



Metody kompilacji

Wykład 13, Generowanie kodu maszynowego II

Włodzimierz Bielecki, Piotr Błaszyński

Wydział Informatyki Zachodniopomorskiego Uniwersytetu Technologicznego

29 kwietnia 2021



Generowanie kodu

Metody
kompilacji

Generowanie
kodu

Deklaracje
Wyrażenia

Przetwarzanie
konstrukcji
sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice
jednowymiarowe
Tablice
dwuwymiarowe
Tablice
wielowymiarowe

Wywołanie
funkcji

Przykłady
generowania
kodu

Do wygenerowania kodu w asemblerze potrzebne są:

- deklaracje,
- wyrażenia,
- przepływ sterowania,
- wywołanie procedur.



Generowanie kodu - deklaracje

Metody kompilacji

Generowanie kodu

Deklaracje

Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe

Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Przetwarzanie deklaracji polega na sprawdzeniu przez generator kodu kilku informacji:

- czy zmienna lokalna czy globalna;
- jak przydzielić pamięć dla zmiennych;
- jakie podstawowe typy występują: *integer*, *boolean*, ...;
- jakie złożone typy występują: *records*, *arrays*,



Generowanie kodu - deklaracje

Metody kompilacji

Generowanie kodu

Deklaracje

Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe

Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Poniżej znajduje się fragment opisujący przydział pamięci; jest to kod wygenerowany na podstawie deklaracji.

```
. data
var_name1:          . word  0
var_name2:          . word 29,10
var_name3:          . space 40
var_name4:          . space 80
```

W powyższym kodzie instrukcje w liniach 2 i 3 oznaczają przydział 4 bajtów do każdego słowa; dodatkowo w linii 3 zaprezentowana została inicjalizacja wartością początkową, a w linii 4 i 5 przydzielone zostały większe obszary pamięci.



Generowanie kodu - deklaracje

Metody kompilacji

Generowanie kodu

Deklaracje

Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe

Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Podstawowe dyrektywy SPIM:

- *.data* – poprzedza dane,
- *.ascii "str"* – zapisuje *str* w pamięci bez znaku końca wiersza `\0`;
- *.asciiz "str"* – to samo jak wyżej, ale z `\0`;
- *.byte 3, 4, 16* – zapisuje 3 wartości; każda zajmuje jeden bajt;
- *.double 3.14, 2.72* – zapisuje 2 wartości zmiennoprzecinkowe z podwójną dokładnością,
- *.float 3.14, 2.72* – zapisuje 2 wartości zmiennoprzecinkowe,
- *.word 3, 4, 16* – zapisuje 3 wartości; każda zajmuje 32 bity;
- *.space 100* – rezerwuje 100 bajtów;
- *.text* – zaczyna segment tekstu z instrukcjami.



Generowanie kodu - wyrażenia

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Przetwarzanie wyrażeń polega na:

- generowaniu poprawnego kodu,
- kontroli typów,
- obliczaniu adresów elementów tablicy,
- obliczaniu wyrażeń warunkowych w konstrukcjach sterowania,
- generowaniu skoków w konstrukcjach sterowania.



Generowanie kodu - wyrażenia

Metody kompilacji

Generowanie kodu

Deklaracje Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Na rysunku przedstawione zostało drzewo parsowania i kolejność generowania kodu z poniższego fragmentu kodu obliczającego wyrażenie $a = b + c + d + e$. Napisy $t0$ i $t1$ oznaczają użyte do przechowania danej wartości zmienne tymczasowe, natomiast liczby w okręgach oznaczają numer linii z poniższego kodu:

```
lw t0 , b
lw t1 , c
add $t0 , $t0 , $t1
sw $t0 , tmp1
lw $t0 , tmp1
lw t1 , d
add $t0 , $t0 , $t1
lw $t1 , e
add $t0 , $t0 , $t1
sw $t0 , a
```



Generowanie kodu - wyrażenia

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Instrukcje z linii 4 i 5:

```
sw tmp1, $t0  
lw $t0, tmp1
```

Są redundatne, a więc nie ma konieczności ich zapisywania.



