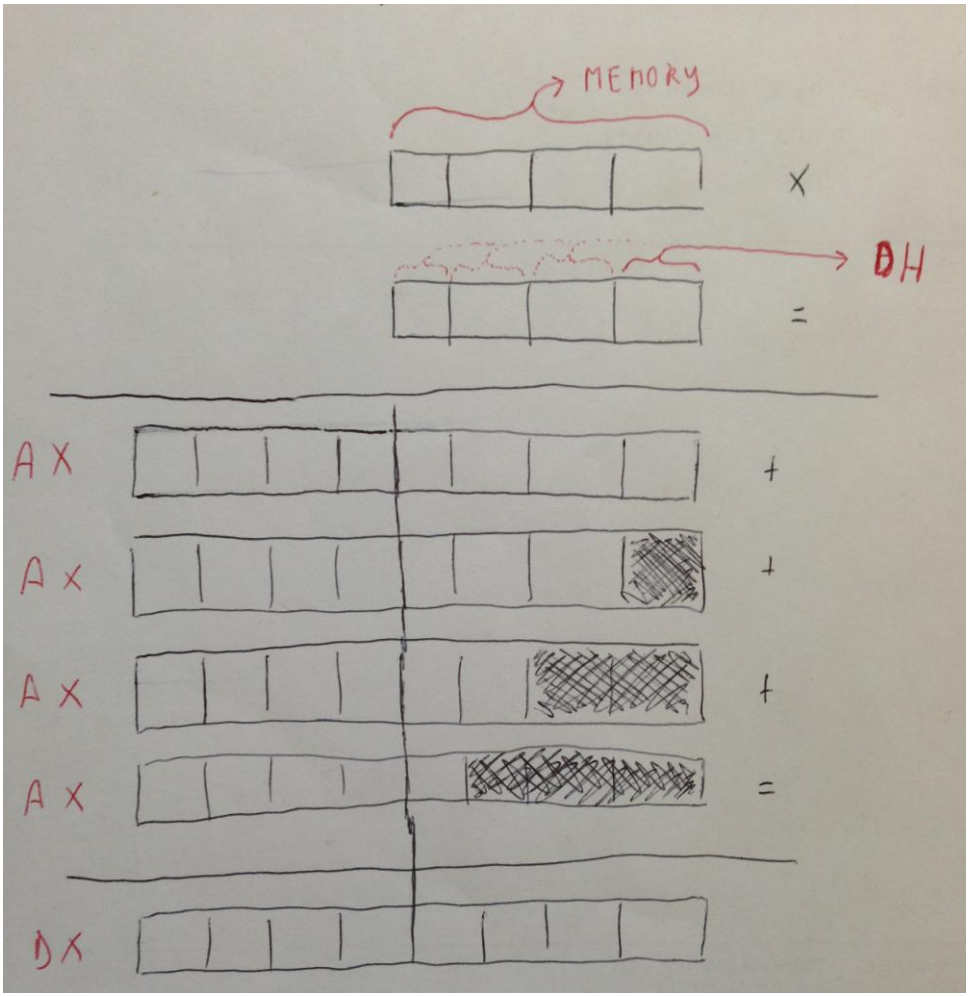
```
; Computer Architectures (02LSEOV) - Exam simulation (18/12/2013).
; Solution proposed by Alessio Canepa

; This solution has been written in order to be as readable
; as possible, without taking into account optimization.
; It has been tested on some different inputs and it
; worked. I hope it doesn't contain any error.
; User input and output have not been implemented.

N   EQU 3
       .MODEL small
       .STACK
       .DATA
FIRST       DB 01100100b,00100101b,01000001b    ;this is 1210,0211,1001
SECOND      DB 00001001b,01010110b,00010001b    ;this is 0021,1112,0101
FIRST_SORTED_HI_TO_LO   DB N DUP(?)
SECOND_SORTED_LO_TO_HI  DB N DUP(?)
ADDED       DW N  DUP(?)
BY_THREE_MULTIPLIED_FIRST   DW N DUP(?)
BY_THREE_MULTIPLIED_SECOND  DW N DUP(?)
BY_TWO_MULTIPLIED_FIRST     DW N DUP(?)
BY_TWO_MULTIPLIED_SECOND    DW N DUP(?)
MULTIPLIED      DW N DUP(?)
