CECS 262

Lab 2: Viewing Registers and Memory with a Simulator

Last, First Name Khan, Umar

Last 4 Student ID: 7331.

Objectives:

- To get familiar with the software development tool Keil uVision, learn how to use Keil to edit, compile
- To examine the PSW register bits by continuing to use MOV and ADD instructions
- To exam the stack.

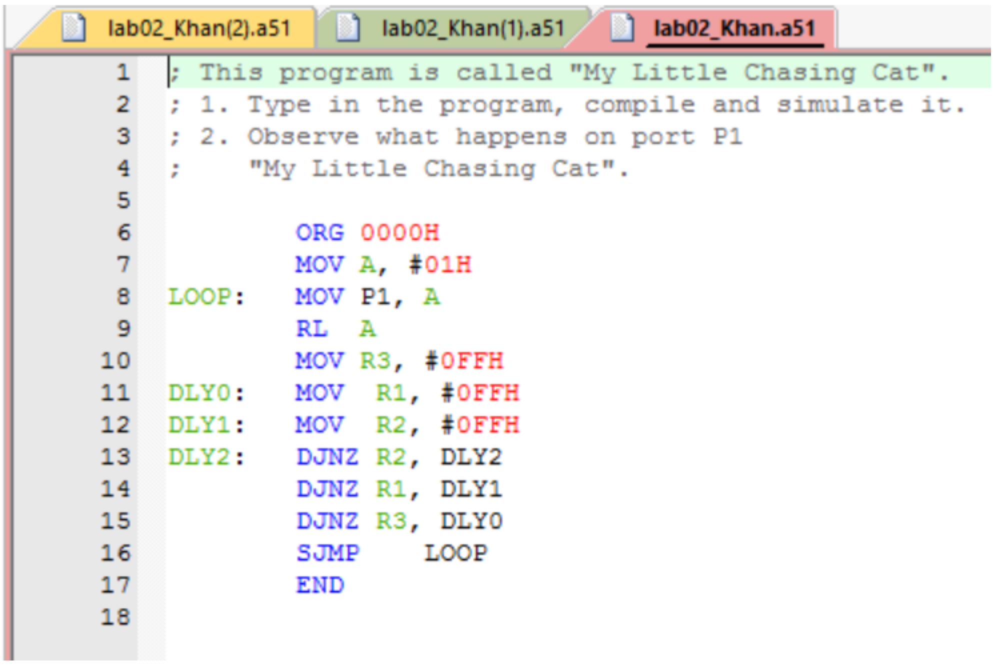
- **Tasks A:**

1. Follow the tutorial, Keil tutorial.pdf posted on Beachboard Labs folder; finish all the steps up to simulation. When you create your source file, use "My Little Chasing Cat" shown below. Run the program on the simulation and *demonstrate* it to the instructor.

Example: My Little Chasing Cat.

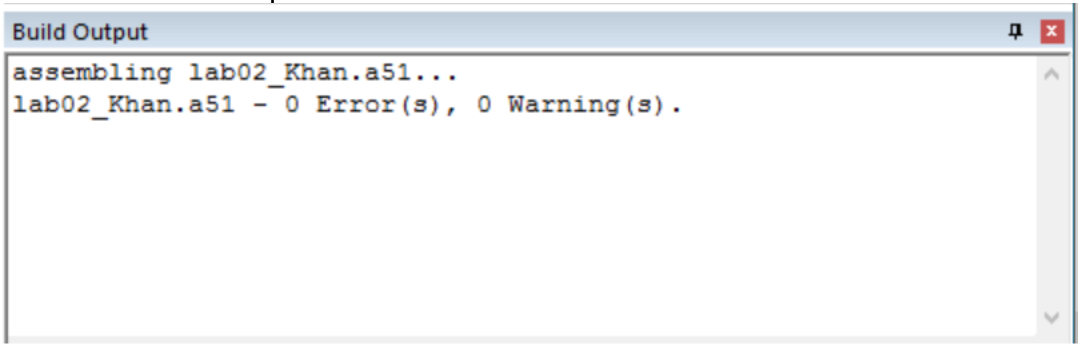
```
=================================================================
; This program is called "My Little Chasing Cat".
; 1. Type in the program, compile and simulate it.
; 2. Observe what happens on port P1
;    "My Little Chasing Cat".

            ORG    0000H
            MOV A, #01H
LOOP:        MOV P1, A
            RL  A
             MOV R3, #0FFH
DLY0:       MOV  R1, #0FFH
DLY1:       MOV  R2, #0FFH
DLY2:       DJNZ R2, DLY2
            DJNZ R1, DLY1
             DJNZ R3, DLY0
            SJMP  LOOP
            END
```

