21BDS0340

Abhinav Dinesh Srivatsa

Microprocessors and Microcontrollers Lab

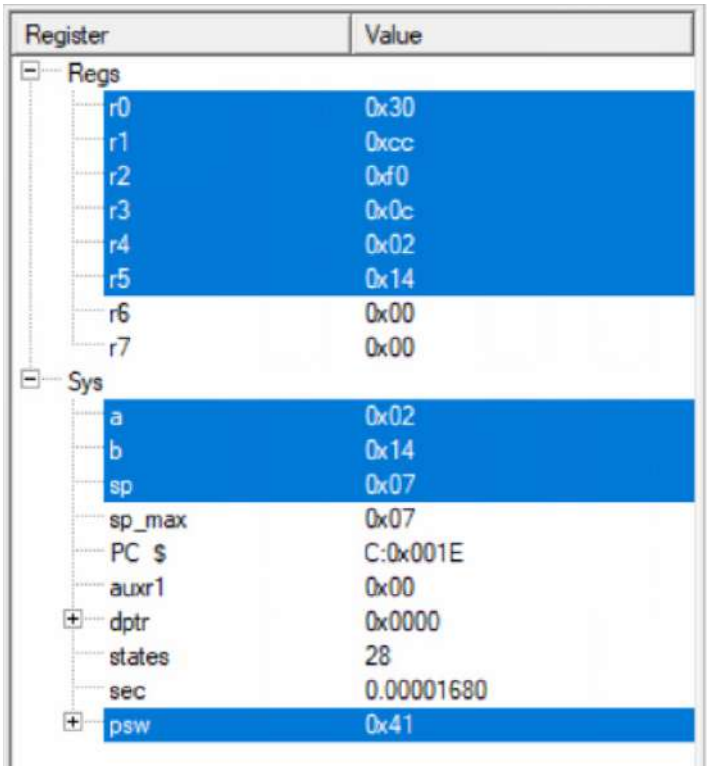
Lab Task – I

Question 1

Code:

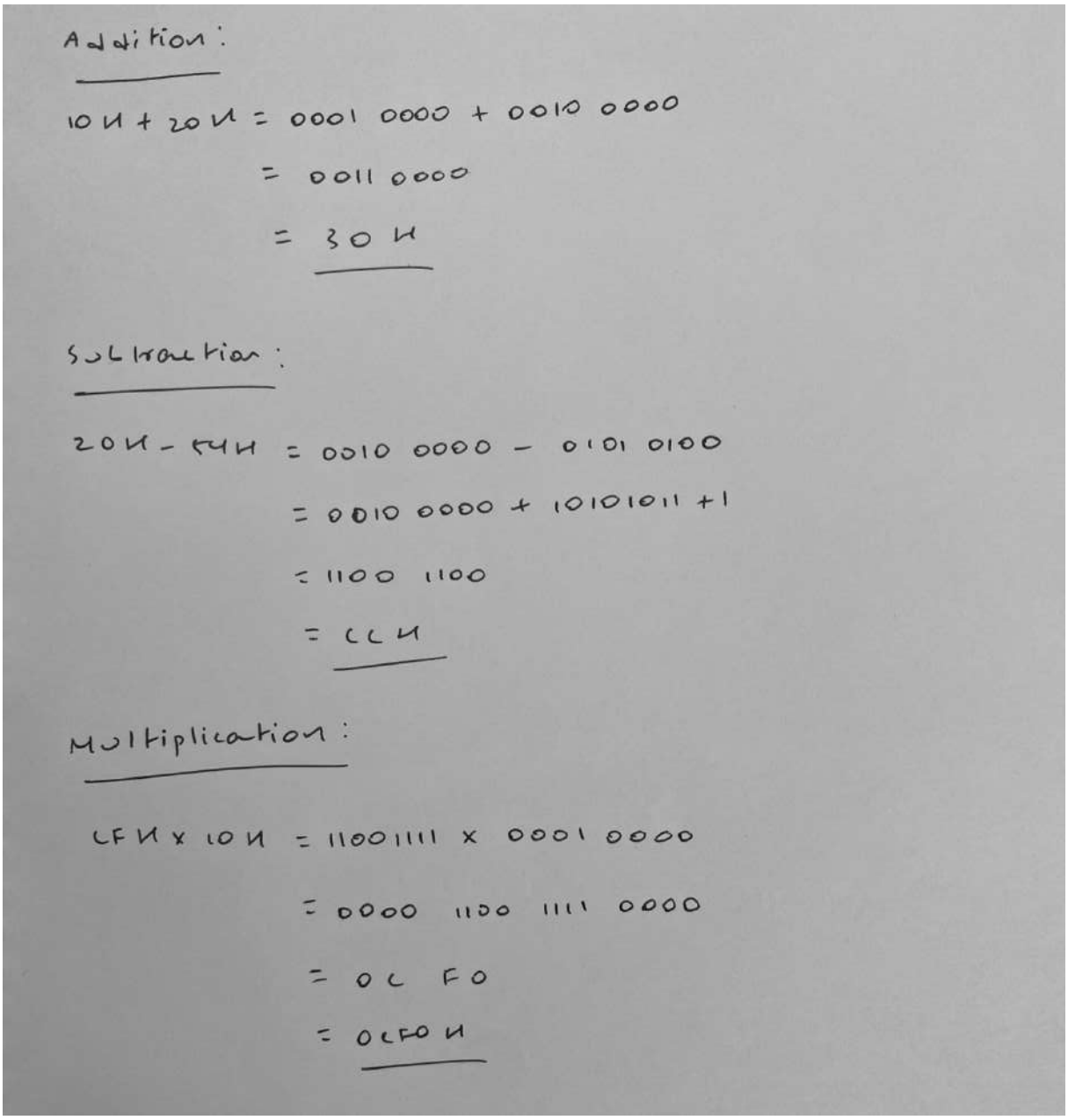
```
; addition
MOV A, #10H ; A = 10H
MOV R0, #20H ; A = 20H
ADD A, R0 ; A = A + R0 = 30H
MOV R0, A ; addition result stored in R0

; subtraction
MOV A, #20H ; A = 20H
MOV R1, #54H ; R1 = 54H
SUBB A, R1 ; A = A – R1 = –34H = CCH in 2's complement
MOV R1, A ; subtraction result stored in R1

; multiplication
MOV A, #0CFH ; A = CFH
MOV B, #10H ; R2 = 10H
MUL AB ; AB = A x B = CF0H
MOV R2, A ; MSB's of AB stored in R2
MOV R3, B ; LSB's of AB stored in R3

; division
MOV A, #54H ; A = 54H
MOV B, #20H ; B = 20H
DIV AB ; AB = A / B = Quotient – 2H, Remainder – 14H
MOV R4, A ; R4 = Quotient of A / B
MOV R5, B ; R5 = Remainder of A / B

END
```

Output:



Register	Value
Regs	
r0	0x30
r1	0xcc
r2	0xf0
r3	0x0c
r4	0x02
r5	0x14
r6	0x00
r7	0x00
Sys	
a	0x02
b	0x14
sp	0x07
sp_max	0x07
PC $	C:0x001E
auxr1	0x00
dptr	0x0000
states	28
sec	0.00001680
psw	0x41

Manual Calculations:



Division:

54H / 20H = 01010100 / 00100000

= 100000) 10
1010100
100000
010100
000000
10100

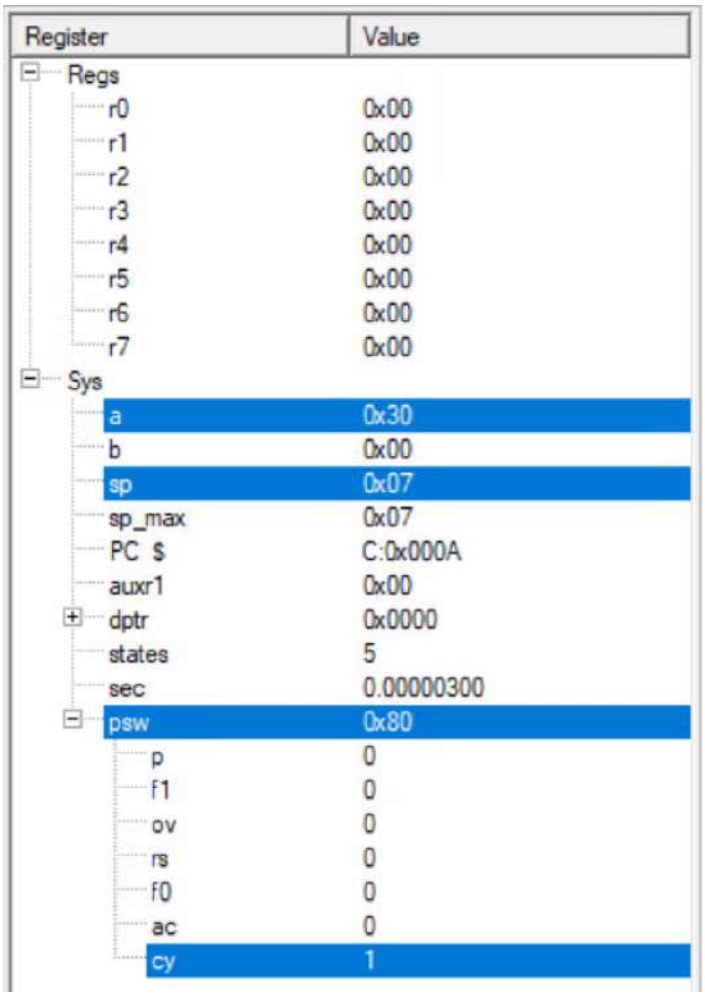
= Quotient = 2H

Remainder = 14H

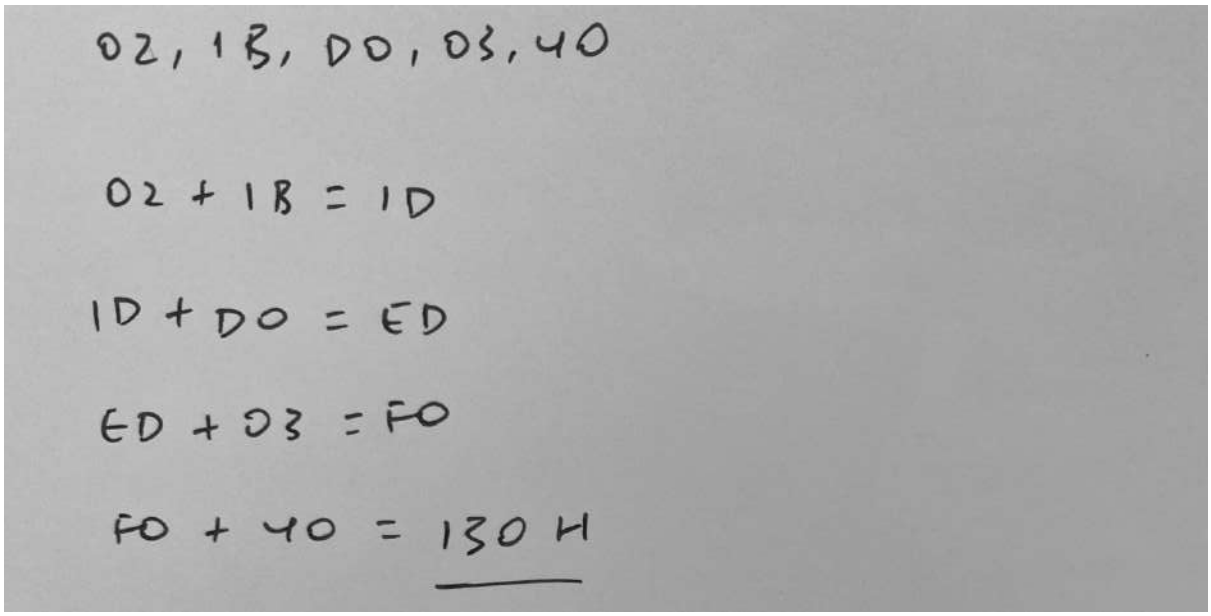
Question 2

```
; registration number = 21BDS0340
; grouped as 02 1B D0 03 40 (S swapped for 0)
MOV A, #02H ; A = 02H

ADD A, #1BH ; A = 1DH

ADD A, #0D0H ; A = EDH

ADD A, #03H ; A = F0H

ADD A, #40H ; A = 130H, but too many bits, A = 30H, CY = 1

END
```

