Ćwiczenie

„Tworzenie i uruchamianie programów asemblerowych”

Tematy ćwiczenia

- przejście do trybu konsolowego,
- kompilacja, opcje kompilatora,
- konsolidacja, opcje konsolidatora,
- opracowanie aplikacji konsolowej

Sprawozdanie

Na każdym ćwiczeniu sporządza się za pomocą *Word'*a sprawozdanie na bazie materiałów ćwiczenia. Bazowa zawartość sprawozdania musi być przygotowana w domu przed ćwiczeniem (sprawozdanie do ćwiczenia pierwszego jest przygotowywane w czasie ćwiczenia). W czasie ćwiczenia do sprawozdania są dodawane wyniki testowania.

Treść sprawozdania:
strona tytułowa,
spis treści sporządzony za pomocą *Word'*a,
dla każdego punktu rozdziały "Zadanie ", "Opracowanie zadania" (rozdział z tekstem programu i komentarzami), "Testowanie" (rozdział z opisem danych wejściowych i wynikami testowania, w tym zrzuty aktywnego okna).

Wzorzec ze stroną tytułowej znajduje się w odrębnym pliku.

Nazwa (bez polskich liter, żeby można było archiwizować) pliku ze sprawozdaniem musi zawierać nazwę przedmiotu, numer ćwiczenia i nazwisko studenta, na przykład "PN_Cw1_Jan_Masztalski".

Plik ze sprawozdaniem musi być przekazany do archiwum grupy.

a) Tworzenie środowiska do opracowania programów asemblerowych

Zadanie

Za pomocą Eksploratora Windows stworzyć "swój" folder.

Uwaga. Długość ścieżki od folderu musi być niewielka, a nazwa folderu - do 8 znaków.

W "swoim" folderu stworzyć foldery **BIN, LIB, INCLUDE**.

Ze środowiska MASM32 z odpowiednich folderów skopiować:

- do folderu **`BIN`** kompilator **`ml.exe`**, konsolidator **`link.exe,`** bibliotekę dynamiczną **`mspdb50.dll`**, oraz plik komunikatów o błędach **`ml.err`**,
- do folderu **`LIB`** biblioteki **`kernel32.lib, user32.lib`**,
- do folderu **`INCLUDE`** pliki nagłówkowe **`windows.inc, kernel32.inc, user32.inc`**.

Stworzyć w swoim folderu **folder** dla pliku „.asm” swojego zadania.

Uwaga. Nazwa pliku „.asm” z programem i nazwa folderu z tym plikiem powinny być jednakowe, aby zastosować pliki wsadowe, które są opisane niżej.

Opracowanie zadania i testowanie

<<zrzut ekranu Eksploratora Windows>>

---

b) Przejście do trybu konsolowego

Zadanie

Przejść do trybu konsolowego:
- przez wywołanie programu z nazwą *cmd.exe*, lub
- przez punkt menu Windows „Start/Wiersz poleceń”, lub
- przez edytor "MASM32 Editor" – *qeditor.exe* - środowiska MASM32.

Ostatni wariant jest lepszy, ponieważ przez punkt menu edytora "Set Current Directory" można szybko ustawiać aktualny folder.

W przypadku wywołania programu *cmd.exe* przejść do swojego folderu za pomocą polecenia „`cd` nazwa_swojego_folderu ”.

Uwaga. W przypadku braku doświadczenia w pracę z poleceniami MS DOS wpisać polecenie „help” i zapoznać się z listą poleceń MS DOS. Można skierować listę poleceń MS DOS do pliku tekstowego. W tym celu wprowadzić wiersz: `help` > *nazwa_pliku_tekstowego*

Opracowanie zadania i testowanie

<<zrzut ekranu MS DOS>>

---

c) Opcje kompilatora ml.exe

Zadanie

Otrzymać informację o strukturze wiersza poleceń oraz listę możliwych opcji asemblera `ml.exe` środowiska MASM32 przez wywołanie aplikacji z opcją:

```
.\BIN\ml.exe /?
```

Aby zapisać tą listę, skierować strumień wyjściowy do pliku, na przykład z nazwą „opc_ML.lst”:

```
.\BIN\ml.exe /? > opc_ML.lst
```

Opracowanie zadania

<<lista opcji asemblera `ml.exe`>>

---

d) Plik wsadowy do pracy z kompilatorem

Zadanie

Przygotować plik wsadowy do pracy z kompilatorem.

