TP 4 : Programmation Matlab

Exercice 1

Expliquer ce que fait chacun des trois programmes suivants :

```
;prgm1
data segment
x dw 5h
y dw 6h
z dw 0
data ends
code segment
assume ds:data,cs:code
start:
  MOV AX,data
  MOV DS,AX
  MOV AX,x
  MOV BX,y
  ADD AX,BX
  MOV z,AX
  MOV BX, z
  hlt
code ends
end start
```

```
;prgm2
data segment
   x dw 5h
   y dw 6h
   msg  db 'Hello'
   tab1 dw 2, 4, 6, 8, 10
   tab2 dw 5 dup(0)
data ends
code segment
assume ds:data,cs:code
start:
      MOV AX,data
      MOV DS,AX
      MOV AX,0
      MOV CX,5
      MOV SI,0
lab: ADD AX,tab1[SI]
      ADD SI,2
      LOOP lab
code ends
end start
```

```
;prgm3
include 'emu8086.inc'
data segment
tab dw 0,2,4,6,8,10,12,14,16,18
data ends
code segment
assume ds:data,cs:code
start:
      MOV AX,data
      MOV DS,AX
      print 'Hello'
      printn ' World!'
      printn 'Enter a number
      between 1 and 10:'
      CALL   scan_num
      CMP CX,1
      JB lab2
      CMP CX,10
      JA lab2
lab0:MOV SI,0
lab1:MOV AX,tab[SI]
      CALL print_num
      PRINTN ''
      ADD SI,2
      LOOP lab1
      JMP lab3
lab2:print 'Error'
lab3:hlt
code ends
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_UNS
DEFINE_SCAN_NUM
end start
```

Declaring (initialized) variables in .data section) :
<label> <type> <value>
Types:

db	**b**yte
dw	**w**ord
dd	**d**ouble-word
dq	quad-word

n dup(value): duplicate value for *n* times.

- AX: the Accumulator
- BX: the Base Register
- **CX: the Count Register**
- DX: the Data Register
- SP: the Stack Pointer
- BP: the Base Pointer
- **SI: the Source Index Register**
- DI: the Destination Register
- IP: the Instruction Pointer
- **CS: the Code Segment Register**
- **DS: the Data Segment Register**
- SS: the Stack Segment Register
- ES: the Extra Segment Register

- LOOP: Decrease CX, jump to label if CX not zero.
- SCAN_NUM: procedure that gets a SIGNED number from the keyboard, and stores the result in CX register. To use it declare: DEFINE_SCAN_NUM before END directive.
- RINT_NUM:procedure that prints a signed number in AX register. To use it declare: DEFINE_PRINT_NUM and DEFINE_PRINT_NUM_UNS before END directive.
- PRINT *string*: macro with one parameter, allows to print out a string.
- PRINTN *string*: macro with 1 parameter prints out a string. It is the same as PRINT but automatically adds *carriage return* at the end of the string.

Exercice 2

1) Ecrire un programme, en langage assembleur 8086, qui permet de compter les nombres nuls dans un tableau d'octets de longueur 30 et débutant à l'adresse [00h], le résultat sera placé à l'adresse [200h].

2) Ajouter ou programme précédent les déclarations ci-dessous, exécuter le programme puis expliquer les résultats.

```
msg  db 'Hello world'
tab2 dw 5 dup(1)
```

Exercice 3

1) Ecrire un programme qui permet de trier par ordre croissant un tableau de longueur N = 100h débutant à l'adresse [200h].

Exercice 4

1) Ecrire une procédure (appelée *somme*) permettant de calculer la somme de deux nombres. Cette procédure utilise le passage de paramètres par registres: Les deux nombres sont passés par les registres AX et AX et le résultat sera placé dans AX.

2) Appeler la procédure *somme* à partir d'un autre programme (le programme appelant).

3) Redéfinir la procédure précédente en utilisant le mode de passage de paramètres par pile.

Exercice 5

Définir puis appeler chacune des procédures suivantes :

1) fact: qui calcule le factoriel n! d'un entier naturel.
2) fibo : qui calcule n termes de la suite de Fibonacci : $U_0=1$, $U_1=1$, $U_{n+1}=U_n + U_{n-1}$.
3) Pgcd : qui calcule le plus *gracd* commun diviseur de deux nombres.
4) Une procédure qui permet de déterminer le maximum dans un tableau d'entiers.