UNIVERSITÀ DEGLI STUDI DI VERONA

Elaborato ASM

Laboratorio di Architettura degli elaboratori
2012/2013

Bianchi Federico VR 369001

Buro Samuele VR 368823

Descrizione del progetto

Sviluppare un programma che implementi la seguente funzione ricorsiva:

$$A_n = \frac{(n^2 - 2n)A_{n-1} + nA_{n-2} + 4(-1)^{n-1}}{n-2}$$

Questa formula è detta *ricorrenza di Laisant* e risolve il problema chiamato *Married Couple Problem*, rispondendo alla domanda "In quanti modi n coppie di sposi possono sedersi a un tavolo circolare in modo che ci sia sempre un uomo seduto tra due donne e che nessun uomo sieda vicino a sua moglie?".

Nella tabella sono riportati i primi 22 risultati della funzione:

n	A_n
0	1
1	0
2	0
3	1
4	2
5	13
6	80
7	579
8	4738
9	43387
10	439792
11	4890741
12	59216642
13	775596313
14	10927434464
15	164806435783
16	2649391469058
17	45226435601207
18	817056406224416
19	15574618910994665
20	312400218671253762
21	6577618644576902053

Variabili utilizzate

Ogni variabile di tipo stringa è accompagnata da un'altra variabile della forma *variabile_len* che rappresenta la sua lunghezza.

File **main.s**

- **title**: stringa contenente il titolo del programma;
- **intest**: stringa contenente l'intestazione della tabella dei risultati;
- **empty_str**: stringa contenente cinque spazi vuoti per distanziare i valori;
- **empty_short**: uguale a *empty_str*, con uno spazio in meno per i valori di n in doppia cifra;
- **w_v**: (wrong value) stringa che viene stampata di fianco ai risultati di A_n con $n > 12$. Questi risultati sono errati in quanto il numeratore della funzione non è rappresentabile utilizzando registri a 32 bit e senza istruzioni in virgola mobile.

File **atoi.s**:

- **char**: variabile di tipo byte che contiene, uno alla volta, i caratteri digitati in input dall'utente;
- **ctrl**: variabile di tipo byte utilizzata come flag per verificare se l'utente ha inserito la stringa vuota (\n) .

File **itoa.s**:

- **char**: variabile di tipo byte che contiene il carattere da stampare.

File **itoa_not_newline.s**:

Vedi file *itoa.s*.

File **checknumber.s**:

- **err1**: stringa che viene stampata se è stata inserita la stringa vuota;
- **err2**: stringa che viene stampata se non è stato inserito un numero naturale;
- **err3**: stringa che viene stampata se il numero naturale inserito è minore di due.

File **getdata.s**:

- **n1**: stringa che chiede all'utente di inserire il valore iniziale;
- **n2**: stringa che chiede all'utente di inserire il valore finale;
- **err**: stringa che viene stampata se $valore\ iniziale > valore\ finale$.

File **laisant.s**:

La funzione ricorsiva non utilizza variabili ma solo stack e registri.

Modalità di passaggio/restituzione dei valori delle funzioni create

Funzione **atoi.s**:

Legge l'input da tastiera e memorizza il risultato in eax: se il valore letto non è un numero naturale viene salvato -1; se viene inserita la stringa vuota viene salvato -2.
La funzione non modifica gli altri registri.

Funzione **itoa.s** e **itoa_not_newline.s**:

Stampa a video l'intero contenuto nel registro eax.
La funzione modifica soltanto il registro eax.

Funzione **getdata.s**:

Chiede all'utente l'inserimento dei valori richiamando due volte la funzione *checknumber* che ne verifica la correttezza.
Se il $valore\ iniziale \leq valore\ finale$, restituisce i due valori nei registri eax (valore iniziale) ed ebx (valore finale), altrimenti vengono richiesti entrambi.
Anche i registri ecx ed edx vengono modificati.

