PRYWATNA WYŻSZA SZKOŁA NAUK SPOŁECZNYCH, KOMPUTEROWYCH I MEDYCZNYCH

WYDZIAŁ NAUK SPOŁECZNYCH I TECHNIK KOMPUTEROWYCH

Ćwiczenie

z programowania niskopoziomowego

„Instrukcje arytmetyczne i logiczne. Przesuwanie i rotacja bitów.”

Wariant N 8

Opracował

Grzegorz Makowski

III rok Informatyki
Studia niestacjonarne

Prowadzący
Prof. dr hab. inż. Aleksandr Timofiejew

Warszawa 2019/2020

Spis treści

Zadanie a.

Operacje arytmetyczne.

Napisać program, w którym obliczyć funkcję y od czterech argumentów całkowitych a, b, c, d według wzoru dla swojego zadania (patrz tabele wariantów). Argumenty wprowadzać pojedynczo wyświetlając zachętę na wprowadzenie każdego argumentu.
Wyświetlić wynik obliczeń.

Opracowanie zadania

```
;Aplikacja "Instrukcje arytmetyczne i logiczne. Przesuwanie i rotacja bitów"
.586
.MODEL flat, STDCALL
;--- stale ---
;--- z pliku ..\include\windows.inc ---
STD_INPUT_HANDLE   equ -10
STD_OUTPUT_HANDLE  equ -11
;--- funkcje API Win32 ---
;--- z pliku  ..\include\user32.inc ---
CharToOemA PROTO :DWORD,:DWORD
;--- z pliku ..\include\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
;-------------
includelib .\lib\user32.lib
includelib .\lib\kernel32.lib
;-------------
_DATA SEGMENT
hout DD ?
hinp DD ?
naglow DB "Autor aplikacji Grzegorz Makowski i53",0
; nagłówek
wzor DB 0Dh,0Ah,"Instrukcje arytmetyczne i logiczne.",0
; tekst formatujący
ALIGN 4
; wyrównanie do granicy 4-bajtowej
rozmN DD $ - naglow
; ilosc znakow w tablocy
nowa DB 0Dh, 0Ah, 0
ALIGN 4
; przesuniece do adresu podzielonego na 4

zadA DB "Zadanie a) ",0
; nagłówek zadania A
rozmzadA DD $ - zadA
naglowA DB 0Dh,0Ah, "Wprowadź 4 parametry do fun. f() = A/B-C+D",0
; nagłówek
rozmnaglA DD $ - naglowA

zaproszenieA DB 0Dh,0Ah,"Proszę wprowadzić argument A [+Enter]: ",0
; Tekst zaproszenia do wprowadzenie zamiennej A
rozmiarA DD $ - zaproszenieA
zmA DD 1
; argument A

zaproszenieB DB 0Dh,0Ah,"Proszę wprowadzić argument B [+Enter]: ",0
; Tekst zaproszenia do wprowadzenie zmiennej B
rozmiarB DD $ - zaproszenieB
zmB DD 1
; argument B
```

```
zaproszenieC DB 0Dh,0Ah,"Proszę wprowadzić argument C [+Enter]: ",0
; Tekst zaproszenia do wprowadzenie zamiennej C
rozmiarC DD $ - zaproszenieC
zmC DD 1
; argument C

zaproszenieD DB 0Dh,0Ah,"Proszę wprowadzić argument D [+Enter]: ",0
; Tekst zaproszenia do wprowadzenie zmiennej D
rozmiarD DD $ - zaproszenieD
zmD DD 1

wynik DB 0Dh,0Ah,"Wynik f() =  %ld",0
rout DD 0
; faktyczna ilość wyprowadzonych znaków
rinp DD 0
; faktyczna ilość wprowadzonych znaków
rbuf DD 128
buforDB    128 dup(?)

;zadB DB "Zadanie b) ",0
; nagłówek zadania B
;rozmB DD $ - zadB
tekstZakoncz DB "Dziękuję za uwagę! PWSBIA@2020",0
rozmZ DD $ - tekstZakoncz
_DATA ENDS
;------------
_TEXT SEGMENT
start:
;--- deskryptory konsoli
push STD_OUTPUT_HANDLE
callGetStdHandle ; wywołanie funkcji GetStdHandle
movhout, EAX       ; deskryptor wyjściowego bufora konsoli
push STD_INPUT_HANDLE
callGetStdHandle ; wywołanie funkcji GetStdHandle
movhinp, EAX       ; deskryptor wejściowego bufora konsoli
;--- nagłówek ---------
push OFFSET naglow
push OFFSET naglow
callCharToOemA    ; konwersja polskich znaków
;--- wyświetlenie ---------
push 0         ; rezerwa, musi być zero
push OFFSET rout     ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmN      ; ilość znaków
push OFFSET naglow ; wskaźnik na tekst
push hout       ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- wyświetlenie nową linii ---
push 0         ; rezerwa, musi być zero
push OFFSET rout  ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2         ; ilość znaków
push OFFSET nowa ; wskaźnik na tekst
push hout       ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- opis funkcji programu ---------

push OFFSET zadA
push OFFSET ZadA
call CharToOemA         ; konwersja polskich znaków
push 0                  ; rezerwa, musi być zero
push OFFSET rout        ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmzadA           ; ilość znaków
push OFFSET ZadA        ; wskaźnik na tekst
push hout               ; deskryptor buforu konsoli
call WriteConsoleA
;--- wyświetlenie nową linii ---
push 0         ; rezerwa, musi być zero
push OFFSET rout  ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2         ; ilość znaków
push OFFSET nowa ; wskaźnik na tekst
push hout       ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- Zaproszenie do zadania A ---------
push OFFSET naglowA
push OFFSET naglowA
call CharToOemA         ; konwersja polskich znaków
```

```
push 0                ; rezerwa, musi być zero
push OFFSET rout      ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmnaglA        ; ilość znaków
push OFFSET naglowA   ; wskaźnik na tekst
push hout             ; deskryptor buforu konsoli
call WriteConsoleA
;--- wyświetlenie nową linii ---
push 0        ; rezerwa, musi być zero
push OFFSET rout  ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2        ; ilość znaków
push OFFSET nowa ; wskaźnik na tekst
push hout     ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- wyświetlenie nową linii ---
push 0        ; rezerwa, musi być zero
push OFFSET rout    ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2        ; ilość znaków
push OFFSET nowa ; wskażnik na tekst
push hout     ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- zaproszenie A ---------
push OFFSET zaproszenieA
push OFFSET zaproszenieA
call CharToOemA       ; konwersja polskich znaków
;--- wyświetlenie zaproszenia A ---
push 0                ; rezerwa, musi być zero
push OFFSET rout      ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmiarA         ; ilość znaków
push OFFSET zaproszenieA ; wskaźnik na tekst
push hout             ; deskryptor buforu konsoli
call WriteConsoleA    ; wywołanie funkcji WriteConsoleA
;--- 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
lea EBX,bufor
mov EDI,rinp
mov BYTE PTR [EBX+EDI-1],0  ;zero na końcu tekstu
;--- przekształcenie A
push OFFSET bufor
call ScanInt
add ESP, 8
mov zmA, EAX
;;;..... B
;--- zaproszenie B---------
push OFFSET zaproszenieB
push OFFSET zaproszenieB
call CharToOemA       ; konwersja polskich znaków
;--- wyświetlenie zaproszenia B ---
push 0                ; rezerwa, musi być zero
push OFFSET rout      ; wskaźnik na faktycznš ilość wyprowadzonych znaków
push rozmiarB         ; ilość znaków
