EE 215 Microprocessors LAB #3

Student name：Li Xianzhe
Student ID number：2022214880

Description of Approach

1. Write a master program that calls two subroutines multiple times. First, the main program calls "max" to find the maximum value in the score list. Second, the main program repeatedly calls "histo" and builds a histogram in memory. Third, the main program calls "max" with the histogram address to find the pattern (the value that occurs the most often). To pass information between the main program and the subprogram, use registers. A common convention is to use registers R12 to R15 for parameter passing.

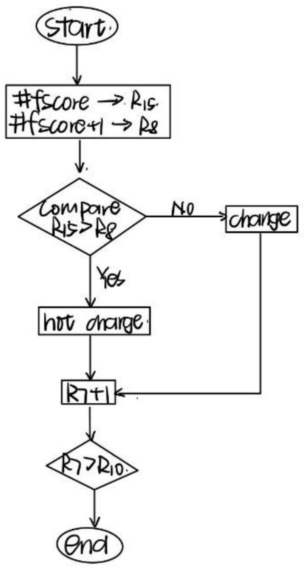
2. Write a subroutine named "max" and take the start address (R13) and end address (R14) as input and return the maximum number in that address range (R15).

3. Write a subroutine called "histo" and take the number (R12), the start address (R13), and the end address (R14) to find as input. The "histo" subroutine should return the number of times the number was found (appeared) in the address range (R15).

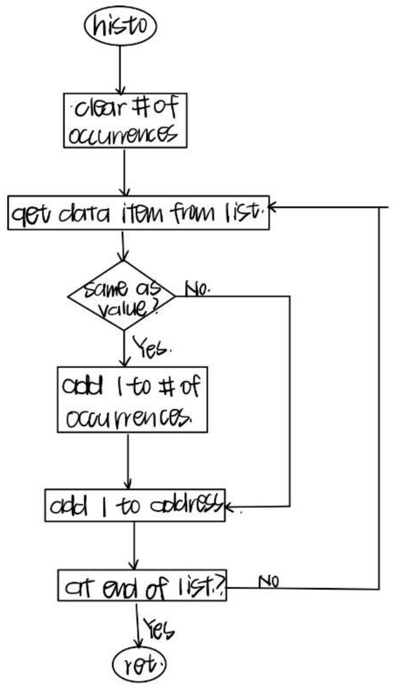
Before the main program, use the assembly directive to: a. Store the score in memory. What size and type of data is this? b. Reserve space for the histogram. What size and type of data is this? c. Reserve space for the highest score. What size and type of data is this? d. Reserve space for the mode. What size and type of data is this?

At the end of the program, the histogram, top score, and pattern should be in memory.

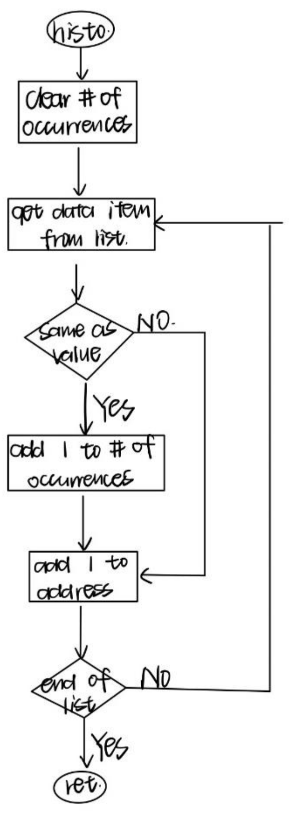
Flowchart


Start
#fscore → R15
#fscore+1 → R8
Compare R15 > R8
NO
change
Yes
not change
R7+1
R7 > R10
End

