Одеський національний університет імені І.І. Мечникова
Факультет математики, фізики й інформаційних технологій

Кафедра комп’ютерних систем та технологій

Берков Ю.М., Левченко А.О., Шугайло Ю.Б., Якимчук В.І.

Архітектура комп'ютерів
«Програмування в середовищі MS MASM»
Частина 1.

Методичні вказівки

для студентів, що здійснюють підготовку в галузях знань:
11 - Математика та статистика
12 - Інформаційні технології
15 - Автоматизація та приладобудування

Одеса 2019

ББК 32.973-018.1
УДК 004.42

Берков Ю.М., Левченко А.О., Шугайло Ю.Б., Якимчук В.І.
Програмування в середовищі MS MASM

Одеса, 2019. - 46 с.

Розглянуті питання використання середовища програмування ms masm 6 на мові асемблер, етапи створення програми на masm, особливості використання відладчика та лабораторні роботи з завданнями й прикладами виконання.
Призначено для студентів спеціальності 123 - «Комп'ютерна інженерія».

Укладачі:
Берков Юрій Миколайович, старший викладач ;
Левченко Андрій Олександрович, доцент, к.т.н.;
Шугайло Юрій Борисович, доцент, к.ф.-м.н.;
Якимчук Володимир Іванович, доцент, к.ф.-м.н.;
кафедри комп’ютерних систем та технологій

Рецензенти:

Ніцук Ю.А.. доктор фіз.-мат. наук, професор кафедри
експериментальної фізики ОНУ імені І.І. Мечникова.

Гунченко Ю.О.,доктор технічних наук, професор, кафедри
комп’ютерних систем та технологій

Рекомендовано Вченою радою ОНУ імені І.І. Мечникова
Протокол №3 від 12.02.2020

Зміст

Вступ

Широке розповсюдження персональних ЕОМ вимагає від сучасного ІТ-фахівця знання й навичок розробки програм високої ефективності, що часто може бути досягнуте за допомогою програмування мовою низького рівня (асемблеру), а також шляхом урахування особливостей операційної системи.

Мова програмування Асемблер - це символьна форма запису машинної мови, її використання істотно спрощує написання машинних програм і, у той же час, значно підвищує ефективність використання ресурсів комп'ютера в порівнянні з мовами високого рівня. Особливе значення при цьому набуває розробка резидентних програм переривань операційної системи. Асемблерний код, отриманий при компіляції, також може бути основою для підвищення ефективності програми. Мова Асемблер, у чистому вигляді, відносно рідко використовується на практиці, що пов'язано з машинною орієнтованістю мови. Це означає, що програми мовою Асемблер працюють безпосередньо з ресурсами комп'ютера і вимагають від програміста гарного знання його архітектури і логіки роботи. Універсальні параметризовані фрагменти асемблерних програм доцільно оформляти у вигляді макросів, з яких поступово формується інструментальна бібліотека програміста.

Не дивлячись на це, вивчення мови Асемблера є важливим етапом у підготовці ІТ-фахівців, оскільки дозволяє краще зрозуміти принципи роботи ЕОМ, операційних систем і трансляторів з мов високого рівня, що є необхідною умовою при розробці високоефективних програм.

Лабораторна робота №1. Програми типу COM і EXE.

Тема: Функціональна організація ЕОМ. Компонування, виконання, трасування програми.

Мета роботи: Розвинути навички компонування, запуску й налагодження програми.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Запустити DOSBoxPortable.
1. Створити .com і .exe програми відповідно вашого варіанту.
2. При реалізації програм кожен рядок повинен супроводжуватися коментарями
3. Зберегти програми в папку D:\Project.
4. Записати всі дані, необхідні для оформлення звіту
5. Закрити всі програми.

Завдання. Написати програму мовою асемблер, що виводить текст вашого варіанта на екран.

1. Використовуючи текстовий редактор TASMED (C:\TASMED), також можна використати Norton Commander: Shift + F4 у вікні, що з'явилося, ввести ім'я створюваного файлу: **файл Lab1a.asm** у папці **D:\Project**.
2. Введіть програмний код, замінивши текст «Hello World» текстом вашого варіанта.

```
; ---------- Програмний код ----------
TITLE Вивід тексту (lab1a.asm)
; Виводить текст на екран
.model          tiny                    ;модель пам'яті для com
.code                                   ;початок сегменту коду
```

```
org             100h                            ;початкове значення лічильника
main:
mov             ah, 9                           ;номер функції DOS
mov             dx, offset message              ;адреса рядка в DX
int             21h                             ;виклик системної функції DOS
ret                                             ;завершення .com-програми
message         db "Hello, World!",0Dh,0Ah,'$'  ;рядок для виводу
end             main                            ;кінець програми
```

3. Зробіть трансляцію й компонування програми.

3.1. Використовуючи команди MS DOS перейдіть у папку **D:\Project\ПІБ студента**.

3.2. Введіть команду з урахуванням регістру символів:

ml /AT lab1a.asm

4. Запустіть створену програму в командному рядку (D:\Project \lab1a.com). Зробіть скриншот для звіту.

5. Використовуючи текстовий редактор створіть файл **Lab1b.asm** у папці **D:\Project**.

6. Введіть програмний код, замінивши текст «Hello World» текстом вашого варіанта.

```
; ---------------- Програмний код ----------------------
TITLE Вивід тексту (lab1a.asm)
; Виводить текст на екран
.model          small                           ; Модель пам'яті для створення .exe файлу
.stack          100h                            ; Виділення пам'яті під стек
.code                                           ; Визначення сегмента коду
main:
mov             ax,DGROUP                       ; Адреса початку сегменту даних в AX
mov             ds,ax                           ; Адреса початку сегменту даних в DS
mov             ah,9                            ; Номер функції виводу рядка в AH
mov             dx,offset message               ; Адреса початку рядка message в DX
int             21h                             ; Виклик функції виводу рядка
mov             ax,4C00h                        ; Номер функції закінчення програми в AX
int             21h                             ; Виклик функції закінчення програми
.data                                           ; Визначення сегменту даних
message         db      "Hello, World!",0Dh,0Ah,'$'  ; Визначення рядка
end             main                            ; Закінчення програми.
```

7. Зробіть трансляцію й компонування програми.

7.1. Використовуючи команди MS DOS перейдіть у папку **D:\Project**.

7.2. Ввести команди з урахуванням регістру символів:

ml -Zi -c -Fl lab1b.asm

link /co lab1b, lab1b;

8. Запустіть створену програму в командному рядку (D:\Project \lab1b.com). Зробіть скриншот для звіту.

Завдання для самостійної підготовки

Повторити: - Регістри процесора і їхнє призначення.

- Архітектура 16-ти й 32-х розрядних МП.

Вказівки до змісту звіту

1. Текст програми. Кожний оператор супроводити коментарями.

2. Результат роботи програми (скриншот виконаної програми).

Варіанти для лабораторної роботи №1

Для свого варіанту замініть в тексті «Hello World» слово World своїм прізвищем і ім'ям, написаних латиницею.

Лабораторна робота №2. Робота з відладчиком.

Тема: Процеси асемблювання. Призначення відладчика.

Мета роботи: Дати поняття про процеси асемблювання, навчитися працювати з відладчиком.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Знайомство з відладчиком CodeView.
2. «Прогін» створених програм у відладчику.
3. Вивчення стану регістрів і прапорів.
4. Вивчення вмісту дампу пам'яті.
5. Закриття всіх програм.

Знайомство з відладчиком

Останній етап розробки програми це її налагодження. Хоча етап налагодження є необов'язковим, однак нам необхідно навчитися користуватися відладчиком, тому що він краще дозволить зрозуміти деякі важливі тонкості асемблера. До того ж складно уявити програміста на асемблері (і взагалі будь-якого програміста), що не вмів би користуватися відладчиком.

Існує велика кількість відладчиків під MS-DOS, найбільш відомими з них є: Turbo Debugger від фірми Borland, CodeView від Microsoft, AFDPRO. Існує навіть стандартний консольний відладчик, вбудований у всі версії DOS/Windows, що викликається командою debug.exe, але він дуже незручний.

Ми будемо використовувати тільки відладчик CodeView, тому що він стандартно входить до пакету MASM і зазвичай розташований у піддиректорії \BINR (файл cv.exe). У більшості відладчиків користувальницький інтерфейс подібний до CodeView, тому якщо ви навчитеся користуватися CodeView, то без проблем при бажанні зможете скористатися будь-яким іншим відладчиком.

Відладчики під DOS не підходять для налагодження Windows-додатків. Під Windows можна порадити такі популярні відладчики як SoftIce і OllyDbg.

Як приклад завантажте файл, створеної в попередній лабораторній роботі програми, lab1a.com у відладчик CodeView.