push OFFSET zaproszenieB ; wskaźnik na tekst
push hout             ; deskryptor buforu konsoli
call WriteConsoleA    ; wywołanie funkcji WriteConsoleA
;--- 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
lea EBX,bufor
mov EDI,rinp
mov BYTE PTR [EBX+EDI-1],0 ;zero na końcu tekstu
;--- przekształcenie B
push OFFSET bufor
call ScanInt
add ESP, 8
mov zmB, EAX
;--- zaproszenie C ---------
push OFFSET zaproszenieC
push OFFSET zaproszenieC
```

```
call CharToOemA          ; konwersja polskich znaków
;--- wyświetlenie zaproszenia C ---
push 0                   ; rezerwa, musi być zero
push OFFSET rout         ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmiarC            ; ilość znaków
push OFFSET zaproszenieC ; wskaźnik na tekst
push hout                ; deskryptor buforu konsoli
call WriteConsoleA       ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
push 0                   ; rezerwa, musi być zero
push OFFSET rinp         ; wskaźnik na faktycznš ilość wprowadzonych znaków
push rbuf                ; rozmiar bufora
push OFFSET bufor        ; wskaźnik na bufor
push hinp                ; deskryptor buforu konsoli
call ReadConsoleA        ; wywołanie funkcji ReadConsoleA
lea EBX,bufor
mov EDI,rinp
mov BYTE PTR [EBX+EDI-1],0  ;zero na końcu tekstu
;--- przekształcenie C
push OFFSET bufor
call ScanInt
add ESP, 8
mov zmC, EAX
;--- zaproszenie D ---------
push OFFSET zaproszenieD
push OFFSET zaproszenieD
call CharToOemA          ; konwersja polskich znaków
;--- wyświetlenie zaproszenia D ---
push 0                   ; rezerwa, musi być zero
push OFFSET rout         ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmiarD            ; ilość znaków
push OFFSET zaproszenieD ; wskaźnik na tekst
push hout                ; deskryptor buforu konsoli
call WriteConsoleA       ; wywołanie funkcji WriteConsoleA
;--- 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
lea EBX,bufor
mov EDI,rinp
mov BYTE PTR [EBX+EDI-1],0  ;zero na końcu tekstu
;--- przekształcenie D
push OFFSET bufor
call ScanInt
add ESP, 8
mov zmD, EAX
;--- obliczenia A / B - C + D
mov EDX,0           ; zerowanie edx
mov EAX, zmA        ; zm a do eax
div zmB             ; dzielimy a / b wynik z eax
mov EDX, 0          ; zerowanie edx
mov EDX, zmD        ; zmienna c do edx
add EDX, zmC        ; dodanie c do D wynik w edx
sub EAX, EDX        ; odejmowanie wyik dzielenia minus wynik dodawania wyik w eax
;--- wyprowadzenie wyniku obliczeń ---
push EAX
push OFFSET wynik
push OFFSET bufor
call wsprintfA          ; zwraca ilość znaków w buforze
add ESP, 16             ; 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
;--- wyświetlenie nową linii ---
push 0         ; rezerwa, musi być zero
push OFFSET rout  ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2         ; ilość znaków
push OFFSET nowa ; wskaźnik na tekst
```

```
push hout       ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- wyświetlenie nową linii ---
push 0          ; rezerwa, musi być zero
push OFFSET rout  ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2          ; ilość znaków
push OFFSET nowa ; wskażnik na tekst
push hout       ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- wyswietlenie opisu zadania b ---------
;push OFFSET zadB
;push OFFSET ZadB
;call CharToOemA        ; konwersja polskich znaków
;push 0                 ; rezerwa, musi być zero
;push OFFSET rout       ; wskaźnik na faktyczną ilość wyprowadzonych znaków
;push rozmB             ; ilość znaków
;push OFFSET ZadB       ; wskażnik na tekst
;push hout              ; deskryptor buforu konsoli
;call WriteConsoleA
;--- wyświetlenie nową linii ---
;push 0          ; rezerwa, musi być zero
;push OFFSET rout  ; wskaźnik na faktyczną ilość wyprowadzonych znaków
;push 2          ; ilość znaków
;push OFFSET nowa ; wskażnik na tekst
;push hout       ; deskryptor buforu konsoli
;call WriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- Zadanie b)----
;--- wyświetlenie nowej lini ---------
;push 0                 ; rezerwa, musi być zero
; push OFFSET rout      ; wskaźnik na faktyczną ilość wyprowadzonych znaków
; push 2                ; ilość znaków
; push OFFSET nowa      ; wskaźnik na tekst
; push hout             ; deskryptor buforu konsoli
; call WriteConsoleA
;--- wyświetlenie zakonczenia ---
push 0                 ; rezerwa, musi być zero
push OFFSET rout       ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmZ
push OFFSET tekstZakoncz
push OFFSET tekstZakoncz
call CharToOemA
push 0                 ; rezerwa, musi być zero
push OFFSET rout       ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmZ             ; ilość znaków
push OFFSET tekstZakoncz       ; wskaźnik na tekst
push hout              ; deskryptor buforu konsoli
call WriteConsoleA      ; wywołanie funkcji WriteConsole
;--- zakończenie procesu ---------
push 0
callExitProcess  ; wywołanie funkcji ExitProcess
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ScanInt PROC
;; funkcja ScanInt przekształca ciąg cyfr do liczby, które jest zwracana przez EAX
;; argument - zakończony zerem wiersz z cyframi
;; rejestry: EBX - adres wiersza, EDX - znak liczby, ESI - indeks cyfry w wierszu, EDI - tymczasowy
;--- poczštek funkcji
push EBP
mov EBP, ESP         ; wskzźnik stosu ESP przypisujemy do EBP
;--- odkładanie na stos
push EBX
push ECX
push EDX
push ESI
push EDI
;--- przygotowywanie cyklu
mov EBX, [EBP+8]
push EBX
call lstrlenA
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, [EBP+8]     ; adres tekstu
;--- 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 EDX, 1
jmp nast
@@:
cmp BYTE PTR [EBX+ESI], 030h ;porównanie z kodem 0
jae @F
jmp nast
@@:
cmp BYTE PTR [EBX+ESI], 039h ;porównanie z kodem 9
jbe @F
jmp nast
;----
@@:
push EDX            ; do EDX procesor może zapisać wynik mnożenia
mov EDI, 10
mul EDI             ; mnożenie EAX * EDI
mov EDI, EAX        ; tymczasowo z EAX do EDI
xor EAX, EAX        ; zerowani EAX
mov AL, BYTE PTR [EBX+ESI]
sub AL, 030h        ; korekta: cyfra = kod znaku - kod 0
add EAX, EDI        ; dodanie cyfry
pop EDX
nast:
inc ESI
loop pocz
;--- wynik
or EDX, EDX    ;analiza znacznika EDX
jz @F
neg EAX
@@:
et4:
;--- zdejmowanie ze stosu
pop EDI
pop ESI
pop EDX
pop ECX
pop EBX
;--- powrót
mov ESP, EBP        ; przywracamy wskaźnik stosu ESP
pop EBP
ret
ScanInt ENDP
_TEXT ENDS
END start
```