Generowanie kodu - wyrażenia

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

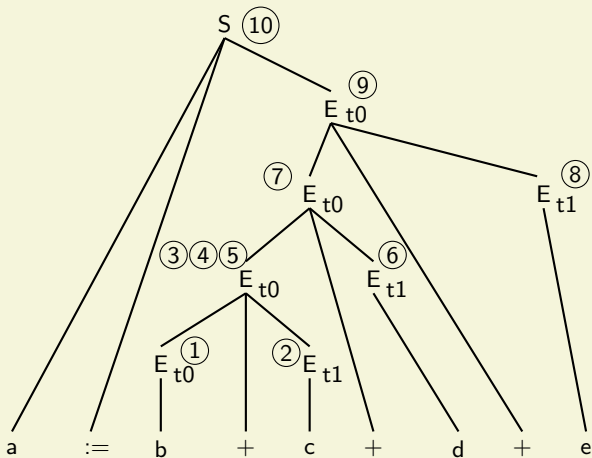
Wprowadzenie instrukcji warunkowej
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu





Generowanie kodu - akcje semantyczne

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie instrukcji warunkowej
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

$S \rightarrow id := E$

```
printf("sw_␣$t%d,%s\n", $3.reg, $1);  
free_reg($3.reg); //zwalnia rejestr
```

$E \rightarrow E + E$

```
$$ .reg = $1.reg;  
printf("add_␣$t%d,␣$t%d,␣$t%d\n",  
    $$ .reg, $1.reg, $3.reg, $3.reg);  
free_reg($3.reg); //zwalnia rejestr
```

$E \rightarrow id$

```
$$ .reg = get_register();  
printf("lw_␣$t%d,%s\n", $$ .reg, $1);
```



Generowanie kodu - akcje semantyczne

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

W akcjach semantycznych z poprzedniego slajdu sw oznacza zapis wartości przechowywanej w rejestrze $\$3.reg$ w pamięci pod adresem $\$1$; symbol $\$3$ odwołuje się do wartości skojarzonej z trzecim symbolem gramatyki po prawej stronie. W drugiej akcji semantycznej symbol $\$1$ oznacza rejestr, który przechowuje wartość pierwszej nazwy, a symbol $\$3$ oznacza rejestr, który przechowuje wartość drugiej nazwy. Symbol $\$\$$ odwołuje się do wartości atrybutu skojarzonej z nieterminalem po lewej stronie. Instrukcja lw przenosi wartość do rejestru; funkcja $get_register$ zwraca wolny rejestr. W trzeciej akcji semantycznej $\$1$ oznacza adres, pod którym w pamięci jest przechowywana wartość zmiennej (id).



Generowanie kodu - konstrukcje sterowania

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Konstrukcje sterowania występujące w językach programowania:

- *if*,
- *while*,
- *repeat*,
- *for*,
- *case*.

Generacja etykiet – wszystkie etykiety muszą być unikatowe.



Generowanie kodu - instrukcje warunkowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe

Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Na rysunku oraz w poniższych fragmentach kodu przedstawiono schemat generowania kodu i przepływ instrukcji dla prostej instrukcji warunkowej.

```
if (y > 0) then begin
... body ...
end
```



Generowanie kodu - instrukcje warunkowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe

Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

```
lw $t0,y  
li $t1,0  
sgt $t2,$t0,$t1 # = 1 if true  
beqz $t2,L2
```

... body ...

L2:



Generowanie kodu - instrukcje warunkowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

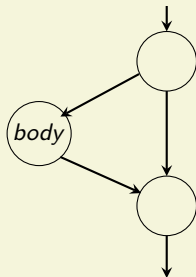
Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu





Generowanie kodu - instrukcje warunkowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe

Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Na rysunku oraz w poniższych fragmentach kodu przedstawiono schemat generowania kodu i przepływ instrukcji dla instrukcji warunkowej z *else*.

```
if (y > 0) then begin
... body_1 ...
end else
... body_2 ...
end
```




