Relazione dell’elaborato ASM di Architettura degli Elaboratori

A.A. 2022/2023

Studenti:

Binosi Dimitri VR486399

Bogoni Zeno VR487733

Consegna:

Realizzare un programma Assembly per la gestione del menù cruscotto di un'automobile. Il menù dovrà visualizzare e permettere l'impostazione delle seguenti informazioni: data, ora, impostazione blocco automatico porte, back-home, check olio. Se acceduto in modalità supervisor, il menù dovrà premettere anche l'impostazione di lampeggi frecce modalità autostrada e reset pressione gomme.

Specifiche:

Il programma potrà essere avviato in due modalità:

- utente (lanciando solo il nome dell'eseguibile da riga di comando)
- supervisor (lanciando il nome dell'eseguibile seguito dal codice 2244).

Se avviato in modalità utente, il programma dovrà visualizzare, una riga alla volta, il seguente menù, partendo dalla prima riga e scorrendo sulle altre premendo il tasto ⬇ / ⬆ +invio (inserire ⬇ +invio da messaggio di riga 6 corrisponde alla visualizzazione del messaggio riga 1):

1. Setting automobile:
2. Data: 15/06/2014
3. Ora: 15:32
4. Blocco automatico porte: ON
5. Back-home: ON
6. Check olio

Se avviato in modalità supervisor, il programma dovrà visualizzare la riga 1 nel seguente modo: "Setting automobile (supervisor)", e poter visualizzare anche le seguenti righe:

7. Frecce direzione
8. Reset pressione gomme

Ad ogni voce visualizzata, premendo i tasti ➡+invio si potrà entrare nel sottomenu corrispondente, nel quale verrà visualizzato lo stato attuale del setting e data la possibilità di impostazione.
Ad esempio, una volta visualizzato il menù "Blocco automatico porte: ON", premendo il tasto ➡+invio, si dovrà permettere il cambiamento del setting ON, tramite i tasti ⬇ o ⬆ +invio. In particolare, si dovrà permettere il setting dei sottomenù 4 e 5, con possibilità di impostazione ON/OFF come nell'esempio sopra. I sottomenu 2, 3, 6, non dovranno essere implementati in questo elaborato. All'interno di un sottomenù, il solo inserimento di invio da tastiera corrisponde al ritorno al menù principale.

Se avviato in modalità supervisor, il sottomenù "Frecce direzione" dovrà visualizzare il numero dei lampeggi modalità autostrada (3 per default) con possibilità di variazione (valore minimo 2, valore massimo 5) tramite tastiera (valori inseriti fuori range corrispondono al setting del max/min valore). Nel sottomenù "Reset pressione gomme", inserendo il carattere ➡ seguito da invio, il menù dovrà restituire il
messaggio "Pressione gomme resettata" e tornare al menù principale.

Introduzione

Prima di addentrarci nello sviluppo del programma in assembly abbiamo ragionato in linguaggio a più alto livello, ovvero in C. Facendo ciò abbiamo tracciato le linee guida che avrebbe seguito anche il codice in assembly.
In particolare il metodo di acquisizione dell'input, la suddivisione in funzioni, l'algoritmo di funzionamento delle funzioni cuore del programma, ossia coloro che gestiscono le opzioni 4, 7 e 8. Superato questo ostacolo abbiamo cominciato a lavorare sul codice assembly. Per semplicità abbiamo aggiunto anche delle funzioni per stampare il testo, "pulire" eventuali input eccessivi o errati, e stampare tabulazioni o new line (tutto questo verrà approfondito più avanti).

Cominciamo ora l'analisi del programma (C e Assembly).

