X36SKD

lab

Phonebook

Miroslav Skrbek, Jiří Buček

Literature

[1] *8-bit AVR® Instruction Set*, ATMEL Corporation, 2002. http://www.atmel.com

[2] *8-bit AVR® Microcontroller width 16K Bytes In-System Programmable Flash ATmega169*. Datasheet. ATMEL Corporation, 2003. http://www.atmel.com

[3] *AVR Assembler User Guide*. ATMEL Corporation, 2002. http://www.atmel.com

[4] *AVR065: LCD Driver for the STK502 and AVR Butterfly.* Application note. ATMEL Corporation, 2004. http://www.atmel.com

Task

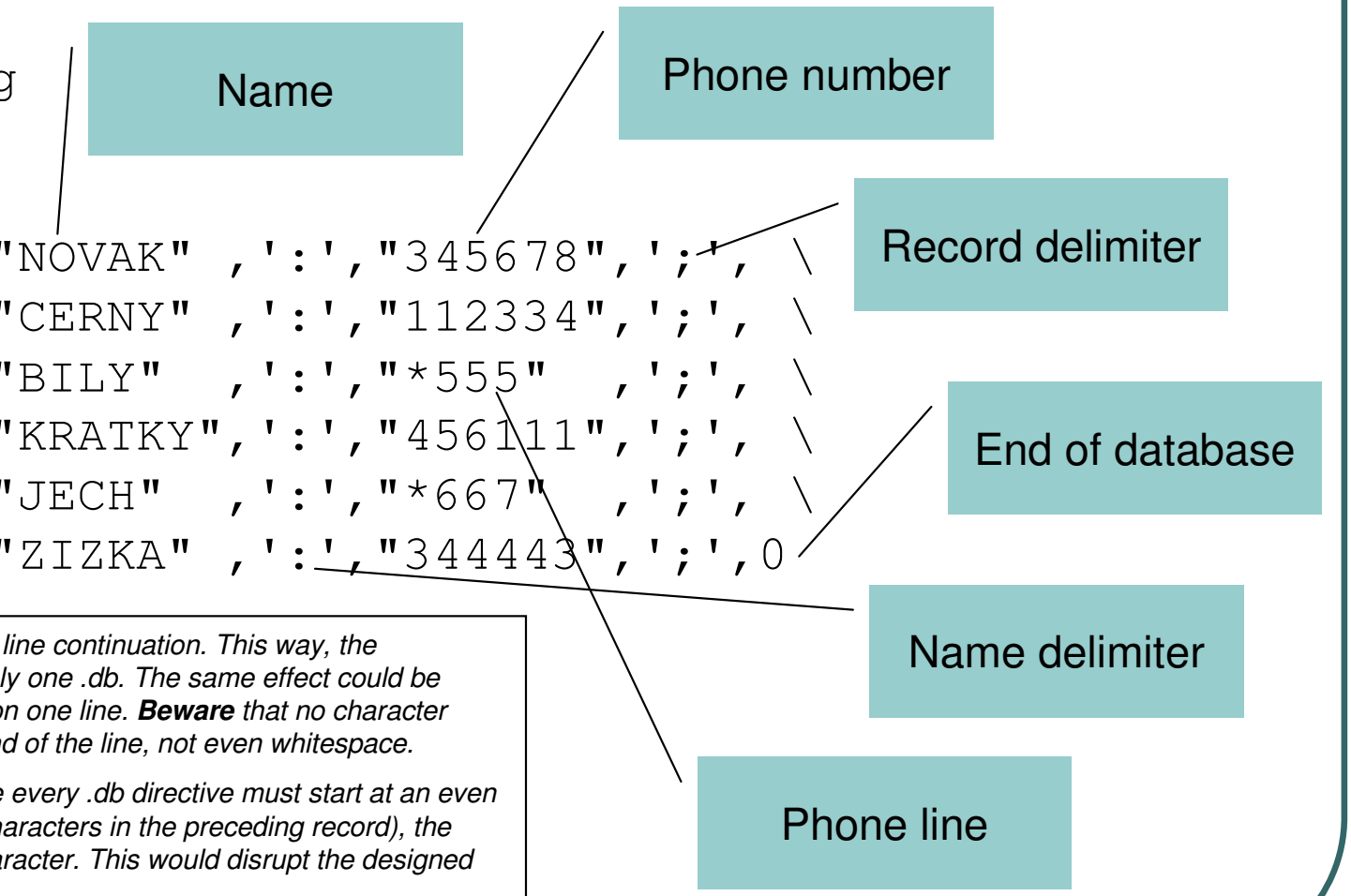
Write a program that implements a phonebook. Use AVR assembler. The program shall contain a database of names and phone numbers. Every record shall contain the name (max. 6 characters) and the phone number (max. 6 digits). The records shall be a read-only, fixed part of the program's data structures.