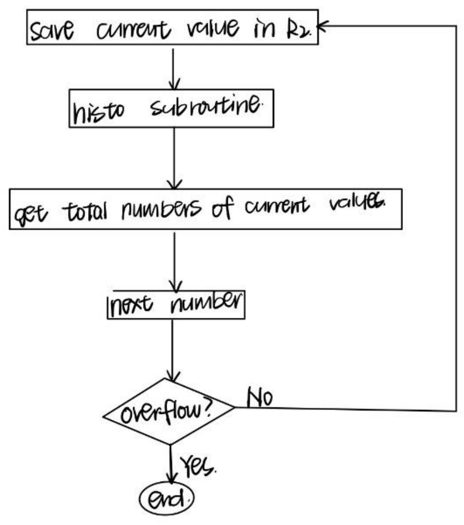
max


histo
clear # of occurrences
get data item from list
same as value?
No.
Yes.
add 1 to # of occurrences
add 1 to address
at end of list?
NO
Yes
ret

histo.



buildhistogram

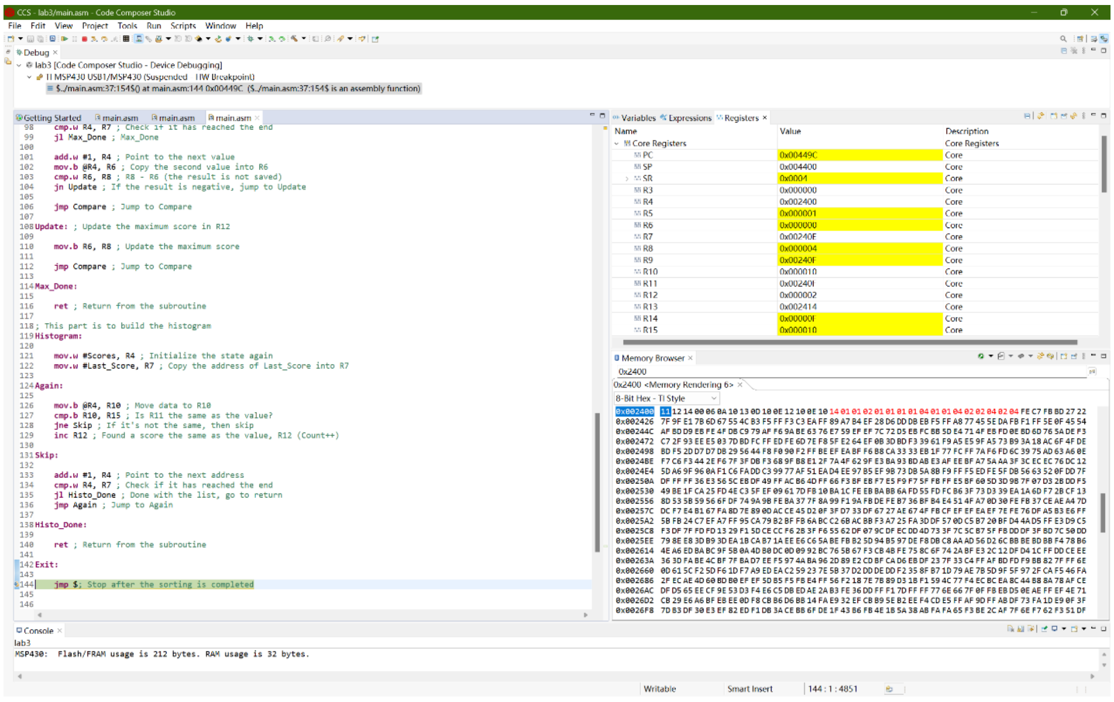

Save current value in R2.
histo subroutine.
get total numbers of current values.
next number
overflow?
No
Yes.
end.