Програма типу COM (lab1a.asm)

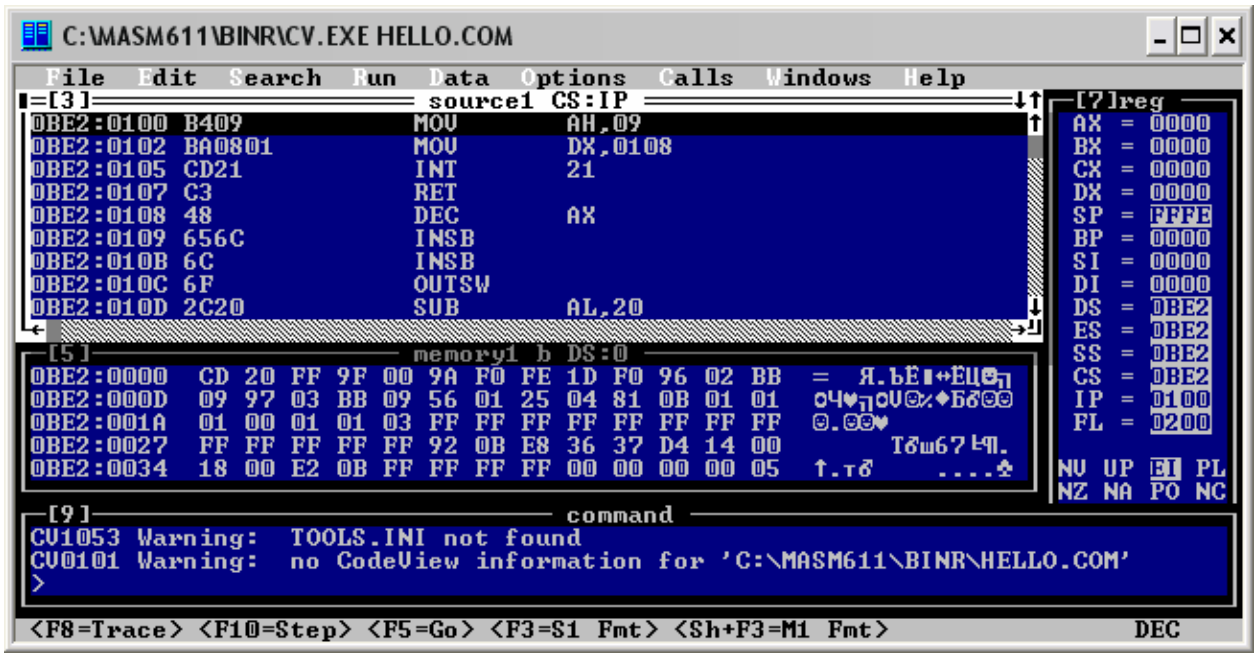
```
.model      tiny
.code
org         100h
main:
```

```
mov             ah,9
mov             dx,offset message
int             21h
ret
message         db      "Hello, World!",0Dh,0Ah,'$'
end             main
```

Для цього запустіть з командного рядка в такий спосіб (усі шляхи до програм вже прописані у DOSBox):

```
cv lab1a.com
```

Відладчик CodeView у вас повинен виглядати приблизно так, як на мал. 1.



Мал. 1. COM-Програма, запущена у CodeView.

Як видно екран відладчика складається з декількох вікон. У кожного вікна в лівому верхньому куті стоїть свій унікальний номер. Звичайно за замовчуванням відкриті наступні вікна:

- Вікно 3 - дизасембльований код.
- Вікно 5 - вміст пам'яті (у шістнадцяткових кодах і у відповідних їм ASCII-символах).
- Вікно 7 - регістри процесора.
- Вікно 9 - командний рядок.

За допомогою меню відладчика "Windows" (меню розташовано в самому верху екрану) можна відкрити інші вікна, просто вибираючи пункти меню з

відповідними назвами. Пункти меню можна вибирати мишею або натисканням клавіші <Alt> і виділяючи стрілками потрібний пункт із наступним натисканням <Enter> для вибору.

Закрити будь-яке вікно можна, клацнувши мишею в його лівому верхньому куті.

Подивимося вміст вікна регістрів (вікно 7). Як бачите відразу після завантаження регістр IP=100h, а сегментні регістри CS, DS, ES, SS мають те саме значення рівне адресі PSP. У нижній частині вікна регістрів показані значення прапорів процесора. Відладчик CodeView використовує особливі умовні позначки, що показують стани прапорів (табл. 1).

Звичайно за замовчуванням у вікні регістрів показуються тільки 16-бітні регістри (AX, BX, CX, ...), але ви можете увімкнути в меню "Options" пункт " 32-bit registers" для відображення 32-бітних регістрів (EAX, EBX, ECX, ...).

Таблиця 1. *Перелік і значення прапорів, виведених у відладчику CodeView*

Назва прапора	**Виведені значення CodeView**	
	Прапор установлений	**Прапор не встановлений**
Ліва колонка		
Прапор переповнення OF	OV	NV
Прапор переривання IF	EI	DI
Прапор нуля ZF	ZR	NZ
Прапор паритету PF	PE	PO
Права колонка		
Прапор напрямку DF	DN	UP
Прапор знака SF	NG	PL
Прапор допоміжного переносу AF	AC	NA
Прапор переносу CF	CY	NC

Тепер подивимося на вміст вікна 3 відладчика. Як видно воно складається із чотирьох колонок:

Адреси	Байти	Інструкції	Операнди
12BD:0100	B409	MOV	AH,09

Перша колонка містить адреси у форматі СЕГМЕНТ:ЗСУВ. Друга колонка машинні коди інструкцій і операндів у шістнадцятковому вигляді. Третя колонка - імена інструкцій. Четверта колонка - операнди інструкцій.

Натискаючи клавішу <F10>, ви можете виконувати трасування програми, тобто порядкове виконання програми, але без заходу в процедури. Для виконання трасування із заходом у процедури потрібно використовувати клавішу <F8>. При цьому можна спостерігати, як у вікні регістрів (вікно 7) будуть змінюватися поточні значення регістрів і прапорів.

Можна виконати відразу цілий фрагмент програми (кілька рядків). Для цього треба встановити курсор у вікні 3 перед тим рядком, на якій потрібно зробити зупинку, і натиснути клавішу <F7>. Виконаються всі рядки програми до того, на якій встановлений курсор; підсвічування переміститься на цей рядок. Далі можна знову виконувати програму порядково, натискаючи клавішу <F10> або, встановивши в потрібному місці курсор, виконати наступний фрагмент, натиснувши <F7>.

У будь-який час можна повернути програму в початковий стан (який був в момент завантаження), тобто виконати рестарт. Для цього треба в головному меню вибрати пункт **Run**, а в підменю цього пункту — пункт **Restart**.

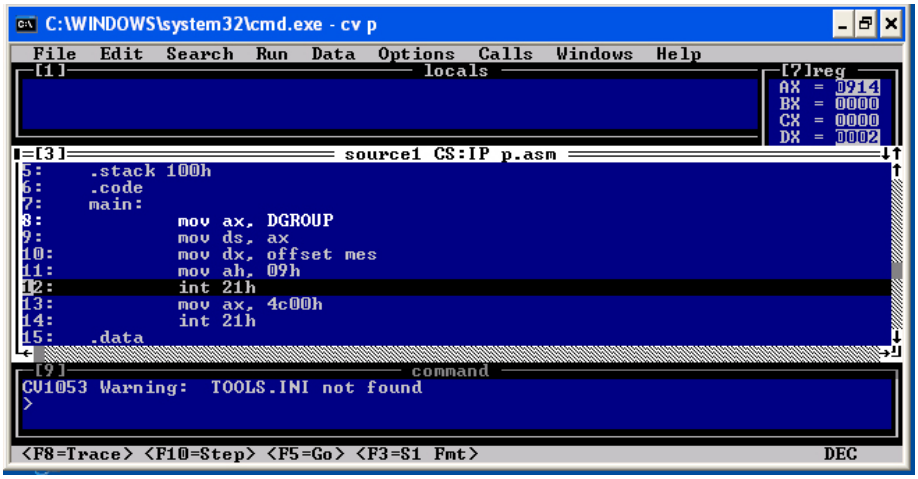
Якщо програма повинна виводити щось на екран, то, не виходячи з відладчика, можна побачити результат роботи програми. Для цього потрібно вибрати меню **View** і в ньому підменю **Output,** або просто натиснути клавішу <F4>. Для повернення у вікно відладчика можна натиснути будь-яку клавішу.

За допомогою меню **File –> Exit** можна вийти з відладчика. Вихід також можна здійснити натисканням комбінації клавіш <Alt>+<F4>.

Програма типу EXE (lab1b.asm)

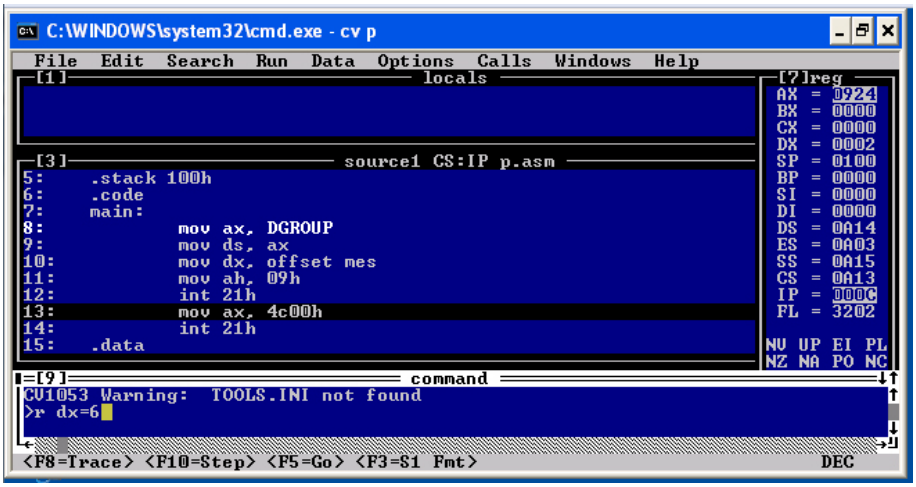
```
.model          small
.stack          100h
.code
main:
mov             ax,DGROUP
mov             ds,ax
mov             ah,9
mov             dx,offset message
int             21h
mov             ax,4C00h
int             21h
.data
Message         db      "Hello, World!",0Dh,0Ah,'$'
end             main
```

1. Запустіть відладчик CodeView зі створеною програмою – у командному рядку введіть **cv lab1b**
2. Поставте курсор на рядок, де перший виклик функції DOS int 21h.



