|  |  |
| --- | --- |
| Course Title: **Microprocessor Laboratory** | Course code**: CSL1543** |
| Lab Session: **5** | Student Name:  USN :  Date : |
| Title: **Introduction to Microprocessors** |
|  |
| Faculty Signature: | Marks(out of 10): |

**1.Write an ALP to illustrate concept of function using BL and BX**

PRESERVE8 ; Indicate the code here preserve

; 8 byte stack alignment

THUMB ; Indicate THUMB code is used

AREA |.text|, CODE, READONLY

EXPORT \_\_main

; Start of CODE area

function1

SUB SP, SP, #0x8 ; Reserve 2 words of stack ;(8 bytes) for local variables ;Data processing in function

MOVS r0, #0x12 ; set a dummy value

STR r0, [sp, #0] ; Store 0x12 in 1st local variable

STR r0, [sp, #4] ; Store 0x12 in 2nd local variable

LDR r1, [sp, #0] ; Read from 1st local variable

LDR r2, [sp, #4] ; Read from 2nd local variable

ADD SP, SP, #0x8; Restore SP to original position

BX LR

\_\_main

BL function1

stop B stop

END

**2.Write a program to illustrate push and pop and also change of stack address**

PRESERVE8 ; Indicate the code here preserve

; 8 byte stack alignment

THUMB ; Indicate THUMB code is used

AREA |.text|, CODE, READONLY

EXPORT \_\_main

; Start of CODE area

\_main

LDR r3,=0x20000100

LDR r0,=0x20000050

LDMIA r3!,{r1,r2}

mov SP,r0

PUSH {r1,r2}

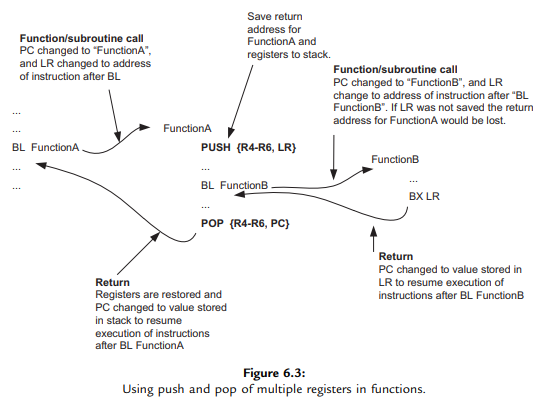
POP {r4,r5}

stop B stop

END

**3.Write an ALP to reflect the concept of nested function**

**Realisation of Nested Functions**

****

**Main.asm**

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB ; Indicate THUMB code is used**

**AREA |.text|, CODE, READONLY**

**EXPORT \_\_main**

**EXTERN func**

**; Start of CODE area**

**\_\_main**

**LDR r0,=0x10;**

**BL func**

**stop B stop**

**END**

**Function2.asm**

**PRESERVE8**

**THUMB**

**AREA |.text|, CODE, READONLY**

**EXPORT func**

**EXTERN func2**

**func**

**push{LR}**

**MOVS R1,#08**

**BL func2**

**pop{PC}**

**END**

**Function3.asm**

**PRESERVE8**

**THUMB**

**AREA |.text|, CODE, READONLY**

**EXPORT func2**

**func2**

**MOVS r2,#08**

**BX LR**

**END**

**4.Write a program to illustrate the processing of data in a stack and realizing of stack using another file say (processing x to read 2x+9)**

**Main.asm**

**PRESERVE8 ; Indicate the code here preserve**

**; 8 byte stack alignment**

**THUMB ; Indicate THUMB code is used**

**AREA |.text|, CODE, READONLY**

**EXPORT \_\_main**

**EXTERN func**

**; Start of CODE area**

**\_\_main**

**LDR r0,=0x10;**

**BL func**

**stop B stop**

**END**

**PRESERVE8**

**THUMB**

**AREA |.text|, CODE, READONLY**

**EXPORT func**

**EXTERN func2**

**func**

**push{LR}**

**MOVS R1,#08**

**BL func2**

**pop{PC}**

**END**

**PRESERVE8**

**THUMB**

**AREA |.text|, CODE, READONLY**

**EXPORT func2**

**func2**

**MOVS r2,#08**

**BX LR**

**END**

|  |
| --- |
|  |

**5.Write a program to clear and extract the bits with starting position P and width of the bits W**

PRESERVE8 ; Indicate the code here preserve

; 8 byte stack alignment

THUMB ; Indicate THUMB code is used

AREA |.text|, CODE, READONLY

EXPORT \_\_main

; Start of CODE area

\_\_main

;extracting

LDR r0,=0xFFC0FFFF

LSLS r0,r0,#(32-16-8)//(32-W-P)

