Optymalizacja

- Stosowanie algorytmów optymalizowania (poprawiania) kodu
  - Niezmienność programu wynikowego
  - Przyspieszenie w średnim przypadku
  - Koszty implementacji a przyspieszenie (ewentualne inne zyski)
- Grupy algorytmów optymalizacyjnych
  - **Optymalizacja przez szparkę** (peephole) - może być wieloprzebiegowa
    - niepotrzebne operacje zapisu i odczytu
    - usuwanie nieosiągalnego kodu (np. usuwanie następników skoków bez etykiet)
    - optymalizacje przepływu sterowania (np. skoki do skoków)
    - uproszczenia algebraiczne
    - osłabienie mocy rozkazów (dodawanie zamiast mnożenia)
    - używanie rozkazów specyficznych dla architektury
  - **Optymalizacje bloków bazowych**
  - **Optymalizacje pętli**
  - **Inne**

Bloki bazowe

- Usuwanie martwego kodu z użyciem grafów DAG

a := b + c
b := b + 1
d := b - d
a := c - d
b := a + c

Każdy korzeń DAG, który na liście wierzchołków nie posiada żadnej "żywej" zmiennej - jest niepotrzebny

- Przekształcenia algebraiczne
  - tożsamości, redukcja mocy, składanie stałych (constant folding)
- Inne
  - propagacja kopii i stałych (ewentualne powstanie kodu martwego)

Zaawansowane kompilatory optymalizujące

Analiza przepływu sterowania (Control Flow) + **Analiza przepływu danych (Data Flow)** + **Przekształcenia kodu**

- Graf przepływu sterowania:
  - bloki bazowe, krawędzie
  - dominacja wierzchołka (*d* dominuje nad *n*): każda ścieżka z początkowego wierzchołka grafu idąca do *n* przechodzi przez *d*
  - drzewo dominatorów
  - krawędź zwrotna - koniec dominuje nad początkiem
  - pętla naturalna dla d -> n to wierzchołek d i wszystkie wierzchołki, z których można dojść do n bez przechodzenia przez d

Graf przepływu - przykład

Rozważmy kod:

Pytania:

- gdzie są pętle ?
- które z nich są wewnętrzne ?
  (pętla wewnętrzna jest całkowicie zawarta w innej)

(1)	L1:	x := a + b
		if x < 0 goto L2
(2)		z := c + d
(3)	L2:	y := a + b
(4)	L3:	a := c + d
		if a > 0 goto L4
(5)		c := b + a
		goto L5
(6)	L4:	x := x + 1
(7)	L5:	a := x - d
		if a < 4 goto L3
(8)		y := a + b
		if y < 0 goto L7
(9)		z := a + b
		goto L1
(10)	L7:	a := z + d
		goto L5

Optymalizacja pętli

- Przemieszczenie kodu - warunek pętli zależny od niezmiennego wyrażenia
- Eliminacja zmiennych indukcyjnych - wartość wyrażenia obliczana na podstawie zmiennej iterującej wewnątrz pętli (ewentualna redukcja mocy)
- Rozpraszanie pętli - np. `while ( i<100 ) { a[i]=0;b[i]=0; }`
- Scalanie pętli - operacja odwrotna do rozpraszania
- Zamiana typu pętli (while-do na do-while)
- Zamiana położenia pętli (wewnętrzna na zewnętrzną - tablice)
- Rozwijanie pętli - zmniejszenie ilości iteracji kosztem zwiększenia ilości instrukcji
- Dzielenie zbioru iteracji (splitting)
- Przesuwanie instrukcji warunkowych na zewnątrz pętli