КРАСНОЛИМАНСЬКИЙ ЗАОЧНИЙ ФАКУЛЬТЕТ

Кафедра „Автоматика та комп'ютерні системи"

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з дисципліни

„МІКРОПРОЦЕСОРНА ТЕХНІКА"

для студентів заочної форми навчання

Харків 2012

Методичні вказівки розглянуто та рекомендовано до друку на засіданні кафедри „Автоматика і комп'ютерні системи" 27 листопада 2010 року, протокол № 3.

Укладачі:

доц. О.Ф. Єнікєєв,
студ. Д.Ю. Захаренков,
старш. викл. Т.С. Шлак

Рецензент

доц. О.І. Семененко

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з дисципліни

„МІКРОПРОЦЕСОРНА ТЕХНІКА"

для студентів заочної форми навчання

Відповідальний за випуск Єнікєєв О.Ф.

Редактор Еткало О.О.

Підписано до друку 10.12.10 р.

Формат паперу 60x84 1/16 . Папір писальний.

Умовн.-друк.арк. 0,7. Тираж 50. Замовлення №

Видавець та виготовлювач Українська державна академія залізничного транспорту
61050, Харків - 50, майдан Фейєрбаха, 7
Свідоцтво суб'єкта видавничої справи ДК № 2874 від 12.06.2007 р.

УКРАЇНСЬКА ДЕРЖАВНА АКАДЕМІЯ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

КРАСНОЛИМАНСЬКИЙ ЗАОЧНИЙ ФАКУЛЬТЕТ

Кафедра „Автоматика та комп'ютерні системи"

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни

„Мікропроцесорна техніка"

для студентів заочної форми навчання

Харків – 2012

Методичні вказівки розглянуто та рекомендовано до друку на засіданні кафедри „Автоматика і комп'ютерні системи" 27 листопада 2010 року, протокол № 3.

Укладачі:
доц. О.Ф. Єнікєєв
Д.Ю. Захаренков
Т.С. Шлак

Рецензент:
доц. О.І. Семененко

ЗМІСТ

ВСТУП

Методичні вказівки до програмування мовою асемблера з дисципліни "Мікропроцесорна техніка" містять опис чотирьох двогодинних лабораторних робіт. Лабораторні роботи орієнтовані на користування IBM PC сумісних ПЕОМ. Метою лабораторних робіт є закріплення практичних навичок складання та налагодження програм.

1 ЗАГАЛЬНІ МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Студенти групи об'єднуються в бригади по 2-3 людини, які працюють на закріпленому комп'ютері. Кожен студент отримує індивідуальне завдання згідно з номером свого прізвища в журналі і оформляє звіт. Виконання лабораторної роботи припускає попереднє вивчення відповідного розділу курсу і методичних вказівок до чергової роботи. Для допуску до виконання лабораторної роботи студент повинен ознайомитися з темами для опрацювання та попередньо написати текст програми відповідно до індивідуального завданням. Протягом виконання лабораторної роботи студент повинен відповісти на контрольне питання.

Обов'язковими є такі пункти у звіті з лабораторної роботи:

- тема лабораторної роботи;
- мета роботи;
- індивідуальне завдання;
- структура програми та схеми алгоритмів;
- тексти програм;
- висновки з роботи.

Звіт повинен бути акуратно оформлений. Зошит, що містить звіти з усіх лабораторних робіт, потрібно здати викладачу після виконання і захисту всіх робіт.

2 ІНДИВІДУАЛЬНІ ЗАВДАННЯ

1 Задані масиви A і B за N = 25 елементів (констант типу WORD). Сформувати масив З за правилом: Ci = Ai + Bi, якщо Ai > Bi, Ci = Ai - Bi, якщо Ai <= Bi.

2 Задано масив A з N = 50 елементів (констант типу WORD). Сформувати масив B з перших 10 негативних елементів масиву A.

3 Задано масив A з N = 50 елементів (констант типу WORD). Сформувати масив B з елементів A, що задовольняють умову Ai > E = 5.

4 Задано масив A з N = 40 елементів (констант типу WORD). Визначити суму і кількість елементів масиву A, що задовольняють умову Ai > E = -10.

5 Задано масиви A і B за N = 50 елементів (констант типу WORD). Визначити кількість пар елементів, що задовольняють умову Ai > Bi.

6 Задано масиви A і B за N = 50 елементів (констант типу WORD). Визначити кількість елементів Ai, що задовольняють умову Ai <= Bi.

