Ce cours

- Deux parties
  - Architecture
  - Test
- En architecture
  - Représenter une architecture existante
  - Concevoir une architecture
  - Styles et tactiques
  - Evaluer une architecture

Méthode de représentation des architectures

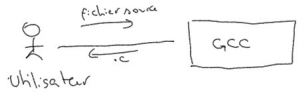
- A l'aide de vue
- Diagramme de contexte
- Liste de fonctionnalités
- Vues Logiques : globale puis détaillées pour décrire la structure
- Vues dynamiques pour illustrer des comportements
- Vue Physique : pour décrire la répartition sur le matériel

Evaluation

- Une description informelle (texte)
- Proposer une architecture
- Justifier vos choix

Diagramme de contexte

On s'intéresse à la suite GCC (GNU Compiler collection). Il s'agit d'une chaine de compilation. L'utilisateur peut écrire son programme dans trois langage différents (C, C++ et



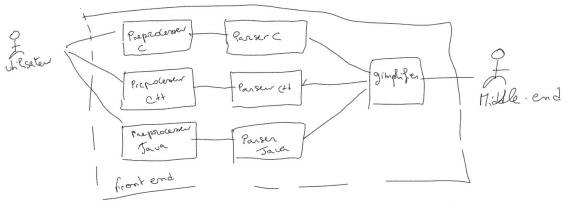
Liste des fonctionnalités

- production d'un AST
- production d'un RTL
- Code objet

Vue logique de haut niveau

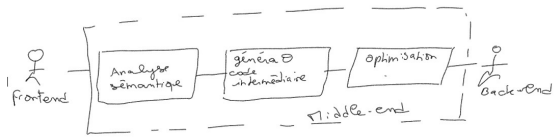
On s'intéresse à la suite GCC (GNU Compiler collection). Il s'agit d'une chaine de compilation. L'utilisateur peut écrire son programme dans trois langage différents (C, C++ et Java), et produire du code exécutable pour des cibles différentes. Au plus haut niveau d'abstraction, la chaine se présente sous la forme de trois parties, appelées respectivement ***frontend***, ***middle-end*** et ***backend***, qui sont exécutées l'une après l'autre.

Vue logique détaillée 1 : le front-end



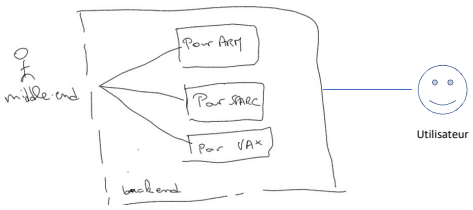
La partie ***frontend*** permet de lire le code source, de le parser et de le convertir en un Arbre Syntaxique Abstrait (AST). Pour cela, chaque programme est d'abord analysé par un ***préprocesseur***, chargé de détecter les fautes de syntaxe. Le résultat est transmis à un ***parseur*** qui produit un AST. Chaque langage a un préprocesseur et un parseur spécifique. A ce stade, la signification d'un AST est différente selon le langage source. C'est pourquoi une étape d'uniformisation est effectuée par un composant appelé ***simplifier***. Le résultat est un AST dans le langage GENERIC.

Vue Logique 2 : le middle-end



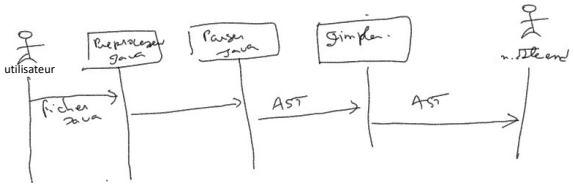
La partie ***middle-end*** est commune à tous les langages. Elle permet d'effectuer plusieurs transformations successives à partir d'un AST. Sans rentrer dans les détails, on considère que ces transformations successives correspondent respectivement à de l'***analyse sémantique***, à de la ***génération de code intermédiaire*** et à de l'***optimisation***. A la fin de l'optimisation, on obtient une représentation du programme en RTL (*Register Transfer Language*).

Vue Logique 3 : le back-end

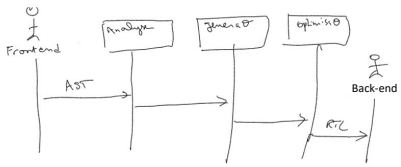


La partie ***backend*** permet la génération de code finale. GCC permet en particulier de produire de code objet pour adaptée à la cible d'exécution. Ainsi, GCC permet de produire du code objet pour ARM, SPARC, VAX, etc. Il y a un composant de génération de code par cible d'exécution.

Vue dynamique 1 : pré-processing d'un fichier Java



Vue dynamique 2 : Production d'un RTL optimisé à partir d'un AST



Vue Physique