UNIVERSITÀ DEGLI STUDI DI VERONA

Relazione Elaborato ASM

Architettura degli elaboratori

Anno Accademico 2015/2016

Breve descrizione:
Programma Assembly che effettua il monitoraggio di un motore a combustione interna restituendo lo stato attuale di funzionamento, il tempo trascorso in tale stato e un allarme fuori giri se sono trascorsi 15 secondi in tale stato.

Studenti

Bertoncelli Giovanni - VR399929
Girelli Alberto - VR397173
Righi Edoardo - VR398499

Sommario

Descrizione del progetto

Il programma ha come obiettivo di monitorare l'andamento dei giri di un motore a combustione interna. Il programma assembly riceve dal file **nameinput.txt** per ogni riga i valori seguenti:

INIT,RESET,RPM

- INIT: valore binario, quando vale 0 il programma restituisce una linea di soli 0, ovvero la macchina è spenta; quando vale 1 il programma deve fornire in uscita tutti i valori valutati in base allo stato della macchina.
- RESET: valore binario, se vale 1 il contatore dei secondi viene azzerato.
- RPM: valore del numero di giri attuali ricevuto dal rilevatore (valore massimo fissato a 6500).

Il programma deve restituire i risultati del calcolo nel file **nameoutput.txt** nel quale ogni riga contiene i valori seguenti:

ALM,MOD,NUMB

- ALM: valore binario, quando vale 1 significa che sono trascorsi più di 15 secondi nello stato di fuori giri (FG)
- MOD: valore che indica in quale modalità di funzionamento si trova il motore, in particolare i vari stati sono:
  - Spento = 00 per RPM < 2000
  - Sotto giri SG = 01 per 2000 ≤ RPM ≤ 4000
  - Regime ottimale OPT = 10 per RPM > 4000
  - Fuori giri FG = 11
- NUMB: valore binario che indica il numero di secondi trascorsi nello stato attuale.

Il programma si sviluppa su più file:

- **elaboratoASM.s** è il *main* del programma
- **atoi.s** contiene la funzione per convertire il codice ASCII della cifra nel numero corrispondente
- **valutazione.s** contiene la funzione che valuta lo stato del motore
- **write_output.s** contiene la funzione per la stampa su file e il messaggio di conferma esecuzione sul terminale
- **makefile** è il file per la compilazione del programma che genera l'eseguibile.

Come da specifiche non è necessaria la stampa a video, tuttavia abbiamo inserito un Done! alla fine dell'esecuzione come debug e per segnalare l'operazione di scrittura sul file nameoutput.txt completata.

Variabili utilizzate

Le variabili presenti nel programma sono le seguenti:

File **elaboratoASM.s**

- counter: variabile di tipo long inizializzata a 0 che serve per scorrere la singola riga del file;
- out_counter: variabile di tipo long inizializzata a 0 che serve per indicare la posizione nell'out_buffer;
- buff_size: variabile di tipo long inizializzata a 900 che serve per la dimensione massima del file input ovvero 100 righe per N caratteri presenti nel file nameinput.txt ovvero 9 caratteri (INIT,RESET,RPM\n).

Variabili di progetto:

- **INIT**: variabile di tipo long inizializzata a 0 in cui viene salvato il valore convertito dalla funzione *atoi* di INIT (primo valore della riga del file input);
- **RESET**: variabile di tipo long inizializzata a 0 in cui viene salvato il valore convertito dalla funzione *atoi* di RESET (secondo valore della riga del file input);
- **RPM1**, **RPM2**, **RPM3** e **RPM4**: variabili di tipo long inizializzate a 0 in cui vengono salvate ad una ad una le cifre convertite dalla funzione *atoi* degli RPM (terzo valore della riga del file input);
- **RPM_S**: variabile di tipo long inizializzata a 0 che contiene il valore reale degli RPM in seguito alla somma dei singoli interi in cui sono stati divisi gli RPM provenienti dal file di input;
- **MOD**: variabile di tipo long inizializzata a 0 che contiene lo stato di funzionamento calcolato nel file valutazione.s;
- **NUMB**: variabile di tipo long inizializzata a 0 che contiene il numero di secondi trascorsi nello stato attuale calcolati nel file valutazione.s;
- **ALM**: variabile di tipo long inizializzata a 0 che contiene il valore dell'allarme calcolato nel file valutazione.s;
- **Temp_counter**: variabile di tipo long inizializzata a 0 che serve da contatore durante la pseudo itoa per i valori a due cifre (MOD e NUMB);
- **MOD_PREC_M**: variabile di tipo long inizializzata a 0 che contiene lo stato precedente a quello valutato, e se è diverso dal MOD attuale porta al reset del conteggio del tempo.

