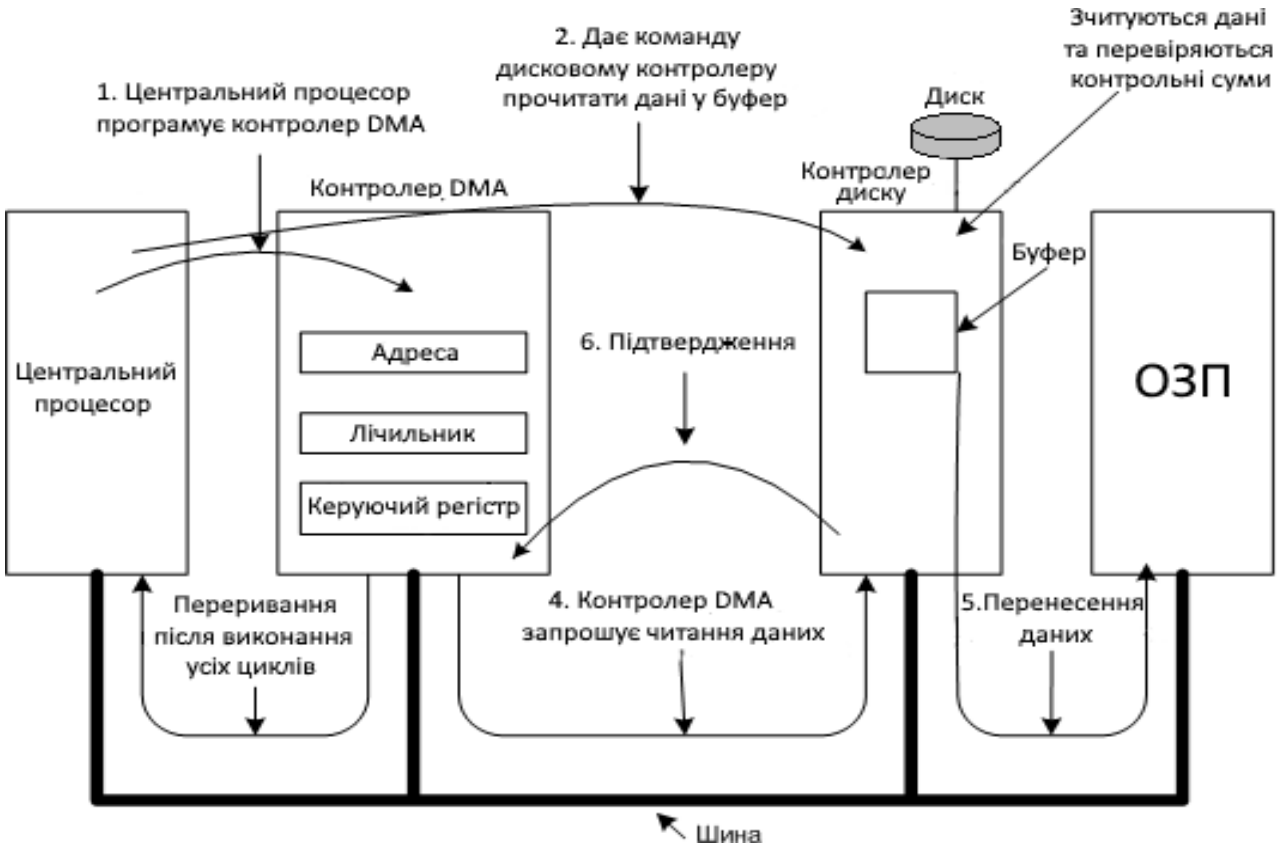

Рисунок 15.6 – Робота DMA-контролера

При підключенні до системи нового пристрою, який уміє використовувати прямий доступ до пам'яті, необхідно програмно або апаратно задати номер каналу DMA, до якого буде приписано пристрій. На відміну від переривань, де один номер переривання міг відповідати декільком пристроям, канали DMA завжди перебувають у монопольному володінні пристроями.