
	Instytut Informatyki Politechniki Śląskiej Zespół Mikroinformatyki i Teorii Automatów Cyfrowych			
Rok akademicki:	Rodzaj studiów*: SSI/NSI/NSM	Języki Asemblerowe	LAB	II

TEMAT:

Tryby adresacji procesorów x86.

CEL:

Celem ćwiczenia jest poznanie trybów adresacji procesora x86. Konstrukcja projektu zakłada możliwość wywoływania funkcji bibliotecznych napisanych w asemblerze z poziomu aplikacji oraz pokazuje prawidłową konfigurację środowiska umożliwiającą debugowanie kodu do poziomu asemblera, obserwację stanu rejestrów i flag procesora czy obszarów pamięci danych.

ZAŁOŻENIA:

W środowisku VS 2008/2010 zakładamy solucję JALab składającą się z dwóch projektów:

- JAApp Projekt aplikacja WIN32 w j. C++,
- JADll Projekt biblioteka DLL w asemblerze,

W bibliotece DLL utworzona zostanie funkcja asemblerowa, której wywołanie i przekazywanie parametrów wystąpi w aplikacji. W ćwiczeniu wykorzystać założenia opisane w ćw. 1

WYKONANIE:

Treść pliku JAAsm.asm jest następująca:

```
.486
.model flat, stdcall

.data
DataString DB 'AGIJKSZ', 0FFH ; definicja ciągu znaków

.code

;*****
;*   Procedura FindChar_1 wyszukiwania znaku 'J' w ciągu 'DataString'      *
;*   *                                                                      *
;*   Bezpośrednia adresacja indeksowa                                     *
;*   Parametry wejściowe:                                                 *
;*       AH - szukany znak 'J'                                           *
;*   Parametry wyjściowe:                                                 *
;*       EAX - BOOL TRUE Found, FALSE not found                         *
;*   *                                                                      *
;*****
FindChar_1 PROC
    MOV     ESI, OFFSET DataString    ; załaduj offset zmiennej 'DataString' do rej.
SI
    MOV     AH, 'J'                  ; załaduj kod litery 'J' do rej. AH
Check_End:
    CMP     BYTE PTR [ESI], 0FFH     ; czy koniec lancucha (znak specjalny FF)?
    JE      Not_Find                 ; znaleziono znak konca (wartownik)
    CMP     AH, [ESI]                 ; porównaj znak z elementem lancucha
'DataString'
    JE      Got_Equal                 ; znaleziono znak!
    ADD     ESI, 1                    ; inkrementuj offset
```

```

        JMP     Check_End                ; petla wyszukiwania
Got_Equal:
        MOV     DL, [ESI]                ; zaladuj znaleziony znak do DL
        JMP     Done
Not_Find:
        MOV     EAX,0                    ; nie znaleziono znaku
        RET                                     ; powrot z procedury
Done:
        MOV     EAX,1                    ; znaleziono znak
        RET                                     ; powrot z procedury
FindChar_1     ENDP                        ; koniec FindChar_1

```

```

;*****
;*      Procedura FindChar_2 wyszukiwania znaku 'J' w ciagu 'LocalString'      *
;*                                                                              *
;*      Bezposrednia adresacja indeksowa                                     *
;*      Parametry wejscowe:                                                  *
;*          AH - szukany znak 'J'                                           *
;*      Parametry wyjscowe:                                                  *
;*          EAX - BOOL TRUE Found, FALSE not found                         *
;*                                                                              *
;*****
FindChar_2 PROC

```

```

LocalString DB 0C3H,'AGIJKSZ', 0FFH      ; definicja ciagu znakow

        MOV     ESI, OFFSET LocalString  ; zaladuj offset zmiennej 'LocalString' do rej.
ESI
        MOV     AH, 'J'                  ; zaladuj kod litery 'J' do rej. AH
Check_End:
        CMP     BYTE PTR [ESI], 0FFH     ; czy koniec lancucha (znak specjalny FF)?
        JE      Not_Find                  ; znaleziono znak konca (wartownik)
        CMP     AH, [ESI]                 ; porownaj znak z elementem lancucha
'LocalString'
        JE      Got_Equal                  ; znaleziono znak!
        ADD     ESI, 1                     ; inkrementuj offset
        JMP     Check_End                  ; petla wyszukiwania
Got_Equal:
        MOV     DL, [ESI]                ; zaladuj znaleziony znak do DL
        JMP     Done
Not_Find:
        MOV     EAX,0                    ; nie znaleziono znaku
        RET                                     ; powrot z procedury
Done:
        MOV     EAX,1                    ; znaleziono znak
        RET                                     ; powrot z procedury
FindChar_2     ENDP                        ; koniec FindChar_2