Korzystając z edytora tekstowego wpisać do pliku na przykład z nazwą „komp.bat”:

```
@echo off
if exist %1\%1.obj del %1\%1.obj
if exist %1\%1.exe del %1\%1.exe
@echo %1\%1
.\bin\ml /c /coff /Cp /Cx /Fo.\%1\%1.obj /Fl.\%1\%1.lst /Zi /Zd .\%1\%1.asm
```

Przy wywołaniu tego pliku wsadowego wyraz „`%1`” będzie zamieniony na parametr wywołania. W wyrazach „`%1\%1.`” parametr wywołania wystąpi jak nazwa folderu, oraz jak nazwa pliku.

Dla kompilatora są ustawione opcje:

`/c`	`Assemble without linking (kompilować bez konsolidacji)`
`/coff`	`Generate COFF format object file (format COFF pliku obiektowego)`
`/Cp`	`Preserve case of user identifiers (chronić wysokość liter w nazwach użytkownika),`
`/Cx`	`Preserve case in publics, externs (chronić wysokość liter w nazwach publicznych, zewnętrznych),`
`/Fl[file]`	`Generate listing (produkowanie listingu),`
`/Zd`	`Add line number debug info (dodać numery linijek dla wykrywacza usterek),`
`/Zi`	`Add symbolic debug info (dodać informację dla wykrywacza usterek)`

Aby wywołać kompilator za pomocą pliku wsadowego, należy podać polecenie:

`komp nazwa_programu`

Uwaga. Nazwa pliku swojego programu i nazwa folderu z tym plikiem powinny być jednakowe.

Opracowanie zadania

<<zawartość pliku wsadowego>>

---

e) Opcje konsolidatora link.exe

Zadanie

Otrzymać informację o strukturze wiersza poleceń oraz listę możliwych opcji konsolidatora link.exe środowiska MASM32 przez wywołanie aplikacji z opcją:

```
.\BIN\link.exe
```

Aby zapisać tą listę, skierować strumień wyjściowy do pliku, na przykład z nazwą opc_LINK.lst:

```
.\BIN\link.exe > opc_LINK.lst
```

Opracowanie zadania

<<lista opcji konsolidatora link.exe>>

---

f) Plik wsadowy do pracy z konsolidatorem

Zadanie

Przygotować plik wsadowy do pracy z konsolidatorem. Korzystając z edytora tekstowego wpisać do pliku na przykład z nazwą „kons.bat" (*uwaga:* wpisać w jednym wierszu!):

```
.\BIN\link /SUBSYSTEM:CONSOLE /LIBPATH:.\LIB /OUT:.\%1\%1.exe
/MAP:.\%1\%1.map /PDB:.\%1\%1.pdb .\%1\%1.obj
```

Dla konsolidatora są ustawione opcje:

`/SUBSYSTEM:CONSOLE`	aplikacja konsolowa
`/LIBPATH:[dir]`	ścieżka do wyszukiwania bibliotek
`/OUT:[file]`	nazwa pliku wynikowego
`/MAP:[file]`	nazwa pliku „.map" z mapą pamięci
`/PDB:[file]`	nazwa pliku „.pdb" (plik „baza danych aplikacji"; PDB – Program Database)

Aby wywołać konsolidator za pomocą pliku wsadowego, należy podać polecenie:

```
kons nazwa_programu
```

Uwaga. Nazwa pliku swojego programu i nazwa folderu z tym plikiem powinny być jednakowe.

Opracowanie zadania

<<zawartość pliku wsadowego>>

---

g) Program w języku asemblera

Zadanie

W folderu „nazwa_programu" prowadzić do pliku „nazwa_programu.asm" program do obliczenia zadanego wzoru – funkcji od argumentu X.

Struktura programu:

1) Wprowadzenie argumentu X.
2) Obliczenie funkcji.
3) Wyprowadzenie wzoru i wyniku.

Fragmenty programu są umieszczone niżej.

Dopełnić obliczeniami według swojego wariantu.

Kompilować program w oknie konsoli za pomocą pliku wsadowego komp.bat.

Konsolidować za pomocą pliku wsadowego kons.bat.

