**F.10 Chapter 10 Solutions**

10.1 The Multiply step works by adding the multiplicand a number of times to an accumulator. The number of times to add is determined by the multiplier. The number of instructions executed to perform the Multiply step = 3 + 3\*n, where n is the value of the multiplier. We will in general do better if we replace the core of the Multiply routine (lines 17 through 19 of Figure 10.14) with the following, doing the Multiply as a series of shifts and adds:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | AND | R0, R0, | #0 |  |
| ADD | R4, R0, | #1 | ;R4 contains the bit mask (x0001) |
| Again  BitZero | AND  BRz ADD  BRn ADD | R5, R2,  BitZero R0, R0,  Restore2 R1, R1, | R4 R1  R1 | ;Is corresponding  ;bit of multiplier=1  ;Multiplier bit=1  ;--> add  ;shifted multiplicand  ;Product has already  ;exceeded range  ;Shift the |
|  | BRn  ADD | Check  R4, R4, | R4 | ;multiplicand bits  ;Mcand too big  ;--> check if any  ;higher mpy bits = 1  ;Set multiplier bit to  ;next bit position |
|  | BRn BRnzp | DoRangeCheck Again | | ;We have shifted mpy  ;bit into bit 15  ;-->done. |
| Check  DoRangeCheck | AND  BRp ADD  BRp | R5, R2, R4  Restore2 R4, R4, R4  Check | |  |

10.3 This program assumes that hex digits are all capitalized.

LD R3, NEGASCII

LD R5, NEGHEX

TRAP x23

ADD R1, R0, R3 ;Remove ASCII template LD R4, HEXTEST ;Check if digit is hex ADD R0, R1, R4

BRnz NEXT1

ADD R1, R1, R5 ;Remove extra

;offset for hex

|  |  |  |  |
| --- | --- | --- | --- |
| NEXT1 | TRAP | x23 |  |
|  | ADD | R0, R0, R3 | ;Remove ASCII template |
|  | ADD  BRnz ADD | R2, R0, R4 NEXT2  R0, R0, R5 | ;Check if digit is hex  ;Remove extra |
|  |  |  | ;offset for hex |
| NEXT2 | ADD | R0, R1, R0 | ;Add the numbers |
|  | ADD  BRnz LD ADD | R1, R0, R4 NEXT3  R2, HEX R0, R0, R2 | ;Check if digit > 9  ;Add offset for hex digits |
| NEXT3 | LD | R2, ASCII |  |
|  | ADD | R0, R0, R2 | ;Add the ASCII template |
| DONE | TRAP | x21 |  |
|  | TRAP | x25 |  |

ASCII .FILL x0030

NEGASCII .FILL x-0030

HEXTEST .FILL #-9

HEX .FILL x0007

NEGHEX .FILL x-7

10.5 ;

; R1 contains the number of digits including ’x’. Hex

; digits must be in CAPS.

ASCIItoBinary AND R0, R0, #0 ; R0 will be used for our result ADD R1, R1, #0 ; Test number of digits.

BRz DoneAtoB ; There are no digits

;

LD R3, NegASCIIOffset ; R3 gets xFFD0, i.e., -x0030 LEA R2, ASCIIBUFF

LD R6, NegXCheck LDR R4, R2, #0 ADD R6, R4, R6

BRz DoHexToBin

ADD R2, R2,R1

ADD R2, R2, #-1 ; R2 now points to "ones" digit

;