Variabili di posizione:

- comma: variabile di tipo byte utilizzata per la stampa della virgola tra i valori ALM MOD e NUMB;
- newline: variabile di tipo byte utilizzata per stampare il capo riga ad ogni fine riga.

Variabili file di lettura:

- **buffer**: alloca dinamicamente 900 byte per i caratteri del file input;
- **out_buffer**: alloca dinamicamente 900 byte per i caratteri da stampare nel file output;
- **descriptor_in**: contiene la descrizione del file di input ottenuto durante l'apertura;
- **file_out**: contiene il nome del file di output

File **atoi.s**

- **car**: variabile di tipo long inizializzata a 0 che contiene il carattere letto per la conversione nel valore corrispondente;

File **valutazione.s**

- **INIT_t**: variabile di tipo long inizializzata a 0 che contiene il valore ricevuto dal registro su cui è salvato il valore della variabile INIT
- **RESET_t**: variabile di tipo long inizializzata a 0 che contiene il valore ricevuto dal registro su cui è salvato il valore della variabile RESET
- **RPM_t**: variabile di tipo long inizializzata a 0 che contiene il valore ricevuto dal registro su cui è salvato il valore della variabile RPM_S
- **MOD_PREC_t**: variabile di tipo long inizializzata a 0 che contiene il valore ricevuto dal registro su cui è salvato il valore della variabile MOD_PREC_M
- **COUNTER_t**: variabile di tipo long inizializzata a 0 che viene incrementata se lo stato di funzionamento valutato rimane invariato mentre viene azzerato se diverso da quello precedente.

File **write_output.s**

Variabili di progetto

- **done**: stringa contenente la scritta Done ! che viene mostrata a fine esecuzione;
  - **done_size**: identifica la dimensione della stringa di tipo .ascii contenuta nella variabile done;
- **out_counter2**: variabile di tipo long inizializzata a 0 che indica la posizione sul file nella quale viene stampato il carattere;
- **out_buffer2**: variabile di tipo long inizializzata a 0 che contiene l'indirizzo dell'out_buffer del file main elaboratoASM.s;
- **out_pointer**: variabile di tipo long inizializzata a 0 che è di supporto ad out_buffer2 nel puntare; l'indirizzo di stampa;
- **char_temp**: variabile di tipo byte usata per salvare temporaneamente i caratteri da stampare.

Variabili file di scrittura

- **descriptor_out**: contiene la descrizione del file di output ottenuto durante l'apertura.

Modalità di passaggio/restituzione dei valori delle funzioni create

Funzione **atoi.s**

La funzione riceve in input dal registro eax il carattere letto (salvato precedentemente in %al, parte del registro eax) che viene salvato nella variabile car, successivamente viene sottratto a car il numero 48 per convertire il codice ASCII della cifra nel numero corrispondente. Infine il contenuto della variabile car viene spostato nel registro eax, che è quindi l'output della funzione.

N.B: essendo una chiamata a funzione viene fatto un push sullo stack prima dell'inizio per salvare i registri ebx ecx ed edx e successivamente per ripristinare la situazione precedente viene fatto un pop. Tuttavia solamente il registro ecx viene modificato durante l'esecuzione della funzione, mentre ebx ed edx non concorrono allo svolgimento di tale funzione.

Funzione **valutazione.s**

La funzione riceve in input dai registri eax, ebx, ecx ed edx i valori contenuti precedentemente in INIT, RESET, RPM_S e MOD_PREC_M che vengono poi salvati nelle variabili INIT_t, RESET_t, RPM__t e MOD_PREC_t. Tramite questa funzione viene effettuata la valutazione dello stato della macchina e vengono restituiti i valori nei vari registri di partenza.

