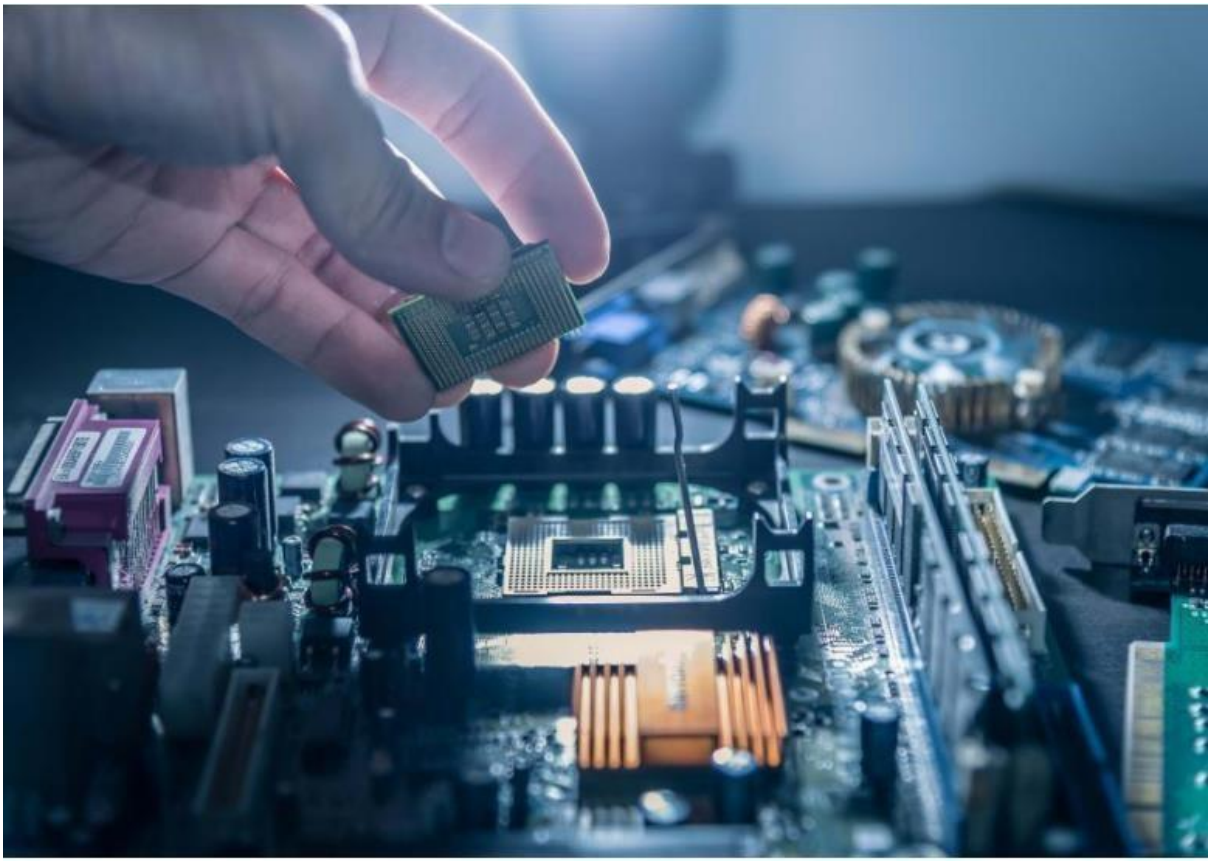
EMBEDDED SYSTEMS LAB

EXPERIMENT 1: Familiarization with 8051 Kit through Assembly language programming

CHANAKYA DUPPATLA: 21EC30017

RAHUL SAMINENI: 21EC30045



EMBEDDED SYSTEMS LABORATORY(EC39302)

DATE: 31/01/24

PART– A

Objective:

Sort an array of 100 random data bytes stored from location 9000H onwards in the external memory of an 8051 kit in ascending order and perform a binary search for the number stored at location 9500H; store the index at location 9550H if found, otherwise store -1 at location 9550H.

Pseudo code

// Initialize array

Set DPTR to start of array

Load counter with size of array

Loop until counter is 0

  Load counter value into A

  Store A to array location (DPTR)

  Increment DPTR

  Decrement counter

// Bubble sort array

Load outer loop counter (R1)

Outer loop:

  Load DPTR with array start address

  Load inner loop counter (R2)

  Inner loop:

    Load current element into A

    Store current element to R3

    Load next element into A

    Store next element to R4

    Compare A and B

    If A > B, swap elements:

    Store R3 to next location

Store R4 to current location

Decrement R2

Loop until R2=0

Decrement R1

Loop until R1=0

// Binary search

Load target value into R7

Load low index to R1, high index to R2

Search loop:

Calculate mid index as (R1 + R2) / 2

Load array element at mid index into A

Compare to target value R7

If match, store mid index to result location

Else if A > R7, update R2 to mid index - 1

Else update R1 to mid index + 1

Check if low and high indices crossed

If not crossed, loop again

If no match found, store -1 to result location

Code :

```
ORG 8100H

MOV DPTR, #9000H
MOV R0, #64H

INIT_LOOP:
   MOV A, R0
   MOVX @DPTR, A
```

```
    INC DPTR
    DJNZ R0, INIT_LOOP

MOV R1, #99

OUTER_LOOP:
    MOV DPTR, #9000H
    MOV R2, #99

INNER_LOOP:
    MOVX A, @DPTR
    MOV R3, A
    MOV B, R3
    INC DPTR
    MOVX A, @DPTR
    MOV R4, A

    CJNE A, B, COMP
    LJMP DEDUCE

COMP:
    JC SWAPPING     LJMP DEDUCE

SWAPPING:
    MOV A, R3
    MOVX @DPTR, A
    DEC DPL
    MOV A, R4
    MOVX @DPTR, A
    INC DPTR
```

```
DEDUCE:
   DJNZ R2, INNER_LOOP
   DJNZ R1, OUTER_LOOP

MOV DPTR, #9500H
MOV A, #67H
MOVX @DPTR, A
MOV R7, A ; SEARCH DATA IN R7

MOV DPTR, #9000H
MOV R1, #00
MOV R2, #63H

SEARCH_LOOP:
   MOV A, R1
   ADD A, R2
   MOV B, #2
   DIV AB

   MOV R3, A
   ADD A, DPL
   MOV DPL, A
   MOVX A, @DPTR
   MOV B, R7
   CJNE A, B, FUNC

   MOV DPTR, #9550H
   MOV A, R3
   MOVX @DPTR, A
```

```
    LJMP ENDL

FUNC:
    JC FUNC1
    LJMP FUNC2

FUNC1:
    MOV A, R3
    ADD A, #1
    MOV R1, A
    LJMP REPEAT

FUNC2:
    MOV A, R3     SUBB A, #1     MOV R2, A
    LJMP REPEAT

REPEAT:
    MOV A, R1     MOV B, R2
    CJNE A, B, CHECK     LJMP INITIAL

CHECK:
    JC INITIAL
    MOV A, #0
    SUBB A, #1
    MOV DPTR, #9550H
    MOVX @DPTR, A
    LJMP ENDL

INITIAL:
    MOV DPTR, #9000H
```

```
    LJMP SEARCH_LOOP

ENDL:
    SJMP $ ; Infinite loop
    END
```

