INF2171
Organisation des ordinateurs et assembleur
Examen Final Hiver 2020
Remettre 26 avril 2020 23:55 au plus tard

Directives

- **Cet examen est personnel et individuel. Toute aide est interdite y compris les forums d'usagers.**
- Toutes vos réponses devront contenir les résultats intermédiaires nécessaires permettant de montrer sans ambiguïté que vous êtes arrivés au résultat demandé.
- À partir de ce document il faut créer un fichier réponse en format pdf contenant toutes les démarches effectuées ainsi que les énoncés (sauf pour la question 3, le programme initial n'est pas nécessaire).
- La question 3 doit contenir votre programme avec les modifications + les dessins dans les endroits appropriés. Le code initial n'est pas nécessaire. Le code modifié doit être remis comme un programme q3.pep sur Moodle et doit être fonctionnel pour avoir la totalité des points pour le critère fonctionnement correct.
- **Remise électronique: q3.pep + final.pdf (Moodle)**
- Afin de ne pas rater la remise, il vaut mieux remettre une copie temporaire le samedi précédant la date limite.
- La présentation est importante et vaut 10 pts.

Code permanent/Matricule :_____**(hiver 2020)**

Nom : _____

Prénom : _____

1. Représentation interne de l'information

Question 1. (5*4 = 20 pts) Soit trois octets de mémoire dont le contenu binaire exprimé en hexadécimal est le suivant :

0x39002C

Décodez ce contenu de mémoire en supposant que cette information représente

a) Un nombre entier non signé
b) Un nombre entier signé
c) Un ensemble de caractères
d) Une instruction

Les octets sont numérotés en utilisant la convention « big end »

Question 2. (5*4 = 20 pts)

a) Représenter le nombre -34.625_{10} dans le format IEEE-754 précision simple.
b) Quels sont les 4 chiffres après la virgule dans la représentation de $\frac{3}{5}$ en base 16.

c) Convertir le nombre $5_{10}*2^{-129}$ selon la norme IEEE-754, simple précision
d) Soit $FF800000_{16}$ la représentation hexadécimal d'un nombre flottant simple précision selon la norme IEEE-754. Quel est ce nombre ?

2. Conventions des programmes appelants/appelés et interruptions

Pour les prochaines questions, considérez le programme présenté plus bas. Le code de ce programme est fourni sur Moodle.

Question 3. (40 pts; -50% si le programme ne fonctionne pas après vos modifications)

Tous les programmes doivent respectés les conventions standards.

Voici la liste des conventions pour les **programmes appelants** :

1. Si un programme appelant est une fonction, il faut réserver l'espace pour le résultat sur la pile.
2. Définir les paramètres, et placer leurs adresses ou leurs valeurs sur la pile.
3. Effectuer le branchement au sous-programme par un CALL au sous-programme
4. Récupérer le résultat de la fonction, si c'est le cas, et désallouer l'espace de la pile

Les responsabilités des **sous-programmes** :

1. Établir l'espace local et les paramètres en définissant un cadre de pile
2. Préserver les contenus des registres sur la pile
3. Exécuter le corps de sous-programme
4. Restaurer les contenus des registres
5. Nettoyer la pile (variables locales + paramètres)
6. Retourner au programme appelant en utilisant l'instruction RETn

Le programme suivant implémente un algorithme de tri d'un tableau créé dynamiquement et initialisé à partir de l'entrée standard. Le programme présenté ne respecte pas les conventions. Votre tâche consiste à réécrire le programme en effectuant les modifications nécessaires. Tout le traitement doit être effectué par les fonctions. Il y a 3 « fonctions » auxiliaires dans le code : `saisir`, `afficher` et `trier` et la fonction principale `main`. Dans cette question, après avoir introduit les modifications nécessaires, vous devez dessinez l'état de la pile :

1. au début de l'exécution de sous-programme `saisir` après l'allocation de l'espace dans la pile pour les variables locales;
2. au début de la fonction `afficher`, après l'allocation des variables locales (premier appel)
3. au début de la fonction `trier`, après l'allocation des variables locales

Pour les dessins, inspirez-vous des dessins dans les notes de cours « Passage des paramètres par la pile ».

Utilisez le simulateur PEP8 pour s'assurer que le programme fonctionne correctement et n'oubliez pas que votre programme doit respecter **toutes les conventions**!

