

Epreuve de Systèmes d'Exploitation (production de code)  
10 septembre 2003 - durée 1h30  
Tous documents autorisés

**Partie 1 (bloc d'exécution)**

On donne la fonction suivante:

```
void f1(int *s1, char *s2, int n) {
    int i;
    for (i=0 ; i<n ; i++) s1[i] = i;
    for (i=0 ; i<n ; i++) s2[i] = i;
}
```

- Donner un exemple de bloc d'exécution qu'on pourrait obtenir lors d'un appel à "f1" (faire un schéma).
- Indiquer comment on peut accéder à la valeur de "i"
- Indiquer comment on peut accéder à n
- Indiquer comment on peut accéder à s1[i]
- Indiquer comment on peut accéder à s2[i]
- Traduire en langage d'assemblage la fonction "f1" ainsi qu'un appel de votre choix à la fonction "f1"

**Partie 2 (codage)**

On donne la séquence suivante:

```
data
x:      dc.l      4      ; donnée 1
y:      dc.l      5      ; donnée 2
        dc.l      6      ; donnée 3
z:      dc.l      7      ; donnée 4
code
        add.l     d3,y    ; instruction 1
        add.l     d3,z    ; instruction 2
        add.w     d3,x    ; instruction 3
        add.w     4(a3),d4 ; instruction 4
fin:    rts          ; instruction 5
```

- Indiquer si dans la zone code, les références à "x", "y", "z" peuvent devenir des adresses absolues, des adresses relatives à PC, des adresses relatives à A5.
- En supposant qu'à l'exécution les données sont chargées juste après le code et qu'on dispose de l'adresse de la première instruction dans A5, donner le code binaire produit ainsi que le contenu des tables et tableaux intermédiaires qui sont nécessaires à la production du code.

**Epreuve de Systèmes d'Exploitation (production de code)**  
**27 janvier 2003 - durée 1h30 - Tous documents autorisés**

**Partie 1 (bloc d'exécution)**

Traduire en langage d'assemblage les fonctions données ci-dessous en prévoyant la création d'un bloc d'exécution et son utilisation pour gérer toutes les variables et tous les paramètres.

```
FONCT1(char *s1, char *s2, int n) {
    int i;
    for (i=0 ; i<n ; i++) s1[i] = s2[i];
}

FONCT2(char *s) {
    char tmp[10];
    FONCT1(tmp,s,10);
}
```

**Partie 2 (codage)**

On donne la séquence suivante:

```
data
x:      dc.w      4          ; donnée 1
y:      dc.w      5          ; donnée 2
        dc.w      6          ; donnée 3
z:      dc.l      7          ; donnée 4
code
        add.l     y,d3       ; instruction 1
        add.l     a2,a3      ; instruction 2
        add.l     z,d2       ; instruction 3
        add.w     d2,4(a3)   ; instruction 4
        beq      fin        ; instruction 5
        add.w     x,d3       ; instruction 6
fin:    rts                ; instruction 7
```

- c) Trouver le codage binaire de l'instructions 4 (expliquer brièvement la solution)
- d) Sachant que les références présentes dans les instructions "ADD" deviendront des adresses absolues, donner les contenus des tableaux de mots de 16 bits et des tables qui seront produits lors de la phase de production de code objet (justifier les valeurs produites).

Production du code des instructions: pour chaque mot produit, indiquer sous forme symbolique ou décimale ce qui a pu être codé dans le mot.

Exemple pour le début de l'instruction 4: add.w d2,d(a3)

- e) Indiquer comment sera calculé le codage de l'instruction 3 lors de la phase d'édition des liens (détailler le calcul)

Indication: à l'exécution les données sont chargées à l'adresse \$1000

- f) Reprendre les questions b) et c) en supposant que les références deviennent des adresses relatives à PC et qu'à l'exécution les données sont chargées avant les instructions.



**Epreuve de Systèmes d'Exploitation (production de code)**  
**13 septembre 2002 - durée 1h30**  
**Tous documents autorisés**

**Exercice 1 (programmation et pile)**

On va travailler sur des listes. Un élément de liste contient deux entiers 16 bits (champs "I1" et "I2") suivis de l'adresse de l'élément suivant (champ "ADRS").

