

M2P CCI : examen Langage Machine, Décembre 2019

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

Avant de répondre aux questions, vous devez impérativement lire la convention d'appel des fonctions et les contraintes de stockage à respecter.

Table des matières

1	Convention d'appel et contraintes de stockage (sans question, 5mn)	1
2	Présentation (sans question) du programme de gestion de tableaux (5mn)	2
3	Déclaration des variables (15mn)	2
4	Traiter_element : indiqage de tableau, pointeur de pointeur (40mn)	2
5	Parcours : appel de fonction, pointeurs (30mn)	2
6	Questions diverses et base 2 (25mn)	2
6.1	Gestion de variables locales	2
6.2	Base 2	3
7	Annexe : procédure parcours et variables globales	3
8	Annexe : procédure traiter_element	4

1 Convention d'appel et contraintes de stockage (sans question, 5mn)

La convention d'appel applicable à toutes les fonctions à traduire est inspirée de celle de gcc :

- Les quatre premiers paramètres explicites sont stockés dans les registres r0 à r3.
- Les paramètres suivants sont empilés (paramètre de gauche en sommet de pile)
- Le paramètre implicite adresse de retour dans l'appelante est passé dans le registre lr (r14).
- Pour les fonctions, le résultat est retourné à la place du premier argument dans le registre r0.
- L'exécution de la routine appelée préserve les contenus des registres : au retour de la fonction, tous les registres¹ autres que le compteur ordinal pc (et, dans le cas d'une fonction, r0 qui contiendra le résultat) ont un contenu identique à celui d'avant l'appel.

Variables ou paramètres seront stockés en mémoire excepté si :

- l'attribut **register** est présent dans leur déclaration ou
- la convention d'appel stipule que le paramètre est passé dans un registre.

Chaque accès à une variable en mémoire devrait générer une lecture ou une écriture en mémoire. La lecture peut être omise (de préférence avec un commentaire approprié) si un registre contient une copie à jour (datant de l'affectation la plus récente) du contenu de la variable, mais pas l'écriture en mémoire, à réaliser pour chaque affectation.

1. fp/r11, ip/r12 et sp/r13 inclus

La mémoire pour les informations locales à la procédure sera allouée dynamiquement dans la pile : vous ne pouvez pas utiliser le schéma d'allocation statique dans la section bss présenté dans le chapitre "procédures simples, sans récursion".

Vous pouvez gérer la pile au choix avec sp seul ou avec le couple (fp,sp)

2 Présentation (sans question) du programme de gestion de tableaux (5mn)

Le programme a traduire

- recopie un tableau dans un autre
- détermine au passage la valeur du plus grand élément du tableau
- détermine au passage l'adresse du petit élément du tableau

La recopie d'une valeur, avec mise à jour de maximum et pointeur de minimum est réalisée par la procédure traiter_element (fichier traiter_element.c en annexe 8).

Le fichier parcours.c (annexe 7) contient les déclarations des variables globales et tableaux et la routine de parcours des tableaux.

Parcours doit être traduite sans connaître le code de traiter_element et traiter_element pourrait être appelée (dans un autre programme) par une ou plusieurs fonctions autres que parcours. La seule source d'information exploitable sur la fonction appelée ou appelante est la convention d'appel rappelée en début de sujet.

3 Déclaration des variables (15mn)

Traduire les déclarations des variables globales (maximum, ptrmin, orig et dest) de parcours.c.

4 Traiter_element : indiqage de tableau, pointeur de pointeur (40mn)

Traduire en langage d'assemblage ARM la fonction traiter_element.

5 Parcours : appel de fonction, pointeurs (30mn)

Traduire en langage d'assemblage ARM la fonction parcours.

6 Questions diverses et base 2 (25mn)

6.1 Gestion de variables locales

Expliquer la différence entre l'allocation de mémoire aux variables locales des fonctions est plus simple que l'allocation de mémoire dans le cas général (par exemple pour des éléments de liste chaînée) et pourquoi la première est plus simple à gérer.

Dans quelle section de la mémoire sera alloué le doublet retourné par la fonction creer_doublet ci-dessous ?

Expliquer pourquoi retourner à l'appelante l'adresse d'une variable locale constitue une erreur grave de programmation.

```
// détail des champs de la structure omis
typedef struct _doublet { ... } doublet_t ;

doublet_t *creer_doublet () {
    struct doublet nouveau;
    return &nouveau;
}
```

6.2 Base 2

Expliquer brièvement ce que calcule la fonction mystère ci-dessous et comment.

```
\\" fichier mystere.h
int mystere(int x);

\\fichier mystere.s
.global mystere          @ exporter mystere vers les autres fichiers

mystere:                 @ sauvegarde de r5 et r6 dans la pile omise
    movS  r5,r0,LSR #31   @ décalage logique à droite (recopie de 0)
    mvnne r0,r0           @ move not si equal
    add  r0,r0,r5
    mov pc,lr             @ restauration de r5 et r6 omise
```

7 Annexe : procédure parcours et variables globales

```
#include "parcours.h"
#include "triter.h"

int16_t maximum;
int16_t *ptrmini;

// Dans parcours.h : #define TAILLE_DEST 20 et #define TAILLE_ORIG 10

int16_t orig[TAILLE_ORIG] = {3,-4,0,7,-20,10,12,-25,8,2};
int16_t dest [TAILLE_DEST];

void parcours () {
    register int32_t i;                  // utiliser r6
    register int16_t *ptr;               // utiliser r7

    // triter element orig[0] : maximum = dest[0] = orig[0], ptrmini = &(dest[0])
    ptr = orig;
```

```

dest[0] = *ptr;
maximum = *ptr;
ptrmini = dest;

i = 1;
ptr = ptr+1;
// Boucle de parcours
while (i < TAILLE_ORIG) {
    traiter_element(&maximum, &ptrmini, dest, i, *ptr);
    i=i+1;
    ptr=ptr+1;
}
}

```

```

.text
@ r6 : i
@ r7 : ptr
@ r8,r9,r10 : tmp1, tmp2, tmp3

```

parcours: @ à compléter

8 Annexe : procédure traiter_element

```

#include "traiter.h"

void traiter_element (int16_t *max, int16_t **ptradrmin, int16_t *tableau,
                     int32_t indice, int16_t valeur) {
    register int16_t v; // utiliser r9
    v = valeur;
    if (v > *max) {
        *max = v;
    }
    tableau [indice] = v;
    if (v < *ptradrmin) {
        *ptradrmin = &(tableau[indice]);
    }
}

@ Convention d'appel :
@     r0 : max   r1 : ptradrmin   r2 : tableau   r3 : indice
@     sommet de pile : valeur
@ Locaux :
@     v : r9
@     temporaires : r6, r7, r8

traiter_element: @ à compléter

```