Le variabili (C)

Variabili del Main

code: variabile di tipo intero, presente nel main e passata come parametro alle altre funzioni, usata per immagazzinare il codice passato come parametro al momento dell'avvio. Se questo codice è 2244 il programma verrà avviato in modalità Supervisor, permettendo quindi di accedere alle funzioni 7 e 8.

num_lamp: variabile di tipo puntatore ad intero, presente nel main e passata come parametro alla funzione 7 e 8, usata per immagazzinare il numero di lampeggi delle frecce dell'auto. Il numero massimo è 5 e il minimo 2. Di default è settato a 3.

blocco[ ]: variabile di tipo puntatore ad intero, presente nel main e passata alle altre funzioni (4, 5, 6, 7, 8) sotto forma di parametro. Essa contiene due celle, in cui sono immagazzinati due valori, il primo rappresenta se il blocco automatico delle porte è OFF (0) o ON (1), di default è ON. Il secondo valore svolge una funzione uguale, ma per il back-home, anch'esso settato a off di default.

Variabili delle funzioni

ch: variabile di tipo char, presente dentro ogni funzione usata per eliminare eventuali caratteri di eccesso inseriti dall'utente. Il modo di funzionamento sarà specificato in seguito.

scelta[ ]: variabile di tipo puntatore a char, lunghezza SIZE, ossia 4. Al suo interno viene immagazzinata la scelta fatta dall'utente, ossia se viene selezionata una delle seguenti frecce: ⬆, ⬇ o ➡ (⬅ non è mai un'opzione valida).

scelta2[ ]: analoga alla precedente, ma riguardante la seconda scelta che in alcune funzioni l'utente deve compiere. Ad esempio nella funzione 4, dopo aver deciso se entrare nel sottomenù o meno (scelta), l'utente dovrà decidere se mettere blocco porte a ON o OFF (scelta2). Si utilizzano due variabili diverse per evitare di sovrascrivere la precedente scelta e quindi capire se rieseguire la funzione o meno.

num_car: variabile di tipo intero, presente nella funzione 7, adibita a contenere il numero inserito dall'utente quando viene richiesto di impostare la nuova velocità di lampeggio delle frecce.

Le variabili (Assembly)

Variabili del Main

code: variabile di tipo long atta a memorizzare il codice passato come parametro al momento dell'attivazione del programma.

blocco_porte: variabile di tipo long atta a memorizzare lo stato del ***blocco_porte***, se OFF (1), se ON (0). Di default è impostata su OFF.

back_home: variabile di tipo long atta a memorizzare lo stato del ***back_home***, se OFF (1), se ON (0). Di default è impostata su OFF.

num_lamp: variabile di tipo long atta a memorizzare il numero di lampeggi, questa può assumere valori tra 2 e 5 (estremi compresi). Il valore di default è 3.

Variabili delle funzioni

scelta_freccia: variabile di tipo ascii, presente in tutte le funzioni riguardanti le opzioni del menù. Come in C per la variabile ***scelta[ ]***, la sua funzione è quella di immagazzinare la scelta dell'utente. Esiste anche la sua complementare ***scelta_freccia_2,*** il cui scopo è analogo a quello precedentemente descritto di ***scelta2[ ]*** nel programma C.

code: variabile di tipo long, utilizzata per importare all'interno delle funzioni lo stato della variabile ***code*** presente nel main, questa permette a tutte le funzioni di sapere se il programma è stato eseguito in Supervisor mode.

blocco_porte: variabile di tipo long, utilizzata per importare all'interno delle funzioni lo stato della variabile ***blocco_porte*** presente nel main.

back_home: variabile di tipo long, utilizzata per importare all'interno delle funzioni lo stato della variabile ***back_home*** presente nel main.

num_lamp: variabile di tipo long, utilizzata per importare all'interno delle funzioni lo stato della variabile ***num_lamp*** presente nel main. Prima di modificare questa variabile si esegue un controllo sul valore inserito per assicurarsi che questo faccia parte dell'intervallo corretto, ovvero [2,5].

