

Systemy komputerowe

Lista zadań nr 5

Na ćwiczenia 2 kwietnia 2025

wersja wstępna

Każde zadanie warte jest 1 punkt.

Zadanie 1. Zdefiniuj analizę łańcuchów użycie-definicja (ang. *use-definition chains*). Wylicz wynik tej analizy dla programu z zadania 5 z listy 3. posługując się uprzednio wyliczonymi zbiorami $RD_o(l)$.

Wskazówka: Slajdy 38-42 z pliku slides2.pdf.

Zadanie 2. Zdefiniuj analizę dostępnych wyrażeń (ang. *available expressions analysis*). Dla programu z zadania 5 z listy 3 zdefiniuj równania opisujące tę analizę. Następnie używając algorytmu stałopunktowego, wylicz rozwiązanie tych równań.

Wskazówka: Slajdy 10-16 z pliku slides2.pdf.

Zadanie 3. Zdefiniuj wariant analizy dostępnych wyrażeń liczący wyrażenie dostępne w konkretnej zmiennej: nietrywialne (tzn. różne od zmiennej) wyrażenie a jest dostępne w zmiennej x w etykiecie l , jeśli

- zostało przypisane do tej zmiennej na wszystkich ścieżkach prowadzących do l , oraz
- zmienna x oraz wartości zmiennych występujących w a nie uległy zmianie od tego czasu.

Zdefiniuj dziedzinę zbiorów występujących w tej analizie, funkcje *kill* oraz *gen*. Podaj ogólną postać równań występujących w tej analizie, określ jej typ (may/must, backward/forward).

Zadanie 4. Podaj równania dla analizy z poprzedniego zadania oraz programu z listy 3. Rozwiąż otrzymany układ równań na zbiorach używając algorytmu stałopunktowego.

Zadanie 5. Przypomnij definicję zmiennej żywej z analizy zmiennych żywych. Rozważmy następujący program

$$[x := 1]^1; [x := x - 1]^2; [x := 2]^3$$

Zmienna x jest martwa (nie żywa) na wyjściu z etykiet 2 i 3. Natomiast x jest żywa na wyjściu z etykiety 1, mimo iż x jest użyta do obliczenia wartości zmiennej martwej. Powiemy, że zmienna jest *zemdłona* jeśli jest martwa lub jeśli jest używana wyłącznie do obliczenia wartości zmiennych zemdłonych. W przeciwnym przypadku zmienną nazwiemy *silnie żywą*. W powyższym przykładzie x jest zemdłona na wyjściu z każdej etykiety. Zdefiniuj analizę przepływu danych, która wykrywa zmienne silnie żywe. Tzn. podaj dziedzinę zbiorów, funkcje *kill* oraz *gen*, ogólną postać równań oraz typ analizy.

Zadanie 6. Przetłumacz następujące instrukcje pętli języka C na kod trójkowy. Postaraj się użyć jak najmniejszej liczby instrukcji skoku. Możesz użyć zmiennych (rejestrów) tymczasowych.

1. *while* (b) { ... }
2. *for* ($i = 0; i < n; i++$) { ... }
3. *do* { ... } *while* (b)

Zmienne i oraz b są całkowite.

Zadanie 7. Przetłumacz następujący program na kod trójkowy.

$$x = a*a*a + 4*a*a*b + 4*a*b*b + b*b*b$$

Występujące w nim zmienne są typu całkowitego i mają po 4 bajty. Zmienne a , b , c są niemodyfikowalne. Użyj jak najmniejszej liczby zmiennych (rejestrów) tymczasowych. Następnie załóż, że w dodatkowym rejestrze o nazwie *mem* zapamiętany jest adres początku tablicy bajtów, którą możesz wykorzystać do pamiętania tymczasowych wyników obliczeń. Wykonaj ponownie tłumaczenie minimalizując liczbę wykorzystanych rejestrów tymczasowych "przelewając" je (ang. *register spilling*) do pamięci.

Zadanie 8. W kodzie trójkowym zaimplementuj dowolny algorytm sortowania tablicy bajtów t o znanym rozmiarze n .

Zadanie 9. Pokaż ścieżkę przepływu danych w prostym jednocyklowym procesorze¹ wykonującym instrukcję $x = *(y + \text{imm})$, gdzie x i y są rejestrami, a imm stałą.

¹ w wariantcie widocznym na slajdzie 15 z pliku Single Cycle & Pipelined Architectures Intro. Uwaga: nie jest to jeszcze kompletny procesor.