Lab 6 – Języki wysokiego poziomu

```
; procedura sumowania dwóch liczb
section .text use32
global      _sum
_sum:
    %idefine        a       [ebp+8]
    %idefine        b       [ebp+12]

            push    ebp
            mov     ebp, esp

            mov     eax, a
            add     eax, b

    ; LEAVE = mov esp, ebp / pop ebp
            leave
            ret
```

```
#include <stdio.h>
extern int _sum (int a, int b); /* external
   function declaration */
int sum (int a, int b);    /* function prototype */
int c=1, d=2;
int main()
{
   printf("%d\n", sum(c,d));
   return 0;
}
```

Kompilacja:

1. nasm -o procedure.obj -f obj procedure.asm
2. bcc32 interface.c procedure.obj

```
#include <iostream>
using namespace std;
extern "C" unsigned int sum (unsigned int, unsigned int);

int main()
{
 unsigned int a,b;
 cout<<"Podaj liczbe a: ";
 cin>>a;

cout<<"Podaj liczbe b: ";
cin>>b;
cout<<"Wynik dodawania to: "<<sum(a,b)<<endl;
return 0;
}
```

- Zadanie:

  Napisać program, który pobierze z klawiatury argumenty oraz wykona jedną z dwóch operacji: dodawanie lub mnożenie oraz wyświetli wynik na ekranie.

Lab 7 – MMX

- **M**ulti**M**edia e**X**tensions
- Zestaw 57 instrukcji do przetwarzania dużych ilości złożonych danych przez pojedynczy algorytm
- W określonych wyżej zastosowaniach możliwy znaczący wzrost szybkości działania programów
- Przykłady:
  - Grafika 3D
  - MPEG, JPEG itp..
  - FFT itp..

- Rejestry MMX (64-bit): **mm0**, **mm1**,... **mm7**.
- Operacje na danych spakowanych:
  - 8 × 8 bitów (*packed byte*),
  - 4 × 16 bitów (*packed word*),
  - 2 × 32 bity (*packed dword*),
  - 1 × 64 bity (*quad word*).
- Arytmetyka nasyceniowa i arytmetyka modulo
- Mnemoniki rozkazów:
  - **p**
  - 3-4 literowy skrót działania jakie dany rozkaz wykonuje (np. add, sub, mul)
  - litera s lub u określająca, czy działanie wykonywane jest – odpowiednio – na liczbach ze znakiem (*signed*) lub bez znaku (*unsigned*)
  - litera s jeśli operacja jest wykonywana z nasyceniem
  - rozmiar komórki wektora: b – bajt (8 bitów), w – słowo (16 bitów), d – podwójne słowo (32 bity)
- Przykłady: PADDB, PADDBUSB

```
%define ITERACJE 8 ; deklarujemy liczbę iteracji
org 100h
start:
    movq mm0, [tablica1] ;ładujemy do pierwszego rejestru 8 bajtów z tablicy1
    paddb mm0, [tablica2] ;dodajemy do wartości rejestru 8 bajtów z tablicy2
    movq [tablica3], mm0 ;wynik do tablicy3
    mov cx, ITERACJE ;liczba iteracji
    mov ah, 2 ;do wypisywania na ekran

petla:
    mov si, tablica3 ;zapisujemy adres tablicy3 do si
    add si, ITERACJE ;
    sub si, cx
    mov dx, [si]
    int 21h
    loop petla

    mov ah, 4Ch
    int 21h

tablica1 times ITERACJE db 41
tablica2 times ITERACJE db 28
tablica3 times ITERACJE db 65 ;;69
```

```
;Program ilustrujący arytmetykę nasycenia
    ;MMX

%define ITERACJE 8 ; deklarujemy liczbę
    iteracji

org 100h

start:
    movq mm0, [tablica1] ;ładujemy do
    pierwszego rejestru 8 bajtów z tablicy1
    paddb mm0, [tablica2] ;dodajemy BEZ
    ;NASYCENIA

    movq [tablica3], mm0 ;wynik do tablicy3

    xor dx,dx
    mov dx, [tablica3]

    mov ah, 2 ;do wypisywania na ekran
    int 21h ;wywołanie funkcji DOSowej

movq mm0, [tablica1] ;ładujemy do
    ;pierwszego rejestru 8 bajtów z tablicy1
paddusb mm0, [tablica2] ;dodajemy Z
    ;NASYCENIEM

movq [tablica4], mm0 ;wynik do tablicy4

xor dx,dx
    mov dx, [tablica4]
    sub dx, 200

    int 21h ;wywołanie funkcji DOSowej

  mov ah, 4Ch
  int 21h

tablica1 times ITERACJE db 121
tablica2 times ITERACJE db 200
tablica3 times ITERACJE db 0
tablica4 times ITERACJE db 0
```

- Zadanie:

  Napisać program, który pobierze z klawiatury napis, a następnie przy pomocy instrukcji MMX zamieni znaki na inne, o zwiększonej wartości kodów ASCII (np. o 8).