МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Кафедра "Обчислювальної техніки та програмування"

АЛЬБОМ ДОКУМЕНТІВ

до курсового проекту з курсу

“Системне програмування”

Тема: “ Робота з записами мови ассемблера ”

Харків

2005

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Кафедра"Обчислювальної техніки та програмування"

Затверджую

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ /

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2005 р.

Робота з записами мови ассемблера

Специфікація

Лист затвердження

xxxxxxxxxxxxxx

Виконав

ст. гр xxxx

xxxxxxxxxxxxxxx

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2005 р.

Перевірив

xxxxxx.

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2005 р.

Харків

2005

ЗАТВЕРДЖЕНО

xxxxxxxxxxxx

Робота з записами мови ассемблера

Специфікація

xxxxxxxxxxxxxxxx

|  |  |  |
| --- | --- | --- |
| *Код* | *Найменування* | *Примітки* |
| *xxxxxx – 00 00-01* | *Специфікація* |  |
| *xxxxxx. 03011 – 13 00-01* | *Опис пргограми* |  |
| *xxxxxxx. 03011 – 12 00-01* | *Текст програми* |  |

Харків

2005

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Кафедра"Обчислювальної техніки та програмування"

Затверджую

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ /

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2005 р.

Робота з записами мови ассемблера

Опис програми

Лист затвердження

xxxxxxxxxxxxx

Виконав

ст. гр xxxxxx

xxxxxxxx.

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2005 р.

Перевірив

xxxxxxxxx.

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2005 р.

Харків

2005

ЗАТВЕРДЖЕНО

xxxxxxxxxxx

Робота з записами мови ассемблера

Опис програми

xxxxxxxxxxx

Харків

2005

**Зміст**

Вступ

Загальні відомості

Функціональне призначення

Опис логічної структури

Використовувані технічні засоби

Виклик і завантаження

Вхідні дані

Вихідні дані

Висновки

Література

**Вступ**

Запис — структурний тип даних, що складається з фіксованого числа елементів завдовжки від одного до декількох біт.

При описі запису для кожного елементу вказується його довжина в бітах і, що необов'язкове, деяке значення. 8, 16 або 32 біт.

Якщо сумарний розмір запису менше вказаних значень, то всі поля запису “притискаються” до молодших розрядів.

Використовуваннязаписів в програмі, так само, як і структур, організовується в три етапи:

Задання *шаблона запису*, тобто визначення набору бітових полів, їх довжин і, при необхідності, ініціалізація полів.

*Визначення екземпляру запису*. Так само, як і для структур, цей етап має на увазі ініціалізацію конкретної змінної типом наперед визначеної за допомогою шаблона запису.

Організація *звертання до элементів* запису.

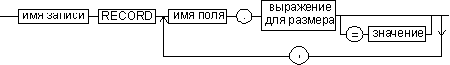
Компілятор TASM, крім стандартних засобів обробки записів, підтримує також і деякі додаткові можливостіїх обробки.

Опис запису

Опис шаблона запису має наступний синтаксис (мал. 6):

Имя\_записи RECORD <описание элементов>

Тут: <описание элементов> *є послідовністю описів окремих елементів запису згідно синтаксичній діаграмі (див. мал. 6):*

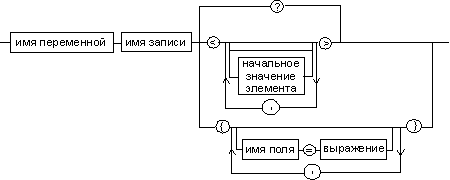


*Мал.1. Синтаксис опису шаблона запису*

При описі шаблона пам'ять не виділяється, оскільки це всього лише інформація для транслятора асемблера про структуру запису.   
Так само, як і для структур, місцеположення шаблона в програмі може бути будь-ким, але при цьому необхідно враховувати логіку роботи однопрохідного транслятора.

Визначення екземпляру запису

Для використовування шаблона запису в програмі необхідно визначити змінну з типом даного запису, для чого застосовується наступна синтаксична конструкція (мал. 7):



*Мал.2. Синтаксис опису екземпляру запису*

Аналізуючи цю синтаксичну діаграму, можна зробити висновок, що ініціалізація елементів запису здійснюється достатньо гнучко. Розглянемо декілька варіантів ініціалізації.