7 Задано масиви A і B за N = 50 елементів (констант типу WORD). Визначити максимальний з елементів масиву A, які задовольняють таку умову Ai <Bi.

8 Задано масиви A і B за N = 25 елементів (констант типу WORD). Сформувати масив З за правилом: Cj = Bi, якщо Ai+Bi>0.

9 Задано масив A з N = 40 елементів (констант типу WORD). Визначити кількість елементів масиву A, які задовольняють таку умову L <Ai <= M, де L = 3 та M = 15.

10 Задано масив A з N = 50 елементів (констант типу WORD). Визначити мінімальний з позитивних елементів масиву A, де Ai E [-100, +100].

11 Задано масив A з N = 60 елементів (констант типу WORD). Визначити максимальний з негативних елементів масиву A, де Ai має такий діапазон: E [-100, +100].

12 Задано масив A з N = 80 елементів (констант типу WORD). Структура масиву A: ----------

|X1|Y1|X2|Y2|.....|

Визначити кількість пар, для яких виконується умова Xi > Yi.

13 Задані масиви A і B за N = 25 елементів (констант типу WORD). Визначити суму і кількість елементів Ai, що задовольняють умову Ai> Bi.

14 Задано масив A з N = 60 елементів (констант типу WORD). Структура масиву A: ----------

|X1|Y1|X2|Y2|.....|

Визначити суму елементів Xi, для яких виконується умова Xi<=Yi.

15 Задано масив A з N = 100 елементів (констант типу WORD). Структура масиву A: ----------

|X1|Y1|X2|Y2|.....|

Сформувати масив B з елементів Xi, для яких виконується умова Xi = Yi.

16 Задано масив A з N = 80 елементів (констант типу WORD). Структура масиву A: ----------

|X1|Y1|X2|Y2|.....|

Сформувати масив B з номерів пар елементів, для яких виконується така умова Xi = Yi.

17 Задано масив A з N = 30 елементів (констант типу WORD). Сформувати масив У з індексів елементів масиву A, що задовольняють таку умову: (A [i] - A [i +1])> C, де C = 5.

18 Задано масив A з N = 50 елементів (констант типу WORD). Визначити суму елементів масиву A, для яких біти 2 і 10 не збігаються.

19 Задано масив A з N = 30 елементів (констант типу WORD). Визначити кількість елементів масиву A, для яких біти 3 і 11 збігаються.

20 Задані масиви A і B за N = 30 елементів (констант типу WORD). Сформувати масив З за правилом: Cj = Ai + Bi, якщо біти 4 і 9 у елементів Ai та Bi збігаються.

21 Задано масив A з N = 100 елементів (констант типу WORD). Сформувати масив У з індексів елементів масиву A, у яких 1, 3 та 5 біти містять одиниці.

22 Задано масив A з N = 25 елементів (констант типу WORD). Сформувати масив B з елементів масиву A, у яких 0, 2 і 5 біти містять нулі.

ЛАБОРАТОРНА РОБОТА 1

СИСТЕМНЕ ОБСЛУГОВУВАННЯ ПРОГРАМ НА АСЕМБЛЕРІ

Мета роботи: навчитися використовувати можливості системних програм, що обслуговують програми на асемблері.

Постановка завдання: створити текстовий файл на асемблері. Отримати файл з розширенням exe, який можливо виконати за допомогою персонального комп'ютера. Записати черговість етапів формування файла, деталі їхньої реалізації та отримані при цьому результати.

Порядок виконання роботи:

- на рисунку 1.1 вказано системні програми та файли, з якими вони працюють. Програми ***masm*** і ***link*** використовуються для створення виконуваного файла за вхідним модулем на асемблері;
- назва файла складається з двох частин, що поділяються точкою. Праворуч від точки записують розширення імені файла, яке може містити не більше трьох символів. Зліва від точки записують власне ім'я файла, яке може містити не більше восьми символів. У даному випадку розширення імен файлів можуть бути, але не обов'язково такими, як зазначено на рисунку 1.1. Імена файлів x, y, z, s записані умовно. Вони можуть бути замінені користувачем на ті, які йому більш зручні для використання. У всіх цих файлів може бути одне і те ж ім'я, але природно різні розширення;
- до використання програми ***masm*** треба створити файл ***x.asm***. Це так званий початковий модуль на асемблері. Це текстовий файл і для його створення можна використовувати будь-яку системну програму, яка називається редактором тексту. Файлу, який створюється, спочатку присвоюється ім'я ***x.asm***, а