Testo: questa è una variabile di tipo ascii, il nome della variabile è solo indicativo, poiché nel codice compare con nomi diversi, tuttavia la sua funzione è sempre la stessa, ossia contenere una stringa di testo che verrà poi stampata a schermo. Compare anche la variabile ***testo_lung*** di tipo long, atta ad indicare la lunghezza della stringa inserita dall'utente.

num_lamp_str: variabile di tipo ascii, atta a contenere il nuovo numero di lampeggi (sotto forma di stringa) inserito dall'utente.

numero: variabile di tipo long, contiene il numero di lampeggi inseriti dall'utente nella funzione 7. Da notare che l'input dell'utente non viene subito immagazzinato in questa variabile, bensì prima in ***num_lamp_str*** e poi, una volta convertito ad intero, memorizzato qua.

counter: contatore utilizzato nelle funzioni 5, 7 e 8.

Spiegazione Codice (C)

Nel main vengono solo chiamate le funzioni corrispondenti alle diverse opzioni del menu, una dopo l'altra. Le funzioni basilari (ossia quelle che devono solo stampare a video un testo) sono strutturate come la seguente (con piccole modifiche):

```
do {
	printf("Testo");
	fgets(scelta, SIZE, stdin);
	while((ch = getchar())!='\n');

	if(scelta[0] == 27 && scelta[1] == 91 && scelta[2] == 65){
	opzione_precedente(code);
	}
} while(scelta[0] != 27 || scelta[1] != 91 || scelta[2] != 66);
```

Questo schema viene usato nelle funzioni 1, 2, 3, 6. I cambiamenti che devono essere apportati sono la modifica del testo da stampare e la funzione precedente che viene chiamata. Nella funzione 1 è anche presente un ulteriore controllo su ***code*** per vedere se è 2244 e nel caso, stampare "Setting automobile (supervisor)".

Ci sono tre passi importanti su cui soffermarsi, il primo è la riga:

```
fgets(scelta, SIZE, stdin);
```

qui infatti si usa la funzione `fgets()` per acquisire da tastiera (stdin) e copiare in scelta solo i primi (SIZE) caratteri inseriti dall'utente. SIZE ricordiamo essere una costante settata a 4.

Il secondo punto importante è questa riga:

```
while((ch = getchar()) != '\n');
```

qui, vengono acquisiti tutti gli eventuali caratteri diversi da `'\n'`, inseriti dopo la scelta della freccia eseguita dall'utente, questa riga, in codice assembly è rispecchiata dalla funzione "pulizia.s".

Il terzo ed ultimo punto, è il seguente:

```
if(scelta[0] == 27 && scelta[1] == 91 && scelta[2] == 65){};
```

Con questa riga vogliamo sottolineare come è stata eseguita la scomposizione della scelta dell'utente e la sua analisi. Vediamo infatti che ⬆ è stata convertita tramite i caratteri *^[[A*, ossia *ESCAPE*, *[* e *A*, che convertiti usando la tabella ascii corrisponderebbero ai numeri 27, 91, 65. Per le restanti frecce ⬇ e ➡, l'unica differenza è la lettera finale, che sarà, B (ossia 67) per ⬇ e C (ossia 66) per ➡.