Generowanie kodu - instrukcje warunkowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

```
lw $t0,y
li $t1,0
sgt $t2,$t0,$t1 # = 1 if true
beqz $t2,L2
... body_1 ..
b L3
L2:
... body_2 ...
L3:
```

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe

Pętle

Tablice

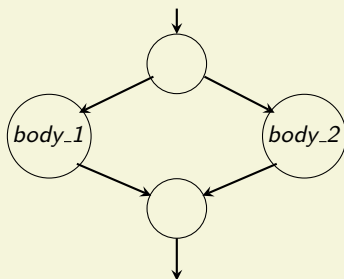
Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu



Rysunek: Przepływ danych dla instrukcji warunkowej z *else*



Generowanie kodu - pętle

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe

Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Na rysunku oraz w poniższych fragmentach kodu przedstawiono schemat generowania kodu i przepływ instrukcji dla pętli.

```
while x < 100 do  
  
    ... body ...  
  
end
```

```
L25:    lw $t0, x  
        li $t1, 100  
        sgt $t2, $t0, $t1  
        beqz $t2, L26  
        ... body ...  
        b L25  
L26:
```

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe

Pętle

Tablice

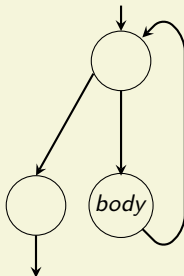
Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu



Rysunek: Przepływ danych dla instrukcji pętli



Generowanie kodu

Metody kompilacji

Generowanie kodu

Deklaracje

Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe

Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Schemat ogólny dla instrukcji warunkowych:

```
if_stmt -> IF expr THEN
    kod do obliczenia expr ($2),
    utwórz dwie nowe etykiety: L1, L2,
    jeśli expr=false ($2=false), to skok do L1,
    ciało if_stmt
skok do L2
    ELSE
L1: ciało else_stmt }
    ENDIF
L2: ...
```



Generowanie kodu

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Schemat ogólny dla pętli:

```
for_stmt -> FOR id = start TO stop
{ kod do obliczenia start ($1) i stop ($2),
  utwórz 2 etykiety L1, L2,
  utwórz kod dla instrukcji id = start,
  L1: kod do porównania id i stop ($3),
  jeśli ($3)=false, to skok do L2,
  kod dla ciała pętli,
  kod dla inkrementacji id,
  skok do L1
END FOR
L2:...
```



Generowanie kodu - tablice

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Obliczenie adresów elementów tablic. Załóżmy, że b oznacza adres bazowy, od którego zaczyna się obszar pamięci, zarezerwowany do przechowywania elementów tablicy $a[l..h]$; każdy element zajmuje s bajtów. Liczba elementów może być obliczona za pomocą wzoru: $e = h - l + 1$. Rozmiar tablicy: $e * s$. Adres elementu to $a[i]$, przy założeniu, że obszar zaczyna się od adresu $b, l \leq i \leq h: b + (i - l) * s$. Rozmieszczenie pamięci w tablicy jest przedstawione na kolejnym slajdzie, razem z postacią uwzględniającą adresy bezwzględne.



Generowanie kodu - tablice

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Tabela: Elementy tablicy jednowymiarowej

a[l]	a[l+1]	a[l+1]	...	a[h]
b				

Tabela: Elementy tablicy jednowymiarowej - adresy bezwzględne

a[3]	a[4]	a[5]		a[100]
100	104			



Generowanie kodu - tablice

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Przykład: $a[3\dots 100]$; każdy element tablicy jest reprezentowany przez 4 bajty. Liczba elementów: $100 - 3 + 1 = 98$. Rozmiar tablicy: $98 * 4 = 392$. Adres elementu to $a[50]$, przy założeniu, że obszar zaczyna się od adresu 100: $100 + (50 - 3) * 4 = 288$.



