Dipartimento di informatica

Corso di laurea triennale in informatica

Elaborato ASM - Laboratorio Architettura degli Elaboratori

Edoardo Bazzotti - Davide Cerullo
27/05/2022

Sommario

Descrizione del progetto

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, nel seguente ordine: il numero di giri massimi rilevati, la temperatura massima rilevata, la velocità di picco e infine la velocità media.
La struttura dell'ultima riga sarà quindi la seguente:
<rpm max>,<temp max>,<velocità max>,<velocità media>

Le soglie per i dati monitorati sono così definite:
- Giri Motore
    - o LOW: rpm <= 5000
    - o MEDIUM: 5000 < rpm <=10000
    - o HIGH: rpm > 10000
- Temperatura
    - o LOW: temp <= 90
    - o MEDIUM: 90 < temp <= 110
    - o HIGH: temp > 110
- Velocità
    - o LOW: speed <= 100
    - o MEDIUM: 100< speed <=250
    - o HIGH: speed > 250

Variabili utilizzate

Le variabili utilizzate nel file sono le seguenti (escludendo quelle già presenti di default es. pilot_0_str ecc…):

- stringa_virgola (tipo ascii): serve per stampare una virgola nell'output
- stringa_invio (tipo ascii): serve per andare a capo nell'output
- contatoreCaratteri (tipo long): variabile che serve a contare i caratteri nella funzione a2file (per l'output)
- count (tipo int): serve per tenere traccia di dove siamo all'interno del file tramite il conteggio delle virgole
- flag (tipo int): variabile sentinella utilizzata in caso di errori nel trovare l'id del pilota
- record_pilota (tipo long): contiene il numero di quante volte è stato trovato il pilota all'interno del file
- id (tipo int): contiene l'id del pilota cercato
- rpmMax (tipo long): contiene il valore degli rmp più alti trovati
- tempMax (tipo long): contiene il valore del tempo più alto trovato
- velocitàMax (tipo long): contiene il valore della velocità più alta trovata
- velocitàMedia (tipo long): contiene il valore della velocità media
- intToPrint (tipo ascii): contiene la stringa da stampare, quando dobbiamo convertire un numero di stringa (valori di ritorno di itoa)
- intToPrint_len (tipo long): numero intero che rappresenta la lunghezza della stringa contenuta nella variabile intToPrint, quando dobbiamo convertire un numero di stringa (valori di ritorno di itoa)
- rpm_long (tipo long): contiene il valore degli rpm attuali
- temperatura_long (tipo long): contiene il valore della temperatura attuale
- velocità_long (tipo long): contiene il valore della velocità attuale
- id_str (tipo ascii): contiene l'id sotto forma di stringa (verrà resettata per il prossimo input)
- temp_str (tipo ascii): contiene il tempo sotto forma di stringa
- temp_str_len (tipo long): contiene il numero dei caratteri utilizzati dal tempo
- LOW (tipo ascii): contiene la stringa LOW

- MEDIUM (tipo ascii): contiene la stringa MEDIUM,
- HIGH (tipo ascii): contiene la stringa HIGH
- num1_len (tipo long): variabile che serve a contare i caratteri nella funzione atoi

Le modalità di passaggio/restituzione dei valori delle funzioni create

- leggi_nome:
  Funzione che permette di leggere il nome del pilota e ne associa il corrispettivo id.
  Utilizzando la stringa contenuta nel registro esi e lo spiazzamento in ecx, scorriamo la stringa carattere per carattere fino allo \n (invio), utilizzando la variabile count (default 0), confrontiamo il carattere con la rispettiva stringa pilot_n_str dove count = n.
  Durante il confronto se otteniamo un carattere non uguale la variabile count viene incrementata in modo da passare al confronto con il prossimo pilota, questo viene ripetuto fino ad ottenere 2 stringhe equivalenti, oppure in caso di nome del pilota inesistente (count = 20) si restituisco la stringa di invalid_pilot_str e la flag viene posta ad 0.
  Alla fine della funzione si copia il valore della variabile count nella variabile id.

- idCompare
  Funzione che confronta l'id preso in input dalla stringa appena letta (convertito in intero con la funzione atoi) e l'id restituito dalla funzione leggi_nome e se sono diversi non do in output niente, ma se sono uguali incremento la variabile record_pilota e do in output la stringa appena letta, nella seguente maniera:
  1. sposto temp_str in %ecx e temp_str_len in %edx, dopodichè chiamo la funzione a2file (per scrivere il tempo sul file in output)
  2. stampo la virgola per separare i valori
  3. confronto rpm_long con rpmMax e copio il valori più grande in rpmMax, dopo confronto rpm_long con dei valori per definira la sua soglia, <5000 è LOW, 5000<rpm_long<10000 è MEDIUM, >10000 è HIGH, una volta fatti questi confronti scelgo quale valore dare in output, spostando la relativa stringa in %ecx e la sua lunghezza in %edx, chiamando poi a2file.
  4. stampo la virgola
  5. per la temperatura faccio gli stessi passaggi del punto 3, con le rispettive soglie <90 è LOW, 90<temperatura_long<110 è MEDIUM, >110 è HIGH; anche in questo caso cerco la temperatura massima (tempMax).

