Université de Jijel 2ème Année Licence Faculté des sciences exactes et informatique Matière : Architecture des ordinateurs

Département d'Informatique 2019-2020

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**



**Equivalent de structures algorithmique avancées**

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

**I/**

Assembleur

**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**

**II/**

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

**Exemple**

Assembleur

**Si Var\_a > Var\_b alors**

**Instructions 1**

**Sinon**

**Instructions 2**

**Finsi**



Assembleur

**Si X = 0 alors**

**<Instructions1>**

**Sinon**

**<Instructions2>**

**Finsi**

**III/ Boucles**

**1)**

Assembleur

**Tantque var\_a > Var\_b faire**

**<instructions >**

**Fintantque**

**2)**

Assembleur

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

**<instructions >**

**Finfaire**

**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) 43 = (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 U0=1, U1=1, Un+1=Un + Un-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) , A0 = 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**

Puissance

Factoriel

**Suite de fibonnacci**

.