DESS CCI : examen Langage Machine, Décembre 2004

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

La convention d'appel de procédure est celle de gcc : les quatre premiers paramètres sont stockés dans les registres r0 à r3 et le résultat des fonctions est stocké dans r0 à la place du premier paramètre.

Q1	50mn	Q2	70mn

1 Entiers, décalages et boucles

Question a : Donner une séquence C équivalente à la boucle de nb_uns du programme ci-dessous, sans instruction for (en utilisant if et goto).

Question b : Traduire en langage d'assemblage ARM la boucle for de nb_uns. On ne vous demande ni le prologue, ni l'épilogue (sauvegarde, restauration de registres, allocation de mémoire dans la pile) de la fonction.

```
unsigned short int a,b,c;

unsigned int nb_uns (short int x)
{
/* une methode originale de comptage du nombre de bits à 1              */
/* dans la representation en binaire de x                               */
/* important pour les decalages : moins_un_si_bit est de type signe     */
/* Les opérateurs << et >> sont les decalages a gauche et droite        */
/* Exemple : y = x << 3 : y <- x decale de 3 bits a gauche              */

register int res;             /* choisir r1 pour traduire en ARM */
register int i;               /* choisir r2 */
register int moins_un_si_bit;   /* choisir r3 */

res = 0;
for (i=0; i< 16; i++)
  {
  moins_un_si_bit = x << (16 + i);
  moins_un_si_bit =  moins_un_si_bit >> 31;
  res = res + moins_un_si_bit;
  }
return (-res);
}
```

```
void main ()
{
register unsigned int r5; /* r5 stockée dans le registre r5  */
b = nb_uns(a);
c = nb_uns(b);
b = nb_uns (nb_uns (nb_uns (c)));.
}
```

Question b : Comment représente-t-on -1 en binaire sur 32 bits ?

Question c : Expliquer en quelques lignes le principe de la méthode de comptage du nombre de uns utilisée ici.

Question d : Traduire en langage d'assemblage ARM les deux appels à la procédure nb_uns présents dans le corps de main.

Question e : Quel est l'effet de la séquence de code ARM suivante ? Ecrire une instruction C correspondant à cette séquence.

```
mov r0, #5
mov lr, pc
ldr pc, =nb_uns
mov r5,r0
```

Questione e : Expliquer en quelques lignes comment traduire en langage d'assemblage ARM la dernière instruction de main. Cette dernière instruction rend-elle le programme récursif ?.

2 Pointeurs et tableaux

On considère le programme C suivant.

```
short int x = 4;
short int y;
short int z;
short int u = 32;

void xplusplus ()
{
x++;
}

void yplusplus ()
{
y++;
```

```
}

void zplusplus ()
{
z++;
}

void uplusplus ()
{
u++;
}

/* Un type procedure sans parametre */
typedef void pproc ();

/* un tableau de 4 pointeurs de short */
short int *tab_ptrshort [4] = {&x,&y,&z,&u};

/* Un tableau de pointeurs de procedures sans parametre : */
/* L'adresse d'une fonction est l'adresse de sa première instruction */
pproc *tab_proc [4] = {&xplusplus,&yplusplus,&zplusplus,&uplusplus};

int main ()
{
register int i; /* a stocker dans r4 */

/* Un pointeur d'entiers courts */
register short int *ps; /* a stocker dans r5 */

/* Un pointeur de procedures sans parametre */
register pproc *ptr_proc;

i = 0;
while (i<4)
  {
  ps = tab_ptrshort[i];
  *ps = *ps + 1;
  ps ++;
  i++;
  }



/* Deuxieme boucle page suivante */
```

```
i = 0;
ptr_proc = &xplusplus;
while (i<4)
  {
  ptr_proc = tab_proc[i];
  (*ptr_proc)();              /* appel de la procédure pointée par ptr_proc */
  ptr_proc ++;
  i++;
  }
}
```

2.1 Déclarations

Question : traduire en langage d'assemblage ARM la procédure xplusplus (inutile de détailler le prologue et l'épilogue de la procédure) ainsi que la déclaration des deux tableaux.

2.2 Tableau de fonctions

Question : traduire en langage d'assemblage ARM l'instruction **ptr_proc = tab_proc[i] ;**.

2.3 Pointeur de fonction

Question : traduire en langage d'assemblage ARM l'instruction **ptr_proc = &xplusplus ;**.

2.4 Appel de fonction pointée

Question : traduire en langage d'assemblage ARM l'instruction **(*ptr_proc) () ;**.
La relecture de la question 1e peut vous être utile.

2.5 Indice versus pointeur

Question : écrire une boucle C while équivalente à la première boucle, n'utilisant que ps (et pas i) comme variable de boucle.