CODE EXPLANATION:

Initialization and Data Storing:

- The program initiates by setting the Data Pointer (DPTR) to 9000H, indicating the array's starting point in external memory.

- R0 is initialized with 64H (100 in decimal), and a loop (INIT_LOOP) is entered. Within this loop, R0 is decremented, and its value is stored at the current DPTR location, effectively storing values from 100 to 1 in the array.

Bubble Sort:

- The array is sorted using the Bubble Sort algorithm. R1 tracks the number of passes (99 in total), and R2 is utilized for comparisons within each pass.

- In each pass, adjacent elements (A and B) are compared. If B is smaller (JC SWAP), a swapping routine (SWAP) is invoked, involving DPL decrementation to move back one position, swapping values, and then DPTR incrementation to proceed.

Adding Data at 9500H:

- The value 58H is stored at 9500H, serving as the target for the binary search.

Binary Search:

- The binary search starts by resetting DPTR to the array's beginning (9000H) and initializing R1 and R2 to denote the lower and upper bounds of the search range.

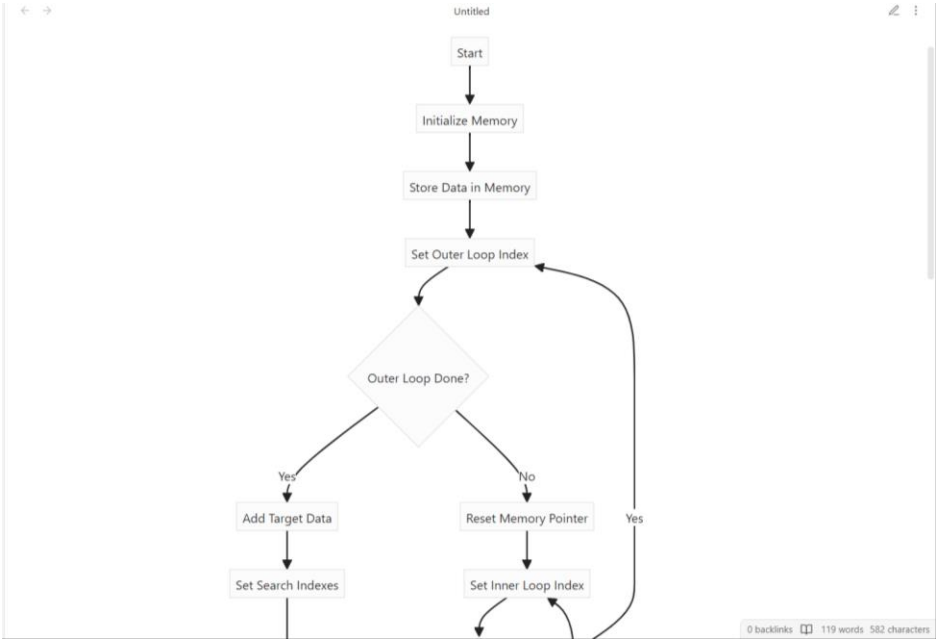
- The search loop (SEARCH_LOOP) calculates the mid-point, compares the mid-point value with the search target (stored in R7, containing 58H), and adjusts the search bounds based on the mid-point value's relation to the search target.

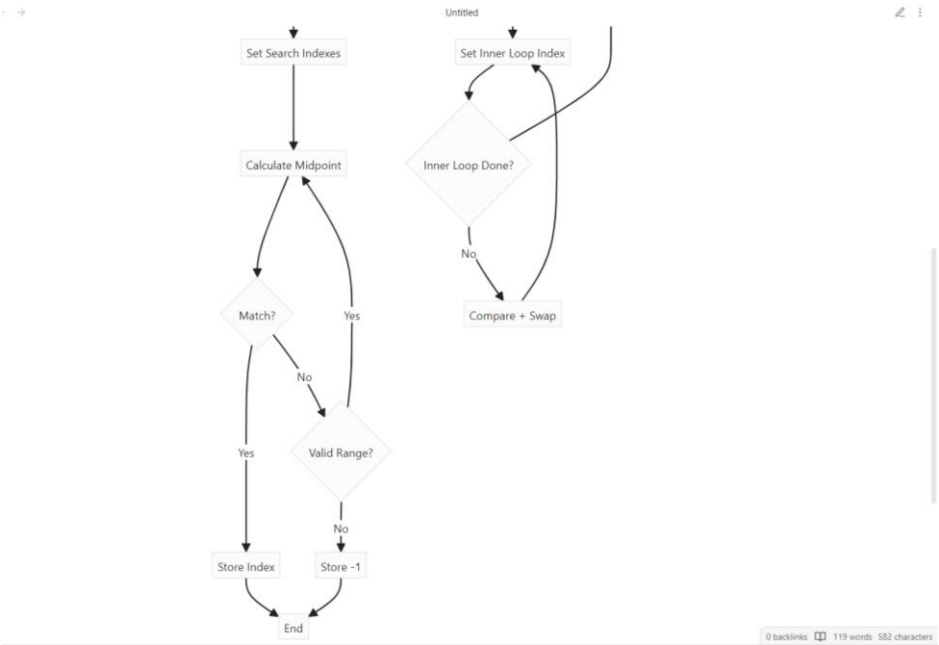
- If the value is found, its index is stored at 9550H. If the search concludes without finding the target (when the lower bound exceeds the upper bound), -1 (represented as FFH due to 8-bit storage) is stored at 9550H.

