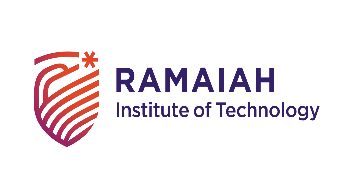
**M S RAMAIAH INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF TELECOMMUNICATION ENGINEERING**



**EMBEDDED SYSTEM DESIGN**

**LAB MANUAL**

**In-charge Faculty**

**Dr. S G SHIVAPRASAD YADAV**

**Prerequisite Courses:**

Microcontrollers, Operating Systems, Digital Electronics

**Course Objectives**

1. To apply the knowledge of microcontroller concepts by getting familiar to use the editor, assembler, compiler, linker, loader and debugger tool chain of Keil IDE
2. To make the students design the software for microcontroller systems by programming using Assembly and Embedded C on Keil IDE
3. To provide the students hands-on experience by interfacing the microcontroller with other devices such as LEDs, Switches, Buzzer, Motor, Interrupts, Keypad, LCD, ADC /DAC, Terminal etc. and build systems using internal Timers, Serial port, Interrupt. the students conduct experiments
4. To make the students understand the procedure for simulating and debugging the software
5. To make the students understand how to validate and debug a microcontroller-based system

**Course Outcomes**

1. Develop “Assembly” and "C" programs for ARM Cortex M3and M4 microcontrollers for various tasks like data transfer, arithmetic and logical operations (PO 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 and PSO 1, 2 and 3)
2. Develop applications to configure the interrupts, factorial of a number, ascending, descending operations and so on. (PO 1, 2, 3, 4, 5, 9,11, 12 and PSO 1, 2 and 3)
3. Develop the Applications to configure various peripherals such as timers, serial communications, and interrupt driven microcontroller applications using Keil IDE (PO 1,2, 3, 4, 7, 11, 12 and PSO 1,2,3)
4. Demonstrate the interfacing of ARM Cortex M3microcontroller with external interfaces like ADC, DAC, Motors, Keypad and so on by integrating the software and hardware modules to build microcontroller based systems using Keil IDE (PO 1, 2, 3, 4, 7, 9,11,12) (PSO 1,2,3)
5. Demonstrate teamwork while building inter-disciplinary microcontroller based systems and proficiency to document their work in a technical record/report (PO 1,2,3, 4,5,6,7, 11,12) (PSO 1,2,3)

|  |  |  |
| --- | --- | --- |
| **Lesson**  **Plan** | **List of Programs** | **Duration** |
| 01 | Program to Add 64-bit no’s and Program to Subtract 64-bit no’s. | 2 hours |
| 02 | Program to Multiply 2nos. | 2 hours |
| 03 | Program to extract bit 24 and 25 status and to bit clear bit 15 and 16 bits. | 2 hours |
| 04 | Program to swap 10bytes locations starting from 2000 0000 to 2000 0100. | 2 hours |
| 05 | Program to copy 10 bytes from locations starting from 2000 0000 to 2000 0200 | 2 hours |
| 06 | Program to implement a factorial of given no | 2 hours |
|  | **Interfacing programs** | 2 hours |
| 07 | Program to light a Led connected port C15 using an interrupt | 2 hours |
| 08 | Program to toggle a led connected to port C12.and also to buzzer when interrupt is given at port b11 | 2 hours |
| 09 | Program to scan the keyboard and to buzzer b11 when key 3 is pressed and light c12 for a brief period when key 2 is pressed |  |
| 10 | Program to display on LCD a string “welcome to msrit” | 2 hours |
| 11 | Program to use ADC to display the analog voltage connected to variable resistor on channel 7 of port A | 2 hours |
| 12 | Program to Display the analog voltage in terms of analog on the LCD of the variable resistor connected to ADC channel 7 | 2 hours |
| 13 | Program to scan the key board and display the key that is pressed on lcd or seven segment | 2 hours |
| 14 | Program to display on seven segment counts 0 to 99 and to toggle LEDs R G B one after the other in sequence | 2 hours |

