LAB NO : 4 DATE : 29/01/2025

Title : BRANCHING AND LOOPING

# Lab Exercise 1: Write an ARM assembly language program to convert a 32-bit BCD number in the unpacked form into packed form

## Code :

## AREA RESET, DATA, READONLY

## EXPORT \_\_Vectors

## \_\_Vectors

## DCD 0x10001000 ; stack pointer value when stack is empty

## DCD Reset\_Handler ; reset vector

## ALIGN

## AREA mycode, CODE, READONLY

## ENTRY

## EXPORT Reset\_Handler

## Reset\_Handler

## LDR R0, =0x01020304 ; Example Unpacked BCD

## ; Extract the first digit

## ; R1 -> holds digits

## AND R1, R0, #0x0F

## MOV R2, R1

## 

## AND R1, R0, #0x0F00

## MOV R1, R1, LSR #4

## ORR R2, R2, R1

## 

## AND R1, R0, #0x0F0000

## MOV R1, R1, LSR #8

## ORR R2, R2, R1

## 

## AND R1, R0, #0x0F000000

## MOV R1, R1, LSR #12

## ORR R2, R2, R1

## STOP

## B STOP

## END

## Output :

# Lab Exercise 2: Write an ARM assembly language program to convert a 32-bit packed BCD number into its equivalent hexadecimal number.

## Code :

AREA RESET, DATA, READONLY

EXPORT \_\_Vectors

\_\_Vectors

DCD 0x10001000 ; stack pointer value when stack is empty

DCD Reset\_Handler ; reset vector

ALIGN

AREA mycode, CODE, READONLY

ENTRY

EXPORT Reset\_Handler

Reset\_Handler

LDR R0, =0x99999999 ; Example packed BCD

MOV R2, #1 ;place

MOV R3, #0 ;Sum

MOV R4, #10

Loop

AND R1, R0, #0xF ;Extract digit

MLA R3, R2, R1, R3

MUL R2, R2, R4

MOVS R0, R0, LSR #4

BNE Loop

STOP

B STOP

END

## Output :

# Lab Exercise 3: Write an ARM assembly language program to convert a 16-bit hex number into its equivalent packed BCD.

## Code :

AREA RESET, DATA, READONLY

EXPORT \_\_Vectors

\_\_Vectors

DCD 0x10001000 ; stack pointer value when stack is empty

DCD Reset\_Handler ; reset vector

ALIGN

AREA mycode, CODE, READONLY

ENTRY

EXPORT Reset\_Handler

Reset\_Handler

LDR R0, =N

LDR R1, =DST

LDRH R2, [R0]

MOV R5, #0

MOV R7, #10

MOV R3, #0

MOV R4, #0

loop CMP R2,R7

BLT exitDiv

SUB R2,R7

ADD R3,#1

B loop

exitDiv MOV R8, #0

shifter CMP R4, R8

BEQ exitshifter

LSL R2, #4

ADD R8, #1

B shifter

exitshifter ADD R4, #1

ADD R5, R2

CMP R3, #0

BEQ exit

MOV R2, R3

MOV R3,#0

B loop

exit STR R5, [R1]

STOP B STOP

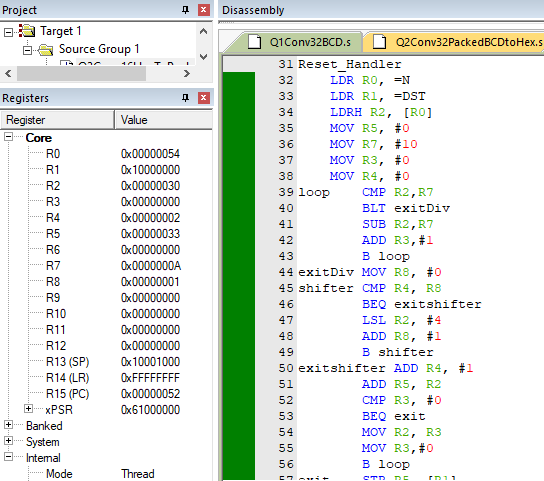
N DCD 0x21

AREA mydata, DATA, READWRITE

DST DCD 1

END

## **Output :**



# Lab Exercise 4: Write an ARM assembly language program to add two 32-bit packed BCD numbers and store the result in packed BCD form.

