	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 2022 zakładamy solucję JALab2 składającą się z dwóch projektów:

- JALab2 Projekt aplikacja WIN32 w j. C++,
- JALib2 Projekt biblioteka DLL w asemblerze,

W bibliotece DLL utworzone są funkcje asemblerowe wyszukujące w ciągu znaków litery ‘J’ wykorzystujące do tego celu różne sposoby adrescji procesorów x86.

WYKONANIE:

Treść pliku JAAsm.asm jest następująca:

```
.model flat, stdcall
.data

DataString  DB  '0123456789012345678 9AJBC', 0FFH  ; definicja ciągu znakow z wartownikiem 0xFF

.code

;************************************************************************
;*    Procedura FindChar_1 wyszukiwania znaku 'J' w ciagu 'DataString'      *
;*                                                                        *
;*      Bezposrednia adresacja indeksowa                                  *
;*      Parametry wejsciowe:                                              *
;*            AH - szukany znak 'J'                                       *
;*      Parametry wyjsciowe:                                              *
;*            EAX - BOOL TRUE Found, FALSE not found                      *
;*                                                                        *
;************************************************************************
FindChar_1 PROC
                MOV   ESI, OFFSET DataString    ; zaladuj offset zmiennej 'DataString' do
rej. SI
            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 '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 wejsciowe:                                              *
;*            AH - szukany znak 'J'                                       *
;*      Parametry wyjsciowe:                                              *
;*            EAX - BOOL TRUE Found, FALSE not found                      *
;*                                                                        *
;************************************************************************
FindChar_2 PROC

                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

LocalString  DB  '12345678901234567890123456789012345678901234567890JKLMWZS1', 0FFH  ;
definicja ciągu znakow z wartownikiem 0xFF

FindChar_3 PROC AppString: DWORD
;*************************************************************************
;*    Procedura FindChar_3 wyszukiwania znaku 'J' w ciagu 'AppString'       *
;*                                                                          *
;*      Bezposrednia adresacja indeksowa                                    *
;*      Parametry wejsciowe:                                                *
;*            AH - szukany znak 'J'                                         *
;*      Rej:  ESI - offset adresu zmiennej 'AppString'                      *
;*      Parametry wyjsciowe:                                                *
;*            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 wejsciowe:                                                *
;*      Rej:  ESI - indeks zmiennej 'DataString'                            *
;*            AH - szukany znak 'J'                                         *
;*      Parametry wyjsciowe:                                                *
;*            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 wejsciowe:                                                *
;*      Rej:  BX - offset zmiennej 'DataString'                             *
;*      Parametry wyjsciowe:                                                *
;*            EAX - BOOL TRUE Found, FALSE not found                        *
;*                                                                          *
;*************************************************************************
FindChar_5  PROC  NEAR
            MOV   EBX, OFFSET DataString      ; zaladuj offset zmiennej 'DataString' do rej.
EBX
            CMP   BYTE PTR [EBX+0], 'J'     ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+1], 'J'     ; porownaj znak z elementem lancucha 'DataString'
                 JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+2], 'J'     ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+3], 'J'     ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+4], 'J'     ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+5], 'J'     ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+6], 'J'     ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+7], 'J'     ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+8], 'J'     ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+9], 'J'     ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+10], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+11], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+12], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+13], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+14], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+15], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+16], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+17], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+18], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+19], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+20], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+21], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+22], 'J'    ; porownaj znak z elementem lancucha 'DataString'
```

```
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+23], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak
            CMP   BYTE PTR [EBX+24], 'J'    ; porownaj znak z elementem lancucha 'DataString'
            JE    Got_It                    ; znaleziono znak

Not_Find:
            MOV   EAX,0                     ; zaladuj znak zapytania do DL
            RET                             ; powrot z procedury
Got_It:
            MOV   EAX,1                     ; wyswietl znak ne ekranie
            RET                             ; powrot z procedury
FindChar_5  ENDP                            ; koniec  FindChar_5


;*************************************************************************
;*    Procedura FindChar_6 wyszukiwania znaku 'J' w ciagu 'DataString'      *
;*                                                                          *
;*      adresacja Disp [EBX+ESI]                                            *
;*      Parametry wejsciowe:                                                *
;*      Rej:  EBX - offset zmiennej 'DataString'                            *
;*            ESI - przemieszczenie                                         *
;*            AH - szukany znak 'J'                                         *
;*      Parametry wyjsciowe:                                                *
;*            EAX - BOOL TRUE Found, FALSE not found                        *
;*                                                                          *
;*************************************************************************
FindChar_6  PROC  NEAR
            MOV   EBX, OFFSET DataString    ; zaladuj offset zmiennej 'String' do rej. ESI
            XOR   ESI, ESI                  ; wyzeruj indeks lancucha 'String' w ESI
            MOV   AH, 'J'                   ; zaladuj kod litery 'J' do rej. AH
Check_End:
            CMP   BYTE PTR [EBX+ESI], 0FFH  ; czy koniec lancucha (znak specjalny FF)?
            JE    Not_Find                  ; znaleziono znak konca (wartownik)
            CMP   AH, BYTE PTR [EBX+ESI]    ; porownaj znak z elementem lancucha 'String'
            JE    Got_Equal                 ; znaleziono znak!
            INC   ESI                       ; inkrementuj indeks
            JMP   Check_End                 ; petla wyszukiwania
Got_Equal:
            MOV   DL, [EBX+ESI]             ; zaladuj znaleziony znak do DL
            JMP   Done
Not_Find:
            MOV   EAX,0                     ; nie znaleziono znaku
            RET
Done:
            MOV   EAX,1                     ; znaleziono znak
            RET                             ; powrot z procedury
FindChar_6  ENDP                            ; koniec FindChar_6


;*************************************************************************
;*    Procedura FindChar_7 wyszukiwania znaku 'J' w ciągu 'DataString'      *
;*                                                                          *
;*      adresacja własna                                                    *
;*      Parametry wejsciowe:                                                *
;*            AH - szukany znak 'J'                                         *
;*            Proponowana adresacja uzyskania najmniejszej liczby taktów    *
;*            procesora w trybie x86                                        *
;*                                                                          *
;*      Parametry wyjsciowe:                                                *
;*            EAX - BOOL TRUE Found, FALSE not found                        *
;*                                                                          *
;*************************************************************************
FindChar_7  PROC  NEAR

; To do .......
```

```
Not_Find:
            MOV   EAX,0                     ; nie znaleziono znaku
            RET
Done:
            MOV   EAX,1                     ; znaleziono znak
            RET                             ; powrot z procedury
FindChar_7  ENDP                            ; koniec FindChar_7

;*************************************************************************
END
```