6. stampo la virgola
7. per la velocità faccio gli stessi passaggi del punto 3, con le rispettive soglie <100 è LOW, 100<temperatura_long<250 è MEDIUM, >250 è HIGH; anche in questo caso cerco la veloctià massima (velocitàMax) ed inoltre sommo la velocità con la variabile velocitàMedia (alla fine del file la dividerò per il numero delle velocità sommare, record_pilota).
8. stampo invio

- a2file
  Funzione che data una stringa in ecx e la sua lunghezza in edx, la scrive in edi (output file).
  Questa funzione come prima cosa controlla che %edx sia diverso da 0 (altrimenti avremmo già scritto tutta la stringa), se è così prende il carattere in %ecx + %eax (variabile contatore) e lo sposta in %edi + contatoreCaratteri (variabile contatore caratteri in output), dopodichè incremento %eax e contatoreCaratteri e decrementa %edx.

- atoi
  Funzione che data una stringa in %eax, la converte in intero e la restituisce in %ecx.
  Prendo il carattere in posizione %eax + num1_len (contatore dei caratteri), controllo che non sia nullo, dopodichè gli tolgo 48 per convertirlo ad intero, ora moltiplico per 10 %edx, così da liberare un posto per poi sommare il numero convertito; incremento num1_len e vado avanti con il ciclo per la conversione.

- itoa
  Funzione che, dato un intero in %ecx, lo converte in ascii e lo restituisce nella variabile intToPrint e restituisce la sua lunghezza in intToPtint_len.
  Come prima cosa la funzione copia il valore di %ecx in %eax, dopodiche entro nel loop dove si conta il numero dei caratteri che dovrò convertire (divido il numero per 10 ed ogni volta incrementi intToPrint_len fino a che il risultato della divisione non è 0), adesso entro nel loop che converte i numeri in ascii nel seguente modo: controllo se intToPrint_len è uguale a 0 (in tal caso ho già convertito tutti i caratteri), se così non è divido %eax per %ebx (impostato a 10), prendo il resto della divisione gli aggiungo 48 per trovare il suo relativo valore in ascii e lo inserisco nella stringa intToPrint nella posizione intToPrint_len-1 (il conteggio inizia da 0), adesso decremento intToPrint_len e continuo il ciclo.