потім відтворюється на екрані потрібний текст на асемблері. Доцільно при друці тексту використовувати мінімально необхідну кількість прогалин. Коли текст буде надруковано, треба його зберегти. У каталозі з'явиться гілка D: \ MASM \ x.asm;

- тепер можна використовувати програму ***masm***, яка за початковим модулем (файл ***x.asm***) створить об'єктний модуль (файл ***y.obj***). Макроасемблер (***masm.exe***) – системна програма (далі асемблер). Програма ***masm*** з файлами ***x.asm*** і ***y.obj*** працює завжди, а файли ***z.lst*** (лістинг) і ***w.crf*** (файл перехресних посилань) ця програма може створити, якщо користувач вкаже, що вони йому необхідні;
- для отримання виконуваного модуля (файл ***n.exe***) треба використовувати системну програму ***link.exe*** (Редактор зв'язків). Для цієї програми файл ***y.obj*** є вхідним;
- після того, як виконуваний файл ***n.exe*** буде отримано, його можна запустити на виконання.

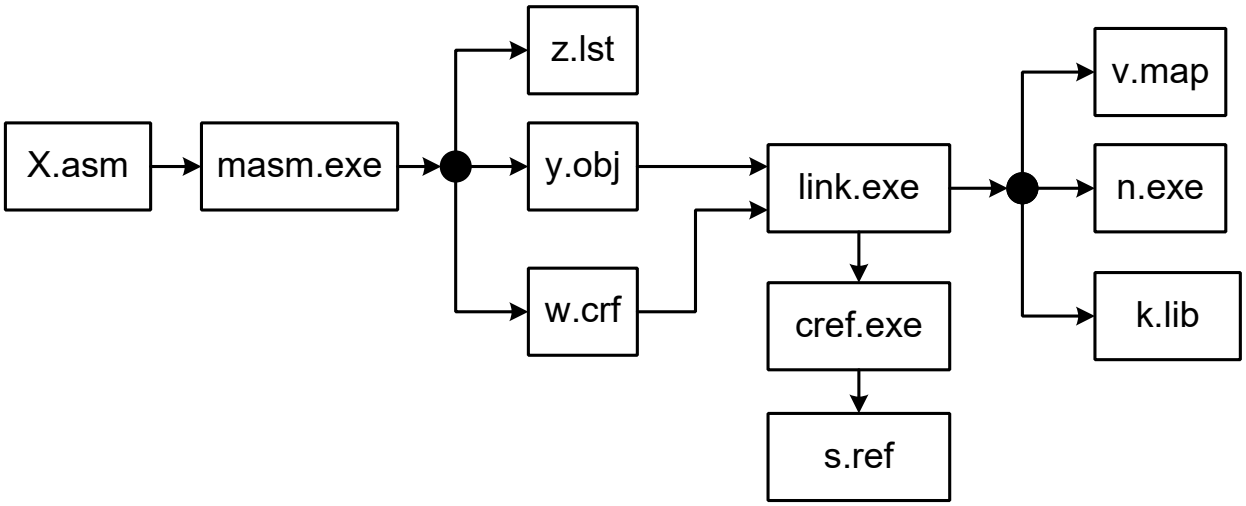

Рисунок 1.1 – Схема формування виконуваного файла

Приклад тексту на асемблері, на підставі якого потрібно створити виконуваний модуль та запустити його на виконання:

```
                title       labrab_1
                page            ,132
sts             segment     stack
                dw          64  dup (?)
vst             label           word
```

```
sts             ends
data            segment
nom_1           db      9h
nom_2           db      4ch
text1           db      'lab_rab_1',0ah,0dh,'$'
text2           db      'microprocessor type:',0ah,0dh,'$'
text_88         db      '8086/8088',10,13,'$'
text_286        db      '80286',10,13,'$'
text_386        db      '80386',10,13,'$'
text3           db      'no 8086/8088, no 80286 and no 80386'
                db      10,13 ,'$'
data            ends
code            segment
                assume  cs:code,ds:data,ss:sts
nach:           mov             ax,data
                mov             ds,ax
                mov             ax,sts
                mov             ss,ax
                lea     sp,vst
                mov             ah,nom_1
                lea     dx,text1
                int     21h
                lea     dx,text2
                int     21h
                xor     ax,ax
                push    ax
                popf
                pushf
                pop     ax
                and     ah,0f0h
                jz      m_286
                cmp             ah,0f0h
                jz      m_88
                test    ah,80h
                jnz     m_386
                lea     dx,text3
                jmp     m1
m_88:           lea     dx,text_88
```

```
                jmp     m1
m_286:          lea     dx,text_286
                jmp     m1
m_386:          lea     dx,text_386
m1:             mov     ah,nom_1
                int     21h
                mov             ah,nom_2
                int     21h
code            ends
                end     nach
```