```

FindChar_3 PROC AppString: DWORD

```

;*****
;*      Procedura FindChar_3 wyszukiwania znaku 'J' w ciagu 'AppString'      *
;*                                                                              *
;*      Bezposrednia adresacja indeksowa                                     *
;*      Parametry wejscowe:                                                  *
;*          AH - szukany znak 'J'                                           *
;*      Rej:  ESI - offset adresu zmiennej 'AppString'                      *
;*      Parametry wyjscowe:                                                  *
;*          EAX - BOOL TRUE Found, FALSE not found                         *
;*                                                                              *
;*****
        MOV     ESI, AppString            ; zaladuj offset zmiennej 'AppString' do rej.
ESI
        MOV     AH, 'J'                  ; zaladuj kod litery 'J' do rej. AH
Check_End:

```

```

        CMP     BYTE PTR [ESI], 0FFh      ; czy koniec lancucha (znak specjalny FF)?
        JE      Not_Find                  ; znaleziono znak konca (wartownik)
        CMP     AH, [ESI]                 ; porownaj znak z elementem lancucha
'AppString'
        JE      Got_Equal                 ; znaleziono znak!
        ADD     ESI, 1                     ; inkrementuj offset
        JMP     Check_End                 ; petla wyszukiwania
Got_Equal:
        MOV     DL, [ESI]                 ; zaladuj znaleziony znak do DL
        JMP     Done
Not_Find:
        MOV     EAX, 0                     ; nie znaleziono znaku
        RET                                     ; powrot z procedury
Done:
        MOV     EAX, 1                     ; znaleziono znak
        RET                                     ; powrot z procedury
FindChar_3     ENDP                       ; koniec FindChar_3

```

```

;*****
;*      Procedura FindChar_4 wyszukiwania znaku 'J' w ciagu 'DataString'      *
;*                                                                           *
;*      Bezposrednia adresacja indeksowa                                     *
;*      Parametry wejscowe:                                                 *
;*      Rej:  ESI - indeks zmiennej 'DataString'                           *
;*           AH - szukany znak 'J'                                         *
;*      Parametry wyjscowe:                                                 *
;*           EAX - BOOL TRUE Found, FALSE not found                       *
;*                                                                           *
;*****
FindChar_4 PROC NEAR                    ; deklaracja procedury FindChar_4
        MOV     ESI, 0                  ; zaladuj indeks lancucha 'DataString' do ESI
        MOV     AH, 'J'                 ; zaladuj kod litery 'J' do rej. AH
Check_End:
        CMP     DataString[ESI], 0FFh  ; czy koniec lancucha (znak specjalny FF)?
        JE      Not_Find                ; znaleziono znak konca (wartownik)
        CMP     AH, DataString[ESI]     ; porownaj znak z elementem lancucha
'DataString'
        JE      Got_Equal                ; znaleziono znak!
        ADD     SI, 1                    ; inkrementuj indeks
        JMP     Check_End                ; petla wyszukiwania
Got_Equal:
        MOV     DL, DataString[ESI]     ; zaladuj znaleziony znak do DL
        JMP     Done
Not_Find:
        MOV     EAX, 0                   ; nie znaleziono znaku
        RET                                     ; powrot z procedury
Done:
        MOV     EAX, 1                   ; znaleziono znak
        RET                                     ; powrot z procedury
FindChar_4     ENDP                       ; koniec FindChar_4

```

```

;*****
;*      Procedura FindChar_5 wyszukiwania znaku 'J' w ciagu 'DataString'      *
;*                                                                           *
;*      Adresacja Base + Index                                               *
;*      Parametry wejscowe:                                                 *
;*      Rej:  BX - offset zmiennej 'DataString'                           *
;*      Parametry wyjscowe:                                                 *
;*           EAX - BOOL TRUE Found, FALSE not found                       *
;*                                                                           *
;*****
FindChar_5 PROC NEAR
        MOV     EBX, OFFSET DataString ; zaladuj offset zmiennej 'DataString' do rej.
EBX

```

```

;*****
END

```

EXPORT FUNKCJI

W pliku JADll.def wyeksportować funkcje:

```
LIBRARY
```

```
EXPORTS
```

```
FindChar_1  
FindChar_2  
FindChar_3  
FindChar_4  
FindChar_5  
FindChar_6
```

