EE 215 Microprocessors LAB #1

Student name：Li Xianzhe
Student ID number：2022214880

Description of Approach

First we need to define variables such as X, Y, Z1, Z2, and so on. Then you need to look up the instruction function used to achieve the experimental goal in the operation manual. Finally, write these instruction functions to build and run the program and debug it, then watch each step change in the register and memory. The experimental results are obtained.

Code

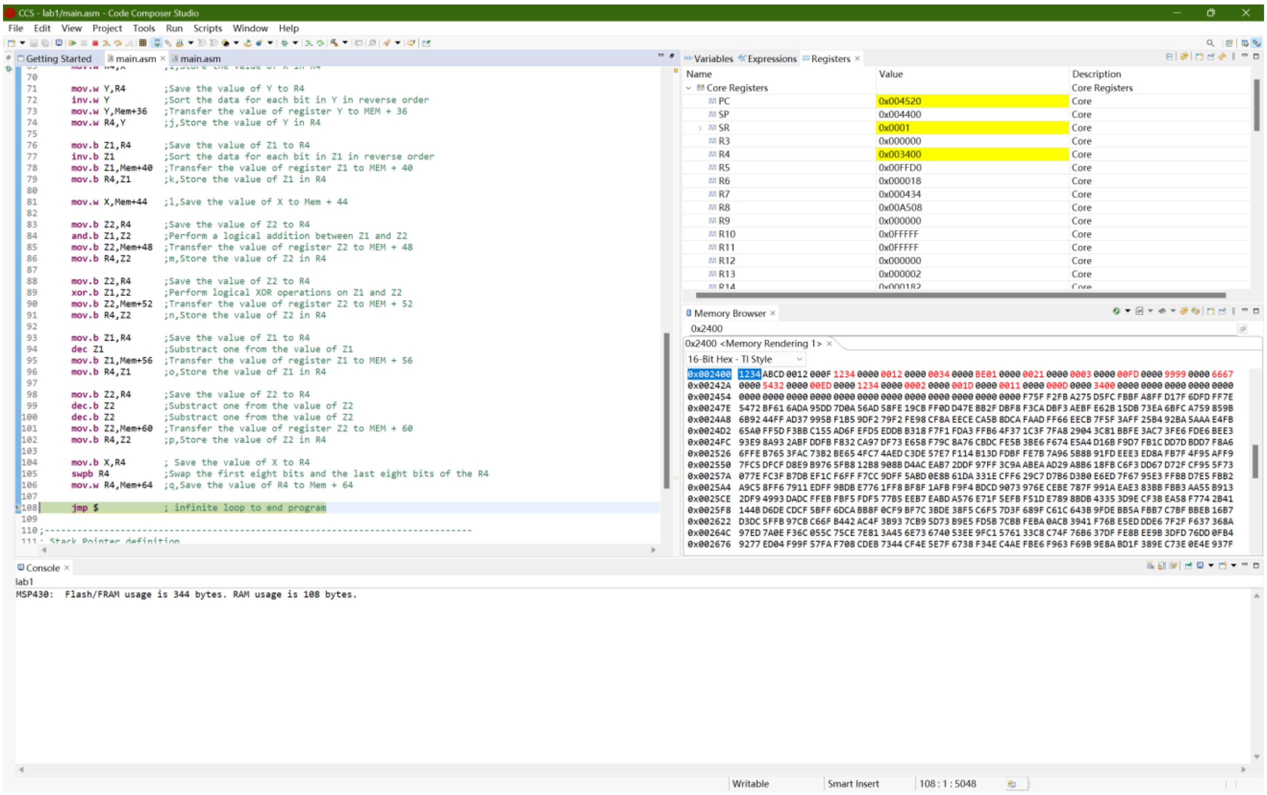
```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device 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
X:     .word 0x1234
Y:     .word 0xABCD
Z1:    .word 0x12
Z2:    .word 0x0F
Mem:   .space 100
```

```
     .text
     mov.w X,Mem  ;a

     swpb Mem        ;b,Swap the first eight bits and the last eight bits of the
Mem
     mov.b Mem,Mem+4  ;Transfer the value of register Mem to MEM + 4
     swpb Mem        ;Restore the Mem exchange

     mov.b X,Mem+8    ;c

     mov.w Y,R4       ;d,Save the value of Y to R4
     add.w X,Y        ;Perform a logical addition between X and Y
     mov.w Y,Mem+12   ;Transfer the value of register Y to MEM + 12
     mov.w R4,Y       ;Store the value of Y in R4

     mov.b Z2,R4      ;Save the value of Z2 to R4
     add.b Z1,Z2      ;Perform a logical addition between Z1 and Z2
     mov.b Z2,Mem+16  ;Transfer the value of register Z2 to MEM + 16
     mov.b R4,Z2      ;e,Store the value of Z2 in R4

     mov.b Z1,R4      ;Save the value of Z1 to R4
     sub.b Z2,Z1      ;Perform a logical subtraction between Z2 and Z1
     mov.b Z1,Mem+20  ;Transfer the value of register Z2 to MEM + 20
     mov.b R4,Z1      ;f,Store the value of Z1 in R4

     mov.b Z2,R4      ;Save the value of Z2 to R4
     sub.b Z1,Z2      ;Perform a logical subtraction between Z1 and Z2
     mov.b Z2,Mem+24  ;Transfer the value of register Z2 to MEM + 24
     mov.b R4,Z2      ;g,Store the value of Z2 in R4

     mov.w Y,R4       ;Save the value of Y to R4
     sub.w X,Y        ;Perform a logical subtraction between X and Y
     mov.w Y,Mem+28   ;Transfer the value of register Y to MEM + 28
     mov.w R4,Y       ;h,Store the value of Y in R4

     mov.w X,R4       ;Save the value of X to R4
     sub.w Y,X        ;Perform a logical subtraction between X and Y
     mov.w X,Mem+32   ;Transfer the value of register X to MEM + 32
     mov.w R4,X       ;i,Store the value of X in R4

     mov.w Y,R4       ;Save the value of Y to R4
     inv.w Y         ;Sort the data for each bit in Y in reverse order
     mov.w Y,Mem+36   ;Transfer the value of register Y to MEM + 36
     mov.w R4,Y       ;j,Store the value of Y in R4
```

```
      mov.b Z1,R4      ;Save the value of Z1 to R4
      inv.b Z1         ;Sort the data for each bit in Z1 in reverse order
      mov.b Z1,Mem+40  ;Transfer the value of register Z1 to MEM + 40
      mov.b R4,Z1      ;k,Store the value of Z1 in R4

      mov.w X,Mem+44   ;l,Save the value of X to Mem + 44

      mov.b Z2,R4      ;Save the value of Z2 to R4
      and.b Z1,Z2      ;Perform a logical addition between Z1 and Z2
      mov.b Z2,Mem+48  ;Transfer the value of register Z2 to MEM + 48
      mov.b R4,Z2      ;m,Store the value of Z2 in R4

      mov.b Z2,R4      ;Save the value of Z2 to R4
      xor.b Z1,Z2      ;Perform logical XOR operations on Z1 and Z2
      mov.b Z2,Mem+52  ;Transfer the value of register Z2 to MEM + 52
      mov.b R4,Z2      ;n,Store the value of Z2 in R4

      mov.b Z1,R4      ;Save the value of Z1 to R4
      dec Z1           ;Substract one from the value of Z1
      mov.b Z1,Mem+56  ;Transfer the value of register Z1 to MEM + 56
      mov.b R4,Z1      ;o,Store the value of Z1 in R4

      mov.b Z2,R4      ;Save the value of Z2 to R4
      dec.b Z2         ;Substract one from the value of Z2
      dec.b Z2         ;Substract one from the value of Z2
      mov.b Z2,Mem+60  ;Transfer the value of register Z2 to MEM + 60
      mov.b R4,Z2      ;p,Store the value of Z2 in R4

      mov.b X,R4       ; Save the value of X to R4
      swpb R4          ;Swap the first eight bits and the last eight bits of the R4
      mov.w R4,Mem+64  ;q,Save the value of R4 to Mem + 64

      jmp $            ; infinite loop to end program

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

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
```

```
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET
```

