UNIVERSITÀ di VERONA | Dipartimento di INFORMATICA

Laboratorio di Architettura degli Elaboratori

Elaborato Assembly

A.A 2021/2022

Studenti:

Vittorio Maria Stano (VR457793)

Stefano Esposito (VR461049)

22/06/2022

INDICE GENERALE

DESCRIZIONE GENERALE DEL PROGETTO

Si intende realizzare un programma **Assembly** che simuli il sistema di telemetria del videogame **F1**.
Il sistema fornisce in input i dati grezzi di giri motore (rpm), temperatura motore e velocità di tutti i piloti presenti in gara per ogni istante di tempo.

Ogni campo è diviso da una virgola usata come separatore.

Ogni riga del file di input è così composta:
<tempo>,<id_pilota>,<velocità>,<rpm>,<temperatura>

Id pilota rappresenta un valore numerico che identifica univocamente un pilota. L'associazione tra id e nome del pilota è il seguente:

ID	Nome
0	Pierre Gasly
1	Charles Leclerc
2	Max Verstappen
3	Lando Norris
4	Sebastian Vettel
5	Daniel Ricciardo
6	Lance Stroll
7	Carlos Sainz
8	Antonio Giovinazzi
9	Kevin Magnussen
10	Alexander Albon
11	Nicholas Latifi
12	Lewis Hamilton
13	Romain Grosjean
14	George Russell
15	Sergio Perez
16	Daniil Kvyat
17	Kimi Raikkonen
18	Esteban Ocon
19	Valtteri Bottas

Nella prima riga del file di input è presente il nome di un pilota che si vuole monitorare.

Il programma restituisce i dati relativi al solo pilota indicato nella prima riga del file, in base alle soglie indicate in seguito (**LOW**, **MEDIUM**, **HIGH**).

Il file output riporta queste soglie per tutti gli istanti di tempo in cui il pilota è monitorato.

Le righe del file di output sono strutturate nel seguente modo:
<tempo>,<livello rpm>,<livello temperatura>,<livello velocità>

Alla fine del file output è richiesto l'inserimento di una riga aggiuntiva che contiene i seguenti dati nell'ordine: numero giri massimi rilevati, temperatura massima rilevata, velocità di picco e velocità media.

La struttura dell'ultima riga è la seguente:
<rpm max>,<temp max>,<velocità max>,<velocità media>

Alla fine del file è richiesta l'aggiunta di un a capo. Le soglie per i dati monitorati sono così definite:

- **Giri Motore**
  - LOW: rpm <= 5000
  - MEDIUM: 5000 < rpm <=10000
  - HIGH: rpm > 10000
- **Temperatura**
  - LOW: temp <= 90
  - MEDIUM: 90 < temp <= 110
  - HIGH: temp > 110
- **Velocità**
  - LOW: speed <= 100
  - MEDIUM: 100< speed <=250
  - HIGH: speed > 250

La velocità media viene calcolata come quoziente intero della divisione. Se il nome del pilota inserito **non è valido**, il programma restituisce la stringa **Invalid**, seguita da un a capo.

Il programma **legge** il contenuto del file **input.txt** e restituisce il **risultato** in **output.txt**. Il file input.txt contiene il nome pilota nella prima riga e tutti i campioni di dati nelle righe successive.

Il file **output.txt** contiene solo le righe del pilota selezionato e infine la riga con i valori massimi e la media della velocità.

Il file **main.c** contiene il sorgente che chiama la funzione assembly **telemetry.s**.

VARIABILI UTILIZZATE

Di seguito riportiamo le variabili utilizzate nei file **.s** che compongono il programma realizzato:

Variabili file telemetry.s:

- **ebp_value** (di tipo long): viene salvato il valore del registro **%ebp** all'ingresso di funzione in modo da poter essere ripristinato al valore iniziale all'uscita della funzione. Questo è molto importante perché durante l'esecuzione del programma, e quindi durante le varie operazioni, i registri vengono modificati ed è necessario ripristinarli una volta terminata l'esecuzione.