       .CODE
       .STARTUP

;ITEM 0

        XOR DI,DI
        MOV CX,N                            ;this is the number of numbers

;in order to make it simple and more readable, firstly copy the two arrays
;into the final arrays, then sort them with bubble sort

;copy

LOOP1:  MOV BL,FIRST[DI]
        MOV BH,SECOND[DI]
        MOV FIRST_SORTED_HI_TO_LO[DI],BL
        MOV SECOND_SORTED_LO_TO_HI[DI],BH
        INC DI
        LOOP LOOP1
;sort

        MOV CX,N
LOOP3:  PUSH CX
        MOV CX,N-1                          ;there are two nested loops
        XOR DI,DI
LOOP2:  MOV BL,FIRST_SORTED_HI_TO_LO[DI]    ;read one number from each array
        MOV BH,SECOND_SORTED_LO_TO_HI[DI]
        CMP BL,FIRST_SORTED_HI_TO_LO[DI+1]  ;...and compare it with the following one
        JG  CNT1                            ;if the first is not greater than the second...

        MOV AL,FIRST_SORTED_HI_TO_LO[DI+1]  ;...swap!
        MOV FIRST_SORTED_HI_TO_LO[DI],AL
        MOV FIRST_SORTED_HI_TO_LO[DI+1],BL

CNT1:   CMP BH,SECOND_SORTED_LO_TO_HI[DI+1] ;do the same for the other array
```

```
        JL  CNT2

        MOV AL,SECOND_SORTED_LO_TO_HI[DI+1]
        MOV SECOND_SORTED_LO_TO_HI[DI],AL
        MOV SECOND_SORTED_LO_TO_HI[DI+1],BH

CNT2:   INC DI
        LOOP LOOP2
        POP CX
        LOOP LOOP3

;ITEM 1

        XOR DI,DI
        XOR SI,SI
        MOV CX,N
LOOP5:  MOV AL,SECOND_SORTED_LO_TO_HI[DI]   ;load into AX the two numbers to be added
        MOV AH,FIRST_SORTED_HI_TO_LO[DI]
        PUSH AX                             ;the procedure SUM, computes the addiction
                                            ;in base 3 of 2 numbers passed as arguments
                                            ;through the stack in a 16 bit register
                                            ;(8 the first number and 8 the second)
        CALL SUM
        POP AX                              ;get the result of the procedure
        MOV ADDED[SI],AX                    ;store the result
        ADD DI,1
        ADD SI,2                            ;here the index is incremented by two as the
                                            ;ADDED array is an array of words.
        LOOP LOOP5

;ITEM 2

        XOR DI,DI
        XOR SI,SI
        MOV CX,N

;Being in base 3,the multiplication by three is a simple shift. Since each digit in base
;3 is represented here with two bits, we have to shift by two bits.

LOOP6:  XOR AX,AX                           ;clean the registers
        XOR BX,BX
        MOV AL,FIRST_SORTED_HI_TO_LO[DI]    ;load into the register
        MOV BL,SECOND_SORTED_LO_TO_HI[DI]
        SHL AX,1                            ;shift both the registers by two positions
        SHL AX,1
        SHL BX,1
        SHL BX,1
        MOV BY_THREE_MULTIPLIED_FIRST[SI],AX    ;store the result in memory
        MOV BY_THREE_MULTIPLIED_SECOND[SI],BX
        INC DI                              ;update indexes
        ADD SI,2
        LOOP LOOP6

;ITEM 3

        XOR DI,DI
        XOR SI,SI
        MOV CX,N
```

```
;The multiplication by two is performed with the addition between a number and itself

LOOP7:  MOV AL,FIRST_SORTED_HI_TO_LO[DI]
        MOV AH,AL                       ;we want to pass to the procedure SUM two
                                        ;identical numbers. They have to be in the
                                        ;higher and lower 8 bits of the register that
                                        ;is passed to the procedure through the stack
        PUSH AX
        CALL SUM                        ;compute the sum
        POP AX
        MOV BY_TWO_MULTIPLIED_FIRST[SI],AX  ;store the result in memory

        MOV AL,SECOND_SORTED_LO_TO_HI[DI]   ;do the same for the second array
        MOV AH,AL
        PUSH AX
        CALL SUM
        POP AX
        MOV BY_TWO_MULTIPLIED_SECOND[SI],AX

        INC DI                          ;update indexes
        ADD SI,2
        LOOP LOOP7

;ITEM 4

;The algorithm is the following. DX is an accumulator of the final result, AX
;stores the result of the internal multiplications. The multiplication is
;performed as shown in figure (below), where AX is computed according to the value
;of DH (DX containing the final result has been previously pushed into the stack):
;DH=0 -> AX=0, DH=1 -> AX=AL, DH=2 -> AX=2*AL. Every time AX is calculated
;the value of the final result DX is updated (considering the needed shifts).

        XOR DI,DI                       ;These are the indexes
        XOR SI,SI
LOOP9:  XOR CX,CX
        XOR DX,DX                       ;DX stores the result
        MOV CH,00000011b                ;This is the mask
LOOP8:  PUSH DX
        MOV DH,SECOND_SORTED_LO_TO_HI[DI]
        AND DH,CH                       ;Apply the mask to extract the
                                        ;right digit.
