TP2 – Manipulation de la Pile et appel de procédures : Correction

## Objectif du TP :

Le but de ce TP est de comprendre à quoi sert la pile et comment elle est gérée lors des appels et retour de procédures anisi que le passage de paramètres. et  :

Exercice 1 : programme de calcul de la moyenne

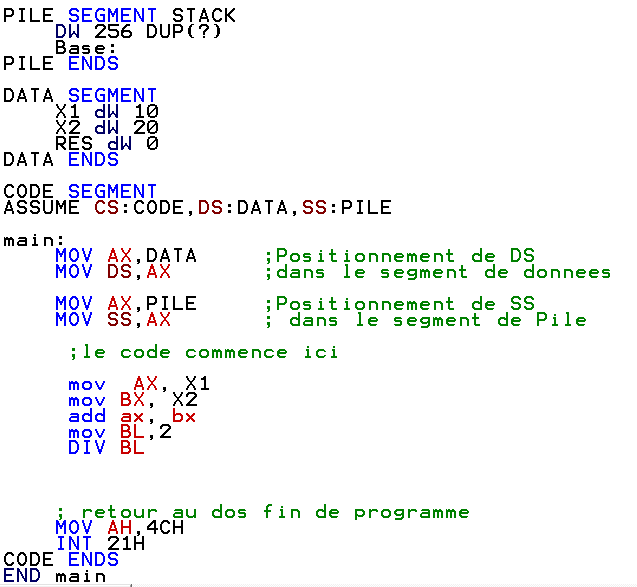
Ecrire un programme moyenne.asm qui calcule la moyenne de deux nombres x1 et x2. Ces deux nombres seront des données stockées dans le segment de données dans un premier temps et le résultat sera stocké dans une variable résultat initialisée à 0 dans le segment de données. On utilisera l’instruction DIV qui réalise une division entière par 2. Vous vérifierez la bonne valeur de résultat dans emu8086.

Exemple :

MOV AX, 25

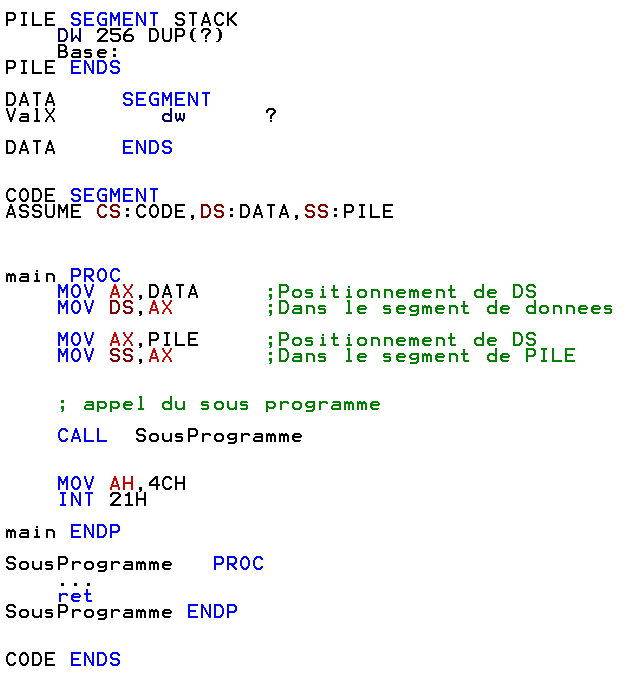
MOV BL, 2

DIV BL ; Division entière de 25 par 2. AH = 1 (reste) et AL = 12 (résultat)

****Utilisation des sous-programmes

On cherche à rendre le code de votre programme plus lisible et à le décomposer en sous procédures. Pour cela, on va créer une procédure moyenne et une procédure lire qui seront appelées dans le programme principal par l’instruction CALL.

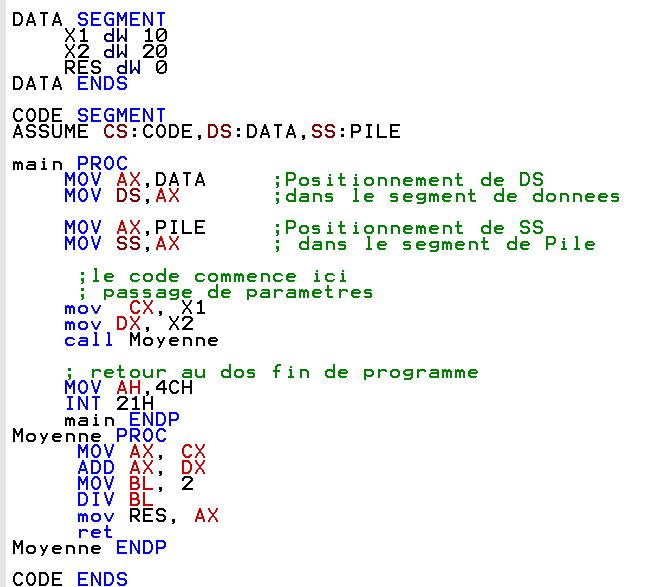
Votre fichier aura la structure suivante :

On va utiliser deux méthodes pour le passage des deux paramètres à la procédure Moyenne. La valeur résultat sera toujours affichée à l’écran.

