M2P CCI : examen Langage Machine, Novembre 2015

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

Table des matières

Conventions communes à toutes les questions

Toutes les variables locales des fonctions et procédures sont stockées dans des registres (le numéro de registre à utiliser est indiqué en commentaire). Seules les variables déclarées à l'extérieur des fonctions et sans l'attribut register sont stockées en mémoire.

La convention d'appel des fonctions et procédures est la suivante :
— Tous les paramètres sont passés par les registres et non en mémoire
— Le premier paramètre est passé dans le registre r0
— Le deuxième paramètre est passé dans le registre r1
— Le résultat des fonctions est retourné dans le registre r0 à la place du premier paramètre

La taille d'un char est de 8 bits, celle d'un int ou unsigned int est de 32 bits.

1 Introduction sans question (5 mn)

En utilisant le format %d, les fonctions de type scanf permet de convertir une chaîne de caractères[1] représentant l'écriture d'un entier en décimal en la valeur entière correspondante.

L'objectif est de traduire en langage d'assemblage un programme C qui effectue un travail analogue et génère par exemple l'entier 1234 à partir de la chaîne de caractères

1. sscanf pour une chaîne stockée dans un tableau, scanf et fscanf pour une chaîne lue au clavier ou dans un fichier

"1234".

Outre main, le programme comprend 3 routines affectuant chacune une tâche :

1. **char10_to_val** prend un (code ASCII de) caractère chiffre décimal en paramètre et calcule sa valeur entière (calcule 5 à partir de '5').
2. **chaine_to_chiffres** applique char10_to_val à chaque caractère de la chaîne et remplit un tableau d'entiers avec les valeurs des chiffres (de "1234" à {1,2,3,4}).
3. **string_to_int** utilise chaine_to_chiffres pour convertir la chaîne en valeur entière suivant la formule $x = \sum_{i=0}^{n-1} x_i 10^i$ avec $x_i \in [0..9]$.

2 String_to_int : boucle for et tableau (1h)

Le programme principal main.c qui convertit deux chaînes, est donné en annexe :

```
mandelbrot> armrun string_to_int
Chaine : 43  -> 43
Chaine : 26543  -> 26543
```

Et voici le fichier string_to_int.c :

```
#include "chaine_to_chiffres.h"
#include <stdio.h>

#define MAX 11
unsigned int valeurs[MAX];

/* Conversion d'une chaine en valeur entiere (s -> valentier) */
void string_to_int (char *s, unsigned int *valentier)
{
   register unsigned int nb_val;   /* a stocker dans r4 */
   register unsigned int somme;    /* a stocker dans r5 */
   register int i;                 /* a stocker dans r6 */

   somme = 0;
   nb_val = chaine_to_chiffres (s,valeurs);
   for (i=0;i<nb_val;i++) {
     somme = 10*somme + valeurs[i];
   }
   *valentier = somme;
}
```

et un squelette de traduction à compléter :

```
                .global   string_to_int
MAX=11
                /* ajouter sections bss et/ou data */

                .text
string_to_int:
prologue:       stmfd sp!,{r4-r9,lr}     @ sauvegarde des registres en pile
                mov   r9,r1              @ copie de parametre reçu valentier
                                         @ (L'appel de chaine_to_chiffres
                                         @ detruira le contenu de r1)
                ldr   r7,= valeurs
corps:          ...

epilogue:       ldmfd sp!,{r4-r9,lr}     @ restauration registres depuis pile
                mov   pc,lr
                .ltorg
```

2.1 Sections data /bss (15mn)

Traduire en langage d'assemblage la déclaration du tableau valeurs.

Traduire en langage d'assemblage la déclaration des variables ch1, val1, ch2 et val2 du fichier main.c.

Traduire en langage d'assemblage (section text) l'affectation **v = val1** de main.c. Vous supposerez que v est stockée dans le registre r4.

Imaginons que l'on ajoute dans le corps de main l'affection suivante : **val1 = 35**. **Comment** la traduirait-on en langage d'assemblage (en veillant à ne pas faire d'accès mémoire inutile) ?

2.2 Boucle for (35mn)

Traduire en langage d'assemblage l'affectation
somme = 10*somme + valeurs[i]

Remplacer la boucle for par une séquence d'instructions C équivalente utilisant une boucle while.

Traduire en langage d'assemblage la boucle for (indiquer par un commentaire l'endroit où placer la traduction de l'affectation précédente de la variable somme). Il est conseillé (mais pas obligatoire) de mettre en commentaires la forme C if ... goto équivalente.

2.3 Appel de fonction (10mn)

L'endroit où se trouvent les paramètres s et valentier est défini par la convention d'appel rappelée en début de sujet.

Traduire en langage d'assemblage l'appel **string_to_int (ch1,&val1)** de main.c.

