UNIVERSITÀ DEGLI STUDI DI VERONA

ELABORATO ASM

Architettura degli Elaboratori
Laboratorio

Corso di Laurea in Informatica

A.A. 2022/2023

PROGETTO:
Progettazione di un menu per la gestione del menu cruscotto di un'automobile

Studenti:

Alessia	**Gennari**	VR488137
Mattia	**Pacchin**	VR461870

SOMMARIO

SPECIFICA DEL PROGETTO

Il presente elaborato descrive la progettazione di un codice in Assembly AT&T per la gestione di un cruscotto di un'automobile.

CARATTERISTICHE DEL CODICE

Il menù si può accedere in due modalità: utente e supervisore.

Nella modalità utente è possibile visualizzare il menu composto nel seguente modo:

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

Nella modalità supervisore (si accede aggiungendo al nome dell'eseguibile il codice 2244) il menu visualizzato sarà il seguente:

1. Setting automobile (supervisor):
2. Data: 15/06/2014
3. Ora: 15:32
4. Blocco automatico porte: ON
5. Back-home: ON
6. Check olio
7. Frecce direzione
8. Reset pressione gomme

In entrambe le modalità sarà possibile spostarsi tra i vari menu con le frecce su e giù).
Premendo la freccia a destra sarà possibile modificare i punti 4, 5, 6, 7 e 8.

- *Per i punti 6 e 8 verrà visualizzato un messaggio di conferma.*
- *I punti 4 e 5 potranno essere modificati premendo freccia in su o freccia in giù, questo modificherà il loro valore da ON a OFF e viceversa.*
- *Per modificare il punto 7 si richiede di inserire un numero, di default è 3 il massimo è 5 ed il minimo 2, se verrà inserito un numero maggiore di cinque verrà salvato cinque, se invece viene inserito un numero minore di due verrà salvato due.*

SUDDIVISIONE DEI FILE

Il codice assembly è diviso in vari file, uno per ogni funzione.

main.s : contiene le variabili globali, controlla se è stata attivata la modalità supervisore, chiama una prima volta la funzione index_position_message (per visualizzare il primo messaggio) e poi all'interno di un ciclo while richiama la funzione navigate_menu.

move.s : è la funzione che si occupa di prendere in input le frecce e restituire -1 in caso di freccia in su, 1 per freccia in giù e 2 per freccia a destra.

set_blinkers.s : prende in input il numero desiderato di blinks, controlla se è nel range corretto (da 2 a 5) se è maggiore di cinque verrà impostato a cinque se è minore di due verrà impostato a due.

index_position_message.s : uno switch case che a seconda della posizione dell'index visualizza l'impostazione corretta del menù. Nel caso dei punti modificabili viene effettuato il controllo se è stata premuto il tasto freccia destra, in caso positivo richiama la funzione per poter modificare il valore (es. per modificare i lampeggi richiama la funzione set_blinkers).

navigate_menu.s : richiama la funzione move, se questa ha dato risultato due controlla che l'index corrente sia modificabile in caso positivo viene messo un flag a true. In caso move non dia come risultato 2 (quindi o 1 o -1) verrà aggiunto il risultato al corrente index tenendo conto se si è oppure no in modalità supervisore.

VARIABILI

DESCRIZIONE DELLE VARIABILI ED IL LORO SCOPO

ind : variabile globale che salva la posizione (index) nel menù. Inizialmente ad 1, viene modificata dalla funzione navigate_menu .

sub : variabile globale che assume valore 1 se è stato premuto il tasto freccia destra per una funzione del menù modificabile, altrimenti vale 0.

door_lock: variabile globale che salva il valore della funzionalità del menu *blocco automatico porte* (4). Di default a 1, che corrisponde ad ON, se viene premuta freccia destra e poi freccia su/giù il valore si modifica e va a 0 (OFF). Se invece il valore salvato è zero allora premendo freccia su/giù la variabile va a uno.

back_home : variabile globale che contiene il valore della funzionalità *Back-Home* del menù. Si modifica con lo stesso funzionamento della variabile door_lock.

blinkers : variabile globale che salva il numero di lampeggi dell'automobile in autostrada (funzione *frecce direzione*). Questa variabile viene modificata dalla funzione set_blinkers.

Per ogni funzione vengono usate delle altre variabili temporanee che servono per salvare per breve tempo i valori, fare calcoli e poi restituirlo.

DESCRIZIONE DELLE MODALITA' DI PASSAGGIO DELLE VARIABILI TRA FUNZIONI

```
supervisor: .byte 0        # ah
ind:        .byte 1        # al
sub:        .byte 0        # bh
door_lock:  .byte 1        # bl
back_home:  .byte 1        # ch
blinkers:   .byte 3        # cl

is_up or is_down           # dh
is_right                   # dl
```

Come segnato nel main, abbiamo attribuito ad ogni variabile locale un registro da 8 bit.

