UNIVERSITA’ DEGLI STUDI DI VERONA

ELABORATO ASM

ARCHITETTURA DEGLI ELABORATORI

TONINI FRANCESCO (VR408686)

CAPOGROSSO LUIGI (VR408776)

DESCRIZIONE DEL PROGETTO

Riprendendo la situazione presentata nello scorso elaborato, si deve ottimizzare un codice in linguaggio C che effettua il monitoraggio di un impianto chimico industriale mediante l’uso di Assembly.

Il programma deve essere lanciato dalla linea di comando rispettando la seguente sintassi:

$ ./controller testin.txt testout.txt

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

* 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 -.
* RESET [1]: quando posto a 1 il controllore deve essere resettato, ovvero tutte le uscite devono essere poste a - 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 [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 considerano i seguenti valori di soglia:

* pH < 6.0: Acido
* 6.0 <= pH <= 8.0: Neutro
* pH >= 80: Basico

STRUTTURA E FUNZIONI CODICE ASSEMBLY

La funzione che è stata implementata per il miglioramento del codice in linguaggio C è controllore.

Tale funzione riceve in ingresso due array di caratteri, bufferin e bufferout\_asm, e viene richiamata all’interno del file controller.c come un extern module, con la seguente sintassi:

extern void controllore(char in[], char out[])

Questa, può essere definita come il “main” del nostro programma.

Il file assembly “ph.s” è stato strutturato come segue:

* controllore: prepara l’ambiente e salva lo stato attuale dei registri perché vengano ripristinati a fine esecuzione. Controlla che il file di input non sia vuoto e salta alla funzione start;
* start: verifica che all’istante attuale la macchina non sia spenta o in stato di reset. Se così fosse il programma salta alla funzione printReset, altrimenti procede con l’analisi del ph. Per l’analisi del PH è stato adottato un particolare controllo. Nello specifico, siano ABC le tre cifre rappresentative del pH
  + Se A >= 1, passa a checkBasic
  + Se B < 6, passa a checkAcid
  + Se B <= 7 (e maggiore di 6 – grazie al controllo precedente), passa a checkNeutral
  + Se B + C = 8, passa a checkNeutral, altrimenti passa a checkBasic;
* checkBasic: elabora pH basico;
* checkAcid: elabora pH acido;
* checkNeutral: elabora pH neutro;
* end: verifica se abbiamo raggiunto la fine del file e incrementa gli indirizzi. Se abbiamo raggiunto la fine del file, ripristina i registri e termina l’esecuzione, altrimenti incrementa i registri e richiama start;
* printClear: vengono modificati nck e vlv a seguito di una variazione di pH. Al termine passa a end;
* printValves: Stampa vlv. Al termine passa a end;
* printStatus: si occupa della stampa su bufferout di st, nck e vlv. Al termine passa a end;
* printReset: si occupa della stampa su bufferout dello stato di reset della macchina. Al termine passa a end.

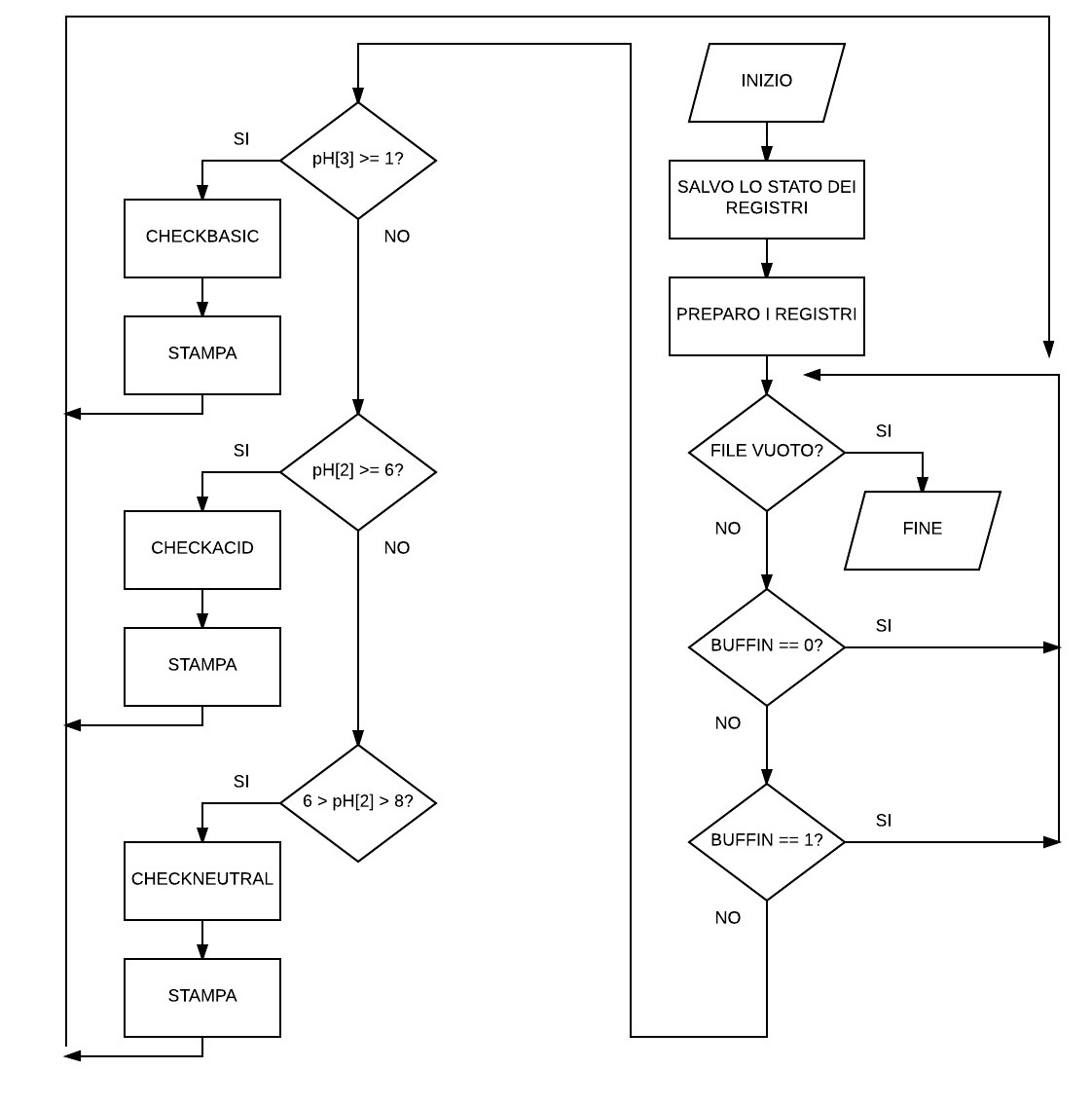
Non si è ritenuto necessario utilizzare variabili, al suo posto si è preferito usare alcuni registri general-purpose per salvare lo stato precedente della macchina. Nello specifico:

* ESI: contiene l’indirizzo di bufferin;
* EDI: contiene l’indirizzo di bufferout\_asm;
* EAX: contiene i valori per la conversione da numero a caratteri ascii (usato per convertire nck in ascii);
* BL: contiene lo stato della macchina (acido, basico, neutro) all’istante precedente, viene comparato con lo stato attuale;
* BH: contiene lo stato della valvola all’istante attuale. In base al suo valore veniva scelto se stampare S o – come secondo carattere di output;
* CL: contava i numeri di cicli nel quale la macchina si trovava nella sostanza attuale.

DESCRIZIONE DEL FLUSSO DEL PROGRAMMA

Il programma è suddiviso in più fasi.

Esse sono qui di seguito prima illustrate attraverso il diagramma di flusso, e poi seguentemente spiegate nello specifico.



In primo luogo salviamo lo stato attuale dei registri che verranno utilizzati durante l’esecuzione del programma per poi, al termine di questo, ripristinarli.

Si passa poi alla preparazione di tali registri secondo gli utilizzi spiegati precedentemente.

Arriviamo ora alla prima condizione in modo tale da verificare se il file è vuoto o meno. Se esso risulta essere vuoto, l’esecuzione del mio programma termina, altrimenti salto alla funzione di elaborazione (start).

Proseguiamo con la seconda condizione.

Compariamo $48 con il primo byte di ESI (cioè init), se questo è 0 (48 in ascii) la macchina è in stato di reset e l’output va azzerato. Si prosegue quindi con la stampa dell’output “vuoto” (printReset), che consta in ordine di:

ST, “divisore”, NCK, “divisore”, VLV

Saltiamo infine alla funzione di termine la quale verifica se vi sono o meno altre righe da analizzare.

Terza condizione.

Consta nella comparazione fra il numero $49 (ovvero 1 in ascii) con il terzo byte di ESI, se questo è zero la macchina è in stato di reset e l'output va azzerato. Anche qui, come in precedenza si proseguirà con la stampa dell’output “vuoto”.

Se tali condizioni non sono soddisfatte vuol dire che vi è una sostanza da monitorare e la macchina è accesa.

Recuperiamo quindi la terza cifra del pH. Se tale cifra è uguale a 1 la sostanza è basica.

Saltiamo quindi all’elaborazione di tale sostanza (check\_basic). Anzitutto compariamo lo stato precedente con lo stato attuale, se è variato pulisco nck/vlv e ricomincio, altrimenti procedo col calcolo di nck e delle valvole.

La funzione di “pulizia” e stampa di nck e vlv (printClear) di seguito ad una variazione di pH consta nell’azzerare nck e vlv, stampe poi nell’ordine seguente

ST, “divisore”, NCK, “divisore”, VLV

Saltiamo infine alla funzione di termine la quale verifica se vi sono o meno altre righe da analizzare.

Procedere con il calcolo di nck e delle valvole consiste invece nel verificare se ho superato i 5 cicli di clock (vedi specifiche) e se non li ho superati, stampo. Se li ho invece superati modifico le valvole e stampo (printStatus).

La stampa su bufferout, a sua volta, nello specifico compie:

ST, “divisore”, NCK (in ascii), divisore, VLV

Saltiamo infine alla funzione di termine la quale verifica se vi sono o meno altre righe da analizzare.

Se la terza cifra di pH non dovesse essere però essere uguale a 1, si recupera la seconda. Se la seconda cifra è < 6, la soluzione è acida.

Saltiamo quindi all’elaborazione della sostanza acida (check\_acid).

Qui viene ripetuto lo stesso procedimento spiegato precedentemente per lo stato basico. Precisiamo qui invece che per la stampa delle valvole è stata implementata un ulteriore etichetta (print\_valves).

Si passa ora alla somma della prima e seconda cifra del pH. Se è maggiore di 8 e minore di 6 allora la sostanza sarà neutrale.

Elaboriamo perciò il pH neutro. Comparo lo stato precedente con l’attuale, aggiorno lo stato, se lo stato è variato pulisco nck/vlv e ricomincio, altrimenti procedo con il calcolo di nck reimposto le valvole e stampo.

SCELTE PROGETTUALI

* Si suppone che il range di valori in ingresso per il pH possa variare da 0 a 140 incluso;
* la macchina può accettare fino a 400 righe di input (come da specifiche controller.c)
* la macchina non ammette righe con più di 8 caratteri (incluso carattere di carry-in) per riga e che l’ultima riga risulti vuota