PRYWATNA WYŻSZA SZKOŁA NAUK SPOŁECZNYCH, KOMPUTEROWYCH I MEDYCZNYCH

WYDZIAŁ NAUK SPOŁECZNYCH I TECHNIK KOMPUTEROWYCH

Ćwiczenie

z programowania niskopoziomowego

„Podprogramy i makrodefinicje.”

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.

Makrodefinicje.

Program opracowany w poprzednim ćwiczeniu przekształcić na program z nie mniej niż trzema makrodefinicjami.

Opracowanie zadania

Wskazówki

Programy opracowane w poprzednich ćwiczeniach zawierają fragmenty związane z otrzymaniem deskryptorów wejściowego i wyjściowego buforów konsoli za pomocą funkcji GetStdHandle, na przykład:

- fragment 1:

```
push STD_OUTPUT_HANDLE
call GetStdHandle
mov hout,EAX ; deskryptor wyjściowego bufora konsoli
```

- fragment 2:

```
push STD_INPUT_HANDLE
call GetStdHandle
mov hinp, EAX ; deskryptor wejściowego bufora konsoli
```

Analizując wystąpienia takich fragmentów, można wydzielić parametry i stworzyć makro, na przykład z nazwą PODAJDESKR:

```
PODAJDESKR MACRO handle, deskrypt
push handle
call GetStdHandle
mov deskrypt,EAX ;; deskryptor bufora konsoli
ENDM
```

W wywołaniach tego makra podajemy parametry faktyczne:

```
PODAJDESKR STD_OUTPUT_HANDLE, hout
```

oraz

```
PODAJDESKR STD_INPUT_HANDLE, hinp
```

W programach opracowanych w poprzednich ćwiczeniach istnieje jeszcze jeden fragment, który jest wygodnie przerobić na makro, - fragment związany z wyświetleniem zawartości tablicy tekstowej, na przykład:

```
;--- wyświetlenie wyniku ---------
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
```