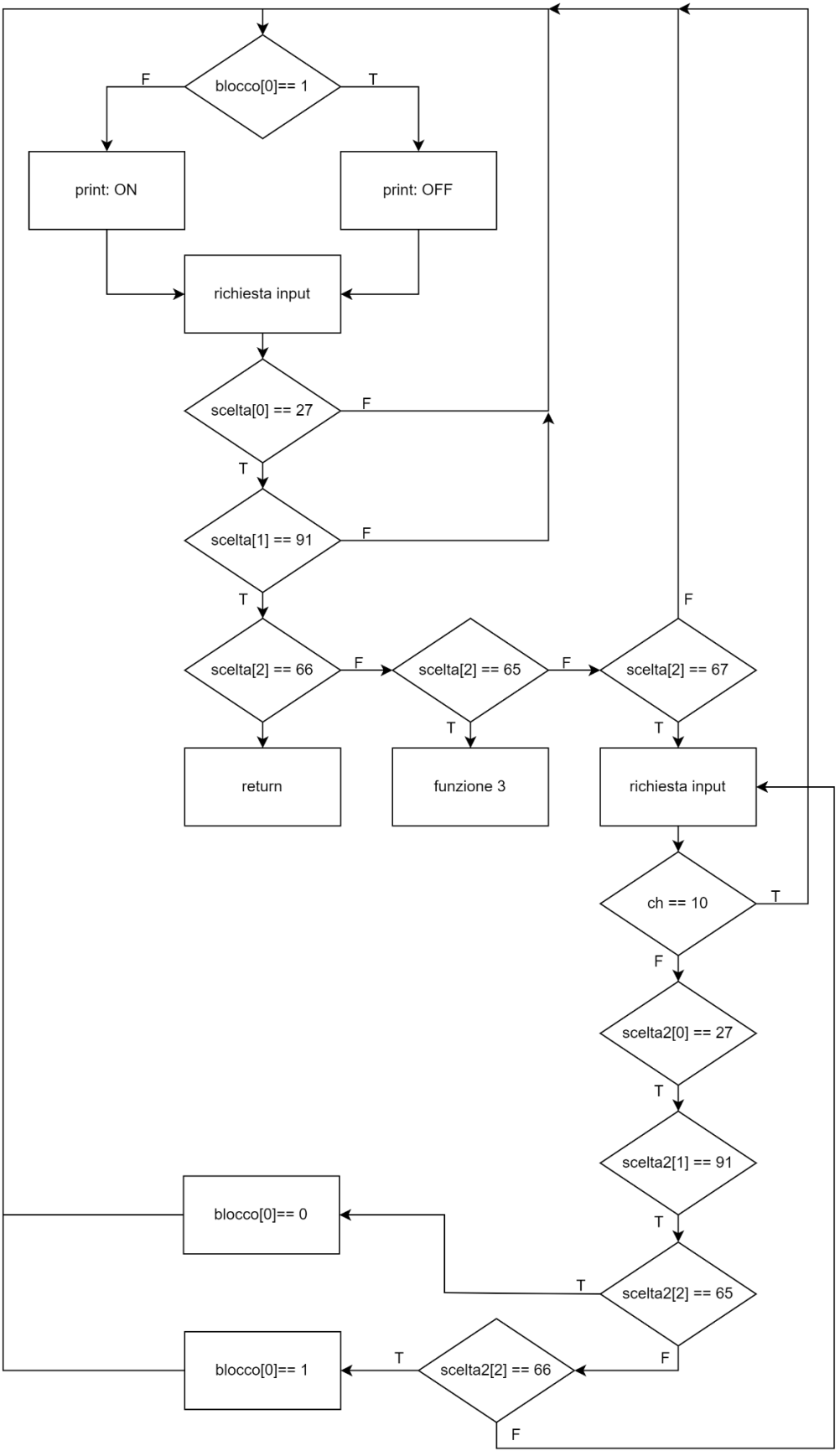
Seguendo le direttive del progetto se l'utente ha inserito ⬆ si tornerà all'opzione precedente del menù, se invece ha inserito ⬇ si procederà chiudendo la funzione attuale, tornando al main e chiamando la funzione successiva. Nel caso di inserimento di altre combinazioni di caratteri, le funzioni (1, 2, 3, 6) stamperanno di nuovo la richiesta di inserimento di un input.

Il metodo adottato per importare nelle diverse funzioni i valori delle variabili del main del programma è un basilare passaggio per parametro. Le variabili che devono essere modificate sono ***blocco[ ]*** e ***num_lamp***. Per quanto riguarda la prima, non si pongono problemi dato che si tratta di un array, quindi modificabile direttamente dalle funzioni. Per quanto riguarda ***num_lamp*** invece, il nuovo valore dopo essere stato controllato viene ritornato dalla funzione.