a. Source code (here is the sample code)



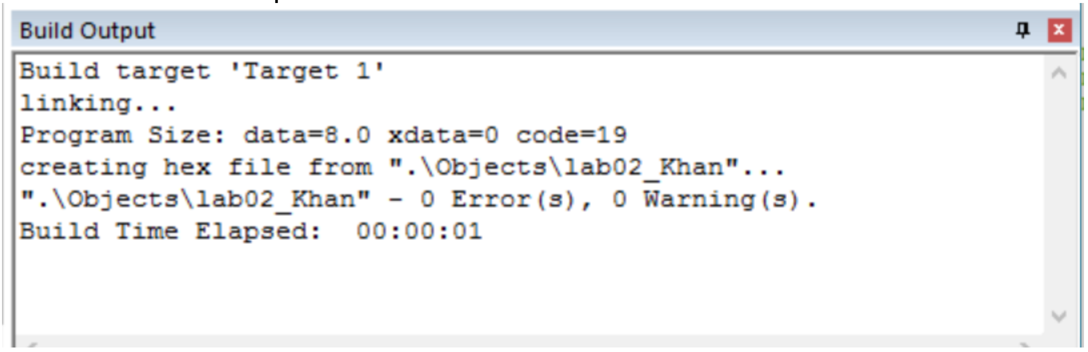
```
; This program is called "My Little Chasing Cat".
; 1. Type in the program, compile and simulate it.
; 2. Observe what happens on port P1
;     "My Little Chasing Cat".

        ORG 0000H
        MOV A, #01H
LOOP:   MOV P1, A
        RL  A
        MOV R3, #0FFH
DLY0:   MOV  R1, #0FFH
DLY1:   MOV  R2, #0FFH
DLY2:   DJNZ R2, DLY2
        DJNZ R1, DLY1
        DJNZ R3, DLY0
        SJMP    LOOP
        END
```

b. Build output after the Translate



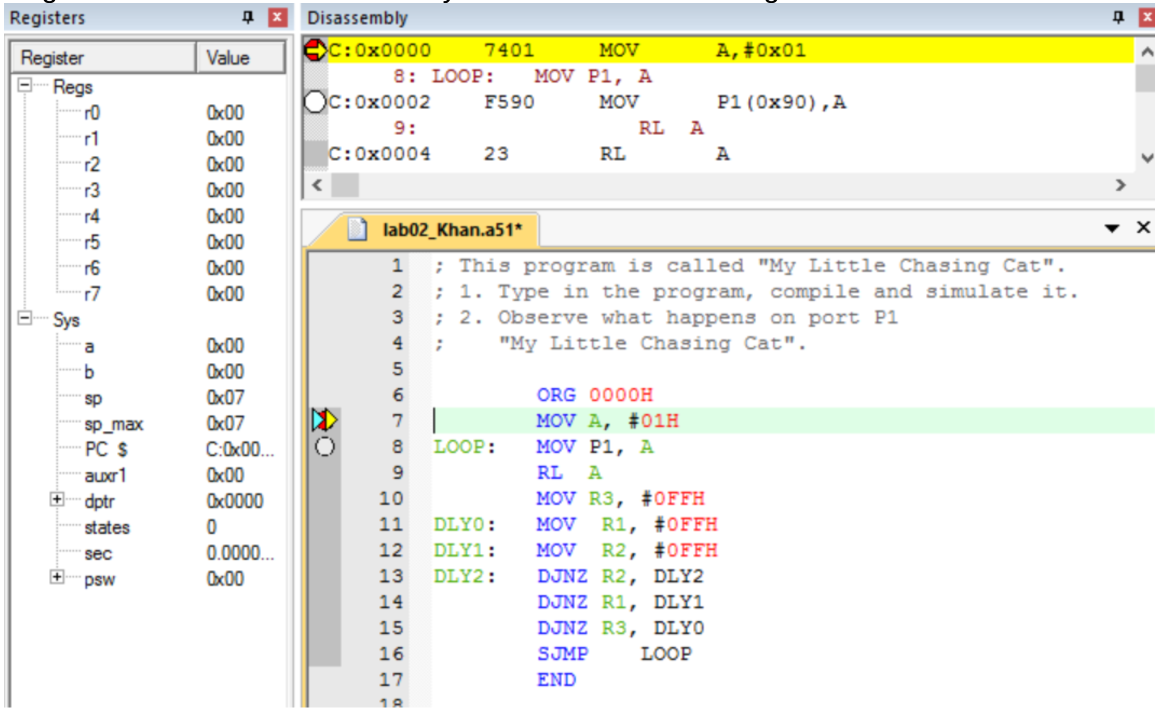
```
assembling lab02_Khan.a51...
lab02_Khan.a51 - 0 Error(s), 0 Warning(s).
```