W programie - kandydacie na wprowadzenie makra należy przeanalizować wystąpienia podobnych fragmentów, wydzielić parametry i stworzyć makro, na przykład z nazwą WYSWIETLENIE.

```
;Ćwiczenie 5, Podprogramy i makrodefinicje
.586P
.MODEL flat, stdcall
;-------------
includelib .\lib\user32.lib
includelib .\lib\kernel32.lib
;--- stale z pliku .\include\windows.inc ---
STD_INPUT_HANDLE    equ -10
STD_OUTPUT_HANDLE   equ -11
```

```
;--- stale ---
mbuf = 128
SYS_exit        equ 0
;--- makra ---
podajdeskr macro handle, deskrypt
 push handle
 call GetStdHandle
 mov deskrypt,eax ; deskryptor bufora konsoli
endm

plznaki macro text, bufor
 invoke CharToOemA, addr text, addr bufor
endm

wyswietl macro bufor, rozmiar
 ;--- wyświetlenie wyniku ---------
 push 0 ; rezerwa, musi być zero
 push offset rout ;wskaźnik na faktyczną ilość wyprowadzonych znaków
 push rozmiar ; ilość znaków
 push offset bufor ; wskażnik na tekst w buforze
 push hout ; deskryptor buforu konsoli
 call WriteConsoleA ; wywołanie funkcji WriteConsoleA
endm

nowalinia macro nowa
;--- 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
endm

zmienna macro deskkons, bufor, rozmb, frozm, zmienna
 invoke ReadConsoleA, deskkons, addr bufor, rozmb, addr frozm, 0
 push offset bufor
 call ScanInt
 add esp, 4
 mov zmienna, eax
endm

;--- 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  hinp:dword,adres_bufor:dword,rbuf:dword,adres_rinp:dword,rezerwa:dword
WriteConsoleA proto hout:dword,adres_bufor:dword,rozm:dword,adres_out:dword,rezerwa:dword
ExitProcess proto :dword
wsprintfA proto c :vararg
lstrlenA proto :dword

;--- funkcje
ScanInt proto c :dword
DrukBin proto stdcall :dword
; --- deklaracje podprogramów
arytm proto c
logika proto stdcall :dword, :dword, :dword, :dword
przesuwanie proto stdcall :dword

;-------------
_data segment
 hout DD?
 hinpDD?
 nl DB 0Dh, 0Ah, 0; nowa linia          ;nowa linia
 align 4 ; przesuniecie do adresu podzielnego na 4
 naglow DB "Autor aplikacji Grzegorz Makowski i53",0              ; nagłówek
 align 4              ; przesuniecie do adresu podzielnego na 4
 rozmN DD $ - naglow                                    ; ilosc znakow
 align 4              ; przesuniecie do adresu podzielnego na 4
 temat DB 0Dh,0Ah,"Podprogramy i makrodefinicje.",0
 align 4 ; przesuniecie do adresu podzielnego na 4
 rozmT DD $ - temat
 align 4 ; przesuniecie do adresu podzielnego na 4
 naglowA DB 0Dh,0Ah, "Wprowadź 4 parametry fun. f() = A/B-C+D",0  ; nagłówek
```

```
 align 4 ; przesuniecie do adresu podzielnego na 4
 rozmnaglA DD $ - naglowA
 align 4
 naglowB DB 0Dh,0Ah, "Wprowadź 4 parametry fun. f() = A#B*~C|D",0  ; nagłówek
 align 4 ; przesuniecie do adresu podzielnego na 4
 rozmnaglB DD $ - naglowB
 align 4 ; przesuniecie do adresu podzielnego na 4
 zaprA DB 0Dh,0Ah,"Proszę wprowadzić argument a [+Enter]: ",0
 align 4
 rozmA DD $ - zaprA ;ilość znaków w tablicy
 zmA DD 1 ; argument a
 zaprB DB 0Dh,0Ah,"Proszę wprowadzić argument b [+Enter]: ",0
 align 4
 rozmB DD $ - zaprB ;ilość znaków w tablicy
 zmB DD 2 ; argument b
 zaprC DB 0Dh,0Ah,"Proszę wprowadzić argument c [+Enter]: ",0
 align 4
 rozmC DD $ - zaprC ;ilość znaków w tablicy
 zmC DD 3 ; argument c
 align 4
 zaprD DB 0Dh,0Ah,"Proszę wprowadzić argument d [+Enter]: ",0
 align 4
 rozmD DD $ - zaprD ;ilość znaków w tablicy
 align 4
 zmD DD 4 ; argument d
 align 4
 zadA DB 0Ah,"Zadanie a) ",0              ; nagłówek zadania A
 align 4 ; przesuniecie do adresu podzielnego na 4
 rozmzadA DD $ - zadA
 align 4 ; przesuniecie do adresu podzielnego na 4
 zadB DB 0Dh,0Ah,"Zadanie b) ",0              ; nagłówek zadania B
 align 4
 rozmzadB DD $ - zadB
 align 4
 zadC DB 0Dh,0Ah,"Zadanie c) ",0              ; nagłówek zadania C
 align 4
 rozmzadC DD $ - zadC
 align 4
 rot1 DB 0Dh,0Ah,"Liczba binarna: ",0
 align 4
 rozmrot1 DD $ - rot1
 rot2 DB 0Dh,0Ah,"Cykl.prawo CF4: ",0
 align 4
 rozmrot2 DD $ - rot2
 rot3 DB 0Dh,0Ah,"W lewo 2 razy : ",0
 align 4
