CCMP - Sujet A

Statut	Terminée
Commencé	lundi 23 juin 2025, 11:45
Terminé	lundi 23 juin 2025, 12:20
Durée	34 min 56 s

Question 1
Terminé
Noté sur 1,00
Marquer la question

Quelle est la fonction principale d'un compilateur ?

- a. Aider l'utilisateur à écrire du code
- b. Détecter des erreurs dans le code utilisateur
- c. Prouver mathématique la validité du code utilisateur
- d. Empêcher l'exécution de code incorrect
- e. Traduire le code utilisateur
- f. Optimiser le code utilisateur
- g. Générer un exécutable
- h. Exécuter le code utilisateur

Question 2
Terminé
Noté sur 0,50
Marquer la question

L'inférence de type s'effectue à la compilation.

- Vrai
- Faux

Question 3
Terminé
Noté sur 0,50
Marquer la question

L'overloading est résolu à l'exécution du programme.

- Vrai
- Faux

Question 4
Terminé
Noté sur 0,50
Marquer la question

La récupération d'erreur est nécessaire au bon fonctionnement d'un compilateur.

- Vrai
- Faux

Question 5
Terminé
Noté sur 0,50
Marquer la question

Le sucre syntaxique peut rendre les erreurs de compilation moins explicites.

- Vrai
- Faux

Description
Marquer la question

En Tiger, les valeurs de types composites (arrays et records) sont allouées dynamiquement, manipulées par pointeurs et peuvent prendre la valeur *nil* (équivalente à NULL en C). Essayer d'accéder à un champ de la valeur nil peut donc causer un segfault.

On souhaite rajouter au langage Tiger la notion de types *nullable*, similaires à ceux de Kotlin ou Zig. Un type sera déclaré nullable via un ?, et nil ne sera une valeur autorisée que pour les types marqués comme nullable. On il effectuant une comparaison avec nil nous permettra de déterminer statiquement si notre valeur de type nullable est déréférençable sans risque.(mlang)

```
let
  type nil_int = int?
  type arr_int = array of int
  type pair = { x: int ; y : nil_int }
  type nil_pair = pair?

  var never_nil_array : arr_int := nil /* Compilation error! Cannot assign nil to a non-nullable type */
  var maybe_nil_record : nil_pair := nil

  /* Primitive, non composite types may also be nullable. */
  var int_or_nil : nil_int := nil (* OK *)
  var int : int := int_or_nil /* Compilation error! Cannot assign nullable to a non-nullable type */
in
  maybe_nil_record := pair { x = 0 ; y = 42 };

  print_int(maybe_nil_array[0]); /* OK. */
  print_int(maybe_nil_record.x);  /* Compilation error! maybe_nil_record may be nil */

  if maybe_nil_record <> nil
  then (
    /* We know that we don't have a nil */
    print_int(maybe_nil_record.x); /* OK. */
    print_int(maybe_nil_record.y)  /* Compilation error! maybe_nil_record.y may be nil */
  )
  else
    /* We know that we do have a nil */
    print("nil")
end
```

On cherche à modifier notre compilateur existant pour supporter cette nouvelle fonctionnalité.

Question 6
Terminé
Noté sur 1,00
Marquer la question

L'ajout de types nullable à notre langage rend impossible...

- a. Le typage statique
- b. L'inférence de type
- c. Toute forme de segfault
- d. Aucune de réponses ci-dessus

Question 7
Terminé
Noté sur 0,50
Marquer la question

Pour représenter les types nullable dans l'AST : doit-on ajouter un nœud ou modifier un nœud ?

- a. Il faut modifier un nœud pré-existant.
- b. Il faut ajouter un nœud dédié.
- c. L'un ou l'autre.
- d. Il n'est pas nécessaire de modifier l'AST.
- e. L'un ou l'autre, mais ajouter un nœud est préférable.

Question 8
Terminé
Noté sur 1,00
Marquer la question

En supposant que nous disposons d'un visiteur par défaut qui sera mis à jour, quelles étapes de notre compilateur nous faudra-t-il modifier ?

- a. Parser
- b. Lexer
- c. Renamer
- d. Binder
- e. Register allocator

Question 9
Terminé
Noté sur 1,50
Marquer la question

On décide d'implémenter nos types nullable en passant par une étape de désucrage. Lesquelles de ces assertions sont vraies ?

