Embedded Systems CSE_ 2263

1. **Write an ALP to add 2 32-bit numbers.**

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=0x12345678
    ldr r1,=0x00000002
    add r2,r0,r1
stop b stop
    end
```

2. **Write an ALP to add 2 64-bit numbers.**

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=0xF2345678;Lower byte of 1st number
    ldr r1,=0x20000001;Higher byte of 1st number
    ldr r2,=0x00000012;Lower byte of 2nd number
    ldr r3,=0x00000022;Higher byte of 2nd number
    adds r4,r1,r0     ;r4 stores the lower byte of answer
    adc r5,r2,r3      ;r5 stores the higher byte of answer
stop b stop
    end
```

3. **Write an ALP to subtract 2 32-bit numbers.**

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=0x12345678
    ldr r1,=0x00000002
    sub r2,r0,r1
stop b stop
    end
```

4. **Write ALP to subtract 2 64-bit numbers.**

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=0xF2345678;Lower byte of 1st number
    ldr r1,=0x20000001;Higher byte of 1st number
    ldr r2,=0x00000012;Lower byte of 2nd number
    ldr r3,=0x00000022;Higher byte of 2nd number
    subs r4,r1,r0     ;r4 stores the lower byte of answer
    sbc r5,r2,r3      ;r5 stores the higher byte of answer
stop b stop
    end
```

5. **Write an ALP for multiplication of 2 32-bit numbers.**

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=0x00000123  ;first operand
    ldr r1,=0x00000003  ; Second operand
    mul r2,r0,r1      ; r2 stores the result of multiplication
stop b stop
    end
```

6. **Write an ALP for multiplication of 2 64-bit numbers.**

```
        AREA aa,CODE,READONLY
        entry
        ldr r0,=0x00000005 ;higher word of first number
        ldr r1,=0x00000000   ;lower word of first number
        ldr r2,=0x00000005   ;higher word of second number
        ldr r3,=0x00000000   ;lower word of second number

        mul r3,r0,r3
        mla r3,r2,r1,r3
        umull r4,r5,r0,r2
        add r5,r3,r5          ;result will be stored in r4 and r5

stop    b stop
        end
```

7. **Write an ALP for division of 2 32-bit numbers (a/b).**

```
     AREA aa,CODE,READONLY
      entry
      ldr r0,=0x00000002   ;divider (b)
      ldr r1,=0x0000000F   ;dividend (a)
      mov r2,r1            ;reminder
      mov r3,#00           ;quotiont

loop  cmp r2,r0
      blt stop
      add r3,r3,#1
      sub r2,r2,r0
      bal loop


stop  b stop
      end
```

8. **Write an ALP to transfer block of data from one memory to another memory.**

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=0x40000000    ;Base adress of the bloack adress of data to be transferred
    ldr r1,=0x40000008    ;the adress which stores the transfered data
    mov r3,#0x00
    ldr r2,[r0]
    str r3,[r0]
    str r2,[r1]
stop b stop
    end
```

9. **Write an ALP to find the greatest of 2 32-bit numbers.**

```
     AREA aa,CODE,READONLY
      entry
      ldr r0,=0x00000005     ;First number
      ldr r1,=0x00000006     ;second number
      cmp r0,r1
      movge r2,r0
      cmp r1,r0
      movge r2,r1            ;Final result will be stores in r2
stop b stop
      end
```

10. Write an ALP to find the greatest of 5 32-bit numbers.

```
     AREA aa,CODE,READONLY
      entry
      ldr r0,=0x00000002    ;First number
      ldr r1,=0x00000003    ;Second number
      ldr r2,=0x00000001    ;Third Number
      ldr r3,=0x00000006    ;Fourth number
      ldr r4,=0x00000005    ;Fifth number
      mov r5,r0             ;Greatest number will be stored in r5
      cmp r1,r5
      movge r5,r1
      cmp r2,r5
      movge r5,r2
      cmp r3,r5
      movge r5,r3
      cmp r4,r5
      movge r5,r4
stop b stop
      end
```

11. Write an ALP to find the greatest of 5 32-bit number with loop and memory.

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=0x40000000      ;starting adress of array
    mov r2,#0x05            ;Size of the array
    ldr r3,[r0]             ;stores the Final result(Greatest of 5 numbers in array)
label
     ldr r4,[r0],#4
     cmp r4,r3
     movge r3,r4
     subs r2,r2,#0x01
     bne label
stop b stop
    end
```

12. ALP to test for the equivalence of 2 number using 3 methods.

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=0x00000002   ;First number
    ldr r1,=0x00000002   ;Second number
    ;First method
    teq r0,r1
    moveq r2,#0x01       ;r2 stores 1 if the number stored in r0 and in r1 are equal

    ;Second method
    eor r3,r0,r1
    moveq r4,#0x01      ;r4 stores 1 if the number stored in r0 and in r1 are equal

    ;Third method
    subs r5,r0,r1
    moveq r6,#0x01       ;r6 stores 1 if the number stored in r0 and in r1 are equal
stop b stop
    end
```

