DST UE21 - M2101
Architecture et programmation des mécanismes de base d'un système informatique

IUT Paris Descartes, Département Informatique
Ph. Darche

Mars 2014
Durée : 1h30

Aucun document autorisé. Calculatrice non autorisée. Sujet de deux pages et trois annexes. Les parties sont indépendantes. Répondez dans l'ordre de numérotation des problèmes. Barème indicatif. **Justifiez clairement vos réponses**.

Partie I : Questions de cours sur le fonctionnement d'un microprocesseur et le langage d'assemblage (7,5 points)

Q1. Dessinez le cycle d'exécution de base d'une instruction d'un microprocesseur (trois étapes) et expliquez-le.

Q2. Quelles sont les fonctions du registre d'état ? A quel moment ce dernier peut-il être mis à jour ? Est-ce que l'état de ses bits est temporaire ou permanent ? Donnez un exemple.

Q3. Donnez la hiérarchie des langages informatiques vue en cours et en TP (trois niveaux).

Q4.a. Quelles sont les deux familles d'instructions de saut ? Pour chacune d'elles, donnez un exemple d'instruction.

Q4.b. Quel est le type d'adressage qui est utilisé par ces instructions de saut de type « `je` » ?

Q4.c. Détaillez le fonctionnement de l'instruction `jc` sous la forme d'un pseudo-code.

Q5. Pour une fonction, quels sont les deux types de passage de paramètres ? Par quelle(s) voie(s) est-il possible de passer des paramètres ? Quelle est la voie la plus rapide ? Quelle est sa limitation ?

Partie II : Ecriture d'un programme en langage d'assemblage (12,5 points)

On décide d'écrire un programme en langage d'assemblage du microprocesseur Intel 8086 qui calcule la somme des n premiers entiers naturels avec $n \in [1, 22]$. Le squelette d'un programme en langage d'assemblage est donné en annexe 1. L'annexe 3 donne les transparents de cours concernant les sauts conditionnels.

Q1. Remplissez le tableau QII-1.

Nombre	Somme
1	
2	
3	
4	
5	
6	

7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	253

Tableau QII-1 : Somme des n premiers entiers naturels (n ≠ 0)

Q2. Deux variables, `x` et `somme`, seront définies. Quelles sont les valeurs minimale et maximale de ces variables ? Quel format allez-vous choisir dans le programme ? Pour répondre à ces questions, remplissez le tableau QII-2. **Le format minimum est le nombre de bits pour coder l'étendue des valeurs des variables.**

Variables	Valeur min.	Valeur max.	Format min. (bit)	Format retenu pour le problème (bit)	Initialisation
`x`					
`somme`					

Tableau QII-2 : Etude du format des variables

Le format de la variable `somme` pour la suite du DST est fixé à 8 bits.

Q3. Donnez le pseudo-code du programme **avec interdiction d'utiliser un sous-programme**. Ecrivez ensuite le programme correspondant. **Il est inutile de recopier les trois premières et les trois dernières lignes de code du modèle de l'annexe 1.**

Q4. Donnez les étapes de développement logiciel sous la forme d'un graphique avec les commandes de la ligne de commande DOS (*i.e. shell*) correspondantes pour que le programme puisse être exécuté sous le contrôle du débogueur `td.exe`. Ne pas oublier d'indiquer les extensions des fichiers générés à chaque étape. Le programme s'appellera « **sum_int.asm** ».

Q5. La valeur initiale de x pourra maintenant atteindre 23. Sans modifier le programme précédent, en particulier le format des variables, complétez-le pour qu'il affiche à l'écran un message (*cf.* annexe 2) indiquant si le résultat est juste ou non. Vous donnerez la valeur du résultat et expliquerez pourquoi le résultat n'est pas juste. **Attention, le test de validité ne doit pas porter sur la valeur de la variable x.**

Q6. Transformez le programme de la question Q3 pour que le calcul de cette somme se réalise dans un sous-programme nommé `sum_int` (*cf.* partie 2 de l'annexe 1). Le passage des paramètres entrant et sortant se fera obligatoirement par valeur et par les registres **cl** (paramètre entrant) et **al** (paramètre sortant). Vous traiterez les cas particuliers x = 0 et x > 22 par un message d'alerte à l'écran (pas de calcul pour ces cas). Ne mettez sur votre copie que la partie code (*i.e.* pas de partie déclaration de données qui est identique à celle de la question Q3).

Annexe 1 : squelette de programme en langage d'assemblage 8086

```
; Nom du programme : sum_int.asm
; Fonction : Somme des 22 premiers entiers naturels (n ≠ 0)
; Type de passage de paramètre : XX à remplir si nécessaire
; Mode de passage de paramètres : XX à remplir si nécessaire

IDEAL
DOSSEG
MODEL TINY
P8086

STACK 100h

DATASEG

    Partie à compléter dans votre copie


CODESEG
; déclaration des sous-programmes

PROC sum_int NEAR

            Partie 2

fin_sp:             ret
ENDP sum_int


; programme principal

debut:
    mov ax,@data
    mov ds,ax
    mov es,ax

; corps du programme

    Partie à compléter dans votre copie


    mov ah,4ch
    int 21h

END debut
```

Annexe 2 : l’interruption n° 33

Il existe une interruption réservée aux appels système qui porte le numéro 33 (ou 21h). Elle permet d’appeler le système d’exploitation DOS qui est alors vu comme un ensemble de fonctions. La fonction N° 9 de cette interruption permet d’afficher une chaîne de caractères à l’écran. La syntaxe est la suivante :

- `int 33` ou `int 21h`

Le passage des paramètres doit suivre la convention suivante :

- le registre de données ah (groupe registres généraux) doit contenir le numéro de la fonction,
- le registre de données dx contiendra l’adresse logique du début de la chaîne à afficher,
- la chaîne à afficher doit se terminer par le caractère “$”.

Annexe 3 : les instructions de saut conditionnel (transparents du cours)