

Examen à livres fermés d'une durée de 2 heures et demie. Ce formulaire comprend 3 questions et pour chacune d'elle un emplacement suffisant pour la réponse.

Ecrivez votre nom sur chacune des feuilles. N'écrivez que sur le recto des feuilles.

Vous devez remettre **toutes** les feuilles de ce questionnaire.

### Question 1

Ecrire un **programme** assemblable (avec DOSSEG...) en langage d'assemblage qui :

- demande à l'utilisateur une valeur entière non signée  $\leq 65535$  (on suppose que l'utilisateur ne se trompe pas...),
- stocke cette valeur dans AX et
- calcule combien de bits sont nécessaires pour sa représentation en base binaire (max : 16 bits !).

Plusieurs méthodes sont possibles...on peut par exemple déterminer combien de divisions entières successives par 2 il faut faire à partir de la valeur de départ pour obtenir un résultat nul.

*Exemple : la valeur 12 nécessite 4 chiffres binaires ( $12_{10} = 1100_2$ )*

$$12 \text{ div} 2 = 6 ; 6 \text{ div} 2 = 3 ; 3 \text{ div} 2 = 1 ; 1 \text{ div} 2 = 0$$

Votre programme doit ensuite afficher à la ligne suivante le résultat sous la forme :

« Il faut ... symboles pour représenter la valeur ... en binaire »

*dans l'exemple ci-dessus, le programme affichera :*

*« Il faut 4 symboles pour représenter la valeur 12 en binaire »*

### IMPORTANT :

Vous disposez, pour ce programme, des procédures décrites ci-dessous. Vous les utiliserez **en respectant leur description** ! Surtout, ne les réécrivez pas...limitez-vous à placer les directives de début et de fin de chaque procédure au bon endroit dans le code de votre programme, et à faire évidemment les appels corrects à celles-ci quand c'est nécessaire.

Procédure *lectnbr* : lit un nombre non-signé au clavier et place sa valeur dans AX. Elle suppose que se trouve dans BX l'offset de l'adresse d'une zone mémoire de 8 bytes nécessaire pour la lecture.

Procédure *affnbr* : affiche la valeur décimale non signée se trouvant dans AX. Elle suppose que se trouve dans BX l'offset de l'adresse d'une zone mémoire de 6 bytes (au moins).

La convention utilisée par ces procédures pour le passage des paramètres est celle du Pascal (c'est donc la procédure qui retire les paramètres qui ont été mis sur la pile).

Réponse à la question 1 :

```
DOSSEG
.MODEL SMALL
.STACK 100h
.DATA
NBR DW ?
AFF DB 6 DUP (?)
LECT DB 8 DUP (?)
MSGUN DB 'Il faut $'
MSGDEUX DB ' symboles pour représenter la valeur $'
MSGTROIS DB ' en binaire$'
.CODE
```

```
lectnbr PROC NEAR
```

```
...
```

```
affnbr PROC NEAR
```

```
...
```

```
DEBUT:  MOV AX,@DATA
        MOV DS,AX
        MOV BX,OFFSET NOMBRE
        CALL lectnbr
        MOV NBR,AX
        MOV DX,0
        MOV CX,0
```

```
DIVISION:  MOV BX,2
           DIV BX
           INC CX
           CMP AX,0
           JNE DIVISION
```

```
AFFICHAGE: MOV AH,09h
```

```
MOV DX,OFFSET MSGUN
INT 21h
MOV AX,CX
MOV BX,OFFSET LECT
PUSH BX
CALL affnbr
MOV AH,09h
MOV DX,OFFSET MSGDEUX
INT 21h
MOV AX,NBR
MOV BX,OFFSET LECT
CALL affnbr
MOV AH,09h
MOV DX,OFFSET MSGTROIS
INT 21h
MOV AH,4Ch
INT 21h
END DEBUT
```

Question 2

2.1. Donnez **un** exemple d'instruction(s) de **type shift, rotation et/ou opérations logiques**, qui permet :

(a) si `AL` contient la valeur ( en hexadécimal) `F8`,

- d'isoler le 8 (donc d'obtenir `08`): `AND AL,00001111b`
- d'isoler le F (donc d'obtenir `0F`): `SHR AL,4`

(b) de voir si le contenu de `AX` est pair sans le modifier et de « sauter » vers une zone (label) `PAIR` ou `IMPAIR` selon le résultat (ces zones contiennent des instructions non précisées ici ! Vous mettrez 3 petits points...) :

```

TEST AX,1b
JZ PAIR
IMPAIR ...
...
JMP SUITE
PAIR ...
...
SUITE ...
...

