Exercise-homework day 8

8.1

Stack is the unique specification of how it is to be accessed. Stack is a LIFO (Last in First Out) structure. This means that the last thing that is put in the stack will be the first one to get out from the stack.

8.7

```
; Stack's locations : x3FFF(BASE) through x3FFB (MAX).
; R6 is the stack pointer.
; R3 contains the size of the stack element.
; R4 is pointer specifying the location of element to PUSH from or the space to
POP to

; The algorithm is going to push or pop elements continuously R3 times.

POP     ST  R2, Save2
        ST  R1, Save1
        ST  R0, Save0
        LD  R1, BASE ; BASE : -x3FFF.
        ADD R1, R1, #-1 ; R1 <-  -x4000.
        ADD R2, R6, R1 ; Compare stack pointer to x4000
        BRz fail_exit ; Branch if stack is empty.
        ADD R0, R4, #0  ;;(R0 <- R4 :origin pointer)
        ADD R1, R3, #0  ;;(R1 <- R3 :size counter )
        ADD R5, R6, R3  ;;(R5 <- R6 + R3)
        ADD R5, R5, #-1 ;;(R5 <- R5 - 1  :pointer that POP to)
        ADD R6, R6, R3  ;;(R6 <- R6 + R3)

;; We need to pop continuously until counter is 0
pop_loop    LDR R2, R5, #0  ;;(R2 <- MEM[R5])
            STR R2, R0, #0  ;;(MEM[R0] <- R2)
            ADD R0, R0, #1  ;;(R0 = R0 + 1)
            ADD R5, R5, #-1 ;;(R5 = R5 - 1)
            ADD R1, R1, #-1 ;;(R1 = R1 - 1 : size counter --)
            BRp pop_loop
            BRnzp success_exit

PUSH        ST  R2, Save2
            ST  R1, Save1
            ST  R0, Save0
            LD  R1,MAX ; MAX :-x3FFB
            ADD R2,R6,R1 ; Compare stack pointer to -x3FFB
            BRz fail_exit ; Branch if stack is full.
            ADD R0, R4, #0  ;;(R0 <- R4 :origin pointer)
            ADD R1, R3, #0  ;;(R1 <- R3 :size counter)
            ADD R5, R6, #-1 ;;(R5 <- R6 - 1)
            NOT R2, R3
            ADD R2, R2, #1  ;;(R2 <- -R3)
            ADD R6, R6, R2  ;;(R6 <- R6 + R2)

;; We need to push continuously until counter is 0
push_loop   LDR R2, R0, #0  ;;(R2 <- MEM[R0])
            STR R2, R5, #0  ;;(MEM[R5] <- R2)
            ADD R0, R0, #1  ;;(R0 <- R0 + 1)
            ADD R5, R5, #-1 ;;(R5 <- R5 - 1)
            ADD R1, R1, #-1 ;;(R1 <- R1 - 1)
            BRp push_loop

success_exit    LD  R0, Save0
                LD  R1, Save1 ; Restore original
                LD  R2, Save2 ; register values.
                AND R5, R5, #0 ; R5 <-- success.
                RET

fail_exit       LD  R0, Save0
                LD  R1, Save1 ; Restore original
                LD  R2, Save2 ; register values.
                AND R5, R5, #0
                ADD R5, R5, #1 ; R5 <-- failure.
                RET
BASE    .FILL   xC001 ;;-x3FFF
MAX     .FILL   xC005 ;;-x3FFB
Save0   .FILL   x0000
Save1   .FILL   x0000
Save2   .FILL   x0000
```