Generowanie kodu - tablice

Metody kompilacji

Generowanie kodu

Deklaracje

Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe

Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Kod w języku C z odwołaniem się do tablicy:

```
A[8] = h + A[8];
```

Odpowiadający powyższemu fragmentowi wynikowy kod MIPS, przy spełnieniu następujących założeń:

- 1) \$s3 zawiera adres pierwszego elementu A (adres bazowy b),
- 2) \$s2 zawiera wartość h,

ma postać:

```
lw      $t0,32($s3)      # $t0 gets A[8]
# i-1 =8, s=4, (i-1)x s =8 x 4 =32
add     $t0,$s2,$t0      # add h
sw      $t0,32($s3)      # store value back in A[8]
```



Generowanie kodu - tablice

Metody
kompilacji

Generowanie
kodu

Deklaracje
Wyrażenia

Przetwarzanie
konstrukcji
sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice
jednowymiarowe

Tablice
dwuwymiarowe

Tablice
wielowymiarowe

Wywołanie
funkcji

Przykłady
generowania
kodu

Kod w języku C z odwołaniem się do tablicy przez wartość zmiennej:

```
g = h + A [ i ];
```

Jeżeli spełnione jest założenie, że \$s4 zawiera i , to zostanie wygenerowany poniższy wynikowy kod MIPS, odpowiadający fragmentowi kodu z odwołaniem do tablicy poprzez wartość zmiennej:

```
# zapisz wartosc w \t1
add    $t1, $s4, $s4    # $t1 = 2 * i
add    $t1, $t1, $t1    # $t1 = 4 * i
# Baza jest przechowywana w $s3
# Adres A[i]
add    $t1, $t1, $s3    # $t1 = Adres(A[i])
lw     $t0, 0($t1)     # $t0 = A[i]
# dodaj A[i] do h
add    $s1, $s2, $t0    # $s1 = h + A[i]
```



Generowanie kodu - tablice dwuwymiarowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Zapis wierszami i kolumnami dla deklaracji $a[4..6, 3..4]$ pokazano w tabeli.

Adres	Wiersz	Kolumna
$b + 0s$	$a[4,3]$	$a[4,3]$
$b + 1s$	$a[4,4]$	$a[5,3]$
$b + 2s$	$a[5,3]$	$a[6,3]$
$b + 3s$	$a[5,4]$	$a[4,4]$
$b + 4s$	$a[6,3]$	$a[5,4]$
$b + 5s$	$a[6,4]$	$a[6,4]$



Generowanie kodu - tablice dwuwymiarowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Dla tablicy zadeklarowanej jako $A[4..7,3..4]$ otrzymujemy pokazany w tabeli zapis wierszami.

Adres	Wiersz
$b + 0s$	$a[4,3]$
$b + 1s$	$a[4,4]$
$b + 2s$	$a[5,3]$
$b + 3s$	$a[5,4]$
$b + 4s$	$a[6,3]$
$b + 5s$	$a[6,4]$
$b + 6s$	$a[7,3]$
$b + 7s$	$a[7,4]$



Generowanie kodu - tablice dwuwymiarowe

Metody kompilacji

Generowanie kodu

Deklaracje

Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe

Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

$a[l_1..h_1, l_2..h_2]$; każdy element jest reprezentowany przez s bajtów.

Liczba elementów: $e = e_1 * e_2$, gdzie $e_1 = (h_1 - l_1 + 1)$ i $e_2 = (h_2 - l_2 + 1)$.

Rozmiar tablicy: $e * s$.

Rozmiar każdego wymiaru: $d_1 = e_2 * d_2$, $e_2 = (h_2 - l_2 + 1)$
 $d_2 = s$.

Adres elementu $a[i, j]$ z bazą b , $l_1 \leq i \leq h_1$,
 $l_2 \leq j \leq h_2$: $b + (i - l_1) * d_1 + (j - l_2) * s$ określa liczbę słów w jednym wierszu.

d_1 określa liczbę bajtów, które zajmuje jeden wiersz.

Iloczyn $(i - l_1) * d_1$ określa liczbę bajtów zajmowanych przez $(i - l_1)$ wierszy.

Iloczyn $(j - l_2) * s$ określa liczbę bajtów, które zajmują $(j - l_2)$ słów.



Generowanie kodu - tablice dwuwymiarowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Przykład:

$A[3..100, 4..50]$; każdy element jest reprezentowany przez 4 bajty.

