Prova d'esame di Architetture dei Sistemi di Elaborazione

Un venditore al dettaglio deve rifornire il proprio negozio acquistando la merce da un grossista. I prodotti venduti dal grossista sono elencati in una tabella `Price_list` di `N` righe (`N` è una costante definita con `EQU`). Ogni riga contiene:

- un codice esadecimale che identifica univocamente il prodotto, espresso su 1 word (4 byte)
- il prezzo di un'unità del prodotto, espresso su 1 word.

Le righe di `Price_list` sono ordinate in base al codice identificativo crescente, ma i codici identificativi non sono necessariamente consecutivi (i prodotti fuori produzione sono rimossi dal catalogo del grossista).
Il venditore ha elencato i prodotti da acquistare in una tabella `Item_list` di `M` righe (`M` è una costante definita con `EQU`). Ogni riga contiene:

- il codice identificativo prodotto, espresso su 1 word
- la quantità del prodotto da acquistare, espressa su 1 word.

Le righe di `Item_list` non sono ordinate.

Si richiede di:

1) Creare una directory **soluzione** (si consiglia all'interno del disco Z:).
2) Creare un nuovo progetto con Keil all'interno della directory **soluzione** per la scheda LPC1768 di NXP. Il nome del progetto può essere scelto a piacere.
3) Scrivere subroutine **debuggate e funzionanti** che rispondano alle 3 specifiche seguenti.
4) Al termine, copiare il file makezip.bat nella stessa posizione in cui si trova la directory **soluzione** (N.B. makezip.bat non deve essere all'interno della directory **soluzione**). Cliccare due volte su makezip.bat. Verificare che sia stato creato il file Z:\ASE\test\soluzione.zip

Specifica 1 (8 punti). Scrivere una subroutine `sequentialSearch` che calcoli la spesa sostenuta dal negoziante per acquistare i prodotti dal grossista.
La subroutine riceve attraverso lo stack i seguenti parametri, nell'ordine indicato:

- spazio per il valore di ritorno: al termine della subroutine, qui sarà salvata la spesa totale
- indirizzo del vettore `Item_list`
- indirizzo del vettore `Price_list`

Attraverso una ricerca lineare (con un doppio ciclo while), la subroutine `sequentialSearch` calcola la spesa complessiva e salva il valore nello stack.
Si supponga che ogni codice identificativo presente in `Item_list` esista anche in `Price_list`.
Di seguito si fornisce un esempio del programma chiamante:

```
N EQU 30
M EQU 4

Reset_Handler   PROC
EXPORT  Reset_Handler    [WEAK]
     LDR r0, =Price_list
     LDR r1, =Item_list
     MOV r2, #0
     PUSH {r0, r1, r2}
     BL sequentialSearch
     POP {r0, r1, r2}
stop B stop
     ENDP
```

```
Price_list DCD 0x004, 20, 0x006, 15, 0x007, 10, 0x00A, 5, 0x010, 8
     DCD 0x012, 7, 0x016, 22, 0x017, 17, 0x018, 38, 0x01A, 22
     DCD 0x01B, 34, 0x01E, 11, 0x022, 3, 0x023, 9, 0x025, 40
     DCD 0x027, 12, 0x028, 11, 0x02C, 45, 0x02D, 10, 0x031, 40
     DCD 0x033, 45, 0x035, 9, 0x036, 11, 0x039, 12, 0x03C, 19
     DCD 0x03E, 1, 0x041, 20, 0x042, 30, 0x045, 12, 0x047, 7

Item_list DCD 0x022, 4, 0x006, 1, 0x03E, 10, 0x017, 2
```

Nell'esempio riportato, il valore restituito dalla subroutine è 71 (0x47).

Specifica 2 (6 punti). Scrivere una subroutine `binarySearch` che calcoli la spesa sostenuta dal negoziante per acquistare i prodotti dal grossista.
La subroutine riceve attraverso lo stack i seguenti parametri, nell'ordine indicato:

- spazio per il valore di ritorno: al termine della subroutine, qui sarà salvata la spesa totale
- indirizzo del vettore `Item_list`
- indirizzo del vettore `Price_list`

La subroutine `binarySearch` sfrutta l'ordinamento di `Price_list` in base ai codici identificativi per trovare i prodotti tramite una ricerca binaria. Si riporta lo pseudocodice della ricerca binaria:

```
first = 0;
last = N - 1;
index = 0;
while (1)
{
     middle = (first + last) / 2;
     if (key == table[middle])
     {
          index = middle;     /* element found */
          break;
     }
     else
          if (key < table[middle])
               last = middle - 1;
          else
               first = middle + 1;
}
```

Si supponga che ogni codice identificativo presente in `Item_list` esista anche in `Price_list`. Il programma chiamante è lo stesso della specifica 1, sostituendo `BL sequentialSearch` con `BL binarySearch`.

Specifica 3 (4 punti). Scrivere due subroutine `robustSequentialSearch` e `robustBinarySearch` che calcolino la spesa sostenuta dal negoziante per acquistare i prodotti dal grossista. Rispetto alle subroutine delle specifiche 1 e 2, le due nuove subroutine devono prevedere la possibilità che uno o più codici identificativi presenti in `Item_list` non esistano in `Price_list`. In tal caso, il valore di ritorno è -1. Ad esempio nel vettore
`Item_list2 DCD 0x022, 4, 0x005, 1, 0x03E, 10, 0x017, 2`
il prodotto con codice identificativo 0x005 non esiste in `Price_list`.
Suggerimento: per effettuare il controllo nella ricerca binaria, nello pseudocodice è sufficiente sostituire il ciclo infinito `while (1)` con `while (first <= last)`.