Output:



Manual Calculations:

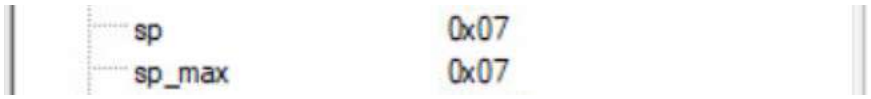


Question 3

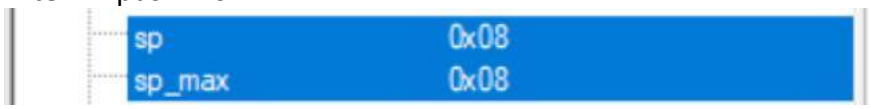
Code:

```
; registration number = 21BDS0340 (ignoring S)
; values = 2, 1, B, D, 0, 3, 4, 0
; moving values into bank 0
MOV R0, #02H
MOV R1, #01H
MOV R2, #0BH
MOV R3, #0DH
MOV R4, #00H
MOV R5, #03H
MOV R6, #04H
MOV R7, #00H

; pushing into stack, bank 1
PUSH 00H
PUSH 01H
PUSH 02H
PUSH 03H
PUSH 04H
PUSH 05H
PUSH 06H
PUSH 07H

END
```

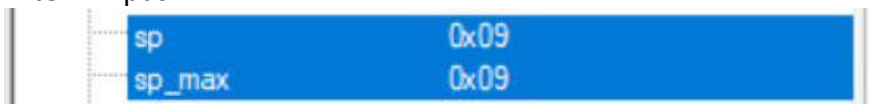
Output:



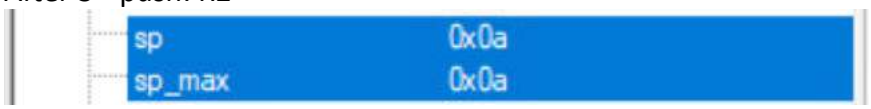
After 1st push: R0



After 2nd push: R1

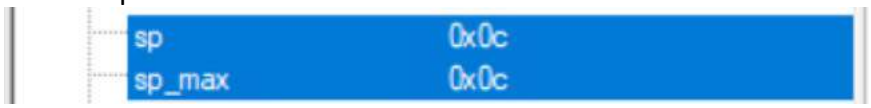


After 3rd push: R2

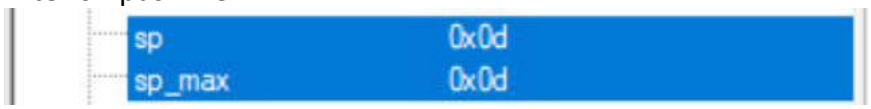


After 4th push: R3

After 5th push: R4



After 6th push: R5



After 7th push: R7

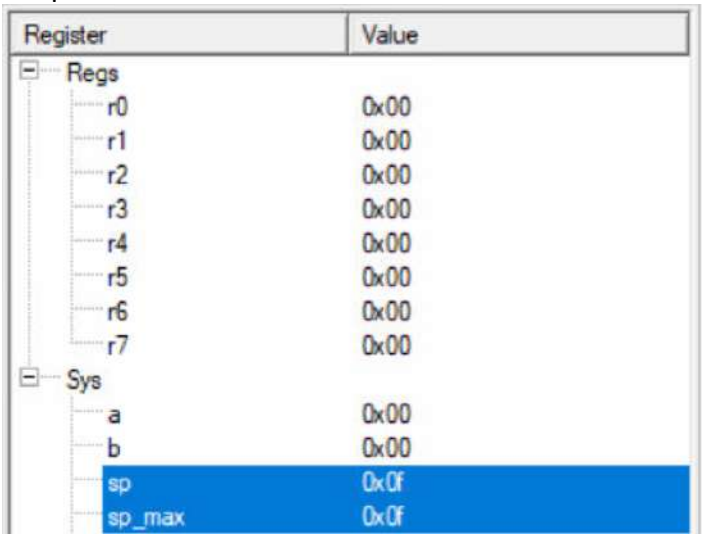
After 8th push: R8

Question 4

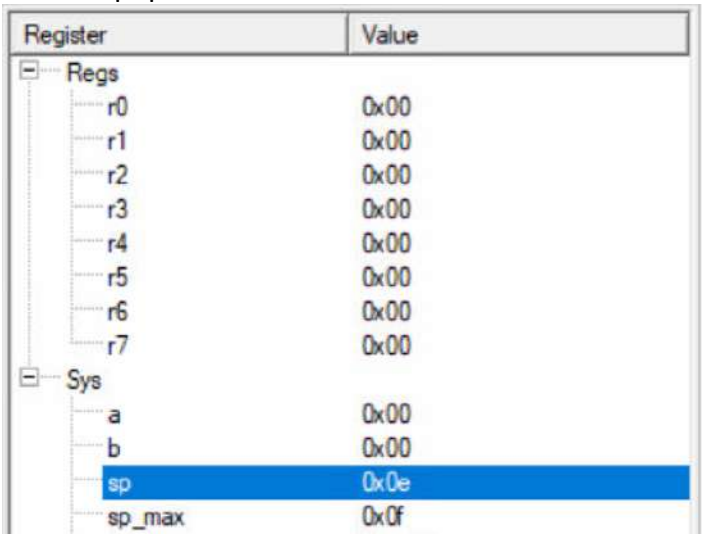
Code:

```
; registration number = 21BDS0340 (ignoring S)
; values = 2, 1, B, D, 0, 3, 4, 0
; moving values into bank 1, stack
MOV 08H, #02H
MOV 09H, #01H
MOV 0AH, #0BH
MOV 0BH, #0DH
MOV 0CH, #00H
MOV 0DH, #03H
MOV 0EH, #04H
MOV 0FH, #00H

; settings stack pointer to 0FH
MOV SP, #0FH

; popping values into bank 0, current bank
POP 07H
POP 06H
POP 05H
POP 04H
POP 03H
POP 02H
POP 01H
POP 00H

END
```

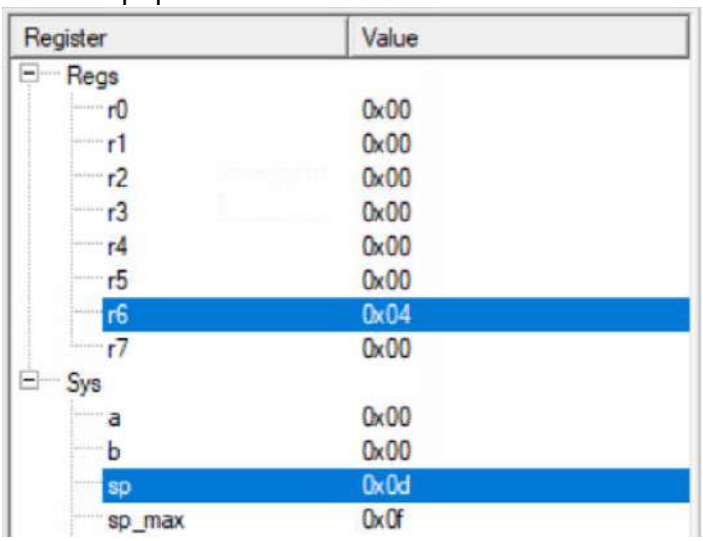
Output:



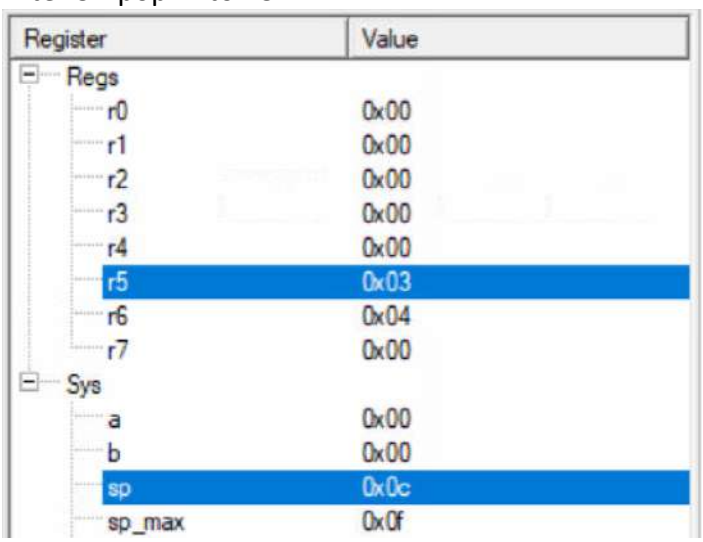
After 1st pop: into R7



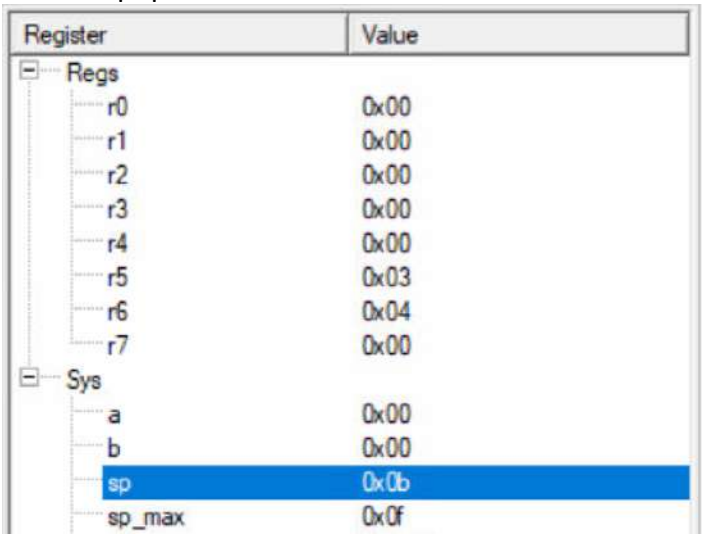
After 2nd pop: into R6



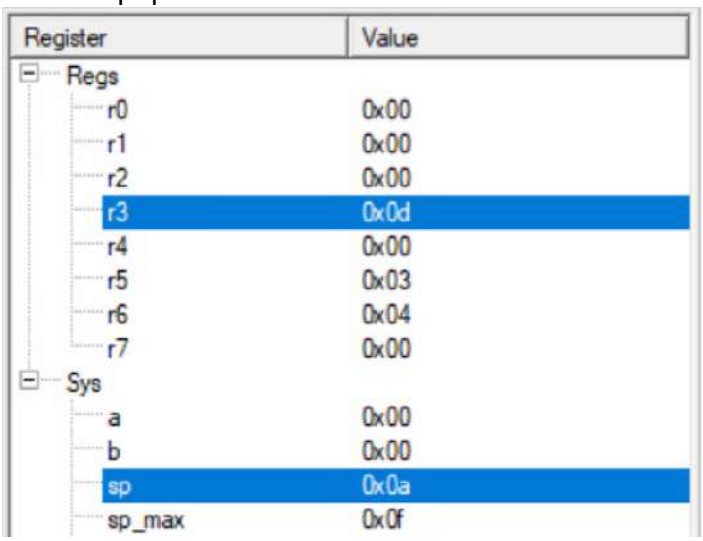
After 3rd pop: into R5



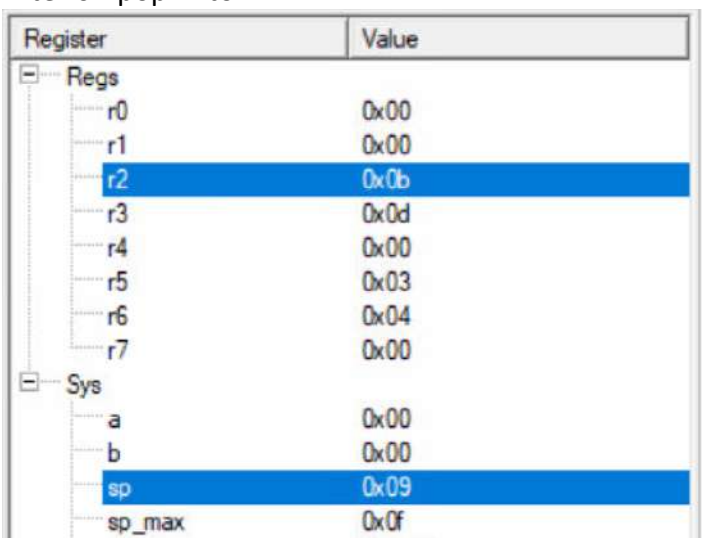
After 4th pop: into R4



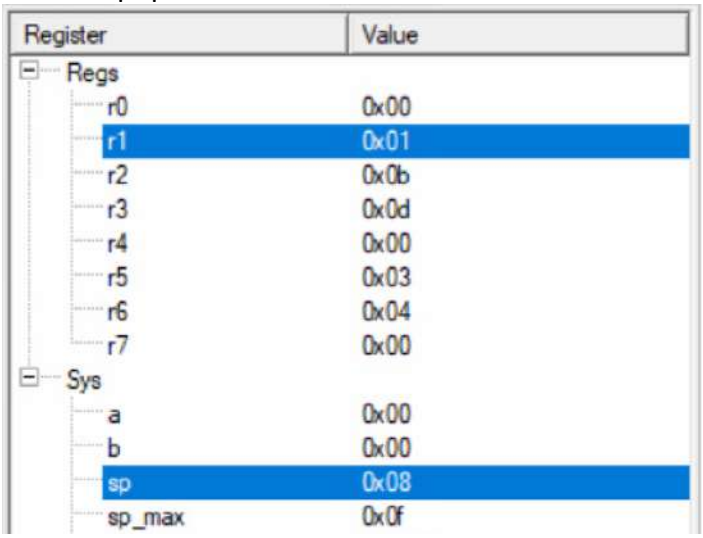
After 5th pop: into R3



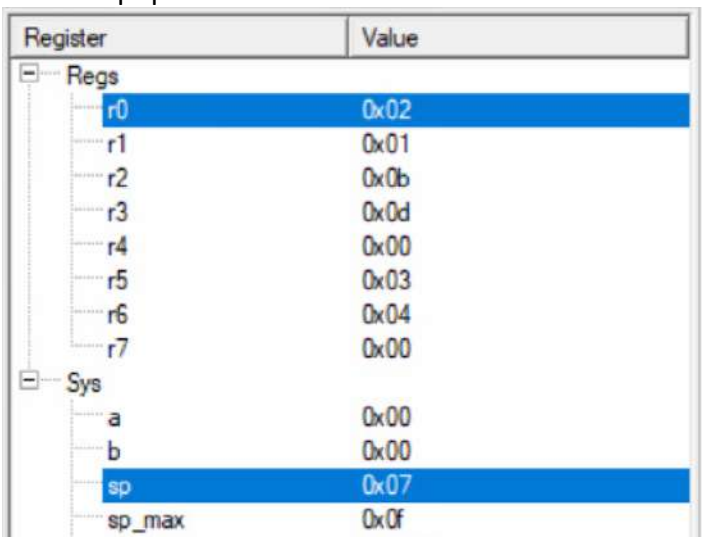
After 6th pop: into R2



After 7th pop: into R1



After 8th pop: into R0



Question 5

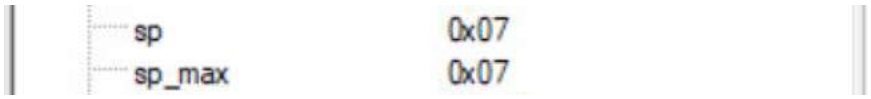
Code:

```
; registration number = 21BDS0340 (ignoring S)