Якщо ініціалізувати поля не потрібен, то достатньо вказати ? при визначенні екземпляру запису:

...

iotest record

i1:1,i2:2=11,i3:1,i4:2=11,i5:2=00

...

flag iotest ?

Якщо ви складете і дослідите у відладчику тестовий приклад з даним визначенням запису, то побачите, що всі поля змінної типу запис *flag обнуляються. Це відбувається не дивлячись на те, що у визначенні запису задані початкові значення полів.*

Якщо потрібна часткова ініціалізація елементів, то вони полягають в *кутові* (< і >) або *фігурні* ({ і }) дужки.   
Відмінність тут в тому, що в кутових дужках елементи повинні бути задані в тому ж порядку, що і у визначенні запису. Якщо значення деякого елементу спывпадає з початковим, то його можна не вказувати, але обов'язково позначити його комі. Для останніх елементів коми, що йдуть підряд, можна опустити.

Наприклад, згодитися із значеннями за умовчанням можна так:

iotest record

i1:1,i2:2=11,i3:1,i4:2=11,i5:2=00

...

flag iotest <> ;согласились со значением по умолчанию

Змінити значення поля i2 можна так:

iotest record

i1:1,i2:2=11,i3:1,i4:2=11,i5:2=00

...

flag iotest <,10,> ; перевизначили i2

Застосовуючи *фігурні дужки, також можна вказати вибіркову ініціалізацію полів,але при цьому необов'язково позачити комами поля, із значеннями за умовчанням яких ми згодні:*

iotest record

i1:1,i2:2=11,i3:1,i4:2=11,i5:2=00

...

flag iotest {i2=10} ; переопределили i2, не обращая внимания на порядок;следования других компонентов записи

**1.Загальні відомості**

Проект написан на мові TASM, реалізовано у трьох файлах:

main.asm, defs.inc, funcs.inc, record.inc. Під час компіляції усі файли об’єднуються в один.

Для своєї роботи проект може використовувати операційні системи MS Dos версії 1.0, або вище, а також операційні системи

MS WINDOWS 95/98/Me/2000/XP. Не виключається робота проекту під усіма іншими операційними системами, що якимсь чином можуть емулювати сеанси роботи під операційною системою MS DOS.

**2. Функціональне призначення**

Проект може слугувати в якості навчального посібника для розуміння роботи записів та в якості прикладу одного з варіантів, такого як створення масиву, добавлення елемента, видалення елемента, змінення поля, очищення поля. Також у проекті можна наглядно побачити ввод з клавіатури і вивід результатів на екран.

Програма написана на мові ассемблера з використовуванням процедур.

Функціональне обмеження програми є те, що вона може бути реалізована тільки у файлі з розширенням COM, якщо розширення буде EXE програма працювати не буде. Більше функціональних обмежень немає.

**3. Опис логічної структури**

Початотк

clrscr

gotoxy

Writestring

Mov ah,8

Int 16h

clrscr

SetcursorPos

WritetoSceree

SetCursorPos

Writtostring

Кінець

main

Dec cx

Inc si

Mov al,byet ptr[si]

Cmp al,9h

ні

If al>9h

add al,30h

так

Add al,37h

Mov byte ptr [di],al

Inc si

Inc di

Loop

SOZD

Mov bp,sp

Push bp

Mov ax,0Ah

Mov dx,[bp+10]

Приведення шляху в asсii

Çì³íà äèñêó çã³äíî øëÿõó

Зміна директорії

“ Ошибка ввода ”

If cf<>1

“ Каталог создан “

Pop bp

Закінчення

Ret 14

HideCursr

Вівод на екран MESX, MES0, MES1,

MES2, MES3, MESX

**gotoxy**

Mov AH,08h

Int 21h

Go8bit

If AL=31h

Go16bit

If AL=32h

Go32bit

If AL=33h

**4.Використовувані технічні записи**

Проект має нормально працювати на комп’ютерах IBM PC із процесорами і8086, або більш старших моделях. Програма потребує оперативної пам’яті не більше 6 Кб, до цього треба додати пам’ять, яку займає операційна система MS DOS чи WINDOWS. Потрібен також відеоадаптер VGA, для організації діалогу з користувачем. При написанні проекту використовувався комп’ютер з процесором AMD Athlon XP 1700+, оперативною пам’ятю 256 Мб та операційною системою Windows XP у середовищі TASM.

