Zad 1.

środa, 12 kwietnia 2023 14:55

Zadanie 1. Poniższy wydruk otrzymano w wyniku deasemblacji rekurencyjnej procedury zadeklarowanej następująco: «`long pointless(long n, long *p)`». Zapisz w języku C kod odpowiadający tej procedurze. Następnie opisz zawartość jej **rekordu aktywacji** (ang. *stack frame*). Wskaż **rejestry zapisane przez funkcję wołaną** (ang. *callee-saved registers*), zmienne lokalne i adres powrotu. Następnie uzasadnij, że wartość rejestru `%rsp` w wierszu 11 jest podzielna przez 16 – zgodnie z [1, 3.2.2]. Zastanów się czemu autorzy **ABI** zdecydowali się na taką konwencję. (*)

```
1  pointless:
2        pushq   %r14
3        pushq   %rbx
4        pushq   %rax
5        movq    %rsi, %r14
6        movq    %rdi, %rbx
7        testq   %rdi, %rdi
8        jle     .L1
9        leaq    (%rbx,%rbx), %rdi
10       movq    %rsp, %rsi
11       callq   pointless
12       addq    (%rsp), %rax
13       jmp     .L3
14 .L1:  xorl    %eax, %eax
15 .L3:  addq    %rax, %rbx
16       movq    %rbx, (%r14)
17       addq    $8, %rsp
18       popq    %rbx
19       popq    %r14
20       retq
```

prolog

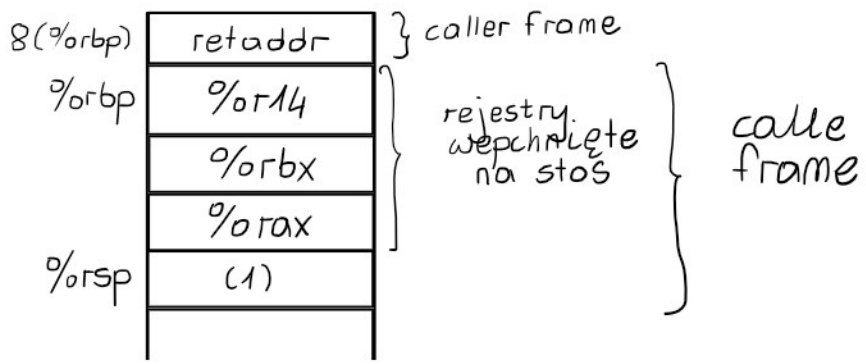
epilog

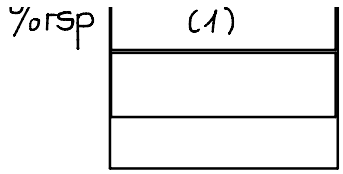
```
50 long pointless(long n, long *p) {
51     long result;
52     if (n > 0)
53         result = pointless(n << 1, &result) + result; // ??
54     else
55         result = 0;
56     *p = result + n;
57     return result;
58 }
```

{ zapisane: %r14, %rbx, %rsp, %rbp ← rejestry zachowywane wartość PO wywołaniu
zmienne lokalne: long result ← (w %rax, nie na stosie)
adres powrotu: 8(%rbp) ↑ na końcu ramki wołającego

%rdi → long n
%rsi → long *p
%rax → long result, wynik zawołania funkcji

stack frame





przy wołaniu instrukcji call, na stos zostanie dodany adres powrotu (1) i widać, że skoro ramka ma wtedy rozmiar 64 bajtów to %rsp jest zaligrowany do 16