Pseudo codice C

```c
#include <stdio.h>

// dichiaro i nomi dei piloti
char pilot0[] = "Pierre Gasly\0";
char pilot1[] = "Charles Leclerc\0";
char pilot2[] = "Max Verstappen\0";
char pilot3[] = "Lando Norris\0";
char pilot4[] = "Sebastian Vettel\0";
char pilot5[] = "Daniel Ricciardo\0";
char pilot6[] = "Lance Stroll\0";
char pilot7[] = "Carlos Sainz\0";
char pilot8[] = "Antonio Giovinazzi\0";
char pilot9[] = "Kevin Magnussen\0";
char pilot10[] = "Alexander Albon\0";
char pilot11[] = "Nicholas Latifi\0";
char pilot12[] = "Lewis Hamilton\0";
char pilot13[] = "Romain Grosjean\0";
char pilot14[] = "George Russell\0";
char pilot15[] = "Sergio Perez\0";
char pilot16[] = "Daniil Kvyat\0";
char pilot17[] = "Kimi Raikkonen\0";
char pilot18[] = "Esteban Ocon\0";
char pilot19[] = "Valtteri Bottas\0";
char invalid_pilot[] = "Invalid";

char *piloti[] = {{pilot0}, {pilot1}, {pilot2}, {pilot3}, {pilot4}, {pilot5}, {pilot6}, {pilot7},
{pilot8}, {pilot9}, {pilot10}, {pilot11}, {pilot12}, {pilot13}, {pilot14}, {pilot15}, {pilot16},
{pilot17}, {pilot18}, {pilot19}};

// Prototipi
int leggiNome(char *, char *, int *, int);

int main(int argc, char *argv[])
{
   // inizializzazione variabili
   char *inputString = argv[1]; // recupero la stringa del nome del file di input
   char *outputString = argv[2];

   char id_str[] = {'\0','\0','\0'};   //Stringa per l'id
   char temp_str[] = {'\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0'}; //Stringa per il
tempo

   int id = 20;   // id del pilota che voglio cercare
   int count = 0; // contatore del numero delle virgole
   int flag = 1;  // variabile per il controllo dell'esecuzione del programma
   int invio = 1; // variabile che controlla la lettura dell'invio
   int inputCount = 0; //variabile per la posizione dell'inserimento dei dati
   int temp_str_len = 0;   //variabile che memorizza la lunghezza della stringa del tempo inserito
```

```
   int record_pilota = 0;  //variabile che memorizza il numero di record del pilota trovati nel
file in input
   int output_count_char = 0;  //variabile che serve per gestire i caratteri da dare in output

   //Variabili che conterranno i relativi dati
   int velocita_long = 0;
   int temperatura_long = 0;
   int rpm_long = 0;

   //variabili per l'output finale
   int rpm_max = 0;
   int temperatura_max = 0;
   int velocità_max = 0;
   int velocità_media = 0;

   // ciclo che scorre la stringa fino alla fine
   for (int i = 0; inputString[i] != '\0'; i++)
   {
       // controllo se ho già trovato l'id del pilota che devo cercare
       if (id == 20)
       {

           // Richiamo la funzione
           id = leggiNome(inputString, outputString, &i, count);
       }

       count = 0; // resetto il numero delle virgole

       // controllo se ci sono stati errori nella lettura dell'id
       if (flag == 0)
       {
           return 0;
       }

       // leggi_dati
       if (inputString[i] == ',') // se leggo una virgola
       {
           count++;
           inputCount = 0;
       }
       if (inputString[i] == '\n') // se leggo l'invio
       {
           idCompare(id, temp_str, id_str, velocita_long, rpm_long, temperatura_long, temp_str_len,
&record_pilota, &rpm_max, &temperatura_max, &velocità_max, &velocità_media, &output_count_char,
outputString); // funzione per il controllo dell'id


           // resetto le variabili
           count = 0;
           inputCount = 0;
```

```
            rpm_long = 0;
            temperatura_long = 0;
            velocita_long = 0;
            temp_str_len = 0;
            id_str[0] = '\0';
            id_str[1] = '\0';
            id_str[2] = '\0';
        }
        else
        {
            //Controlli per l'inserimento dei dati
            switch (count)
            {
                case 0:     //tempo
                    temp_str[inputCount] = inputString[i];
                    inputCount++;
                    temp_str_len = inputCount;
                    break;
                case 1:     //id
                    id_str[inputCount] = inputString[i];
                    inputCount++;
                    break;
                case 2:     //velocità
                    //converto il tempo in intero
                    velocita_long = velocita_long*10;
                    velocita_long += (inputString[i]-48);
                    break;
                case 3:     //rpm
                    //converto gli rpm in intero
                    rpm_long = rpm_long*10;
                    rpm_long += (inputString[i]-48);
                    break;
                case 4:     //temperatura
                    //converto la temperatura in intero
                    temperatura_long = temperatura_long*10;
                    temperatura_long += (inputString[i]-48);
                    break;
            }
        }
    }

    //output finale
    a2file(outputString, &output_count_char, itoa(rpm_max), strlen(itoa(rpm_max)));  //output rpm
    a2file(outputString, &output_count_char, ',', 1);    //stampo la virgola ','

    a2file(outputString, &output_count_char, itoa(temperatura_max), strlen(itoa(temperatura_max)));
//output temperatura max
    a2file(outputString, &output_count_char, ',', 1);    //stampo la virgola ','
```

```
   a2file(outputString, &output_count_char, itoa(velocità_max), strlen(itoa(velocità_max)));
//output velocità max
   a2file(outputString, &output_count_char, ',', 1);    //stampo la virgola ','

   velocità_media = velocità_media/record_pilota;  //calcolo della velocità media