Testowanie

```
 Assembling: .\cw4\cw4.asm

D:\workspace\Programowanie\rok3\asm>kons cw4

D:\workspace\Programowanie\rok3\asm>.\BIN\link /SUBSYSTEM:CONSOLE /LIBPATH:.\
PDB:.\cw4\cw4.pdb .\cw4\cw4.obj
Microsoft (R) Incremental Linker Version 5.12.8078
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.


D:\workspace\Programowanie\rok3\asm>cw4\cw4
Autor aplikacji Grzegorz Makowski i53
Instrukcje arytmetyczne i logiczne.
Zadanie a)

Wprowadź 4 parametry do fun. f() = A/B-C+D


Proszę wprowadzić argument A [+Enter]:  50

Proszę wprowadzić argument B [+Enter]:  50

Proszę wprowadzić argument C [+Enter]:  4

Proszę wprowadzić argument D [+Enter]:  4

Wynik f() =  -7

Dziękuję za uwagę! PWSBIA@2020
D:\workspace\Programowanie\rok3\asm>
```

Zadanie b

Operacje logiczne

Napisać program, w którym obliczyć funkcję y od czterech argumentów logicznych a, b, c, d według wzoru dla swojego zadania (patrz tabele wariantów), na przykład:
y = (a | b) * c # d
Znaki operacji w tabelę wariantów:
* - mnożenie logiczne bitowe („AND”),
| - suma logiczna bitowa („OR”),
- suma symetryczna bitowa („XOR”),
~ - negacja logiczna bitowa („NOT”).
Argumenty wprowadzać pojedynczo wyświetlając zachętę na wprowadzenie każdego argumentu.
Wartości logiczne wprowadzać za pomocą liczb "0" i "1".
Wyświetlić wynik obliczeń.
Uwaga. W wyniku operacji NOT ma miejsce negacja we wszystkich bitach, w tym w starszym bicie, i liczba po negacji jest interpretowana jako ujemna.
Negacja bitowa liczby 0 daje w wyniku liczbę -1, a negacja bitowa liczby 1 daje w wyniku liczbę -2.
Jeśli po operacji NOT dodamy liczbę 2, to otrzymamy poprawny wynik:

```
mov EAX, a ; logiczna zmienna a przypisana do rejestru EAX
not EAX ; negacja bitowa
add EAX, 2 ; korekta wyniku negacji; w EAX negacja zmiennej a.
```