Мал. 2. EXE-Програма, запущена у CodeView.

3. Натисніть **F7** (Щоб прогнати програму до 11 рядка)
4. Запишіть у вікні command команду **r dx=6** (для переходу між вікнами використовуйте **F6**)
5. Натисніть **F10** для виконання наступної команди програми
6. Зробіть скриншот для звіту до лабораторної роботи.



Мал. 3. Відображення вмісту регістрів у CodeView.

7. Натисніть Alt+F4 для виходу з відладчика.

Розміщення даних у пам'яті

Відладчик зараз нам допоможе зрозуміти важливий принцип розміщення даних у пам'яті, що ви повинні добре знати.

Якщо запис у пам'ять здійснюється окремими байтами, то вони розташовуються в природному порядку - у порядку зростання адрес пам'яті. Але якщо здійснюється запис багатобайтових одиниць інформації, то байти розташовуються в пам'яті у зворотному порядку за принципом: молодший байт по молодшій адресі. Це особливість не мови асемблера, а архітектури мікропроцесорів Intel.

Щоб було більш зрозуміло, розглянемо на конкретному прикладі:

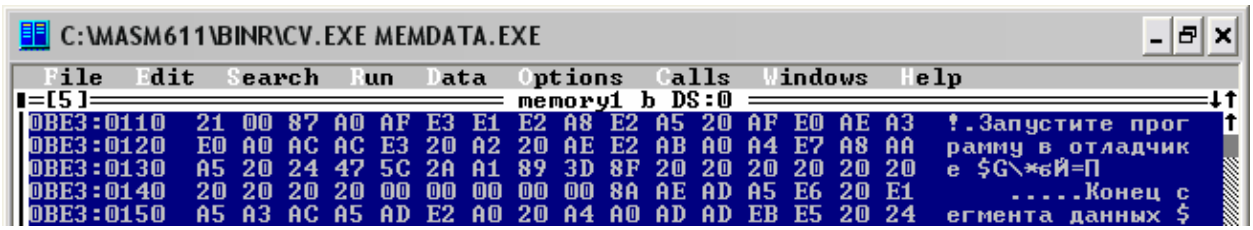
```
.model      small
.stack      100h
.code
start:
mov         ax,@data
mov         ds,ax
mov         ah,9
mov         dx,offset message1
int         21h
mov         ax,4C00h
int         21h
.data
message1    db      "Запустіть програму у відладчику $"
var1        db      47h             ; однобайтове значення
var2        dw      2a5ch           ; двобайтове значення
```

```
var3        dd      8f3d89a1h       ; чотирьохбайтове значення
mas1        db      10 dup (" ")
mas2        db      5 dup (?)
message2    db      "Кінець сегменту даних $"
end         start
```

Запустіть програму у відладчику з командного рядка в такий спосіб:

```
cv memdata.com
```

Потім помістіть курсор у вікні Dump і промотайте трохи у низ, поки не побачите вміст пам'яті сегмента даних нашої програми (мал. 2.4).



Мал. 4. Дамп пам'яті у вікні CodeView.

Як видно відразу за рядком "Запустіть програму у відладчику $" у пам'яті розташовується вміст змінної var1 рівне байту 47h. Далі зверніть увагу, як розміщені в пам'яті байти, що входять у слово змінної var2. Спочатку йде байт 5ch, а потім байт 2ah. Тобто значення розташовані навпаки: спочатку розташований молодший байт значення, а потім старший. Точно також відповідно до принципу "молодший байт по молодшій адресі" розташовані байти, що входять у подвійне слово змінної var3. Потім ідуть десять пробілів масиву mas1, потім п'ять випадкових значень масиву mas2 і потім рядок message2.

Зауважте, що рядки розташовуються в пам'яті не в перевернеому вигляді, тому що визначення виду:

```
message1 db "Запустіть програму у відладчику $"
```

змушує процесор розглядати кожний символ як окремий байт, тому рядок не є багатобайтним значенням.

Варто запам'ятати, що відповідно до принципу "молодший байт по молодшій адресі" розміщаються в пам'яті тільки *багатобайтні значення*, визначаємі в основному за допомогою таких директив як DW, DD, DF, DP, DQ і DT, але тільки не DB.

Коли ви будете виконувати операції з даними за допомогою процесорних команд, таких як ADD - скласти, SUB - відняти та ін., то вам не потрібно замислюватися про розміщення даних у пам'яті, тому що всі перетворення буде здійснювати процесор.

Але якщо ви будете працювати з оперативною пам'яттю на фізичному рівні, то необхідно враховувати можливе перевернене розташування даних.

Завдання для самостійної підготовки

Повторити: - Регістри процесора і їхнє призначення.

- Архітектура 16-ти й 32-х розрядних МП.

Вказівки до змісту звіту

1. Текст програми. Кожний оператор супроводити коментарями.
2. Результат роботи програми (скриншоти виконаної програми й відладчика).

Варіанти для лабораторної роботи №2

Для свого варіанта замініть в тексті «Hello World» слово World своїм прізвищем і ім'ям, написаних латиницею.

Лабораторна робота №3. Найпростіші арифметичні дії.

Тема: Процеси асемблювання. Призначення компілятора, компоновщика, завантажника й відладчика.

Мета роботи: Дати поняття про процеси асемблювання, призначення компілятора, компоновщика, завантажника й відладчика

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Створити програму відповідно вашого варіанту.
2. При реалізації програми кожен рядок повинен супроводжуватися коментарями.
3. Зберегти програму в папку **D:\PROJECT**.
4. Записати всі дані, необхідні для оформлення звіту.
5. Закрити всі програми.

Завдання. Написати програму мовою асемблер, у якій від знайденої суми двох чисел віднімається третє число.

1. Використовуючи текстовий редактор, створіть файл lab3.asm у папці D:\PROJECT.

2. Введіть програмний код, заміняючи текст числами вашого варіанта

```
; ---------------- Програмний код ----------------------
TITLE Додавання й вирахування   (lab3.asm)
; У цій програмі складаються й віднімаються 16-розрядні цілі числа.

.model          small
.stack          100h

.data
.code

main:
mov             ax,@data            ; встановлення адреси сегменту даних
mov             ds,ax               ; копіювання в регістр DS
mov             es,ax               ; копіювання в регістр ES

mov             ax,1000h            ; AX =   1000h
add             ax,4000h            ; AX =   5000h
sub             ax,2000h            ; AX =   3000h

mov              ah,4Ch             ; Номер функції закінчення програми
mov             al,0                ; повертає код = 0
int             21h                 ; виклик функції DOS після закінчення програми
end             main
```

Перейдемо до порядкового опису програми. Спочатку наводиться рядок коду, що аналізується, а потім його опис.

TITLE Додавання й віднімання (lab3.asm)

Директива TITLE по суті є рядком коментарю, у який ви можете помістити будь-який текст. Звичайно після цієї директиви міститься назва програми.

; У цій програмі складаються й віднімаються 16-розрядні цілі числа.

Текст, розташований після символу крапки з комою, є коментарем і тому ігнорується компілятором. Звичайно в коментарі, розташованому після директиви TITLE, наводиться короткий опис програми.

.MODEL small

Ця директива .MODEL указує компіляторові, що потрібно генерувати код для реального режиму роботи процесора.

.data

Директива .data позначає початок *сегменту даних,* у якому розміщаються всі дані, що використовуються в програмі.

.code

Директива .code позначає початок *сегмента коду,* у якому розміщаються всі команди програми, виконувані процесором.

main:

Для єдиної процедури нашої тестової програми ми вибрали ім'я main.

mov ax, 1000h ; AX=1000h

Команда MOV завантажує в регістр AX ціле число 1000h. Її перший операнд (AX) називається *одержувачем,* а другий операнд — *джерелом,*

add ax,4000h ; AX=5000h

Команда ADD додає до вмісту регістра AX число 4000h.

sub ax,2000h ; AX = 3000h

Команда SUB віднімає з регістру AX число 2000h.

END main

Директива END відзначає останній рядок програми, що буде оброблена асемблером. Крім того, у ній вказується ім'я точки входу в програму (тобто адреса, по якій операційна система передасть керування програмі при її запуску).

3. Зробіть трансляцію й компонування прогграммы.

3.1. Використовуючи команди MS DOS перейдіть у папку **D:\Project**.

3.2. Увести команди з урахуванням регістру символів:

ml -Zi -c -Fl lab3.asm

link /co lab3, lab3;

4. Запустіть створену програму в командному рядку (D:\Project\lab3.exe), при цьому не повинно бути ні яких повідомлень про помилки або зависання системи.

5. Відкрийте програму у відладчику CodeView і проженіть в покроковому режимі, перевіряючи правильність виконання арифметичних дій.

