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

IUT de Paris Descartes, Département Informatique
Ph. Darche

Mars 2017
Durée : 1h30

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

Partie I : Questions de cours (7,75 points)

A. La chaîne de développement logiciel

Q1. Soit le programme source *mon_prog.c* écrit en langage C. Donnez les quatre étapes qui permettent d'obtenir un exécutable. Précisez pour chacune d'entre elles l'outil logiciel (*i.e.* le programme) utilisé et le type de fichier généré. Vous pouvez reproduire le dessin du cours.

B. Appel d'une fonction

Q1. Quelles sont les deux formes d'appel d'une fonction en langage d'assemblage ?

Q2. Les deux lignes grisées et encadrées de l'annexe 2 sont un exemple d'une des formes d'appel.

Q2.1. Expliquez-les.

Q2.2. Quel est le mode et le type de passage du paramètre dans cet exemple ?

Q3. Dans la seconde forme d'appel de fonction, comment le processeur revient à l'appelant ?

Q4. Pourquoi l'instruction *jmp* n'est-elle pas adaptée pour un appel de fonction ?

Q5. Dans le cadre d'un appel de fonction, à quoi peut servir la pile lorsqu'un sous-programme est appelé ?

C. La segmentation

Q1. Donnez la définition d'un segment.

Q2. Comment le CPU repère-t-il les segments en mémoire principale ?

Q3. Qu'indiquent les directives STACK 100h, DATASEG et CODESEG de l'annexe 1 ?

Q4. Quel est l'intérêt de la segmentation ?

Partie II : Ecriture d'une application en langage d'assemblage (12,5 points)

On décide d'écrire un programme en langage d'assemblage 8086 qui calcule une division euclidienne **par soustractions successives**. L'équation de base de la division euclidienne avec le dividende N, le diviseur D, le quotient Q et le reste R, est :

$$N = D \times Q + R$$

avec N, Q, R $\in \mathbb{N}$ et D $\in \mathbb{N}^*$.
La démarche de développement sera d'écrire une première version (version 1) sans se préoccuper des deux cas particuliers D = 0 et N < D. La version 2 prendra en compte ces deux cas particuliers. La version 3 consistera à afficher un message d'erreur. La version 4 transforme le programme de la version 2 en sous-programme.

Vous éviterez au maximum d'utiliser dans le code les variables mémoires déclarées pour une question de rapidité d'exécution en privilégiant les registres du microprocesseur. Par contre, en fin de chaque programme, vous n'oublierez pas de ranger le résultat du calcul dans la variable concernée. La dernière annexe donne les instructions de sauts conditionnels qui peuvent vous être utile.

Version 1. Ecriture d'un programme principal sans traitement des cas particuliers

Ecriture d'un programme en langage d'assemblage 8086 qui calcule une division euclidienne **par soustractions successives** et **de manière itérative** (*i.e.* pas de récursivité). **Un squelette de programme est fourni en annexe 1. Les parties 1 et 2 seront écrites sur votre copie.**

Q1. Quatre variables nommées `N, D, Q et R` seront définies. La valeur maximale pour `N` et `Q` sont 65535 (rappel : $2^{16} = 65536$). Pour vous aider à définir leur format et comme vu en TP, veuillez remplir le tableau QII-2.

Variables	Valeur minimale	Valeur maximale	Format minimum pour l'intervalle de validité (bits)	Format retenu pour le programme (bits)
N				
D				
Q				
R				

Tableau QII-2 : Etude du format des variables pour la division euclidienne

Q2. Quel sera le code numérique qui sera utilisé **en machine** pour les valeurs des variables ?

Q3. Donnez la définition des variables correspondantes en langage d'assemblage avec une initialisation. Précisez le numéro de la partie concernée dans le programme squelette. **Pour la suite de cette partie 1, aucune autre variable ne devra être déclarée.** Les variables Q et R seront impérativement initialisées à zéro.

Q4. Quelle est la condition d'arrêt de la boucle qui réalise la soustraction ?

Q5. Ecrivez le programme correspondant appelé *div.asm* **avec interdiction d'utiliser un sous-programme**. Vous ne traiterez pas les deux cas particuliers D = 0 et N < D. La structure de contrôle utilisée sera obligatoirement un **répéter_jusqu'à <condition>**.

Version 2. Ecriture d'un programme principal avec traitement des cas particuliers

Q1. Modifiez votre programme pour qu'il traite les cas particuliers D = 0 et N < D (nouvelle partie 2).

Version 3. Affichage d'un texte

Q2. Modifiez le programme précédent (déclaration des variables et code) pour qu'il affiche le message « division par zéro impossible » si le diviseur D est nul. Le nom du programme sera div_aff.asm. Le calcul ne s'effectuera donc pas. Vous vous inspirerez du programme *affiche.asm* de l'annexe 2.

Version 4. Transformation du programme de la version 2 en sous-programme

Q5. Transformez votre programme de la version 2 (*i.e.* division euclidienne avec traitement des deux cas particuliers mais pas d'affichage du message d'erreur) en sous-programme que vous appellerez *div_sp*. Ecrivez aussi le programme l'appelant (nouvelle partie 2 à écrire). **Le passage des paramètres se fera obligatoirement par valeur et par registre. Un squelette de sous-programme est fourni en annexe 3 (partie 3 à écrire).**

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

```
; nom du programme : div.asm
; fonction : division par soustractions successives

            IDEAL
            DOSSEG
            MODEL TINY
            P8086

            STACK 100h

            DATASEG

            Partie 1

            CODESEG

; programme principal

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

; corps du programme

            Partie 2

; fin « propre » du programme
            mov ah,4ch
            int 21h
END debut
```

Annexe 2 : Squelette du programme affiche.asm en langage d'assemblage 8086

```
; nom du programme : affiche.asm
; fonction : affichage d'une chaîne de caractères

            IDEAL
            DOSSEG
            MODEL TINY
            P8086

            STACK 100h

            DATASEG

; déclaration de la chaîne à afficher
msg1        DB "Ceci est un message !",10,13,"$"

            CODESEG

; programme principal

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

; corps du programme

            mov ah,9
            mov dx,OFFSET msg1
            int 21h

; fin « propre » du programme
            mov ah,4ch
            int 21h
END debut
```

Annexe 3 : Squelette de sous-programme

```
; Déclaration d'un sous-programme

PROC div_sp NEAR

            Partie 3

fin_sp:

ENDP div_sp
```

Saut sur état d'indicateur

- Représentation entière non signée ($\mathbb{N}$)
  - supérieur ($>$)
    - JA (*Jump on Above*)
    - JNBE (*Jump on Not Below or Equal*)
      - (CF and ZF) = 0
  - supérieur ou égal ($\geq$)
    - JAE (*Jump on Above or Equal*)
    - JNB (*Jump on Not Below*)
      - CF = 0
  - inférieur ($<$)
    - JB (*Jump on Below*)
    - JNAE (*Jump on Not Above or Equal*)
      - CF = 1

Saut sur état d'indicateur

- Représentation entière non signée (suite)
  - inférieur ou égal (≤)
    - JBE (*Jump on Below or Equal*)
    - JNA (*Jump on Not Above*)
      - (CF or ZF) = 1
  - non égal (≠)
    - JNE (*Jump on Not Equal*)
    - JNZ (*Jump on Not Zero*)
      - ZF = 0
  - égal (=)
    - JE (*Jump on Equal*)
    - JZ (*Jump on Zero*)
      - ZF = 1