Opracowanie zadania

```
;Aplikacja "Instrukcje arytmetyczne i logiczne. Przesuwanie i rotacja bitów"
.586
.MODEL flat, STDCALL
;--- stale ---
;--- z pliku ..\include\windows.inc ---
STD_INPUT_HANDLE                    equ -10
STD_OUTPUT_HANDLE                   equ -11
;--- funkcje API Win32 ---
;--- z pliku  ..\include\user32.inc ---
CharToOemA PROTO :DWORD,:DWORD
;--- z pliku ..\include\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
;-------------
includelib .\lib\user32.lib
includelib .\lib\kernel32.lib
;-------------
```

```
_DATA SEGMENT
hout DD ?
hinp DD ?
naglow DB "Autor aplikacji Grzegorz Makowski i53",0              ; nagłówek
wzor DB 0Dh,0Ah,"Instrukcje arytmetyczne i logiczne.",0                          ; tekst formatujący
ALIGN 4                                          ; wyrównanie do granicy 4-bajtowej
rozmN DD $ - naglow                                    ; ilosc znakow w tablocy
nowa DB 0Dh, 0Ah, 0
ALIGN 4                                          ; przesuniece do adresu podzielonego na 4

zadA DB "Zadanie b) ",0                                  ; nagłówek zadania A
rozmzadA DD $ - zadA
naglowA DB 0Dh,0Ah, "Wprowadź 4 parametry do fun. f() = a # b * ~c | d",0  ; nagłówek
rozmnaglA DD $ - naglowA

zaproszenieA DB 0Dh,0Ah,"Proszę wprowadzić argument a [+Enter]: ",0 ; Tekst zaproszenia do wprowadzenie amiennej A
rozmiarA DD $ - zaproszenieA
zmA DD 1                                         ; argument A

zaproszenieB DB 0Dh,0Ah,"Proszę wprowadzić argument b [+Enter]: ",0 ; Tekst zaproszenia do wprowadzenie amiennej B
rozmiarB DD $ - zaproszenieB
zmB DD 1                                         ; argument B

zaproszenieC DB 0Dh,0Ah,"Proszę wprowadzić argument c [+Enter]: ",0 ; Tekst zaproszenia do wprowadzenie amiennej C
rozmiarC DD $ - zaproszenieC
zmC DD 1                                         ; argument C

zaproszenieD DB 0Dh,0Ah,"Proszę wprowadzić argument d [+Enter]: ",0 ; Tekst zaproszenia do wprowadzenie amiennej D
rozmiarD DD $ - zaproszenieD
zmD DD 1

wynik DB 0Dh,0Ah,"Wynik f() =  %ld",0
rout DD 0                                        ; faktyczna ilość wyprowadzonych znaków
rinp DD 0                                        ; faktyczna ilość wprowadzonych znaków
rbuf DD 128
buforDB    128 dup(?)

tekstZakoncz DB "Dziękuję za uwagę! PWSBIA@2020",0                ; nagłówek
rozmZ DD $ - tekstZakoncz
_DATA ENDS
;------------
_TEXT SEGMENT
start:
;--- deskryptory konsoli
push STD_OUTPUT_HANDLE
callGetStdHandle ; wywołanie funkcji GetStdHandle
movhout, EAX      ; deskryptor wyjściowego bufora konsoli
push STD_INPUT_HANDLE
callGetStdHandle ; wywołanie funkcji GetStdHandle
movhinp, EAX      ; deskryptor wejściowego bufora konsoli
;--- nagłówek ---------
push OFFSET naglow
push OFFSET naglow
callCharToOemA      ; konwersja polskich znaków
;--- wyświetlenie ---------
push 0       ; rezerwa, musi być zero
push OFFSET rout       ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmN       ; ilość znaków
push OFFSET naglow ; wskaźnik na tekst
push hout        ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- wyświetlenie nową linii ---
push 0       ; rezerwa, musi być zero
push OFFSET rout  ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2       ; ilość znaków
push OFFSET nowa ; wskaźnik na tekst
push hout        ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- opis funkcji programu ---------

push OFFSET zadA
push OFFSET ZadA
call CharToOemA              ; konwersja polskich znaków
push 0                     ; rezerwa, musi być zero
push OFFSET rout             ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmzadA                 ; ilość znaków
```

```
push OFFSET ZadA              ; wskaźnik na tekst
push hout                     ; deskryptor buforu konsoli
call WriteConsoleA
;--- wyświetlenie nową linii ---
push 0        ; rezerwa, musi być zero
push OFFSET rout  ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2        ; ilość znaków
push OFFSET nowa ; wskaźnik na tekst
push hout         ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- Zaproszenie do zadania A ---------
push OFFSET naglowA
push OFFSET naglowA
call CharToOemA               ; konwersja polskich znaków
push 0                        ; rezerwa, musi być zero
push OFFSET rout              ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmnaglA                ; ilość znaków
push OFFSET naglowA           ; wskaźnik na tekst
push hout                     ; deskryptor buforu konsoli
call WriteConsoleA
;--- wyświetlenie nową linii ---
push 0        ; rezerwa, musi być zero
push OFFSET rout  ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2        ; ilość znaków
push OFFSET nowa ; wskaźnik na tekst
push hout         ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- wyświetlenie nową linii ---
push 0          ; rezerwa, musi być zero
push OFFSET rout        ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2          ; ilość znaków
push OFFSET nowa    ; wskaźnik na tekst
push hout           ; deskryptor buforu konsoli
callWriteConsoleA   ; wywołanie funkcji WriteConsoleA
;--- zaproszenie A ---------
push OFFSET zaproszenieA
push OFFSET zaproszenieA
call CharToOemA             ; konwersja polskich znaków
;--- wyświetlenie zaproszenia A ---
push 0                    ; rezerwa, musi być zero
push OFFSET rout            ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmiarA               ; ilość znaków
push OFFSET zaproszenieA    ; wskaźnik na tekst
push hout                 ; deskryptor buforu konsoli
call WriteConsoleA          ; wywołanie funkcji WriteConsoleA
;--- 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
lea EBX,bufor
mov EDI,rinp
mov BYTE PTR [EBX+EDI-1],0  ;zero na końcu tekstu
;--- przekształcenie A
push OFFSET bufor
call ScanInt
add ESP, 8
mov zmA, EAX
;;;..... B
;--- zaproszenie B---------
push OFFSET zaproszenieB
push OFFSET zaproszenieB
call CharToOemA             ; konwersja polskich znaków
;--- wyświetlenie zaproszenia B ---
push 0                    ; rezerwa, musi być zero
push OFFSET rout            ; wskaźnik na faktycznš ilość wyprowadzonych znaków
push rozmiarB               ; ilość znaków
push OFFSET zaproszenieB    ; wskaźnik na tekst
push hout                 ; deskryptor buforu konsoli
call WriteConsoleA          ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
push 0                    ; rezerwa, musi być zero
push OFFSET rinp            ; wskaźnik na faktycznš ilość wprowadzonych znaków
push rbuf                 ; rozmiar bufora
```

