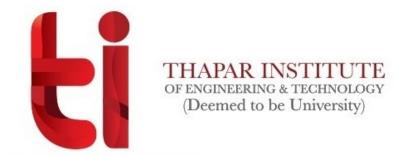
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING



EMBEDDED SYSTEMS Assignment 1

Submitted to,

Dr. Shireesh Kumar Rai

Asst. Professor

Submitted by,

T Raja Aadhithan

602162021

M.Tech (VLSI Design)

Task:

Write Different Programs in ARM assembly language to Set/Reset the flag register and apply all the conditional code suitable to perform the specific task.

Solution:

The various conditions are mentioned below in the table.

Condition	Meaning	Flag Status
EQ	equal	Z
NE	not equal	Z'
CS	carry set/unsigned higher or same	С
CC	carry Clear/unsigned lower	C'
MI	minus/negative	N
PL	plus/positive or Zero	N'
VS	overflow	V
VC	no overflow	V'
HI	unsigned higher	Z'C
LS	unsigned lower or same	Z or C'
GE	signed greater than or equal	N = V
LT	signed less than	N != V
GT	signed greater than	Z' (N = V)
LE	signed less than or equal	Z' or (N != V)
AL	always (unconditional)	ignored

In the Code there are 15 loops corresponding to 15 conditions above mentioned, only when a condition is satisfied it is moved to a loop on another half of the code. To ensure that any other false path is not taken an unconditional branch to the end of the program is inserted at the middle of the program which force exits the program in case of an error.

Registers R0, R1, R2, R3 is used for compare and arithmetic operations to set/reset the flags. Code R5 displays 0x76396869 only if all the loops are activated. R7 increments by 1 when it enters a loop, hence the value of R7 at the end of the program if successful should be 15 which is 0x0000 000F.

Code:

```
LOOP14 ADD R7, R7, #1
                                                   LDR R5,=0x76396869
                                                   BAL LOOP15
                                                                  ;ALWAYS
AREA PROGRAM, CODE, READONLY
   ENTRY
MAIN
                                           LOOP1
                                                   ADD R7, R7, #1
       MOVS R0, #0
                                                   ADDS R2, R2, #5
       MOV R1, #1
                                                   BPL LOOP2 ;PLUS OPERATION
       SUBS R2, R0, R1
       BMI LOOP1
                   ;MINUS OPERATION
                                           LOOP3
                                                   ADD R7, R7, #1
                                                   ADDS R3, R2, #1
LOOP2
                                                   BNE LOOP4 ; NOT EQUAL
       ADD R7, R7, #1
       SUBS R2, R2, R2
       BEQ LOOP3 ; EQUAL OPERATION LOOP5
                                                   ADD R7, R7, #1
                                                   CMN R3,R2
L00P4
       ADD R7, R7, #1
                                                   BLS LOOP6 ; LOWER THAN
       CMP R3, R2; R3-R2
       BHI LOOP5 ;HIGHER OPERATION
                                           LOOP7
                                                   ADD R7, R7, #1
                                                   ADDS R2,R2,#0xFF
LOOP6
       MOV R2,#5
                                                   BCC LOOP8 ; CARRY CLEAR
       ADD R7, R7, #1
       ADDS R2,R2,#0xFFFFFFF
                                           L00P9
                                                   ADD R7, R7, #1
       BCS LOOP7 ;CARRY SET
                                                   LDR R2,=0x7FFFFFF
                                                   ADDS R2, R2, #1
L00P8
       ADD R7, R7, #1
                                                   BVS LOOP10 ;OVERFLOW SET
       ADDS R2,R2,#0x0FFFFFF
       BVC LOOP9 ;OVERFLOW CLEAR LOOP11
                                                  ADD R7, R7, #1
                                                   MOV R3, R2
LOOP10 ADD R7,R7,#1
                                                   CMP R3,R2
       MOV R3,#1
                                                   BGE LOOP12; GREATER THAN/ EQUAL
       MOV R2,#5
       CMP R2,R3
                                           LOOP13 ADD R7, R7, #1
       BGT LOOP11 ;GREATER THAN
                                                   MOV R3, R2
                                                   CMP R3,R2
LOOP12 ADD R7, R7, #1
                                                   BLE LOOP14; LESSER THAN / EQUAL
       MOV R3,#1
       MOV R2,#5
       CMP R3,R2
                                           LOOP15 ADD R7,R7,#1
                                                   ADDS R0, R0, R0 ; TO END
       BLT LOOP13 ;LESSER THAN
                                           LOOP32 B LOOP32
       BAL LOOP15
                                                   END
```

Output:

```
Running with Code Size Limit: 32K
Load "C:\\Users\\User\\Documents\\Code-sync\\Keil\\ARM\\Assignment l\\Assignment_l.axf"

*** Restricted Version with 32768 Byte Code Size Limit

*** Currently used: 240 Bytes (0%)
```

Register	Value	
Current		_
R0	0x00000000	
R1	0x00000001	
R2	0x00000005	
R3	0x00000005	
R4	0x00000000	
R5	0x76396869	
R6	0x00000000	
R7	0x000000F	
R8	0x00000000	
R9	0x00000000	
R10	0x00000000	
R11	0x00000000	
R12	0x00000000	
R13 (SP)	0x00000000	
R14 (LR)	0x00000000	
R15 (PC)	0x000000E4	
□ CPSR	0x400000D3	
N	0	
Z	1	
С	0	
V	0	
1	1	
F	1	
ТТ	0	
М	0x13	
± SPSR	0x00000000	

```
Build target 'Target 1'
assembling flags.asm...
linking...
Program Size: Code=240 RO-data=0 RW-data=0 ZI-data=0
"Assignment_1.axf" - 0 Error(s), 0 Warning(s).
```

Result:

The output of the registers is as expected hence we can conclude that the program has run as expected.