3 Char10_to_val : if, paramètre adresse (30 mn)

Voici le code C de la procédure char10_to_val et un squelette de traduction à compléter :

```
#include "char10_to_val.h"
#include <stdio.h>

/* Conversion d'un caractere chiffre -> valeur du chiffre ('4' -> 4) */
/* Convention d'appel : chiffre dans r0, pvalchiffe dans r1,         */
/*                      resultat dans r0 à la place de chval         */
/* Retourne 0 en cas d'erreur (pas un caractere dans ['0' ... '9']   */
/* Retourne 1 si la conversion est reussie                           */

int char10_to_val (char chiffre, unsigned int *pvalchiffre)
{
  register unsigned int decimal;        /* a stocker dans le registre r4 */
  register int r;                       /* a stocker dans le registre r5 */

  decimal = chiffre;
  if (decimal >= '0' && decimal <= '9') {
    decimal   = decimal - '0';
    *pvalchiffre = decimal;
    r = 1;
  } else {
    r = 0;
  }
  return r;
}
```

```
                    .global      char10_to_val
                    .text
char10_to_val:
prologue:           stmfd     sp!, {r4,r5}     @ sauvegarde des registres en pile

corps:              ...
```

```
si:                     ...

alors:                  ...

sinon:                  ...

finsi:
epilogue:               ...                             @ return r
                        ldmfd       sp!,{r4,r5}
                        mov  pc,lr
```

Avec une condition composée de type ET, il suffit que l'un ou l'autre des termes de la condition soit faux pour que l'on saute dans la branche sinon du if. La sémantique du C indique que l'opérateur && est de type "et puis" : la deuxième partie de la condition n'est évaluée que si la première partie de la condition est vérifiée.

La traduction d'un if avec un condition de type ET suit le même principe que celle d'un if avec une condition simple, mais avec une séquence d'autant couples test+branchement si faux que de termes dans la condition composée.

Traduire en langage d'assemblage les instructions du corps de char10_to_val en décomposant au préalable la construction if en C équivalent avec if ... goto. Vous pouvez écrire deux blocs de code séparés ou bien placer la forme if ... goto en commentaire des instructions en langage d'assemblage.

4 Chaine_to_chiffres : paramètres, pointeurs (25mn)

Voici le contenu du fichier chaine_to_chiffres.c et un squelette de traduction :

```
#include "char10_to_val.h"

/* 1er parametre  : une chaine de caracteres parmi ['0'... '9']            */
/* 2eme parametre : tableau chiffre a remplir avec les valeurs des chiffres */
/* La taille du tableau doit etre au moins egale a la longueur de la chaine s   */
/* Valeur retournee  : nombre de valeurs stockees dans chiffres                 */

/* Convention d'appel :                                                        */
/*   s dans r0, chiffres dans r1, valeur de retour dans r0 a la place de s     */

unsigned int chaine_to_chiffres (char *s, unsigned int *chiffres)
{
  register unsigned *ptri;              /* a stocker dans r4 */
  register char *ptrc;                  /* a stocker dans r5 */
  register unsigned int longueur;       /* a stocker dans r6 */
```

```
  register int ok;                    /* a stocker dans r7 */

  ptri = chiffres;
  ptrc = s;
  longueur = 0;

  /* do...while : comme while mais le corps est executé au moins 1 fois */

  do {
     ok = char10_to_val(*ptrc, ptri);
     if (ok != 0) {
        longueur = longueur + 1;
        ptrc = ptrc + 1;
        ptri = ptri + 1;
        }
  } while (ok != 0);
  return longueur;
}
```

```
                        .global chaine_to_chiffres
                        .text
chaine_to_chiffres:
prologue:               stmfd   sp!, {...}   @ sauvegarde des registres dans la pile
corps:                  ...

epilogue:               ...                  @ return longueur
                        ldmfd   sp!,{...}    @ restauration des registres
                        mov     pc,lr
```

Traduire en langage d'assemblage l'affectation **ptri = chiffres**.

Traduire en langage d'assemblage l'affectation **ptri = ptri + 1**.

Traduire en langage d'assemblage l'affectation
ok = char10_to_val(*ptrc, ptri)

Donner la liste des registres à sauvegarder et restaurer (dans stmfd et ldmfd). Un registre doit impérativement être sauvegardé et restauré, faute de quoi l'exécution du programme ne sortira jamais de la fonction chaine_to_chiffres. **Quel est ce registre ?** (expliquer pourquoi).

5 Annexe 1 : main.c

```
#include <stdio.h>
#include "string_to_int.h"
```

```
char ch1 [ ] = "43";  /* La taille du tableau est deduite de celle de la chaine */
unsigned int val1;
char ch2 [ ] = "26543";
unsigned int val2=7;

int main (void)
{
  register unsigned int v;

  string_to_int (ch1,&val1);
  v = val1;
  printf ("Chaine : %s  -> %u \n",ch1,v);
  string_to_int (ch2,&val2);
  printf ("Chaine : %s  -> %u \n",ch2,val2);

  return 0;
}
```