UNIVERSITÀ DEGLI STUDI DI VERONA

Dipartimento di Informatica

Laboratorio di Architettura degli Elaboratori

a.a. 2016/2017

ELABORATO ASM

"Dispositivo monitoraggio impianto chimico industriale"

Studenti:
Lutteri Nicolò
Nikollbibaj Donika

Docente:
Setti Francesco

indice

1

Specifiche del Progetto

Si ottimizzi un codice in linguaggio C che effettua il monitoraggio di un impianto chimico industriale mediante l'uso di Assembly inline. Ricevendo come input il pH di una soluzione contenuta in un serbatoio e, una volta impostate le soglie minima e massima per il funzionamento ottimale, il programma fornisce in uscita lo stato della soluzione in termini di acido (*A*), basico (*B*) e neutro (*N*). Il sistema deve portare sempre la soluzione allo stato neutro, accettando un transitorio di 5 cicli di clock; pertanto si richiede che al sesto ciclo di clock in cui il sistema sia allo stato *A*, venga aperta una valvola con soluzione basica (*BS*), e analogamente se allo stato *B* si apra la valvola con soluzione acida (*AS)*.
Il sistema deve inoltre fornire in uscita il numero di cicli di clock da cui si trova nello stato attuale.
Il programma deve essere lanciato da riga di comando con due stringhe come parametri, la prima stringa identifica il nome del file `.txt` da usare come input, la seconda quello da usare come output:

```
$ ./controller testin.txt testout.txt
```

Il programma deve leggere il contenuto di `testin.txt` contenente in ogni riga i seguenti valori:

INIT,RESET,PH

- **INIT** [1]: valore binario, quando vale **1** il sistema è acceso; quando vale **0** il sistema è spento e deve restituire una linea composta da soli **0**.
- **RESET** [1]: quando posto a **1** il controllore deve essere resettato, ovvero tutte le uscite devono essere poste a **0** e il sistema riparte.
- **PH** [3]: valore del pH misurato dal rilevatore. Il range di misura è compreso tra **0** e **14** con risoluzione di **0,1**. Il valore è espresso in decimi di pH e sempre riportato in 3 cifre, ad esempio 065 corrisponde a 6,5.

Il programma deve restituire i risultati del calcolo in `testout.txt` in cui ogni riga contiene:

ST,NCK,VLV

- **ST** [1]: indica in quale stato si trova la soluzione al momento corrente (acida – A, basica – B o neutra - N)
- **NCK** [2]: indica il numero di cicli di clock trascorsi nello stato corrente.
- **VLV** [2]: indica quale valvola aprire per riportare la soluzione allo stato neutro nel caso in cui la soluzione si trovi da più di 5 cicli di clock in stato acido (BS) o basico (AS).

Si considerino i seguenti valori di soglia:

- pH < 6.0 : Acido
- 6.0 ≤ pH ≤ 8.0 : Neutro

2 Descrizione cartella

La cartella consegnata contiene i seguenti file:

- **controller.c**

  Sorgente C fornito, integrato con la macchina realizzata tramite un extern module;

- **controller-asm.s**

  Macchina vera e propria scritta in assembly;

- **testin.txt**

  File di test fornito al fine di verificare il corretto funzionamento del codice;

- **testout.txt**

  File di uscita risultate della nostra macchina;

