Cross-Compilation avec GCC : générer du code MIPS

Christophe Cérin
cerin@laria.u-picardie.fr

Problème : à partir d'un programme écrit dans une description haut niveau (en langage C par exemple), on veut produire de l'assembleur MIPS réutilisable par le simulateur SPIM. La machine sur laquelle on travaille est un PC (architecture intel 32 bits) et cette machine tourne sous Linux.

Notre machine a par défaut un compilateur : GCC (http://gcc.gnu.org). GCC peut etre configuré pour générer de l'assembleur MIPS : cette manipulation s'appelle une cross-compilation (compilation croisée).

Pour arriver à nos fins, on installe si nécéssaire la dernière version de GCC et on se déplace sour le répertoire d'installation. Dans l'exemple qui suit il s'agit du répertoire `gcc-3.2.` Puis vous suivez les instructions suivantes pour recompiler GCC afin qu'il puisse produire du code MPIS :

```
[root@e023 gcc-3.2]# ./configure –--target=mips
[root@e023 gcc-3.2]# make

On obtient une erreur mais ça passe à l'installation

/usr/local/gcc-3.2/gcc/xgcc -B/usr/local/gcc-3.2/gcc/
-B/usr/local/mips/bin/ -B/usr/local/mips/lib/ -isystem
/usr/local/mips/include -O2 -DIN_GCC -DCROSS_COMPILE -W -Wall -Wwrite-
strings -Wstrict-prototypes -Wmissing-prototypes -isystem
./include -G 0 -g -DIN_LIBGCC2 -D__GCC_FLOAT_NOT_NEEDED -Dinhibit_libc
-I. -I. -I. -I./. -I./config -I./../include -DL_m16addsf3 -xassembler-
with-cpp -c ./config/mips/mips16.S -o libgcc/./_m16addsf3.o
as: option non reconnue `-G'
make[2]: *** [libgcc/./_m16addsf3.o] Erreur 1
make[2]: Quitte le répertoire `/usr/local/gcc-3.2/gcc'
make[1]: *** [stmp-multilib] Erreur 2
make[1]: Quitte le répertoire `/usr/local/gcc-3.2/gcc'
make: *** [install-gcc] Erreur 2
[root@e023 gcc-3.2]#

[root@e023 gcc-3.2]# make install
```

Voyons un exemple maintenant. Soit le programme C suivant :

```
/*#include<stdio.h>*/

#define MAX 10

int T[MAX];

main(){

  int i;
  for(i=0;i<MAX;i++) T[i] = 2*i;
  printf("bonjour\n");
}
```

On le compile par (option -S pour générer uniquement l'assembleur qui est rangé dans le fichier toto.s – l'option -b permet de choisir le format MIPS) :

```
gcc -b mips -S toto.c
```

et on obtient le code qui suit. On a du réécrire à la main l'opération d'affichage (et plus généralement toutes les opérations d'entrées sorties).

```
        .file   1 "toto.c"
        .rdata
        .align  2
$LC0:
        .ascii  "bonjour\n\000"
        .text
        .align  2
        .globl  main
        .ent    main
main:
        .frame  $fp,32,$31  # vars= 8, regs= 2/0,
                            # args= 16, extra= 0
        .mask   0xc0000000,-4
        .fmask  0x00000000,0
        subu    $sp,$sp,32
        sw      $31,28($sp)
        sw      $fp,24($sp)
        move    $fp,$sp
        sw      $0,16($fp)
$L2:
        lw      $2,16($fp)
        slt     $2,$2,10
        bne     $2,$0,$L5
        j       $L3
$L5:
        lw      $2,16($fp)
        sll     $3,$2,2
        la      $2,T
        addu    $3,$3,$2
        lw      $2,16($fp)
        sll     $2,$2,1
        sw      $2,0($3)
        lw      $2,16($fp)
        addu    $2,$2,1
        sw      $2,16($fp)
        j       $L2
$L3:
        # la    $4,$LC0  # on commente cette ligne
        li $v0, 4        # system call code for print_str
        la $a0, $LC0     # address of string to print
        syscall          # print the string
        move    $sp,$fp
        lw      $31,28($sp)
        lw      $fp,24($sp)
        addu    $sp,$sp,32
        j       $31
        .end    main
        # on commente la ligne suivante
        # .comm T,40
```

A l'exécution on obtient :

```
[christophe@e023 SPIM]$ spim
SPIM Version 6.1 of January 16, 1998
Copyright 1990-1997 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
```

```
See the file README for a full copyright notice.
Loaded: ./trap.handler
(spim) load "/home/christophe/toto.s"
(spim) run
bonjour
\000(spim)
```