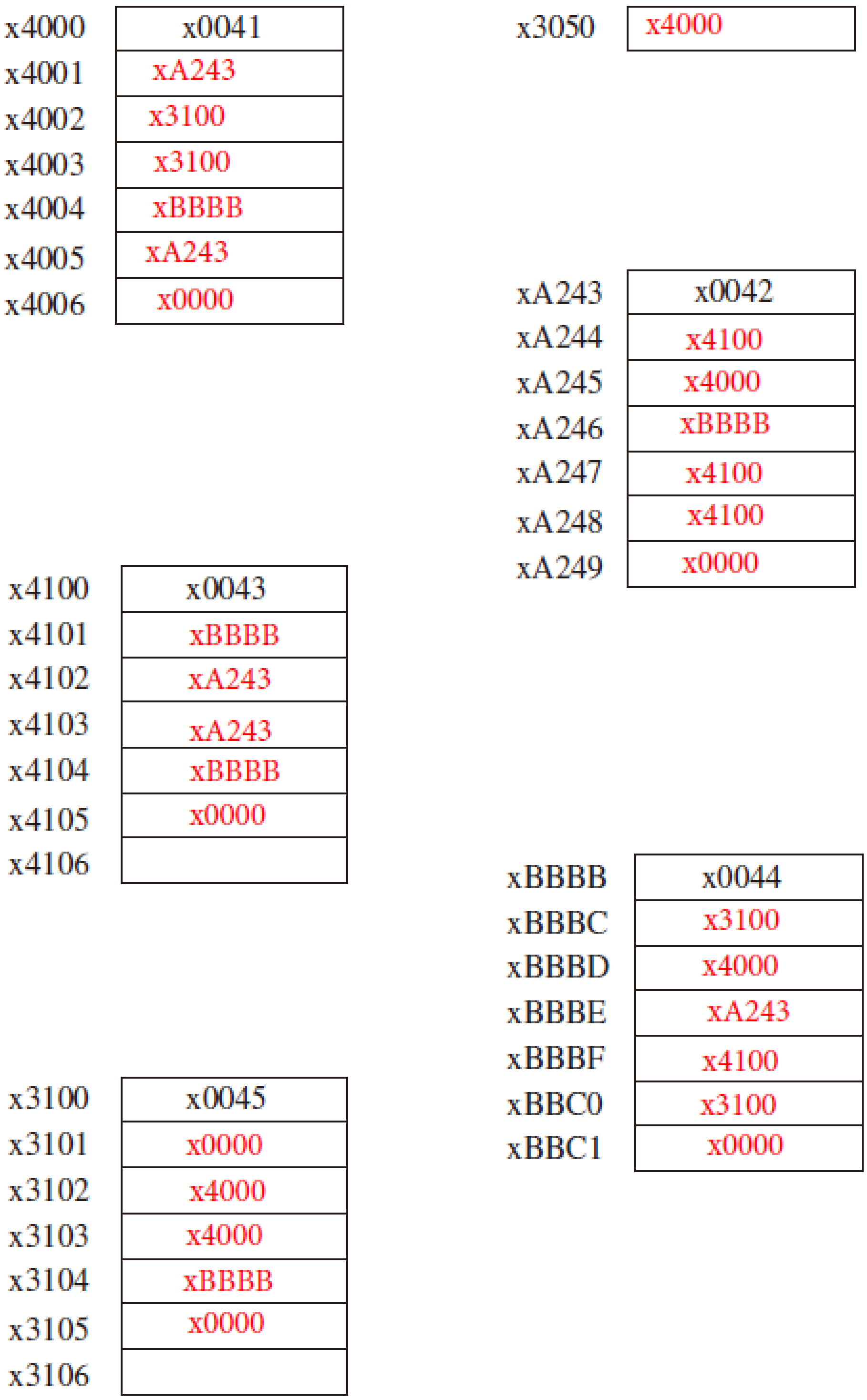
8.8

a. A F

b. stack contains most elements after `PUSH J` and after`PUSH K`

c. A F M

8.12



x4000	x0041
x4001	xA243
x4002	x3100
x4003	x3100
x4004	xBBBB
x4005	xA243
x4006	x0000

x4100	x0043
x4101	xBBBB
x4102	xA243
x4103	xA243
x4104	xBBBB
x4105	x0000
x4106	

x3100	x0045
x3101	x0000
x3102	x4000
x3103	x4000
x3104	xBBBB
x3105	x0000
x3106	

x3050	x4000

xA243	x0042
xA244	x4100
xA245	x4000
xA246	xBBBB
xA247	x4100
xA248	x4100
xA249	x0000

xBBBB	x0044
xBBBC	x3100
xBBBD	x4000
xBBBE	xA243
xBBBF	x4100
xBBC0	x3100
xBBC1	x0000

8.14

a: JSR X

b: LDR R1, R3, #1

c: LDR R2, R4, #1

d: ADD R1, R1, R0

e: ADD R0, R1, R2

f: STR R0, R5, #1

g: BRn LABEL

h: BRn ADDING

i: ADD R2,R2,#0