Université d'Orléans **Maîtrise d'Informatique**

Feuille de Travaux Dirigés 9

La machine cible : machine à registres, adressable par octets avec des mots de 4 octets et n registres généraux $R_0,\ R_1, \ldots,\ R_{n-1}$. Elle possède des instructions à 2 adresses de la forme :

$$op \quad source,\ destination$$

dans laquelle *op* est un code opération, et *source* et *destination* sont des champs de données. Ex :

- MOV (charger *source* dans *destination*)
- ADD (ajouter *source* à *destination*)
- SUB (soustraire *source* à *destination*)

Les modes d'adressage : (contenu(a) dénote le contenu du registre ou de l'emplacement mémoire représenté par a).

Mode	Format	Adresse	Coût additionnel
absolu	M	M	1
registre	R	R	0
indexé	c(R)	c+contenu(R)	1
registre indirect	*R	contenu(R)	0
indirect indexé	*c(R)	contenu(c + contenu(R))	1

Le coût des instructions : c'est la somme des coûts des modes d'adressage(référencés par des coûts additionnels dans les tableaux) augmentée de 1. Ce coût correspond à la longueur (en mots) de l'instruction. Les modes d'adressage impliquant les registres ont un coût égal à 0, tandis que les coûts afférant à un emplacement en mémoire ou à un littéral ont un coût de 1, parce que de tels opérandes doivent être spécifiés dans un mot additionnel à l'instruction.

Exercice 1 Traduire l'expression arithmétique $a * -(b + c)$ en :
- un arbre abstrait ;
- du code à 3 adresses.
- du code à 2 adresses.

Exercice 2 Traduire l'expression $-(a + b) * (c + d) + (a + b + c)$ en :
- DAG.
- code à 2 adresses ;

Exercice 3 Traduire les instructions exécutables du programme C suivant :

```
main()
{
   int i;
   int a[10];
   i = 0;
   while (i < 10)
   {
```

```
        a[i] = 0;
        i = i + 1;
    }
}
```

en :
- un arbre abstrait ;
- du code à 2 adresses.

Exercice 4 Traduire l'instruction d'affectation :

$$A[i,j] := B[i,j] + C[A[k,l]] + D[i+j]$$

en code à 2 adresses pour les déclarations suivantes :
var $A,\ B\ :\ array[0..10, 1..5]$ of integer ;
$C,\ D\ :\ array[1..15]$ of integer ;

Exercice 5 En C, l'instruction *for* a la forme suivante :

$$for\ (e_1\,;\,e_2\,;\,e_3)\ ins$$

Elle a la même signification que :

```
e1;
while (e2) {
    ins;
    e3;
}
```

Construire une définition dirigée par la syntaxe pour traduire les instructions écrites à la manière de C en code à 3 adresses. On suppose qu'une instruction à 3 adresses peut être étiquetée symboliquement et que la fonction *NouvEtiq* retourne une nouvelle étiquette symbolique à chaque fois qu'elle est appelée.

Exercice 6 Proposer un algorithme qui détermine le nombre de registres nécessaires à la production de code à partir d'un arbre abstrait.

Déterminer le nombre de registres nécessaires pour les instructions :
- $x = a + b * c$
- $x = a/(b+c) - d * (e+f)$