ЛАБОРАТОРНА РОБОТА 2

НАЛАГОДЖЕННЯ І ВИКОНАННЯ ПРОГРАМИ КОРИСТУВАЧА ПІД УПРАВЛІННЯМ СИСТЕМНОГО ВІДЛАДЧИКА

Мета роботи: вивчити структуру команд процесора і навчитися використовувати їхні машинні коди. Задіяти можливості, що надаються системною програмою **debug**.

Постановка завдання: створити текстовий файл на асемблері. Отримати файл, який потрібно виконати. Навчитися використовувати команди відладчика, котрі потрібні в конкретній ситуації і в потрібній послідовності. Задіяти команди процесора, що впливають на прапори. Налагодити за допомогою програми **debug** файл, який потрібно виконати. Продемонструвати, використовуючи для цього команди відладчика, одержані результати. Зафіксувати машинні коди команд процесора в робочому зошиті.

Порядок виконання роботи:

• з машинними кодами команд CPU, призначенням і особливостями використання його регістрів, способами адресації операндів дає можливість ознайомитися системна програма ***debug***. Але спочатку треба набути навичок у використанні команд самого відладчика;

• для завантаження програми ***debug*** в пам'ять і передачі їй управління, введемо командний рядок

D: \ MASM \ debug.exe.

Після цього програма видає запит у вигляді символу "-" (дефіс) і переходить у режим очікування введення користувачем однієї з її команд. Можна отримати на екрані перелік усіх команд програми ***debug***, якщо у відповідь на її запит ввести символ "?". Усі команди відладчика починаються з латинської літери (неважливо великої або малої), за якою можна легко визнати призначення кожної команди. Нагадуємо, що натискання клавіші ***Enter*** служить ознакою кінця введення;

• команда R [ім'я-регістра] дозволяє відобразити на екрані вміст усіх регістрів МП або одного, ім'я якого зазначено в її тексті.

Приклади

R

R IP

У другому прикладі показано, як відобразити або змінити вміст тільки одного регістра, а саме IP. Нове значення можна ввести після появи на екрані символу ":". Щоб залишити наявне значення, досить натиснути клавішу ***Enter***;

• команда A (***Assemble***) дозволяє нам вводити команди до процесора в символьному вигляді. Треба тільки вказати зсув, який відповідає початковій адресі завантаження регістра (вміст регістра CS береться до уваги автоматично).

Приклад

```
A 100
MOV AL, 25
MOV BL, 32
ADD AL, BL
RET
```

Щоб вийти з режиму виконання команди A, треба натиснути клавішу ***Enter***;

• команда U (***Unassemble***) відображає одночасно машинні коди і символьне подання команд процесора. Треба в команді вказати початкову адресу і можливе зміщення кінцевого адреси тих байтів у пам'яті, вміст яких у вигляді команд процесора нас зараз цікавить. Для початкової адреси також можна вказати

тільки зсув (вміст регістра CS буде враховано автоматично). Можна в команді U не вказувати адресну частину, якщо після попереднього використання цієї команди доцільно її використовувати знову. Треба звернути увагу на те, що після завантаження програми ***debug*** в пам'ять ми можемо використовувати вже завантажене в регістри процесора значення. Команда R дозволяє ці значення побачити на екрані і, якщо треба, змінити. Тому там, де потрібно, в тексті команди можна замість значення записати назву регістра (наприклад: cs, ds,...).

Приклади

U 15
U CS: 0,10
U
U 0
U CS: 100

В останньому прикладі використання команди U нам дає можливість побачити на екрані сукупність команд процесора, розміщених у пам'яті за допомогою команди A відладчика. Якщо в цій команді U вказати також кінцеве значення зсуву, то відображення можна закінчити командою ***RET***;

• команда E (***Enter***) дозволяє ввести програму в машинних кодах або дані безпосередньо в пам'ять машини. Для цього в команді після букви E спочатку треба вказати початкову адресу розміщення програми, а потім машинні коди команд або значення даних у шістнадцятковому вигляді.

