**Introduction :**

Le processeur 8086 d'Intel est à la base des processeurs Pentium actuels.

Les processeurs successifs (de PC) se sont en effet construits petit à petit   en   ajoutant   à   chaque   processeurs   des   instructions   et   des fonctionnalités  supplémentaires,  mais  en  conservant  à  chaque  fois  les spécificités  du  processeur  précédent.  C'est  cette  façon  d'adapter  les processeurs à chaque étape qui permet qu'un ancien programme écrit pour un 8086 fonctionne toujours sur un nouvel ordinateur équipé d'un Pentium IV.

**REPONSES :**

1. **Les registres à usage générales de processeur 8086 sont :**

* AX, BX, CX, DX

1. **Les registres segments de processeur 8086 sont :**

##### **Le registre CS (code segment) :**

Il pointe sur le segment qui contient les codes des instructions du programme en cours.

***Remarque :***

Si la taille du programme dépasse les 65535 octets alors on peut diviser le code sur plusieurs segments (chacun ne dépasse pas les 65535 octets) et pour basculer d'une partie à une autre du programme il suffit de changer la valeur du registre CS et de cette manière on résout le problème des programmes qui ont une taille supérieure à 65535 octets.

##### **Le registre DS (Data segment) :**

Le  registre  segment  de  données  pointe  sur  le  segment  des variables  globales  du  programme,  bien  évidemment  la  taille  ne  peut excéder 65535 octets (si on a des données qui dépassent cette limite, on utilise la même astuce citée dans la remarque précédente mais dans ce cas on change la valeur de DS).

##### **Le registre ES (Extra segment) :**

Le registre de données supplémentaires ES est utilisé par le microprocesseur lorsque l'accès aux autres registres est devenu difficile ou impossible pour modifier des données, de même ce segment est utilisé pour le stockage des chaînes de caractères.

##### **Le segment SS (Stack segment) :**

Le registre SS pointe sur la pile : la pile est une zone mémoire ou on peut sauvegarder les registres ou les adresses ou les données pour les récupérer après l'exécution d'un sous programme ou l'exécution d'un programme d'interruption, en général il est conseillée de ne pas changer le contenu de ce registre car on risque de perdre des informations très importantes  (exemple  les  passages  d'arguments  entre  le  programme principal et le sous programme)

1. **IP - Instruction Pointer :**

Ou Compteur de Programme, contient l'adresse de l'emplacement mémoire où se situe la prochaine instruction à exécuter. Autrement dit, il doit indiquer au processeur la prochaine instruction à exécuter. Le registre IP est constamment modifié après l'exécution de chaque instruction afin qu'il pointe sur l'instruction suivante.

1. **Explication de CF = 1  (CF : Carry Flag - Indicateur de retenue) :**

       A la suite d'une opération, si le résultat est codé sur un nombre supérieur de bits, le bit de trop sera mis dans CF. Plusieurs instructions peuvent modifier son état : CLC pour le mettre à 0, STC à 1 et CMC inverse sa valeur.

1. **La fonction d’utilitaire debug :**

Debug est un utilitaire mis à la disposition des utilisateurs des PC pour la mise au point des programmes. Il convenait parfaitement à cette tâche quand on programmait les premiers PC en assembleur ce qui ne se fait plus que très exceptionnellement aujourd'hui. Le langage assembleur ne sert plus que pour des tâches dites de bas niveau. Entendons par-là les fonctions très proches du hardware.

Actuellement les programmeurs utilisent des programmes de mise au point beaucoup plus performants. Les débogueurs actuels sont des outils graphiques adaptés aux langages de haut niveau et au développement d'applications graphiques.

L'intérêt du programme DEBUG est qu'il est fourni avec tous les systèmes d'exploitations de Microsoft depuis le DOS 2.0 jusqu'aux plus récents comme Windows XP. Ce petit programme est présent dans tous les PC et même sur les disquettes de démarrage.

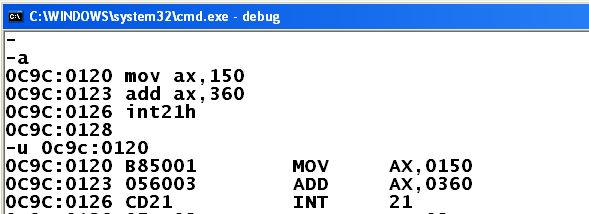
**Travaille demandé :**

1. **Remplissage de tableau :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Adresse** | **Valeur Hexa** | **Carecter ASCII** | **Nouvelle valeur** |
| **154E:100** | **0A** | **.** | **a** |
| **154E:103** | **0B** | **.** | **B** |
| **17A0 :105** | **0A** | **.** | **A** |
| **1730 :250** | **25** | **%** | **25** |

**Il s’agit de segment DS.**

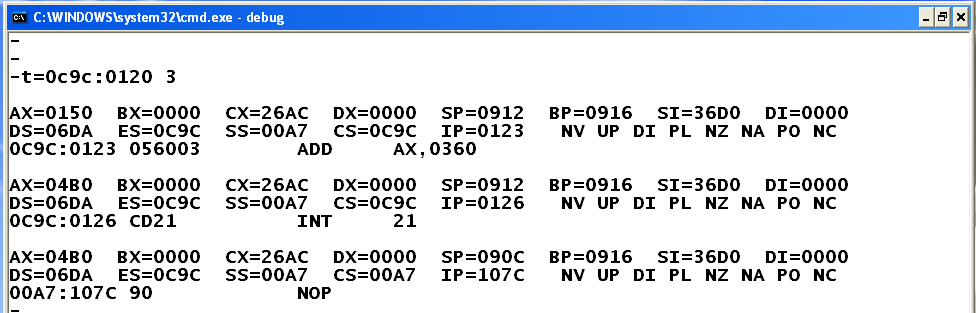
1. **Adressage Directe :**
2. Le programme assembleur qui permet d’additionner les nombres 150 et 360 :



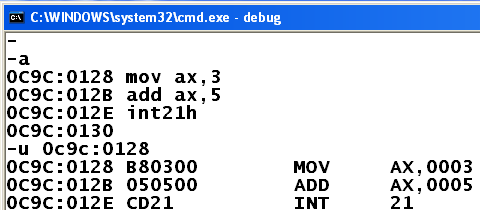
1. les adresses de programme + les mnémoniques :

|  |  |  |
| --- | --- | --- |
| **Adresse** | **Instruction en mnémonique** | **Code opératoire** |
| **0C9C :0120**  **0C9C :0123**  **0C9C :0126** | **mov ax,0150**  **add ax,0360**  **int21h** | **B85001**  **056003**  **CD21** |

L’exécution de programme :

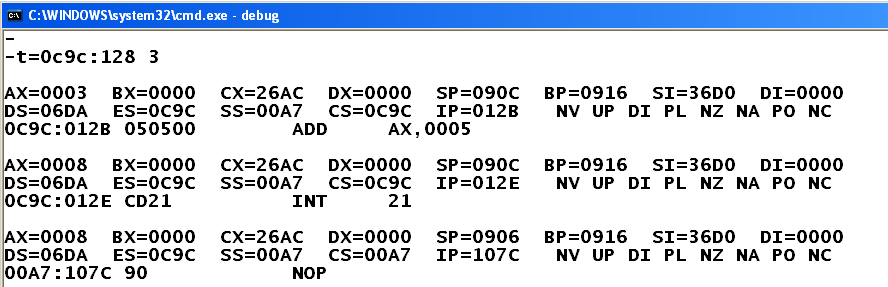


1. **Application**
2. Programme d’addition 3+5 ;

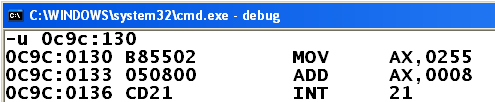


|  |  |  |
| --- | --- | --- |
| **Adresse** | **Instruction en mnémonique** | **Code opératoire** |
| **0C9C :0128**  **0C9C :012B**  **0C9C :012E** | **mov ax,3**  **add ax,5**  **int21h** | **B80300**  **050500**  **CD21** |

L’exécution de programme :