 rozmrot3 DD $ - rot3
 wzorf      DB 0Dh,0Ah,"Funkcja f() = %4ld",0
 align 4
 rout DD 0 ;faktyczna ilość wyprowadzonych znaków
 rinp DD 0 ;faktyczna ilość wprowadzonych znaków
 rinp2 DD 0 ;faktyczna ilość wprowadzonych znaków
 bufor DB mbuf dup(?)
 bufor2 DB mbuf dup(?)
 rbuf DD mbuf
   wyn   DD  0 ; zienna do przechowywania wyniku
   st0   DD  10100110001110000111100000111110b
_data ends
;------------

_text segment
start:
;--- wywołanie funkcji GetStdHandle - MAKRO
 podajdeskr STD_OUTPUT_HANDLE, hout
 podajdeskr STD_INPUT_HANDLE, hinp
;--- nagłówek ---------
 plznaki naglow, bufor ; konwersja polskich znaków - MAKRO
 wyswietl bufor, rozmN
 plznaki temat, bufor ; konwersja polskich znaków - MAKRO
 wyswietl bufor, rozmT
;--- wyświetlenie nowej linni---------
 nowalinia nl
;--- zaproszenie A ---------
 plznaki zadA, bufor2
 wyswietl bufor2, rozmzadA
 ;--- new line ---------
```

```
 nowalinia nl
 plznaki naglowA, bufor ; konwersja polskich znaków - MAKRO
 wyswietl bufor, rozmnaglA
 plznaki zaprA, bufor
;--- wyświetlenie zaproszenia A ---
 wyswietl bufor, rozmA   ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmA
;--- zaproszenie B ---------
 plznaki zaprB, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia B ---
 wyswietl bufor, rozmB ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmB
;--- zaproszenie C ---------
 plznaki zaprC, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia C ---
 wyswietl bufor, rozmC ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmC
;--- zaproszenie D ---------
 plznaki zaprD, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia D ---
 wyswietl bufor, rozmD ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmD
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- obliczenia Funkcja y =a/b-c+d
     push zmD
     push zmC
     push zmB
     push zmA
     call arytm
     add  esp, 16
     mov  wyn, eax
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- wyprowadzenie wyniku obliczeń ---
 invoke wsprintfA,OFFSET bufor,OFFSET wzorf,wyn ; zwraca ilość znaków w buforze
 movrinp, eax; zapamiętywanie ilości znaków
 ;--- new line ---------
 nowalinia nl
 wyswietl bufor, rinp
;--- wyświetlenie nowej linni---------
 nowalinia nl
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- zaproszenie B ---------
 plznaki zadB, bufor2
 wyswietl bufor2, rozmzadB
 ;--- wyświetlenie nowej linni---------
 nowalinia nl
 plznaki naglowB, bufor ; konwersja polskich znaków - MAKRO
 wyswietl bufor, rozmnaglB ; wywołanie funkcji WriteConsoleA
 plznaki zaprA, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia A ---
 wyswietl bufor, rozmA ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmA
;--- zaproszenie B ---------
 plznaki zaprB, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia B ---
 wyswietl bufor, rozmB ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmB
;--- zaproszenie C ---------
 plznaki zaprC, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia C ---
 wyswietl bufor, rozmC ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmC
;--- zaproszenie D ---------
 plznaki zaprD, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia D ---
 wyswietl bufor, rozmD ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmD
 ;--- new line ---------
```

```
 nowalinia nl
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- obliczenia Funkcja y =a#b*~c|d
 invoke logika,zmA,zmB,zmC,zmD
 add   esp,16
   mov   wyn,eax
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- wyprowadzenie wyniku obliczeń ---
 invoke wsprintfA,OFFSET bufor,OFFSET wzorf,wyn ; zwraca ilość znaków w buforze
 movrinp2, eax ; zapamiętywanie ilości znaków
;--- wyświetlenie wyniku ---------
 wyswietl bufor, rinp2
;--- new line ---------
 nowalinia nl
;----------------
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- zaproszenie C ---------
 plznaki zadC, bufor2
 wyswietl bufor2, rozmzadC
;;;; w prawo CFc4, w lewo 2
;--- new line ---------
 nowalinia nl
