# Лабораторна робота № 14

***Тема*: Використання макрозасобів у програмах асемблера.**

*Мета*: Вивчення методики розробки програм з використанням макрозасобів асемблера.

## Вказівки для самостійної підготовки

*Під час підготовки необхідно ознайомитися з матеріалом:*

1. макрокоманди;
2. макродирективи повторення;
3. директиви умовної компіляції;
4. директиви керування процесом генерації макророзширень.

# 1.Теоретичні відомості

* 1. Макрокоманди

***Макрокоманда*** представляє собою рядок, що вміщує деяке символічне ім’я – *ім’я\_макрокоманди***,** призначений для заміщення одним або кількома рядками програмного коду.  
 ***Макровизначенням*** називається шаблон - опис макрокоманди.

Синтаксис макровизначення:

*ім’я\_макрокоманди* macro *список\_формальних\_аргументів* тіло макровизначення  
 endm

Варіанти розташування макровизначення:

1.на початку вихідного тексту програми до сегменту коду і даних;  
2.в окремому файлі. Для доступу до макровизначення в конкретній програмі слід записати на початку цієї програми директиву:

include *ім’я\_файлу*

3.в макробібліотеці. Актуалізація команд з макробібліотеки виконується директивою include.  В результаті використання макрокоманди в програмі формальні аргументи в макровизначенні замінюються відповідними фактичними параметрами. процес такого заміщення називається ***макрогенерацією***, а результатом цього процесу є ***макророзширення***.

**Приклад 1**. Створення і використання макрокоманд

init\_ds macro

;Макрос настройки ds на сегмент даних

mov ax,data

mov ds,ax

endm

out\_str macro str

;Макрос виведення рядка на екран.

;На вході - виведений рядок.

;На виході- повідомлення на екрані.

push ax

mov ah,09h

mov dx,offset str

int 21h

pop ax

endm

clear\_r macro rg

;очищення регістра rg

xor rg,rg

endm

get\_char macro

;введення символу

;введений символ у al

mov ah,1h

int 21h

endm

conv\_16\_2 macro

;макрос перетворення символу шістнадцятерічної цифри

;у її двійковий еквівалент у al

sub al,30h

cmp al,9h

jle $+4

sub al,7h

endm

exit macro

;макрос кінця програми

mov ax,4c00h

int 21h

endm

data segment para public 'data'

message db 'Введіть дві шістнадцятерічні цифри (букви A,B,C,D,E,F - прописні): $'

data ends

stk segment stack

db 256 dup('?')

stk ends

code segment para public 'code'

assume cs:code,ds:data,ss:stk

main proc

init\_ds

out\_str message

clear\_r ax

get\_char

conv\_16\_2

mov cl,4h

shl al,cl

mov dl,al

get\_char

conv\_16\_2

add dl,al

xchg dl,al ;результат у al

exit

main endp

code ends

end main

1.2. Макродирективи

1.2.1. Директиви повторення WHILE, REPTСинтаксис:  
 WHILE *константний\_вираз*  
 *послідовність\_рядків*

ENDM  
 REPT *константний\_вираз*  
 *послідовність\_рядків*

ENDM

При використанні цих директив транслятор буде повторювати послідовність\_рядків до тих пір, поки константний\_вираз не стане рівним нулю. При використанні директиви WHILE константний\_вираз має змінюватися всередині послідовність\_рядків, директива REPT автоматично зменшує на одиницю значення константного виразу після кожної ітерації.

**Приклад 2**. Використання директив повторення WHILE, REPT

def\_sto\_1 macro id\_table, ln:=<5>

;макрос резервування пам'яті довжиною len в сегменті даних

;Використовується WHILE

id\_table label byte

len=ln

while len

db 0

len=len-1

endm

endm

def\_sto\_2 macro id\_table,len

;макрос резервування пам'яті довжиною len  
;Використовується REPT

id\_table label byte

rept len

db 0

endm

endm

data segment para public 'data'

def\_sto\_1 tab\_1,10

def\_sto\_2 tab\_2,10

data ends

end

1.2.2. Директиви повторення IRP, IRPC

Синтаксис директиви IRP:

IRP формальний\_аргумент, < рядок\_символів\_1, рядок\_символів\_2, ... рядок\_символів\_n>

послідовність\_рядків

ENDM

Директива IRP повторює послідовність\_рядків n разів, замінюючи формальний\_аргумент рядком символів з другого операнду.

Синтаксис директиви IRPC:

IRPC формальний\_аргумент, < рядок\_символів >

послідовність\_рядків

ENDM

Дія даної директиви схожа до IRP, але вона при кожній черговій ітерації замінює формальний\_аргумент черговим символом із рядок\_символів. Кількість повторень послідовності рядків визначається кількістю символів в рядок\_символів.

1.2.3.Директиви умовної компіляції IF, IFE

Синтаксис цих директив:

IF(E) логічний\_вираз  
 фрагмент\_програми\_1

ELSE

фрагмент\_програми\_2

ENDIF

Якщо директиві IF логічний вираз істинний, то в об’єктний модуль поміщується фрагмент\_програми\_1. Якщо логічний вираз хибний, то при наявності директиви ELSE в об’єктний код поміщується фрагмент\_програми\_2. Хибним логічний вираз буде, якщо його значення дорівнює нулю, а істинним – при будь-якому значенні, відмінному від нуля.