End of Program:

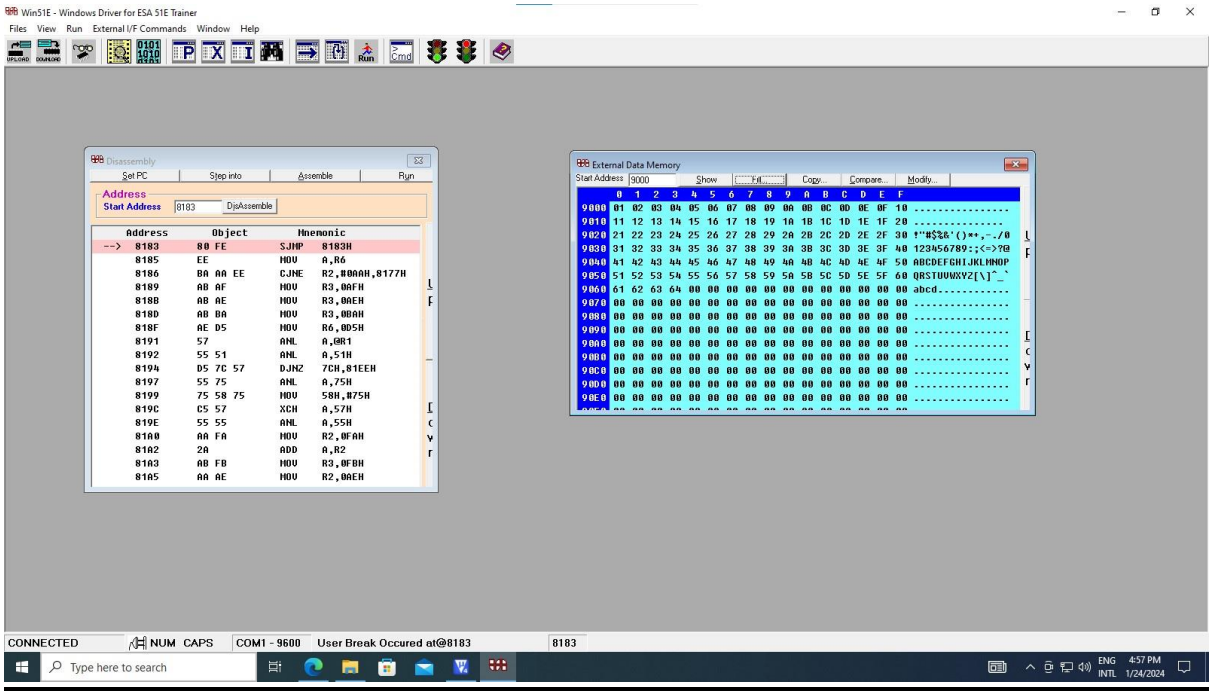
- The program concludes with an infinite loop (SJMP $), signifying the end of execution.

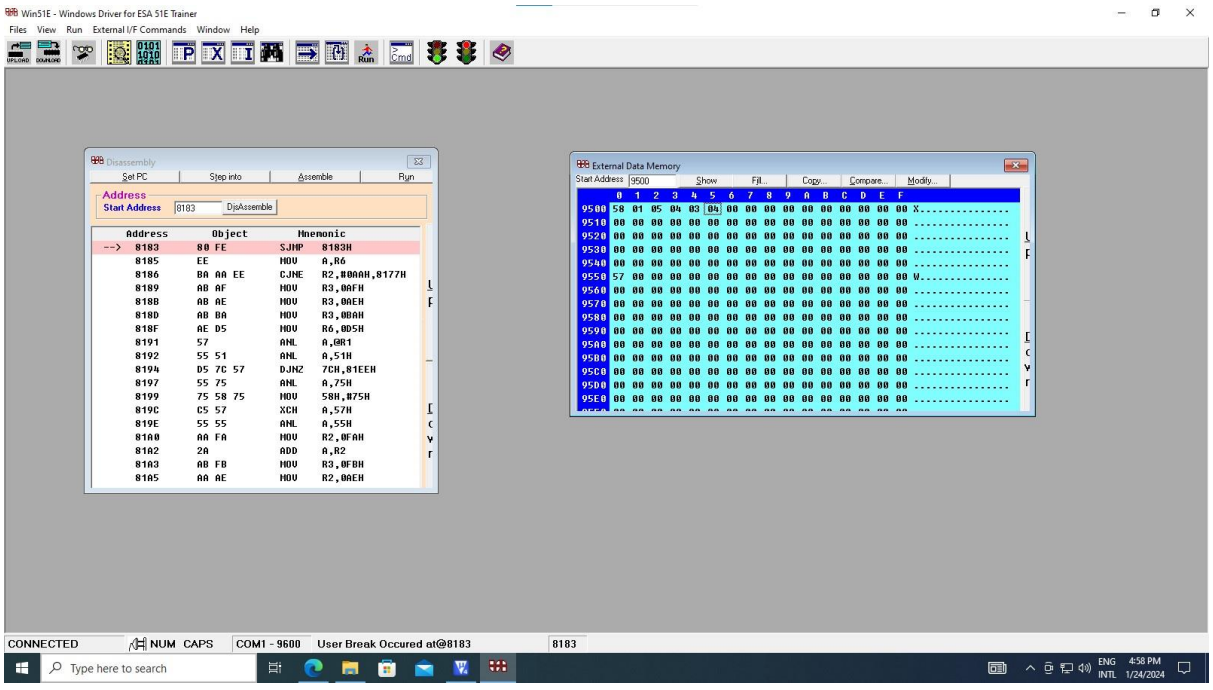
FLOWCHART:

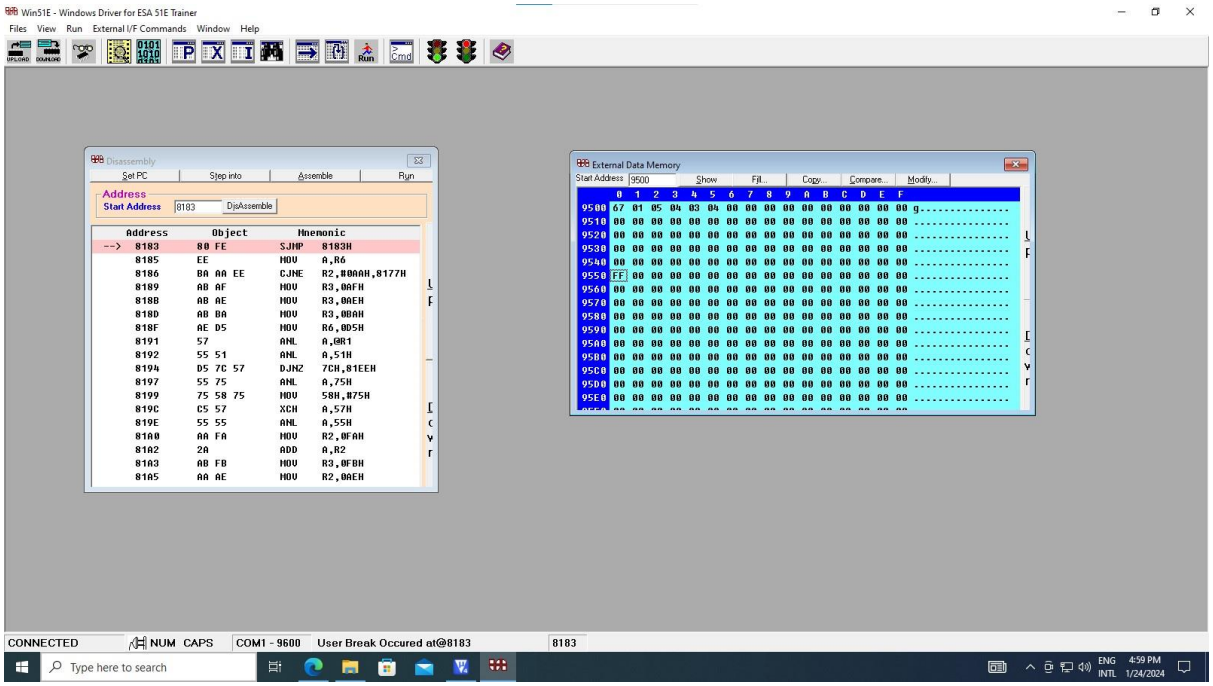




RESULTS:


Win51E - Windows Driver for ESA 51E Trainer
Files View Run External I/F Commands Window Help
Disassembly
Set PC Step into Assemble Run
Address
Start Address 8183 DisAssemble
Address Object Mnemonic
--> 8183 80 FE SJMP 8183H
8185 EE MOV A,R6
8186 BA AA EE CJNE R2,#0AAH,8177H
8189 AB AF MOV R3,0AFH
818B AB AE MOV R3,0AEH
818D AB BA MOV R3,0BAH
818F AE D5 MOV R6,0D5H
8191 57 ANL A,@R1
8192 55 51 ANL A,51H
8194 D5 7C 57 DJNZ 7CH,81EEH
8197 55 75 ANL A,75H
8199 75 58 75 MOV 58H,#75H
819C C5 57 XCH A,57H
819E 55 55 ANL A,55H
81A0 AA FA MOV R2,0FAH
81A2 2A ADD A,R2
81A3 AB FB MOV R3,0FBH
81A5 AA AE MOV R2,0AEH
External Data Memory
Start Address 9000 Show Fill... Copy... Compare... Modify...
CONNECTED NUM CAPS COM1 - 9600 User Break Occured at@8183 8183
Type here to search
ENG INTL 4:57 PM 1/24/2024


Win51E - Windows Driver for ESA 51E Trainer
Files View Run External I/F Commands Window Help
Disassembly
Set PC Step into Assemble Run
Address
Start Address 8183 DisAssemble
Address Object Mnemonic
--> 8183 80 FE SJMP 8183H
8185 EE MOV A,R6
8186 BA AA EE CJNE R2,#0AAH,8177H
8189 AB AF MOV R3,0AFH
818B AB AE MOV R3,0AEH
818D AB BA MOV R3,0BAH
818F AE D5 MOV R6,0D5H
8191 57 ANL A,@R1
8192 55 51 ANL A,51H
8194 D5 7C 57 DJNZ 7CH,81EEH
8197 55 75 ANL A,75H
8199 75 58 75 MOV 58H,#75H
819C C5 57 XCH A,57H
819E 55 55 ANL A,55H
81A0 AA FA MOV R2,0FAH
81A2 2A ADD A,R2
81A3 AB FB MOV R3,0FBH
81A5 AA AE MOV R2,0AEH
External Data Memory
Start Address 9500 Show Fill... Copy... Compare... Modify...
CONNECTED NUM CAPS COM1 - 9600 User Break Occured at@8183 8183
Type here to search
ENG INTL 4:58 PM 1/24/2024



PART – B

OBJECTIVE:

Evaluate the polynomial with coefficients stored at memory locations starting from 9502H for the given values of 'n' and 'x' stored at locations 9500H and 9501H, and store the result at location 9550H.

Pseudo code:

// Load number of terms (n) from location 9500H

Load n from 9500H into register R7

// Load value to substitute (x) from 9501H

Load x from 9501H into register R1

// Initialize result sum register to 0

Set register R3 (sum) to 0

// Loop n times to evaluate each term

Set data pointer DPTR to coefficient address 9502H

For i = 0 to n-1

  Load coefficient from DPTR into register A

  Multiply A and R1 to get term

  Add term to sum register R3

  Increment DPTR to next coefficient address

End loop

// Store final result sum at address 9550H

Store R3 (sum) at address 9550H

Code :

```
;polynomial 5x^3+4x^2+3x+4
; x= 1
ORG 8100H
MOV DPTR, #9500H;
MOV A, #3;`
MOVX @DPTR, A; //n value loaded

INC DPTR;
MOV A, #1H;
MOVX @DPTR, A; // x value loaded

INC DPTR;
MOV A, #5H;
MOVX @DPTR, A;

INC DPTR;
MOV A, #4H;
```

```
MOVX @DPTR, A;

INC DPTR;
MOV A, #3H;
MOVX @DPTR, A;

INC DPTR;
MOV A, #4H;
MOVX @DPTR, A;

;polynomial calculation



MOV DPTR,#9500H
MOVX A,@DPTR
MOV R7,A
INC DPTR
MOVX A,@DPTR
MOV R1,A
MOV DPTR,#9502H
MOVX A,@DPTR
MOV R3,A SUM:
  MOV A,R3    MOV B,R1
  MUL AB
  MOV R0,A
  INC DPTR
  MOVX A,@DPTR
  ADD A,R0
  MOV R3,A
```

```
  DJNZ R7,SUM

MOV A,R3
MOV DPTR,#9550H
MOVX @DPTR,A

SJMP $
END
```

CODE EXPLANATION:

Initialization:

- The program commences at memory address 8100H.
- It initializes the degree of the polynomial n = 3 and stores it at 9500H.
- The value of x = 1 is set and stored at 9501H.

Storing Polynomial Coefficients:

- Coefficients of the polynomial 5x^3 + 4x^2 + 3x + 4 are stored sequentially starting from 9502H.

