Antoine Gicquel
A2

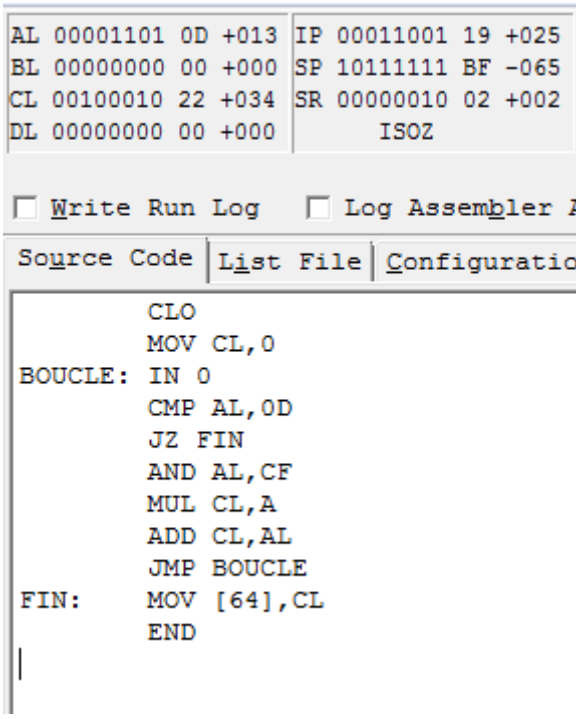
TP4 Assembleur : E/S en ASCII

P. Carreno - P. Portejoie

Exercices 4-1 et 4-2

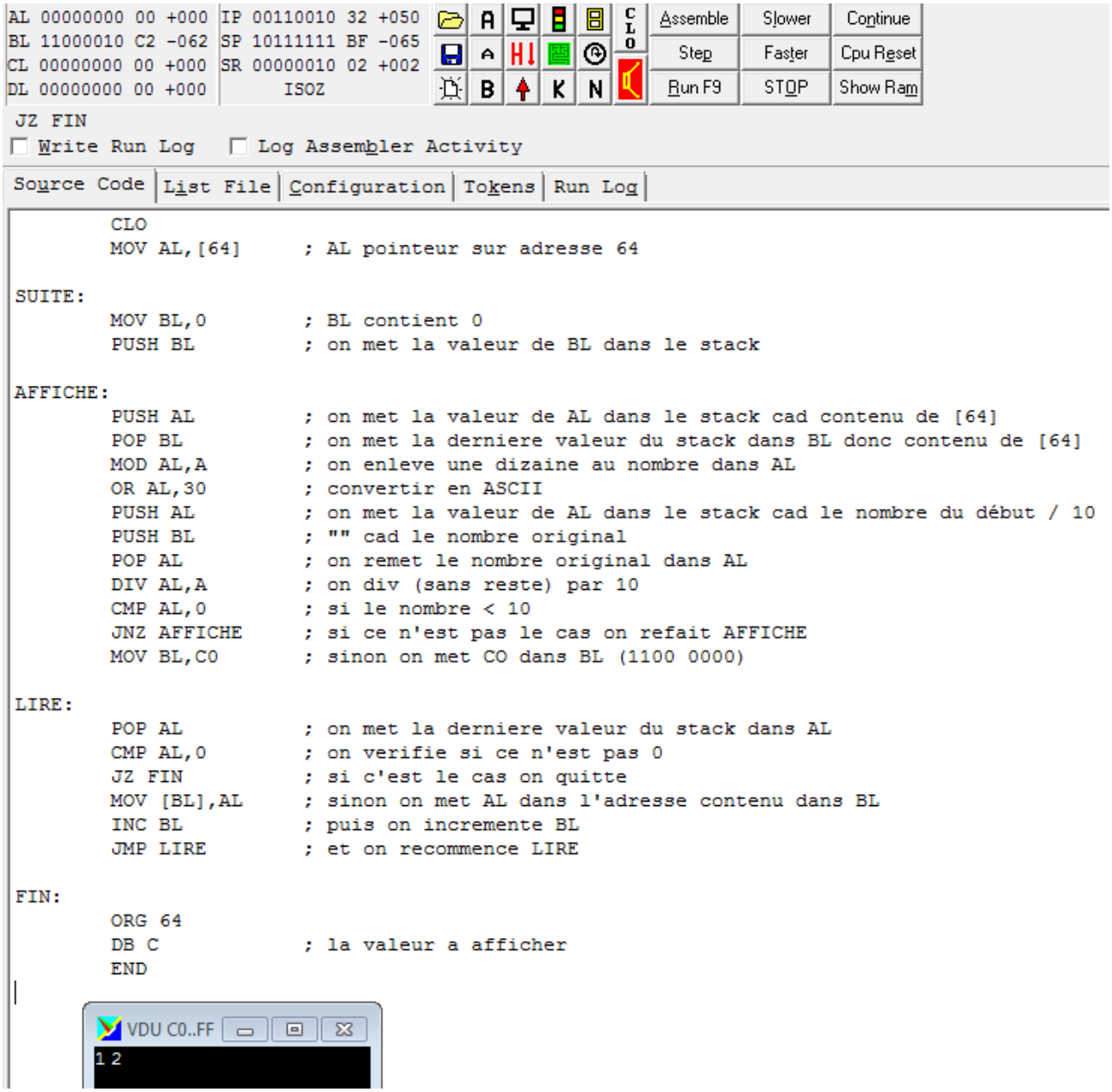
Reprenez les programmes des exercices 4-1 et 4-2 du TD et faites-les fonctionner. Vérifiez la conformité des résultats.