Директива IFE також аналізує значення логічного виразу, але для включення в об’єктний модуль фрагмент\_програми\_1 потрібно, щоб логічний\_вираз мав хибне значення.

**Приклад 3.** Використання умовних директив IF та IFE

masm

model small

stack 256

def\_tab\_50 macro len

if len GE 50

GOTO exit

endif

if len LT 10

:exit

EXITM

endif

rept len

db 0

endm

endm

.data

def\_tab\_50 15

def\_tab\_50 5

.code

main:

mov ax,@data

mov ds,ax

exit:

mov ax,4c00h

int 21h

end main

**1.2.4.Директиви умовної компіляції IFDEF, IFNDEF**

Синтаксис цих директив:

IF(N)DEF символічне\_ім’я

фрагмент\_програми\_1

ELSE

фрагмент\_програми\_2

ENDIF

Директива IFDEF перевіряє, описане чи ні символічне\_ім’я і якщо це так, то в об’єктний модуль поміщується фрагмент\_програми\_1. В протилежному випадку при наявності директиви ELSE в об’єктний код поміщується фрагмент\_програми\_2.

Дія директиви IFNDEF зворотня IFDEF. Якщо символічне\_ім’я в програмі відсутнє, то транслюється фрагмент\_програми\_1; якщо воно присутнє, то при наявності ELSE транслюється фрагмент\_програми\_2.

1.2.5.Директиви умовної компіляції IFB, IFNB

Синтаксис цих директив:

IF(N)B аргумент

фрагмент\_програми\_1

ELSE

фрагмент\_програми\_2

ENDIF

Директива IFB перевіряє рівність аргументу пробілу. Аргументом може бути ім’я або число. Якщо його значення дорівнює пробілу (фактичний аргумент при виклику макрокоманди не був заданий), то поміщується в об’єктний модуль фрагмент\_програми\_1. В протилежному випадку, при наявності директиви ELSE, в об’єктний модуль поміщується фрагмент\_програми\_2.

Дія директиви IFNB зворотня IFB. Якщо значення аргумента в програмі не дорівнює пробілу, то транслюється фрагмент\_програми\_1. В протилежному випадку, при наявності директиви ELSE, в об’єктний модуль поміщується фрагмент\_програми\_2.

1.2.6.Директиви умовної компіляції IFIDN, IFIDNI, IFDIF, IFDIF

Синтаксис цих директив:

IFIDN(I) аргумент\_1, аргумент\_2

фрагмент\_програми\_1

ELSE

фрагмент\_програми\_2

ENDIF

IFDIF(I) аргумент\_1, аргумент\_2

фрагмент\_програми\_1

ELSE

фрагмент\_програми\_2

ENDIF

В цих директивах перевіряються аргумент\_1 і аргумент\_2 як рядки символів. Директиви IFDNI та IFDIFI ігнорують різницю між малими і великими буквами, а директиви IFIDN i IFDIF – враховують.

Директива IFIDN(I) порівнює символьні значення аргумент\_1 і аргумент\_2. Якщо результат порівняння позитивний (рядки співпадають), то фрагмент\_програми\_1 транслюється і поміщується в об’єктний код. В протилежному випадку, при наявності директиви ELSE, в об’єктний модуль поміщується фрагмент\_програми\_2.

Дія директиви IFDIF(I) зворотня IFIDN(I). Якщо результат порівняння негативний (рядки не співпадають), то фрагмент\_програми\_1 транслюється в об’єктний код. В протилежному випадку, при наявності директиви ELSE, в об’єктний модуль поміщується фрагмент\_програми\_2.

Ці директиви використовують для перевірки фактичних аргументів макрокоманд.

1. Завдання для студентів

1.Розробити програму, яка використовує макрос, що реалізує операції розгалуження і циклу згідно індивідуального завдання.

|  |  |  |  |
| --- | --- | --- | --- |
| Вар. | Завдання | Вар. | Завдання |
| 1 | x + 5 якщо x > 0  y = x якщо x< 0  2x2 якщо x = 0 | 9 | 2x якщо x < 0  y = 5 + x якщо x> 0  x3 якщо x = 0 |
| 2 | 5  y = Π (i + 1 )  i=1 | 10 | 7  y = Π i  i=1 |
| 3 | 10  y = Σ ( i + 2 )  i=1 | 11 | 10  y = Σ ( 2 ‘ i )  i=1 |
| 4 | x + 2 якщо x < 0  y = x2 + 1 якщо x= 0  x2  якщо x > 0 | 12 | x + 4 якщо x > 2  y = x3 якщо x< 2  x2 - 1 якщо x = 2 |
| 5 | 10  y = Π ( i +2 )  i=1 | 13 | 8  y = Σ ( 1+ i )  i=1 |
| 6 | 7  y = Σ ( 3 + i )  i=1 | 14 | 10  y = Π i  i=1 |
| 7 | x2 + 1 якщо x > 1  y = 5x якщо x< 1  x якщо x = 1 | 15 | x + 1 якщо x = 0  y = 5 + x2 якщо x< 0  x3 якщо x > 0 |
| 8 | 9  y = Σ ( 5 + i )  i=1 | 16 | 9  y = Π i 2  i=1 |

2.Написати лістинг програми.

Література

1. Юров В. Assembler. СПБ: Питер, 2001.

1. Использование Turbo Assembler при разработке программ. Киев: "Диалектика". 1994
2. Абель П. Язык Ассемблера для IBM PC и программирования