Université des Sciences et de la Technologie HOUARI BOUMEDIENE
Faculté d'Electronique et d'Informatique
Département d'Informatique

Alger, Algérie, **16** Avril **2020**

Génération du Code Intermédiaire

1ère Partie

IL - IV

Génération du Code Intermédiaire

Structure de données

- La structure des quadruplets est une matrice $(n,4)$, où n est le nombre de quadruplets. La structure peut être représentée comme suit :

	OP	ARG1	ARG2	RES
1				
2				
	... ...			

- Où : **OP** est l'opérateur, **ARG1** et **ARG2** sont les opérandes, et **RES** est le résultat.

Génération du Code Intermédiaire

Structure de données

- L'expression $a = b - (c \div d) * e + f$ est traduite par les quadruplets suivants :

$x_1 \leftarrow c \div d$

$x_2 \leftarrow x_1 * e$

$x_3 \leftarrow b - x_2$

$x_4 \leftarrow x_3 + f$

$a \leftarrow x_4$

OP	ARG1	ARG2	RES
...			
/	c	d	x_1
*	x_1	e	x_2
-	b	x_2	x_3
+	x_3	f	x_4
=	x_4		a
...			

- Les variables x_1, x_2, x_3, x_4 sont des **variables temporaires**. Elles seront **insérées dans la table des symboles** lors de la génération des quadruplets.

Génération du Code Intermédiaire

Génération d'un quadruplet: GEN()

- L'action paramétrée suivante permet de générer un quadruplet :

```
void GEN (char *OP,  *ARG1, *ARG2, *RES)
{
        Quad[QuadSuiv][0] = OP;
        Quad[QuadSuiv][1] = ARG1;
        Quad[QuadSuiv][2] = ARG2;
        Quad[QuadSuiv][3] = RES;
}
```

Quad, est une variable globale de type matrice permettant de stocker les quadruplets.
QuadSuiv, est une variable globale représentant la ligne (adresse) du quadruplet suivant dans Quad.

- La déclaration de Quad et QuadSuiv se fait au niveau de la partie **définitions** du fichier .y comme suit:

```
%{
        #include<stdio.h>
        int Quad[100][4];
        int QuadSuiv = 0;          /* initialement, la matrice Quad est vide */
%}
...

%%
```

Génération du Code Intermédiaire

Création d'une variable temporaire: CreerTemp()

- L'action paramétrée suivante permet de créer une variable temporaire (un nom d'identificateur):

```
void CreerTemp (char *temp)
{
        Sprintf(temp, "x%d", indice);
}
```

Indice, est une variable globale initialisée à 1
QuadSuiv, est une variable globale représentant la ligne (adresse) du quadruplet suivant dans Quad

- La procédure CreerTemp permet de retourner séquentiellement les identificateurs suivants : x1, x2, …
- La déclaration de Indice se fait au niveau de la partie **définitions** du fichier .y comme suit:

```
%{
        #include<stdio.h>
        int Quad[100][4];
        int QuadSuiv = 0;          /* initialement, la matrice Quad est vide */
        int Indice = 1;
%}
…

%%
```

Génération du Code Intermédiaire

Traduction des expressions arithmétiques:

- Afin de générer les quadruplets, nous devons traduire l'expression arithmétique en plusieurs expressions contenant chacune un seul opérateur arithmétique. Le code de l'expression arithmétique est donc :

Expression Arithmétique	Traduction
$b - (c \div d) * e + f$	$x1 = c \div d$ $x2 = x1 * e$ $x3 = b - x2$ $x4 = x3 + f$

- Pour chaque expression, il faut créer une **variable temporaire** et générer un quadruplet
- chaque règle de grammaire contenant un opérateur arithmétique, l'action sémantique associée doit appeler la procédure CreerTemp pour créer une variable temporaire et appeler la procédure GEN pour générer le quadruplet correspondant. Les règles de traduction sont donc :

Génération du Code Intermédiaire

Traduction des expressions arithmétiques:

```
$$              $1          $2     $3
EXP_A     :     EXP_A(1)    '+'    EXP_A(2)
                 {
                   //Vérifier la compatibilité des types de $1 et $3
                   CréerTemp (temp);
                   //Insérer temp dans la TS et MAJ son type dans TS
                   GEN ($2, $1, $3, temp);
                   $$ = temp;   }

                $1          $2     $3
          |     EXP_A(1)    '-'    EXP_A(2)
                 {
                   //Vérifier la compatibilité des types de $1 et $3
                   CréerTemp (temp);
                   //Insérer temp dans la TS et MAJ son type dans TS
                   GEN ($2, $1, $3, temp);
                   $$ = temp;   }
```

Génération du Code Intermédiaire

■ Traduction des expressions arithmétiques:

$$		$1	$2	$3
EXP_A	**:**	**EXP_A⁽¹⁾**	**'*'**	**EXP_A⁽²⁾**

{
//Vérifier la compatibilité des types de $1 et $3
CréerTemp (**temp**);
//Insérer **temp** dans la TS et MAJ son type dans TS
GEN (**$2**, **$1**, **$3**, **temp**);
$$ = **temp**; }

	$1	$2	$3
|	**EXP_A⁽¹⁾**	**'/'**	**EXP_A⁽²⁾**

{
//Vérifier la compatibilité des types de $1 et $3
CréerTemp (**temp**);
//Insérer **temp** dans la TS et MAJ son type dans TS
GEN (**$2**, **$1**, **$3**, **temp**);
$$ = **temp**; }

Génération du Code Intermédiaire

- **Traduction des expressions arithmétiques:**

$$		$1	$2	$3
EXP_A	:	**'('**	**EXP_A(1)**	**')'**
		{ $$ = $2; }		
		$1		
	|	**Idf**		
		{ $$ = $1; }		
		$1		
	|	**Cste**		
		{ $$ = $1; }		

Génération du Code Intermédiaire

Traduction des instructions d'affectation:

- Pour l'instruction d'affectation $a = (b - c) \div d$, il faut d'abord traduire l'expression arithmétique $b - (c \div d)$ ensuite générer le quadruplet affectant le résultat à a. Le code de l'instruction d'affectation :

Expression Arithmétique	Traduction
$a = (b - c) \div d$	$x1 = b - c$ $x2 = x1 \div d$ $a = x2$

- La règle de traduction :

```
$$      :   $1      $2      $3
INS     :   Id      '='     EXP_A (1)
        { GEN ($2, $3, ,$1); }
```