Tentamen: Compilerconstructie

29 januari 2002, 14.00 – 17.00u.

Dit tentamen bestaat uit 4 opgaven, waarmee in totaal 100 punten te behalen zijn. Iedere opgave valt uiteen in verschillende onderdelen. Het maximaal aantal te behalen punten per onderdeel staat vermeld tussen [..].
Het is niet toegestaan andere informatie te raadplegen dan eigen kennis en hetgeen eventueel tijdens het tentamen door de surveillanten verteld of toegelicht wordt.
Tenslotte: vergeet niet naam te vermelden, BEARGUMENTEER ALLE ANTWOORDEN (losse getallen, ja/nee, en dergelijke worden NIET gehonoreerd!!!), en schrijf duidelijk. Sukses!

Opgave 1 Parsing

Gegeven de volgende grammatica

$$S \rightarrow S \,-\, A \mid A$$

$$A \rightarrow A\, B \mid B$$

$$B \rightarrow B + \mid p \mid q$$

a. Wat is het verschil tussen 'top-down' en 'bottom-up' parsing? [**5**]

b. Construeer de SLR parsing tabel voor de gegeven grammatica. [**10**]

c. Construeer eveneens de LALR parsing tabel. [**10**]

Opgave 2 Code generation

Gegeven het volgende Pascal-achtige programma

```
BEGIN
   i := 1;
   WHILE i <= 100 DO
   BEGIN
      IF A[i] = 0 THEN GOTO found
      ELSE i := i + 1
   END;
found:
END.
```

a. Geef de definitie van een 'basic block'. [**5**]

b. Geef de vertaling van bovenstaand programma in three-address code, geef de basic blocks en geef de control flow graph. Neem aan dat de elementen van A 1 byte groot zijn. [**10**].

c. Wat is een 'natural loop'? Geef een algoritme om in een control flow graph de natural loops te vinden. [**10**].

Opgave 3 Code optimization

Gegeven de volgende three-address code

```
 (1)   i := m-1
 (2)   j := n
 (3) t1 := 4*n
 (4)   v := a[t1]
 (5)   i := i+1
 (6) t2 := 4*i
 (7) t3 := a[t2]
 (8) if t3 < v goto (5)
 (9)   j := j-1
(10) t4 := 4*j
(11) t5 := a[t4]
(12) if t5 > v goto (9)
(13) if i >= j goto (23)
(14) t6 := 4*i
(15)  x := a[t6]
(16)      t7 := 4*i
(17)      t8 := 4*j
(18)      t9 := a[t8]
(19)   a[t7] := t9
(20)     t10 := 4*j
(21)  a[t10] := x
(22)      goto (5)
(23)     t11 := 4*i
(24)       x := a[t11]
(25)     t12 := 4*i
(26)     t13 := 4*n
(27)     t14 := a[t13]
(28)  a[t12] := t14
(29)     t15 := 4*n
(30)  a[t15] := x
```

a. Geef de flow graph met bijbehorende basic blocks voor deze three-address code. [**5**]

b. Geef de flow graph na 'common subexpression elimination'. [**5**]

c. Hoe ziet de flow graph eruit na 'copy propagation' en 'dead-code elimination'? [**5**]

d. Welke andere optimalisaties zijn nog mogelijk? Geef de uiteindelijke flow graph. [**5**]

e. Kwantificeer de uiteindelijke winst na alle optimalisaties. [**5**]

Opgave 4 Varia

a. Wanneer is 'left recursion' van een grammatica ongewenst? Wanneer en hoe kun je het elimineren? [**5**]

b. Wat betekent het als een grammatica LL(k) is? [**5**]

c. Wat verstaan we onder 'syntax directed translation'? [**5**]

d. Leg uit wat 'unification' van twee expressies betekent. [**5**]

e. Verklaar de begrippen 'scope', 'activation', 'binding', en 'lifetime' in de context van run-time environments. [**5**]

# # # # #