Uruchomić program w oknie konsoli. Pokazać prowadzającemu.

Opracowanie zadania

<<tekst programu>>

Testowanie

<<zrzut ekranu MS DOS>>

Tabela wariantów punktu g)

Np	Funkcja	Np	Funkcja
1	X + X + X * X	**17**	X + X + X / X
2	2X - X + X * X	**18**	2X - X + X / X
3	X * X + X * X	**19**	X * X + X / X
4	X / X + X * X	**20**	X / X + X / X
5	X + X - X * X	**21**	X + X - X / X
6	2X - X - X * X	**22**	2X - X - X / X
7	2X * X - X * X	**23**	X * X - X / X
8	X / X - X * X	**24**	2X / X - X / X
9	X + X * X * X	**25**	X + X * X / X
10	X - X * X * X	**26**	X - X * X / X
11	X * X * X * X	**27**	X * X * X / X
12	X / X * X * X	**28**	X / X * X / X
13	X + X / X * X	**29**	X + X / X / X
14	2X - X / X * X	**30**	X - X / X / X
15	X * X / X * X	**31**	X * X / X / X
16	X / X / X * X	**32**	X / X / X / X

Wskazówki

Firma Microsoft proponuje do kompilacji programów asemblerowych dla 32-bitowych procesorów Intel kompilator ml.exe (ML) i konsolidator link.exe (LINK).

1. Opcje asemblera ML

Asembler ML jest wywoływany z wiersza poleceń z następującą składnią:

```
ml.exe [ /options ] filelist [ /link linkoptions ]
```

Pole *filelist* zawiera listę plików do kompilacji.

Jeżeli plik źródłowy ma rozszerzenie „asm”, to rozszerzenie można nie wypisywać. Kompilator produkuje pliki z nazwami plików źródłowych i z rozszerzeniem „obj”.

Listę możliwych opcji asemblera ML można otrzymać przez wywołanie aplikacji *ml.exe* z opcją „/?”:

```
ml.exe /?
```

Aby zapisać tą listę, można skierować strumień wyjściowy do pliku, na przykład z nazwą opc_ML.lst:

```
ml.exe /? > opc_ML.lst
```

W tab. 1 jest umieszczona lista opcji asemblera ML wersji 6.14.

Przykładowo wiersz poleceń w przypadku pliku - źródła z nazwą „progr.asm” wygląda następująco:

```
ml.exe /c /coff /Cp /Cx /Zi /Zd prog.asm
```

Tabela 1

Lista opcji asemblera ML wersji 6.14

Opcja	Operacja	Opis
/AT	Enable tiny model (.COM file)	Produkować model “tiny”
/nologo	Suppress copyright message	Stłumić kopie komunikatów
/Bl<linker>	Use alternate linker	Użyć inny konsolidator
/Sa	Maximize source listing	Maksymalny listing
/c	Assemble without linking	Kompilować bez konsolidacji
/Sc	Generate timings in listing	Wyprodukować w listingu elementy synchronizacji
/Cp	Preserve case of user identifiers	Zachować wysokość liter w identyfikatorach użytkownika
/Sf	Generate first pass listing	Wyprodukować listing po fazie pierwszej
/Cu	Map all identifiers to upper case	Drukować wszystkie identyfikatory literami dużymi
/Sl<width>	Set line width	Ustawić szerokość linijki
/Cx	Preserve case in publics, externs	Chronić wysokość liter w identyfikatorach publicznych, zewnętrznych
/Sn	Suppress symbol-table listing	Stłumić w listingu tabelę identyfikatorów
/coff	Generate COFF format object file	Tworzyć plik obiektowy w formacie COFF
/Sp<length>	Set page length	Ustawić rozmiar strony
/D<name>[=text]	Define text macro	Zdefiniować makro tekstowe
/Ss<string>	Set subtitle	Ustawić podtytuł
/EP	Output preprocessed listing to stdout	Skierować listing do strumienia stdout
/St<string>	Set title	Ustawić tytuł
/F<hex>	Set stack size (bytes)	Ustawić rozmiar stosu (w bajtach)
/Sx	List false conditionals	Lista nieprawdziwych warunków
/Fe<file>	Name executable	Nazwa pliku wykonywalnego
/Ta<file>	Assemble non-.ASM file	Kompilować plik z innym rozszerzeniem niż „.asm”
/Fl[file]	Generate listing	Wyprodukować listing
/w	Same as /W0 /WX	To samo, co /W0 /WX
/Fm[file]	Generate map	Wyprodukować map - plik
/WX	Treat warnings as errors	Rozpatrywać ostrzeżenia jako błędy
/Fo<file>	Name object file	Nazwa pliku obiektowego
/W<number>	Set warning level	Ustawić poziom ostrzeżeń
/FPi	Generate 80x87 emulator encoding	Produkować emulację rozkazów 80x87
/X	Ignore INCLUDE environment path	Ignorować ścieżkę przestrzeni INCLUDE
/Fr[file]	Generate limited browser info	Produkować ograniczoną informację dla przeglądarki
/Zd	Add line number debug info	Dodać numery linijek do informacji debuggera
/FR[file]	Generate full browser info	Produkować pełną informację dla debuggera
/Zf	Make all symbols public	Zaznaczyć wszystkie identyfikatory jako publiczne
/G<c|d|z>	Use Pascal, C, or Stdcall calls	Stosować reguły wywołania Pascal, C lub Stdcall

