**2010 – 2011**

**Deuxième année**

**Corrigé Examen 2 / Compilation**

1. Génération de tous les états et pour chaque état les actions entreprises par chaque élément LR(1)

Regle rajoutée : NT := . P / "#"

\*\*\* ETAT 1

 NT := . P / "#" Decaler P etat2

 P := . L / "#" Decaler L etat3

 L := . L ";" I / "#" ";" Decaler L etat3

 L := . I / "#" ";" Decaler I etat4

 I := . "i" " :=" "e" / "#" ";" Décaler “i” etat5

 I := . "TQ" "e" L "Ftq" / "#" ";" Decaler “TQ” etat 6

GOTO(ETAT 1 , P) = ETAT 2

 NT := P . / "#" Accept

GOTO(ETAT 1 , L) = ETAT 3

 P := L . / "#" Reduire ‘#’ Régle 1

 L := L . ";" I / "#" ";" Décaler  ‘ ;’ Etat7

GOTO(ETAT 1 , I) = ETAT 4

 L := I . / "#" ";" Reduire ‘#’ Régle 3 // Reduire ‘ ;’ Régle 3

GOTO(ETAT 1 , "i") = ETAT 5

 I := "i" . " :=" "e" / "#" ";" Décaler  ‘ :=’ Etat8

GOTO(ETAT 1 , "TQ") = ETAT 6

 I := "TQ" . "e" L "Ftq" / "#" ";" Décaler  ‘ e’ Etat9

GOTO(ETAT 3 , ";") = ETAT 7

 L := L ";" . I / "#" ";" Décaler  I Etat10

 I := . "i" " :=" "e" / "#" ";" Décaler  ‘ i’ Etat5

 I := . "TQ" "e" L "Ftq" / "#" ";" Décaler  ‘TQ’ Etat6

GOTO(ETAT 5 , " :=") = ETAT 8

 I := "i" " :=" . "e" / "#" ";" Décaler   "e" Etat13

GOTO(ETAT 6 , "e") = ETAT 9

 I := "TQ" "e" . L "Ftq" / "#" ";" Décaler  L Etat14

 L := . L ";" I / "Ftq" ";" Décaler  L Etat14

 L := . I / "Ftq" ";" Décaler  I Etat15

 I := . "i" ":=" "e" / "Ftq" ";" Décaler  "i" Etat16

 I := . "TQ" "e" L "Ftq" / "Ftq" ";" Décaler  "TQ" Etat17

GOTO(ETAT 7 , I) = ETAT 10

 L := L ";" I . / "#" ";" Reduire ‘#’ Régle 2 // Reduire ‘ ;’ Régle 2

GOTO(ETAT 7 , "i") = ETAT 5 // Existe

GOTO(ETAT 7 , "TQ") = ETAT 6 // Existe

GOTO(ETAT 8 , "e") = ETAT 13

 I := "i" " :=" "e" . / "#" ";" Reduire ‘#’ Régle 4 // Reduire ‘ ;’ Régle 4

GOTO(ETAT 9 , L) = ETAT 14

 I := "TQ" "e" L . "Ftq" / "#" ";" Décaler  "Ftq" Etat18

 L := L . ";" I / "Ftq" ";" Décaler  " ;" Etat19

GOTO(ETAT 9 , I) = ETAT 15

 L := I . / "Ftq" ";" Reduire ‘Ftq’ Régle 3 // Reduire ‘ ;’ Régle 3

GOTO(ETAT 9 , "i") = ETAT 16

 I := "i" . ":=" "e" / "Ftq" ";" Décaler  " :=" Etat20

GOTO(ETAT 9 , "TQ") = ETAT 17

 I := "TQ" . "e" L "Ftq" / "Ftq" ";" Décaler  "e" Etat21

GOTO(ETAT 14 , "Ftq") = ETAT 18

 I := "TQ" "e" L "Ftq" . / "#" ";" Reduire ‘#’ Régle 5 // Reduire ‘ ;’ Régle 5

GOTO(ETAT 14 , ";") = ETAT 19

 L := L ";" . I / "Ftq" ";" Décaler  I Etat22

 I := . "i" ":=" "e" / "Ftq" ";" Décaler  " i" Etat16

 I := . "TQ" "e" L "Ftq" / "Ftq" ";" Décaler  " TQ" Etat17

GOTO(ETAT 16 , "=") = ETAT 20

 I := "i" " :=" . "e" / "Ftq" ";" Décaler  " e" Etat25

