ПРАКТИЧНЕ ЗАНЯТТЯ 6
ВИВЧЕННЯ ЗАСОБІВ ОРГАНІЗАЦІЇ ВИВОДУ ІНФОРМАЦІЇ НА ЕКРАН МОВОЮ АСЕМБЛЕРА

6.1 Мета заняття

Ознайомитися з засобами організації виводу інформації на екран мовою Асемблера.

6.2 Теоретичні положення

Існує багато засобів організації виводу інформації на екран. Розглянемо на прикладі два з них.

1. За допомогою функції MessageBox.

Функція MessageBox

Функція MessageBox створює, показує на екрані і використовує вікно повідомлення. Вікно повідомлення містить визначне програмою повідомлення, заголовок і будь-яку комбінацію значків і командних кнопок.

Параметри

hWnd Дескриптор вікна власника, яке створює вікно повідомлення. Якщо цей параметр – ПУСТО (NULL), вікно повідомлення не має вікна власника.

lpText Покажчик на символьний рядок з нулем у кінці, який містить текст повідомлення.

lpCaption Покажчик на символьний рядок з нулем у кінці, який містить заголовок діалогового вікна (вікна повідомлення). Якщо цей параметр – ПУСТО (NULL), використовується заданий за замовчуванням заголовок Error (Помилка).

uType Встановлює зміст і режим роботи діалогового вікна. Цим параметром може бути комбінація прапорців з нижче перерахованих груп.

Щоб позначити кнопки, які відображаються на екрані у вікні повідомлення, задайте одне з нижче перерахованих значень.

Значення	Призначення
MB_ABORTRETRYIGNORE	Вікно повідомлення містить три командних кнопки: Припинити (Abort), Повторити (Retry) і Пропустити (Ignore).
MB_CANCELTRYCONTINUE	Microsoft® Windows® 2000 / XP: Вікно повідомлення містить три командних кнопки: Скасувати (Cancel), Спробувати знову (Try Again), Продовжити (Continue). Використовуйте цей тип вікна повідомлення замість типу MB_ABORTRETRYIGNORE.
MB_HELP	Windows 95/98 / Me, Windows NT® 4.0 і вище: Додає до вікна повідомлення кнопку Довідка (Help). Коли користувач натискає на кнопку Довідка (Help) або натискає кнопку F1, система відправляє власникові повідомлення WM_HELP.

MB_OK	Вікно повідомлення містить одну командну кнопку: OK. Це – значення за замовчуванням.
MB_OKCANCEL	Вікно повідомлення містить дві командних кнопки: OK і Скасувати (Cancel).
MB_RETRYCANCEL	Вікно повідомлення містить дві командних кнопки: Повторити (Retry) і Скасувати (Cancel).
MB_YESNO	Вікно повідомлення містить дві командних кнопки: Так (Yes) і Ні (No).
MB_YESNOCANCEL	Вікно повідомлення містить три командних кнопки: Так (Yes), Ні (No) і Скасувати (Cancel).

Щоб показати на екрані значок у вікні повідомлення, встановіть одне з нижче перерахованих значень.

Значення	Призначення
MB_ICONEXCLAMATION	У вікні повідомлення з'являється іконка знака оклику.
MB_ICONWARNING	У вікні повідомлення з'являється іконка знака оклику.
MB_ICONINFORMATION	У вікні повідомлення з'являється значок мала літера і в колі.
MB_ICONASTERISK	У вікні повідомлення з'являється значок мала літера і в колі.
MB_ICONQUESTION	У вікні повідомлення з'являється іконка знака питання.
MB_ICONSTOP	У вікні повідомлення з'являється значок стоп-сигналу
MB_ICONERROR	У вікні повідомлення з'являється значок стоп-сигналу.
MB_ICONHAND	У вікні повідомлення з'являється значок стоп-сигналу.

Щоб вказати основну кнопку (за замовчуванням), встановіть одне з нижче перерахованих значень.

