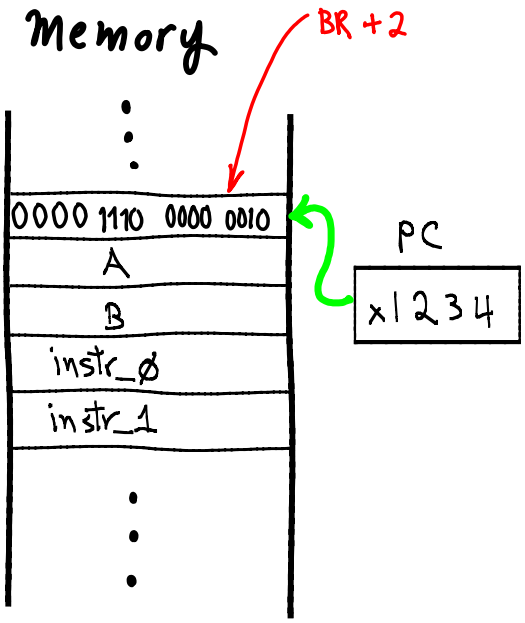
Lec-4-HW-1-selfModify-b-SOLN



Here's the code in symbolic form

```
BRnzp +3          ;--- jump over data
x000C             ;--- data, loop counter value (12)
x0001             ;--- data, A
x0002             ;--- data, B
LEA R4, #-3       ;--- R4 <=== A's address
LDR R0, R4, #0    ;--- R0 <=== A
LDR R1, R4, #1    ;--- R1 <=== B
LDR R2, R4, #2    ;--- R2 <=== instr_0
LDR R3, R4, #3    ;--- R3 <=== instr_1
LDR R5, R4, #-1   ;--- R5, loop counter <=== 12
                  ;--- LOOP
ADD R0, R0, R0    ;---    shift A left 1 bit
ADD R1, R1, R1    ;---    shift B left 1 bit
ADD R5, R5, #-1   ;---    decrement loop counter
BRnp, #-4         ;--- UNTIL R5 == 0
                  ;--- ENDLOOP
ADD R2, R2, R0    ;--- R2 <=== instr_0 + A
ADD R3, R3, R1    ;--- R3 <=== instr_1 + B
STR R2, R4, #0    ;--- store instr_0
STR R3, R4, #1    ;--- store instr_1
```

The same code w/ machine language translation

```
0000 111 000000011          BRnzp #+3          ;--- jump over data
x000C                                           ;--- data, loop counter value (12)
x0001                                           ;--- data, A
x0002                                           ;--- data, B
1110 100 111111101          LEA R4, #-3        ;--- R4 <=== A's address
0110 000 100 000000         LDR R0, R4, #0     ;--- R0 <=== A
0110 001 100 000001         LDR R1, R4, #1     ;--- R1 <=== B
0110 010 100 000010         LDR R2, R4, #2     ;--- R2 <=== instr_0
0110 011 100 000011         LDR R3, R4, #3     ;--- R3 <=== instr_1
0110 101 100 111111         LDR R5, R4, #-1    ;--- R5, loop counter <=== 12
                                    ;--- LOOP
0001 000 000 0 00 000       ADD R0, R0, R0     ;---    shift A left 1 bit
0001 001 001 0 00 001       ADD R1, R1, R1     ;---    shift B left 1 bit
0001 101 101 1 11111        ADD R5, R5, #-1    ;---    decrement loop counter
0000 101 111111100          BRnp, #-4          ;--- UNTIL R5 == 0
                                    ;--- ENDLOOP
0001 010 010 0 00 000       ADD R2, R2, R0     ;--- R2 <=== instr_0 + A
0001 011 011 0 00 001       ADD R3, R3, R1     ;--- R3 <=== instr_1 + B
0111 010 100 000010         STR R2, R4, #2     ;--- store instr_0
0111 011 100 000011         STR R3, R4, #3     ;--- store instr_1
```

Binary machine code Translated to hex.

```
0000 111 000000011      BRnzp #+3          ;--- jump over data
x0E03
x000C                                      ;--- data, loop counter value (12)
x0001                                      ;--- data, A
x0002                                      ;--- data, B
1110 100 111111101      LEA R4, #-3        ;--- R4 <=== A's address
xE9fd
0110 000 100 000000     LDR R0, R4, #0     ;--- R0 <=== A
x6100
0110 001 100 000001     LDR R1, R4, #1     ;--- R1 <=== B
x6301
0110 010 100 000010     LDR R2, R4, #2     ;--- R2 <=== instr_0
x6301
0110 011 100 000011     LDR R3, R4, #3     ;--- R3 <=== instr_1
x6703
0110 101 100 111111     LDR R5, R4, #-1    ;--- R5, loop counter <=== 12
x6b3f
                                           ;--- LOOP
0001 000 000 0 00 000   ADD R0, R0, R0     ;---     shift A left 1 bit
x1000
0001 001 001 0 00 001   ADD R1, R1, R1     ;---     shift B left 1 bit
x1241
0001 101 101 1 11111    ADD R5, R5, #-1    ;---     decrement loop counter
x1b7f
0000 101 111111100      BRnp, #-4          ;--- UNTIL R5 == 0
x0bfc
                                           ;--- ENDLOOP
0001 010 010 0 00 000   ADD R2, R2, R0     ;--- R2 <=== instr_0 + A
x1480
0001 011 011 0 00 001   ADD R3, R3, R1     ;--- R3 <=== instr_1 + B
x16c1
0111 010 100 000010     STR R2, R4, #2     ;--- store instr_0
x7502
0111 011 100 000011     STR R3, R4, #3     ;--- store instr_1
x7703
```