 ;----------------------------
 invoke przesuwanie, st0
 add esp, 16
;--- zakończenie procesu ---------
 invoke ExitProcess, SYS_exit; wywołanie funkcji ExitProcess
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;---------Podprogramy
ScanInt proc c adres
;; funkcja ScanInt przekształca ciąg cyfr do liczby, którą jest zwracana przez eax
;; argument - zakończony zerem wiersz z cyframi
;; rejestry: ebx - adres wiersza, edx - znak liczby, esi - indeks cyfry w wierszu, edi - tymczasowy
;--- początek funkcji
;--- odkładanie na stos
 push ebx
 push ecx
 push edx
 push esi
 push edi
;--- przygotowywanie cyklu
 invoke lstrlenA, adres
 movedi, eax ;ilość znaków
 movecx, eax ;ilość powtórzeń = ilość znaków
 xor esi, esi ; wyzerowanie esi
 xor edx, edx ; wyzerowanie edx
 xor eax, eax ; wyzerowanie eax
 movebx, adres
;--- cykl
pocz: cmpBYTE PTR [ebx+esi], 02Dh ;porównanie z kodem '-'
 jne @F
 movedx, 1
 jmp nast
@@:cmpBYTE PTR [ebx+esi], 030h;porównanie z kodem '0'
 jae @F
 jmp nast
@@:cmpBYTE PTR [ebx+esi], 039h;porównanie z kodem '9'
 jbe @F
 jmp nast
;----
@@:push edx ; do edx procesor może zapisać wynik mnożenia
 movedi, 10
 mul edi  ;mnożenie eax * edi
 movedi, eax ; tymczasowo z eax do edi
 xor eax, eax;zerowani eax
 movAL, 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
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DrukBin proc stdcall liczba:dword
;; funkcja DrukBin wyswietla liczbę-argument w postaci binarnej
;; rejestry: ecx - cykl, edi - maska, esi - indeks w buforze, ebx - przesunięcie bufora
;--- odkładanie na stos
   push    ecx
   push    edi
   push    esi
   push    ebx
;---
   mov     ecx,32
   mov     edi,80000000h
   mov     esi,0
   mov     ebx,OFFSET bufor
et1:
   mov     BYTE PTR [ebx+esi],'0'
   test    liczba,edi
   jz      @F
   inc     BYTE PTR [ebx+esi]
@@:
   shr     edi,1
   inc     esi
   loopnz  et1
   mov     BYTE PTR [ebx+32],0Dh
   mov     BYTE PTR [ebx+33],0Ah
;--- wyświetlenie wyniku ---------
   invoke WriteConsoleA,hout,OFFSET bufor,34,OFFSET rout,0
;--- zdejmowanie ze stosu
   pop     ebx
   pop     esi
   pop     edi
   pop     ecx
;--- powrót
   ret     8
DrukBin ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
option prologue: none
option epilogue: none
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
arytm proc c
;--- obliczenia A / B - C + D
 push ebp
 mov ebp, esp
 mov edx, 0      ; zerowanie edx
 mov eax, 0            ; zerowanie eax
 mov eax, dword ptr [ebp+8]   ; zm A do eax
 div dword ptr [ebp+12]  ; dzielenie  A / B wynik w eax
 mov edx, dword ptr [ebp+16] ; zmienna C do edx
 sub eax, edx     ; odejumjemy od wyniku dzielenia C, wynik w eax
 add eax, dword ptr [ebp+20] ; dodajemy do eax zmienną D, wynik w eax
 mov edx, 0            ; sprzątanie, zerowanie edx
 mov esp,ebp
 pop ebp
 ret 8 ;ominięcie ramki stosu z parametrami
arytm endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
option prologue: prologuedef
option epilogue: epiloguedef
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
logika proc stdcall argA:dword,argB:dword,argC:dword,argD:dword
;--- obliczenia a # b * ~c | d
;--- kolejność ~ * | #
 push ebp ;przechowywanie ebp na stosie
 mov ebp, esp ;zamiana ebp
 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
 mov esp, ebp ;zamiana esp
 pop ebp ;przewrócenie ebp ze stosu
 ret 8 ;ominięcie ramki stosu z parametrami
logika endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
przesuwanie proc stdcall arg:dword
 push ebp
 mov ebp, esp
 plznaki rot1, bufor
 wyswietl bufor, rozmrot1
 invoke DrukBin, st0
 mov eax, 0
 mov eax, st0
 rcr eax, 4
 mov st0, eax
 plznaki rot2, bufor
 wyswietl bufor, rozmrot2
 invoke DrukBin, st0
 mov eax, 0
 mov eax, st0
 rol eax, 2
 mov st0, eax
 plznaki rot3, bufor
 wyswietl bufor, rozmrot3
 invoke DrukBin, st0
 mov esp, ebp ;zamiana esp
 pop ebp ;przewrócenie ebp ze stosu
 ret 8 ;ominięcie ramki stosu z parametrami
przesuwanie endp
end start
_text ends
```

