ENSSAT EII1

DS de Systèmes à microprocesseurs

Documents autorisés - Durée 1 heure

Vendredi 21 avril 2006

1 Développement de code assembleur pour le processeur ARM (···/4pt)

Soient X et Y deux vecteurs composés respectivement de N nombres complexes. Les différents éléments sont stockés dans le vecteur en alternant partie réelle et partie imaginaire de chaque élément :

$$[\mathbf{Re}(X_0), \mathbf{Im}(X_0), \mathbf{Re}(X_1), \mathbf{Im}(X_1) \ldots, \mathbf{Re}(X_N), \mathbf{Im}(X_N)]$$

Soit le vecteur Z correspondant à la différence de ces deux vecteurs :

$$Z = X - Y$$

L'objectif de la routine (`Norm2Diff`) est de calculer le carré de la norme de ce vecteur Z

$$||Z||^2 = \sum_i |Z_i|^2 \tag{1}$$

Les conventions suivantes sont utilisées pour l'interface avec cette routine :

- Entrée
  - `r0` : adresse du vecteur X
  - `r1` : adresse du vecteur Y
  - `r2` : nombre d'éléments complexes (N) des vecteurs
- Sortie
  - `r0` : norme carrée du vecteur différence.

Nous disposons du code suivant pour tester cette routine (`Norm2Diff`) :

```
#define N       10    /* Nombre d'éléments des vecteurs */

int X[2*N] = {....};
int Y[2*N] = {....};

extern???? Norm2Diff(????);


int main(void)
{

     Norm2Diff(????);

   while(1);
}
```

Q.1.1) Donner le prototype C de la routine `Norm2Diff` et l'appel en C de cette routine. Vous utiliserez les conventions de passage de paramètres associées au processeur ARM.

Q.1.2) Écrire le code C de la routine `Norm2Diff`.

Q.1.3) Écrire le code assembleur (avec commentaires) de cette routine en utilisant les conventions définies ci-dessus.

2 Analyse de code (···/6pt)

Nous considérons une fonction pour laquelle nous allons analyser le code et étudier le contenu de certains registres. Le code assembleur de cette fonction est le suivant

```
main        mov     r0,#3
            stmfa   r13!,{r0}
            bl      fact
            sub     r13,r13,#4
exit        nop
fact        stmfa   r13!,{r1,r2,r14}
            ldr     r1,[r13,#-12]
            cmp     r1,#0
            beq     un
            sub     r2,r1,#1
            stmfa   r13!,{r2}
            bl      fact
            sub     r13,r13,#4
            mul     r0,r1,r0
            b       finfact
un          mov     r0,#1
finfact     ldmfa   r13!,{r1,r2,r14}
            mov     r15,r14
```

La zone mémoire associée au code de cette fonction est fournie en annexe 1. La première colonne représente l'adresse des éléments de la mémoire. La seconde colonne représente le contenu de la mémoire. La troisième colonne représente l'instruction associée à chaque case de la mémoire.
Cette fonction comporte une certaine récursivité et par conséquent certaines instructions sont exécutées plusieurs fois (ce qui explique la présence de plusieurs colonnes E_i pour un même registre et pour la même ligne d'instruction sur le tableau de remplissage de l'annexe 1).

La zone mémoire associée à la pile est fournie en annexe 2. La première colonne représente l'adresse des éléments de la mémoire. La seconde colonne représente le contenu de la mémoire.

Q.2.4) Dans le code opératoire sur 32 bits d'une instruction de branchement, les bits 24 à 31 donnent le type de branchement et la condition éventuelle, tandis que les bits 0 à 23 sont réservés à la valeur du décalage en mémoire programme.
Expliquer la valeur de ce décalage pour la troisième instruction de la fonction étudiée (`bl fact`).

Q.2.5) Compléter à la fois

i) l'annexe 1 pour connaître le contenu des registres SP(R13), LR(R14), R0, R1 et R2 (ne remplir que les cases blanches, les cases foncées indiquent que le registre n'est pas concerné par l'instruction en cours).

ii) l'annexe 2 pour représenter l'évolution du contenu de la pile.

Q.2.6) Quel est l'objectif de cette fonction et dans le cas présent quelle est la valeur renvoyée ?