

Le package nasm

Alexis Nasr
Carlos Ramisch
Manon Scholivet
Franck Dary

Compilation – L3 Informatique
Département Informatique et Interactions
Aix Marseille Université

Le package nasm

Il est composé de 32 classes se décomposant de la façon suivante :

- 17 classes correspondant à des instructions
- 6 classes correspondant à des opérandes
- 1 visiteur (NasmVisitor)
- 1 classe principale (Nasm)

La classe principale Nasm

```
public class Nasm{
public List<NasmInst> listeInst;
private int tempCounter;
Ts tableGlobale;
public static int REG_EAX = 0;
public static int REG_EBX = 1;
public static int REG_ECX = 2;
public static int REG_EDX = 3;
public static int REG_ESP = -1;
public static int REG_EBP = -2;
public static int REG_UNK = -3;

public Nasm(Ts tableGlobale){...}
public int getTempCounter(){return this.tempCounter;}
public int setTempCounter(int c){return this.tempCounter = c;}

public void ajouteInst(NasmInst inst){...}
public NasmRegister newRegister(){return new NasmRegister(tempCounter++)}
public void affichePreamble(PrintStream out){...}
public void affiche(String baseFileName){...}
```

La méthode ajouteInst

```
public void ajouteInst(NasmInst inst){
    if(inst instanceof NasmMov && inst.dest instanceof NasmAddress
        && inst.src instanceof NasmAddress){
        NasmRegister newReg = newRegister();
        listeInst.add(new NasmMov(inst.label, newReg, inst.src, inst.comment))
        listeInst.add(new NasmMov(inst.label, inst.dest, newReg, "on passe par"))
        return;
    }
    if(inst instanceof NasmCmp && inst.dest instanceof NasmConstant){
        NasmRegister newReg = newRegister();
        listeInst.add(new NasmMov(inst.label, newReg, inst.dest, inst.comment))
        listeInst.add(new NasmCmp(inst.label, newReg, inst.src, "on passe par"))
        return;
    }
    listeInst.add(inst);
}
```

Permet de gérer le cas où :

- les deux opérandes d'un mov sont des adresses;
- la dest d'un cmp est une constante.

Opérandes

- Une classe abstraite : `NasmOperand`
- Quatre classes héritant de `NasmOperand`
 - `NasmAddress`
 - `NasmConstant`
 - `NasmRegister`
 - `NasmLabel`

NasmOperand

```
public abstract class NasmOperand{
    public boolean use = false;
    public boolean def = false;

    public boolean isGeneralRegister(){
        return false;
    }
}
```

- use permet de savoir si la valeur de l'opérande est utilisée par l'instruction dont elle est une opérande.
- def permet de savoir si la valeur de l'opérande est modifiée par l'instruction dont elle est une opérande.
- Nécessaire pour l'allocation de registres.
- isGeneralRegister() renvoie true pour eax, ebx, ecx, edx

NasmAddress

```
public class NasmAddress extends NasmOperand {
    public NasmOperand base;
    public char direction;
    public NasmOperand offset;

    public NasmAddress(NasmOperand base, char direction, NasmOperand of
    public NasmAddress(NasmOperand base){...}
    public String toString(){...}
}
```

Une adresse est formée de trois éléments :

- Une base
- Une direction : + ou -
- Un déplacement

Utile pour :

- L'accès aux éléments d'un tableau : `mov dword [t+4*5], 123;`
- L'accès aux éléments de la trame de pile : `mov dword [ebp-4*1], 2;`

NasmRegister

```
public class NasmRegister extends NasmOperand {  
    public int val;  
    public int color = Nasm.REG_UNK;  
  
    public NasmRegister(int val){...}  
    public void colorRegister(int color){this.color = color;}  
    public String toString(){...}  
    public boolean isGeneralRegister(){return this.val >=0;}  
}
```

Un registre possède deux identifiants :

- val est l'identifiant temporaire du registre. C'est le registre "fictif" avant l'étape d'allocation de registre
- color est l'identifiant final du registre : eax, ebx, ecx, edx, esp, ebp

NasmConstant

```
public class NasmConstant extends NasmOperand {  
    public int val;  
    public NasmConstant(int val){  
        this.val = val;  
    }  
}
```

NasmLabel

```
public class NasmLabel extends NasmOperand {  
    public String val;  
    public NasmLabel(String val){  
        this.val = val;  
    }  
}
```

Les instructions

- Une classe abstraite `NasmInst`
- 23 classes héritant de `NasmInst`
 - Transfert : `NasmMov`
 - Opérations arithmétiques : `NasmAdd`, `NasmSub`, `NasmMul`, `NasmDiv`
 - Opérations logiques : `NasmAnd`, `NasmOr`, `NasmXor`, `NasmNot`
 - Comparaison : `NasmCmp`
 - Sauts : `NasmJmp`, `NasmJge`, `NasmJg`, `NasmJne`, `NasmJle`, `NasmJe`, `NasmJl`
 - Pile : `NasmPop`, `NasmPush`
 - Fonctions : `NasmCall`, `NasmRet`
 - Interruption : `NasmInt`
 - Vide : `NasmEmpty`

NasmInst

```
public abstract class NasmInst{  
    public NasmOperand label = null;  
    public NasmOperand dest = null;  
    public NasmOperand src = null;  
    public NasmOperand address = null;  
    String comment;  
}
```

NasmAdd

```
public class NasmAdd extends NasmInst {
    public NasmAdd(NasmOperand label, NasmOperand dest, NasmOperand src) {
        dest.use = true;
        dest.def = true;
        src.use = true;
        this.label = label;
        this.dest = dest;
        this.src = src;
        this.comment = comment;
    }
    public String toString(){...}
}
```

C'est là que l'on indique que l'instruction `add dest src` :

- utilise les valeurs de `dest` et `src` ;
- modifie la valeur de `dest`.

Le visiteur NasmVisitor

```
public interface NasmVisitor <T> {
    public T visit(NasmAdd inst);
    public T visit(NasmCall inst);
    public T visit(NasmDiv inst);
    public T visit(NasmJe inst);
    public T visit(NasmJle inst);
    public T visit(NasmJne inst);
    ...
    public T visit(NasmAddress operand);
    public T visit(NasmConstant operand);
    public T visit(NasmLabel operand);
    public T visit(NasmRegister operand);
}
```