GOTO(ETAT 17 , "e") = ETAT 21

 I := "TQ" "e" . L "Ftq" / "Ftq" ";" Décaler  L Etat26

 L := . L ";" I / "Ftq" ";" Décaler  L Etat26

 L := . I / "Ftq" ";" Reduire ‘Ftq’ Régle 3 // Reduire ‘ ;’ Régle 3

 I := . "i" ":=" "e" / "Ftq" ";" Décaler  " i" Etat16

 I := . "TQ" "e" L "Ftq" / "Ftq" ";" Décaler  " TQ" Etat17

GOTO(ETAT 19 , I) = ETAT 22

 L := L ";" I . / "Ftq" ";" Reduire ‘Ftq’ Régle 2 // Reduire ‘ ;’ Régle 2

GOTO(ETAT 19 , "i") = ETAT 16 // Existe

GOTO(ETAT 19 , "TQ") = ETAT 17 // Existe

GOTO(ETAT 20 , "e") = ETAT 25

 I := "i" " :=" "e" . / "Ftq" ";" Reduire ‘Ftq’ Régle 4 // Reduire ‘ ;’ Régle 4

GOTO(ETAT 21 , L) = ETAT 26

 I := "TQ" "e" L . "Ftq" / "Ftq" ";" Décaler  " Ftq" Etat30

 L := L . ";" I / "Ftq" ";" Décaler  " ;" Etat19

GOTO(ETAT 21 , I) = ETAT 15 // Existe

GOTO(ETAT 21 , "i") = ETAT 16 // Existe

GOTO(ETAT 21 , "TQ") = ETAT 17 // Existe

GOTO(ETAT 26 , "Ftq") = ETAT 30

 I := "TQ" "e" L "Ftq" . / "Ftq" ";" Reduire ‘Ftq’ Régle 5 // Reduire ‘ ;’ Régle 5

GOTO(ETAT 26 , ";") = ETAT 19 // Existe

1. **Quadruples générés**

0  : ( :=, e, , i)

1 : (B, e, 2, 10)

2 : (B, e, 3, 8)

3 : ( :=, e, , i)

4 : (B, e, 5, 7)

5 : ( :=, e, , i)

6 : (Br, 4,,)

7 : (Br, 2,,)

8 : (:=, e, ,i)

9 : (Br, 1)

10:

**2 Analyseur Syntaxico-sémantique**

Ptquad : variable globale initialisée à 0

Proc P

I; Tq syn=’;’ scan (;) I Ftq

Proc I : Sauv, X : variables locales

Si syn = i

Scan(i);

Check(:=);

Check(e) Générer quadruplé (:=, e,,i) ; Incrémenter Ptquad

Sinon

Check(Tq); Sauv := Ptquad

Check(e); Générer quadruplé (B, e, Ptquad+1, X) ; Incrémenter Ptquad

P

Check(Ftq) Dans le quadruplet n° Sauv remplacer X par Ptquad + 1

Fsi

1. **Instructions MSIL générées**

0 ldloc.s e

2 stloc.s i

4 ldloc.s e

6 ldc.i4 0

7 blt 42

11 ldloc.s e

13 ldc.i4 0

14 blt 27

18 ldloc.s e

20 stloc.s i

22 ldloc.s e

24 ldc.i4 0

25 blt 12

29 ldloc.s e

31 stloc.s i

33 br -11

37 br -26

41 ldloc.s e

43 stloc.s i

45 br -39

49 …

**Analyseur Syntaxico-sémantique**

Ptinst : variable globale initialisée à 0

Proc P

I; Tq syn=’;’ scan (;) I Ftq

Proc I : Sauv, X : variables locales

Si syn = i

Scan(i);

Check(:=);

Check(e) Générer "ldloc.s e" et "stloc.s i" ; Ptinst := Ptinst + 2 + 2

Sinon

Check(Tq);

Check(e); Générer " ldloc.s e" et "ldc.i4 0" ; Ptinst := Ptinst + 2 + 1

Sauv := Ptinst

Générer quadruplé "blt +z" ; Ptinst := Ptinst + 4

P

Check(Ftq) Dans l’instruction n° Sauv remplacer z par Ptinst + 1

v := ptins – Sauv ; Générer "br –v” ; ; Ptinst := Ptinst + 4

fsi