Variabili file controllo_piloti.s:

- **pilot_n_str** (di tipo string): dove n è l'ID pilota (per ogni n compreso tra 0 e 19). Viene salvata la stringa relativa al nome del pilota per il successivo confronto con il file outptut.txt.
- **pilot_n_str_len** (di tipo long): dove n è l'ID pilota (per ogni n compreso tra 0 e 19). Viene salvata la lunghezza della stringa relativa al nome del pilota per il successivo confronto con il file outptut.txt.

Variabili file verifica_riga.s:

- **tempo** (di tipo ascii): viene salvato il **valore tempo** (cifra per cifra) letto dal file input.txt. Al termine della lettura (carattere virgola) viene inserito il terminatore di stringa \0.
- **address_output** (di tipo long): viene salvato, in ogni momento, **l'indirizzo** di memoria corrispondente al file output.txt per la scrittura nello stesso.
- **offset_output** (di tipo long): viene salvato, in ogni momento, **l'offset** (posizione) al quale eseguire la scrittura nel file output.txt.
- **velocita** (di tipo long): viene salvato il valore della **velocità** letta dal file input.txt per il confronto del valore massimo.
- **velocita_max** (di tipo long): viene salvato volta per volta il valore della **velocità massima** da riportare nell'ultima riga.
- **velocita_media** (di tipo long): viene salvato il valore della **velocità media** (quoziente intero della divisione) da riportare nell'ultima riga.
- **rpm_max** (di tipo long): viene salvato volta per volta il valore dei **giri motore massimi** da riportare nell'ultima riga.
- **temp_max** (di tipo long): viene salvato volta per volta il valore della **temperatura massima** da riportare nell'ultima riga.
- **counter_velocita** (di tipo long): viene incrementato volta per volta il **contatore** per il calcolo della velocità media da riportare nell'ultima riga.
- **sum_velocita** (di tipo long): viene salvata volta per volta la **somma** delle **velocità** per il calcolo della velocità media da riportare nell'ultima riga.

FUNZIONI IMPLEMENTATE

Di seguito riportiamo le funzioni implementate che compongono il programma realizzato:

Funzione telemetry.s:

La funzione si occupa della **gestione** completa **del flusso del programma**. Viene chiamata la funzione "**controllo_piloti**" che si occupa della verifica del pilota nel file input.txt. Il funzionamento di questa funzione verrà dettagliatamente spiegato successivamente.

Dopo aver eseguito la funzione, nel registro **%dl**, si trova **l'ID pilota**. telemetry.s esegue un confronto tra **%dl** e il valore **20** (assegnato come pilota non trovato). Se il confronto da esito **positivo**, il programma stampa in output.txt la stringa "**Invalid**" e termina l'esecuzione.

Nel caso in cui ID pilota fosse corretto, il controllo dei valori inseriti in input.txt avviene attraverso la funzione "**verifica_riga**". Dopo l'esecuzione di quest'ultima, si ripristinano i registri di funzione e si termina l'esecuzione del programma. Nel file output.txt viene salvata l'intera analisi effettuata.

Funzione controllo_piloti.s:

La funzione ha il compito di **verificare** la correttezza del **nome del pilota** indicato nel file input.txt confrontandolo con le stringhe pilota dichiarate nel file **controllo_piloti**. Dopo aver verificato la corrispondenza, viene effettuata l'associazione tra ID pilota e nome.
La funzione viene chiamata dal file telemetry.s. Il controllo del pilota avviene in due fasi.

Il programma esegue il controllo in modo sequenziale da pilot_0 a pilot_19. Al registro **%dl** viene assegnato, volta per volta, l'ID corrispondente al controllo che il programma sta effettuando in quel momento (da 0 a 19).

Durante la **prima fase**, si confronta la posizione dell'ultimo carattere della stringa pilota nel file **controllo_piloti** (decrementata due volte per la presenza del carattere di terminazione \0) con la stessa posizione, nel file input.txt.