Question 4. Interruptions (chapitre 12 du manuel) (10 pts)

Dans le code du programme fourni, cherchez l'instruction `STRO msg1,d` et expliquez la procédure de l'exécution de cette interruption étape par étape.

Programme initial, question **3**

```
main: STRO  msg,d
      BR    saisir
ret:  CALL  afficher
      STRO  msg1,d
      BR    trier
aff:  CALL  afficher
      STOP
; saisir: alloue dynamiquement un tableau et l'initialise
; IN : void
; OUT : valeurs de retour : une adresse de tableau et la taille
taille:     .WORD 0     ; taille de tableau
adresse:    .WORD 0     ; adresse du premier élément = adresse du tableau
cmp:        .BLOCK 2    ; variable locale, compteur
saisir:     DECI  taille,d ; taille de tableau
            LDA   taille,d
            ASLA
            STA   taille,d
            LDA   0,i
            STA   cmp,d ; cmp = 0
; premier élément = adresse de la table sur HEAP
            LDA   2,i   ; 2 octets pour un entier
            CALL  new
            STX   adresse,d
            DECI  0,x ;
            ADDX  2,i
            LDA   cmp,d
            ADDA  2,i
            STA   cmp,d
sloop:      CPA   taille,d
            BRGE  ret
            LDA   2,i   ; 2 octets pour un entier
            CALL  new
            LDX   cmp,d
            ADDX  adresse,d
            DECI  0,x
            LDA   cmp,d ;
            ADDA  2,i
            STA   cmp,d
            BR    sloop ; } // fin for
; afficher: affiche un tableau
; IN: adresse = adresse de tableau; taille = taille de tableau
; OUT: void
afficher:   LDX   adresse,d
            LDA   0,i
aloop:      CPA   taille,d
            BRGE  afin
            DECO  0,x
            CHARO ' ',i
            ADDX  2,i
            ADDA  2,i
            BR    aloop
afin:       CHARO '\n',i
            RET0
;var trier = function (t) { // tri bulle
; var echange;
; do {
; echange = false;
; for (var i=0; i<t.length-1; i++) {
; if (t[i+1] < t[i]) {
; var temp = t[i];
; t[i] = t[i+1];
; t[i+1] = temp;
```

```
; echange = true;
; }
; }
; } while (echange);
;}
; trier: trie un tableau in situ
; IN: adresse = adresse de tableau; taille = taille de tableau
; OUT: void
; Variables locales
lng1: .WORD 0     ; valeur de t.length-1
tI:   .WORD 0     ; t[i]
i:    .WORD 0     ; i
ech:  .WORD 0     ; echange, 0 - FAUX, 1 - VRAI
;***********************************************
trier:      LDA   taille,d
            ADDA  adresse,d
            SUBA  2,i
            STA   lng1,d
tWhile:     LDX   adresse,d
            LDA   0,i
            STA   ech,d ; ech = false
cFor:       CPX   lng1,d
            BRGE  finF  ; for (var i=0; i<t.length-1; i++) {
            LDA   0,x   ; A = t[i]
            STA   tI,d
            ADDX  2,i
            CPA   0,x   ; t[i] > t[i+1] ? echanger : continue for
            BRLE  cFor
            LDA   0,x   ; A = t[i+1];
            SUBX  2,i
            STA   0,x   ; t[i] = t[i+1];
            ADDX  2,i
            LDA   tI,d  ; t[i+1] = tmp;
            STA   0,x
            LDA   1,i
            STA   ech,d ; ech = true
            BR    cFor
finF:       LDA   ech,d
            CPA   1,i   ; ech == true ?
            BREQ  tWhile
            BR    aff
msg:        .ASCII "Tableau non trié\n\x00"
msg1:       .ASCII "Tableau trié\n\x00"
;******* operator new
; Precondition: A contains number of bytes
; Postcondition: X contains pointer to bytes
new:        LDX   hpPtr,d     ;returned pointer
            ADDA  hpPtr,d     ;allocate from heap
            STA   hpPtr,d     ;update hpPtr
            RET0
hpPtr:      .ADDRSS     heap  ;address of next free byte
heap:       .BLOCK      1     ;first byte in the heap
            .END
```