МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ІКНІ
Кафедра **ПЗ**

ЗВІТ

до лабораторної роботи № 7
з дисципліни: *"Архітектура комп'ютера"*
на тему: *"Опрацювання рядка символів засобами асемблера мікропроцесорів x86. Робота з файлами"*

Лектор:
доц. каф. ПЗ
Крук О.Г.

Виконав:
ст. гр. ПЗ-22
Ясногородський Н.В.

Прийняв:
доц. каф. ПЗ
Крук О.Г.

« ____ » _____ 2022 р.

Σ= _____

Львів – 2022

Тема роботи: Опрацювання рядка символів засобами асемблера мікропроцесорів x86. Робота з файлами

Мета роботи: освоїти команди асемблера для роботи з рядками символів; опанувати функції Win32 для роботи з файлами; розвинути навики складання програми для опрацювання рядка символів та програми для створення, записування і читання текстового файла; відтранслювати і виконати в режимі відлагодження програми, складені відповідно до свого індивідуального завдання.

Варіант: 30

30	$П_7, П_1, П_4, П_5, П_6, П_2, П_3$

Теоретичні відомості

В системі команд процесорів Intel передбачено п'ять груп команд для оброблення масивів байтів, слів та подвійних слів. Незважаючи на те, що всі вони називаються рядковими примітивами, область їх використання не обмежується тільки масивами рядків. З огляду на це доцільніше використовувати їх іншу назву – ланцюжкові команди.

Для адресації пам'яті в цих командах використовуються регістри ESI та EDI. Особливість цих команд полягає в тому, що обидва операнди розташовані в пам'яті. При обробленні рядкових примітивів ці команди можуть автоматично повторюватися, що робить їх застосування особливо зручним для роботи з довгими рядками та масивами.

При роботі програми в захищеному режимі адресація пам'яті в командах оброблення рядкових примітивів може здійснюватися через регістри ESI або EDI. При цьому зміщення, що міститься в регістрі ESI, відраховується відносно сегмента, чий дескриптор вказаний в регістрі DS, а зміщення, вказане в регістрі EDI, відраховується відносно сегмента, чий дескриптор вказаний в регістрі ES.

При використанні лінійної моделі пам'яті в сегментних регістрах DS та ES міститься одне і те ж значення, яке в програмі неможна змінювати.

Використання префікса повторення. Самі по собі команди оброблення рядкових примітивів виконують тільки одну операцію над байтом, словом або подвійним словом пам'яті. Однак, якщо перед ними вказати префікс повторення, виконання команди буде повторено стільки разів, скільки вказано в регістрі ECX. Тобто з допомогою префікса можна виконати оброблення цілого

масиву за допомогою всього однієї команди. Існує кілька типів префіксів повторення:

REP - Повторювати команду, поки ECX> 0;

REPZ, REPE - Повторювати команду, поки ECX > 0 і прапорець нуля установлений (ZF = 1);

REPNZ, REPNE - Повторювати команду, поки ECX > 0 і прапорець нуля скинутий (ZF = 0).

Прапорець напрямку DF. Стан цього прапорця впливає на те, який напрямок переміщення по рядку і як в процесі виконання команд оброблення рядкових примітивів змінюються значення регістрів ESI та EDI. Якщо прапорець DF скинутий (напрямок - прямий), вони збільшуються на розмір оброблюваного операнда (1, 2 або 4 байти), а якщо встановлений (напрямок - зворотний), то відповідно зменшуються.

Значення прапорця напрямку DF можна явно задати за допомогою команд CLD та STD:

CLD ; Скидає прапорець напрямку DF (напрямок – прямий)

STD ; Встановлює прапорець напрямку DF (напрямок - зворотний)

Індивідуальне завдання

1. В сегменті даних опишіть рядок символів, в якому є такі поля: 1) прізвище; 2) ім’я; 3) по батькові; 4) дата народження; 5) місто/село; 6) область; 7) навчальна група (номерів полів не ставити). Перед кожним полем повинна бути довільна (різна) кількість пропусків (пробілів). Після останнього поля теж має бути хоча би один пропуск. В полях пропусків неповинно бути. Всі символи мають бути латинськими. Рядок символів має закінчуватися '00'.
2. В сегменті даних опишіть окремі рядки для кожного поля довжиною, що дорівнює номеру групи, тобто 21/22/23/24/25/26. Опишіть також однобайтові змінні, в яких потрібно буде вказати довжину кожного поля. Опишіть другий рядок символів, довжина якого більша від довжини першого рядка, створеного в пункті 1, на 30 символів. Опишіть третій рядок символів, довжина якого дорівнює довжині першого рядка.
3. Для роботи з рядками символів використовувати лише ланцюжкові команди! Складіть підпрограму, яка підраховує і пропускає символи пропуску. Вхідним параметром підпрограми має бути індекс пропуска в рядку, з якого починати пошук. Вихідним параметром має бути індекс першого символа, який не є пропуском.
4. Складіть підпрограму, яка визначає довжину поля. Вхідним параметром підпрограми має бути індекс символа в рядку, з якого починається поле. Вихідним параметром має бути індекс першого пропуску після поля.
5. В головній програмі організуйте пошук і пересилання кожного поля у відповідний рядок, підрахунок кількості символів в кожному полі.