6. Зробіть всі необхідні скришоты для складання звіту.

Завдання для самостійної підготовки

Повторити: - Регістри процесора і їхнє призначення.

- Архітектура 16-ти й 32-х розрядних МП.

Вказівки до змісту звіту

1. Текст програми. Кожний оператор супроводити коментарями.

2. Результат роботи програми.

3 Необхідні скриншоти.

Варіанти завдань для лабораторної роботи №3

№ вар.	**Змінна 1**	**Змінна 2**	**Змінна 3**
1.	1000h	4000h	2000h
2.	2000h	5000h	3000h
3.	6000h	5000h	2000h
4.	4000h	4000h	1000h
5.	4000h	4000h	2000h
6.	7000h	6000h	2000h
7.	5000h	6000h	2000h
8.	5000h	6000h	4000h
9.	5000h	6000h	7000h
10.	8000h	6000h	1000h
11.	9000h	7000h	1000h
12.	8000h	7000h	1000h
13.	8000h	6000h	1000h
14.	1000h	6000h	2000h
15.	8000h	6000h	7000h

Лабораторна робота №4. Типи даних.

Тема: Вивчення способів визначення даних мовою Assembler. Представлення даних у пам'яті ЕОМ.

Мета роботи: навчитися способів визначення даних і представлення їх в пам'яті ЕОМ.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Написати програму з даними відповідно вашого варіанту.
2. Одержати файл програми, що виконується, а також файл лістингу.
3. Зберегти отримані файли в папку **D:\PROJECT**.
4. Запустити програму й перевірити правильність рішення завдання
5. Записати всі дані, необхідні для оформлення звіту
6. Закрити всі програми

Завдання. 1. Написати програму мовою Асемблер у якій:

1.1. Ініціалізувати змінні val1,val2 і val3 значення яких указані у вашім варіанті, а тип займає мінімальне значення в пам'яті (у програмі значення змінних вказувати **в шістнадцятковому вигляді**);

1.2. Ініціалізувати змінні val4, val5, val6, val7 і val8 значення яких указані у вашім варіанті, а тип займає мінімальне значення в пам'яті (у програмі значення змінних вказувати в десятковому й експонентному вигляді);

1.3. Ініціалізувати 20 змінних значенням рівним номеру вашого варіанту, а також масив myList з 20 слів, значення яких не визначено;

1.4. Оформити вивід рядка, що містить ваше прізвище й ім'я записані англійською мовою.

val1=100d, val2=4025d, val3=900024d, val4=-20000d, val5=-8000000d, val6=4.1d, val7=3.3E+280, val8=5.6E+400.

2**.** Використовуючи текстовий редактор, створіть файл **lab4.asm** у папці **D:\PROJECT**.

3. Введіть програмний код програми вашого варіанта

```
; ---------- Програмний код ----------
.model          small           ; Модель пам'яті для створення EXE файлу
.stack          100h            ; Виділення пам'яті під стек
.data                           ; Визначення сегменту даних
val1    BYTE    64h             ; Ініціалізація змінної val1
val2    WORD    0FB9h           ; Ініціалізація змінної val2
val3    DWORD   0DBBB8h         ; Ініціалізація змінної val3
val4    SWORD   -20000d         ; Ініціалізація змінної val4
val5    sDWORD  -8000000d       ; Ініціалізація змінної val5
val6    REAL4   -4.1            ; Ініціалізація змінної val6
val7    REAL8   3.3E+280        ; Ініціалізація змінної val7
val8    REAL10  5.6E+400        ; Ініціалізація змінної val8
BYTE            20 DUP(1)       ; Ініціалізація 20-ти змінних рівних 1 (номер вар.)
myList WORD     20 DUP(?)       ; Ініціалізація 20-ти невизначених слів
mes db  " Variant # Name Last_name ", 0Dh, 0Ah, '$'          ; Визначення рядка
.code                           ; Визначення сегменту коду
main:
mov             ax, DGROUP      ; Адреса початку сегменту даних в AX
mov             ds,ax           ; Адреса початку сегменту даних в DS
mov             dx, offset mes  ; Адреса початку рядка mes в DX
mov             ah,9            ; Номер функції виводу рядка в AH
int             21h             ; Виклик функції виводу рядка
mov             ax, 4C00h       ; Номер функції закінчення програми в AX
int             21h             ; Виклик функції закінчення програми
end             main            ; Закінчення програми.
```

4**.** Зробіть трансляцію й компонування програми.

4.1 Використовуючи команди MSDOS перейдіть у папку D:\PROJECT

4.2 Ввести команди з урахуванням регістру символів:

ml -Zi -c -Fl lab4.asm

link /co lab4, lab4;

5**.** Запустіть створену програму в командному рядку (D:\PROJECT\lab4.exe).

Зробіть скриншот для звіту.

6**.** Зміст файлу лістингу **lab4.lst**

```
Microsoft (R) Macro Assembler Version 6.11              06/02/19 02:57:41
lab4.asm                                                 Page 1 - 1
                                .model small ; Модель пам'яті для створення EXE файлу
                                .stack 100h ; Виділення пам'яті під стек
 0000                           .data ; Визначення сегменту даних
 0000 64                                val1 BYTE 64h ; Ініціалізація змінної val1
 0001 0FB9                      val2 WORD 0FB9h ; Ініціалізація змінної val2
```

```
0003 000DBBB8                   val3 DWORD 0DBBB8h ; Ініціалізація змінної val3
0007 B1E0                   val4 SWORD -20000d ; Ініціалізація змінної val4
0009 FF85EE00                   val5 sDWORD -8000000d ; Ініціалізація змінної val5
000D C0833333                   val6 REAL4 -4.1 ; Ініціалізація змінної val6
0011                        val7 REAL8 3.3E+280 ; Ініціалізація змінної val7
     7A2D166D42CB361F
0019                        val8 REAL10 5.6E+400 ; Ініціалізація змінної val8
     453298EC5FD95B566221
0023  0014 [                BYTE 20 DUP(1) ; Ініціалізація 20-ти змінних рівних 1
      01
     ]
0037  0014 [                myList WORD 20 DUP(?) ; Ініціалізація 20-ти невизначених слів
      0000
     ]
005F 56 61 72 69 61 6E      mes db "Variant # Name Last_name", 00h, 0Ah, '$' ; Визначення рядка
     74 20 31 20 52 75
     73 6C 61 6E 20 43
     68 6C 69 73 74 61
     00 0A 24
0000                        .code ; Визначення сегменту коду
0000  B8 ---- --R           main:movax,DGROUP;Адреса початку сегменту даних в AX
0003  8E D8                 mov ds,ax ; Адреса початку сегменту даних в DS
0005  BA 005F R             mov dx, offset mes ; Адреса початку рядка mes в DX
0008  B4 09                 mov ah,9 ; Номер функції виводу рядка в AH
000A  CD 21                 int 21h ; Виклик функції виводу рядка
000C  B8 4C00                   mov ax, 4C00h ; Номер функції закінчення програми в AX
000F  CD 21                 int 21h ; Виклик функції закінчення програми
                            end main ; Закінчення програми.
Microsoft (R) Macro Assembler Version 6.11                  06/02/19 02:57:41
lab4.asm                                                     Symbols 2 - 1
Segments and Groups:
            N a m e             Size     Length   Align   Combine Class
DGROUP . . . . . .              GROUP
_DATA  . . . . . .              16 Bit   007A     Word    Public  'DATA'
STACK  . . . . . .              16 Bit   0100     Para    Stack   'STACK'
_TEXT  . . . . . .16 Bit   0011     Word    Public  'CODE'
Symbols:
            N a m e             Type     Value    Attr
@CodeSize  . . . . . .          Number           0000h
@DataSize  . . . . . .          Number           0000h
@Interface . . . . . .          Number           0000h
@Model . . . . . .              Number           0002h
@code  . . . . . .              Text       _TEXT
@data  . . . . . .Text      DGROUP
@fardata?  . . . . . .          Text       FAR_BSS
@fardata . . . . . .            Text       FAR_DATA
@stack . . . . . .Text      DGROUP
main . . . . . .  L Near   0000     _TEXT
mes  . . . . . .  Byte     005F     _DATA
myList . . . . . .  Word     0037     _DATA
val1 . . . . . .  Byte     0000     _DATA
val2 . . . . . .  Word     0001     _DATA
val3 . . . . . .  DWord    0003     _DATA
val4 . . . . . .  Word     0007     _DATA
val5 . . . . . .  DWord    0009     _DATA
val6 . . . . . .  DWord    000D     _DATA
val7 . . . . . .  QWord    0011     _DATA
val8 . . . . . .  TWord    0019     _DATA
            0 Warnings
            0 Errors
```

Завдання для самостійної підготовки

Повторити: - Мова Assembler, її синтаксис.

- Структура .com і .exe програм.
- Формат надання базових даних у ПК.