**LAB 1**

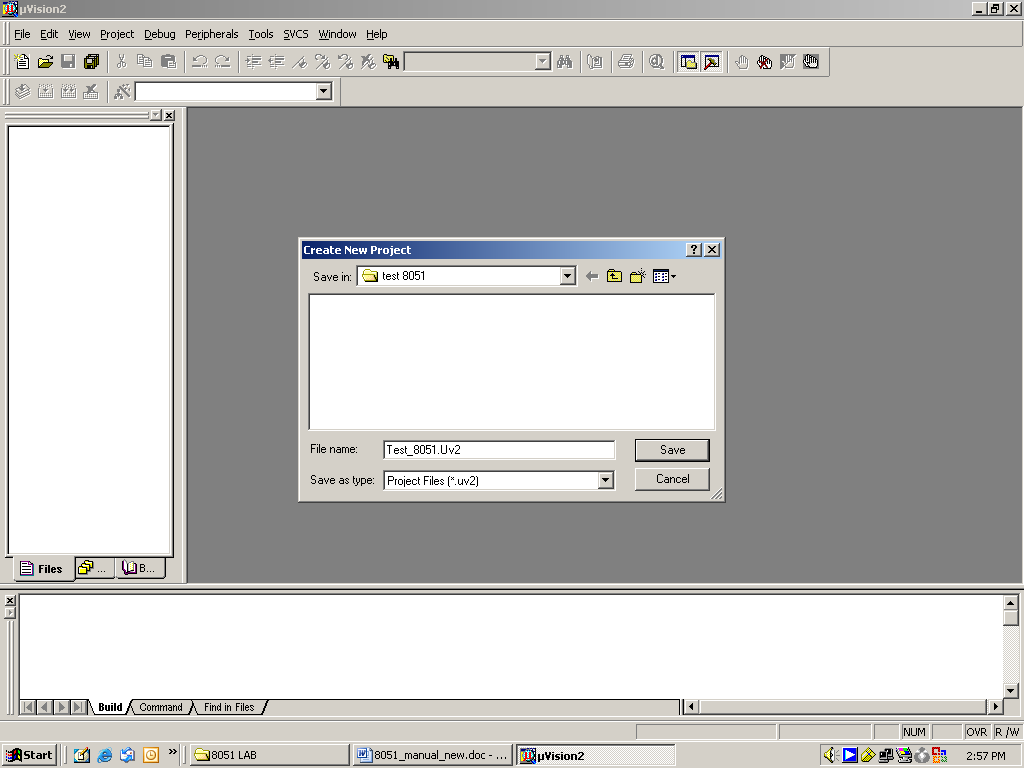
**Working with KEIL uVision3:**

**Introduction to KEIL uVision2 IDE**

**Step1:**

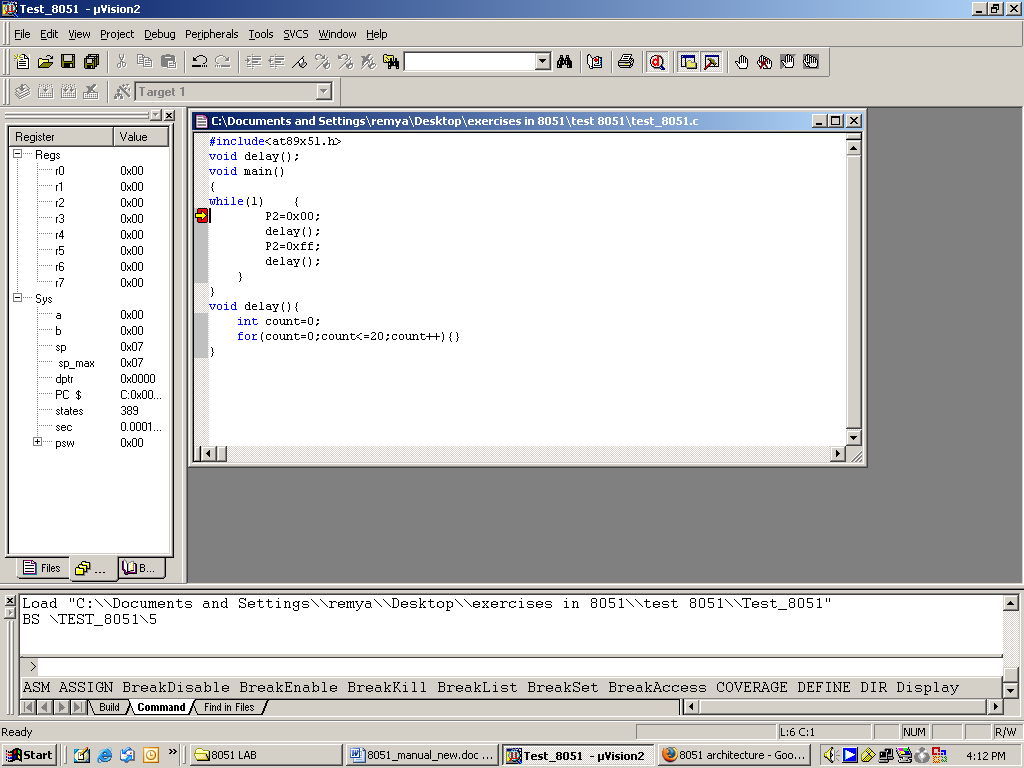
**Open Keil uvision2 IDE from your PC , start-> program->keil uvision2.**