6. Обчисліть загальну кількість пропусків в початковому рядку символів.
7. Перешліть в другий рядок символів кожне поле відповідно до послідовності, вказаної в індивідуальному завданні. Перед кожним полем мають бути пропуски, кількість яких дорівнює номеру поля. Рядок символів має закінчуватися '00'.
8. Для контролю виведіть другий рядок символів на екран.
9. За допомогою функції CreateFile створіть текстовий файл для читання – записування, ім’я файла – ваше прізвище.
10. Запишіть у створений файл спочатку другий, а потім перший рядки сиволів.
11. Закрийте файл.
12. Відкрийте файл.
13. Організуйте у файлі доступ і прочитайте перший рядок символів (записаний у файлі після другого) в третій рядок.
14. Підрахуйте, скільки разів перший символ (не пропуск) присутній в третьому рядку.
15. Допишіть в кінець файлу назви двох дисциплін (на вибір), які вивчали на першому курсі і оцінки з них.
16. Закрийте файл.
17. Перевірте результат роботи програми.
18. Збережіть програму.
19. У звіті наведіть текст програми, копії вікон зі всіма змінними, а також створений текстовий файл.

Код програми на мові Асемблера

```
        INCLUDE Irvine32.inc
       .686
       .model  flat, c
       .stack

       .data

       Info    byte "  yasnogorodskyi   nikita       victorivich    02.12.2003       kyiv
kyivska        pz22  ", 00
       InfoLen EQU $-Info

       Subjects    byte 0Dh, 0Ah, "OP - 99, OOP - 99", 00
       SubjectsLen EQU $-Subjects

       Surname         byte     22 DUP(?)
       FirstName       byte     22 DUP(?)
       MiddleName      byte     22 DUP(?)
       Birth           byte     22 DUP(?)
       Town            byte     22 DUP(?)
       Region          byte     22 DUP(?)
       UniversityGroup byte     22 DUP(?)
```

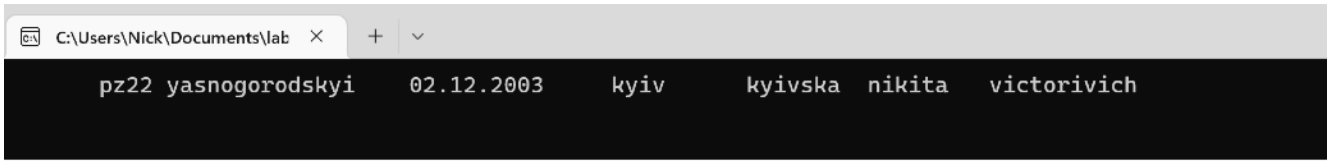
```
        SurnameL    byte     0
        NameL       byte     0
        MiddleNameL byte     0
        BirthL      byte     0
        TownL       byte     0
        RegionL     byte     0
        GroupL      byte     0

        Info2    byte  30+InfoLen DUP(?), 0Dh, 0Ah, 00
        Info2Len EQU $-Info2

        Info3    byte  InfoLen DUP(?)
        Info3Len EQU $-Info3

        AllSpaces byte 0

        CountFirstSym byte 0

        fileHandle DWORD ?

        .code

START:

        mov EDI, OFFSET Info; EDI - pointer to Info
        mov ECX, InfoLen; ECX - length of Info
        cld ; direction flag - forward

surnameField:
        ;    count spaces
        call CountingSpaces
        add  AllSpaces, DL

        ;    count Field Length
        call FieldLenght
        mov  SurnameL, DL

        ;    copy from Info to Field variable
        mov  EAX, OFFSET Surname
        call CopyInfoToVariable

firstNameField:
        call CountingSpaces
        add  AllSpaces, DL

        call FieldLenght
        mov  NameL, DL

        mov  EAX, OFFSET FirstName
        call CopyInfoToVariable

middleNameField:
        call CountingSpaces
        add  AllSpaces, DL

        call FieldLenght
        mov  MiddleNameL, DL

        mov  EAX, OFFSET MiddleName
```

```
       call CopyInfoToVariable

birthField:
       call CountingSpaces
       add  AllSpaces, DL

       call FieldLenght
       mov  BirthL, DL

       mov  EAX, OFFSET Birth
       call CopyInfoToVariable

townField:
       call CountingSpaces
       add  AllSpaces, DL

       call FieldLenght
       mov  TownL, DL

       mov  EAX, OFFSET Town
       call CopyInfoToVariable

regionField:
       call CountingSpaces
       add  AllSpaces, DL

       call FieldLenght
       mov  RegionL, DL

       mov  EAX, OFFSET Region
       call CopyInfoToVariable

groupField:
       call CountingSpaces
       add  AllSpaces, DL

       call FieldLenght
       mov  GroupL, DL

       mov  EAX, OFFSET UniversityGroup
       call CopyInfoToVariable

       call CountingSpaces
       add  AllSpaces, DL

startCopyToInfo2:
       mov AL, ' '
       mov EDI, OFFSET Info2
       cld

copyGroupToInfo2:
       mov ECX, 7; number of spaces to write
       rep stosb
       mov CL, GroupL; CL is part of ECX
       mov ESI, OFFSET UniversityGroup
       rep movsb

copySurnameToInfo2:
       mov ECX, 1
       rep stosb
```