LSRS r0,r0,#(32-8)

;clearing

LDR r0,=0xFFC0FFFF

MOVS r1,#16

MOVS R2,#08;(32-16-08)//( 32-W-P)

MOVS r3,#08

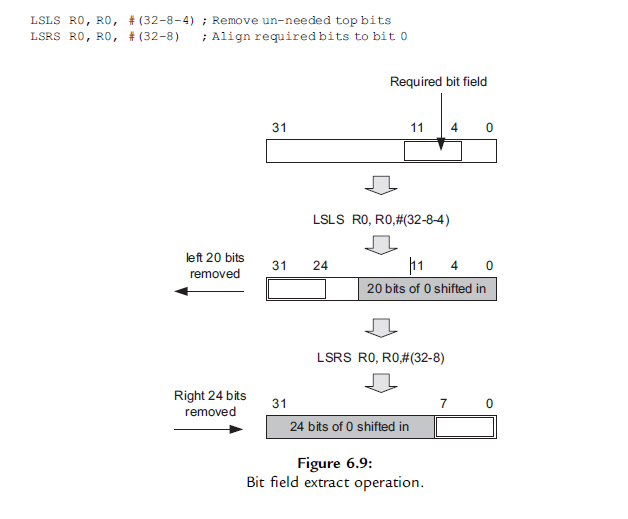
RORS r0,r0,r1

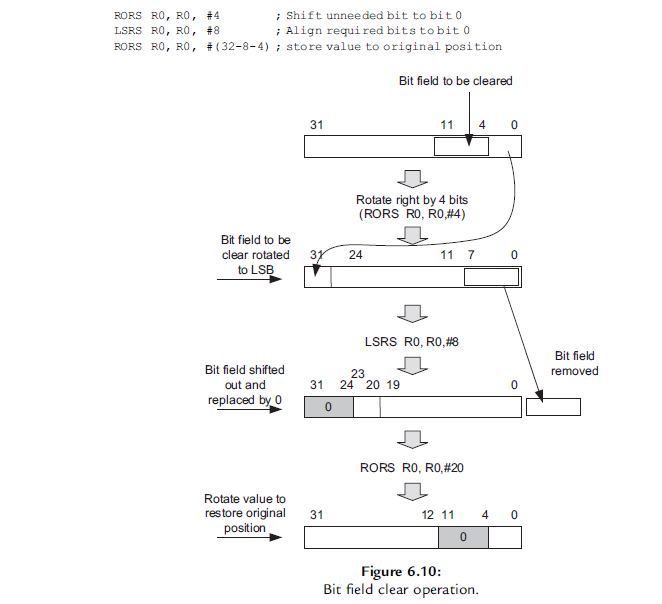
LSRS r0,r0,r2

RORS r0,r0,r3

stop B stop

END





**6.Write a program to use switch case**

PRESERVE8 ; Indicate the code here preserve

; 8 byte stack alignment

THUMB ; Indicate THUMB code is used

AREA |.text|, CODE, READONLY

EXPORT \_\_main

; Start of CODE area

\_\_main

LDR R0, =0

CMP R0, #3 ; Compare input to maximum valid choice

BHI default\_case ; Branch to default case if higher than 3

MOVS R2, #4 ; Multiply branch table offset by 4

MULS R0, R2, R0 ; (size of each entry)

LDR R1, =BranchTable ; Get base address of branch table(0x284)

LDR R2,[R1,R0] ; Get the actual branch destination

BX R2 ; Branch to destination

ALIGN 4 ; Alignment control. The table has

BranchTable ; to be word aligned to prevent unaligned read ;table of each destination address

DCD Dest0

DCD Dest1

DCD Dest2

DCD Dest3

default\_case

stop B stop; Instructions for default case

Dest0 ldr r0, =10

stop1 B stop1 ; Instructions for case ‘0’

Dest1 ldr r0, =20

stop2 B stop2 ; Instructions for case ‘1’

Dest2 ldr r0, =30

stop3 B stop3 ; Instructions for case ‘2’

Dest3 ldr r0, =40

stop4 B stop4 ; Instructions for case ‘3’

END