2015-02-23

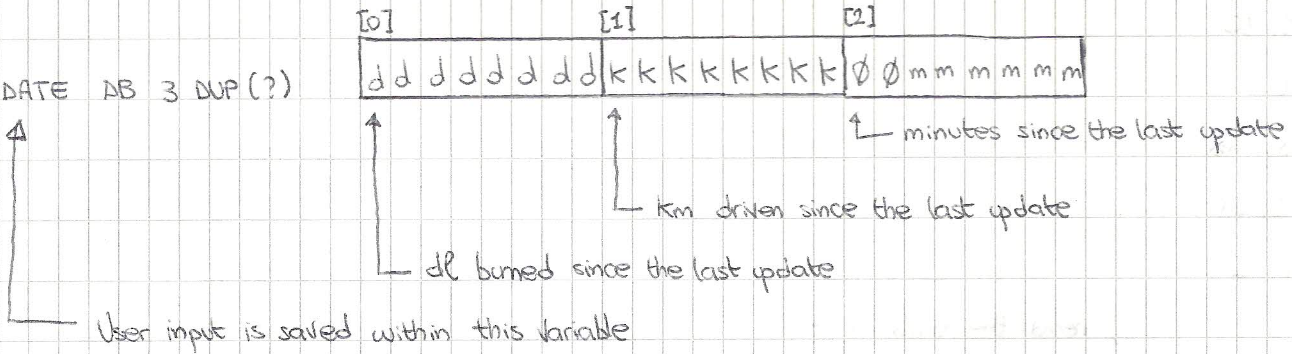
Parameters:

- Number of km driven [0,800] → 16 bits (needed = 10)
- Duration of the drive [0,240] → 8 bits
- Number of km/l [0,30] → 8 bits (needed = 5)
- Number of dl/100km [0,250] → 8 bits
- Average speed [km/hour] [0,200] → 8 bits
- Number of dl in the tank [0,320] → 16 bits (needed = 9)
- Number of drivable km with the available fuel in the tank [max 800] → 16 bits (needed = 10)

Tank fully refilled = 32 l = 320 dl (max each 4 hours)

Initialization: tank full, all parameters equal to Ø

UPDATE DB 3 DUP (?)


[0]
[1]
[2]
d d d d d d d d
k k k k k k k k
Ø Ø m m m m m m
minutes since the last update
km driven since the last update
dl burned since the last update
User input is saved within this variable

When the user select "Enter an update" from the menu, the 3 values are read and put in UPDATE variable.

Needed variables:

```
KM_DRIVEN         DW  Ø
DRIVE_DURATION    DB  Ø     ; overall duration in minutes
DL_USED           DW  Ø
L_TANK            DB  32    ; liters still in the tank (at the beginning = 32)
AVERAGE_SPEED     DB  Ø
DURATION_HOUR     DB  Ø
DURATION_MIN      DB  Ø
KM_PER_L          DB  Ø
DL_PER_100KM      DB  Ø
DRIVABLE_KM       DW  Ø
```

ESEMPIO PER LEGGERE NUMERI A PIÙ DI UNA CIFRA

```
; Example of update (all the 3 Values - dl, km, m - are read in the same way)
    mov cx, 0          ; read the first digit
    mov ah, 1          ↵
    int 21h
    sub al,'Ø'
    add cl, al
    int 21h            ; read the second digit
    cmp al, 13         ; compare with new line ascii code...
    je insertkm        ; ... to stop the reading
    sub al, 'Ø'
    mov dl, al
    mov al, 10
    mul cl
    mov cl, al
    add cl, dl
    mov ah , 1
    int 21h            ; read the third digit
    cmp al, 13
    je insertkm
    sub al, 'Ø'
    mov dl, al
    mov al, 10
    mul cl
    mov cl, al
    add cl, dl

insertkm:
    mov UPDATE[0], cl          ; save deciliters
    ...                        ; make the same thing to read kilometers
```

Each time a new update is inserted, all the variables representing the parameters are updated (and also the supporting variables), so, when the user select "display", the program simply prints all those variables.
(Variables needed are reported for each item)

```
; ITEM 1
    KM_DRIVEN        DW  Ø
    DRIVE_DURATION   DB  Ø
    DL_USED          DW  Ø
    L_TANK           DB  32
```

