Лекція-4. Написання програм

- Після введення на диск початкової програми під ім'ям Ім’я.ASM необхідно виконати два основні кроки, перш ніж програму можна буде виконати.

- 1. Спочатку необхідно асемблювати програму,

- 2. Потім виконати компоновку.

Підготовка програми до виконання

- Крок асемблювання включає трансляцію початкового коду в машинний об'єктний код і генерацію OBJ-модуля. Крок компановки включає перетворення OBJ-модуля в EXE (COM) модуль, що містить машинний код. Програма LINK, що знаходиться на диску DOS, виконує наступне:
- 1. Завершує формування в OBJ-модулі адрес, які залишилися невизначеними після асемблювання.
- 2. Компонує, якщо необхідно, більш одного окремо асемблюючого модуля в одну завантажувальну програму;
- 3. Ініціалізує EXE(COM)-модуль командами завантаження для виконання.

Трансляція програми до виконання

- В процесі трансляції початкової програми асемблер робить два перегляди (проходи) початкового тексту. Однією з основних причин цього є посилання вперед, що відбувається у тому випадку, коли в деякій команді кодується мітка, значення якої ще не визначене асемблером.

- У першому проході асемблер проглядає всю початкову програму і будує таблицю ідентифікаторів, використовуваних в програмі, тобто імен полів даних і міток програми і їх відносних адрес в програмі. У першому проході обраховується об'єм об'єктного коду, але сам об'єктний код не генерується.

Трансляція програми до виконання

- У другому проході асемблер використовує таблицю ідентифікаторів, побудовану в першому проході. Оскільки тепер уже відомі довжини і відносні адреси всіх полів даних і команд, то асемблер може згенерувати об'єктний код для кожної команди. Асемблер створює, якщо потрібен, файли: OBJ, LST і CRF.

ВІДМІННОСТІ МІЖ ПРОГРАМАМИ В EXE і COM-файлах

Розмір програми. EXE-програма може мати будь-який розмір, тоді як COM-файл обмежений розміром одного сегменту і не перевищує 64К. COM-файл завжди менше, ніж відповідний EXE-файл; одна з причин цього - відсутність в COM-файлі 512-байтового початкового блоку EXE-файлу.

Сегмент стеку. У EXE-програмі визначається сегмент стека, тоді як COM-програма генерує стек автоматично. Таким чином при створенні асемблерної програми, яка буде перетворена в COM-файл, стек повинен бути опущений.

Сегмент даних. У EXE програмі звичайно визначається сегмент даних, а регістр DS ініціалізується адресою цього сегменту. У COM-програмі всі дані повинні бути визначені в сегменті коду.

ВІДМІННОСТІ МІЖ ПРОГРАМАМИ В EXE і COM-файлах

- І н і ц і а л і з а ц і я. EXE-програма записує нульове слово в стек і ініціалізує регістр DS. Оскільки COM-програма не має ні стека, ні сегменту даних, то ці кроки відсутні. Коли COM-програма починає працювати, всі сегментні регістри містять адресу префікса програмного сегменту (PSP), - 256-байтового блоку, який резервується операційною системою DOS безпосередньо перед COM або EXE програмою в пам'яті. Оскільки адресація починається з шістнадцятирічного зсуву 100 від початку PSP, то в програмі після оператора SEGMENT кодується директива ORG 100H.

  Якщо ж програма створюється для виконання як COM-файл, то компановщиком буде видане повідомлення: Warning: No STACK Segment (Попередження: сегмент стека не визначений) Це повідомлення можна ігнорувати, оскільки визначення стека в програмі не передбачалося.

СТЕК ДЛЯ COM-ПРОГРАМИ

Для COM-файлу DOS автоматично визначає стек і встановлює однакову загальну сегментну адресу у всіх чотирьох сегментних регістрах. Якщо для програми розмір сегменту в 64К є достатнім, то DOS встановлює в регістрі SP адресу кінця сегменту -FFFE. Це буде вершина стеку. Якщо 64К байтовий сегмент не має досить місця для стека, то DOS встановлює стек в кінці пам'яті. У обох випадках DOS записує потім в стек нульове слово. Можливість використовування стека залежить від розміру програми і обмеженості пам'яті. За допомогою команди DIR можна визначити розмір файлу і обчислити необхідний простір для стека. Всі невеликі програми в цій книзі в основному расчитани на COM-формат.