**You need a project to build the program. Follow the steps below to create a new project. From KEIL uvision2, select Project>New project ; the New Project dialog box will open as shown below.**

****

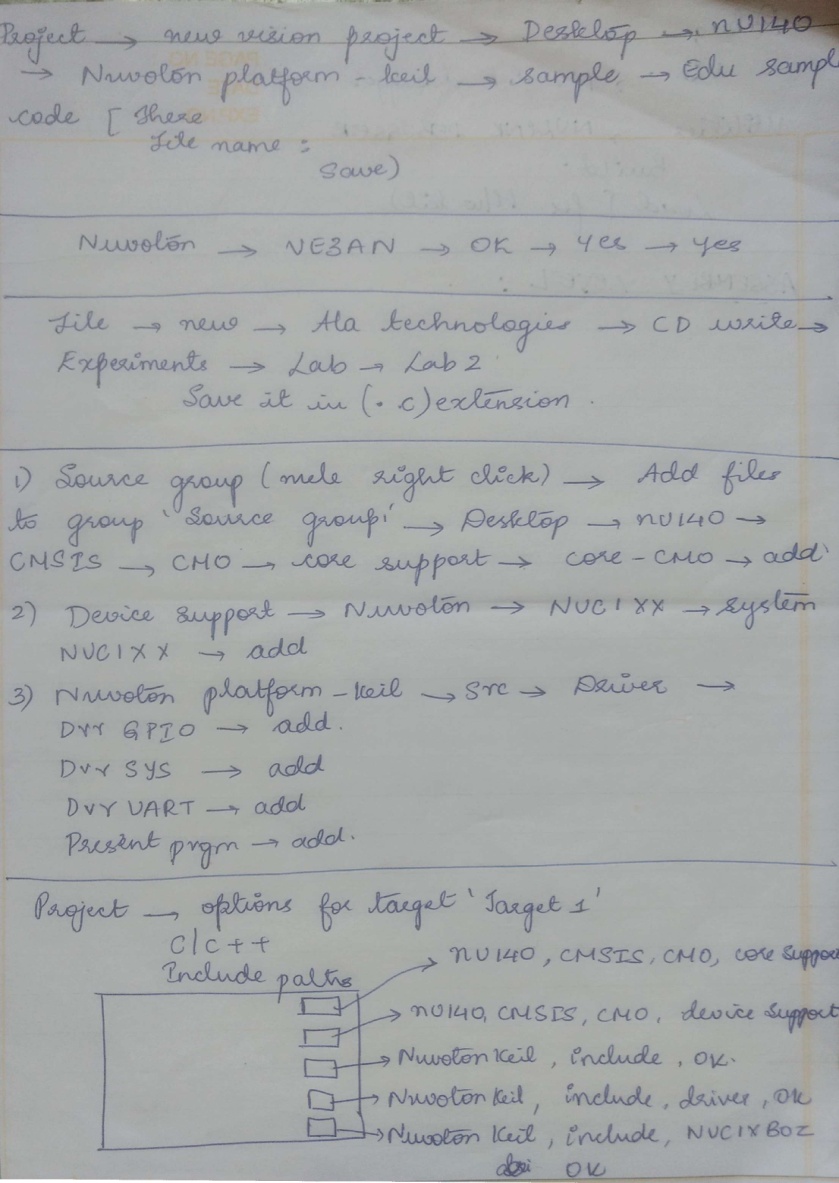
* **DEBUG SESSION**
* **Start the debugging session as follows**