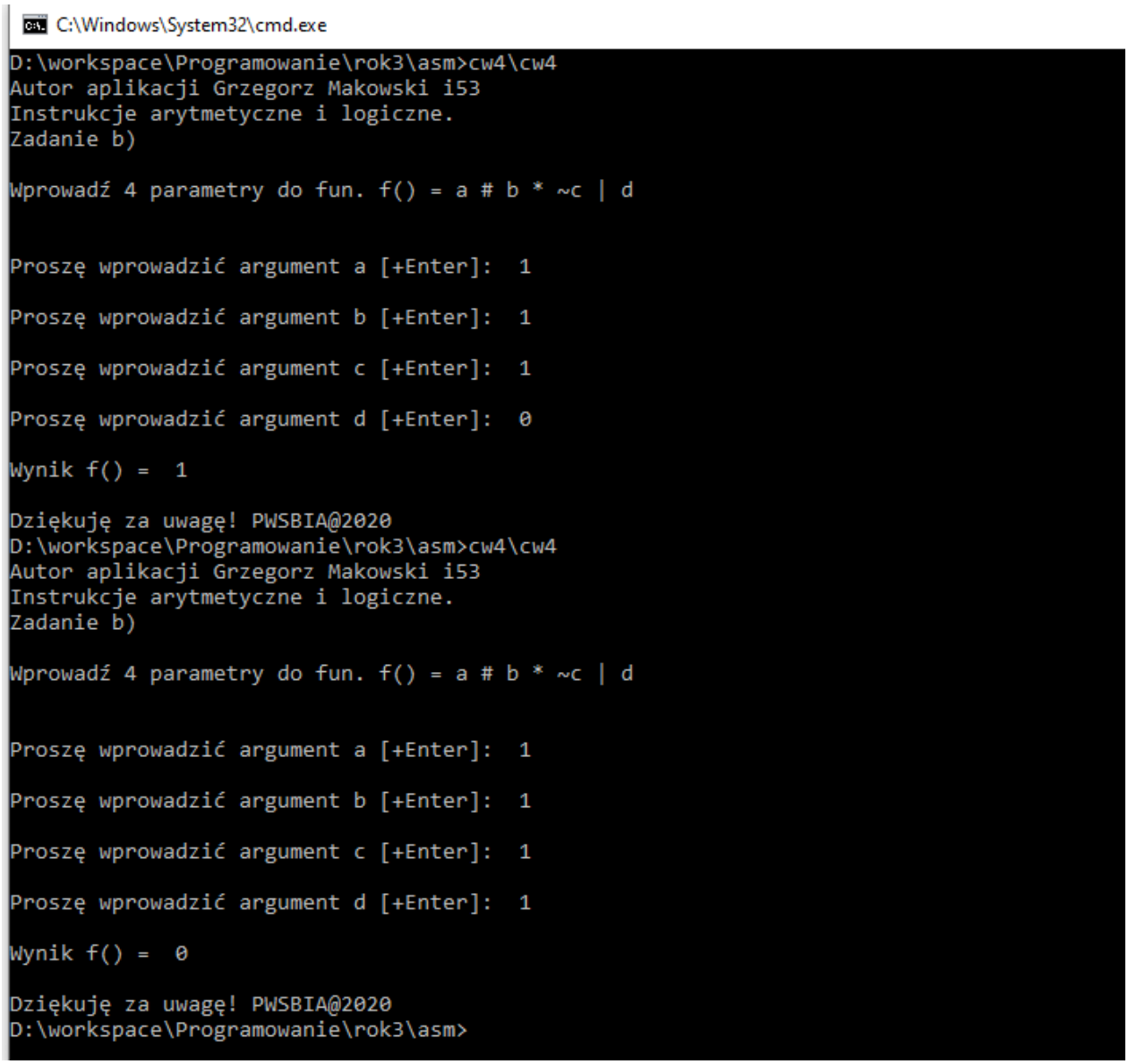
```
push OFFSET bufor          ;wskaźnik na bufor
push hinp                  ; deskryptor buforu konsoli
call ReadConsoleA          ; wywołanie funkcji ReadConsoleA
lea EBX,bufor
mov EDI,rinp
mov BYTE PTR [EBX+EDI-1],0 ;zero na końcu tekstu
;--- przekształcenie B
push OFFSET bufor
call ScanInt
add ESP, 8
mov zmB, EAX
;--- zaproszenie C ---------
push OFFSET zaproszenieC
push OFFSET zaproszenieC
call CharToOemA            ; konwersja polskich znaków
;--- wyświetlenie zaproszenia C ---
push 0                     ; rezerwa, musi być zero
push OFFSET rout           ; wskaźnik na faktycznš ilosć wyprowadzonych znaków
push rozmiarC              ; ilość znaków
push OFFSET zaproszenieC   ; wskaźnik na tekst
push hout                  ; deskryptor buforu konsoli
call WriteConsoleA         ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
push 0                     ; rezerwa, musi być zero
push OFFSET rinp           ; wskaźnik na faktycznš ilość wprowadzonych znaków
push rbuf                  ; rozmiar bufora
push OFFSET bufor          ; wskaźnik na bufor
push hinp                  ; deskryptor buforu konsoli
call ReadConsoleA          ; wywołanie funkcji ReadConsoleA
lea EBX,bufor
mov EDI,rinp
mov BYTE PTR [EBX+EDI-1],0  ;zero na końcu tekstu
;--- przekształcenie C
push OFFSET bufor
call ScanInt
add ESP, 8
mov zmC, EAX
;--- zaproszenie D ---------
push OFFSET zaproszenieD
push OFFSET zaproszenieD
call CharToOemA            ; konwersja polskich znaków
;--- wyświetlenie zaproszenia D ---
push 0                     ; rezerwa, musi być zero
push OFFSET rout           ; wskaźnik na faktycznš ilość wyprowadzonych znaków
push rozmiarD              ; ilość znaków
push OFFSET zaproszenieD   ; wskaźnik na tekst
push hout                  ; deskryptor buforu konsoli
call WriteConsoleA         ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
push 0                     ; rezerwa, musi być zero
push OFFSET rinp           ; wskaźnik na faktycznš ilość wprowadzonych znaków
push rbuf                  ; rozmiar bufora
push OFFSET bufor          ;wskaźnik na bufor
push hinp                  ; deskryptor buforu konsoli
call ReadConsoleA          ; wywołanie funkcji ReadConsoleA
lea EBX,bufor
mov EDI,rinp
mov BYTE PTR [EBX+EDI-1],0  ;zero na końcu tekstu
;--- przekształcenie D
push OFFSET bufor
call ScanInt
add ESP, 8
mov zmD, EAX
;--- obliczenia a # b * ~c | d
;--- kolejność ~ * | #
mov EAX, 0    ; zerowanie eax
mov EAX, zmC  ; c do eax
not EAX     ; negacja bitowa eax (c)
add EAX, 2    ; korekta wyniku negacji
and EAX, zmB  ; mnożenie logiczne b * ~c
or EAX, zmD   ; wynik mnożenia logicznego or (|) d
xor EAX, zmA

;;;; .................
;--- wyprowadzenie wyniku obliczeń ---
push EAX
```

```
push OFFSET wynik
push OFFSET bufor
call wsprintfA              ; zwraca ilość znaków w buforze
add ESP, 16                 ; czyszczenie stosu
mov rinp, EAX               ; zapamiętywanie ilości znaków
;--- wyświetlenie wynika ---------
push 0                      ; rezerwa, musi być zero
push OFFSET rout            ; wskaźnik na faktycznš ilość wyprowadzonych znaków
push rinp                   ; ilość znaków
push OFFSET bufor           ; wskażnik na tekst w buforze
push hout                   ; deskryptor buforu konsoli
call WriteConsoleA          ; wywołanie funkcji WriteConsoleA
;--- wyświetlenie nową linii ---
push 0        ; rezerwa, musi być zero
push OFFSET rout  ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2        ; ilość znaków
push OFFSET nowa ; wskażnik na tekst
push hout        ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- wyświetlenie nową linii ---
push 0        ; rezerwa, musi być zero
push OFFSET rout  ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2        ; ilość znaków
push OFFSET nowa ; wskażnik na tekst
push hout        ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- wyświetlenie zakonczenia ---
push 0                 ; rezerwa, musi być zero
push OFFSET rout       ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmZ
push OFFSET tekstZakoncz
push OFFSET tekstZakoncz
call CharToOemA
push 0                        ; rezerwa, musi być zero
push OFFSET rout              ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmZ                    ; ilość znaków
push OFFSET tekstZakoncz      ; wskażnik na tekst
push hout                     ; deskryptor buforu konsoli
call WriteConsoleA            ; wywołanie funkcji WriteConsole
;--- zakończenie procesu ---------
push 0
callExitProcess ; wywołanie funkcji ExitProcess
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ScanInt PROC
;; funkcja ScanInt przekształca ciąg cyfr do liczby, które jest zwracana przez EAX
;; argument - zakończony zerem wiersz z cyframi
;; rejestry: EBX - adres wiersza, EDX - znak liczby, ESI - indeks cyfry w wierszu, EDI - tymczasowy
;--- poczštek funkcji
push EBP
mov EBP, ESP       ; wskzźnik stosu ESP przypisujemy do EBP
;--- odkładanie na stos
push EBX
push ECX
push EDX
push ESI
push EDI
;--- przygotowywanie cyklu
mov EBX, [EBP+8]
push EBX
call lstrlenA
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, [EBP+8]   ; adres tekstu
;--- 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 EDX, 1
jmp nast
@@:
cmp BYTE PTR [EBX+ESI], 030h ;porównanie z kodem 0
jae @F
jmp nast
@@:
cmp BYTE PTR [EBX+ESI], 039h ;porównanie z kodem 9
jbe @F
jmp nast
;----
@@:
push EDX          ; do EDX procesor może zapisać wynik mnożenia
mov EDI, 10
mul EDI           ; mnożenie EAX * EDI
mov EDI, EAX      ; tymczasowo z EAX do EDI
xor EAX, EAX      ; zerowani EAX
mov AL, BYTE PTR [EBX+ESI]
sub AL, 030h      ; korekta: cyfra = kod znaku - kod 0
add EAX, EDI      ; dodanie cyfry
pop EDX
nast:
inc ESI
loop pocz
;--- wynik
or EDX, EDX ;analiza znacznika EDX
jz @F
neg EAX
@@:
et4:
;--- zdejmowanie ze stosu
pop EDI
pop ESI
pop EDX
pop ECX
pop EBX
;--- powrót
mov ESP, EBP      ; przywracamy wskaźnik stosu ESP
pop EBP
ret
ScanInt ENDP
_TEXT ENDS
END start
```