Значення	Призначення
MB_DEFBUTTON1	Перша кнопка – основна (кнопка за замовчуванням). MB_DEFBUTTON1 – значення за замовчуванням, якщо MB_DEFBUTTON2, MB_DEFBUTTON3, або MB_DEFBUTTON4 не визначені.
MB_DEFBUTTON2	Друга кнопка – основна кнопка.
MB_DEFBUTTON3	Третя кнопка – основна кнопка.
MB_DEFBUTTON4	Четверта кнопка – основна кнопка.

Щоб вказати модальність діалогового вікна, встановіть одне з нижче перерахованих значень.

Значення	Призначення

MB_APPLMODAL	Користувач повинен відповісти вікну повідомлення перед продовженням роботи у вікні, ідентифікованому параметром hWnd. При цьому, користувач може переміщатися у вікна інших потоків і працювати в цих вікнах. Залежно від ієрархії вікон в додатку, користувач може переміщатися в інші вікна у межах потоку. Всі дочірні вікна батьківського вікна повідомлення автоматично блокуються, спливаючі вікна не блокуються. MB_APPLMODAL є значенням за замовчуванням, якщо MB_SYSTEMMODAL або MB_TASKMODAL не визначені.
MB_SYSTEMMODAL	Те саме, що і MB_APPLMODAL за винятком того, що вікно повідомлення має стиль WS_EX_TOPMOST. Використовуйте для вікон, які працюють в системному режимі (недоступному для користувача), щоб повідомити користувача про серйозні, потенційно руйнівні помилки, які вимагають негайної уваги (наприклад, вихід за межі обсягу пам'яті). Цей прапорець не має ніякого впливу на здатність користувача взаємодіяти з іншими вікнами, не тільки пов'язаними з hWnd.
MB_TASKMODAL	Те саме, що і MB_APPLMODAL за винятком того, що всі вікна верхнього рівня, які належать поточному потоку блокуються, якщо параметр hWnd дорівнює ПУСТО (NULL). Використовуйте цей прапорець тоді, коли програма виклику або бібліотека не мають доступного дескриптора вікна, але тим не менше повинні перешкодити введенню даних в інші вікна в потоці виклику, не припиняючи роботу інших потоків.

Щоб встановлювати інші параметри, використовуйте одне або кілька нижченаведених значень.

Значення	Призначення
MB_DEFAULT_DESKTOP_ONLY	Windows NT / 2000 / XP: Те ж саме, що і MB_SERVICE_NOTIFICATION за винятком того, що система показує на екрані вікно повідомлення тільки на заданому за замовчуванням робочому столі взаємодіє віконної станції. Windows NT 4.0 і раніше: Якщо поточний робочий стіл введення

	даних – не заданий за замовчуванням, MessageBox завершується помилкою. Windows 2000 / XP: Якщо поточний робочий стіл введення даних – не задані за замовчуванням, MessageBox не повертає значення до тих пір, поки користувач не переключиться на заданий за замовчуванням робочий стіл.
MB_RIGHT	Текст вирівнюється по правому краю.
MB_RTLREADING	Виведений на екран текст повідомлення і заголовка, використовує порядок читання справа наліво як в системах на єврейському і арабських мовах.
MB_SETFOREGROUND	Вікно повідомлення стає високопріоритетним вікном. Внутрішньо, система викликає функцію SetForegroundWindow для вікна повідомлення.
MB_TOPMOST	Вікно повідомлення створюється зі стилем вікна WS_EX_TOPMOST.
MB_SERVICE_NOTIFICATION	Windows NT / 2000 / XP: Програма виклику – сервісний модуль, який повідомляє про подію користувача. Функція показує на екрані вікно повідомлення на поточному активному робочому столі, навіть якщо немає користувача, який розпочав роботу на комп'ютері. Термінальні служби: Якщо потік виклику має маркер запозичення прав, функція спрямовує вікно повідомлення в сесію, задану у маркері запозичення прав. Якщо цей прапорець встановлений, параметр hWnd повинен мати значення ПУСТО (NULL). Це робиться для того, щоб вікно повідомлення можна було показати на робочому столі, а не робочий стіл, відповідний hWnd.
MB_SERVICE_NOTIFICATION_NT3X	Windows NT / 2000 / XP: Це

	значення відповідає значенню, яке визначається для MB_SERVICE_NOTIFICATION в Windows NT версії 3.51.