The phonebook shall work in two modes:

1. **Automatic**. The names appear sequentially on the display in one-second intervals, sorted lexicographically in ascending (descending) order. The ordering direction shall be controlled using joystick (up/down). Pressing joystick (enter) shall stop the sequence and the phonebook shall switch to mode 2.
2. **Manual.** The display contains the last displayed name from mode 1. Horizontal joystick controls shall switch between name (left) and phone number (right). Vertical joystick controls shall step through the sequence by one record towards the beginning (up) or end (down). The stepping is enabled only when the name is displayed. Pressing joystick (enter) shall switch to mode 1.

Database structure

```
        .cseg
database:
        .db "NOVAK" ,':',"345678",';', \
            "CERNY" ,':',"112334",';', \
            "BILY"  ,':',"*555"  ,';', \
            "KRATKY",':',"456111",';', \
            "JECH"  ,':',"*667"  ,';', \
            "ZIZKA" ,':',"344443",';',0
```

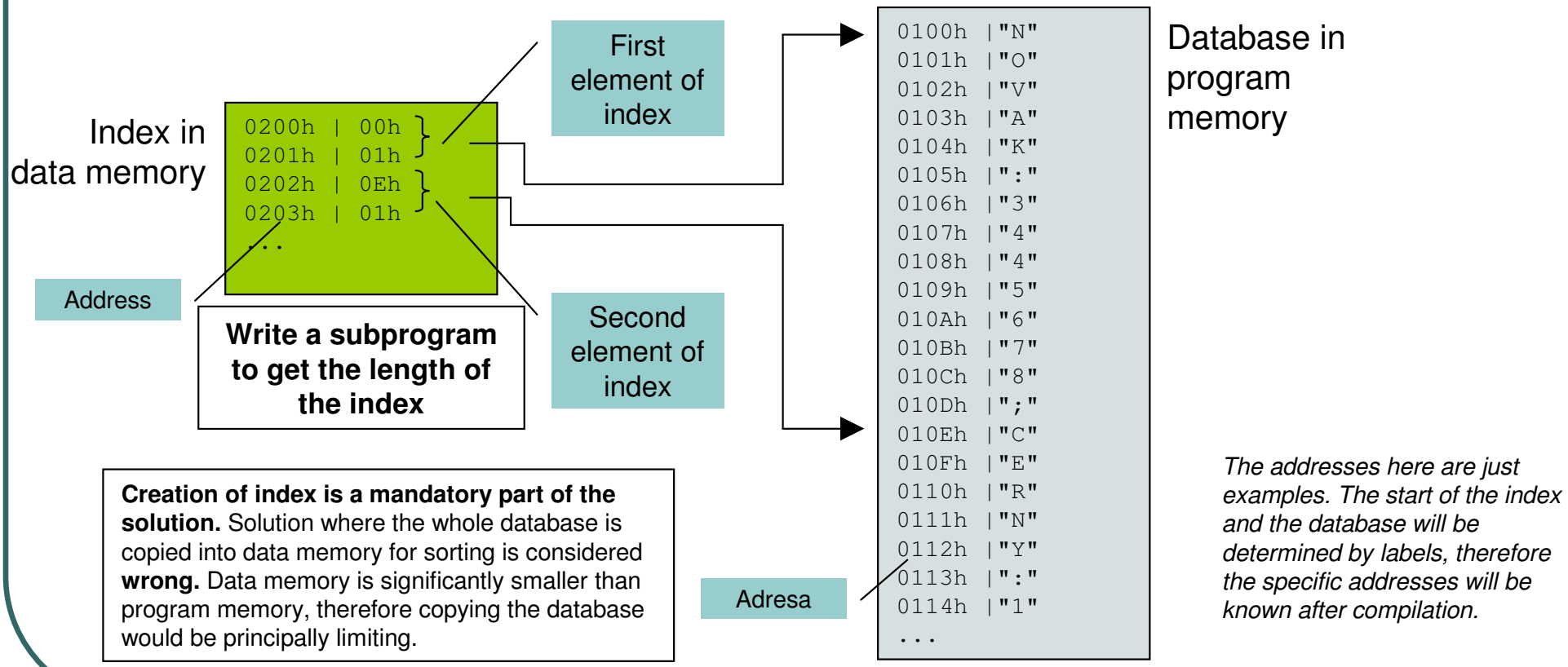


Backslash at the end of a line means line continuation. This way, the declaration can be achieved using only one .db. The same effect could be accomplished by placing everything on one line. ***Beware*** *that no character may appear after the backslash till end of the line, not even whitespace.*

This notation had to be used because every .db directive must start at an even address, and if not (odd number of characters in the preceding record), the compiler automatically fills a zero character. This would disrupt the designed database format.

Sorting the database I

First write a program that scans the database and creates an *index* of the database in memory. The index is an array of addresses of all records in the database.



Creation of index is a mandatory part of the solution. Solution where the whole database is copied into data memory for sorting is considered **wrong.** Data memory is significantly smaller than program memory, therefore copying the database would be principally limiting.

The addresses here are just examples. The start of the index and the database will be determined by labels, therefore the specific addresses will be known after compilation.

Sorting the database II

Let us prepare a subprogram, whose inputs are the addresses of names from two records (record A and B). The output will be:

- -1 when the name in record A lexicographically precedes the name in record B (i.e. A.name < B.name),