$98 * 47 = 4606$ elementów,

$4606 * 4 = 18424$ bajtów,

$d_2 = 4, d_1 = 47 * 4 = 188$.

Dla $b = 100$; adres $a[5, 5]$:

$100 + (5 - 3) * 188 + (5 - 4) * 4 = 720$



Generowanie kodu - tablice dwuwymiarowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

`a[3,5]` : zapis wierszami; przydział pamięci:

```
.data  
a:      .space 60      # 3x5=15 word-size elements * 4
```




Generowanie kodu - tablice dwuwymiarowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe

Pętle

Tablice

Tablice jednowymiarowe

Tablice dwuwymiarowe

Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Obliczanie adresu:

$$\text{Adres} = b + (i - l_1) * d_1 + (j - l_2) * s$$

$$d_1 = e_2 * d_2 = 5 * 4 = 20$$

$$e_2 = (h_2 - l_2 + 1) = 5$$

$$d_2 = s = 4$$

```
la $t0, a           # baza b w $t0
lw $t1, x           # x w $t1
mul $t1, $t1, 20    # (x - l1) * d1, d1=20
add $t0, $t0, $t1   # b + (x - l1) * d1 w $t0
lw $t1, y           # y w $t1
mul $t1, $t1, 4     # (j - l2) * s, s=4
add $t0, $t0, $t1   # Adres dla a[x,y]: b + (i - l1) *
                    # d1 + (j - l2) * s
lw $t1, ($t0)       # t1 zawiera a[x,y]
```



Generowanie kodu - tablice wielowymiarowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

$a[4..7, 3..4, 8..9]$

Rozmiar trzeciego wymiaru = s

Rozmiar drugiego wymiaru = $s * 2$

Rozmiar pierwszego wymiaru = $s * 2 * 2$



Generowanie kodu - Elementy tablicy trójwymiarowej

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Adres	Odwołanie
$b + 0s$	$a[4,3,8]$
$b + 1s$	$a[4,3,9]$
$b + 2s$	$a[4,4,8]$
$b + 3s$	$a[4,4,9]$
$b + 4s$	$a[5,3,8]$
$b + 5s$	$a[5,3,9]$
$b + 6s$	$a[5,4,8]$
$b + 7s$	$a[5,4,9]$
$b + 8s$	$a[6,3,8]$
$b + 9s$	$a[6,3,9]$
$b + 10s$	$a[6,4,8]$
$b + 11s$	$a[6,4,9]$
$b + 12s$	$a[7,3,8]$
$b + 13s$	$a[7,3,9]$
$b + 14s$	$a[7,4,8]$
$b + 15s$	$a[7,4,9]$



Generowanie kodu - tablice wielowymiarowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

$a[l_1..h_1, l_2..h_2, l_3..h_3]$; każdy element zajmuje s bajtów.

Liczba elementów: $e = e_1 * e_2 * e_3$, gdzie: $e_i = (h_i - l_i + 1)$.

Rozmiar tablicy: $e * s$.

Rozmiar poszczególnych wymiarów: $d_1 = e_2 * d_2$, $d_2 = e_3 * d_3$,
 $d_3 = s$.

Adres elementu $a[i, j, k]$ z bazą b , $l_1 \leq i \leq h_1$ i

$l_2 \leq j \leq h_2$: $b + (i - l_1) * d_1 + (j - l_2) * d_2 + (k - l_3) * s$.

Przykład:

$A[3..100, 4..50, 1..4]$; każdy element zajmuje 4 bajty:

$98 * 47 * 4 = 18424$ elementów,

$18424 * 4 = 73696$ bajtów,

$d_3 = 4$, $d_2 = 4 * 4 = 16$, $d_1 = 16 * 47 = 752$.