Il controllo fornisce esito **positivo** quando viene trovata una corrispondenza con il **carattere a capo** (ASCII 10) in input.txt. In questo caso il nome pilota in input.txt ha la stessa lunghezza del nome pilota trovato in **controllo_piloti**. Tuttavia, questo confronto non ci assicura la correttezza di ogni singolo carattere del nome pilota, perciò si procede alla seconda fase del controllo.

Se invece non viene trovata alcuna corrispondenza, il nome pilota inserito nel file input.txt è errato. Al registro **%dl** viene assegnato il valore **20** (**pilota non trovato**) da restituire alla funzione chiamante telemetry.s.

La **seconda fase** del controllo consiste nel confronto, carattere per carattere, del pilota inserito nel file input.txt e della stringa pilota individuata dal programma come una possibile corrispondenza. Se il controllo fornisce esito positivo allora si termina l'esecuzione uscendo dalla funzione (**ID pilota** corrispondente si trova nel registro **%dl**). Se il controllo fornisce esito negativo si continua con la prima fase per la ricerca di un altro candidato.

Funzione verifica_riga.s:

La funzione si occupa del **controllo dei valori di telemetria** inseriti nel file input.txt e della stampa delle relative analisi. Dopo aver verificato la correttezza dell'ID pilota, questa funzione viene chiamata da telemetry.s.

Ogni riga del file di input, escludendo la prima del nome pilota, è così composta:
<tempo>,<id_pilota>,<velocità>,<rpm>,<temperatura>

Le righe del file di output sono strutturate nel seguente modo:
<tempo>,<livello rpm>,<livello temperatura>,<livello velocità>

La funzione restituisce i dati relativi al solo pilota indicato nella prima riga del file.

La funzione inizia la lettura del file input.txt a partire dalla **seconda riga** del file stesso. Il **primo parametro** è il **tempo**, il quale viene salvato nella variabile ASCII "tempo", per la successiva stampa in output.txt. Il tempo viene memorizzato fino a quando il programma non rileva il carattere virgola (44 in ASCII).

Si procede alla lettura del **secondo parametro**, **ID pilota**. Viene eseguita la conversione **ASCII→INT** fino a quando il programma non rileva il carattere di separazione virgola (44 in ASCII). Completata la conversione in intero, il valore viene confrontato con il registro **%dl** dove si trova l'ID pilota rilevato in precedenza dalla funzione **controllo_piloti**.
Se il confronto fornisce esito **positivo**, si procede alla stampa, nel file output.txt, del tempo memorizzato nella variabile ascii tempo, altrimenti, se il controllo fornisce esito **negativo**, il programma incrementa la posizione di lettura del file input.txt fino a quando non si arriva alla riga successiva (dopo il carattere a capo = 10 in ASCII) e viene ri-eseguito il primo controllo del tempo.

Si procede alla stampa di una virgola (carattere ASCII 44) come separatore nel file output.txt.

Dopo aver stampato il valore tempo, si procede alla verifica del **terzo parametro**, la **velocità**. Viene eseguita la conversione **ASCII→INT** fino a quando il programma non rileva il carattere di separazione virgola (44 in ASCII). Completata la conversione in intero, il valore viene salvato nella variabile "**velocita**" per la successiva analisi.
Ad ogni ciclo viene eseguito anche un controllo della velocità massima letta. In caso di velocità maggiore della precedente, questa viene salvato il valore nella variabile "**velocita_max**", per la

successiva stampa, nell'ultima riga del file output.txt. Viene inoltre incrementata, ad ogni lettura di velocità, una variabile "**counter_velocita**", indispensabile per il successivo calcolo della velocità media. Nella variabile **sum_velocita** si memorizza volta per volta la somma delle velocità lette fino a quel momento.

