1.

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
```

2.

Write a C program to Light a LED connected to port C12.

```
//
// Smpl_GPIO_LED : GPC12 to control on-board LEDs
//                 low-active output to control Red LEDs
//
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvSYS.h"

void Init_LED() // Initialize GPIO pins
{
  DrvGPIO_Open(E_GPC, 12, E_IO_OUTPUT); // GPC12 pin set to output mode
  DrvGPIO_SetBit(E_GPC, 12);   // output Hi to turn off LED
}
int main (void)
```

```
{
  UNLOCKREG(); // unlock register for programming
  DrvSYS_Open(48000000);// set to run at 48MHz
  // 12MHz crystal input, PLL output 48MHz
  LOCKREG();   // lock register  from programming

  Init_LED(); // Initialize LEDs (four on-board LEDs)

  while (1) //   forever loop to keep flashing four LEDs one at a time
  {
      DrvGPIO_ClrBit(E_GPC, 12); // output Low  turn on LED
      DrvSYS_Delay(30000); // delay
      DrvGPIO_SetBit(E_GPC, 12); // output Hi  turn off LED
      DrvSYS_Delay(300000);  // delay
  }

}
```

6.

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
```

9.

Use Labels to calculate the sum of say 10 numbers

```
;additon 32 bit



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
DataIn EQU 0x20000000
Sum EQU 0x20000040

__main
LDR r0,=DataIn; Get the address of variable 'DataIn'
MOVS r1, #10 ; loop counter
MOVS r2, #0 ; Result - starting from 0
add_loop
LDM r0!,{r3} ; Load result and increment address
ADDS r2, r3 ; add to result
SUBS r1, #1 ; increment loop counter
BNE add_loop
LDR r0,=Sum ; Get the address of variable 'Sum'
STR r2,[r0] ; Save result to Sum
stop B stop
END
```

11.

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
```

12.

Write a C program to beep a buzzer connected to port B11.

```
//
// Smpl_GPIO_Buzzer : GPB11 low-active output control Buzzer
// Note: Nu-LB-NUC140 R1 should be 0 ohm
//
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvADC.h"
```

```
int main (void)
{
UNLOCKREG();   // unlock register for programming
 DrvSYS_Open(48000000);     // set System Clock to run at 48MHz
LOCKREG();   // lock register from programming

DrvGPIO_Open(E_GPB, 11, E_IO_OUTPUT); // initial GPIO pin GPB11 for controlling Buzzer

while(1) {
 DrvGPIO_ClrBit(E_GPB,11); // GPB11 = 0 to turn on Buzzer
DrvSYS_Delay(100000);    // Delay
DrvGPIO_SetBit(E_GPB,11); // GPB11 = 1 to turn off Buzzer
DrvSYS_Delay(100000);    // Delay
}
}
```

13.

```
import RPi.GPIO as GPIO
import time

# Set the GPIO mode to BCM
GPIO.setmode(GPIO.BCM)

# Set up GPIO pin 17 as an output pin
led_pin = 17
GPIO.setup(led_pin, GPIO.OUT)

try:
    while True:
        # Turn on the LED
        GPIO.output(led_pin, GPIO.HIGH)
```

```
        time.sleep(1)  # Wait for 1 second

        # Turn off the LED
        GPIO.output(led_pin, GPIO.LOW)
        time.sleep(1)  # Wait for 1 second

    except KeyboardInterrupt:
        # Clean up GPIO configuration on keyboard interrupt (Ctrl+C)
        GPIO.cleanup()
```

15.

Same as 2