13. Write an ALP to set the bits 2,4,8 in the register r4.

```
    AREA aa,CODE,READONLY
    entry
    ldr r4,=0x00011111  ;storing the number
    ldr r0,=0x0000008A
    orr r4,r4,r0        ;r4 stores the result after setting the 2,4,8th bit

stop b stop
    end
```

14. Write an ALP to clear the contents of register r5 using 4 different methods.

```
    AREA aa,CODE,READONLY
    entry
    ;first method
    ldr r5,=0x12345678
    and r5,r5,#0x00000000

    ;second method
    ldr r5,=0x12345678
    bic r5,r5,#0xffffffff

    ;third method
    ldr r5,=0x12345678
    sub r5,r5,r5

    ;fourth method
    ldr r5,=0x12345678
    mov r5,r5,lsr #32


stop b stop
    end
```

15. Write an ALP to check whether the number is +ve or -ve using TST and shift instruction.

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=0x00000004  ;stores the number to be checked wether +ve or -ve
    mov r2,#0x00        ;Stores 1 if the number stored in r0 is +ve otherwise it will store 0
    tst r0,#0x80000000
    moveq r2,#0x01

    ;using shift instruction
    ldr r3,=0x80000005   ;stores the number to be checked wether +ve or -ve
    mov r4,#0x00         ;Stores 1 if the number stored in r3 is +ve otherwise it will store 0
    mov r5,r3,lsr #31
    teq r5,#0x00000001
    movne r4,#0x01

stop b stop
    end
```

16. Write an ALP to check whether the number is even or odd using TST and shift instruction.

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=0x00000004  ;stores the number to be checked wether even or odd
    mov r2,#0x00        ;Stores 1 if the number stored in r0 is even otherwise it will store 0
    tst r0,#0x01
    moveq r2,#0x01

    ;using shift instruction
    ldr r3,=0x00000005   ;stores the number to be checked wether even or odd
    mov r4,#0x00          ;Stores 1 if the number stored in r3 is even otherwise it will store 0
    mov r5,r3,lsl #31
    teq r5,#0x80000000
    movne r4,#0x01

stop b stop
    end
```

17. Write an ALP to generate first 10 odd numbers.

```
     AREA aa,CODE,READONLY
     entry
     ldr r0,=0x00000001     ;First odd number
     ldr r1,=0x0000000A     ;Counter
     ldr r2,=0x40000000     ;Starting adress of the array
     BL odd
stop b stop
odd
      str r0,[r2],#4
      add r0,r0,#0x02
      subs r1,r1,#0x01
      bne odd
      mov pc,lr

     end
```

18. Write an ALP to generate first 10 even numbers.

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=0x00000000     ;First even number
    ldr r1,=0x0000000A     ;Counter
    ldr r2,=0x40000000     ;Starting adress of the array
    BL even
stop b stop
even
     str r0,[r2],#4
     add r0,r0,#0x02
     subs r1,r1,#0x01
     bne even
    mov pc,lr

    end
```

19. Write an ALP to GCD of 2 32-bit numbers.

```
        AREA gcdcalc, CODE, READONLY
         ENTRY
         MOV  R0,#30 ; number 1
         MOV R1,#45 ; number 2    (finally it stores the gcd of 2 32 bit numbers)
gcd
while   CMP R0,R1
         BEQ endw
         BGT cond1
         B cond2
cond1   SUB R0,R0,R1
         B gcd
cond2   SUB R1,R1,R0
         B gcd
         B while
endw
stop    B stop
         END
```

20. Write an ALP to LCM of 2 32-bit numbers.

```
        AREA gcdcal, CODE, READONLY
        ENTRY
        MOV  R0,#0x12 ; test values
        MOV R1,#0x0f ; test values
        mul r3,r0,r1 ;dividend(a)
gcd
while   CMP R0,R1
        BEQ endw
        BGT cond1
        B cond2
cond1   SUB R0,R1
        B gcd
cond2   SUB R1,R0
        B gcd
        B while
endw
     mov r4,r0     ;devider(b)
     mov r5,#0x00 ;r5 stores Quotiont (q)    (lcm of 2 numnbers)
     mov r6, r3    ;r6 stores reminder (r)
devide  cmp r6,r4
         blt stop
         add r5,r5,#1
         sub r6,r6,r4
         bal devide
stop    B stop
        END
```

21. Write an ALP to find a factorial of the given number.