Testowanie

```
D:\workspace\Programowanie\rok3\asm>cw5\cw5
Autor aplikacji Grzegorz Makowski i53
Podprogramy i makrodefinicje.

Zadanie a)

Wprowadź 4 parametry fun. f() = A/B-C+D
Proszę wprowadzić argument a [+Enter]:    200

Proszę wprowadzić argument b [+Enter]:    2

Proszę wprowadzić argument c [+Enter]:    80

Proszę wprowadzić argument d [+Enter]:    1


Funkcja f() =   21

Zadanie b)

Wprowadź 4 parametry fun. f() = A#B*~C|D
Proszę wprowadzić argument a [+Enter]:    0

Proszę wprowadzić argument b [+Enter]:    1

Proszę wprowadzić argument c [+Enter]:    0

Proszę wprowadzić argument d [+Enter]:    1


Funkcja f() =    1

Zadanie c)

Liczba binarna:  11010011000111000011110000011110

Cykl.prawo CF4:  11100101001100011100001111000011

W lewo 2 razy :  00010100110001110000111100001111

D:\workspace\Programowanie\rok3\asm>
```

Zadanie b

Podprogramy

Program opracowany w punkcie "a" przekształcić na program z podprogramami „arytm”, „logika”, "przesuw". W podprogramie „arytm”, wywoływanej przez instrukcję „call”, operować argumentami przez wyrażenia typu „[EBP+8]” oraz operować lokalnymi zmiennymi przez wyrażenia typu „[EBP-4]”. Przed tym wariantem podprogramu zakazać generowanie prologu standardowego dwoma dyrektywami:

```
OPTION PROLOGUE: NONE
OPTION EPILOGUE: NONE
```

a po tekstu podprogramu zezwolić generowanie prologu standardowego dwoma dyrektywami:

```
OPTION PROLOGUE: PROLOGUEDEF
OPTION EPILOGUE: EPILOGUEDEF.
```

Zastosować dyrektywę INVOKE dla wywołania podprogramów „logika” i „przesuw”, dyrektywę `PROTO` dla opisu podprogramów „logika” i „przesuw” oraz dyrektywę `LOCAL` dla lokalnych zmiennych podprogramów.
Obliczoną wartość zwracać przez rejestr-akumulator EAX.
Przy tym stosować makrodefinicje z punktu "a".