        SHR DH,CL                       ;Shift to have that digit in the
                                        ;lowest bits of the register.
        CMP DH,0                        ;If it is 0...
        JNE CNT5
        MOV AX,0                        ;...then AX=0
        JMP END1
CNT5:   CMP DH,1                        ;otherwise, if it is 1...
        JNE CNT6
        MOV AL,FIRST_SORTED_HI_TO_LO[DI]    ;...then AX=AL
        XOR AH,AH
        JMP END1
CNT6:   MOV AL,FIRST_SORTED_HI_TO_LO[DI]    ;otherwise it (DH) is 2, then
                                        ;AX=2*AL
        MOV AH,AL                       ;Put into AL and AH what we want
                                        ;to add. Here those two values are the same
        PUSH AX                         ;Push into the stack the parameters
```

```
        CALL SUM                        ;procedure
        POP AX                          ;get result

;Now AX has been calculated. Let's update DX.

END1:   POP DX                          ;restore the result in DX
        SHL AX,CL                       ;Shift AX (see the figure)

;Here AX is ready for the sum with DL. In order to reuse the procedure SUM, the addition
;is performed 8 bits at a time. The carry of the first addition is saved into BH
;and used in the second 8 bit addition. So, do first DL=AL+DL, then DH=AH+DH, finally
;DH=DH+BH.

;DL=AL+DL

        MOV BL,AL                       ;use BX for the arguments
        MOV BH,DL
        PUSH BX
        CALL SUM
        POP BX
        MOV DL,BL                       ;the result is in BL, so update DL
        PUSH BX                         ;save the carry

;DH=AH+DH

        MOV BL,AH
        MOV BH,DH
        PUSH BX
        CALL SUM
        POP BX
        MOV DH,BL

;DH=DH+BH

        POP BX                          ;restore the carry. It is into BH
        MOV BL,DH
        PUSH BX
        CALL SUM
        POP BX
        MOV DH,BL

;DX is updated.

        SHL CH,1                        ;shift the mask
        SHL CH,1
        ADD CL,2                        ;update the shift counter
        CMP CL,8                        ;if it is less than 8, jump to the
                                        ;multiplication with the next digit
        JNE LOOP8

        MOV MULTIPLIED[SI],DX           ;otherwise store the result in memory

        INC DI                          ;update registers
        ADD SI,2
        CMP DI,N
        JNE LOOP9
        .EXIT
```

```
SUM PROC FAR
        MOV BP,SP                       ;This is to manage arguments passed through
                                        ;the stack. See the theory.
        PUSH AX                         ;Save the registers
        PUSH BX
        PUSH CX
        PUSH DX

        MOV CH,00000011b                ;This will be used as a mask
        XOR CL,CL                       ;This is the "shift counter".
        XOR DX,DX                       ;DL will be the accumulator of the final result,
                                        ;DH is used to store the carry.


LOOP4:  MOV AH,[BP+4]                   ;As the procedure is FAR, we have to "climb" the
                                        ;stack by 4 bytes to reach the two arguments
        MOV BH,[BP+5]

        AND AH,CH                       ;Apply the mask to both the number to select only
                                        ;one base-three digit.
        AND BH,CH
        SHR AH,CL                       ;Shift right by "CL" positions. Initially CL is
                                        ;zero, then 2, 4 ecc... The shift is done to
                                        ;have the 2 base-three digits ready for the
                                        ;addition
        SHR BH,CL
        ADD AH,BH                       ;Addition
        ADD AH,DH                       ;Add eventual carry obtained in the previous
        addiction
        CMP AH,3                        ;If the result is higher than three...
        JL TRUE1
        SUB AH,3                        ;...subtract three...
        MOV DH,1                        ;...and set the carry to 1
        JMP CONT4
TRUE1:  MOV DH,0                        ;...otherwise set the carry to 0
CONT4:  SHL AH,CL                       ;shift the result back to the correct position
        ADD DL,AH                       ;add it to the final result

        SHL CH,1                        ;Update the mask in order to make it select the
                                        ;next base-three digit
        SHL CH,1
        ADD CL,2                        ;Update the shift counter
        CMP CL,8                        ;If all the 4 digits have been considered, finish
        JNE LOOP4

        ;The result is in DX (DL + the carry saved in DH)

        MOV [BP+4],DX                   ;put the result into the stack

        POP DX                          ;restore the registers
        POP CX
        POP BX
        POP AX

        RET
SUM     ENDP
        END
```


MEMORY
X
DH
=
AX
+
AX
+
AX
+
AX
=
DX