Значення, яке повертається

Якщо вікно повідомлення має кнопку Відмінити (Cancel), то функція повертає значення IDCANCEL, у випадку обробки клавіші ESC, або обрання кнопки Скасувати (Cancel). Якщо вікно повідомлення не має кнопки Скасувати (Cancel), натискання ESC не має ніякої дії.

Якщо функція завершується помилкою, повернуте значення дорівнює нулю. Щоб отримати додаткову інформацію про помилку, викличте GetLastError.

Якщо функція завершується успішно, повертається одне з нижче перерахованих значень пункту меню.

Значення	Призначення
IDABORT	Була обрана кнопка Припинити (Abort).
IDCANCEL	Була обрана кнопка Скасувати (Cancel).
IDCONTINUE	Була обрана кнопка Продовжити (Continue).
IDIGNORE	Була обрана кнопка Пропустити (Ignore).
IDNO	Була обрана кнопка Ні (No).
IDOK	Була обрана кнопка OK.
IDRETRY	Була обрана кнопка Поторій (Retry).
IDTRYAGAIN	Була обрана кнопка Спробувати знову (Try Again).
IDYES	Була обрана кнопка Так (Yes).

Зауваження

Коли Ви використовуєте вікно повідомлення, яке працює в системному (недоступному) режимі, і звертає увагу на те, що в системі мало пам'яті, рядки, на які вказують параметри lpText і lpCaption, не повинні братися з файлу ресурсу, тому що спроба завантажити ресурс може завершитися помилкою.

Якщо Ви створюєте вікно повідомлення, в той час, коли присутне діалогове вікно, використовуйте дескриптор блоку діалогу як параметр hWnd. Параметр hWnd не повинен ідентифікувати дочірнє вікно, таке як орган управління в блоці діалогу.

Для використання функції MessageBox необхідно додати бібліотеку User32.lib.

Додати необхідну бібліотеку у середовищі MASM можна за допомогою директиви INCLUDE та INCLUDELIB.

Синтаксис:

INCLUDE *filename* або INCLUDE "*filename*"

INCLUDELIB *filename* або INCLUDELIB "*filename*"

INCLUDE Додає вихідний код з файлу з зазначеним ім'ям у поточній позиції модуля під час асемблювання.

Назва файлу повинна бути у кутових дужках, якщо вона містить зворотний слэш, крапку з комою, знак більше або менш ніж символ, одиночні або подвійні лапки.

INCLUDELIB Призводить до того, що під час компонування додається бібліотечний файл із зазначеним ім'ям. Якщо розширення імені файлу не вказується, то передбачається розширення .LIB.

Назва файлу повинна бути у кутових дужках, якщо вона містить зворотний слэш, крапку з комою, знак більше або менш ніж символ, одиночні або подвійні лапки.

Приклад програми:

```
.....
.data
MsgBoxCaption  db "My first program",0
MsgBoxText     db "Hello World!",0
  ……..
.code
start
invoke MessageBox, NULL, addr MsgBoxText, addr MsgBoxCaption, MB_OK
invoke ExitProcess, NULL
end start
```

INVOKE Директива спрощеного виклику процедури у MASM. Синтаксис: invoke *вислів* [, *аргументи*]

Висловом може бути ім'я функції або покажчик на функцію. Аргументи функції відокремлюються комою та можуть бути її параметрами, регістрами, змінними, адресами тощо.

Оператор **addr** використаний для передачі адреси мітки (і не тільки) функції. Він дійсний тільки в контексті директиви **invoke**. Його не можна використовувати, щоб присвоїти адресу мітки регістрів або змінної. У наведеному прикладі можна використати **offset** замість **addr**. Проте, між ними є відмінності.

1. **addr** не може бути використаний з мітками, визначеними попереду, а **offset** може. Наприклад, якщо мітка визначена у коді далі, ніж рядок з **invoke**, addr НЕ буде працювати.