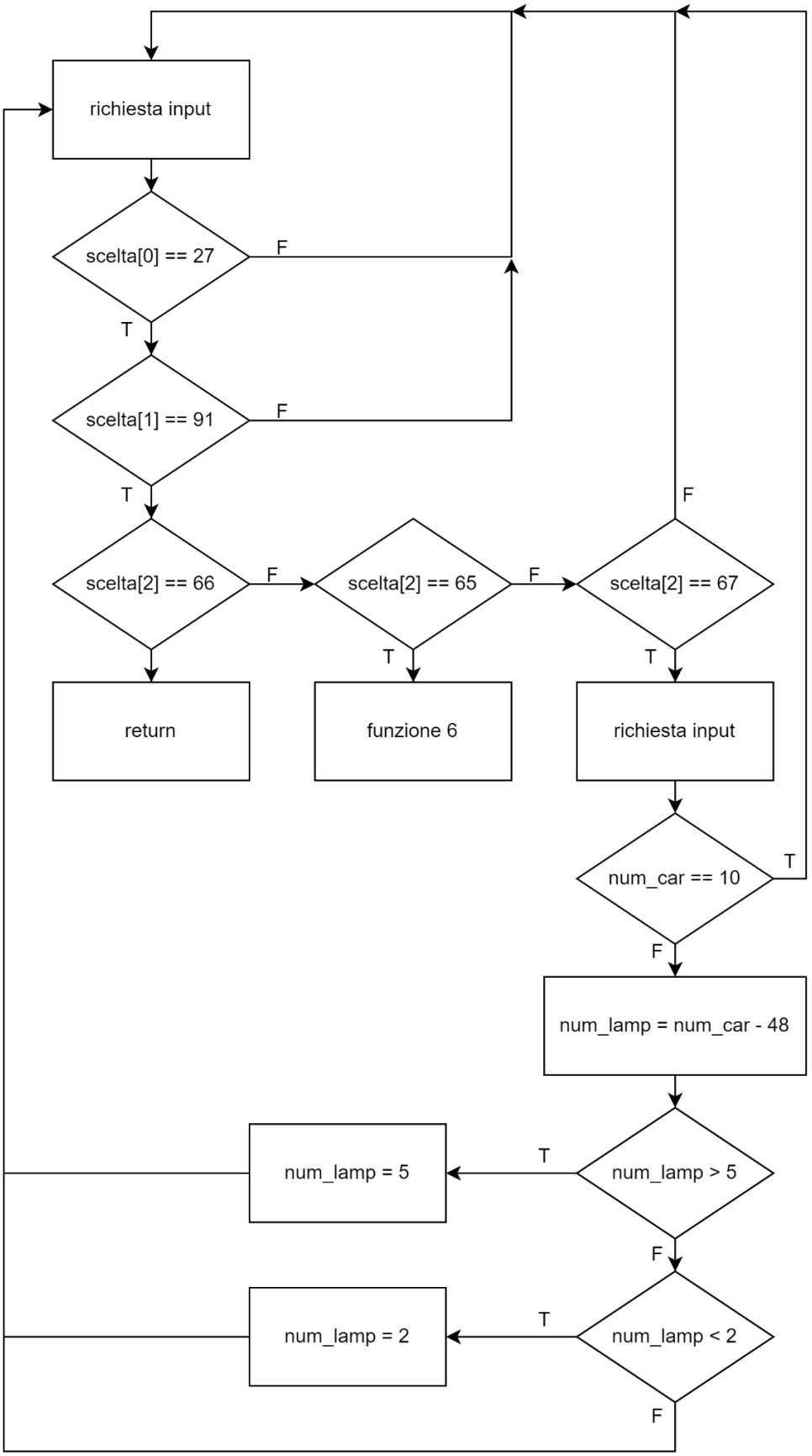
Passiamo ora alle funzioni 4, 7, 8 che sono le più complesse nel programma generale (la funzione 5 è analoga alla 4 con qualche modifica).

Per trattare queste funzioni ci serviremo dei diagrammi di flusso riportati nelle prossime pagine.