/Zi	Add symbolic debug info	Dodać informację o identyfikatorach dla debuggera
/H<number>	Set max external name length	Ustawić maksymalną długość nazw
/Zm	Enable MASM 5.10 compatibility	Zezwolić kompatybilność z MASM 5.10
/I<name>	Add include path	Dodać ścieżkę przestrzeni INCLUDE
/Zp[n]	Set structure alignment	Ustawić krok rozmieszczenia struktur
/link <linker options and libraries>		Dołączyć opcje konsolidatora oraz biblioteki
/Zs	Perform syntax check only	Sprawdzić tylko błędy syntaktyczne

2. Opcje konsolidatora LINK

Po otrzymaniu pliku obiektowego z rozszerzeniem „.obj" należy wywołać konsolidator (linker) a w wierszu poleceń wskazać nazwę pliku i opcji konsolidacji.

Wiersz poleceń konsolidatora *Incremental Linker (LINK)* ma następującą składnie:

```
link.exe [Opcje] plik_źródlowy [ pliki_źródlowe]
```

Jeżeli plik źródłowy ma rozszerzenie „obj", to rozszerzenie można nie wypisywać. Jeżeli w opcjach nie będzie napisana nazwa pliku wynikowego, to konsolidator wyprodukuje plik z nazwą pierwszego pliku źródłowego i z rozszerzeniem „exe".

Można stosować plik pośredni:

```
link.exe @plik_posredni
```

Listę możliwych opcji konsolidatora LINK można otrzymać przez wywołanie konsolidatora bez opcji:

```
link.exe
```

Aby zapisać tą listę, można skierować strumień wyjściowy do pliku, na przykład z nazwą opc_LINK.lst:
link.exe > opc_LINK.lst

Lista opcji konsolidatora LINK wersji 5.12 jest umieszczona w tab. 2.

Można polecić dołączenie opcji:

`/MAP[:filename]` do produkowania mapy pamięci (pliku z rozszerzeniem .map),

`/PDB:{filename}` do dodania pełnej informacji dla wykrywacza usterek.

Konsolidator LINK korzysta z pliku „.def" do definicji resursów (ikon, kursorów, wierszy, okien dialogu itp.).