- **trueout.txt**

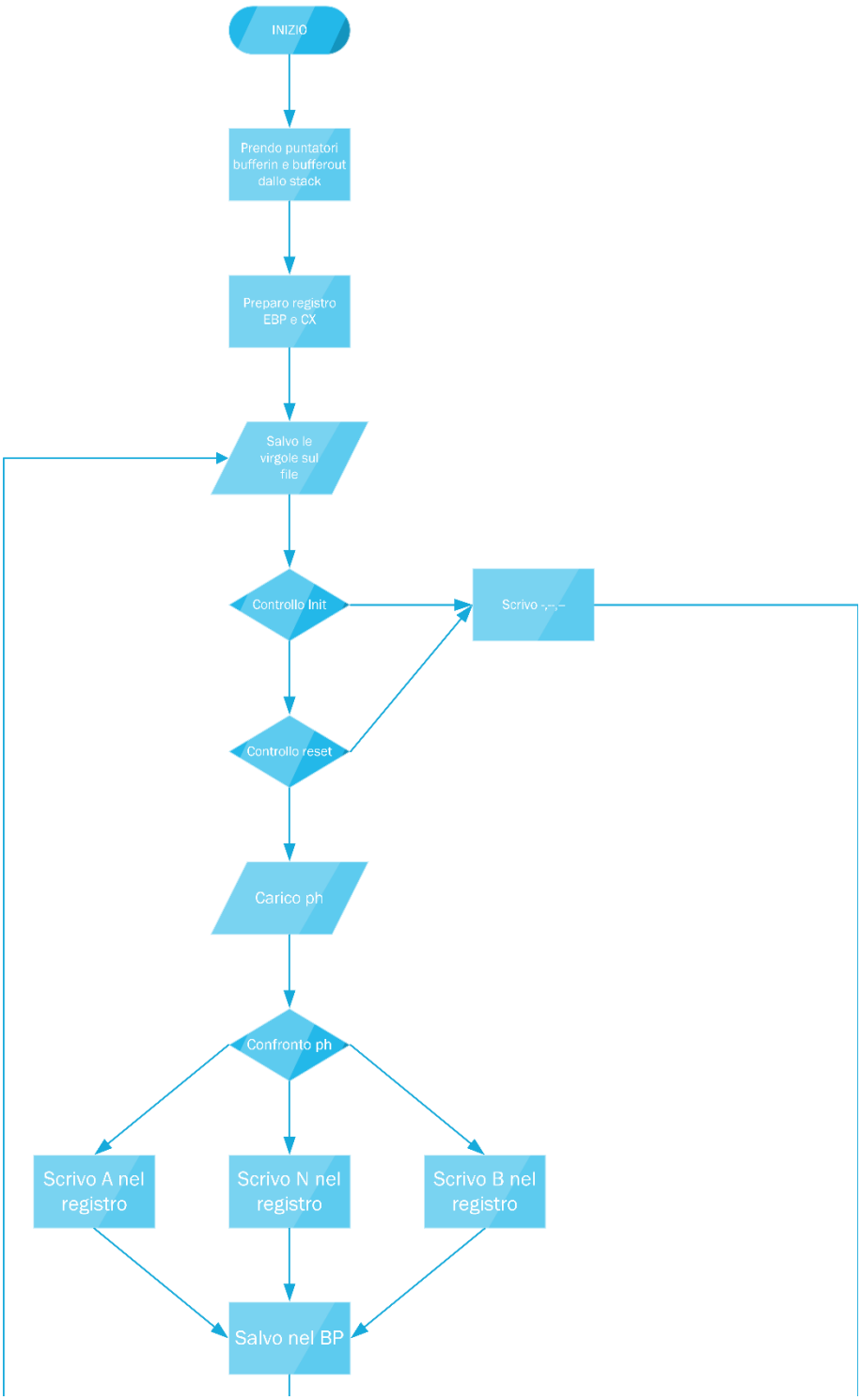
  Test di uscita risultante.