LDR R4, R2, #0 ; R4 <-- "ones" digit

ADD R4, R4, R3 ; Strip off the ASCII template

|  |  |  |
| --- | --- | --- |
| ; | ADD | R0, R0, R4 ; Add ones contribution |
|  | ADD | R1, R1, #-1 |
|  | BRz | DoneAtoB ; The original number had one digit |
|  | ADD | R2, R2, #-1 ; R2 now points to "tens" digit |
| ; |  |  |
|  | LDR | R4, R2, #0 ; R4 <-- "tens" digit |
|  | ADD | R4, R4, R3 ; Strip off ASCII template |
|  | LEA | R5, LookUp10 ; LookUp10 is BASE of tens values |
|  | ADD | R5, R5, R4 ; R5 points to the right tens value |
|  | LDR | R4, R5, #0 |
|  | ADD | R0, R0, R4 ; Add tens contribution to total |
| ; |  |  |
|  | ADD | R1, R1, #-1 |
|  | BRz | DoneAtoB ; The original number had two digits |
|  | ADD | R2, R2, #-1 ; R2 now points to "hundreds" digit |
| ; |  |  |
|  | LDR | R4, R2, #0 ; R4 <-- "hundreds" digit |
|  | ADD | R4, R4, R3 ; Strip off ASCII template |
|  | LEA | R5, LookUp100 ; LookUp100 is hundreds BASE |
|  | ADD | R5, R5, R4 ; R5 points to hundreds value |
|  | LDR | R4, R5, #0 |
|  | ADD | R0, R0, R4 ; Add hundreds contribution to total |
|  | RET |  |

DoHexToBin ; R3 = NegASCIIOffset

; R2 = Buffer Pointer

; R1 = Num of digits + x

;

ST R7, SaveR7

LD R6, NumCheck ADD R1, R1, #-1

ADD R2, R2,R1

;

LDR R4, R2, #0 ; R4 <-- "ones" digit

ADD R4, R4, R3 ; Strip off the ASCII template ADD R7, R4, R6

BRnz Cont1

LD R7, NHexDiff ADD R4, R4, R7

Cont1 ADD R0, R0, R4 ; Add ones contribution

;

ADD R1, R1, #-1

BRz DoneAtoB ; The original number had one digit ADD R2, R2, #-1 ; R2 now points to "tens" digit

;

LDR R4, R2, #0 ; R4 <-- "tens" digit

ADD R4, R4, R3 ; Strip off ASCII template ADD R7, R4, R6

BRnz Cont2

LD R7, NHexDiff ADD R4, R4, R7

|  |  |  |  |
| --- | --- | --- | --- |
| Cont2 | LEA  ADD | R5,  R5, | LookUp16  R5, R4 |
|  | LDR | R4, | R5, #0 |
|  | ADD | R0, | R0, R4 |
| ; |  |  |  |
|  | ADD | R1, | R1, #-1 |

BRz DoneAtoB ; The original number had two digits ADD R2, R2, #-1 ; R2 now points to "hundreds" digit

;

LDR R4, R2, #0

ADD R4, R4, R3 ; Strip off ASCII template ADD R7, R4, R6

BRnz Cont3

LD R7, NHexDiff ADD R4, R4, R7

Cont3 LEA R5, LookUp256 ADD R5, R5, R4 LDR R4, R5, #0 ADD R0, R0, R4

;

DoneAtoB LD R7, SaveR7 RET

NegASCIIOffset .FILL xFFD0 NumCheck .FILL #-9

NHexDiff .FILL #-7

NegXCheck .FILL xFF88

SaveR7 .FILL x0000

ASCIIBUFF .BLKW 4

LookUp10 .FILL #0

.FILL #10

.FILL #20

|  |  |  |
| --- | --- | --- |
| ; | .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL | #30  #40  #50  #60  #70  #80  #90 |
| LookUp100  LookUp16 | .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL | #0  #100  #200  #300  #400  #500  #600  #700  #800  #900  #0  #16  #32  #48  #64  #80  #96  #112  #128  #144  #160  #176  #192  #208  #224  #240 |
| LookUp256 | .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL  .FILL | #0  #256  #512  #768  #1024  #1280  #1536  #1792  #2048  #2304 |

|  |  |
| --- | --- |
| .FILL | #2560 |
| .FILL | #2816 |
| .FILL | #3072 |
| .FILL | #3328 |
| .FILL | #3584 |
| .FILL | #3840 |

10.7 This program reverses the input string. For example, given an input of “Howdy”, the output is “ydwoH”.

10.9 NOTE: This question is redundant. The PUSH\_VALUE routine is already robust in that is does test to be sure that each character typed is a decimal digit. No further work needs to be done.