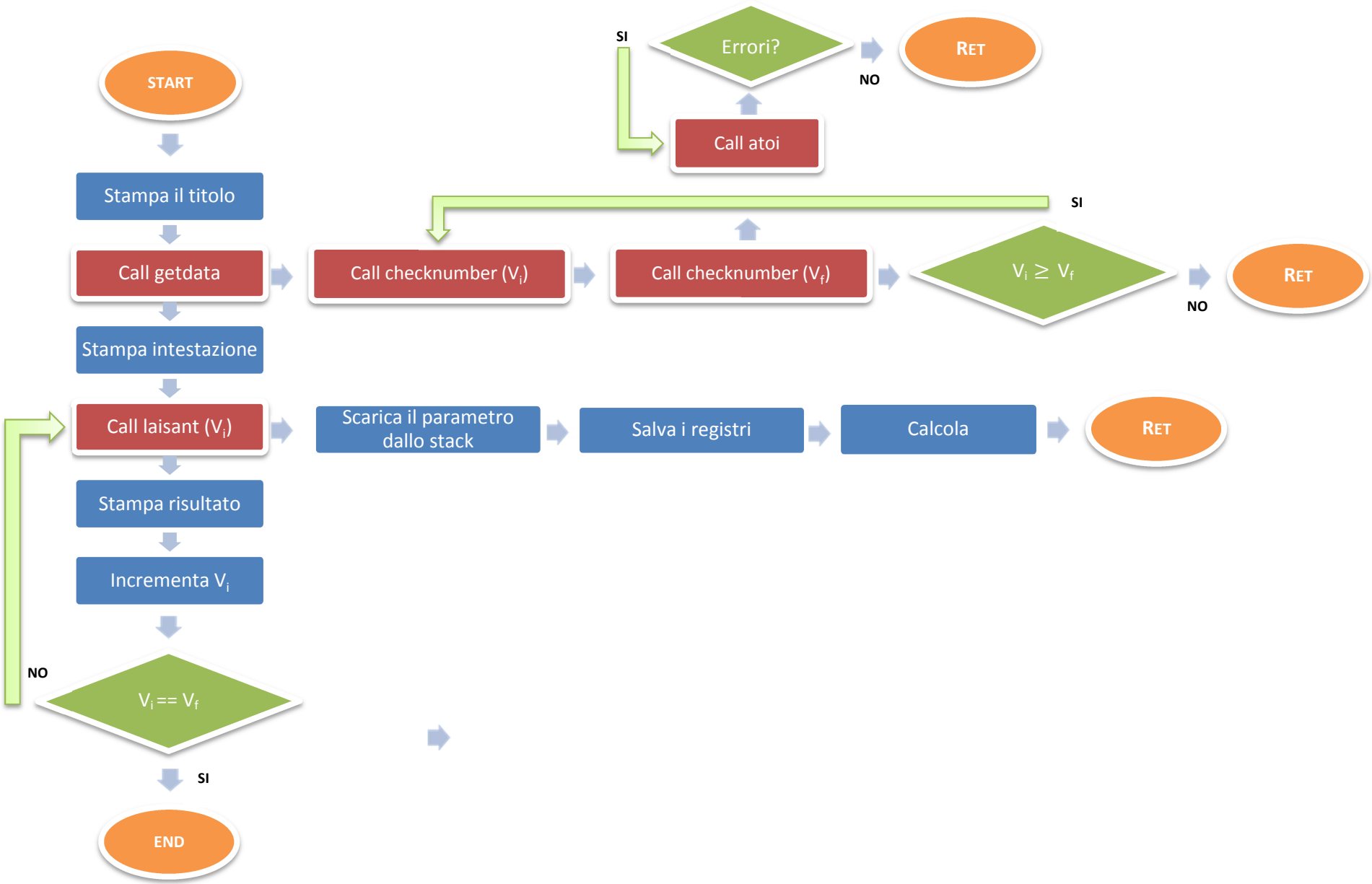
Funzione **checknumber.s**:

Chiama la funzione *atoi* per leggere un intero da tastiera. Se è stata inserita la stringa vuota, se non è stato inserito un numero naturale oppure se il numero inserito è minore di due stampa un preciso messaggio d'errore e richiede nuovamente l'inserimento.

Funzione **laisant.s**:

Scarica dallo stack il parametro su cui deve lavorare e salva i registri. Carica i successivi valori sulla pila per le chiamate ricorsive. Al termine, salva il risultato in edx e ripristina i registri.

L'interazione tra le funzioni e il chiamante principale verrà spiegato più dettagliatamente nella parte che analizza il flusso del programma.


START
Stampa il titolo
Call getdata
Stampa intestazione
Call laisant (V_i)
Stampa risultato
Incrementa V_i
$V_i == V_f$
NO
SI
END
Call checknumber (V_i)
Call checknumber (V_f)
$V_i \geq V_f$
SI
NO
RET
Scarica il parametro dallo stack
Salva i registri
Calcola
RET
Errori?
SI
NO
RET
Call atoi

Descrizione del flusso del programma

Il programma è suddiviso in quattro fasi:

- Input dei valori;
- Controllo dei valori;
- Calcolo della ricorrenza di Laisant;
- Output.

La procedura principale chiama la funzione *getdata* che richiede all'utente l'inserimento di due numeri. Quest'ultima si appoggia alla funzione *checknumber* per richiedere in input i valori (attraverso l'uso di *atoi*) e ne verifica la correttezza. In caso di input non corretti, vengono stampati opportuni messaggi d'errore e viene richiesta nuovamente l'immissione dei parametri.

Dopo la seconda fase, viene effettuato il calcolo tramite la procedura ricorsiva *laisant* per tutti i valori compresi nell'intervallo. I rispettivi risultati vengono stampati ciclicamente in una tabella.

Descrizione delle scelte progettuali

Controllo dei valori immessi

In caso di valori non corretti per l'esecuzione del programma vengono stampati i seguenti messaggi di errore:

- "Input mancante" nel caso venga inserita la stringa vuota;
- "Inserire un numero naturale" nel caso vengano inseriti numeri negativi, decimali o caratteri alfabetici;
- "Il valore inserito deve essere > 1";
- "Il valore iniziale deve essere minore del valore finale".

Avviso di output errato

Per valori maggiori di 12 il numeratore della formula di Laisant non è rappresentabile su 32 bit. Di conseguenza, per questi risultati non corretti, viene stampato un messaggio di errore "wrong value" di fianco all'output.

Gestione dello stack

Il parametro che viene passato alla funzione Laisant viene messo sullo stack. All'inizio della procedura ricorsiva, vengono salvati i registri e viene scaricato il parametro. Per le chiamate successive ricorsive viene utilizzato lo stesso metodo. Al termine di ogni chiamata vengono ripristinati i registri.