**Debug -> start or stop debug session ( ctrl + F5). You may get a window as shown here.**

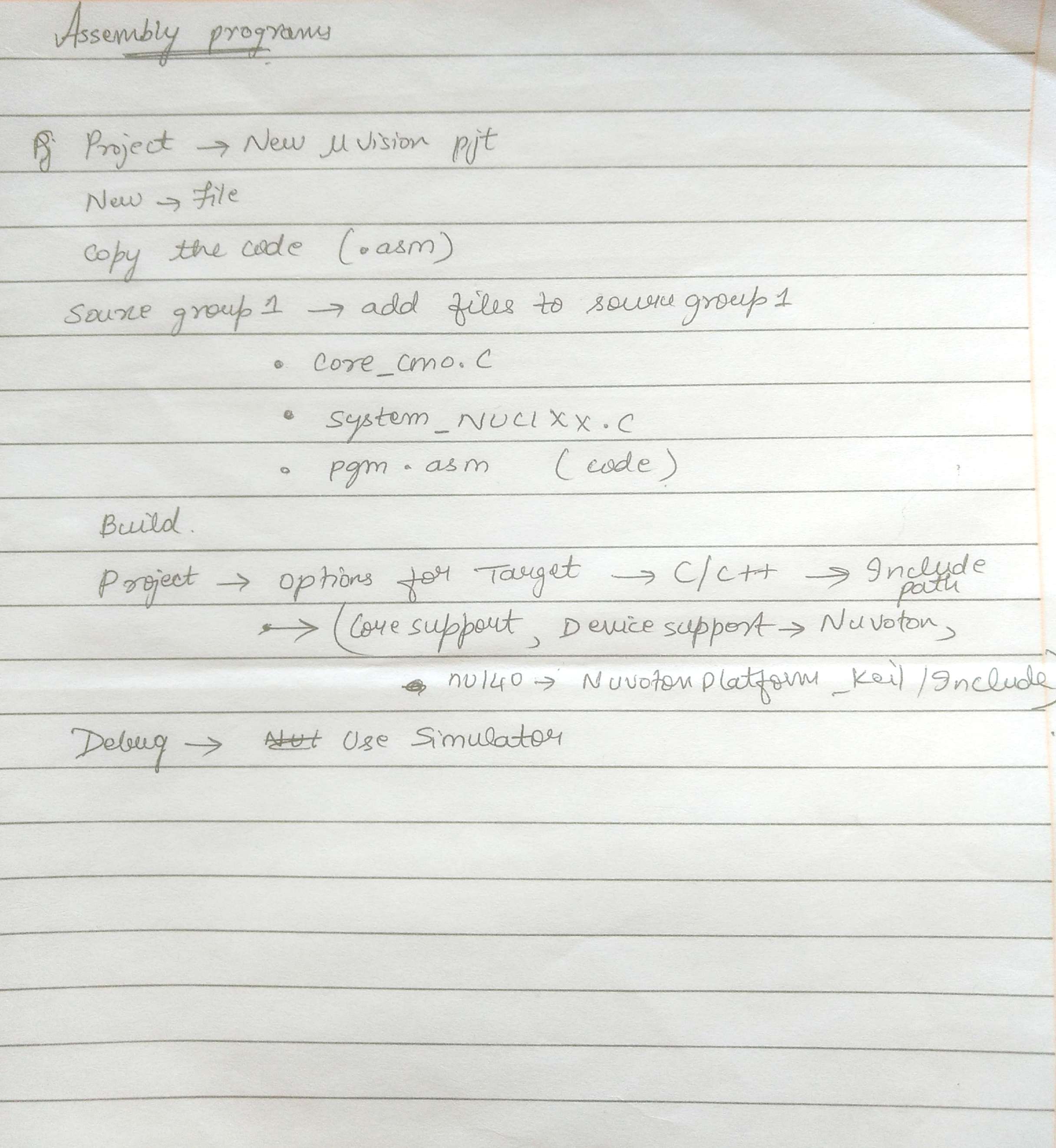
****

**You can see the status of all general purpose registers ( r0 – r7) , system registers( Accumulator and B register ), Stack pointer, data pointer(dptr), program counter(PC), Program status word(PSW).**