Results



```
        mov.w R4,X        ;i,Store the value of X in R4

        mov.w Y,R4        ;Save the value of Y to R4
        inv.w Y           ;Sort the data for each bit in Y in reverse order
        mov.w Y,Mem+36    ;Transfer the value of register Y to MEM + 36
        mov.w R4,Y        ;j,Store the value of Y in R4

        mov.b Z1,R4       ;Save the value of Z1 to R4
        inv.b Z1          ;Sort the data for each bit in Z1 in reverse order
        mov.b Z1,Mem+40   ;Transfer the value of register Z1 to MEM + 40
        mov.b R4,Z1       ;k,Store the value of Z1 in R4

        mov.w X,Mem+44    ;l,Save the value of X to Mem + 44

        mov.b Z2,R4       ;Save the value of Z2 to R4
        and.b Z1,Z2       ;Perform a logical addition between Z1 and Z2
        mov.b Z2,Mem+48   ;Transfer the value of register Z2 to MEM + 48
        mov.b R4,Z2       ;m,Store the value of Z2 in R4

        mov.b Z2,R4       ;Save the value of Z2 to R4
        xor.b Z1,Z2       ;Perform logical XOR operations on Z1 and Z2
        mov.b Z2,Mem+52   ;Transfer the value of register Z2 to MEM + 52
        mov.b R4,Z2       ;n,Store the value of Z2 in R4

        mov.b Z1,R4       ;Save the value of Z1 to R4
        dec Z1            ;Substract one from the value of Z1
        mov.b Z1,Mem+56   ;Transfer the value of register Z1 to MEM + 56
        mov.b R4,Z1       ;o,Store the value of Z1 in R4

        mov.b Z2,R4       ;Save the value of Z2 to R4
        dec.b Z2          ;Substract one from the value of Z2
        dec.b Z2          ;Substract one from the value of Z2
        mov.b Z2,Mem+60   ;Transfer the value of register Z2 to MEM + 60
        mov.b R4,Z2       ;p,Store the value of Z2 in R4

        mov.b X,R4        ; Save the value of X to R4
        swpb R4           ;Swap the first eight bits and the last eight bits of the R4
        mov.w R4,Mem+64   ;q,Save the value of R4 to Mem + 64

        jmp $             ; infinite loop to end program

;-------------------------------------------------------------------------------
; Stack Pointer definition
```

