Министерство образования Российской Федерации

Институт переподготовки кадров

Уральского государственного технического

университета

##### Кафедра микропроцессорной техники

###### Курсовая работа

ТЕМА: **Создать меню без файла описания ресурсов с помощью функций CreateMenu и CreatePopupMenu.**

**Пояснительная записка**

Руководитель В.П.Кулюкин

Слушатель гр. СП-923 Г.Г.

**2001г.**

**Содержание**

Введение 3

Постановка задачи 5

Создание меню без описания ресурсов с помощью функций CreateMenu и CreatePopupMenu 6

Текст программы 7

Заключение 13

Библиографический список 14

# Введение

Ассемблер- один из самых старых из существующих сегодня языков программирования.Когда-то это был самый основной язык, без которого нельзя было заставить компьютер сделать что-либо полезное.

Постепенно ситуация менялась. Появились более удобные средства общения с компьютером. Но в отличии от других языков ассемблер не умирал, более того он немог этого сделать в принципе.

Если коротко, то ассемблер- это символическое представление машинного языка. Все процессы в машине на самом низком, аппаратном уровне приводятся в действие только командами машинногшо языка.

Отсюда понятно, что несмотря на общее название, язык Ассемблера для каждого типа компьютера свой. Это касается и внешнего вида программ, написанных на Ассемблере, и идей, отражением которых этот язык является.

Известно, что к программированию на языке ассемблера обращаются тогда, когда от программы требуется максимальная скорость исполнения, когда необходимо обеспечить взаимодействие с нестандартными внешними устройствами, когда необходимо полностью использовать возможности процессора и операционной системы. На языке ассемблера можно запрограммировать все, на что способна соответствующая вычислительная машина, то есть ассемблер является машинно-ориентированным языком программирования. Программируя на ассемблере иногда в силу привычки, иногда в силу необходимости, особенно при организации интерфейса пользователя, приходится многократно программировать одни и те же элементарные задачи. В языках высокого уровня эта проблема решена применением стандартных функций и процедур. В ассемблере эта проблема могла бы быть решена за счет библиотек стандартных функций как на уровне исходных текстов, так и объектных кодов, но такие библиотеки не стандартизованы и не распространяются вместе с компиляторами. С появлением Windows 95 ситуация несколько изменилась. Создание приложений на языке ассемблера представляет собой весьма сложную задачу в связи с высоким уровнем интеграции прикладной программы и операционной системы, но теперь нет необходимости многократно решать проблемы пользовательского интерфейса и управления исполнением команд на уровне машинных команд [I]. Они решаются теперь с помощью операционной системы за счет обращения к функциям интерфейса прикладного программирования – Application Programming Interface (API).

Программирование пользовательского интерфейса с применением функций Win32 API основано на использовании так называемых ресурсов. Ресурсами являются соответствующим образом оформленные данные, используемые операционной системой для создания внешнего отображения органов управления приложением, и средства, обеспечивающие ввод данных в процессе исполнения программы в режиме диалога. Ресурсы описываются в текстовом файле с расширением

rc. Файл ресурсов после обработки компилятора ресурсов и получения двоичного файла с расширением res с помощью компоновщика объединяется с исполняемым файлом.

# Постановка задачи

Наиболее очевидным средством управления приложением является меню. Строка меню выводится на экран непосредственно под строкой заголовка. Это меню называется главным. Выбор элемента меню влечет за собой выдачу приложению сообщения WM\_COMMAND, содержащего идентификатор пункта меню. Идентификаторы анализируются в оконной процедуре приложения, что обеспечивает соответствующую реакцию на полученное сообщение. Каждый пункт меню определяется тремя характеристиками. Первая определяет то, что будет отображаться в данном пункте меню – это либо строка текста, либо картинка. Вторая характеристика определяет либо константу, посылаемую оконной процедуре в сообщении WM\_COMMAND, либо всплывающее меню, которое выводится на экран, когда пользователь выбирает данный пункт меню. Третья характеристика указывает, является ли данный пункт меню разрешенным (enabled), запрешенным (disabled), недоступным (grayed) или помеченным (checked). Эта характеристика пункта меню не является обязательной.