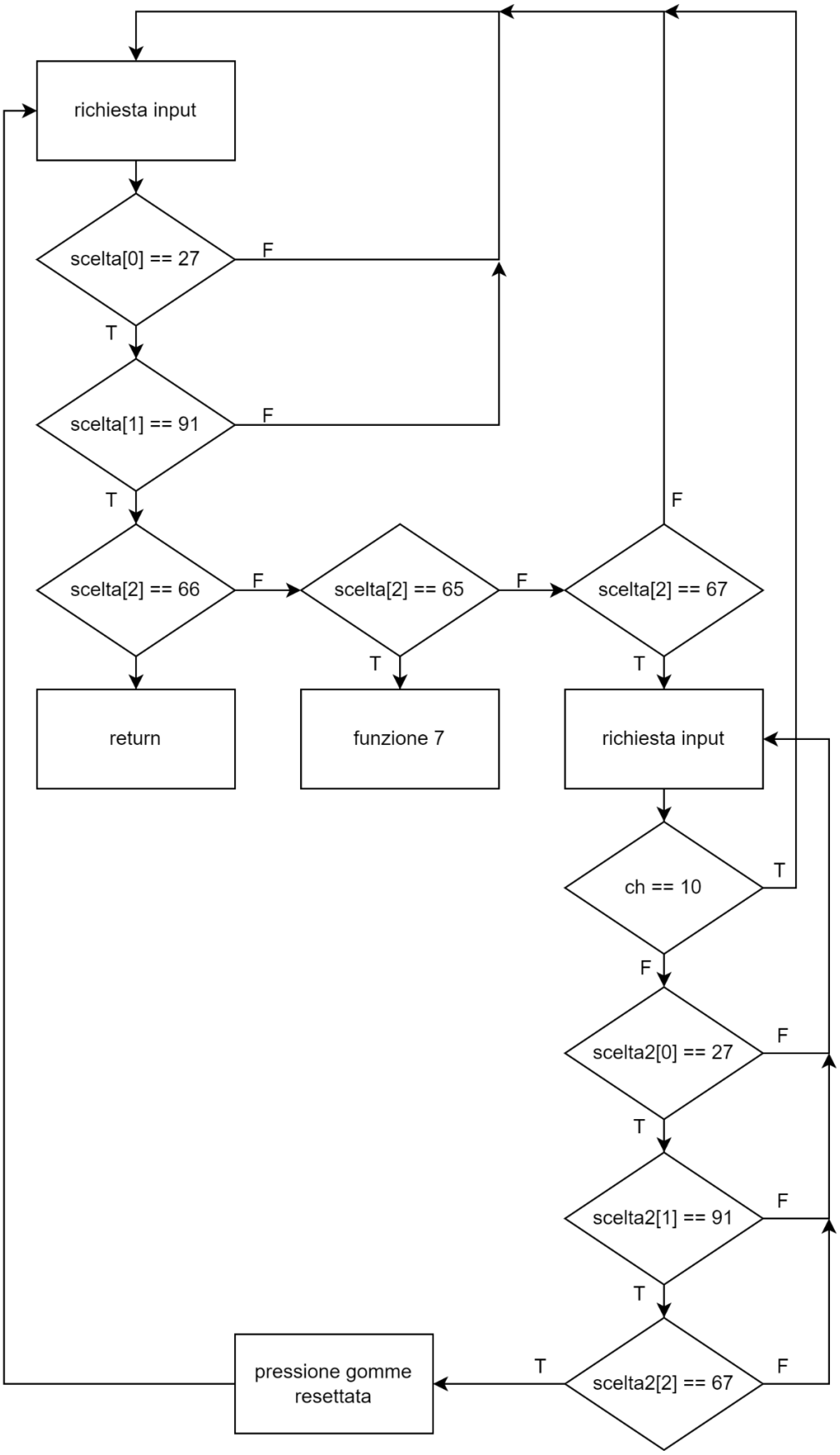
FUNZIONE 4


blocco[0]== 1
F
T
print: ON
print: OFF
richiesta input
scelta[0] == 27
F
T
scelta[1] == 91
F
T
F
scelta[2] == 66
F
scelta[2] == 65
F
scelta[2] == 67
T
T
return
funzione 3
richiesta input
ch == 10
T
F
scelta2[0] == 27
T
scelta2[1] == 91
T
blocco[0]== 0
T
scelta2[2] == 65
F
blocco[0]== 1
T
scelta2[2] == 66
F

