Le pré-processor / macro-expanseur

Assembleur

Younès El Amrani

Date de création: Mai 2006

Dernière modification: mai 2010

Le pré-processeur / macro-expanseur NASM

- Toutes les macro-instructions commencent par le symbole « ***%*** »
- Le préprocesseur Nasm offre plusieurs possibilités:
  - L'assemblage conditionnel
  - L'inclusion multiple de fichiers

un fichier qui inclut un fichier qui inclut un autre fichier...

  - Deux formes de macros:
    - Les macros définies sur une seule ligne
    - Les macros définies sur plusieurs lignes

Macros définies sur une ligne

- On utilise la macro instruction *%define*
- L'utilisation de *%define* est similaire au C
- Exemple:

```
;; Soient les deux macros suivantes:
%define crtl 0x1F  &
%define param( a, b )  ( ( a ) + ( a ) * ( b ) )
;; Soit  l'instruction assembleur suivante:
mov byte [  param( 2, ebx ) ]  , crtl   'D'
;; sera expansée en:
mov byte [  ( 2 ) + ( 2 ) * ebx ) ]  ,  0x1F    &    'D'
```

Macros expansées lors de l'invocation

- Les macros sont expansées lors de leurs invocations et non lors de leurs définitions
- Exemple:

;; Soient les deux macros suivantes:

%define a(x) 1 + b(x)

*%define b(x) 2 * x*

;; Soit l'instruction assembleur suivante:

mov ax, a (8)

;; sera expansée en:

*mov ax, 1 + 2 * 8 ;; cette expansion a lieu bien que b a été définie après a*

noms de macros: minuscules ou majuscules significatives !

- Les noms de macros sont sensibles aux minuscules et majuscules
- Exemple:

;; Soient la macro

%define foo bar

;; soit l'instruction assembleur suivante:

opcode foo, Foo

;; sera expansée en:

opcode bar, Foo ;; seule foo a été expansée car Foo diffère de foo.

Cependant la définition %idefine foo bar est insensible aux majuscules / minuscules. Le « i » de idefine signifie en anglais « insensitive »

Les macros circulaires

- Le préprocesseur n'expansera une macro qu'une seule fois si la macro contient une référence à elle-même. On parle alors de macros circulaires.
- Exemple:

;; Soient la macro définition suivante:

%define a(x) 1 + a(x)

;; Soit la macro instruction suivante:

mov ax, a(3)

;; sera expansée en:

mov ax, 1 + a(3) ;; aucune autre expansion n'aura lieu

Les macros surchargées

- Il est possible de surcharger les noms de macros
- Cependant une définition de macro sans paramètres ne permet plus de définition de macros avec paramètres et vice-versa.
- Exemple:

;; Soient la macro définition suivante:

%define foo(x) 1 + x

;; Il est possible de surcharger la macro définition ci-dessus:

*%define foo(x , y) 1 + (x * y)*

;; Le macro processeur fera la différence entre les deux macros en comptant

;; les paramètres

La concaténation de chaînes dans les macros

- Il est possible de concaténer des chaînes de caractère à l'aide de l'opérateur %+
- Exemple:

;; Soient le(s) macro(s) définition(s) suivante(s):

%define _mafonction _autrefonction

%define cextern(x) _ %+ x

;; Soit la macro suivante:

cextern (mafonction)

;; Le macro processeur dans une première passe génèrera

_mafonction

;; et dans une seconde passe il génèrera

_autrefonction

Un-définir des macros

- Les macros définies par une seule ligne peuvent-être in-définies à l'aide de la commande

%undef

- Exemple:

;; Soient le(s) macro(s) définition(s) suivante(s):

%define foo bar

;;; some code dans lequel « foo » sera relplacé par « bar »

%undef foo

;; Soit l'instruction

mov eax , foo

;; Le macroprocesseur va générer

mov eax, foo ;; plus de changements car la macro foo a été undéfinie

Macros définies sur plusieurs lignes (syntaxe NASM)

- Exemple:

```
;; Soient le(s) macro(s) définition(s) suivante(s):
%macro prologueC  1  ;; le 1 signifie 1 seul argument, il sera référencé par %1
push ebp
mov  ebp , esp
sub   esp , %1
%endmacro
;; De manière générale:
%macro   <name>     <n>   ;; n est le nombre d'arguments
    <instructions>              ;;  %1 référence le 1er argument...%n réf. le nième
%endmacro      ;; pas de changements car  la macro   foo a été undéfinie
```

Macros avec des étiquettes

- Si on définit une macro qui contient une étiquette, on doit alors écrire l 'étiquette avec le préfixe %%
- Ainsi une étiquette unique sera générée à chaque appel de la macro.