


|   |  |                    |   |    |
|---|--|--------------------|---|----|
|  | Instytut Informatyki Politechniki Śląskiej<br><br>Zespół Mikroinformatyki i Teorii Automatów Cyfrowych |                    |  |    |
| Rok akademicki:   | Rodzaj studiów*: SSI/NSI/NSM   | Języki Asemblerowe | LAB   | II |

## TEMAT:

Przekazywanie parametrów do procedur asemblerowych dla procesorów x64.

## CEL:

Celem ćwiczenia jest poznanie metod przekazywania parametrów procesora x64. Wykorzystując szablon projektu z ćw. 1 napisać procedury asemblerowe umożliwiające przekazywanie parametrów typu int, float, mixed int, \_m64 i \_m128 (SSE) w celu dokonania obliczeń ilustrujących ich działanie. Wynik działania należy zwrócić do programu głównego i udokumentować.

Konstrukcja projektu zakłada możliwość wywoływania funkcji bibliotecznych napisanych w asemblerze z poziomu aplikacji oraz pokazuje prawidłową konfigurację środowiska umożliwiającą debugowanie kodu do poziomu asemblera, obserwację stanu rejestrów i flag procesora czy obszarów pamięci danych.

## ZAŁOŻENIA:

W środowisku VS 2022 zakładamy solucję JALab3 składającą się z dwóch projektów:

- JALab3 Projekt aplikacja X64 w j. C++,
- JALib3 Projekt biblioteka DLL w asemblerze,

W bibliotece DLL utworzone są funkcje asemblerowe przekazywania parametrów zgodnie z podanym niżej schematem.

## WYKONANIE:

Treść pliku JAAsm.asm jest następująca:

1. Przecwiczyć przekazywanie do procedury *MyProc1* parametrów wg wzoru poniżej. W procedurze *MyProc1* należy wykonać operacje na rejestrach w sposób podobny, jak w przykładzie powyżej wykonując wybrane operacje matematyczne pokazujące prawidłowość przekazania parametrów do procedury ASM:

### Example of argument passing 1 - all integers

```
func1 (int a, int b, int c, int d, int e, int f);
// a in RCX, b in RDX, c in R8, d in R9, f then e pushed on stack
```

### Example of argument passing 2 - all floats

```
func2 (float a, double b, float c, double d, float e, float f);
// a in XMM0, b in XMM1, c in XMM2, d in XMM3, f then e pushed on stack
```

### Example of argument passing 3 - mixed ints and floats

```
func3 (int a, double b, int c, float d, int e, float f);
// a in RCX, b in XMM1, c in R8, d in XMM3, f then e pushed on stack
```

#### Example of argument passing 4 - \_\_m64, \_\_m128, and aggregates

```
func4(__m64 a, __m128 b, struct c, float d, __m128 e, __m128 f);  
// a in RCX, ptr to b in RDX, ptr to c in R8, d in XMM3,  
// ptr to f pushed on stack, then ptr to e pushed on stack
```

#### Przykład użycia:

```
#include <iostream>  
using namespace std;  
  
extern "C" void myProc1(int a, float b, double c);  
  
int main()  
{  
    int a = 1;  
    float b = 2.0;  
    double c = 3.0;  
    myProc1(a, b, c);  
    return 0;  
}  
  
.code  
myProc1 proc a:DWORD, b:REAL4, c:REAL8  
    ; code here  
    ret  
myProc1 endp
```

#### ZADANIE

1. Utworzyć rozwiązanie **JALab3** wraz z projektami **JALab3** oraz **JALib3**. Napisać kod przekazujący parametry do procedury ASM wg opisu powyżej. Wykonać obliczenia w procedurze asemblerowej tak aby udokumentować prawidłowość przekazania parametrów
2. W programie **JALab3** utworzyć bufor typu BYTE wypełniony wzorcem. Przekazać jego adres do procedury asemblerowej w pliku JAAsm.asm, w procedurze dokonać modyfikacji zawartości bufora,
3. W programie głównym wykonać prezentację, że bufor został zmodyfikowany w procedurze asemblerowej.
4. Wygenerować sprawozdanie dokumentujące działania opisane w zadaniu w postaci PDF