Generowanie kodu - tablice wielowymiarowe

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Dla $b = 100$ adres elementu $a[5, 5, 2]$:

$$100 + (5 - 3) * 752 + (5 - 4) * 16 + (2 - 1) * 4 = 1624.$$



Generowanie kodu - funkcje

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

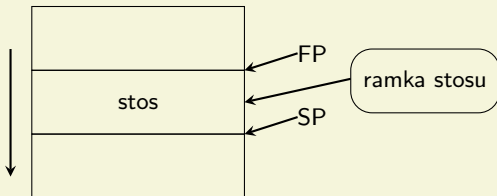
Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Założenie: jedna funkcja (wywołująca) wywołuje drugą funkcję (wywoływaną). Jakie są czynności realizowane przez pierwszą i drugą funkcję?

Wskaźniki SP i FP – dla każdej funkcji trzeba przydzielić ramkę stosu. FP wskazuje na początek bieżącej aktywacji (pierwsze słowo ramki stosu). SP wskazuje na ostatnie słowo ramki stosu. Stos rośnie w kierunku niższych adresów, dlatego chcąc zwiększyć stos, trzeba zastosować operację odejmowania (!). Na rysunku przedstawiona jest budowa stosu i umiejscowienie wskaźników SP i FP.





Generowanie kodu - funkcje

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Funkcja wywołująca zapisuje argumenty funkcji wywoływanej w standardowych miejscach i wykonuje następujące czynności:

- 1 Przekazuje argumenty. Zgodnie z konwencją; pierwsze cztery argumenty przekazywane są do rejestrów $\$a0 - \$a3$. Wszystkie pozostałe argumenty są odkładane na stos i pojawiają się na początku stosu.
- 2 Procedura wywoływana może korzystać z następujących rejestrów ($\$a0 - \$a3$ i $\$t0 - \$t9$). Jeśli funkcja wywołująca zamierza korzystać z tych rejestrów, to musi zapisać zawartość tych rejestrów w pamięci przed wywołaniem.
- 3 Wykonuje instrukcję *jal*, która przekazuje sterowanie do pierwszej instrukcji funkcji wywołanej i zapisuje adres powrotu w rejestrze $\$ra$.



Generowanie kodu - funkcje

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Przed wykonaniem obliczeń funkcja wywoływana musi wykonać następujące kroki:

- 1 Przydzielić obszar pamięci (ramkę) poprzez odjęcie wielkości ramki od wskaźnika stosu **FS** (stos zaczyna się od większych adresów).
- 2 Zapisać zawartość rejestrów funkcji wywoływanej w przydzielonym obszarze pamięci (w ramce). Funkcja wywoływana musi zapisać w ramce dane przechowywane w rejestrach ($\$s0 - \$s7$, $\$fp$, $\$ra$) przed korzystaniem z tych rejestrów, ponieważ funkcja wywołująca spodziewa się korzystać z danych w tych rejestrach po zakończeniu wykonania funkcji wywoływanej. Zawartość rejestru $\$fp$ musi być zapisana w pamięci przez każdą procedurę, która przydziela obszar pamięci dla stosu. Natomiast zawartość rejestru $\$ra$ musi być zapisywana tylko wtedy, gdy funkcja wywoływana sama wywołuje inną funkcję.



Generowanie kodu - funkcje

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Przed wykonaniem obliczeń funkcja wywoływana musi wykonać następujące kroki, cd.:

- 1 Zawartość pozostałych rejestrów, z których korzysta funkcja wywoływana, musi być także zapisana.
- 2 Ustawić wskaźnik stosu, dodając rozmiar ramki minus 4 (co zwiększa stos o 4 bajty) do zawartości rejestru $\$sp$ i zapisać wynik w rejestrze $\$fp$.



Generowanie kodu - funkcje

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Zakończenie:

- 1 Jeśli funkcja wywoływana zwraca wartość, to zapisuje ją w rejestrze \$v0.
- 2 Funkcja wywoływana przywraca zawartość wszystkich rejestrów, które zostały zapisane w momencie wywołania procedury.
- 3 Funkcja wywoływana zdejmuje ramkę stosu, dodając rozmiar ramki do \$sp.
- 4 Następuje powrót do adresu podanego w rejestrze \$ra.