1)



Essai avec les nombres 3 et 4 qui donne 34 en CL.

2)



Exemple avec C qui vaut 12.

CODE :

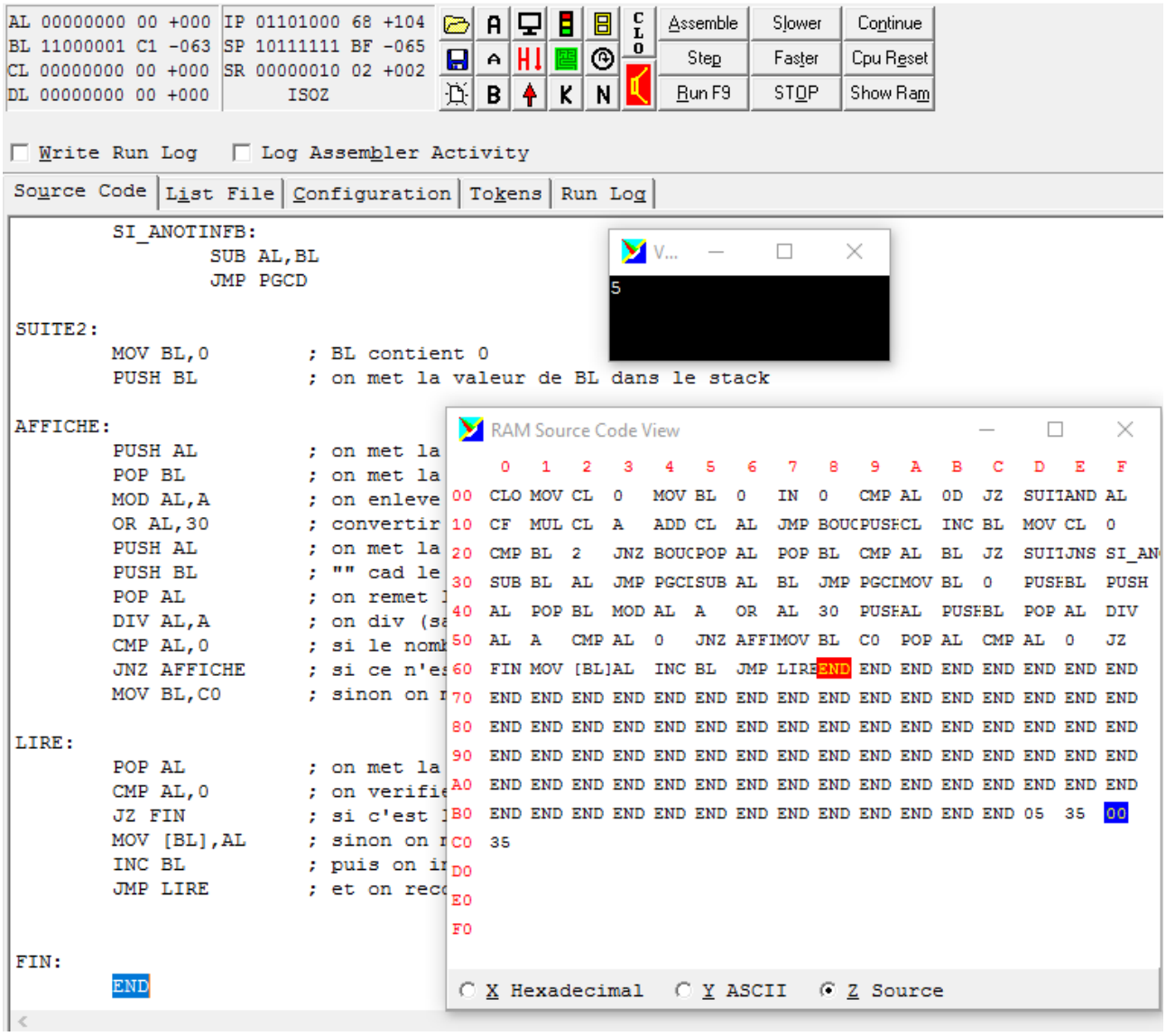
```
        CLO
        MOV AL,[64]     ; AL pointeur sur adresse 64
SUITE:
        MOV BL,0        ; BL contient 0
        PUSH BL         ; on met la valeur de BL dans le stack
AFFICHE:
        PUSH AL         ; on met la valeur de AL dans le stack cad contenu de [64]
        POP BL          ; on met la derniere valeur du stack dans BL donc contenu de [64]
```

```
        MOD AL,A        ; on enleve une dizaine au nombre dans AL
        OR AL,30        ; convertir en ASCII
        PUSH AL                 ; on met la valeur de AL dans le stack cad le nombre du début / 10
        PUSH BL                 ; "" cad le nombre original
        POP AL                  ; on remet le nombre original dans AL
        DIV AL,A        ; on div (sans reste) par 10
        CMP AL,0        ; si le nombre < 10
        JNZ AFFICHE ; si ce n'est pas le cas on refait AFFICHE
        MOV BL,C0       ; sinon on met CO dans BL (1100 0000)

LIRE:
        POP AL                  ; on met la derniere valeur du stack dans AL
        CMP AL,0        ; on verifie si ce n'est pas 0
        JZ FIN          ; si c'est le cas on quitte
        MOV [BL],AL     ; sinon on met AL dans l'adresse contenu dans BL
        INC BL          ; puis on incremente BL
        JMP LIRE        ; et on recommence LIRE

FIN:
        ORG 64
        DB C            ; la valeur a afficher
        END
```

Exercice 4-3

Ré-écrivez le programme de calcul du PGCD vu lors du TD-TP n°2 en lui ajoutant la saisie des 2 entiers à traiter ainsi que l'affichage du résultat.

Test avec 25 et 5 :