Приклади

E CS: 150 B8 23 01 25
E CS: 156 8B D8 03 D8 8B CB
E CS: 15C 28 C8 2B C0 90 C3

(Це все можна ввести також за допомогою тільки однієї команди E).

E DS: 00 23 01 25 00 00 00
E DS: 06 2A 2A 2A

Команда U CS: 150,161 забезпечить у даному випадку символьне подання команд процесора, машинні коди яких записані в зазначених командах E;

• команда D (***Dump***) дозволяє відтворити на екрані вміст одного або відразу декількох байтів пам'яті.

Приклади

D ds: 4,4

D ds: 4 5

D 2AA3: 0,80

D ds: 0120

D cs: 100

D CS: 2a 2b

• після розміщення за допомогою відладчика в пам'яті програма або дані можуть бути збережені на диску. Для цього, використовуючи команду R відладчика, спочатку в регістр CX занести 16-й код числа, що вказує кількість байтів, зайнятих введеної програмою або даними. Потім файлу, який створюється, присвоїти ім'я, використовуючи для цього команду

N імя_файла.com

У команді N (***Name***) перед іменем файла треба вказати шлях у каталозі, якщо запис буде здійснюватися не на поточний накопичувач. Команда W (***Write***) забезпечує передачу інформації з пам'яті на диск. Після виконання команди W на екрані з'являється повідомлення:

Writing nnnn bytes;

• для завантаження програми, яка налагоджується, n.exe в пам'ять і виконання її під керуванням програми ***debug***, вводиться командний рядок

D: \ MASM \ debug.exe n.exe

Після цього доцільно виконати команду R відладчика, а потім T або G в залежності від ступеня готовності користувача сприймати результати після використання цих команд. Команда T [кількість_команд] дозволяє виконати тільки одну або вказану в її тексті кількість команд з одночасним відтворенням вмісту всіх регістрів і відомостей про наступну команду програми після виконання кожної команди T;

• команда G [зсув] дозволяє виконати програму, яка налагоджується, із зупинкою перед командою, зсув адреси якої в кодовому сегменті (тобто вміст регістра IP) зазначено в тексті команди G (GO). У момент зупинки відтворюються вміст усіх регістрів і відомості про команду, перед виконанням якої сталася зупинка. Продовжити виконання програми можна після введення

команди T або чергової команди G. Але треба мати на увазі, що використання команди G без зазначення в ній компоненти "зсув" призводить до виконання всіх команд, що залишилися в програмі, і виходу з неї. Або при цьому може статися "зависання" системи, якщо в програмі є помилки, які створюють таку ситуацію. Тому не варто з допомогою команди G перевіряти великі ділянки програми, яка налагоджується. Точки (тобто зміщення) для зупинок треба вибрати з файла *Lst* з урахуванням логіки реалізованого в програмі алгоритму.

Приклад тексту на асемблері

```
                title   labrab_2
                page    ,132
stc             segment stack
                db      128 dup (?)
tos             label   word
stc             ends
dats            segment
text1           db      'begin labrab_2',0ah,0dh,'$'
text2           db      'end labrab_2',0ah,0dh,'$'
A               dw      3 dup (-5),7,-10,15,2 dup (0,-3)
B               dw      10 dup (?)
kol             dw      10
dats            ends
cods            segment
                assume  cs:cods,ds:dats,ss:stc
beg:            mov             ax,stc
                mov             ss,ax
                mov     sp,offset tos
                mov             ax,dats
                mov     ds,ax
                mov             ah,9h
                lea     dx,text1
                int     21h
                mov     cx,kol
                xor     si,si
                mov     di,18
l1:             mov     ax,A[si]
```

```
                mov     B[di],ax
                add     si,2
                sub     di,2
                loop    l1
                mov     ah,9h
                lea     dx,text2
                int     21h
                mov     ah,4ch
                int     21h
cods            ends
                end     beg
```

ЛАБОРАТОРНА РОБОТА 3

РОБОТА ЗІ СТЕКОМ

Мета роботи: навчитися використовувати стек і задіяти регістри процесора, також його команди і директиви асемблера, що забезпечують роботу зі стеком.

Постановка завдання: створити текстовий файл на асемблері з використанням команд, що звертаються до стека. Отримати виконуваний файл. Використовуючи відладчик, створену програму завантажити в пам'ять і виконати в режимі трасування, фіксуючи при цьому моменти, які пов'язані з роботою зі стеком.

