MŰEGYETEM 1782

8051 Assembly Assignment

Documentation

Microcontroller Based Systems

BMEVIAUAC06

Student:	**Khongmeng Kormoua**
Neptun ID:	**I3MLPQ**
Lecturers:	**Kiss Domokos**
	Viktor Kovács
Date:	**15 October 2022**

Budapest University of Technology and Economics

Department of Automation and Applied Informatics

2022

“I, Khongmeng Kormoua, declare that this is my and only my own solution”

Task description

- Search for the largest element in a number sequence (array) stored in the code memory.
- Every element is an 8-bit unsigned integer.
- Inputs: Start address of the array (pointer), number of elements
- Output: Largest element in a register

Solution

- For this task I will work on the array in the code memory directly, the task does not require to copy the data from code memory to internal memory. That is why I implemented my solution based on the requirement of the task.

Detailed Implementation of the subroutine “FIND_MAX_NUMBER”

- Inputs: - DPTR - start address of the array (first element of the array).
  - R7 - number of elements.
- Output: R3 - maximum number from the array
- First, when enter the subroutine the subroutine will save all the contents of the memory to the stack using the PUSH instruction as shown below.

```
PUSH AR2
PUSH AR4
PUSH AR5
PUSH AR6
PUSH AR7
PUSH PCON
PUSH IE
PUSH IP
PUSH PSW
PUSH ACC
PUSH B
PUSH DPH
PUSH DPL
```

- After that the subroutine will read the first element from the array as below.

```
CLR A
MOVC A, @A+DPTR
```

- since this is the first element, this is automatically the biggest value for now since this is the only value that the subroutine has so far. Therefore, the subroutine will assign this value to R3 (which is the output) immediately

```
MOV R3, A
```

- Also, the first element has been loaded and taken care of so the pointer to the array "DPTR" and the remaining size of the array will be incremented and decremented respectively

```
INC DPTR
DEC R7
```

- Now all the preparation is ready for a subroutine loop to process the data if there is a data remain in the array to be processed. This loop will be called "SUBROUTINELOOP" and right after going inside the loop, the subroutine will read a new value from the array and save it to temporary variable which in this case is register R4

```
SUBROUTINELOOP:

CLR A
MOVC A, @A+DPTR
MOV R4, A
```

- Here comes to the trickiest part of the subroutine, how to compare and decide which one is the biggest. For this, I could use other compare instruction, but I just decided to use subtraction instruction "SUBB A, Rn". Which will subtract Rn from A. if A > Rn then the result is saved in A. if A == Rn then result will be 0 and save to A. if A < Rn then overflow will happen, and the carry bit will be set. The idea to solve our task which is to find out which one is bigger is to put the biggest that we know so far to A then subtract the new element from A. if the carry bit is set. It means that the new element is bigger than the current biggest element. On the other hand, if the carry bit is not set, then it means that the new element is less than the current biggest element. For each case the subroutine will jump to a specific label accordingly

```
MOV A, R3
SUBB A, R4
JC NEWVALUEISBIGGER
JMP NEWVALUEISLESS
```

- Now after we can decide where to jump to, it is straight forward how to implement in each case. For the case that we value is bigger than we just copy the new value of R4 to R3 (the biggest element that we know so far). On the other case, when the new value is less than the current biggest value, there is nothing to do in other word just skip the assigning part from R3 to R4.

```
NEWVALUEISBIGGER:

MOV B, R4
MOV R3, B

NEWVALUEISLESS:

```

- From this point, since the subroutine has already taken care of the new element. The subroutine is going to Increment the DPTR, and Decrement R7. One more thing to do is to check if R7 is zero or not. If it is that means, there is nothing left in the array to process furthermore. If R7 is not zero yet, that means there is still some element left in the array. Therefore, the program should jump back to the subroutine loop and execute the whole process again.

```
INC DPTR
DJNZ R7, SUBROUTINELOOP
```