Si procede con il **quarto parametro**, il valore dei **giri del motore**. Viene eseguita la conversione **ASCII→INT** fino a quando il programma non rileva il carattere di separazione virgola (44 in ASCII). Completata la conversione in intero, il valore **RPM** viene salvato nel registro **%eax** per procedere all'analisi delle soglie **LOW, MEDIUM, HIGH**. Viene confrontato il valore con le soglie stabilite e viene stampato nel file output.txt il range corrispondente **LOW, MEDIUM, HIGH**.
Ad ogni ciclo viene eseguito anche un controllo dei giri motore massimi letti. In caso di valore maggiore del precedente, questo viene salvato nella variabile "**rpm_max**", per la successiva stampa, nell'ultima riga del file output.txt.

Si procede alla stampa di una virgola (carattere ASCII 44) come separatore nel file output.txt.

Dopo il controllo dei giri motore, si procede al **quinto parametro**, la **temperatura**. Viene eseguita la conversione **ASCII→INT** fino a quando il programma non rileva il carattere di separazione virgola (44 in ASCII). Completata la conversione in intero, il valore della temperatura viene salvato nel registro **%eax** per procedere all'analisi delle soglie **LOW, MEDIUM, HIGH**. Viene confrontato il valore con le soglie stabilite e viene stampato nel file output.txt il range corrispondente **LOW, MEDIUM, HIGH**.
Ad ogni ciclo viene eseguito anche un controllo della temperatura massima letta. In caso di valore maggiore del precedente, questo viene salvato nella variabile "**temp_max**", per la successiva stampa, nell'ultima riga del file output.txt.

Si procede alla stampa di una virgola (carattere ASCII 44) come separatore nel file output.txt.

Si procede ora con la stampa delle soglie **LOW, MEDIUM, HIGH** per la **velocità**. Il valore intero è memorizzato nella variabile "velocita". Viene spostato il valore nel registro **%eax**, confrontato con le soglie stabilite e successivamente viene stampato nel file output.txt il range corrispondente **LOW, MEDIUM, HIGH**.

Si procede alla stampa di un carattere a capo (carattere ASCII 10).

Arrivati a questo punto, è terminata la verifica dei valori presenti in una riga. Per verificare la presenza di altre righe da sottoporre a controllo, si confronta il **carattere successivo** del file input.txt con il **terminatore di stringa** (carattere ASCII 0). Se il confronto restituisce esito **negativo**, significa che è presente un'altra riga nel file input.txt. Si incrementa la posizione del puntatore al file e si effettua nuovamente il controllo di tutti i valori tempo, ID pilota, velocità, giri motore, temperatura. Se il controllo restituisce esito **positivo** allora non sono presenti altre righe da verificare e si è terminato il controllo.

A questo punto si procede alla stampa dell'ultima riga del file output.txt così formata:
<rpm max>,<temp max>,<velocità max>,<velocità media>

Viene calcolata la velocità media sum_velocita/counter_velocita e salvata nella variabile **velocita_media** per la successiva stampa.

Si procede alla stampa del valore dei giri motore massimi rilevati. Il valore è memorizzato nella variabile **rpm_max**. Successivamente viene spostato il valore nel registro **%eax** e si esegue la conversione **INT→ASCII** bit a bit tramite l'implementazione delle istruzioni "**itoa**". Viene stampato il valore **rpm_max**.

Si procede alla stampa di una virgola (carattere ASCII 44) come separatore nel file output.txt.

Si procede alla stampa del valore della temperatura massima rilevata. Il valore è memorizzato nella variabile **temp_max**. Successivamente viene spostato il valore nel registro **%eax** e si esegue la conversione **INT→ASCII** bit a bit tramite l'implementazione delle istruzioni "**itoa**". Viene stampato il valore **temp_max**.

Si procede alla stampa di una virgola (carattere ASCII 44) come separatore nel file output.txt.

Si procede alla stampa del valore della velocità massima rilevata. Il valore è memorizzato nella variabile **velocita_max**. Successivamente viene spostato il valore nel registro **%eax** e si esegue la conversione **INT→ASCII** bit a bit tramite l'implementazione delle istruzioni "**itoa**". Viene stampato il valore **velocita_max**.