Name	Value	Description
Core Registers		Core Registers
PC	0x004520	Core
SP	0x004400	Core
SR	0x0001	Core
R3	0x000000	Core
R4	0x003400	Core
R5	0x00FFD0	Core
R6	0x000018	Core
R7	0x000434	Core
R8	0x00A508	Core
R9	0x000000	Core
R10	0x0FFFFF	Core
R11	0x0FFFFF	Core
R12	0x000000	Core
R13	0x000002	Core
R14	0x000182	Core

0x2400

0x2400 <Memory Rendering 1>

16-Bit Hex - TI Style

```
0x002400  1234 ABCD 0012 000F 1234 0000 0012 0000 0034 0000 BE01 0000 0021 0000 0003 0000 00FD 0000 9999 0000 6667
0x00242A  0000 5432 0000 00ED 0000 1234 0000 0002 0000 001D 0000 0011 0000 000D 0000 3400 0000 0000 0000 0000 0000
0x002454  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 F75F F2FB A275 D5FC FBBF A8FF D17F 6DFD FF7E
0x00247E  5472 BF61 6ADA 95DD 7D0A 56AD 58FE 19CB FF0D D47E BB2F DBF8 F3CA DBF3 AEBF E62B 15DB 73EA 6BFC A759 859B
0x0024A8  6B92 44FF AD37 995B F1B5 9DF2 79F2 FE98 CF8A EECE CA5B 8DCA FAAD FF66 EECB 7F5F 3AFF 25B4 92BA 5AAA E4FB
0x0024D2  65A0 FF5D F3BB C155 AD6F EFD5 EDDB B318 F7F1 FDA3 FFB6 4F37 1C3F 7FA8 2904 3C81 BBFE 3AC7 3FE6 FDE6 BEE3
0x0024FC  93E9 8A93 2ABF DDFB F832 CA97 DF73 E658 F79C 8A76 CBDC FE5B 3BE6 F674 E5A4 D16B F9D7 FB1C DD7D BDD7 F8A6
0x002526  6FFE B765 3FAC 73B2 BE65 4FC7 4AED C3DE 57E7 F114 B13D FDBF FE7B 7A96 5B8B 91FD EEE3 ED8A FB7F 4F95 AFF9
0x002550  7FC5 DFCF D8E9 B976 5FB8 12B8 908B D4AC EAB7 2DDF 97FF 3C9A ABEA AD29 A8B6 18FB C6F3 DD67 D72F CF95 5F73
0x00257A  077E FC3F B7DB EF1C F6FF F7CC 9DFF 5ABD 0E8B 61DA 331E CFF6 29C7 D7B6 D3B0 E6ED 7F67 95E3 FFBB D7E5 FBB2
0x0025A4  A9C5 8FF6 7911 EDFF 9BDB E776 1FF8 BF8F 1AFB F9F4 BDCD 9073 976E CEBE 787F 991A EAE3 83BB FBB3 AA55 B913
0x0025CE  2DF9 4993 DADC FFEB FBF5 FDF5 77B5 EEB7 EABD A576 E71F 5EFB F51D E789 8BDB 4335 3D9E CF3B EA58 F774 2B41
0x0025F8  144B D6DE CDCF 5BFF 6DCA BB8F 0CF9 BF7C 3BDE 38F5 C6F5 7D3F 689F C61C 643B 9FDE BB5A FBB7 C7BF BBEB 16B7
0x002622  D3DC 5FFB 97CB C66F B442 AC4F 3B93 7CB9 5D73 B9E5 FD5B 7CBB FEBA 0ACB 3941 F76B E5ED DDE6 7F2F F637 368A
0x00264C  97ED 7A0E F36C 055C 75CE 7E81 3A45 6E73 6740 53EE 9FC1 5761 33C8 C74F 76B6 37DF FE8B EE9B 3DFD 76DD 0FB4
0x002676  9277 ED04 F99F 57FA F708 CDEB 7344 CF4E 5E7F 6738 F34E C4AE FBE6 F963 F69B 9E8A BD1F 389E C73E 0E4E 937F
```

lab1

```
MSP430:  Flash/FRAM usage is 344 bytes. RAM usage is 108 bytes.
```

Description of Results

The starting bit of the msp430f5529 address is 0x2400, so the variables X, Y, Z1, Z2, and MEM(which takes up 100 Spaces) are stored in place. In this order we know the changes in registers and memory as each step occurs. The specified MEM will be assigned to the value specified by the program after each part of the actions a-q.