Opracowanie zadania

```
;Ćwiczenie 5, Podprogramy i makrodefinicje
.586P
.MODEL flat, stdcall
;-------------
includelib .\lib\user32.lib
includelib .\lib\kernel32.lib
;--- stale z pliku .\include\windows.inc ---
STD_INPUT_HANDLE    equ -10
STD_OUTPUT_HANDLE   equ -11
;--- stale ---
mbuf = 128
SYS_exit        equ 0
;--- makra ---
podajdeskr macro handle, deskrypt
 push handle
 call GetStdHandle
 mov deskrypt,eax ; deskryptor bufora konsoli
endm

plznaki macro text, bufor
 invoke CharToOemA, addr text, addr bufor
endm

wyswietl macro bufor, rozmiar
 ;--- wyświetlenie wyniku ---------
 push 0 ; rezerwa, musi być zero
 push offset rout ;wskaźnik na faktyczną ilość wyprowadzonych znaków
 push rozmiar ; ilość znaków
 push offset bufor ; wskażnik na tekst w buforze
 push hout ; deskryptor buforu konsoli
 call WriteConsoleA ; wywołanie funkcji WriteConsoleA
endm

nowalinia macro nowa
;--- 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
endm
```

```
zmienna macro deskkons, bufor, rozmb, frozm, zmienna
 invoke ReadConsoleA, deskkons, addr bufor, rozmb, addr frozm, 0
 push offset bufor
 call ScanInt
 add esp, 4
 mov zmienna, eax
endm

;--- 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  hinp:dword,adres_bufor:dword,rbuf:dword,adres_rinp:dword,rezerwa:dword
WriteConsoleA proto hout:dword,adres_bufor:dword,rozm:dword,adres_out:dword,rezerwa:dword
ExitProcess proto :dword
wsprintfA proto c :vararg
lstrlenA proto :dword

;--- funkcje
ScanInt proto c :dword
DrukBin proto stdcall :dword
; --- deklaracje podprogramów
arytm proto c
logika proto stdcall :dword, :dword, :dword, :dword
przesuwanie proto stdcall :dword

;-------------
_data segment
 hout DD ?
 hinpDD ?
 nl DB 0Dh, 0Ah, 0 ; nowa linia           ;nowa linia
 align 4 ; przesuniecie do adresu podzielnego na 4
 naglow DB "Autor aplikacji Grzegorz Makowski i53",0                ; nagłówek
 align 4               ; przesuniecie do adresu podzielnego na 4
 rozmN DD $ - naglow                                        ; ilosc znakow
 align 4               ; przesuniecie do adresu podzielnego na 4
 temat DB 0Dh,0Ah,"Podprogramy i makrodefinicje.",0
 align 4 ; przesuniecie do adresu podzielnego na 4
 rozmT DD $ - temat
 align 4 ; przesuniecie do adresu podzielnego na 4
 naglowA DB 0Dh,0Ah, "Wprowadź 4 parametry fun. f() = A/B-C+D",0  ; nagłówek
 align 4 ; przesuniecie do adresu podzielnego na 4
 rozmnaglA DD $ - naglowA
 align 4
 naglowB DB 0Dh,0Ah, "Wprowadź 4 parametry fun. f() = A#B*~C|D",0  ; nagłówek
 align 4 ; przesuniecie do adresu podzielnego na 4
 rozmnaglB DD $ - naglowB
 align 4 ; przesuniecie do adresu podzielnego na 4
 zaprA DB 0Dh,0Ah,"Proszę wprowadzić argument a [+Enter]: ",0
 align 4
 rozmA DD $ - zaprA ;ilość znaków w tablicy
 zmA DD 1 ; argument a
 zaprB DB 0Dh,0Ah,"Proszę wprowadzić argument b [+Enter]: ",0
 align 4
 rozmB DD $ - zaprB ;ilość znaków w tablicy
 zmB DD 2 ; argument b
 zaprC DB 0Dh,0Ah,"Proszę wprowadzić argument c [+Enter]: ",0
 align 4
 rozmC DD $ - zaprC ;ilość znaków w tablicy
 zmC DD 3 ; argument c
 align 4
 zaprD DB 0Dh,0Ah,"Proszę wprowadzić argument d [+Enter]: ",0
 align 4
 rozmD DD $ - zaprD ;ilość znaków w tablicy
 align 4
 zmD DD 4 ; argument d
 align 4
 zadA DB 0Ah,"Zadanie a) ",0              ; nagłówek zadania A
 align 4 ; przesuniecie do adresu podzielnego na 4
 rozmzadA DD $ - zadA
 align 4 ; przesuniecie do adresu podzielnego na 4
 zadB DB 0Dh,0Ah,"Zadanie b) ",0               ; nagłówek zadania B
 align 4
 rozmzadB DD $ - zadB
 align 4
```

```
 zadC DB 0Dh,0Ah,"Zadanie c) ",0              ; nagłówek zadania C
 align 4
 rozmzadC DD $ - zadC
 align 4
 rot1 DB 0Dh,0Ah,"Liczba binarna: ",0
 align 4
 rozmrot1 DD $ - rot1
 rot2 DB 0Dh,0Ah,"Cykl.prawo CF4: ",0
 align 4
 rozmrot2 DD $ - rot2
 rot3 DB 0Dh,0Ah,"W lewo 2 razy : ",0
 align 4
 rozmrot3 DD $ - rot3
 wzorf      DB 0Dh,0Ah,"Funkcja f() = %4ld",0
 align 4
 rout DD 0 ;faktyczna ilość wyprowadzonych znaków
 rinp DD 0 ;faktyczna ilość wprowadzonych znaków
 rinp2 DD 0 ;faktyczna ilość wprowadzonych znaków
 bufor DB mbuf dup(?)
 bufor2 DB mbuf dup(?)
 rbuf DD mbuf
   wyn   DD  0 ; zienna do przechowywania wyniku
   st0   DD  10100110001110000111100000111110b
