

Konstrukcja kompilatorów

Lista zadań nr 0

Na zajęcia 2 października 2024

UWAGA! Poniższe zadania należy rozwiązać używając kompilatora GCC w wersji 6 (lub wyższej) generującego kod dla procesorów x86-64. W linii poleceń kompilatora **musisz** dodać następujące opcje: «-march=nehalem -fomit-frame-pointer -Os»!

Wskazówka: Przed przystąpieniem do rozwiązywania zadań należy zapoznać się z rozdziałem piątym trzeciego wydania podręcznika „Computer Systems: A Programmer’s Perspective”.

Zadanie 1. Skompiluj poniższą procedurę, a następnie na podstawie wyprodukowanego kodu w asemblerze zapisz na tablicy zoptymalizowany program w języku C. Jakie właściwości programu wykrył kompilator i wykorzystał je w trakcie optymalizacji?

```
1 void set_row(long *a, long *b, long i, long n) {
2     for (long j = 0; j < n; j++)
3         a[n * i + j] = b[j];
4 }
```

Zadanie 2. Skompiluj poniższą procedurę i przeanalizuj wygenerowany kod w asemblerze. Jakie założenie poczynił kompilator na podstawie oznaczenia wskaźników «a» i «b» słowem kluczowym «restrict»?

```
1 /* a - macierz n x n, b - wektor n */
2 void sum_rows(double *restrict a, double *restrict b, long n) {
3     for (long i = 0; i < n; i++) {
4         b[i] = 0;
5         for (long j = 0; j < n; j++)
6             b[i] += a[i * n + j];
7     }
8 }
```

Wskazówka: Rozważ liczbę generowanych dostępow do poszczególnych elementów tablicy «b».

Zadanie 3. Z użyciem składni «__attribute__» programista może poinformować kompilator o szczególnych właściwościach procedur, zmiennych, itd. W poniższym przypadku oznaczyliśmy procedurę «my_strlen», która zachowuje się identycznie jak «strlen» z biblioteki standardowej, jako **funkcję czystą**. Jakiej optymalizacji nie byłby w stanie przeprowadzić kompilator bez tej informacji?

```
1 __attribute__((pure)) unsigned long my_strlen(const char *s);
2
3 const char *my_index(const char *s, char v) {
4     for (unsigned long i = 0; i < my_strlen(s); i++)
5         if (s[i] == v)
6             return &s[i];
7     return 0;
8 }
```

Wskazówka: Ile razy procedura «my_index» wywoła funkcję «my_strlen»?

Zadanie 4. Poniżej podano kod procedury «strchr» z biblioteki standardowej. Przetłumacz go na **kod trójkowy** (ang. *three-address code*), podziel na **bloki podstawowe** (ang. *basic block*), a następnie narysuj **graf przepływu sterowania** (ang. *control flow graph*).

```

1 char *strchr(const char *cp, int ch) {
2     char *p = (char *)cp;
3     for (char *save = 0;; ++p) {
4         if (*p == ch)
5             save = p;
6         if (*p == '\0')
7             return save;
8     }
9 }

```

Zapisz powyższą procedurę do pliku «strchr.c». Następnie skompiluj go z opcją «-fdump-tree-all» i porównaj swoje rozwiązanie z plikiem o sufiksie «cfg» (np. «strchr.c.011t.cfg»).

Zadanie 5. Skompiluj poniższy kod z opcją «-O0» i uruchom otrzymany program. Następnie ponownie skompiluj go z opcją «-Os» i przeanalizuj otrzymany kod w assemblerze. Wyłumacz dlaczego kompilator aż tak drastycznie skrócił kod.

```

1 #include <stdio.h>
2
3 static int magic(int y) {
4     int sum = 0, x = 1;
5     while (x > 0) {
6         sum += x ^ y;
7         y *= 13;
8         x += x / 2 + 1;
9     }
10    return sum * 42;
11 }
12
13 int main() {
14     printf("%d\n", magic(33));
15     return 0;
16 }

```

Wskazówka: Znajdź co standard języka C definiuje dla przepętlenia podczas dodawania liczb całkowitych ze znakiem. Co dzięki temu może założyć kompilator o dodawaniu dwóch liczb dodatnich?