Написання програм

- Почнемо з написання COM програми.
- Для написання програми можна використовувати будь-який текстовий редактор (наприклад, Блокнот), при цьому при написанні програми пригадаємо основні елементи структури програми:

1. ***Задамо модель пам'яті (хоча можна і без неї)***

2. В кодовому сегменті запишемо ряд команд

3. В кінці програми опишемо змінні

Написання програм

Введемо в файлі код:

```
model tiny
.code
.startup
    mov dx,offset testm
    mov ah,09h
    int 21h
    ret
testm  db "Hello, world!!! $"
    end
```

Збережемо файл Lab1.asm

Написання програм

- В командному рядку виконаємо такі команди (для Turbo Assembler (TASM))

> tasm Lab1.asm

> tlink Lab1.obj -t

> lab1.com

Написання програм

- **Для Macro Assembler версии 5.00 – 5.10 (MASM 5.00 – 5.10))**

> masm Lab1.asm

> link Lab1.obj /t

> lab1.com

- В результаті виконання на екрані з’явиться рядок:
- Hello, world!!!

Приклад командного файлу

tasm.exe %1

tlink.exe %1 %2

%1

Де %1 – перший параметр (набір символів без пробілів) з рядку запуску

%2 – другий параметр з рядку запуску

Приклад запуску

```
D:\BCPP\BIN>sss.bat lab1c -t

D:\BCPP\BIN>tasm.exe lab1c
Turbo Assembler  Version 3.1  Copyright (c) 1988, 1992 Borland International

Assembling file:   lab1c.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  423k


D:\BCPP\BIN>tlink.exe lab1c -t
Turbo Link  Version 5.1 Copyright (c) 1992 Borland International

D:\BCPP\BIN>lab1c
yjghjfsgjfsg
Entered string:
yjghjfsgjfsg
```

Розбирання програми

Введемо в файлі код:

```
model tiny
.code
.startup
    mov dx,offset tes
    mov ah,09h
    int 21h
    ret
testm  db "Hello, worl
    end
```

Директива, яка забезпечує код ініціалізації, поточній моделі і операційної системи. Вона відзначає також початок програми. Ця директива має наступний синтаксис: STARTUPCODE або .STARTUP.

дозволяє одержати значення зсуву виразу у байтах відносно початку того сегмента, у якому вираження визначене.

Розбирання програми

- int 21h - Це переривання служить головним входом більшості функцій DOS.
- Програма, що запрошує сервіс DOS, повинна підготувати всю необхідну інформацію в регістрах і управляючих блоках, вказати в регістрі AH номер бажаної функції DOS і потім викликати переривання INT 21H.

- Роздивимось основні функції і вимоги до їх використання

Ряд основних функцій переривання 21h

- Функція DOS 00H: завершити програму
- Функція DOS 01H: введення з клавіатури
- Функція DOS 02H: виведення на дисплей
- Функція DOS 05H: виведення на принтер
- Функція DOS 06H: Обмін з консоллю
- Функція DOS 07H: Нефільтруюче консольне введення без луни
- Функція DOS 08H: Консольне введення без луни
- Функція DOS 09H: Видати рядок
- Функція DOS 0aH: буферизуюче введення рядка
- Функція DOS 0bH: перевірити статус введення
- Функція DOS 0cH: введення з очищенням
- Функція DOS 0dH: Скинути диск
- Функція DOS 1bH: дати інформацію FAT (поточний диск)
- Функція DOS 1cH: дати інформацію FAT (будь-який диск)

Ряд основних функцій переривання 21h

- Функція DOS 21H: читати запис довільного файлу
- Функція DOS 22H: писати запис довільного файлу
- Функція DOS 23H: дати розмір файлу через FCB
- Функція DOS 27H: читати блок довільного файлу
- Функція DOS 28H: писати блок довільного файлу
- Функція DOS 29H: Розібрати ім'я файлу
- Функція DOS 2aH: дати дату DOS
- Функція DOS 2bH: встановити дату DOS
- Функція DOS 2cH: дати час DOS
- Функція DOS 2dH: встановити час DOS
- Функція DOS 2eH: встановити/скинути перемикач верифікації
- Функція DOS 30H: дати номер версії DOS
- Функція DOS 31H: завершитися і залишитися резидентним -- KEEP