2. **addr** підтримує локальні змінні, тоді як offset – ні. Локальна змінна – це лише зарезервоване місце у стеку. Відома тільки його адреса під час виконання програмі. **Offset** інтерпретується під час компіляції асемблером, тому не дивно, що він не підтримує локальні змінні. **Addr** працює з ними, тому що асемблер спочатку перевіряє – глобальна змінна чи локальна. Якщо вона глобальна, він поміщає адресу цієї змінної до об'єктного файлу. У цьому випадку оператор працює як **offset**.

2. За допомогою консолі.

Приклад програми:

```
.....
.data
HelloWorld db "Hello World!", 0
……..
.code
start

invoke crt_puts, ADDR HelloWorld
call    crt__getch
invoke ExitProcess, 0
end start
```

CALL Директива, яка викликає процедуру методу. Синтаксис:
CALL <покажчик_екземпляру> METHOD [<ім’я_об'єкта>:] < ім’я_метода> {USES [{сегм_регістр:} регістр_зсуву} {<Параметри_росширеного_виклику}

Бібліотека Microsoft Visual C Run-Time (MSVCRT.DLL) експортує понад 700 функцій, які спрощують багато стандартних завдань, таких як: форматування консольного виведення, робота з динамічною пам'яттю, маніпуляція текстових рядків, сортування та ін.

Щоб викликати функцію CRT в MASM32 з використанням invoke, необхідно попередньо вказати прототип msvcrt.inc, який містить набір прототипів для CRT.

Функції CRT можуть бути об'явлені, як покажчики на функції замість «звичайних» функцій. Щоб уникнути конфліктів імен с іншими функціями та зарезервованими словами MASM, імена функцій мають префікс **crt_**.

Функція **puts** пише у потік стандартного виводу (stdout) рядок, заміняючи нуль, який цей рядок завершує, на символ нового рядка (0Ah). Таким чином, для правильного відображення рядки, додані до структури, повинні завершуватися символом з кодом 0. У іншому випадку необхідно користуватися функцією, яка приймає покажчик на рядок і його довжину.

Функція **getch** читає окремі символи з консолі та повертає прочитаний символ.

Для запуску цього коду необхідно додати бібліотеку msvcrt.dll.

Це бібліотека динамічних посилань, яка запускається автоматично при запуску програм і містить всі основні функції і методи, типові для мов Сі і Сі ++, такі як printf, memcpy, cos та ін. Вона є частиною бібліотеки Microsoft C Runtime.

2.1 Ініціація та резервування даних

Під час написання коду програми необхідно чітко розрізняти, коли операції проводяться над символьними даними, а коли над числовими. Так,

наприклад, рядок з нульовим завершителем "1472" може бути заданий у такий спосіб:

```
Sdb     49,52,55,50,0
```

або

```
Sdb     "1472",0
```

і буде займати у пам'яті 5 байт. У той час як число +1472 займатиме у пам'яті 2 байти і може бути задане як:

```
Sdw     1472
```

Для виведення на екран числових даних їх спочатку необхідно подати у вигляді рядка.

Перетворення числа на рядок здійснюється функціями бібліотеки masm32.lib.

Директиви резервування та ініціалізації даних простих типів:

db - резервування пам'яті для даних розміром 1 байт.

dw - резервування пам'яті для даних розміром 2 байти.

dd - резервування пам'яті для даних розміром 4 байта.

df - резервування пам'яті для даних розміром 6 байт;

dp - резервування пам'яті для даних розміром 6 байт.

dq - резервування пам'яті для даних розміром 8 байт.

dt - резервування пам'яті для даних розміром 10 байт.

Приклади форматів резервування та ініціалізації даних:

1) Perem1 db 100 ; статичні дані
2) Massiv dw 0,0,0,0,0 ; послідовність статичних даних
3) Message db “Hello World!”,0 ; рядок
4) Perem1 db ? ; дані не ініційовані
5) Для повторів операндів можна використати директиву DUP (duplicate)

Massiv1 dw 5 DUP (0) ; масив з п’яти 0
Massiv2 db 10 DUP(0),10 DUP (1) ;масив з 10 «0» і 10 «1»
Massiv3 db 3 DUP(0),1,1 ; масив з трьох «0» і двох «1»
Massiv4 db 100 DUP(?) ; масив з 100 невизначених значень
Stroka db 10 DUP(‘ ‘) ; рядок з 10 пробілів

2.2 Операції, які використовують у макровизначеннях

Оператор виразу % – оператор, який розглядає значення виразу в аргументі макроса як текст.