PROGRAM GŁÓWNY

W projekcie **JALab** zawartość pliku **JAApp.cpp** jest następująca:

```
// JAApp.cpp : Defines the entry point for the application.  
//  
  
#include "stdafx.h"  
#include "JALab.h"  
  
#define MAX_LOADSTRING 100  
  
// Global Variables:  
HINSTANCE hInst;                                // current instance  
TCHAR szTitle[MAX_LOADSTRING];                  // The title bar text  
TCHAR szWindowClass[MAX_LOADSTRING];            // the main window class name  
  
// Forward declarations of functions included in this code module:  
ATOM                        MyRegisterClass(HINSTANCE hInstance);  
BOOL                        InitInstance(HINSTANCE, int);  
LRESULT CALLBACK           WndProc(HWND, UINT, WPARAM, LPARAM);  
INT_PTR CALLBACK           About(HWND, UINT, WPARAM, LPARAM);  
  
int APIENTRY _twinMain(HINSTANCE hInstance,  
                        HINSTANCE hPrevInstance,  
                        LPTSTR    lpCmdLine,  
                        int        nCmdShow)  
{  
    UNREFERENCED_PARAMETER(hPrevInstance);  
    UNREFERENCED_PARAMETER(lpCmdLine);  
  
    // TODO: Place code here.  
    MSG msg;  
    HACCEL hAccelTable;  
  
    // Initialize global strings  
    LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);  
    LoadString(hInstance, IDC_JALAB, szWindowClass, MAX_LOADSTRING);  
    MyRegisterClass(hInstance);  
  
    // Perform application initialization:  
    if (!InitInstance (hInstance, nCmdShow))  
    {  
        return FALSE;  
    }  
  
    hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_JALAB));
```

```

char szString[] = {'A','G','I','J','K','S', 0xFF}; // definicja ciagu znakow

if (FindChar_1 ())
    MessageBox (NULL,L"Found J",L"FindChar_1",MB_OK);
else
    MessageBox (NULL,L"Not Found J",L"FindChar_1",MB_OK);
if (FindChar_2 ())
    MessageBox (NULL,L"Found J",L"FindChar_2",MB_OK);
else
    MessageBox (NULL,L"Not Found J",L"FindChar_2",MB_OK);

if (FindChar_3 (szString))
    MessageBox (NULL,L"Found J",L"FindChar_3",MB_OK);
else
    MessageBox (NULL,L"Not Found J",L"FindChar_3",MB_OK);
if (FindChar_4 ())
    MessageBox (NULL,L"Found J",L"FindChar_4",MB_OK);
else
    MessageBox (NULL,L"Not Found J",L"FindChar_4",MB_OK);
if (FindChar_5 ())
    MessageBox (NULL,L"Found J",L"FindChar_5",MB_OK);
else
    MessageBox (NULL,L"Not Found J",L"FindChar_5",MB_OK);
if (FindChar_6 ())
    MessageBox (NULL,L"Found J",L"FindChar_6",MB_OK);
else
    MessageBox (NULL,L"Not Found J",L"FindChar_6",MB_OK);

// Main message loop:
while (GetMessage(&msg, NULL, 0, 0))
{
    if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}

return (int) msg.wParam;
}

...

```

W projekcie **JAApp** zawartość pliku **JAApp.h** jest następująca:

```
#pragma once

#include "resource.h"
#include <windows.h>

extern "C" int __stdcall FindChar_1 (void);
extern "C" int __stdcall FindChar_2 (void);
extern "C" int __stdcall FindChar_3 (LPSTR String);
extern "C" int __stdcall FindChar_4 (void);
extern "C" int __stdcall FindChar_5 (void);
extern "C" int __stdcall FindChar_6 (void);
```

ZADANIE

Utworzyć solucję **JALab** wraz z projektami **JAApp** oraz **JADII** (tak jak w ćw. 1).

Przeprowadzić analizę działania procedur **FindChar_1 .. FindChar_6**.

1. Usunąć ew. błędy
2. Dokonać analizy czasu wykonania zliczając takty maszynowe wykonywanych rozkazów wewnątrz procedur **FindChar_1 .. FindChar_6**.
3. Napisać własną procedurę **My_Procedure** wyszukiwania znaku w łańcuchu starając się aby jej czas wykonania był najszybszy,
4. Wygenerować indywidualne sprawozdanie w formacie PDF zawierające:
 - opis zauważonych błędów,
 - wygenerować pełny listing asemblacji JAAsm.log
 - tabelę czasów wykonania procedur **FindChar_1 .. FindChar_6**

• Procedura	• Liczba taktów zegarowych
• FindChar_1	•
• FindChar_2	•
• FindChar_3	•
• FindChar_4	•
• FindChar_5	•
• FindChar_6	•
• My_Procedure	•

- Wyciągnąć wnioski z tabeli