EXPORT FUNKCJI

W pliku JAAsm.def wyeksportować funkcje:

```
LIBRARY JALib2

EXPORTS
FindChar_1
FindChar_2
FindChar_3
FindChar_4
FindChar_5
FindChar_6
FindChar_7
```

PROGRAM GŁÓWNY

W projekcie **JALab2** zawartość pliku **JALab2.cpp** jest następująca:

```
/*****************************************************************************************
 *                              JA Ćwiczenie laboratoryjne 2                             *
 *                                                                                       *
 *   Temat: Sposoby adresacji procesora x86 w trybie x86                                 *
 *                                                                                       *
 *   Zadanie:                                                                            *
 *   1. Porównać różne metody adresacji stringu szString[] w celu wyznaczenia najszybszego     *
 *      dostępu do elementu bufora pamięci                                               *
 *   2. Zaproponować własne rozwiązanie wyszukania znaku 'J' w stringu tak aby liczba taktów  *
 *      procesora była najmniejsza. Kod ASM wpisać do procedury FindChar7 w pliku JAAsm.asm   *
 *                                                                                       *
 *****************************************************************************************/


#include "framework.h"
#include "JALab2.h"

#pragma intrinsic(__rdtsc)

#define MAX_LOADSTRING 100

// Global Variables:
HINSTANCE hInst;                                // current instance
WCHAR szTitle[MAX_LOADSTRING];                  // The title bar text
WCHAR 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 wWinMain(_In_ HINSTANCE hInstance,
                      _In_opt_ HINSTANCE hPrevInstance,
                      _In_ LPWSTR    lpCmdLine,
                      _In_ int       nCmdShow)
{
```

```
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);

    // TODO: Place code here.

    uint64_t dwBegin  = 0;
    uint64_t dwEnd    = 0;
    uint64_t dwTics   = 0;
    int      nRet     = 0;

    LONGLONG start    = 0;
    LONGLONG finish   = 0;
    LONGLONG Duration = 0;

    TCHAR szMessage[100];

    // Initialize global strings
    LoadStringW(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
    LoadStringW(hInstance, IDC_JALAB2, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass(hInstance);

    // Perform application initialization:
    if (!InitInstance (hInstance, nCmdShow))
    {
        return FALSE;
    }

    HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_JALAB2));


    /********************************* JALab2 *********************************************/

    BYTE szString[] = {
'0','1','2','3','4','5','6','7','8','9','0','0','1','2','3','4','5','6','7','8','9','A','J','B',
'C', 0xFF };


/*****************************************************************************************
 *
 *   Metody pomiaru czasu wykonania kodu maszynowego dla procesorów x86:
 *   1. Funkcja GetNowTickCount ()
 *   2. Funkcja PeroformanceCounter ()
 *   3. rozkaz RDTSC wynik pomiaru dla x86 w rejestrach EDX:EAX wykorzystywany w
 *      programach typu CPUID do wyliczania benchmarków procesorów
 *
 *   Funkcje dają różne wyniki zależne od wyliczania czasu procesora w środowisku
 *   wielozadaniowego systemu operacyjnego np. Windows
 *
 *****************************************************************************************/

    dwBegin = GetNowTickCount();
    nRet = FindChar_1();
    dwEnd = GetNowTickCount();
    dwTics = dwEnd - dwBegin;
    wsprintf(szMessage, L"Char J found in string: %d Tics number: %lu", nRet, (LONG)dwTics);
    MessageBox(NULL,szMessage, L"FindChar_1", MB_OK);

    start = PerformanceCounter();                               // Record start time
    nRet = FindChar_1();
    finish = PerformanceCounter();                              // Record end time
    Duration = finish - start;                                  // liczba tików zegarowych x86
    wsprintf(szMessage, L"Char J found in string: %d Tics number: %lu", nRet, (ULONG)Duration);
    MessageBox(NULL, szMessage, L"FindChar_1", MB_OK);

    start = __rdtsc();
    nRet = FindChar_1();
    finish = __rdtsc();
    Duration = finish - start;                                  // liczba tików zegarowych x86
    wsprintf(szMessage, L"Char J found in string: %d Tics number: %lu", nRet, (ULONG)Duration);
    MessageBox(NULL, szMessage, L"FindChar_1", MB_OK);



    if (FindChar_2())                                           // wywołanie procedury
FindChar_2() z biblioteki JAAsm.lib
```

```
      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))                                    // wywołanie procedury
FindChar_3() z biblioteki JAAsm.lib
      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())                                            // wywołanie procedury
FindChar_4() z biblioteki JAAsm.lib
      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())                                            // wywołanie procedury
FindChar_5() z biblioteki JAAsm.lib
      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())                                            // wywołanie procedury
FindChar_6() z biblioteki JAAsm.lib
      MessageBox(NULL, L"Found J", L"FindChar_6", MB_OK);
    else
      MessageBox(NULL, L"Not Found J", L"FindChar_6", MB_OK);

    if (FindChar_7())                                            // wywołanie procedury
FindChar_7() z biblioteki JAAsm.lib
      MessageBox(NULL, L"Found J", L"FindChar_7", MB_OK);
    else
      MessageBox(NULL, L"Not Found J", L"FindChar_7", MB_OK);

/************************************ JALab2 ****************************************/
```