Tabela 2

Lista opcji konsolidatora LINK wersji 5.12

Opcja	Opis
/ALIGN:nn	Krok rozmieszczenia sekcji równy nn. wartość domyślna 4096
/BASE:{address|@filename,key}	Adres do załadowania EXE (domyślnie 400000h) lub DLL (domyślnie 10000000h). Adres DLL może być w pliku „filename" w wierszu „key"
/COMMENT:comment	(opcja nie jest stosowana)
/DEBUG	Przygotować dane dla debuggera w pliku „.pdb", a do EXE lub DLL dołączyć odsyłacz
/DEBUGTYPE:{CV|COFF}	(opcja wyeliminowana)
/DEF:filename	Nazwa pliku „.def" (module-definition file)
/DEFAULTLIB:library	Nazwa biblioteki, która jest stosowana poza bibliotekami wskazanymi w wierszu poleceń
/DLL	Nawiązuje tworzenie DLL
/DRIVER[:{UPONLY|WDM}]	Konsolidacja sterownika dla Windows NT
/ENTRY:symbol	Funkcja startowa dla EXE lub DLL
/EXETYPE:DYNAMIC	Nawiazuje konsolidację sterownika wirtualnego
/EXPORT:symbol	Funkcja lub zmienna eksportowana z DLL
/FIXED[:NO]	Konsolidować EXE lub DLL, która będzie ładowana z adresu opcji /BASE
/FORCE[:{MULTIPLE|UNRESOLVED}]	Konsolidować w warunkach braku importowanych funkcji
/GPSIZE:#	(opcja wyeliminowana)
/HEAP:reserve[,commit]	Definuje rozmiar sterty (domyślnie 1 MB)
/IMPLIB:filename	Nazwa produkowanej biblioteki importu
/INCLUDE:symbol	Dodać do tablicy nazw
/INCREMENTAL:{YES|NO}	Wyznaczyć tryb przyrostkowy
/LARGEADDRESSAWARE[:NO]	Informowanie o tym, że aplikacja może operować adresami większymi niż 2 GB
/LIBPATH:dir	Ścieżka do wyszukiwania bibliotek
/MACHINE:{ALPHA|ARM|IX86|MIPS|MIPS16|MIPSR41XX|PPC|SH3|SH4}	Specyfikuje typ platformy sprzętowej
/MAP[:filename]	Nazwa pliku „.map" z mapą pamięci
/MAPINFO:{EXPORTS|FIXUPS|LINES}	Specyfikuje rodzaj pliku „.map"
/MERGE:from=to	Połączenie sekcji
/NODEFAULTLIB[:library]	Nazwa biblioteki, którą linker zignoruje
/NOENTRY	Informuję, że DLL zawiera tylko resursy
/NOLOGO	Tłumi wyświetlenie informacji o wersji i prawach kopiowania
/OPT:{ICF[,iterations]|NOICF|NOREF|NOWIN98|REF|WIN98}	Definuje rodzaj optymizacji
/ORDER:@filename	Ustawia kolejność właczenia funkcji
/OUT:filename	Nazwa pliku wynikowego
/PDB:{filename|NONE}	Nazwa pliku „.pdb" (plik „baza danych aplikacji"; PDB – Program Database)
/PDBTYPE:{CON[SOLIDATE]|SEPT[YPES]}	Definuje rodzaj pliku „.pdb"
/PROFILE	Specyfikuje typ sekcji przemieszczalnej
/RELEASE	Definuje dołaczenie sumy kontrolnej (dla sterownika)

/SECTION:name, [E][R][W][S][D] [K][L][P][X]	Specyfikuje nazwę i atrybuty sekcji
/STACK:reserve[,commit]	Definuje rozmiar stosu
/STUB:filename	Nazwa stub–exe serwera
/SUBSYSTEM:{NATIVE| WINDOWS|CONSOLE| WINDOWSCE|POSIX}[,#[.##]]	Definuje rodzaj aplikacji
/SWAPRUN:{CD|NET}	Wskazuje, że aplikacja musi być najpierw skopiowana (np. z sieci), a później wywołana
/VERBOSE[:LIB]	Informować komunikatami o przebiegu konsolidacji
/VERSION:#[.#]	Dołączyć do aplikacji numer wersji
/VXD	Nawiązuje kreację sterowanika wirtualnego
/WARN[:warninglevel]	(opcja wyeliminowana)
/WINDOWSCE:{CONVERT| EMULATION}	(opcja nie jest stosowana)
/WS:AGGRESSIVE	Nie generować plik wykonywalny, jeśli były ostrzeżenia

