CS1021 Tutorial 7

Advanced LDR and STR Instructions

Q1 If R1 = 0x1000 and R4 = 8, what memory location (in hexadecimal) is loaded into R0 and what is the value of R1 (in hexadecimal) after each of the following instructions has been executed.

(i)  LDR  R0, [R1, #8]  ; R0=MEM[0x1008], R1=0x1000
(ii) LDR  R0, [R1], #8-8 ; R0=MEM[0x1000], R1=0x0FF8
(iii) LDR  R0, [R1, #12]! ; R1=0x100C, R0=MEM[0x100C]
(iv)  LDR  R0, [R1, R4]  ; R0=MEM[0x1008], R1=0x1000
(v)  LDR  R0, [R1], R4  ; R0=MEM[0x1000], R1=0x1008
(vi) LDR  R0, [R1, R4]! ; R1=0x1008, R0=MEM[0x1008]
(vii) LDR  R0, [R1, R4, LSL #3]  ; R0=MEM[0x1040], R1=0x1000
(viii) LDR  R0, [R1], R4, LSR #1  ; R0=MEM[0x1000], R1=0x1004
(ix)  LDR  R0, [R1, R4, LSL #2]!  ; R1=0x1020, R0=MEM[0x1020]

Q2 Given an array b at memory address 0x40001000 containing 64 32-bit integers b[0] to b[63], write ARM assembly language instructions for the following pseudo code statements. Assume i and j are 32-bit unsigned integers stored in memory locations 0x40000000 and 0x40000004 respectively.

(i)  R0 = b[7]
    LDR  R1, =0x40001000  ; R1 -> b
    LDR  R0, [R1, #7*4]  ; R0 = b[7] (MEM[b + 28])

(ii) R0 = b[i]
    LDR  R1, =0x40001000  ; R1 -> b
    LDR  R2, =0x40000000  ; R2 -> i
    LDR  R2, [R2]  ; R2 = i
    LDR  R0, [R1, R2, LSL #2]  ; R0 = b[i] (MEM[b + i*4])

(iii) i = b[i] + b[j]
    LDR  R1, =0x40001000  ; R1 -> b
    LDR  R2, =0x40000000  ; R2 -> i
    LDR  R3, [R2], #4  ; R3 = i AND R2 -> j
    LDR  R0, [R1, R3, LSL #2]  ; R0 = b[i] (MEM[b + i*4])
    LDR  R3, [R2], #4  ; R3 = j AND R2 -> i
    LDR  R3, [R1, R3, LSL #2]  ; R3 = b[j] (MEM[b + j*4])
    ADD  R0, R0, R3  ; R0 = b[i] + b[j]
    STR  R0, [R2]  ; i = R0 (b[i] + b[j])
(iv) \( b[i] = b[10] + b[j] \)

\[
\begin{align*}
\text{LDR} & \quad R1, =0x40001000 \quad ; R1 \rightarrow b \\
\text{LDR} & \quad R0, [R1, \#10*4] \quad ; R0 = b[10] \\
\text{LDR} & \quad R2, =0x40000004 \quad ; R2 \rightarrow j \\
\text{LDR} & \quad R3, [R2], \#-4 \quad ; R3 = j \text{ AND R2} \rightarrow i \\
\text{LDR} & \quad R3, [R1, R3, LSL \#2] \quad ; R3 = b[j] \ (\text{MEM}[b + j*4]) \\
\text{ADD} & \quad R0, R0, R3 \quad ; R0 = b[10] + b[j] \\
\text{LDR} & \quad R3, [R2] \quad ; R3 = j \\
\text{STR} & \quad R0, [R1, R3, LSL \#2] \quad ; b[i] \ (\text{MEM}[b + i*4]) = R0 \ (b[10] + b[j])
\end{align*}
\]
Q3  In a Scrabble® like game, players form words and each word is awarded a score that is the sum of the points for each letter in the word. English language editions of Scrabble contain 100 letter tiles with the following letter points and letter distribution:

2 blank tiles (scoring 0 points)

1 point:  E×12, A×9, I×9, O×8, N×6, R×6, T×6, L×4, S×4, U×4
2 points:  D×4, G×3
3 points:  B×2, C×2, M×2, P×2
4 points:  F×2, H×2, V×2, W×2, Y×2
5 points:  K×1
8 points:  J×1, X×1
10 points: Q×1, Z×1

For example, the word “MAZE” would have a score of 15 \((3 + 1 + 10 + 1)\).

Write an ARM assembly language program that will compute the word score for a NUL terminated string containing UPPER CASE alphabetic characters and spaces (for blanks). The word is stored in memory at the address contained in R1. The score for each letter is stored in flash memory as a sequence (or table or array) of 26 byte values. The first byte is the score for “A”, the second byte is the score for “B”, and so on (use the DCB assembler directive to create this table). Your program should calculate the word score in R0.

```
LDR R0, =0          ; score = 0
LDR R1, =MAZE       ; R1 -> word
LDR R4, =POINTS     ; R4 -> points table
L0  LDRB R2, [R1], #1 ; get ch AND R1 = R1 + 1
CMP R2, #0          ; if ch == 0
BEQ L1              ; finished
CMP R2, #0x20        ; if ch = ' ' 
BEQ L0              ; ignore as points == 0
SUB R2, R2, #0x41    ; index from 'A'
LDRB R2, [R4, R2]   ; get points for letter
ADD R0, R0, R2       ; add to score
B L0                 ; next ch
L1 ...
...
```

; points for each letter

POINTS DCB 1, 3, 3, 2, 1 ; A=1, B=3, C=3, D=2, E=1
DCB 4, 2, 4, 1, 8 ; F=4, G=2, H=4, I=1, J=8
DCB 5, 1, 3, 1, 1 ; K=5, L=1, M=3, N=1, O=1
DCB 3, 10, 1, 1, 1 ; P=3, Q=10, R=1, S=1, T=1
DCB 1, 4, 4, 8, 4 ; U=1, V=4, W=4, X=8, Y=4
DCB 10 ; Z=10

; test word

MAZE  DCB "MAZE", 0, 0