Master CCI

Langage machine

Contrôle continu écrit 2020

Durée 1h30, documents autorisés, calculatrices et ordinateurs interdits

Les questions peuvent être traitées dans n'importe quel ordre (mais 2.2 et 3.1 portent sur la même portion de code) et sont d'importance égale.

Du point de vue timing, la partie base 2 est la plus susceptible de déraper : soyez attentif à ne pas perdre trop de temps dessus.

Table des matières

Conventions et contraintes

L'ordre dans lequel les variables sont déclarées doit être respecté dans la traduction en langage d'assemblage. Les variables déclarées sans attribut register **doivent** être stockées en mémoire (contenu mémoire mis à jour à chaque affectation).

Voici les types de variables entières utilisées dans le code des exercices. Par dérogation, considérez pour la traduction en langage d'assemblage le type char comme synonyme de uint8_t (entier naturel).

types d'entier relatif		taille	types d'entier naturel	
synonyme sur ARM	type	bits	type	synonyme sur ARM
	int8_t	8	uint8_t	unsigned char
short	int16_t	16	uint16_t	unsigned short
int	int32_t	32	uint32_t	unsigned int

Les constantes symboliques peuvent être définies par = en langage d'assemblage (TAILLE=2), après quoi elles peuvent être utilisées comme opérande dans une directive ou une instruction : (mov r0,#TAILLE).

Les directives de réservation de place acceptent des expressions simples composées d'entiers, de constantes entières symboliques et d'opérateurs arithmétiques de base : +,-,* (.skip 4*TAILLE).

1 Variables et accès mémoire (30mn)

1.1 Déclarations (10mn)

Traduire en langage d'assemblage les déclarations de variables suivantes :

```
uint8_t    x = 35;
uint8_t *pt2 = &y;
uint8_t    y;
register uint8_t *pt1;  // utiliser le registre r0
```

1.2 Accès aux variables (20mn)

Traduire en code ARM chacune des affectations suivantes :
- vous pouvez utiliser d'autres registres que r0 comme temporaires
- sans factorisation de code avec celui des autres affectations
- en supposant que d'autres instructions ont pu modifier les variables depuis leur déclaration : x et pt2 peuvent contenir autre chose que 35 et &y.

1. y=x+3 ;
2. pt1=&x ;
3. *pt1=2 ;
4. x=*pt2 ;

2 Constructions algorithmiques (25mn)

On considère des extraits d'un programme de conversion binaire.c (code complet détaillé en annexe), qui affiche la valeur de l'entier x en base 2. Contrairement à la méthode classique de conversion le programme détermine et affiche les chiffres binaires directement dans le bon ordre (chiffre des unités en dernier, affiché à droite).

Les délarations de variables dans main sont rappelées ci-dessous :

```
register int32_t val;      // utiliser r4
register char car;         // utiliser r5
register unsigned int c;   // utiliser r6
register uint32_t borne;   // utiliser r7
```

2.1 Traduction d'un if (10 mn)

Traduire en code ARM la construction if de la deuxième boucle :

```
if (val <0) {
   c=1;
} else {
   c=0;
}
```

2.2 Traduction d'une boucle (15mn)

Traduire en code ARM la deuxième boucle. L'ensemble du if et l'appel de printf seront remplacés chacun par un simple commentaire.

```
while (borne > 0) {
   // code du if
   car = 0x30+c;
   val = val*2;
   borne = 2*borne;
   // appel de printf
}
```

3 Représentation d'informations en binaire (35mn)

3.1 Fonctionnement du programme de conversion (section 2.2) (15mn)

Expliquer brièvement

1. pourquoi la valeur du chiffre courant est déterminée par la condition val<0 et pourquoi val est multiplié et non divisé par 2.
2. le critère d'arrêt de la deuxième boucle (section 2.2) et préciser si l'attribut unsigned (uint32_t et non int32_t) de la variable borne est important ou non
3. la raison de l'ajout de la constante 0x30

3.2 Addition binaire et indicateurs (20mn)

Compléter l'addition sur 4 bits ci-dessous en détaillant les retenues
Que valent les indicateurs Z,N,C et V (préciser brièvement la méthode de lecture de C et V)
D'après les indicateurs, le **résultat** apparent est-il **correct** ?

Ecrire en binaire le complément à 2 du premier opérande. Quel est la signification arithmétique du complémen,t à 2 ?

$$\begin{array}{rcccc} & 1 & 0 & 1 & 1 \\ + & 0 & 1 & 1 & 0 \\ \hline \end{array}$$

4 Annexe : source du programme de conversion en binaire

Pour donner un exemple d'exécution sans utiliser scanf, la variable x est initialisée avec une constante. La connaissance de cette constante ne peut évidement pas être utilisée pour simplifier le code de main.

```
int32_t x=0x39ab5675;

int main () {
   register int32_t val;      // utiliser r4
   register char car;         // utiliser r5
   register unsigned int c;   // utiliser r6
   register uint32_t borne;   // utiliser r7

   val = x;
   borne=1;

  if (val==0) {
     printf ("0");
  } else {

       while (borne >0 && val >0) {
         val = 2*val;
         borne = 2*borne;
       }

       while (borne > 0) {
         if (val <0) {
           c=1;
         } else {
           c=0;
         }
         car = 0x30+c;
         val = 2*val;
         borne = 2*borne;
         printf ("%c",car);
       }       // fin du while
   }           // fin du else
   printf ("\n");
   return 0;
}

mandelbrot> ./binaire
11100110101011010101011001110101
mandelbrot>
```