UNIVERSITÀ di VERONA

Dipartimento di INFORMATICA

# Elaborato Assembly

## Laboratorio di Architettura degli Elaboratori

Simulazione sistema di telemetria di F1

A.A. 2021/2022

*Filippo Barbieri (VR472408)*
*Alessio Brighenti (VR471509)*

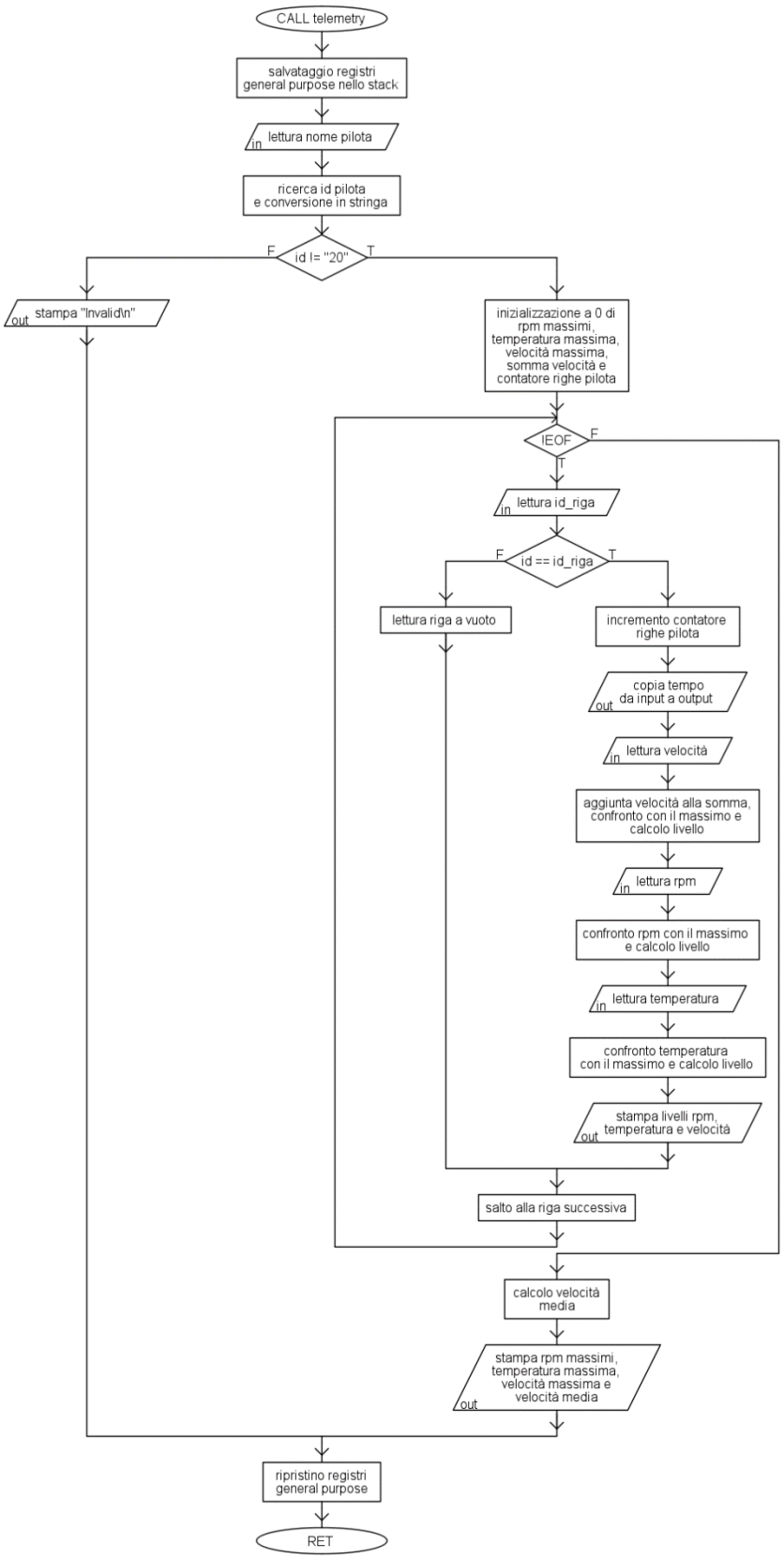
# Indice

# 1. Specifiche

Si descriva un programma 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>`
Ad eccezione della prima riga nella quale è presente il nome di un pilota che si vuole monitorare.
Ogni pilota è associato ad un id numerico da 0 a 19.