Generowanie kodu - funkcje

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Przykład w C:

```
int main ()
{
    x=addthem(a,b);
}
int addthem(int a, int b)
{
    return a+b;
}
```



Generowanie kodu - funkcje

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Przy wywołaniu procedur, w kodzie SPIM, funkcja *addthem* wymaga ramki w stosie (4 bajty) do zapisania wartości rejestru *t0*, którą należy przywrócić po zakończeniu obliczeń funkcji:

```
.text
main:  #założenia: a jest w $t0, b jest w $t1
    add $a0,$0,$t0    # Przekazanie argumentów
    add $a1,$0,$t1    # do rejestrów a0, a1
    jal addthem      # wywołanie funkcji addthem
    #Miejsce powrotu
    add $t3,$0,$v0    # gdy funkcja wywołana zwróci
    # wartość do $v0, jest ona przesłana do $t3
    syscall
```



Generowanie kodu - funkcje

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Przy wywołaniu procedur, w kodzie SPIM, funkcja *addthem* wymaga ramki w stosie (4 bajty) do zapisania wartości rejestru *t0*, którą należy przywrócić po zakończeniu obliczeń funkcji:

```
addthem :
    addi $sp, $sp, -4      # zarezerwowanie ramki stosu
    sw $t0, 0($sp)       # zapis poprzedniej wartosci (
                          $t0)
    add $t0, $a0, $a1     # instrukcja implementująca ciało
                          o funkcji
    add $v0, $0, $t0      # wynik
    lw $t0, 0($sp)       # ładowanie poprzedniej wartości
    addi $sp, $sp, 4     # Zdejmij ramkę ze stosu
    jr $ra               # powrót
```



Generowanie kodu z użyciem AST

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

(*Abstract Syntax Tree* i **CFG** (*Control Flow Graph* zostały zaprezentowane na rysunkach.)

W kodzie MIPS w komentarzach umieszczone są numery linii ze źródłowego kodu w C, z których wyprowadzono daną linię kodu asemblera. Linia 4 ma 3 istotne elementy: a) inicjalizację licznika pętli, b) sprawdzenie warunku, c) inkrementację.

Identyczne oznaczenia linii kodu znajdują się w okręgach na rysunkach.

```
int popcount(int i){
    int c=0;
    int j;
    for (j=0 ; j<32 ;j++){
        if (i & (1 << j))
            c++;
    }
    return c
}
```



Generowanie kodu z użyciem AST

Metody kompilacji

Generowanie kodu

Deklaracje

Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie

Instrukcje warunkowe

Pętle

Tablice

Tablice

jednowymiarowe

Tablice

dwuwymiarowe

Tablice

wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

```
popcount :
    ori $v0, $zero, 0                #Linia 2
    ori $t1, $zero, 0                #4a
top:    slti $t2, $t1, 32             #4b
        beq $t2, $zero, end          #4b
        nop
        addi $t3, $zero, 1           #5
        sllv $t3, $t3, $t1           #5
        and $t3, $a0, $t3            #5
        beq $t3, $zero, notone       #5
        nop
        addi $v0, $v0, 1             #6
notone: beq $zero, $zero, top
        addi $t1, $t1, 1             #4c
end:    jr $ra                       #8
        nop
```

Nowe instrukcje w powyższym kodzie: *ori* – operator OR, *nop* – pusty operator, *sllv* – *shift left logical variable*.