; values = 2, 1, B, D, 0, 3, 4, 0
; moving values into bank 0
MOV R0, #02H
MOV R1, #01H
MOV R2, #0BH
MOV R3, #0DH
MOV R4, #00H
MOV R5, #03H
MOV R6, #04H
MOV R7, #00H

; pushing into stack, bank 1
```

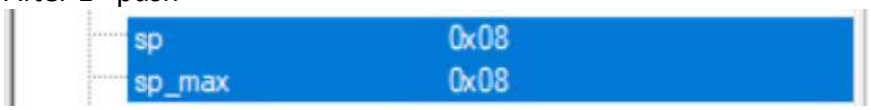
```
PUSH 00H
PUSH 01H
PUSH 02H
PUSH 03H
PUSH 04H
PUSH 05H
PUSH 06H
PUSH 07H

; popping from stack into R0-R7, bank 0
POP 00H
POP 01H
POP 02H
POP 03H
POP 04H
POP 05H
POP 06H
POP 07H

END
```

Output:

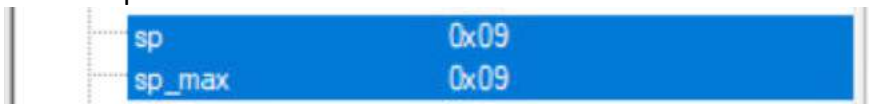
Push Operations:



After 1st push