Вказівки до змісту звіту

1. Ім'я створюваного фала lab3.asm
2. При написанні програми й оформленні лістингу кожний рядок повинен супроводжуватися коментарями
3. Відобразити результат роботи програми
4. У звіті відобразити файл лістингу lab4.lst

Варіанти завдань для лабораторної роботи №4

№ вар.	val1	val2	val3	val4	val5	val6	val7	val8
1.	100d	4025d	900024d	-20000d	-800000d	4.1d	3.3E+280	5.6E+400.
2.	102d	3002d	800002d	-10000d	-900000d	5.1d	4.3E+280	6.6E+400.
3.	103d	3003d	800003d	-15000d	-850000d	3.1d	3.3E+280	5.6E+400.
4.	104d	3004d	800004d	-16000d	-870000d	7.1d	5.3E+280	4.6E+400.
5.	105d	3005d	800005d	-17002d	-870004d	6.1d	5.4E+280	4.7E+400.
6.	116d	3006d	800016d	-17003d	-570004d	4.1d	5.2E+280	4.4E+390.
7.	117d	3007d	800017d	-17004d	-570005d	4.3d	4.2E+280	5.4E+390.
8.	118d	3018d	800118d	-17008d	-570008d	4.8d	3.4E+280	5.8E+390.
9.	109d	3009d	800019d	-17009d	-570009d	4.3d	4.7E+280	6.4E+390.
10.	110d	3010d	800110d	-15010d	-570009d	4.7d	4.6E+290	6.4E+395.
11.	111d	3011d	800111d	-15011d	-570011d	4.1d	4.5E+290	6.5E+395.
12.	112d	3012d	800112d	-15012d	-570012d	4.2d	3.5E+290	4.5E+395.
13.	113d	3013d	800113d	-15013d	-570013d	4.3d	6.5E+290	4.3E+395.
14.	114d	3014d	800104d	-15114d	-570114d	4.3d	6.5E+291	4.3E+396.
15.	115d	4015d	800115	-15115d	-570115d	4.6d	6.6E+291	4.3E+396.

Лабораторна робота №5. COM - програма з організацією циклу.

Тема: Створення простої .com програми.

Мета роботи: навчити студентів створювати й аналізувати прості .com програми.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Створити файл, що виконується, типу .com, що містить програму, дією якої є вивід на екран усіх ASCII-Символів.
2. Запустити створений файл у командному рядку.
3. Записати всі дані необхідні для звіту.

Завдання. Створити файл, що виконується, типу com, який містить програму, дією якої є вивід на екран всіх ASCII-Символів.

1. Використовуючи текстовий редактор, створіть файл lab5.asm у папці D:\PROJECT.

2. Введіть програмний код

```
; ---------- Програмний код ----------
.model      tiny          ; Модель пам'яті для створення COM файлу
.code                     ; Визначення сегменту коду
org         100h          ; початок COM-файлу
main:
mov         cx,256        ; вивести 256 символів
mov         dl, 0         ; перший символ - з кодом 00
mov         ah,2          ; номер функції DOS "вивід символу"
cloop:      int 21h       ; виклик DOS
inc         dl            ; збільшення DL на 1 - наступний символ
test        dl,0Fh        ; якщо DL не кратний 16,
jnz         continue_loop ; продовжити цикл,
push        dx            ; інакше: зберегти поточний символ
mov         dl, 0Dh       ; вивести CR
int         21h           ; виклик DOS
mov         dl,0Ah        ; вивести LF
int         21h           ; виклик DOS
```

```
pop             dx              ; відновити поточний символ
continue_loop:
loop            cloop           ; продовжити цикл
ret                             ; завершення COM-файлу
end             main            ; завершення main.
```

3. Зробіть трансляцію й компонування програми.

3.1 Використовуючи команди MSDOS перейдіть у папку D:\PROJECT (cd D:\PROJECT)

3.2 Ввести команди з урахуванням регістру символів:

ml /AT lab5.asm

4. Запустіть створену програму в командному рядку (D:\PROJECT\lab4.com). Зробіть скриншот для звіту.

Завдання для самостійної підготовки

Повторити: - Мова Assembler, її синтаксис.

- Структура .com і .exe програм.
- Формат надання базових даних у ПК.

Вказівки до змісту звіту

1. Ім'я створюваного фала lab5.asm

2. При написанні програми й оформленні лістингу кожний рядок повинен супроводжуватися коментарями.

3. Відобразити результат роботи програми.

Лабораторна робота №6. Множення й ділення.

Тема: Використання команд пересилання даних, арифметичних і логічних команд у програмі.

Мета роботи: навчити студентів використовувати арифметичні, логічні й команди пересилання даних.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Написати програму з даними відповідно вашого варіанту
2. Одержати файл програми, що виконується, а також файл лістингу
3. Зберегти отримані файли в папку для лабораторної роботи 5
4. Запустити програму й перевірити правильність рішення завдання
5. Записати всі дані, необхідні для оформлення звіту
6. Закрити всі програми

Завдання. 1. Створіть програму Lab6.exe, для якої виконайте наступні дії:

1.1. Перемістите val1 в DL і виконайте швидке множення val1 на 4 командою shl (результат в DL)

1.2. Виконайте швидке ділення DL на 2 командою shr (результат в DL)

1.3. Виконайте швидке ділення числа зі знаком, що знаходиться в регістрі BL

1.4. Поміняйте місцями значення регістрів BL і BH, використовуючи команду циклічного зсуву

1.5. Обнуліть регістри AX, BX, DX, скиньте значення прапорів

1.6. Виконайте множення змінних (беззнакових слів) val3 на val4

1.7. Обнуліть регістри AX, BX, DX, скиньте значення прапорів

1.8. Виконайте множення 8-розрядних чисел зі знаком, значення яких знаходяться у змінних val5 і val6

1.9. Обнуліть регістри AX, BX, DX, скиньте значення прапорів

1.10. Виконайте ділення беззнакових чисел, що знаходяться у змінних val7 і val8

1.11. Обнуліть регістри AX, BX, DX, скиньте значення прапорів

1.12. Виконайте ділення чисел зі знаком, що знаходяться у змінних val9 і val10

val1=5 ; val2= -128; val3=65534; val4=2; val5=-4;

val6=-4; val7=131; val8=2; val9=-48; val10=5;

2. Використовуючи текстовий редактор, створіть файл lab6.asm у папці D:\PROJECT.

3. Введіть програмний код програми вашого варіанта. Для додавання значень регістрів після виконання математичних операцій у коментарі, скористайтеся відладчиком .

```
; ---------- Програмний код ----------
.model small
.stack 100h
.data
val1    BYTE        5
val2    SBYTE       -128    ;('F'80h)
val3    WORD        65534   ;(FFFEh) найбільше 65535
val4    WORD        2       ; найбільше 65535
val5    SBYTE       -4      ;(FCh)
val6    SBYTE       4       ;(4h)
val7    WORD        131     ; (0083h)
val8    BYTE        2       ; (02)
val9    SBYTE       -48     ;('F'D0h)
val10   SBYTE       5       ;(05h)
.code
.386
main:
mov         ax, @data       ; Сегментна адреса
mov         ds,ax           ; початок визначення даних в DS

;=== Виконайте швидке множення val1 на 4 командою shl (результат в DL) ===
mov         dl, val1        ; DL=5   У DL переміщаємо 5
shl         dl, 2           ; DL=20(14h) Зміщаємо біти DL на 2 ліворуч, помноживши на 2^2

;=== Виконайте швидке ділення DL на 2 командою shr (результат в DL) ===
Shr         dl, 1           ; DL=10(A) Зміщаємо біти DL на 2 праворуч, поділивши на 2^1

;=== Виконайте швидке ділення числа зі знаком, що знаходиться в регістрі BL ===
mov         bl, val2        ; BL=-128('F'80h) (max 127 ; min -128)
sar         bl, 3           ; BL=-16 ('F'F0h) Зміщаємо BL за допомогою команди
                            ;арифметичного зсуву праворуч поділивши тим самим його на 2^3=8

;=== Поміняйте місцями значення регістрів BL і BH використовуючи команду циклічного зсуву ==
rol         bx, 8           ; bh=F0h bl=00
```

```
;=== Обнуліть регістри AX, BX, DX    скиньте значення прапорів===
mov             ax,0
mov             bx,0
mov             dx,0
CLC                             ;Скинути значення прапорів

;=== Виконайте множення змінних (без знакових слів) val3 на val4 ===
mov             ax,val3         ;AX=FFFEh
mul             val4            ;(Результат ) DX=0001 AX=FFFC    CF=1

;=== Обнуліть регістри AX, BX, DX і скиньте значення прапорів===
mov             ax,0
mov             bx,0
mov             dx,0
CLC                             ;Скинути значення прапорів

;=== Множення 8-розрядних чисел зі знаком, що знаходяться у змінних val5 і val6 ===
mov             al, val5        ; AL=FCh
imul            val6            ; AX=FFF0h   OF=0

;=== Обнуліть регістри AX, BX, DX і скиньте значення прапорів===
mov             ax,0
mov             bx,0
mov             dx,0
CLC                             ;Скинути значення прапорів

;=== Виконайте ділення без знакових чисел, що знаходяться в змінних val7 і val8 ===
mov             ax, val7        ; AX=131(0083h)    ділене
div             val8            ; AL=65(41h)-частка AH=1(01h) - залишок

;=== Обнуліть регістри AX, BX, DX і скиньте значення прапорів===
mov             ax,0
mov             bx,0
mov             dx,0
CLC                             ;Скинути значення прапорів

;=== Виконайте ділення чисел зі знаком, що знаходяться в змінних val9 і val10 ===
mov             al, val9        ; AL=-48('F'D0h)    ділене
cbw                             ; Розширюємо знак регістру AL в AH
idiv            val10           ; AL = -9(09h), AH = -3(03h)
mov             ax, 4C00h       ; Номер функції DOS:4C00h завершити програму в AX
int             21h             ; Виклик функції DOS з AX
end             main
```

4. Зробіть трансляцію й компонування програми.