**Follow the Below Steps for Interfacing Programs:**



**Follow the Below Steps for Assembly Programs:**



**ASSEMBLY LANGUAGE PROGRAMS**

1. **Write a Program to Add 64 bit no’s.**

PRESERVE8

THUMB

AREA |.text|,CODE,READONLY

EXPORT \_\_main

\_\_main

LDR r0,=0xffffffff

LDR r1,=0x00000000

LDR r3,=0x00000033

LDR r4,=0x00000001

ADDS r0,r0,r3

ADCS r1,r1,r4

stop B stop

END

**R1 1234 5678 9876 5432 RO**

**R2 1234 5678 9786 5321 R3**

1. **Write a Program to Subtract 64 bit no’s.**

PRESERVE8

THUMB

AREA |.text|,CODE,READONLY

EXPORT \_\_main

\_\_main

LDR r0,=0x10000001

LDR r1,=0x00000000

LDR r2,=0x00000003

LDR r3,=0x00000000

SUBS r0,r0,r2

SBCS r1,r1,r3

stop B stop

END

**3) Write a Program to Multiply 2nos.**

PRESERVE8

THUMB

AREA |.text|,CODE,READONLY

EXPORT \_\_main

\_\_main

LDR r0,=0x00000008

LDR r1,=0x00000009

LDR r2,=0x00000000

loop1 CMP r0,#0

BEQ stop

ADDS r2,r2,r1

SUBS r0,r0,#1

B loop1

stop B stop

END

**4) Write a Program to extract bit 24 and 25 status.**

PRESERVE8

THUMB

AREA |.text|,CODE,READONLY

EXPORT \_\_main

\_\_main

LDR r0,=0xAB567832

LDR r1,=0x00000000

LDR r2,=0x00000000

LSLS r0,r0,#8

ADCS r1,r1,r1

LSLS r0,r0,#1

ADCS r2,r2,r2

stop B stop

END

**5) Write a Program to bit clear bit 15 and 16 bits.**

PRESERVE8

THUMB

AREA |.text|, CODE, READONLY

EXPORT \_\_main

\_\_main

LDR r1,=0xffffffff

LDR r2,=0x00000003

LSLS r2,r2,#14

BICS r1,r1,r2

stop b stop

end

1111 1111 1111 1111 1111 1111 1111 1111

0000 0000 0000 0001 1000 0000 0000 0000

1111 1111 1111 1110 0111 1111 1111 1111

1111 1111 1111 1110 0111 1111 1111 1111

**6) Write a Program to swap 10bytes locations starting from 2000 0000 to 2000 0100.**

PRESERVE8

THUMB

AREA |.text|,CODE,READONLY

EXPORT main

main

LDR r0,=0x20000000

LDR r1,=0x20000020

MOVS r2,#10

loop

LDRB r3,[r0]

LDRB r4,[r1]

STRB r3,[r1]

STRB r4,[r0]

ADDS r0,r0,#1

ADDS r1,r1,#1

SUBS r2,r2,#1

BNE loop

stop B stop

END

**7) Write a Program to copy 10 bytes from locations starting from 2000 0000 to 2000 0200.**

PRESERVE8

THUMB

AREA |.text|, CODE, READONLY

EXPORT main

main

LDR r0,=0x20000000 ; Source address

LDR r1,=0x20000020 ; Destination address

MOVS r2, #10 ; number of bytes to copy

copy\_loop

LDRB r3, [r0] ; read 1 byte

ADDS r0, r0, #1 ; increment source pointer

STRB r3, [r1] ; write 1 byte

ADDS r1, r1, #1 ; increment destination pointer

SUBS r2, r2, #1 ; decrement loop counter

BNE copy\_loop ; loop until all data copied

stop B stop

END

**8)Write a Program to implement case.**

PRESERVE8

THUMB

AREA |.text|, CODE, READONLY

EXPORT main

main

LDR R0, =2

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’

stop5 B stop5

END

**9)Write a Program to implement a factorial of given no.**

PRESERVE8

THUMB

AREA |.text|, CODE, READONLY

EXPORT main

main

MOVS r0,#5

MOVS r1,r0

SUBS r1,r1,#1

loop

MULS r0,r1,r0

SUBS r1,r1,#1

CMP r1,#1

BNE loop

stop B stop

END

**Interfacing Programs**

**10) Write a program to light a Led connected port C15 using an interrupt.**

#include <stdio.h>

#include "NUC1xx.h"

#include "Driver\DrvSYS.h"

#include "Driver\DrvGPIO.h"

void Init\_LED() // Initialize GPIO pins

{

DrvGPIO\_Open(E\_GPC, 15, E\_IO\_OUTPUT); // GPC15 pin set to output mode

DrvGPIO\_SetBit(E\_GPC, 15); // Goutput Hi to turn off LED

}

int main (void)

{

UNLOCKREG(); // unlock register for programming

DrvSYS\_Open(48000000);// set System Clock to run at 48MHz

// 12MHz crystal input, PLL output 48MHz

LOCKREG(); // lock register from programming

Init\_LED(); // Initialize LEDs (four on-board LEDs below LCD panel)

while (1) // forever loop to keep flashing four LEDs one at a time

{

DrvGPIO\_ClrBit(E\_GPC, 15); // output Low to turn on LED

DrvSYS\_Delay(300000);

