Systemy komputerowe

Lista zadań nr 4

Na zajęcia 21 – 25 marca 2019

Lista zawiera zadania do rozwiązania podczas pierwszej godziny ćwiczeń. Podczas drugiej godziny ćwiczeń prowadzący, przy pomocy rzutnika, zademonstruje podstawy użytkowania debuggera GDB oraz wybranych narzędzi z pakietu `binutils`: GAS, `ld` i `objdump`. Rozwiąże też jedno z zadań z samouczka, który pojawi się na stronie wykładu w SKOSie. Za wyręczenie prowadzącego w tym ostatnim zadaniu będzie można otrzymać punkt bonusowy, spoza domyślnej puli punktów.

Zadanie 1. Przeczytaj poniższy kod w języku C i odpowiadający mu kod w asemblerze, a następnie wywnioskuj jaka jest wartość stałej «CNT» i jak wygląda definicja struktury «`a_struct`».

```
1  typedef struct {
2    int32_t first;
3    a_struct a[CNT];
4    int32_t last;
5  } b_struct;
6
7  void test (int64_t i, b_struct *bp) {
8    int32_t n = bp->first + bp->last;
9    a_struct *ap = &bp->a[i];
10   ap->x[ap->idx] = n;
11 }
```

```
1 test:
2 movl   0x120(%rsi),%ecx
3   addl   (%rsi),%ecx
4   leaq   (%rdi,%rdi,4),%rax
5   leaq   (%rsi,%rax,8),%rax
6   movq   0x8(%rax),%rdx
7   movslq %ecx,%rcx
8   movq   %rcx,0x10(%rax,%rdx,8)
9   retq
```

Zadanie 2. Zdefiniuj semantykę operatora «`?:`» z języka C. Jakie zastosowanie ma poniższa funkcja.

```
1 int32_t cread(int32_t *xp) {
2   return (xp ? *xp : 0);
3 }
```

Używając serwisu `godbolt.org` (kompilator `x86-64 gcc 8.2`) sprawdź, czy istnieje taki poziom optymalizacji (`-O0`, `-O1`, `-O2` lub `-O3`), że wygenerowany dla `cread` kod asemblerowy nie używa instrukcji skoku. Jeśli nie, to zmodyfikuj funkcję `cread` tak, by jej tłumaczenie na asembler spełniało powyższy warunek.

Wskazówka: Dążysz do wygenerowania kodu używającego instrukcji cmov. Końcowej instrukcji ret nie uważamy w tym zadaniu za instrukcję skoku.

Zadanie 3. Zapoznaj się z tłumaczeniem do asemblera C-owej instrukcji wielokrotnego wyboru `switch`. Następnie przetłumacz poniższą funkcję «`long switch_prob(long x, long n)`» z powrotem na język C.

```
400590 <switch_prob>:
400590: 48 83                   subq  $0x3c,%rsi
400594: 48 83 fe 05             cmpq  $0x5,%rsi
400598: 77 29                   ja    *0x4005c3
40059a: ff 24 f5 f8 06 40 00    jmpq  *0x4006f8(,%rsi,8)
4005a1: 48 8d 04 fd 00 00 00 00 lea   0x0(,%rdi,8),%rax
4005a9: c3                      retq
4005aa: 48 89 f8                movq  %rdi,%rax
4005ad: 48 c1 f8 03             sarq  $0x3,%rax
4005b1: c3                      retq
4005b2: 48 89 f8                movq  %rdi,%rax
4005b5: 48 c1 e0 04             shlq  $0x4,%rax
4005b9: 48 29 f8                subq  %rdi,%rax
4005bc: 48 89 c7                movq  %rax,%rdi
4005bf: 48 0f af ff             imulq %rdi,%rdi
4005c3: 48 8d 47 4b             leaq  0x4b(%rdi),%rax
4005c7: c3                      retq
```

Zrzut pamięci przechowującej tablicę skoków:

```
(gdb) x/6gx 0x4006f8
0x4006f8: 0x4005a1
0x400700: 0x4005a1
0x400708: 0x4005b2
0x400710: 0x4005c3
0x400718: 0x4005aa
0x400720: 0x4005bf
```

Wskazówka: CSAPP3e:3.6.9

Zadanie 4. W języku C struktury mogą być zarówno argumentami funkcji, jak i wartościami zwracanymi przez funkcje[1]. Za pomocą serwisu `godbolt.org` zapoznaj się z tłumaczeniem do asemblera funkcji przyjmujących pojedynczy argument każdego z poniższych typów strukturalnych. Następnie zapoznaj się z tłumaczeniem funkcji zwracających wartość każdego z tych typów. Jakie reguły dostrzegasz?

```c
struct first {
int32_t val1;
};
struct second {
int32_t val2;
int32_t val1;
};
struct third {
int32_t val3;
int32_t val2;
int32_t val1;
};
```

```c
struct fourth {
int32_t val4;
int32_t val3;
int32_t val2;
int32_t val1;
};
struct fifth {
int16_t val6;
int16_t val5;
int32_t val3;
int32_t val2;
int32_t val1;
};
```

```c
struct sixth {
int32_t val7;
int32_t val4;
int32_t val3;
int32_t val2;
int32_t val1;
};
struct seventh {
int32_t val8[10000];
int32_t val4;
int32_t val3;
int32_t val2;
int32_t val1;
};
```

Czy przekazywanie/zwracanie dużych struktur funkcjom jest dobrym pomysłem?

[1] Konwencja przekazywania/zwracania argumentów takich typów jest częścią ABI: https://github.com/hjl-tools/x86-psABI/wiki/x86-64-psABI-1.0.pdf, strony 19–25