Code

```
;-----------------------------------------------------------------------------
-----
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
```

```
;
;-------------------------------------------------------------------------
-----
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------
-----
            .def    RESET                  ; Export program entry-point to
                                           ; make it known to linker.
;-------------------------------------------------------------------------
-----
            .text                          ; Assemble into program memory.
            .retain                        ; Override ELF conditional linking
                                           ; and retain current section.
            .retainrefs                     ; And retain any sections that
have
                                           ; references to current section.

;-------------------------------------------------------------------------
-----
RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer


;-------------------------------------------------------------------------
-----
; Main loop here
;-------------------------------------------------------------------------
-----
     .data
Scores: .byte 17,18,20,0,6,10,16,19,13,16,14,18,16,14
Last_Score: .byte 16
Max_Score: .byte 0
Count_Score: .space 15
Mode: .byte 0

    .text
main: ; Main program

    ; Max part
    clr.b R4 ; Initialize the first address
    clr.b R5 ; Initialize the first comparison bit
    clr.b R6 ; Initialize the second comparison bit
    clr.b R7 ; Initialize the last address
    clr.b R8 ; Initialize the variable to store the maximum score
    clr.b R9 ; Initialize the last address++
    ; Histogram part
    clr.b R10 ; Temporary bit
    clr.b R11 ; Compare scores
    clr.b R12 ; Count bit
    clr.b R14 ; Count_Score
    clr.b R15 ; Compare bit

    mov.w #Scores, R4 ; Copy the first address of Scores into R4
    mov.w #Scores, R11 ; Histogram scores used for counting
    mov.w #Last_Score, R7 ; Copy the address of Last_Score into R7
    mov.w R7, R9 ; Copy the address of Last_Score into R9
    add.w #1, R9 ; Increment the Last address++

    ; Call Max part
    call #Max ; Find the maximum score and store it in R8
    mov.b R8, Max_Score ; Copy R8 (maximum score) into Max_Score

; Histogram_Build part
Histogram_Build:

    clr.b R12 ; Reset the Count bit
    call #Histogram ; Build the histogram

    mov.b R12, Count_Score(R14) ; Count the number of each score that has
occurred
    inc R14 ; Move to the next score's histogram
    mov.b @R11, R15 ; Move the data to R15
    add.w #1, R11 ; Point to the next score

    cmp.w R11, R9 ; Check if it has reached the end
    jne Histogram_Build ; Continue

; Mode_Find part
Mode_Find:

    mov.w #Count_Score, R4 ; Copy the first address of Count_Score into R4
    mov.w #Mode, R7 ; Copy the address of Mode into R7
    sub.w #1, R7 ; Now R7 is the last address of Count_Score
    call #Max ; Find the maximum Count_Score (mode) and store it in R8
    mov.b R8, Mode ; Copy R8 (maximum Count_Score mode) into Mode

    mov.w #Scores, R4 ; Initialize the state finally
    mov.w #Last_Score, R7 ; Copy the address of Last_Score into R7

    jmp Exit ; Once all tasks are finished, exit

; This part is to find the maximum score and store it in R12
Max: ; Initialize the state

    mov.b @R4, R5 ; Copy the first score into R5
    mov.w R5, R8 ; Store the maximum score

    jmp Compare ; Jump to the Compare section

Compare: ; Compare the maximum score with other scores

    cmp.w R4, R7 ; Check if it has reached the end
    jl Max_Done ; Max_Done

    add.w #1, R4 ; Point to the next value
    mov.b @R4, R6 ; Copy the second value into R6
    cmp.w R6, R8 ; R8 - R6 (the result is not saved)
    jn Update ; If the result is negative, jump to Update

    jmp Compare ; Jump to Compare

Update: ; Update the maximum score in R12

    mov.b R6, R8 ; Update the maximum score

    jmp Compare ; Jump to Compare

Max_Done:

    ret ; Return from the subroutine

; This part is to build the histogram
Histogram:

    mov.w #Scores, R4 ; Initialize the state again
    mov.w #Last_Score, R7 ; Copy the address of Last_Score into R7

Again:

    mov.b @R4, R10 ; Move data to R10
    cmp.b R10, R15 ; Is R11 the same as the value?
    jne Skip ; If it's not the same, then skip
    inc R12 ; Found a score the same as the value, R12 (Count++)

Skip:

    add.w #1, R4 ; Point to the next address
    cmp.w R4, R7 ; Check if it has reached the end
    jl Histo_Done ; Done with the list, go to return
    jmp Again ; Jump to Again

Histo_Done:

    ret ; Return from the subroutine

Exit:

    jmp $; Stop after the sorting is completed




;---------------------------------------------------------------------------
-----
; Stack Pointer definition
;---------------------------------------------------------------------------
-----
           .global __STACK_END
           .sect   .stack

;---------------------------------------------------------------------------
-----
; Interrupt Vectors
;---------------------------------------------------------------------------
-----
           .sect   ".reset"                ; MSP430 RESET Vector
           .short  RESET
```