Queste variabili locali sono presenti anche nelle altre funzioni e vengono inizializzate con il metodo *initiate* all'inizio di ogni funzione:

```
initiate:
    movb %ah, supervisor
    movb %al, ind
    movb %bh, sub
    movb %bl, door_lock
    movb %ch, back_home
    movb %cl, blinkers
```

Durante l'esecuzione vengono modificate ed i registri sovrascritti, quindi per passarne il valore alla funzione chiamante bisogna mettere il valore di nuovo nei vari registri. Per questo ogni funzione al termine delle altre istruzioni chiama il metodo *return* per poi chiamare la ret:

```
return:
```

```
movb supervisor,    %ah
movb ind,           %al
movb sub,           %bh
movb door_lock,     %bl
movb back_home,     %ch
movb blinkers,      %cl
ret
```

ATTUAZIONE

PSEUDO-CODICE AD ALTO LIVELLO

```
# Pseudocodice

// initiate some global variables shared by all functions
ind = 1 // index var
sub = 0 // submenu var
door_lock = 1 // bool, true if doors auto lock is on
back_home = 1 // bool, true if back home mode is on
blinkers = 3 // how many times blinkers blink (supervisor mode)

### MAIN(terminal input) // return int
   supervisor = 0
   if terminal input = 2244
      supervisor = 1

   index_position_message(supervisor);

   while (true)
      navigate_menu(supervisor)

   return 0

### INDEX_POSITION_MESSAGE(supervisor)
   switch (ind)
      case 1:
         if (supervisor = 1)
            print "1. Setting automobile (supervisor):"
         else
            print "1. Setting automobile:"
      case 2:
         print "2. Data: 15/06/2014"
      case 3:
         print "3. Ora: 15:32"
      case 4:
         if (sub)
            read = move()
            if (read = -1 or read = 1)
```

```
            door_lock = not(door_lock)
          sub = 0
        if (door_lock)
          print "4. Blocco automatico porte: ON"
        else
          print "4. Blocco automatico porte: OFF"
      case 5:
        if (sub)
          read = move();
          if (read = -1 or read = 1)
            back_home = not(back_home)
          sub = 0
        if (back_home)
          print "5. Back-home: ON"
        else
          print "5. Back-home: OFF"
      case 6:
        print "6. Check olio"
      case 7:
        if (sub)
          set_blinkers()
          sub = 0
        print "7. Frecce direzione: *blinkers*"
      case 8:
        if (sub)
          print "Pressione gomme resettata"
          sub = 0
        print "8. Reset pressione gomme"

### NAVIGATE_MENU(supervisor)
  read = move()
  if (pressed right arrow and index has a submenu)
    sub = 1
  else if (pressed up or down)
    ind = ind + read
    if (supervisor = true)
      if (ind < 1)
        ind = 8
      else if (ind > 8)
        ind = 1
    else
      if (ind < 1)
        ind = 6
      else if (ind > 6)
        ind = 1
  index_position_message(supervisor);

### MOVE() // return int
```

```
  get char c

  if (c = up)
     return -1
  else if (c = down)
     return 1
  else if (c = right)
     return 2

  return 0

### SET_BLINKERS()
  get n
  if (n > 5)
     blinkers = 5
  else if (n < 2)
     blinkers = 2
  else
     blinkers = n
```

SCELTE PROGETTUALI

PARTE GRAFICA

- **Visualizzazione voci del menu**: quando si scorre il menu si possono vedere le diverse voci con già il loro valore attuale (es. 2. Data: 15/06/2014 o 5. Back-Home: ON) senza dover andare nel sottomenu. Abbiamo fatto questa scelta dopo aver letto l'elaborato e abbiamo ritenuto che così facendo sarebbe stato più semplice visualizzare il valore delle variabili
- **Ristampa voce menu con valore modificato**: quando si va nel sottomenu di back-home, di blocco automatico porte e frecce direzione è possibile cambiare l'attuale valore delle variabili. Una volta completata l'operazione la voce viene visualizzata nuovamente con il valore modificato. Abbiamo scelto di fare questo per dare subito la conferma all'utilizzatore dell'avvenuta modifica del campo senza doverci ritornare sopra.

CODICE

- **Divisione delle funzioni in diversi file**: abbiamo scelto di dividere le funzioni in diversi file per avere più chiarezza all'interno del codice. Per creare queste funzioni ci siamo attenuti allo pseudo-codice.
- **Passaggio delle variabili locali del main**: per gestire il passaggio delle variabili del main abbiamo deciso di usare i registri a 8 bit: ah, al, bh, bl, ch, cl, in quanto i valori da passare sono principalmente valori booleani (0 o 1) o comunque numeri non maggiori di 8 bit.

CRITICITA’ RISCONTRATE E SOLUZIONI

- **Passare le variabili tra funzioni**: se fosse stato un unico file sarebbe bastato salvare di volta in volta i valori modificati nelle variabili locali, ma nel nostro caso, con le funzioni in diversi file, questo non era possibile, avevamo bisogno di un modo per immagazzinare i dati e passarli in ogni funzione. Per questo abbiamo scelto di usare i registri già precedentemente nominati.
- **Prendere gli elementi da cmd**: gli elementi passati da linea di comando vengono salvati dentro lo stack ed ognuno ha la sua posizione. Per controllare se ci fossero elementi oltre all’eseguibile è stato necessario prendere l’elemento interno a %esp, mentre per prendere il codice bisognava spostarsi di 8 da %esp e prendere il valore contenuto all’indirizzo.
- **Frecce**: per prendere come input da tastiera freccia su, giù e destra sono servite tre call ad interrupt ed ognuna prendeva una parte del codice dell’input. L’ultima conteneva la lettera collegata al verso della freccia e ci ha permesso di riconoscere l’input impartito.