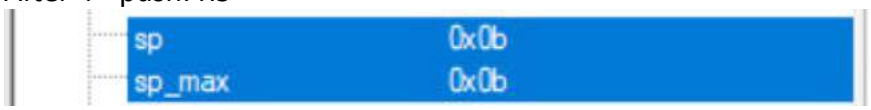


After 2nd push

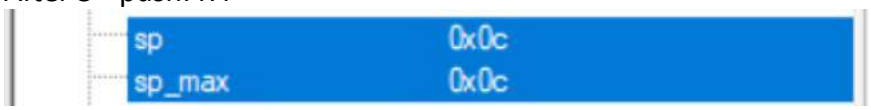


After 3rd push: R2

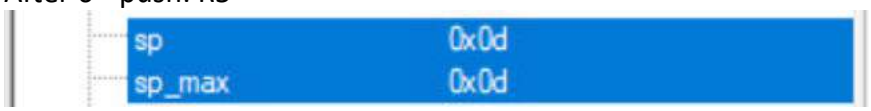
After 4th push: R3



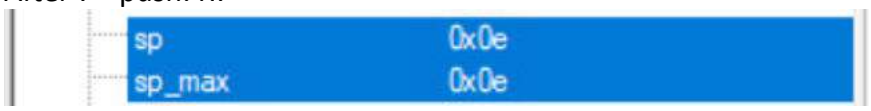
After 5th push: R4



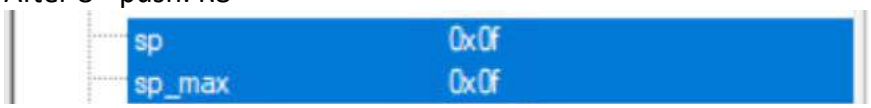
After 6th push: R5



After 7th push: R7



After 8th push: R8



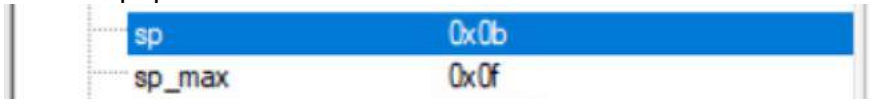
Pop Operations:

After 1st pop:

After 2nd pop:

After 3rd pop:

After 4th pop:



After 5th pop:

After 6th pop:

After 7th pop:

After 8th pop: