

Exercice 1 (12 Pts) (Temps recommandé : 30 mn)

On donne les valeurs des registres index suivants : SI = 1000H ; DI = 5000H ;

On considère également le programme ASM 80x86 suivant :

| | | |
|-------------|------------------------------|-----|
| 5000 : 0500 | MOV CX , 10 ; | (1) |
| 5000 : 0502 | MOV BX , [SI] ; | (2) |
| 5000 : 0504 | BCL : MOV [DI] , BX ; | (3) |
| 5000 : 0506 | ADD SI , 2 ; | (4) |
| 5000 : 0508 | ADD DI , 2 ; | (5) |
| 5000 : 050A | (instruction manque) ; | (6) |
| 5000 : 050C | JNZ BCL ; | (7) |
| 5000 : 050E | END ; | (8) |

1. Trouver l'instruction manquante en ligne (6). Justifier votre réponse (1pt)

(6) : DEC CX ; Le retour vers BCL est conditionné par une modif. du compteur

2. Parmi les 5 propositions suivantes, indiquer celle(s) qui est (sont) VRAIE(S) ? (1pt)

- Ce programme transfère 10 données de l'@ DI vers l'@ SI ;
- Ce programme transfère 10 fois la même donnée de l'@ SI vers l'@ DI ;
- Ce programme transfère 5000 données de l'@ SI vers l'@ DI ;
- Ce programme transfère 10 instructions de l'@ SI vers l'@ DI ;
- Ce programme copie 10 données de l'@ DI vers le registre BX ;

TOUTES ces propositions sont FAUSSES sauf (b) (voir 4/°)

3. Indiquer la taille de l'instruction (6). Justifier votre réponse (1pt)

Taille = (@_suiv - @_prés) = 050C – 050A = 2 octets

4. Indiquer de manière précise et concise la fonction de ce programme. (1pt)

Ce programme transfère 10 données de l'@ SI vers l'@ DI (via BX) (ou reprendre l'expression (b)).

5. Le programme tel qu'il est présenté ci-dessus comporte une erreur générant une impossibilité d'exécution : trouver cette erreur et proposer une correction. Justifier votre réponse (1.5pt)

ERREUR : CS = DI = 5000 H : impossible car CS & DI sont des @ absolues, donc elles ne peuvent être égales ;

Proposition de correction : CS = 7000 H (par exemple) (≠ de DI et de SI)

Réponse/ERREUR « admise » : [« BCL » doit être en ligne (2)]

6. On remplace les instructions (2) et (3) par les instructions suivantes :

(MOV BX , [SI] ;) est remplacée par (MOV BL , [SI] ;)

(MOV [DI] , BX ;) est remplacée par (MOV [DI] , BL ;)

Quelle autre modification faudra-t-il apporter au programme ci-haut pour reproduire son résultat ? (1.5pt)

Trois modifications : (MOV CX , 20 ;) & (ADD SI , 1 ;) & (ADD DI , 1 ;)

7. On suppose une pile avec un pointeur SP = 0 ; cette pile est-elle complètement vide ou complètement pleine ? (justifier) (1.5pt)

« complètement pleine » car toute la zone entre SP_max et 0 remplie : SP pointe tjrs sur un octet « libre » : ici, « libre » = 0 .

8. Quelle devra être la valeur minimale de SP (exprimée en hexadécimal) pour que la pile puisse recevoir les données issues de (SI) ? (1.5pt)

Le programme ci-haut transfère 10 données (valeur de CX) de 2 octets chacune (taille de BX = 2 octets) ; on doit pouvoir stocker 20 octets en pile, donc (SP) devra être \geq à 20 = 14H au minimum.

9. Réécrire le programme ci-haut de façon à transférer les données depuis la pile. (2pts)

| | | | |
|-------------|------------|----------------------------|---------------------|
| 5000 : 0500 | | MOV CX , 10 ; | (1) |
| 5000 : 0502 | | MOV BX , [SI] ; | (2) .. → à enlever |
| | | POP BX ; | (2) |
| 5000 : 0504 | BCL | : MOV [DI] , BX ; | (3) |
| 5000 : 0506 | | ADD SI , 2 ; | (4) |
| 5000 : 0506 | | ADD DI , 2 ; | (5) |
| 5000 : 0508 | | DEC CX ; | (6) .. → inst manq. |
| 5000 : 050A | | JNZ BCL ; | (7) |
| 5000 : 050C | | END ; | (8) |

Exercice 2 (8 Pts) (Temps recommandé : 30 mn)

Rédiger un programme en ASSEMBLEUR 80x86 qui calcule la somme des 100 premiers nombres pairs, saisis depuis une zone RAM commençant à l'adresse pointée par (SI) (SI quelconque).

Attention : Tous les nombres saisis depuis SI ne sont pas nécessairement « pairs ».

| | | |
|-------------------|-----------------------|---------------------------------------------------------------|
| | MOV CX , 100 ; | <u>Compteur</u> " 100 nbres pairs" ds. CX |
| | MOV DX , 0 ; | <u>Résultat (somme)</u> dans DX |
| BCL1 : | MOV AX , [SI] ; | Saisi .. |
| | MOV BX, AX ; | .. puis .. <u>Sauvegarde</u> .. |
| | AND BX , 1 ; | .. puis <u>masque</u> pour extraire le bit (b0) : parité |
| | JZ BCL2 ; | <u>si</u> (b0=0/ <u>cas impaire</u>), <u>alors</u> saut BCL2 |
| | ADD DX, AX ; | <u>sinon</u> (b0=1/ <u>cas paire</u>), <u>MAJ (Somme)</u> .. |
| | JMP Prochain ; | .. et on ne fait pas d'incrément sur CX . |
| BCL2 : | INC CX ; | .. cas (impaire) : CX= CX+1(itération non ; comptabilisée) |
| Prochain : | ADD SI , 2 ; | prépare <u>prochaine donnée à traiter</u> |
| | LOOP BCL1 ; | Retour BCL1 <u>Tant que</u> CX≠0 |
| | End ; | |

Ce qui doit apparaitre essentiellement c'est :

- 1- Acquisition depuis [SI] ;
- 2- Masque « 0000 ..1 »
- 3- Sauvegarde
- 4- Saut conditionnel cohérent (JZ, JNZ, JMP ..)
- 5- INC SI dans les 2 cas