← Those variables are updated each time a new input is given and displayed by request.

```
  ; Assume of having the input in UPDATE variable
  ; - Compute the overall number of km driven:        UPDATE [1] = kkkkkkkk
    xor ah, ah
    mov al, UPDATE [1]
    add KM_DRIVEN, ax
  ; - Compute overall duration (min):                  UPDATE [2] = ØØmmmmmm
    mov al, UPDATE [2]
    add DRIVE_DURATION, al
  ; - Compute the overall deciliters of fuel:          UPDATE [Ø] = dddddddd
    mov al, UPDATE [Ø]
    xor ah, ah
    add DL_USED, ax
  ; - Compute the number of liters still in the tank
  ; To have the maximun precision, liters still in the tank are computed each time starting from the maximum
    capacity of the tank (320) and the used deciliters of fuel
    MAX = 320 dl
```

$$L\_TANK = \frac{MAX - DL\_USED}{10} \Rightarrow \frac{[16\ bits] - [16\ bits]}{[8\ bits]} = \frac{[16\ bits]}{[8\ bits]} \Rightarrow \underset{\uparrow\ 8\ bits}{AL \leftarrow RESULT}$$

```
    mov ax, 320
    sub ax, DL_USED
    mov cl, 10
    div cl
    mov L_TANK, al
```

; ITEM 2

; AVERAGE_SPEED DB Ø

You have to use KM_DRIVEN and DRIVE_DURATION Variables.

Pay attention: drive duration is stored in minutes, so you have to convert minutes in hours

hours = minutes / 60

$$\text{AVERAGE\_SPEED} = \frac{\text{KM\_DRIVEN}}{\text{DRIVE\_DURATION}/60} = \frac{\overset{\text{16 bits}}{\text{KM\_DRIVEN}} * \overset{\text{AX}}{60}}{\underset{\text{8 bits (to be transformed in 16)}}{\text{DRIVE\_DURATION}}} \longrightarrow \frac{[\text{32 bits}]}{[\text{16 bits}]} \longrightarrow \text{AX} = \text{result}$$

For MAXIMUM PRECISION: FIRST MULTIPLICATE, THEN DIVIDE

Average speed can be represented on 8 bits, so only the content of AL is useful.

```
mov ax, 60
mov cx, KM_DRIVEN
mul cx
mov cl, DRIVE_DURATION
xor ch, ch
div cx
mov AVERAGE_SPEED, al
```

; ITEM 3

; DURATION_HOUR DB 0

DURATION_MIN DB 0

DRIVE_DURATION / 60 ⟶ [16bits] / [8 bits] ⟹ AL = result (HOURS)

↳ 8 bits (to be tranformed in 16 bits) AH = remainder (MINUTES)

```
xor ab,ah
mov al, DRIVE_DURATION
mov cl, 60
div cl
mov DURATION_HOUR, al
mov DURATION_MIN, ah
```

; ITEM 4

; To compute km/liter you need: KM_DRIVEN (DW) and DL_USED (DW)

The new variable is KM_PER_L DB Ø (at the beginning)

$$\text{KM\_PER\_L} = \frac{\text{KM\_DRIVEN}}{\text{DL\_USED} / 10} = \frac{\overset{16\text{bits}}{\text{KM\_DRIVEN}} * \overset{16\text{bits}}{10}}{\underset{16\text{ bits}}{\text{DL\_USED}}} \longrightarrow \frac{[32\text{ bits}]}{[16\text{ bits}]} \longrightarrow AX = \text{result}$$

(only AL is useful: 8 bits)

To obtain maximum precision: first multiplicate, then divide

```
mov ax, KM_DRIVEN
mov cx, 100
mul cx
mov cx, DL_USED
div cx
mov KM_PER_L, al
```

; ITEM 5

; DL_PER_100KM DB Ø

Variables needed: DL_USED (WORD) and KM_DRIVEN

$$\mathrm{DL\_PER\_100KM} = \frac{\mathrm{DL\_USED}}{\mathrm{KM\_DRIVEN}} * 100 = \frac{\overset{\text{16 bits}}{\overset{\downarrow}{\mathrm{DL\_USED}}} * \overset{\text{16 bits}}{\overset{\downarrow}{100}}}{\underset{\text{16 bits}}{\underset{\uparrow}{\mathrm{KM\_DRIVEN}}}} \longrightarrow \frac{[\text{32 bits}]}{[\text{16 bits}]} \longrightarrow AX = \text{result}$$

(only AL contains useful information)

```
mov ax, DL_USED
mov cx, 100
mul cx
mov cx, KM_DRIVEN
div cx
mov DL_PER_100KM, al
```

; ITEM 6

$$\frac{\mathrm{KM\_DRIVEN}}{\mathrm{DL\_USED}} = \frac{\mathrm{DRIVABLE\_KM}}{\mathrm{DL\_AVAILABLE}} \longleftarrow \text{To obtain maximum precision}$$

$$\frac{\mathrm{KM\_DRIVEN} * (320 - \mathrm{DL\_USED})}{\mathrm{DL\_USED}} \longrightarrow \frac{[\text{32 bits}]}{[\text{16 bits}]} \longrightarrow AX = \text{result}$$

```
mov ax, 320
sub ax, DL_USED
mov cx, KM_DRIVEN
mul cx
mov cx, DL_USED
div cx
mov DRIVABLE_KM, ax
```