Порядок виконання роботи:

- створити текстовий файл. Отримати виконуваний файл;
- використовуючи файл *.map, визначити обсяг пам'яті, який виділено для всіх сегментів даної програми;
- за файлом *.Lst* визначити зміщення для тих команд, перед виконанням яких необхідно організувати зупинку, щоб зафіксувати факт звернення до стека. Виписати також значення зміщень, на які вказують на задіяні в стековому сегменті імена;
- за допомогою програми ***debug*** завантажити створений виконуваний файл у пам'ять. Використовуючи команду R

відладчика визначити спочатку вміст регістрів SS, SP, BP, а також регістра прапорів;

- реалізувати зупинку після першої команди ***int 21h***. Виявити, які зміни відбулися в стековому сегменті. Після її виконання вміст пам'яті в стековому сегменті зміниться. Команда ***int*** заносить у стек вміст регістра прапорів і адреса наступної команди. Команда ***iret*** (яка виконується останньою в будь-якій програмі обробки переривання) зі стека витягує у зворотній послідовності те, що туди занесла команда ***int***. Обидві команди автоматично змінюють вміст регістра SP. Але значення, занесені в пам'ять командою ***int***, там залишаються. Їх можна замінити іншими при черговому зверненні до стека. Перед виконанням записати в робочий зошит вміст регістрів CS, IP, SP і регістра прапорів;
- виконати в покроковому режимі як чергові 2 команди завантаженої програми, так і програму обробки переривання ***12h***. Простежити динаміку зміни вмісту регістрів CS, IP, SP і регістра прапорів. Відобразити в робочому зошиті, які зміни відбулися в стековому сегменті. При виконанні мати на увазі, що команда ***int*** не тільки заносить у стек указані значення, але й скидає прапори IF і TF, а в пару регістрів CS: IP вписує адресу точки входу в програму обробки переривання. Команда ***iret*** в ці регістри повертає адресу наступної після ***int*** команди і відновлює поточний вміст регістра прапорів;
- виконати в покроковому режимі інші ділянки завантаженої програми, котрі виділено коментарем. Команда ***call*** заносить у стек адресу наступної після неї команди. Команда ***ret*** ці значення витягує із стека і завантажує їх у вказані регістри. Якщо в команді ***ret*** є операнд - число, то після вилучення зі стека адреси повернення, вміст регістра SP буде збільшено на це число. Так можна видалити з стека параметри, які занесено туди перед викликом процедури. Коли вміст регістра SP стане рівним 0, наступна команда ***push*** занесе інформацію в стек за адресою з вмістом регістра SP, рівним FFFE. Ніякі попереджуючі повідомлення при цьому не з'являються. Якщо з тексту програми виключити стековий сегмент, за замовчуванням буде задіяний стек об'ємом в 64К байтів, і початковим вмістом регістра SP буде FFFF. Регістр SP можна використовувати, щоб звернутися тільки

до вершини стека. Якщо є необхідність звернутися до вмісту будь-яких інших, розміщених у стековому сегменті, адрес пам'яті, треба використовувати регістр BP. Перед викликом процедури або в ній треба в стек занести вміст тих регістрів, які будуть у процедурі задіяні. Потім занесені в стек значення повинні бути повернені у відповідні регістри. У процесорі 80286 є команди ***pusha*** і ***popa***. Перша заносить у стек, друга витягує із стека вміст відразу всіх регістрів.

- Приклад тексту на асемблері

```
                .286
                title       labrab_3
                page        ,132
sts             segment     stack
                dw          8 dup (?)
vst             label       word
                dw          12 dup (?)
                db          3 dup ('stack   ')
tos             label       word
sts             ends
data            segment
text1           db          'begin labrab_3',0ah,0dh,'$'
text2           db          'end labrab_3',0ah,0dh,'$'
data            ends
code            segment
                assume      cs:code,ds:data,ss:sts
nach:           mov             ax,data
                mov             ds,ax
                mov         ax,sts
                mov         ss,ax
                lea         sp,tos
                mov         ah,9h
                lea         dx,text1
                int         21h
                mov         sp,offset tos-24
                int         12h
                lea         sp,vst-4
```

```
                call    pr1
                call    far ptr pr2
                lea     sp,tos
                pusha
                mov             ah,9h
                lea     dx,text2
                int     21h
                popa
                mov     ah,4ch
                int     21h
pr1             proc            near
                mov     cx,5
                mov     ax,'*\'
m1:             push            ax
                loop    m1
                mov     ax,'/\'
                push    ax
                mov     bp,sp
                add     bp,12
                mov     sp,bp
                ret     4
pr1             endp
pr2             proc    far
                mov             bp,sp
                mov     bx,[bp-10]
                mov     dh,byte ptr [bp+28]
                add     bp,31
                mov             dl,byte ptr [bp]
                ret
pr2             endp
                code    ends
                end     nach
```

ЛАБОРАТОРНА РОБОТА 4

РОБОТА З МАСИВАМИ МОВОЮ АСЕМБЛЕРА

Мета роботи: набування і закріплення практичних навичок у складанні і налагодженні програм мовою асемблера, які обробляють дані у вигляді масивів.

