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

IUT de Paris Descartes, Département Informatique
Ph. Darche

Mars 2016
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 (5,5 points)

**Q1. Les registres.**

**Q1.1.** Donnez la définition d'un registre.

**Q1.2.** A quoi sert le compteur ordinal ? Comment s'appelle-t-il dans le microprocesseur 8086 (acronyme et nom complet anglais) ?

**Q1.3.** Qu'est-ce qu'un registre implicite ? Citez deux instructions qui en utilisent et le nom du ou des registres correspondants.

**Q2.** Quelle est la fonction d'un compilateur ? Quelle est la fonction d'un assembleur ? Quelle est la fonction d'un éditeur de liens ?

**Q3.** Soient les lignes **indépendantes** d'instruction du langage d'assemblage du 8086 suivantes :

```
ligne_1:  mov ax,3
ligne_2:  mov 3,ax
ligne_3:  add [n],[m] ; avec n et m au format 16 bits
ligne_4:  add ax,[n] ; avec la déclaration n DB 16
ligne_5:  mul [n]
ligne_6:  mul [n],al ; avec la déclaration n DB 0
ligne_7:  mul ax,bx
ligne_8:  mov [n],[m] ; avec n et m au même format
```

Pour chacune d'elles, dites ce qu'elle fait et si elle est valide ou non. Justifiez votre réponse. L'absence de déclaration d'une variable n'est pas considérée comme une erreur.

## Partie II : Ecriture d'une application en langage d'assemblage (4,5 + 2,5 + 7,5 points)

On décide d'écrire un programme en langage d'assemblage 8086 qui calcule la factorielle d'une variable **n** et qui range le résultat dans une variable **résultat**.

fac(n) = n! = n × (n - 1) × (n - 2) × ..... × 1 avec le cas particulier 0! = 1, n ∈ ℕ.

Le tableau QII-1 rappelle quelques valeurs de factorielle.

| n | n! |
|---|---|
| 0 | 1 |
| 1 | 1 |
| 2 | 2 |
| 3 | 6 |
| 4 | 24 |
| 5 | 120 |
| 6 | 720 |
| 7 | 5040 |
| 8 | 40320 |
| 9 | 362880 |

**Tableau QII-1 : Quelques valeurs de fac(n)**

**La démarche de développement sera d'écrire une première version sans se préoccuper des valeurs particulières n = 0 ou 1 (étape 1). Une deuxième version (question Q4 de l'étape 1) prendra en compte ces deux valeurs particulières. L'étape 2 consistera à afficher un message d'erreur. L'étape 3 transforme le programme de l'étape 1 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.

# Etape 1. Ecriture d'un programme principal

Ecriture d'un programme en langage d'assemblage 8086 qui calcule la factorielle de n **($n \in \mathbb{N}$, $n < 9$) 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.** Deux variables, `n` et `résultat`, seront définies. Pour vous aider à définir leur format, veuillez remplir le tableau QII-2. Rappel : $2^{16} = 65536$.

| Variables | Valeur minimale | Valeur maximale | Format minimum pour l'intervalle de validité (bits) | Format retenu pour le programme (bits) |
|---|---|---|---|---|
| `n` | | | | |
| `résultat` | | | | |

**Tableau QII-2 : Etude du format des variables pour le calcul de la factorielle (*i.e.* n!)**

**Q2.** 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.** La variable `résultat` sera impérativement initialisée à zéro.

**Q3.** Ecrivez le programme correspondant appelé **avec interdiction d'utiliser un sous-programme**. Vous ne traiterez pas les deux cas particuliers n = 0 et n = 1. Vous devrez par ailleurs utiliser impérativement l'instruction `mul`. La structure de contrôle sera un **répéter_jusqu'à <condition>** obligatoirement et l'instruction `loop` est interdite.

**Q4.** Modifiez votre programme pour qu'il traite les cas particuliers n = 0 et n = 1 (nouvelle partie 2).

# Etape 2. Affichage d'un texte

**Q1.** Expliquez les trois lignes qui permettent d'afficher la chaîne de caractères (voir annexe 2).

**Q2.** Modifiez le programme précédent (déclaration des variables et code) pour qu'il affiche un message si n dépasse 8 et que le calcul ne se fasse pas. Vous vous inspirerez du programme *affiche.asm* de l'annexe 2.

# Etape 3. Transformation du programme de l'étape 1 en sous-programme

**Q1.** Quelle est la fonction de l'instruction *call* ? Expliquez en détail son exécution interne.

**Q2.** Quelle est la fonction de l'instruction *ret* ? Expliquez en détail son exécution interne.

**Q3.** En programmation, à quoi peut servir la pile lorsque des sous-programmes sont utilisés ?

**Q4.** Quelles sont les deux formes de passage d'un paramètre ? Précisez les trois voies de passage possibles ?

**Q5.** Transformez votre programme de la question Q4 de l'étape 1 (*i.e.* factorielle avec traitement des deux cas particuliers mais pas de test de valeur max., ni d'affichage de message d'erreur) en sous-programme que vous appellerez *fac_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).**

**Q6.** Quels sont les avantages et les inconvénients du passage des paramètres par les registres ?

**Q7.** Combien de registres entrants et sortants avez-vous utilisés pour ce passage ? Nommez-les.

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

```
; Nom du programme : fac.asm
; Fonction : calcul d'une factorielle

            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

; Terminaison du programme par appel système
            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

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

; Terminaison du programme par appel système
            mov ah,4ch
            int 21h
END debut
```

# Annexe 3 : Squelette de sous-programme

```
; Déclaration d'un sous-programme

PROC fac_sp NEAR

            Partie 3

fin_sp:

ENDP fac_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