CODE:

```
  CLO
  MOV CL,0
  MOV BL,0
BOUCLE:
  IN 0
  CMP AL,0D     ; si on appuye sur entrée
  JZ SUITE
  AND AL,CF
  MUL CL,A
  ADD CL,AL
  JMP BOUCLE

SUITE:
  PUSH CL
  INC BL
  MOV CL,0
  CMP BL,2
  JNZ BOUCLE
```

```
  POP AL
  POP BL

PGCD:
  CMP AL,BL
  JZ SUITE2
  JNS SI_ANOTINFB
  SI_AINFB:
          SUB BL,AL
          JMP PGCD
  SI_ANOTINFB:
          SUB AL,BL
          JMP PGCD

SUITE2:
  MOV BL,0      ; BL contient 0
  PUSH BL               ; on met la valeur de BL dans le stack

AFFICHE:
  PUSH AL               ; on met la valeur de AL dans le stack cad contenu de [64]
  POP BL                ; on met la derniere valeur du stack dans BL donc contenu de [64]
  MOD AL,A      ; on enleve une dizaine au nombre dans AL
  OR AL,30      ; convertir en ASCII
  PUSH AL               ; on met la valeur de AL dans le stack cad le nombre du début / 10
  PUSH BL               ; "" cad le nombre original
  POP AL                ; on remet le nombre original dans AL
  DIV AL,A      ; on div (sans reste) par 10
  CMP AL,0      ; si le nombre < 10
  JNZ AFFICHE ; si ce n'est pas le cas on refait AFFICHE
  MOV BL,C0     ; sinon on met CO dans BL (1100 0000)

LIRE:
  POP AL                ; on met la derniere valeur du stack dans AL
  CMP AL,0      ; on verifie si ce n'est pas 0
