DESS CCI : examen Langage Machine, Décembre 2005

Deux heures, tous documents et calculatrices autorisés. Ordinateurs (PC) interdits.

Les deux premières des quatre questions de la deuxième partie peuvent être traitées dans n'importe quel ordre.

La première partie est une petite question indépendante simple sur les constructeurs algorithmiques qui ne devrait pas prendre plus d'un quart du temps.

1 Constructions algorithmiques

Traduire en langage d'assemblage les déclarations de variables et les instructions C de l'extrait de programme C suivant :

```
/* Declarations */
short int t[7] = {2, 4, 5, 7, -1};
int i = 2;
int j = 3;

/* ... */

/* instructions */
if ((t[i] >= 2) && (t[j] < 4))
  {
  j = j + 1;
  }
i = i * 8;
```

2 Pointeurs et procédures

2.1 Présentation du programme C

On considère un programme C, composé de trois fichiers (x5py.c, x5py.h et ptr.c). Le fichier x5py.h définit le prototype des deux fonctions x5py et xplusy et donne le nom fonc à ce type de fonctions.

On supposera que lors de l'exécution les sections data et bss débutent respectivement aux adresses 0x1000 (data) et 0x2000 (bss).

```
/*************************************************************/
/* Contenu du fichier x5py.h                                 */
/*************************************************************/
```

```
/* Typedef permet de nommer fonc le type construit suivant :   */
/* fonction                                                    */
/*   + a deux paramètres entiers naturels courts               */
/*   + retournant un entier naturel court                      */

typedef unsigned short int fonc (unsigned short int, unsigned short int);

/* extern : déclare uniquement le prototype, pas la fonction        */
/*          ==> définit simplement la manière d'appeler la fonction */

extern fonc x5py;
extern fonc xplusy;

extern void appeler();
```

Les deux fonctions sont définies dans le fichier x5py.c, dont voici un extrait ci-dessous.

```
/*************************************************************/
/* Extrait du fichier x5py.c */
/*************************************************************/

#include "x5py.h"

unsigned short int a=4;
unsigned short int b=5;
unsigned short int c;

unsigned short int x5py (unsigned short int x, unsigned short int y)
{
return (...);     /* Retrouvez l'expression C */
                  /* utilisée dans ce return  */
}

unsigned short int xplusy (unsigned short int x, unsigned short int y)
{
return (x+y);
}

void test_x5py()
{
...               /* On ne vous donne pas le corps de test_x5py */
                  /* Retrouvez-le à partir de test_x5py.s       */
}

void main  ()
```

```
{
test_x5py();
appeler ();
printf ("%d <-- %d %d \n", c, a, b);
}
```

Dans le fichier ptr.c, l'une des fonctions est appelée directement, et l'autre via un pointeur de fonction (type fonc *).

```
/*************************************************************/
/* Extrait du fichier ptr.c */
/*************************************************************/

#include "x5py.h"

extern unsigned short int d;     /* précise à l'avance le type de d */
                                 /* pour la déclaration de ps       */

unsigned short int x;
unsigned short int *ps = &d;     /* pointeur initialisé */
unsigned short int d=4;          /* vraie déclaration de d */
unsigned short int e=5;
unsigned short int f;
fonc *pointe_fonc = &xplusy;   /* pointeur pour le stockage d'une adresse */
                               /* initialisé à l'étiquette d'une fonction */

void appel_via_pointeur ()
{
d ++;
f = (*pointe_fonc) (d, 9*e);    /* A traduire */
                                /* appelle la fonction pointée */
e--;
}

void appel ()
{
appel_via_pointeur(); /* --> exécutera xplusy */
pointe_fonc = &x5py;
appel_via_pointeur(); /* --> exécutera x5py */
}
```

2.2 Pointeurs

On rappelle que traduction en langage d'assemblage de la déclaration d'un pointeur est indépendante du type d'objet pointé. L'unique spécificité des pointeurs de

fonctions est que l'étiquette utilisée comme valeur initiale est définie dans text au lieu de data ou bss.

Question a : Traduire en langage d'assemblage les déclarations de a, ps, b, c et pointe_fonc.

Question b : Donner l'adresse (à la laquelle sera stocké le premier des octets) de chacune des variables (a, b, ... pointe_fonc).

Question c : Traduire en langage d'assemblage l'instruction **f = *ps ;**.

Question d : Supposons maintenant que ps pointe sur un élément d'un tableau de fonctions. Traduire en langage d'assemblage l'instruction **ps++ ;**.

3 Questions sur les procédures

3.1 Appel ordinaire

Le programmeur a traduit manuellement le fichier x5py.c en langage d'assemblage dans le fichier x5py.s.

```
/*************************************************************/
/* Extraits du fichier x5py.s */
/*************************************************************/

     .global test_x5py
test_x5py: @ prologue de la fonction et sauvegarde des registres sont omis

     @ debut du corps de test_x5py
     ldr     r0,=a                              @ instr 1
     ldrh    r0, [r0]                           @ instr 2
     add     r0,r0, #1

     ldr     r1,=b
     ldrh    r1, [r1]
     add     r1, r1, #2

     mov     lr, pc
     ldr     pc, =x5py

     ldr     r1, =c
     strh    r0, [r1]                           @ instr 10
     @ fin du corps de test_x5py
```

```
    @ epilogue de la fonction et restauration des registres sont omis
    mov     pc, lr

    .global x5py
x5py:   @ prologue de la fonction et sauvegarde des registres sont omis

    @ debut du corps de x5py
    add     r4, r0, r0, LSL #2   @ parametres x et y  dans r0 et r1
    add     r0, r4, r1           @ résultat dans r0
    @ fin du corps de test_x5py

    @ epilogue de la fonction et restauration des registres sont omis
    mov     pc, lr
    .ltorg
```

Question a : Retrouver comment s'écrit en C le corps de test_x5py.

Question b : Existe-t-il une instruction ARM qui pourrait remplacer la séquence **mov lr, pc ; ldr pc,=x5py** ?

Question c : Expliquer en quelques mots la différence entre un branchement relatif et un branchement absolu.

Question d : Quel calcul réalise l'instruction add r4,r0,r0, LSL #2 ?. Retrouver comment s'écrivait en C la fonction x5py.

Question e : Quelle est la valeur de r0 après l'exécution des instructions 1, 2 et 10 du corps de test_x5py ?

3.2 Transformation de x5py en procédure

Suite à un malentendu lors de la phase de spécification, x5py a été programmée comme une fonction à deux arguments retournant un entier court de type **unsigned short int x5py** (...), alors qu'elle devait être programmée comme une procédure de type **void x5py** (...).

Question : Réécrire en C x5py comme une procédure, ainsi que l'instruction d'appel de x5py dans test_x5py. Comment est modifié le corps de x5py dans x5py.s ?

3.3 Appel via un pointeur

Question : traduire en langage d'assemblage l'instruction **f = (*pointe_fonc) (d, 9*e) ;**