- Donner un exemple de liste circulaire de 3 éléments (faire un schéma indiquant les valeurs choisies pour les champs "I1", "I2" et "ADRS" des éléments)
- Ecrire une subroutine qui fait la somme des champs "I1" et "I2" des éléments d'une liste circulaire. Supposer que l'adresse de la liste est disponible dans le registre A0.
- Même question qu'en b) mais en utilisant la pile. Supposer que l'adresse de la pile a été empilée et utiliser une variable locale définie dans la pile pour effectuer la somme.

**Exercice 2 (codage et édition des liens)**

On donne la séquence suivante:

```
data
x1:      dc.l  1
x2:      dc.w  2
x3:      dc.w  3
code
        move.l  x1,8(a2)
        move.w  x2,d2
        move.w  x3,d3
rts
```

On suppose qu'à l'exécution le registre A5 contiendra l'adresse de la zone de code et que les données seront placées immédiatement après le code.

- Sachant que les références présentes dans les instructions deviendront des adresses relatives à A5, donner les contenus des tableaux et des tables qui seront produits lors de la phase de production de code objet (justifier les valeurs produites).
- Donner le code exécutable (justifier les valeurs produites).



**Epreuve de Systèmes d'Exploitation (production de code)**  
**23 janvier 2002 - durée 2h**  
**Tous documents autorisés**

**Exercice 1 (programmation)**

On va travailler sur des listes. Un élément de liste contient deux entiers 16 bits suivis de l'adresse de l'élément suivant. Ecrire une subroutine qui indique si une telle liste est circulaire ou non. La subroutine dispose de l'adresse de la liste dans le registre A0. On précise qu'une liste est circulaire si le champ "élément suivant" d'un des éléments contient l'adresse de la liste.

**Exercice 2 (blocs d'exécution)**

On va travailler sur une subroutine qui dispose des variables locales et des paramètres suivants:

- paramètre 1: "P1" de type entier 16 bits
- paramètre 2: "P2" de type entier 32 bits
- paramètre 3: "P3" de type entier 16 bits
- variable locale 1: "L1" de type entier 16 bits
- variable locale 2: "L2" de type tableau de deux entiers 32 bits
- variable locale 3: "L3" de type entier 16 bits

Ecrire une telle subroutine sachant qu'à l'exécution elle devra disposer d'un bloc d'exécution pointé par le registre A6 et ne fera qu'initialiser ses paramètres et variables locales à 1.

Ecrire un exemple d'appel de cette subroutine

### Exercice 3 (codage)

On donne la séquence suivante:

```

                                add.b      #4,d3      ; instruction 1
                                add.l      2(a6),a2    ; instruction 2
somme1:  move.w      6(a3),8(a2)  ; instruction 3
                                move.w      12,d2     ; instruction 4
                                bra         somme1      ; instruction 5
                                rts                               ; instruction 6
```

- Trouver le codage binaire de l'instruction 2 (expliquer les valeurs des champs).
- Trouver la liste des mots de 16 bits nécessaires au codage de cette séquence d'instructions en indiquant ce que contient chaque mot (ne pas donner la valeur précise du premier mot résultant du codage des instructions).

### Exercice 4 (édition des liens)

On donne la séquence suivante:

```

                                data
x1:      dc.l      1
x2:      dc.w      2
x3:      dc.l      3
                                code
                                move.l      x1,d1
                                move.w      x2,d2
                                move.l      x3,d3
                                move.l      x3,d4
                                rts
```

- Sachant que les références présentes dans les instructions deviendront des adresses absolues, donner les contenus des tableaux et des tables qui seront produits lors de la phase de production de code objet (justifier les valeurs produites).
- Sachant qu'à l'exécution les codes des instructions seront chargés à l'adresse 1000 et que les codes des données seront chargés juste après les codes des instructions, donner le code exécutable (justifier les valeurs produites).