_data ends
;------------

_text segment
start:
;--- wywołanie funkcji GetStdHandle - MAKRO
 podajdeskr STD_OUTPUT_HANDLE, hout
 podajdeskr STD_INPUT_HANDLE, hinp
;--- nagłówek ---------
 plznaki naglow, bufor ; konwersja polskich znaków - MAKRO
 wyswietl bufor, rozmN
 plznaki temat, bufor ; konwersja polskich znaków - MAKRO
 wyswietl bufor, rozmT
;--- wyświetlenie nowej linni---------
 nowalinia nl
;--- zaproszenie A ---------
 plznaki zadA, bufor2
 wyswietl bufor2, rozmzadA
 ;--- new line ---------
 nowalinia nl
 plznaki naglowA, bufor ; konwersja polskich znaków - MAKRO
 wyswietl bufor, rozmnaglA
 plznaki zaprA, bufor
;--- wyświetlenie zaproszenia A ---
 wyswietl bufor, rozmA   ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmA
;--- zaproszenie B ---------
 plznaki zaprB, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia B ---
 wyswietl bufor, rozmB ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmB
;--- zaproszenie C ---------
 plznaki zaprC, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia C ---
 wyswietl bufor, rozmC ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmC
;--- zaproszenie D ---------
 plznaki zaprD, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia D ---
 wyswietl bufor, rozmD ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmD
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- obliczenia Funkcja y =a/b-c+d
     push zmD
     push zmC
     push zmB
     push zmA
     call arytm
     add  esp, 16
```

```
      mov  wyn, eax
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- wyprowadzenie wyniku obliczeń ---
 invoke wsprintfA,OFFSET bufor,OFFSET wzorf,wyn ; zwraca ilość znaków w buforze
 movrinp, eax; zapamiętywanie ilości znaków
 ;--- new line ---------
 nowalinia nl
 wyswietl bufor, rinp
;--- wyświetlenie nowej linni---------
 nowalinia nl
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- zaproszenie B ---------
 plznaki zadB, bufor2
 wyswietl bufor2, rozmzadB
 ;--- wyświetlenie nowej linni---------
 nowalinia nl
 plznaki naglowB, bufor ; konwersja polskich znaków - MAKRO
 wyswietl bufor, rozmnaglB ; wywołanie funkcji WriteConsoleA
 plznaki zaprA, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia A ---
 wyswietl bufor, rozmA ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmA
;--- zaproszenie B ---------
 plznaki zaprB, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia B ---
 wyswietl bufor, rozmB ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmB
;--- zaproszenie C ---------
 plznaki zaprC, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia C ---
 wyswietl bufor, rozmC ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmC
;--- zaproszenie D ---------
 plznaki zaprD, bufor; konwersja polskich znaków
;--- wyświetlenie zaproszenia D ---
 wyswietl bufor, rozmD ; wywołanie funkcji WriteConsoleA
;--- czekanie na wprowadzenie znaków, koniec przez Enter ---
 zmienna hinp, bufor, rbuf, rinp, zmD
 ;--- new line ---------
 nowalinia nl
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- obliczenia Funkcja y =a#b*~c|d
 invoke logika,zmA,zmB,zmC,zmD
 add   esp,16
    mov   wyn,eax
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- wyprowadzenie wyniku obliczeń ---
 invoke wsprintfA,OFFSET bufor,OFFSET wzorf,wyn ; zwraca ilość znaków w buforze
 movrinp2, eax ; zapamiętywanie ilości znaków
;--- wyświetlenie wyniku ---------
 wyswietl bufor, rinp2
;--- new line ---------
 nowalinia nl
;----------------
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;--- zaproszenie C ---------
 plznaki zadC, bufor2
 wyswietl bufor2, rozmzadC
;;;; w prawo CFc4, w lewo 2
;--- new line ---------
 nowalinia nl
 ;---------------------------
 invoke przesuwanie, st0
 add esp, 16
;--- zakończenie procesu ---------
 invoke ExitProcess, SYS_exit; wywołanie funkcji ExitProcess
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;---------Podprogramy
ScanInt proc c adres
;; funkcja ScanInt przekształca ciąg cyfr do liczby, którą jest zwracana przez eax
;; argument - zakończony zerem wiersz z cyframi
;; rejestry: ebx - adres wiersza, edx - znak liczby, esi - indeks cyfry w wierszu, edi - tymczasowy