Меню можно создать тремя способами. Можно определить меню в файле описания ресурсов, создать меню без файла описания ресурсов непосредственно в программе с помощью функций CreatMenu, AppendMenu и InsertMenu или с помощью функции LoadMenuIndirect, создав предварительно структуру MENUITEMPLATE. Наиболее простым и понятным при программировании под Windows95 на языке ассемблера является определение меню в файле описания ресурсов с помощью любого текстового редактора.

Моя задача заключается в создание меню без файла описания ресурсов с помощью функций CreateMenu и CreatePopupMenu. Одним из средств создания меню без использования файла описания ресурсов является функции CreateMenu с последующим добавлением необходимого числа пунктов с помощью функции AppendMenu.

# Создание меню без описания ресурсов с помощью функций CreateMenu и CreatePopupMenu

Windows-приложение на ассемблере содержит один сегмент данных .data и один сегмент кода .code. В программе использовано 20 функций API. Параметры этим функциям передаются через стек, результат возвращается в регистре ЕАХ.

Директива mode1 задает плоскую модель памяти (flat) и стиль передачи параметров при входе в процедуры программы и выходе из них (stdcall). Модель памяти flat обозначает плоскую модель памяти. В соответствии с плоской моделью памяти компилятор создает программу, которая содержит 32-битовый сегмент для данных и кода программы. Указание этой модели памяти заставляет компоновщик создать исполняемый файл с расширением .exe.

Параметр stdcall определяет порядок передачи параметров через стек справа налево.

Директива include включае в программу файл win32.inc.

Функции Win32 API, используемые в программе должны быть объявлены с помощью директивы extrn для того, чтобы компилятор мог сгенерировать правильный код.

При создании окна приложения с помощью CreateMenu получаем дескриптор меню и используем его в качестве одного из параметров функций CreateWindowExA. Затем с помощью функции CreatePopupMenu получаем дескриптор выпадающего меню и вызовом AppendMenu определяем все пункты наших меню.

Результат работы программы приведен на Рис.1.

# Текст программы

1. **p386** ; эта директива разрешает транслятору обрабатывать команды

; процессора i386

1. **jumps**  ;транслятор автоматически преобразует команду

;условной

; передачи управления в комбинацию условной ибезусловной

; команд если условная команда в силу ограниченности

; области своего действия ;не обеспечивает передачу

; управления по нужному адресу

1. **model flat, STDCALL** ; выбирается модель памяти для 32- разрядного

; программирования и правила передачи

; параметров функциям STDCALL, далее

; действующие по умолчанию

1. **include win32.inc** ; файл описания структур и констант
2. **L equ <LARGE>** ; последовательности символов LARGE,

; являющейся именем операции, объявляющей

; следующий за ней операнд 32-разрядным,

; присваивается имя L

1. **IDM\_QUIT equ 100**
2. **IDM\_ABOUT equ 101**
3. **MF\_ENABLED equ 0**
4. **MF\_POPUP equ 10h**
5. **MF\_OWNERDRAW equ 100h**

**;Функции Win32 API, используемые программой**

1. **extrn BeginPaint :PROC**
2. **extrn CreateWindowExA :PROC**
3. **extrn DefWindowProcA :PROC**
4. **extrn DispatchMessageA :PROC**
5. **extrn EndPaint :PROC**
6. **extrn ExitProcess :PROC**
7. **extrn GetMessageA :PROC**
8. **extrn GetModuleHandleA :PROC**
9. **extrn LoadCursorA :PROC**
10. **extrn LoadIconA :PROC**
11. **extrn RegisterClassA :PROC**
12. **extrn PostQuitMessage :PROC**
13. **extrn ShowWindow :PROC**
14. **extrn TranslateMessage :PROC**
15. **extrn UpdateWindow :PROC**
16. **extrn CreateMenu :PROC**
17. **extrn MessageBoxA :PROC**
18. **extrn AppendMenuA :PROC**
19. **extrn DrawMenuBar :PROC**
20. **extrn CreatePopupMenu :PROC**
21. **.data**  ; переопределенное имя, означающее начало сегмента