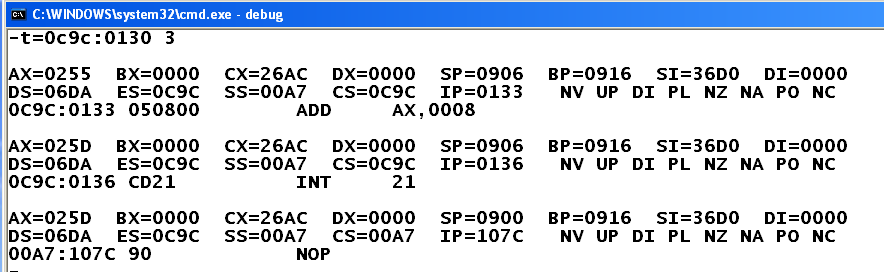


1. Programme d’addition 3+5 ;

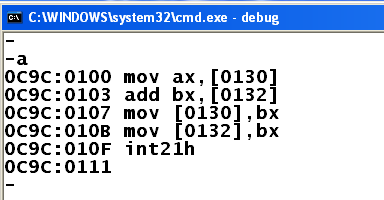


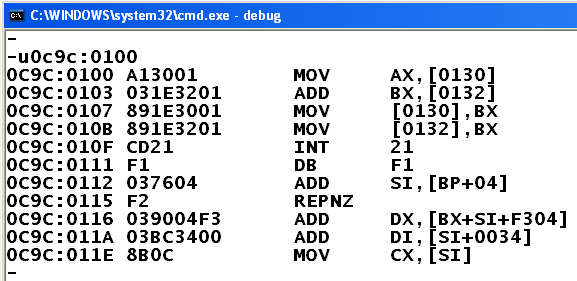
|  |  |  |
| --- | --- | --- |
| **Adresse** | **Instruction en mnémonique** | **Code opératoire** |
| **0C9C :0130**  **0C9C :0133**  **0C9C :0136** | **mov ax,3**  **add ax,5**  **int21h** | **B85502**  **050800**  **CD21** |

**L’exécution de programme ;**



1. **Programme assembleur qui permet de permuter les données enregistrées respectivement aux adresses 130h et 132h :**





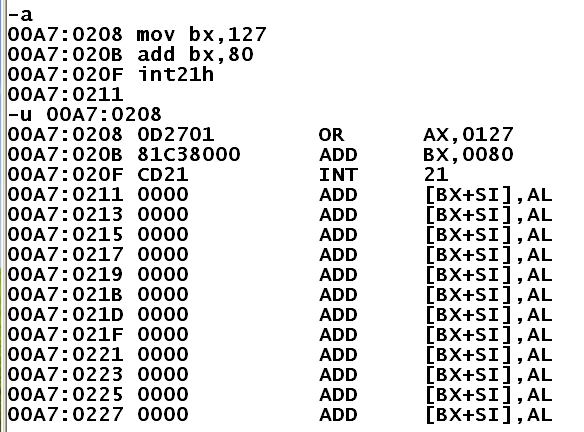
1. **Les adresses :**

|  |  |  |
| --- | --- | --- |
| **Adresse** | **Instruction en mnémonique** | **Code opératoire** |
| **0C9C :0100**  **0C9C :0103**  **0C9C :0107**  **0C9C :010B**  **0C9C :010F** | **mov ax,[0130]**  **add bx,[0132]**  **mov [0130],bx**  **mov [0132],bx**  **int21h** | **A13001**  **031E3201**  **891E3001**  **891E3201**  **CD21** |

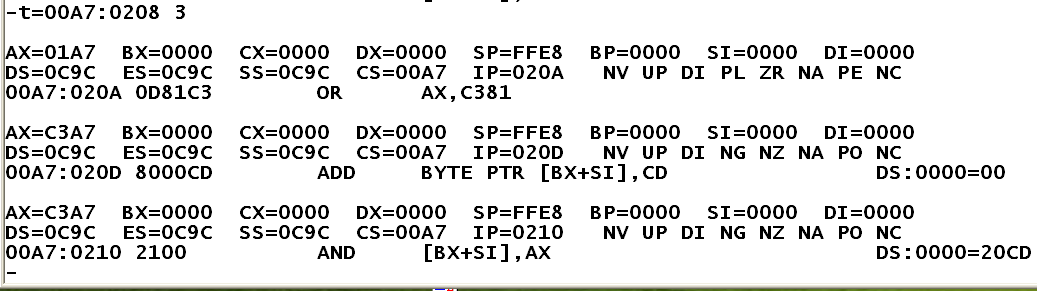
1. **Application :**

* 127 + 80

Programme

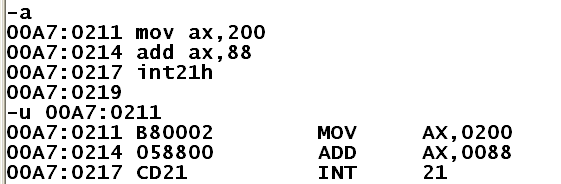


**Exécution**

****

* 200 + 88

**Programme**



**L’exécution**

