**Question 1**



C:\Users\Vivian A. Lam\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2208-A3-prog1.png

;Vivian Lam

;program to determine whether a string of 12 ASCII encoded digits

;stored in memory

;is a valid UPC or not. If valid, store 1 in r0. if not, store 2 in

;r0.

AREA prog1, CODE, READONLY

ENTRY

;HINT 1: You can implement the division operation using repeated

;subtractio

;HINT 2: To calculate 3 × Z, you can do so using only one ADD

;instruction with LSL#1 shift.

;HINT 3: To load a byte to a register, use LDRB not LDR.

;r0 points to UPC

;r1 points to the position in UPC

;r2 for getting the current byte at that position

;r3 is the total for the first sum

;r4 is the total for the second sum

;convInt is a constant used for converting ASCII into integer

ADR r0, UPC ;r0 to point to UPC

convInt EQU 48 ;Constant for converting ASCII into integer

;(subtract 48)

;compute first sum

AddOdd ;Loop to add numbers in odd positions

LDRB r2,[r0,r1] ;load byte into register. the loaded byte is the

;byte r1 points to in UPC

SUB r2,#convInt ;convert that byte ASCII into int (subtract the

;constant)

ADD r3, r2 ;add integer value to total (first

;sum)

ADD r1, #2 ;point to next odd digit (increase

;counter by 2)

CMP r1, #12 ;loop condition: check if r1 pointer is

;outside of UPC (12 because 12 digits)

BNE AddOdd ;loop UNTIL we have iterated enough to

;add all the digits at odd positions

MOV r1, #1 ;Reset value of r1 to be 1 (this is so

;we can deal with even digits)

;compute second sum (add numbers in even positions)

AddEven ;Looop

LDRB r2,[r0,r1] ;load byte into register. the loaded byte is the

;byte r1 points to in UPC

SUB r2,#convInt ;convert that byte ASCII into int(subtract the

;constant)

ADD r4, r2 ;add integer value to total (second

;sum)

ADD r1, #2 ;point to next even digit (increase

;counter by 2)

CMP r1, #11 ;loop condition: check if r1 pointer is

;outside of UPC (11 because 12-1 digits)

BNE AddEven ;loop UNTIL we have iterated enough to

;add all the digits at even positions

;multiply first sum by 3

ADD r3, r3, r3, LSL #1 ;left shift to double the first sum

;and add first sum to it

;and add it to second sum (store in r4)

ADD r4, r3

;subtract 1 (store in r4)

SUB r4, #1;

;compute remainder when adjusted total is divided by 10

RptSub ;LOOP until the remaining total is less

;than or equal to 9

CMP r4,#9 ;check if total is less than or equal to 9

SUBGT r4,#10 ;subtract 10 from total if greater than 9

BHI RptSub ;end loop

;subtract remainder form 9

RSB r5,r4,#9 ;Subtract the remainder from 9, store in r5

;(calculated check digit)

LDRB r6,[r0,r1] ;Load UPC check digit into r6 (r1 is already

;pointing at check digit position)

SUB r6,#convInt ;Subtract 48 from check digit's ASCII value to

;obtain its integer value

MOV r0,#2 ;Set r0 to 2 - if check digits match this will

;change to 1 else, stays 2

;check result

CMP r5,r6 ;Compare UPC check digit with calculated check

;digit valid, store 1 in r0 invalid, store 2 in r0

MOVEQ r0,#1 ;store 1 in r0 if check digits match and UPC is

;valid

MOVNE r0,#2 ;store 2 in r0 if check digits match and

;UPC is invalid

Loop B Loop ;Infinite loop to prevent error

;test values

;UPC DCB "013800150738" ;UPC String

;UPC DCB "060383755577" ;UPC String

UPC DCB "065633454712" ;UPC String

END

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

Test 1:

UPC = 013800150738

r0= 0x00000001

r5= 0x00000008

r6= 0x00000008

r5 (supposed check digit) and r6(actual check digit on UPC) are the SAME values and thus is VALID and 1 is stored in r0.

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

Test 2:

UPC = 060383755577

r0= 0x00000001

r5= 0x00000007

r6= 0x00000007

r5 (supposed check digit) and r6(actual check digit on UPC) are the SAME values and thus is VALID and 1 is stored in r0.