- 0 when A.name is equal to B.name (A.name == B.name)
- 1 when the name in record A succeeds the name in record B (i.e. A.name > B.name).

Comparison algorithm

1. Assign i = 0
2. Compare ASCII codes of the characters at position i from name A (A.name[i]) and from name B (B.name[i]).
   - If A.name[i] < B.name[i], then A.name < B.name and we return -1
   - If A.name[i] > B.name[i], then A.name > B.name and we return 1
   - If A.name[i] == B.name[i], then repeat step 2 for the next character (i.e.. for i = i + 1).

   If we reach end of one of the strings:
   - return 1, when A.name is longer than B.name
   - return -1, when B.name is longer than A.name
   - return 0, when we reached the end of both strings.

Sorting the database III

We use Bubble-Sort for sorting the index:

1. Go to the beginning of index
   (i.e. the first index element is the current now)
2. From the index, take the current element (A) and the next one (B)
3. If database[A].name > database[B].name, exchange the elements A and B in the index.
4. Go to the next element (step one element towards the end)
5. If this is not the last element, continue with step 2
6. If there was at least one exchange in step 3, continue with step 1
7. End, the index is now sorted in ascending order

Descending sorting is similar, only change the comparison in step 3 to database[A].name < database[B].name

How to display

Sequential display of sorted database:

Go through the index one element at a time. Each element contains the address of one record in program memory. Get the address and use it to access the name and the number from the database. Periodic display update in 1 second intervals shall be done using interrupts.

Name and number display:

Create a subprogram whose parameters are – an address of the first character of a string in program memory, and a delimiter. The subprogram displays the following (max. 6) characters until the delimiter. You can call the subprogram with the delimiter ":" to display the name and ";" to display the number.

Access the phone number:

The name always ends with a colon (":"). To get the **number**, you know that it follows the name after the colon. The number always ends with a semicolon (";").

We recommend to create a subprogram, whose parameter will be the address of the beginning of a record. The subprogram will increment the address, until it finds a colon. It will then return the address + 1, which points directly to the beginning of the number.