Screen Snapshots(Results)


CCS - lab3/main.asm - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Debug
Getting Started
main.asm
main.asm
main.asm
Variables
Expressions
Registers
Memory Browser
Console

```
lab3 [Code Composer Studio - Device Debugging]
  TI MSP430 USB1/MSP430 (Suspended - HW Breakpoint)
    $../main.asm:37:154$() at main.asm:144 0x00449C  ($../main.asm:37:154$ is an assembly function)
```

```
    cmp.w R4, R7 ; Check if it has reached the end
    jl Max_Done ; Max_Done

    add.w #1, R4 ; Point to the next value
    mov.b @R4, R6 ; Copy the second value into R6
    cmp.w R6, R8 ; R8 - R6 (the result is not saved)
    jn Update ; If the result is negative, jump to Update

    jmp Compare ; Jump to Compare

Update: ; Update the maximum score in R12

    mov.b R6, R8 ; Update the maximum score

    jmp Compare ; Jump to Compare

Max_Done:

    ret ; Return from the subroutine

; This part is to build the histogram
Histogram:

    mov.w #Scores, R4 ; Initialize the state again
    mov.w #Last_Score, R7 ; Copy the address of Last_Score into R7

Again:

    mov.b @R4, R10 ; Move data to R10
    cmp.b R10, R15 ; Is R11 the same as the value?
    jne Skip ; If it's not the same, then skip
    inc R12 ; Found a score the same as the value, R12 (Count++)

Skip:

    add.w #1, R4 ; Point to the next address
    cmp.w R4, R7 ; Check if it has reached the end
    jl Histo_Done ; Done with the list, go to return
    jmp Again ; Jump to Again

Histo_Done:

    ret ; Return from the subroutine

Exit:

    jmp $; Stop after the sorting is completed
```

Name	Value	Description
Core Registers		Core Registers
PC	0x00449C	Core
SP	0x004400	Core
SR	0x0004	Core
R3	0x000000	Core
R4	0x002400	Core
R5	0x000001	Core
R6	0x000000	Core
R7	0x00240E	Core
R8	0x000004	Core
R9	0x00240F	Core
R10	0x000010	Core
R11	0x00240F	Core
R12	0x000002	Core
R13	0x002414	Core
R14	0x00000F	Core
R15	0x000010	Core

```
0x2400
0x2400 <Memory Rendering 6>
8-Bit Hex - TI Style
0x002400  11 12 14 00 06 0A 10 13 0D 10 0E 12 10 0E 10 14 01 01 02 01 01 01 01 04 01 01 04 02 02 04 02 04 FE C7 FB BD 27 22
0x002426  7F 9F E1 7B 6D 67 55 4C B3 F5 FF F3 C3 EA FF 89 A7 B4 EF 28 D6 DD DB EB F5 FF A8 77 45 5E DA FB F1 FF 5E 0F 45 54
0x00244C  AF BD D9 EB FE 4F DB C9 79 AF F6 9A BE 63 76 E7 59 EF EF 7C 72 D5 EB FC BB 5D E4 71 4F EB FD 0E BD 6D 76 5A DE F3
0x002472  C7 2F 93 EE E5 03 7D BD FC FF ED FE 6D 7E F8 5F E2 64 EF 0B 3D BD F3 39 61 F9 A5 E5 9F A5 73 B9 3A 18 AC 6F 4F DE
0x002498  BD F5 2D D7 D7 DB 29 56 44 F8 F0 90 F2 FF BE EF EA BF F6 B8 CA 33 33 EB 1F 77 FC FF 7A F6 FD 6C 39 75 AD 63 A6 0E
0x0024BE  F7 C6 F3 44 2E F6 7F 3F DB F3 68 9F B8 E1 2F 7A 4F 62 9F E3 BA 93 BD AB E3 AF EE BF A7 5A AA 3F 3C EC EC 76 DC 12
0x0024E4  5D A6 9F 96 0A F1 C6 FA DD C3 99 77 AF 51 EA D4 EE 97 B5 EF 9B 73 DB 5A 8B F9 FF F5 ED FE 5F DB 56 63 52 0F DD 7F
0x00250A  DF FF FF 36 E3 56 5C EB DF 49 FF AC B6 4D FF 66 F3 BF EB F7 E5 F9 F7 5F FB FF E5 BF 60 5D 3D 9B 7F 07 D3 2B DD F5
0x002530  49 BE 1F CA 25 FD 4E C3 5F EF 09 61 7D FB 10 BA 1C FE EB BA BB 6A FD 55 FD FC B6 3F 73 D3 39 EA 1A 6D F7 2B CF 13
0x002556  8D 53 5B 59 56 6F DF 74 9A 9B FE BA 37 7F 8A 99 F1 9A FB DE FE B7 36 BF B4 E4 51 4F A7 0D 30 FE FB 37 CE AE A4 7D
0x00257C  DC F7 E4 B1 67 FA 8D 7E 89 0D AC CE 45 D2 0F 3F D7 33 DF 67 27 AE 67 4F FB CF EF EF EA EF 7E FE 76 DF A5 B3 E6 FF
0x0025A2  5B FB 24 C7 EF A7 FF 95 CA 79 B2 BF FB 6A BC C2 6B AC BB F3 A7 25 FA 3D DF 57 0D C5 B7 20 BF D4 4A D5 FF E3 D9 C5
0x0025C8  F3 DF 7F FD FD 13 29 F1 5D CE CC F6 2B 3F F6 55 62 DF 07 9C DF EC DD 4D 73 3F 7C 5C B7 5F FB DD DF 3F BD 7C 50 DD
0x0025EE  79 8E E8 3D B9 3D EA 1B CA B7 1A EE E6 C6 5A BE FB B2 5D 94 B5 97 DE F8 DB C8 AA AD 56 D2 6C BB BE BD BB F4 78 B6
0x002614  4E A6 ED BA BC 9F 5B 0A 4D B0 DC 0D 09 92 BC 76 5B 67 F3 CB 4B FE 75 8C 6F 74 2A BF E3 2C 12 DF D4 1C FF DD CE EE
0x00263A  36 3D FA BE 4C BF 7F BA D7 EE F5 97 4A BA 96 2D 89 E2 CD BF CA D6 EB DF 23 7F 33 C4 FF AF BD FD F9 BB 82 7F FF 6E
0x002660  0D 61 5C F2 5D F6 1D F7 A9 ED EA C2 59 23 7E 5B 37 D2 DD DE DD F2 35 8F B7 1D 79 AE 7B 5D 9F 5F 97 2F CA F5 46 FA
0x002686  2F EC AE 4D 60 BD B0 EF EF 5D B5 F5 FB E4 FF 56 F2 18 7E 7B 89 D3 1B F1 59 4C 77 F4 EC BC EA 8C 44 B8 8A 78 AF CE
0x0026AC  DF D5 65 EE CF 9E 53 D3 F4 E6 C5 DB ED AE 2A B3 FE 36 DD FF F1 7D FF FF 77 6E 66 7F 0F FB EB D5 0E AE FF EF 4E 71
0x0026D2  CB 29 E6 A6 BF EB EE 0D F8 CB B6 D6 BB 14 FA E9 32 EF CB B9 5E B2 EE F4 CD E5 FF AF 9D FF AB DF 73 FA 1D E9 0F 3F
0x0026F8  7D B3 DF 30 E3 EF 82 ED F1 DB 3A CE BB 6F DE 1F 43 B6 FB 4E 1B 5A 38 AB FA FA 65 F3 BE 2C AF 7F 6E F7 62 F3 51 DF
```

```
lab3
MSP430:  Flash/FRAM usage is 212 bytes. RAM usage is 32 bytes.
```

Writable   Smart Insert   144 : 1 : 4851

Description of Results

Max can be used to find the maximum value of this set of numbers, and histo can be used to find the number of occurrences of a certain number (e.g., in my code, it's 17). Building a histogram allows you to make a histogram and find patterns at the same time.