1. Méthode n°1 : passage de paramètres par registres.

Les paramètres seront lus avant l’appel à Moyenne et stockés dans les registres CX et DX. La procédure Moyenne utilisera ces registres pour faire la moyenne. Ecrivez la procédure Moyenne en réalisant un passage de paramètres par registres et stockant le résultat dans la variable resultat.

Vous exécuterez en mode pas-à-pas le programme sous emu86 et vous serez attentif à l’évolution des registres SS, SP, BP ainsi que la zone mémoire de la pile ?



1. Méthode n°2 : passage de paramètres par la pile.

Vous allez empiler vos paramètres avant l’appel de la fonction.

L’algorithme va être le suivant :

Lecture du 1er chiffre x1 rentré au clavier

Empilement du chiffre

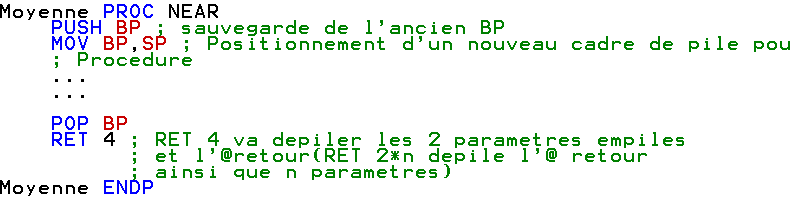
Lecture du 2ème chiffre x2 rentré au clavier

Empilement du chiffre

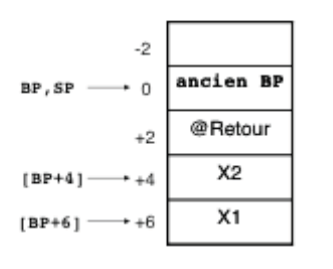
Appel de Moyenne

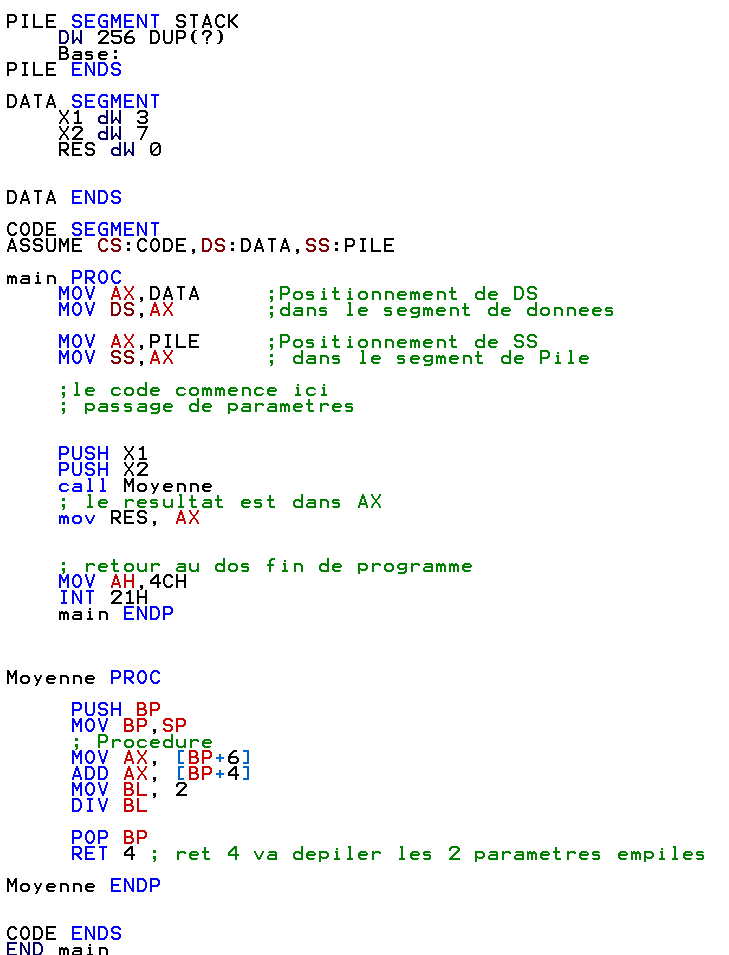
Lors de l’instruction CALL, l’adresse retour est empilée afin de pouvoir revenir au programme principal.