DrvGPIO\_SetBit(E\_GPC, 15);

DrvSYS\_Delay(300000);

}

}

**11) Write a program to toggle a led connected to port C12.and also to buzzer when interrupt is given at port b11.**

#include <stdio.h>

#include "NUC1xx.h"

#include "Driver\DrvSYS.h"

#include "Driver\DrvGPIO.h"

void Init\_LED() // Initialize GPIO pins

{

DrvGPIO\_Open(E\_GPC, 12, E\_IO\_OUTPUT);

DrvGPIO\_SetBit(E\_GPC, 12);

DrvGPIO\_Open(E\_GPC, 13, E\_IO\_OUTPUT);

DrvGPIO\_SetBit(E\_GPC, 13);

DrvGPIO\_Open(E\_GPC, 14, E\_IO\_OUTPUT);

DrvGPIO\_SetBit(E\_GPC, 14);

DrvGPIO\_Open(E\_GPC, 15, E\_IO\_OUTPUT);

DrvGPIO\_SetBit(E\_GPC, 15);

}

void Init\_Buzzer()

{

DrvGPIO\_Open(E\_GPB, 11, E\_IO\_OUTPUT);

DrvGPIO\_SetBit(E\_GPB, 11);

}

int main (void)

{

int i=0,j=12;

UNLOCKREG(); // unlock register for programming

DrvSYS\_Open(48000000);// set System Clock to run at 48MHz

// 12MHz crystal input, PLL output 48MHz

LOCKREG(); // lock register from programming

Init\_LED(); // Initialize LEDs (four on-board LEDs below LCD panel)

Init\_Buzzer();

while (i<10&&j<16) // forever loop to keep flashing four LEDs one at a time

{

i++;

DrvGPIO\_ClrBit(E\_GPC, j); // output Low to turn on LED

DrvSYS\_Delay(300000); // delay

DrvGPIO\_SetBit(E\_GPC, j); // output Hi to turn off LED

DrvSYS\_Delay(300000); // delay

DrvGPIO\_ClrBit(E\_GPB, 11);

DrvSYS\_Delay(300000);

DrvGPIO\_SetBit(E\_GPB, 11);

DrvSYS\_Delay(300000);

j++;

if(i%4==0)

j=12;

}

}

**12) Write a program to scan the keyboard and to buzzer b11 when key 3 is pressed and light c12 for a brief period when key 2 is pressed.**

#include <stdio.h>

#include "NUC1xx.h"

#include "DrvSYS.h"

#include "Seven\_Segment.h"

#include "scankey.h"

#include "Driver\DrvSYS.h"

#include "Driver\DrvGPIO.h"

void Init\_LED() // Initialize GPIO pins

{

DrvGPIO\_Open(E\_GPC, 12, E\_IO\_OUTPUT); // GPC15 pin set to output mode

DrvGPIO\_SetBit(E\_GPC, 12); // Goutput Hi to turn off LED

}

void Init\_Buzzer()

{

DrvGPIO\_Open(E\_GPB, 11, E\_IO\_OUTPUT);

DrvGPIO\_SetBit(E\_GPB, 11);

}

int32\_t main (void)

{

int8\_t number;

UNLOCKREG();

DrvSYS\_Open(48000000);

LOCKREG();

Init\_LED(); // Initialize LEDs (four on-board LEDs below LCD panel)

Init\_Buzzer();

OpenKeyPad();//????

while(1)

{

number = Scankey(); // scan keypad to get a number (1~9)

if(number==3)

{

DrvGPIO\_ClrBit(E\_GPB, 11);

DrvSYS\_Delay(300000);

DrvGPIO\_SetBit(E\_GPB, 11);

DrvSYS\_Delay(300000);

}

else if(number==2)

{

DrvGPIO\_ClrBit(E\_GPC, 12); // output Low to turn on LED

DrvSYS\_Delay(300000); // delay

DrvGPIO\_SetBit(E\_GPC, 12); // output Hi to turn off LED

DrvSYS\_Delay(300000);

}

}

}

**13) Write a program to display on LCD a string “welcome to msrit”.**

#include <stdio.h>

#include "NUC1xx.h"

#include "Driver\DrvSYS.h"

#include "Driver\DrvGPIO.h"

#include "NUC1xx-LB\_002\LCD\_Driver.h" //????

int main(void)

{

UNLOCKREG();

DrvSYS\_Open(48000000); // set to 48MHz

LOCKREG();