Testowanie


C:\Windows\System32\cmd.exe
D:\workspace\Programowanie\rok3\asm>cw4\cw4
Autor aplikacji Grzegorz Makowski i53
Instrukcje arytmetyczne i logiczne.
Zadanie b)
Wprowadź 4 parametry do fun. f() = a # b * ~c | d
Proszę wprowadzić argument a [+Enter]: 1
Proszę wprowadzić argument b [+Enter]: 1
Proszę wprowadzić argument c [+Enter]: 1
Proszę wprowadzić argument d [+Enter]: 0
Wynik f() = 1
Dziękuję za uwagę! PWSBIA@2020
D:\workspace\Programowanie\rok3\asm>cw4\cw4
Autor aplikacji Grzegorz Makowski i53
Instrukcje arytmetyczne i logiczne.
Zadanie b)
Wprowadź 4 parametry do fun. f() = a # b * ~c | d
Proszę wprowadzić argument a [+Enter]: 1
Proszę wprowadzić argument b [+Enter]: 1
Proszę wprowadzić argument c [+Enter]: 1
Proszę wprowadzić argument d [+Enter]: 1
Wynik f() = 0
Dziękuję za uwagę! PWSBIA@2020
D:\workspace\Programowanie\rok3\asm>

Zadanie c

Przesuwanie i rotacja bitów

Napisać program, w którym wykonać operacje przesuwania lub rotacji bitów według swojego zadania (w prawo CFc4; w lewo 2).
Liczbę testową 10100110001110000111100000111110b zapisać w sekcji danych korzystając z formatu liczb binarnych.
Wyświetlić liczbę testową oraz liczby po każdemu przesuwaniu.

