Apprendre le langage assembleur avec l'aide du compilateur

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 28 septembre 2011. Dernière mise à jour le 29 septembre 2011

http://www.bortzmeyer.org/assembleur-compilo.html

Tout programmeur le sait, programmer en langage assembleur est difficile. Avant d'avoir pu écrire un "*Hello world*", on doit maîtriser beaucoup de détails. Même chose pour des opérations triviales comme de la simple arithmétique. Mais le compilateur peut nous aider dans notre démarche de formation : il a souvent une option pour produire du code assembleur lisible, permettant au débutant d'écrire dans un langage de plus haut niveau, avant de voir le code produit par son programme.

Supposons qu'on veuille mettre 3 dans une variable, en langage assembleur. Et l'incrémenter de 1. Mais l'assembleur, c'est dur. On va donc écrire en C, car c'est plus facile :

```
% cat plus.c
main()
{
  int a;
  a = 3;
  a++;
}
```

Et le compilateur gcc le traduit en langage assembleur (option `-S`) pour nous (le `-O0` est pour éviter toute optimisation), ici pour i386 :

```
% gcc -O0 -S -l plus.c
% cat plus.s
        .file   "plus.c"
        .text
.globl main
        .type   main, @function
main:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $16, %esp         # C'est moi qui ait ajouté les commentaires
        movl    $3, -4(%ebp)      # Mettre 3 dans la mémoire (movl = Move Long)
        addl    $1, -4(%ebp)      # Ajouter 1 à ce qui est dans la mémoire
        leave
        ret
        .size   main, .-main
        .ident  "GCC: (Debian 4.4.5-8) 4.4.5"
        .section        .note.GNU-stack,"",@progbits
```

Pour lire ce code, il faut se souvenir que :
— un entier fait quatre octets (d'où le -4 devant l'identificateur de l'adresse %ebp)
— $ désigne un littéral donc $3, c'est 3

Des options de gcc comme `-fverbose-asm` permettent d'avoir davantage de détails dans le code assembleur. On peut donc ensuite faire des programmes C de plus en plus compliqué et apprendre l'assembleur en voyant ce que fait le compilateur.

Si on veut du vrai langage machine à partir du code assembleur :

```
% as -alhnd plus.s
```

Encore plus fort, si on veut un "*listing*" avec le code machine et le code C en commentaire, pour mieux suivre ce qu'a fait le compilateur :

```
% gcc -O0 -c -g -Wa,-a,-ad plus.c
...
  4:plus.c        ****   a = 3;
  26                            .loc 1 4 0
  27 0006 C745FC03              movl    $3, -4(%ebp)
  27      000000
   5:plus.c        ****   a++;
  28                            .loc 1 5 0
  29 000d 8345FC01              addl    $1, -4(%ebp)
```

Et avec d'autres processeurs (et donc d'autres langages assembleur) que le i386 ? André Sintzoff propose, pour les ARM :

```
; generated by ARM C/C++ Compiler, 4.1 [Build 481]
; commandline armcc [-S -O0 plus.c]
...
main PROC
        MOV      r1,#3      ; mettre 3 dans le registre r1
        ADD      r1,r1,#1   ; ajouter 1 à ce registre
        MOV      r0,#0      ; mettre 0 comme valeur de retour de la fonction
        BX       lr         ; aller à l'adresse contenue dans le registre lr
        ENDP
...
```

Et avec MIPS (sp = "*stack pointer*", fp = "*frame pointer*"), un code très verbeux :

```
...
main:
        .frame  $fp,16,$ra              # vars= 8, regs= 1/0, args= 0, extra= 0
        .mask   0x40000000,-8
        .fmask  0x00000000,0
        subu    $sp,$sp,16
        sw      $fp,8($sp)
        move    $fp,$sp
        li      $v0,3                   # 0x3
        sw      $v0,0($fp)
        lw      $v0,0($fp)
        addu    $v0,$v0,1
        sw      $v0,0($fp)
        move    $sp,$fp
        lw      $fp,8($sp)
        addu    $sp,$sp,16
        j       $ra
        .end    main
...
```

Pour ceux qui veulent apprendre le langage assembleur en partant d'un langage de plus haut niveau, Olivier Bonaventure me recommande le livre « "*The Art of Assembly Language Programming*" <http://homepage.mac.com/randyhyde/webster.cs.ucr.edu/www.artofasm.com/index.html> », qui part d'un langage de haut niveau proche du C pour aller progressivement vers l'assembleur. Sinon, le livre de Hennessy & Patterson, « "*Computer Architecture : A Quantitative Approach*" » démarre aussi sur une description du langage assembleur en partant du C.