Generowanie kodu - Drzewo składniowe (AST) dla kodu funkcji *popcount*

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

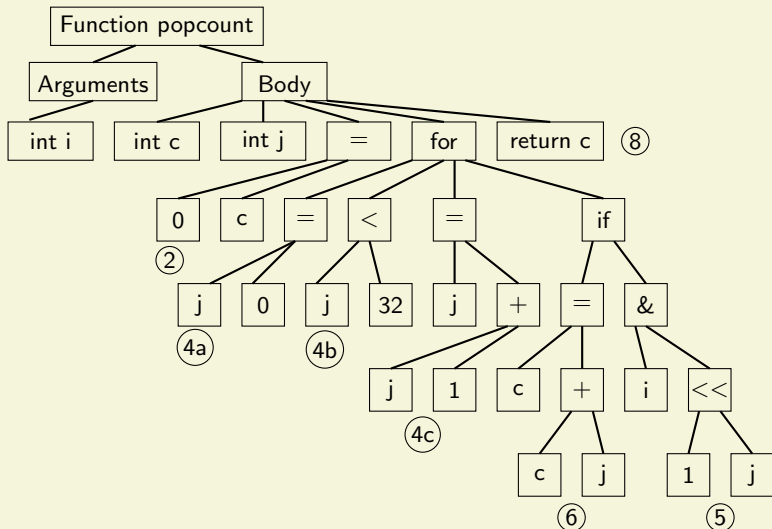
Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu





Generowanie kodu - CFG (*control flow graph*) dla kodu funkcji *popcount*

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

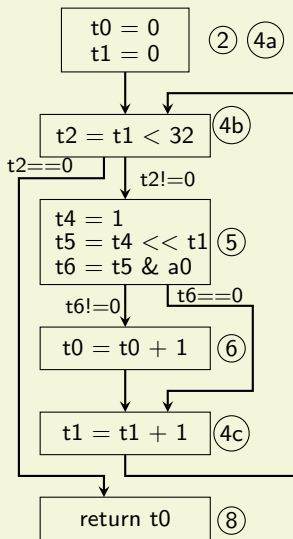
Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu





Generowanie kodu

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

Kolejne dwa programy przedstawiają przykłady kodów w asemblerze. Dodawanie dwóch liczb:

```
# $t2 - used to hold the sum of the $t0 and $t1.
# $v0 - syscall number, and syscall return value.
# $a0 - syscall input parameter.
.text # Code area starts here
main: li $v0, 5 # read number into $v0
      syscall # make the syscall read_int
      move $t0, $v0 # move the number read into $t0
      li $v0, 5 # read second number into $v0
      syscall # make the syscall read_int
      move $t1, $v0 # move the number read into $t1
      add $t2, $t0, $t1
      move $a0, $t2 # number to print into $a0
      li $v0, 1 # load syscall print_int into $v0
      syscall
      li $v0, 10 # syscall code 10 is for exit
      syscall #
# end of main
```



Generowanie kodu – Dodawanie N liczb

Metody
kompilacji

Generowanie
kodu

Deklaracje
Wyrażenia

Przetwarzanie
konstrukcji
sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice
jednowymiarowe
Tablice
dwuwymiarowe
Tablice
wielowymiarowe

Wywołanie
funkcji

Przykłady
generowania
kodu

```
# Input: number of inputs, n, and n integers;
# Output: Sum of integers

.data # Data memory area.
prmp1: .asciiz "How many inputs?"
prmp2: .asciiz "Next input:"
sumtext: .asciiz "The sum is"

.text # Code area starts here
main: li $v0, 4 # Syscall to print prompt string
      la $a0, prmp1 # li and la are pseudo instr.
      syscall
      li $v0, 5 # Syscall to read an integer
      syscall
      move $t0, $v0 # n stored in $t0
```



Generowanie kodu – Dodawanie N liczb

Metody kompilacji

Generowanie kodu

Deklaracje
Wyrażenia

Przetwarzanie konstrukcji sterowania

Wprowadzenie
Instrukcje warunkowe
Pętle

Tablice

Tablice jednowymiarowe
Tablice dwuwymiarowe
Tablice wielowymiarowe

Wywołanie funkcji

Przykłady generowania kodu

```
li    $t1, 0    # sum will be stored in $t1
while: blez $t0, endwhile    # (pseudo instruction)
li    $v0, 4    # syscall to print string
la    $a0, prmpt2
syscall
li    $v0, 5
syscall
add   $t1, $t1, $v0    # Increase sum by new input
sub   $t0, $t0, 1      # Decrement n
j     while
endwhile: li    $v0, 4    # syscall to print string
la    $a0, sumtext
syscall
move  $a0, $t1    # Syscall to print an integer
li    $v0, 1
syscall
li    $v0, 10    # Syscall to exit
syscall
```