Architetture dei Sistemi di Elaborazione	Delivery date: **By 2:00 AM on 20th November 2024**
Laboratory 7	Expected delivery of lab_07.zip must include: - zipped project folder of the exercises 1 and 2 - this document compiled possibly in pdf format.

Exercise 1)

A videogame speedrunner is tracking their daily attempts at speedrunning a game, recording both their best times and their total attempts per day. Write a program in **ARM assembly** language that analyzes their **speedrunning performance data**.

```
Days                DCB 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07

Best_times          DCD 0x06, 1300, 0x03, 1700, 0x02, 1200, 0x04, 1900,
                    DCD 0x05, 1110, 0x01, 1670, 0x07, 1000

Failed_runs         DCD 0x02, 50, 0x05, 30, 0x06, 100, 0x01, 58,
                    DCD 0x03, 40, 0x04, 90, 0x07, 25

Num_days           DCB 7
```

`Days` is a table where each entry consists of a day of the week (e.g., 0x01 is Monday, 0x02 Tuesday, ..)
`Best_times` is a table where each entry consists of two integer values: the ID of the day (4 bytes) and the best time (in seconds) achieved that day by the speedrunner (4 bytes).
`Failed_runs` is a table where each entry consists of two integer values: the ID of the day (4 bytes) and the number of times the player had to reset the game (4 bytes). Notice that not all days he plays videogames.
`Num_days` is a 1-byte constant and indicates the number of days in a week.

Compute the **total number of days** the speedrunner best time was better or equal to 1300 and store it in register R11. Then for each day this time was better or equal to 1300 sum the number of Failed_runs and store it in register R10.

Note: The constant data section must be defined in the code section, with a 2byte alignment and 4096 boundary zero bytes.
Example:

```
…
// ALIGNMENT
// BOUNDARY (SPACE ….)
MY DATA
// BOUNDARY (SPACE ….)
..
```

Exercise 2)

Save in two separate vectors `Best_times_ordered` and `Failed_runs_ordered`, the ID of the days in descending order by best times and failed runs, respectively.
For example at the end the vectors would be ordered as follows:

```
Best_times_ordered            DCD   0x04,0x03,0x01,0x06,0x02,0x05, 0x07
Failed_runs_ordered           DCD   0x06,0x04,0x01, 0x02, 0x03, 0x05, 0x07
```

Then, save in R11 the ID of the worst "best_time" day.

Compute the needed bytes for the above vectors.

Vector	Size [bytes]
`Best_times_ordered`	28
`Failed_runs_ordered`	28

Report the following program characteristics (Hint: See the build output window in Keil).

	Size [bytes]
Program Size	8584
Read Only data	764
Read Write data	168
Zero Initialized data	512

And provide a brief explanation about which directives can influence the previous program characteristics.

La direttiva SPACE influisce sulla dimensione del programma poiché è stata inserita all'interno della sezione CODE usando una literal pool (LTORG). Se fosse stata posizionata all'interno dell'area DATA, avrebbe influenzato la caratteristica "Read Write data" del programma, poiché quell'area è etichettata come READWRITE.

La direttiva READONLY utilizzata nella sezione CODE impedisce al programma di modificare i vettori e gli spazi vuoti definiti all'interno di questa area, rendendo necessaria la creazione di un'area READWRITE separata. In questo modo è possibile copiare e ordinare i vettori (che, essendo definiti con la direttiva DCD, non sarebbero comunque modificabili anche se l'area fosse scrivibile) in un'area che consente la scrittura.

La direttiva ALIGN ha un impatto minimo sulla dimensione del programma, poiché allinea i dati a byte pari (ALIGN 2). Ad esempio, quando abbiamo una lista di byte (come la tabella Days, definita con 7 valori DCB), si aggiunge un byte vuoto per garantire che i dati successivi siano allineati correttamente al prossimo indirizzo.

La direttiva SPACE riserva una quantità specifica di spazio di memoria senza assegnare valori iniziali. Nel mio codice, SPACE è utilizzato per riservare spazio per i vettori di output ordinati, come Best_times_ordered e Failed_runs_ordered, e per le aree di lavoro (rw) che servono a manipolare i dati senza alterare quelli originali. Posizionata in CODE, SPACE influenza la dimensione del programma; se fosse in DATA, influenzerebbe la sezione READWRITE.