FUNZIONE 7


richiesta input
scelta[0] == 27
F
T
scelta[1] == 91
F
T
F
scelta[2] == 66
F
scelta[2] == 65
F
scelta[2] == 67
T
T
return
funzione 6
richiesta input
num_car == 10
T
F
num_lamp = num_car - 48
num_lamp = 5
T
num_lamp > 5
F
num_lamp = 2
T
num_lamp < 2
F

FUNZIONE 8


richiesta input
scelta[0] == 27
F
T
scelta[1] == 91
F
T
scelta[2] == 66
F
scelta[2] == 65
F
scelta[2] == 67
F
T
T
return
funzione 7
richiesta input
ch == 10
T
F
scelta2[0] == 27
F
T
scelta2[1] == 91
F
T
scelta2[2] == 67
F
T
pressione gomme resettata

Codice (Assembly)

I passi importanti da sottolineare in questo caso sono due:

1) Il passaggio delle variabili ***blocco_porte***, ***code***, ***back_home*** e ***num_lamp*** non avviene tramite parametri come in C, ma tramite indirizzo, ossia i vari indirizzi delle variabili vengono spostati nei registri EAX, EBX, ECX, EDX e poi una volta nella sottofunzione all'interno di variabili. In questo modo è possibile modificare direttamente le variabili dalle funzioni.

2) In assembly il compito di eliminare eventuali caratteri aggiunti dopo la scelta dell'utente, verranno eliminati attraverso la funzione Pulizia.s, la quale cattura e consuma un carattere per volta, poiché come si può vedere qui sotto, "forza" l'acquisizione di un solo carattere, per poi consumarlo e ripetere il loop finché non trova `'\n'`.

```
Pulizia:
  movl $3, %eax
  movl $1, %ebx
  leal car, %ecx
  movl $1, %edx
  int $0x80

  leal car, %esi
  movb (%esi), %bl
  cmpb $10, %bl
jne pulizia
```

Scelte progettuali

Vasto utilizzo di funzioni

Abbiamo deciso di servirci di funzioni anche per operazioni basilari come l'inserimento di tabulazioni, in modo da mantenere pulito il codice ed aumentare la leggibilità, inoltre questo è stato utile anche a noi per evitare numerose ripetizioni.

Gestione del loop menu

Data la struttura delle funzioni del nostro progetto abbiamo deciso di non permettere il loop da funzione 1 a 8/6 bensì solo quello da funzione 8/6 a 1 come richiesto da consegna, questo per evitare errori a tempo di esecuzione di *segmentation fault,* limitando il numero di funzioni che rimangono aperte poiché chiamando la funzione precedente non si ritorna quella attuale.

Gestione ON/OFF

Abbiamo interpretato il cambio tra ON e OFF come un interruttore, ⬆+invio imposta l'interruttore su ON mentre ⬇+invio imposta l'interruttore su OFF, quindi se la funzione si trova già su ON e si preme ⬆+invio l'interruttore resterà ON, lo stesso vale per OFF e ⬇+invio.

Gestione uscita dai sottomenù

Abbiamo interpretato la procedura di uscita da un sottomenu nei seguenti modi: è possibile uscire da ogni sottomenu semplicemente premendo invio, oppure inserendo l'input richiesto (numero lampeggi o ⬆/⬇/➡) + invio. Una volta usciti dal sottomenu si verrà riportati all'opzione del menu in cui ci si trovava prima di entrare nel sottomenu.