Si scriva un programma in assembly che restituisca i dati relativi al solo pilota indicato nella prima riga del file, in base a delle soglie indicate.
Vengono definite tre soglie per tutti i dati monitorati: LOW, MEDIUM, HIGH.
Il file di output dovrà riportare queste soglie per tutti gli istanti di tempo in cui il pilota è monitorato.
Le righe del file di output saranno strutturate nel seguente modo e ordine:
`<tempo>,<livello rpm>,<livello temperatura>,<livello velocità>`
Inoltre, viene richiesto di aggiungere alla fine del file di output una riga aggiuntiva che contenga i seguenti dati nel seguente ordine:
`<rpm max>,<temp max>,<velocità max>,<velocità media>`

# 2. Flowchart


CALL telemetry
salvataggio registri general purpose nello stack
in lettura nome pilota
ricerca id pilota e conversione in stringa
id != "20"
F
T
out stampa "Invalid\n"
inizializzazione a 0 di rpm massimi, temperatura massima, velocità massima, somma velocità e contatore righe pilota
!EOF
F
T
in lettura id_riga
id == id_riga
F
T
lettura riga a vuoto
incremento contatore righe pilota
out copia tempo da input a output
in lettura velocità
aggiunta velocità alla somma, confronto con il massimo e calcolo livello
in lettura rpm
confronto rpm con il massimo e calcolo livello
in lettura temperatura
confronto temperatura con il massimo e calcolo livello
out stampa livelli rpm, temperatura e velocità
salto alla riga successiva
calcolo velocità media
out stampa rpm massimi, temperatura massima, velocità massima e velocità media
ripristino registri general purpose
RET

# 3. Variabili

| Etichetta | Funzione | Descrizione |
|---|---|---|
| invalid_string | telemetry | Stringa da stampare nel caso in cui il nome del pilota da monitorare non sia valido |
| invalid_pilot_number | telemetry | Stringa “20,” cioè id pilota non valido |
| input | telemetry | Prima riga input, cioè nome del pilota da monitorare |
| pilot_*n* | find_id | Nome del pilota con id *n* (0 <= *n* <= 19) |
| num | itoa | Stringa di supporto per la conversione dell’id trovato |
| rpm_max | elab | Massimi giri motore registrati dal pilota monitorato |
| temp_max | elab | Massima temperatura motore registrata dal pilota monitorato |
| speed_max | elab | Massima velocità registrata dal pilota monitorato |
| speed_sum | elab | Somma di tutti i valori di velocità registrati dal pilota monitorato, utilizzata per il calcolo della velocità media |
| rpm_level | elab | Livello (intero simbolico) di giri motore registrato dal pilota monitorato in un certo istante |
| temp_level | elab | Livello (intero simbolico) di temperatura motore registrato dal pilota monitorato in un certo istante |
| speed_level | elab | Livello (intero simbolico) di velocità registrato dal pilota monitorato in un certo istante |
| rpm_lower_bound | elab | Soglia tra i livelli LOW e MEDIUM degli rpm (si utilizza una variabile perché il valore è troppo grande per essere utilizzato come immediato nell’istruzione) |
| rpm_higher_bound | elab | Soglia tra i livell MEDIUM e HIGH degli rpm (si utilizza una variabile perché il valore è troppo grande per essere utilizzato come immediato nell’istruzione) |
| low | print_level | Stringa “LOW” da stampare per il relativo livello |
| high | print_level | Stringa “HIGH” da stampare per il relativo livello |
| medium | print_level | Stringa “MEDIUM” da stampare per il relativo livello |

# 4. Funzioni

Ogni volta che viene chiamata una funzione, il chiamante svolge le seguenti operazioni

1. Qualora la funzione preveda un valore di ritorno, creazione di spazio nello stack decrementando di 4 l'indirizzo nell'ESP
2. Caricamento dei parametri necessari alla funzione nello stack
3. Chiamata alla funzione e attesa che termini
4. Rimozione dallo stack dei parametri caricati, incrementando di 4*$x$ l'indirizzo nell'ESP, dove $x$ è il numero di parametri
5. Qualora la funzione preveda un valore di ritorno, scaricamento del valore in un registro