Ряд основних функцій переривання 21h

- Функція DOS 36H: дати вільну пам'ять диска
- Функція DOS 39H: створити новий зміст -- MKDIR
- Функція DOS 3aH: Видалити зміст -- RMDIR
- Функція DOS 3bH: встановити замовчуваний зміст DOS -- CHDIR
- Функція DOS 3cH: створити описувач файлу
- Функція DOS 3dH: відкрити описувач файлу
- Функція DOS 3eH: Закрити описувач файлу
- Функція DOS 3fH: читати файл через описувач
- Функція DOS 40H: писати у файл через описувач
- Функція DOS 41H: Видалити файл
- Функція DOS 42H: встановити покажчик файлу -- LSEEK
- Функція Функція DOS 43H: встановити/опитати атрибут файлу -- CHMOD

Ряд основних функцій переривання 21h

- Функція DOS 44H: управління введенням-висновком пристрою -- IOCTL
- Функція DOS 45H: Дублювати описувач файлу -- DUP
- Функція DOS 46H: перепризначувати описувач -- FORCDUP
- Функція DOS 47H: дати замовчуваний зміст DOS
- Функція DOS 48H: розподілити пам'ять (дати розмір пам'яті)
- Функція DOS 49H: Звільнити блок розподіленої пам'яті
- Функція DOS 4aH: Стиснути або розширити блок пам'яті
- Функція DOS 4bH: виконати або завантажити програму -- EXEC
- Функція DOS 4cH: завершити програму -- EXIT
- Функція DOS 4dH: дати код виходу програми -- WAIT
- Функція DOS 4eH: Знайти 1-й співпадаючий файл
- Функція DOS 4fH: Знайти наступний співпадаючий файл
- Функція DOS 54H: дати перемикач верифікації DOS

Ряд функцій переривання 21h

- Функція DOS 56H: Перейменувати/перемістити файл
- Функція DOS 57H: встановити/опитати час/дату файлу
- Функція DOS 59H: дати розширену інформацію про помилку
- Функція DOS 5aH: створити унікальний тимчасовий файл
- Функція DOS 5bH: створити новий файл
- Функція DOS 5cH: блокувати/розблоковувати доступ до файлу
- Функція DOS 5eH: різні мережеві функції
- Функція DOS 5fH: перепризначення пристроїв в мережі
- Функція DOS 62H: дати адресу префікса програмного сегменту

Описання дії і вимог основних функцій переривання 21h

- **DOS Fn 00H: завершити програму**
- **Вхід AH = 00H**
- **CS = сегмент PSP процесу, що завершується**
- **Вихід = (непридатний)**
- **Опис:** передає управління на вектор завершення в PSP (виходить в батьківський процес). Ідентична функції INT 20H Terminate. регістр CS повинен указувати на PSP. відновлює вектори переривань DOS 22H-24H (завершення, Ctrl-Break і Критична помилка), встановлюючи значення, збережені в батьківському PSP. виконує скидання файлових буферів. файли повинні бути заздалегідь закриті, якщо їх довжина змінилася.
- **Зауваження:** Простіше і більш акуратно - використовувати функцію DOS Fn 4cH Exit.

Описання дії і вимог основних функцій переривання 21h

- **DOS Fn 01H:ввод з клавіатури**
- **Вхід AH = 01H**
- **Вихід AL = символ, одержаний із стандартного введення**
- **Опис:**
- Прочитує (чекає) символ із стандартного вхідного пристрою. Відображає цей символ на стандартний вихідний пристрій (луна). при розпізнаванні Ctrl-Break виконується INT 23H.
- **Зауваження:**
- введення розширених клавіш ASCII (F1-F12, PgUp, курсор і т.п.) вимагає двох звернень до цієї функції. перший виклик повертає AL=0. Другий виклик повертає в AL розширений код ASCII.

Описання дії і вимог основних функцій переривання 21h

- **DOS Fn 02H: виведення на дисплей**
- **Вхід**
- **AH = 02H**
- **DL = символ, що виводиться на стандартний виведення**
- **Вихід**
- **Опис:**
- Посилає символ з DL на стандартний виведення. обробляє символ Backspace (ASCII 8), переміщаючи курсор вліво на одну позицію і залишаючи його в новій позиції. при виявленні Ctrl-Break виконується INT 23H.

Описання дії і вимог основних функцій переривання 21h

- **DOS Fn 05H: виведення на принтер**
- **Вхід**
- **AH = 05H**
- **DL = символ, записуваний на стандартний принтер**
- **Вихід**
- **Опис:**
- Посилає символ в DL на стандартний пристрій принтера, звичне LPT1.
- **Зауваження:**
- команда DOS MODE може перенаправити цей виведення в послідовний порт.