4.1. Використовуючи команди MSDOS перейдіть у папку D:\PROJECT

4.2. Ввести команди з урахуванням регістру символів:

ml /c /Zi /Fl lab6.asm

link /co lab6.obj;

Завдання для самостійної підготовки

Повторити: - Мова Assembler, її синтаксис.

- Структура .com і .exe програм.
- Формат представлення базових даних у ПК.

Вказівки до змісту звіту

1. Заповнити бланк звіту.
2. Ім'я створюваного фала lab6.asm
3. При написанні програми й оформленні звіту кожний рядок повинен супроводжуватися коментарями.
4. При написанні звіту скористайтеся відладчиком CodeView

Варіанти лабораторної роботи №6.

№ вар.	val1	val2	val3	val4	val5	val6	val7	val8	val9	val10
1.	4	-128	65534	2	2	-4	131	65	-48	5
2.	5	-126	65533	2	3	-4	129	65	-51	5
3.	6	-124	65532	2	4	-4	128	65	-61	5
4.	7	-122	65531	2	5	-4	127	65	-52	5
5.	8	-120	65530	2	6	-4	126	65	-53	5
6.	9	-118	65529	2	7	-4	125	65	-54	5
7.	10	-116	65528	2	8	-4	124	65	-57	5
8.	11	-114	65527	2	9	-4	116	65	-56	5
9.	12	-112	65526	2	10	-4	123	65	-58	5
10.	13	-110	65525	2	11	-4	122	65	-59	5
11.	14	-108	65524	2	12	-4	121	65	-62	5
12.	15	-106	65523	2	13	-4	120	65	-63	5
13.	16	-104	65522	2	14	-4	119	65	-64	5
14.	17	-102	65521	2	15	-4	118	65	-66	5
15.	18	-100	65522	2	16	-4	117	65	-67	5

Лабораторна робота №7. Лінійні програми.

Тема: Програмування лінійних обчислювальних процесів.

Мета роботи: Надати навички програмування лінійних обчислювальних процесів.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Написати програму з даними відповідно вашого варіанту.
2. Одержати файл, що виконується, програми з файлом лістингу.
3. Зберегти отримані файли в папку для лабораторної роботи 7.
4. Запустити програму й перевірити правильність рішення завдання.
5. Записати всі дані, необхідні для оформлення звіту
6. Закрити всі програми

Завдання. Реалізуйте обчислення:

val4=(val1+val2)*val3, і
val5=(val1*5)/(val 2-3)

у вигляді асемблерної програми, використовуючи 32-розрядні цілі змінні, а також обчислення

val9=(val6*(-5))/(val7/val8)

використовуючи 16-розрядні змінні зі знаком.

2; val2=4; val3=3; val4=?; val5=?; val6=-2; val7=-100; val8=-50; val9=?;

1. Використовуючи текстовий редактор, створіть файл lab7.asm у папці **D:\PROJECT**.

2. Введіть програмний код програми вашого варіанта. Для додавання значень регістрів після виконання математичних операцій у коментарі, скористайтеся відладчиком.

```
; ---------- Програмний код ----------
.model              small .
stack   1           00h

.data
message_Error BYTE  "Error! Overflow data.", 0Dh,0Ah, "$"
val1    DWORD       2
val2    DWORD       4
val3    DWORD       3
val4    DWORD       ?
val5    DWORD       ?
val6    SWORD       -2
val7    SWORD       -100
val8    SWORD       -50
val9    SWORD       ?

.code
.386
main:
mov                 ax, @data       ; Сегментна адреса
mov                 ds, ax          ; початок визначення даних в DS

COMMENT !
1)      Реалізуйте обчислення val4 = (val1 + val2) * val3
у вигляді асемблерної програми, використовуючи 32-розрядні
цілі змінні
!
CLC
mov                 eax, val1
add                 eax, val2
mul                 val3
jc                  massage
mov                 val4, eax
jmp                 next
; Вивід повідомлення про помилку при переповненні даних.
massage:
mov                 ah,9
mov                 dx, offset message_Error
int                 21h
next:

COMMENT !
2) Реалізуйте обчислення val5 = (val1 * 5) / (val2 - 3)
```

у вигляді ассмблерної програми, використовуючи 32-розрядні
цілі змінні
!

```
CLC
mov             eax, val1       ; Лівий вираз
mov             ebx, 5
mul             ebx             ; EDX:EAX = добуток
mov             ebx, val2       ; Правий вираз
sub             ebx, 3
div             ebx             ; Фінальне ділення
mov             val5, eax
COMMENT !
```

3) Реалізуйте обчислення val9 = (val6 * (-5)) / (val7 / val8)
у вигляді асемблерної програми, використовуючи 16-розрядні
змінні зі знаком
!

```
mov             ax, val7        ; Починаємо із правого виразу
cwd                             ; Розширимо знак діленого
idiv            val8
mov             bx, ax          ; BX значення правого виразу
mov             ax, -5          ; Приступимо до лівого виразу
imul            val6
idiv            bx              ; Фінальне ділення
mov             val9, ax        ; Частка

mov             ax, 4C00h       ; Номер функції DOS:4C00h завершити програму в AX
int             21h             ; Виклик функції DOS з AX
end             main
```

3. Зробіть трансляцію й компонування програми.

3.1. Використовуючи команди MSDOS перейдіть у папку D:\PROJECT (cd D:\PROJECT)

3.2. Ввести команди з урахуванням регістру символів:

ml /c /Zi /Fl lab7.asm

link /co lab7.obj;

Завдання для самостійної підготовки

Повторити:
- Арифметичні команди.
- Типи даних, особливості використання.
- Логічні команди й команди переходу.
- Формат, типи даних, особливості використання.

Вказівки до змісту звіту

1. Заповнити бланк звіту.
2. Ім'я створюваного фала lab7.asm
3. При написанні програми й оформленні звіту кожний рядок повинен супроводжуватися коментарями.
4. При написанні звіту скористайтеся відладчиком CodeView
5. Видалити створені файли, коли робота буде зроблена.

Варіанти завдань для лабораторної роботи №7

№ вар.	val1	val2	val3	val4	val5	val6	val7	val8	val9
1.	2	4	3	?	?	-2	-100	-50	?
2.	20	13	3	?	?	-20	-125	-5	?
3.	20	23	3	?	?	-20	-100	-5	?
4.	5	8	3	?	?	-30	-125	-5	?
5.	10	28	3	?	?	-18	-90	-30	?
6.	25	28	3	?	?	-10	-25	-5	?
7.	30	5	3	?	?	-500	-25	-5	?
8.	40	103	3	?	?	-500	-4	-2	?
9.	50	53	3	?	?	-1000	-200	-10	?
10.	12	33	3	?	?	-8	-12	-3	?
11.	24	33	3	?	?	-12	-90	-3	?
12.	6	8	3	?	?	-24	-80	-4	?
13.	9	8	3	?	?	-48	-120	-2	?
14.	10	4	3	?	?	-6	-12	-4	?
15.	30	53	3	?	?	-30	-10	-2	?

Лабораторна робота №8. Задачі з розгалуженням.

Тема: Програмування задач із розгалуженням.

Мета роботи: Надати навички програмування задач із розгалуженням.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Написати програму з даними відповідно вашого варіанту.
2. Зберегти отримані файли в папку для лабораторної роботи 8.
3. Запустити програму й перевірити правильність рішення завдання
4. Записати всі дані, необхідні для оформлення звіту
5. Закрити всі програми

Завдання. Створіть програму Lab8.exe, що видає текстові повідомлення “N=M”, “N>M” або “N<M” залежно від результатів обчислень

N=5*(val1+val2)
M=val3*(3+val4)

де val1, val2, val3, val4 відповідають вашому варіанту.

11, val2=3, val3=3, val4=8

1. Використовуючи текстовий редактор, створіть файл **lab8.asm** у папці **D:\PROJECT**.
2. Введіть програмний код програми вашого варіанта. Для додавання у коментарі значень регістрів після виконання математичних операцій, скористайтеся відладчиком.

```
; ---------- Програмний код ----------
.model          small
.stack          100h
.data
N_greater_M   BYTE "N>M", 0Dh,0Ah, "$"
N_less_M      BYTE "N<M", 0Dh,0Ah, "$"
N_equal_M     BYTE "N=M", 0Dh,0Ah, "$"

val1    WORD ?
val2    WORD ?
val3    WORD ?
val4    WORD ?
```

```
N       WORD ?
M       WORD ?

.code
.386
main:
mov             ax, @data       ; Сегментна адреса
mov             ds,ax           ; початок визначення даних в DS

COMMENT !
1) Визначте змінні val1, val2, val3, val4 числами, які відповідають вашому варіанту.
!
Mov             ax, 11          ; Визначаємо змінні
mov             val1, ax        ; val1, val2, val3, val4 числами,
mov             ax, 3           ; які відповідають варіанту
mov             val2, ax
mov             ax, 3
mov             val3, ax
mov             ax, 8
mov             val4, ax

COMMENT !
2)Обчисліть N=5*(val1+val2).
Віддповідь запишіть у змінну N.
!
mov             ax, val1
add             ax, val2
mov             bx,5
mul             bx
mov             N, ax

COMMENT !
3)Обчисліть M=val3*(3+val4)
Відповідь запишіть у змінну M.
!
mov             ax, val4
add             ax, 3
mul             val3
mov             M, ax

COMMENT !
4) Зрівняйте значення N і M. Залежно від отриманого
результату сформуйте висновок використовуючи певні
змінні N_greater_M, N_less_M, N_equal_M і мітки
write_N_greater_M, write_N_less_M, write_N_equal_M.
!
mov             ax, N
mov             bx, M
cmp             ax, bx
jg              write_N_greater_M
jl              write_N_less_M
je              write_N_equal_M

write_N_greater_M:
```

```
mov             ah,9
mov             dx, offset N_greater_M
int             21h
jmp             exit_program

write_N_less_M:
mov             ah,9
mov             dx, offset N_less_M
int             21h
jmp             exit_program

write_N_equal_M:
mov             ah,9
mov             dx, offset N_equal_M
int             21h
jmp             exit_program

exit_program:
mov             ax, 4C00h       ; Номер функції DOS:4C00h завершити програму в AX
int             21h             ; Виклик функції DOS із AX
end             main
```

3. Зробіть трансляцію й компонування програми.