  JZ FIN            ; si c'est le cas on quitte
  MOV [BL],AL   ; sinon on met AL dans l'adresse contenu dans BL
  INC BL            ; puis on incremente BL
  JMP LIRE      ; et on recommence LIRE


FIN:
  END
```

Exercice 4-4 (si le temps le permet...)

Ecrivez un programme qui trie sur place (ie sans zone de stockage intermédiaire), par ordre croissant, des caractères situés à partir du début de la mémoire vidéo (vous pourrez ainsi visualiser leur progression lors du tri). L'initialisation en sera faite à l'assemblage.

Le nombre de caractères à traiter sera disponible à l'adresse de votre choix et devra faire également l'objet d'une initialisation à l'assemblage.

Algorithme de tri proposé – Tri à bulles :

Il existe de nombreux algorithmes de tri. Vous mettrez en oeuvre le tri à bulle dont l'algorithme vous est donné ci-après. Ce n'est pas le plus performant mais c'est un des plus simples à programmer.

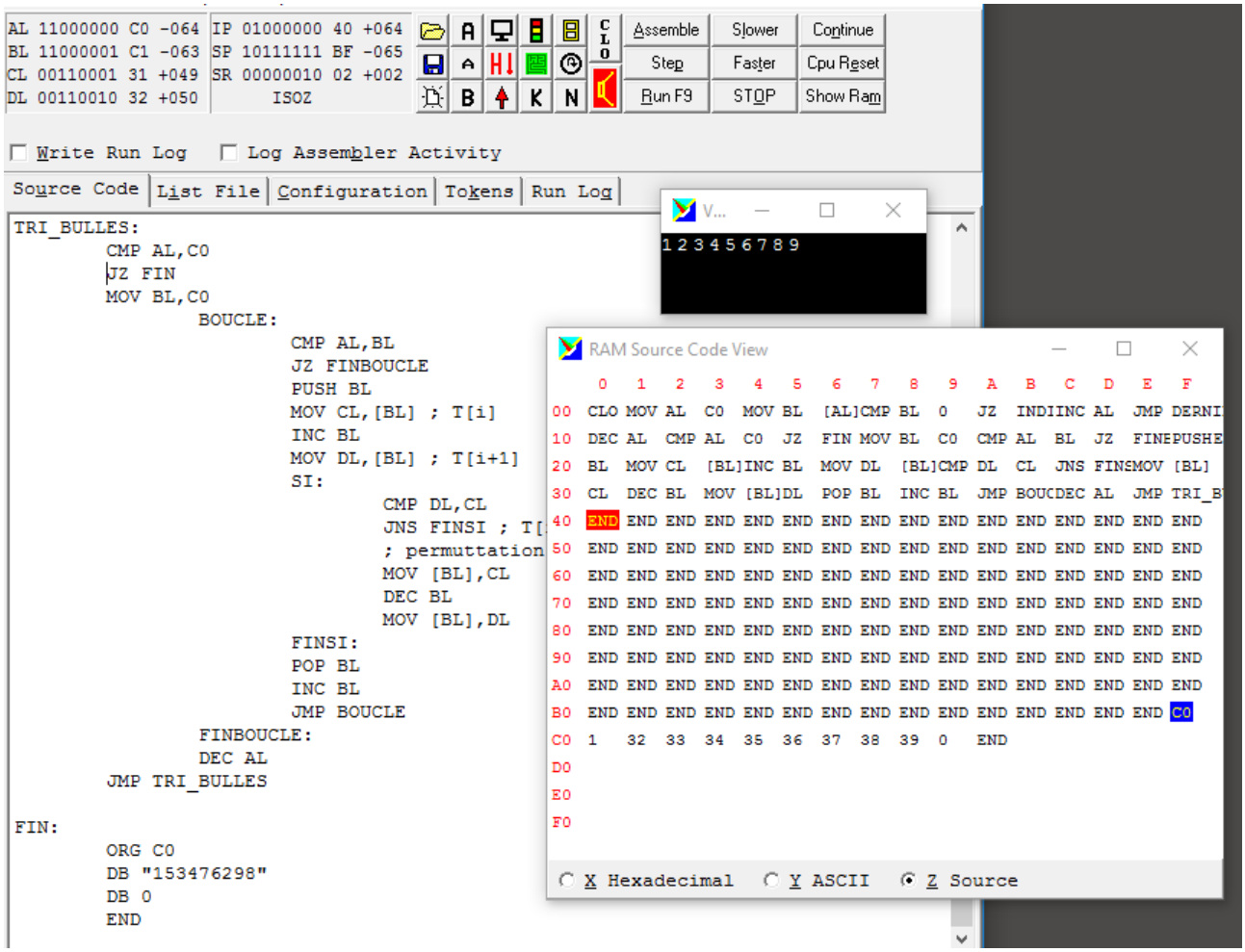
Son principe est de parcourir la liste (a_1, a_2, ... , a_n) en intervertissant toute paire d'éléments consécutifs (a_{i-1}, a_i) non ordonnés. Ainsi après le premier parcours, l'élément maximum se retrouve en a_n. On suppose que l'ordre croissant s'écrit de gauche à droite (à gauche le plus petit élément, à droite le plus grand élément).