## Code :

AREA RESET, DATA, READONLY

EXPORT \_\_Vectors

\_\_Vectors

DCD 0x10001000 ; stack pointer value when stack is empty

DCD Reset\_Handler ; reset vector

ALIGN

AREA mycode, CODE, READONLY

ENTRY

EXPORT Reset\_Handler

Reset\_Handler

LDR R0, =N1

LDR R10, =N2

LDR R1, =DST

LDR R11, =CARRY

MOV R2, #0

MOV R5, #0

MOV R7, #10

MOV R12, #0

loop LDRB R3, [R0]

AND R4, R3, #0x0F

LDRB R3, [R10]

AND R6, R3, #0x0F

BL subroutine

BL raise

ADD R5, R4

ADD R2, #1

LDRB R3, [R0], #1

AND R4, R3, #0xF0

LSR R4, #4 ;shift 4 bits

LDRB R3, [R10], #1

AND R6, R3, #0xF0

LSR R6, #4 ;shift 4 bits

BL subroutine

BL raise

ADD R5, R4

ADD R2, #1

CMP R2, #8

BNE loop

STR R5, [R1]

STR R12, [R11]

B STOP

subroutine ADD R4, R12

MOV R12, #0

ADD R4, R6

CMP R4, #10

BLT skip

SUB R4, #10

MOV R12, #1

skip MOV R6, #1

MOV R8, #0

BX LR

raise CMP R2, R8

BEQ exitraise

MUL R4, R4, R7

ADD R8, #1

B raise

exitraise BX LR

STOP B STOP

N1 DCD 2\_00000000000000000000000000111001

N2 DCD 2\_00000000000000000000000000010010

AREA mydata, DATA, READWRITE

DST DCD 1

CARRY DCD 1

END

## **Output :**

# Additional Exercise 1 : Unpack a 32-bit BCD number into 8 32-bit numbers

## Code :

### AREA RESET, DATA, READONLY

### EXPORT \_\_Vectors

### \_\_Vectors

### DCD 0x10001000 ; stack pointer value when stack is empty

### DCD Reset\_Handler ; reset vector

### ALIGN

### AREA mycode, CODE, READONLY

### ENTRY

### EXPORT Reset\_Handler

### Reset\_Handler

### LDR R0, =0x99999999 ; Example packed BCD

### AND R1, R0, #0xF ;Extract digit

### MOV R0, R0, LSR #4

### AND R2, R0, #0xF ;Extract digit

### MOV R0, R0, LSR #4

### AND R3, R0, #0xF ;Extract digit

### MOV R0, R0, LSR #4

### AND R4, R0, #0xF ;Extract digit

### MOV R0, R0, LSR #4

### AND R5, R0, #0xF ;Extract digit

### MOV R0, R0, LSR #4

### AND R6, R0, #0xF ;Extract digit

### MOV R0, R0, LSR #4

### AND R7, R0, #0xF ;Extract digit

### MOV R0, R0, LSR #4

### AND R8, R0, #0xF ;Extract digit

### MOV R0, R0, LSR #4

### STOP

### B STOP

### END

## **Output :**

# Additional Exercise 2 : Multiply two 16-bit packed BCD numbers and store result in BCD form.

## Code :

### AREA RESET, DATA, READONLY

### EXPORT \_\_Vectors

### \_\_Vectors

### DCD 0x10001000

### DCD Reset\_Handler

### ALIGN

### AREA mycode, CODE, READONLY

### ENTRY

### EXPORT Reset\_Handler

### Reset\_Handler

### MOV R0, #0x1234

### MOV R1, #0x5678

### MOV R2, #0

### MOV R3, #0

### MOV R4, #4

### bcd\_to\_int\_1

### LSR R5, R0, #12

### AND R5, R5, #0xF

### MOV R6, #10

### MUL R2, R2, R6

### ADD R2, R2, R5

### LSR R0, R0, #4

### SUBS R4, R4, #1

### BNE bcd\_to\_int\_1

### MOV R4, #4

### bcd\_to\_int\_2

### LSR R5, R1, #12

### AND R5, R5, #0xF

### MUL R3, R3, R6

### ADD R3, R3, R5

### LSR R1, R1, #4

### SUBS R4, R4, #1

### BNE bcd\_to\_int\_2

### MUL R4, R2, R3

### MOV R0, #0

### MOV R5, #4

### int\_to\_bcd

### MOV R6, R4

### MOV R7, #10

### MOV R8, #0

### subtraction\_loop

### CMP R6, R7

### BGE subtract

### B done\_subtract

### subtract

### SUB R6, R6, R7

### ADD R8, R8, #1

### done\_subtract

### ADD R0, R0, R8

### MUL R0, R0, R7

### MOV R9, R4

### MOV R10, #10

### MOV R11, #0

### subtraction\_loop\_2

### CMP R9, R10

### BGE subtract\_2

### B done\_subtract\_2

### subtract\_2

### SUB R9, R9, R10

### ADD R11, R11, #1

### done\_subtract\_2

### ADD R0, R0, R11

### MUL R0, R0, R10

### SUBS R5, R5, #1

### BNE int\_to\_bcd

### STOP

### B STOP

## **Output :**