Постановка завдання: розробити і налагодити програму розв’язання задачі з використанням масивів мовою асемблера.

Порядок виконання роботи:

- відповідно до номера прізвища у журналі вибрати індивідуальне завдання та ознайомитися з постановкою завдання. Розробити алгоритм розв’язання задачі. Для задання значень елементів масивів слід використовувати директиви визначення та ініціалізації даних. Вибрати змінні. Побудувати графічну схему алгоритму розв’язання задачі. При побудові схеми алгоритму слід використовувати вибрані імена змінних. Скласти текст програми. Схему алгоритму і текст програми слід узгодити з викладачем;
- за допомогою NORTON COMMANDER (NC.EXE) і вбудованого в нього символьного редактора створити файл з розширенням. ASM. Ввести текст програми з клавіатури і зберегти його у створеному файлі. Ім'я файла визначається відповідно до буквеного індексу групи, номера лабораторної роботи та номера індивідуального завдання;
- використовуючи MASM.EXE виконати асемблювання програми зі створенням файла лістингу (розширення .LST) та об'єктного файла (розширення .OBJ). Провести аналіз і виправлення виявлених синтаксичних помилок у тексті програми;
- використовуючи LINK.EXE для об'єктного файла (розширення .OBJ) отримати виконавчий файл з розширенням .EXE, що містить завантажувальний модуль. Провести аналіз і виправлення виявлених помилок за допомогою символьного редактора;
- запустити відладчик DEBUG.EXE і використовуючи його команди (завдання імені файла, завантаження файла, відображення даних, виконання програми, трасування, вихід) переконатися в правильності розв’язання задачі. При виявленні помилок слід усунути їх за допомогою символьного редактора.

Після усунення всіх помилок слід продемонструвати викладачеві остаточний варіант програми, її роботу та отримані результати;

- вставити дискету і зберегти файл з текстом програми на дискеті. Після цього необхідно видалити всі файли, створені в процесі виконання лабораторної роботи.

Приклад розв'язання задачі

Індивідуальне завдання: мовj. асемблера розробити і налагодити програму розв'язання вищенаведеної задачі (варіант 0). Задано масив A з N = 20 елементів (констант типу WORD). Його структура:

```
-------------------------------------------------------
| X (1) | Y (1) | X (2) | Y (2) |. . . . | X (I) | Y (I) |
-------------------------------------------------------
A (1) A (2) A (3) A (4). . . . A (N)
```

Сформувати масив B з елементів X(i) і визначити їхню суму і кількість за умови збігу бітів 2,5 і 11.

Розроблення алгоритму розв'язання задачі:

- вибір змінних. Змінні, які використовуються у програмі при розв'язанні задачі, наведені в таблиці 4.1.

Таблиця 4.1 – Список змінних

Ім'я змінної	Призначення змінної
a	Початок масиву A
n	Кількість пар елементів у масиві A
b	Початок масиву B
k	Кількість елементів у масиві B
s	Сума елементів масиву B
m	Маска для виділення 2, 5 і 11 бітів
s1	Початок рядка повідомлення про запуск програми
s2	Початок рядка повідомлення про завершення програми
s3	Початок рядка повідомлення про збіг бітів

- алгоритм розв’язання задачі складається з нижченаведених блоків.

Блок 2. Визначення та ініціалізація використовуваних даних.

Блок 3. Виведення повідомлення про запуск завдання.

Блок 4. Обнуління лічильника елементів у вихідному масиві.

Блок 5. Обнуління лічильника та суми елементів вихідного масиву.

Блок 6. Занесення до лічильника циклів кількості пар елементів у результуючому масиві.

Блок 7. Побітна операція нееквівалентності над парою елементів вихідного масиву.

Блок 8. Виділення заданих бітів шляхом побітної операції кон'юнкції з маскою.

Блок 9. Аналіз збігу заданих бітів і перехід при незбігу до блока 14.