On recommence l'opération avec la nouvelle sous-suite (a_1, a_2, ... , a_{n-1}), et ainsi de suite jusqu'à épuisement de toutes les sous-suites (la dernière est un couple). A tout instant, la partie gauche du tableau est non triée alors que la partie droite l'est. L'algorithme s'arrête lorsque la partie non triée devient vide.

Le nom de tri à bulle vient de ce qu'à la fin de chaque itération interne, le plus grand élément de chaque sous-suite se déplace vers la droite à l'image d'une bulle.

```
début
  j := dernier_indice
  ttque j <> 0          ; tant que la partie non triée n'est pas vide
      i := 0
      ttque i <> j      ; remontée de la plus grande valeur du sous-tableau non trié
           si T[i+1] < T[i] alors permuttation de T[i+1] avec T[i]  fsi
           i := i+1
      finttque
      j := j-1          ; la dernière valeur traitée est considérée comme triée
  finttque              ; il y a donc une valeur de moins dans le sous-tableau non trié
fin
```

Essai avec "153476298"



CODE:

```
  CLO
  MOV AL,C0

; AL contiendra le dernier indice
DERNIER_INDICE:
  MOV BL,[AL]
  CMP BL,0
  JZ INDICE
  INC AL
  JMP DERNIER_INDICE
INDICE:
  DEC AL ; de 0 a TAILLE-1

TRI_BULLES:
  CMP AL,C0
  JZ FIN
  MOV BL,C0
        BOUCLE:
              CMP AL,BL
              JZ FINBOUCLE
              PUSH BL
              MOV CL,[BL] ; T[i]
              INC BL
              MOV DL,[BL] ; T[i+1]
              SI:
```

```
                        CMP DL,CL
                        JNS FINSI ; T[i+1] > T[i]
                        ; permuttation
                        MOV [BL],CL
                        DEC BL
                        MOV [BL],DL
                FINSI:
                POP BL
                INC BL
                JMP BOUCLE
        FINBOUCLE:
        DEC AL
  JMP TRI_BULLES

FIN:
  ORG C0
  DB "153476298"
  DB 0
  END
```