Master CCI

Langage machine

Corrigé du contrôle continu écrit 2009

Durée 1h30 heure, document autorisés, calculatrices et ordinateurs interdits

Prévoir environ 1 heure pour la première partie et 30 mn pour la deuxième partie.

1 Base 2 et conversion asm → C

Les exercices classiques consistent à partir d'un programme C normal, à le convertir en "C expansé" dans lequel

– toutes les variables temporaires nécessaires sont déclarées et

– toutes les constructions algorithmiques sont converties en **goto** et **if ... goto** puis à le traduire en langage d'assemblage.

Il s'agit ici d'effectuer le travail inverse : remonter du langage d'assemblage à du "C expansé", puis de ce dernier à un programme C classique.

On considère le fragment de code en langage d'assemblage suivant.

```
        .text
        .global calcul
calcul:
debut:  ldr r2,= x                  @ r2 = &x;
        ldr r0,[r2]                 @ r0 = *r2;
        cmp r0,#0                   @ if (r0 >= 0) goto Etiq2;
        bge Etiq2

Etiq1:  mvn r1,r0                   @ r1 = ~r0;
        add r1,r1,#1                @ r1 ++;        ou r1 = r1 + 1;
        bal Etiq3                   @ goto Etiq3

Etiq2:  mov r1, r0                  @ r1 = r0;

Etiq3:  ldr r3,= res                @ r3 = &res;
        str r1,[r3]                 @ *r3 = r1;
        mov  pc,lr
```

```
        .data
x:      .word   5
format:.asciz   "valabs(x) = %d\012"

        .bss
res:    .skip   4
```

Mvn (move not) calcule le complément à 1 de son opérande ($r1_i = \overline{r0_i} = 1 - r0_i$).

En langage C, l'opérateur de complément à 1 est noté **~**.

L'instruction **mov pc,lr** est un branchement de retour à la procédure qui a appelé **calcul**.

Question a : On considère une machine dans laquelle les entiers sont représentés sur 6 bits. Quels entiers naturels et relatifs (écrits en base 10) s'écrivent ainsi en hexadécimal :

1. 0x15 naturel : 21 relatif : +21
2. 0x35 naturel : 53 relatif : -11 (-32+21)
3. 0x2f naturel : 47 relatif : -17 (-32+15)

Question b : Quel(s) indicateur(s) teste la condition **ge** (dans le branchement conditionnel) ? En déduire de quel type (long int ou unsigned long int) doit être déclarée la variable x dans le programme C.
Bge teste la condition N identique à V (N=0 et V=0 ou N=1 et V=1)
Bge teste donc le signe d'un entier relatif, donc on peut en déduire que x n'est pas de type unsigned.

Question c : On suppose que lors de l'exécution les sections débutent respectivement aux adresses 0x1000 (text), 0x2000 (data) et 0x3000 (bss). Donner le contenu des registres r0 à r3 en fin d'exécution.
La valeur initiale de x (5) étant positive, r1 sera initialisé par l'instruction mov r1,r0. D'où r1 = r0 = 5. R2 contient l'adresse de x, sotcké au début de la section data, d'où r2 = 0x2000. R3 contient l'adresse de res, stocké au début de la section bss, d'où r3 = 0x3000. Si res avait été déclaré avec une valeur initiale, il aurait été stocké dans data et nous aurions r3 = 0x2004.

Question d : Donner pour chacun des registres (r0, r1, r2 et r3) utilisés la déclaration en C de la variable temporaire correspondante.
long int r0, r1, *r2, *r3 ;

Question e : Donner pour chacune des instructions en langage d'assemblage une instruction équivalente en langage C .

Question f : Donner un programme C équivalent ne contenant ni opérateur ~, ni goto. Quel calcul arithmétique réalise ce programme ?

```
/* declarations identiques */

/* Calcul de |x| : valeur absolue de x */
if (x <0) {
   res = -x;
} else {
   res = x
}
```

2 Traduction C → asm

Le corps de la procédure main suivante échange les contenus des deux variables a et b. Rappel pour la traduction en langage d'assemblage : en l'absence d'attribut register dans la déclaration en C, la variable doit être stockée en mémoire.

Question g : Traduire les déclarations de a, b et pta.

Question h : Traduire individuellement en langage d'assemblage les instructions 1 2, 3 et 4 de main. Votre traduction des trois dernières instructions doit résister à une permutation des contenus des pointeurs (short int *pta = &b suivi de ptb = &a).

```
#include <stdio.h>

/* variables globales */
/* hors main          */
                            @                .data
short int a = 13;           @ a:             .short  13  @ ou .half 13
                            @                .balign 4
short int *pta = &a;        @ pta:           .word a
                            @
short int b = 25;           @ b:             .short 25

void main (void)
{
    register short temp;                  @ temp : r1
    register short int *ptb;              @ ptb  : r2
                                          @ temporaires adr : r4 val : r3
    printf ("a = %d, b= %d\n",a,b);
    ptb = & b;                            @ ldr    r2,= b
```

```
    temp = *ptb;                            @ ldrsh  r1, [r2]

    *ptb = *pta;  /* *ptb = **&pta */       @ ldr    r4,= pta
                                            @ ldr    r4, [r4]
                                            @ ldrsh  r3, [r4]
                                            @ strh   r3, [r2]

    *pta = temp;  /* **&pta = temp */       @ ldr    r4,=pta
                                            @ ldr    r4,[r4]
                                            @ strh   r1, [r4]
    printf ("a = %d, b= %d\n",a,b);
}
```