; данных

1. **newhwnd dd 0**
2. **lppaint PAINTSTRUCT <?>**
3. **msg MSGSTRUCT <?>**
4. **wc WNDCLASS <?>**
5. **hInst dd 0**
6. **fl dw 0**
7. **szTitleName db 'Это наше окно',0**
8. **szClassName db 'ASMCLASS32',0**
9. **szHello db 'Привет!',0**
10. **szAppName db 'Сообщение',0**
11. **hMenu dd 0**
12. **hsubmenu dd 0**
13. **item1 db 'Меню',0**
14. **item2 db 'Выход',0**
15. **item3 db 'Возврат',0**
16. **.code** ; переопределенное имя, означающее начало

; сегмента кода

1. **start:**
2. **push L 0**
3. **call GetModuleHandleA**
4. **mov [hInst], eax**

**;инициализация структуры WndClass**

1. **mov [wc.clsStyle], CS\_HREDRAW + CS\_VREDRAW +**

**CS\_GLOBALCLASS**

1. **mov [wc.clsLpfnWndProc], offset WndProc**
2. **mov [wc.clsCbClsExtra], 0**
3. **mov [wc.clsCbWndExtra], 0**
4. **mov eax, [hInst]**
5. **mov [wc.clsHInstance], eax**
6. **push L IDI\_APPLICATION**
7. **push L 0**
8. **call LoadIconA**
9. **mov [wc.clsHIcon], eax**
10. **push L IDC\_ARROW**
11. **push L 0**
12. **call LoadCursorA**
13. **mov [wc.clsHCursor], eax**
14. **mov [wc.clsHbrBackground], COLOR\_WINDOW + 1**
15. **mov dword ptr [wc.clsLpszMenuName],0**
16. **mov dword ptr [wc.clsLpszClassName], offset szClassName**
17. **push offset wc**
18. **call RegisterClassA**
19. **call CreateMenu**
20. **mov [hMenu],eax**
21. **push L 0 ; lpParam**
22. **push [hInst] ; hInstance**
23. **push [hMenu] ; menu**
24. **push L 0 ; parent hwnd**
25. **push L CW\_USEDEFAULT ; height**
26. **push L CW\_USEDEFAULT ; width**
27. **push L CW\_USEDEFAULT ; y**
28. **push L CW\_USEDEFAULT ; x**
29. **push L WS\_OVERLAPPEDWINDOW ; Style**
30. **push offset szTitleName ; Title string**
31. **push offset szClassName ; Class name**
32. **push L 0 ; extra style**
33. **call CreateWindowExA**
34. **mov [newhwnd], eax**
35. **call CreatePopupMenu**
36. **mov [hsubmenu], eax**
37. **push offset item2**
38. **push L IDM\_QUIT**
39. **push 0**
40. **push eax**
41. **call AppendMenuA**
42. **push offset item3**
43. **push L IDM\_ABOUT**
44. **push 0**
45. **push [hsubmenu] ;eax**
46. **call AppendMenuA**
47. **push offset item1**
48. **push [hsubmenu]**
49. **push MF\_POPUP**
50. **push [hMenu]**
51. **call AppendMenuA**
52. **nop**
53. **push L SW\_SHOWNORMAL**
54. **push [newhwnd]**
55. **call ShowWindow**
56. **push [newhwnd]**
57. **call UpdateWindow**
58. **push [newhwnd]**
59. **call DrawMenuBar**
60. **msg\_loop:**
61. **push L 0**
62. **push L 0**
63. **push L 0**
64. **push offset msg**
65. **call GetMessageA**
66. **cmp ax, 0**
67. **je end\_loop**
68. **push offset msg**
69. **call TranslateMessage**
