Mohamed Moselhy

Assignment 3

CS 2208

Dr. Mahmoud El-Sakka

Due: March 8, 2018

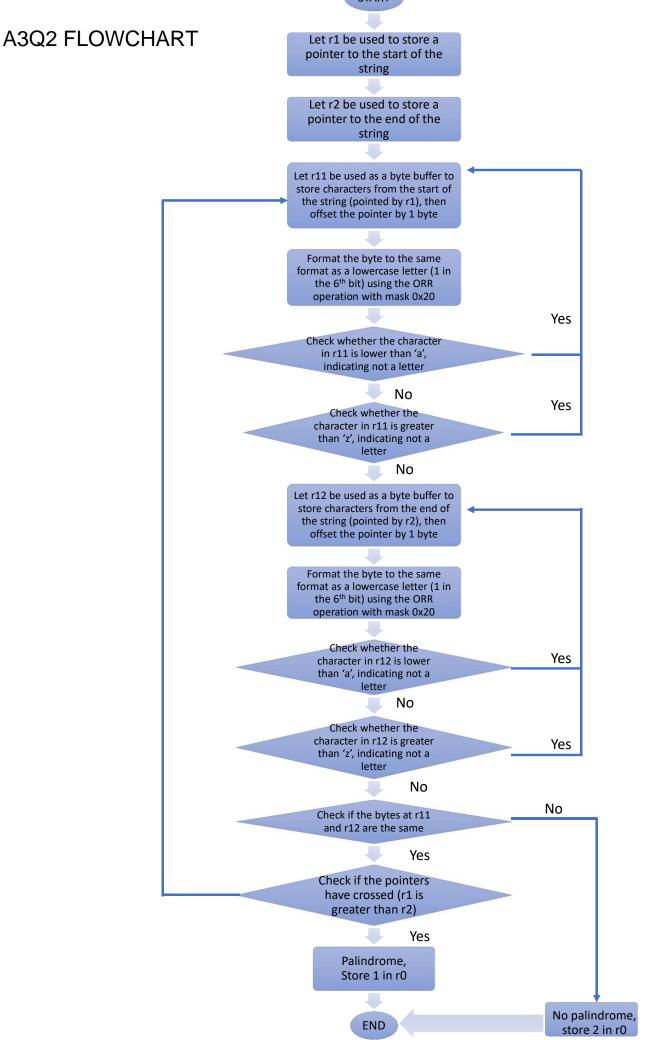
START A3Q1 FLOWCHART Let r10 store a pointer of the start of the UPC string Let r0 be a flag to indicate that we went through even indices of the UPC string, default = no = 0Offset the pointer by 1 byte to start pointing at even indices Use register r1 as a buffer to store the sum of the bytes that pointed to by r10 Reset the string pointer r10 to the start of the string again Increment the pointer r10 by 2 bytes Store 1 in r0 Let register r7 be used as a buffer to store the decoded byte in r1 to an integer by using the mask 0x0F Add the integer to the sum r2 Test whether we have reached the end of the string No Yes No Check whether we have already went through even indices (r0 = 1) Yes Check whether the sum in r2 is less than 10 Subtract 10 from r2 No Yes Check whether the sum in r2 is equal to 0 (it is a multiple of 10) No Yes

END

Set r0 to 2, because

the check digit is not valid

START



```
1
                                     ; This program will validate the check digit of any standard 12-digit
2
         AREA A3q1, CODE, READONLY
                                     ; Header line to specify program name
3
         ENTRY
                                     ; The program will start here
4
5
     HexToDec
                EOU 0x0F
                                     ; Mask to decode Hex-encoded bytes into integers
6
7
        LDR r10, =UPC
                                     ;Use register r10 to store a string pointer of the UPC code from memory
8
9
     StartOdd
10
       B EveryTwo
                                     ;Start looping to add the digits to register r2, used to keep the sum,
                                     starting at odd indices
11
                                     ; This section initializes the pointer to start at the first even index
12
     StartEven
                                     of the string
13
        LDR r10, =UPC
                                     ; Restart the string pointer in r10 back to the start of the string
         LDRB r1, [r10], #1
                                    ;Offset the string pointer by 1 to start pointing at even indices
14
15
         MOV r0, #1
                                    ; Use register r0 as a flag that we already started calculating even
                                     indices when it is 1
16
17
                                     ; This loop iterates through the string and stores the sum of the
     EveryTwo
                                     integers into r2
                                     ;Use register r1 as a buffer to store the byte that's currently pointed
18
        LDRB r1, [r10], #2
                                     at by r10 then increment the pointer by 2 bytes
         AND r7, r1, #HexToDec
19
                                     ;Decode the byte in r1 from Hex to integer using the mask `HexToDec`
                                     and store the integer in r7
20
         ADD r2, r7
                                     ;Add the integer to our sum in r2
21
22
         CMP r1, #0
                                     ; Test whether we have reached the end of the string (null character)
23
        BNE EveryTwo
                                     ; If we have not yet reached the end of the string, keep iterating
                                     through the string
24
        CMP r0, #1
                                     ; If we got here, then we have reached the end of the string, test
                                     whether we already went through even indices by looking at the flag
25
26
         ADDNE r2, r2, LSL#1
                                     ; Multiply the first sum by 3 when we're done calculating it
                                     (left-logical shift by 1, then add it to itself)
27
         BNE StartEven
                                     ; If we have not yet gone through the even indices, then go through them
28
29
                                     ; This loop calculates the remainder of r2 when divided by 10
     GetRemainder
30
         CMP r2, #10
                                     ; Check whether the r2 is less than 10, if it is then we got the remainder
31
         SUBGE r2, #10
                                     ; If it is not less than 10, then subtract 10 from it
                                     ; Keep doing this until we have a remainder
32
         BGE GetRemainder
33
34
       CMP r2, #0
                                    ; If the sum is a multiple of 10 (remainder is 0), then the check digit
                                    is correct, and r0 is already set to 1, which is the correct output for the program
35
        MOVNE r0, #2
                                     ; If it is not a multiple of 10, then the check digit is incorrect, so
                                     adjust the output by making r0 equal to 2
36
37
    STP B STP
                                     ; Halt the program using infinite loop
38
39
40
    UPC DCB "013800150738"
                                    ;UPC string
41
```

