# Master CCI

# Langage machine

# Contrôle continu écrit 2017

**Durée 1h30, documents autorisés, calculatrices et ordinateurs interdits**

## Table des matières



## Conventions et contraintes

Attention : ne perdez pas de temps à analyser le principe des algorithmes et concentrez-vous sur leur traduction en langage d'assemblage ARM.

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.

Voici les types de variables entières utilisées dans le code des exercices.

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

# 1 Première partie : gestion des variables (45 mn)

## 1.1 Traduction des éclarations (20mn)

```
uint16_t a = 3;
int32_t x = 5;
uint16_t b = 16;
int32_t y = -2;
int32_t e = 120;
int32_t f = 24;
uint16_t *ptminab;
int32_t *pt32g = &x;
int32_t *pt32d = &y;
int32_t *ptminxy;
```

**Traduire** en langage d'assemblage les déclarations de variables ci-contre.

**A quelle adresse** sera stockée la variable y, en supposant que la section data débute à l'adresse 00018fd4 (justifier brièvement vôtre réponse) ?

## 1.2 Traductions des accès aux variables (25mn)

Voici deux fonctions C qui accèdent à ces variables et un squelette, à compléter, de leur traduction en langage d'assemblage ARM.
**Traduire** le corps de calcul1 et calcul2, excepté l'appel de printf.
**Que contiennent** les variables x et y à la fin de l'exécution de main ?

```
void calcul1 () {
  register int32_t regint32;  // dans registre r4
  register int32_t regvalf;   // dans registre r5
                              // registres r8 et suivants : temporaires
  regint32=e;
  regvalf=f;
  if (regvalf<regint32) {
    regint32=regvalf;
  }
  *pt32d = *pt32g + 1;
  printf ("minimum (e=%d, f=%d) = %d\n",e,f,regint32);
}

void calcul2 () {
   if (a<b) {
     ptminab=&a;
   } else   {
     ptminab=&b;
   }
   x= y + 6;
}

void main () {
  calcul1();
  calcul2();
}
```

```
            .text
calcul1:     stmfd      sp!,{r0-r5,r9,r10,lr}
             @ Utiliser les registres r0 à r5, r9 et r10 sauvegardés ci-dessus

             @  à compléter : corps à traduire

callprintf: adr      r0,format  @ code d'appel de printf
            ...                 @ détail du code omis  (utilise r0 à r3)
            bl      printf
            ldmfd   sp!,{r0-r5,r9,lr}        @ restauration
            mov      pc,lr

calcul2:    stmfd      sp!,{r0-r7}           @ sauvegarde
            @ Utiliser les registres r0 à r7 sauvegardés ci-dessus

            @  à compléter : corps à traduire

            ldmfd   sp!,{r0-r7}              @ restauration
            mov      pc,lr

format:     .asciz      "minimum (e=%d, f=%d) = %d\n"
```

# 2 Boucle for (30mn)

La fonction somme calcule itérativement [1] $\sum_{i=0}^{N-1} 2^i$, pour N=8.

```
#define N 8

uint32_t somme ()
{
  register unsigned int sigma;  // utiliser le registre r0
  register unsigned int i;      // utiliser le registre r1
  register unsigned int ajout;  // utiliser le registre r2

  ajout=1;
  sigma=0;
  for(i=0;i<N;i++) {
    sigma = sigma + ajout;
    ajout = ajout * 2;
  }
  return sigma;
}
```

Compléter le code ARM ci-dessous : **traduire** le corps de somme. Donner 2 versions différentes : test de la condition de boucle après, puis avant le corps de la boucle.

1. On pourrait calculer directement $2^N - 1$ : return (1 « N) -1 ;

```
            .text
            N=8
somme:      stmfd    sp!,{r1-r2}   @ sauvegarde registres

            ... à compléter ...    @ corps de somme à écrire

suite_for:  ldm    sp!,{r1-r2}     @ restauration registres
            mov    pc,lr           @ return sigma (déjà dans r0)
```

**Quelle** serait la traduction en une seule instruction ARM de l'affectation suivante : ajout = ajout *8 ?

# 3 Arithmétique en base 2 (15mn)

On considère une machine fictive travaillant sur 5 bits.

**Compléter et interpréter** l'addition binaire suivante :

1. Compléter les lignes des retenues et du résultat apparent
2. Indiquer quels bits sont les indicateurs C et N
3. Que vaut l'indicateur V (expliquer brèvement comment vous déterminez sa valeur) ?
4. A partir des indicateurs, déduire si l'opération est correcte en supposant les entiers de type
   — naturel
   — relatif
5. Donner les valeurs décimales des opérandes et du résutat pour les deux types d'entier

| | | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 1 | 0 | 1 | 1 | a |
| + | 0 | 1 | 0 | 1 | 0 | b |
| | | | | | | retenues |
| | | | | | | résultat apparent |