Microcontroller programs

1) Generating Series 5,10,15,20,25

```
Area Example, CODE, ReadOnly
    ENTRY
    MOV R4, #5
    MOV R0, #1
    MOV R1, #2
    MOV R2, #3
    MOV R1, #0
LOOP MLA R3, R0, R1, R2
    ADD R6, R3
    SUBS R4, #1
    BNE LOOP
L   B L
    END
```

2) Generating Series 2,4,6,8,10

```
Area Example, Code, ReadOnly
    ENTRY
    MOV R4, #5
    MOV R0, #1
    MOV R1, #1
    MOV R2, #2
    MOV R5, #1
    MOV R6, #0
LOOP MLA R3, R0, R1, R2
    ADD R6, R3
    SUBS R4, #1
    BNE LOOP
L   B L
    END
```

3) Generating Odd numbers

```
Area Example, Code, Readonly
    ENTRY
    MOV R4, #5
    MOV R0, #1
    MOV R1, #1
    MOV R2, #1
    MOV R6, #1
LOOP MLA R3, R0, R1, R2
    ADD R6, R3
    SUBS R4, R1
    BNE LOOP
L   B L
    END.
```

4) Generating factorial of a number

```
Area FACT, CODE, READONLY
START
    MOV R0, #5
    MOV R1, #01
    CMP R0, #0
    BEQ STOP
    MOV R1, R0
NEXT  SUBS R0, #1
    CMP R0, #0
    BEQ STOP
    MUL R2, R1, R0
    MOV R1, R2
    B NEXT
STOP
    NOP
    END.
```

5) ALP Program to find largest number in an array.

```
AREA, LNO, CODE, READONLY
    ENTRY
    ORG 0000
    MOV R0, #30H
    MOV R1, #05H
    MOV B, #00H
BACK: MOV A, @R0
    CJNE A, B, LOOP
LOOP: JC LOOP1
    MOV B, A
LOOP1: INC R0
    DJNZ R1, BACK
    Sjump $
    END.
```

```
AREA  AREA LARGEST, CODE, READONLY
      ENTRY
      LDR R0, =VALUE1
      LDR, R5, [R0], #4
      SUBS R5, R5, #1
      LDR R2, [R0], #4
LOOP
      LDR R4, [R0], #4
      CMP R2, R4
      BHI LOOP1
      MOV R2, R4
LOOP1
      SUBS R5, R5, #1
      CMP R5, #0
      BNE LOOP
      LDR R4, =RESULT
      STR, R2, [R4]
HERE  B HERE
```

```
VALUE1 DCD 0x00004, 0x00000001, 0xAAA1025B,
0x11111111, 0xFFFFFFFF
      AREA DATA2, DATA, READWRITE
RESULT DCD 0x0
      END
```

Jump List syntax and explain each instructions with example.

a) ~~Shift instructions~~ Shift instructions

Syntax: Logical Shift left by register. Rm, LSL Rs.

~~Logical Shift left by immediate Rm, LSL #Shift_imm~~

~~Logical Shift left by register Rm, LSL Rs~~

```
PRE  r5=5
     r7=8
MOV r7, r5, LSL #2   let r7 = r5 #4  (r5 << 2)
POST r5=5
     r7=20 //
```

b) Arithmetic instructions.

Syntax: <Instruction>{<cond>}{S} Rd, Rn, N.

Exp: It implements arithmetic operations such as add, sub, mul, div, etc of 32 bit signed and unsigned values.

Eg.

```
PRE  r0= 0x00000000
     r1 = 0x00000002
     r2=0x00000001
     SUB r0, r1, r2
POST  r0= 0x00000001
```

c) Logical ~~operators~~ instructions

Logical instructions perform bitwise logical operations on the two source registers

Syntax: <instruction>{<cond>}{S} Rd, Rn, N

Example:

```
PRE   r0 = 0x00000000
      r1 = 0x02040608
      r2 = 0x10305070
      ORR r0, r1, r2
POST  r0 = 0x12345678
```

d) Multiply Instructions. Multiply instructions multiply the contents of a pair of registers and depending upon the instruction, accumulate the results in with another register. The long multiplies accumulate onto a pair of registers representing a 64-bit value.

Syntax: MLA {<cond>}{S} Rd, Rm, Rs, Rn

MUL {<cond>}{S} Rd, Rm, Rs.

~~DALA~~

Syntax: <instruction>{<cond>}{S} RdLo, RdHi, Rm, Rs.

Example:

```
PRE   r0 = 0x00000000
      r1 = 0x00000002
      r2 = 0x00000002
      MUL r0, r1, r2 ; r0 = r1 * r2
POST
      r0 = 0x00000004
      r1 = 0x00000002
      r2 = 0x00000002
```

e) Comparison Instructions

The comparison instructions are used to compare or test a register with a 32 bit value. They update the cpsr flag bits according to the result, but don't affect other registers

Syntax: <instruction>{<cond>} Rn, N

Example:

```
PRE   cpsr = nzcvqift_USER
      r0 = 4
      r1 = 4
      CMP r0, r1
POST  cpsr = nZCvqiFt_USER
```