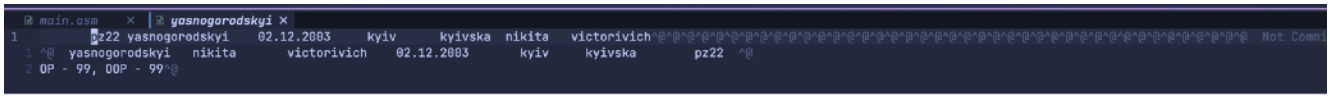
```
	mov CL, SurnameL
	mov ESI, OFFSET Surname
	rep movsb

copyBirthToInfo2:
	mov ECX, 4
	rep stosb
	mov CL, BirthL
	mov ESI, OFFSET Birth
	rep movsb

copyTownToInfo2:
	mov ECX, 5
	rep stosb
	mov CL, TownL
	mov ESI, OFFSET Town
	rep movsb

copyRegionToInfo2:
	mov ECX, 6
	rep stosb
	mov CL, RegionL
	mov ESI, OFFSET Region
	rep movsb

copyNameToInfo2:
	mov ECX, 2
	rep stosb
	mov CL, NameL
	mov ESI, OFFSET FirstName
	rep movsb

copyMiddleNameToInfo2:
	mov ECX, 3
	rep stosb
	mov CL, MiddleNameL
	mov ESI, OFFSET MiddleName
	rep movsb; copy MiddleName to Info2 (ESI → EDI)

writeToInfoFile:
	;	display Info2 before writting
	mov	EDX, OFFSET Info2
	call	Writestring
	;	create file and write Info2, Info to it
	INVOKE CreateFile, ADDR Surname, GENERIC_WRITE, DO_NOT_SHARE, NULL, CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL, 0
	mov	fileHandle, EAX

	.IF eax =/= INVALID_HANDLE_VALUE

	INVOKE WriteFile, fileHandle, ADDR Info2, Info2Len, 0, 0
	INVOKE WriteFile, fileHandle, ADDR Info, InfoLen, 0, 0

	INVOKE CloseHandle, fileHandle

	.ENDIF

readFromFileAndWriteToInfo3:
	;	open file, read Info2 from it and write to Info3
```

```
	INVOKE CreateFile, ADDR Surname, GENERIC_READ or GENERIC_WRITE, DO_NOT_SHARE, NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0
	mov    fileHandle, EAX

	INVOKE SetFilePointer, fileHandle, Info2Len, 0, FILE_BEGIN

	INVOKE ReadFile, fileHandle, ADDR Info3, Info3Len, 0, 0

	;    Count first symbol occurences in Info3
	mov  EDI, OFFSET Info3
	mov  ECX, Info3Len
	call CountingSpaces
	mov  AL, [EDI]; search for value in AL
	mov  CountFirstSym, 0

	;----------------
	; Procedures
	;----------------

CountSym:
	repne scasb
	inc   CountFirstSym
	inc   ECX

	loop CountSym
	dec  CountFirstSym

	INVOKE WriteFile, fileHandle, ADDR Subjects, SubjectsLen, 0, 0
	INVOKE CloseHandle, fileHandle

	RET

CopyInfoToVariable:
	;   EAX  - Field variable offset
	mov EBX, ECX; EBX - length of Info after Fragment
	mov CL, DL; ECX = CL - length of Fragment
	sub EDI, ECX; EDI - pointer to Fragment
	mov ESI, EDI; ESI - pointer to Fragment in Info
	mov EDI, EAX; EDI - pointer to Fragment in Fragment variable

	rep movsb; copy Fragment from Info to Fragment variable (ESI → EDI)

	mov EDI, ESI
	mov ECX, EBX
	ret

CountingSpaces:
	mov  EDX, ECX; save current idx
	mov  AL, ' '
	repe scasb; repeat while equal to ' '

	dec EDI
	inc ECX

	sub EDX, ECX; diff starting and ending idx to find length
	ret

FieldLenght:
	mov   EDX, ECX; save current idx
	mov   AL, ' '
```

```
repne scasb; find first space

dec EDI
inc ECX

sub EDX, ECX; diff starting and ending idx to find length
ret

END START
```



Вивід Info2 в консоль



Вміст файлу з вихідними данними

Висновки

Під час виканання лабораторної, я освоїв команди асемблера для роботи з рядками символів та опанувати функції Win32 для роботи з файлами