3.1. Використовуючи команди MSDOS перейдіть у папку D:\PROJECT (cd D:\PROJECT)

3.2. Ввести команди з урахуванням регістру символів:

ml /c /Zi /Fl lab8.asm

link /co lab8.obj;

Завдання для самостійної підготовки

Повторити: - Арифметичні команди.

- Формат, типи даних, особливості використання.
- Логічні команди й команди переходу.

Вказівки до змісту звіту

1. Заповнити бланк звіту.
2. Ім'я створюваного фала lab8.asm
3. При написанні програми й оформленні звіту кожний рядок повинен супроводжуватися коментарями.

4. При написанні звіту скористайтеся відладчиком CodeView
5. Видалити створені файли, коли робота буде зроблена.

Варіанти завдань для лабораторної роботи №8

№ вар.	**val1**	**val2**	**val3**	**val4**
1.	11	3	3	8
2.	1	1	1	7
3.	1	2	10	4
4.	20	35	2	4
5.	5	5	5	2
6.	30	20	7	4
7.	10	10	20	2
8.	6	4	10	2
9.	2	2	1	1
10.	40	5	7	100
11.	120	80	5	197
12.	15	7	4	10
13.	10	2	25	2
14.	50	2	100	2
15.	100	200	5	57

Література:

1. Кип Р. Ирвин. Язык Ассемблера для процессоров Intel, 4-е издание.: Пер. с англ. – М.: Издательский дом ―Вильямс, 2005. – 912 с.:ил.
2. Абель «Системное программирование», М. Высшая школа, 1990 г., 456 с.
3. Юров В. Assembler: учебник. - СПб.: Питер, 2001. - 624 с.: ил.
4. Н.Г.Голубь “Исскуство программирования на асемблере.Лекции и упражнения”. – Киев “DiaSoft”, 2002 – 642с.
5. Гордеев А. В. Молчанов Л. Ю. Системное программное обеспечение, – СПб.: Питер. 2002. – 734с.
6. Гордеев А. В. Операционные системы: Учебник для вузов. 2-е изд.–СПб.: Питер, 2004. – 416 с.
7. А.Шнайдер.Язык ассемблера для персонального компъютера фирмы IBM.Пер. с англ.- М.,"Мир",1988.

Додаток 1. Microsoft Assembler (ML)

Для компіляції й компонування одного або декількох початкових файлів мовою асемблера використовується програма ML (ML.EXE). Параметри її командного рядка чутливі до регістру символів. Її синтаксис наведений нижче:

ML [[*параметри_асемблера]] ім'я_файлу [[[[параметр]]*

ім'я_файлу]] . . . [[/link *параметри_компоновщика]]*

У командному рядку потрібно вказати як мінімум один параметр — *ім'я_файлу* з початковим кодом програми, написаної мовою асемблера. Наприклад, наведена нижче команда виконує компіляцію початкового файлу Lab.asm і генерує об'єктний файл Lab.obj:

ML -c Lab.asm

Як необов'язкові *параметри* можуть бути вказані один або кілька ключів, кожний з яких починається із символу косої риси (/) або дефіса (-). Ключі вказуються в командному рядку через один або кілька пробілів. Повний список параметрів командного рядка компілятора MASM наведений у табл. 1. Врахуйте, що всі параметри чутливі до регістру символів.

Таблиця 1. Список параметрів командного рядка компілятора MASM

Параметр	***Опис***
/AT	Забезпечує підтримку малюсінької (TINY) моделі пам'яті. Компілятор контролює команди програми й виводить повідомлення про помилки у випадку порушення угод по використанню файлів формату .com. Зверніть увагу, що даний параметр не еквівалентний директиві компілятора .MODEL TINY
/B1*ім'я_файлу*	Вибирає іншу програму-компоновщик
/c	Виконується тільки компіляція програми без компонування
/coff	Генерує об'єктний файл (.obj) у форматі coff (Microsoft Common Object File Format)
/Cp	Зберігає регістр символів всіх ідентифікаторів програми. Вони стають регістрозалежними
/Cu	Перетворює всі ідентифікатори програми до верхнього регістра
/Cx	Зберігає регістр символів у зовнішніх і загальних символах (прийнято за замовчуванням)
/Dсимвол	Визначає текстовий макрос із зазначеним ім'ям. Якщо

[[=значення]]	початкове *значення* не указане, привласнюється порожнє значення. Якщо в *значенні* містяться пробіли, воно повинне бути укладене в подвійні лапки
/EP	Генерується лістинг початкового коду після препроцесора (виводиться на пристрій STDOUT). Див. опис опції /Sf
/F *значення*	Визначає розмір стека в байтах (ця опція є аналогом /link /STACK: *значення). Значення* задається у вигляді шістнадцяткового числа й повинне бути відділене від ключа /F пробілом
/Fe*имя_файлу*	Визначає ім'я файлу, що виконується
/Fl[[ім'я_файлу]]	Генерується лістинг асемблерного коду. Див. опис опції /Sf
/Fm[[*ім'я_файлу*]]	Генерується план (.map-файл) файлу, що виконується, після компонування
/Fo*[[ім'я файлу]]*	Визначає ім'я об'єктного файлу
/Fpi	Генерує список адресних прив'язок для команд із плаваючою комою, використовуваних емулятором (тільки для багатомовного середовища)
/Fr[[ім'я_файлу]]	Генерує файл із розширенням .SBR для програми Source Browser
/FR[[ім'я_*файлу]]*	Генерує розширену версію файлу .SBR
/Gc	Встановлює метод виклику функцій і угода про присвоєння імен, прийнята в мовах FORTRAN або Pascal. Аналогічно директиві MASM OPTION LANGUAGE: PASCAL
/Gd	Установлює метод виклику функцій і угоди про присвоєння імен, прийнята в мові C. Аналогічно директиві MASM OPTION LANGUAGE: З
/H *значення*	Встановлює максимально можливу довжину зовнішніх символів. За замовчуванням прийняте значення 31
/help	Викликає утиліту QuickHelp, що відображає на екрані довідкову інформацію для програми ML
/I *шлях*	Задає шлях до каталогу, у якому містятся файлы, що включаються. У командному рядку допускається використання до 10 параметрів /I
/nologo	Не виводить повідомлення на екран у випадку успішного виконання етапу компіляції
/Sa	Виводить повний лістинг із результатами компіляції
/Sc	Додає в лістинг інформацію про час виконання команд
/Sf	Додає у файл лістингу дані, сгенеровані на першому проході
/Sg	Відображає в лістингу команди, автоматично сгенеровані асемблером
/Sl *ширина*	Задає розмір рядка лістингу в символах. Значення *ширини* можна вибрати в межах від 60 до 255 символів,

	або 0. За замовчуванням встановлений 0. Аналогічний директиві компілятора PAGE , *ширина*
/Sn	Видаляє з лістингу таблицю символів
/Sp *довжина*	Задає довжину сторінки лістингу в рядках. Значення *довжини* можна вибрати в межах від 10 до 255 рядків, або 0. За замовчуванням встановлений 0. Аналогічний директиві компілятора PAGE *довжина*
/Ss *текст*	Визначає підзаголовок для лістингу. Аналогічний директиві компілятора SUBTITLE *текст*
/St *текст*	Визначає заголовок для лістингу. Аналогічний директиві компілятора TITLE *текст*
/Sx	Поміщає в лістинг команди, які не були сгенеровані асемблером через не виконання умови в програмі
/Ta *ім'я_файлу*	Задає ім'я вихідного файлу для асемблювання, що має інше розширення (не . ASM)
/w	Теж, що й /w0
/W*уровень*	Встановлює рівень висновку попереджувальних повідомлень (0, 1,2 або 3)
/WX	Повертає код помилки при генерації попереджувального повідомлення
/Zd	Додає в об'єктний файл інформацію про номери рядків вихідного файлу
/Zf	Робить всі символи програми загальними
/Zi	Додає в об'єктний файл налагоджувальну інформацію для відладчика CodeView
/Zm	Активізує підтримку режиму сумісності з MASM 5.1
/Zp[[вирівнювання]]	Додає структурні змінні на зазначену границю байта, слова або подвійного слова. Значення параметра *вирівнювання* може бути 1, 2 або 4
/Zs	Виконується тільки синтаксичний аналіз програми
/?	Відображає короткий перелік параметрів командного рядка компілятора MASM

Додаток 2. Компоновщик (LINK)

Нижче наведений опис 16-розрядного компоновщика, що входить у поставку MASM. Утиліта LINK призначена для об'єднання декількох об'єктних файлів в один файл, що виконується, або створення бібліотеки, що завантажується динамічно. Її синтаксис наведений нижче:

LINK *параметри об'єктні_файли [[, [[ім'я_ EXE-Файлу]]* [[,
[[*ім'я_ MAP-Файлу]] [[, [[бібліотеки]]* [[,
[*[ім'я_ DEF-Файлу]]]]]]]]]] [[;]]*

Список параметрів командного рядка компоновщика LINK наведений у табл2. У ньому опущені тільки рідко використовувані опції, опис яких можна знайти в довідковій системі.