70. **push offset msg**
71. **call DispatchMessageA**
72. **jmp msg\_loop**
73. **end\_loop:**
74. **push [msg.msWPARAM]**
75. **call ExitProcess**
76. **;----Оконная процедура----**
77. **WndProc proc uses ebx edi esi, hwnd:DWORD, wmsg:DWORD,\**
78. **wparam:DWORD, lparam:DWORD**
79. **LOCAL hDC:DWORD**
80. **cmp [wmsg], WM\_DESTROY**
81. **je wmdestroy**
82. **cmp [wmsg], WM\_SIZE**
83. **je wmsize**
84. **cmp [wmsg], WM\_CREATE**
85. **je wmcreate**
86. **cmp [wmsg],WM\_PAINT**
87. **je wmpaint**
88. **;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***
89. **cmp [wmsg],WM\_COMMAND**
90. **je wmcommand**
91. **;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***
92. **jmp defwndproc**
93. **wmcommand:**
94. **mov eax,lparam**
95. **cmp ax,0**
96. **jne m1**
97. **mov eax,wparam**
98. **cmp ax,IDM\_ABOUT**
99. **jne m2**
100. **call MessageBoxA,0,offset szHello,offset szAppName,MB\_OK**
101. **jmp m1**
102. **m2: cmp ax,IDM\_QUIT**
103. **jne m1**
104. **push 0**
105. **call PostQuitMessage**
106. **m1: mov eax,0**
107. **jmp finish**
108. **wmcreate:**
109. **mov eax, 0**
110. **jmp finish**
111. **defwndproc:**
112. **push [lparam]**
113. **push [wparam]**
114. **push [wmsg]**
115. **push [hwnd]**
116. **call DefWindowProcA**
117. **jmp finish**
118. **wmdestroy:**
119. **push L 0**
120. **call PostQuitMessage**
121. **mov eax, 0**
122. **jmp finish**
123. **wmsize:**
124. **mov eax, 0**
125. **jmp finish**
126. **wmpaint:**
127. **push offset lppaint**
128. **push [hwnd]**
129. **call BeginPaint**
130. **mov [hDC],eax**
131. **push offset lppaint**
132. **push [hwnd]**
133. **call EndPaint**
134. **mov eax,0**
135. **jmp finish**
136. **finish:**
137. **ret**
138. **WndProc endp**
139. **;---------------------------------**
140. **public WndProc**
141. **end start ;** конец программы

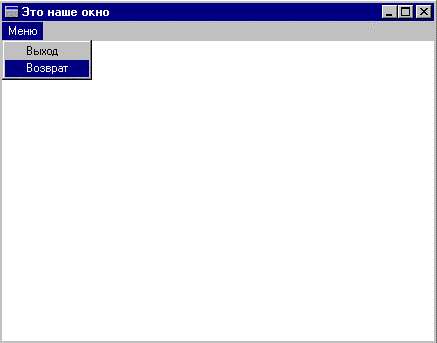


рис1. Окно программы

# Заключение

Моя задача состояла в создании программыбез файла описания ресурсов с помощью функций CreateMenu и CreatePopupMenu. Это приложение не выполняет никакой полезной работы и служит для демонстрации некоторых возможностей системы в организации интерфейса пользователя.

# Библиографический список

1. Зубков С.В. Assembler. Для DOS Windows и Unix. М.: ДМК, 1999
2. Пустоваров В.И. АССЕМБЛЕР. Программирование и анализ корректности машинных программ.
3. Применение TURBO ASSEMBLER для программирования ПЭВМ. Часть1,2.: Методические указания к лабораторному практикуму по дисциплине : «Системное программное обеспечение» / Составитель В.П.Кулюкин. Екатеринбург: изд. ИПК УГТУ, 2000.