c. Build output after the Build



```
Build target 'Target 1'
linking...
Program Size: data=8.0 xdata=0 code=19
creating hex file from ".\Objects\lab02_Khan"...
".\Objects\lab02_Khan" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:01
```

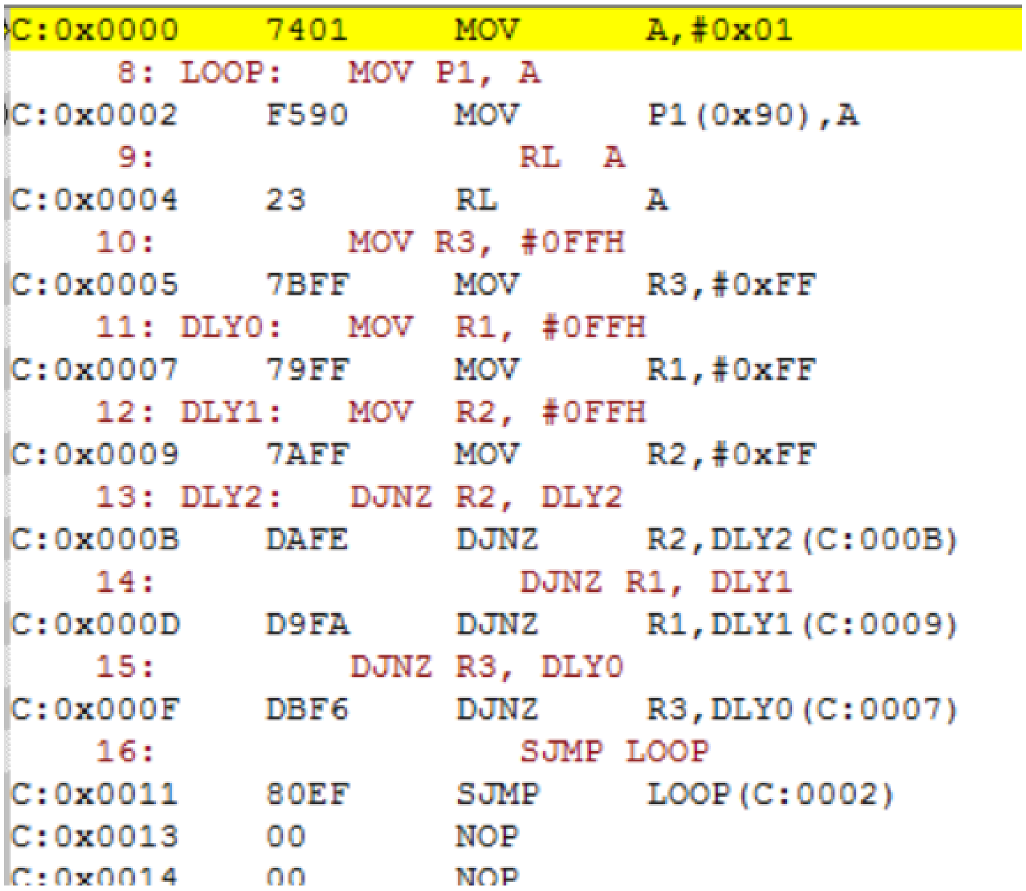
d. Register Window and Disassembly Window after the Debug



Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
PC $	C:0x00...
auxr1	0x00
dptr	0x0000
states	0
sec	0.0000...
psw	0x00

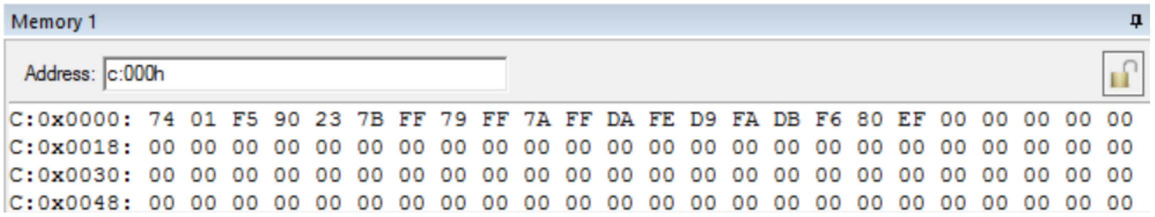
```
C:0x0000    7401     MOV      A,#0x01
     8: LOOP:   MOV P1, A
C:0x0002    F590     MOV      P1(0x90),A
     9:                 RL  A
C:0x0004    23       RL       A
```

```
; This program is called "My Little Chasing Cat".
; 1. Type in the program, compile and simulate it.
; 2. Observe what happens on port P1
;    "My Little Chasing Cat".

        ORG 0000H
        MOV A, #01H
LOOP:   MOV P1, A
        RL  A
        MOV R3, #0FFH
DLY0:   MOV  R1, #0FFH
DLY1:   MOV  R2, #0FFH
DLY2:   DJNZ R2, DLY2
        DJNZ R1, DLY1
        DJNZ R3, DLY0
        SJMP    LOOP
        END
```



```
C:0x0000    7401     MOV      A,#0x01
     8: LOOP:   MOV P1, A
C:0x0002    F590     MOV      P1(0x90),A
     9:                 RL  A
C:0x0004    23       RL       A
    10:         MOV R3, #0FFH
C:0x0005    7BFF     MOV      R3,#0xFF
    11: DLY0:   MOV  R1, #0FFH
C:0x0007    79FF     MOV      R1,#0xFF
    12: DLY1:   MOV  R2, #0FFH
C:0x0009    7AFF     MOV      R2,#0xFF
    13: DLY2:   DJNZ R2, DLY2
C:0x000B    DAFE     DJNZ     R2,DLY2(C:000B)
    14:                 DJNZ R1, DLY1
C:0x000D    D9FA     DJNZ     R1,DLY1(C:0009)
    15:         DJNZ R3, DLY0
C:0x000F    DBF6     DJNZ     R3,DLY0(C:0007)
    16:                 SJMP LOOP
C:0x0011    80EF     SJMP     LOOP(C:0002)
C:0x0013    00       NOP
C:0x0014    00       NOP
```

