Corrigé Examen final
Compilation

exo1 :

a> Différence entre un langage statique et dynamique sur le plan Compilation et Exécution ?

- Un langage statique : tous les données, ainsi que leur taille et adresse sont connus - la taille globale est connue avant l'exécution - Pendant l'exécution, la taille est fixe aussi - Pas de récursivité, les allocations de structures dynamiques ne pris - Espace réservé à la compilation -
- un langage dynamique : taille des objets non connue à la compilation - Préparation d'un code en compilation et sera exécuté dans la phase exécution -

Lors de l'exécution d'un pgme, la taille est alors calculée, dès que l'on sort du pgme, l'espace mémoire est libéré -

b - Schéma de Traduction correspondant à l'appel

Call proc1 (< list - param >)

(R1) ↑ (R2) ↑

Grammaire syntaxique :

{ < inst - call > → Call proc1 (< list - param > (R1) (R3)

< list - param > → param , < list - param > (R2) (R2)

Analyse descendante, grammaire sémantique :

{ < inst - call > → Call proc1 < A > (< list - param >) <<

< list - param > → param < B > , < list - param

< A > , < B > , < C > → ε

Routine A → ε

lookup (id.nom , p) ;

Si p = 0 alors 'Erreur Sem'

Sinon ' Cela correspond au nom d'une procédure'

- Vérifier qu'elle a été déclaré

nbre de param := 0

fsi

Routine B → ε

- Rencontre d'un paramètre effectif, association avec le paramètre formel de la procédure.
- Si ¬ compatibilité de type alors 'Erreur Sem'
- Compter le param : nbre de param ++

Fonction <a> → E

- compte le nbre de param effectifs -
  Si nbre param ≠ nbre param formels
  alors 'erreur sem' ;
- Sauvegarde de l'adresse de retour.
- Générer un BR vers l'adresse de début de la procédure.

R_4 : Mise à jour du taille_elt selon le type du tableau

c) les quadruplets : (1,5)

$sc := \underbrace{x + y}_{T_3} + \underbrace{(z / a)}_{T_1} * b$

T_2

T_4

1. (/, z, a, T_1)
2. (*, T_1, b, T_2)
3. (+, x, y, T_3)
4. (+, T_3, T_2, T_4)
5. (:=, T_4, , sc)

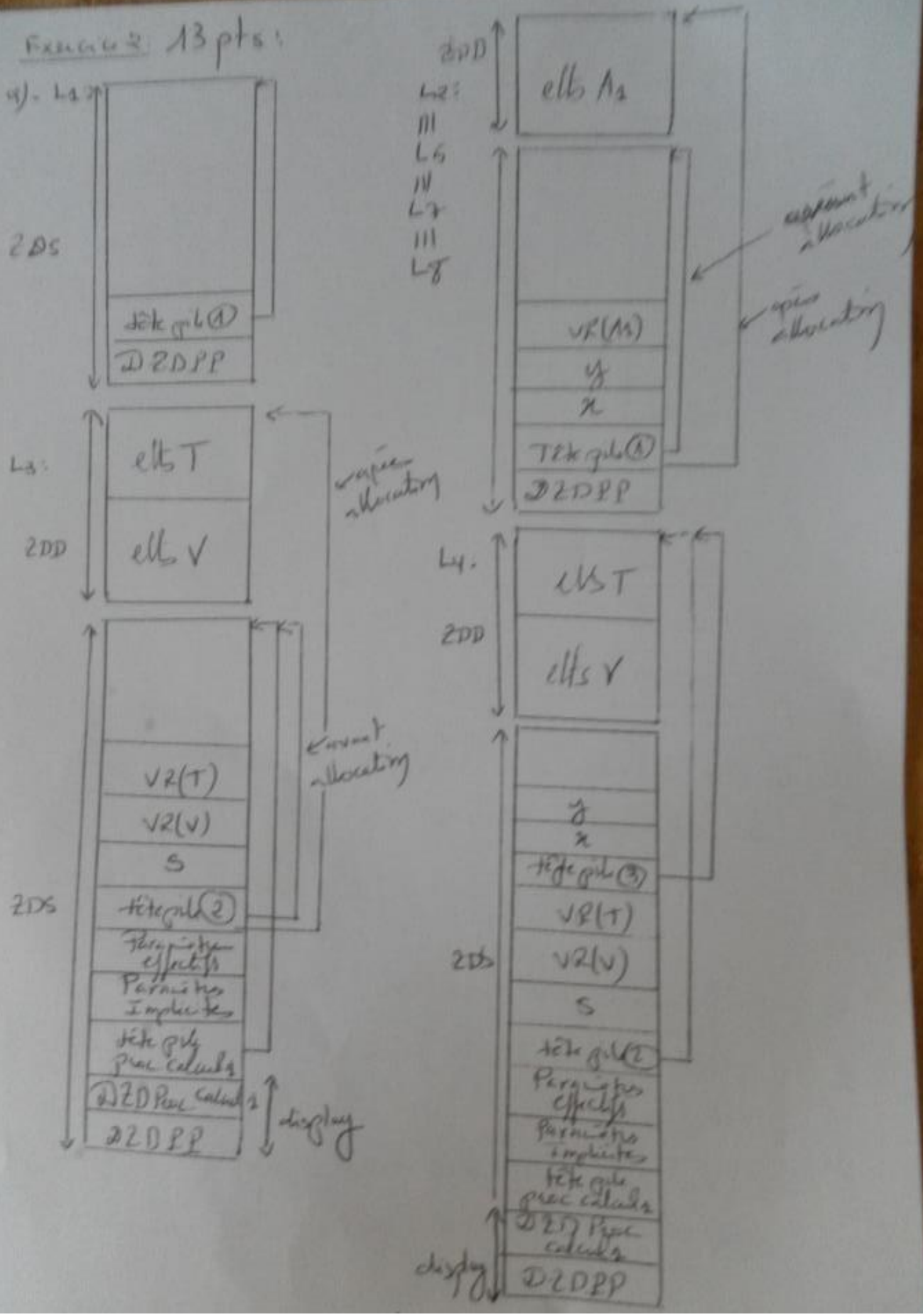
d) le code objet généré : (1,5)

GetinAcc	Acc	code
	vide	
GetinAcc ('z',)	z	load z
	T_1	Div a
		STORE T1
GetinAcc ('T1','b')	T2	MULt b
		store T2
GetinAcc ('x','y')	x	load x
	T3	ADD y
GetinAcc (T_2', T_3')		store T_3
Permutation des	T_4	ADD T2
opérandes dans		store X
le quadruplet		
(+, T_3, T_2, T_4)		

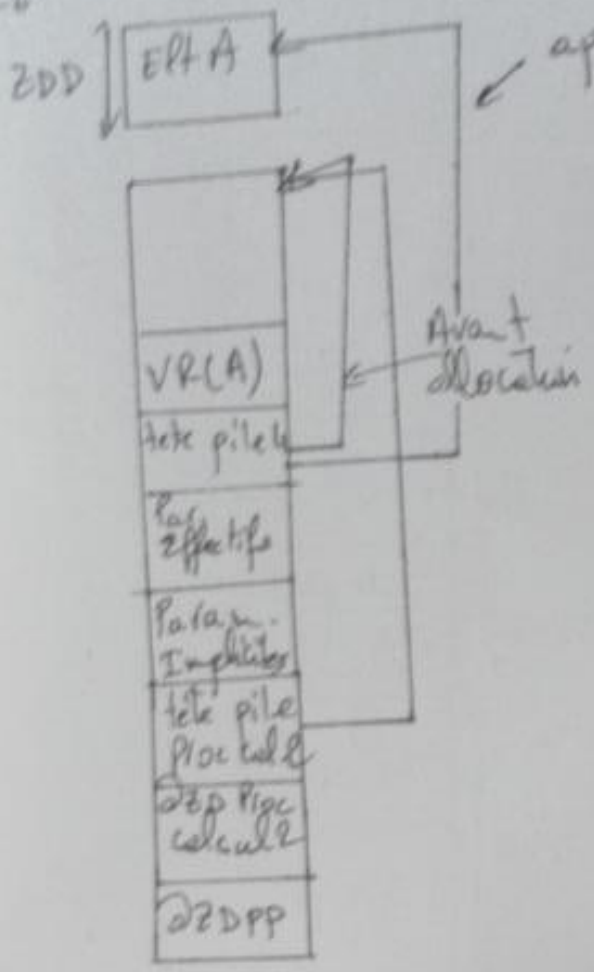
d) GetinACC : est une fonction de compilation, qui permet la génération du code objet pour le chargement des opérandes dans l'ACC et la mise à jour des quadruplets.

(1pt)

Exercice 2 13 pts :



L6



4 pts :
6 pour chaque pile

b) La déclaration d'un tableau A se traduit par : (Des différentes routines sémantiques) : 5 pts

$S \longrightarrow id\ f\ \langle R1 \rangle [\text{born\_inf}\ \langle R2 \rangle .. \text{born\_sup}\ \langle R3 \rangle]\ \text{array of type}\ \langle R4 \rangle$

R_1 : Vérification et Insertion dans la TS.

R_2 : Evaluation de la born_inf et vérification de son type

R_3 : Evaluation de la born_sup, vérification de son type et comparaison avec la born_inf