Polynomial Calculation:

- The program loads n into register R0 and x into register R1.
- It starts with the first coefficient (5 for x^3) in register R3.
- In a loop, it multiplies the current value in R3 by x (from R1), storing the intermediate result in R2.
- It then moves to the next coefficient, adds it to the intermediate result in R2, and updates R3 with this new value.
- This loop continues for each term of the polynomial, decrementing n each time until all terms have been processed.

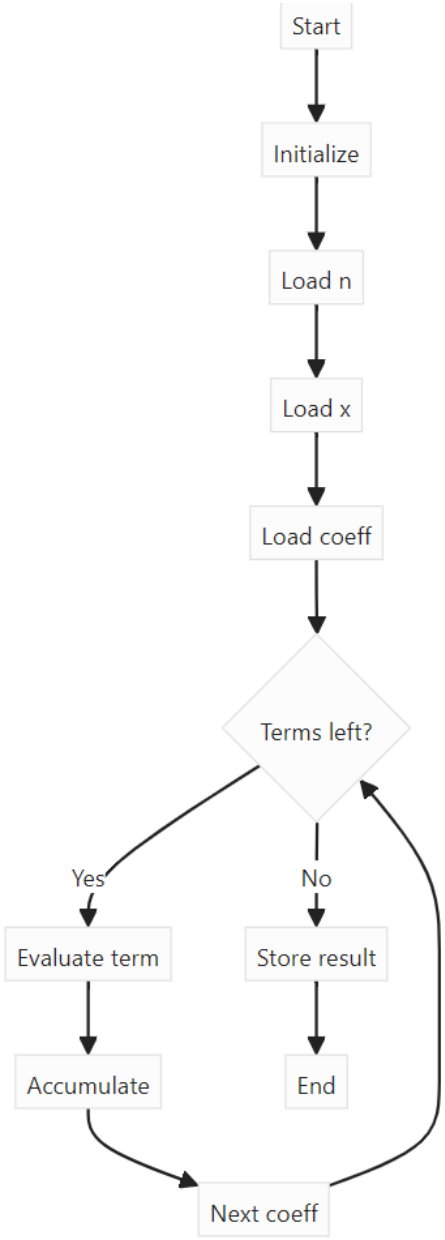
Storing the Result:

- The final result, which is the value of the polynomial for x=1, is stored in R3.
- This result is then stored at memory location 9550H.

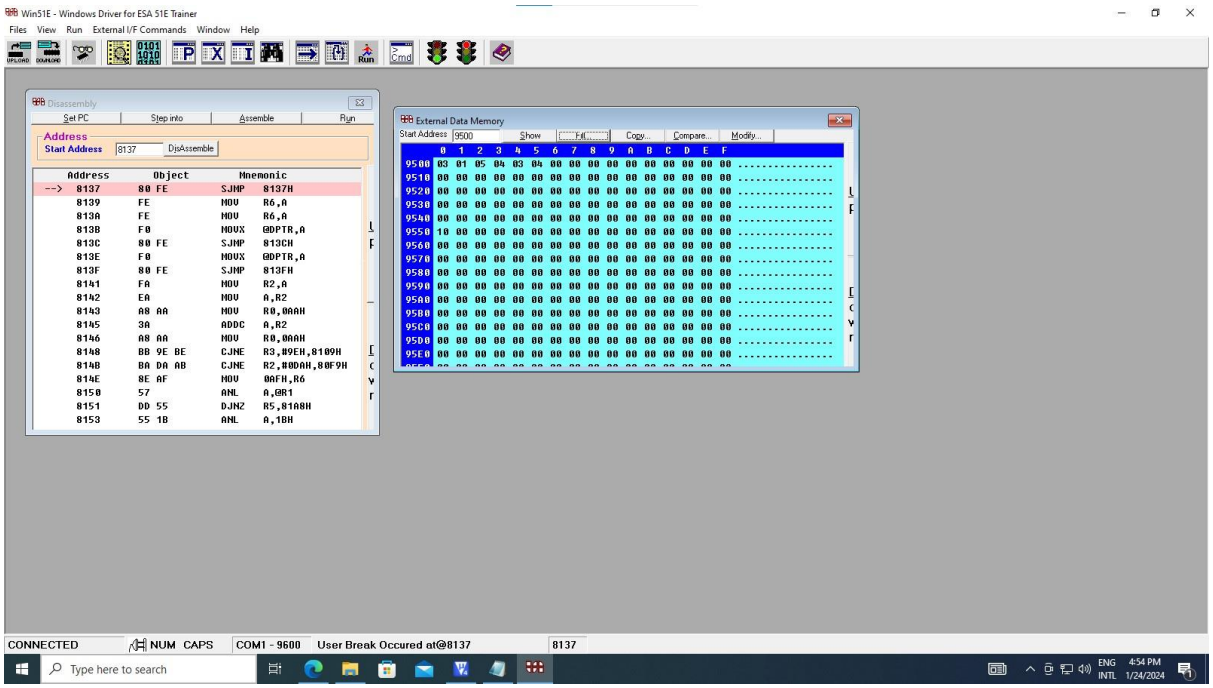
End of Program:

- The program concludes with an infinite loop (SJMP $), indicating the end.

FLOWCHART:



RESULTS:



PART – C

Objective:

Check if the NULL-terminated string at location 9000H is a substring of the string at location 9200H; if true, store the start index in memory location 9500H, otherwise store -1.

Pseudo Code:

// Initialize result index to -1

Set index result to -1

// Load string 1 starting at 9000H

Load string 1 from 9000H

// Load string 2 starting at 9200H

Load string 2 from 9200H

// Set counter for string 1 length (R0)

Set R0 to length of string 1

// Set counter for testing all indices of string 2 (R1)

Set R1 to test all indices of string 2

Outer loop:

 // Set current test index for string 2 (R3)

 Set R3 to current index of string 2

 // Compare characters at current indexes

 Compare string 1 char at R4 to string 2 char at R3

 If characters match:

  Increment R4 and R3 to next char of each string

  Decrement R0 counter

  Repeat character match loop until R0=0

 If all characters matched:

  // Full match found, store index

  Store R3 index in result

 Else:

// No match, increment string 2 index

Increment R3 index of string 2

End outer loop

CODE:

```
ORG 8100H
MOV A, #00H
SUBB A, #01H
MOV DPTR, #9500H
MOVX @DPTR, A


MOV DPTR , #9000H

MOV A, #'R'
MOVX @DPTR, A
INC DPTR

MOV A, #'A'
MOVX @DPTR, A
INC DPTR

MOV A, #10
MOVX @DPTR, A

MOV DPTR , #9200H
MOV A, #'R'
MOVX @DPTR, A
INC DPTR
```

```
MOV A, #'A'
MOVX @DPTR, A
INC DPTR
MOV A, #'H'
MOVX @DPTR, A
INC DPTR
MOV A, #'U'
MOVX @DPTR, A
INC DPTR
MOV A, #'L'
MOVX @DPTR, A
INC DPTR
MOV A, #10
MOVX @DPTR, A

MOV R1, #5
MOV R7, #2
MOV A, R7
MOV R0, A
MOV R3, #00H


outerloop:
MOV DPH, #92H
MOV DPL, R3
MOVX A, @DPTR
MOV B, A
MOV DPH, #90H
MOV R4, #00H
```

```
MOV DPL, R4
MOVX A, @DPTR

CJNE A, B, NOT_EQUAL
DEC R0
MOV A, R3
MOV R5, A

innerloop:
INC R5
INC R4

MOV DPH, #92H
MOV DPL, R5
MOVX A, @DPTR
MOV B, A
MOV DPH, #90H
MOV DPL, R4
MOVX A, @DPTR

CJNE A, B, NOT_EQUALin


DJNZ R0, innerloop
MOV DPTR, #9500H
MOV A, R3
MOVX @DPTR, A
SJMP endd
```

```
NOT_EQUAL :
INC R3
DJNZ R1, outerloop

NOT_EQUALin :
INC R3
MOV R4, #00H
MOV A, R7
MOV R0, A
DJNZ R1, outerloop


endl:
SJMP endl

END
```

CODE EXPLANATION:

Initialization:

- The code initializes by subtracting 01H from 00H and storing the result (FFH) at 9500H. This serves as an initial value indicating "no match found" for the substring search.

- It then stores the substring "HU" at 9000H and the string "RAHUL" at 9200H.

Storing Strings:

- The substring and the main string are stored in memory character by character, with DPTR incremented after each character is stored.

Search Setup:

- Registers R1 and R7 are set to represent the lengths for the main comparison loop and the substring, respectively.

- Register R3 is used to iterate through "RAHUL", starting from the first character.

Outer Loop:

- The code compares each character in "RAHUL" with the first character of "HU" (Primary_comparison). If a match is found, it proceeds to a secondary comparison (Secondary_comparison) to check if the subsequent characters match the substring.

Inner Loop:

- If the first characters match, the secondary loop checks the rest of the substring. If all characters in "HU" match the corresponding characters in "RAHUL", the starting index of the match is stored at 9500H.

Continuation and Exit:

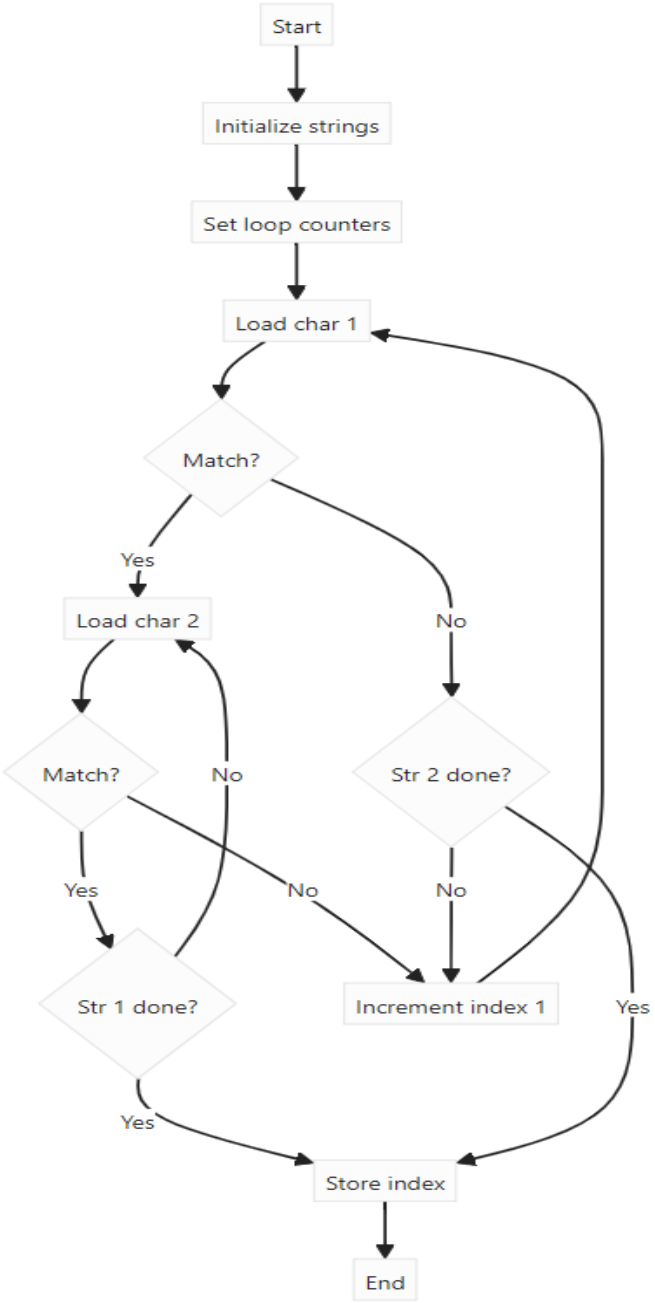
- If a match is not found in the secondary comparison, the primary loop continues with the next character in "RAHUL".

- The process repeats until either a match is found (and the index is stored) or all starting positions in "RAHUL" have been checked without finding a match.