e. Memory Window



Memory 1

Address: c:000h

```
C:0x0000: 74 01 F5 90 23 7B FF 79 FF 7A FF DA FE D9 FA DB F6 80 EF 00 00 00 00 00
C:0x0018: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

==

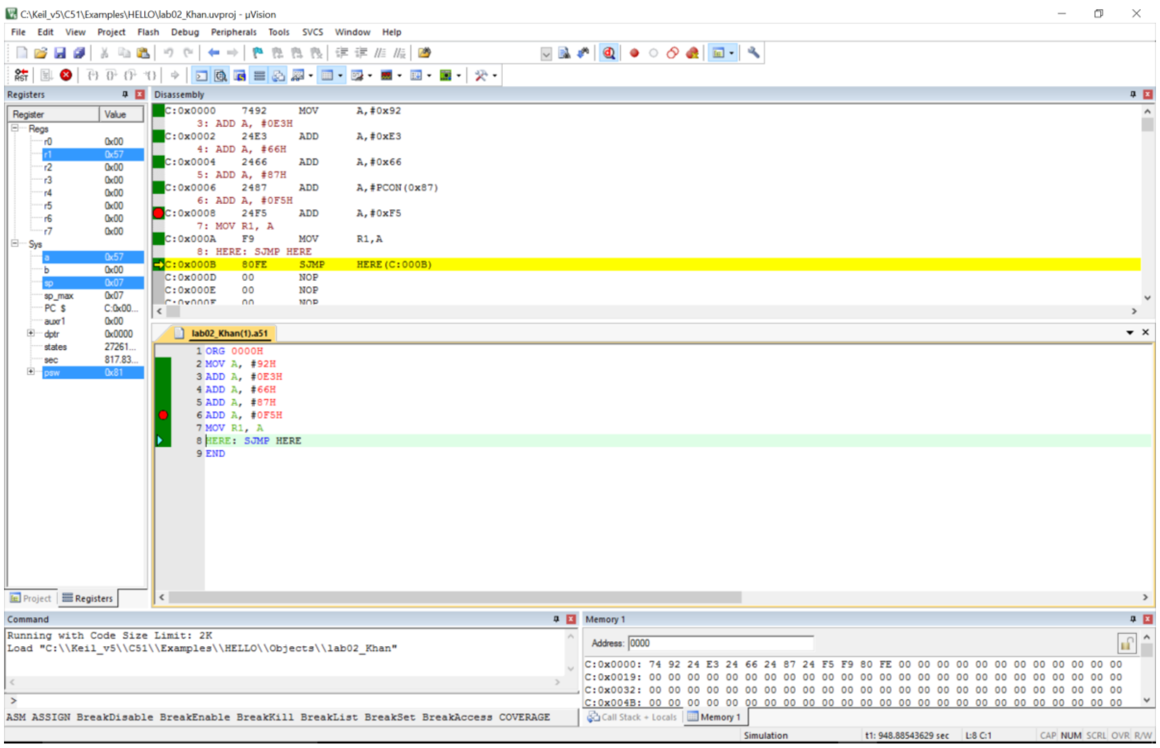
- **Task B:**

Write and assemble a program to add the following data and save the final result to register R1. Use the simulator to examine the CY/AC/P flag. Screen-shot whenever the addition generates a carry.
92H, E3H, 66H, 87H, F5H

