COMPILATORI - ESERCIZI D'ESAME

Per ognuno dei seguenti assegnamenti in C,
- generare il codice a tre indirizzi assumendo che tutti gli elementi degli array siano interi di 4 byte ciascuno;
- convertire quindi il codice ottenuto in codice macchina seguendo il modello di macchina visto a lezione. Utilizzare il numero minore di registri possibile.

1) x = a/(b+c) - d*(e+f);
2) a[i][j] = b[i][k] + c[k][j];
3) *p++ = a[*q++];

---

1. Data la seguente grammatica aumentata con V_T={**x,i,u**,**p**}:

S' → S
S → S **x** A
S → A
A → E
A → **i u** E
E → **i**
E → E **p i**

i) costruire l'automa LR(0);
ii) costruire la tabella di parsing SLR;
iii) individuare eventuali conflitti

2. Effettuare la traduzione in codice intermedio di

```
        if(x==y)
      x=x+1;
   else y=x;
z=1;
```

3. Data la seguente grammatica con V_T={**a,e,b,c,d**}:

S → **a**S**e**
S → B
B → **b**B**e**
B → C
C → **c**C**e**
C → **d**

i) calcolare gli insiemi **FIRST** e **FOLLOW**;

ii) G è LL(1)? (Motivare la risposta).

---

1. Data la seguente grammatica aumentata con $\mathbf{V_T}$={**s**,**d**,**?**,**c**}:

S' → S
S → **s**L**d**
S → **c**
L → S
S → L**?**S

i) costruire l'automa LR(0);
ii) costruire la tabella di parsing SLR(0);
iii) usando le regole SLR(0), individuare eventuali conflitti **shift-reduce;**
iv) dopo aver risolto tutti gli eventuali conflitti **shift-reduce** scegliendo **shift**, mostrare tutti i passi del parser SLR(0) assumendo di avere in ingresso la stringa **sc?sc?cdd**

2. Effettuare la traduzione in codice intermedio di

x=2;**while**(x < 3 && 1 < 2) x=x+4;

(la traduzione deve essere effettuta usando le regole di traduzione illustrate a lezione).

Dopo aver prodotto l'albero annotato, si scriva il codice completo corrispondente alla traduzione dello statement.

3. Data la grammatica G:

S → **?**
S → ZFS
Z → F
Z → **>**
F → ε
F → **#**

i) calcolare gli insiemi **FIRST** e **FOLLOW**;
ii) G è LL(1)? Si risponda costruendo la tabella LL(1).

Costruire il DAG per le seguenti espressioni assumendo che + sia associativo a sinistra.

COMPILATORI - ESERCIZI D'ESAME

Identificare inoltre i `value numbers' delle sottoespressioni per l'allocazione dei nodi del DAG in un array.

i. a+b+(a+b)
ii. a+b+a+b
iii. a+a+(a+a+a+(a+a+a+a))

---

1. Data la seguente grammatica aumentata con V_T={**a, b, e,d**}:

S'	→	S
S	→	AaBB
A	→	Cd
A	→	Ab
C	→	e
B	→	aA
B	→	b

i) costruire l'automa LR(0);
ii) costruire la tabella di parsing SLR;
iii) individuare eventuali conflitti **shift-reduce;**

2. Effettuare la traduzione in codice intermedio di

while((z>x||z==x)&&(z<3||x>5)z=0;

Esibire l'albero di parsing annotato con gli attributi e la traduzione finale del codice.

3. Data la grammatica G con V_T= {**a,d,x,z,s**}

A → CxA
A → ε
B → zCu
B → sC
C → aBa
C → d

COMPILATORI - ESERCIZI D'ESAME

Scrivere la tabella di parsing LL(1) per G dopo aver calcolato gli insiemi FIRST e FOLLOW.
La grammatica è LL(1)? (**rispondere utilizzando la definizione di grammatica LL(1)**).

Data la grammatica G con V_T= {**a,b,c,d,f**}

E → TZ
Z → fTZ
Z → ε
T → FU
U → bFU
U → ε
F → d
F → aEc

i. Scrivere la tabella di parsing LL(1) per G dopo aver calcolato gli insiemi FIRST e FOLLOW.
ii.
La grammatica è LL(1)? (rispondere utilizzando la tabella di parsing).
iii.
Costruire per la stessa grammatica l'automa LR(0) e LR(1).