```

(c) si `CL` contient (en binaire) `00000SSSS` (où `S` est indifféremment 1 ou 0) et si `DL` contient (en binaire) `000MMMMM` (où `M` est indifféremment 1 ou 0), de compacter le tout dans `CL` sous la forme `MMMMMSSSS` :

```

SHL DL,3
ADD CL,DL // OR CL,DL

```

2.2.

En partant de la situation suivante:

`AX = 000E`      `BX = 0000`      `CX = 0003`      `DX = 0000`

quel sera l'effet sur ces registres de l'instruction suivante:

`DIV CL`

`AX = 0204`      `BX = 0000`      `CX = 0003`      `DX = 0000`

2.3.

Expliquez ce que fait le bloc d'instructions suivant (soyez le plus complet possible !) :

CLD

REPE CMPSB

CLD met à zéro le flag de direction (DF). REPE est synonyme de REPZ, cela aura pour effet de répéter l'instruction tant que le flag zéro (ZF) est égal à 1 et que CX est différent de 0. CMPSB compare les bytes dont l'adresse est DS:SI et ES:DI, les flags sont positionnés comme lors de l'exécution de l'instruction CMP et SI et DI sont incrémentés. Cela permet par exemple de comparer deux chaînes de caractères. Lorsque deux lettres non identiques sont comparées, l'instruction de répétition s'arrête mais SI et DI seront tout de même positionnés sur le caractère suivant.

Question 3

Un palindrome est un mot ou un groupe de mots qui peut être lu de gauche à droite ou de droite à gauche en gardant le même sens. (*ex : solos, 2002, esope reste ici et se repose,...*). Vous remarquez que les « blancs » sont ignorés.

Nous allons pour simplifier nous limiter ici à des chaînes de caractères où les blancs seront traités comme les autres caractères (*ex : «2 52» n'est pas un palindrome mais «2 5 2» bien*). Il suffit donc de vérifier que le premier et le dernier caractère sont égaux, puis le deuxième et l'avant-dernier,...jusqu'à arriver au « milieu » de la chaîne...attention, pensez bien à votre condition d'arrêt (nombre de caractères dans la chaîne pair ou impair...essayez sur un exemple !)

Ecrivez une **procédure** PALINDROME qui reçoit trois paramètres via la pile:

1. la partie offset de l'adresse d'une chaîne de caractères
2. le nombre de caractères de la chaîne
3. la partie offset de l'adresse d'une zone mémoire de 1 byte (pour recevoir la réponse)

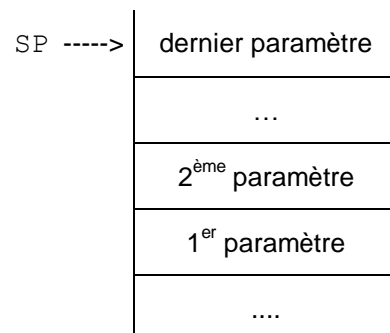
Cette procédure doit tester si la chaîne est un palindrome (selon notre définition). Elle mettra 1 dans la zone mémoire de réponse si c'est le cas, et 0 sinon.

Il vous est demandé de respecter les conventions suivantes :

1. concernant l'utilisation des registres, le programme principal sauve lui-même les registres AX, BX, CX, DX, SI et DI si nécessaire, la procédure peut donc les utiliser sans précautions.
2. les paramètres sont transmis du programme principal aux procédures via la pile en se conformant aux conventions du Pascal.

Pour rappel :

Pour se conformer aux conventions du Pascal, les paramètres sont mis sur la pile en commençant par le premier et en terminant par le dernier. Avant l'appel de la procédure, la pile a l'aspect suivant:



C'est la procédure qui retire les paramètres qui ont été mis sur la pile.

Réponse à la question 3 :

PALINDROME PROC NEAR

PUSH BP

MOV BP,SP

MOV AX,[BP+6] ; longueur

MOV BX,2

MOV DX,0

DIV BX

MOV CX,AX ; quotient AX devient le compteur dans CX

MOV SI,[BP+8] ; au début

MOV DI,[BP+8] ; adresse de la chaîne

ADD DI,[BP+6] ; à la fin

BOUCLE: MOV AL,[SI]

CMP AL,[DI]

JNE PASEGAL

INC SI ; incrémente le 'début'

DEC DI ; décrémente la 'fin'

DEC CX ; décrémente le compteur

JNZ BOUCLE

EGAL: MOV BX,[BP+4] ; l'offset d'une zone mémoire d'1 byte

MOV byte ptr [BX],1

JMP FINPROC

PASEGAL: MOV BX,[BP+4] ; l'offset d'une zone mémoire d'1 byte

MOV byte ptr [BX],0

FINPROC: POP BP

RET 6

*Haute Ecole Léonard de Vinci*

*IPL* - Baccalauréat en Informatique - 1ère année

**Examen de Langage d'Assemblage -- Juin 2008**

Nom :

Prénom :

Numéro :

---

PALINDROME ENDP