Описання дії і вимог основних функцій переривання 21h

- **DOS Fn 09H: Видати рядок на дисплей**
- **Вхід AH = 09H**
- **DS:DX = адреса рядка, що закінчується символом '$' (ASCII 24H)**
- **Вихід**
- **Опис:**
- рядок, виключаючи завершальний її символ '$', посилається на стандартний виведення. символи Backspace обробляються як у функції 02H Display Char. звичайно, щоб перейти на новий рядок, включають в текст пару CR/LF (ASCII 13H і ASCII 0aH). рядки, що містять '$', можна видати через 40H Write Handle (BX=0).

Описання дії і вимог основних функцій переривання 21h

- **DOS Fn 0aH: введення рядка в буфер**
- **Вхід AH = 0aH**
- **DS:DX = адреса вхідного буфера (дивися нижче)**
- **Вихід = буфер містить введення, що закінчується символом CR (ASCII 0dH)**
- **Опис:** при вході буфер за адресою DS:DX повинен бути оформлений так:

```
+---+---+---+---+---+---+- - -
|max| ? | ?   ?   ?   ?   ?
+---+---+---+---+---+---+ -  -
```

MAX - максимально допустима довжина введення (від 1 до 254)

при виході буфер заповнений даними таким чином:

Описання дії і вимог основних функцій переривання 21h

- +---+---+---+---+---+---+- - -

  |max|len| T E X T 0dH LEN - дійсна довжина даних

  +---+---+---+---+---+---+- - без завершального CR (тут - 04H).
- символи прочитуються із стандартного введення аж до CR (ASCII 0dH) або до досягнення довжини MAX-1. якщо досягнутий MAX-1, включається консольний дзвінок для кожного чергового символу, поки не буде введене повернення каретки CR (натиснення Enter).
- Другий байт буфера заповнюється дійсного завдовжки введеного рядка, не рахуючи завершального CR. останній символ в буфері - завжди CR (який не зарахований в байті довжини). символи в буфері (включаючи LEN) у момент виклику використовуються як "шаблон".

Описання дії і вимог основних функцій переривання 21h

- **DOS Fn 0bH: перевірити статус введення**
- **Вхід AH = 0bH**
- **Вихід AL = 0ffH, якщо символ доступний із стандартного введення**
- **Опис:** перевіряє стан стандартного введення. при розпізнаванні Ctrl-Break виконується INT 23H. Замечення: використовуйте перед функціями 01H 07H і 08H, щоб уникнути очікування натиснення клавіші.
- Ця функція дає простий неруйнуючий спосіб перевірки Ctrl-Break в процесі довгих обчислень або іншої обробки, що звичайно не вимагає введення. це дозволяє вам знімати рахунок по натисненню Ctrl-Break.

Описання дії і вимог основних функцій переривання 21h

- **DOS Fn 4cH: завершити програму -- EXIT**
- **Вхід AH = 4cH**
- **AL = код виходу**
- **Вихід = (непридатний)**
- **Опис:** повертає управління від породженого процесу його батькові, встановлюючи код виходу, який можна опитати функцією 4dH WAIT. Управління передається за адресою завершення в PSP програми, що завершується. вектори Ctrl-Break і Critical Error відновлюються до старих адрес, сохраненнми в батьківському PSP.
- **Зауваження:** значення ERRORLEVEL (використовуване в пакетних файлах DOS) можна використовувати для перевірки коду виходу самої останньої програми.

Розбирання програми

- Введемо в файлі код:

```
model tiny
.code
.startup
    mov dx,offset tm
    mov ah,0ah
    int 21h
    mov dx,offset testm
    mov ah,09h
    int 21h
    mov dx,offset tm
    add dx,2h
    mov ah,09h
    int 21h
    ret
tm  db 255,255,255 dup("$")
testm  db "Hello, world!!! $"
    end
```

Розбирання програми

```
model tiny
.code
.startup
     mov dx, offset Sos ;
    встановлюємо покажчик на
    строку
   mov ah,0Ah
    int 21h ; введення строки
   mov ah,0
   mov bx,dx
   mov cx,[bx + 1] ;встановлюємо
    лічильник
lst: ; початок циклу
     mov di,cx
     cmp Sos[di+1],60h ; перевірка,
    чи літера маленька
jl next ; якщо літера велика – йдемо далі
   sub Sos[di+1],20h
; якщо маленька –змінюємо її
   next:
      loop lst ; перехід на початок циклу
      mov dx, offset Sos + 2
; встан. вказівник на рядок
      mov ah,9
      int 21h ; виведення строки
      ret
   Sos     db 20 dup($) ; змінна-рядок
      end
```