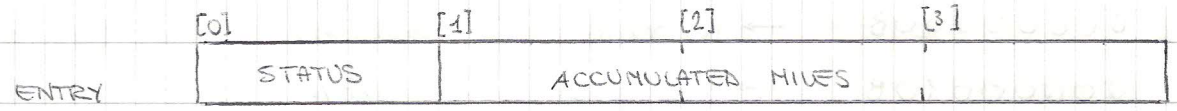
2015-06-15 (1)

2 DATABASES:

- MEMBERS' INFORMATION : FFM DB 4*4 DUP (?) ← TO BE UPDATED

  (initialized with the value in the example)



- FLIGHTS FLOWN BY THE MEMBERS (max 20 lines) ← INPUT

  FOM DB 30*5 DUP(?)

  (filled in at execution time by the user)

SUPPORTING VARIABLES (STATIC):

- MULTIPLIER DB 8 DUP(?)

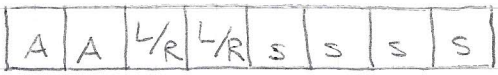
  Each entry contains the multiplier of the corresponding class of flight.

  Classes of flight are A-H encoded into 0-7 (used as index for this array)

  Multipliers are:

1	→ no shifts →	00000000B	(classes D and C, indexes 2 and 3)
2	→ 1 shift SHL →	00000001B	(classes H and G, indexes 6 and 7)
0.5	→ 1 shift SHR →	00110001B	(class B, index 1)
0.25	→ 2 shifts SHR →	00110010B	(class A, index 0)
1.25	→ itself + 2 shifts SHR →	11110010B	(class E, index 4)
1.5	→ itself + 1 shift SHR →	11110001B	(class F, index 5)

MULTIPLIER entry



A → sum the value itself : 0 ⇒ NO , 1 ⇒ YES

L/R → 0 ⇒ SHL (MUL) , 1 ⇒ SHR (DIV)

S → how many shifts must be performed

- BONUS DB 4 DUP (?) (one entry for each status, the array is indexed by the status itself)