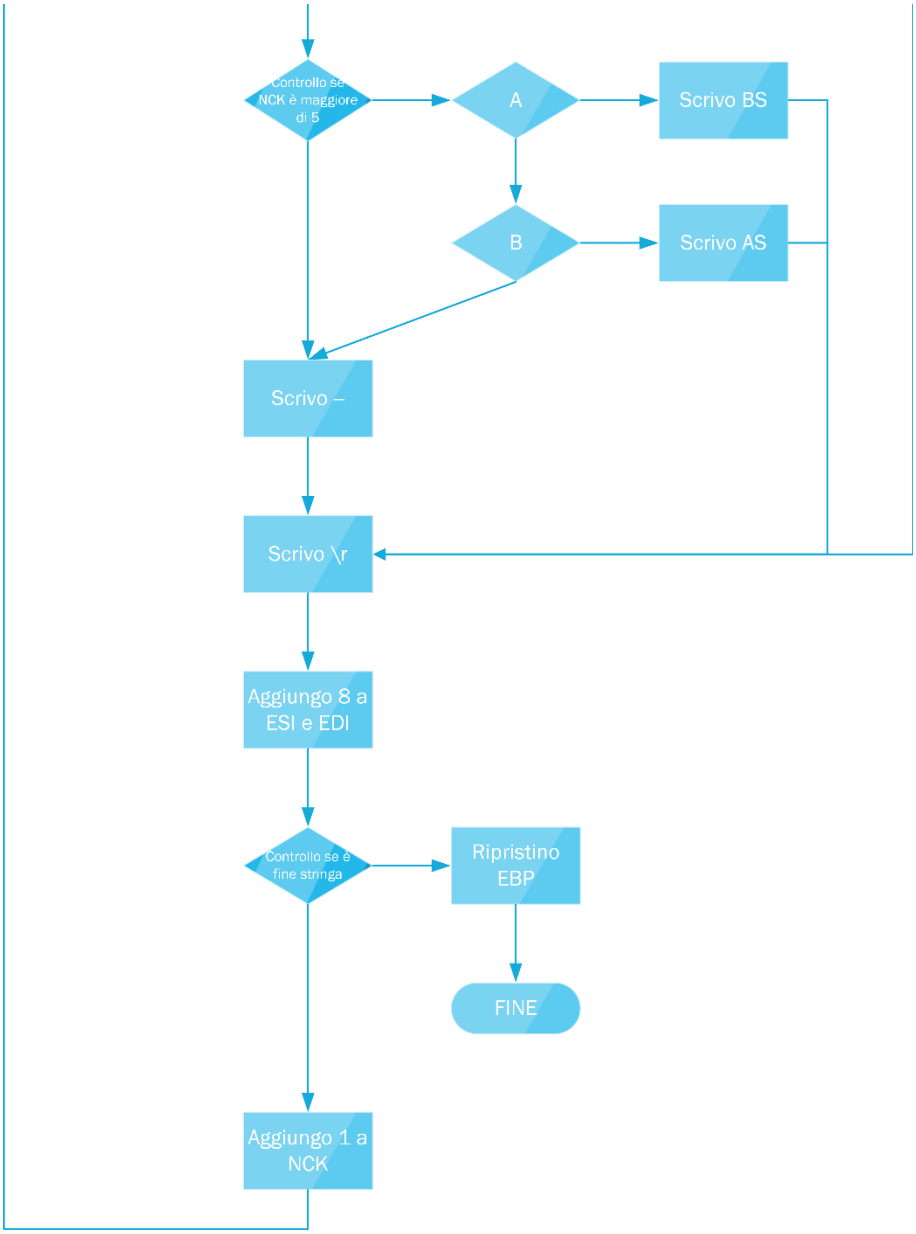
3 Ipotesi aggiuntive

Durante lo svolgimento del progetto abbiamo formulato come ipotesi aggiuntiva che l'NCK non supera il 99.

4

Descrizione codice ASM





Abbiamo costruito un diagramma di flusso, flow chart, sul quale poi ci siamo basati per la realizzazione del codice ASM.

Riportiamo in forma più dettagliata alcune componenti:

Ingressi 4.1

Come ingresso abbiamo un array di char contenente gli ingressi per l'esecuzione della macchina chiamato ***bufferin***.

Uscite 4.2

Come uscite abbiamo un array di char contenente i valori risultanti dell'algoritmo chiamato ***bufferout_asm***.

Utilizzo dei registri 4.3

Nel codice ASM sono stati utilizzati i seguenti registri per i seguenti scopi:

- **EAX, EBX, EDX**
  Registri sono stati utilizzati come registri temporanei;

- **ECX**
  Registro contenente l'NCK attuale;

- **EDI**
  Registro usato come puntatore per il ***bufferout_asm;***

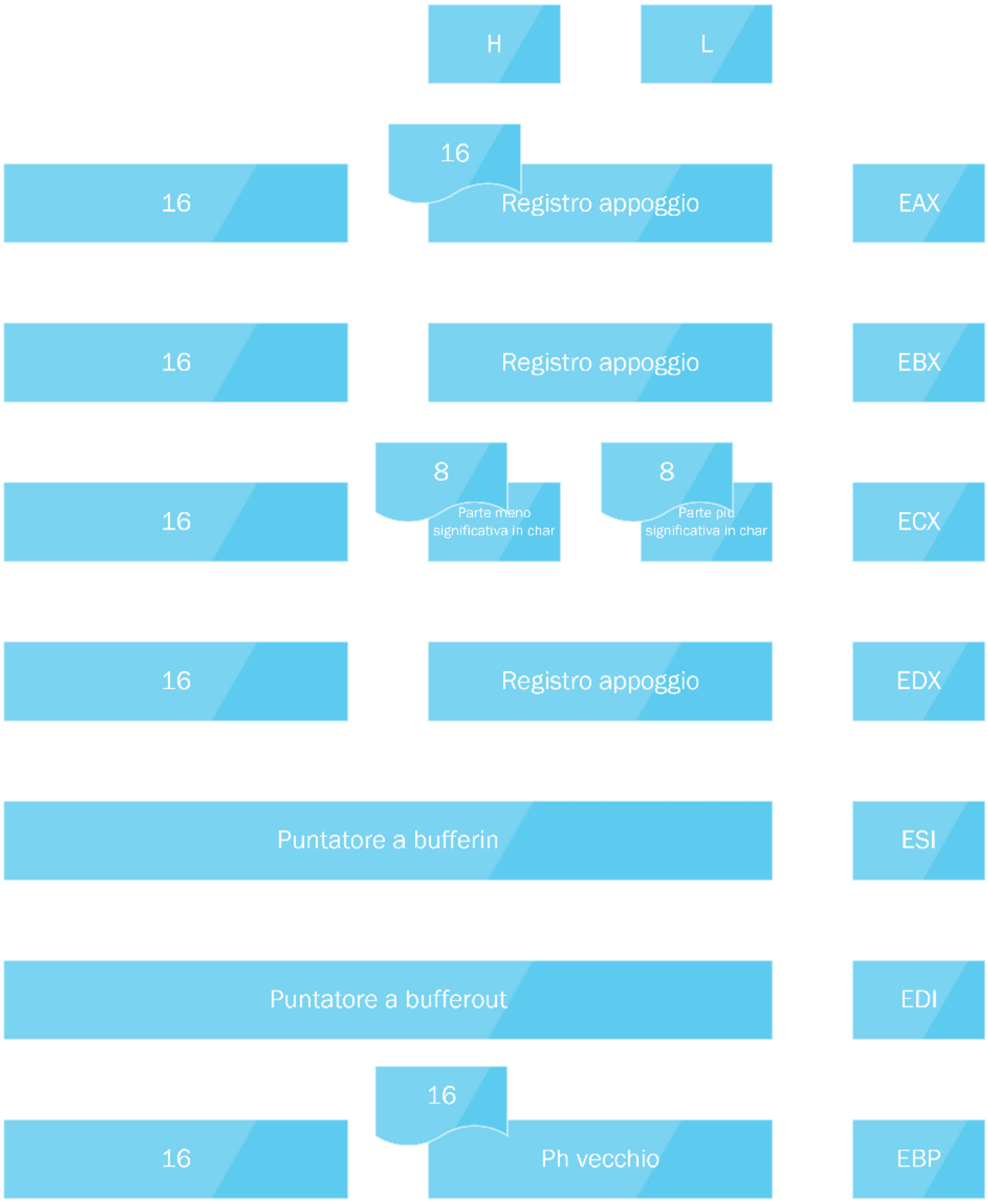
- **ESI**
  Registro usato come puntatore del ***bufferin;***