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

Test 3:

UPC = 065633454712

r0= 0x00000001

r5= 0x00000002

r6= 0x00000002

r5 (supposed check digit) and r6(actual check digit on UPC) are the SAME values and thus is VALID and 1 is stored in r0.

**$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$**

**Question 2**

**C:\Users\Vivian A. Lam\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2208-A3-prog2.png**

;Vivian Lam

;program to check if string is a palindrome or not

;ignores case and special characters

AREA prog2, CODE, READONLY

ENTRY

;r0 is a pointer that points to the string

;r1 is a pointer to point the the current character in the string

;r2 is another character to point to the current character in the string from the other end

;r3 is the current character that r1 points to

;r4 is the current character that r2 points to

;r5 points to end of string

;toLower is a constant

LDR r0,=STRING ;make r0 to point to the string

LDR r5,=EoS ;make r5 to point to EoS (end of string, so we can check if we reach the end)

MOV r1,#-1 ;make pointer in r1 to -1 (incremented later to 0)

convLow EQU 32 ;Add 32 to convert uppercase to lowercase letters in ASCII

LEN ;LOOP to find out the length of the string

LDRB r3,[r0,r2] ;Load a byte of the string (character at position pointed at by r2)

CMP r3,r5 ;Check if the character is the null character (end of string)

BEQ Check ;If so, stop length count and exit loop

ADD r2,#1 ;Else, increment pointer to point to next character

B LEN ;UNTIL end of string is reached & r2 points at EoS

Check ;Pointers of r1 and r2 are located at opposite ends of the string

CMP r1,r2 ;Check if pointers have crossed paths yet

BGT CheckPal ;If so, string is a palindrome as letter pairs have all matched

ADD r1,#1 ;Increment pointer at the start of the string

SUB r2,#1 ;Decrement pointer at the end of the string

Char1 ;LOOP

LDRB r3,[r0,r1] ;Get character 1 at position pointed at by r1

CMP r3,#'A' ;Check if character 1 is possibly not a letter

ADDLT r1,#1 ;If possibly not a letter, increment this pointer

BLT Char1 ;Get next character

CMP r3,#'z' ;Check if character is greater than 'z'

ADDGT r1,#1 ;If so, character is not a letter so increment pointer and get next character

BGT Char1 ;UNTIL character is a letter

CMP r3,#'a' ;Check if character 1 is uppercase

ADDLT r3,#convLow ;If so, add 32 to convert character to lowercase equivalent

Char2 ;LOOP

LDRB r4,[r0,r2] ;Get character 2 at position pointed at by r2

CMP r4,#'A' ;Check if character 2 is possibly not a letter

SUBLT r2,#1 ;If possibly not a letter, decrement this pointer

BLT Char2 ;Get next character

CMP r4, #'z' ;Check if character 2 is greater than 'z'

SUBGT r2,#1 ;If so, character is not a letter so decrement pointer and get next character

BGT Char2 ;UNTIL character is a letter

CMP r4,#'a' ;Check if character 2 is uppercase

ADDLT r4,#convLow ;If so, add 32 to convert character to lowercase equivalent Now r3 and r4 contain two lowercase letters

;checking

CMP r3,r4 ;Compare character 1 and character 2

BEQ Check ;If equal, the string is possibly a palindrome. Continue comparing character pairs. If they are not equal, the string is not a palindrome

MOV r0,#0 ;Set r0 to 0 to indicating that the string is not a palindrome

B Loop ;Skip to end

CheckPal MOV r0,#1 ;Set r0 to 1 indicating that the string is a palindrome

Loop B Loop ;End program with infinite loop to prevent error

STRING DCB "He lived as a devil, eh?" ;string test value

;STRING DCB "asdfg";string test value

EoS DCB 0x00 ;End of string ASCII value

END

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

Test 1:

String = “He lived as a devil, eh?”

r0 = 0x00000001

Since r0 stores 1, the string IS a palindrome.

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

Test 2:

String = “asdfg”

r0 = 0x00000000

Since r0 stores 0, the string IS NOT a palindrome.

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

Test 3:

String = “sauidhi”

r0 = 0x00000000

Since r0 stores 0, the string IS NOT a palindrome.

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

Test 4:

String = “a”

r0 = 0x00000001

Since r0 stores 1, the string IS a palindrome.

**$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$**