Kod programu…

```
LONGLONG PerformanceCounter()
{
 LARGE_INTEGER li;
 ::QueryPerformanceCounter(&li);
 return li.QuadPart;
}


#ifdef _WIN32
// get frequency of the performance counter
uint64_t GetPCFrequency()
{
 uint64_t freq = 0;
 if (!QueryPerformanceFrequency((LARGE_INTEGER*)&freq))
 {
  DWORD dwRet = GetLastError();
 }
 return freq;
}
#endif

uint64_t GetNowTickCount()
{
#ifdef _WIN32
 static uint64_t freq = GetPCFrequency();
 uint64_t now = 0;
 if (!QueryPerformanceCounter((LARGE_INTEGER*)&now))
 {
```

```
   DWORD dwRet = GetLastError();
  }
  return (now * 1000000000UL) / freq;

#else
  timespec ts;
  if (clock_gettime(CLOCK_MONOTONIC, &ts) < 0)
  {
    LOG(FATAL) << errno;
  }
  return (ts.tv_sec * 1000000000UL) + ts.tv_nsec;
#endif
}

uint64_t elapsed(uint32_t start_hi, uint32_t start_lo, uint32_t end_hi, uint32_t end_lo)
{
  uint64_t start = (((uint64_t)start_hi) << 32) | start_lo;
  uint64_t end = (((uint64_t)end_hi) << 32) | end_lo;
  return end - start;
}
```

W projekcie **JALab2** zawartość pliku **JALab2.h** jest następująca:

```
#pragma once

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

uint64_t GetNowTickCount();
LONGLONG PerformanceCounter();
uint64_t elapsed(uint32_t start_hi, uint32_t start_lo, uint32_t end_hi, uint32_t
end_lo);

extern "C" int _stdcall FindChar_1(void);
extern "C" int _stdcall FindChar_2(void);
extern "C" int _stdcall FindChar_3(LPBYTE String);
extern "C" int _stdcall FindChar_4(void);
extern "C" int _stdcall FindChar_5(void);
extern "C" int _stdcall FindChar_6(void);
extern "C" int _stdcall FindChar_7(void);
```

ZADANIE

Utworzyć solucję **JALab2** wraz z projektami **JALab2** oraz **JALib2l** (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 **`FindChar_7`** wyszukiwania znaku w łańcuchu starając się aby jej czas wykonania był najszybszy,
4. Za pomocą różnych metod pomiaru czasu zamieszczonych w programie dokonać analizy wykonania procedur FindChar_1 do FindChar_6. Dane pomiarowe przedstawić w tabeli:

• **Procedura**	• **Liczba taktów zegarowych**
• FindChar_1	•
• FindChar_2	•
• FindChar_3	•
• FindChar_4	•
• FindChar_5	•
• FindChar_6	•
• My_Proc_7	•

5. Wygenerować indywidualne sprawozdanie w formacie PDF zawierające:
   - opis zauważonych błędów,
   - Wyciągnąć wnioski z tabeli