- **EBP**
  Registro usato come "contenitore" per il Ph precedente.

  Nota: Il valore di EBP viene salvato prima dell'esecuzione del codice ASM. Effettuata l'esecuzione del codice viene ripristinato al fine di evitare problemi.

Schema utilizzo registri 4.3.1

Di seguito riportiamo uno schema da noi implementato sul quale ci simo basati per l'utilizzo dei registri nel codice:



5

Scelte progettuali/Ottimizzazione

Nella realizzazione finale della macchina abbiamo fatto delle scelte con l'obiettivo di ottenere un risultato migliore e un codice più efficiente. Sono state eseguite, dunque, le seguenti ottimizzazioni per massimizzare le prestazioni del codice assembly:

- Abbiamo usato solo i registri del processore per contenere valori temporanei, quindi non abbiamo ricorso all'utilizzo della memoria poiché gli accessi alla RAM richiedono più tempo.

- Abbiamo fatto utilizzo dell'intera lunghezza del bus (32 bit) al fine di trasferire informazioni in meno tempo evitando cioè scrittura e lettura di byte.

- Abbiamo preferito "risparmiare" tempo anche per la conversione del numero a tre cifre evitando di usare calcoli come SUB e MUL, che normalmente richiedono più tempo per essere eseguite dal processore, e confrontando direttamente i valori in memoria.

- Abbiamo inoltre scelto di utilizzare l'istruzione test, che effettua l'AND tra due parametri, invece dell'istruzione compare (cmp), che effettua la sottrazione, poiché impiega meno tempo per l'esecuzione.

- Per azzerare i registri abbiamo usato il xor invece di usare la mov $0, %registro che risulta più lenta.

- Al fine di evitare l'utilizzo dei jump (jmp), i quali richiedono maggiore tempo di esecuzione, abbiamo fatto utilizzo di cmovcc; istruzione che permette di spostare un registro se la condizione (cc) è vera.

6
Velocità rispetto a C

Il nostro codice risulta all'incirca 31 volte più veloce rispetto al codice C.

```
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 11898 ns
ASM time elapsed: 635 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 19135 ns
ASM time elapsed: 662 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 17408 ns
ASM time elapsed: 657 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 15983 ns
ASM time elapsed: 742 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 11785 ns
ASM time elapsed: 873 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 13714 ns
ASM time elapsed: 509 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 14122 ns
ASM time elapsed: 497 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 12496 ns
ASM time elapsed: 798 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 11445 ns
ASM time elapsed: 656 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 14310 ns
ASM time elapsed: 571 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 18822 ns
ASM time elapsed: 878 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 17821 ns
ASM time elapsed: 594 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 19101 ns
ASM time elapsed: 820 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 17838 ns
ASM time elapsed: 613 ns
ubuntu@ubuntu:~$ ./controller testin.txt testout.txt
C time elapsed: 11606 ns
ASM time elapsed: 515 ns
ubuntu@ubuntu:~$
```