Słownik nieludzko-ludzki

AT&T	Intel
`(%ebx),%eax`	`eax,[ebx]`
`3(%ebx),%eax`	`eax,[ebx+3]`
`(%ebx,%ecx,0x2),%eax`	`eax,[ebx+ecx*2h]`
`(%ebx, %ecx),%eax`	`eax,[ebx+ecx]`
`-0x2(%ebx,%ecx,0x4),%eax`	`eax,[ebx+ecx*4h-2h]`
`0xA(,%rcx,4),%rdx`	`rdx,[rcx*4+0xA]`
`(base,index,scale),foo`	`[base+index*scale+disp]`

Prefixy

```
b -> byte  -> 8bit  -> Xh/Xl
w -> word  -> 16bit -> Xx
s -> short -> 16bit -> Xx (unused?)
l -> long  -> 32bit -> EXY
q -> quad  -> 64bit -> RXY
```

Zadanie 5

```
[bits 64]

; rdi input
; rax output

mov rdi, 0x1122334455667788

; ------
mov rax, rdi

; mov ebx, eax
; ror bx, 8
; ror ebx, 16
; ror bx, 8
mov r8d, eax
ror r8w, 8
ror r8d, 16
ror r8w, 8

ror rax, 32

; mov ecx, eax
; ror cx, 8
; ror ecx, 16
; ror cx, 8
mov r9d, eax
ror r9w, 8
ror r9d, 16
ror r9w, 8

; mov eax, ebx
; shl rax, 32
; add rax, rcx
mov eax, r8d
shl rax, 32
add rax, r9
```

Zadanie 6

```
[bits 64]

mov rdi, 0xffffffffffffffff ; high a
mov rsi, 0xffffffffffffffff ; low a
mov rdx, 0xffffffffffffffff ; high b
mov rcx, 0x1                ; low b

; output = rax:rdx
; tmp = r8 .. r11
; r8 = low
; r9 = carry

mov r8, rsi     ; r8 = rsi
shr rsi, 63     ; rasi >>> 63
add r8, rcx     ; r8 = r8 + rcx
mov r9, rdx     ; r9 = rdx
shr r9, 63      ; r9 >>> 63
xor rsi, r9     ; rsi = rsi ^ r9

mov rax, rdi    ; rax <- rdi
add rax, rdx    ; rax = rax + rdx
add rax, rsi    ; rax = rax + rsi
mov rdx, r8     ; rdx = r8
```

Zadanie 7

```
[bits 64]

mov rdi, 0xfffffffffce1232 ; high a
mov rsi, 0xffffffffffffffff ; low a
mov rdx, 0xffffffff565854648 ; high b
mov rcx, 0x00000000007a7bd1 ; low b

; Algorytm Karacuby
; RAX = (High A * Low B + High B * Low A) + High bits of Lows multiplication
; RDX = Low bits of Lows multiplication
; mul r/m64, r/m64 -> RDX:RAX
; r8  = rdx copy
; r9  = low multiplication
; r10 = rest of low multiplication
; r11 = rest of first high multiplication

mov r8, rdx  ; r8 = rdx
mov rax, rsi ; rax = rsi
mul rcx      ; rcx * rax -> rdx:rax
mov r9, rax  ; r9 = rax
mov r10, rdx ; r10 = rdx
; Tu jest k
mov rax, rdi ; rax = rdi
; High a * Low b
mul rcx      ; rdi * rcx -> rdx:rax
mov r11, rax ; r11 = rax
mov rax, r8  ; rax = r8
; High b * Low a
mul rsi      ; r8 * rsi -> rdx:rax
add rax, r10 ; rax = rax + r10
add rax, r11 ; rax = rax + r11
mov rdx, r9  ; rdx = r9
```