Switch- Case – Break

Analiza kodu asemblera automatycznie generowanego i optymalizowanego przez kompilator.

Zadania - plik `swcs3.c`

1. Zwiększać stopniowo liczbę przypadków w bloku `switch`, za każdym razem obserwując strukturę generowanego kodu asemblera.

Uwaga: w s*witchu* nie powinno być zbyt dużych „dziur”, tj maksymalny skok między wartościami sąsiednich przypadków nie powinien przekraczać np. 2 – 4.

Kod asemblera (plik `*.s`) można otrzymać kompilując plik `*.c` z parametrem `-S`

`gcc -S plik.c`

lub wyświetlić zawartość pliku wykonywalnego narzędziem `objdump`:

`objdump -D plik_wykonywalny`

Kiedy generowany jest łańcuch porównań, a kiedy używane zostaje tablicowanie?

2. Stworzyć s*witcha* z grupą kilkunastu przypadków z wartościami „leżącymi blisko siebie” (maksymalny skok np. 2 w zakresie 1 – 16) oraz kilkoma przypadkami o wartościach znacznie odstających od wspomnianej grupy (np. 100, 1000, 1000000).

Przeanalizować wygenerowany kod. Jak kompilator sobie poradził w tym przypadku?

3. Zbudować s*witcha*, w którym część przypadków będzie wywoływała funkcję `printf` ze stałym ciągiem znakowym, np.

```
case 2:  printf("2\n");break;
```

natomiast inne będą drukowały bezpośrednio wartość zmiennej „`a`”, sterującej *switchem*:

```
case 15: printf("%u\n",a);break;
```

Program należy skompilować z różnymi opcjami optymalizacji automatycznej (`-O0`, `-O3`) oraz z parametrem `-ffreestanding`.

Sprawdzić jakie funkcje biblioteczne są naprawdę wywoływane w utworzonym pliku wykonywalnym, w porównaniu do funkcji zawartych w kodzie źródłowym napisanym w języku C.