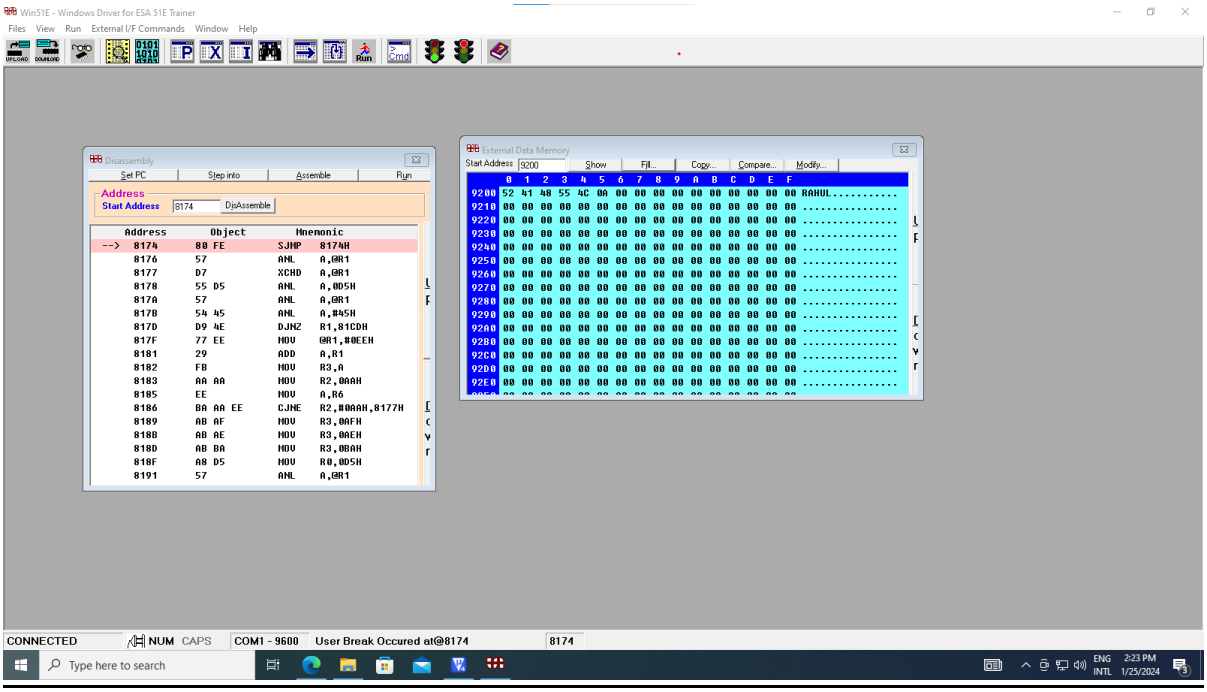
End of Program:

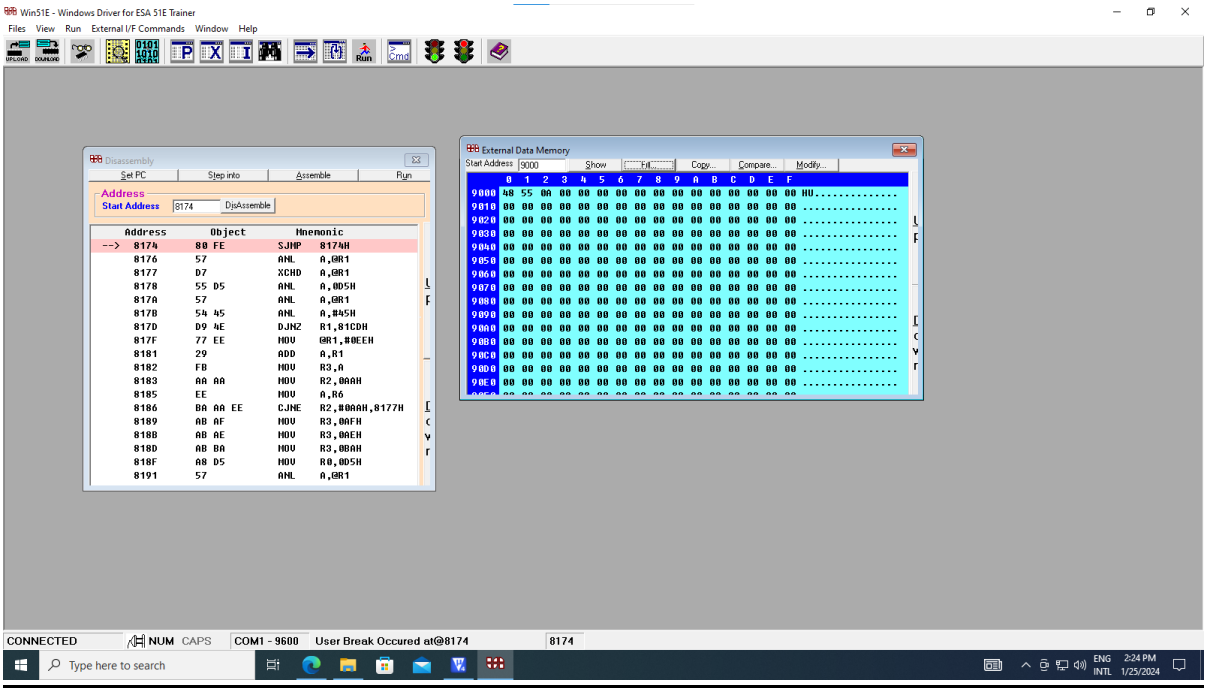
- The program ends with an infinite loop (endl:), signifying the end of the search operation.

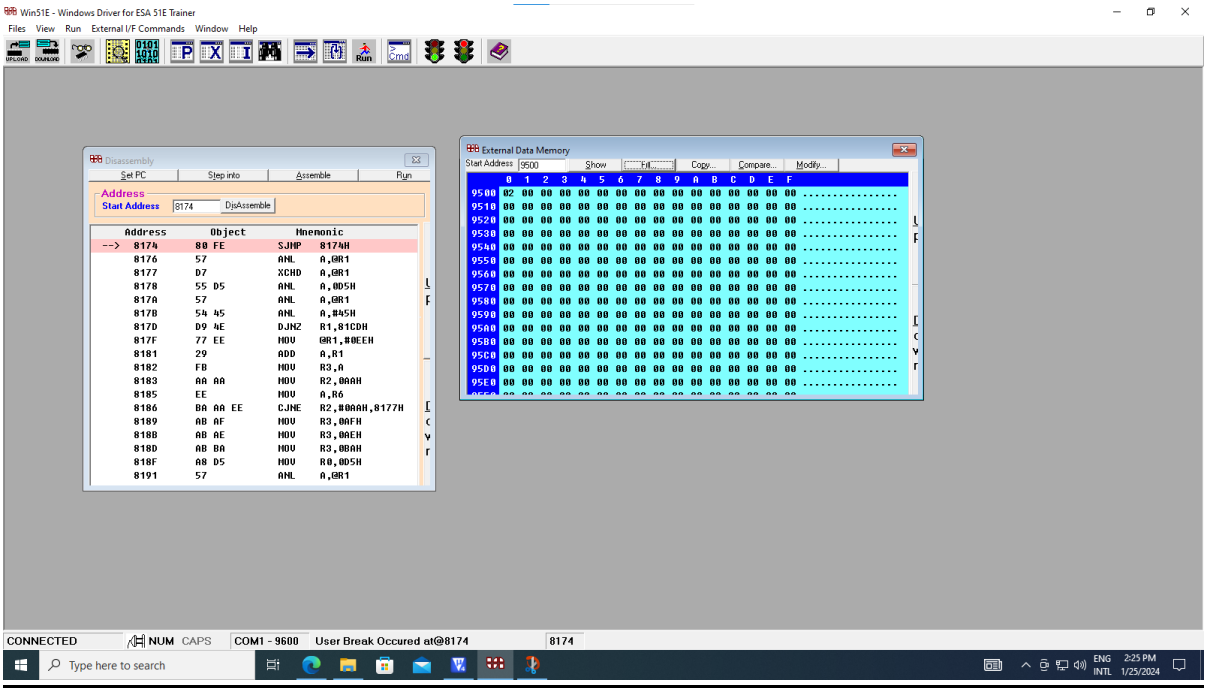
FLOWCHART:


Start
Initialize strings
Set loop counters
Load char 1
Match?
Yes
No
Load char 2
Match?
No
Str 2 done?
Yes
No
No
Str 1 done?
Increment index 1
Yes
Yes
Store index
End

RESULTS:


Win51E - Windows Driver for ESA 51E Trainer
Files View Run External I/F Commands Window Help
Disassembly
Set PC Step into Assemble Run
Address
Start Address 8174 DisAssemble
Address Object Mnemonic
--> 8174 80 FE SJMP 8174H
8176 57 ANL A,@R1
8177 D7 XCHD A,@R1
8178 55 D5 ANL A,0D5H
817A 57 ANL A,@R1
817B 54 45 ANL A,#45H
817D D9 4E DJNZ R1,81CDH
817F 77 EE MOV @R1,#0EEH
8181 29 ADD A,R1
8182 FB MOV R3,A
8183 AA AA MOV R2,0AAH
8185 EE MOV A,R6
8186 BA AA EE CJNE R2,#0AAH,8177H
8189 AB AF MOV R3,0AFH
818B AB AE MOV R3,0AEH
818D AB BA MOV R3,0BAH
818F AB D5 MOV R0,0D5H
8191 57 ANL A,@R1
External Data Memory
Start Address 9200 Show Fill... Copy... Compare... Modify...
0 1 2 3 4 5 6 7 8 9 A B C D E F
9200 52 41 48 55 4C 0A 00 00 00 00 00 00 00 00 00 00 RAHUL.....
9210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9250 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9280 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9290 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
92A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
92B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
92C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
92D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
92E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
CONNECTED NUM CAPS COM1 - 9600 User Break Occured at@8174 8174
Type here to search
ENG INTL 2:23 PM 1/25/2024


Win51E - Windows Driver for ESA 51E Trainer
Files View Run External I/F Commands Window Help
Disassembly
Set PC Step into Assemble Run
Address
Start Address 8174 DisAssemble
Address Object Mnemonic
--> 8174 80 FE SJMP 8174H
8176 57 ANL A,@R1
8177 D7 XCHD A,@R1
8178 55 D5 ANL A,0D5H
817A 57 ANL A,@R1
817B 54 45 ANL A,#45H
817D D9 4E DJNZ R1,81CDH
817F 77 EE MOV @R1,#0EEH
8181 29 ADD A,R1
8182 FB MOV R3,A
8183 AA AA MOV R2,0AAH
8185 EE MOV A,R6
8186 BA AA EE CJNE R2,#0AAH,8177H
8189 AB AF MOV R3,0AFH
818B AB AE MOV R3,0AEH
818D AB BA MOV R3,0BAH
818F AB D5 MOV R0,0D5H
8191 57 ANL A,@R1
External Data Memory
Start Address 9000 Show Fill... Copy... Compare... Modify...
0 1 2 3 4 5 6 7 8 9 A B C D E F
9000 48 55 0A 00 00 00 00 00 00 00 00 00 00 00 00 HU..............
9010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
9090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
90A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
90B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
90C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
90D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
90E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
CONNECTED NUM CAPS COM1 - 9600 User Break Occured at@8174 8174
Type here to search
ENG INTL 2:24 PM 1/25/2024



DISCUSSION:

Here is one way to rewrite the background and experiment steps using more advanced vocabulary:

Theoretical Background on the 8051 Microcontroller:

The 8051 microcontroller, developed by Intel in 1980, is an 8-bit microcontroller that contains several salient features:

- An 8-bit arithmetic logic unit (ALU) with 8-bit accumulator and data registers

- A 16-bit address bus, enabling access to 64KB of memory

- 4KB of on-chip read-only memory (ROM) to store immutable program instructions

- 128 bytes of on-chip random access memory (RAM) for dynamic data storage

- Four 8-bit I/O ports enabling interfacing with peripheral devices

- Two 16-bit timers/counters for timekeeping and event counting

- An integrated full-duplex serial interface for synchronous data transmission

- Interrupt and flag mechanisms for event handling

The Harvard architecture employed segregates program storage in ROM from data storage in RAM, facilitating simultaneous instruction fetch and execution. The provided 8051 kit

encompasses external RAM and ROM for capacious software and data, and plentiful ports to interface various peripherals like LCD displays and keypads.

Elaborate Experimental Methodology:

a) Sorting and Searching a Numeric Array

- A 100-byte array initialized with arbitrary numbers is situated in external RAM starting at address 9000H
- To sort the array, a rudimentary bubble sort algorithm will be implemented
- The algorithm iterates through the array, performing adjacent element comparisons and permutations to position elements in ascending order
- This process repeats until the array is fully sorted
- Subsequently, a binary search algorithm will locate the number stored at address 9500H
- If found, the index will be recorded at location 9550H, otherwise -1 denotes no match b)

Polynomial Evaluation

- Polynomial coefficients are stored in sequence starting at address 9502H
- The number of terms is specified at 9500H
- The substitution value is defined at 9501H
- The result variable is initialized to 0
- An iterative process computes each term, multiplying the coefficient by the value, adding to the result
- This computes n+1 iterations from degree n down to 0 - The final result is stored at 9550H

c) Substring Matching

- One string is stored starting at 9000H, null-terminated
- A second string is stored starting at 9200H, null-terminated
- Two index variables traverse the respective strings
- At each index, the characters are compared
- If all the characters match, both indexes are incremented until the end of the 1st string
- If the full 1st string matches, the starting index of the 2nd string is stored at 9500H
- Otherwise, -1 at 9500H indicates no match was found