TP N° 3

PROGRAMMATION EN ASSEMBLEUR 8086

Structure de contrôle en assembleur

En assembleur il n'y a pas de structures de contrôles (boucles, choix multiples…etc.) comme on peut les trouver dans les langages structurés C ou Pascal.
Nous pouvons réaliser des structures assembleur équivalentes en C ou Pascal, en combinant plusieurs instructions assembleur.

- **Tableau récapitulatif des sauts conditionnels**

✓ **Entiers signés**

Conditions après CMP a, b	mnémonique	Conditions de branchement
a = b	JE / JZ	ZF = 1
a ≠ b	JNE / JNZ	ZF = 0

✓ **Entiers non signés**

Conditions après CMP a, b	mnémonique	Conditions de branchement
a > b	JA / JNBE	CF = 0 et ZF = 0
a ≥ b	JAE / JNB /JNC	CF = 0
a < b	JB / JNAE / JC	CF = 1
a ≤ b	JBE / JNA	CF = 1 ou ZF = 1

Correspondance des operateurs avec le langage C

Opérateur C	Instruction 80x86	
+	ADD	addition ;
-	SUB	soustraction ;
<<	SHL	décalage à gauche ;
>>	SHR	décalage à droite ;
|	OR	ou bit à bit ;
&	AND	et bit à bit ;
^	XOR	ou exclusif bit à bit.

Equivalent de structures algorithmique avancées

Instructions de saut conditionnelles : Donner des exemples de structures conditionnelles classiques et leurs implantations en assembleur

I/

Si X > Y alors **<Instructions>** **Fsi**	Assembleur

Si X < Y alors **<Instructions>** **Fsi**	Assembleur

Si X = Y alors **<Instructions>** **Fsi**	Assembleur

Si X = 0 alors **<Instructions>** **Fsi**	Assembleur

II/

En combinant sauts conditionnel et inconditionnel, on peut mettre en place des structures plus complexes comme donner un exemple :

Exemple

a)

Si Var_a > Var_b alors **Instructions 1** **Sinon** **Instructions 2** **Finsi**	Assembleur

b)

Si X = 0 alors **<Instructions1>** **Sinon** **<Instructions2>** **Finsi**	Assembleur

III/ Boucles

1)

Tantque var_a > Var_b faire **<instructions >** **Fintantque**	Assembleur

2)

Do <instructions > while (a>b) faire **<instructions >** **Finfaire**	Assembleur

3)

for (i=0; i<=10; i++) **{** **<instructions>** **}**	Assembleur

4)

switch (a) **{** **case 'a': <instruction_a>** **break ;** **case 'b': <instruction_b>** **break ;** **default: <instruction_c>** **}**	Assembleur

VI) Exercices : (utiliser les structures algorithmique cité ci-dessus)

- **Ecrire un programme assembleur qui calcul n ! , n > 0**
  0 ! = 1, 1 ! = 1 , n ! = 1 x 2 x 3 x ……n
  Remarque : Mul dx multiplication de dx par ax
  -Exemple : prendre n = 7 , 7 ! = $(5040)_{10}$ = $(13B0)_{16}$

- **Ecrire un programme Y^x**
  Y^x = Y x Y x Y x ….. (X fois) Exemple Y = 4 , X = 3,
  Y^x = 4 x 4 x 4 (3 fois) 4^3 = $(64)_{10}$ = $(40)_{16}$
- **Ecrire un programme en langage assembleur 8086 qui calcule n termes de la suite de Fibonacci (Ici n = 16), et les range en mémoire à la suite de n U_0=1, U_1=1, $U_{n+1}=U_n + U_{n-1}$**

  Après l'exécution du programme la mémoire contient donc les valeurs suivantes, 16, 1,1,2, 3, 5, 8, 13,21,…. Etc

- **Ecrire un programme assembleur qui calcul la somme des éléments d'une suite de nombre le résultat sera dans AX , La suite : pour n = 1 à 10 , A = A+ 2(A+ r) , A_0 = 2 , r = 5**
- **Ecrire un programme assembleur qui cherche le caractère 'c' dans une suite de caractère terminé par le symbole $, on le range dans le registre CX la suite ' Tp architecture des ordinateurs $'**
  Remarque : déclarer les variables dans la partie déclaration du programme

Factoriel	Puissance

Suite de fibonnacci .