Цей оператор призначений для виконання заміщення текстових макросів і перетворення значення константних виразів в їх текстове подання. Подібне перетворення може виконуватися декількома способами.

При використанні в директиві TEXTQ оператор % наказує препроцесору обчислити значення константного виразу і перетворити отриманий цілочисельний результат на еквівалентне текстове значення.

Якщо в якості параметра макрокоманди має використовуватися цілочисельне значення, за допомогою оператора (%) в макрокоманду можна передати значення цілочисельного виразу. При цьому спочатку виконується

обчислення значення виразу, потім воно перетворюється на текстовий вигляд і передається в якості параметра до макрокоманди. Всі елементи виразу повинні бути відомі на етапі трансляції.

Використання оператора (%) на початку рядка. Якщо оператор виразу (%) розташовується у першій позиції рядка вихідного коду, це наказує препроцесорові замінити у поточному рядку всі текстові макроси і макрофункції на значення, які вони генерують.

2.3 Приклад використання макровизначень

```
  ……
.data
…….
message byte 16 dup (0BBh)
sfc db "AX = %.4X", 13,0
……
.code
start
……
invoke wsprintf, addr message, addr sfc, AX
invoke MessageBox, 0,ADDR message,ADDR mestitle,MB_OK
end start
```

Функція wsprintf форматує блоки даних відповідно до зазначеного формату і поміщає дформатований рядок у буфер відповіді. Параметри функції у прикладі: покажчик на буфер для відповіді; покажчик на зразок формату; один або кілька блоків даних через кому, які будуть відформатовані і поміщені в буфер відповіді. При успішному виконанні в ах повертається кількість символів, поміщених в буфер відповіді.

Найпростіший зразок формату – рядок, який містить символ відсотка і символ – тип формату. %d позначає формат десяткового числа з урахуванням знака (еквівалент %i), %u – ціле число без урахування знака, %x і %X – шістнадцяткове число відповідно в нижньому і верхньому регістрі, %s – рядок символів.

2.3 Контрольні завдання.

Написати додаток, який виводить на екран:

1. За допомогою MessageBox матрицю 2x4 (матриця складається з 0 та 1), кожний елемент має бути відокремлений трьома пробілами. За допомогою MessageBox вміст регістру eax у вигляді матриці 2x4, кожний елемент має бути відокремлений трьома пробілами.

2. За допомогою консолі матрицю 2x4 (матриця складається з 0 та 1), кожний елемент має бути відокремлений трьома пробілами. За допомогою консолі вміст регістру eax у вигляді матриці 2x4, кожний елемент має бути відокремлений трьома пробілами.

3. За допомогою MessageBox матрицю 4x2 (матриця складається з 0 та 1), кожний елемент має бути відокремлений двома пробілами. За допомогою

MessageBox вміст регістру eax у вигляді матриці 4x2, кожний елемент має бути відокремлений двома пробілами.

4. За допомогою консолі матрицю 4x2 (матриця складається з 0 та 1), кожний елемент має бути відокремлений двома пробілами. За допомогою консолі вміст регістру eax у вигляді матриці 4x2, кожний елемент має бути відокремлений двома пробілами.

5. За допомогою MessageBox матрицю 3x2 (матриця складається з 0 та 1), кожний елемент має бути відокремлений двома пробілами. За допомогою MessageBox вміст регістру ax у вигляді матриці 2x2, кожний елемент має бути відокремлений двома пробілами.

6. За допомогою консолі матрицю 3x2 (матриця складається з 0 та 1), кожний елемент має бути відокремлений двома пробілами. За допомогою консолі вміст регістру ax у вигляді матриці 2x2, кожний елемент має бути відокремлений двома пробілами.