Opracowanie zadania

```
;Przesuwanie i rotacja bitów
.586P
.MODEL flat, STDCALL
option casemap :none
;--- stale z pliku .\include\windows.inc ---
STD_INPUT_HANDLE equ -10
STD_OUTPUT_HANDLE equ -11
;--- funkcje API Win32 z pliku .\include\user32.inc ---
CharToOemA PROTO :DWORD,:DWORD
;--- funkcje API Win32 z pliku .\include\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
;--- podprogramy ----
ScanInt PROTO C adres:DWORD
ScanBin PROTO STDCALL adres:DWORD
DrukBin PROTO STDCALL liczba:DWORD
DrukShortBin PROTO STDCALL liczba:DWORD
;--- funkcje
;-------------
includelib .\lib\user32.lib
includelib .\lib\kernel32.lib
;-------------
_DATA SEGMENT
hout DD ?
hinp DD ?
naglow DB "Autor aplikacji Grzegorz Makowski.",0Dh,0Ah,0
rozmN DD $ - naglow
newline DB 0Dh,0Ah,0
ALIGN 4
naglrot DB 0Dh,0Ah,"Liczba binarna przed rotacja: ",0
poarot DB 0Dh,0Ah,"Cyklicznie przez znacznik CF 4 w prawo razy: ",0
ponrot DB 0Dh,0Ah,"W lewo 2 razy: ",0
align 4
rout DD 0 ;faktyczna ilosc wyprowadzonych znaków
rinp DD 0 ;faktyczna ilosc wprowadzonych znaków
rinp2 DD 0 ;faktyczna ilosc wprowadzonych znaków
bufor DB 128 dup(?)
rbuf DD 128
zmY DD 0
st0 DD 10100110001110000111100000111110b

tekstZakoncz DB "Dziękuję za uwagę! PWSBiA@2020",0                ; nagłówek
rozmZ DD $ - tekstZakoncz
_DATA ENDS
;------------
_DATA? SEGMENT
rbin dd ? ;ilosc znaków liczby binarnej
rrot dd ? ;ilosc znaków poszczególnych naglównków przy rotacji
_DATA? ENDS
_TEXT SEGMENT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
start:
```

```
;--- wywolanie funkcji GetStdHandle
push STD_OUTPUT_HANDLE
call GetStdHandle ; wywolanie funkcji GetStdHandle
mov hout, EAX ; deskryptor wyjsciowego bufora konsoli
push STD_INPUT_HANDLE
call GetStdHandle ; wywolanie funkcji GetStdHandle
mov hinp, EAX ; deskryptor wejsciowego bufora konsoli
;--- naglówek ---------
push OFFSET naglow
push OFFSET naglow
call CharToOemA ; konwersja polskich znaków
;--- wyswietlenie ---------
push 0 ; rezerwa, musi byc zero
push OFFSET rout ; wskaznik na faktyczna ilosc wyprowadzonych znaków
push rozmN ; ilosc znaków
push OFFSET naglow ; wskaznik na tekst
push hout ; deskryptor buforu konsoli
call WriteConsoleA ; wywolanie funkcji WriteConsoleA
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
invoke CharToOemA, offset naglrot, offset bufor ; konwersja polskich znaków
; liczymy dlugosc stringu do wyswietlenia
invoke lstrlenA, offset bufor
mov rrot, eax
; wyswietlamy powitanie
invoke WriteConsoleA, hout, offset bufor, rrot, offset rout, 0
; i wyswietlamy nasz ciag binarny
invoke DrukBin, st0
invoke CharToOemA, offset poarot, offset bufor ; konwersja polskich znaków
; liczymy dlugosc stringu do wyswietlenia
invoke lstrlenA, offset bufor
mov rrot, eax
; wyswietlamy powiadomienia o rotacji
invoke WriteConsoleA, hout, offset bufor, rrot, offset rout, 0
mov eax, st0 ; kopiujemy nasz ciag bitów do akumulatora
shr eax, 4 ; przesuwamy go w prawo o 4 pozycje
mov st0, eax ; i kopiujemy spowrotem do zmiennej st0 i wyswietlamy ciag binarny
invoke DrukBin, st0
invoke CharToOemA, offset ponrot, offset bufor ; konwersja polskich znaków
; liczymy dlugosc stringu do wyswietlenia
invoke lstrlenA, offset bufor
mov rrot, eax
; wyswietlamy powiadomienia o rotacji
invoke WriteConsoleA, hout, offset bufor, rrot, offset rout, 0
mov eax, st0 ; kopiujemy nasz ciag bitów do akumulatora
rol eax, 2 ; przesuwamy go w lewo o 2 pozycje
mov st0, eax ; i kopiujemy spowrotem do zmiennej st0
; i wyswietlamy nasz ciag binarny
invoke DrukBin, st0
;--- wyświetlenie nową linii ---
push 0        ; rezerwa, musi być zero
push OFFSET rout  ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push 2        ; ilość znaków
push OFFSET newline  ; wskaźnik na tekst
push hout      ; deskryptor buforu konsoli
callWriteConsoleA ; wywołanie funkcji WriteConsoleA
;--- wyświetlenie zakonczenia ---
push 0                ; rezerwa, musi być zero
push OFFSET rout       ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmZ
push OFFSET tekstZakoncz
push OFFSET tekstZakoncz
call CharToOemA
push 0                     ; rezerwa, musi być zero
push OFFSET rout               ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmZ                  ; ilość znaków
push OFFSET tekstZakoncz       ; wskażnik na tekst
push hout                   ; deskryptor buforu konsoli
call WriteConsoleA             ; wywołanie funkcji WriteConsole
;--- zakonczenie procesu ---------
push 0
call ExitProcess ; wywolanie funkcji ExitProcess
ScanInt PROC C adres
;; funkcja ScanInt przeksztalca ciag cyfr do liczby, która jest zwracana przez EAX
;; argument - zakonczony zerem wiersz z cyframi
;; rejestry: EBX - adres wiersza, EDX - znak liczby, ESI - indeks cyfry w wierszu, EDI - tymcza-sowy
;--- poczatek funkcji
```

```
;--- odkladanie na stos
push EBX
push ECX
push EDX
push ESI
push EDI
;--- przygotowywanie cyklu
INVOKE lstrlenA, adres
mov EDI, EAX ;ilosc znaków
mov ECX, EAX ;ilosc powtórzen = ilosc znaków
xor ESI, ESI ; wyzerowanie ESI
xor EDX, EDX ; wyzerowanie EDX
xor EAX, EAX ; wyzerowanie EAX
mov EBX, adres
;--- cykl --------------------------
pocz: cmp BYTE PTR [EBX+ESI], 02Dh ;porównanie z kodem '-'
jne @F
mov EDX, 1
jmp nast
@@: cmp BYTE PTR [EBX+ESI], 030h ;porównanie z kodem '0'
jae @F
jmp nast
@@: cmp BYTE PTR [EBX+ESI], 039h ;porównanie z kodem '9'
jbe @F
jmp nast
;----
@@: push EDX ; do EDX procesor moze zapisac wynik mnozenia
mov EDI, 10
mul EDI ;mnozenie EAX * EDI
mov EDI, EAX ; tymczasowo z EAX do EDI
xor EAX, EAX ;zerowani EAX
mov AL, BYTE PTR [EBX+ESI]
sub AL, 030h ; korekta: cyfra = kod znaku - kod '0'
add EAX, EDI ; dodanie cyfry
pop EDX
nast: inc ESI
dec ECX
jz @F
jmp pocz
;--- wynik
@@: or EDX, EDX ;analiza znacznika EDX
jz @F
neg EAX
@@:
;--- zdejmowanie ze stosu
pop EDI
pop ESI
pop EDX
pop ECX
pop EBX
;--- powrót
ret
ScanInt ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ScanBin PROC STDCALL adres:DWORD
;; funkcja ScanBin przeksztalca ciag cyfr do liczby, która jest zwracana przez EAX
;; argument - zakonczony zerem wiersz z cyframi binarnymi '0' badz '1'
;; rejestry: EBX - adres wiersza, ESI - indeks cyfry w wierszu, EDI - tymczasowy
;--- poczatek funkcji
;--- odkladanie na stos
push ebx
push ecx
push edx ; podczas mnozenia moze sie zdazyc ze EDX zostanie zmodyfikowane
push esi
push edi
;--- przygotowanie cyklu
invoke lstrlenA, adres
