Zad 7.

piątek, 24 marca 2023 22:03

Zadanie 7. Zaimplementuj w asemblerze x86-64 funkcję liczącą wyrażenie «x * y». Argumenty i wynik funkcji są 128-bitowymi liczbami całkowitymi bez znaku. Argumenty i wynik są przypisane do tych samych rejestrów co w poprzednim zadaniu. Instrukcja «mul» wykonuje co najwyżej mnożenie dwóch 64-bitowych liczb i zwraca 128-bitowy wynik. Wiedząc, że $n = n_{127\ldots64} \cdot 2^{64} + n_{63\ldots0}$, zaprezentuj metodę obliczenia iloczynu, a dopiero potem przetłumacz algorytm na asembler.

UWAGA! Zapoznaj się z dokumentacją instrukcji «mul» ze względu na niejawne użycie rejestrów %rax i %rdx.

$$x \cdot y = (x_{127..64} \cdot 2^{64} + x_{63..0})(y_{127..64} \cdot 2^{64} + y_{63..0}) =$$

$$= x_{127..64} \cdot y_{127..64} \cdot 2^{128} + x_{63..0} y_{127..64} \cdot 2^{64} +$$

$$+ y_{63..0} x_{127..64} \cdot 2^{64} + x_{63..0} y_{63..0}$$

$y_{63..0}$: 64 bity wszędzie

$\cdot 2^{64}$: << 64

$x_{63..0} y_{63..0}$: to się nie zmieści tak czy siak

```
ELSE (* OperandSize = 64 *)
    RDX:RAX := RAX * SRC;
```

```
# x -> %rdi:rsi, y -> %rdx:rcx, res -> %rdx:rax
# mul: %rdx:rax = %rax * SRC
multiplication: movq %rdx, %r8 # %r8 = y[127..64]
                movq %rsi, %rax # %rax = x[63..0]
                mul %r8 # %rdx:rax = y[127..64] * x[63..0]
                movq %rax, %r8 # %r8 = y[127..64] * x[63..0] * 2^64
                movq %rcx, %rax # %rax = y[63..0]
                mul %rdi # %rdx:rax = y[63..0] * x[127..0]
                add %rax, %r8 # %r8 = (y[127..0]*x[63..0] + y[63..0]*x[127..0])*2^64
                movq %rsi, %rax # %rax = x[63..0]
                mul %rcx # %rdx:rax = x[63..0]*y[63..0]
                add %r8, %rdx # add high-order bytes and leave low-order bytes
                ret
```