- a. Le désucrage nous évite d'avoir à modifier notre étape de vérification de types.
- b. Ce désucrage doit s'effectuer avant la passe de vérification de types.
- c. Ce désucrage doit s'effectuer après la passe de vérification de types.
- d. Le désucrage diminue nécessairement la qualité des messages d'erreurs.
- e. Le désucrage nous évite d'avoir à modifier notre étape de génération de code intermédiaire.

Question 10
Terminé
Noté sur 3,00
Marquer la question

Proposer un désucrage pour tous nos types nullables (types, définitions et utilisations). Ce désucrage doit être valide aussi bien pour nos types primitifs que composites. Y a-t-il des désavantages à ce désucrage ? Comment les résoudre ?

Pour desucrer les types nullable une possibilite est d'ajouter un noeud OU avant celui du type nullable avec comme fils gauche nul et comme fils droit le type en question. Le OU mentionne precedemment n'est pas un ou logique mais donne une information au type-checker. Cela permettra lors du type-checking de verifier si null et le type en question sont compatibles avec l'operateur en question.

Question 11
Non répondue
Noté sur 3,00
Marquer la question

En HIR, l'instruction `Jump(e,labs)` prend en paramètre une expression `e` et une liste de labels `labs`. Expliquer brièvement le rôle de `e`. En particulier, pourquoi n'est il pas suffisant de sauter directement à un label.

Question 12
Terminé
Noté sur 1,50
Marquer la question

Connectez les types de graphes suivants à leurs définitions

Représentation de tous les chemins qui peuvent être suivis par un programme durant son éxecution. — Graphe de flot de controle

Graphe non orienté où chaque noeud représente une variable, et une arête entre deux noeuds indique que les variables correspondantes sont vivantes simultanément à un moment donné du programme. — Graphe d'interférence

Graphe orienté où chaque noeud représente une instruction et chaque arête contraint l'ordre d'exécution des noeuds qu'elle connecte pour préserver la sémantique du programme. — Graphe de dépendance

Question 13
Terminé
Noté sur 1,00
Marquer la question

Le problème d'allocation des registres est lié à un problème de

- a. Voyageur de commerce dans un graphe non-orienté
- b. Plus court chemin dans un graphe
- c. Coloration de graphes
- d. Voyageur de commerce dans un graphe orienté

Question 14
Terminé
Noté sur 1,00
Marquer la question

Dans la transformation d'un programme en forme SSA, à quoi servent les phi-fonctions ?

- a. Remplacer les instructions conditionnelles par des affectations parallèles
- b. Choisir, à un point de jonction du flot de contrôle, la bonne valeur d'une variable définie dans plusieurs blocs précédents
- c. Fusionner plusieurs instructions identiques pour optimiser le code
- d. Séparer les définitions de variables pour éviter les conflits de noms

Question 15
Terminé
Noté sur 1,00
Marquer la question

Quel est le principal avantage du découpage d'instructions en plusieurs étages (pipeline) dans une architecture RISC ?

- a. Augmenter le débit d'exécution en permettant l'exécution simultanée de plusieurs instructions à différents stades
- b. Simplifier la conception des circuits arithmétiques
- c. Réduire la taille des instructions pour économiser de la mémoire
- d. Éliminer complètement les dépendances entre instructions

Question 16
Terminé
Noté sur 1,00
Marquer la question

Dans un pipeline RISC, pourquoi les instructions de saut (jump) compliquent-elles la détermination de la prochaine instruction à exécuter ?

- a. Parce que les sauts modifient la taille des instructions
- b. Parce que les sauts empêchent le décodage des registres sources
- c. Parce que les instructions de saut sont toujours exécutées en parallèle
- d. Parce que la cible du saut n'est connue qu'après l'étape d'exécution, ce qui peut entraîner des bulles (stalls) dans le pipeline

Question 17
Terminé
Noté sur 0,50
Marquer la question

L'analyse de vivacité est une analyse avant (forward).

- Vrai
- Faux

Question 18
Terminé
Noté sur 0,50
Marquer la question

Qu'est-ce qu'une analyse avant (forward analysis) en analyse statique de programmes ?

- a. Une analyse où les informations sont propagées du début vers la fin du flot de contrôle
- b. Une analyse utilisée uniquement pour optimiser la mémoire
- c. Une analyse où les informations sont propagées de la fin vers le début du flot de contrôle
- d. Une analyse qui ne prend pas en compte le flot de contrôle

Question 19
Terminé
Noté sur 0,50
Marquer la question

Le spilling consiste à stocker temporairement une variable dans la mémoire quand il n'y a pas assez de registres disponibles

- Vrai
- Faux