7. За допомогою MessageBox матрицю 5x2 (матриця складається з 0 та 1), кожний елемент має бути відокремлений трьома пробілами. За допомогою MessageBox вміст регістрів ax, bx у вигляді десяткових чисел.

8. За допомогою консолі матрицю 5x2 (матриця складається з 0 та 1), кожний елемент має бути відокремлений трьома пробілами. За допомогою консолі вміст регістрів ax, bx у вигляді десяткових чисел

9. За допомогою MessageBox матрицю 2x5 (матриця складається з 0 та 1), кожний елемент має бути відокремлений трьома пробілами. За допомогою MessageBox вміст регістрів al, bl

10. За допомогою консолі матрицю 2x5 (матриця складається з 0 та 1), кожний елемент має бути відокремлений трьома пробілами. За допомогою консолі вміст регістрів al, bl

11. За допомогою MessageBox матрицю 2x4 (елементи матриці 0,1,2,3,4,5,6,7), кожний елемент має бути відокремлений трьома пробілами. За допомогою MessageBox вміст регістрів ax, bl

12. За допомогою консолі матрицю 2x4 (елементи матриці 0,1,2,3,4,5,6,7), кожний елемент має бути відокремлений трьома пробілами. За допомогою консолі вміст регістрів ax, bl

13. За допомогою MessageBox матрицю 4x2 (елементи матриці 0,1,2,3,4,5,6,7), кожний елемент має бути відокремлений двома пробілами. За допомогою MessageBox вміст регістрів bx, al

14. За допомогою консолі матрицю 4x2 (елементи матриці 0,1,2,3,4,5,6,7), кожний елемент має бути відокремлений двома пробілами. За допомогою консолі вміст регістрів bx, al

15. За допомогою MessageBox матрицю 3x2 (елементи матриці 0,1,2,3,4,5), кожний елемент має бути відокремлений двома пробілами. За допомогою MessageBox вміст регістрів ax, al

16. За допомогою консолі матрицю 3x2 (елементи матриці 0,1,2,3,4,5), кожний елемент має бути відокремлений двома пробілами. За допомогою консолі вміст регістрів ax, al

17. За допомогою MessageBox матрицю 5x2 (елементи матриці 0,1,2,3,4,5,6,7,8,9), кожний елемент має бути відокремлений трьома пробілами. За допомогою MessageBox вміст регістрів al, bl у вигляді десяткових чисел

18. За допомогою консолі матрицю 5x2 (елементи матриці 0,1,2,3,4,5,6,7,8,9), кожний елемент має бути відокремлений трьома пробілами. За допомогою консолі вміст регістрів al, bl у вигляді десяткових чисел

19. За допомогою MessageBox матрицю 2x5 (елементи матриці 0,1,2,3,4,5,6,7,8,9), кожний елемент має бути відокремлений трьома пробілами. За допомогою MessageBox вміст регістрів ax, bl у вигляді десяткових чисел

20. За допомогою консолі матрицю 2x5 (елементи матриці 0,1,2,3,4,5,6,7,8,9), кожний елемент має бути відокремлений трьома пробілами. За допомогою консолі вміст регістрів ax, bl у вигляді десяткових чисел

21. За допомогою MessageBox матрицю 2x4 (матриця складається з 2-х 0 та 6-ти 1), кожний елемент має бути відокремлений трьома пробілами. За допомогою MessageBox вміст регістрів bx, al у вигляді десяткових чисел

22. За допомогою консолі матрицю 2x4 (матриця складається з 2-х 0 та 6-ти 1), кожний елемент має бути відокремлений трьома пробілами. За допомогою консолі вміст регістрів bx, al у вигляді десяткових чисел

23. За допомогою MessageBox матрицю 4x2 (матриця складається з 2-х 0 та 6-ти 1), кожний елемент має бути відокремлений двома пробілами. За допомогою MessageBox вміст регістрів ax, al у вигляді десяткових чисел

24. За допомогою консолі матрицю 4x2 (матриця складається з 2-х 0 та 6-ти 1), кожний елемент має бути відокремлений двома пробілами. За допомогою консолі вміст регістрів ax, al у вигляді десяткових чисел.