Silnia:

```
.code
        Test64 proc
                XOR rax, rax;
        skok:
                ADD rax, rcx;
                DEC rcx;
                JNZ skok;
                RET
                Test64 endp
                end
```

Coś z macierzami: // Mnożenie macierzy

```
Ex tern „C” INT64 mnoz(INT64**,    INT64*,      INT64*,          INT64,    INT64);
                    Macierz(rcx)   wektor_wej(rdx) wektor_wyj(r8) N1 (R9)  N2 (R10)
.code
    mnoz proc uses rbx rsi, macierz: ptr, wektor_wej: ptr, wektor_wyj: ptr, N1: qword, N2: qword;

    MOV r11, N2;
P1: MOV rsi, [rcx + r9 * 8 – 8];
    MOV rbx, r11;
    XOR r10, r10;
P2: MOV rax, [rsi + rbx * 8 – 8];
    IMUL rax, [rdx + rb. * 8 - 8];
    ADD r10, rax;
    DEC rbx;
    JNZ p2;
    MOV [r8 + r9 * 8 – 8], r10;
    DEC r9;
    JNZ p1;
    ret;
mnoz endp
end
```

```
    // Suma macierzy
    Ex tern „C” INT64 suma(INT64**, INT64**, INT64, INT64);
                                rcx         rdx         r8 (n1)   r9(n2)
    .code
    suma proc uses rbx rsi rdi, A: ptr, B: ptr, N1: qword, N2: qword;
    p1:     MOV rdi, [rcx + r8 * 8 – 8];
            MOV rsi, [rdx + r8 * 8 – 8];
            MOV r10, N2;
            MOV rax, [rsi + 8 * r10 – 8];
            ADD rax, [rdi + 8 * r10 – 8];
            MOV [rdi + 8 * r10 – 8], rax;
            DEC r10;
            JNZ p2;
            DEC r8;
            JNZ p1;
    suma endp
    end
```

```
//  Petla dodająca liczby np. a = 3, 1+2+3

.code
Test64 proc
        XOR rax, rax;
Skok:
        Add rax, rcx;
        Dec rcx;
        Jnz skok;
        Ret
Test64 endp
End
```

```
// Dodawanie liczb a oraz b
.code
Test64 proc
        CMP rcx, rdx;
        JNL abc;
        MOV rcx, rdx;
abc:    MOV rax, rcx;
ret
Test64 endp
end
```