Examen de Compilation

Licence Informatique 3

19 juin 2013

Le sujet est composé de 4 exercices indépendants. Aucun document n'est autorisé. Durée de l'épreuve : 0x78 minutes. La note finale tiendra trés largement compte de la présentation.

1 Automate

On considère l'alphabet ternaire $T := \{0, 1, 2\}$. Dans l'ensemble T^*, un mot $x_n x_{n-1} \dots x_1$ représente la *valeur* entière $\sum_{i=1}^{n} x_i 3^{i-1}$. Conformément à l'usage un *nombre* ternaire est un mot commençant par une lettre différente de 0, à l'exception du nombre zéro.

1. Quel est l'ensemble des mots T^*0 ?
2. Quel est l'ensemble des mots $0 + (1 + 2)T^*0$?
3. Donner un AFD pour le langage des valeurs multiples de 2.
4. Donner un AFD complet pour le langage des nombres multiples de 2.

2 Théorie des langages

Dans l'ensemble $\{a, b\}^*$, on considère le langage $X := \{a^n b^n \mid n \geq 0\}$.

1. Montrer que X est un langage algébrique.
2. Enoncer le lemme d'itération des langages réguliers.
3. Montrer que X n'est pas régulier.

3 flex-bison

On considère le langage des expressions régulières définies par la grammaire :

$$\begin{array}{lllllll} E & \longrightarrow & E + E & E & \longrightarrow & E.E \\ E & \longrightarrow & (E) & E & \longrightarrow & E^* \\ E & \longrightarrow & E\{\text{NB}\} & E & \longrightarrow & a|b|\dots|z \end{array}$$

où le symbole terminal NB représente un nombre en base 10. Il s'agit d'écrire une commande `reg.x` pour vérifier la syntaxe des expressions passées sur l'entrée standard.

1. Quels sont les symboles terminaux de cette grammaire ?
2. Quels sont les symboles non terminaux ?
3. Quel est l'axiome ?
4. La grammaire est ambigue, pourquoi ?

TAB. 1 – Jeu d'instruction d'une machine

0	READ x	mem[x] := lecture	1	WRITE x	écrire mem[x]
2	LOAD x	ACC := mem[x]	3	STO x	mem[x] := ACC
4	ADD x	ACC+= mem[x]	5	MUL x	ACC*=mem[x]
6	CONST x	ACC := x	7	STOP	arrêt
8	JMP x	PTR := x	9	ZERO x	si ACC=0 alors PTR=x

TAB. 2 – Un exemple d'exécutable.

0x04	0x00	0x00	0x00	0x10	0x20	0x12	0x24	0x35	0x51	0x09	0x00	...

5. Ecrire un analyseur lexical `reg.l`.
6. Ecrire un analyseur syntaxique `reg.y`.
7. Donner un `makefile` pour compiler une commande `reg.x`.

4 machine

Le jeu d'instruction d'une machine similaire à la machine du cours MILXI est partiellement décrit dans la table (1). La structure d'un exécutable se présente comme un tableau de 1024 mots de 16 bits. Une instruction est codée sur un mot, le code opération sur 4 bits, l'opérande sur 12 bits.

0	1		var		code
point d'entrée	variables initialisées	...	variables temporaires	...	instruction

Le point d'entrée du code est à l'adresse 0, la zone des données débute à l'adresse 1, suivi d'une zone de code. Les données variables et constante du programme sont placées avant les variables temporaires.

```
PTR := meme[0]
repeter
       tmp := mem[ PTR ]       ; PTR  := PTR + 1
       operation := tmp & 15 ; operande  := tmp >> 4
       execute( operation, operande )
jusqua operation  == STOP
```

1. Que représentent les symboles ACC et PTR ?
2. Deux zones mémoires standards ne sont pas utilisées par cette machine. Lesquelles ?
3. Commenter la boucle de la machine.
4. Exécuter le progamme (2).
5. Ecrire un simulateur `void run(short int prog[1024])` d'exécution du programme `prog`.
6. Donner un programme en assembleur qui lit un entier n, calcule la somme des carrés des nombres strictement inférieurs à n, et écrit le résultat sur la sortie standard.