Mentre all'interno della funzione si provvede a

1. Salvare nello stack il valore corrente di EBP e successivamente aggiornarlo affinché punti nello stack all'ultimo parametro caricato dal chiamante (considerando che nel mezzo si trova l'indirizzo alla successiva istruzione del chiamante, il precedente PC)
2. Salvare nello stack dei registri che vengono sovrascritti durante l'esecuzione
3. Ripristinare i registri ai valori precedenti alla chiamata scaricandoli dallo stack prima dell'istruzione `ret`

## *4.1. Telemetry*

*Telemetry* è la funzione, entry point del programma assembly, in cui viene letto il nome del pilota dal file di input e, a seconda della validità del nome, viene chiamata la funzione per l'elaborazione oppure stampato "`Invalid\n`" in output.
Riceve come primo parametro un puntatore all'inizio dell'input e come secondo un puntatore all'inizio dell'output, non restituisce alcun valore.

## *4.2. Find_id*

*Find_id* è una funzione chiamata da *telemetry* creata appositamente per confrontare il nome del pilota in input con i nomi dei diversi piloti e trovare il corrispondente id.
Prima di restituire l'id trovato, viene chiamata la funzione *itoa* che converte il numero in stringa per facilitare i successivi controlli.
La funzione riceve come unico parametro un puntatore al nome in input e restituisce un puntatore alla stringa contenente l'id trovato.

## *4.3. Strcmp*

*Strcmp* è una funzione utilizzata in più punti del programma per confrontare due stringhe (o sottostringhe). Riceve come parametri due puntatori a stringhe e confronta carattere per carattere, se i due caratteri non sono uguali la funzione restituisce 0, altrimenti quando si arriva al termine della stringa (`'\0'`, `'\n'` o `','`) restituisce 1.

## *4.4. Itoa*

*Itoa* è la funzione chiamata da *find_id* per convertire l'id trovato da intero a stringa. Riceve come parametro l'intero da convertire e lo divide per 10 finché non rimane una cifra sola, così facendo ad ogni divisione si salva il resto nello stack. Successivamente viene recuperata una cifra alla volta e inserita nella stringa di destinazione sommata a 48 (dato che le codifiche ASCII dei caratteri da `'0'` a `'9'` corrispondono ai numeri da 48 a 57), infine viene restituito un puntatore alla stringa creata.

## *4.5. Elab*

*Elab* viene chiamata da *telemetry* per svolgere la logica principale del programma se il nome del pilota in input risulta valido.
Un ciclo scorre l'input riga per riga dalla seconda all'ultima, per ognuna viene confrontato l'id del pilota a cui fanno riferimento quei dati con quello trovato in input e se non corrispondono si prosegue fino a fine riga a vuoto per passare alla successiva, altrimenti si procede con l'elaborazione.
L'elaborazione consiste nel copiare il tempo della riga nell'output, leggere in ordine i valori istantanei di velocità, rpm e temperatura motore, calcolare i livelli dei valori, confrontarli con i massimi, aggiornare i dati necessari al calcolo della velocità media e stampare in ordine i livelli di rpm, temperatura, velocità.
Una volta letto tutto l'input si procede con il calcolo della velocità media e con la stampa dell'ultima riga, formata in ordine da rpm massimi, temperatura motore massima, velocità massima e velocità media.
Riceve come unico parametro il puntatore alla stringa dell'id trovato e non restituisce alcun valore.

## *4.6. Strcpy_to_int*

*Strcpy_to_int* è una funzione chiamata da *elab* per convertire le stringhe numeriche dei valori in interi. Riceve come primo parametro un puntatore iniziale e come secondo un puntatore finale, scorre uno ad uno i caratteri numerici nell'intervallo e aggiunge la cifra (carattere-48) al valore finale dopo averlo moltiplicato per 10, infine restituisce l'intero risultante.

## *4.7. Find_level*

*Find_level* è una funzione chiamata da *elab* per trovare il livello di un certo valore a seconda della grandezza (quindi degli intervalli) di riferimento. Riceve come primo parametro la soglia per il livello MEDIUM, come secondo la soglia per il livello HIGH e come terzo il valore da paragonare. Con un utilizzo a modo dei salti condizionati e incondizionati, viene trovato il livello e viene restituito un intero simbolico: 1=LOW, 2=MEDIUM, 3=HIGH.