  [0] 00000000B → No BONUS
  [1] 00000010B → 2 shifts (SHR) ⟹ 25%
  [2] 00000010B → 2 shifts (SHR) ⟹ 25%
  [3] 00000001B → 1 shift (SHR) ⟹ 50%

```
JT DW FILLFOM, DISPLAY, RESET   ; jump table

userMenu:
    ; print user menu
        [1] Fill the array FOM
        [2] Compute and display new parameters
        [3] Reset all databases
        [0] Exit

    ; read the value chosen by the user and jump to the corresponding job
    MOV AH,1
    INT 21H
    SUB AL,'0'
    CMP AL,0
    JE exit
    MOV BL,AL
    XOR BH,BH
    SUB BX,1
    ADD BX,BX
    JMP WORD PTR JT[BX]
```

```
FILLFOM:
    ; Read member number  (readDecimal PROCEDURE)      --> FOM[si]
    ; Read flight code    (readDecimal PROCEDURE)      --> FOM[si+1]
    ; Read class:         (suppose a correct input [A-H])  --> FOM[si+2]
        MOV AH, 1
        INT 21H
        SUB AL, 'A'              ; in this way, the code of the class can be used as index for
        MOV FOM[si+2], AL        ;      the MULTIPLIER array
    ; Read flown miles  (readDecimal PROCEDURE)      --> FOM[si+3], FOM[si+4]
    ; Ask if the user wants to insert a new entry or not:
        NO => jmp userMenu, YES: repeat fillFOM loop

DISPLAY:
    XOR SI, SI
computeLoop:
    XOR AH, AH
    MOV AL, FOM[SI]          ; if member number = 0 => empty entry => END of FOM database
    cmp AL, 0
    JE statusCheckAndPrint
    MOV AL, FOM[si+2]        ; class to be used as index in MULTIPLIER array
    XOR DI, DI
    XOR AH, AH
    ADD DI, AX
    MOV CL, MULTIPLIER[DI]
    AND CL, 00001111B        ; how many shifts
    CMP CL, 0
    JE checkAdd
    MOV DH, FOM[SI+3]        ; flown miles
    MOV DL, FOM[SI+4]
    MOV AL, MULTIPLIER[DI]
```

```
    TEST AL, 00110000B
    JE   multiply
    SHR  DX, CL          ; otherwise divide
    JMP  checkAdd
multiply:
    SHL, DX, CL
checkAdd:
    TEST AL, 11000000B
    JE   computeBonus
    MOV  CH, FOM[si+3]
    MOV  CL, FOM[si+4]
    ADD  DX, CX          ; <DX> = M*X

computeBonus:
    ...
```

```
computeBonus:
    MOV AL, FQM [SI]      ; member number (used as index for FFM to retrieve the status)
    SUB AL, 200
    XOR DI, DI
    XOR AH, AH
    PUSH DX               ; that maintains  M * X  value
    MOV BX, 4
    MUL BX                ; because each entry of FFM is composed of 4 bytes
    ADD DI, AX
    POP DX
    MOV AL, FFM [DI]
    XOR AH, AH
    CMP AX, 0             ; NO BONUS for status 0
    JE totalMiles
    XOR DI, DI
    ADD DI, AX
    MOV CL, BONUS [DI]    ; bonus B
    MOV AH, FQM [SI+3]    ; flown miles M
    MOV AL, FQM [SI+4]
    SHR AX, CL            ; <AX> = M * B

totalMiles:
    ADD AX, DX
    MOV CX, AX            ; <CX> = M * X + M * B
    MOV AL, FQM [SI]      ; member number to be used as index in FFM array
    ; DI ← AL x 4         (because each entry of FFM is composed of 4 bytes)
    ADD  FFM [di + 3], CL
    ADC  FFM [di + 2], CH
    ADC  FFM [di + 1], 0  ; miles updated
    ADD SI, 5             ; next FQM entry
    jmp computeloop
```

```
; after updating miles, compute new status and print variations
statusCheckAndPrint:
    XOR SI, SI      ; index for FFM
    XOR DI, DI      ; member number (e.g. 0 --> member 200)
checkAndPrint:
    MOV DL, FFM[SI+1]       ; miles
    MOV AH, FFM[SI+2]
    MOV AL, FFM[SI+3]
    CMP DL, 0               ; find new status
    JE status3              ; if the highest part is ≠0 => miles > 40000
    CMP AX, 40000
    JAE status3
    CMP AX, 10000
    JAE status2
    CMP AX, 3000
    JAE status1
    MOV CH, 0       ; otherwise status=0
    jmp compareStatus
status3:
    MOV CH, 3
    jmp compareStatus
status2:
    MOV CH, 2
    jmp compareStatus
status1:
    MOV CH, 1
```

```
compareStatus:
    MOV CL, FFM[SI]          ; old status
    CMP CH, CL
    JE noChange
    MOV FFM[SI], CH          ; otherwise, update status
    ; print new STATUS
    JMP printMiles
noChange:
    ; print "no change" message

printMiles:
    ; print total miles (old + new) for each member
    ; miles are store in  FFM[SI+1], FFM[SI+2], FFM[SI+3]
    JMP userMenu

RESET:
    ; reset the value of FFM and FOM databases:
    ; FOM empty (filled in by the user)
    ; FFM ← initialized again with the values in the example
    JMP userMenu

exit:
    . exit
```

PROCEDURE TO READ A DECIMAL NUMBER UP TO 65.536 (16 bits)

```
readDecimal proc
    PUSH BP
    MOV BP, SP
    ; push AX, CX and DX
    MOV CX, [BP+4]      ; max number of digits to be read
    MOV DX, 0           ; will store the decimal number
readloop:
    MOV AH, 1
    INT 21H
    CMP AL, 13          ;new line
    JE endReadloop
    SUB AL, '0'
    MOV CH, AL          ; maintain the value that has been read
    MOV AX, DX
    MOV DX, 10
    MUL DX              ; multiply the old value by 10
    MOV DX, AX
    ADD DL, CH          ; add the read value
    ADC DH, 0
    XOR CH, CH
    LOOP readloop
endReadloop:
    MOV [BP+4], DX
    ; pop all pushed registers
    ret
readDecimal endp
```