Si procede alla stampa di una virgola (carattere ASCII 44) come separatore nel file output.txt.

Si procede alla stampa del valore della velocità media, calcolata in precedenza. Il valore è memorizzato nella variabile **velocita_media**. Successivamente viene spostato il valore nel registro **%eax** e si esegue la conversione **INT→ASCII** bit a bit tramite l'implementazione delle istruzioni "**itoa**". Viene stampato il valore **velocita_media**.

Si procede alla stampa di un carattere a capo (carattere ASCII 10), come richiesto dalle specifiche e si termina l'esecuzione della funzione ritornando a telemetry.s.

FILE MAIN.C

L'avvio del programma avviene tramite un file di codice C (**main.c**) non editabile fornitoci durante l'assegnazione del progetto. In particolare, questo codice permette di passare come parametro al file **telemetry.s** un file **.txt** contenente i dati di telemetria dei piloti F1 da analizzare. Il codice è il seguente:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <stdint.h>
#include <sys/time.h>
#include <sys/stat.h>

/* Nome funzione Assembly dichiarata con extern */
extern int telemetry(char* input,char* output);
/* ************************************ */

// funzione di supporto che prende in input il nome del file
size_t get_size(char* filename);

// e modifica la stringa input da passare come parametro alla funzione assembly
char* retrieve_input(char* filename,char* input,size_t size);

// Scrive il risultato su un file di output
void  write_output(char* filename,char* output);



int main(int argc, char *argv[]) {

// recupero la stringa del nome del file di input
char* inputFilename     = argv[1];
// recupero la stringa del nome del file di output
char* outputFilename    = argv[2];
// stringa che conterrà il contenuto del file di input
char* inputString;
// stringa che conterrà il contenuto del file di output
char* outputString;
// variabile che conterrà la dimensione del file di input
size_t size;

   //*********************************************************
   // Alloco lo spazio delle stringhe in base al file di input
   //*********************************************************
```

```
    // recupero la dimensione del file di input
    size=get_size(argv[1]);
    // alloco lo spazio per la stringa che conterrà il contenuto del file di input
    inputString=malloc(sizeof(char)*size);
    // alloco lo spazio per la stringa di output della funzione assembly
    outputString=malloc(sizeof(char)*size);

    //**********************************
    // Recupera input dal file
    //**********************************

    retrieve_input(inputFilename,inputString,size);

    //**********************************
    // Chiamata funzione assembly
    //**********************************
    telemetry(inputString,outputString);

    //printf("%s",outputString);

    //**********************************
    // Scrivi output della funzione sul file di output
    //**********************************

    write_output(outputFilename,outputString);

    // elimino dalla memoria lo spazio allocato dalle stringhe
    free(inputString);
    free(outputString);
    return 0;
}


size_t get_size(char* filename){

size_t size;
struct stat st;

    // funzione per recuperare info del file e salvarle in struttura st
    stat(filename, &st);
    // aggiungo un carattere per  aggiungere il  tappo \0 a fine stringa
    size = st.st_size+1;

return size;
}
```

```
void write_output(char* filename, char* output){

// apre il file da scrivere. Se non esiste lo crea. Se esiste lo resetta
FILE *outputFile = fopen (filename, "w");

   // scrive sul file
   fprintf (outputFile, "%s", output);
   // chiude il file
   fclose (outputFile);

}

char* retrieve_input(char* filename,char* input,size_t size){

   FILE *inputFile = fopen(filename, "r");
   int i=0;
   char c;

      if (inputFile == 0)
      {
         fprintf(stderr, "failed to open the input file. Syntax ./test <input_file> <output_file>\n");
         exit(1);
      }

   // copio dal file alla stringa, parametro della funzione assembly
   while ( EOF != (c = fgetc( inputFile )) && i < size ){
      input[ i ] = c;
      i++;
   }

   // tappo!
   input[strlen(input)]='\0';

   fclose(inputFile);
return input;

}
```

FLOWCHART DEL PROGRAMMA