; The program will end here

42

END

```
1
                                     ; This program will check whether a string stored in memory is a
                                     case/special-character-insensitive palindrome
2
         AREA A3q2, CODE, READONLY
                                     ; Header line to specify program name
3
                                     ;The program will start here
         ENTRY
4
5
                    EOU 'a'
    LowBounds
                                     ;A byte with a value lower than this is not a lowercase letter
6
    UpBounds
                     EOU 'z'
                                     ; A byte with a value higher than this is not a letter
7
    ToLower
                     EQU 0x20
                                     ;A lowercase letter always has 1 in the 6th bit
8
9
     Start
10
        LDR r1, =STRING
                                    ;Let r1 store a pointer to the start of the string address
11
         LDR r2, =EoS
                                     ;Let r2 store a pointer to the end of the string address
12
13
                                    ; This loop keeps going until a valid letter byte is stored into the
    FromStart
                                     buffer r11
        LDRB r11, [r1], #1
14
                                     ;Let r11 be our buffer register from the start and load the first byte
                                     of the string into it (in the first iteration), then increment the pointer by 1 byte
15
         ORR r11, r11, #ToLower
                                    ;Convert it to lowercase format using the mask `ToLower` (make the 6th
                                     bit a `1`)
16
17
         CMP r11, #LowBounds
                                     ; Check whether this byte is not a letter (it is less than `a`)
18
         BLT FromStart
                                     ; If not, then repeat the loop
19
20
         CMP r11, #UpBounds
                                     ; Check whether this byte is not a letter (it is greater than `z`)
21
         BGT FromStart
                                     ; If not, then repeat the loop
22
23
    FromEnd
                                     ; This loop keeps going until a valid letter byte is stored into the
                                     buffer r12
24
         LDRB r12, [r2], #-1
                                     ;Let r12 be our buffer register from the end and load the last byte of
                                     the string into it (in the first iteration), then decrement the pointer by 1 byte
25
         ORR r12, r12, #ToLower
                                     ;Convert it to lowercase format using the mask `ToLower` (make the 6th
                                     bit a `1`)
26
27
         CMP r12, #LowBounds
                                     ;Check whether this byte is not a letter (it is less than `a`)
28
         BLT FromEnd
                                     ; If not, then repeat the loop
29
30
         CMP r12, #UpBounds
                                    ; Check whether this byte is not a letter (it is greater than `z`)
31
         BGT FromEnd
                                     ; If not, then repeat the loop
32
33
    Compare
                                    ;This loop compares the characters of the two buffer registers and
                                     checks whether it is a palindrome so far
34
                                    ;Compare the characters of both buffers (one from each end)
         CMP r11, r12
35
        MOVNE r0, #2
                                     ; If they are not equal, then we do not have a palindrome, and the
                                     correct output is to store 2 in r0
36
     LOS BNE LOS
                                     ; If we do not have a palindrome, end the program using an infinite loop
37
                                     ; If we got to this point then they are equal, and we should check
38
         CMP r1, r2
                                     whether our pointers crossed (all ends were checked)
                                     ; If our pointers crossed (if the left pointer is now on the right and
39
        MOVGE r0, #1
                                     vice versa), then we have a palindrome since all the bytes were equal at each end, and our correct
                                     output is 1 in r0
40
    WIN BGE WIN
                                     ;Stop the program with an infinite loop
41
42
         BLT FromStart
                                     ; If our pointers have not yet crossed, then keep setting and comparing
                                     the buffers until they have crossed or we have a mismatch
43
44
45
     STRING DCB "He lived as a devil, eh?" ; string
46
             DCB 0x00
                                            ;end of string
47
```

; The program will end here

48

END