```
ORG 0000H
Xxxxx


xxxxx
HERE: SJMP HERE
END
```

You need to:

i. Show the values for CY/AC/P after detailed manual addition for each step
ii. Include related screenshot of the register window for each step and compare with your manual calculation


C:\Keil_v5\C51\Examples\HELLO\lab02_Khan.uvproj - µVision

Register	Value
Regs	
r0	0x00
r1	0x57
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x57
b	0x00
sp	0x07
sp_max	0x07
PC $	C:0x00...
auxr1	0x00
dptr	0x0000
states	27261...
sec	817.83...
psw	0x81

Disassembly
```
C:0x0000    7492     MOV      A,#0x92
     3: ADD A, #0E3H
C:0x0002    24E3     ADD      A,#0xE3
     4: ADD A, #66H
C:0x0004    2466     ADD      A,#0x66
     5: ADD A, #87H
C:0x0006    2487     ADD      A,#PCON(0x87)
     6: ADD A, #0F5H
C:0x0008    24F5     ADD      A,#0xF5
     7: MOV R1, A
C:0x000A    F9       MOV      R1,A
     8: HERE: SJMP HERE
C:0x000B    80FE     SJMP     HERE(C:000B)
C:0x000D    00       NOP
C:0x000E    00       NOP
C:0x000F    00       NOP
```

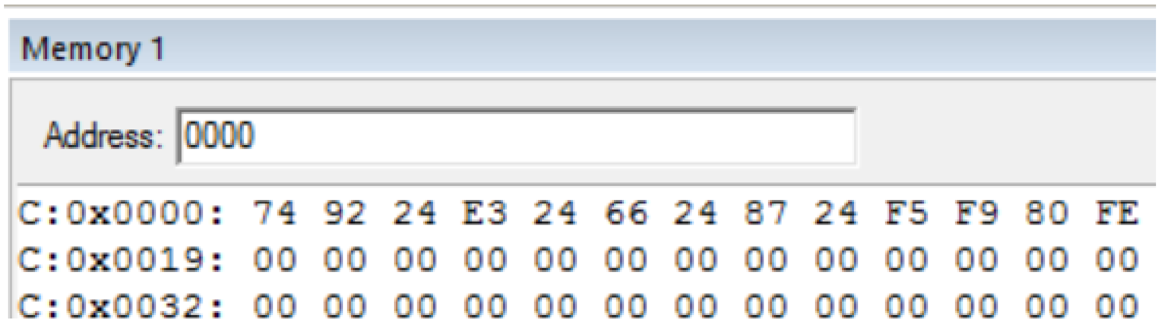
lab02_Khan(1).a51
```
ORG 0000H
MOV A, #92H
ADD A, #0E3H
ADD A, #66H
ADD A, #87H
ADD A, #0F5H
MOV R1, A
HERE: SJMP HERE
END
```

Command
```
Running with Code Size Limit: 2K
Load "C:\\Keil_v5\\C51\\Examples\\HELLO\\Objects\\lab02_Khan"
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

Memory 1
Address: 0000
```
C:0x0000: 74 92 24 E3 24 66 24 87 24 F5 F9 80 FE 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0019: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0032: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x004B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```
Simulation   t1: 948.88543629 sec   L:8 C:1

For example, here presents the register window the executing ADD A, #0F5H.

iii. Show the memory window with all involved machine code


Memory 1
Address: 0000
```
C:0x0000: 74 92 24 E3 24 66 24 87 24 F5 F9 80 FE
C:0x0019: 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0032: 00 00 00 00 00 00 00 00 00 00 00 00 00
```

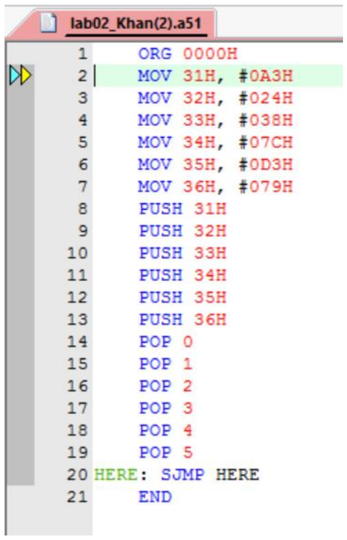
- **Task C**

Write and assemble a program to:

- MOV the following 6 different values to internal RAM locations: 31H - 36H,
- then PUSH the contents in those memory locations to the stack, assuming the initial value of SP is 7. When finish the above operation, screenshot the contents in the stack and contents in RAM location 31H – 36H. Data to be moved to 31H – 36H: A3H, 24H, 38H, 7CH, D3H, 79H
- POP the top 6 items from the stack into registers R0 – R5.

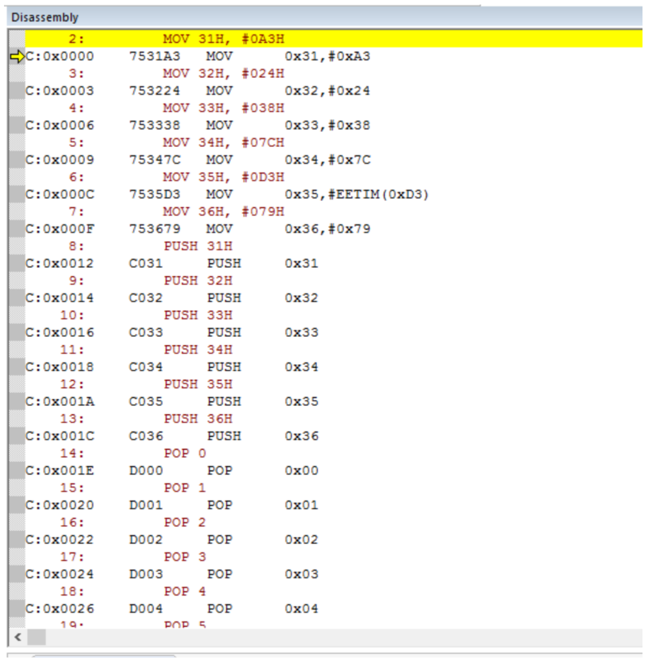
- Use the simulator to single-step and examine the registers, the stack, and the stack register. After poping all 6 items, screen-shot the contents in registers R0 – R5, the stack, and SP register.