Program przykładowy

```
;Aplikacja korzystająca z otwartego okna konsoli
.586
.MODEL flat, STDCALL
;--- stale ---
;--- z pliku windows.inc ---
STD_INPUT_HANDLE                    equ          -10
STD_OUTPUT_HANDLE                   equ   -11
;--- funkcje API Win32 ---
;--- z pliku  user32.inc ---
CharToOemA PROTO :DWORD,:DWORD
;--- z pliku kernel32.inc ---
GetStdHandle PROTO :DWORD
ReadConsoleA PROTO :DWORD,:DWORD,:DWORD,:DWORD,:DWORD
WriteConsoleA PROTO :DWORD,:DWORD,:DWORD,:DWORD,:DWORD
ExitProcess PROTO :DWORD
wsprintfA PROTO C :VARARG
lstrlenA PROTO :DWORD
ScanIntPROTO C     adres:DWORD
;-------------
includelib .\lib\user32.lib
includelib .\lib\kernel32.lib
;-------------
_DATA SEGMENT
 hout  DD    ?
 hinp  DD    ?
 naglow      DB    "Autor aplikacji ....",0 ; nagłówek
 zaprX DB    0Dh,0Ah,"Proszę wprowadzić argument X [+Enter]: ",0 ; zaproszenie
 wzor  DB    0Dh,0Ah,"Funkcja f(X) = ...... = %ld ",0 ; tekst formatujący
```

```
 ALIGN 4     ;wyrównanie do granicy 4-bajtowej
 rozmN DD    0 ;ilość znaków w nagłówku
 rozmX DD    0 ;ilość znaków w zaproszeniu X
 zmX   DD    1     ; argument X
 rout  DD    0 ;faktyczna ilość wyprowadzonych znaków
 rinp  DD    0 ;faktyczna ilość wprowadzonych znaków
 bufor DB    128 dup(0) ;rezerwacja miejsca na bufor i inicjalizacja 0
 rbuf  DD    128 ;rozmiar bufora
_DATA ENDS
_TEXT SEGMENT
start:
;--- wywołanie funkcji GetStdHandle
 push  STD_OUTPUT_HANDLE ;odkładanie na stos
 call  GetStdHandle      ; funkcja GetStdHandle = podaj deskryptor ekranu
 mov   hout, EAX   ; deskryptor wyjściowego bufora konsoli
 push  STD_INPUT_HANDLE ;odkładania na stos
 call  GetStdHandle      ; funkcja GetStdHandle = podaj deskryptor klawiatury
 mov   hinp, EAX   ; deskryptor wejściowego bufora konsoli
;--- nagłówek ---------
 push  OFFSET naglow ;odkładanie na stos
 push  OFFSET naglow ;odkładanie na stos
 call  CharToOemA  ;wywołanie funkcji konwersji polskich znaków
;--- wyświetlenie ---------
   push   OFFSET naglow
   call   lstrlenA ;wywołanie funkcji
   mov   rozmN, EAX   ;ilość znaków
 push  0           ;odkładanie na stos: rezerwa, musi być zero
 push  OFFSET rout ;odkładanie na stos: wskaźnik na faktyczną ilość
wyprowadzonych znaków
 push  rozmN       ;odkładanie na stos: ilość znaków
 push  OFFSET naglow     ;odkładanie na stos: wskażnik na tekst
 push  hout        ;odkładanie na stos: deskryptor wyjściowego buforu konsoli
 call  WriteConsoleA     ; wywołanie funkcji WriteConsoleA
;--- zaproszenie A ---------
 push  OFFSET zaprX ;odkładanie na stos
 push  OFFSET zaprX ;odkładanie na stos
 call  CharToOemA  ; wywołanie funkcji konwersji polskich znaków
;--- wyświetlenie zaproszenia A ---
 push   OFFSET zaprX ;odkładanie na stos
 call   lstrlenA
 mov   rozmX, EAX   ;ilość znaków z akumulatora do pamięci
 push  0           ; rezerwa, musi być zero
 push  OFFSET rout       ; wskaźnik na faktyczną ilość wyprowadzonych znaków
 push  rozmX       ; ilość znaków
```

```
 push  OFFSET zaprX      ; wskaźnik na tekst
 push  hout        ; deskryptor buforu konsoli
 call  WriteConsoleA     ; funkcja WriteConsoleA = wyświetlenie na ekranie
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 push  0           ; rezerwa, musi być zero
 push  OFFSET rinp       ; wskaźnik na faktyczną ilość wprowadzonych znaków
