

TP05 - GÉNÉRATION DE CODE PRÉ-ASSEMBLEUR

1. OBJECTIF

L'objectif de ce TP est de générer du code pré-assembleur à partir du code trois adresses. Pour chaque instruction du code trois adresses, une ou plusieurs instructions assembleur sont produites. L'implémentation du module de génération du pré-assembleur peut être réalisé à l'aide du visiteur `C3aVisitor` et du package `nasm`.

2. LE PACKAGE `nasm`

Le package `nasm` définit une classe pour chaque instruction `nasm`. Ces classes définissent chacune un constructeur qui permet de créer une instance de l'instruction. Il définit aussi une classe pour chaque type d'opérande. Ces dernières sont au nombre de quatre :

- `NasmAddress`
- `NasmConstant`
- `NasmRegister`
- `NasmLabel`

Le package `nasm` définit aussi la classe générale `Nasm` qui comporte en particulier la liste des instructions `nasm` qui vont être produites.

La classe `C3a` définit quatre méthodes importantes ;

- `public void ajouteInst(NasmInst inst)` ajoute une instruction à la séquence d'instructions créées.
- `public NasmRegister newRegister()` crée un nouveau registre, dont l'indice est calculé à partir de la variable d'instance `regCounter`
- `public void affiche(String baseFileName)` écrit dans le fichier `baseFileName` une représentation textuelle du code `nasm`.

La classe `nasm` définit aussi sept macros : `REG_EAX`, `REG_EBX`, `REG_ECX`, `REG_EDX`, `REG_ESP`, `REG_EBP` et `REG_UNK` qui permettent de colorer des registres grâce à la méthode `colorRegister(int color)` de la classe `NasmRegister`.

Pour créer un registre, qui devra correspondre dans le code final au registre général `eax`, par exemple, il faut faire :

```
NasmRegister reg_eax = nasm.newRegister();
reg_eax.colorRegister(Nasm.REG_EAX);
```

La procédure est la même pour `ebx`, `ecx` et `edx`.

Pour créer un registre qui n'est pas un registre général, par exemple `ebp`, il faut faire :

```
NasmRegister reg_ebp = new NasmRegister(Nasm.REG_EBP);
reg_ebp.colorRegister(Nasm.REG_EBP);
```

La procédure est la même pour `esp`.

La différence entre ces deux manières de procéder est que dans la première, le registre créé est un registre général, c'est lors de l'allocation de registre qu'il sera associé à un véritable registre du processeur, mais on indique que ce dernier devra être `eax`. Dans la seconde, il ne s'agit pas d'un registre général,

3. LA CLASSE `c3a2nasm`

C'est la classe que vous devez implémenter (elle peut avoir le nom que vous voulez). Elle implémente l'interface `C3aVisitor`, qui est composée de 17 méthodes correspondant à des instructions `c3a` et 5 méthodes correspondant à des opérandes. Ce sont ces méthodes que vous devez écrire.

Le principe de la traduction de chaque type d'instruction et type d'opérande est disponible dans les transparents du cours. Voici à titre d'exemple la méthode permettant de traiter l'instruction `C3aInstAdd` :

```
public NasmOperand visit(C3aInstAdd inst)
{
    NasmOperand label = (inst.label != null) ? inst.label.accept(this) : null;
    NasmOperand oper1 = inst.op1.accept(this);
    NasmOperand oper2 = inst.op2.accept(this);
    NasmOperand dest = inst.result.accept(this);
    nasm.ajouteInst(new NasmMov(label, dest, oper1, ""));
    nasm.ajouteInst(new NasmAdd(null, dest, oper2, ""));
    return null;
}
```