mov ecx, eax ; ilosc powtórzen = ilosc znaków
mov ebx, adres ; do rejetru EBX przenosimy adres ciagu cyfr
xor esi, esi ; zerujemy numer kolejnych cyfr w tablicy
xor edi, edi ; zerujemy rejestr tymczasowy
xor eax, eax ; zerujemy akumulator
;--- cykl --------------------
pocz:
cmp byte ptr[ebx + esi], '0' ; porównujemy znak z bufora z znakiem '0'
jae @F ; jesli jest wiekszy badz równy to dobrze, przechodzimy do kolejnej etykiety.
```

```
jmp nast ; jesli jest mniejszy to oznacza iz jest to niepoprawny znak,
; przechodzimy do nastepnego znaku.
@@:
cmp byte ptr[ebx + esi], '1' ; porównujemy znak z bufora z kodem znaku '1'
jbe @F ; jesli jest mniejszy badz równy to dobrze i przechodzimy do kolejnej etykiety
jmp nast ; jesli jest wiekszy to jest to niepoprawny znak i przechodzimy do nastep-nego zna-ku.
@@:
mov edi, 2 ; do rejestru edi wstawiamy 2
mul edi ; przy kazdym przejsciu petli mnozymy eax przez 2 aby przesunac poprzedni wynik w lewo
mov edi, eax ; czasowo nasza cyfre przenosimy do rejestru edi
xor eax, eax ; i zerujemy akumulator
mov al, byte ptr[ebx + esi] ; do akumulatora przenosimy bajt naszego znaku '0' badz '1'
sub al, '0' ; odejmujemy kod znaku '0' aby uzyskac cyfre 1 badz 0
add eax, edi ; i dodajemy to do zapisanej liczby
nast: ; przechodzenie do nastepnej pozycji polega na:
inc esi ; zwiekszeniu indeksu cyfry aby przejsc na kolejny znak
dec ecx ; zmniejszeniu kiczby znaków do przejscia o 1
jz @F ; jesli ilosc znaków do przejscia wynosi 0 to przechodzimy do kolejnej etykiety
jmp pocz ; jesli jeszcze mamy znaki do przejscia to zaczynamy wykonywac od po-czatku wszystko,
; tylko tym razem na kolejnym znaku
@@:
;--- zdejmowanie ze stosu
pop edi
pop esi
pop edx
pop ecx
pop ebx
;--- powrót
ret 4 ; poniewaz funkcja uzywa konwencji STDCALL wiec sprzatamy po sobie na stosie
; odejmujac od esp 4
ScanBin ENDP
;.....
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DrukBin PROC STDCALL liczba:DWORD
;; funkcja DrukBin wyswietla liczbe-argument w postaci binarnej
;; rejestry: ECX - cykl, EDI - maska, ESI - indeks w buforze, EBX - przesuniecie bufo-ra
;--- odkladanie na stos
push ECX
push EDI
push ESI
push EBX
;---
mov ECX,32
mov EDI,80000000h
mov ESI,0
mov EBX,OFFSET bufor
@@d1:
mov BYTE PTR [EBX+ESI],'0'
test liczba,EDI
jz @@d2
inc BYTE PTR [EBX+ESI]
@@d2:
shr EDI,1
inc ESI
loopnz @@d1
mov BYTE PTR [EBX+32],0Dh
mov BYTE PTR [EBX+33],0Ah
;--- wyswietlenie wynika ---------
push 0 ; rezerwa, musi byc zero
push OFFSET rout ; wskaznik na faktyczna ilosc wyprowadzonych znaków
push 34 ; ilosc znaków
push OFFSET bufor ; wskaznik na tekst w buforze
push hout ; deskryptor buforu konsoli
call WriteConsoleA ; wywolanie funkcji WriteConsoleA
;--- zdejmowanie ze stosu
pop EBX
pop ESI
pop EDI
pop ECX
;--- powrót
ret 4
DrukBin ENDP
;.....
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DrukShortBin PROC STDCALL liczba:DWORD
;; funkcja DrukShortBin wysfietla liczbe-argument w postaci binarnej,
;; pomijamy nieznaczace zera na poczatku
;; rejestr ecx - liczba pozycji znaczacych, rejestr ebx - wskaznik na bufor,
```

```
;; esi - indeks w buforze, eax - liczba, edx - reszta z dzielenia(czyli nasze '1' i '0')
;; edi - liczbe do dzielenia
;; funkcja wykorzystuje najprostrzy algorytm zamiany liczby na postac dwujkowa
;; czyli dzielimy liczbe przez 2 i reszte zapisujemy na stosie
;; i tak do momentu kiedy liczba bedzie równa 0, wtedy odczytujemy liczbe binarna
;; w odwrotnej koljnosci do zapisywania na stosie
;--- odkladanie na stos
push ebx
push ecx
push edx
push esi
push edi
;--- przygotowywanie rejestrów ---
mov ebx, offset bufor ; do ebx kopiujemy adres bufora,
;do którego bedziemy wstawiac kolejne cyfry liczby binarnej
xor esi, esi ; zerujemy indeks liter w buforze
mov eax, liczba ; do akumulatora kopiujemy nasza liczbe
xor edx, edx ; zerujemy edx
xor ecx, ecx ; zerujemy licznik znaków
mov edi, 2 ; do edi wstawiamy 2
;--- cykl dzielenia i wkladania na stos ---
pocz:
xor edx, edx ; zerujemy edx poniewaz instrukcja div uzywa rejestrów EDX:EAX
div edi ; dzielimy nasza liczbe przez 2
add edx, '0' ; dodajemy do niej kod znaku zero aby zamienic ja na znak
push edx ; i wstawiamy ja na stos
inc ecx ; zwiekszamy licznik cyfr na stosie
or eax, eax ; wykonyjeme instrukcje or aby ustawic flage ZF jesli eax równa sie zero
jz @F ; jesli eax równa sie zero to przechodzimy do kolejne etykiety
jmp pocz ; a jesli jest jeszcze co dzielic to skaczemy na poczatek
;--- tworzenie stringu(napisu)
; polega na pobraniu ze stosu w odwrotnej kolejnosci niz wstawialismy naszyc zna-ków
@@:
pop eax ; pobieramy nasz znak i umieszczamy go w akumulatorze
mov byte ptr [ebx + esi], al ; nastepnie bit znaku umieszczamy w buforze
inc esi ; przesówamy pozycje do wstawiania na kolejna w buforze
dec ecx ; i zmniejszamu ilosc liczb do wstawienia
jz @F ; jesli nie ma juz wiecej cyfr to przechodzimy dalen
jmp @B ; lecz jesli sa to przechodzimy do pobierania kolejnego znaku ze stosu
;--- nak koniec dodajemy 0Dh, 0Ah oraz 0 ---
@@:
mov byte ptr[ebx + esi], 0Dh ; dodajemy jeszcze na koniec znak noweh lini
mov byte ptr[ebx + esi + 1], 0Ah ; oraz znak powrotu karetki
mov byte ptr[ebx + esi + 2], 0 ; no i na koniec zakanczamy nasz ciag zerem
;--- wyswietlamy
invoke lstrlenA, offset bufor ; liczymy dlugosc naszego napisu
mov rbin, eax ; kopiujemy go do rbin
invoke WriteConsoleA, hout, offset bufor, rbin, offset rout, 0
; i wyswietlamy go
;--- zdejmowanie ze stosu
pop edi
pop esi
pop edx
pop ecx
pop ebx
ret 4
DrukShortBin ENDP
_TEXT ENDS
END start
```

Testowanie

```
D:\workspace\Programowanie\rok3\asm>komp cw4
cw4\cw4
Microsoft (R) Macro Assembler Version 6.14.8444
Copyright (C) Microsoft Corp 1981-1997.  All rights reserved.

 Assembling: .\cw4\cw4.asm

D:\workspace\Programowanie\rok3\asm>kons cw4

D:\workspace\Programowanie\rok3\asm>.\BIN\link /SUBSYSTEM:CONSOLE /LIBPATH:.\LIB /OUT:.\cw4\cw4
:.\cw4\cw4.pdb .\cw4\cw4.obj
Microsoft (R) Incremental Linker Version 5.12.8078
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.


D:\workspace\Programowanie\rok3\asm>cw4\cw4
Autor aplikacji Grzegorz Makowski.

Liczba binarna przed rotacja: 10100110001110000111100000111110

Cyklicznie przez znacznik CF 4 w prawo razy: 00001010011000111000011110000011

W lewo 2 razy: 00101001100011100001111000001100

Dziękuję za uwagę! PWSBiA@2020
D:\workspace\Programowanie\rok3\asm>
```