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 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 a `start`;
- `start`: verifica che all'istante attuale la macchina non sia spenta o in stato di reset. Se così fosse il programma salta a `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 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, stampa ed apre valvole (se necessario);
- `checkAcid`: elabora pH acido, stampa ed apre valvole (se necessario);
- `checkNeutral`: elabora pH neutro, stampa ed apre valvole (se necessario);
- `printClear`: vengono azzerati nck e vlv a seguito di una variazione di pH. Al termine passa a `end`;
- `printValves`: stampa valvole chiuse. Al termine passa a `end`;
- `printReset`: si occupa della stampa dello stato di reset della macchina. Al termine passa a `end`;

- `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`.

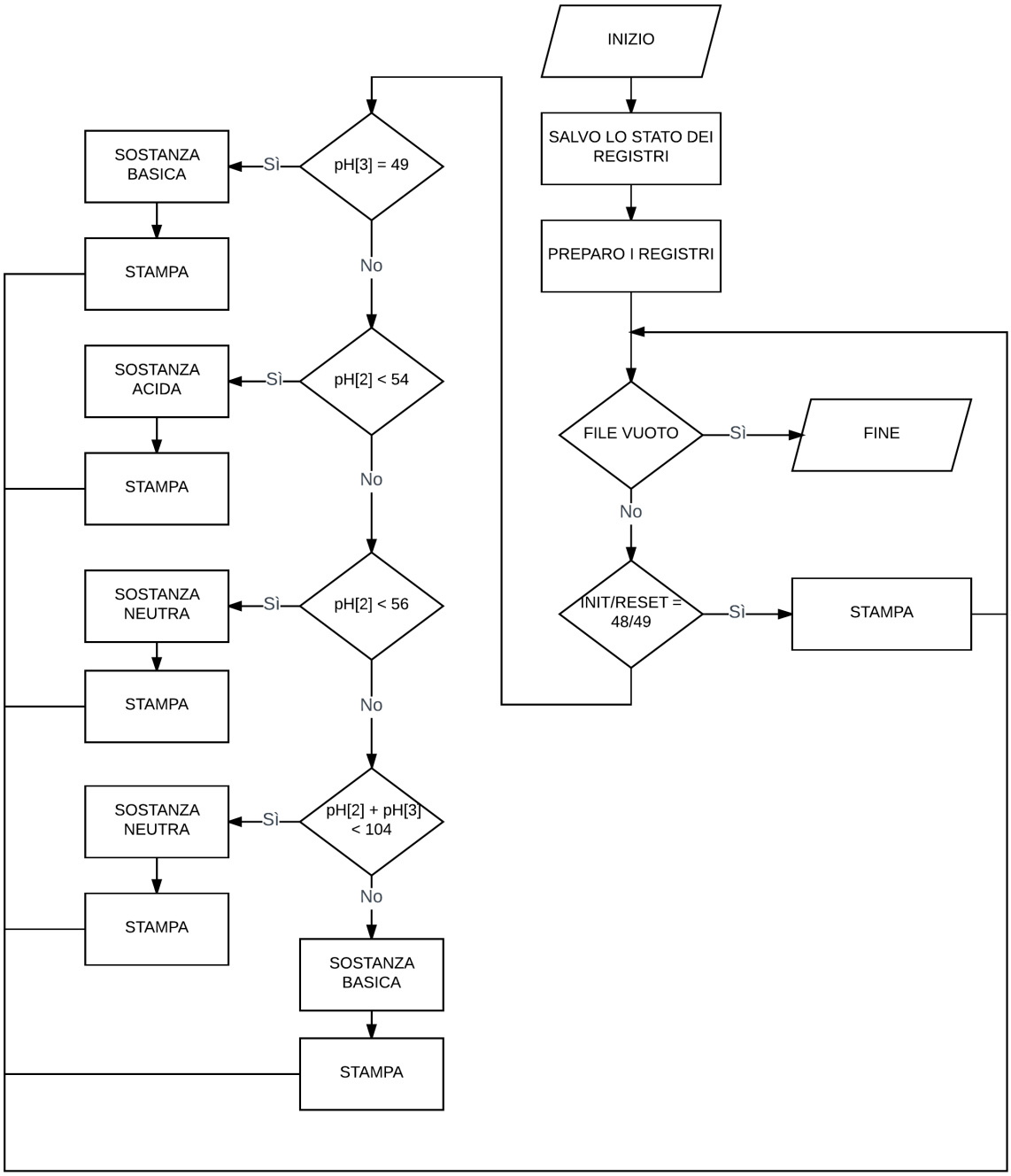
Non si è ritenuto necessario utilizzare variabili, al suo posto si è preferito usare alcuni registri general-purpose per salvare lo stato precedente della macchina. Nello specifico:

- ESI: contiene l'indirizzo di `bufferin`;
- EDI: contiene l'indirizzo di `bufferout_asm`;
- EAX: necessario 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 viene scelto se stampare S o – come secondo carattere di output;
- CL: conta 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.



Prima di procedere con l'elaborazione del file, si salva lo stato attuale dei registri e si controlla se il file stesso è vuoto. Se il file risulta vuoto, il programma termina, altrimenti si procede con il controllo dei byte relativi ad init e reset. Tale controllo viene effettuato comparando i primi 4 byte della riga. Se e solo se i primi 4 byte corrispondo a "1,0," allora la macchina è accesa, in caso contrario procedo con l'azzeramento dell'output e passo all'etichetta di termine (`end`).

In base al tipo di sostanza il programma può saltare su tre etichette, una per ogni possibile stato del pH. Al loro interno le tre etichette effettuano gli stessi controlli: si verifica se vi è stata una variazione di pH, si incrementa il contatore e si apre la relativa valvola (se le condizioni lo permettono). Se non vi sono state eccezioni (es. variazione pH), si procede con la stampa. La variazione di pH tra l'esecuzione precedente e l'attuale porta alla chiamata di una etichetta di "semi-reset" (`printClear`), che si occupa di azzerare il contatore e chiudere le valvole (se aperte).

Saltiamo infine all'etichetta di termine la quale verifica se vi sono o meno altre righe da analizzare. Se vi sono altre righe, si salta nuovamente all'inizio (`start`) e si ricomincia, altrimenti ripristino i registri salvati in precedenza e termino l'esecuzione.

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 l'ultima riga deve essere vuota;
- il controllo di init e reset viene effettuato in una singola comparazione verificando se i primi 4 byte della stringa corrispondono all'unica combinazione rappresentante una situazione di macchina accesa.