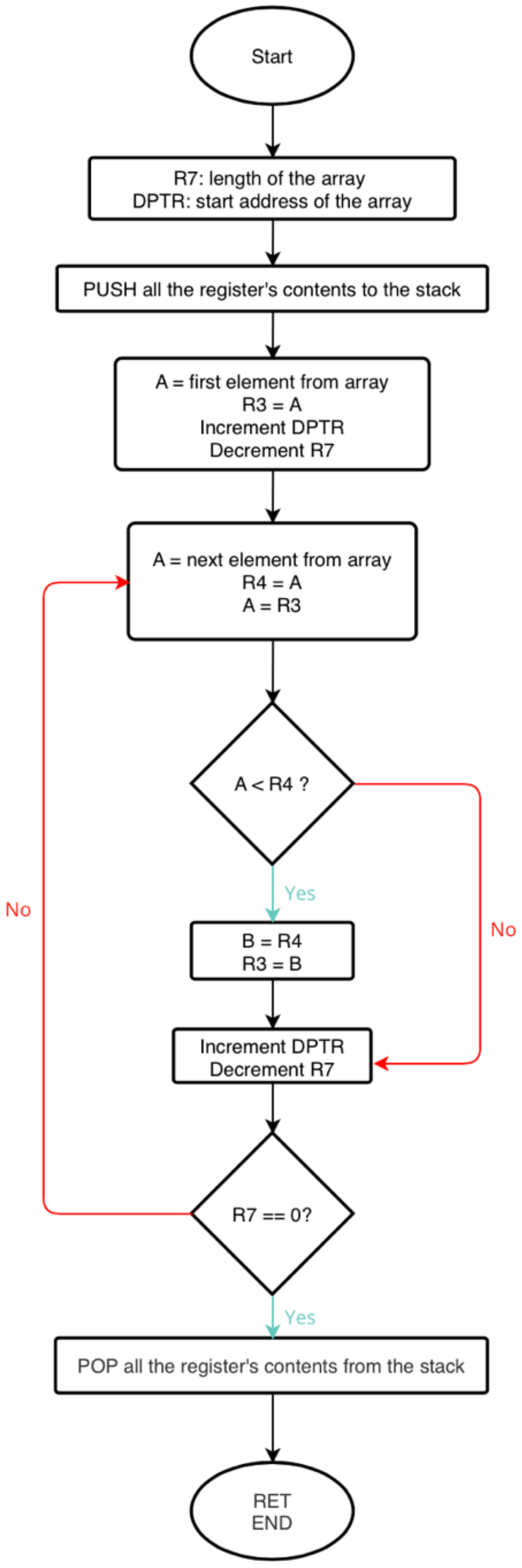
- When the processing of the subroutine loop is done, then one more thing to do before return to the main program is to load all register's contents back from the stack. Note that it is necessary to load them back in the reverse order when the subroutine save them.

```
POP DPL
POP DPH
POP B
POP ACC
POP PSW
POP IP
POP IE
POP PCON
POP AR7
POP AR6
POP AR5
POP AR4
POP AR2

RET
```

Flow chart of "FIND_MAX_NUM"

- Below is the flow chard diagram of the subroutine "FIND_MAX_NUMBER" regarding each execution and decision to fulfill the task description

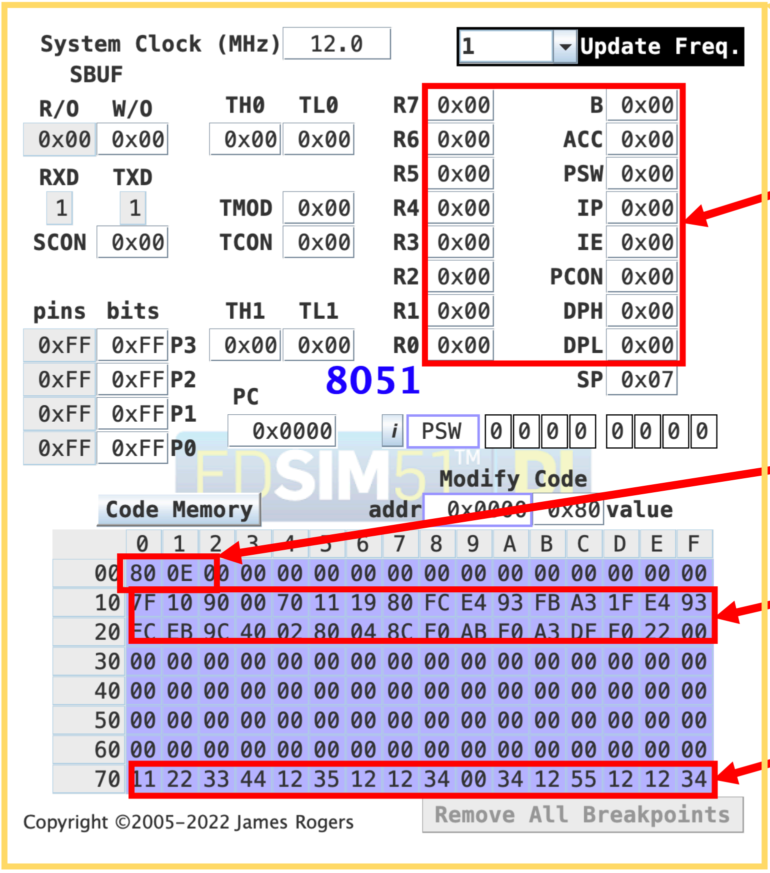

Start
R7: length of the array
DPTR: start address of the array
PUSH all the register's contents to the stack
A = first element from array
R3 = A
Increment DPTR
Decrement R7
A = next element from array
R4 = A
A = R3
A < R4 ?
Yes
No
B = R4
R3 = B
Increment DPTR
Decrement R7
R7 == 0?
No
Yes
POP all the register's contents from the stack
RET
END

<u>Subroutine:</u> "FIND_MAX_NUM"

<u>Inputs:</u> R7 - length of the array
DPTR - start address of the array

<u>Output:</u> R3 - Max value from the array.