Funzione **write_output.s**

La funzione riceve in input l'indirizzo dell'out_buffer, apre il file di output e, successivamente, a partire dall'inizio della stringa ottenuta nel file *main*, scrive su file tutti i caratteri dell'output.

Pseudo funzione itoa incorporata nel *main* e preparazione per la scrittura su file di output

Dopo la valutazione dello stato della macchina è necessario riconvertire i valori in caratteri stampabili, dunque per fare questo non è stata implementata una funzione come per atoi, ma per ogni variabile di output (ALM, MOD e NUMB) viene aggiunto il numero 48 per convertire in codice ASCII il valore, il risultato viene salvato sulla variabile out_buffer e si incrementa out_counter che servirà successivamente per scorrere i caratteri durante la scrittura su file. In particolare per MOD e NUMB, essendo composti da due cifre, per convertire il valore in carattere è necessario operare su una cifra alla volta perciò si impiega la divisione per 10 del valore di partenza per prendere rispettivamente la prima e la seconda cifra.
Inoltre per separare i valori con una virgola viene salvato il carattere virgola sull'out_buffer incrementando sempre l'out_counter. Stessa cosa per il caporiga.

Descrizione del flusso del programma

Il programma si suddivide in cinque fasi principali:

- Lettura dal file nameinput.txt (apertura file e lettura)
- Conversione dei caratteri del codice ASCII in valori numerici
- Valutazione dello stato della macchina
- Conversione dei valori in caratteri per la scrittura su file
- Scrittura su file nameoutput.txt (apertura file e scrittura)

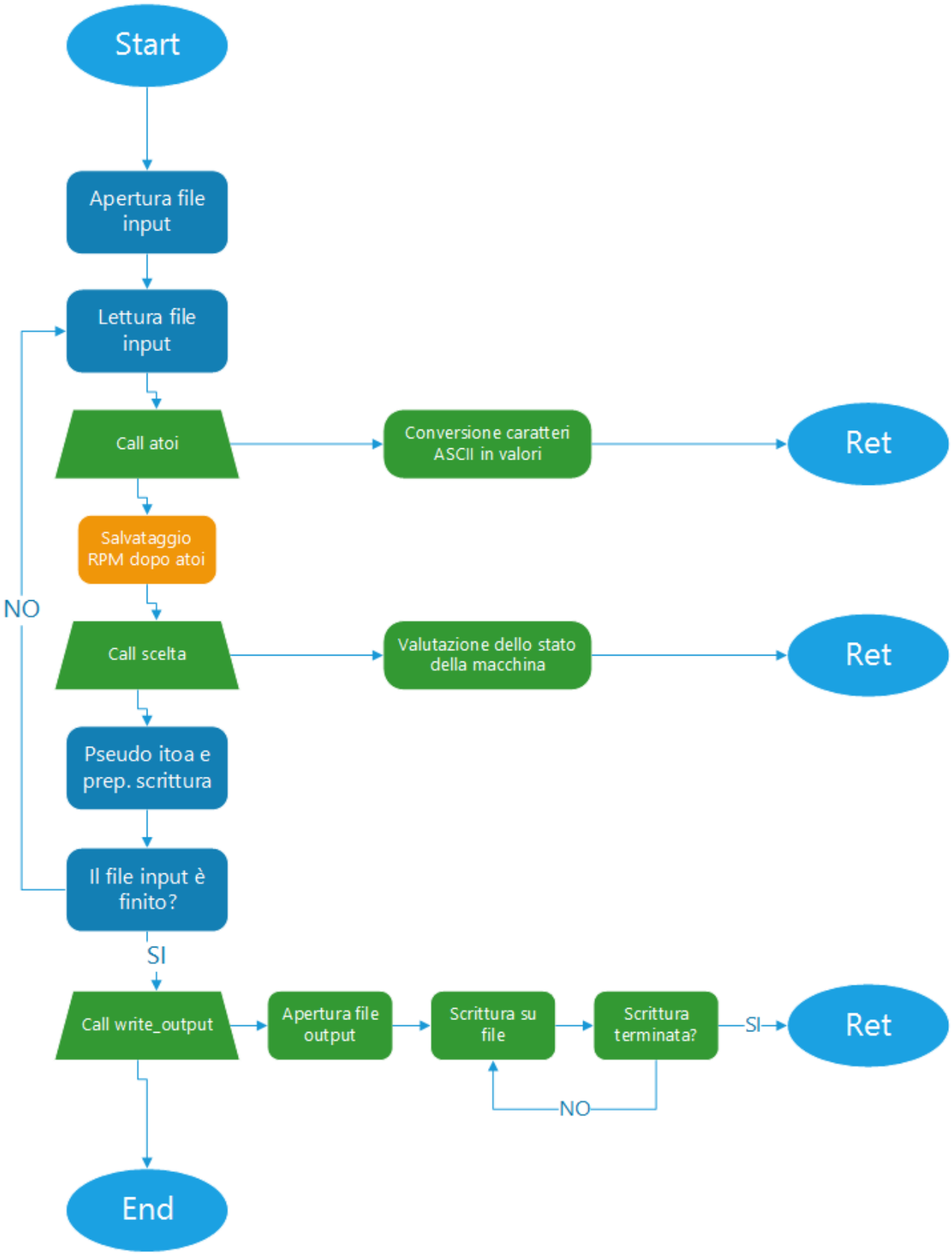
In fase di esecuzione vengono designati due file con estensione .txt uno per l'input e un altro per l'output. Il programma dunque apre il primo file, legge i caratteri contenuti all'interno e li converte in valori (funzione atoi contenuta nel file atoi.s), successivamente avviene la valutazione in base agli RPM in ingresso e a INIT e RESET (funzione scelta contenuta nel file valutazione.s). Fatto ciò si passa di nuovo alla conversione dei valori in caratteri stampabili (che stavolta avviene nel *main*) e dopo l'apertura del secondo file avviene la scrittura (funzione write_output contenuta nel file write_output.s).