**5. Виклик і завантаження**

Основним є файл main.com з його допомогою працює проект. Запускається він з будь-якого носія. Перед користувачем на синьому фоні з’являється меню, в якому він спочатку може виконати наступні дії, за допомогою функціональних клавіш: F2 - створити масив, або F10 – залишити програму.

Далі, користувачеві пропонується додати елемент, клавішею F3, або залишити програму. При додаванні елемента, пропонується ввести прізвищє, вік, місто. Потім, користувач може додати новий елемент, змінити елемент, видалити його, а також переміщуватись серед елементів, і звичайно вийти з програми.

**6. Вхідні дані**

Вхідними даними програми компілятора є файли main.asm, defs.inc, funcs.inc, record.inc, що розташовані в одному каталозі. Зкомпільований виконуючий файл main.com запускається і користувач працює з інформацією.

**7.Вихідні дані**

Вихідні дані програми з’являються на дисплеї в ході роботи програми.

Вихідними даними програми проекту є створення масиву, додавання елемента, видалення елемента, зміна поля та очистка поля.

**Висновки**

У ході розробки курсового проекту були вивчені навички роботи з записами мови ассемблера, зокрема, створення масивів записів, доступу до них, створення в них записів, з можливістю управління елементами і очищення поля. Розроблена програма дозволяє ввести з клавіатури запис, представлений у вигляді прізвища, імені, по батькові, назва міста і дати народження, а потім за допомогою функціональних клавіш дозволяючих:

-додати запис;

-видалити запис;

-змінити поле;

-очистити поле.

**Література**

1. Финагенов К.Г. “Основы системного программирования” изд.”Радио

и Связь”2001 год.

2. Питер Абель “Ассемблер- язык и программирование для IBM PC”

изд. Киев “HTИ” 2003 год.

3. Зубков С.В.”Ассемблер для DOS,WINDOWS И UNIX ” Москва 2004г.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Кафедра"Обчислювальної техніки та програмування"

Затверджую

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ /

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2005 р.

Робота з записами мови ассемблера

Текст програми

Лист затвердження

xxxxxxxxxxxx

Виконав

ст. гр xxxxxx

xxxxxxx

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2005 р.

Перевірив

xxxxxxxxxx.

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2005 р.

Харків

2005

ЗАТВЕРДЖЕНО

xxxxxxxxxxxx

Робота з записами мови ассемблера

Текст програми

xxxxxxxxx – 12 00-01

Харків

2005

РАЗРАБОТКА ТЕКСТА ПРОГРАММЫ

Главная программа:

LOCALS

include defs.inc

.model tiny

.386

.387

.code

org 100h

start:

call clrscr ;очитска экрана

mov al,17h

mov bx,0000h

mov cx,80\*25

call settextattr ;установка атрибутов текста

mov ax,0016h

call gotoxy

lea dx,title1

call writestring ;вывод заголовка

mov ax,0100h

call gotoxy

lea dx,separ

call writestring ;вывод разделителя

mov ax,1700h

call gotoxy

lea dx,separ

call writestring

call main ;вызов главной финкции

call clrscr

ret

main proc near

call writemenu ;вывод меню

cmp proglevel,3

jne @@l1

call viewel ;вывод текущего элемента

@@l1:

mov ax,1900h

call gotoxy

xor ax,ax

int 16h ;ожидание и чтение сканкода нажатой клавиши

cmp ah,sF10

je @@exit1

cmp proglevel,1

jne @@l2

call keyb1lvl

jmp main

@@l2:

cmp proglevel,2

jne @@l3

call keyb2lvl

jmp main

@@l3:

call keyb3lvl

jmp main

@@exit1:

ret

main endp

writemenu proc near ;вывод меню

push ax dx si

mov ax,1800h

call gotoxy

lea dx,clr

call writestring

mov ax,1800h

call gotoxy

xor ax,ax

mov al,proglevel

dec al

shl ax,1

lea si,menux

add si,ax

mov dx,[si]

call writestring

pop si dx ax

ret

writemenu endp

keyb1lvl proc near

cmp ah,sF2

jne @@ex1

mov proglevel,2

@@ex1:

ret

keyb1lvl endp

keyb2lvl proc near

cmp ah,sF3

jne @@ex1

mov ax,0

call input\_record

mov proglevel,3

mov curitem,1

mov itemscnt,1

@@ex1:

ret

keyb2lvl endp

keyb3lvl proc near

cmp ah,sF3

jne @@l1

mov ax,itemscnt

call input\_record

inc itemscnt

jmp @@ex1

@@l1:

cmp ah,sF6

jne @@l2

cmp curitem,1

je @@ex1

dec curitem

jmp @@ex1

@@l2:

cmp ah,sF7

jne @@l3

push ax

mov ax,curitem

cmp ax,itemscnt

je @@ll2

inc curitem

@@ll2:

pop ax

jmp @@ex1

@@l3:

cmp ah,sF4

jne @@l4

mov ax,curitem

dec ax

call input\_record

jmp @@ex1

@@l4:

cmp ah,sF8

jne @@l5

pusha

mov ax,curitem

dec ax

xor dx,dx

mov bl,size STUD

mul bl

add ax,offset mas

mov di,ax

mov si,ax

add si,size STUD

mov ax,itemscnt

sub ax,curitem

xor dx,dx

mul bl

mov cx,ax

rep movsb

mov ax,curitem

cmp ax,itemscnt

jne @@ll4

dec curitem

@@ll4:

dec itemscnt

cmp itemscnt,0

jne @@lll4

mov proglevel,2

call clrworkfield

@@lll4:

popa

jmp @@ex1

@@l5:

@@ex1:

ret

keyb3lvl endp

include funcs.inc

include record.inc

proglevel db 1 ; 1 - массива еще нет

; 2 - создан пустой массив

; 3 - есть не пустой массив

title1 db 'Работа с записями языка ассемблера',0

data1 db 'Элемент ',0

data2 db ' из ',0

inpName db ' Имя : ',0

inpAge db ' Возраст: ',0

inpCity db ' Город : ',0

separ db 80 dup (196),0

clr db 79 dup (' '),0

menux dw offset menu1,offset menu2,offset menu3

menu1 db ' F2 - Создание массива F10 - Выход',0

menu2 db ' F3 - Добавление элемента F10 - Выход',0

menu3 db ' F3-Добав. эл. F4-Изм. эл. F6-<< F7->>',\

F8-Удал. эл. F10-Выход',0

АП 02093

curitem dw 1

itemscnt dw 0

mas: ;начало массива записей

end start

Ввод записи с клавиатуры:

input\_record proc near ;ввод записи с клавиатуры

pusha

push ax

call clrworkfield

mov ax,0500h

call gotoxy

lea dx,inpName

call writestring

lea dx,TMP.SName

mov al,20

call readstring

mov ax,0600h

call gotoxy

lea dx,inpAge

call writestring

call readword

mov TMP.SAge,ax

mov ax,0700h

call gotoxy

lea dx,inpCity

call writestring

lea dx,TMP.SCity

mov al,20

call readstring

xor dx,dx

mov ax, size STUD

mov cx,ax

pop bx

mul bx

add ax,offset mas

mov di,ax

lea si,TMP

rep movsb

popa

ret

TMP STUD <>

input\_record endp

viewel proc near ; вывод записи на экран

pusha

mov ax,curitem

dec ax

xor dx,dx

mov bl,size STUD

mul bl

add ax,offset mas

mov si,ax

call clrworkfield

cmp itemscnt,0

je @@ex2

mov ax,0500h

call gotoxy

lea dx,inpName

call writestring

lea dx,[si].SName

call writestring

mov ax,0600h

call gotoxy

lea dx,inpAge

call writestring

mov ax,[si].SAge

call writeword

mov ax,0700h

call gotoxy

lea dx,inpCity

call writestring

lea dx,[si].SCity

call writestring

@@ex2:

mov ax,1600h

call gotoxy

lea dx,data1

call writestring

mov ax,curitem

call writeword

lea dx,data2

call writestring

mov ax,itemscnt

call writeword

@@ex1:

mov ax,1900h

call gotoxy

popa

ret

viewel endp

memzero proc near ;Обнуление памяти

;DS:DX->Mem

;CX=SIZE

push ax cx di es

xor ax,ax

mov di,dx

push ds

pop es

cld

rep stosb

pop es di cx ax

ret

memzero endp

readstring proc near ;Чтение строки с клавиатуры

;DS:DX->буффер

;AL=макс. кол-во символов

push ax dx

xor cx,cx

mov cl,al

call memzero

mov @@buf,al

mov ah,0Ah

lea dx,@@buf

int 21h

lea si,@@buf

inc si

xor cx,cx

mov cl,[si]

inc si

pop di

push dx

rep movsb

pop dx ax

ret

@@buf db 0, 60 dup (?)