Le programme s'exécute sans aucun problème ! En fait il y a un problème de mémoire car la zone mémoire pour le vecteur T n'est pas réservée ! La version `spim-6.5` détecte une référence inconnue à T. On doit donc ajouter au code la déclaration suivante :

```
T:
        .word   1 : 1
        .word   2 : 1
        .word   3 : 1
        .word   4 : 1
        .word   5 : 1
        .word   6 : 1
        .word   7 : 1
        .word   8 : 1
        .word   9 : 1
        .word   10 : 1
        .word   11 : 1
        .word   12 : 1
        .word   13 : 1
        .word   14 : 1
        .word   15 : 1
        .word   16 : 1
        .word   17 : 1
        .word   18 : 1
        .word   19 : 1
        .word   20 : 1
        .word   21 : 1
        .word   22 : 1
        .word   23 : 1
        .word   24 : 1
        .word   25 : 1
        .word   26 : 1
        .word   27 : 1
        .word   28 : 1
        .word   29 : 1
        .word   30 : 1
        .word   31 : 1
        .word   32 : 1
        .word   33 : 1
        .word   34 : 1
        .word   35 : 1
        .word   36 : 1
        .word   37 : 1
        .word   38 : 1
        .word   39 : 1
        .word   40 : 1
        .align  2
```

Au final on observe la valeur des registres après une execution :

```
[christophe@e023 spim-6.5]$ spim
SPIM Version 6.5 of January 4, 2003
Copyright 1990-2003 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: ./trap.handler
(spim) load "../cross.s"
(spim) run
bonjour
\000(spim) print_all_regs
 PC     = 00000000     EPC    = 00000000     Cause  = 00000000     BadVAddr= 000
```

```
00000
 Status = 00000000     HI     = 00000000     LO     = 00000000
                              General Registers
R0  (r0) = 0          R8  (t0) = 0          R16 (s0) = 0          R24 (t8) = 0

R1  (at) = 268500992  R9  (t1) = 0          R17 (s1) = 0          R25 (t9) = 0

R2  (v0) = 10         R10 (t2) = 0          R18 (s2) = 0          R26 (k0) = 0

R3  (v1) = 268501028  R11 (t3) = 0          R19 (s3) = 0          R27 (k1) = 0

R4  (a0) = 268501152  R12 (t4) = 0          R20 (s4) = 0          R28 (gp) = 268
468224
R5  (a1) = 2147477140 R13 (t5) = 0          R21 (s5) = 0          R29 (sp) = 214
7477136
R6  (a2) = 2147477144 R14 (t6) = 0          R22 (s6) = 0          R30 (s8) = 0

R7  (a3) = 0          R15 (t7) = 0          R23 (s7) = 0          R31 (ra) = 419
4328
                                 Double Floating Point Registers
FP0  = 0.00000       FP8  = 0.00000       FP16 = 0.00000       FP24 = 0.00000

FP2  = 0.00000       FP10 = 0.00000       FP18 = 0.00000       FP26 = 0.00000

FP4  = 0.00000       FP12 = 0.00000       FP20 = 0.00000       FP28 = 0.00000

FP6  = 0.00000       FP14 = 0.00000       FP22 = 0.00000       FP30 = 0.00000

                                 Single Floating Point Registers
FP0  = 0.00000       FP8  = 0.00000       FP16 = 0.00000       FP24 = 0.00000

FP1  = 0.00000       FP9  = 0.00000       FP17 = 0.00000       FP25 = 0.00000

FP2  = 0.00000       FP10 = 0.00000       FP18 = 0.00000       FP26 = 0.00000

FP3  = 0.00000       FP11 = 0.00000       FP19 = 0.00000       FP27 = 0.00000

FP4  = 0.00000       FP12 = 0.00000       FP20 = 0.00000       FP28 = 0.00000

FP5  = 0.00000       FP13 = 0.00000       FP21 = 0.00000       FP29 = 0.00000

FP6  = 0.00000       FP14 = 0.00000       FP22 = 0.00000       FP30 = 0.00000

FP7  = 0.00000       FP15 = 0.00000       FP23 = 0.00000       FP31 = 0.00000



(spim) print $a0
Reg 4 = 0x100100a0 (268501152)
(spim) print $v0
Reg 2 = 0x0000000a (10)
(spim) print $3                          # REGARDER BIEN CES DERNIERES LIGNES
Reg 3 = 0x10010024 (268501028)
(spim) print 0x10010024
Data seg @ 0x10010024 (268501028) = 0x00000012 (18)
(spim) print 0x10010000
Data seg @ 0x10010000 (268500992) = 0x00000000 (0)
(spim) print 0x10010004
Data seg @ 0x10010004 (268500996) = 0x00000002 (2)
```

Exercice : dans le code assembleur, retouver la structure d'organisation du programme C. Peut etre que vous pouvez écrire le programme assembleur de manière plus concise ? Faites-le.

20/01/2003 Christophe Cérin