Таблиця 2. Список параметрів командного рядка компоновщика LINK

Параметр	***Опис***
/A:*розмір*	Повна назва опції: /A [[LIGNMENT]]. Пропонує компоновщику вирівнювати сегменти даних у багатосегментному файлі, що виконується, на зазначену границю. При цьому значення розміру повинне бути кратне 2^n
/B	Повна назва опції: /B [[ATCH]]. Не виводить запит на введення ім'я об'єктного файлу або бібліотеки у випадку, якщо вони не зазначені в командному рядку
/CO	Повна назва опції: /CO [[DEVIEW]]. Поміщає у файл, що виконується, налагоджувальну інформацію (дані про символи й номери рядків вихідної програми), що використовується відладчиком Microsoft CodeView. Дана опція не сумісна з опцією /EXEPACK
/CP:*число*	Повна назва опції: /CP [[ARMAXALLOC]]. Задає максимальну кількість пам'яті в параграфах, виділювану програмі при завантаженні
/DO	Повна назва опції: /DO [[SSEG]]. Упорядковує сегменти так, як це прийнято за замовчуванням у компіляторах високого рівня фірми Microsoft

/DS	Повна назва опції: /DS [[ALLOCATE]]. Пропонує компоновщику розміщати дані починаючи з кінця сегмента даних. Ця опція використовується тільки для асемблерних програм, з яких створюється .Exe-Файл для системи MS DOS
/E	Повна назва опції: /E [[ХЕРАСК]]. Упаковує вміст файлу, що виконується. Ця опція не сумісна з опціями /INCR і /CO. Не використовуйте опцію /EXEPACK при створенні додатків для Windows
/F	*Повна* назва опції: /F [[ARCALLTRANSLATION]] Оптимізує виклик далеких процедур у виконується модуле, що. Ця опція автоматично активізується при використанні опції /TINY. При створенні файлів, що виконуються, для Windows не рекомендується використовувати разом З опцією /FARCALLTRANSLATION ОПЦІЮ /РАСКС
/HE	Повна назва опції: /HE [[LP]]. Викликає утиліту QuickHelp, що відображає на екрані довідкову інформацію для програми LINK
/HI	Повна назва опції: /HI [[GH]]. Пропонує операційній системі завантажувати файл, що виконується, у старші адреси оперативної пам'яті. Ця опція використовується тільки для асемблернbх програм, з яких створюється .Exe-Файл для системи MS DOS
/INC	Повна назва опції: /INC [[REMENTAL]]. Підготовляє файл для виконання компонування із приростом за допомогою утиліти ILINK. Ця опція не сумісна з опціями /EXEPACK і /TINY
/INF	Повна назва опції: / INF [[ORMATION]]. Виводить на консолі стандартні повідомлення про фази побудови вихідного файлу й іменах використовуваних об'єктних файлів
/LI	Повна назва опції: /LI [[NENUMBERS]]. Поміщає в .MAP-Файл інформацію про номери рядків вихідного файлу й відповідних їм адресах. Для роботи цієї опції в об'єктному файлі повинна перебувати інформація про номери рядків. При використанні цієї опції завжди створюється .Map-Файл, навіть якщо його ім'я не зазначене в командному рядку
/M	Повна назва опції: /M [[AP]]. Додає в .Map-Файл інформацію про зовнішні посилання

/NOD[[:*бібліотека*]]	Повна назва опції: /NOD[[EFAULTLIBRARYSEARCH]]. Компоновщик буде ігнорувати зазначену стандартну бібліотеку. При використанні опції /NOD без ім'я бібліотеки, компоновщик буде ігнорувати всі зазначені стандартні бібліотеки
/NOE	Повна назва опції:/NOE[[XTDICTIONARY]]. Забороняє компоновщику шукати додаткові словники в бібліотеках. Звичайно ця опція використовується, якщо при перевизначенні символу виникає помилка L2044
/NOF	Повна назва опції: /NOF [[ARCALLTRANSLATION]]. Оптимізація викликів далеких процедур не виконується
/NOI	Повна назва опції: /NOI [[GNORECASE]]. Зберігає регістр символів в ідентифікаторах
/NOL	Повна назва опції: /NOL [[OGO]]. Не виводить стандартний заголовок програми, що містить інформацію про авторське право
/NON	Повна назва опції: /NON [[ULLSDOSSEG]] Упорядковує сегменти так само, як і при використанні опції /DOSSEG, однак на початку сегмента TEXT (якщо такий визначений), не містяться додаткові байти. Ця опція перекриває дію опції /DOSSEG
/NOP	Повна назва опції: /NOP [[ACKCODE]]. Не впаковує сегменти коду
/PACKC[[:*число*]]	Повна назва опції: /PACKC [[ODE]]. Упаковує сусідні сегменти коду в один сегмент. Якщо зазначено *число,* то воно визначає максимальний розмір упакованого фізичного сегмента
/PACKD[[*-.число*]]	Повна назва опції: /PACKD [[ATA]]. Упаковує сусідні сегменти даних в один сегмент. Якщо зазначено *число,* то воно визначає максимальний розмір упакованого фізичного сегмента. Ця опція використовується тільки при створенні програм, що виконуються, для Windows
/PAU	Повна назва опції: /PAU [[SE]]. При побудові вихідного файлу виконується пауза в роботі компоновщика безпосередньо перед записом файлу на диск. Використовується для заміни носія (дискети)
/PM:*тип*	Повна назва опції: /PM[[TYPE]]. Визначає тип додатка для системи Windows. Замість параметра *тип* використовується одне з наведених значень: PM (АБО WINDOWAPI), VIO (АБО WINDOWCOMPAT) або NOVIO (АБО NOTWINDOWCOMPAT)|

/ST:*розмір*	Повна назва опції: /ST [[АСК]]. Задає розмір сегмента стека від 1 байта до 64 Кбайт
/T	Повна назва опції: /T [[INY]]. Створює програму для MS DOS з використанням малюсінької (TINY) моделі пам'яті. При цьому замість .Exe-Файлу створюється . Сом-Файл. Не сумісний з опцією / INCR
/ ?	Відображає короткий перелік параметрів командного рядка компоновщика LINK

У табл. 3 наведений список використовуваних змінних оточення.

Таблиця 3. Список змінних оточення, використовуваних компоновщиком

Змінна	***Опис***
INIT	Задає шлях до файлу TOOLS.INI
LIB	Задає шлях для пошуку бібліотечних файлів
LINK	Визначає стандартні параметри командного рядка
TMP	Задає шлях до файлу VM.TMP

Додаток 3. Відладчик CodeView (CV)

Відладчик Microsoft CodeView дозволяє запускати файлы, що виконуються, у покроковому режимі й, при наявності налагоджувальної інформації, відображати в окремому вікні вихідний код програми, її змінні, вміст пам'яті, стан регістрів процесора й іншу важливу інформацію. Синтаксис запуску відладчика наступний:

CV *[[параметри]] ім'я_ Exe-Файлу [[аргументи]]*

Список параметрів командного рядка віладчика CodeView для середовища MS DOS наведений у табл. 4.

Таблиця 4. Список параметрів командного рядка відладчика CodeView.

Параметр	***Опис***
/2	Дозволяє використання двох моніторів
/25	При запуску встановлюється відеорежим з 25 рядками
/43	При запуску встановлюється відеорежим з 43 рядками
/50	При запуску встановлюється відеорежим з 50 рядками
/B	При запуску встановлюється чорно-білий відеорежим
/C*команди*	Виконує зазначений список команд при запуску
/F	Заміна вмісту екранів виконується шляхом перемикання відеосторінок
/G	Усуває "сніг" при виводі зображення на CGA-Монітор
/I[[0 | 1]]	Активізує (/I1) або блокує (/I0) виникнення немаскованого переривання й обробку переривань, що надійшли від контролера 8259
/ДО	Блокує перехоплення переривання від клавіатури з боку відладжуваної програми
/M	Блокує використання миші у відладчику CodeView. Ця опція використовується при налагодженні програм, що працюють із мишею в середовищі Windows 3.x
/N[[0|1]]	Опція /N0 пропонує відладчику CodeView обробляти немасковане переривання, а /N1 — ігнорувати це переривання
/R	Дозволяє використання відладочних регістрів процесорів IA-32
/S	Заміна вмісту екранів виконується шляхом копіювання буферів (ця опція використовується при налагодженні графічних програм)
/TSF	Викликає читання інформації з файлу TOOLS.INI і ігнорування файлу CURRENT.STS