IUT 'A' Paul SABATIER
Dpt INFORMATIQUE
S2

Mardi 12 Juin 2012

ASR => Architecture des ordinateurs : Ass2

Langage d'assemblage des microprocesseurs ARM

Durée : 2 heures
Documents : autorisés

I - Soit les types Resultat, NomDuMois et DateJour définis en C de la façon suivante :

```
typedef enum
     { INF=-1, EGAL=0, SUP=1 } Resultat ;
typedef char * NomDuMois ;
typedef struct
     {
      int an ;
      NomDuMois mois ;
      int jour
     } DateJour ;
```

-1- donner les déclarations correspondantes en langage d'assemblage.

Soit la spécification de la fonction compDate :

-- renvoie la valeur EGAL si deux dates fournies en paramètre sont identiques
-- renvoie la valeur INF si d1 < d2 et SUP si d1 > d2.

fonction compDate (**entrée** d1 <DateJour>**entrée** d2 <DateJour>) **retourne** <Resultat>;

- On suppose dans un premier temps un passage des paramètres d'entrée par valeur :
  -2- Expliquer sur l'exemple ci-après comment l'appelant passe les 2 paramètres : on dessinera la pile et on indiquera la valeur des registres juste avant l'appel du sous-programme lorsque d1={10,"mai",1981} et d2={6,"mai",2012}. Quand on ne connaît pas explicitement la valeur d'un pointeur (! il n'y a pas de piège) on utilisera la notation « @ ???? »

  -3- Donner le schéma de la pile après que l'appelé a exécuté :

  strmfd sp!, {r0-r3}

  Quel est le rôle de cette instruction et quel avantage le programmeur en tire-t-il ?

- On suppose maintenant un passage des paramètres d'entrée par adresse :
  -4- Quel en est l'intérêt ?
  -5- Reprendre en les adaptant éventuellement les questions **2** et **3**.

Remarque : On rappelle que les entiers, les entiers relatifs et les pointeurs sont représentés sur 1 mot mémoire, les caractères sont représentés sur un octet..

II – On considère la séquence de code qui effectue la somme de la partie commune (**en caractères gras dans l'exemple suivant**) de deux valeurs naturelles représentées par des tableaux de caractères.

	7	6	5	4	3	2	1	0	rang
			'1'	'3'	**'8'**	**'9'**	**'2'**	**'1'**	t2
+					**'1'**	**'6'**	**'3'**	**'9'**	t1
					'0'	**'5'**	**'6'**	**'0'**	t1

Traduire la séquence suivante en langage d'assemblage ARM. On suppose que t1 et t2 sont des tableaux en mémoire de 5000 caractères (au maximum), les variables n et m (entier relatif) sont représentées sur un mot en mémoire, la variable i (entier relatif) est le registre R0. On utilisera R1 (respectivement R2) pour l'adresse de t1[0] (respectivement l'adresse de t2[0]). On utilisera R3 (respectivement R4) pour t1[i] (respectivement t2[i]). La variable résultat sera synthétisée par R5 et la retenue par R6. On accédera aux variables n et m via les registres R7 et R8.

```
i ← 0 ;
retenue ← 0 ;
Tantque (i < n) et (i < m) faire
     resultat ← t1[i] + t2[i] - '0' + retenue;
     si resultat > '9'
          alors
               t1[i] ← resultat - 10 ;
               retenue ← 1 ;
          sinon
               t1[i] ← resultat ;
               retenue ← 0 ;
     finsi ;
     i ← i + 1 ;
fintantque ;
```

On donnera la séquence complète : définition des constantes, des variables (t1, t2, n et m) ainsi que de leurs pointeurs relais, l'algorithme détaillé et la traduction en langage d'assemblage. On donnera 2 (**deux**) versions de la séquence encadrée : la première sans exécution conditionnelle, la seconde utilisant cette technique.

III- Soit la nouvelle définition du type ChaineDyn :

```
typedef struct
     {
      char *ptrCar ;
      int nbCar ;
     } ChaineDyn ;
```

Donner la définition de la structure, le schéma de la pile, le rôle des registres, l'algorithme détaillé et la traduction en langage d'assemblage du sous-programme longueurChaine.

Remarque : Consigne non respectée entraîne question non corrigée !!!
Barème probable : 8,7, 5