 push  rbuf        ; rozmiar bufora
 push  OFFSET bufor ;wskaźnik na bufor
 push  hinp        ; deskryptor buforu konsoli
 call  ReadConsoleA      ; wywołanie funkcji ReadConsoleA = odczyt z kławiatury
 lea   EBX,bufor
 mov   EDI,rinp
 mov   BYTE PTR [EBX+EDI-1],0 ;zero na końcu tekstu
;--- przekształcenie A
 push  OFFSET bufor ;odkładanie na stos
 call  ScanInt ; wywołanie funkcji przekształcenie tekstu do postaci binarnej
 add   ESP, 8
 mov   zmX, EAX
;--- obliczenia ---
 mov   EAX, zmX
;;;;   ................
;--- wyprowadzenie wyniku obliczeń ---
 push  EAX ;odkładanie na stos
 push  OFFSET wzor ;odkładanie na stos
 push  OFFSET bufor ;odkładanie na stos
 call  wsprintfA   ;funkcja przekształcenia liczby; zwraca ilość znaków
 add   ESP, 12           ; czyszczenie stosu
 mov   rinp, EAX   ; zapamiętywanie ilości znaków
;--- wyświetlenie wynika ---------
 push  0           ; rezerwa, musi być zero
 push  OFFSET rout       ; wskaźnik na faktyczną ilość wyprowadzonych znaków
 push  rinp        ; ilość znaków
 push  OFFSET bufor      ; wskaźnik na tekst w buforze
 push  hout        ; deskryptor buforu konsoli
 call  WriteConsoleA     ; wywołanie funkcji WriteConsoleA
;--- zakończenie procesu ---------
 push  0
 call  ExitProcess ; wywołanie funkcji ExitProcess
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ScanInt PROC C adres
;; funkcja ScanInt przekształca ciąg cyfr do liczby, którą będzie w EAX
;; argument - zakończony zerem wiersz z cyframi
;; rejestry: EBX - adres wiersza, EDX - znak liczby, ESI - indeks cyfry w
wierszu, EDI - tymczasowy
```

```
;--- początek funkcji
LOCAL number,znacz
;--- odkładanie na stos
 push  EBX
 push  ECX
 push  EDX
 push  ESI
 push  EDI
;--- przygotowywanie cyklu
 INVOKE lstrlenA, adres
 mov   EDI, EAX    ;ilość znaków
 mov   ECX, EAX    ;ilość powtórzeń = ilość znaków
 xor   ESI, ESI    ; wyzerowanie ESI
 xor   EDX, EDX    ; wyzerowanie EDX
 xor   EAX, EAX    ; wyzerowanie EAX
 mov   EBX, adres
 ;-----------
 mov   znacz,0
 mov   number,0
;--- cykl --------------------------
pocz:
   cmp   BYTE PTR [EBX+ESI], 0h ; porównanie z kodem \0
   jne   @F
   jmp   et4
@@:
   cmp   BYTE PTR [EBX+ESI], 0Dh ; porównanie z kodem CR
   jne   @F
   jmp   et4
@@:
   cmp   BYTE PTR [EBX+ESI], 0Ah   ; porównanie z kodem LF
   jne   @F
   jmp   et4
@@:
 cmp   BYTE PTR [EBX+ESI], 02Dh     ; porównanie z kodem '-'
 jne   @F
 mov   znacz, 1
 jmp   nast
@@:    cmp   BYTE PTR [EBX+ESI], '0' ; porównanie z kodem '0'
 jae   @F
 jmp   nast
@@:    cmp   BYTE PTR [EBX+ESI], '9' ; porównanie z kodem '9'
 jbe   @F
 jmp   nast
;----
```

```
@@:    push  EDX   ; do EDX procesor może zapisać wynik mnożenia
 mov   EAX,number
 mov   EDI, 10
 mul   EDI         ; mnożenie EAX * (EDI=10)
 mov   number, EAX ; tymczasowo z EAX do EDI
 xor   EAX, EAX    ; zerowanie EAX
 mov   AL, BYTE PTR [EBX+ESI]
 sub   AL, '0'     ; korekta: cyfra = kod znaku - kod '0'
 add   number,EAX  ; dodanie cyfry
 pop   EDX
nast:  inc   ESI
      dec   ECX
      jz    @F
      jmp   pocz
;--- wynik
@@:
et4:
 cmp   znacz,1     ; analiza znacznika
 jne   @F
 neg   number
@@:
 mov   EAX,number
;--- zdejmowanie ze stosu
 pop   EDI
 pop   ESI
 pop   EDX
 pop   ECX
 pop   EBX
;--- powrót
 ret
ScanIntENDP
_TEXT  ENDS
END start
```