Votre fonction Moyenne devra obligatoirement contenir les lignes suivantes :

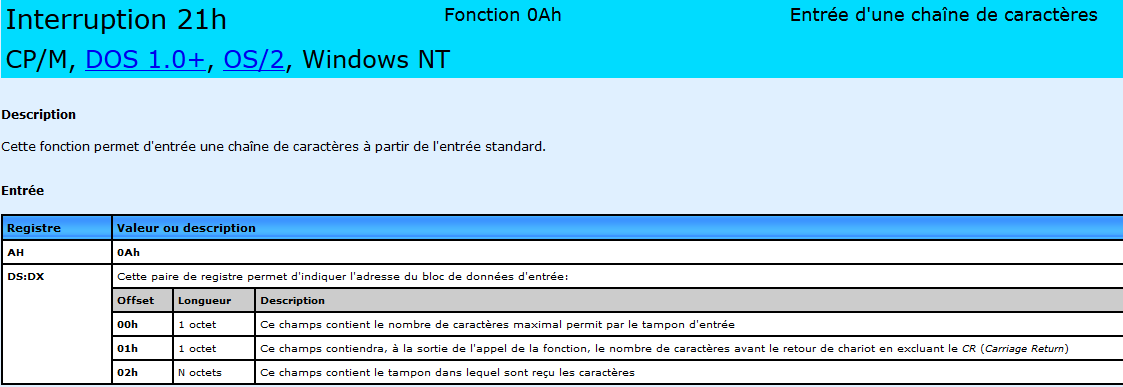


A l’intérieur de la procédure Moyenne, vous pouvez accéder aux 2 paramètres aux adresses [BP+6] et [BP+4], à l’aide d’un déplacement basé par rapport à la base de la pile





Réécrivez la procédure Moyenne et faites un passage des paramètres x1 et x2 par la pile. Pensez à écrire une procédure propre qui sauvegarde en début de procédure tous les registres qui seront utilisés. A la fin de la procédure, ces registres seront restaurés.



PILE SEGMENT STACK

DW 256 DUP(?)

Base:

PILE ENDS

DATA SEGMENT

X1 dW 0

X2 dW 0

RES dW 0

msg1 db 0Dh,0Ah, " Entrez un nombre entre -32768 et 65535: $"

bufferX1 db 7,?, 5 dup (1),'$', 0

bufferX2 db 7,?, 5 dup (2),'$', 0

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA,SS:PILE

main PROC

MOV AX,DATA ;Positionnement de DS

MOV DS,AX ;dans le segment de donnees

MOV AX,PILE ;Positionnement de SS

MOV SS,AX ; dans le segment de Pile

;le code commence ici

; passage de parametres

; Saisir la premiere chaine

; aparametre @chaine dans SI

mov SI, offset bufferX1

call SaisieChaine

; Saisir la deuxieme chaine

; aparametre @chaine dans SI

mov SI, offset bufferX2

call SaisieChaine

; On appelle la fonction ChaineToEntier pour le 1er buffer

; calcul de la valeur entiere de la chaine

; format dans la memoire ?00000$ , ?est la taille

; si on tape 123 buffer aura comme format 312300$

; on met le nombre de digit dans CX

mov CL, bufferX1[1]

mov CH, 0

; on met l'adresse du buffer dans SI

mov SI, offset bufferX1 +2

CALL ChaineToEntier

; on recupere le resultat dans X1

mov X1,AX

; On appelle la fonction ChaineToEntier pour le 2eme buffer

; on met le nombre de digit dans CX

mov CL, bufferX2[1]

mov CH, 0

; on met l'adresse du buffer dans SI

mov SI, offset bufferX2 +2

; On appelle la fonction ChaineToEntier

CALL ChaineToEntier

; on recupere le resultat dans X1

mov X2,AX

mov CX,X1

mov DX,X2

call Moyenne

; le resultat est dans AX

mov RES, AX

;retour au dos fin de programme

MOV AH,4CH

INT 21H

main ENDP

SaisieChaine PROC

; SI contient l'adresse de la zone ou on doit stocker la chaine

; print welcome message:

mov dx, offset msg1

mov ah, 9

int 21h

; input string:

mov dx, SI

mov ah, 0ah

int 21h

; affichage chaine

add SI, 1

mov dX, SI

mov ah, 09h

int 21h

ret

SaisieChaine ENDP

ChaineToEntier PROC

; parametres: CX contient le nb digit

; SI adresse du début de la chaine

; AX porte le resultat

mov AX,0

mov DX,0

boucle:

mov DL,[SI] ; on met le premier caractere dans DL

sub DL,'0' ; transformation en chiffre

ADD AX,DX ; ajout a BX

INC SI ;incrémentation de SI

cmp CX,1 ; verification si autre iteration

je fin

MOV BX,10

MUL BX

fin:

loop boucle

ret

ChaineToEntier ENDP

Moyenne PROC

MOV AX, CX

ADD AX, DX

MOV BL, 2

DIV BL

ret

Moyenne ENDP

CODE ENDS

END main

Voir solution sur saisie des entiers dans ExoMoyenneAvecSaisieNombres.asm

Exercice 3 :

Réécrivez la procédure Moyenne de sorte que quand vous retournez dans le programme principal, le résultat soit stocké en haut de la pile, et non pas dans la variable resultat stockée dans le segment de données. Pour récupérer le résultat après CALL Moyenne, vous n’aurez qu’à faire un POP AX.

Dans votre programme, vous ferez aussi un affichage du résultat par une procédure affiche.