Initial panel();

clr\_all\_panel();

print\_lcd(0, "Welcome to MSRIT");

}

**14) Write a program to use ADC to display the analog voltage connected to variable resistor on channel 7 of port A.**

#include <stdio.h> #include "NUC1xx.h"

#include "Driver\DrvSYS.h"

#include "Seven\_Segment.h"

void InitADC(void)

{

/\* Step 1. GPIO initial \*/

GPIOA->OFFD|=0x00800000; //Disable digital input path

SYS->GPAMFP.ADC7\_SS21\_AD6=1; //Set ADC function

/\* Step 2. Enable and Select ADC clock source, and then enable ADC module \*/

SYSCLK->CLKSEL1.ADC\_S = 2; //Select 22Mhz for ADC

SYSCLK->CLKDIV.ADC\_N = 1; //ADC clock source = 22Mhz/2 =11Mhz;

SYSCLK->APBCLK.ADC\_EN = 1; //Enable clock source

ADC->ADCR.ADEN = 1; //Enable ADC module

/\* Step 3. Select Operation mode \*/

ADC->ADCR.DIFFEN = 0; //single end input

ADC->ADCR.ADMD = 0; //single mode

/\* Step 4. Select ADC channel \*/

ADC->ADCHER.CHEN = 0x80;

/\* Step 5. Enable ADC interrupt \*/

ADC->ADSR.ADF =1; //clear the A/D interrupt flags for safe

ADC->ADCR.ADIE = 1;

// NVIC\_EnableIRQ(ADC\_IRQn);

/\* Step 6. Enable WDT module \*/

ADC->ADCR.ADST=1;

}

void seg\_display(int16\_t value)

{

int8\_t digit;

digit = value / 1000;

close\_seven\_segment();

show\_seven\_segment(3,digit);

DrvSYS\_Delay(5000);

value = value - digit \* 1000;

digit = value / 100;

close\_seven\_segment();

show\_seven\_segment(2,digit);

DrvSYS\_Delay(5000);

value = value - digit \* 100;

digit = value / 10;

close\_seven\_segment();

show\_seven\_segment(1,digit);

DrvSYS\_Delay(5000);

value = value - digit \* 10;

digit = value;

close\_seven\_segment();

show\_seven\_segment(0,digit);

DrvSYS\_Delay(5000);

}

int32\_t main (void)

{

int32\_t adc\_value;

UNLOCKREG();

SYSCLK->PWRCON.XTL12M\_EN = 1; //Enable 12Mhz and set HCLK->12Mhz

SYSCLK->CLKSEL0.HCLK\_S = 0;

LOCKREG();

InitADC();

while(1)

{

while(ADC->ADSR.ADF==0); // ADC Flag, wait till 1 (A/DC conversion done)

ADC->ADSR.ADF=1; // write 1 to ADF is to clear the flag

adc\_value=ADC->ADDR[7].RSLT; // input 12-bit ADC value

seg\_display(adc\_value); // display value to 7-segment display

ADC->ADCR.ADST=1; // activate next ADC sample

// 1 : conversion start

// 0 : conversion stopped, ADC enter idle state

}

}

**15) Display the analog voltage in terms of analog on the LCD of the variable resistor connected to ADC channel 7.**

#include <stdio.h> #include "NUC1xx.h"

#include "Driver\DrvSYS.h"

#include "Seven\_Segment.h"

#include "DrvADC.h"

#include "LCD\_Driver.h"

int32\_t main (void)

{ uint16\_t value;

char TEXT[16];

UNLOCKREG();

SYSCLK->PWRCON.XTL12M\_EN = 1; //Enable 12Mhz and set HCLK->12Mhz

SYSCLK->CLKSEL0.HCLK\_S = 0;

LOCKREG();

Initial\_panel(); // initialize LCD panel

clr\_all\_panel(); // clear LCD panel

print\_lcd(0,"variable resistor");

DrvADC\_Open(ADC\_SINGLE\_END,ADC\_SINGLE\_OP , 0x80,INTERNAL\_HCLK , 1);

while(1)

{

DrvADC\_StartConvert(); // start A/D conversion

while(DrvADC\_IsConversionDone()==FALSE);

value = ADC->ADDR[7].RSLT & 0xFFF;

sprintf(TEXT,"Value: %d",value); // convert ADC0 value into text

print\_lcd(1, TEXT); // output TEXT to LCD

}

}

**16) Write a program to scan the key board and display the key that is pressed on lcd or seven segment.**

#include <stdio.h>

#include "NUC1xx.h"