```
          ORG 0000H
          MOV 31H,#0A3H
          xxx
          PUSH 31H
          xxx
          POP 0
          xxx
HERE: SJMP HERE
          END
```

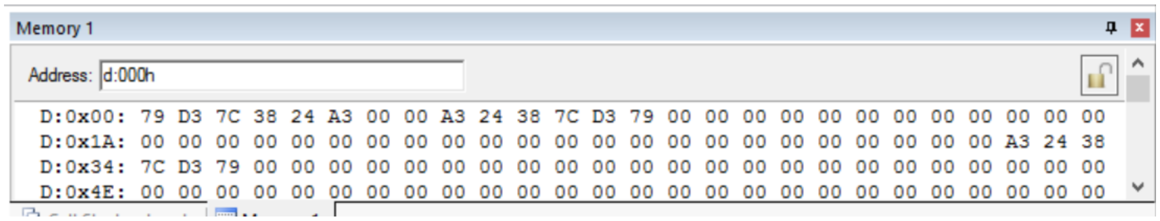


lab02_Khan(2).a51

```
      ORG  0000H
      MOV  31H, #0A3H
      MOV  32H, #024H
      MOV  33H, #038H
      MOV  34H, #07CH
      MOV  35H, #0D3H
      MOV  36H, #079H
      PUSH 31H
      PUSH 32H
      PUSH 33H
      PUSH 34H
      PUSH 35H
      PUSH 36H
      POP  0
      POP  1
      POP  2
      POP  3
      POP  4
      POP  5
HERE: SJMP HERE
      END
```



```
      2:          MOV 31H, #0A3H
C:0x0000    7531A3   MOV      0x31,#0xA3
      3:          MOV 32H, #024H
C:0x0003    753224   MOV      0x32,#0x24
      4:          MOV 33H, #038H
C:0x0006    753338   MOV      0x33,#0x38
      5:          MOV 34H, #07CH
C:0x0009    75347C   MOV      0x34,#0x7C
      6:          MOV 35H, #0D3H
C:0x000C    7535D3   MOV      0x35,#EETIM(0xD3)
      7:          MOV 36H, #079H
C:0x000F    753679   MOV      0x36,#0x79
      8:          PUSH 31H
C:0x0012    C031     PUSH     0x31
      9:          PUSH 32H
C:0x0014    C032     PUSH     0x32
     10:          PUSH 33H
C:0x0016    C033     PUSH     0x33
     11:          PUSH 34H
C:0x0018    C034     PUSH     0x34
     12:          PUSH 35H
C:0x001A    C035     PUSH     0x35
     13:          PUSH 36H
C:0x001C    C036     PUSH     0x36
     14:          POP 0
C:0x001E    D000     POP      0x00
     15:          POP 1
C:0x0020    D001     POP      0x01
     16:          POP 2
C:0x0022    D002     POP      0x02
     17:          POP 3
C:0x0024    D003     POP      0x03
     18:          POP 4
C:0x0026    D004     POP      0x04
     19:          POP 5
```



Address: d:000h

```
D:0x00: 79 D3 7C 38 24 A3 00 00 A3 24 38 7C D3 79 00 00 00 00 00 00 00 00 00 00 00 00
D:0x1A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 A3 24 38
D:0x34: 7C D3 79 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x4E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```