Projet Programmation 1

1 Implémentation générale

Le code se compose des fichiers suivants :
— `aritha.ml` contient le code principal qui appelle les autres fonctions et crée le fichier assembleur `expression.s` ;
— `asyntax.ml` définit l'arbre syntaxique ;
— `lexer.mll` est le fichier `ocamllex` qui crée l'analyseur lexical ;
— `parser.mly` est le fichier `ocamlyacc` qui crée l'analyseur syntaxique (et en particulier s'occupe des questions de précédence) ;
— `typing.ml` vérifie que l'arbre syntaxique est bien typé, et fournit de plus le type de l'expression ;
— `generator.ml` convertit une expression sous forme d'arbre syntaxique (obtenu en appelant le parser) en code assembleur ;
— `functions.ml` contient les codes assembleur des fonctions puissance et factorielle (cf Bonus), qui sont ajoutées au code assembleur si elles doivent être appelées.

Remarque : En testant avec l'expression $-3/2$, on obtient le résultat -1 (on pourrait s'attendre à obtenir -2), tandis qu'en testant avec $-3\%2$, on obtient 1. Ceci est dû à l'opérateur `idiv` de l'assembleur, qui fournit ces résultats.

Le répertoire `tests` contient une série de fichiers `.exp` et `.rep`. L'ensemble des tests peuvent être exécutés avec l'instruction `make test`. Il contient en particulier des tests pour les opérations bonus.

2 Bonus

2.1 Opérations supplémentaires

On a ajouté les opérations suivantes : la puissance, la factorielle, et la division de flottants.

La division de flottants a la syntaxe suivante : `exp /. exp`. Elle prend en paramètre deux flottants, et calcule le flottant résultant de la division du premier par le deuxième.
La puissance a la syntaxe suivante : `exp ^ exp`. Elle prend en paramètre deux entiers et calcule la puissance du premier par le deuxième. Si l'exposant est négatif, elle renvoie 1.
La factorielle a la syntaxe suivante : `exp!`. Elle prend en paramètre un entier et renvoie sa factorielle.

2.2 Variables

La gestion de variables a été implémentée. Une affectation d'une valeur (entière ou flottante) à une variable se fait sur une ligne, de la forme `var = exp`. Le code assembleur affiche la valeur de la première ligne qui n'est pas une affectation. Les noms de variables disponibles sont seulement les lettres minuscules : il ne peut donc y avoir que 26 variables. Ainsi un fichier `expression.exp` utilisant des variables est de la forme suivante :

```
x = 2
y = float(x) -. 0.5
int(y)-x
```

Le programme renverrait ici la valeur : `-1`.