#include "Driver\DrvSYS.h"

#include "Driver\DrvGPIO.h"

#include "NUC1xx-LB\_002\LCD\_Driver.h"//????

int main(void)

{

int number;

char text[100];

UNLOCKREG();

DrvSYS\_Open(48000000); // set to 48MHz

LOCKREG();

Initial\_panel();

clr\_all\_panel();

while(1)

{

number=Scankey();

if(number!=0)

{

sprintf(text,"Number is : %d",number);

print\_lcd(0, text);

}

}

}

**17) Write a program to display on seven segment counts 0 to 99.**

#include <stdio.h> #include "NUC1xx.h"

#include "DrvSYS.h"

#include "Seven\_Segment.h"

// display an integer on four 7-segment LEDs

void seg\_display(int16\_t value)

{

close\_seven\_segment();

show\_seven\_segment(0,value);

DrvSYS\_Delay(50000);

close\_seven\_segment();

show\_seven\_segment(1,value);

DrvSYS\_Delay(50000);

close\_seven\_segment();

show\_seven\_segment(2,value);

DrvSYS\_Delay(50000);

close\_seven\_segment();

show\_seven\_segment(3,value);

DrvSYS\_Delay(50000);

}

int32\_t main (void)

{

uint32\_t i=0;

UNLOCKREG(); // Unlock System Registers

DrvSYS\_Open(48000000); // set CPU clock to 25~50MHz

LOCKREG(); // Lock System Registers

while(1)

{

DrvSYS\_Delay(100000);

seg\_display(i);

i++;

if (i>9) i=0;

}

}

**18) Write a program to toggle LEDs R G B one after the other in sequence**

#include <stdio.h>

#include "NUC1xx.h"

#include "Driver\DrvGPIO.h"

#include "Driver\DrvUART.h"

#include "Driver\DrvSYS.h"

// Initial GPIO pins (GPA 12,13,14) to Output mode

void Init\_LED()

{

// initialize GPIO pins

DrvGPIO\_Open(E\_GPA, 12, E\_IO\_OUTPUT); // GPA12 pin set to output mode

DrvGPIO\_Open(E\_GPA, 13, E\_IO\_OUTPUT); // GPA13 pin set to output mode

DrvGPIO\_Open(E\_GPA, 14, E\_IO\_OUTPUT); // GPA14 pin set to output mode

// set GPIO pins output Hi to disable LEDs

DrvGPIO\_SetBit(E\_GPA, 12); // GPA12 pin output Hi to turn off Blue LED

DrvGPIO\_SetBit(E\_GPA, 13); // GPA13 pin output Hi to turn off Green LED

DrvGPIO\_SetBit(E\_GPA, 14); // GPA14 pin output Hi to turn off Red LED

}

int main (void)

{

UNLOCKREG ( ); // unlock register for programming

DrvSYS\_Open(48000000); // set System Clock to run at 48MHz (PLL with 12MHz crystal input)

LOCKREG ( ); // lock register from programming

Init\_LED();

while (1)

{

// GPA12 = Blue, 0: on, 1: off

// GPA13 = Green, 0: on, 1: off

// GPA14 = Red, 0: on, 1: off

// set RGBled to Blue

DrvGPIO\_ClrBit(E\_GPA,12); // GPA12 = Blue, 0 : on, 1 : off

DrvGPIO\_SetBit(E\_GPA,13);

DrvGPIO\_SetBit(E\_GPA,14);

DrvSYS\_Delay(1000000);

// set RGBled to Green

DrvGPIO\_SetBit(E\_GPA,12);

DrvGPIO\_ClrBit(E\_GPA,13); // GPA13 = Green, 0 : on, 1 : off

DrvGPIO\_SetBit(E\_GPA,14);

DrvSYS\_Delay(1000000);

// set RGBled to Red

DrvGPIO\_SetBit(E\_GPA,12);

DrvGPIO\_SetBit(E\_GPA,13);

DrvGPIO\_ClrBit(E\_GPA,14); // GPA14 = Red, 0 : on, 1 : off

DrvSYS\_Delay(1000000);

// set RGBled to off

DrvGPIO\_SetBit(E\_GPA,12); // GPA12 = Blue, 0 : on, 1 : off

DrvGPIO\_SetBit(E\_GPA,13); // GPA13 = Green, 0 : on, 1 : off

DrvGPIO\_SetBit(E\_GPA,14); // GPA14 = Red, 0 : on, 1 : off

DrvSYS\_Delay(1000000);

}

}