Simulation result



```
System Clock (MHz) 12.0              1   Update Freq.
  SBUF
R/O   W/O      TH0   TL0    R7 0x00        B 0x00
0x00  0x00     0x00  0x00   R6 0x00      ACC 0x00
RXD   TXD                   R5 0x00      PSW 0x00
 1     1       TMOD  0x00   R4 0x00       IP 0x00
SCON  0x00     TCON  0x00   R3 0x00       IE 0x00
                            R2 0x00     PCON 0x00
pins  bits     TH1   TL1    R1 0x00      DPH 0x00
0xFF  0xFF P3  0x00  0x00   R0 0x00      DPL 0x00
0xFF  0xFF P2          8051               SP 0x07
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x0000   i  PSW  0 0 0 0  0 0 0 0
                          Modify Code
  Code Memory          addr 0x0000  0x80 value
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
  00 80 0E 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  10 7F 10 90 00 70 11 19 80 FC E4 93 FB A3 1F E4 93
  20 FC EB 9C 40 02 80 04 8C F0 AB F0 A3 DF F0 22 00
  30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  70 11 22 33 44 12 35 12 12 34 00 34 12 55 12 12 34
Copyright ©2005-2022 James Rogers      Remove All Breakpoints
```

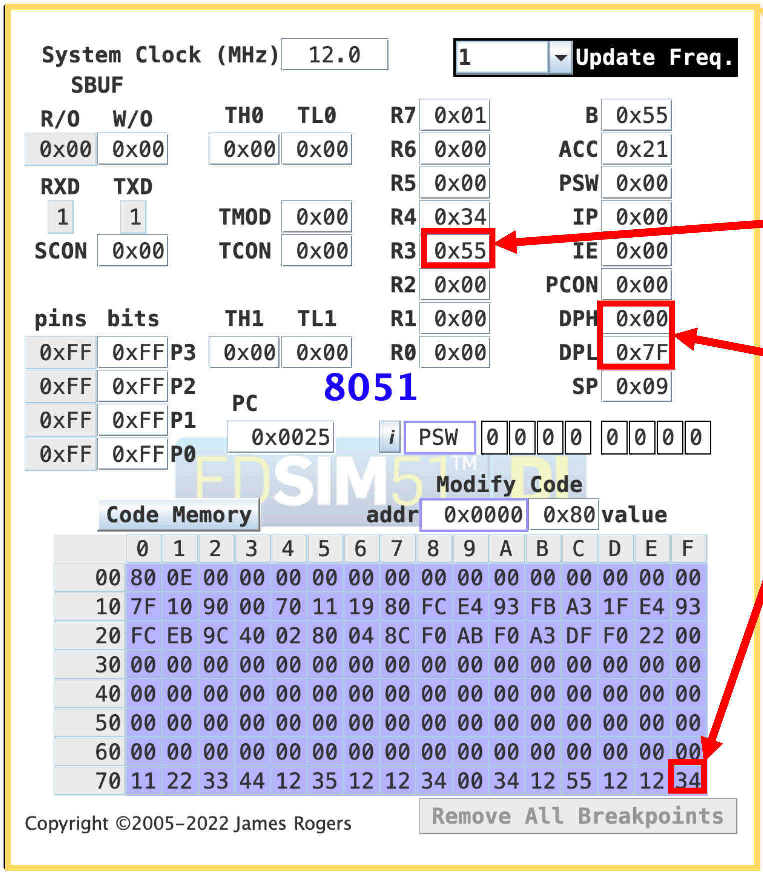
A register and memory contents right after assembly

Initially all the register value has no much meaning for us just to note that all of them are 0 in this case

At address 0000, there is only one instruction there which is SJMP instruction

All of our instructions begin at address 0010 as expected

The array contents is start at address 0070 as defined



```
System Clock (MHz) 12.0              1   Update Freq.
  SBUF
R/O   W/O      TH0   TL0    R7 0x01        B 0x55
0x00  0x00     0x00  0x00   R6 0x00      ACC 0x21
RXD   TXD                   R5 0x00      PSW 0x00
 1     1       TMOD  0x00   R4 0x34       IP 0x00
SCON  0x00     TCON  0x00   R3 0x55       IE 0x00
                            R2 0x00     PCON 0x00
pins  bits     TH1   TL1    R1 0x00      DPH 0x00
0xFF  0xFF P3  0x00  0x00   R0 0x00      DPL 0x7F
0xFF  0xFF P2          8051               SP 0x09
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x0025   i  PSW  0 0 0 0  0 0 0 0
                          Modify Code
  Code Memory          addr 0x0000  0x80 value
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
  00 80 0E 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  10 7F 10 90 00 70 11 19 80 FC E4 93 FB A3 1F E4 93
  20 FC EB 9C 40 02 80 04 8C F0 AB F0 A3 DF F0 22 00
  30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  70 11 22 33 44 12 35 12 12 34 00 34 12 55 12 12 34
Copyright ©2005-2022 James Rogers      Remove All Breakpoints
```

A register and memory contents after iterating through the array

R3 now contain the max value from the array which is 0x55

The array pointer is currently pointing to the last element of the array

Reference

- 8051 Instruction Set (Short Version).pdf
- Microcontroller Based System Lecture and Practice materials
- Personal consultation with professors
- www.win.tue.nl/~aeb/comp/8051/set8051.html