Блок 10. Занесення виявленого елемента вихідного масиву до кінцевого масиву.

Блок 11. Інкремент лічильника елементів вихідного масиву.

Блок 12. Підсумовування виявленого елемента вихідного масиву.

Блок 13. Виведення повідомлення про збіг бітів.

Блок 14. Вибір наступної пари елементів вихідного масиву.

Блок 15. Декремент лічильника циклів.

Блок 16. Перевірка обнуління лічильника циклів і перехід при ненулювому значенні до блока 7.

Блок 17. Виведення повідомлення про завершення завдання.

Блок 18. Завершення завдання і вихід в операційну систему.

- текст програми

```
data        segment
n           dw      10
a           dw      10 dup (0937h,0824h)
b           dw      10 dup (0)
k           dw      0
```

```
s               dw      0
m               dw      0824h
s1              db      'start',10,13,'$'
s2              db      'stop',10,13,'$'
s3              db      'pause',10,13,'$'
data            ends
code            segment
                assume  cs:code,ds:data
m0:             mov     ax,data
                mov     ds,ax
                mov     ah,9h
                mov     dx,offset s1
                int     21h
                xor     ax,ax
                mov     si,ax
                mov     di,ax
                mov     k,ax
                mov     s,ax
                mov     dx,ax
                mov     bx,ax
                mov     cx,n
m1:             mov     ax,a[si]
                mov     dx,a[si+2]
                xor     dx,ax
                and     dx,m
                jnz     m2
                mov     b[di],ax
                add     di,2
                inc     k
                add     s,ax
                mov     ah,9h
                mov     dx,offset s3
                int     21h
m2:             add     si,4
                loop    m1
                mov     ah,9h
                mov     dx,offset s2
                int     21h
```

```
                mov     ah,4ch
                int     21h
code            ends
                end     m0
```

Результати роботи програми. Стан області даних до виконання програми:

```
-D ds: 100 15f
1B26: 0100 0A 00 37 09 24 08 37 09-24 08 37 09 24 08 37 09
1B26: 0110 24 08 37 09 24 08 37 09-24 08 37 09 24 08 37 09
1B26: 0120 24 08 37 09 24 08 37 09-24 08 00 00 00 00 00 00
1B26: 0130 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1B26: 0140 00 00 24 08 73 74 61 72-74 0A 0D 24 73 74 6F 70
1B26: 0150 0A 0D 24 70 61 75 73 65-0A 0D 24 00 00 00 00 00
```

Контроль процесу виконання програми з використанням відладчика і застосуванням трасуючих повідомлень

```
-G
start
pause
pause
pause
pause
pause
pause
pause
pause
pause
pause
pause
stop

Program terminated normally
```

Стан області даних після виконання програми

```
-d ds:100 15f
1B26:0100 0A 00 37 09 24 08 37 09-24 08 37 09 24 08 37 09
```

1B26:0110 24 08 37 09 24 08 37 09-24 08 37 09 24 08 37 09
1B26:0120 24 08 37 09 24 08 37 09-24 08 37 09 37 09 37 09
1B26:0130 37 09 37 09 37 09 37 09-37 09 37 09 37 09 0A 00
1B26:0140 26 5C 24 08 73 74 61 72-74 0A 0D 24 73 74 6F 70
1B26:0150 0A 0D 24 70 61 75 73 65-0A 0D 24 00 00 00 00 00

СПИСОК ЛІТЕРАТУРИ

1 Скэнлон Л. Персональные ЭВМ IBM PC и XT. Программирование на языке ассемблера: Пер. с англ. – М.: Радио и связь, 1989. – 336 с.

2 Брэдли Д. Программирование на языке ассемблера для персональной ЭВМ фирмы IBM: Пер. с англ. – М.: Радио и связь, 1988. – 448 с.

3 Абель П. Язык Ассемблера для IBM PC и программирования: Пер. с англ. – М.: Высш. шк., 1992. – 447 с.

4 Морс С.П., Алберт Д.Д. Архитектура микропроцессора 80286: Пер. с англ. – М.: Радио и связь, 1990. – 304 с.

5 Хвощ С.Т., Варлинский Н.Н., Попов Е.А. Микропроцессоры и микроЭВМ в системах автоматического управления: Справочник. – Л.: Машиностроение, 1987. – 640 с.

6 Микропроцессорный комплект К1810: Структура, программирование, применение: Справочная книга / Ю.М. Казаринов. – М.: Высш. шк., 1990. – 269 с.