;--- początek funkcji
```

```
;--- odkładanie na stos
 push ebx
 push ecx
 push edx
 push esi
 push edi
;--- przygotowywanie cyklu
 invoke lstrlenA, adres
 movedi, eax ;ilość znaków
 movecx, eax;ilość powtórzeń = ilość znaków
 xor esi, esi ; wyzerowanie esi
 xor edx, edx; wyzerowanie edx
 xor eax, eax; wyzerowanie eax
 movebx, adres
;--- cykl
pocz: cmpBYTE PTR [ebx+esi], 02Dh ;porównanie z kodem '-'
 jne @F
 movedx, 1
 jmp nast
@@:cmpBYTE PTR [ebx+esi], 030h;porównanie z kodem '0'
 jae @F
 jmp nast
@@:cmpBYTE PTR [ebx+esi], 039h;porównanie z kodem '9'
 jbe @F
 jmp nast
;----
@@:push edx ; do edx procesor może zapisać wynik mnożenia
 movedi, 10
 mul edi  ;mnożenie eax * edi
 movedi, eax ; tymczasowo z eax do edi
 xor eax, eax;zerowani eax
 movAL, 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
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DrukBin proc stdcall liczba:dword
;; funkcja DrukBin wyswietla liczbę-argument w postaci binarnej
;; rejestry: ecx - cykl, edi - maska, esi - indeks w buforze, ebx - przesunięcie bufora
;--- odkładanie na stos
   push    ecx
   push    edi
   push    esi
   push    ebx
;---
   mov     ecx,32
   mov     edi,80000000h
   mov     esi,0
   mov     ebx,OFFSET bufor
et1:
   mov     BYTE PTR [ebx+esi],'0'
   test    liczba,edi
   jz      @F
   inc     BYTE PTR [ebx+esi]
@@:
   shr     edi,1
   inc     esi
   loopnz  et1
```

```
   mov     BYTE PTR [ebx+32],0Dh
   mov     BYTE PTR [ebx+33],0Ah
;--- wyświetlenie wyniku ---------
   invoke WriteConsoleA,hout,OFFSET bufor,34,OFFSET rout,0
;--- zdejmowanie ze stosu
   pop     ebx
   pop     esi
   pop     edi
   pop     ecx
;--- powrót
   ret     8
DrukBin ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
option prologue: none
option epilogue: none
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
arytm proc c
;--- obliczenia A / B - C + D
 push ebp
 mov ebp, esp
 mov edx, 0      ; zerowanie edx
 mov eax, 0           ; zerowanie eax
 mov eax, dword ptr [ebp+8]  ; zm A do eax
 div dword ptr [ebp+12]  ; dzielenie  A / B wynik w eax
 mov edx, dword ptr [ebp+16] ; zmienna C do edx
 sub eax, edx     ; odejumjemy od wyniku dzielenia C, wynik w eax
 add eax, dword ptr [ebp+20] ; dodajemy do eax zmienną D, wynik w eax
 mov edx, 0           ; sprzątanie, zerowanie edx
 mov esp,ebp
 pop ebp
 ret 8 ;ominięcie ramki stosu z parametrami
arytm endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
option prologue: prologuedef
option epilogue: epiloguedef
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
logika proc stdcall argA:dword,argB:dword,argC:dword,argD:dword
;--- obliczenia a # b * ~c | d
;--- kolejność ~ * | #
 push ebp ;przechowywanie ebp na stosie
 mov ebp, esp ;zamiana ebp
 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
 mov esp, ebp ;zamiana esp
 pop ebp ;przewrócenie ebp ze stosu
 ret 8 ;ominięcie ramki stosu z parametrami
logika endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
przesuwanie proc stdcall arg:dword
 push ebp
 mov ebp, esp
 plznaki rot1, bufor
 wyswietl bufor, rozmrot1
 invoke DrukBin, st0
 mov eax, 0
 mov eax, st0
 rcr eax, 4
 mov st0, eax
 plznaki rot2, bufor
 wyswietl bufor, rozmrot2
 invoke DrukBin, st0
 mov eax, 0
 mov eax, st0
 rol eax, 2
 mov st0, eax
 plznaki rot3, bufor
 wyswietl bufor, rozmrot3
 invoke DrukBin, st0
 mov esp, ebp ;zamiana esp
 pop ebp ;przewrócenie ebp ze stosu
 ret 8 ;ominięcie ramki stosu z parametrami
```

przesuwanie endp
end start
_text ends

Testowanie

```
D:\workspace\Programowanie\rok3\asm>cw5\cw5
Autor aplikacji Grzegorz Makowski i53
Podprogramy i makrodefinicje.

Zadanie a)

Wprowadź 4 parametry fun. f() = A/B-C+D
Proszę wprowadzić argument a [+Enter]:    50

Proszę wprowadzić argument b [+Enter]:    2

Proszę wprowadzić argument c [+Enter]:    5

Proszę wprowadzić argument d [+Enter]:    1


Funkcja f() =   21

Zadanie b)

Wprowadź 4 parametry 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


Funkcja f() =    0

Zadanie c)

Liczba binarna:  01010011000111000011110000011110

Cykl.prawo CF4:  11100101001100011100001111000011

W lewo 2 razy :  00010100110001110000111100001111

D:\workspace\Programowanie\rok3\asm>
```