INFORMATIQUE — Mias 2 — TP 9

Codage et arithmétique entière

Récupération des fichiers

Les fichiers utilisés dans ce TP sont disponibles dans le répertoire `Info Mias-SM\Java\TP9` sur `jupiter`. Ce répertoire contient les fichiers suivants :

Nom du fichier	Contenu du fichier
`UALInter.java`	la classe qui définit l'interface d'un simulateur de calculateur
`UAL.java`	la classe qui définit le calculateur binaire
`Registre.java`	la classe qui définit les registres
`Bit.java`	la classe qui définit les bits

Ce répertoire contient également un répertoire `doc`, contenant des pages `HTML` décrivant la documentation de chacune des classes.

Avant de commencer à travailler, il vous faut copier les fichiers `.java` dans un répertoire qui vous est propre. Pour cela, effectuez les opérations suivantes :

1. créez un nouveau répertoire `TP9` dans le répertoire `Java` de votre répertoire personnel (`Z:`);
2. copiez les quatres fichiers `.java` de `jupiter` dans votre répertoire `TP9` (utilisez les raccourcis `Ctrl-C` et `Ctrl-V`, ou cliquez sur le bouton droit et sélectionnez les libellés `copier` et `coller`);

La documentation n'a pas besoin d'être copiée. Pour y accéder plus facilement, vous pouvez créer dans votre répertoire `TP9` un *lien* (appelé *raccourci* sous Windows) vers le répertoire `TP9\doc` de `jupiter`. Ce lien peut être créé par le menu contextuel (obtenu d'un clic sur le bouton droit de la souris) de votre répertoire `TP9`.

Cela étant fait, vous pouvez lancer `Emacs`.

Objectifs de l'application

L'application fournie est un simulateur de calculateur binaire. Les calculs s'effectuent sur des valeurs codées sur n bits (n est un paramètre du calculateur). Un bit est représenté par un objet de type `Bit`, un mot est représenté par un objet de type `Registre`.

Les codages binaires sont standards : le bit de poids fort code le signe et les nombres négatifs sont codés en « complément à deux ». Le bit de poids faible est situé à l'indice maximum du tableau, le bit de signe est situé à l'indice 1.

Comme dans un processeur réel, les opérations de calcul élémentaires mettent à jour un certain nombre d'indicateurs (les "flags"). Ces indicateurs servent lorsqu'on programme des suites de calculs. Les indicateurs sont constitués d'un bit (un caractère). Il en existe cinq :

- la retenue du dernier pas d'opération,
- un bit indiquant si le résultat du calcul est nul,
- un bit indiquant si le résultat du calcul est strictement négatif,
- un bit indiquant si si le calcul a produit un dépassement de capacité (similaire à la retenue) et
- la valeur du bit qui "sort" lors d'un décalage de registre.

La version fournie implante deux opérations : l'addition et la multiplication. Cependant, il nést pas possible d'utiliser des entiers négatifs, le calcul de l'opposé d'un nombre ne fonctionnant pas

.../...

encore. Nous vous proposons d'implanter les autres opérations arithmétiques (opposé, soustraction et division) en définissant des méthodes dans les classes `Registre` et `UAL`.

Attention : quelques modification ont été faites par rapport aux documents que vous avez eus lundi dernier (en particulier dans la classe `Registre`).

La soustraction

Implantez cette opération dans la méthode `soustrait` de la classe `Registre`, en utilisant les opérations de complément à deux (à définir) et d'addition (déjà définie) de la même classe . Vérifiez la validité de cette méthode en essayant l'interface.

Le décalage de registres

Cette opération consiste à décaler un registre dans un autre. Ainsi, le décalage suivant :

$$X \leftarrow 01010111 \leftarrow 00110101 \leftarrow 0$$

donne pour résultat :

$$0 \leftarrow 10101110 \leftarrow 01101010 \leftarrow -$$

Cette opération sera utilisée dans l'algorithme de division. Lisez le commentaire de la méthode `decale` de la classe `Registre`, qui donne des indications sur le comportement de cette opération.

La comparaison

Implantez, dans la méthode `compare` de la classe `Registre`, une opération de comparaison de deux registres. Le résultat est présenté dans les deux bits passés en paramètre, avec les conventions suivantes :

- si les deux nombres sont égaux, le bit `egaux` vaut 1,
- si les deux nombres sont différents, le bit `egaux` vaut 0 et
  - le bit `plusPt` vaut 1 si le registre courant est supérieur à celui donné en paramètre,
  - le bit `plusPt` vaut 0 sinon.

La division

L'algorithme de la division utilise quatre registres contenant respectivement le quotient, le dividende, le diviseur et le dividende courant. Initialement, le dividende courant et le quotient sont mis à 0. Un pas de division consiste à faire les opérations suivantes :

1. on décale à gauche le quotient,
2. on décale le dividende dans le dividende courant (méthode `decale`),
3. on compare le dividende courant et le diviseur (méthode `compare`),
4. si le diviseur est plus petit,
   - on soustrait le diviseur du dividende courant, en s'assurant que le résultat est bien dans ce dernier (méthode `soust`) et
   - on met le bit de poids faible du quotient à 1.

On répète cette boucle autant de fois qu'il y a de bits dans les mots (jusqu'à ce que le dividende ait été vidé par les décalages).

Commencez par définir la méthode `divise` de la classe `Registre`, puis utilisez celle-ci dans la méthode `divise` de la classe `UAL` (nécessaire au fonctionnement de l'interface).