DESS CCI : examen Langage Machine, Septembre 2005

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

1 Introduction

On s'intéresse à la traduction en langage d'assemblage d'un programme C qui calcule les puissances de 2 ainsi que la suite de Syracuse et remplit un tableau de résultats sous forme de triplets. Ce programme est annexé en fin de document.

2 Forme C équivalente avec goto

Question : Donner un code C équivalent à la boucle do de la procédure syracuse_puis2 dans lequel les constructeurs **do** et **if** sont traduits en branchements **goto** et **if**...**goto**.

3 Calcul en binaire

3.1 Décalages

Appliqué à (la représentation en binaire d') un entier naturel, l'opérateur de décalage logique à droite (respectivement à gauche), noté LSR (respectivement LSL), élimine le bit de poids faible (respectivement fort) et ajoute un bit à 0 en poids fort (respectivement faible). A titre d'exemple sur 4 bits, $LSR_{1bit}(1010) = 0101$.

L'opérateur de décalage arithmétique à droite, noté ASR, élimine le bit de poids faible et duplique le bit de poids fort ($ASR_{1bit}(1010) = 1101$).

Question : Expliquer brièvement le principe de la multiplication par 20 utilisé dans la séquence mov ; add suivante. Expliquer comment réaliser une multiplication par 24 ou par 12.

```
@ multiplication par 20
        mov r1, r0, LSL #2      @ decal_gauche(r0) de 2 bits
        add r0, r1, r0 LSL #4   @ decal_gauche(r0) de 4 bits + r1
```

Question : Expliquer la raison de la présence dans les jeux d'instructions des processeurs des deux variantes de décalage à droite (alors qu'il n'existe qu'un seul opérateur de décalage à gauche).

3.2 Et bit à bit

Voici un exemple d'utilisation des opérateurs ET et OU bit à bit.

```
Opération ET (|,AND) et OU (|, ORR) bit à bit entre deux entiers
Bit_j (resultat) == 1 ssi Bit_j(op1)==1  ET/OU_bien Bit_j(op2)==1

      00101001001110001        00101001001110001
&                           |
bb    1111111100000000   bb   1111111100000000
      -----------------        -----------------
      00101001000000000        1111111101110001
```

L'opérateur $\&_{bb}$ a la propriété suivante : x est pair si et seulement si $x\&_{bb}1 == 0$.

Question : Expliquer cette propriété.

4 Langage d'assemblage

Question : Traduire en langage d'assemblage la boucle do du programme C. Pour vous aider, voici un exemple de traduction de fragment de code C.

```
#define MAX_ITERATIONS 32

triplet exemple = {3, 8, 2};
triplet resultat[MAX_ITERATIONS];

void ecrire ()
{
exemple.indice = 3;
exemple.puissance2 = 8;
}
```

```
               .data
@ Constantes de position des champs dans la structure
               EXEMPLE_INDICE=0
               EXEMPLE_PUISSANCE2=4
               EXEMPLE_SYRACUSE=8

exemple:      .word     3
              .word     8
              .word     2

              .bss
resultat:     .skip     384

              .text
ecrire:       @ ... prologue de la procedure omis
              ldr       r0, =exemple
```

```
        mov       r1, #3
        str       r1, [r0, #EXEMPLE_INDICE]
        mov       r1, #8
        str       r1, [r0, #EXEMPLE_PUISSANCE2]
        @ epilogue et retour omis
        .ltorg
```

5 Le programme à traduire

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_ITERATIONS 32

/*****************************************************************/
/* Resultats : triplet <i, 2^i, syracuse(i,valeur_initiale)>     */
/*****************************************************************/

typedef struct s_triplet {
unsigned long indice;
unsigned long puissance2;
unsigned long syracuse;
} triplet;

triplet resultat[MAX_ITERATIONS];

unsigned long valinit;
unsigned nb_res;

/*****************************************************************/
/* Syracuse_puissance2                                           */
/*                                                               */
/* indice=0, puissance = 1; syracuse = valeur initiale           */
/* Répéter                                                       */
/*     resultat[indice] = <indice, puissance, syracuse>          */
/*     puissance = puissance * 2                                 */
/*     Si syracuse est pair                                      */
/*        syracuse = syracuse / 2                                */
/*     sinon                                                     */
/*        syracuse = 3 * syracuse + 1                            */
/*     indice = indice + 1                                       */
/* Tant que i<MAX_ITERATIONS et syracuse <> 1                    */
/* nb_res = indice                                               */
/*****************************************************************/
```

```
void afficher ()
{
int i;
printf ("Affichage\n");
for (i=0; i< nb_res; i++)
  printf ("i = %2lu     puis2(i) = %10lu   syracuse = %10lu\n",
           resultat[i].indice, resultat[i].puissance2, resultat[i].syracuse);
}

void syracuse_puis2 ()
{
register long i;
register long puis;
register long syra;

i=0;
puis = 1;
syra = valinit;
do {
  resultat[i].indice=i;
  resultat[i].puissance2=puis;
  resultat[i].syracuse=syra;
  puis = puis * 2;
  if ((syra % 2) == 0)                      /* if syra pair */
    syra = syra / 2;
  else
    syra = 3 * syra + 1;
  i++;
   }
while ((i<MAX_ITERATIONS) && (syra != 1));
nb_res = i;
}

int main (int argc, char *argv[])
{
if (sscanf (argv[1],"%lu",&valinit) != 1)
  {
  printf ("Erreur : %s non entier\n",argv[1]);
  printf ("Usage : syracuse valeur_entiere_initiale\n");
  exit (1);
  }
syracuse_puis2();  /* on peut en plus appeler afficher */
return 0;
}
```