In particolare per la valutazione dello stato della macchina abbiamo dapprima rappresentato il possibile procedimento di risoluzione con un codice scritto in C riportato di seguito:

```
if (INIT !=  0)
     if (RESET != 1) {
          if (RPM > 4000) {
               if (C > clock_max)
{
                    ALM = 1;
               }
          MOD =  FG (11);
          C++;
     }
else if (RPM < 4000) {
     if (RPM > 2000)
          MOD = OPT (10);
     else
          MOD = SG (01);
     C++;
}
     else //RESET
          C = 0;
else //INIT
     ALM = 0;
     NUMB = 00;
     MOD = 00;
```

/* se INIT e RESET sono diversi da da 0
si procede a valutare gli RPM:
se sono maggiori di 4000 e il C
(conteggio dei secondi) ha superato il
il clock_max di 15 secondi ALM è 1
Inoltre viene impostato MOD in FG
Incremento C i secondi
se invece gli RPM sono inferiori a
4000 e superiori a 2000 MOD è OPT
invece se RPM sono inferiori a 2000
MOD è SG
Incremento C per i secondi
Se invece RESET è uguale a 1
C viene azzerato
Se invece INIT è uguale a 0
ALM, NUMB e MOD sono posti a 0 */

Schema approssimativo di funzionamento del programma:



Descrizione delle scelte progettuali

Pseudo funzione itoa incorporata nel *main*

Durante l'implementazione del codice si è visto che non era strettamente necessario operare la conversione da valore a carattere ASCII tramite una funzione separata dal codice principale, dunque è stata integrata nella fase di preparazione (scrittura su out_buffer) per la scrittura sul file output.

Apertura e scrittura su file output tramite funzione separata write_output

Per ottenere un codice più ordinato si è deciso di separare la stampa su file dal *main*.

NUMB posto a -1 in caso di RESET = 1

La variabile NUMB riceve il valore di -1 in caso di RESET = 1 a causa di un successivo incremento, in questo modo alla fine si otterrà 0.