Compiladors: Examen de teoria

12 de juny de 2017

1 Anàlisi sintàctica (2.5 punts)

Donada la gramàtica següent:

$$
\begin{array}{lrcl}
(1) & S' & \rightarrow & S\$ \\
(2) & S & \rightarrow & AAS \\
(3) & S & \rightarrow & \varepsilon \\
(4) & A & \rightarrow & aA \\
(5) & A & \rightarrow & b
\end{array}
$$

- Descriure el llenguatge que genera [0.5 punts].
- Calcular *First* i *Follow* pels símbols S i A. Afegiu ε en el *First* d'aquells símbols que siguin anul.lables [0.5 punts].
- Generar l'autòmat i la taula SLR(1) [1.5 punts].

2 Gramàtiques d'atributs (2.5 punts)

Cal dissenyar una gramàtica per definir tipus de dades en C on hi només hi pot haver tipus bàsics, estructures (`struct`) o unions (`union`). Els tipus bàsics poden ser `char` (1 byte), `int` (4 bytes), `double` (8 bytes) i punters als tipus bàsics (4 bytes). Exemples de `struct` i `union`:

```
union { int i; struct {double a; int* b} s; } u;
struct { struct { int i, j; double x; } s; char * p; } z;
struct { double * q; union {char c; int* k; } u; } s;
```

La mida d'una estructura és la suma de la mida dels seus components. La mida d'una unió és el màxim de la mida dels seus components.

- Definiu una gramàtica d'atributs per tal de calcular la mida dels tipus de dades. Descriviu el significat dels atributs que es facin servir. Podeu suposar que estan definits els tokens per les paraules claus (`STRUCT`, `UNION`, `CHAR`, `INT`, `DOUBLE`) i pels identificadors (`IDENT`). Per la resta de tokens podeu fer servir la nomenclatura `'{'`, `'}'`, `';'`, etc. Es valorarà especialment la senzillesa i legibilitat de la gramàtica [2 punts].
- Dibuixeu un arbre abstracte de sintàxi de l'estructura següent, indicant el valor dels atributs a cada node de l'arbre [0.5 punts].

```
struct {
  int i;
  union {
    int j;
    struct {double *z; char c;} s;
    double x;
  } u;
  struct {char* p; int n;} t;
} y;
```

3 Generació de codi (2.5 punts)

Sigui el codi següent:

```
struct {int a[10]; int b;} x[100];

r = n > max or n < 0 ? k : 2*n + (i < n ? i + 1 : x[k].b);
```

- Dibuxeu un diagrama genèric de flux (caixes i etiquetes) de l'estructura de codi per avaluar una expressió condicional del tipus `c?e1:e2`. Podeu fer servir la notació `t=e1` per indicar que el valor de l'expressió `e1` serà assignat a la variable `t` [0.5 punts].
- Dibuixeu l'AST complet de la instrucció d'assignació. Per a representar l'expressió condicional (`c?e1:e2`), feu servir un arbre amb arrel `'?'` i tres fills (`c`, `e1` i `e2`) [0.5 punts].
- Escriviu codi de tres adreces suposant que l'avaluació de expressions booleanes és fa segons la semàntica del llenguatge C++ i utilitzant la tècnica de *backpatching*. Suposeu que totes les variables de l'enunciat són de tipus `int` i que la mida d'un `int` és de 4 bytes. Optimitzeu les instruccions de salt tant com sigui possible. Només cal que doneu el codi final després d'optimitzar [1.5 punts].

4 Optimització de codi (2.5 punts)

Suposeu que el codi següent correspon al cos d'una funció declarada com a

```
int f(int b);
```

on `b` és un paràmetre i la resta de variables són locals i enteres.

```
    a = 0
    x = a + b
L1: if a != 0 goto L2
    x = x - 1
    z = a * b
    y = 6 * x
    goto L3
L2: b = 2 * b
    z = z + 1
    x = x + 1
L3: if x > a goto L1
    a = a + z
    a = a + y
    return a
```

- Dibuixeu el graf de blocs bàsics indicant les instruccions que van dins de cada bloc [0.5 punts].
- Dibuixeu l'arbre de blocs dominadors [0.5 punts].
- Indiqueu les variables vives després d'executar les instruccions `x=x-1`, `x=x+1` i `a=a+z` [0.5 punts].
- Apliqueu iterativament totes les optimitzacions fins que el codi no es pugui reduïr més. Expliqueu el raonament de cadascuna de les optimizacions. Escriviu el codi resultant després de cada optimització individual [1 punt].