readstring endp

readword proc near ;Чтение числа с клавиатуры

push bx dx si

lea dx,@@tmps

mov al,3

call readstring

xor ax,ax

lea si,@@tmps

cmp byte ptr [si+1],0

jne @@l2

mov al,[si]

xor bx,bx

jmp @@l1

@@l2:

mov al,[si]

cmp al,'0'

jb @@ex1

cmp al,'9'

ja @@ex1

sub al,'0'

mov bl,10

mul bl

mov bx,ax

mov al,[si+1]

@@l1:

cmp al,'0'

jb @@ex1

cmp al,'9'

ja @@ex1

sub al,'0'

add ax,bx

jmp @@ex2

@@ex1:

xor ax,ax

@@ex2:

pop si dx bx

ret

@@tmps db 3 dup (?)

readword endp

writestring proc near ;Вывод строки на экран

;DS:DX->STR

push ax si

mov si,dx

cld

@@l1:

lodsb

test al,al

jz @@l2

push si

mov ah,0Eh

int 10h

pop si

jmp @@l1

@@l2:

pop si ax

ret

writestring endp

writeword proc near ;Вывод числа на экран

;EAX=WORD

pusha

cmp ax,0

jne @@ll1

mov ax,0E30h

int 10h

popa

ret

@@ll1:

lea dx,@@res\_str

mov cx,20

call memzero

mov @@value,eax

fild @@value ;загрузка числа в сопроцессор

fbstp @@bcd\_value ;выгрузка из сопроцессора в память

;в BCD формате (двоично-десятичный)

mov si,offset @@bcd\_value+10-1

lea di,@@res\_str

std

@@l1:

lodsb

test al,al

jz @@l1

push ax

shr al,4

test al,al

jz @@l3

add al,'0'

mov [di],al

inc di

@@l3:

pop ax

and al,0Fh

add al,'0'

mov [di],al

inc di

cmp si,offset @@bcd\_value

jb @@l2

jmp @@l1

@@l2:

lea dx,@@res\_str

call writestring

popa

ret

@@value dd 0

@@bcd\_value dt 0

@@res\_str db 20 dup (0)

writeword endp

gotoxy proc near ;Установить позицию курсора

;AH=Y AL=X

push ax bx dx

mov dx,ax

xor bh,bh

mov ah,2

int 10h

pop dx bx ax

ret

gotoxy endp

clrscr proc near ;Очитска экрана

push ax

mov ax,0003

int 10h

pop ax

ret

clrscr endp

settextattr proc near ;установка атрибутов текста

;AL=атрибут

;BL=X BH=Y - начальная позиция

;CX - количество знакомест

push ax bx cx di es

mov dl,80\*2

push ax

xor ax,ax

mov al,bh

mul dl

xor dx,dx

mov dl,bl

shl dx,1

add ax,dx

mov di,ax

inc di

pop ax

mov dx,0B800h

push dx

pop es

@@l1:

stosb

inc di

loop @@l1

pop es di cx bx ax

ret

settextattr endp

clrworkfield proc near ; очитска рабочей части экрана

push ax cx dx

mov ax,0500h

lea dx,clr

mov cx,3

@@l1:

call gotoxy

call writestring

inc ah

loop @@l1

mov ax,1600h

call gotoxy

lea dx,clr

call writestring

pop dx cx ax

ret

clrworkfield endp.

sF2 equ 3Ch

sF3 equ 3Dh

sF4 equ 3Eh

sF5 equ 3Fh

sF6 equ 40h

sF7 equ 41h

sF8 equ 42h

sF9 equ 43h

sF10 equ 44h

STUD STRUC

SName db 20 dup (' ')

SAge dw 0

SCity db 20 dup (' ')

STUD ends