## *4.8. Print_level*

*Print_level* è una funzione chiamata da *elab* per stampare la stringa corretta a seconda del livello. La funzione riceve come primo parametro l'intero simbolico rappresentate il livello e, seguendo la regola sopra descritta, carica in un registro la stringa del livello. Infine, viene stampata la stringa e il carattere finale ricevuto come secondo parametro. Non restituisce alcun valore.

## *4.9. Print_int*

*Print_int* è una funzione chiamata da *elab* per stampare un valore intero. La funzione riceve come primo parametro l'intero da stampare, lo carica nello stack cifra per cifra come mostrato per *itoa*, scarica dalla pila le singole cifre e le stampa dopo avervi aggiunto 48. Infine, stampa il carattere finale ricevuto come secondo parametro e non restituisce niente.

# 5. Scelte progettuali

## *5.1. Stringhe nomi piloti*

I nomi dei piloti sono memorizzati in stringhe contigue in *find_id* e terminano con ‘\n’. Per poter confrontare facilmente tutte le stringhe con quella in input abbiamo pensato di renderle tutte di lunghezza pari alla più lunga, così facendo si può caricare il puntatore alla stringa del primo pilota come parametro per *strcmp* insieme alla stringa in input e se il confronto non va a buon fine basta incrementare il puntatore del numero di caratteri di ogni stringa (19) e procedere dunque con il secondo pilota. Mantenendo un contatore della riga corrente, se questo diventa pari a 20 (piloti numerati da 0 a 19) vuol dire che non c’è alcuna corrispondenza e il nome in input non è valido, altrimenti se *strcmp* restituisce 1 il numero nel contatore è l’id del pilota da monitorare.
La scelta implica un’allocazione di memoria di 80 byte in più, per mettere le stringhe a pari distanza, tuttavia, il tempo di esecuzione è minore perché non è necessario caricare nello stack tutte le stringhe e ne guadagna anche la leggibilità del codice.

## *5.2. Suddivisione in funzioni*

Abbiamo cercato di suddividere il più possibile il progetto in funzioni (e file), dato che assembly è per natura molto verboso questo aiuta notevolmente la stesura e la manutenzione del codice. Inoltre, operazioni come la comparazione di due stringhe, il calcolo e la stampa dei livelli e altre vengono eseguite molteplici volte, quindi, la definizione di funzioni atte solo a un certo scopo è risultata fondamentale nella nostra progettazione.

## *5.3. Itoa e print_int*

Seppur entrambe convertano un numero intero in stringa, abbiamo deciso di separarle e di renderle specifiche per un certo fine.
*Itoa* viene utilizzata per convertire l’id trovato da intero a stringa ed essendo il suo unico indirizzo la stringa di destinazione è di soli 3 caratteri inizializzati a ‘,’. La stringa risultante è occupata da uno o due caratteri di id, mentre i restanti sono virgole con il ruolo di terminatore per il successivo confronto con gli id nelle diverse righe.
*Print_int* è invece pensata per stampare direttamente il numero, dunque, non necessita di una variabile per salvare il risultato e conseguentemente la lunghezza della stringa non è limitata (se non dal numero massimo di cifre che può avere un intero rappresentato con 32 bit).

## *5.4. ESI ed EDI*

I registri general purpose ESI ed EDI vengono utilizzati in alcune funzioni senza essere prima salvati nello stack. Questo perché contengono rispettivamente il puntatore al prossimo carattere in input e al prossimo carattere in output che vengono aggiornati durante l’esecuzione di queste funzioni, dunque, non c’è la necessità di passarli come parametri e di doverli sincronizzare una volta tornati al chiamante.
Talvolta il loro contenuto viene utilizzato normalmente come parametro per delle funzioni dato che le modifiche da queste apportate non influenzano la progressione dei puntatori.
ESI ed EDI vengono salvati e successivamente ripristinati con l’ausilio dello stack in

- *strcpy_to_int*, perché vengono sovrascritti con altri valori di supporto alla funzione
- *telemetry*, perché, non avendo controllo sull’utilizzo dei registri precedentemente alla sua chiamata, se ne preserva lo stato a priori