```
         AREA aa,CODE,READONLY
          entry
          ldr r0,=0x00000005          ;Loading the number
          mov r1,r0
label
          subs r1,r1,#0x01            ;Decrementing  a number by 1
          mul r2,r0,r1                ;multiplying previous result by decremented number
          mov  r0,r2
          teq r1,#0x01
          bne label

stop     b stop                      ;At final result will be stored in r0
          end
```

22. Write an ALP to generate first 10 numbers of Fibonacci numbers.

```
    AREA AA,CODE,READONLY
    ENTRY
    LDR R0 ,=0X00000000 ;FIRST NUMBER
    LDR R1 ,=0X00000001  ;SECOND NUMBER
    LDR R2 ,=0X00000009 ;COUNTER
    LDR R3 ,=0X40000000 ;ADRESS
    STR R0,[R3],#4
    BL FBS
STOP B STOP

FBS STR R1,[R3],#4
    MOV R4,R1
    ADD R1,R1,R0
    MOV R0,R4
    SUBS R2,R2,#0X01
    BNE FBS
    MOV PC,LR
    END
```

23. Write an ALP to swap the contents of two register in 3 ways.

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=0x00011234
    ldr r1,=0x00002123
    ;First way of swapping
    mov r2,r1
    mov r1,r0
    mov r0,r2
    ;Second way of swapping
    ldr r0,=0x00011234
    ldr r1,=0x00002123
    add r0,r0,r1
    sub r1,r0,r1
    sub r0,r0,r1
    ;Third way of swapping
    ldr r0,=0x00011234
    ldr r1,=0x00002123
    eor r0,r0,r1
    eor r1,r0,r1
    eor r0,r0,r1

stop b stop
    end
```

24. Write an ALP to search a given character in given string.

```
cr EQU 0x0d
    AREA aa,CODE,READONLY
    entry
    ldr r0,=array
    ldr r4,="h"     ;Stores the character wich is to be seacrched in given string
label
    ldrb r1,[r0],#1
    cmp r1,#cr
    beq stop
    cmp r4,r1
    beq store
    bal label

stop b stop
store
     mov r5,#0xFF      ;Register r5 nstores the 0xFF if the searched charecter is found in the given string
array  DCB "hello word",cr   ;Given string
    end
```

25. Write an ALP to reverse the given string.

```
cr EQU 0x0d
     AREA aa,CODE,READONLY
     entry
     ldr r0,=array
     ldr r2,=0x40000000
     ldr r3,=array
golast
     ldrb r1,[r0],#1
     cmp r1,#cr
     bne golast

     ldrb r1,[r0],#-2

main
     ldrb r1,[r0],#-1
     strb r1,[r2],#1
     cmp r1,r3
     beq stop
     bal main



stop b stop
array DCB "HELLO",cr
     end
```

26. Write an ALP to check given string is palindrome or not.

```
cr EQU 0x0d
    AREA aa,CODE,READONLY
    entry
    ldr r0,=array
    ldr r1,=array
    mov r2,#0x00     ;Stores the length of the string
    bl length
    sub r2,r2,#1
    mov r2,r2,lsr #1
    ldrb r3,[r0],#-1
    ldrb r3,[r0],#-1
main
    ldrb r4,[r1],#1
    ldrb r3,[r0],#-1
    cmp r4,r3
    bne store
    subs r2,r2,#1
    bne main
stop1 b stop1
length
       ldrb r3,[r0],#1
       add r2,r2,#1
       cmp r3,#cr
       moveq pc,lr
       bal length
store
     mov r5,#0xFF    ;R5 stores FF if the given string is not palindrom
stop b stop
array DCB "AABB",cr
    end
```

27. Write an ALP to convert the alphabet from lower case to uppercase.

```
    AREA aa,CODE,READONLY
    entry
    ldr r0,=array
    ldr r2,=0x40000000
    mov r3,#00          ;Stores the length of the string
    mov r4,r0
    bl find_length


main  ldrb r1,[r0],#1
      sub r1,r1,#0x20
      strb r1,[r2],#1
      subs r3,r3,#1
      bne main

stop b stop
find_length
      ldrb r5,[r4],#1
      cmp r5,#cr
      moveq pc,lr
      add r3,r3,#1
      bal find_length

array DCB "helloword",cr
    end
```

28. Write an ALP to count the number of blank spaces in a given string.

```
cr  EQU 0x0d
    AREA aa,CODE,READONLY
    entry
    ldr r0,=array
    mov r1,#0x00   ;stores the count of blank spaces in the string
    mov r3,#0x20

main
     ldrb r2,[r0],#1
     cmp r2,#cr
     beq stop
     cmp r2,r3
     addeq r1,r1,#1
     bal main
stop b stop
array DCB "A A A A",cr
    end
```