   a2file(outputString, &output_count_char, itoa(velocità_media), strlen(itoa(velocità_media)));
//output velocità media

   return 0;
}

// Funzione che permette di leggere il nome del pilota
int leggiNome(char *inputString, char *outputString, int *numChar, int count)
{
   char carattere = inputString[*numChar];

   if (count == 20)
   {
       // Output invalid pilot
       outputString = invalid_pilot;
       return count;
   }

   // se leggo il carattere \n vuol dire che sono arrivato alla fine del nome
   if (carattere == '\n')
   {
       return count;
   }

   // Controllo il carattere con il carattere del pilota numero count
   if (carattere != piloti[count][*numChar])
   {
       *numChar = 0;
       count++;
       leggiNome(inputString, outputString, numChar, count);
   }
   else
   {
       *numChar++;
       leggiNome(inputString, outputString, numChar, count);
   }
}

void idCompare(int id, char *temp_str, char *id_str, int velocità_long, int rpm_long, int
temperatura_long, int temp_str_len, int *record_pilota, int *rpm_max, int *temperatura_max, int
*velocità_max,int *velocità_media, int *output_count_char, char *outputString)
{
   //se gli id corrispondono
   if(id == atoi(id_str))
   {
```

```
        *record_pilota++;

        //tempo
        a2file(outputString, output_count_char, temp_str, temp_str_len);

        a2file(outputString, output_count_char, ',', 1);    //stampo al virgola ','

        //rpm
        //Controllo per gli rpm massimi
        if(rpm_long > *rpm_max)
        {
            *rpm_max = rpm_long;
        }

        if(rpm_long <= 5000)
        {
            a2file(outputString, output_count_char, "LOW", 3);
        }
        else if (rpm_long <= 10000)
        {
            a2file(outputString, output_count_char, "MEDIUM", 6);
        }
        else
        {
            a2file(outputString, output_count_char, "HIGH", 4);
        }

        a2file(outputString, output_count_char, ',', 1);    //stampo al virgola ','

        //temperatura

        //Controllo per la temperatura massima
        if(temperatura_long > *temperatura_max)
        {
            *temperatura_max = temperatura_long;
        }
        if(temperatura_long <= 90)
        {
            a2file(outputString, output_count_char, "LOW", 3);
        }
        else if (temperatura_long <= 110)
        {
            a2file(outputString, output_count_char, "MEDIUM", 6);
        }
        else
        {
            a2file(outputString, output_count_char, "HIGH", 4);
        }

        a2file(outputString, output_count_char, ',', 1);    //stampo al virgola ','
```

```
        //Velocità massima
        if(velocità_long > *velocità_max)
        {
            *velocità_max = velocità_long;
        }
        if(velocità_long <= 100)
        {
            a2file(outputString, output_count_char, "LOW", 3);
        }
        else if (velocità_long <= 250)
        {
            a2file(outputString, output_count_char, "MEDIUM", 6);
        }
        else
        {
            a2file(outputString, output_count_char, "HIGH", 4);
        }

        a2file(outputString, output_count_char, '\n', 1);    //stampo lo '\n'

        //Velocità media
        *velocità_media += velocità_long;
    }

    return;
}

//metodo che stampa sul file
void a2file(char *outputString, int *output_count_char, char *stringa, int str_len)
{
    //scorro la stringa che devo dare in output e la inserisco caratttere per carattere
    for(int i = 0; i < str_len; i++)
    {
        outputString[*output_count_char] = stringa[i];
        *output_count_char++;
    }

    return;
}

//metodo che converte i numeri da interi ad ascii e li memorizza in una stringa
char* itoa(int num)
{
    //inizializzazione veriabili
    int count = 0;
    int tempNum = num;

    //ciclo che conta il numero dei caratteri che dovrò inserire
    while(tempNum > 0)
```

```
    {
        count++;
        tempNum = tempNum/10;
    }

    int copyCount = count;
    char *strNum = (char*)malloc((count+1)*sizeof(char));   //dichiaro la stringa che conterrà il numero convertito

    //ciclo che converte le cifre in ascii e le memorizza in una stringa
    while(num > 0)
    {
        strNum[count-1] = (num%10)+48;    //converto la cifra da intero a stringa, uso count-1 perché il numero che va alla seconda posizione per l'array va messo alla posizione 1
        count--;
        num = num/10;
    }

    strNum[copyCount] = '\0';     //setto già il terminatore alla stringa

    return strNum;
}
```

Scelte Progettuali

- Per la realizzazione di questo programma, abbiamo deciso che prima di dare in output i valori del record del pilota leggiamo tutta la riga fino in fondo, memorizzando i relativi valori di tempo, temperatura e velocità nelle apposite variabili, per fare queste effettuiamo un conteggio delle virgole ed in base al numero di virgole contate sappiamo quale variabile stiamo andando a leggere; una volta giunti al termine della riga controlliamo se l'id è quello che cerchiamo, in caso affermativo avviene l'output, altrimenti resettiamo le variabili usate per contenere i valori ed il contatore delle virgole.
- Per memorizzare i valori della velocità e della temperatura abbiamo scelto di convertire la stringa in intero, in modo da rendere poi più agevole le operazioni di confronto per l'output; invece per quanto riguarda il tempo lo memorizziamo in una stringa ed utilizziamo un apposito contatore per tenere traccia del numero dei caratteri utilizzati; per l'id memorizziamo anch'esso in una stringa che dopo forniremo all idCompare per il confronto con l'id da cercare.